AIAA JOURNAL
Vol. 47, No. 8, August 2009

Adjoint-Based Sensitivity Formulation for Fully Coupled Unsteady Aeroelasticity Problems

Karthik Mani[*] and Dimitri J. Mavriplis[†]
University of Wyoming, Laramie, Wyoming 82071-3295

DOI: 10.2514/1.40582

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 to demonstrate the proof of concept.

Nomenclature

A	=	control volume
a	=	nondimensional elastic axis location along the airfoil chord
$a_{\max}$	=	weight-controlling magnitude of the bump function
B	=	surface bounding a control volume
b	=	semichord of the airfoil
b_i	=	displacement in the y direction provided by the bump function
C_l, C_m	=	section lift coefficient and moment coefficient about the elastic axis
c	=	airfoil chord length
$\mathbf{D}$	=	vector of design variables
E_t	=	total energy
$\mathbf{F}$	=	Cartesian inviscid flux vector
$\mathbf{F}_e^{\perp}$	=	normal flux across an edge e
h	=	vertical displacement (positive downward)
I_α	=	sectional moment of inertia of the airfoil
$\mathbf{J}$	=	structural residual
$[\mathbf{K}]$	=	mesh stiffness matrix
K_h	=	plunging spring coefficient
K_α	=	pitching spring coefficient
k_c	=	reduced frequency defined as $\omega_\alpha c/2U_\infty$
L	=	objective functional
L_a	=	sectional lift of the airfoil
M_{ea}	=	sectional moment of the airfoil about the elastic axis (positive nose up)
m	=	mass of the airfoil
$\mathbf{n}$	=	unit normal vector of the face or edge
$[\mathbf{P}]$	=	preconditioner for defect correction
p	=	fluid pressure
$\mathbf{R}$	=	flow residual
$\mathbf{r}$	=	structural state vector
r_α^2	=	structural parameter defined as I_α/mb^2
S	=	surface integral of inviscid flux over a closed control volume
S_α	=	static imbalance
$[T]$	=	matrix of eigenvectors of the flow Jacobian
t	=	time
t_b	=	parameter controlling width of the bump function
$\mathbf{U}$	=	vector of conserved flow state variables
U_∞	=	freestream velocity
u, v	=	Cartesian fluid velocity components
$V^{\perp}$	=	fluid velocity normal to an edge
V_e	=	physical velocity of an edge e in the normal direction
V_f	=	flutter velocity defined as $U_\infty/\omega_\alpha b\sqrt{\mu}$
W_1, W_2	=	functional weights
$\mathbf{x}$	=	vector of two-dimensional mesh coordinates
$\dot{\mathbf{x}}$	=	vector of mesh edge velocity components
$\mathbf{x}_{\mathrm{int}}$	=	vector of two-dimensional interior mesh coordinates
x_{M_i}	=	x location at which the bump is centered
$\mathbf{x}_{\mathrm{surf}}$	=	vector of two-dimensional surface or boundary mesh coordinates
x_α	=	structural parameter defined as S_α/mb
α	=	angle of attack
$\alpha_{\max}$	=	amplitude of pitch
α_0	=	mean angle of attack
Δt	=	time step for flow equations
$\Delta\tau$	=	time step for structural equations
γ	=	ratio of specific heats
$\kappa^{(4)}$	=	parameter for controlling artificial dissipation
$\Lambda_{\mathbf{U}}$, $\Lambda_{\mathbf{x}}$, $\Lambda_{\mathbf{r}}$	=	flow, mesh, and structure adjoint variables
$[\lambda]$	=	diagonal matrix of eigenvalues of the flow Jacobian
μ	=	airfoil mass ratio defined as $m/\pi\rho b^2$
ρ	=	fluid density
ω_h, ω_α	=	uncoupled natural frequencies of plunge and pitch
$(\nabla^2\mathbf{U})$	=	undivided Laplacian of $\mathbf{U}$

Superscript

n	=	time-step index

I. Introduction

WITH advances in computational power, large-scale aeroelastic simulations in computational fluid dynamics are becoming more and more commonplace. This paper is concerned with

Presented as Paper 6242 at the 26th AIAA Applied Aerodynamics Conference, Honolulu, HI, 18–21 August 2008; received 19 September 2008; revision received 25 March 2009; accepted for publication 21 April 2009.

[*]Graduate Student, Department of Mechanical Engineering; kmani@uwyo.edu. Member AIAA.

[†]Professor, Department of Mechanical Engineering; email: mavripl@uwyo.edu. Associate Fellow AIAA.

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 lies 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]. Although adjoint methods have only recently 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, independently 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, because 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 uses 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. Although 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.

II. Analysis Problem Formulation

A. Equations of the Flow Problem in Arbitrary Lagrangian–Eulerian Form

The conservative form of the Euler equations is used in solving the flow problem. The paper is limited to inviscid flow problems, because 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

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

For an ideal gas, the equation of state relates the pressure to total energy by

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

where $\gamma = 1.4$ is the ratio of specific heats. Integrating Eq. (1) over a moving control volume $A(t)$ that is bounded by the control surface $B(t)$ and then applying the divergence theorem yields

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

Using Leibnitz's rule for differentiating integrals,

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

Equation (4) is rewritten as

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

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

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

This is the arbitrary Lagrangian–Eulerian (ALE) finite volume form of the Euler equations. The equations are required in ALE form, because the problem involves deforming meshes in which 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 Eq. (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 in which the inviscid flux integral S around a closed control volume is discretized as

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

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 $F^{\perp}_e$ 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 next:

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

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

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

The matrix $[\lambda]$ is diagonal and consists of the eigenvalues (adjusted by normal edge velocity V_e) of the flux Jacobian matrix $\partial\mathbf{F}^\perp/\partial\mathbf{U}$, 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 and, more important, eliminates the need for nondifferentiable limiters. 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 $u\hat{n}_x + v\hat{n}_y$, with $\hat{n}_x$ and $\hat{n}_y$ being the unit edge normal vector components.

D. Discrete Geometric Conservation Law

The discrete geometric conservation law (GCL) requires that a uniform flowfield be preserved when Eq. (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 Eq. (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 Eq. (7) results in the mathematical description of the GCL, as stated next [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 Eqs. (8) and (9) as

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

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 Eq. (14) is satisfied. Although 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 schemes has been developed in [20]. This formulation is used exclusively in the current work. Although 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 interconnected 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

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

where $[\mathbf{K}]$ 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. To ensure smooth geometry shapes, any displacement of a surface node is controlled by a bump function that influences neighboring nodes with an effect that diminishes moving away from the displaced node. The bump function used for the work presented here is the Hicks–Henne sine bump function [22]. The design variables or inputs for the optimization examples presented form a vector of weights controlling the magnitude of bump functions placed at various chordwise locations. The bump function, although a function of x, does not have any influence in the x direction. The Hicks–Henne bump functions are defined as

$$b_i(x) = a_{\max} \cdot \sin^{t_b}(\pi x^{m_i}) \tag{16}$$

$$m_i = \ln(0.5)/\ln(x_{M_i}) \tag{17}$$

where x_{M_i} refers to the location at which the bump is to be placed, $b_i(x)$ are the displacements of points with x coordinates ranging from 0 to 1 as a result of the bump placed at x_{M_i} in accordance with the sine bump function, $a_{\max}$ is the magnitude of the bump placed at x_{M_i}, and t_b is a parameter that controls the width of the bump. In most situations, a value of 4 is used for t_b. The bump function is valid in the range of $0 \leq x, x_{M_i} < 1$ and is ideally suited for airfoil problems. For geometries extending beyond this range, the bump function may be mapped locally over some predetermined radius about the surface node of interest. The superposition of bump functions is used to augment the existing shape of the baseline airfoil to deform its surface.

G. Aeroelastic Structural Equations

The aeroelastic model is based on the response of an airfoil with 2 degrees of freedom: namely, pitch and plunge. Referring to Fig. 1, the equations of motion for such a system can be summarized as

$$m\ddot{h} + S_\alpha\ddot{\alpha} + K_h h = -L_a \tag{18}$$

$$S_\alpha\ddot{h} + I_\alpha\ddot{\alpha} + K_\alpha\alpha = M_{\text{ea}} \tag{19}$$

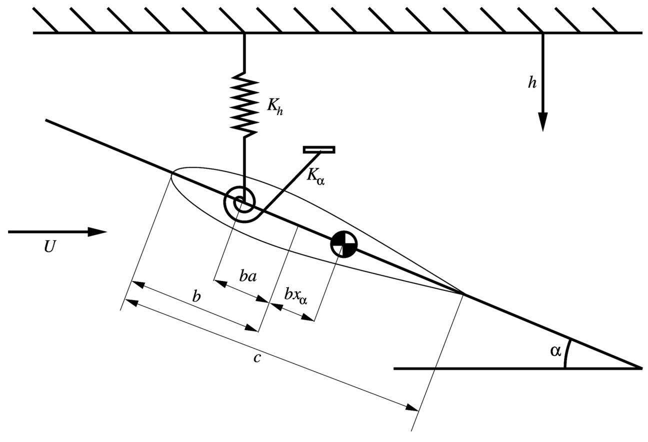


Fig. 1 Illustration of the 2-D aeroelastic model of an airfoil with 2 degrees of freedom (pitch and plunge).

