Higher-order Time Integration Schemes for Aeroelastic Applications on Unstructured Meshes

Zhi Yang [*], Dimitri J. Mavriplis [†]

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

Efficient techniques for computational aeroelasticity (CA) on unstructured meshes are investigated. These include the formulation of a fourth-order implicit Runge-Kutta time-integration scheme (IRK64) and a third-order backwards difference time-integration scheme (BDF3) as well as a second-order backwards difference time-integration scheme (BDF2) for the coupled system of flow and structural equations. Strong coupling between the flow solution and structural equations is maintained by fully converging the fully coupled system at each time step or stage. The time integration schemes are constructed such that discrete conservation is maintained in the flow solution in the presence of dynamically deforming meshes by verifying the Discrete Geometric Conservation (DGCL) law. Efficient multigrid solution techniques are devised for solving both the governing flow equations, and the mesh motion equations, using the same agglomerated coarse levels for both problems. The flutter boundary prediction of the AGARD 445.6 wing is used to demonstrate the accuracy and efficiency of these techniques as well as the sensitivity of flutter boundary prediction to time step size and temporal errors.

I. Introduction

Aeroelasticity involves the interaction between the fluid flow and the flexible structure of an aerospace vehicle. Aeroelastic problems include panel flutter, vertical tail buffet, fluid-blade interaction in rotorcraft and active wings. Aeroelasticity plays an important role in the design of modern aircraft due to the requirement of being able to predict performance degradation or even structural failure resulting from strong fluid-structure interaction. Computational aeroelasticity, which combines computational fluid dynamics (CFD) and computational structural dynamics (CSD), provides an efficient method for predicting and understanding aeroelastic phenomena.

Unstructured mesh approaches have become well established for steady-state flow simulations due to the flexibility they afford for dealing with complex geometries. For unsteady flows with moving boundaries, such as aeroelastic problems, implicit time-integration strategies are required for the efficient solution of the flow equations, while at the same time robust mesh deformation techniques are necessary for maintaining a suitable discretization of the evolving computational domain. In order to develop an efficient unsteady flow simulation capability, both mesh deformation and flow solution aspects must be considered, as well as the interaction between these two phenomena. In previous work,[10] we investigated the use of fast multigrid solvers for high-order implicit time-integration of unsteady flows on static meshes. In reference,[32] we extended our investigation to the simultaneous development of robust and efficient unstructured mesh deformation techniques, as well as efficient higher-order time-integration strategies, and investigated the requirements for maintaining high time-accuracy in the presence of deforming meshes.

For unsteady flow simulations, computational time remains an important issue. When small temporal errors are desired, higher-order time-integration (higher than second-order) has been shown to be more efficient than low-order time integration. Bijl et. al.[5] investigated and compared higher-order implicit Runge-Kutta schemes and Backward Differencing schemes on structured grids, while Jothiprasad, Mavriplis and Caughey[10] showed how a fourth order Runge-Kutta scheme (IRK64) outperforms second-order Backward Differencing (BDF2) on unstructured grids using a multigrid algorithm for solving the implicit system arising

[*]Postdoctoral research associate, AIAA member; email: zyang@uwyo.edu
[†]Professor, AIAA Associate Fellow; email: mavriplis@uwyo.edu

at each time step. In reference,[32] a third-order backwards difference scheme (BDF3) was also shown to be competitive for time-dependent dynamic mesh problems. In this work, we investigate the application of BDF2, BDF3, and IRK64 schemes for computational aeroelasticity.

When dynamic meshes are used, the mesh velocities and other parameters related to geometry need to be considered carefully so that the errors introduced by the deformation of the mesh do not degrade the formal accuracy of the flow simulation. The discrete geometric conservation law (DGCL) provides a guideline on how to evaluate these quantities. First-order and second-order time-accurate and geometrically conservative schemes were presented and discussed in,[15,13] respectively. Guillard and Farhat have proved that to obtain at least first-order time accuracy, a DGCL condition must be satisfied.[8] For higher-order time integration schemes such as BDF3 and IRK64, a DGCL strategy which preserves the design order of the scheme must be explicitly constructed, as shown in references.[32,24,25,30]

In addition to preserving time-accuracy, efficient solution strategies must be employed to avoid excessive computational times for long-time integration problems. Non-linear and linear multigrid methods have been investigated previously for steady and unsteady flow simulations using unstructured meshes.[17,19,21] In references[32,24,25] the use of a linear unstructured agglomeration multigrid method was shown to provide the most efficient strategy for integrating the non-linear flow equations in time.

Several mesh-deformation strategies, such as the tension spring analogy,[3,31] the torsion spring analogy[6,26] and the linear elasticity analogy,[2,27] have been successfully demonstrated in the literature. In previous work[32] we have investigated various methods for computing deformed mesh configurations, and found the linear elasticity approach to be the most robust approach, particularly when a variable modulus of elasticity is incorporated. However, for simple or smooth surface deformations, the basic spring analogy mesh motion technique is often sufficient to obtain smooth and valid deformed mesh configurations. In both cases, the resulting mesh motion equations can be solved very efficiently using the same agglomeration multigrid approach used to solve the flow equations.[32]

In the following sections, we first outline the governing equations and the base flow solver. We then discuss our choice and implementation of high-order time integration schemes including the fourth-order implicit Runge-Kutta scheme (IRK64), as well as the second and third-order backwards difference schemes (BDF2 and BDF3). This is followed by a presentation of the geometric conservation law (GCL), which must be verified by all of these schemes. The spring analogy mesh deformation strategy is then briefly described. The computational structural dynamic (CSD) equations are presented next, followed by a description of the algorithm employed for solving the fully coupled CFD and CSD problem at each time step or stage. Finally, a brief description of the agglomeration multigrid solver is given, as well as the application of this method to solve the fully coupled aeroelastic problem, including relative timings for the various components of the solver. These techniques are then used to compute the dynamic response and flutter boundary of the AGARD 445.6 in order to illustrate the performance of these methods and the sensitivity of flutter boundary prediction methods to numerical temporal errors.

II. Flow solver

A. Governing Equations in Arbitrary-Lagrangian-Eulerian (ALE) Form and Base Flow Solver

The Navier-Stokes equations in conservative form can be written as:

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

where $\mathbf{U}$ represents the vector of conserved quantities (mass, momentum, and energy), $\mathbf{F}(\mathbf{U})$ represents the convective fluxes and $\mathbf{G}(\mathbf{U})$ represents the viscous fluxes. Integrating over a (moving) control volume $\Omega(t)$, we obtain:

$$\int_{\Omega(t)} \frac{\partial \mathbf{U}}{\partial t} dV + \int_{\partial\Omega(t)} (\mathbf{F}(\mathbf{U}) \cdot \vec{\mathbf{n}}) dS + \int_{\partial\Omega(t)} (\mathbf{G}(\mathbf{U}) \cdot \vec{\mathbf{n}}) dS = 0 \tag{2}$$

Using the differential identity

$$\frac{\partial}{\partial t} \int_{\Omega(t)} \mathbf{U} dV = \int_{\Omega(t)} \frac{\partial \mathbf{U}}{\partial t} dV + \int_{\partial\Omega(t)} \mathbf{U}(\dot{\mathbf{x}} \cdot \vec{\mathbf{n}}) dS \tag{3}$$

where $\dot{\mathbf{x}}$ and $\vec{\mathbf{n}}$ are the velocity and normal of the interface $\partial\Omega(t)$, respectively, equation (2) becomes:

$$\frac{\partial}{\partial t}\int_{\Omega(t)} \mathbf{U} dV + \int_{\partial\Omega(t)} (\mathbf{F}(\mathbf{U}) - \mathbf{U}\dot{\mathbf{x}}) \cdot \vec{\mathbf{n}} dS + \int_{\partial\Omega(t)} \mathbf{G}(\mathbf{U}) \cdot \vec{\mathbf{n}} dS = 0 \tag{4}$$

Considering $\mathbf{U}$ as cell averaged quantities, these equations are discretized in space as:

$$\frac{\partial}{\partial t}(V\mathbf{U}) + \mathbf{R}(\mathbf{U}, \dot{\mathbf{x}}(\mathbf{t}), \vec{\mathbf{n}}(\mathbf{t})) + \mathbf{S}(\mathbf{U}, \vec{\mathbf{n}}(\mathbf{t})) = \mathbf{0} \tag{5}$$

where $\mathbf{R}(\mathbf{U}, \dot{\mathbf{x}}, \vec{\mathbf{n}}) = \int_{\partial\Omega(t)} (\mathbf{F}(\mathbf{U}) - \dot{\mathbf{x}}\mathbf{U}) \cdot \vec{\mathbf{n}} dS$ represents the discrete convective fluxes in ALE form, $S(\mathbf{U}, \vec{\mathbf{n}})$ represents the discrete viscous fluxes, and V denotes the control volume. In the discrete form, $\dot{\mathbf{x}}(\mathbf{t})$ and $\vec{\mathbf{n}}(\mathbf{t})$ now represent the time varying velocities and surface normals of the control-volume boundary faces.

