Optimal Error Control using Discrete Adjoint Error Estimates in Unsteady Flow Problems

Bryan T. Flynt* and Dimitri J. Mavriplis†

Department of Mechanical Engineering, University of Wyoming, Laramie WY, 82071, USA

This paper presents a framework for the reduction of the spatial, temporal and algebraic error by use of the discrete adjoint solution in a time integrated functional (i.e. Lift, Drag, etc.) for problems of aerodynamic interest. The three types of error, along with cost estimates to refine each, are used to selectively adapt the discretization to reduce the error in the functional in the most efficient manner. Three test problems are used to demonstrate the algorithm and the resulting convergence tolerances, temporal and spatial discretization are displayed.

I. Introduction

Achieving satisfactory results with computational fluid dynamics has traditionally been a balancing act between available computational resources and the desired accuracy the practitioner hopes to achieve. This is especially true for time accurate simulations where the number of time steps, step size, grid resolution and convergence criteria all must be chosen to accurately capture the flow physics of interest without resulting in excessive computational cost. An experienced practitioner uses past results to chose these parameters expecting similar results but has no way to directly evaluate the error resulting from their choices. To try and reduce the expertise required, researchers have developed adaptive mesh refinement and adaptive time stepping schemes which track flow features or approximate errors within the solution. These methods only identify localized errors in the flow field or track high gradient features but do nothing to evaluate the influence these localized features have on the final quantity of interest (i.e. Lift, Drag, etc.). Recently, adjoint based adaptive methods have been gaining popularity to provide answers to this problem for both steady and unsteady problems.

An adjoint variable based approach permits the estimation of error relevant to a functional and the corresponding distribution of this error in the time and spatial domains. The method relies on applying discrete adjoint equations on a Taylor series expansion of a functional, where a linear approximation of the functional between two different resolutions (space or time) is estimated by solving the flow problem and the adjoint problem at the coarse resolution and then projecting it to the finer resolution. The method only predicts the error in the flow solution that is relevant to the functional of interest between two different resolutions and not against the exact analytical value of the functional. Adjoint based approaches also permit distinct identification of the contributions to this error arising from the three primary sources, namely the spatial/temporal resolution and the effect of partial convergence of the equations. This is different from common adaption or control schemes which are based on algebraic estimates of the local error within the solution, in that the global error in the functional can be estimated and adapted on.

This method is most mature for steady problems where researchers have used the error estimates to adaptively refine the grid[1–3] and put error bounds on solution accuracy.[4–6] Unsteady problems have received considerably less attention and the primary focus has been on performing adaption in time,[7–9] space[9,10] and convergence tolerances[7,8] individually. In our previous papers[11],[12] we combined all three types of error estimates into the same framework and presented results showing each of the three components can be separated into individual contributions. Furthermore, we demonstrated the three individual error estimates could be used to adapt the discretization and improve the efficiency of the time accurate solution.

*Ph.D. Candidate, email: bflynt@uwyo.edu

†Professor, AIAA Associate Fellow, email: mavripl@uwyo.edu

The present study builds on our previous works[11,12] by developing a methodology where the error originating from the three different sources can be compared against each other. Furthermore each error has a weighting assigned by the cost to refine the error thereby allowing the elimination of discretization error in the most efficient order. The methodology is demonstrated using an implicit cell-centered finite volume solver with gradient reconstruction and a Roe flux splitting scheme[13] to solve the Euler equations on an unstructured mesh up to 2nd order accuracy in both space and time. The following sections contain the formulations required for calculation of each of the error estimates, implementation details and test case results showing the benefits of the proposed adaption strategy using the adjoint based error estimates.

II. Governing Equations

II.A. Euler Equations

The Euler equations are the governing equations for inviscid flow where the dissipative phenomena of viscosity and thermal conductivity are ignored. The three-dimensional equations in Cartesian coordinates for a compressible flow minus any source terms are represented by a system of five non-linear partial differential equations.

$$\frac{\partial \rho}{\partial t} + \nabla \cdot (\rho \vec{V}) = 0 \tag{1}$$

$$\frac{\partial (\rho u)}{\partial t} + \nabla \cdot (\rho u \vec{V}) = -\frac{\partial P}{\partial x} \tag{2}$$

$$\frac{\partial (\rho v)}{\partial t} + \nabla \cdot (\rho v \vec{V}) = -\frac{\partial P}{\partial y} \tag{3}$$

$$\frac{\partial (\rho w)}{\partial t} + \nabla \cdot (\rho w \vec{V}) = -\frac{\partial P}{\partial z} \tag{4}$$

$$\frac{\partial}{\partial t}\left[\rho\left(e + \frac{V^2}{2}\right)\right] + \nabla \cdot \left[\rho\left(e + \frac{V^2}{2}\right)\vec{V}\right] = -\frac{\partial uP}{\partial x} - \frac{\partial vP}{\partial y} - \frac{\partial wP}{\partial z} + \rho \vec{f} \cdot \vec{V} \tag{5}$$

By examining the equations we see there are five equations but six unknowns (ρ, u, v, w, e, P). Therefore, in this paper an ideal gas is assumed and the equation of state relates the pressure to total energy by:

$$P = (\gamma - 1)\left[E - \frac{1}{2}\rho\left(u^2 + v^2 + w^2\right)\right] \tag{6}$$

To facilitate solution by computer the equations are manipulated into vector form by re-writing the system as:

$$\frac{\partial \vec{U}}{\partial t} + \frac{\partial \vec{F_x}}{\partial x} + \frac{\partial \vec{F_y}}{\partial y} + \frac{\partial \vec{F_z}}{\partial z} = 0 \tag{7}$$

where the vector of conserved quantities becomes:

$$\vec{U} = \begin{pmatrix} \rho \\ \rho u \\ \rho v \\ \rho w \\ \rho E \end{pmatrix} \tag{8}$$

and the Cartesian inviscid flux vectors reduced to:

$$\vec{F_x} = \begin{pmatrix} \rho u \\ \rho u^2 + P \\ \rho uv \\ \rho uw \\ \rho u\left(E + \frac{P}{\rho}\right) \end{pmatrix} \quad \vec{F_y} = \begin{pmatrix} \rho v \\ \rho vu \\ \rho v^2 + P \\ \rho vw \\ \rho v\left(E + \frac{P}{\rho}\right) \end{pmatrix} \quad \vec{F_z} = \begin{pmatrix} \rho w \\ \rho wu \\ \rho wv \\ \rho w^2 + P \\ \rho w\left(E + \frac{P}{\rho}\right) \end{pmatrix} \tag{9}$$

The Euler equations (Eqn. 7) are solved implicitly in time using the second order backwards difference formula (BDF2) with second order accurate spatial derivatives on an unstructured grid. This is done for each time step by introducing an implicit residual as

$$\vec{R^n}(\vec{U}^n, \vec{U}^{n-1}, \vec{U}^{n-2}, \vec{x}) = 0 \tag{10}$$

where $\vec{U}^n$ is the solution at the current time step, $\vec{U}^{n-1}$ and $\vec{U}^{n-2}$ are the solutions at the two previous time steps and $\vec{x}$ is the vector of grid coordinates. This system is linearized with respect to the unknown solution $\vec{U}^n$ and solved using Newton's method.

$$\begin{gathered}\left[\frac{\partial \vec{R}(\vec{U}^k,\vec{x})}{\partial \vec{U}^k}\right]\delta\vec{U}^k = -\vec{R}(\vec{U}^k,\vec{x}) \\ \vec{U}^{k+1} = \vec{U}^k + \delta\vec{U}^k \\ \vec{U}^n = \vec{U}^{k+1}, \delta\vec{U}^k \to 0\end{gathered} \tag{11}$$

II.B. Error due to Spatial Resolution

Consider an objective function L that is a quantity dependent on the flow solution U for all time steps of the simulation. For this case, the spatial error will be derived by using a Taylor series expansion of the true objective calculated on a fine grid $L_s(\vec{U}_s)$.

$$L_s(\vec{U}_s) = L_s(\tilde{U}_s) + \frac{\partial L_s(\tilde{U}_s)}{\partial \tilde{U}_s}\left(\vec{U}_s - \tilde{U}_s\right) + \cdots \tag{12}$$

where $L_s(\tilde{U}_s)$ is the fine grid objective computed using an approximate solution $\tilde{U}_s$. In general, L_s may be a time integrated objective and will depend on the entire time history of the solution $\vec{U}_s$. Therefore no time level superscripts exist with the understanding that unsuperscripted variables refer to the entire time history of the flow field.

To avoid the need for computing the true solution $\vec{U}_s$ on the fine mesh in the proceeding equation we will use a Taylor series expansion about the fine level residual equation

$$R_s(\vec{U}_s) = 0 = R_s(\tilde{U}_s) + \frac{\partial R_s(\tilde{U}_s)}{\partial \tilde{U}_s}\left(\vec{U}_s - \tilde{U}_s\right) + \cdots \tag{13}$$

where the unsuperscripted residual denotes the residuals over all time steps and all mesh cells of the simulation. Using this expression to substitute for the term $\left(\vec{U}_s - \tilde{U}_s\right)$ we can rewrite the objective as:

$$L_s(\vec{U}_s) \cong L_s(\tilde{U}_s) - \left[\frac{\partial L_s(\tilde{U}_s)}{\partial \tilde{U}_s}\right]\left[\frac{\partial R_s(\tilde{U}_s)}{\partial \tilde{U}_s}\right]^{-1} R_s(\tilde{U}_s) \tag{14}$$

Denoting the matrix product as the adjoint variable ${\Lambda_s}^T$

$${\Lambda_s}^T = -\left[\frac{\partial L_s(\tilde{U}_s)}{\partial \tilde{U}_s}\right]\left[\frac{\partial R_s(\tilde{U}_s)}{\partial \tilde{U}_s}\right]^{-1} \tag{15}$$

leads to the following expression for the linear approximation to the fine grid objective.

$$L_s(\vec{U}_s) \cong L_s(\tilde{U}_s) + {\Lambda_s}^T R_s(\tilde{U}_s) \tag{16}$$

The inner product given by the last term is taken over all mesh cells and all time steps of the simulation. The values in this expansion are obtained by constructing the approximate solution $\tilde{U}_s$ by interpolating a coarse grid solution onto the fine grid at each time step and evaluating $L_s(\tilde{U}_s)$ and $R_s(\tilde{U}_s)$. The difficulty lies in obtaining the adjoint value Λ_s^T. A direct calculation of this quantity costs as much as obtaining the true solution $\vec{U}_s$, in which case, we might as well evaluate $L_s(\vec{U}_s)$ directly. To alleviate this expense we

approximate the fine grid adjoint value by calculating the coarse grid adjoint denoted as Λ and interpolating onto the fine grid using a projection matrix (I_S^s) to arrive at the final linear approximation to the fine grid objective function.

$$\begin{gathered}\tilde{\Lambda}_s^T = I_S^s \Lambda^T \\ L_s(\vec{U}_s) \cong L_s(\tilde{U}_s) + \tilde{\Lambda}_s^T R_s(\tilde{U}_s)\end{gathered} \tag{17}$$

$$\underbrace{L_s(\vec{U}_s) - L_s(\tilde{U}_s)}_{\varepsilon_s} \cong I_S^s \Lambda^T R_s(\tilde{U}_s) \tag{18}$$

From this derivation we see that the term $\tilde{\Lambda}_s^T R_s(\tilde{U}_s)$ is a direct estimation of the linear error in our calculation from the lack of spatial resolution. Therefore, the elements with the largest value can be flagged for future refinement.

One term of this derivation, that cannot be directly evaluated and has yet to be solved for, is the adjoint variable on the coarse grid Λ^T. It is most easily solved by rearranging the matrix product expression in equation (15) to be:

$$\left[\frac{\partial R(\vec{U})}{\partial \vec{U}}\right]^T \vec{\Lambda} = -\left[\frac{\partial L(\vec{U})}{\partial \vec{U}}\right]^T \tag{19}$$

Notice the approximate solutions $\tilde{U}$ have been replaced with converged solutions $\vec{U}$ since this equation is solved using the coarse grid solution. The matrices in equation (19) represent each equation of every element on every time step therefore we can expand the equations into an expression where each row represents the solution to a single time step (shown here for BDF1).

$$\begin{bmatrix} \begin{bmatrix}\frac{\partial R(\vec{U})}{\partial \vec{U}}\\ \frac{\partial R(\vec{U})}{\partial \vec{U}}\end{bmatrix} & & & & \\ & \begin{bmatrix}\frac{\partial R(\tilde{U})}{\partial \tilde{U}}\\ \frac{\partial R(\vec{U})}{\partial \vec{U}}\end{bmatrix} & \begin{bmatrix}\frac{\partial R(\vec{U})}{\partial \vec{U}}\end{bmatrix} & 0 & \\ & & \ddots & \ddots & \\ & 0 & & \begin{bmatrix}\frac{\partial R(\vec{U})}{\partial \vec{U}}\end{bmatrix} & \begin{bmatrix}\frac{\partial R(\vec{U})}{\partial \vec{U}}\\ \frac{\partial R(\vec{U})}{\partial \vec{U}}\end{bmatrix} \end{bmatrix}^T \begin{bmatrix}\Lambda^1\\ \Lambda^2\\ \Lambda^3\\ \vdots\\ \Lambda^{n-1}\\ \Lambda^n\end{bmatrix} = - \begin{bmatrix}\begin{bmatrix}\frac{\partial L(\vec{U})}{\partial \vec{U}}\\ \frac{\partial L(\vec{U})}{\partial \vec{U}}\\ \frac{\partial L(\vec{U})}{\partial \vec{U}}\end{bmatrix}^T \\ \vdots \\ \begin{bmatrix}\frac{\partial L(\vec{U})}{\partial \vec{U}}\\ \frac{\partial L(\vec{U})}{\partial \vec{U}}\end{bmatrix}^T \end{bmatrix} \tag{20}$$

By carrying out the transpose operator the system of equations that must be solved to acquire the adjoint solution becomes evident.

$$\begin{bmatrix} \left[\frac{\partial R(\vec{U})}{\partial \vec{U}}\right]^T & \left[\frac{\partial R(\vec{U})}{\partial \vec{U}}\right]^T & & & \\ & \left[\frac{\partial R(\tilde{U})}{\partial \tilde{U}}\right]^T & \left[\frac{\partial R(\vec{U})}{\partial \vec{U}}\right]^T & 0 & \\ & & \ddots & \ddots & \\ & & & \left[\frac{\partial R(\vec{U})}{\partial \vec{U}}\right]^T & \left[\frac{\partial R(\vec{U})}{\partial \vec{U}}\right]^T \\ & 0 & & \left[\frac{\partial R(\vec{U})}{\partial \vec{U}}\right]^T & \left[\frac{\partial R(\vec{U})}{\partial \vec{U}}\right]^T \\ & & & & \left[\frac{\partial R(\vec{U})}{\partial \vec{U}}\right]^T \end{bmatrix} \begin{bmatrix}\Lambda^1\\ \Lambda^2\\ \Lambda^3\\ \vdots\\ \Lambda^{n-1}\\ \Lambda^n\end{bmatrix} = - \begin{bmatrix}\begin{bmatrix}\frac{\partial L(\vec{U})}{\partial \vec{U}}\\ \frac{\partial L(\vec{U})}{\partial \vec{U}}\\ \frac{\partial L(\vec{U})}{\partial \vec{U}}\end{bmatrix}^T \\ \vdots \\ \begin{bmatrix}\frac{\partial L(\vec{U})}{\partial \vec{U}}\\ \frac{\partial L(\vec{U})}{\partial \vec{U}}\end{bmatrix}^T \end{bmatrix} \tag{21}$$

From this transposed system of equation it is evident that the unsteady adjoint variables can be solved by sweeping backwards through all time steps of the solution solving for each adjoint solution using the flow and adjoint solutions from one step in the future of the time series.