nondimensionalizing Eqs. (18) and (19) and using the uncoupled natural pitch frequency ω_α for nondimensionalizing time yields

$$[M]\{\ddot{q}\} + [K]\{q\} = \{F\} \tag{20}$$

where

$$[M] = \begin{bmatrix} 1 & x_\alpha \\ x_\alpha & r_\alpha^2 \end{bmatrix}, \qquad [K] = \begin{bmatrix} \left(\frac{\omega_h}{\omega_\alpha}\right)^2 & 0 \\ 0 & r_\alpha^2 \end{bmatrix}$$
$$\{F\} = \frac{1}{\pi\mu k_c^2}\begin{Bmatrix} -C_l \\ 2C_m \end{Bmatrix}, \qquad \{q\} = \begin{Bmatrix} \frac{h}{b} \\ \alpha \end{Bmatrix} \tag{21}$$

are the nondimensional mass matrix, stiffness matrix, load vector, and displacement vector. The reduced frequency k_c is typically written in terms of the flutter velocity V_f as

$$k_c = \frac{\omega_\alpha c}{2U_\infty} = \frac{1}{V_f\sqrt{\mu}} \tag{22}$$

H. Transformation of Structural Equations

The aeroelastic equations as shown in Eq. (20) are second-order partial differential equations. These are transformed into two first-order linear equations to facilitate a direct solution procedure. The transformation is given as [23]

$$\mathbf{r}_1 = \{q\}, \qquad \mathbf{r}_2 = \dot{\mathbf{r}}_1 \tag{23}$$

The resulting first-order equations are then

$$\dot{\mathbf{r}}_1 = \mathbf{r}_2, \qquad \dot{\mathbf{r}}_2 = -[M]^{-1}[K]\mathbf{r}_1 + [M]^{-1}\{F\} \tag{24}$$

or, in matrix notation, as

$$\begin{Bmatrix} \dot{\mathbf{r}}_1 \\ \dot{\mathbf{r}}_2 \end{Bmatrix} = \begin{bmatrix} 0 & [I] \\ -[M]^{-1}[K] & 0 \end{bmatrix}\begin{Bmatrix} \mathbf{r}_1 \\ \mathbf{r}_2 \end{Bmatrix} + \begin{Bmatrix} 0 \\ [M]^{-1}\{F\} \end{Bmatrix} \tag{25}$$

$$\dot{\mathbf{r}} = [\Psi]\mathbf{r} + \{\Phi\} \tag{26}$$

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 nondimensional time step used for the flow equations. This is due to the nondimensionalization 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

$$\mathbf{R}^n = \frac{\partial A\mathbf{U}}{\partial t} + S^n(\mathbf{U}^n, V_e^n, \mathbf{n}^n) = \mathbf{0} \tag{27}$$

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

$$\mathbf{R}^n(\mathbf{U}^n, \mathbf{U}^{n-1}, \mathbf{U}^{n-2}, \mathbf{x}^n, \mathbf{x}^{n-1}, \mathbf{x}^{n-2}) = 0 \tag{28}$$

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

$$\left[\frac{\partial \mathbf{R}^k}{\partial \mathbf{U}^k}\right]\delta\mathbf{U}^k = -\mathbf{R}^k, \qquad \mathbf{U}^{k+1} = \mathbf{U}^k + \delta\mathbf{U}^k, \qquad \delta\mathbf{U}^k \to 0$$
$$\mathbf{U}^n = \mathbf{U}^k \tag{29}$$

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

$$\mathbf{J}^n = \frac{\partial \mathbf{r}}{\partial \tau} - [\Psi]\mathbf{r}^n - \{\Phi\} = 0 \tag{30}$$

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

$$\mathbf{J}^n(\mathbf{r}^n, \mathbf{r}^{n-1}, \mathbf{r}^{n-2}, \mathbf{U}^n, \mathbf{x}^n) = 0 \tag{31}$$

The corresponding Newton scheme is then

$$\left[\frac{\partial \mathbf{J}^k}{\partial \mathbf{r}^k}\right]\delta\mathbf{r}^k = -\mathbf{J}^k \tag{32}$$

Although the preceding equation is derived from a Newton scheme, the solution can be obtained linearly because 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

$$[K]\delta\mathbf{x}^n = \delta\mathbf{x}^n_{\text{surf}}, \qquad \delta\mathbf{x}^n = \mathbf{x}^n - \mathbf{x}^0, \qquad \delta\mathbf{x}^n_{\text{surf}} = \mathbf{x}^n_{\text{surf}} - \mathbf{x}^0_{\text{surf}} \tag{33}$$

where

$$\mathbf{x}^n_{\text{surf}} = f(\mathbf{x}^0_{\text{surf}}, \mathbf{r}^n) \tag{34}$$

Here, $\mathbf{x}^0_{\text{surf}}$ refers to the baseline airfoil coordinates at the neutral position, and in the case of an optimization, it refers to the shape of the current optimum airfoil, and $\mathbf{x}^0$ refers to the baseline interior mesh coordinates. The coupling between the fluid and structure equations is now apparent, because 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 Eq. (32).

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 Eq. (33).

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, and 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 with 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 Fig. 2.

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

$$\frac{dL}{d\mathbf{D}} = \sum_{n=1}^{n_{\text{steps}}} \left\{ \frac{\partial L}{\partial \mathbf{r}^n}\frac{\partial \mathbf{r}^n}{\partial \mathbf{D}} + \frac{\partial L}{\partial \mathbf{U}^n}\frac{\partial \mathbf{U}^n}{\partial \mathbf{D}} + \frac{\partial L}{\partial \mathbf{x}^n}\frac{\partial \mathbf{x}^n}{\partial \mathbf{D}} \right\} = \sum_{n=1}^{n_{\text{steps}}} \left\{ \frac{\partial L}{\partial \mathbf{D}} \right\}^n \tag{35}$$

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, in which the only nonzero term in the summation would be at $n = n_{\text{steps}}$. Each term in the sum can be written in the form of an inner product of two vectors as

$$\frac{\partial L}{\partial \mathbf{D}}^n = \left\{ \frac{\partial L}{\partial \mathbf{r}^n} \quad \frac{\partial L}{\partial \mathbf{U}^n} \quad \frac{\partial L}{\partial \mathbf{x}^n} \right\} \left\{ \frac{\partial \mathbf{r}^n}{\partial \mathbf{D}}^T \quad \frac{\partial \mathbf{U}^n}{\partial \mathbf{D}}^T \quad \frac{\partial \mathbf{x}^n}{\partial \mathbf{D}}^T \right\}^T \tag{36}$$

The elements in the first vector of the inner product shown in Eq. (36) 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., $\mathbf{r}$, $\mathbf{U}$, and $\mathbf{x}$) are vectors and easily computable because L is a scalar quantity. However, the linearization of the state variables with respect to the design inputs are matrices. To determine convenient expressions for the state variable sensitivity matrices, we linearize the corresponding residual equations. Considering the final time step in the summation ($n = n_{\text{steps}}$), 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

$$\frac{\partial \mathbf{J}^n}{\partial \mathbf{r}^n}\frac{\partial \mathbf{r}^n}{\partial \mathbf{D}} + \frac{\partial \mathbf{J}^n}{\partial \mathbf{r}^{n-1}}\frac{\partial \mathbf{r}^{n-1}}{\partial \mathbf{D}} + \frac{\partial \mathbf{J}^n}{\partial \mathbf{r}^{n-2}}\frac{\partial \mathbf{r}^{n-2}}{\partial \mathbf{D}} + \frac{\partial \mathbf{J}^n}{\partial \mathbf{U}^n}\frac{\partial \mathbf{U}^n}{\partial \mathbf{D}} + \frac{\partial \mathbf{J}^n}{\partial \mathbf{x}^n}\frac{\partial \mathbf{x}^n}{\partial \mathbf{D}} = 0 \tag{37}$$

$$\frac{\partial \mathbf{R}^n}{\partial \mathbf{U}^n}\frac{\partial \mathbf{U}^n}{\partial \mathbf{D}} + \frac{\partial \mathbf{R}^n}{\partial \mathbf{U}^{n-1}}\frac{\partial \mathbf{U}^{n-1}}{\partial \mathbf{D}} + \frac{\partial \mathbf{R}^n}{\partial \mathbf{U}^{n-2}}\frac{\partial \mathbf{U}^{n-2}}{\partial \mathbf{D}} + \frac{\partial \mathbf{R}^n}{\partial \mathbf{x}^n}\frac{\partial \mathbf{x}^n}{\partial \mathbf{D}} + \frac{\partial \mathbf{R}^n}{\partial \mathbf{x}^{n-1}}\frac{\partial \mathbf{x}^{n-1}}{\partial \mathbf{D}} + \frac{\partial \mathbf{R}^n}{\partial \mathbf{x}^{n-2}}\frac{\partial \mathbf{x}^{n-2}}{\partial \mathbf{D}} = 0 \tag{38}$$

$$[\mathbf{K}]\frac{\partial \mathbf{x}^n}{\partial \mathbf{D}} - \frac{\partial \mathbf{x}^n_{\text{surf}}}{\partial \mathbf{r}^n}\frac{\partial \mathbf{r}^n}{\partial \mathbf{D}} - \frac{\partial \mathbf{x}^n_{\text{surf}}}{\partial \mathbf{x}^0_{\text{surf}}}\frac{\partial \mathbf{x}^0_{\text{surf}}}{\partial \mathbf{D}} = 0 \tag{39}$$

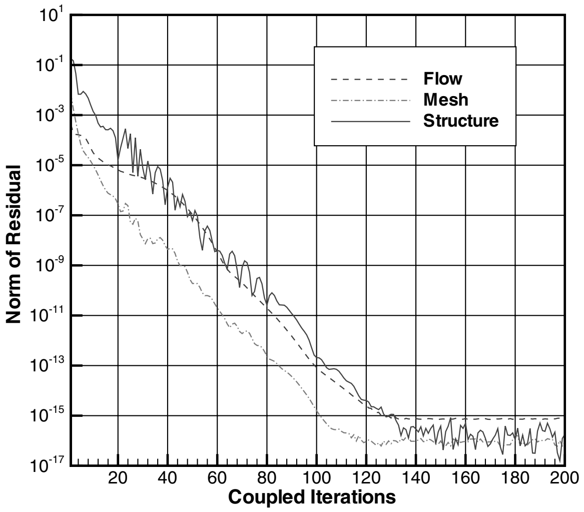


Fig. 2 Convergence of coupled analysis problem at a single time level.

