Application of Improved Truncation Error Estimation Techniques to Adjoint Based Error Estimation and Grid Adaptation

Joseph M. Derlaga

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
Robert A. Canfield
Wayne L. Neu

July 1, 2015
Blacksburg, Virginia

Keywords: CFD, adjoint methods, defect correction, error transport equations, solution adaptation, truncation error, discretization error
Copyright 2015, Joseph M. Derlaga

Application of Improved Truncation Error Estimation Techniques to Adjoint Based Error Estimation and Grid Adaptation

Joseph M. Derlaga

(ABSTRACT)

Numerical solutions obtained through the use of Computational Fluid Dynamics (CFD) are subject to discretization error, which is locally generated by truncation error. The discretization error is extremely difficult to properly estimate and this in turn leads to uncertainty over the quality of the numerical solutions obtained via CFD methods and the engineering functionals computed using these solutions. Adjoint error estimation techniques specifically seek to estimate the error in functionals, but are dependent upon accurate truncation error estimates. This work examines the application of new, single-grid, truncation error estimation procedures to the problem of adjoint error estimation for both the quasi-1D and 2D Euler equations. The new truncation error estimation techniques are based on local reconstructions of the computed solutions and comparisons are made for the quasi-1D study in order to determine the most appropriate solution variables to reconstruct as well as the most appropriate reconstruction method. In addition, comparisons are made between the single-grid truncation error estimates and methods based on uniformally refining or coarsening the underlying numerical mesh on which the computed solutions are obtained. A method based on an refined grid error estimate is shown to work well for a non-isentropic flow for the quasi-1D Euler equations, but all truncation error estimations methods ultimately result in overprediction of functional discretization error in the prescence of a shock in 2D. Alternatives to adjoint methods, which can only estimate the error in a single functional for each adjoint solution obtained, are examined for the 2D Euler equations. The defection correction method and error transport equations are capable of locally improving the entire computed solution, allowing for error estimates in multiple functionals. It is found that all three functional discretization error estimates perform similarly for the same truncation error estimate, although the defect correction method is the most costly from a computational viewpoint. Comparisons are made between truncation error and adjoint weighted truncation error based adaptive indicators. For the quasi-1D Euler equations it is found that both methods are competitive, however the truncation error based method is cheaper as a separate adjoint solve is avoided. For the 2D Euler equations, the truncation error estimates on the adapted meshes suffer due to a lack of smooth grid transformations which are used in reconstructing the computed solutions. In order to complete this work, a new CFD code incorporating a variety of best practices from the field of Computer Science is developed as well as a new method of performing code verification using the method of manufactured solutions which is significantly easier to implement than traditional manufactured solution techniques.

This work was partially supported 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 through the NASA Pathways Program during the author's time at NASA's Langley Research Center.

To my wife, Anna Pearl

Contents

List of Figures

List of Tables

Chapter 1

Introduction

While experiments have not been fully replaced by computational simulations, the use of computational fluid dynamics (CFD) to provide predictions of engineering quantities of interest is increasingly accepted, if not the norm, in research and industry. This is in part due to the decreasing cost, in terms of hardware investments and time, of CFD simulations compared to the experiments, but also to the improvements in the overall fidelity of simulations due to increases in grid resolution, algorithm improvements, and an increased focus on ensuring that codes are verified and that validation activities are performed.

Despite how much CFD simulations have improved, the quantities they predict are still subject to a variety of errors and uncertainties because the equations that govern the physical phenomena of interest must be discretized in order for them to be interpreted by a computer. The discretization process results in a truncation error (TE) due to the loss of higher order terms in a manner dependent on the method used to discretize the equations, whether finite difference, finite volume, finite element, discontinuous Galerkin, etc., which in turn affects how the 'quality' of the mesh on which a given problem is solved contributes to the TE. Formally, the TE is defined as the difference between the exact governing *equations* and the discretization of these *equations*.

The TE results in the *solution* to the discretized equations differing from the exact *solution* of the governing partial (or ordinary) differential equations by a quantity known as discretization error (DE). This is a difficult, if not impossible, quantity to calculate simply due the lack of a known exact solution to the continuous governing equations for most problems of interest. Methods are available to estimate both the TE and DE, and have been utilized in the past to correct computational solutions or drive the order/mesh refinement process, as reviewed by Roy [1, 2]. Of these, adjoint methods have recently come to the forefront of the field and have proven to be an invaluable tool, not only for error estimation, but also for design optimization. In addition, much work has been undertaken to reduce the effects of DE without necessarily quantifying it. This can been seen through the use of order refinement, and/or mesh adaptation to improve computed solutions.

The main focus of this work is the use of adjoint methods to address both the problem of improving a computed solution through mesh refinement and of providing a DE estimate for functional outputs, through the use of improved TE estimation procedures. Given the increased use of CFD, the ability to accurately predict the effects of TE is extremely important. Additionally, in spite of the decreasing costs associated with performing computations, it is still necessary to ensure that the adaptation and error estimation costs are reasonable. To this end, comparisons are made between the adjoint method, defect correction method, and error transport equations.

1.1 Previous Work

While closely related, error estimation and grid adaptation are discussed separately in order to better understand the different methods that are available not only to drive grid adaptation, but the different ways in which grid adaptation can be performed.

1.1.1 Error Estimation

While adjoint-methods have been more recently developed for adaptation and error-estimation purposes, the need for error estimates has long been known. Perhaps the earliest error estimator is due to the work of Richardson [3, 4], who used the solutions on multiple systematically refined grids, and knowledge of the formal order of the numerical scheme, in order to obtain a better solution to the load on a masonry dam. While an extremely simple approach based on post processing data, the need for a systematically refined mesh can be prohibitive on large problems, and questions remain on exactly what constitutes a systematically refined mesh for unstructured grid methods.

More advanced residual based methods, such as defect correction and error transport equations, are more code intrusive and require TE estimates in order to estimate the DE. As introduced by Fox [5], and generalized by Pereyra [6, 7], the discrete defect correction methods essentially operate a higher order discretization on a solution obtained from a lower order discretization to form a defect (which could be considered a form of TE estimate) that can be used to drive the lower order method to an improved solution.

More recently, the error transport equations (ETEs) have been studied in the works of Zhang et al. [8, 9], Qin et al. [10, 11, 12, 13], and Cavallo et al. [14, 15, 16]. In the works of Cavallo et al., the ETEs were extended to 3D Euler and RANS equations. Phillips and Roy [17] examined the formulation of the ETEs and demonstrated how the technique must be derived for each discretization method in order to avoid linearization errors. The linearized operator (in the discrete sense, a large, sparse matrix) applied to the DE should produce the TE; in other words, the TE drives the DE. However, the linearized operator should not be approximated, which is often the case in most CFD solvers due to memory issues and the

complexity of the formulation.

While the previous approaches can be used to improve the entire solution, adjoint methods specifically provide an error estimate for functional outputs. Proposed by Pironneau [18] and effectively implemented by Jameson [19, 20] for design purposes, Giles and Pierce [21] and Venditti and Darmofal [22] independently developed the use of adjoint error estimates for Galerkin and finite volume methods, respectively. The adjoint (or dual) problem, while providing output functional derivatives with respect to the residual operator, essentially indicates the domain of influence for a functional output. In simplified terms, if the derivative is zero at a point, then the functional does not depend on the value of the solution at that point.

In order to provide the error estimate, or correction term, an inner product of the adjoint solution and an estimate of the TE is performed. Giles and Pierce [23] noted that the TE estimate could be formed by operating the strong form of the governing equations on a reconstructed solution, and later made comparisons with the defect correction method [24]. An alternative approach was suggested by Venditti and Darmofal [22, 25, 26], where instead of a true TE estimate, a residual error on a finer mesh was created for the 2D Euler and Navier-Stokes equations in order to create a computable correction to predict the functional output on a uniformly refined grid. It should be noted that this finer/embedded grid approach is memory intensive. For either approach, a Galerkin method utilizing a discrete adjoint with dual consistent boundary conditions [27] should produce superconvergent error estimates, i.e., the error estimates should converge at a higher than design order rate. However, Hicken and Zingg [28] state that higher order finite volume methods will not, in general, see the benefits of superconvergence without the use of Richardson extrapolation, as used in [22]. The finer mesh (or embedded grid) method became extremely popular as it was able to utilize the existing residual calculation routines available in most CFD solvers without needing complex reconstruction and differentiation routines.

Restricting the discussion to finite volume methods, Park [29] first extended the embedded grid error estimation procedure to three dimensions and cut-cell simulations for the Euler and laminar Navier-Stokes equations, and created a single grid, residual-based, error estimate and adaptation indicator in order to address some of the memory issues of the embedded grid approach. Balasubramanian [30, 31] studied the incompressible Navier-Stokes equations and examined the use of moving least-squares reconstructions in place of the nearest neighbor reconstruction patches used by Venditti [32]. In addition, the uniformly refined grid approach was extended to account for variable refinement factors. Nemec and Aftosmis [33] developed the adjoint error estimation procedure for overset, Cartesian based cut-cell grids. For a more complete discussion of adjoint methods to other numerical schemes, see the review papers of Giles et al. [34] and Fidkowski and Darmofal [35].

The adjoint solution itself can be obtained through either a continuous or discrete approach, as summarized by Giles and Pierce [21]. In the continuous approach, the governing equations of interest are linearized, the adjoint equations are formed and then discretized. On the other

hand, the discrete approach is based on linearizing the discretized governing equations and taking the transpose of this linear operator. While each method has pros and cons, the methods are consistent and should produce the same adjoint solution in the limit of mesh refinement [36]. Therefore, the option of which method to implement is most often viewed as a matter of personal choice or driven by project constraints. The continuous method allows for a simpler discretization as it can be easily formulated in an explicit manner, while the discrete adjoint requires the exact flux Jacobians of the underlying numerical scheme, which may be only partially developed in many CFD codes. The use of automatic differentiation methods [21, 37, 36] or complex step methods [21, 38]. eases this development burden, but at the potential cost of increased run time. Boundary conditions for the continuous adjoint method must be carefully derived in order to properly cancel the boundary adjoint terms in a manner consistent with the primal boundary conditions. For the discrete adjoint, weak boundary conditions are naturally included in the adjoint residual, while strong boundary conditions require slight modifications.

1.1.2 Grid Adaptation

It is well known that a naive approach to error reduction would be to simply increase the order of the numerical scheme being used; however, this is not without drawbacks. Many schemes, such as discontinuous Galerkin methods, suffer from increased computational cost due to an increased number of degrees of freedom. There is also the potential for a decrease in stability as shown by Godunov's theorem [39] and exacerbated due to increased stencil sizes and stencil effects. In the case of some high-order finite difference schemes, there is a lack of flexibility simply due to limitations in meshing a geometry, and while overset grid technology can mitigate some of these problems, the interpolation between meshes is generally of lower order than the numerical schemes and/or is subject to loss of conservation.

More commonly, the problem of reducing DE is attacked by refining the mesh. Once again, the naive approach would be to simply uniformly increase the number of mesh cells (or nodes), but this can quickly become intractable as a 3D, hexahedral mesh would require an eight fold increase in the number of cells. Instead, adaptive mesh refinement techniques seek to selectively improve the mesh only where needed. This improvement can take several forms depending on the adaptation technique being used, as reviewed in [40, 41, 42, 43, 44]. Basic h-adaptation will insert grid nodes where more refinement is needed, but will not move nodes in order to improve the mesh. While commonly used by unstructured grid methods, this type of adaptation is prohibited by structured grid methods where regular connectivity is required. Instead, r-adaptation, or node movement, can be used by both structured and unstructured meshes. In either case, an adaptation indicator is necessary in order drive the node insertion/relocation procedure, as discussed in Roy [2].

Early work [45, 46, 47] focused on using solution features or gradients to drive adaptation. The motivation behind this was twofold: 1) features/gradients generally correspond to 'in-

teresting' regions of the domain, and 2) flow gradient terms would show up in the TE terms and improvement in their resolution should therefore reduce the DE. In many cases, the adaptation parameters were chosen in an ad hoc manner on a case by case basis, or, as noted in Hawken et al. [40], with no justification at all. Warren et. al [48] pointed out a serious problem with feature based adaptation, namely that feature based adaptation, while improving resolution as requested, does not take into account the areas of the domain necessary to properly predict the flow physics.

An obvious choice for driving adaptation would be to use estimates of the DE itself, but this neglects the fact that the local source of the DE is the TE, and adapting on DE has been shown to fail [49].

In contrast to feature and DE based adaptation, Pereyra and Sewell [50] and Fung et al. [51] clearly noted the need to use TE to drive adaptation, but the methodology did not see widespread use. Choudhary and Roy [52, 53] revisited using exact TE to drive adaptation in a more rigorous manner, resulting in an order of magnitude reductions in DE for 2D Euler cases while using r-adaptation.

An alternative adaptation approach, as discussed above, is to use adjoint-based methods, as pioneered by Venditti and Darmofal [22, 25, 26] for finite-volume schemes. Muller and Giles [54] used the adjoint correction term as an adaptation indicator, while Venditti [32] argued for the use of a duality gap indicator. This indicator seeks to improve the DE in a functional output by targeting weighted residual error estimates of the primal and dual problems. The combined works of Venditti and Darmofal essentially launched the wide spread use of adjoint-methods for adaptation as well as error estimation. This is perhaps best seen in the works of Alauzet [55], Alauzet et al. [56], and Loseille et al. [57], who further developed the combined adjoint and Hessian based adaptation method of Venditti and Darmofal, leading to the optimization of fixed degrees of freedom meshes, i.e., meshes where the total number of grid nodes is specified.

1.2 Contributions

The application of a novel TE estimation procedures to formulate an adjoint based error estimate for a variety of finite volume methods is developed. In addition, comparisons of adjoint and TE based grid adaptation methods are made. Initially studied within the context of the quasi-1D Euler equations, that work was extended into the development of a new, 3D, structured grid, multi-block CFD solver capable of solving the discrete adjoint equations for both the Euler and laminar Navier-Stokes equations. Compared to previous work in structured grid solvers with adjoint capabilities, this new code has greater range of freedom in terms of available numerical methods and utilizes modern programming practices aimed at providing a framework for future research efforts. In the process of developing the new CFD solver, a new method of formulating the source terms for the Method of Man-

ufactured Solutions (MMS) was developed in order to simplify code verification activities. The new adjoint and TE estimation procedure is applied to several 2D Euler cases where exact solutions are available for comparison and which were chosen to highlight how flow features affect the convergence order of the solution and the error estimates. First of their kind comparisons between adjoint methods, error transport equations, and defect correction methods are made for the two-dimensional Euler equations and it is shown that all methods perform in a similar manner for the same TE estimate. The adjoint solution and TE error estimate are used to drive grid adaptation and can be formulated to have a smaller memory footprint as compared to traditional embedded grid methods. Due to the use of a local stencil operation to form the TE estimate, it can be easily extended to leverage the compute power of massively parallel devices such as GPUs.

1.3 Outline

The first chapter is arranged to introduce the reader to the concepts of numerical error and error estimation and discuss relevant background materials. The second chapter discusses the initial application of the improved truncation error estimation procedure and its application to adjoint based error estimates for the quasi-1D Euler equations. The third chapter covers the theory and development of a new CFD flow solver utilizing modern Fortran and agile programming practices. The fourth chapter discusses applying the methods of the second chapter to the code base described in the third chapter as well as comparisons of various TE estimation procedures to three functional DE estimation methods (the error transport equations, the defect correction method, and the adjoint method) on both uniformly refined and adapted meshes. The final chapter draws conclusions from the work and discusses further research topics and open questions.

1.4 Attribution

As this work is in the 'manuscript' format, each of Chapters 2 through 4 show multiple authors. The first author, Joseph M. Derlaga, contributed the bulk of the work in developing and producing the results discussed within each paper, but a more complete breakdown of the scope of each author's contributions are detailed below.

Chapter 2: The first author (Joseph M. Derlaga), created both the quasi-1D Euler solver, its adjoint, and the TE estimation procedures. The grid adaptation method used within the work was contributed by Aniruddha Choudhary. The second (Christopher J. Roy) and third (Jeffrey T. Borggaard) authors provided valuable feedback and guidance.

Chapter 3: The first author (Joseph M. Derlaga) served as the main contributing writer to this work and served as the main code architect, introducing many new features of modern

Fortran programming to the coding team and worked closely with the second author (Tyrone S. Phillips) on the development of the core structure of the SENSEI CFD code. Specifically, the first author created a majority of the code necessary to construct the inviscid and viscous residuals, performed the linearization of the residuals and the boundary conditions, created the linear solver suite and its associated polymorphic compressed storage capabilities, and implemented the discrete adjoint solver. The third author (Christopher J. Roy) provided guidance and requirements for the development of the new solver.

Chapter 4: The first author (Joseph M. Derlaga) served as the main contributing writer to this work and performed all the simulations contained within the work. The SENSEI CFD code was developed collaboratively between the first and second authors, as discussed in Chapter 3. The adjoint solver was developed by the first author, while the truncation error estimation procedures were implemented by the second author (Tyrone S. Phillips). The second author contributed the defect correction implementation and the original implementation of the error transport equations which were extended to second order by the first author. The grid adaptation method was contributed by Aniruddha Choudhary. As with the second chapter, the third (Christopher J. Roy) and fourth (Jeffrey T. Borggaard) authors provided valuable feedback and guidance.

1.5 Bibliography

[1] Roy, C. J., "Review of Discretization Error Estimators in Scientific Computing," AIAA Paper 2010-0126, 48th AIAA Aerospace Sciences Meeting, Orlando, Florida, January 4-7, 2010.

[2] Roy, C. J., "Strategies for Driving Mesh Adaptation in CFD," Invited Paper for Session on Error Estimation and Control, AIAA Paper 2009-1302, 47th AIAA Aerospace Sciences Meeting, Orlando, Florida, January 5-8, 2009.

[3] Richardson, L. F., "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. Series A, Containing Papers of a Mathematical and Physical Character*, Vol. 83, No. 563, 1910, pp. pp. 335–336.

[4] Richardson, L. F., "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 of London. Series A, Containing Papers of a Mathematical or Physical Character*, Vol. 210, 1911, pp. pp. 307–357.

[5] Fox, L., "Some Improvements in the Use of Relaxation Methods for the Solution of Ordinary and Partial Differential Equations," *Procedings of the Royal Society of London. Series A*, Vol. 190, 1947, pp. pp. 31–59.

[6] Pereyra, V., "Iterated Deferred Corrections for Nonlinear Operator Equations," *Numerische Mathematik*, Vol. 10, No. 4, 1967, pp. 316–323.

[7] Pereyra, V., "Iterated Deferred Corrections for Nonlinear Boundary Value Problems," *Numerische Mathematik*, Vol. 11, No. 2, 1968, pp. 111–125.

[8] Zhang, X. D., Trepanier, J.-Y., and Camarero, R., "A posteriori error estimation for finite-volume solutions of hyperbolic conservation laws," *Computer Methods in Applied Mechanics and Engineering*, Vol. 185, 2000, pp. 1–19.

[9] Zhang, X. D., Pelletier, D., and Trepanier, J.-Y., "Verification of error estimators for the Euler equations," AIAA Paper 2000-1001, 38th AIAA Aerospace Sciences Meeting and Exhibit, Reno, NV, January 10-13, 2000.

[10] Qin, Y. and Shih, T. I.-P., "A Discrete Transport Equation for Error Estimation in CFD," AIAA Paper 2002-0906, 40th AIAA Aerospace Sciences Meeting and Exhibit, Reno, NV, January 14-17, 2002.

[11] Qin, Y. and Shih, T. I.-P., "A Method for Estimating Grid-Induced Errors in Finite-Difference and Finite-Volume Methods," AIAA Paper 2003-845, 41st AIAA Aerospace Sciences Meeting and Exhibit, Reno, NV, January 6-9, 2003.

[12] Qin, Y., Keller, P. S., Sun, R. L., Hernandez, E. C., Perng, C.-Y., Trigui, N., amd F. Z. Shen, Z. H., Shieh, T., and Shih, T. I.-P., "Estimating Grid-Induced Errors in CFD by Discrete-Error-Transport Equations," AIAA Paper 2004-656, 42nd AIAA Aerospace Sciences Meeting and Exhibit, Reno, NV, January 5-8, 2004.

[13] Qin, Y., Chi, K., and Shih, T. I.-P., "Modeling the Residual in Error-Transport Equations for Estimating Grid-Induced Errors in CFD Solutions," AIAA Paper 2006-892, 44th AIAA Aerospace Sciences Meeting and Exhibit, Reno, NV, January 9-12, 2006.

[14] Cavallo, P. and Sinha, N., "An Error Transport Equation with Practical Applications," AIAA Paper 2007-4092, 18th AIAA Computational Fluid Dynamics Conference, Miami, FL, June 25-28, 2007.

[15] Cavallo, P. and Sinha, N., "Error Quantification for Computational Aerodynamics Using an Error Transport Equation," *Journal of Aircraft*, Vol. 44, No. 6, 2007, pp. 778–790.

[16] Cavallo, P., Sinha, N., and OGara, M. R., "Viscous Error Transport Equation for Error Quantification of Turbulent Flows," AIAA Paper 2008-3851, 38th AIAA Fluid Dynamics Conference and Exhibit, Seattle, WA, June 23-26, 2008.

[17] Phillips, T. S. and Roy, C. J., "Residual Methods for Discretization Error Estimation," AIAA Paper 2011-3870, 20th AIAA Computational Fluid Dynamics Conference, Honolulu, Hawaii, June 27-30, 2011.

[18] Pironneau, O., "On optimum design in fluid mechanics," *Journal of Fluid Mechanics*, Vol. 64, No. 1, 1974, pp. 97–110.

[19] Jameson, A., "Aerodynamic design via control theory," *Journal of Scientific Computing*, Vol. 3, No. 3, 1988, pp. 233–260.

[20] Jameson, A., "Optimum Aerodynamic Design Using CFD and Control Theory," AIAA Paper 95-1729-CP, AIAA, 1995.

[21] Giles, M. and Pierce, N., "An Introduction to the Adjoint Approach to Design," *Flow, Turbulence and Combustion*, Vol. 65, 2000, pp. 393–415.

[22] Venditti, D. A. and Darmofal, D. L., "Adjoint Error Estimation and Grid Adaptation for Functional Outputs: Application to Quasi-One-Dimensional Flow," *Journal of Computational Physics*, Vol. 164, No. 1, 2000, pp. 204 – 227.

[23] Giles, M. and Pierce, N., "Adjoint equations in CFD: duality, boundary conditions and solution behaviour," AIAA Paper 97-1850, AIAA, 1997.

[24] Pierce, N. A. and Giles, M. B., "Adjoint and defect error bounding and correction for functional estimates," *Journal of Computational Physics*, Vol. 200, No. 2, 2004, pp. 769 – 794.

[25] Venditti, D. A. and Darmofal, D. L., "Grid Adaptation for Functional Outputs: Application to Two-Dimensional Inviscid Flows," *Journal of Computational Physics*, Vol. 176, No. 1, 2002, pp. 40 – 69.

[26] Venditti, D. A. and Darmofal, D. L., "Anisotropic grid adaptation for functional outputs: application to two-dimensional viscous flows," *Journal of Computational Physics*, Vol. 187, No. 1, 2003, pp. 22 – 46.

[27] Lu, J., *An a posteriori error control framework for adaptive precision optimization using discontinuous Galerkin finite element method*, Ph.D. thesis, Massachusetts Institute of Technology, 2005.

[28] Hicken, J. E. and Zingg, D. W., "The Role of Dual Consistency in Functional Accuracy: Error Estimation and Superconvergence," AIAA Paper 2011-3855, AIAA 20th Computational Fluid Dynamics Conference, 2011.

[29] Park, M. A., *Anisotropic Output-Based Adaptation with Tetrahedral Cut Cells for Compressible Flows*, Ph.D. thesis, Massachusetts Institute of Technology, September 2008.

[30] Balasubramanian, R., *Adjoint-Based Error Estimation and Grid Adaptation for Functional Outputs from CFD Simulations*, Ph.D. thesis, Mississippi State, December 2005.

[31] Balasubramanian, R. and Newman, J. C., "Discrete direct and adjoint sensitivity analysis for arbitrary Mach number flows," *International Journal for Numerical Methods in Engineering*, Vol. 66, No. 2, 2006, pp. 297–318.

[32] Venditti, D. A., *Grid Adaptation for Functional Outputs of Compressible Flow Simulations*, Ph.D. thesis, Massachusetts Institute of Technology, June 2002.

[33] Nemec, M. and Aftosmis, M. J., "Adjoint Error Estimation and Adaptive Refinement for Embedded-Boundary Cartesian Meshes," AIAA Paper 2007-4187, 18th AIAA Computational Fluid Dynamics Conference, Miami, Florida, June 25-28, 2007.

[34] Giles, M., Duta, M., Muller, J., and Pierce, N., "Algorithm Developments for Discrete Adjoint Methods," *AIAA Journal*, Vol. 41, No. 2, February 2003, pp. 198–205.

[35] Fidkowski, K. J. and Darmofal, D. L., "Review of Output-Based Error Estimation and Mesh Adaptation in Computational Fluid Dynamics," *AIAA Journal*, Vol. 49, No. 4, April 2011, pp. 673–694.

[36] Alauzet, F. and Pironneau, O., "Continuous and discrete adjoints to the Euler equations for fluids," *International Journal for Numerical Methods in Fluids*, Vol. 70, No. 2, September 2012, pp. 135–137.

[37] Noack, A. and Walther, A., "Adjoint concepts for the optimal control of Burgers equation," *Computational Optimization and Applications*, Vol. 36, 2007, pp. 109–133.

[38] Nielsen, E. and Kleb, W., "Efficient Construction of Discrete Adjoint Operators on Unstructured Grids Using Complex Variables," *AIAA Journal*, Vol. 44, No. 4, 2006, pp. 827–836.

[39] Hirsch, C., *Numerical Computation of Internal and External Flows*, Vol. 2: Computational Methods for Inviscid and Viscous Flows, John Wiley & Sons, Ltd, 1990.

[40] Hawken, D., Gottlieb, J., and Hansen, J., "Review of some adaptive node-movement techniques in finite-element and finite-difference solutions of partial differential equations," *Journal of Computational Physics*, Vol. 95, No. 2, 1991, pp. 254 – 302.

[41] McRae, D. S., "r-Refinement grid adaptation algorithms and issues," *Computer Methods in Applied Mechanics and Engineering*, Vol. 189, No. 4, 2000, pp. 1161 – 1182, Adaptive Methods for Compressible CFD.

[42] Habashi, W., Dompierre, J., Bourgault, Y., Ait-ali Yahia, D., Fortin, M., and Vallet, M.-G., "Anisotropic mesh adaptation: towards user-independent, mesh-independent and solver-independent CFD. Part I. general principles," *International Journal for Numerical Methods in Fluids*, Vol. 32, No. 6, March 2000, pp. 725–744.

[43] Ait-ali Yahia, D., Baruzzi, G., Habashi, W. G., Fortin, M., Dompierre, J., and Vallet, M.-G., "Anisotropic mesh adaptation: towards user-independent, mesh-independent and solver-independent CFD. Part II. Structured grids," *International Journal for Numerical Methods in Fluids*, Vol. 39, No. 8, July 2002, pp. 657–673.

[44] Habashi, Wagdi G., D. J., Vallet, M.-G., Bourgault, Y., and Fortin, M., "Anisotropic mesh adaptation: towards user-independent, mesh-independent and solver-independent CFD. Part III. Unstructured meshes," *International Journal for Numerical Methods in Fluids*, Vol. 39, No. 8, Jul 2002, pp. 675–702.

[45] Brackbill, J. and Saltzman, J., "Adaptive Zoning for Singular Problems in Two Dimensions," *Journal of Computational Physics*, Vol. 46, 1982, pp. 342 – 368.

[46] Peraire, J., Vahdati, M., Morgan, K., and Zienkiewicz, O., "Adaptive remeshing for compressible flow computations," *Journal of Computational Physics*, Vol. 72, No. 2, 1987, pp. 449–466.

[47] Jeng, Y. N. and Liou, Y. C., "A new adaptive grid generation by elliptic equations with orthogonality at all of the boundaries," *Journal of Scientific Computing*, Vol. 7, No. 1, 1992, pp. 63–80.

[48] Warren, G. P., Anderson, W. K., Thomas, J. L., and Krist, S. L., "Grid Convergence for Adaptive Methods," AIAA Paper 1991-1592, AIAA 10th Computational Fluid Dynamics Conference, 1991.

[49] Gu, X. and Shih, T. I.-P., "Differentiating between Source and Location of Error for Solution-Adaptive Mesh Refinement," AIAA Paper 2001-2660, 15th AIAA Computational Fluid Dynamics Conference, 2001.

[50] Pereyra, V. and Sewell, E. G., "Mesh selection for discrete solution of boundary problems in ordinary differential equations," *Numerische Mathematik*, Vol. 23, No. 3, 1975, pp. 261–268.

[51] Fung, K.-Y., Tripp, J., and Goble, B., "Adaptive refinement with truncation error injection," *Computer Methods in Applied Mechanics and Engineering*, Vol. 66, No. 1, 1988, pp. 1–16.

[52] Choudhary, A. and Roy, C. J., "Efficient Residual-Based Mesh Adaptation for 1D and 2D CFD Applications," AIAA Paper 2011-0214, 49th AIAA Aerospace Sciences Meeting, Orlando, Florida, January 4-7, 2011.

[53] Choudhary, A. and Roy, C. J., "Structured Mesh r-Refinement using Truncation Error Equidistribution for 1D and 2D Euler Problems," AIAA Paper 2013-2444, AIAA Fluid Dynamics and Co-located Conferences and Exhibit, San Diego, California, June 24-27, 2013.

[54] Muller, J.-D. and Giles, M., "Solution Adaptive Mesh Refinement Using Adjoint Error Analysis," AIAA Paper 2001-2550, AIAA, 2001.

[55] Alauzet, F., "Size gradation control of anisotropic meshes," *Finite Elements in Analysis and Design*, Vol. 46, No. 1-2, 2010, pp. 181 – 202, Mesh Generation - Applications and Adaptation.

[56] Alauzet, F. and Loseille, A., "High-order sonic boom modeling based on adaptive methods," *Journal of Computational Physics*, Vol. 229, No. 3, 2010, pp. 561 – 593.

[57] Loseille, A., Dervieux, A., and Alauzet, F., "Fully anisotropic goal-oriented mesh adaptation for 3D steady Euler equations," *Journal of Computational Physics*, Vol. 229, No. 8, 2010, pp. 2866 – 2897.

Chapter 2

Adjoint and Truncation Error Based Adaptation and Error Estimation for 1D Finite Volume Schemes

Joseph M. Derlaga[a], Christopher J. Roy[a],
Jeffrey T. Borggaard[b]
[a]Department of Aerospace and Ocean Engineering, Virginia Tech, Blacksburg, VA 24061, USA
[b]Department of Mathematics, Virginia Tech, Blacksburg, VA 24061, USA

Attribution

The first author (Joseph M. Derlaga), created both the quasi-1D Euler solver, its adjoint, and the TE estimation procedures. The grid adaptation method used within this work was contributed by Aniruddha Choudhary. The second (Christopher J. Roy) and third (Jeffrey T. Borggaard) authors provided valuable feedback and guidance.

Abstract

In addition to design, control, and optimization applications, adjoint methods can be used to provide discretization error estimation and solution adaptation for solution functionals. This paper seeks to demonstrate the link between adjoint based error estimation, truncation errors, residuals, and discretization errors. The generalized truncation error expression is extended from finite difference schemes to finite volume schemes and is used to provide truncation error estimates for the quasi-1D form of the Euler equations. Comparisons between different types of adaptation indicators and error estimation techniques are made. Additionally, functional error estimates from the defect correction method and error transport equations are compared to adjoint error estimates and are shown to provide similar accuracy.

2.1 Introduction

In the field of computational fluid dynamics (CFD), mesh adaptation can be used to improve solution accuracy by attempting to efficiently place the mesh in order to properly resolve the flow field. This can also reduce discretization error, which is defined as the difference between the computed solution to the discretized system and the solution to the continuous, differential or integral equations. The problem of how to efficiently place the mesh is normally split into two parts: one which deals with where adaptation is needed, i.e., an adaptation indicator, and another which deals with how the mesh should be adapted.