II.C. Error due to Temporal Resolution

The use of the adjoint variable to identify errors in temporal resolution follows a similar path to that of spatial error (Section II.B). We start by using a Taylor series expansion of the true time integrated objective

calculated using a small time step $L_t(\vec{U}_t)$

$$L_t(\vec{U}_t) = L_t(\tilde{U}_t) + \frac{\partial L_t(\tilde{U}_t)}{\partial \tilde{U}_t}\left(\vec{U}_t - \tilde{U}_t\right) + \cdots \tag{22}$$

where $L_t(\tilde{U}_t)$ is the small time step objective calculated with an approximate solution. The need to calculate the true small time-step solution $\vec{U}_t$ is eliminated by using a Taylor series expansion of the small time step residual equation:

$$R_t(\vec{U}_t) = 0 = R_t(\tilde{U}_t) + \frac{\partial R_t(\tilde{U}_t)}{\partial \tilde{U}_t}\left(\vec{U}_t - \tilde{U}_t\right) + \cdots \tag{23}$$

Following the steps previously presented we can arrive at the expression for the small time step objective using an approximate small time step solution $\tilde{U}_t$ obtained by projecting a coarse time step solution to the finer time step space:

$$L_t(\vec{U}_t) \cong L_t(\tilde{U}_t) + {\Lambda_t}^T R_t(\tilde{U}_t) \tag{24}$$

The requirement for the small time step adjoint solution (Λ_t) is relaxed by approximating it using the adjoint solution from a coarse time step solution. This results in an equation that is a linear approximation to the small time step objective using only solutions from the coarse time step simulation.

$$L_t(\vec{U}_t) \cong L_t(\tilde{U}_t) + \tilde{\Lambda}_t^T R_t(\tilde{U}_t) \tag{25}$$

Again we can re-arrange the terms to arrive at a direct estimation of the linear error in our calculation, this time from a lack of resolution in the time domain.

$$\underbrace{L_t(\vec{U}_t) - L_t(\tilde{U}_t)}_{\varepsilon_t} \cong \tilde{\Lambda}_t^T R_t(\tilde{U}_t) \tag{26}$$

II.D. Error due to Partial Convergence

In the former sections we assumed the solution on the coarse space and time domain was obtained by full convergence of the flow equations at each implicit time step resulting in a coarse level residual equation that evaluates to zero. If the flow equations were only partially converged at each time step, the fully converged objective can be linearly approximated using the the Taylor series expansion:

$$L(\vec{U}) = L(\tilde{U}_c) + \frac{\partial L(\tilde{U}_c)}{\partial \tilde{U}_c}\left(\vec{U} - \tilde{U}_c\right) + \cdots \tag{27}$$

where the objective $L(\tilde{U}_c)$ is obtained through partially converging the flow equations to an approximate solution $\tilde{U}_c$. Again, we use the expansion of the residual equation to eliminate the need for the fully converged solution $\vec{U}$ and introduce an approximate adjoint variable into the equation.

$$L(\vec{U}) \cong L(\tilde{U}_c) + \tilde{\Lambda}_c^T R(\tilde{U}_c) \tag{28}$$

The approximate adjoint $\tilde{\Lambda}_c$ is obtained by partially converging the adjoint equations using the same mesh and time step distribution as for the analysis problem. The contribution to the integrated linear error (ε_c) for the partially converged solution can be obtained by re-arranging the equation.

$$\underbrace{L(\vec{U}) - L(\tilde{U}_c)}_{\varepsilon_c} \cong \tilde{\Lambda}_c^T R(\tilde{U}_c) \tag{29}$$

It is of interest to note the residual $R(\tilde{U}_c)$ is non-zero only because the flow equations are not fully converged. If the system is fully converged, the residual is zero and the approximate objective is no longer approximate and becomes the true objective.

II.E. Combined Error

Our goal is to estimate the total error in a time dependent simulation objective and to determine the respective contributions to the total error from spatial discretization, temporal discretization, and incomplete convergence in order that adaptive refinement methods may be used to efficiently reduce the error. Using a Taylor series expansion, we can estimate the exact objective L_{st} computed on a fine time step and fine mesh with full convergence using an approximate solution $\tilde{U}_{cst}$ computed on a coarser mesh with a larger time step with partial convergence as:

$$L_{st}(\vec{U}_{cst}) = L_{st}(\tilde{U}_{cst}) + \frac{\partial L_{st}(\tilde{U}_{cst})}{\partial \tilde{U}_{cst}} \left(\vec{U}_{cst} - \tilde{U}_{cst}\right) + ... \tag{30}$$

In the above expression, $\tilde{U}_{cst}$ is obtained from the partially converged approximate solution $\tilde{U}_c$ computed on a coarser grid with a larger time step, projected onto the finer mesh and time step space. Although the above equation could be used to obtain an expression for the total error, it does not provide a mechanism for separating out the various error components. However, since the adjoint error estimation procedure relies on a linearization, we can assume these different error components are additive. Therefore, the error in the solution may be written as:

$$\vec{U}_{cst} - \tilde{U}_{cst} = \vec{U}_s - \tilde{U}_s + \vec{U}_t - \tilde{U}_t + \vec{U} - \tilde{U}_c \tag{31}$$

Here $\vec{U}_s - \tilde{U}_s$ corresponds to the difference between the solution computed on the fine grid and the approximate coarse level solution $\tilde{U}_c$ projected onto the fine grid with a fixed time step value. Similarly, $\vec{U}_t - \tilde{U}_t$ corresponds to the change in the solution when the time step is refined, with all other simulation parameters (mesh size, convergence tolerance) remaining fixed, and $\vec{U} - \tilde{U}_c$ corresponds to the change in the coarse mesh and time step solution that would be observed if full convergence was enforced at each time step. Clearly, interactions between these various sources of error will exist in actual simulations, but these will be higher order non-linear error interactions that cannot be accounted for in an adjoint formulation. Furthermore, as the errors are reduced, the additive assumption will become asymptotically more exact. Inserting the above expression into equation (33), and making use of an adjoint solution computed on the coarse space and time discretization level at the partially converged state leads to the following expression for the objective error:

$$L_{st}(\vec{U}_{cst}) \cong L_{st}(\tilde{U}_{cst}) + \tilde{\Lambda}_s^T R_s(\tilde{U}_s) + \tilde{\Lambda}_t^{\ T} R_t(\tilde{U}_t) + \tilde{\Lambda}_c^T R(\tilde{U}_c) \tag{32}$$

where the adjoint variables $\tilde{\Lambda}_s$, and $\tilde{\Lambda}_t$ correspond to the approximate coarse level adjoint $\tilde{\Lambda}_c$ projected onto the fine mesh and fine time step domain, respectively. The various residual operators R, R_s and R_t correspond to the space-time residuals evaluated on the coarse mesh and time step domain, the fine mesh domain (with coarse time step), and the fine time step domain (with coarse mesh), respectively. The one remaining term that is not readily available is the fine time and space approximate objective value $L_{st}(\tilde{U}_{cst})$. Instead of separately constructing this term it is formed using a linear combination of the terms already used for each individual error correction. To do this we let the variables $(\delta s, \delta t)$ represent the difference between the individual fine domain objective values calculated using approximate solutions and the coarse approximate objective value $L(U_c)$ already calculated.

$$\begin{aligned} \delta s &= L(U_c) - L_s(\tilde{U}_s) \\ \\ \delta t &= L(U_c) - L_t(\tilde{U}_t) \end{aligned} \tag{33}$$

Using these two values we can linearly predict what the approximation would be for the combined approximation $L_{st}(\tilde{U}_{cst})$:

$$L_{st}(\tilde{U}_{cst}) = L(U_c) - \delta s - \delta t \tag{34}$$

Inserting this approximation into the combined linear approximation from earlier (Eqn. 32) we get the final form of the combined approximation.

$$L_{st}(\vec{U}_{cst}) \cong L_s(\tilde{U}_s) + L_t(\tilde{U}_t) - L(U_c) + \tilde{\Lambda}_s^T R_s(\tilde{U}_s) + \tilde{\Lambda}_t^T R_t(\tilde{U}_t) + \tilde{\Lambda}_c^T R(\tilde{U}_c) \tag{35}$$

By re-arranging we can isolate the terms to again produce an estimate of integrated linear error

$$\underbrace{L_{st}(\vec{U}_{cst}) - L_s(\tilde{U}_s) - L_t(\tilde{U}_t) + L(U_c)}_{\varepsilon_{cst}} \cong +\tilde{\Lambda}_s^T R_s(\tilde{U}_s) + \tilde{\Lambda}_t^T R_t(\tilde{U}_t) + \tilde{\Lambda}_c^T R(\tilde{U}_c) \tag{36}$$

III. Implementation Details

III.A. Mesh Refinement

In the current work the conformal nature of a mixed element grid is maintained by using element subdivision for refinement and un-refinement.[14] This is accomplished using a set of allowable subdivision types for each element as shown in Figure 1. An element that is flagged for refinement is isotropically split into 4 elements. The non-flagged neighboring element is partially subdivided using an allowable pattern to keep the grid conformal. On subsequent steps, if one of the elements resulting from a partial subdivision is flagged for refinement, the partial element is collapsed back into the parent element which is then isotropically split into 4 smaller elements. This method of refinement limits the formation of sliver elements within the grid and lends itself to a simple tree data structure which can be unrolled to un-refine elements within the grid. Furthermore, no smoothing of the adaptive grid is performed so that un-refinement of elements will result in the starting grid.

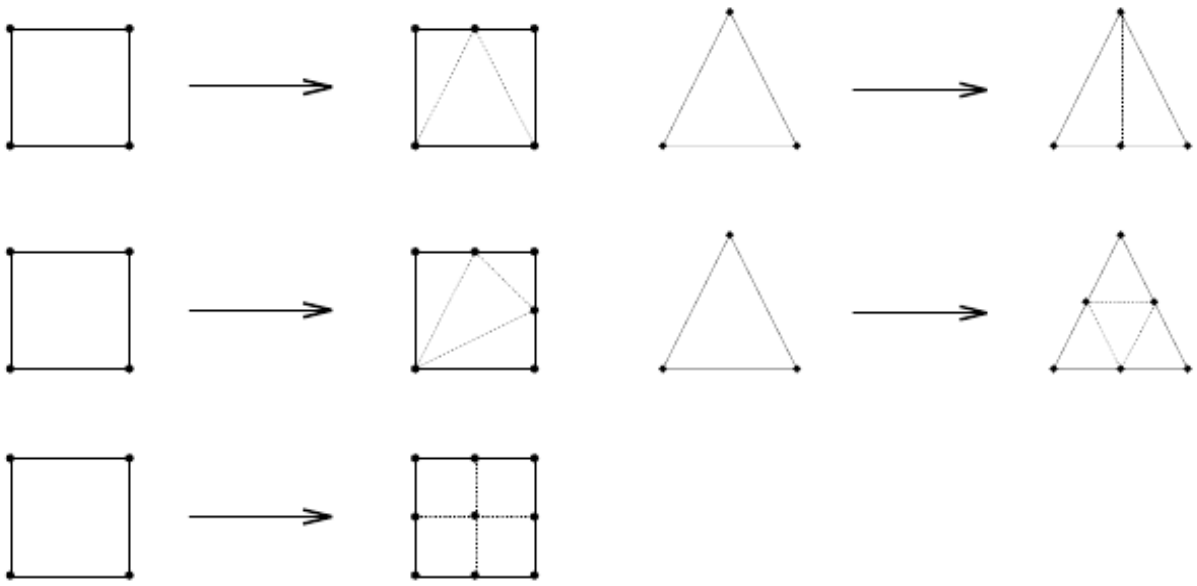

Figure 1. Allowable subdivision patterns for triangle and quadrilateral cells

III.B. Spatial Projection

To project the solutions to grids of different spatial refinement levels a general interpolation scheme is used based on compact radial basis functions.[15] Specifically, the compact functions of Wendland[16] are used with an additional polynomial constraint to ensure 2nd order accuracy for all cases. The approximations have the general form

$$s(\vec{x}) = \sum_{j=1}^{N} \alpha_j \phi\left(\vec{x} - \vec{x_j}\right) + p(\vec{x}) \tag{37}$$

where α_j is the weight of the known value at element j, $\phi\left(\vec{x} - \vec{x_j}\right)$ is the radial function evaluated between the two centers and $p(\vec{x})$ is a polynomial of the required order of accuracy. The required number of nearest neighbors is found using an alternating digital tree[17] followed by a linear search over the remaining candidates.

