Computation of Aircraft Stability Derivatives Using an Automatic Differentiation Adjoint Approach

Charles A. Mader* University of Toronto, Toronto, Ontario M3H 5T6, Canada and Joaquim R. R. A. Martins† University of Michigan, Ann Arbor, Michigan 48109

DOI: 10.2514/1.J051147

This paper presents a method for the computation of the static and dynamic stability derivatives of arbitrary aircraft configurations. Three-dimensional computational fluid dynamics are used in this method to simulate the flow characteristics around the configuration, and a moving-grid formulation is included in the flow solver to handle the rotational physics necessary for the computation of the dynamic derivatives. To obtain the stability derivatives, the computational fluid dynamics code is differentiated using the automatic differentiation adjoint (ADjoint) approach. This approach enables the efficient and accurate computation of derivatives for a wide variety of variables, including the dynamic model states that are typical of the stability derivatives. To demonstrate the effectiveness of this approach, stability derivatives are computed for a NACA 0012 airfoil and an ONERA M6 wing.

Nomenclature

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>b</td>
<td>span</td>
</tr>
<tr>
<td>CD</td>
<td>aircraft drag coefficient</td>
</tr>
<tr>
<td>CL</td>
<td>aircraft lift coefficient</td>
</tr>
<tr>
<td>Cm</td>
<td>airfoil lift momentum coefficient (Sec. IV)</td>
</tr>
<tr>
<td>Cn</td>
<td>aircraft yaw moment coefficient</td>
</tr>
<tr>
<td>cp</td>
<td>coefficient of pressure</td>
</tr>
<tr>
<td>cy</td>
<td>aircraft side force coefficient</td>
</tr>
<tr>
<td>c</td>
<td>chord</td>
</tr>
<tr>
<td>e</td>
<td>total energy</td>
</tr>
<tr>
<td>f</td>
<td>flux term</td>
</tr>
<tr>
<td>h</td>
<td>altitude</td>
</tr>
<tr>
<td>I</td>
<td>identity matrix (Sec. III-A), function of interest (Sec. III-C)</td>
</tr>
<tr>
<td>k</td>
<td>thermal conductivity of fluid</td>
</tr>
<tr>
<td>M</td>
<td>Mach number</td>
</tr>
<tr>
<td>N</td>
<td>number of surface cells (PMARC), number of volume cells (SUmb)</td>
</tr>
<tr>
<td>p</td>
<td>pressure (Sec. III-A), roll rate (Sec. III-C), order of convergence (Sec. IV-A)</td>
</tr>
<tr>
<td>phi</td>
<td>normalized roll rate (pb/2V)</td>
</tr>
<tr>
<td>q</td>
<td>pitch rate</td>
</tr>
<tr>
<td>phi</td>
<td>normalized pitch rate (qc/2V)</td>
</tr>
<tr>
<td>R</td>
<td>flow residuals</td>
</tr>
<tr>
<td>r</td>
<td>yaw rate</td>
</tr>
<tr>
<td>phi</td>
<td>normalized yaw rate (rb/2V)</td>
</tr>
<tr>
<td>s</td>
<td>source terms</td>
</tr>
<tr>
<td>T</td>
<td>temperature</td>
</tr>
<tr>
<td>t</td>
<td>time</td>
</tr>
<tr>
<td>u</td>
<td>flow velocity with respect to the fixed frame</td>
</tr>
<tr>
<td>v</td>
<td>flow velocity with respect to the moving grid</td>
</tr>
<tr>
<td>V</td>
<td>aircraft speed</td>
</tr>
<tr>
<td>w</td>
<td>velocity of the moving grid</td>
</tr>
<tr>
<td>x</td>
<td>grid coordinates (Sec. III-A), design variables (Sec. III-C)</td>
</tr>
<tr>
<td>xref</td>
<td>center of moment and rotation with respect to wing root/airfoil leading edge</td>
</tr>
<tr>
<td>a</td>
<td>angle of attack</td>
</tr>
<tr>
<td>b</td>
<td>sideslip angle</td>
</tr>
<tr>
<td>b_1</td>
<td>ith control surface deflection (Sec. III-C), elements of the identity matrix (Sec. III-A)</td>
</tr>
<tr>
<td>c</td>
<td>flow states</td>
</tr>
<tr>
<td>rho</td>
<td>density</td>
</tr>
<tr>
<td>s</td>
<td>viscous stress tensor</td>
</tr>
<tr>
<td>t</td>
<td>generalized time variable</td>
</tr>
<tr>
<td>psi</td>
<td>adjoint vector</td>
</tr>
<tr>
<td>Omega</td>
<td>rotational rate of the moving grid</td>
</tr>
</tbody>
</table>

