AIAA Aviation
13-17 June 2016, Washington, D.C.
46th AIAA Fluid Dynamics Conference

Improved Functional-Based Error Estimation and Adaptation without Adjoints

William C. Tyson*, Katarzyna (Kasia) Świrydowicz†, Joseph M. Derlaga*,
Christopher J. Roy‡ and Eric de Sturler§
Virginia Tech, Blacksburg, VA 24061

Adjoint methods have become the state of the art in recent years for both functional error estimation and adaptation. But, since most engineering applications rely upon multiple functionals to assess a physical process or system, an adjoint solution must be obtained for each functional of interest which can increase the overall computational cost significantly. In this paper, new techniques are presented for functional error estimation and adaptation which provide the same error estimates and adaptation indicators as a conventional adjoint method, but do so much more efficiently, especially when multiple functionals must be examined. For functional error estimation, the adjoint solve is replaced by the solution of an error transport equation for the local solution error and an inner product with the functional linearization. This method is shown to produce the same functional error estimate as an adjoint solve. For functional-based adaptation, the bilinear form of the adjoint correction to the functional is exploited in a manner so as to obtain the adjoint variables in an approximate sense that are still accurate enough to form useful adaptation indicators. These new methods for functional error estimation and adaptation are tested using the quasi-1D nozzle problem.

I. Introduction

FOR many computational fluid dynamics (CFD) applications, engineers require information about multiple solution functionals, such as aerodynamic forces and moments, in order to make a design choice. Such functionals are usually integrated quantities of the solution over a portion of the computational domain but can also be volume integrals or discrete values at a given point. As engineers attempt to optimize their designs to balance performance, efficiency, and cost, the success or failure of an engineering device can hinge on the accuracy of a simulation. Therefore, equipping engineers with the tools to accurately assess and reduce the numerical error in solution functionals is critical.

Adjoint methods have emerged as the state of the art for both functional error estimation and functional-based adaptation. These methods have gained popularity due to their rigorous mathematical foundation as well as their ability to correct solution functionals to higher-order. The solution to the adjoint problem can also be formulated into an adaptation indicator which targets areas of the domain for adaptation which contribute most to the overall error in a functional. Giles and Pierce pioneered much of the work on adjoints for functional error estimation by applying the continuous adjoint method to several canonical problems [1, 2]. Venditti and Darmofal [3] successfully applied the discrete adjoint method to quasi-1D, inviscid flows using an embedded grid approach where a reconstructed coarse grid solution was restricted to an embedded fine grid. In this way, they were able to obtain a higher-order error estimate in a solution functional on a fine grid using a coarse grid solution. The solution to the adjoint problem was then used to adapt the coarse grid to obtain a better error estimate on the fine grid. Venditti and Darmofal have also had much success extending their work to 2D inviscid and viscous flows [4, 5].

While adjoint methods have shown great promise for both functional error estimation and adaptation, the solution to the adjoint problem only supplies an error estimate and adaptivity information for one functional. Typically, engineers require information about multiple functionals such as three forces and three moments for a 3D aerodynamics application. With the current state of adjoint methods, engineers must solve as many adjoint problems as functionals for which they require an error estimate. As for functional-based adaptation, Park et al. [6] and Fidkowski and Roe [7] have found that adaptation based solely on the adjoint solution for one functional can actually increase the error in another functional. Therefore, even for functional-based adaptation, it is still necessary to solve multiple adjoint problems in order to improve all functionals of interest. For large engineering problems, solving multiple adjoint problems can become quite costly as the number of functionals of interest increases both in terms of storage and computational time. This could ultimately hinder the widespread use of adjoint methods in commercial applications.

*Graduate Research Assistant, Department of Aerospace and Ocean Engineering, 215 Randolph Hall, AIAA Student Member
†Graduate Research Assistant, Department of Mathematics, 460 McBryde Hall
‡Professor, Department of Aerospace and Ocean Engineering, 215 Randolph Hall, AIAA Associate Fellow
§Professor, Department of Mathematics, 460 McBryde Hall

In this paper, new techniques are proposed for performing functional error estimation and adaptation which provide the same error estimates and adaptation indicators as a conventional adjoint approach, but do so much more efficiently. For functional error estimation, this paper builds upon the work of Derlaga [8] who was able to show that local residual-based error estimates can provide the same functional error estimate as an adjoint method. Here, this is proven by showing that the solution to the adjoint problem is mathematically equivalent to the inner product of a functional linearization with a local residual-based error estimate. This novel approach not only provides local error estimates for all primitive variables (e.g., density, velocity, and pressure) everywhere in the domain but also improves the efficiency of functional error estimation for multiple functionals. Rather than solving n adjoint problems for n functionals, the same functional error estimates are obtained with one linear solve for the local solution error. For functional-based adaptation, the bilinear form of the adjoint correction to the functional is exploited using a Krylov subspace method so that the solution to the adjoint problem can be obtained in an approximate sense but is still accurate enough to drive adaptation. As byproducts of this approach, a reasonably accurate local error estimate is produced, and, due to the quadratic convergence of the Krylov solver, an accurate correction to the functional can also be attained. The quasi-1D Euler equations are used in this work to demonstrate the benefits of the proposed methods. Results are presented for both subsonic and supersonic flow regimes.

II. Residual-Based Error Estimation

A. Truncation Error

At the center of residual-based error estimation, of which adjoint methods are a subset, is the ability to accurately estimate the truncation error. Truncation error may be defined using the Generalized Truncation Error Expression (GTEE) [9, 10] which relates a given set of continuous governing equations, $L(\cdot)$, to a consistently discretized form of the equations, $L_h(\cdot)$, as

$$L_h(I^h u) = I^h L(u) + \tau_h(u) \tag{1}$$

where u is a continuous function and $\tau_h(\cdot)$ represents the truncation error. From Eq. 1, it can be seen that the truncation error is the difference between the continuous and discrete forms of the governing equations and represents higher order terms which are truncated in the discretization process. For model problems, the truncation error can be shown to be a function of continuous solution derivatives, local mesh resolution, and grid metrics [11]. The operator, I_a^b, in Eq. 1 is a restriction or prolongation operator which allows for the transition between a continuous space and a discrete space; the subscript, a, denotes the starting space and the superscript, b, denotes the resultant space. For the GTEE, a subscript or superscript h denotes a discrete space on a mesh with a characteristic size, h, and an empty subscript or superscript represents a continuous space. It should also be noted that there is no restriction on the form of the continuous and discrete operators, $L(\cdot)$ and $L_h(\cdot)$; the governing equations can be strong form (ODEs and PDEs) or weak form (integral equations).

In practice, truncation error is usually estimated by operating the continuous governing equations on a reconstruction of the discrete solution [12, 13]. Since truncation error estimation is a difficult enough task in itself and not the focus of this paper, it is assumed that an accurate estimate of the truncation error can be made. For all results in this work, exact truncation error is used instead. To evaluate the exact truncation error, the exact solution to the continuous governing equations, $\tilde{u}$, is first inserted into Eq. 1

$$L_h(I^h \tilde{u}) = I^h L(\tilde{u}) + \tau_h(\tilde{u}). \tag{2}$$

By definition, $\tilde{u}$ exactly satisfies the continuous governing equations, $L(\tilde{u}) = 0$. Therefore, the exact truncation error is evaluated by operating the discrete equations on a restriction of the exact solution

$$\tau_h(\tilde{u}) = L_h(I^h \tilde{u}). \tag{3}$$

The restriction operation required for Eq. 3 for finite volume schemes, for example, consists of integrating the exact solution over each cell to obtain the exact cell average.

B. Error Transport Equations

Using the GTEE, an error transport equation equation can be derived which relates the truncation error to the local solution error. First, inserting the exact solution to the continuous equations, $\tilde{u}$, into Eq. 1 and requiring that the continuous equations be exactly satisfied, i.e. $L(\tilde{u}) = 0$, the GTEE may be simplified as

$$\begin{aligned} L_h(I^h \tilde{u}) &= I^h \cancelto{0}{L(\tilde{u})} + \tau_h(\tilde{u}) \\ &= \tau_h(\tilde{u}). \end{aligned} \tag{4}$$

Next, the discrete equations operating on the exact discrete solution can be subtracted from Eq. 4 since they are identically zero

$$L_h(I^h\tilde{u}) - L_h(u_h) = \tau_h(\tilde{u}). \tag{5}$$

Finally, if the solution error, or discretization error, is defined as the difference between the exact solution to the discrete equations and the exact solution to the continuous equations, $\varepsilon_h = u_h - I^h\tilde{u}$, and if the governing equations are linear or linearized [10, 13], the discrete operators in Eq. 5 can be combined and the definition of discretization error may be inserted to form the discrete form of the error transport equations (ETE),

$$L_h(\varepsilon_h) = -\tau_h(\tilde{u}). \tag{6}$$

An equivalent continuous form of the ETE can also be derived. The ETE provide a great deal of information regarding how discretization error behaves and how it is related to the truncation error. Eq. 6 demonstrates that truncation error acts as the local source for discretization error and that discretization error is convected and diffused in the same manner as the solution.