III.C. Temporal Projection

For the temporal refinement a 2-to-1 refinement pattern is enforced where a single time interval is split into 2 equal intervals. In addition, any adjacent time intervals that vary more than 2 times the size of the current time interval will be flagged for splitting to ensure no large jumps in the time step size exist.

To perform the temporal projections from coarse to fine time steps linear interpolation is used to arrive at the intermediate solution.

$$\begin{aligned} \tilde{\Lambda}_t^{n-\frac{1}{2}} &= \tfrac{1}{2}\left(\Lambda_T^n + \Lambda_T^{n-1}\right) \\ \tilde{\vec{U}}_t^{n-\frac{1}{2}} &= \tfrac{1}{2}\left(\vec{U}_T^n + \vec{U}_T^{n-1}\right) \end{aligned} \tag{38}$$

III.D. Solver Details

The adaptive solver consists of three parts namely the forward flow solver, backward adjoint solver and a time/grid adaption module. The projection operations are done within the adaption module while the error computations are built into the adjoint solver. The simulation starts by using the adaption module to write out all grids for all time steps to file for the current (first) sweep. The flow solver then performs the forward integration in time by reading in the grid for the new time step and projecting the previous solution to it for the initial condition and time history information. Once the flow solution has been solved on all time steps, the adjoint solver reads in the flow solution and associated grid and performs the backwards sweep in time to compute the adjoint variables at each time step and saves them to file. Next the adaption module will create an isotropically refined grid for each of the fine domain time steps and perform the necessary interpolation onto each. These are saved to disk so that the flow solver followed by the adjoint solver can read them in and calculate the fine grid approximate objective solution, evaluate the residual and form the contribution to the error for each grid element of each time step and save them to file. In addition, smoothing of the approximate flow or adjoint solution can be done by their respective solver during this part of the process. For this research only the spatially projected approximate adjoint was smoothed by one order of magnitude in the L_2 norm for each time step. Finally, the individual error contributions are read from disk one time step at a time and the error is used to flag time intervals and grid elements for refinement based on the desired refinement criteria selected. If another more refined simulation is desired the adaption module will again generate and save out a newly refined grid for each adapted time step and the process will repeat again.

IV. Adaption Strategy and Threshold Selection

IV.A. Global Adaption Strategy

In our previous work[12] we examined the effect of equally distributing the error relevant to a functional within the spatial, temporal and algebraic discretizations independent of each other. This was done with no consideration for the cost of each refinement operation and therefore only targeted the elements, time steps or steps convergence criteria with the largest error for refinement. A more optimal solution, and the one presented in this paper, is to incorporate the cost of each refinement operation into the selection of which errors to refine. By doing this, all errors in the simulation can be compared, irrespective of the discretization source of the error (algebraic, spatial or temporal), and refined at a more optimal rate. The resulting solution is the one with the lowest amount of error in the function of interest for a given computational effort.

To accomplish this we first develop a cost to refine the individual spatial, temporal and algebraic errors. A common measure that all refinement types can be measured by is the number of additional non-linear iterations on an element each refinement will require. Therefore, we define one unit of cost to be one iteration of the non-linear solver on one element for one time step. By choosing this definition the type of linear solver, and its convergence rate, used within the non-linear solver is not important as long as two additional conditions are met. The first condition is all equations of all time steps must be solved using the same method. Secondly, the linear solver is run a fixed number of iterations within each non-linear iteration (a common practice).

To estimate the number of additional non-linear iterations that will be required from a refinement we start by looking at the non-linear method used to solve the system of equations. The convergence of the non-linear solver can be written in terms of the starting error ξ_n, the convergence rate q and the asymptotic error constant μ to arrive at the error after one iteration ξ_{n+1} as shown below.

$$\lim_{n\to\infty} \frac{|\xi_{n+1}|}{|\xi_n|^q} = \mu \tag{39}$$

By expanding the equation into a series and combining terms into a summation the equation can be re-written to directly solve for the error after k additional iterations.

$$\xi_{n+k} = \xi_n^{q^{(k)}} \mu^{\sum_{i=1}^{k} q^{(k-i)}} \tag{40}$$

If we assume the asymptotic error constant is one ($\mu = 1$) we can eliminate the summation in the exponent and re-arrange the equation to solve for the number of iterations k that it will take to achieve an error value.

$$k = \frac{\log(\xi_{n+k})}{q \log(\xi_n)} \tag{41}$$

In this research the flow solver uses Newton's method for the non-linear solver which has a quadratic convergence rate ($q = 2$) but because the linear solver is not fully converged at each non-linear iteration the rate is somewhat lower. For the results section of this paper a convergence rate of 1.6 ($q = 1.6$) is used based on our experience with this specific code and solution strategy.

IV.A.1. Cost to Increase Convergence Tolerance

The cost to increase the convergence tolerance on a single element is simply the additional non-linear iterations Δk, calculated using equation 41, required to go from the current convergence criteria to the next refined one which in this paper is a factor of 10. In addition, the flow solver has the restriction that all elements must be iterated on equally within a time step so the cost to refine the convergence tolerance on a single time step, CoR_c, is the additional non-linear iteration times the number of elements in the grid as shown in Equation 42.

$$CoR_c = (\#\text{ of Elements within Time Step}) \times \Delta k \tag{42}$$

IV.A.2. Cost to Refine Time Step

To refine a time step requires the entire solution on a second mesh with an equal number of elements to the time step being refined. In addition, the number of additional non-linear iterations k calculated by 41 must be calculated using the starting error ($L_2(R) \approx 0.1$) and the final convergence tolerance requested. This is represented in Equation 43 where one can see that refining a time step is a costly operation.

$$CoR_t = (\#\text{ of Elements within Time Step}) \times k \tag{43}$$

IV.A.3. Cost to Refine Spatial Element

To determine the cost to refine a single spatial element, CoR_s, we must return to the section on mesh refinement III.A. In the allowable subdivisions of Figure 1 we can see that each refinement of a grid element (quad or triangle) results in 3 additional elements. This number ignores any splits of neighboring elements to remain conformal but this effect will be accounted for in selecting which elements to refine later. Each one of the 3 additional elements must proceed through k non-linear iterations calculated by using the starting error ($L_2(R) \approx 0.1$) and the final convergence tolerance requested at that time step

$$CoR_s = (\#\text{ of New Elements} = 3) \times k \tag{44}$$

IV.A.4. Selection of Discretization to Refine

The final piece in deciding what to refine and where is combining the resulting discretization error with the cost to refine each. This is done by calculating the algebraic and temporal error at every time step along with the spatial error for each element on every time step. The absolute value of each error is divided by the cost to perform the refinement using the formulas derived in the proceeding sections. In this way, each of the errors can be compared against each other, irrespective of the discretization source, to determine which refinement opportunity is predicted to result in the most error reduction for a given amount of cost. An ordering of these error per cost values from largest to smallest identifies which refinements are most beneficial to make but does nothing to determine how many to refine on a given refinement sweep.

To determine how many refinements are needed we begin by calculating the total amount of error that we wish to eliminate on this refinement sweep. To do this we create a value of the error excess ε_{ex} by subtracting the actual combined error ε_{cst}, calculated using equation 36, from the user requested error tolerance Tol_R as shown in Equation 45.

$$\varepsilon_{ex} = \varepsilon_{cst} - Tol_R \tag{45}$$

Now returning to our sorted list of error per cost values for each refinement opportunity we proceed, in descending order, to flag errors for refinement until the cumulative sum of the flagged errors meets or exceeds the value of the error excess on each refinement sweep. The value of error per cost where the cumulative sum of the refined error exceeds that of the error excess will be referred to as the error per cost threshold value. The value of the error per cost threshold in and of itself has no real importance but it does prove to be graphically useful when displaying the discretization errors and the cost to refine each in the results section (Section V) of the paper. In this context, the error per cost threshold is the dividing line where all errors above the threshold are flagged for refinement and all discretization errors below are left unrefined.

To choose how many opportunities to refine at any refinement step the simplest choice would be all opportunities that exceed the error per cost threshold value. Doing this at every refinement interval would insure that eventually the global error relevant to the functional would become less than the user specified tolerance. Previous work has shown that refining a large percentage of the error early in the refinement process results in the total degrees of freedom in the simulation quickly getting large. A smarter choice and one that has already been explored for steady state problems by Nemec, Aftosmis and Wintzer[18] is to gradually increase the amount of error refined on any one refinement sweep. In this scenario the error excess ε_{ex} is initially divided by a value larger than 1 so that a small fraction of the highest error per cost opportunities are flagged for refinement then on subsequent steps the error excess is divided by decreasing values until eventually the full error excess is refined upon. This has the effect of only refining the high error while not increasing the computational cost significantly, and then in the last couple refinement sweeps heavily refining to meet the desired error goals. This modified error excess equation is shown with the included error fraction λ parameter included in equation 46.

$$\varepsilon_{ex} = \frac{\varepsilon_{cst} - Tol_R}{\lambda} \tag{46}$$

V. Results

V.A. Convecting Vortex

The first test of the new algorithm involves the time integrated density inside a region of the flow field as a 2-D inviscid isentropic vortex[19,20] convects through uniform flow. In this case the mean flow density (ρ_∞), velocity (u_∞,v_∞), pressure (P_∞), and temperature (T_∞) are taken to be free stream values. We set these flow parameters to be (ρ_∞,u_∞,u_∞,P_∞,T_∞) = (1.0,0.5,0.0,1.0,1.0) with characteristic boundary conditions[21] on all free stream boundaries. At the initial time the flow is perturbed by an isentropic vortex $(\delta u, \delta v, \delta T)$ centered at $(x_o, y_o) = (-15.0, 0.0)$ with the form:

$$\begin{aligned} \delta u &= -\frac{V_m}{R_c}(y - y_o)\, e^{\left[\frac{1}{2} - \left(\frac{r}{2R_c}\right)^2\right]} \\ \delta v &= \frac{V_m}{R_c}(x - x_o)\, e^{\left[\frac{1}{2} - \left(\frac{r}{2R_c}\right)^2\right]} \\ \delta T &= -\frac{V_m^2(\gamma-1)}{2\gamma R_c^2} e^{\left[1 - \left(\frac{r}{R_c}\right)^2\right]} \end{aligned} \tag{47}$$

where V_m is the maximum perturbed velocity, R_c is the distance r that this maximum velocity occurs at from the vortex center and $\gamma = 1.4$ is the ratio of specific heats of the fluid. From the relationship for an ideal gas and assuming isentropic flow the density is found for every point in the domain as:

$$\rho = T^{1/(\gamma-1)} = (T_\infty + \delta T)^{1/(\gamma-1)} \tag{48}$$

In this case the strength of the vortex and size were specified by assigning the values of 0.2 to the maximum velocity ($V_m = 0.2$) at a core radius of 0.5 ($R_c = 0.5$) grid units.

To properly evaluate the results a single quantity is needed as the objective function and for this we define the objective (Eqn. 49) to be the time and space integrated density inside a square box centered about the origin $(x, y) = (0, 0)$ with equal length sides of 2 grid units. The vortex is convected for 60 non-dimensional time units allowing it to travel from the starting position of $(x_o, y_o) = (-15.0, 0.0)$ to the final position of $(x, y) = (15.0, 0.0)$ and being centered in the integrated region exactly half way through the computation. A portion of the coarsest grid (1176 elements) with the prescribed initial condition and highlighted integration region is shown in Figure 2. The exact analytic solution to this problem was found using a common computer math program to be 239.52558800471 which we will compare against as the true analytic solution in subsequent analysis.

$$L(U) = \int_0^{60} \int_{-1}^{1} \int_{-1}^{1} \rho \, \mathrm{d}x\mathrm{d}y\mathrm{d}t \tag{49}$$

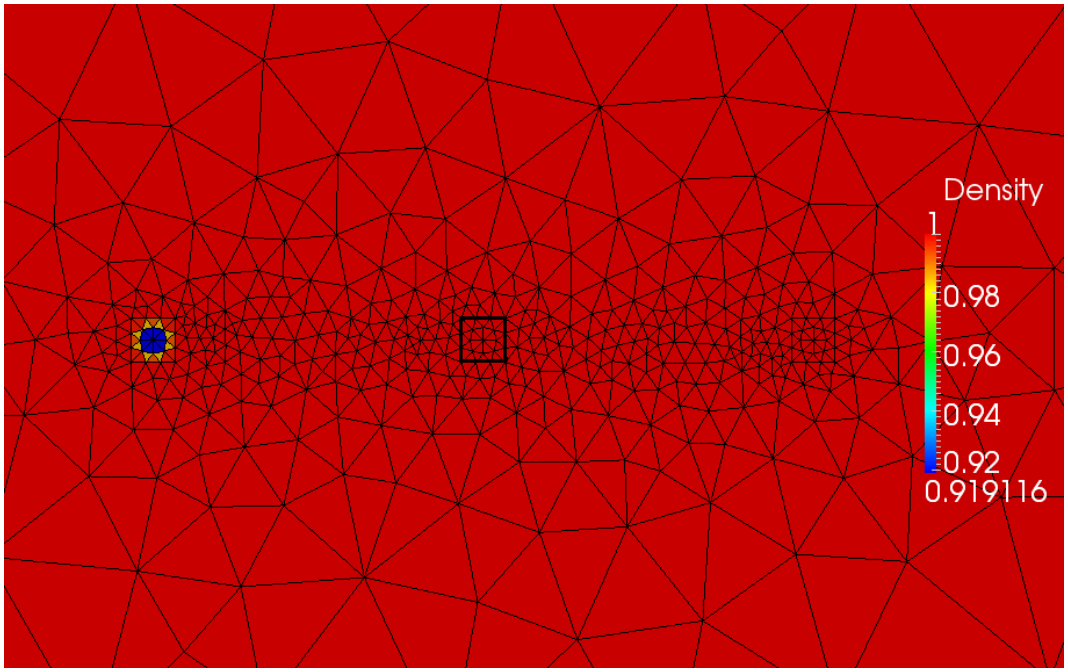

Figure 2. Initial density distribution with highlighted integrated region for grid of 1176 elements

For this problem the vortex is modeled with a starting discretization of 1176 grid elements over 16 equal spaced time steps converged 1 order in magnitude with the final relative error in the objective set to $Tol_{Global} <= 10^{-6}$. The simulation is allowed to progress through the refinement algorithm presented in Section IV.A.4 for 6 cycles. The error fraction parameter λ is decreased on each successive refinement sweep from 16, 8, 4, 2 and then fixed at 1 until the error tolerance is achieved. A plot of the error versus cost for each refinement opportunity on each sweep is plotted in Figure 3 along with a line showing the error per cost threshold. Recall that each refinement opportunity, irrespective of discretization source, with values of error above the threshold value for a given cost are flagged for refinement on the next iteration of the algorithm. The final plot at the completion of sweep 6 does not have a line representing the error per cost threshold because the requested error tolerance has been met and no further refinement is needed. To see where and how many refinements are taking place, plots of the adapted convergence tolerance, time step size and number of mesh elements at each time step are displayed in Figures 4, 5 and 6 for each cycle of the algorithm.