The Navier-Stokes equations are discretized by a central difference finite-volume scheme with additional matrix-based artificial dissipation on hybrid meshes which may include triangular and quadrilateral elements in two dimensions, or tetrahedra, pyramids, prisms and hexahedra in three dimensions. Second-order accuracy is achieved using a two-pass construction of the artificial dissipation operator, which corresponds to an undivided biharmonic operator. A single unifying edge-based data-structure is used in the flow solver for all types of elements. For multigrid calculations, a first-order accurate discretization is employed for the convective terms on coarse grid levels.[23, 22]

B. Higher-order Time Integration and the Discrete Geometric Conservation Law (GCL)

For unsteady flow simulations, a fully implicit time-integration strategy is most often adopted, using either multistep Backward Difference Formulae (BDF) or multistage Implicit Runge-Kutta (IRK) schemes. Although first-order (BDF1) and second-order (BDF2) backwards difference schemes are A-stable, higher-order multistep BDF schemes (beyond second-order) are not A-stable. However, the unstable region of the BDF3 scheme is very small and this scheme has often been used successfully for unsteady flow simulations.[32] Multistage implicit Runge-Kutta schemes (IRK) of high-order which are A-stable and L-stable can easily be constructed. However, multiple nonlinear problems need to be solved at each time step using IRK schemes, which makes these more expensive alternatives to the BDF schemes.[5] The application of higher-order IRK schemes for dynamic unstructured mesh problems has been investigated in references.[24, 25, 30] In the current paper, our discussion will be limited to the two backwards difference schemes BDF2 and BDF3, and a six stage, fourth-order accurate IRK scheme (IRK64).

Equation (5) can be discretized in time using the general formula for a k-step backward difference scheme as:[5, 10] (dropping the viscous flux terms for simplicity)

$$\alpha_1(V\mathbf{U})^{n+1} + \sum_{i=0}^{1-k} \alpha_i(V\mathbf{U})^{n+i} = \Delta t\mathbf{R}((V\mathbf{U})^{n+1}, t^{n+1}) \tag{6}$$

For the BDF2 scheme, k=2 and the coefficients are given as: $\alpha_1 = \frac{3}{2}$, $\alpha_0 = -2$,$\alpha_{-1} = \frac{1}{2}$, while for the BDF3 scheme, k=3 and the coefficients take on the values: $\alpha_1 = \frac{11}{6}$, $\alpha_0 = -3$, $\alpha_{-1} = \frac{3}{2}$, $\alpha_{-2} = -\frac{1}{3}$. By defining a nonlinear residual[10]

$$\Re^{n+1}(\mathbf{U}) \equiv \Re(\mathbf{U}^{n+1}) \equiv \alpha_1(V\mathbf{U})^{n+1} - \sum_{i=0}^{k-1} \alpha_i(V\mathbf{U})^{n-i} - \Delta t\mathbf{R}((V\mathbf{U})^{n+1}, t^{n+1}) \tag{7}$$

the solution of equation (6) can be obtained by solving the non-linear problem $\Re^{n+1}(\mathbf{U}) = 0$ at each time step.

A fourth-order time-accurate implicit Runge-Kutta scheme (IRK64) which obeys the discrete geometric conservation law is also employed in this paper. The general form for an s-stage implicit Runge-Kutta scheme applied to equation (5) is given as:

$$(V\mathbf{U})^k = (V\mathbf{U})^n + \Delta t\sum_{j=1}^{s} a_{kj}\mathbf{Q}((V\mathbf{U})^j, t_n + c_j\Delta t)), \qquad k = 1, ..., s \tag{8}$$

$$(V\mathbf{U})^{n+1} = (V\mathbf{U})^n + \Delta t\sum_{j=1}^{s} b_j\mathbf{Q}((V\mathbf{U})^j, t_n + c_j\Delta t)) \tag{9}$$

where the n and $n+1$ superscripts on the dependent variable refer to the values at the beginning and end of the current time step, and the k and j superscripts refer to stage values computed by the Runge-Kutta scheme within the current time-step. In this paper, we will concentrate on the Explicit first-stage, Single diagonal coefficient, Diagonally Implicit class of RK schemes (ESDIRK). The coefficients for the considered schemes can be represented by the Butcher tableau:

$c_1 = 0$	0	0	0	0	0	0
c_2	a_{21}	a_{66}	0	0	0	0
c_3	a_{31}	a_{32}	a_{66}	0	0	0
c_4	a_{41}	a_{42}	a_{43}	a_{66}	0	0
c_5	a_{51}	a_{52}	a_{53}	a_{54}	a_{66}	0
$c_6 = 1$	b_1	b_2	b_3	b_4	b_5	a_{66}
$\mathbf{w}^{n+1}$	b_1	b_2	b_3	b_4	b_5	a_{66}

Table 1. Butcher Tableau for the ESDIRK class of RK schemes with number of stages, $s = 6$.

where the numerical values of the coefficients for the scheme used in this work are given in references.[5,10] These schemes are characterized by a lower triangular form of the coefficient table, thus resulting in a single implicit solve at each individual stage. The $c_k = \sum_{j=1}^{k} a_{kj}, k = 1, ...s$ coefficients denote the corresponding point in time for each individual stage. The first stage is explicit ($a_{1j} = 0$), and the last stage coefficients are such that $a_{kj} = b_j, j = 1, .., s$, which omits the requirement of evaluating equation (9), since the value of $(V\mathbf{U})^{n+1}$ at the new time step is given by the last stage value $(V\mathbf{U})^{k=s}$.

To apply the multistep BDF schemes or the multistage IRK scheme in the presence of dynamic unstructured meshes, the so called geometric conservation law (GCL) should be satisfied to avoid degrading the formal accuracy of the scheme, and to ensure non-linear stability. The original statement of the geometric conservation law was introduced by Thomas and Lombard[29] for structured meshes. The discrete geometric conservation law requires that the state $\mathbf{U} = constant$ be an exact solution of equation (5). In this case, we have $S(\mathbf{U}, \vec{\mathbf{n}}) = \mathbf{0}$, since the viscous fluxes are based on gradients of $\mathbf{U}$. Additionally, we have:

$$\int_{\partial\Omega(t)} (\mathbf{F}(\mathbf{U}) - \mathbf{U}\dot{\mathbf{x}}) \cdot \vec{\mathbf{n}} dS = \mathbf{R}(\mathbf{U}, \dot{\mathbf{x}}, \vec{\mathbf{n}}) = -\mathbf{U}\bar{\mathbf{R}}(\dot{\mathbf{x}}, \vec{\mathbf{n}}) \tag{10}$$

since the integral of the convective fluxes $\mathbf{F}(\mathbf{U})$ around a closed control volume must be zero for constant $\mathbf{U}$, for any spatially conservative scheme, with $\bar{\mathbf{R}}$ referring to the discretization of the second term in the above boundary integral. Thus, the GCL can be stated, in semi-discrete form as:

$$\frac{\partial V}{\partial t} - \bar{\mathbf{R}}(\dot{\mathbf{x}}(\mathbf{t}), \vec{\mathbf{n}}(\mathbf{t})) = 0 \tag{11}$$

The Geometric Conservation Law must be satisfied in the discrete form, or so called Discrete Geometric Conservation Law. Lesoinne and Farhat[15] presented a first-order time-accurate backwards difference scheme (BDF1) which obeys the discrete geometric conservative law, while Koobus and Farhat[12] derived a second-order accurate backwards-difference scheme (BDF2) which obeys the discrete conservation law. In order to construct the DGCL for the BDF3 scheme, equation (6) is rewritten as

$$\sum_{i=1}^{-2} \alpha_i (V\mathbf{U})^{n+i} = \Delta t \mathbf{R}((V\mathbf{U})^{n+1}, t^{n+1}, \dot{\mathbf{x}}(\mathbf{t}), \vec{\mathbf{n}}(\mathbf{t})) \tag{12}$$

where, $\dot{x}(t)$ and $n\vec{(t)}$ represent the velocity and the normal vector of a control volume boundary face, which are evaluated in such a manner that the above equation is satisfied exactly in the presence of uniform flow, as determined by the DGCL. A full derivation of the DGCL for BDF3 can be found in reference.[32]

Similarly to the derivation for BDF schemes, a nonlinear residual for each stage of the IRK scheme is defined as

$$\Re^{n+1}(\mathbf{U}) \equiv \Re(\mathbf{U}^{n+1}) \equiv (V\mathbf{U})^k - (V\mathbf{U})^n + \Delta t \sum_{j=1}^{k} \alpha_{kj} \mathbf{R}(\mathbf{U}^k, \mathbf{x}^k, \dot{\mathbf{X}}^k) \quad k = 1, ..., s \tag{13}$$

with the corresponding values: $\mathbf{x}^k = \mathbf{x}(t + c_k\Delta t)$, and $\dot{\mathbf{X}}^{\mathbf{k}} = \dot{\mathbf{x}}(t + c_k\Delta t) \cdot \vec{\mathbf{n}}(t + c_k\Delta t)$, with the values of the c_k being given by the RK scheme. To satisfy the DGCL at each stage, the $\dot{\mathbf{X}}^k$ can be computed by solving the system:

$$\begin{pmatrix} 1 & 0 & 0 & 0 \\ \alpha_{21} & \alpha_{22} & 0 & 0 \\ \alpha_{31} & \alpha_{32} & \alpha_{33} & 0 \\ \alpha_{41} & \alpha_{42} & \alpha_{43} & \alpha_{44} \end{pmatrix} \begin{Bmatrix} \dot{\mathbf{X}}^1 \\ \dot{\mathbf{X}}^2 \\ \dot{\mathbf{X}}^3 \\ \dot{\mathbf{X}}^4 \end{Bmatrix} = \frac{1}{\Delta t} \begin{Bmatrix} \Delta t \dot{\mathbf{X}}^{(1)} \\ V^2 - V^n \\ V^3 - V^n \\ V^4 - V^n \end{Bmatrix} \tag{14}$$

where the explicit first stage $\alpha_{11} = 0$ (c.f. Table 1) corresponds to equating $\dot{\mathbf{X}}^1$ with the value determined at the end of the previous time step. Because the IRK64 scheme is an SDIRK scheme (single coefficient diagonally implicit RK), the matrix of a_{kj} coefficients is of lower triangular form, and equation (14) is easily solved by forward substitution. A full derivation of the DGCL for IRK schemes can be found in references.[24,25,30]

At each time step or RK stage, equations (7) or (13) are solved using the agglomeration multigrid technique described subsequently.