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

$$\begin{bmatrix} \frac{\partial \mathbf{J}^n}{\partial \mathbf{r}^n} & \frac{\partial \mathbf{J}^n}{\partial \mathbf{U}^n} & \frac{\partial \mathbf{J}^n}{\partial \mathbf{x}^n} \\ 0 & \frac{\partial \mathbf{R}^n}{\partial \mathbf{U}^n} & \frac{\partial \mathbf{R}^n}{\partial \mathbf{x}^n} \\ -\frac{\partial \mathbf{x}^n_{\text{surf}}}{\partial \mathbf{r}^n} & 0 & [\mathbf{K}] \end{bmatrix} \begin{Bmatrix} \frac{\partial \mathbf{r}^n}{\partial \mathbf{D}} \\ \frac{\partial \mathbf{U}^n}{\partial \mathbf{D}} \\ \frac{\partial \mathbf{x}^n}{\partial \mathbf{D}} \end{Bmatrix} = \begin{Bmatrix} -\frac{\partial \mathbf{J}^n}{\partial \mathbf{r}^{n-1}}\frac{\partial \mathbf{r}^{n-1}}{\partial \mathbf{D}} - \frac{\partial \mathbf{J}^n}{\partial \mathbf{r}^{n-2}}\frac{\partial \mathbf{r}^{n-2}}{\partial \mathbf{D}} \\ -\frac{\partial \mathbf{R}^n}{\partial \mathbf{U}^{n-1}}\frac{\partial \mathbf{U}^{n-1}}{\partial \mathbf{D}} - \frac{\partial \mathbf{R}^n}{\partial \mathbf{U}^{n-2}}\frac{\partial \mathbf{U}^{n-2}}{\partial \mathbf{D}} - \frac{\partial \mathbf{R}^n}{\partial \mathbf{x}^{n-1}}\frac{\partial \mathbf{x}^{n-1}}{\partial \mathbf{D}} - \frac{\partial \mathbf{R}^n}{\partial \mathbf{x}^{n-2}}\frac{\partial \mathbf{x}^{n-2}}{\partial \mathbf{D}} \\ \frac{\partial \mathbf{x}^n_{\text{surf}}}{\partial \mathbf{x}^0_{\text{surf}}}\frac{\partial \mathbf{x}^0_{\text{surf}}}{\partial \mathbf{D}} \end{Bmatrix} \tag{40}$$

Equation (40) 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, because 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, in which 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 (35). The transposed sensitivity equation can be written as

$$\frac{dL}{d\mathbf{D}}^T = \sum_{n=1}^{n_{\text{steps}}} \left\{ \frac{\partial \mathbf{r}^n}{\partial \mathbf{D}} \quad \frac{\partial \mathbf{U}^n}{\partial \mathbf{D}} \quad \frac{\partial \mathbf{x}^n}{\partial \mathbf{D}} \right\} \left\{ \frac{\partial L}{\partial \mathbf{r}^n} \quad \frac{\partial L}{\partial \mathbf{U}^n} \quad \frac{\partial L}{\partial \mathbf{x}^n} \right\}^T \tag{41}$$

Transposing and rearranging Eq. (40) to obtain an expression for the state variable sensitivities, we get

$$\left\{ \frac{\partial \mathbf{r}^n}{\partial \mathbf{D}} \quad \frac{\partial \mathbf{U}^n}{\partial \mathbf{D}} \quad \frac{\partial \mathbf{x}^n}{\partial D} \right\} = \begin{Bmatrix} -\frac{\partial \mathbf{J}^n}{\partial \mathbf{r}^{n-1}}\frac{\partial \mathbf{r}^{n-1}}{\partial \mathbf{D}} - \frac{\partial \mathbf{J}^n}{\partial \mathbf{r}^{n-2}}\frac{\partial \mathbf{r}^{n-2}}{\partial \mathbf{D}} \\ -\frac{\partial \mathbf{R}^n}{\partial \mathbf{U}^{n-1}}\frac{\partial \mathbf{U}^{n-1}}{\partial \mathbf{D}} - \frac{\partial \mathbf{R}^n}{\partial \mathbf{U}^{n-2}}\frac{\partial \mathbf{U}^{n-2}}{\partial \mathbf{D}} - \frac{\partial \mathbf{R}^n}{\partial \mathbf{x}^{n-1}}\frac{\partial \mathbf{x}^{n-1}}{\partial \mathbf{D}} - \frac{\partial \mathbf{R}^n}{\partial \mathbf{x}^{n-2}}\frac{\partial \mathbf{x}^{n-2}}{\partial \mathbf{D}} \\ \frac{\partial \mathbf{x}^n_{\text{surf}}}{\partial \mathbf{x}^0_{\text{surf}}}\frac{\partial \mathbf{x}^0_{\text{surf}}}{\partial \mathbf{D}} \end{Bmatrix}^T \times \begin{bmatrix} \frac{\partial \mathbf{J}^n}{\partial \mathbf{r}^n}^T & 0 & -\frac{\partial \mathbf{x}^n_{\text{surf}}}{\partial \mathbf{r}^n}^T \\ \frac{\partial \mathbf{J}^n}{\partial \mathbf{U}^n}^T & \frac{\partial \mathbf{R}^n}{\partial \mathbf{U}^n}^T & 0 \\ \frac{\partial \mathbf{J}^n}{\partial \mathbf{x}^n}^T & \frac{\partial \mathbf{R}^n}{\partial \mathbf{x}^n}^T & [\mathbf{K}]^T \end{bmatrix}^{-1} \tag{42}$$

Substituting back into the final term of the summation in the complete sensitivity equation (41) and defining a vector of adjoint variables as

$$\begin{Bmatrix} \Lambda^n_{\mathbf{r}} \\ \Lambda^n_{\mathbf{U}} \\ \Lambda^n_{\mathbf{x}} \end{Bmatrix} = \begin{bmatrix} \frac{\partial \mathbf{J}^n}{\partial \mathbf{r}^n}^T & 0 & -\frac{\partial \mathbf{x}^n_{\text{surf}}}{\partial \mathbf{r}^n}^T \\ \frac{\partial \mathbf{J}^n}{\partial \mathbf{U}^n}^T & \frac{\partial \mathbf{R}^n}{\partial \mathbf{U}^n}^T & 0 \\ \frac{\partial \mathbf{J}^n}{\partial \mathbf{x}^n}^T & \frac{\partial \mathbf{R}^n}{\partial \mathbf{x}^n}^T & [\mathbf{K}]^T \end{bmatrix}^{-1} \begin{Bmatrix} \frac{\partial L}{\partial \mathbf{r}^n}^T \\ \frac{\partial L}{\partial \mathbf{U}^n}^T \\ \frac{\partial L}{\partial \mathbf{x}^n}^T \end{Bmatrix} \tag{43}$$

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

$$\begin{bmatrix} \frac{\partial \mathbf{J}^n}{\partial \mathbf{r}^n}^T & 0 & -\frac{\partial \mathbf{x}^n_{\text{surf}}}{\partial \mathbf{r}^n}^T \\ \frac{\partial \mathbf{J}^n}{\partial \mathbf{U}^n}^T & \frac{\partial \mathbf{R}^n}{\partial \mathbf{U}^n}^T & 0 \\ \frac{\partial \mathbf{J}^n}{\partial \mathbf{x}^n}^T & \frac{\partial \mathbf{R}^n}{\partial \mathbf{x}^n}^T & [\mathbf{K}]^T \end{bmatrix} \begin{Bmatrix} \Lambda^n_{\mathbf{r}} \\ \Lambda^n_{\mathbf{U}} \\ \Lambda^n_{\mathbf{x}} \end{Bmatrix} = \begin{Bmatrix} \frac{\partial L}{\partial \mathbf{r}^n}^T \\ \frac{\partial L}{\partial \mathbf{U}^n}^T \\ \frac{\partial L}{\partial \mathbf{x}^n}^T \end{Bmatrix} \tag{44}$$

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

$$\left[\frac{\partial \mathbf{J}^n}{\partial \mathbf{r}^n}\right]^T \Lambda^n_{\mathbf{r}} = \frac{\partial L}{\partial \mathbf{r}^n}^T + \frac{\partial \mathbf{x}^n_{\text{surf}}}{\partial \mathbf{r}^n}^T \Lambda^n_{\mathbf{x}} \tag{45}$$

$$\left[\frac{\partial \mathbf{R}^n}{\partial \mathbf{U}^n}\right]^T \Lambda_{\mathbf{U}}^n = \frac{\partial L}{\partial \mathbf{U}^n}^T - \frac{\partial \mathbf{J}^n}{\partial \mathbf{U}^n}^T \Lambda_{\mathbf{r}}^n \tag{46}$$

$$[\mathbf{K}]^T \Lambda_{\mathbf{x}}^n = \frac{\partial L}{\partial \mathbf{x}^n}^T - \frac{\partial \mathbf{J}^n}{\partial \mathbf{x}^n}^T \Lambda_{\mathbf{r}}^n - \frac{\partial \mathbf{R}^n}{\partial \mathbf{x}^n}^T \Lambda_{\mathbf{U}}^n \tag{47}$$

for which the solution process can be broken down as follows:

1) Construct right-hand sides of Eqs. (45–47) 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 a coupled adjoint system converges.

Assuming that the coefficient matrices are not ill-conditioned, the convergence of the adjoint equations is similar to the convergence of the linear systems in the solution process of the nonlinear analysis problem because the coefficient matrices are identical with the exception of the transpose. The typical convergence of such a coupled linear adjoint system is shown in Fig. 3. 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

$$\frac{\partial L^{n^T}}{\partial \mathbf{D}} = \left\{ \begin{matrix} -\frac{\partial \mathbf{J}^n}{\partial \mathbf{r}^{n-1}}\frac{\partial \mathbf{r}^{n-1}}{\partial \mathbf{D}} - \frac{\partial \mathbf{J}^n}{\partial \mathbf{r}^{n-2}}\frac{\partial \mathbf{r}^{n-2}}{\partial \mathbf{D}} \\ -\frac{\partial \mathbf{R}^n}{\partial \mathbf{U}^{n-1}}\frac{\partial \mathbf{U}^{n-1}}{\partial \mathbf{D}} - \frac{\partial \mathbf{R}^n}{\partial \mathbf{U}^{n-2}}\frac{\partial \mathbf{U}^{n-2}}{\partial \mathbf{D}} - \frac{\partial \mathbf{R}^n}{\partial \mathbf{x}^{n-1}}\frac{\partial \mathbf{x}^{n-1}}{\partial \mathbf{D}} - \frac{\partial \mathbf{R}^n}{\partial \mathbf{x}^{n-2}}\frac{\partial \mathbf{x}^{n-2}}{\partial \mathbf{D}} \\ \frac{\partial \mathbf{x}_{\text{surf}}^n}{\partial \mathbf{x}_{\text{surf}}^0}\frac{\partial \mathbf{x}_{\text{surf}}^0}{\partial \mathbf{D}} \end{matrix} \right\}^T \left\{ \begin{matrix} \Lambda_{\mathbf{r}}^n \\ \Lambda_{\mathbf{U}}^n \\ \Lambda_{\mathbf{x}}^n \end{matrix} \right\} \tag{48}$$

