On Numerical Error Estimation for the Finite-Volume Method with an Application to Computational Fluid Dynamics

William C. Tyson

Dissertation submitted to the Faculty of the
Virginia Polytechnic Institute and State University
in partial fulfillment of the requirements for the degree of

Doctor of Philosophy
in
Aerospace Engineering

Christopher J. Roy, Chair
Jeffrey T. Borggaard
Heng Xiao
Bhuvana Srinivasan

October 25, 2018
Blacksburg, Virginia

Keywords: Computational Fluid Dynamics, Discretization Error Estimation, Truncation Error Estimation, Adjoint Methods, Numerical Error Estimation

On Numerical Error Estimation for the Finite-Volume Method with an Application to Computational Fluid Dynamics

William C. Tyson

(ABSTRACT)

Computational fluid dynamics (CFD) simulations can provide tremendous insight into complex physical processes and are often faster and more cost-effective to execute than experiments. However, each CFD result inherently contains numerical errors that can significantly degrade the accuracy of a simulation. Discretization error is typically the largest contributor to the overall numerical error in a given simulation. Discretization error can be very difficult to estimate since the generation, transport, and diffusion of these errors is a highly nonlinear function of the computational grid and discretization scheme. As CFD is increasingly used in engineering design and analysis, it is imperative that CFD practitioners be able to accurately quantify discretization errors to minimize risk and improve the performance of engineering systems.

In this work, improvements are made to the accuracy and efficiency of existing error estimation techniques. Discretization error is estimated by deriving and solving an error transport equation (ETE) for the local discretization error everywhere in the computational domain. Truncation error is shown to act as the local source for discretization error in numerical solutions. An equivalence between adjoint methods and ETE methods for functional error estimation is presented. This adjoint/ETE equivalence is exploited to efficiently obtain error estimates for multiple output functionals and to extend the higher-order properties of adjoint methods to ETE methods. Higher-order discretization error estimates are obtained when truncation error estimates are sufficiently accurate. Truncation error estimates are demonstrated to deteriorate on grids with a non-smooth variation in grid metrics (e.g., unstructured grids) regardless of how smooth the underlying exact solution may be. The loss of accuracy is shown to stem from noise in the discrete solution on the order of discretization error. When using conventional least-squares reconstruction techniques, this noise is exactly captured and introduces a lower-order error into the truncation error estimate. A novel reconstruction method based on polyharmonic smoothing splines is developed to smoothly reconstruct the discrete solution and improve the accuracy of error estimates. Furthermore, a method for iteratively improving discretization error estimates is devised. Efficiency and robustness considerations are discussed. Results are presented for several inviscid and viscous flow problems. To facilitate the study of discretization error estimation, a new, higher-order finite-volume solver is developed. A detailed description of the code base is provided along with a discussion of best practices for CFD code design.

On Numerical Error Estimation for the Finite-Volume Method with an Application to Computational Fluid Dynamics

William C. Tyson

(GENERAL AUDIENCE ABSTRACT)

Computational fluid dynamics (CFD) is a branch of computational physics at the intersection of fluid mechanics and scientific computing in which the governing equations of fluid motion, such as the Euler and Navier-Stokes equations, are solved numerically on a computer. CFD is utilized in numerous fields including biomedical engineering, meteorology, oceanography, and aerospace engineering. CFD simulations can provide tremendous insight into physical processes and are often preferred over experiments because they can be performed more quickly, are typically more cost-effective, and can provide data in regions where it may be difficult to measure. While CFD can be an extremely powerful tool, CFD simulations are inherently subject to numerical errors. These errors, which are generated when the governing equations of fluid motion are solved on a computer, can have a significant impact on the accuracy of a CFD solution. If numerical errors are not accurately quantified, ill-informed decision-making can lead to poor system performance, increased risk of injury, or even system failure.

In this work, research efforts are focused on numerical error estimation for the finite-volume method, arguably the most widely used numerical algorithm for solving CFD problems. The error estimation techniques provided herein target discretization error, the largest contributor to the overall numerical error in a given simulation. Discretization error can be very difficult to estimate since these errors are generated, convected, and diffused by the same physical processes embedded in the governing equations. In this work, improvements are made to the accuracy and efficiency of existing discretization error estimation techniques. Results are presented for several inviscid and viscous flow problems. To facilitate the study of these error estimators, a new, higher-order finite-volume solver is developed. A detailed description of the code base is provided along with a discussion of best practices for CFD code design.

On Numerical Error Estimation for the Finite-Volume Method with an Application to Computational Fluid Dynamics

William C. Tyson

(GRANT INFORMATION)

This work was partially funded by the Air Force Office of Scientific Research (AFOSR) Computational Mathematics Program managed by Dr. Fariba Fahroo under grant FA9550-12-1-0173.

Additional funding was provided by the NASA Graduate Aeronautics Scholarship through the Aeronautics Scholarship and Advanced STEM Training and Research Fellowship (AS&ASTAR) program.

To my wife, Abigail,

for her endless love and support

Acknowledgments

God the Father, God the Son, and God the Holy Spirit, I humbly thank you for being the quiet, guiding hand in my life, for steering and sometimes pulling me in the direction you have planned. I ask that you forgive me for trying to be in control of my own life. How can one claim to love and serve you if he does not completely surrender his life? I also thank you for forming me with a thirst for knowledge, learning, and truth. Uncovering your beauty and majesty in science, mathematics, and the natural world bring me great joy. Please help me to use the gifts you have granted me to glorify you throughout my life and my career.

To my advisor, Dr. Chris Roy, I am very grateful for your guidance and support throughout my graduate work. Coming from a small town on the Eastern Shore of Virginia, the idea of obtaining an advanced degree seemed foreign and unattainable. Thank you for encouraging me to pursue a Ph.D. and reassuring me that I was smart enough and capable enough to be in a Ph.D. program.

To my committee members, Dr. Jeff Borggaard, Dr. Heng Xiao, and Dr. Bhuvana Srinivasan, thank you for serving on my committee and providing helpful insight throughout my Ph.D. program.

To the staff of the AOE department, especially Kelsey Wall, Jonathan Spence, and Steve Edwards, thank you so much for your administrative and I.T. help. I am very grateful for your willingness to provide assistance whenever I have needed it.

To my high school math teacher, Mrs. Judith Wood, thank you for your infectious love of mathematics and for showing me how beautiful and fun math can be. More importantly, thank you for the sacrifices that you made to teach me. Only time and experience can mold an ungrateful high school student into an appreciative adult.

To the researchers of the Computational Aerosciences branch at NASA Langley, especially Dr. Michael Park and Dr. Joseph Derlaga, thank you for your financial and technical support throughout the latter half of my Ph.D. program. It has been an honor to work with and learn from such a renowned group of researchers in the field of aerospace engineering.

To my graduate school friends and labmates, especially Chip Jackson and Tom Battista, thank you for all of the entertaining conversations and game nights. Thank you for being a sounding board for ideas and providing me an outlet for venting frustration. Your friendship

means a great deal to me. If you ever need anything, you know where to find me (probably at the nearest Moe's).

To the St. Mary's community, especially Bob and Glenda Canfield, Dennis Brisendine, and all of the members of the St. Mary's young adults group, past and present, thank you for welcoming my family and me into your faith community with open arms. Thank you for the love you have shown us and for nurturing our faith. Because of you, St. Mary's has felt like our home away from home. We will truly miss our church family.

To my uncle Philip and aunt Eileen, thank you so much for opening up your home to me while I worked at NASA Langley. It was very generous and gracious of you to allow me to stay with you, especially during such a transient time. I greatly enjoyed our conversations and getting to see loving family members at the end of the day.

To my sisters-in-law, Catie, Elizabeth, Vivian, Sophia, and Madeline, thank you for accepting me as the second boy in the Zadnik clan and allowing me to be a big brother to you. Your joy and laughter have taught me to never take myself too seriously. You have also shown me that spontaneously breaking into song is a perfectly acceptable behavior. When the Zadnik girls get together, there is never a dull moment. You all are crazy, and I love you very much.

To my brother and sister, Matthew and Holly, thank you for putting up with me when I "nerd out" about aerospace engineering. I know it can be painful to sit through my long rants about math and why the S.I. system is clearly superior to the English system of units. Thank you for your encouragement and support throughout our lives. Although I may not always show it, I love you both very much.

To Mom and Dad Zadnik, thank you for allowing me to marry the love of my life and graciously welcoming me as your son. I have truly gained a mother and a father. Thank you for supporting Abi and me throughout my program and encouraging me every step of the way. Thank you for all of the sacrifices that you make for us and for our family. We love you so much.

To my Mom and Dad, I cannot express how grateful I am for your love and support. Thank you for all of the sacrifices you made so that we could get a good education. Thank you for helping me find a home at Virginia Tech. Thank you for the many trips you made out to Blacksburg to help me move or just to visit. Thank you for reassuring me that I was smart enough to be in a Ph.D. program. Thank you for teaching me the value of education and hard work. Mom and Dad, I love you. And Mom, I promise I can get a job with this stuff.

To my son, Oliver, and daughter, Joanna, your daddy loves you more than you know. It brings me so much joy every single day to see your smiling faces and watch you grow. Becoming your father has undoubtedly been my greatest and proudest achievement in life.

To my wife, Abigail, you are my best friend, teammate, and the love of my life. You are an amazing wife and mother. Your unconditional love and support is what has kept me sane throughout this process. Thank you for calming me down during my relatively frequent

freak-out moments and reassuring me that I could finish this. Thank you for going on this adventure with me. I know that it has not always been easy. However, I could not have done this without you. I love you so much, and I cannot wait to see what our next adventure may be.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

The Navier-Stokes equations are a coupled set of nonlinear, partial differential equations (PDEs) governing the motion of fluids. These equations, which are a statement of conservation of mass, momentum, and energy, can describe a wide variety of physical phenomenon such as the flow of liquids through pipes, aerodynamic forces on bodies, weather patterns, and turbulence. Due to their complexity, the Navier-Stokes equations can only be solved in a closed form for certain classes of problems where assumptions (e.g, inviscid, irrotational, etc.) can be made to drastically simplify the equation set. For general problems, the Navier-Stokes equations cannot, as of yet, be solved analytically and must instead be solved numerically.

The numerical solution of the Navier-Stokes equations, or a simplified subset (e.g., the Euler equations), is often referred to as computational fluid dynamics (CFD). CFD is situated at the unique intersection of mathematics, computer science, and engineering. In the last 50 years, the use of CFD has grown exponentially due to advances in numerical algorithms and the development of high-performance computing architectures. CFD simulations are sometimes favored over experiments because they are much faster and cheaper to perform and can provide data in regions where measurements may be difficult to obtain. Today, CFD simulations are used to study the human vascular system [1], forecast the weather [2], model tsunamis [3], and even predict flow fields around full aircraft configurations [4].

For most applications, a CFD simulation generally proceeds as outlined in Figure 1.1. First, the domain of interest (i.e., the problem geometry) as well as the governing physical model (i.e., the Euler or Navier-Stokes equations) are defined. The geometry of the problem is typically prescribed with a computer-aided design (CAD) package in which a continuous representation of the geometry is constructed from splines and predefined geometric entities. Next, both the continuous geometry and continuous governing equations are converted into consistent discrete forms through a process called *discretization*. This discretization step is necessary since computers largely operate using discrete mathematics rather than continuous mathematics. The discrete geometry, often referred to as a *mesh* or *grid*, utilizes a finite

number of nodes to describe surface geometry and the surrounding domain volume. With regard to the governing equations, there are three primary discretization methods in use today which constitute the majority of CFD simulations: finite-difference methods (FDM) [5], finite-element methods (FEM) [6], and finite-volume methods (FVM) [7]. While fundamentally different in approach, each of these discretization methods essentially transforms the continuous equations into a set of nonlinear, algebraic equations. Upon discretization, the discrete geometry and equation set are passed to a solver which computes a numerical solution. The CFD simulation concludes with post-processing of the simulation result via visualization and data-reduction software.

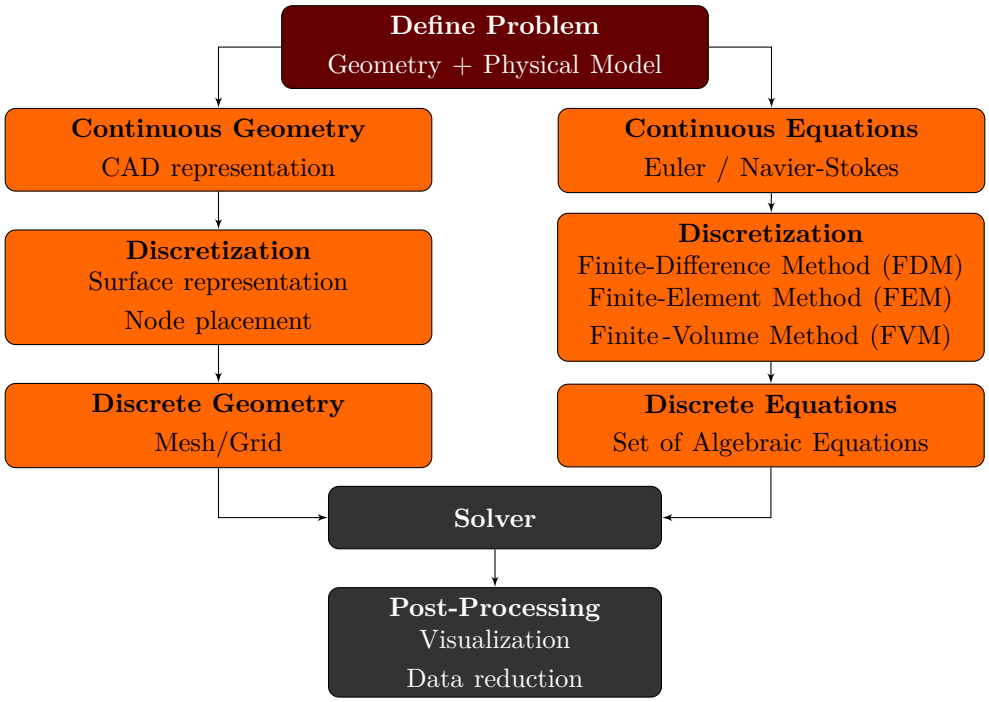


Figure 1.1: Outline of a standard CFD simulation.

CFD is a powerful tool that can lend tremendous insight into complex physical processes. However, CFD practitioners must be aware that errors are inherently present in each simulation result. These errors typically stem from modeling assumptions, numerical approximations, and finite-precision arithmetic. Simulation errors can have a significant impact on the accuracy of a given CFD result. As CFD is increasingly used in both engineering design and analysis, it is imperative that practitioners be able to accurately assess simulation errors, especially for applications involving high-consequence systems [8]. In recent years, there has been a serious push within the CFD community to incorporate rigorous verification and validation practices into CFD workflows [9, 10, 11, 12, 13, 14]. As part of this effort, error estimators have been developed in an attempt to quantify simulation errors and increase confidence in CFD predictions.

The research efforts presented in this dissertation focus specifically on error estimators for the finite-volume method. At present, error estimators for finite-volume methods are somewhat lacking in their accuracy, efficiency, and robustness despite finite-volume methods comprising approximately 80% - 90% of commercial CFD codes. In subsequent sections, simulation errors are classified in more detail and a thorough review of finite-volume error estimators is presented.

1.1 Classification of Simulation Errors

The errors in any given CFD simulation can be classified as follows:

1. **Modeling Error**: An error arising from an incomplete description of the underlying physics. For example, when simulating turbulence, the following term arises when the Navier-Stokes equations are averaged over a finite time interval,

$$\tau_{ij} = -\rho\overline{u_i' u_j'} \tag{1.1}$$

   where ρ is the fluid density, u_i' and u_j' are instantaneous fluctuating components of velocity, and τ_{ij} is referred to as the Reynolds stress [15]. The Reynolds stress represents a correlation between velocity components that arises from the nonlinear convective terms in the Navier-Stokes equations. Since the exact form of the Reynolds stress cannot be determined, τ_{ij} must instead be modeled in order to close the system of equations. By prescribing the form of the Reynolds stress, modeling errors are introduced. Modeling errors can only be reduced with a more accurate representation of the true physics or through model calibration [16].

2. **Round-off Error**: A numerical error arising from finite-precision arithmetic, where "precision" refers to the number of digits that a computer uses to represent a given number. As an example, consider the following mathematical operations. With continuous mathematics,

$$3.0 \times \frac{1.0}{3.0} = 1.0 \ . \tag{1.2}$$

   However, when performing *single precision* calculations, the same operation yields

$$3.0 \times \frac{1.0}{3.0} = 0.999999 \ . \tag{1.3}$$

   A round-off error of 0.000001 is generated because the fraction 1.0/3.0 can only be represented with approximately 6 digits of precision. Round-off errors can accumulate as the number of consecutive arithmetical operations increases. The effect of round-off errors can be minimized by increasing the numerical precision used in arithmetical calculations.

3. **Iterative Error**: A numerical error arising from incomplete convergence of iterative procedures. For instance, consider the following function:

$$f(x) = (x-3)\Big(x^2+5\Big) \; . \tag{1.4}$$

If one desires to find the roots of this function (i.e., where $f(x) = 0$), an iterative, root finding method can be employed, such as the secant method or Newton's method [17]. The root of this function exists at $x = 3.0$. If, however, the rooting finding method is prematurely terminated, the algorithm may locate the root instead at $x = 2.995$. In this case, an iterative error of 0.005 is introduced. As illustrated in this example, the effect of iterative errors can be minimized by fully converging iterative procedures to machine precision.

4. **Discretization Error (DE)**: A numerical error arising from the process of discretization. To provide some clarity, consider the following equation

$$\mathbf{N}(\tilde{\mathbf{u}}) = \mathbf{0} \tag{1.5}$$

where $\mathbf{N}(\cdot)$ represents a continuous, nonlinear PDE and $\tilde{\mathbf{u}}$ is defined as the exact continuous solution. After discretization, $\mathbf{N}(\cdot)$ is represented by the following

$$\mathbf{N}_h^p(\mathbf{u}_h) = \mathbf{0} \tag{1.6}$$

where $\mathbf{N}_h^p(\cdot)$ are the discrete equations and $\mathbf{u}_h$ is defined as the exact discrete solution. The term p corresponds to the *formal order of accuracy* of the discretization [18]. Given Eq. 1.5 and Eq. 1.6, discretization error, ε_h, is formally defined as

$$\varepsilon_h = \mathbf{u}_h - I^h\tilde{\mathbf{u}} \tag{1.7}$$

where I^h is an operator which restricts the exact continuous solution, $\tilde{\mathbf{u}}$, to the grid. Since $\tilde{\mathbf{u}}$ is rarely known, discretization errors cannot be directly quantified and must instead by estimated. Discretization errors can be reduced through grid refinement (i.e., increasing the number of nodes in the grid) or increasing the formal order of accuracy of the discretization.

5. **Truncation Error (TE)**: The functional difference between a given continuous equation and its discretized form. More simply, truncation error can be defined as follows

$$\tau_h^p(\mathbf{u}) = \mathbf{N}_h^p\Big(I^h\mathbf{u}\Big) - \mathbf{N}(\mathbf{u}) \tag{1.8}$$

where $\mathbf{u}$ is a general, continuous function. For model problems, truncation error can be shown to be a function of grid metrics (e.g., cell size, mesh smoothness, etc.) and solution derivatives [19]. Truncation error can be thought of as higher-order terms which are neglected during the process of discretization. For some discretizations, the exact form of the truncation error can be analytically determined. However, for most discretizations, especially those on general topology grids, truncation error cannot be explicitly written and must instead be estimated. Truncation errors indirectly affect the accuracy of a CFD simulation through the number of terms retained in the discretized equations, $\mathbf{N}_h^p(\cdot)$.

1.2 Objectives

In this dissertation, attention is focused on numerical error estimation, in particular DE estimation. Despite being classified as numerical errors, round-off error and iterative error are often negligible if enough precision is used and iterative procedures are fully converged. DE, on the other hand, is typically the largest contributor to the numerical error in a given simulation. DE can be difficult to quantify as the generation, transport, and dissipation of these errors is a highly nonlinear function of the grid and the discrete solution. TE, as demonstrated in subsequent chapters, acts as the local source of DE in numerical solutions [18]. Consequently, the accuracy of many DE estimators is directly impacted by the accuracy of TE estimates.

In this work, the author seeks to improve the accuracy of DE estimators for the finite-volume method. Due to the relationship between TE and DE, significant effort is directed toward improving the accuracy of finite-volume TE estimation techniques. In an attempt to make numerical error estimation more adoptable by the broader CFD community, consideration is also given to holistically improving the computational efficiency of DE estimation methods.

1.3 Literature Review

1.3.1 Discretization Error Estimation

Within the context of finite-volume methods, DE estimators are typically categorized as *multiple-grid* error estimators or *single-grid* error estimators. This terminology simply refers the number of computational grids / discrete solutions involved in forming the error estimate. Arguably the first multiple-grid error estimator is a method known as *Richardson extrapolation* [20]. Using the discrete solutions from a series of systematically refined grid levels, Richardson extrapolation forms a DE estimate by differencing the solution on the finest grid level with an extrapolated guess of the exact solution. Richardson extrapolation is non-intrusive to existing CFD codes and can easily be performed as a post-processing step. However, in order to obtain an accurate DE estimate, the discrete solution must be smooth and all grid levels must be in the *asymptotic range* (i.e., the regime where grid spacing is sufficiently fine such that errors converge at the rate of the lowest order term). In theory, three grid levels are necessary to perform Richardson extrapolation. In practice, it often takes five to six grid levels to firmly establish that all required grids are in the asymptotic range. For large engineering problems (e.g., problems requiring millions of grid nodes), especially those with complex flow fields [21, 22], Richardson extrapolation can quickly become impractical as achieving the asymptotic range may be prohibitively expensive.

A common single-grid alternative to Richardson extrapolation is a method known as *defect correction*. Defect correction, which was originally proposed by Fox [23] and further

developed by Pereyra [24, 25] and Stetter [26], solves the *primal problem* (i.e., the discrete governing equations), inserts an estimate of the TE as a source term, and then re-solves the primal problem. By including the TE as a source term, the discrete solution is driven toward a potentially more accurate solution. In the literal sense, this process corrects a defect in the original solution. A DE estimate can be formed from the difference between the original solution and the corrected solution. Defect correction is much cheaper than Richardson extrapolation since it only requires a discrete solution on one grid level. However, a significant computational cost is still incurred from re-solving the primal problem, albeit with a better initial guess.

Babuška et al. [27, 28] and Gu and Shih [29] identified that DE can be generated in one location and transported and diffused via physical processes to other locations in the computational domain. This realization motivated the development error estimators based upon the solution of a transport equation. These types of error estimators are referred to as *error transport equations* (ETE). ETE methods typically vary in three respects: (1) the derivation of the ETE (e.g, continuous or discrete), (2) simplifications made to the ETE (e.g., linearized or fully nonlinear), and (3) the method used to estimate the TE (n.b., often referred to as the *residual* in early ETE literature).

Much of the initial ETE research focused on the linearized form of the discrete ETE. For instance, see the works of Zhang et al. [30, 31], Qin et al. [32, 33, 34, 35], Celik and Hu [36], and Shih and Williams [37]. In comparison to the heuristic gradient-based error estimators available at the time [38, 39, 40, 41], linearized ETE error estimates proved to be a significant improvement. However, the accuracy of error estimates largely suffered due to inaccurate estimates of TE. Furthermore, linearized ETE error estimates were only obtained for model problems, such as the 1D convection-diffusion equation and 1D Burgers' equation. Cavallo et al. [42, 43, 44] were arguably the first researchers to demonstrate the utility of the linearized ETE for practical aerospace applications. Conservative ETE error estimates were obtained for subsonic, turbulent flow over an airfoil as well as subsonic flow over a UCAV 1303 configuration.

In contrast to the discrete ETE, Hay and Visonneau [45] and Banks et al. [46, 47] adopted the formalism of the continuous ETE. Hay and Visonneau were able to obtain asymptotically correct, higher-order estimates of DE for several hyperbolic flow problems. As can be done with any error estimate, Hay and Visonneau also showed that the ETE error estimate could be used to increase the accuracy of the entire solution. Banks et al. demonstrated that for problems with strong discontinuities (e.g., shock waves) the linearized ETE can lead to unbounded growth in the error estimate in the immediate vicinity of the discontinuity. However, with the fully nonlinear form of the ETE, Banks et al. were able to obtain bounded error estimates for all test cases.

More recently, Derlaga [48] compared the accuracy of linearized ETE error estimates when using different TE estimators. ETE error estimates were also compared to DE estimates obtained with defect correction. Yan and Ollivier-Gooch [49, 50, 51] extended ETE methods

to general unstructured grids and were able to obtain higher-order estimates of DE, albeit by solving the ETE to higher-order. Yan and Ollivier-Gooch [52] also employed the nonlinear ETE to estimate DE for unsteady flow problems. Furthermore, Derlaga et al. [53] demonstrated the utility of the ETE for estimating errors in practical aerospace applications, such as sonic-boom and drag prediction.

For a more detailed review of DE estimators, refer to the review article of Roy [54].

1.3.2 Output-based Error Estimation

Often in engineering applications, practitioners are primarily interested in obtaining accurate predictions of output functionals. These output functionals are typically integrated quantities of the discrete solution (e.g., lift and drag for an aerodynamics application) but may also be quantities at a specific location in space. Concerning error estimators for output functionals, research efforts in the last two decades have largely focused on the development of *adjoint methods*. With an adjoint method, functional error estimates may be obtained for any output functional of interest by taking the inner product of the adjoint solution with the local TE. Under certain conditions, error estimates can even be applied as a correction to increase the order of accuracy of the computed functional [55]. Adjoint methods have also become quite popular for driving an adaptation procedure aimed at reducing the error in a given functional [56] or multiple functionals [57].

Similar to the ETE, adjoint methods can vary in two ways: (1) the derivation of the adjoint problem (e.g, continuous or discrete) and (2) the method used to estimate the TE (n.b., often referred to as the *primal residual* in adjoint literature). With regard to the continuous adjoint method, the governing equations are first linearized followed by the formulation and discretization of the adjoint problem. With the continuous adjoint approach, the proper implementation of adjoint spatial operators and boundary conditions can be easily verified. However, deriving the continuous adjoint equations for different equation sets can be a difficult and laborious task. On the other hand, with the discrete adjoint method, the governing equations are directly discretized and linearized. The discrete adjoint problem is then formulated from this linearized, discrete equation set. The discrete adjoint method is typically easier to implement than the continuous adjoint method since most CFD solvers already contain a linearization of the primal problem. But, it can be more difficult to ensure a *dual-consistent* discretization (i.e., one which is equivalent to a discretization of the continuous adjoint equations). While formulated slightly differently, the continuous adjoint and discrete adjoint methods are asymptotically equivalent if a dual-consistent discretization is used.

In the context of finite-volume methods, Giles and Pierce [58, 59] are among the first researchers to use adjoint methods for functional error estimation. Their work has provided much of the theoretical background for both continuous and discrete adjoint methods, including but not limited to the principle of duality [60] and the proper implementation of

strong boundary conditions [61]. In a subsequent work, Pierce and Giles [62] even coupled defect correction with a continuous adjoint method to increase the observed order of accuracy of output functionals by approximately 3.5x.

Concurrently with the efforts of Giles and Pierce, Venditti and Darmofal [63, 64, 65, 56] pioneered much of the development for the discrete adjoint method. In their work, coarse-grid functional error estimates were applied as corrections to drive coarse-grid functionals toward their corresponding values on a finer mesh. Additionally, the discrete adjoint solution was formulated into an adaptation indicator and used to drive an adaptation procedure for 2D inviscid and viscous flow problems. Park [66] extended the work of Venditti and Darmofal to 3D applications by performing adjoint-based error estimation and adaptation on an ONERA M6 wing and a transport aircraft configuration. Park [67] further extended the discrete adjoint method to cut cell finite-volume discretizations. Similarly, Nemec et al. [68, 69, 70] extended the discrete adjoint method to Cartesian mesh finite-volume discretizations.

For a more detailed review of adjoint-based error estimation, refer to the review articles of Giles and Süli [71] and Fidkowski and Darmofal [72].

1.3.3 Truncation Error Estimation

For both adjoint-based error estimators and DE estimators, the accuracy of the TE estimate is generally what determines the accuracy of the resulting functional and DE estimates. Particularly in ETE literature, determining the proper method of estimating the TE has been the primary focus of research. TE estimators for finite-volume methods can be classified as follows: (1) continuous residual methods, (2) higher-order discrete residual methods, (3) embedded grid methods, or (4) modified equation methods.

In the continuous residual (CR) approach, the discrete solution is reconstructed to a continuous function, and the governing equations (either weak form or strong form) are then operated onto the reconstructed solution. This TE estimator has been successfully employed in the works of Giles et al. [55, 58, 59, 60, 61, 62, 71]. More recently, Celik and Parsons [73] utilized a strong form CR TE estimator to solve the ETE for several incompressible flow problems. CR TE estimators have a rigorous mathematical foundation and can provide asymptotically correct TE estimates. However, CR TE methods can be cumbersome to implement. For instance, the reconstructed solution must satisfy a certain degree of continuity as set by the governing equations, typically C^0 or C^1 for most PDEs. This continuity requirement necessitates the implementation of a global reconstruction, which for large problems is often too expensive. Additionally, the continuous boundary conditions must be enforced on the reconstructed solution otherwise boundary terms must be explicitly accounted for to obtain an accurate TE estimate [71, 74, 75]. For complex problems, enforcement of the continuous boundary conditions may be challenging.

The higher-order discrete residual (HODR) method uses a higher-order discretization of the

governing equations to formulate a TE estimate. HODR methods are typically referred to as p-TE estimators. This approach was adopted by Yan and Ollivier-Gooch [49, 50, 51, 52] for ETE error estimation on unstructured grids as well as by Sharbatdar and Ollivier-Gooch [76, 77, 78] for adjoint-based error estimation. Ollivier-Gooch and Roy [79] utilized a HODR TE estimate in a mesh movement algorithm aimed at minimizing the TE. In comparison to CR TE estimators, HODR estimators relax the assumptions on continuity for the reconstruction and do not require enforcement of the continuous boundary conditions. However, the resulting TE estimates may not be as accurate as a CR TE estimate. HODR methods are relatively straightforward to implement in existing finite-volume codes as long as the discretization is not explicitly limited to a low formal order of accuracy (i.e., $p = 1$ or $p = 2$).

Embedded grid methods have been primarily used by the adjoint community to formulate functional error estimates. With this approach, a TE estimate is obtained by restricting the discrete solution to an "embedded grid" (i.e., either a coarse grid or fine grid generated from the current mesh) and performing a discrete residual evaluation. Embedded grid estimators are often referred to as h-TE estimators. Venditti and Darmofal [63, 64, 65, 56] and Nemec et al. [68, 69, 70] employed fine-grid TE estimators when computing functional error estimates for 2D and 3D aerodynamic applications. Fraysee et al. [80] adopted a coarse-grid TE estimator and also incorporated effects of iterative error on the accuracy of the TE estimate. To increase the accuracy of the TE estimate, Phillips [81] added a correction factor to account for the additional TE associated with the embedded grid. Embedded grid methods are typically less burdensome to implement in comparison to CR and HODR TE estimators, but can be computationally expensive if the embedded grid is not formulated in an efficient manner. Embedded grid methods also make an implicit assumption that the TE converges at the formal order of accuracy, but, in general, this may not be the case [82].

Modified equation TE estimators are formulated by analytically determining the exact form of the TE and using an approximation of the leading order terms as a TE estimate. For finite-volume schemes in higher dimensions (i.e., $d > 1$), obtaining the exact form of the TE is a painstaking, error-prone process and often the leading order terms of the TE are approximated using a 1D analogy. The modified equation approach was adopted in early ETE literature by Zhang et al. [30, 31], Qin et al. [32, 33, 34, 35], Cavallo et al. [42, 43, 44], and Shih and Williams [37]. Accurate TE estimates were obtained for uniform or smoothly varying grids. However, on more general grid topologies, the accuracy of TE estimates ultimately suffered.

1.4 Contributions

The contributions of the author (William C. Tyson) to the scientific community are outlined below for each chapter of this dissertation.

Chapter 2: *Improved Functional-Based Error Estimation and Adaptation using Error Transport Equations and Krylov Subspace Methods*

- Analytically demonstrates the equivalency between discrete adjoint methods and discrete ETE methods for functional error estimation.
- Improves the efficiency of functional error estimation when multiple functional error estimates are required.
- Demonstrates that approximate adjoint solutions can be used to drive an output-based adaptation algorithm in a manner that is just as effective as using exact adjoint solutions but much more efficient.

Chapter 3: *A Higher-Order Error Estimation Framework for Finite-Volume CFD*

- Extends the higher-order properties of adjoint methods to linearized discrete ETE methods using adjoint/error transport equivalency.
- Improves higher-order discrete residual TE estimates by incorporating boundary conditions into the reconstruction of the discrete solution.
- Demonstrates that the loss of higher-order accurate DE estimates on unstructured grids is a direct result of an inaccurate TE estimate.

Chapter 4: *A Novel Reconstruction Technique for Finite-Volume Truncation Error Estimation*

- Demonstrates for grids with a non-smooth variation in grid metrics (e.g., unstructured grids) that noise in the discrete solution on the order of DE degrades the accuracy of TE estimates when conventional least-squares reconstruction methods are employed.
- Develops a new reconstruction technique for finite-volume TE estimation based on polyharmonic smoothing splines.
- Improves the accuracy of TE and DE estimates for grids with a non-smooth variation in grid metrics.

Chapter 5: *Relinearization of the Error Transport Equations for Arbitrarily High-Order Error Estimates*

- Develops an algorithm for iteratively improving DE estimates.

Chapter 6: *BlueRidge: A Higher-Order Finite-Volume Solver*

- Develops a higher-order finite-volume solver with multi-physics and error estimation capabilities.
- Enables future research efforts in the areas of higher-order finite-volume methods, numerical error estimation, and algorithm development.

1.5 Outline

This dissertation is written in a manuscript format. As such, each chapter has multiple authors and is a distinct and complete document. Authorship attribution is outlined in the following section. The first chapter provides the reader with background information regarding numerical simulation, error estimation, and literature review of previous work. The second chapter analytically demonstrates an equivalency between adjoint methods and error transport methods. This equivalency is utilized to improve the efficiency of functional error estimation and adaptation techniques. The third chapter extends the work of the second chapter to improve the accuracy of local DE estimates. Higher-order discrete residual TE estimates are improved by incorporating boundary conditions into the reconstruction of the discrete solution. The fourth chapter presents a new reconstruction method for finite-volume TE estimation on "unstructured" grids (i.e., grids with a non-smooth variation in grid metrics). The resulting error estimates are significantly more accurate than those obtained with a traditional reconstruction method. The fifth chapter outlines a new method for iteratively improving the accuracy of DE estimates. The sixth chapter outlines the development of a new higher-order finite-volume solver, BlueRidge. The discretization implemented in the code base is outlined in detail, and a discussion of best practices for code development is provided. The solver is utilized in the sixth chapter for DE estimation in higher dimensions. The seventh chapter provides a broad discussion of the research presented in this dissertation, makes concluding remarks, and offers suggestions for future areas of research.

1.6 Attribution

Since this dissertation is written in a manuscript format, each chapter has multiple authors. William C. Tyson, the first author, served as the primary contributor for each chapter. For a complete description of the contributions from each author, please refer to the following:

Chapter 2: *Improved Functional-Based Error Estimation and Adaptation using Error Transport Equations and Krylov Subspace Methods*

- William C. Tyson (First Author): The first author served as the main contributor and primary author of the work. The first author implemented the primal and dual solvers

for Burgers' equation and the quasi-1D nozzle problem. Additionally, the first author performed all numerical studies presented.

- Katarzyna Świrydowicz (Second Author): The second author assisted in drafting the manuscript and provided codes for estimating the bilinear form and for computing the Sparse Approximate Inverse (SAI) preconditioner.
- Joseph M. Derlaga (Third Author): The third author provided valuable guidance and insight during the course of the study. The third author also provided helpful comments on the manuscript.
- Christopher J. Roy (Fourth Author): The fourth author provided valuable guidance and insight during the course of the study. The fourth author also provided helpful comments on the manuscript.
- Eric de Sturler (Fifth Author): The fifth author provided valuable guidance and insight during the course of the study. The fifth author also provided helpful comments on the manuscript.

Chapter 3: *A Higher-Order Error Estimation Framework for Finite-Volume CFD*

- William C. Tyson (First Author): The first author served as the main contributor and primary author of the work. The first author implemented the primal and error transport solvers for the quasi-1D nozzle problem. Additionally, the first author implemented all routines for TE estimation and performed all numerical studies.
- Christopher J. Roy (Second Author): The second author provided valuable guidance and insight during the course of the study. The second author also provided helpful comments on the manuscript.

Chapter 4: *A Novel Reconstruction Technique for Finite-Volume Truncation Error Estimation*

- William C. Tyson (First Author): The first author served as the main contributor and primary author of the work. The first author extended the smoothing spline reconstruction algorithm to finite-volume TE estimation and performed all comparison studies with the k-exact reconstruction. Additionally, the first author provided tools for assessing the quality of TE estimates.
- Christopher J. Roy (Second Author): The second author provided valuable guidance and insight during the course of the study. The second author also provided helpful comments on the manuscript.

- Carl F. Ollivier-Gooch (Third Author): The third author provided valuable guidance and insight during the course of the study. The third author also provided helpful comments on the manuscript.

Chapter 5: *Relinearization of the Error Transport Equations for Arbitrarily High-Order Error Estimates*

- William C. Tyson (First Author): The first author served as the main contributor and primary author of the work. The first author implemented ETE relinearization within the BlueRidge code base and performed numerical simulations for 2D viscous Burgers' equation, the quasi-1D nozzle problem, and the 2D Euler equations.
- Gary K. Yan (Second Author): The second author provided results for the 2D RANS equations. The second author also provided helpful comments on the manuscript.
- Christopher J. Roy (Third Author): The third author provided valuable guidance and insight during the course of the study. The third author also provided helpful comments on the manuscript.
- Carl F. Ollivier-Gooch (Fourth Author): The fourth author provided valuable guidance and insight during the course of the study. The fourth author also provided helpful comments on the manuscript.

Chapter 6: *BlueRidge: A Higher-Order Finite-Volume Solver*

- William C. Tyson (First Author): The first author served as the main contributor and primary author of the work. The first author wrote the majority of the source code in BlueRidge including but not limited to input/output routines, linear and nonlinear solvers, reconstruction routines, and physics subclasses. The first author wrote all unit tests for BlueRidge and performed rigorous code verification.
- Charles W. Jackson (Second Author): The second author provided valuable guidance with regard to code organization and data management. The second author contributed to the code development section of the manuscript and provided helpful comments throughout the remaining sections.
- Christopher J. Roy (Third Author): The third author provided valuable guidance and insight during the course of the study. The third author also provided helpful comments on the manuscript.

Bibliography

[1] P. D. Morris, A. Narracott, H. von Tengg-Kobligk, D. A. Silva Soto, S. Hsiao, A. Lungu, P. Evans, N. W. Bressloff, P. V. Lawford, D. R. Hose, J. P. Gunn, Computational fluid dynamics modelling in cardiovascular medicine, Heart 102 (2015) 18–28. doi:10.1136/heartjnl-2015-308044.
URL https://heart.bmj.com/content/early/2015/10/28/heartjnl-2015-308044

[2] K. A. Lundquist, F. K. Chow, J. K. Lundquist, An immersed boundary method for the weather research and forecasting model, Monthly Weather Review 138 (2010) 796–817. doi:10.1175/2009MWR2990.1.
URL https://doi.org/10.1175/2009MWR2990.1

[3] C. Audiffren, V. Forgues, R. Marcer, CFD simulation of a tsunami impacting a coastal city including numerous buildings., in: P. Gourbesville, J. Cunge, G. Caignaert (Eds.), Advances in Hydroinformatics., Springer Water, Springer, Singapore, 2018. doi:10.1007/978-981-10-7218-5_45.
URL https://doi.org/10.1007/978-981-10-7218-5_45

[4] M. Park, B. Lee-Rausch, C. Rumsey, FUN3D and CFL3D computations for the first high lift prediction workshop, in: 49th AIAA Aerospace Sciences Meeting including the New Horizons Forum and Aerospace Exposition, American Institute of Aeronautics and Astronautics (AIAA), 2011. doi:10.2514/6.2011-936.
URL http://dx.doi.org/10.2514/6.2011-936

[5] O. Axelsson, Finite difference methods, in: E. Stein, R. Borst, T. Hughes (Eds.), Encyclopedia of Computational Mechanics, 2nd Edition, John Wiley & Sons, Ltd., 2017. doi:10.1002/9781119176817.ecm2002.
URL https://doi.org/10.1002/9781119176817.ecm2002

[6] S. Brenner, C. Carstensen, Finite element methods, in: E. Stein, R. Borst, T. Hughes (Eds.), Encyclopedia of Computational Mechanics, 2nd Edition, John Wiley & Sons, Ltd., 2017. doi:10.1002/9781119176817.ecm2003.
URL https://doi.org/10.1002/9781119176817.ecm2003

[7] T. Barth, R. Herbin, M. Ohlberger, Finite volume methods: Foundations and analysis, in: E. Stein, R. Borst, T. Hughes (Eds.), Encyclopedia of Computational Mechanics, 2nd Edition, John Wiley & Sons, Ltd., 2017. doi:10.1002/9781119176817.ecm2010.
URL https://doi.org/10.1002/9781119176817.ecm2010

[8] W. Oberkampf, T. Trucano, Verification and validation in computational fluid dynamics, Progress in Aerospace Sciences 38 (3) (2002) 209–272. doi:10.1016/S0376-0421(02)00005-2.
URL https://doi.org/10.1016/S0376-0421(02)00005-2

[9] P. J. Roache, Quantification of uncertainty in computational fluid dynamics, Annual Review of Fluid Mechanics 29 (1) (1997) 123 – 160. doi:10.1146/annurev.fluid.29.1.123.
URL http://dx.doi.org/10.1146/annurev.fluid.29.1.123

[10] AIAA, AIAA guide for the verification and validation of computational fluid dynamics simulations (1998). doi:10.2514/4.472855.
URL https://arc.aiaa.org/doi/book/10.2514/4.472855

[11] C. J. Roy, M. A. McWherter-Payne, W. L. Oberkampf, Verification and validation for laminar hypersonic flowfields, Part 1: Verification, AIAA Journal 41 (10) (2003) 1934 – 1943. doi:10.2514/2.1909.
URL http://dx.doi.org/10.2514/2.1909

[12] C. Roy, W. Oberkampf, M. McWherter-Payne, Verification and validation for laminar hypersonic flowfields, Part 2: Validation, AIAA Journal 41 (10) (2003) 1944–1954. doi:10.2514/2.1884.
URL https://doi.org/10.2514/2.1884

[13] C. Rumsey, B. Smith, G. Huang, Description of a website resource for turbulence modeling verification and validation, in: 40th Fluid Dynamics Conference and Exhibit, American Institute of Aeronautics and Astronautics (AIAA), 2010. doi:10.2514/6.2010-4742.
URL http://dx.doi.org/10.2514/6.2010-4742

[14] J. Slotnick, A. Khodadoust, J. Alonso, D. Darmofal, W. Gropp, E. Lurie, D. Mavriplis, CFD Vision 2030: A path to revolutionary computational aerosciences, Tech. Rep. NASA/CR-2014-218178, NASA (March 2014).
URL https://ntrs.nasa.gov/search.jsp?R=20140003093

[15] D. Wilcox, Turbulence Modeling for CFD, 3rd Edition, DCW Industries, La Canada, CA, 2010.

[16] H. Xiao, J.-L. Wu, J.-X. Wang, R. Sun, C. Roy, Quantifying and reducing model-form uncertainties in Reynolds-averaged Navier-Stokes simulations: A data-driven, physics-informed Bayesian approach, Journal of Computational Physics 324 (2016) 115–136. doi:10.1016/j.jcp.2016.07.038.
URL http://dx.doi.org/10.1016/j.jcp.2016.07.038

[17] R. Burden, J. Faires, Numerical Analysis, Brooks/Cole, Cengage Learning, 2011.

[18] W. L. Oberkampf, C. J. Roy, Verification and Validation in Scientific Computing, Cambridge University Press (CUP), 2010. doi:10.1017/cbo9780511760396.
URL http://dx.doi.org/10.1017/CBO9780511760396

[19] A. Choudhary, C. Roy, Efficient residual-based mesh adaptation for 1D and 2D CFD applications, in: 49th AIAA Aerospace Sciences Meeting including the New Horizons Forum and Aerospace Exposition, American Institute of Aeronautics and Astronautics (AIAA), 2011. doi:10.2514/6.2011-214.
URL http://dx.doi.org/10.2514/6.2011-214

[20] L. F. Richardson, The approximate arithmetical solution by finite differences of physical problems involving differential equations, with an application to the stresses in a masonry dam, Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences 210 (459-470) (1911) 307 – 357. doi:10.1098/rsta.1911.0009.
URL http://dx.doi.org/10.1098/rsta.1911.0009

[21] K. Laflin, O. Brodersen, M. Rakowitz, J. Vassberg, R. Wahls, J. Morrison, E. Tinoco, J.-L. Godard, Summary of data from the second AIAA CFD drag prediction workshop (invited), in: 42nd AIAA Aerospace Sciences Meeting and Exhibit, American Institute of Aeronautics and Astronautics (AIAA), 2004. doi:10.2514/6.2004-555.
URL http://dx.doi.org/10.2514/6.2004-555

[22] J. Vassberg, E. Tinoco, M. Mani, O. Brodersen, B. Eisfeld, R. Wahls, J. Morrison, T. Zickuhr, K. Laflin, D. Mavriplis, Summary of the third AIAA CFD drag prediction workshop, in: 45th AIAA Aerospace Sciences Meeting and Exhibit, American Institute of Aeronautics and Astronautics (AIAA), 2007. doi:10.2514/6.2007-260.
URL http://dx.doi.org/10.2514/6.2007-260

[23] L. Fox, Some improvements in the use of relaxation methods for the solution of ordinary and partial differential equations, Proceedings of the Royal Society of London. Series A, Mathematical and Physical Sciences 190 (1020) (1947) 31 – 59.
URL http://www.jstor.org/stable/98009

[24] V. Pereyra, Iterated deferred corrections for nonlinear operator equations, Numerische Mathematik 10 (4) (1967) 316 – 323. doi:10.1007/bf02162030.
URL http://dx.doi.org/10.1007/BF02162030

[25] V. Pereyra, Iterated deferred corrections for nonlinear boundary value problems, Numerische Mathematik 11 (2) (1968) 111 – 125. doi:10.1007/bf02165307.
URL http://dx.doi.org/10.1007/BF02165307

[26] H. J. Stetter, The defect correction principle and discretization methods, Numerische Mathematik 29 (4) (1978) 425 – 443. doi:10.1007/bf01432879.
URL http://dx.doi.org/10.1007/BF01432879

[27] I. Babuška, T.Strouboulis, C. Upadhyay, A model study of the quality of a posteriori error estimators for linear elliptic problems. Error estimation in the interior of patchwise uniform grids of triangles, Computer Methods in Applied Mechanics and Engineering

114 (1994) 307–378. doi:10.1016/0045-7825(94)90177-5.
URL https://doi.org/10.1016/0045-7825(94)90177-5

[28] I. Babuška, T.Strouboulis, S. Gangaraj, C. Upadhyay, Pollution error in the h-version of the finite element method and the local quality of the recovered derivatives, Computer Methods in Applied Mechanics and Engineering 140 (1997) 1–37. doi:10.1016/S0045-7825(96)01013-4.
URL https://doi.org/10.1016/S0045-7825(96)01013-4

[29] X. Gu, T. Shih, Differentiating between source and location of error for solution-adaptive mesh refinement, in: 15th AIAA Computational Fluid Dynamics Conference, American Institute of Aeronautics and Astronautics (AIAA), 2001. doi:10.2514/6.2001-2660.
URL http://dx.doi.org/10.2514/6.2001-2660

[30] X. Zhang, J.-Y. Trépanier, R. Camarero, A posteriori error estimation for finite-volume solutions of hyperbolic conservation laws, Computer Methods in Applied Mechanics and Engineering 185 (1) (2000) 1 – 19. doi:10.1016/s0045-7825(99)00099-7.
URL http://dx.doi.org/10.1016/S0045-7825(99)00099-7

[31] X. Zhang, D. Pelletier, J.-Y. Trépanier, R. Camarero, Numerical assessment of error estimators for Euler equations, AIAA Journal 39 (9) (2001) 1706–1715. doi:10.2514/2.1528.
URL https://arc.aiaa.org/doi/pdf/10.2514/2.1528

[32] Y. Qin, T. Shih, A discrete transport equation for error estimation in CFD, in: 40th AIAA Aerospace Sciences Meeting and Exhibit, American Institute of Aeronautics and Astronautics (AIAA), 2002. doi:10.2514/6.2002-906.
URL http://dx.doi.org/10.2514/6.2002-906

[33] Y. Qin, T. Shih, A method for estimating grid-induced errors in finite-difference and finite-volume methods, in: 41st AIAA Aerospace Sciences Meeting and Exhibit, American Institute of Aeronautics and Astronautics (AIAA), 2003. doi:10.2514/6.2003-845.
URL https://arc.aiaa.org/doi/pdf/10.2514/6.2003-845

[34] Y. Qin, P. Keller, R. Sun, E. Hernandez, C.-Y. Perng, N. Trigui, Z. Han, F. Shen, T. Shieh, T.-P. Shih, Estimating grid-induced errors in CFD by discrete-error-transport equations, in: 42nd AIAA Aerospace Sciences Meeting and Exhibit, American Institute of Aeronautics and Astronautics (AIAA), 2004. doi:10.2514/6.2004-656.
URL https://arc.aiaa.org/doi/pdf/10.2514/6.2004-656

[35] Y. Qin, X. Chi, T.-P. Shih, Modeling the residual in error-transport equations for estimating grid-induced errors in CFD solutions, in: 44th AIAA Aerospace Sciences Meeting and Exhibit, American Institute of Aeronautics and Astronautics (AIAA), 2006.

doi:10.2514/6.2006-892.
URL https://arc.aiaa.org/doi/pdf/10.2514/6.2006-892

[36] I. Celik, G. Hu, Single grid error estimation using error transport equation, Journal of Fluids Engineering 126 (5) (2004) 778. doi:10.1115/1.1792254.
URL http://dx.doi.org/10.1115/1.1792254

[37] T. Shih, B. Williams, Development and evaluation of an a posteriori method for estimating and correcting grid-induced errors in solutions of the Navier-Stokes equations, in: 47th AIAA Aerospace Sciences Meeting including the New Horizons Forum and Aerospace Exposition, American Institute of Aeronautics and Astronautics (AIAA), 2009. doi:10.2514/6.2009-1499.
URL https://doi.org/10.2514%2F6.2009-1499

[38] B. Palmério, A. Dervieux, 2D and 3D unstructured mesh adaption relying on physical analogy, in: Proceedings of the 2nd International Conference on Numerical Grid Generation in CFD, Pineridge, Swansea, Wales, U.K., 1988, pp. 178–185.

[39] J. Peraire, M. Vahdati, K. Morgan, O. Zienkiewicz, Adaptive remeshing for compressible flow computations, Journal of Computational Physics 72 (2) (1987) 449 – 466. doi:10.1016/0021-9991(87)90093-3.
URL http://dx.doi.org/10.1016/0021-9991(87)90093-3

[40] M. Reggio, J. Trépanier, H. Zhang, R. Camarero, Numerical simulation of the gas flow in a circuit-breaker, International Journal for Numerical Methods in Engineering 34 (2) (1992) 607–618. doi:10.1002/nme.1620340213.
URL https://onlinelibrary.wiley.com/doi/abs/10.1002/nme.1620340213

[41] C. Ilinca, X. Zhang, J.-Y. Trépanier, R. Camarero, A comparison of three error estimation techniques for finite-volume solutions of compressible flows, Computer Methods in Applied Mechanics and Engineering 189 (4) (2000) 1277–1294. doi:10.1016/S0045-7825(99)00377-1.
URL https://doi.org/10.1016/S0045-7825(99)00377-1

[42] P. Cavallo, N. Sinha, An error transport equation with practical applications, in: 18th AIAA Computational Fluid Dynamics Conference, American Institute of Aeronautics and Astronautics (AIAA), 2007, p. 4092. doi:10.2514/6.2007-4092.
URL https://arc.aiaa.org/doi/pdf/10.2514/6.2007-4092

[43] P. A. Cavallo, N. Sinha, Error quantification for computational aerodynamics using an error transport equation, Journal of Aircraft 44 (6) (2007) 1954–1963. doi:10.2514/1.33154.
URL https://arc.aiaa.org/doi/pdf/10.2514/1.33154

[44] P. Cavallo, N. Sinha, M. O'Gara, Viscous error transport equation for error quantification of turbulent flows, in: 38th AIAA Fluid Dynamics Conference and Exhibit, American Institute of Aeronautics and Astronautics (AIAA), 2008. doi:10.2514/6.2008-3851.
URL https://arc.aiaa.org/doi/pdf/10.2514/6.2008-3851

[45] A. Hay, M. Visonneau, Error estimation using the error transport equation for finite-volume methods and arbitrary meshes, International Journal of Computational Fluid Dynamics 20 (7) (2006) 463 – 479. doi:10.1080/10618560600835934.
URL http://dx.doi.org/10.1080/10618560600835934

[46] J. Banks, J. Hittinger, J. Connors, C. Woodward, A posteriori error estimation via nonlinear error transport, Tech. Rep. LLNL-JRNL-562712, Lawrence Livermore National Laboratory (June 2012).

[47] J. Banks, J. Hittinger, J. Connors, C. Woodward, Numerical error estimation for nonlinear hyperbolic PDEs via nonlinear error transport, Computer Methods in Applied Mechanics and Engineering 213-216 (2012) 1 – 15. doi:10.1016/j.cma.2011.11.021.
URL http://dx.doi.org/10.1016/j.cma.2011.11.021

[48] J. M. Derlaga, Application of improved truncation error estimation techniques to adjoint based error estimation and grid adaptation, Ph.D. thesis, Virginia Polytechnic Institute and State University, Blacksburg, VA (Jul. 2015).
URL http://hdl.handle.net/10919/54592

[49] G. Yan, C. F. Ollivier-Gooch, Accuracy of discretization error estimation by the error transport equation on unstructured meshes - Nonlinear systems of equations, in: 22nd AIAA Computational Fluid Dynamics Conference, American Institute of Aeronautics and Astronautics (AIAA), 2015. doi:10.2514/6.2015-2747.
URL http://dx.doi.org/10.2514/6.2015-2747

[50] G. Yan, C. F. Ollivier-Gooch, Accuracy of discretization error estimation by the error transport equation on unstructured meshes, in: 53rd AIAA Aerospace Sciences Meeting, American Institute of Aeronautics and Astronautics (AIAA), 2015. doi:10.2514/6.2015-1264.
URL http://dx.doi.org/10.2514/6.2015-1264

[51] G. Yan, C. F. Ollivier-Gooch, Discretization error estimation by the error transport equation on unstructured meshes - Applications to viscous flows, in: 54th AIAA Aerospace Sciences Meeting, American Institute of Aeronautics and Astronautics (AIAA), 2016. doi:10.2514/6.2016-0836.
URL http://dx.doi.org/10.2514/6.2016-0836

[52] G. Yan, C. Ollivier-Gooch, Towards higher order discretization error estimation by error transport using unstructured finite-volume methods for unsteady problems, Computers

& Fluids 154 (2017) 245–255. doi:10.1016/j.compfluid.2017.06.012.
URL https://doi.org/10.1016/j.compfluid.2017.06.012

[53] J. M. Derlaga, M. A. Park, S. K. Rallabhandi, Application of exact error transport equations and adjoint error estimation to AIAA workshops, in: 55th AIAA Aerospace Sciences Meeting, American Institute of Aeronautics and Astronautics (AIAA), 2017, p. 0076. doi:10.2514/6.2017-0076.
URL https://doi.org/10.2514/6.2017-0076

[54] C. Roy, Review of discretization error estimators in scientific computing, in: 48th AIAA Aerospace Sciences Meeting Including the New Horizons Forum and Aerospace Exposition, American Institute of Aeronautics and Astronautics (AIAA), 2010. doi:10.2514/6.2010-126.
URL http://dx.doi.org/10.2514/6.2010-126

[55] M. B. Giles, N. Pierce, E. Süli, Progress in adjoint error correction for integral functionals, Computing and Visualization in Science 6 (2-3) (2004) 113 – 121. doi:10.1007/s00791-003-0115-y.
URL http://dx.doi.org/10.1007/s00791-003-0115-y

[56] D. A. Venditti, Grid adaptation for functional outputs of compressible flow simulations, Ph.D. thesis, Massachusetts Institute of Technology (Jun. 2002).
URL http://hdl.handle.net/1721.1/29246

[57] R. Hartmann, Multitarget error estimation and adaptivity in aerodynamic flow simulations, SIAM Journal on Scientific Computing 31 (1) (2008) 708 – 731. doi:10.1137/070710962.
URL https://doi.org/10.1137/070710962

[58] N. A. Pierce, M. B. Giles, Adjoint recovery of superconvergent functionals from PDE approximations, SIAM Review 42 (2) (2000) 247 – 264. doi:10.1137/s0036144598349423.
URL http://dx.doi.org/10.1137/S0036144598349423

[59] M. B. Giles, N. A. Pierce, Analytic adjoint solutions for the quasi-one-dimensional Euler equations, Journal of Fluid Mechanics 426 (2001) 327 – 345. doi:10.1017/s0022112000002366.
URL http://dx.doi.org/10.1017/S0022112000002366

[60] M. Giles, N. Pierce, Adjoint equations in CFD - duality, boundary conditions and solution behaviour, in: 13th AIAA Computational Fluid Dynamics Conference, American Institute of Aeronautics and Astronautics (AIAA), 1997. doi:10.2514/6.1997-1850.
URL http://dx.doi.org/10.2514/6.1997-1850

[61] M. Giles, M. Duta, J.-D. Müller, N. Pierce, Algorithm developments for discrete adjoint methods, AIAA Journal 41 (2) (2003) 198 – 205. doi:10.2514/2.1961.
URL http://dx.doi.org/10.2514/2.1961

[62] N. A. Pierce, M. B. Giles, Adjoint and defect error bounding and correction for functional estimates, Journal of Computational Physics 200 (2) (2004) 769 – 794. doi:10.1016/j.jcp.2004.05.001.
URL http://dx.doi.org/10.1016/j.jcp.2004.05.001

[63] D. A. Venditti, D. L. Darmofal, Adjoint error estimation and grid adaptation for functional outputs: Application to quasi-one-dimensional flow, Journal of Computational Physics 164 (1) (2000) 204 – 227. doi:10.1006/jcph.2000.6600.
URL http://dx.doi.org/10.1006/jcph.2000.6600

[64] D. A. Venditti, D. L. Darmofal, Grid adaptation for functional outputs: Application to two-dimensional inviscid flows, Journal of Computational Physics 176 (1) (2002) 40 – 69. doi:10.1006/jcph.2001.6967.
URL http://dx.doi.org/10.1006/jcph.2001.6967

[65] D. A. Venditti, D. L. Darmofal, Anisotropic grid adaptation for functional outputs: Application to two-dimensional viscous flows, Journal of Computational Physics 187 (1) (2003) 22 – 46. doi:10.1016/s0021-9991(03)00074-3.
URL http://dx.doi.org/10.1016/S0021-9991(03)00074-3

[66] M. Park, Adjoint-based, three-dimensional error prediction and grid adaptation, in: 32nd AIAA Fluid Dynamics Conference and Exhibit, American Institute of Aeronautics and Astronautics (AIAA), 2002. doi:10.2514/6.2002-3286.
URL http://dx.doi.org/10.2514/6.2002-3286

[67] M. A. Park, Anisotropic output-based adaptation with tetrahedral cut cells for compressible flows, Ph.D. thesis, Massachusetts Institute of Technology (Sep. 2008).
URL http://hdl.handle.net/1721.1/46363

[68] M. Nemec, M. Aftosmis, S. Murman, T. Pulliam, Adjoint formulation for an embedded-boundary cartesian method, in: 43rd AIAA Aerospace Sciences Meeting and Exhibit, American Institute of Aeronautics and Astronautics (AIAA), 2005. doi:10.2514/6.2005-877.
URL http://dx.doi.org/10.2514/6.2005-877

[69] M. Nemec, M. Aftosmis, Adjoint error estimation and adaptive refinement for embedded-boundary Cartesian meshes, in: 18th AIAA Computational Fluid Dynamics Conference, American Institute of Aeronautics and Astronautics (AIAA), 2007. doi:10.2514/6.2007-4187.
URL http://dx.doi.org/10.2514/6.2007-4187

[70] M. Nemec, M. Aftosmis, M. Wintzer, Adjoint-based adaptive mesh refinement for complex geometries, in: 46th AIAA Aerospace Sciences Meeting and Exhibit, American Institute of Aeronautics and Astronautics (AIAA), 2008. doi:10.2514/6.2008-725.
URL http://dx.doi.org/10.2514/6.2008-725

[71] M. B. Giles, E. Süli, Adjoint methods for PDEs: a posteriori error analysis and postprocessing by duality, Acta Numerica 11 (2002) 145 – 236. doi:10.1017/S096249290200003X.
URL https://doi.org/10.1017/S096249290200003X

[72] K. Fidkowski, D. Darmofal, Review of output-based error estimation and mesh adaptation in computational fluid dynamics, AIAA Journal 49 (4) (2011) 673–694. doi:10.2514/1.J050073.
URL https://arc.aiaa.org/doi/pdf/10.2514/1.J050073

[73] I. B. Celik, D. R. Parsons, Prediction of discretization error using the error transport equation, Journal of Computational Physics 339 (2017) 96 – 125. doi:10.1016/j.jcp.2017.02.058.
URL https://doi.org/10.1016%2Fj.jcp.2017.02.058

[74] M. B. Giles, N. A. Pierce, Improved lift and drag estimates using adjoint Euler equations, in: 14th AIAA Computational Fluid Dynamics Conference, American Institute of Aeronautics and Astronautics (AIAA), 1999. doi:10.2514/6.1999-3293.
URL https://doi.org/10.2514/6.1999-3293

[75] M. B. Giles, N. A. Pierce, Adjoint Error Correction for Integral Outputs, Springer Berlin Heidelberg, Berlin, Heidelberg, 2003, Ch. 2, pp. 47 – 95. doi:10.1007/978-3-662-05189-4_2.
URL http://dx.doi.org/10.1007/978-3-662-05189-4_2

[76] M. Sharbatdar, C. F. Ollivier Gooch, Eigenanalysis of truncation and discretization error on unstructured meshes, in: 21st AIAA Computational Fluid Dynamics Conference, 2013, p. 3089. doi:10.2514/6.2013-3089.
URL https://doi.org/10.2514/6.2013-3089

[77] M. Sharbatdar, A. Jalali, C. Ollivier-Gooch, Smoothed truncation error in functional error estimation and correction using adjoint methods in an unstructured finite-volume solver, Computers & Fluids 140 (2016) 406–421. doi:10.1016/j.compfluid.2016.10.019.
URL https://doi.org/10.1016/j.compfluid.2016.10.019

[78] M. Sharbatdar, Error estimation and mesh adaptation paradigm for unstructured mesh finite volume methods, Ph.D. thesis, University of British Columbia (Jan. 2017).
URL http://hdl.handle.net/2429/60359

[79] C. Ollivier-Gooch, C. Roy, Reducing truncation error on unstructured meshes by vertex movement, in: 42nd AIAA Fluid Dynamics Conference and Exhibit, American Institute of Aeronautics and Astronautics (AIAA), 2012. doi:10.2514/6.2012-2712.
URL http://dx.doi.org/10.2514/6.2012-2712

[80] F. Fraysse, J. de Vicente, E. Valero, The estimation of truncation error by τ-estimation revisited, Journal of Computational Physics 231 (9) (2012) 3457 – 3482. doi:10.1016/j.jcp.2011.09.031.
URL http://dx.doi.org/10.1016/j.jcp.2011.09.031

[81] T. Phillips, Residual-based discretization error estimation for computational fluid dynamics, Ph.D. thesis, Virginia Polytechnic Institute and State University (Sep. 2014).
URL http://hdl.handle.net/10919/50647

[82] B. Diskin, J. L. Thomas, Notes on accuracy of finite-volume discretization schemes on irregular grids, Applied Numerical Mathematics 60 (3) (2010) 224 – 226. doi:10.1016/j.apnum.2009.12.001.
URL http://dx.doi.org/10.1016/j.apnum.2009.12.001

Chapter 2

Improved Functional-Based Error Estimation and Adaptation using Error Transport Equations and Krylov Subspace Methods

William C. Tyson,[a] Katarzyna Świrydowicz,[b] Joseph M. Derlaga,[c]
Christopher J. Roy,[a] Eric de Sturler[b]

[a] Kevin T. Crofton Department of Aerospace and Ocean Engineering, Virginia Tech
215 Randolph Hall, 460 Old Turner Street, Blacksburg, VA, 24061

[b] Department of Mathematics, Virginia Tech
460 McBryde Hall, 225 Stanger Street, Blacksburg, VA 24061

[c] Computational Aerosciences Branch, NASA Langley Research Center
1 NASA Drive, Hampton, VA 23681

Keywords: Computational fluid dynamics, Discrete adjoint method, Error estimation, Grid adaptation, Finite-volume method

Attribution

- William C. Tyson (First Author): The first author served as the main contributor and primary author of the work. The first author implemented the primal and dual solvers for Burgers' equation and the quasi-1D nozzle problem. Additionally, the first author performed all numerical studies presented.

- Katarzyna Świrydowicz (Second Author): The second author assisted in drafting the manuscript and provided codes for estimating the bilinear form and for computing the Sparse Approximate Inverse (SAI) preconditioner.
- Joseph M. Derlaga (Third Author): The third author provided valuable guidance and insight during the course of the study. The third author also provided helpful comments on the manuscript.
- Christopher J. Roy (Fourth Author): The fourth author provided valuable guidance and insight during the course of the study. The fourth author also provided helpful comments on the manuscript.
- Eric de Sturler (Fifth Author): The fifth author provided valuable guidance and insight during the course of the study. The fifth author also provided helpful comments on the manuscript.

Abstract

Adjoint methods have become quite popular for both functional error estimation and adaptation. A disadvantage of traditional adjoint methods is that one adjoint solution can only provide an error estimate and adaptation indicator for one functional of interest. Since most engineering applications rely upon multiple functionals to assess a given system, an adjoint solution must be obtained for each functional of interest, consequently increasing the overall computational cost. In this paper, new techniques are presented, which provide the same error estimates and adaptation indicators as a conventional adjoint method, but do so much more efficiently, especially when multiple functionals must be examined. For functional error estimation, the adjoint solve is replaced by the solution of an error transport equation for the local discretization error. Functional error estimates are then formed by taking an inner product of the discretization error with linearizations of the functional. This method is shown to produce the same functional error estimates as an adjoint solve. For functional-based adaptation, the functional error estimate is written as a bilinear form and efficiently solved using a Krylov subspace method. With this approach, the adjoint solution is obtained in an approximate sense but is still accurate enough to form useful adaptation indicators. Results are presented for 1D inviscid and viscous flow problems.

2.1 Introduction

Within the context of computational fluid dynamics, adjoint methods have gained popularity in recent years for the purposes of both error estimation and adaptation. Adjoint methods provide error estimates for output functionals, which can be used to correct the functionals to

higher-order. In addition, adjoint methods can provide an adaptation indicator that marks regions of the domain for adaptation that contribute most to the overall error in a functional. Giles and Pierce applied the continuous adjoint method to several canonical problems [1, 2, 3]. For the discrete adjoint method, Venditti and Darmofal [4, 5, 6] computed coarse-grid error estimates that, when applied as a correction, drove the coarse-grid functionals toward their corresponding values on a finer mesh. Park [7] used the discrete adjoint solution to drive an adaptation scheme aimed at improving drag estimates for a simplified aircraft configuration.

While adjoint methods have shown great promise for both functional error estimation and adaptation, the solution to the adjoint problem only supplies an error estimate and adaptivity information for one functional. For aerodynamics applications, multiple solution functionals are typically required, such as lift, pressure drag, viscous drag, and moments. With the current state of adjoint methods, an adjoint solution must be obtained for each functional for which an error estimate is required. Furthermore, adjoint-based adaptation, which targets one functional, can actually increase the error in another functional [8, 9]. Therefore, a separate grid and adjoint solution must be obtained for each functional of interest. For large engineering problems, solving multiple adjoint problems can become expensive as memory requirements and computation time increase with the number of functionals. This could ultimately hinder the widespread use of adjoint-based error estimation and adaptation in a commercial setting.

In this paper, new techniques are proposed for performing functional error estimation and adaptation that provide the same error estimates and adaptation indicators as a conventional adjoint method, but do so much more efficiently. This paper builds upon the work of Derlaga [10], who experimentally showed that local discretization error estimates can provide the same functional error estimate as an adjoint method. In this paper, it is analytically shown that an adjoint-based error estimate is equivalent to the inner product of a functional linearization with a local discretization error estimate. This approach not only provides local error estimates for all solution variables, but also improves the efficiency of functional error estimation for multiple functionals. For instance, n adjoint problems for n functionals may be replaced with one linear solve for the local solution error. For functional-based adaptation, the functional error estimate, or equivalently the functional correction, is viewed as a bilinear form and solved for using a Krylov subspace method. Due to the convergence properties of the Krylov solver, an accurate error estimate may be obtained in addition to an approximate adjoint solution, which is still accurate enough to drive an adaptation procedure. The applications examined in this work include 1D inviscid and viscous flow problems.

2.2 Background

2.2.1 Finite-Volume Discretization

In this paper, efforts are focused on improving error estimation and adaptation techniques for the finite-volume method (FVM). First, consider a general conservation law

$$\frac{\partial \tilde{\mathbf{u}}}{\partial t} + \nabla \cdot \mathbb{F}(\tilde{\mathbf{u}}) = \mathbf{s}(\tilde{\mathbf{u}}) \tag{2.1}$$

defined over the domain $\Omega \subset \mathbb{R}^m \times [0, \infty), \mathbf{x} \in \mathbb{R}^m, m \in \{1, 2, 3\}$. In general, the flux, $\mathbb{F}(\cdot)$, and source term, $\mathbf{s}(\cdot)$, can also be functions of $\nabla\tilde{\mathbf{u}}$. However, this gradient dependency has been disregarded here for the sake of brevity. Let the domain of interest be discretized into a set of non-overlapping control volumes, $\mathcal{T}_h$, such that

$$\Omega = \bigcup_{i=1}^{N_v} \Omega_i, \quad \Omega_i \in \mathcal{T}_h \ . \tag{2.2}$$

For each control volume in $\mathcal{T}_h$, Eq. 2.1 is recast into a weak form by integrating over the volume and using the divergence theorem to convert the flux divergence integral into a surface integral

$$\mathbf{N}(\tilde{\mathbf{u}}) \coloneqq \frac{\partial}{\partial t} \int_{\Omega_i} \tilde{\mathbf{u}} \ d\Omega + \oint_{\partial\Omega_i} \mathbb{F}(\tilde{\mathbf{u}}) \cdot \hat{\mathbf{n}} \ dS - \int_{\Omega_i} \mathbf{s}(\tilde{\mathbf{u}}) \ d\Omega = \mathbf{0} \ . \tag{2.3}$$

The continuous operator, $\mathbf{N}(\cdot)$, is defined for a control volume fixed in space and operates on a continuous function. In the finite-volume method, a discrete solution, $\mathbf{u}_h$, is sought that approximates the control volume average of $\tilde{\mathbf{u}}$. The discrete operator corresponding to Eq. 2.3 is given by the following:

$$\mathbf{N}_h^p(\mathbf{u}_h) \coloneqq |\Omega_i| \frac{d\mathbf{u}_h}{dt} + \oint_{\partial\Omega_i} \mathbb{F}_h(I_h^p \mathbf{u}_h^+, I_h^p \mathbf{u}_h^-) \cdot \hat{\mathbf{n}} \ dS - \int_{\Omega_i} \mathbf{s}(I_h^p \mathbf{u}_h) \ d\Omega = \mathbf{0} \ , \tag{2.4}$$

where $|\Omega_i|$ is the volume of control volume Ω_i and $\mathbb{F}_h(\cdot, \cdot)$ is a discrete flux function. The notation $(\cdot)^{+/-}$ is used to indicate that the reconstruction, I_h^p, which is continuous in control volume Ω_i, is discontinuous across control volume boundaries, $\partial\Omega_i$. Throughout this paper, the operator, I_a^b, is used as restriction or prolongation operator, which denotes the transition between a continuous and discrete space; the subscript, a, indicates the starting space and the superscript, b, indicates the resultant space. An empty subscript or superscript represents a continuous space, $\mathcal{V}$. A subscript or superscript h denotes a control volume average on a mesh with a characteristic size, h. Any other subscript or superscript (e.g., p) is used to denote a discrete function space, $\mathcal{V}_h \subset \mathcal{V}$. The continuous and discrete operators may be directly related using the Generalized Truncation Error Expression (GTEE) [11, 12],

$$\mathbf{N}_h^p(I^h \mathbf{u}) = \mathbf{N}(\mathbf{u}) + \tau_h^p(\mathbf{u}) \ , \tag{2.5}$$

where $\mathbf{u}$ is a continuous function, and $\tau_h^p(\cdot)$ is the truncation error. For finite-volume schemes, it is generally difficult to determine the exact form of the truncation error, but, for model problems, truncation error is a function of the discretization scheme, solution derivatives, local mesh resolution, and grid metrics [13].

The discrete solution, $\mathbf{u}_h$, often referred to as the primal solution, is obtained by combining Eq. 2.4 for each control volume into a system of ordinary differential equations

$$|\mathbf{\Omega}|\frac{d\mathbf{u}_h}{dt} + \mathbf{R}(\mathbf{u}_h) = \mathbf{0} \ . \tag{2.6}$$

The solution is integrated in pseudo-time to a steady-state using backward Euler time integration. The solution update, $\Delta\mathbf{u}_h^n = \mathbf{u}_h^{n+1} - \mathbf{u}_h^n$, is determined by linearizing the residual, $\mathbf{R}(\mathbf{u}_h)$, about the current solution and solving the following linear system

$$\left[\frac{|\mathbf{\Omega}|}{\Delta t} + \frac{\partial \mathbf{R}}{\partial \mathbf{u}_h}\right]\Delta\mathbf{u}_h^n = -\mathbf{R}(\mathbf{u}_h^n) \ . \tag{2.7}$$

The linearization of the primal residual is performed via hand differentiation. The residual Jacobian matrix, $\frac{\partial \mathbf{R}}{\partial \mathbf{u}_h}$, need not be an exact linearization of the discrete operator since the left-hand side of Eq. 2.7 is only used to drive the residual to zero. While forming an exact linearization can be prohibitively expensive in terms of storage and coding requirements, an exact linearization can improve convergence properties of the solver [14]. The temporal term along the diagonal may be adjusted to improve the conditioning of the system, especially when the current solution is far from the steady-state. As $\Delta t \to \infty$, Eq. 2.7 approaches Newton's method.

