18th AIAA Computational Fluid Dynamics Conference
25 - 28 June 2007, Miami, FL

AIAA 2007-3944

Discrete Adjoint Based Time-Step Adaptation and Error Reduction in Unsteady Flow Problems

Karthik Mani [*] and Dimitri J. Mavriplis [†]

Department of Mechanical Engineering, University of Wyoming, Laramie, Wyoming 82071-3295

The paper presents an adjoint based approach for determining global error in the time domain that is relevant to scalar outputs computed as functions of the unsteady flow solution. The algorithm is derived for the unsteady Euler equations and takes into account the effect of dynamic meshes. Two primary components of the total error are specifically identified, namely, the error due to temporal resolution and the error due to partial convergence of the governing equations at each implicit time step. The primary error components are further decomposed into individual contributions arising from the flow equations and the mesh motion equations. The distribution of the global error from these various components is then used as the criterion for adaptation. The developed method is applied to a simple unsteady test case involving a sinusoidally pitching airfoil in order to demonstrate its strength and features.

I. Introduction

The time-integration process in unsteady flow problems is typically carried out using a uniform time step which is applied throughout the time domain of interest. The limitation on the size of the time step is mainly governed by the time resolution required to capture essential unsteady flow features and deliver acceptably low temporal error. The inherent non-linearity of the flow equations makes it difficult to determine regions in the time domain that are most significant for these tasks. The traditional approach is to use a uniform time step of a size that by engineering knowledge is known to be sufficient to capture essential flow features. The disadvantage of such a procedure is that it becomes expensive due to high resolution in the regions of the time domain where such resolution may not be necessary. Additionally, while the overall cost of an unsteady simulation may be correlated against the resolution of the time domain, the primary cost of obtaining the solution of the flow equations at each time-level arises from the non-linear iterative solution procedure employed. Typically the equations are never solved to machine precision unless the spatial resolution of the problem under consideration is relatively small. Additionally, convergence to small tolerance levels can become time consuming for stiff problems particularly in the presence of anisotropic mesh effects. A sensible guideline for determining suitable implicit system convergence levels is that the algebraic error resulting from incomplete system convergence at each time step be reduced to levels below the local temporal discretization error at each time step.[1,2] However, this requires a good measure of the local temporal error as well as some estimate of the algebraic error due to incomplete system convergence, and is seldom enforced. The end result is that, as in the case of temporal resolution, some predetermined constant convergence limit is set based purely on engineering judgment such that the resulting solution has acceptably low overall error at practical computational expense. It is quite possible that the equations converged to limits set by such ad-hoc methods could result in unnecessarily restrictive convergence in some regions of the time domain and insufficient convergence in others. Under most circumstances the effect or error due to insufficient convergence becomes difficult to quantify and even harder to remedy.

This paper presents a novel approach for estimating the error in a scalar output computed as a function of the unsteady flow solution, and the distribution of this error in the time domain. The contributions to the total error are distinctly identified as arising from the two primary sources, namely the temporal resolution and the effect of partial convergence of the equations at each time step. Additional separation of the error into flow and mesh components provides the criteria necessary to target specifically the set of governing equations that influences the functional output the most. This is uniquely different from common adaptation or time-step control schemes which are based on estimates of the local temporal error at each time step, in that we estimate and adapt based on the global temporal error on the

[*]Graduate Student, AIAA Member; email: kmani@uwyo.edu.
[†]Professor, AIAA Associate Fellow; email: mavripl@uwyo.edu.

final solution objective. The primary mechanism used for this purpose is the application of time dependent discrete adjoint equations on a Taylor expansion of the scalar functional.

The total error incurred in time dependent simulations can be broken down into spatial error, temporal error, and algebraic error. For steady-state problems, adjoint methods have been used successfully to drive adaptive mesh refinement schemes for reducing the spatial discretization error for output functionals of interest.[3–6] For time dependent solutions, the use of adaptive mesh refinement techniques requires the construction of a new refined mesh at each time step, which results in discontinuous processes in time, as new grid points are added and deleted between time steps. While the full error reduction and control problem for time dependent problems must involve simultaneous adaptation in time and in space, the current work is restricted to adaptation in the time domain, in order to reduce temporal error, with no attempt to reduce spatial discretization error. However, algebraic error due to incomplete convergence of the governing equations at each time step is also targeted in this work.

Much work has been done on temporal adaptation and error control in the absence of spatial adaptation, particularly for ordinary differential equations.[7,8] The most common approach consists of using local temporal error estimation to drive time-step size selection, where the temporal error at a given time step is estimated by discretizing the time derivative with different orders of accuracy and comparing the corresponding results.[9] Also, such methods implicitly assume that any solutions obtained are numerically exact, or at least converged to levels below the local temporal discretization error, and do not consider the effect of partially converged solutions. While local error estimation has been used successfully for temporal refinement, a more appealing approach would be one that targets the global temporal error for an objective of interest directly, since for any particular objective, there is no guarantee that a gradient based or any other local error based adaptation of the time domain would necessarily lead to an improved estimate of the objective. As in the case of spatial error estimation, this arises from the fact that there may be little or no correlation between the flow solution gradients in the time domain and the scalar objective of interest. If direct information such as the sensitivity of the objective to the solution at each time interval, which in turn is a function of the time-step size and convergence criterion at that interval were available, a more targeted and efficient time adaptation scheme can be developed. While adjoint methods have been used in the spatial domain for goal oriented error estimation and control, their use in the time domain has mostly been confined to simpler problems such as those involving ordinary differential equations.[10–12]

In this paper, we apply time-dependent adjoint-based techniques for targeting and reducing directly the global temporal error for engineering functional outputs of unsteady flow simulations with dynamically deforming meshes. The work presented in this paper is a direct result of recent advances made in solving discrete adjoint equations in unsteady flow problems with dynamic meshes for the purpose of determining sensitivities used in shape optimization.[13] This unsteady adjoint solution procedure forms the basis for computing derivatives in the linearization of the objective in the time domain. Also, the developed algorithm imposes no restrictions on the temporal evaluation of the scalar objective. The scalar objective may be defined as a function of the flow solution only at the end of the time-integration process or be a time-integrated quantity by itself.

In steady problems, intuitive estimates may be made to determine regions in the spatial domain that would most likely affect a scalar functional quantity of interest. For example, the computation of drag on an airfoil intuitively requires added resolution near the surface of the airfoil for accurate drag values. The process is far more difficult to apply when extended to time dependent flows. The local temporal error from a particular region in the time domain could either have a large or small impact on objective accuracy, and possibly completely counter to intuition. Additionally, it would appear to be practically impossible to apply intuition for the separation of two sources of error, namely temporal discretization effects and algebraic error effects due to incomplete system convergence. The algorithm presented in the paper provides a straightforward method for determining not only the total temporal error that is relevant to the scalar functional, but also for decomposing it into time resolution and partial convergence components, and within each component for identifying quantitatively the contributions from the flow equations and the mesh equations, all in a cost effective manner.

II. Governing Equations of the Flow Problem in ALE Form

The conservative form of the Euler equations is used in solving the flow problem. The paper is limited to inviscid flow problems since the primary focus is the development and validation of a global temporal error estimation procedure. Extension of the algorithm to viscous flow problems with the inclusion of turbulence models should prove to be straightforward. The differences arise only in the linearization of the additional flux contributions and not in the base formulation presented in the paper. In vectorial form the conservative form of the Euler equations may be written as:

$$\frac{\partial \mathbf{U}(\mathbf{x},t)}{\partial t} + \nabla \cdot \mathbf{F}(\mathbf{U}) = 0 \tag{1}$$

Applying the divergence theorem and integrating over a moving control volume $A(t)$ yields:

$$\int_{A(t)} \frac{\partial \mathbf{U}}{\partial t} dA + \int_{dB(t)} \mathbf{F}(\mathbf{U}) \cdot \mathbf{n} dB = 0 \tag{2}$$

Using the differential identity:

$$\frac{\partial}{\partial t} \int_{A(t)} \mathbf{U} dA = \int_{A(t)} \frac{\partial \mathbf{U}}{\partial t} dA + \int_{dB(t)} \mathbf{U}(\dot{\mathbf{x}} \cdot \mathbf{n}) dB \tag{3}$$

equation (2) is rewritten as:

$$\frac{\partial}{\partial t} \int_{A(t)} \mathbf{U} dA + \int_{dB(t)} [\mathbf{F}(\mathbf{U}) - \dot{\mathbf{x}}\mathbf{U}] \cdot \mathbf{n} dB = 0 \tag{4}$$

or when considering cell-averaged values for the state $\mathbf{U}$ and integrating over the entire time domain as:

$$\int_0^T \frac{\partial A\mathbf{U}}{\partial t} dt + \int_0^T \int_{dB(t)} [\mathbf{F}(\mathbf{U}) - \dot{\mathbf{x}}\mathbf{U}] \cdot \mathbf{n} dB dt = 0 \tag{5}$$

This is the Arbitrary-Lagrangian-Eulerian (ALE) finite-volume form of the Euler equations. The equations are required in ALE form since the problem involves deforming meshes where mesh elements change in shape and size at each time-step. Here A refers to the area of the control volume, $\dot{\mathbf{x}}$ is the vector of mesh face or edge velocities, and $\mathbf{n}$ is the unit normal of the face or edge. The state vector $\mathbf{U}$ of conserved variables and the cartesian inviscid flux vector $\mathbf{F} = (\mathbf{F}^x, \mathbf{F}^y)$ are:

$$\mathbf{U} = \begin{pmatrix} \rho \\ \rho u \\ \rho v \\ E_t \end{pmatrix}, \quad \mathbf{F}^x = \begin{pmatrix} \rho u \\ \rho u^2 + p \\ \rho u v \\ u(E_t + p) \end{pmatrix}, \quad \mathbf{F}^y = \begin{pmatrix} \rho v \\ \rho u v \\ \rho v^2 + p \\ v(E_t + p) \end{pmatrix}, \tag{6}$$

Here ρ is the fluid density, (u, v) are the cartesian fluid velocity components, p is the pressure and E_t is the total energy. For an ideal gas, the equation of state relates the pressure to total energy by:

$$p = (\gamma - 1)\left[E_t - \frac{1}{2}\rho(u^2 + v^2)\right] \tag{7}$$

where $\gamma = 1.4$ is the ratio of specific heats.

III. Mesh Motion Strategy

The deformation of the mesh is achieved through the linear tension spring analogy which approximates the mesh as a network of inter-connected springs. The spring coefficient is taken to be inversely proportional to the edge length. Two independent force balance equations are formulated for each node based on displacements of neighbors. This results in a nearest neighbor stencil for the final linear system to be solved. The stiffness matrix [K] is a sparse block matrix based on the initial configuration of the mesh and remains unchanged through the time-integration process. The linear system in equation (8) relates the interior vertex displacements in the mesh to known displacements on the boundaries.

$$[K]\delta \mathbf{x}_{int} = \delta \mathbf{x}_{surf} \tag{8}$$

For each node in the mesh, the displacement can be solved as:

$$\delta \mathbf{x}_{int_i} = \frac{\delta \mathbf{x}_{surf_i} - \sum_{j=1}^{nj} [O]_{ij} \delta \mathbf{x}_{\mathbf{j}}}{[D]_i} \tag{9}$$

where the off-diagonal term $[O]_{ij}$ and diagonal term $[D]_i$ are 2×2 diagonal matrices as follows:

$$[O]_{ij} = -k_{ij}[I] \tag{10}$$

$$[D]_i = \sum_{j=1}^{nj} k_{ij}[I] \tag{11}$$

The mesh motion equations are elliptic in nature and can be solved using conventional smoothers such as Jacobi or Gauss-Seidel iterations. The convergence of the system is a function of the mesh resolution and becomes prohibitively expensive as the mesh resolution is increased, since the mesh motion equations have to be solved at each time step once for the flow solution and once for the sensitivity computation. An agglomeration multigrid approach is used to speed up convergence of the linear system. Being elliptic in nature the system is particularly well suited for acceleration using multigrid.

