Linearization of the Coupled Unsteady Fluid-Structure Equations: Application to Flutter Control

Karthik Mani *
Dimitri J. Mavriplis †
Department of Mechanical Engineering, University of Wyoming, Laramie, Wyoming 82071-3295

A method to compute sensitivities for use in gradient-based shape optimization applied to unsteady aeroelasticity problems is presented. The method takes into account the coupling between all three fundamental aspects of computational aeroelasticity, namely, unsteady flow equations, time varying structural response to aerodynamic loads, and dynamic meshes that accommodate geometric deformations. The devised method provides discretely exact sensitivities of time-integrated and non-time-integrated objective functions with respect to design variables that control the shape of geometry. The algorithm is formulated in a general manner and can be readily extended to coupled multidisciplinary problems involving any number of disciplines. The algorithm is applied to a simple two-dimensional flutter model in order to demonstrate the proof-of-concept.

I. Introduction

With advances in computational power, large scale aeroelastic simulations in CFD are becoming more and more commonplace. This paper is concerned with advancing the role of unsteady aeroelastic simulations in aircraft design by complementing them with the strength of adjoint equations for obtaining functional sensitivities. The strength of adjoint equations lie in the fact that they permit the computation of functional sensitivity derivatives at a cost that is essentially independent of the number of design variables. Adjoint equations have become very popular in solving aerodynamic shape optimization problems, particularly for steady-state conditions.[1–8] While only recently adjoint methods have begun gaining ground in the aerodynamics community especially for time-dependent problems, they have been well established in the field of structural optimization for some time now.[9] The coupling between the fluid and structure equations and the use of sensitivity analysis on such a system has been addressed but primarily from a steady-state standpoint.[10,11] Until now relatively little work has been done toward addressing unsteady aeroelastic optimization problems mainly due to prohibitive cost and complexities in the linearization of coupled time-dependent systems. Recently, efficient linearization techniques for the unsteady governing flow equations both viscous and inviscid have been developed and applied to unsteady shape optimization problems.[12–16] It is only natural that such methodology be extended to include coupling effects brought on by the introduction of structural equations or any other set of governing equations. This paper presents work done on developing a generalized modular framework for obtaining sensitivities for the coupled unsteady fluid-structure equations. The method is modular in the sense that multiple sets of governing equations from various disciplines that are coupled may be addressed in a unified manner independent of the number of disciplines. Also, the choice of solution technique employed for the individual disciplines has little impact on the overall framework. This is particularly important from a cost standpoint since efficient solution techniques such as multigrid methods and high-order time integration schemes exploited for the purpose of reducing overall costs inherent in unsteady simulations should carry over to the sensitivity computations. It should also be noted that no approximations in the process of linearization have to be made and all components contributing to the solution of the multidisciplinary analysis problem are taken into account. The work presented here utilizes the time-dependent inviscid Euler equations in an unstructured finite-volume framework for the flow solver and a simple two-dimensional flutter model with pitch and plunge degrees-of-freedom to represent the structural response. While we use a direct solution procedure for the structures discipline by transforming the governing equations into linear differential equations, comprehensive problems based on modal analysis or nonlinear structural models may also be treated by this method.

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

II. Analysis Problem Formulation

A. 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 to demonstrate the linearization of the coupling between unsteady fluid and structure equations. Extension of the algorithm to viscous flow problems with the inclusion of turbulence models should prove straightforward, based on previous work for time-dependent adjoints of the Navier-Stokes equations.[13,14] 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

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

where the state vector $\mathbf{U}$ of conserved variables and the cartesian inviscid flux vector $\mathbf{F} = (\mathbf{F}^x, \mathbf{F}^y)$ are

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

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

$$p = (\gamma - 1)\left[E_t - \frac{1}{2}\rho(u^2 + v^2)\right] \tag{3}$$

where $\gamma = 1.4$ is the ratio of specific heats. Applying the divergence theorem and integrating over a moving control volume $A(t)$ that is bounded by the control surface $B(t)$ yields

$$\int_{A(t)} \frac{\partial \mathbf{U}}{\partial t} dA + \int_{B(t)} \mathbf{F}(\mathbf{U}) \cdot \mathbf{n} dB = 0 \tag{4}$$

Using the differential identity

$$\frac{\partial}{\partial t}\int_{A(t)} \mathbf{U} dA = \int_{A(t)} \frac{\partial \mathbf{U}}{\partial t} dA + \int_{B(t)} (\dot{\mathbf{x}} \cdot \mathbf{n}) \mathbf{U} dB \tag{5}$$

equation (4) is rewritten as

$$\frac{\partial}{\partial t}\int_{A(t)} \mathbf{U} dA + \int_{B(t)} [\mathbf{F}(\mathbf{U}) - \dot{\mathbf{x}}\mathbf{U}] \cdot \mathbf{n} dB = 0 \tag{6}$$

or when considering cell-averaged values for the state $\mathbf{U}$ as

$$\frac{\partial A\mathbf{U}}{\partial t} + \int_{B(t)} [\mathbf{F}(\mathbf{U}) - \dot{\mathbf{x}}\mathbf{U}] \cdot \mathbf{n} dB = 0 \tag{7}$$

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 or volume of the element, $\dot{\mathbf{x}}$ is the vector of mesh face or edge velocities, $\mathbf{n}$ is the unit normal of the face or edge, and B refers to the area or length of the bounding surface or edge.

B. Temporal discretization

The time derivative term in the Euler equations is discretized using a second-order accurate backward-difference-formula (BDF2) scheme as shown in equation (8). The index n is used to indicate the current time-level as the convention throughout the paper. The discretization shown is based on a uniform time-step size.

$$\frac{\partial A\mathbf{U}}{\partial t} = \frac{\frac{3}{2}A^n\mathbf{U}^n - 2A^{n-1}\mathbf{U}^{n-1} + \frac{1}{2}A^{n-2}\mathbf{U}^{n-2}}{\Delta t} \tag{8}$$

C. Spatial discretization

The flow solver uses a cell-centered finite-volume formulation where the inviscid flux integral S around a closed control volume is discretized as

$$S = \int_{dB(t)} [\mathbf{F}(\mathbf{U}) - \dot{\mathbf{x}}\mathbf{U}] \cdot \mathbf{n} dB = \sum_{i=1}^{n_{edge}} \mathbf{F}_{e_i}^{\perp}(V_{e_i}, \mathbf{U}, \mathbf{n}_{ei}) B_{e_i} \tag{9}$$

where B_e is the edge length, V_e is the normal edge velocity, $\mathbf{n}_e$ is the unit normal of the edge, and $F_e^{\perp}$ is the normal flux across the edge. The normal flux across the edge is computed using the second-order accurate matrix dissipation scheme[17] as the sum of a central difference and an artificial dissipation term as shown below,

$$\mathbf{F}_e^{\perp} = \frac{1}{2}\left\{\mathbf{F}_L^{\perp}(\mathbf{U}_L, V_e, \mathbf{n}_e) + \mathbf{F}_R^{\perp}(\mathbf{U}_R, V_e, \mathbf{n}_e) + \kappa^{(4)}[T]|[\lambda]|[T]^{-1}\left\{(\nabla^2\mathbf{U})_L - (\nabla^2\mathbf{U})_R\right\}\right\} \tag{10}$$

where $\mathbf{U}_L$, $\mathbf{U}_R$ are the left and right state vectors and $(\nabla^2\mathbf{U})_L$, $(\nabla^2\mathbf{U})_R$ are the left and right undivided Laplacians computed for any element i as

$$(\nabla^2\mathbf{U})_i = \sum_{k=1}^{neighbors} (\mathbf{U}_k - \mathbf{U}_i) \tag{11}$$

The matrix $[\lambda]$ is diagonal and consists of the eigenvalues (adjusted by normal edge velocity V_e) of the flux Jacobian matrix $\frac{\partial \mathbf{F}^{\perp}}{\partial \mathbf{U}}$, and the matrix $[T]$ consists of the corresponding eigenvectors. The scalar parameter $\kappa^{(4)}$ is empirically determined and controls the amount of artificial dissipation added to the centrally differenced flux. For transonic problems this is usually taken as 0.1. The advantage of using the difference of the undivided Laplacians in the construction of the convective flux is that it offers second-order spatial accuracy without the need for state reconstruction techniques. The normal native flux vector is computed as

$$\mathbf{F}^{\perp} = \begin{pmatrix} \rho(V^{\perp} - V_e) \\ \rho(V^{\perp} - V_e)u + \hat{n}_x p \\ \rho(V^{\perp} - V_e)v + \hat{n}_y p \\ E_t(V^{\perp} - V_e) + pV^{\perp} \end{pmatrix} \tag{12}$$

where the fluid velocity normal to the edge $V^{\perp}$ is defined as $u\hat{n}_x + v\hat{n}_y$, where $\hat{n}_x$ and $\hat{n}_y$ are the unit edge normal vector components.

D. The discrete geometric conservation law (GCL)

The discrete geometric conservation law (GCL) requires that a uniform flow field be preserved when equation (7) is integrated in time. In other words the 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 (7). 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 (7) results in the mathematical description of the GCL as stated below:[18,19]

$$\frac{\partial A}{\partial t} - \int_{dB(t)} \dot{\mathbf{x}} \cdot \mathbf{n} dB = 0 \tag{13}$$

For the second-order accurate BDF2 time-integration scheme used in this work, this can be discretely represented using equations (8) and (9) as:

$$\left(\frac{\frac{3}{2}A^n - 2A^{n-1} + \frac{1}{2}A^{n-2}}{\Delta t}\right) - \sum_{i=1}^{n_{edge}} V_{e_i} B_i = 0 \tag{14}$$

Equation (14) implies that the change in area of an element in the mesh should be discretely equal to the area swept by the bounding edges of the element. The edge velocities V_{e_i} for each of the edges encompassing the element must therefore be chosen such that equation (14) is satisfied. While various methods for computing the edge velocities satisfying the GCL have been developed for different temporal discretizations,[18,19] a unifying approach applicable to first, second and third-order backward differencing schemes (BDF) as well as higher-order accurate implicit Runge-Kutta (IRK) schemes has been developed in reference.[20] This formulation is used exclusively in the current work. While the details of the formulation are not central to the current work, the functional form of the face-integrated edge velocities is important for the derivation of the adjoint equations. In our formulation, these values depend on the mesh coordinates $\mathbf{x}^n$, $\mathbf{x}^{n-1}$ and $\mathbf{x}^{n-2}$ for the BDF2 scheme.