I. Introduction

Flight dynamic characteristics are an essential factor in aircraft design. However, due to the unsteady nature of the flow around the aircraft during maneuvers, it is extremely challenging to determine the aerodynamic characteristics of aircraft for flight dynamics purposes. One common way to make this problem more tractable is to assume that the aerodynamic forces on the aircraft react in a linear fashion for small variations from a given steady-state flight condition. This assumption allows the forces to be characterized by a series of aerodynamic derivatives, typically called stability derivatives [1]. These derivatives can be calculated using empirical methods (e.g., DATCOM [2] and ESDU [3,4]), wind-tunnel testing, or computational fluid dynamics (CFD). Each of these approaches has its own advantages and disadvantages with respect to cost, range of applicability, and computational time. In this work, we seek to use adjoint methods and automatic differentiation to remove some of the barriers limiting the use of high-fidelity CFD for these computations.

II. Background

The task of computing aerodynamic information for stability and control purposes remains a difficult challenge for aircraft designers. This difficulty is especially acute in the early stages of the design process, when detailed information about the design is less certain. There are several well-documented cases, even for relatively modern aircraft such as the Boeing 777 and the Learjet 23, in which stability and control problems were not diagnosed until the flight-test...
stage of the design, resulting in costly late-stage modifications to the aircraft [5]. CFD has been identified as a tool that may be able to address these issues, since it can give a better understanding of the stability and control characteristics of a proposed configuration at various stages of the design. Some CFD-based methods have already made inroads into this area of analysis. For example, according to Johnson et al. [6], Boeing uses their A502 panel code to compute stability derivatives at the preliminary design stage. However, while the methods currently in use provide rapid turnaround times and provide useful results for a wide variety of cases, they still have some significant limitations, which have diminished their general acceptance for use in stability and control. As demonstrated by the COMSAC working group [5], in order for CFD to truly be accepted in stability and control prediction, it must be able to handle computations for the full flight envelope, including flight conditions exhibiting massive flow separation, which often occur at the edges of the flight envelope. In the working group’s opinion, to accurately predict the flow for these difficult flight conditions, higher-fidelity CFD methods are needed, such as Reynolds-averaged Navier–Stokes (RANS), large eddy simulation, or detached eddy simulation (DES).

Considering this outlook, we can divide the problem of computing stability and control information from CFD into two parts: the task of developing CFD methods to the point where they accurately and robustly model the flows necessary to examine the entire flight envelope, and the task of developing computational techniques to compute usable stability and control information from these solutions efficiently. In this paper, we focus on the second of these two tasks.

Toward this end, many scholars have used a variety of techniques to compute stability and control information using CFD. Charlton [7] conducted simple $\alpha$ and $\beta$ sweeps to obtain the force and moment information required for falling-leaf predictions for tailless aircraft. The study concluded that, in most instances, the required stability data could be computed accurately, but also that highly nonlinear regions of the flow incurred a large computational cost. Godfrey and Cliff [8] explored the use of analytic sensitivity methods (in particular, the direct method) for the computation of static stability derivatives. They computed the $\alpha$ and $\beta$ derivatives for the YB-48 flying wing using a three-dimensional inviscid flow solver. They achieved good accuracy, but no comments were made on computational efficiency. Limache and Cliff [9] followed up this work by examining the use of the same method for the computation of dynamic stability derivatives. They showed the use of dynamic pitching derivatives of an airfoil using a two-dimensional Euler CFD code. This study showed the promise of analytic sensitivity methods for the computation of stability derivatives and demonstrated that multiple stability derivatives, including the rotational derivatives, could be computed from a single steady solution. The study also highlighted the importance of using, at a minimum, Euler CFD to model the shock waves present in transonic flow. The task of extending this method to a three-dimensional RANS flow solver is not easy and would be a significant barrier to the general adoption of this method.

Another avenue that has been pursued is the use of automatic differentiation to compute stability derivatives. Park et al. [10,11] applied ADIFOR [12], an automatic differentiation tool, to a three-dimensional viscous flow solver to compute the static and dynamic derivatives of various configurations. The results from this work showed promise, providing accurate results across a variety of flight conditions. However, the computational cost of running the differentiated code to compute derivatives with respect to five independent states ($\alpha$, $\beta$, $p$, $q$, and $r$) was equivalent to eight flow solutions. Green et al. [13,14] applied a similar technique to the PMARC panel code. This work focused on the development of techniques, such as uncertainty propagation and derivative separation, so the results are of little direct relevance to the current study.

The previous two approaches to dynamic derivative computation (those of Limache and Cliff [9] and Park et al. [10,11]) relied on a noninertial reference frame CFD formulation to handle the rotational rates needed for the dynamic derivatives. Babcock and Arena [15] handled the dynamic derivatives by modifying the boundary conditions in a finite-element-based Euler CFD solver to separate the velocity and position boundary conditions. With this approach, they were able to perturb the static states ($\alpha$, $\beta$) and the dynamic states ($p$, $q$, $r$) independently to determine the stability derivatives using finite differences. The results from this approach compare relatively well with theoretical, empirical, and experimental results, confirming the validity of the method. However, no details on computational cost are included in the study.