IV. The Discrete Geometric Conservation Law (GCL)

The discrete geometric conservation law (or GCL) requires that a uniform flow field be preserved when equation (4) is integrated in time. In other words the motion or deformation of the computational mesh should not introduce conservation errors in the solution of the flow problem. This translates into $\mathbf{U} = constant$ being an exact solution of equation (4). For a conservative scheme, the integral of the inviscid fluxes around a closed contour goes to zero when $\mathbf{U} = constant$. Applying these conditions to equation (4) results in the mathematical description of the GCL as stated below:

$$\frac{\partial A}{\partial t} - \int_{\partial\Omega(t)} \dot{\mathbf{x}} \cdot \mathbf{n} d\Omega = 0 \tag{12}$$

Equation (12) implies that change in area(volume in 3D) of a control volume should be equal to the area(volume in 3D) swept by the boundary of the control volume. The vector of cartesian edge velocities $\dot{x}$ for each of the edges encompassing the control volume must therefore be chosen such that equation (12) is satisfied. There are many different methods to compute the edge velocities while satisfying the GCL, but while it has been shown that satisfying the GCL is essential for non-linear stability, it is not a sufficient condition to achieve the underlying accuracy of the chosen time-integration scheme .[14] A GCL formulation has been specifically derived for first, second and third order backward difference (BDF) time-integration schemes and also for higher-order accurate implicit Runge-Kutta schemes in Ref.,[15] and this form is used exclusively in this work.

V. Convergence Acceleration using Linear Multigrid

A linear geometric agglomeration multigrid strategy[16] is used to accelerate convergence of all the linear systems encountered in the solution process, be either, the flow, mesh motion, flow adjoint or the mesh adjoint equations. The linear multigrid method uses a correction scheme where coarser levels recursively provide corrections to the fine grid solution. The coarse level meshes are built by repeated agglomeration or merging of neighboring control volumes to form a single control volume. The subscripts or superscripts H and h in this section refer to coarse and fine mesh levels in the spatial domain.

Consider a linear system developed by discretizing and linearizing a non-linear equation on the fine level mesh.

$$[A]_h \mathbf{x}_h = \mathbf{b}_h \tag{13}$$

We can obtain an approximate solution to the linear system using conventional iterative methods such as Jacobi or Gauss-Seidel iterations. The residual of the linear system can then written as:

$$[A]_h \bar{\mathbf{x}}_h - \mathbf{b}_h = \mathbf{r}_h \tag{14}$$

where $\bar{\mathbf{x}}_h$ is the approximate solution to the linear system. Let us define a correction $\mathbf{x}'_h$ to the approximate solution such that the residual of the linear system on the fine level mesh goes to zero.

$$\mathbf{x}_h = \bar{\mathbf{x}}_h + \mathbf{x}'_h \tag{15}$$

By taking the difference between the correction equation and the residual equation on the fine mesh we obtain a new linear system as:

$$[A]_h \mathbf{x}'_h = -\mathbf{r}_h \tag{16}$$

We now transfer this equation to a coarse level to form a coarse level correction equation:

$$[A]_H \mathbf{x}'_h = -I_h^H \mathbf{r}_h \tag{17}$$

where the subscript H refers to the coarse level and I_h^H is the fine to coarse level restriction operator. This system can now be iteratively solved approximately or exactly if H is coarse enough. If H is not coarse enough a new approximate solution is obtained and the residual of this system is then transferred to a coarser level. This process is repeated until H is coarse enough to obtain a numerically exact solution to the system. The exact or approximate solutions obtained from the coarse levels are used to correct the fine grid solution as

$$\bar{\mathbf{x}}_h^{new} = \bar{\mathbf{x}}_h + I_H^h \mathbf{x}'_H \tag{18}$$

where I_H^h refers to the coarse to fine grid prolongation operator. For our purposes, we use summation of parent residuals as the restriction operator for the right hand side of the linear system. For the flow problem, the coefficient matrix $[A]_H$ (which is the flow Jacobian matrix) for the coarse levels are constructed by first restricting the fine level flow solution through a parent element area weighted approach, and then using the restricted flow solution to build the flow Jacobian matrix. In the case of mesh motion, we use summation of the edge spring coefficients as the restriction operator to construct the coarse level stiffness matrix $[K]_H$. For all cases we use direct injection of coarse level corrections into parent elements as the prolongation operator.

VI. Error due to Temporal Resolution

A. Objective formulation and linearization with respect to time resolution

Consider an objective that is either a function of the flow solution and mesh coordinates at the end of the time-integration process or a time-integrated quantity that is dependent on the flow solution and mesh coordinates at each time step. Mathematically this may be represented as $L = L(U,x)$ where L is the generic scalar objective function, U is the set of all flow solutions in the time domain and x is the set of all spatial meshes in the time domain. The objective evaluated on a fine resolution time domain is represented by $L_h(U_h, x_h)$ and the same function evaluated on a coarser time domain is represented by $L_H(U_H, x_H)$. From this point onward, H refers to the coarse time domain and h refers to the fine resolution time domain (as opposed to the previous use of this notation for the coarse and fine spatial domains). We can estimate a value for L_h on the fine time domain without solving the flow equations on the fine domain by first projecting the coarse solution U_H and the coarse time domain mesh coordinates x_H onto the fine time domain and then constructing the objective based on these projections as $L_h = L_h(U_h^H, x_h^H)$. Here U_h^H and x_h^H represent the coarse time domain solution and the coarse time domain mesh coordinates projected onto the fine resolution time domain. For the purposes of the derivation presented in this section we shall assume that the coarse time domain flow solution U_H and the coarse time domain mesh coordinates x_H have been obtained via full convergence of the respective equations. The Taylor series expansion of the exact fine time domain objective about the estimate $L_h(U_h^H, x_h^H)$ can then be expressed as:

$$L_h(U_h, x_h) = L_h(U_h^H, x_h^H) + \left[\frac{\partial L}{\partial U}\right]_{U_h^H, x_h^H} \left(U_h - U_h^H\right) + \left[\frac{\partial L}{\partial x}\right]_{x_h^H, U_h^H} \left(x_h - x_h^H\right) + \cdots \tag{19}$$

The vectors $[\partial L/\partial U]_{U_h^H, x_h^H}$ and $[\partial L/\partial x]_{x_h^H, U_h^H}$ are the sensitivities of the fine objective function L_h with respect to the fine solution U_h and fine mesh coordinates x_h evaluated using U_h^H and x_h^H which are the coarse solution and mesh coordinates projected onto the fine time domain. The differences $\left(U_h - U_h^H\right)$ and $\left(x_h - x_h^H\right)$ represent the error between the exact values of U_h and x_h, which would be obtained by solving the problem on the fine domain, against the values projected onto fine time domain from the coarse domain. If $nsteps$ is defined as the number of time steps in the time-integration, and $ncells$ and $nnodes$ the number of elements and nodes in the spatial domain, then U_h, U_h^H, x_h and x_h^H are vectors of size $[nsteps \times 1]$ where each of the elements in the vectors are by themselves a vector of size $[ncells \times 1]$ for the flow variables and vector of size $[nnodes \times 1]$ for the mesh nodes. This indicates that $[\partial L/\partial U]_{U_h^H, x_h^H}$ is a vector of size $[1 \times nsteps]$, where each element in the vector is a again a vector of size $[1 \times ncells]$. The same argument applies for the sensitivity to mesh coordinates. If the objective L is evaluated as a function of the solution and coordinates only at the final time interval, then the vector of sensitivities of the objective to the solution and the vector of sensitivities of the objective to the mesh coordinates are non-zero only at the final time interval.

The procedure for computing an unsteady solution involves obtaining the solution to the nonlinear residual operator $R(U,x)$ at each time interval. The nonlinear operator $R(U,x)$ is constructed by some appropriate discretization of the governing flow equations. For this work, a cell-centered finite-volume discretization of the Euler equations on an unstructured triangular mesh is used. When the time derivative term in the Euler equations is based on a simple first order backward difference formula (BDF1), the nonlinear residual is defined as $R^n = R^n(U^n, U^{n-1}, x^n, x^{n-1}) = 0$, where n refers to the time index. The solution of the nonlinear system at each time interval is obtained using Newton iterations of the form:

$$\left[\frac{\partial R(U^k)}{\partial U^k}\right] \delta U^k = -R(U^k, x) \tag{20}$$

$$U^{k+1} = U^k + \delta U^k \tag{21}$$

$$\delta U^k \to 0, U^{k+1} = U^n \tag{22}$$

where the Jacobian is inverted (iteratively) using the linear multigrid algorithm. Expanding a fine time-domain residual set about the coarse time domain set of residuals yields:

$$R_h(U_h, x_h) = R_h(U_h^H, x_h^H) + \left[\frac{\partial R}{\partial U}\right]_{U_h^H, x_h^H} (U_h - U_h^H) + \left[\frac{\partial R}{\partial x}\right]_{x_h^H, U_h^H} (x_h - x_h^H) + \cdots = 0 \tag{23}$$

As in the case of the flow solution vectors, the vector R_h represents the set of all residuals in the discretized time domain with each element in the vector representing the set of all residuals in the spatial domain for that time interval. Along the same lines, the matrices $[\partial R/\partial U]$ and $[\partial R/\partial x]$ are block matrices consisting of the sensitivities of all the residuals R in the time domain to the set of all solutions U and set of all mesh coordinates x in the time domain. Since for a BDF1 time scheme, the residual R^n at time index n is a function of only the solutions and mesh coordinates at time indices n and $n-1$, the matrices $[\partial R/\partial U]$ and $[\partial R/\partial x]$ are nonzero at only two locations for each row (each time interval) namely the diagonal and the immediate off-diagonal to the left. This is more clearly exemplified in equations (24) (25).

$$\left[\frac{\partial R}{\partial U}\right] = \begin{pmatrix} \ddots & & & \\ \ddots & \ddots & & \\ & \left[\frac{\partial R^{n-1}}{\partial U^{n-2}}\right] & \left[\frac{\partial R^{n-1}}{\partial U^{n-1}}\right] & \\ & & \left[\frac{\partial R^{n}}{\partial U^{n-1}}\right] & \left[\frac{\partial R^{n}}{\partial U^{n}}\right] \end{pmatrix} \tag{24}$$

$$\left[\frac{\partial R}{\partial x}\right] = \begin{pmatrix} \ddots & & & \\ \ddots & \ddots & & \\ & \left[\frac{\partial R^{n-1}}{\partial x^{n-2}}\right] & \left[\frac{\partial R^{n-1}}{\partial x^{n-1}}\right] & \\ & & \left[\frac{\partial R^{n}}{\partial x^{n-1}}\right] & \left[\frac{\partial R^{n}}{\partial x^{n}}\right] \end{pmatrix} \tag{25}$$

Each diagonal element of the matrix in equation (24) is the flow Jacobian matrix used in the nonlinear solution process at that time interval. Referring back to equation (23), since the nonlinear residual operator R must vanish for a converged solution at each time interval, an estimate for the error vector $(U_h - U_h^H)$ at each time interval in equation (19) can obtained by rearranging equation (23) as:

$$(U_h - U_h^H) \approx -\left[\frac{\partial R}{\partial U}\right]^{-1}_{U_h^H, x_h^H} \left\{ R_h(U_h^H, x_h^H) + \left[\frac{\partial R}{\partial x}\right]_{x_h^H, U_h^H} (x_h - x_h^H) \right\} \tag{26}$$

It should be noted that this is merely an estimate for the error since higher-order terms in the Taylor expansion are ignored. Substituting equation (26) into equation (19) results in:

$$L_h(U_h,x_h) \approx L_h(U_h^H,x_h^H) \underbrace{- \left[\frac{\partial L}{\partial U}\right]_{U_h^H,x_h^H} \left[\frac{\partial R}{\partial U}\right]^{-1}_{U_h^H,x_h^H} \left\{ R_h(U_h^H,x_h^H) + \left[\frac{\partial R}{\partial x}\right]_{x_h^H,U_h^H} \left(x_h - x_h^H\right) \right\} + \left[\frac{\partial L}{\partial x}\right]_{x_h^H,U_h^H} \left(x_h - x_h^H\right)}_{\varepsilon_{cc}} \tag{27}$$

$L_h(U_h,x_h)$ in equation (27) can be described as an estimate for the exact objective evaluated directly on the fine time domain. Based on the derivation it is approximately equal to the sum of the objective 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}.