E. Mesh deformation strategy

Deformation of the mesh is achieved through the linear tension spring analogy[4,21] which approximates the mesh as a network of inter-connected springs. The spring coefficient is assumed 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 linear system that relates the interior node displacements in the mesh to known displacements on the boundaries is given as

$$[K]\delta\mathbf{x}_{int} = \delta\mathbf{x}_{surf} \tag{15}$$

where $[K]$ is the stiffness matrix assembled using the spring coefficients of each of the edges in the computational mesh.

F. Geometry parametrization

Modification of the baseline geometry is achieved through displacements of the surface nodes defining the geometry. In order to ensure smooth geometries, surface node displacements are controlled by bump functions placed at various chordwise locations on the airfoil. Each bump function influences the displacements in the y direction of all surface nodes with a diminishing effect moving away from the location of the bump. The bump function although a function of x does not have any influence in the x direction. The bump function used for the work presented in this paper is the Hicks-Henne Sine bump function.[22] The design variables or inputs for the optimization example presented form a vector of weights controlling the magnitudes of bump functions placed at various chordwise locations.

G. Aeroelastic structural equations

The aeroelastic model is based on the response of an airfoil with two degrees-of-freedom, namely pitch and plunge. The equations of motion for such a system can be summarized as

$$\begin{aligned} m\ddot{h} + S_\alpha\ddot{\alpha} + K_h h &= -L \\ S_\alpha\ddot{h} + I_\alpha\ddot{\alpha} + K_\alpha\alpha &= M_{ea} \end{aligned} \tag{16}$$

where

- m: mass of airfoil
- S_α: static imbalance
- I_α: sectional moment of inertia of airfoil
- K_h: plunging spring coefficient
- K_α: pitching spring coefficient
- h: vertical displacement (positive downward)
- α: angle-of-attack
- L: sectional lift of airfoil
- M_{ea}: sectional moment of airfoil about elastic axis (positive nose up)

Non-dimensionalizing equations (16) and (17) yields

$$[M]\{\ddot{q}\} + [K]\{q\} = \{F\} \tag{17}$$

where

$$[M] = \begin{bmatrix} 1 & x_\alpha \\ x_\alpha & r_\alpha^2 \end{bmatrix}, \quad [K] = \begin{bmatrix} \left(\frac{\omega_h}{\omega_\alpha}\right)^2 & 0 \\ 0 & r_\alpha^2 \end{bmatrix}, \quad \{F\} = \frac{1}{\pi\mu k_c^2}\begin{Bmatrix} -C_l \\ 2C_m \end{Bmatrix}, \quad \{q\} = \begin{Bmatrix} \frac{h}{b} \\ \alpha \end{Bmatrix} \tag{18}$$

are the non-dimensional mass matrix, stiffness matrix, load vector and displacement vector and

b: semichord of airfoil

k_c: reduced frequency of pitch, $k_c = \frac{\omega c}{2U_\infty}$

μ: airfoil mass ratio, $\mu = \frac{m}{\pi\rho b^2}$

ω_h, ω_α: uncoupled natural frequencies of plunge and pitch

x_α: structural parameter defined as $\frac{S_\alpha}{mb}$

r_α^2: structural parameter defined as $\frac{I_\alpha}{mb}$

C_l, C_m: section lift coefficient and section moment coefficient about the elastic axis

The reduced frequency k_c is typically written in terms of the flutter velocity V_f as

$$k_c = \frac{\omega_\alpha c}{2U_\infty} = \frac{1}{V_f\sqrt{\mu}} \tag{19}$$

where c is the chord length of the airfoil, U_∞ is the freestream velocity, and the flutter velocity V_f is defined as

$$V_f = \frac{U_\infty}{\omega_\alpha b\sqrt{\mu}} \tag{20}$$

H. Transformation of structural equations

The aeroelastic equations as shown in Equation (17) are second-order partial differential equations. These are transformed into two first-order linear equations in order to facilitate a direct solution procedure. The transformation is given as[23]

$$\begin{aligned} \mathbf{r}_1 &= \{q\} \\ \mathbf{r}_2 &= \dot{\mathbf{r}}_1 \end{aligned} \tag{21}$$

The resulting first-order equations are then

$$\begin{aligned} \dot{\mathbf{r}}_1 &= \mathbf{r}_2 \\ \dot{\mathbf{r}}_2 &= -[M]^{-1}[K]\mathbf{r}_1 + [M]^{-1}\{F\} \end{aligned} \tag{22}$$

or in matrix notation as

$$\begin{Bmatrix} \dot{\mathbf{r}}_1 \\ \dot{\mathbf{r}}_2 \end{Bmatrix} = \begin{bmatrix} 0 & [I] \\ -[M]^{-1}[K] & 0 \end{bmatrix} \begin{Bmatrix} \mathbf{r}_1 \\ \mathbf{r}_2 \end{Bmatrix} + \begin{Bmatrix} 0 \\ [M]^{-1}\{F\} \end{Bmatrix} \tag{23}$$

$$\dot{\mathbf{r}} = [\Psi]\mathbf{r} + \{\Phi\} \tag{24}$$

The matrix $[\Psi]$ is a constant and can be precomputed and stored for use during the time-integration process. The time derivative term $\dot{\mathbf{r}}$ is discretized using the second-order accurate BDF2 scheme as in the case of the time derivative term in the flow equations. It should be noted that the time-step $\Delta\tau$ appearing in the denominator of the discretized version of the structural equations is different from the time-step of the flow equations, and their relation is $\Delta\tau = 2k_c\Delta t$, where Δt is the non-dimensional time-step used for the flow equations. This is due to the non-dimensionalization of time in the structural equations by the uncoupled natural frequency in pitch ω_α. Once the vector $\mathbf{r}^n$ at a time-level n has been solved for, the displacement vector $\{q\}$ is known and the configuration of the newly deformed mesh (i.e. $\mathbf{x}^n$) can be computed using the mesh deformation equations (15).

III. Solution Procedure for Analysis Problem

An implicit solution method is employed to solve both the flow and structure equations. At each time-level n an implicit flow residual equation $\mathbf{R}^n$ can be defined as

$$\mathbf{R}^n = \frac{\partial A\mathbf{U}}{\partial t} + S^n(\mathbf{U}^n, V_e^n, \mathbf{n}^\mathbf{n}) = \mathbf{0} \tag{25}$$

where a BDF2 time discretization of the flow residual equation in functional form can be written as

$$\mathbf{R}^n(\mathbf{U}^n, \mathbf{U}^{n-1}, \mathbf{U}^{n-2}, \mathbf{x}^n, \mathbf{x}^{n-1}, \mathbf{x}^{n-2}) = 0 \tag{26}$$

Constructing a Newton scheme by linearizing the flow residual equation about the flow state variables $\mathbf{U}^n$, the solution can be iteratively obtained as

$$\begin{aligned} \left[\frac{\partial \mathbf{R}^k}{\partial \mathbf{U}^k}\right] \delta \mathbf{U}^k &= -\mathbf{R}^k \\ \mathbf{U}^{k+1} &= \mathbf{U}^k + \delta \mathbf{U}^k \\ \delta \mathbf{U}^k \rightarrow 0, \mathbf{U}^n &= \mathbf{U}^k \end{aligned} \tag{27}$$

Similarly, at each time-level n an implicit structural residual equation can be defined as

$$\mathbf{J}^n = \frac{\partial \mathbf{r}}{\partial \tau} - [\Psi]\mathbf{r}^n - \{\Phi\} = 0 \tag{28}$$

In functional form, for a BDF2 time discretization the residual can be written as

$$\mathbf{J}^n(\mathbf{r}^n, \mathbf{r}^{n-1}, \mathbf{r}^{n-2}, \mathbf{U}^n, \mathbf{x}^n) = 0 \tag{29}$$

The corresponding Newton scheme is then

$$\left[\frac{\partial \mathbf{J}^k}{\partial \mathbf{r}^k}\right] \delta \mathbf{r}^k = -\mathbf{J}^k \tag{30}$$

Although the above equation is derived from a Newton scheme, the solution can be obtained linearly since the Jacobian matrix is a constant. The interior mesh coordinates at any time-level n can be described as a function of the boundary mesh coordinates at that time-level using the mesh deformation equations as

$$\begin{aligned} [K]\delta \mathbf{x}^n &= \delta \mathbf{x}^n_{surf} \\ \delta \mathbf{x}^n &= \mathbf{x}^n - \mathbf{x}^0 \\ \delta \mathbf{x}^n_{surf} &= \mathbf{x}^n_{surf} - \mathbf{x}^0_{surf} \end{aligned} \tag{31}$$

where

$$\mathbf{x}^n_{surf} = f(\mathbf{x}^0_{surf}, \mathbf{r}^n) \tag{32}$$

Here $\mathbf{x}^0_{surf}$ refers to the baseline airfoil coordinates at the neutral position, and in the case of an optimization refers to the shape of the current optimum airfoil, while $\mathbf{x}^0$ refers to the baseline interior mesh coordinates. The coupling between the fluid and structure equations is now apparent, since the flow residual equation depends on the vector $\mathbf{r}^n$ through the mesh deformation equations, and the structural residual equation depends on the flow solution $\mathbf{U}^n$ and mesh coordinates $\mathbf{x}^n$.

The solution procedure for the coupled system at each time-step can be broken down as follows.