III. Mesh Moving Strategies

The mesh deformation strategy used in this work is based on the spring analogy approach, which has been used extensively in the past for aeroelastic calculations (see for example reference[3]). Each edge of the mesh is represented by a spring whose stiffness is related to the length of the edge. The governing equations are closely related to a simple Laplace equation, as the displacements in each coordinate direction become decoupled and are governed by the equations:

$$(\Delta x_i)_m = \frac{\sum_j k_{ij}((x_j)_m^{n+1} - (x_j)_m^n)}{\sum_j k_{ij}}, \quad m = 1, 2, 3 \tag{15}$$

where $k_{ij} = \frac{1}{l_{ij}^p}$ and $l_{ij} = \sum_{m=1}^{3}((x_i)_m - (x_j)_m)^2)^{\frac{1}{2}}$. The parameter p usually is set to 2 as recommended in reference.[31] Although these equations are relatively simple to solve, this approach tends to produce deformed meshes with collapsed or negative cell volumes especially for large deformations about complex geometries. However, for the types of deformations considered in this work, the spring analogy was found to be sufficient for generating smooth and valid meshes at all time steps. In previous work,[32] we have advocated the use of linear elasticity methods for more demanding mesh motion cases, and plan to incorporate these techniques into the aeroelasticity framework in the near future.

IV. Structure Equation

The governing equation for the structure of a typical wing under aerodynamic loading can be written as

$$[M]\{\ddot{w}\} + [C]\{\dot{w}\} + [K]\{w\} = \{F(t)\} \tag{16}$$

where $[M]$, $[C]$ and $[K]$ represent the mass matrix, damping matrix and stiffness matrix, respectively. $\{F(t)\}$ is the aerodynamic loading and w is the displacement. Using modal analysis, the dependent variables are expanded in terms of the natural free vibration modes as

$$\{w(\mathbf{x}, \mathbf{t})\} = \sum_{\mathbf{i=1}}^{\mathbf{N}} \mathbf{q_i}(\mathbf{t})\{\phi_{\mathbf{i}}(\mathbf{x})\} \tag{17}$$

where $q_i(t)$ and $\{\phi_i(\mathbf{x})\}$ are generalized coordinates and comparison functions satisfying the free vibration modes, respectively. The comparison functions are obtained by solving the eigenproblem of the free vibration problem, using only the first N modes of the expansion. Substituting equation (17) into equation (16) and then multiplying by ϕ^T, we obtain[1,11]

$$\ddot{\mathbf{q}} + [\xi]\dot{\mathbf{q}} + [\omega]\mathbf{q} = \mathbf{Q} \tag{18}$$

where

$$[\xi] = \phi^T[C]\phi \qquad [\omega] = \phi^T[K]\phi \qquad Q = \phi^T\{F\}$$

Because $[\omega]$ and $[\xi]$ are diagonal matrices and their diagonal terms are ω_i and $2\xi_i\omega_i$, respectively, the coupled system of equations (18) can be rewritten as a set of uncoupled scalar equations:

$$\ddot{q}_i + 2\xi_i\omega_i\dot{q}_i + \omega_i^2 q = Q_i, \qquad i = 1, 2, \cdots, N \tag{19}$$

where ξ_i and ω_i are the modal damping parameter and the natural frequency for the ith mode, respectively. To convert equation (19) to a first order system, let

$$y_{1i} = q_i, \qquad y_{2i} = \dot{q}_i$$

then, equation (19) can be rewritten as

$$\dot{Y}_i = A_i Y_i + B_i \tag{20}$$

where

$$Y_i = \left\{ \begin{array}{c} y_{1i} \\ y_{2i} \end{array} \right\}, \qquad A_i = \left[\begin{array}{cc} 0 & 1 \\ -\omega_i^2 & -2\xi_i\omega_i \end{array} \right], \qquad B_i = \left\{ \begin{array}{c} 0 \\ Q_i \end{array} \right\} \tag{21}$$

The implicit time integration schemes (BDF or IRK) which are used in solving the flow equations are used to solve equation (20) as well to obtain q and the displacement w. The general form for a backwards difference scheme(BDF) for equation (20) can be written as

$$\alpha_1 {Y_i}^{n+1} + \sum_{i=0}^{1-k} \alpha_i {Y_i}^{n+i} = \Delta t(A_i {Y_i}^{n+1} + {B_i}^{n+1}) \tag{22}$$

while implicit Runge-Kutta schemes (IRK) for equation (20) can be written as

$$\begin{aligned} {Y_i}^k &= {Y_i}^n + \Delta t \sum_{j=1}^{s} a_{kj}(A_i {Y_i}^j + B_i({Y_i}^j, t_n + c_j\Delta t)), \qquad k = 1, ..., s & (23) \\ {Y_i}^{n+1} &= {Y_i}^n + \Delta t \sum_{j=1}^{s} b_j(A_i {Y_i}^j + B_i({Y_i}^j, t_n + c_j\Delta t)) & (24) \end{aligned}$$

V. Coupling Algorithm for CFD and CSD, Method of Solution

Computational aeroelasticity is a coupled fluid-structure problem which requires the exchange of forces and displacements between the flow and the structure, as well as the simultaneous solution of the CFD and CSD equations. In the general case, the boundary surfaces of the mesh used in the flow computation are different from those used in the structure computation. Many researchers have presented interface algorithms which ensure energy conserving force transfers and smooth displacement transfers between CFD and CSD surface meshes. Sadeghi et al[28] provide a good comparison of the infinite-plate spline interpolation(IPS)[9] method, the constant-volume tetrahedron method and the inverse boundary element method. The IPS method is based on the equilibrium equation of an infinite thin plate. According to Harder et al,[9] the displacement of a plate surface can be written as

$$w(x, y) = a_0 + a_1 x + a_2 y + \sum_{i=1}^{N} F_i r_i^2 ln(r_i^2) \tag{25}$$

where $r_i^2 = (x - x_i)^2 + (y - y_i)^2$ and the coefficients a_0, a_1, a_2 and F_i are obtained by solving

$$\sum_{i=1}^{N} F_i = 0, \quad \sum_{i=1}^{N} x_i F_i = 0, \quad \sum_{i=1}^{N} y_i F_i = 0, \quad w_i = a_0 + a_1 x_i + a_2 y_i + \sum_{j=1}^{N} F_i r_{ij}^2 ln(r_{ij}^2) \tag{26}$$

where $r_{ij}^2 = (x_i - x_j)^2 + (y_i - y_j)^2$, which corresponds to matching the displacements on the CSD and CFD meshes at the $i = 1, ..., N$ prescribed points. The advantage of IPS is that the connections between the N prescribed points are not requied.

A conservative force and energy transfer method has also been given by Farhat[7] and is illustrated in Figure 1(a). The vertices of the CFD mesh are first projected to the corresponding enclosing element of the structural mesh. The natural coordinates (ξ, η) of this projection point in each enclosing element are computed and the displacements originally defined at the vertices of the CSD mesh are transferred to the the CFD mesh vertices as:

$$w_{aj} = \sum_{i=1}^{i_e} N_i(\xi_j, \eta_j) w_{si} \tag{27}$$

where w_{aj} and w_{si} are displacements of the CFD and CSD mesh vertices, respectively, and $N_i(\xi_j, \eta_j)$ are the CSD mesh element shape functions.

It should be noted that the infinite-plate spline interpolation(IPS) is a two-dimensional method. However, the test case used in this work is the AGARD 445.6 wing which is a thin wing. The interface of the CFD mesh and the CSD mesh can be approximated by a plane.[28] Both methods are used in this paper, and no effect on the final results could be observed between the two methods.

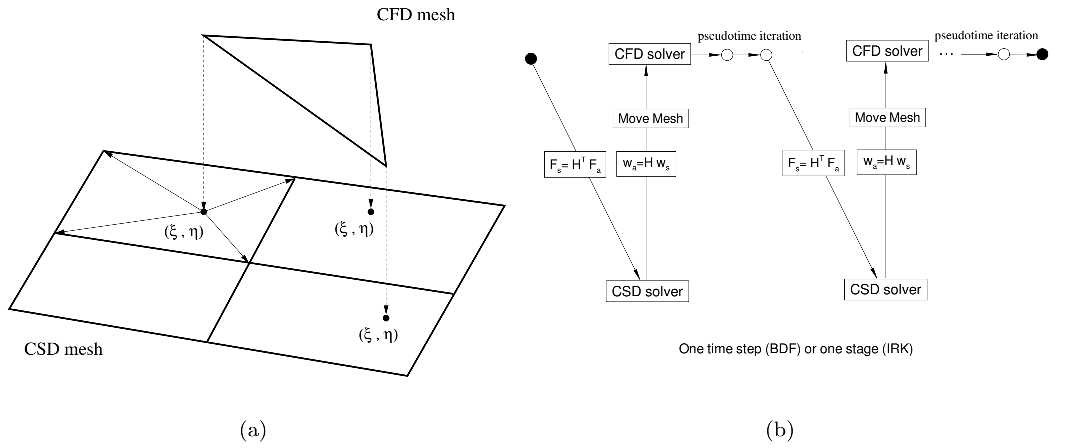


Figure 1. (a) Illustration of mesh displacement transfer algorith proposed in reference.[7] (b) Illustration of strong flow-structural equation coupling algorithm used at each time step.