Currently, most adaptation methods use either a feature-based or adjoint-based adaptation indicator. Feature-based methods locally target areas of solution peaks, gradients, or curvature and are computationally inexpensive to implement. Unfortunately, while 'interesting' regions are refined, the final solution may be incorrect due to error transport from 'non-interesting' areas of the domain as discussed by Warren et al. [1]. That is, they neglect areas of the flow domain necessary to accurately resolve relevant flow physics. Alternatively, adjoint-based methods seek to remedy this by use of the adjoint, or dual, solution to the primal flow problem. Through the use of a Lagrangian, a new equation is formed that determines the sensitivity of a functional, such as lift or drag in an aerodynamic application, to perturbations in the governing equations. The sensitivities, i.e. the adjoint solution, can then be used to identify regions of the flow domain that impact the calculation of the functional of interest. In Venditti's embedded grid approach [2], the solution to the primal and dual problems are interpolated to a grid created by uniformly refining an initial grid and the result is used to provide an *a priori* estimate of the functional on the embedded grid. As will be discussed, the adaptation indicator is formed so that the discretization and truncation errors in the primal and dual problems are reduced. It has often been found that while adjoint-based adaptation does indeed improve the predictions of the functional of interest, other functional outputs may not improve as shown in the works of Park et al. [3] and Fidkowski [4]. This is in part due to how the second part of the grid adaptation process

is addressed.

For unstructured, Cartesian mesh solvers, the adaptation indicator flags cells for simple subdivision and then uniformly divides a flagged cells into smaller ones without changing the underlying grid structure. For unstructured/mixed-element solvers, mesh structure is often determined by the Hessian matrix of a flow quantity. As first described by Peraire [5], the Hessian and adaptation indicator are combined to form a compact metric that describes a continuous metric field that prescribes mesh density and structure throughout the computational domain. While this system has become the de facto standard for prescribing mesh adaptation, feature-based Hessian methods fail to properly account for how discretization error is locally generated (via the truncation error) and transported along with the solution through the domain.

This paper reviews how discretization error and truncation error are related for finite volume methods, explains the adjoint-based error estimation and mesh refinement technique, reviews defect correction and error transport equations in comparison to adjoint error estimation, suggests an improved method to drive adaptation, and provides comparisons with current methods in the literature.

2.2 Truncation and Discretization Error

Solution adaptation seeks to improve solution accuracy by ultimately reducing truncation errors that cause discretization error. In order to better understand this process, it is useful to first examine the generalized truncation error expression, or GTEE [6].

Consider a system of governing equations $L(u) = 0$, that are discretized to form the discrete system $L_h(u) = 0$. These two systems have exact solutions of $\tilde{u}$ and u_h, respectively. The process of discretizing the governing equations produces a system where the discretized equations are equal to the continuous governing equations plus another set of terms called the truncation error, as described by the GTEE, where τ_h is the truncation error:

$$L_h(u) = L(u) + \tau_h(u). \tag{2.1}$$

Note that appropriate restriction and prolongation operators are needed since, in the case of finite volume methods, the discrete equations operate on piecewise constant values while the continuous governing equations operate on continuous, or piecewise continuous, functions. Before continuing, it is necessary to define some operators, $I^{\cdot}_{\cdot}$, that can be used to indicate interpolation or restriction. For example, I^h_{2h} indicates interpolation from a coarse grid, $2h$, to a fine grid, h, or I^h acting as a restriction from continuous space to a discrete grid with mesh spacing h. A term such as I^q_h indicates a reconstruction from discrete space to a q-th order polynomial.

If the exact solution to the discrete equations $I_h u_h$ is substituted into the GTEE and $L(\tilde{u})$

is subtracted from both sides, the GTEE becomes:

$$0 = L\left(I_h u_h\right) - L\left(\tilde{u}\right) + \tau_h\left(I_h u_h\right) \tag{2.2}$$

If the equations are linear (or linearized), as discussed by Phillips and Roy [7], and with the discrete form of the discretization error defined as $\epsilon_h = u_h - I^h\tilde{u}$, the above becomes:

$$L\left(I_h \epsilon_h\right) = -\tau_h\left(I_h u_h\right) \tag{2.3}$$

which is a continuous transport equation for the discretization error. A discrete form of this error transport equation can also be derived that results in:

$$L_h\left(\epsilon_h\right) = -\tau_h\left(\tilde{u}\right). \tag{2.4}$$

In either case, it is shown that truncation error acts as the source term for discretization error, which is transported in the same manner as the flow solution. The dependence of the discretization error on the truncation error is why it is important to reduce truncation error in order to improve a computed solution, rather than simply target solution gradients or other flow features, and this serves as the basis for adjoint-based adaptation methods.

2.3 Review of Adjoint Methods

Most CFD simulations are performed to estimate engineering functionals rather than for just obtaining the solution of the dependent variables throughout the entire domain. Due to the truncation error, the computed functionals are affected by the discretization error of the solution, as discussed above. When used to provide discretization error estimates in functionals, adjoint methods generally combine the adjoint variable (which provides the sensitivity of the functional to local equation perturbations) with the truncation error (or its residual-based estimate). Suppose there is a continuous system of governing equations with an exact solution $\tilde{u}$ and an approximate solution u which is expanded using a Taylor series as given below:

$$L\left(\tilde{u}\right) = L\left(u\right) + \left.\frac{\partial L}{\partial U}\right|_{U=u}\left(\tilde{u}-u\right) + \left.\frac{\partial^2 L}{\partial U^2}\right|_{U=u}\frac{\left(\tilde{u}-u\right)^2}{2} + \cdots = 0. \tag{2.5}$$

Consider also a functional dependent upon the solution to the system:

$$J\left(\tilde{u}\right) = J\left(u\right) + \left.\frac{\partial J}{\partial U}\right|_{U=u}\left(\tilde{u}-u\right) + \left.\frac{\partial^2 J}{\partial U^2}\right|_{U=u}\frac{\left(\tilde{u}-u\right)^2}{2} + \cdots. \tag{2.6}$$

Truncating the Taylor series at 2nd order, and combining Equations 2.5 and 2.6 via a Lagrangian, we have:

$$\begin{aligned} J\left(\tilde{u}\right) \approx J\left(u\right) + \left.\frac{\partial J}{\partial U}\right|_{U=u}\left(\tilde{u}-u\right) + \left.\frac{\partial^2 J}{\partial U^2}\right|_{U=u}\frac{\left(\tilde{u}-u\right)^2}{2}\cdots \\ +\lambda\left[L\left(u\right) + \left.\frac{\partial L}{\partial U}\right|_{U=u}\left(\tilde{u}-u\right) + \left.\frac{\partial^2 L}{\partial U^2}\right|_{U=u}\frac{\left(\tilde{u}-u\right)^2}{2}\right]. \end{aligned} \tag{2.7}$$

It should be noted that the above equation is possible since the undefined variables, λ, are multiplying terms which are equal to zero, although there is error introduced due to neglecting higher order terms. Now, if u is an appropriately prolonged solution from a discretized to continuous space, i.e. $u = I_h^q u_h$, we have $\tilde{u} - I_h^q u_h = -\epsilon_h$, and so

$$\begin{aligned} J\left(\tilde{u}\right) \approx J\left(I_h^q u_h\right) - \left.\frac{\partial J}{\partial U}\right|_{I_h^q u_h}\epsilon_h + \left.\frac{\partial^2 J}{\partial U^2}\right|_{I_h^q u_h}\frac{\epsilon_h^2}{2}\cdots \\ +\lambda\left[L\left(I_h^q u_h\right) - \left.\frac{\partial L}{\partial U}\right|_{I_h^q u_h}\epsilon_h + \left.\frac{\partial^2 L}{\partial U^2}\right|_{I_h^q u_h}\frac{\epsilon_h^2}{2}\right]. \end{aligned} \tag{2.8}$$

Rewriting the above:

$$\begin{aligned} J\left(\tilde{u}\right) \approx J\left(I_h^q u_h\right) + \lambda L\left(I_h^q u_h\right) - \epsilon_h\left[\left.\frac{\partial J}{\partial U}\right|_{I_h^q u_h} + \lambda\left.\frac{\partial L}{\partial U}\right|_{I_h^q u_h}\right]\cdots \\ +\frac{\epsilon_h^2}{2}\left[\left.\frac{\partial^2 J}{\partial U^2}\right|_{I_h^q u_h} + \lambda\left.\frac{\partial^2 L}{\partial U^2}\right|_{I_h^q u_h}\right] \end{aligned} \tag{2.9}$$

The second term on the RHS is a correction term, the inner product of the dual solution and the continuous primal problem operating on the prolonged solution, which can be shown to be a weighted truncation error. Consider again the GTEE:

$$L_h\left(u\right) = L\left(u\right) + \tau_h\left(u\right) \tag{2.10}$$

If a prolongation of the exact solution to the discretized equations, u_h, is inserted into Equation 2.10, the above then becomes:

$$L\left(I_h^q u_h\right) = -\tau_h\left(I_h^q u_h\right) \tag{2.11}$$

where the left hand side is the continuous residual which approximates the truncation error. Equation 2.9 then becomes

$$J(\tilde{u}) \approx J(I_h^q u_h) - \lambda \tau_h (I_h^q u_h) - \epsilon_h \left[\left. \frac{\partial J}{\partial U} \right|_{I_h^q u_h} + \lambda \left. \frac{\partial L}{\partial U} \right|_{I_h^q u_h} \right] \cdots + \frac{\epsilon_h^2}{2} \left[\left. \frac{\partial^2 J}{\partial U^2} \right|_{I_h^q u_h} + \lambda \left. \frac{\partial^2 L}{\partial U^2} \right|_{I_h^q u_h} \right] \tag{2.12}$$

The adjoint equation, which is the third term on the RHS of Equation 2.12, is solved for λ in order to remove the influence of 1st order discretization error effects on the functional correction, i.e., it drives the first term in brackets to zero. Once the adjoint solution has been obtained, the 2nd term on the RHS acts as the functional error estimate, where the inner product of the adjoint solution and the truncation error seeks to approximate the error between the continuous space and the discrete space caused by solving the discrete equations. While the above derivation has been performed in continuous space, there is generally no simple method to obtain a continuous adjoint solution, and so the the adjoint equations and error estimation are performed in discrete space.

Venditti [2, 8] worked in discrete space where the goal was to obtain a better functional result on a fine grid created by uniformly refining a coarse grid. The use of the embedded grid is based on the idea that it better approximates the continuous space for which a solution is actually sought. The primal and dual problems are both computed on the coarse grid and then reconstructed on the embedded grid in order to obtain the functional estimate and its error estimate (or computable correction) as given by the first and second terms, respectively, on the RHS of

$$J_h(u_h) = J_h\left(I_{2h}^h u_{2h}\right) - \left(I_{2h}^h \lambda_{2h}\right)^T L_h\left(I_{2h}^h u_{2h}\right) + \cdots \tag{2.13}$$

where L_h is the discrete residual and I_{2h}^h is a prolongation operator from grid $2h$ to grid h.

Venditti found that the computable correction term could be more accurately calculated if discretization error and truncation error (or estimates thereof) in both the primal and dual problems were targeted for reduction, rather than adapting on the cell-by-cell contribution to the error estimate. To this end, Venditti developed an adaptation parameter that sought to flag cells where these estimates were too large and needed to be reduced, as will be discussed below. This adaptation parameter was then coupled to a Hessian of a flow quantity, such as the Mach number, in order to create the metric field that described the stretching and node placement throughout the domain. As already alluded to, while this method targets truncation error, the Hessian does not account for how discretization error is produced by the truncation error.

2.4 Solution Improvement via Defect Correction and the Error Transport Equations

As alternatives to the adjoint method, which can only provide a DE for a single functional output per dual solve, both defect correction methods (DC) [9, 10, 11] and the error transport equations (ETEs) [7] provide local DE estimates driven by TE estimates and can be used to estimate error in any functional of interest for a single solve. DC methods treat the TE estimate as a source term to drive the primal solver towards a higher order solution. This TE source term can be thought of in a similar manner to the source term utilized by the method of manufactured solutions (MMS) [12, 13] to drive the discrete governing equations towards a preselected solution. If the exact TE is known, then the discrete governing equations can be driven towards computing the exact solution of the continuous governing equations. In practice, the TE is approximated and the computed solution should converge towards the continuous solution at a higher order rate. DC methods are extremely simple to implement as they only require the formulation of the TE estimate and the ability to include a source term in the discrete solver. In addition, DC methods are generally much less costly to solve than the original discrete system as they can be initialized using the already available discrete solution.

An alternative approach, the discrete ETEs are based on the linearization of the discrete governing equations and form a DE estimate rather than directly computing a higher order solution like the DC method. If the restriction of the exact solution to the continuous governing equations is substituted into the GTEE of Equation 2.10, and $L_h(u_h) = 0$ is subtracted from both sides, we have:

$$L_h(u_h) - L_h(I^h\tilde{u}) = -\tau_h(\tilde{u}). \tag{2.14}$$

If we linearize $L_h(I^h\tilde{u})$ about the discrete solution u_h then we have:

$$L_h(I^h\tilde{u}) = L_h(u_h) - \frac{\partial L_h(u_h)}{\partial u}\epsilon_h + O(\epsilon_h^2). \tag{2.15}$$

This allows us to rewrite Equation 2.14 as:

$$\frac{\partial L_h(u_h)}{\partial u}\epsilon_h = -\tau_h(I_h u_h) + O(\epsilon_h^2), \tag{2.16}$$

where the TE is now approximated using the discrete solution. The ETEs are much more costly to implement than the DC method as they require a full linearization of the discrete operator L_h. Many implicit CFD codes only approximate the full linearization due to the increased storage needs of the full linearization and potential performance reductions, both in terms of stability and cost to iteratively solve the larger system, and would need to be modified. However, if an adjoint solver is already in place, then the full linearization should already be available. Neither method is particularly difficult to solve, as the primal solution

that was obtained to form the TE estimate is available to initialize the DC solve while the ETEs are even cheaper as they are a linear system which is much simpler to solve than the nonlinear DC method.

2.5 Sources of Truncation Error

In order to better understand how truncation error is created, we will use the weak form of the one-dimensional, steady, Burgers' equation as an example

$$L\left(u\right) = \int_{\partial\Omega} \left(\frac{u^2}{2} - \nu \frac{\partial u}{\partial x} \right) \mathrm{d}s = 0 \tag{2.17}$$

This equation can be discretized using the finite volume method with the inviscid flux determined from a central difference and the viscous flux determined through either a central difference or Greens' theorem approximation [14]:

$$L_h\left(u\right) = \frac{1}{x_\xi} \frac{\left(F_{hi+1/2} - F_{hi-1/2}\right)}{\Delta\xi} = 0 \tag{2.18}$$

where

$$F_{hi+1/2} = \frac{\left(\dfrac{u_i + u_{i+1}}{2} \right)^2}{2} - \frac{\nu}{x_\xi\big|_{i+1/2}} \frac{\left(u_{i+1} - u_{i-1}\right)}{\Delta\xi} \tag{2.19}$$

The truncation error for non-uniform meshes was derived by Choudhary and Roy [15] and is given by:

$$\begin{aligned} \tau_h\left(u\right) = {} & \frac{\Delta\xi^2}{8} x_\xi^2 \left(2 \frac{\partial u}{\partial x} \frac{\partial^2 u}{\partial x^2} + \frac{4u_i}{3} \frac{\partial^3 u}{\partial x^3} - \frac{2\nu}{3} \frac{\partial^4 u}{\partial x^4} \right) + \frac{\Delta\xi^2}{8} x_{\xi\xi} \left(4u_i \frac{\partial^2 u}{\partial x^2} + 2 \left(\frac{\partial u}{\partial x} \right)^2 - \frac{8\nu}{3} \frac{\partial^3 u}{\partial x^3} \right) \cdots \\ & + \frac{\Delta\xi^2}{8} \frac{x_{\xi\xi\xi}}{x_\xi} \left(\frac{-5\nu}{3} \frac{\partial^2 u}{\partial x^2} \right) + \frac{\Delta\xi^2}{8} \frac{x_{\xi\xi\xi} x_{\xi\xi}}{x_\xi^3} \left(\frac{\nu}{3} \frac{\partial u}{\partial x} \right) - \frac{\Delta\xi^2}{8} \frac{x_{\xi\xi\xi\xi} x_{\xi\xi}}{x_\xi^2} \left(\frac{\nu}{3} \frac{\partial u}{\partial x} \right) + O\left(\Delta\xi^4\right). \end{aligned} \tag{2.20}$$

As noted in [15], the first term in the above equation is a measure of the inverse of local mesh density, x_ξ, and the second term, $x_{\xi\xi}$, is a measure of the mesh stretching and quality. Wherever the coefficients of these terms are large, the mesh density should be increased and/or the mesh stretching should be decreased. As such, adaptation based on a Hessian of a solution quantity does nothing to directly address the truncation error as they are not designed to address these mesh density versus quality issues.

2.6 Numerical Methods

A cell-centered finite volume scheme is used for the primal solve of the quasi-1D form of the Euler equations. The generic form of the finite volume scheme is given by:

$$\frac{\partial}{\partial t}\int_{\Omega}\vec{Q}\,\mathrm{d}\Omega + \oint_{\partial\Omega}\left(\vec{F}_i - \vec{F}_v\right)\,\mathrm{d}s = \int_{\Omega}\vec{S}\,\mathrm{d}\Omega \tag{2.21}$$

For the quasi-1D Euler equations $\vec{Q} = \{\rho, \rho u, \rho e_t\}^T$, $\vec{F}_i = \{\rho u, \rho u^2 + p, \rho h_t\}^T$, $\vec{F}_v = \vec{0}$, and $\vec{S} = \{0, p\,\partial A/\partial x, 0\}^T$, where the source term is due to the effects of the quasi-1D area change. For these equations ρ is the density, u is the velocity, p is the pressure, $e_t = \frac{p}{\rho(\gamma-1)} + \frac{u}{2}$ and $h_t = \frac{\gamma p}{\rho(\gamma-1)} + \frac{u^2}{2}$ are the total energy and enthalpy, respectively, with a ratio of specific heats of $\gamma = 1.4$ The equations are closed assuming a perfect gas equation of state of $p = \rho RT$, where R is the perfect gas constant and T is the temperature.

Both central and upwind fluxes are available, and a MUSCL scheme with a variety of limiters is used to provide 2^{nd} order accuracy for the upwind schemes [16]. Steady-state solutions are achieved for both systems through the use of an Euler implicit scheme [17].

The dual problem is solved in a discrete manner by using the matrix transpose of the 2^{nd} order linearized residual matrix used by the primal solver, however the limiters used by the primal solver are treated as frozen constants. An implicit time stepping scheme, similar to that used for the primal problem, is used to march the dual problem to steady state as discussed in [18]. While a direct inverse solution process could be used for this simple problem, formulating the time stepping solution method provides a better connection to higher spatial dimension problems which cannot be as easily inverted.

Boundary conditions are weakly enforced through the fluxes. This results in the boundary conditions being naturally included into the linearized adjoint system. Care is taken to ensure that variable extrapolation to the boundaries is enforced in a manner so that downwind information is not upwinded. Figure 2.1 shows the effect of an improper boundary condition where the back pressure for subsonic outflow is set in the ghost cell rather than at the outflow face. While only the adjoint variable for the momentum equation is shown, the improper reconstruction scheme results in an oscillation in all the adjoint variables at the outflow. For supersonic outflow, all variables are extrapolated to the boundary face and no such oscillation exists. At the inflow boundary, the total pressure and total temperature are set at the face and the Mach number is extrapolated from the interior.

2.7 Truncation Error Estimation

While it has been shown that TE acts as the local source of discretization error and can be used to drive adaptation or provide an adjoint based error estimate, a method to estimate

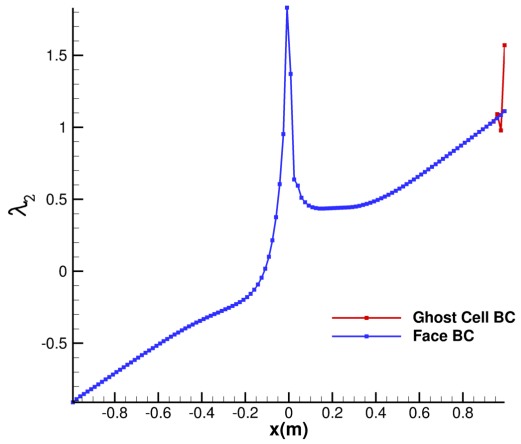


Figure 2.1: Effect of ghost cell and face based boundary conditions on the discrete adjoint.

the TE is needed. TE approximations can be formed by several methods, as reviewed by Phillips [19]; in this work, an embedded grid formulation [20] and a weak formulation via a flux balance are used [21].

The embedded grid approach, as developed by Venditti and Darmofal [8, 20, 22] and used extensively in the literature, (see Fidkowski and Darmofal [23]), operates a fine grid residual on a reconstruction of a coarser grid solution. The residual can then remain on the finer grid or be restricted back to the coarser grid. Both methods can be viewed as a residual estimate on the fine grid, but through knowledge of the formal order of the discretization scheme, the residual estimate can be transformed into an improved TE estimate using a correction term of $r^p/(r^p - 1)$, as per the work of Phillips [19], where r is the refinement factor and p is the formal order of accuracy, and will be applied as part of the adjoint study.

The weak formulation used in this work is based on the concept that the computed solutions should satisfy a flux balance condition. While the computed solutions satisfy a discrete flux balance, prolonging the solution to a continuous space and evaluating the flux balance will produce an approximation of the truncation error. The source term is evaluated using a three point Gauss quadrature in each computational cell. Pierce and Giles [24] utilized a strong formulation which evaluates the continuous residual by evaluating the differential form of the governing equations at three Gauss points over each cell. Due to Green's theorem, the weak and strong forms should produce the same TE estimate given the same reconstruction method and a quadrature capable of integrating the polynomial of interest. The benefit of the weak form is that it avoids the need for differentiating the reconstructions for the convective terms, or if the Navier-Stokes equations are being solved, twice differentiating for the dissipation terms.

No matter which formulation is chosen, a reconstruction of the constant-in-cell finite volume solution is needed. This reconstruction can be formed via either the primitive, conserved, or flux variables, or even a combination of the three. For instance, when the weak formulation is used with a fit of the flux variables, a separate curve fit of the pressure is performed to calculate the source term. Several methods for reconstructing the working variables are examined in this work, including k-exact, piecewise polynomial, and cubic spline. The k-exact reconstruction [25] fits a polynomial such that conservation of the mean is satisfied, which naturally agrees with the finite volume formulation. The piecewise polynomial method is similar to the k-exact method, except that instead of satisfying the conservation of the mean it satisfies the condition that the average cell value is coincident with the cell center. Like the piecewise polynomial curves, the k-exact fits are formed in a moving style, with each cell having a separate curve fit. However, the spline fit is formed in a global sense, as it is defined over the entire domain, or, in the case of a shock, in distinct pre- and post-shock sections. Like the piecewise polynomial method, the average cell value is taken to be coincident with the cell center.

As will be discussed below, an example isentropic expansion and shocked case are examined, with the results on the isentropic example being used to guide the shock study.

2.8 Adjoint Based Grid Adaptation

For this work, the adaptation is provided through a structured adaptation module, SAM, that performs mesh redistribution, also called r-adaptation, via a mass-spring analogy [26]. The library receives the grid and an adaptation parameter and returns a new grid after equi-distributing the adaptation parameter. There is no Hessian specification involved during adaptation, instead the adaptation library seeks to equi-distribute the weight function over the domain.

Two different adaptation parameters (weight functions) have been created and several different methods of estimating the DE in the functional output are studied. The first adaptation parameter is based on Venditti's embedded grid approach [2, 8] and is given by:

$$w = \frac{1}{2}\sum \left\{ \left| \left[I^h_{2h_{HO}} u_{2h} - I^h_{2h_{LO}} u_{2h} \right] L_{h,\lambda} \left(I^h_{2h_{LO}} \lambda_{2h} \right) \right| + \left| \left[I^h_{2h_{HO}} \lambda_{2h} - I^h_{2h_{LO}} \lambda_{2h} \right] L_h \left(I^h_{2h_{LO}} u_{2h} \right) \right| \right\} \tag{2.22}$$

where the summation is over the number of governing equations. The dual and primal solution from a coarser grid are interpolated to an embedded grid, $I^h_{2h_{LO}} u_{2h}$ and $I^h_{2h_{LO}} \lambda_{2h}$, using a lower order reconstruction technique. As discussed above, evaluating the residuals of the primal and dual problems, L_h and $L_{h,\lambda}$, on the reconstructed solutions produce an estimate of the truncation error is formed. The estimated truncation is then weighted by an estimate of the discretization error by finding the difference between a higher order reconstruction and the lower order reconstruction. The weight function attempts to produce a grid that

provides a good solution for both the primal and dual problems by targeting areas of high truncation error or high discretization error. The high order reconstruction methods are the same as discussed above for the truncation error estimates, while the lower order reconstructions, $I^h_{2h_{LO}} u_{2h}$ and $I^h_{2h_{LO}} \lambda_{2h}$, were constructed by taking a simple averaging between the embedded cell faces. Since the indicator exists on the embedded grid, it is restricted to the coarser grid by a simple averaging in order to create the adaptation parameter that is passed to the adaptation routine. Although not currently explored in this work, another option is to perform the adaptation on the embedded grid and then coarsen the adapted grid. Unlike Venditti's work [2, 22], this adaptation indicator is not combined with a flow quantity Hessian to control cell size as SAM simply seeks to equi-distribute a weight function.

The second adaptation parameter is based on using the weak form estimate of the truncation error in the primal problem using only a single grid, as discussed above, with the goal of equi-distributing the TE, i.e, equalizing the TE multiplied by the cell volume, across the domain.

The resulting TE for each equation at each cell, i, is then normalized by the corresponding maximum value of TE over the domain of the starting grid at $t = 1$, and then averaged together to form the weight function such that we have:

$$w_i = \frac{1}{N} \sum \frac{\left| \tau_h \left(I^q_h u_h \right)^t_i \right|}{max \left| \tau_h \left(I^q_h u_h \right)^{t=1} \right|}, \tag{2.23}$$

where the summation is once again taken over the number of equations, N. This TE based method has the benefit that it does not require a solution of the dual problem for every solution of the primal problem, which results in massive time savings, but it can be coupled with an adjoint-based error estimate after the adaptation process is complete.

Due to the use of r-adaptation, the adaptation process is stopped when the values of the weight function multiplied by the cell area on the current mesh vary by no more than 5% of the mean. For the adjoint-based methods, the functional error estimate is also monitored during the adaptation process, and if on the final adapted grid the functional discretization error is more than a certain percentage of the computed functional, it is a sign that the grid must be refined, i.e., more nodes added, in order to meet the required error level. This is also used as an indicator that a finer grid is needed for the TE based adaptation methods; if the adjoint-based correction on the final TE adapted grid is too large, then the grid must be refined.

2.9 Results

For the results given below, the quasi-1D Euler equations are solved using a MUSCL scheme [16] with $\kappa = 1/3$ (corresponding to parabolic reconstructions on a Cartesian mesh), the van Albada flux limiter [27], and Roe's approximate Riemann solver flux scheme [28]. The Mach

number distributions and nozzle geometry are shown in Figure 2.2 for inflow stagnation conditions of $p_o = 300\,kPa$ and $T_o = 600\,K$, and an outflow pressure of $p = 120\,kPa$ for the shocked case. The nozzle is described by a Gaussian bump that avoids curvature discontinuities that can cause TE spikes:

$$A(x) = 1.0 - 0.8e^{-\left(\frac{5^2x^2}{2}\right)}, \quad -1 \leq x \leq 1.$$

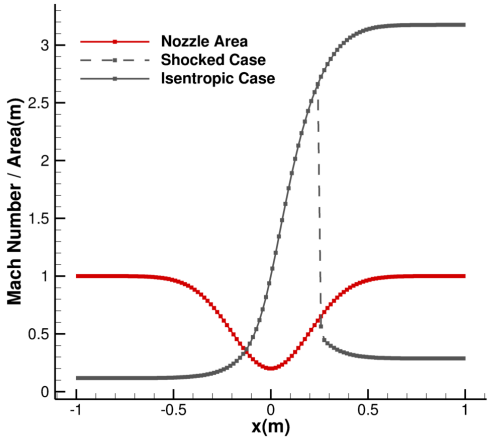


Figure 2.2: Quasi-1D nozzle area and Mach number distributions.

The functional of interest for the adjoint error estimation is the integral of pressure along the nozzle normalized by the inflow stagnation pressure as given by:

$$\int_{-1}^{1} p/p_o \, \mathrm{d}x \tag{2.24}$$

and has a value of 1.0187219 for the isentropic case and 1.2972908 for the shocked case as calculated using a seven point Gauss quadrature of the analytic solution on a grid with approximately 1 million cells. For the discrete value of the integral, the trapezoidal rule is applied to the reconstructed solution.

2.9.1 Isentropic Expansion

As noted above, there are many methods available to prolong the constant-in-cell finite volume solution to a higher order space. A representative test case using 120 cells is examined below for studying the reconstruction schemes and their effect on the TE estimation methods.

Figures 2.3-2.4b show the behavior of the weak TE estimates using primitive, conserved, and flux variables for the momentum equation. The quartic k-exact reconstruction of the

primitive variables clearly outperforms all the other methods with the quartic reconstruction being required to capture the solution curvature and will be retained for the rest of the study. Although not shown here, a quintic reconstruction was also examined and differed only slightly from the quartic reconstruction. Both the conserved variable and flux reconstructions can suffer from non-physical reconstructed states at the faces, as the kinetic energy calculated from the reconstructed density and momentum variables can be greater than the reconstructed total energy. For all cases, the cubic spline and quartic polynomial closely shadowed each other, and had similar derivative values at the faces despite the difference in order.