1. Initiate $\mathbf{r}^n$, $\mathbf{U}^n$, and $\mathbf{x}^n$ with values from the previous time-level.
2. Obtain an approximate solution for $\mathbf{r}^n$ by partially solving the structural dynamics system as per equation (30).
3. Use the approximate $\mathbf{r}^n$ to determine an approximate $\mathbf{x}^n$ by partially solving the linear mesh deformation equations as given by equation (31).
4. Obtain an approximate solution for $\mathbf{U}^n$ by partially solving the nonlinear flow problem based on the approximate mesh coordinates $\mathbf{x}^n$.
5. Use the new estimate of $\mathbf{U}^n$ and $\mathbf{x}^n$ to solve for an improved estimate of $\mathbf{r}^n$.
6. Keep repeating the procedure until $\mathbf{r}^n$, $\mathbf{x}^n$ and $\mathbf{U}^n$ converge.
7. Proceed to the next time-level.

It is important to note at this point that only partial solutions of each set of equations are required during the coupled iterative solution procedure. Completely solving each system during the coupled iterations does not change the final result but will impose severe additional cost. In our work the Newton solver for the flow equations is driven by an agglomeration linear multigrid algorithm, while the mesh and structural equations are solved using Gauss-Seidel iterations. Typically a single coupled iteration consists of one nonlinear Newton iteration for the flow equations driven by two linear multigrid cycles, 50 Gauss-Seidel iterations for the mesh motion equations and three Gauss-Seidel iterations for the structural equations. The larger number of iterations used for the mesh motion equations is due to the absence of a multigrid acceleration scheme for these equations. However, the overall cost of solving the mesh motion equations remains small compared to that required by the flow solution process. Based on these settings, it takes approximately 130 coupled iterations to reach machine precision on all three sets of equations. A plot of the typical convergence of the coupled system is shown in Figure (1(a)).

IV. Adjoint Sensitivity Formulation

Consider a functional L computed using the solution set of an unsteady aeroelastic simulation. The functional when linearized with respect to a vector of design inputs $\mathbf{D}$ can then be expressed as

$$\frac{dL}{d\mathbf{D}} = \sum_{n=1}^{n_{steps}} \left\{ \frac{\partial L}{\partial \mathbf{r}^n}\frac{\partial \mathbf{r}^n}{\partial \mathbf{D}} + \frac{\partial L}{\partial \mathbf{U}^n}\frac{\partial \mathbf{U}^n}{\partial \mathbf{D}} + \frac{\partial L}{\partial \mathbf{x}^n}\frac{\partial \mathbf{x}^n}{\partial \mathbf{D}} \right\} = \sum_{n=1}^{n_{steps}} \left\{ \frac{\partial L}{\partial \mathbf{D}} \right\}^n \tag{33}$$

This represents the linearization of a time-integrated functional L that depends on the state variables at all time-levels, and we use this case to present the derivation of the sensitivity. The non time-integrated case is a subset of this where the only non-zero term in the summation would be at $n = n_{steps}$. Each term in the sum can be written in the form of an inner product of two vectors as

$$\frac{\partial L}{\partial \mathbf{D}}^n = \left\{ \begin{array}{ccc} \frac{\partial L}{\partial \mathbf{r}^n} & \frac{\partial L}{\partial \mathbf{U}^n} & \frac{\partial L}{\partial \mathbf{x}^n} \end{array} \right\} \left\{ \begin{array}{ccc} \frac{\partial \mathbf{r}^n}{\partial \mathbf{D}}^T & \frac{\partial \mathbf{U}^n}{\partial \mathbf{D}}^T & \frac{\partial \mathbf{x}^n}{\partial \mathbf{D}}^T \end{array} \right\}^T \tag{34}$$

The elements in the first vector of the inner product shown in equation (34) are vectors by themselves and the elements in the second vector are matrices. The linearization of the functional L with respect to the various state variables (i.e. $\mathbf{r}, \mathbf{U}, \mathbf{x}$) are vectors and easily computable since L is a scalar quantity. However, the linearization of the state variables with respect to the design inputs are matrices. In order to determine convenient expressions for the state variable sensitivity matrices, we linearize the corresponding residual equations. Considering the final time-step in the summation ($n = n_{steps}$), the three residual equations arising from each discipline (i.e. flow, mesh and structure) when linearized against design inputs $\mathbf{D}$ can be written as

$$\frac{\partial \mathbf{J}^n}{\partial \mathbf{r}^n}\frac{\partial \mathbf{r}^n}{\partial \mathbf{D}} + \frac{\partial \mathbf{J}^n}{\partial \mathbf{r}^{n-1}}\frac{\partial \mathbf{r}^{n-1}}{\partial \mathbf{D}} + \frac{\partial \mathbf{J}^n}{\partial \mathbf{r}^{n-2}}\frac{\partial \mathbf{r}^{n-2}}{\partial \mathbf{D}} + \frac{\partial \mathbf{J}^n}{\partial \mathbf{U}^n}\frac{\partial \mathbf{U}^n}{\partial \mathbf{D}} + \frac{\partial \mathbf{J}^n}{\partial \mathbf{x}^n}\frac{\partial \mathbf{x}^n}{\partial \mathbf{D}} = 0 \tag{35}$$

$$\frac{\partial \mathbf{R}^n}{\partial \mathbf{U}^n}\frac{\partial \mathbf{U}^n}{\partial \mathbf{D}} + \frac{\partial \mathbf{R}^n}{\partial \mathbf{U}^{n-1}}\frac{\partial \mathbf{U}^{n-1}}{\partial \mathbf{D}} + \frac{\partial \mathbf{R}^n}{\partial \mathbf{U}^{n-2}}\frac{\partial \mathbf{U}^{n-2}}{\partial \mathbf{D}} + \frac{\partial \mathbf{R}^n}{\partial \mathbf{x}^n}\frac{\partial \mathbf{x}^n}{\partial \mathbf{D}} + \frac{\partial \mathbf{R}^n}{\partial \mathbf{x}^{n-1}}\frac{\partial \mathbf{x}^{n-1}}{\partial \mathbf{D}} + \frac{\partial \mathbf{R}^n}{\partial \mathbf{x}^{n-2}}\frac{\partial \mathbf{x}^{n-2}}{\partial \mathbf{D}} = 0 \tag{36}$$

$$[K]\frac{\partial \mathbf{x}^n}{\partial \mathbf{D}} - \frac{\partial \mathbf{x}^n_{surf}}{\partial \mathbf{r}^n}\frac{\partial \mathbf{r}^n}{\partial \mathbf{D}} - \frac{\partial \mathbf{x}^n_{surf}}{\partial \mathbf{x}^0_{surf}}\frac{\partial \mathbf{x}^0_{surf}}{\partial \mathbf{D}} = 0 \tag{37}$$

Since the governing system of equations are coupled between the three disciplines, the linearized residual equations can be expressed in combined matrix form as

$$\left[\begin{array}{ccc} \frac{\partial \mathbf{J}^n}{\partial \mathbf{r}^n} & \frac{\partial \mathbf{J}^n}{\partial \mathbf{U}^n} & \frac{\partial \mathbf{J}^n}{\partial \mathbf{x}^n} \\ 0 & \frac{\partial \mathbf{R}^n}{\partial \mathbf{U}^n} & \frac{\partial \mathbf{R}^n}{\partial \mathbf{x}^n} \\ -\frac{\partial \mathbf{x}^n_{surf}}{\partial \mathbf{r}^n} & 0 & [K] \end{array} \right] \left\{ \begin{array}{c} \frac{\partial \mathbf{r}^n}{\partial \mathbf{D}} \\ \frac{\partial \mathbf{U}^n}{\partial \mathbf{D}} \\ \frac{\partial \mathbf{x}^n}{\partial \mathbf{D}} \end{array} \right\} = \left\{ \begin{array}{c} -\frac{\partial \mathbf{J}^n}{\partial \mathbf{r}^{n-1}}\frac{\partial \mathbf{r}^{n-1}}{\partial \mathbf{D}} - \frac{\partial \mathbf{J}^n}{\partial \mathbf{r}^{n-2}}\frac{\partial \mathbf{r}^{n-2}}{\partial \mathbf{D}} \\ -\frac{\partial \mathbf{R}^n}{\partial \mathbf{U}^{n-1}}\frac{\partial \mathbf{U}^{n-1}}{\partial \mathbf{D}} - \frac{\partial \mathbf{R}^n}{\partial \mathbf{U}^{n-2}}\frac{\partial \mathbf{U}^{n-2}}{\partial \mathbf{D}} - \frac{\partial \mathbf{R}^n}{\partial \mathbf{x}^{n-1}}\frac{\partial \mathbf{x}^{n-1}}{\partial \mathbf{D}} - \frac{\partial \mathbf{R}^n}{\partial \mathbf{x}^{n-2}}\frac{\partial \mathbf{x}^{n-2}}{\partial \mathbf{D}} \\ \frac{\partial \mathbf{x}^n_{surf}}{\partial \mathbf{x}^0_{surf}}\frac{\partial \mathbf{x}^0_{surf}}{\partial \mathbf{D}} \end{array} \right\} \tag{38}$$

Equation (38) represents the forward or tangent linearization of the state variables and the corresponding sensitivity matrices may be constructed by solving the coupled system once per design variable by integrating forward in time as indicated by the summation representing the total sensitivity vector. This process is inefficient as the cost of constructing the sensitivity vector is identical to a finite-difference procedure since there is a direct dependence on the size

of the design input vector $\mathbf{D}$. The principal advantage of using the forward linearization is that it provides discretely exact sensitivities unlike finite-differencing where the sensitivity is approximated by a heuristically determined step size. Adjoint methods circumvent this dependence on the size of the design input vector $\mathbf{D}$ by transposing the entire sensitivity equation (33). The transposed sensitivity equation can be written as

$$\frac{dL}{d\mathbf{D}}^T = \sum_{n=1}^{n_{steps}} \left\{ \begin{array}{ccc} \frac{\partial \mathbf{r}^n}{\partial \mathbf{D}} & \frac{\partial \mathbf{U}^n}{\partial \mathbf{D}} & \frac{\partial \mathbf{x}^n}{\partial \mathbf{D}} \end{array} \right\} \left\{ \begin{array}{ccc} \frac{\partial L}{\partial \mathbf{r}^n} & \frac{\partial L}{\partial \mathbf{U}^n} & \frac{\partial L}{\partial \mathbf{x}^n} \end{array} \right\}^T \tag{39}$$

