Functional Error Estimation and Control for Time-Dependent Problems

Dimitri Mavriplis *
Department of Mechanical Engineering
University of Wyoming
Laramie WY USA

September 14-17, 2015

Contents

*email:mavripl@uwyo.edu

1 Introduction

Although adjoint techniques have gained widespread use for sensitivity analysis and optimization methods, they can also be used for error estimation purposes, which in turn can be used to drive adaptive error control schemes in the interest of producing more accurate, reliable and predictive simulation outcomes. It stands to reason that if adjoint methods can be used to compute sensitivities of a functional with respect to any number of parameters, they can be used to determine regions of the computational domain that are most *sensitive* to overall functional accuracy and thus identify regions of large error contributions. The principal characteristic of adjoint error estimation techniques is that they provide an estimate of the error in a specific simulation objective or quantity of interest, rather than an estimate of total simulation error. For many engineering problems this is an important advantage as it allows one to focus computational resources on optimizing the accuracy of the specific simulation objective while de-emphasizing other aspects of the simulation that may be less important for the objective of interest. For example, the requirements for computing an accurate far-field sonic-boom signature for a supersonic aircraft are substantially different than those for obtaining an accurate pitching moment on the same configuration at the same conditions, and an adjoint-based approach can provide guidance in this respect, compared to traditional local discretization error estimators which target all error in all regions of the domain.

The use of adjoint methods for estimating errors in simulation output objectives is now well established, particularly for spatial discretization errors in steady-state aerodynamic problems [21, 11, 16, 4]. However, there is a need to extend these methods to more complex time-dependent and multidisciplinary problems. Whereas for steady-state single discipline problems the dominant error source is generally associated with spatial discretization error (excluding modeling error), multiple competing error sources must be considered for more complex simulations. These include not only spatial discretization error, but also temporal error, algebraic error due to incomplete convergence at each time step, and coupling error for coupled multidisciplinary problems. Furthermore, for multidisciplinary problems, it is of interest to understand the relative importance of the error contributions of the various disciplines to the overall computed objectives. Based on the development of discrete adjoint methods for time-dependent and multidisciplinary problems given in the previous chapter, we will develop a framework for estimating and controlling these various error contributions to specific output functionals for time-dependent and multidisciplinary problems in this chapter.

1.1 Continuous Linear Formulation

The basic derivation for adjoint error estimation can be performed in several manners. Most commonly, this derivation has been reported in the literature for steady-state spatial discretization error. For linear problems, a variational approach is often adopted. Following reference [18], we consider the linear problem with homogeneous boundary conditions

$$Au = f \tag{1}$$

where A is a linear operator on the domain Ω. We are interested in a specific scalar functional output L from this problem, which is computed from the solution u as

$$L = (g, u) \tag{2}$$

where the parentheses denote the functional inner product. For example, for aerodynamic problems the objective could be a surface integrated force coefficient which is computed as a weighted integral of the solution u with the weighting function g (although this may involve additional boundary conditions). If A^* denotes the adjoint operator of A, defined as the operator which satisfies

$$(Au, v) = (u, A^*v) \tag{3}$$

this functional can be evaluated in the dual form $L = (v, f)$ where v is the solution of the dual or adjoint problem:

$$A^*v = g \tag{4}$$

This is easily shown using the definition of the adjoint operator :

$$\begin{aligned} L = (g, u) &= (A^*v, u) && (5)\\ &= (v, Au) && (6)\\ &= (v, f) && (7) \end{aligned}$$

v represents the solution to the adjoint problem, and is often referred to as the adjoint variable. v can also be thought of as a generalized Green's function for the operator A where the usual Dirac delta function used in the Green's function definition is replaced with the function g [6].

If $\tilde{u}$ represents an approximate solution to equation (1) which produces an approximate estimate for the objective denoted as $\tilde{L}$, the error in the objective can be written as:

$$\begin{aligned} \Delta L = L - \tilde{L} &= (g, u) - (g, \tilde{u}) && (8)\\ &= (g, u - \tilde{u}) && (9)\\ &= (A^*v, u - \tilde{u}) && (10)\\ &= (v, A(u - \tilde{u})) && (11)\\ &= (v, A\tilde{u} - f) && (12) \end{aligned}$$

illustrating the fact that the change in the objective is given (exactly in this case) by the inner product of the adjoint solution with the residual of the original problem. However, finding the exact solution v of the adjoint problem given by equation (4) is often no less difficult than solving exactly the original problem statement given by equation (1), so little has been gained at this point in terms of computational efficiency.

On the other hand, if we assume that we only have an approximate solution $\tilde{v}$ to the adjoint problem, the above equation may be rewritten as:

$$\Delta L = (\tilde{v}, A\tilde{u} - f) + (v - \tilde{v}, A\tilde{u} - f) \tag{13}$$

Assuming that the exact solution of the adjoint problem is not known, then the first term on the left-hand side constitutes the computable correction while the second term

corresponds to the remaining error which is not known. However, we can show that, under certain conditions, the remaining error term is much smaller than the computable error. For example, if the approximate values $\tilde{u}$ and $\tilde{v}$ are obtained from consistent discretizations on a given grid level, then both the residual $A\tilde{u} - f$ and the adjoint error $v - \tilde{v}$ decrease asymptotically with increasing grid resolution. Therefore the remaining error term decreases asymptotically faster than the computable error term, leading to a superconvergence of the accuracy of the corrected functional [17] defined as:

$$L \approx L_{corrected} = \tilde{L} + (\tilde{v}, A\tilde{u} - f) \tag{14}$$

We emphasize that the computable error is not a strict error bound for the functional, since the remaining error can only be shown to be *asymptotically* smaller than the computable error, and the relative size of these terms cannot be determined a priori for a given approximate solution. We also note that the success of this error estimation technique is predicated on the asymptotic convergence of the $v - \tilde{v}$ term. This means that the approximate adjoint solution $\tilde{v}$ must be obtained from a discretization which is consistent with the continuous adjoint problem such that this term vanishes asymptotically faster than the computable error term with increasing resolution. For discrete adjoint implementations, there is no guarantee that the discretization inherited from the original analysis problem discretization is consistent with that of the continuous adjoint problem. Therefore, particular attention must be paid to the dual consistency of the discrete adjoint implementation which can be seen to be much more critical for adjoint error estimation purposes as compared to optimization purposes [19, 9, 10, 15].

1.2 Application to Discrete Non-Linear Problems

In order to extend the adjoint error estimation formulation to the discrete non-linear case, we adopt the approach described in reference [20, 21] based on Taylor series expansions. Consider a functional $L_h(\mathbf{u}_h)$ where $\mathbf{u}_h$ denotes the exact solution of the discrete problem $\mathbf{R}_h(\mathbf{u}_h) = 0$ evaluated on a discretization denoted by h. Given an approximate solution $\tilde{\mathbf{u}}_h$, we can approximate the corresponding change in the functional using a Taylor series expansion as:

$$L_h(\mathbf{u}_h) = L_h(\tilde{\mathbf{u}}_h) + \left(\frac{\partial L_h}{\partial \mathbf{u}_h}\right)_{\tilde{\mathbf{u}}_h} (\mathbf{u}_h - \tilde{\mathbf{u}}_h) + \cdots \tag{15}$$

Using the Taylor series of the residual statement about the approximate solution $\tilde{\mathbf{u}}_h$

$$\mathbf{R}_h(\mathbf{u}_h) = \mathbf{R}_h(\tilde{\mathbf{u}}_h) + \left[\frac{\partial \mathbf{R}_h}{\partial \mathbf{u}_h}\right]_{\tilde{\mathbf{u}}_h} (\mathbf{u}_h - \tilde{\mathbf{u}}_h) + \cdots = 0 \tag{16}$$

we obtain an expression for the difference between the exact and approximate solution as:

$$\mathbf{u}_h - \tilde{\mathbf{u}}_h \approx -\left[\frac{\partial \mathbf{R}_h}{\partial \mathbf{u}_h}\right]^{-1}_{\tilde{\mathbf{u}}_h} \mathbf{R}_h(\tilde{\mathbf{u}}^h) \tag{17}$$

Substituting this into equation (15) yields

$$L_h(\mathbf{u}_h) \approx L_h(\tilde{\mathbf{u}}_h) - \left(\frac{\partial L_h}{\partial \mathbf{u}_h}\right)_{\tilde{\mathbf{u}}_h} \left[\frac{\partial \mathbf{R}_h}{\partial \mathbf{u}_h}\right]^{-1}_{\tilde{\mathbf{u}}_h} \mathbf{R}_h(\tilde{\mathbf{u}}_h) \tag{18}$$

We now define an adjoint variable corresponding to the second and third terms on the right-hand side of the above equation:

$$\mathbf{\Lambda}_h^T = -\left(\frac{\partial L_h}{\partial \mathbf{u}_h}\right)_{\tilde{\mathbf{u}}_h} \left[\frac{\partial \mathbf{R}_h}{\partial \mathbf{u}_h}\right]^{-1}_{\tilde{\mathbf{u}}_h} \tag{19}$$

which after transposing leads to the adjoint equation:

$$\left[\frac{\partial \mathbf{R}_h}{\partial \mathbf{u}_h}\right]^T_{\tilde{\mathbf{u}}_h} \mathbf{\Lambda}_h^T = -\left(\frac{\partial L_h}{\partial \mathbf{u}_h}\right)^T_{\tilde{\mathbf{u}}_h} \tag{20}$$

If the adjoint variable is first obtained by solving this equation, the expression for the change in the objective becomes:

$$L_h(\mathbf{u}_h) - L_h(\tilde{\mathbf{u}}_h) \approx \mathbf{\Lambda}_h^T \mathbf{R}_h(\tilde{\mathbf{u}}_h) \tag{21}$$

Thus, the change in the objective is given as the inner product of the adjoint variable with the residual evaluated using the approximate solution $\tilde{\mathbf{u}}_h$. We note that this error expression is exact for linear problems as discussed previously for equation (12), but becomes approximate for nonlinear problems, even in the presence of the exact adjoint solution $\mathbf{\Lambda}_h$. The fact that changes in the objective are related to changes in the residual through the adjoint variable can be seen by rearranging equation (19) to obtain

$$\mathbf{\Lambda}_h^T = -\frac{\partial L_h}{\partial \mathbf{R}_h} \tag{22}$$

where the adjoint variable is seen to correspond to the sensitivity of the objective with respect to the residual. Thus, a small change in the residual produces a corresponding change in the objective as

$$\delta L_h = -\mathbf{\Lambda}_h^T \delta \mathbf{R}_h \tag{23}$$

This behavior can also be explained through the interpretation of the adjoint variable as the discrete analogue to a generalized Green's function of the residual operator, as mentioned previously in the case of the continuous problem.

At this point, we note that this error estimate requires the exact discrete adjoint solution $\mathbf{\Lambda}_h^T$, which in general may be as expensive to compute as the exact discrete solution $\mathbf{u}_h$ itself. For example, if the approximate solution $\tilde{\mathbf{u}}_h$ was obtained by solving the corresponding problem on a coarser discretization H and projecting this solution onto the finer discretization h as:

$$\tilde{\mathbf{u}}_h = I_H^h \mathbf{u}_H \tag{24}$$

where I_H^h represents a suitably accurate projection operator, and where $\mathbf{u}_H$ is the exact solution of the coarse discretization residual equation $\mathbf{R}_H(\mathbf{u}_H) = 0$, obtaining the exact adjoint solution on the fine discretization h could be as expensive as solving the exact solution $\mathbf{u}_h$ on the fine level itself. Thus, we assume we only have an approximate adjoint solution $\tilde{\mathbf{\Lambda}}_h^T$. For example this could be obtained as a projection of a coarse level adjoint solution onto the fine level as

$$\tilde{\mathbf{\Lambda}}_h = I_H^h \mathbf{\Lambda}_H \tag{25}$$

where $\mathbf{\Lambda}_H$ is the exact solution of the corresponding coarse level discrete adjoint equation given as:

$$\left[\frac{\partial \mathbf{R}_H}{\partial \mathbf{u}_H}\right]^T_{\mathbf{u}_H} \mathbf{\Lambda}_H^T = -\left(\frac{\partial L_H}{\partial \mathbf{u}_H}\right)^T_{\mathbf{u}_H} \tag{26}$$

We can then write the estimate for the change in the objective as:

$$L_h(\mathbf{u}_h) - L_h(\tilde{\mathbf{u}}_h) \approx \tilde{\mathbf{\Lambda}}_h^T \mathbf{R}_h(\tilde{\mathbf{u}}_h) + (\mathbf{\Lambda}_h^T - \tilde{\mathbf{\Lambda}}_h^T)\mathbf{R}_h(\tilde{\mathbf{u}}_h) \tag{27}$$

Once again, we recover a computable error term and a remaining error term that depends on the exact discrete adjoint solution $\mathbf{\Lambda}_h$, which is generally not known. If the approximate adjoint solution $\tilde{\mathbf{\Lambda}}_h$ is close to the exact discrete adjoint solution $\mathbf{\Lambda}_h$, then the remaining error term will be significantly smaller than the computable error term and the computable error term can be used as an error estimate. We emphasize again that the computable error term does not constitute an error bound for the functional, since the remaining error term can only be shown to be *asymptotically* smaller than this latter term. Additionally, the current analysis does not take into account any non-linear errors due to the omission of all higher order terms in the Taylor series.

Despite these drawbacks, in many cases, the computable error term provides a good estimate of the error in the functional as a result of the approximate nature of the current solution $\tilde{\mathbf{u}}_h$. However, it is important to keep in mind that this analysis does not provide an estimate of the error between the current functional value and the exact value obtained in the continuous limit. Rather it provides a prediction of the difference between the current functional value (for example evaluated with an approximate solution from a coarse discretization level H) and the value of the functional evaluated with the exact discrete solution at the target discretization level h. Additionally, for the remaining error term to be small, we only require that the approximate adjoint solution be close to the exact discrete adjoint solution on the fine level h. Thus, even in the presence of a discrete adjoint implementation that is not dually consistent (i.e. where the adjoint solution may not converge to the continuous adjoint problem) using an approximate adjoint solution which is close to the exact discrete adjoint $\mathbf{\Lambda}_h$ (or solving the fine level discrete adjoint problem directly) can be expected to result in a good approximation of the objective functional evaluated at that discretization level h. Of course the drawback is that dual inconsistency may slow convergence of the functional to the continuous limit as the resolution is increased.

Figure 1 illustrates some of these characteristics of adjoint-based error prediction. In this case, the adjoint error estimation is used to drive an adaptive meshing scheme for

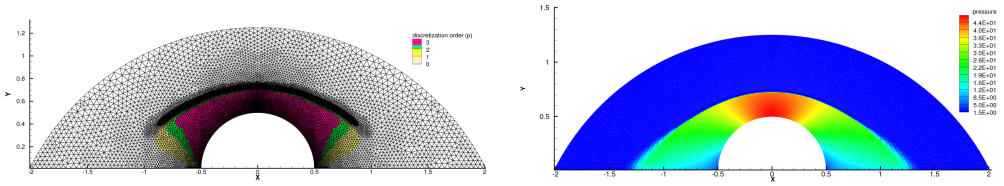


Figure 1: Illustration of adjoint error driven adaptive refinement for Mach 6 flow over cylinder using variable order discontinuous Galerkin discretization: 3 level h-p refined mesh (left); Computed pressure contours on final mesh (right)

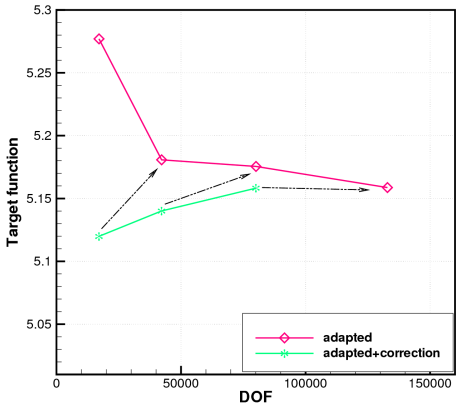


Figure 2: Functional and adjoint-corrected functional values as a function of degrees of freedom throughout refinement process for hypersonic flow over cylinder test case

the case of hypersonic flow over a cylinder. The discretization consists of a discontinuous Galerkin scheme with both h and p refinement capabilities [22]. After 3 levels of adaptation, the resulting mesh and solution in terms of computed pressure contours are shown in Figure 1. In this case, the targeted functional is the average surface temperature of the cylinder, and Figure 2 depicts the convergence of the functional value with refinement levels. At each level, the value of the functional is plotted, along with the corrected value using the computable error term. The corrected functional values are estimates of the functional values on the next finer level, and the arrows are used to point to the value they are intended to predict. As can be seen, the prediction accuracy improves with refinement levels and the final prediction becomes very accurate (horizontal arrow). This case also demonstrates the ability of the adjoint error prediction to focus on regions of importance for the objective at hand. For example, in Figure 1 the mesh is highly refined only in the shock regions ahead of the cylinder, and the lateral shock regions are not targeted since the flow in these regions does not affect the cylinder surface temperature significantly.

In the work that follows, we focus on the formulation and evaluation of the computable error term for time-dependent and multidisciplinary problems, and use this error estimate to drive adaptive processes for controlling error. Although other approaches have been developed which seek to estimate the remaining error or the non-linear error, for example by forming a duality gap term [21], we focus exclusively on the computable error term for simplicity.

In the above derivation, in general we did not specify the approximate nature of the solution $\tilde{\mathbf{u}}_h$. Thus, the derivation holds for any approximate solution and may be used to estimate spatial discretization error due to solutions computed on coarser grids, temporal error due to solutions computed using larger time steps, algebraic error due to incompletely converged solutions, and even disciplinary model error due to solutions computed with lower fidelity models. In the following section, we first derive the adjoint-based error estimates for a time-dependent moving mesh two-dimensional inviscid flow problem and illustrate the potential of these estimates for controlling temporal and algebraic error. In the subsequent section, we combine spatial, temporal, and algebraic error estimation and adaptive control for problems with stationary (but adaptive) meshes. We also develop a

strategy for targeting reduction in overall simulation error at optimal computational cost. In the final sections we present a general formulation for multidisciplinary problems and speculate on future areas of research.

2 Temporal and Algebraic Functional Error Estimation and Control

In this section, we derive and demonstrate the use of adjoint-based error estimation techniques for time-dependent problems. This requires the use of a time-dependent adjoint formulation, and we rely on the discrete adjoint formulation for time-dependent and multidisciplinary problems derived in the previous chapter. We also include the effect of dynamically deforming meshes, which provides a glimpse of multidisciplinary error estimation techniques, considering the mesh motion equations as a separate coupled discipline. We focus exclusively on the estimation and control of temporal and algebraic error in this section and defer the inclusion of spatial discretization error within time-dependent problems to the next section. This is partly due to the fact that adjoint error estimation for spatial discretization has been covered extensively for steady-state problems, but also because this allows us to avoid the complications that arise in implementing dynamic adaptive mesh refinement in conjunction with deforming meshes.

Although the need to estimate temporal error is self-evident for time-dependent problems, the consideration of algebraic error is particularly important in unsteady problems since it is seldom practical to converge each implicit time step to deep levels of convergence. In fact, temporal discretization error and algebraic error are intricately linked in time-dependent problems. For example, reducing the time-step size generally results in faster convergence, although the algebraic error resulting from partial convergence accumulates over the total number of time steps. Therefore, it is important to consider both error sources simultaneously.