2.2.2 Linearized Error Transport Equations

Estimates of the local solution error may be obtained by deriving and solving an error transport equation (ETE), which relates truncation error to the discretization error. First, consider an expansion of $\mathbf{N}_h^p(I^h\tilde{\mathbf{u}})$ about the exact discrete solution, $\mathbf{u}_h$,

$$\mathbf{N}_h^p(I^h\tilde{\mathbf{u}}) = \mathbf{N}_h^p(\mathbf{u}_h) - \frac{\partial \mathbf{N}_h^p}{\partial \mathbf{u}_h}\varepsilon_h^p + O\Big(||\varepsilon_h^p||^2\Big) \ , \tag{2.8}$$

where $\varepsilon_h^p = \mathbf{u}_h - I^h\tilde{\mathbf{u}}$ is the local discretization error. Assuming $\mathbf{u}_h$ is in the asymptotic range and any discontinuous features are sufficiently resolved, the local discretization error converges at $O(h^p)$. The left-hand side of Eq. 2.8 is equivalent to truncation error by the following relation

$$\begin{aligned} \tau_h^p(\tilde{\mathbf{u}}) &= \mathbf{N}_h^p(I^h\tilde{\mathbf{u}}) - \cancelto{0}{\mathbf{N}(\tilde{\mathbf{u}})} \\ &= \mathbf{N}_h^p(I^h\tilde{\mathbf{u}}) \ . \end{aligned} \tag{2.9}$$

In practice, the exact continuous solution, $\tilde{\mathbf{u}}$, is not known, and truncation error is typically estimated [15, 16, 17]. Since truncation error estimation is not the focus of this paper, it is

assumed that an accurate truncation error estimate can be obtained, and exact truncation error is used instead. By neglecting higher-order terms and using Eq. 2.4 and Eq. 2.9, the linearized ETE may be written as

$$\frac{\partial \mathbf{N}_h^p}{\partial \mathbf{u}_h}\varepsilon_h^p = -\tau_h^p(\tilde{\mathbf{u}}) \tag{2.10}$$

where ε_h^p is now used to represent an estimate of the exact discretization error. Although Eq. 2.10 is simply a linear system, the linearized ETE are solved in a similar manner to the primal problem to improve the conditioning of the system, that is

$$\left[\frac{|\mathbf{\Omega}|}{\Delta t} + \frac{\partial \mathbf{R}}{\partial \mathbf{u}_h}\right]\Delta\varepsilon_h^{p,n} = -\left[\tau_h^p(\tilde{\mathbf{u}}) + \frac{\partial \mathbf{R}}{\partial \mathbf{u}_h}\varepsilon_h^{p,n}\right] . \tag{2.11}$$

Since this paper focuses only on steady-state error estimation, the Jacobian in Eq. 2.10 has been replaced with the residual Jacobian, $\frac{\partial \mathbf{R}}{\partial \mathbf{u}_h}$. Similar to the primal problem, the residual Jacobian matrix on the left-hand side of Eq. 2.11 need not be an exact linearization of the discrete operator while the residual Jacobian on the right-hand side must be an exact linearization in order to obtain an error estimate which asymptotically converges at the correct rate. Eq. 2.11 is solved at each pseudo-time step using GMRES [18].

2.2.3 Discrete Adjoint Methods

Adjoint methods were originally used in design optimization to obtain sensitivities of a functional with respect to a set of design parameters [19], but have been largely adopted in the CFD community for functional error estimation and adaptation [7, 9, 20, 21]. Functionals are typically integrated quantities over a surface or a given portion of the domain but may also be point values. In this work, we utilize a discrete adjoint method rather than a continuous adjoint method. Asymptotically, the two methods are equivalent assuming a dual consistent discretization. First, consider an expansion of a discrete solution functional, J_h, about the exact discrete solution, $\mathbf{u}_h$,

$$J_h(I^h\tilde{\mathbf{u}}) = J_h(\mathbf{u}_h) - \frac{\partial J_h}{\partial \mathbf{u}_h}\varepsilon_h^p + O\Big(||\varepsilon_h^p||^2\Big). \tag{2.12}$$

By rearranging Eq. 2.12, the error in the functional, ε_{J_h}, may be defined as

$$\varepsilon_{J_h} \coloneqq J_h(\mathbf{u}_h) - J_h(I^h\tilde{\mathbf{u}}) = -\lambda_h^T\tau_h^p(\tilde{\mathbf{u}}) + O\Big(||\varepsilon_h^p||^2\Big) \tag{2.13}$$

where λ_h are the discrete adjoint variables which satisfy the following linear adjoint equation

$$\left[\frac{\partial \mathbf{N}_h^p}{\partial \mathbf{u}_h}\right]^T \lambda_h = \left[\frac{\partial J_h}{\partial \mathbf{u}_h}\right]^T . \tag{2.14}$$

Because the adjoint variables represent discrete sensitivities of the functional to perturbations in the discrete operator, as illustrated by Eq. 2.14, the adjoint solution is often used to flag areas of the domain for adaptation that contribute most to the error in the functional. Similar to the primal problem and the linearized ETE, the dual problem is marched in pseudo-time according to the following relation in order to increase the robustness of the adjoint solver,

$$\left[\frac{|\mathbf{\Omega}|}{\Delta t} + \left(\frac{\partial \mathbf{R}}{\partial \mathbf{u}_h}\right)^T\right]\Delta\lambda_h^n = -\left[-\left(\frac{\partial J_h}{\partial \mathbf{u}_h}\right)^T + \left(\frac{\partial \mathbf{R}}{\partial \mathbf{u}_h}\right)^T \lambda_h^n\right] . \tag{2.15}$$

This approach to solving the adjoint problem was originally proposed by Nielsen et al. [22]. Again, an approximate linearization may be used on the left-hand side, while an exact linearization is required on the right-hand side. Once an adjoint solution has been obtained, a functional error estimate can be computed as the inner product of the adjoint solution with the truncation error. The functional error estimate can then be used to correct the functional to higher-order. Since the discretization error, ε_h^p, converges at $O(h^p)$, the corrected functional will converge at a rate of $O(h^{2p})$ assuming the discrete solution, $\mathbf{u}_h$, is in the asymptotic range and is sufficiently smooth in the region where $J_h(\mathbf{u}_h)$ is computed.

2.3 Methodology

2.3.1 Error Estimation for Multiple Functionals

Adjoint methods can provide an error estimate for any output functional of interest. However, when multiple functionals must be analyzed, traditional adjoint methods can become computationally expensive since an adjoint problem must be solved for each functional. As the number of functionals increases, the computational cost of solving multiple adjoint problems can become significant. For such applications, more efficient methods should be used. In this section, a new approach for obtaining functional error estimates is presented, which is more efficient than a standard adjoint method when multiple functionals are required [23]. With this approach, n adjoint solves for n functionals are replaced by one linear solve for the local discretization error everywhere in the domain. First, recall that the error in a given functional can be approximated as

$$\varepsilon_{J_h} \approx -\lambda_h^T \tau_h^p(\tilde{\mathbf{u}}) . \tag{2.16}$$

Now, rather than explicitly solving for the adjoint variables, Eq. 2.14 can be used to instead rewrite the functional error estimate as follows

$$\varepsilon_{J_h} \approx -\frac{\partial J_h}{\partial \mathbf{u}_h}\left[\frac{\partial \mathbf{N}_h^p}{\partial \mathbf{u}_h}\right]^{-1} \tau_h^p(\tilde{\mathbf{u}}) . \tag{2.17}$$

Computing the inverse of the residual Jacobian is extremely expensive and should never be evaluated for estimating functional errors. Alternatively, using Eq. 2.10, it can be seen that

the product of the inverse of the residual Jacobian and the truncation error is equivalent to the local discretization error estimate obtained from the linearized ETE. Therefore, the functional error estimate can be viewed as either the inner product of the adjoint solution with the truncation error or as the inner product of the linearized ETE solution with the functional linearization

$$\varepsilon_{J_h} \approx \overbrace{-\frac{\partial J_h}{\partial \mathbf{u}_h}}^{\text{Adjoint Problem, } -\lambda_h} \underbrace{\left[\frac{\partial \mathbf{N}_h^p}{\partial \mathbf{u}_h}\right]^{-1} \tau_h^p(\tilde{\mathbf{u}})}_{\text{Error Transport Equations, } -\varepsilon_h^p} . \tag{2.18}$$

By adopting the ETE viewpoint, n adjoint problems for n functionals are replaced by the solution of one ETE problem. The computation time for the required inner product is present in both the standard adjoint approach and the ETE approach. Therefore, it is expected that the computational savings of the ETE approach will be $O(n-1)$ separate adjoint solves. In practice, adjoint methods can be implemented with multiple right-hand side vectors when multiple functionals are required. However, the ETE approach is still expected to be more efficient than this implementation of the adjoint as only one right-hand side vector is required. Another important benefit of the ETE approach is that, unlike a standard adjoint, a discretization error estimate is obtained everywhere in the domain for all solution variables.

2.3.2 Approximate Adjoints for Functional Error Estimation and Adaptation

Since the adjoint solution represents sensitivities of the functional to perturbations of the residual operator, the adjoint variables are logical candidates for the formulation of an adaptation indicator. An adjoint-based adaptation indicator will target regions of the domain that contribute most to the error in a given functional. One disadvantage of the ETE approach to functional error estimation is that practitioners no longer have access to the adjoint variables if adaptation is to be performed. In order to perform both functional error estimation and adaptation, either the ETE or the adjoint problem would first need to be solved for functional error estimates. Then, should the ETE approach be used for error estimation, an additional adjoint solve would be required to compute adaptation indicators. In this situation, practitioners would need to use their best judgment to determine which combination of methods to use for a given application. It should be noted that we do not seek to answer the question of which adjoint solution should be used to drive adaptation as indicators for different functionals can lead to different adapted meshes. Instead, we propose a new method that addresses the case when a discretization error estimate, a functional error estimate, and an adaptation indicator are all desired for a single functional.

The method outlined in this section utilizes Krylov subspace methods for bilinear form estimation. With this approach, the adjoint problem and the ETE are solved in an approximate

sense, and, due to the convergence properties of the Krylov solver, a functional error estimate is obtained that is just as accurate as the adjoint or ETE approaches to functional error estimation. By approximately solving the adjoint problem, some accuracy in the adjoint solution is sacrificed. However, for the purposes of adaptation, the adjoint solution need not be extremely accurate, since adaptation indicators only need to be accurate enough to qualitatively capture the main features of the fully converged adjoint-based adaptation indicator. Some accuracy in the ETE solution is also sacrificed, but, in the context of adaptation, the ETE solution is likely only used to qualitatively indicate regions of the domain with large errors. Moreover, the use of Krylov solvers allows flexibility in choosing stopping criteria / accuracies.

2.3.2.1 Approximating Bilinear Forms by Krylov Subspace Methods

The functional error estimate in Eq. 2.17 is equivalent to a bilinear form

$$(b, c)_A = c^T A^{-1} b, \tag{2.19}$$

where A is a real-valued $n \times n$ matrix, and b and c are real-valued $n \times 1$ vectors. Bilinear forms appear in many applications, such as the computation of the scattering amplitude [24], computational fluid dynamics [25], quantum physics and inverse problems [26], and naturally arise in numerical linear algebra while computing error bounds. The goal is to approximate Eq. 2.19 without computing the inverse of A. The bilinear form can be obtained by simultaneously approximating the solution to the following linear systems

$$\text{ETE System:} \quad Ax = b \Leftrightarrow x = A^{-1} b, \tag{2.20}$$

$$\text{Adjoint System:} \quad A^T y = c \Leftrightarrow y = A^{-T} c, \tag{2.21}$$

and then computing

$$y^T A x = \left(A^{-T} c\right)^T A A^{-1} b = c^T A^{-1} b. \tag{2.22}$$

Several Krylov subspace methods, such as BiCG [27, 28] and LSQR [29], provide the capability of solving two systems simultaneously; this property has frequently been exploited in the literature [24, 30, 31]. The approach adopted in this work for approximating the bilinear form uses BiCG and is based upon the work of Strakoš and Tichý [30]. With this approach, a Krylov subspace for each linear system is constructed, $V_k = \text{span}\{r_0, Ar_0, \ldots, A^{k-1} r_0\}$ and $\tilde{V}_k = \text{span}\{s_0, A^T s_0, \ldots, \left(A^T\right)^{k-1} s_0\}$, where r_0 and s_0 are the initial residuals of the respective systems. Biorthogonality is enforced between the subspaces such that $\tilde{V}_k^T V_k$ is diagonal. This approach has two major advantages. Firstly, the error in the bilinear form estimate is proportional to the product of the residuals. In this way, the resulting bilinear form estimate is more accurate than if $Ax = b$ were individually solved to some modest tolerance and the bilinear form were subsequently computed as $c^T x$. Secondly, the bilinear form approximation is computed and updated as a part of the BiCG iteration with minimal additional cost

using the recurrences given in Ref. [30]. These recurrences rely only on the biorthogonality of vectors in consecutive iterations and, therefore, minimize the adverse effects of loss of orthogonality resulting from finite precision arithmetic.

An approximation to the bilinear form is derived using the following

$$s_0^T A^{-1} r_0 - s_i^T A^{-1} r_i = \sum_{j=0}^{i-1} \Big(s_j^T A^{-1} r_j - s_{j+1}^T A^{-1} r_{j+1} \Big), \tag{2.23}$$

where $s_j = c - A^T y_j$ and $r_j = b - A x_j$ are residuals for the adjoint and ETE systems, respectively, computed at iteration j of BiCG. Let x_0 and y_0 be arbitrary initial guesses to the solutions of the linear systems. Rearranging Eq. 2.23 and substituting r_0 and s_0 gives

$$\begin{aligned} \sum_{j=0}^{i-1} \Big(s_j^T A^{-1} r_j - s_{j+1}^T A^{-1} r_{j+1} \Big) + s_i^T A^{-1} r_i &= \Big(c - A^T y_0 \Big)^T A^{-1} (b - A x_0) \\ &= c^T A^{-1} b - \Big(c^T x_0 + y_0^T r_0 \Big). \end{aligned} \tag{2.24}$$

Next, Eq. 2.24 is solved for $c^T A^{-1} b$ to obtain

$$c^T A^{-1} b = \Big(c^T x_0 + y_0^T r_0 \Big) + \sum_{j=0}^{i-1} \Big(s_j^T A^{-1} r_j - s_{j+1}^T A^{-1} r_{j+1} \Big) + s_i^T A^{-1} r_i. \tag{2.25}$$

By using local biorthogonality, the following relation may be obtained

$$s_j^T A^{-1} r_j - s_{j+1}^T A^{-1} r_{j+1} = \alpha_j s_j^T r_j \tag{2.26}$$

where α_j is a scalar computed in the jth BiCG iteration. Finally, as i increases, the last term in Eq. 2.25 becomes small and can be neglected. Therefore, the bilinear form can be approximated as

$$(b, c)_A \approx \Big(c^T x_0 + y_0^T r_0 \Big) + \sum_{j=0}^{i-1} \alpha_j s_j^T r_j. \tag{2.27}$$

When approximating the bilinear form, the BiCG iteration is terminated when $||r_i||\ ||s_i|| \leq \kappa_{BF} ||r_0||\ ||s_0||$ where κ_{BF} is a defined convergence tolerance. At the conclusion of the BiCG iteration, the approximate solutions to the ETE and adjoint systems, x_j and y_j, respectively, provide a local discretization error estimate and approximate adjoint variables for formulating an adaptation indicator. The bilinear form approximation can be used as a functional error estimate. For more information, refer to Refs. [30, 32].

2.3.2.2 Preconditioning

Krylov methods often need a preconditioner for fast convergence. There exist many preconditioning techniques, such as Incomplete LU factorization (ILU) [33] and Sparse Approximate Inverse (SAI) [34, 35]. While ILU is arguably the most widely used preconditioning

technique, due to the low dimensionality of the problems examined in this work, the ILU preconditioner is very close to a true LU factorization and does not allow us to illustrate the benefits of the bilinear form approach to functional error estimation and adaptation. Therefore, SAI is instead used for preconditioning. In general, SAI is not as effective of a preconditioner compared to ILU, but it has recently gained popularity due to its favorable parallel properties [32, 36]. Since the inverse of a sparse matrix is typically dense, it is common to select an initial nonzero pattern for SAI. The nonzero pattern of the SAI preconditioner is chosen as the nonzero pattern of A unless otherwise specified. Furthermore, all preconditioners are applied from the right.

2.3.2.3 Grid Adaptation

Grid adaptation is performed using the mesh optimization algorithm outlined in Ref. [37] where cells are locally refined using the concept of error equidistribution and an *a priori* estimate of the convergence properties of the local error. At each adaptation cycle, a given cell is only allowed to be refined by one grid level. Adaptation indicators are formulated using the local contribution to the functional error estimate. Adaptation is terminated when $\frac{\varepsilon_{J_h}}{J_h(u_h)} < 1\text{E-}05$.

2.4 Test Cases

2.4.1 1D Viscous Burgers' Equation

The 1D viscous Burgers' equation is a simplified form of the Navier-Stokes equations and can be used to model the coalescence of two viscous shock waves. The 1D viscous Burgers' equation in strong, conservation form is given by

$$\frac{\partial u}{\partial t} + \frac{\partial}{\partial x}\left(\frac{u^2}{2}\right) = \nu \frac{\partial^2 u}{\partial x^2} \tag{2.28}$$

where u is velocity and ν is viscosity. A steady, exact solution to 1D viscous Burgers' equation is given by

$$u(x) = -u_{ref} \tanh\left[\frac{Re}{2}\frac{x}{L_{ref}}\right], \quad \forall x \in \left[-\frac{L_{ref}}{2}, +\frac{L_{ref}}{2}\right] \tag{2.29}$$

where u_{ref} is a reference velocity and $Re = \frac{u_{ref} L_{ref}}{\nu}$ is the Reynolds number. The reference quantities used in this work are $u_{ref} = 2\ m/s$, $Re = 32$, and $L_{ref} = 8\ m$. The exact solution described here may be seen in Figure 2.1.

The 1D viscous Burgers' equation is solved using a cell-centered, finite-volume discretization. The inviscid flux is computed using Roe's flux difference splitting scheme [38, 39]. Green-Gauss face gradients are used to compute the viscous flux [40]. Second-order spatial accuracy is obtained using MUSCL extrapolation [41]. The numerical solution is marched in pseudo-time to a steady-state using backward Euler time integration. Boundary conditions are applied by reconstructing the solution to the boundary face and applying the numerical flux function with the exact solution prescribed as the outer state.

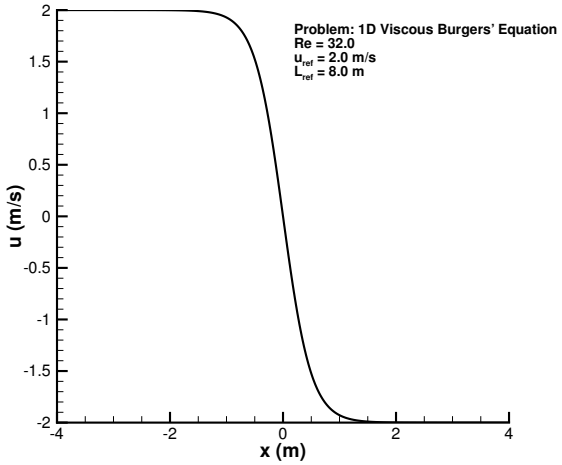


Figure 2.1: Exact solution for 1D viscous Burgers' equation.

In this work, three discrete solution functionals are used to test the proposed methods. The continuous and discrete forms of these functionals are given below.

	Continuous Functional	Discrete Functional	
Integral of Velocity :	$J(\tilde{u}) = \int_{-4}^{4} u^n \, dx$	$J_h(u_h) = \sum_{i=1}^{N_{cells}} u_i^n \Delta x_i$	$n \in \{2, 4, 6\}$

2.4.2 Quasi-1D Euler Equations

The quasi-1D Euler equations in strong, conservation form are

$$\frac{\partial(A\mathbf{u})}{\partial t} + \frac{\partial[A\mathbf{F}(\mathbf{u})]}{\partial x} = \mathbf{s}(\mathbf{u}) \tag{2.30}$$

where $\mathbf{u}$ is the vector of conserved variables, $\mathbf{F}(\mathbf{u})$ is the vector of inviscid fluxes, and $\mathbf{s}(\mathbf{u})$ is a source term. These equations are a statement of conservation of mass, momentum, and energy for an inviscid, adiabatic fluid with no body forces. The vector of conserved variables,

the inviscid fluxes, and the source term are given by

$$\mathbf{u} = \begin{bmatrix} \rho \\ \rho u \\ \rho e_t \end{bmatrix}, \quad \mathbf{F}(\mathbf{u}) = \begin{bmatrix} \rho u \\ \rho u^2 + p \\ \rho u h_t \end{bmatrix}, \quad \mathbf{s}(\mathbf{u}) = \begin{bmatrix} 0 \\ p\frac{dA}{dx} \\ 0 \end{bmatrix} \tag{2.31}$$

where ρ is the fluid density, u is the fluid velocity, p is the static pressure, e_t is the total energy, h_t is the total enthalpy, and $\frac{dA}{dx}$ describes how the cross-sectional area of the domain changes as a function of the spatial variable. The system of equations is closed using the equation of state for a perfect gas such that the total energy, e_t, and total enthalpy, h_t, are given by

$$e_t = \frac{p}{\rho(\gamma - 1)} + \frac{u^2}{2}, \quad h_t = \frac{\gamma p}{\rho(\gamma - 1)} + \frac{u^2}{2} \tag{2.32}$$

where $\gamma = 1.4$ is the ratio of specific heats for air. An exact solution to the quasi-1D Euler equations may be obtained using the following Area-Mach number relationship

$$\frac{A(x)}{A^*} = \frac{1}{M^2}\left[\frac{2}{\gamma + 1}\left(1 + \frac{\gamma - 1}{2}M^2\right)\right]^{\frac{\gamma+1}{\gamma-1}} \tag{2.33}$$

where $A(x)$ is the local cross-sectional area and A^* is the cross-sectional area corresponding to $M = 1$. Once the local Mach number is known, the flow state may be determined using the local stagnation conditions.

The quasi-1D Euler equations are discretized using a second-order, cell-centered, finite-volume discretization. The inviscid fluxes are computed using Roe's flux difference splitting scheme [38]. To obtain second-order spatial accuracy, MUSCL extrapolation [41] is performed in physical space to reconstruct the primitive variables to the face. Boundary conditions are enforced weakly through the fluxes. The quasi-1D Euler equations are coded in a nondimensional form to improve the scaling of the residual Jacobian matrix [23]. The nondimensionalization used in this work may be found in Appendix A. To illustrate the numerical impact of properly scaling the equations, the condition number of the residual Jacobian matrix for a range of uniform grids is included in Appendix B.

2.4.2.1 Quasi-1D Nozzle

The quasi-1D Euler equations are used to model flow through a nozzle. The nozzle is characterized by a contraction through which the flow accelerates followed by a diverging section, which allows the flow to expand subsonically or supersonically depending upon the pressure at the nozzle exit. The geometry of the nozzle is taken from Jackson and Roy [42] and has a Gaussian area distribution given by

$$A(x) = 1 - 0.8e^{\left(\frac{-x^2}{2\sigma^2}\right)}, \quad x \in [-1, 1] \tag{2.34}$$

where σ is chosen as 0.2. Two flow conditions are examined: (1) isentropic, subsonic flow through the diverging section and (2) isentropic, supersonic flow through the diverging section. The area distribution and Mach number distributions for both the subsonic and supersonic cases can be seen in Figure 2.2.

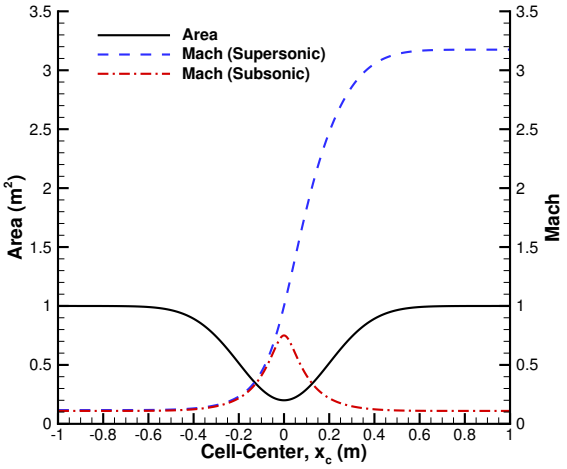


Figure 2.2: Gaussian Nozzle: Mach number and area distributions.

At the inflow of the nozzle, the stagnation pressure and temperature are fixed at 300 kPa and 600 K, respectively. The Mach number is extrapolated from the interior to set the inflow state at the face. The outflow boundary condition depends upon the local character of the flow. For supersonic flow, all variables in the interior are extrapolated to the face. For subsonic flow, a back pressure is set, and velocity and density are extrapolated from the interior. The back pressure for all subsonic results is set to 297.485 kPa.

In this work, five discrete solution functionals are used to test the proposed methods. The continuous and discrete forms of each functional are given below.

	Continuous Functional	Discrete Functional
Integral of Pressure :	$J(\tilde{u}) = \int_{-1}^{1} p \, dx$	$J_h(u_h) = \sum_{i=1}^{N_{cells}} p_i \Delta x_i$
Integral of Entropy :	$J(\tilde{u}) = \int_{-1}^{1} \log\left(\frac{p}{\rho^\gamma}\right) dx$	$J_h(u_h) = \sum_{i=1}^{N_{cells}} \log\left(\frac{p_i}{\rho_i^\gamma}\right) \Delta x_i$
Total Mass :	$J(\tilde{u}) = \int_{-1}^{1} \rho A \, dx$	$J_h(u_h) = \sum_{i=1}^{N_{cells}} \rho_i A_i \Delta x_i$
Integral of Mass Flow :	$J(\tilde{u}) = \int_{-1}^{1} \rho u A \, dx$	$J_h(u_h) = \sum_{i=1}^{N_{cells}} \rho_i u_i A_i \Delta x_i$
Static Thrust :	$J(\tilde{u}) = \rho_e u_e^2 A_e + (p_e - p_a) A_e$	$J_h(u_h) = \rho_e u_e^2 A_e + (p_e - p_a) A_e$

For the static thrust functional, the ambient pressure, p_a, is selected as $p_a = 5.5$ kPa for the supersonic case so that the nozzle is operating at an underexpanded condition. For the subsonic case, the ambient pressure is set to the previously specified back pressure.

2.5 Results & Discussion

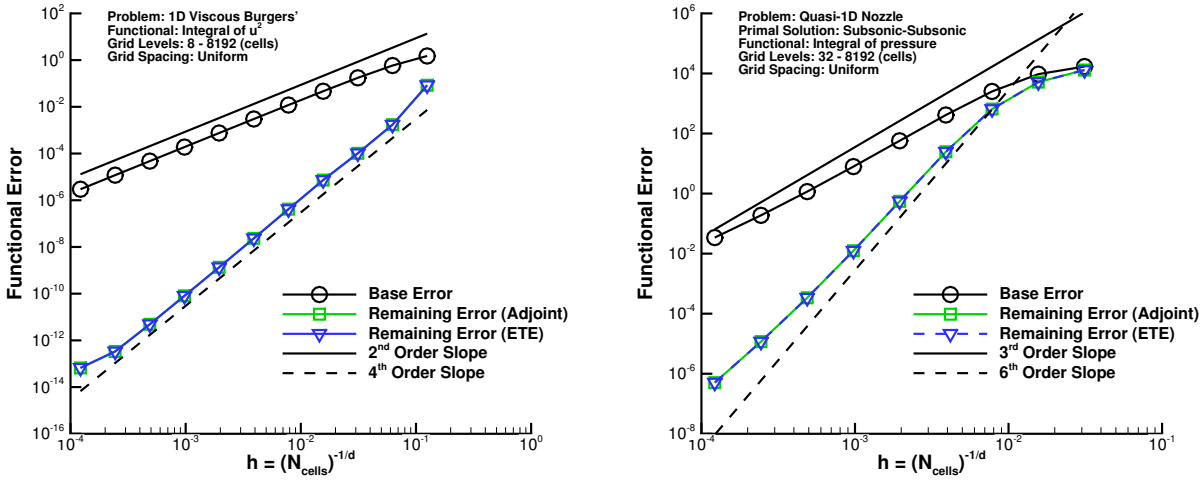
In this section, results are presented for 1D viscous Burgers' equation and the quasi-1D nozzle problem, which demonstrate the benefits of the proposed methods over a traditional adjoint approach.

2.5.1 Error Estimation for Multiple Functionals

The ETE and adjoint problems are solved on a series of systematically refined, uniformly spaced grids. The solution to each system is obtained using the pseudo-time-stepping techniques outlined in Section 2.2.2 and Section 2.2.3. To illustrate the equivalence of the adjoint and ETE approaches for functional error estimation, the base functional error and the remaining error after correction are plotted in Figure 2.3a and Figure 2.3b for Burgers' equation and the quasi-1D nozzle problem, respectively. The functional of interest for Burgers' equation is the integral of u^2 across the entire domain. Likewise, the functional of interest for the quasi-1D nozzle is the integral of pressure along the nozzle. Quasi-1D nozzle results presented in this section are only for the subsonic case. Similar results were obtained for the supersonic case. Figure 2.3 illustrates that the ETE are able to obtain the same functional correction as a standard adjoint approach. The differences between the adjoint and ETE results are indistinguishable, and, in fact, are on the order of machine precision. Upon application of the functional error estimate as a correction, the functional converges to the true value at a higher-order rate. For Burgers' equation, the corrected functional converges at a 4^{th}-order rate. While the formal order of the discretization for the quasi-1D nozzle problem is $p_f = 2$, the base error converges at approximately a 3^{rd}-order rate and the remaining error converges at approximately a 6^{th}-order rate. This behavior was also observed in Ref. [4] for a similar subsonic test case and can likely be attributed to error cancellation. For the supersonic case, the base error and remaining error converge at the expected rates, 2^{nd}-order and 4^{th}-order, respectively. The corresponding adjoint solutions for Burgers' equation and quasi-1D nozzle may be found in Figure 2.4a and Figure 2.4b, respectively.

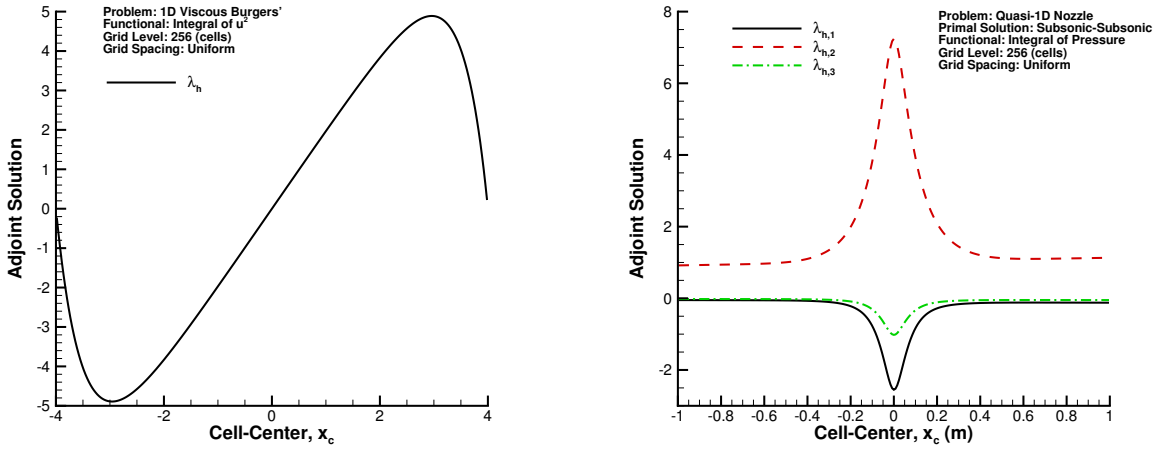
One of the benefits of the ETE approach to functional error estimation is that, in addition to functional error estimates, a local discretization error estimate is obtained everywhere in the domain. Plots of the exact discretization error and the ETE discretization error estimate for both Burgers' equation and quasi-1D nozzle may be found in Figure 2.5 for a 256 cell grid. For quasi-1D nozzle, discretization error is only plotted for pressure. Qualitatively similar results were found for the other primitive variables. For Burgers' equation, the maximum

relative difference between the ETE estimate and the exact discretization error is 0.73%. Similarly, the maximum relative difference between the error estimate and the true error for quasi-1D nozzle is 7.21%. As illustrated in Figure 2.5, the ETE are able to provide accurate estimates of discretization error for both test cases examined.



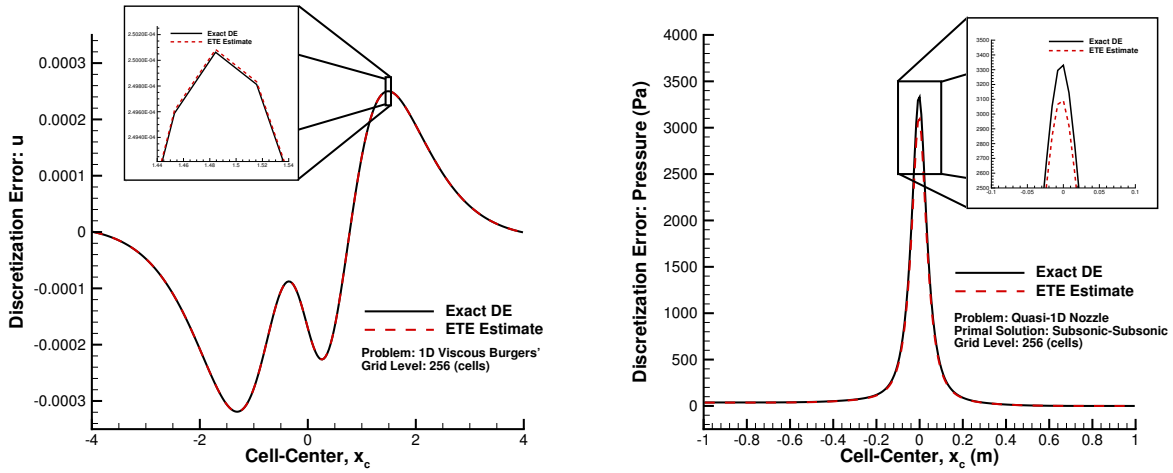
(a) 1D viscous Burgers' equation (Integral of u^2) (b) Quasi-1D nozzle (Integral of pressure)

Figure 2.3: Illustration of adjoint / ETE equivalence for functional error estimation.



(a) 1D viscous Burgers' equation (Integral of u^2) (b) Quasi-1D nozzle (Integral of Pressure)

Figure 2.4: Adjoint solutions for 1D viscous Burgers' equation and the quasi-1D nozzle problem.



(a) 1D viscous Burgers' equation

(b) Quasi-1D nozzle

Figure 2.5: Comparisons of ETE discretization error estimates and exact discretization error.

In addition to a local discretization error estimate, another benefit of the ETE approach to functional error estimation is that it is computationally more efficient than a standard adjoint when multiple functional error estimates are required. Functional error estimates are computed for all functionals given in Section 2.4. The number of functionals examined for Burgers' equation and quasi-1D nozzle are 3 and 5, respectively. Computational efficiency is assessed by monitoring the solution times of both the ETE and adjoint approaches. When monitoring programs with short runtimes, background processes can introduce a certain amount of variability into the timing results. To minimize this effect, runtimes are averaged over 50 separate runs. Results are reported in Table 2.1. The speed-up of the ETE approach, defined as the ratio of the adjoint runtime to the ETE runtime, is also reported. Simulations are performed on a Dell 5810 workstation with an Intel Xeon E5-1650 3.60 GHz processor. To properly compare the performance of each approach, shared computations such as residual Jacobian setup, preconditioning, and post-processing were excluded from the reported runtimes. As illustrated in Table 2.1, significant computational savings can be achieved with the ETE approach to functional error estimation. For Burgers' equation, the average speed-up over all grid levels is 2.3x for 3 functionals. The average speed-up for quasi-1D nozzle is 5.3x for 5 functionals. With the ETE approach, the computational efficiency of functional error estimation is drastically improved without any loss of accuracy.

Table 2.1: Runtime comparison of adjoint and ETE solves for multiple functionals.

(a) 1D viscous Burgers' equation

N_{cells}	Adjoint (sec)	ETE (sec)	Speed-up
8	5.9106E-04	4.5400E-04	1.3019
16	9.4890E-04	2.7870E-04	3.4047
32	9.2754E-04	4.8494E-04	1.9128
64	9.9090E-04	4.8666E-04	2.0361
128	1.1179E-03	5.0878E-04	2.1972
256	1.4337E-03	6.6222E-04	2.1651
512	1.9933E-03	1.1265E-03	1.7695
1024	2.9273E-03	1.2602E-03	2.3229
2048	5.0750E-03	1.8878E-03	2.6883
4096	1.2392E-02	4.2653E-03	2.9053

(b) Quasi-1D nozzle

N_{cells}	Adjoint (sec)	ETE (sec)	Speed-up
32	6.1352E-03	1.4508E-03	4.2288
64	9.5558E-03	2.0927E-03	4.5563
128	2.1322E-02	4.1929E-03	5.0853
256	4.9169E-02	9.3389E-03	5.2650
512	1.4120E-01	2.3905E-02	5.9067
1024	5.7255E-01	9.7009E-02	5.9020
2048	1.9735E+00	3.2249E-01	6.1198
4096	8.2946E+00	1.2351E+00	6.7157

2.5.2 Approximate Adjoints for Functional Error Estimation and Adaptation

The Krylov subspace method outlined in Section 2.3.2 is applied to the quasi-1D nozzle problem for a series of systematically refined, uniformly spaced grids ranging from 32 - 1024 cells. BiCG is tested using three bilinear form convergence tolerances: $\kappa_{BF} \in \{\text{1E-01}, \text{1E-02}, \text{1E-03}\}$. These tolerances are selected to assess how the accuracy of the bilinear form solve impacts the accuracy of the adjoint solution, the local discretization error estimate, and the functional error estimate. SAI is used to precondition each bilinear form solve. The L_2-norm of the ETE residual, $||Ax - b||_2$, the L_2-norm of the adjoint residual, $||A^T y - c||_2$, and the relative error in the bilinear form approximation (i.e., functional error estimate),

$$\varepsilon_{BF} = \frac{\left|(b,c)_A^{approx} - (b,c)_A^{exact}\right|}{\left|(b,c)_A^{exact}\right|}, \tag{2.35}$$

are monitored and reported in Table 2.2 for each bilinear form tolerance. All results in this section are presented for the supersonic case of the quasi-1D nozzle; similar results were found for the subsonic case. The functional of interest is the integral of pressure along the nozzle.

Table 2.2: Comparison of ETE residual errors, adjoint residual errors, and relative errors in the bilinear form approximation.

(a) κ_{BF} = 1e-1

N_{cells}	$\|\|Ax-b\|\|_2$	$\|\|A^Ty-c\|\|_2$	ε_{BF}
32	9.16E-02	3.49E-01	2.40E+00
64	2.18E-02	4.00E-01	6.71E-01
128	3.17E-03	2.16E-01	9.00E-01
256	1.81E-04	9.79E-02	8.45E-06
512	1.72E-04	9.95E-02	5.75E-01
1024	2.19E-05	3.76E-02	1.48E-04

(b) κ_{BF} = 1e-2

N_{cells}	$\|\|Ax-b\|\|_2$	$\|\|A^Ty-c\|\|_2$	ε_{BF}
32	4.27E-02	1.04E-01	5.35E-01
64	2.05E-03	1.11E-01	2.53E-03
128	1.40E-04	6.38E-02	4.64E-04
256	1.41E-04	1.16E-02	4.27E-08
512	3.98E-05	1.11E-02	9.48E-05
1024	8.82E-06	1.13E-02	1.48E-04

(c) κ_{BF} = 1e-3

N_{cells}	$\|\|Ax-b\|\|_2$	$\|\|A^Ty-c\|\|_2$	ε_{BF}
32	1.04E-02	2.84E-02	2.58E-05
64	5.10E-04	6.50E-02	2.99E-05
128	8.61E-05	2.50E-02	4.60E-04
256	1.83E-05	2.28E-03	5.76E-08
512	1.15E-05	5.56E-03	9.48E-05
1024	3.68E-06	4.41E-03	1.48E-04

For each bilinear form tolerance, the ETE residual errors are small and do not vary significantly as the tolerance is decreased. Therefore, even for modest convergence of the bilinear form, a relatively accurate discretization error estimate can still be obtained. On the other hand, the adjoint residual errors are on average 2 - 3 orders of magnitude less accurate than the ETE residual errors. We postulate that the accuracy of the approximate adjoint solution is still sufficient for driving an adaptation procedure. This theory will be explored later in

this section. As expected, decreasing the bilinear form tolerance increases the accuracy of the bilinear form approximation. We believe that the bilinear form approximation is likely accurate enough if the relative error is less than 1%. To confirm this tolerance, further study is required with more complex problems. For the quasi-1D nozzle problem, a relative error less than 1% is achieved for all grid levels with a tolerance of κ_{BF} = 1E-03. To better demonstrate the impact of κ_{BF} on the accuracy of the bilinear form approximation, the bilinear form approximation is applied as a correction to the functional. The remaining error in the functional is compared to that obtained with an exact adjoint solution in Figure 2.6. For this problem, a bilinear form tolerance of κ_{BF} = 1E-03 is sufficient to accurately estimate functional error over the full range of grid levels. It should also be noted that the erratic behavior of ε_{BF}, and similarly, the remaining error, for a given κ_{BF} can be attributed to the characteristically non-smooth convergence of BiCG.

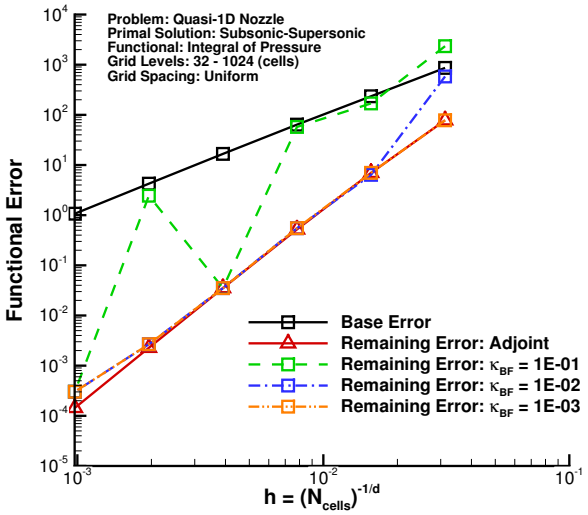


Figure 2.6: Bilinear form functional error convergence.

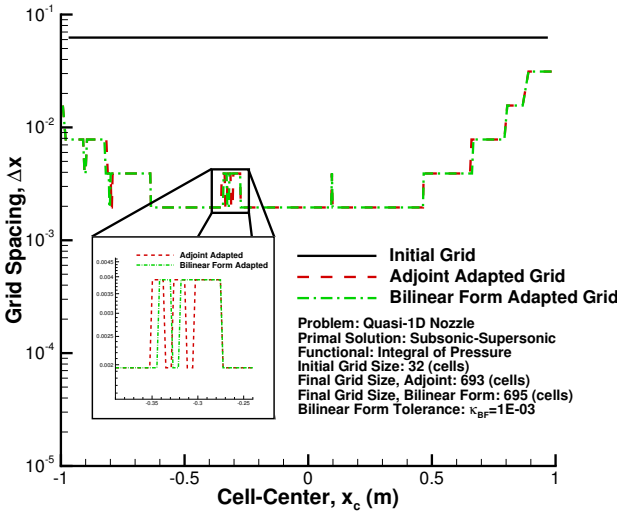


Figure 2.7: Comparison of final adapted grids.

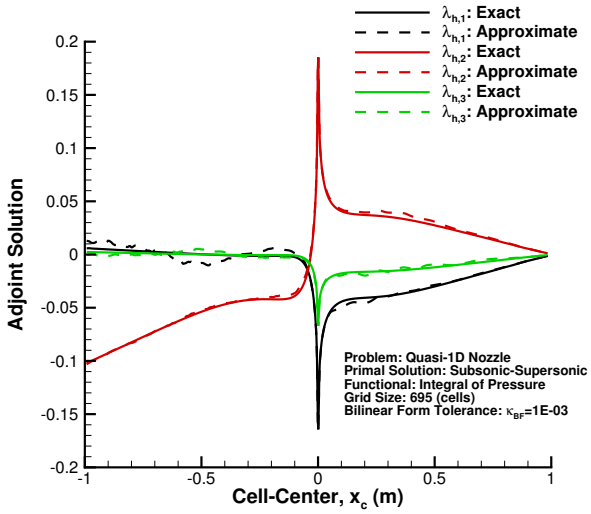


Figure 2.8: Comparison of adjoint solutions.

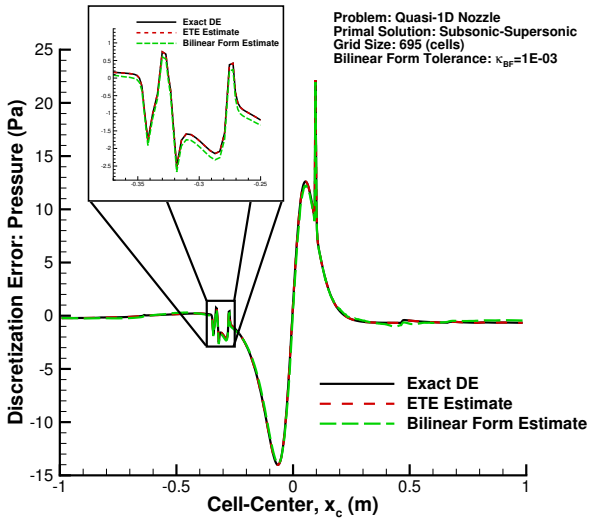


Figure 2.9: Comparison of pressure discretization error.

To demonstrate the effectiveness of the bilinear form approach for driving an adaptation procedure, grid adaptation is performed on an initially uniformly spaced 32 cell grid. Adaptation indicators are formulated using an exact adjoint solution as well as an approximate adjoint solution obtained via a bilinear form solve with a tolerance of $\kappa_{BF} = 1\text{E-}03$. Due to the numerical stiffness introduced by grid nonuniformities, the bilinear form solver does not converge with the initial sparsity pattern selected for the SPAI preconditioner. To ensure convergence of the bilinear form solve throughout the adaptation process, the sparsity pattern of the SPAI preconditioner is increased to twice that of the original matrix. The initial grid and the adapted grids produced by each adjoint solution are plotted in Figure 2.7 as cell size versus cell-center. The final adapted grids are qualitatively the same. Final grid sizes are 693 cells and 695 cells for exact and approximate adjoints, respectively. The majority of grid refinement takes place immediately upstream and downstream of the throat with the grid steadily coarsening in the aft section of the nozzle. The approximate adjoint and ETE solutions on the final adapted grid are compared to the exact adjoint and ETE solutions in Figure 2.8 and Figure 2.9. The approximate adjoint solution is able to capture the primary features of the exact adjoint solution, especially the $\log(x)$ singularity that occurs at the throat. Similarly, the bilinear form estimate of discretization error is qualitatively very accurate in comparison to the exact discretization error and the ETE estimate. It should be noted that local spikes in discretization error along the nozzle are due to the non-smooth variation in cell size in those regions.

2.6 Conclusions

In this work, new approaches to functional error estimation and adaptation have been presented. Instead of using a conventional adjoint method, functional error estimates are computed using local discretization error estimates obtained via an error transport equation (ETE). This approach is shown to be mathematically equivalent to a discrete adjoint method, but is much more efficient in the case where multiple functional error estimates are required. Computational speed-ups on the order of n adjoint solves have been observed, where n is the number of functionals of interest. A disadvantage of the ETE approach to functional error estimation is that, if adjoint-based adaptation is to be performed, the adjoint variables are no longer available for formulating adaptation indicators. To address this issue, a technique based on Krylov subspace methods has been proposed which can simultaneously compute an accurate functional error estimate and sufficiently accurate adjoint and ETE solutions. While this technique only solves the adjoint and ETE problems in an approximate sense, it has been shown that the accuracy of the resulting adjoint and ETE solutions is still adequate for driving an adaptation procedure and gaining insight into the variation of errors across the domain.

Acknowledgments

Work by W. Tyson, J. Derlaga (while at Virginia Tech), and C. Roy was supported by the Collaborative Center for Aeronautical Sciences with Dr. John Benek of the Air Force Research Laboratory in Dayton, Ohio serving as the program manager. Work by K. Świrydowicz and E. de Sturler was supported by NSF Grant 1217156.

Bibliography

[1] M. B. Giles, N. A. Pierce, Analytic adjoint solutions for the quasi-one-dimensional Euler equations, Journal of Fluid Mechanics 426 (2001) 327 – 345. doi:10.1017/s0022112000002366.
URL http://dx.doi.org/10.1017/S0022112000002366

[2] M. Giles, M. Duta, J.-D. Müller, N. Pierce, Algorithm developments for discrete adjoint methods, AIAA Journal 41 (2) (2003) 198 – 205. doi:10.2514/2.1961.
URL http://dx.doi.org/10.2514/2.1961

[3] M. B. Giles, N. Pierce, E. Süli, Progress in adjoint error correction for integral functionals, Computing and Visualization in Science 6 (2-3) (2004) 113 – 121. doi:10.1007/s00791-003-0115-y.
URL http://dx.doi.org/10.1007/s00791-003-0115-y

[4] D. A. Venditti, D. L. Darmofal, Adjoint error estimation and grid adaptation for functional outputs: Application to quasi-one-dimensional flow, Journal of Computational Physics 164 (1) (2000) 204 – 227. doi:10.1006/jcph.2000.6600.
URL http://dx.doi.org/10.1006/jcph.2000.6600

[5] D. A. Venditti, D. L. Darmofal, Grid adaptation for functional outputs: Application to two-dimensional inviscid flows, Journal of Computational Physics 176 (1) (2002) 40 – 69. doi:10.1006/jcph.2001.6967.
URL http://dx.doi.org/10.1006/jcph.2001.6967

[6] D. A. Venditti, D. L. Darmofal, Anisotropic grid adaptation for functional outputs: Application to two-dimensional viscous flows, Journal of Computational Physics 187 (1) (2003) 22 – 46. doi:10.1016/s0021-9991(03)00074-3.
URL http://dx.doi.org/10.1016/S0021-9991(03)00074-3

[7] M. Park, Adjoint-based, three-dimensional error prediction and grid adaptation, in: 32nd AIAA Fluid Dynamics Conference and Exhibit, American Institute of Aeronautics and Astronautics (AIAA), 2002. doi:10.2514/6.2002-3286.
URL http://dx.doi.org/10.2514/6.2002-3286

[8] M. Park, B. Lee-Rausch, C. Rumsey, FUN3D and CFL3D computations for the first high lift prediction workshop, in: 49th AIAA Aerospace Sciences Meeting including the New Horizons Forum and Aerospace Exposition, American Institute of Aeronautics and Astronautics (AIAA), 2011. doi:10.2514/6.2011-936.
URL http://dx.doi.org/10.2514/6.2011-936

[9] K. Fidkowski, P. Roe, Entropy-based mesh refinement, I: The entropy adjoint approach, in: 19th AIAA Computational Fluid Dynamics Conference, American Institute of Aeronautics and Astronautics (AIAA), 2009. doi:10.2514/6.2009-3790.
URL http://dx.doi.org/10.2514/6.2009-3790

[10] J. M. Derlaga, Application of improved truncation error estimation techniques to adjoint based error estimation and grid adaptation, Ph.D. thesis, Virginia Polytechnic Institute and State University, Blacksburg, VA (Jul. 2015).
URL http://hdl.handle.net/10919/54592

[11] C. Roy, Strategies for driving mesh adaptation in CFD (invited), in: 47th AIAA Aerospace Sciences Meeting including the New Horizons Forum and Aerospace Exposition, American Institute of Aeronautics and Astronautics (AIAA), 2009. doi:10.2514/6.2009-1302.
URL http://dx.doi.org/10.2514/6.2009-1302

[12] W. L. Oberkampf, C. J. Roy, Verification and Validation in Scientific Computing, Cambridge University Press (CUP), 2010. doi:10.1017/cbo9780511760396.
URL http://dx.doi.org/10.1017/CBO9780511760396

[13] A. Choudhary, C. Roy, Efficient residual-based mesh adaptation for 1D and 2D CFD applications, in: 49th AIAA Aerospace Sciences Meeting including the New Horizons Forum and Aerospace Exposition, American Institute of Aeronautics and Astronautics (AIAA), 2011. doi:10.2514/6.2011-214.
URL http://dx.doi.org/10.2514/6.2011-214

[14] D. Mavriplis, On convergence acceleration techniques for unstructured meshes, in: 29th AIAA Fluid Dynamics Conference, American Institute of Aeronautics and Astronautics (AIAA), 1998, p. 2966. doi:10.2514/6.1998-2966.
URL https://doi.org/10.2514/6.1998-2966

[15] A. Hay, M. Visonneau, Error estimation using the error transport equation for finite-volume methods and arbitrary meshes, International Journal of Computational Fluid Dynamics 20 (7) (2006) 463 – 479. doi:10.1080/10618560600835934.
URL http://dx.doi.org/10.1080/10618560600835934

[16] T. Phillips, J. M. Derlaga, C. J. Roy, J. Borggaard, Finite volume solution reconstruction methods for truncation error estimation, in: 21st AIAA Computational Fluid Dynamics Conference, American Institute of Aeronautics and Astronautics (AIAA), 2013. doi:10.2514/6.2013-3090.
URL http://dx.doi.org/10.2514/6.2013-3090

[17] T. Phillips, C. Roy, Residual methods for discretization error estimation, in: 20th AIAA Computational Fluid Dynamics Conference, American Institute of Aeronautics and Astronautics (AIAA), 2011. doi:10.2514/6.2011-3870.
URL http://dx.doi.org/10.2514/6.2011-3870

[18] Y. Saad, M. H. Schultz, GMRES: A generalized minimal residual algorithm for solving nonsymmetric linear systems, SIAM Journal on Scientific and Statistical Computing 7 (3) (1986) 856 – 869. doi:10.1137/0907058.
URL http://dx.doi.org/10.1137/0907058

[19] A. Jameson, Aerodynamic design via control theory, Journal of Scientific Computing 3 (3) (1988) 233 – 260. doi:10.1007/bf01061285.
URL http://dx.doi.org/10.1007/BF01061285

[20] M. Nemec, M. Aftosmis, Adjoint error estimation and adaptive refinement for embedded-boundary Cartesian meshes, in: 18th AIAA Computational Fluid Dynamics Conference, American Institute of Aeronautics and Astronautics (AIAA), 2007. doi:10.2514/6.2007-4187.
URL http://dx.doi.org/10.2514/6.2007-4187

[21] B. A. Rothacker, M. Ceze, K. Fidkowski, Adjoint-based error estimation and mesh adaptation for problems with output constraints, in: 32nd AIAA Applied Aerodynamics Conference, American Institute of Aeronautics and Astronautics (AIAA), 2014. doi:

10.2514/6.2014-2576.
URL http://dx.doi.org/10.2514/6.2014-2576

[22] E. J. Nielsen, J. Lu, M. A. Park, D. L. Darmofal, An implicit, exact dual adjoint solution method for turbulent flows on unstructured grids, Computers & Fluids 33 (9) (2004) 1131 – 1155. doi:10.1016/j.compfluid.2003.09.005.
URL http://dx.doi.org/10.1016/j.compfluid.2003.09.005

[23] W. C. Tyson, K. Swirydowicz, J. M. Derlaga, C. J. Roy, E. de Sturler, Improved functional-based error estimation and adaptation without adjoints, in: 46th AIAA Fluid Dynamics Conference, American Institute of Aeronautics and Astronautics (AIAA), 2016. doi:10.2514/6.2016-3809.
URL http://dx.doi.org/10.2514/6.2016-3809

[24] G. H. Golub, M. Stoll, A. Wathen, Approximation of the scattering amplitude and linear systems, Electronic Transactions on Numerical Analysis 31 (2008) 178 – 203.

[25] M. B. Giles, N. A. Pierce, An introduction to the adjoint approach to design, Flow, Turbulence and Combustion 65 (3–4) (2000) 393 – 415. doi:10.1023/a:1011430410075.
URL http://dx.doi.org/10.1023/A:1011430410075

[26] A. Ramm, Symmetry properties of scattering amplitudes and applications to inverse problems, Journal of Mathematical Analysis and Applications 156 (2) (1991) 333 – 340. doi:10.1016/0022-247x(91)90401-k.
URL http://dx.doi.org/10.1016/0022-247X(91)90401-K

[27] C. Lanczos, Solution of systems of linear equations by minimized iterations, Journal of Research of the National Bureau of Standards 49 (1) (1952) 33 – 53. doi:10.6028/jres.049.006.
URL http://dx.doi.org/10.6028/jres.049.006

[28] R. Fletcher, Conjugate gradient methods for indefinite systems, in: G. Watson (Ed.), Numerical Analysis 1975, Proceedings of the Dundee Conference on Numerical Analysis, no. 506, Springer Science + Business Media, 1976, pp. 73 – 89. doi:10.1007/bfb0080116.
URL http://dx.doi.org/10.1007/BFb0080116

[29] C. C. Paige, M. A. Saunders, LSQR: An algorithm for sparse linear equations and sparse least squares, ACM transactions on mathematical software 8 (1) (1982) 43–71. doi:10.1145/355984.355989.

[30] Z. Strakoš, P. Tichý, On efficient numerical approximation of the bilinear form $c^*A^{-1}b$, SIAM Journal on Scientific Computing 33 (2) (2011) 565 – 587. doi:10.1137/090753723.
URL http://dx.doi.org/10.1137/090753723

[31] G. H. Golub, G. Meurant, Matrices, moments and quadrature, in: D. Griffiths, G. Watson (Eds.), Numerical Analysis 1993, Proceedings of the 15th Dundee Conference, Vol. 303, 1994, pp. 105 – 156.

[32] K. Świrydowicz, Strategies for recycling Krylov subspace methods and bilinear form estimation, Ph.D. thesis, Virginia Polytechnic Institute and State University (Jul. 2017).

[33] Y. Saad, Iterative Methods for Sparse Linear Systems, Society for Industrial and Applied Mathematics, 2003. doi:10.1137/1.9780898718003.
URL http://dx.doi.org/10.1137/1.9780898718003

[34] M. Benson, P. Frederickson, Iterative solution of large sparse linear systems arising in certain multidimensional approximation problems, Utilitas Math 22 (127 - 140) (1982) 154 – 155.

[35] M. J. Grote, T. Huckle, Parallel preconditioning with sparse approximate inverses, SIAM Journal on Scientific Computing 18 (3) (1997) 838 – 853. doi:10.1137/s1064827594276552.
URL http://dx.doi.org/10.1137/S1064827594276552

[36] M. M. Dehnavi, D. M. Fernandez, J. Gaudiot, D. D. Giannacopoulos, Parallel sparse approximate inverse preconditioning on graphic processing units, IEEE Transactions on Parallel and Distributed Systems 24 (9) (2013) 1852 – 1862. doi:10.1109/tpds.2012.286.
URL http://dx.doi.org/10.1109/TPDS.2012.286

[37] K. Fidkowski, A simplex cut-cell adaptive method for high-order discretizations of the compressible Navier-Stokes equations, Ph.D. thesis, Massachusetts Institute of Technology (Jun. 2007).

[38] P. Roe, Approximate Riemann solvers, parameter vectors, and difference schemes, Journal of Computational Physics 43 (2) (1981) 357 – 372. doi:10.1016/0021-9991(81)90128-5.
URL http://dx.doi.org/10.1016/0021-9991(81)90128-5

[39] P. L. Roe, The use of the Riemann problem in finite difference schemes, in: W. C. Reynolds, R. W. MacCormack (Eds.), Seventh International Conference on Numerical Methods in Fluid Dynamics, Springer Berlin Heidelberg, Berlin, Heidelberg, 1981, pp. 354–359. doi:https://doi.org/10.1007/3-540-10694-4_54.

[40] J. Blazek, Computational Fluid Dynamics - Principles and Applications, 2nd Edition, Elsevier Science Publishing Co., Inc., 2001.

[41] B. van Leer, Towards the ultimate conservative difference scheme. V. a second-order sequel to Godunov's method, Journal of Computational Physics 32 (1) (1979) 101 –

136. doi:10.1016/0021-9991(79)90145-1.
URL http://dx.doi.org/10.1016/0021-9991(79)90145-1

[42] C. W. Jackson, C. J. Roy, A multi-mesh CFD technique for adaptive mesh solutions, in: 53rd AIAA Aerospace Sciences Meeting, American Institute of Aeronautics and Astronautics (AIAA), 2015. doi:10.2514/6.2015-1958.
URL http://dx.doi.org/10.2514/6.2015-1958

Chapter 3

A Higher-Order Error Estimation Framework for Finite-Volume CFD

William C. Tyson,[a] Christopher J. Roy[a]

[a] Kevin T. Crofton Department of Aerospace and Ocean Engineering, Virginia Tech
215 Randolph Hall, 460 Old Turner Street, Blacksburg, VA, 24061

Keywords: Computational fluid dynamics, Error transport equations, Discrete adjoint method, Discretization error estimation, Truncation error estimation, Finite-volume method

Attribution

- William C. Tyson (First Author): The first author served as the main contributor and primary author of the work. The first author implemented the primal and error transport solvers for the quasi-1D nozzle problem. Additionally, the first author implemented all routines for TE estimation and performed all numerical studies.
- Christopher J. Roy (Second Author): The second author provided valuable guidance and insight during the course of the study. The second author also provided helpful comments on the manuscript.

Abstract

Computational fluid dynamics is an invaluable tool for both the design and analysis of aerospace vehicles. Reliable error estimation techniques are needed to ensure that simula-

tion results are accurate enough to be used in engineering decision-making processes. In this work, a framework for estimating error and improving solution accuracy is presented. A linearized error transport equation (ETE) is used to estimate local discretization errors. A truncation error estimation technique is proposed which combines aspects of higher-order residual methods and continuous residual methods. The equivalence between adjoint and ETE methods for functional error estimation is demonstrated. Using adjoint/ETE equivalence, the higher-order properties of adjoint methods are extended to ETE methods. Consequently, ETE error estimates are shown to converge to the true discretization error at a higher-order rate. ETE error estimates are then used to correct the entire primal solution, and by extension, all output functionals, to higher order. The computational advantages of this ETE approach are discussed. Results are presented for grids with smoothly varying and non-smoothly varying grid metrics.

3.1 Introduction

In recent years, computational fluid dynamics (CFD) has been extensively employed in the design of aerospace vehicles with a growing trend toward CFD-based design. This increased reliance upon CFD necessitates the development of accurate and robust error estimation techniques to avoid making decisions based on inaccurate information. The dominant errors in any given CFD simulation are typically modeling errors and numerical errors, with the latter being the focus of this paper. Numerical errors are generated through a variety of mechanisms including finite-precision arithmetic, incomplete convergence of iterative procedures, and discretization of the governing equations. With advances in numerical algorithms and modern computing architectures, the errors associated with discretization, or discretization errors, are often the primary source of numerical errors.

Discretization error can be difficult to quantify as the generation, transport, and dissipation of these errors is a highly nonlinear function of the mesh and flow solution. Historically, discretization error has been estimated using Richardson extrapolation [1] which establishes an estimate using discrete solutions from multiple systematically-refined grid levels. While Richardson extrapolation is non-intrusive to existing flow solvers and can easily be performed as a post-processing step, it is often not practical for large problems as solving on a family of grids can be computationally expensive. Also, asymptotic grid convergence for all grid levels is necessary to obtain an accurate error estimate. This requirement can be challenging to meet especially for complex flow fields [2, 3]. Alternatively, defect correction [4, 5, 6, 7, 8] provides an error estimate by re-solving the governing equations with a truncation error estimate added as a source term. In this way, the discrete solution is driven toward the exact solution to the PDEs. The difference between discrete solutions can then be used as an error estimate. Although defect correction is generally less expensive than Richardson extrapolation, a significant computational cost is incurred from the need to re-solve the original nonlinear problem, albeit with a better initial guess. More recently, advancements

have been made with the introduction of adjoint methods [9, 10, 11, 12, 13] and error transport equation (ETE) methods [14, 15, 16]. Adjoint methods provide an error estimate and adaptation indicator for any output functional of interest. However, adjoint methods can be prohibitively expensive when multiple functionals are required, since an adjoint solution must be obtained for each functional of interest. Adjoint methods also do not provide local discretization error estimates. ETE methods, on the other hand, can simultaneously provide local discretization error estimates and error estimates for multiple functionals due to a certain equivalence that exists between adjoint methods and ETE methods [17, 18].

The vast majority of error estimation techniques currently available require an accurate estimate of truncation error. Truncation error, which is defined as the difference between the continuous and discrete governing equations, acts as the source term for the generation, transport, and diffusion of discretization errors. Common truncation error estimation methods include traditional analysis via operator expansion [14], embedded grid methods [19], higher-order residual methods [20], and continuous residual methods [21]. Once an accurate truncation error estimate has been computed, functional errors and discretization errors may be estimated. Under certain conditions, error estimates can be obtained which converge to the true error at a faster rate than the primal order of accuracy. This has been exploited within the adjoint community in regards to functional error estimates [11, 22]. Within the ETE community, Hay and Visonneau [23] as well as Banks et al. [24] observed higher-order (i.e., greater than second-order accurate) error estimates for the continuous implementation of the ETE. Similarly, Yan and Ollivier-Gooch [25] obtained higher-order error estimates using the discrete ETE on unstructured grids, albeit with a higher-order discretization of the ETE.

In this work, a general framework is presented for estimating discretization error and improving solution accuracy. A truncation error estimation technique is presented which combines aspects of both higher-order residual methods and continuous residual methods. In particular, the importance of enforcing boundary conditions on the reconstruction is illustrated. Furthermore, we extend the higher-order properties of discrete adjoint methods to discrete ETE methods using adjoint/ETE equivalency [18]. In order to improve the computational efficiency of the ETE, the ETE are discretized at the same order as the primal problem. For "structured grids" (i.e., grids with smoothly varying grid metrics), ETE error estimates are shown to converge to the true discretization error at a higher-order rate. While higher-order properties are not observed for "unstructured grids" (i.e., grids without smoothly varying grid metrics), we do provide suggestions for recovering higher-order properties. Similar to an adjoint method for output functionals, we apply the ETE error estimate as a correction to the primal solution in order to increase the observed order of accuracy. Traditionally, higher-order primal solutions are computed using a higher-order discretization. In comparison to a conventional lower-order method, higher-order discretizations are computationally more expensive and generally have more challenging stability issues. By using an ETE error estimate to correct the lower-order solution to higher-order, these problems can be avoided. Efficiency comparisons are made between a conventional higher-order method and the ETE

approach to higher-order solutions.

3.2 Background

3.2.1 Finite-Volume Discretization

Consider a general conservation law of the form

$$\frac{\partial \tilde{\mathbf{u}}}{\partial t} + \nabla \cdot \mathbb{F}(\tilde{\mathbf{u}}) = \mathbf{s}(\tilde{\mathbf{u}}) \tag{3.1}$$

defined over the domain $\Omega \subset \mathbb{R}^m \times [0, \infty), \mathbf{x} \in \mathbb{R}^m, m \in \{1, 2, 3\}$ where $\tilde{\mathbf{u}}$ are the conserved quantities, $\mathbb{F}(\cdot)$ is a flux tensor, and $\mathbf{s}(\cdot)$ is a source term. In general, the flux tensor and source term in Eq. 3.1 may also be functions of $\nabla\tilde{\mathbf{u}}$, but this dependency has been omitted here for the sake of brevity. The solution domain, Ω, is decomposed into a set of non-overlapping control volumes, $\mathcal{T}_h$, such that

$$\Omega = \bigcup_{i=1}^{N_v} \Omega_i, \quad \Omega_i \in \mathcal{T}_h \tag{3.2}$$

where N_v is the number of control volumes. The general conservation law may be cast into a weak form by integrating over each control volume, Ω_i, and using the Gauss divergence theorem to convert the flux divergence integral into a surface integral. For a fixed control volume, the weak form of Eq. 3.1 is given as

$$\mathbf{N}(\tilde{\mathbf{u}}) \coloneqq \frac{\partial}{\partial t} \int_{\Omega_i} \tilde{\mathbf{u}} \; d\Omega + \oint_{\partial\Omega_i} \mathbb{F}(\tilde{\mathbf{u}}) \cdot \hat{\mathbf{n}} \; dS - \int_{\Omega_i} \mathbf{s}(\tilde{\mathbf{u}}) \; d\Omega = \mathbf{0} \tag{3.3}$$

where $\hat{\mathbf{n}}$ is the outward unit normal vector on the surface of Ω_i. In the finite-volume method, a discrete solution, $\mathbf{u}_h$, is sought which approximates the control volume average of the exact continuous solution, $\tilde{\mathbf{u}}$. The discrete solution, $\mathbf{u}_h$, may be obtained in each control volume by solving the following semi-discrete form of Eq. 3.3

$$\mathbf{N}_h^p(\mathbf{u}_h) \coloneqq |\Omega_i| \frac{d\mathbf{u}_h}{dt} + \oint_{\partial\Omega_i} \mathbb{F}_h(I_h^p \mathbf{u}_h^+, I_h^p \mathbf{u}_h^-) \cdot \hat{\mathbf{n}} \; dS - \int_{\Omega_i} \mathbf{s}(I_h^p \mathbf{u}_h) \; d\Omega = \mathbf{0} \; , \tag{3.4}$$

where $|\Omega_i|$ is the volume of Ω_i. The analytic flux within the flux integral is discretized with a flux function, $\mathbb{F}_h(\cdot, \cdot)$, to account for the discontinuous solution at control volume boundaries. The operator I_a^b is a restriction or prolongation operator; the subscript, a, denotes the starting space and the superscript, b, denotes the resultant space. A subscript or superscript h denotes a discrete space on $\mathcal{T}_h$, and an empty subscript or superscript denotes a continuous space. The restriction operator, I^h, is used extensively throughout the remainder this work and is defined for finite-volume schemes as

$$I^h(\cdot) \coloneqq \frac{1}{|\Omega_i|} \int_{\Omega_i} (\cdot) \; d\Omega \; . \tag{3.5}$$

The notation $(\cdot)^{+/-}$ denotes the trace of the solution taken from the exterior and interior of Ω_i, respectively. Higher-order spatial accuracy is achieved by constructing a p^{th} order continuous function, $I_h^p \mathbf{u}_h$, in each control volume using information from a local stencil of cells. Eq. 3.4 may be written for each Ω_i and combined to form a system of ODEs

$$|\boldsymbol{\Omega}| \frac{d\mathbf{u}_h}{dt} + \mathbf{R}(\mathbf{u}_h) = \mathbf{0} \tag{3.6}$$

where $\mathbf{R}(\mathbf{u}_h)$ is the steady-state residual. Eq. 3.6 may be marched in time from some initial state using any ODE time integrator.

3.2.2 Error Transport Equations

The accuracy of the discrete solution, $\mathbf{u}_h$, may be assessed by quantifying the discretization error,

$$\varepsilon_h^{\hat{p}} = \mathbf{u}_h - I^h \tilde{\mathbf{u}} \ . \tag{3.7}$$

The discretization error converges with grid refinement at an order $\hat{p}$ which is the same as the formal order of the discretization, p, for smooth problems, but is often lower due to the presence of geometric and flow-field discontinuities and/or singularities [26]. Since the exact continuous solution, $\tilde{\mathbf{u}}$, is rarely known, discretization error must typically be estimated. In this work, discretization error is estimated by solving a separate transport equation for the local discretization error in each control volume. In order to derive such an equation, the continuous equations, $\mathbf{N}(\cdot)$, must first be related to the corresponding discrete equations, $\mathbf{N}_h^p(\cdot)$, through the Generalized Truncation Error Expression (GTEE) [27, 28] as follows

$$\mathbf{N}_h^p(I^h \mathbf{u}) = I^h \mathbf{N}(\mathbf{u}) + \tau_h^p(\mathbf{u}) \tag{3.8}$$

where $\mathbf{u}$ is an arbitrary continuous function and $\tau_h^p(\cdot)$ is the truncation error. Upon further examination of Eq. 3.8, truncation error can be viewed as higher-order terms which are neglected during the discretization of $\mathbf{N}(\cdot)$. Error transport equations (ETE) are now derived by inserting $\tilde{\mathbf{u}}$ into Eq. 3.8 and substituting the definition of the discretization error to form

$$\begin{aligned} \mathbf{N}_h^q(\mathbf{u}_h - \varepsilon_h^{\hat{p}}) &= I^h \cancelto{\mathbf{0}}{\mathbf{N}(\tilde{\mathbf{u}})} + \tau_h^q(\tilde{\mathbf{u}}) \\ &= \tau_h^q(\tilde{\mathbf{u}}) \ . \end{aligned} \tag{3.9}$$

For the sake of generality, it has been assumed that the ETE are discretized to an order q. If the discrete operator, $\mathbf{N}_h^q(\cdot)$, were instead linear, denoted by $\mathbf{L}_h^q(\cdot)$, and the ETE were discretized to an order p, the more common form of the ETE could be derived as follows

$$\mathbf{L}_h^p(\varepsilon_h^p) = -\tau_h^p(\tilde{\mathbf{u}}) \tag{3.10}$$

which states that the discretization error is governed by the same equations as the discrete solution with truncation error acting as the local source.

In this work, we solve a linearized form of the ETE rather than the full nonlinear ETE. The linearized form of the ETE may be obtained by performing a Newton linearization of the discrete operator as follows

$$\begin{aligned}\mathbf{N}_h^p(I^h\tilde{\mathbf{u}}) &= \mathbf{N}_h^p(\mathbf{u}_h) - \frac{\partial \mathbf{N}_h^p}{\partial \mathbf{u}_h}\varepsilon_h^{\hat{p}} + O\left(\left|\left|\varepsilon_h^{\hat{p}}\right|\right|^2\right) \\ &= -\frac{\partial \mathbf{N}_h^p}{\partial \mathbf{u}_h}\varepsilon_h^{\hat{p}} + O\left(\left|\left|\varepsilon_h^{\hat{p}}\right|\right|^2\right) .\end{aligned} \tag{3.11}$$

where it is assumed that $q = p$ and that the discrete solution is iteratively converged. Next, Eq. 3.3 and Eq. 3.8 are used to insert the truncation error as follows

$$\frac{\partial \mathbf{N}_h^p}{\partial \mathbf{u}_h}\varepsilon_h^{\hat{p}} = -\tau_h^p(\tilde{\mathbf{u}}) + O\left(\left|\left|\varepsilon_h^{\hat{p}}\right|\right|^2\right). \tag{3.12}$$

Finally, by neglecting higher-order terms, the linearized ETE may be written as

$$\frac{\partial \mathbf{N}_h^p}{\partial \mathbf{u}_h}\varepsilon_h^{\hat{p}} = -\tau_h^p(\tilde{\mathbf{u}}) . \tag{3.13}$$

The linearized ETE often produce very accurate discretization error estimates as long as the neglected $O(||\varepsilon_h^{\hat{p}}||^2)$ terms are small, which is typically the case when $\mathbf{u}_h$ is in the asymptotic range and in the absence of strong nonlinear effects such as shocks and other types of discontinuities. The Jacobian in Eq. 3.13 must be the full linearization of the discrete operator and can be determined via hand-differentiation, automatic differentiation [29], or complex-step methods [30]. The linearized ETE are preferred over the full nonlinear ETE because minimal code modifications are required for their implementation as long as an implicit solver with the full linearization is readily available. Also, the linearized ETE are computationally cheaper since a full nonlinear solve may be replaced by the solution of one linear system.