For this work, the ETE are formulated in a slightly different manner. Consider an expansion of $L_h(I^h\tilde{u})$ about the exact discrete solution, u_h,

$$L_h(I^h\tilde{u}) = L_h(u_h) - \left.\frac{\partial L_h}{\partial u}\right|_{u_h} \varepsilon_h + O(\varepsilon_h^2). \tag{7}$$

Inserting Eq. 7 into Eq. 5 and canceling terms, the ETE can instead be written as

$$\left.\frac{\partial L_h}{\partial u}\right|_{u_h} \varepsilon_h = -\tau_h(\tilde{u}) + O(\varepsilon_h^2). \tag{8}$$

To obtain a second order estimate of the discretization error, the Jacobian matrix on the left hand side of Eq. 8 must be the full, second order linearization of the discrete operator. This matrix can be expensive to compute and store, but for an implicit code which already contains an adjoint solver, the full linearization should already be available. As long as an accurate representation of the truncation error is known, Eq. 8, excluding higher order terms, can be used to solve for a second order accurate estimate of the discretization error in all primal solution variables everywhere in the computational domain.

C. Adjoint Methods

Since the accuracy of functional error estimates obtained with adjoint methods, like the ETE, rely upon the accuracy of the truncation error estimate, adjoint methods can be considered a subset of residual-based error estimation. Adjoint methods were originally used in design optimization to obtain sensitivities of a functional to a set of design parameters [14], but have gained much popularity within the CFD community for obtaining functional error estimates and adaptation indicators. Solutions from the adjoint, or dual, problem can be used to drive an adaptation procedure which only targets regions of the domain which contribute to the error in the functional of interest. To illustrate how adjoint methods can be used in this way, first consider an expansion of $L_h(I^h\tilde{u})$ about the exact discrete solution, u_h,

$$L_h(I^h\tilde{u}) = L_h(u_h) - \left.\frac{\partial L_h}{\partial u}\right|_{u_h} \varepsilon_h + \left.\frac{\partial^2 L_h}{\partial u^2}\right|_{u_h} \frac{\varepsilon_h^2}{2} + O\left(\varepsilon_h^3\right). \tag{9}$$

Likewise, consider a similar expansion of a discrete solution functional, J_h, about the exact discrete solution, u_h,

$$J_h(I^h\tilde{u}) = J_h(u_h) - \left.\frac{\partial J_h}{\partial u}\right|_{u_h} \varepsilon_h + \left.\frac{\partial^2 J_h}{\partial u^2}\right|_{u_h} \frac{\varepsilon_h^2}{2} + O\left(\varepsilon_h^3\right). \tag{10}$$

Combining Eq. 9 and Eq. 10 in a Lagrangian and recalling that $L_h(u_h) = 0$, the discrete solution functional, J_h, can be rewritten as

$$J_h(I^h\tilde{u}) = \left[J_h(u_h) - \left.\frac{\partial J_h}{\partial u}\right|_{u_h} \varepsilon_h + \left.\frac{\partial^2 J_h}{\partial u^2}\right|_{u_h} \frac{\varepsilon_h^2}{2} + O\left(\varepsilon_h^3\right)\right] + \lambda^T \left[L_h(I^h\tilde{u}) + \left.\frac{\partial L_h}{\partial u}\right|_{u_h} \varepsilon_h - \left.\frac{\partial^2 L_h}{\partial u^2}\right|_{u_h} \frac{\varepsilon_h^2}{2} + O\left(\varepsilon_h^3\right)\right] \tag{11}$$

where λ^T is a 1xN vector of Lagrange multipliers and N is the total number of unknowns in the system. This operation is made possible by the fact that the terms in brackets following the Lagrange multipliers are exactly equal to zero by

Eq. 9. Moving the first term on the right hand side to the left hand side, the error in the functional, ε_{J_h}, can be defined as

$$\varepsilon_{J_h} = J_h(u_h) - J_h(I^h\tilde{u}) = \left[\left.\frac{\partial J_h}{\partial u}\right|_{u_h}\varepsilon_h - \left.\frac{\partial^2 J_h}{\partial u^2}\right|_{u_h}\frac{\varepsilon_h^2}{2} + O\left(\varepsilon_h^3\right)\right] - \lambda^T\left[L_h(I^h\tilde{u}) + \left.\frac{\partial L_h}{\partial u}\right|_{u_h}\varepsilon_h - \left.\frac{\partial^2 L_h}{\partial u^2}\right|_{u_h}\frac{\varepsilon_h^2}{2} + O\left(\varepsilon_h^3\right)\right]. \quad (12)$$

Upon inspection, Eq. 12 can be viewed as a constrained optimization problem which seeks to minimize the error in the functional subject to satisfying the discrete governing equations, $L_h(u_h) = 0$. It is left as an exercise for the reader to show that Eq. 12 can be rewritten in the following form

$$\varepsilon_{J_h} = \underbrace{J_h(u_h) - J_h(I^h\tilde{u})}_{\text{Functional Error}} \approx \underbrace{-\lambda^T\tau_h(\tilde{u})}_{\text{Correction}} + \underbrace{\left[\left.\frac{\partial J_h}{\partial u}\right|_{u_h} - \lambda^T\left.\frac{\partial L_h}{\partial u}\right|_{u_h}\right]}_{\text{Adjoint Problem}}\varepsilon_h + \underbrace{\left[-\left.\frac{\partial^2 J_h}{\partial u^2}\right|_{u_h} + \lambda^T\left.\frac{\partial^2 L_h}{\partial u^2}\right|_{u_h}\right]\frac{\varepsilon_h^2}{2}}_{\text{Remaining Error}}. \quad (13)$$

Eq. 13 represents the discrete adjoint formulation of the functional error. An equivalent continuous formulation can be derived in a similar manner. In the context of adjoint based error estimation, the Lagrange multipliers, λ, are referred to as the adjoint variables or adjoint sensitivities.

The functional error estimate consists of three parts: (1) the computable adjoint correction, (2) the adjoint problem, and (3) remaining error terms. The adjoint correction is computed by first solving the adjoint problem such that the terms in brackets are equal to zero

$$\left.\frac{\partial J_h}{\partial u}\right|_{u_h} - \lambda^T\left.\frac{\partial L_h}{\partial u}\right|_{u_h} = 0. \quad (14)$$

Rearranging Eq. 14 yields a system of linear equations for the adjoint variables, λ,

$$\left[\left.\frac{\partial L_h}{\partial u}\right|_{u_h}\right]^T\lambda = \left[\left.\frac{\partial J_h}{\partial u}\right|_{u_h}\right]^T. \quad (15)$$

The first term on the left hand side of Eq. 15 is the transpose of the Jacobian of the primal solution residual which, for a code that includes an implicit solver, should already be computed and stored. However, many codes which implement an implicit solver only store a first order approximation of the Jacobian matrix to decrease storage and computational costs. For the adjoint solver, a full linearization of the primal residual is required in order to obtain a second order error estimate. Upon further inspection of Eq. 15, it can be seen that the adjoint variables represent the discrete sensitivities of the functional to perturbations in the discrete operator making them an ideal candidate for flagging areas of the domain for adaptation which contribute most to error in the chosen functional.

Once the adjoint variables have been determined, the adjoint correction to the functional may be computed as the inner product of the adjoint variables with the truncation error. For a second order discretization, for example, the adjoint correction to the functional converges at a second order rate with grid refinement. If the exact value of the functional, $J_h(I^h\tilde{u})$, is known, the adjoint correction can be used to quantify the remaining error in the functional. On the other hand, for cases where the exact value of the functional is not known, the adjoint correction can serve as an estimate of the functional error or can be used to correct the functional. Since the discretization error, ε_h, also converges at a second order rate, application of the adjoint correction increases the convergence rate of the functional error to fourth order. This higher order behavior coupled with the extension to functional-based adaptation have made adjoints the method of choice for functional error estimation and adaptation in CFD.

III. Improved Functional Error Estimation and Adaptation

A. Error Estimation for Multiple Functionals

Adjoint methods appear to be ideal for functional error estimation and adaptation since both error estimates and adaptation indicators can be obtained in one compact procedure. But, since most engineering problems require analyzing multiple solution functionals, the adjoint problem must be solved for each functional of interest. As the number of required functionals increases, the cost of solving several adjoint problems can become impractical. For such applications, more efficient methods for functional error estimation should be used. In this section, a new approach is presented for obtaining functional error estimates which can increase the efficiency of functional error estimation when multiple functionals are required. With this approach, n adjoint solves for n functionals are replaced by one linear solve for the local solution error everywhere in the domain. First, recall that the functional error estimate, Eq. 13, with the remaining error terms truncated can be written as

$$\varepsilon_{J_h} \approx -\lambda^T\tau_h(\tilde{u}) \quad (16)$$

where λ^T are the adjoint variables obtained from solving the adjoint problem and $\tau_h(\tilde{u})$ is the truncation error. It should be noted that the right hand side of Eq. 16 is often referred to as the functional correction since it can also be used to increase the accuracy of the computed functional. Now, rather than explicitly solve for the adjoint variables, Eq. 14 can instead be used to rewrite the adjoint correction as

$$\varepsilon_{J_h} \approx -\left.\frac{\partial J_h}{\partial u}\right|_{u_h} \left[\left.\frac{\partial L_h}{\partial u}\right|_{u_h}\right]^{-1} \tau_h(\tilde{u}) \tag{17}$$

where $\left.\frac{\partial J_h}{\partial u}\right|_{u_h}$ is a linearization of the discrete functional with respect to the solution everywhere in the domain and $\left[\left.\frac{\partial L_h}{\partial u}\right|_{u_h}\right]^{-1}$ is the inverse of the residual Jacobian. Computing the inverse of the residual Jacobian is extremely expensive and should never be used to determine the adjoint correction. Alternatively, using Eq. 8, we can see that the right two terms in Eq. 17 are equivalent to the local solution error obtained from solving the ETE. Therefore, the adjoint correction can either be thought of as the inner product of the adjoint solution with the truncation error or as the inner product of the ETE solution with the functional linearization

$$\varepsilon_{J_h} \approx \overbrace{-\left.\frac{\partial J_h}{\partial u}\right|_{u_h} \underbrace{\left[\left.\frac{\partial L_h}{\partial u}\right|_{u_h}\right]^{-1} \tau_h(\tilde{u})}_{\text{Error Transport Equations}}}^{\text{Adjoint Problem}} . \tag{18}$$

Rather than solve n adjoint problems for n functionals, the ETE are instead solved once. It is important to note that the computation time associated with the required inner product for computing the functional correction is present in both the standard adjoint approach and the ETE approach. Therefore, the computational savings of the ETE approach are on the order of the time required for $n-1$ linear solves. Another important benefit of the ETE approach is that, unlike the standard adjoint approach, the local solution error is also obtained everywhere in the domain. In Section V, the ETE approach is shown to obtain the same functional error estimate as an adjoint method. The computational savings that can be achieved are also presented.

