Application of the Pseudo-Time Accurate Formulation of the Adjoint to Output-Based Adaptive Mesh Refinement

Emmett Padway *
National Institute of Aerospace, Hampton, VA, 23606
Dimitri Mavriplis †
Department of Mechanical Engineering, University of Wyoming, Laramie, WY 82071

This paper presents application of the pseudo-time accurate formulation of the adjoint problem to error estimation and adaptive mesh refinement to allow output-based mesh refinement in partially converged simulations. Some previous papers have attempted to quantify the error in the output of interest by partial convergence of the non-linear problem, but here we try to obtain the discretization error despite a lack of convergence in the primal problem. This paper presents a more general derivation of these error estimation methods that have parallels in methods used for steady state adjoint error estimation. These derivations are applied to an inexact-quasi-newton algorithm. Finally we apply this to two non-convergent cases demonstrating small-scale and medium-scale unsteadiness respectively and show the results for the varying error estimation methods.

I. Introduction

Adaptive Mesh Refinement (AMR) has become a larger part of the expanding field of Computational Fluid Dynamics (CFD) as computers and algorithms have developed. As the field has matured and seen an explosion in computing power and developed stronger and more efficient methods, it has moved to attempting more complicated and expensive simulations. In order to make these simulations more computationally tractable the field has moved towards mesh refinement methods and they can now be found in many cutting edge research codes and in some industrial codes. The simplest mesh refinement technology is a preprocessing refinement, that takes the initial mesh and refines in an area the user believes will have important flow features. This suffers from being non-adaptive, however often these methods are used in conjunction with an adaptive method; an example of such a code is the Cart3D code which has an a priori refinement tool combined with an output-based error estimate to drive the adaptive refinement technology.[1,2] When looking at adaptive refinement, the two main branches of adaptive methodologies are feature-based and error/output-based refinement. Feature based refinement uses flow features to refine the mesh in areas of high flow gradients and coarsen in smooth areas. The refine package from NASA can calculate the Mach Hessian for the user from a flow field and can use that to drive the adaptation process on simplex meshes.[3] This method is popular due to ease and low cost of implementation. Many large scale codes use feature-based adaptation when error estimates are difficult or expensive to implement; one such example is refining based on q-criterion in exascale simulations of wind turbines.[4]

The error-based or output-based refinement methods use measures of error in the partial differential equation (PDE) discretization or the output of interest to drive the mesh adaption. In error based refinement, without an output of interest, the error is computed in one of two ways. The first way is a more general approach that can be used in both Finite Volume (FVM) and Finite Element Method (FEM) codes; the mesh is uniformly refined and the converged flow state is interpolated onto the refined mesh and the residual is calculated on the fine mesh, the residual from the fine mesh is then restricted back to the coarse mesh. The second method is often used in FEM codes, the residual operator for a higher order discretization than was used in the analysis

*Research Scholar, Member AIAA; email: emmett.padway@nianet.org
†Professor, AIAA Associate Fellow; email: mavripl@uwyo.edu

is calculated using the converged flow state. The magnitude of the residual in each element becomes the refinement criterion and drives the mesh refinement module.[5] The issue with error-based approaches that do not factor in the output-of-interest is that the mesh may refine in regions to better satisfy the governing equation that have no impact on the output of engineering interest. The output-based error estimates have seen the most development in recent years due to their usefulness in engineering applications. These began with the formulation by Becker and Rannacher[6] for FEM solvers, these use the adjoint to weight the residual from a higher order discretization to drive the refinement process, this is referred to as the dual-weighted residual (DWR) error estimate. These were then moved to FVM solvers by Venditti and Darmofal including functional corrections.[7] Mani and Mavriplis[8] then applied these error estimation techniques to unsteady aerostructural problems refining both the spatial and temporal mesh. As part of this project they also investigated the impact of partial convergence of the nonlinear problem on the accuracy of the output. Padway and Mavriplis introduce the idea of a partial convergence adjoint for sensitivity computation[9] and then later applied this to more wide-spread nonlinear solvers and design optimization.[10] This work is the sequel of Mani and Mavriplis and Padway and Mavriplis, in that it aims toward obtaining reliable error estimates through using the Padway and Mavriplis adjoint formulation without being hampered by the partial convergence of the nonlinear problem that Mani and Mavriplis investigated.

As stated above, both the adjoint and tangent systems require that the analysis problem be fully converged – or in the case of the unsteady adjoint, each implicit time step be fully converged – indicating that the governing equations have been satisfied to machine precision. However, as CFD has developed and matured and the field has attempted more difficult problems (higher-order formulations, blunt geometries, or time-accurate simulations) this constraint has become difficult or impractical to obey, and numerous design or mesh refinement cycles have been done using adjoint systems linearized about partially converged analysis solutions. This defect shows itself in less robust and more difficult to solve adjoint systems that are also sensitive to the specific state of the analysis problem about which they are linearized when the analysis simulation is terminated.[9,11] Krakos and Darmofal illustrate that for a nonconvergent case, the state about which the adjoint is linearized can notably affect the sensitivity calculations. The authors then show that by running the non-convergent steady-state case as a time-accurate case and applying the unsteady adjoint to the time-accurate case returns useful and accurate adjoint computed sensitivities for the time-averaged lift. The authors suggest that, for steady-state cases which can be solved by strong solvers but which may show physical unsteadiness, the time-accurate approach is in fact the proper analysis framework to use lest CFD practitioners risk obtaining unphysical and non-useful sensitivity vectors. However even for time-accurate formulations, Mishra et al.[12] have demonstrated growing sensitivity error during the adjoint reverse time-integration due to partial convergence of the analysis problem at each implicit time step, which is a common practice in applied CFD problems. That these are issues for sensitivities in partially converged simulations naturally leads to a similar concern for adjoint-based error estimates in the context of AMR. The concerns are that the adjoint would become very difficult to solve as was previously shown, and that for such cases our error estimates and mesh refinement become too reliant on the final state achieved in the simulation, thus harming error control and convergence criteria. This can also lead to obtaining different AMR patterns which can lead to refining the mesh in areas that may not contribute to the output of interest, and not refining in areas where the output of interest is affected. This can impact the accuracy of the analysis problem as the mesh adaptation proceeds.

This work comes from the work of Padway and Mavriplis[9,10] on the effects of incomplete analysis problem convergence in steady state problems. In this paper we develop the dual-weighted constraint method which applies the adjoints and constraints developed in those two works to error estimation and adaptive mesh refinement.

II. Background and In-House Solver

II.A. Governing Equations

We developed an in-house flow solver to solve the steady-state Euler equations on unstructured meshes. The steady-state compressible Euler equations (which may also be referred to as the primal/analysis problem) can be written as follows.

$$\nabla \cdot F(u(D)) = 0 \tag{1}$$

Which can also be written as:

$$R(u(D), D) = 0 \tag{2}$$

where u is the conservative variable vector, D is the design variable vector, and $F(u)$ is the conservative variable flux.

II.B. Spatial Discretization

The residual about the closed control volume is given as:

$$R = \int_{dB} [F(u(D))] \cdot n(x(D)), \mathrm{dB} = \sum_{i=1}^{n_{edge}} F_{e_i}^{\perp}(u(D), n_{e_i}(x(D)))B_{e_i}(x(D)) \tag{3}$$

This equation shows the discretized residual operator which must be equal to 0 in a converged flow; the constraint that $R = 0$ is used for the derivation of the adjoint problem. In the discretized form of the residual operator, x represents the mesh point coordinates, F is the numerical flux across the element boundary, B is the element boundary, and n is the edge normal on the element boundary. The solver used in this work is a steady-state finite-volume cell-centered Euler solver with second-order spatial accuracy implemented for triangular elements. Second-order accuracy is implemented through weighted least squares gradient reconstruction.[13] The solver has three different flux calculations implemented and linearized, these are the Lax-Friedrichs,[14] Roe,[15] and van Leer[16] schemes. In this work, only the Lax-Friedrichs and Van-Leer schemes are used. In order to slope limit the solution reconstruction near shock discontinuities, a modified Venkatakrishnan limiter is used; Venkatakrishnan's limiter (VK limiter)[17] is modified in a few important ways.

1. All max and min functions use smooth max and mins to allow differentiability
2. All switches in the code dependent on the flow state are done in a smooth and blended manner using a sin shut-off function to aid differentiability
3. The limiter is augmented with a stagnation point fix[18] that turns off the limiter entirely if the local Mach number is below a certain value (again done in a smooth manner)
4. Finally, the limiter contains a realizability check[19] that, if violated (pressure, energy, or density are reconstructed to below 5% of the cell center value), the cell gradient is set to 0. Then the cell (as a whole) becomes first order, this is used rather than employing face-based limiting.

This limiter in algorithm (1) provides better convergence properties than the standard VK Limiter, and is used in this work. Every step in this limiter is differentiable due to the use of smooth max and min functions and the smoothly blending shut-off (SSO) function outlined in equation (4).[20] This makes the linearization of the limiters possible and much easier to code, as we avoid multiple branching statements.

$$SSO(x, x_s, x_e) = \begin{cases} 0 & if x < x_s \\ \frac{1}{2}\left(sin\left(\frac{\pi}{2}\frac{2x-(x_e+x_e)}{x_e-x_s}\right)+1\right) & if xs < x < x_e \\ 1 & if x > x_e \end{cases} \tag{4}$$

Algorithm 1 Augmented Venkatakrishnan Limiter

procedure VK LIMITER
  $M_1 = .80, M_2 = .85, \epsilon_{lim} = 1e-5, \epsilon_{TFO_s} = -.95, \epsilon_{TFO_e} = -.9$
  $\epsilon = \sqrt{\kappa(r_1)}^3$, where r_1 is the radius of the circumscribed circle of the triangular cell
  $M_{max} = \bar{M}$
  for $i = 1, ..., fields$ **do**
    $\Phi_i = 1$
    $\delta u_i^{min} = \infty$, $\delta u_i^{max} = 0$
    for $j = 1, ..., neighbors$ **do**
      $\delta u_i^{min} = \min(\delta u_i^{min}, \bar{u_i} - \bar{u_i^j})$
      $\delta u_i^{max} = \max(\delta u_i^{max}, \bar{u_i} - u_i^j)$
      $M_{max} = \max(M_{max}, M_i)$
    $\sigma_{Mach} = 1 - SSO(M_{max}, M_1, M_2)$
    for $j = 1, ..., neighbors$ **do**
      $\Delta_- = \nabla \bar{u_i} \cdot \vec{r_j}$
      $s = SSO(\Delta_-, 0, \epsilon_{lim})$
      $\Delta_+ = (1-s)(u_i^{min} - \bar{u}_i) + s(u_i^{max} - \bar{u}_i)$
      $\phi = \frac{\Delta_+^2 + \epsilon^2 + 2\Delta_+\Delta_-}{\Delta_+^2 + \epsilon^2 + 2\Delta_+\Delta_- + 2\Delta_-^2}$
      Stagnation point fix
      $\phi = \sigma_{Mach} + (1 - \sigma_{Mach})\phi$
      $\Phi_i = \min(\Phi_i, \phi)$
  Begin realizability check
  for $j = 1, ..., neighbors$ **do**
    $u_{rc_j} = u_i + \Phi \nabla \bar{u_i} \cdot \vec{r_j}$
    $\delta_\rho = \rho_{rc_j} - \bar{\rho}$
    $\delta_P = P_{rc_j} - \bar{P}$
    $\delta_E = E_{rc_j} - \bar{E}$
    $s_e = SSO(\delta_e, \epsilon_{TFO_s}, \epsilon_{TFO_e})$
    $s_p = SSO(\delta_p, \epsilon_{TFO_s}, \epsilon_{TFO_e})$
    $s_\rho = SSO(\delta_\rho, \epsilon_{TFO_s}, \epsilon_{TFO_e})$
    $s = \min(s_\rho, s_e, s_p)$
    $\Phi = s\Phi$

II.C. Steady-State Solver

The solver technology for this code uses either explicit time stepping through pseudo time using a forward Euler time discretization, or a low storage five stage Runge-Kutta scheme, or a quasi-Newton method. The quasi-Newton method is the only method used in this work and is implemented using pseudo-transient continuation (PTC) with a first order backward difference (BDF1) pseudo temporal discretization scheme. For Newton's method the time-stepping procedure is written as:

$$u^k = u^{k-1} + \Delta u \tag{5}$$

where we compute Δu by solving the following system of linear equations.

$$[P]\,\Delta u = -R(u) \tag{6}$$

We can substitute our expression for Δu into the time-stepping equation (5) to obtain our final form of this equation.

$$u^k = u^{k-1} - [P_{k-1}]^{-1}\,R \tag{7}$$

Here $[P_{k-1}]$ is the Jacobian of the first-order accurate spatial discretization augmented with a diagonal term to ensure that it is diagonally dominant, shown in equation (8).

$$[P_{k-1}] = \left[\frac{\partial R}{\partial u^{k-1}}\right]_1 + \frac{vol}{\Delta t CFL} \tag{8}$$

Please note the subscript on the Jacobian above denotes that it is a 1st order Jacobian; in this work the subscripts of 1 and 2 will denote Jacobians of the first and second order accurate spatial discretizations respectively. The equation for the local explicit time step limit Δt is given as:

$$\Delta t_i = \frac{r_i}{\sqrt{(u^2+v^2)}+c} \tag{9}$$

where r_i is the circumference of the inscribed circle for mesh cell i, u and v are the horizontal and vertical velocity components respectively, and c is the speed of sound in the triangular element.
Furthermore, the CFL is scaled either with a simple ramping coefficient (β) and cap criterion:

$$CFL^{k+1} = min(\beta \cdot CFL, CFL_{max}) \tag{10}$$

or with a line-search and CFL controller,[21,22] which seeks to minimize the L_2 norm of the pseudo-temporal residual, defined as:

$$R_t(u+\alpha\Delta u) = \frac{vol}{\Delta t}\alpha\Delta u^n + R(u+\alpha\Delta u^n) \tag{11}$$

When the pseudo-temporal residual decreases, we consider this to be a satisfactory value for α and we change the CFL accordingly. The pseudocode explains the actual process for the line-search and CFL controller. The parameters $iter_{max}$, c, α_{l_1}, α_{l_2}, β_1, and β_2 are all user defined input values, defaulted to 30, .9, .1, .75, .1, and 1.5 respectively. One benefit of this combined line-search and CFL controller is that we do not have to differentiate it, and can simply store the values of CFL and α and use these fixed values in the forward and reverse linearizations and still obtain machine-level correspondence when comparing those sensitivity values to those of the complex-step differentiated solution process. This is possible because the combined CFL controller and line-search represents a piece-wise constant function with a zero derivative.

Algorithm 2 CFL controller

procedure LINE SEARCH AND CFL CONTROLLER
  $r_{t0} = \|R_t(u + \Delta u)\|_2$
  $r_{s0} = \|R(u)\|_2$
  $r_{s1} = \|R(u + \Delta u)\|_2$
  if $r_{s0} < r_{s1}$ **then**
    for $k = 1, ..., iter_{max}$ **do**
      $\alpha = c\alpha$
      $r_{t1} = \|R_t(u + \alpha\Delta u)\|_2$
      if $r_{t1} < r_{t0}$) **then** exit
  if $\alpha < \alpha_{l_1}$ **then**
    $\alpha = 0$
    $CFL = \beta_1 CFL$
  else if $\alpha_{l_1} < \alpha < \alpha_{l_2}$ **then**
    $CFL = CFL$
  else if $\alpha < \alpha_{l_2}$ **then**
    $CFL = min(\beta_2 CFL, CFL_{max})$

In order to solve the linear system we use a point-implicit Jacobi or point-implicit Gauss-Seidel solver. This is done by lagging the off-diagonal components, with the right hand side being the linear residual of the original system. In this work the residual is the second-order accurate spatial residual operator, and $[P_{k-1}]$ is based off the first-order accurate residual operator outlined in equation (8). Equation (6) is then solved iteratively as:

$$[D]\,\Delta(\Delta u)^l = -R(u) - [P_{k-1}]\,\Delta u^l \tag{12}$$

where the matrix $[D]$ is the element block diagonal entry in the Jacobian matrix. Updates are then implemented as below, where *omega* is an underrelaxation parameter.

$$\Delta u^{l+1} = \Delta u^l + \omega(\Delta(\Delta u))^l \tag{13}$$

These linear solvers can also be applied as smoothers either to a BiCGStab or a GMRES linear solver. The flexible GMRES linear solver is the more commonly used one in this Newton-Krylov nonlinear solver, and the algorithm below shows its implementation, where the operator M^{-1} is the preconditioning matrix using the block Jacobi or block Gauss-Seidel relaxation schemes outlined above. The FGMRES solver outlined in algorithm (3)[23] is implemented to solve the stiff steady state tangent and adjoint systems as well.

Algorithm 3 Flexible Restarted GMRES

procedure FLEXIBLE GMRES
  for $k = 1, ..., ncycles$ **do**
    $r_0 = b - Ax_0, \beta = \|r_0\|, v_1 = r_0/\beta$
    for $j = 1, ..., m$ **do**
      $z_j = M^{-1}v_j$
      $v_{j+1} = Az_j$
      for $i = 1, ..., j$ **do**
        $h_{i,j} = (v_{j+1}, v_i)$
        $v_{j+1} = v_{j+1} - h_{i,j}v_i$
      $h_{j+1,j} = \|v_{j+1}\|, v_{j+1} = v_{j+1}/h_{j+1,j}$
      Define $Z_m = [z_1, ..., z_m], \bar{H}_m = [h_{i,j}]_{1<i<j+1,1<j<m}$
    Solve least squares problem for y_m
    $x_0 = x_0 + Z_m y_m$

III. A Review of the Dual-Weighted Residual

A straightforward approach to error estimation would be to get an estimate of a hypothetical fine mesh solution (u_h) using the coarse mesh solution (u_H), and calculate error estimates based on the error in the solution. However, for engineering purposes, the practitioner's interest lies not in the actual values of the solution but instead in the values of the outputs of engineering interest (lift, drag, moment, etc.). For this purpose the dual-weighted residual method was developed as it provides estimates of error in the output of interest and allows for automatic error control of simulations.[2]

The method begins by approximating the output of interest (L_h) evaluated on a hypothetical fine (embedded) mesh (h) using the fine mesh solution (u_h) and performing a Taylor series expansion about it to calculate the output of interest based off the solution on the coarse (working) mesh (H) interpolated onto the fine mesh ($\tilde{u}_h$).

$$L_h(u_h) \approx L_h(\tilde{u}_h) + \frac{\partial L_h(\tilde{u}_h)}{\partial u_h}(u_h - \tilde{u}_h) \tag{14}$$

This would indicate that to get a good estimate of the fine mesh output of interest it is necessary to solve both the fine and coarse mesh primal problems, and take the difference between the coarse mesh solution prolongated onto the fine mesh and the fine mesh solution to multiply the derivative of the output of interest. This of course defeats the purpose of mesh refinement as it would require solution of the nonlinear problem on the more expensive fine mesh. Instead a Taylor series expansion about the fine grid is used to move from the coarse grid onto the fine grid, which will have a nonzero residual for the coarse grid state interpolated onto the fine grid.

$$R_h(u_h) = 0 \approx R_h(\tilde{u}_h) + \frac{\partial R_h(\tilde{u}_h)}{\partial u_h}(u_h - \tilde{u}_h) \tag{15}$$

Combining the two equations returns the following.

$$L_h(u_h) \approx L_h(\tilde{u}_h) - \frac{\partial L_h(\tilde{u}_h)}{\partial u_h}\left[\frac{\partial R_h(\tilde{u}_h)}{\partial u_h}\right]^{-1} R_h(\tilde{u}_h) \tag{16}$$

Defining the adjoint on the fine mesh, ψ_h, as:

$$\left[\frac{\partial R_h(\tilde{u}_h)}{\partial u_h}\right]^T \psi_h = -\frac{\partial L_h(\tilde{u}_h)}{\partial u_h} \tag{17}$$

and substituting in ψ_h into equation 16 returns the below.

$$L_h(u_h) \approx L_h(\tilde{u}_h) + \psi_h^T R_h(\tilde{u}_h) \tag{18}$$

The error estimate is calculated as follows:

$$\eta_H = |(\tilde{\psi}_h)^T R_h(\tilde{u}_h)| \tag{19}$$

and a cell parent element error estimate vector can be calculated by looping over all the child elements of the larger parent cell as follows.

$$\eta_H = |\sum_{j \in V_{parent}} \psi_h^T R_h(\tilde{u}_h)_j| \tag{20}$$

The second technique is the dual weighted residual with computable correction method. To compute an adjoint-based computable correction it is necessary to split the adjoint into two terms, the approximate adjoint and the error in the approximation. To obtain the computable correction of the objective function the approximate adjoint is dotted with the residual, whereas to obtain the remaining error estimate the error in the adjoint is dotted with the residual.

$$L_h(u_h) \approx L_h(\tilde{u}_h) - (\tilde{\psi}_h)^T R_h(\tilde{u}_h) - (\psi_h - \tilde{\psi}_h)^T R_h(\tilde{u}_h) \tag{21}$$

The first term is the adjoint correction to the functional, and the second is the remaining error which is used to drive the mesh adaptation. Having shown the general derivations we will now discuss the two different implementation options for these error estimation methods.