3.2.3 Discrete Adjoint Methods

Error estimates for output functionals, such as lift or drag, are commonly computed using adjoint methods. Furthermore, adjoint methods can provide adaptation indicators that target regions of the computational domain which contribute most to the error in the functional. First, consider an expansion of a discrete solution functional, $J_h(\cdot)$, about $\mathbf{u}_h$,

$$J_h(I^h\tilde{\mathbf{u}}) = J_h(\mathbf{u}_h) - \frac{\partial J_h}{\partial \mathbf{u}_h}\varepsilon_h^{\hat{p}} + O\left(\left|\left|\varepsilon_h^{\hat{p}}\right|\right|^2\right). \tag{3.14}$$

By rearranging, the error in the functional may be defined as

$$\varepsilon_{J_h} = J_h(\mathbf{u}_h) - J_h(I^h\tilde{\mathbf{u}}) = -\lambda_h^T\tau_h^p(\tilde{\mathbf{u}}) + O\left(\left|\left|\varepsilon_h^{\hat{p}}\right|\right|^2\right) \tag{3.15}$$

where λ_h are discrete adjoint variables which satisfy the following linear adjoint equation

$$\left[\frac{\partial \mathbf{N}_h^p}{\partial \mathbf{u}_h}\right]^T \lambda_h = \left[\frac{\partial J_h}{\partial \mathbf{u}_h}\right]^T. \tag{3.16}$$

Similar to the linearized ETE, the Jacobian on the left hand side of Eq. 3.16 must be the full linearization of the discrete operator. A functional error estimate can then be obtained by taking the inner product of the adjoint variables with the truncation error. As with any error estimation procedure, the error estimate may be applied as a correction to the functional. A benefit of adjoint methods is that upon application of the error estimate as a correction, the corrected functional, $\hat{J}_h(\mathbf{u}_h)$, may converge to the exact discrete functional at a higher rate, namely,

$$\begin{aligned} \hat{J}_h(\mathbf{u}_h) - J_h(I^h\tilde{\mathbf{u}}) &= \left[J_h(\mathbf{u}_h) + \lambda_h^T \tau_h^p(\tilde{\mathbf{u}})\right] - J_h(I^h\tilde{\mathbf{u}}) \\ &= O\left(\left|\left|\varepsilon_h^{\hat{p}}\right|\right|^2\right) \\ &= O(h^{2\hat{p}}). \end{aligned} \tag{3.17}$$

3.2.4 Truncation Error Estimation

Both ETE and adjoint methods require the quantification of truncation error, $\tau_h^p(\tilde{\mathbf{u}})$. Since the exact continuous solution, $\tilde{\mathbf{u}}$, is generally not known, a truncation error estimate must be obtained instead. Several common techniques for estimating truncation error are listed below.

- Embedded Grid Methods [31, 32]: Operate a coarse/fine grid discrete operator on a fine/coarse grid solution
- Higher-Order Residual Methods [20, 25]: Operate a higher-order discrete operator on a lower-order solution
- Traditional Truncation Error Analysis [14]: Expand the governing equations to examine the leading order truncation error terms
- Continuous Residual Methods [21]: Operate the continuous equations on a higher-order reconstruction of the discrete solution

Each of these truncation error estimation techniques has been utilized with some degree of success. When selecting a truncation error estimator, practitioners must weigh the advantages and disadvantages of each approach. Embedded grid estimators have largely been employed for adjoint-based error estimation [19, 33, 34] and are particularly useful when a

higher-order geometry representation is not available. But, embedded grid estimators require storage of the primal solution on another grid level which may not be feasible for large problems due to limited computational resources. Additionally, embedded grid estimators assume that the truncation error converges at the formal order of the primal discretization, which in general may not be the case [35]. Higher-order residual methods can provide reasonably accurate truncation error estimates and are commonly used when a higher-order reconstruction is readily available. However, for discretizations explicitly limited to a low order, the implementation of a higher-order discrete residual operator can be burdensome in terms of the required coding effort. If the leading order terms of the truncation error can be analytically derived via traditional truncation error analysis, tremendous insight can be gained regarding how discretization errors are generated and transported. Although, for finite-volume schemes, deriving the explicit form of the leading order truncation error terms is often a difficult, tedious, and error-prone process, especially on general unstructured grids. Due to their mathematical foundation, continuous residual (CR) methods often yield very accurate truncation error estimates. Yet, in the context finite-volume methods, the CR approach can be challenging to implement. In order to properly operate the continuous equations, the reconstructed solution must be continuous across control volume boundaries and satisfy the continuous boundary conditions.

3.2.5 Adjoint/ETE Equivalence for Functional Error Estimation

Once an accurate truncation error estimate has been obtained, error estimates may be computed via an adjoint method and/or the ETE. In a typical error estimation framework, the adjoint is used to compute functional error estimates while the ETE are used to compute local discretization error estimates. In this work, we utilize the equivalence between adjoint and ETE methods, which was stated in previous work [18], in order to provide a more unified error estimation framework. Using Eq. 3.15 and Eq. 3.16 and neglecting higher-order terms, the error estimate for an output functional of interest may be rewritten as

$$\varepsilon_{J_h} \approx -\overbrace{\frac{\partial J_h}{\partial \mathbf{u}_h}\underbrace{\left[\frac{\partial \mathbf{N}_h^p}{\partial \mathbf{u}_h}\right]^{-1}}_{\text{Adjoint Problem} \,=\, -\lambda_h^T}}^{\text{Error Transport Equations} \,=\, -\varepsilon_h^{\hat{p}}} \tau_h^p(\tilde{\mathbf{u}}) \quad . \tag{3.18}$$

The functional error estimate can either be viewed as the inner product of the adjoint solution with the truncation error or as the inner product of the functional linearization with the ETE solution. By taking the ETE approach, error estimates may be obtained for any functional of interest in addition to local discretization error estimates for all state variables. The ETE approach is also substantially cheaper than a conventional adjoint approach since an adjoint solution would be needed for each functional of interest.

3.3 Methodology

3.3.1 Pseudo-CR Truncation Error Estimation

We propose a truncation error estimation technique for finite-volume methods which combines aspects of both higher-order residual methods and continuous residual (CR) methods. The goal of this approach is to maintain the accuracy of CR methods while leveraging the implementation advantages of higher-order residual methods. With CR truncation error estimation, the discrete solution is first projected onto an r^{th} order polynomial space. It is assumed that the reconstructed solution, $I_h^r \mathbf{u}_h$, sufficiently approximates the exact continuous solution such that

$$\left|\left|\tau_h^p(\tilde{\mathbf{u}}) - \tau_h^p(I_h^r \mathbf{u}_h)\right|\right| = O(h^r) \ . \tag{3.19}$$

Using Eq. 3.8, the truncation error estimate may be written as

$$\begin{aligned} \tau_h^p(I_h^r \mathbf{u}_h) &= \mathbf{N}_h^p(I_r^h I_h^r \mathbf{u}_h) - \mathbf{N}(I_h^r \mathbf{u}_h) \\ &= \mathbf{N}_h^p(\cancelto{\mathbf{0}}{\mathbf{u}_h}) - \mathbf{N}(I_h^r \mathbf{u}_h) \\ &= -\mathbf{N}(I_h^r \mathbf{u}_h) \ . \end{aligned} \tag{3.20}$$

The truncation error estimate is simply the residual from operating the continuous equations onto the reconstruction. From Eq. 3.19 and Eq. 3.20, the following conditions may be inferred:

1. $I_h^r \mathbf{u}_h$ must satisfy conservation of the mean:

$$\mathbf{u}_h = I_r^h I_h^r \mathbf{u}_h \tag{3.21}$$

2. $\mathbf{u}_h$ must be iteratively converged to machine precision such that

$$\mathbf{N}_h^p(\mathbf{u}_h) = \mathbf{0} \tag{3.22}$$

3. If $\tilde{\mathbf{u}}$ is a smooth, continuous function, then $I_h^r \mathbf{u}_h$ should also be a smooth, continuous function, and vice-versa.

4. $I_h^r \mathbf{u}_h$ must satisfy the continuous boundary conditions.

Conservation of the mean is not a difficult condition to meet as it is commonly enforced during the reconstruction step in most finite-volume schemes. For reconstruction methods which do not satisfy the conservativeness property, the discrete residual contribution that was neglected in Eq. 3.20 must be explicitly taken into account since $\mathbf{N}_h^p\left(I_r^h I_h^r \mathbf{u}_h\right) \neq \mathbf{N}_h^p(\mathbf{u}_h)$. Furthermore, requiring an iteratively converged solution may be impractical for large problems. In the case of an unconverged solution, it is unclear whether or not accounting for

the discrete residual contribution would still yield an accurate truncation error estimate. We believe that iterative error in the solution will likely degrade the accuracy of the truncation error estimate. Since iterative error is not the focus of this paper, the study of its effects on truncation error estimates is left for future work. With regards to the continuity of the reconstruction, a global reconstruction would need to be formulated in order to enforce inter-element continuity, as in Ref. [36]. For finite-volume codes, which typically employ local reconstructions, the implementation of a global reconstruction would be a significant undertaking.

In this work, we propose to reconstruct the solution using local reconstructions instead, an approach which can be easily adopted by the broader CFD community since much of the machinery for local reconstructions is already in place in most CFD solvers. With relaxation of the inter-element continuity requirement, the continuous operator must now be approximated with a higher-order discrete residual operator

$$\tau_h^p(I_h^r \mathbf{u}_h) = -\mathbf{N}(I_h^r \mathbf{u}_h) \approx -\mathbf{N}_h^r(\mathbf{u}_h), \quad \forall\ r > p \tag{3.23}$$

as in Refs. [20, 25]. In order to maintain the accuracy of the truncation error estimate, it is assumed that the reconstruction is sufficiently smooth such that the inter-element jumps in the reconstruction are at most $O(h^r)$. Moreover, within the context of CR truncation error estimation, several authors have alluded to the importance of enforcing boundary conditions on the reconstruction [23, 37]. Therefore, in hopes of improving the accuracy of the truncation error estimate, we extend this approach to higher-order residual methods by directly enforcing the primal boundary conditions. The accuracy of the truncation error estimate is assessed using an effectivity index

$$\theta_2 = \frac{\left|\left|\tau_h(I_h^r \mathbf{u}_h)\right|\right|_2}{\left|\left|\tau_h(\tilde{\mathbf{u}})\right|\right|_2} , \tag{3.24}$$

which should converge to $\theta_2 = 1.0$ with grid refinement for an asymptotically correct truncation error estimate. The accuracy of the truncation error estimate is also assessed by monitoring the error in the truncation error estimate, i.e. $\left|\left|\tau_h^p(\tilde{\mathbf{u}}) - \tau_h^p(I_h^r \mathbf{u}_h)\right|\right|_2$. Exact truncation error is evaluated using the GTEE as follows

$$\tau_h^p(\tilde{\mathbf{u}}) = \mathbf{N}_h^p(I^h \tilde{\mathbf{u}}) . \tag{3.25}$$

3.3.1.1 Solution Reconstruction

Solution reconstruction is performed using k-exact least-squares reconstruction [38, 39]. A reconstruction is considered k-exact if it is able to reconstruct polynomials of degree k exactly. Therefore, for any smooth function, $\mathbf{u}(\mathbf{x})$, the error in the reconstructed function, $\mathbf{u}_i^R(\mathbf{x})$, converges as

$$\left|\left|\mathbf{u}_i^R(\mathbf{x}) - \mathbf{u}(\mathbf{x})\right|\right| = O(h^{k+1}) . \tag{3.26}$$

For simplicity, the k-exact reconstruction is presented here for 1D functions only. For details regarding the extension to higher dimensions, refer to Refs. [39, 40]. The reconstruction in Ω_i is first written as an expansion about the centroid of the control volume as follows

$$u_i^R(x - x_i) = u|_i + \left.\frac{\partial u}{\partial x}\right|_i (x - x_i) + \left.\frac{\partial^2 u}{\partial x^2}\right|_i \frac{(x - x_i)^2}{2} + \cdots \ . \tag{3.27}$$

The coefficients of the reconstructed polynomial are determined by requiring the reconstruction to satisfy conservation of the mean in Ω_i as well as in a stencil of neighboring control volumes, Ω_j,

$$\begin{aligned} \frac{1}{|\Omega_i|}\int_{\Omega_i} u_i^R(x - x_i)\ d\Omega &= u|_i + \left.\frac{\partial u}{\partial x}\right|_i \overline{x}_i + \left.\frac{\partial^2 u}{\partial x^2}\right|_i \frac{\overline{x^2}_i}{2} + \cdots = \bar{u}_i, \\ \frac{1}{|\Omega_j|}\int_{\Omega_j} u_i^R(x - x_i)\ d\Omega &= u|_i + \left.\frac{\partial u}{\partial x}\right|_i \widehat{x}_{ij} + \left.\frac{\partial^2 u}{\partial x^2}\right|_i \frac{\widehat{x^2}_{ij}}{2} + \cdots = \bar{u}_j \end{aligned} \tag{3.28}$$

where $\bar{u}_i$ is the control volume average of $u(x)$. The terms $\overline{x^n}_i$ and $\widehat{x^n}_{ij}$ are referred to as control volume moments and may be computed as follows

$$\begin{aligned} \overline{x^n}_i &= \frac{1}{|\Omega_i|}\int_{\Omega_i} (x - x_i)^n\ d\Omega \\ \widehat{x^n}_{ij} &= \frac{1}{|\Omega_j|}\int_{\Omega_j} ((x - x_j) + (x_j - x_i))^n\ d\Omega = \sum_{m=0}^{n} \frac{n!}{m!(n-m)!}(x_j - x_i)^m \overline{x^{n-m}}_j. \end{aligned} \tag{3.29}$$

The volume integrations in Eq. 3.29 are performed using Gaussian quadrature with a sufficient integration order to preserve the accuracy of the reconstruction. By writing $\widehat{x^n}_{ij}$ in the form given in Eq. 3.29, the memory requirements of the reconstruction may be reduced; only the control volume moments $\overline{x^n}_i$ need to be computed and stored while $\widehat{x^n}_{ij}$ may be computed from $\overline{x^n}_i$ as needed. Using Eq. 3.28 and Eq. 3.29, the linear system for the reconstruction coefficients may be written as

$$\left[\begin{array}{cccc} 1 & \overline{x}_i & \overline{x^2}_i & \cdots \\ \hline w_{i1} & w_{i1}\widehat{x}_{i1} & w_{i1}\widehat{x^2}_{i1} & \cdots \\ w_{i2} & w_{i2}\widehat{x}_{i2} & w_{i2}\widehat{x^2}_{i2} & \cdots \\ \vdots & \vdots & \vdots & \ddots \\ w_{iN} & w_{iN}\widehat{x}_{iN} & w_{iN}\widehat{x^2}_{iN} & \cdots \end{array}\right] \left(\begin{array}{c} u \\ \frac{\partial u}{\partial x} \\ \frac{1}{2}\frac{\partial^2 u}{\partial x^2} \\ \vdots \end{array}\right)_i = \left(\begin{array}{c} \overline{u}_i \\ \hline w_{i1}\overline{u}_1 \\ w_{i2}\overline{u}_2 \\ \vdots \\ w_{iN}\overline{u}_N \end{array}\right) \tag{3.30}$$

where N is the total number of cells in the stencil. The weights, w_{ij}, have been added to emphasize geometrically close data and are defined as

$$w_{ij} = \frac{1}{|x_j - x_i|^\alpha}\ . \tag{3.31}$$

Following Ref. [40], $\alpha = 1$ for all results presented in this work. All rows in the linear system above the horizontal line represent constraint rows which are enforced via Gaussian elimination. Therefore, the conservation of the mean requirement in Eq. 3.21 is exactly satisfied. The remaining unconstrained least-squares problem is solved using QR factorization [41]. For grids with control volumes fixed in space, the left-hand side matrix in Eq. 3.30 may be inverted as a pre-processing step to reduce computational costs. For more details regarding k-exact reconstructions, the reader is referred to Refs. [39, 40, 42].

3.3.1.2 Boundary Condition Enforcement

The primal boundary conditions are directly enforced on the reconstruction to improve the accuracy of truncation error estimates. Boundary condition enforcement for a k-exact reconstruction is relatively straightforward [39]. Consider a function, $f_1(x)$, that is to be enforced as a Dirichlet boundary condition at a quadrature point, x_g, on the boundary. It is required that the reconstruction evaluated at x_g satisfy the Dirichlet boundary condition, i.e.

$$\begin{aligned} f_1(x_g) &= u_i^R(x_g - x_i) \\ &= u|_i + \left.\frac{\partial u}{\partial x}\right|_i (x_g - x_i) + \left.\frac{\partial^2 u}{\partial x^2}\right|_i \frac{(x_g - x_i)^2}{2} + \cdots . \end{aligned} \tag{3.32}$$

Inserting Eq. 3.32 into Eq. 3.30, a new linear system for the reconstruction coefficients may be written as

$$\left[\begin{array}{cccc} 1 & \overline{x}_i & \overline{x^2}_i & \cdots \\ 1 & (x_g - x_i) & (x_g - x_i)^2 & \cdots \\ \hline w_{i1} & w_{i1}\widehat{x}_{i1} & w_{i1}\widehat{x^2}_{i1} & \cdots \\ w_{i2} & w_{i2}\widehat{x}_{i2} & w_{i2}\widehat{x^2}_{i2} & \cdots \\ \vdots & \vdots & \vdots & \ddots \\ w_{iN} & w_{iN}\widehat{x}_{iN} & w_{iN}\widehat{x^2}_{iN} & \cdots \end{array}\right] \left(\begin{array}{c} u \\ \frac{\partial u}{\partial x} \\ \frac{1}{2}\frac{\partial^2 u}{\partial x^2} \\ \vdots \end{array}\right)_i = \left(\begin{array}{c} \overline{u}_i \\ \hline f_1(x_g) \\ w_{i1}\overline{u}_1 \\ w_{i2}\overline{u}_2 \\ \vdots \\ w_{iN}\overline{u}_N \end{array}\right). \tag{3.33}$$

As before, all constraints are enforced exactly by performing Gaussian elimination on the constraint rows. Neumann boundary conditions can be enforced in a similar fashion [39].

3.3.2 ETE Approach to Higher-Order Primal Solutions

In addition to improved truncation error estimation techniques, we propose a new method for obtaining higher-order error estimates, or equivalently higher-order primal solutions [43]. This ETE approach utilizes adjoint/ETE equivalence in order to extend the higher-order properties of adjoint methods to ETE methods. First, recall that the error in any functional of interest may be written as

$$\varepsilon_{J_h} = J_h(\mathbf{u}_h) - J_h(I^h\tilde{\mathbf{u}}) = -\lambda_h^T \tau_h^p(\tilde{\mathbf{u}}) + O\left(\left|\left|\varepsilon_h^{\hat{p}}\right|\right|^2\right). \tag{3.34}$$

Typically, output functionals are integrated quantities of the solution, such as lift or drag for an aerodynamics application. However, consider the selection of the primal solution in Ω_i as the functional of interest. If the primal solution in each control volume were to be corrected, a standard adjoint approach would require an adjoint solution for each primitive variable in each control volume in $\mathcal{T}_h$. This would be prohibitively expensive and impractical. If adjoint/ETE equivalency is utilized instead, correction of the entire solution becomes feasible.

Using adjoint/ETE equivalency, the error in the functional may be expressed as

$$\varepsilon_{J_h} = J_h(\mathbf{u}_h) - J_h(I^h\tilde{\mathbf{u}}) = \frac{\partial J_h}{\partial \mathbf{u}_h}\varepsilon_h^{\hat{p}} + O\left(\left|\left|\varepsilon_h^{\hat{p}}\right|\right|^2\right). \tag{3.35}$$

By selecting the primal solution in Ω_i as the functional of interest, the functional error becomes

$$\varepsilon_{J_h} = \mathbf{u}_{h,i} - \left(I^h\tilde{\mathbf{u}}\right)_i = \delta_{ij}\varepsilon_{h,j}^{\hat{p}} + O\left(\left|\left|\varepsilon_h^{\hat{p}}\right|\right|^2\right) \tag{3.36}$$

where δ_{ij} is the Kronecker delta. In practice, the exact discretization error, $\varepsilon_h^{\hat{p}}$, is not known, and an ETE error estimate is used instead. At this point, it is easy to see that the ETE error estimate can be viewed as a higher-order discretization error estimate since it converges to the true discretization error at a rate of $O(h^{2\hat{p}})$. Furthermore, the ETE error estimate may be applied as a correction to the solution. By moving the discrete solution, $\mathbf{u}_{h,i}$, to the right-hand side, the corrected solution $\hat{\mathbf{u}}_{h,i}$, is written as

$$\begin{aligned}\left(I^h\tilde{\mathbf{u}}\right)_i &= \mathbf{u}_{h,i} - \delta_{ij}\varepsilon_{h,j}^{\hat{p}} + O\left(\left|\left|\varepsilon_h^{\hat{p}}\right|\right|^2\right) \\ &= \hat{\mathbf{u}}_{h,i} + O\left(\left|\left|\varepsilon_h^{\hat{p}}\right|\right|^2\right).\end{aligned} \tag{3.37}$$

Since there is no inter-element dependence in the functional linearization, the correction step can easily be carried out in each control volume across the domain. Similar to an adjoint method, application of the correction increases the order of accuracy to twice that of the original solution (i.e., $O(h^{2\hat{p}})$) assuming the truncation error estimate is sufficiently accurate. We expect ETE error estimates to be suitable for the correction step as long as the primal solution is in the asymptotic range and is reasonably smooth.

This ETE approach to higher-order solutions can be thought of as a type of defect correction in the literal sense in that a defect in the solution is being corrected away. In comparison to a standard adjoint method, which only increases one functional to higher-order, application of the ETE error estimate as a correction increases all solution functionals to higher-order simultaneously. This approach to higher-order solutions is shown to be more efficient than a conventional higher-order method since a full higher-order solve may be replaced with a lower-order solve plus one ETE solution.

3.4 Test Case

3.4.1 Quasi-1D Euler Equations

The quasi-1D Euler equations are a statement of conservation of mass, momentum, and energy for a fluid. The domain of interest is one dimensional with a given area distribution, $A(x)$. The flow is assumed to be uniform in the transverse direction at a each station along the domain. The quasi-1D Euler equations in strong, conservation form are

$$\frac{\partial\Big(A(x)\mathbf{u}\Big)}{\partial t} + \frac{\partial\Big[A(x)\mathbf{F}(\mathbf{u})\Big]}{\partial x} = \mathbf{s}(\mathbf{u}) \tag{3.38}$$

where $\mathbf{u}$ is the vector of conserved variables, $\mathbf{F}(\mathbf{u})$ is the vector of inviscid fluxes, and $\mathbf{s}(\mathbf{u})$ is a source term. The conserved variables, inviscid fluxes, and source term are given by

$$\mathbf{u} = \begin{bmatrix} \rho \\ \rho u \\ \rho e_t \end{bmatrix}, \quad \mathbf{F}(\mathbf{u}) = \begin{bmatrix} \rho u \\ \rho u^2 + p \\ \rho u h_t \end{bmatrix}, \quad \mathbf{s}(\mathbf{u}) = \begin{bmatrix} 0 \\ p\frac{dA}{dx} \\ 0 \end{bmatrix}, \tag{3.39}$$

where ρ is the fluid density, u is the fluid velocity, p is the static pressure, e_t is the total energy, h_t is the total enthalpy, and $\frac{dA}{dx}$ describes the variation of the cross-sectional area. The system of equations is closed using the equation of state for a perfect gas such that the total energy and total enthalpy are given by

$$e_t = \frac{p}{\rho(\gamma - 1)} + \frac{u^2}{2}, \quad h_t = \frac{\gamma p}{\rho(\gamma - 1)} + \frac{u^2}{2} \tag{3.40}$$

where $\gamma = 1.4$ is the ratio of specific heats for air. Since the quasi-1D Euler equations have an exact solution [44, 45], the effectiveness of the proposed methods may be directly evaluated.

3.4.1.1 Quasi-1D Nozzle

The quasi-1D Euler equations are used to model flow through a converging-diverging nozzle. The nozzle is characterized by a contraction through which the flow accelerates followed by a diverging section where the flow expands subsonically or supersonically depending upon the pressure at the nozzle exit. The nozzle has a Gaussian area distribution given by

$$A(x) = 1 - 0.8e^{\left(\frac{-x^2}{2\sigma^2}\right)}, \quad x \in [-1, 1] \tag{3.41}$$

where $\sigma = 0.2$ [46]. Two isentropic flow conditions are examined: (1) fully subsonic flow and (2) fully supersonic flow through the diverging section. The area and Mach number distributions for both the subsonic and supersonic cases can be seen in Figure 3.1.

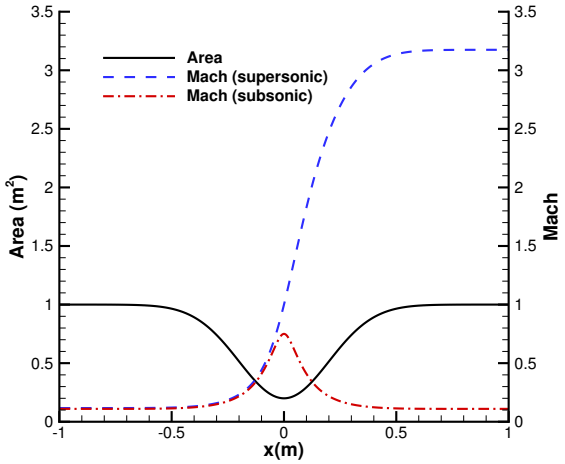


Figure 3.1: Mach number and area distributions along the nozzle.

At the nozzle inflow, stagnation pressure, p_0, and stagnation temperature, T_0, are fixed at 300 kPa and 600 K, respectively. The inflow Mach number is extrapolated from the interior to set the state at the inflow face. Outflow boundary conditions depend upon the local character of the flow in the diverging section. For supersonic flow, all variables are extrapolated from the interior of the domain to set the outflow flux. For subsonic flow, a back pressure, p_b, is specified and velocity and density are extrapolated from the interior. In this work, the back pressure is set to 297.485 kPa such that the Mach number at the throat is 0.75.

The quasi-1D Euler equations are discretized using the method of lines. The spatial discretization is a second order accurate, cell-centered, finite-volume discretization. The inviscid fluxes are computed using Roe's flux difference splitting scheme [47]. Second order spatial accuracy is achieved using a physical space MUSCL extrapolation [48] to reconstruct the primitive variables to the face. Boundary conditions are enforced weakly through the fluxes. Upon discretization of the spatial dimension, a system of ordinary differential equations is obtained which may be advanced in time. The temporal dimension is discretized using backward Euler time integration [49] to march the numerical solution to a steady-state. Since the primitive variables are vastly different orders of magnitude (e.g., density is $O(1)$ and pressure is $O(10^5)$), the quasi-1D Euler equations are coded in a nondimensional form to improve the numerical scaling of the solution procedure. For more details regarding the primal solver implementation, the reader is referred to Ref. [18].

3.5 Results

3.5.1 Pseudo-CR Truncation Error Estimation

Truncation error estimation is performed on a series of "structured" and "unstructured" grids ranging in size from 32 cells to 8192 cells. The reader should note that in 1D there is no difference in the actual data structure used to store the grids or the solution algorithm. The terminology "structured" and "unstructured" is used in a loose sense to refer to the typical geometric characteristics of higher-dimensional grids. For example, a "structured" grid refers to a grid in which the grid metrics exhibit a smooth variation across the domain. On the other hand, an "unstructured" grid refers to a grid in which the grid metrics exhibit a non-smooth variation across the domain. In this work, "structured" grids are modeled using uniformly-spaced grid nodes. "Unstructured" grids are modeled by applying random perturbations to the nodes of a "structured" grid to mimic the geometric non-uniformity typically found in a higher-dimensional unstructured grid. The nodal coordinates of a given "unstructured" grid are generated using the following relation

$$x_p = x_u + \Big[\Delta \times dx_u\Big] \times \text{rand}(-1, +1), \quad \Delta \in [0.0, 0.5) \tag{3.42}$$

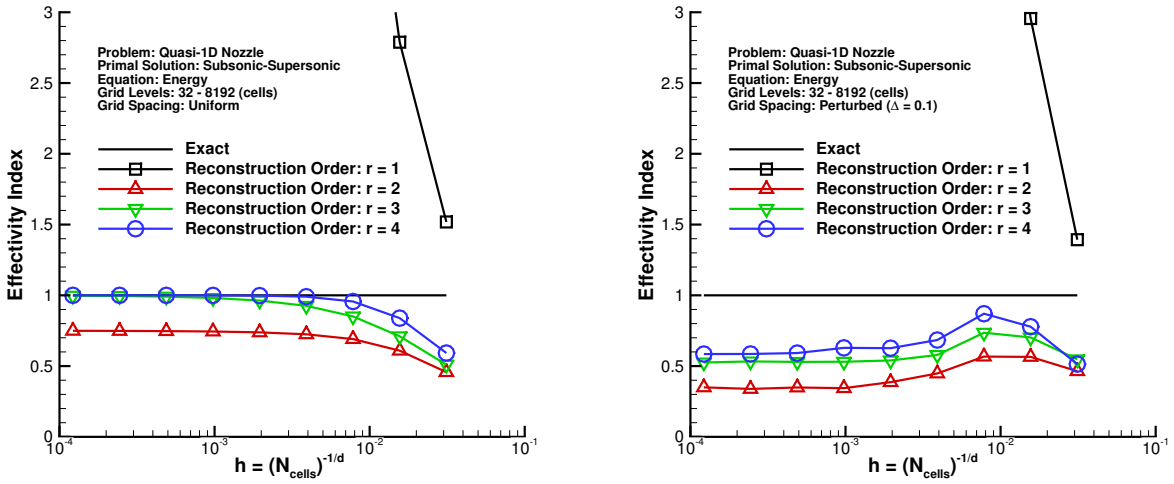
where x_p and x_u are the nodes of the perturbed grid and uniform grid, respectively, Δ is the magnitude of the perturbation, dx_u is the uniform grid spacing, and $\text{rand}(a, b)$ is a function which generates a uniformly-distributed random number between a and b. The magnitude of the perturbation is limited to $\Delta < 0.5$ to ensure positive cell volumes.

3.5.1.1 Accuracy of Truncation Error Estimate

An r^{th} order polynomial is reconstructed in each control volume, Ω_i, using the discrete solution, $\mathbf{u}_h$, from Ω_i and a patch of neighboring control volumes, Ω_j. Reconstruction orders of $r \in \{1, 2, 3, 4\}$ are examined. Truncation error estimates are computed using the pseudo-CR method outlined in Section 3.3.1. The accuracy of the truncation error estimate is assessed using an effectivity index and by directly monitoring the error in the truncation error estimate. In this section, results are only presented for truncation error in the energy equation. Similar results are observed for the continuity and momentum equations.

The convergence of the effectivity index for "structured" and "unstructured" grids may be found in Figure 3.2a and Figure 3.2b, respectively. For "structured" grids, the effectivity index asymptotically approaches 1.0 for reconstruction orders of $r \geq 3$. Since $p = 2$ for the primal problem, the reconstruction order must be greater than p in order for the higher-order residual calculation to capture any of the terms in the truncation error. It should be noted that the effectivity index for $r \leq 2$ is non-zero because different reconstructions are used for the primal problem and the truncation error estimate. For "unstructured grids", the effectivity index does not approach 1.0 for any reconstruction order, albeit increasing the

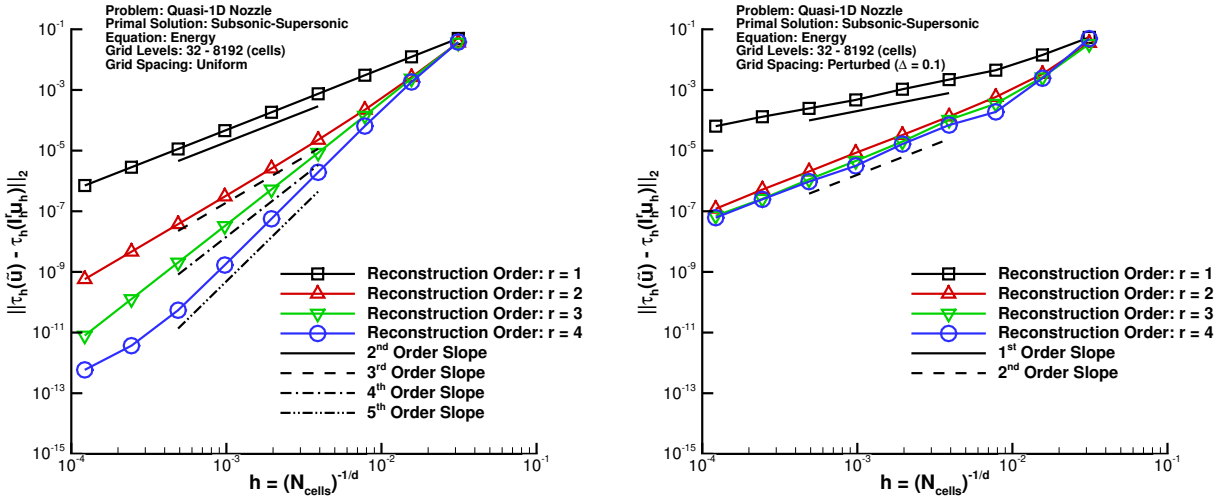
reconstruction order does improve the value of the effectivity index. This behavior indicates that the truncation error estimate is asymptotically different than the exact truncation error. This phenomenon will be explained later in this section.



(a) "Structured" grids (b) "Unstructured" grids

Figure 3.2: Convergence of truncation error effectivity indices for "structured" and "unstructured" grids.

The convergence of the error in the truncation error estimate for "structured" and "unstructured" grids may be found in Figure 3.3a and Figure 3.3b, respectively. For "structured" grids, the truncation error estimate converges to the exact truncation error at a rate of $r + 1$, i.e. faster than the expected rate of r. This superconvergent behavior can likely be attributed to the uniformity of the grid and cancellation of truncation errors. The reader should note that during testing non-uniform but smoothly spaced (e.g. hyperbolic tangent spaced) "structured" grids were also examined. The results for these grids were qualitatively similar to those of the uniformly spaced "structured" grids and were excluded here for brevity. For "unstructured" grids, the truncation error estimate converges at the expected rate for $r = 1$. However for $r \geq 2$, all truncation error estimates converge at a 2^{nd} order rate.



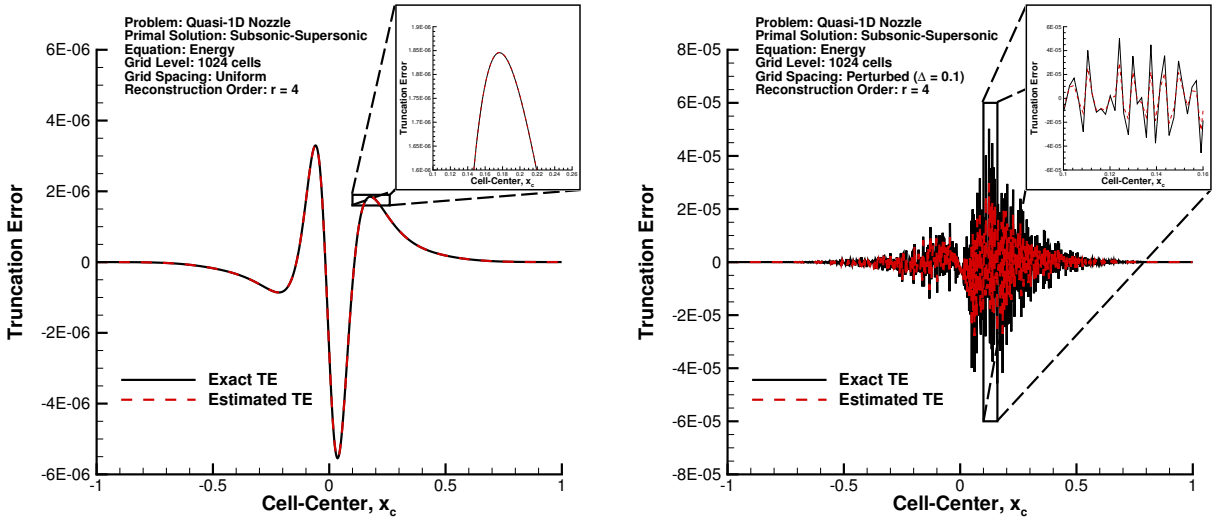
(a) "Structured" grids (b) "Unstructured" grids

Figure 3.3: Convergence of error in truncation error estimates for "structured" and "unstructured" grids.

In order to better understand the behavior of the truncation error estimate for an "unstructured" grid, it is helpful to examine the truncation error estimate in relation to the exact truncation error. Energy truncation error is plotted for a 1024 cell grid in Figure 3.4. Truncation error on the "structured" grid smoothly varies across the entire computational domain. The sign change in the slope of the truncation error at $x = \pm 0.2\ m$ is due to a sign change in the derivative of the area distribution at that location. The truncation error varies significantly near $x = 0.0\ m$ due to the singularity which occurs at the nozzle throat. At this grid level, the truncation error estimate is indistinguishable from the exact truncation error. On the contrary, the truncation error on the "unstructured" grid is noisy across much of the domain. The truncation error inherits the non-smoothness present in the grid. In comparison to the "structured" grid, the magnitude of the truncation error on the "unstructured" grid is asymptotically larger. While the truncation error estimate is able qualitatively capture much content in the exact truncation error, the truncation error estimate consistently underpredicts the true magnitude of the truncation error. This behavior is corroborated by an effectivity index less than 1.0.

We postulate that the accuracy loss of the truncation error estimate on "unstructured" grids is a direct result of the non-smoothness of the grid. For higher-dimensional unstructured grids, noise on the order of discretization error is often present in the discrete solution [20]. If this noise is present, we suspect that the reconstruction captures the noise in a way that violates the assumption on inter-element jumps, namely that the inter-element jumps are of $O(h^r)$ or better. For this test case, the primal solution is solved to $O(h^p)$ where $p = 2$. Therefore, it is expected that the noise in the discrete solution, and consequently the inter-element jumps in the reconstruction, will also be $O(h^p)$. This hypothesis could explain why

the error in the truncation error estimate for "unstructured" grids is at best $O(h^2)$. In order to improve the accuracy of the truncation error estimate, the reconstruction step would likely need to account for the noise in the discrete solution.



(a) "Structured" grid

(b) "Unstructured" grid

Figure 3.4: Energy truncation error on a "structured" and "unstructured" grid.

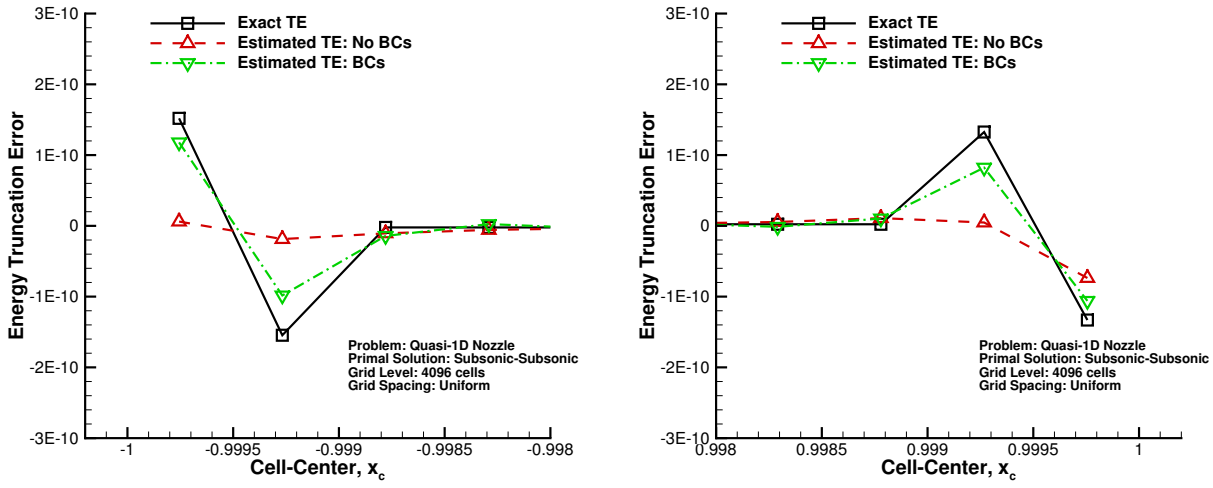
3.5.1.2 Boundary Condition Truncation Error Estimation

Since most engineering outputs are computed at domain boundaries, it is essential to accurately estimate the truncation error in these regions. In an attempt to improve the accuracy of truncation error estimates, we now combine aspects of CR truncation error estimation with higher-order residual methods. In addition to enforcing conservation of the mean, the continuous boundary conditions are enforced on the reconstruction. Boundary conditions for the quasi-1D nozzle problem are imposed on the reconstruction as follows:

- Subsonic Inflow Boundary:
  - Reconstruct Mach number to the face only enforcing conservation of the mean.
  - Use p_0 and T_0 to enforce Dirichlet constraints on static pressure and static temperature.
- Subsonic Outflow Boundary:
  - Use the back pressure, p_b, to enforce a Dirichlet constraint on static pressure.
  - Reconstruct density and velocity only enforcing conservation of the mean.
- Supersonic Outflow Boundary:

– Reconstruct density, velocity, and pressure only enforcing conservation of the mean.

Energy truncation error is plotted for a 4,096 cell "structured" grid in Figure 3.5 to illustrate how boundary condition enforcement impacts the accuracy of the truncation error estimate. Truncation error estimates are shown at the inflow and outflow boundaries for the subsonic-subsonic case. Similar results are observed for the subsonic-supersonic case and do not change the conclusions drawn here. From Figure 3.5, it can be observed that the boundary truncation error is fundamentally different than the interior truncation error. While the interior truncation error is relatively smooth near the inflow and outflow regions, the boundary truncation error is non-smooth in the first two cells adjacent to the boundaries. This difference arises from the change in discretization associated with the enforcement of the boundary conditions. When only enforcing conservation of the mean, the truncation error estimate is unable to capture the behavior of the exact truncation error at the inflow and outflow boundaries. By enforcing the continuous boundary conditions, the accuracy of the truncation error estimate is greatly improved. Despite a slight underprediction in magnitude, the truncation error estimate is able to capture the general behavior of the exact truncation error.



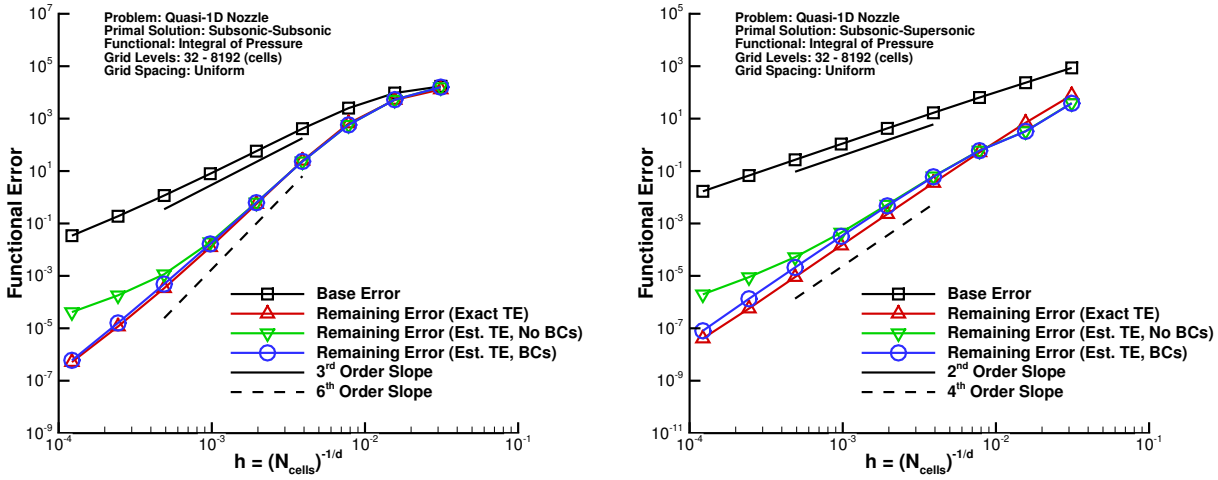
(a) Inflow boundary

(b) Outflow boundary

Figure 3.5: Boundary truncation error for a "structured" grid: subsonic-subsonic flow.

While improvements in the truncation error estimate may seem relatively minor, enforcing the boundary conditions on the reconstruction can have a significant impact on the accuracy of error estimates. To illustrate this effect, functional error estimates are obtained on 9 systematically refined "structured" grids ranging in size from 32 cells to 8,192 cells. The functional of interest is the integral of pressure along the nozzle. Functional error estimates are plotted for the subsonic-subsonic and subsonic-supersonic cases in Figure 3.6a and Figure 3.6b, respectively. The remaining error in the functional after correction by the error

estimate is used to assess the accuracy of the functional error estimate. It should be noted that all functional error estimates are computed with the ETE approach to functional error estimation [18].



(a) Subsonic-subsonic flow

(b) Subsonic-supersonic flow

Figure 3.6: Effect of boundary condition truncation error estimation on functional error convergence for "structured" grids.

Upon application of the error estimate as a correction, the remaining error converges at a higher-order rate for grids ranging in size from 32 cells to 1,024 cells. For the subsonic-supersonic case, the remaining error converges at the expected 4^{th} order rate. For the subsonic-subsonic case, the base error and the remaining error converge at approximately a 3^{rd} order and 6^{th} order rate, respectively. This superconvergent behavior was also observed for a similar subsonic test case in Ref. [11]. Beyond the 1,024 cell grid level, an $O(h^2)$ error is introduced when only enforcing conservation of the mean. This error only appears on finer grid levels once the incorrect contribution of the boundary cells to the functional error estimate is no longer small relative to the contributions from the rest of the domain. This $O(h^2)$ error can also be seen for $r = 4$ on the finest grid levels in Figure 3.3a. However, when boundary conditions are enforced on the reconstruction, the truncation error estimate is accurate enough to maintain higher-order accuracy across the entire range of grids. These results illustrate the importance of boundary condition enforcement on the reconstruction for obtaining higher-order error estimates. The reader should note that enforcing boundary conditions on the reconstruction does not improve the convergence rate of the truncation error estimate for "unstructured" grids. The effect of non-smooth grid metrics is still the dominant effect on these grids.

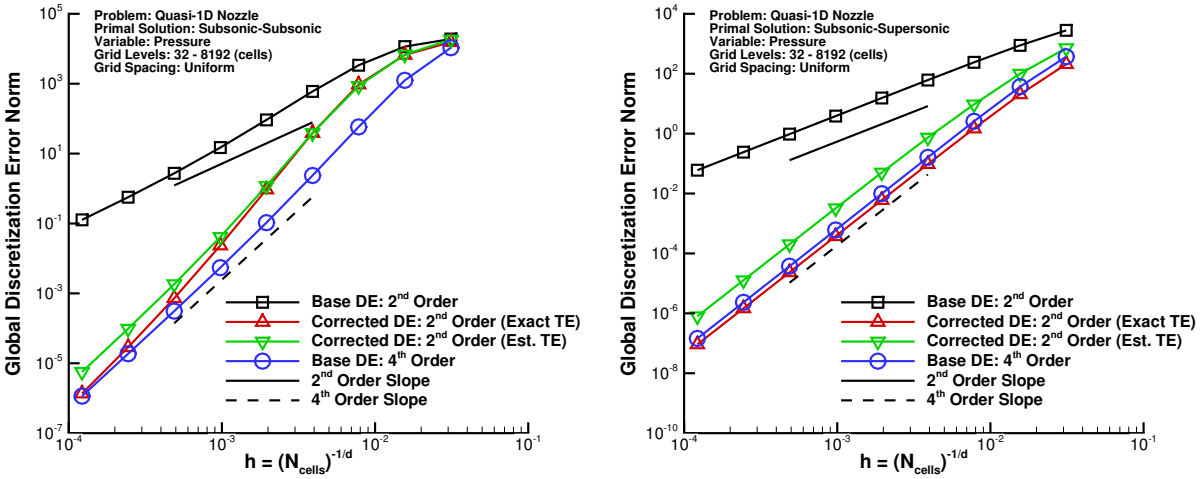
3.5.2 ETE Approach to Higher-Order Primal Solutions

The ETE approach to higher-order primal solutions outlined in Section 3.3.2 is applied to a series of systematically refined "structured" and "unstructured" grids. The ETE error estimate is applied as a correction to the primal solution according to Eq. 3.37. Results are presented for the subsonic-subsonic case and the subsonic-supersonic case. Corrections are applied to the $p = 2$ primal solution; for the $p = 1$ primal solution, the reader is referred to Ref. [43] for results. Discretization error estimates are obtained using exact truncation error and estimated truncation error. To evaluate this approach, corrected primal solutions are compared to the exact discretization error from a higher-order solver. The higher-order solver employed in this work uses a k-exact reconstruction to determine the left and right primitive states at each face. Boundary conditions are enforced in a similar manner to the lower-order solver. For all error estimates presented in this section, boundary conditions are imposed on the reconstruction in order to eliminate errors in the estimation of boundary truncation error.

3.5.2.1 "Structured" Grids

The convergence of pressure discretization error is plotted for "structured" grids in Figure 3.7. Similar results are also found for the remaining primitive variables. For the subsonic-subsonic case, the corrected 2^{nd} order solution is slightly higher than 4^{th} order on coarse grids but eventually asymptotes to 4^{th} order on finer grids. The error level for the corrected solution is the same as the higher-order solution on the finest grid level when exact truncation error is used for the ETE error estimate. When using estimated truncation error, the error level of the corrected solution is higher than that of the higher-order solution but is still comparable. For the subsonic-supersonic case, the corrected 2^{nd} order solution converges at a 4^{th} order rate for all grid levels. When using exact truncation error, the corrected solution is actually more accurate than the higher-order solution. Similar to the subsonic-subsonic case, the error level of the corrected solution when using estimated truncation error is slightly higher than the higher-order solution but is still comparable.

These results indicate that the ETE error estimate can indeed correct the entire primal solution, and by extension all solutions functionals, to higher-order. For this problem, corrected solutions are of similar accuracy to those obtained via a higher-order discretization. As a word of caution, practitioners must ensure that the solution is asymptotic, the truncation error estimate is sufficiently accurate, and strong nonlinearities are not present. If these conditions are not satisfied, applying the error estimate as a correction could locally decrease solution accuracy. Furthermore, if the full nonlinear ETE are used instead of the linearized ETE, we expect the correction step to perform well as long as an accurate truncation error estimate may be obtained. Additional studies are needed to verify this hypothesis.



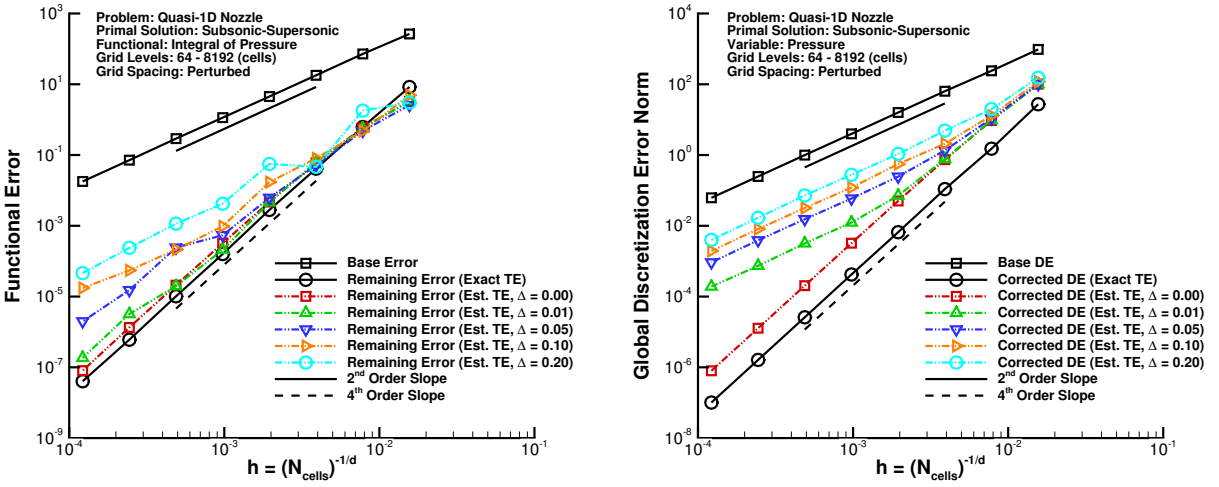
(a) Subsonic-subsonic flow

(b) Subsonic-supersonic flow

Figure 3.7: Pressure discretization error for uncorrected and corrected primal solutions on "structured" grids.

3.5.2.2 "Unstructured" Grids

In this section, the performance of the ETE approach to higher-order solutions is examined in the case when the truncation error estimate is not sufficiently accurate. In particular, the effect of non-smooth grid metrics on the accuracy of error estimates is studied in more detail. To aid in this investigation, functional error estimates are computed in addition to ETE error estimates. Functional error estimates and ETE error estimates are plotted for a series of "unstructured" grids in Figure 3.8. The functional of interest is the integral of pressure along the nozzle. Norms of the ETE error estimates are shown for static pressure. Error estimates are computed for grid perturbations of $\Delta \in \{0.0, 0.01, 0.05, 0.10, 0.20\}$. Base error and error estimates obtained with exact truncation error do not vary significantly with grid perturbation. Therefore, for clarity, base error and error estimates obtained with exact truncation error are only plotted for grids with $\Delta = 0.20$. Results are only presented here for the subsonic-supersonic case. Similar results are found for the subsonic-subsonic case.

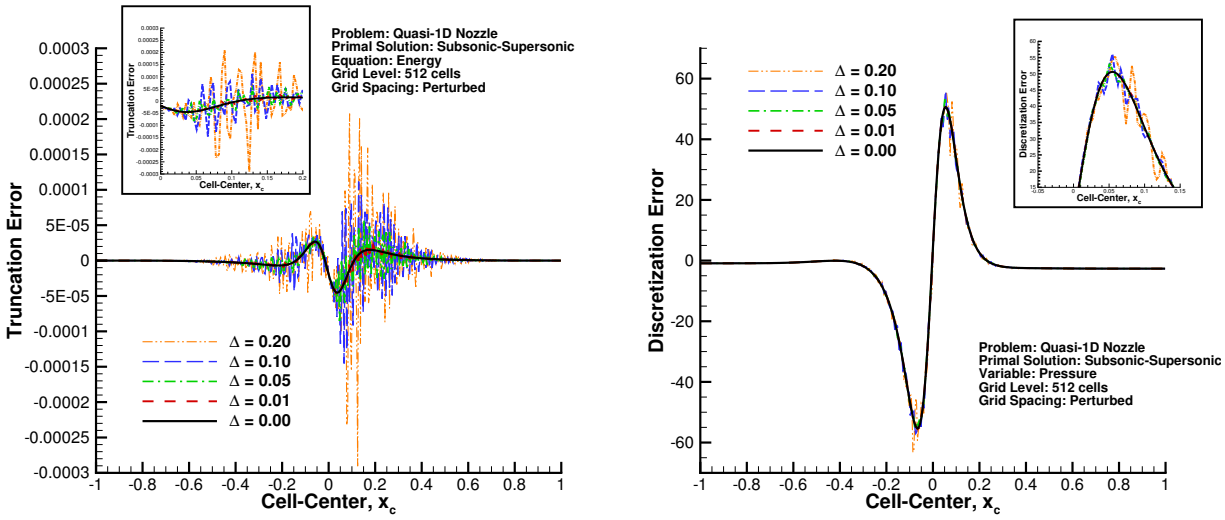


(a) Functional error (b) Discretization error

Figure 3.8: Effect of non-smooth grid metrics on the convergence of error estimates on "unstructured" grids.

For both functional and discretization error estimates, the accuracy of the error estimate degrades as the magnitude of the grid perturbation is increased. The higher-order convergence of the error estimates deteriorates to 2^{nd} order even for grid perturbations as small as $\Delta = 0.01$. Due to the cancellation of errors, degradation to 2^{nd} order occurs much more slowly for functional error estimates than for discretization error estimates. While the error estimates are no longer higher-order, their application as a correction still yields a constant factor improvement in the solution. It should be noted that order properties are only lost when using estimated truncation error to compute functional and discretization error estimates. Higher-order properties are maintained for all grid levels when using exact truncation error. Therefore, if the truncation error estimate could be improved, higher-order error estimates could potentially be ensured for "unstructured" grids, even for cases where $q = p$.

Local truncation error and local discretization error are plotted in Figure 3.9 for a 512 cell grid to further examine the impact of the grid perturbation. As the grid perturbation is increased, the magnitude of the noise in the truncation error increases significantly. Similar behavior is observed in the discretization error, but to a much lesser extent. As mentioned in Section 3.5.1, Figure 3.9b does illustrate that some amount of noise is in fact present in the discrete solution. Since the k-exact reconstruction attempts to satisfy conservation of the mean in each cell in the stencil, the reconstruction is likely capturing the non-physical noise in the discrete solution. By reconstructing this noise, the resulting truncation error estimate is not sufficiently accurate for higher-order discretization error estimates when $q = p$.



(a) Truncation error

(b) Discretization error

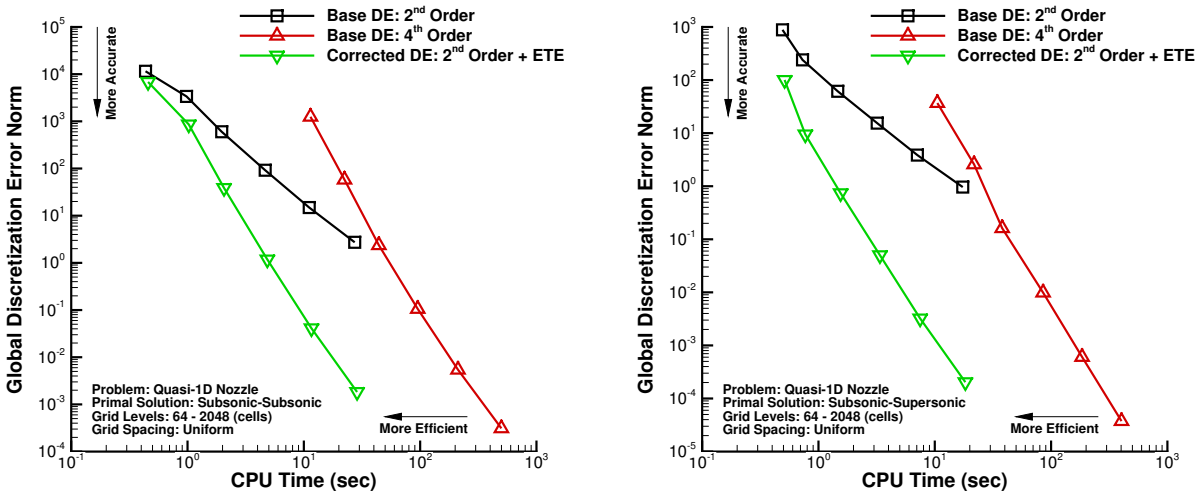
Figure 3.9: Effect of non-smooth grid metrics on truncation error and discretization error on "unstructured" grids.

3.5.2.3 Runtime Comparison: Lower-Order Discretization + ETE vs. Higher-Order Discretization

In order be competitive with conventional higher-order discretizations, the ETE approach to higher-order solutions must achieve significant computational savings. To assess the computational savings of the ETE approach, the total runtime of a lower-order primal solve plus an ETE correction is monitored and compared to the runtime of a higher-order primal solver. Higher-order primal solutions are obtained using the k-exact reconstruction outlined in Section 3.3.1.1. Time integration is performed in the same manner as the lower-order solver. All runs of the lower-order and higher-order primal solvers are initialized with the same initial state. Runtimes are monitored on a series of "structured" grids ranging in size from 64 cells to 2048 cells. Results are presented for both the subsonic-subsonic and subsonic-supersonic cases of the quasi-1D nozzle problem. All computations are performed in serial on a Dell Precision 5810 workstation with an Intel Xeon E5-1650 3.60 GHz processor. Results are presented in Figure 3.10. All reported runtimes are determined by averaging 5 separate runs.

Significant computational savings can be achieved with the ETE approach. For the cases examined, the ETE approach is approximately an order of magnitude faster than the traditional higher-order discretization. These speed-ups are due to the fact that the linearized ETE only adds the cost of one extra linear solve to the cost of the lower-order primal solve. By only solving a lower-order problem, the ETE approach also avoids some of the stability issues commonly associated with higher-order solvers. Most importantly, as illustrated in

the previous section, the ETE approach attains comparable error levels to that of a higher-order discretization, but does so much more efficiently. As long as higher-order accurate error estimates can be obtained, this technique could be a viable alternative to traditional higher-order discretizations for certain classes of problems.



(a) Subsonic-subsonic flow

(b) Subsonic-supersonic flow

Figure 3.10: Discretization error versus runtime for lower-order discretization + ETE and higher-order discretization.

3.6 Conclusions

In this work, a framework for estimating discretization error and improving solution accuracy was presented. A method for truncation error estimation was proposed which combines aspects of higher-order residual methods and continuous residual methods. In particular, the continuous boundary conditions for the primal problem were enforced on the reconstruction of the discrete solution to improve the accuracy of truncation error estimates. Additionally, the equivalence between adjoint methods and ETE methods for functional error estimation was utilized to extend the higher-order properties of adjoint methods to ETE methods. Similar to an adjoint method, ETE error estimates were applied as a correction to the primal solution. The resulting error levels as well as runtimes were compared to a conventional higher-order discretization. Results were presented for "structured" and "unstructured" grids.

For "structured" grids, truncation error varied smoothly across the domain. Truncation error estimates were asymptotically correct in the effectivity index and converged to the exact truncation error at approximately the expected rate. Enforcing boundary conditions on the reconstruction significantly improved the accuracy of truncation error estimates near

boundaries. In fact, boundary condition enforcement was necessary to maintain higher-order functional error estimates across all grid levels. Upon application of ETE error estimates as a correction, the entire primal solution and all solution functionals were improved to higher-order. The error levels of corrected solutions were found to be comparable to those obtained via a higher-order solver. In a runtime comparison, the ETE approach to higher-order solutions was approximately an order of magnitude faster than a higher-order discretization for the test problem examined.

For "unstructured" grids (i.e., those with random perturbations of the grid nodes), the truncation error exhibited a non-smooth behavior across the domain and was asymptotically larger in magnitude than the truncation error on a "structured" grid. The noise in the truncation error, and likewise the noise in the discrete solution, increased with the degree of non-smoothness in the grid. Truncation error estimates did not converge in the effectivity index and consistently underpredicted the magnitude of the exact truncation error. Consequently, the order of accuracy of ETE error estimates degraded to 2^{nd} order accurate. Despite the loss of higher-order accuracy, application of ETE error estimates as a correction still provided a constant factor improvement in solution accuracy. We suspect that the reconstruction captured the non-physical noise that was present in the discrete solution which led to a truncation error estimate that was not sufficiently accurate for higher-order error estimation. If more accurate reconstruction methods could be developed for finite-volume truncation error estimation, the accuracy of error estimates for "unstructured" grids could potentially be improved.

Acknowledgments

This work was supported by the Collaborative Center for Aeronautical Sciences with Dr. John Benek of the Air Force Research Laboratory in Dayton, Ohio serving as the program manager. Additional support was provided by the NASA Graduate Aeronautics Scholarship as part of the Aeronautics Scholarship and Advanced STEM Training and Research Fellowship (AS&ASTAR) Program. We wish to thank Dr. Michael Park and Dr. Joseph Derlaga of NASA Langley Research Center for serving as technical advisors and providing comments on this manuscript.

Bibliography

[1] L. F. Richardson, The approximate arithmetical solution by finite differences of physical problems involving differential equations, with an application to the stresses in a masonry dam, Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences 210 (459-470) (1911) 307 – 357. doi:10.1098/rsta.1911.

0009.
URL http://dx.doi.org/10.1098/rsta.1911.0009

[2] K. Laflin, O. Brodersen, M. Rakowitz, J. Vassberg, R. Wahls, J. Morrison, E. Tinoco, J.-L. Godard, Summary of data from the second AIAA CFD drag prediction workshop (invited), in: 42nd AIAA Aerospace Sciences Meeting and Exhibit, American Institute of Aeronautics and Astronautics (AIAA), 2004. doi:10.2514/6.2004-555.
URL http://dx.doi.org/10.2514/6.2004-555

[3] J. Vassberg, E. Tinoco, M. Mani, O. Brodersen, B. Eisfeld, R. Wahls, J. Morrison, T. Zickuhr, K. Laflin, D. Mavriplis, Summary of the third AIAA CFD drag prediction workshop, in: 45th AIAA Aerospace Sciences Meeting and Exhibit, American Institute of Aeronautics and Astronautics (AIAA), 2007. doi:10.2514/6.2007-260.
URL http://dx.doi.org/10.2514/6.2007-260

[4] L. Fox, Some improvements in the use of relaxation methods for the solution of ordinary and partial differential equations, Proceedings of the Royal Society of London. Series A, Mathematical and Physical Sciences 190 (1020) (1947) 31 – 59.
URL http://www.jstor.org/stable/98009

[5] L. Fox, The numerical solution of two-point boundary value problems in ordinary differential equations, Oxford University Press, 1957.

[6] V. Pereyra, Iterated deferred corrections for nonlinear operator equations, Numerische Mathematik 10 (4) (1967) 316 – 323. doi:10.1007/bf02162030.
URL http://dx.doi.org/10.1007/BF02162030

[7] V. Pereyra, Iterated deferred corrections for nonlinear boundary value problems, Numerische Mathematik 11 (2) (1968) 111 – 125. doi:10.1007/bf02165307.
URL http://dx.doi.org/10.1007/BF02165307

[8] H. J. Stetter, The defect correction principle and discretization methods, Numerische Mathematik 29 (4) (1978) 425 – 443. doi:10.1007/bf01432879.
URL http://dx.doi.org/10.1007/BF01432879

[9] M. B. Giles, N. Pierce, E. Süli, Progress in adjoint error correction for integral functionals, Computing and Visualization in Science 6 (2-3) (2004) 113 – 121. doi:10.1007/s00791-003-0115-y.
URL http://dx.doi.org/10.1007/s00791-003-0115-y

[10] M. B. Giles, N. A. Pierce, Analytic adjoint solutions for the quasi-one-dimensional Euler equations, Journal of Fluid Mechanics 426 (2001) 327 – 345. doi:10.1017/s0022112000002366.
URL http://dx.doi.org/10.1017/S0022112000002366

[11] D. A. Venditti, D. L. Darmofal, Adjoint error estimation and grid adaptation for functional outputs: Application to quasi-one-dimensional flow, Journal of Computational Physics 164 (1) (2000) 204 – 227. doi:10.1006/jcph.2000.6600.
URL http://dx.doi.org/10.1006/jcph.2000.6600

[12] D. A. Venditti, D. L. Darmofal, Grid adaptation for functional outputs: Application to two-dimensional inviscid flows, Journal of Computational Physics 176 (1) (2002) 40 – 69. doi:10.1006/jcph.2001.6967.
URL http://dx.doi.org/10.1006/jcph.2001.6967

[13] D. A. Venditti, D. L. Darmofal, Anisotropic grid adaptation for functional outputs: Application to two-dimensional viscous flows, Journal of Computational Physics 187 (1) (2003) 22 – 46. doi:10.1016/s0021-9991(03)00074-3.
URL http://dx.doi.org/10.1016/S0021-9991(03)00074-3

[14] X. Zhang, J.-Y. Trépanier, R. Camarero, A posteriori error estimation for finite-volume solutions of hyperbolic conservation laws, Computer Methods in Applied Mechanics and Engineering 185 (1) (2000) 1 – 19. doi:10.1016/s0045-7825(99)00099-7.
URL http://dx.doi.org/10.1016/S0045-7825(99)00099-7

[15] Y. Qin, T. Shih, A discrete transport equation for error estimation in CFD, in: 40th AIAA Aerospace Sciences Meeting and Exhibit, American Institute of Aeronautics and Astronautics (AIAA), 2002. doi:10.2514/6.2002-906.
URL http://dx.doi.org/10.2514/6.2002-906

[16] I. Celik, G. Hu, Single grid error estimation using error transport equation, Journal of Fluids Engineering 126 (5) (2004) 778. doi:10.1115/1.1792254.
URL http://dx.doi.org/10.1115/1.1792254

[17] J. M. Derlaga, Application of improved truncation error estimation techniques to adjoint based error estimation and grid adaptation, Ph.D. thesis, Virginia Polytechnic Institute and State University, Blacksburg, VA (Jul. 2015).
URL http://hdl.handle.net/10919/54592

[18] W. C. Tyson, K. Swirydowicz, J. M. Derlaga, C. J. Roy, E. de Sturler, Improved functional-based error estimation and adaptation without adjoints, in: 46th AIAA Fluid Dynamics Conference, American Institute of Aeronautics and Astronautics (AIAA), 2016. doi:10.2514/6.2016-3809.
URL http://dx.doi.org/10.2514/6.2016-3809

[19] D. A. Venditti, Grid adaptation for functional outputs of compressible flow simulations, Ph.D. thesis, Massachusetts Institute of Technology (Jun. 2002).
URL http://hdl.handle.net/1721.1/29246

[20] M. Sharbatdar, Error estimation and mesh adaptation paradigm for unstructured mesh finite volume methods, Ph.D. thesis, University of British Columbia (Jan. 2017).
URL http://hdl.handle.net/2429/60359

[21] M. B. Giles, N. A. Pierce, Adjoint Error Correction for Integral Outputs, Springer Berlin Heidelberg, Berlin, Heidelberg, 2003, Ch. 2, pp. 47 – 95. doi:10.1007/978-3-662-05189-4_2.
URL http://dx.doi.org/10.1007/978-3-662-05189-4_2

[22] N. A. Pierce, M. B. Giles, Adjoint recovery of superconvergent functionals from PDE approximations, SIAM Review 42 (2) (2000) 247 – 264. doi:10.1137/s0036144598349423.
URL http://dx.doi.org/10.1137/S0036144598349423

[23] A. Hay, M. Visonneau, Error estimation using the error transport equation for finite-volume methods and arbitrary meshes, International Journal of Computational Fluid Dynamics 20 (7) (2006) 463 – 479. doi:10.1080/10618560600835934.
URL http://dx.doi.org/10.1080/10618560600835934

[24] J. Banks, J. Hittinger, J. Connors, C. Woodward, Numerical error estimation for non-linear hyperbolic PDEs via nonlinear error transport, Computer Methods in Applied Mechanics and Engineering 213-216 (2012) 1 – 15. doi:10.1016/j.cma.2011.11.021.
URL http://dx.doi.org/10.1016/j.cma.2011.11.021

[25] G. Yan, C. F. Ollivier-Gooch, Accuracy of discretization error estimation by the error transport equation on unstructured meshes - Nonlinear systems of equations, in: 22nd AIAA Computational Fluid Dynamics Conference, American Institute of Aeronautics and Astronautics (AIAA), 2015. doi:10.2514/6.2015-2747.
URL http://dx.doi.org/10.2514/6.2015-2747

[26] J. Banks, T. Aslam, W. Rider, On sub-linear convergence for linearly degenerate waves in capturing schemes, Journal of Computational Physics 227 (14) (2008) 6985 – 7002. doi:10.1016/j.jcp.2008.04.002.
URL http://dx.doi.org/10.1016/j.jcp.2008.04.002

[27] C. Roy, Strategies for driving mesh adaptation in CFD (invited), in: 47th AIAA Aerospace Sciences Meeting including the New Horizons Forum and Aerospace Exposition, American Institute of Aeronautics and Astronautics (AIAA), 2009. doi:10.2514/6.2009-1302.
URL http://dx.doi.org/10.2514/6.2009-1302

[28] W. L. Oberkampf, C. J. Roy, Verification and Validation in Scientific Computing, Cambridge University Press (CUP), 2010. doi:10.1017/cbo9780511760396.
URL http://dx.doi.org/10.1017/CBO9780511760396

[29] A. Griewank, On automatic differentiation and algorthimic linearization, Pesquisa Operacional 34 (3) (2014) 621 – 645. doi:10.1590/0101-7438.2014.034.03.0621.
URL https://doi.org/10.1590%2F0101-7438.2014.034.03.0621

[30] J. M. Derlaga, M. A. Park, S. K. Rallabhandi, Application of exact error transport equations and adjoint error estimation to AIAA workshops, in: 55th AIAA Aerospace Sciences Meeting, American Institute of Aeronautics and Astronautics (AIAA), 2017, p. 0076. doi:10.2514/6.2017-0076.
URL https://doi.org/10.2514/6.2017-0076

[31] T. Shih, B. Williams, Development and evaluation of an a posteriori method for estimating and correcting grid-induced errors in solutions of the Navier-Stokes equations, in: 47th AIAA Aerospace Sciences Meeting including the New Horizons Forum and Aerospace Exposition, American Institute of Aeronautics and Astronautics (AIAA), 2009. doi:10.2514/6.2009-1499.
URL https://doi.org/10.2514%2F6.2009-1499

[32] F. Fraysse, J. de Vicente, E. Valero, The estimation of truncation error by τ-estimation revisited, Journal of Computational Physics 231 (9) (2012) 3457 – 3482. doi:10.1016/j.jcp.2011.09.031.
URL http://dx.doi.org/10.1016/j.jcp.2011.09.031

[33] M. Park, Adjoint-based, three-dimensional error prediction and grid adaptation, in: 32nd AIAA Fluid Dynamics Conference and Exhibit, American Institute of Aeronautics and Astronautics (AIAA), 2002. doi:10.2514/6.2002-3286.
URL http://dx.doi.org/10.2514/6.2002-3286

[34] M. Nemec, M. Aftosmis, Adjoint error estimation and adaptive refinement for embedded-boundary Cartesian meshes, in: 18th AIAA Computational Fluid Dynamics Conference, American Institute of Aeronautics and Astronautics (AIAA), 2007. doi:10.2514/6.2007-4187.
URL http://dx.doi.org/10.2514/6.2007-4187

[35] B. Diskin, J. L. Thomas, Notes on accuracy of finite-volume discretization schemes on irregular grids, Applied Numerical Mathematics 60 (3) (2010) 224 – 226. doi:10.1016/j.apnum.2009.12.001.
URL http://dx.doi.org/10.1016/j.apnum.2009.12.001

[36] M. Sharbatdar, A. Jalali, C. Ollivier-Gooch, Smoothed truncation error in functional error estimation and correction using adjoint methods in an unstructured finite-volume solver, Computers & Fluids 140 (2016) 406–421. doi:10.1016/j.compfluid.2016.10.019.
URL https://doi.org/10.1016/j.compfluid.2016.10.019

[37] N. A. Pierce, M. B. Giles, Adjoint and defect error bounding and correction for functional estimates, Journal of Computational Physics 200 (2) (2004) 769 – 794. doi:10.1016/j.jcp.2004.05.001.
URL http://dx.doi.org/10.1016/j.jcp.2004.05.001

[38] T. Barth, Recent developments in high order k-exact reconstruction on unstructured meshes, in: 31st Aerospace Sciences Meeting, American Institute of Aeronautics and Astronautics (AIAA), 1993. doi:10.2514/6.1993-668.
URL http://dx.doi.org/10.2514/6.1993-668

[39] C. Ollivier-Gooch, M. Van Altena, A high-order-accurate unstructured mesh finite-volume scheme for the advection-diffusion equation, Journal of Computational Physics 181 (2) (2002) 729 – 752. doi:10.1006/jcph.2002.7159.
URL http://dx.doi.org/10.1006/jcph.2002.7159

[40] C. Ollivier-Gooch, A. Nejat, K. Michalak, Obtaining and verifying high-order unstructured finite volume solutions to the Euler equations, AIAA Journal 47 (9) (2009) 2105 – 2120. doi:10.2514/1.40585.
URL http://dx.doi.org/10.2514/1.40585

[41] G. H. Golub, C. Van Loan, Matrix Computations, 3rd Edition, Johns Hopkins University Press, 1996.