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

$$\frac{\partial L^{n^T}}{\partial \mathbf{D}} = \underbrace{\frac{\partial \mathbf{x}_{\text{surf}}^0}{\partial \mathbf{D}}^T \frac{\partial \mathbf{x}_{\text{surf}}^n}{\partial \mathbf{x}_{\text{surf}}^0}^T \Lambda_{\mathbf{x}}^n}_{(A)} + \underbrace{\left\{ \frac{\partial \mathbf{r}^{n-1}}{\partial \mathbf{D}} \quad \frac{\partial \mathbf{U}^{n-1}}{\partial \mathbf{D}} \quad \frac{\partial \mathbf{x}^{n-1}}{\partial \mathbf{D}} \right\} \left\{ \begin{matrix} -\frac{\partial \mathbf{J}^n}{\partial \mathbf{r}^{n-1}}^T \Lambda_{\mathbf{r}}^n \\ -\frac{\partial \mathbf{R}^n}{\partial \mathbf{U}^{n-1}}^T \Lambda_{\mathbf{U}}^n \\ -\frac{\partial \mathbf{R}^n}{\partial \mathbf{x}^{n-1}}^T \Lambda_{\mathbf{U}}^n \end{matrix} \right\}}_{(B)}$$

$$+ \underbrace{\left\{ \frac{\partial \mathbf{r}^{n-2}}{\partial \mathbf{D}} \quad \frac{\partial \mathbf{U}^{n-2}}{\partial \mathbf{D}} \quad \frac{\partial \mathbf{x}^{n-2}}{\partial \mathbf{D}} \right\} \left\{ \begin{matrix} -\frac{\partial \mathbf{J}^n}{\partial \mathbf{r}^{n-2}}^T \Lambda_{\mathbf{r}}^n \\ -\frac{\partial \mathbf{R}^n}{\partial \mathbf{U}^{n-2}}^T \Lambda_{\mathbf{U}}^n \\ -\frac{\partial \mathbf{R}^n}{\partial \mathbf{x}^{n-2}}^T \Lambda_{\mathbf{U}}^n \end{matrix} \right\}}_{(C)} \tag{49}$$

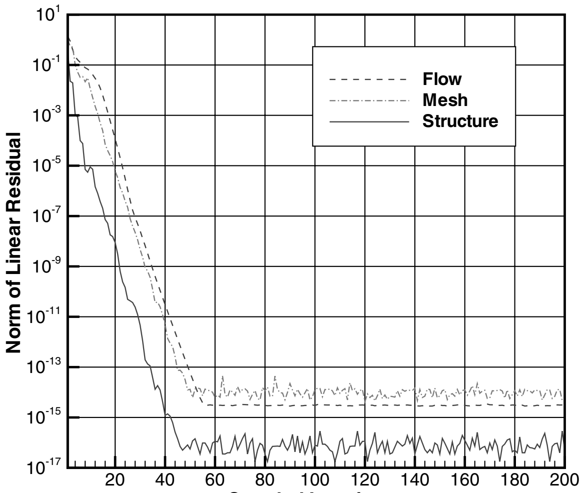


Fig. 3 Convergence of coupled adjoint problem at a single time level.

Term A in the preceding expression forms the contribution from time level n to the total sensitivity vector $[dL/dD]^T$. 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 (41) 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 $[dL/dD]^T$ 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

$$\left[\frac{\partial \mathbf{J}^n}{\partial \mathbf{r}^n}\right]^T \Lambda_{\mathbf{r}}^n = \frac{\partial L}{\partial \mathbf{r}^n}^T + \frac{\partial \mathbf{x}_{\text{surf}}^n}{\partial \mathbf{r}^n}^T \Lambda_{\mathbf{x}}^n - \frac{\partial \mathbf{J}^{n+1}}{\partial \mathbf{r}^n}^T \Lambda_{\mathbf{r}}^{n+1} - \frac{\partial \mathbf{J}^{n+2}}{\partial \mathbf{r}^n}^T \Lambda_{\mathbf{r}}^{n+2} \tag{50}$$

$$\left[\frac{\partial \mathbf{R}^n}{\partial \mathbf{U}^n}\right]^T \Lambda_{\mathbf{U}}^n = \frac{\partial L}{\partial \mathbf{U}^n}^T - \frac{\partial \mathbf{J}^n}{\partial \mathbf{U}^n}^T \Lambda_{\mathbf{r}}^n - \frac{\partial \mathbf{R}^{n+1}}{\partial \mathbf{U}^n}^T \Lambda_{\mathbf{U}}^{n+1} - \frac{\partial \mathbf{R}^{n+2}}{\partial \mathbf{U}^n}^T \Lambda_{\mathbf{U}}^{n+2} \tag{51}$$

$$[\mathbf{K}]^T \Lambda_{\mathbf{x}}^n = \frac{\partial L}{\partial \mathbf{x}^n}^T - \frac{\partial \mathbf{J}^n}{\partial \mathbf{x}^n}^T \Lambda_{\mathbf{r}}^n - \frac{\partial \mathbf{R}^n}{\partial \mathbf{x}^n}^T \Lambda_{\mathbf{U}}^n - \frac{\partial \mathbf{R}^{n+1}}{\partial \mathbf{x}^n}^T \Lambda_{\mathbf{U}}^{n+1} - \frac{\partial \mathbf{R}^{n+2}}{\partial \mathbf{x}^n}^T \Lambda_{\mathbf{U}}^{n+2} \tag{52}$$

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,

$$\frac{\partial \mathbf{x}_{\text{surf}}^0}{\partial \mathbf{D}}^T$$

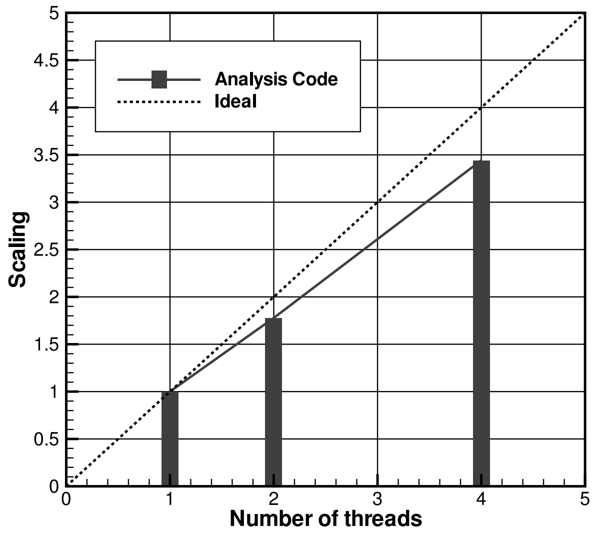
appearing at each time level in the form of term A in Eq. (49). This particular matrix is essentially the linearization of the shape functions that modify the airfoil geometry with respect to the design inputs and is easily computable.

An important observation at this point is that the method described previously 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.

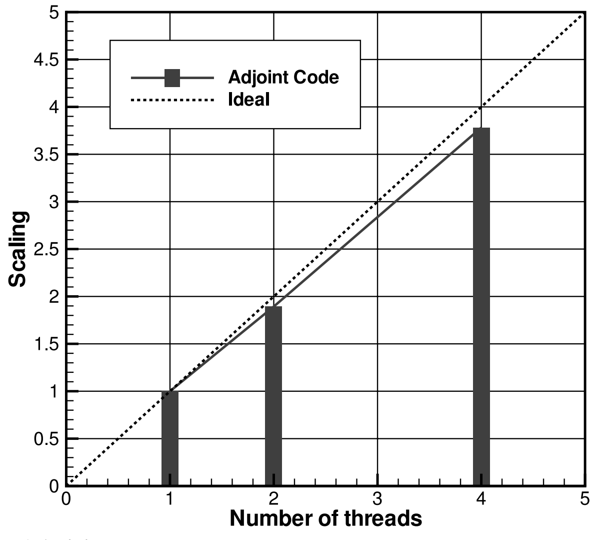
V. Implementation, Validation, and Results

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 Sec. II, the analysis code is second-order-accurate both spatially and temporally, and the adjoint code is a discretely exact linearization of all aspects of the analysis code. Both codes run in parallel on multicore 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 GB 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, in which, once the coefficient matrices have been set up, 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 4a and 4b 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, whereas 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, because 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.



a) Analysis code



b) Adjoint code

Fig. 4 Performance scaling of analysis and adjoint codes using OpenMP parallelization.

B. Linearization for Second-Order Spatial Accuracy

Computation of the flow adjoint variable requires the linearization of the flow residual equation with respect to the flow state variables. Such a linearization results in the flow Jacobian matrix $\partial\mathbf{R}/\partial\mathbf{U}$ and is also used in the Newton solver employed for the flow solution. In the case of a spatially-first-order-accurate discretization, the flow Jacobian matrix is constructed using a nearest-neighbor stencil, in which, for any element in the mesh, the flow residual $\mathbf{R}$ is a function of its own state variables and the state variables of its immediate neighbors. On triangular unstructured meshes, this translates into the flow Jacobian matrix being sparse, with each row consisting of a dominant diagonal block element and three offdiagonal block elements. A simple edge-based data structure is sufficient for the purpose of constructing and storing the elements of the Jacobian matrix. In the case of second-order spatial accuracy, the flow residual of each element is no longer restricted to a nearest-neighbor stencil, because the residual $\mathbf{R}$ depends not only on the state variables, but also their undivided Laplacian. This is can be seen in Eq. (10), describing the matrix dissipation scheme used to construct the fluxes with second-order accuracy, where $(\nabla^2\mathbf{U})$ is the undivided Laplacian of the state vector $\mathbf{U}$. Because the residual is a now a function of element states extending beyond the nearest-neighbor stencil, constructing and storing an exact flow Jacobian matrix quickly becomes impractical, due to large memory requirements and the lack of a simple data structure. For the flow solver, it is not required that an exact linearization of the flow residual $\mathbf{R}$ be available because only the solution of $\mathbf{R}(\mathbf{U}) = 0$ is necessary. To achieve second-order accuracy, it is only required that the flow residual itself be constructed for second-order accuracy of $\mathbf{U}$. An inexact Newton's method is employed in which the first-order-accurate flow Jacobian matrix is used in solving second-order-accurate residual equations $\mathbf{R}_2(\mathbf{U})$. The method being inexact no longer exhibits the quadratic rate of convergence typical of a Newton solver, but greatly simplifies the solution procedure.