III.A. Embedded Mesh

The first implementation option is referred to as the embedded mesh method and is typically used in Finite Volume discretizations; it entails creating a subdivided fine/embedded mesh from the coarse mesh and interpolating the coarse mesh values onto the embedded mesh. Adjoint values can be computed on this mesh or interpolated onto it. In order to do this, we must first establish our interpolation mechanics. The first step of the interpolation is computing a bilinear approximation in each cell where f is the value of the variable of interest in this reconstruction – in this work this is one of the conservative or adjoint variables.

$$f(x, y) = c_0 + c_1 x + c_2 y + c_3 xy \tag{22}$$

This is done by looping over the cells and reconstructing the variables to each of the vertices and averaging the reconstructed variable and gradient values at each vertex from the cells that it forms. Each cell then sets up an overconstrained linear least squares problem for each individual variable. Having averaged the variable values and gradients to compute values at each node of the mesh, the values and gradients at each of the three vertices of the triangle are used to create a system with nine constraints (f, f_x, f_y at each of the three vertices) and, as can be seen in the approximation above, only four unknowns (c_0, c_1, c_2, c_3). For the biquadratic approximation, the variable constraints are populated using the bilinear interpolation in equation 22, whereas the gradient constraints are populated using the same vertex averaging used in the bilinear interpolation.

$$f(x, y) = c_0 + c_1 x + c_2 y + c_3 xy + c_4 x^2 + c_5 y^2 + c_6 x^2 y + c_7 x y^2 \tag{23}$$

This biquadratic approximation neglects the double quadratic term as it has been shown to add undesired oscillations into the solution and would also provide the ninth unknown which would entail no longer being an overconstrained system. The biquadratic approximation is then used to evaluate the variable values in each of the child cells for the embedded mesh. For the dual weighted residual, the error due to the spatial discretization is written as follows, where the biquadratic interpolation of the adjoint ($\tilde{\psi}_{BQ}$) is used as a proxy for the adjoint on the fine mesh ($\tilde{\psi}_h$), rather than actually solving the adjoint problem on the fine mesh.

$$\eta_H = |(\psi_{BQ})^T R_h(u_{BQ})| \tag{24}$$

A parent element error estimate vector can be calculated by looping over all the child elements of the larger parent cell as follows.

$$\eta_H = |\sum_{j \in V_{parent}} \psi_{BQ}^T R_h(\tilde{u}_{BQ})_j| \tag{25}$$

For the dual weighted residual with computable correction method, as shown above, it is necessary to split the adjoint into two terms, the approximate adjoint and the error in the approximation. To obtain the computable correction of the objective function the approximate adjoint (computed by the bilinear interpolation) is dotted with the residual. To obtain the remaining error estimate, the error in the adjoint (calculated by subtracting the bilinear interpolation from the biquadratic interpolation) is dotted with the residual. Please note that in this work the biquadratic interpolation (which is an estimate of ψ_h) is used for the adjoint correction even though the adjoint correction term actually requires $\tilde{\psi}_h$.

$$L_h(u_h) \approx L_h(\tilde{u}_{BQ}) - (\psi_{BQ})^T R_h(\tilde{u}_{BQ}) - (\psi_{BQ} - \psi_{BL})^T R_h(\tilde{u}_{BQ}) \tag{26}$$

Others have dealt with the issue of solving the adjoint on the fine mesh by solving the adjoint on the coarse mesh, using this as an initial guess for the linear solver of the fine mesh adjoint problem, and then running a few fine grid smoothing passes to get a better approximation of the fine mesh adjoint at a low cost. That implementation uses the coarse adjoint prolongated onto the fine mesh as the approximate adjoint and the smoothed adjoint on the fine mesh as the the exact adjoint.[5]

III.B. Virtual Mesh Method

The virtual mesh method allows for an error estimate and refinement without needing to actually create the embedded mesh. The virtual mesh uses differing order reconstructions of the conservative variables to

produce estimates of the conservative variable and gradient values at the centroids of the virtual child cells and then uses those values to compute the residual restricted back to the coarse mesh.
The first step is computing a bilinear approximation of the conservative variables in each cell using the node aggregation and interpolation step shown previously. The biquadratic approximation is then used to evaluate the conservative variable values and gradients at each of the 3 exterior virtual children of the cell in question. These values and gradients are then used to reconstruct to the cell edge, and the flux across the child face (half of the original face) is computed.

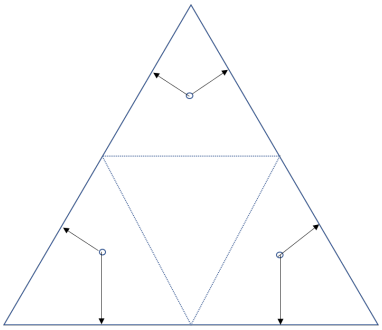


Figure 1. Virtual Mesh Residual Diagram

As can be seen in Figure 1, no fluxes are computed across the interior edges into the central child element of the virtual mesh as these fluxes will cancel out when they are restricted back to the working mesh. The exterior edge fluxes will be added to the residual vector of the working mesh and this vector will then multiply the working mesh adjoint vector (this treats the working mesh adjoint as the fine mesh adjoint) to obtain a cell-wise error estimate. If one instead uses the biquadratic approximation to extrapolate to the edges, this could be viewed as calculating the fine mesh residual with a third order finite volume method with a non-k-exact least-squares reconstruction and without curved geometry and then weighting that higher accuracy residual with the working mesh adjoint. The virtual mesh method is only used for the error estimation without functional correction. The cellwise error estimate is expressed below:

$$\eta_H = |(\psi_H)^T R_H^h(\tilde{u}_h)| \tag{27}$$

where ψ_H is the coarse mesh adjoint, R_H^h is the fine mesh residual restricted back to the coarse mesh through use of the virtual mesh subroutines, and $\tilde{u}_h$ is the coarse mesh conservative variables interpolated onto the virtual fine mesh using the interpolation routines previously outlined.

Having completed the review of the dual-weighted residual approach for the virtual mesh method, the embedded mesh error estimate and the embedded mesh error estimate with functional correction, we move on to their analogues in the dual-weighted constraint formulation.

IV. Development of the Pseudo-Time Accurate Dual-Weighted Constraint

As stated previously, the pseudo-time accurate adjoint method is drawn from the derivation of the unsteady adjoint; the algorithm works backwards through pseudo-time to obtain the pseudo-time accurate adjoint solution. This formulation seeks to find an estimate of how much of the error in the quantity of interest is due to the spatial discretization for these unconverged and oscillatory flows. In this derivation the output of interest is a pseudo-time averaged functional, averaged over the last m steps for a simulation that runs through n pseudo-time steps. This gives an output of interest L as:

$$L = L(u_h^n, u_h^{n-1}, ..., u_h^{n-m}) \tag{28}$$

where u_h^n is the conservative variable vector at the final time step n on the fine mesh. A Taylor series expansion about the fine mesh objective is then performed, where $\tilde{u}_h^k$ is the coarse (working) mesh (H)

solution at iteration k interpolated onto the embedded mesh:

$$\begin{aligned} L = L(u_h^n, u_h^{n-1}, ..., u_h^{n-m}) \approx L(\tilde{u}_h^n, \tilde{u}_h^{n-1}, ..., \tilde{u}_h^{n-m}) + \frac{\partial L}{\partial \tilde{u}_h^n}(u_h^n - \tilde{u}_h^n) \\ + \frac{\partial L}{\partial \tilde{u}_h^{n-1}}(u_h^{n-1} - \tilde{u}_h^{n-1}) \\ + ... \\ + \frac{\partial L}{\partial \tilde{u}_h^{n-m}}(u_h^{n-m} - \tilde{u}_h^{n-m}) \end{aligned} \tag{29}$$

Unlike the typical adjoint and dual weighted residual method where the constraint that the steady state residual is zero is used, instead, a constraint is selected based on the pseudo-time evolution of the solution, for which, the k^{th} constraint will be referred to as G^k. Because of the algorithm implementation it is clear that the constraint is dependent only on the old time-step and the new time-step, expressed as follows.

$$G^k = G^k(u^k, u^{k-1}) = 0 \tag{30}$$

Linearizing about the states gives:

$$G^k(u_h^k, u_h^{k-1}) = 0 \approx G^k(\tilde{u}_h^k, \tilde{u}_h^{k-1}) + \frac{\partial G^k}{\partial \tilde{u}_h^k}(u_h^k - \tilde{u}_h^k) + \frac{\partial G^{k-1}}{\partial \tilde{u}_h^{k-1}}(u_h^{k-1} - \tilde{u}_h^{k-1}) \tag{31}$$

by isolating for the error in the conservative variable reconstruction the below equation is obtained.

$$(u_h^{k-1} - \tilde{u}_h^{k-1}) = -\left[\frac{\partial G^{k-1}}{\partial \tilde{u}_h^{k-1}}\right]^{-1}\left[G^k(\tilde{u}_h^k, \tilde{u}_h^{k-1}) + \frac{\partial G^k}{\partial \tilde{u}_h^k}(u_h^k - \tilde{u}_h^k)\right] \tag{32}$$

Substituting in the error in the conservative variables returns the below expression.

$$\begin{aligned} &L(u_h^n, u_h^{n-1}, ..., u_h^{n-m}) = L(\tilde{u}_h^n, \tilde{u}_h^{n-1}, ..., \tilde{u}_h^{n-m}) \\ &- \left(\frac{\partial L}{\partial u}\right)_{\tilde{u}_h^n}\left[\frac{\partial G^n}{\partial u_h^n}\right]^{-1}_{u_h^n}\left[G(\tilde{u}_h^n, \tilde{u}_h^{n-1}) + \left(\frac{\partial G^n}{\partial u_h^{n-1}}\right)_{u_h^{n-1}}(u_h^{n-1} - \tilde{u}_h^{n-1})\right] \\ &+ \left(\frac{\partial L}{\partial u}\right)_{\tilde{u}_h^{n-1}}(u_h^{n-1} - \tilde{u}_h^{n-1}) \\ &+ ... \\ &+ \left(\frac{\partial L}{\partial u}\right)_{\tilde{u}_h^{n-m}}(u_h^{n-m} - \tilde{u}_h^{n-m}) \end{aligned} \tag{33}$$

Defining the equation for the final state adjoint returns the equation below:

$$\Lambda^{nT} = -\left(\frac{\partial L}{\partial u}\right)_{\tilde{u}_h^n}\left[\frac{\partial G^n}{\partial u_h^n}\right]^{-1}_{u_h^n} \tag{34}$$

It is then necessary to substitute and separate terms to get the following expression.

$$\begin{aligned} L(u_h^n, u_h^{n-1}, ..., u_h^{n-m}) = L(\tilde{u}_h^n, \tilde{u}_h^{n-1}, ..., \tilde{u}_h^{n-m}) \\ + \Lambda^{nT}\left[G(\tilde{u}_h^n, \tilde{u}_h^{n-1})\right] \\ + \left[\left(\frac{\partial L}{\partial u}\right)_{\tilde{u}_h^{n-1}} + \Lambda^{nT}\left(\frac{\partial G^n}{\partial u_h^{n-1}}\right)_{u_h^{n-1}}\right](u_h^{n-1} - \tilde{u}_h^{n-1}) \\ + ... \\ + \left(\frac{\partial L}{\partial u}\right)_{\tilde{u}_h^{n-m}}(u_h^{n-m} - \tilde{u}_h^{n-m}) \end{aligned} \tag{35}$$

Looking at the above equation, it is necessary to define the following adjoint recurrence relation.

$$\Lambda^{n-1T} = -\left[\left(\frac{\partial L}{\partial u}\right)_{\tilde{u}_h^{n-1}} + \Lambda^{nT}\left(\frac{\partial G^n}{\partial u_h^{n-1}}\right)_{u_h^{n-1}}\right]\left[\frac{\partial G^{n-1}}{\partial u_h^{n-1}}\right]^{-1}_{u_h^{n-1}} \tag{36}$$

Substituting equation 36 into equation 35 returns the expression below.

$$\begin{aligned} L(u_h^n, u_h^{n-1}, ..., u_h^{n-m}) = {} & L(\tilde{u}_h^n, \tilde{u}_h^{n-1}, ..., \tilde{u}_h^{n-m}) \\ & + \Lambda^{nT}\left[G(\tilde{u}_h^n, \tilde{u}_h^{n-1})\right] \\ & + \Lambda^{n-1T}\left[G(\tilde{u}_h^{n-1}, \tilde{u}_h^{n-2})\right] \\ & + \left[\left(\frac{\partial L}{\partial u}\right)_{\tilde{u}_h^{n-2}} + \Lambda^{n-1T}\left(\frac{\partial G^{n-1}}{\partial u_h^{n-2}}\right)_{u_h^{n-2}}\right](u_h^{n-2} - \tilde{u}_h^{n-2}) \\ & + ... \\ & + \left(\frac{\partial L}{\partial u}\right)_{\tilde{u}_h^{n-m}}(u_h^{n-m} - \tilde{u}_h^{n-m}) \end{aligned} \tag{37}$$

Working this recurrence relation m times back in pseudo-time out through the averaging window returns:

$$\begin{aligned} L(u_h^n, u_h^{n-1}, ..., u_h^{n-m}) = {} & L(\tilde{u}_h^n, \tilde{u}_h^{n-1}, ..., \tilde{u}_h^{n-m}) \\ & + \Lambda^{nT}\left[G(\tilde{u}_h^n, \tilde{u}_h^{n-1})\right] \\ & + \Lambda^{n-1T}\left[G(\tilde{u}_h^{n-1}, \tilde{u}_h^{n-2})\right] \\ & + \Lambda^{n-2T}\left[G(\tilde{u}_h^{n-2}, \tilde{u}_h^{n-3})\right] \\ & + ... \\ & + \Lambda^{n-mT}\left[G(\tilde{u}_h^{n-m}, \tilde{u}_h^{n-(m+1)})\right] \\ & + \Lambda^{n-mT}\left(\frac{\partial G^{n-m}}{\partial u_h^{n-(m+1)}}\right)(u_h^{n-(m+1)} - \tilde{u}_h^{n-(m+1)}) \end{aligned} \tag{38}$$

When the backwards-in-pseudo-time integration exits the averaging window the recurrence relation loses the source term from the linearization of the output of interest and equation 36 simplifies to the equation below.

$$\Lambda^{k-1T} = -\left[\Lambda^{kT}\left(\frac{\partial G^k}{\partial u_h^{k-1}}\right)_{u_h^{k-1}}\right]\left[\frac{\partial G^{k-1}}{\partial u_h^{k-1}}\right]^{-1}_{u_h^{k-1}} \tag{39}$$

This returns a final expression for the averaged output of interest on the fine mesh as shown below.

$$\begin{aligned} L(u_h^n, u_h^{n-1}, ..., u_h^{n-m}) = {} & L(\tilde{u}_h^n, \tilde{u}_h^{n-1}, ..., \tilde{u}_h^{n-m}) \\ & + \Lambda^{nT}\left[G(\tilde{u}_h^n, \tilde{u}_h^{n-1})\right] \\ & + \Lambda^{n-1T}\left[G(\tilde{u}_h^{n-1}, \tilde{u}_h^{n-2})\right] \\ & + ... \\ & + \Lambda^{1T}\left[G(\tilde{u}_h^1, \tilde{u}_h^0)\right] \\ & + \Lambda^{1T}\left[\frac{\partial G^1}{\partial u_h^0}\right](u_h^0 - \tilde{u}_h^0) \end{aligned} \tag{40}$$

Grouping the output of interest terms returns:

$$\begin{aligned} \Delta L(\tilde{u}_h^n, \tilde{u}_h^{n-1}, ..., \tilde{u}_h^{n-m}) \approx {} & \Lambda^{nT}\left[G(\tilde{u}_h^n, \tilde{u}_h^{n-1})\right] \\ & + \Lambda^{n-1T}\left[G(\tilde{u}_h^{n-1}, \tilde{u}_h^{n-2})\right] \\ & + ... \\ & + \Lambda^{1T}\left[G(\tilde{u}_h^1, \tilde{u}_h^0)\right] \\ & + \Lambda^{1T}\left[\frac{\partial G^1}{\partial u_h^0}\right](u_h^0 - \tilde{u}_h^0) \end{aligned} \tag{41}$$

This could be viewed as the error in the output of interest due to the spatial discretization for an unconverged fixed point iteration with a correction included to compute the impact of the initial condition on the fixed point. For a uniform flow there is no correction introduced from the initial condition, but for a restart file the error in the solution reconstruction would lead to a nonzero value due to the nonuniform initial flow. In an AMR case, this could theoretically allow for calculating the error introduced by a mesh adaptation cycle, and then going all the way back through the adaptation chain back to the first mesh and freestream flow. While this may be theoretically possible, this is untenable in practice. This does however open up the door to only integrating partially back in iteration-space as was done in previous work.[9,10] This error could be modeled through the subtraction of the bilinear reconstruction from the biquadratic reconstruction and this could perhaps be sufficient as an correction to the error estimate. One item of importance is that the above derivation used pseudo-temporal adjoint vectors calculated on the fine mesh in addition to the fixed-point iterations calculated on the fine mesh; this is impractical. The cost of linearizing the fixed -iterations on the fine mesh is significant and so we linearize them on the coarse mesh, calculate the pseudo-temporal adjoint and prolongate it onto the fine mesh to be multiplied by the residual to create the error estimate.