B. Evaluation of temporal resolution error correction term ε_{cc} and decomposition into flow and mesh contributions

The correction term ε_{cc} as defined in the previous subsection is:

$$\varepsilon_{cc} = - \left[\frac{\partial L}{\partial U}\right]_{U_h^H,x_h^H} \left[\frac{\partial R}{\partial U}\right]^{-1}_{U_h^H,x_h^H} \left\{ R_h(U_h^H,x_h^H) + \left[\frac{\partial R}{\partial x}\right]_{x_h^H,U_h^H} \left(x_h - x_h^H\right) \right\} + \left[\frac{\partial L}{\partial x}\right]_{x_h^H,U_h^H} \left(x_h - x_h^H\right) \tag{28}$$

Since computing, storing and inverting the flow Jacobian matrix is expensive, a flow adjoint variable Λ_U is defined to aid the evaluation procedure. The flow adjoint variable is defined as:

$$\Lambda_{U_h}^T|_{U_h^H x_h^H} = - \left[\frac{\partial L}{\partial U}\right]_{U_h^H,x_h^H} \left[\frac{\partial R}{\partial U}\right]^{-1}_{U_h^H x_h^H} \tag{29}$$

Transposing and rearranging equation (29) yields

$$\left[\frac{\partial R_h}{\partial U_h}\right]^T_{U_h^H x_h^H} \Lambda_{Uh}|_{U_h^H,x_h^H} = - \left[\frac{\partial L}{\partial U}\right]^T_{U_h^H} \tag{30}$$

The solution of equation (30) 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 for the flow adjoint. 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 (30) recast on the coarse time domain becomes:

$$\left[\frac{\partial R}{\partial U}\right]^T_{U_H,x_H} \Lambda_{UH} = - \left[\frac{\partial L}{\partial U}\right]^T_{U_H,x_H} \tag{31}$$

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

$$\Lambda_{Uh} = I_h^H \Lambda_{UH} \tag{32}$$

where I_h^H is some appropriate projection operator.

Given the structure of the matrix $\partial R/\partial U$, the evaluation of the flow adjoint involves an integration backward in time beginning at the final time step. This becomes clear by substituting equation (24) into equation (30) and rewriting in the expanded form shown below.

$$\begin{pmatrix} \ddots & \ddots & & \\ & \left[\frac{\partial R^{n-2}}{\partial U^{n-2}}\right]^T & \left[\frac{\partial R^{n-1}}{\partial U^{n-2}}\right]^T & \\ & & \left[\frac{\partial R^{n-1}}{\partial U^{n-1}}\right]^T & \left[\frac{\partial R^{n}}{\partial U^{n-1}}\right]^T \\ & & & \left[\frac{\partial R^{n}}{\partial U^{n}}\right]^T \end{pmatrix} \begin{pmatrix} \vdots \\ \Lambda_{U_H}^{n-2} \\ \Lambda_{U_H}^{n-1} \\ \Lambda_{U_H}^{n} \end{pmatrix} = - \begin{pmatrix} \vdots \\ \left[\frac{\partial L}{\partial U^{n-2}}\right]^T \\ \left[\frac{\partial L}{\partial U^{n-1}}\right]^T \\ \left[\frac{\partial L}{\partial U^{n}}\right]^T \end{pmatrix} \tag{33}$$

This system can be solved by (block) back-substitution, which corresponds to a backward integration in time. The procedure begins at the final time step, by first obtaining the solution of the following linear system on the coarse time domain.

$$\left[\frac{\partial R^n}{\partial U^n}\right]^T \Lambda_{U_H^n} = -\left[\frac{\partial L}{\partial U^n}\right]^T \tag{34}$$

The solutions of the adjoint at time indices $n-1$,$n-2$ and so forth cascading all the way to the first time index can be obtained by solving linear systems of the form:

$$\left[\frac{\partial R^{n-1}}{\partial U^{n-1}}\right]^T \Lambda_{U_H^{n-1}} = -\left[\frac{\partial L}{\partial U^{n-1}}\right]^T - \left[\frac{\partial R^n}{\partial U^{n-1}}\right]^T \Lambda_{U_H^n} \tag{35}$$

Once the vector of adjoint variables Λ_{UH} is obtained the first contributions to the computed correction may be determined by projecting the flow adjoint onto the fine time domain and then evaluating a vector inner product as follows:

$$\varepsilon_{cc_1} = \Lambda_{U_h^H}^T R_h(U_h^H, x_h^H) \tag{36}$$

ε_{cc_1} represents the contribution from the flow equations to the error arising due to insufficient temporal mesh resolution. The residual $R_h(U_h^H, x_h^H)$ is nonzero since it is computed using the projection of the coarse time domain flow solution onto the fine time domain. Closer examination reveals that equation (36) is actually a summation of the vector inner products of the flow adjoint and the nonzero residual at each time step on fine time domain. The remaining contributions to the total correction term may be written in combined form as:

$$\varepsilon_{cc_2} = \left\{ \Lambda_{U_h}^T \left[\frac{\partial R_h}{\partial x_h}\right]_{U_h^H x_h^H} + \left[\frac{\partial L_h}{\partial x_h}\right]_{x_h^H} \right\} \left(x_h - x_h^H\right) = \lambda_x \left(x_h - x_h^H\right) \tag{37}$$

using the notation λ_x as shorthand for the large bracketed term in the middle of the above equation. We now define a mesh residual equation G and write a Taylor expansion about its value on the coarse time domain as:

$$G(x_{int}) = [K]\delta x_{int} - \delta x_{surf} = 0 \tag{38}$$

$$G(x_h) = G(x_h^H) + \left[\frac{\partial G}{\partial x}\right]_{x_h^H} (x_h - x_h^H) + \cdots = 0 \tag{39}$$

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

$$(x_h - x_h^H) = -[\mathbf{K}]^{-1} G(x_h^H) \tag{40}$$

Substituting this back into equation (37) gives us an expression for the second and final correction term as:

$$\varepsilon_{cc_2} = -\lambda_{x_h} [\mathbf{K}]^{-1} G(x_h^H) \tag{41}$$

Defining a mesh adjoint variable Λ_x permits an iterative solution and avoids inversion of the stiffness matrix:

$$[\mathbf{K}]^T \Lambda_{xh} = -\lambda_{xh}{}^T \tag{42}$$

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

$$[\mathbf{K}]^T \Lambda_{xH} = -\lambda_{xH}{}^T \tag{43}$$

$$\Lambda_{xh} = I_h^H \Lambda_{xH} \tag{44}$$

A mesh adjoint vector must be solved for at each time step during the backward sweep in time. The stiffness matrix $[\mathbf{K}]$ is actually a block diagonal matrix in time consisting of the constant spatial stiffness matrix $[K]$ for each time step. The final form for the correction term can now be expressed as:

$$\varepsilon_{cc_2} = \Lambda_{x_h^H}^T G(x_h^H) \tag{45}$$

ε_{cc_2} represents the contribution of the mesh motion equations to error arising due to insufficient time resolution. Just as in the case of the flow adjoint, equation (45) 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 example are sinusoidal 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 objective 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 total error distribution ($\varepsilon_{cc_1} + \varepsilon_{cc_2}$)and can thus be conveniently used as the adaptation parameter for identifying regions that require higher time resolution.

VII. Error due to Partial Convergence

A. Objective formulation and linearization with respect to partial convergence

Consider the solution of the problem on the coarse time domain. In the former sections we assumed that the solution on the coarse time domain was 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. Additionally, also consider the objective L evaluated on the coarse time domain as a function of the fully converged solution. If the flow and mesh equations were partially converged on the coarse time domain, the error resulting in the objective evaluated on the coarse time domain may be linearly approximated as:

$$L_H(U_H,x_H) - \overline{L}_H(\overline{U}_H,\overline{x}_H) \approx \left[\frac{\partial L}{\partial U}\right]_{\overline{U}_H,\overline{x}_H} (U_H - \overline{U}_H) + \left[\frac{\partial L}{\partial x}\right]_{\overline{x}_H,\overline{U}_H} (x_H - \overline{x}_H) \tag{46}$$

where $\overline{U}_H$ and $\overline{x}_H$ refer to the approximate values of the flow variables and mesh coordinates obtained through partial convergence of the respective equations. If the residual equations were evaluated using partially converged solutions then they would not be equal to zero. The Taylor expansions for the residual equations about the partially converged values can be written as:

$$R(U_H,x_H) = R(\overline{U}_H,\overline{x}_H) + \left[\frac{\partial R}{\partial U}\right]_{\overline{U}_H,\overline{x}_H} (U_H - \overline{U}_H) + \left[\frac{\partial R}{\partial x}\right]_{\overline{x}_H,\overline{U}_H} (x_H - \overline{x}_H) + \cdots = 0 \tag{47}$$

$$G(x_H) = G(\overline{x}_H) + \left[\frac{\partial G}{\partial x}\right]_{\overline{x}_H} (x_H - \overline{x}_H) + \cdots = 0 \tag{48}$$

B. Decomposition of error due to partial convergence into flow and mesh components

The similarity between the current derivation and the derivation of the time resolution error in the former sections is clear. Following the same procedure as the time resolution error derivation, we can obtain two contributions that add up to the total error due to partial convergence as:

$$\varepsilon_{cc_1} = {\Lambda_{UH}}^T R(\overline{U}_H,\overline{x}_H) \tag{49}$$

$$\varepsilon_{cc_2} = {\Lambda_{xH}}^T G(\overline{x}_H) \tag{50}$$

where the adjoint variables are solved for as:

$$\left[\frac{\partial R}{\partial U}\right]^T_{\overline{U}_H,\overline{x}_H} \Lambda_{UH} = -\left[\frac{\partial L}{\partial U}\right]^T_{\overline{U}_H,\overline{x}_H} \tag{51}$$

$$[\mathbf{K}]^T \Lambda_{xH} = -\lambda_{xH}^T \tag{52}$$

$$\lambda_{xH} = \left\{ \Lambda_{UH}^T \left[\frac{\partial R}{\partial x}\right]_{\overline{U}_H,\overline{x}_H} + \left[\frac{\partial L}{\partial x}\right]_{\overline{x}_H,\overline{U}_H} \right\} \tag{53}$$

Closer examination reveals that these systems are nearly identical to equations (31) and (43) with the only difference being that all quantities are evaluated using the partially converged flow and mesh solutions, rather than the fully converged values. In the case of the time resolution error, the adjoint variables were solved for on the coarse time domain using the 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, the adjoint variables do not have to be projected and are multiplied with non-zero residuals on the coarse time domain. As described earlier, the residuals on the coarse time domain are non-zero due to partial convergence.