In the case of the adjoint solver, it has been shown that approximating a second-order-accurate flow Jacobian matrix with a first-order-accurate matrix leads to significant errors in the computed sensitivities [24]. The linear systems that involve the flow Jacobian matrix are Eqs. (46) and (51). Although computing and storing the exact second-order Jacobian is impractical, it is quite straightforward to construct the product of the second-order Jacobian and a vector using a two-pass approach [4,5]. This property is taken advantage of and a defect correction method is used to solve the linear system involving the second-order flow Jacobian matrix. For example, consider the left-hand side of the linear system shown in Eq. (46). In the case in which the flow Jacobian matrix is second-order-accurate, this can be split into the sum of two matrix-vector products as follows:

$$\left[\frac{\partial\mathbf{R}}{\partial\mathbf{U}}\right]_2^T\Lambda_\mathbf{U} = \left\{\left[\frac{\partial\mathbf{R}}{\partial\mathbf{U}}\right]_1 + \left[\frac{\partial\mathbf{R}}{\partial(\nabla^2\mathbf{U})}\right]\left[\frac{\partial(\nabla^2\mathbf{U})}{\partial\mathbf{U}}\right]\right\}^T\Lambda_\mathbf{U} = \underbrace{\left[\frac{\partial\mathbf{R}}{\partial\mathbf{U}}\right]_1^T\Lambda_\mathbf{U}}_{(1)} + \underbrace{\left[\frac{\partial(\nabla^2\mathbf{U})}{\partial\mathbf{U}}\right]^T\left[\frac{\partial\mathbf{R}}{\partial(\nabla^2\mathbf{U})}\right]^T\Lambda_\mathbf{U}}_{(2)} \tag{53}$$

where $[\partial\mathbf{R}/\partial\mathbf{U}]_1$ is the first-order Jacobian. Term 1 can be easily computed because the first-order Jacobian is already stored for the solution of the analysis problem. Term 2 can be computed via a series of matrix-vector products on-the-fly if $\Lambda_\mathbf{U}$ is available. The defect correction method is a pseudononlinear solution method, and for Eq. (46), it can be expressed as

$$[\mathbf{P}]\delta\Lambda_\mathbf{U}^k = -\left\{\left[\frac{\partial\mathbf{R}^n}{\partial\mathbf{U}^n}\right]_2^T\Lambda_\mathbf{U}^k + \frac{\partial L}{\partial\mathbf{U}^n}^T - \frac{\partial\mathbf{J}^n}{\partial\mathbf{U}^n}^T\Lambda_\mathbf{r}^n\right\}$$

$$\Lambda_\mathbf{U}^{k+1} = \Lambda_\mathbf{U}^k + \delta\Lambda_\mathbf{U}^k, \qquad \delta\Lambda_\mathbf{U}^k \to 0, \qquad \Lambda_\mathbf{U}^k = \Lambda_\mathbf{U}^n \tag{54}$$

Here, $[\mathbf{P}]$ refers to the preconditioner and $\delta\Lambda_\mathbf{U}$ is the correction for $\Lambda_\mathbf{U}$. When the correct value for $\Lambda_\mathbf{U}$ has been achieved, the right-hand side of Eq. (54) goes to zero. For the system to converge, the preconditioner [**P**] must be closely related to the second-order flow Jacobian matrix. Typically, the preconditioner is taken to be the transpose of the first-order flow Jacobian matrix [5,25]. The right-hand side of Eq. (54) includes the effect of the second-order flow Jacobian matrix and is evaluated as described by Eq. (53).

C. Damping for Linear Adjoint Systems

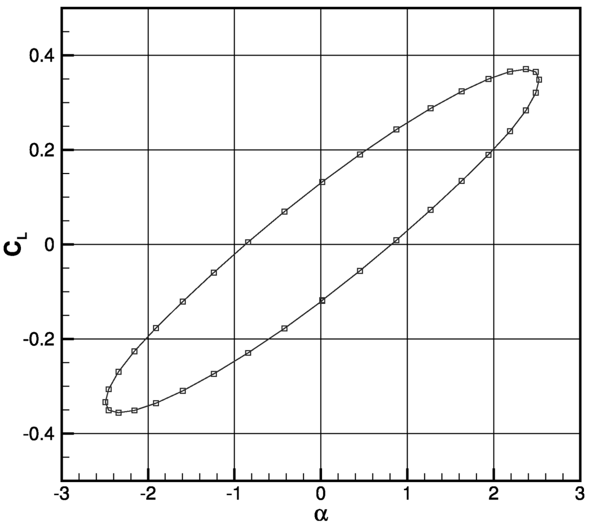
The flow adjoint equation may contain linearly unstable modes and be ill-conditioned, resulting in difficulties during the solution process, particularly at steady state. This is not a major concern for unsteady flow adjoint equations, however, especially when the time-step size is small. In either case, the defect correction approach offers improved robustness during the adjoint solution process without sacrificing discretely exact linearizations of the governing equations. Should an adjoint equation, steady or unsteady, pose problems due to ill-conditioning of the coefficient matrix, the defect correction approach can be invoked to solve the system. The preconditioner [**P**] is then taken as the damped version of the coefficient matrix by artificially augmenting the diagonal terms with some predetermined damping factor.

D. Solver Validation

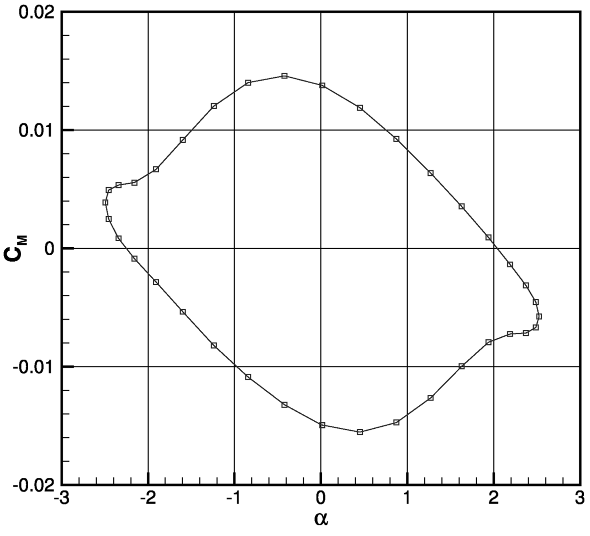
The spatial and temporal discretizations of the analysis solver were validated by performing mesh refinement studies and shown to be able to achieve near-design accuracy. Details of the study can be found in [26]. The next step in the validation of the analysis code is to verify that computed time-dependent 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 [27]. 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 nondimensional time t is defined as

$$\alpha = \alpha_0 + \alpha_{\max} \sin(2k_c t) \tag{55}$$

For AGARD test case no. 5, the airfoil pitches around its quarter-chord with a mean angle of attack α_0 of 0.016 deg and an amplitude of pitch $\alpha_{\max}$ of 2.51 deg at a reduced frequency of $k_c = 0.0814$. We use a computational mesh consisting of approximately 10,000 elements, shown in Fig. 5, for the unsteady validation. Figures 6a and 6b show the time-varying lift and moment coefficients computed by the solver, which match well with results from [21,23,27,28], thus establishing confidence in the nonaeroelastic unsteady aspect of the solver.



a) Time variation of lift coefficient for unsteady solver validation



b) Time variation of moment coefficient for unsteady solver validation

Fig. 6 Unsteady load coefficients for transonic pitching NACA0012 airfoil.

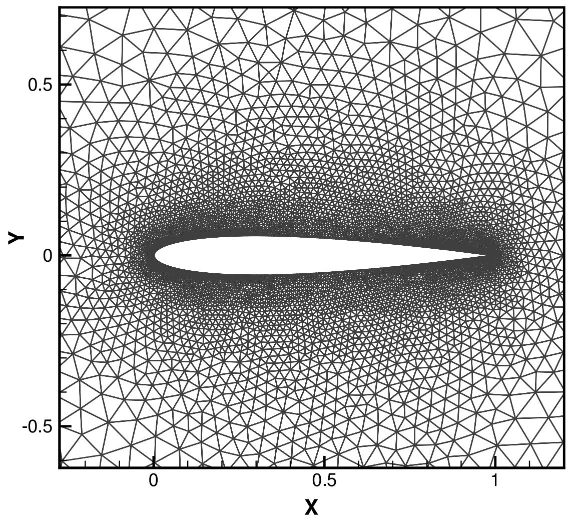


Fig. 5 NACA0012 mesh consisting of approximately 10,000 elements used for unsteady solver validation.

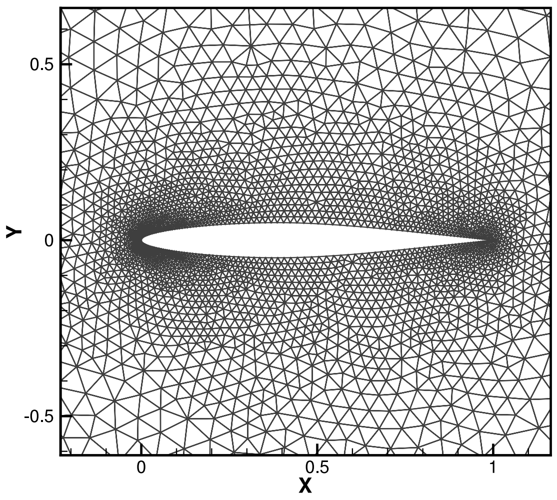


Fig. 7 NACA64A010 mesh consisting of approximately 6600 elements used for aeroelastic solver validation and in optimization example A.