At each physical time step, or each stage of the IRK scheme, the combined system of fluid and structural equations must be solved in a fully coupled manner. This requires the use of a a strongly coupled implicit method,[1, 16] and is necessary to eliminate any time lag between the flow solution and the structure solution at each physical time step. Figure 1(b) depicts the strategy for simultaneously converging the fully coupled system. At each physical time step of a BDF scheme or at each stage of the IRK scheme, the following steps are taken:

1. Calculate the aerodynamic loading and transfer the loading to the structure mesh;
2. Solve the structural equations (c.f. equations (22) or equations (23) and (24));
3. Transfer displacements to the interface of the CFD mesh;
4. Solve the mesh motion equations (15);
5. Solve the flow equations in pseudo-time;
6. If appreciable flow field changes occur, repeat step $1 \sim 5$ until fully converged.

The entire procedure operates in a parallel computing environment, except for step 2, where the modal structural solver is executed on the master node of the parallel computer, since this step requires negligible computational time compared to the other tasks.

The transfer of displacements from the CSD mesh to the CFD mesh in step 3 proceeds as determined by equations (25) or (27), and may be written as:

$$w_a = H w_s \tag{28}$$

where w_a is the interface displacement of the CFD mesh, and w_s is the interface displacement of the CSD mesh, and H is the transfer operator, which is determined by equations (25) or (27).

The transfer of the aerodynamic loading in step 1 can then be written as:

$$F_s = H^T F_a \tag{29}$$

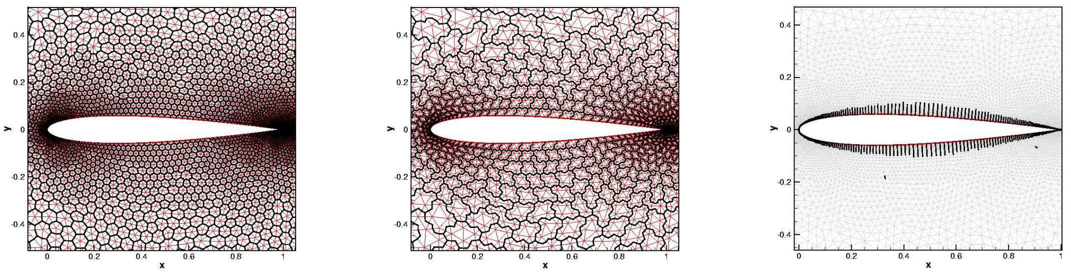
where F_s represents the CSD interface forces, F_a represents the CFD interface forces, and where the force transfer operator H^T is taken as the transpose of the operator used for displacement transfers. This approach has been shown to result in a transfer mechanism which is energy conserving.[7]

In the fully coupled solution algorithm, overall convergence depends on the number of outer iterations, as well as the number of inner iterations for the flow solver and mesh motion equations in pseudo-time. The inner pseudo-time solution is performed using the agglomeration multigrid algorithm described in the next section.

VI. Acceleration Strategies and Computational Cost

Agglomeration multigrid methods are used to accelerate the solution of the non-linear problem arising at each time step of the unsteady flow equations, as well as for solving the mesh motion equations. Agglomeration multigrid was originally developed as a steady-state solver for the Euler and Navier-Stokes equations on unstructured grids.[17, 22, 19, 21] The idea of multigrid is to accelerate the solution on a fine grid by iteratively computing corrections to the fine grid problem on coarser grid levels where the cost of the iterations is lower, and the global error components are more easily reduced. Figures 2a-b show an example of the agglomeration multigrid procedure. Figure 2a shows the fine grid, while Figure 2b shows the 2nd level coarse grid where each coarse cell is the combination of several fine cells.

For highly stretched meshes, the effectiveness of the multigrid approach degrades due to the anisotropic stiffness induced by the grid stretching. To relax this stiffness, an implicit line solution technique was introduced in references.[20, 18] The lines are constructed along the strong coupling direction in the mesh, as shown by the example depicted in Figure 2c for an unstructured mesh about a NACA0012 airfoil. In these regions, a block tridiagonal algorithm is used to solve all quantities along each line implicitly, thus replacing the simple explicit approach on all grid levels. The same line structures and coarse agglomerated levels are used for solving both the mesh motion equations, and the implicit time-dependent flow equations. However, for the non-linear flow equations, either a non-linear multigrid (NMG) approach may be employed, or a linear multigrid method (LMG) may be used to drive an outer approximate Newton iteration. Figure 3 illustrates the convergence history obtained at a particular BDF2 time-step of the AGARD 445.6 wing test case described below, using the non-linear and linear multigrid approaches respectively for a Mach number of $M = 1.141$, both in terms of non-linear iterations, and in terms of cpu time. For the linear multigrid case, three linear multigrid cycles are used for each outer non-linear Newton update, while a single non-linear multigrid cycle constitutes a non-linear iteration in the case of the NMG solver. This difference accounts for the faster convergence of the LMG solver in terms of iterations. The linear multigrid approach is also found to be substantially more economical in terms of cpu time than the non-linear multigrid solver, requiring about one third of the overall cpu time of the non-linear multigrid method to achieve the same level of convergence. This is largely due to the lower cpu time required by the linear multigrid cycles, as documented in reference.[32] Similar results are observed for the convergence of the implicit system of flow equations which occur at each stage of the IRK scheme. Therefore, the linear multigrid method is used exclusively in the computations described in the remainder of this paper. Figure 3c shows the convergence



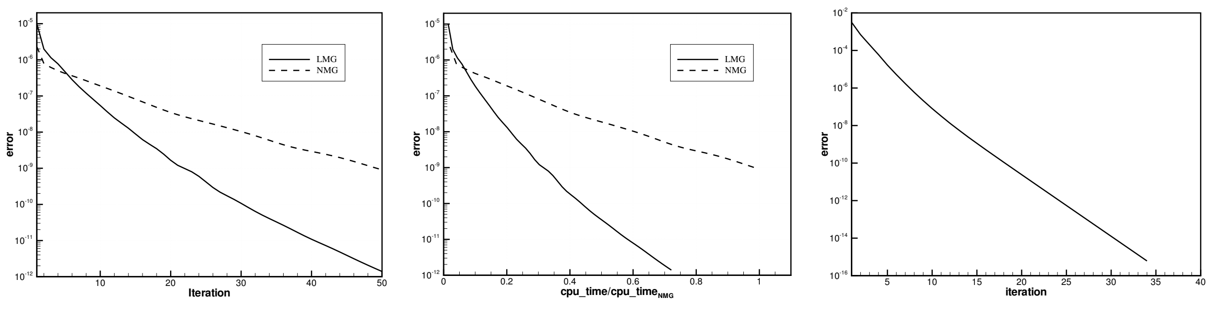
(a) 1st level (b) 2nd level (c) implicit lines

Figure 2. Illustration of agglomeration multigrid and implicit line construction.

history of the mesh motion solver for the same test case at the same time step. These equations can be converged to machine precision in less than 40 multigrid cycles.

For aeroelastic computations, full convergence of the flow equations and mesh motion equations at each time step is not required. In fact, the optimal level of convergence in pseudo-time for dual-iteration strategy implicit time-integration schemes remains an area of research. For the computational aeroelastic problem, the important convergence criterion is that based on the fully coupled system of flow and structural equations. In the present work, the fully coupled system is partially converged by using three outer CFD/CSD coupling iterations, as depicted in Figure 1(b), with each coupling iteration performing 10 non-linear iterations on the flow equations, while the mesh motion equations are converged to 10^{-10} at each coupling iteration, requiring a variable number (but less than 20) of multigrid cycles. This strategy results in a minimum of two-orders of magnitude reduction in the fully coupled system residuals.

Table 2 shows the relative cpu time of the flow solver and the mesh motion solver for BDF2 and IRK64 within a single coupled CFD/CSD time step. All values are normalized by the cpu time of the BDF2 flow solver. The cpu time of the mesh motion solver varies from 24% to 36% of the cpu time of the corresponding flow solver. The cpu time of IRK64 for one time step is about 4.7 times of the cpu time of BDF2, since the IRK64 scheme requires the solution of 5 implicit systems (one for each stage except for the first stage). Note that the mesh motion equations are close to fully converged at each coupling iteration in order to avoid any possible errors from these terms, and the time required by the mesh motion solver could likely be reduced without any adverse effect on the overall solution by easing this convergence tolerance. Additionally, the cpu time of the CSD solver is negligible because the modal analysis used to solve the CSD equations requires a trivial amount of cpu time.



(a) LMG vs. NMG (b) Relative CPU time (c) Mesh motion

Figure 3. Convergence history of mesh motion, LMG and NMG.

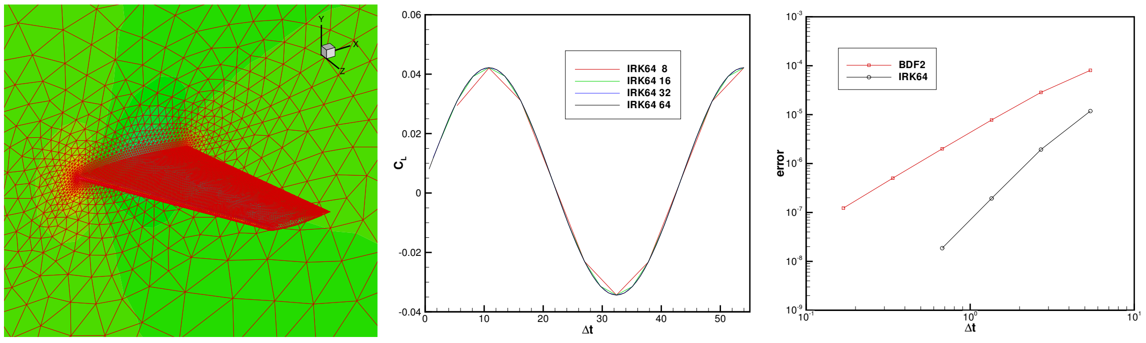
	Flow solver	Mesh Motion	Total
BDF2	1.00	0.358	1.358
IRK64	5.16	1.243	6.400

Table 2. Relative CPU time for flow solver and mesh motion.