IV.A. Error Estimation for Newton's Method

For this section the analogues of the error estimation methods shown in the section III and then developed in this section are shown for Newton's method using the approximate adjoint linearization with the inverse identity.[10] To begin, it is necessary to refer once again to equation (7):

$$u^k = u^{k-1} - [P_{k-1}]^{-1} R \tag{42}$$

with a corresponding constraint/shifted fixed-point iteration:

$$G^k = u^k - u^{k-1} + [P_{k-1}]^{-1} R \tag{43}$$

Taking the derivative of the shifted fixed-point iteration at each pseudo-time-step with respect to both states returns the equation below.

$$\begin{aligned} \frac{\partial G^k}{\partial u_h^k} &= I \\ \frac{\partial G^k}{\partial u_h^{k-1}} &= -I + [P_{k-1}]^{-1} \left[\frac{\partial R(u_h^{k-1})}{\partial u_h^{k-1}} \right]_2 + \frac{\partial [P_{k-1}]^{-1}}{\partial u_h^{k-1}} R(u_h^{k-1}) \end{aligned} \tag{44}$$

This work uses the differentiation of a matrix inverse shown below in order to avoid having to exactly compute derivatives of the approximate inverse of the preconditioner matrix, as this would entail differentiating the linear solution process which is algorithmically difficult.

$$\frac{d [K]^{-1}}{dx} = - [K]^{-1} \left[\frac{dK}{dx} \right] [K]^{-1} \tag{45}$$

The derivative of the constraint term substituting in equation (45) is shown below.

$$\begin{aligned} \frac{\partial G^k}{\partial u_h^k} &= I \\ \frac{\partial G^k}{\partial u_h^{k-1}} &= -I + [P_{k-1}]^{-1} \left[\frac{\partial R(u_h^{k-1})}{\partial u_h^{k-1}} \right]_2 - [P_{k-1}]^{-1} \frac{\partial [P_{k-1}]}{\partial u_h^{k-1}} [P_{k-1}]^{-1} R(u_h^{k-1}) \end{aligned} \tag{46}$$

Using the definition of the nonlinear solver increment (Δu), the above equation can be simplified as follows:

$$\begin{aligned} \frac{\partial G^k}{\partial u_h^k} &= I \\ \frac{\partial G^k}{\partial u_h^{k-1}} &= -I + [P_{k-1}]^{-1} \left[\left[\frac{\partial R(u_h^{k-1})}{\partial u_h^{k-1}} \right]_2 - \frac{\partial [P_{k-1}]}{\partial u_h^{k-1}} \Delta u_h \right] \end{aligned} \tag{47}$$

These Hessian vector products can be computed using complex Frechét derivatives, rather than hand differentiating the residual operator twice to obtain the Hessian operator. Using the equation for the adjoint at the final pseudo-time step with the constraint derivatives returns the identity below.

$$[I]\,\Lambda_h^n = -\left[\frac{\partial L}{\partial u_h^n}\right]^T \tag{48}$$

Substituting in the constraint derivatives from equation (47) into equation (36) returns:

$$[I]\,\Lambda^{k-1} = -\left[-I + [P_{k-1}]^{-1}\left[\left[\frac{\partial R(u_h^{k-1})}{\partial u_h^{k-1}}\right]_2 - \frac{\partial\,[P_{k-1}]}{\partial u_h^{k-1}}\Delta u_h\right]\right]^T \Lambda_h^k \tag{49}$$

This recurrence relation can be rewritten in delta form as below.

$$\Delta\Lambda_h = -\left[[P_{k-1}]^{-1}\left[\left[\frac{\partial R(u_h^{k-1})}{\partial u_h^{k-1}}\right]_2 - \frac{\partial\,[P_{k-1}]}{\partial u_h^{k-1}}\Delta u_h\right]\right]^T \Lambda_h^k \tag{50}$$

Distributing the transpose returns:

$$\Delta\Lambda_h = -\left[\left[\frac{\partial R(u_h^{k-1})}{\partial u_h^{k-1}}\right]_2 - \frac{\partial\,[P_{k-1}]}{\partial u_h^{k-1}}\Delta u_h\right]^T [P_{k-1}]^{-T}\,\Lambda_h^k \tag{51}$$

which motivates the definition of a secondary adjoint variable for each recurrence relation.

$$[P_{k-1}]^T\,\psi_h^k = \Lambda_h^k \tag{52}$$

It is possible to rewrite again into the delta form of the adjoint recurrence relation as follows.

$$\Delta\Lambda = -\left[\left[\frac{\partial R(u_h^{k-1})}{\partial u_h^{k-1}}\right]_2 - \frac{\partial\,[P_{k-1}]}{\partial u_h^{k-1}}\Delta u_h\right]^T \psi_h^k \tag{53}$$

It is important to note that exact dual correspondence here between the adjoint solver and the forward one is not required. Additionally, all these operations are on the embedded mesh, which makes this adjoint tremendously expensive, as it requires nearly the expense of a full nonlinear solution on the much finer embedded mesh. To make this more tractable, in this work the adjoint is calculated on the coarse mesh, and then uses the same interpolants as in the steady state adjoint error estimate to get an estimate of the adjoint on the embedded mesh. In other words, the linearization of the nonlinear process – the calculation of the pseudo-temporal adjoints – is done on the coarse mesh, while the fixed-point iterations weighted by pseudo-temporal adjoints are computed on the fine mesh. This means that convergence of the adjoint problem is guaranteed if the dual solver of the analysis linear solver is used at each iteration. While this does make the process cheaper, it is still necessary to interpolate the coarse mesh solution onto the fine mesh and compute the constraints on the fine mesh, which is a nontrivial expense. Hence the interest in using the flow reconstruction correction mentioned previously. The error equation can then be rendered as:

$$L = L(u_h^n, u_h^{n-1}, ..., u_h^{n-m}) - L(\tilde{u}_h^n, \tilde{u}_h^{n-1}, ..., \tilde{u}_h^{n-m}) \approx -\epsilon_c - \epsilon_a \tag{54}$$

where ϵ_c (or the functional correction) is calculated similarly to the method in the dual-weighted residual method in section III and is shown below.

$$\begin{aligned}\epsilon_c = {} & {\Lambda_{BQ}^n}^T G^n(\tilde{u}_h^n, \tilde{u}_h^{n-1}) \\ & + {\Lambda_{BQ}^{n-1}}^T G^{n-1}(\tilde{u}_h^{n-1}, \tilde{u}_h^{n-2}) \\ & + ... \\ & + {\Lambda_{BQ}^1}^T G^1(\tilde{u}_h^1, \tilde{u}_h^0) \\ & + {\Lambda_{BQ}^1}^T \left[\frac{\partial G^1}{\partial \tilde{u}_h^0}\right](u_{BQ}^0 - u_{BL}^0)\end{aligned} \tag{55}$$

ϵ_a (or the remaining error) is calculated again using the adjoint error correction similar to in the dual-weighted residual method:

$$\begin{aligned}\epsilon_a &= (\Lambda^n_{BQ} - \Lambda^n_{BL})^T G^n(\tilde{u}^n_h, \tilde{u}^{n-1}_h) \\ &+ (\Lambda^{n-1}_{BQ} - \Lambda^{n-1}_{BL})^T G^{n-1}(\tilde{u}^{n-1}_h, \tilde{u}^{n-2}_h) \\ &+ ... \\ &+ (\Lambda^1_{BQ} - \Lambda^1_{BL})^T G^1(\tilde{u}^1_h, \tilde{u}^0_h) \\ &+ (\Lambda^1_{BQ} - \Lambda^1_{BL})^T \left[\frac{\partial G^1}{\partial \tilde{u}^0_h}\right](u^0_{BQ} - u^0_{BL})\end{aligned} \tag{56}$$

The flow reconstruction correction at iteration k, given by $(\Lambda^k_{BQ} - \Lambda^k_{BL})^T \left[\frac{\partial G^k}{\partial \tilde{u}^{k-1}_h}\right](u^{k-1}_{BQ} - u^{k-1}_{BL})$ is implemented by differentiating the fixed point iteration on the fine mesh, this being the only instance of differentiation of the fixed-point iteration on the fine mesh as all other linearizations of the fixed-point iterations are done on the coarse mesh.

$$\frac{\partial G^k_h}{\partial u_h} = \left[-I + [P_{k-1}]^{-1}_h \frac{\partial P_{k-1}}{\partial u^{k-1}_h}[P_{k-1}]^{-1}_h R_h(\tilde{u}^{k-1}_h) - [P_{k-1}]^{-1}_h \frac{\partial R_h}{\partial u^{k-1}}\right] \tag{57}$$

IV.A.1. Error Estimation Using the Virtual Mesh Method

For the virtual mesh method, the fine mesh fixed point iteration is done using the residual evaluations on the virtual fine mesh as shown in III.B, with the coarse mesh preconditioner matrix. This is expressed mathematically as:

$$G^k(u^k, u^{k-1})^h_H = u^k_H - u^{k-1}_H - [P_{k-1}]_H R(u^{k-1}_h)^h_H \tag{58}$$

where $R(\tilde{u}^{(k-1)}_h)^h_H$ is the fine mesh residual calculated on the virtual mesh from the interpolated flow solution $\tilde{u}^{k-1}_h$ and restricted back to the coarse mesh. A parallel Δu^h_H is defined that corresponds to the fine mesh Δu restricted back to the coarse mesh, and the error estimation derivation proceeds.

$$\Delta u^h_H = -[P_{k-1}]_H R(\tilde{u}^{k-1}_h)^h_H \tag{59}$$

Alternatively, the preconditioner matrix used could also be the fine mesh preconditioner matrix restricted back to the coarse mesh, in which case the fine mesh fixed point iteration becomes:

$$G^k(u^k, u^{k-1})_h = u^k_H - u^{k-1}_H - [P_{k-1}]^h_H R(u^{k-1}_h)^h_H \tag{60}$$

where $[P_{k-1}]^h_H$ uses the gradient reconstructed values to the face on the virtual fine mesh to form the preconditioner on the coarse mesh. In such a case the restricted Δu becomes:

$$\Delta u^h_H = -[P_{k-1}]^h_H R(\tilde{u}^{k-1}_h)^h_H \tag{61}$$

While this method showed promise it did not meaningfully affect the results and so the cheaper method that uses the virtual mesh for the fine mesh residual evaluation only was used. The computation of the adjoint is unaffected, but the computation of the error becomes:

$$d\eta^k_H = |\Lambda^{kT}_H G^k(u^k, u^{k-1})^h_H| \tag{62}$$

where:

$$\eta_H = \sum_{k=1}^{n} d\eta^k_H \tag{63}$$

The analogue of the flow reconstruction correction shown in the embedded mesh methods on the virtual mesh uses the derivative of the fixed point on the virtual fine mesh and could be expressed as:

$$\frac{\partial G^k_H}{\partial u^{k-1}} = \left[-I + [P_{k-1}]^{-1}_H \frac{\partial P_{k-1}}{\partial u^{k-1}}_H [P_{k-1}]_H R(\tilde{u}^{k-1}_h)^h_H - [P_{k-1}]^{-1}_H \frac{\partial R}{\partial u^{k-1}}\right] \tag{64}$$

where the flow reconstruction error is computed on the virtual mesh and restricted back to the coarse mesh following the same procedure for the virtual mesh residual. This is done to avoid linearization of the

interpolation processes, so that the flow Jacobian can be computed off the coarse mesh values, however this correction returned poor results, and so the correction is neglected for the virtual mesh method, and the error estimate is kept to the expressions shown in equations 62 and 63. Having presented the three pseudo-temporal-adjoint-based error estimation routines, it is necessary to discuss the mesh refinement mechanics used in this work.

V. Mesh Refinement

For uniform mesh refinement, which is used in the embedded mesh error estimation subroutine, the driver simply marks all cells for a 4:1 refinement and all edges for a 2:1 refinement. It stores the parent child maps for the cells and uses those for the interpolations and computations required for the adjoint based error estimation. The residual is calculated on the fine mesh, and the adjoint is interpolated appropriately to form an error estimate. The refinement module is used for uniform refinement in the error estimation algorithm to provide an elementwise error estimate. The computed error distribution is then passed to the Refine code developed at NASA Langley[3] to produce adaptively refined meshes through the refinement process. The Refine code uses a multiscale-metric based refinement criteria where the metric at each node is defined by the error and the anisotropy of the mesh is determined through the gradient of the Mach number. The metric (Me) at each node is area averaged from the cells it comprises and is expressed as:

$$Me = \left(\frac{1}{\eta_g \eta_k}\right)^{\omega} \tag{65}$$

where ω is an under-relaxation parameter, and η_g and η_k are global and local error ratios. η_g is the integrated error in the mesh divided by the median error multiplied by the number of cells in the mesh. η_k is the cellwise ratio between the error in that cell and the median error in the mesh. This corresponds to targeting an error value in each subsequent mesh that is based on the median error in the previous mesh.[24]

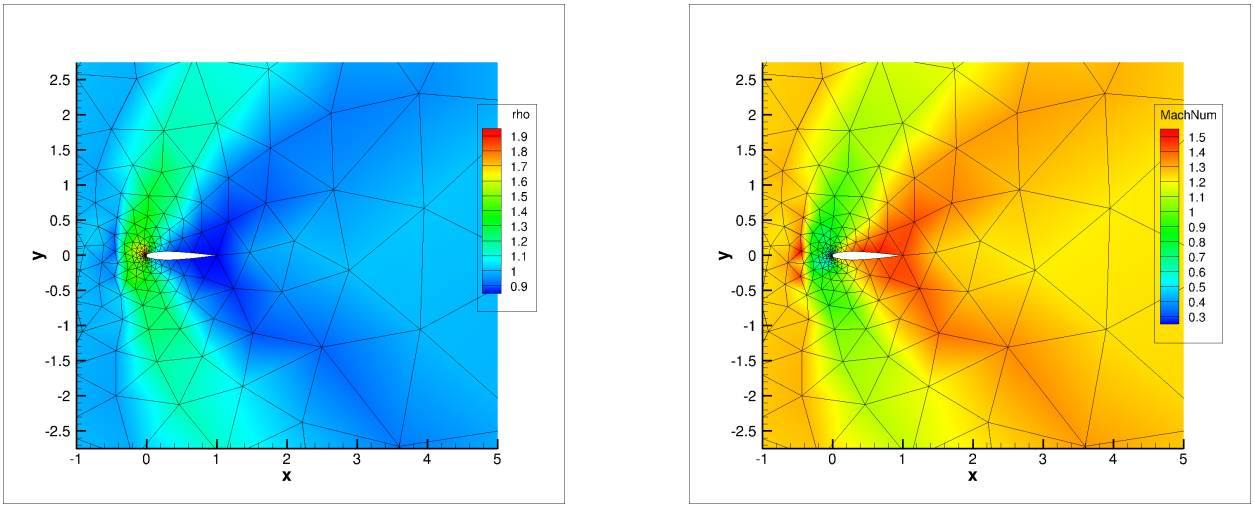
VI. Mesh Refinement Results

The test cases in this section show the results of a supersonic case and a transonic case. They are examined on the following criteria: consistency of the error estimate, convergence of the error estimate, and the mesh quality itself. The consistency of the error estimate means that as the error estimate is calculated on successively finer meshes, not only does the magnitude of the error estimate shrink, but the cells with the highest error estimates are refined; i.e. the highest magnitude error estimates are targeted first. This is the most important factor in our analysis, as we expect to see oscillations or lack of convergence in the objective due to the averaging windows used in this analysis. This method computes the error due to the spatial discretization, not due to the partial convergence or the averaging window and oscillatory behavior of the function. There are two caveats to keep mind for the results shown here as the adjoint correction formulation assumes smoothness of the fixed-point iteration and use of the fine mesh adjoint vector. First, the functional correction is computed using a biquadratic interpolation of the coarse mesh adjoint, this was done to decrease expense, but it also decreases accuracy of the functional correction, which becomes small rapidly with refinement. Second, these algorithms make heavy use of gradient reconstruction, which becomes highly inaccurate on stretched meshes.[25]

VI.A. Detached Bow Shock Test Case

The supersonic error estimation case is a NACA0012 airfoil in $M = 1.25$ flow, with zero angle of incidence. This case shows small scale unsteadiness due to lack of convergence because of limiter behavior. The initial mesh is very coarse as shown in Figure 2, and does not resolve the flow features well. On the coarse mesh the detached bow shock is not well resolved and the trailing edge fishtail shock that will appear as the mesh refines is not present except as a general smeared high Mach number region near the trailing edge. The mesh was then refined by the three different algorithms for pseudo-temporal error estimation presented previously: virtual mesh error estimate, embedded mesh error estimate, and the embedded mesh error estimate with functional correction, with the results of these refinements presented in the following sections. The output of interest for these cases is a sum of lift, drag, and entropy, $J = c_L + c_D + s$. The choice of entropy is used to illustrate the fineness of the refinement of the shock as it passes to the edges of the domain as the

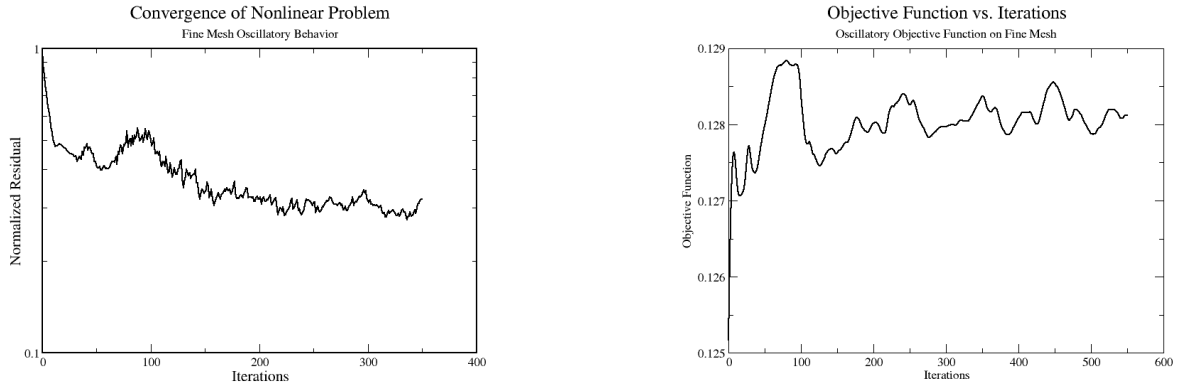
mesh is refined. For this case, the objective function was calculated at the final converged state for the first five refinement levels, at which point the mesh was fine enough that simulations would no longer converge and the objective became the pseudo-time averaged objective function calculated over the final 75 iterations. The mesh refinement was held isotropic for the first eleven meshes, at which point the mesh refinement used the Mach gradient to determine anisotropy. The error estimate was computed with backwards-in-pseudo-time integration back only through the objective function averaging window to reduce cost and the flow reconstruction correction was used to correct for this partial backwards-in-pseudo-time integration. The behavior on the final adapted mesh in Figure 3 shows a very ill-converged flow with an oscillatory objective function that looks to have oscillations with an amplitude of approximately 2% of the average value.