[42] T. Barth, P. Frederickson, Higher order solution of the Euler equations on unstructured grids using quadratic reconstruction, in: 28th Aerospace Sciences Meeting, American Institute of Aeronautics and Astronautics (AIAA), 1990. doi:10.2514/6.1990-13.
URL http://dx.doi.org/10.2514/6.1990-13

[43] W. C. Tyson, C. J. Roy, A hybrid adjoint/error transport approach to error estimation, adaptation, and higher-order solutions for computational fluid dynamics, in: 55th AIAA Aerospace Sciences Meeting, American Institute of Aeronautics and Astronautics (AIAA), 2017. doi:10.2514/6.2017-0741.
URL https://doi.org/10.2514%2F6.2017-0741

[44] C. Hirsch, Numerical Computation of Internal and External Flows: Volume 2, John Wiley & Sons Ltd., 1990.

[45] K. Masatsuka, I do like CFD, Vol. 1, Governing Equations and Exact Solutions, 2nd Edition, Cradle, 2013.

[46] C. W. Jackson, C. J. Roy, A multi-mesh CFD technique for adaptive mesh solutions, in: 53rd AIAA Aerospace Sciences Meeting, American Institute of Aeronautics and Astronautics (AIAA), 2015. doi:10.2514/6.2015-1958.
URL http://dx.doi.org/10.2514/6.2015-1958

[47] P. Roe, Approximate Riemann solvers, parameter vectors, and difference schemes, Journal of Computational Physics 43 (2) (1981) 357 – 372. doi:10.1016/0021-9991(81)90128-5.
URL http://dx.doi.org/10.1016/0021-9991(81)90128-5

[48] B. van Leer, Towards the ultimate conservative difference scheme. V. a second-order sequel to Godunov's method, Journal of Computational Physics 32 (1) (1979) 101 – 136. doi:10.1016/0021-9991(79)90145-1.
URL http://dx.doi.org/10.1016/0021-9991(79)90145-1

[49] R. M. Beam, R. Warming, An implicit finite-difference algorithm for hyperbolic systems in conservation-law form, Journal of Computational Physics 22 (1) (1976) 87 – 110. doi:10.1016/0021-9991(76)90110-8.
URL http://dx.doi.org/10.1016/0021-9991(76)90110-8

Chapter 4

A Novel Reconstruction Technique for Finite-Volume Truncation Error Estimation

William C. Tyson,[a] Christopher J. Roy,[a] Carl F. Ollivier-Gooch[b]

[a] Kevin T. Crofton Department of Aerospace and Ocean Engineering, Virginia Tech
215 Randolph Hall, 460 Old Turner Street, Blacksburg, VA, 24061

[b] Department of Mechanical Engineering, University of British Colombia
2054-6250 Applied Science Lane, Vancouver, B.C., V6T 1Z4, Canada

Keywords: Truncation error estimation, Discretization error estimation, Finite-volume method, Smoothing spline

Attribution

- William C. Tyson (First Author): The first author served as the main contributor and primary author of the work. The first author extended the smoothing spline reconstruction algorithm to finite-volume TE estimation and performed all comparison studies with the k-exact reconstruction. Additionally, the first author provided tools for assessing the quality of TE estimates.
- Christopher J. Roy (Second Author): The second author provided valuable guidance and insight during the course of the study. The second author also provided helpful comments on the manuscript.
- Carl F. Ollivier-Gooch (Third Author): The third author provided valuable guidance

and insight during the course of the study. The third author also provided helpful comments on the manuscript.

Abstract

Higher-order discretization error estimates yield asymptotically tighter error bounds than conventional error estimates and can even be used to correct the entire primal solution to higher-order. However, higher-order error estimates are difficult to obtain, especially for grids with non-smooth variations in mesh quality. In this work, we demonstrate that convergence rates for error estimates deteriorate when the reconstruction matches the underlying finite-volume data as closely as possible. Grid-induced solution variations on the order of discretization error are exactly captured and induce small, non-physical oscillations in the reconstruction. Consequently, the accuracy of truncation error estimates, and ultimately discretization error estimates, degrades due to excessively large inter-element jumps in the reconstruction. To improve the accuracy of truncation error estimates, a new polyharmonic smoothing spline reconstruction is proposed which employs regularization to balance fidelity to the data with smoothness of the fit. The amount of added smoothness is automatically regulated using a variant of the generalized cross-validation criterion. The resulting truncation error and discretization error estimates are significantly more accurate than those obtained with a standard k-exact least-squares reconstruction.

4.1 Introduction

The numerical simulation of physical processes has become an integral part of the engineering design process as simulations are often faster and cheaper than experiments. Numerical simulations can provide insight into physical phenomena that may be difficult to measure and enable rapid exploration of the design space. However, numerical simulations inherently contain errors that impact the fidelity of results. To minimize risk, especially in the design of safety-critical systems, simulation errors must be accurately quantified. The errors in a given simulation can generally be categorized as either modeling errors or numerical errors. Modeling errors arise from a mathematical model that incompletely describes the underlying physics. Numerical errors are generated by partial convergence of nonlinear iterative procedures, finite-precision arithmetic, and the discretization of the mathematical model. This work focuses on numerical error, in particular, errors associated with discretization.

Discretization error, or the difference between the numerical solution and the exact solution to the mathematical model, is typically the dominant error in a given simulation and can be difficult to estimate as it is a highly nonlinear function of the grid, boundary conditions, and the chosen discretization scheme. Recent efforts to estimate discretization error have focused on attaining higher-order error estimates, or error estimates which converge to the true error

at a faster rate than the formal order of accuracy of the discretization [1, 2, 3]. Higher-order error estimates can drastically reduce the uncertainty in a simulation result, and, if desired, may even be used to correct the entire discrete solution to higher-order [3]. Popular approaches to higher-order error estimation include adjoint methods for discretization errors in output functionals [4, 5] and error transport methods for discretization errors in state variables [6, 7, 8].

Most discretization error estimators, especially higher-order error estimators, require an accurate estimate of truncation error. Truncation error is defined as the functional difference between the continuous and discrete forms of the mathematical model and is the local source of discretization error in numerical solutions [9, 10]. Within the context of finite-volume discretizations, higher-order residual methods are arguably one of the most widely used and accurate truncation error estimation techniques available. Truncation error is estimated by reconstructing the discrete solution (i.e., control volume averages) to a higher-order piecewise continuous space and evaluating a higher-order discrete residual operator. Higher-order residual methods perform well for grids with smoothly varying grid metrics, such as uniform or curvilinear grids, and are able to provide asymptotically higher-order discretization error estimates [3, 1]. However, on grids with non-smoothly varying grid metrics, such as general unstructured grids, the higher-order properties of the discretization error estimate degrade to the same order of accuracy as the primal problem. This behavior is rather unfortunate as a large percentage of numerical simulations, if not most, are performed on unstructured grids.

In this work, the authors demonstrate that the loss of higher-order accuracy is a direct result of an insufficiently accurate truncation error estimate. Moreover, the authors show that a key assumption of higher-order residual methods is being violated, namely that the reconstructed solution does not sufficiently approximate the exact continuous solution. In comparison to structured grids, unstructured grid truncation error is asymptotically larger and is typically non-smooth [11]. Similarly, the discrete solution can exhibit non-smooth variations on the order of discretization error. Reconstructions which seek to match the data as closely possible, such as k-exact [12], will capture this numerical noise and may yield a reconstruction that is overly oscillatory. For grids with non-smooth grid metrics and, therefore, non-smooth discretization error, the reconstruction used for truncation error estimation should balance fidelity to the data with smoothness of the fit.

A new reconstruction technique is proposed to improve the accuracy of finite-volume truncation error estimates. The efforts outlined here are motivated by the works of Giles and Süli [13] and Giles [14] where smooth solution approximations were obtained by solving a biharmonic reconstruction equation. Sharbatdar and Ollivier-Gooch suggested a similar approach in the context of adjoint-based adaptation [15]. The proposed reconstruction method is based on polyharmonic smoothing splines and is similar to traditional least-squares reconstructions except that a regularization term is added to penalize non-smooth fits of the data. The amount of regularization is determined using a variant of the generalized cross-validation criterion [16]. Truncation error estimates are computed using a higher-order discrete resid-

ual operator. Furthermore, discretization error estimates are obtained via a linearized error transport equation. Continuity and smoothness requirements for the reconstruction are discussed. Results are compared to a standard k-exact least-squares reconstruction.

4.2 Background

4.2.1 Finite-Volume Discretization

In many engineering applications, the physical phenomenon governing the state of the system may be modeled by a general conservation law of the form

$$\frac{\partial \tilde{\mathbf{u}}}{\partial t} + \nabla \cdot \mathbb{F}(\tilde{\mathbf{u}}) = \mathbf{s}(\tilde{\mathbf{u}}) \tag{4.1}$$

where $\tilde{\mathbf{u}}$ are conserved quantities, such as mass or momentum, $\mathbb{F}(\cdot)$ is a flux tensor, and $\mathbf{s}(\cdot)$ is a source term. Conservation laws may also be functions of $\nabla\tilde{\mathbf{u}}$; however, without loss of generality, this dependency has been neglected to simplify notation. The exact continuous solution, $\tilde{\mathbf{u}}$, is defined for a domain $\Omega \subset \mathbb{R}^d \times [0,\infty), \mathbf{x} \in \mathbb{R}^d, d \in \{1,2,3\}$. To obtain a numerical solution to Eq. 4.1, the solution domain is discretized into a set of non-overlapping control volumes, $\mathcal{T}_h$, such that

$$\Omega = \bigcup_{i=1}^{N_v} \Omega_i, \quad \Omega_i \in \mathcal{T}_h \tag{4.2}$$

where N_v is the total number of control volumes in the computational mesh.

To admit weak solutions, Eq. 4.1 is cast into an integral form by integrating over each control volume, Ω_i, as follows

$$\mathbf{N}(\tilde{\mathbf{u}}) \coloneqq \frac{\partial}{\partial t}\int_{\Omega_i} \tilde{\mathbf{u}}\; d\Omega + \oint_{\partial\Omega_i} \mathbb{F}(\tilde{\mathbf{u}}) \cdot \hat{\mathbf{n}}\; dS - \int_{\Omega_i} \mathbf{s}(\tilde{\mathbf{u}})\; d\Omega = \mathbf{0} \tag{4.3}$$

The Gauss divergence theorem has been used to convert the flux divergence integral into an integral over the surface of the control volume. Furthermore, it has been assumed that each control volume is fixed in space so that, by the Reynolds transport theorem, the time derivative may be pulled outside of the first integral. In the finite-volume method, a numerical solution, $\mathbf{u}_h$, is sought which approximates the control volume average of $\tilde{\mathbf{u}}$. In accordance with this assumption, the discretized form of Eq. 4.3 may be written as follows

$$\mathbf{N}_h^p(\mathbf{u}_h) \coloneqq |\Omega_i|\frac{d\mathbf{u}_h}{dt} + \oint_{\partial\Omega_i} \mathbb{F}_h\Big(I_h^p\mathbf{u}_h^+, I_h^p\mathbf{u}_h^-\Big) \cdot \hat{\mathbf{n}}\; dS - \int_{\Omega_i} \mathbf{s}(I_h^p\mathbf{u}_h)\; d\Omega = \mathbf{0} \tag{4.4}$$

where $|\Omega_i|$ is the volume of cell Ω_i. The operator I_a^b is a restriction/prolongation operator used to project the solution from one space to another; the subscript a is the starting space and the superscript b is the resultant space. An empty subscript or superscript is used to denote

a C^∞ continuous space. Likewise, a subscript or superscript h is used to denote a discrete space on $\mathcal{T}_h$. Higher-order spatial accuracy is achieved by reconstructing the discrete solution to a p^{th} order piecewise continuous space. The reconstructed solution, $I_h^p \mathbf{u}_h$, is defined locally in each control volume and is discontinuous across control volume boundaries. The notation $(\cdot)^{+/-}$ is used to denote the trace of the reconstructed solution from the exterior and interior of Ω_i, respectively. To account for the discontinuous representation of the solution, the analytic flux is discretized with a flux function, $\mathbb{F}_h(\cdot,\cdot)$. Eq. 4.4 may be written for each control volume to form the following system of ordinary differential equations (ODEs)

$$|\mathbf{\Omega}|\frac{d\mathbf{u}_h}{dt} + \mathbf{R}(\mathbf{u}_h) = \mathbf{0} \tag{4.5}$$

where $\mathbf{R}(\mathbf{u}_h)$ is referred to as the steady-state residual. Eq. 4.5 may be discretized in time using any ODE time integrator. This discretization scheme, which separately discretizes the spatial and temporal dimensions, is referred to as the method of lines.

4.2.2 Discretization Error Estimation

The numerical accuracy of the discrete solution, $\mathbf{u}_h$, can be assessed by quantifying the discretization error,

$$\varepsilon_h^{\hat{p}} = \mathbf{u}_h - I^h\tilde{\mathbf{u}} \ . \tag{4.6}$$

The operator I^h is used to restrict the exact continuous solution to $\mathcal{T}_h$, and, in a finite-volume context, corresponds to the cell-averaging operator,

$$I^h(\cdot) = \frac{1}{|\Omega_i|}\int_{\Omega_i}(\cdot)\ d\Omega \ . \tag{4.7}$$

Discretization error converges toward zero at a rate $\hat{p}$ which is identical to the formal order of accuracy of the discretization, p, for smooth problems, but is often lower when geometric and flowfield discontinuities are present [17]. Since $\tilde{\mathbf{u}}$ is not generally known, discretization error cannot be directly computed, and must instead be estimated.

Historically, discretization error has been estimated by comparing the numerical solution on multiple grid levels using techniques such as Richardson extrapolation [18]. However, these multiple-grid methods are computationally expensive and impractical for most engineering applications. In this work, discretization error is estimated by deriving and solving an auxiliary transport equation. Consider the following Generalized Truncation Error Expression (GTEE) [10, 19]

$$\mathbf{N}_h^p(I^h\mathbf{u}) = \mathbf{N}(\mathbf{u}) + \tau_h^p(\mathbf{u}) \tag{4.8}$$

where $\mathbf{u}$ is a general continuous function and τ_h^p is the truncation error for a p^{th} order accurate discretization. Truncation error is simply a function describing the difference between the discrete operator, $\mathbf{N}_h^p(\cdot)$, and the continuous operator, $\mathbf{N}(\cdot)$, and can be viewed as higher-order terms which are neglected during the discretization process. By inserting $\tilde{\mathbf{u}}$ into Eq. 4.8

and substituting Eq. 4.6, a set of nonlinear error transport equations (ETE) for the local discretization error may be defined as follows

$$\begin{aligned} \mathbf{N}_h^q(\mathbf{u}_h - \varepsilon_h^{\hat{p}}) &= \cancelto{\mathbf{0}}{\mathbf{N}_h^q(\tilde{\mathbf{u}})} + \tau_h^q(\tilde{\mathbf{u}}) \\ &= \tau_h^q(\tilde{\mathbf{u}}) \ . \end{aligned} \tag{4.9}$$

It is assumed that the ETE are discretized at an order $q \geq p$. The ETE demonstrate that truncation error acts as the local source for discretization error in numerical solutions. Furthermore, discretization error is generated, convected, and diffused by the same physical processes governing the discrete solution.

To minimize the computational cost associated with solving the ETE, it is common to derive a simplified form of Eq. 4.9. In this work, discretization error is estimated using a linearized form of the ETE. Consider the following Newton linearization of the discrete operator

$$\begin{aligned} \mathbf{N}_h^p(I^h\tilde{\mathbf{u}}) &= \cancelto{\mathbf{0}}{\mathbf{N}_h^p(\mathbf{u}_h)} - \frac{\partial \mathbf{N}_h^p}{\partial \mathbf{u}_h}\varepsilon_h^{\hat{p}} + O\left(\left\|\varepsilon_h^{\hat{p}}\right\|^2\right) \\ &= -\frac{\partial \mathbf{N}_h^p}{\partial \mathbf{u}_h}\varepsilon_h^{\hat{p}} + O\left(\left\|\varepsilon_h^{\hat{p}}\right\|^2\right) \end{aligned} \tag{4.10}$$

where it is assumed that $q = p$ and that the discrete solution is fully converged. Using the GTEE, the left-hand side of Eq. 4.10 can easily be shown to be equivalent to the truncation error. By inserting the truncation error and rearranging, Eq. 4.10 may be rewritten as

$$\frac{\partial \mathbf{N}_h^p}{\partial \mathbf{u}_h}\varepsilon_h^{\hat{p}} = -\tau_h^p(\tilde{\mathbf{u}}) + O\left(\left\|\varepsilon_h^{\hat{p}}\right\|^2\right). \tag{4.11}$$

By neglecting higher-order terms, the linearized ETE may be defined as

$$\frac{\partial \mathbf{N}_h^p}{\partial \mathbf{u}_h}\bar{\varepsilon}_h^{\hat{p}} = -\tau_h^p(\tilde{\mathbf{u}}) \tag{4.12}$$

where $\bar{\varepsilon}_h^{\hat{p}}$ is an estimate of the local discretization error. For steady problems, the left-hand side Jacobian simplifies to the Jacobian of the steady-state residual. The left-hand side Jacobian must correspond to the exact linearization of the discrete operator; if any approximations are made when formulating the Jacobian, the order of accuracy of the error estimate may be reduced. The linearized ETE are preferred over the nonlinear ETE because they are computationally cheaper; for steady problems, a nonlinear solve is replaced by the solution of one additional linear system.

4.2.3 Truncation Error Estimation

Before the linearized ETE can be solved, the truncation error, $\tau_h^p(\tilde{\mathbf{u}})$, must be determined. However, since the exact continuous solution, $\tilde{\mathbf{u}}$, is generally not known, truncation error

cannot be directly computed and must instead be estimated. In this paper, a higher-order discrete residual method is used to estimate the leading order terms in the truncation error. First, the discrete solution is prolonged to a r^{th} order continuous space. It is assumed that $I_h^r \mathbf{u}_h$ sufficiently approximates $\tilde{\mathbf{u}}$ such that

$$\|\tau_h^p(\tilde{\mathbf{u}}) - \tau_h^p(I_h^r \mathbf{u}_h)\| = O(h^r) \ . \tag{4.13}$$

Using the GTEE (Eq. 4.8), an estimate of the truncation error may be formulated as follows

$$\begin{aligned} \tau_h^p(I_h^r \mathbf{u}_h) &= \mathbf{N}_h^p\Big(I_r^h I_h^r \mathbf{u}_h\Big) - \mathbf{N}(I_h^r \mathbf{u}_h) \\ &= \cancelto{\mathbf{0}}{\mathbf{N}_h^p(\mathbf{u}_h)} - \ \mathbf{N}(I_h^r \mathbf{u}_h) \\ &= -\mathbf{N}(I_h^r \mathbf{u}_h) \end{aligned} \tag{4.14}$$

where $\tau_h^p(I_h^r \mathbf{u}_h)$ is found by operating the continuous residual onto the reconstructed solution. The following requirements can be inferred from the derivation of Eq. 4.14:

1. The reconstruction operator should satisfy conservation of the mean such that

$$\mathbf{u}_h = I_r^h I_h^r \mathbf{u}_h \ . \tag{4.15}$$

2. The discrete solution should be converged to machine precision such that

$$\mathbf{N}_h^p(\mathbf{u}_h) = \mathbf{0} \ . \tag{4.16}$$

3. The reconstructed solution should be sufficiently representative of the exact continuous solution, i.e.:

   - $I_h^r \mathbf{u}_h$ should satisfy the continuous boundary conditions.
   - $I_h^r \mathbf{u}_h$ should satisfy any continuity / smoothness requirements imposed by the PDE or $\tilde{\mathbf{u}}$.

Conservation of the mean is naturally satisfied by most finite-volume reconstruction methods. If, however, the reconstruction does not conserve the mean, the p^{th} order discrete residual contribution that was neglected in Eq. 4.14 must be taken into account. Similarly, if the solution is not fully converged, the p^{th} order discrete residual contribution is required. In the case of a partially converged solution, it is unclear if accounting for the discrete residual term will be sufficient to provide an accurate truncation error estimate. Since the effects of iterative error are not the focus of this work, this investigation is left for future work. For more information regarding the effects of iterative error on truncation error estimates, refer to Ref. [20].

The last requirement is possibly the most difficult to satisfy. Imposing the continuous boundary conditions on the reconstructed solution is challenging, especially for boundary conditions which couple different state variables. Boundary condition enforcement can typically be managed if the boundary conditions are linear functions of the state variables [21]. Nonlinear boundary conditions may potentially be enforced if they can be properly linearized or if specialized basis functions can be devised. The continuity and smoothness of the reconstructed solution is largely dictated by the form of the reconstruction operator and the data being fit. A continuous representation of the solution can be crafted if the reconstruction operator globally couples each control volume in the domain. A global reconstruction, however, can be quite burdensome to implement since most finite-volume codes only have the machinery to reconstruct the solution on a local patch of cells. Furthermore, a global reconstruction can be computationally expensive and impractical for large problems. In practice, it is common to relax the continuity requirements by instead performing local reconstructions in each control volume and approximating the continuous residual by a higher-order discrete residual [22, 2],

$$\tau_h^p(I_h^r \mathbf{u}_h) = -\mathbf{N}(I_h^r \mathbf{u}_h) \approx -\mathbf{N}_h^r(\mathbf{u}_h), \quad \forall\ r > p\ . \tag{4.17}$$

The condition $r > p$ is imposed to prevent a vanishing truncation error estimate when $r = p$ and to ensure that the truncation error estimate is asymptotically correct. It is assumed that the reconstruction is sufficiently smooth so that inter-element jumps are at most $O(h^r)$. The accuracy of each truncation error estimate is evaluated through qualitative comparisons with the exact truncation error,

$$\tau_h^p(\tilde{\mathbf{u}}) = \mathbf{N}_h^p\Big(I^h \tilde{\mathbf{u}}\Big)\ , \tag{4.18}$$

and by monitoring the effectivity index,

$$\theta = \frac{\|\tau_h(I_h^r \mathbf{u}_h)\|_2}{\|\tau_h(\tilde{\mathbf{u}})\|_2}\ , \tag{4.19}$$

where $\theta \to 1.0$ with grid refinement for an asymptotically correct truncation error estimate.

4.2.4 Expected Order of Accuracy for Discretization Error Estimates

Within in the context of continuous residual truncation error estimation, Banks et al. [23] derive the expected order of accuracy for ETE error estimates. The expected order of accuracy for the full nonlinear ETE is given as follows

$$\left\|\bar{\varepsilon}_h^{\hat{p}} - \varepsilon_h^{\hat{p}}\right\| = O\Big(h^{\min(p+q,r)}\Big) \tag{4.20}$$

where p is the formal order of accuracy of the primal problem, q is the order of accuracy of the ETE, and r is the order of accuracy of the truncation error estimate. Higher-order

error estimates may be obtained for $q \geq 1$ and $r > p$. By linearizing the ETE, an additional $O(h^{2p})$ error is introduced. Therefore, the expected order of accuracy for the linearized ETE may be written as

$$\left\| \bar{\varepsilon}_h^{\hat{p}} - \varepsilon_h^{\hat{p}} \right\| = O\left(h^{\min(2p, p+q, r)}\right) . \tag{4.21}$$

Eq. 4.20 and Eq. 4.21 are valid for smooth problems and serve as a limiting case for problems with weak solutions or strong nonlinearities. Within the context of higher-order discrete residual truncation error estimation, Yan and Ollivier-Gooch [7, 24] show that Eq. 4.20 and Eq. 4.21 hold for grids with smoothly varying grid metrics. However, on general unstructured grids, the expected order of accuracy for ETE error estimates is empirically found to be

$$\left\| \bar{\varepsilon}_h^{\hat{p}} - \varepsilon_h^{\hat{p}} \right\| = O\left(h^{p+(r-p)\delta_{qr}}\right) \tag{4.22}$$

where δ_{qr} is the Kronecker delta. Higher-order error estimates can be obtained for $q = r$ and $r > p$. Eq. 4.22 requires the ETE to be discretized at the same order as the truncation error estimate. We speculate that this additional constraint may result from the characteristically non-smooth behavior of discretization error on unstructured grids. In this work, linearized ETE error estimates are referred to using the ordered triplet $(p, q, r)_L$.

4.3 Methodology

Truncation error is estimated by reconstructing the discrete solution, $\mathbf{u}_h$, to a piecewise continuous function and performing a higher-order discrete residual evaluation. The reconstructed solution is defined locally within each control volume. In this work, two reconstruction methods are examined: (1) a standard k-exact least-squares reconstruction and (2) a novel polyharmonic smoothing spline reconstruction.

4.3.1 *k*-Exact Least-Squares Reconstruction

The k-exact reconstruction [12, 25] is arguably the most common reconstruction technique used for finite-volume methods. The reconstruction is referred to as k-exact since it reconstructs polynomials of degree k exactly. Therefore, for any smooth function, $\mathbf{u}(\mathbf{x})$, the error in the reconstruction converges according to the following

$$\left\| \mathbf{u}_i^R(\mathbf{x}) - \mathbf{u}(\mathbf{x}) \right\| = O\left(h^{k+1}\right) . \tag{4.23}$$

The reconstructed function, $\mathbf{u}_i^R(\cdot)$, is defined in each control volume as a linear combination of basis functions

$$\mathbf{u}_i^R(\mathbf{x} - \mathbf{x}_i) = \sum_{\nu=0}^{N_t - 1} \boldsymbol{\alpha}_\nu \phi_\nu(\mathbf{x} - \mathbf{x}_i) \tag{4.24}$$

where $\boldsymbol{\alpha}_\nu$ are coefficients, ϕ_ν are a set of basis functions, N_t is the total number of basis functions, and $\mathbf{x}_i$ is the centroid of Ω_i. For k-exact reconstructions, the basis used to represent the reconstructed function is typically the monomial basis

$$\phi(\mathbf{x}-\mathbf{x}_i) = \left\{ (x-x_i)^a (y-y_i)^b (z-z_i)^c \;\middle|\; a+b+c \leq k \right\} . \tag{4.25}$$

By using the monomial basis, the reconstructed function can be interpreted as a Taylor series expansion about the control volume centroid. The number of basis functions for a given reconstruction degree, k, is given by

$$N_t = \frac{1}{N_d!} \prod_{d=1}^{N_d} (k+d) \tag{4.26}$$

where N_d is the number of spatial dimensions of the problem.

The reconstruction in each control volume is uniquely determined by enforcing conservation of the mean in Ω_i as well as in a stencil of neighboring control volumes, Ω_j,

$$\frac{1}{|\Omega_i|} \int_{\Omega_i} \mathbf{u}_i^R(\mathbf{x}-\mathbf{x}_i)\, d\Omega = \sum_{\nu=0}^{N_t-1} \boldsymbol{\alpha}_\nu \overline{\phi}_\nu(\mathbf{x}-\mathbf{x}_i) = \mathbf{u}_{h,i} \tag{4.27a}$$

$$\frac{1}{|\Omega_j|} \int_{\Omega_j} \mathbf{u}_i^R(\mathbf{x}-\mathbf{x}_i)\, d\Omega = \sum_{\nu=0}^{N_t-1} \boldsymbol{\alpha}_\nu \hat{\phi}_\nu(\mathbf{x}-\mathbf{x}_i) = \mathbf{u}_{h,j} \ . \tag{4.27b}$$

The terms $\bar{\phi}(\cdot)$ and $\hat{\phi}(\cdot)$ are referred to as control volume moments and relative control volume moments, respectively, and correspond to volume integrals of basis functions over each control volume in the stencil. The control volume moments and relative control volume moments are given by the following

$$\overline{\phi}(\mathbf{x}-\mathbf{x}_i) = \left\{ \overline{x^a y^b z^c}_i \;\middle|\; a+b+c \leq k \right\} \tag{4.28a}$$

$$\hat{\phi}(\mathbf{x}-\mathbf{x}_i) = \left\{ \widehat{x^a y^b z^c}_{ij} \;\middle|\; a+b+c \leq k \right\} \tag{4.28b}$$

where

$$\overline{x^a y^b z^c}_i = \frac{1}{|\Omega_i|} \int_{\Omega_i} (x-x_i)^a (y-y_i)^b (z-z_i)^c \, d\Omega \tag{4.29a}$$

$$\begin{aligned} \widehat{x^a y^b z^c}_{ij} &= \frac{1}{|\Omega_j|} \int_{\Omega_j} (x-x_i)^a (y-y_i)^b (z-z_i)^c \, d\Omega \\ &= \frac{1}{|\Omega_j|} \int_{\Omega_j} [(x-x_j)+(x_j-x_i)]^a [(y-y_j)+(y_j-y_i)]^b [(z-z_j)+(z_j-z_i)]^c \, d\Omega \\ &= \sum_{l=0}^{a} \sum_{m=0}^{b} \sum_{n=0}^{c} \binom{a}{l} \binom{b}{m} \binom{c}{n} (x_j-x_i)^l (y_j-y_i)^m (z_j-z_i)^n \overline{x^{a-l} y^{b-m} z^{c-n}}_j \ . \end{aligned} \tag{4.29b}$$

The relative control volume moments, $\widehat{x^a y^b z^c}_{ij}$, are rewritten by adding and subtracting components of $\mathbf{x}_j$ [26]. In doing so, the overall number of required integrals is reduced; only those integrals associated with $\overline{x^a y^b z^c}_i$ need to be evaluated. Control volume moments in each Ω_i are computed and stored, and relative control volume moments are computed as needed. To further reduce the amount of computational effort, Eq. 4.29a is reformulated into a surface integral using the Gauss divergence theorem

$$\overline{x^a y^b z^c}_i = \frac{1}{|\Omega_i|} \frac{1}{(a+1)} \oint_{\partial\Omega_i} (x - x_i)^{a+1} (y - y_i)^b (z - z_i)^c \; n_x \; dS \tag{4.30}$$

where n_x is the component of the surface normal in the x-direction. All surface integrals are evaluated using Gaussian quadrature where the quadrature order is set high enough so as to not introduce any integration errors.

A linear system for the reconstruction coefficients, $\boldsymbol{\alpha}_\nu$, may now be written as follows

$$\begin{bmatrix} w_{i1}(\widehat{x}_{i1} - \overline{x}_i) & w_{i1}(\widehat{y}_{i1} - \overline{y}_i) & w_{i1}(\widehat{z}_{i1} - \overline{z}_i) & \cdots \\ w_{i2}(\widehat{x}_{i2} - \overline{x}_i) & w_{i2}(\widehat{y}_{i2} - \overline{y}_i) & w_{i2}(\widehat{z}_{i2} - \overline{z}_i) & \cdots \\ \vdots & \vdots & \vdots & \ddots \\ w_{iN_s}(\widehat{x}_{iN_s} - \overline{x}_i) & w_{iN_s}(\widehat{y}_{iN_s} - \overline{y}_i) & w_{iN_s}(\widehat{z}_{iN_s} - \overline{z}_i) & \cdots \end{bmatrix} \begin{pmatrix} \frac{\partial u}{\partial x} \\ \frac{\partial u}{\partial y} \\ \frac{\partial u}{\partial z} \\ \vdots \end{pmatrix}_i = \begin{pmatrix} w_{i1}(u_{h,1} - u_{h,i}) \\ w_{i2}(u_{h,2} - u_{h,i}) \\ \vdots \\ w_{iN_s}(u_{h,N_s} - u_{h,i}) \end{pmatrix}. \tag{4.31}$$

where N_s is the number of control volumes in the stencil. For simplicity, Eq. 4.31 has only been written for a single component of $\mathbf{u}_i^R(\cdot)$. Conservation of the mean is enforced in Ω_i exactly by performing Gaussian elimination on the mean constraint row. The reconstruction coefficients have been written as derivatives of u to further illustrate the connection with Taylor series expansions. The reconstruction stencil is generated by growing outward from Ω_i using face connectivity information. We require 50% more control volumes in the stencil than what is required by the reconstruction degree, k [26]. Consequently, conservation of the mean is only enforced in Ω_j in a least-squares sense. Each stencil row has been multiplied by a weighting factor, $w_{ij} = \frac{1}{\|\mathbf{x}_i - \mathbf{x}_j\|}$, to emphasize geometrically close data. The coefficients $\{\frac{\partial u}{\partial x}, \frac{\partial u}{\partial y}, \frac{\partial u}{\partial z}, \ldots\}_i$ are computing using QR factorization [27]; the remaining coefficient $u|_i$ is computed by back-substitution. For more background information regarding the k-exact reconstruction operator, refer to Refs. [21, 26].

4.3.2 Polyharmonic Smoothing Spline Reconstruction

4.3.2.1 Formulation

The polyharmonic smoothing spline is a generalization of the well-known thin plate spline [28]. When noise is present in the data, this reconstruction seeks to balance fidelity to the data with smoothness of the fit. Polyharmonic smoothing splines are quite popular due to the

absence of free parameters and the ease with which spatial data of any number of dimensions may be reconstructed. Polyharmonic smoothing splines are used in a variety of applications including image reconstruction [29], the numerical solution of PDEs [30], and reduced-order modeling of physical systems [31].

Consider a data model of the form

$$y_i = \mathcal{L}_i f(\mathbf{x}_i) + \epsilon_i, \quad i = 1, \ldots, n \tag{4.32}$$

where $y = (y_1, \ldots, y_n)$ is the data to be reconstructed, $\mathcal{L}_1, \ldots, \mathcal{L}_n$ are bounded linear functionals, $\epsilon = (\epsilon_1, \ldots, \epsilon_n)' \sim \mathcal{N}(0, \sigma^2 I)$ are presumed random errors, $f(\mathbf{x}) \in \mathcal{H}_R$ is the underlying smooth function, and $\mathcal{H}_R$ is a reproducing kernel Hilbert space. An estimate of $f(\mathbf{x})$ is sought which minimizes the following

$$\frac{1}{n}\sum_{i=1}^{n}\Big(y_i - \mathcal{L}_i f(\mathbf{x}_i)\Big)^2 + \lambda J_m^d\Big(f\Big) \tag{4.33}$$

where $\lambda \geq 0$ is a smoothing parameter. The first term in Eq. 4.33 is a standard least-squares term meant to minimize the error between the data and the function estimate while the second term is used to apply smoothing and penalize roughness. For polyharmonic smoothing splines, the roughness penalty takes the following form

$$J_m^d\Big(f\Big) = \sum_{\alpha_1+\ldots+\alpha_d=m} \frac{m!}{\alpha_1!\ldots\alpha_d!} \int_{-\infty}^{+\infty} \cdots \int_{-\infty}^{+\infty} \left(\frac{\partial^m f}{\partial x_1^{\alpha_1}\ldots\partial x_d^{\alpha_d}}\right)^2 \prod_{j=1}^{d} dx_j \tag{4.34}$$

where d is the number of spatial dimensions and $m-1$ is the number of continuous derivatives in the function estimate. k-Exact reconstructions may be viewed as the limiting case when $\lambda = 0$ (i.e., when no smoothing is applied).

Duchon [32] showed that Eq. 4.33 has a unique minimizer, f_λ, of the form

$$f_\lambda(\mathbf{x}) = \sum_{\nu=1}^{p} d_\nu \phi_\nu(\mathbf{x}) + \sum_{i=1}^{n} c_i \mathcal{L}_{i(\mathbf{x})} E_m(\mathbf{x}, \mathbf{x}_i) \tag{4.35}$$

where $E_m(\cdot,\cdot)$ is a Green's function for the m^{th}-iterated Laplacian, $\mathbf{x}_i$ is the centroid of Ω_i, d_ν and c_i are coefficients, and $\phi_\nu(\mathbf{x})$ are the p polynomials of at most degree $m-1$. The Green's function, $E_m(\cdot,\cdot)$, is given by

$$E_m(\mathbf{x}_1, \mathbf{x}_2) = \begin{cases} \theta_{m,d}\|\mathbf{x}_1 - \mathbf{x}_2\|^{2m-d} & : d = \text{odd} \\ \theta_{m,d}\|\mathbf{x}_1 - \mathbf{x}_2\|^{2m-d}\log(\|\mathbf{x}_1 - \mathbf{x}_2\|) & : d = \text{even} \end{cases} \tag{4.36}$$

where $\mathbf{x}_1$ and $\mathbf{x}_2$ are coordinates in Euclidean space and $\theta_{m,d}$ are constants given by

$$\theta_{m,d} = \begin{cases} \frac{\Gamma\left(\frac{d}{2}-m\right)}{2^{2m}\pi^{\frac{d}{2}}(m-1)!} & : d = \text{odd} \\ \frac{(-1)^{\frac{d}{2}+1+m}}{2^{2m-1}\pi^{\frac{d}{2}}(m-1)!\left(m-\frac{d}{2}\right)!} & : d = \text{even} \end{cases} \tag{4.37}$$

where $\Gamma(\cdot)$ is the gamma function. In this work, the data being reconstructed, y_i, corresponds to control volume averages. Therefore, for consistency, $\mathcal{L}_{i(\mathbf{x})}$ is defined as the cell-averaging functional

$$\mathcal{L}_{i(\mathbf{x})} g(\mathbf{x}, \mathbf{z}) = \frac{1}{|\Omega_i|} \int_{\Omega_i} g(\mathbf{x}, \mathbf{z}) \; d\mathbf{x} \tag{4.38}$$

where $g(\mathbf{x}, \mathbf{z})$ is a general function of two spatial coordinates, $\mathbf{x}$ and $\mathbf{z}$. By inserting Eq. 4.35 into Eq. 4.33 and minimizing over the reconstruction coefficients, $\mathbf{c}$ and $\mathbf{d}$, the following linear system may be derived

$$\begin{aligned} (K + n\lambda I)\mathbf{c} + T\mathbf{d} &= \mathbf{y} \\ T'\mathbf{c} &= \mathbf{0} \end{aligned} \tag{4.39}$$

where

$$\begin{aligned} K &= \left\{ \mathcal{L}_{i(\mathbf{x}_1)} \mathcal{L}_{j(\mathbf{x}_2)} E_m(\mathbf{x}_1, \mathbf{x}_2) \right\}_{i=1 \; j=1}^{n \quad n} \\ T &= \left\{ \mathcal{L}_{i(\mathbf{x})} \phi_\nu(\mathbf{x}) \right\}_{\nu=1}^{p} . \end{aligned} \tag{4.40}$$

The side conditions $T'\mathbf{c} = \mathbf{0}$ ensure that $E_m(\cdot, \cdot)$ is conditionally positive definite so that $\mathbf{c}' K \mathbf{c} > 0$.

4.3.2.2 Numerical Solution

For a given value of the smoothing parameter, λ, Eq. 4.39 is solved for the reconstruction coefficients by performing a QR decomposition of the matrix T as follows

$$T = Q \begin{bmatrix} R \\ \mathbf{0} \end{bmatrix} = (Q_1 : Q_2) \begin{bmatrix} R \\ \mathbf{0} \end{bmatrix} \tag{4.41}$$

where $Q \in \mathbb{R}^{n \times n}$ is an orthogonal matrix, $Q_1 \in \mathbb{R}^{n \times p}$, $Q_2 \in \mathbb{R}^{n \times (n-p)}$, and $R \in \mathbb{R}^{p \times p}$ is an upper triangular matrix [27]. Now, using Eq. 4.41, the reconstruction coefficients are computed as

$$\begin{aligned} \mathbf{c} &= Q_2 (Q_2' M Q_2)^{-1} Q_2' \mathbf{y} \\ \mathbf{d} &= R^{-1} Q_1' (\mathbf{y} - M\mathbf{c}) \end{aligned} \tag{4.42}$$

where $M = K + n\lambda I$. In practice, Eq. 4.42 is evaluated by performing an additional singular value decomposition of the matrix $Q_2' M Q_2$.

4.3.2.3 Selection of Smoothing Parameter

When employing a polyharmonic smoothing spline, practitioners must ensure that the proper amount of smoothing is applied. If too little smoothing is applied, the reconstruction will capture all noise present in the data. On the other hand, if too much smoothing is applied, the reconstruction will not be representative of the underlying function. In general, the amount of smoothing to be applied should be data-dependent and should be selected in a

way that requires minimal user intervention. When the variance of the random errors in the data is known, methods such as *unbiased risk* or the *discrepancy principle* may be used [28]. However, in cases where the error variance is not known, another approach must be used.

In this work, we utilize a variant of the generalized cross-validation (GCV) criterion [16, 33, 28] to select the smoothing parameter, λ. The GCV criterion provides an estimate of the predictive mean-square error,

$$T(\lambda) = \frac{1}{n}\sum_{i=1}^{n}(\mathcal{L}_i f_\lambda - \mathcal{L}_i f)^2 \ , \tag{4.43}$$

and is given by the following

$$V(\lambda) = \frac{\frac{1}{n}\|(I - A(\lambda))y\|^2}{\left[\frac{1}{n}tr(I - A(\lambda))\right]^2} \tag{4.44}$$

where $I - A(\lambda) = n\lambda Q_2(Q_2'MQ_2)^{-1}Q_2$. The matrix $A(\lambda)$, which is commonly referred to as the *influence matrix*, serves as the transformation between the data and the function estimate, that is

$$\mathcal{L}_i f_\lambda(\mathbf{x}_i) = A(\lambda)y \ . \tag{4.45}$$

The smoothing parameter, λ, is typically selected as that which minimizes the GCV criterion. For small data sets, however, the minimum GCV criterion can occur near a smoothing parameter of $\lambda = 0$ and can lead to a reconstruction which undersmooths the data. To provide sufficient smoothing, we instead utilize the robust GCV criterion [34, 35] which takes the form

$$\overline{V}(\lambda) = [\gamma + (1-\gamma)\mu_2(\lambda)]V(\lambda), \quad \gamma \in [0,1] \tag{4.46}$$

where

$$\mu_2(\lambda) = n^{-1}tr(A(\lambda)^2) \ . \tag{4.47}$$

As $\gamma \to 1$, the standard GCV criterion is recovered. The optimal smoothing parameter is computed in each control volume by solving a simple 1D minimization problem. The minimization algorithm used in this work is Brent's method [36]. For low noise levels, it can be numerically difficult to determine the true minimum of Eq. 4.46. Consequently, the optimal smoothing parameter in adjacent control volumes may vary significantly. To prevent excessive jumps in the smoothing parameter, the smoothing parameter in each control volume is replaced by the mean value across the entire domain. This step ensures that the behavior of the reconstruction does not fluctuate too drastically from one control volume to the next.

4.4 Test Case

4.4.1 Quasi-1D Euler Equations

The quasi-1D Euler equations are a statement of conservation of mass, momentum, and energy for a fluid where the effects of viscosity, body forces, and heat conduction have been neglected. The domain of interest is one dimensional with a given area distribution, $A(x)$. The flow is assumed to be uniform in the transverse direction at each station along the domain. The quasi-1D Euler equations in strong, conservation form are given as follows

$$\frac{\partial\Big(A(x)\mathbf{u}\Big)}{\partial t} + \frac{\partial[A(x)\mathbf{F}(\mathbf{u})]}{\partial x} = \mathbf{s}(\mathbf{u}) \tag{4.48}$$

where $\mathbf{u}$ is the vector of conserved variables, $\mathbf{F}(\mathbf{u})$ is the vector of inviscid fluxes, and $\mathbf{s}(\mathbf{u})$ is a source term. The conserved variables, inviscid fluxes, and source term are given by the following

$$\mathbf{u} = \begin{bmatrix} \rho \\ \rho u \\ \rho e_t \end{bmatrix}, \quad \mathbf{F}(\mathbf{u}) = \begin{bmatrix} \rho u \\ \rho u^2 + p \\ \rho u h_t \end{bmatrix}, \quad \mathbf{s}(\mathbf{u}) = \begin{bmatrix} 0 \\ p\frac{dA}{dx} \\ 0 \end{bmatrix} \tag{4.49}$$

where ρ is the fluid density, u is the fluid velocity, p is static pressure, e_t is the total energy per unit mass, h_t is the total enthalpy per unit mass, and $\frac{dA}{dx}$ describes the variation of the cross-sectional area. The system of equations is closed using the equation of state for a perfect gas.

The quasi-1D Euler equations are used to model flow through a converging-diverging nozzle. The nozzle is characterized by a contraction through which the flow accelerates followed by a diverging section where the flow accelerates supersonically or decelerates subsonically depending upon the static pressure at the nozzle exit. The nozzle is defined by a Gaussian area distribution

$$A(x) = 1 - 0.8e^{\left(\frac{-x^2}{2\sigma^2}\right)}, \quad x \in [-1, +1] \tag{4.50}$$

where $\sigma = 0.2$ [37]. This test case has two isentropic, exact solutions: (1) fully subsonic flow and (2) fully supersonic flow through the diverging section. The Mach number distribution for both exact solutions as well as the area distribution may be seen in Figure 4.1. In the our experience, the conclusions which are drawn do not change from one exact solution to the other. Therefore, in this work, only the supersonic solution is examined.

The quasi-1D Euler equations are solved using a second order accurate, cell-centered, finite-volume discretization. Higher-order spatial accuracy is achieved using a fully-upwinded, physical space MUSCL reconstruction [38]. Inviscid fluxes are computed using Roe's flux difference splitting scheme [39]. The numerical solution is marched in pseudo-time to a steady-state using backward Euler time integration [40]. The quasi-1D Euler equations are solved in a nondimensional form to improve the scaling of the numerical procedure [41].

Boundary conditions are enforced weakly through the fluxes. At the inflow boundary, the stagnation pressure and stagnation temperature are set to 300 kPa and 600 K, respectively.

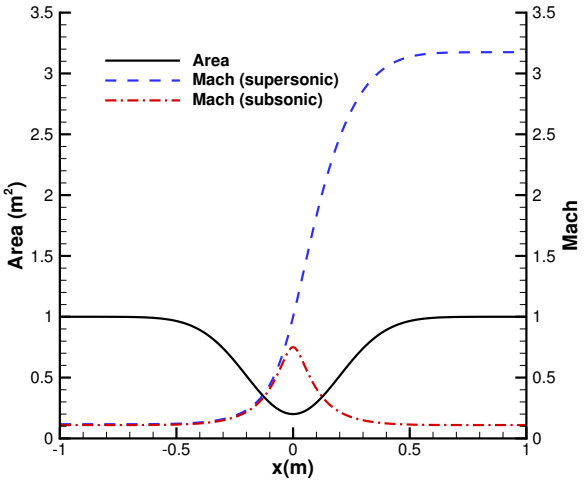


Figure 4.1: Area and Mach number distributions along the nozzle.

4.4.2 Model of an Unstructured Grid

Higher-order residual truncation error estimates are typically very accurate on grids with smoothly varying grid metrics and are able to yield higher-order discretization error estimates via the ETE [3]. However, on grids with non-smoothly varying grid metrics, such as general unstructured grids, the order of accuracy of ETE error estimates degrades to the formal order of the primal problem [2, 1]. To mimic the geometric non-uniformity found in higher dimensional unstructured grids, the spatial domain is partitioned by first generating a uniformly spaced grid and then applying a random perturbation to each grid node. The nodal coordinates of the grids are generated using the following

$$x_p = x_u + [\Delta \times dx_u] \times \text{rand}(-1, +1), \quad \Delta \in [0.0, 0.5) \tag{4.51}$$

where x_p and x_u are the nodes of the perturbed grid and uniform grid, respectively, Δ is the magnitude of the grid perturbation, dx_u is the node spacing on the uniform grid, and $\text{rand}(a, b)$ is a function that returns a uniformly-distributed random number between a and b. The magnitude of the grid perturbation is limited to $\Delta < 0.5$ to ensure positive cell volumes. During preliminary testing, it was found that the conclusions to be drawn do not change significantly with the value of Δ. Therefore, for all results presented in this work, the grid perturbation is set to $\Delta = 0.10$.

4.5 Results

4.5.1 Violation of Assumptions by *k*-exact Reconstruction

Despite reconstructing the solution to a higher-order space, k-exact reconstruction methods fail to yield higher-order ETE error estimates on general unstructured grids. It has been speculated that the loss of higher-order accuracy is a direct result of an insufficiently accurate truncation error estimate. In this section, truncation error estimates obtained via a k-exact reconstruction are examined. Numerical accuracy is assessed using exact truncation error. In particular, the requirements outlined in Section 4.2.3 are inspected to determine if and how a violation may be occurring.

Truncation error estimation is performed on a series of uniform and perturbed grids ranging in size from 32 cells to 2,048 cells. Truncation error estimates are obtained by prolonging the discrete solution to a r^{th} order ($r = 4$) piecewise continuous space and performing a higher-order discrete residual evaluation. All numerical solutions are converged to machine precision, and conservation of the mean is enforced exactly in each control volume. The convergence of error in the truncation error estimate (i.e., $\|\tau_h^p(\tilde{\mathbf{u}}) - \tau_h^p(I_h^r \mathbf{u}_h)\|_2$) is plotted in Figure 4.2. On uniform grids, the truncation error converges to the exact truncation error at a $r+1$ order rate. This superconvergent behavior can likely be attributed to the cancellation of truncation error terms which frequently occurs on uniformly-spaced grids. However, on perturbed grids, the error in the truncation error estimate only converges at a 2^{nd} order rate. Given this behavior, it can be deduced from Eq. 4.21 that ETE error estimates on these perturbed grids will ultimately be limited 2^{nd} order accurate.

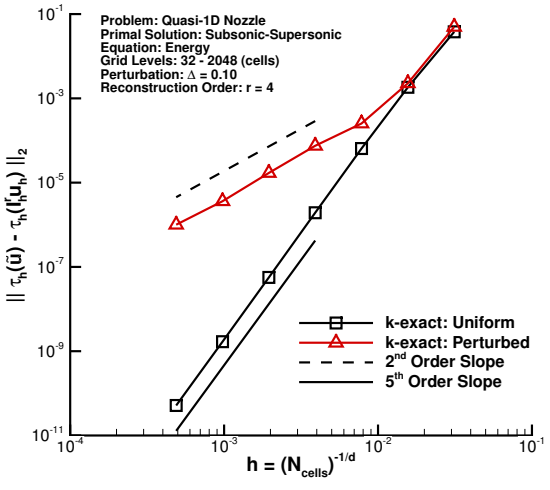


Figure 4.2: Error in truncation error estimate.

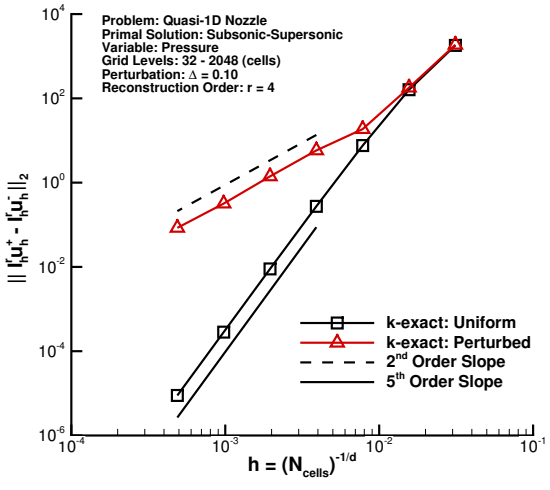
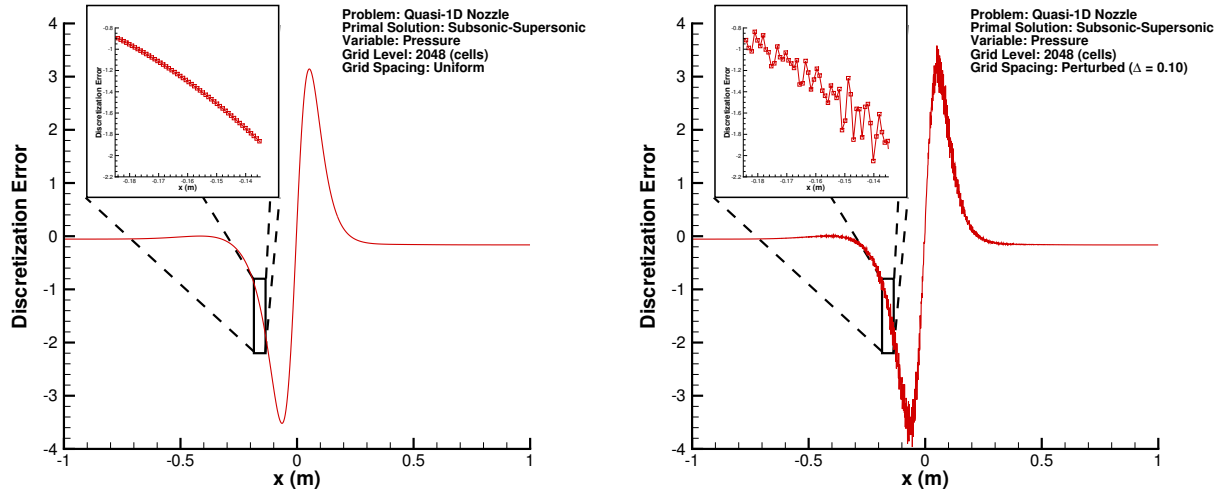


Figure 4.3: Inter-element jumps in the reconstruction.

To determine the underlying cause for this loss of accuracy, we examine the requirements

given in Section 4.2.3. Since conservation of the mean is enforced exactly and all numerical solutions are converged, the reconstructed solution must not be sufficiently representative of the exact continuous solution. When the requirement of inter-element continuity was relaxed and a higher-order discrete residual was substituted for the continuous residual, recall that inter-element jumps in the reconstruction were assumed to remain small, i.e., $O(h^r)$ or better. This assumption is necessary to ensure that the truncation error estimate converges at the proper rate. Norms of inter-element jumps (i.e., $\left\| I_h^r \mathbf{u}_h^+ - I_h^r \mathbf{u}_h^- \right\|_2$) in the reconstruction of static pressure are plotted in Figure 4.3 to examine the smoothness of the reconstructed solution. The convergence behavior of the inter-element jumps is identical to that of the error in the truncation error estimate; uniform grids converge at a rate of $r+1$, and perturbed grids converge at a 2^{nd} order rate. Figure 4.3 suggests that the reconstruction is not a smooth enough representation of the exact continuous solution and that inter-element jumps introduce a 2^{nd} order error which degrades the accuracy of the truncation error estimate.

The accuracy loss observed in Figure 4.2 and Figure 4.3 may be better understood by examining the discretization error. Exact discretization error in static pressure is plotted for a 2,048 cell grid in Figure 4.4. Results are given for both a uniform grid and a perturbed grid. The discretization error is approximately symmetric about the nozzle throat and decays toward zero near the inflow and outflow boundaries. On the uniform grid, discretization error varies smoothly over the entire computational domain. While discretization error on the perturbed grid is qualitatively similar to that of the uniform grid, the discretization error is no longer smooth. The perturbed grid discretization error appears to oscillate about the uniform grid discretization error and indicates the presence of small-scale variations in the discrete solution. These oscillations seem to scale with the magnitude of the average local discretization error. This noisy behavior is typical of the discretization error observed on higher-dimensional unstructured grids [11]. For reconstruction operators which seek to match the data as closely as possible, such as k-exact, these oscillations in the discrete solution are exactly reconstructed. Consequently, a 2^{nd} order error is introduced into the truncation error estimate through excessively large inter-element jumps in the reconstruction. To improve the accuracy of the truncation error estimate, these grid-induced solution oscillations must be explicitly accounted for in the formulation of reconstruction operator.

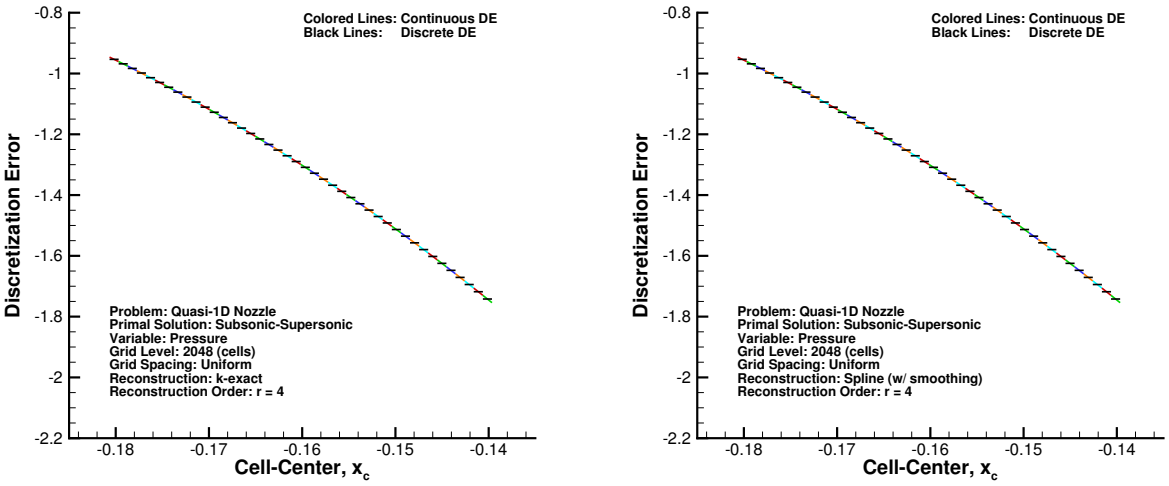


(a) Uniform grid (b) Perturbed grid

Figure 4.4: Local discretization error in pressure for a uniform grid and a perturbed grid.

4.5.2 Reconstruction of Non-smooth Data using Polyharmonic Smoothing Splines

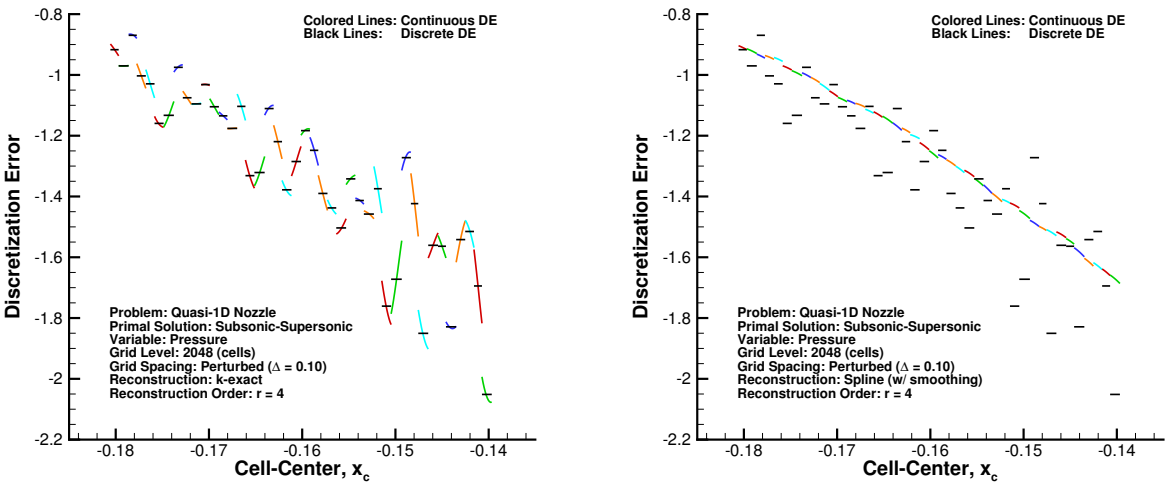
Polyharmonic smoothing splines are employed to minimize inter-element jumps in the reconstruction and recover a smooth approximation of the discrete solution. Reconstructions of the discrete solution are plotted over a portion of the computational domain; the domain of interest corresponds to that of the inset in Figure 4.4. Reconstructions are presented for a 2,048 cell uniform grid and perturbed grid in Figure 4.5 and Figure 4.6, respectively. Since the oscillations in the discrete solution are on the order of discretization error, it would be difficult to differentiate and compare the reconstructions if the reconstructed solution were directly plotted. Instead, exact discretization error is plotted to better illustrate the minute variations in the discrete solution and the reconstruction. The exact solution, which is a smooth, continuous function, is used here simply as a reference. All colored lines correspond to continuous discretization error, i.e., the difference between the exact solution and the reconstructed solution. All black lines correspond to discrete discretization error, i.e., the difference between the discrete solution and the exact solution restricted to the grid. Discrete discretization error, which is a control volume averaged quantity, is depicted by horizontal lines for each control volume.



(a) k-exact (b) Smoothing spline

Figure 4.5: Comparison of reconstructions on a uniform grid.

On the uniform grid, the discrete solution does not exhibit any grid-induced oscillations, and the resulting reconstructions, despite being local to each control volume, qualitatively represent a smooth, continuous function. Furthermore, all inter-element jumps in the reconstruction are small relative to the discretization error. On the perturbed grid, however, the effect of the grid perturbation on the reconstruction is clearly evident. The k-exact reconstruction, which seeks to match the data as closely as possible in a control volume averaged sense, is highly oscillatory from one control volume to the next and does not resemble a smooth, continuous function. Compared to the uniform grid, inter-element jumps in the reconstruction are significantly larger. With the k-exact reconstruction, the small oscillations in the discrete solution are exactly reconstructed. The polyharmonic smoothing spline, on the other hand, provides a much smoother reconstruction which is more representative of a smooth, continuous function. By activating the regularization term, fidelity to the data is sacrificed for smoothness of the fit. This smoothing step is necessary to account for the non-physical variations in the discrete solution. The smoothing spline reconstruction is qualitatively similar to the uniform grid reconstructions and does not exhibit the oscillations observed with k-exact. Moreover, inter-element jumps in the reconstruction are noticeably smaller.

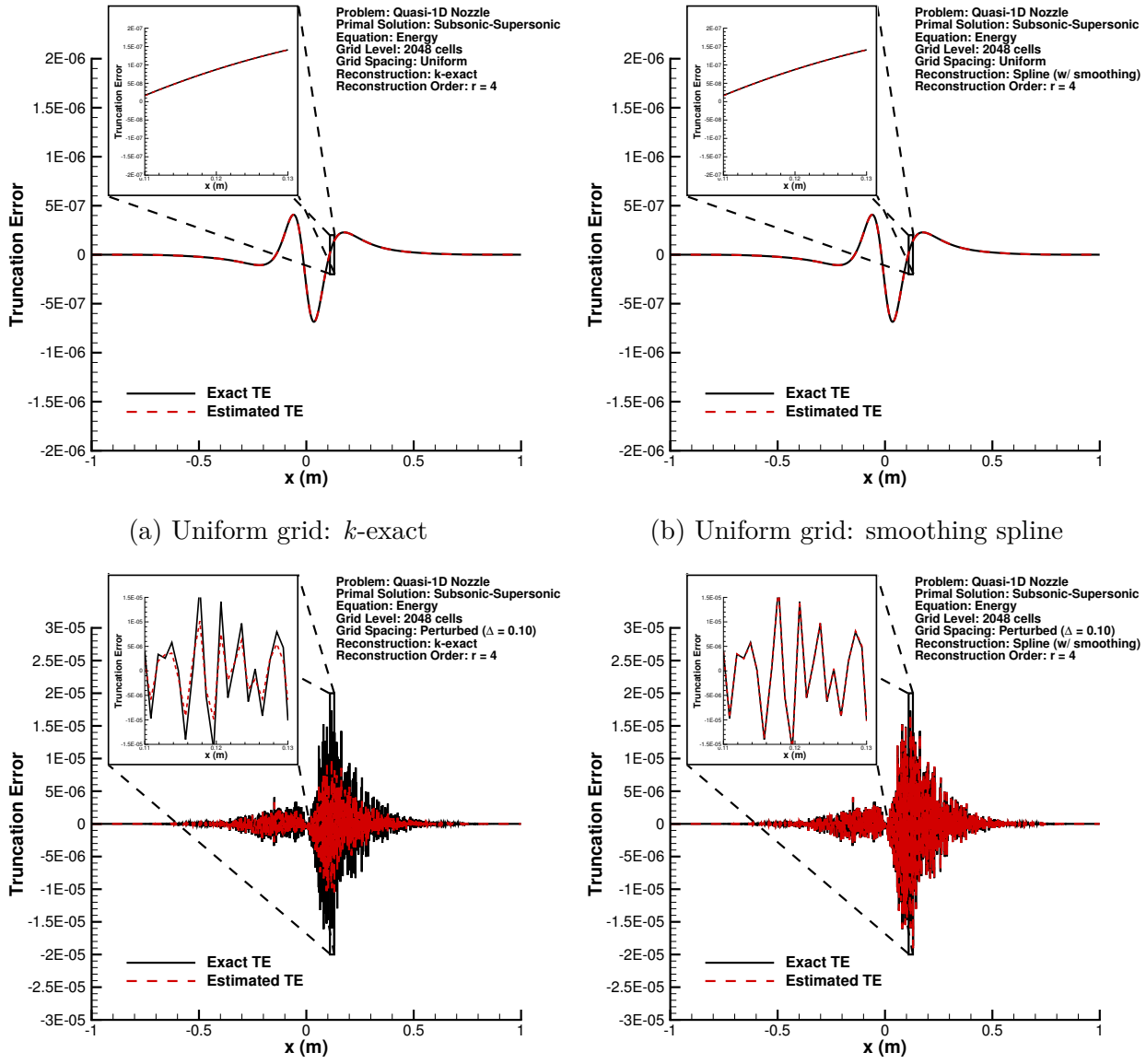


(a) k-exact (b) Smoothing spline

Figure 4.6: Comparison of reconstructions on a perturbed grid.

4.5.3 Error Estimation Results

Truncation error estimation is performed on a 2,048 cell uniform grid and perturbed grid to compare the effectiveness of the k-exact and polyharmonic smoothing spline reconstructions. The discrete solution is reconstructed to a r^{th} order piecewise continuous space ($r = 4$), and higher-order discrete residual truncation error estimates are obtained. Truncation error for the energy equation is plotted in Figure 4.7. Exact truncation error is also plotted to qualitatively verify the accuracy of truncation error estimates. Truncation error on the uniform grid is smooth over the entire computational domain. Both the k-exact reconstruction and the smoothing spline reconstruction yield remarkably accurate truncation error estimates; the truncation error estimates are indistinguishable from the exact truncation error. Truncation error on the perturbed grid is highly oscillatory, non-smooth, and asymptotically larger in magnitude compared truncation error on the uniform grid. This behavior is a direct result of a non-smooth variation in grid metrics. Unlike the uniform grid, it is difficult to identify any underlying smooth function. The k-exact truncation error estimate is qualitatively similar to the exact truncation error; however, it underpredicts the true magnitude. In comparison, the polyharmonic smoothing spline truncation error estimate is significantly more accurate and better captures the magnitude of the exact truncation error. It should be noted that the polyharmonic smoothing spline does not satisfy conservation of the mean when the regularization term is activated. Therefore, to ensure an accurate truncation error estimate, the p^{th} order discrete residual contribution that was neglected in Eq. 4.14, namely $\mathbf{N}_h^p\left(I_r^h I_h^r \mathbf{u}_h\right)$, is explicitly taken into account.



(a) Uniform grid: k-exact

(b) Uniform grid: smoothing spline

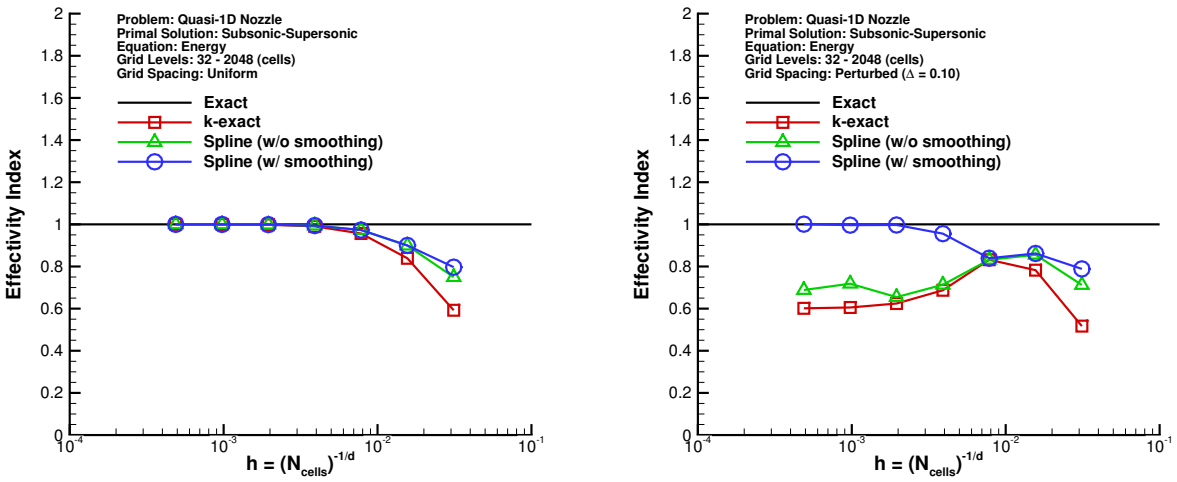
(c) Perturbed grid: k-exact

(d) Perturbed grid: smoothing spline

Figure 4.7: Comparison of truncation error estimates on a uniform grid and a perturbed grid.

Truncation error estimates are further evaluated and compared using an effectivity index. Truncation error effectivity indices are computed for a series of uniform and perturbed grids ranging in size from 32 cells to 2,048 cells. Results are plotted in Figure 4.8. Smoothing spline results are presented with and without the regularization term engaged to illustrate the benefits of added smoothing. On uniform grids, all truncation error effectivity indices approach $\theta = 1.0$ with grid refinement indicating that all truncation error estimates are asymptotically correct. On perturbed grids, k-exact truncation error estimates do not approach $\theta = 1.0$ with grid refinement. Since $\theta < 1.0$ for all grid levels, k-exact truncation error

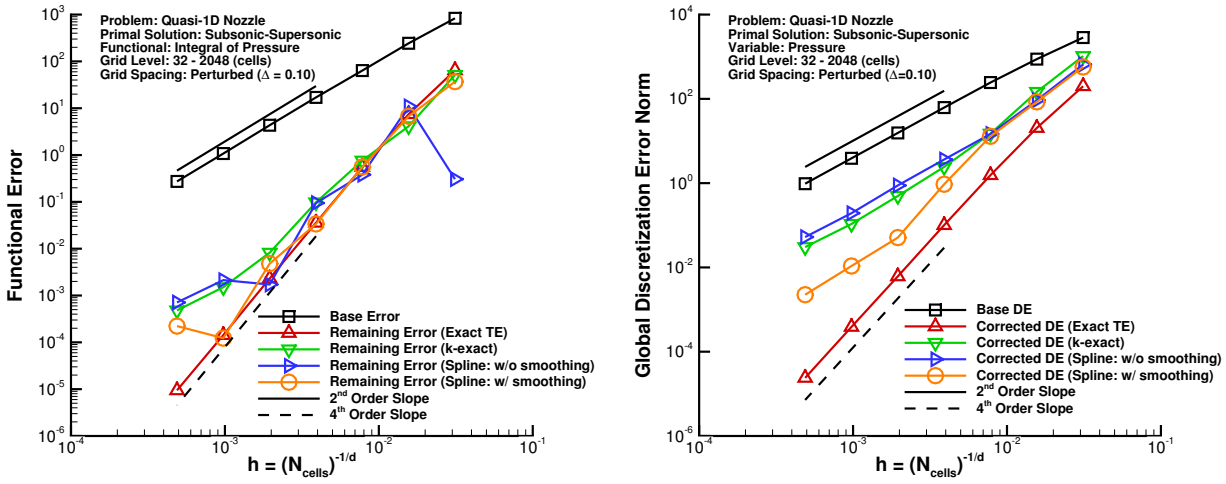
estimates underpredict the magnitude of the exact truncation error; this result is consistent with the behavior observed in Figure 4.7c. Similarly, smoothing spline truncation error estimates are asymptotically incorrect when the regularization term is not active. However, when smoothing is applied, smoothing spline truncation error estimates approach $\theta = 1.0$ with grid refinement. This indicates that smooth reconstructions of the discrete solution are necessary to obtain asymptotically correct truncation error estimates for grids with a non-smooth variation in grid metrics.



(a) Uniform grids (b) Perturbed grids

Figure 4.8: Truncation error effectivity indices on uniform grids and perturbed grids.

Functional error estimates and local discretization error estimates are computed on a series of perturbed grids ranging in size from 32 cells to 2,048 cells. The convergence of error estimates with grid refinement is given in Figure 4.9. The functional of interest is the integral of pressure along the nozzle. Local discretization error is estimated for static pressure. All functional error estimates are computed using the ETE approach to functional error estimation [41]. To assess the accuracy of discretization error estimates resulting from k-exact and smoothing spline truncation error estimates, functional and discretization error estimates are applied as corrections, and the convergence rate of the remaining error is examined. The error estimate for each grid level corresponds to the ordered triplet $(p, q, r) = (2, 2, 4)_L$. Therefore, for a sufficiently accurate truncation error estimate, discretization error estimates should converge at a 4^{th} order rate. For comparison, error estimates are also computed using exact truncation error.



(a) Functional error for the integral of pressure.

(b) Discretization error in pressure.

Figure 4.9: Convergence of error estimates on perturbed grids.

Both the k-exact reconstruction and the smoothing spline reconstruction yield higher-order error estimates over a range of coarse grids, but error estimates ultimately degrade to 2^{nd} order accurate on finer grids. By virtue of how the error is defined, functional error estimates admit error cancellation and, consequently, retain higher-order accuracy longer than the local discretization error estimates. Although higher-order accuracy is lost, the smoothing spline reconstruction does maintain higher-order accuracy for about two grid levels longer than the k-exact reconstruction. Furthermore, on the finest grid levels, the smoothing spline reconstruction yields error estimates which are approximately an order of magnitude more accurate than those of k-exact. If exact truncation error is used, higher-order accuracy can be maintained over the entire range of grids; this behavior indicates that the accuracy of truncation error estimates is indeed the limiting factor in estimating functional errors and discretization errors to higher-order accuracy. Upon closer inspection of truncation error estimates, we found that higher-order accuracy was not retained with the smoothing spline reconstruction because the reconstruction was not sufficiently smooth in just a few control volumes. We suspect that this behavior can occur when the noise level is much smaller in magnitude than the actual solution being reconstructed. As previously mentioned, when noise levels are small, it can be numerically difficult to find the true minimum of robust GCV criterion. Coupled with the ill-conditioning of the reconstruction problem, this can lead to significant under-smoothing or potentially over-smoothing of the data. Further investigation is needed to determine a more robust algorithm for selecting the smoothing parameter in the presence of low noise levels.