B. Approximate Adjoints for Functional Error Estimation and Adaptation

For functional-based adaptation, the adjoint variables are a logical candidate for an adaptation indicator since they represent the sensitivity of the functional with respect to perturbations of the residual operator. An adaptation indicator formed with the adjoint variables will target regions of the domain which contribute most to the error in the functional. One flaw of the ETE approach to functional error estimation is that the engineer no longer has access to the adjoint variables for performing adaptation. In order to perform both adaptation and error estimation, the adjoint problem must typically be solved to convergence. But, since the formulation of the adaptation indicator is somewhat heuristic in nature, for adaptation purposes the adjoint variables do not need to be extremely accurate. They only need to be accurate enough to capture the main features of an adaptation indicator based on the fully converged adjoint solution. In this section, an alternative approach based upon Krylov subspace methods is presented which can meet both needs: an accurate functional error estimate as well as approximate adjoint variables for adaptation. This technique only requires the adjoint problem and the ETE be solved approximately. With one linear solve, approximate adjoint variables are obtained for adaptation alongside a reasonably accurate local error estimate. Due to the quadratic convergence of the Krylov subspace method, the adjoint correction can also be obtained just as accurately as the standard adjoint approach or the ETE approach to functional error estimation.

Approximating Bilinear Form by Krylov Subspace Methods

The evaluation of the adjoint correction in Eq. 17 is equivalent to evaluating a bilinear form

$$BF(b, c; A) = c^T A^{-1} b, \tag{19}$$

where A is a real-valued $n \times n$ matrix, and b and c are real-valued $n \times 1$ vectors. Problems of this form appear in many applications, such as *scattering amplitude* [15], computational fluid dynamics [16], quantum physics and inverse problems [17], and naturally arise in numerical linear algebra, for example, while computing error bounds. With this approach, the goal is to approximate Eq. 19 without directly computing the inverse of A. The bilinear form in Eq. 19 can be approximated by solving two linear systems simultaneously

$$\begin{aligned} Ax = b \quad &\Leftrightarrow \quad x = A^{-1} b, \\ A^T y = c \quad &\Leftrightarrow \quad y = A^{-T} c, \end{aligned} \tag{20}$$

and computing

$$y^T Ax = \left(A^{-T}c\right)^T AA^{-1}b = c^T A^{-1}b. \tag{21}$$

The second system in Eq. 20 is referred to as an *adjoint system* whereas the first system is referred to as the *main system*. Many Krylov subspace methods (i.e. BiCG [18,19], LSQR [20]) solve the adjoint system and the main system simultaneously. This property has been exploited in the literature [15,21,22].

One approach to approximating the bilinear form presented in [21] is based on BiCG [19]. BiCG is used to solve the main system and the solution of the adjoint system is obtained concurrently. With this method, one constructs two Krylov subspaces associated with the main and the adjoint systems, $V_k = span\{r_0, Ar_0, \ldots, A^{k-1}r_0\}$ and $\tilde{V}_k = span\{s_0, A^T s_0, \ldots, A^{T^{k-1}} s_0\}$, where r_0 and s_0 are the initial residuals of the respective systems. The method enforces, in exact arithmetic, $V_k \perp_b \tilde{V}_k$. This approach has two major advantages. Firstly, the convergence of the method is quadratic. Therefore, if $\|r_i\| \leq h_1$, $\|s_i\| \leq h_2$, where r_i and s_i are the residuals for the main and adjoint systems, respectively, and $h = \max(h_1, h_2)$, then the error in the approximation to the bilinear form is smaller than a constant multiple of $h_1 h_2$. Also, if an approximation of the bilinear form is needed to an accuracy ε, then the main and adjoint systems need to only be solved to an accuracy of $\sqrt{\varepsilon}$. With regards to functional error estimation and adaptation, this means that approximate adjoint variables can be obtained for adaptation while also obtaining an accurate functional correction. The second advantage is that the approximation to Eq. 19 is computed and updated as a part of the BiCG iteration with very minimal additional cost due to the recurrences demonstrated by Strakoš and Tichý [21]. The recurrences rely only on the orthogonality relationships between vectors generated and updated in the neighboring iterations, which minimizes the adverse effects of deterioration from orthogonality resulting from finite precision arithmetic.

In this work, the approach of Strakoš and Tichý [21] is used. In their paper, the bilinear form is approximated at iteration n of the solver using the property

$$c^T A^{-1}b = \sum_{j=0}^{n-1} \alpha_j s_j^T r_j + s_n^T A^{-1} r_n, \tag{22}$$

where $s_j = A^T y_j - c$, $r_j = Ax_j - b$, and x_j and y_j are approximations to the solutions of main and adjoint systems, respectively, computed at iteration j of BiCG. The quantity α_j is a scalar used by the BiCG iteration. As n increases, the residuals and the quantity $s_n^T A^{-1} r_n$ become small. Thus, the last term can be skipped, and Eq. 22 becomes

$$BF(b, c; A) \approx \sum_{j=0}^{n-1} \alpha_j s_j^T r_j \tag{23}$$

where the initial approximation is computed as

$$\begin{aligned} c^T A^{-1}b &= \left(\tilde{r}_0 + A^T \tilde{x}_0\right)^T A^{-1}\left(r_0 + Ax_0\right) \\ &= \tilde{r}_0^T A^{-1} r_0 + \left(\tilde{r}_0 + A^T \tilde{x}_0\right)^T x_0 + \tilde{x}_0^T r_0 \\ &= \tilde{r}_0^T A^{-1} r_0 + c^T x_0 + \tilde{x}_0^T r_0 \quad \Rightarrow \quad BF_0 = c^T x_0 + \tilde{x}_0^T r_0. \end{aligned} \tag{24}$$

At the conclusion of the BiCG iteration, the approximate solutions to the main and adjoint systems, x_j and y_j, provide a local error estimate as well as variables for formulating an adaptation indicator. The approximate solution to the bilinear form, Eq. 19, can also be used as a functional error estimate.

IV. Test Case: Quasi-1D Nozzle

The quasi-1D nozzle problem is used to investigate the proposed methods for functional error estimation and adaptation. The nozzle is characterized by a contraction through which the flow accelerates followed by a diverging section which allows the flow to expand subsonically or supersonically depending upon the pressure at the nozzle exit. Since the quasi-1D nozzle problem has an exact solution, it is the ideal test bed for investigating these new methods.

The geometry of the nozzle is taken from Jackson and Roy [23] and has a Gaussian area distribution given by

$$A(x) = 1 - 0.8e^{\left(\frac{-x^2}{2\sigma^2}\right)}, \quad x \in [-1, 1] \tag{25}$$

where σ is chosen as 0.2. For this study, two flow conditions are examined: (1) fully subsonic flow through the diverging section and (2) fully supersonic flow through the diverging section. The area distribution and the Mach number distributions for both the subsonic and supersonic cases can be seen in Fig. 1.

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

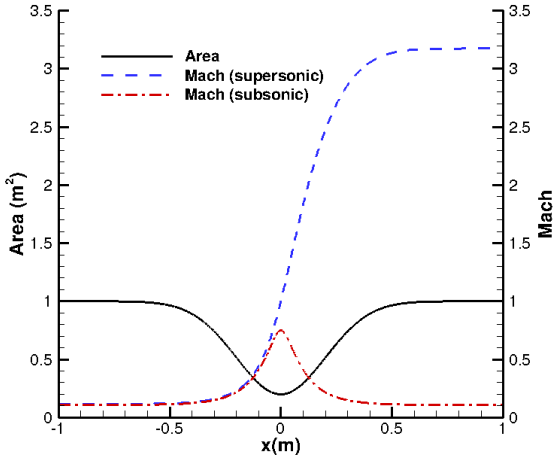


Figure 1: Gaussian Nozzle: Area/Mach Number Distribution

For this study, five discrete solution functionals are used to test the proposed methods for functional error estimation and adaptation. A listing of these functionals is given below. Take note that the continuous form of the functional is given by $J(\cdot)$ and the discrete form of the functional is given by $J_h(\cdot)$.