Relaxing the assumption of fully converged coarse time domain solutions for the time resolution error does not affect the error estimated by that procedure. The important realization is that the error computed as temporal resolution error by relaxing this assumption includes the error due to the partial convergence of the flow and mesh equations on the coarse time domain. The time resolution error in reality is blind to what solution is projected from the coarse time domain, since our only enforcement in the derivation is that $R_h(U_h^H, x_h^H) = 0$ which can also be enforced as $R_h(\overline{U}_h^H, \overline{x}_h^H) = 0$. The error due to time resolution can be separated from the total error by subtraction of the error due to partial convergence.

VIII. Implementation Details

Evaluation of the correction terms and the objective requires projections of the solution, mesh coordinates, flow adjoint and mesh adjoint from the coarse time domain to the fine time domain. The fine resolution time domain is constructed by dividing each time step in the coarse time domain by two, thus doubling the number of time steps. A quintic spline of the time distribution of variables in the coarse time domain is used as the projection operator for interpolation onto the fine domain. Since the the evaluation of the adjoint variables involves a backward integration in time, the entire solution history is required a priori. It would be impractical to hold the solution history in memory while the unsteady solution is evaluated for later use by the adjoint solver. The total process is broken down into four separate parts in order to improve efficiency and lower the memory overhead. The flow solver integrates the governing equations forward in time for a given temporal mesh and writes the flow variables to the hard drive on-the-fly. The adjoint solver is then invoked which reads the unsteady flow solution from the hard drive and performs a backward integration in time. The computed adjoint variables are also written to the hard drive on-the-fly by the adjoint solver during the backward sweep. An interpolation program reads in the time history of the flow and adjoint variables from the hard drive and uses a quintic spline construction in time to determine the projection onto the fine time domain. The projected variables are again written to file. The final step is the actual error computation, where the projected variables are read-in from file and the error computed. The error computation program also applies the adaptation criteria to determine the new temporal mesh resolution and the new convergence thresholds. The most expensive aspect of the process remains the unsteady flow solution, with a rough breakdown into individual costs as 70% flow solution, 25% adjoint solution, 3% interpolation, and 2% error estimation and adaptation.

The adaptation procedure currently used is fairly straightforward where time intervals with resolution error higher than the time-integrated average of the total resolution error distribution are adapted by subdividing into two time steps. Similarly, time intervals where the convergence error is higher than the time-integrated average of the total convergence error have their convergence thresholds scaled by some predetermined value. Although primitive in nature, the adaptation procedure allows for equidistribution of error in the time domain. Once a fairly uniform distribution of error in the time domain has been achieved, it indicates that further adaptation is not possible since the adaptation algorithm would request uniform refinement in all regions of the time domain. This can be interpreted as having achieved the near optimum distribution of time steps and convergence limits for the given combination of objective function, flow conditions and initial time domain resolution.

IX. Validation

A. Test Case Description

The test case used to implement and verify the developed algorithm involves the computation of the lift coefficient on a pitching transonic NACA0012 airfoil at some arbitrary point in the pitch cycle. The goal is to improve the estimate for the lift coefficient using a combination of time step adaptation, convergence threshold adaptation, and inclusion of the correction term which is the summation of the error distribution in time. The free-stream Mach number and mean angle of attack were chosen to be $M_\infty = 0.825$ and $\alpha_0 = 0^o$. An amplitude of $\alpha_{max} = 5^o$ and a reduced frequency of $\omega = 0.1$ were selected to control the pitch cycle. The pivot point was chosen to be two chord lengths ahead of the leading edge of the airfoil. The time domain extends from $t = 0$ and $t = 31.4159$ in non-dimensional time and corresponds to a single period. The objective function i.e. the lift of the airfoil was chosen to be evaluated at the end of the time-integration process. The initial time integration consisted of 8 uniformly distributed time steps of $dt = 3.92699$ each. For reference purposes, the exact lift coefficient was determined by performing the time integration with 16384 fully converged uniform time steps. Figure (1) shows the unstructured triangular mesh around a NACA0012 airfoil that was used for all computations in the test case. Figures (2(a)) and (2(b)) show the time-variation of the lift through the time domain of interest. The parameter that was monitored in order to establish convergence was the root-mean-square average of the element residuals for the flow equations and the nodal residuals for the mesh motion equations. Figure (3) illustrates a typical convergence history for the flow and mesh residuals at one time step of the simulation. For this case, which employed a total of 16 time steps over the temporal domain, the flow residuals are reduced to machine zero in 38 multigrid cycles, while the mesh motion residuals require 30 cycles to achieve full convergence. The number of cycles required to acheive the partial convergence tolerances discussed in the subsequent section can be inferred to some degree from these figures, although these convergence rates depend to some degree on the size of the prescribed time step. The initial condition was determined by first solving the described unsteady problem over one period starting with a steady state solution and then using the solution from the final iteration which also corresponds to zero angle-of-attack during the pitch upstroke. We assume that the residuals, both flow and mesh at the initial condition are always zero irrespective of the adaptation or refinement that occurs beyond the starting point.

B. Validation of Error Correction Terms

Before applying the adaptation algorithm, numerical experiments using the test case described above were run in order to validate the correction terms. Although the ultimate goal of an adjoint error estimation technique is to quantify the total objective error (with respect to the infinite resolution value), the formulation of these methods is based on predicting a new value of the objective on a more accurate discretization, rather than the exact continuous value. Therefore, in order to validate the correctness of the implementation, the predicted errors must be compared against the errors obtained by recomputing the solution on the reference (refined) discretization rather than the exact values.

The first set of experiments (Case Set 1) consist of validating the correction computed for either partial convergence of the mesh motion equations or the partial convergence of the flow equations, or the combined correction provided for simultaneous partial convergence of the mesh motion and flow equations. The validation cases were designed such that the effect of all error components except the one being validated in each case are removed. Particularly, while validating partial convergence error estimates, the temporal resolution effects are removed by performing all computations using the same temporal resolution. The description of each case is presented below.

1. **Validation 1A** (Error due to partial mesh convergence): This case validates the effect of only partially converging the mesh motion equations by removing the effects of all other error components. The test case was solved on a time domain consisting of 16 uniformly spaced time steps by converging the flow equations to machine precision, but converging the mesh motion equations only partially to a limit of $1E^{-3}$. The error predicted by this computation was then compared against the exact error which was determined based on the fully converged solutions of the mesh and flow equations on the same time domain, i.e 16 uniform steps. The results of this test are shown in Table 1.

2. **Validation 1B** (Error due to partial flow convergence): The procedure employed for validating the error due to partial convergence of the flow equations is similar to that of verifying the predicted error due to partial mesh convergence. The test case was solved on the time domain consisting of 16 uniform time steps by fully coverging the mesh motion equations to machine precision, but only partially converging the flow at each time step to a limit of $0.8E^{-4}$. The predicted error was again compared against the exact error computed using the solution based on fully converged mesh and flow equations at a temporal resolution of 16 time steps. The results are shown in Table 1

3. **Validation 1C** (Error due to combined partial convergence of mesh and flow equations): As in the previous two cases, the procedure remains identical, with the only difference being the partial convergence of both the flow and mesh equations simultaneously to the previously prescribed limits rather than only one of the sets of the governing equations. The reference was once again the error based on the fully converged flow and mesh solution at 16 time steps. The results are shown in Table 1. It is interesting to note from Table 1 that partial convergence of the mesh equations has little effect on the accuracy of the functional value at least for this particular example.

Table 1. Validation of partial convergence correction terms

Description	Functional value	% Error vs. Target	% Predicted Error
Target functional - exact at 16 steps (fully converged)	-0.28576836616489	-	-
Partial mesh convergence at 16 steps with limit = 1e-3 (flow fully converged)	-0.285995701438330	-0.0796	-0.0743
Corrected for partial mesh convergence	-0.285783421090738	-0.0052	-
Partial flow convergence at 16 steps with limit = 0.8e-4 (fully converged mesh)	-0.289033070451402	-1.142	-1.143
Corrected for partial flow convergence	-0.285765384379180	0.001	-
Partial flow and mesh convergence at 16 steps (flow=0.8e-4,mesh=1e-3)	-0.289270768414725	-1.226	-1.223
Corrected for partial flow and mesh convergence	-0.285778143449756	-0.003	

The second case (Case 2) validates the correction provided for temporal resolution error. The effect of partial convergence is removed by employing full convergence to machine precision of all equations at all time levels.

1. **Validation Case 2** (Error due to temporal resolution): The goal of this test case is to validate the temporal resolution error correction term provided for a resolution scaling from 16 uniformly spaced time steps to 32 uniformly spaced time steps. The unsteady test case was solved on the coarse resolution time domain consisting of 16 time steps and the error between this domain and the fine resolution domain consisting of 32 steps was predicted. The predicted error was then compared against the exact error due to resolution which was computed by solving the test case directly on the fine time domain consisting of 32 time steps. All equations were converged to machine precision in order to remove the effect of partial convergence. The results from this case are shown in Table 2. It should be noted that although the temporal resolution error can be further decomposed into flow and mesh components for individual validation, this has not been done. In a practical sense, doing so would translate into solving one set of governing equations (either flow or mesh) directly on the fine resolution time domain and utilizing the results in the error estimation procedure for the remaining set of equations. The individual components are always available, since the total temporal resolution error is assembled from the sum of these individual components. In our work, we use only the total temporal resolution error for the purpose of adapting the temporal resolution.

Table 2. Validation of temporal resolution correction term

Description	Functional value	% Error vs. Target	% Predicted Error
Target functional - exact at 32 steps (fully converged)	-0.309957065250867	-	-
Fully converged flow and mesh at 16 steps	-0.285768366164898	+7.804	+7.463
Corrected for resolution from 16 to 32 steps	-0.3089006774509025	+0.341	-

The advantage of the adjoint error estimation procedure described in this work is that it permits the identification of the various sources of error, such as those due to partial convergence of the mesh motion or flow equations, and those due to the effect of temporal discretization. However, for computational efficiency reasons it is not practical to fully converge the flow or mesh motion equations or to compute solutions on the fine time discretization soley for error estimation purposes. According to the formulation described in the previous section, only the error components due to partial convergence, and due to the combined effect of partial convergence and temporal resolution can be obtained. In order to isolate the error solely due to temporal resolution, the error due to partial convergence must be subtracted from the combined error term, yielding an error estimate which corresponds to the effect of refining the time step while converging the governing equations to machine precision. Ultimately, it is important to be able to isolate these individual error components in order to be able to give the adaptation algorithm the ability to choose between increased convergence tolerances or time step refinement (or both) at any location in time.

The first part of Case Set 3 validates the combined predicted error due to temporal resolution and partial convergence, while the second part of Case Set 3 validates each individual component, which is the most crucial part of the problem.

1. **Validation Case 3A** (Total error due to temporal resolution and partial convergence): The methodology employed for this validation is a loose combination of those described in the first two cases. The reference functional was computed by solving the test case on 32 uniformly spaced time steps which represents the fine resolution time domain combined with full convergence of the flow and mesh motion equations. The predicted error was computed by solving the same problem on 16 time steps with partial convergence limits for both the flow and mesh motion equations. The predicted total error therefore includes the effect of partially converging the governing equations on the coarse time domain and also the effect of temporal resolution between 16 and 32 time steps. Table 3 shows the results of this validation case.