Another way of computing the dynamic derivatives, one that has been used in the experimental community for many years, is the forced-oscillation approach. There has been a recent resurgence in interest in this technique, as it can be used with CFD. A number of papers from the recent NATO RTO Task Group AVT-161 have explored the use of forced-oscillation techniques with a variety of CFD solvers [16], including RANS [17,18], DES [19,20], and harmonic balance [21] solvers. The results in these papers were shown to correlate well with experimental data.

Murman [22] also presented a method for computing stability derivatives based on the traditional forced-oscillation approach. He used a frequency-domain method to produce periodic data for the forced oscillation of the configuration of interest. The data were then analyzed with the same techniques used to produce stability derivatives from forced-oscillation wind-tunnel data, which allowed the method to take advantage of the large body of knowledge in that field. Murman’s study demonstrated good accuracy for a wide variety of configurations and flight conditions. However, because of the time periodic nature of the solution, the computational cost was higher than for an equivalent steady-state solution of the same configuration.

The work presented here builds on the work of Limache and Cliff [9] and Park et al. [10,11]. We apply the automatic differentiation adjoint (ADjoint) approach, which we previously developed [23], to a moving-grid CFD formulation (which is equivalent to the noninertial formulation used by both Limache and Cliff [9] and Park et al. [10,11]) to compute the stability derivatives. As in the previous work of those authors, this CFD formulation allows both the static and dynamic derivatives to be computed from a single steady flow solution. The main advantages of this approach are that it combines the computational efficiency of analytic sensitivity methods with the relatively straightforward implementation of automatic differentiation. This enables the rapid development of an efficient method for the computation of stability derivatives. As we will show, we are able to compute a complete set of static and dynamic stability derivatives for roughly seven times the cost of a single steady flow solution. Note that while the current method is implemented for the Euler equations, the extension to the RANS equations does not require significant new insights. Given the nature of our approach, the inclusion of the RANS terms is a straightforward extension of the presented method.

### III. Theory

The stability derivative formulation described in this work is based on two key components. The first is a CFD code that can compute solutions for rotating geometries. This can be accomplished with either a noninertial reference frame formulation or a moving-grid formulation. The moving-grid formulation is used in this work and will be discussed in Sec. III.A; The noninertial formulation can be found in Limache and Cliff [9] or and Park and Green [11]. The second key component of the stability derivative formulation is an efficient, robust, and accurate sensitivity analysis method for the CFD. In our case, this comes in the form of the ADjoint method. A brief summary of this method is provided in Sec. III.C, with more details available in previous work by the authors [23].

#### A. CFD for Rotating Geometries

In this research, our goal is to compute the derivatives for a given configuration from a single flow solution. To accomplish this for both static ($\alpha$, $\beta$, $V$) and dynamic ($p$, $q$, $r$) parameters, we need a flow solver that can compute steady-state solutions for constant,
nonzero values of each of the parameters. Most CFD solvers can perform this computation for a range of static parameters, but few CFD solvers can handle nonzero dynamic parameters.

To handle nonzero dynamic parameters we use a moving-grid formulation. The flow solution is computed using the global velocities as the flow velocities. These velocities are expressed in terms of the moving-grid base vectors. This transformation introduces additional terms into the governing equations that account for the moving coordinates of the grid, and eliminates the need to add the centrifugal and Coriolis forces as source terms in the momentum equations, as required in the noninertial formulation. The moving-grid formulation is derived below.

1. Moving-Grid Formulation

To begin, we define three velocities, \( u, v, \) and \( w \), such that
\[
u = u + w
\]
where \( u \) is the velocity of the fluid in the fixed frame, \( v \) is the velocity of the fluid with respect to the moving grid, and \( w \) is the velocity of the moving grid. Using the approach of Warsi [24], we can write a general form of the conservation law in moving coordinates as
\[
\frac{\partial A}{\partial t} - (V A) \cdot w + \nabla \cdot F = C
\]
where, for the conservation of mass, momentum, and energy, \( F \) takes the form
\[
F = Au + B, \quad A, B, C \text{ and } \tau
\]
and \( \tau \) can represent various quantities, depending on which quantity is being conserved. In addition, from Warsi [24], we have the identity,
\[
\nabla \cdot (Au) = (V A) \cdot w + A(\nabla \cdot w)
\]
which can be derived by applying the product rule to the left-hand side and rearranging the components. This identity can then be rearranged as follows:
\[
(V A) \cdot w = \nabla \cdot (Au) - A(\nabla \cdot w)
\]
Using this relationship in the conservation law (2), we can write
\[
\frac{\partial A}{\partial t} - \nabla \cdot (Au) + A(\nabla \cdot w) + \nabla \cdot F = C
\]
Now substituting \( F \) as defined above, we get
\[
\frac{\partial A}{\partial t} - \nabla \cdot (Au) + A(\nabla \cdot w) + \nabla \cdot (Au + B) = C
\]
Rearranging the above equation, we obtain
\[
\frac{\partial A}{\partial t} + \nabla \cdot (Au - Aw) + A(\nabla \cdot w) + \nabla \cdot (B) = C
\]
Since \( \nabla \cdot w = 0 \), i.e., the grid is incompressible, this equation simplifies to
\[
\frac{\partial A}{\partial t} + \nabla \cdot (Au - Aw) + \nabla \cdot (B) = C
\]
Following the work of Warsi [24] and Ghosh [25], we can then use this conservation equation for mass, momentum, and energy as follows:

a. Mass Conservation. For mass conservation, \( A = \rho, B = 0, C = 0 \), and \( \tau = t \). This yields
\[
\frac{\partial \rho}{\partial t} + \nabla \cdot (\rho u - \rho w) + \nabla \cdot (0) = 0
\]
which can be simplified to
\[
\frac{\partial \rho}{\partial t} + \nabla \cdot (\rho(u - w)) = 0
\]
b. Momentum Conservation. In this case, \( A = \rho u, B = pl - \sigma, C = 0 \), and \( \tau = t \), which yields
\[
\frac{\partial \rho u}{\partial t} + \nabla \cdot (\rho u \otimes u - \rho u \otimes w) + \nabla \cdot (pl - \sigma) = 0
\]
Rearranging this equation, we obtain
\[
\frac{\partial \rho u}{\partial t} + \nabla \cdot (\rho u \otimes u - \rho u \otimes w + pl - \sigma) = 0
\]
Up to this point, the derivation has been general. Now we cast it in the specific moving-grid base vectors to obtain
\[
\frac{\partial \rho u^i}{\partial t} + \nabla \cdot (\rho u \otimes u - \rho u \otimes w + pl - \sigma) = 0
\]
Since it can be shown that \( \partial a_i / \partial t - \partial u^i / \partial x^j = 0 \), we can write
\[
\frac{\partial \rho u^i}{\partial t} + \rho u^i + \nabla \cdot (\rho u \otimes u - \rho u \otimes w + pl - \sigma) = 0
\]
Furthermore, if we let \( w = \Omega \times x \), it can be shown that \( \rho u \otimes w / \partial x^j = \rho (\omega \times u) \), and therefore in the moving-grid coordinate system the momentum equations are
\[
\frac{\partial \rho u^i}{\partial t} + \nabla \cdot (\rho u \otimes u - \rho u \otimes w + pl - \sigma) + \rho (\omega \times u) = 0
\]
c. Energy Conservation. To obtain the energy conservation equations, we set \( A = \rho e, B = (pl - \sigma) \cdot u + k\nabla T, C = 0 \), and \( \tau = t \), which yields
\[
\frac{\partial \rho e}{\partial t} + \nabla \cdot (\rho e u - \rho e w) + \nabla \cdot ((pl - \sigma) \cdot u + k\nabla T) = 0
\]
Rearranging this equation, we obtain
\[
\frac{\partial \rho e}{\partial t} + \nabla \cdot (\rho e u - \rho e w) + (pl - \sigma) \cdot u + k\nabla T = 0
\]
where the total energy is given by \( e_i = p(\gamma - 1) + 1/2|u|^2 \).

d. Final Formulation. Combining Eqs. (10), (15), and (17) yields the following set of governing equations:
\[
\frac{\partial \rho}{\partial t} + \nabla \cdot (\rho (u - w)) = 0
\]
\[
\frac{\partial \rho u}{\partial t} + \nabla \cdot [\rho u \otimes u - \rho u \otimes w + pl - \sigma] + \rho (\omega \times u) = 0
\]
\[
\frac{\partial \rho e}{\partial t} + \nabla \cdot [\rho e u - \rho e w + pu - \sigma u + k\nabla T] = 0
\]
Note that we have neglected the body forces in this derivation. Inclusion of body forces can be accomplished by setting a nonzero value of \( C \) in each case.