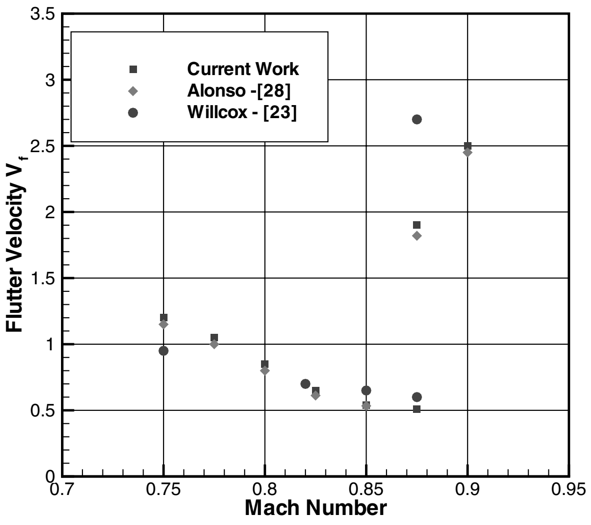


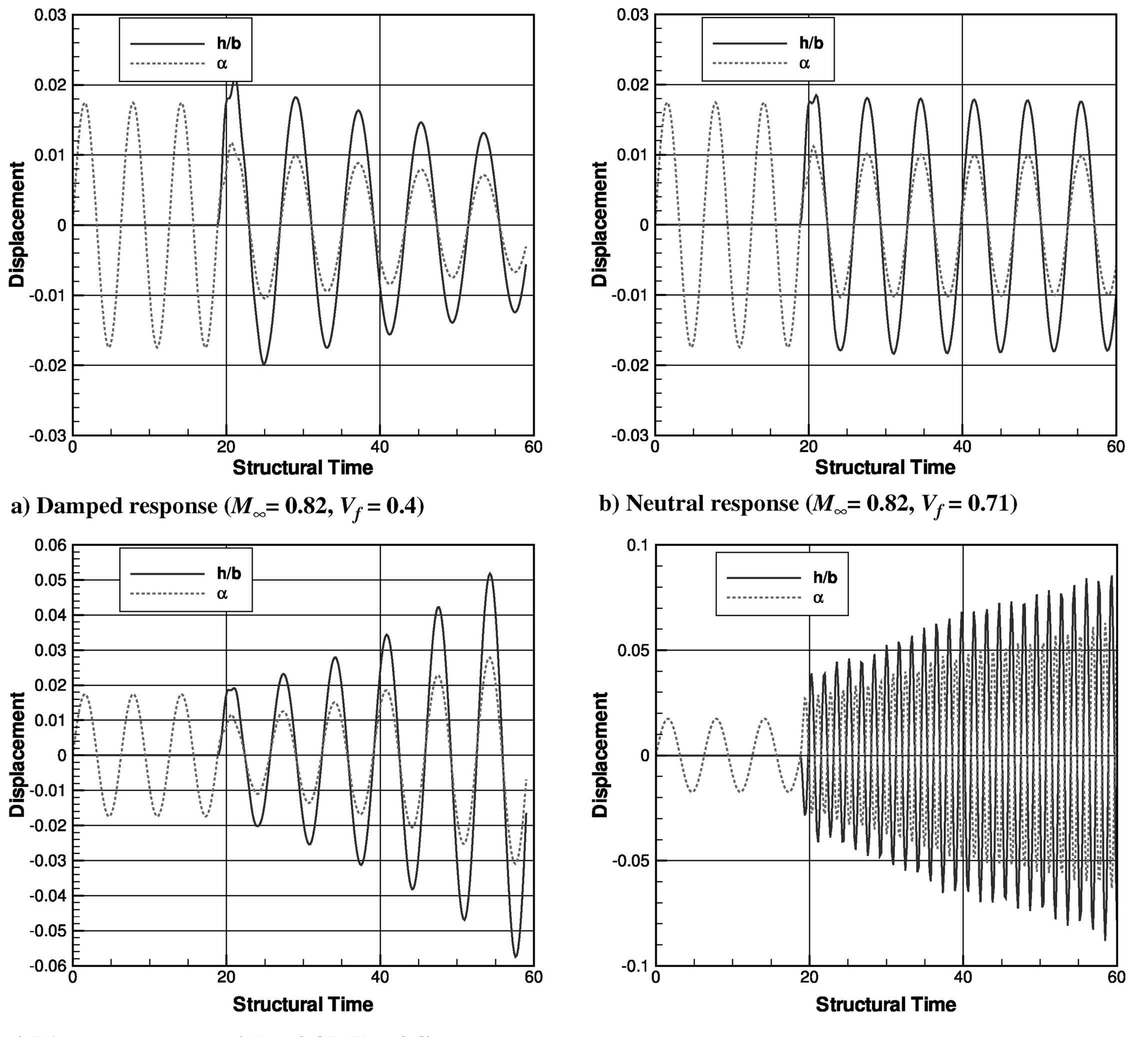
Fig. 8 Comparison of predicted flutter boundary against other references.

The final 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 [29] is chosen as the test case. The structural parameters for this case are

$$x_\alpha = 1.8, \qquad r_\alpha^2 = 3.48, \qquad \omega_h, \omega_\alpha = 100 \text{ rad/s}$$

$$\mu = 60, \qquad a = -2.0$$

where the quantity a is the nondimensional elastic axis location along the chord of the airfoil measured from the midchord of the airfoil when it is in the neutral position. Because it is nondimensionalized by the semichord 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 (NASA Ames Research Center) airfoil operating with a mean angle of attack α_0 of zero and an amplitude of forced pitching $\alpha_{\max}$ of 1 deg. The computational mesh used for this case consists of approximately 6600 elements and is shown in Fig. 7. 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 [23,28], as indicated by the flutter diagram in Fig. 8. The aeroelastic responses of the airfoil in both pitch and plunge degrees of freedom at various locations (stable and unstable) in the flutter diagram are shown in Fig. 9 and follow expected trends.



a) Damped response (M_∞= 0.82, V_f = 0.4)

b) Neutral response (M_∞= 0.82, V_f = 0.71)

c) Divergent response (M_∞= 0.85, V_f = 0.8)

d) Second mode response (M_∞= 0.875, V_f = 2.6)

Fig. 9 Aeroelastic response of the NACA64A010 airfoil at various points in the flutter diagram.

E. Sensitivity Validation

The procedure to validate the adjoint-based sensitivity is twofold. The forward linearization is typically more intuitive, because 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 [30] 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, and Fig. 10 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 Fig. 10.

F. 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 [31]. The algorithm not only uses gradient information, but also builds an approximate Hessian matrix (second derivative of the functional with respect to design parameters) on-the-fly using the optimization history to speed up the overall convergence to minimum. The bounded, rather than the unbounded, LBFGS algorithm was chosen 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 that do not collapse the geometry.

G. Optimization Example A

The goal of this optimization example is to consider a point in the flutter diagram of the described aeroelastic test case at which the airfoil exhibits divergent behavior and to change the shape of the airfoil such

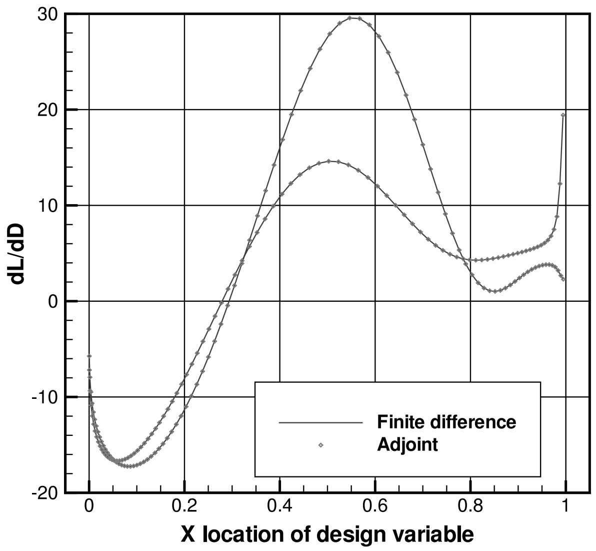


Fig. 10 Comparison of the gradients computed by linearization and finite differencing.

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

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

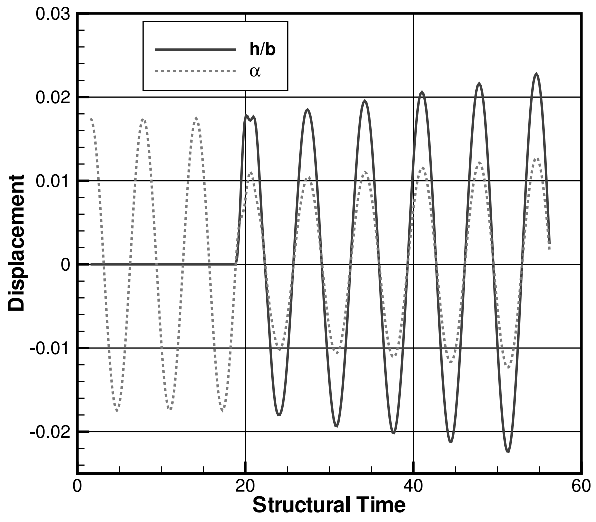
that it produces a damped aeroelastic response while satisfying a constraint on the steady-state lift of the airfoil. The time-integration process consisted of 36 uniform time steps per forced pitch cycle combined with 3 pitch cycles and 214 time steps of the same size for the aeroelastic response to 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 6600-element mesh described earlier, a single design cycle consisting of a forward analysis solution and a backward adjoint solution roughly cost 30 min of wall time. The objective function for the optimization was chosen to be

$$L = \mathbf{r}^{f^T}[W_1]\mathbf{r}^f + W_2(C_{Ls} - C_{Ls_{\text{target}}}) \tag{56}$$

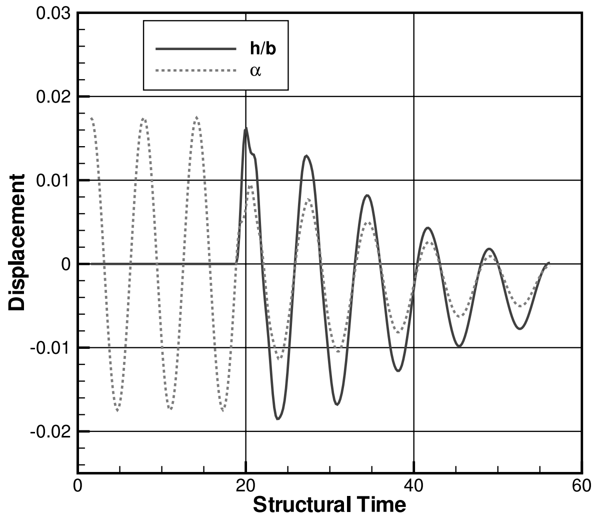
where $\mathbf{r}^f$ refers to the structural state vector at the end of the time-integration process; $[W_1]$ is a diagonal weighting matrix with equal weighting parameters of 100, W_2 is a weight on the constraint, chosen to be unity; and C_{Ls} is the steady-state lift coefficient of the airfoil. Because 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 deg. The objective function formulation is such that a vanishing objective function results in zero displacements and zero velocities in both degrees of freedom of pitch and plunge. Because the aeroelastic time-integration ends fairly quickly ($5\frac{1}{2}$ periods), this in fact 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 11a shows the aeroelastic response of the airfoil at the starting design point. The divergent behavior is clearly visible in both pitch and plunge axes. Before 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) 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 Fig. 11b shows the aeroelastic response of the optimized airfoil. In terms of wall time, the optimization cost approximately 21 h. The entire process required about 200 MB of disk space at each design iteration to hold the unsteady solution set for use by the adjoint



a) Diverging aeroelastic response of baseline airfoil



b) Decaying aeroelastic response of optimized airfoil

Fig. 11 Aeroelastic response of baseline and optimized airfoils in optimization example A.

solver, which s then overwritten by the following design iteration. Although the displacements and the velocities have not been driven to zero, the plot clearly indicates a rapidly decaying response. Figure 12 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 because 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 13 indicates that the constraint satisfies the set tolerance only at 41 design iterations, at which point the optimization was terminated. Figure 14 compares the optimized and the baseline airfoil geometries using 1:1 and exaggerated scales.