$$\begin{aligned}
&\text{Integral of Pressure :} && J(\tilde{u}) = \int_{-1}^{1} p\, dx && \Rightarrow && J_h(u_h) = \sum_{i=1}^{N_{cells}} p_i \Delta x_i \\
&\text{Integral of Entropy :} && J(\tilde{u}) = \int_{-1}^{1} \log\left(\frac{p}{\rho^\gamma}\right) dx && \Rightarrow && J_h(u_h) = \sum_{i=1}^{N_{cells}} \log\left(\frac{p_i}{\rho_i^\gamma}\right) \Delta x_i \\
&\text{Total Mass :} && J(\tilde{u}) = \int_{-1}^{1} \rho A\, dx && \Rightarrow && J_h(u_h) = \sum_{i=1}^{N_{cells}} \rho_i A_i \Delta x_i \\
&\text{Integral of Mass Flow :} && J(\tilde{u}) = \int_{-1}^{1} \rho u A\, dx && \Rightarrow && J_h(u_h) = \sum_{i=1}^{N_{cells}} \rho_i u_i A_i \Delta x_i \\
&\text{Static Thrust :} && J(\tilde{u}) = \rho_e u_e^2 A_e + (p_e - p_a) A_e && \Rightarrow && J_h(u_h) = \rho_e u_e^2 A_e + (p_e - p_a) A_e
\end{aligned} \tag{26}$$

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

A. Numerical Methods

1. *Quasi-1D Euler Equations*

The flow in the nozzle is modeled using the quasi-1D form of the Euler equations. The quasi-1D Euler equations in weak, conservation form for a control volume, Ω^+, fixed in space are

$$\frac{\partial}{\partial t^+} \int_{\Omega^+} \vec{Q^+}\, d\Omega^+ + \oint_{\partial\Omega^+} \vec{F^+}\, dS^+ = \int_{\Omega^+} \vec{S^+}\, d\Omega^+ \tag{27}$$

where $\vec{Q^+}$ is the vector of conserved variables, $\vec{F^+}$ is the vector of inviscid fluxes, and $\vec{S^+}$ is a source term. The notation $(\cdot)^+$ denotes a dimensional quantity. These equations are a statement of conservation of mass, momentum, and energy

for the fluid. For the quasi-1D Euler equations, the vector of conserved variables, $\vec{Q^+}$, the inviscid fluxes, $\vec{F^+}$, and the source term, $\vec{S^+}$, are given by

$$\vec{Q^+} = \begin{bmatrix} \rho^+ \\ \rho^+ u^+ \\ \rho^+ e_t^+ \end{bmatrix}, \quad \vec{F^+} = \begin{bmatrix} \rho^+ u^+ \\ \rho u^{+2} + p^+ \\ \rho^+ u^+ h_t^+ \end{bmatrix}, \quad \vec{S^+} = \begin{bmatrix} 0 \\ p^+ \frac{dA^+}{dx^+} \\ 0 \end{bmatrix} \tag{28}$$

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

$$e_t^+ = \frac{p^+}{\rho^+(\gamma - 1)} + \frac{u^{+2}}{2}, \quad h_t^+ = \frac{\gamma p^+}{\rho^+(\gamma - 1)} + \frac{u^{+2}}{2} \tag{29}$$

where γ is the ratio of specific heats for a perfect gas which for air is $\gamma = 1.4$.

2. *Primal Solver*

The quasi-1D Euler equations are discretized using a second order, cell-centered finite-volume discretization. The inviscid fluxes are computed using Roe's flux difference splitting scheme [24]. MUSCL extrapolation [25] is used to reconstruct the primitive variables to the face in order to obtain second order spatial accuracy. Boundary conditions are enforced weakly through the fluxes.

The quasi-1D Euler equations are coded in a dimensional and non-dimensional form to allow for greater flexibility. In order to write the quasi-1D Euler equations in a non-dimensional form, Eq. 27 is first discretized for cell i as

$$\frac{A_i^+ \Delta x_i^+}{\Delta t^+} \left[\vec{Q^+}_i^{n+1} - \vec{Q^+}_i^{n} \right] + \vec{R^+}_i^{n+1} = 0 \tag{30}$$

where A_i^+ is the cross-sectional area at the center of cell i, Δx_i^+ is the size of cell i, Δt^+ is the time step, and $\vec{R^+}_i^{n+1}$ is the steady-state residual. The steady-state residual can be written as

$$\vec{R^+}_i^{n+1} = \vec{F^+}_{i+\frac{1}{2}}^{n+1} A_{i+\frac{1}{2}}^+ - \vec{F^+}_{i-\frac{1}{2}}^{n+1} A_{i-\frac{1}{2}}^+ - \Delta x_i^+ \vec{S^+}_i^{n+1} \tag{31}$$

where $\vec{F^+}_{i+\frac{1}{2}}^{n+1}$ and $\vec{F^+}_{i-\frac{1}{2}}^{n+1}$ are the inviscid fluxes at the faces on each side of cell i and $A_{i+1/2}^+$ and $A_{i-1/2}^+$ are the corresponding face areas. The non-dimensionalization is performed by normalizing each dimensional quantity in the following manner

$$\begin{aligned} T &= \frac{T^+}{T_{ref}} & \rho &= \frac{\rho^+}{\rho_{ref}} & u &= \frac{u^+}{a_{ref}} \\ p &= \frac{p^+}{\rho_{ref} a_{ref}^2} & e_t &= \frac{e_t^+}{a_{ref}^2} & h_t &= \frac{h_t^+}{a_{ref}^2} \\ x &= \frac{x^+}{L_{ref}} & t &= \frac{t^+}{L/a_{ref}} & A &= \frac{A^+}{A_{ref}} \end{aligned} \tag{32}$$

where the reference values are chosen as

$$\begin{aligned} T_{ref} &= T_0 & \rho_{ref} &= \rho_0 & a_{ref} &= \sqrt{\gamma R T_{ref}} \\ L_{ref} &= L_{nozzle} & A_{ref} &= A_{throat}. \end{aligned} \tag{33}$$

It should be noted that T_0 is the stagnation temperature, ρ_0 is the stagnation density, L_{nozzle} is the length of the nozzle, and A_{throat} is the minimum area of the nozzle cross-section. By normalizing Eq. 30 with Eq. 32, the quasi-1D Euler equations may be written in a non-dimensional form as

$$\frac{A_i \Delta x_i}{\Delta t} \left[\vec{Q}_i^{n+1} - \vec{Q}_i^{n} \right] + \vec{R}_i^{n+1} = 0 \tag{34}$$

where the non-dimensional steady-state residual $\vec{R}_i^{n+1}$ is given by

$$\vec{R}_i^{n+1} = \vec{F}_{i+\frac{1}{2}}^{n+1} A_{i+\frac{1}{2}} - \vec{F}_{i-\frac{1}{2}}^{n+1} A_{i-\frac{1}{2}} - \Delta x_i \vec{S}_i^{n+1}. \tag{35}$$

Numerical solutions are marched in pseudo-time to a steady-state using an implicit time-stepping scheme developed by Beam and Warming [26]

$$\left[\frac{\Omega}{\Delta t}I + \frac{\partial \vec{R}}{\partial \vec{Q}}\right]^n \Delta \vec{Q}^n = -\vec{R}^n \tag{36}$$

where $\frac{\Omega}{\Delta t}I$ is a matrix with the volume and time step contribution from each cell along the main diagonal, $\frac{\partial \vec{R}}{\partial \vec{Q}}$ is the residual Jacobian matrix, $\Delta \vec{Q}^n$ is a forward difference of the conserved variable vector given by $\Delta \vec{Q}^n = \vec{Q}^{n+1} - \vec{Q}^n$, and $\vec{R}^n$ is the steady-state residual evaluated at time level n. For this implementation, only the primitive variables are stored, therefore, a conversion matrix is added to convert the conserved variable vector, $\vec{Q}$, to the primitive variable vector, $\vec{q}$, such that Eq. 36 is instead written as

$$\left[\frac{\Omega}{\Delta t}\frac{\partial \vec{Q}}{\partial \vec{q}} + \frac{\partial \vec{R}}{\partial \vec{q}}\right]^n \Delta \vec{q}^n = -\vec{R}^n \tag{37}$$

where $\frac{\partial \vec{Q}}{\partial \vec{q}}$ is the conversion matrix and $\vec{q}$ is the primitive variable vector given by $\vec{q} = [\rho, u, p]^T$. For the primal solver, the residual Jacobian matrix, $\frac{\partial \vec{R}}{\partial \vec{q}}$, need not be the full second order linearization since the left hand side of Eq. 37 is only used to drive the right hand side to zero. In order to maintain second order spatial accuracy, though, it is mandatory that the residual be a second order discretization.

3. *Dual Solver & Error Transport Equation Solver*

The discrete adjoint problem is solved in a similar manner to that outlined in [27]. The adjoint problem is marched in pseudo-time to a steady-state using an implicit time-stepping algorithm much like the primal solver

$$\left[\frac{\Omega}{\Delta t}\frac{\partial \vec{Q}}{\partial \vec{q}} + \frac{\partial \vec{R}}{\partial \vec{q}}\right]^T \Delta \vec{\lambda}^n = -\vec{R}^n_{adj} \tag{38}$$

where $\vec{R}^n_{adj}$ is the adjoint residual given by

$$\vec{R}^n_{adj} = -\left[\frac{\partial J_h}{\partial \vec{q}}\right]^T + \left[\frac{\partial \vec{R}}{\partial \vec{q}}\right]^T \vec{\lambda}^n \tag{39}$$

and $\Delta \vec{\lambda}^n$ is a forward difference of the adjoint variables given by $\Delta \vec{\lambda}^n = \vec{\lambda}^{n+1} - \vec{\lambda}^n$. This type of algorithm is often used because the pseudo-time-stepping term increases the diagonal dominance of the left hand side matrix making for a more robust adjoint solver. Also, similar to the primal solver, the residual Jacobian of the primal solution on the left hand side of Eq. 38 need not be the full second order linearization. But, in order to obtain a usable adjoint solution, the residual Jacobian on the right hand side of Eq. 38 is required to be the full second order linearization. The linearization of the primal residual is performed via hand differentiation.