The general simulation problem consists of a set of flow equations and a set of mesh motion equations to be solved at each time step which we denote in residual form as:

$$\mathbf{R}_h(\mathbf{U}_h, \mathbf{x}_h) = 0 \tag{28}$$

$$\mathbf{G}_h(\mathbf{x}_h) = 0 \tag{29}$$

for the flow and mesh motion problems, respectively. The flow equations consist of the Euler equations solved in arbitrary Langrange Eulerian (ALE) form, using a second-order backwards difference (BDF2) time-stepping scheme. The mesh motion equations are based on a linear spring analogy approach where internal mesh coordinate displacements are computed by solving the linear equation

$$[K]\,\delta x = \delta x_{surf} \tag{30}$$

in response to prescribed surface grid point displacements δx_{surf}. In order to be consistent with the flow residual equations, the mesh motion residual is written in time integrated form as:

$$\mathbf{G}_h(\mathbf{x}_h) = \frac{1}{\Delta t}\left\{[K]\,\delta x - \delta x_{surf}\right\} = 0 \tag{31}$$

where Δt corresponds to the time step at resolution h. In the above equations, the subscript h denotes the temporal resolution. We generally consider fine h and coarse H temporal resolutions where the time steps are divided by a factor of 2 in going from H to h temporal discretizations, while the spatial resolution is held fixed in all cases. We adopt the shorthand notation where the above equations apply over all space and time, and thus the variables $\mathbf{U}_h$ and $\mathbf{x}_h$ also represent the flow and grid coordinates at all spatial and temporal locations.

2.1 Error due to temporal resolution

Consider a functional computed based on the unsteady solutions of the flow and mesh motion equations. Mathematically this may be represented as $L = L(\mathbf{U}, \mathbf{x})$ where L is the functional, $\mathbf{U}$ is the unsteady flow solution set and $\mathbf{x}$ is the set of unsteady mesh solutions. The functional evaluated on two different temporal meshes of discrete resolutions h and H can be represented by $L_h(\mathbf{U}_h, \mathbf{x}_h)$ and $L_H(\mathbf{U}_H, \mathbf{x}_H)$. Here H refers to some arbitrary coarse resolution temporal mesh and h is a fine resolution temporal mesh constructed by nested subdivision of H using a ratio of 2-to-1. An estimate of $L_h(\mathbf{U}_h, \mathbf{x}_h)$ can be obtained based on the Taylor expansion around the functional $L_h(\mathbf{U}_h^H, \mathbf{x}_h^H)$, where $\mathbf{U}_h^H$ and $\mathbf{x}_h^H$ refer to projections of the flow and mesh solutions from the coarse to the fine level. Assuming that the coarse time domain flow solution $\mathbf{U}_H$ and the coarse time domain mesh coordinates $\mathbf{x}_H$ have been obtained via full convergence of the respective equations, the Taylor series expansion of the exact fine time domain functional can be written as:

$$L_h(\mathbf{U}_h, \mathbf{x}_h) = L_h(\mathbf{U}_h^H, \mathbf{x}_h^H) + \left[\frac{\partial L_h}{\partial \mathbf{U}_h}\right]_{\mathbf{U}_h^H, \mathbf{x}_h^H} \left(\mathbf{U}_h - \mathbf{U}_h^H\right) + \left[\frac{\partial L_h}{\partial \mathbf{x}_h}\right]_{\mathbf{x}_h^H, \mathbf{U}_h^H} \left(\mathbf{x}_h - \mathbf{x}_h^H\right) + \cdots \quad (32)$$

If *nsteps* is defined as the number of time-steps in the temporal mesh h, and if *ncells* and *nnodes* are the number of elements and nodes in the spatial mesh, then $\mathbf{U}_h$, $\mathbf{U}_h^H$, $\mathbf{x}_h$ and $\mathbf{x}_h^H$ are vectors of size $[nsteps \times 1]$ where each element in the vectors are by themselves a vector of size $[ncells \times 1]$ for the flow variables and a vector of size $[nnodes \times 1]$ for the mesh variables. This indicates that $[\partial L/\partial \mathbf{U}]_{\mathbf{U}_h^H, \mathbf{x}_h^H}$ is a vector of size $[1 \times nsteps]$, where each element in the vector is again a vector of size $[1 \times ncells]$. The same argument applies for the sensitivity of the functional to the mesh coordinates.

The procedure for computing an unsteady solution involves obtaining the solution to the non-linear residual operator $\mathbf{R}(\mathbf{U}, \mathbf{x})$ at each time interval. The non-linear operator $\mathbf{R}(\mathbf{U}, \mathbf{x})$ is constructed by some appropriate discretization of the governing flow equations and for this work we use a cell-centered finite-volume discretization of the Euler equations on an unstructured triangular mesh. When the time derivative term in the Euler equations is based on the second-order BDF2 discretization, and the equations are discretized in ALE form, the non-linear residual takes on the functional form: $\mathbf{R}^n = \mathbf{R}^n(\mathbf{U}^n, \mathbf{U}^{n-1}, \mathbf{U}^{n-2}, \mathbf{x}^n, \mathbf{x}^{n-1}, \mathbf{x}^{n-2}) = 0$, where n refers to the time index. The solution of the non-linear system can then be obtained at each time interval using Newton iterations of the form:

$$\left[\frac{\partial \mathbf{R}(\mathbf{U}^k, \mathbf{x})}{\partial \mathbf{U}^k}\right] \delta \mathbf{U}^k = -\mathbf{R}(\mathbf{U}^k, \mathbf{x}) \tag{33}$$
$$\mathbf{U}^{k+1} = \mathbf{U}^k + \delta \mathbf{U}^k$$
$$\delta \mathbf{U}^k \to 0, \mathbf{U}^{k+1} = \mathbf{U}^n$$

Expanding a fine time-domain residual set about the coarse time domain set of residuals yields:

$$\mathbf{R}_h(\mathbf{U}_h, \mathbf{x}_h) = \mathbf{R}_h(\mathbf{U}_h^H, \mathbf{x}_h^H) + \left[\frac{\partial \mathbf{R}_h}{\partial \mathbf{U}_h}\right]_{\mathbf{U}_h^H, \mathbf{x}_h^H} \left(\mathbf{U}_h - \mathbf{U}_h^H\right) + \left[\frac{\partial \mathbf{R}_h}{\partial \mathbf{x}_h}\right]_{\mathbf{x}_h^H, \mathbf{U}_h^H} \left(\mathbf{x}_h - \mathbf{x}_h^H\right) + \cdots = 0 \tag{34}$$

and since the non-linear residual operator $\mathbf{R}$ must vanish for a converged solution at each time interval, an estimate for the error vector $(\mathbf{U}_h - \mathbf{U}_h^H)$ at each time interval in equation (32) can obtained by rearranging equation (34) as:

$$\left(\mathbf{U}_h - \mathbf{U}_h^H\right) \approx -\left[\frac{\partial \mathbf{R}_h}{\partial \mathbf{U}_h}\right]^{-1}_{\mathbf{U}_h^H, \mathbf{x}_h^H} \left\{ \mathbf{R}_h(\mathbf{U}_h^H, \mathbf{x}_h^H) + \left[\frac{\partial \mathbf{R}_h}{\partial \mathbf{x}_h}\right]_{\mathbf{x}_h^H, \mathbf{U}_h^H} \left(\mathbf{x}_h - \mathbf{x}_h^H\right) \right\} \tag{35}$$

It should be noted that this is merely an estimate for the error between $\mathbf{U}_h$ and $\mathbf{U}_h^H$ since higher-order terms in the Taylor expansion are ignored. Substituting equation (35) into equation (32) results in:

$$L_h(\mathbf{U}_h, \mathbf{x}_h) \approx L_h(\mathbf{U}_h^H, \mathbf{x}_h^H) - \tag{36}$$
$$\underbrace{\left[\frac{\partial L_h}{\partial \mathbf{U}_h}\right]_{\mathbf{U}_h^H, \mathbf{x}_h^H} \left[\frac{\partial \mathbf{R}_h}{\partial \mathbf{U}_h}\right]^{-1}_{\mathbf{U}_h^H, \mathbf{x}_h^H} \left\{ \mathbf{R}_h(\mathbf{U}_h^H, \mathbf{x}_h^H) + \left[\frac{\partial \mathbf{R}_h}{\partial \mathbf{x}_h}\right]_{\mathbf{x}_h^H, \mathbf{U}_h^H} \left(\mathbf{x}_h - \mathbf{x}_h^H\right) \right\} + \left[\frac{\partial L_h}{\partial \mathbf{x}_h}\right]_{\mathbf{x}_h^H, \mathbf{U}_h^H} \left(\mathbf{x}_h - \mathbf{x}_h^H\right)}_{\epsilon_{cc}}$$

The right-hand-side in equation (36) can be described as an estimate for the exact discrete functional evaluated directly on the fine time domain. Based on the derivation it is approximately equal to the sum of the functional evaluated on the fine time domain using a projected solution and projected mesh coordinates from the coarse domain and a computable correction term ϵ_{cc}.

The correction term ϵ_{cc} as defined above is:

$$\epsilon_{cc} = - \tag{37}$$
$$\left[\frac{\partial L_h}{\partial \mathbf{U}_h}\right]_{\mathbf{U}_h^H, \mathbf{x}_h^H} \left[\frac{\partial \mathbf{R}_h}{\partial \mathbf{U}_h}\right]^{-1}_{\mathbf{U}_h^H, \mathbf{x}_h^H} \left\{ \mathbf{R}_h(\mathbf{U}_h^H, \mathbf{x}_h^H) + \left[\frac{\partial \mathbf{R}_h}{\partial \mathbf{x}_h}\right]_{\mathbf{x}_h^H, \mathbf{U}_h^H} \left(\mathbf{x}_h - \mathbf{x}_h^H\right) \right\} + \left[\frac{\partial L_h}{\partial \mathbf{x}_h}\right]_{\mathbf{x}_h^H, \mathbf{U}_h^H} \left(\mathbf{x}_h - \mathbf{x}_h^H\right)$$

Since computing, storing and inverting the flow Jacobian matrix is expensive, a flow adjoint variable $\Lambda_{\mathbf{U}}$ is defined to aid the evaluation procedure. The flow adjoint variable is defined as:

$$(\Lambda_{\mathbf{U}})_h^T|_{\mathbf{U}_h^H \mathbf{x}_h^H} = -\left[\frac{\partial L_h}{\partial \mathbf{U}_h}\right]_{\mathbf{U}_h^H, \mathbf{x}_h^H} \left[\frac{\partial \mathbf{R}_h}{\partial \mathbf{U}_h}\right]^{-1}_{\mathbf{U}_h^H \mathbf{x}_h^H} \tag{38}$$

Transposing and rearranging equation (38) yields

$$\left[\frac{\partial \mathbf{R}_h}{\partial \mathbf{U}_h}\right]^T_{\mathbf{U}_h^H \mathbf{x}_h^H} \Lambda_{\mathbf{U}h}|_{\mathbf{U}_h^H, \mathbf{x}_h^H} = -\left[\frac{\partial L_h}{\partial \mathbf{U}_h}\right]^T_{\mathbf{U}_h^H, \mathbf{x}_h^H} \tag{39}$$

The solution of equation (39) involves projecting the coarse time domain solution and mesh coordinates onto the fine time domain, reconstructing the flow Jacobian matrices on the fine time domain and then solving the linear system iteratively for the flow adjoint variable. Since the goal is to avoid direct solutions of any nature on the fine time domain, an approximation is used where the adjoint is evaluated on the coarse time domain and then projected onto the fine domain. This circumvents the expensive evaluations on the fine time domain. Equation (39) recast on the coarse time domain becomes:

$$\left[\frac{\partial \mathbf{R}_H}{\partial \mathbf{U}_H}\right]^T_{\mathbf{U}_H, \mathbf{x}_H} \Lambda_{\mathbf{U}H} = -\left[\frac{\partial L_H}{\partial \mathbf{U}_H}\right]^T_{\mathbf{U}_H, \mathbf{x}_H} \tag{40}$$

The coarse adjoint variable can then be projected onto to the fine domain as:

$$\Lambda_{\mathbf{U}_h^H} = I_h^H \Lambda_{\mathbf{U}H} \tag{41}$$

where I_h^H is some appropriate projection operator. Once the vector of adjoint variables $\Lambda_{\mathbf{U}H}$ has been obtained, the first contributions to the computable correction term may be determined by projecting the flow adjoint onto the fine time domain and then evaluating a vector inner product as follows:

$$\epsilon_{cc_1} = (\Lambda_{\mathbf{U}_h^H})^T R_h(\mathbf{U}_h^H, \mathbf{x}_h^H) \tag{42}$$

ϵ_{cc_1} represents the contribution from the flow equations to the error arising due to insufficient temporal mesh resolution. The residual $\mathbf{R}_h(\mathbf{U}_h^H, \mathbf{x}_h^H)$ is non-zero since it is computed using the projection of the coarse time domain flow solution onto the fine time domain. Closer examination reveals that equation (42) is actually a summation of the vector inner products of the flow adjoint and the non-zero residual at each time-step on the fine time domain. The remaining contributions to the total correction term may be written in combined form as:

$$\epsilon_{cc_2} = \left\{(\Lambda_{\mathbf{U}})_h^T \left[\frac{\partial \mathbf{R}_h}{\partial \mathbf{x}_h}\right]_{\mathbf{U}_h^H \mathbf{x}_h^H} + \left[\frac{\partial L_h}{\partial \mathbf{x}_h}\right]_{\mathbf{x}_h^H}\right\} \left(\mathbf{x}_h - \mathbf{x}_h^H\right) = \lambda_{\mathbf{x}} \left(\mathbf{x}_h - \mathbf{x}_h^H\right) \tag{43}$$

using the notation $\lambda_{\mathbf{x}}$ as shorthand for the large bracketed term in the middle of the above equation. We now write a Taylor expansion of the mesh residual equation $\mathbf{G}_h$ on the fine time domain about its value constructed using projected mesh coordinates from the coarse to the fine domain as:

$$\mathbf{G}_h(\mathbf{x}_h) \;=\; \mathbf{G}_h(\mathbf{x}_h^H) + \left[\frac{\partial \mathbf{G}_h}{\partial \mathbf{x}_h}\right]_{\mathbf{x}_h^H} (\mathbf{x}_h - \mathbf{x}_h^H) + \cdots = 0 \tag{44}$$

Here the residual $\mathbf{G}_h(\mathbf{x}_h^H)$ is evaluated using the projected values for $\mathbf{x}_{int}$ from the coarse time domain to the fine domain and the exact values of $\mathbf{x}_{surf}$ on the fine time domain. The surface coordinates $\mathbf{x}_{surf}$ are defined as a prescribed function of time and can be easily evaluated on the fine time domain. Rearranging and simplifying the mesh residual Taylor expansion yields:

$$(\mathbf{x}_h - \mathbf{x}_h^H) = -\left[\frac{\partial \mathbf{G}_h}{\partial \mathbf{x}_h}\right]^{-1}_{\mathbf{x}_h^H} \mathbf{G}_h(\mathbf{x}_h^H) \tag{45}$$

However, referring to equation (31) it is seen that the linearization of the mesh residual $\mathbf{G}$ with respect to the mesh coordinates $\mathbf{x}$ is simply the mesh stiffness matrix $[\mathbf{K}]$ multiplied by the local temporal element size Δt. Therefore,

$$(\mathbf{x}_h - \mathbf{x}_h^H) = -\left(\frac{1}{\Delta t}\right)[\mathbf{K}]^{-1}\mathbf{G}_h(\mathbf{x}_h^H) \tag{46}$$

Substituting this back into equation (43) gives us an expression for the second and final contribution to the computable correction term as:

$$\epsilon_{cc_2} = -\lambda_{\mathbf{x}_h}\left(\frac{1}{\Delta t}\right)[\mathbf{K}]^{-1}\mathbf{G}_h(\mathbf{x}_h^H) \tag{47}$$

Defining a mesh adjoint variable $\Lambda_{\mathbf{x}}$ permits an iterative solution and avoids inversion of the stiffness matrix.

$$[\mathbf{K}]^T\Lambda_{\mathbf{x}h} = -\left(\frac{1}{\Delta t}\right)(\lambda_{\mathbf{x}h})^T \tag{48}$$

As in the case of the flow adjoint, the above system is solved on the coarse time domain and the coarse mesh adjoint variable then projected onto the fine time domain by some appropriate operator as:

$$[\mathbf{K}]^T\Lambda_{\mathbf{x}H} = -\left(\frac{1}{\Delta t}\right)(\lambda_{\mathbf{x}H})^T \tag{49}$$

$$\Lambda_{\mathbf{x}_h}^{\ H} = I_h^H\Lambda_{\mathbf{x}H} \tag{50}$$

The compact space-time notation used in the above derivation obscures the details of the adjoint solution process. This involves a backward sweep in time during which both a flow adjoint variable vector and a mesh adjoint variable vector are solved for at each time-step. For example, expanding the flow adjoint equation (c.f. equation (40)) in the time domain (and dropping the subscripts for clarity) we obtain:

$$\begin{bmatrix} \left[\frac{\partial \mathbf{R}^1(\mathbf{U})}{\partial \mathbf{U}^1}\right] & & & & 0 & \\ \left[\frac{\partial \mathbf{R}^2(\mathbf{U})}{\partial \mathbf{U}^1}\right] & \left[\frac{\partial \mathbf{R}^2(\mathbf{U})}{\partial \mathbf{U}^2}\right] & & & & \\ \left[\frac{\partial \mathbf{R}^3(\mathbf{U})}{\partial \mathbf{U}^1}\right] & \left[\frac{\partial \mathbf{R}^3(\mathbf{U})}{\partial \mathbf{U}^2}\right] & \left[\frac{\partial \mathbf{R}^3(\mathbf{U})}{\partial \mathbf{U}^3}\right] & & & \\ & & \ddots & \ddots & & \\ 0 & & \left[\frac{\partial \mathbf{R}^{n-1}(\mathbf{U})}{\partial \mathbf{U}^{n-3}}\right] & \left[\frac{\partial \mathbf{R}^{n-1}(\mathbf{U})}{\partial \mathbf{U}^{n-2}}\right] & \left[\frac{\partial \mathbf{R}^{n-1}(\mathbf{U})}{\partial \mathbf{U}^{n-1}}\right] & \\ & & & \left[\frac{\partial \mathbf{R}^{n}(\mathbf{U})}{\partial \mathbf{U}^{n-2}}\right] & \left[\frac{\partial \mathbf{R}^{n}(\mathbf{U})}{\partial \mathbf{U}^{n-1}}\right] & \left[\frac{\partial \mathbf{R}^{n}(\mathbf{U})}{\partial \mathbf{U}^{n}}\right] \end{bmatrix}^T \begin{bmatrix} \Lambda^1 \\ \Lambda^2 \\ \Lambda^3 \\ \vdots \\ \Lambda^{n-1} \\ \Lambda^n \end{bmatrix} = -\begin{bmatrix} \frac{\partial L}{\partial \mathbf{U}^1} \\ \frac{\partial L}{\partial \mathbf{U}^2} \\ \frac{\partial L}{\partial \mathbf{U}^3} \\ \vdots \\ \frac{\partial L}{\partial \mathbf{U}^{n-1}} \\ \frac{\partial L}{\partial \mathbf{U}^n} \end{bmatrix}^T \tag{51}$$

since each residual depends on the current and previous two time instances. When transposed we obtain