e. Flux Vector Form. If we restrict ourselves to just the inviscid portion of the equations, the flux vector form of the governing equations is
\[
\frac{\partial \xi}{\partial t} + \frac{\partial f_i}{\partial x_j} + s = 0
\]
\[
\begin{align*}
\mathbf{\zeta} &= \begin{bmatrix}
\rho \\
\rho u_1 \\
\rho u_2 \\
\rho u_3 \\
\rho e_1
\end{bmatrix}, \\
\mathbf{f}_1 &= \begin{bmatrix}
\rho u_1 - \rho w_1 \\
\rho u_1 u_1 - \rho w_1 u_1 + p \delta_{11} \\
\rho u_2 u_2 - \rho w_1 u_2 + p \delta_{12} \\
\rho u_3 u_3 - \rho w_1 u_3 + p \delta_{13} \\
\rho u_1 (e_1 + p) - \rho w_1 e_1
\end{bmatrix}
\end{align*}
\]

\[
\begin{align*}
s &= \begin{bmatrix}
0 \\
\rho \alpha_2 u_1 - \rho \alpha_3 u_2 \\
\rho \alpha_1 u_1 - \rho \alpha_2 u_3 \\
\rho \alpha_1 u_2 - \rho \alpha_2 u_1 \\
0
\end{bmatrix}
\end{align*}
\]

where \( \mathbf{w} = \mathbf{w}_0 + \Omega \times \mathbf{x} \).

This is the formulation implemented in SUmb [26] and used in this work. SUmb is a cell-centered multiblock solver for the Reynolds-averaged Navier–Stokes equations (steady, unsteady, and time-spectral), and it has options for a variety of turbulence models with one, two, and four equations. In this work, we limit ourselves to solving the steady Euler equations.

2. Constant-Parameter Motions

Having developed the ability to compute solutions for rotating configurations, we now consider how to compute the required derivatives. To accomplish this, we develop a series of motions that allow for the variation of a single parameter while holding all other parameters constant. Consider pitch rate, \( \dot{\alpha} \). As described by Limache and Cliff [9], a loop performed at constant \( \dot{\alpha} \) for a given \( \dot{\alpha} \) generates a steady solution. The radius of the loop is inversely proportional to \( \dot{\alpha} \). Thus, as \( \dot{\alpha} \) reduces to zero, the radius approaches infinity and steady level flight is recovered.

Similar ideas can be applied to roll rate \( \dot{\beta} \) and yaw rate \( \dot{\gamma} \). However, in these two other cases, achieving a steady rotating flow is more complicated. When the body axis of the configuration coincides with the wind axis, the logic is the same as for pitching motion. However, if we incline the body axis at an angle of attack, \( \alpha \), relative to the wind, rotation about the body yaw and roll axes no longer yields a

![Fig. 1](image-url)
steady-state solution. In this case, rotation about the wind axis is required to generate a steady solution. This requirement is not of great importance for small values of angle of attack, but should be considered for large values.

The results presented in Sec. IV.D show the effect of the two different reference frames. The results presented below are all computed for small angles of attack, so we use the wind-axis reference frame for the computation of the derivatives.

3. Grid-Motion Considerations

We selected the moving-grid formulation for this implementation of the stability derivatives, since it was already implemented in SUmb. However, the methods discussed in this paper would apply equally well to a noninertial reference frame formulation. One key observation is that regardless of the chosen formulation, the grid motion must be such that the magnitude of the grid velocity is independent of the rotational rate. More specifically, the grid motion must be specified such that the velocity of the flow at the center of rotation is that of the desired freestream velocity. This condition falls out naturally from the noninertial reference frame formulation, because the velocity is specified in terms of the local grid. For the moving-grid formulation, the correct way to implement this condition is not so obvious. To do so, one needs to specify a grid velocity that is linked to the rotational velocity, such that the velocity of grid’s center of rotation is independent of the rotational speed.

B. Verification of Moving-Grid Formulation

To verify the implementation of the moving-grid formulation, we compare results for a NACA 0012 airfoil rotating at a finite \( \dot{\alpha} \) to those produced by Limache [27]. In this comparison, we simulate inviscid flow around a NACA 0012 airfoil at Mach = 0.2 for \( \dot{\alpha} = 0, 0.01, 0.03, \) and 0.05.

The \( C_p \) distributions around the airfoil are shown in Fig. 1 for both the present and reference results. This figure provides visual verification that the moving-grid formulation implemented in SUmb is consistent with the noninertial reference frame formulation used by Limache [27]. For \( \dot{\alpha} \) values of both 0.0 and 0.05, the \( C_p \) distributions and streamlines of relative velocity match those presented by Limache.

In Fig. 2 we show the pressure coefficient contours and streamlines of relative velocity in the whole computational domain for \( \dot{\alpha} = 0.05. \) Note that in both the present work and the reference work, the point of zero relative velocity occurs at the expected coordinates, (0, –20).

Finally, in Table 1 we compare the values of \( C_l \) and \( C_m \) from our implementation to the references results. The two implementations match very well over a range of \( \dot{\alpha} \) values. The largest discrepancy is 0.011 in \( C_l \) at Mach = 0.8 and \( \dot{\alpha} = 0.05. \) This close correlation is further confirmation that the formulation is correct.

C. ADjoint Approach

Having modified the CFD code to handle rotating geometries, we can now differentiate the code to obtain stability derivatives. To do this efficiently, we use the ADjoint method, which we have developed in previous work [23]. In this approach, automatic differentiation techniques are combined with the adjoint method to generate the sensitivities for the CFD equations. The application of this method to the computation of stability derivatives is described in this section.