The ETE are solved in a similar manner using

$$\left[\frac{\Omega}{\Delta t}\frac{\partial \vec{Q}}{\partial \vec{q}} + \frac{\partial \vec{R}}{\partial \vec{q}}\right] \Delta \varepsilon_h^n = -\vec{R}^n_{ETE} \tag{40}$$

where $\vec{R}^n_{ETE}$ is the ETE residual given by

$$\vec{R}^n_{ETE} = \tau_h(\tilde{u}) + \frac{\partial \vec{R}}{\partial \vec{q}}\varepsilon_h^n \tag{41}$$

and $\Delta \varepsilon_h^n$ is a forward difference of the discretization error given by $\Delta \varepsilon_h^n = \varepsilon_h^{n+1} - \varepsilon_h^n$. Both the adjoint problem and the ETE, Eq. 38 and Eq. 40, are solved using GMRES [28]. For the code used in this study, there is also the capability of solving the adjoint problem and the ETE directly using GMRES. This design choice is made so that runtimes of the adjoint and ETE solves can be fairly compared with the bilinear form solve.

4. Preconditioning

Krylov methods, such as BiCG and GMRES, often need a preconditioner. By multiplying by a preconditioner matrix, either on the left, on the right, or on both sides of the system, one obtains a new system with much better convergence properties, i.e.

$$Ax = b \quad \Rightarrow \quad M_1 A M_2 \tilde{x} = M_1 b, \quad x = M_2 \tilde{x}. \tag{42}$$

The matrix M_1 in Eq. 42 is called a *left preconditioner*, and the matrix M_2 is called a *right preconditioner*. There exist many preconditioning techniques, such as Incomplete LU factorization (ILU) [29] and Sparse Approximate Inverse (SAI) [30, 31]. While ILU is arguably the most popular preconditioning technique, due to the small dimensionality of the quasi-1D nozzle problem, the ILU preconditioner is very close to a true LU factorization and does not allow us to show the benefits of the bilinear form approach to functional error estimation and adaptation. When testing the bilinear form solver, SAI is used instead to precondition both BiCG and GMRES. SAI is considered a *weak preconditioner*, yet due to its exceptional parallel properties has been rediscovered in recent years and gained popularity as a GPU preconditioning strategy [32]. Since the true inverse of a sparse matrix is often dense, it is necessary to select an initial non-zero pattern for SAI. In this paper, the non-zero pattern of the SAI preconditioner is set to be identical to the non-zero pattern of A unless specified otherwise. Since the code used in this study is capable of solving the governing equations in a dimensional and non-dimensional form, matrix scaling is also included as a preconditioning technique for the dimensional matrix systems. Matrix scaling is performed by applying row scaling from the left and column scaling from the right

$$Ax = b \quad \Rightarrow \quad D_r A D_c \tilde{x} = D_r b \tag{43}$$

where D_r and D_c are diagonal matrices whose entries are

$$D_{r,i} = \frac{1}{\sum_{j=1}^{n} |A_{i,j}|} \quad D_{c,j} = \frac{1}{\sum_{i=1}^{n} |A_{i,j}|}. \tag{44}$$

5. Grid Adaptation

There are many types of solution adaptation procedures, but for this study, solution adaptation is performed via mesh movement, or r-adaptation. Mesh movement is accomplished by applying an equidistribution principle to the mesh given some adaptation indicator or weight function. During the equidistribution process, nodes of the mesh are shifted, as illustrated in Fig. 2, in order to make the quantity cell size (i.e. Δx in 1D) times weight function equal-valued in all cells across the domain.

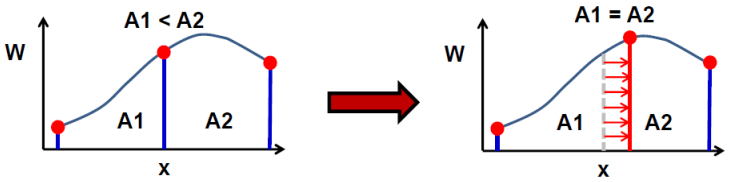


Figure 2: Weight Function Equidistribution [23]

With this type of adaptation, the number of nodes in the mesh remains constant throughout the adaptation process. This can arguably lead to significant computational savings over other forms of adaptation, such as h-adaptation, where the number of nodes in the mesh can increase. The equidistribution principle in 1D can be stated as the following

$$\frac{d}{d\xi}\left(w(x)\frac{dx}{d\xi}\right) = 0 \tag{45}$$

where $w(x)$ is the weight function, x is the physical location of the mesh, and ξ is the location of the mesh in a uniformly spaced computational domain [33]. Eq. 45 is discretized using second order central finite differences as

$$\frac{1}{\Delta\xi}\left[\left(\frac{w(x_{i+1}^n) + w(x_i^n)}{2}\right)\frac{x_{i+1}^{n+1} - x_i^{n+1}}{\Delta\xi} - \left(\frac{w(x_i^n) + w(x_{i-1}^n)}{2}\right)\frac{x_i^{n+1} - x_{i-1}^{n+1}}{\Delta\xi}\right] = 0 \tag{46}$$

Prior to conducting adaptation, care must be taken to formulate the weight function in a manner which will achieve the desired adaptive behavior. For this work, the weight function developed by Derlaga et al. [34, 35] and Derlaga [8] is used to drive the adaptation process. This weight function is formulated by normalizing the adjoint-weighted

truncation error in cell i on a given mesh, k, by the L_2-norm of the adjoint-weighted truncation error on the initial mesh and averaging over N governing equations

$$w_i = \frac{1}{N} \sum_{j=1}^{N} \frac{\left| \lambda_{i,j} \tau_h (I^h \tilde{u})_{i,j}^k \right|}{\left\| \lambda_j \tau_h (I^h \tilde{u})_j^{k=1} \right\|_2} \tag{47}$$

where the number of equations for the quasi-1D Euler equations is $N = 3$.

Often with this type of adaptation, the weight function can exhibit high frequency behavior which can destabilize the mesh movement procedure. In order to maintain a stable adaptation process, the weight function is smoothed prior to its application using an elliptic smoother,

$$w_i^{smooth} = \frac{w_{i-1} + 4w_i + w_{i+1}}{6}. \tag{48}$$

For this work, the number of smoothing passes is set to $N_{smooth} = 15$. Depending upon the strength of the weight function, under-relaxation of the grid update may also need to be performed to ensure the adaptation process converges. Under-relaxation of the grid update is accomplished using

$$x^{new} = x^{old} + \omega \left(x_{computed}^{new} - x^{old} \right) \tag{49}$$

where the under-relaxation factor, ω, is set to $\omega = 0.05$. The L_2-norm of the weight function times the cell area for each cell in mesh k normalized by the initial value on the starting mesh

$$\Upsilon = \frac{\|w_i \Delta x_i\|_2^k}{\|w_i \Delta x_i\|_2^{k=1}} \tag{50}$$

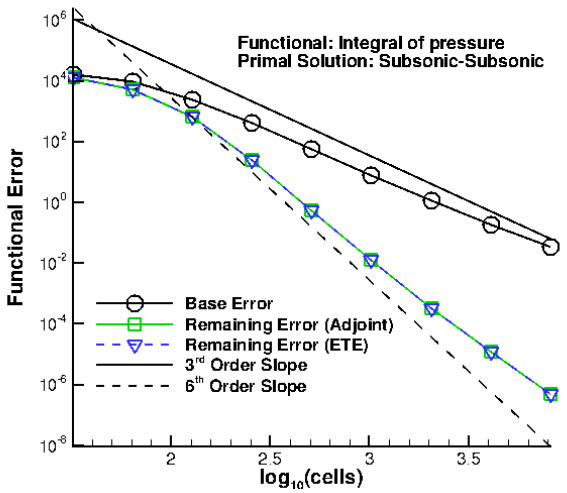
is used to monitor the convergence of the adaptation process. Adaptation is said to be converged when $\Upsilon \leq 0.08$.

V. Results & Discussion

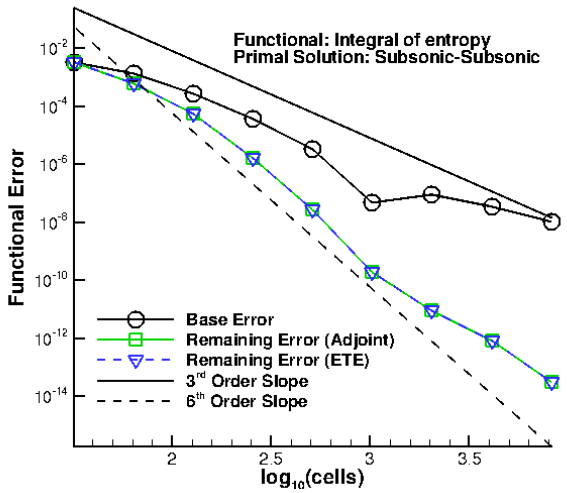
In this section, results are presented for subsonic and supersonic cases of the quasi-1D nozzle problem which demonstrate the benefits of the proposed methods over a traditional adjoint approach. It should be noted that during preliminary testing the dimensional and non-dimensional form of the quasi-1D Euler equations were examined. Since the dimensional primitive variables are vastly different magnitudes, for example density is $O(1)$ and pressure is $O(10^5)$, the residual Jacobian matrices are very poorly scaled. For the finest grids, the residual Jacobian matrices are essentially singular with conditions numbers of $O(10^{14})$. Row and column scaling is used to improve the condition of the matrix, but, even with scaling, the non-dimensional residual Jacobian matrices are still better conditioned. A comparison of the condition numbers for the dimensional and non-dimensional residual Jacobian matrices for both subsonic and supersonic cases may be found in Appendix A. It is also found that the poor conditioning of the dimensional residual Jacobian matrices affects the accuracy of the bilinear form solution. For these reasons, all results in this section are produced using the non-dimensional form of the quasi-1D Euler equations.