$$\begin{bmatrix} \left[\frac{\partial \mathbf{R}^1(\mathbf{U})}{\partial \mathbf{U}^1}\right]^T & \left[\frac{\partial \mathbf{R}^2(\mathbf{U})}{\partial \mathbf{U}^1}\right]^T & \left[\frac{\partial \mathbf{R}^3(\mathbf{U})}{\partial \mathbf{U}^1}\right]^T & & & \\ & \left[\frac{\partial \mathbf{R}^2(\mathbf{U})}{\partial \mathbf{U}^2}\right]^T & \left[\frac{\partial \mathbf{R}^3(\mathbf{U})}{\partial \mathbf{U}^2}\right]^T & \left[\frac{\partial \mathbf{R}^4(\mathbf{U})}{\partial \mathbf{U}^2}\right]^T & 0 & \\ & & \ddots & \ddots & & \\ & & & \left[\frac{\partial \mathbf{R}^{n-2}(\mathbf{U})}{\partial \mathbf{U}^{n-2}}\right]^T & \left[\frac{\partial \mathbf{R}^{n-1}(\mathbf{U})}{\partial \mathbf{U}^{n-2}}\right]^T & \left[\frac{\partial \mathbf{R}^{n}(\mathbf{U})}{\partial \mathbf{U}^{n-2}}\right]^T \\ & 0 & & & \left[\frac{\partial \mathbf{R}^{n-1}(\mathbf{U})}{\partial \mathbf{U}^{n-1}}\right]^T & \left[\frac{\partial \mathbf{R}^{n}(\mathbf{U})}{\partial \mathbf{U}^{n-1}}\right]^T \\ & & & & & \left[\frac{\partial \mathbf{R}^{n}(\mathbf{U})}{\partial \mathbf{U}^{n}}\right]^T \end{bmatrix} \begin{bmatrix} \Lambda^1 \\ \Lambda^2 \\ \Lambda^3 \\ \vdots \\ \Lambda^{n-1} \\ \Lambda^n \end{bmatrix} = - \begin{bmatrix} \frac{\partial L}{\partial \mathbf{U}^1} \\ \frac{\partial L}{\partial \mathbf{U}^2} \\ \frac{\partial L}{\partial \mathbf{U}^3} \\ \vdots \\ \frac{\partial L}{\partial \mathbf{U}^{n-1}} \\ \frac{\partial L}{\partial \mathbf{U}^{n}} \end{bmatrix}^T \tag{52}$$

which can be solved by back substitution, at a typical time step k as

$$\left[\frac{\partial \mathbf{R}^k(\mathbf{U})}{\partial \mathbf{U}^k}\right]^T \Lambda^k = \frac{\partial L}{\partial \mathbf{U}^k} - \left[\frac{\partial \mathbf{R}^{k+1}(\mathbf{U})}{\partial \mathbf{U}^k}\right]^T \Lambda^{k+1} - \left[\frac{\partial \mathbf{R}^{k+2}(\mathbf{U})}{\partial \mathbf{U}^k}\right]^T \Lambda^{k+2} \tag{53}$$

A similar approach is adopted for the solution of the mesh adjoint equation at each time step. The final form for the second contribution to the correction term can now be expressed as:

$$\epsilon_{cc_2} = (\Lambda_{\mathbf{x}_h}^H)^T \mathbf{G}_h(\mathbf{x}_h^H) \tag{54}$$

ϵ_{cc_2} represents the contribution from the mesh motion equations to the temporal resolution error. Just as in the case of the temporal resolution error due to the flow equations, equation (54) represents a summation of vector products of the mesh adjoint and the mesh residual at each time interval. Although the mesh motion equations are linear, the resulting mesh coordinate variations in time are not linear, since these are driven by the prescribed surface mesh displacements, which in our following examples are non-linear in time. This causes the mesh residual to be non-zero when computed on the fine time domain using projected mesh coordinates from the coarse time domain.

The contribution to the total correction from each individual time-step can be interpreted as the representation of the error in the functional arising from that time interval. From the derivation it is clear that there are two distinct sources of error, specifically the flow and the mesh that contribute to the total temporal resolution error. The resulting distribution in time of the total error ϵ_{cc} can thus be conveniently used as the criteria for identifying regions that require higher temporal resolution in order to drive an adaptive time-step selection strategy.

2.2 Error due to partial convergence

Consider the solution of the problem on the coarse time domain. Previously we assumed that the solutions on the coarse time domain were obtained by full convergence of the flow and mesh motion equations. This translates into the flow residual and the mesh residual equations evaluating to zero on the coarse time domain. However, we now consider the functional L evaluated on the coarse time domain using partially converged solutions. If the flow and mesh equations were partially converged on the coarse time domain, the error resulting in the functional as a consequence may be linearly approximated as:

$$L_H(\mathbf{U}_H, \mathbf{x}_H) - L_H(\bar{\mathbf{U}}_H, \bar{\mathbf{x}}_H) \approx \left[\frac{\partial L_H}{\partial \mathbf{U}_H}\right]_{\bar{\mathbf{U}}_H, \bar{\mathbf{x}}_H} (\mathbf{U}_H - \bar{\mathbf{U}}_H) + \left[\frac{\partial L_H}{\partial \mathbf{x}_H}\right]_{\bar{\mathbf{x}}_H, \bar{\mathbf{U}}_H} (\mathbf{x}_H - \bar{\mathbf{x}}_H) \tag{55}$$

where $\bar{\mathbf{U}}_H$ and $\bar{\mathbf{x}}_H$ refer to the approximate values of the flow variables and mesh coordinates obtained through partial convergence of the respective equations. Evaluating the flow and mesh residual equations using partially converged solutions would result in non-zero residuals. The fully converged zero residuals can then be written using a Taylor expansion about the partially converged values as:

$$\begin{aligned}\mathbf{R}_H(\mathbf{U}_H,\mathbf{x}_H) &= \mathbf{R}_H(\bar{\mathbf{U}}_H,\bar{\mathbf{x}}_H)+\left[\frac{\partial \mathbf{R}_H}{\partial \mathbf{U}_H}\right]_{\bar{\mathbf{U}}_H,\bar{\mathbf{x}}_H}(\mathbf{U}_H-\bar{\mathbf{U}}_H)+\left[\frac{\partial \mathbf{R}_H}{\partial \mathbf{x}_H}\right]_{\bar{\mathbf{x}}_H,\bar{\mathbf{U}}_H}(\mathbf{x}_H-\bar{\mathbf{x}}_H)+\cdots=0\\ \mathbf{G}_H(\mathbf{x}_H) &= \mathbf{G}_H(\bar{\mathbf{x}}_H)+\left[\frac{\partial \mathbf{G}_H}{\partial \mathbf{x}_H}\right]_{\bar{\mathbf{x}}_H}(\mathbf{x}_H-\bar{\mathbf{x}}_H)+\cdots=0\end{aligned}$$

At this point the similarity with the derivation of the temporal resolution error becomes clear. Following the same procedure as the temporal resolution error derivation, we can obtain two contributions that add up to the total error due to partial convergence as:

$$\epsilon_{cc_{1p}} = (\Lambda_{\mathbf{U}H})^T\mathbf{R}(\bar{\mathbf{U}}_H,\bar{\mathbf{x}}_H) \quad (56)$$

$$\epsilon_{cc_{2p}} = (\Lambda_{\mathbf{x}H})^T\mathbf{G}(\bar{\mathbf{x}}_H) \quad (57)$$

where the adjoint variables are solved for as:

$$\left[\frac{\partial \mathbf{R}_H}{\partial \mathbf{U}_H}\right]^T_{\bar{\mathbf{U}}_H,\bar{\mathbf{x}}_H}\Lambda_{\mathbf{U}H} = -\left[\frac{\partial L_H}{\partial \mathbf{U}_H}\right]^T_{\bar{\mathbf{U}}_H,\bar{\mathbf{x}}_H} \quad (58)$$

$$[\mathbf{K}]^T\Lambda_{\mathbf{x}H} = -\left(\frac{1}{\Delta t}\right)(\lambda_{\mathbf{x}H})^T \quad (59)$$

$$with\ \ \lambda_{\mathbf{x}H} = \left\{(\Lambda_{\mathbf{U}H})^T\left[\frac{\partial \mathbf{R}_H}{\partial \mathbf{x}_H}\right]_{\bar{\mathbf{U}}_H,\bar{\mathbf{x}}_H}+\left[\frac{\partial L_H}{\partial \mathbf{x}_H}\right]_{\bar{\mathbf{x}}_H,\bar{\mathbf{U}}_H}\right\} \quad (60)$$

Closer examination reveals that these systems are nearly identical to equations (40) and (49) with the only difference being that all quantities are evaluated using the partially converged flow and mesh solutions on the coarse time domain, rather than the fully converged values. In the case of the temporal resolution error, the adjoint variables were solved on the coarse time domain using the fully converged coarse time domain flow solution and mesh coordinates. These were then projected onto the fine time domain and multiplied with the non-zero residuals constructed on the fine time domain. The residuals on the fine time domain were non-zero due to the projection of the flow solution and mesh coordinates from the coarse to the fine time domain. In the case of partial convergence on the coarse time domain, there are no projections involved and the error due to partial convergence on the coarse time domain can be estimated directly by multiplying the non-zero partially converged coarse residuals with the coarse adjoint variables. The distributions in time of $\epsilon_{cc_{1p}}$ and $\epsilon_{cc_{2p}}$ arise from the error due to partial convergence of the flow and mesh equations respectively and can be used to drive adaptation of the convergence tolerances of the corresponding disciplines.

2.3 Combined Error

In a practical simulation, the time-dependent solution is only partially converged at each time step. Thus, the resulting approximate solution $\tilde{\mathbf{U}}_h,\tilde{\mathbf{x}}_h$ obtained by projecting the

partially converged coarse level solution to the finer level contains both temporal and algebraic error. However, the algebraic error can be estimated separately based on the equations derived in the previous section. The temporal error is then obtained by subtracting this algebraic error estimate from the combined error obtained on the finer temporal discretization. While interactions between these two error types are certainly possible, these effects cannot be accounted using an adjoint approach, since the adjoint formulation is based on a linear analysis which is consistent with the assumption of additive error behavior.

2.4 Solver details

The adaptive solver consists of two parts namely the forward flow solver and backward adjoint solver. The projection and error computation routines are built into the adjoint solver. The flow solver performs the forward integration in time and writes out the flow solution, mesh solution and partially converged residual values to the hard drive in a piecewise manner (i.e. at each time level). The adjoint solver reads in the flow and mesh solutions and performs the backward sweep in time to compute the flow and adjoint variables at each time level. The adjoint solver also reads in the non-zero partially converged flow and mesh residuals written out by the flow solver and multiplies them with the corresponding adjoint variables to determine the algebraic partial convergence error at each time-step. A simple linear interpolation of the flow solution, mesh coordinates and the adjoint variables on the coarse time domain is used as the projection operator to determine unknown values on the fine time domain. We use a refinement ratio of 2-to-1 where each temporal element in the coarse level temporal mesh is divided into two. Although more advanced interpolation techniques such as third- and fifth-order splines may be used as the projection operator, it has been determined through our own work that they do not offer any significant advantages in terms of gain in accuracy compared to a simpler linear interpolation. The adjoint solver also constructs the non-zero residuals resulting from the projected solutions from the coarse to the fine time levels and multiplies them with the corresponding projected adjoint variables to determine the total error (resolution+algebraic) at each time level. The algebraic or partial convergence error at each time-step is then subtracted out of the total error in order to obtain the temporal resolution error.

2.5 Verification of method

The verification of the method is done in several steps where each component of the computed total error is individually verified. The process can be broken down as follows.

1. Verify temporal resolution error without the effect of partially converged solutions.
2. Verify error due to partially converged flow equations without the effect of partially converged mesh equations or temporal resolution.
3. Similarly verify error due to partially converged mesh equations.
4. Verify total combined error which includes temporal resolution and partial convergence of flow and mesh equations.

2.5.1 Temporal resolution error

Since the method computes a linear approximation of the error between temporal meshes successively refined using a ratio of 2-to-1, it would be wise to choose a test problem that exhibits minimum non-linear behavior with respect to increasing temporal resolution for the purpose of verification. Our approach is to consider a sinusoidally pitching NACA64A010 airfoil and zoom into a small temporal window in the pitch cycle to use as our time domain of interest. The purpose of this is to reduce the effect of non-linear behavior as the temporal resolution is increased. The conditions chosen for the problem include a free-stream Mach number of 0.8, a zero mean angle-of-attack, an amplitude of pitch of 5 degrees, a reduced frequency of 0.1 and a pitch center located two chord lengths ahead of the leading edge of the airfoil. The time domain we consider is the first quarter of the first pitch period starting from a steady-state solution. The functional of interest is the time-integrated lift coefficient given as

$$L = \sum_{n=1}^{n_{steps}} \Delta t_n C_{Ln} \tag{61}$$

The coarsest temporal mesh consists of 8 temporal elements of uniform size and fine meshes are constructed by successive nested subdivision of these 8 elements. The finest mesh consists of 128 elements. In order to remove the effect of partially converged solutions and to verify only the error due to temporal resolution, all of the equations (governing and corresponding adjoint) are converged to machine precision. At each temporal resolution we compare against the exact discrete functional computed directly on the next temporal mesh that has twice the resolution. Table 1 shows the results from this study and indicates the expected trends. As the temporal resolution is increased, the ratio between the predicted and exact errors asymptotically approaches unity.

2.5.2 Flow partial convergence error

In order to verify the error only due to partial convergence of the flow equations, we fully converge the mesh motion equations and partially converge the flow equations (to a tolerance of $1e^{-5}$) while performing all computations on a single temporal mesh of

Table 1: Verification of temporal resolution error prediction

Flow/Mesh Tol.	H/h	$L_h(\mathbf{U}_h, \mathbf{x}_h)$	$L_h(\mathbf{U}_h^H, \mathbf{x}_h^H)$	Exact error	Predicted Error	Ratio
2e-14/1e-15	8/16	4.762774807543	4.741995206312	0.020779601231	0.022229938649	1.069796210333
2e-14/1e-15	16/32	4.676917052941	4.660231481605	0.016685571335	0.016163145847	0.968689985072
2e-14/1e-15	32/64	4.635246667094	4.625792451272	0.009454215822	0.009450398778	0.999596260174
2e-14/1e-15	64/128	4.614970587970	4.609729346486	0.005241241484	0.005240241484	0.999809205510

Table 2: Verification of flow partial convergence error prediction

Flow/Mesh Tol.	H	$L_H(\mathbf{U}_H, \mathbf{x}_H)$	$L_H(\mathbf{U}_H, \mathbf{x}_H)$	Exact error	Predicted Error	Ratio
1e-5/1e-15	8	4.947790735115	4.933513225068	0.014277510046	0.014301503945	0.998322281441

Table 3: Verification of mesh partial convergence error prediction

Flow/Mesh Tol.	H	$L_H(\mathbf{U}_H, \mathbf{x}_H)$	$L_H(\mathbf{U}_H, \bar{\mathbf{x}}_H)$	Exact error	Predicted Error	Ratio
2e-14/1e-5	8	4.947790735115	4.947770355040	2.038007523e-5	2.063119286e-5	1.012321722715

Table 4: Verification of combined temporal resolution and partial convergence error prediction

Flow/Mesh Tol.	H/h	$L_h(\mathbf{U}_h, \mathbf{x}_h)$	$L_h(\tilde{\mathbf{U}}_h^H, \bar{\mathbf{x}}_h^H)$	Exact error	Predicted Error	Ratio
1e-5/1e-4	2/4	5.328712532500	5.236186429207	0.092526103292	0.089741068685	0.969900011908
1e-6/1e-5	4/8	4.947790735115	4.914257485675	0.033533249440	0.036424973742	1.086234538864
1e-7/1e-6	8/16	4.762774807543	4.741995206312	0.020779601231	0.022229938632	1.069796209509
1e-8/1e-7	16/32	4.676917052940	4.660231481605	0.016685571335	0.016163145822	0.968689983568
1e-9/1e-8	32/64	4.635246667094	4.625792451272	0.009454215822	0.009450398727	0.999596254759
1e-10/1e-9	64/128	4.614904286716	4.609729346486	0.005174940229	0.005173852683	0.999789843672

some arbitrary resolution. This is done in order to remove the effect of mesh partial convergence and temporal resolution. The chosen temporal mesh resolution consists of 8 uniform temporal elements. The comparison is made against a functional computed on the same temporal mesh by fully converging both the flow and mesh equations. Table 2 shows the results from the study. The predicted error due to partial convergence is very close to the exact error as indicated by the near unity of the ratio between the two.

2.5.3 Mesh partial convergence error

The method is nearly identical to that of verifying the flow partial convergence error, except now the flow equations are fully converged while the mesh equations are partially converged to a tolerance of $1e^{-5}$. Table 3 shows the results from the study and once again the near unity of the ratio between the predicted and exact errors establishes confidence in the method.

2.5.4 Combined total error due to resolution and partial convergence

In this study, we successively refine the temporal resolution and simultaneously tighten the convergence tolerances from loosely set values in order to verify the total predicted error. The coarsest temporal mesh consists of 2 equally sized temporal elements while the finest mesh consists of 128 elements. The convergence tolerances for the flow equations and mesh equations begin at $1e^{-5}$ and $1e^{-4}$ respectively on the coarsest temporal mesh and are tightened by an order-of-magnitude during each refinement of the temporal resolution. These starting limits were chosen such that the algebraic and temporal errors are roughly the same order-of-magnitude so as to not mask any discrepancies from either. The comparison at each temporal resolution is made against the functional computed using full convergence of flow and mesh equations on the next temporal mesh of double the resolution. It is important that the method be verified for the total combined error prediction, since it is to be applied in this form to adaptation problems in general. Table 4 shows the results from this study and indicates the expected trend of more accurate

predictions as the resolution is increased while simultaneously convergence tolerances are tightened.

2.6 Adaptation Strategy

The adjoint error estimation is now used to drive an adaptive time-step and convergence tolerance strategy. The overall adaptive simulation procedure begins with a forward integration in time to determine the solution to the analysis problem and a backward sweep in time to compute the time-dependent adjoint variables and the corresponding necessary error distributions (temporal and algebraic). The sum of the individual error distributions gives us an estimate of the correction from each type of error. The temporal elements are first sorted in decreasing order by their contribution to the total error of each type. Then parsing down each of the lists, elements are flagged for refinement or tolerance tightening until 99% of the total error of the type under consideration is reached. The method assures that only the most offending elements are targeted in the case of an extremely non-uniform error distribution. The method also assures that near uniform refinement or tolerance tightening is requested once the error in the temporal domain has been equidistributed. Elements flagged for refinement are subdivided into two, while elements flagged for algebraic error have their convergence tolerances tightened by a factor of three and a global convergence tolerance for all mesh elements for each time step is determined (for the flow and mesh disciplines separately).

2.7 Comparative Methods

The adjoint-driven adaptive strategy is compared against the more commonly used but basic method of uniform temporal refinement until acceptable error levels are achieved, and the more sophisticated method of utilizing temporal discretizations of varying orders-of-accuracy for estimating local temporal error.