Transposing and rearranging equation (38) to obtain an expression for the state variable sensitivities we get

$$\left\{ \begin{array}{ccc} \frac{\partial \mathbf{r}^n}{\partial \mathbf{D}} & \frac{\partial \mathbf{U}^n}{\partial \mathbf{D}} & \frac{\partial \mathbf{x}^n}{\partial D} \end{array} \right\} = \left\{ \begin{array}{c} -\frac{\partial \mathbf{J}^n}{\partial \mathbf{r}^{n-1}} \frac{\partial \mathbf{r}^{n-1}}{\partial \mathbf{D}} - \frac{\partial \mathbf{J}^n}{\partial \mathbf{r}^{n-2}} \frac{\partial \mathbf{r}^{n-2}}{\partial \mathbf{D}} \\ -\frac{\partial \mathbf{R}^n}{\partial \mathbf{U}^{n-1}} \frac{\partial \mathbf{U}^{n-1}}{\partial \mathbf{D}} - \frac{\partial \mathbf{R}^n}{\partial \mathbf{U}^{n-2}} \frac{\partial \mathbf{U}^{n-2}}{\partial \mathbf{D}} - \frac{\partial \mathbf{R}^n}{\partial \mathbf{x}^{n-1}} \frac{\partial \mathbf{x}^{n-1}}{\partial \mathbf{D}} - \frac{\partial \mathbf{R}^n}{\partial \mathbf{x}^{n-2}} \frac{\partial \mathbf{x}^{n-2}}{\partial \mathbf{D}} \\ \frac{\partial \mathbf{x}^n_{surf}}{\partial \mathbf{x}^0_{surf}} \frac{\partial \mathbf{x}^0_{surf}}{\partial \mathbf{D}} \end{array} \right\}^T \left[\begin{array}{ccc} \frac{\partial \mathbf{J}^n}{\partial \mathbf{r}^n}^T & 0 & -\frac{\partial \mathbf{x}^n_{surf}}{\partial \mathbf{r}^n}^T \\ \frac{\partial \mathbf{J}^n}{\partial \mathbf{U}^n}^T & \frac{\partial \mathbf{R}^n}{\partial \mathbf{U}^n}^T & 0 \\ \frac{\partial \mathbf{J}^n}{\partial \mathbf{x}^n}^T & \frac{\partial \mathbf{R}^n}{\partial \mathbf{x}^n}^T & [K]^T \end{array} \right]^{-1} \tag{40}$$

Substituting back into the final term of the summation in the complete sensitivity equation (39) and defining a vector of adjoint variables as

$$\left\{ \begin{array}{c} \Lambda^n_{\mathbf{r}} \\ \Lambda^n_{\mathbf{U}} \\ \Lambda^n_{\mathbf{x}} \end{array} \right\} = \left[\begin{array}{ccc} \frac{\partial \mathbf{J}^n}{\partial \mathbf{r}^n}^T & 0 & -\frac{\partial \mathbf{x}^n_{surf}}{\partial \mathbf{r}^n}^T \\ \frac{\partial \mathbf{J}^n}{\partial \mathbf{U}^n}^T & \frac{\partial \mathbf{R}^n}{\partial \mathbf{U}^n}^T & 0 \\ \frac{\partial \mathbf{J}^n}{\partial \mathbf{x}^n}^T & \frac{\partial \mathbf{R}^n}{\partial \mathbf{x}^n}^T & [K]^T \end{array} \right]^{-1} \left\{ \begin{array}{c} \frac{\partial L}{\partial \mathbf{r}^n}^T \\ \frac{\partial L}{\partial \mathbf{U}^n}^T \\ \frac{\partial L}{\partial \mathbf{x}^n}^T \end{array} \right\} \tag{41}$$

we can recover a coupled linear adjoint system at time-level n as

$$\left[\begin{array}{ccc} \frac{\partial \mathbf{J}^n}{\partial \mathbf{r}^n}^T & 0 & -\frac{\partial \mathbf{x}^n_{surf}}{\partial \mathbf{r}^n}^T \\ \frac{\partial \mathbf{J}^n}{\partial \mathbf{U}^n}^T & \frac{\partial \mathbf{R}^n}{\partial \mathbf{U}^n}^T & 0 \\ \frac{\partial \mathbf{J}^n}{\partial \mathbf{x}^n}^T & \frac{\partial \mathbf{R}^n}{\partial \mathbf{x}^n}^T & [K]^T \end{array} \right] \left\{ \begin{array}{c} \Lambda^n_{\mathbf{r}} \\ \Lambda^n_{\mathbf{U}} \\ \Lambda^n_{\mathbf{x}} \end{array} \right\} = \left\{ \begin{array}{c} \frac{\partial L}{\partial \mathbf{r}^n}^T \\ \frac{\partial L}{\partial \mathbf{U}^n}^T \\ \frac{\partial L}{\partial \mathbf{x}^n}^T \end{array} \right\} \tag{42}$$

This system of equations is coupled and can be solved in a manner similar to the nonlinear coupled analysis problem. In segregated form the adjoint system can be written as

$$\left[\frac{\partial \mathbf{J}^n}{\partial \mathbf{r}^n}\right]^T \Lambda^n_{\mathbf{r}} = \frac{\partial L}{\partial \mathbf{r}^n}^T + \frac{\partial \mathbf{x}^n_{surf}}{\partial \mathbf{r}^n}^T \Lambda^n_{\mathbf{x}} \tag{43}$$

$$\left[\frac{\partial \mathbf{R}^n}{\partial \mathbf{U}^n}\right]^T \Lambda^n_{\mathbf{U}} = \frac{\partial L}{\partial \mathbf{U}^n}^T - \frac{\partial \mathbf{J}^n}{\partial \mathbf{U}^n}^T \Lambda^n_{\mathbf{r}} \tag{44}$$

$$[K]^T \Lambda^n_{\mathbf{x}} = \frac{\partial L}{\partial \mathbf{x}^n}^T - \frac{\partial \mathbf{J}^n}{\partial \mathbf{x}^n}^T \Lambda^n_{\mathbf{r}} - \frac{\partial \mathbf{R}^n}{\partial \mathbf{x}^n}^T \Lambda^n_{\mathbf{U}} \tag{45}$$

For which the solution process can be broken down as:

1. Construct right-hand-sides of equations (43) through (45) using estimates of adjoint variables.
2. Partially solve each individual disciplinary adjoint equation to obtain improved estimates for the corresponding disciplinary adjoint variables.
3. Keep repeating steps 1 and 2 until coupled adjoint system converges.

The convergence of these adjoint equations is similar to the convergence of the linear systems in the solution process of the nonlinear analysis problem since the coefficient matrices are identical with the exception of the transpose. The typical convergence of such a coupled linear adjoint system is shown in Figure (1(b)). Once the vector of adjoint variables has been obtained at time-level n, the final term in the sensitivity equation sum may be written as

$$\frac{\partial L}{\partial \mathbf{D}}^{nT} = \left\{ \begin{array}{c} -\frac{\partial \mathbf{J}^n}{\partial \mathbf{r}^{n-1}} \frac{\partial \mathbf{r}^{n-1}}{\partial \mathbf{D}} - \frac{\partial \mathbf{J}^n}{\partial \mathbf{r}^{n-2}} \frac{\partial \mathbf{r}^{n-2}}{\partial \mathbf{D}} \\ -\frac{\partial \mathbf{R}^n}{\partial \mathbf{U}^{n-1}} \frac{\partial \mathbf{U}^{n-1}}{\partial \mathbf{D}} - \frac{\partial \mathbf{R}^n}{\partial \mathbf{U}^{n-2}} \frac{\partial \mathbf{U}^{n-2}}{\partial \mathbf{D}} - \frac{\partial \mathbf{R}^n}{\partial \mathbf{x}^{n-1}} \frac{\partial \mathbf{x}^{n-1}}{\partial \mathbf{D}} - \frac{\partial \mathbf{R}^n}{\partial \mathbf{x}^{n-2}} \frac{\partial \mathbf{x}^{n-2}}{\partial \mathbf{D}} \\ \frac{\partial \mathbf{x}^n_{surf}}{\partial \mathbf{x}^0_{surf}} \frac{\partial \mathbf{x}^0_{surf}}{\partial \mathbf{D}} \end{array} \right\}^T \left\{ \begin{array}{c} \Lambda^n_{\mathbf{r}} \\ \Lambda^n_{\mathbf{U}} \\ \Lambda^n_{\mathbf{x}} \end{array} \right\} \tag{46}$$