H. Optimization Example B

The purpose of the second optimization example is to demonstrate the advantage of using an unstructured mesh framework. The problem remains identical to the first optimization example in that the goal is to suppress divergent flutter behavior of an airfoil using gradient-based optimization. However, the airfoil under consideration contains two elements separated by a slot and is thus easier to treat using unstructured methods. Figure 15 shows the unstructured computational mesh consisting of approximately 7800 elements

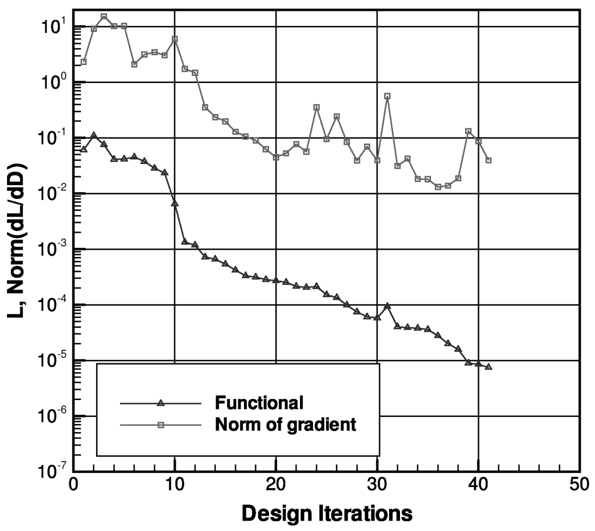


Fig. 12 Convergence of objective function L and L_2 norm of the gradient vector.

around the slotted airfoil for this problem. Because the primary focus of this optimization example is the demonstration of the advantage of using unstructured meshes for complex geometries in conjunction with the derived linearization, the physical aspects of the aeroelastic problem itself are not very important. With this in mind, the structural parameters from the Isogai [29] test case used in the previous optimization example are carried over to this example. A suitable starting point at which the airfoil exhibits a divergent aeroelastic response was chosen by trial and error to be a flutter velocity of 1.85 at a Mach number of 0.6. The response of the slotted airfoil at this point in the flutter diagram is shown in Fig. 16a. The offsets in the mean of the oscillations of both the pitch and plunge axes from zero indicate that for the given structural and aerodynamic parameters there exists a nontrivial steady-state aeroelastic solution. The objective function for this example was chosen to be

$$L = \sum_{n=n_f-10}^{n_f} \mathbf{r}^{nT}[W_1]\mathbf{r}^n + W_2(C_{Ls} - C_{Ls_{\text{target}}}) \tag{57}$$

In contrast to the first optimization example, the aeroelastic term in the objective function has been converted to a summation over the last ten time steps. The reasoning being that in addition to the case of flutter-free behavior, zero velocities and zero displacements occur at the peaks of the oscillations. Considering the offset in the oscillatory

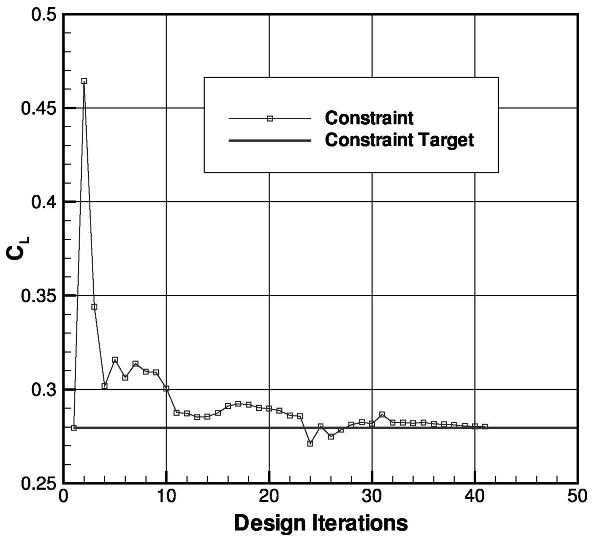
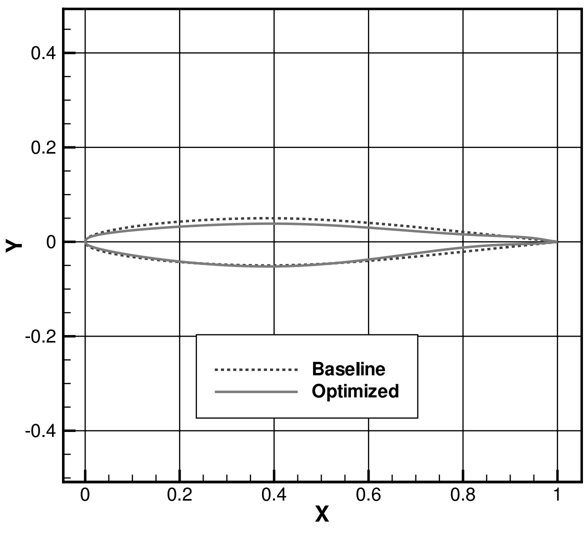
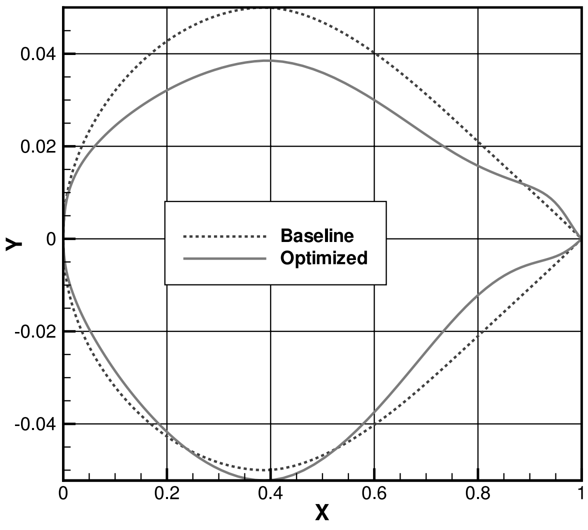


Fig. 13 Convergence of constraint on objective function.

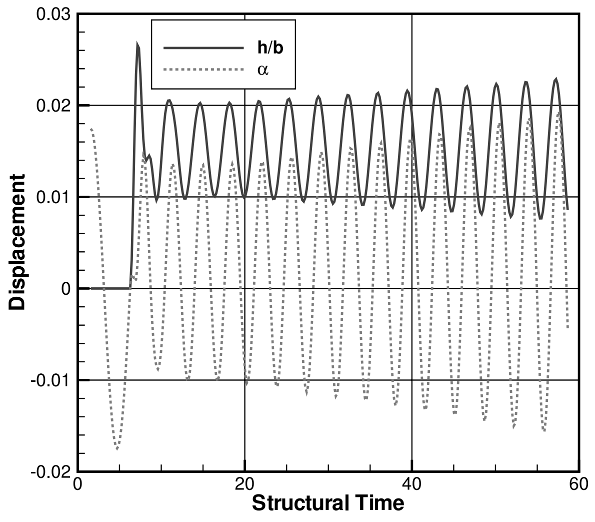


a) 1:1 scale

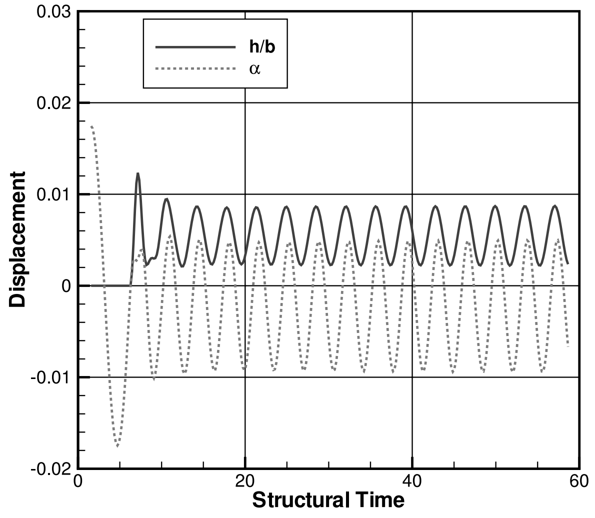


b) Exaggerated scale

Fig. 14 Comparison of baseline and optimized airfoils.



a) Diverging aeroelastic response of baseline slotted airfoil



b) Neutral aeroelastic response of optimized slotted airfoil

Fig. 16 Aeroelastic response of baseline and optimized response in optimization example B.

mean values from zero, this change in the objective formulation effectively constrains the optimization from moving to a shape that produces an oscillatory peak at zero once the time-integration process terminates. The weights in the objective remain identical to the first example, and the target steady-state lift coefficient is again that of the baseline slotted airfoil at the maximum pitch angle of the forced pitching cycles. The forced pitch parameters are also identical to the first example, with the exception of the number of forced pitch

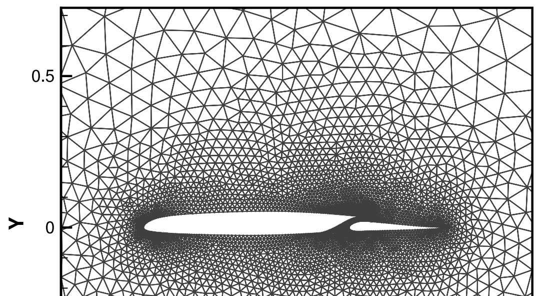


Fig. 15 Slotted airfoil mesh consisting of approximately 7800 elements used for aeroelastic solver validation and in optimization example B.

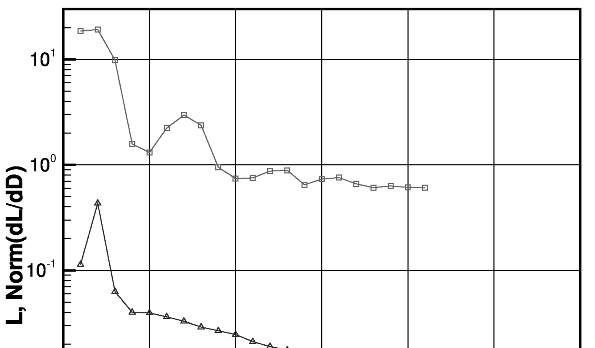


Fig. 17 Convergence of objective function L and L_2 norm of the gradient vector.