VII. Results and Discussion

A. Time-Accuracy Validation for Dynamic Unstructured Mesh Problem

The time accuracy of the BDF2 and IRK64 implicit time-integration schemes is examined for a simplified test case involving prescribed motion. A three-dimensional inviscid flow test case is constructed by forced twisting of an ONERA M6 wing, at a Mach number of 0.755, and an incidence of 0.016 degrees. A periodic pitching motion is prescribed at the quarter chord point of the wing tip, with a reduced frequency of 0.1628 and an amplitude of 2.51 degrees. The wing root is held fixed, and a linear variation of the pitching motion is prescribed between the wing tip and root, thus resulting in a twisting motion. A fully tetrahedral mesh, shown in Figure 4a is used for this case, containing 53961 vertices. Mesh deformation is computed at each



(a) Mesh (b) Lift coefficient (c)

Figure 4. Flow solution and measured temporal error as a function of time step size for BDF2 and IRK64 schemes for three-dimensional twisting M6 wing .

time-step or stage by solving the spring-analogy equations subject to a fixed outer-boundary condition. The computed lift coefficient versus time is shown in Figure 4b for the BDF2 and the IRK64 scheme, using 8, 16, 32, or 64 time steps per period. Good temporal accuracy for the IRK64 scheme is observed using as few as 8 time steps per period. Figure 4c illustrates the computed temporal error as a function of the time-step for both schemes, at the time t=54, using a highly resolved IRK64 solution with a time step of 0.3375 (128 time steps per period) as the reference solution, where the error is computed as the RMS difference of all flow variables between the computed solution and the reference solution at the final time. The slope of the error curve achieved for the BDF scheme is 2, which matches the design accuracy of the scheme, while the IRK64 scheme results in an error curve slope of 3.3, which is slightly lower than the design value of 4, for this fourth-order accurate scheme.

B. Flutter Boundary Prediction for AGARD 445.6 Wing

The AGARD 445.6 wing test case,[33] which was tested in the Transonic Dynamics Tunnel at NASA Langley Research Center, has been widely used as a benchmark for evaluating aeroelastic simulations.[4, 14, 16] The wing is a semispan model that has a quarter-chord sweep angle of 45°, a panel aspect ratio of 1.65, a taper ratio of 0.66 and NACA 65A004 airfoil cross section. The wing flutter speed index is defined as

$$V_f = \left(\frac{U_\infty}{b\omega_\alpha \sqrt{\mu}} \right)_{neutral} \tag{30}$$

where, U_∞ is the velocity of freestream, b is the half chord at wing root, ω_α is the natural circular frequency of the first uncoupled torsion mode and $\mu = m/(\rho v)$ is the wing mass ratio. m is the mass of the wing, v is the volume related to the wing, and ρ is density of the fluid. To find a flutter point at one freestream Mach number M_∞, the Mach number and corresponding freestream velocity U_∞ are kept fixed. The fluid density ρ and corresponding dynamic pressure $\rho U_\infty^2/2$ are adjusted and the wing transients are calculated. If the transients of the generalized coordinates do not grow or decay, a neutral point or a flutter boundary point is obtained.

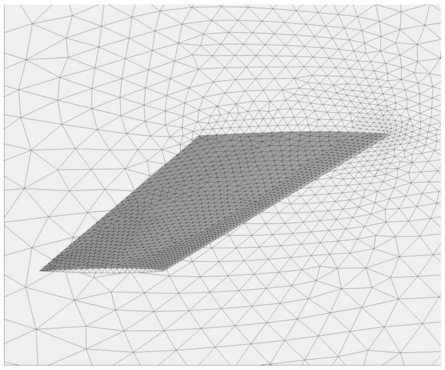


Figure 5. Meshes for AGARD wing 445.6.

The IRK64 and BDF3 schemes are used to investigate the effect of higher order time-integration on the wing flutter boundary predictions. BDF2 is also used as a baseline reference scheme. The CFD mesh used in this paper is shown in Figure 5. The mesh consists of 40,460 nodes and 224,531 tetrahedral elements. Inviscid flow simulation is performed on this mesh, and the first four modes are used in the modal analysis of the structural computations. These modes correspond to the first four modes of the weakened model

3,[33] which are identified as the first bending, first torsional, second bending and second torsional modes, as shown in Figure 6.

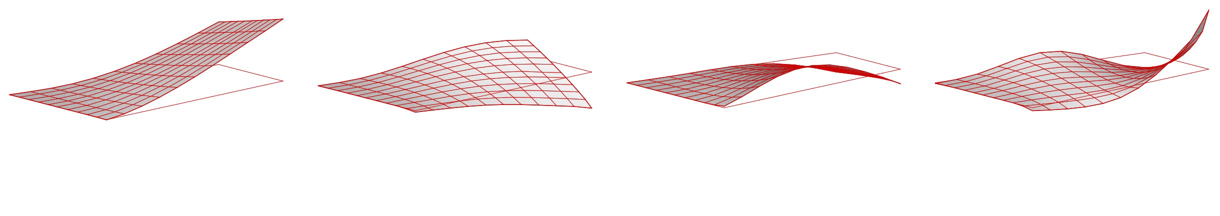
In order to demonstrate the difference between the low order scheme and the higher order schemes, the time history of the wing displacements computed using BDF2, BDF3 and IRK64 at different Mach numbers are examined in detail below.

Figure 7 through 10 depict the first, second, third and fourth modal dynamic responses at a speed index of 0.4459 and a Mach number of $M = 0.499$ for BDF2, BDF3 and IRK64. The time step varies from $\Delta t = 0.0375$ to $\Delta t = 1.8$, with the largest time step corresponding to approximately $\frac{1}{8}$ of the second mode period. The IRK scheme uses the largest time steps ($\Delta t = 1.8$ and $\Delta t = 0.6$), while the lower order schemes use a range of smaller time steps. All four mode displacements show a decaying response. The details of the response shown in Figures (Figure 7b, 8b, 9b and 10b) clearly show the BDF2 and BDF3 results converging to the IRK64 results as the time step size is decreased. For a given time step size, the BDF3 results are closer to the IRK64 results than those of the BDF2 scheme, except for the fourth mode at $\Delta t = 0.6$. The growing response for BDF3 at $\Delta t = 0.6$ (Figure 10a) shows the BDF3 scheme is unstable at this larger time step. Note that the response of the higher modes is more sensitive to the time step size for the lower order BDF schemes, while the fourth-order IRK scheme displays relatively little sensitivity to the time step size, even for the higher modes.

Figure 11a and 11b provide a comparison of the temporal error versus the time step size and the relative cpu time for the various time integration schemes. The error is computed based on the first mode response, using results computed with a smaller time step ($\Delta t = 0.3$) on the IRK64 scheme as the reference "exact" solution. The BDF2 scheme delivers second-order accuracy based on the slope of the error curve, while BDF3 does not recover its design accuracy, although it delivers higher absolute accuracy at any given time-step compared to the lower order scheme. The IRK64 scheme is more than third-order accurate, but falls somewhat short of the fourth-order design accuracy. Nevertheless, this scheme delivers the highest absolute accuracy for any given time step. Figure 11b shows IRK64 to be up to 5 times faster in terms of cpu time compared to the BDF2 scheme for small error tolerances of the order of 5×10^{-7}.

Figures 12 and 13 show the first and the third modal dynamic responses at a speed index of 0.49 and a Mach number of $M = 0.499$ for BDF2, BDF3 and IRK64. The time steps varies from $\Delta t = 0.075$ to $\Delta t = 1.8$. The first mode is the dominating mode and the first mode response is growing from the beginning. The oscillating amplitude of the third mode decreases before t = 50 and then increases, which indicates that energy is transfered from other modes. The growing of both the first and the third mode displacements show an unstable response. The details in the zoomed in Figures (Figure 12b, 13b) show that the BDF2 and BDF3 schemes converge to the IRK64 results as the time step is decreased. At the same Δt, the results of the BDF3 scheme are closer to the IRK64 results than those of BDF2. The error curves of Figures 14a and 14b, obtained in the same manner as those described previously, show second order accuracy for the BDF2 scheme, and better than third order accuracy for the IRK64 scheme, while the BDF3 scheme is seen to lag its design accuracy. Again, the IRK64 scheme is seen to be up to 5 times faster than the BDF2 scheme for small error tolerances.

Figure 15 and 16 show the first and the third modal dynamic response at a speed index of 0.37 and a Mach number of $M = 0.901$ for BDF2, BDF3 and IRK64. The time step varies from $\Delta t = 0.125$ to $\Delta t = 1.5$, which is about $\frac{1}{8}$ of the second mode period. The first mode shows a neutral response. As previously, the BDF2 and BDF3 results converge to the IRK64 results as the time step is reduced, and the IRK64 results remain the most efficient particularly for high temporal accuracy cases, as seen from Figure 17b.



(a) Mode 1, f= 9.599Hz (b) Mode 2, f=38.165Hz (c) Mode 3, f=48.348Hz (d) Mode 4, f=91.545Hz

Figure 6. Modal deflection for AGARD wing 445.6, weakened model 3.