We start by considering the functions of interest, \( I \), which may be either the forces \( (C_L, C_D, C_V) \) or moments \( (C_l, C_m, C_n) \) acting on the aircraft. These are functions of both the states of the system (\( \zeta \)) and the values of the independent variables (\( \alpha \)). In this case, the independent variables are the states of the dynamic model (\( \alpha, \beta, V, p, q, r, h, \delta, \) etc.). The function of interest can be expressed as

![Fig. 2](image)

**Fig. 2** \( C_p \) contour and streamline comparison between present work (left) and Limache [27] (right) for rotating NACA 0012 at Mach = 0.2, \( \alpha = 0, \) and \( \dot{\alpha} = 0.05. \)

<table>
<thead>
<tr>
<th>Mach</th>
<th>Coefficient</th>
<th>( \dot{\alpha} = 0.00 )</th>
<th>( \dot{\alpha} = 0.01 )</th>
<th>( \dot{\alpha} = 0.03 )</th>
<th>( \dot{\alpha} = 0.05 )</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.2</td>
<td>( C_l )</td>
<td>–0.001 (0.000)</td>
<td>–0.053 (–0.053)</td>
<td>–0.156 (–0.157)</td>
<td>–0.260 (–0.262)</td>
</tr>
<tr>
<td></td>
<td>( C_m )</td>
<td>0.000 (0.000)</td>
<td>–0.018 (–0.018)</td>
<td>–0.053 (–0.053)</td>
<td>–0.088 (–0.088)</td>
</tr>
<tr>
<td>0.5</td>
<td>( C_l )</td>
<td>0.000 (0.000)</td>
<td>–0.060 (–0.060)</td>
<td>–0.179 (–0.180)</td>
<td>–0.298 (–0.299)</td>
</tr>
<tr>
<td></td>
<td>( C_m )</td>
<td>0.000 (0.000)</td>
<td>–0.020 (–0.020)</td>
<td>–0.060 (–0.060)</td>
<td>–0.100 (–0.100)</td>
</tr>
<tr>
<td>0.8</td>
<td>( C_l )</td>
<td>0.000 (0.000)</td>
<td>–0.107 (–0.108)</td>
<td>–0.310 (–0.316)</td>
<td>–0.487 (–0.498)</td>
</tr>
<tr>
<td></td>
<td>( C_m )</td>
<td>0.000 (0.000)</td>
<td>–0.041 (–0.042)</td>
<td>–0.121 (–0.124)</td>
<td>–0.195 (–0.201)</td>
</tr>
</tbody>
</table>

*Results from Limache [27] are in parentheses.*
We can also represent the solution of the CFD equations as a set of governing equations that are functions of the states and the independent variables. The residuals of these equations can be written as

\[ \mathbf{R} (x, \zeta(x)) = 0 \quad (24) \]

To derive the adjoint equations for this system, we first write the total derivative for both the function of interest (23) and the residuals (24), which yields

\[
\frac{dI}{dx} = \frac{\partial I}{\partial x} + \frac{\partial I}{\partial \zeta} \frac{d\zeta}{dx} \quad (25)
\]

and

\[
\frac{d\mathbf{R}}{dx} = \frac{\partial \mathbf{R}}{\partial x} + \frac{\partial \mathbf{R}}{\partial \zeta} \frac{d\zeta}{dx} = 0 \quad (26)
\]

because \( \mathbf{R} = 0 \), regardless of \( x \), when the governing equations are satisfied. Therefore, we can rewrite the total derivative from Eq. (26) as

\[
\frac{d\zeta}{dx} = -\left[ \frac{\partial \mathbf{R}}{\partial \zeta} \right]^{-1} \frac{\partial \mathbf{R}}{\partial x} \quad (27)
\]

Combining Eqs. (25) and (27), we obtain

\[
\frac{dI}{dx} = \frac{\partial I}{\partial x} - \frac{\partial I}{\partial \zeta} \left[ \frac{\partial \mathbf{R}}{\partial \zeta} \right]^{-1} \frac{\partial \mathbf{R}}{\partial x} \quad (28)
\]

We now have the total derivative \( dI/dx \) expressed in terms of four partial derivatives that do not require a solution of the residual equations in their computation. Instead, to compute the total derivative \( dI/dx \), we must perform a series of linear solutions. In our case, we solve the adjoint system,

\[
\left[ \frac{\partial \mathbf{R}}{\partial \zeta} \right]^T \Psi = -\frac{\partial I}{\partial \zeta} \quad (29)
\]

which requires a separate linear solution for each component of \( I \). The other option, the direct method, involves solving

\[
\frac{\partial I}{\partial \zeta} \frac{d\zeta}{dx} = -\frac{\partial \mathbf{R}}{\partial x} \quad (30)
\]

which requires a separate linear solution for each component of \( x \).