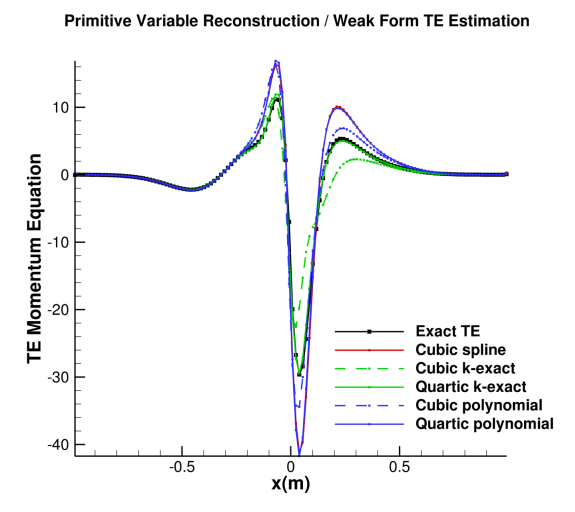
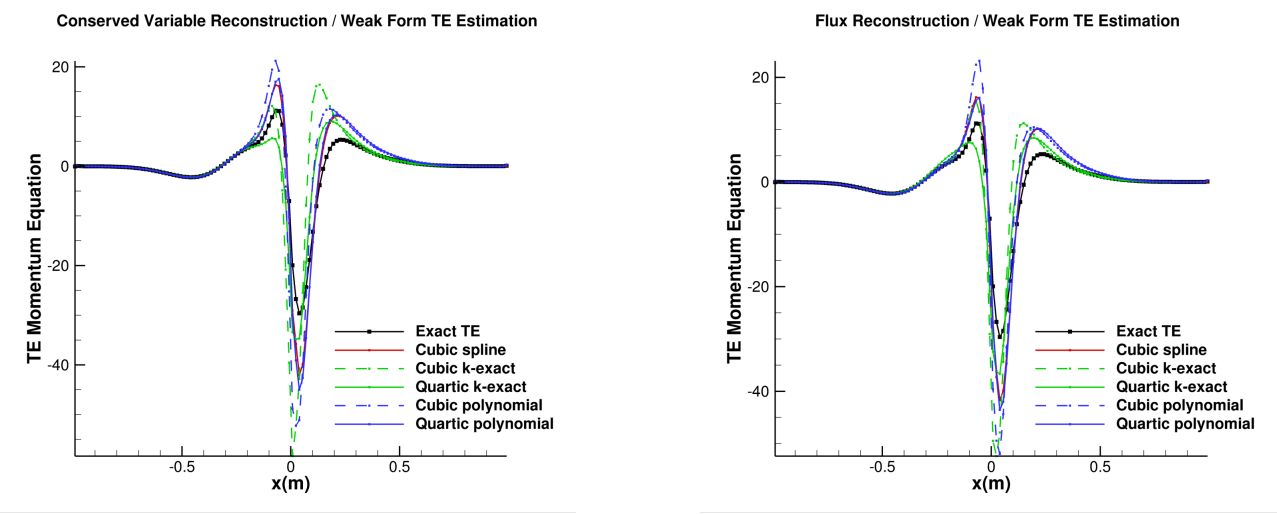


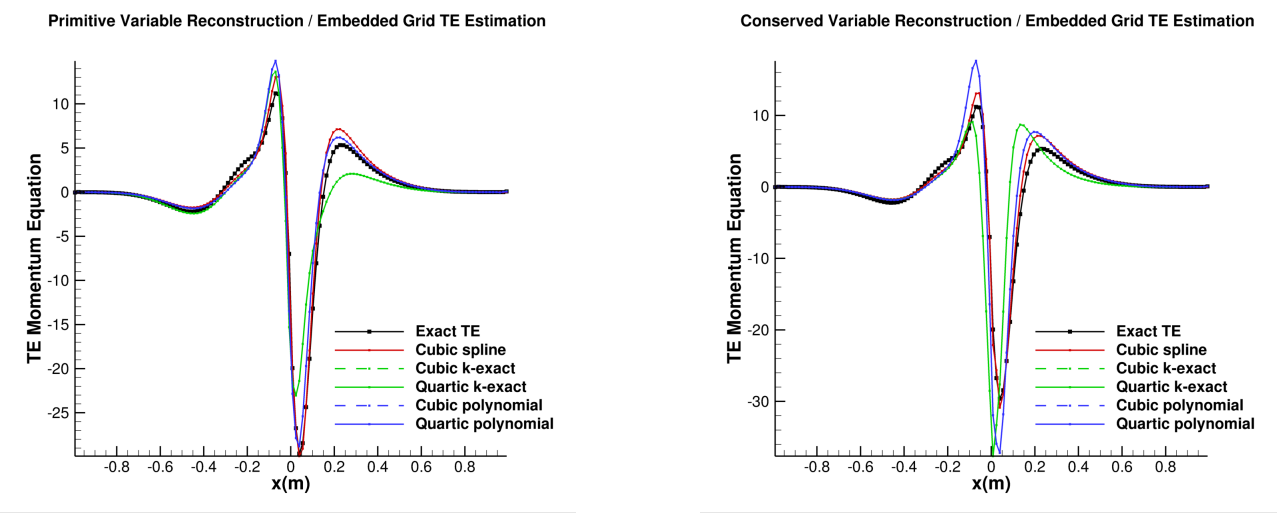
Figure 2.3: Weak formulation TE estimate with primitive variable reconstruction.



(a) (b)

Figure 2.4: Weak formulation TE estimate using reconstructions of conserved variables (a) and flux variables (b).

For the embedded grid approach, Figures 2.5a and 2.5b, the cubic spline approach worked well with reconstructions of both the primitive and conserved variables. The flux variables were not examined for the embedded grid approach as solving for the primitive or conserved variables from the flux value is not as applicable to higher dimensioned problems. For the conserved variables, both the k-exact and polynomial reconstructions overpredicted the peak truncation error by approximately 30% of the true peak. For the primitive variable reconstructions, the k-exact scheme demonstrated underprediction of the TE downstream of the throat location. However, with the correction term of Phillips [19], this method outperforms the cubic spline of the conserved variables. An interesting result is that there is essentially no difference in the TE estimated by the quartic and cubic curve fits for the same reconstruction scheme as the MUSCL reconstruction and limiting procedure is the main driver of this method. Based on these results, the k-exact reconstruction of the primitive variables is also used for the embedded grid method and comparisons will be made both with and without the $r^p/(r^p - 1)$ correction term.



(a) (b)

Figure 2.5: Embedded grid TE estimate using reconstructions of primitive variables (a) and conserved variables (b).

Figure 2.6a shows the behavior of the base solution error and remaining error after adjoint correction for the functional using the exact TE, the weak estimate of the TE, the embedded grid approach, and the corrected embedded grid approach on a series of Cartesian meshes. As expected, the base error converges at a second order rate. The standard embedded grid approach only demonstrates a half order of magnitude reduction in the remaining error with a second order convergence rate, a result seen elsewhere in literature [29]. For the exact TE and weak form estimate, the remaining error converges at an approximately third to fourth order rate, before beginning to level off towards a first to second order rate at finer grid levels. The corrected embedded grid method shows a third order convergence rate and surpasses the weak form TE estimate in a region where the weak form begins to demonstrate first order convergence. This lower order convergence is due to a sign change in the TE which creates a zeroth order spike in the TE, as shown in Figure 2.6c near $x = 0.95\,m$. The discrete residual includes a limiter function which smooths out this oscillation in the embedded grid TE estimate and prevents it from dominating the error estimate. If the domain is shortened to remove the region which contains the TE spike, then the performance of the weak form based adjoint error estimate is improved, as shown in Figure 2.6d, but it now demonstrates a scalloped region due to a change in the sign of the predicted remaining error. It should be noted that the results of Figure 2.6d are obtained through new solutions of the primal and dual problems on the shortened domain instead of simply removing the contributions of the adjoint weighted TE from the cropped region of the domain.

Comparisons between the ETEs, DC, and the adjoint method are made in Figure 2.6b. Most notably, when the exact TE is used, the DC essentially returns the exact functional value, and the the ETEs and adjoint method converge exactly the same. The better performance

of the DC method can be attributed to the fact that it is solving the nonlinear governing equations, while both the adjoint and ETEs are solving linearized problems. A lower order approximation for the ETEs is included where the linearized equations only consists of first order Jacobians. On coarser meshes, the low order formulation of the ETEs will not converge and perform about two orders of magnitude worse the the fully linearized ETEs on the finest mesh. When an approximate TE estimate is used, in this case it is the weak formulation discussed above, the ETEs, DC, and the adjoint method all perform similarly well, but the DC does outperform the others on the coarser mesh levels. Given the similarity in performance, it appears that either the DC method or ETEs could be used in place of an adjoint method in order to provide functional error estimates. Results for the first order ETEs are not included for the approximate TE method as the DE estimates were much worse than the rest.

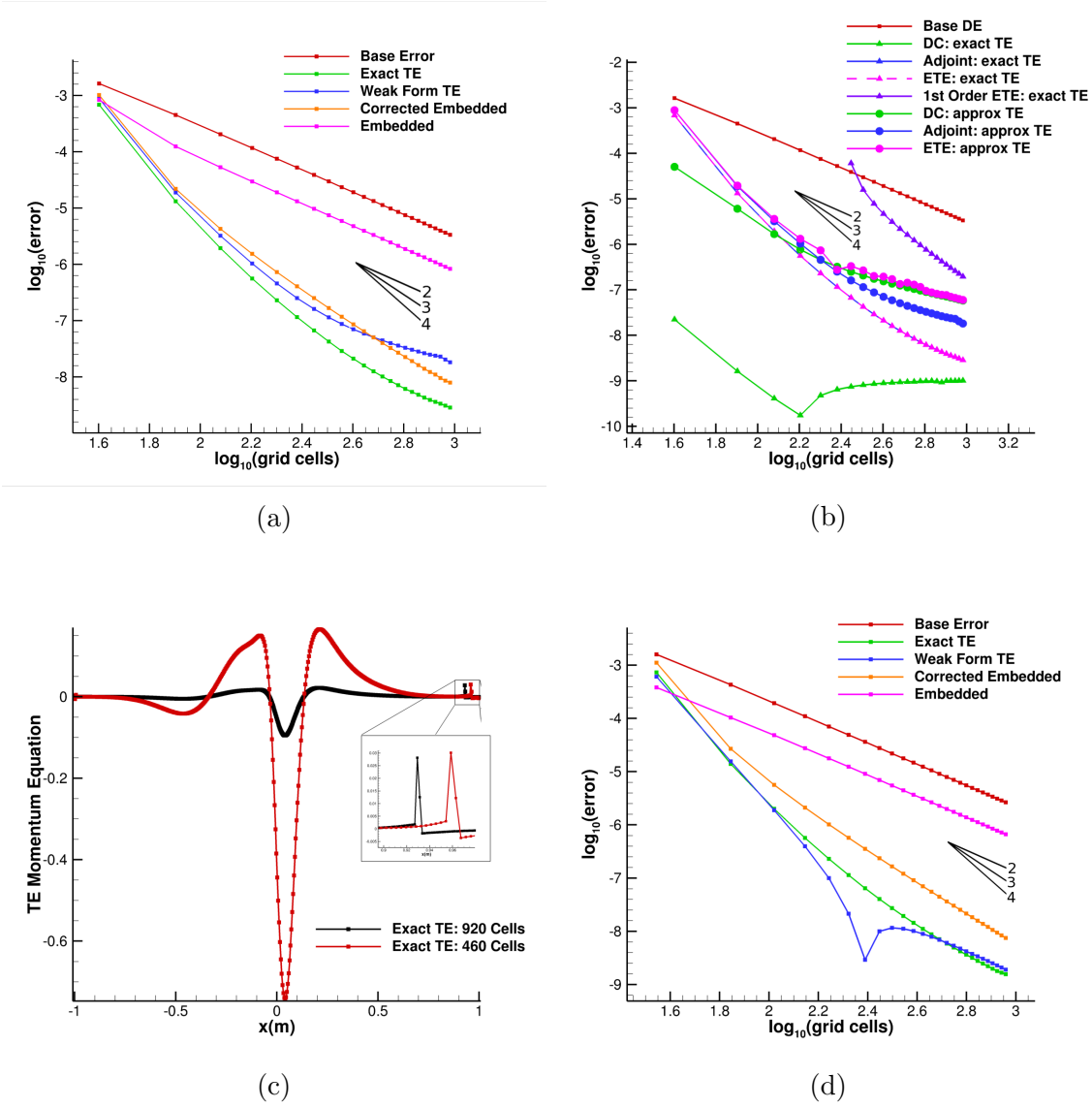


Figure 2.6: Functional base error and remaining error for isentropic expansion for the adjoint method with various TE estimation techniques (a) and for the adjoint, DC, and ETEs in (b). Note the difference in scales. Demonstration of sign change in TE causing TE spike (c) and functional base error and remaining error for cropped domain which removes the TE spike (d).

Adaptation is performed on the 120 cell mesh using both of the adaptation indicators described above. Table 2.1 reports the results and shows that both the adjoint and TE adapted meshes reduce the discretization error in the pressure functional by a factor of three. The resultant error estimates, however, are overly conservative and overestimate the amount of error resulting in higher remaining error. Both adaptation methods served to reduce the

base discretization error in the functional, but it should be noted that the total time from initial grid to final adapted grid for the TE based method was approximately 40% less than the time for the adjoint-based approach due to the extra cost associated with obtaining the adjoint solution.

		Remaining DE		
Mesh	Base DE	Exact TE	Weak Form TE	Corrected Embedded TE
Cartesian	0.020123	0.000190	0.000317	0.000419
Adjoint Adapted	0.006015	0.000431	0.001588	0.001505
TE Adapted	0.005857	0.000007	0.001078	0.001207

Table 2.1: Percent relative error in base error and remaining error after correction for the isentropic test case pressure integral.

In order to test how well a truncation error adapted mesh and the pressure functional adapted mesh can be used to provide error estimates for other functionals, the integral of the entropy along the nozzle is used as a second functional:

$$\int_{-1}^{1} log\left(\frac{p}{\rho^{\gamma}}\right) \mathrm{d}x \tag{2.25}$$

With the ratio of specific heats, $\gamma = 7/5$, the integral is exactly equal to 23.6687230358 Jm/K as based on the inflow conditions. As seen in Table 2.2, the TE adapted mesh reduces the discretization error in the entropy integral by a factor of four, although it has negligible impact on improving the error estimate. The adjoint mesh demonstrates a factor of two reduction in the base discretization error, but performs slightly worse than the TE adapted mesh's remaining errors.

		Remaining DE		
Mesh	Base DE	Exact TE	Weak Form TE	Corrected Embedded TE
Cartesian	0.008064	0.001936	0.001698	0.002094
Adjoint Adapted	0.003900	0.001634	0.001831	0.002172
TE Adapted	0.002148	0.001911	0.001705	0.001890

Table 2.2: Percent relative error in base error and remaining error after correction for the isentropic test case entropy integral.

In order to examine the robustness of the TE estimation methods, the TE adapted mesh was perturbed by 1% of the local cell spacing yielding the results of Figure 2.7. Both the weak formulation and the embedded grid method show some oscillations near the boundaries, but both perform remarkably well even on a non-uniform mesh.

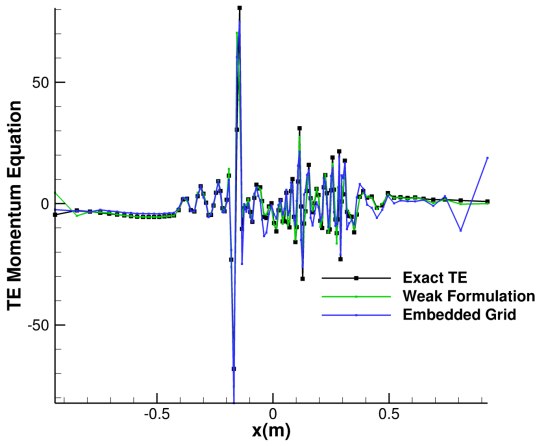


Figure 2.7: Adapted and perturbed mesh exact and estimated truncation error.

2.9.2 Shocked Flow

Based on inviscid compressible flow theory, the shock should be realized as an infinitely thin discontinuity where the conditions on either side of the shock satisfy the Rankine-Hugoniot equations [30]. For the discretized equations, the presence of the truncation error (or artificial dissipation added to stabilize the numerical scheme and acting as TE) tends to diffuse the shock, causing the shock to exist across multiple cells. Generally, one cell of the solution will exist within the shock, and many attempts have been made to create a flux scheme with the ability to provide sharp shock capturing, such as the AUSM schemes [31]. Due to this behavior, the domain is split into three regions in order to provide better reconstructions: pre-shock, shock, and post-shock, but the same TE estimation methods of the isentropic flow study are retained. The shock is detected by determining the location of the maximum Mach number in the domain and then bounding the shock region based on where the sign of forward and backward differences of the Mach number change sign.

It is expected for the solution of the cell within the shock to affect the TE of the surrounding cells which include the shock cell in their stencils. For the 2^{nd} order finite volume schemes of this study, that would be five cells in total. This is shown in Figure 2.8, where the shock is located near $x = 0.25\,m$ and produces TE spikes on the order of a thousand $kg\,m/s^2$ for the momentum equation. The shock has additional effects outside of the TE it produces, it causes a buildup of small numerical errors in front of the shock. These numerical errors affect the TE estimation schemes. Figure 2.8 also shows the weak formulation TE estimates upstream of the shock using both the shocked solution and the isentropic solution patched into the shocked solution. Note the massive oscillations in the TE estimate obtained with the shocked solution. The same method of splitting the the domain into three regions is

retained for both data sets in order to separate the reconstruction method from the solution perturbations, meaning that the oscillations in the TE are purely an artifact of the oscillations in the underlying shocked solution. A more dissipative numerical scheme may be able to remove these oscillations, but at the cost of increased TE. The embedded grid approach does not show these oscillations to as high an amplitude; the MUSCL reconstruction and limiting procedure serves to damp the TE estimates.

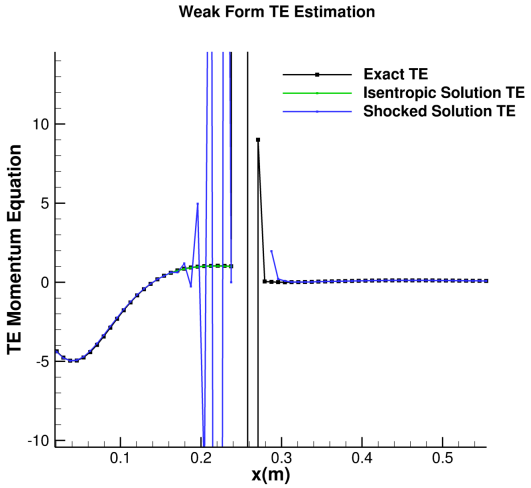


Figure 2.8: Truncation error estimation in presence of shocked flow with quartic, k-exact method. Note that the estimated TE in the shock region has been removed for clarity.

Downstream of the shock, the TE estimates are smooth and track the exact TE well; the weak form TE estimate shows no oscillations. In the shock region, where a simple linear fit is used, the estimated TE tends towards having the same magnitude, but opposite sign, of the exact TE. While it may be unique to this problem, the TE is generally balanced on either side of the shock, and combined with the fact that the derivative of the adjoint solution goes to zero at the shock, this does not harm the adjoint error estimate, at least for the embedded grid method.

Figure 2.9 shows the behavior of the base error and remaining error after correction for the pressure integral. Despite the shock, the base error converges at a 2^{nd} order rate. Once again, the standard embedded grid approach only displays a half order of magnitude reduction in error. The corrected embedded approach initially demonstrates a 3^{rd} order convergence rate, before reducing in order; once again, this is due to round-off errors beginning to become important. Both the exact and estimated TE remaining error show a scalloped behavior resulting from sign changes in the estimated remaining error. While this is initially disheartening, the adapted results of Table 2.3 demonstrate a marked improvement in the the error estimate, where the remaining error after correction is over thirty times lower than the

base error. The adjoint adapted mesh also provides impressive results, with the weak form estimate resulting in a ten times reduction and the embedded grid method showing an 800 times improvement over the base error. While not shown here, the ETEs and DC perform similarly to the adjoint method. Due to the behavior of the exact TE, the DC does not show the same performance as with the isentropic case, but this could perhaps be improved by performing separate solution quadratures on each side of the shock when calculating the exact TE.

Once again, the entropy integral is examined as an alternate functional, with an exact value of 23.9431506435 Jm/K. In all cases, the error estimates on the adapted meshes overpredict the true error, resulting in no error reduction in the entropy functional, with only the weak formulation on the Cartesian mesh showing any error reductions.

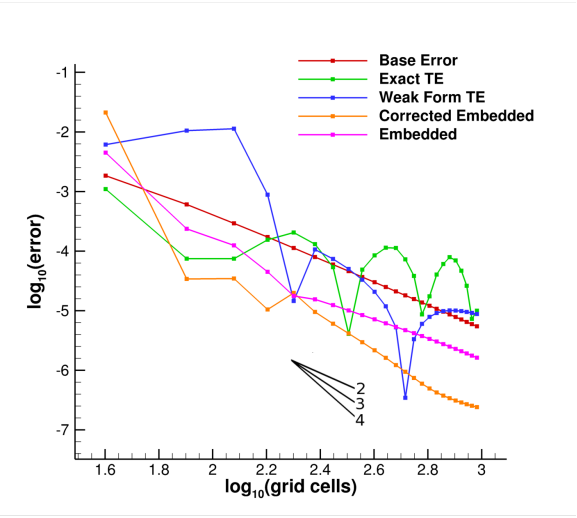


Figure 2.9: Functional base error and remaining error for shocked flow.

		Remaining DE		
Mesh	Base Error	Exact TE	Weak Form TE	Corrected Embedded TE
Cartesian	0.022518	0.005764	0.873686	0.002667
Adjoint Adapted	0.020907	0.000891	0.002093	0.000026
TE Adapted	0.025299	0.000744	0.000763	0.000973

Table 2.3: Percent relative error in base error and remaining error after correction for the shocked test case pressure integral.

		Remaining DE		
Mesh	Base Error	Exact TE	Weak Form TE	Corrected Embedded TE
Cartesian	0.070790	0.073946	0.016294	0.090116
Adjoint Adapted	0.072469	0.074696	0.074151	0.088094
TE Adapted	0.073017	0.070593	0.073564	0.080037

Table 2.4: Percent relative error in base error and remaining error after correction for the shocked test case entropy integral.

2.10 Conclusions

A method has been presented to drive adaptation in order to achieve improved functional error estimates by adapting based on truncation error and then applying a single adjoint-based error estimate. Truncation error based adaptation coupled with a single adjoint solve outperforms purely adjoint-based adaptation for grids of fixed complexity in terms of accurately computing an error estimate between the exact functional and the computed functional. By avoiding the need to calculate an adjoint solution after every adaptation cycle, the TE based adaptation process is significantly faster than the standard adjoint-based method. The new method potentially removes the need to adapt a grid for each functional of interest, as a single grid adapted with the purpose of equi-distributing truncation error and combined with an adjoint-based error estimate can compete with an adjoint adapted grid driven by a single functional. In addition, it was found that the use of k-exact reconstruction techniques are well suited for both estimating truncation error and for creating reconstructions for embedded grid based error estimates for finite volume schemes due to their conservation property, even on perturbed meshes.

For functional DE estimates, the ETEs, DC, and adjoint method all provided similar performance, indicating that both the ETEs and DC could be used as alternatives to the adjoint method. An important benefit of the ETEs and DC is that they can provide error estimates for any functional of interest at the cost of one solution, whereas the adjoint can only target a single functional per dual solve. It should be noted that while the ETEs are significantly less costly to solve than DC, a full linearization of the governing equations is still necessary.

Future work is needed to examine better reconstruction techniques, or the application of solution smoothing, near shocks in order to avoid oscillations. However, the severity of the solution oscillations may only be a factor in one-dimensional problems, where there is limited ability to diffuse these features.

2.11 Bibliography

[1] Warren, G. P., Anderson, W. K., Thomas, J. L., and Krist, S. L., "Grid Convergence for Adaptive Methods," AIAA Paper 1991-1592, AIAA 10th Computational Fluid Dynamics Conference, 1991.

[2] Venditti, D. A., *Grid Adaptation for Functional Outputs of Compressible Flow Simulations*, Ph.D. thesis, Massachusetts Institute of Technology, June 2002.

[3] Park, M. A., Lee-Rausch, E. M., and Rumsey, C. L., "FUN3D and CFL3D Computations for the First High Lift Prediction Workshop," AIAA Paper 2011-936, 49th AIAA Aerospace Sciences Meeting, Orlando, Florida, January 4-7, 2011.

[4] Fidkowski, K. J. and Roe, P. L., "Entropy-based Mesh Refinement, I: The Entropy Adjoint Approach," AIAA Paper 2009-3790, 19th AIAA Computational Fluid Dynamics Conference, San Antonio, Texas, June 22-25, 2009.

[5] Peraire, J., Vahdati, M., Morgan, K., and Zienkiewicz, O., "Adaptive remeshing for compressible flow computations," *Journal of Computational Physics*, Vol. 72, No. 2, 1987, pp. 449–466.

[6] Roy, C. J., "Strategies for Driving Mesh Adaptation in CFD," Invited Paper for Session on Error Estimation and Control, AIAA Paper 2009-1302, 47th AIAA Aerospace Sciences Meeting, Orlando, Florida, January 5-8, 2009.

[7] Phillips, T. S. and Roy, C. J., "Residual Methods for Discretization Error Estimation," AIAA Paper 2011-3870, 20th AIAA Computational Fluid Dynamics Conference, Honolulu, Hawaii, June 27-30, 2011.

[8] Venditti, D. A. and Darmofal, D. L., "Grid Adaptation for Functional Outputs: Application to Two-Dimensional Inviscid Flows," *Journal of Computational Physics*, Vol. 176, No. 1, 2002, pp. 40 – 69.

[9] Fox, L., "Some Improvements in the Use of Relaxation Methods for the Solution of Ordinary and Partial Differential Equations," *Procedings of the Royal Society of London. Series A*, Vol. 190, 1947, pp. pp. 31–59.

[10] Pereyra, V., "Iterated Deferred Corrections for Nonlinear Operator Equations," *Numerische Mathematik*, Vol. 10, No. 4, 1967, pp. 316–323.

[11] Pereyra, V., "Iterated Deferred Corrections for Nonlinear Boundary Value Problems," *Numerische Mathematik*, Vol. 11, No. 2, 1968, pp. 111–125.

[12] Roache, P. J., *Fundamentals of Verification and Validation*, Hermosa Publishers, 2009.

[13] Oberkampf, W. and Roy, C. J., *Verification and Validation in Scientific Computing*, Cambridge University Press, 2010.

[14] Hirsch, C., *Numerical Computation of Internal and External Flows*, Vol. 2: Computational Methods for Inviscid and Viscous Flows, John Wiley & Sons, Ltd, 1990.

[15] Choudhary, A. and Roy, C. J., "Efficient Residual-Based Mesh Adaptation for 1D and 2D CFD Applications," AIAA Paper 2011-0214, 49th AIAA Aerospace Sciences Meeting, Orlando, Florida, January 4-7, 2011.

[16] van Leer, B., "Towards the Ultimate Conservative Difference Scheme. V. A Second-order Sequel to Godunov's Method," *Journal of Computational Physics*, Vol. 32, No. 1, 1979, pp. 101 – 136.

[17] Beam, R. M. and Warming, R., "An implicit finite-difference algorithm for hyperbolic systems in conservation-law form," *Journal of Computational Physics*, Vol. 22, No. 1, 1976, pp. 87 – 110.

[18] Nielsen, E. J., Lu, J., Park, M. A., and Darmofal, D. L., "An implicit, exact dual adjoint solution method for turbulent flows on unstructured grids," *Computers & Fluids*, Vol. 33, No. 9, 2004, pp. 1131 – 1155.

[19] Phillips, T. S., *Residual-based Discretization Error Estimation for Computational Fluid Dynamics*, Ph.D. thesis, Virginia Tech, October 2014.

[20] Venditti, D. A. and Darmofal, D. L., "Adjoint Error Estimation and Grid Adaptation for Functional Outputs: Application to Quasi-One-Dimensional Flow," *Journal of Computational Physics*, Vol. 164, No. 1, 2000, pp. 204 – 227.

[21] Derlaga, J. M., Roy, C. J., and Borggaard, J., "Adjoint and Truncation Error Based Adaptation for 1D Finite Volume Schemes," AIAA Paper 2013-2865, AIAA Co-located Conferences and Exhibit, San Diego, California, June 24-27, 2013.

[22] Venditti, D. A. and Darmofal, D. L., "Anisotropic grid adaptation for functional outputs: application to two-dimensional viscous flows," *Journal of Computational Physics*, Vol. 187, No. 1, 2003, pp. 22 – 46.

[23] Fidkowski, K. J. and Darmofal, D. L., "Review of Output-Based Error Estimation and Mesh Adaptation in Computational Fluid Dynamics," *AIAA Journal*, Vol. 49, No. 4, April 2011, pp. 673–694.

[24] Pierce, N. A. and Giles, M. B., "Adjoint and defect error bounding and correction for functional estimates," *Journal of Computational Physics*, Vol. 200, No. 2, 2004, pp. 769 – 794.

[25] Barth, T. J. and Frederickson, P. O., "Higher Order Solution of the Euler Equations on Unstructured Grids Using Quadratic Reconstruction," AIAA Paper 1990-0013, 28th AIAA Aerospace Sciences Meeting, Reno, Nevada, January 8-11, 1990.

[26] Choudhary, A. and Roy, C. J., "Structured Mesh r-Refinement using Truncation Error Equidistribution for 1D and 2D Euler Problems," AIAA Paper 2013-2444, AIAA Fluid Dynamics and Co-located Conferences and Exhibit, San Diego, California, June 24-27, 2013.

[27] van Albada, G., van Leer, B., and Roberts, W. J., "A comparative study of computational methods in cosmic gas dynamics," *Astronomy and Astrophysics*, Vol. 108, No. 1, 1982, pp. 76–84.

[28] Roe, P., "Approximate Riemann Solvers, Parameter Vectors and Difference Schemes," *Journal of Computational Physics*, Vol. 43, 1981, pp. 357 – 372.

[29] Nemec, M. and Aftosmis, M. J., "Adjoint Error Estimation and Adaptive Refinement for Embedded-Boundary Cartesian Meshes," AIAA Paper 2007-4187, 18th AIAA Computational Fluid Dynamics Conference, Miami, Florida, June 25-28, 2007.

[30] John D. Anderson, J., *Modern Compressible Flow: With Historical Perspective*, McGraw-Hill, 2003.

[31] Liou, M.-S., "A Sequel to AUSM: AUSM+," *Journal of Computational Physics*, Vol. 129, No. 2, 1996, pp. 364 – 382.

Chapter 3

SENSEI Computational Fluid Dynamics Code: A Case Study in Modern Fortran Software Development

Joseph M. Derlaga[a], Tyrone S. Phillips[b], Christopher J. Roy[a]
[a]Department of Aerospace and Ocean Engineering, Virginia Tech, Blacksburg, VA 24061, USA
[b]Department of Mechanical Engineering, University of British Columbia, Vancouver, B.C. V6T 1Z4, Canada

Attribution

The first author (Joseph M. Derlaga) served as the main contributing writer to this work and served as the main code architect, introducing many new features of modern Fortran programming to the coding team and worked closely with the second author (Tyrone S. Phillips) on the development of the core structure of the SENSEI CFD code. Specifically, the first author created a majority of the code necessary to construct the inviscid and viscous residuals, performed the linearization of the residuals and the boundary conditions, created the linear solver suite and its associated polymorphic compressed storage capabilities, and implemented the discrete adjoint solver. The third author (Christopher J. Roy) provided guidance and requirements for the development of the new solver.

Abstract

This paper discusses the development of a new Computational Fluid Dynamics code. SENSEI is a multi-block, structured grid, cell-centered, second-order finite volume solver developed using agile programming techniques and written in Fortran 03/08. The code base integrates the work of several projects into a single system for performing error estimation and grid adaptation and is designed to provide an easily extensible common framework for future development. In addition, it utilizes a new formulation of the method of manufactured solutions ideally suited for solvers which are based on discretizations of the weak form of the governing equations of interest.

3.1 Introduction

Building on previous efforts to create an in-house flow solver, SENSEI (Structured, Euler/Navier-Stokes Explicit-Implicit) is a new Computational Fluid Dynamics (CFD) research code under development in the Aerospace and Ocean Engineering Department at Virginia Tech.

As a research code, SENSEI is designed to be a well structured and easily modifiable code that is built using the benefits of modern programming techniques. In addition, SENSEI provides an interface to several other projects and will serve as a basis for future research. As to whether to use an already existing external code, such as OpenFOAM [1], it was decided that the use of an in house solver would allow for more freedom of modification and allow us to perform order of accuracy studies on a code base we were more familiar with.

This paper will cover the current and proposed features of SENSEI and what makes it unique, and will then discuss how our choice of language and design philosophy affects the clarity, extensibility, and maintainability of the code. Since SENSEI includes both a flow solver (primal solver) and a discrete adjoint solver (dual solver), the theory and background will be discussed separately for each of these in order to discuss how the unique needs of both solvers affect design decisions. In addition, this paper will discuss our coding standard, which provides a standard code format and commonality between code written by different programmers.