Figure 18 and 19 show the first and the third modal dynamic response at a speed index of 0.442 and a Mach number of $M = 1.141$ for BDF2, BDF3 and IRK64. The time step varies from $\Delta t = 0.03125$ to $\Delta t = 1.5$, which is about $\frac{1}{8}$ of the second mode period. Both the first and the third mode responses are decaying. The blow up in Figures 18b and 19b show that the BDF2 and BDF3 schemes converge to the IRK64 results as the time step decreases, but Figure 19 shows the BDF3 scheme going unstable for larger time step values. The superior efficiency of the IRK64 scheme is again documented in the accuracy curves of Figure 20.

Figure 21 shows the comparison of the flutter boundary computed by the BDF and IRK schemes as well as the corresponding experimental values. Good agreement with experimental data is obtained in the subsonic region for the IRK and BDF schemes. In most cases, the results of these schemes are indistinguishable in predicting the flutter boundary, using the smallest time step for each scheme. However, the IRK64 scheme has the potential for significantly reducing the computational time required to predict the flutter boundary for a given accuracy level through the use of larger time steps. For the supersonic region, all methods show substantial underestimation of the flutter boundary, which is consistent with previously reported comptuational results.[14,16] The mesh used in this study is relatively coarse, and the resulting spatial erorr, as well as the use of an inviscid flow model may be responsible for these discrepancies. However, the inviscid coarse mesh simulations are well suited for studying the temporal error of the various time integration schemes in detail, since the effects of temporal and spatial error are treated separately in our approach.

VIII. Concluding Remarks

The principal elements required for the efficient solution of time dependent aeroelasticity simulations have been developed and validated. These elements include the formulation of the DGCL of high-order BDF and IRK time-integration schemes, the development of a robust mesh motion solver and a fast linear multigrid solver for the flow equations. These have been applied to a standard aeroelastic test case in order to validate their accuracy and efficiency. The time histories of the wing deflection computed using the BDF2, BDF3 and IRK64 schemes have been compared and the IRK64 scheme shows superior overall efficiency. The responses of the higher modes are most sensitive to the temporal error, and thus to time step sizes and the order of the time integration scheme. While the BDF3 scheme offers high temporal accuracy at modest computational cost, the tendency of this scheme to go unstable for large time step sizes may limit its use, particularly when higher structural modes are included. Although one may suspect the absence of unconditional stability as the cause of the BDF3 instabilities, the precise cause of this behavior remains to be determined. For cases where many higher modes are considered, the use of high-order time integration schemes such as IRK64 yields stable and accurate results, and may prove to be essential. Future work will include larger test cases involving more structural modes, and viscous flow simulations.

IX. Acknowledgments

This work was supported in part by grant NNL04AA63G from the NASA Langley Research center.

References

[1]J. J. Alonso and A. Jameson. Fully-implicit time-marching aeroelastic solutions. AIAA 1994-0056, 1994.

[2]T. Baker and P. A. Cavallo. Dynamic adaptation for deforming tetrahedral meshes. AIAA 1999-3253, 1999.

[3]J. T. Batina. Unsteady euler airfoil solutions using unstructured dynamic meshes. *AIAA Journal*, 28(8):1381–1288, 1990.

[4]R. M. Bennett, J. T. Batina, and H. J. Cummingham. Wing-flutter calculations with the cap-tsd unsteady transonic small-disturbance program. *Journal of Aircraft*, 26(9):876–882, 1989.

[5]H. Bijl, M. H. Carpenter, V. N. Vatsa, and C. A. Kennedy. Implicit time integration schemes for the unsteady incompressible Navier-Stokes equations: laminar flow. *Journal of Computational Physics*, 179(1):313–329, 2002.

[6]C. Farhat, C. Degand, B. Koobus, and M. Lesoinne. Torsional spring for two-dimensional dynamic unstructured fluid meshes. *Computer Methods in Applied Mechanics and Engineering*, 163:231–245, 1998.

[7]C. Farhat, M. Lesoinne, and P. LeTallec. Load and motion transfer algorithms for fluid/structure interaction problems with non-matching discrete interfaces: Momentum and energy conservation, optimal discretization and application to aeroelasticity. *Computer Methods in Applied Mechanics and Engineering*, 157:95–114, 1998.

[8]H. Guillard and C. Farhat. On the significance of the geometric conservation law for flow computations on moving meshes. *Computer Methods in Applied Mechanics and Engineering*, 190:1467–1482, 2000.

[9]R. L. Harder and R. N. Desmarais. Interpolation using surface splines. *Journal of Aircraft*, 9(2):189–191, 1972.

[10]G. Jothiprasad, D. J. Mavriplis, and D. A. Caughey. Higher-order time integration schemes for the unsteady navier stokes equations on unstructured meshes. *Journal of Computational Physics*, 191:542–566, 2003.

[11]R. Kamakoti, W. Shyy, S. Thakur, and B. Sankar. Time dependent RANS computation for an aeroelastic wing. AIAA 2004-0886, 2004.

[12]B. Koobus and C. Farhat. On the implicit time integration of semi-discrete viscous fluxes on unstructured dynamic meshes. *International Journal for Numerical Methods in Fluids*, 29:975–996, 1999.

[13]B. Koobus and C. Farhat. Second-order time-accurate and geometrically conservative implicit schemes for flow computations on unstructured dynamic meshes. *Computer Methods in Applied Mechanics and Engineering*, 170:103–129, 1999.

[14]E. M. Lee-Rausch and J. T. Batina. Calculation of agard wing 445.6 flutter using navier-stokes aerodynamics. AIAA 1993-3476, 1993.

[15]M. Lesoinne and C. Farhat. Geometric conservation laws for flow problems with moving boundaries and deformable meshes, and their impact on aeroelastic computations. *Computer Methods in Applied Mechanics and Engineering*, 134:71–90, 1996.

[16]F. Liu, J. Cai, Y. Zhu, H. Tsai, and A. Wong. Calculation of wing flutter by a coupled fluid-structure method. *Journal of Aircraft*, 38(2):334–342, 2001.

[17]D. J. Mavriplis. Multigrid techniques for unstructured meshes. In *VKI Lecture Series*, number 1995-02 in VKI-LS. 1995.

[18]D. J. Mavriplis. Multigrid strategies for viscous flow solvers on anisotropic unstructured meshes. *Journal of Computational Physics*, 145(1):141–165, Sept. 1998.

[19]D. J. Mavriplis. On convergence acceleration techniques for unstructured meshes. AIAA 1998-2966, 1998.

[20]D. J. Mavriplis. Directional agglomeration multigrid techniques for high-reynolds number viscous flow. *AIAA Journal*, 27(10):1222–1230, 1999.

[21]D. J. Mavriplis. An assessment of linear versus non-linear multigrid methods for unstructured mesh solvers. *Journal of Computational Physics*, 175:302–325, 2002.

[22]D. J. Mavriplis and S. Pirzadeh. Large-scale parallel unstructured mesh computations for 3D high-lift analysis. *AIAA Journal of Aircraft*, 36(6):987–998, Dec. 1999.

[23]D. J. Mavriplis and V. Venkatakrishnan. A unified multigrid solver for the navier-stokes equations on mixed element meshes. *International Journal of Computational Fluid Dynamics*, (8):247–263, 1997.

[24]D. J. Mavriplis and Z. Yang. Achieving higher-order time accuracy for dynamic unstructured mesh fluid flow simulations: Role of the GCL. AIAA 2005-5114, June 2005.

[25]D. J. Mavriplis and Z. Yang. Construction of the discrete geometric conservation law for high-order time accurate simulations on dynamic meshes. *To appear in Journal of Computational Physics*, 2005.

[26]M. Murayama, K. Nakahashi, and K. Matsushima. Unstructured dynamic mesh for large movement and deformation. AIAA 2002-0122, 2002.

[27]E. J. Nielsen and W. K. Anderson. Recent improvements in aerodynamic optimization on unstructured meshes. *AIAA Journal*, 40(6):1155–1163, June 2002.

[28]M. Sadeghi, F. Liu, K. Lai, and H. Tsai. Application of three-dimensional interface for data transfer in aeroelastic computations. 22nd Applied Aerodynamics Conference and Exhibit, AIAA 2004-5376, 2004.

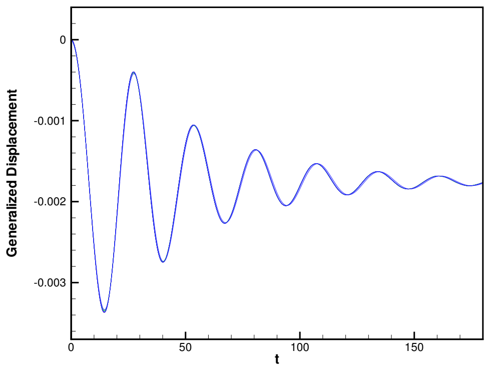
[29]P. D. Thomas and C. K. Lombard. Geometric conservation law and its application to flow computations on moving grids. *AIAA Journal*, 17(10):1030–1037, 1979.

[30]A. van Zuijlen and H. Bijl. High order time integration for fluid-structure interaction on moving meshes. AIAA 2005-5247, 2005.

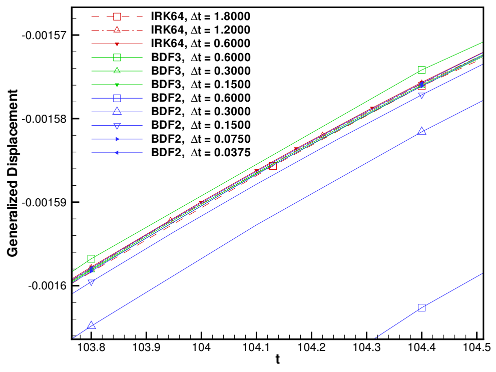
[31]V. Venkatakrishnan and D. J. Mavriplis. Implicit method for the computation of unsteady flows on unstructured grids. *Journal of Computational Physics*, 127:380–397, 1996.