The relative efficiency of the two approaches depends on the relative sizes of \( I \) and \( x \). In this case, the size of \( I \) is six and the size of \( x \) is six or more, depending on the number of control derivatives required. Therefore, it is slightly more favorable to use the adjoint method, but the relative numbers are close enough that there is no significant advantage. In our case, the adjoint approach is used because it has
already been implemented to compute the derivatives for design optimization [28]. In design optimization, the number of design variables generally exceeds the number of functions of interest, and the adjoint approach is decidedly advantageous.

The other consideration associated with Eq. (28) is how to calculate the four partial derivatives that make up the expression. This is where automatic differentiation is used. One of the most significant drawbacks of the adjoint method outlined above is that the calculation of the partial derivatives making up the expression can be extremely complex. In many cases, such as those involved in complex CFD schemes, the effort required to differentiate the code used to compute the residuals is tremendous. By using automatic differentiation to compute these derivatives, the amount of effort required to complete the differentiation is significantly reduced. In addition, no approximations are made in the differentiation, and as a result, the derivatives computed with the ADjoint method are extremely accurate.

D. Verification of the ADjoint for Stability Derivatives

To verify the ADjoint implementation for the computation of stability derivatives, we reproduce the NACA 0012 airfoil case of Limache and Cliff [9]. We use a pseudo-two-dimensional mesh, which is two cells in the spanwise direction, with symmetry planes at the ends of the wing segment to reproduce two-dimensional flow. Each slice of the mesh contains 65,536 cells for a total mesh size of 131,072 cells. The flow is simulated at \( \alpha = 0 \) and Mach = 0.1, 0.5 and 0.8, with the flow solutions converged to a relative convergence tolerance of \( 10^{-12} \). We make two comparisons: one to verify consistency within our implementation, and another to verify our results against those of Limache and Cliff.

To verify the consistency within our implementation, we compare the ADjoint derivatives against derivatives computed using the complex-step derivative approximation [29]. The purpose of this verification is to show that the ADjoint implementation accurately computes the derivative of the code in the SUmb solver. Using the complex-step approach, the derivatives are computed by

\[
\frac{dI(x)}{dx} = \frac{\text{Im}[I(x + ih)]}{h} + O(h^2)
\]

where \( i = \sqrt{-1} \). This approximation is not subject to the subtractive cancellation errors inherent in finite differences. Therefore, with a sufficiently small step size (in this case, \( 10^{-20} \)), the method is able to produce derivatives with the same accuracy as the flow solution, making it a solid benchmark for our results.

The ADjoint derivatives are compared with the complex-step results in Table 2, where we can see that the results match by 9 to 11 digits. This is an extremely accurate match, far beyond the accuracy of the underlying physical model used. Furthermore, given the iterative nature of the solvers used, the accuracy is consistent with the \( 10^{-12} \) relative convergence tolerance that was achieved.

The verification against the results of Limache and Cliff [9] is shown in Table 3. This comparison is done to show that our method accurately captures the dependencies of the coefficients on \( q \). As can be seen in Table 3, the code from this work is accurate relative to the
reference implementation of Limache and Cliff [9]. For the Mach = 0.1 and 0.5 cases, the differences relative to the reference results are less than 1%. In the Mach = 0.8 case, $C_m$ differs by approximately 5%. This larger discrepancy can be attributed to the fact that there is a shock wave in the solution for Mach = 0.8. The precise location of the shock has a significant impact on the value of the moment coefficient and hence on the moment coefficient derivatives as well. Given that SUMb is a structured multiblock code and that the reference results were computed with an unstructured code, slight variations in the prediction of the shock location are expected. Based on these results, we conclude that the stability derivatives predicted by the code are correct for the NACA 0012 airfoil case.

IV. ONERA M6 Stability Derivatives

To verify the derivative computation in three dimensions, we analyze the ONERA M6 wing, which is a common test case in the CFD community [30]. The configuration is a simple swept wing; the geometry parameters for the wing are listed in Table 4.
In this study, we compare the derivatives for the ONERA M6 wing at Mach = 0.1 against those calculated with PMARC, a well-established panel code used for low-speed flow prediction [31]. The comparisons show the values of various coefficients and derivatives for a variety of flight conditions (i.e., various values of $\alpha$ and $\beta$) and rotational center locations ($x_{ref}$). The purpose of these comparisons is to show that the proposed method accurately captures the various dependencies in the stability derivative computation.