Expanding and factoring terms with respect to state variable sensitivities at time levels $n-1$ and $n-2$ we obtain

$$\frac{\partial L}{\partial \mathbf{D}}^{nT} = \underbrace{\frac{\partial \mathbf{x}^0_{surf}}{\partial \mathbf{D}}^T \frac{\partial \mathbf{x}^n_{surf}}{\partial \mathbf{x}^0_{surf}}^T \Lambda^n_{\mathbf{x}}}_{(A)} + \underbrace{\left\{ \begin{array}{ccc} \frac{\partial \mathbf{r}^{n-1}}{\partial \mathbf{D}} & \frac{\partial \mathbf{U}^{n-1}}{\partial \mathbf{D}} & \frac{\partial \mathbf{x}^{n-1}}{\partial \mathbf{D}} \end{array} \right\} \left\{ \begin{array}{c} -\frac{\partial \mathbf{J}^n}{\partial \mathbf{r}^{n-1}}^T \Lambda^n_{\mathbf{r}} \\ -\frac{\partial \mathbf{R}^n}{\partial \mathbf{U}^{n-1}}^T \Lambda^n_{\mathbf{U}} \\ -\frac{\partial \mathbf{R}^n}{\partial \mathbf{x}^{n-1}}^T \Lambda^n_{\mathbf{U}} \end{array} \right\}}_{(B)} + \underbrace{\left\{ \begin{array}{ccc} \frac{\partial \mathbf{r}^{n-2}}{\partial \mathbf{D}} & \frac{\partial \mathbf{U}^{n-2}}{\partial \mathbf{D}} & \frac{\partial \mathbf{x}^{n-2}}{\partial \mathbf{D}} \end{array} \right\} \left\{ \begin{array}{c} -\frac{\partial \mathbf{J}^n}{\partial \mathbf{r}^{n-2}}^T \Lambda^n_{\mathbf{r}} \\ -\frac{\partial \mathbf{R}^n}{\partial \mathbf{U}^{n-2}}^T \Lambda^n_{\mathbf{U}} \\ -\frac{\partial \mathbf{R}^n}{\partial \mathbf{x}^{n-2}}^T \Lambda^n_{\mathbf{U}} \end{array} \right\}}_{(C)} \tag{47}$$

Term (A) in the above expression forms the contribution from time-level n to the total sensitivity vector $\frac{dL}{dD}^T$. The state variable sensitivity vectors that appear in terms (B) and (C) may now be combined with the corresponding vectors that appear in the summation of the total sensitivity equation (39) at time-levels $n-1$ and $n-2$. The procedure of linearizing the residual equations and constructing a coupled adjoint system can now be repeated at time levels $n-1$, $n-2$ and so forth cascading all the way to the first time-level. It is clear that constructing the complete sensitivity vector involves a recurrence relation backward in time beginning at the final time-level and ending at the first time-level. At each time level n in the recurrence, a coupled adjoint system is to be solved, and a contribution to the total sensitivity vector is computed. When the backward recurrence relation terminates at the first time step, the complete sensitivity vector $\frac{dL}{dD}^T$ is available. The general form of the system of adjoint equations to be solved at time-levels ranging between $n-2$ and the first time-level can be expressed as

$$\left[\frac{\partial \mathbf{J}^n}{\partial \mathbf{r}^n}\right]^T \Lambda^n_{\mathbf{r}} = \frac{\partial L}{\partial \mathbf{r}^n}^T + \frac{\partial \mathbf{x}^n_{surf}}{\partial \mathbf{r}^n}^T \Lambda^n_{\mathbf{x}} - \frac{\partial \mathbf{J}^{n+1}}{\partial \mathbf{r}^n}^T \Lambda^{n+1}_{\mathbf{r}} - \frac{\partial \mathbf{J}^{n+2}}{\partial \mathbf{r}^n}^T \Lambda^{n+2}_{\mathbf{r}} \tag{48}$$

$$\left[\frac{\partial \mathbf{R}^n}{\partial \mathbf{U}^n}\right]^T \Lambda^n_{\mathbf{U}} = \frac{\partial L}{\partial \mathbf{U}^n}^T - \frac{\partial \mathbf{J}^n}{\partial \mathbf{U}^n}^T \Lambda^n_{\mathbf{r}} - \frac{\partial \mathbf{R}^{n+1}}{\partial \mathbf{U}^n}^T \Lambda^{n+1}_{\mathbf{U}} - \frac{\partial \mathbf{R}^{n+2}}{\partial \mathbf{U}^n}^T \Lambda^{n+2}_{\mathbf{U}} \tag{49}$$

$$[K]^T \Lambda^n_{\mathbf{x}} = \frac{\partial L}{\partial \mathbf{x}^n}^T - \frac{\partial \mathbf{J}^n}{\partial \mathbf{x}^n}^T \Lambda^n_{\mathbf{r}} - \frac{\partial \mathbf{R}^n}{\partial \mathbf{x}^n}^T \Lambda^n_{\mathbf{U}} - \frac{\partial \mathbf{R}^{n+1}}{\partial \mathbf{x}^n}^T \Lambda^{n+1}_{\mathbf{U}} - \frac{\partial \mathbf{R}^{n+2}}{\partial \mathbf{x}^n}^T \Lambda^{n+2}_{\mathbf{U}} \tag{50}$$

where the adjoint variables at time-level n are being solved for, and the adjoint variables at $n+1$ and $n+2$ are known.

By transposing the sensitivity linearization and introducing the vector of adjoint variables at each time-level, we have eliminated the need for the state variable sensitivity matrices, and hence the direct dependence of the cost of obtaining the final sensitivity on the number of design variables. The dependence on the design variables has been pushed to a single term, namely $\frac{\partial \mathbf{x}^0_{surf}}{\partial \mathbf{D}}^T$ appearing at each time-level in the form of term (A) in equation (47). This particular matrix is essentially the linearization of the shape functions which modify the airfoil geometry with respect to the design inputs and is easily computable.

An important observation at this point is that the method described above for computing sensitivities can be generalized for any number of coupled disciplines. The functional linearization may be extended to any number of disciplines by simply linearizing with respect to the state variables of each discipline. Similarly the residual equations for each discipline may be linearized with respect to the design input vector $\mathbf{D}$ to construct expressions for the corresponding state variable sensitivity matrices. The assembly of the total sensitivity vector then involves introducing an adjoint variable per discipline at each time-level and sweeping backward in time while solving a coupled linear adjoint system at each time-level.

V. Implementation, Validation and Results

A. General implementation details

The analysis solver and the adjoint solver form two separate codes that share several subroutines in the implementation. As described in section (II), the analysis code is second-order accurate both spatially and temporally, while the adjoint code is a discretely exact linearization of all aspects of the analysis code. Both codes run in parallel on multi-core hardware architectures with commonly addressable memory using OpenMP parallelization. All results presented in the paper were run on a quad-core Intel desktop with 4 gigabytes of memory using 4 OpenMP threads. Both codes exhibit good scaling between 1,2 and 4 cores, with the adjoint code scaling slightly better than the analysis code. This is most likely due to the linear nature of the adjoint code where, once the coefficient matrices have been setup, only a single linear solution of a coupled system per time-level is required. On the contrary, the nonlinear nature of the analysis problem requires repeated construction of coefficient matrices for multiple linear solutions at each time-level. Figures (2(a)) and (2(b)) indicate the performance scaling between 1,2 and 4 cores for the analysis and adjoint problems respectively. It should also be noted that the solution of the adjoint problem typically consumes less time than the analysis problem. Again, this is attributed to the fact that only linear solutions are required in the adjoint problem while the analysis problem requires both nonlinear and linear solutions. The analysis code performs the forward integration in time and writes the solution of the coupled system at each time-level to the hard-drive, which is then read by the adjoint solver sweeping backward in time. It is necessary that the unsteady solution set be written to disk and read-in by the adjoint solver since the adjoint code involves a backward sweep in time. Holding the entire solution set in memory would quickly become impractical for large problems. However, disk I/O operations consume trivial computational resources especially when done in a piecewise manner (i.e. at each time-level) as in this case.

B. Solver validation

The first step in the validation of the analysis code is to verify that computed time-dependent load coefficients (independent of structural response) match experimental and other computational results. The problem chosen for this purpose is the AGARD test case No.5.[24] The problem involves a NACA0012 airfoil sinusoidally pitching at a transonic Mach number of $M_\infty = 0.755$. The angle-of-attack of the airfoil as a function of non-dimensional time t is defined as

$$\alpha = \alpha_0 + \alpha_{max} sin(2k_c t) \tag{51}$$

For AGARD test case No.5, the airfoil pitches around its quarter-chord with a mean angle-of-attack α_0 of 0.016^o, an amplitude of pitch α_{max} as 2.51^o at a reduced of $k_c = 0.0814$. We use a computational mesh consisting of approximately 10,000 elements, shown in Figure (3(a)), for the unsteady validation. Figures (4(a)) and (4(b)) show the time varying lift and moment coefficients computed by the solver which match well with results from references,[21,23–25] thus establishing confidence in the non-aeroelastic unsteady aspect of the solver.

The next step is to validate the aeroelastic aspect of the solver for which the two-dimensional swept wing model exhibiting the transonic dip phenomenon suggested by Isogai[26] is chosen as the test case. The structural parameters for this case are

$$\begin{aligned} x_\alpha &= 1.8 \\ r_\alpha^2 &= 3.48 \\ \omega_h, \omega_\alpha &= 100 \text{ rad/s} \\ \mu &= 60 \\ a &= -2.0 \end{aligned}$$

The quantity a is the non-dimensional elastic axis location along the chord of the airfoil measured from the mid-chord of the airfoil when it is in the neutral position. Since it is non-dimensionalized by the semi-chord of the airfoil, the elastic axis is located half a chord length ahead of the leading edge of the airfoil in this particular case. The airfoil under consideration is the NACA64A010 (Ames) airfoil operating with a mean angle-of-attack α_0 of zero and an amplitude of forced pitching α_{max} of 1^o. The computational mesh used for this case consists of approximately 6,600 elements and is shown in Figure (3(b)). The solution process involves obtaining a steady-state solution at the mean angle-of-attack and then forcing the airfoil in pitch for three periods at the natural frequency of pitch before allowing it to respond aeroelastically. The goal of the validation procedure is to compute and compare a flutter boundary against existing data. Computation of the flutter boundary is performed by manually modifying the flutter velocity at various Mach numbers with the objective of obtaining a neutral aeroelastic response. Although the aeroelastic solver was not directly validated against experimental data, the predicted flutter boundary matches well with computational results

from References[23,25] as indicated by the flutter diagram in Figure (5). The aeroelastic response of the airfoil in both pitch and plunge degrees-of-freedom at various locations (stable and unstable) in the flutter diagram are shown in Figure (6) and follow expected trends.