3.2 Primal Solver: Theory and Background

SENSEI is a multi-block, structured grid, cell-centered, finite volume code and therefore discretizes the weak form of the Euler and Navier-Stokes equations, as given in Equation (3.1):

$$\frac{\partial}{\partial t}\int_{\Omega} \vec{Q}\,\mathrm{d}\Omega \; + \oint_{\partial\Omega} \left(\vec{F}_i - \vec{F}_v\right)\,\mathrm{d}\vec{s} = \int_{\Omega} \vec{S}\,\mathrm{d}\Omega \tag{3.1}$$

where $\vec{Q}$ is the vector of conserved variables, $\vec{F}_i$ and $\vec{F}_v$ are, respectively, the inviscid and viscous flux contributions as given by:

$$\vec{Q} = \begin{bmatrix} \rho \\ \rho u \\ \rho v \\ \rho w \\ \rho e_t \end{bmatrix}, \quad \vec{F}_i = \begin{bmatrix} \rho V_n \\ \rho u V_n + n_x p \\ \rho v V_n + n_y p \\ \rho w V_n + n_z p \\ \rho h_t V_n \end{bmatrix}, \quad \vec{F}_v = \begin{bmatrix} 0 \\ n_x\tau_{xx} + n_y\tau_{xy} + n_z\tau_{xz} \\ n_x\tau_{yx} + n_y\tau_{yy} + n_z\tau_{yz} \\ n_x\tau_{zx} + n_y\tau_{zy} + n_z\tau_{zz} \\ n_x\Theta_x + n_y\Theta_y + n_z\Theta_z \end{bmatrix} \tag{3.2}$$

$\vec{S}$ can be viewed as a general source term from either body forces, chemistry source terms, axisymmetric flows, or the method of manufactured solutions [2, 3]. For clarity, ρ is density, u, v, and w are the Cartesian velocity components, the total energy is defined as $e_t = \frac{p}{\rho(\gamma-1)} + \frac{u^2+v^2+w^2}{2}$, where γ is the ratio of specific heats, the total enthalpy is defined as, $h_t = \frac{\gamma p}{\rho(\gamma-1)} + \frac{u^2+v^2+w^2}{2}$, and are currently closed by assuming a calorically perfect gas equation of state, $p = \rho RT$, where R is the specific gas constant. $V_n = n_x u + n_y v + n_z w$ and the n_i terms are the components of the outward-facing face normal unit vector. The shear stresses in the viscous flux, assuming Stokes's hypothesis [4], are given by:

$$\begin{aligned} \tau_{xx} &= 2\mu\left(\frac{\partial u}{\partial x} - \frac{1}{3}\nabla\cdot\vec{v}\right) & \tau_{xy} &= \tau_{yx} = \mu\left(\tfrac{\partial u}{\partial y} + \tfrac{\partial v}{\partial x}\right) \\ \tau_{yy} &= 2\mu\left(\frac{\partial v}{\partial y} - \frac{1}{3}\nabla\cdot\vec{v}\right) & \tau_{xz} &= \tau_{zx} = \mu\left(\tfrac{\partial u}{\partial z} + \tfrac{\partial w}{\partial x}\right) \\ \tau_{zz} &= 2\mu\left(\frac{\partial w}{\partial z} - \frac{1}{3}\nabla\cdot\vec{v}\right) & \tau_{yz} &= \tau_{zy} = \mu\left(\tfrac{\partial v}{\partial z} + \tfrac{\partial w}{\partial y}\right) \end{aligned} \tag{3.3}$$

where μ is the dynamic viscosity and $\vec{v} = [u\, v\, w]^T$. The remaining terms in the viscous flux represent the heat conduction and work from the viscous stresses:

$$\begin{aligned} \Theta_x &= u\tau_{xx} + v\tau_{xy} + w\tau_{xz} + k\frac{\partial T}{\partial x} \\ \Theta_y &= u\tau_{yx} + v\tau_{yy} + w\tau_{yz} + k\frac{\partial T}{\partial y} \\ \Theta_z &= u\tau_{zx} + v\tau_{zy} + w\tau_{zz} + k\frac{\partial T}{\partial z} \end{aligned} \tag{3.4}$$

where k is the coefficient of thermal conductivity and T is temperature.

Sensei can solve for flow fields which are either 2D, 3D, or axisymmetric. Two-dimensional grids are treated as 3D grids with unit depth, while cell volumes, Ω, are calculated by decomposing each hexahedron into four tetrahedra and summing the resulting sub-volumes [5]. Face areas and face normals on the boundaries of the control volumes, $\partial\Omega$, are calculated using a cross product of opposite nodes of each face, which is exact when the nodes are co-planar and acts as a best-fit approximation when they are not. Ghost/halo cells are used

to communicate data between blocks and currently require point matching between interfaces. To provide for future extensibilty to larger finite volume stencils, the user may set the number of ghost cells being used, with the default being three. SENSEI is designed to handle grid sequencing and multigrid, although this is not yet fully implemented.

SENSEI currently uses PLOT3D ASCII and Fortran unformatted file types for grid input. ASCII files, while human readable, are slower to load into memory due to their large size. In addition, this can present difficulties when sharing files that can easily be gigabytes in size. Fortran unformatted files are small and more quickly read into memory, but not as portable. Our hope is to make CGNS [6] SENSEI's standard grid system in order to leverage the portability that CGNS provides as well as its small file sizes. The CGNS website [7] has detailed usage instructions and provides some conversion utilities to make the transition easier. In addition, many popular grid generators and post processing tools already support the CGNS standard.

Time marching is accomplished through an explicit M-step Runge-Kutta scheme [8] or a first-order backwards Euler method [9]. Discretizing and rewriting Equation (3.1) in terms of the spatial residual, $\vec{R}$, at time level n, we have:

$$\frac{\Omega}{\Delta t}\Delta\vec{Q} + \vec{R}^n = 0, \tag{3.5}$$

which is linearized as a first-order in time, fully-implicit scheme given by:

$$\left[\frac{\Omega}{\Delta t}I + \frac{\partial\vec{R}}{\partial\vec{Q}}\right]^n \Delta Q^n = -\vec{R}^n \tag{3.6}$$

where I is the identity matrix, $\frac{\partial\vec{R}}{\partial\vec{Q}}$ is the Jacobian of the residual with respect to the conserved variables, and $\Delta Q^n = Q^{n+1} - Q^n$. Since SENSEI operates with, and only stores, the primitive variables, $\vec{q} = [\rho\, u\, v\, w\, p]^T$, the scheme is rewritten as:

$$\left[\frac{\Omega}{\Delta t}\frac{\partial\vec{Q}}{\partial\vec{q}} + \frac{\partial\vec{R}}{\partial\vec{q}}\right]^n \Delta q^n = -\vec{R}^n \tag{3.7}$$

where $\partial\vec{Q}/\partial\vec{q}$ is a conversion Jacobian between the conserved ($\vec{Q}$) and primitive ($\vec{q}$) variables. SENSEI provides a suite of iterative solvers, including GMRES [10], BiCGSTAB [11], and GCROT [12], as well as several others. The linear system described above is generally preconditioned with row and column scaling before a more advanced preconditioner, such as ILU(0) [13] or ILUT [14], is applied. As will be discussed below, the iterative solvers and preconditioners are designed to work with *any* form of compressed matrix storage scheme used to store the sparse left hand side (LHS) created by SENSEI.

Second-order spatial accuracy for the inviscid fluxes is achieved by using MUSCL extrapolation [15] to reconstruct an approximate value of the primitive variables, q_L and q_R, on

each side of each cell face. The MUSCL scheme uses a thirteen point stencil per cell in three dimensions, as shown in Figure 3.1, and provisions are made to approximate the inviscid flux with a number of different inviscid flux functions, $\vec{F}^*$, such that we have $\vec{F}_i \approx \vec{F}^* (q_L, q_R)$ [16]. Currently, Roe's flux difference splitting [17], Steger-Warming flux vector splitting [18], van Leer's flux vector splitting [19], AUSM [20], and AUSM+ [21] are all implemented. The use of MUSCL extrapolation has meaningful impact on the structure of the flux Jacobian matrices discussed above. SENSEI allows for either a full linearization or a first-order approximation of the full flux Jacobian. The full linearization allows for improved convergence rates at the cost of greater storage and the potential to lose diagonal dominance, while the first-order approximation generally yields lower convergence rates, but makes up for it with lower storage requirements and per iteration cost. Regardless of the choice of LHS linearization, the solver will reach essentially the same solution as the physics of the problem is contained in the right hand side (RHS) residual vector. The viscous flux is calculated using a Green's theorem approach to calculate the derivatives at cell faces [16] and central differencing is used to calculate the scalar values. The viscous fluxes require a further twelve points to be added to the inviscid stencil for a total of twenty-five cells in the stencil per cell in three dimensions, as shown in Figure 3.2. While not currently available, the necessary framework to implement turbulence models for the Reynolds Averaged Navier-Stokes (RANS) equations is in place.

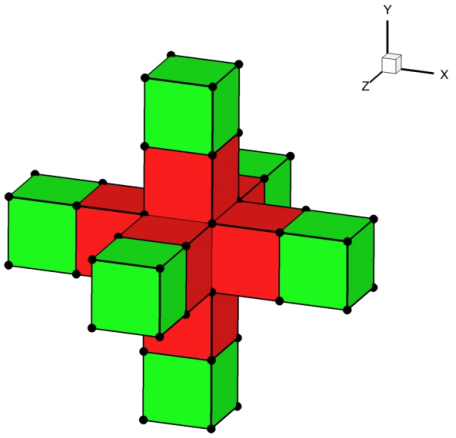


Figure 3.1: Inviscid stencil with first-order cells in red and additional second-order stencil cells in green

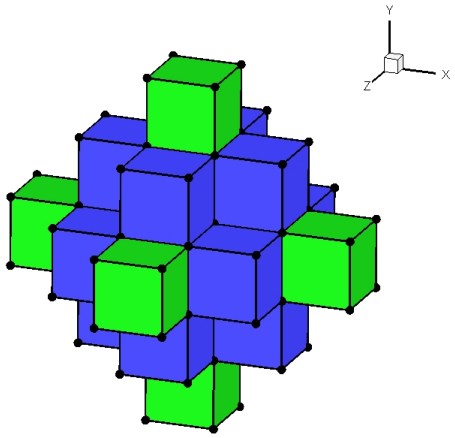


Figure 3.2: Viscous stencil with viscous stencil cells in blue and second-order cells in green

As will be discussed, the code is designed for easy extensibility, and so switching between different models in the code is rather simple. For example, rather than hard wiring the code for a perfect gas, the data structures are written to handle a future extension to gas mixtures.

A variety of boundary conditions are provided within SENSEI. Each block is formulated such

that each cell face on the edge of the block can have a different boundary condition. Ghost cells on interblock boundaries contain data copied from the adjoining block during a syncing routine so that each block can then be solved independently of the others.

Walls can be treated as either slip walls for inviscid flow simulations or no-slip walls with adiabatic or set temperature conditions for viscous simulations. Pressure on the wall is found through either a first-, second-, or third-order extrapolation from the interior. The primitive variables are set in the ghost cells such that the velocities are either mirrored in the case of a slip wall or negated for a no-slip wall so as to be consistent with the interior MUSCL scheme. The flux at the wall can be evaluated by either explicitly setting $V_n = 0$ and calculating only the pressure effects, or by applying the interior flux scheme by utilizing the ghost cell values.

Farfield boundary conditions are based on a Riemann invariant formulation as discussed in Hirsch [16]. A purely supersonic inflow BC is available where all the primitive variables are specified either in the ghost cell, so that the chosen flux function calculates the boundary flux, or at the boundary face by directly setting the flux. A subsonic outflow BC is specified by setting the back pressure and extrapolating all other variables at either first-, second-, or third-order in space.

Since Sensei is a structured grid code, wake cuts can exist in a block. However, due to the multiblock nature of Sensei, this proves to be no difficulty to implement as the wake cut is treated the same as an interblock boundary where all data necessary to form the complete residual is copied into the ghost cells of the appropriate face.

All of the boundary conditions are fully linearized, including wake cuts, even if a first order linearization of the interior flux scheme is specified. However, off block contributions of interblock boundaries are treated as frozen values such that each decomposed grid block solves its own local flow problem without need for a global iterative solve.

Parallelism is currently achieved through OpenMP [22], either over decomposed grid blocks or over loops within blocks. This two level parallelism is made possible by preprocessing the source code using configuration flags. It is planned to expand the parallelism within Sensei to create a hybrid MPI/OpenMP system to work with the HokieOne SGI UV-1000 machine at Virginia Tech. Due to the recent introduction of HokieSpeed, a hybrid CPU-GPGPU machine, plans are underway to make use of general purpose graphical processing units (GPGPU's) in collaboration with the Virginia Tech Math and Computer Science departments through the use of OpenACC [23] and/or the OpenMP 4.0 standard [24]. Current work has already seen GPU offloading via CUDA [25], however, that language is locked to nVidia hardware, while OpenACC and OpenMP allow for use on generic hardware, providing greater platform robustness.

3.3 Code Verification

One of the primary focuses of the research group is on error estimation and uncertainty quantification. A necessary prerequisite for this work is to ensure that SENSEI is coded correctly and demonstrates algorithm consistency. As a result, SENSEI has the method of manufactured solution (MMS) integrated into its code base in order to verify code correctness [26, 2, 3]. There are actually two different formulations of the MMS included in SENSEI, which we will briefly discuss here. Traditional MMS techniques are based on operating the strong, i.e. partial differential equation, form of the governing equations on a manufactured solution, and can require significant effort to differentiate and code the resulting source terms [27], which consist of not only the solution, but in the case of the Navier-Stokes equations, its first and second derivatives as well. The use of the strong form of the governing equations naturally makes it more applicable to finite difference methods, but a more convenient approach for finite volume codes, and any method based on the weak, i.e. integral, form of the governing equations, is to generate the source terms with the weak formulation. This is the second MMS formulation used by SENSEI and is based on calculating the source terms directly through the integral equations of Equation (4.15) through a simple quadrature. This avoids the necessity of creating the second derivatives needed by the strong form of the Navier-Stokes equations, or even first derivatives for the Euler equations. This means that it is far simpler to evaluate the source terms of discontinuous solutions [28] for the purposes of verifying an Euler solver as compared to the traditional MMS formulation. In addition, higher order MMS source terms can be generated by using a higher order quadrature to integrate the fluxes over the cell faces or the body force terms within the cell volumes. SENSEI includes a library to perform Clenshaw-Curtis nested quadrature [29] to easily allow for increasing the order of the quadrature without having to recompute solutions at lower order quadrature points.

From the authors' experience, the weak formulation of the MMS has been far easier to implement than the traditional formulation due to the fewer terms that need to be coded. In addition, the use of higher order quadratures to compute the source terms seems to lead to faster convergence rates during the solution process due to the improved approximation over a simple cell-centered approximation of the average source term over the cell.

Both methods use the same set of generating functions, which are based on either generic polynomials or sinusoidal functions. This allows for a great deal of flexibility in creating the manufactured solution, such that it is a simple matter to create fully subsonic, fully supersonic, or mixed sub- and supersonic solutions.

SENSEI's inviscid Euler solver has been verified second-order accurate based on a variety of test cases. Figure 3.3 demonstrates this on a series of curvilinear meshes with domain sizes varying from 512x512 cells to 8x8 cells for the particular case of a purely supersonic manufactured solution designed to have cross-derivative terms. The numerical method used for this test is a fully upwinded MUSCL method [15] with van Leer's flux vector splitting

scheme [19]. The oscillation in the order of accuracy around the h = 16 mesh is due to the lower discretization error encountered at that mesh level.

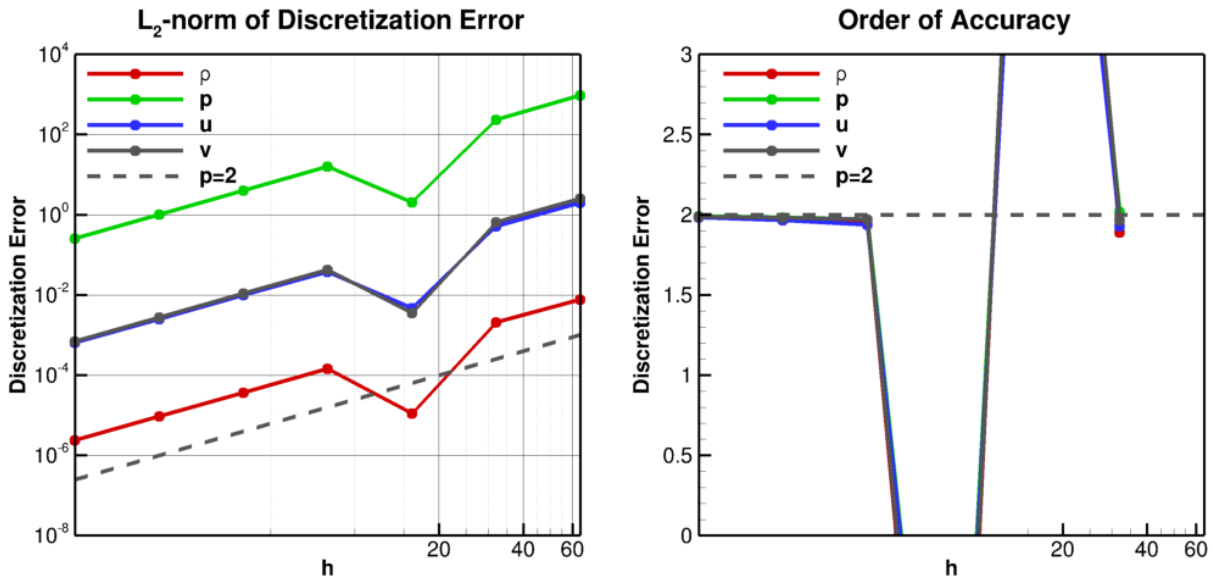


Figure 3.3: Order of Accuracy Verification for 2D Euler Equations.

Methods for estimating truncation error and discretization error, in development from other projects, are built into Sensei. Truncation error, the local source of discretization error [30], has been estimated using a variety of procedures [31, 32]. These truncation error estimates are used in conjunction with the error transport equations [33] to provide discretization error estimates. Methods of estimating the discretization error using Richardson's extrapolation [34] and defect correction [35] are included as well. Estimates of the discretization error in functionals is provided by adjoint-based error estimates [36]. Among the goals for Sensei is to couple these error estimation techniques to a separate adaptation library [37], currently under development. This will allow for a closed loop process to go from an initial grid to a final adapted grid and solution with a specified discretization error tolerance.

3.4 Dual Solver: Theory and Background

Sensei was planned from the start to include an adjoint solver, either in a continuous or discrete form, to be used for error estimation of solution functionals such as lift or drag. Discussed extensively in the literature [38, 39, 40, 41, 42, 43, 44, 45], both adjoint formulations have their pros and cons, and the overall choice of which to implement is up to the programmer. In this case, a discrete adjoint formulation was chosen. Since Sensei was also planned

to use an implicit solver, the Jacobian information necessary for the discrete linearization was already available. In addition, the use of a discrete method simplifies the addition of functionals in the future, since a continuous formulation would require a rederivation of the adjoint boundary conditions for each functional, while a discrete adjoint has the necessary boundary conditions built into the discrete Jacobians.

The adjoint problem can be formed via calculus of variations as a Lagrange multiplier problem. Consider the following

$$L(\vec{q}) = J(\vec{q}) + \lambda^T \vec{R}(\vec{q}) \tag{3.8}$$

where L is the Lagrangian as a function of primitive variables, J is the discrete functional of interest, λ is a row vector of arbitrary Lagrange multipliers, and $R(\vec{q})$ is the discrete residual, which is equal to zero. Differentiating Equation (3.8) with respect to the solution, we have

$$\frac{\partial L}{\partial \vec{q}} = \frac{\partial J}{\partial \vec{q}} + \lambda^T \left[\frac{\partial \vec{R}}{\partial \vec{q}} \right]. \tag{3.9}$$

By equating the above to zero and rearranging, we have the discrete adjoint problem

$$\left[\frac{\partial \vec{R}}{\partial \vec{q}} \right]^T \lambda = -\frac{\partial J}{\partial \vec{q}}, \tag{3.10}$$

where the first term on the LHS is just the transpose of the discrete flux Jacobian matrix, including boundary conditions, used by the primal solver and the RHS is the discrete linearization of the functional of interest, and λ now gives values for the derivative of the functional of interest with respect to perturbations to the discrete residual.

This equation, while linear, would be costly to invert. Therefore, the adjoint is solved for in an iterative manner following Nielsen et al. [42], creating the following iterative form:

$$\left(\frac{\Omega}{\Delta t} + \left[\frac{\partial \vec{R}}{\partial \vec{q}} \right]^T \right)^n \Delta \lambda^n = -\frac{\partial J}{\partial q} - \left[\frac{\partial \vec{R}}{\partial \vec{q}} \right]^T \lambda^n, \tag{3.11}$$

where $\Delta \lambda^n = \lambda^{n+1} - \lambda^n$, which can be solved with any of the iterative solvers discussed above.

It should be noted that the transposed Jacobian matrix on the RHS of Equation (3.11) must be exact, while the matrix on the LHS may be approximated, since, as with the primal solver, the RHS contains the physics of the problem. Unlike the primal solver, the dual solver does include off block Jacobian contributions so that interblock boundaries appear to be just like any other interior face. This is made possible by the use of three ghost cells, which allow the Jacobians contributions of on block cells to off block residuals to be calculated independently. Sensei builds and stores the transposed Jacobian matrix at the beginning

of the adjoint solve in order to avoid recomputing the Jacobians at each time step. While this increases storage, it simplifies the solver since the code used to formulate the LHS of the primal solve can be reused, easing the programming effort and reducing the burden on code maintainability.

3.5 Code Development and Version Control

SENSEI is, and will be, primarily developed and used by graduate students. As such, we are trying to create a code that is easy to add new features to, easy to read and understand ('pseudocode-esque'), and is easy to maintain.

The first step in the development of SENSEI was to choose a version control system (VCS) since part of the success of any software project is the use of a VCS. Version control essentially acts as an archive for code, allowing the programmer to view a code base exactly as it was on a certain day or certain commit.

Our group had been using Apache Subversion (SVN) [46] for several years, mainly for individual projects. Based on the first author's experiences, SVN can be difficult to work with on group projects due to how SVN is designed. SVN is a centralized VCS, where everyone checks their code in and out of a central repository that holds the entire commit history. The method of committing code presents problems when working with others due to every commit being seen by every other developer. Whenever a new feature is being developed, or a current method being modified, every commit, and any bug, is seen by everyone else, and if broken code is committed everyone must deal with it.

As a result, we decided to make the change to Git [47]. Git is a distributed VCS where every local copy of the code is its own repository. We work in an environment where every developer has access to a central repository to share pushed commits, but works in their own local repositories that they can commit code to without fear of affecting another's work. This means that a feature can get fully debugged, and its development stages tracked, without having to send broken code to others. In addition, it is a simple matter to branch and merge code in Git when testing a new feature or idea. While SVN does allow for branching, SVN branches can be difficult to work with and merge back into the mainline code. Over the last several months, the switch to Git has been well received, as it has proven to be easier to work with than SVN.

The second step was to choose the development language, with choices between C, C++, and Fortran 90/95 or 03/08.[1] It was decided to not implement SENSEI using C due to a lack of experience with pure C programming. While the object-oriented programming paradigm of C++ is useful, it can be easily abused and result in difficult to understand and overly

[1]We will follow a convention of referring to Fortran with a major revision/minor extension standard descriptor.

complicated code which would be difficult for future students to learn in a short time period. Even though many abhor Fortran, it is well suited for engineering projects due to its design as a FORmula TRANslation language, making it very simple for most to read and understand the basics of an algorithm. While the authors have the most experience with Fortran 90/95, it was decided to implement Sensei using Fortran 03/08, which supports more modern object-oriented programming paradigms such as polymorphism and has extensions to make C-interoperability easier, and should therefore make it easier to interface with many other external packages written in C/C++.

3.6 Use of Modern Fortran Features

The decision to use Fortran 03/08 was greatly influenced by many of the new, or upgraded, features of the standards [48]. In order to ensure better extensibility and compatibility with other programming languages, Fortran 03/08 provides the `ISO_C_BINDING` interoperability module. As a result, all integer and real types in Sensei are of types compatible with C types. In addition, basic derived types, a feature present since Fortran 90 that allows for grouping related variables of different types, which are declared with the `BIND(C)` attribute can interact with C structs. This eases the issues with interfacing to external C/C++ routines, such as those provided by PETSc [49] and the ForTrilinos Fortran interface to Trilinos [50], or for interacting with GPU sparse solvers written in CUDA C [25]. GPU offloading of the iterative solve has already been accomplished using CUDA ITSOL [51], where all that was needed was a basic Fortran to C interface to handle passing the C bound Fortran data to the solver package.

Sensei essentially uses two main derived types, `grid`, which holds all the geometry and grid information, and `soln`, which holds solution variables. Since Fortran allows for arrays of derived types, the main `grid` and `soln` types contain arrays of subblock derived types specific to each block in the grid, thereby grouping the variables for each block together. These then hold additional derived types that further subdivide and group data. For example, the solution derivatives necessary to create the viscous flux for block `N` of the grid would be stored in `soln%sblock(N)%soln_derivatives`. The grouping of variables into derived types has an important impact on how Sensei is easily parallelized over blocks and, as will be discussed below, how data is passed to routines. In addition, our derived types allow for allocating the minimum amount of memory needed by each grid block. This can not be done with simple arrays-of-arrays without ensuring every block is the same size or without obfuscating the data by placing it in a large work array.

Fortran 03/08 introduced procedure pointers to external, i.e. routines that are not within a `MODULE` or `PROGRAM` declaration such as routines in a library, and internal procedures, and we have found this to be an extremely powerful feature. Often, situations are encountered where the programmer wishes to switch between two or more different functions at run time, for example, different viscosity models, or even switch different functions between subdomains.

One option is to separately program each method and use a decision statement to select between them. These decision statements must be scattered throughout the code and may need to be executed, for example, at every iteration, or even for every computational cell. An option to avoid placing decision statement throughout the main code body is to place it within the function. While this hides the decision statement, it breaks one of the fundamental rules of functions; a function should only do one thing and do it well. Procedure pointers allow a way around this problem. If a standardized interface to each function or subroutine of a certain type can be decided upon, a pointer of `type(procedure)` can be declared and used to point to a specific routine. Different equations of state, flux functions, flux Jacobians, even boundary condition routines, can all be set during initialization rather than on a per iteration, or even per node/cell, basis. While the older Fortran `EXTERNAL` statement could be used to switch procedures, it requires that the code be relinked whenever the procedure is changed rather than simply changing a configuration option at runtime.

In order to examine the impact of using procedure pointers on execution times, several tests were run. A benchmark case was run using several programs performing the same calculations but coded in different ways: 1) a program executing an included function as a baseline (no decision), 2) a program choosing which function to execute based on a decision statement, whether an `IF` or `SELECT CASE` statement with logical, integer, and character variables, and 3) a program setting a procedure pointer to a function. In order to avoid the compiler optimizing out the decision statements, the decision variables were read from an input file at the beginning of programming execution.

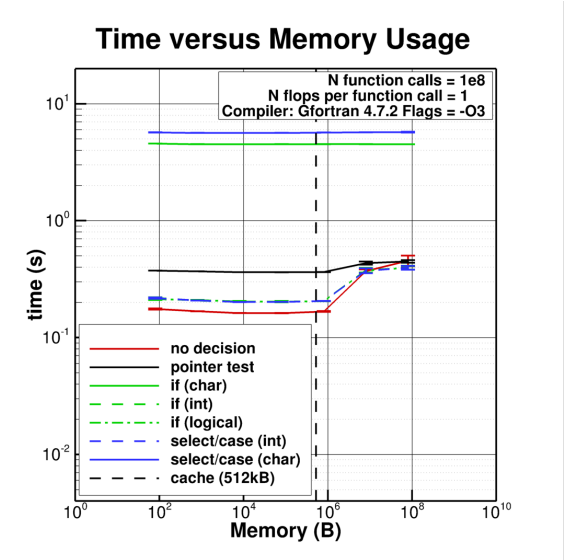


Figure 3.4: Execution time for different memory usage

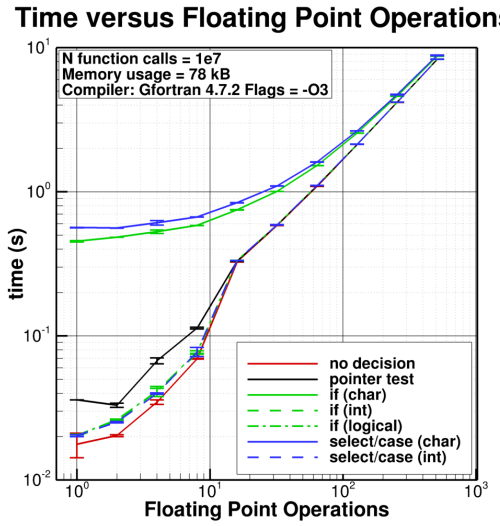


Figure 3.5: Execution time for different FLOP counts

Figure 3.4 shows the results for a test where the number of decisions being made is held constant and the amount of data passed to the function varies and only one floating point

operation is performed within the calling function. This shows the overhead costs that result from the choice of decision statement used. Essentially, any decision made by using a character string is costly, and any integer decision statement is cheap, while the procedure pointer is twice as costly as the integer or logical decision statements. This figure also shows that once the size of the data being passed to the function exceeds the L1 cache size of the processor, there is a dramatic slow down. While it may initially appear that the procedure pointers may create an unacceptable slowdown, there reaches a point where the floating point operations in the called function or subroutine hide the cost of the procedure pointer call. This is demonstrated in Figure 3.5, which shows the result of changing the number of floating point operations within the called function. For small FLOP counts, less than 16 in this example, the procedure pointers cost twice as much time as integer or logical decision statements. However, once a modest FLOP count is reached, each method becomes approximately equal. Once again, character based decision statements are the slowest. We note here that a workaround that maintains the readability of character based decisions while avoiding their slowness is to declare integer variables with names that correspond to the character names and use those in the decision statements, such as the simple example in the appendix.

For further testing, a code used to solve the quasi-1D Euler equations, and which serves as a 1D analogue to Sensei, was refactored to use procedure pointers to select which flux function would be used; i.e. Roe's, van Leer's, etc. It was found that there was at best a slight performance improvement, and at worse no change in performance; the overall benefit comes from the much cleaner reading code. Instead of burying a decision statement inside where the RHS is calculated, it is done up front during the code's initialization phase and the flux calculation looks like `flux = flux_function(ql, qr, normal)`, where only information from the left and right sides of a face and the face normal vector are passed. Choosing which `flux_function` to call has already been decided, and the code looks and reads more like pseudocode. The refactoring was fairly simple, all that was required was to create an interface to the flux function procedures and the procedure pointer itself. See the appendix for a simple example of the procedure interface and references [52] and [53] for further discussion, and examples, of creating and using procedure pointers.

Since Sensei is under active development and not all features have been fully implemented, we have found the need to change interfaces as we better define functionality. Due to our use of unit tests, as discussed below, this has been found to be fairly easy and should hopefully present no difficulty to future developers.

The use of procedure pointers has also allowed us to explore the use of code generation to create highly optimized code. By using procedure pointers, we have already defined sections of code that could be easily inlined due to their common interface. Instead of having the code choose which methods are to be used at run time, different variations of options are automatically generated at compile time by having a preprocessor, such as COG [54], expand code for us. The overall result is code where methods are directly inlined by the compiler and so excess overhead caused by function or subroutine calls is eliminated. The only downsides

are that the code may take longer to compile, the final executable is larger, setting up the sections of code with generator statements can create cluttered code, and properly debugging the resulting executable can be difficult.

The introduction of the `CLASS` attribute to F03/08 has extended Fortran towards an OOP paradigm. Derived types declared as `CLASS` variables are polymorphic and have the ability to contain type bound procedures. This has allowed SENSEI to use any type of sparse matrix storage. SENSEI defines a 'super' class, `matrix`, from which any type of sparse storage scheme can be `EXTENDED`. The `matrix` class includes `DEFERRED` type bound procedures that must be defined by any of the subclasses, such as compressed sparse row (CSR) or jagged diagonal (JAD) sparse matrices. Because any of the subclasses are seen as polymorphic versions of the `matrix` class, the iterative solvers SENSEI uses can be passed any sparse matrix that has extended the `matrix` class. SENSEI is capable of building any sparse matrix scheme due to the fact that each storage format must have the ability to add an array to itself due to a deferred binding. This allows the flux Jacobians to be calculated in a generic array and the storage scheme handles how to add the Jacobian contribution. This has important impacts on the use of SENSEI with GPU's. Compressed storage schemes that may work well on CPU's may not necessarily perform as well on GPU's [51]. If SENSEI was only capable of building CSR matrices, while a GPU iterative solver performs best with a JAD matrix, a conversion would have to occur every time an iterative solve was offloading to the GPU. By using the polymorphic features of F03/08 `CLASS` variables, SENSEI is capable of building whatever storage scheme is best suited to the hardware architecture being used without the need to maintain two copies of the LHS matrix. The abstraction of building the LHS by the use of polymorphic variables greatly simplifies the code base by placing the burden of storing a Jacobian on the `matrix` class, meaning that all the programmer is concerned with is passing the proper location of the Jacobian within the LHS to the type bound procedure. As long as memory is properly allocated for the matrix, SENSEI will work with any sparse storage scheme.