2. **Validation Case 3B** (Validation of individual error components within the total correction term): Since only the total combined correction which includes both the partial convergence errors and the temporal resolution error can be computed due to cost reasons, it is important that we separate individual error components and validate their accuracy, since any adaptation performed is based on these individual estimates. The partial convergence error on any particular temporal mesh resolution can be computed provided the solutions to the forward and adjoint problems were obtained on that temporal resolution. For this particular case, we predict the total error which is an estimate of the difference between the functional obtained directly on the fine resolution time domain with full convergence of equations and the functional computed on the fine time domain using partially converged solutions projected onto the fine time domain from the coarse time domain. Since the solutions to both the forward and adjoint problems were obtained on the coarse time domain through partial convergence, we can estimate the error due to partial convergence exactly. The temporal resolution error between 16 and 32 steps is then obtained by subtracting the partial convergence error at 16 steps from the total error. The total error can be intepreted in the following manner for easy understanding. The solution at 16 steps needs to be corrected for partial convergence and then the error in projecting this corrected solution onto the fine time domain is the component of the total error that represents temporal resolution error. The results of this case are shown in Table 4. The partial convergence errors shown in Table 4 were computed using exactly the same numbers as those shown in Table 1 with the only difference being that the percentage error is computed against the exact functional at 32 steps rather than 16 steps.

Table 3. Validation of total error correction term

Description	Functional value	% Error vs. Target	% Predicted Error
Target functional - exact at 32 steps (fully converged)	-0.309957065250867	-	-
Partially converged flow and mesh at 16 steps (flow = 0.8e-4, mesh = 1e-3)	-0.289270768414725	+6.674	+6.562
Corrected for partial convergence and also resolution from 16 to 32 steps	-0.3096094927905224	+0.112	-

Table 4. Validation of individual error correction terms for total correction case

Description	% Predicted Error	% Exact Error
Predicted total error (partial convergence at 16 steps + resolution between 16 and 32 steps)	+6.562	+6.674
Predicted mesh partial convergence error at 16 steps (referenced against exact functional at 32 steps)	-0.0685	-0.0733
Predicted flow partial convergence error at 16 steps(referenced against exact functional at 32 steps)	-1.054	-1.053
Predicted resolution error between 16 and 32 steps	+7.684	+7.804

X. Results

If no information about the temporal error, either global or local were available, it would be intuitive to successively double the resolution in time uniformly until functional scalar values of interest converge. This is equivalent to performing a grid sensitivity study in the spatial domain where meshes are successively refined in a uniform manner until mesh insensitivity is observed. This is typically done beforehand on a sample problem that is representative of the real problem of interest. A reasonable resolution that provides the required accuracy in the solution is then chosen from the results of study. All other problems that are represented by the sample problem are then solved using the chosen time resolution. The goal of adapting the temporal resolution is to hone onto only the regions of the time domain that are most relevant to the functional scalar of interest and target them for refinement. The reasoning is that if such refinement were possible, then a solution of the required accuracy may be obtained with lesser cost. Once a reasonable temporal distribution has been determined, it may be applied to similar problems to produce equivalent results as uniform refinement but with significant cost savings. In order to be viable such a procedure must be able to outperform uniform refinement of the resolution.

As in the case of validating the correction terms individually where the sum of the errors in time of each component were validated, four different adaptation cases were run in order to test the validity of the error distributions of each component. The designed test cases were:

1. **Case A**: Adaptation of temporal resolution only with full convergence of flow and mesh motion equations.
2. **Case B**: Adaptation of convergence limits with a fixed number of uniformly spaced time steps.
3. **Case C**: Adaptation of both temporal resolution and convergence limits utilizing separate time-integrated averages of each error component.
4. **Case D**: Adaptation of both temporal resolution and convergence limits utilizing the combined time-integrated average of all error components.

For all four cases, the starting point consisted of 8 uniformly spaced time steps with a convergence limit of $5E^{-4}$ for the flow equations and $1E^{-3}$ for the mesh equations. The complete adaptive procedure can be broken down into the following steps:

1. Run flow solver with some initial uniform time step distribution and convergence limits to obtain an unsteady flow and mesh motion solution. This is the solution set U_H.
2. Run adjoint solver to obtain flow and mesh adjoint variables on the coarse time domain.
3. Interpolate flow solution, mesh coordinates, flow adjoint and mesh adjoint onto the fine time domain. The fine time domain is constructed by subdividing each time step into two.
4. Use interpolated variables to estimate the total error distribution.
5. Compute error distribution due to partial convergence of flow and mesh motion equations.
6. Separate temporal resolution error distrubution from the total error distribution by subtracting partial convergence error distribution.
7. Sum the total error distribution to obtain the correction term ε_{cc}.
8. Augment the scalar objective estimate $L_h(U_h^H)$ by adding the correction term ε_{cc} to get an estimate for $L_h(U_h)$
9. Compute time-integrated averages of individual error distributions (temporal resolution and partial convergence).
10. Identify regions where temporal resolution error is greater than time-integrated mean of error and refine by dividing those time steps into two. Similarly, identify regions of insufficient convergence and tighten convergence tolerance.
11. Keep repeating the procedure until the estimate $L_h(U_h)$ stops changing, or the correction term reaches some predetermined error tolerance.

A. Case A: Temporal Resolution Adaptation

Figures (4(a)) to (4(d)) show the results of adapting the temporal resolution. The starting resolution consists of 8 equally spaced time steps. The initial error in the functional value is roughly 30%. The convergence criterion for both uniform refinement and adaptive refinement was set at machine zero in order to remove the influence of partial convergence. During the initial stages of the adaptation, the uniform refinement of resolution performs similarly to adaptive refinement. This is primarily because of the total cost involved in adaptive refinement which includes the cost of not only the unsteady flow solution but also the cost of computing the adjoint, interpolating the solution and estimating the error. The number of time steps however remains lower than that of uniform refinement. It should be noted that at this point the difference in cost is irrelevant since the functional error remains above 10% for all cases. As the adaptation proceeds, the selective refinement of the temporal resolution begins to perform better than uniform refinement. It should be noted however, the inclusion of the computed correction always provides a better estimate of the functional than uniform refinement. By the time the error in the functional reaches 1%, the performance scaling between the adapted domain and the uniformly refined domain is roughly 1.5. When the functional value corrected by the computed correction is taken into account, the scaling is roughly 3.5.

B. Case B: Convergence Limits Adaptation

This case was designed to study the effect of convergence of the flow and mesh motion equations on the overall accuracy of the functional scalar. The test case consisted of 32 uniformly spaced time steps and starting convergence limits of $5E^{-4}$ for the flow equations and $1E^{-3}$ for the mesh motion equations. Figures (5(a)) and (5(b)) show the results from this case. The procedure that is equivalent to uniform refinement of the temporal resolution in this case is the uniform tightening of the convergence tolerance. In this particular case, the convergence limit was tightened by one order of magnitude per adaptive step. Such a rate allows for roughly ten adaptive cycles before uniform tightening reaches machine precision. From the error plot it is clear that the initial error by itself is fairly low at 0.2%. The adapted convergence curve in the absence of the computed correction term lies above the uniform refinement curve, indicating that the adaptation procedure does not recover the substantial additional cost of having to compute an adjoint solution. However, when the computed correction term is added to the objective value, the adaptive algorithm outperforms the uniform convergence tolerance approach. Note that in reality, the cost of the adjoint solution will be amortized over convergence error as well as temporal error estimation, and thus Figures (5(a)) and (5(b)) may present a pessimistic viewpoint. Additionally, although the convergence of the flow and mesh motion equations is relatively consistent as shown in Figures (3(a)) and (3(b)), more complicated cases often display convergence rates which degrade as the convergence tolerances are tightened, a factor which will further improve the competitiveness of this adaptive approach.

C. Case C: Combined Temporal Resolution and Convergence Limits Adaptation

The third case is that of adapting both the temporal resolution and the convergence limits. Within this case, comparisons were made against two separate reference curves. Both reference curves begin with a uniform resolution of 8 time steps and are uniformly refined thereon by doubling the resolution at each adaptation cycle. The difference between the two reference curves is that one begins with the flow and mesh convergence limits of $5E^{-4}$ and $1E^{-3}$ combined with uniform tightening at each adaptation cycle, while the other is set to machine precision at all adaptation cycles including the starting point. Figures (6(a)) through (6(d)) show the results for Case-C1 which is the comparison against the reference curve with uniform resolution refinement and uniform convergence tolerance tightening. Although the adapted curve without the correction costs about the same as uniform refinement, when combined with the correction the adaptation provides on average a cost saving factor of 3. Referring to Figures (7(a)) through (7(d)), where the comparison is made against the curve where uniform temporal refinement with convergence to machine precision is employed, the average cost saving is nearly an order of magnitude. Figures (8) and (9) show the distribution of time steps, the flow and mesh convergence limits, and the errors from various components. As the adaptive procedure advances, the error distribution of these components becomes approximately equidistributed over the time steps, as expected. On the other hand, the heterogeneity of the adapted convergence limits and time step values in the plots confirm that intuition would be of limited use in selecting these values, even for a simple unsteady problem such as this test case.

D. Case D: Overall Optimized Error Reduction and Control

In the previous example, although the temporal resolution and convergence tolerances were selected adaptively, these were selected individually without regards for the relative contribution to the total error due to the various components such as mesh motion convergence, flow convergence, and temporal resolution. In the final example, we formulate an adaptive criteria that considers the relative magnitude of these different components at each time step and chooses the most effective refinement strategy locally in order to reduce the total error. This is done by simply comparing each individual error contribution at each time step versus the average error contribution for all time steps from all sources. Thus, at any given location in time, the adaptation strategy may choose between tighter convergence tolerances on the mesh equations, the flow equations, or a reduction in the time step size, or any combination of these individual strategies. Furthermore, the computed error correction term is used to determine when the overall error has reached the desired tolerance.

Figure (10) illustrates the initial and final convergence tolerances and time step distributions for this case. The adaptive procedure was terminated after 5 cycles, when the computed error estimate of 0.3% was found to be lower than the original prescribed error tolerance of 0.5%. Figure (10(e)) indicates that the mesh convergence tolerance in the final solution remains relatively loose throughout most of the time domain, which is consistent with the previously reported fact that the errors due to the partial convergence of the mesh motion equations are substantially smaller than the other error sources in the computation. The principal effect of the adaptation is seen to be the reduction of the time step size throughout large parts of the domain, with an accompanying but less pronounced tightening of the tolerance of the flow equations at various locations in the temporal domain. Refering to Figure (11), we can infer that the complete adaptation procedure required a total of 1038 cpu seconds, which compares favorably with a total of 2132 cpu seconds to obtain a result of similar accuracy with a fixed time step size of $dt = 0.06136$ and convergence criteria of $1E^{-10}$ for the flow and mesh motion equations. Evidently, efficiency comparisons of this type depend strongly on the convergence criteria and time step sizes used for the uniform refinement case. However, a principal advantage of the adjoint-based error approach is that it provides an estimate of the remaining error in the solution as well as an indication of the various sources of the error, whereas in the uniform refinement case, the levels of convergence and time step sizes must simply be selected by experience.