C. Sensitivity validation

The procedure to validate the adjoint based sensitivity is twofold. The forward linearization is typically more intuitive since it follows a framework very similar to that of the analysis code (i.e. forward in time). Issues with the discrete derivatives of the different pieces of the code contributing to the total linearization can be tracked down far more easily using the forward linearization than with the adjoint or backward linearization. For this reason, the forward linearization is first verified against finite-difference values, and then the adjoint linearization is validated by verifying the property of dual consistency[27] with respect to the forward linearization. Once the issues with the general framework of the adjoint linearization have been worked out in this manner, minor modifications to the analysis code can typically be directly carried over to the adjoint code and verified against finite-difference. Table (1) compares adjoint-based and finite-differenced sensitivity values for a few design variables, while Figure (7) shows the comparison over all design variables. The adjoint-based sensitivity values typically match to a minimum of 4 digits against finite-difference values as indicated by Table (1), and are virtually indistinguishable from one another when overlaid on a plot as shown in Figure (7).

Design Variable ID	Adjoint Sensitivity	Finite-difference
183	**6.368**45285306436	**6.368**25832067700
184	**6.7973**2288068724	**6.7973**4654118747
185	**7.490**93700072817	**7.490**87240015101
186	**8.831**56670890565	**8.831**24061656915
187	**12.261**2224955801	**12.261**3669883975
188	**19.422**2465094799	**19.422**4766714157

Table 1. Comparison of adjoint sensitivity and finite-difference (matching digits in bold font)

D. Optimization algorithm

The obvious choice for an optimization algorithm is one that is gradient based and proceeds in a direction that minimizes the objective of interest. The simplest approach is to use steepest-descent or some form of a line-search algorithm, but it has been determined in previous work[12] that these methods perform poorly on stiff nonlinear optimization problems particularly when the number of design parameters is large. The optimization example presented uses the LBFGS-B bounded reduced Hessian algorithm developed in Reference.[28] The algorithm not only uses gradient information but also builds an approximate Hessian matrix (2^{nd} derivative of the functional with respect to design parameters) on-the-fly using the optimization history in order to speed up the overall convergence to minimum. The bounded rather than the unbounded LBFGS algorithm was chosen in order to prevent the generation of degenerate geometries during the optimization process. The actual bounds themselves were established a priori by manually exploring the effect of the design variables on the shape of the airfoil and limiting them to regions which do not collapse the geometry.

E. Optimization example

The goal of the optimization example is to consider a point in the flutter diagram of the described aeroelastic test case where the airfoil exhibits divergent behavior, and to change the shape of the airfoil such that it produces a damped aeroelastic response while satisfying a constraint on the steady-state lift of the airfoil. The time-integration process consisted of 36 uniform time-steps per forced pitch cycle combined with 3 pitch cycles and 214 time-steps of the same size for the aeroelastic response to evolve. The chosen number of time-steps for the aeroelastic response evolution correspond to approximately $5\frac{1}{2}$ periods in both the pitch and plunge axes at the starting design point of the optimization. With these time-step parameters and the 6,600 element mesh described earlier, a single design cycle consisting of a forward analysis solution and a backward adjoint solution roughly cost 30 minutes of wall-time. The

objective function for the optimization was chosen to be

$$L = \mathbf{r}^{f^T}[W_1]\mathbf{r}^f + W_2(C_{Ls} - C_{L_{Starget}}) \tag{52}$$

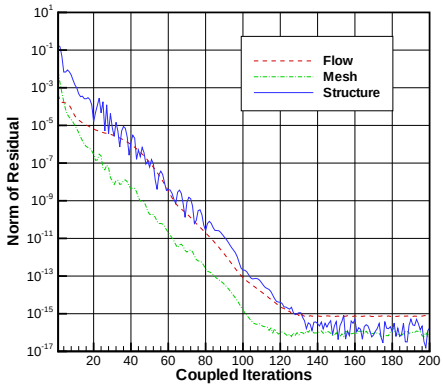
where $\mathbf{r}^f$ refers to the structural state vector at the end of the time-integration process, $[W_1]$ is a diagonal weighting matrix with equal weighting parameters of 100, W_2 is a weight on the constraint, chosen to be unity, and C_{Ls} is the steady-state lift coefficient of the airfoil. Since the airfoil is symmetric and has zero lift at zero angle-of-attack, the steady-state solution is computed at the maximum pitch angle of the forced pitch cycles. The forced unsteady pitching of the airfoil around the elastic axis begins at this point and stops at the neutral position of zero angle-of-attack (after 3 pitch cycles) at which point the aeroelastic response is allowed to evolve. The target value for the steady-state lift coefficient was taken to be that of the baseline NACA64A010 airfoil at the maximum pitch angle of 1^o. The objective function formulation when zero results in zero displacements and zero velocities in both degrees-of-freedom of pitch and plunge. Since the aeroelastic time-integration ends fairly quickly ($5\frac{1}{2}$ periods), this infact creates a stiff objective formulation. However, it is not necessary to drive the objective to zero and the optimization may be stopped once a damped response is observed and the constraint on the steady-state lift is satisfied. For the described test case, the airfoil produces a neutral response for a flutter velocity $V_f = 0.65$ at a Mach number of $M_\infty = 0.825$. An unstable point of $V_f = 0.75$ at this Mach number was chosen to be the starting design point for the optimization. Figure (8(a)) shows the aeroelastic response of the airfoil at the starting design point. The divergent behavior is clearly visible in both pitch and plunge axes. Prior to running the optimization, the time-integration for the aeroelastic response was carried out over a temporal domain that was 10 times the size of the chosen domain (214 time-steps) in order to ensure that the divergent response did not lead to limit-cycle behavior. The shape of the airfoil was controlled by an equal distribution in the chordwise direction of 16 bump functions on the upper surface and 16 bump functions on the lower surface resulting in a total of 32 design variables. As mentioned earlier, the design variables are the weights controlling the magnitudes of each one of the bump functions. Although in theory any modification of the shape of the airfoil would result in changes to the structural parameters controlling the aeroelastic response, we assume them to be constant and independent of geometry for the optimization example presented here. The optimization was terminated after 41 design iterations and Figure (8(b)) shows the aeroelastic response of the optimized airfoil. In terms of wall-time the optimization cost approximately 21 hours. The entire process required about 200 megabytes of disk space at each design iteration to hold the unsteady solution set for use by the adjoint solver which is then overwritten by the following design iteration. While the displacements and the velocities have not been driven to zero, the plot clearly indicates a rapidly decaying response. Figure (9) shows the convergence of the objective function and the L_2 norm of the sensitivity vector. During the course of the optimization, a decaying response was indeed observed after only 11 design iterations, but the optimization was not terminated at this point since the tolerance on the constraint was not satisfied. The tolerance was set to be a maximum of 0.25% deviation from the target constraint over at least two consecutive design iterations. Figure (10) indicates that the constraint satisfies the set tolerance only at 41 design iterations at which point the optimization was terminated. Figure (11) compares the optimized and the baseline airfoil geometries using 1:1 and exaggerated scales.

VI. Conclusions

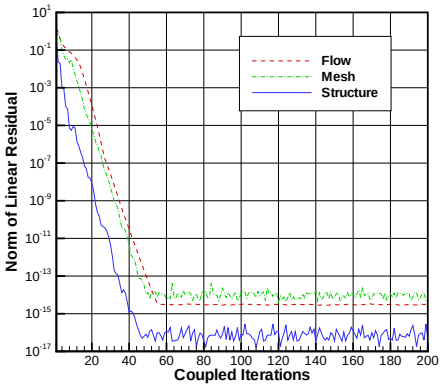
The strength of adjoint equations has been employed in computing sensitivity derivatives for the coupled unsteady fluid-structure equations. The algorithm developed was demonstrated in the context of shape optimization applied to a standard two-dimensional flutter problem. Extensions to three dimensions will require the additional step of linearizing the transfer of forces and displacements between fluids and structures grids, which is trivial for the two-dimensional case. The observed computational cost of the two-dimensional problem suggests that full three-dimensional unsteady aeroelastic optimization problems, although relatively expensive, should be tractable on current-day large parallel machines both in terms of computational time, memory, and disk space for storing the solution time history. We also note that relaxing the criterion of fully converging the coupled problem to machine precision at each time step can be expected to yield significant efficiency improvements, although this will require a quantification of the effect of coupling error on the analysis and optimization processes. Finally, the current approach should be easily extendable to computing discretely exact sensitivity derivatives for any number of coupled discplines.

References

[1]Jameson, A., "Aerodynamic Shape Optimization using the Adjoint Method," *VKI Lecture Series on Aerodynamic Drag Prediction and Reduction, von Karman Institute of Fluid Dynamics, Rhode St Genese, Belgium*, 2003.