(a) Coarse Mesh Density

(b) Coarse Mesh Mach Number

Figure 2. Coarse mesh for detached bow shock error estimation case



(a) Fine Mesh Convergence

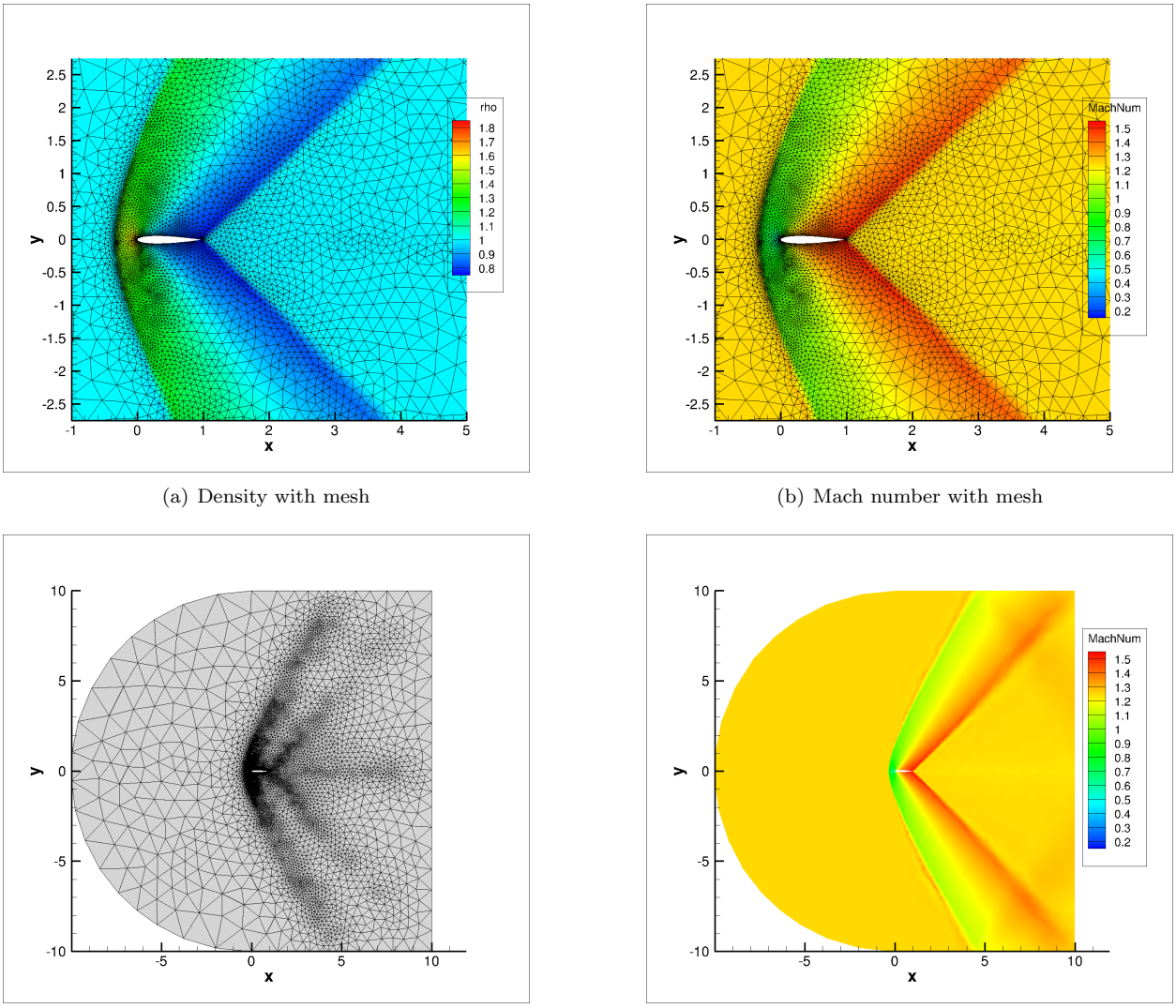
(b) Fine Mesh Objective

Figure 3. Objective and convergence behavior on final adapted mesh

VI.A.1. Virtual Mesh Error Estimation

This section shows a summary of the mesh refinement behavior for the detached bow shock test case when refined by the virtual mesh error estimate. This includes the mesh and flow field at the tenth adapted mesh (the last isotropic mesh), the mesh and flow field at the 18th adapted mesh (the final mesh), error histograms showing the distribution of the error in the mesh, and plots showing the convergence of the output-of-interest of error estimate as the mesh is refined. The mesh from the tenth adaptation cycle shows good behavior in the refinement pattern as shown in Figure 4. The detached bow shock is resolved well and is carried all the

way to the outer boundary, where it reflects off the boundary and crosses through the refined fishtail shock coming off the trailing edge. The reflection happens because the characteristic boundary condition used at the outer boundary is not a non-reflecting boundary condition. There also looks to be the beginning of a refinement of a line of increased entropy coming off the trailing edge of the airfoil. This is an effective and efficient refinement pattern, as shown by the heavy coarsening in front of the bow shock.



(a) Density with mesh

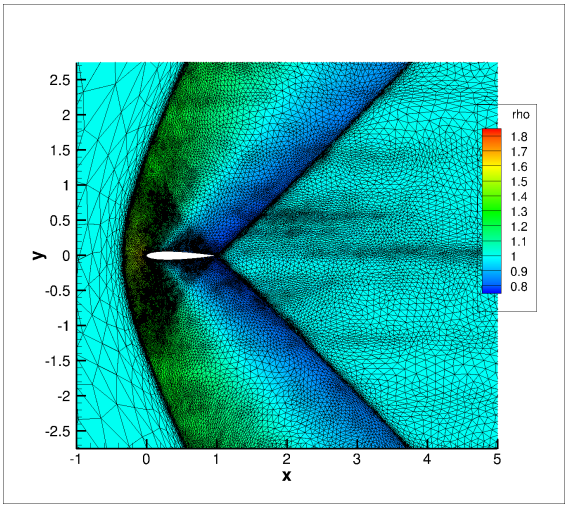
(b) Mach number with mesh

(c) Mesh in full domain

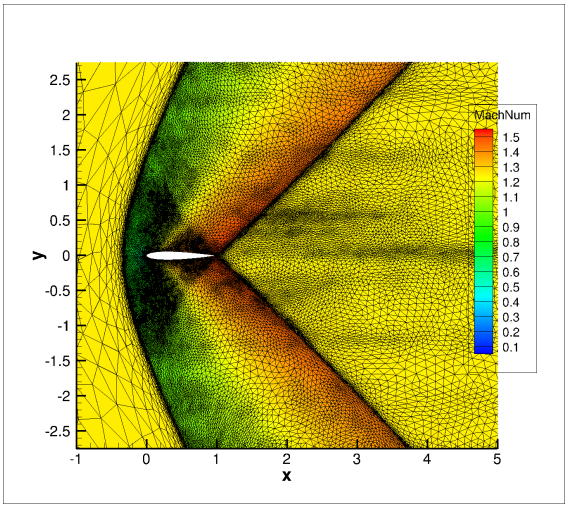
(d) Mach number in full domain

Figure 4. Tenth adaptation cycle for detached bow shock test case with virtual mesh error estimation (final isotropic mesh)

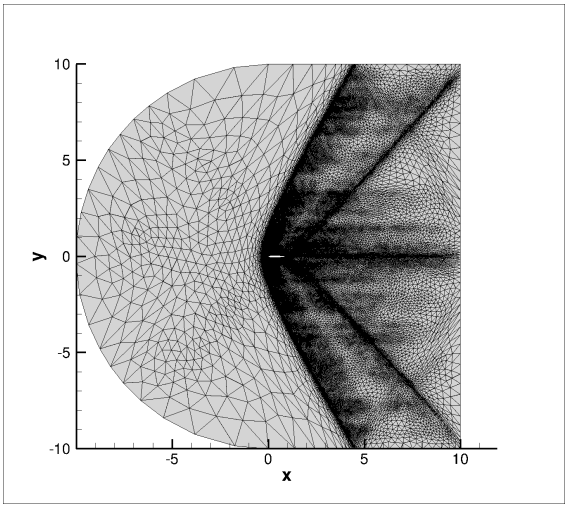
Figure 5 shows the final adapted anisotropic mesh. The final mesh shows a great deal of refinement along the bow shock and the reflection off the boundary, the fishtail shock, and the lines of increased entropy coming off the shocks. It also shows a high degree of refinement coming off the trailing edge forward to an intersection with the bow shock, enveloping the airfoil in an area of high refinement. The line of increased entropy has seen increased refinement as well. The Mach number plot shows the high degree of detail for the physical flow features, the sharpness of the shocks is specifically notable and occurs because, by including entropy in the objective function, the error estimation criteria drives the AMR to capture the constant entropy along the oblique shock. It is noticeable that there are streaks of heavier refinement corresponding to lines of increased Mach number due to the reflective behavior of the rear outer boundary. The next case shows these streaks of heavier refinement corresponding to lines of increased Mach number more prominently; this is because the virtual mesh method does not capture the lines of increased Mach number as well as the embedded mesh methods do.



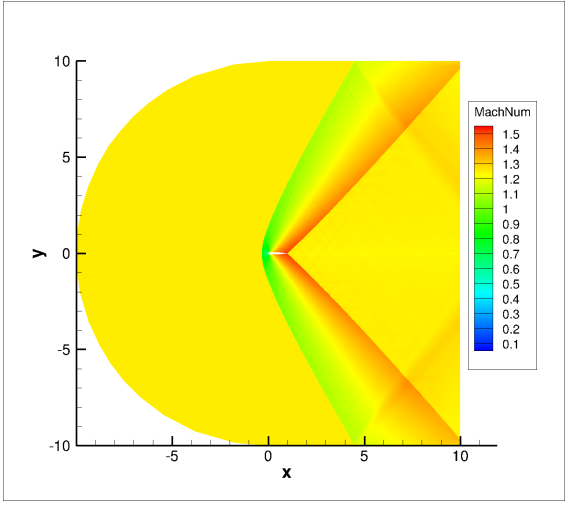
(a) Density with mesh



(b) Mach number with mesh



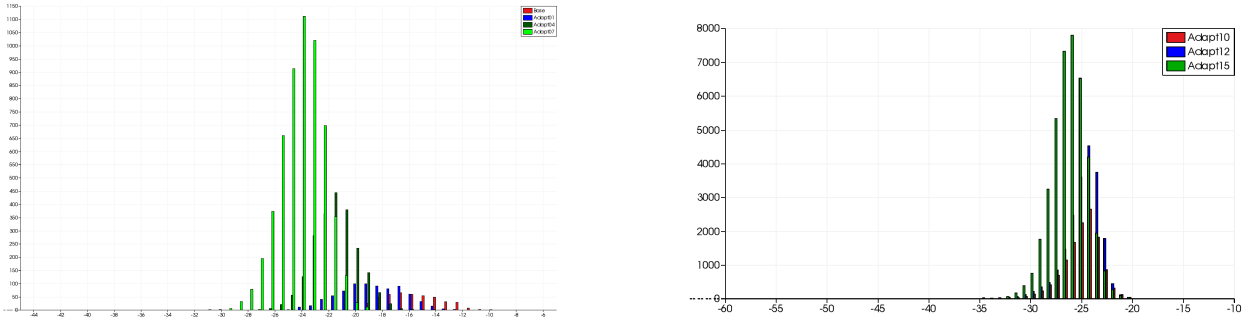
(c) Mesh in full domain



(d) Mach number in full domain

Figure 5. 18th and final adaptation cycle for detached bow shock test case with virtual mesh error estimation

Having investigated the behavior of the mesh refinement, the next concern is the consistency of the error estimate. For the purposes of this analysis, this is the most important aspect, as we should see consistency from mesh to mesh to have an accurate error estimate. It is expected that there will be oscillations in the functional output from mesh to mesh as the output is oscillatory on any given mesh. As such the consistency of the error estimate is our primary concern. The histograms of the error estimates in Figure 6 show the cells sorted into bins according to the $log_2(\eta_H)$, where the y-axis is the number of elements in a given bin. The expected behavior is that the highest error elements are consistently refined and that the mean steadily moves to the left – signifying decreasing error. The histogram showing the early adaptation behavior shows good consistency, whereas the histogram with the finer mesh error values shows less consistency, indicating a possible issue with this approach.

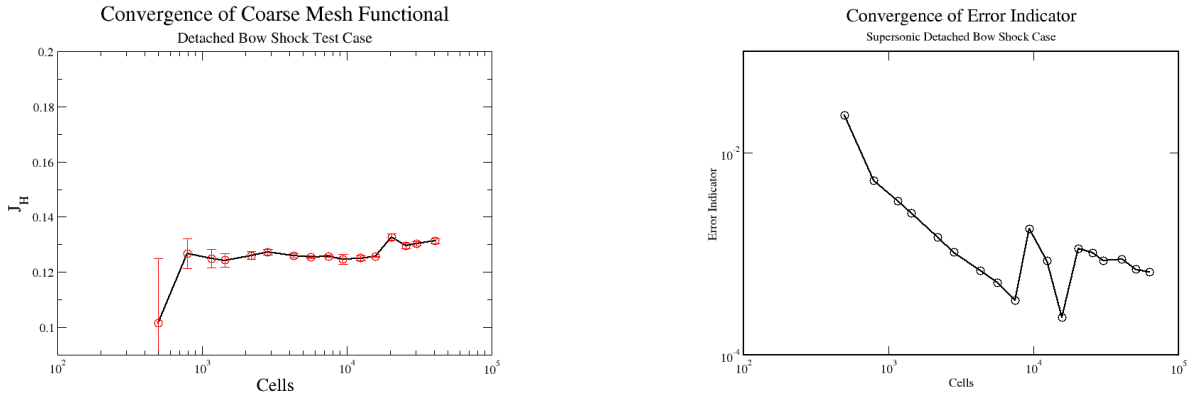


(a) Baseline to seventh adaptation cycle

(b) Tenth to 15th adaptation cycle

Figure 6. Error histograms for detached bow shock test case with virtual mesh error estimation

Figure 7 shows the convergence of the functional with red error bars in the left plot and the logarithmic behavior of the error indicator on the right. The functional convergence looks to converge within a bound acceptable for an oscillatory function like the ones examined in this work. The error convergence is well behaved until the finer meshes, where as was shown in the histograms, consistency and uniform decrease of error is lost.



(a) Coarse mesh functional with error bars

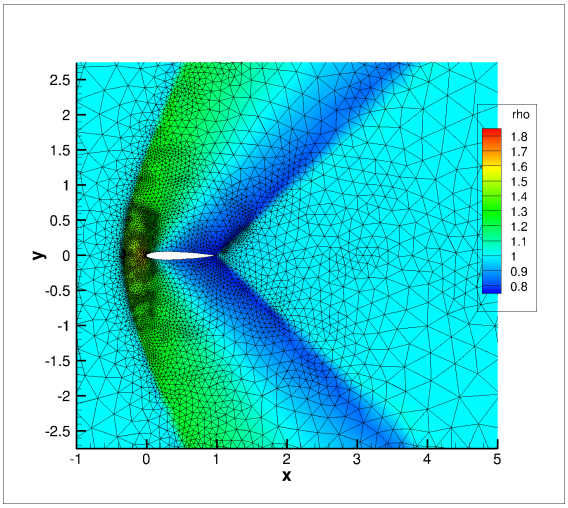
(b) Error convergence

Figure 7. Functional and error estimate convergence for supersonic detached bow shock test case with virtual mesh error estimation

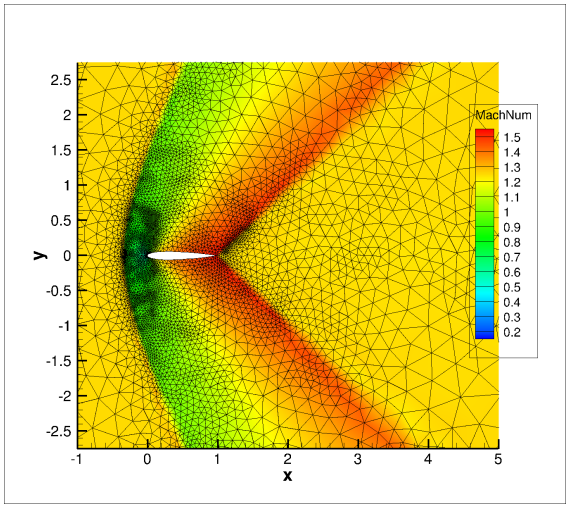
VI.A.2. Embedded Mesh Error Estimation

This section shows a summary of the mesh refinement behavior for the detached bow shock test case when refined by the embedded mesh error estimate. This includes the mesh and flow field at the tenth adapted mesh (the last isotropic mesh), the mesh and flow field at the 18th adapted mesh (the final mesh), error histograms showing the distribution of the error in the mesh, and plots showing the convergence of the output-of-interest of error estimate as the mesh is refined. Figure 8 shows the tenth adapted mesh, the last isotropic mesh, for the embedded mesh error estimation technique. This shows, as in the virtual mesh error

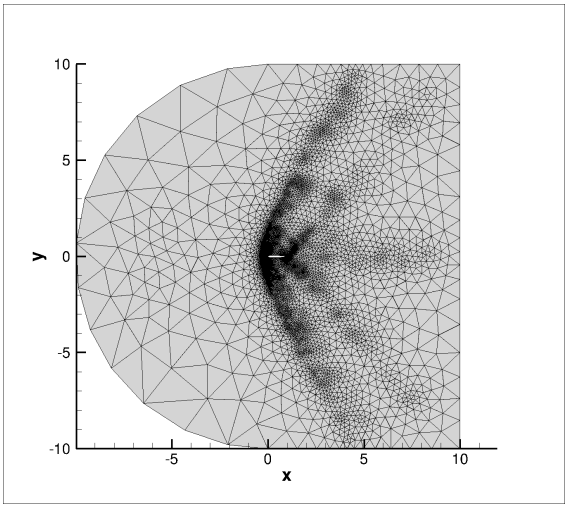
estimation technique, refinement along the detached bow shock, the fish tail shock, and the area of increased entropy behind the trailing edge of the airfoil. The mesh is refined well enough along both shocks to carry the bow shock and the fishtail shock all the way to the boundary and captures the reflecting shock interaction –even though it is fainter than on the final mesh. This case shows more refinement near the airfoil leading edge than that shown in Figure 8 for the virtual mesh refinement.