The uniform refinement method we employ involves the doubling of the temporal mesh resolution at each adaptive cycle using a 2-to-1 nested subdivision of temporal elements, while simultaneously tightening the convergence tolerances by a factor of three for all temporal elements. As for the estimation of local temporal error, the method proposed in reference [3] is used. The solution to the analysis problem obtained using the second-order accurate BDF2 scheme is used to determine the temporal derivative term discretized based on a third-order accurate BDF3 time-integration scheme. The estimate of the local error at each time-step is then computed as

$$e_{local} = \left| \left[\frac{d\mathbf{U}}{dt} \right]_{BDF3} - \left[\frac{d\mathbf{U}}{dt} \right]_{BDF2} \right| \tag{62}$$

The adaptation for this method is based on refining temporal elements that have local error higher than the computed mean of the local error distribution in the time domain. The convergence tolerances at each time-step are set to be equal to that of the local temporal error in order to ensure that the algebraic error is less than or equal to the temporal discretization error. As a convention, the cost shown in the comparative plots at any particular adaptive cycle regardless of the adaptation method employed includes the cost

of all preceding adaptive cycles. All adaptation methods are compared in reference to numerically exact results computed using a uniform time domain of resolution at least two orders-of-magnitude higher than those achieved through adaptation. Therefore, the reference solution corresponds to the solution with vanishing temporal and algebraic error, and the same spatial error as the test solutions.

2.8 Time-Dependent Test Case

The performance of the adjoint-driven adaptive method is demonstrated on a sample test problem consisting of a convecting vortex in the vicinity of a pitching airfoil. This is a time-dependent problem that includes mesh deformation through the action of the pitching airfoil. The airfoil is a NACA0012 undergoing very slow pitching at a reduced frequency of 0.001 in a flow with a freestream Mach number of 0.4225. The amplitude of pitch is 5^o with the pitch center located at the quarter chord of the airfoil. Figure 3 shows the computational spatial mesh and a zoomed in view of the airfoil within the mesh used for this example. The mesh consists of approximately 8,600 elements. The vortex is seeded in the flow 15 chord lengths ahead and 2 chord lengths below the airfoil, from where it is allowed to convect with the freestream and interact with the airfoil. Figure 4 shows the density contours of the initial condition for the flow in the spatial domain. Figure 5 shows the time variation of the lift coefficient of the airfoil with and without vortex interaction. The size of the time domain was chosen such that at the end of the time-integration process the vortex is located approximately 15 chord lengths downstream of the airfoil. The functional of interest is the time-integrated value of the lift coefficient as described in equation (61). The effect of the vortex on the airfoil should in theory increase as it approaches the airfoil and diminish rapidly once it passes the airfoil. Engineering judgement tells us that in order to correctly predict the effect of the vortex on the airfoil load, it must be convected to the region of interaction with the least amount of dissipation. Both temporal and spatial errors contribute toward dissipating the vortex as it convects through the domain. The sources of spatial error include the spatial discretization order-of-accuracy and the resolution of the spatial mesh. However, since all comparisons are made on the same spatial mesh with identical spatial discretization, the spatial error can be considered to be independent of the temporal error. Our goal is to identify and reduce the effect of temporal error on the functional. The temporal error relevant to the functional most likely occurs in the region of the temporal domain when the vortex is still ahead of the airfoil, and also in the region when it interacts with the airfoil. It is likely that the temporal error once the vortex passes the airfoil is not relevant to the functional. The time-integration must therefore use smaller time-steps while the vortex is still ahead of the airfoil in order to ensure minimal dissipation of the vortex, and smaller time-steps have to be used during the interaction phase in order to ensure that all of the details in the perturbation of the airfoil lift are captured. The adjoint method being goal-based should predict the error distribution in accordance with this expectation, while local error based adaptation will likely request refinement of the entire time domain. This happens as a consequence of the vortex continuing to convect and dissipate downstream of the airfoil, although its effect is not relevant to the load on the airfoil. Therefore local error based adaptation is likely to perform no better than uniform refinement of the whole time domain.

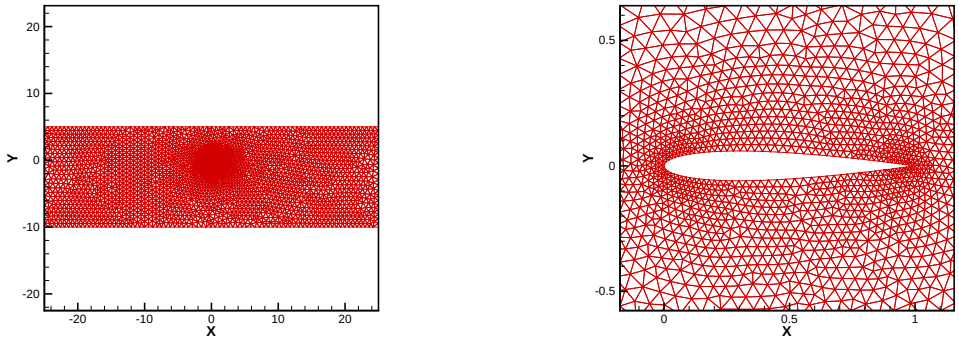


Figure 3: Computational mesh consisting of approximately 8,600 elements; (far-field view (left), close up near airfoil (right)

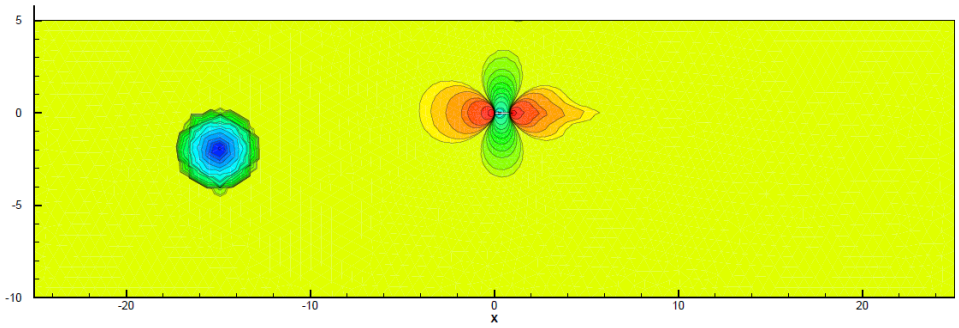


Figure 4: Density contours of initial condition

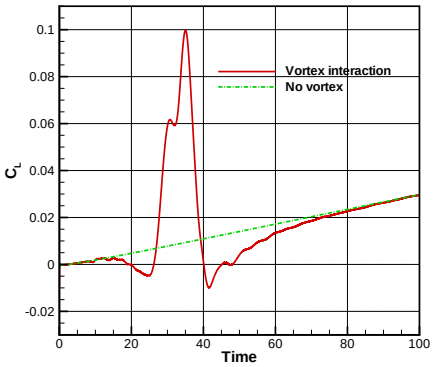


Figure 5: Time variation of airfoil lift coefficient

Figures 6.1 and 6.2 show the convergence of the functional and functional error with respect to the temporal mesh resolution, while Figures 7.1 and 7.2 compare the same with the computational expense. Additionally, the corrected functional values obtained by adding the computed error estimates to the current functional value are plotted in these figures. For this particular case, adjoint-based adaptation is able to achieve similar error levels as uniform refinement and local error based adaptation with fewer than half the number of time-steps. From a cost perspective, the plots indicate that at least initially the expense is similar between all methods but after two to three adaptive cycles, adjoint-based adaptation is able to outperform uniform refinement and local error based adaptation. The plots also indicate the expected trend of local error based adaptation performing similarly to uniform refinement. Figures 8 compares the time-step size distribution between adjoint-based and the local error based method after termination of the adaptation. The adjoint-based method has not touched the region in the time domain when the vortex has passed the airfoil, while local error based adaptation has requested fairly small time-step sizes in this region. On the other hand, significant adaptation of the convergence tolerances occurs in the region when the vortex is already downstream of the airfoil as shown in Figure 9. This is a good example where engineering judgement in lieu of a mathematically rigorous procedure to determine convergence tolerances could potentially lead to poor results.

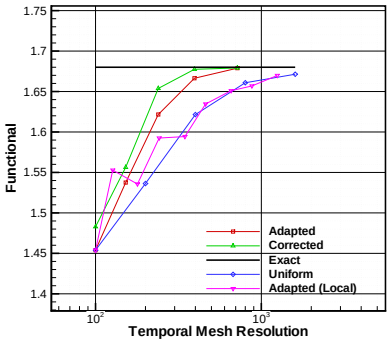
3 Combined Spatial Temporal and Algebraic Error Estimation and Control

3.1 Motivation

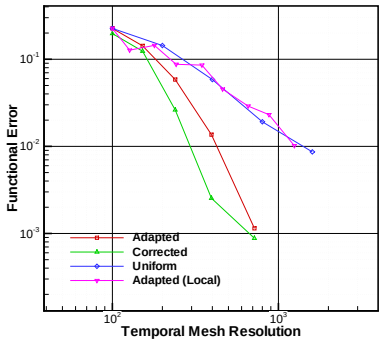
In the previous section we considered the estimation and adaptive control of temporal and algebraic error for time-dependent flow problems with dynamically moving meshes. In this section we now consider the additional effect of spatial discretization error estimation and control through adaptive mesh refinement. We focus on the single discipline given by the flow equations and omit the capability of including dynamically deforming meshes. Obviously, in order to enable more accurate time-dependent simulations for realistic problems, all three error sources must be considered simultaneously. This is important since in general the relative magnitudes of these various error types are not known a priori and an automated error estimation technique is crucial for identifying and controlling the dominant error sources.

Although adjoint-driven error estimation for time-dependent problems can be an expensive proposition, since this requires storage of the entire time history and backward time-integration operation, an advantage of the present formulation is that a single adjoint solution can be used to estimate all the various error sources under consideration. Thus the cost of the adjoint solution can be amortized over the various error estimates and accuracy gains can be increased by targeting the most important error components.

In order to illustrate the various error contributions to a time-dependent flow problem, we construct a simple convecting vortex problem with an objective that has an exact solution. The objective consists of the time integrated density inside a region of the flow field as a 2-D inviscid isentropic vortex [5, 25] convects through uniform flow. In this case,

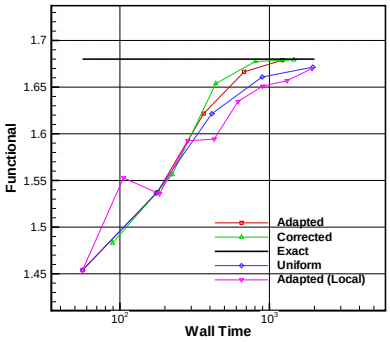


6.1: Functional convergence

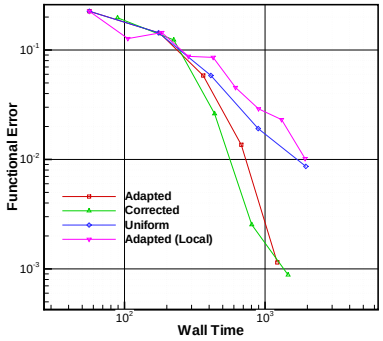


6.2: Functional error convergence

Figure 6: Convergence of functional and error with respect to temporal mesh resolution



7.1: Functional convergence



7.2: Functional error convergence

Figure 7: Convergence of functional and error with respect to cost in terms of wall time

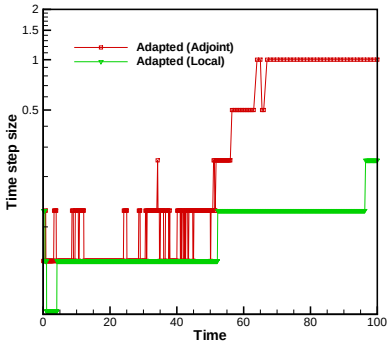


Figure 8: Comparison of time-step size distribution for adjoint-based and local error based adaptation after final adaptive cycle

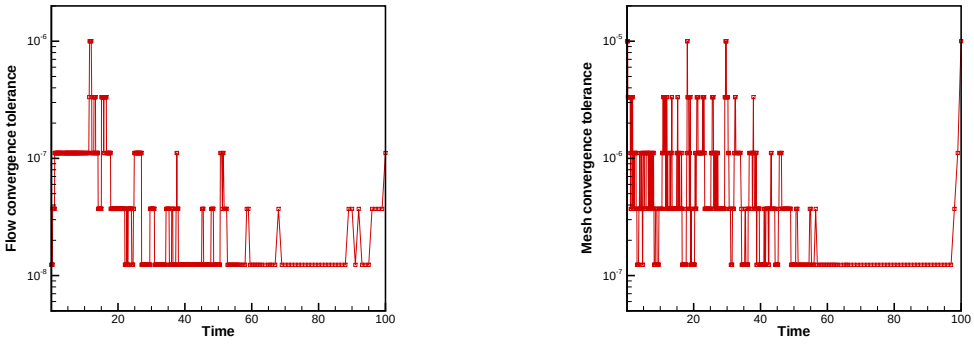


Figure 9: Flow (left) and mesh (right) convergence tolerances after final adaptive cycle

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 [24] 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:

$$\begin{aligned} \delta u &= -\frac{V_m}{R_c}(y - y_o)\, e^{\left[\frac{1}{2}-\left(\frac{r}{2R_c}\right)^2\right]} \\ \delta v &= \frac{V_m}{R_c}(x - x_o)\, e^{\left[\frac{1}{2}-\left(\frac{r}{2R_c}\right)^2\right]} \\ \delta T &= -\frac{V_m^2(\gamma-1)}{2\gamma R_c^2} e^{\left[1-\left(\frac{r}{R_c}\right)^2\right]} \end{aligned} \tag{63}$$

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

$$\rho = T^{1/(\gamma-1)} = (T_\infty + \delta T)^{1/(\gamma-1)} \tag{64}$$

In this case, the strength and size of the vortex 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 validate the results, a single quantity is needed as the objective function, and for this we define the objective 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, given as:

$$L(U) = \int_0^{60}\int_{-1}^{1}\int_{-1}^{1} \rho \; \mathrm{d}x\mathrm{d}y\mathrm{d}t \tag{65}$$

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 with the prescribed initial condition and highlighted integration

region is shown in Figure 10. 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 the subsequent analysis.

The importance of considering all three error sources can be demonstrated through this test case. For example, for a steady-state problem, using a consistent discretization, continually refining the mesh results in a progressive reduction of the overall error. However, this is not the case for time-dependent simulations, since at some level the temporal and algebraic errors become dominant for fine spatial discretizations. This can be seen in Figure 11(left) where the error in the objective versus the exact solution for the above described test case is plotted as the mesh is isotropically refined four times while holding the time steps constant at 16, 32, 64, 128 and 256 steps to march through the 60 time units of the full simulation. For the larger time step size (smaller number of steps) it is evident that refining the mesh past a certain point does not adequately reduce the total error with respect to the true solution because the temporal error is the dominant error. A similar test was carried out with fixed grid resolution and isotropically refined time steps and plotted in Figure 11(center). Again, the limitations of refining in only one dimension (time) are evident. When a grid of only 1176 elements is used for each time step there is almost no improvement in solution accuracy by using more than 64 time steps to complete the time accurate simulation unless the mesh resolution is increased to 4704 elements.

In the previous two examples we converged the solution to a sufficiently small residual to eliminate the error resulting from partial convergence from the results. If instead, we vary the convergence tolerance while holding the number of grid elements at each step constant for all equal spaced time steps, we can again see how the largest error term will dominate the solution. In rightmost part of Figure 11 we have plotted the relative error with respect to the true solution for simulations isotropically refined in both space and time (i.e 16 steps with 1176 cells, 32 steps with 4704 cells, etc.) for varying levels of convergence tolerance from 10^{-1} to 10^{-14}.

From this we can see that at each space and time resolution we eventually reach a point where converging the solution further does not get us closer to the true solution (i.e. we are converging to the wrong solution). The only way to get closer to the solution is to increase the spatial and temporal resolution such as going from 150624 total degrees of freedom to 1204224 and even more.

3.2 General formulation for multiple error sources

The previous example demonstrates the importance of estimating all error sources and enabling the simultaneous adaptive control of these errors. As mentioned previously, due to the computational expense of solving the adjoint problem, it is desirable to be able to estimate all error types using a single adjoint solution. In the following, we derive the adjoint-based estimates for spatial, temporal and algebraic errors, summarizing the results of the previous section for temporal and algebraic errors in the absence of deforming meshes. We then describe how these error estimates can be obtained using a single adjoint solution and demonstrate simultaneous adaptive error control of all three error types.

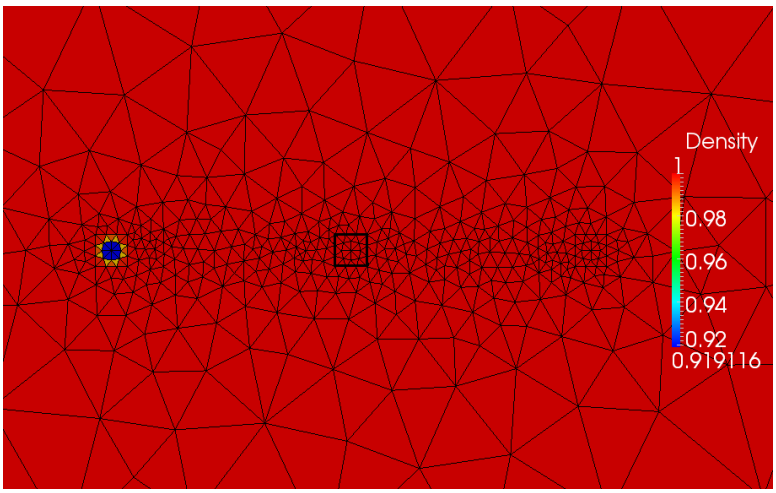


Figure 10: Initial density distribution with highlighted integrated region for grid of 1176 elements

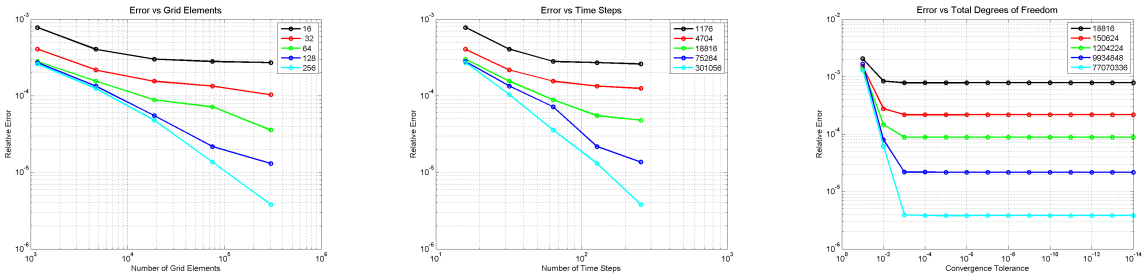


Figure 11: (left) Effect of spatial refinement on accuracy when temporal resolution is held constant at 16, 32, 64, 128 and 256 time steps; (center) Effect of temporal refinement on accuracy when spatial resolution is held constant at 1176, 4704, 18816, 75264 and 301056 mesh elements per time step; (right) Effect of convergence tolerance on accuracy for 5 iso-tropically refined time and space domains starting with 16 equal time steps of 1176 mesh elements producing 18816 total elements.