A. Error Estimation for Multiple Functionals

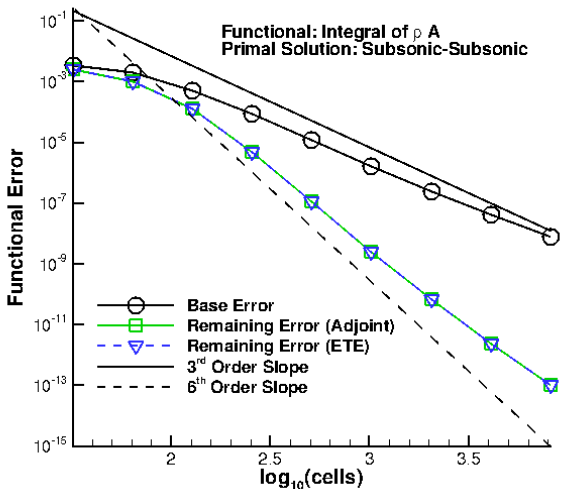
The ETE and the adjoint problem are solved on 9 systematically refined, uniformly spaced grids ranging from 32 cells to 8192 cells where the refinement factor between grid levels is $r = 2$. The solution to each system is obtained using the pseudo-time-stepping technique outlined in Section IV with the pseudo-time-step fixed at $\Delta t = 0.5$. Functional corrections are computed for the five functionals given in Section IV. The base error and remaining error after correction are plotted for each functional in Fig. 3 against the number of cells in the grid. Results presented here are only for the subsonic case. Similar results were obtained for the supersonic case. Fig. 3 illustrates that the ETE are able to obtain the same functional correction as a standard adjoint approach. The differences between the remaining error obtained with a standard adjoint approach and the remaining error obtained with the ETE approach are indistinguishable. Upon application of the functional correction, the remaining error in each functional converges at a higher order. While the formal order of the discretization is $p_f = 2$, the base error in each functional converges at a 3rd order rate and the remaining error converges at a 6th order rate. This behavior was also observed by Venditti and Darmofal [3] for a similar subsonic test case. For the supersonic case, the base error and remaining error converge at the expected rates, 2nd order and 4th order, respectively. In addition to an accurate functional error estimate, it should also be noted that with the ETE approach a local error estimate is obtained everywhere in the domain. Plots of the exact discretization error and the error estimate obtained with the ETE for each primitive variable on a 256 cell grid may be seen in Fig. 4.



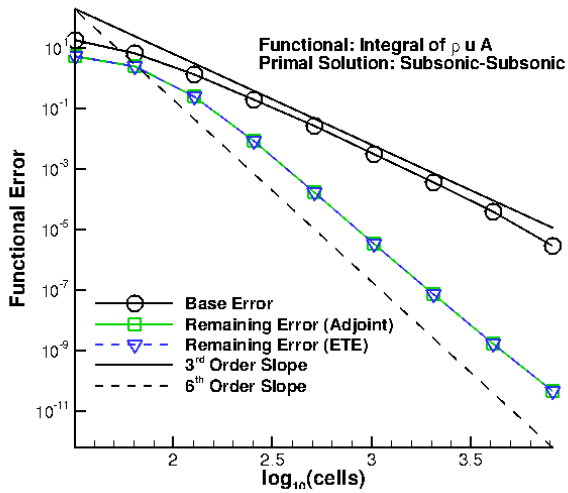
a) Functional: Integral of Pressure



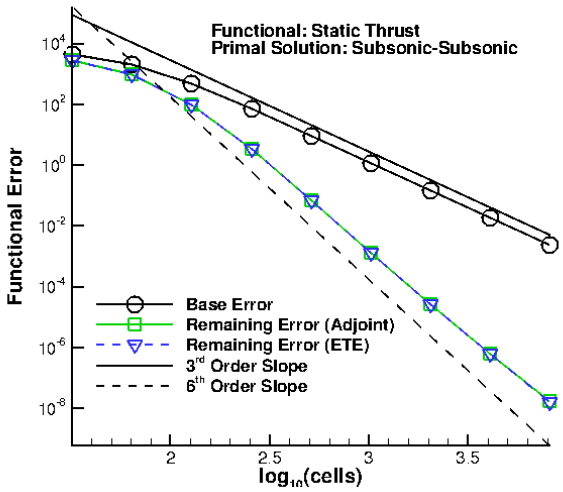
b) Functional: Integral of Entropy



c) Functional: Total Mass

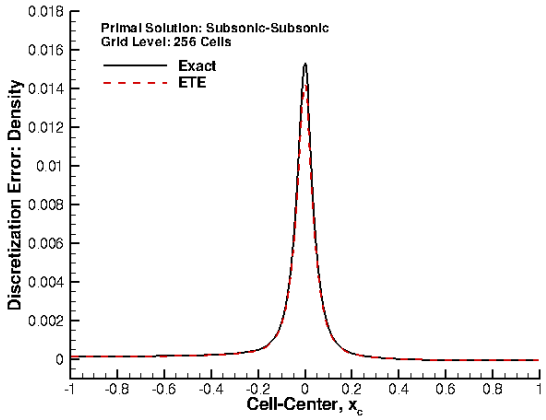


d) Functional: Integral of Mass Flow

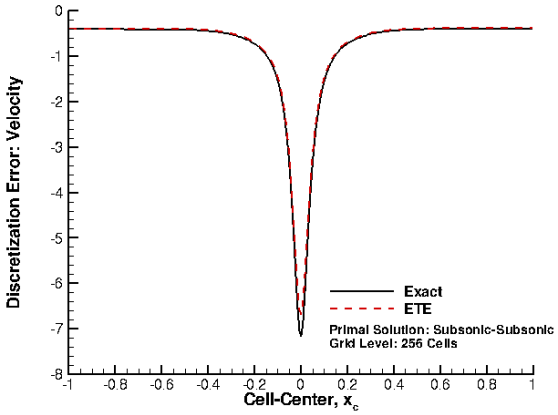


e) Functional: Static Thrust

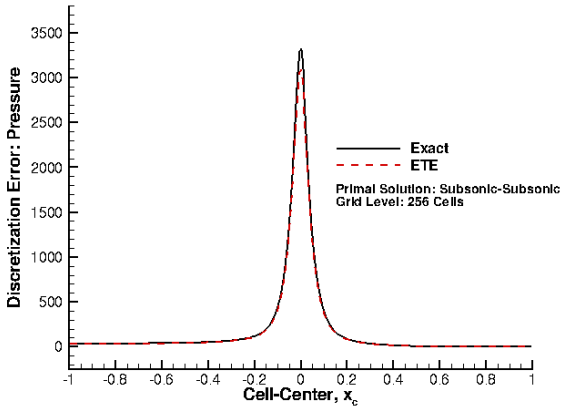
Figure 3: Functional Error Convergence – Subsonic Case



a) Discretization Error: Density



b) Discretization Error: Velocity



c) Discretization Error: Pressure

Figure 4: Discretization Error – Subsonic Case

The total runtime need to compute functional corrections for all five functionals is also monitored. Runtimes for the standard adjoint approach and the ETE approach for each grid level are given in Table 1. The speedup gained by using the ETE approach is also quoted. Speedup is defined as the total solve time of the adjoint approach divided by the total solve time of the ETE approach. Table 1 indicates the significant computational savings that can be attained with the ETE approach to functional error estimation. Over most grid levels, approximately a 5x speedup is achieved. For this test case, this can be expected since five adjoint solves are replaced with one ETE solve. As long as the runtime for each individual adjoint solve and the ETE solve are of the same order magnitude, computational savings on the order of $n - 1$ adjoint solves can be expected with the ETE approach where n is the number of functionals of interest. With the ETE approach, the computational efficiency of functional error estimation is drastically improved without any loss of accuracy. These findings have important implications for the application of functional error estimation, especially for large engineering problems which typically require multiple functionals. Engineers should no longer be restricted to a limited number of solution functionals because of computational costs.