3.7 Maintainability

It has been estimated that most of the time spent developing a code is devoted to debugging the code [3]. Unit testing, the process of writing tests for the simplest building blocks of code to ensure their correctness and regularly checking the code against said tests, not only helps to reduce the time spent debugging, but it changes your approach to writing code. Unit testing is so powerful that is serves as the basis of Test Driven Development, or TDD [55]. Instead of attacking a problem as a whole, a programmer following TDD is forced to think about the problem procedurally. In order to unit test the pieces of their algorithm, the developer must first decide what steps are necessary to solve their problem and how they can be logically grouped. TDD demands that you write your tests first, which will fail because there is no code, and then only writing the code that makes the tests pass.

That being said, we normally do not fully adhere to TDD. Since SENSEI is a research code, we are not always sure what a new algorithm is going to look or act like; but we do know that we want to be able to unit test our code. We typically start by trying to define our interface to a function or subroutine in order to understand the data flow. One of our overall goals is to only pass the data that is needed to a routine and avoid passing whole derived types if possible to low level routines. If a derived type contains many individual components, and the routine only uses one or two pieces, should the whole derived type really be passed? No! Passing a whole derived type can make a routine difficult to test as it requires the hand coded creation of the derived type and all the data contained within. In addition, this will only hurt future code reusability since a derived type available in one routine may not be available in another. When a pattern of only passing certain pieces of a derived type occurs frequently, we may refactor code so to separate those pieces into a better designed derived type. In the end, SENSEI has a series of small, low-level functions and subroutines that are tied together through wrappers that handle funneling only the necessary data down to the routines.

Alongside the code which is compiled into the final executable lies unit tests, snippets of code which test that routines produce the proper output for a given input. These tests are based on the pFunit/f2kunit [56] framework and are compiled into their own separate test suite. Available as part of the NASA Open Source Agreement, pFunit is written entirely in Fortran with support for some Fortran 03/08 features. While there exist other Fortran unit testing harnesses, we have found that pFunit better fits our development style. Since we have unit tests running in our continuous integration system (defined below), we are always assured that our low-level code is correct, at least for what we have written tests for. This means that most of our debugging is not spent on debugging low-level routines, but their interfaces to the rest of the code, which we think is a much easier task. While we do spend time making the unit tests, we view it as time well spent so as to spend less time debugging later. It also leaves a form of documentation for future users and developers of the code; they can see what is going into a routine and what is expected to be returned simply by examining the unit tests.

Part of using unit tests is automating the process through the use of a continuous integration system. Continuous integration is a paradigm where committed code is automatically checked through a gauntlet of tests. Many continuous integration systems exist, such as Buildbot [57] or the CruiseControl [58] system and its derivatives; however, SENSEI uses Jenkins [59], which is derived from Hudson [60]. The basic task flow for our build chain is as follows: Jenkins polls the central repository of SENSEI for changes and pulls a new copy when a change is detected. The first test level simply consists of parsing the code, looking for lines where we list 'FIXME' or 'TODO', and writing them to file. The next level checks to make sure that the code can compile from scratch. This helps to catch files that were not committed, stale modules or object files that allowed the code to compile for the developer but not from scratch, or other basic errors. After this, code documentation is built using Doxygen [61] and uploaded to a central location where the entire research group has access.

The next test level then runs our unit tests. If this passes, both 'optimized' and 'debug' versions of the code are compiled and any warnings are stored so that we can trace them and fix them. We are working to add more test levels, with the highest on our list being for order of accuracy studies.

The method of manufactured solutions (MMS) is a powerful tool to use to test code correctness. While unit tests attempt to verify individual components of our code, system level tests are still necessary to verify the correct behavior of all the individual routines working together. As previously stated, the MMS provides a way to test the code as a whole by using a predefined solution to create a source term that is used to drive the computed solution. If errors are present in the computed solution, or if the code does not reach the expected order of accuracy on a series of systematically refined grids, then it is time to debug and retest until the code is properly verified. It should be noted that many times a code may produce seemingly correct solutions, but not be of the proper order of accuracy.

For almost any code, the process of performing an order of accuracy study can be time consuming. Different source terms must be created for every equation being solved, boundary conditions must also be accounted for, and different grid types must be tested. Once appropriate source terms are created, the code must be run and all of the data compiled and examined. We hope to use our continuous integration system to unload much of this burden, and therefore ensure that the code is always verified.

By using unit testing and a properly configured continuous integration system, we hope to greatly reduce the maintenance burden. However, another factor that affects how easy the code is to maintain is how it looks and reads, and therefore we have developed a coding standard.

3.8 Coding Standard and Software Engineering

One problem with students writing code, especially student engineers who have zero to almost no formal software design experience, is that each individual tends to have wildly different coding experiences and styles. As part of our agile development practices, we have tried to enforce a certain style on the code base in order to try to assure some inherent level of comfort or familiarity with each piece of code. Our standard and guidelines were developed during pair programming sessions so as to reach a consensus on how things should be done instead of having one person dictate. Rather than just stating our standard, we wish to explain the reasoning behind it, as that is what truly matters. A section of example code is included in the appendix.

Sensei uses a free form style since it is written in Fortran 03/08, but we enforce an 80 column limit. While this may seem to some to be a 'legacy' decision, it has proven quite helpful. It helps to eliminate a mix of very long and very short lines that can make reading and comprehending the code difficult. The column limit allows for multiple text editor windows

to be open side by side, which helps improve our work flow and debugging. Limiting the column length prevents leaving long, trailing comments that can be cut off by certain text editors when word wrapping is not present. As a matter of preference, we find that wrapped lines are distracting and having to constantly adjust editor windows to adjust for wrapped lines annoying. Having a set maximum column length avoids these problems entirely.

We do not allow trailing white space, which can cause noise when checking diffs, nor do we allow tab characters, which text editors are free to set to any number of columns and therefore cause a portability nightmare. We also avoid writing in all caps, simply based on stylistic preference, although we use 'CamelCasing' or 'under_scores' where it helps to read long variable names.

All files contain either a module or function, and the file name matches the name of the contained module or function. Functions are allowed to live outside a file if they are to be '`included`' into a module for inlining, but no subroutine is allowed to live outside a module so that the subroutine will have an explicit interface, which allows for all the type, kind, rank, and intent information of the dummy, or formal, arguments to be checked during compilation. SENSEI is developed in a Linux environment, and while learning how to use '`grep`' or '`find`' is fairly easy, most students are more familiar with the GUI systems of Windows or OS X than working with the command line. Therefore, linking module names and file names is a boon to easily navigating the source code. It is especially helpful that we are not constrained by the FORTRAN 77 restriction of 6 character routine names and can therefore use descriptive, self-documenting, routine names. In addition, all derived types, functions, subroutines, and modules have named end statements, which serve as an additional visual indication within a file that a certain procedure has ended since it is easy to miss the three character `END` statement.

We firmly believe that modules should not just act as glorified `COMMON` blocks, but that they are a helpful way to group and organize routines. The entire code base is structured so that the code is built into a series of libraries that are linked together. Our goal here is to be able to create a set of libraries to use as the foundation of other codes within our research group, thereby avoiding needless duplication.

At the top of every module file, `IMPLICIT NONE` is declared to prevent ambiguous variable declarations, and unintended consequences of a variable being accidentally initialized to the wrong type. These consequences can become apparent during order of accuracy testing, where conversions between implicitly typed variables of different precisions, or even types, can cause round-off errors. We note that compiler flags, such as gfortran's `-fimplicit-none` and `-Wconversion`, can catch this during compiling, but think that the use of `IMPLICIT NONE` is an element of good style, and that it is easy to forget to compile with a certain flag.

In order to help understand data flow, since all code is essentially just moving data around, we make use of '`USE <module_name>, ONLY : <sub_1>, <sub_2>, <func_1>`' statements to make it clear where data and routines are coming from. When combined with use of '`PRIVATE/PUBLIC`' clauses in modules, this can have a significant impact on compile time

and code size. In addition, we use '`INTENT`' statements for all dummy arguments so that the programmer can understand what data is available for modification. In order to make this easier to read at a glance, we column-align all the '`INTENT`' statements. As much as possible, we avoid using '`INTENT(INOUT)`' when a variable can better be described as either '`INTENT(IN)`' or '`INTENT(OUT)`' in order to make use of compiler optimizations which improve code performance due to the compiler better understanding how data will or will not be modified. As such, we often refactor code to avoid any '`INTENT(INOUT)`' statements.

All local variables are declared in a certain order, with single logical, integer, real, complex, and character variables followed by arrays of these types, and then derived types. To the authors, this represents a logical flow of variable complexity from booleans to basic through complex numbers and then to strings.

At the top of each routine is a comment block that explains what the routine does. These are comments are formatted such that Doxygen [61], a tool which automatically parses code to produce documentation, can read them. We have found these comment blocks to be helpful when deciding if a routine should be refactored; if the comment block is large and takes a while to write, perhaps the routine itself is too long and complicated. Comments in the code explain why something works, rather than just rehashing what it does. We have found that most routines in Sensei are short enough so that what they do can be easily understood, but why they work is not always clear, especially with hand-optimized code. In addition, the use of named control constructs act as another form of a comment, as they help the programmer follow large chunks of code, such as when there is a multi-page `DO` loop.

Along these lines, we attempt to avoid repetition in the code. This is a self policing policy where we wish to avoid having copies of code go out of date. While most people are very familiar with the code they write, they often are less familiar with what happens in other parts of the code that they have not written. We do not expect a developer five years in the future to have the same inside and out knowledge of Sensei as the original developers. By writing a function or subroutine, the burden of maintenance is less than if a piece of code is copied and pasted, since if a bug is found it only needs to fixed in one place. This also allows us to unit test these pieces of code and prevent bugs in the first place. It also makes it easier for future developers to understand what Sensei is doing since they can work to understand smaller, individual pieces of the code instead of larger, monolithic pieces.

We avoid deprecated features, such as arithmetic `IF` statements, or difficult to follow constructs like `GOTO` statements, especially assigned or computed `GOTO`'s. Numbered `FORMAT` statements are avoided as they always seem to get separated from the code to which they are coupled. Instead, character strings, such as the example in the appendix, are used as they are consistently easier to search for than a number in a computational code. For example, a search for where the `FORMAT` with the label '100' is used can take longer to find than a character string if the code uses 100, 1000, 10000, etc. as part of its computations.

In general, we wish to keep Sensei as up to date as possible and think that old coding practices should be retired as clearer techniques and methods are learned. One new feature of

Fortran 03/08 which is simple, but useful, is the use of square brackets for array declarations. This helps ease the transition for new developers who may be more familiar with using MATLAB [62].

We also make use of 'FIXME' and 'TODO' comments throughout the code to leave reminders of what needs to be revisited or finished. While these could be placed in commit statements, it is too easy to misread a commit message and miss a major problem. Since these comment lines are automatically found by our continuous integration system, it lets each developer know what might not work properly and what needs to be done, as well as serving as a reminder to remove the comments once the work has been completed. We are currently looking into merging these logged messages into our online Wiki system that already shares information on how to build and use SENSEI, and contains the coding guidelines as described above.

The wiki also explains how to use profiling tools and memory checking tools such as the freely available Valgrind [63] and GNU gprof [64] tools. While it is important that the code is easy to read and maintain, we want to make sure that it will not crash and is as fast as possible. Using Valgrind has helped us to find numerous places where memory is not properly allocated and deallocated and helps us to prevent memory leaks. Code profiling shows where hotspots in the code are and indicates areas that may need to be rethought or refactored. When combined with Git branches, we have found it to be easy to create multiple versions of a routine and then profile and test for the fastest.

3.9 Conclusions

The development of a new CFD code has gone smoothly and the work of several projects is being rolled into one common infrastructure for error estimation and grid adaptation. The use of new features provided by Fortran 03/08 has made the coding process much easier. Adding new features has proven to be remarkably simple due to the code structure, its use of clearly defined derived types, and the use of procedure pointers. One member of the research group, who has far more experience in programming C and C++ than with Fortran, has commented on how similar each separate file in the code looks and how much easier it is to read and follow as a result. As another example, our first attempt to run a 3D case (a multi-block ONERA M6 wing with a symmetry plane [65]) after only running 2D cases, required only slight debugging (approximately 2 hours) of an untested 3D multi-block grid matching routine. Everything else in SENSEI just worked, and the time spent developing the code as a 3D solver from the start, and using unit tests, was vindicated. Our use of agile programming practices and a clearly defined coding standard should enable SENSEI to be safely modified and used by future students.

Acknowledgments

The first author would like to thank the members of the NASA Langley Computational AeroSciences Branch for their help in developing his programming skills and for their discussions on the development of CFL3D, FUN3D, ISAAC, OVERFLOW, and VULCAN. 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.

3.10 Bibliography

[1] "The OpenFOAM Foundation," 2012, http://www.openfoam.org/.

[2] Roache, P. J., *Fundamentals of Verification and Validation*, Hermosa Publishers, 2009.

[3] Oberkampf, W. and Roy, C. J., *Verification and Validation in Scientific Computing*, Cambridge University Press, 2010.

[4] Stokes, C., "On the Theories of Internal Friction of Fluids in Motion," *Transactions of the Cambridge Philosophical Society*, Vol. 8, 1845, pp. 287 – 305.

[5] Kordulla, W. and Vinokur, M., "Efficient Computation of Volume in Flow Predictions," *AIAA Journal*, Vol. 21, No. 6, June 1983, pp. 917–918.

[6] "CFD General Notation System: Overview and Entry-Level Document, Version 3.1.2," Tech. rep., CGNS Steering Committee, 2011.

[7] "CFD General Notation System Web page," 2012, http://cgns.sourceforge.net/.

[8] Jameson, A., Schmidt, W., and Turkel, E., "Numerical Simulation of the Euler Equations by Finite Volume Methods using Runge-Kutta Time Stepping Schemes," AIAA Paper 1981-1259, AIAA 5th Computational Fluid Dynamics Conference, 1981.

[9] Beam, R. M. and Warming, R., "An implicit finite-difference algorithm for hyperbolic systems in conservation-law form," *Journal of Computational Physics*, Vol. 22, No. 1, 1976, pp. 87 – 110.

[10] Saad, Y. and Schultz, M., "GMRES: a generalized minimal residual algorithm for solving nonsymmetric linear systems," *SIAM Journal on Scientific and Statistical Computing*, Vol. 7, No. 3, 1986, pp. 856 – 869.

[11] van der Vorst, H. A., "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*, Vol. 13, No. 2, 1992, pp. 631 – 644.

[12] de Sturler, E., "Truncation Strategies for Optimal Krylov Subspace Methods," *SIAM Journal on Numerical Analysis*, Vol. 36, 1999.

[13] Meijerink, J. and van der Vorst, H., "An iterative solution method for linear systems of which the coefficient matrix is a symmetric M-matrix," Vol. 31, No. 137, 1977, pp. 148–162.

[14] Saad, Y., "ILUT: A dual threshold incomplete LU factorization," *Numerical Linear Algebra with Applications*, Vol. 1, No. 4, 1994, pp. 387 – 402.

[15] van Leer, B., "Towards the Ultimate Conservative Difference Scheme. V. A Second-order Sequel to Godunov's Method," *Journal of Computational Physics*, Vol. 32, No. 1, 1979, pp. 101 – 136.

[16] Hirsch, C., *Numerical Computation of Internal and External Flows*, Vol. 2: Computational Methods for Inviscid and Viscous Flows, John Wiley & Sons, Ltd, 1990.

[17] Roe, P., "Approximate Riemann Solvers, Parameter Vectors and Difference Schemes," *Journal of Computational Physics*, Vol. 43, 1981, pp. 357 – 372.

[18] Steger, J. and Warming, R., "Flux Vector Splitting of the Inviscid Gas-Dynamic Equations with Application to Finite Difference Methods," *Journal of Computational Physics*, Vol. 40, 1981, pp. 263–293.

[19] van Leer, B., "Flux Vector Splitting for the Euler Equations," *Proc. 8th International Conference on Numerical Methods in Fluid Dynamics*, Springer Verlag, 1982.

[20] Liou, M.-S. and Steffen, C. J., "A New Flux Splitting Scheme," *Journal of Computational Physics*, Vol. 107, No. 1, 1993, pp. 23 – 39.

[21] Liou, M.-S., "A Sequel to AUSM: AUSM+," *Journal of Computational Physics*, Vol. 129, No. 2, 1996, pp. 364 – 382.

[22] "OpenMP Application Interface: Version 3.1," Api reference manual, OpenMP Architecture Review Board, 2011.

[23] "The OpenACC Application Programming Interface," API Reference Manual Version 1.0, OpenACC, November 2011.

[24] "OpenMP Application Programming Interface, Version 4.0 - RC2 - March 2013, Public Review Release Candidate 2," Api reference manual, OpenMP Architecture Review Board, March 2013.

[25] Sanders, J. and Kandrot, E., *CUDA by Example: An Introduction to General-Purpose GPU Programming*, Addison-Wesley Professional, 2010.

[26] Knupp, P. and Salari, K., *Verification of Computer Codes in Computational Science and Engineering*, Chapman & Hall/CRC, 2003.

[27] Roy, C. J. and Sinclair, A. J., "On the generation of exact solutions for evaluating numerical schemes and estimating discretization error," *Journal of Computational Physics*, Vol. 228, No. 5, 2009, pp. 1790 – 1802.

[28] Grier, B., Alyanak, E., White, M., Camberos, J., and Figliola, R., "Numerical integration techniques for discontinuous manufactured solutions," *Journal of Computational Physics*, Vol. 278, No. 0, 2014, pp. 193 – 203.

[29] Clenshaw, C. and Curtis, A., "A Method for Numerical Integration on an Automatic Computer," *Numerische Mathematik*, Vol. 2, 1960, pp. 197–205.

[30] Roy, C. J., "Review of Discretization Error Estimators in Scientific Computing," AIAA Paper 2010-0126, 48th AIAA Aerospace Sciences Meeting, Orlando, Florida, January 4-7, 2010.

[31] Phillips, T. S., Derlaga, J. M., Roy, C. J., and Borggaard, J., "Finite Volume Solution Reconstruction Methods for Truncation Error Estimation," AIAA Paper 2013-3090, AIAA Fluid Dynamics and Co-located Conferences and Exhibit, San Diego, California, June 24-27, 2013.

[32] Phillips, T. S., *Residual-based Discretization Error Estimation for Computational Fluid Dynamics*, Ph.D. thesis, Virginia Tech, October 2014.

[33] Phillips, T. S. and Roy, C. J., "Residual Methods for Discretization Error Estimation," AIAA Paper 2011-3870, 20th AIAA Computational Fluid Dynamics Conference, Honolulu, Hawaii, June 27-30, 2011.

[34] Richardson, L. F., "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 of London. Series A, Containing Papers of a Mathematical or Physical Character*, Vol. 210, 1911, pp. pp. 307–357.

[35] Pereyra, V., "On improving an approximate solution of a functional by deferred correction," *Numerische Mathematik*, Vol. 8, No. 4, 1965, pp. 376–391.

[36] Derlaga, J. M., Roy, C. J., and Borggaard, J., "Adjoint and Truncation Error Based Adaptation for 1D Finite Volume Schemes," AIAA Paper 2013-2865, AIAA Co-located Conferences and Exhibit, San Diego, California, June 24-27, 2013.

[37] Choudhary, A. and Roy, C. J., "Structured Mesh r-Refinement using Truncation Error Equidistribution for 1D and 2D Euler Problems," AIAA Paper 2013-2444, AIAA Fluid Dynamics and Co-located Conferences and Exhibit, San Diego, California, June 24-27, 2013.

[38] Jameson, A., "Aerodynamic design via control theory," *Journal of Scientific Computing*, Vol. 3, No. 3, 1988, pp. 233–260.

[39] Nadarajah, S. and Jameson, A., "A Comparison of the Continuous and Discrete Adjoint Approach to Automatic Aerodynamic Optimization," AIAA Paper 2000-0667, 38th AIAA Aerospace Sciences Meeting, Reno, Nevada, January 10-13, 2000.

[40] Venditti, D. A. and Darmofal, D. L., "Grid Adaptation for Functional Outputs: Application to Two-Dimensional Inviscid Flows," *Journal of Computational Physics*, Vol. 176, No. 1, 2002, pp. 40 – 69.

[41] Venditti, D. A. and Darmofal, D. L., "Anisotropic grid adaptation for functional outputs: application to two-dimensional viscous flows," *Journal of Computational Physics*, Vol. 187, No. 1, 2003, pp. 22 – 46.

[42] Nielsen, E. J., Lu, J., Park, M. A., and Darmofal, D. L., "An implicit, exact dual adjoint solution method for turbulent flows on unstructured grids," *Computers & Fluids*, Vol. 33, No. 9, 2004, pp. 1131 – 1155.

[43] Balasubramanian, R. and Newman, J., "Adjoint-based error estimation and grid adaptation for functional outputs: Application to two-dimensional, inviscid, incompressible flows," *Computers & Fluids*, Vol. 38, No. 2, 2009, pp. 320 – 332.

[44] Fidkowski, K. J. and Darmofal, D. L., "Review of Output-Based Error Estimation and Mesh Adaptation in Computational Fluid Dynamics," *AIAA Journal*, Vol. 49, No. 4, April 2011, pp. 673–694.

[45] Alauzet, F. and Pironneau, O., "Continuous and discrete adjoints to the Euler equations for fluids," *International Journal for Numerical Methods in Fluids*, Vol. 70, No. 2, September 2012, pp. 135–137.

[46] "Apache Subversion Web page," 2012, http://subversion.apache.org/.

[47] "git Web page," 2012, http://git-scm.com/.

[48] "Information technology - Programming languages - Fortran - Part 1: Base Language," 2010.

[49] Balay, S., Brown, J., Buschelman, K., Gropp, W. D., Kaushik, D., Knepley, M. G., McInnes, L. C., Smith, B. F., and Zhang, H., "PETSc Web page," 2012, http://www.mcs.anl.gov/petsc.

[50] Heroux, M. A., Bartlett, R. A., Howle, V. E., Hoekstra, R. J., Hu, J. J., Kolda, T. G., Lehoucq, R. B., Long, K. R., Pawlowski, R. P., Phipps, E. T., Salinger, A. G., Thornquist, H. K., Tuminaro, R. S., Willenbring, J. M., Williams, A., and Stanley, K. S., "An overview of the Trilinos project," *ACM Trans. Math. Softw.*, Vol. 31, No. 3, 2005, pp. 397–423.

[51] Li, R. and Saad, Y., "GPU-accelerated preconditioned iterative linear solvers," *The Journal of Supercomputing*, Vol. 63, No. 2, 2013, pp. 443–466.

[52] Brainerd, W., *Guide to Fortran 2003 Programming*, Springer-Verlag, 2009.

[53] Markus, A., *Modern Fortran in Practice*, Cambridge University Press, 2012.

[54] Batchelder, N., "Cog," February 2012, nedbatchelder.com/code/cog/.

[55] Beck, K., *Test Driven Development: By Example*, Addison-Wesley Professional, 2002.

[56] "pFunit Web page," 2012, http://opensource.gsfc.nasa.gov/projects/FUNIT/index.php.

[57] "Buildbot Web page," 2012, http://trac.buildbot.net/.

[58] "CruiseControl Web page," 2012, http://cruisecontrol.sourceforge.net/.

[59] "Jenkins Web page," 2012, http://jenkins-ci.org/.

[60] "Hudson Web page," 2012, http://hudson-ci.org/.

[61] van Heesch, D., "Doxygen," Web manual, 2013, www.stack.nl/ dimitri/doxygen/.

[62] MATLAB, *version 7.14.0 (R2012a)*, The MathWorks Inc., Natick, Massachusetts, 2012.

[63] "Valgrind User Manual," Web manual, August 2012, http://www.valgrind.org/docs/manual/index.html.

[64] Fenlason, J. and Stallman, R., *GNU gprof: The GNU Profiler*, January 1993, www.cs.utah.edu/dept/old/texinfo/as/gprof_toc.html.

[65] Slater, J. W., "NPARC Alliance Validation Archive: ONERA M6 Wing," 2008, http://www.grc.nasa.gov/WWW/wind/valid/m6wing/m6wing.html.

Chapter 4

Comparison of Functional Corrections from Defect Correction, Error Transport Equation, and Adjoint Methods for Finite Volume Schemes

Joseph M. Derlaga[a], Tyrone S. Phillips[b], Christopher J. Roy[a], Jeffrey T. Borggaard[c]

[a]Department of Aerospace and Ocean Engineering, Virginia Tech, Blacksburg, VA 24061, USA

[b]Department of Mechanical Engineering, University of British Columbia, Vancouver, B.C. V6T 1Z4, Canada

[c]Department of Mathematics, Virginia Tech, Blacksburg, VA 24061, USA

Attribution

The first author (Joseph M. Derlaga) served as the main contributing writer to this work and performed all the simulations contained within the work. The Sensei CFD code was developed collaboratively between the first and second authors. The adjoint solver was developed by the first author, while the truncation error estimation procedures were implemented by the second author (Tyrone S. Phillips). The second author contributed the defect correction implementation and the original implementation of the error transport equations which were extended to a full linearization of the discrete operator by the first author. The grid adaptation method was contributed by Aniruddha Choudhary. The third (Christopher J. Roy) and fourth (Jeffrey T. Borggaard) authors provided valuable feedback and guidance.

Abstract

In addition to design, control, and optimization applications, adjoint methods can be used to provide discretization error estimation and solution adaptation for solution functionals. Defect correction methods and error transport equations are able to improve discrete solutions or provide discretization error estimates. In this paper, comparisons are made between all three methods for three test cases on a series of uniformly refined meshes. In addition, truncation error and adjoint-based adaptation are performed and the error estimation procedures are compared on the adapted meshes. It has been found that all three discretization error estimation methods perform equally well for the same truncation error estimation procedure.

4.1 Introduction

Solutions obtained through the application of computational fluid dynamics (CFD) are often used to predict engineering functionals of interest, such as the lift and drag on a wing. These solutions are subject to a variety of errors – not only from the assumptions behind the governing equations being solved, but also from the necessary discretization to represent the flow physics in a machine solvable manner – errors which pollute the computed functionals.

The discretization process results in the loss of higher order terms from the continuous governing equations. These lost terms form the truncation error (TE), defined as the difference between the discretized and continuous governing equations, while the discrete solution obtained by solving the truncated equations differs from the true solution of the continuous governing equations by a quantity called the discretization error (DE) [1].

Much effort has been expended on improving computed solutions by using higher order discretizations in order to reduce the truncation error and thereby reduce the discretization error. However, these methods are often more costly to implement and solve than lower order finite volume schemes, and are subject to grid and stability restrictions. An alternative, and perhaps more pragmatic, approach is to attempt to quantify and correct for the effects of truncation error on the computed solutions, leveraging the time and effort already put into developing many lower order CFD solvers. Three common DE estimation procedures driven by TE are defect correction (DC), error transport equations (ETEs), and adjoint methods. DC methods drive a low order method towards a higher order solution and the ETEs directly estimate the DE; both methods result in a locally corrected solution. Adjoint methods target a single functional output, but still require a TE estimate. In order for any of these methods to be truly effective, accurate TE estimates are a necessity.

Additionally, it is well known that mesh adaptation can be used to improve solution accuracy by attempting to efficiently place the mesh to properly resolve the flow field and reduce DE. The problem of how to efficiently place the mesh is normally split into two parts; one which deals with where adaptation is needed (i.e., an adaptation indicator) and another which

deals with how the mesh should be adapted.

Currently, most adaptation routines use either a feature-based or adjoint-based adaptation indicator. Feature-based methods locally target areas of solution peaks, gradients, or curvature and are computationally inexpensive to implement. Unfortunately, while 'interesting' regions are refined, the final solution may be incorrect due to error transport from 'non-interesting' areas of the domain as discussed by Warren et al. [2]. That is, they neglect to refine areas of the flow domain necessary to accurately resolve the relevant flow physics. Adjoint-based methods seek to remedy this by use of the adjoint, or dual, solution to the primal flow problem. Through the use of a Lagrangian, a new equation is formed indicating the regions of the flow domain that impact the calculation of an engineering functional, such as lift or drag in an aerodynamic computation. In Venditti's embedded grid approach [3], the solution to the primal and dual problems are interpolated to a mesh created by uniformly refining an available grid and the result is used to provide an *a priori* estimate of the functional on the embedded grid. As discussed below, the adaptation indicator is formed so that rough estimates of the DE and TE in the primal and dual problems are reduced. It has often been found that, while adjoint-based adaptation does indeed improve the predictions of the functional of interest, other functional outputs may not improve, as shown in the works of Park et al. [4] and Fidkowski [5]. This is in part due to how the second part of the grid adaptation process is addressed.

For unstructured, Cartesian mesh solvers, the adaptation indicator flags cells for simple subdivision or coarsening (h-adaptation). The flagged cells are then uniformly divided/coarsened into smaller/larger ones without changing the underlying grid structure. For structured mesh solvers the mesh is adapted through node movement (r-adaptation), often by seeking to equidistribute a weight function [6]. For unstructured solvers, mesh structure is often determined by the Hessian matrix of a flow quantity. As first described by Peraire [7], the Hessian and adaptation indicator are combined to form a metric that prescribes the mesh density and structure throughout the computational domain. While this system has become the de facto standard for prescribing mesh adaptation [8], the simple Hessian method fails to properly account for how DE is locally generated (via the TE) and transported along with the solution through the domain.

This paper reviews how DE and TE are related for finite volume methods as well as how TE can be estimated. DC, ETEs, and adjoint methods are all reviewed, and a mesh refinement technique based on both the adjoint solution and TE are discussed. Finally comparisons are made between the adjoint, DC, and ETE methods for functional outputs computed using the 2D Euler equations for a number of compressible flows of interest.

4.2 Truncation and Discretization Errors

To better understand the relationship between discretization error and truncation error, it is useful to first examine the generalized truncation error expression, or GTEE [1].

Consider a system of governing equations $L(u) = 0$, that are discretized to form the system $L_h(u) = 0$. These two systems have exact solutions of $\tilde{u}$ and u_h, respectively. The process of discretizing the governing equations produces a system where the discretized equations are equal to the continuous governing equations plus another set of terms called the truncation error, as described by the GTEE, where τ_h is the truncation error:

$$L_h(u) = L(u) + \tau_h(u). \tag{4.1}$$

Note that appropriate restriction and prolongation operators are needed since, in the case of finite volume methods, the discrete equations operate on piecewise constant values while the continuous governing equations operate on continuous, or piecewise continuous, functions. Before proceeding further, it is necessary to define some operators, $I^{\cdot}_{\cdot}$, that indicate interpolation or restriction. For example, I^h_{2h} indicates interpolation from a coarse grid, with cell spacing $2h$, to a fine grid, with cell spacing h, or I^h acting as a restriction from continuous space to a discrete grid with mesh spacing h. An operator such as I^q_h indicates a reconstruction from discrete space to a q-th order polynomial.

If the exact solution to the discrete equations $I_h u_h$ is substituted into the GTEE, and $L(\tilde{u})$ is subtracted from both sides, the GTEE becomes:

$$0 = L(I^q_h u_h) - L(\tilde{u}) + \tau_h(I^q_h u_h). \tag{4.2}$$

If the equations are linear (or linearized), as discussed by Phillips and Roy [9], and with the discrete form of the discretization error defined as $\epsilon_h = u_h - I^h\tilde{u}$, the above becomes:

$$L(I^q_h \epsilon_h) = -\tau_h(I^q_h u_h), \tag{4.3}$$

which is a continuous transport equation for the discretization error. A discrete form of this error transport equation can also be derived that results in:

$$L_h(\epsilon_h) = -\tau_h(\tilde{u}). \tag{4.4}$$

In either case, it is shown that the truncation error acts as the source term for the discretization error, which is transported in the same manner as the flow solution. That is, if the original governing equation contains convective and diffusive terms, then the discretization error is also convected and diffused through the domain. This is why it is important to reduce truncation error in order to improve a computed solution, rather than simply target solution gradients or other flow features. Targeting the truncation error serves as the basis for adjoint-based adaptation methods.