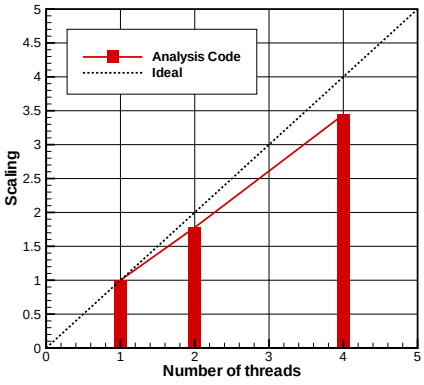


(a) Convergence of coupled analysis problem

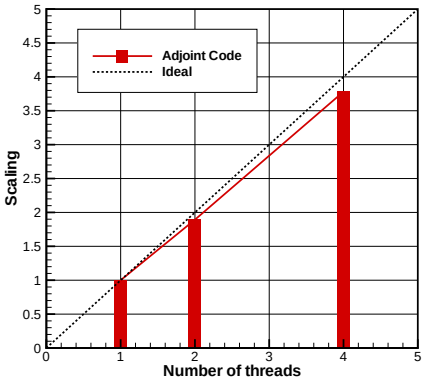


(b) Convergence of coupled adjoint problem

Figure 1. Convergence of the coupled analysis and coupled adjoint problems at a single time-level

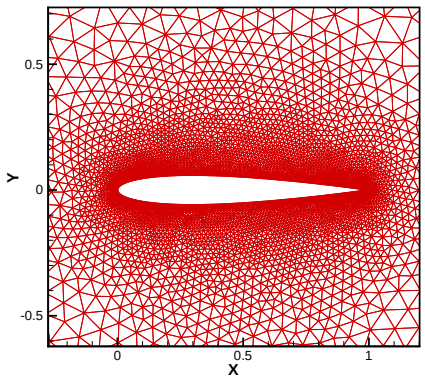


(a) Analysis Code

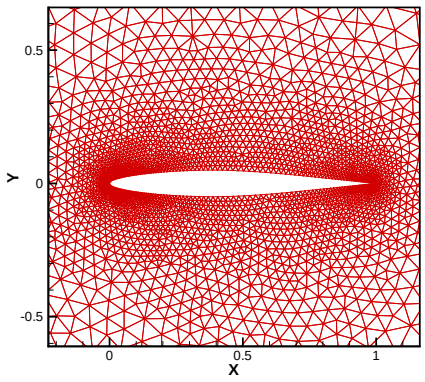


(b) Adjoint Code

Figure 2. Performance scaling of analysis and adjoint codes using OpenMP parallelization

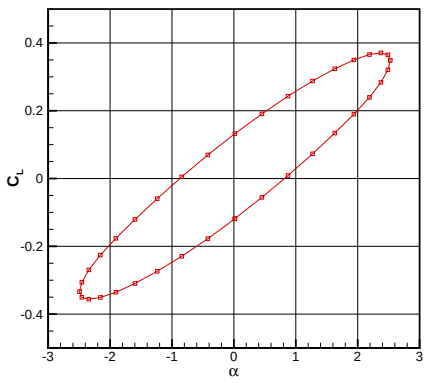


(a) NACA0012 mesh consisting of approximately 10,000 elements

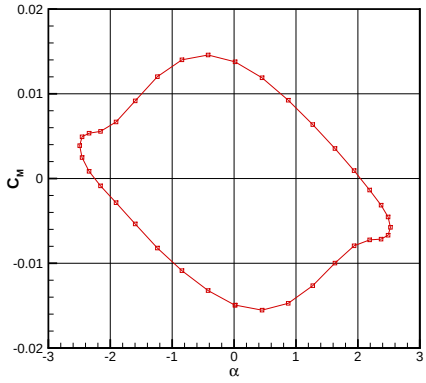


(b) NACA64A010 mesh consisting of approximately 6,600 elements

Figure 3. Computational meshes used for solver validation and optimization example



(a) Time variation of lift coefficient for unsteady solver validation



(b) Time variation of moment coefficient for unsteady solver validation

Figure 4. Unsteady load coefficients for transonic pitching NACA0012 airfoil

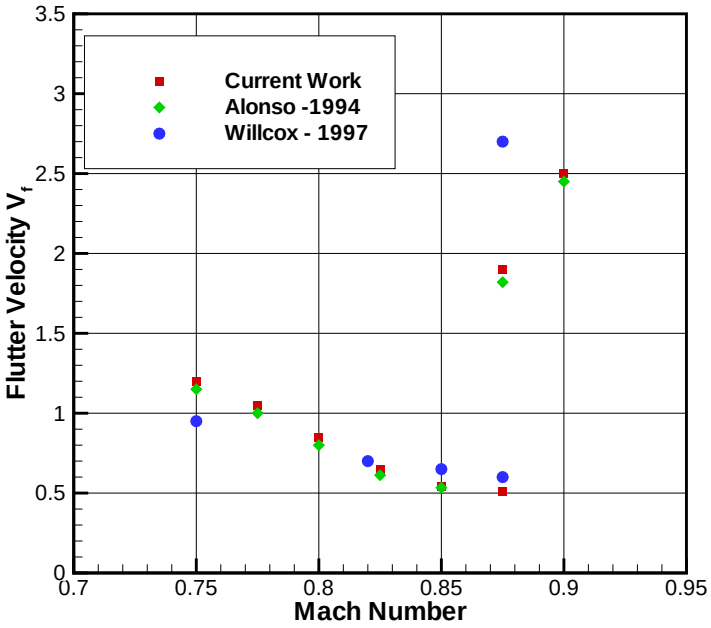


Figure 5. Comparison of predicted flutter boundary against other references

[2]Jameson, A. and Vassberg, J., "Computational Fluid Dynamics for Aerodynamic Design: Its Current and Future Impact," *Proceedings of the 39th Aerospace Sciences Meeting and Exhibit, Reno NV*, 2001, AIAA Paper 2001–0538.

[3]Jameson, A., Alonso, J., Reuther, J., Martinelli, L., and Vassberg, J., "Aerodynamic shape optimization techniques based on control theory," 1998, AIAA Paper 98–2538.

[4]Mavriplis, D. J., "Multigrid Solution of the Discrete Adjoint for Optimization Problems on Unstructured Meshes," *AIAA Journal*, Vol. 44-1, January 2006, pp. 42–50.

[5]Mavriplis, D. J., "Discrete Adjoint-Based Approach for Optimization Problems on Three-Dimensional Unstructured Meshes," *AIAA Journal*, Vol. 45-4, April 2007, pp. 741–750.

[6]Nielsen, E. and Anderson, W., "Recent Improvements in Aerodynamic Design Optimization on Unstructured Meshes," *AIAA Journal*, Vol. 40-6, June 2002, pp. 1155–1163.

[7]Elliott, J. and Peraire, J., "Practical Three-Dimensional Aerodynamic Design and Optimization Using Unstructured Meshes," *AIAA Journal*, Vol. 35-9, 1997, pp. 1479–1485.

[8]Soto, R. and Yang, C., "An Adjoint-Based Design Methodology for CFD Optimization Problems," 2003, AIAA Paper 2003–0299.

[9]Kirsch, U., *Structural Optimization: Fundamentals and Applications*, Springer, Berlin, 1993.

[10]Maute, K., Nikbay, M., and Farhat, C., "Coupled Analytical Sensitivity Analysis and Optimization of Three-Dimensional Nonlinear Aeroelastic Systems," *AIAA Journal*, Vol. 39, 2001, pp. 2051–2061.

[11]Martins, J., Alonso, J., and Reuther, J., "Aero-Structural Wing Design Optimization Using High-Fidelity Sensitivity Analysis," *CEAS Conference on Multidisciplinary Analysis and Optimization, Cologne, Germany, June 2001*, 2001.

[12]Mani, K. and Mavriplis, D. J., "Unsteady Discrete Adjoint Formulation for Two-Dimensional Flow Problems with Deforming Meshes," *AIAA Journal*, Vol. 46-6, June 2008, pp. 1351–1364.

[13]Mavriplis, D. J., "Solution of the Unsteady Discrete Adjoint for Three-Dimensional Problems on Dynamically Deforming Unstructured Meshes," *Proceedings of the 46th Aerospace Sciences Meeting and Exhibit, Reno NV*, 2008, AIAA Paper 2008–0727.

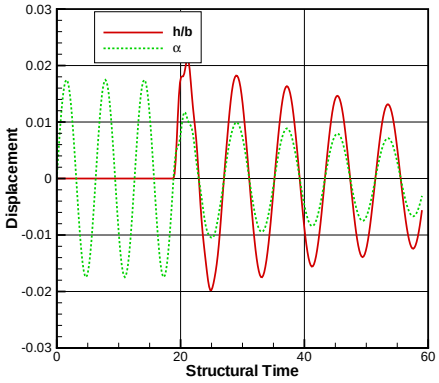
[14]Rumpfkeil, M. and Zingg, D., "A General Framework for the Optimal Control of Unsteady Flows with Applications," *45th AIAA Aerospace Sciences Meeting and Exhibit, Reno, NV, January*, 2007, AIAA Paper 2007–1128.

[15]Nadarajah, S. and Jameson, A., "Optimal Control of Unsteady Flows using A Time Accurate Method," *Proceedings of the 9th AIAA/ISSMO Symposium on Multidisciplinary Analysis and Optimization Conference, Atlanta GA*, 2002, AIAA Paper 2002–5436.

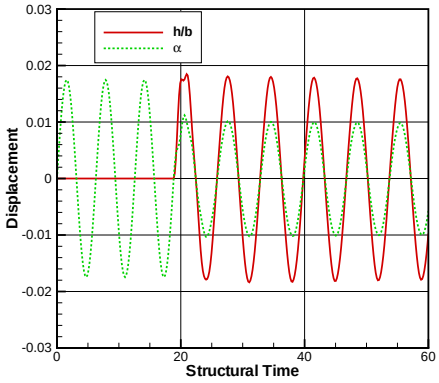
[16]Ghayour, K. and Baysal, O., "Limit-Cycle Shape Optimization using Time-Dependent Transonic Equation," *Proceedings of the 14th Computational Fluid Dynamics Conference, Norfolk VA*, 1999, AIAA Paper 99–3375.

[17]Mavriplis, D. J., "Unstructured-Mesh Discretizations and Solvers for Computational Aerodynamics," *AIAA Journal*, Vol. 46-6, June 2008, pp. 1281–1298.

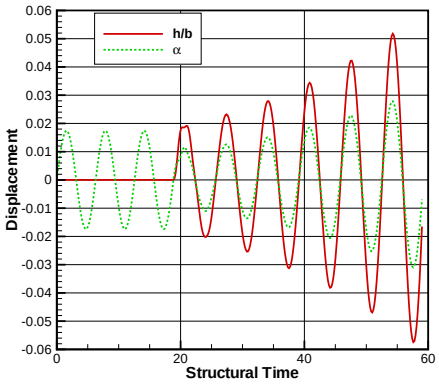
[18]Geuzaine, P., Grandmont, C., and Farhat, C., "Design and Analysis of ALE Schemes with Provable Second-Order Time-Accuracy for Inviscid and Viscous Flow Simulations," *Journal of Computational Physics*, Vol. 191, 2003, pp. 206–227.