The plots of the convergence tolerance (Fig.4) and error versus cost (Fig. 3) show the requested residual reduction of 10^{-1} is sufficient for the first 3 refinement cycles but after using the same convergence tolerance on the 3rd cycle the error per cost of the convergence tolerance refinements are some of the largest of all refinement opportunities. As a result, the convergence tolerance on every time step is above the line for the error per cost threshold which indicates those opportunities will be adjusted for an increased convergence

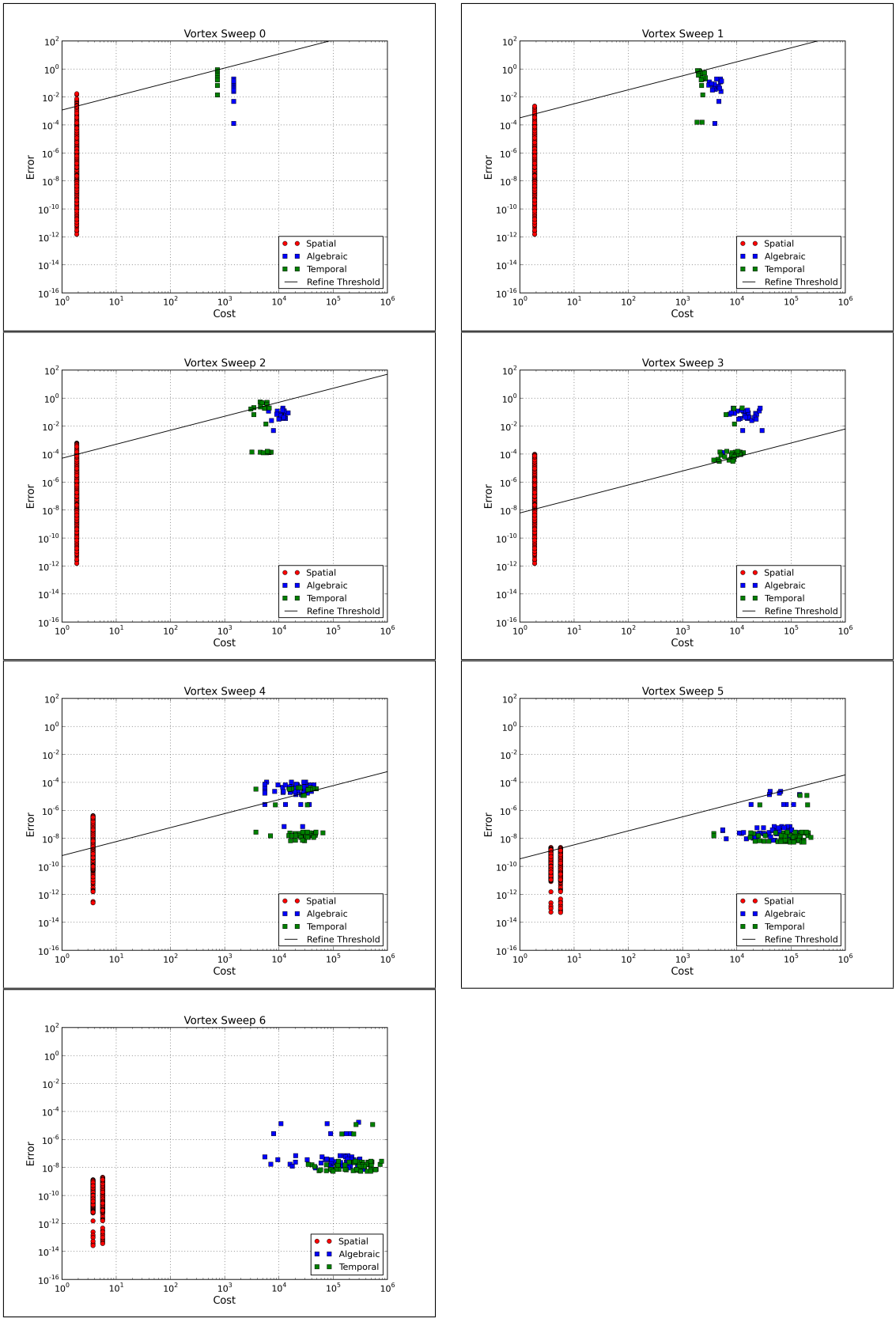

Figure 3. Error vs. Cost plots for the convecting vortex over 6 different refinement sweeps

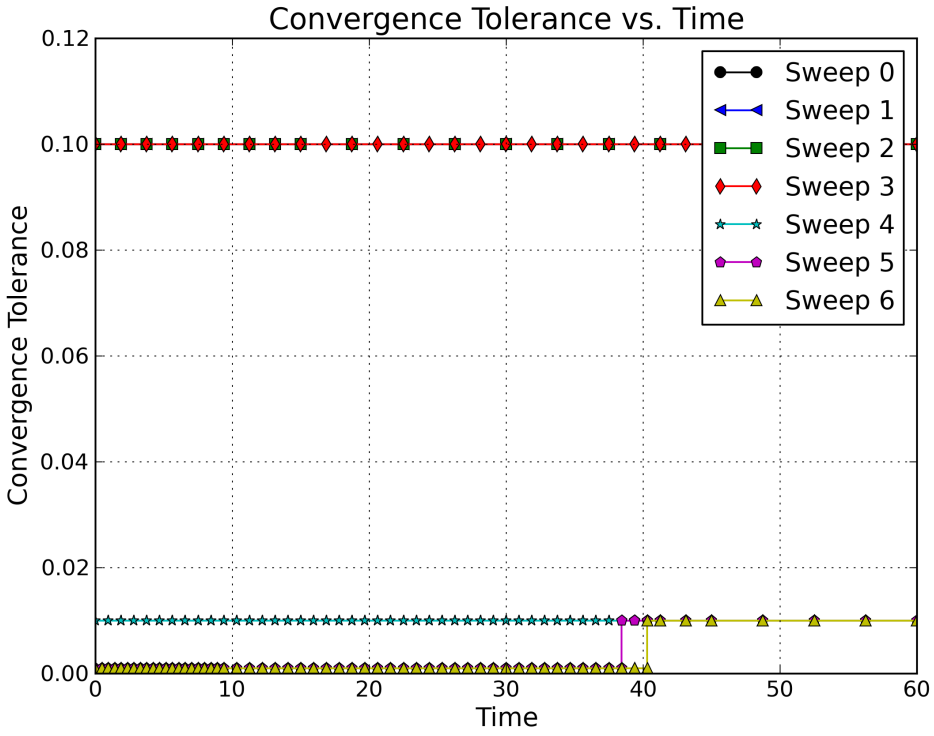

Figure 4. Convergence Tolerance as a function of time for each refinement sweep of the convecting vortex

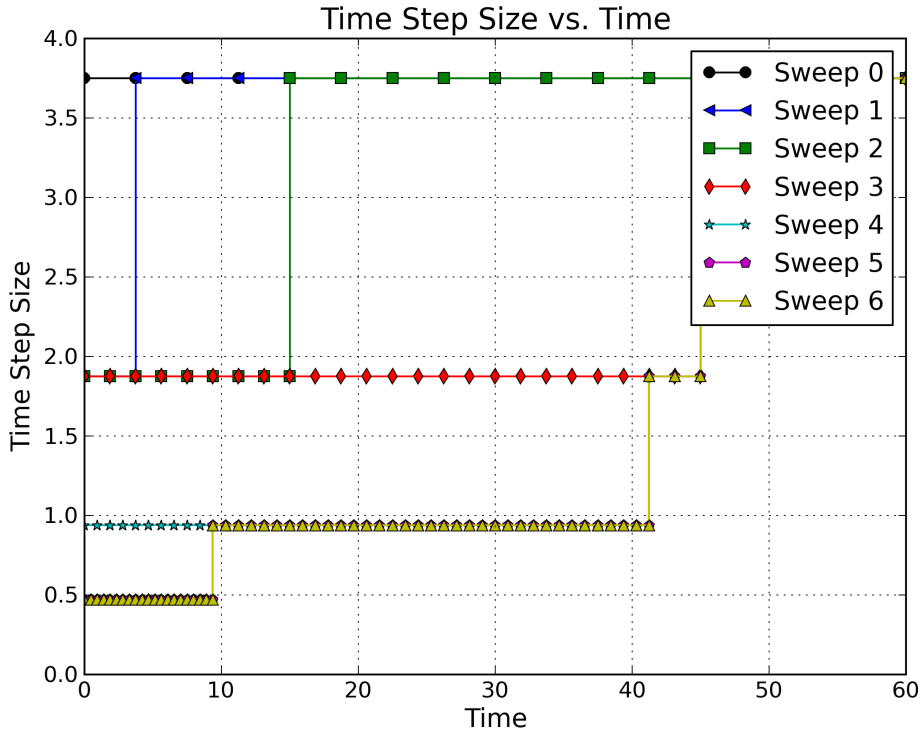

Figure 5. Time step size as a function of time for each refinement sweep of the convecting vortex

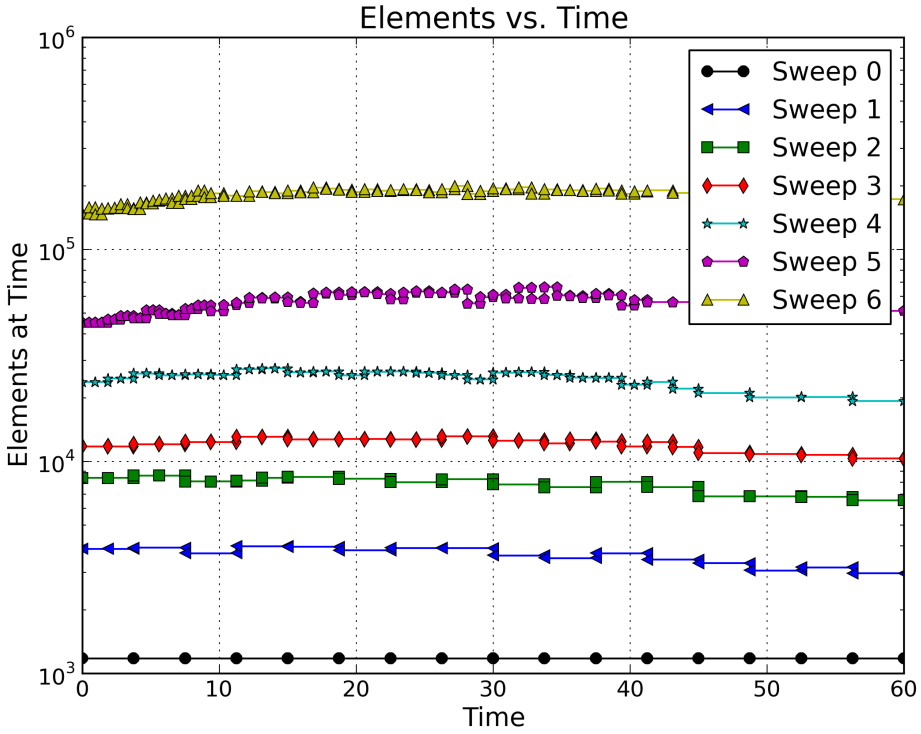

Figure 6. Elements as a function of time for each refinement sweep of the convecting vortex

criteria (i.e. lower final residual) on the subsequent refinement sweep. On the next refinement sweep (Sweep 4) the initial time steps still have comparatively large error per cost values and again exceed the error per cost threshold for refinement. These time steps are again flagged for refinement further reducing the requested residual reduction to 10^{-3} on the initial time steps while the time steps later in the simulation remain at a residual reduction of 10^{-2}. This difference in convergence criteria for different time steps of the time accurate simulation become important in subsequent element refinements as seen in the figures for Sweep 5 & 6. In this plot, the elements now have two different costs to refine them depending on the convergence criteria of the time step in which they reside. The elements within grids for time steps with a smaller convergence tolerance of 10^{-3} now take more non-linear iterations to converge than those in the time steps with a tolerance of 10^{-2} therefore they are slightly more expensive to refine.

The time step plot (Fig. 5) shows a different pattern, where the temporal error of the first time step, using only 16 time steps to model the full time accurate simulation, was found to exceed the error threshold on the first sweep. This time step was split in half resulting in two steps to cover the same amount of simulation time while leaving the remaining 15 steps at the same time step size. On the subsequent refinement sweeps (Sweeps 1-5) the initial time steps are repeatedly flagged for having temporal errors which exceed the error threshold eventually resulting in 3 different time step sizes being used to cover the full 60 units of non-dimensional time of the simulation. It is of interest to note, that a time step refinement does not result in additional cost to refine a grid element as was the case for a convergence tolerance refinement. This is because the refinement of a time step produces an additional grid with the same convergence tolerance as the original time step that was split. Therefore, there are now two times as many grid elements to represent the same amount of non-dimensional time, but they are all the same cost to refine as the original time step that was split.