4.3 Truncation Error Estimation

In order to obtain residual-based error estimates, it is necessary to estimate the TE. Phillips et al. [10] uses various methods to reconstruct the piecewise constant solution provided by finite volume solution techniques to a space suitable for use with the integral form of the governing equations. By operating the continuous governing equations on this reconstructed solution, an approximation of the TE is created. In Phillips et al. [11], extensions were made to address non-smooth problems by attempting to adaptively limit the reconstructions in discontinuous regions. In addition, embedded grid techniques were developed, both a method where a solution from a finer grid is restricted to a coarser grid to provide an estimate on the coarser grid and a method where a coarser grid solution is prolonged to a finer grid.

This work focuses on six different TE estimation methods where solution reconstructions are performed over local patches that have been mapped to a logically Cartesian space. The 'kexact' method is a continuous residual method where the discrete solution is prolonged to continuous space using a k-exact method [12] which guarantees conservation of the mean over all cells involved in the fit, i.e., upon integration of the fit over each cell, the value on which the fit was based is recovered. However, the method is adaptively limited near shocks in order to avoid vacuum conditions. The 'lsq' method is similar to the 'kexact' method, except a least-squares problem is solved which only guarantees conservation of the mean for the current cell of interest. The remaining four methods are all based on discrete residual approximations. The 'coarse-grid/cg' and 'coarse-grid-corrected/cgc' methods restrict the current mesh solution to a uniformly coarsened mesh formed by removing every other mesh node. The discrete operator is applied on the coarse mesh and the resulting residual is then prolonged to the original mesh, resulting in a TE estimate. In the 'coarse-grid-corrected' method, a constant based on the formal order of the discrete scheme is used to 'correct' the TE [10]. The 'fine-grid/fg' and 'fine-grid-corrected/fgc' methods prolong the current mesh solution using a k-exact reconstruction to a mesh created by uniformly refining/subdividing the current mesh by a factor of 2. As with the 'coarse-grid' methods, the discrete operator is then applied to the prolonged solution and the resulting residual is then restricted back to the original mesh. As will be discussed below, these TE estimates can be weighted by an adjoint solution to provide DE estimates for functional outputs.

One additional benefit of these new TE estimation schemes is their low memory footprint and potential for parallel scaling. As noted by Park [4], the use of the embedded mesh approach can use significant memory resources, and a TE estimate / adaptive indicator formed on the current mesh can allow for larger grid sizes to be utilized on limited computational resources. In Park's work, the solution was globally reconstructed and the residual was evaluated on the current mesh to form the TE estimate. The methods described above which are based on evaluating a flux balance in continuous space do not require global reconstructions, lowering their memory overhead even further, and can be evaluated in a cell-by-cell manner with the potential for a massively parallel implementation.

4.4 Solution Improvement via Defect Correction and the Error Transport Equations

While globally improving the solution can be accomplished through Richardson extrapolation, DC and ETEs are driven by the TE. DC methods treat the TE estimate as a source term to drive the primal solver towards a higher order solution. This TE source term can be thought of in a similar manner to the source term utilized by the method of manufactured solutions (MMS) [13, 14] to drive the discrete governing equations towards a preselected solution. If the exact TE is known, then the discrete governing equations can be driven towards computing the exact solution of the continuous governing equations. In practice, the TE is approximated and the computed solution should converge towards the continuous solution at a higher order rate. DC methods are extremely simple to implement as they only require the formulation of the TE estimate and the ability to include a source term in the discrete solver. In addition, DC methods are generally much less costly to solve than the original discrete system as they can be initialized using the already available discrete solution.

An alternative approach, the discrete ETEs are based on the linearization of the discrete governing equations and form a DE estimate rather than directly computing a higher order solution like the DC method. If the restriction of the exact solution to the continuous governing equations is substituted into the GTEE of Equation 4.1, and $L_h(u_h) = 0$ is subtracted from both sides, we have:

$$L_h(u_h) - L_h(I^h\tilde{u}) = -\tau_h(\tilde{u}). \tag{4.5}$$

If we linearize $L_h(I^h\tilde{u})$ about the discrete solution u_h then we have:

$$L_h(I^h\tilde{u}) = L_h(u_h) - \frac{\partial L_h(u_h)}{\partial u}\epsilon_h + O(\epsilon_h^2). \tag{4.6}$$

This allows us to rewrite Equation 4.5 as:

$$\frac{\partial L_h(u_h)}{\partial u}\epsilon_h = -\tau_h(I_h u_h) + O(\epsilon_h^2), \tag{4.7}$$

where the TE is now approximated using the discrete solution. The ETEs are much more costly to implement than the DC method as they require a full linearization of the discrete operator L_h. Many implicit CFD codes only approximate the full linearization due to the increased storage needs of the full linearization and potential performance reductions, both in terms of stability and cost to iteratively solve the larger system, and would need to be modified. However, if an adjoint solver is already in place, then the full linearization should already be available. Neither method is particularly difficult to solve, as the primal solution that was obtained to form the TE estimate is available to initialize the DC solve while the

ETEs are even cheaper as they are a linear system which is much simpler to solve than the nonlinear DC method.

4.5 Review of Adjoint Methods

Most CFD simulations are performed to estimate engineering functionals rather than for just obtaining the solution of the dependent variables throughout the entire domain. Due to the TE, the computed functionals are affected by the DE of the solution, as discussed above. When used to provide DE estimates in functionals, adjoint methods generally combine the adjoint variable (which provides the sensitivity of the functional to local equation perturbations) with the TE (or its residual-based estimate). Suppose there is a continuous system of governing equations with an exact solution $\tilde{u}$ and an approximate solution u which is expanded using a Taylor series as given below:

$$L\left(\tilde{u}\right) = L\left(u\right) + \left.\frac{\partial L}{\partial U}\right|_u \left(\tilde{u} - u\right) + \left.\frac{\partial^2 L}{\partial U^2}\right|_u \frac{\left(\tilde{u} - u\right)^2}{2} + \cdots = 0. \tag{4.8}$$

Consider also a functional dependent upon the solution to the system:

$$J\left(\tilde{u}\right) = J\left(u\right) + \left.\frac{\partial J}{\partial U}\right|_u \left(\tilde{u} - u\right) + \left.\frac{\partial^2 J}{\partial U^2}\right|_u \frac{\left(\tilde{u} - u\right)^2}{2} + \cdots . \tag{4.9}$$

Truncating the Taylor series at 2nd order, and combining Equations 4.8 and 4.9 via a Lagrangian, we have:

$$\begin{aligned} J\left(\tilde{u}\right) = J\left(u\right) + \left.\frac{\partial J}{\partial U}\right|_u \left(\tilde{u} - u\right) + \left.\frac{\partial^2 J}{\partial U^2}\right|_u \frac{\left(\tilde{u} - u\right)^2}{2} \cdots \\ + \lambda \left[L\left(u\right) + \left.\frac{\partial L}{\partial U}\right|_u \left(\tilde{u} - u\right) + \left.\frac{\partial^2 L}{\partial U^2}\right|_u \frac{\left(\tilde{u} - u\right)^2}{2} \right]. \end{aligned} \tag{4.10}$$

It should be noted that the above equation is possible since the undefined variables, λ, are multiplying an equation whose solution is 0, i.e., the expansion for $L\left(\tilde{u}\right)$. Now, if u is a numerical solution appropriately prolonged from a discretized to continuous space, i.e. $u = I_h^q u_h$, we have $\tilde{u} - I_h^q u_h = -\epsilon_h$, and so:

$$\begin{aligned} J\left(\tilde{u}\right) = J\left(I_h^q u_h\right) - \left.\frac{\partial J}{\partial U}\right|_{I_h^q u_h} \epsilon_h + \left.\frac{\partial^2 J}{\partial U^2}\right|_{I_h^q u_h} \frac{\epsilon_h^2}{2} \cdots \\ + \lambda \left[L\left(I_h^q u_h\right) - \left.\frac{\partial L}{\partial U}\right|_{I_h^q u_h} \epsilon_h + \left.\frac{\partial^2 L}{\partial U^2}\right|_{I_h^q u_h} \frac{\epsilon_h^2}{2} \right]. \end{aligned} \tag{4.11}$$

Rewriting Equation 4.11 yields:

$$J(\tilde{u}) = J(I_h^q u_h) + \lambda L(I_h^q u_h) - \epsilon_h \left[\left. \frac{\partial J}{\partial U} \right|_{I_h^q u_h} + \lambda \left. \frac{\partial L}{\partial U} \right|_{I_h^q u_h} \right] \cdots + \frac{\epsilon_h^2}{2} \left[\left. \frac{\partial^2 J}{\partial U^2} \right|_{I_h^q u_h} + \lambda \left. \frac{\partial^2 L}{\partial U^2} \right|_{I_h^q u_h} \right]. \tag{4.12}$$

The second term on the RHS is an error estimate, which is the inner product of the dual solution and the continuous primal problem operating on the prolonged solution, which can be shown to be a weighted TE. By the use of the GTEE, we can replace $L(I_h^q u_h)$ with the TE to get:

$$J(\tilde{u}) = J(I_h^q u_h) - \lambda \tau_h(I_h^q u_h) - \epsilon_h \left[\left. \frac{\partial J}{\partial U} \right|_{I_h^q u_h} + \lambda \left. \frac{\partial L}{\partial U} \right|_{I_h^q u_h} \right] + \frac{\epsilon_h^2}{2} \left[\left. \frac{\partial^2 J}{\partial U^2} \right|_{I_h^q u_h} + \lambda \left. \frac{\partial^2 L}{\partial U^2} \right|_{I_h^q u_h} \right]. \tag{4.13}$$

This error estimate can also be used as a correction term, where the remaining term should be on the order of the DE squared, meaning that for a second order discretization, the remaining error should reduce at a fourth order rate. The adjoint equation, which is the third term on the RHS of Equation 4.13, is solved for λ in order to remove the influence of DE effects on the functional correction. Once this is done, the 2nd term on the RHS acts as the error estimate, where the inner product of the adjoint solution and the TE seeks to approximate the error between the continuous space and the discrete space caused by solving the discrete equations. While the above derivation has been performed in continuous space, there is in general no simple method to obtain a continuous adjoint solution, and so the the adjoint equations and error estimation are performed in discrete space.

Venditti [3] and Venditti and Darmofal [15] worked in discrete space where the goal was to obtain a better functional result on a fine grid created by uniformly refining a coarse grid. The use of the embedded grid is based on the idea that it better approximates the continuous space for which a solution is actually sought. The primal and dual problems are both computed on the coarse grid and then reconstructed on the embedded grid in order to obtain the functional estimate and its error estimate (or computable correction) as given by the first and second terms, respectively, on the RHS of:

$$J_h(u_h) = J_h\left(I_{2h}^h u_{2h}\right) - \left(I_{2h}^h \lambda_{2h}\right)^T R_h\left(I_{2h}^h u_{2h}\right) + \cdots \tag{4.14}$$

where R_h is the discrete residual, as described in Section 4.6 and I_{2h}^h is a prolongation operator from grid $2h$ to grid h. It should be noted that this current work takes a slightly different approach, where we seek to approximate the continuous functional value by using TE estimates.

Venditti [15] found that the computable correction term could be more accurately calculated if DE and TE (or estimates thereof) in both the primal and dual problems were targeted in order to decrease the error terms caused by approximating the actual embedded grid solution with the reconstructed coarse grid solution. To this end, Venditti developed an adaptation parameter that sought to flag cells where these estimates were too large and needed to be reduced, as will be discussed below. This adaptation parameter was then coupled to a Hessian of a flow quantity such as the Mach number in order to create the metric field that described the stretching and node placement throughout the domain. As already discussed, while this method targets TE, the Hessian does not directly account for how DE is produced by the TE.

4.6 Numerical Methods

The code used for the study, SENSEI, has been developed in the Department of Aerospace and Ocean Engineering at Virginia Tech [16]. SENSEI is a multi-block, structured grid, cell-centered, finite volume code and therefore discretizes the weak form of the Euler and Navier-Stokes equations. SENSEI solves equations of the form

$$\frac{\partial}{\partial t}\int_{\Omega}\vec{Q}\,\mathrm{d}\Omega\;+\oint_{\partial\Omega}\left(\vec{F}_i-\vec{F}_v\right)\,\mathrm{d}s=\int_{\Omega}\vec{S}\,\mathrm{d}\Omega, \tag{4.15}$$

where $\vec{Q}$ is the vector of conserved variables, $\vec{F}_i$ and $\vec{F}_v$ are, respectively, the inviscid and viscous flux contributions as given by:

$$\vec{Q}=\begin{bmatrix}\rho\\ \rho u\\ \rho v\\ \rho w\\ \rho e_t\end{bmatrix},\qquad \vec{F}_i=\begin{bmatrix}\rho V_n\\ \rho u V_n+n_x p\\ \rho v V_n+n_y p\\ \rho w V_n+n_z p\\ \rho h_t V_n\end{bmatrix},\qquad \vec{F}_v=\begin{bmatrix}0\\ n_x\tau_{xx}+n_y\tau_{xy}+n_z\tau_{xz}\\ n_x\tau_{yx}+n_y\tau_{yy}+n_z\tau_{yz}\\ n_x\tau_{zx}+n_y\tau_{zy}+n_z\tau_{zz}\\ n_x\Theta_x+n_y\Theta_y+n_z\Theta_z\end{bmatrix}, \tag{4.16}$$

$\vec{S}$ can be viewed as a general source term from either body forces, chemistry source terms, the method of manufactured solutions, or a TE estimate for the DC method. For clarity, ρ is density, u, v, and w are the Cartesian velocity components, the total energy is defined as $e_t=\frac{p}{\rho(\gamma-1)}+\frac{u^2+v^2+w^2}{2}$, where γ is the ratio of specific heats, the total enthalpy is defined as $h_t=\frac{\gamma p}{\rho(\gamma-1)}+\frac{u^2+v^2+w^2}{2}$, and are currently closed by assuming a calorically perfect gas equation of state, $p=\rho RT$, where R is the specific gas constant. The normal velocity is $V_n=n_x u+n_y v+n_z w$ and the n_i terms are the components of the outward-facing face normal unit vector. The shear stresses in the viscous flux, assuming Stokes's hypothesis, are given

by:

$$\begin{aligned}
\tau_{xx} &= 2\mu\left(\frac{\partial u}{\partial x} - \frac{1}{3}\nabla\cdot\vec{v}\right) \quad \tau_{xy} = \tau_{yx} = \mu\left(\tfrac{\partial u}{\partial y} + \tfrac{\partial v}{\partial x}\right) \\
\tau_{yy} &= 2\mu\left(\frac{\partial v}{\partial y} - \frac{1}{3}\nabla\cdot\vec{v}\right) \quad \tau_{xz} = \tau_{zx} = \mu\left(\tfrac{\partial u}{\partial z} + \tfrac{\partial w}{\partial x}\right), \\
\tau_{zz} &= 2\mu\left(\frac{\partial w}{\partial z} - \frac{1}{3}\nabla\cdot\vec{v}\right) \quad \tau_{yz} = \tau_{zy} = \mu\left(\tfrac{\partial v}{\partial z} + \tfrac{\partial w}{\partial y}\right)
\end{aligned} \tag{4.17}$$

where μ is the dynamic viscosity and $\vec{v} = [u\, v\, w]^T$. The remaining terms in the viscous flux represent the heat conduction and work from the viscous stresses:

$$\begin{aligned}
\Theta_x &= u\tau_{xx} + v\tau_{xy} + w\tau_{xz} + k\frac{\partial T}{\partial x} \\
\Theta_y &= u\tau_{yx} + v\tau_{yy} + w\tau_{yz} + k\frac{\partial T}{\partial y} \\
\Theta_z &= u\tau_{zx} + v\tau_{zy} + w\tau_{zz} + k\frac{\partial T}{\partial z}
\end{aligned} \tag{4.18}$$

where k is the coefficient of thermal conductivity and T is temperature.

Time marching is accomplished through either an explicit M-step Runge-Kutta scheme [17] or a linear solve of an implicit time stepping scheme [18]. Rewriting Equation (4.15) in terms of the residual, $\vec{R}$, at time level $n+1$, we have:

$$\frac{\Omega}{\Delta t}\Delta\vec{Q} + \vec{R}^{n+1} = 0. \tag{4.19}$$

The expression above can be linearized as a first order in time, fully-implicit scheme given by

$$\left[\frac{\Omega}{\Delta t}I + \frac{\partial\vec{R}}{\partial\vec{Q}}\right]^n \Delta\vec{Q}^n = -\vec{R}^n, \tag{4.20}$$

where I is the identity matrix, $\frac{\partial\vec{R}}{\partial\vec{Q}}$ is the Jacobian of the residual with respect to the conserved variables, and the Δ represents a forward difference operator, i.e., $\Delta(\bullet)^n = (\bullet)^{n+1} - (\bullet)^n$. Since Sensei operates with, and only stores, the primitive variables, $\vec{q} = [\rho\, u\, v\, w\, p]^T$, the scheme is rewritten as:

$$\left[\frac{\Omega}{\Delta t}\frac{\partial\vec{Q}}{\partial\vec{q}} + \frac{\partial\vec{R}}{\partial\vec{q}}\right]^n \Delta\vec{q}^n = -\vec{R}^n, \tag{4.21}$$

where $\partial\vec{Q}/\partial\vec{q}$ is a conversion Jacobian between the conserved ($\vec{Q}$) and primitive ($\vec{q}$) variables.

Second order spatial accuracy for the inviscid fluxes is achieved by using MUSCL extrapolation [19], which uses a thirteen point stencil per cell in three dimensions, and provisions

are made to support a number of different inviscid flux functions. Limiting is applied to the primitive variables in order to maintain monotonicity [20]. The viscous flux is calculated using a Green's theorem approach to approximate the derivatives at cell faces [20] and central differencing is used to calculate the scalar values. The viscous fluxes require a further twelve points to be added to the inviscid stencil, for a total of twenty-five cells in the stencil for a fully 2^{nd} order method.

The dual solve is accomplished in a similar manner to the implicit primal solve, where the adjoint equations are discretized as [21]:

$$\left[\frac{\Omega}{\Delta t}\frac{\partial \vec{Q}}{\partial \vec{q}} + \frac{\partial \vec{R}}{\partial \vec{q}}\right]^T \Delta\vec{\lambda}^n = -\left(\frac{\partial J}{\partial \vec{q}} + \left[\frac{\partial \vec{R}}{\partial \vec{q}}\right]^T \vec{\lambda}^n\right). \tag{4.22}$$

It should be noted that the residual Jacobian on the RHS of (4.22), $\frac{\partial \vec{R}}{\partial \vec{q}}$, must be the full second order linearization of the residual operator, while the residual Jacobian on the LHS may be of lower order for computational efficiency. This is due to how the RHS holds the physics of the numerical problem, while the LHS is simply a method to drive the RHS to zero. The availability of the full linearization makes the implementation of the ETEs possible, although it must be noted that special care is needed at grid block boundaries where off-block contributions are typically treated as frozen conditions.

4.7 Grid Adaptation

For this work, the adaptation is provided through a structured adaptation module, SAM, that performs mesh redistribution, also called r-adaptation, via a center-of-mass approach [22]. The library receives the grid and an adaptation parameter and returns a new grid after attempting to equidistribute the adaptation parameter. It should be noted that there is no Hessian specification involved during adaptation, instead the adaptation library seeks to equidistribute the weight function over the domain.

In this work, the adaptation parameters are based on estimates of the TE, with and without adjoint weighting. The first adaptation parameter is based on an estimate of the TE in the primal problem using only a single grid with the goal of evenly distributing the TE across the domain, thereby minimizing the creation of DE. The 'fg' method was utilized due to its robustness in the face of deforming meshes that occur during the adaptation process due to the limiting process of the discrete operator. The resulting TE for each equation at each cell, i, is then normalized by the corresponding maximum value of TE over the domain of the starting grid at $t = 1$ and then averaged together to form the weight function such that

we have:

$$w_i = \frac{1}{N} \sum \frac{\left| \tau_h \left(I_h^q u_h \right)_i^t \right|}{max \left| \tau_h \left(I_h^q u_h \right)^{t=1} \right|}, \tag{4.23}$$

where N is the number or equations being solved. The second method is similar to the first, except that the TE is weighted by the adjoint variable in order to provide adaptation only to areas where the TE is important to the functional of interest. This is a similar adaptation indicator to that used by Nemec and Aftosmis [23], and the weight function is normalized in a similar manner as Equation 4.23.

The first method has the benefit that it does not require a solution of the dual problem for every solution of the primal problem, which results in massive time savings, but the resulting mesh and solution can of course be coupled with an adjoint-based error estimate after the adaptation process is complete. In addition, the first method could be used to gradually adapt the mesh as part of the solution process, although that was not done for this study.

4.8 Results

Three 2D Euler test cases, specifically chosen for their different convergence rate behaviors, have been studied: the supersonic vortex problem [24], a Mach 2 expansion over a 10 degree corner, and a Mach 2 compression by a 10 degree wedge. For each case a uniform grid refinement study was performed with grids ranging in size from 32x16 to 512x256 cells for the supersonic vortex problem, and 32x32 to 512x512 cells for the other two cases. The TE was estimated using reconstructions of degree 2 for the uniform refinement tests. For the supersonic vortex, both lift and drag functionals were examined, while the expansion fan and shock case used a normal (to the x-axis) force functional. Qualitative comparisons are made of the DE distributions predicted by the DC method and ETEs (formulated with both a fully linearized, and a lower order approximation, of the discrete operator). For these comparisons, the DE estimation procedures are driven by the 'kexact' and 'fgc' TE estimation methods, which had previously been shown to be the best performing TE estimation procedures [11]. Mesh adaptation through equidistribution was performed on the 64x32 cell grid for the supersonic vortex problem and the 64x64 cell grid for the other two cases, all using the 'fg' method for TE estimation. For the adapted meshes, reconstructions of degree 1, 2, 3, and 4 were used in order to examine the impact of the reconstruction on the TE in the presence of non-smooth mesh transformations.

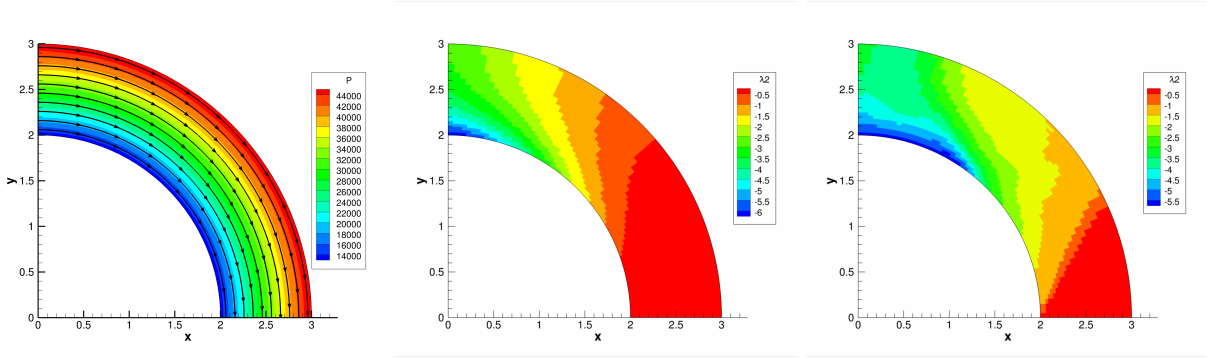
Second order reconstruction using a fully upwinded MUSCL scheme and van Leer's flux vector splitting [25] was utilized for all cases, along with the van Albada [26] limiter for stability.

4.8.1 Supersonic Vortex Flow

The supersonic vortex problem models a 90 degree turn of a supersonic flow between two inviscid slip walls, as shown in Figure 4.1a. For this study, the inner radius was set to $2.0\,m$, the outer radius to $3.0\,m$, the density at the inner radius to $1.0\,kg/m^3$, the pressure at the inner radius to $12780.0\,Pa$, and the Mach number at the inner radius to 2. For the lift and drag functionals of interest, the exact two-dimensional lift and drag forces on the inner radius are equal to $25560.0\,N/m$. Figures 4.1b and 4.1c show the adjoint variable for the x-momentum equation for the lift and drag functionals, respectively. While both functionals show high sensitivities near the wall, the drag functional adjoint shows high sensitivities along a longer extent than the lift functional due to the need to accurately convect information to the outflow region, where a large amount of the drag force is produced.

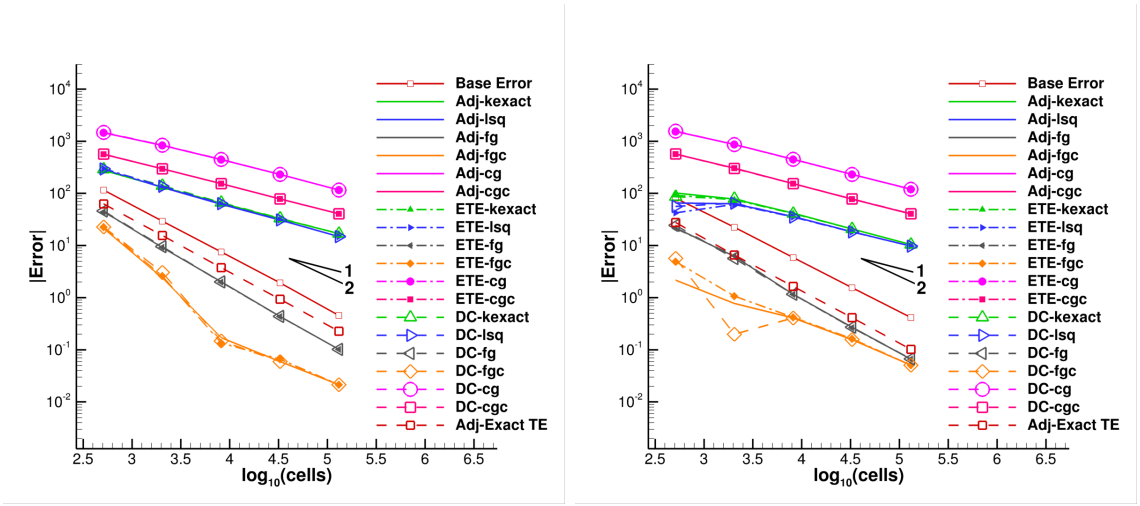
Figures 4.2a and 4.2b show the base DE in each functional and the remaining DE after each error estimation procedure is applied as a correction term. Due to the second order discretization scheme and the smoothness of the problem, the base DE converges at a second order rate. While the mechanism of incorporating the TE estimate differs, all three error estimation methods predict essentially the same functional DE estimate for the same TE estimate. While this result further reinforces the role that TE has as the source of DE, it also appears to indicate that there is some degree of freedom in choosing which error estimation procedure to implement.

The fine-grid method with correction ('fgc'), performs the best, but displays a 'knee' in the remaining error due to a sign change in the predicted remaining error. The 'exact' TE was used to drive the adjoint-based error estimation procedure and was found to perform slightly worse than the fine-grid method. While the exact TE would be expected to perform the best, the faceted boundaries result in an error in the TE, implying that higher order mesh representations are needed at the boundary. The fine-grid methods use a reconstructed boundary representation, which leads to better DE estimates. The coarse-grid methods and the continuous residual methods actually increase the error in the functionals. As will be discussed below, this is due to poor TE estimates near the inflow boundary that cause the DC and ETEs to predict excessive DE that is then transported throughout the domain. As noted by Giles and Pierce [27], low order errors in the TE estimate can interfere with the ability of the adjoint to properly predict the base DE, and the same argument should logically hold for DC and the ETEs. This can be viewed as an analogue to the order of accuracy issue of the primal problem; one low order error is all that is needed to globally reduce the primal scheme's observed order of accuracy. It should be noted that the 'kexact' and 'lsq' methods behave essentially the same due to the smoothness of the problem, which is not surprising given their similar algorithmic nature.



(a) (b) (c)

Figure 4.1: (a) Pressure distribution for supersonic vortex flow, (b) x-momentum adjoint variable for lift functional, and (c) x-momentum adjoint variable for drag functional.



(a) (b)

Figure 4.2: Base DE and remaining DE after correction with uniform refinement for (a) lift and (b) drag.

Comparisons of the DE predicted by DC and the ETEs to the exact DE are shown in Figures 4.3a- 4.3g. It is quite clear that the DE predicted by DC and the fully linearized ETEs are quite similar, but the ETEs with a lower order approximation of the discrete operator is much more dissipative. The DE predicted by the 'kexact' method shows little resemblance to the exact DE, while the 'fgc' method performs quite well. This is in full agreement with the results of Figures 4.2a and 4.2b. The poor performance of the 'kexact' method can be attributed to spikes in the predicted TE at the domain boundaries, as shown in Figure 4.4a. These spikes are an order of magnitude greater than the predicted TE through the rest of the domain, as shown by Figure 4.4b, reinforcing their spurious nature.

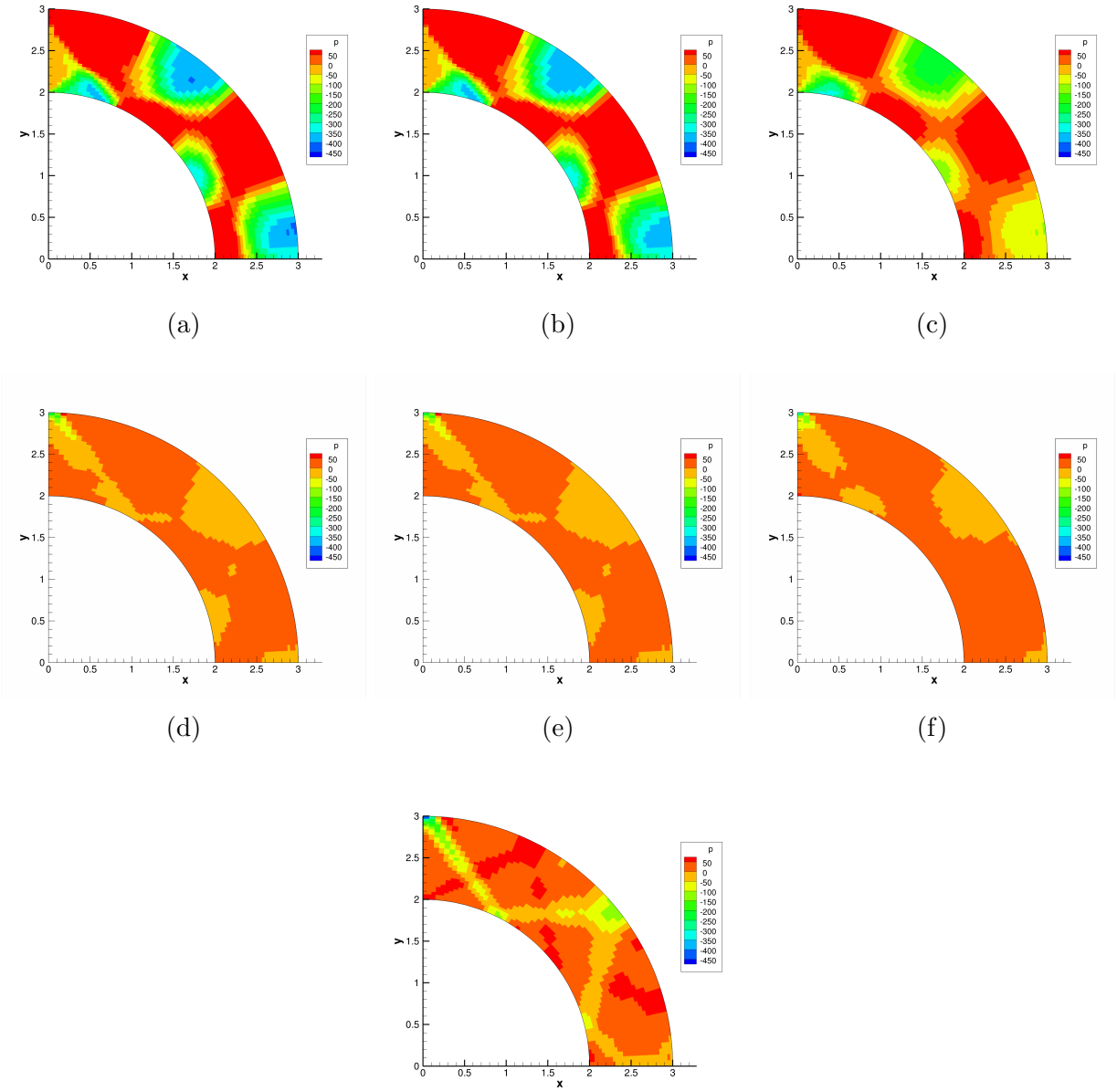
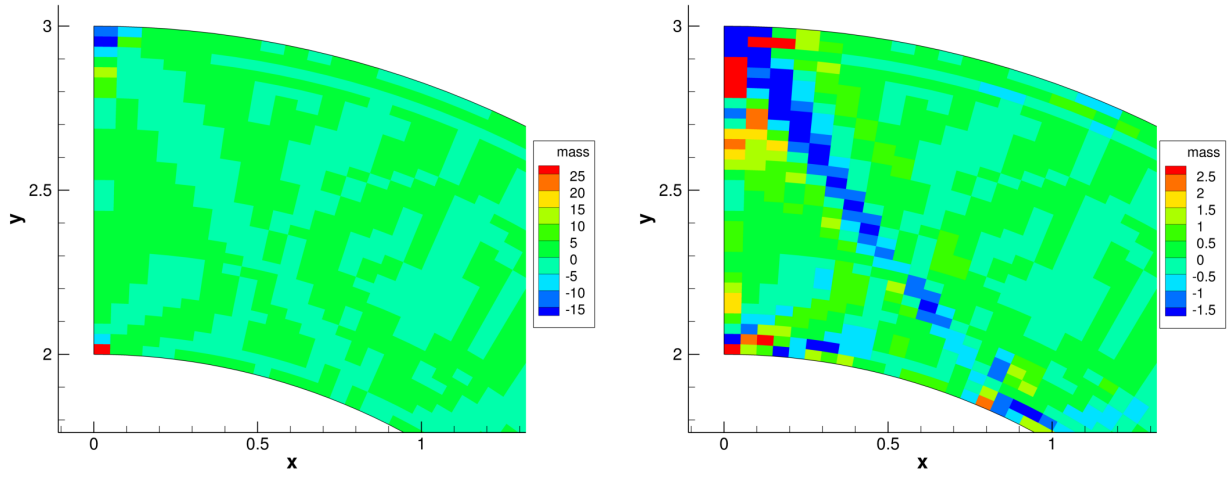


Figure 4.3: (a)-(c) Discretization error estimated using DC, fully linearized ETEs, and lower order ETEs driven by 'kexact' TE estimate, (d)-(f) discretization error estimated using DC, fully linearized ETEs, and lower order ETEs driven by 'fgc' TE estimate, (g) exact DE.