3.2.1 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 $L_s(\mathbf{U}_s)$.

$$L_s(\mathbf{U}_s) = L_s(\tilde{\mathbf{U}}_s) + \frac{\partial L_s(\tilde{\mathbf{U}}_s)}{\partial \tilde{\mathbf{U}}_s}\left(\mathbf{U}_s - \tilde{\mathbf{U}}_s\right) + \cdots \tag{66}$$

where $L_s(\tilde{\mathbf{U}}_s)$ is the fine grid objective computed using an approximate solution $\tilde{\mathbf{U}}_s$. In general, L_s may be a time integrated objective and will depend on the entire time history of the solution $\mathbf{U}_s$. 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 $\mathbf{U}_s$ on the fine mesh in the proceeding equation we will use a Taylor series expansion about the fine level residual equation

$$\mathbf{R}_s(\mathbf{U}_s) = 0 = \mathbf{R}_s(\tilde{\mathbf{U}}_s) + \frac{\partial \mathbf{R}_s(\tilde{\mathbf{U}}_s)}{\partial \tilde{\mathbf{U}}_s}\left(\mathbf{U}_s - \tilde{\mathbf{U}}_s\right) + \cdots \tag{67}$$

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 $\left(\mathbf{U}_s - \tilde{\mathbf{U}}_s\right)$ we can rewrite the objective as:

$$L_s(\mathbf{U}_s) \cong L_s(\tilde{\mathbf{U}}_s) - \left[\frac{\partial L_s(\tilde{\mathbf{U}}_s)}{\partial \tilde{\mathbf{U}}_s}\right]\left[\frac{\partial \mathbf{R}_s(\tilde{\mathbf{U}}_s)}{\partial \tilde{\mathbf{U}}_s}\right]^{-1}\mathbf{R}_s(\tilde{\mathbf{U}}_s) \tag{68}$$

Denoting the matrix product as the adjoint variable ${\mathbf{\Lambda}_s}^T$

$${\mathbf{\Lambda}_s}^T = -\left[\frac{\partial L_s(\tilde{\mathbf{U}}_s)}{\partial U_s}\right]\left[\frac{\partial \mathbf{R}_s(\tilde{\mathbf{U}}_s)}{\partial \tilde{\mathbf{U}}_s}\right]^{-1} \tag{69}$$

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

$$L_s(\mathbf{U}_s) \cong L_s(\tilde{\mathbf{U}}_s) + {\mathbf{\Lambda}_s}^T\mathbf{R}_s(\tilde{\mathbf{U}}_s) \tag{70}$$

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 $\tilde{\mathbf{U}}_s$ by interpolating a coarse grid solution onto the fine grid at each time step and evaluating $L_s(\tilde{\mathbf{U}}_s)$ and $\mathbf{R}_s(\tilde{\mathbf{U}}_s)$. The difficulty lies in obtaining the adjoint value $\mathbf{\Lambda}_s^T$. A direct calculation of this quantity costs as much as obtaining the true solution $\mathbf{U}_s$, in which case, we might as well evaluate $L_s(\mathbf{U}_s)$ directly. To alleviate this expense, we approximate the fine grid adjoint value by calculating the coarse grid adjoint denoted as $\mathbf{\Lambda}$ 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.

$$\begin{aligned} \tilde{\mathbf{\Lambda}}_s^T &= I_S^s\mathbf{\Lambda}^T \\ L_s(\mathbf{U}_s) &\cong L_s(\tilde{\mathbf{U}}_s) + \tilde{\mathbf{\Lambda}}_s^T\mathbf{R}_s(\tilde{\mathbf{U}}_s) \end{aligned} \tag{71}$$

$$\underbrace{L_s(\mathbf{U}_s) - L_s(\tilde{\mathbf{U}}_s)}_{\varepsilon_s} \cong I_S^s \boldsymbol{\Lambda}^T \mathbf{R}_s(\tilde{\mathbf{U}}_s) \tag{72}$$

From this derivation we see that the term $\tilde{\boldsymbol{\Lambda}}_s^T \mathbf{R}_s(\tilde{\mathbf{U}}_s)$ is a direct estimate 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 $\boldsymbol{\Lambda}^T$. It is most easily solved by rearranging the matrix product expression in equation (69) to be:

$$\left[\frac{\partial \mathbf{R}(\mathbf{U})}{\partial \mathbf{U}}\right]^T \boldsymbol{\Lambda} = -\left[\frac{\partial L(\mathbf{U})}{\partial \mathbf{U}}\right]^T \tag{73}$$

Notice the approximate solutions $\tilde{\mathbf{U}}$ have been replaced with converged solutions $\mathbf{U}$ since this equation is solved using the coarse grid solution. The matrices in equation (73) 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:

$$\begin{bmatrix} \left[\frac{\partial \mathbf{R}^1(\mathbf{U})}{\partial \mathbf{U}^1}\right] & & & & & \\ \left[\frac{\partial \mathbf{R}^2(\mathbf{U})}{\partial \mathbf{U}^1}\right] & \left[\frac{\partial \mathbf{R}^2(\mathbf{U})}{\partial \mathbf{U}^2}\right] & & 0 & & \\ \left[\frac{\partial \mathbf{R}^3(\mathbf{U})}{\partial \mathbf{U}^1}\right] & \left[\frac{\partial \mathbf{R}^3(\mathbf{U})}{\partial \mathbf{U}^2}\right] & \left[\frac{\partial \mathbf{R}^3(\mathbf{U})}{\partial \mathbf{U}^3}\right] & & & \\ & & \ddots & \ddots & & \\ 0 & & \left[\frac{\partial \mathbf{R}^{n-1}(\mathbf{U})}{\partial \mathbf{U}^{n-3}}\right] & \left[\frac{\partial \mathbf{R}^{n-1}(\mathbf{U})}{\partial \mathbf{U}^{n-2}}\right] & \left[\frac{\partial \mathbf{R}^{n-1}(\mathbf{U})}{\partial \mathbf{U}^{n-1}}\right] & \\ & & & \left[\frac{\partial \mathbf{R}^{n}(\mathbf{U})}{\partial \mathbf{U}^{n-2}}\right] & \left[\frac{\partial \mathbf{R}^{n}(\mathbf{U})}{\partial \mathbf{U}^{n-1}}\right] & \left[\frac{\partial \mathbf{R}^{n}(\mathbf{U})}{\partial \mathbf{U}^{n}}\right] \end{bmatrix}^T \begin{bmatrix} \Lambda^1 \\ \Lambda^2 \\ \Lambda^3 \\ \vdots \\ \Lambda^{n-1} \\ \Lambda^n \end{bmatrix} = -\begin{bmatrix} \frac{\partial L}{\partial \mathbf{U}^1} \\ \frac{\partial L}{\partial \mathbf{U}^2} \\ \frac{\partial L}{\partial \mathbf{U}^3} \\ \vdots \\ \frac{\partial L}{\partial \mathbf{U}^{n-1}} \\ \frac{\partial L}{\partial \mathbf{U}^n} \end{bmatrix}^T \tag{74}$$

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

$$\begin{bmatrix} \left[\frac{\partial \mathbf{R}^1(\mathbf{U})}{\partial \mathbf{U}^1}\right]^T & \left[\frac{\partial \mathbf{R}^2(\mathbf{U})}{\partial \mathbf{U}^1}\right]^T & \left[\frac{\partial \mathbf{R}^3(\mathbf{U})}{\partial \mathbf{U}^1}\right]^T & & & \\ & \left[\frac{\partial \mathbf{R}^2(\mathbf{U})}{\partial \mathbf{U}^2}\right]^T & \left[\frac{\partial \mathbf{R}^3(\mathbf{U})}{\partial \mathbf{U}^2}\right]^T & \left[\frac{\partial \mathbf{R}^4(\mathbf{U})}{\partial \mathbf{U}^2}\right]^T & 0 & \\ & & \ddots & \ddots & & \\ & & & \left[\frac{\partial \mathbf{R}^{n-2}(\mathbf{U})}{\partial \mathbf{U}^{n-2}}\right]^T & \left[\frac{\partial \mathbf{R}^{n-1}(\mathbf{U})}{\partial \mathbf{U}^{n-2}}\right]^T & \left[\frac{\partial \mathbf{R}^{n}(\mathbf{U})}{\partial \mathbf{U}^{n-2}}\right]^T \\ 0 & & & & \left[\frac{\partial \mathbf{R}^{n-1}(\mathbf{U})}{\partial \mathbf{U}^{n-1}}\right]^T & \left[\frac{\partial \mathbf{R}^{n}(\mathbf{U})}{\partial \mathbf{U}^{n-1}}\right]^T \\ & & & & & \left[\frac{\partial \mathbf{R}^{n}(\mathbf{U})}{\partial \mathbf{U}^{n}}\right]^T \end{bmatrix} \begin{bmatrix} \Lambda^1 \\ \Lambda^2 \\ \Lambda^3 \\ \vdots \\ \Lambda^{n-1} \\ \Lambda^n \end{bmatrix} = -\begin{bmatrix} \frac{\partial L}{\partial \mathbf{U}^1} \\ \frac{\partial L}{\partial \mathbf{U}^2} \\ \frac{\partial L}{\partial \mathbf{U}^3} \\ \vdots \\ \frac{\partial L}{\partial \mathbf{U}^{n-1}} \\ \frac{\partial L}{\partial \mathbf{U}^n} \end{bmatrix}^T \tag{75}$$

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.

3.2.2 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 3.2.1). We start by using a Taylor series expansion of the true time integrated objective calculated using a small time step $L_t(\mathbf{U}_t)$

$$L_t(\mathbf{U}_t) = L_t(\tilde{\mathbf{U}}_t) + \frac{\partial L_t(\tilde{\mathbf{U}}_t)}{\partial \tilde{\mathbf{U}}_t}\left(\mathbf{U}_t - \tilde{\mathbf{U}}_t\right) + \cdots \tag{76}$$

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

$$\mathbf{R}_t(\mathbf{U}_t) = 0 = \mathbf{R}_t(\tilde{\mathbf{U}}_t) + \frac{\partial \mathbf{R}_t(\tilde{\mathbf{U}}_t)}{\partial \tilde{\mathbf{U}}_t}\left(\mathbf{U}_t - \tilde{\mathbf{U}}_t\right) + \cdots \tag{77}$$

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

$$L_t(\mathbf{U}_t) \cong L_t(\tilde{\mathbf{U}}_t) + \mathbf{\Lambda_t}^T \mathbf{R}_t(\tilde{\mathbf{U}}_t) \tag{78}$$

The requirement for the small time step adjoint solution ($\mathbf{\Lambda_t}$) 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.

$$L_t(\mathbf{U}_t) \cong L_t(\tilde{\mathbf{U}}_t) + \tilde{\mathbf{\Lambda}}_t^T \mathbf{R}_t(\tilde{\mathbf{U}}_t) \tag{79}$$

We note that the adjoint error estimates alone correspond to the error term given as:

$$\underbrace{L_t(\mathbf{U}_t) - L_t(\tilde{\mathbf{U}}_t)}_{\varepsilon_t} \cong \tilde{\mathbf{\Lambda}}_t^T \mathbf{R}_t(\tilde{\mathbf{U}}_t) \tag{80}$$

3.2.3 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:

$$L(\mathbf{U}) = L(\tilde{\mathbf{U}}_c) + \frac{\partial L(\tilde{\mathbf{U}}_c)}{\partial \tilde{\mathbf{U}}_c}\left(\mathbf{U} - \tilde{\mathbf{U}}_c\right) + \cdots \tag{81}$$

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

$$L(\mathbf{U}) \cong L(\tilde{\mathbf{U}}_c) + \tilde{\mathbf{\Lambda}}_c^T \mathbf{R}(\tilde{\mathbf{U}}_c) \tag{82}$$

The approximate adjoint $\tilde{\mathbf{\Lambda}}_c$ 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.

$$\underbrace{L(\mathbf{U}) - L(\tilde{\mathbf{U}}_c)}_{\varepsilon_c} \cong \tilde{\mathbf{\Lambda}}_c^T \mathbf{R}(\tilde{\mathbf{U}}_c) \tag{83}$$

It is of interest to note the residual $\mathbf{R}(\tilde{\mathbf{U}}_c)$ 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.

3.2.4 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 total 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 $\tilde{\mathbf{U}}_{cst}$ computed on a coarser mesh with a larger time step with partial convergence as:

$$L_{st}(\mathbf{U}_{cst}) = L_{st}(\tilde{\mathbf{U}}_{cst}) + \frac{\partial L_{st}(\tilde{\mathbf{U}}_{cst})}{\partial \tilde{\mathbf{U}}_{cst}} \left(\mathbf{U}_{cst} - \tilde{\mathbf{U}}_{cst} \right) + ... \tag{84}$$

In the above expression, $\tilde{\mathbf{U}}_{cst}$ is obtained from the partially converged approximate solution $\tilde{\mathbf{U}}_c$ 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:

$$\mathbf{U}_{cst} - \tilde{\mathbf{U}}_{cst} = \mathbf{U}_s - \tilde{\mathbf{U}}_s + \mathbf{U}_t - \tilde{\mathbf{U}}_t + \mathbf{U} - \tilde{\mathbf{U}}_c \tag{85}$$

Here $\mathbf{U}_s - \tilde{\mathbf{U}}_s$ corresponds to the difference between the solution computed on the fine grid and the approximate coarse level solution $\tilde{\mathbf{U}}_c$ projected onto the fine grid with a fixed time step value. Similarly, $\mathbf{U}_t - \tilde{\mathbf{U}}_t$ 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 $\mathbf{U} - \tilde{\mathbf{U}}_c$ 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 (84), 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:

$$L_{st}(\mathbf{U}_{cst}) \cong L_{st}(\tilde{\mathbf{U}}_{cst}) + \tilde{\mathbf{\Lambda}}_s^T \mathbf{R}_s(\tilde{\mathbf{U}}_s) + \tilde{\mathbf{\Lambda}}_{\mathbf{t}}^{\ T} \mathbf{R}_t(\tilde{\mathbf{U}}_t) + \tilde{\mathbf{\Lambda}}_c^T \mathbf{R}(\tilde{\mathbf{U}}_c) \tag{86}$$

where the adjoint variables $\tilde{\mathbf{\Lambda}}_s$, and $\tilde{\mathbf{\Lambda}}_t$ correspond to the approximate coarse level adjoint $\tilde{\mathbf{\Lambda}}_c$ projected onto the fine mesh and fine time step domain, respectively. The various residual operators $\mathbf{R}$, $\mathbf{R}_s$ and $\mathbf{R}_t$ 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 $L_{st}(\tilde{\mathbf{U}}_{cst})$. 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.

$$\begin{aligned} \delta s &= L(\mathbf{U}_c) - L_s(\tilde{\mathbf{U}}_s) \\ \delta t &= L(\mathbf{U}_c) - L_t(\tilde{\mathbf{U}}_t) \end{aligned} \tag{87}$$

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

$$L_{st}(\tilde{\mathbf{U}}_{cst}) = L(\mathbf{U}_c) - \delta s - \delta t \tag{88}$$

Inserting this approximation into the combined linear approximation (i.e. equation (86)) we get the final form of the combined approximation.

$$L_{st}(\mathbf{U}_{cst}) \cong L_s(\tilde{\mathbf{U}}_s) + L_t(\tilde{\mathbf{U}}_t) - L(\mathbf{U}_c) + \tilde{\mathbf{\Lambda}}_s^T \mathbf{R_s}(\tilde{\mathbf{U}}_s) + \tilde{\mathbf{\Lambda}}_t^T \mathbf{R_t}(\tilde{\mathbf{U}}_t) + \tilde{\mathbf{\Lambda}}_c^T \mathbf{R}(\tilde{\mathbf{U}}_c) \tag{89}$$

Thus, the change between the functional computed at low resolution with partially converged solution values and the exact discrete functional at higher spatial and temporal resolution is given by:

$$\Delta L = L_{st}(\mathbf{U}_{cst}) - L(\mathbf{U}_c) \cong L_s(\tilde{\mathbf{U}}_s) + L_t(\tilde{\mathbf{U}}_t) - 2L(\mathbf{U}_c) + \tilde{\mathbf{\Lambda}}_s^T \mathbf{R_s}(\tilde{\mathbf{U}}_s) + \tilde{\mathbf{\Lambda}}_t^T \mathbf{R_t}(\tilde{\mathbf{U}}_t) + \tilde{\mathbf{\Lambda}}_c^T \mathbf{R}(\tilde{\mathbf{U}}_c)$$

noting that the adjoint computed error terms correspond to the error given by:

$$\underbrace{L_{st}(\mathbf{U}_{cst}) - L_s(\tilde{\mathbf{U}}_s) - L_t(\tilde{\mathbf{U}}_t) + L(\mathbf{U}_c)}_{\varepsilon_{cst}} \cong + \tilde{\mathbf{\Lambda}}_s^T \mathbf{R_s}(\tilde{\mathbf{U}}_s) + \tilde{\mathbf{\Lambda}}_t^T \mathbf{R_t}(\tilde{\mathbf{U}}_t) + \tilde{\mathbf{\Lambda}}_c^T \mathbf{R}(\tilde{\mathbf{U}}_c) \tag{90}$$

3.3 Implementation Details

3.3.1 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. [13] This is accomplished using a set of allowable subdivision types for each element as shown in Figure 12. 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.

3.3.2 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 [2]. Specifically, the compact

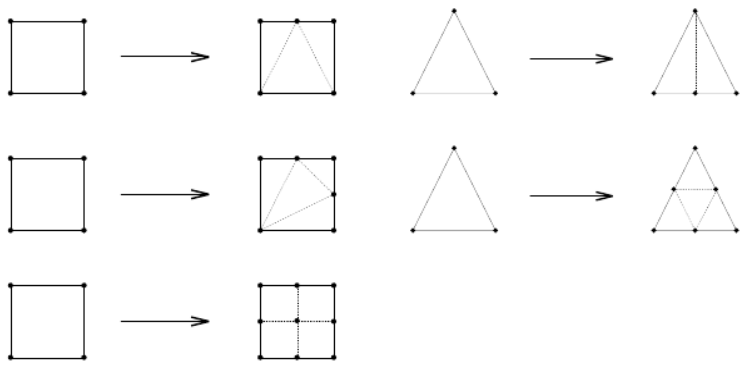


Figure 12: Allowable subdivision patterns for triangle and quadrilateral cells

functions of Wendland [23] are used with an additional polynomial constraint to ensure 2nd order accuracy for all cases. The approximations have the general form

$$s(\vec{x}) = \sum_{j=1}^{N} \alpha_j \phi\left(\vec{x} - \vec{x_j}\right) + p(\vec{x}) \tag{91}$$

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 [1] followed by a linear search over the remaining candidates.

3.3.3 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.

$$\begin{aligned} \tilde{\mathbf{\Lambda}}_t^{n-\frac{1}{2}} &= \tfrac{1}{2}\left(\mathbf{\Lambda}_T^n + \mathbf{\Lambda}_T^{n-1}\right) \\ \tilde{\mathbf{U}}_t^{n-\frac{1}{2}} &= \tfrac{1}{2}\left(\mathbf{U}_T^n + \mathbf{U}_T^{n-1}\right) \end{aligned} \tag{92}$$

3.3.4 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.

3.4 Adaptation Strategy and Threshold Selection

Taking into consideration the above discussion we have developed a process which uses the linear estimate of the error in time, space, and convergence tolerance to adapt the discretization in a manner that effectively reduces overall error. The flow chart for this process is shown in Figure 13 where the simulation starts at the top flowing down to the bottom and can be repeated until the desired level of accuracy has been achieved.