XI. Conclusions

A method to determine temporal error in unsteady flows was developed and tested showing significant cost saving. The algorithm replaces the ambiguity and guess work typically involved in intuitive temporal adaptation with a rigorous mathematical consistent procedure. Additionally, the method also accounts for the various sources of error, such as those due to partial convergence of the governing equations for the mesh motion equations and the flow equations. Because global adjoint error estimation is relatively expensive, due to the need to compute an unsteady adjoint solution which must be integrated in time, rapid convegence of the adaptive procedure is required to make these techniques cost effective. Although at this preliminary stage we have used a first-order temporal discretization and simple refinement criteria, in future work second or higher-order temporal schemes will be investigated, as well as the use of more sophisticated adaptive criteria. Additionally, more agressive refinement schemes such as 4:1 time step refinements may prove to be more cost effective, although this may require more accurate solution interpolation techniques. On the other hand, the current approach provides not only an error estimate for the objective of interest, but also a breakdown of the important contributing sources of error. In principle, this technique can be extended to more complicated multiphysics simulations, where the overall error sources from various disciplines such as fluid flow, structural analysis, or conjuguate heat transfer can be assessed, with implications for how best to deploy computational resources for maximum error reduction impact.

Another area of interst concerns the combination of spatial and temporal adaptation for functional outputs. Considering the results demonstrated in this paper combined with existing work that has been done on spatial adaptation, the machinery now exists to adapt both the spatial and time domains independently. The obvious path from this point would be to address issues that arise from the coupling of spatial and temporal adaptation.

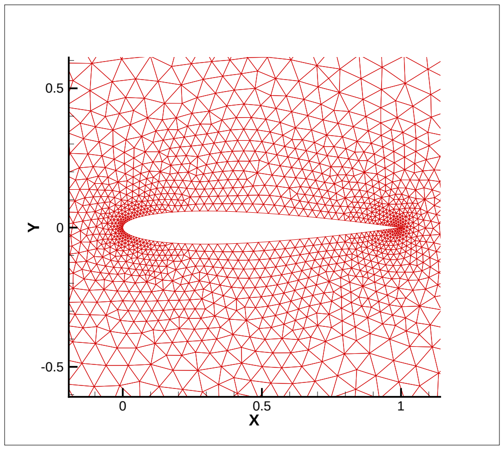
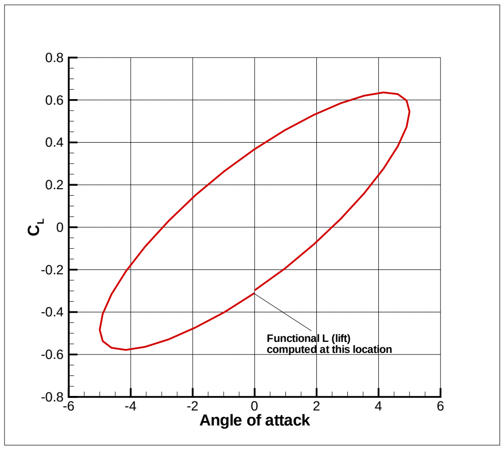
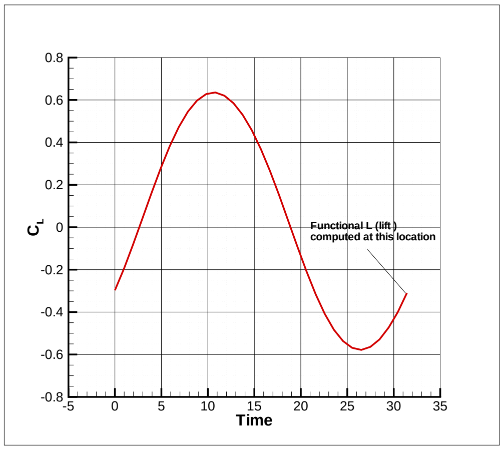


Figure 1. Unstructured mesh around NACA0012 airfoil used in computations

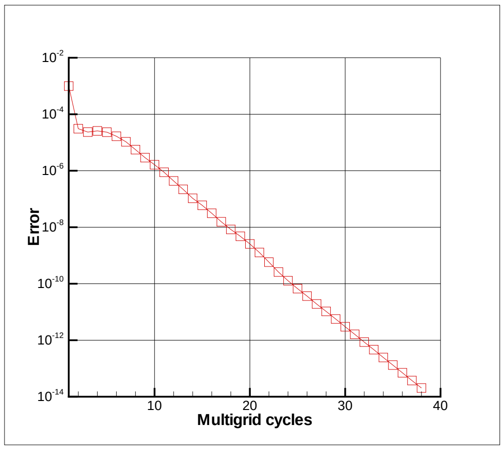


(a) Time variation of lift

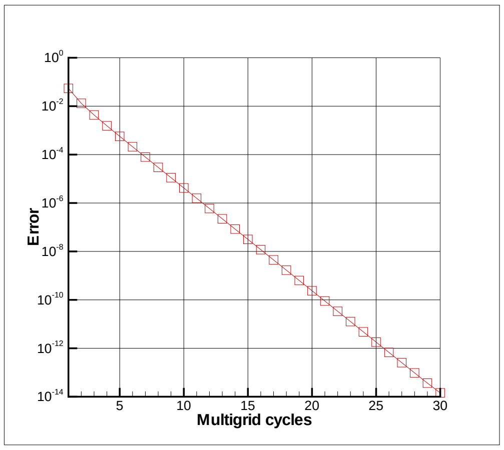


(b) Lift hysteresis curve for sinusoidally pitching airfoil

Figure 2. Lift variation for test case

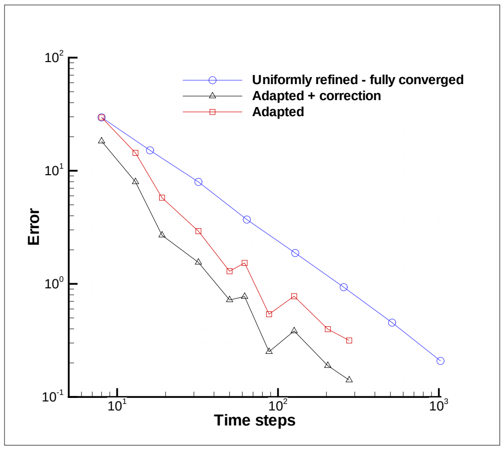


(a) Multigrid convergence of flow equations

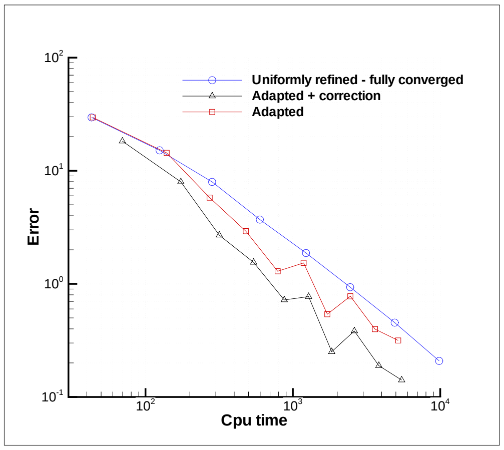


(b) Multigrid convergence of mesh equations

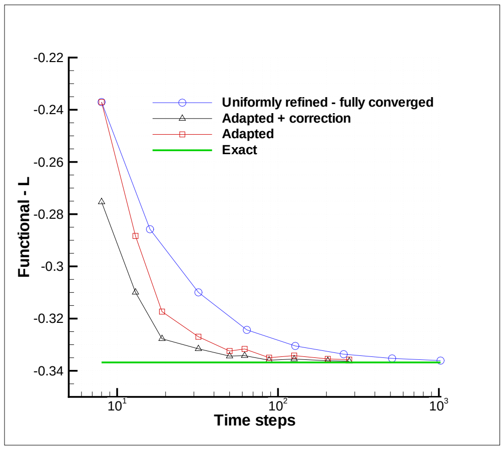
Figure 3. Typical convergence rates of governing equations



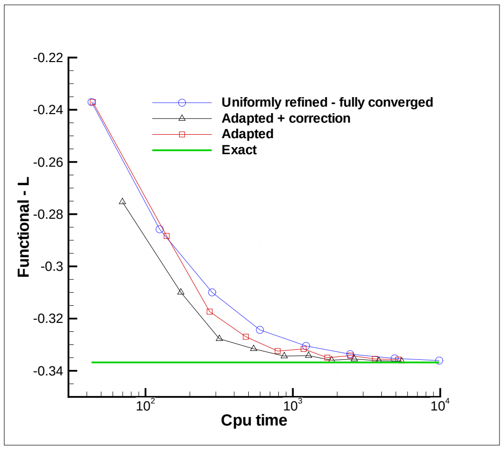
(a) Functional error reduction with number of time steps



(b) Cost of functional error reduction

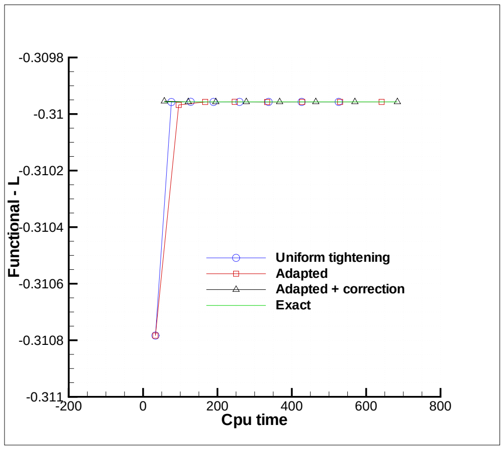


(c) Functional value convergence with number of time steps

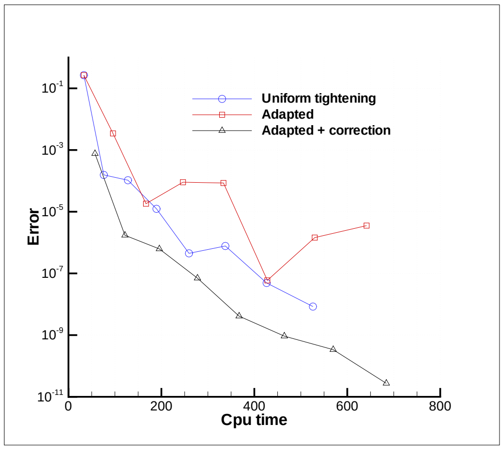


(d) Functional value convergence against cost

Figure 4. Case A: Adaptation of temporal resolution with full convergence of flow and mesh equations

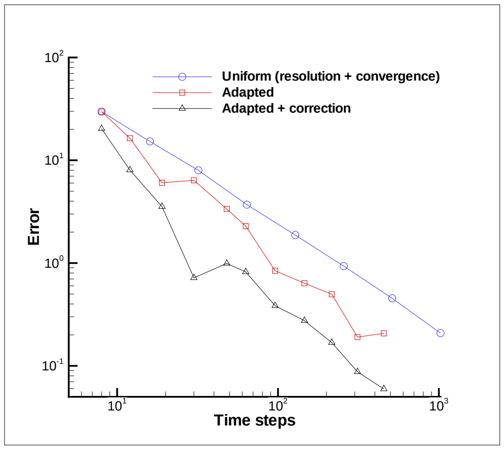


(a) Functional value convergence

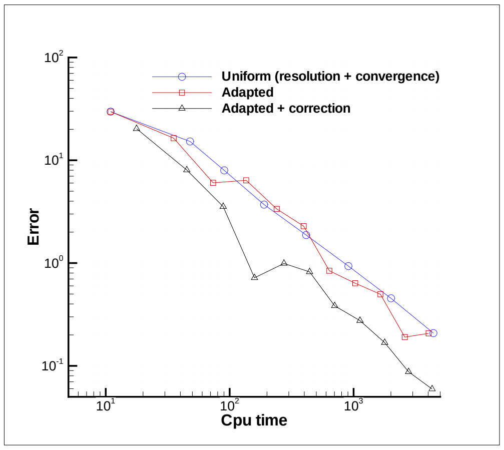


(b) Functional error reduction

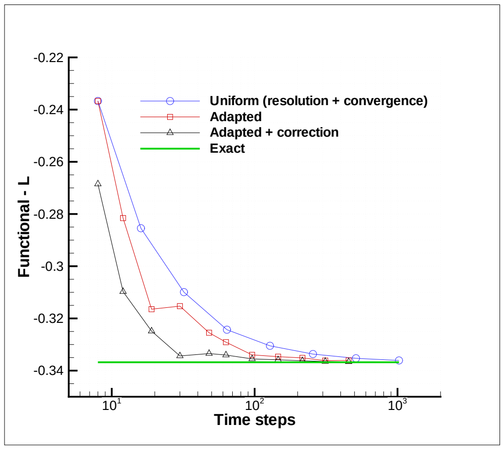
Figure 5. Case B: Adaptation of flow and mesh convergence limits with 32 fixed uniform time steps