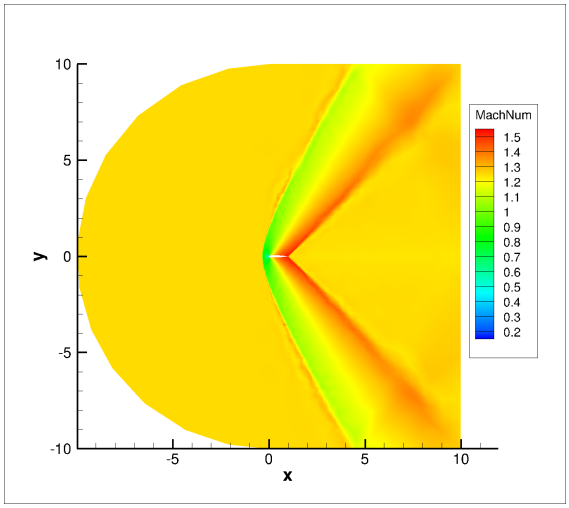
(a) Density with mesh



(b) Mach number with mesh



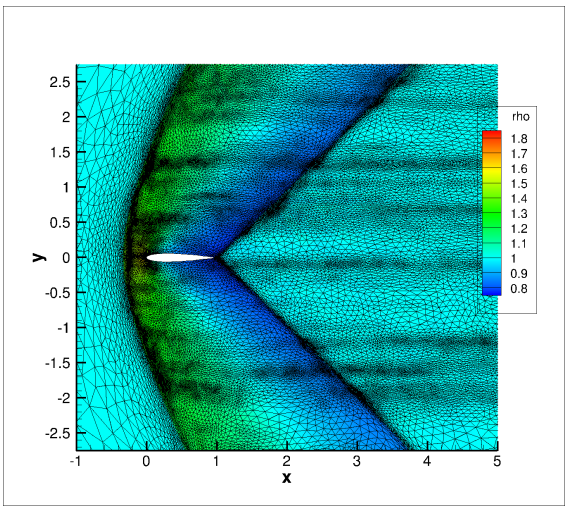
(c) Mesh in full domain



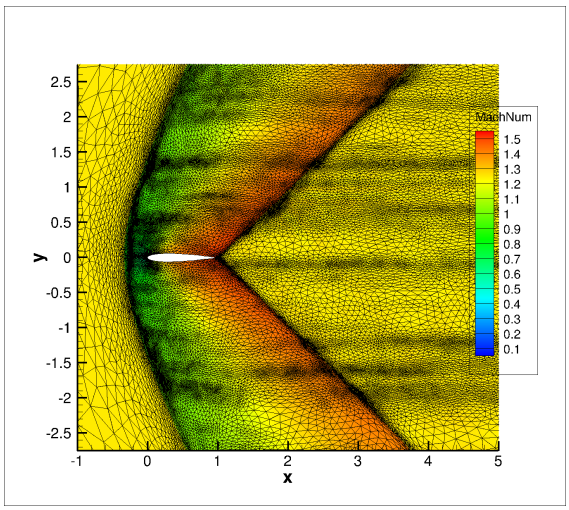
(d) Mach number in full domain

Figure 8. Tenth adaptation cycle for detached bow shock test case with embedded mesh error estimation (final isotropic mesh)

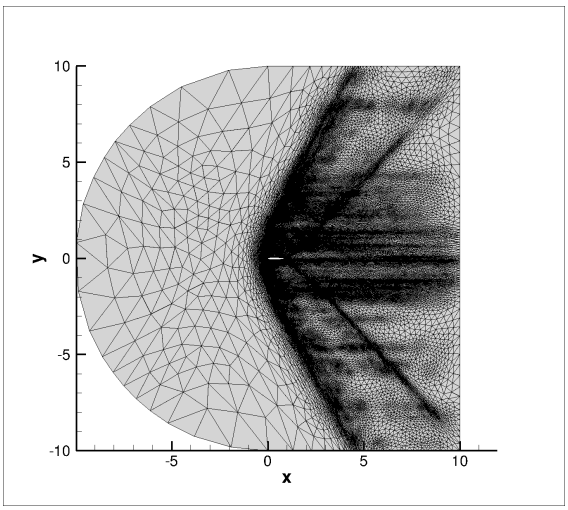
Figure 9 shows the final adapted anisotropic mesh; both shocks are more heavily refined and the shock reflection and interaction is well captured. The mesh is heavily coarsened in front of the bow shock, and tightly refined all along areas of increased entropy due to the shock interactions and reflections. The lines of increased refinement post fishtail shock correspond well to the areas of slightly higher Mach number due to the shock reflections and entropy creation due to the flux function. The high level of refinement along these lines of increased entropy points to the importance of the behavior of the flux functions on the embedded mesh when it comes to entropy creation, as these lines were nearly absent for the virtual mesh refinement.
Figure 10 shows the histograms for the same refinement cycles as in Figure 6, but using the embedded mesh error estimate. In this case, the error estimates show the expected consistency even on the fine meshes.
Figure 11 shows the expected functional convergence with the exception of the penultimate mesh. Similarly, the error convergence decreases and then stagnates, with the exception being a spike at that same mesh. Examining that mesh shows that the issue is due to the stagnation of the nonlinear problem at a non-physical



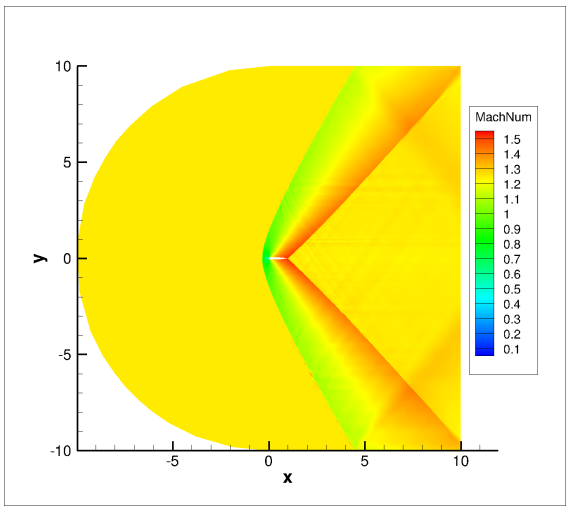
(a) Density with mesh



(b) Mach number with mesh

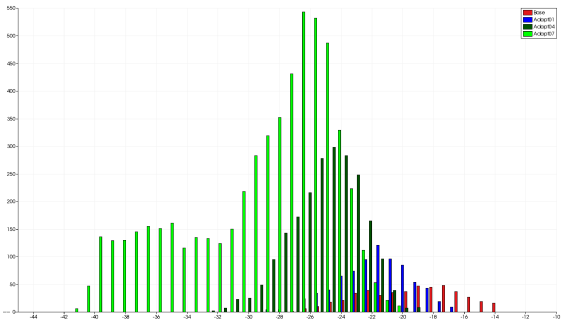


(c) Mesh in full domain

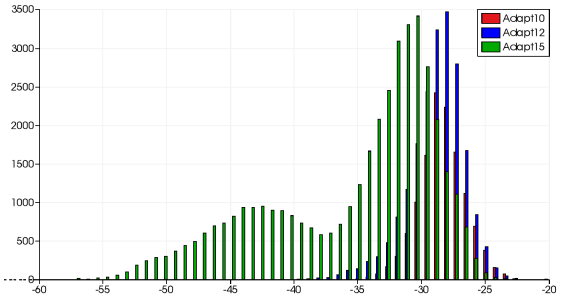


(d) Mach number in full domain

Figure 9. 18th and final adaptation cycle for detached bow shock test case with embedded mesh error estimation



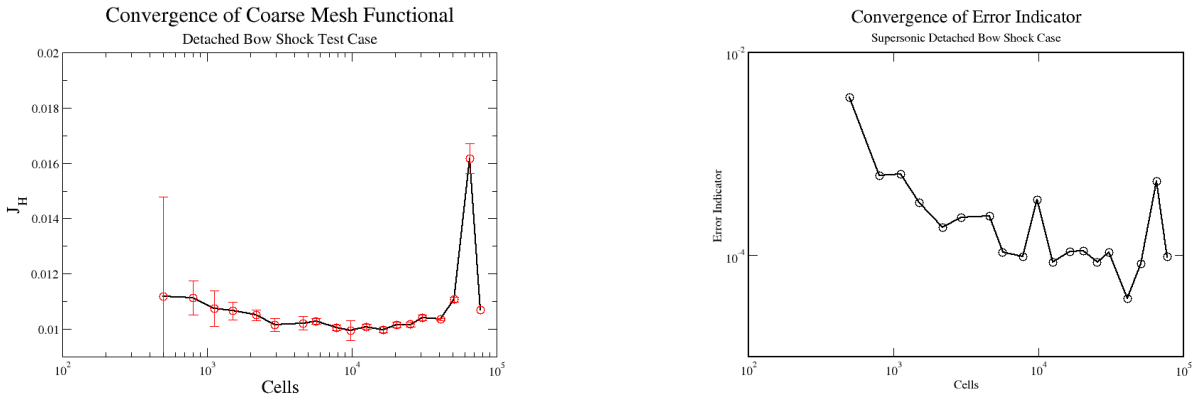
(a) Baseline to seventh adaptation cycle



(b) Tenth to 15th adaptation cycle

Figure 10. Error histograms for detached bow shock test case with embedded mesh error estimation

flow state markedly different from that on every other mesh, and the error estimate increases significantly. If the fine mesh pseudo-temporal adjoint were utilized then the adjoint based error estimate would be more likely to pick up such errors, but the biquadratic reconstruction of the adjoint is only a good approximation of the fine mesh adjoint for smooth flows.



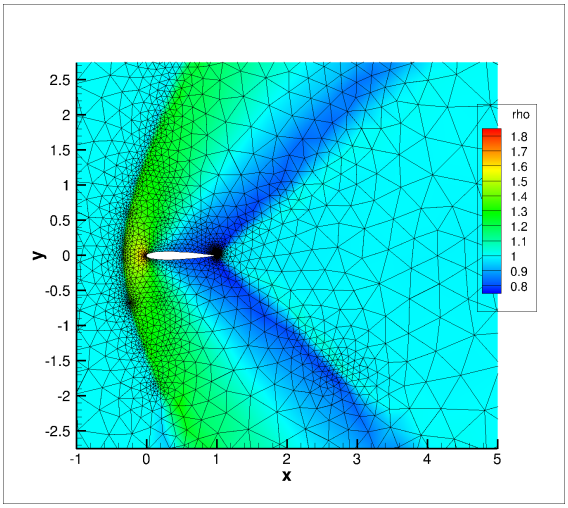
(a) Coarse mesh functional with error bars

(b) Error convergence

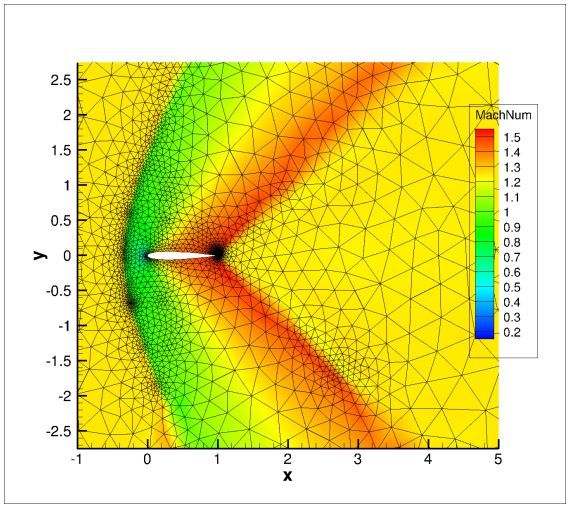
Figure 11. Functional and error estimate convergence for supersonic detached bow shock test case with embedded mesh error estimation

VI.A.3. Embedded Mesh Error Estimation and Functional Correction

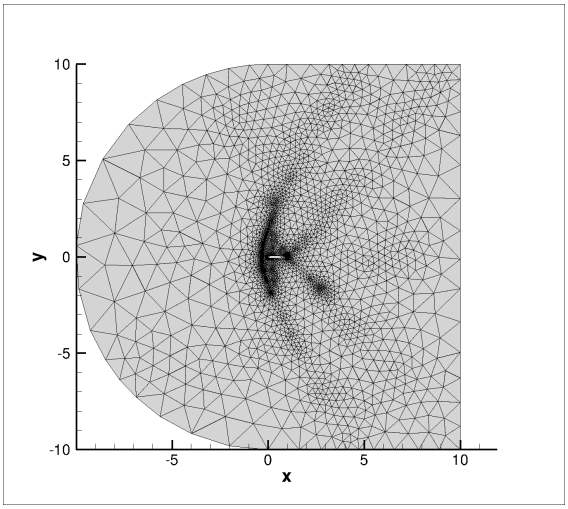
This section shows a summary of the mesh refinement behavior for the detached bow shock test case when refined by the embedded mesh error estimate with functional correction. This includes the mesh and flow field at the tenth adapted mesh (the last isotropic mesh), the mesh and flow field at the 18th adapted mesh (the final mesh), error histograms showing the distribution of the error in the mesh, and plots showing the convergence of the corrected output-of-interest of error estimate as the mesh is refined. This case has similar behavior as that exhibited by the previous two methods, carrying the shocks to the domain boundaries and capturing the shock reflection, as shown in Figure 12. This method showed less refinement along the leading edge and coming off the trailing edge. This can be attributed to the use of the functional correction.
The final adapted mesh in Figure 13 shows high refinement along the shocks and the shock reflection as well as some refinement from the trailing edge forward to the detached bow shock. There is also refinement along the line coming off the trailing edge, but there is not a high degree of refinement in other places due to these areas of higher entropy. This again can be attributed to the functional correction.
Figure 14 shows consistency in the error estimate as desired with distinct peaks for each mesh, moving towards the left with decreasing error.
Figure 15 shows the expected functional and error convergence, with stagnation of the error at approximately $1e-4$. The one outlier is the behavior at the 15th adapted mesh; this can again be attributed to pathological behavior in the flow field, reflected by an outlier in the objective function and a spike in the error.



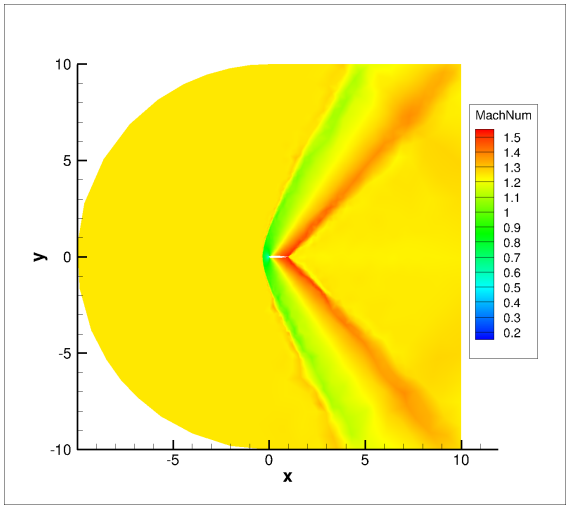
(a) Density with mesh



(b) Mach number with mesh

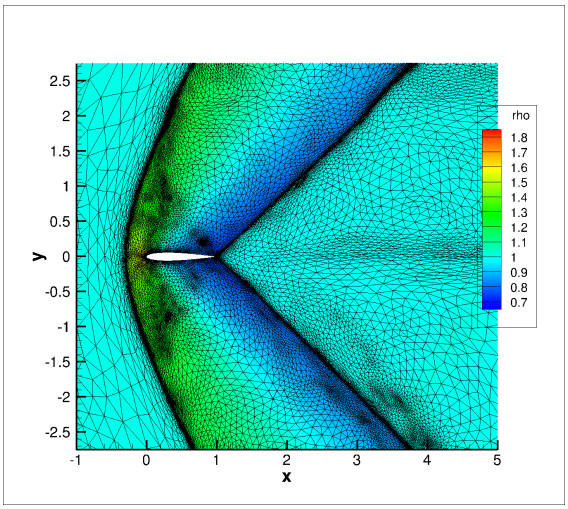


(c) Mesh in full domain

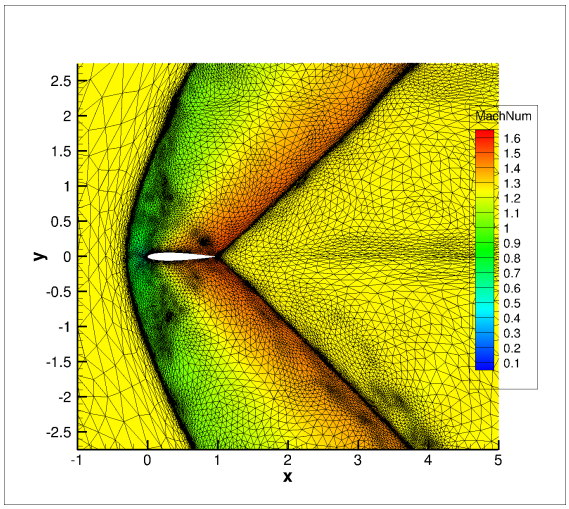


(d) Mach number in full domain

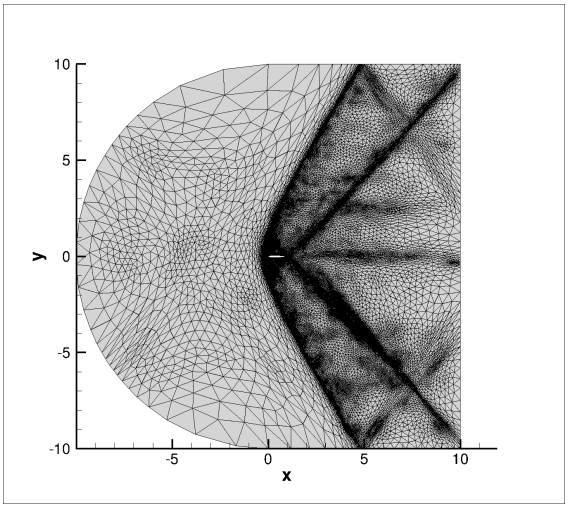
Figure 12. Tenth adaptation cycle for detached bow shock test case with embedded mesh error estimation and functional correction



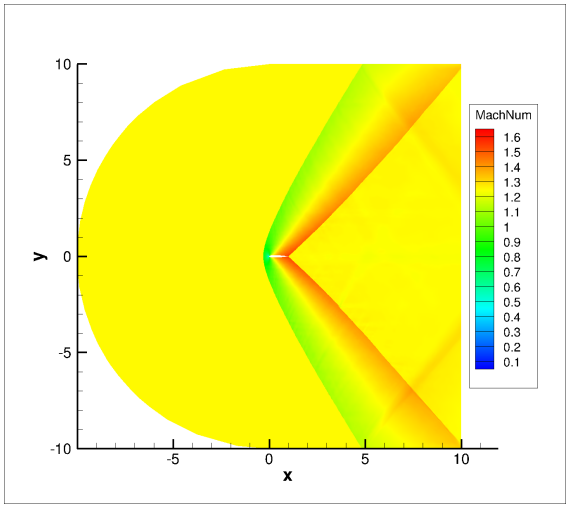
(a) Density with mesh



(b) Mach number with mesh

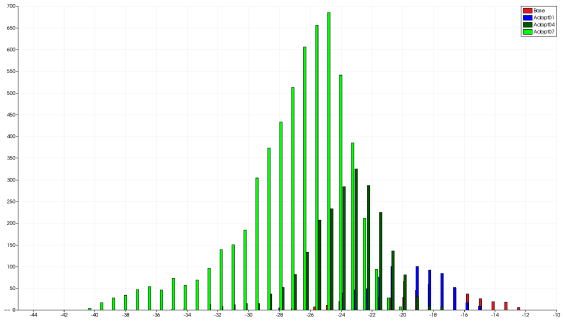


(c) Mesh in full domain

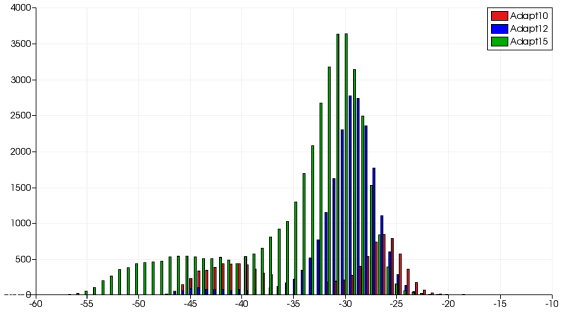


(d) Mach number in full domain

Figure 13. 18th and final adaptation cycle for detached bow shock test case with embedded mesh error estimation and functional correction