[32]Z. Yang and D. J. Mavriplis. Unstructured dynamic meshes with higher-order time integration schemes for the unsteady navier-stokes equations. AIAA Paper 2005-1222, 2005.

[33]E. C. J. Yates. AGARD standard aeroelastic configurations for dynamic response I - Wing 445.6. Technical Report TM 100492, NASA, 1987.

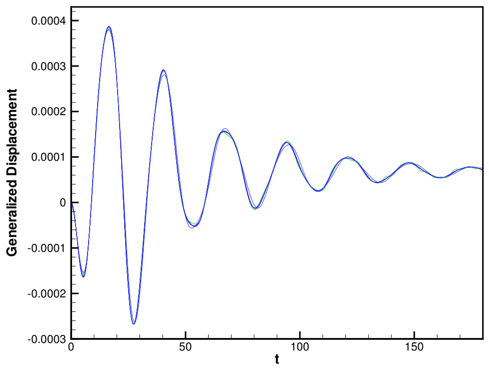


(a)

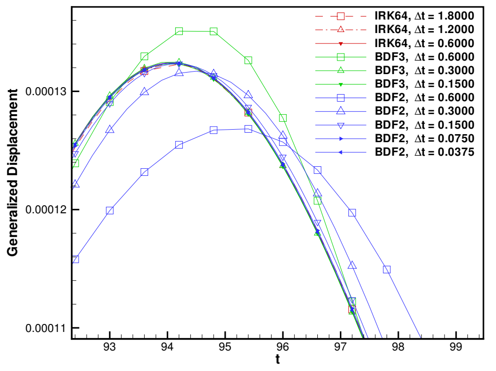


(b) zoom in

Figure 7. Time history of the first mode, M = 0.499, speed index of 0.4459, stable response.

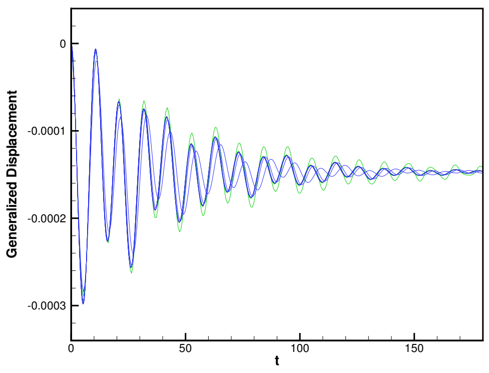


(a)

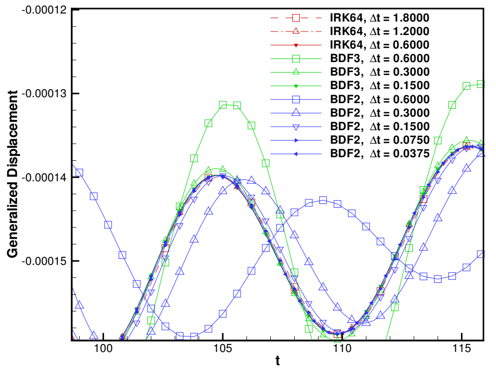


(b) zoom in

Figure 8. Time history of the second mode, M = 0.499, speed index of 0.4459, stable response.

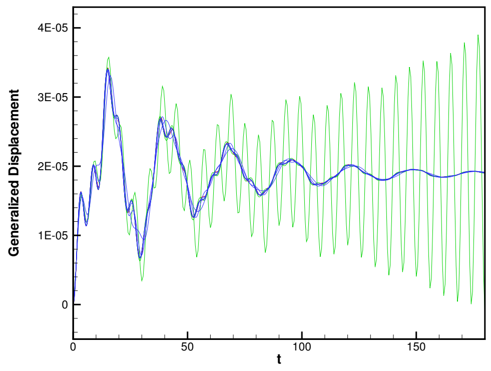


(a)

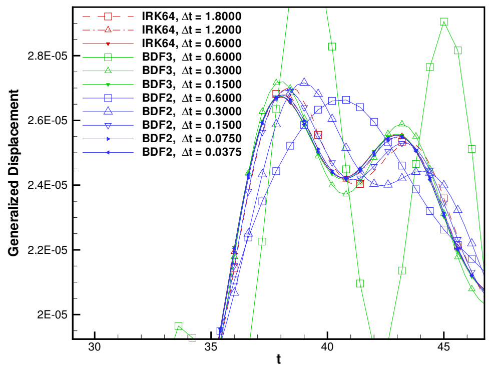


(b) zoom in

Figure 9. Time history of the third mode, M = 0.499, speed index of 0.4459, stable response.

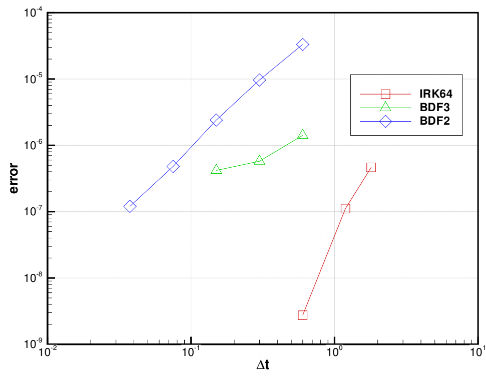


(a)

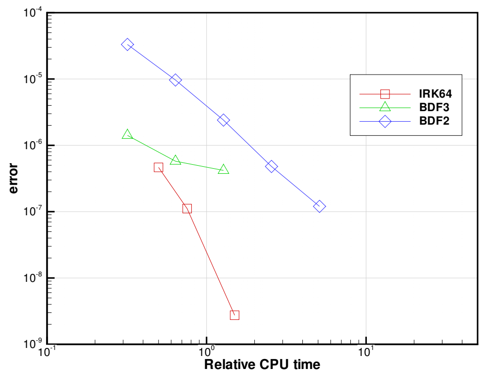


(b) zoom in

Figure 10. Time history of the fouth mode, M = 0.499, speed index of 0.4459, stable response.

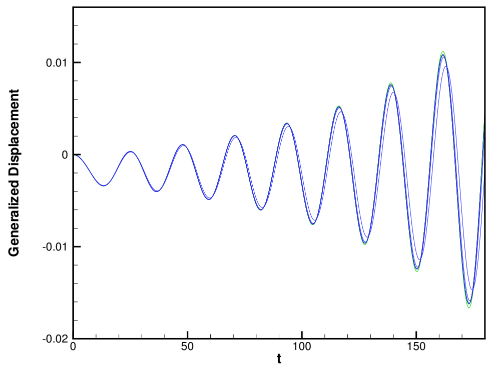


(a)

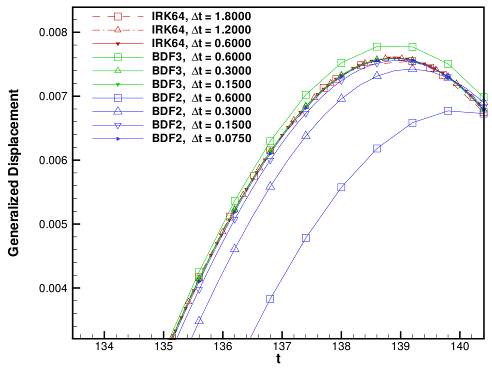


(b)

Figure 11. Error vs. Δt and CPU time, M = 0.499, speed index of 0.4459, stable response.

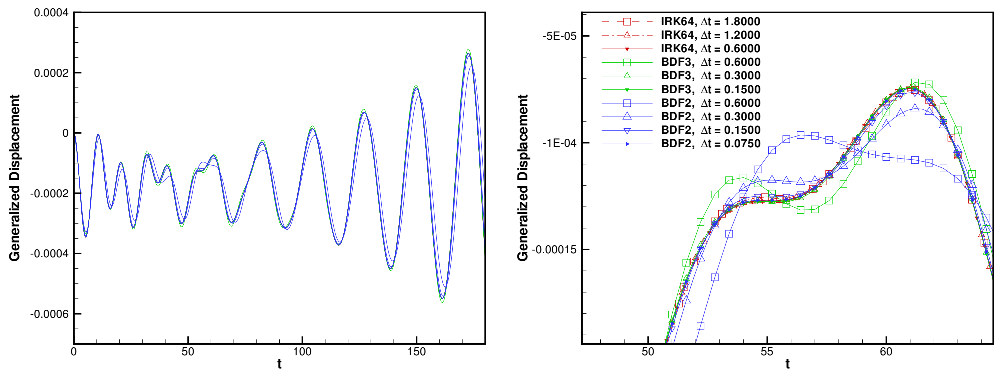


(a)



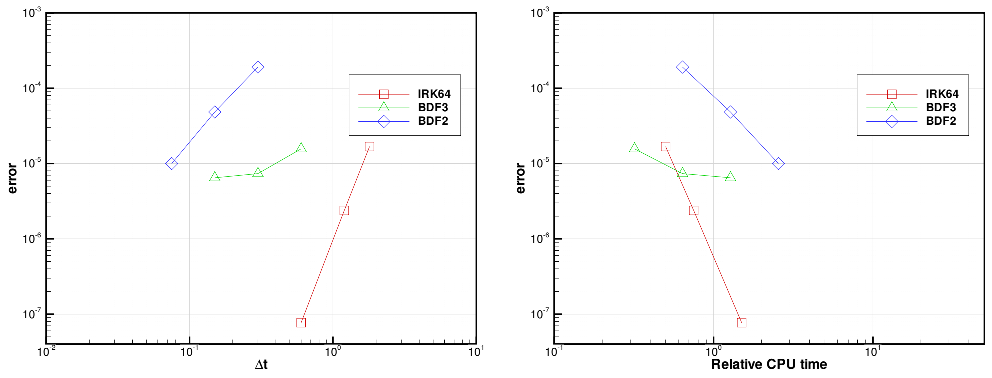
(b) zoom in

Figure 12. Time history of the first mode, M = 0.499, speed index of 0.49, unstable response.