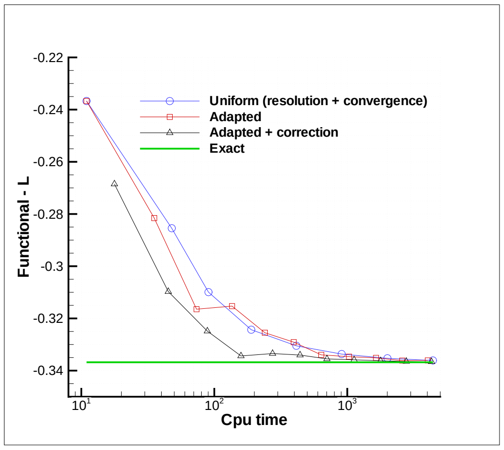
(a) Functional error reduction with number time of steps



(b) Cost of functional error reduction

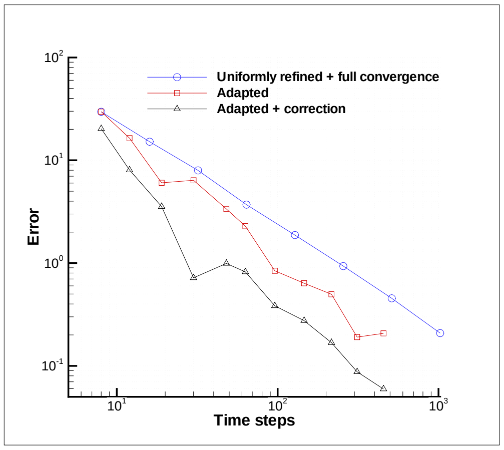


(c) Functional value convergence with number of time steps

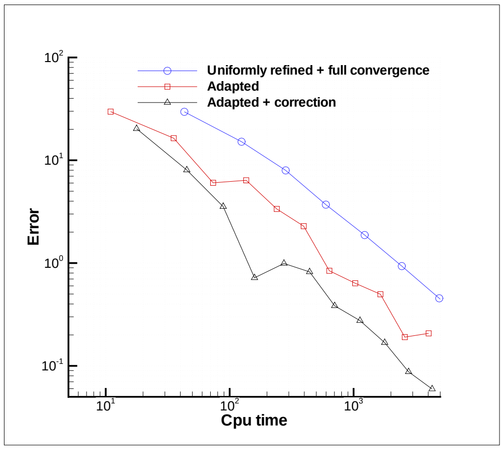


(d) Functional value convergence against cost

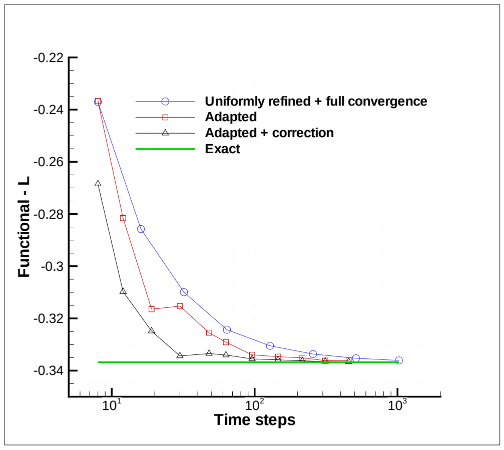
Figure 6. Case C1: Combined adaptation of temporal resolution and convergence limits compared against uniform refinement of temporal resolution and uniform tightening of convergence limits



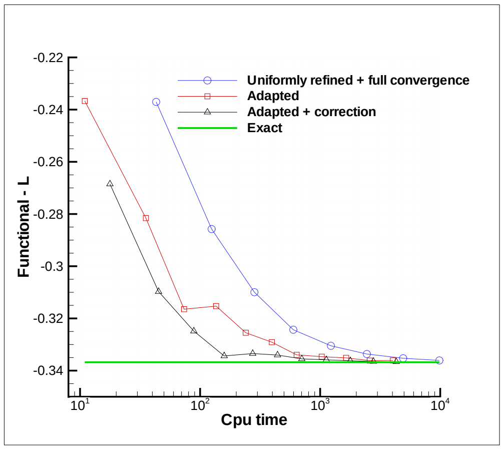
(a) Functional error reduction with number of time steps



(b) Cost of functional error reduction

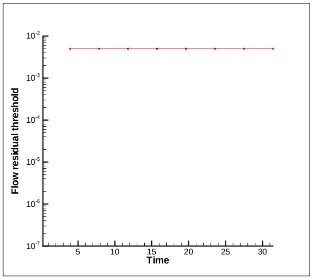


(c) Functional value convergence with number of time steps

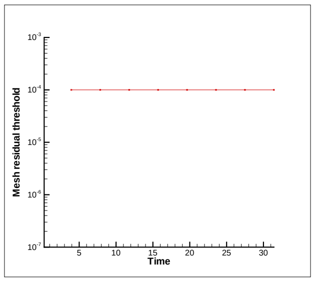


(d) Functional value convergence against cost

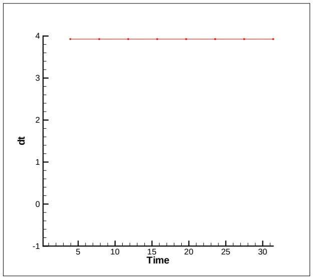
Figure 7. Case C2: Combined adaptation of temporal resolution and convergence limits compared against uniform refinement of temporal resolution and fixed full convergence of flow and mesh equations



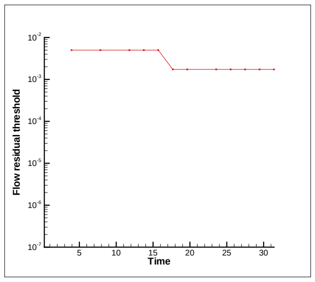
(a) flow limits at adaptation cycle = 0



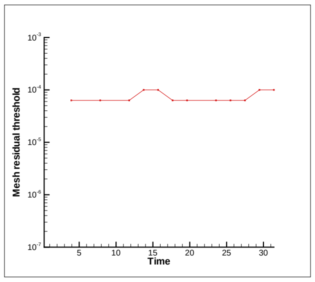
(b) mesh limits at adaptation cycle = 0



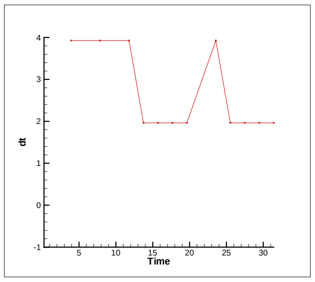
(c) dt distribution at adaptation cycle = 0



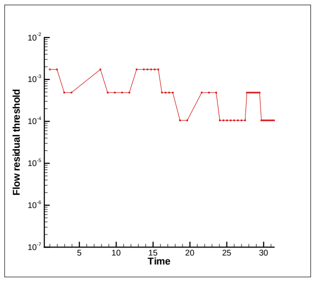
(d) flow limits at adaptation cycle = 2



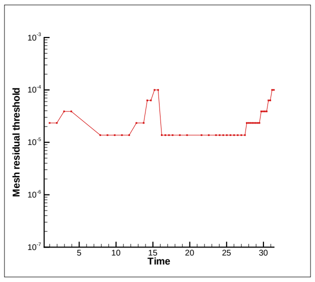
(e) mesh limits at adaptation cycle = 2



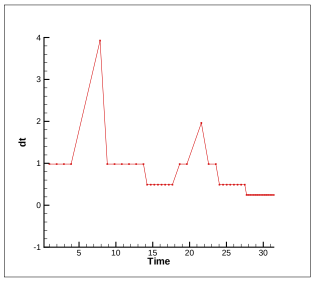
(f) dt distribution at adaptation cycle = 2



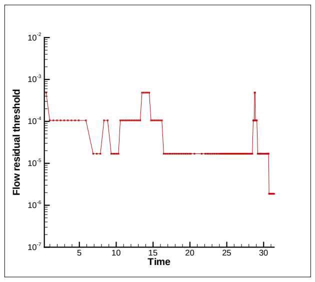
(g) flow limits at adaptation cycle = 5



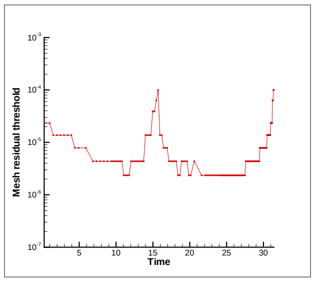
(h) mesh limits at adaptation cycle = 5



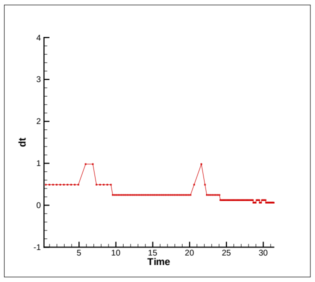
(i) dt distribution at adaptation cycle = 5



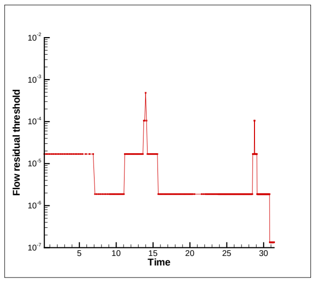
(j) flow limits at adaptation cycle = 8



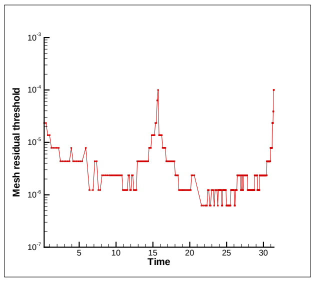
(k) mesh limits at adaptation cycle = 8



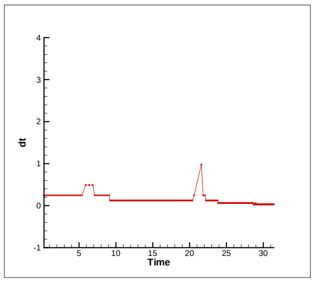
(l) dt distribution at adaptation cycle = 8



(m) flow limits at adaptation cycle = 10

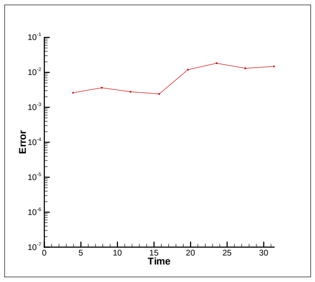


(n) mesh limits at adaptation cycle = 10

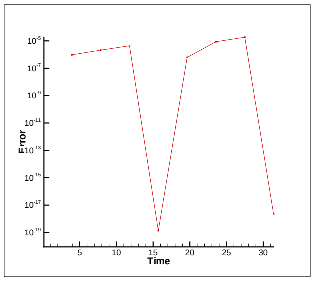


(o) dt distribution at adaptation cycle = 10

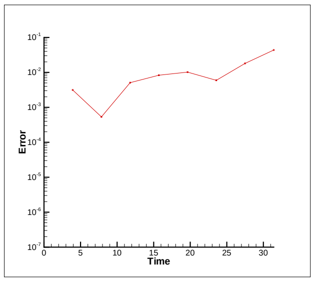
Figure 8. Time step and convergence limit distribution for Case C1