(a) (b)

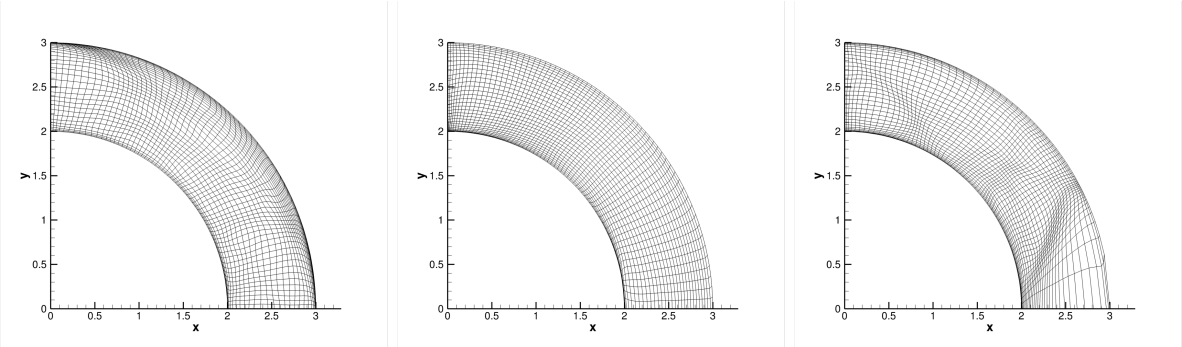
Figure 4.4: Estimated truncation error in continuity equation using the 'kexact' method (a) and with contours reduced by a factor of 10 (b), demonstrating the low order spike at the domain corners.

Adapted meshes are shown in Figures 4.5a- 4.5c. The mesh adapted for the drag functional demonstrates the most pronounced adaptation, pulling nodes away from the outflow region which does not contribute error to the functional. The lift mesh targeted more adaptation towards the lower wall, while the TE mesh targeted both the upper and lower walls. Despite the apparent lack of adaptation in the lift functional adapted mesh, it produced the only reduction in the base DE, while the need to fix crossovers of the mesh at the boundaries for the other adapted meshes resulted in poor grid quality which was detrimental to the DE.

Figures 4.6a- 4.7c compare each of the functional DE estimates on the adapted meshes, as well as to how the base DE on the adapted mesh compares to the base DE on the original, uniform mesh. Of note is that the TE estimates using the coarse-grid methods on the lift adapted mesh caused the DC method to diverge, and therefore these methods were removed from the plots. All the DE estimation methods agree quite well with each other for a given TE estimate, but the 'cg' TE methods massively overpredicted the functional DE. Out of all the methods, the 'fg' method performed fairly well on all the adapted meshes, although it does struggle on the lift functional adapted mesh when compared to the 'lsq' method.

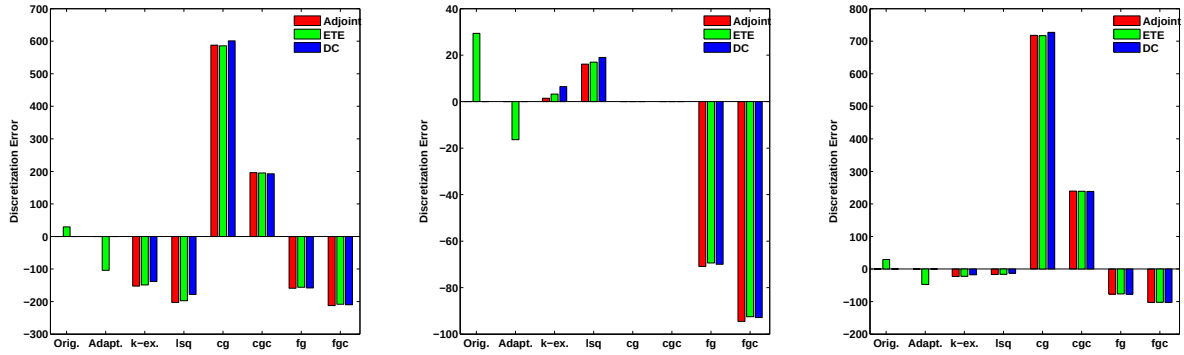
Figures 4.8a- 4.9c show comparisons of the adjoint DE estimates using four different degree polynomial reconstructions. Disregarding the behavior of the coarse-grid methods, the first degree polynomials almost always predicted the wrong sign of the DE, but the higher degree polynomials produced comparable results. In general, higher degree polynomials generally increased the predicted DE due to the increasingly poor grid metrics and increased stencil

sizes involved in the higher degree reconstructions. The 'lsq' method is an exception to this trend, and the fine-grid methods on the TE adapted mesh did perform remarkably well.



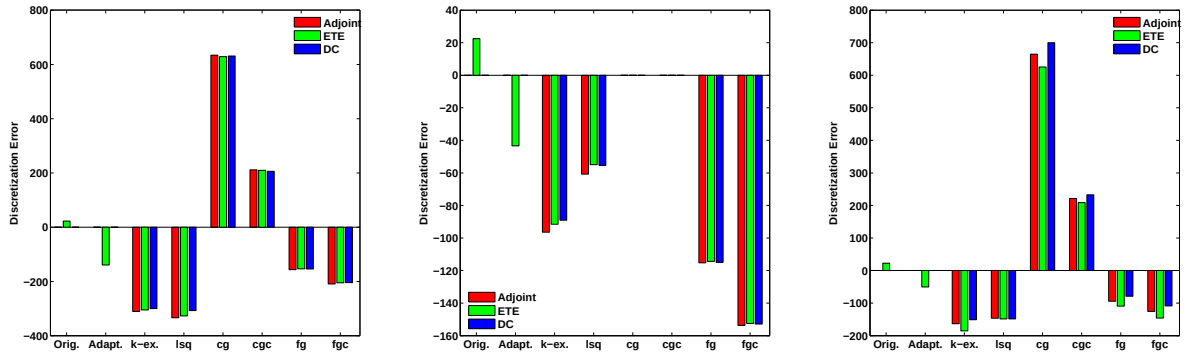
(a) (b) (c)

Figure 4.5: Vortex flow meshes adapted on (a) truncation error, (b) lift functional, and (c) drag functional



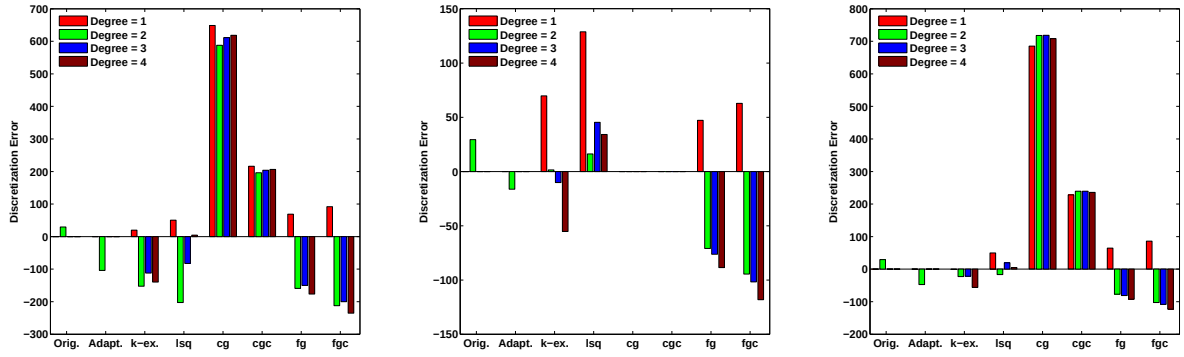
(a) (b) (c)

Figure 4.6: Lift functional DE estimates for meshes adapted on (a) TE, (b) lift functional, and (c) drag functional with true DE on the original and adapted meshes for comparison. Note the different scales.



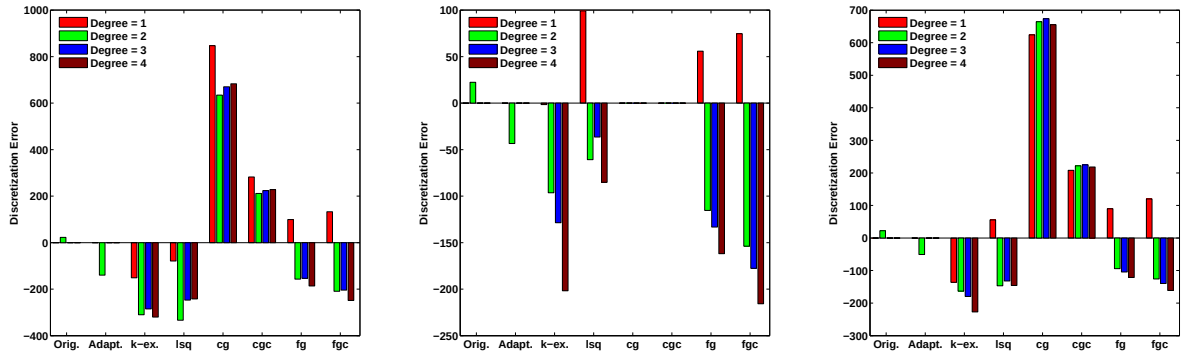
(a) (b) (c)

Figure 4.7: Drag functional DE estimates for meshes adapted on (a) TE, (b) lift functional, and (c) drag functional with true DE on the original and adapted meshes for comparison. Note the different scales.



(a) (b) (c)

Figure 4.8: Effect of reconstruction order on adjoint error estimates for lift on (a) TE, (b) lift functional, and (c) drag functional adapted meshes.



(a) (b) (c)

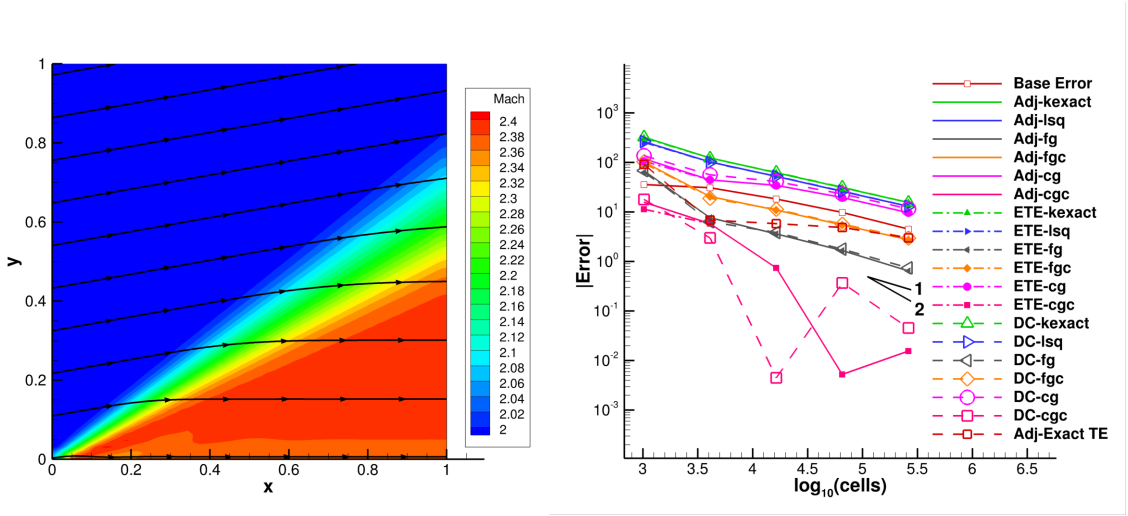
Figure 4.9: Effect of reconstruction order on adjoint error estimates for drag on (a) TE, (b) lift functional, and (c) drag functional adapted meshes.

4.8.2 Expansion Fan

A Mach 2 flow with inflow static pressure of $100\,kPa$ and static temperature of $300\,K$ was allowed to expand around a 10 degree corner in a square domain, as shown in Figure 4.10a. A slip wall is modeled at the bottom of the domain, supersonic inflow conditions were applied at the left boundary, and the remaining two sides were treated as supersonic outflow.

Five uniformly refined Cartesian grids, ranging in size from 32x32 to 512x512 cells are studied. The functional of interest for this case is the absolute value of the normal force on the wall, which has an exact value of $54796.6300409\,N/m$. Due to the use of a slightly dispersive limiter, the functional converges at a rate of nearly 1 rather than the expected rate of 2/3 due to the presence of the linear discontinuity at the beginning and end of the expansion fan [28]. Previous work [29] had shown a convergence rate of 2/3, but had utilized a different limiter. Once again, the 'fg' method performs very well, but the 'fgc' performs slightly worse, as shown in Figure 4.10b. The adjoint-based error estimation procedure using the 'exact' TE results in DE estimates that perform in between these two methods, which can be attributed to how the root of the expansion fan is contained by a single cell, and the flux quadrature used to estimate the TE may not be capable of accurately capturing the linear discontinuity at the beginning and end of the expansion fan. The coarse-grid methods perform the best, as they do not overestimate the DE, but show differences between the DC method as compared to the ETEs and adjoint method. This can be attributed to the nonlinear nature of the DC problem, resulting in slightly different DE estimates downstream of the expansion fan root, as shown in Figures 4.11a- 4.11f. The exact DE, Figure 4.11g, is actually much higher than any of the predictions indicate. For the 'kexact' approach, Figures 4.11a- 4.11c, the predicted DE is dominated by a feature along the termination of the expansion. The 'fgc' method (Figures 4.11d- 4.11f) shows a similar feature, although over

a smaller domain, whereas the exact DE is dominated by the leading edge of the expansion fan. Each estimation procedure results in an alternating DE distribution near the wall, and just downstream of the root of the expansion fan, which contributes the most to the errors in the functional DE estimates.



(a) (b)

Figure 4.10: Mach number field of supersonic expansion fan on uniform mesh (a) and base DE and remaining DE after correction for force functional with uniform refinement (b)

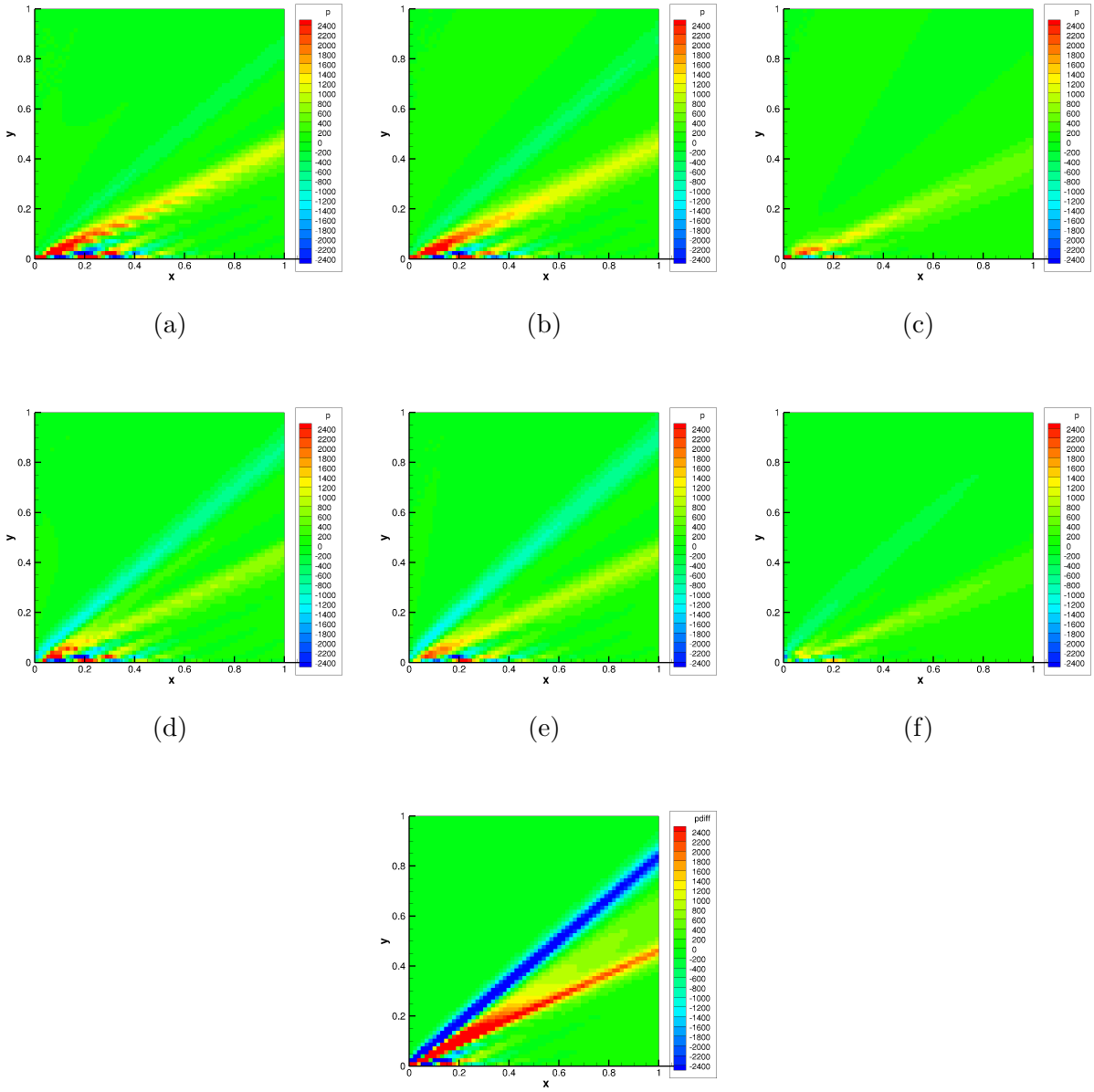
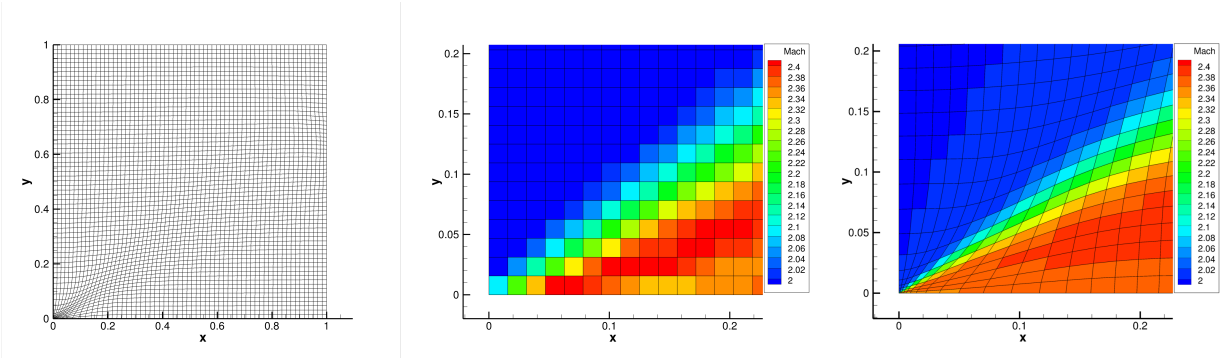


Figure 4.11: (a)-(c) Discretization error estimated using DC, fully linearized ETEs, and lower order ETEs driven by 'kexact' TE estimate, (d)-(f) discretization error estimated using DC, fully linearized ETEs, and lower order ETEs driven by 'fgc' TE estimate, (g) exact DE.

As previously noted, adaptation is performed on the 64x64 cell mesh. Each of the adaptation methods produced fairly similar meshes, although the TE adapted mesh sought to adapt near where the expansion fan hit the outflow boundary, while the adjoint based adaptation

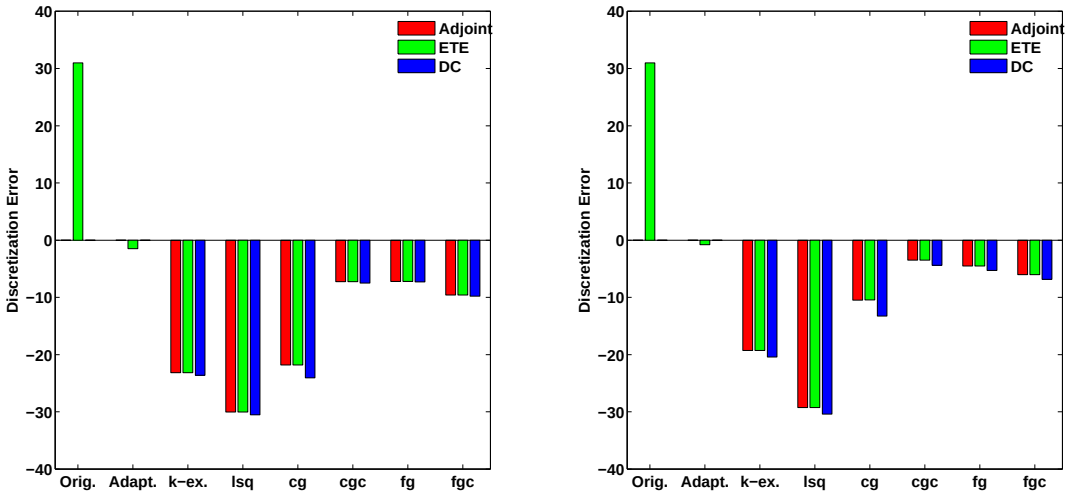
ignored this region as it is unimportant to the force functional. Due to the smoothness of the continuous expansion, the TE estimation procedure barely registers TE in the expansion, and the high TE estimates near the root drive the adaptation methods. Figure 4.12a shows the typical refinement pattern near the base of the expansion fan for the TE adapted mesh while Figures 4.12b and 4.12c show a zoomed in view of the expansion fan root in order to show the increase in sharpness of the Mach contours due to the adaptation.



(a) (b) (c)

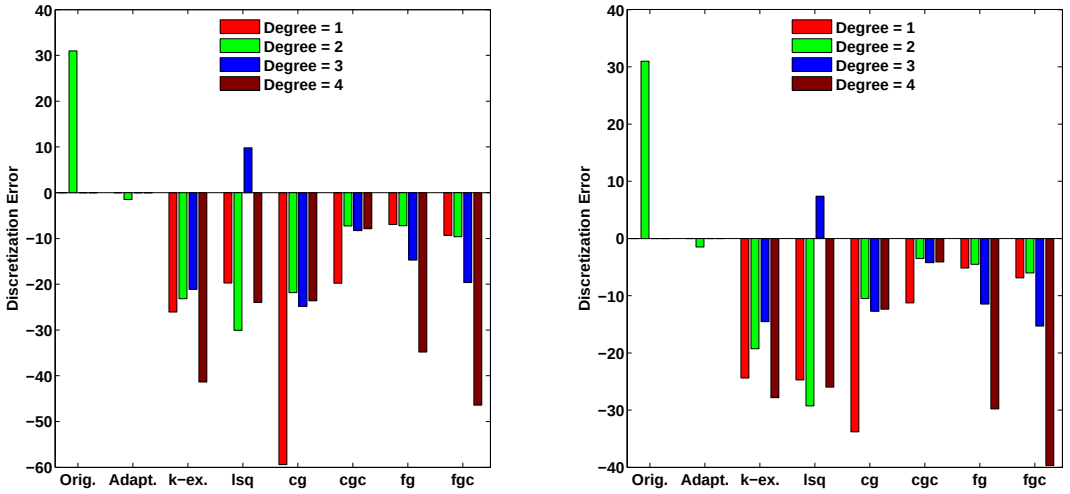
Figure 4.12: (a) Truncation error adapted mesh, (b) Mach contours on original mesh and (c) on adapted mesh. Note the domain extents.

Regardless of the adaptation method used, both meshes produced an order of magnitude reduction in the base DE as compared to the non-adapted meshes, as shown in Figures 4.13a and 4.13b. However, all the error estimation methods struggled to predict the DE, most likely due to the skewed mesh and resulting mesh metrics, although all the estimates have the correct sign. The 'cgc' and 'fg' TE estimates result in remarkably similar DE estimates from all methods for the 2^{nd} degree polynomials being used. Figures 4.14a and 4.14b show the behavior of the adjoint DE estimates for polynomial reconstruction of degree 1 through 4. The 'fine-grid' methods show a doubling in the error estimates for polynomials of degrees 2 through 4, indicating an increasingly large 1^{st} order error in the TE estimates. The 'coarse-grid' methods behave similarly for polynomial degrees 2 through 4, most likely due to the smoothing effect of the coarse grid restriction operation. The continuous residual methods do not demonstrate a consistent change due to reconstruction order, although the 'lsq' method changes sign for the 3^{rd} degree polynomial due to alternating sign changes in the estimated TE near the root of the expansion fan.



(a) (b)

Figure 4.13: Force functional DE estimates for meshes adapted on (a) TE and (b) force functional with true DE on the original and adapted meshes for comparison.



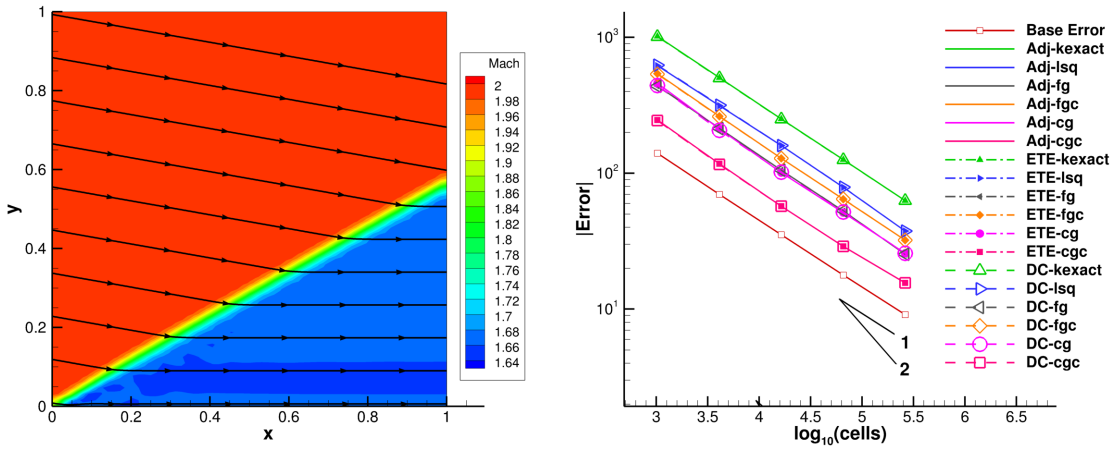
(a) (b)

Figure 4.14: Effect of reconstruction order on adjoint error estimates for force functional on (a) TE and (b) force functional adapted meshes.

4.8.3 Shocked Flow

The shocked flow case uses the exact same inflow properties as the expansion fan case, but is instead compressed by a 10 degree wedge angle, as shown in Figure 4.15a. Ideally, the shock represents a discontinuity between two separate regions of uniform flow. A slip wall is modeled at the bottom of the domain while, supersonic outflow conditions were applied at the right side of the domain, and the remaining two sides were treated as supersonic inflow.

The functional is exactly the same as the expansion fan case, with an exact force per unit depth of 168065.1836 N/m. Due to the presence of the shock, the functional converges at an expected rate of 1 [28], as shown in Figure 4.15b. While none of the TE estimation methods proved to be effective, as all increased the functional DE, there are several interesting results. The continuous residual 'lsq' method was the only TE estimation scheme that resulted in the correct sign of the DE in the functional, and it performed better than the 'kexact' method near the shock due to inherently smoother curve fits generated by the least-squares procedure. For this case, the 'fg' and 'cgc' methods produced almost identical DE estimates. Downstream of the root of the shock, the DE predicted by DC and the ETEs demonstrate oscillations that contribute the most to the error in the functional estimates, as shown in Figures 4.16a- 4.16f. Massive over/undershoots near the shock dominate the TE estimates, as shown in Figures 4.17a and 4.17b for the 'kexact' method, while the rest of domain is essentially uniform flow with low TE. An adaptive reconstruction scheme driven by a shock sensor could potentially reduce the order of the reconstruction near the shock in order to minimize overshoots, although this may negatively impact the behavior of the continuous residual flux balance. Qualitatively, the DE estimates compare fairly well with the exact DE, Figure 4.16g, and once again, the lower order ETEs are more dissipative. It should be noted that Sensei lacks the ability to properly integrate the solution on either side of the shock, see [30] for a discussion, and so the exact DE is most likely subject to some error and the 'exact' TE is not used to drive the DE estimation procedures for this case.