Table 1: Runtime (sec) – Adjoint and Error Transport Equation Solves

N_{cells}	Adjoint Solve	ETE Solve	Speedup
32	0.6755	0.2385	2.8326
64	1.3021	0.3638	3.5790
128	2.1773	0.5380	4.0469
256	3.7016	0.8409	4.4022
512	6.5797	1.3798	4.7686
1024	13.9797	2.7002	5.1772
2048	32.3345	5.5734	5.8015
4096	63.9561	11.7629	5.4371
8192	130.0427	23.7000	5.4870

B. Approximate Adjoints for Functional Error Estimation and Adaptation

The Krylov subspace method outlined in Section III is applied to 6 systematically refined, uniformly spaced grids ranging in size from 32 cells to 1024 cells. BiCG is tested using three bilinear form convergence tolerances: κ_{BF} = 1e-1, 1e-2, 1e-3. These tolerances are selected so that an assessment can be made of how accurate the bilinear form solve actually needs to be in order to obtain an accurate local error estimate, approximate adjoint variables with which to perform adaptation, and an accurate functional correction. SAI is used to precondition each bilinear form solve. The L_2-norm of the error in the ETE solve, $\|Ax - b\|_2$, the L_2-norm of the error in the adjoint solve, $\|A^T y - c\|_2$, and the relative error in the approximation of the adjoint correction,

$$\varepsilon_{BF} = \frac{|BF_{approx} - BF_{exact}|}{|BF_{exact}|}, \tag{51}$$

are monitored for each bilinear form tolerance. These errors are tabulated for each grid level in Table 2. All results in this section are presented for the supersonic case of the quasi-1D nozzle. Similar results were found for the subsonic case but are not shown here. Also, the functional of interest is selected as the integral of pressure along the nozzle. As can be seen in Table 2, the error in the ETE solve is small and does not vary greatly with the bilinear form tolerance. This means that, even for modest convergence of the bilinear form, a relatively accurate local error estimate can still be obtained. The approximate adjoint variables, on the other hand, are not nearly as accurate as the local error estimate. But, an argument can be made that the accuracy of the adjoint variables produced by the bilinear form solve are still accurate enough to be used to perform adaptation. To illustrate this, adaptation is performed on a 128 cell grid whose cells are initially uniformly spaced. Adaptation indicators are formed using exact adjoint variables as well as approximate adjoint variables obtained from the bilinear form solve where the bilinear form tolerance is selected as κ_{BF} = 1e-3. The adapted grids produced by each method are plotted together in Fig. 5 as the cell size versus the cell-center location. Although some variation can be seen downstream of the throat, the adapted grids in Fig. 5 are qualitatively very similar. The bilinear form approach is able to capture many of the main features of the adjoint adapted grid. In terms of functional-based adaptation, this application of the bilinear form result demonstrates that adjoint variables produced in an approximate sense can achieve the same end as adjoint variables produced from a traditional adjoint approach.

As for the approximation of the adjoint correction, the selected bilinear form tolerance affects how accurate the computed correction is for a given grid level. Decreasing the bilinear form tolerance increases the accuracy of the adjoint correction. As can be seen in Table 2, the approximation to the adjoint correction is more than accurate enough to be used to correct the functional. For instance, on the finest grid, the approximate adjoint correction obtained with a bilinear form tolerance of κ_{BF} = 1e-3 differs from the true value by less than 0.003%. To further demonstrate the accuracy of the bilinear form result, the approximate adjoint correction is applied to the functional, and the remaining error computed with the bilinear form approach is compared to the true remaining error. Fig. 6 illustrates that the bilinear form approach is able to achieve the same remaining error as a traditional adjoint approach. A bilinear form tolerance of κ_{BF} = 1e-3 performs the best and is able to capture the remaining error over a larger range of grids since it enforces a more accurate bilinear form solve. It should be noted that the bilinear form solve did not initially converge on the finest grid level with the non-zero pattern of the SAI preconditioner taken to be the same as the residual Jacobian matrix. In order to achieve convergence, the non-zero pattern of the SAI preconditioner is increased approximately 5x.

Table 2: Bilinear Form Errors – Adjoint Solve, Error Transport Equation Solve, and Adjoint Correction

	κ_{BF} = 1e-1			κ_{BF} = 1e-2			κ_{BF} = 1e-3		
N_{cells}	$\|Ax-b\|_2$	$\|A^Ty-c\|_2$	ε_{BF}	$\|Ax-b\|_2$	$\|A^Ty-c\|_2$	ε_{BF}	$\|Ax-b\|_2$	$\|A^Ty-c\|_2$	ε_{BF}
32	9.93e-02	3.26e-01	3.04e+00	4.64e-02	1.01e-01	1.21e-01	2.38e-03	9.51e-03	5.04e-05
64	1.58e-02	3.97e-01	3.33e-01	1.12e-03	5.79e-02	1.63e-03	6.97e-04	1.33e-02	9.70e-06
128	4.15e-03	2.64e-01	2.04e+00	8.31e-04	4.96e-02	9.72e-04	3.61e-04	1.49e-02	1.91e-05
256	3.41e-04	3.60e-02	5.71e-07	3.30e-04	1.42e-02	2.53e-07	4.35e-05	3.88e-03	2.50e-07
512	1.49e-04	3.31e-02	1.89e-05	3.82e-05	4.47e-03	1.89e-05	1.41e-05	3.84e-03	1.89e-05
[a]1024	2.76e-05	3.71e-02	1.37e-02	1.67e-06	2.70e-01	1.29e-04	1.68e-07	6.30e-02	2.43e-05

[a]Required better preconditioner

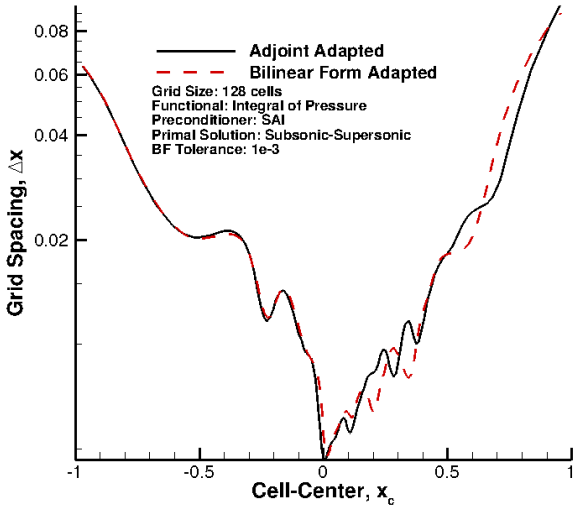


Figure 5: Comparison of Adapted Grids

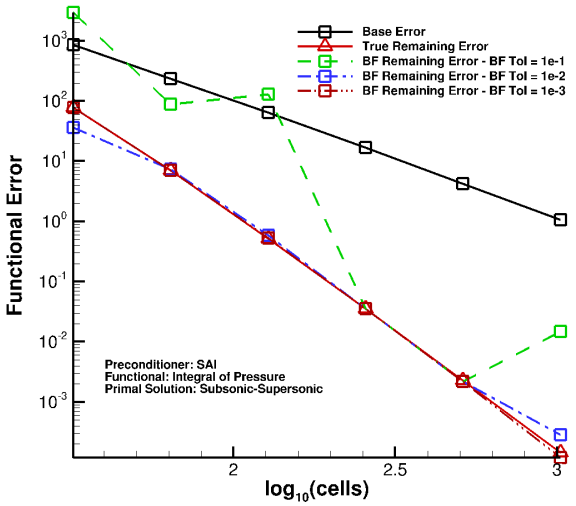


Figure 6: Bilinear Form Functional Error Convergence

Lastly, runtimes of the bilinear form solve are examined in order to assess the cost of decreasing the bilinear form tolerance. Since the bilinear form solve produces a local error estimate, adjoint variables for adaptation, and an adjoint correction, the runtime of the bilinear form solve is compared to the traditional approaches which achieve the same end, i.e. solving the adjoint problem and the ETE independently. In order to make a fair comparison, the adjoint problem and the ETE are solved directly using GMRES with the Krylov size fixed at 45. Runtimes for each grid level and each bilinear form tolerance are tabulated in Table 3. The bilinear form solve achieves tremendous speedups over the combined runtime of the adjoint and ETE solves with speedups ranging on average between 30x - 50x on the finest grids. For this problem, it does not appear that much performance is lost by decreasing the bilinear form tolerance. This feature could be problem dependent and should be examined further. Some care should be taken when comparing runtimes for bilinear form tolerances of κ_{BF} = 1e-2 and κ_{BF} = 1e-3 since faster runtimes are reported for the lower bilinear form tolerance. This is attributed to the small runtimes of the bilinear form solve at which the variability of computer performance can have a greater effect. It should also be noted that for the 5 finest grid levels the adjoint and ETE solves require a much better preconditioner. The non-zero pattern of the SAI preconditioner had to be increased by about 25x. Typically a pseudo-time-stepping method would be used to improve the conditioning of the system. Depending on the selection of the pseudo-time-step needed converge the solve, one could argue that the runtimes for the pseudo-time-stepping method might even be greater than what is quoted in Table 3. Therefore, even with a much cheaper preconditioner, the bilinear form approach can achieve the same ends as separate adjoint and ETE solves at a fraction of the cost.

Table 3: Runtime (sec) – Adjoint, Error Transport Equation, and Bilinear Form Solves

	Standard Approach			Bilinear Form Approach					
N_{cells}	Adjoint	ETE	Adjoint+ETE	κ_{BF} = 1e-1	Speedup	κ_{BF} = 1e-2	Speedup	κ_{BF} = 1e-3	Speedup
32	0.0877	0.0515	0.1392	0.0305	4.57	0.0390	3.57	0.0302	4.62
64	0.3581	0.9034	1.2615[a]	0.0376	33.57	0.0431	29.28	0.0438	28.78
128	1.5549	1.5054	3.0602[a]	0.0599	51.08	0.0830	36.85	0.0731	41.85
256	5.6034	5.8143	11.4177[a]	0.1479	77.21	0.1502	76.02	0.1512	75.49
512	13.7266	14.3274	28.0540[a]	0.5213	53.82	0.5443	51.54	0.5490	51.10
1024	53.2667[b]	35.0792	88.3459[a b]	1.7163[c]	51.48	2.0403[c]	43.30	1.9616[c]	45.04

[a]Required better preconditioner for both Adjoint and ETE Solves
[b]Did not converge
[c]Required better preconditioner

VI. Conclusions & Future Work

In this work, a new approach to functional error estimation is presented based upon the solution of an error transport equation (ETE) for the local solution error which is much more efficient than a conventional adjoint approach when multiple functional error estimates are required. The ETE approach is able to obtain the same functional error estimates as a standard adjoint approach in a more efficient manner. Computational speedups on the order of n adjoint solves are achieved where n is the number of functionals of interest. In addition, the proposed approach also provides local error estimates in the primal solution variables (e.g., density, velocity, and pressure).