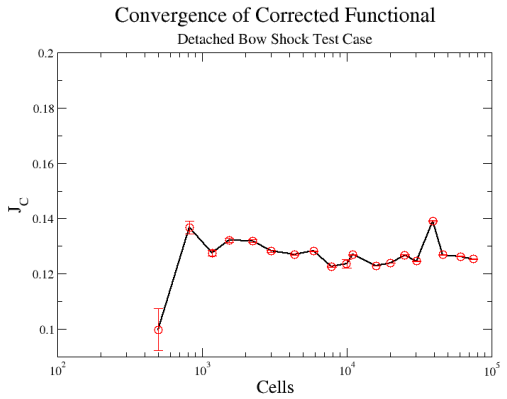


(a) Baseline mesh to seventh adaptation cycle

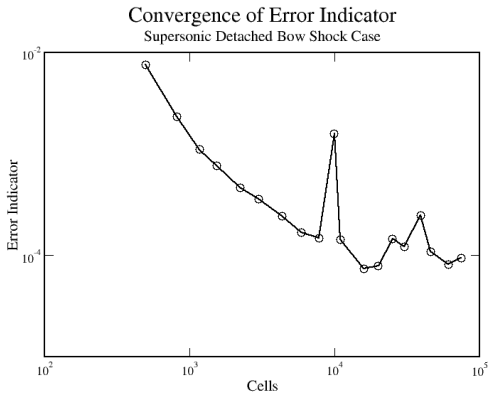


(b) 10th to 15th adaptation cycle

Figure 14. Error histograms for detached bow shock test case with embedded mesh error estimation and functional correction



(a) Corrected functional with error bars

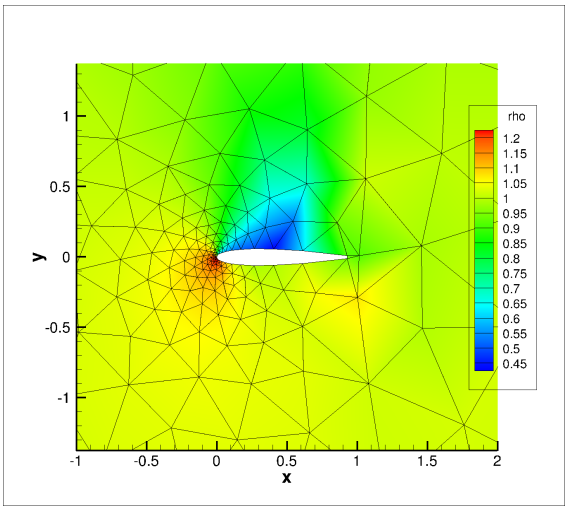


(b) Error convergence

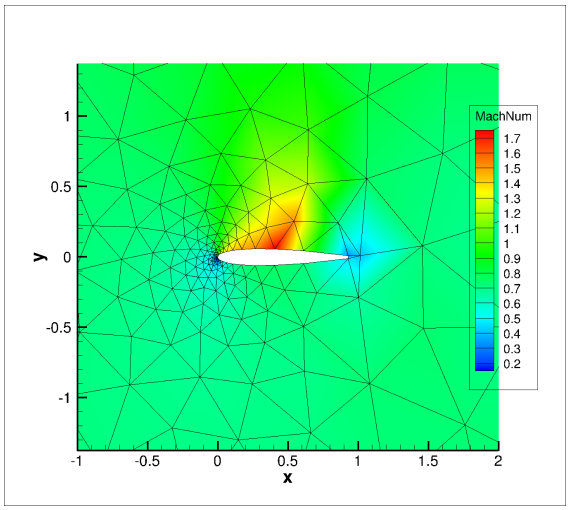
Figure 15. Corrected functional and error estimate convergence for detached bow shock test case with embedded mesh error estimation and functional correction

VI.B. Transonic Airfoil With Blunt Trailing Edge Test Case

Having shown good behavior on a case with very small trailing edge unsteadiness, it is necessary to look at a case with stronger unsteadiness; a NACA0012 airfoil truncated at 93% of the chord in $M = .75, \alpha = 5^o$ flow. This case shows mid-scale unsteadiness due to lack of convergence at the trailing edge, where the flow enters noticeable limit cycle oscillations on finer meshes. The initial mesh is very coarse as shown in Figure 16, and does not resolve the flow features well, not showing the upper surface shock well or the trailing edge unsteadiness and allowing the flow to converge. It was then refined by the same three algorithms as in the detached bow shock test case. The output of interest for these cases is a sum of lift and drag, $J = c_L + c_D$. The first 6 meshes are refined isotropically, at which point anisotropic refinement is invoked.



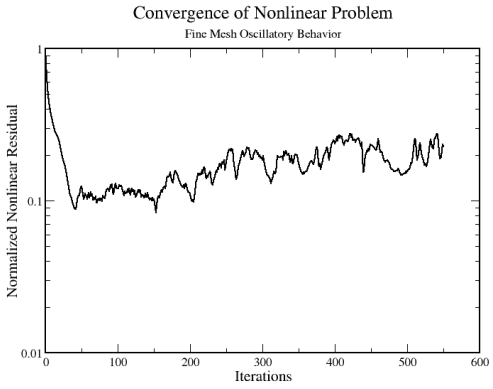
(a) Coarse Mesh Density



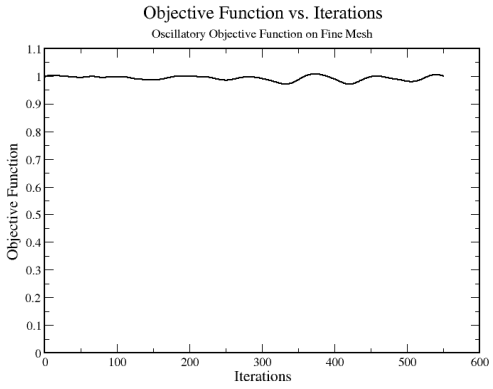
(b) Coarse Mesh Mach Number

Figure 16. Coarse mesh for transonic blunt trailing edge error estimation case

The behavior on the final adapted mesh shows a very ill-converged flow with an oscillatory objective function that has oscillations with an amplitude of approximately 5% of the average value. This shows much more oscillatory behavior than the detached bow shock case, which can be attributed to the blunt trailing edge. The functional behavior with the refined meshes reflect this oscillatory behavior.



(a) Fine Mesh Convergence



(b) Fine Mesh Objective

Figure 17. Objective and convergence behavior on final adapted mesh

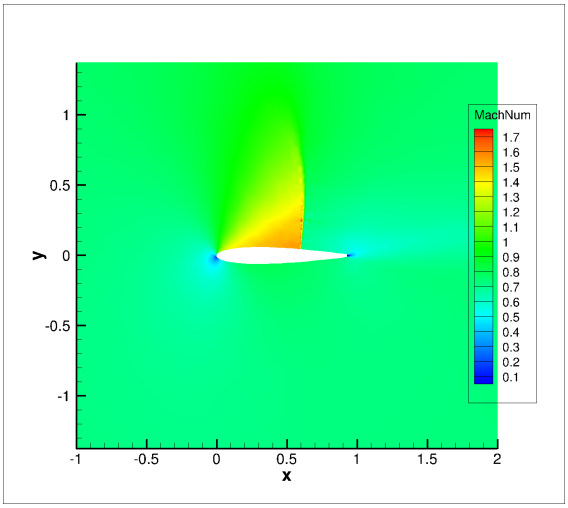
VI.B.1. Virtual Mesh Error Estimation

This section shows a summary of the mesh refinement behavior for the transonic blunt trailing edge NACA0012 test case when refined by the virtual mesh error estimate. This includes the mesh and flow field at the sixth adapted mesh (the last isotropic mesh), the mesh and flow field at the 16th adapted mesh (the final mesh), error histograms showing the distribution of the error in the mesh, and plots showing the convergence of the output-of-interest of error estimate as the mesh is refined. The sixth adapted mesh is the final isotropic mesh and shows noticeable refinement along the shock and the region leading to the trailing edge, as expected. The flow shows noticeable unsteadiness at the trailing edge as can be seen in Figure 18.

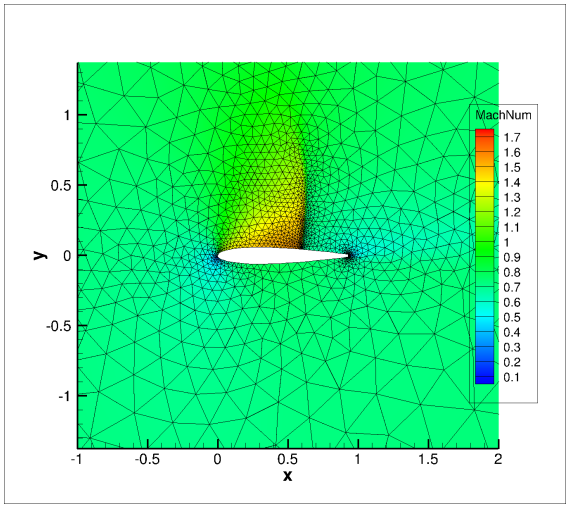
The final mesh and flow shown in Figure 19 show poor behavior. The mesh shows minimal refinement along the shock which will poorly capture the drag and lift. The streamline Mach number plot shows very strong unsteadiness which is expected in this case.

Figure 20 shows poor error estimate consistency on the coarse meshes, whereas before the virtual mesh error estimate was consistent on the coarse meshes but lost consistency as the meshes became finer. Here, the finer meshes show almost no consistency in the virtual mesh error estimate, and consistency of the error estimate is lost early on.

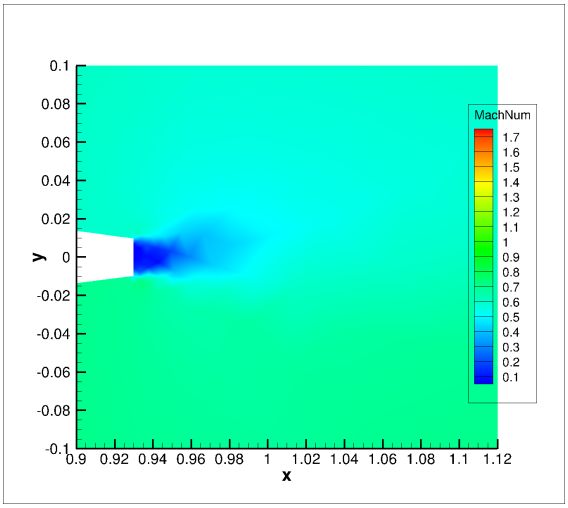
Figure 21 explains the loss of consistency; the error estimate spikes on later meshes. The reason for this has to do with calculating the residual on the virtual fine mesh. The virtual fine mesh appears to have highly inaccurate residual computation for the flow states when the flow expands around the trailing edge. The lack of limiters on the virtual fine mesh combined with the anisotropy of the mesh leads to states with very large residuals, which destroys the ability of this method to provide reliable or consistent error estimates. Investment in better interpolation and gradient reconstruction techniques could solve this problem. Due to the lack of smoothness in the flow states, it is hypothesized that embedded mesh techniques work best for this kind of problem.



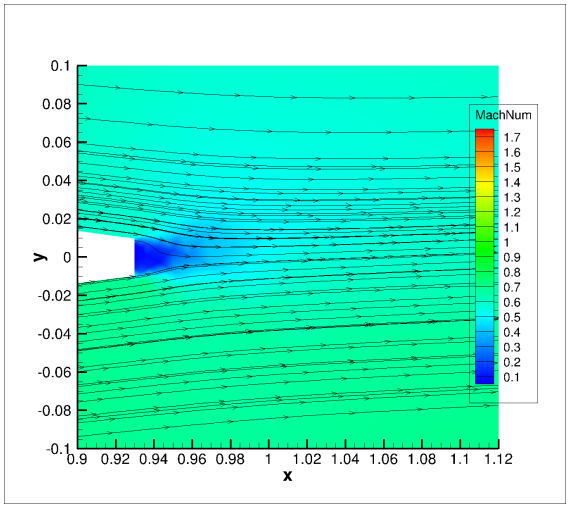
(a) Mach number without mesh



(b) Mach number with mesh

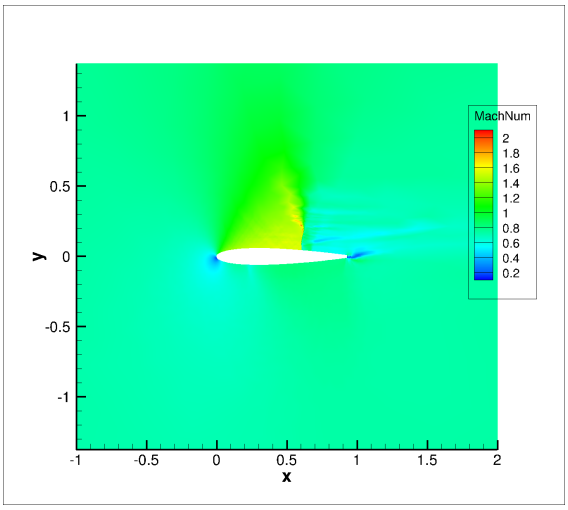


(c) Mach number at trailing edge

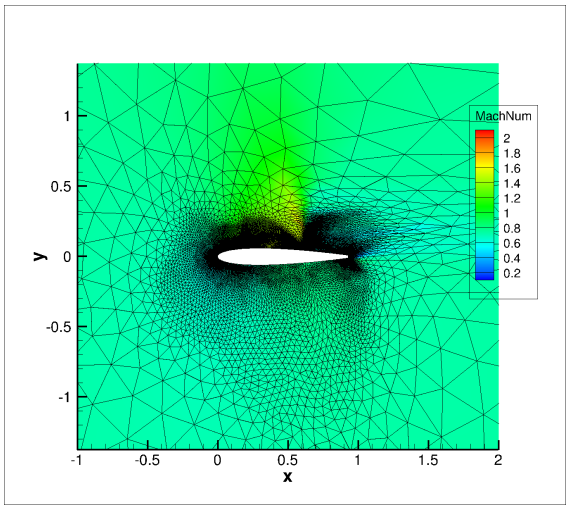


(d) Mach number with streamlines

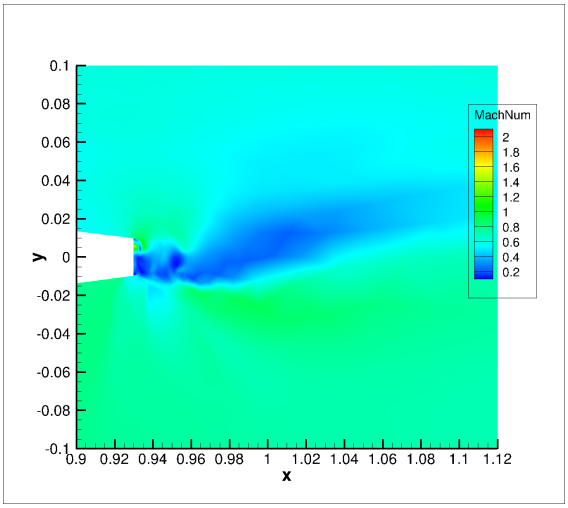
Figure 18. Sixth adaptation cycle for transonic blunt trailing edge test case with virtual mesh error estimation (final isotropic mesh)



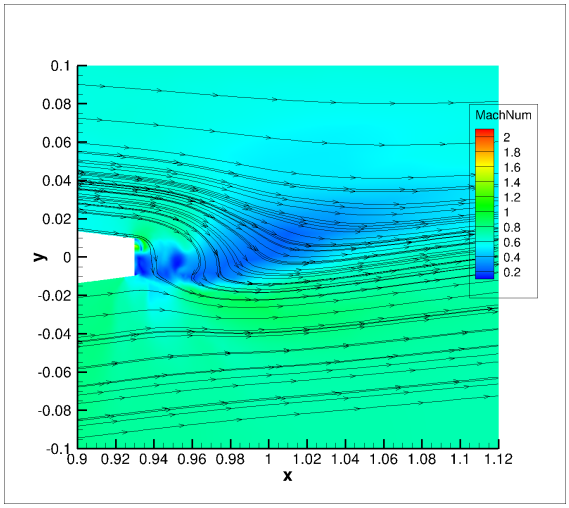
(a) Mach number without mesh



(b) Mach number with mesh

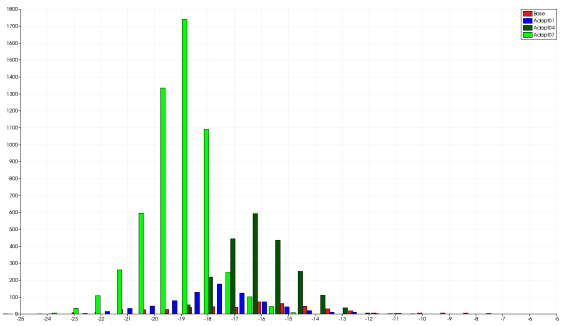


(c) Mach number at trailing edge

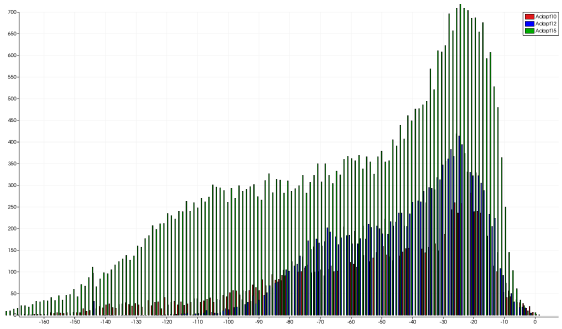


(d) Mach number with streamlines

Figure 19. 16th and final adaptation cycle for transonic blunt trailing edge test case with virtual mesh error estimation