The process allows each component of the error to be reduced if it is above some defined tolerance. To choose the individual temporal, spatial, and algebraic error tolerance (Tol_T, Tol_S and Tol_C respectively) in the refinement decision steps of the flow chart in Figure 13, we equidistribute the user defined global error tolerance Tol_{Global} over the three types of error refinement possibilities as shown in equation (93). This ensures no single error type can dominate the others and all three errors will be roughly equal in magnitude when the global error tolerance has been achieved.

$$Tol_T = \frac{Tol_{Global}}{3} \qquad Tol_S = \frac{Tol_{Global}}{3} \qquad Tol_C = \frac{Tol_{Global}}{3} \tag{93}$$

The above discussion is concerned with choosing the dimension of the problem (time, space or convergence tolerance) to refine but still to be discussed is what to refine within each discretization, how much to refine and how frequently to refine. To examine these aspects, we will discuss the spatial, temporal and convergence error contributions separately.

3.4.1 Space Refinement

We have studied the issue of how often to apply spatial refinements within a time-dependent simulation in previous work [8] and have determined that more frequent refinements result in more rapid error reduction at lower cost, but that refining every 2 to 4 time steps is usually optimal. To develop the rationale for how much spatial error to refine at every time step we can look at previous work [14] for the steady state problem where the goal was to equidistribute the spatial error among all the cells. For the unsteady problem we will extend this to treat the time accurate problem as one single system of equations

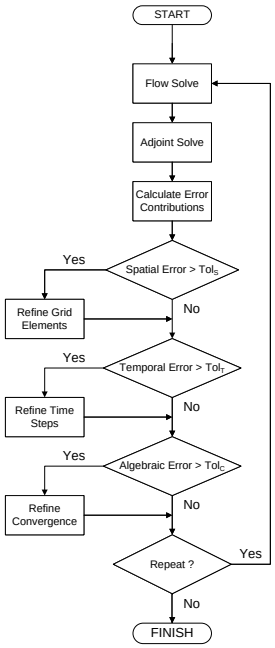


Figure 13: Global solution and refinement process

that spans both space and time. To determine which elements to refine we define a maximum allowable error level for each cell s on every time step n by equidistributing the spatial error tolerance Tol_S over all the cells of the N time step simulation.

$$s = \frac{Tol_S}{\sum_{n=1}^{N} Elements_n} \tag{94}$$

Again, extending the steady state case, a spatial refinement parameter r_s^i for each element i is defined as the ratio of the cell-wise error ε_s^i to the maximum allowed error s as given by:

$$r_s^i = \frac{\varepsilon_s^i}{s} \tag{95}$$

Using this definition we can flag an element at any time step for refinement when its refinement parameter exceeds a threshold λ_s that is pre-defined. This has the desirable effect of allowing spatial refinement every time step, if elements within the time step exceed the error threshold, but it does not force refinement to occur every time step if no elements are above the threshold.

To choose how many elements to refine at any refinement step the simplest choice would be all elements that exceed a threshold of one ($\lambda_s = 1$) where the allowable element spatial error s is determined by the global error as shown before in equation (94). Doing this at every refinement interval would insure that eventually the error would become equidistributed and result in an acceptable global spatial error. A smarter choice, and

one that has already been explored for steady state problems in reference [14], is to use a decreasing threshold value. In this scenario the threshold is initially set high, where only a small fraction of the highest error cells are refined, and then on subsequent steps the threshold is lowered until eventually a threshold of 1 is reached. This has the effect of equidistributing the error while not increasing the computational cost significantly, and then in the last couple refinement sweeps heavily refining to meet the desired error goals.

3.4.2 Time Refinement

The time refinement will follow a similar path to the spatial refinement although it will be modified by the requirement that all elements within a time step are equally stepped through time. Thus we do not include the possibility of having spatially varying time steps, although such formulations are possible and have been considered particularly in the context of space-time formulations [12, 7]. To begin, we will again seek to equidistribute the temporal error evenly among all elements of all time steps by developing an allowable error t for each element i by taking the allowable time error tolerance Tol_T and dividing it by the total number of elements in the N time step simulation.

$$t = \frac{Tol_T}{\sum_{n=1}^{N} Elements_n} \tag{96}$$

The temporal refinement parameter r_t^n is then implemented to sum all the cell-wise error contributions ε_t^i within a single time step and dividing it by the allowable error for a step with the same number of elements.

$$r_t^n = \frac{\sum_{i=1}^{Elem^n} \varepsilon_t^i}{t \times Elements_n} \tag{97}$$

By summing over all the elements within the time step we have reduced the spatially varying cell-wise temporal error down to a single value for each time step to indicate which time steps contain the most temporal error. To control the number of time steps that are refined at the completion of any one complete simulation sweep we will use a threshold for the time λ_t which can be varied in a similar descending manner as was discussed for the spatial refinement (Sec.3.4.1).

3.4.3 Convergence Refinement

Each element i has an allowable error c calculated using the pre-determined convergence tolerance Tol_C based on how many total elements within the time accurate simulation as shown.

$$c = \frac{Tol_C}{\sum_{n=1}^{N} Elements_n} \tag{98}$$

The refinement parameter r_c^n will again be formed by summing all the cell-wise error contributions ε_c^i within a single time step and dividing it by the allowable error for a step with the same number of elements.

$$r_c^n = \frac{\sum_{i=1}^{Elem^n} \varepsilon_c^i}{c \times Elements_n} \tag{99}$$

Another threshold variable λ_c will be used to control the number of time steps at which the convergence tolerance is reduced at the completion of any one sweep of the simulation.

3.5 Adaptive Error Control of Combined Error

In order to demonstrate the adaptive control of these combined error sources we use the model test problem described in section 3.2. and Figure 10 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 $Tol_{Global} \leq 10^{-6}$. The simulation is allowed to progress through the refinement algorithm presented in Figure 13 for 7 cycles of the loop with the threshold value decreasing in each successive loop from 32, 16, 8, 4, 2 and then fixed at 1 until the error tolerance is achieved. The plots of the adapted convergence tolerance, time step size and number of mesh elements at each time step are displayed in Figure 14 for each cycle of the algorithm. The plot of the convergence tolerance shows the requested reduction of 10^{-1} is sufficient for the first 3 refinement cycles but after using the same convergence tolerance on the 4th cycle the decreasing error threshold of 4 is exceeded. Once the tolerance has been exceeded, the algorithm branches and identifies the time steps at the start of the simulation as those exceeding the error threshold for the number of elements at each time step. These time steps are flagged for an increased convergence criteria (i.e. lower final residual) and leaves the other steps as they were before. A similar pattern is followed on the 5th and 6th cycles of the algorithm where again the initial time steps are found to need a tighter convergence tolerance to meet the error threshold.

The time step plot in Figure 14 shows a different pattern, where the first flow simulation using only 16 time steps to model the full time accurate simulation was found to exceed the temporal error threshold. Once inside the refinement branch the algorithm determined every time step was exceeding the allowable threshold. For the next 2 cycles of the algorithm the temporal error resulting from the now 32 equally spaced time steps is below the decreasing threshold values but on the 4th, 5th and 6th cycles again the temporal error threshold is exceeded and the time steps at the start of the simulation are flagged for refinement.

The spatial refinement plots in Figure 14 show yet a different pattern where the initial 2 cycles through the algorithm exceed the spatial error threshold. Within these cycles, when the spatial refinement branch has been taken, all elements for all time steps are evaluated against the element error threshold and in each case at least some elements in each time step are flagged for refinement. On the subsequent 2 cycles of the algorithm the spatial error is found to be below the error threshold, and so no spatial refinement occurs until the 5th and 6th cycles where again at least some elements on each time step are

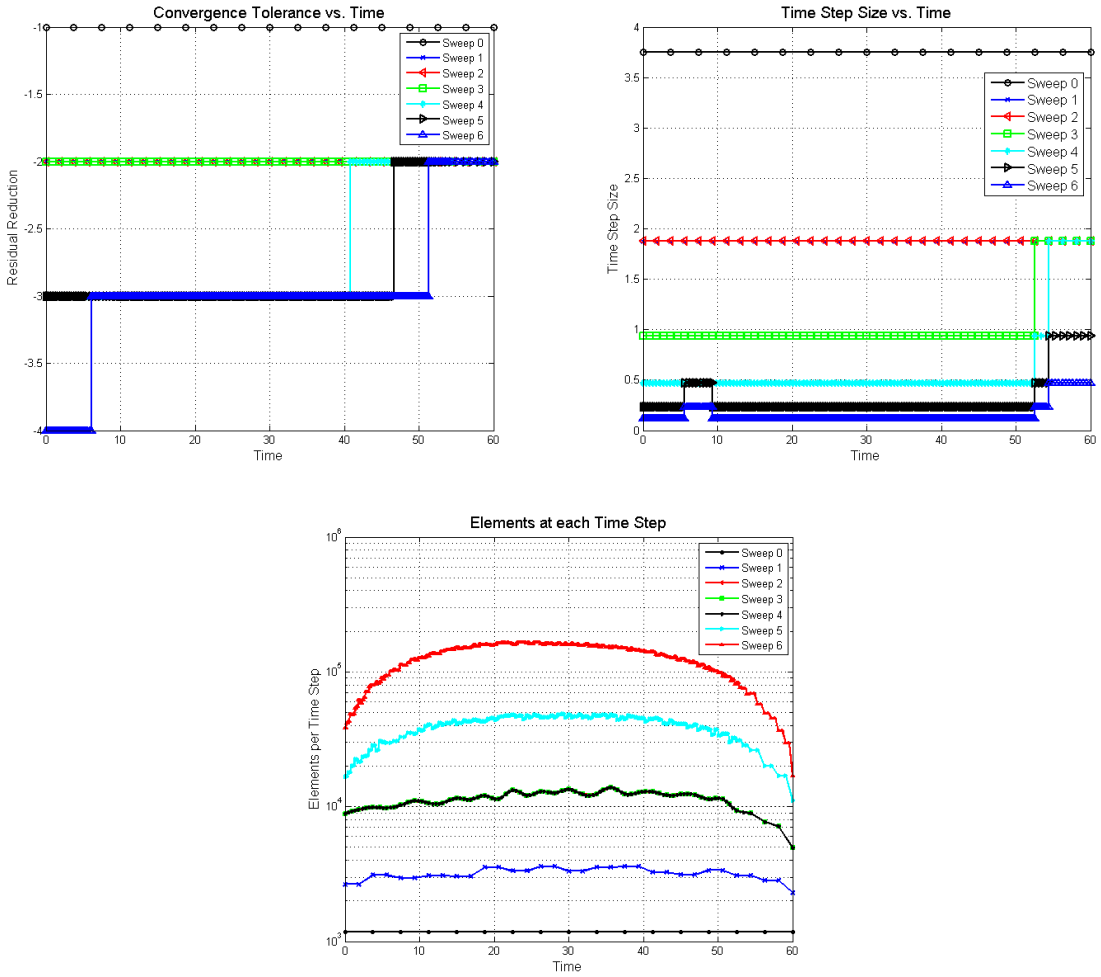


Figure 14: (Top Left) Convergence tolerance for each sweep of the time accurate simulation; (Top Right) Time step size for each sweep of the time accurate simulation; (Bottom) Number of elements for each sweep of the time accurate simulation.

refined. Images of the refined region within the computational mesh at non-dimensional times of t = 0, 30 and 60 are shown in Figure 15 for each sweep of the algorithm. The mesh refinement patterns clearly show refinement from the location of the vortex to the region over which the time-integrated is integrated. This later feature is important and seen as a major advantage over simpler feature tracking algorithms such as gradient based refinement.

Calculating the flow and adjoint solution at each cycle of the algorithm is expensive and it is unknown whether going through multiple solution cycles to arrive at a final answer is cheaper than refining the discretization and running it once. To evaluate this we have defined our own measure of computational expense called the "Work Unit" which is the cost to converge the residual of one element one order of magnitude on one time step. For example, a 16 time step simulation on a fixed grid of 1176 elements with the residual on each step being converged 3 orders in magnitude would have a cost of $16 * 1176 * 3 = 56448$ work units. Using this definition we have plotted the cost of each adapted flow solution in Figure 16 along with the relative error of the objective when compared to the value of the true analytic objective. As a comparison, a second curve is plotted resulting from uniformly refining the temporal and spatial discretization 4 times, where the coarsest simulation was converged 1 order of magnitude, and each subsequent refined discretization had the convergence tightened by an additional order of magnitude (i.e. 5th solution is converged 5 orders of magnitude).

The cost of a single solution alone doesn't tell the true expense of arriving at the final adapted solution using adjoint-based refinement, therefore we have plotted the relative error as a function of cumulative work units in the second plot in Figure 16 along with the same uniform curve as before. The cumulative cost is the cost to solve the flow and adjoint on all previous adaptation cycles plus the cost of the current solution. From this measure of the total cost to arrive at the final adapted solution we can see that the final solution is the major expense and all previous adaptation cycles do not add much to the overall cost to reach the final result. What the previous adaptation cycles do though, is more accurately target the regions for refinement so that each cycle produces nearly the same relative error as uniform refinement but with far less expense. In fact, the efficiency gains for a given level of accuracy are dramatic, showing over 1.5 orders of magnitude reduction in work units for a relative error of 1.e-5 in Figure 16. More importantly, these results involve total simulation error and are thus more realistic than those shown in section 2.8, where the measured error was based on a fine time discretization reference solution and did not include any sptial discretization error.

3.6 Optimal cost effective error reduction

Although the above results demonstrate large efficiency gains for achieving prescribed accuracy levels compared to a global refinement strategy, the approach is based on equidistribution of all error types and does not take into account the computational cost of reducing each type of error. A more optimal approach 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

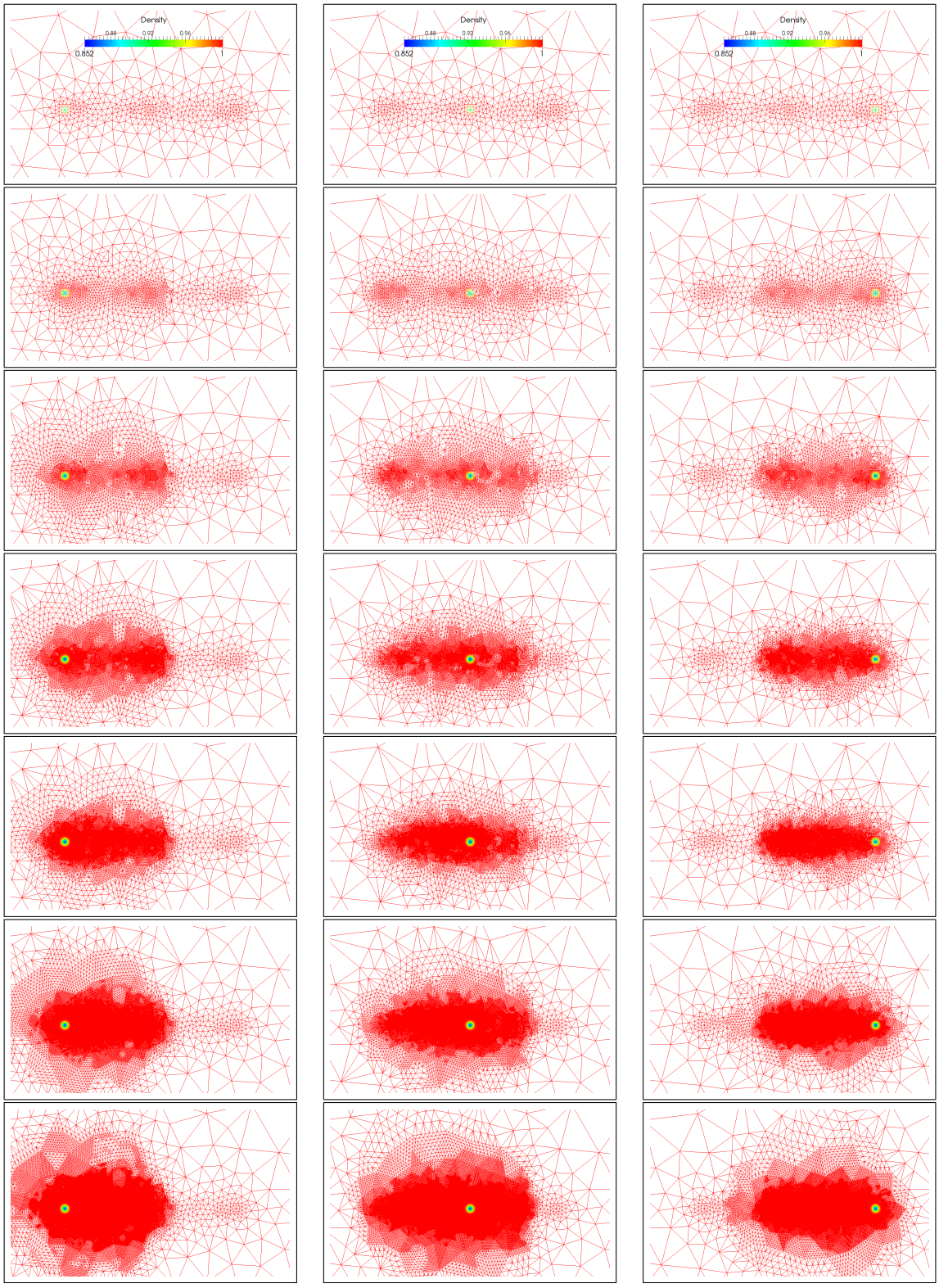


Figure 15: Convecting vortex and adapted grid at time = 0, 30, and 60 for 7 different refinement sweeps.

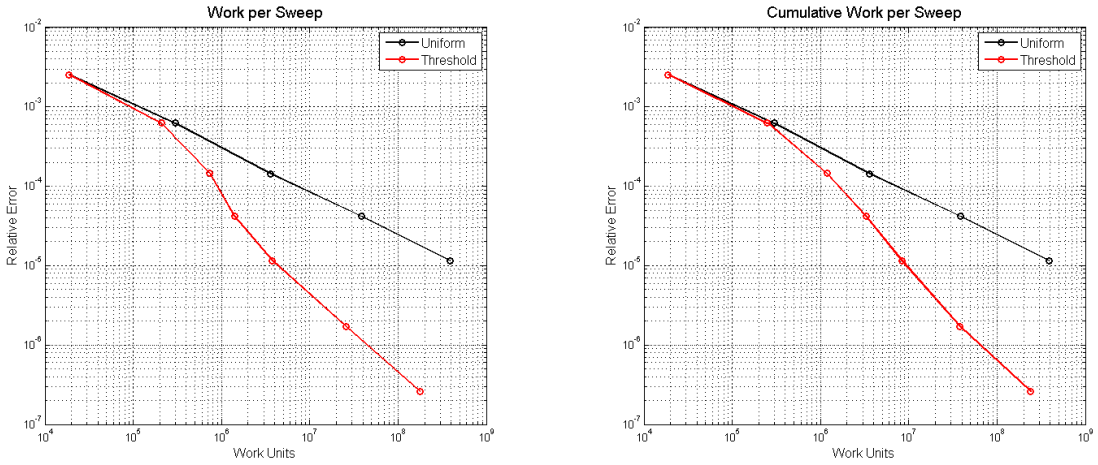


Figure 16: Work units and cumulative work units for each adaptive sweep of the time accurate simulation.

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.

$$\lim_{n\to\infty} \frac{|\xi_{n+1}|}{|\xi_n|^q} = \mu \tag{100}$$

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.

$$\xi_{n+k} = \xi_n^{q^{(k)}} \mu^{\sum_{i=1}^{k} q^{(k-i)}} \tag{101}$$

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.

$$k = \frac{\log(\xi_{n+k})}{q \log(\xi_n)} \tag{102}$$

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.