(a) (b)

Figure 4.15: Mach number field of shocked flow on uniform mesh (a) and base DE and remaining DE after correction for force functional with uniform refinement (b)

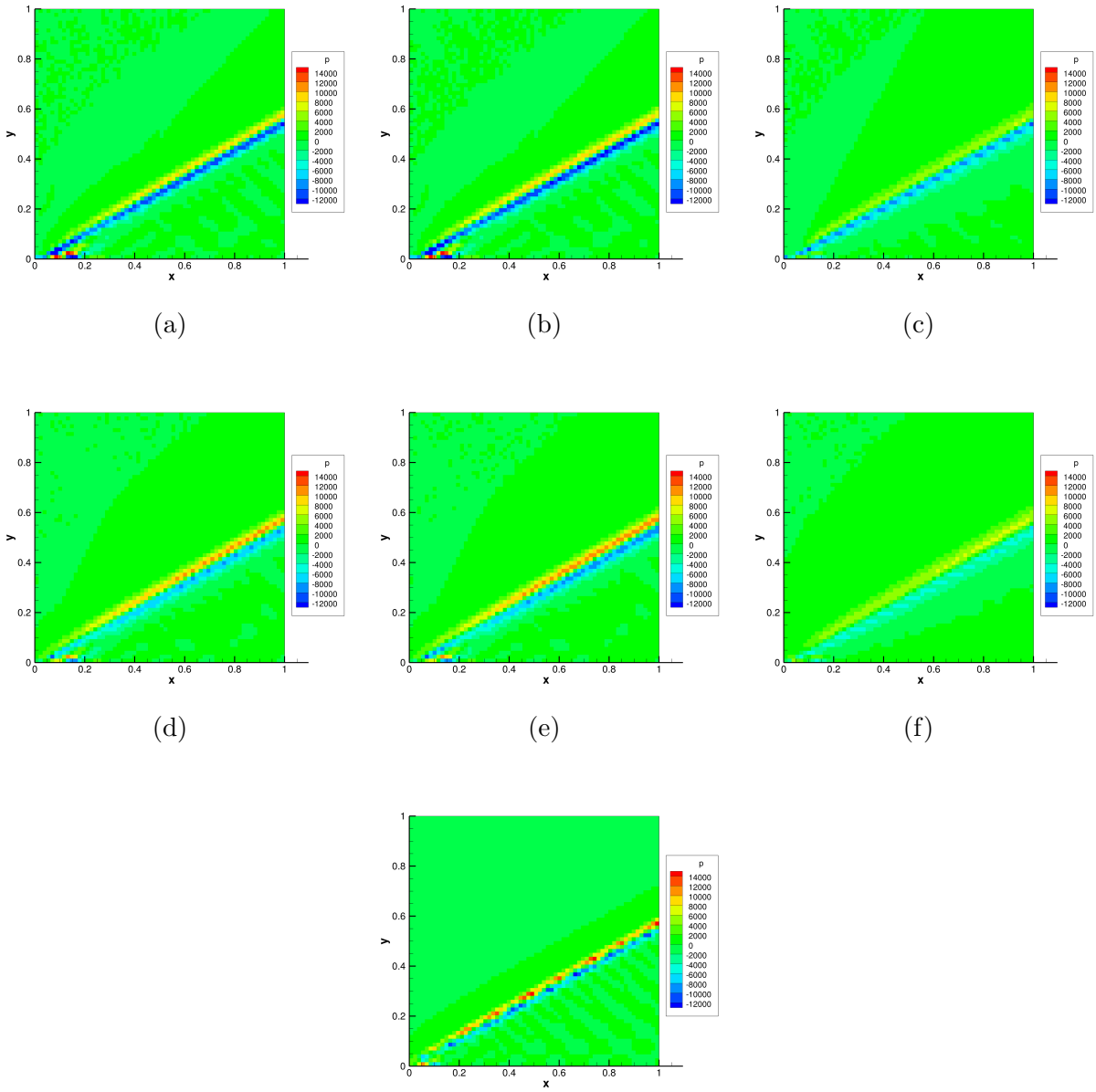
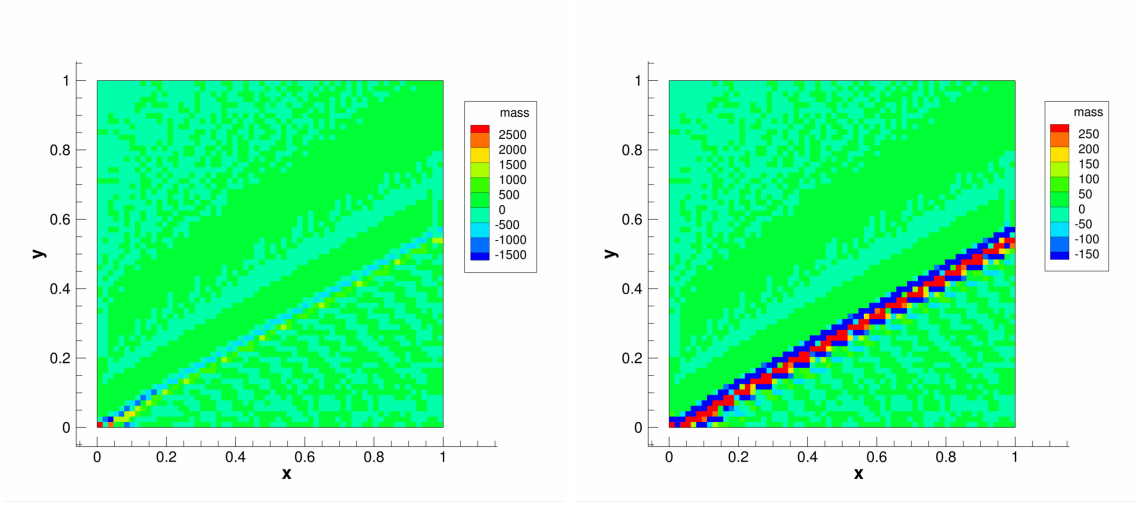


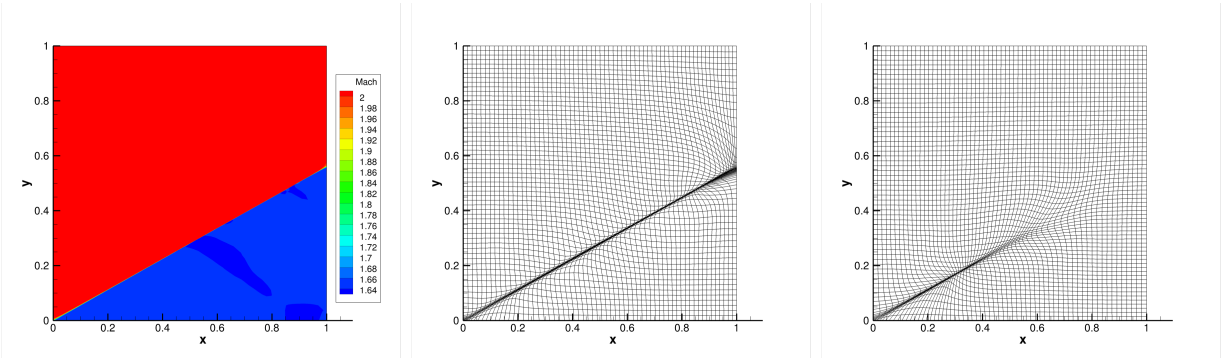
Figure 4.16: (a)-(c) Discretization error estimated using DC, fully linearized ETEs, and lower order ETEs driven by 'kexact' TE estimate, (d)-(f) discretization error estimated using DC, fully linearized ETEs, and lower order ETEs driven by 'fgc' TE estimate, (g) exact DE.



(a) (b)

Figure 4.17: Estimated truncation error in continuity equation in continuity equation using the 'kexact' method (a) and with contours reduced by a factor of 10 (b).

The adaptation process clearly targeted the shock, as shown in Figures 4.18b and 4.18c. It is interesting to note the 'cut-off' in adaptation that Figure 4.18c shows. Due to the supersonic nature of the flow field, the TE located along the shock past an x-coordinate value of approximately 0.6 has no effect on the force functional. The adapted meshes skew the cells across the shock, most likely to keep one cell in the shock region. It also appears that the adjoint adapted mesh has attempted to pull more points towards the shock given the mesh density near the region of x = 0.2, y = 0.1.



(a) (b) (c)

Figure 4.18: Mach contour on TE adapted mesh (a) and meshes adapted on TE (b) and the force functional (c).

The adaptation does succeed in reducing the base DE in the functionals, however the error

estimates do not improve, as shown in Figures 4.19a and 4.19b. The DC method using the 'lsq' TE estimate is the only method that captures the correct sign of the DE on the adjoint adapted mesh, although this may be due to a bias in the DC procedure that appears to offset the DC error estimates from the ETEs and adjoint error estimates. Figures 4.20a and 4.20b compare the degrees of the polynomial reconstructions for the adjoint error estimates. Due to the behavior of the solution as essentially two regions of uniform flow separated by a singularity, the 1^{st} degree reconstructions appear to perform the best compared to the other reconstruction orders, with the 'cgc' method performing the best.

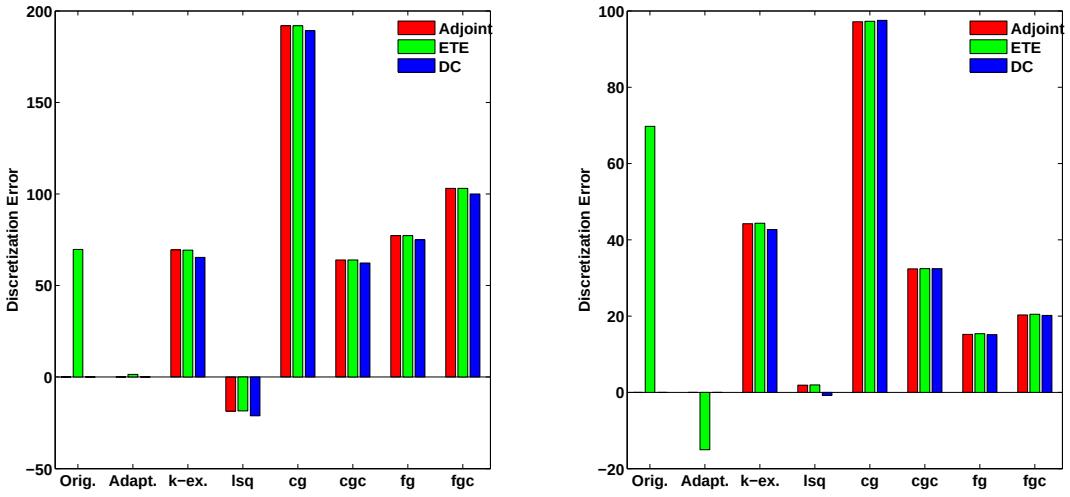
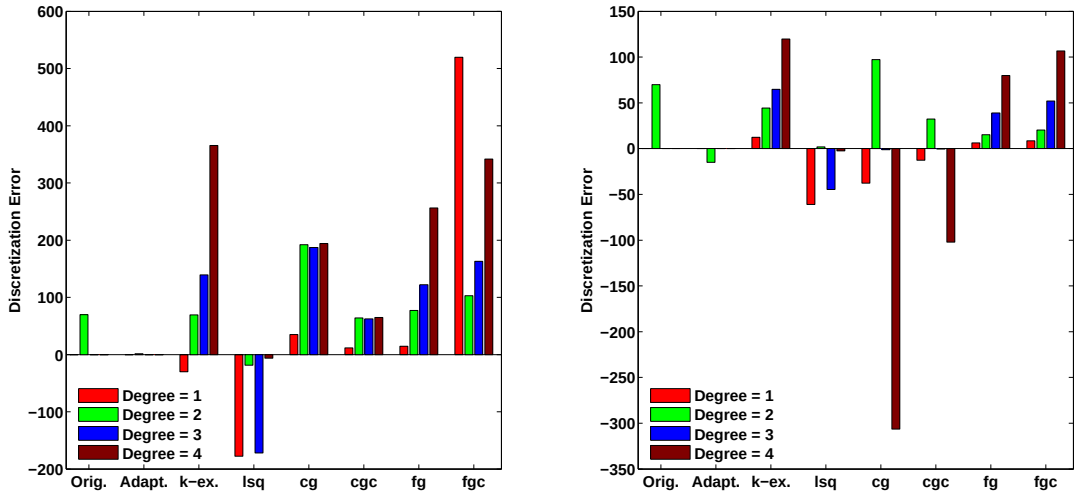


Figure 4.19: Force functional DE estimates for meshes adapted on (a) TE and (b) force functional with true DE on the original and adapted meshes for comparison.



(a) (b)

Figure 4.20: Effect of reconstruction order on adjoint error estimates for force functional on (a) TE and (b) force functional adapted meshes.

4.9 Conclusions

Several truncation error estimation procedures have been applied to the defect correction method, error transport equations, and the adjoint error estimation procedure in order to estimate functional discretization error. As compared to previous work [11], there is no clearly superior TE estimation procedure. This can most likely be ascribed to the lack of smooth grid transformations after mesh adaptation, which are necessary to perform the reconstructions in a computational space, and therefore the polynomial reconstructions should be limited to a degree of 2 in order to minimize the stencil size and impact of the poor mesh metrics. One possible solution is to use an adaptation process that ensures some level of smoothness and orthogonality in the meshes, such as the variational approach of Brackbill and Saltzman [31] as recently implemented by Tyson [6]. Despite these problems, the discretization error in the functionals of interest were reduced for the expansion fan and shocked flow test cases.

Regardless of the error estimation scheme being used, the functional discretization error estimate is essentially the same for the same truncation error estimate. From a pragmatic standpoint, if improved functional outputs are the only concern of a code user, then a defect correction or error transport equation corrected solution, or an adjoint correction for a single functional, could be used interchangeably. This lends a degree of flexibility to the implementer. In the absence of an implicit solver capability, or a fully linearized LHS, the defect correction procedure could be implemented with minimal effort. If a full linearization

for the LHS could be developed, either by hand differentiation, automatic differentiation, or complex variable perturbations, either the error transport or adjoint equations could be implemented. Even if an adjoint solver has already been developed, Pierce and Giles [32] demonstrated that a functional output can be further improved by first correcting the primal solution using the defect correction method before calculating the dual solution. This finding could logically be extended to the error transport equations, which as noted above, could be easily implemented in a code which already has an adjoint solver.

If either defect correction or error transport equations are implemented, an entropy adjoint approach [5] could be easily implemented to provide an adaptation indicator. Using the original primal solution and the defect or error transport corrected primal solution to provide the low order and high order entropy adjoint, an indicator like that used in the work of Nemec and Aftosmis [23] could be formed.

This work further reinforces the importance of the truncation error as the driver of discretization error and the need for improved truncation error estimation procedures in the future. Previous work [10] based on manufactured solutions have shown much promise in being able to accurately predict truncation error, but these cases were able to treat the domain boundaries and their ghost cells as extensions of the interior. For this study, the domain boundaries cannot be as naturally extended, resulting in one sided reconstructions that reduce the smoothness of the truncation error estimates. The smoothness of the truncation error is naturally dependent on having appropriate boundary conditions which, in the case of a discrete adjoint correction procedure, needs to be consistent with the adjoint problem [33]. A possible solution to the errors introduced by one sided reconstructions is to add additional constraints based on the domain boundary values when a reconstruction domain intersects the boundary. This may also require the use of higher order geometry reconstructions at walls, rather than the faceted walls that most finite volume methods typically implement, in order to appropriately capture the continuous truncation error behavior.

4.10 Bibliography

[1] Roy, C. J., "Strategies for Driving Mesh Adaptation in CFD," Invited Paper for Session on Error Estimation and Control, AIAA Paper 2009-1302, 47th AIAA Aerospace Sciences Meeting, Orlando, Florida, January 5-8, 2009.

[2] Warren, G. P., Anderson, W. K., Thomas, J. L., and Krist, S. L., "Grid Convergence for Adaptive Methods," AIAA Paper 1991-1592, AIAA 10th Computational Fluid Dynamics Conference, 1991.

[3] Venditti, D. A., *Grid Adaptation for Functional Outputs of Compressible Flow Simulations*, Ph.D. thesis, Massachusetts Institute of Technology, June 2002.

[4] Park, M. A., Lee-Rausch, E. M., and Rumsey, C. L., "FUN3D and CFL3D Computations for the First High Lift Prediction Workshop," AIAA Paper 2011-936, 49th AIAA Aerospace Sciences Meeting, Orlando, Florida, January 4-7, 2011.

[5] Fidkowski, K. J. and Roe, P. L., "Entropy-based Mesh Refinement, I: The Entropy Adjoint Approach," AIAA Paper 2009-3790, 19th AIAA Computational Fluid Dynamics Conference, San Antonio, Texas, June 22-25, 2009.

[6] Tyson, W. C., Derlaga, J. M., Roy, C. J., and Choudhary, A., "Comparison of r-Adaptation Techniques for 2-D CFD Applications," AIAA Paper 2015-2611, 22nd AIAA Computational Fluid Dynamics Conference, Dallas, Texas June 22-26, 2013.

[7] Peraire, J., Vahdati, M., Morgan, K., and Zienkiewicz, O., "Adaptive remeshing for compressible flow computations," *Journal of Computational Physics*, Vol. 72, No. 2, 1987, pp. 449–466.

[8] Habashi, Wagdi G., D. J., Vallet, M.-G., Bourgault, Y., and Fortin, M., "Anisotropic mesh adaptation: towards user-independent, mesh-independent and solver-independent CFD. Part III. Unstructured meshes," *International Journal for Numerical Methods in Fluids*, Vol. 39, No. 8, Jul 2002, pp. 675–702.

[9] Phillips, T. S. and Roy, C. J., "Residual Methods for Discretization Error Estimation," AIAA Paper 2011-3870, 20th AIAA Computational Fluid Dynamics Conference, Honolulu, Hawaii, June 27-30, 2011.

[10] Phillips, T. S., Derlaga, J. M., Roy, C. J., and Borggaard, J., "Finite Volume Solution Reconstruction Methods for Truncation Error Estimation," AIAA Paper 2013-3090, AIAA Fluid Dynamics and Co-located Conferences and Exhibit, San Diego, California, June 24-27, 2013.

[11] Phillips, T. S., *Residual-based Discretization Error Estimation for Computational Fluid Dynamics*, Ph.D. thesis, Virginia Tech, October 2014.

[12] Barth, T. J. and Frederickson, P. O., "Higher Order Solution of the Euler Equations on Unstructured Grids Using Quadratic Reconstruction," AIAA Paper 1990-0013, 28th AIAA Aerospace Sciences Meeting, Reno, Nevada, January 8-11, 1990.

[13] Roache, P. J., *Fundamentals of Verification and Validation*, Hermosa Publishers, 2009.

[14] Oberkampf, W. and Roy, C. J., *Verification and Validation in Scientific Computing*, Cambridge University Press, 2010.

[15] Venditti, D. A. and Darmofal, D. L., "Grid Adaptation for Functional Outputs: Application to Two-Dimensional Inviscid Flows," *Journal of Computational Physics*, Vol. 176, No. 1, 2002, pp. 40 – 69.

[16] Derlaga, J. M., Phillips, T. S., and Roy, C. J., "SENSEI Computational Fluid Dynamics Code: A Case Study in Modern Fortran Software Development," AIAA Paper 2013-2450, AIAA Fluid Dynamics and Co-located Conferences and Exhibit, San Diego, California, June 24-27, 2013.

[17] Jameson, A., Schmidt, W., and Turkel, E., "Numerical Simulation of the Euler Equations by Finite Volume Methods using Runge-Kutta Time Stepping Schemes," AIAA Paper 1981-1259, AIAA 5th Computational Fluid Dynamics Conference, 1981.

[18] Beam, R. M. and Warming, R., "An implicit finite-difference algorithm for hyperbolic systems in conservation-law form," *Journal of Computational Physics*, Vol. 22, No. 1, 1976, pp. 87 – 110.

[19] van Leer, B., "Towards the Ultimate Conservative Difference Scheme. V. A Second-order Sequel to Godunov's Method," *Journal of Computational Physics*, Vol. 32, No. 1, 1979, pp. 101 – 136.

[20] Hirsch, C., *Numerical Computation of Internal and External Flows*, Vol. 2: Computational Methods for Inviscid and Viscous Flows, John Wiley & Sons, Ltd, 1990.

[21] Nielsen, E. J., Lu, J., Park, M. A., and Darmofal, D. L., "An implicit, exact dual adjoint solution method for turbulent flows on unstructured grids," *Computers & Fluids*, Vol. 33, No. 9, 2004, pp. 1131 – 1155.

[22] Choudhary, A. and Roy, C. J., "Structured Mesh r-Refinement using Truncation Error Equidistribution for 1D and 2D Euler Problems," AIAA Paper 2013-2444, AIAA Fluid Dynamics and Co-located Conferences and Exhibit, San Diego, California, June 24-27, 2013.

[23] Nemec, M. and Aftosmis, M. J., "Adjoint Error Estimation and Adaptive Refinement for Embedded-Boundary Cartesian Meshes," AIAA Paper 2007-4187, 18th AIAA Computational Fluid Dynamics Conference, Miami, Florida, June 25-28, 2007.

[24] Ollivier-Gooch, C., Nejat, A., and Michalak, K., "Obtaining and Verifying High-Order Unstructured Finite Volume Solutions to the Euler Equations," *AIAA Journal*, Vol. 47, No. 9, 2009, pp. 2105–2120.

[25] van Leer, B., "Flux Vector Splitting for the Euler Equations," *Proc. 8th International Conference on Numerical Methods in Fluid Dynamics*, Springer Verlag, 1982.

[26] van Albada, G., van Leer, B., and Roberts, W. J., "A comparative study of computational methods in cosmic gas dynamics," *Astronomy and Astrophysics*, Vol. 108, No. 1, 1982, pp. 76–84.

[27] Giles, M. and Pierce, N., "Superconvergent lift estimates through adjoint error analysis," *Innovative Methods for Numerical Solutions of Partial Differential Equations*, World Scientific Publishing Co. Pte. Ltd., 2002, pp. 198–211.

[28] Banks, J., Aslam, T., and Rider, W., "On sub-linear convergence for linearly degenerate waves in capturing schemes," *Journal of Computational Physics*, Vol. 227, No. 14, 2008, pp. 6985 – 7002.

[29] Derlaga, J. M., Phillips, T. S., Roy, C. J., and Borggaard, J., "Adjoint and Truncation Error Based Adaptation for Finite Volume Schemes with Error Estimates," AIAA Paper 2015-1262, AIAA SciTech, Kissimmee, Florida, January 5-9, 2015.

[30] Grier, B., Alyanak, E., White, M., Camberos, J., and Figliola, R., "Numerical integration techniques for discontinuous manufactured solutions," *Journal of Computational Physics*, Vol. 278, No. 0, 2014, pp. 193 – 203.

[31] Brackbill, J. and Saltzman, J., "Adaptive Zoning for Singular Problems in Two Dimensions," *Journal of Computational Physics*, Vol. 46, 1982, pp. 342 – 368.

[32] Pierce, N. A. and Giles, M. B., "Adjoint and defect error bounding and correction for functional estimates," *Journal of Computational Physics*, Vol. 200, No. 2, 2004, pp. 769 – 794.

[33] Lu, J., *An a posteriori error control framework for adaptive precision optimization using discontinuous Galerkin finite element method*, Ph.D. thesis, Massachusetts Institute of Technology, 2005.

Chapter 5

Discussion and Conclusions

Out of all numerical errors, discretization error is the largest, and the ability to estimate it is of extreme importance due to the continued growth and reliance on CFD. In this work, comparisons of functional discretization error estimates using adjoint methods, error transport equations, and the defect correction method have been made for a variety of truncation error estimation procedures; comparisons which have previously never been performed in the field. The discretization error estimates are dependent on accurate estimates of the truncation error, which is itself the local source of the discretization error. For both the quasi-1D and 2D Euler equations, functional discretization error was approximated through the use of multiple truncation error estimation methods in order to determine the most effective error estimation procedure.

For the quasi-1D Euler equations, a new method of truncation error estimation based on the reconstruction of the discrete solution and operating the weak form of the governing equations upon it was examined. Various methods of reconstruction of the primitive, conserved, and flux variables were compared and it was found that the use of a k-exact reconstruction of the primitive variables was the most effective for the isentropic test case as it properly conserved the solution in a finite volume consistent manner and did not result in physically inconsistent conditions. When mesh adaptation was performed, the functional discretization error and the adjoint error estimates were improved and adaptation based on truncation error alone resulted in a mesh that performed well with the adjoint error estimation procedure, indicating that adjoint based error adaptation may not always be necessary.

For the shocked case, difficulties in estimating the truncation error near the shock decreased the effectiveness of the adjoint error estimates when using the new weak formulation truncation error estimate. The exact truncation error did not perform well and this indicates that a better quadrature method, like that found in [1], may be needed. However, a truncation error estimate using an embedded grid was capable of providing an accurate adjoint error estimate as the limiting procedure of the discrete scheme was able to reduce truncation error spikes. Mesh adaptation was not able to improve the functional discretization error, as the

shock was still the primary source of error, but the adjoint error estimates did improve for the pressure integral due to better mesh resolution near the shock.

Comparisons of the defect correction method and error transport equations showed that they performed in a similar manner to the adjoint method when using estimated truncation error. When using the exact truncation error, the defect correction method was able to exactly calculate the functional error, while the error transport equations and adjoint method approximated the discretization error in an asymptotic manner. This is due to the fact that the defect correction method solves a nonlinear problem, while the other methods are linearized.

For the 2D Euler equations, a series of test cases reinforced that the defect correction, error transport equations, and the adjoint method all produce similar functional discretization error estimates for the same truncation error estimate, even with mesh adaptation. This means that code developers have some latitude in choosing which method to implement. The defect correction method is the easiest to implement, but it is also the most costly to solve due to its nonlinear nature. The error transport equations and adjoint method have similar solution costs, but are more costly to implement than defect correction due to the need to linearize the governing equations. If an implicit solver is in place, some of the necessary linearization work may already be complete. It is important to note that if mesh adaptation is desired, it may be necessary to implement an adjoint solver, although the entropy adjoint [2] may provide an alternative process.

While it was expected that the exact truncation error would result in perfect functional error estimates for the defect correction method, it was found that due to the faceted geometry representation that the error estimates were slightly worse than the 'fine-grid-corrected' method which uses a better representation of the curved geometry in its formulation. This implies that higher order representations of grid lines on curved boundaries may be needed in order to truly calculate the exact TE. Overall, the 'fine-grid' and 'fine-grid-corrected' methods appeared to perform the best for the isentropic problems. While none of the truncation error estimation techniques resulted in accurate functional discretization error estimates for the shocked case, the 'coarse-grid' and 'coarse-grid-corrected' methods performed the best as they were able to smooth out the large truncation error spikes caused by the shock.

For both the expansion fan and shock cases, mesh adaptation decreased the amount of functional discretization error, often by a factor of 10. However, the error estimation procedures suffered due to the non-smooth grid metrics encountered by the solution reconstruction procedures. Recent work on mesh adaptation [3] may provide smoother grid transformations and improve the quality of the truncation error estimates on the adapted meshes. For the supersonic vortex flow, mesh adaptation near the curved boundaries often resulted in mesh crossover, and the resulting untangling of the mesh would result in very skewed and/or stretched cells and resulted in an increase in the discretization error in the functionals. It should be noted that accurate geometry representation and adaptation on boundaries is a challenging problem for any mesh adaptation algorithm and more research [3] has been un-

dertaken in order to improve adaptation on curved surfaces by increasing the fidelity of the geometry representation. As with the non-isentropic quasi-1D test case, mesh adaptation focused on the shock feature. However, the adjoint-based adaptation only focused on the portion of the shock that affected the functional of interest whereas the purely truncation error based method focused on the entire feature across the domain. For problems where shock waves exist that do not have any influence on any functionals of interest, purely truncation error based adaptation may prove to be overly costly, as it will seek to better refine the entire shock wave.

All of the previously described tests were made possible by creating new CFD codes which were designed with the benefit of software engineering best practices. Often overlooked by engineers, the importance of design and proper coding discipline is of vital importance in creating a large software project that is to be shared amongst researchers. With this knowledge, a common framework was created in Sensei which made it possible to leverage the work of other researchers [4, 5] and increase collaboration, resulting in perhaps the only code in existence which is capable of solving the adjoint equations, defect correction, and error transport equations. The Sensei code incorporates many new programming paradigms made possible through the use of modern Fortran, enabling greater code flexibility and allow for future modifications. As modern Fortran incorporates polymorphic objects and type bound procedures, this has enabled the development of a new iterative solver library and method of building the discrete linearization matrix which will allow Sensei to seamlessly transition between data types meant for a CPU or GPU. As a result, a large barrier has been removed from the path of preparing code for execution on GPUs, hardware which has shown massive potential for decreasing the time-to-solution for CFD computations. Additionally, by developing a new method of manufactured solution technique, it has become easier to perform code verification studies, a necessary step for demonstrating that a code has properly implemented a given algorithm.

5.1 Future Work

Improvements in truncation error estimation methods are still needed, and choosing the proper reconstruction stencil, or reduction in order of the reconstruction when discontinuities are present, is a pressing issue. Flow feature recognition, based on feature-based adaptation indicators, may be able to automatically detect and adapt the reconstruction stencil to discontinuities. The unexpected spike in the quasi-1D Euler study highlights the issue that the truncation error is not well understood and needs further study if it is to be properly estimated. Better methods of reconstruction are needed near boundaries, and the truncation error methods used in this work may benefit from the use of Dirichlet constraints when they are available or a hybrid scheme of multiple truncation error estimation techniques patched together.

Further research is needed into the nature of appropriate boundary conditions for finite vol-

ume solvers which maintain consistent truncation error properties at the boundaries but do not degrade the discrete adjoint or the error transport equations, both methods which are dependent on the linearization of the boundary conditions. Previous work [6] had utilized higher order extrapolations to ghost cells in order to provide smooth truncation error estimates. However, these types of extrapolations can violate the characteristic nature of the governing equations and are therefore inconsistent with adjoint methods.

For both the primal and dual solvers, implementation of Jacobians obtained via complex number manipulation [7] would allow for easier implementation of new boundary conditions and avoid the need to debug hand differentiated Jacobians whenever new flux schemes or functionals are added. The reduction in maintenance burden and source code would simplify the Sensei code base and remove the need for an expert-in-the-loop to maintain the dual solver.

More recent work [3] has focused on implementing an adaptation package within Sensei rather than as a separate library interfacing with it like SAM [8], which would greatly decrease the total adaptation process time. The new adaptation techniques include methods which may result in smoother grid metrics and should improve the truncation error estimates on adapted meshes. Another possible solution is to perform the reconstructions used by the truncation error estimation techniques in physical, rather than computational, space, as was performed in the quasi-1D study of this work and [9].

5.2 Bibliography

[1] Grier, B., Alyanak, E., White, M., Camberos, J., and Figliola, R., "Numerical integration techniques for discontinuous manufactured solutions," *Journal of Computational Physics*, Vol. 278, No. 0, 2014, pp. 193 – 203.

[2] Fidkowski, K. J. and Roe, P. L., "Entropy-based Mesh Refinement, I: The Entropy Adjoint Approach," AIAA Paper 2009-3790, 19th AIAA Computational Fluid Dynamics Conference, San Antonio, Texas, June 22-25, 2009.

[3] Tyson, W. C., Derlaga, J. M., Roy, C. J., and Choudhary, A., "Comparison of r-Adaptation Techniques for 2-D CFD Applications," AIAA Paper 2015-2611, 22nd AIAA Computational Fluid Dynamics Conference, Dallas, Texas June 22-26, 2013.

[4] Phillips, T. S., *Residual-based Discretization Error Estimation for Computational Fluid Dynamics*, Ph.D. thesis, Virginia Tech, October 2014.

[5] Choudhary, A., *Verification of Compressible and Incompressible Computational Fluid Dynamics Codes and Residual-based Mesh Adaptation*, Ph.D. thesis, Virginia Tech, November 2014.

[6] Phillips, T. S., Derlaga, J. M., Roy, C. J., and Borggaard, J., "Finite Volume Solution Reconstruction Methods for Truncation Error Estimation," AIAA Paper 2013-3090, AIAA Fluid Dynamics and Co-located Conferences and Exhibit, San Diego, California, June 24-27, 2013.

[7] Nielsen, E. and Kleb, W., "Efficient Construction of Discrete Adjoint Operators on Unstructured Grids Using Complex Variables," *AIAA Journal*, Vol. 44, No. 4, 2006, pp. 827–836.

[8] Choudhary, A. and Roy, C. J., "Structured Mesh r-Refinement using Truncation Error Equidistribution for 1D and 2D Euler Problems," AIAA Paper 2013-2444, AIAA Fluid Dynamics and Co-located Conferences and Exhibit, San Diego, California, June 24-27, 2013.

[9] Derlaga, J. M., Roy, C. J., and Borggaard, J., "Adjoint and Truncation Error Based Adaptation for 1D Finite Volume Schemes," AIAA Paper 2013-2865, AIAA Co-located Conferences and Exhibit, San Diego, California, June 24-27, 2013.