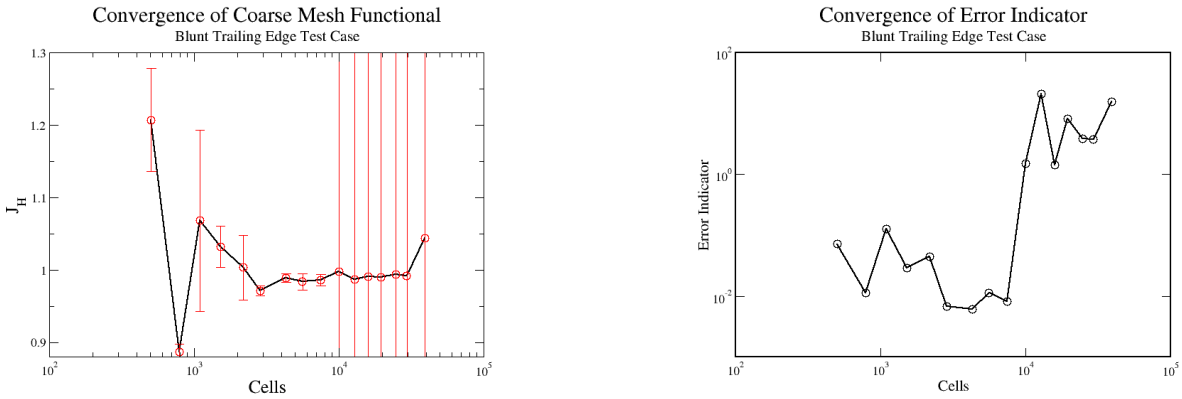


(a) Baseline to 7th adaptation cycle



(b) Tenth to 15th adaptation cycle

Figure 20. Error histograms for transonic blunt trailing edge test case with virtual mesh error estimation



(a) Coarse mesh functional with error bars

(b) Error convergence

Figure 21. Functional and error estimate convergence for transonic blunt trailing edge test case with virtual mesh error estimation

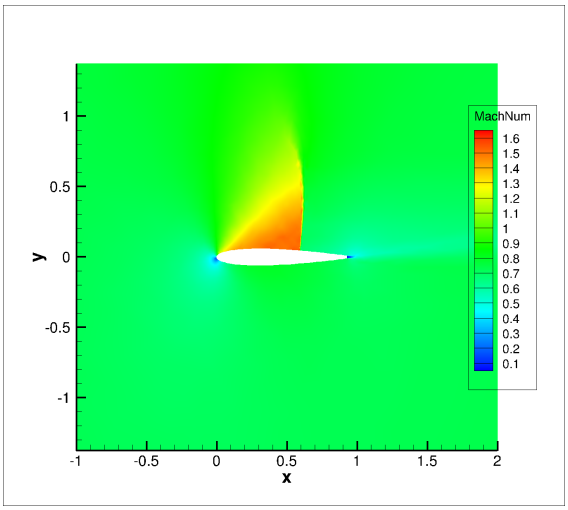
VI.B.2. Embedded Mesh Error Estimation

This section shows a summary of the mesh refinement behavior for the transonic blunt trailing edge NACA0012 test case when refined by the embedded mesh error estimate. This includes the mesh and flow field at the sixth adapted mesh (the last isotropic mesh), the mesh and flow field at the 16th adapted mesh (the final mesh), error histograms showing the distribution of the error in the mesh, and plots showing the convergence of the output-of-interest of error estimate as the mesh is refined. As in the virtual mesh adaptation, the sixth mesh, the final isotropic mesh, shows good refinement patterns which accord with expected results as shown in Figure 22. The mesh is adapted in the region behind the trailing edge where the flow is unsteady, the leading edge and along the shock, and the flow also shows noticeable unsteadiness as expected.

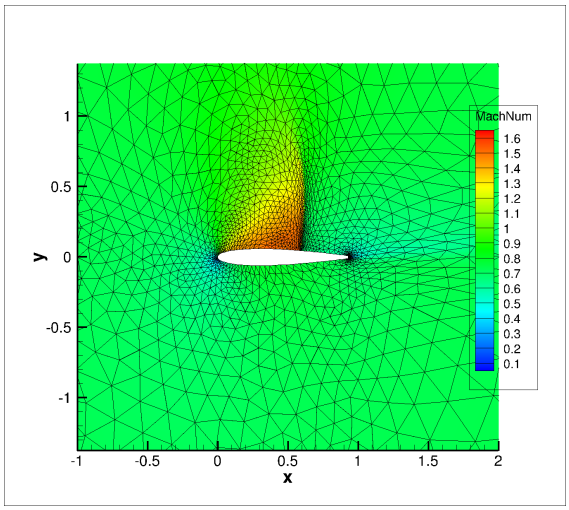
The final adapted mesh is refined well along the lower portion of the shock as well as showing good refinement along the forward streamline portion of the flow and the trailing edge as shown in Figure 23. The flow field at the trailing edge shows noticeable unsteadiness, which explains the oscillatory nature of the flow as was expected.

Figure 24 shows the histograms of the error, and it shows good consistency with the exception of the seventh adapted mesh; the seventh adapted mesh appears to have higher mean error than the baseline mesh. The error distribution on the finer meshes regains consistency as shown in the finer mesh histogram in Figure 24b.

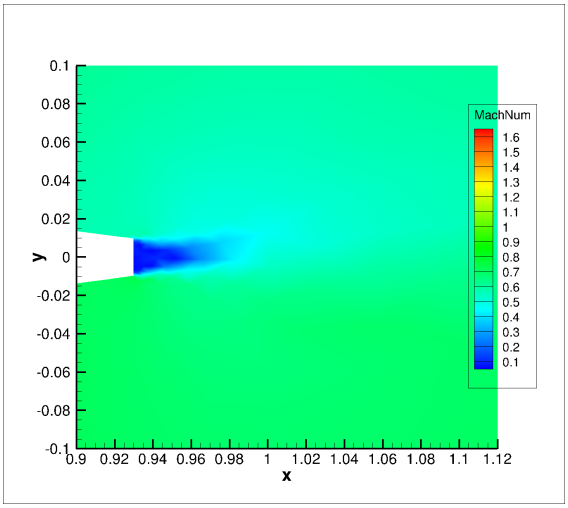
Figure 25 explains why the seventh adapted mesh loses the consistency. The seventh and ninth adapted meshes see spikes in the error for the same reason observed in the virtual mesh error estimation, the large fine mesh residuals that occur from reconstruction onto the fine mesh in non-smooth iterations. The functional convergence is reasonable for such an oscillatory case.



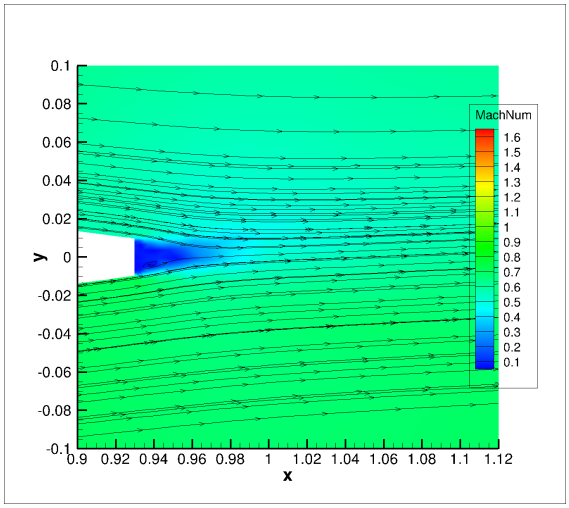
(a) Mach number without mesh



(b) Mach number with mesh

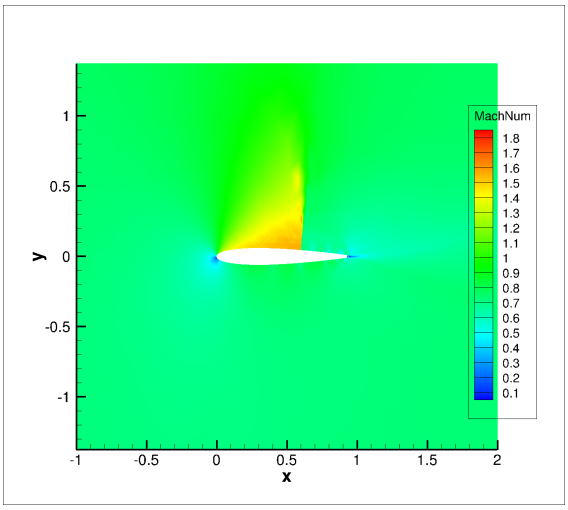


(c) Mach number at trailing edge

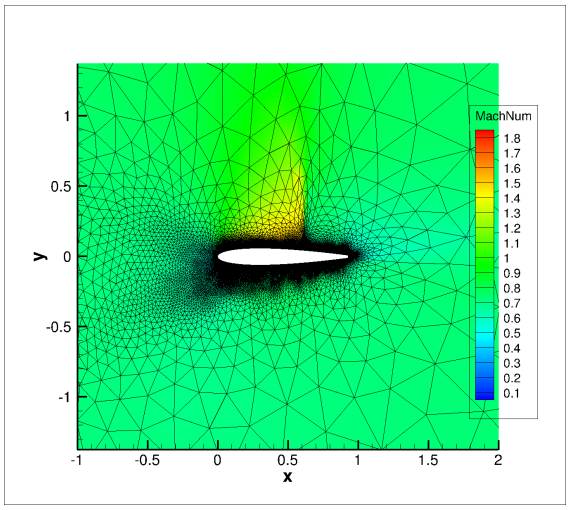


(d) Mach number with streamlines

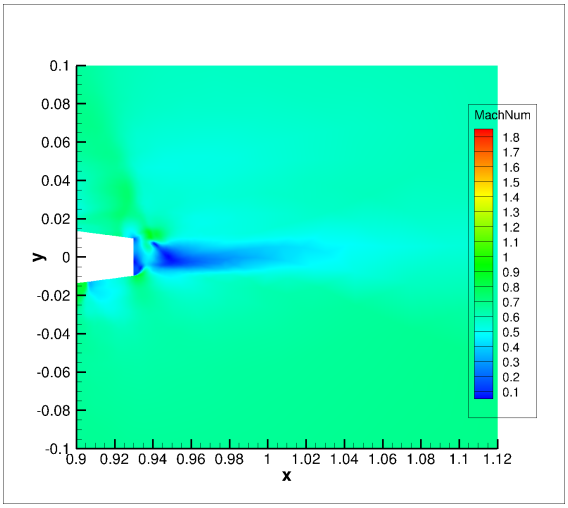
Figure 22. Sixth adaptation cycle for transonic blunt trailing edge test case with embedded mesh error estimation (final isotropic mesh)



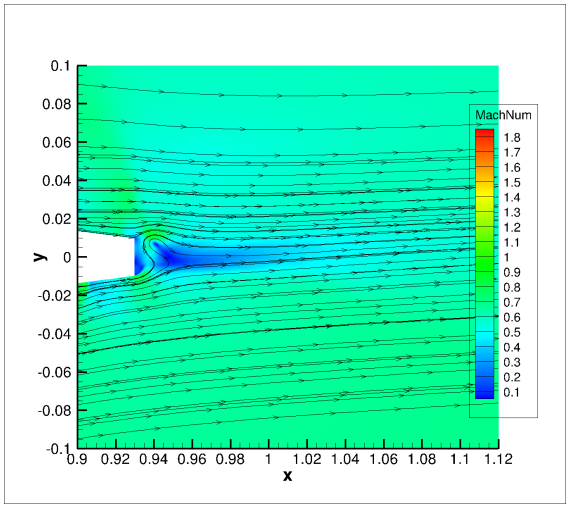
(a) Mach number without mesh



(b) Mach number with mesh

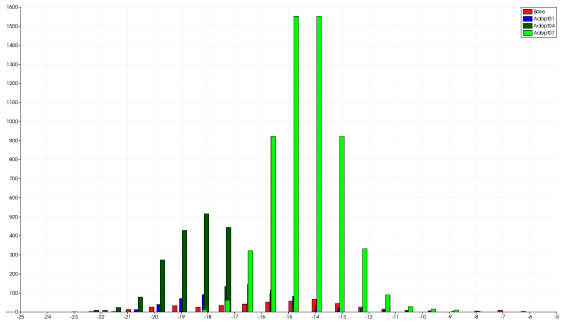


(c) Mach number at trailing edge

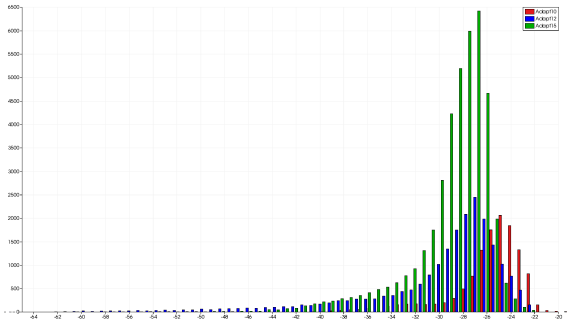


(d) Mach number with streamlines

Figure 23. 16th and final adaptation cycle for transonic blunt trailing edge test case with embedded mesh error estimation

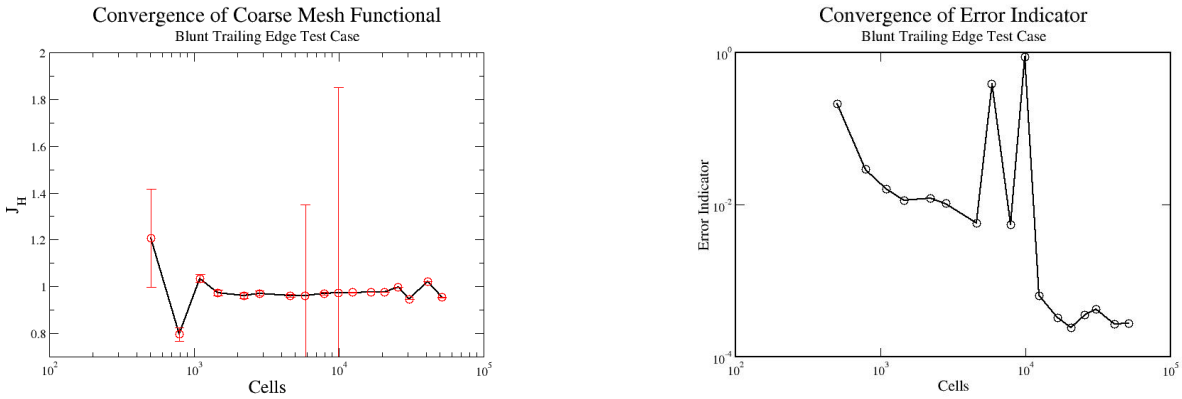


(a) Baseline to 7th adaptation cycle



(b) Tenth to 15th adaptation cycle

Figure 24. Error histograms for transonic blunt trailing edge test case with embedded mesh error estimation



(a) Coarse mesh functional with error bars

(b) Error convergence

Figure 25. Functional and error estimate convergence for transonic blunt trailing edge test case with embedded mesh error estimation

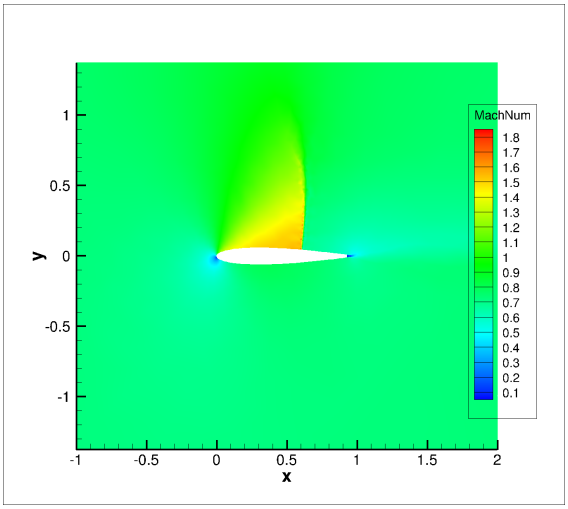
VI.B.3. Embedded Mesh Error Estimation and Functional Correction

This section shows a summary of the mesh refinement behavior for the transonic blunt trailing edge NACA0012 test case when refined by the embedded mesh error estimate with functional correction. This includes the mesh and flow field at the sixth adapted mesh (the last isotropic mesh), the mesh and flow field at the 16th adapted mesh (the final mesh), error histograms showing the distribution of the error in the mesh, and plots showing the convergence of the corrected output-of-interest of error estimate as the mesh is refined. We see excellent refinement behavior on the sixth adapted mesh shown in Figure 26. The shock is well refined, and the leading and trailing edges show refinement.

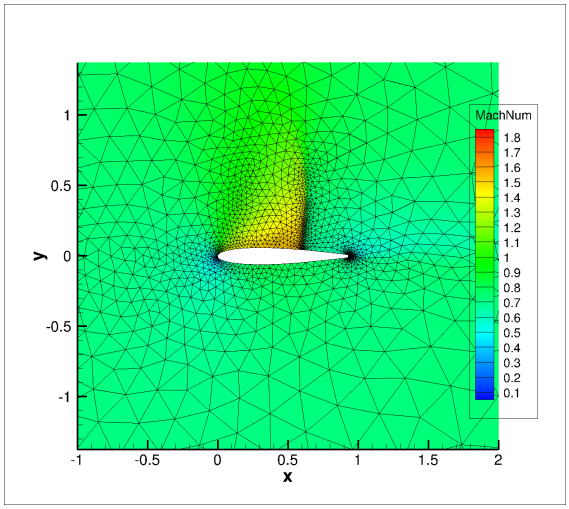
The final adapted mesh shown in Figure 27 shows good refinement along the shock as well as along the leading edge and the in the region behind the trailing edge where the flow is unsteady. This pattern is very similar to what is seen in steady state adjoint refinement for converged flows. This mesh does show greater refinement along the latter half of the airfoil than is typical due to the impact of the trailing edge unsteadiness along the rear section of the airfoil. In contrast with the previous error estimate, the method with the functional correction does not refine the forward half of the airfoil or the incoming flow region as heavily. This could be because the functional correction can calculate the error in that smooth flow region, and so the refinement gets focused more heavily on the shock and the unsteadiness at the trailing edge which are less smooth. The Mach number plots show the high degree of unsteadiness at the trailing edge, as is expected.

The error histograms shown in 28 show good consistency of the error estimate with the mean error steadily decreasing even on the finer meshes.

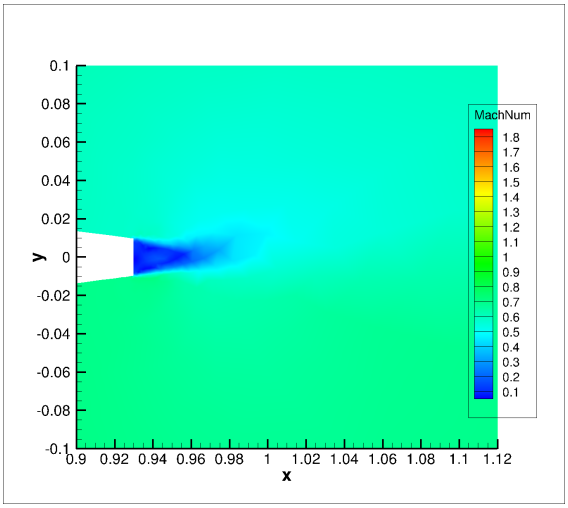
Figure 29 shows mostly the expected behavior for an oscillatory flow such as this, with mostly good behavior in the corrected functional and convergence in the error estimate until stagnation. The third adapted mesh has a very large error estimate and its corrected functional is a large outlier – this is due to the interpolation issues highlighted previously. While the interpolation becomes less important for the embedded mesh methods as they use limiters on the fine mesh, it can still lead states with very high residuals, especially for the highly non-smooth iterations that occur early on in the solution process in the highly transient domain.