3.6.1 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 (102), 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 (103).

$$CoR_c = (\#\text{ of Elements within Time Step}) \times \Delta k \tag{103}$$

3.6.2 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 equation (102) must be calculated using the starting error ($L_2(R) \approx 0.1$) and the final convergence tolerance requested. This is represented in equation (104) where one can see that refining a time step is a costly operation.

$$CoR_t = (\#\text{ of Elements within Time Step}) \times k \tag{104}$$

3.6.3 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 3.3.1. In the allowable subdivisions of Figure 12 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

$$CoR_s = (\#\text{ of New Elements} = 3) \times k \tag{105}$$

3.6.4 Selection of Discretization to Refine

The final piece in deciding what and where to refine is consists of combining the resulting discretization error with the cost to refine error source. 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 preceding 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 computational 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 (90), from the user requested error tolerance Tol_R as shown in equation (106).

$$\varepsilon_{ex} = \varepsilon_{cst} - Tol_R \tag{106}$$

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 error type in the following results. 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. As mentioned previously, a more effective strategy is to gradually increase the amount of error refined on any one refinement sweep [14]. 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 several refinement sweeps refining aggressively to meet the desired error goals. This modified error excess equation is shown with the error fraction λ parameter included in equation (107).

$$\varepsilon_{ex} = \frac{\varepsilon_{cst} - Tol_R}{\lambda} \tag{107}$$

3.7 Demonstration of Optimal Cost Error Control

In order to demonstrate the optimal cost error control approach, we consider the problem of a vortex 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 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

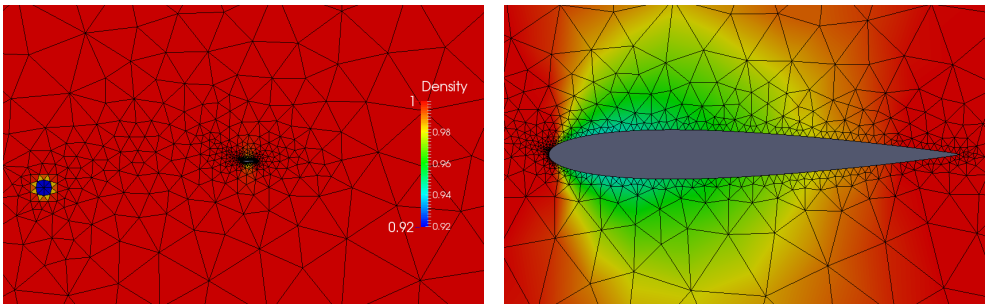


Figure 17: Initial density distribution around airfoil for grid of 2401 elements (left), Close up view near airfoil (right)

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 Figure 17.

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

The plots of error versus cost convergence tolerance 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 containing the considered element.

The time step plot Figure 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 Figure 19 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

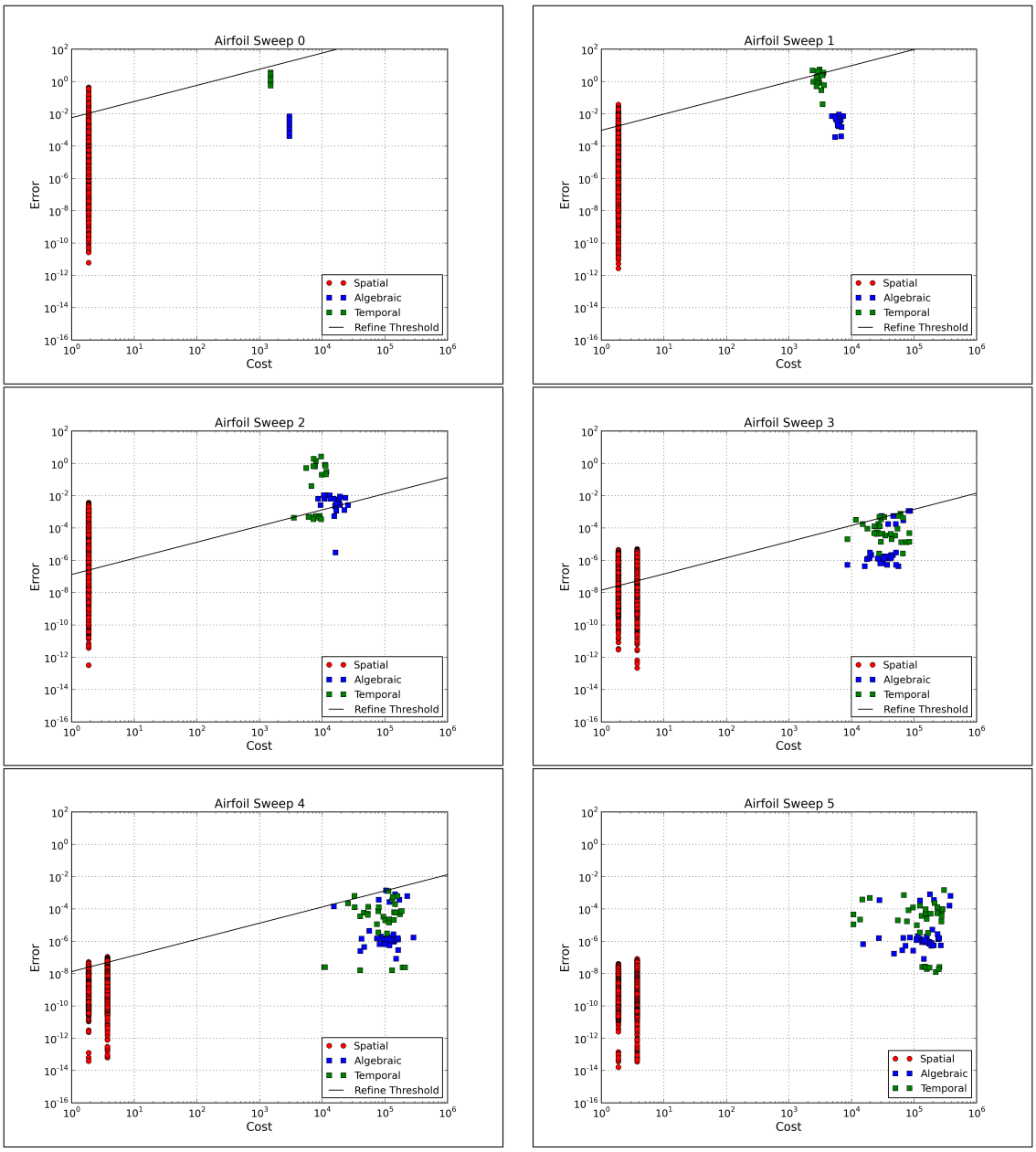


Figure 18: Error vs. cost plots for the airfoil with convecting vortex over 5 different refinement sweeps.

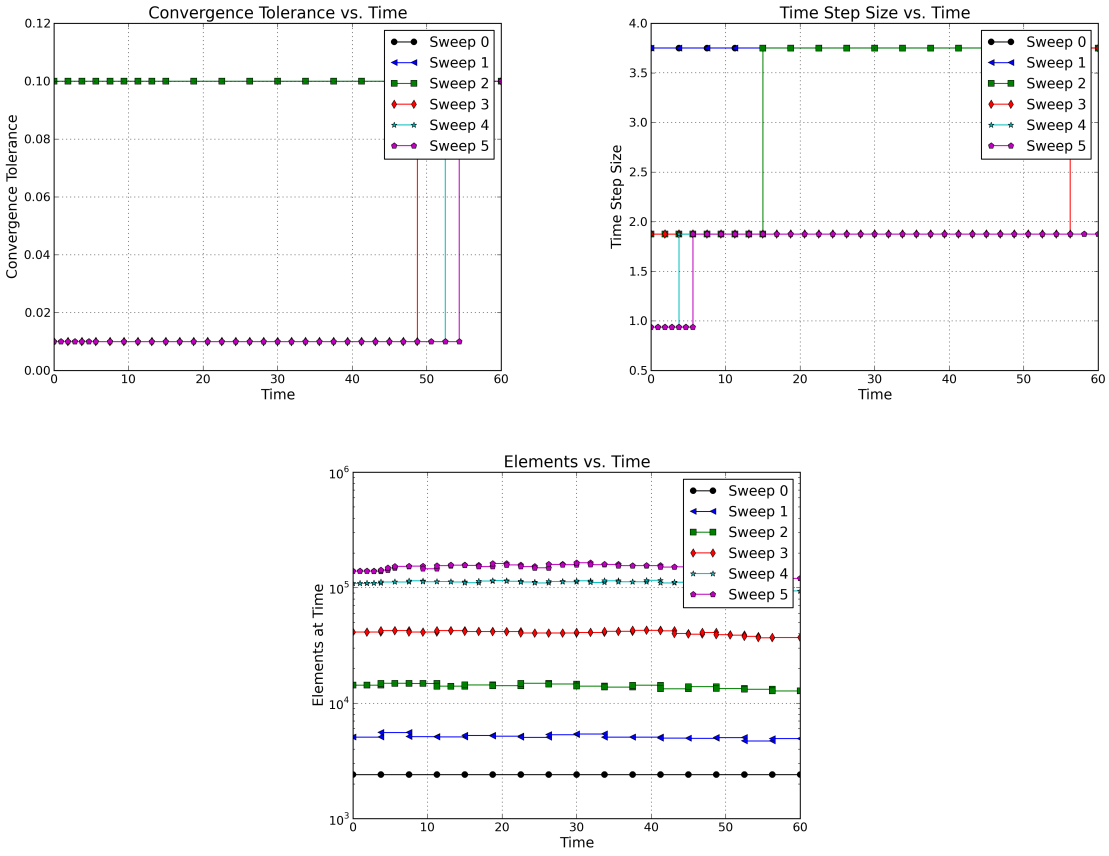


Figure 19: (Top Left) Convergence Tolerance as a function of time for each refinement sweep of the airfoil; (Top Right) Time step size as a function of time for each refinement sweep of the airfoil; (Bottom) Elements as a function of time for each refinement sweep of the airfoil

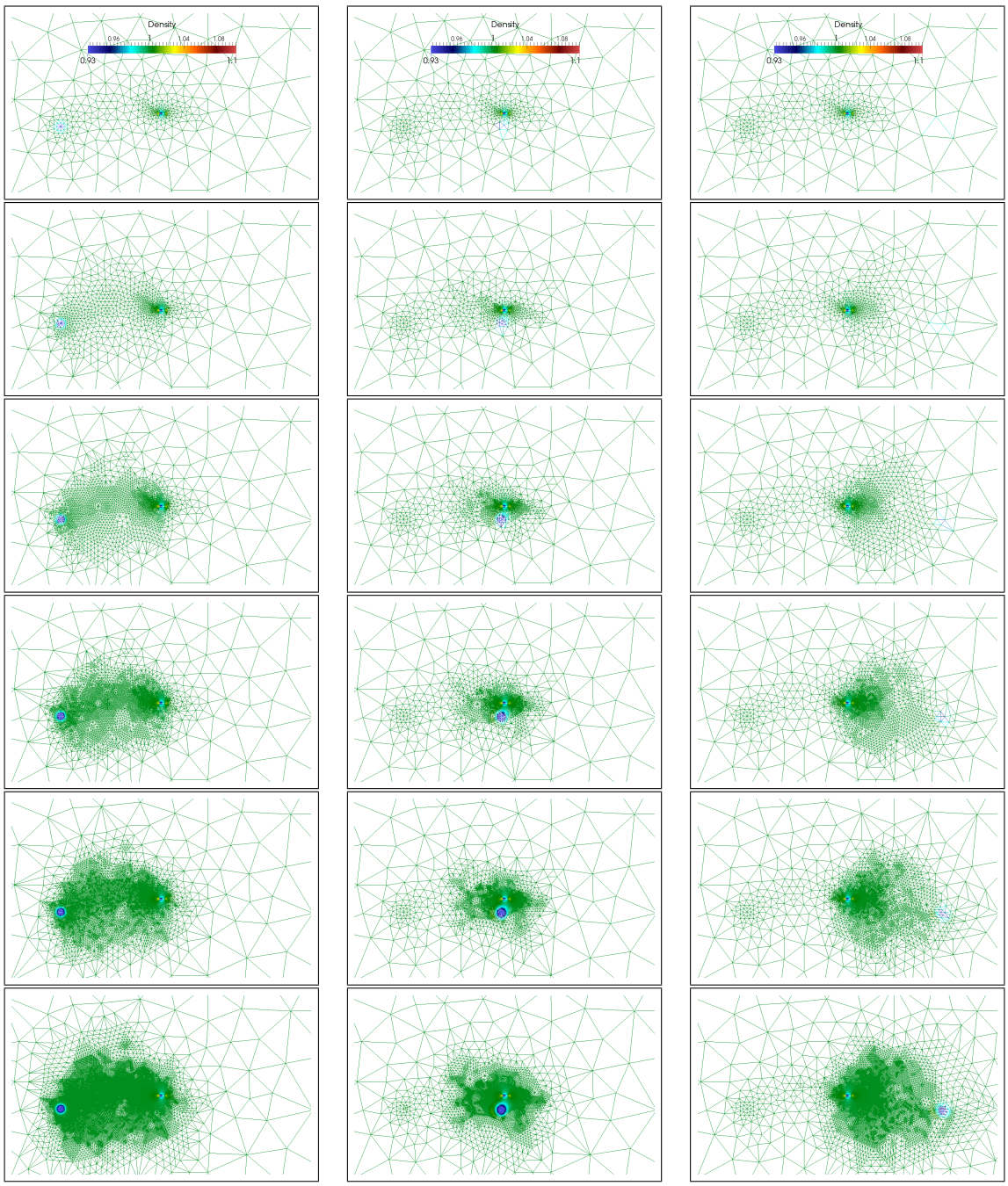


Figure 20: Airfoil with vortex solution and adapted grid at time = 0, 30, and 60 for 5 different refinement sweeps

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 Figure 18 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 (107)) to be more or less aggressive (e.g. $\lambda = 6$, 4, 2, 1 or $\lambda = 16$, 8, 4, 2, 1). Figure 20 illustrates the refinement patterns at various time locations for different sweeps of the adaptive algorithm.

4 Extension to Coupled Multidisciplinary Problems

The adjoint error estimation approach described and demonstrated in the previous sections can be extended to time-dependent coupled multidisciplinary problems in a relatively straight-forward manner. In fact, the initial formulation developed in section 2 can be considered to be an example of a simple multidisciplinary problem which required the coupled solution of the flow field and mesh motion equations. In this section, we present a formulation for general multidisciplianry problems with arbitrary number of coupled disciplines. As previously, the formulation is useful for isolating and estimating spatial, temporal and algebraic error components of the overall simulation. In addition, estimates of the individual disciplinary components for each of these error types can be obtained, as well as coupling error between specific disciplines. The formulation can even be used to obtain disciplinary error due to the use of lower fidelity models.

In order to simplify the presentation, we adopt a higher level notation for describing the governing equations, constraints and objective functionals for multidisciplinary problems. From the outset, we do not specify any particular functional forms and assume all disciplines are fully coupled in the most general formulation. For each discipline, we denote the equations to be solved as a single residual equation that spans all space and time and take inner products over space and time.

Consider an arbitrary number of m coupled unsteady disciplines, with the exact (discrete) solution represented by the set of state variables $\mathscr{U}$ spanning the spatial and temporal domains. A functional of interest computed using the coupled solution set can be represented as:

$$L = L(\mathscr{U}) = L(\mathscr{U}_1, \mathscr{U}_2, \cdots, \mathscr{U}_m) \tag{108}$$

where the state variables are obtained as the solution to the following coupled multidisciplinary problem

$$\begin{aligned} \mathscr{R}_1(\mathscr{U}_1, \mathscr{U}_2, \cdots, \mathscr{U}_m) &= 0 \\ \mathscr{R}_2(\mathscr{U}_1, \mathscr{U}_2, \cdots, \mathscr{U}_m) &= 0 \\ &\vdots \\ \mathscr{R}_m(\mathscr{U}_1, \mathscr{U}_2, \cdots, \mathscr{U}_m) &= 0 \end{aligned} \tag{109}$$

Here $\mathscr{R}_n = 0$ represents the residual of the discrete equations for the n^{th} discipline. Assuming we only have an approximate solution $\tilde{\mathscr{U}} = (\tilde{\mathscr{U}}_1, \tilde{\mathscr{U}}_2, \cdots, \tilde{\mathscr{U}}_m)$, we can express

the functional as a Taylor expansion about the approximate functional value as:

$$\begin{aligned} L(\boldsymbol{\mathcal{U}}) = L(\tilde{\boldsymbol{\mathcal{U}}}) &+ \left[\frac{\partial L}{\partial \boldsymbol{\mathcal{U}}_1}\right]_{\tilde{\boldsymbol{\mathcal{U}}}} (\boldsymbol{\mathcal{U}}_1 - \tilde{\boldsymbol{\mathcal{U}}}_1) \\ &+ \left[\frac{\partial L}{\partial \boldsymbol{\mathcal{U}}_2}\right]_{\tilde{\boldsymbol{\mathcal{U}}}} (\boldsymbol{\mathcal{U}}_2 - \tilde{\boldsymbol{\mathcal{U}}}_2) \\ &\vdots \\ &+ \left[\frac{\partial L}{\partial \boldsymbol{\mathcal{U}}_m}\right]_{\tilde{\boldsymbol{\mathcal{U}}}} (\boldsymbol{\mathcal{U}}_m - \tilde{\boldsymbol{\mathcal{U}}}_m) \\ &+ \mathcal{O}\left(\boldsymbol{\mathcal{U}} - \tilde{\boldsymbol{\mathcal{U}}}\right)^2 + \mathcal{O}\left(\boldsymbol{\mathcal{U}} - \tilde{\boldsymbol{\mathcal{U}}}\right)^3 \cdots \end{aligned} \tag{110}$$

where $\tilde{\boldsymbol{\mathcal{U}}}$ refers to the approximate state variables for all disciplines. Ignoring higher-order terms in the Taylor expansion, the linear approximation of the error in the functional due to the presence of the approximate solution can be written as the inner product between the vector of functional sensitivities to states and the error vector of the states as:

$$L(\boldsymbol{\mathcal{U}}) - L(\tilde{\boldsymbol{\mathcal{U}}}) = \left[\begin{array}{cccc} \frac{\partial L}{\partial \boldsymbol{\mathcal{U}}_1} & \frac{\partial L}{\partial \boldsymbol{\mathcal{U}}_2} & \cdots & \frac{\partial L}{\partial \boldsymbol{\mathcal{U}}_m} \end{array}\right]_{\tilde{\boldsymbol{\mathcal{U}}}} \left[\begin{array}{c} (\boldsymbol{\mathcal{U}}_{1_h} - \tilde{\boldsymbol{\mathcal{U}}}_1) \\ (\boldsymbol{\mathcal{U}}_{2_h} - \tilde{\boldsymbol{\mathcal{U}}}_2) \\ \vdots \\ (\boldsymbol{\mathcal{U}}_{m_h} - \tilde{\boldsymbol{\mathcal{U}}}_m) \end{array}\right] \tag{111}$$