4.6 Conclusions

An auxiliary transport equation for the local discretization error in all solution variables was derived. Truncation error was shown to act as a source term driving the generation, transport, and diffusion of discretization error. Truncation error was estimated by reconstructing the discrete solution to a piecewise continuous space and performing a higher-order discrete residual evaluation. Truncation error estimates were inserted into the ETE, and discretization error estimates for output functionals and local solution variables were obtained. For grids with smoothly varying grid metrics, discretization error estimates were shown to converge to the true discretization error at a higher-order rate. However, for grids with non-smoothly varying grid metrics, the order of accuracy of discretization error estimates reduced to that of the primal discretization. On these meshes, truncation error estimates were not accurate enough to yield higher-order accurate discretization error estimates. Non-physical, grid-induced solution variations on the order of discretization error were shown to deteriorate the accuracy of truncation error estimates when reconstruction operators conformed to the data as closely as possible. In particular, lower-order errors were introduced into truncation error estimates through excessive jumps in adjacent reconstructions. To improve the accuracy of truncation error estimates, a new reconstruction method based on polyharmonic smoothing splines was proposed. This reconstruction was designed to balance fidelity to the data with smoothness of the resulting fit. The amount of smoothing to be applied was determined automatically using a variant of the generalized cross-validation criterion. The polyharmonic smoothing spline yielded truncation error estimates which were asymptotically correct in an effectivity index and were considerably more accurate than those of a standard k-exact reconstruction. Furthermore, discretization error estimates were an order of magnitude more accurate on the finest grid levels. The smoothing spline reconstruction outlined in this work provided a significant improvement over existing methods and highlighted the need for reconstruction algorithms which can efficiently approximate a smooth function given non-smooth data.

Acknowledgments

This work was supported by the NASA Graduate Aeronautics Scholarship as part of the Aeronautics Scholarship and Advanced STEM Training and Research Fellowship (AS&ASTAR) Program. We wish to thank Dr. Michael Park and Dr. Joseph Derlaga of NASA Langley Research Center for serving as technical advisors. In addition, we wish to thank Dr. Julianne Chung, Dr. Pang Du, and Dr. Jeffrey Borggaard of Virginia Tech for their helpful discussions.

Bibliography

[1] G. Yan, Numerical estimation of discretization error on unstructured meshes, Ph.D. thesis, University of British Columbia (Mar. 2018).
URL http://hdl.handle.net/2429/64792

[2] M. Sharbatdar, Error estimation and mesh adaptation paradigm for unstructured mesh finite volume methods, Ph.D. thesis, University of British Columbia (Jan. 2017).
URL http://hdl.handle.net/2429/60359

[3] W. C. Tyson, C. J. Roy, A hybrid adjoint/error transport approach to error estimation, adaptation, and higher-order solutions for computational fluid dynamics, in: 55th AIAA Aerospace Sciences Meeting, American Institute of Aeronautics and Astronautics (AIAA), 2017. doi:10.2514/6.2017-0741.
URL https://doi.org/10.2514%2F6.2017-0741

[4] N. A. Pierce, M. B. Giles, Adjoint recovery of superconvergent functionals from PDE approximations, SIAM Review 42 (2) (2000) 247 – 264. doi:10.1137/s0036144598349423.
URL http://dx.doi.org/10.1137/S0036144598349423

[5] D. A. Venditti, D. L. Darmofal, Adjoint error estimation and grid adaptation for functional outputs: Application to quasi-one-dimensional flow, Journal of Computational Physics 164 (1) (2000) 204 – 227. doi:10.1006/jcph.2000.6600.
URL http://dx.doi.org/10.1006/jcph.2000.6600

[6] T. Phillips, Residual-based discretization error estimation for computational fluid dynamics, Ph.D. thesis, Virginia Polytechnic Institute and State University (Sep. 2014).
URL http://hdl.handle.net/10919/50647

[7] G. Yan, C. F. Ollivier-Gooch, Accuracy of discretization error estimation by the error transport equation on unstructured meshes - Nonlinear systems of equations, in: 22nd AIAA Computational Fluid Dynamics Conference, American Institute of Aeronautics and Astronautics (AIAA), 2015. doi:10.2514/6.2015-2747.
URL http://dx.doi.org/10.2514/6.2015-2747

[8] I. B. Celik, D. R. Parsons, Prediction of discretization error using the error transport equation, Journal of Computational Physics 339 (2017) 96 – 125. doi:10.1016/j.jcp.2017.02.058.
URL https://doi.org/10.1016%2Fj.jcp.2017.02.058

[9] C. Roy, Review of discretization error estimators in scientific computing, in: 48th AIAA Aerospace Sciences Meeting Including the New Horizons Forum and Aerospace Exposition, American Institute of Aeronautics and Astronautics (AIAA), 2010. doi:

10.2514/6.2010-126.
URL http://dx.doi.org/10.2514/6.2010-126

[10] W. L. Oberkampf, C. J. Roy, Verification and Validation in Scientific Computing, Cambridge University Press (CUP), 2010. doi:10.1017/cbo9780511760396.
URL http://dx.doi.org/10.1017/CBO9780511760396

[11] M. Sharbatdar, C. F. Ollivier Gooch, Eigenanalysis of truncation and discretization error on unstructured meshes, in: 21st AIAA Computational Fluid Dynamics Conference, 2013, p. 3089. doi:10.2514/6.2013-3089.
URL https://doi.org/10.2514/6.2013-3089

[12] T. Barth, P. Frederickson, Higher order solution of the Euler equations on unstructured grids using quadratic reconstruction, in: 28th Aerospace Sciences Meeting, American Institute of Aeronautics and Astronautics (AIAA), 1990. doi:10.2514/6.1990-13.
URL http://dx.doi.org/10.2514/6.1990-13

[13] M. B. Giles, E. Süli, Adjoint methods for PDEs: a posteriori error analysis and postprocessing by duality, Acta Numerica 11 (2002) 145 – 236. doi:10.1017/S096249290200003X.
URL https://doi.org/10.1017/S096249290200003X

[14] M. B. Giles, N. Pierce, E. Süli, Progress in adjoint error correction for integral functionals, Computing and Visualization in Science 6 (2-3) (2004) 113 – 121. doi:10.1007/s00791-003-0115-y.
URL http://dx.doi.org/10.1007/s00791-003-0115-y

[15] M. Sharbatdar, C. Ollivier-Gooch, Mesh adaptation using c1 interpolation of the solution in an unstructured finite-volume solver, International Journal for Numerical Methods in Fluidsdoi:10.1002/fld.4471.
URL https://doi.org/10.1002/fld.4471

[16] P. Craven, G. Wahba, Smoothing noisy data with spline functions, Numerische Mathematik 31 (4) (1978) 377–403. doi:10.1007/BF01404567.
URL https://doi.org/10.1007/BF01404567

[17] J. Banks, T. Aslam, W. Rider, On sub-linear convergence for linearly degenerate waves in capturing schemes, Journal of Computational Physics 227 (14) (2008) 6985 – 7002. doi:10.1016/j.jcp.2008.04.002.
URL http://dx.doi.org/10.1016/j.jcp.2008.04.002

[18] L. F. Richardson, The approximate arithmetical solution by finite differences of physical problems involving differential equations, with an application to the stresses in a masonry dam, Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences 210 (459-470) (1911) 307 – 357. doi:10.1098/rsta.1911.

0009.
URL http://dx.doi.org/10.1098/rsta.1911.0009

[19] C. Roy, Strategies for driving mesh adaptation in CFD (invited), in: 47th AIAA Aerospace Sciences Meeting including the New Horizons Forum and Aerospace Exposition, American Institute of Aeronautics and Astronautics (AIAA), 2009. doi:10.2514/6.2009-1302.
URL http://dx.doi.org/10.2514/6.2009-1302

[20] F. Fraysse, J. de Vicente, E. Valero, The estimation of truncation error by τ-estimation revisited, Journal of Computational Physics 231 (9) (2012) 3457 – 3482. doi:10.1016/j.jcp.2011.09.031.
URL http://dx.doi.org/10.1016/j.jcp.2011.09.031

[21] C. Ollivier-Gooch, M. Van Altena, A high-order-accurate unstructured mesh finite-volume scheme for the advection-diffusion equation, Journal of Computational Physics 181 (2) (2002) 729 – 752. doi:10.1006/jcph.2002.7159.
URL http://dx.doi.org/10.1006/jcph.2002.7159

[22] M. Sharbatdar, C. F. Ollivier-Gooch, Comparison of truncation error estimators for defect correction and output error estimation in the unstructured mesh finite-volume method, in: 22nd AIAA Computational Fluid Dynamics Conference, American Institute of Aeronautics and Astronautics (AIAA), 2015. doi:10.2514/6.2015-2748.
URL http://dx.doi.org/10.2514/6.2015-2748

[23] J. Banks, J. Hittinger, J. Connors, C. Woodward, Numerical error estimation for nonlinear hyperbolic PDEs via nonlinear error transport, Computer Methods in Applied Mechanics and Engineering 213-216 (2012) 1 – 15. doi:10.1016/j.cma.2011.11.021.
URL http://dx.doi.org/10.1016/j.cma.2011.11.021

[24] G. Yan, C. F. Ollivier-Gooch, Discretization error estimation by the error transport equation on unstructured meshes - Applications to viscous flows, in: 54th AIAA Aerospace Sciences Meeting, American Institute of Aeronautics and Astronautics (AIAA), 2016. doi:10.2514/6.2016-0836.
URL http://dx.doi.org/10.2514/6.2016-0836

[25] T. Barth, Recent developments in high order k-exact reconstruction on unstructured meshes, in: 31st Aerospace Sciences Meeting, American Institute of Aeronautics and Astronautics (AIAA), 1993. doi:10.2514/6.1993-668.
URL http://dx.doi.org/10.2514/6.1993-668

[26] C. Ollivier-Gooch, A. Nejat, K. Michalak, Obtaining and verifying high-order unstructured finite volume solutions to the Euler equations, AIAA Journal 47 (9) (2009) 2105 – 2120. doi:10.2514/1.40585.
URL http://dx.doi.org/10.2514/1.40585

[27] G. H. Golub, C. Van Loan, Matrix Computations, 3rd Edition, Johns Hopkins University Press, 1996.

[28] G. Wahba, Spline Models for Observational Data, Vol. 59, SIAM, 1990. doi:10.1137/1.9781611970128.
URL https://doi.org/10.1137/1.9781611970128

[29] J. C. Carr, R. K. Beatson, B. C. McCallum, W. R. Fright, T. J. McLennan, T. J. Mitchell, Smooth surface reconstruction from noisy range data, in: Proceedings of the 1st International Conference on Computer Graphics and Interactive Techniques in Australasia and South East Asia, ACM, 2003, pp. 119 – ff. doi:10.1145/604471.604495.
URL http://dx.doi.org/10.1145/604471.604495

[30] T. Aboiyar, E. H. Georgoulis, A. Iske, High order weno finite volume schemes using polyharmonic spline reconstruction, in: Proceedings on the International Conference on Numerical Analysis and Approximation Theory, Dept. of Mathematics. University of Leicester, 2006, pp. 1 – 14.
URL http://hdl.handle.net/2381/4274

[31] J. G. Wendelberger, Smoothing noisy data with multidimensional splines and generalized cross-validation, Ph.D. thesis, University of Wisconsin - Madison (Dec. 1983).

[32] J. Duchon, Splines minimizing rotation-invariant semi-norms in sobolev spaces, Constructive Theory of Functions of Several Variables (1977) 85 – 100doi:10.1007/BFb0086566.
URL https://doi.org/10.1007/BFb0086566

[33] G. Wahba, Smoothing noisy data with spline functions, Numerische Mathematik 24 (5) (1975) 383 – 393. doi:10.1007/BF01437407.
URL https://doi.org/10.1007/BF01437407

[34] M. A. Lukas, Robust generalized cross-validation for choosing the regularization parameter, Inverse Problems 22 (5) (2006) 1883. doi:10.1088/0266-5611/22/5/021.
URL http://dx.doi.org/10.1088/0266-5611/22/5/021

[35] M. A. Lukas, Robust gcv choice of the regularization parameter for correlated data, Journal of Integral Equations and Applications (2010) 519 – 547.
URL https://www.jstor.org/stable/26163718

[36] W. Press, S. Teukolsky, W. Vetterling, B. Flannery, Numerical Recipes in FORTRAN 77: The Art of Scientific Computing, Cambridge Univ. Press, 1992.

[37] C. W. Jackson, C. J. Roy, A multi-mesh CFD technique for adaptive mesh solutions, in: 53rd AIAA Aerospace Sciences Meeting, American Institute of Aeronautics and Astronautics (AIAA), 2015. doi:10.2514/6.2015-1958.
URL http://dx.doi.org/10.2514/6.2015-1958

[38] B. van Leer, Towards the ultimate conservative difference scheme. V. a second-order sequel to Godunov's method, Journal of Computational Physics 32 (1) (1979) 101 – 136. doi:10.1016/0021-9991(79)90145-1.
URL http://dx.doi.org/10.1016/0021-9991(79)90145-1

[39] P. Roe, Approximate Riemann solvers, parameter vectors, and difference schemes, Journal of Computational Physics 43 (2) (1981) 357 – 372. doi:10.1016/0021-9991(81)90128-5.
URL http://dx.doi.org/10.1016/0021-9991(81)90128-5

[40] R. M. Beam, R. Warming, An implicit finite-difference algorithm for hyperbolic systems in conservation-law form, Journal of Computational Physics 22 (1) (1976) 87 – 110. doi:10.1016/0021-9991(76)90110-8.
URL http://dx.doi.org/10.1016/0021-9991(76)90110-8

[41] W. C. Tyson, K. Swirydowicz, J. M. Derlaga, C. J. Roy, E. de Sturler, Improved functional-based error estimation and adaptation without adjoints, in: 46th AIAA Fluid Dynamics Conference, American Institute of Aeronautics and Astronautics (AIAA), 2016. doi:10.2514/6.2016-3809.
URL http://dx.doi.org/10.2514/6.2016-3809

Chapter 5

Relinearization of the Error Transport Equations for Arbitrarily High-Order Error Estimates

William C. Tyson,[a] Gary K. Yan,[b] Christopher J. Roy,[a] Carl F. Ollivier-Gooch[b]

[a] Kevin T. Crofton Department of Aerospace and Ocean Engineering, Virginia Tech
215 Randolph Hall, 460 Old Turner Street, Blacksburg, VA, 24061

[b] Department of Mechanical Engineering, The University of British Columbia
2054-6250 Applied Science Lane, Vancouver, B.C., V6T 1Z4, Canada

Keywords: Error transport equations, Computational fluid dynamics, Finite-volume method, Discretization error estimation, Higher-order error estimates

Attribution

- William C. Tyson (First Author): The first author served as the main contributor and primary author of the work. The first author implemented ETE relinearization within the BlueRidge code base and performed numerical simulations for 2D viscous Burgers' equation, the quasi-1D nozzle problem, and the 2D Euler equations.
- Gary K. Yan (Second Author): The second author provided results for the 2D RANS equations. The second author also provided helpful comments on the manuscript.
- Christopher J. Roy (Third Author): The third author provided valuable guidance and insight during the course of the study. The third author also provided helpful comments

on the manuscript.

- Carl F. Ollivier-Gooch (Fourth Author): The fourth author provided valuable guidance and insight during the course of the study. The fourth author also provided helpful comments on the manuscript.

Abstract

A higher-order accurate discretization error estimation procedure for finite-volume schemes is presented. Discretization error estimates are computed using the linearized error transport equations (ETE). ETE error estimates are applied as a correction to the primal solution. The ETE are then relinearized about the corrected primal solution, and discretization error estimates are recomputed. This process, referred to as ETE relinearization, is performed in an iterative manner to successively increase the accuracy of discretization error estimates. Under certain conditions, ETE relinearization is shown to correct error estimates, or equivalently the entire primal solution, to higher-order accuracy. In terms of computational cost, ETE relinearization has a competitive advantage over conventional higher-order discretizations when used as a form of defect correction for the primal solution. Furthermore, ETE relinearization is shown to be particularly useful for problems where the error incurred by the linearization of the ETE cannot be neglected. Results are presented for several inviscid and viscous flow problems using both structured and unstructured meshes.

5.1 Introduction

Computational fluid dynamics (CFD) simulations can provide tremendous insight into complex physical phenomena and are a powerful tool for both engineering design and analysis. For inexperienced users, it can be tempting to view CFD simulations as fully representative of the true physics observed in nature. Discrepancies, however, almost always exist between CFD results and reality due to errors inherent in each simulation. These simulation errors can have a significant impact on the accuracy of a given CFD result and, if not properly quantified, can adversely affect engineering decision-making leading to poor system performance, system failure, or risk to public safety. Simulation errors can generally be categorized as modeling errors or numerical errors; modeling errors arise from an incomplete description of the true physics while numerical errors arise when the governing equations are solved in an approximate sense on a computer. The focus of this work is on the estimation of numerical error, and in particular, discretization error. Discretization error is defined as the difference between the exact continuous solution to the governing equations and the corresponding discrete solution. Assuming that numerical solutions are iteratively converged and that round-off errors are negligible, discretization error is the primary contributor to

the total numerical error in a given simulation. These errors can be extremely difficult to estimate as they are a highly nonlinear function of the computational grid, the solution, and the discretization scheme.

Discretization error estimators are commonly classified as multiple-grid or single-grid error estimators; terminology which simply refers to the number of computational grids and flow solutions used to derive the error estimate. Richardson extrapolation [1], which is arguably the first multiple-grid error estimator, utilizes the discrete solutions from a series of systematically-refined grids to extrapolate an estimate of the exact solution. Subsequently, an error estimate is formulated from the difference between the extrapolated solution and the discrete solution on the finest grid level. Richardson extrapolation is non-intrusive to existing code bases and can be easily applied as a post-processing step. However, in order to obtain a reliable error estimate, all discrete solutions must be in the asymptotic range. For large engineering applications, this requirement can be difficult to satisfy as the spatial and temporal resolution needed to attain asymptotic range for all grid levels can become computationally expensive and impractical.

Single-grid error estimators, on the other hand, are often computationally cheaper than multiple-grid error estimators, albeit more difficult to implement. Gradient-based error estimators, which are frequently used in the context of grid adaptation [2, 3], formulate a single-grid error estimate from gradients of the solution variables. These error estimators, however, incorrectly assume that discretization errors accumulate in regions with large solution gradients (e.g., an exactly resolved shear layer). Furthermore, gradient-based error estimators do not account for the physical mechanisms which govern the generation, transport, and diffusion of discretization error throughout the computational domain [4, 5, 6]. To properly model the behavior of discretization error, single-grid error estimators based on the solution of an auxiliary transport equation were developed [7, 8, 9]. These error estimators, known as error transport equations (ETE), can provide very accurate discretization error estimates, are computationally efficient, and are versatile in the sense that they are agnostic to the discretization scheme and can be applied to nearly any PDE or ODE for both steady and unsteady problems [10, 11, 12].

Often in engineering applications, field data is synthesized into output functionals of interest to facilitate design choices and assess system performance. Output functionals can be integrated quantities of the solution (e.g., lift or drag for an aerodynamics application) or can be scalar quantities at a given spatial location. For directly estimating discretization errors in output functionals, adjoint methods can be employed [13, 14, 15, 16]. Adjoint methods can provide error estimates for any output functional of interest, and if applied as a correction, adjoint-based error estimates can increase the observed order of accuracy of their corresponding functionals [17]. In a linearized sense, an equivalence between adjoint methods and ETE methods can be demonstrated for functional error estimation [18]. Moreover, the adjoint solution can be formulated into an adaptation indicator [19, 20]. Adjoint-based adaptation indicators are more mathematically rigorous than heuristic gradient-based indicators and can target regions of the domain which contribute most to the error in a single functional or

multiple functionals [21].

More recently, research efforts have focused on utilizing the ETE to obtain higher-order estimates of discretization error (i.e., error estimates which converge to the true error at a higher rate than that of the primal discretization) [12, 22, 23]. Higher-order error estimates can provide asymptotically tighter error bounds on the solution than conventional lower-order error estimates, albeit stricter conditions must be satisfied to ensure higher-order accuracy. If obtaining an error estimate in the lower-order solution is not of primary concern, higher-order error estimates may be used to correct the entire primal solution in a similar manner to defect correction [24, 25, 26]. For certain classes of problems, it can be computationally advantageous to use a linearized form of the ETE for higher-order error estimation rather than the full nonlinear ETE [27, 28]. When coupled with a correction step, the linearized ETE provides a more efficient and arguably more stable means of obtaining higher-order solutions than a standard higher-order discretization.

In this work, we explore the possibility of iteratively applying linearized ETE error estimates as corrections to the solution, a method henceforth referred to as ETE relinearization. The primary goals of this approach are to successively improve the order of accuracy of discretization error estimates, or equivalently the primal solution, and ultimately provide a more reliable quantification of numerical uncertainty. ETE error estimates are computed on structured and unstructured grids. p-Truncation error estimates [29, 30] are inserted into the ETE as a source term to drive the error estimation procedure. Furthermore, we examine the effectiveness of ETE relinearization for problems where the error incurred by the linearization of the ETE cannot be neglected. The efficiency of ETE relinearization is compared to that of a conventional higher-order solver. Accuracy and robustness considerations are also discussed. Results are presented for several inviscid and viscous flow problems.

5.2 Background

5.2.1 Finite-Volume Discretization

Consider a general conservation law of the form

$$\frac{\partial \tilde{\mathbf{u}}}{\partial t} + \nabla \cdot \mathbb{F}(\tilde{\mathbf{u}}) = \mathbf{s}(\tilde{\mathbf{u}}) \tag{5.1}$$

defined over the domain $\Omega \subset \mathbb{R}^m \times [0, \infty), \mathbf{x} \in \mathbb{R}^m, m \in \{1, 2, 3\}$, where $\tilde{\mathbf{u}}$ is a vector of conserved variables, $\mathbb{F}(\cdot)$ is a flux tensor, and $\mathbf{s}(\cdot)$ is a source term. The domain of interest is discretized into a set of non-overlapping elements, $\mathcal{T}_h$, such that

$$\Omega = \bigcup_{i=1}^{N_e} \Omega_i, \quad \Omega_i \in \mathcal{T}_h \tag{5.2}$$

where h is a characteristic mesh size. Eq. 5.1 is cast into a weak form by integrating over each control volume, Ω_i. Consequently, the continuous primal operator, $\mathbf{N}(\cdot)$, may be defined as

$$\mathbf{N}(\tilde{\mathbf{u}}) \coloneqq \frac{\partial}{\partial t}\int_{\Omega_i} \tilde{\mathbf{u}}\ d\Omega + \mathcal{R}(\tilde{\mathbf{u}}) = \mathbf{0} \tag{5.3}$$

where $\mathcal{R}(\cdot)$ is the steady-state, continuous residual given by

$$\mathcal{R}(\tilde{\mathbf{u}}) = \oint_{\partial\Omega_i} \mathbb{F}(\tilde{\mathbf{u}}) \cdot \hat{\mathbf{n}}\ dS - \int_{\Omega_i} \mathbf{s}(\tilde{\mathbf{u}})\ d\Omega\ . \tag{5.4}$$

In Eq. 5.4, the flux divergence integral has been converted to a surface integral using the Gauss divergence theorem. Furthermore, it has been assumed that all control volumes are fixed in space so that, by the Reynolds transport theorem, the time derivative can be pulled out of the integral in Eq. 5.3.

In accordance with the finite-volume method, a discrete solution, $\mathbf{u}_h$, is sought which approximates the control volume average of $\tilde{\mathbf{u}}$. The analogous discrete operator to Eq. 5.3 is defined as

$$\mathbf{N}_h^p(\mathbf{u}_h) \coloneqq |\Omega_i|\frac{d\mathbf{u}_h}{dt} + \mathcal{R}_h^p(\mathbf{u}_h) = \mathbf{0} \tag{5.5}$$

where $|\Omega_i|$ is the volume of Ω_i, p is the formal order of accuracy of the discretization, and $\mathcal{R}_h^p(\cdot)$ is the steady-state, discrete residual given by

$$\mathcal{R}_h^p(\mathbf{u}_h) = \oint_{\partial\Omega_i} \mathbb{F}_h\Big(I_h^p\mathbf{u}_h^+, I_h^p\mathbf{u}_h^-\Big) \cdot \hat{\mathbf{n}}\ dS - \int_{\Omega_i} \mathbf{s}(I_h^p\mathbf{u}_h)\ d\Omega\ . \tag{5.6}$$

The operator I_a^b is a restriction/prolongation operator used to project the solution from one space to another; the subscript a is the starting space and the superscript b is the resultant space. An empty subscript or superscript is used to denote a C^∞ continuous space. Likewise, a subscript or superscript h is used to denote a discrete space on $\mathcal{T}_h$. Higher-order spatial accuracy is achieved by prolonging the discrete solution to a p^{th} order piecewise continuous space. The notation $(\cdot)^{+/-}$ is used to denote the trace of the reconstructed solution from the exterior and interior of Ω_i, respectively. The flux is approximated with a discrete flux function, $\mathbb{F}_h(\cdot,\cdot)$, to resolve the discontinuous states at control volume interfaces. Eq. 5.5 may be written for each control volume in $\mathcal{T}_h$ and may be marched in time using any ODE time integrator. For all discrete solutions in this work, time integration is performed using the backward Euler time integrator.

5.2.2 Discretization Error

By solving the discrete problem, Eq. 5.5, rather than the continuous problem, Eq. 5.3, a numerical error known as discretization error is incurred. Discretization error is defined as the difference between the exact discrete solution and the exact continuous solution, i.e.

$$\varepsilon_h = \mathbf{u}_h - I^h\tilde{\mathbf{u}}\ . \tag{5.7}$$

The operator, I^h, is used to restrict the exact continuous solution to $\mathcal{T}_h$ and, in the context of finite-volume methods, corresponds to the cell averaging operator,

$$I^h(\cdot) \coloneqq \frac{1}{|\Omega_i|} \int_{\Omega_i} (\cdot)\ d\Omega\ . \tag{5.8}$$

Eq. 5.7 represents the discrete form of discretization error. Assuming the reconstruction operator preserves the control volume average, an analogous continuous form of discretization error may be written as

$$\varepsilon = \mathbf{u} - \tilde{\mathbf{u}} \tag{5.9}$$

where $\mathbf{u}$ is a continuous approximation of $\mathbf{u}_h$. Since the exact continuous solution is rarely known, discretization error cannot be directly evaluated and instead must be estimated.

5.3 Methodology

5.3.1 Error Transport Equations (ETE)

Local discretization error estimates are obtained by deriving and solving auxiliary error transport equations (ETE) [23]. The ETE are derived by inserting Eq. 5.9 into Eq. 5.3 and subtracting $\mathcal{R}(\mathbf{u})$ from both sides. In this manner, a continuous ETE operator, $\hat{\mathbf{N}}(\mathbf{u}, \varepsilon)$, may be defined as

$$\hat{\mathbf{N}}(\mathbf{u}, \varepsilon) = -\frac{\partial}{\partial t} \int_{\Omega_i} \varepsilon\ d\Omega + \mathcal{R}(\mathbf{u} - \varepsilon) - \mathcal{R}(\mathbf{u}) = -\left[\frac{\partial}{\partial t} \int_{\Omega_i} \mathbf{u}\ d\Omega + \mathcal{R}(\mathbf{u}) \right]\ . \tag{5.10}$$

The continuous residual, $\mathcal{R}(\mathbf{u})$, is subtracted from both sides of Eq. 5.10 so that the right-hand side becomes the standard residual for the primal problem. Furthermore, when $\mathcal{R}(\cdot)$ is restricted to a linear operator, Eq. 5.10 simplifies to

$$\hat{\mathbf{L}}(\mathbf{u}, \varepsilon) = \frac{\partial}{\partial t} \int_{\Omega_i} \varepsilon\ d\Omega + \mathcal{R}(\varepsilon) = \frac{\partial}{\partial t} \int_{\Omega_i} \mathbf{u}\ d\Omega + \mathcal{R}(\mathbf{u})\ . \tag{5.11}$$

Eq. 5.11 demonstrates a common finding in numerical analysis in that, for linear operators, the discretization error is governed by the same equation as the numerical solution with the addition of the residual as a source term. Now, in a similar manner to the primal problem, Eq. 5.10 may be discretized as follows

$$\hat{\mathbf{N}}_h(\mathbf{u}_h, \bar{\varepsilon}_h) = -|\Omega_i| \frac{d\bar{\varepsilon}_h}{dt} + \mathcal{R}_h^q(\mathbf{u}_h - \bar{\varepsilon}_h) - \mathcal{R}_h^q(\mathbf{u}_h) = -\left[|\Omega_i| \frac{d\mathbf{u}_h}{dt} + \mathcal{R}_h^r(\mathbf{u}_h) \right], \quad \forall\ r > p \tag{5.12}$$

where $\bar{\varepsilon}_h$ is an estimate of ε_h. The left-hand side is discretized to order q and the right-hand side to order r. The condition $r > p$ is enforced to prevent the discrete residual on the right-hand side from vanishing and to ensure an asymptotically correct error estimate [7, 22, 23].

Eq. 5.10 and Eq. 5.12 represent the full nonlinear form of the ETE. To reduce computational costs, it is common to solve a simplified form of the nonlinear ETE. In this work, we consider a Newton linearization [7, 9, 11, 22]. First, the nonlinear discrete residual, $\mathcal{R}_h^q(\mathbf{u}_h - \varepsilon)$, is expanded in a Taylor series expansion as follows

$$\mathcal{R}_h^q(\mathbf{u}_h - \bar{\varepsilon}_h) = \mathcal{R}_h^q(\mathbf{u}_h) - \frac{\partial \mathcal{R}_h^q}{\partial \mathbf{u}_h}\bar{\varepsilon}_h + O\Big(\|\bar{\varepsilon}_h\|^2\Big) \tag{5.13}$$

where $\frac{\partial \mathcal{R}_h^q}{\partial \mathbf{u}_h}$ is the residual Jacobian matrix. It is assumed that $\|\varepsilon\| \ll \|\tilde{\mathbf{u}}\|$ so that all $O\Big(\|\bar{\varepsilon}_h\|^2\Big)$ terms may be neglected. Finally, by inserting Eq. 5.13 into Eq. 5.12, the linearized form of the ETE may be written as

$$\hat{\mathbf{L}}_h(\mathbf{u}_h, \bar{\varepsilon}_h) = |\Omega_i|\frac{d\bar{\varepsilon}_h}{dt} + \frac{\partial \mathcal{R}_h^q}{\partial \mathbf{u}_h}\bar{\varepsilon}_h = |\Omega_i|\frac{d\mathbf{u}_h}{dt} + \mathcal{R}_h^r(\mathbf{u}_h) \ . \tag{5.14}$$

While slightly less accurate, the linearized ETE can be computationally cheaper than the full nonlinear ETE. For steady-state problems, the numerical solution of a nonlinear PDE is replaced by the numerical solution of one linear system.

In this work, error estimation results for the nonlinear ETE are referred to by the ordered triplet (p, q, r) where p is the order of the primal problem, q is the order of the ETE, and r is the order of the residual approximation. Similarly, error estimation results for the linearized ETE are referred to by the ordered triplet $(p, q, r)_L$. Numerical solutions of the primal problem are referred to by the ordered triplet $(p, 0, 0)$.

5.3.1.1 Expected Order of Accuracy for Discretization Error Estimate

For a given ordered triplet (p, q, r), the expected order of accuracy for the nonlinear ETE error estimate is given by the following [23]

$$\|\bar{\varepsilon}_h - \varepsilon_h\| = O\Big(h^{\min(p+q,r)}\Big) \ . \tag{5.15}$$

Based on Eq. 5.15, higher-order error estimates, that is error estimates which converge to the true error faster than the formal order of the primal problem, may be obtained for any combination of $q \geq 1$ and $r > p$. For the linearized ETE, an additional $O(h^{2p})$ error is incurred during the linearization process. Consequently, the expected order of accuracy for linearized ETE error estimates is given by [23]

$$\|\bar{\varepsilon}_h - \varepsilon_h\| = O\Big(h^{\min(2p,p+q,r)}\Big) \ . \tag{5.16}$$

Eq. 5.15 and Eq. 5.16 are valid for smooth solutions and serve as limiting cases in the presence of flowfield and geometric discontinuities. Furthermore, Eq. 5.15 and Eq. 5.16 hold for discretizations on uniform meshes and meshes with smoothly varying grid metrics. For discretizations on general unstructured meshes and non-smooth structured meshes, the

expected orders of accuracy for the nonlinear ETE error estimate and the linearized ETE error estimate are given by [28]

$$\|\bar{\varepsilon}_h - \varepsilon_h\| = O\Big(h^{p+(q-p)\delta_{qr}}\Big) \tag{5.17}$$

and

$$\|\bar{\varepsilon}_h - \varepsilon_h\| = O\Big(h^{\min(2p,p+(q-p)\delta_{qr})}\Big) \ , \tag{5.18}$$

respectively, where δ_{qr} is the Kronecker delta. Based on Eq. 5.17 and Eq. 5.18, higher-order error estimates may be obtained for any combination of $q > p$ and $r = q$.

5.3.1.2 Equivalence of Discrete Residual Approximation and Truncation Error Estimate

It is possible to show that a certain equivalence exists between the discrete residual approximation in Eq. 5.12 and an estimate of truncation error. Consider the following Generalized Truncation Error Expression (GTEE) [31, 32]

$$\mathbf{N}_h^p\Big(I^h\mathbf{u}\Big) = \mathbf{N}(\mathbf{u}) + \tau_h^p(\mathbf{u}) \tag{5.19}$$

where $\mathbf{u}$ is an arbitrary continuous function and $\tau_h^p(\cdot)$ is the truncation error. Eq. 5.19 demonstrates that truncation error is simply the functional difference between the continuous and discrete governing equations and can be viewed as higher-order terms which are neglected during the process of discretization. For certain discretizations on simple geometries, the exact form of the truncation error may be written analytically. However, for finite-volume schemes, especially on complex geometries, it is very difficult to analytically determine the exact form of the truncation error. Instead, truncation error is typically estimated. Using Eq. 5.19, a truncation error estimate may be formulated as follows

$$\begin{aligned} \tau_h^p(I_h^r\mathbf{u}_h) &= \mathbf{N}_h^p(I_r^h I_h^r\mathbf{u}_h) - \mathbf{N}(I_h^r\mathbf{u}_h) \\ &= \cancelto{\mathbf{0}}{\mathbf{N}_h^p(\mathbf{u}_h)} - \mathbf{N}(I_h^r\mathbf{u}_h) \\ &= -\mathbf{N}(I_h^r\mathbf{u}_h) \ . \end{aligned} \tag{5.20}$$

The truncation error estimate is simply the residual from operating the continuous equations onto a reconstruction of the discrete solution. Given Eq. 5.20, the following assumptions regarding the truncation error estimate are required:

1. The reconstruction operator conserves the mean.: $I_r^h I_h^r\mathbf{u}_h = \mathbf{u}_h$
2. The discrete solution is converged to machine precision.: $\mathbf{N}_h^p(\mathbf{u}_h) = \mathbf{0}$
3. The reconstructed solution satisfies the continuity and boundary condition requirements imposed by the continuous operator.

In the context of finite-volume schemes, most reconstruction operators enforce conservation of the mean by default. Furthermore, nonlinear convergence can be achieved with a robust solver and sufficient computational resources. However, satisfying continuity and boundary condition requirements in a global sense can be challenging. In order to relax these continuity and boundary condition requirements, the truncation error can instead be approximated by a higher-order discrete residual evaluation [30]:

$$\tau_h^p(I_h^r \mathbf{u}_h) \approx -\mathbf{N}_h^r(\mathbf{u}_h), \quad \forall\ r > p\ . \tag{5.21}$$

Upon examination, one can see that Eq. 5.21 and the right-hand side of Eq. 5.12 are equivalent.

5.3.1.3 Equivalence of Linearized ETE and Discrete Adjoint

In previous work [18], an equivalence between the linearized ETE and discrete adjoint methods for functional error estimation was demonstrated. This equivalence is reiterated here for completeness. For a given output functional, $J_h(\cdot)$, an error estimate, ε_{J_h}, may be computed as follows

$$\varepsilon_{J_h} \approx -\overbrace{\frac{\partial J_h}{\partial \mathbf{u}_h}\underbrace{\left[\frac{\partial \mathbf{N}_h^p}{\partial \mathbf{u}_h}\right]^{-1}\tau_h^p(\tilde{\mathbf{u}})}_{\text{Linearized ETE } = -\bar{\varepsilon}_h}}^{\text{Discrete Adjoint Problem } = -\lambda_h^T}\ . \tag{5.22}$$

Eq. 5.22 illustrates that the functional error estimate may be computed as either the inner product of the discrete adjoint solution, λ_h, with the truncation error or as the inner product of the functional Jacobian, $\frac{\partial J_h}{\partial \mathbf{u}_h}$, with the linearized ETE solution. This equivalence holds for the triplet $(p, p, r)_L$. For a discussion of the advantages/disadvantages of the linearized ETE approach to functional error estimation, refer to Ref. [18].

5.3.2 *k*-Exact Least-Squares Reconstruction

A k-exact least-squares reconstruction operator is used here in the numerical solution of the ETE [33, 34]. The reconstruction is referred to as "k-exact" since it can exactly reconstruct polynomials of degree k. For brevity, the k-exact reconstruction operator is outlined here for a 1D scalar variable. Consider the following Taylor series expansion,

$$u_i^R(x - x_i) = u|_i + \left.\frac{\partial u}{\partial x}\right|_i (x - x_i) + \ldots + \left.\frac{\partial^k u}{\partial x^k}\right|_i \frac{(x - x_i)^k}{k!} \tag{5.23}$$

where $u_i^R(\cdot)$ is the reconstructed solution in Ω_i, $\left\{u|_i, \left.\frac{\partial u}{\partial x}\right|_i, \ldots, \left.\frac{1}{k!}\frac{\partial^k u}{\partial x^k}\right|_i\right\}$ are coefficients of the reconstruction, and x_i is the centroid of Ω_i. The reconstruction coefficients are uniquely

determined by requiring conservation of the mean in Ω_i as well as in a stencil of neighboring control volumes Ω_j, namely

$$\begin{aligned}
\frac{1}{|\Omega_i|}\int_{\Omega_i} u_i^R(x-x_i)\ d\Omega = u|_i + \left.\frac{\partial u}{\partial x}\right|_i \overline{x}_i + \ldots + \left.\frac{\partial^k u}{\partial x^k}\right|_i \frac{\overline{x^k}_i}{k!} &= u_{h,i} \\
\frac{1}{|\Omega_j|}\int_{\Omega_j} u_i^R(x-x_i)\ d\Omega = u|_i + \left.\frac{\partial u}{\partial x}\right|_i \hat{x}_{ij} + \ldots + \left.\frac{\partial^k u}{\partial x^k}\right|_i \frac{\widehat{x^k}_{ij}}{k!} &= u_{h,j}\ .
\end{aligned} \tag{5.24}$$

The terms $\overline{x^n}_i$ and $\widehat{x^n}_{ij}$ are control volume moments which are defined as follows

$$\begin{aligned}
\overline{x^n}_i &\coloneqq \frac{1}{|\Omega_i|}\int_{\Omega_i}(x-x_i)^n\ d\Omega \\
\widehat{x^n}_{ij} &\coloneqq \sum_{m=0}^{n}\binom{n}{m}(x_j-x_i)^m\overline{x^{n-m}}_j\ .
\end{aligned} \tag{5.25}$$

By writing Eq. 5.24 for Ω_i and for each Ω_j, the following linear system may be formulated for the reconstruction coefficients

$$\begin{bmatrix} w_{i1}(\hat{x}_{i1}-\overline{x}_i) & \ldots & w_{i1}\left(\widehat{x^k}_{i1}-\overline{x^k}_i\right) \\ \vdots & \ddots & \vdots \\ w_{iN}(\hat{x}_{iN}-\overline{x}_i) & \ldots & w_{i1}\left(\widehat{x^k}_{iN}-\overline{x^k}_i\right) \end{bmatrix} \begin{pmatrix} \frac{\partial u}{\partial x} \\ \vdots \\ \frac{1}{k!}\frac{\partial^k u}{\partial x^k} \end{pmatrix}_i = \begin{pmatrix} w_{i1}(u_{h,1}-u_{h,i}) \\ \vdots \\ w_{iN}(u_{h,N}-u_{h,i}) \end{pmatrix} \tag{5.26}$$

where N is the number of control volumes in the stencil and $w_{ij} \coloneqq \frac{1}{\|x_i-x_j\|^\gamma}, \gamma \geq 0$ is a weight used to emphasize geometrically close data. The row corresponding to Ω_i has been analytically eliminated to enforce conservation of the mean in Ω_i exactly. Eq. 5.26 is solved using a truncated singular-value decomposition (TSVD) [35].

5.3.3 Relinearization of the ETE

Higher-order discretization error estimates are computed using a process referred to as ETE relinearization. First, the lower-order primal problem is solved to convergence. A higher-order discrete residual truncation error estimate is computed and inserted as the source term for the linearized ETE. The resulting linearized ETE error estimate is applied as a correction to the discrete solution. The procedure continues for a prescribed number of relinearization steps by recomputing a truncation error estimate, relinearizing the ETE, and reapplying the error estimate as a correction. At the conclusion of the relinearization loop, a final discretization error estimate is obtained by differencing the lower-order primal solution with the corrected solution. Functional error estimates can be computed with an additional post-processing step using Eq. 5.22. The complete ETE relinearization procedure is outlined in Algorithm 1.

Algorithm 1 ETE Relinearization

Compute: $\mathbf{N}_h^p(\mathbf{u}_{h,0}) = \mathbf{0}$ ▷ Solve Primal
for $i = 1$ to N_{relin} **do**
  Compute: $\mathcal{R}_h^r(\mathbf{u}_{h,i-1})$ ▷ Estimate Residual/Truncation Error
  Compute: $\frac{\partial \mathcal{R}_h^q}{\partial \mathbf{u}_{h,i-1}} \bar{\varepsilon}_{h,i} = \mathcal{R}_h^r(\mathbf{u}_{h,i-1})$ ▷ Solve Linearized ETE
  Compute: $\mathbf{u}_{h,i} = \mathbf{u}_{h,i-1} - \bar{\varepsilon}_{h,i}$ ▷ Update Discrete Solution
end for
Compute: $\bar{\varepsilon}_h = \mathbf{u}_{h,0} - \mathbf{u}_{h,N_{relin}}$ ▷ Define Error Estimate
for $j = 1$ to N_{func} **do**
  Compute: $\varepsilon_{J_{h,j}} = \frac{\partial J_{h,j}}{\partial \mathbf{u}_h} \bar{\varepsilon}_h$ ▷ Evaluate Functional Errors
end for

Assuming that the primal solution is asymptotic and that truncation error estimates converge at the expected rate, discretization error estimates of an arbitrary order of accuracy can be obtained. According to Eq. 5.16, each relinearization step can theoretically correct away lower-order errors in the primal solution, and consequently, increase the order of accuracy of the discretization error estimate. In practice, several correction steps may be needed to fully correct out a given lower-order error. Additionally, ETE relinearization can be used to improve error estimates for problems where the errors incurred by the linearization of the ETE may be large; for instance, a problem with an under-resolved flow feature. In an equivalent manner to discretization error estimates, the order of accuracy for the entire primal solution can be increased if ETE relinearization is used to directly correct the lower-order primal solution. The computational advantages of this approach over a traditional higher-order discretization are examined in the following sections. In contrast, standard adjoint methods do not provide local discretization error estimates and only allow one correction step to be performed. If multiple correction steps are required to eliminate a lower-order error, the accuracy of the adjoint-based error estimate will be limited to what can be achieved with one correction step.

5.4 Results

ETE relinearization is applied to several inviscid and viscous flow problems. Most of the test problems examined have an exact solution, and, therefore, the effectiveness of ETE relinearization can be directly assessed. To quantify the accuracy of each error estimate, ETE error estimates are applied as a correction to the primal solution, and, using the exact solution, the remaining discretization error is evaluated. In this section, the remaining discretization error after correction is referred to as *corrected DE*. The corrected DE is monitored because, by the following relation, it is equivalent to the error in the error estimate,

$$\varepsilon_h - \bar{\varepsilon}_h = \left(\mathbf{u}_h - I^h \tilde{\mathbf{u}}\right) - \left(\mathbf{u}_h - \mathbf{u}_{h,corrected}\right) = \mathbf{u}_{h,corrected} - I^h \tilde{\mathbf{u}} \ . \tag{5.27}$$

To illustrate convergence properties, corrected DE is compared to the exact discretization error in the lower-order primal solution, henceforth referred to as *base DE*. For instances when the ETE error estimate is to be used explicitly as a correction for the primal solution, efficiency comparisons are made with a higher-order solver. Runtime testing for select cases is performed in serial on a Dell Precision Tower 5810 workstation with an Intel Xeon E5-1650 v4 3.60 GHz processor. All reported runtimes represent the average runtime of three separate runs.

5.4.1 2D Viscous Burgers' Equation

Burgers' equation is a simplified form of the Navier-Stokes equations and is frequently used to model inviscid and viscous flow phenomenon [36, 37]. Burgers' equation may be written in conservation form as

$$\frac{\partial u}{\partial t} + \frac{\partial}{\partial x}\left(\frac{u^2}{2}\right) + \frac{\partial}{\partial y}\left(\frac{u^2}{2}\right) = \nu\left(\frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2}\right) \tag{5.28}$$

where u is a velocity and ν is a viscosity. A one dimensional form of Eq. 5.28 may be derived by neglecting all derivatives in the y-direction. The test case examined here simulates the coalescence of two viscous shock waves. Numerical results are compared to the following exact solution

$$u(\mathbf{x}) = -u_{ref}\tanh\left[(x+y)\frac{Re}{2L_{ref}}\right] \tag{5.29}$$

where $u_{ref} = 2\ m/s$ is a reference velocity, $Re = \frac{u_{ref}L_{ref}}{\nu_{ref}} = 32$ is a Reynolds number, and $L_{ref} = 8\ m$ is a reference length. The exact solution for Burgers' equation is plotted in Figure 5.1. The final steady-state, viscous shock wave can be seen along the diagonal of the computational domain.

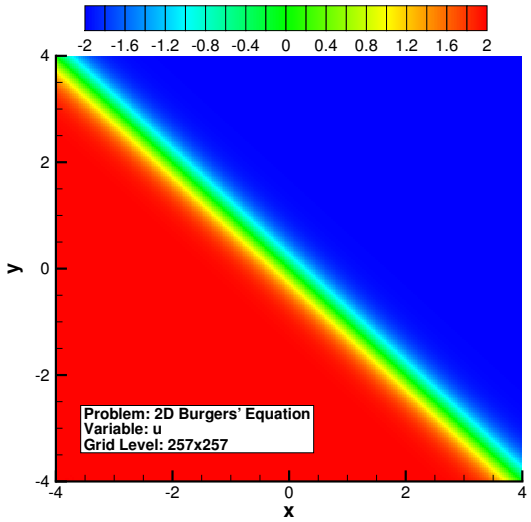


Figure 5.1: 2D Burgers' equation exact solution.

Burgers' equation is discretized using a second order, cell-centered, finite-volume discretization. Higher-order spatial accuracy is achieved using a k-exact least-squares reconstruction [33, 34]. Inviscid fluxes are computed using the Roe's flux difference splitting scheme [38, 39]. Viscous fluxes are computed using an arithmetic average of the left and right fluxes in conjunction with an additional jump term for stability [40, 41]. Dirichlet boundary conditions are enforced via Nitsche's method [42].

ETE relinearization is performed for a series of systematically-refined, uniformly-spaced "structured" grids. The reader should note that, while the grid topology is identical to that of a structured quadrilateral mesh, the grid is stored in an unstructured format and solved using an unstructured solver. Error estimates are computed using the linearized ETE. Correction steps are performed for $q = 2$ and $r \in \{4, 6, 8\}$. Therefore, according to Eq. 5.16, the order of accuracy for the error estimate should progress through $4 \rightarrow 6 \rightarrow 8$ as the relinearization loop iterates. For each value of r, 2 to 3 correction steps are performed to fully capture and correct all error terms of a given order. The convergence of discretization error norms with grid refinement is given in Figure 5.2. Asymptotically, the ETE error estimates converge at the expected rate, and ETE relinearization is able to provide higher-order accurate error estimates. As r increases, grid resolution becomes a limiting factor on the accuracy of the error estimate. If the grid is not sufficiently refined, the reconstruction may not be able to accurately capture higher-order terms in the truncation error. Consequently, the order of accuracy of the discretization error estimate may be diminished, or the error in the error estimate may even increase. For instance, on the coarsest grid level, the correction steps for $r = 8$ increase the error in the error estimate. For excessively coarse grids, correction steps for large values of r should be applied with caution.

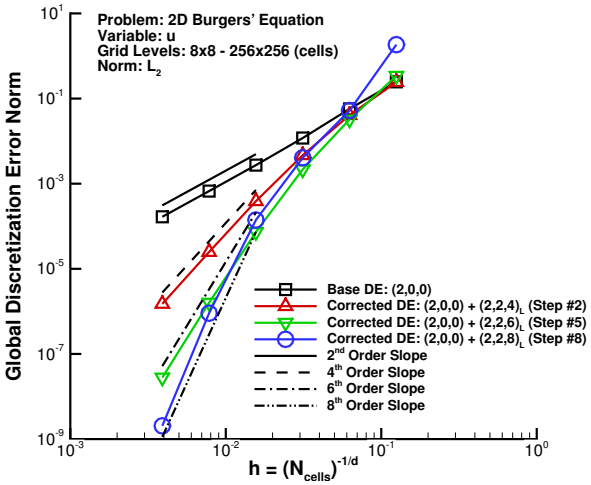


Figure 5.2: Convergence of discretization error norms.

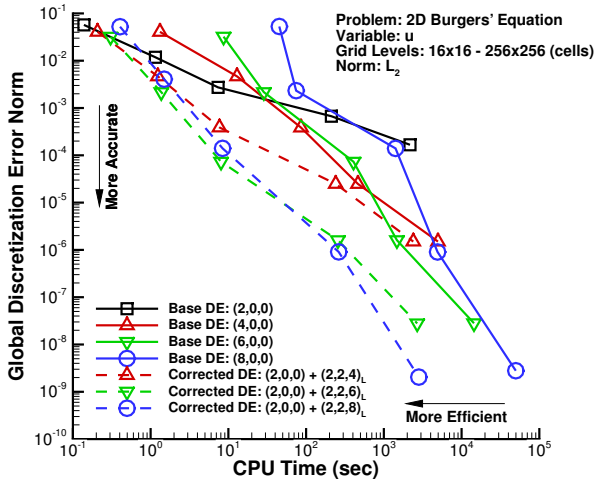
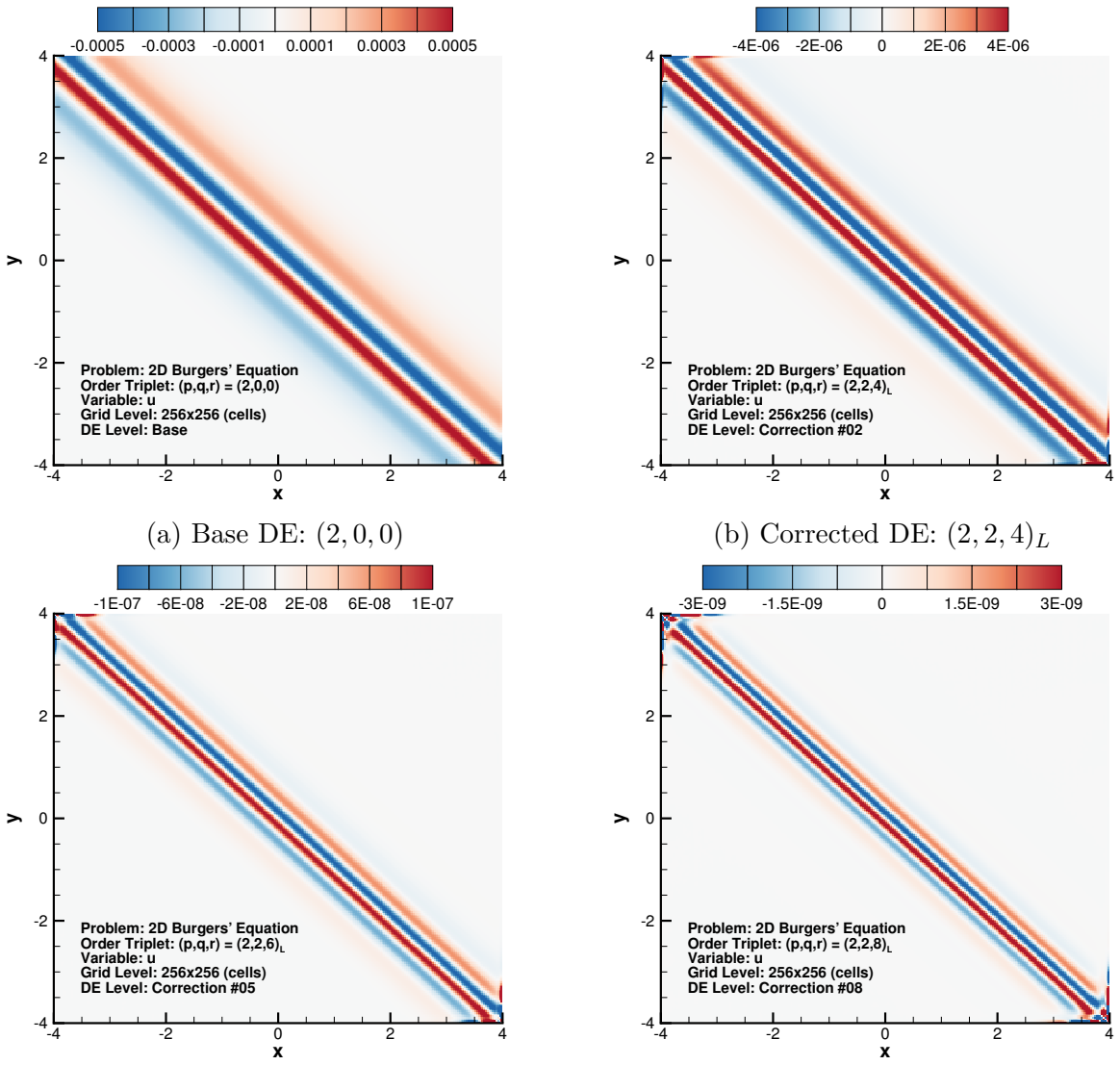


Figure 5.3: Comparison of runtimes.



(a) Base DE: $(2, 0, 0)$

(b) Corrected DE: $(2, 2, 4)_L$

(c) Corrected DE: $(2, 2, 6)_L$

(d) Corrected DE: $(2, 2, 8)_L$

Figure 5.4: Discretization error in u. This illustrates that correcting the primal solution with the linearized ETE error estimate reduces the discretization error by approximately 4-5 orders of magnitude. The reader should take note of the change in scale.

The efficiency of ETE relinearization is compared to that of a conventional higher-order finite-volume solver. The higher-order discretization is identical to the lower-order discretization with the exceptions that the solution is reconstructed to a higher degree polynomial and a higher order quadrature is used. The higher-order solver is run for $p \in \{4, 6, 8\}$, and runtimes are compared to the combined runtime of the lower-order primal solve and the ETE relinearization procedure (i.e., $(2, 0, 0)+(2, 2, r)_L$). Runtimes are reported in Figure 5.3. Asymptotically, the higher-order solver is more cost-effective in terms of computation time

than the lower-order solver for a given error level. In comparison, ETE relinearization is approximately 1 to 3 orders of magnitude faster than the higher-order solver with the benefit of ETE relinearization becoming more pronounced as r increases. Furthermore, ETE relinearization provides error levels which are just as accurate as the higher-order solver.

Base DE and corrected DE are plotted for the 256x256 grid level in Figure 5.4. Discretization error in the lower-order solution is rotationally symmetric about the diagonal and is isolated to the region of the viscous shock wave. With each set of corrections, the discretization error in the corrected solution is significantly reduced. For the final corrected solution, discretization error is the largest where the regions of highest curvature in the discrete solution intersect with the domain boundaries.

5.4.2 Quasi-1D Euler Equations

The quasi-1D Euler equations are used to model inviscid flow through a converging-diverging nozzle. These equations are a statement of conservation of mass, momentum, and energy for a fluid [43]. The quasi-1D Euler equations take the following form

$$\frac{\partial(A\mathbf{u})}{\partial t} + \frac{\partial[A\mathbf{F}(\mathbf{u})]}{\partial x} = \mathbf{s}(\mathbf{u}) \ . \tag{5.30}$$

The conserved variable vector, flux vector, and source term are given by

$$\mathbf{u} = \begin{bmatrix} \rho \\ \rho u \\ \rho e_t \end{bmatrix}, \quad \mathbf{F}(\mathbf{u}) = \begin{bmatrix} \rho u \\ \rho u^2 + p \\ \rho u h_t \end{bmatrix}, \quad \mathbf{s}(\mathbf{u}) = \begin{bmatrix} 0 \\ p\frac{dA}{dx} \\ 0 \end{bmatrix} \tag{5.31}$$

where ρ is the fluid density, u is the fluid velocity, p is the static pressure, e_t is the total energy per unit mass, h_t is the total enthalpy per unit mass, and $\frac{dA}{dx}$ describes how the cross-sectional area varies along the nozzle. The system of equations is closed using the equation of state for a perfect gas. The nozzle geometry is characterized by a Gaussian area distribution

$$A(x) = 1 - 0.8e^{\left(\frac{-x^2}{2\sigma^2}\right)}, \quad x \in [-1, +1] \tag{5.32}$$

where $\sigma = 0.2$ [44]. The quasi-1D Euler equations are nondimensionalized [18] and solved using a second order, cell-centered, finite-volume discretization. Higher-order spatial accuracy is achieved using a physical-space MUSCL reconstruction [45]. Interface fluxes are computed using Roe's flux difference splitting scheme [38]. All output data are redimensionalized to provide a better physical understanding of the results. The quasi-1D Euler equations admit two isentropic, exact solutions: (1) fully subsonic flow through the diverging section, and (2) fully supersonic flow through the diverging section. The Mach number distribution for both exact solutions as well as the area distribution may be seen in Figure 5.5. In this work, only the supersonic exact solution is examined since, in our experience, the conclusions to be drawn do not change from one solution to the other.

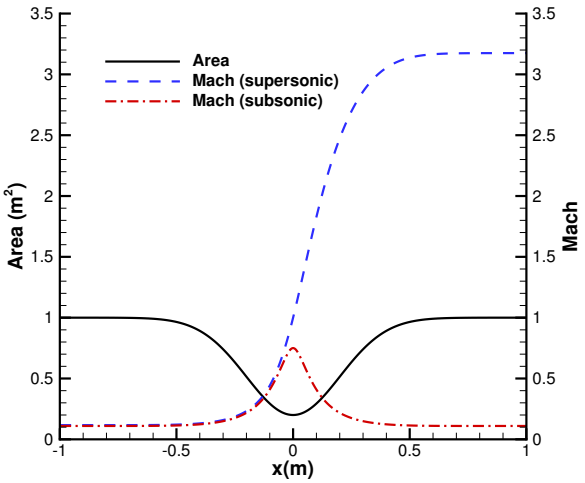


Figure 5.5: Mach number and area distribution along the nozzle.

ETE relinearization is performed for a series of systematically-refined, uniformly-spaced grids ranging in size from 32 cells to 2,048 cells. Discretization error estimates are computed using the linearized ETE with $q = 2$ and $r \in \{4, 6, 8\}$. Similar to Burgers' equation, the order of accuracy of the ETE error estimates is expected to pass through $4 \rightarrow 6 \rightarrow 8$ as the relinearization loop progresses. Furthermore, multiple correction steps are applied at each value of r. Discretization error norms for static pressure are plotted in Figure 5.6 for each set of correction steps. ETE error estimates converge at the expected rate across nearly all grid levels, and ETE relinearization is able to yield higher-order error estimates. On the two finest grid levels, the convergence rate for the error estimate with $r = 8$ does begin to degrade slightly. However, this accuracy loss results from discretization error in the nondimensional solution being near machine precision and not from a breakdown of the method.

A runtime comparison between a conventional higher-order solver and ETE relinearization is given in Figure 5.7. The higher-order solver for this test case implements a k-exact reconstruction while the lower-order solver implements a MUSCL reconstruction. The relative computational cost of a MUSCL reconstruction compared to a k-exact reconstruction is apparent; the lower-order MUSCL solve is approximately 1 to 2 orders of magnitude cheaper than the lower-order k-exact solve, albeit with a slightly higher discretization error level. When ETE relinearization is applied, the corrected MUSCL solution achieves comparable error levels to that of the higher-order k-exact solution. For a given error level, ETE relinearization is approximately 2 to 3 orders of magnitude faster than the higher-order solver. This highlights the utility of ETE relinearization; assuming that higher-order accuracy can be realized, accurate solutions can be obtained with an inexpensive lower-order solver.

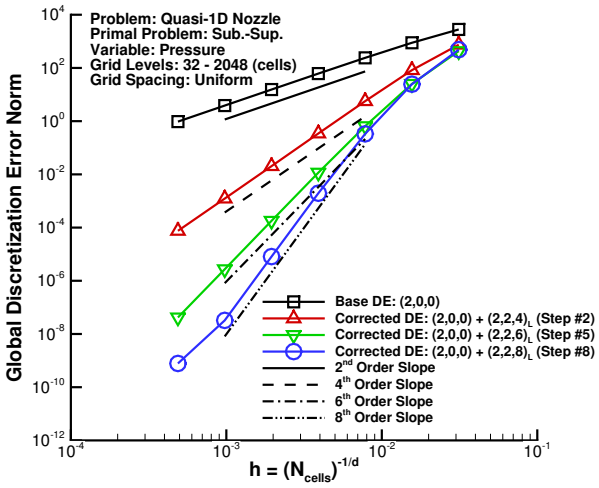


Figure 5.6: Convergence of discretization error norms.

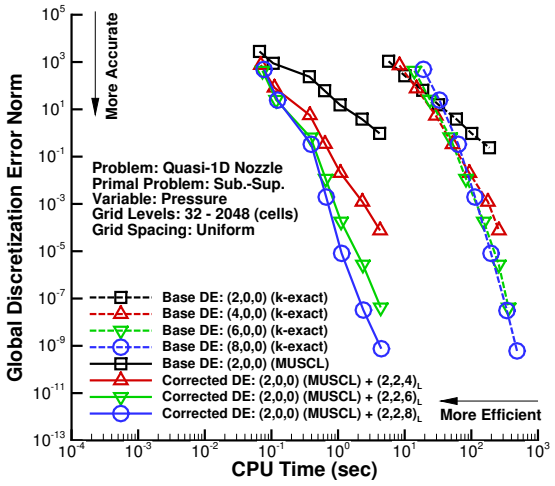


Figure 5.7: Comparison of runtimes.

5.4.3 2D Euler Equations

The Euler equations are used to model the motion of an inviscid fluid and take the same form as Eq. 5.1. The conserved variable vector, flux tensor, and source term are given by the following

$$\mathbf{u} = \begin{bmatrix} \rho \\ \rho\mathbf{v} \\ \rho e_t \end{bmatrix}, \quad \mathbb{F}(\mathbf{u}) = \begin{bmatrix} \rho\mathbf{v}^T \\ \rho\mathbf{v}\mathbf{v}^T + p\mathbf{I} \\ \rho h_t \mathbf{v}^T \end{bmatrix}, \quad \mathbf{s}(\mathbf{u}) = \begin{bmatrix} 0 \\ \mathbf{0} \\ 0 \end{bmatrix} \tag{5.33}$$

where ρ is the fluid density, p is static pressure, e_t is the total energy per unit mass, h_t is the total enthalpy per unit mass, and $\mathbf{v}$ is the velocity vector. The system of equations is closed using the equation of state for a perfect gas. The Euler equations are discretized in a similar manner to Burgers' equation.

5.4.3.1 Gaussian Bump Manufactured Solution

The proposed methods are directly evaluated using the method of manufactured solutions [32, 46]. The prescribed manufactured solution is a Gaussian bump which takes the following form

$$\phi(\mathbf{x}) = \alpha_0 + \alpha_1 \exp\Big\{-\Big[a_1(x-x_0)^2 + 2b_1(x-x_0)(y-y_0) + c_1(y-y_0)^2\Big]\Big\} \tag{5.34}$$

where $x_0 = y_0 = 0.5$ and the coefficients a_1, b_1, c_1 are given by

$$a_1 = +\frac{\cos^2(\theta)}{2\sigma_x^2} + \frac{\sin^2(\theta)}{2\sigma_y^2}, \quad b_1 = -\frac{\sin(2\theta)}{4\sigma_x^2} + \frac{\sin(2\theta)}{4\sigma_y^2}, \quad c_1 = +\frac{\sin^2(\theta)}{2\sigma_x^2} + \frac{\cos^2(\theta)}{2\sigma_y^2}\ . \tag{5.35}$$

The terms θ, σ_x, and σ_y are used to define the size and orientation of the bump. In this work, $\theta = \frac{\pi}{4}$, $\sigma_x = 0.075$, and $\sigma_y = 0.050$. The remaining coefficients, α_0 and α_1, are defined in Table 5.1 for each primitive variable. The Gaussian bump manufactured solution for static pressure is plotted in Figure 5.8.

Table 5.1: Coefficients for the Gaussian bump manufactured solution.

α \ ϕ	ρ	u	v	p
α_0	1.0	2.0	0.0	2.0
α_1	0.1	0.1	1.0	0.1

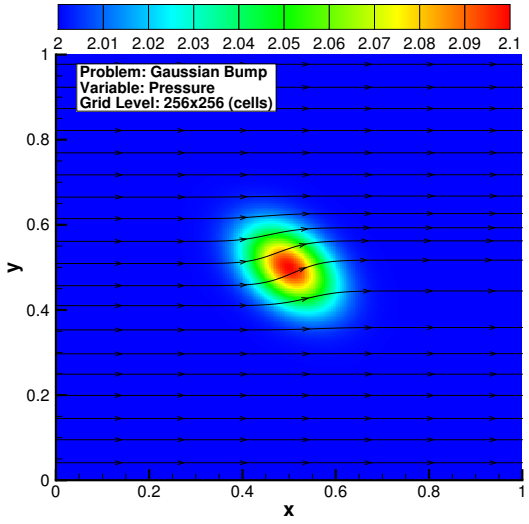


Figure 5.8: Manufactured solution for static pressure, p.

ETE relinearization is performed for a series of systematically-refined, uniformly-spaced quadrilateral grids. The linearized ETE are discretized at $q = 2$, and error estimates are obtained for $r \in \{4, 6\}$. Consequently, ETE error estimates should achieve 4^{th} order and 6^{th} order accuracy, respectively. Discretization error norms for static pressure are plotted for each grid level in Figure 5.9. ETE relinearization yields higher-order accurate error estimates once the asymptotic range has been reached. While the final set of correction steps enforces $r = 6$, the resulting error estimate is only 5^{th} order accurate. Upon further investigation, we determined that an order of accuracy was lost because of the truncation error estimate. Rather than converging at the expected 6^{th} order rate, the truncation error estimate proved to be only 5^{th} order accurate. Similar to the results for Burgers' equation, the error in the error estimate on the coarsest grid level increases because the grid is not sufficiently refined, and, as a result, the discrete solution is not asymptotic.

A runtime comparison between ETE relinearization and a higher-order solver is given in Figure 5.10. The higher-order solver utilizes the same discretization as the lower-order solver, albeit with a higher polynomial degree for the reconstruction. Since the corrected DE is only able to achieve 5^{th} order accuracy, the higher-order solver is run for $p \in \{4, 5\}$ rather than $p \in \{4, 6\}$. Similar to the previous test cases, ETE relinearization is approximately 1 to 2 orders of magnitude faster than the higher-order solver for a given error level.

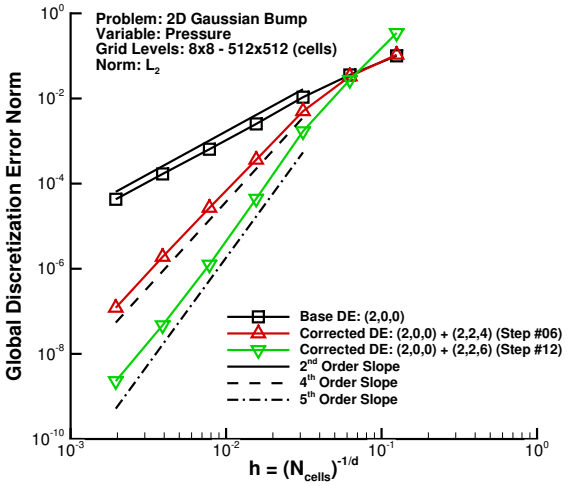


Figure 5.9: Convergence of discretization error norms.

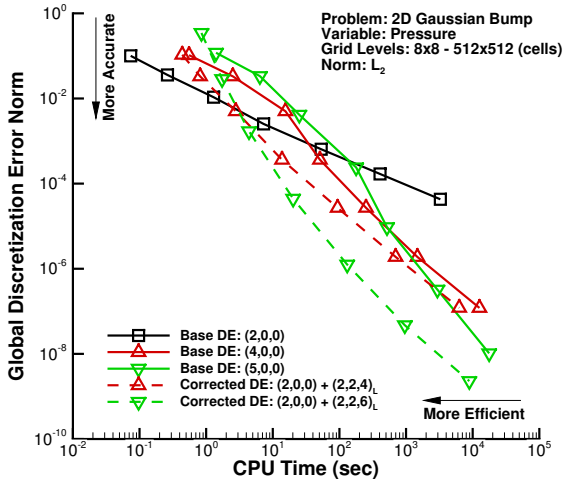
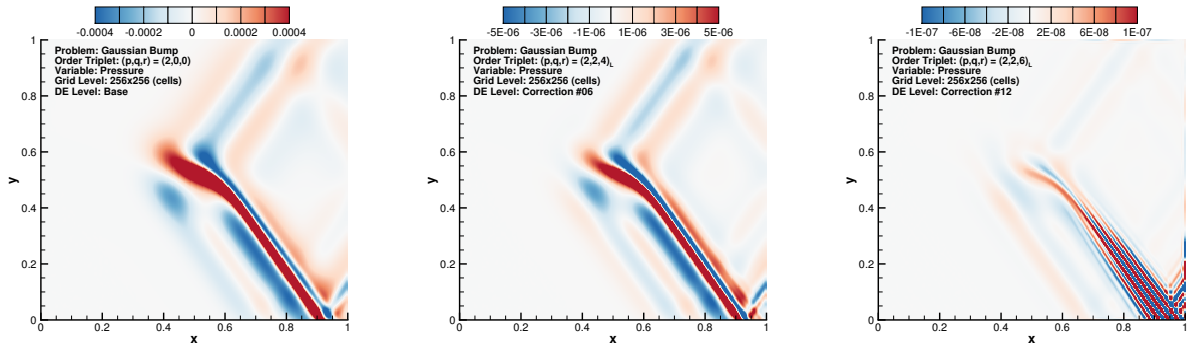


Figure 5.10: Comparison of runtimes.

Base DE and corrected DE for static pressure are plotted for a 256x256 cell grid in Figure 5.11. Discretization error in the lower-order solution is generated at the location of the bump. Since the manufactured solution is supersonic with flow traveling from left to right, discretization error is convected downstream along Mach lines. Discretization error interacts with the boundary condition along the bottom boundary and reflects toward the outflow boundary (i.e., the right boundary). With each set of correction steps, the discretization error in the corrected solution is dramatically reduced. After the final correction step, almost all of the discretization error generated at the bump has convected out of the domain. In Figure 5.11c, take note of the wave-like oscillation of the discretization error along the bottom boundary. This is the same region of the domain where the accuracy of the truncation error estimate suffered for $r = 5$. We speculate that this non-physical oscillation is exactly captured in the solution reconstruction step of the truncation error estimation procedure and degrades the accuracy of the truncation error estimate. To improve the accuracy of the truncation error estimate, a more accurate reconstruction procedure likely must be devised.



(a) Base DE: $(2, 0, 0)$ (b) Corrected DE: $(2, 2, 4)_L$ (c) Corrected DE: $(2, 2, 6)_L$

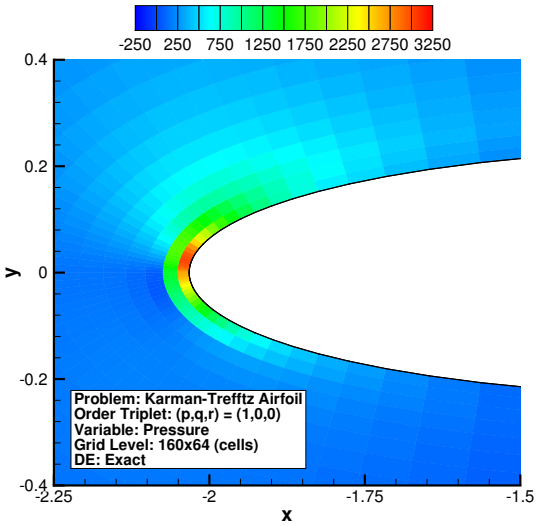
Figure 5.11: Discretization error in static pressure, p. This demonstrates that applying successive corrections can significantly reduce the amount of discretization error in the primal solution. The reader should take note of the change in scale.

5.4.3.2 Karman-Trefftz Airfoil

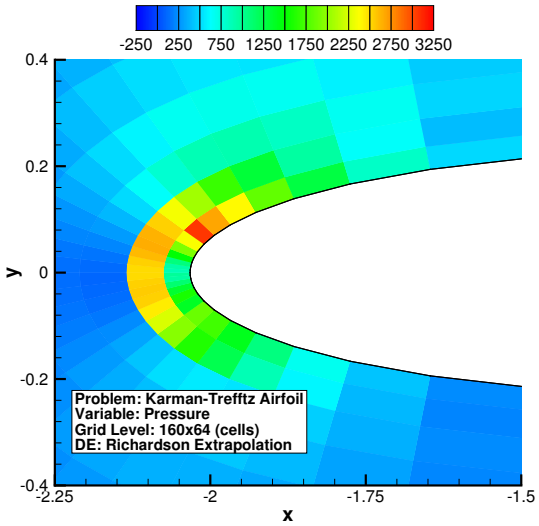
ETE error estimates are compared to those obtained via Richardson extrapolation [1] for an incompressible flow over an airfoil. The exact solution for this case is generated by mapping a potential flow over a circular cylinder using the Karman-Trefftz airfoil mapping [47]. For compressible flow, this exact solution is treated as a manufactured solution to account for any minor violations of the incompressibility assumption. The freestream conditions are $\alpha = 2°$, $M_\infty = 0.2$, $p_\infty = 100$ kPa, and $T_\infty = 300$ K. Since the code used for this case currently does not have a higher-order geometry capability, error estimates are only computed for the $p = 1$ solution. For $p \geq 2$, higher-order geometry is necessary to accurately estimate the truncation error, and subsequently, the discretization error.