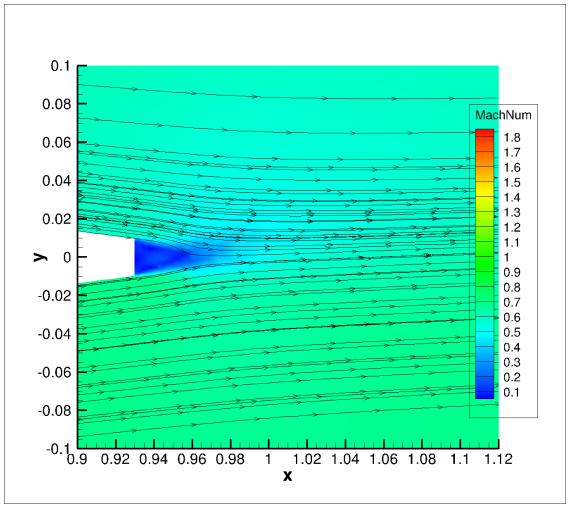
(a) Mach number without mesh



(b) Mach number with mesh

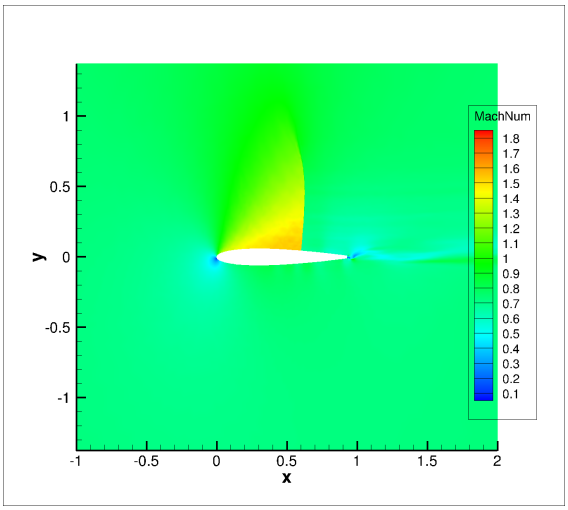


(c) Mach number at trailing edge

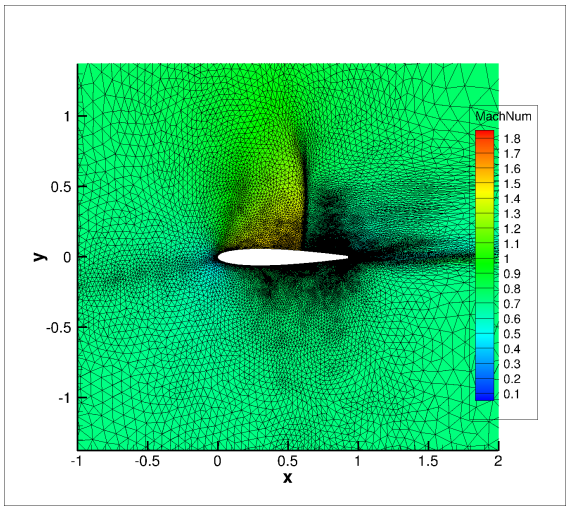


(d) Mach number with streamlines

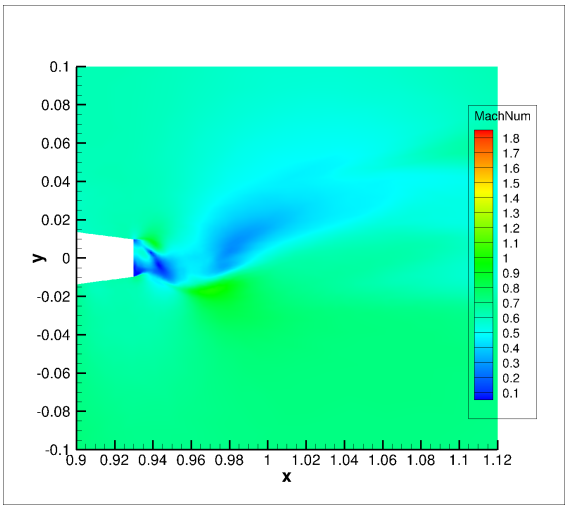
Figure 26. Sixth adaptation cycle for transonic blunt trailing edge test case with embedded mesh error estimation and functional correction (final isotropic adaptation)



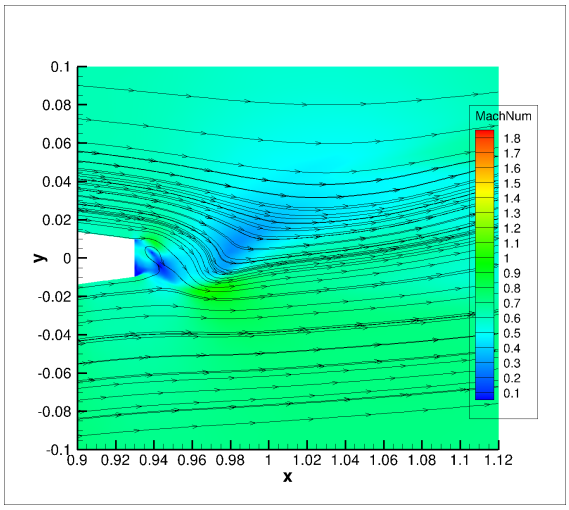
(a) Mach number without mesh



(b) Mach number with mesh

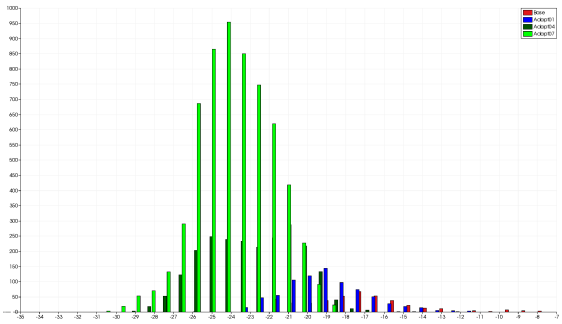


(c) Mach number at trailing edge

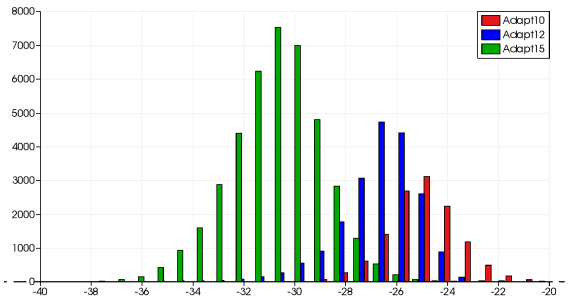


(d) Mach number with streamlines

Figure 27. 16th and final adaptation cycle for transonic blunt trailing edge test case with embedded mesh error estimation and functional correction

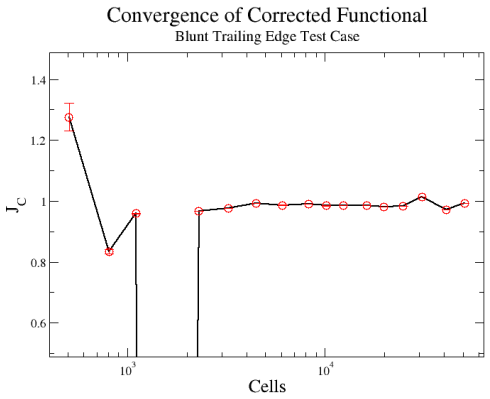


(a) Baseline mesh to seventh adaptation cycle

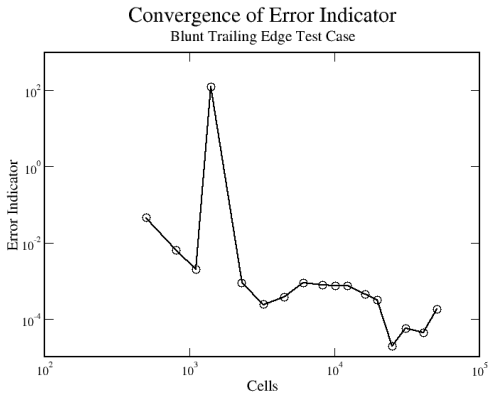


(b) Tenth to 17th adaptation cycle

Figure 28. Error histograms for transonic blunt trailing edge case test case with embedded mesh error estimation and functional correction



(a) Corrected functional with error bars



(b) Error Convergence

Figure 29. Corrected functional and error estimate convergence for transonic blunt trailing edge test case with embedded mesh error estimation and functional correction

VII. Conclusions

In this work we applied the pseudo-time accurate adjoint formulation to error estimation and created the dual-weighted constraint/fixed-point family of error estimation techniques. The three methods derived and implemented in this work were the virtual mesh method, the embedded mesh method, and the embedded mesh method with functional correction; all derived for the pseudo-time accurate adjoint applied to an inexact-quasi-Newton nonlinear solver. These methods were then applied to two test cases with different physical behaviors and degrees of unsteadiness.

All three error estimation methods perform well on the detached bow shock test case with smaller scale unsteadiness. The virtual mesh error estimate loses consistency on the finer meshes when measured by the mean on the error histogram, but the embedded mesh methods maintain consistency even on the finer meshes. All methods show decreasing error and converging objective functions in the presence of small scale oscillations. The virtual mesh method and the functional correction method do best in terms of refinement patterns due to their lack of refinement on the lines of increased Mach number. The virtual mesh method does not refine in these places because it cannot detect them, whereas the functional correction method corrects for the error in those smoother areas without having to refine in them.

In the transient blunt trailing edge test case, the virtual mesh method, which had previously shown promise is shown to be unworkable. It loses consistency of the error estimate on the coarser meshes and on the fine meshes its high dependence on the interpolation functions leads to issues in the computation of the error estimate as it integrates back in pseudo-time on the anisotropic meshes. The embedded mesh error estimate performs much better –it still has some interpolation related issues with the error estimation, but these show up with less frequency. Additionally, it keeps consistency of the error estimate until stagnation of the convergence of the error estimate. The best results come from the functional correction method, it has only one non-convergent error estimate due to the interpolation and keeps consistency of the error estimate throughout the finer meshes. The final adapted mesh shows nice adaptation patterns of the shock, the incoming flow, and the trailing edge unsteadiness, with the functional correction preventing over-refinement in smooth regions of the flow.

This method is tempting due to the high quality mesh refinement patterns and the consistent error estimates even in nonconvergent cases – showing approximately two to three orders of magnitude reduction in the error, with consistent error estimates for even the very low error cells with error values of approximately $1e{-}11$ even for these cases where the residual is not converging more than half an order of magnitude. However it shows some undesirable pathologies due to the heavy reliance on many different assumptions to limit the expense and the high reliance on gradient reconstruction and interpolation, methods that can be highly inaccurate on anisotropic meshes. In some of the finer meshes used, the inaccuracy of the gradient reconstruction leads to near divergence of the residual operator in some of the less smooth time-steps in the transient dominated portion of these flows. As such, for these meshes the error estimates and functional corrections are very inaccurate and dominated by the pathological residual evaluations. In order for this method to work more

consistently in flows with high-scale unsteadiness – such as the blunt trailing edge case – an investment in better interpolation mechanics and gradient reconstruction is necessary, high order limiting techniques may also be useful. Barycentric interpolation methods, which were designed for triangles and unstructured interpolation, could assist with interpolation for these non-smooth flows on highly anisotropic meshes. An interpolation related improvement could be handling of the adjoint reconstruction on the boundaries, for which currently the gradient in those cells is set to zero, and this impacts the nodal aggregation portion of the combined bilinear and biquadratic interpolation. It should be noted that the virtual mesh method works best for smooth solutions and is limited in that regard and the embedded mesh methods should work better on the non-smooth solutions that are shown and investigated in this work. An improvement that could generally assist with the utility of this method would be the windowing regularization methods mentioned in the optimization chapter and investigated by Krakos et al.[26] and Schotthofer et al.[27] These regularization methods have been shown to be effective for optimization methods and could assist with the functional convergence as the mesh is refined.

VIII. Acknowledgments

This work was supported in part by NASA Grant NNX16AT23H and the NASA Graduate Aeronautics Scholars Program. Computing time was provided by ARCC on the Teton supercomputer. The first author would like to thank Mike Aftosmis and Marian Nemec at NASA Ames for steering him towards an interest in error estimation and their help in the early portions of this work. We would also like to thank Mike Park at NASA Langley for his assistance with using the Refine code in this work.

References

[1] "Cart3D Web Page," https://www.nas.nasa.gov/publications/software/docs/cart3d/pages/cart3Dhome.html, Accessed: 2020-06-08.

[2] Nemec, M. and Aftosmis, M. J., "Toward Automatic Verification of Goal-Oriented Flow Simulations," Tech. Rep. Tech. Rep. TM-2014-218386, NASA Ames Research Center, August 2014.

[3] "Refine Repository," https://github.com/nasa/refine, Accessed: 2020-06-08.

[4] Kirby, A. C., Brazell, M. J., Yang, Z., Roy, R., Ahrabi, B. R., Stoellinger, M. K., Sitaraman, J., and Mavriplis, D. J., "Wind farm simulations using an overset hp-adaptive approach with blade-resolved turbine models," *The International Journal of High Performance Computing Applications*, Vol. 33, No. 5, 2019, pp. 897–923.

[5] Burgess, N. and Mavriplis, D., "An hp-Adaptive Discontinuous Galerkin Solver for Aerodynamic flows on Mixed-Element Meshes," AIAA Paper 2011-490, AIAA Aerospace Sciences Meeting, Orlando, FL, January 2011, https://doi.org/10.2514/6.2011-490.

[6] Becker, R. and Rannacher, R., "An optimal control approach to a posteriori error estimation in finite element methods," *Acta numerica*, Vol. 10, 2001, pp. 1–102.

[7] Venditti, D. A. and Darmofal, D. L., "Anisotropic Grid Adaptation for Functional Outputs: Application to Two-Dimensional Viscous Flows," *J. Comput. Phys.*, Vol. 187, No. 1, May 2003, pp. 22–46.

[8] Mani, K. and Mavriplis, D., "Error Estimation and Adaptation for Functional Outputs in Time-Dependent Flow Problems," AIAA Paper 2009-1495, AIAA Aerospace Sciences Meeting, Orlando, FL, January 2009, https://doi.org/10.2514/6.2009-1495.

[9] Padway, E. and Mavriplis, D. J., "Toward a Pseudo-Time Accurate Formulation of the Adjoint and Tangent Systems," 57th AIAA Aerospace Sciences Meeting, AIAA Paper 2019-0699, San Diego CA, January 2019. https://doi.org/10.2514/6.2019-0699.

[10] Padway, E. and Mavriplis, D. J., "Advances in the Pseudo-Time Accurate Formulation of the Adjoint and Tangent Systems for Sensitivity Computation and Design," 59th AIAA Aerospace Sciences Meeting, AIAA Paper 2020-3136, Virtual Event, June 2020. https://doi.org/10.2514/6.2020-3136.

[11] Krakos, J. A. and Darmofal, D. L., "Effect of Small-Scale Output Unsteadiness on Adjoint-Based Sensitivity," *AIAA Journal*, Vol. 48, No. 11, 2010, pp. 2611–2623.

[12] Mishra, A., Mavriplis, D. J., and Sitaraman, J., "Multipoint Time-Dependent Aero-elastic Adjoint-based Aerodynamic Shape Optimization of Helicopter Rotors," May 2015, pp. 828 – 844, AHS Forum 71,Virginia Beach VA, May 2015, pp 828 - 844.

[13] Mavriplis, D., "Revisiting the Least-Squares Procedure for Gradient Reconstruction on Unstructured Meshes," 16th AIAA Computational Fluid Dynamics Conference, Fluid Dynamics and Co-located Conferences, AIAA Paper 2003-3986, Orlando, Florida, 06/2003. https://doi.org/10.2514/6.2003-3986.

[14] LeVeque, R. J., *Numerical Methods for Conservation Laws*, Vol. 3, Springer, 1992.

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

[16] van Leer, B., "Flux-vector splitting for the Euler equations," *Eighth International Conference on Numerical Methods in Fluid Dynamics*, edited by E. Krause, Springer Berlin Heidelberg, Berlin, Heidelberg, 1982, pp. 507–512.

[17] Venkatakrishnan, V., "On the accuracy of limiters and convergence to steady state solutions," 31st Aerospace Sciences Meeting, AIAA Paper 1993-880, Reno NV, January 1993. https://doi.org/10.2514/6.1993-880.

[18]Michalak, C. and Ollivier-Gooch, C., "Accuracy preserving limiter for the high-order accurate solution of the Euler equations," *Journal of Computational Physics*, Vol. 228, No. 23, 2009, pp. 8693 – 8711.
[19]Nishikawa, H., "Robust numerical fluxes for unrealizable states," *Journal of Computational Physics*, Vol. 408, 2020.
[20]Anderson, W. K., Newman, J. C., and Karman, S. L., "Stabilized finite elements in FUN3D," *Journal of Aircraft*, Vol. 55, No. 2, 2018, pp. 696–714.
[21]Ceze, M. and Fidkowski, K., "Pseudo-transient Continuation, Solution Update Methods, and CFL Strategies for DG Discretizations of the RANS-SA Equations," .
[22]Ahrabi, B. R. and Mavriplis, D. J., "An implicit block ILU smoother for preconditioning of Newton–Krylov solvers with application in high-order stabilized finite-element methods," *Computer Methods in Applied Mechanics and Engineering*, Vol. 358, 2020, pp. 112637.
[23]Saad, Y., *Iterative methods for sparse linear systems*, Vol. 82, Society for Industrial and Applied Mathematics, 2003.
[24]Venditti, D. A., *Grid Adaptation for Functional Outputs of Compressible Flow Simulations, Ph.D. Dissertation*, Department of Aeronautics and Astronautics, Massachusetts Institute of Technology, USA, 2002.
[25]Diskin, B. and Thomas, J., "Accuracy of Gradient Reconstruction on Grids with High Aspect Ratio," Tech. Rep. NIA Report No. 2008-12, Neational Institute of Aerospace, December 2008.
[26]Krakos, J. A., Wang, Q., Hall, S. R., and Darmofal, D. L., "Sensitivity analysis of limit cycle oscillations," *Journal of Computational Physics*, Vol. 231, No. 8, 2012, pp. 3228 – 3245.
[27]Schotthofer, S., Zhou, B. Y., Albring, T. A., and Gauger, N. R., "Windowing Regularization Techniques for Unsteady Aerodynamic Shape Optimization," AIAA Aviation 2020 Forum, AIAA Paper 2020-3130, Virtual Conference, June 2020, https://arc.aiaa.org/doi/abs/10.2514/6.2020-3130.