Taking a first order Taylor series expansion of the coupled residual equations (i.e. equations (109)) about the approximate state, we obtain:

$$\left[\begin{array}{c} \boldsymbol{\mathcal{R}}_1(\boldsymbol{\mathcal{U}}) \\ \boldsymbol{\mathcal{R}}_2(\boldsymbol{\mathcal{U}}) \\ \vdots \\ \boldsymbol{\mathcal{R}}_m(\boldsymbol{\mathcal{U}}) \end{array}\right] \approx \left[\begin{array}{c} \boldsymbol{\mathcal{R}}_1(\tilde{\boldsymbol{\mathcal{U}}}) \\ \boldsymbol{\mathcal{R}}_2(\tilde{\boldsymbol{\mathcal{U}}}) \\ \vdots \\ \boldsymbol{\mathcal{R}}_m(\tilde{\boldsymbol{\mathcal{U}}}) \end{array}\right] + \left[\begin{array}{cccc} \frac{\partial \boldsymbol{\mathcal{R}}_1}{\partial \boldsymbol{\mathcal{U}}_1} & \frac{\partial \boldsymbol{\mathcal{R}}_1}{\partial \boldsymbol{\mathcal{U}}_2} & \cdots & \frac{\partial \boldsymbol{\mathcal{R}}_1}{\partial \boldsymbol{\mathcal{U}}_m} \\ \frac{\partial \boldsymbol{\mathcal{R}}_2}{\partial \boldsymbol{\mathcal{U}}_1} & \frac{\partial \boldsymbol{\mathcal{R}}_2}{\partial \boldsymbol{\mathcal{U}}_2} & \cdots & \frac{\partial \boldsymbol{\mathcal{R}}_2}{\partial \boldsymbol{\mathcal{U}}_m} \\ \vdots & & & \\ \frac{\partial \boldsymbol{\mathcal{R}}_m}{\partial \boldsymbol{\mathcal{U}}_1} & \frac{\partial \boldsymbol{\mathcal{R}}_m}{\partial \boldsymbol{\mathcal{U}}_2} & \cdots & \frac{\partial \boldsymbol{\mathcal{R}}_m}{\partial \boldsymbol{\mathcal{U}}_m} \end{array}\right]_{\tilde{\boldsymbol{\mathcal{U}}}} \left[\begin{array}{c} (\boldsymbol{\mathcal{U}}_1 - \tilde{\boldsymbol{\mathcal{U}}}_1) \\ (\boldsymbol{\mathcal{U}}_2 - \tilde{\boldsymbol{\mathcal{U}}}_2) \\ \vdots \\ (\boldsymbol{\mathcal{U}}_m - \tilde{\boldsymbol{\mathcal{U}}}_m) \end{array}\right] = 0 \tag{112}$$

which yields an approximation to the vector of the error in the state variables:

$$\begin{bmatrix} (\boldsymbol{\mathcal{U}}_1 - \tilde{\boldsymbol{\mathcal{U}}}_1) \\ (\boldsymbol{\mathcal{U}}_2 - \tilde{\boldsymbol{\mathcal{U}}}_2) \\ \vdots \\ (\boldsymbol{\mathcal{U}}_m - \tilde{\boldsymbol{\mathcal{U}}}_m) \end{bmatrix} = - \begin{bmatrix} \frac{\partial \boldsymbol{\mathcal{R}}_1}{\partial \boldsymbol{\mathcal{U}}_1} & \frac{\partial \boldsymbol{\mathcal{R}}_1}{\partial \boldsymbol{\mathcal{U}}_2} & \cdots & \frac{\partial \boldsymbol{\mathcal{R}}_1}{\partial \boldsymbol{\mathcal{U}}_m} \\ \frac{\partial \boldsymbol{\mathcal{R}}_2}{\partial \boldsymbol{\mathcal{U}}_1} & \frac{\partial \boldsymbol{\mathcal{R}}_2}{\partial \boldsymbol{\mathcal{U}}_2} & \cdots & \frac{\partial \boldsymbol{\mathcal{R}}_2}{\partial \boldsymbol{\mathcal{U}}_m} \\ \vdots & & & \\ \frac{\partial \boldsymbol{\mathcal{R}}_m}{\partial \boldsymbol{\mathcal{U}}_1} & \frac{\partial \boldsymbol{\mathcal{R}}_m}{\partial \boldsymbol{\mathcal{U}}_2} & \cdots & \frac{\partial \boldsymbol{\mathcal{R}}_m}{\partial \boldsymbol{\mathcal{U}}_m} \end{bmatrix}^{-1}_{\tilde{\boldsymbol{\mathcal{U}}}} \begin{bmatrix} \boldsymbol{\mathcal{R}}_1(\tilde{\boldsymbol{\mathcal{U}}}) \\ \boldsymbol{\mathcal{R}}_2(\tilde{\boldsymbol{\mathcal{U}}}) \\ \vdots \\ \boldsymbol{\mathcal{R}}_m(\tilde{\boldsymbol{\mathcal{U}}}) \end{bmatrix} \tag{113}$$

The expression may now be substituted into the functional error equation (111) yielding

$$L(\boldsymbol{\mathcal{U}}) - L(\tilde{\boldsymbol{\mathcal{U}}}) = \underbrace{- \begin{bmatrix} \frac{\partial L}{\partial \boldsymbol{\mathcal{U}}_1} & \frac{\partial L}{\partial \boldsymbol{\mathcal{U}}_2} & \cdots & \frac{\partial L}{\partial \boldsymbol{\mathcal{U}}_m} \end{bmatrix}_{\tilde{\boldsymbol{\mathcal{U}}}} \begin{bmatrix} \frac{\partial \boldsymbol{\mathcal{R}}_1}{\partial \boldsymbol{\mathcal{U}}_1} & \frac{\partial \boldsymbol{\mathcal{R}}_1}{\partial \boldsymbol{\mathcal{U}}_2} & \cdots & \frac{\partial \boldsymbol{\mathcal{R}}_1}{\partial \boldsymbol{\mathcal{U}}_m} \\ \frac{\partial \boldsymbol{\mathcal{R}}_2}{\partial \boldsymbol{\mathcal{U}}_1} & \frac{\partial \boldsymbol{\mathcal{R}}_2}{\partial \boldsymbol{\mathcal{U}}_2} & \cdots & \frac{\partial \boldsymbol{\mathcal{R}}_2}{\partial \boldsymbol{\mathcal{U}}_m} \\ \vdots & & & \\ \frac{\partial \boldsymbol{\mathcal{R}}_m}{\partial \boldsymbol{\mathcal{U}}_1} & \frac{\partial \boldsymbol{\mathcal{R}}_m}{\partial \boldsymbol{\mathcal{U}}_2} & \cdots & \frac{\partial \boldsymbol{\mathcal{R}}_m}{\partial \boldsymbol{\mathcal{U}}_m} \end{bmatrix}^{-1}_{\tilde{\boldsymbol{\mathcal{U}}}}}_{\Lambda^T_{\boldsymbol{\mathcal{U}}}} \begin{bmatrix} \boldsymbol{\mathcal{R}}_1(\tilde{\boldsymbol{\mathcal{U}}}) \\ \boldsymbol{\mathcal{R}}_2(\tilde{\boldsymbol{\mathcal{U}}}) \\ \vdots \\ \boldsymbol{\mathcal{R}}_m(\tilde{\boldsymbol{\mathcal{U}}}) \end{bmatrix} \tag{114}$$

where a vector of disciplinary adjoint variables $\Lambda_{\boldsymbol{\mathcal{U}}}$ is introduced in order to recover a system of coupled adjoint equations as

$$\begin{bmatrix} \frac{\partial \boldsymbol{\mathcal{R}}_1}{\partial \boldsymbol{\mathcal{U}}_1}^T & \frac{\partial \boldsymbol{\mathcal{R}}_2}{\partial \boldsymbol{\mathcal{U}}_1}^T & \cdots & \frac{\partial \boldsymbol{\mathcal{R}}_m}{\partial \boldsymbol{\mathcal{U}}_1}^T \\ \frac{\partial \boldsymbol{\mathcal{R}}_1}{\partial \boldsymbol{\mathcal{U}}_2}^T & \frac{\partial \boldsymbol{\mathcal{R}}_2}{\partial \boldsymbol{\mathcal{U}}_2}^T & \cdots & \frac{\partial \boldsymbol{\mathcal{R}}_m}{\partial \boldsymbol{\mathcal{U}}_2}^T \\ \vdots & & & \\ \frac{\partial \boldsymbol{\mathcal{R}}_1}{\partial \boldsymbol{\mathcal{U}}_m}^T & \frac{\partial \boldsymbol{\mathcal{R}}_2}{\partial \boldsymbol{\mathcal{U}}_m}^T & \cdots & \frac{\partial \boldsymbol{\mathcal{R}}_m}{\partial \boldsymbol{\mathcal{U}}_m}^T \end{bmatrix}_{\tilde{\boldsymbol{\mathcal{U}}}} \begin{bmatrix} \Lambda_{\boldsymbol{\mathcal{U}}_1} \\ \Lambda_{\boldsymbol{\mathcal{U}}_2} \\ \vdots \\ \Lambda_{\boldsymbol{\mathcal{U}}_m} \end{bmatrix} = - \begin{bmatrix} \frac{\partial L}{\partial \boldsymbol{\mathcal{U}}_1}^T \\ \frac{\partial L}{\partial \boldsymbol{\mathcal{U}}_2}^T \\ \vdots \\ \frac{\partial L}{\partial \boldsymbol{\mathcal{U}}_m}^T \end{bmatrix}_{\tilde{\boldsymbol{\mathcal{U}}}} \tag{115}$$

The vector of disciplinary adjoint variables can then be substituted back into the func-

tional error equation to obtain:

$$\varepsilon_{total} = L(\boldsymbol{\mathscr{U}}) - L(\tilde{\boldsymbol{\mathscr{U}}}) = + \begin{bmatrix} \Lambda^T_{\boldsymbol{\mathscr{U}}_1} & \Lambda^T_{\boldsymbol{\mathscr{U}}_2} & \cdots & \Lambda^T_{\boldsymbol{\mathscr{U}}_m} \end{bmatrix} \begin{bmatrix} \boldsymbol{\mathscr{R}}_1(\tilde{\boldsymbol{\mathscr{U}}}) \\ \boldsymbol{\mathscr{R}}_2(\tilde{\boldsymbol{\mathscr{U}}}) \\ \vdots \\ \boldsymbol{\mathscr{R}}_m(\tilde{\boldsymbol{\mathscr{U}}}) \end{bmatrix} \tag{116}$$

This expression can be interpreted as the sum of the inner products of the disciplinary adjoint variables and their corresponding residuals as:

$$\begin{aligned} \varepsilon_{total} = L(\boldsymbol{\mathscr{U}}) - L(\tilde{\boldsymbol{\mathscr{U}}}) = \\ + \left\{ \Lambda^T_{\boldsymbol{\mathscr{U}}_1} \boldsymbol{\mathscr{R}}_1(\tilde{\boldsymbol{\mathscr{U}}}) + \Lambda^T_{\boldsymbol{\mathscr{U}}_2} \boldsymbol{\mathscr{R}}_2(\tilde{\boldsymbol{\mathscr{U}}}) + \cdots + \Lambda^T_{\boldsymbol{\mathscr{U}}_m} \boldsymbol{\mathscr{R}}_m(\tilde{\boldsymbol{\mathscr{U}}}) \right\} \end{aligned} \tag{117}$$

The residuals are nonzero since they are evaluated using the approximate state variables $\tilde{\boldsymbol{\mathscr{U}}}$. As previously, a lower cost adjoint solution may be used in the place of the adjoint vector obtained as the exact solution of equation (115). Provided the approximate adjoint solution is consistent and converges asymptotically towards the exact solution in the limit of increasing resolution, equation (117) computed using the approximate adjoint solution provides an accurate estimate of the total error in the functional, since the additional error term will be asymptotically smaller than the computed error, as discussed in the introduction.

Since we have not specified the nature of the approximate solution, this error estimate can be applied for various error types. For example, if the approximate solution is obtained by solving the coupled multidisciplinary problem using coarser spatial or temporal discretizations, then equation (117) gives an estimate of the spatial or temporal discretization error in the overall multidisciplinary objective. Furthermore, since this error is obtained as the sum of individual disciplinary components, an estimate of the error contribution from each discipline is obtained. Similarly, in the case of partially converged solutions, we obtain the contributions to the overall objective error due to the convergence levels of each individual discipline. Since equations (109) are coupled, in the event each discipline is converged with frozen or lagged values from the other disciplines, the individual residual expressions in equation (117) will still be non-zero due to the fact that these must be evaluated with the latest values of the solution from all disciplines, and the error estimate will correspond to coupling error. Finally, if a low fidelity model is used for a particular discipline, and the solution from this model can be projected onto the same discrete space used by the higher fidelity space represented in equations (109), the error due to the use of the lower fidelity model on the final coupled objective would be included in the estimate provided by equation (117).

5 Conclusion and Further Work

In these notes we have shown how adjoint methods may be used to estimate error in specific simulation output functionals or quantities of interest for time-dependent and

multidisciplianry problems. The presentation was based on a discrete adjoint implementation and a Taylor series approach, rather than a variational approach, although the correspondence between these methodologies was discussed in the introduction. The adjoint error estimation implementation was verified on a canonical time-dependent problem and used to estimate and adaptively control temporal, algebraic and spatial error sources both individually and simultaneously. A general formulation for coupled multidisciplinary problems has also been also described.

Various extensions to the error estimation techniques developed in these notes can be considered. For example, although we have only considered the control of temporal error through global time step refinement, the adjoint error formulation provides local estimates of temporal error within each mesh cell of the spatial discretization for all time locations. Thus, different regions of the mesh may be advanced using different size time steps for additional efficiency gains. Such an approach fits well into a space-time formulation, and there is significant previous and on-going work in this area [12, 7]. Similarly, based on the adjoint algebraic error estimates, spatially local convergence tolerances could be prescribed in order to reduce overall computational cost for a fixed error level. However, such an implementation may be difficult to load balance in a parallel computing environment.

Although additional benefits can be obtained through more advanced formulations of these types, the presentation in these notes is focused on developing techniques that can be applied directly to existing production level simulation codes, without a need for reformulating the entire discretization or approach. On the other hand, even in this context, extending the combined spatial, temporal and algebraic error estimation and adaptive control to three dimensional time-dependent multidisciplinary problems remains a formidable task, involving not only the efficient computation of coupled multidisciplinary adjoints, but also the implementation of adaptive mesh refinement and time-step selection which are known to be challenging within a parallel computing framework. Nevertheless, adjoint error estimation is a powerful technique for improving simulation outcomes and extensions to increasingly complex problems will remain an important objective for the foreseeable future.

6 Acknowledgements

Significant portions of these notes are based on work performed by Karthik Mani and Bryan Flynt during their PhD graduate work at the University of Wyoming. We are also thankful for suggestions provided by Professor D. Darmofal and Dr. Behzad Ahrabi.

References

[1] Javier Bonet and Jaime Peraire. An alternating digital tree (ADT) algorithm for 3D geometric searching and intersection problems. *International Journal for Numerical Methods in Engineering*, 31(1):1–17, 1991.

[2] Martin D. Buhmann. *Radial Basis Functions*. Cambridge University Press, 2004.

[3] M. H. Carpenter and V. Vatsa. Higher order temporal schemes with error controllers for unsteady Navier-Stokes equations. In *17th AIAA Computational Fluid Dynamics Conference, Toronta, Ontario, Canada, June 2005*, 2005. AIAA Paper 2005–5245.

[4] Marco Ceze and Krzysztof J Fidkowski. Drag prediction using adaptive discontinuous finite elements. *Journal of Aircraft*, 51(4):1284–1294, 2014.

[5] F. Davoudzadeh, H. Mcdonald, and B.E. Thompson. Accuracy evaluation of unsteady CFD numerical schemes by vortex preservation. *Computer in FLuids*, 24(1):1995, 1995.

[6] D. Estep, M. Holst, and M. Larson. Generalized Green's functions and the effective domain of influence. *SIAM Journal on Scientific Computing*, 26(4):1314–1339, 2005.

[7] Krzysztof J Fidkowski and Yuxing Luo. Output-based space–time mesh adaptation for the compressible navier–stokes equations. *Journal of Computational Physics*, 230(14):5753–5773, 2011.

[8] Bryan T. Flynt and Dimitri J. Mavriplis. Efficient adaptation using discrete adjoint error estimates in unsteady flow problems. 2013. AIAA Paper 2012-520.

[9] M. B. Giles and E. Suli. Adjoint methods for PDEs: a posteriori error analysis and postprocessing by duality. *Acta Numerica*, pages 145–236, 2002.

[10] R. Hartmann. Adjoint consistency analysis of discontinuous Galerkin discretizations. *SIAM Journal on Numerical Analysis*, 45(6):2671–2696, 2007.

[11] P. Houston, R. Rannacher, and E. Suli. A posteriori error analysis for stabilized finite-element approximations of transport problems. *Comput. Methods Appl. Mech. Engr.*, 190:1483–1508, 2000.

[12] K. Mani and D. J. Mavriplis. Efficient solutions of the Euler equations in a time-adaptive space-time framework. AIAA Paper 2011-774, 49th Aerospace Sciences Meeting, Orlando, FL, January 2011.

[13] Dimitri J. Mavriplis. Adaptive meshing techniques for viscous flow calculations on mixed element unstructured meshes. *International Journal for Numerical Methods in Fluids*, 34(2):93–111, 2000.

[14] Marian Nemec, Michael Aftosmis, and Mathius Wintzer. Adjoint-based adaptive mesh refinement for complex geometries. 2008. AIAA Paper 2008-0725.

[15] T. A. Oliver and D. Darmofal. Analysis of dual consistency for discontinuous Galerkin discretizations of source terms. *SIAM Journal on Numerical Analysis*, 47(5):3507–3525, 2009.

[16] Michael A Park and David L Darmofal. Validation of an output-adaptive, tetrahedral cut-cell method for sonic boom prediction. *AIAA journal*, 48(9):1928–1945, 2010.

[17] N. A. Pierce and M.B. Giles. Adjoint recovery of superconvergent functionals from PDE approximations. *SIAM Review*, 42(2):247–264, 2000.

[18] Niles A Pierce and Michael B Giles. Adjoint and defect error bounding and correction for functional estimates. *Journal of Computational Physics*, 200(2):769–794, 2004.

[19] A. Sei and W. W. Symes. A note on consistency and adjointness for numerical schemes. Technical Report TR95-04, Department of Computational and Applied Mathematics, Rice University, 1995.

[20] David A Venditti and David L Darmofal. Adjoint error estimation and grid adaptation for functional outputs: Application to quasi-one-dimensional flow. *Journal of Computational Physics*, 164(1):204–227, 2000.

[21] David A Venditti and David L Darmofal. Grid adaptation for functional outputs: application to two-dimensional inviscid flows. *Journal of Computational Physics*, 176(1):40–69, 2002.

[22] L. Wang and D. J. Mavriplis. Adjoint based h-p adaptive discontinuous Galerkin methods for the 2D compressible Euler equations. *Journal of Computational Physics*, 228(20):7643–7661, November 2009.

[23] Holger Wendland. Piecewise polynomial, positive definite and compactly supported radial functions of minimal degree. *Advances in Computational Mathematics*, 4(1):389–396, 1995.

[24] D.L. Whitfield and J.M. Janus. Three-dimensional unsteady Euler equations solution using flux vector splitting. 1984. AIAA Paper 84-1552.

[25] H.C. Yee, N. Sandham, and M Djomehri. Low dissipative high order shock-capturing methods using characteristic-based filters. *Journal of Computational Physics*, 150(1):199–238, 1999.