Error estimates in static pressure are plotted at the leading edge of the airfoil in Figure 5.12. Discretization error is largest near the leading edge suction peak in the region just downstream of the stagnation point. Richardson extrapolation is able to capture the general behavior of the discretization error, but the resulting error estimate is not particularly accurate. The reader should also note that the Richardson extrapolation error estimate is plotted on a coarse grid. This is one of the deficiencies of Richardson extrapolation; fine grid error estimates can only be expressed at coarse grid nodes. Richardson extrapolation error estimates can be interpolated to the fine grid at the cost of introducing more error into the error estimate. In comparison, the ETE error estimate is significantly more accurate than that of Richardson extrapolation. The ETE error estimate not only provides fine grid resolution of the discretization error, but also properly identifies the region with the largest error. Moreover, when ETE relinearization is performed, the accuracy of the ETE error estimate noticeably improves. The corrected ETE error estimate is qualitatively identical to the exact discretization error. The relative accuracy of each error estimation procedure can be better assessed by examining Figure 5.13, where the error in each error estimate for

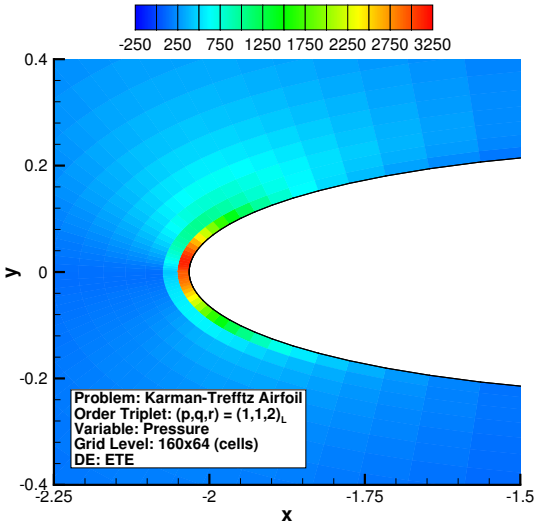
static pressure is plotted near the leading edge of the airfoil. Figure 5.13 demonstrates that ETE relinearization can provide error estimates which are much more accurate than that of Richardson extrapolation.



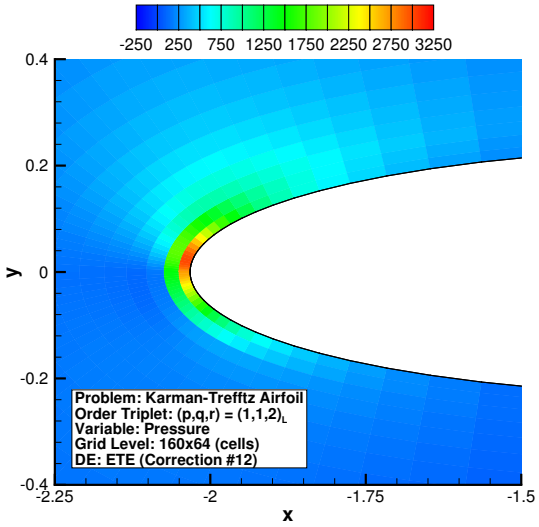
(a) Exact DE for $(1, 0, 0)$



(b) Error estimate from Richardson extrapolation

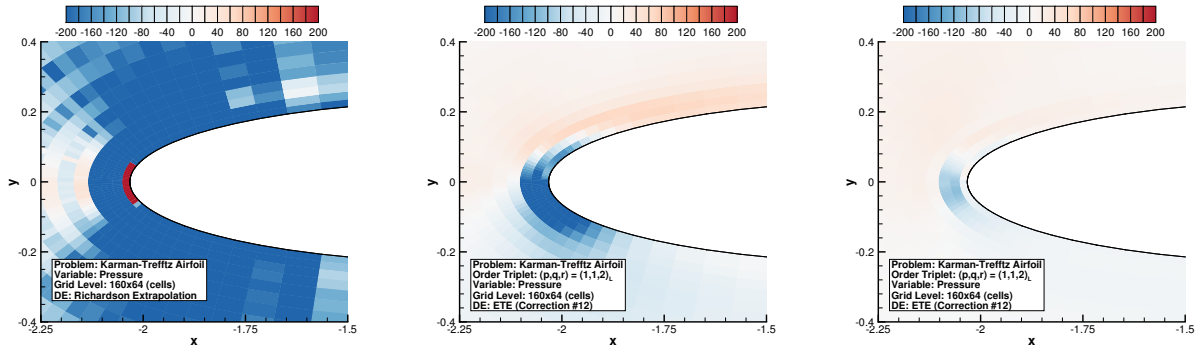


(c) ETE error estimate from $(1, 1, 2)_L$



(d) Corrected ETE error estimate from $(1, 1, 2)_L$

Figure 5.12: Discretization error in static pressure, p. This demonstrates that ETE error estimates are more accurate than a conventional Richardson extrapolation error estimate.



(a) Richardson Extrapolation (b) ETE error estimate (c) Corrected ETE error estimate

Figure 5.13: Error in error estimates for static pressure, p. ETE error estimates correspond to $(p, q, r) = (1, 1, 2)_L$. This demonstrates that ETE relinearization can provide error estimates which are significantly more accurate than a conventional Richardson extrapolation error estimate.

5.4.4 Reynolds-Averaged Navier-Stokes + SA-neg

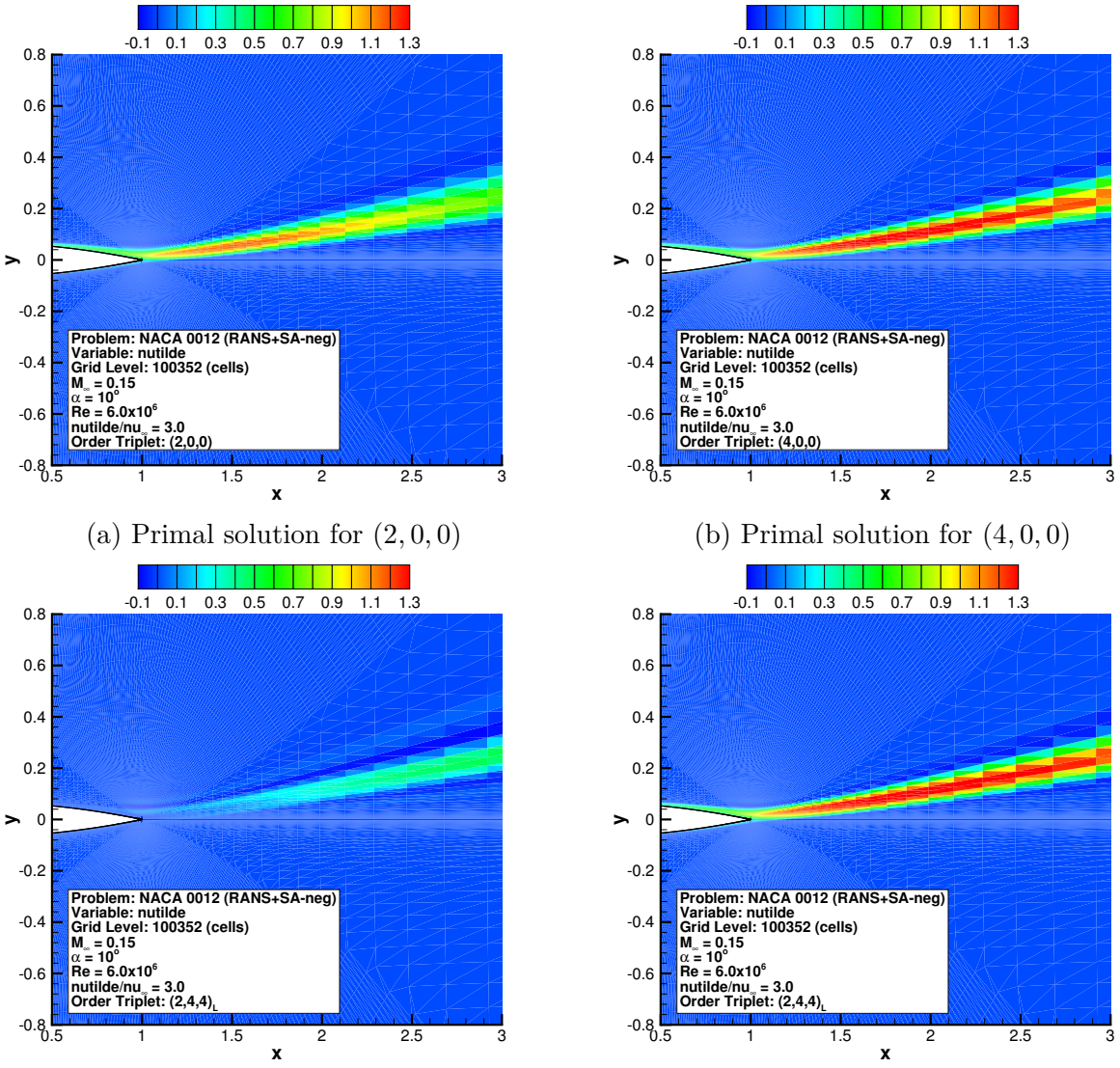
The Reynolds-averaged Navier-Stokes (RANS) equations are obtained by time-averaging [48] the Navier-Stokes equations. These equations are used to model the effects of turbulence on the mean flow of a viscous fluid. Through the process of time-averaging, additional terms known as the Reynolds stresses arise. Since the Reynolds stresses are functions of instantaneous velocity fluctuations, these terms must be modeled in order to close the system of equations. In this work, the RANS equations are closed using the negative variant of the Spalart-Allmaras turbulence model (SA-neg) [49, 50]. The RANS equations take the same form as Eq. 5.1 with the exception that the flux tensor and source term are also functions of $\nabla \mathbf{u}$. The conserved variable vector and flux tensor for the RANS equations are given by

$$\mathbf{u} = \begin{bmatrix} \rho \\ \rho \mathbf{v} \\ \rho e_t \\ \rho \tilde{\nu} \end{bmatrix}, \quad \mathbb{F}(\mathbf{u}, \nabla \mathbf{u}) = \begin{bmatrix} \rho \mathbf{v}^T \\ \rho \mathbf{v}\mathbf{v}^T + p\mathbf{I} - \tau \\ \rho h_t \mathbf{v}^T - (\tau \cdot \mathbf{v} + k_{\mathit{eff}} \nabla T) \\ \rho \mathbf{v}^T \tilde{\nu} - (\frac{\rho}{\sigma}(\nu + \tilde{\nu} f_n) \nabla \tilde{\nu}) \end{bmatrix} \tag{5.36}$$

where $\tilde{\nu}$ is the turbulence working variable, τ are viscous stresses, T is static temperature, k_{eff} is an effective thermal conductivity accounting for turbulence effects, and σ and f_n are terms associated with the turbulence model. Furthermore, the source term for the RANS equations is given by

$$\mathbf{s}(\mathbf{u}, \nabla \mathbf{u}) = \begin{bmatrix} 0 \\ \mathbf{0} \\ 0 \\ \rho(P - D) + \frac{\rho}{\sigma} c_{b2} \nabla \tilde{\nu} \cdot \nabla \tilde{\nu} - \frac{1}{\sigma}(\nu + \tilde{\nu}) \nabla \rho \cdot \nabla \tilde{\nu} \end{bmatrix} \tag{5.37}$$

where P and D are terms describing the production and destruction of turbulence, respectively, and c_{b2} is a constant. For a complete description of the SA-neg turbulence model, refer to Ref. [50].



(a) Primal solution for $(2, 0, 0)$

(b) Primal solution for $(4, 0, 0)$

(c) Error estimate from $(2, 4, 4)_L$

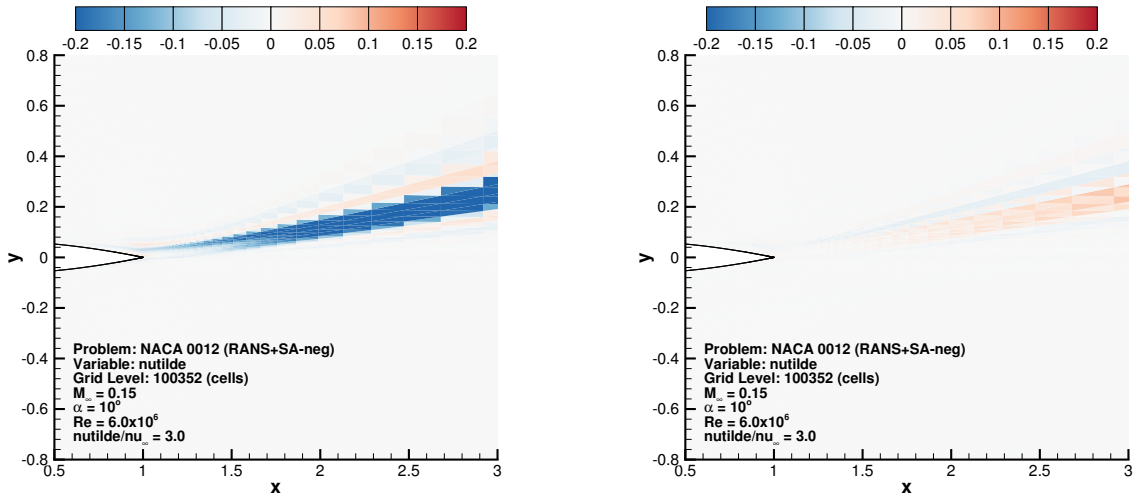
(d) Corrected solution using $(2, 4, 4)_L$

Figure 5.14: Scaled turbulence variable, error estimates, and corrected solutions. The error estimate is large in some parts of the domain, warranting the need for ETE relinearization.

The RANS+SA-neg equation set is used to model turbulent flow over an NACA 0012 airfoil at a 10° angle of attack. The freestream Mach number and Reynolds number are set to $M_\infty = 0.15$ and $Re = 6.0 \times 10^6$, respectively. The freestream turbulence working variable is set to $\tilde{\nu}_\infty = 3.0\nu_\infty$. Numerical solutions are obtained on an unstructured triangle mesh for $p \in$

$\{2, 4\}$. Higher-order spatial accuracy is achieved with a k-exact least-squares reconstruction. Characteristic boundary conditions are enforced in the farfield, and an adiabatic, no-slip boundary condition is imposed at the airfoil surface. Inviscid fluxes are computed using Roe's scheme [38]. Viscous fluxes are computed in a similar manner to Burgers' equation. For this test case, the linearized ETE are discretized at $q = 4$ rather than $q = 2$. A higher-order discretization of the ETE is necessary because current unstructured grid truncation error estimation techniques are unable to approximate truncation error to a sufficient level of accuracy for higher-order error estimation [28].

Since this test case does not have an exact solution, the accuracy of ETE error estimates cannot be directly evaluated. However, this case does illustrate the utility of ETE relinearization for practical problems, especially when linearization errors may not be small. Linearization errors, which predominantly vary according to the square of the second-order discretization error, can be sizable when flow features are unresolved. In turbulence modeling, it is common for flow features to go unresolved until the grid is sufficiently refined and the turbulent source terms are fully engaged. Consider the primal solutions for $p = 2$ and $p = 4$ plotted in Figure 5.14a and Figure 5.14b, respectively, where a scaled turbulence variable [51] is used as the solution. The difference between the lower-order and higher-order solutions is quite noticeable. The ETE error estimate, which is plotted in Figure 5.14c, identifies that discretization error may be large in the airfoil wake. Since the ETE error estimate is of a similar magnitude to the solution and the difference between $p = 2$ and $p = 4$ solutions is so drastic, ETE relinearization may be warranted. After two relinearization steps, the corrected primal solution, which is plotted in Figure 5.14d, appears qualitatively identical to the higher-order solution. The improvement in the lower-order solution can be more clearly seen in Figure 5.15. After correction, the difference between the lower-order solution and higher-order solution is largely reduced.



(a) Difference between $(2, 0, 0)$ and $(4, 0, 0)$

(b) Difference between $(2, 4, 4)_L$ corrected and $(4, 0, 0)$

Figure 5.15: Estimate of error in the scaled turbulence variable for the second order primal solution and the linearized ETE corrected solution. This demonstrates that the first linearization error is large and can potentially dominate.

5.4.5 Accuracy and Robustness Considerations

Assuming higher-order accuracy can be obtained, ETE relinearization can be computationally competitive with existing higher-order finite-volume schemes. By utilizing ETE relinearization as a type of defect correction, practitioners may retain their lower-order solver even for problems where a high degree of accuracy is required. Furthermore, lower-order solvers are typically more stable and require less memory than higher-order solvers. However, when applying the ETE error estimate as a correction, care must be taken to ensure that all requirements for higher-order accuracy are met. For instance, the ETE source term (i.e., the truncation error estimate) must be at least r^{th} order accurate or else the convergence rate of the discretization error estimate will be limited. The discrete solution must also be sufficiently near or within the asymptotic range so that linearization errors are small enough to be captured and corrected by the relinearization procedure. In the presence of strong flow-field discontinuities, it is still unclear how ETE relinearization will perform. Banks et al. [23] have demonstrated that linearized ETE error estimates can become unbounded in the region immediately surrounding a discontinuity. Therefore, we speculate that ETE relinearization will work well in smooth regions of the flow upstream of a discontinuity. At the discontinuity, the full nonlinear ETE may have to be substituted for the linearized ETE. In smooth regions of the flow downstream of a discontinuity, further study is required to determine the applicability of ETE relinearization. Discretization errors generated at the discontinuity may convect downstream and could potentially affect the accuracy of the relinearization

procedure.

5.5 Conclusion

A linearized error transport equation (ETE) was derived and solved to estimate local discretization errors for finite-volume schemes. Truncation error was shown to act as the source term for discretization error in numerical solutions. Under certain conditions, ETE error estimates were shown to converge to the true discretization error at a higher-order rate. A new procedure, referred to as ETE relinearization, was presented which can theoretically yield error estimates of an arbitrary order of accuracy. ETE relinearization was applied to several inviscid and viscous flow problems of varying complexity. Discretization error estimates as high as 8^{th} order accurate were obtained. ETE relinearization was also demonstrated as a correction procedure for the primal solution. In comparison to a conventional higher-order solver, ETE relinearization provided similar levels of accuracy at a significantly reduced computational cost. For a given error level, ETE relinearization was on average 1 to 3 orders of magnitude faster than a higher-order solver. These speedups were achieved because ETE relinearization replaced a full higher-order nonlinear solve with a lower-order nonlinear solve plus the solution of just a few additional linear systems. ETE relinearization was compared to a standard Richardson extrapolation error estimation procedure. Corrected ETE error estimates were shown to be significantly more accurate than those obtained with Richardson extrapolation. For turbulent flows with unresolved features, ETE relinearization provided a mechanism for reducing linearization error and improving solution quality.

Acknowledgments

This work was supported by the NASA Graduate Aeronautics Scholarship as part of the Aeronautics Scholarship and Advanced STEM Training and Research Fellowship (AS&ASTAR) Program. We wish to thank Dr. Michael Park and Dr. Joseph Derlaga of NASA Langley Research Center for serving as technical advisors.

Bibliography

[1] L. Richardson, On the approximate arithmetical solution by finite differences of physical problems involving differential equations, with an application to the stresses in a masonry dam, Proceedings of the Royal Society of London A: Mathematical, Physical and Engineering Sciences 83 (563) (1910) 335–336. doi:10.1098/rspa.1910.0020. URL http://rspa.royalsocietypublishing.org/content/83/563/335

[2] J. Peraire, M. Vahdati, K. Morgan, O. Zienkiewicz, Adaptive remeshing for compressible flow computations, Journal of Computational Physics 72 (2) (1987) 449 – 466. doi:10.1016/0021-9991(87)90093-3.
URL http://dx.doi.org/10.1016/0021-9991(87)90093-3

[3] G. Warren, W. Anderson, J. Thomas, S. Krist, Grid convergence for adaptive methods, in: 10th Computational Fluid Dynamics Conference, American Institute of Aeronautics and Astronautics (AIAA), 1991. doi:10.2514/6.1991-1592.
URL http://dx.doi.org/10.2514/6.1991-1592

[4] I. Babuška, T.Strouboulis, C. Upadhyay, A model study of the quality of a posteriori error estimators for linear elliptic problems. Error estimation in the interior of patchwise uniform grids of triangles, Computer Methods in Applied Mechanics and Engineering 114 (1994) 307–378. doi:10.1016/0045-7825(94)90177-5.
URL https://doi.org/10.1016/0045-7825(94)90177-5

[5] I. Babuška, T.Strouboulis, S. Gangaraj, C. Upadhyay, Pollution error in the h-version of the finite element method and the local quality of the recovered derivatives, Computer Methods in Applied Mechanics and Engineering 140 (1997) 1–37. doi:10.1016/S0045-7825(96)01013-4.
URL https://doi.org/10.1016/S0045-7825(96)01013-4

[6] X. Gu, T. Shih, Differentiating between source and location of error for solution-adaptive mesh refinement, in: 15th AIAA Computational Fluid Dynamics Conference, American Institute of Aeronautics and Astronautics (AIAA), 2001. doi:10.2514/6.2001-2660.
URL http://dx.doi.org/10.2514/6.2001-2660

[7] X. Zhang, J.-Y. Trépanier, R. Camarero, A posteriori error estimation for finite-volume solutions of hyperbolic conservation laws, Computer Methods in Applied Mechanics and Engineering 185 (1) (2000) 1 – 19. doi:10.1016/s0045-7825(99)00099-7.
URL http://dx.doi.org/10.1016/S0045-7825(99)00099-7

[8] Y. Qin, P. Keller, R. Sun, E. Hernandez, C.-Y. Perng, N. Trigui, Z. Han, F. Shen, T. Shieh, T.-P. Shih, Estimating grid-induced errors in CFD by discrete-error-transport equations, in: 42nd AIAA Aerospace Sciences Meeting and Exhibit, American Institute of Aeronautics and Astronautics (AIAA), 2004. doi:10.2514/6.2004-656.
URL https://arc.aiaa.org/doi/pdf/10.2514/6.2004-656

[9] I. Celik, G. Hu, Single grid error estimation using error transport equation, Journal of Fluids Engineering 126 (5) (2004) 778. doi:10.1115/1.1792254.
URL http://dx.doi.org/10.1115/1.1792254

[10] P. Cavallo, N. Sinha, An error transport equation with practical applications, in: 18th AIAA Computational Fluid Dynamics Conference, American Institute of Aeronautics

and Astronautics (AIAA), 2007, p. 4092. doi:10.2514/6.2007-4092.
URL https://arc.aiaa.org/doi/pdf/10.2514/6.2007-4092

[11] J. M. Derlaga, M. A. Park, S. K. Rallabhandi, Application of exact error transport equations and adjoint error estimation to AIAA workshops, in: 55th AIAA Aerospace Sciences Meeting, American Institute of Aeronautics and Astronautics (AIAA), 2017, p. 0076. doi:10.2514/6.2017-0076.
URL https://doi.org/10.2514/6.2017-0076

[12] G. Yan, C. Ollivier-Gooch, Towards higher order discretization error estimation by error transport using unstructured finite-volume methods for unsteady problems, Computers & Fluids 154 (2017) 245–255. doi:10.1016/j.compfluid.2017.06.012.
URL https://doi.org/10.1016/j.compfluid.2017.06.012

[13] M. B. Giles, N. A. Pierce, Adjoint Error Correction for Integral Outputs, Springer Berlin Heidelberg, Berlin, Heidelberg, 2003, Ch. 2, pp. 47 – 95. doi:10.1007/978-3-662-05189-4_2.
URL http://dx.doi.org/10.1007/978-3-662-05189-4_2

[14] M. B. Giles, N. Pierce, E. Süli, Progress in adjoint error correction for integral functionals, Computing and Visualization in Science 6 (2-3) (2004) 113 – 121. doi:10.1007/s00791-003-0115-y.
URL http://dx.doi.org/10.1007/s00791-003-0115-y

[15] D. A. Venditti, Grid adaptation for functional outputs of compressible flow simulations, Ph.D. thesis, Massachusetts Institute of Technology (Jun. 2002).
URL http://hdl.handle.net/1721.1/29246

[16] M. Sharbatdar, A. Jalali, C. Ollivier-Gooch, Smoothed truncation error in functional error estimation and correction using adjoint methods in an unstructured finite-volume solver, Computers & Fluids 140 (2016) 406–421. doi:10.1016/j.compfluid.2016.10.019.
URL https://doi.org/10.1016/j.compfluid.2016.10.019

[17] N. A. Pierce, M. B. Giles, Adjoint recovery of superconvergent functionals from PDE approximations, SIAM Review 42 (2) (2000) 247 – 264. doi:10.1137/s0036144598349423.
URL http://dx.doi.org/10.1137/S0036144598349423

[18] W. C. Tyson, K. Swirydowicz, J. M. Derlaga, C. J. Roy, E. de Sturler, Improved functional-based error estimation and adaptation without adjoints, in: 46th AIAA Fluid Dynamics Conference, American Institute of Aeronautics and Astronautics (AIAA), 2016. doi:10.2514/6.2016-3809.
URL http://dx.doi.org/10.2514/6.2016-3809

[19] D. A. Venditti, D. L. Darmofal, Adjoint error estimation and grid adaptation for functional outputs: Application to quasi-one-dimensional flow, Journal of Computational Physics 164 (1) (2000) 204 – 227. doi:10.1006/jcph.2000.6600.
URL http://dx.doi.org/10.1006/jcph.2000.6600

[20] M. Nemec, M. Aftosmis, M. Wintzer, Adjoint-based adaptive mesh refinement for complex geometries, in: 46th AIAA Aerospace Sciences Meeting and Exhibit, American Institute of Aeronautics and Astronautics (AIAA), 2008. doi:10.2514/6.2008-725.
URL http://dx.doi.org/10.2514/6.2008-725

[21] R. Hartmann, Multitarget error estimation and adaptivity in aerodynamic flow simulations, SIAM Journal on Scientific Computing 31 (1) (2008) 708 – 731. doi:10.1137/070710962.
URL https://doi.org/10.1137/070710962

[22] A. Hay, M. Visonneau, Error estimation using the error transport equation for finite-volume methods and arbitrary meshes, International Journal of Computational Fluid Dynamics 20 (7) (2006) 463 – 479. doi:10.1080/10618560600835934.
URL http://dx.doi.org/10.1080/10618560600835934

[23] J. Banks, J. Hittinger, J. Connors, C. Woodward, Numerical error estimation for nonlinear hyperbolic PDEs via nonlinear error transport, Computer Methods in Applied Mechanics and Engineering 213-216 (2012) 1 – 15. doi:10.1016/j.cma.2011.11.021.
URL http://dx.doi.org/10.1016/j.cma.2011.11.021

[24] L. Fox, Some improvements in the use of relaxation methods for the solution of ordinary and partial differential equations, Proceedings of the Royal Society of London. Series A, Mathematical and Physical Sciences 190 (1020) (1947) 31 – 59.
URL http://www.jstor.org/stable/98009

[25] V. Pereyra, Iterated deferred corrections for nonlinear operator equations, Numerische Mathematik 10 (4) (1967) 316 – 323. doi:10.1007/bf02162030.
URL http://dx.doi.org/10.1007/BF02162030

[26] H. J. Stetter, The defect correction principle and discretization methods, Numerische Mathematik 29 (4) (1978) 425 – 443. doi:10.1007/bf01432879.
URL http://dx.doi.org/10.1007/BF01432879

[27] W. C. Tyson, C. J. Roy, A hybrid adjoint/error transport approach to error estimation, adaptation, and higher-order solutions for computational fluid dynamics, in: 55th AIAA Aerospace Sciences Meeting, American Institute of Aeronautics and Astronautics (AIAA), 2017. doi:10.2514/6.2017-0741.
URL https://doi.org/10.2514%2F6.2017-0741

[28] G. K. Yan, C. Ollivier-Gooch, On efficiently obtaining higher order accurate discretization error estimates for unstructured finite volume methods using the error transport equation, Journal of Verification, Validation and Uncertainty Quantification 2 (4) (2017) 1–17. doi:10.1115/1.4039188.
URL http://dx.doi.org/10.1115/1.4039188

[29] C. Ollivier-Gooch, C. Roy, Reducing truncation error on unstructured meshes by vertex movement, in: 42nd AIAA Fluid Dynamics Conference and Exhibit, American Institute of Aeronautics and Astronautics (AIAA), 2012. doi:10.2514/6.2012-2712.
URL http://dx.doi.org/10.2514/6.2012-2712

[30] M. Sharbatdar, Error estimation and mesh adaptation paradigm for unstructured mesh finite volume methods, Ph.D. thesis, University of British Columbia (Jan. 2017).
URL http://hdl.handle.net/2429/60359

[31] C. Roy, Strategies for driving mesh adaptation in CFD (invited), in: 47th AIAA Aerospace Sciences Meeting including the New Horizons Forum and Aerospace Exposition, American Institute of Aeronautics and Astronautics (AIAA), 2009. doi:10.2514/6.2009-1302.
URL http://dx.doi.org/10.2514/6.2009-1302

[32] W. L. Oberkampf, C. J. Roy, Verification and Validation in Scientific Computing, Cambridge University Press (CUP), 2010. doi:10.1017/cbo9780511760396.
URL http://dx.doi.org/10.1017/CBO9780511760396

[33] T. Barth, P. Frederickson, Higher order solution of the Euler equations on unstructured grids using quadratic reconstruction, in: 28th Aerospace Sciences Meeting, American Institute of Aeronautics and Astronautics (AIAA), 1990. doi:10.2514/6.1990-13.
URL http://dx.doi.org/10.2514/6.1990-13

[34] C. Ollivier-Gooch, A. Nejat, K. Michalak, Obtaining and verifying high-order unstructured finite volume solutions to the Euler equations, AIAA Journal 47 (9) (2009) 2105 – 2120. doi:10.2514/1.40585.
URL http://dx.doi.org/10.2514/1.40585

[35] G. H. Golub, C. Van Loan, Matrix Computations, 3rd Edition, Johns Hopkins University Press, 1996.

[36] H. Bateman, Some recent researches on the motion of fluids, Monthly Weather Review 43 (4) (1915) 163–170. doi:10.1175/1520-0493(1915)43<163:SRROTM>2.0.CO;2.
URL https://doi.org/10.1175/1520-0493(1915)43<163:SRROTM>2.0.CO;2

[37] J. Burgers, A mathematical model illustrating the theory of turbulence, Advances in Applied Mechanics 1 (1948) 171 – 199. doi:10.1016/S0065-2156(08)70100-5.
URL https://doi.org/10.1016/S0065-2156(08)70100-5

[38] P. Roe, Approximate Riemann solvers, parameter vectors, and difference schemes, Journal of Computational Physics 43 (2) (1981) 357 – 372. doi:10.1016/0021-9991(81)90128-5.
URL http://dx.doi.org/10.1016/0021-9991(81)90128-5

[39] R. H. Pletcher, J. C. Tannehill, D. Anderson, Computational fluid mechanics and heat transfer, CRC Press, 2012.

[40] H. Nishikawa, Beyond interface gradient: A general principle for constructing diffusion schemes, in: 40th Fluid Dynamics Conference and Exhibit, American Institute of Aeronautics and Astronautics (AIAA), 2010, p. 5093. doi:10.2514/6.2010-5093.
URL https://doi.org/10.2514/6.2010-5093

[41] A. Jalali, M. Sharbatdar, C. Ollivier-Gooch, Accuracy analysis of unstructured finite volume discretization schemes for diffusive fluxes, Computers & Fluids 101 (2014) 220 – 232. doi:10.1016/j.compfluid.2014.06.008.
URL https://doi.org/10.1016/j.compfluid.2014.06.008

[42] J. Nitsche, Über ein Variations zur Lösung von Dirichlet-Problemen bei Verwendung von Teilräumen die keinen Randbedingungen unterworfen sind, Abhandlungen aus dem mathematischen Seminar der Universität Hamburg 36 (1971) 9 – 15.

[43] C. Hirsch, Numerical Computation of Internal and External Flows: Volume 2, John Wiley & Sons Ltd., 1990.

[44] C. W. Jackson, C. J. Roy, A multi-mesh CFD technique for adaptive mesh solutions, in: 53rd AIAA Aerospace Sciences Meeting, American Institute of Aeronautics and Astronautics (AIAA), 2015. doi:10.2514/6.2015-1958.
URL http://dx.doi.org/10.2514/6.2015-1958

[45] B. van Leer, Towards the ultimate conservative difference scheme. V. a second-order sequel to Godunov's method, Journal of Computational Physics 32 (1) (1979) 101 – 136. doi:10.1016/0021-9991(79)90145-1.
URL http://dx.doi.org/10.1016/0021-9991(79)90145-1

[46] P. J. Roache, Code verification by the method of manufactured solutions, Journal of Fluids Engineering 124 (1) (2002) 4. doi:10.1115/1.1436090.
URL http://dx.doi.org/10.1115/1.1436090

[47] K. Karamcheti, Principles of Ideal-Fluid Aerodynamics, Krieger Publishing Compansion, 1966.

[48] O. Reynolds, On the dynamical theory of incompressible viscous fluids and the determination of the criterion, Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences 186 (0) (1895) 123 – 164. doi:10.1098/rsta.1895.

0004.
URL http://dx.doi.org/10.1098/rsta.1895.0004

[49] P. Spalart, S. Allmaras, A one-equation turbulence model for aerodynamic flows, 30th Aerospace Sciences Meeting and Exhibitdoi:10.2514/6.1992-439.
URL http://dx.doi.org/10.2514/6.1992-439

[50] S. Allmaras, F. Johnson, P. Spalart, Modifications and clarifications for the implementation of the Spalart-Allmaras turbulence model, in: Seventh International Conference on Computational Fluid Dynamics, 2012.

[51] M. Ceze, K. Fidkowski, Pseudo-transient continuation, solution update methods, and CFL strategies for DG discretizations of the RANS-SA equations, in: 21st AIAA Computational Fluid Dynamics Conference, American Institute of Aeronautics and Astronautics (AIAA), 2013, p. 2686. doi:10.2514/6.2013-2686.
URL https://doi.org/10.2514/6.2013-2686

Chapter 6

BlueRidge: A Higher-Order Finite-Volume Solver

William C. Tyson,[a] Charles W. Jackson,[a] Christopher J. Roy[a]

[a] Kevin T. Crofton Department of Aerospace and Ocean Engineering, Virginia Tech
215 Randolph Hall, 460 Old Turner Street, Blacksburg, VA, 24061

Keywords: Finite-volume method, Higher-order method, Software engineering, Code verification

Attribution

- William C. Tyson (First Author): The first author served as the main contributor and primary author of the work. The first author wrote the majority of the source code in BlueRidge including but no limited to input/output routines, linear and nonlinear solvers, reconstruction routines, and physics subclasses. The first author wrote all unit tests for BlueRidge and performed rigorous code verification.
- Charles W. Jackson (Second Author): The second author provided valuable guidance with regard to code organization and data management. The second author contributed to the code development section of the manuscript and provided helpful comments throughout the remaining sections.
- Christopher J. Roy (Third Author): The third author provided valuable guidance and insight during the course of the study. The third author also provided helpful comments on the manuscript.

Abstract

This paper outlines the development of BlueRidge, a higher-order finite-volume solver. Higher-order spatial accuracy is achieved using a k-exact least-squares reconstruction. The equation sets currently implemented in BlueRidge include the Euler and Navier-Stokes equations, Burgers' equation, and Laplace's equation. Numerical solutions are marched in time using explicit and implicit time integrators. Adjoint and error transport solvers are included for the purpose of numerical error estimation. BlueRidge is verified at higher-order accuracy using the method of manufactured solutions. BlueRidge is written using the Fortran 03/08 standards and makes extensive use of abstraction mechanisms and object-oriented programming to enable a modular and flexible code base. A discussion of best practices is provided to help developers improve code quality and expedite programming workflows.

6.1 Introduction

Numerous physical phenomena, such as heat transfer, electrostatics, elasticity, and fluid mechanics, can be modeled by partial differential equations (PDEs). Due to their complexity, PDEs generally cannot be solved analytically, and must instead be solved numerically. With the advent of modern computing hardware and efficient numerical algorithms, computer codes which numerically solve PDEs have become an indispensable part of an engineer's toolbox. In fact, simulation-based design has become increasingly more common as simulations are typically faster and cheaper than experiments.

Traditionally, engineers have written computer codes explicitly for their given application. This narrow approach to code design, however, can often have unintended consequences. For instance, in interdisciplinary applications, specialty codes must be daisy-chained together via an elaborate series of input/output scripts. Furthermore, as customer demands change over time, the addition of new features may be required. If the original code base is not sufficiently flexible, the addition of new features can be challenging and may even require a complete overhaul of the code. To ensure the longevity of a code base, developers must write code with flexibility and maintainability in mind. This approach to code development requires considerable planning as well as modularization and abstraction of code functionality. Modularization can reduce the amount of duplicate code in a code base, allows programmers to independently test different procedures, and encapsulates data to prevent unintended modification. Abstraction refers to a programming technique where a common task is performed without knowledge of its actual implementation. Through the definition of a standard interface, abstraction mechanisms enable high-level routines to easily access different implementations of a particular procedure. Code abstraction allows future developers to add functionality without having to rewrite code or needing intimate knowledge of the entire code base. Many popular programming languages, such as C++ and Fortran, facilitate modularity and abstraction through object-oriented programming constructs. While

typically front-ended in terms of labor costs, this coding philosophy can provide a flexible code base and can lead to significant savings in future development time.

In this work, the development of BlueRidge, a generic, finite-volume, PDE solver, is described. We make extensive use of the object-orientation and abstraction mechanisms currently available in the Fortran 03/08 coding standards. High-level routines are designed to be agnostic to the underlying physics being simulated. Consequently, a large percentage of the code base can be reused as new equation sets are added. The equation sets currently available within BlueRidge are the Euler and Navier-Stokes equations, Burgers' equation, and Laplace's equation. For the primal solver, BlueRidge employs a higher-order finite-volume discretization. BlueRidge also includes adjoint and error transport solvers for the purpose of numerical error estimation. BlueRidge is verified using the method of manufactured solutions. A suite of exact and manufactured solutions is developed and included in the code base to provide a test bed for code verification and ongoing research efforts. To assist other programmers and engineers who seek code longevity, a discussion of best practices is provided in addition to explanations of design choices.

6.2 Generic Finite-Volume Solver

6.2.1 Finite-Volume Discretization

BlueRidge employs a higher-order, finite-volume discretization. The finite-volume method is one of the most widely used techniques for numerically solving PDEs as it is robust and naturally conservative. Consider a general conservation law of the form

$$\frac{\partial \tilde{\mathbf{u}}}{\partial t} + \nabla \cdot \mathbb{F}(\tilde{\mathbf{u}}) = \mathbf{s}(\tilde{\mathbf{u}}) \tag{6.1}$$

defined over the domain $\Omega \subset \mathbb{R}^m \times [0, \infty), \mathbf{x} \in \mathbb{R}^m, m \in \{1, 2, 3\}$ where $\tilde{\mathbf{u}}$ are the conserved quantities, $\mathbb{F}(\cdot)$ is a flux tensor, and $\mathbf{s}(\cdot)$ is a source term. In general, the flux tensor and source term can also be functions of $\nabla\tilde{\mathbf{u}}$, but this dependency has been omitted here for the sake of brevity. It is assumed that the domain of interest is discretized into a set of non-overlapping control volumes, $\mathcal{T}_h$, such that

$$\Omega = \bigcup_{i=1}^{N_v} \Omega_i, \quad \Omega_i \in \mathcal{T}_h \tag{6.2}$$

where N_v is the total number of control volumes in the computational domain. Eq. 6.1 is recast into a weak form by integrating over each control volume, Ω_i, and using the Gauss divergence theorem to convert the flux divergence integral into a surface integral,

$$\mathbf{N}(\tilde{\mathbf{u}}) \coloneqq \frac{\partial}{\partial t} \int_{\Omega_i} \tilde{\mathbf{u}} \; d\Omega + \oint_{\partial\Omega_i} \mathbb{F}(\tilde{\mathbf{u}}) \cdot \hat{\mathbf{n}} \; dS - \int_{\Omega_i} \mathbf{s}(\tilde{\mathbf{u}}) \; d\Omega = \mathbf{0} \; . \tag{6.3}$$

It is also assumed that each control volume is fixed in space so that, by the Reynolds transport theorem, the time derivative may be pulled out of the first integral. In the finite-volume method, a discrete solution, $\mathbf{u}_h$, is sought which approximates the control volume average of the exact continuous solution, $\tilde{\mathbf{u}}$. In accordance with this assumption, the discrete form of Eq. 6.3 may be written as

$$\mathbf{N}_h^p(\mathbf{u}_h) \coloneqq |\Omega_i| \frac{d\mathbf{u}_h}{dt} + \oint_{\partial\Omega_i} \mathbb{F}_h\Big(I_h^p \mathbf{u}_h^+, I_h^p \mathbf{u}_h^-\Big) \cdot \hat{\mathbf{n}} \; dS - \int_{\Omega_i} \mathbf{s}\Big(I_h^p \mathbf{u}_h\Big) \; d\Omega = \mathbf{0} \tag{6.4}$$

where $|\Omega_i|$ is the volume of Ω_i. The analytic flux, $\mathbb{F}(\cdot)$, has been replaced by a discrete flux function, $\mathbb{F}_h(\cdot,\cdot)$, to account for the discontinuous solution at control volume interfaces. The superscript p denotes the design order of accuracy for the primal discretization. The operator I_a^b is a restriction or prolongation operator. The subscript a denotes the starting space and the superscript b denotes the resultant space. An empty subscript or superscript denotes a continuous space, and a subscript or superscript h denotes a discrete space on a mesh with a characteristic cell size h. In Eq. 6.4, the operator I_h^p prolongs the discrete solution, $\mathbf{u}_h$, to a p^{th} order continuous space, typically a polynomial space. The notation $(\cdot)^{+/-}$ is used to indicate the trace of the reconstructed solution from the exterior and interior of the control volume, respectively. Eq. 6.4 may be written for each control volume in $\mathcal{T}_h$ to form the following system of ODEs

$$|\boldsymbol{\Omega}| \frac{d\mathbf{u}_h}{dt} + \mathbf{R}(\mathbf{u}_h) = \mathbf{0} \tag{6.5}$$

where $\mathbf{R}(\mathbf{u}_h)$ is the steady-state residual. Eq. 6.5 may be marched in time using any ODE time integrator.

6.2.2 Geometry

The finite-volume discretization implemented in BlueRidge is a cell-centered discretization, meaning that the discrete solution is stored at the centroid of the control volume, $\mathbf{r}_c$. In contrast, a node-centered discretization stores the discrete solution at the nodes of the grid. The terminology "cell-centered" does not imply that the solution at $\mathbf{r}_c$ is equal to the control volume average, $\mathbf{u}_h$, but rather states how the solution is represented and stored with respect to the grid. A cell-centered discretization is selected over a node-centered discretization to simplify the weak implementation of boundary conditions and eliminate the need to construct dual control volumes. Each control volume in the grid is treated as an arbitrary polyhedron to allow future BlueRidge developers to perform adaptive mesh refinement on general meshes [1, 2]. Since each control volume is treated as an arbitrary polyhedron, special care must be taken when computing the volume and centroid of each cell [3]. First, the volume of each control volume may be defined as follows

$$|\Omega_i| = \int_{\Omega_i} d\Omega \; . \tag{6.6}$$

To simplify the integration, Eq. 6.6 is recast into a surface integral using the Gauss divergence theorem

$$|\Omega_i| = \frac{1}{3}\oint_{\partial\Omega_i} \mathbf{r}\cdot\hat{\mathbf{n}}\; dS = \frac{1}{3}\sum_{j=1}^{N_f}\oint_{\partial\Omega_{i,j}} \mathbf{r}\cdot\hat{\mathbf{n}}\; dS \tag{6.7}$$

where N_f is the number of faces that comprise Ω_i and $\mathbf{r}$ is a position vector in Euclidean space. Next, the centroid of each control volume may be defined as

$$\mathbf{r}_c = \frac{\int_{\Omega_i}\mathbf{r}\; d\Omega}{\int_{\Omega_i} d\Omega} = \frac{1}{|\Omega_i|}\int_{\Omega_i}\mathbf{r}\; d\Omega\;. \tag{6.8}$$

In a similar manner, Eq. 6.8 is recast into a surface integral using the Gauss divergence theorem

$$\begin{aligned} r_{c,d} = \mathbf{r}_c\cdot\hat{\mathbf{e}}_d &= \frac{1}{2|\Omega_i|}\oint_{\partial\Omega_i}(\mathbf{r}\cdot\hat{\mathbf{e}}_d)^2\hat{\mathbf{e}}_d\cdot\hat{\mathbf{n}}\; dS, \quad d\in[1,2,3] \\ &= \frac{1}{2|\Omega_i|}\sum_{j=1}^{N_f}\oint_{\partial\Omega_{i,j}}(\mathbf{r}\cdot\hat{\mathbf{e}}_d)^2\hat{\mathbf{e}}_d\cdot\hat{\mathbf{n}}\; dS, \quad d\in[1,2,3] \end{aligned} \tag{6.9}$$

where d is a coordinate direction in Euclidean space, $r_{c,d}$ is a component of the centroid position, and $\hat{\mathbf{e}}_d$ is a unit vector in a given coordinate direction. All required surface integrals are performed using Gaussian quadrature.

6.2.3 Solution Reconstruction

Higher-order spatial accuracy is achieved using a k-exact least-squares reconstruction [4, 5, 6, 7]. The reconstruction is considered k-exact if it is able to exactly reconstruct polynomials of degree k. In other words, for any smooth function, $\mathbf{u}(\mathbf{x})$, the error in the reconstruction converges at

$$\left\|\mathbf{u}_i^R(\mathbf{x}) - \mathbf{u}(\mathbf{x})\right\| = O\left(h^{k+1}\right)\;. \tag{6.10}$$

The reconstruction in each control volume is represented as a linear combination of basis functions, $\phi_\nu(\cdot)$,

$$\mathbf{u}_i^R(\mathbf{x}-\mathbf{x}_i) = \sum_{\nu=0}^{N_t-1}\boldsymbol{\alpha}_\nu\phi_\nu(\mathbf{x}-\mathbf{x}_i) \tag{6.11}$$

where $\mathbf{x}_i$ is a reference location for the reconstruction, $\boldsymbol{\alpha}_\nu$ are coefficients, and N_t is the total number of basis functions. In this work, the reference location for each reconstruction is chosen as the control volume centroid, $\mathbf{r}_c$. As is common for a k-exact reconstruction, the set of basis functions is selected as the monomial basis, which in 3D may be written as

$$\phi(\mathbf{x}-\mathbf{x}_i) = \left\{(x-x_i)^a(y-y_i)^b(z-z_i)^c \;\middle|\; a+b+c\le k\right\}\;. \tag{6.12}$$

With the selection of the monomial basis, the reconstructed function can be viewed as a Taylor series expansion about the control volume centroid. Furthermore, the number of terms in the reconstruction becomes

$$N_t = \frac{1}{N_d!}\prod_{d=1}^{N_d}(k+d) \tag{6.13}$$

where k is the polynomial degree of the reconstruction and N_d is the number of spatial dimensions of the problem.

The reconstruction coefficients are computed by requiring the reconstruction to conserve the mean in Ω_i as well as in a stencil of neighboring cells, Ω_j, that is

$$\frac{1}{|\Omega_i|}\int_{\Omega_i}\mathbf{u}_i^R(\mathbf{x}-\mathbf{x}_i)\;d\Omega = \sum_{\nu=0}^{N_t-1}\boldsymbol{\alpha}_\nu\overline{\phi}_\nu(\mathbf{x}-\mathbf{x}_i) = \mathbf{u}_{h,i} \tag{6.14a}$$

$$\frac{1}{|\Omega_j|}\int_{\Omega_j}\mathbf{u}_i^R(\mathbf{x}-\mathbf{x}_i)\;d\Omega = \sum_{\nu=0}^{N_t-1}\boldsymbol{\alpha}_\nu\hat{\phi}_\nu(\mathbf{x}-\mathbf{x}_i) = \mathbf{u}_{h,j}\;. \tag{6.14b}$$

The terms $\overline{\phi}(\cdot)$ and $\hat{\phi}(\cdot)$ are referred to as control volume moments and relative control volume moments, respectively, and are given by the following

$$\overline{\phi}(\mathbf{x}-\mathbf{x}_i) = \left\{\overline{x^a y^b z^c}_i \;\middle|\; a+b+c\le k\right\} \tag{6.15a}$$

$$\hat{\phi}(\mathbf{x}-\mathbf{x}_i) = \left\{\widehat{x^a y^b z^c}_{ij} \;\middle|\; a+b+c\le k\right\} \tag{6.15b}$$

where

$$\overline{x^a y^b z^c}_i = \frac{1}{|\Omega_i|}\int_{\Omega_i}(x-x_i)^a(y-y_i)^b(z-z_i)^c\;d\Omega \tag{6.16a}$$

$$\begin{aligned}\widehat{x^a y^b z^c}_{ij} &= \frac{1}{|\Omega_j|}\int_{\Omega_j}(x-x_i)^a(y-y_i)^b(z-z_i)^c\;d\Omega \\ &= \frac{1}{|\Omega_j|}\int_{\Omega_j}[(x-x_j)+(x_j-x_i)]^a[(y-y_j)+(y_j-y_i)]^b[(z-z_j)+(z_j-z_i)]^c\;d\Omega \\ &= \sum_{l=0}^{a}\sum_{m=0}^{b}\sum_{n=0}^{c}\binom{a}{l}\binom{b}{m}\binom{c}{n}(x_j-x_i)^l(y_j-y_i)^m(z_j-z_i)^n\overline{x^{a-l}y^{b-m}z^{c-n}}_j\;.\end{aligned} \tag{6.16b}$$

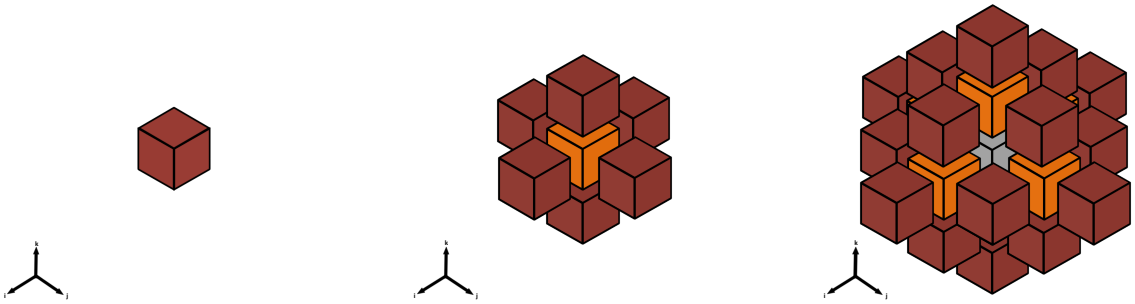
The relative control volume moments, $\widehat{x^a y^b z^c}_{ij}$, are reformulated by adding and subtracting components of $\mathbf{r}_j$. In doing so, the number of required integrals can be reduced. Now, integrals only need to be performed for control volume moments in each Ω_i, and relative control volume moments can be computed as needed. Since BlueRidge operates on arbitrary

polyhedra, the volume integrals defined in Eq. 6.16a can be rather cumbersome to perform. In practice, the Gauss divergence theorem is used to convert the volume integral into a surface integral so that control volume moments may be reformulated as follows

$$\overline{x^a y^b z^c}_i = \frac{1}{|\Omega_i|} \frac{1}{(a+1)} \oint_{\partial\Omega_i} (x - x_i)^{a+1} (y - y_i)^b (z - z_i)^c \; n_x \; dS \tag{6.17}$$

where n_x is the component of the surface unit normal vector in the x-direction.

Neighboring control volumes to Ω_i are added to the reconstruction stencil by growing the stencil outward using face connectivity information. The first control volume added to the stencil is Ω_i. In the second generation, all face neighbors to Ω_i are added. Likewise, in the third generation, all face neighbors of second generation control volumes not already present in the stencil are then added. The algorithm continues until the stencil contains a predefined number of control volumes. We require at least 50% more control volumes in the stencil than what is dictated by the reconstruction degree [7]. In an attempt to offset any accuracy loss that may be incurred by one-sided stencils, we require boundary cells to have at least as many stencil control volumes as interior cells. An illustration of the stencil construction process is given in Figure 6.1. While Figure 6.1 illustrates the algorithm for a group of logically Cartesian cells, the behavior of the algorithm is identical for a group of unstructured cells.



(a) 1^{st} Generation (b) 2^{nd} Generation (c) 3^{rd} Generation

Figure 6.1: Stencil Construction

Using Eq. 6.14, a linear system for the reconstruction coefficients, $\boldsymbol{\alpha}$, may now be constructed as follows

$$\left[\begin{array}{c|cccc} 1 & \overline{x}_i & \overline{y}_i & \overline{z}_i & \cdots \\ \hline w_{i1} & w_{i1}\widehat{x}_{i1} & w_{i1}\widehat{y}_{i1} & w_{i1}\widehat{z}_{i1} & \cdots \\ w_{i2} & w_{i2}\widehat{x}_{i2} & w_{i2}\widehat{y}_{i2} & w_{i2}\widehat{z}_{i2} & \cdots \\ \vdots & \vdots & \vdots & \vdots & \ddots \\ w_{iN_s} & w_{iN_s}\widehat{x}_{iN_s} & w_{iN_s}\widehat{y}_{iN_s} & w_{iN_s}\widehat{z}_{iN_s} & \cdots \end{array}\right] \left(\begin{array}{c} u \\ \frac{\partial u}{\partial x} \\ \frac{\partial u}{\partial y} \\ \frac{\partial u}{\partial z} \\ \vdots \end{array}\right)_i = \left(\begin{array}{c} u_{h,i} \\ \hline w_{i1}u_{h,1} \\ w_{i2}u_{h,2} \\ \vdots \\ w_{iN_s}u_{h,N_s} \end{array}\right) \tag{6.18}$$

where N_s is the number of control volumes in the stencil. The reconstruction coefficients have been written as derivatives of u to further illustrate the connection to Taylor series expansions. All rows above the horizontal line signify constraint rows. For simplicity, the linear system has only been written for one component of $\mathbf{u}_i^R$. A weight, w_{ij}, is premultiplied to emphasize geometrically close data and is defined as

$$w_{ij} = \frac{1}{|\mathbf{r}_{c,i} - \mathbf{r}_{c,j}|^\gamma} \tag{6.19}$$

where γ is a non-negative constant. While $\gamma = 0.0$ can typically lead to a more robust solver, it can produce inaccurate gradients in regions of high curvature [8]. For most cases, we use $\gamma = 1.0$ [9]. In order to conserve the mean in Ω_i exactly, the mean constraint row in Eq. 6.18 is eliminated analytically resulting in the following linear system

$$\begin{bmatrix} w_{i1}(\widehat{x}_{i1} - \overline{x}_i) & w_{i1}(\widehat{y}_{i1} - \overline{y}_i) & w_{i1}(\widehat{z}_{i1} - \overline{z}_i) & \cdots \\ w_{i2}(\widehat{x}_{i2} - \overline{x}_i) & w_{i2}(\widehat{y}_{i2} - \overline{y}_i) & w_{i2}(\widehat{z}_{i2} - \overline{z}_i) & \cdots \\ \vdots & \vdots & \vdots & \ddots \\ w_{iN_s}(\widehat{x}_{iN_s} - \overline{x}_i) & w_{iN_s}(\widehat{y}_{iN_s} - \overline{y}_i) & w_{iN_s}(\widehat{z}_{iN_s} - \overline{z}_i) & \cdots \end{bmatrix} \begin{pmatrix} \frac{\partial u}{\partial x} \\ \frac{\partial u}{\partial y} \\ \frac{\partial u}{\partial z} \\ \vdots \end{pmatrix}_i = \begin{pmatrix} w_{i1}(u_{h,1} - u_{h,i}) \\ w_{i2}(u_{h,2} - u_{h,i}) \\ \vdots \\ w_{iN_s}(u_{h,N_s} - u_{h,i}) \end{pmatrix} . \tag{6.20}$$

Eq. 6.20 is solved in a least-squares sense using a truncated singular value decomposition [10] to safeguard against ill-conditioning and numerical rank deficiency. The pseudo-inverse of the left-hand side matrix is computed and stored as a pre-processing step. Consequently, the determination of the reconstruction coefficients becomes a simple matrix-vector multiplication at each iteration of the solver. Once Eq. 6.20 has been solved, the coefficient $u|_i$ is determined via back substitution. For more details regarding the numerical solution of Eq. 6.20, the reader is referred to Appendix C.

For problems with large gradients, it is often necessary to limit the reconstruction to prevent the introduction of new extrema. Overshoots can be avoided by reducing the higher-order contribution to the reconstruction as follows

$$\mathbf{u}_i^R(\mathbf{x} - \mathbf{x}_i, \Psi_i) = \boldsymbol{\alpha}_0 + \Psi_i \sum_{\nu=1}^{N_t - 1} \boldsymbol{\alpha}_\nu \phi_\nu(\mathbf{x} - \mathbf{x}_i), \quad \Psi_i \in [0, 1] \tag{6.21}$$

where Ψ_i is a solution limiter defined for Ω_i. Solution limiters are commonly computed by measuring the variation in the unlimited reconstruction between adjacent control volumes. In smooth regions, the reconstruction defaults to its original unlimited state, i.e.,$\Psi_i = 1$. The limiters implemented in BlueRidge include that of Barth and Jespersen [11], Venkatakrishnan [12], and Michalak [13].

BlueRidge also provides the capability of enforcing boundary conditions on the reconstructed solution. Consider a Dirichlet boundary condition given by the function, $f_1(\cdot)$, at the quadrature point, $\mathbf{x}_g$. The reconstructed function evaluated at $\mathbf{x}_g$ is required to satisfy the boundary condition, that is

$$\mathbf{f}_1(\mathbf{x}_g) = \mathbf{u}_i^R(\mathbf{x}_g - \mathbf{x}_i) \ . \tag{6.22}$$

By inserting Eq. 6.22 into Eq. 6.20, a new linear system for the reconstruction coefficients may be written as

$$\left[\begin{array}{cccc} (x_g - x_i) - \overline{x}_i & (y_g - y_i) - \overline{y}_i & (z_g - z_i) - \overline{z}_i & \cdots \\ \hline w_{i1}(\widehat{x}_{i1} - \overline{x}_i) & w_{i1}(\widehat{y}_{i1} - \overline{y}_i) & w_{i1}(\widehat{z}_{i1} - \overline{z}_i) & \cdots \\ w_{i2}(\widehat{x}_{i2} - \overline{x}_i) & w_{i2}(\widehat{y}_{i2} - \overline{y}_i) & w_{i2}(\widehat{z}_{i2} - \overline{z}_i) & \cdots \\ \vdots & \vdots & \vdots & \ddots \\ w_{iN}(\widehat{x}_{iN} - \overline{x}_i) & w_{iN}(\widehat{y}_{iN} - \overline{y}_i) & w_{iN}(\widehat{z}_{iN} - \overline{z}_i) & \cdots \end{array}\right] \left(\begin{array}{c} \frac{\partial u}{\partial x} \\ \frac{\partial u}{\partial y} \\ \frac{\partial u}{\partial z} \\ \vdots \end{array}\right)_i = \left(\begin{array}{c} f_1(\mathbf{x}_g) - u_{h,i} \\ \hline w_{i1}(u_{h,1} - u_{h,i}) \\ w_{i2}(u_{h,2} - u_{h,i}) \\ \vdots \\ w_{iN}(u_{h,N} - u_{h,i}) \end{array}\right). \tag{6.23}$$

A row is added to the linear system for each boundary constraint to be enforced. Neumann boundary conditions are enforced in a similar manner [6]. For boundary conditions which couple different variables, a combined linear system for all variables is constructed and solved. For more details regarding coupled boundary conditions, the reader is referred to Ref. [14]. Depending upon the boundary face geometry, boundary cell connectivity, and the polynomial degree of the reconstructed solution, the number of boundary constraints can become large and may even exceed the number of terms in the reconstruction. Therefore, boundary constraints are instead enforced in an approximate sense using the method of weights [15]

$$\begin{bmatrix} w_c\mathbf{I}_c & 0 \\ 0 & \mathbf{I}_a \end{bmatrix} \begin{bmatrix} C \\ A \end{bmatrix} \boldsymbol{\alpha} = \begin{bmatrix} w_c\mathbf{I}_c & 0 \\ 0 & \mathbf{I}_a \end{bmatrix} \begin{bmatrix} \mathbf{d} \\ \mathbf{b} \end{bmatrix} \tag{6.24}$$

where w_c is a large, positive weighting factor, $C \in \mathbb{R}^{N_c \times n}$ is the constraint matrix, $\mathbf{d} \in \mathbb{R}^{N_c}$ are the evaluations of each constraint, $A\boldsymbol{\alpha} = \mathbf{b}$ is the unconstrained reconstruction, and $\mathbf{I}_c$ and $\mathbf{I}_a$ are appropriately sized identity matrices. Eq. 6.24 is solved in a similar manner to Eq. 6.20. The weight, w_c, is used to emphasize boundary constraint rows more than the original unconstrained problem. To ensure boundary constraints are adequately satisfied, the error in the constraints may be monitored, and the weight, w_c, may be adjusted appropriately.

6.3 Code Development

6.3.1 Modern Fortran Usage

BlueRidge is written in modern Fortran using the 03/08 standards [16, 17]. To meet the design goals of modularity and flexibility, we make considerable use of modern programming constructs. The Fortran 03/08 standards introduced many modern programming features into the language, such as abstraction, polymorphism, and object-oriented programming. These standards also provided increased interoperability with C/C++ allowing programmers to interface with external programs or libraries written in these languages. Possibly the most important modern programming construct utilized in BlueRidge is a module. With modules, programmers may organize code into sections of common functionality in a manner that is easily accessible by other parts of the code. Similar to C header files, modules provide explicit

interfaces for procedures allowing the compiler to check the consistency between the expected arguments and the actual arguments. In BlueRidge, all procedures, both subroutines and functions, are placed into modules so that each procedure has an explicit interface. Explicit interfaces are extremely useful for locating usage mistakes as the compiler will typically throw an error before the linking stage.

Additionally, modules foster encapsulation, a programming technique used to restrict access to specific variables or routines. Only variables and procedures declared as `public` may be used elsewhere in the code. When module variables or procedures are used in other parts of the code (e.g., other modules, subroutines, or programs), they are accessed with the statement:

```
use <module_name>, only : <module_procedures>, <module_variables>
```

While the contents of an entire module can technically be used, this practice is highly discouraged. If variables or procedures are not specifically imported, the following problems can potentially arise:

1. A namespace conflict can occur if the imported module creates a variable with the same name as a local variable, or if multiple modules have variables with the same name.

2. It may be difficult for a future programmer to determine the file from which a variable or procedure originated.

Another modern Fortran feature which fosters both modularity and encapsulation is a derived type. Derived types can hold multiple variables of different data types. Furthermore, procedures can be bound to a derived type and used to modify data that is internal or external to the derived type. Several instances of a given derived type may be created and passed in and out of routines as needed. To ensure that all memory is freed, derived types may be bound with `final` routines which automatically deallocate memory when an object goes out of scope. In BlueRidge, however, derived types are bound with `destructor` routines, or routines that must be manually called to deallocate the object, in favor of `final` routines for instances when a derived type needs to be deallocated before it goes out of scope.

Derived types become more powerful when coupled with the concepts of polymorphism and abstraction. A polymorphic variable is an entity whose data type is dynamic at run time. For instance, in the context of derived types, a polymorphic variable may be declared as a common parent type then, at run time, be instantiated as a given child type which extends the functionality of the parent type. In BlueRidge, polymorphic derived types are primarily used in conjunction with abstract derived types, or derived types which cannot be instantiated and whose variables and procedures are only fully defined once extended by another derived type. It is common for type-bound procedures in an abstract type to have

the `deferred` attribute, meaning that their implementation is to be defined in a sub-derived type, or subclass. An abstract derived type may extend another abstract derived type; however, the levels of abstraction must terminate in a non-abstract derived type in order for the data structure to be instantiated. A code example which utilizes abstract and polymorphic types to describe geometric shapes is given in Appendix D. The superclass, `shape_t`, contains variables that are common to all shapes such as the number of sides defining the shape and the lengths of each side. The superclass also contains deferred procedures whose implementations are unique to each shape, such as calculating area (`compute_area`). The `triangle_t` subclass is an intermediate abstract derived type used to further describe a three-sided shape. The final non-abstract types, which extend `triangle_t`, are the `isosceles_triangle_t` and `equilateral_triangle_t` derived types. As non-abstract types, these derived types must create an implementation for all inherited deferred procedures that have not been implemented by parent classes. The primary benefit of polymorphic derived types is that the main part of the code does not need to know the subclass of the object. For instance, by abstracting the area calculation, the code may treat all shapes in the same way. For any given shape, a call to the `compute_area` routine performs the calculation associated with the instantiated subclass and returns the proper area for the shape.

In order to ensure the longevity of a code base, the code must be sufficiently flexible. To this end, BlueRidge employs allocatable arrays and pointers. Allocatable arrays are arrays whose size may be determined at run time. These types of arrays are useful for the storage of data that depends on problem size, such as the grid and solution variables. A variable pointer allows a programmer to access the memory address of a given variable. Variable pointers are helpful when data needs to be accessed in multiple locations and the programmer cannot afford to waste memory by creating a copy. Furthermore, if the data is actually copied, it may also become multi-valued if it is changed in one location but not another. In BlueRidge, variable pointers are used during the creation of reconstruction stencils to point to the memory location of the solution in each stencil control volume. In this way, the reconstruction can access solution information for stencil control volumes without having to pass the solution vector or check whether solution data has been updated. Similar to variable pointers, procedure pointers may be used to reference a particular procedure. Procedure pointers allow the programmer to define a procedure with a common name and interface but different implementations. At compile time or even run time, the programmer may point the procedure pointer to the desired implementation, and the functionality of the procedure may be accessed through the common procedure name. In BlueRidge, procedure pointers are seldom used in favor of polymorphic type-bound procedures.

6.3.2 Stylistic Practices

For any large software project, it is essential that the code be clear and easy to understand. Since BlueRidge is written in an academic setting by programmers with varying levels of experience, code clarity is especially important. Code clarity can be enforced and maintained

through the codification of stylistic practices. With a common coding style, all parts of the code base look similar despite being written by different programmers. This allows future developers to quickly comprehend how the code works without getting bogged down by multiple coding styles. In this section, the stylistic guidelines used for BlueRidge development are listed along with justifications for their use.

Since the introduction of the Fortran 95 standard, fixed form source files have been marked as obsolete. Therefore, all BlueRidge source files are written in free form. Free form source files facilitate code reading and comprehension, especially for nested structures and decision trees. However, a strict limit of 80 characters per line is still imposed for both comments and source code. While this may seem to be an unnecessary restriction, we have found it is much easier to read short, compact lines of code. With short lines of code, constant resizing of the editor window can be avoided in addition to any confusion caused by word wrapping or lines that run off the side of the screen.

Tab characters are not allowed in favor of two space characters. The use of tab characters can often lead to strange formatting in editors which may not use two spaces to represent the tab character. The two space rule is used, as opposed to say four spaces, for readability as it still provides order and organization of code structures without making lines too long. Trailing white space is avoided as it can muddle code "diffs", for instance, when the only change to the code is the number of spaces at the end of the line.

In many programs, it is common to perform similar calculations on multiple lines. Alignment of like terms is highly encouraged as it can greatly assist with reading the code. Code alignment is particularly advantageous in variable declaration sections where the blocked structure of the code lends itself naturally to alignment. In our experience, code alignment can even help locate bugs stemming from an overzealous copy-paste.

Stylistic practices are also established to help avoid common programming pitfalls. For modules, the default visibility for all member variables and procedures is set to `private`. Variables and procedures can be explicitly set to `public` if they must be accessed by another part of the code. By limiting the number of public variables and procedures, any side-effects which may occur as a result of unintended calls changing the internal state of the module can be avoided. In addition, to force explicit typing, all modules are required to begin with `implicit none`. It is our opinion that implicit typing has no practical benefit in modern programming and only leads to confusion in the event that a programmer accidentally forgets to declare a variable. Explicit typing forces developers to intimately know the type of each variable, facilitates code comprehension, and enables safe modification of the code.

Since BlueRidge is written in an academic environment where the turn-over of developers is relatively frequent, it is important for future developers to be able to understand the code base and begin development as quickly as possible. To foster code comprehension, a header section is provided for each procedure describing its function as well as the inputs and outputs. Furthermore, the intent of each input and output is explicitly set when each variable is declared. To specify the intent of a variable, it is declared with the attribute

`intent(in)` if purely an input, the attribute `intent(out)` if purely an output, or the attribute `intent(inout)` if both an input and an output. By specifying the intent of inputs and outputs, future developers can better understand the flow of data through a procedure, and compilers can provide additional optimization of the code. BlueRidge developers are also encouraged to write self-commenting code, meaning the purpose of the code is self-evident from how variables or procedures are named and used. Likewise, developers are discouraged from using pointless comments such as:

```
! Set the number of points equal to 2
N_points = 2
```

In a large code base, it can be difficult for new developers to locate specific portions of the code. To facilitate navigation of the code base, several common naming conventions are adopted. All derived types are appended with "`_t`" to indicate they are a type. For example, a derived type for triangles may be given the name `triangle_t`. Similarly, each derived type is placed in a module named `<derived_type_name>_derived_type`. Typically, only one derived type is defined in each module, however in rare circumstances, it may be sensible to include multiple closely related derived types in the same module. If procedure interfaces are needed, the interface is given the same name as the procedure but appended with "`_i`". Lastly, the implementation of a deferred procedure in a child class is appended with "`_obj`" to indicate that the procedure is part of an object. For examples of the stylistic practices listed in this section, refer to Appendix D.

6.3.3 Version Control

Version control is a crucial part of any code development project. A version control system saves snapshots of the code base whenever changes, or commits, are made. Version control allows a code developer to monitor the evolution of the code, inspect routines at a specific point in time, and even examine previous versions of the entire code base. Tracking the code in this manner can be especially helpful when debugging. For instance, if a previously working feature is found to no longer work, the programmer can go through the code history, locate the offending commit, and re-enable the feature. Bugs can typically be identified quickly if developers commit early and often. If each commit only changes a small portion of the code, bugs should be more apparent and easier to fix.

For the development of BlueRidge, we use Git, a distributed version control system [18]. In a distributed version control paradigm, a primary repository is stored on a server, and each developer clones a copy of the repository to their machine. Developers may update the code in their local repository, make commits, and push to the main repository when convenient. This type of version control is quite practical as it allows developers to continue working even when not connected to the internet. To facilitate access to the code base, BlueRidge is stored in a private repository on GitHub rather than on a local server.

Version control systems also facilitate code development among a group of programmers. Since BlueRidge is developed by a relatively small number of programmers, there is no need for a complicated workflow with trusted branches or committers. However, branching is still used extensively so that the master branch remains a tested, working version of the code and is relatively free of bugs and poorly written code. In a larger group of developers, each developer typically works on their own branch. Once a branch has been sufficiently tested, it is then merged into the master branch. This code development model works well as long as branches do not drastically diverge. When branches diverge, programmers can introduce bugs and break code features in the process of resolving merge conflicts. To prevent branch divergence, it is helpful to regularly pull changes from the master branch into working branches. Difficult merges can also be prevented by frequent communication between developers.

6.3.4 Portability, Performance, and Maintainability

To improve the portability and performance of the code base, BlueRidge makes considerable use of external libraries and tools. The BLAS library [19] provides routines for basic linear algebra operations such as matrix-vector multiplies, norms, and dot products. Building off of the BLAS library, the LAPACK library [20] provides routines for more complicated linear algebra tasks such as matrix decompositions and the computation of eigenvalues. While many of the linear algebra routines included in BLAS and LAPACK could be written by hand, it would be senseless to ignore the amount of development and optimization that have gone into making these libraries fast and efficient. Input/output for BlueRidge is primarily managed by the CGNS library [21, 22, 23, 24]. The CGNS library provides a flexible input/output format for organizing, storing, and processing data from scientific computing software. CGNS files are written in a binary format to conserve disk space and are cross-platform compatible. CGNS is an industry standard and is supported by most grid generation and post-processing tools. CGNS fosters collaboration between research groups since different codes can be used to run the same test case and the need for complicated conversion scripts can be eliminated. To ensure a stable build, BlueRidge is compiled with flags that check for out of bound array accesses and other common memory errors. Valgrind [25] is used to locate any remaining memory errors not caught by the compiler. Valgrind is particularly useful for identifying memory leaks.

Previous efforts by the code development group utilized GNU Autotools for the build system. However, current developers do not have much experience with Autotools and find its archaic commands confusing and difficult to use. We would frequently break the build just by moving a file or changing the dependency structure. In an attempt to circumvent the difficulties of Autotools, we would typically make minor modifications to existing `Makefile.in` files and hope the build system remained functioning. This approach is not adequate for creating and maintaining a build system for any code base, much less one developed by graduate students who are not experts in build systems. To simplify and streamline the build system, we use

CMake to build BlueRidge [26, 27]. While CMake can have its quirks and finding up-to-date documentation can be difficult, it has removed much of the burden associated with compiling the code. As with nearly all build systems, fully understanding CMake can be challenging; but, creating a basic build system that is robust and works well is relatively straightforward. Once the general structure of the build system is established, adding new files, moving files, and changing dependencies is simple. In the previous Autotools build system, developers were required to know the dependencies of each source file and correctly place them in the build order. With CMake, the build system automatically parses source files to determine dependencies and adds them to the build without any developer intervention. In addition, automatic parsing can accelerate parallel builds as CMake is able to better determine where parallelization may be used. To further expedite the build process, we have created macros for adding source files to libraries, adding source file dependencies to executables, and even for adding unit tests. Whenever a new source file is created, it is included in the build by simply adding its name to the `CMakeLists.txt` file in its directory. With regard to compiler flags, CMake gives programmers a great deal of flexibility. By setting the `CMAKE_BUILD_TYPE` variable to `RELEASE` or `DEBUG`, the programmer can easily switch between the appropriate debug and release flags. Furthermore, CMake can remember build configurations; in a given build directory, the code will be built in the same manner until the build settings are explicitly updated. Finally, CMake can automatically locate external libraries by querying the developer's workstation for library paths.

6.4 Code Structure

Modularity and flexibility are the primary design objectives for BlueRidge development. To achieve these goals, BlueRidge is composed of several classes and derived types which execute high-level, complex tasks, such as defining the equation set and marching the solution in time. The primary classes and derived types utilized in BlueRidge are discussed in detail in the following sections.