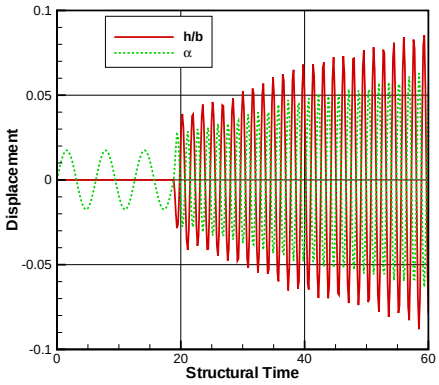
(a) Damped response ($M_\infty = 0.82, V_f = 0.4$)



(b) Neutral response ($M_\infty = 0.82, V_f = 0.71$)



(c) Divergent response ($M_\infty = 0.85, V_f = 0.8$)



(d) 2^{nd} mode response ($M_\infty = 0.875, V_f = 2.6$)

Figure 6. Aeroelastic response

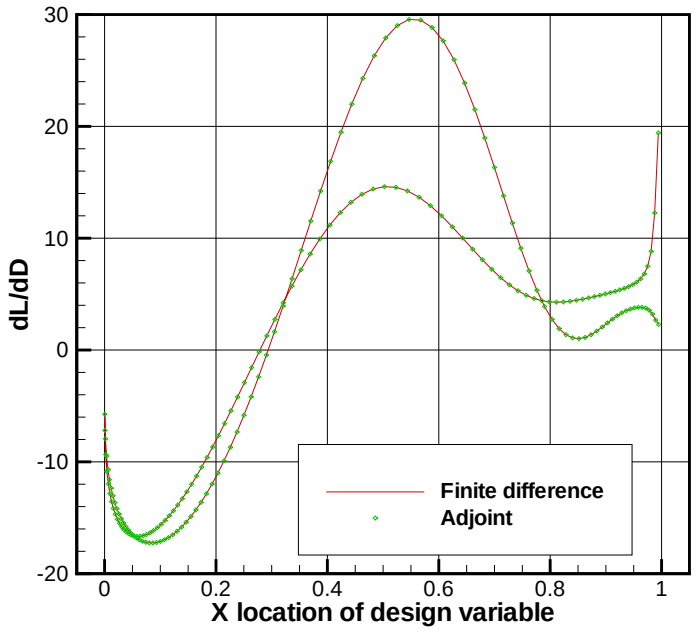
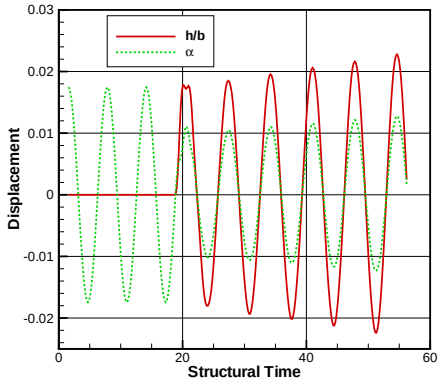
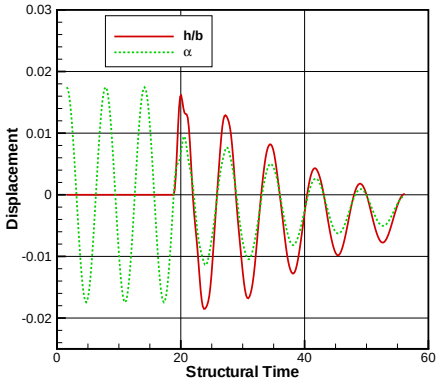


Figure 7. Comparison of gradients computed by linearization and finite-differencing



(a) Diverging aeroelastic response of baseline airfoil



(b) Decaying aeroelastic response of optimized airfoil

Figure 8. Aeroelastic response

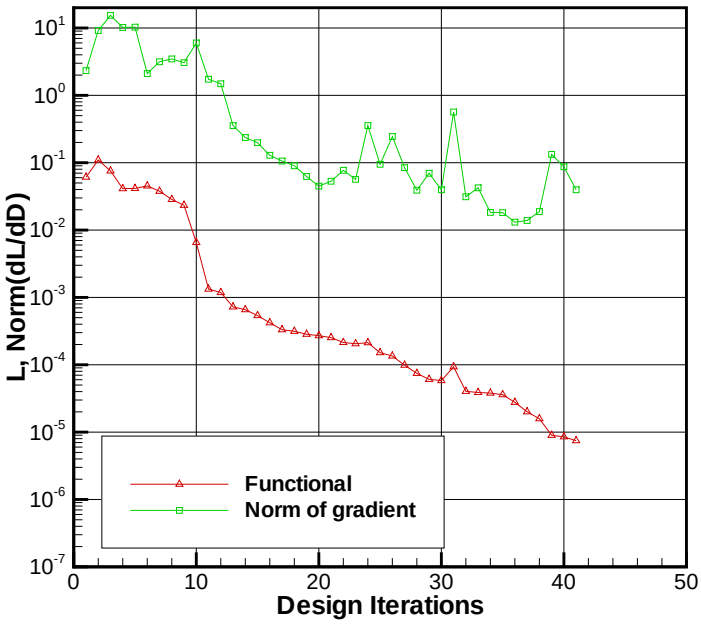


Figure 9. Convergence of objective function *L* and L2 norm of gradient vector

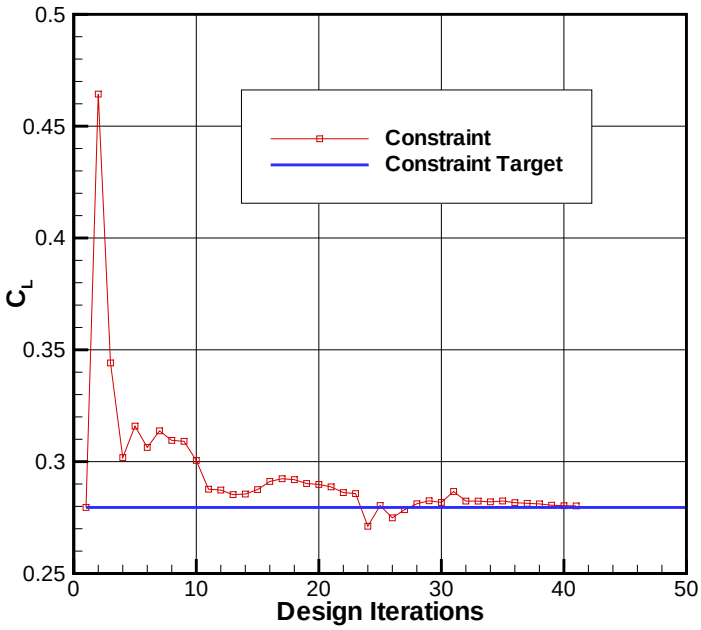
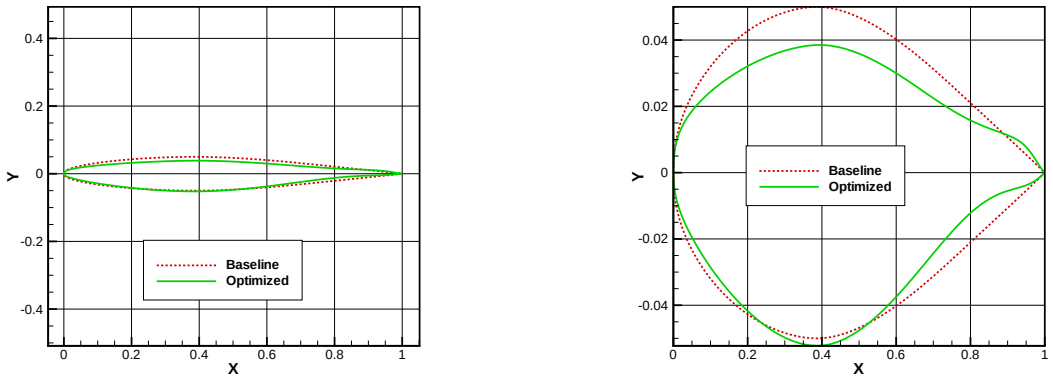


Figure 10. Convergence of constraint on objective function



(a) 1:1 scale

(b) Exaggerated scale

Figure 11. Comparison of baseline and optimized airfoils

[19]Yang, Z. and Mavriplis, D. J., "Higher-Order Time Integration Schemes for Aeroelastic Applications on Unstructured Meshes," *AIAA Journal*, Vol. 45-1, January 2007, pp. 138–150.

[20]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.

[21]Batina, J. T., "Unsteady Euler Airfoil Solutions Using Unstructured Dynamic Meshes," *AIAA Journal*, Vol. 28-8, August 1990, pp. 1381–1388.

[22]Hicks, R. and Henne, P., "Wing Design by Numerical Optimization," *Journal of Aircraft*, Vol. 15-7, 1978, pp. 407–412.

[23]Willcox, K. and Peraire, J., "Aeroelastic computations in the time domain using unstructured meshes," *International Journal for Numerical Methods in Engineering*, Vol. 40-13, 1997.

[24]AGARD, *Compendium of Unsteady Aerodynamic Measurements*, Report No.702, 1982.

[25]Alonso, J. J. and Jameson, A., "Fully-Implicit Time-Marching Aeroelastic Solutions," *32nd AIAA Aerospace Sciences Meeting and Exhibit, Reno, NV, January*, 1994, AIAA Paper 94–0056.

[26]Isogai, K., "Transonic Dip Mechanism of Flutter of a Sweptback Wing: Part II," *AIAA Journal*, Vol. 17, 1981, pp. 1240–1242.

[27]Giles, M., Duta, M., and Muller, J., "Adjoint Code Developments Using Exact Discrete Approach," 2001, AIAA Paper 2001–2596.

[28]Byrd, R. H., Lu, P., and Nocedal, J., "A Limited Memory Algorithm for Bound Constrained Optimization," *SIAM Journal on Scientific and Statistical Computing*, Vol. 16,5, 1995, pp. 1190–1208.