Another technique, based upon Krylov subspace methods, is presented which takes advantage of the bilinear form of the adjoint correction. This method only requires the adjoint problem and the ETE be solved approximately, yet is still able to obtain an adjoint correction that is just as accurate as a standard adjoint solve. As a result, an accurate functional error estimate, a moderately accurate local error estimate, and approximate adjoint variables for adaptation are obtained in a more efficient manner. For the quasi-1D nozzle problem, the bilinear form approach achieved speedups on average between 30x and 50x compared to the total time required to solve the adjoint problem and the ETE separately. The approximate adjoint variables produced with this method are also shown to produce a similar adapted grid to one obtained with exact adjoint variables.

These new approaches could have important implications for the application of functional error estimation and adaptation to large engineering problems. Based on the results presented here, engineers might no longer be restricted to a limited number of functionals because of computational costs. Work still needs to be conducted to see how the speedups obtained with these approaches scale with problem size. Also, in a practical application, exact truncation error will not be known. Therefore, further testing will need to be performed which assesses the performance of these methods with estimated truncation error. In addition, the authors hope to investigate Krylov subspace recycling techniques [36–38] which could improve the performance of the bilinear form approach to functional error estimation when multiple functionals are required or when multiple bilinear forms must be solved on a sequence of adapted meshes.

Acknowledgments

This work was supported by the Collaborative Center for Aeronautical Sciences with Dr. John Benek of the Air Force Research Laboratory in Dayton, Ohio serving as the program manager.

Appendix

A. Appendix A - Condition Number of Dimensional and Non-dimensional Residual Jacobians

Table A.1: Residual Jacobian Matrix Condition Number

	Subsonic			Supersonic		
N_{cells}	Dimensional	Dimensional (Scaled)	Non-dimensional	Dimensional	Dimensional (Scaled)	Non-dimensional
32	6.28e+10	1.38e+06	2.25e+04	1.07e+13	1.96e+06	9.58e+03
64	3.45e+12	5.30e+06	1.06e+05	2.29e+13	4.22e+06	1.86e+04
128	2.57e+13	1.75e+07	3.47e+05	5.25e+13	8.69e+06	3.71e+04
256	7.78e+13	4.48e+07	8.64e+05	1.20e+14	1.76e+07	7.42e+04
512	1.69e+14	9.47e+07	1.81e+06	2.71e+14	3.54e+07	1.49e+05
1024	3.42e+14	1.91e+08	3.65e+06	6.04e+14	7.10e+07	2.97e+05

References

[1] Giles, M. B., Pierce, N., and Suli, E., "Progress in Adjoint Error Correction for Integral Functionals," *Comput. Visual Sci.*, Vol. 6, 2004, pp. 113 – 121.

[2] Giles, M. B. and Pierce, N. A., "Analytic Adjoint Solutions for the Quasi-One-Dimensional Euler Equations," *J. Fluid. Mech.*, Vol. 426, 2001, pp. 327 – 345.

[3] 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, Oct 2000, pp. 204 – 227.

[4] 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, Feb 2002, pp. 40 – 69.

[5] 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, May 2003, pp. 22 – 46.

[6] Park, M., Lee-Rausch, B., and Rumsey, C., "FUN3D and CFL3D Computations for the First High Lift Prediction Workshop," *49th AIAA Aerospace Sciences Meeting including the New Horizons Forum and Aerospace Exposition*, Jan 2011.

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

[8] Derlaga, J. M., *Application of Improved Truncation Error Estimation Techniques to Adjoint Based Error Estimation and Grid Adaptation*, Ph.D. thesis, Virginia Polytechnic Institute and State University, Blacksburg, VA, July 2015.

[9] Roy, C., "Strategies for Driving Mesh Adaptation in CFD (Invited)," *47th AIAA Aerospace Sciences Meeting including The New Horizons Forum and Aerospace Exposition*, Jan 2009.

[10] Oberkampf, W. L. and Roy, C. J., "Verification and Validation in Scientific Computing," 2010.

[11] Choudhary, A. and Roy, C., "Efficient Residual-Based Mesh Adaptation for 1D and 2D CFD Applications," *49th AIAA Aerospace Sciences Meeting including the New Horizons Forum and Aerospace Exposition*, Jan 2011.

[12] Phillips, T., Derlaga, J. M., Roy, C. J., and Borggaard, J., "Finite Volume Solution Reconstruction Methods For Truncation Error Estimation," *21st AIAA Computational Fluid Dynamics Conference*, June 2013.

[13] Phillips, T. and Roy, C., "Residual Methods for Discretization Error Estimation," *20th AIAA Computational Fluid Dynamics Conference*, June 2011.

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

[15] Golub, G. H., Stoll, M., and Wathen, A., "Approximation of the scattering amplitude and linear systems," *J-ELECTRON-TRANS-NUMER-ANAL*, Vol. 31, 2008, pp. 178 – 203.

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

[17] Ramm, A., "Symmetry properties of scattering amplitudes and applications to inverse problems," *Journal of mathematical analysis and applications*, Vol. 156, No. 2, 1991, pp. 333 – 340.

[18] Lanczos, C., "Solution of systems of linear equations by minimized iterations," *J. Res. Nat. Bur. Standards*, Vol. 49, No. 1, 1952, pp. 33 – 53.

[19] Fletcher, R., "Conjugate gradient methods for indefinite systems," *Numerical analysis*, Springer, 1976, pp. 73 – 89.

[20] Paige, C. C. and Saunders, M. A., "LSQR: An Algorithm for Sparse Linear Equations and Sparse Least Squares," *ACM Trans. Math. Softw.*, Vol. 8, No. 1, March 1982, pp. 43 – 71.

[21] Strakoš, Z. and Tichý, P., "On Efficient Numerical Approximation of the Bilinear Form c*A-1b." *SIAM J. Scientific Computing*, Vol. 33, No. 2, 2011, pp. 565 – 587.

[22] Golub, G. H. and Meurant, G., "Matrices, Moments and Quadrature," *Numerical Analysis* 1993*, Proceedings of the* 15*th Dundee Conference*, edited by D. F. Griffiths and G. A. Watson, Vol. 303 of *Pitman Research Notes in Mathematics*, ,, 1994, pp. 105 – 156.

[23] Jackson, C. W. and Roy, C. J., "A Multi-Mesh CFD Technique for Adaptive Mesh Solutions," *53rd AIAA Aerospace Sciences Meeting*, Jan 2015.

[24] Roe, P., "Approximate Riemann Solvers, Parameter Vectors, and Difference Schemes," *Journal of Computational Physics*, Vol. 135, No. 2, Aug 1997, pp. 250 – 258.

[25] van Leer, B., "Towards the ultimate conservative difference scheme. V. A second-order sequel to Godunovs method," *Journal of Computational Physics*, Vol. 32, No. 1, Jul 1979, pp. 101 – 136.

[26] 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, Sep 1976, pp. 87 – 110.

[27] 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, Nov 2004, pp. 1131 – 1155.

[28] Saad, Y. and Schultz, M. H., "GMRES: A Generalized Minimal Residual Algorithm for Solving Nonsymmetric Linear Systems," *SIAM Journal on Scientific and Statistical Computing*, Vol. 7, No. 3, Jul 1986, pp. 856 – 869.

[29] Saad, Y., "Iterative Methods for Sparse Linear Systems," Jan 2003.

[30] Benson, M. W. and Frederickson, P. O., "Iterative solution of large sparse linear systems arising in certain multidimensional approximation problems," *Utilitas Math*, Vol. 22, No. 127-140, 1982, pp. 154 – 155.

[31] Grote, M. J. and Huckle, T., "Parallel preconditioning with sparse approximate inverses," *SIAM Journal on Scientific Computing*, Vol. 18, No. 3, 1997, pp. 838 – 853.

[32] Dehnavi, M. M., Fernandez, D. M., Gaudiot, J., and Giannacopoulos, D. D., "Parallel sparse approximate inverse preconditioning on graphic processing units," *Parallel and Distributed Systems, IEEE Transactions on*, Vol. 24, No. 9, 2013, pp. 1852 – 1862.

[33] Huang, W. and Russell, R., *Adaptive Moving Mesh Methods*, Springer, 2011.

[34] Derlaga, J. M., Roy, C. J., and Borggaard, J., "Adjoint and Truncation Error Based Adaptation for 1D Finite Volume Schemes," *21st AIAA Computational Fluid Dynamics Conference*, Jun 2013.

[35] Derlaga, J. M., Phillips, T., Roy, C. J., and Borggaard, J., "Adjoint and Truncation Error Based Adaptation for Finite Volume Schemes with Error Estimates," *53rd AIAA Aerospace Sciences Meeting*, Jan 2015.

[36] Ahuja, K., *Recycling Bi-Lanczos Algorithms: BiCG, CGS, and BiCGSTAB*, Master's thesis, Virginia Polytechnic Institute and State University, 2009.

[37] Ahuja, K., *Recycling Krylov subspaces and preconditioners*, Ph.D. thesis, Virginia Polytechnic Institute and State University, 2011.

[38] Ahuja, K., de Sturler, E., Gugercin, S., and Chang, E. R., "Recycling BiCG with an Application to Model Reduction," *SIAM J. Sci. Comput.*, Vol. 34, No. 4, July 2012, pp. A1925 – A1949.