6.4.1 Physics Class (`physics_t`)

Equations sets are implemented in BlueRidge via the physics class. The superclass, `physics_t`, houses definitions for the state variables, $\mathbf{u}$, working variables, $\mathbf{q}$, analytic flux, $\mathbb{F}(\cdot)$, and source term, $\mathbf{s}(\cdot)$. Deferred procedures implemented by `physics_t` subclasses include the following:

- Conversion routines between state variables and working variables
- Extraction and fill procedures for individual state and working variables
- Implementation of the analytic flux and Jacobian

- Implementation of flux functions and Jacobians
- Implementation of output functionals and Jacobians
- Implementation of a physics-based time step

The general hierarchy of the physics class is given in Figure 6.2. BlueRidge currently supports equation sets for modeling fluid motion as well as equation sets for model problems. To this end, the physics class contains two intermediate abstract classes: the `fluid_mechanics_t` subclass and the `model_problem_t` subclass. These subclasses hold variables and procedures common to all fluid models and model problems, respectively. In these subclasses, deferred procedures are defined for each boundary condition to be implemented. For a complete list and description of the boundary conditions implemented in the `fluid_mechanics_t` class and the `model_problem_t` class, refer to Appendix E and Appendix F, respectively.

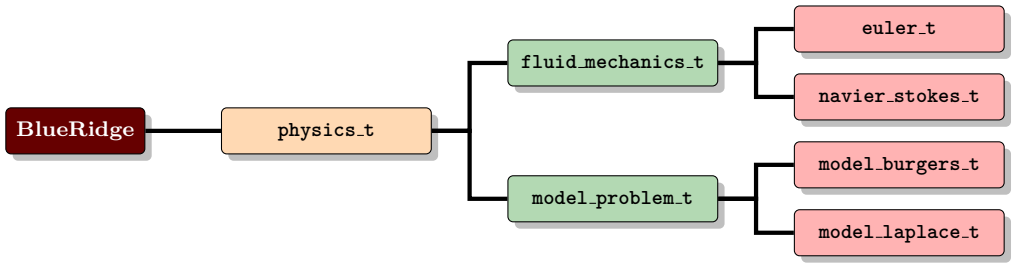


Figure 6.2: Hierarchy of Physics Class

In this work, it is assumed that the analytic flux for each fluid model and model problem may be decomposed as follows

$$\mathbb{F}(\mathbf{u}, \nabla\mathbf{u}) = \mathbb{F}_i(\mathbf{u}) - \mathbb{F}_v(\mathbf{u}, \nabla\mathbf{u}) \tag{6.25}$$

where $\mathbb{F}_i(\mathbf{u})$ and $\mathbb{F}_v(\mathbf{u}, \nabla\mathbf{u})$ are the inviscid and viscous contributions to the flux, respectively. All equation sets are coded in a nondimensional form to improve the numerical scaling of the solution procedure. The nondimensionalization used in BlueRidge is similar to that of Ref. [28]. For more details regarding the nondimensionalization of each subclass, refer to Appendix G. Due to the flexible design of the physics class, other physical phenomena, such as solid mechanics or electrostatics, can easily be modeled by future developers.

6.4.1.1 Fluid Mechanics: Euler Equations (`euler_t`)

The Euler equations are used to model the motion of an inviscid fluid. The state variables, inviscid flux, viscous flux, and source term are given by the following

$$\mathbf{u} = \begin{bmatrix} \rho \\ \rho\mathbf{v} \\ \rho e_t \end{bmatrix}, \quad \mathbb{F}_i(\mathbf{u}) = \begin{bmatrix} \rho\mathbf{v}^T \\ \rho\mathbf{v}\mathbf{v}^T + p\mathbf{I} \\ \rho\mathbf{v}^T h_t \end{bmatrix}, \quad \mathbb{F}_v(\mathbf{u}, \nabla\mathbf{u}) = \begin{bmatrix} 0 \\ \mathbf{0} \\ 0 \end{bmatrix}, \quad \mathbf{s}(\mathbf{u}, \nabla\mathbf{u}) = \begin{bmatrix} 0 \\ \mathbf{0} \\ 0 \end{bmatrix} \tag{6.26}$$

where ρ is the fluid density, $\mathbf{v}$ is the fluid velocity, p is the static pressure, e_t is the total energy per unit mass, and h_t is the total enthalpy per unit mass. The system of equations is closed using the equation of state for a perfect gas such that

$$e_t = \frac{p}{\rho(\gamma-1)} + \frac{\mathbf{v}\cdot\mathbf{v}}{2}, \quad h_t = \frac{\gamma p}{\rho(\gamma-1)} + \frac{\mathbf{v}\cdot\mathbf{v}}{2} \tag{6.27}$$

where γ is the ratio of specific heats for the fluid. The analytic flux is discretized at each control volume interface using a flux function. The flux functions currently available in BlueRidge include Roe's flux difference splitting scheme [29], Van Leer's flux vector splitting scheme [30], AUSM [31], AUSM+ [32], and the Steger-Warming flux vector splitting scheme [33, 34]. The Jacobians for each flux function are obtained via hand differentiation and are checked using complex step finite-differences [35, 36]. Following Ref. [37], a physics-based time step is computed in each control volume as

$$\Delta t_i = \mathrm{CFL}\frac{|\Omega_i|}{\left(\sum_{d=1}^{N_d}\hat{\Lambda}_c^d\right)_i} \tag{6.28}$$

where CFL is the Courant-Friedrichs-Lewy number, and $\hat{\Lambda}_c^d$ are convective spectral radii given by

$$\hat{\Lambda}_c^d = (|\mathbf{v}\cdot\hat{\mathbf{e}}_d| + a)\Delta\hat{S}^d, \tag{6.29}$$

where a is the local speed of sound. The terms $\Delta\hat{S}^d$ are control volume projections onto the spatial plane opposing the vector $\hat{\mathbf{e}}_d$ and are given by

$$\Delta\hat{S}^d = \frac{1}{2}\sum_{j=1}^{N_f}\oint_{\partial\Omega_{i,j}} |\hat{\mathbf{n}}\cdot\hat{\mathbf{e}}_d|\; dS\;. \tag{6.30}$$

6.4.1.2 Fluid Mechanics: Navier-Stokes Equations (`navier_stokes_t`)

The Navier-Stokes equations are used to model the motion of a viscous fluid. In BlueRidge, the effects of radiation and body forces have been neglected. The state variables, inviscid flux, viscous flux, and source term are given by the following

$$\mathbf{u} = \begin{bmatrix}\rho\\ \rho\mathbf{v}\\ \rho e_t\end{bmatrix},\;\; \mathbb{F}_i(\mathbf{u}) = \begin{bmatrix}\rho\mathbf{v}^T\\ \rho\mathbf{v}\mathbf{v}^T + p\mathbf{I}\\ \rho\mathbf{v}^T h_t\end{bmatrix},\;\; \mathbb{F}_v(\mathbf{u},\nabla\mathbf{u}) = \begin{bmatrix}0\\ \boldsymbol{\tau}\\ \boldsymbol{\tau}\mathbf{v} + k\nabla T\end{bmatrix},\;\; \mathbf{s}(\mathbf{u},\nabla\mathbf{u}) = \begin{bmatrix}0\\ \mathbf{0}\\ 0\end{bmatrix} \tag{6.31}$$

where $\boldsymbol{\tau}$ is the viscous stress tensor, k is the thermal conductivity of the fluid, and T is the static temperature. The fluid is assumed to behave as a Newtonian fluid such that

$$\boldsymbol{\tau} = \mu\Big[\nabla\mathbf{v} + (\nabla\mathbf{v})^T\Big] - \frac{2}{3}\mu(\nabla\cdot\mathbf{v})\mathbf{I} \tag{6.32}$$

where μ is the dynamic viscosity. The dynamic viscosity may either be an input to the code or may be modeled using Sutherland's law. The system is closed using the equation of state

for a perfect gas. The inviscid flux is discretized using the flux functions given for the Euler equations. The viscous flux is discretized as follows

$$\mathbb{F}_v(\mathbf{u}, \nabla\mathbf{u}) \approx \mathbb{F}_v(\mathbf{u}_F, \nabla\mathbf{u}_F) \tag{6.33}$$

where the interface state, $\mathbf{u}_F$, and interface gradient, $\nabla\mathbf{u}_F$, are given by

$$\mathbf{u}_F = \frac{1}{2}\Big(\mathbf{u}^+ + \mathbf{u}^-\Big) \tag{6.34a}$$

$$\nabla\mathbf{u}_F = \frac{1}{2}\Big(\nabla\mathbf{u}^+ + \nabla\mathbf{u}^-\Big) + \frac{\alpha}{|\mathbf{r}_{ij}\cdot\hat{\mathbf{n}}|}\Big(\mathbf{u}^+ - \mathbf{u}^-\Big)\hat{\mathbf{n}}\ . \tag{6.34b}$$

In Eq. 6.34b, $\mathbf{r}_{ij}$ is the vector extending from $\mathbf{r}_{c,i}$ to $\mathbf{r}_{c,j}$, $\hat{\mathbf{n}}$ is the interface unit normal vector, and α is a positive constant. The jump term has been added to the interface gradient for stabilization. For more details regarding the viscous flux, refer to Refs. [38, 39]. Similar to the Euler equations, a physics-based time step is computed in each control volume as

$$\Delta t_i = \mathrm{CFL}\frac{|\Omega_i|}{\Big(\sum_{d=1}^{N_d}\hat{\Lambda}_c^d\Big)_i + C\Big(\sum_{d=1}^{N_d}\hat{\Lambda}_v^d\Big)_i} \tag{6.35}$$

where $1 \leq C \leq 4$. The convective spectral radii are given by Eq. 6.29, and the viscous spectral radii are computed as follows

$$\hat{\Lambda}_v^d = \max\left(\frac{4}{3\rho}, \frac{\gamma}{\rho}\right)\frac{\mu}{Pr}\frac{\Big(\Delta S^d\Big)^2}{|\Omega_i|} \tag{6.36}$$

where Pr is the laminar Prandtl number. Control volume projections are computed according to Eq. 6.30.

6.4.1.3 Model Problem: Burgers' Equation (`model_burgers_t`)

Burgers' equation [40] is a model problem that is often used to simulate the physical phenomenon embedded in the Navier-Stokes equations. The state variables, inviscid flux, viscous flux, and source term are given by the following

$$\mathbf{u} = \Big[u\Big],\ \ \mathbb{F}_i(\mathbf{u}) = \Big[\tfrac{u^2}{2}\mathbf{1}\Big],\ \ \mathbb{F}_v(\mathbf{u}, \nabla\mathbf{u}) = \Big[\nu\nabla u\Big],\ \ \mathbf{s}(\mathbf{u}, \nabla\mathbf{u}) = \Big[0\Big] \tag{6.37}$$

where u is a velocity and ν is a viscosity. The inviscid contribution to the flux is discretized using an average of the left and right analytic fluxes. The viscous flux is discretized in the same manner as for the Navier-Stokes equations. A physics-based time step is computed using a similar expression to Eq. 6.35.

6.4.1.4 Model Problem: Laplace's Equation (`model_laplace_t`)

Laplace's equation is a model problem that is often used to simulate diffusion processes, such as heat conduction. The state variables, inviscid flux, viscous flux, and source term are given by the following

$$\mathbf{u} = \Big[u\Big], \;\; \mathbb{F}_i(\mathbf{u}) = \Big[0\Big], \;\; \mathbb{F}_v(\mathbf{u}, \nabla\mathbf{u}) = \Big[\nu\nabla u\Big], \;\; \mathbf{s}(\mathbf{u}, \nabla\mathbf{u}) = \Big[0\Big] \tag{6.38}$$

where ν is a positive constant governing the diffusion rate of u. The viscous flux is discretized in the same manner as for the Navier-Stokes equations. Likewise, a physics-based time step can be computed using a similar expression to Eq. 6.35 with the exception that all convective terms are set to zero.

6.4.2 Time Integration Class (`time_integrate_t`)

The time integration class is used to march a given equation set in time. The superclass `time_integrate_t` stores residual norms and monitors convergence for steady-state problems. Each subclass provides a different temporal discretization of Eq. 6.5. BlueRidge implements both implicit and explicit time integration schemes. All time integrators permit order ramping to overcome initial transients and to stabilize the solution procedure. A hierarchy of the time integration class may be found in Figure 6.3.

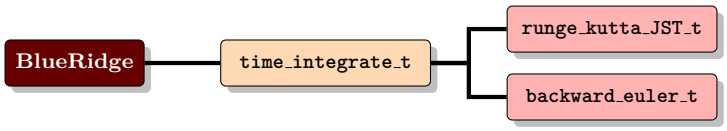


Figure 6.3: Hierarchy of Time Integration Class

6.4.2.1 Runge-Kutta JST (`runge_kutta_JST_t`)

The `runge_kutta_JST_t` subclass implements the time integration scheme of Jameson et al. [41]. As an explicit time integration scheme, the temporal update to a given control volume only depends on the previous solution and residual in that control volume. At each time step, m subiterations or stages are computed to advance the solution to the next time step. An

example of a 4-stage Runge-Kutta integrator is given as follows

$$\begin{aligned}
\mathbf{u}_{h,i}^{(1)} &= \mathbf{u}_{h,i}^{n} - \alpha_1 \frac{\Delta t}{|\Omega_i|}\mathbf{R}\Big(\mathbf{u}_{h,i}^{n}\Big) \\
\mathbf{u}_{h,i}^{(2)} &= \mathbf{u}_{h,i}^{n} - \alpha_2 \frac{\Delta t}{|\Omega_i|}\mathbf{R}\Big(\mathbf{u}_{h,i}^{(1)}\Big) \\
\mathbf{u}_{h,i}^{(3)} &= \mathbf{u}_{h,i}^{n} - \alpha_3 \frac{\Delta t}{|\Omega_i|}\mathbf{R}\Big(\mathbf{u}_{h,i}^{(2)}\Big) \\
\mathbf{u}_{h,i}^{n+1} &= \mathbf{u}_{h,i}^{n} - \alpha_4 \frac{\Delta t}{|\Omega_i|}\mathbf{R}\Big(\mathbf{u}_{h,i}^{(3)}\Big)
\end{aligned} \tag{6.39}$$

where α_m are positive constants designed to improve the accuracy and stability of the integrator. In comparison to classical Runge-Kutta methods, this time integrator is considered a low storage scheme since each stage only depends on the residual at the previous stage. The stability region, and therefore the maximum CFL number, is set by the the number of stages and the stage coefficients. For the 4-stage scheme given, the maximum CFL number is approximately $2\sqrt{2}$. Convergence to steady-state is accelerated using local time stepping and implicit residual smoothing [42, 43]. BlueRidge provides both upwind and central residual smoothing algorithms.

6.4.2.2 Backward Euler (`backward_euler_t`)

The `backward_euler_t` subclass implements a time integration scheme similar to that of Beam and Warming [44]. First, Eq. 6.5 is written with the residual evaluated at time level $n+1$,

$$|\mathbf{\Omega}|\frac{d\mathbf{u}_h}{dt} + \mathbf{R}\Big(\mathbf{u}_h^{n+1}\Big) = \mathbf{0}\ . \tag{6.40}$$

The residual is expanded in a Taylor series about time level n as follows

$$\mathbf{R}\Big(\mathbf{u}_h^{n+1}\Big) = \mathbf{R}(\mathbf{u}_h^n) + \frac{\partial \mathbf{R}}{\partial \mathbf{u}_h}\Delta\mathbf{u}_h^n + O\Big(||\Delta\mathbf{u}_h||^2\Big) \tag{6.41}$$

where $\Delta\mathbf{u}_h^n = \mathbf{u}_h^{n+1} - \mathbf{u}_h^n$. By inserting Eq. 6.41 into Eq. 6.40, discretizing the time derivative with a backward finite-difference, and neglecting higher-order terms, the temporal update for the solution may be written as

$$\left[\frac{|\mathbf{\Omega}|}{\Delta t} + \frac{\partial \mathbf{R}}{\partial \mathbf{u}_h}\right]\Delta\mathbf{u}_h^n = -\mathbf{R}(\mathbf{u}_h^n)\ . \tag{6.42}$$

Eq. 6.42 represents a linear system for the solution update across the entire computational domain and is solved at each time step using the linear solver derived type discussed in Section 6.4.4.

For steady-state problems, the solution is advanced using pseudo-transient continuation (PTC) [45, 46]. At each time step, the solution update, $\Delta \mathbf{u}_h^n$, is first modified to ensure that the updated state, $\mathbf{u}_h^{n+1}$, is realizable. If the update cannot provide a realizable state, the update is rejected, and the CFL number is reduced. Otherwise, a simple line search is performed to ensure a sufficient decrease in the steady-state residual [47, 48]. During initial transients, it may be necessary for the residual to rise. In this case, the increase in residual norms is limited to 20%. For more details regarding PTC, refer to Algorithm 2 and Algorithm 3. The solution procedure is further accelerated using local time stepping and CFL evolution [49, 50, 51]. For a complete list and description of the CFL evolution schemes available in BlueRidge, refer to Appendix H.

Algorithm 2 Pseudo-Transient Continuation

Inputs: $\mathbf{u}_h^n$, $\Delta \mathbf{u}_h^n$
$\omega \leftarrow 1.0$
Call Algorithm 3 ▷ Enforce Realizability
if $\omega = 0.0$ **then** ▷ Reject update & reduce CFL
  return $\mathbf{u}_h^{n+1} \leftarrow \mathbf{u}_h^n$, ω
end if ▷ Perform simple line search
while $\left|\left|\mathbf{R}\left(\mathbf{u}_h^{n+1}\right)\right|\right|_2 > ||\mathbf{R}(\mathbf{u}_h^n)||_2 + \alpha\omega\left[\mathbf{R}(\mathbf{u}_h^n)^T \frac{\partial \mathbf{R}}{\partial \mathbf{u}_h} \Delta \mathbf{u}_h^n\right]$ **do**
  $\omega \leftarrow \frac{\omega}{2}$
  if $\omega < \omega_{min}$ .and. $\frac{\left|\left|\mathbf{R}\left(\mathbf{u}_h^{n+1}\right)\right|\right|_2}{\left|\left|\mathbf{R}\left(\mathbf{u}_h^{n}\right)\right|\right|_2} > 1.2$ **then** ▷ Limit residual rise to 20%
    return $\mathbf{u}_h^{n+1} \leftarrow \mathbf{u}_h^n$, $\omega \leftarrow 0.0$ ▷ Reject update & reduce CFL
  end if
  Call Algorithm 3 ▷ Enforce Realizability
  if $\omega = 0.0$ **then** ▷ Reject update & reduce CFL
    return $\mathbf{u}_h^{n+1}$, ω
  end if
end while
return $\mathbf{u}_h^{n+1}$, ω ▷ Accept update & adjust CFL

Algorithm 3 Enforcement of Realizable Solution Update

Inputs: $\mathbf{u}_h^n$, $\Delta\mathbf{u}_h^n$, ω
$\omega_p \leftarrow \omega$
$\mathbf{u}_h^{n+1} \leftarrow \mathbf{u}_h^n + \omega_p \Delta\mathbf{u}_h^n$
while $\mathbf{u}_h^{n+1}$ is not realizable **do**
  $\omega_p \leftarrow \frac{\omega_p}{2}$
  if $\omega_p < \omega_{min}$ **then**

    return $\mathbf{u}_h^{n+1} \leftarrow \mathbf{u}_h^n$, $\omega \leftarrow 0.0$
  else
    $\mathbf{u}_h^{n+1} \leftarrow \mathbf{u}_h^n + \omega_p \Delta\mathbf{u}_h^n$
  end if
end while

return $\mathbf{u}_h^{n+1}$, $\omega \leftarrow \omega_p$

All entries of the residual Jacobian given in Eq. 6.42 are computed via hand differentiation and checked for correctness with complex step finite-differences [35, 36]. In a similar manner, the build-up of the full residual Jacobian matrix is checked using standard finite-differences. The residual Jacobian matrix may directly correspond to the right-hand side residual vector or to that of a lower-order discretization. While a lower-order Jacobian requires significantly less memory than a higher-order Jacobian and is often more diagonally dominant, a lower-order Jacobian can result in a decreased convergence rate [52]. With the higher-order Jacobian, the quadratic convergence of Newton's method can be recovered as $\Delta t \to \infty$.

6.4.3 Nonlinear Solver Class (`solver_t`)

The nonlinear solver class, `solver_t`, is used to distinguish which solver, and ultimately which residual, is evolved by the `time_integrate_t` class. Up to this point, the discussion has centered around the `primal_solver_t` class which is employed to solve a given equation set subject to prescribed geometry and boundary conditions. BlueRidge also provides an intermediate abstract class, `error_solver_t`, which can be used to estimate numerical errors for a given primal solution. Each `solver_t` subclass is designed to interface with the `time_integrate_t` class in the same way. Therefore, all acceleration techniques available in the `time_integrate_t` class may be used by any solver. The only differences between subclasses are the definitions of the residual and how solution updates are formed and applied. Since much of the `primal_solver_t` subclass has already been described in detail, the remaining discussion of the `solver_t` class is focused on `error_solver_t` subclasses. A hierarchy of the `solver_t` class can be found in Figure 6.4.

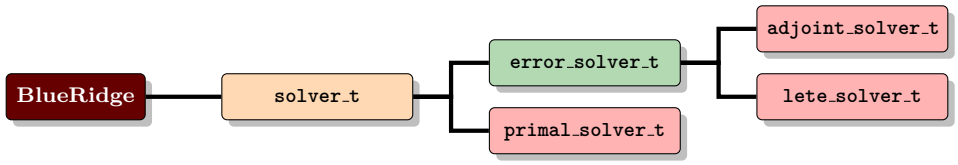


Figure 6.4: Hierarchy of Nonlinear Solver Class

6.4.3.1 Adjoint Solver (`adjoint_solver_t`)

The `adjoint_solver_t` subclass is used to implement a discrete adjoint solver [53, 54, 55, 56]. Adjoint methods are commonly used to estimate the discretization error in an output functional, $J_h(\mathbf{u}_h)$, and to formulate output-based adaptation indicators. Output functionals are defined for a given equation set and are accessible through the `physics_t` class. The residual for the `adjoint_solver_t` subclass is defined as follows

$$\mathbf{R}^{adj}(\lambda_h) = -\frac{\partial J_h}{\partial \mathbf{u}_h} + \left(\frac{\partial \mathbf{R}}{\partial \mathbf{u}_h}\right)^T \lambda_h \tag{6.43}$$

where λ_h are discrete adjoint variables. The adjoint residual is marched in pseudo-time in a similar manner to Ref. [57]. The form of Eq. 6.42 used for the `adjoint_solver_t` subclass is identical to that of the `primal_solver_t` subclass except that the left-hand side residual Jacobian is transposed. To obtain an accurate error estimate, the Jacobian in Eq. 6.43 must correspond to exact linearization of the primal residual.

6.4.3.2 Error Transport Solver (`lete_solver_t`)

The `lete_solver_t` subclass is used to implement a linearized error transport solver (ETE) [54, 58]. ETE methods are used to estimate local discretization errors in the primal solution, i.e., $\varepsilon_h = \mathbf{u}_h - I^h\tilde{\mathbf{u}}$. The residual for the `lete_solver_t` subclass is defined as follows

$$\mathbf{R}^{ETE}(\varepsilon_h) = \tau_h^p(\tilde{\mathbf{u}}) + \frac{\partial \mathbf{R}}{\partial \mathbf{u}_h}\varepsilon_h \tag{6.44}$$

where $\tau_h^p(\tilde{\mathbf{u}})$ is the truncation error. Since the exact continuous solution, $\tilde{\mathbf{u}}$, is not typically known, truncation error must be estimated [59, 60]. Similar to the `adjoint_solver_t` subclass, the residual Jacobian in Eq. 6.44 must be the exact linearization of the primal residual if an accurate error estimate is to be obtained.

6.4.4 Linear Solver Type (`linear_system_t`)

The `linear_system_t` derived type stores all data necessary to solve a linear system of the form

$$A\mathbf{x} = \mathbf{b} \ . \tag{6.45}$$

Due to the compact support of most PDE discretizations, the left-hand side matrix is predominantly sparse. Therefore, storing the left-hand side matrix as a dense matrix would be a gross misuse of computational resources. To conserve memory, all left-hand side matrices are stored in a compressed sparse row (CSR) format [61].

The `linear_system_t` derived type houses two abstract classes, the `iterative_solver_t` class and the `preconditioner_t` class. The `iterative_solver_t` class defines a specific iterative solver to be used in the solution of Eq. 6.45. The only iterative solver currently available in BlueRidge is GMRES [62]. The addition of other iterative solvers, such as BiCGStab [63], is relatively straightforward. The `preconditioner_t` class contains all procedures and data structures necessary to calculate and apply a preconditioner matrix, M. BlueRidge provides the capability of preconditioning from the left,

$$M^{-1}(A\mathbf{x} - \mathbf{b}) = \mathbf{0} \ , \tag{6.46}$$

preconditioning from the right,

$$AM^{-1}M\mathbf{x} = \mathbf{b} \ , \tag{6.47}$$

or split preconditioning,

$$M_1^{-1}\left(AM_2^{-1}M_2\mathbf{x} - \mathbf{b}\right) = \mathbf{0} \ . \tag{6.48}$$

The preconditioners currently available include ILU(0), ILU(k), and ILUTP [61]. A hierarchy of the `linear_system_t` derived type is given in Figure 6.5.

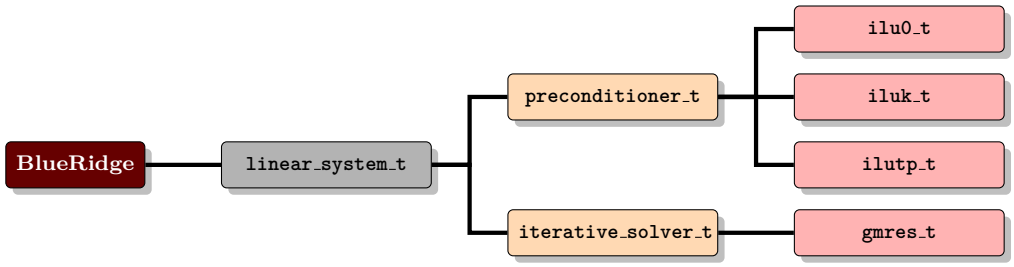


Figure 6.5: Hierarchy of Linear Solver Type

6.4.5 Exact Solution Class (`exact_t`)

Included in the BlueRidge code base is a suite of exact and manufactured solutions. This suite is intended to be used as a test bed for research applications and to assist with code verification. The `exact_t` class is designed to loosely interface with BlueRidge so that it can be extracted for use in other code bases, if necessary. Each subclass is defined for a specific equation set. A listing of the exact and manufactured solutions available in BlueRidge is given in Figure 6.6. For a complete description of each test problem, refer to Appendix I.

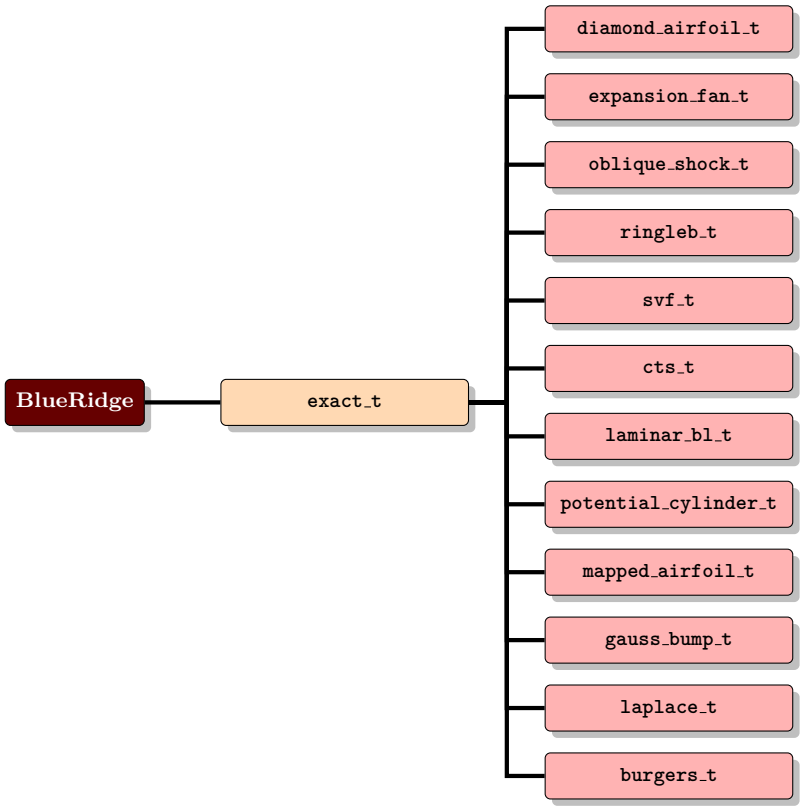


Figure 6.6: Hierarchy of Exact Solution Class

6.5 Code Verification

When developing a scientific computing code, it is critical to ensure that the code is producing the expected result. The portion of the code development process devoted to establishing correctness is referred to as code verification [64, 65]. At a component level, unit tests are employed to verify individual algorithms and routines within BlueRidge where an exact result can be determined. At a system level, however, it can be more difficult to assess correctness as an exact solution for a given PDE typically does not exist. Furthermore, the "correct" solution is largely affected by problem-dependent factors such as the spatial discretization, mesh resolution, and the time step [66]. To evaluate system-level correctness, exact solutions are instead fabricated using the method of manufactured solutions (MMS), and order of accuracy tests are performed to monitor the accuracy of the numerical solution.

6.5.1 Unit Testing

A unit test, as the name suggests, is a fine grain test for a specific unit of the code. A unit test can be a simple function/subroutine or can even be the size of a small module. Unit tests are the basis of the test-driven development (TDD) paradigm, where programmers first write unit tests followed by the code needed to pass the unit tests. This approach to programming requires developers to fully plan the structure of the code from the definition of variables to the design of procedures. In BlueRidge, however, developers adopt an agile programming paradigm where source code and unit tests are written concurrently. When writing a unit test for a given procedure, it is critical that each logical path through the code be tested. Once a procedure has been thoroughly unit tested, it can typically be disregarded if an error occurs. In this way, unit testing allows a programmer to focus debugging efforts on larger portions of the code. While writing comprehensive unit tests can be time consuming, it can save a significant amount of debugging time; it has been heuristically appraised that the ratio of debugging time to programming time for programmers who do not write unit tests is approximately 5:1 [66].

BlueRidge employs pFUnit, a Fortran-based testing framework, for all unit testing [67]. pFUnit contains all testing procedures commonly found in testing frameworks such as `assertEqual` and `assertTrue`. pFUnit provides support for most modern Fortran features and can even be used to test codes with MPI parallelism. Within the pFUnit testing framework, unit tests are written in a Fortran-based scripting language. Once unit tests are written, pFUnit converts the unit test files into pure Fortran code that is then compiled and linked against BlueRidge to create executables. Each unit test executable can be run to check whether a test passed or failed. To encourage and facilitate unit testing, we have created CMake macros to automatically parse and compile unit test source files and to generate a script which runs all unit test executables.

Currently, unit tests have been written for nearly all low-level math procedures such as vector norms, set routines, and sorting routines. Unit tests are included for mid-level procedures such as numerical quadratures, reconstruction routines, fluxes, and flux Jacobians. Unit tests are also available for high-level procedures that would not traditionally be considered for unit testing due to their size and complexity. These larger unit tests cover functionals, functional Jacobians, iterative solvers, preconditioners, and the build-up of the full residual Jacobian. Many of the unit tests included in BlueRidge are written as parameterized tests where a procedure is tested several times with different inputs and expected outputs. By parameterizing unit tests, multiple unit tests can be created with a minimal amount of code. In the future, these unit tests will be added to a continuous integration system, such as Jenkins [68], so that unit testing can be performed automatically whenever changes are committed.

6.5.2 Order of Accuracy Testing

Order of accuracy tests are used to evaluate whether or not the error in the numerical solution converges with grid refinement at the design order of the discretization. These tests compute numerical solutions on a series of systematically refined grids and monitor the convergence of discretization error. As previously mentioned, complex equation sets often do not have an exact solution, and, therefore, discretization error cannot be directly evaluated. However, even when an exact solution is not known, order of accuracy tests may still be performed using the method of manufactured solutions (MMS) [66, 69, 70]. With MMS, a smooth, analytic, manufactured solution is defined with as much or more continuity than what is prescribed by the PDE. Since the manufactured solution does not exactly satisfy the PDE, a residual is produced by operating the PDE onto the manufactured solution. This residual is inserted into the original PDE as a source term to drive the numerical solution toward the prescribed manufactured solution. In this manner, discretization error can be directly computed, and the observed order of accuracy of the scheme may be assessed.

MMS order of accuracy tests are an invaluable code verification tool. The MMS procedure is independent of the discretization scheme and can be rather sensitive to coding mistakes. Consequently, programmers can identify any lingering bugs in the code. If a bug is identified, terms in the discretization may be commented out to isolate and correct the bug. Through careful design of the manufactured solution, MMS may even be used to verify the implementation of boundary conditions as well as the temporal accuracy of the code [71]. Best practice dictates that MMS order of accuracy tests be performed on non-Cartesian grids since Cartesian grids exhibit error cancellation which ultimately can disguise bugs. Non-Cartesian grids typically prevent error cancellation and offer a more rigorous order of accuracy test. Furthermore, to ensure comprehensive code coverage, verification cases should be created to exercise each branch of the code.

In BlueRidge, we choose to implement the weak form MMS procedure by operating the integral form of the PDE onto the manufactured solution [54]. While strong form MMS lends itself nicely to finite-difference discretizations, weak form MMS is more suitable to the finite-volume discretization used in BlueRidge. For complex equation sets, strong form MMS is rather tedious to implement as it requires extensive algebraic manipulation of complicated functions. Even with algebraic manipulation software, the determination and coding of strong form MMS source terms can be a very error-prone process. With weak form MMS, the amount of algebraic manipulation required to perform code verification is reduced as fewer derivatives are needed. However, the programmer must ensure that the procedures used to implement weak form MMS, such as analytic flux routines, are thoroughly unit tested so that bugs are not directly introduced into the order of accuracy test.

The manufactured solution used for verification of BlueRidge takes the following form

$$\begin{aligned}\phi(\mathbf{x}) = a_0 + a_x f_x\left(\frac{b_x \pi x}{L}\right) + a_y f_y\left(\frac{b_y \pi y}{L}\right) + a_z f_z\left(\frac{b_z \pi z}{L}\right) \\ + a_{xy} f_{xy}\left(\frac{b_{xy} \pi xy}{L}\right) + a_{yz} f_{yz}\left(\frac{b_{yz} \pi yz}{L}\right) + a_{xz} f_{xz}\left(\frac{b_{xz} \pi xz}{L}\right)\end{aligned} \tag{6.49}$$

where $\{a_0, a_x, \ldots, a_{xz}\}$ and $\{b_x, b_y, \ldots, b_{xz}\}$ are constants, L is a reference length, and the functions $\{f_x, f_y, \ldots, f_{xz}\}$ are sines and cosines. Eq. 6.49 is selected for code verification since it is smooth, analytic, and infinitely differentiable. The coefficients and functions used in Eq. 6.49 may be found in Appendix I.6. A plot of the manufactured solution for 2D and 3D grids may be found in Figure 6.7 and Figure 6.8, respectively. BlueRidge is verified for all combinations of the following code options:

- Spatial Dimensions: $d \in \{2, 3\}$
- Equation Set: Euler, Navier-Stokes
- Flux Functions: Roe, Van Leer
- Flow State: Subsonic, Supersonic
- Primal Order of Accuracy: $p \in \{1, 2, 3, 4\}$

For simplicity, results are only presented for one combination set. The convergence of discretization error norms with grid refinement may be seen in Figure 6.9 for the combination {Navier-Stokes, Roe, Supersonic, $p \in \{1, 2, 3, 4\}$}. The corresponding observed orders of accuracy may be found in Figure 6.10.

As the number of code options increases, the number of verification cases required to perform comprehensive order of accuracy tests becomes quite large. While rigorous code verification can be time consuming, it is also automatable. To expedite the code verification process, Python scripts are included with the BlueRidge code base to generate verification cases, run the solver, and post-process results. As BlueRidge continues to be developed and code verification needs to be reperformed, developers may utilize these scripts to relieve some of the burden of code verification. Moreover, these scripts will eventually be incorporated into a continuous integration system so that code verification can be performed automatically without any developer intervention.

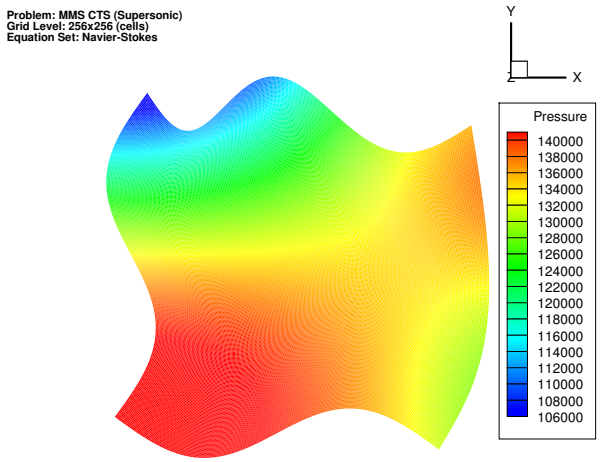


Figure 6.7: 2D Manufactured Solution

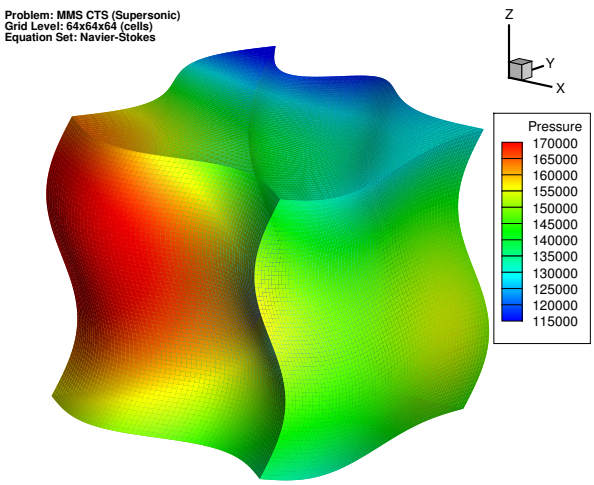


Figure 6.8: 3D Manufactured Solution

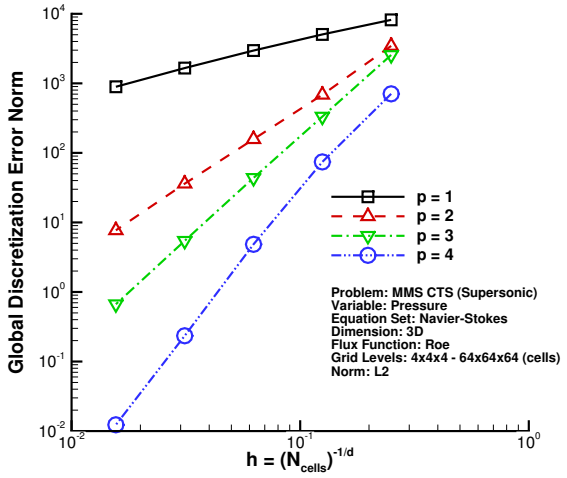


Figure 6.9: Convergence of Discretization Error

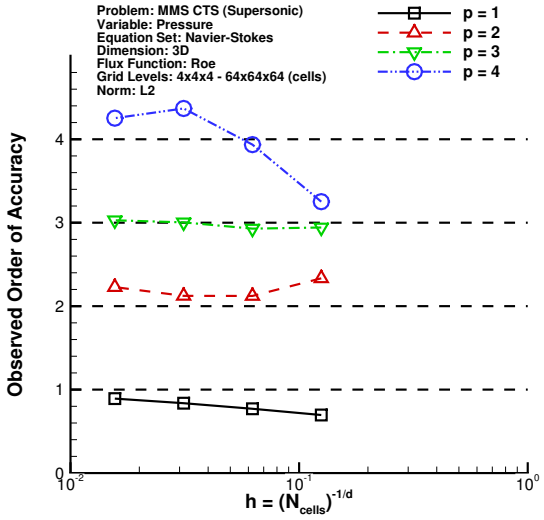


Figure 6.10: Observed Order of Accuracy

6.6 Conclusion

In this work, the development of BlueRidge, a higher-order finite-volume solver, has been described. BlueRidge developers have made a conscious effort to design a modular, flexible code base to increase the longevity of the code and to make the addition of features relatively painless for future developers. To meet design goals, BlueRidge makes extensive use of the

abstraction mechanisms and object-oriented programming features available in the Fortran 03/08 coding standards. BlueRidge developers have adhered to a strict coding standard to ensure high code quality and to improve the development workflow. Future development efforts will focus on the addition of turbulence models, parallelism via hybrid MPI + OpenMP, and a continuous integration system.

Acknowledgments

This work was supported by the NASA Graduate Aeronautics Scholarship as part of the Aeronautics Scholarship and Advanced STEM Training and Research Fellowship (AS&ASTAR) Program. We wish to thank Dr. Michael Park and Dr. Joseph Derlaga of NASA Langley Research Center for serving as technical advisors.

Bibliography

[1] V. Senguttuvan, S. Chalasani, E. Luke, D. Thompson, Adaptive mesh refinement using general elements, in: 43rd AIAA Aerospace Sciences Meeting and Exhibit, American Institute of Aeronautics and Astronautics, 2005. doi:10.2514/6.2005-927.
URL https://doi.org/10.2514%2F6.2005-927

[2] S. Chalasani, V. Senguttuvan, D. Thompson, E. Luke, On the use of general elements in fluid dynamics simulations, Communications in Numerical Methods in Engineering 24 (6) (2007) 435 – 448. doi:10.1002/cnm.1017.
URL https://doi.org/10.1002%2Fcnm.1017

[3] C. W. Bruner, Geometric properties of arbitrary polyhedra in terms of face geometry, AIAA Journal 33 (7) (1995) 1350 – 1350. doi:10.2514/3.12556.
URL https://doi.org/10.2514/3.12556

[4] T. Barth, P. Frederickson, Higher order solution of the Euler equations on unstructured grids using quadratic reconstruction, in: 28th Aerospace Sciences Meeting, American Institute of Aeronautics and Astronautics (AIAA), 1990. doi:10.2514/6.1990-13.
URL http://dx.doi.org/10.2514/6.1990-13

[5] T. Barth, Recent developments in high order k-exact reconstruction on unstructured meshes, in: 31st Aerospace Sciences Meeting, American Institute of Aeronautics and Astronautics (AIAA), 1993. doi:10.2514/6.1993-668.
URL http://dx.doi.org/10.2514/6.1993-668

[6] C. Ollivier-Gooch, M. Van Altena, A high-order-accurate unstructured mesh finite-volume scheme for the advection-diffusion equation, Journal of Computational Physics

181 (2) (2002) 729 – 752. doi:10.1006/jcph.2002.7159.
URL http://dx.doi.org/10.1006/jcph.2002.7159

[7] C. Ollivier-Gooch, A. Nejat, K. Michalak, Obtaining and verifying high-order unstructured finite volume solutions to the Euler equations, AIAA Journal 47 (9) (2009) 2105 – 2120. doi:10.2514/1.40585.
URL http://dx.doi.org/10.2514/1.40585

[8] D. Mavriplis, Revisiting the least-squares procedure for gradient reconstruction on unstructured meshes, in: 16th AIAA Computational Fluid Dynamics Conference, American Institute of Aeronautics and Astronautics (AIAA), 2003, p. 3986. doi:10.2514/6.2003-3986.
URL https://doi.org/10.2514/6.2003-3986

[9] A. Jalali, An adaptive higher-order unstructured finite volume solver for turbulent compressible flows, Ph.D. thesis, Univeristy of British Columbia (Jan. 2017).
URL http://hdl.handle.net/2429/60365

[10] G. H. Golub, C. Van Loan, Matrix Computations, 3rd Edition, Johns Hopkins University Press, 1996.

[11] T. Barth, D. Jespersen, The design and application of upwind schemes on unstructured meshes, in: 27th Aerospace Sciences Meeting, American Institute of Aeronautics and Astronautics (AIAA), 1989. doi:10.2514/6.1989-366.
URL http://dx.doi.org/10.2514/6.1989-366

[12] V. Venkatakrishnan, On the accuracy of limiters and convergence to steady state solutions, in: 31st AIAA Aerospace Sciences Meeting, American Institute of Aeronautics and Astronautics (AIAA), 1993. doi:10.2514/6.1993-880.
URL https://doi.org/10.2514/6.1993-880

[13] C. Michalak, Efficient high-order accurate unstructured finite-volume algorithms for viscous and inviscid compressible flows, Ph.D. thesis, University of British Columbia (April 2009).
URL http://hdl.handle.net/2429/7094

[14] C. Ollivier-Gooch, A toolkit for numerical simulation of PDEs: I. Fundamentals of generic finite-volume simulation, Computer Methods in Applied Mechanics and Engineering 192 (9-10) (2003) 1147 – 1175. doi:10.1016/s0045-7825(02)00602-3.
URL https://doi.org/10.1016%2Fs0045-7825%2802%2900602-3

[15] C. Van Loan, On the method of weighting for equality-constrained least-squares problems, SIAM Journal on Numerical Analysis 22 (5) (1985) 851 – 864. doi:10.1137/0722051.
URL https://doi.org/10.1137/0722051

[16] S. J. Chapman, Fortran 90/95 for Scientists and Engineers, McGraw-Hill, Inc., 2003.

[17] N. Clerman, W. Spector, Modern Fortran: Style and Usage, Cambridge University Press (CUP), 2011.

[18] S. Chacon, B. Straub, Pro Git, Apress, 2014.

[19] C. L. Lawson, R. J. Hanson, D. R. Kincaid, F. T. Krogh, Basic linear algebra subprograms for fortran usage, ACM Trans. Math. Softw. 5 (3) (1979) 308 – 323. doi:10.1145/355841.355847.
URL http://doi.acm.org/10.1145/355841.355847

[20] E. Anderson, Z. Bai, C. Bischof, S. Blackford, J. Demmel, J. Dongarra, J. Du Croz, A. Greenbaum, S. Hammarling, A. McKenney, D. Sorensen, LAPACK Users' Guide, 3rd Edition, Society for Industrial and Applied Mathematics, 1999.

[21] D. Poirier, S. Allmaras, D. McCarthy, M. Smith, F. Enomoto, The CGNS system, in: 29th AIAA Fluid Dynamics Conference, American Institute of Aeronautics and Astronautics (AIAA), 1998, p. 3007. doi:10.2514/6.1998-3007.
URL https://doi.org/10.2514/6.1998-3007

[22] D. Poirier, R. Bush, R. Cosner, C. Rumsey, D. McCarthy, Advances in the CGNS database standard for aerodynamics and CFD, in: 38th Aerospace Sciences Meeting and Exhibit, American Institute of Aeronautics and Astronautics (AIAA), 2000, p. 681. doi:10.2514/6.2000-681.
URL https://doi.org/10.2514/6.2000-681

[23] C. Rumsey, B. Wedan, T. Hauser, M. Poinot, Recent updates to the CFD general notation system (CGNS), in: 50th AIAA Aerospace Sciences Meeting including the New Horizons Forum and Aerospace Exposition, American Institute of Aeronautics and Astronautics (AIAA), 2012, p. 1264. doi:10.2514/6.2012-1264.
URL https://doi.org/10.2514/6.2012-1264

[24] M. Poinot, C. L. Rumsey, Seven keys for practical understanding and use of CGNS, in: 2018 AIAA Aerospace Sciences Meeting, American Institute of Aeronautics and Astronautics (AIAA), 2018, p. 1503. doi:10.2514/6.2018-1503.
URL https://doi.org/10.2514/6.2018-1503

[25] N. Nethercote, J. Seward, Valgrind: A framework for heavyweight dynamic binary instrumentation, in: ACM Sigplan notices, Vol. 42, ACM, 2007, pp. 89 – 100. doi:10.1145/1250734.1250746.
URL http://dx.doi.org/10.1145/1250734.1250746

[26] K. Martin, B. Hoffman, Mastering CMake: A cross-platform build system, Kitware, 2010.

[27] D. Pfeifer, Effective CMake: A random selection of best practices, C++ Now 2017.
URL https://cppnow2017.sched.com/event/A8J6/effective-cmake

[28] R. T. Biedron, J.-R. Carlson, J. M. Derlaga, P. A. Gnoffo, D. P. Hammond, W. T. Jones, B. Kleb, E. M. Lee-Rausch, E. J. Nielsen, M. A. Park, C. L. Rumsey, J. L. Thomas, W. A. Wood, FUN3D user's manual version 13.3, Tech. Rep. NASA-TM-2018-219808, NASA (2018).

[29] P. Roe, Approximate Riemann solvers, parameter vectors, and difference schemes, Journal of Computational Physics 43 (2) (1981) 357 – 372. doi:10.1016/0021-9991(81)90128-5.
URL http://dx.doi.org/10.1016/0021-9991(81)90128-5

[30] B. van Leer, Flux-vector splitting for the Euler equations, Lecture Notes in Physics (1982) 507 – 512doi:10.1007/3-540-11948-5_66.
URL http://dx.doi.org/10.1007/3-540-11948-5_66

[31] M.-S. Liou, C. J. Steffen, A new flux splitting scheme, Journal of Computational Physics 107 (1) (1993) 23 – 39. doi:10.1006/jcph.1993.1122.
URL http://dx.doi.org/10.1006/jcph.1993.1122

[32] M.-S. Liou, A sequel to AUSM: AUSM+, Journal of Computational Physics 129 (2) (1996) 364 – 382. doi:10.1006/jcph.1996.0256.
URL https://doi.org/10.1006%2Fjcph.1996.0256

[33] J. L. Steger, R. Warming, Flux vector splitting of the inviscid gasdynamic equations with application to finite-difference methods, Journal of Computational Physics 40 (2) (1981) 263 – 293. doi:10.1016/0021-9991(81)90210-2.
URL http://dx.doi.org/10.1016/0021-9991(81)90210-2

[34] P. Buning, J. Steger, Solution of the two-dimensional Euler equations with generalized coordinate transformation using flux vector splitting, in: 3rd Joint Thermophysics, Fluids, Plasma and Heat Transfer Conference, American Institute of Aeronautics and Astronautics (AIAA), 1982. doi:10.2514/6.1982-971.
URL https://doi.org/10.2514%2F6.1982-971

[35] R. Abreu, Complex steps finite differences, Ph.D. thesis, Universidad de Granada (December 2013).

[36] E. J. Nielsen, W. L. Kleb, Efficient construction of discrete adjoint operators on unstructured grids using complex variables, AIAA Journal 44 (4) (2006) 827 – 836. doi:10.2514/1.15830.
URL http://dx.doi.org/10.2514/1.15830

[37] J. Blazek, Computational Fluid Dynamics - Principles and Applications, 2nd Edition, Elsevier Science Publishing Co., Inc., 2001.

[38] H. Nishikawa, Beyond interface gradient: A general principle for constructing diffusion schemes, in: 40th Fluid Dynamics Conference and Exhibit, American Institute of Aeronautics and Astronautics (AIAA), 2010, p. 5093. doi:10.2514/6.2010-5093.
URL https://doi.org/10.2514/6.2010-5093

[39] A. Jalali, M. Sharbatdar, C. Ollivier-Gooch, Accuracy analysis of unstructured finite volume discretization schemes for diffusive fluxes, Computers & Fluids 101 (2014) 220 – 232. doi:10.1016/j.compfluid.2014.06.008.
URL https://doi.org/10.1016/j.compfluid.2014.06.008

[40] J. Burgers, A mathematical model illustrating the theory of turbulence, Advances in Applied Mechanics 1 (1948) 171 – 199. doi:10.1016/S0065-2156(08)70100-5.
URL https://doi.org/10.1016/S0065-2156(08)70100-5

[41] A. Jameson, W. Schmidt, E. Turkel, Numerical solutions of the Euler equations by finite volume methods using runge-kutta time stepping schemes, in: 14th Fluid and Plasma Dynamics Conference, American Institute of Aeronautics and Astronautics (AIAA), 1981. doi:10.2514/6.1981-1259.
URL http://dx.doi.org/10.2514/6.1981-1259

[42] A. Jameson, T. J. Baker, N. P. Weatherill, Calculation of inviscid transonic flow over a complete aircraft, in: 24th AIAA Aerospace Sciences Meeting, Vol. 86, American Institute of Aeronautics and Astronautics (AIAA), 1986, p. 0103. doi:10.2514/6.1986-103.
URL https://doi.org/10.2514/6.1986-103

[43] J. Blazek, N. Kroll, R. Radespiel, C.-C. Rossow, Upwind implicit residual smoothing method for multi-stage schemes, in: 10th Computational Fluid Dynamics Conference, American Institute of Aeronautics and Astronautics (AIAA), 1991. doi:10.2514/6.1991-1533.
URL http://dx.doi.org/10.2514/6.1991-1533

[44] R. M. Beam, R. Warming, An implicit finite-difference algorithm for hyperbolic systems in conservation-law form, Journal of Computational Physics 22 (1) (1976) 87 – 110. doi:10.1016/0021-9991(76)90110-8.
URL http://dx.doi.org/10.1016/0021-9991(76)90110-8

[45] C. T. Kelley, D. E. Keyes, Convergence analysis of pseudo-transient continuation, SIAM Journal on Numerical Analysis 35 (2) (1998) 508 – 523. doi:10.1137/S0036142996304796.
URL https://doi.org/10.1137/S0036142996304796

[46] M. Ceze, K. Fidkowski, Pseudo-transient continuation, solution update methods, and CFL strategies for DG discretizations of the RANS-SA equations, in: 21st AIAA Computational Fluid Dynamics Conference, American Institute of Aeronautics and Astro-

nautics (AIAA), 2013, p. 2686. doi:10.2514/6.2013-2686.
URL https://doi.org/10.2514/6.2013-2686

[47] J. E. Dennis Jr, R. B. Schnabel, Numerical methods for unconstrained optimization and nonlinear equations, SIAM, 1996. doi:10.1137/1.9781611971200.
URL https://doi.org/10.1137/1.9781611971200

[48] C. T. Kelley, Iterative Methods for Linear and Nonlinear Equations, Society for Industrial and Applied Mathematics, 1995. doi:10.1137/1.9781611970944.
URL http://dx.doi.org/10.1137/1.9781611970944

[49] W. A. Mulder, B. Van Leer, Experiments with implicit upwind methods for the Euler equations, Journal of Computational Physics 59 (2) (1985) 232 – 246. doi:10.1016/0021-9991(85)90144-5.
URL https://doi.org/10.1016/0021-9991(85)90144-5

[50] H. M. Bücker, B. Pollul, A. Rasch, On CFL evolution strategies for implicit upwind methods in linearized Euler equations, International Journal for Numerical Methods in Fluids 59 (1) (2009) 1 – 18. doi:10.1002/fld.1798.
URL https://doi.org/10.1002/fld.1798

[51] M. Ceze, A robust hp-adaptation method for discontinuous galerkin discretizations applied to aerodynamic flows, Ph.D. thesis, University of Michigan (2013).
URL http://hdl.handle.net/2027.42/97795

[52] D. Mavriplis, On convergence acceleration techniques for unstructured meshes, in: 29th AIAA Fluid Dynamics Conference, American Institute of Aeronautics and Astronautics (AIAA), 1998, p. 2966. doi:10.2514/6.1998-2966.
URL https://doi.org/10.2514/6.1998-2966

[53] M. Giles, M. Duta, J.-D. Müller, N. Pierce, Algorithm developments for discrete adjoint methods, AIAA Journal 41 (2) (2003) 198 – 205. doi:10.2514/2.1961.
URL http://dx.doi.org/10.2514/2.1961

[54] J. M. Derlaga, Application of improved truncation error estimation techniques to adjoint based error estimation and grid adaptation, Ph.D. thesis, Virginia Polytechnic Institute and State University, Blacksburg, VA (Jul. 2015).
URL http://hdl.handle.net/10919/54592

[55] D. A. Venditti, Grid adaptation for functional outputs of compressible flow simulations, Ph.D. thesis, Massachusetts Institute of Technology (Jun. 2002).
URL http://hdl.handle.net/1721.1/29246

[56] M. Park, Adjoint-based, three-dimensional error prediction and grid adaptation, in: 32nd AIAA Fluid Dynamics Conference and Exhibit, American Institute of Aeronautics

and Astronautics (AIAA), 2002. doi:10.2514/6.2002-3286.
URL http://dx.doi.org/10.2514/6.2002-3286

[57] E. J. Nielsen, J. Lu, M. A. Park, D. L. Darmofal, An implicit, exact dual adjoint solution method for turbulent flows on unstructured grids, Computers & Fluids 33 (9) (2004) 1131 – 1155. doi:10.1016/j.compfluid.2003.09.005.
URL http://dx.doi.org/10.1016/j.compfluid.2003.09.005

[58] C. Roy, Review of discretization error estimators in scientific computing, in: 48th AIAA Aerospace Sciences Meeting Including the New Horizons Forum and Aerospace Exposition, American Institute of Aeronautics and Astronautics (AIAA), 2010. doi:10.2514/6.2010-126.
URL http://dx.doi.org/10.2514/6.2010-126

[59] T. Phillips, Residual-based discretization error estimation for computational fluid dynamics, Ph.D. thesis, Virginia Polytechnic Institute and State University (Sep. 2014).
URL http://hdl.handle.net/10919/50647

[60] M. Sharbatdar, Error estimation and mesh adaptation paradigm for unstructured mesh finite volume methods, Ph.D. thesis, University of British Columbia (Jan. 2017).
URL http://hdl.handle.net/2429/60359

[61] Y. Saad, Iterative Methods for Sparse Linear Systems, Society for Industrial and Applied Mathematics, 2003. doi:10.1137/1.9780898718003.
URL http://dx.doi.org/10.1137/1.9780898718003

[62] Y. Saad, M. H. Schultz, GMRES: A generalized minimal residual algorithm for solving nonsymmetric linear systems, SIAM Journal on Scientific and Statistical Computing 7 (3) (1986) 856 – 869. doi:10.1137/0907058.
URL http://dx.doi.org/10.1137/0907058

[63] H. A. van der Vorst, Bi-CGSTAB: A fast and smoothly converging variant of Bi-CG for the solution of nonsymmetric linear systems, SIAM Journal on Scientific and Statistical Computing 13 (2) (1992) 631 – 644. doi:10.1137/0913035.
URL http://dx.doi.org/10.1137/0913035

[64] AIAA, AIAA guide for the verification and validation of computational fluid dynamics simulations (1998). doi:10.2514/4.472855.
URL https://arc.aiaa.org/doi/book/10.2514/4.472855

[65] ASME, Guide for verification and validation in computational solid mechanics: ASME V&V 10-2006 (2006).

[66] W. L. Oberkampf, C. J. Roy, Verification and Validation in Scientific Computing, Cambridge University Press (CUP), 2010. doi:10.1017/cbo9780511760396.
URL http://dx.doi.org/10.1017/CBO9780511760396

[67] M. Rilee, T. Clune, Towards test driven development for computational science with pFUnit, in: Proceedings of the 2nd International workshop on Software Engineering for High Performance Computing in Computational Science and Engineering, IEEE Press, 2014, pp. 20 – 27. doi:10.1109/SE-HPCCSE.2014.5.
URL https://doi.org/10.1109/SE-HPCCSE.2014.5

[68] Jenkins continuous integration system, https://jenkins.io/ (2018).

[69] S. Steinberg, P. J. Roache, Symbolic manipulation and computational fluid dynamics, Journal of Computational Physics 57 (2) (1985) 251 – 284. doi:10.1016/0021-9991(85)90045-2.
URL https://doi.org/10.1016/0021-9991(85)90045-2

[70] P. J. Roache, Code verification by the method of manufactured solutions, Journal of Fluids Engineering 124 (1) (2002) 4. doi:10.1115/1.1436090.
URL http://dx.doi.org/10.1115/1.1436090

[71] A. Choudhary, Verification of compressible and incompressible computational fluid dynamics codes and residual-based mesh adaptation, Ph.D. thesis, Virginia Polytechnic Institute and State University (November 2014).
URL http://hdl.handle.net/10919/51169

Chapter 7

Discussion and Conclusions

Discretization error is often the largest numerical error in any given CFD simulation and is exceedingly difficult to estimate. If not properly quantified, discretization errors can significantly impact the accuracy and reliability of simulation results. In this work, research efforts were focused on improving the accuracy, efficiency, and robustness of discretization error estimators for the finite-volume method, arguably the most common discretization method used for CFD problems. Truncation error was shown to act as the driving source term for the generation, transport, and diffusion of discretization error. For the discretization error estimators examined, the accuracy of error estimates was determined to be largely dictated by the accuracy of truncation error estimates. Therefore, research efforts also addressed the accuracy of finite-volume truncation error estimation techniques.

7.1 Discretization Error Estimation

Discretization error estimates were computed by deriving and solving an auxiliary error transport equation (ETE) for the local discretization error in all primitive variables. A linearized form of the ETE was employed to reduce computational costs and minimize the burden of implementation. In addition to local discretization error estimates, the linearized ETE were extended to provide error estimates for output functionals of interest by analytically demonstrating an equivalence between the linearized ETE and discrete adjoint methods. This ETE approach to functional error estimation was shown to be particularly advantageous when error estimates are required for multiple output functionals. For 1D Burgers' equation and the quasi-1D nozzle problem, ETE-based functional error estimation was approximately n times faster than a standard adjoint method where n is the number of output functionals of interest.

With ETE-based functional error estimation, local discretization error estimates and functional error estimates may be obtained, and the solution of the adjoint problem may be

avoided. However, with this approach, practitioners no longer have access to the adjoint solution, a key component in the formulation of adaptation indicators. To leverage the advantages of both the ETE and adjoint methods, a new technique based on Krylov subspace methods was proposed to simultaneously provide local discretization error estimates, functional error estimates, and adjoint-based adaptation indicators. Functional error estimates were rewritten as bilinear forms, and BiCG [1] was utilized to concurrently solve the adjoint system and the ETE system, albeit in an approximate sense. For the quasi-1D nozzle problem, an adaptation procedure driven by approximate adjoint-based adaptation indicators yielded effectively the same adapted grid as an adaptation procedure driven with exact adjoint-based adaptation indicators. Furthermore, local discretization error estimates were in qualitatively good agreement with the fully converged ETE solution. Due to properties of the Krylov solver, highly accurate functional error estimates were obtained even when the bilinear form was only converged to a modest tolerance.

Adjoint/ETE equivalence was exploited to extend the higher-order properties of adjoint methods to the ETE. Similar to adjoint-based functional error estimates, linearized ETE discretization error estimates were shown to converge to the exact discretization error at a higher-order rate (i.e., faster than the convergence rate of the original primal solution). For the subsonic-supersonic solution of the quasi-1D nozzle problem, discretization error estimates for a 2^{nd} order primal solution converged to the true discretization error at a 4^{th} order rate. Rather than explicitly use the ETE solution as an error estimate, practitioners may choose to apply the ETE solution as a correction similar to defect correction methods [2]. When applied as a correction, ETE error estimates increased the observed order of accuracy of the entire primal solution; for the quasi-1D nozzle problem, the 2^{nd} order primal solution was improved to 4^{th} order with one ETE correction step. The computational advantages of this ETE approach to higher-order primal solutions was explored. In comparison to a conventional higher-order solution, the ETE approach to higher-order solutions was approximately an order of magnitude faster and yielded primal solutions with similar error levels. Higher-order solutions were obtained on uniform grids and grids with smoothly varying grid metrics. However, on grids with a non-smooth variation of grid metrics, higher-order accuracy was lost due to an insufficiently accurate truncation error estimate.

A method referred to as *ETE relinearization* was proposed to iteratively improve the accuracy of ETE error estimates. With this approach, the ETE were relinearized about a corrected primal solution and re-solved for an error estimate. The resulting error estimate was then re-applied as a correction to the primal solution. This iterative procedure was designed to terminate after a user-defined number of correction steps. Following the correction loop, an error estimate was formed by taking the difference between the corrected primal solution and the original lower-order primal solution. With ETE relinearization, error estimates, or equivalently, higher-order primal solutions, were computed with an observed order of accuracy as high as 8^{th} order accurate. For a given error level, ETE relinearization was approximately 1 - 3 orders of magnitude faster than a standard higher-order solver. Applications included the 2D viscous Burgers' equation, the quasi-1D nozzle problem, and

the 2D Euler equations. When the ETE are linearized, the primal solution is assumed to be sufficiently accurate such that higher-order terms in the residual expansion are small. For some problems, linearization errors (i.e., the leading higher-order terms) may not be negligible, especially when flow features are under-resolved. When applied to a turbulent airfoil problem, ETE relinearization reduced the effects of linearization error and yielded a corrected primal solution qualitatively identical to that of a higher-order solver.

7.2 Truncation Error Estimation

Truncation error estimates were computed using a higher-order discrete residual method. The lower-order primal solution was prolonged to a higher-order, piecewise continuous space using a k-exact least-squares reconstruction [3]. The leading order terms in the truncation error were then estimated by performing a discrete residual evaluation. Boundary conditions were explicitly enforced on the reconstructed primal solution to improve the accuracy of truncation error estimates. For the quasi-1D nozzle problem, truncation error estimates near domain boundaries improved significantly. In the absence of boundary condition enforcement, the convergence rate of ETE error estimates ultimately deteriorated to that of the lower-order primal solution on the finest grid levels. By enforcing boundary conditions on the reconstruction, higher-order accuracy was maintained for all grid levels.

For grids with smoothly varying grid metrics, higher-order discrete residual truncation error estimates were shown to be asymptotically correct in an effectivity index. However, for grids with non-smoothly varying grid metrics (e.g., general unstructured grids), truncation error estimates consistently underpredicted the magnitude of the exact truncation error. Consequently, ETE error estimates converged at a lower-order rate. It was determined that a key requirement of higher-order discrete residual methods was being violated, namely that the reconstructed solution should sufficiently approximate the exact continuous solution. More specifically, grid-induced fluctuations in the primal solution on the order of discretization error were exactly captured and produced a reconstruction which was overly oscillatory. Inter-element jumps in the reconstructed solution were excessively large and introduced a lower-order error into the truncation error estimate. To improve the accuracy of truncation error estimates, a reconstruction based on polyharmonic smoothing splines was presented which formulated a smooth approximation of the non-smooth finite-volume data by penalizing roughness of the fit. In contrast to a k-exact least-squares reconstruction, the smoothing spline reconstruction was much more representative of the exact continuous solution and yielded truncation error estimates that were asymptotically correct. Furthermore, ETE error estimates maintained higher-order accuracy with the smoothing spline reconstruction for several more grid levels. On the finest grids, ETE error estimates were approximately an order of magnitude more accurate than those obtained with a k-exact reconstruction.

7.3 Code and Algorithm Development

The test problems examined in this work were simulated using newly developed CFD codes. All codes were developed using a codified set of software engineering "best practices" to promote code longevity and to facilitate code maintenance. Furthermore, all codes were written in a nondimensional form to improve the numerical scaling of the solution procedure. To highlight the importance of writing scientific computing codes in a nondimensional form, a brief examination of residual Jacobian condition numbers was performed for the quasi-1D nozzle problem. The condition numbers of the dimensional residual Jacobians were found to be approximately 7 orders of magnitude larger than those of the nondimensional residual Jacobians. These findings indicated that dimensional CFD codes may be more susceptible to round-off errors and may have less favorable convergence properties.

Two-dimensional results were obtained using BlueRidge, a generic higher-order finite-volume solver. BlueRidge was written using the Fortran 2003/2008 standard and made extensive use of advanced programming constructs to provide a modular and flexible code base. The physics implementation within BlueRidge was abstracted to reduce to amount of repeated code and to expedite the addition of new equation sets. The equation sets currently implemented in BlueRidge include the Euler equations, the laminar Navier-Stokes equations, Burgers' equation, and Laplace's equation. Primal, adjoint, and ETE solvers are all included within the code base. Code verification was performed using the method of manufactured solutions [4] for primal orders of accuracy of $p \in \{1, 2, 3, 4\}$. A unit testing framework was established using pFUnit [5]. An exact solution class was provided to facilitate code verification and future research efforts.

7.4 Recommendations and Future Work

7.4.1 Discretization Error Estimation

The ETE are the recommended error estimator for finite-volume schemes. ETE error estimates are, in general, highly accurate assuming a suitable truncation error estimate can be obtained. Furthermore, the ETE only require the primal solution on one grid level to compute an error estimate. The ETE are computationally more efficient than other existing error estimators, such as Richardson extrapolation [6] or defect correction, especially if the linearized ETE are employed. With regard to functional error estimation, the ETE produce error estimates which are equivalent to that of an adjoint method. However, the ETE are computationally more efficient when multiple functional error estimates are required and are much simpler to implement, especially if an implicit primal solver is available. Moreover, when ETE relinearization is utilized, the ETE can provide much more accurate functional error estimates than an adjoint method since the ETE are not limited to just one correction

step.

The linearized ETE perform remarkably well when the discrete solution is in the asymptotic range. When the discrete solution is outside of the asymptotic range, higher-order terms in the residual expansion may not be small compared to the leading order term, and therefore, the accuracy of the ETE error estimate may suffer. In this regime, the full nonlinear ETE may be required. Similarly, the nonlinear ETE may be necessary in the presence of flowfield and geometric discontinuities to properly account for higher-order terms in the discretization error. Further research is needed to ensure accurate error estimates when the discrete solution is outside of the asymptotic range and when strong nonlinearities are present.

More recently, research efforts have focused on extending the ETE to unsteady flow problems [7]. With this approach, the ETE are co-evolved in time alongside the primal solution, albeit lagged by a fixed number of time steps. Consequently, the unsteady ETE are much more cost effective than an unsteady adjoint method since the entire time series of the primal solution does not need to be stored. Comparisons between the unsteady ETE and an unsteady adjoint method should be performed to fully demonstrate the potential and utility of the ETE. As the majority of CFD simulations shift from steady-state problems to unsteady problems, it is imperative that researchers continue to study and improve unsteady ETE methods.

ETE relinearization can significantly improve the accuracy of error estimates. However, since ETE relinearization is equivalent to executing a prescribed number of nonlinear iterations of a higher-order solver and formulating an error estimate from difference in solutions, ETE relinearization is likely subject to the same stability requirements as a higher-order solver. Further investigation is needed to verify the stability of ETE relinearization and ensure that ETE correction steps will actually improve the accuracy of the discrete solution.

In this work, the ETE were investigated within the context of the finite-volume method. As CFD methods continue to progress, the ETE will need to be examined within the context of new and emerging discretization methods, such as the discontinuous Galerkin (DG) method [8, 9] and the Streamline-upwind Petrov-Galerkin (SUPG) method [10, 11].

7.4.2 Truncation Error Estimation

The higher-order discrete residual method is the recommended truncation error estimator for finite-volume schemes. For grids with smoothly varying grid metrics, this approach yields truncation error estimates which are asymptotically very accurate. No assumption is made regarding the convergence rate of the exact truncation error as is required for embedded grid methods. Furthermore, a global reconstruction of the discrete solution can be avoided, and local reconstructions may be used instead. Future research efforts should focus on improving solution reconstruction methods for finite-volume truncation error estimation. In particular, reconstruction techniques for unstructured grids must be further developed

to properly account for non-smooth data. Additionally, reconstruction methods must be devised which can accurately represent the solution across flowfield discontinuities.

7.4.3 Code and Algorithm Development

The BlueRidge code base has been explicitly designed to facilitate the addition of new capabilities. Based on the current state of BlueRidge, the author identifies the following items as capabilities which should be added to the code base:

- RANS turbulence modeling
- A higher-order geometry capability
- An overset mesh capability
- Parallelism via MPI and OpenMP
- A design optimization module

Bibliography

[1] R. Fletcher, Conjugate gradient methods for indefinite systems, in: G. Watson (Ed.), Numerical Analysis 1975, Proceedings of the Dundee Conference on Numerical Analysis, no. 506, Springer Science + Business Media, 1976, pp. 73 – 89. doi:10.1007/bfb0080116.
URL http://dx.doi.org/10.1007/BFb0080116

[2] H. J. Stetter, The defect correction principle and discretization methods, Numerische Mathematik 29 (4) (1978) 425 – 443. doi:10.1007/bf01432879.
URL http://dx.doi.org/10.1007/BF01432879

[3] T. Barth, P. Frederickson, Higher order solution of the Euler equations on unstructured grids using quadratic reconstruction, in: 28th Aerospace Sciences Meeting, American Institute of Aeronautics and Astronautics (AIAA), 1990. doi:10.2514/6.1990-13.
URL http://dx.doi.org/10.2514/6.1990-13

[4] P. J. Roache, Code verification by the method of manufactured solutions, Journal of Fluids Engineering 124 (1) (2002) 4. doi:10.1115/1.1436090.
URL http://dx.doi.org/10.1115/1.1436090

[5] M. Rilee, T. Clune, Towards test driven development for computational science with pFUnit, in: Proceedings of the 2nd International workshop on Software Engineering for

High Performance Computing in Computational Science and Engineering, IEEE Press, 2014, pp. 20 – 27. doi:10.1109/SE-HPCCSE.2014.5.
URL https://doi.org/10.1109/SE-HPCCSE.2014.5

[6] L. Richardson, On the approximate arithmetical solution by finite differences of physical problems involving differential equations, with an application to the stresses in a masonry dam, Proceedings of the Royal Society of London A: Mathematical, Physical and Engineering Sciences 83 (563) (1910) 335–336. doi:10.1098/rspa.1910.0020.
URL http://rspa.royalsocietypublishing.org/content/83/563/335

[7] G. Yan, C. Ollivier-Gooch, Towards higher order discretization error estimation by error transport using unstructured finite-volume methods for unsteady problems, Computers & Fluids 154 (2017) 245–255. doi:10.1016/j.compfluid.2017.06.012.
URL https://doi.org/10.1016/j.compfluid.2017.06.012

[8] M. Ceze, K. J. Fidkowski, Drag prediction using adaptive discontinuous finite elements, Journal of Aircraft 51 (4) (2014) 1284 – 1294. doi:10.2514/1.c032622.
URL https://doi.org/10.2514%2F1.c032622

[9] K. Fidkowski, An output-based adaptive hybridized discontinuous galerkin method on deforming domains, 22nd AIAA Computational Fluid Dynamics Conferencedoi:10.2514/6.2015-2602.
URL http://dx.doi.org/10.2514/6.2015-2602

[10] J. C. Newman, W. K. Anderson, Investigation of unstructured higher-order methods for unsteady flow and moving domains, 22nd AIAA Computational Fluid Dynamics Conferencedoi:10.2514/6.2015-2917.
URL http://dx.doi.org/10.2514/6.2015-2917

[11] W. K. Anderson, J. Newman, High-order stabilized finite elements on dynamic meshes, in: 2018 AIAA Aerospace Sciences Meeting, 2018, p. 1307.

Appendix A

Nondimensionalization of the Quasi-1D Euler Equations

The quasi-1D Euler equations are nondimensionalized by normalizing each dimensional quantity in the following manner

$$
\begin{aligned}
T &= \frac{T^+}{T_0} & \rho &= \frac{\rho^+}{\rho_0} & u &= \frac{u^+}{a_0} \\
p &= \frac{p^+}{\rho_0 a_0^2} & e_t &= \frac{e_t^+}{a_0^2} & h_t &= \frac{h_t^+}{a_0^2} \\
x &= \frac{x^+}{L_d} & t &= \frac{t^+}{L_d/a_0} & A &= \frac{A^+}{A_{min}}
\end{aligned}
$$

where the notation $(\cdot)^+$ denotes a dimensional quantity, the notation $(\cdot)_0$ denotes a stagnation quantity, L_d is the length of the domain, and A_{min} is the minimum cross-sectional area.