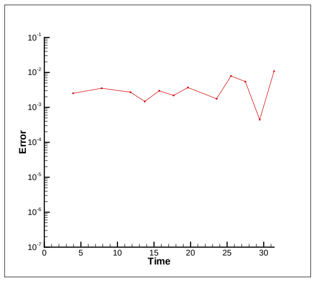
(a) flow conv. error at adaptation cycle = 0



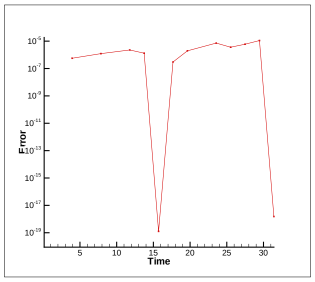
(b) mesh conv. error at adaptation cycle = 0



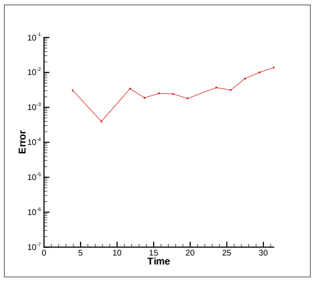
(c) resolution error at adaptation cycle=0



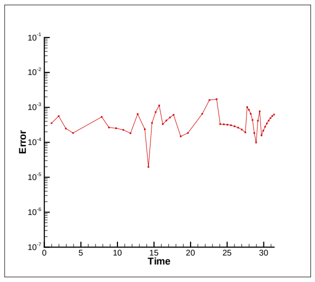
(d) flow conv. error at adaptation cycle = 2



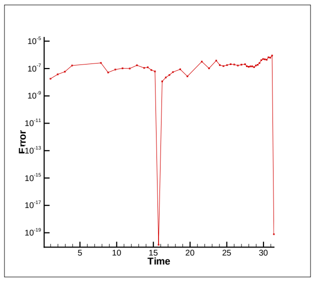
(e) mesh conv. error at adaptation cycle = 2



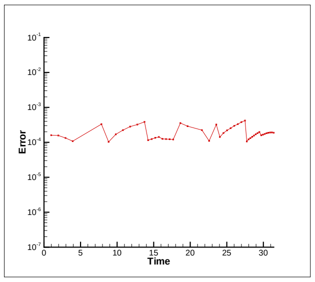
(f) resolution error at adaptation cycle=2



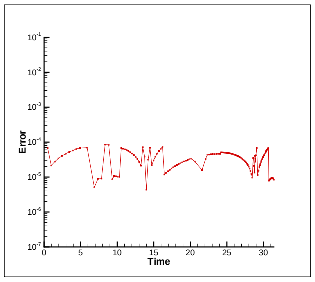
(g) flow conv. error at adaptation cycle = 5



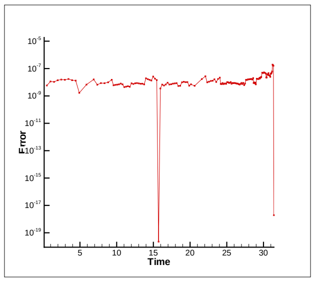
(h) mesh conv. error at adaptation cycle = 5



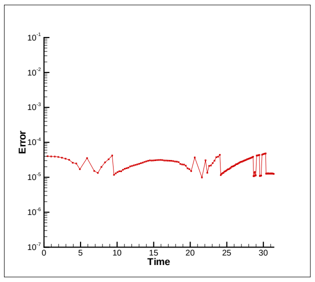
(i) resolution error at adaptation cycle=5



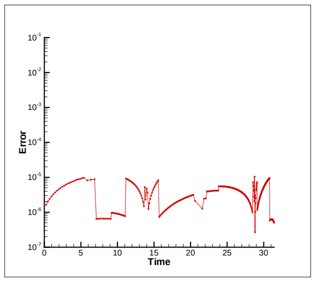
(j) flow conv. error at adaptation cycle = 8



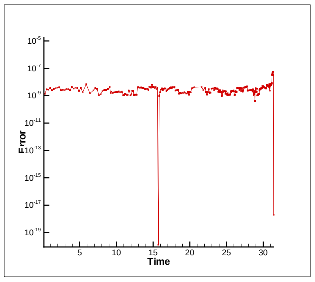
(k) mesh conv. error at adaptation cycle = 8



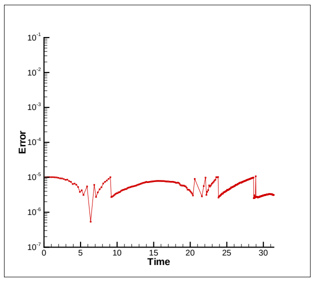
(l) resolution error at adaptation cycle=8



(m) flow conv. error at adaptation cycle = 10

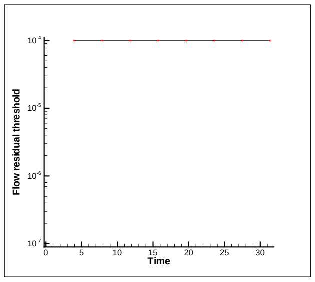


(n) mesh conv. error at adaptation cycle = 10

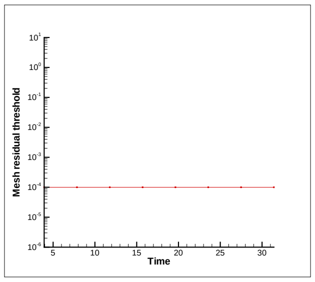


(o) resolution error at adaptation cycle=10

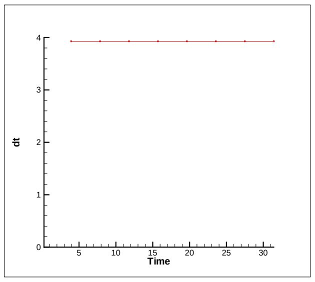
Figure 9. Resolution and partial convergence error distributions for Case C1



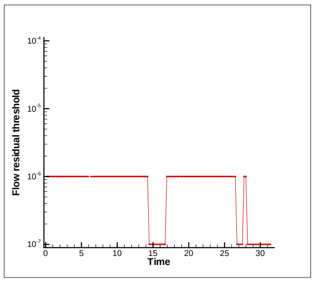
(a) flow limits at adaptation cycle = 0



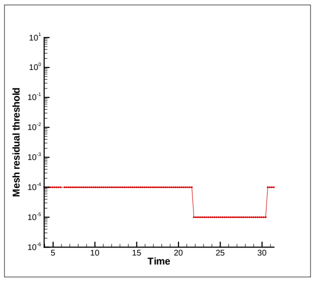
(b) mesh limits at adaptation cycle = 0



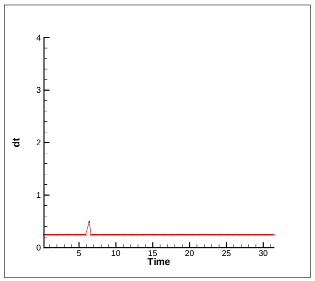
(c) dt distribution at adaptation cycle = 0



(d) flow limits at adaptation cycle = 5



(e) mesh limits at adaptation cycle = 5



(f) dt distribution at adaptation cycle = 5

Figure 10. Time step and convergence limit distribution for Case D

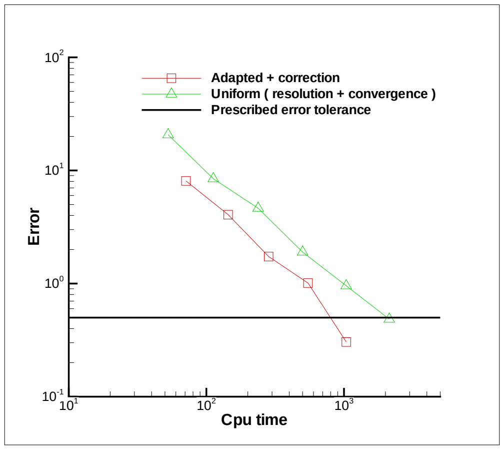


Figure 11. Comparison of efficiency in error reduction for Case D

References

[1]Carpenter, M. H., Viken, S., and Nielsen, E., "The Efficiency of High-Order Temporal Schemes," AIAA Paper 2003-0086.

[2]Johnson, C., "Error Estimates and Adaptive Time Step Control for a Class of One-Step Methods for Stiff Ordinary Differential Equations," *SIAM Journal of Numerical Analysis*, Vol. 25, 1988, pp. 908–926.

[3]Vendetti, D. and Darmofal, D., "Grid Adaptation for Functional Outputs: Application to Two-Dimensional Inviscid Flows," *Journal of Computational Physics*, Vol. 176, 2002, pp. 40–69.

[4]Vendetti, D. and Darmofal, D., "Anisotropic grid adaptation for functional outputs: application to two-dimensional viscous flows," *Journal of Computational Physics*, Vol. 187, 2003, pp. 22–46.

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

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

[7]Butcher, J. C., *Numerical methods for ordinary differential equations*, Wiley, Chicester, UK, 2003.

[8]Lambert, J. D., *Numerical methods for ordinary differential systems*, Wiley, Chicester, UK, 1991.

[9]Hindmarsh, A. C. and Taylor, A. G., "PVODE and KINSOL: Parallel software for differential and nonlinear systems," UCRL–ID–129739, Lawrence Livermore National Laboratory.

[10]Estep, D., "A Posteriori Error Bounds and Global Error Control for Approximation of Ordinary Differential Equations," *SIAM Journal of Numerical Analysis*, Vol. 32, 1995, pp. 1–48.

[11]Cao, Y. and Petzold, L., "A Posteriori Error Estimation and Global Error Control for Ordinary Differential Equations by the Adjoint Method," *SIAM J. Scientific Computing*, Vol. 26, No. 2, 2004, pp. 359–374.

[12]Li, S. and Petzold, L., "Adjoint Sensitivity Analysis for Time-Dependent Partial Differential Equations with Adaptive Mesh Refinement," *Journal of Computational Physics*, Vol. 198, No. 1, 2004, pp. 310–325.

[13]Mani, K. and Mavriplis, D. J., "An Unsteady Discrete Adjoint Formulation for Two-Dimensional Flow Problems with Deforming Meshes," *To be presented at the 45th Aerospace Sciences Meeting and Exhibit, Reno NV*, 2007, AIAA Paper 2007–0060.

[14]Farhat, C., Geuzaine, P., and Crandmont, C., "The Discrete Geometric Conservation Law and the Nonlinear Stability of ALE Schemes for the Solution of Flow Problems on Moving Grids," *Journal of Computational Physics*, Vol. 174, 2001, pp. 669–694.

[15]Mavriplis, D. and Yang, Z., "Construction of the discrete geometric conservation law for high-order time-accurate simulations on dynamic meshes," *Journal of Computational Physics*, Vol. 213, 2006, pp. 557–573.

[16]Mavriplis, D., "Mulgtigrid Techniques for Unstructured Meshes," *Notes prepared for 26th Computational Fluid Dynamics Lecture Series Program of the von Karman Institute of Fluid Dynamics, Rhode St Genese, Belgium*, 1995.