The spatial refinement plots (Fig. 6) show yet a different pattern where every refinement sweep has some elements that exceed the error threshold. In each refinement sweep the elements that exceed the threshold are flagged for refinement which results in the grid for each time step having a different number of elements. This is reflected in the spread of the cost to refine each time step or convergence tolerance. Initially, all time

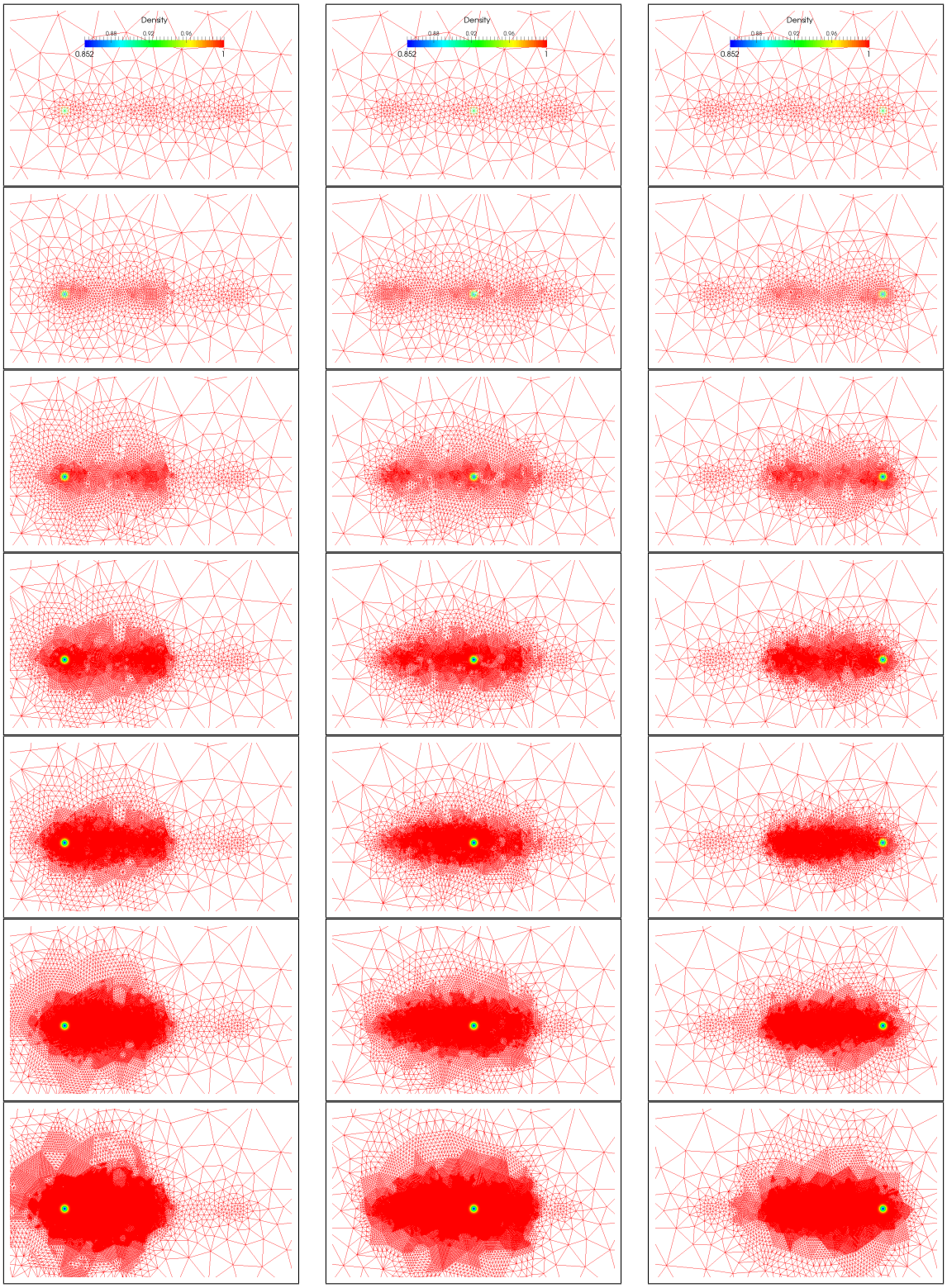

Figure 7. Convecting vortex and adapted grid at time = 0, 30, and 60 for 6 different refinement sweeps

step refinements or convergence tolerance refinements have equal cost since the initial grid (i.e. same number of elements) is used for all time steps of the simulation. On subsequent sweeps the number of elements within a time steps grid increases unevenly and the cost to split a time step or increase the convergence tolerance on a step increases a different amount depending on the number of elements within the grid (see Equations 42 & 43).

Images of the refined region within the computational mesh at non-dimensional times of t = 0, 30 and 60 are shown in Figure 7 for each sweep of the algorithm. The mesh refinement patterns clearly show refinement from the location of the vortex to the region the time integrated functional is integrated over. This feature is important and seen as a major advantage over simpler feature tracking algorithms such as gradient based refinement which would only refine around the high gradient of the vortex.

V.B. Triangle in Shock Tube

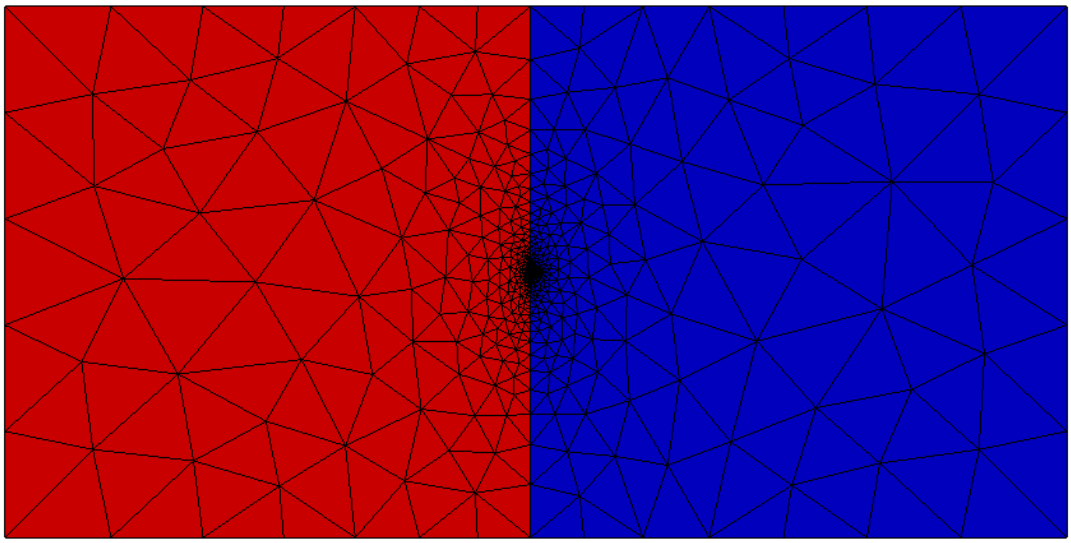

Figure 8. Full gridded domain of the shock tube problem with triangle. Red region is the high pressure side while the blue is the driven side

The second test case used to demonstrate the algorithm involves the time integrated lift on a triangle positioned within the driven side of a 2-D shock tube as seen in Figures 8 and 9. The triangle is equilateral with unit length sides and the leading edge is positioned 10 grid units away from the diaphragm location. The value of the integrated lift should be nearly zero and only affected by asymmetric vortex shedding off the back of the triangle after the shock passes. We set the initial flow parameters in the high pressure region to be (ρ_{up},u_{up},v_{up},P_{up}) = (1.0,0.0,0.0,2.5) and in the low pressure region to be (ρ_{dn},u_{udn},v_{dn},P_{dn}) = (0.125,0.0,0.0,0.25) with frictionless walls on all sides. At time zero ($t = 0$) the diaphragm is broken and the simulation is run for 26 non-dimensional time units. In this amount of time the shock wave will have impacted the triangle and traveled approximately 30 grid units past the leading edge if the triangle did not affect the flow. The starting discretization consists of 925 elements over 16 equal spaced time steps converged 1 order in magnitude with the final error in the objective set to $Tol_R = 10^{-2}$. The simulation is allowed to progress through the refinement algorithm for 4 cycles with the threshold values decreasing at each successive loop from 4, 2 and then fixed at 1 until the error tolerance is achieved. The resulting error versus cost for each refinement opportunity on each sweep is plotted in Figure 10 along with a line showing the resulting error per cost threshold. Again, recall that each refinement opportunity, irrespective of discretization source, with values of error above the threshold value for a given cost are flagged for refinement on the next iteration of the algorithm. The final plot at the completion of sweep 4 does not have a line representing the error per cost threshold because the requested error tolerance has been met and no further refinement is needed. To

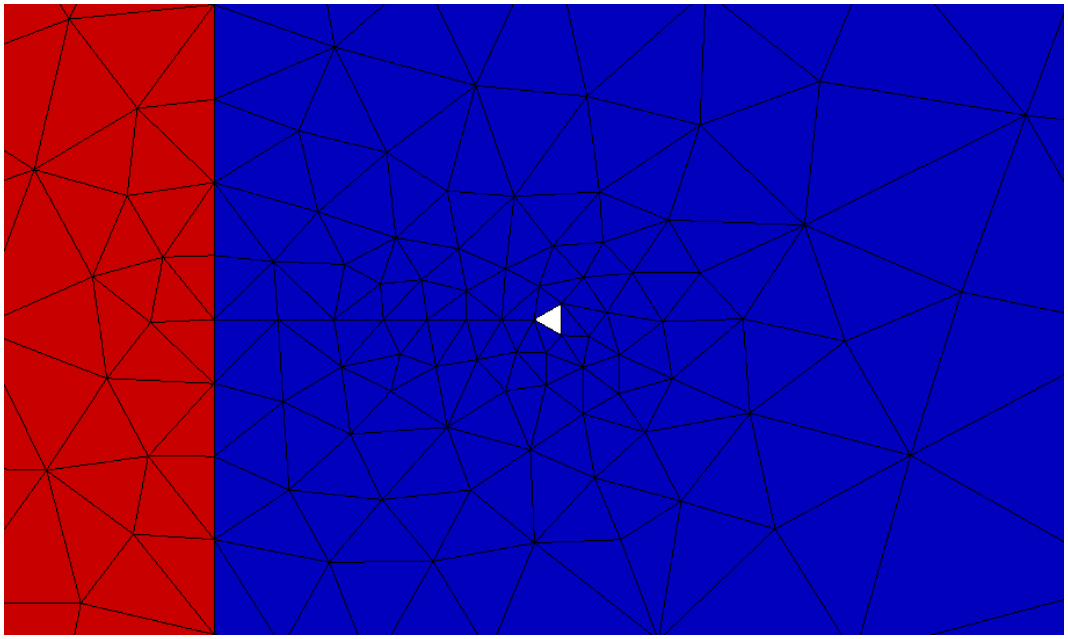

Figure 9. Zoomed image of the domain showing the triangle in the driven region. Red region is the high pressure side while the blue is the driven side

see where and how many refinements are taking place, plots of the adapted convergence tolerance, time step size and number of mesh elements at each time step are displayed in Figures 11, 12 and 13 for each cycle of the algorithm.

The plots of error versus cost (Fig. 10) and the convergence tolerance (Fig.11) show the convergence tolerance was not identified as having a large enough error per cost value to flag any time steps for convergence tolerance refinement after the first sweep. On the subsequent solution sweep (Sweep 1) we see the algebraic error actually increases for a couple of time steps due to the refinement of elements and time steps at the completion of Sweep 0. While this was not accounted for in the derivations of the linear error estimates in Sections II.B, II.C & II.D it is to be expected. Just as the solution becomes more accurate with a refined discretization so does the calculation of the error. From this result it can be inferred that on the initial sweep (Sweep 0) the algebraic error was being under predicted but as the time and space discretizations were refined a better estimate of the algebraic error was calculated. As such, now the initial two time steps are flagged for convergence tolerance refinement on sweep 2 of the time accurate simulation. After the second sweep there are 3 convergence tolerances being used in a single sweep of the simulation with a tolerance of 10^-3 on the initial time steps followed by a tolerance of 10^-2 in the middle time steps and the last time step still at the original tolerance of 10^-1. This produces three different costs to refine elements depending on which time step the element is located within. As such there is a preference to refine an element on a grid which has a lower convergence tolerance than one on a grid with a smaller convergence tolerance if the errors are comparable. This preference manifests into a tiered error versus cost plot for the element refinements as seen in Sweep 4 where the lower cost elements have been refined to a slightly less error than the larger cost elements.

The time step plot (Fig. 12) shows that at the completion of the initial sweep through the simulation only the first time step is flagged for refinement. On the next sweep (Sweep 1) all time steps are below the error per cost threshold line but at the completion of the 2nd sweep the remainder of the time steps which were not already refined are flagged for refinement. At the completion of the 3rd sweep and final refinement only the first time step is flagged for refinement breaking the one step into 2 steps on the subsequent sweep (Sweep 4) of the simulation.