Appendix B

Residual Jacobian Condition Numbers for the Quasi-1D Euler Equations

To illustrate the numerical impact of scaling the governing equations, the condition number of the residual Jacobian matrix for several uniform grids is presented for both subsonic and supersonic cases of the quasi-1D nozzle problem. Matrix scaling is also included as a reference. Matrix scaling is performed by applying row scaling from the left and column scaling from the right

$$Ax = b \quad \Leftrightarrow \quad D_r A D_c \tilde{x} = D_r b$$

where D_r and D_c are diagonal matrices whose entries are

$$D_{r,i} = \frac{1}{\sum_{j=1}^{n} |A_{ij}|} \quad D_{c,j} = \frac{1}{\sum_{i=1}^{n} |A_{ij}|}.$$

As can be seen in Table B.1, condition numbers are drastically improved by scaling the governing equations. Matrix scaling alone improves condition numbers by approximately 6-7 orders of magnitude. Directly nondimensionalizing the equations provides approximately an additional 2 orders of magnitude improvement.

Table B.1: Dimensional and nondimensional residual Jacobian condition numbers.

(a) Subsonic-subsonic

N_{cells}	Dimensional	Dimensional (Scaled)	Nondimensional
32	6.28e+10	1.38e+06	2.25e+04
64	3.45e+12	5.30e+06	1.06e+05
128	2.57e+13	1.75e+07	3.47e+05
256	7.78e+13	4.48e+07	8.64e+05
512	1.69e+14	9.47e+07	1.81e+06
1024	3.42e+14	1.91e+08	3.65e+06

(b) Subsonic-supersonic

N_{cells}	Dimensional	Dimensional (Scaled)	Nondimensional
32	1.07e+13	1.96e+06	9.58e+03
64	2.29e+13	4.22e+06	1.86e+04
128	5.25e+13	8.69e+06	3.71e+04
256	1.20e+14	1.76e+07	7.42e+04
512	2.71e+14	3.54e+07	1.49e+05
1024	6.04e+14	7.10e+07	2.97e+05

Appendix C

Solution of Least-Squares Problems by Singular Value Decomposition

Consider a linear system

$$A\mathbf{x} = \mathbf{b} \tag{C.1}$$

where $A \in \mathbb{R}^{m\times n}$ with $m \geq n$, $\mathbf{x} \in \mathbb{R}^n$, and $\mathbf{b} \in \mathbb{R}^m$. The matrix A may be decomposed using a singular value decomposition (SVD) as follows

$$A = U\Sigma V^T \tag{C.2}$$

where the columns of $U \in \mathbb{R}^{m\times m}$ and $V \in \mathbb{R}^{n\times n}$ are left and right singular vectors, respectively, and $\Sigma \in \mathbb{R}^{m\times n}$ is a diagonal matrix of singular values. The pseudo-inverse of A may be defined as follows

$$A^\dagger = V\Sigma^\dagger U^T \tag{C.3}$$

where

$$\Sigma^\dagger = \text{diag}\Big(\frac{1}{\sigma_1}, \ldots, \frac{1}{\sigma_r}, 0, \ldots, 0\Big), \quad r = \text{rank}(A)\ . \tag{C.4}$$

To reduce the impact of numerical errors introduced by ill-conditioning or rank deficiency, all singular values below $\tau = \sigma_1\sqrt{\epsilon}$ are set to zero where ϵ is machine precision. The least-squares solution to Eq. C.1 is given by

$$\mathbf{x} = A^\dagger\mathbf{b}\ . \tag{C.5}$$

For more information regarding the SVD and least-squares problems, refer to Ref. [1].

Bibliography

[1] G. H. Golub, C. Van Loan, Matrix Computations, 3rd Edition, Johns Hopkins University Press, 1996.

Appendix D

Code Example

Listing D.1: Derived Type: `shape_t`

```
module shape_derived_type

  use set_precision, only : rd

  implicit none

  private

  public :: shape_t

!================================== shape_t ==================================80
!>
!! Description: Abstract derived type for defining shapes.
!<
!=============================================================================80
  type, abstract :: shape_t

    ! Type of Shape
    character(30) :: shape_type = ''

    ! Number of Sides
    integer       :: N_sides = 0

    ! Length of Sides
    real(rd), dimension(:), allocatable :: side_length

  contains

    private

    ! Type-Bound Procedures
    procedure, public, pass :: get_sides
    procedure, public, pass :: destructor

    ! Deferred Procedures
    procedure(print_type_i),   public, pass, deferred :: print_type
    procedure(compute_area_i), public, pass, deferred :: compute_area

  end type shape_t
```

```
!================================== interfaces ================================80
!>
!! Description: Interfaces for the shape_t derived type.
!<
!==============================================================================80
  abstract interface
    !============================== print_type_i ==============================80
    !>
    !! Description: Prints the type information of the shape.
    !<
    !==========================================================================80
    subroutine print_type_i( this )

      import shape_t

      class(shape_t), intent(in) :: this

    end subroutine print_type_i

    !============================== compute_area_i ============================80
    !>
    !! Description: Returns the area of the shape.
    !<
    !==========================================================================80
    function compute_area_i( this ) result( area )

      use set_precision, only : rd

      import shape_t

      class(shape_t), intent(in) :: this
      real(rd)                   :: area

    end function compute_area_i
  end interface

contains

!================================== get_sides =================================80
!>
!! Description: Returns information about the side of the shape.
!!
!! Inputs:      this:        Access to the shape_t derived type
!!
!! Outputs:     N_sides:     Number of sides
!!              side_length: Length of each side
!<
!==============================================================================80
  subroutine get_sides( this, N_sides, side_length )

    class(shape_t),                    intent(in)  :: this
    integer,                           intent(out) :: N_sides
    real(rd), dimension(this%N_sides), intent(out) :: side_length

    continue

    N_sides     = this%N_sides
    side_length = this%side_length

  end subroutine get_sides

!================================== destructor ================================80
!>
```

```
!! Description: Destructor routine for the shape_t derived type
!!
!! Inputs:      this: Access to the shape_t derived type
!!
!! Ouputs:      this: Updated shape_t derived type
!<
!================================================================================80
  subroutine destructor( this )

    class(shape_t), intent(inout) :: this

    continue

    if ( allocated(this%side_length) ) then
      deallocate(this%side_length)
    end if

  end subroutine destructor

end module shape_derived_type
```

Listing D.2: Derived Type: `triangle_t`

```
module triangle_derived_type

  use set_precision,      only : rd
  use set_constants,      only : zero
  use shape_derived_type, only : shape_t

  implicit none

  private

  public :: triangle_t

!================================== triangle_t ================================80
!>
!! Description: Abstract derived type for defining triangles.
!<
!==============================================================================80
  type, abstract, extends(shape_t) :: triangle_t

    ! Type of Triangle
    character(30) :: triangle_type = ''

  contains

    ! Type-Bound Procedures
    procedure, public, pass :: setup_triangle_class

    ! Deferred Procedures
    procedure, public, pass :: print_type => print_type_obj

  end type triangle_t

contains

!============================ setup_triangle_class ============================80
!>
!! Description: Sets up the triangle class.
!!
!! Inputs:      this:          Access to the triangle_t derived type
!!              triangle_type: Type of triangle
!!
!! Outputs:     this:          Updated triangle_t derived type
!<
!==============================================================================80
  subroutine setup_triangle_class( this, triangle_type )

    use set_constants, only : zero

    class(triangle_t), intent(inout) :: this
    character(*),      intent(in)    :: triangle_type

    continue

    ! Store Type of Shape
    this%shape_type    = 'triangle'

    ! Store Type of Triangle
    this%triangle_type = trim(triangle_type)

    ! Set Number of Sides
    this%N_sides       = 3
```

```
    ! Allocate Storage for Side Lengths & Initialize
    allocate( this%side_length(3) )

    this%side_length = zero

  end subroutine setup_triangle_class

!================================== print_type ==============================80
!>
!! Description: Prints the type information of the shape.
!!
!! Inputs:      this: Access to the triangle_t derived type
!<
!============================================================================80
  subroutine print_type_obj( this )

    class(triangle_t), intent(in) :: this

    continue

    write(*,*)
    write(*,*) ' Primary   Shape Type: ', trim(this%shape_type)
    write(*,*) ' Secondary Shape Type: ', trim(this%triangle_type)
    write(*,*)

  end subroutine print_type_obj

end module triangle_derived_type
```

Listing D.3: Derived Type: `isosceles_triangle_t`

```
module isosceles_triangle_derived_type

  use triangle_derived_type, only : triangle_t

  implicit none

  private

  public :: isosceles_triangle_t

!============================== isosceles_triangle_t ========================80
!>
!! Description: Derived type for defining isosceles triangles.
!<
!============================================================================80
  type, extends(triangle_t) :: isosceles_triangle_t

  contains

    procedure, public, pass :: compute_area => compute_area_obj

  end type isosceles_triangle_t

  interface isosceles_triangle_t
    procedure constructor
  end interface isosceles_triangle_t

contains

!================================= constructor ==============================80
!>
!! Description: Constructor routine for the isosceles_triangle_t derived tpye.
!!
!! Inputs:      side_A:      Length of side A
!!              side_B:      Lenght of side B
!!
!! Outputs:     constructor: Constructed isosceles_triangle_t derived type
!<
!============================================================================80
  function constructor( side_A, side_B )

    use set_precision, only : rd

    real(rd),       intent(in) :: side_A, side_B
    type(isosceles_triangle_t) :: constructor

    continue

    ! Setup Triangle Class
    call constructor%setup_triangle_class( 'isosceles' )

    ! Store Length of each Side
    constructor%side_length = [ side_A, side_B, side_B ]

  end function constructor

!================================ compute_area ==============================80
!>
!! Description: Computes the area of an isosceles triangle.
!!
!! Inputs:      this: Access to the isosceles_triangle_t derived type
```

```
!!
!! Outputs:      area: Area of the isosceles triangle
!<
!================================================================================80
  function compute_area_obj( this ) result( area )

    use set_precision, only : rd
    use set_constants, only : zero, fourth, four

    class(isosceles_triangle_t), intent(in) :: this
    real(rd)                                :: area

    real(rd) :: side_A, side_B

    continue

    ! Initialize
    area = zero

    ! Extract Sides
    side_A = this%side_length(1)
    side_B = this%side_length(2)

    ! Compute Area
    area = fourth*side_B*sqrt( four*side_A**2 - side_B**2 )

  end function compute_area_obj

end module isosceles_triangle_derived_type
```

Listing D.4: Derived Type: `equilateral_triangle_t`

```
module equilateral_triangle_derived_type

  use triangle_derived_type, only : triangle_t

  implicit none

  private

  public :: equilateral_triangle_t

!============================== equilateral_triangle_t ========================80
!>
!! Description: Derived type for defining equilateral triangles.
!<
!==============================================================================80
  type, extends(triangle_t) :: equilateral_triangle_t

  contains

    procedure, public, pass :: compute_area => compute_area_obj

  end type equilateral_triangle_t

  interface equilateral_triangle_t
    procedure constructor
  end interface equilateral_triangle_t

contains

!================================== constructor ===============================80
!>
!! Description: Constructor routine for the equilateral_triangle_t derived tpye.
!!
!! Inputs:      side_A:      Length of side A
!!
!! Outputs:     constructor: Constructed equilateral_triangle_t derived type
!<
!==============================================================================80
  function constructor( side_A )

    use set_precision, only : rd

    real(rd),          intent(in) :: side_A
    type(equilateral_triangle_t) :: constructor

    continue

    ! Setup Triangle Class
    call constructor%setup_triangle_class( 'equilateral' )

    ! Store Length of each Side
    constructor%side_length = [ side_A, side_A, side_A ]

  end function constructor

!================================ compute_area ================================80
!>
!! Description: Computes the area of an equilateral triangle.
!!
!! Inputs:      this: Access to the equilateral_triangle_t derived type
!!
```

```
!! Outputs:      area: Area of the equilateral triangle
!<
!================================================================================80
  function compute_area_obj( this ) result( area )

    use set_precision, only : rd
    use set_constants, only : zero, fourth, three

    class(equilateral_triangle_t), intent(in) :: this
    real(rd)                                  :: area

    real(rd) :: side_A

    continue

    ! Initialize
    area = zero

    ! Extract Sides
    side_A = this%side_length(1)

    ! Compute Area
    area = fourth*sqrt(three)*side_A**2

  end function compute_area_obj

end module equilateral_triangle_derived_type
```

Appendix E

Boundary Conditions for Fluid Mechanics Subclasses

In BlueRidge, all `fluid_mechanics_t` boundary conditions are enforced weakly through the flux. Wall boundary conditions are imposed via a modified analytic flux. Inflow/outflow boundary conditions are imposed by applying a flux function at the boundary face with a specific outer state, $\mathbf{q}_R$. The determination of the outer state is unique to each boundary condition and is often a function of the local character of the flow field, the interior state, $\mathbf{q}_L$, freestream conditions, $\mathbf{q}_\infty$, and specified parameters, Φ. Figure E.1 is provided to clarify the nomenclature used for inflow/outflow boundaries. In subsequent sections, each `fluid_mechanics_t` boundary condition is described in detail. Accompanying each description is a table indicating the type of boundary condition, the compatible equation set, the variables to be specified and/or extrapolated from the interior for the computation of the outer state, and the form of the modified flux, if applicable. Unless otherwise specified, all boundary face gradients are taken from the interior reconstruction. For more information regarding the implementation of boundary conditions for the Euler or Navier-Stokes equations, refer to Refs. [1, 2, 3].

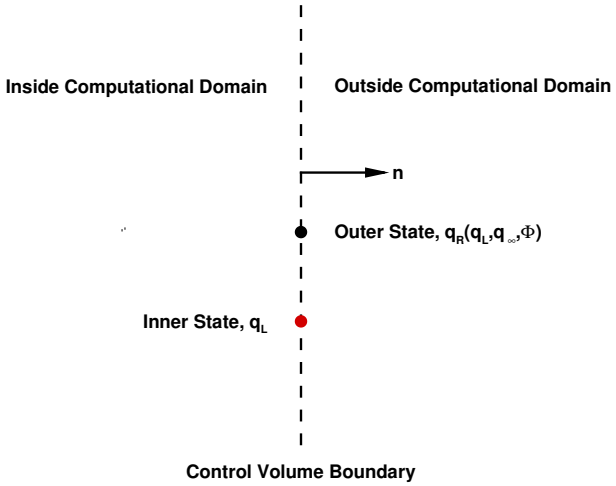


Figure E.1: Nomenclature for Inflow/Outflow Boundaries

E.1 Exact Solution Boundary Condition

Boundary Condition Type	Equation Set	Specify	Extrapolate
Inflow / Outflow	Euler / Navier-Stokes	$\mathbf{q}_R = f_{exact}(\mathbf{x})$	None

The exact solution boundary condition may be used in conjunction with any test problem given in Appendix I. This boundary condition simply uses the exact solution evaluated at a given quadrature point to specify the outer state. The boundary flux is computed using a flux function, the reconstructed inner state, and the specified outer state.

E.2 Farfield Riemann

Boundary Condition Type	Equation Set	Specify	Extrapolate
Inflow / Outflow	Euler / Navier-Stokes	$\mathbf{M}_\infty$, p_∞, T_∞, α, β	ρ_L, p_L, $\mathbf{v}_L$

The farfield Riemann boundary condition selects how to apply the boundary condition based on the local flow direction relative to the domain boundary. First, the velocity normal to the boundary face is computed as

$$\begin{aligned} V_{n,\infty} &= \mathbf{v}_\infty \cdot \hat{\mathbf{n}} \\ V_{n,L} &= \mathbf{v}_L \cdot \hat{\mathbf{n}} \ . \end{aligned} \tag{E.1}$$

Next, the local Riemann invariants, $R^{+/-}$, are computed at the boundary face according to direction and magnitude of the local normal velocity,

$$\begin{aligned} R^{+} &= \begin{cases} V_{n,\infty} + \frac{2a_\infty}{\gamma-1}, & \frac{V_{n,\infty}}{a_\infty} < -1.0 \\ V_{n,L} + \frac{2a_L}{\gamma-1}, & \text{otherwise} \end{cases} \\ R^{-} &= \begin{cases} V_{n,L} - \frac{2a_L}{\gamma-1}, & \frac{V_{n,L}}{a_L} > +1.0 \\ V_{n,\infty} - \frac{2a_\infty}{\gamma-1}, & \text{otherwise} \end{cases} . \end{aligned} \tag{E.2}$$

Using the Riemann invariants, the outer normal velocity and speed of sound may be computed as follows

$$\begin{aligned} V_{n,R} &= \frac{1}{2}\Big(R^{+} + R^{-}\Big) \\ a_R &= \frac{1}{4}(\gamma-1)\Big(R^{+} - R^{-}\Big) . \end{aligned} \tag{E.3}$$

The entropy of the outer state is defined according to the direction of the outer normal velocity,

$$s_R = \begin{cases} \frac{a_L^2}{\gamma \rho_L^{\gamma-1}} & V_{n,R} > 0.0 \\ \frac{a_\infty^2}{\gamma \rho_\infty^{\gamma-1}} & \text{otherwise} \end{cases} . \tag{E.4}$$

Finally, the outer state is computed as

$$\begin{aligned} \rho_R &= \left(\frac{a_R^2}{\gamma s_R}\right)^{\frac{1}{\gamma-1}} \\ \mathbf{v}_R &= \begin{cases} \mathbf{v}_L + (V_{n,R} - V_{n,L})\hat{\mathbf{n}}, & V_{n,R} > 0.0 \\ \mathbf{v}_\infty + (V_{n,R} - V_{n,\infty})\hat{\mathbf{n}}, & \text{otherwise} \end{cases} \\ p_R &= \frac{\rho_R a_R^2}{\gamma} . \end{aligned} \tag{E.5}$$

E.3 Inviscid Wall

Boundary Condition Type	Equation Set	Conditions	Modified Flux
Wall	Euler	$\mathbf{v} \cdot \hat{\mathbf{n}} = 0$	$\mathbb{F}_i(\mathbf{u}) \cdot \hat{\mathbf{n}} = [0 \quad p\hat{\mathbf{n}} \quad 0]^T$

The inviscid wall boundary condition is imposed by requiring no penetration through the surface. Since the flow is assumed to be inviscid, the wall can support a tangential velocity. The only component of the flux that remains is the pressure flux. Static pressure is extrapolated to the wall using the reconstructed interior state.

E.4 Null Boundary Condition

The null boundary condition is only used for 2D simulations. In BlueRidge, 2D simulations are performed using a 2D grid extruded a unit length in the $\hat{k}$-direction. When null boundary condition is encountered, the solver simply cycles to the next boundary face without computing a flux.

E.5 Subsonic Inflow / Outflow

Boundary Condition Type	Equation Set	Specify	Extrapolate
Inflow	Euler / Navier-Stokes	p_0, T_0, α, β	p_L
Outflow	Euler / Navier-Stokes	p_b	ρ_L, $\mathbf{v}_L$

For a subsonic inflow boundary condition, the outer static pressure, p_R, is set using the extrapolated interior state,

$$p_R = p_L \ . \tag{E.6}$$

The stagnation pressure, p_0, and stagnation temperature, T_0, are used to compute the outer Mach number and static temperature as follows

$$\begin{aligned} M_R &= \sqrt{\left[\left(\frac{p_R}{p_0}\right)^{-\frac{\gamma-1}{\gamma}} - 1\right]\left(\frac{2}{\gamma-1}\right)} \\ T_R &= T_0\left(\frac{p_R}{p_0}\right)^{\frac{\gamma-1}{\gamma}} \ . \end{aligned} \tag{E.7}$$

Using the equation of state for a perfect gas, the outer density is given by

$$\rho_R = \frac{p_R}{RT_R} \tag{E.8}$$

where R is the specific gas constant of the fluid. The outer velocity vector is set using the angle of attack, α, and side-slip angle, β, as follows

$$\begin{aligned} \|\mathbf{v}_R\| &= M_R\sqrt{\frac{\gamma p_R}{\rho_R}} \\ \hat{\mathbf{v}} &= \hat{\mathbf{v}}(\alpha,\beta) \\ \mathbf{v}_R &= \|\mathbf{v}_R\|\hat{\mathbf{v}} \end{aligned} \tag{E.9}$$

where $\|\mathbf{v}_R\|$ is the velocity magnitude and $\hat{\mathbf{v}}$ is the local flow direction. If locally reversed or supersonic flow are encountered during initial transients, the outer state is set to the stagnation conditions in an attempt to correct the behavior of the solution.

At a subsonic outflow, density and velocity are extrapolated to the boundary face using the interior reconstruction. The pressure at the outflow, p_b, is prescribed. Therefore, the outer state may be written as

$$\mathbf{q}_R = \begin{bmatrix} \rho_L & \mathbf{v}_L & p_b \end{bmatrix}^T . \tag{E.10}$$

If locally supersonic flow is encountered during initial transients, the outflow pressure is set to the interior pressure to allow pressure waves to exit the computational domain. Likewise, if locally reversed flow is encountered, the direction of the velocity vector is flipped in an attempt to correct the flow direction.

E.6 Supersonic Inflow / Outflow

Boundary Condition Type	Equation Set	Specify	Extrapolate
Inflow	Euler / Navier-Stokes	ρ_∞, p_∞, $\mathbf{v}_\infty$	None
Outflow	Euler / Navier-Stokes	None	ρ_L, p_L, $\mathbf{v}_L$

At a supersonic inflow, all variables are prescribed by the freestream conditions. Therefore, the outer state is defined as

$$\mathbf{q}_R = \begin{bmatrix} \rho_\infty & \mathbf{v}_\infty & p_\infty \end{bmatrix}^T . \tag{E.11}$$

On the other hand, at a supersonic outflow, all outer state variables are set using the interior reconstruction. Therefore, the outer state is defined as

$$\mathbf{q}_R = \begin{bmatrix} \rho_L & \mathbf{v}_L & p_L \end{bmatrix}^T . \tag{E.12}$$

E.7 Symmetry Plane

Boundary Condition Type	Equation Set	Conditions	Modified Flux
Wall	Euler / Navier-Stokes	$\mathbf{v} \cdot \hat{\mathbf{n}} = 0$	$\mathbb{F}_i(\mathbf{u}) \cdot \hat{\mathbf{n}} = [0 \quad p\hat{\mathbf{n}} \quad 0]^T$
		$\nabla\phi \cdot \hat{\mathbf{n}} = 0$	$\mathbb{F}_v(\mathbf{u}, \nabla\mathbf{u}) = [0 \quad \boldsymbol{\tau} \quad \boldsymbol{\tau}\mathbf{v}]^T$

At a symmetry plane, no flow is permitted in the direction normal to the boundary face. In a similar manner, the gradient of all scalar quantities, ϕ, normal to the boundary face are set to zero. The modified inviscid and viscous fluxes may be found in the above table. The reader should note that terms in the stress tensor, $\boldsymbol{\tau}$, also cancel to zero. However, for clarity and ease of notation, the stress tensor has been explicitly left in the viscous flux. For more information on the symmetry plane boundary condition, refer to Ref. [3].

E.8 Adiabatic, Viscous Wall

Boundary Condition Type	Equation Set	Conditions	Modified Flux
Wall	Navier-Stokes	$\mathbf{v} = \mathbf{0}$ $k\nabla T = \mathbf{0}$	$\mathbb{F}_i(\mathbf{u}) = [0 \quad p\mathbf{I} \quad 0]^T$ $\mathbb{F}_v(\mathbf{u}, \nabla\mathbf{u}) = [0 \quad \boldsymbol{\tau} \quad 0]^T$

At an adiabatic, viscous wall, the velocity stagnates at the surface, and no heat transfer is permitted. Consequently, the remaining contributions to the flux are those from the wall pressure and viscous stress. Static pressure is extrapolated to the wall using the interior reconstruction.

E.9 Isothermal, Viscous Wall

Boundary Condition Type	Equation Set	Conditions	Modified Flux
Wall	Navier-Stokes	$\mathbf{v} = \mathbf{0}$ $\mu = \mu(T_w)$ $k = k(T_w)$	$\mathbb{F}_i(\mathbf{u}) = [0 \quad p\mathbf{I} \quad 0]^T$ $\mathbb{F}_v(\mathbf{u}, \nabla\mathbf{u}) = [0 \quad \boldsymbol{\tau} \quad k\nabla T]^T$

At an isothermal, viscous wall, a wall temperature, T_w, is imposed in a weak sense through the viscosity and thermal conductivity at the wall. Similar to an adiabatic, viscous wall, the velocity stagnates at the surface. The only contribution to the inviscid flux is from the wall pressure. The viscous flux has contributions from the viscous stress and heat conduction terms.

Bibliography

[1] J. Carlson, Inflow - Outflow boundary conditions with application to FUN3D, Tech. Rep. TM–2011-217181, NASA (October 2011).

[2] C. Hirsch, Numerical Computation of Internal and External Flows: Volume 2, John Wiley & Sons Ltd., 1990.

[3] J. Blazek, Computational Fluid Dynamics - Principles and Applications, 2nd Edition, Elsevier Science Publishing Co., Inc., 2001.

Appendix F

Boundary Conditions for Model Problem Subclasses

All `model_problem_t` boundary conditions are enforced weakly through the flux. At present, only Dirichlet and Neumann boundary conditions are implemented. These boundary conditions are imposed via a modified analytic flux. In subsequent sections, each boundary condition is described in detail. Accompanying each description is a table indicating the type of boundary condition, the compatible equation set, and the form of the modified flux.

F.1 Dirichlet Boundary Condition

Boundary Condition Type	Equation Set	Conditions	Modified Flux
Dirichlet	Laplace / Burgers'	$\mathbf{q}_F = \mathbf{u}_{set}$	$\mathbb{F}_i(\mathbf{u}) = \mathbb{F}_i(\mathbf{u}_{set})$ $\mathbb{F}_v(\mathbf{u}, \nabla\mathbf{u}) = \mathbb{F}_v(\mathbf{u}_{set}, \nabla\mathbf{u})$

The Dirichlet boundary condition explicitly sets the value of state variable at the boundary face. For equation sets whose analytic flux does not depend on the state variable (e.g., Laplace's equation), Dirichlet boundary conditions are enforced using Nitsche's method [1]. With Nitsche's method, a term is added to the boundary residual to penalize any boundary state that does not match the specified state. This additional boundary term takes the following form

$$\mathcal{R}_{Nitsche} = \gamma \frac{\nu}{L_{eff}} \oint_{\partial\Omega_i} (u - u_{set})\, dS, \quad \gamma > 0 \tag{F.1}$$

where L_{eff} is a characteristic length of the grid, ν is a viscosity, and γ is a positive constant used to emphasize the Nitsche contribution to the boundary residual.

F.2 Neumann Boundary Condition

Boundary Condition Type	Equation Set	Conditions	Modified Flux
Neumann	Laplace / Burgers'	$\nabla \mathbf{q}_F = \nabla \mathbf{u}_{set}$	$\mathbb{F}_v(\mathbf{u}, \nabla \mathbf{u}) = \mathbb{F}_v(\mathbf{u}, \nabla \mathbf{u}_{set})$

The Neumann boundary condition explicitly sets the gradient of the state variable at the boundary face. The actual value of the state variable is determined from the interior reconstruction.

Bibliography

[1] J. Nitsche, Über ein Variations zur Lösung von Dirichlet-Problemen bei Verwendung von Teilräumen die keinen Randbedingungen unterworfen sind, Abhandlungen aus dem mathematischen Seminar der Universität Hamburg 36 (1971) 9 – 15.

Appendix G

Nondimensionalization of Physics Subclasses

When solving PDEs, it is common for variables to be of vastly different orders of magnitudes. In fluid mechanics problems, for example, density and pressure are typically $O(1)$ and $O(10^5)$, respectively. If an equation set is solved in a dimensional form, the resulting numerical problem can be poorly conditioned, and significant round-off errors can accumulate. To avoid these issues, PDEs are typically solved in a nondimensional form. The process of nondimensionalization scales all solution variables to roughly the same order of magnitude. The nondimensionalization used in BlueRidge for each `physics_t` subclass is summarized in Table G.1 - Table G.5. The notation $(\cdot)^*$ refers to a dimensional quantity, the notation $(\cdot)$ refers to a nondimensional quantity, and the notation $(\cdot)_{ref}$ refers to a reference quantity.

Table G.1: Description of `fluid_mechanics_t` reference variables.

Variable	Description	Value
ρ^*_{ref}	Density	Code Input
T^*_{ref}	Temperature	Code Input
a^*_{ref}	Speed of Sound	Code Input
L^*_{ref}	Length	Code Input
L_{ref}	Length	Code Input
$\lvert\mathbf{v}\rvert^*_{ref}$	Velocity Magnitude	Code Input
R^*_{ref}	Specific Gas Constant	$R^*_{ref} = \frac{a^{*2}_{ref}}{T^*_{ref}}$
μ^*_{ref}	Viscosity	$\mu^*_{ref} = \mu^*\left(T^*_{ref}\right)$
$Re_{L_{ref}}$	Reynolds Number	$Re_{L_{ref}} = \frac{\rho^*_{ref}\lvert\mathbf{v}\rvert^*_{ref}\left(L^*_{ref}/L_{ref}\right)}{\mu^*_{ref}}$
M_{ref}	Mach Number	$M_{ref} = \frac{\lvert\mathbf{v}\rvert^*_{ref}}{a^*_{ref}}$

Table G.2: Nondimensionalization of `fluid_mechanics_t` variables.

Variable	Description	Nondimensionalization
$\mathbf{x}$	Cartesian Coordinates	$\mathbf{x}^* = \mathbf{x}\left(\frac{L^*_{ref}}{L_{ref}}\right)$
t	Time	$t^* = t\left[\frac{1}{a^*_{ref}}\left(\frac{L^*_{ref}}{L_{ref}}\right)\right]$
ρ	Static Density	$\rho^* = \rho\left(\rho^*_{ref}\right)$
p	Static Pressure	$p^* = p\left(\rho^*_{ref} a^{*2}_{ref}\right)$
T	Static Temperature	$T^* = T\left(T^*_{ref}\right)$
$\mathbf{v}$	Velocity Vector	$\mathbf{v}^* = \mathbf{v}\left(a^*_{ref}\right)$
e_t	Total Energy per unit mass	$e^*_t = e_t\left(a^{*2}_{ref}\right)$
h_t	Total Enthalpy per unit mass	$h^*_t = h_t\left(a^{*2}_{ref}\right)$
a	Speed of Sound	$a^* = a\left(a^*_{ref}\right)$
k	Thermal Conductivity	$k^* = k\left(\mu^*_{ref}\frac{a^{*2}_{ref}}{T^*_{Ref}}\right)$
R	Specific Gas Constant	$R^* = R\left(R^*_{ref}\right)$
μ	Dynamic Viscosity	$\mu^* = \mu\left(\mu^*_{ref}\right)$

Table G.3: Nondimensionalization of `fluid_mechanics_t` auxiliary equations.

Auxiliary Equation	Dimensional Form	Nondimensional Form
Equation of State	$p^* = \rho^* R^* T^*$	$p = \rho R T$
Speed of Sound	$a^{*2} = \gamma R^* T^*$	$a^2 = \gamma R T$
Sutherland's Law	$\mu^* = \mu^*_S\left(\frac{T^*}{T^*_S}\right)^{\frac{3}{2}}\frac{T^*_S+S^*}{T^*+S^*}$	$\mu = \mu_S\left(\frac{T}{T_S}\right)^{\frac{3}{2}}\frac{T_S+S}{T+S}$
Viscous Stresses	$\boldsymbol{\tau}^* = \mu^*\left[\nabla\mathbf{v}^* + (\nabla\mathbf{v}^*)^T\right] - \frac{2}{3}\mu^*(\nabla\cdot\mathbf{v}^*)\mathbf{I}$	$\boldsymbol{\tau}^* = \mu^*_{ref}a^*_{ref}\left(\frac{L_{ref}}{L^*_{ref}}\right)\boldsymbol{\tau}$

Table G.4: Description of `model_problem_t` reference variables.

Variable	Description	Value
u^*_{ref}	Model Problem Variable	Code Input
ν^*_{ref}	Viscosity	Code Input
L^*_{ref}	Length	Code Input
L_{ref}	Length	Code Input

Table G.5: Nondimensionalization of `model_problem_t` variables.

Variable	Description	Nondimensionalization
$\mathbf{x}$	Cartesian Coordinates	$\mathbf{x}^* = \mathbf{x}\left(\frac{L^*_{ref}}{L_{ref}}\right)$
t	Time	$t^* = t\left[\frac{1}{u^*_{ref}}\left(\frac{L^*_{ref}}{L_{ref}}\right)\right]$
u	Model Problem Variable	$u^* = u\left(u^*_{ref}\right)$
ν	Viscosity	$\nu^* = \nu\left(\nu^*_{ref}\right)$

Appendix H

CFL Evolution Schemes

CFL evolution schemes are designed to accelerate convergence toward a steady-state. While these schemes provide automatic control of the CFL number, they can sometimes initiate a feedback loop where the interaction between the time integrator and the CFL evolution scheme stagnates convergence. Therefore, when applying a given CFL evolution scheme, the user must take special care to ensure proper convergence to the steady-state. In the author's experience, this feedback loop can typically be broken by freezing the CFL number. Most CFL evolution schemes are heuristic formulas based on residual norms or CFL numbers from previous time steps. Several CFL evolution schemes are implemented in BlueRidge and are described in detail in the following sections.

H.1 Switched Evolution Relaxation (SER)

Switched evolution relaxation (SER) [1, 2, 3, 4] is one of the most common CFL evolution schemes. SER advances the time step according to the change in the residual norm between time steps n and $n-1$. The CFL number is limited from above by the maximum theoretical CFL number for the time integrator, $\mathrm{CFL}_{\max}$. The CFL number at time step n is computed according to the following

$$\mathrm{CFL}^n = \min\left(\mathrm{CFL}^{n-1}\frac{\|\mathbf{R}^{n-1}\|}{\|\mathbf{R}^n\|}, \mathrm{CFL}_{\max}\right). \tag{H.1}$$

H.2 Exponential Progression with Under-Relaxation (EXPur)

Exponential progression with under-relaxation (EXPur) [4] advances the CFL number using the CFL number at time step $n-1$ and the under-relaxation factor, ω. When the full update step is taken by the time integrator, the CFL number is increased by a constant factor, β. Similarly, when $\omega < \omega_{min}$, the CFL number is reduced by a constant factor, κ. Otherwise, the CFL number is held constant. The CFL number at time step n is computed as follows

$$\mathrm{CFL}^n = \begin{cases} \beta \cdot \mathrm{CFL}^{n-1}, & \omega^{n-1} = 1 \quad (\beta > 1) \\ \mathrm{CFL}^{n-1}, & \omega_{min} < \omega^{n-1} < 1 \\ \kappa \cdot \mathrm{CFL}^{n-1}, & \omega^{n-1} < \omega_{min} \quad (\kappa < 1) \end{cases} \tag{H.2}$$

where $\beta = 1.05$, $\kappa = 0.10$, and $\omega_{min} = 0.01$.

H.3 Residual Difference Method (RDM)

The residual difference method (RDM) [2, 4] advances the CFL number according to the relative change in the residual norm. Similar to SER, the CFL number is limited from above by the maximum theoretical CFL number for the time integrator. From time step $n-1$ to n, the change in the CFL number is limited by a constant factor, β. The CFL number at time step n is computed as

$$\mathrm{CFL}^n = \min\left(\mathrm{CFL}^{n-1} \cdot \beta^{\frac{\left\|\mathbf{R}^{n-1}\right\| - \|\mathbf{R}^n\|}{\|\mathbf{R}^{n-1}\|}}, \mathrm{CFL}_{\max}\right), \quad \beta > 1 \tag{H.3}$$

where $\beta = 2.0$.

H.4 Monotonic Residual Difference Method (mRDM)

The monotonic residual difference method (mRDM) [4] is identical to RDM when residual norms decrease from time step $n-1$ to n. When residual norms increase, the CFL number is fixed at its previous value. The CFL number at time step n is computed as follows

$$\begin{aligned} \mathrm{CFL}^n &= \min\left(\mathrm{CFL}^{n-1} \cdot \beta^{\gamma}, \mathrm{CFL}_{\max}\right), \quad \beta > 1 \\ \gamma &= \begin{cases} \frac{\left\|\mathbf{R}^{n-1}\right\| - \|\mathbf{R}^n\|}{\|\mathbf{R}^{n-1}\|}, & \|\mathbf{R}^n\| \le \|\mathbf{R}^{n-1}\| \\ 0, & \|\mathbf{R}^n\| > \|\mathbf{R}^{n-1}\| \end{cases}. \end{aligned} \tag{H.4}$$

Bibliography

[1] W. A. Mulder, B. Van Leer, Experiments with implicit upwind methods for the Euler equations, Journal of Computational Physics 59 (2) (1985) 232 – 246. doi:10.1016/0021-9991(85)90144-5.
URL https://doi.org/10.1016/0021-9991(85)90144-5

[2] H. M. Bücker, B. Pollul, A. Rasch, On CFL evolution strategies for implicit upwind methods in linearized Euler equations, International Journal for Numerical Methods in Fluids 59 (1) (2009) 1 – 18. doi:10.1002/fld.1798.
URL https://doi.org/10.1002/fld.1798

[3] C. T. Kelley, D. E. Keyes, Convergence analysis of pseudo-transient continuation, SIAM Journal on Numerical Analysis 35 (2) (1998) 508 – 523. doi:10.1137/S0036142996304796.
URL https://doi.org/10.1137/S0036142996304796

[4] M. Ceze, A robust hp-adaptation method for discontinuous galerkin discretizations applied to aerodynamic flows, Ph.D. thesis, University of Michigan (2013).
URL http://hdl.handle.net/2027.42/97795

Appendix I

Suite of Exact and Manufactured Solutions

A suite of exact and manufactured solutions is included in the BlueRidge code base to provide a test bed for code verification and ongoing research efforts. Test problems are available for each equation set. Python scripts are include for select problems to help users set up their problem of interest. Each Python script generates a namelist input file for the solver as well as a CGNS file containing the grid and boundary conditions. In the subsequent sections, the test problems included in BlueRidge are described in detail. Each section includes the following information:

1. A brief overview of the problem which includes the problem name, the type of solution (exact or manufactured), the applicable equation set, and the status of a Python script.
2. A short description of problem.
3. A table indicating the namelist inputs for the problem and their default values.
4. A figure depicting the boundary conditions.
5. A figure depicting the exact or manufactured solution.

I.1 Diamond Airfoil

Overview

- Case Name: "diamond_airfoil" (`diamond_airfoil_t`)
- Solution Type: Exact
- Equation Set: Euler (`euler_t`)
- Python Script: Y

Description

The `diamond_airfoil_t` subclass is used to simulate supersonic flow over a diamond airfoil at a 0° angle of attack. The flowfield is characterized by shock waves which emanate from the leading and trailing edges of the airfoil and an expansion fan which emanates from the top of the airfoil. The exact solution is computed using the Rankine-Hugoniot conditions and the method of characteristics [1]. To ensure the exact solution remains defined everywhere in the domain, users must carefully select the location of domain boundaries, the inflow Mach number, and the airfoil half-angle to prevent the intersection of the shock waves and expansion fan.

Table I.1: Namelist variables.

Variable Name	Variable Description	Default Value
diamond_wedge_angle	Airfoil Half-Angle	15.0°
diamond_LE_xy	Position of LE	$[0.0, 0.0]$ m
diamond_mid_xy	Position of Mid-chord	$[0.5, 0.5 \times \text{tand}(15°)]$ m
diamond_TE_xy	Position of TE	$[1.0, 0.0]$ m
diamond_p1	Inflow Pressure	1.0E+05 Pa
diamond_T1	Inflow Temperature	300.0 K
diamond_M1	Inflow Mach Number	1.75

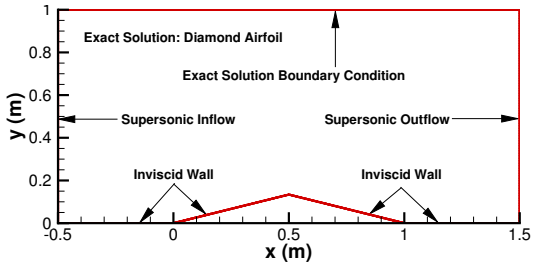


Figure I.1: Boundary conditions.

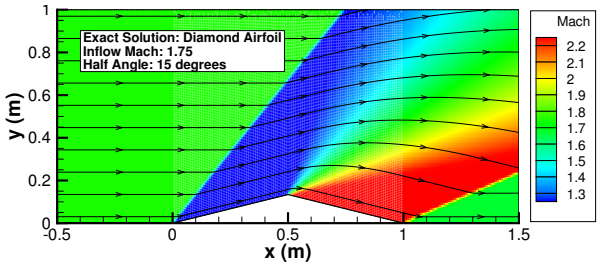


Figure I.2: Exact solution.

I.2 Expansion Fan

Overview

- Case Name: "expansion_fan" (`expansion_fan_t`)
- Solution Type: Exact
- Equation Set: Euler (`euler_t`)
- Python Script: N

Description

The `expansion_fan_t` subclass is used to simulate supersonic flow around a downward turn. The flow accelerates isentropically creating an expansion fan which emanates from the corner of the turn. The exact solution is determined everywhere in the domain using the method of characteristics [1].

Table I.2: Namelist variables.

Variable Name	Variable Description	Default Value
expfan_turn_angle	Turn Angle	12.0°
expfan_origin	Origin of Expansion Fan	$[0.0, 0.0]$ m
expfan_p1	Inflow Pressure	1.01325E+05 Pa
expfan_T1	Inflow Temperature	273.16 K
expfan_M1	Inflow Mach Number	1.20

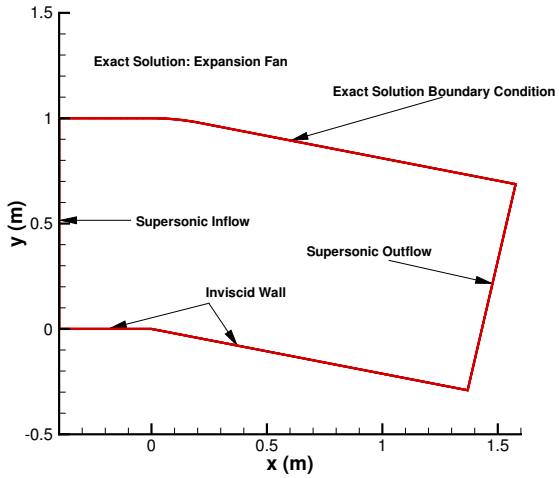


Figure I.3: Boundary conditions.

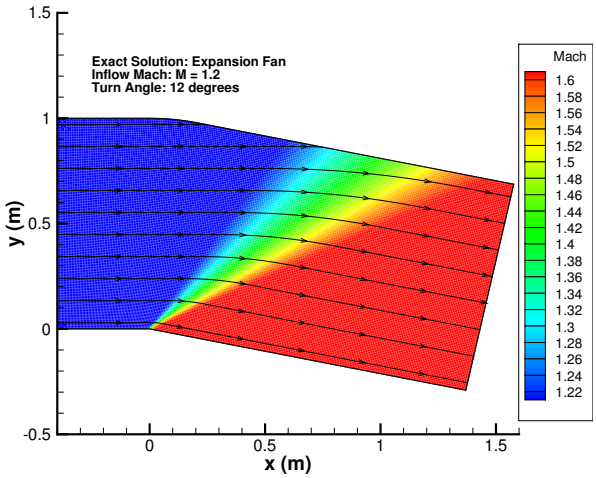


Figure I.4: Exact solution.

I.3 Oblique Shock

Overview

- Case Name: "oblique_shock" (`oblique_shock_t`)
- Solution Type: Exact
- Equation Set: Euler (`euler_t`)
- Python Script: N

Description

The `oblique_shock_t` subclass is used to simulate supersonic flow around a sharp compression turn. An oblique shock wave emanates from the corner of the turn. The flow is isentropic on either side of the shock wave, but exhibits an entropy increase across the shock. The jump in the solution across the shock is determined using the Rankine-Hugoniot conditions [1]. When altering namelist defaults, users must ensure that the shock is attached so that the exact solution remains defined.

Table I.3: Namelist variables.

Variable Name	Variable Description	Default Value
shock_turn_angle	Turn Angle	12.0°
shock_origin	Origin of Shock Wave	$[0.0, 0.0]$ m
shock_p1	Inflow Pressure	1.0E+05 Pa
shock_T1	Inflow Temperature	300.0 K
shock_M1	Inflow Mach Number	2.0

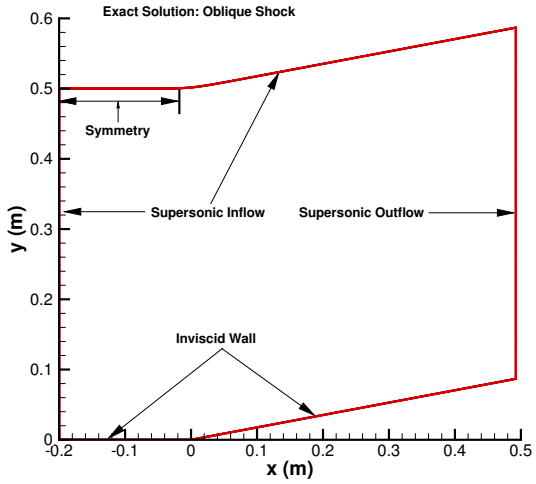


Figure I.5: Boundary conditions.

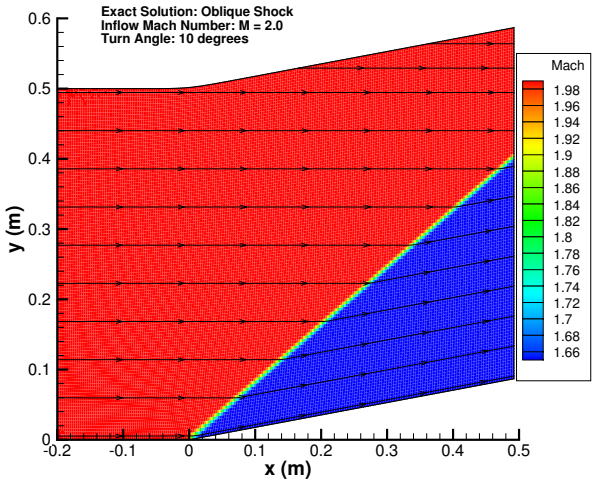


Figure I.6: Exact solution.

I.4 Ringleb Flow

Overview

- Case Name: "ringleb" (`ringleb_t`)
- Solution Type: Exact
- Equation Set: Euler (`euler_t`)
- Python Script: Y

Description

The `ringleb_t` subclass is used to simulate isentropic flow between two curved walls. Subsonic flow enters the domain and accelerates isentropically to supersonic near the inner wall and transonic near the outer wall. Since the exact solution is symmetric about $y = 0.0$, only the top half of the domain is simulated. The exact solution may be computed everywhere in the domain using the hodograph plane transformation [2, 3].

Table I.4: Namelist variables.

Variable Name	Variable Description	Default Value
ringleb_V_min	Inflow Velocity	0.50
ringleb_psi_min	Minimum Stream Function	0.69
ringleb_psi_max	Maximum Stream Function	1.20

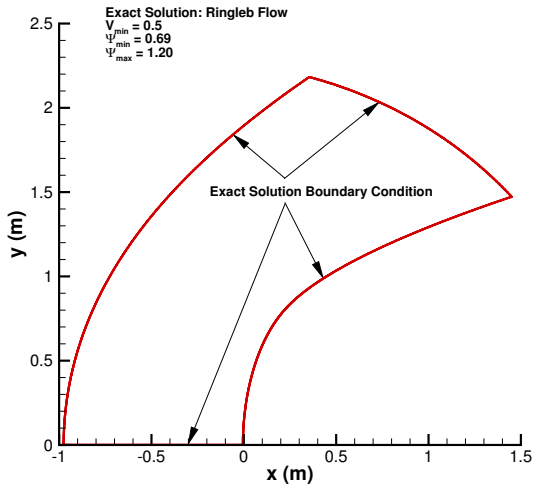


Figure I.7: Boundary conditions.

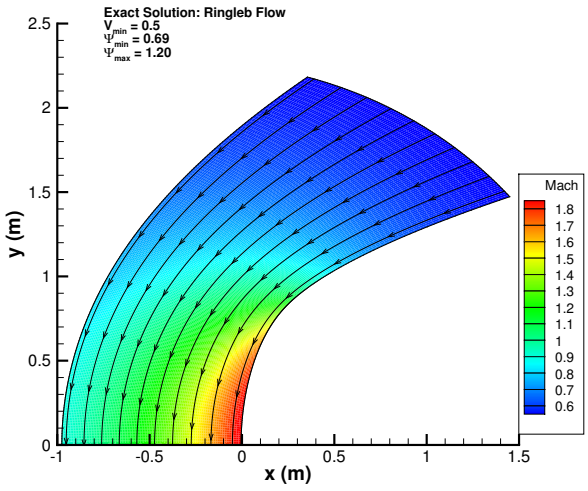


Figure I.8: Exact solution.

I.5 Supersonic Vortex Flow

Overview

- Case Name: "svf" (`svf_t`)
- Solution Type: Exact
- Equation Set: Euler (`euler_t`)
- Python Script: Y

Description

The `svf_t` subclass is used to simulate a stationary supersonic vortex. The exact solution is isentropic everywhere in the domain and is characterized by concentric circles of constant Mach number flow. Supersonic flow enters the domain at the top left boundary and exits the domain at the bottom right boundary. The inner and outer streamlines of the flow are modeled using inviscid walls. For more information regarding this exact solution, refer to Ref. [4].

Table I.5: Namelist variables.

Variable Name	Variable Description	Default Value
svf_rho_inner	Density at Inner Radius	1.0 kg/m^3
svf_Mach_inner	Mach Number at Inner Radius	2.0
svf_radius_inner	Inner Radius	2.0 m

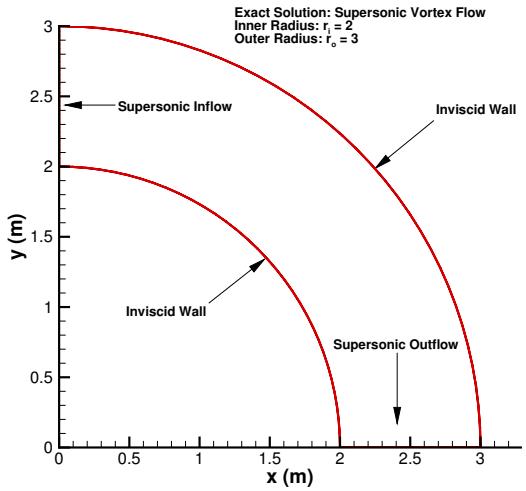


Figure I.9: Boundary conditions.

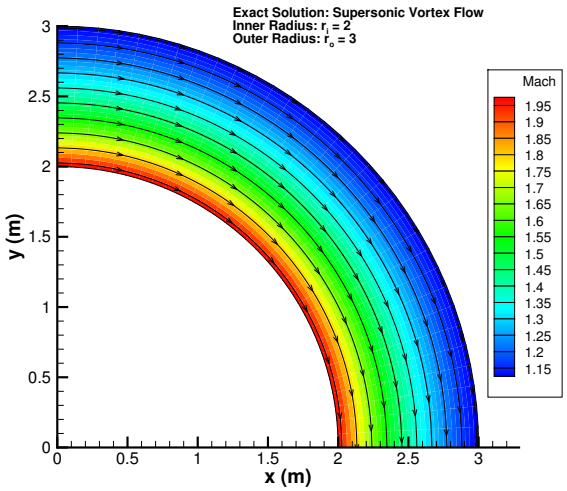


Figure I.10: Exact solution.

I.6 Cross-term Sinusoidal

Overview

- Case Name: "cts_subsonic"/"cts_supersonic" (`cts_t`)
- Solution Type: Manufactured
- Equation Set: Euler/Navier-Stokes (`euler_t`, `navier_stokes_t`)
- Python Script: Y

Description
The `cts_t` subclass is a manufactured solution designed for code verification. The subclass implements both subsonic and supersonic solutions and may be used with either the `euler_t` equation set or the `navier_stokes_t` equation set. The form of the manufactured solution is given by Eq. 6.49. The coefficients and functions used are given in Table I.6 - Table I.8. For more information, refer to Ref. [5].

Table I.6: MMS amplitude coefficients.

Variable, ϕ	a_0	a_x	a_y	a_z	a_{xy}	a_{yz}	a_{xz}
ρ	1.0	0.15	-0.10	0.1	0.08	0.05	0.12
u	70 (800)	5 (50)	-15 (-150)	-10 (-100)	7 (70)	4 (40)	-4 (-40)
v	90 (800)	-5 (-50)	10 (100)	5 (50)	-11 (-110)	-5 (-50)	5 (50)
w	80 (800)	-10 (-100)	10 (100)	12 (120)	-12 (-120)	11 (110)	5 (50)
p ($\times 10^5$)	1.0	0.2	0.5	0.2	-0.25	-0.1	0.1

Table I.7: MMS functions.

Variable, ϕ	f_x	f_y	f_z	f_{xy}	f_{yz}	f_{xz}
ρ	cos	sin	sin	cos	sin	cos
u	sin	cos	cos	cos	sin	cos
v	sin	cos	cos	cos	sin	cos
w	cos	sin	cos	sin	sin	cos
p	cos	cos	sin	cos	sin	cos

Table I.8: MMS frequency coefficients.

Variable, ϕ	b_x	b_y	b_z	b_{xy}	b_{yz}	b_{xz}
ρ	0.75	0.45	0.80	0.65	0.75	0.50
u	0.50	0.85	0.40	0.60	0.80	0.90
v	0.80	0.80	0.50	0.90	0.40	0.60
w	0.85	0.90	0.50	0.40	0.80	0.75
p	0.40	0.45	0.85	0.75	0.70	0.80

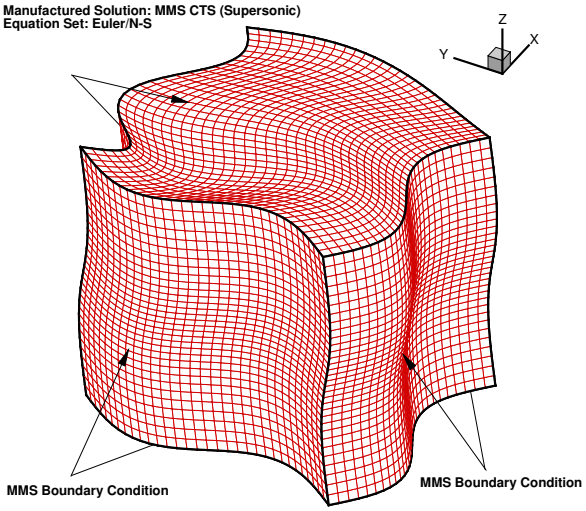


Figure I.11: Boundary conditions.

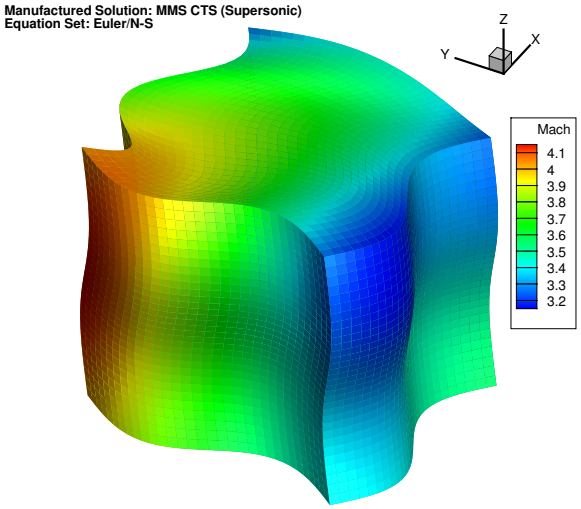


Figure I.12: Manufactured solution.

I.7 Laminar Boundary Layer

Overview

- Case Name: "laminar_bl" (`lbl_t`)
- Solution Type: Manufactured
- Equation Set: Navier-Stokes (`navier_stokes_t`)
- Python Script: Y

Description

The `lbl_t` subclass is used to simulate a zero-pressure gradient boundary layer over a flat plate. The flow stagnates at the plate and increases toward the freestream velocity with distance from the wall. Starting from the leading edge of the plate, users have the option to apply heating over a length, x_T. The scale of Figure I.14 has been enlarged to better illustrate the variation of the solution. The solution is determined by solving the integral energy equation with the assumption that the velocity and temperature profiles above the plate behave as polynomials. For more information, refer to Refs. [6, 7].

Table I.9: Namelist variables.

Variable Name	Variable Description	Default Value
lbl_x0	Starting Location of Plate	0.0 m
lbl_xT	Starting Location of Heating	0.0 m
lbl_Tw	Wall Temperature	350.0 K
lbl_Ue	Edge Velocity	35.0 m/s
lbl_Te	Edge Temperature	350.0 K
lbl_pe	Edge Pressure	1.01325E+05 Pa

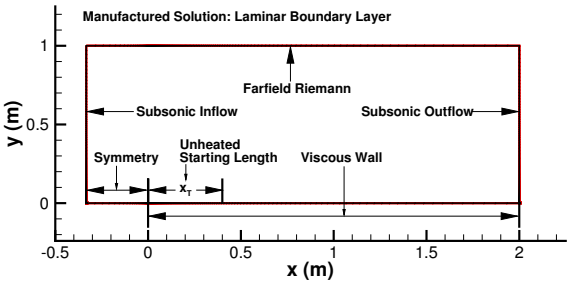


Figure I.13: Boundary conditions.

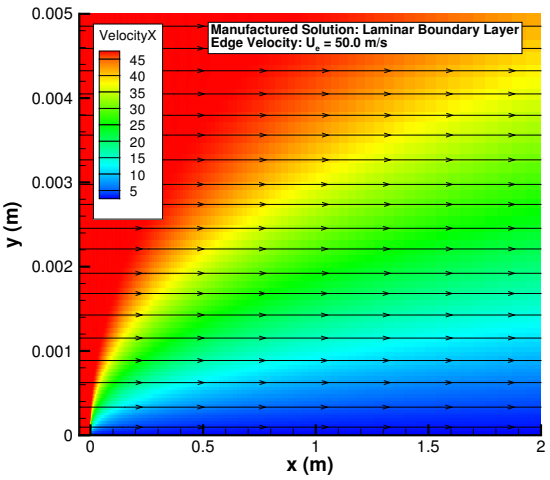


Figure I.14: Manufactured solution.

I.8 Potential Cylinder

Overview

- Case Name: "potential_cylinder" (`potential_cylinder_t`)
- Solution Type: Manufactured
- Equation Set: Euler (`euler_t`)
- Python Script: Y

Description

The `potential_cylinder_t` subclass is used to simulate subsonic, inviscid flow over a circular cylinder. The flow is symmetric with stagnation points at the front and rear of the cylinder and with regions of accelerated flow on the top and bottom of the cylinder. This problem is treated as a manufactured solution in the event that users want to run at a higher Mach number. If the inflow Mach number is chosen low enough, this problem can be treated as an exact solution. However, users should keep in mind that numerical dissipation can often cause non-physical vortex shedding. The solution is determined everywhere in the domain using potential flow theory. For more information, refer to Ref. [8].

Table I.10: Namelist variables.

Variable Name	Variable Description	Default Value
cylinder_rho_inf	Freestream Density	1.0 kg/m^3
cylinder_p_inf	Freestream Pressure	1.0E+05 Pa
cylinder_V_inf	Freestream Velocity	35.0 m/s
cylinder_AOA	Angle of Attack	0.0°

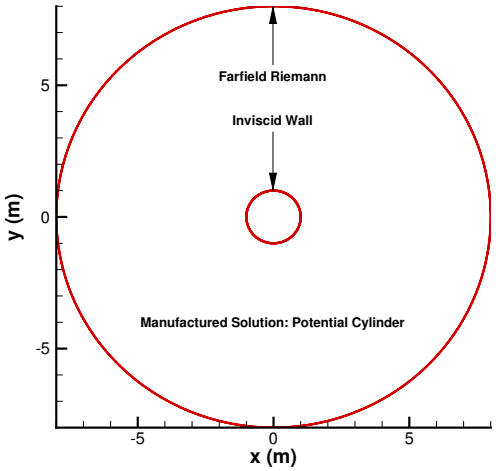


Figure I.15: Boundary conditions.

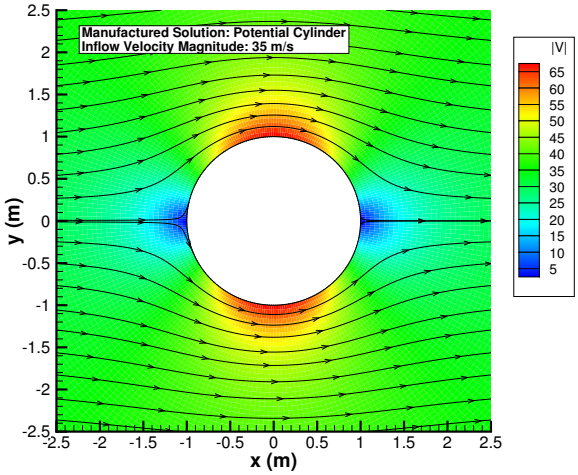


Figure I.16: Manufactured solution.

I.9 Mapped Airfoil

Overview

- Case Name: "mapped_airfoil" (`mapped_airfoil_t`)
- Solution Type: Manufactured
- Equation Set: Euler (`euler_t`)
- Python Script: Y

Description

The `mapped_airfoil_t` subclass is used to simulate subsonic flow over an airfoil. The solution is determined everywhere in the domain by applying a conformal map to the `potential_cylinder_t` solution. The available mappings include the Joukowsky transform and the Karman-Trefftz transform. Similar to the `potential_cylinder_t` subclass, this problem is treated as manufactured solution in case users would like to run at higher Mach numbers. For more information, refer to Ref. [8].

Table I.11: Namelist variables.

Variable Name	Variable Description	Default Value
airfoil_rho_inf	Pressure Density	1.0 kg/m^3
airfoil_p_inf	Freestream Pressure	1.0E+05 Pa
airfoil_V_inf	Freestream Velocity	35.0 m/s
airfoil_AOA	Angle of Attack	0.0°
airfoil_z1_xy	Center of Mapped Circle	$(-0.1 + 0.0i)$
airfoil_which_mapping	Airfoil Mapping	"Joukowsky"

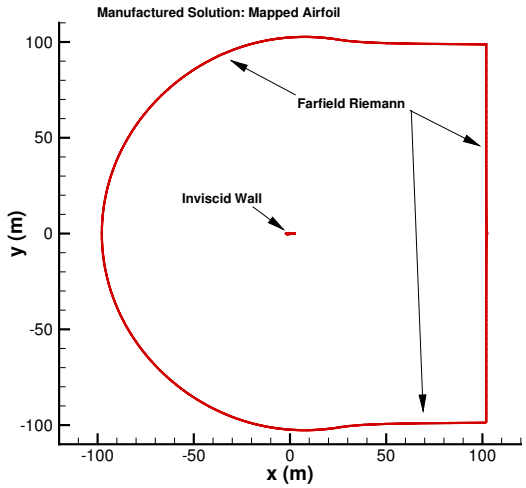


Figure I.17: Boundary conditions.

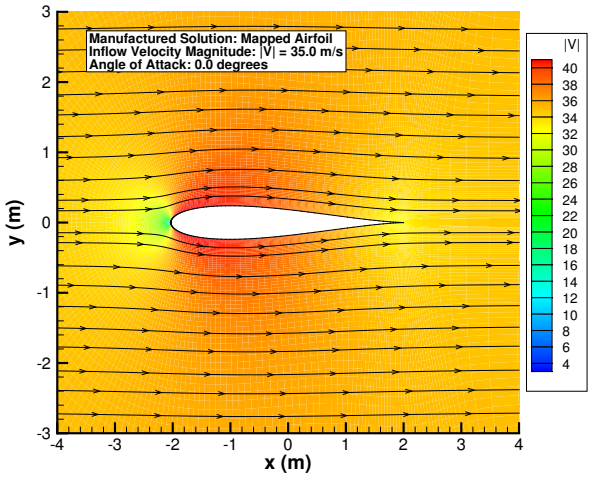


Figure I.18: Manufactured solution.

I.10 Gaussian Bump

Overview

- Case Name: "gaussian_bump_subsonic"/
- "gaussian_bump_supersonic" (`gauss_bump_t`)
- Solution Type: Manufactured
- Equation Set: Euler/Navier-Stokes (`euler_t`, `navier_stokes_t`)
- Python Script: Y

Description

The `gauss_bump_t` subclass is a manufactured solution designed for code verification and evaluating discretization error estimators. The subclass implements both subsonic and supersonic solutions and may be used with either the `euler_t` equation set or the `navier_stokes_t` equation set. The manufactured solution is a Gaussian bump of the following form

$$\phi(\mathbf{x}) = \alpha_0 + \alpha_1 \exp\Big[-\Big[a_1(x-x_0)^2 + 2b_1(x-x_0)(y-y_0) + c_1(y-y_0)^2\Big]\Big] \tag{I.1}$$

where $[x_0, y_0]$ is the center of the bump and the coefficients a_1, b_1, c_1 are given by

$$a_1 = +\frac{\cos^2(\theta)}{2\sigma_x^2} + \frac{\sin^2(\theta)}{2\sigma_y^2}, \quad b_1 = -\frac{\sin(2\theta)}{4\sigma_x^2} + \frac{\sin(2\theta)}{4\sigma_y^2}, \quad c_1 = +\frac{\sin^2(\theta)}{2\sigma_x^2} + \frac{\cos^2(\theta)}{2\sigma_y^2}\ . \tag{I.2}$$

The terms θ, σ_x, and σ_y define the size and orientation of the bump. The coefficients α_0 and α_1 are defined for each variable in Table I.12. Default values for θ, σ_x, σ_y, and $[x_0, y_0]$ are given in Table I.13.

Table I.12: Coefficients for the Gaussian bump manufactured solution.

α \ ϕ	ρ	u	v	p
α_0	1.0	0.5 (2.0)	0.0	2.0
α_1	0.1	0.1	1.0	0.1

Table I.13: Namelist variables.

Variable Name	Variable Description	Default Value
gauss_bump_theta	Angle of the Bump	$\pi/4$
gauss_bump_sigma_x	Standard Deviation in X	0.075
gauss_bump_sigma_y	Standard Deviation in Y	0.050
gauss_bump_center	Center of the Bump	$[0.5, 0.5]$

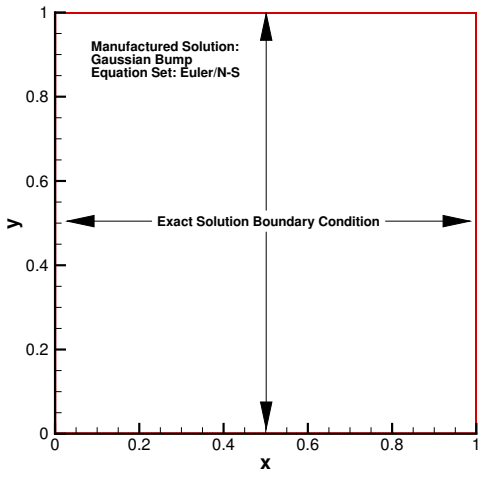


Figure I.19: Boundary conditions.

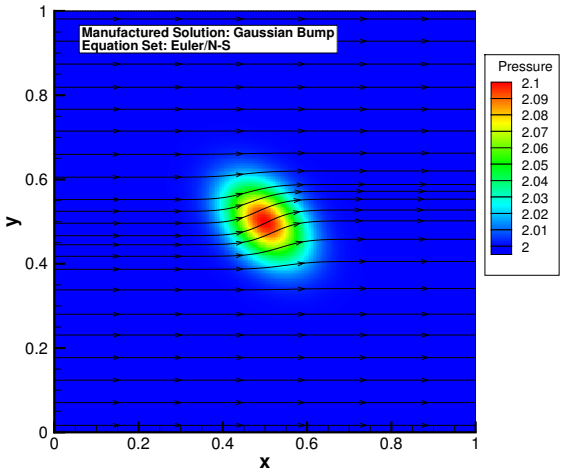


Figure I.20: Manufactured solution.

I.11 Laplace's Equation

Overview

- Case Name: "laplace" (`laplace_t`)
- Solution Type: Manufactured
- Equation Set: Laplace (`model_laplace_t`)
- Python Script: Y

Description

The `laplace_t` subclass is a manufactured solution designed for code verification and evaluating discretization error estimators. This subclass could also be used to model heat conduction in the presence of a heat source or heat sink. The manufactured solution takes the following form

$$u = u_{ref} \sin\left(\pi \frac{x}{L_{ref}}\right) \sin\left(\pi \frac{y}{L_{ref}}\right) \sin\left(\pi \frac{z}{L_{ref}}\right) \tag{I.3}$$

where u_{ref} sets the magnitude of u and L_{ref} is a reference length. By utilizing the MMS framework in BlueRidge in conjunction with the `model_laplace_t` equation set, users are able to solve Poisson's equation instead of Laplace's equation.

Table I.14: Namelist variables.

Variable Name	Variable Description	Default Value
laplace_uref	Reference Value of u	1.0
laplace_Lref	Reference Length	1.0

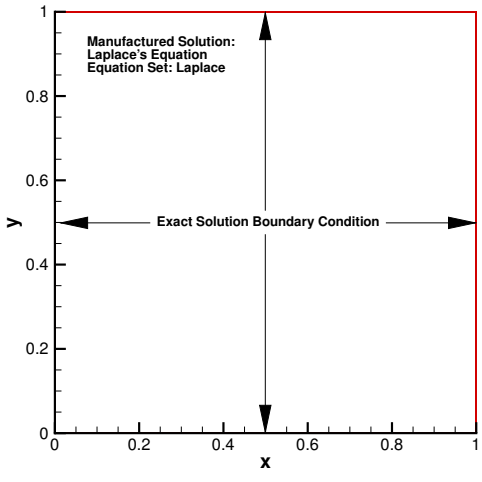


Figure I.21: Boundary conditions.

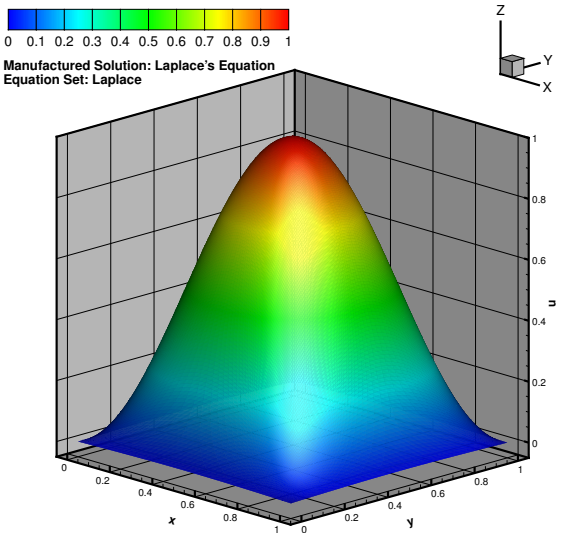


Figure I.22: Manufactured solution.

I.12 Burgers' Equation

Overview

- Case Name: "burgers" (`burgers_t`)
- Solution Type: Exact
- Equation Set: Burgers' (`model_burgers_t`)
- Python Script: Y

Description

The `burgers_t` subclass is used to simulate the coalescence of viscous shock waves. Flow enters from the left and right sides of the domain, and a shock wave forms along the diagonal. The steady-state, exact solution to Burgers' equation is given by the following

$$u = -u_{ref} \tanh\left[(x + y + z)\frac{Re}{2L_{ref}}\right] \tag{I.4}$$

where u_{ref} sets the magnitude of u, $Re = \frac{u_{ref} L_{ref}}{\nu}$ is a Reynolds number, ν is a viscosity, and L_{ref} is a reference length.

Table I.15: Namelist variables.

Variable Name	Variable Description	Default Value
burgers_uref	Reference Value of u	2.0
burgers_Re	Reynolds Number	16.0
burgers_Lref	Reference Length	1.0

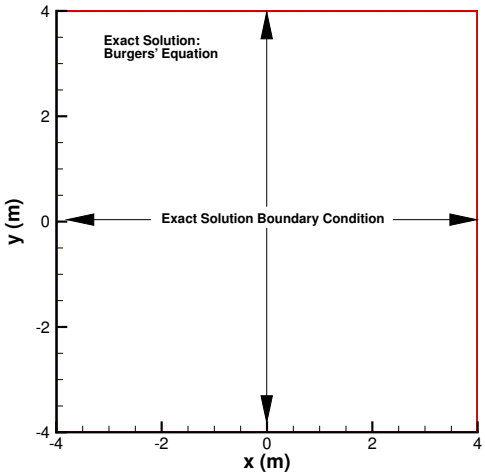


Figure I.23: Boundary conditions.

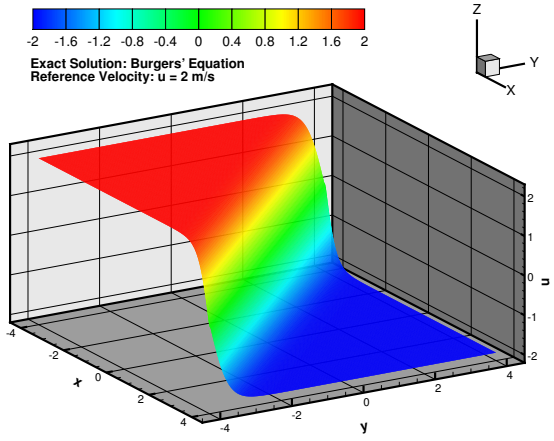


Figure I.24: Exact solution.

Bibliography

[1] J. D. Anderson, Modern Compressible Flow with Historical Perspective, McGraw-Hill, New York, New York, 2003.

[2] K. Masatsuka, I do like CFD, Vol. 1, Governing Equations and Exact Solutions, 2nd Edition, Cradle, 2013.

[3] F. Ringleb, Exakte lösungen der differentialgleichungen einer adiabatischen gasströmung, ZAMM-Journal of Applied Mathematics and Mechanics/Zeitschrift für Angewandte Mathematik und Mechanik 20 (4) (1940) 185–198.

[4] M. Aftosmis, D. Gaitonde, T. S. Tavares, Behavior of linear reconstruction techniques on unstructured meshes, AIAA Journal 33 (11) (1995) 2038 – 2049. doi:10.2514/3.12945. URL http://dx.doi.org/10.2514/3.12945

[5] A. Choudhary, Verification of compressible and incompressible computational fluid dynamics codes and residual-based mesh adaptation, Ph.D. thesis, Virginia Polytechnic Institute and State University (November 2014).
URL http://hdl.handle.net/10919/51169

[6] E. R. Eckert, R. M. Drake, Heat and Mass Transfer, McGraw-Hill, 1959.

[7] J. A. Schetz, R. D. Bowersox, Boundary Layer Analysis, American Institute of Aeronautics and Astronautics, 2011.

[8] K. Karamcheti, Principles of Ideal-Fluid Aerodynamics, Krieger Publishing Compansion, 1966.