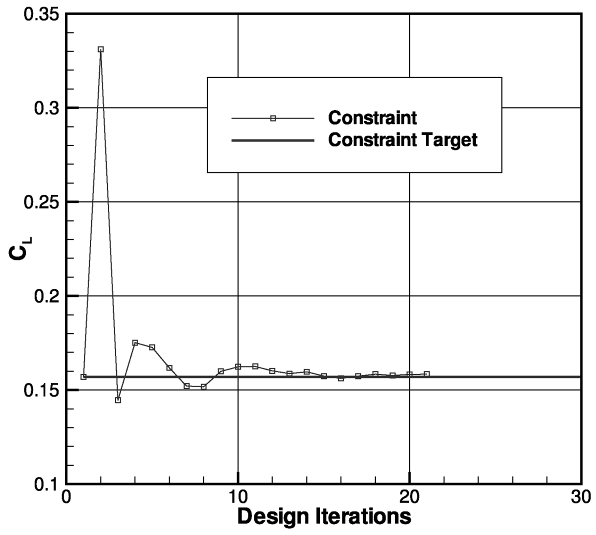


Fig. 18 Convergence of constraint on objective function.

cycles. For this example, the forced pitching occurs only over three-quarters of a period before the aeroelastic response is allowed to evolve. Because the overall size of the time domain remains identical to the previous example, this modification to the forced pitching cycles allows more time for the aeroelastic response evolution. Four sets of eight shape functions corresponding to the upper and lower surfaces of both the main airfoil element and the flap result in a total of 32 design variables that control the overall shape of the airfoil. Figure 16b shows the aeroelastic response of the optimized airfoil and clearly indicates neutral behavior. Figures 17 and 18 show the convergence of the objective function, the norm of the gradient and the constraint. The optimization was terminated after 21 design iterations and cost approximately 25 h of wall time. Figure 19 compares the baseline and optimized airfoil shapes.

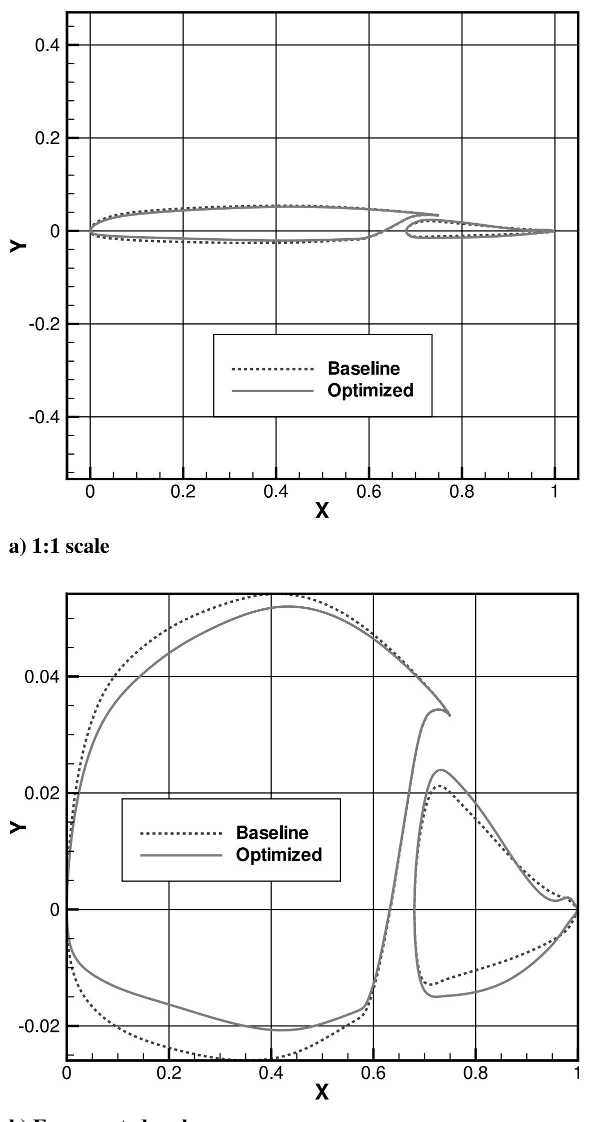


Fig. 19 Comparison of baseline and optimized airfoils.

VI. Conclusions

An adjoint-based approach has been developed and employed for 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 costs of the two-dimensional problems suggest that full three-dimensional unsteady aeroelastic optimization problems, although relatively expensive, should be tractable on current-day large parallel machines in terms of computational time, memory, and disk space for storing the solution time history. For the two-dimensional inviscid examples presented in the paper, the solution files that were written to disk at each time step were approximately 350 kB in size. Estimates based on simplistic calculations indicate that realistic three-dimensional problems running in parallel with approximately 200,000 elements per processor would result in file sizes anywhere between 10 and 15 MB at each time step. The adjoint solver while sweeping backward in time only has to read two or three such files at each time step. Although the cost of file I/O operations can depend greatly on machine and file system specifications, our work indicates that constructing and solving the adjoint equations is significantly more expensive than reading the required files at each time step. The main concern is the storage of the entire solution history for the three-dimensional problem described here, which results in approximately 75 GB of files per processor when using about 5000 time steps. 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.

A major concern when applying the developed method to realistic three-dimensional problems is the stiffness of the optimization problem itself. Objective functionals in real-world aeroelasticity problems typically involve large numbers of constraints arising due to complicated structures. This can lead to extremely noisy or even discontinuous design spaces, which may result in optimization difficulties. Although the two-dimensional examples presented in the paper are a proof of concept of the method, extension to three dimensions will require addressing of such issues. The main thrust of the paper has been on deriving a generalized unsteady adjoint method for coupled systems, which can be extended to three dimensions as is. Finally, the current approach should be easily extendable to computing discretely exact sensitivity derivatives for any number of coupled disciplines.

References

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

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

[3] Jameson, A., Alonso, J., Reuther, J., Martinelli, L., and Vassberg, J., "Aerodynamic Shape Optimization Techniques Based on Control

Theory," AIAA Paper 98–2538, 1998.

[4] Mavriplis, D. J., "Multigrid Solution of the Discrete Adjoint for Optimization Problems on Unstructured Meshes," *AIAA Journal*, Vol. 44, No. 1, Jan. 2006, pp. 42–50. doi:10.2514/1.15696

[5] Mavriplis, D. J., "Discrete Adjoint-Based Approach for Optimization Problems on Three-Dimensional Unstructured Meshes," *AIAA Journal*, Vol. 45, No. 4, Apr. 2007, pp. 740–750. doi:10.2514/1.22743

[6] Nielsen, E., and Anderson, W., "Recent Improvements in Aerodynamic Design Optimization on Unstructured Meshes," *AIAA Journal*, Vol. 40, No. 6, June 2002, pp. 1155–1163. doi:10.2514/2.1765

[7] Elliott, J., and Peraire, J., "Practical Three-Dimensional Aerodynamic Design and Optimization Using Unstructured Meshes," *AIAA Journal*, Vol. 35, No. 9, 1997, pp. 1479–1485. doi:10.2514/2.271

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

[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, No. 11, 2001, pp. 2051–2061. doi:10.2514/2.1227

[11] Joaquim, R. R. A., Alonso, J. J., and Reuther, J., "Aero-Structural Wing Design Optimization Using High-Fidelity Sensitivity Analysis," *Proceedings—CEAS Conference on Multidisciplinary Aircraft Design Optimization*, Council of European Aerospace Societies, June 2001, pp. 211–226.

[12] Mani, K., and Mavriplis, D. J., "Unsteady Discrete Adjoint Formulation for Two-Dimensional Flow Problems with Deforming Meshes," *AIAA Journal*, Vol. 46, No. 6, June 2008, pp. 1351–1364. doi:10.2514/1.29924

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

[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, AIAA Paper 2007–1128, Reno, NV, Jan. 2007.

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

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

[17] Mavriplis, D. J., "Unstructured-Mesh Discretizations and Solvers for Computational Aerodynamics," *AIAA Journal*, Vol. 46, No. 6, June 2008, pp. 1281–1298. doi:10.2514/1.34681

[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, No. 1, 2003, pp. 206–227. doi:10.1016/S0021-9991(03)00311-5

[19] Yang, Z., and Mavriplis, D. J., "Higher-Order Time Integration Schemes for Aeroelastic Applications on Unstructured Meshes," *AIAA Journal*, Vol. 45, No. 1, Jan. 2007, pp. 138–150. doi:10.2514/1.22847

[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, No. 2, 2006, pp. 557–573. doi:10.1016/j.jcp.2005.08.018

[21] Batina, J. T., "Unsteady Euler Airfoil Solutions Using Unstructured Dynamic Meshes," *AIAA Journal*, Vol. 28, No. 8, Aug. 1990, pp. 1381–1388. doi:10.2514/3.25229

[22] Hicks, R., and Henne, P., "Wing Design by Numerical Optimization," *Journal of Aircraft*, Vol. 15, No. 7, 1978, pp. 407–412. doi:10.2514/3.58379

[23] Willcox, K., and Peraire, J., "Aeroelastic Computations in the Time Domain Using Unstructured Meshes," *International Journal for Numerical Methods in Engineering*, Vol. 40, No. 13, 1997, pp. 2413–2431. doi:10.1002/(SICI)1097-0207(19970715)40:13<2413::AID-NME170>3.0.CO;2-E

[24] Dwight, R., and Brezillon, J., "Effect of Various Approximations of the Discrete Adjoint on Gradient-Based Optimization," 44th Aerospace Sciences Meeting and Exhibit, AIAA Paper 2006–0690, Reno, NV, 2006.

[25] Nielsen, E., Lu, J., Park, M., and Darmofal, D., "An Exact Dual Adjoint Solution Method for Turbulent Flows on Unstructured Grids," AIAA Paper 2003–0272, 2003.

[26] Mani, K., "Application of the Discrete Adjoint Method to Coupled Multidisciplinary Unsteady Flow Problems for Error Estimation and Optimization," Ph.D. Thesis, Univ. of Wyoming, Laramie, WY, 2009.

[27] *Compendium of Unsteady Aerodynamic Measurements*, AGARD Rept. 702, Neuilly sur Seine, France, 1982.

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

[29] Isogai, K., "Transonic Dip Mechanism of Flutter of a Sweptback Wing: Part 2," *AIAA Journal*, Vol. 19, No. 9, 1981, pp. 1240–1242. doi:10.2514/3.7853

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

[31] 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, No. 5, 1995, pp. 1190–1208. doi:10.1137/0916069

A. Messac
Associate Editor