The spatial refinement plots (Fig. 13) show each sweep has an increase in the number of elements within a grid at any given solution time but in general there is not much change between solution times within the same sweep. The only real variation is observed on sweeps 3 and 4 around 10 non-dimensional time units when there is a slight decrease in the number of elements within the grids. This time coincides with the shock wave contacting the triangle which indicates not as many elements have influence on the lift functional

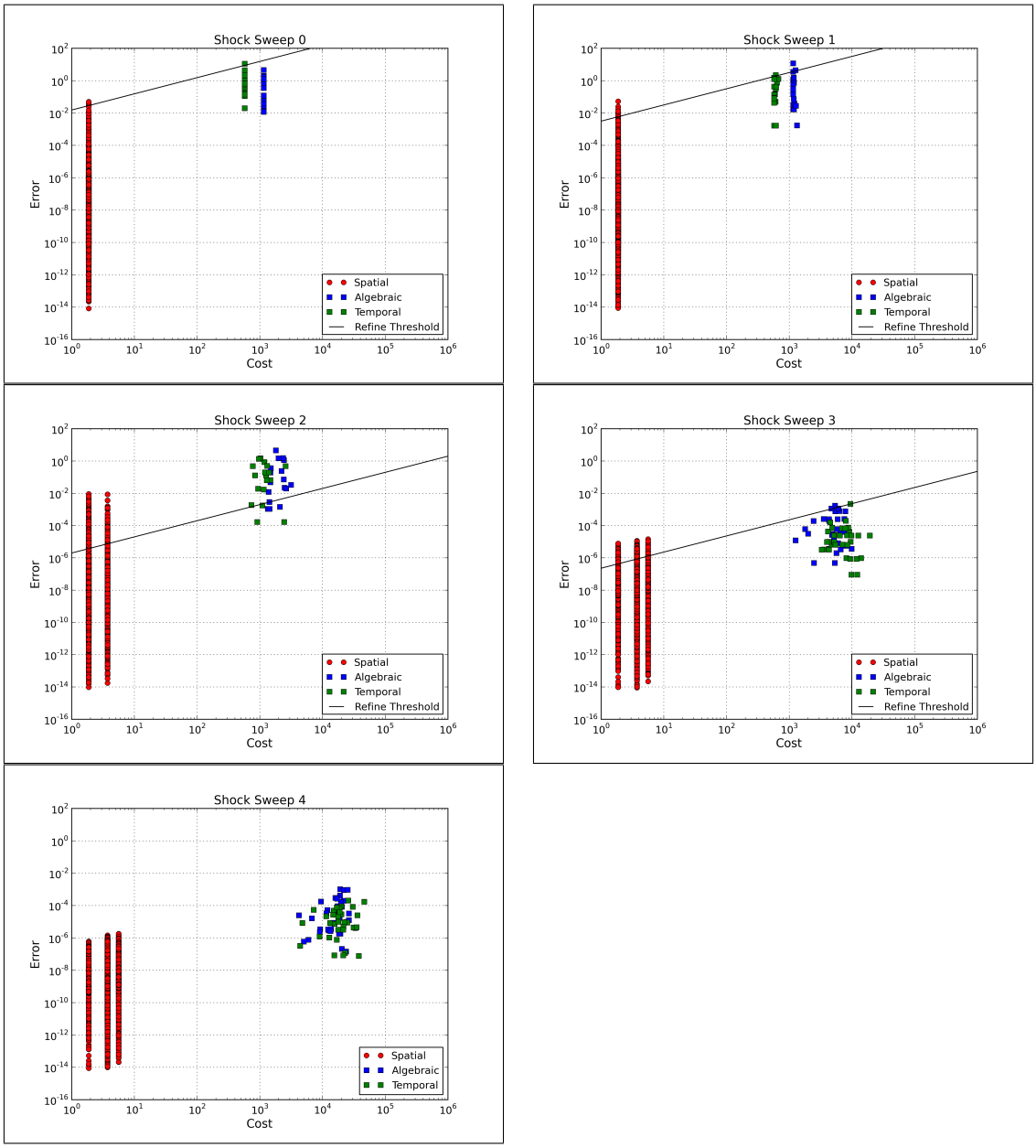

Figure 10. Error vs. Cost plots for the shock-triangle problem over 4 different refinement sweeps

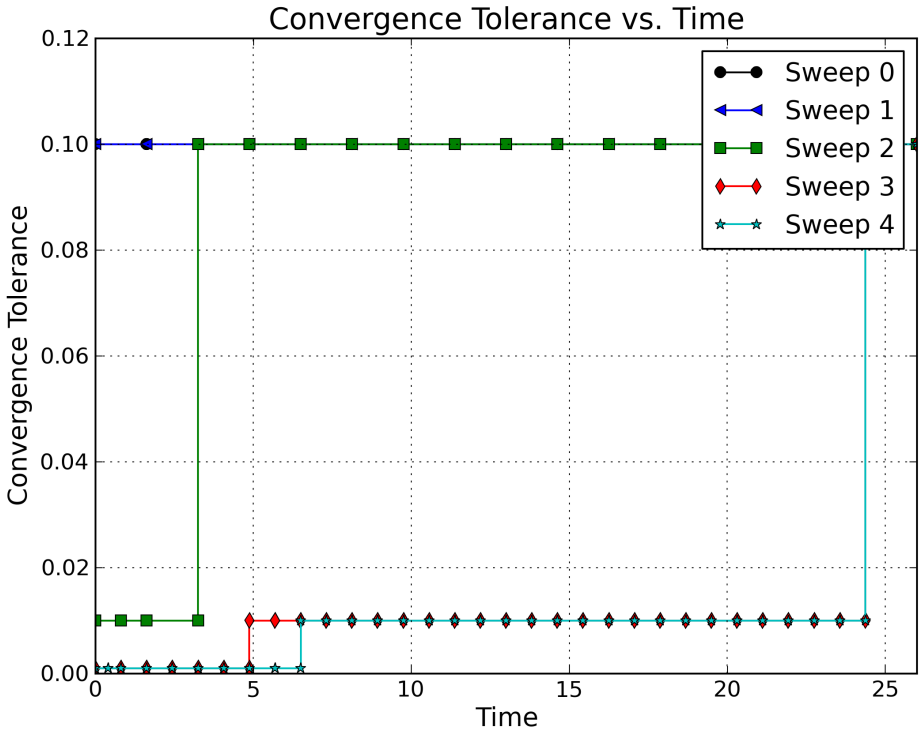

Figure 11. Convergence Tolerance as a function of time for each refinement sweep of the shock

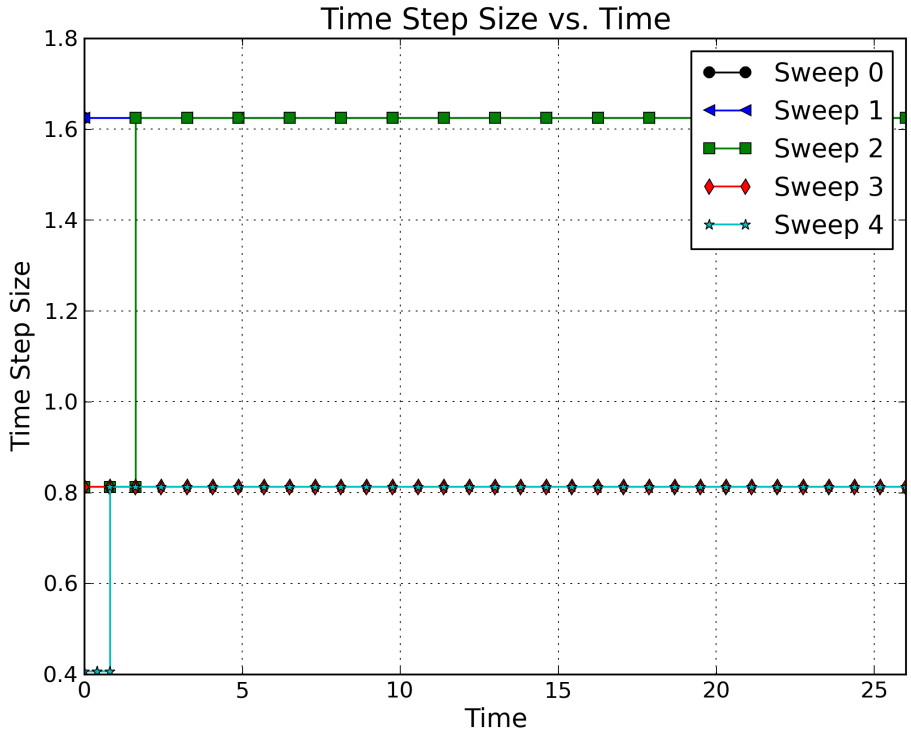

Figure 12. Time step size as a function of time for each refinement sweep of the shock

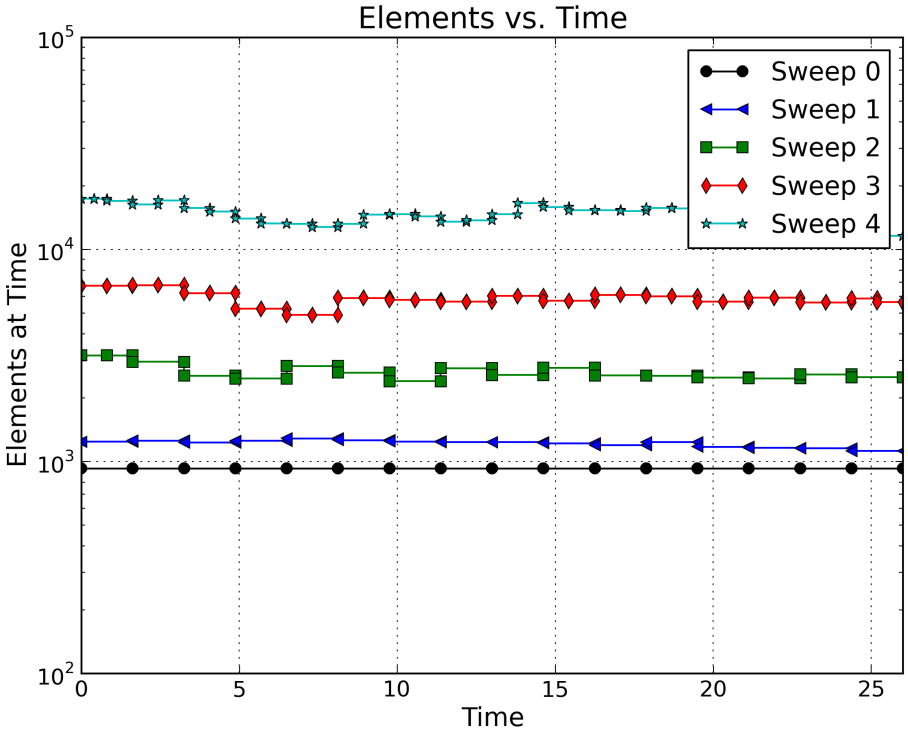

Figure 13. Elements as a function of time for each refinement sweep of the shock

at this point in time.

Images of the refined region within the computational mesh at non-dimensional times of t = 0, 6.5 and 26.0 are shown in Figure 14 for each sweep of the algorithm. The mesh refinement patterns clearly show element refinement tracking the shock wave until it impacts with the triangle. Of note, though, is the elements along the shock wave which do not influence the lift on the triangle at the top and bottom of imaged region are not refined as they would be if a gradient based refinement algorithm were used. In addition, once the wave passes the triangle the adjoint based error adaption correctly identifies the region behind the triangle for refinement to capture the asymmetric vortex shedding that is influencing the lift functional.

V.C. Airfoil with Vortex

The third test problem is an identical strength vortex as to that from the first problem of section V.A convecting past a NACA 0012 airfoil placed with the leading edge at the origin. The vortex is initially placed 2 chord lengths below the airfoil and 15 chord lengths in front of the airfoil leading edge. The free stream velocity is again $M_\infty = 0.5$ and the simulation is run for 60 non-dimensional time units placing the vortex center roughly 14 chord lengths behind the airfoil trailing edge at the final time step. The initial grid contains 2401 elements and again 16 initial time steps converged 1 order of magnitude were used to march through the initial simulation. The starting condition for the time accurate steps is the fully converged flow around the airfoil with the vortex superimposed into the flow field at the starting position. This is not the exact solution at this point in time but it is assumed the vortex is far enough away from the airfoil to have little effect on the objective function of the time integrated lift coefficient on the airfoil. Images of the density within the region containing the vortex and a zoomed image of the initial grid around the airfoil are shown in Figures 15 and 16.

Starting from the initial condition the simulation is allowed to cycle through the refinement sweeps until the global error estimate is less than 10^{-2} (i.e. $Tol_{Global} < 10^{-2}$). The threshold values are again set to decrease each cycle through the refinement algorithm using values of 8, 4, 2 and 1.

Using this setup the requested convergence tolerance is achieved after 5 refinement sweeps and the

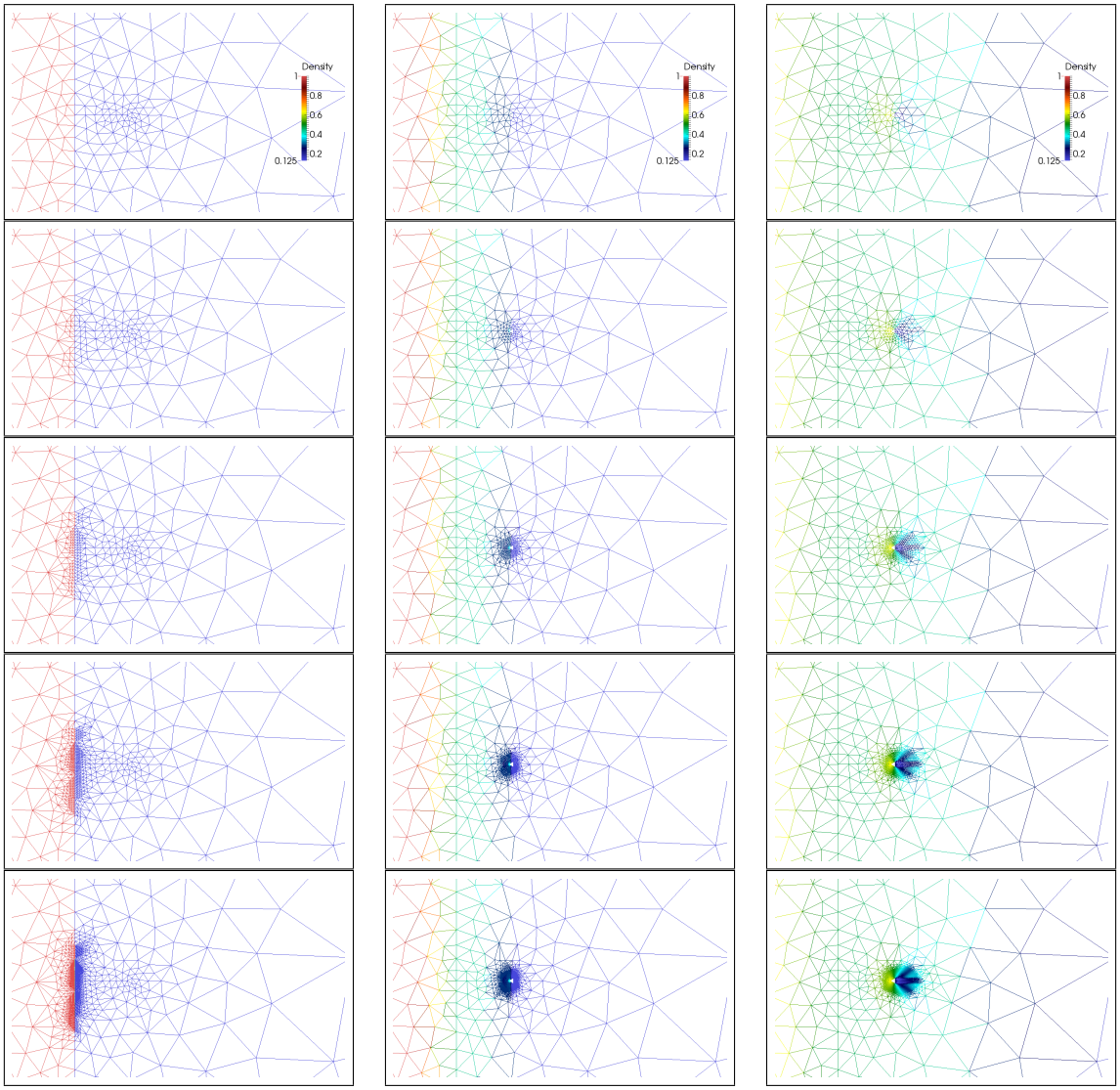

Figure 14. Shock triangle adapted grid and solution at time = 0, 6.5, and 26.0 for 4 different refinement sweeps