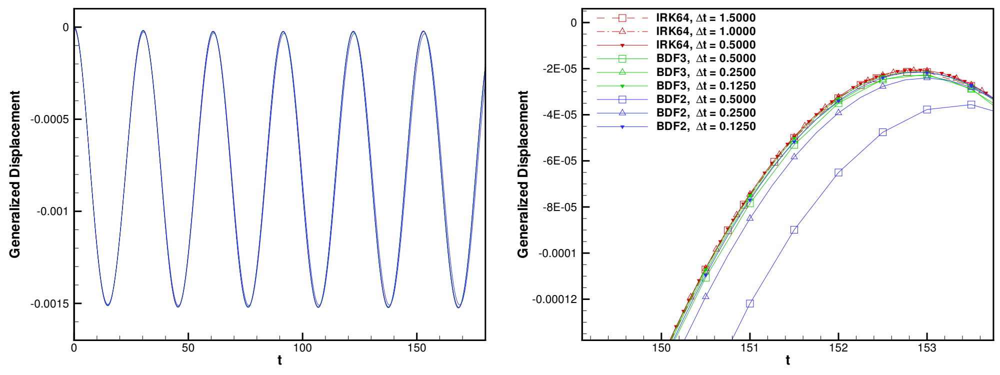
(a) (b) zoom in

Figure 13. Time history of the third mode, M = 0.499, speed index of 0.49, unstable response.



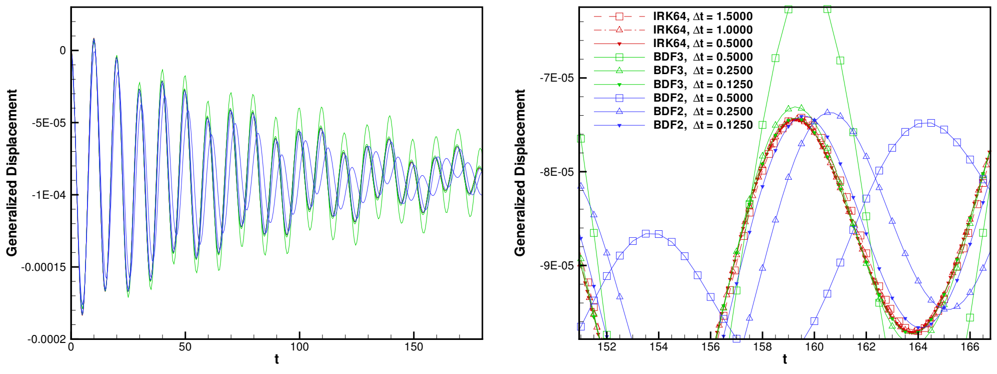
(a) (b)

Figure 14. Error vs. Δt and CPU time, M = 0.499, speed index of 0.49, unstable response.



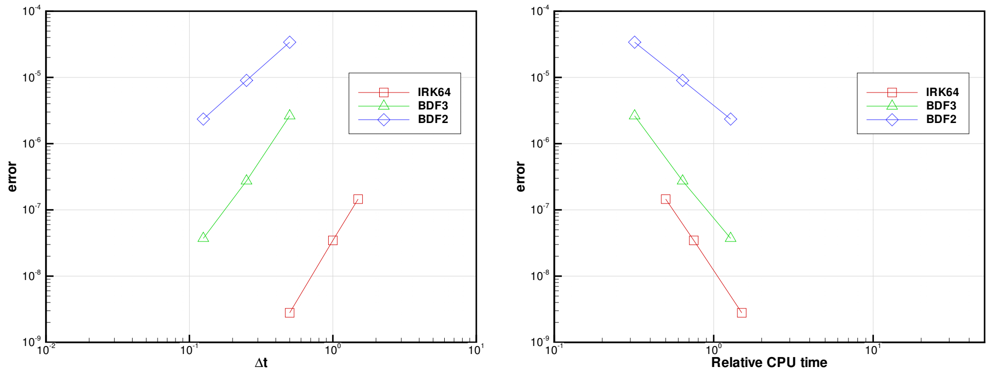
(a) (b) zoom in

Figure 15. Time history of the first mode, M = 0.901, speed index of 0.37, neutral response.



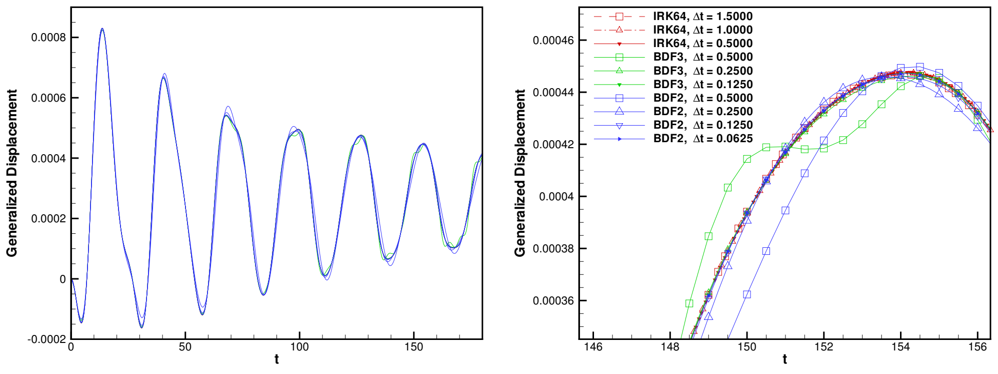
(a) (b) zoom in

Figure 16. Time history of the third mode, M = 0.901, speed index of 0.37, neutral response.



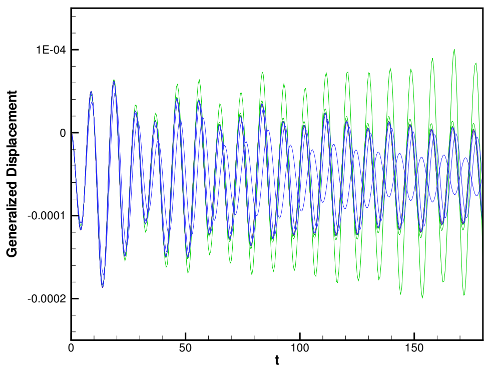
(a) (b)

Figure 17. Error vs. Δt and CPU time, M = 0.901, speed index of 0.37, neutral response.

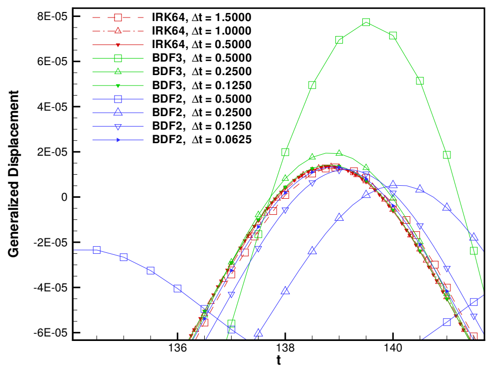


(a) (b) zoom in

Figure 18. Time history of the first mode, M = 1.141, speed index of 0.442, stable response.

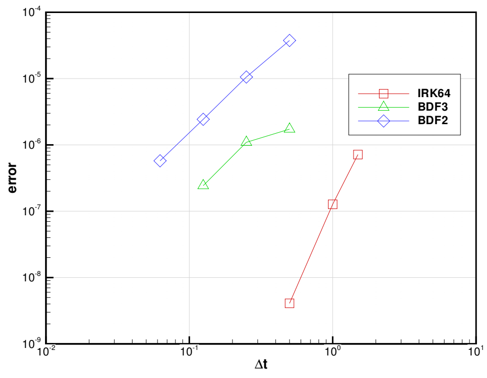


(a)

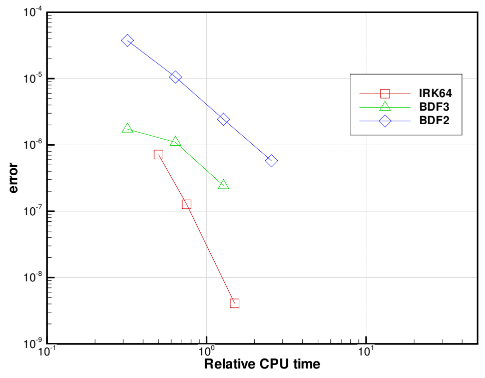


(b) zoom in

Figure 19. Time history of the third mode, M = 1.141, speed index of 0.442, stable response.



(a)



(b)

Figure 20. Error vs. Δt and CPU time, M = 1.141, speed index of 0.442, stable response.

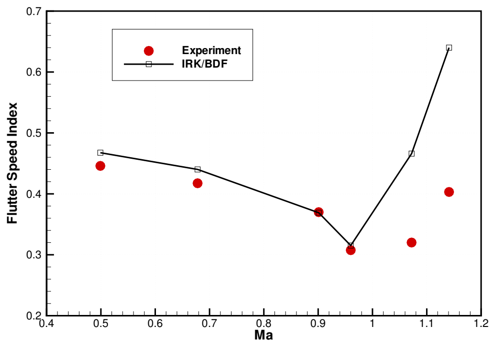


Figure 21. Flutter speed index.