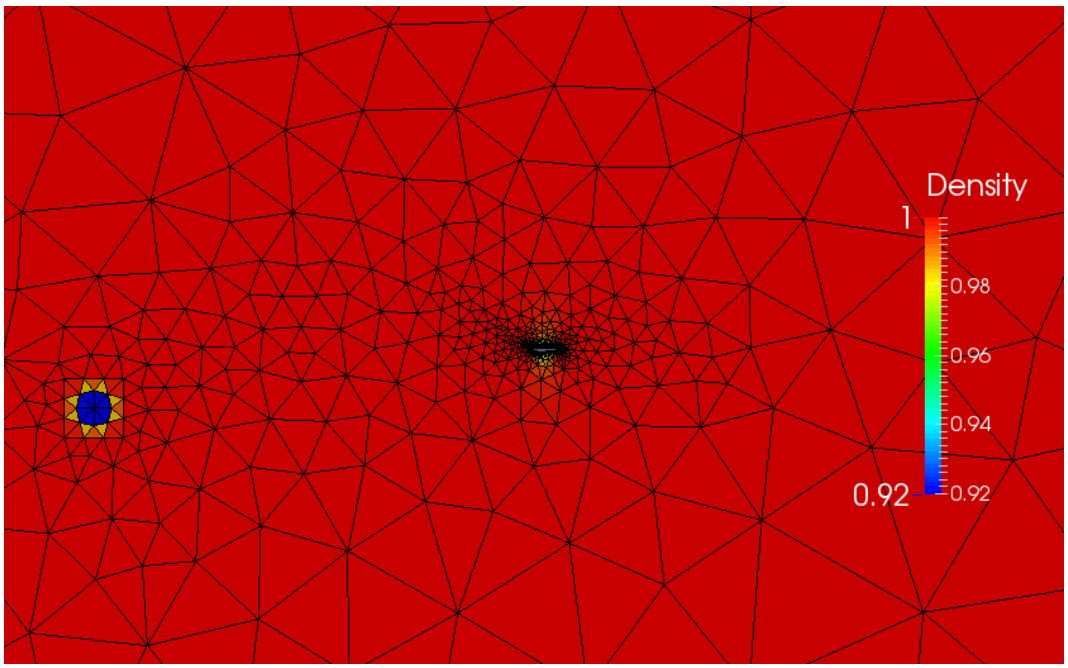

Figure 15. Initial density distribution around airfoil for grid of 2401 elements

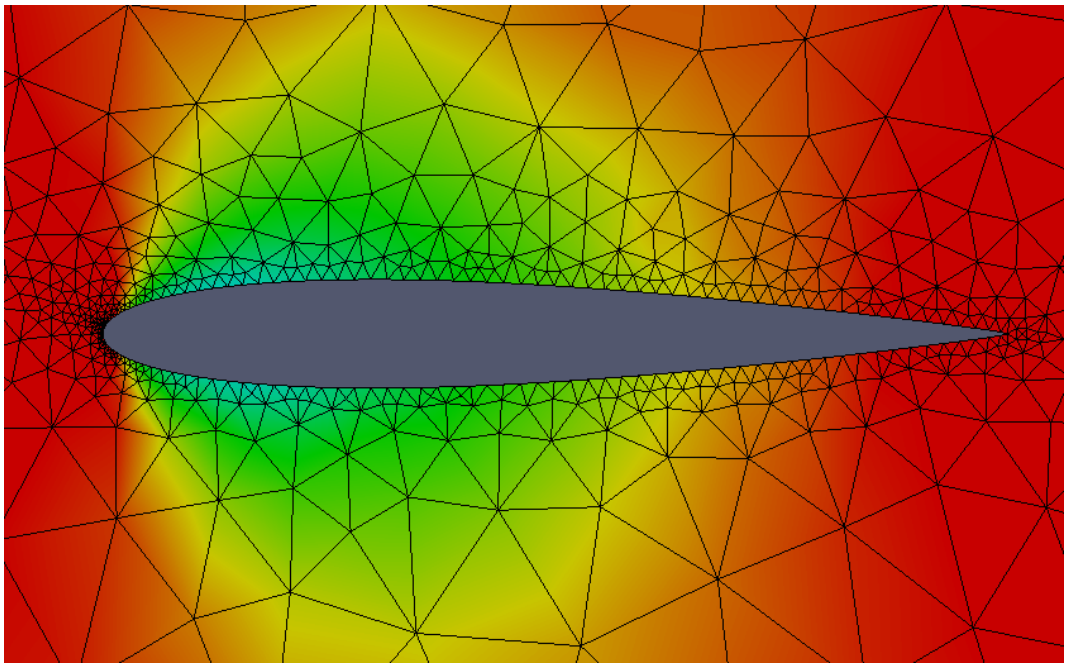

Figure 16. Initial density distribution around zoomed airfoil with grid elements

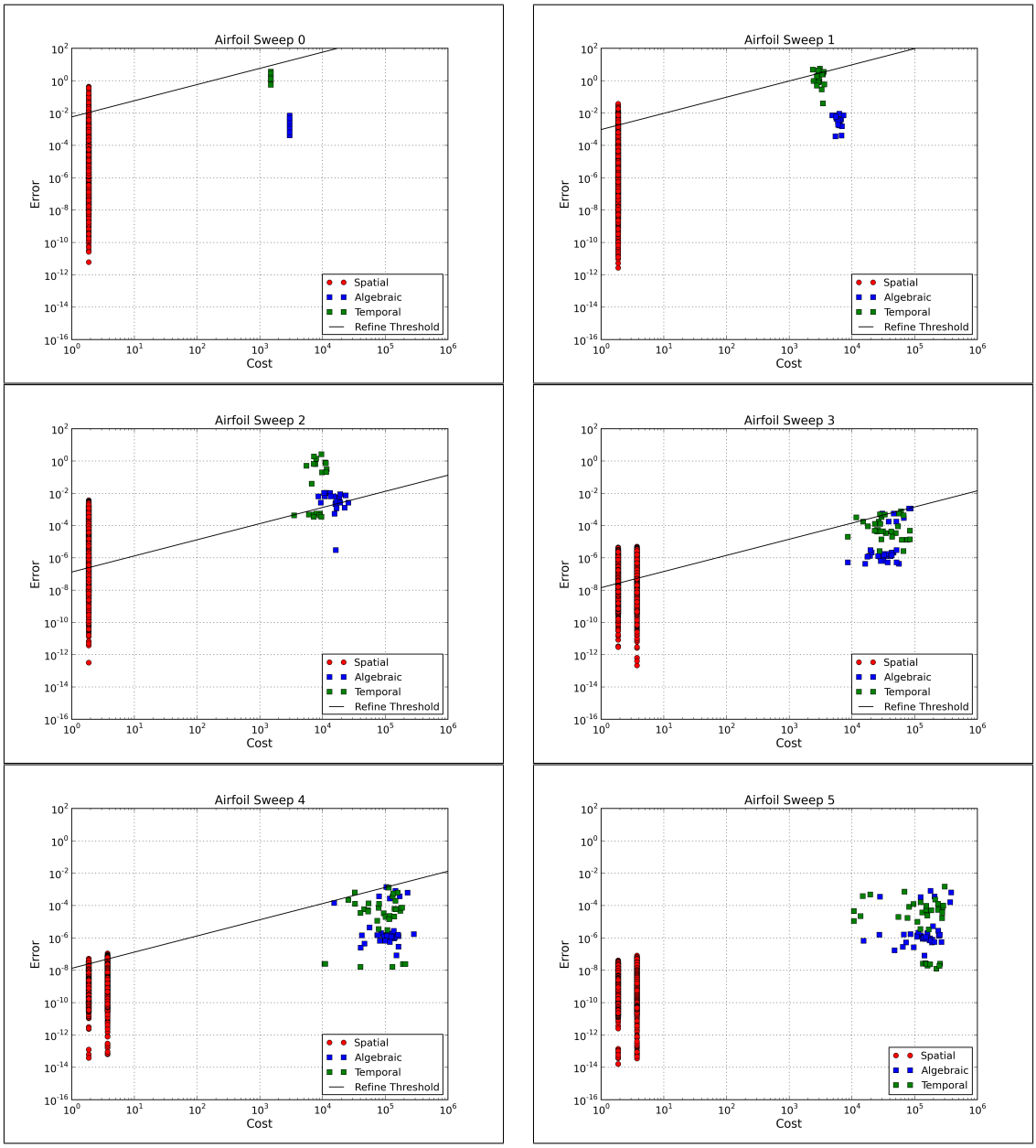

Figure 17. Error vs. Cost plots for the airfoil with convecting vortex over 5 different refinement sweeps.

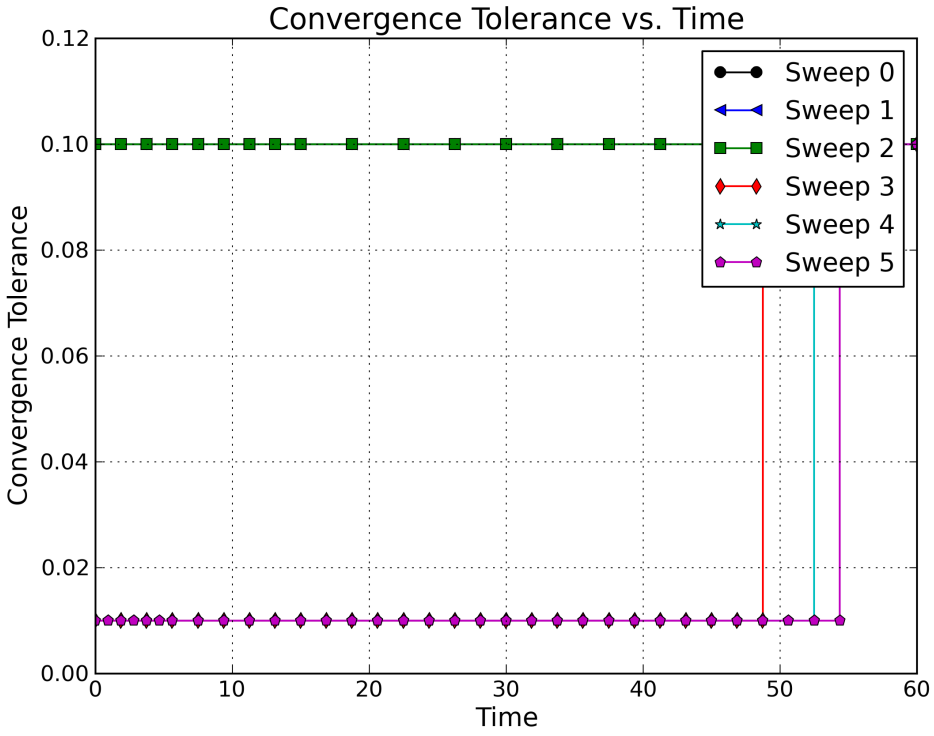

Figure 18. Convergence Tolerance as a function of time for each refinement sweep of the airfoil

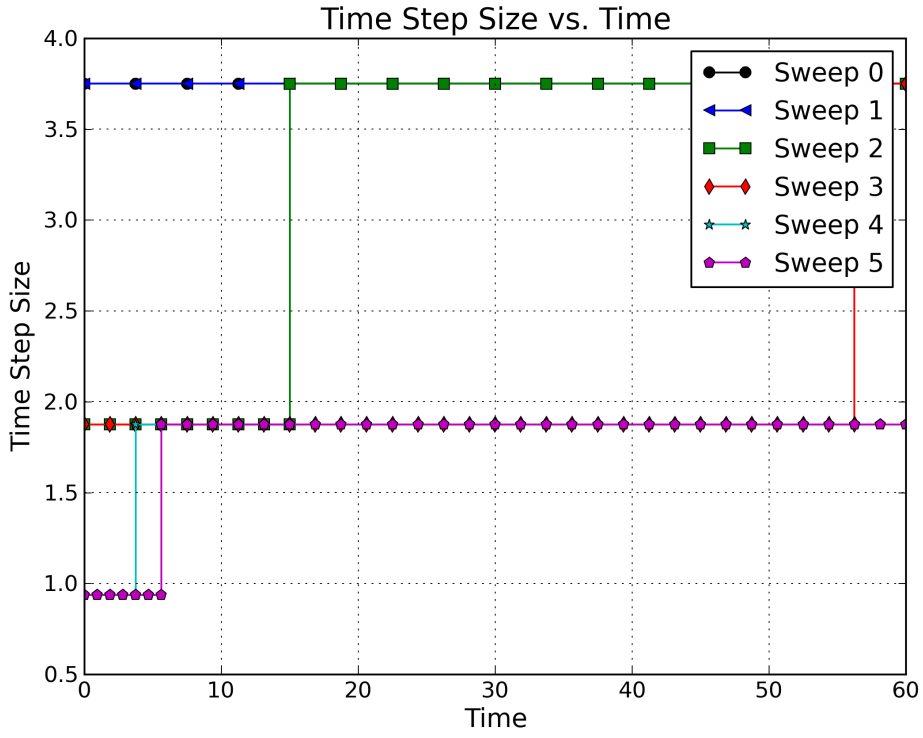

Figure 19. Time step size as a function of time for each refinement sweep of the airfoil

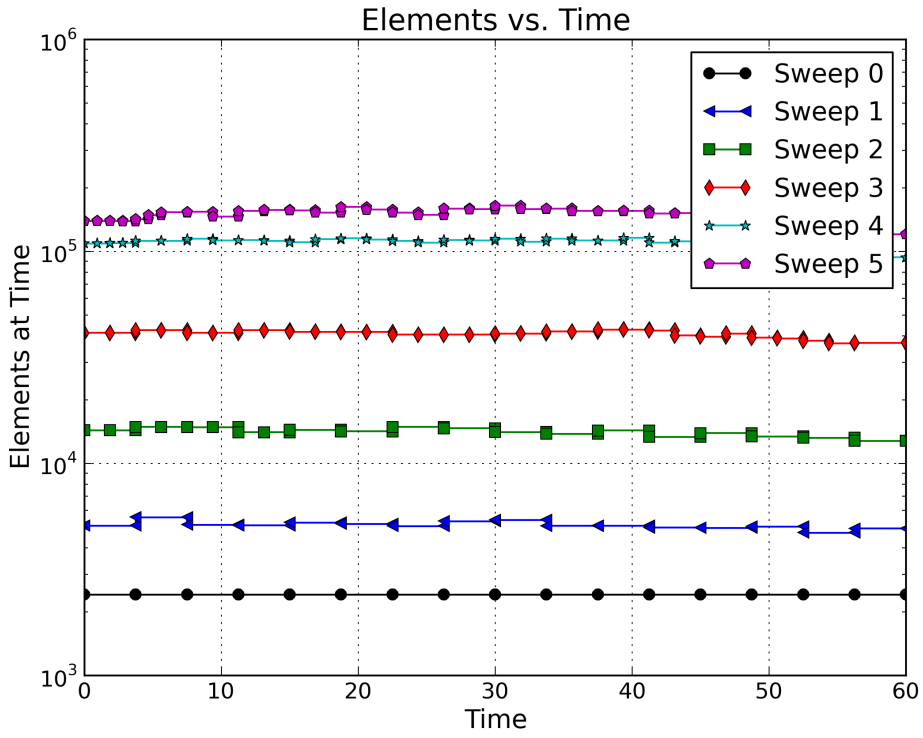

Figure 20. Elements as a function of time for each refinement sweep of the airfoil

resulting error versus cost plots are shown in Figure 17. Additionally, the plots of the adapted convergence tolerance, time step size and number of mesh elements at each time step are displayed in Figures 18, 19 and 20 for each cycle of the algorithm.

The plots of error versus cost (Fig. 17) and the convergence tolerance (Fig.18) show the convergence tolerance was not identified as having a large enough error per cost value to flag any time steps for tolerance refinement after the first two sweeps. After the completion of the 3rd sweep the majority of the time steps are flagged for a reduced convergence tolerance with only the last couple of time steps remaining with the initial tolerance of 10^-1. On subsequent refinement sweeps the convergence tolerance on time steps in the later part of the simulation are flagged for refinement until only the last time step remains with the original convergence tolerance. Again, the different convergence tolerances on different time steps within the same time accurate simulation produce two different values of the cost to refine an element depending on which convergence tolerance is applied to the grid the element is in.

The time step plot (Fig. 19) shows that at the completion of the initial sweep through the simulation no time steps were identified as having errors in need of refinement. After the first refined sweep (Sweep 1) and on all subsequent refinement sweeps a fraction of the steps were identified as needing refinement. The refinement is focused on the initial time steps of the simulation which is expected since an error in the initial steps has a negative influence on the remainder of the time steps within the simulation sweep.

The spatial refinement plots (Fig. 20) show each sweep has an increase in the number of elements within a grid at any given solution time but in general there is not much change between solution times within the same sweep. The one thing that stands out in the figure is the lack of spatial refinement when going from Sweep 4 to 5 as indicated by the small increase in the number of elements between the sweeps. Comparing this to the error versus cost (Fig. 17) we can see that only one time step and a few elements were refined after the 4th sweep. This indicates that the overall cost to achieve the requested error tolerance could potentially be reduced by adjusting the fraction of excess error refined on each refinement sweep (λ in Equation 46) to be more or less aggressive (e.g. $\lambda = 6, 4, 2, 1$ or $\lambda = 16, 8, 4, 2, 1$).

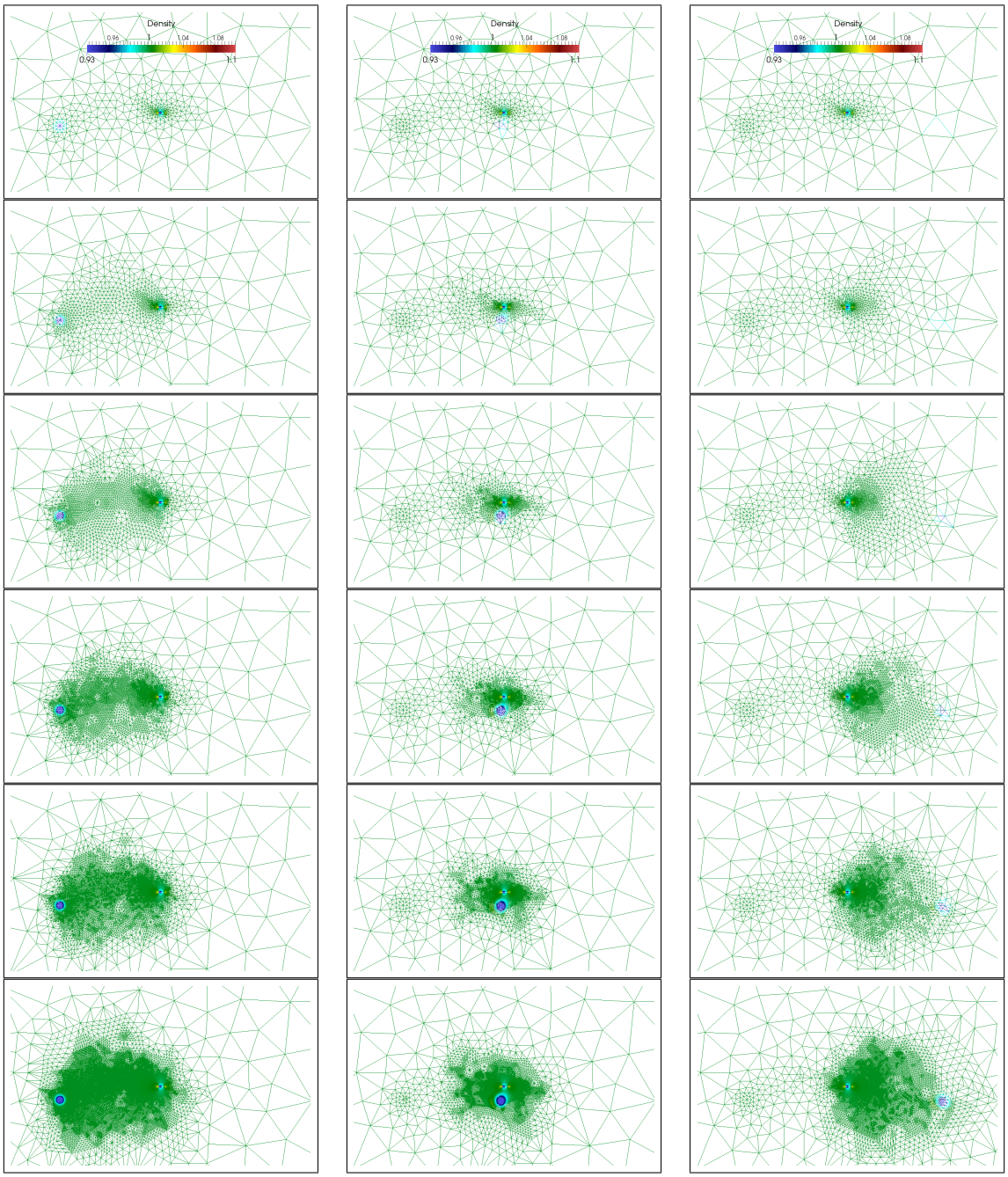

Figure 21. Airfoil with vortex solution and adapted grid at time = 0, 30, and 60 for 5 different refinement sweeps

VI. Conclusions

This work presented a framework for the reduction of the spatial, temporal and algebraic error by use of the discrete adjoint solution. An estimate of the cost to refine each error was developed which enabled an error per cost ratio to be determined for each refinement opportunity. Furthermore, this ratio allowed all error types, irrespective of source, to be compared against each other so that refinement could be target on the refinement opportunities with the largest error per cost.

Three test cases were shown where the algorithm was tested and the resulting discretizations are shown. This research has focused on 2 dimensional problems of short time duration but the real benefit would come in 3 dimensional problems were the extra dimension equates to a much larger computational expense and consequently an algorithm to more efficiently target error would result in larger cost savings. In the long term, the ideal application of this research would be within a space-time framework where all elements could be selectively refined in space, time and convergence without regard for keeping all elements at the same time or converging all elements within a time step an equal amount.

References

[1]Venditti, D. A. and Darmofal, D. L., "Grid adaptation for functional outputs: application to two-dimensional inviscid flows," *Journal of Computational Physics*, Vol. 176, No. 1, 2002, pp. 40–69.

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

[3]Nemec, M. and Aftosmis, M., "Adjoint error estimation and adaptive refinement for embedded-boundary cartesian meshes," No. 2007-4187, AIAA Paper, 2007.

[4]Houston, P., Rannacher, R., and Suli, E., "A posteriori error analysis for stabilised finite element approximations of transport problems," *Computer Methods in Applied Mechanics and Engineering*, Vol. 190, No. 11-12, 2000, pp. 1483–1508.

[5]Becker, R. and Rannacher, R., "An optimal control approach to a posteriori error estimation in finite element methods," *Acta Numerica*, , No. 10, 2001, pp. 1–102.

[6]Giles, M. B. and Suli, E., "Adjoint methods for PDEs: a posteriori error analysis and postprocessing by duality," *Acta Numerica*, , No. 11, 2002, pp. 145–236.

[7]Mani, K. and Mavriplis, D. J., "Discrete adjoint based time-step adaptation and error reduction in unsteady flow problems," No. 2007-3944, AIAA Paper, 2007.

[8]Mani, K. and Mavriplis, D. J., "Error estimation and adaptation for functional outputs in time-dependent flow problems," *Journal of Computational Physics*, Vol. 229, No. 2, 2010, pp. 415–440.

[9]Luo, Y. and Fidkowski, K. J., "Output-based space-time mesh adaption for unsteady aerodynamics," No. 2011-491, AIAA Paper, 2011.

[10]Li, S. and Petzold, L., "Adjoint sensitivity analysis for time-dependent partial differential equations with adaptive mesh refinement," *Journal of Computational Physics*, Vol. 198, No. 1, 2004, pp. 310–325.

[11]Flynt, B. T. and Mavriplis, D. J., "Discrete Adjoint Based Adaptive Error Control in Unsteady Flow Problems," No. 2012-78, AIAA Paper, 2012.

[12]Flynt, B. T. and Mavriplis, D. J., "Efficient Adaptation using Discrete Adjoint Error Estimates in Unsteady Flow Problems," No. 2013-520, AIAA Paper, 2013.

[13]Roe, P., "Approximate Riemann solvers, parameter vectors, and difference schemes," *Journal of Computational Physics*, Vol. 43, No. 1, 1981, pp. 357–372.

[14]Mavriplis, D. J., "Adaptive meshing techniques for viscous flow calculations on mixed element unstructured meshes," *International Journal for Numerical Methods in Fluids*, Vol. 34, No. 2, 2000, pp. 93–111.

[15]Buhmann, M. D., *Radial Basis Functions*, Cambridge University Press, 2004.

[16]Wendland, H., "Piecewise polynomial, positive definite and compactly supported radial functions of minimal degree," *Advances in Computational Mathematics*, Vol. 4, No. 1, 1995, pp. 389–396.

[17]Bonet, J. and Peraire, J., "An alternating digital tree (ADT) algorithm for 3D geometric searching and intersection problems," *International Journal for Numerical Methods in Engineering*, Vol. 31, No. 1, 1991, pp. 1–17.

[18]Nemec, M., Aftosmis, M., and Wintzer, M., "Adjoint-Based Adaptive Mesh Refinement for Complex Geometries," No. 2008-0725, AIAA Paper, 2008.

[19]Davoudzadeh, F., Mcdonald, H., and Thompson, B., "Accuracy evaluation of unsteady cfd numerical schemes by vortex preservation," *Computer in FLuids*, Vol. 24, No. 1, 1995, pp. 1995.

[20]Yee, H., Sandham, N., and Djomehri, M., "Low dissipative high order shock-capturing methods using characteristic-based filters," *Journal of Computational Physics*, Vol. 150, No. 1, 1999, pp. 199–238.

[21]Whitfield, D. and Janus, J., "Three-Dimensional Unsteady Euler Equations Solution Using Flux Vector Splitting," No. 84-1552, AIAA Paper, 1984.