Finally, we present derivative values for the ONERA M6 wing at Mach = 0.8395 and $\alpha = 3.06$ deg, one of the flight conditions tested by AGARD [30]. These transonic derivatives are intended to become reference values for future work. Note that the rotational derivatives are expressed in terms of normalized rotational rates $\dot{p}$, $\dot{q}$, and $\dot{r}$. The derivatives are expressed in a typical body-axis reference frame with the $x$ axis pointing forward and the $z$ axis pointing down. Sum computations are conducted in a reference frame with the $x$
axis pointing in the downstream direction and the z axis pointing out toward the left half of the wing.

### A. Mesh Convergence

To verify the stability derivative formulation discussed in the present work, solutions from PMARC are compared against solutions from SUmb.

The PMARC mesh is shown in Fig. 3a. Each wing half is composed of three patches: one that wraps around the leading edge forming the wing surface, another for the wing tip, and a third one for the wake, which is attached to the trailing edge. The wake patch is rigid and aligned with the freestream direction.

To demonstrate the numerical accuracy of the PMARC solutions, a convergence study was performed by increasing the number of cells per side in each patch from 5 to 20 in increments of 5. The wake patches and wing tip patches have the same number of cells in each direction, while the main wing patches have twice as many cells in the chordwise direction as in the spanwise direction. The convergence of the $C_L$ and $C_{L/\alpha}$ values for this series of meshes is shown in Fig. 4. This plot shows that $C_L$ and $C_{L/\alpha}$ converge as the meshes are refined. For the PMARC results, the error between the two finest meshes is 1.5% for $C_L$ and 1.6% for $C_{L/\alpha}$.

A sample of the SUmb mesh is shown in Fig. 3b. It is an H-H mesh for which the wing tip is closed with a rounded tip, and the trailing edge of the wing has zero thickness. The far-field boundary is approximately 30 mean aerodynamic chords away from the wing. To quantify the numerical accuracy of the solution, a series of four meshes have been generated. Each mesh is exactly 8 times larger than the previous mesh, which is the result of doubling the mesh points for each edge. The meshes have a total of 28,000, 228,000, 1.8 million, and 14.7 million cells, respectively. The offwall spacing for the 14.7-million-cell mesh is $1 \times 10^{-3}$ m. The leading-edge spacing is also $1 \times 10^{-3}$ m and the trailing-edge spacing is $5 \times 10^{-3}$ m. The mesh shown in Fig. 3b is the 228,000 cell mesh. $C_L$ and $C_{L/\alpha}$ convergence results for this series of meshes are shown in Fig. 5. These plots show that the difference between the finest two meshes is 0.7% for $C_L$ and 1.1% for $C_{L/\alpha}$. The fact that the error for the PMARC results is on the order of 2% and the error for the SUmb results is on the order of 1% gives confidence in the quality of the results presented. The plots shown in Secs. IV.B–IV.D are generated using the 20-cell-per-side PMARC meshes and the 1.8-million-cell SUmb mesh. The benchmark derivative results presented in Sec. IV.E are calculated with the 14.7-million-cell SUmb mesh.

### B. Low-Speed Verification: Longitudinal Derivatives

The first verification we show is the variation of the longitudinal coefficients ($C_L$, $C_{\alpha}$) and their derivatives with respect to $\alpha$. Figure 6 shows the variation the coefficients and Fig. 7 shows the variation of the various derivatives, both with respect to $\alpha$. We then compare the performance of these same coefficients and derivatives for a variety of longitudinal reference points, $x_{ref}$. This reference point location acts as both the center of moment and the center of rotation. Figure 8 shows the variation of the coefficients with respect to $x_{ref}$ and Fig. 9 shows the variation of the derivatives with respect to the same point.
As we can see from these figures, the longitudinal characteristics of the wing are well captured by our computations. There is essentially no variation between the slopes predicted by SUm and PMARC for the coefficients shown in Figs. 6a and 6b. The comparisons in Fig. 7 show consistent performance over the full range of \( \alpha \)’s tested. The derivative values predicted by PMARC and SUm are slightly different, but the trends for both methods match, leading to a consistent difference between the two values. We see similar trends over the range of \( x_{ref} \) values considered in Figs. 8 and 9. In each case, the trend of the PMARC results closely matches the trend in the SUm result. This even holds for the more complex curved trend shown in Fig. 9b.

From all these comparisons, we conclude that the longitudinal stability derivatives computed by SUm are correct, as long as the physics captured by the Euler equations accurately model the problem of interest.

Fig. 12 ONERA M6: \( C_n \) and derivatives.

Fig. 13 ONERA M6: Comparison of wind-axis versus body-axis computations.
Table 5 Comparison for $C_l$ for wind-axis and body-axis computations

<table>
<thead>
<tr>
<th>$\alpha$</th>
<th>Wind</th>
<th>Body</th>
<th>Error</th>
<th>% error</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>-1.4926016</td>
<td>-1.4926016</td>
<td>0.0000</td>
<td>0.00%</td>
</tr>
<tr>
<td>1</td>
<td>-1.4908012</td>
<td>-1.4929310</td>
<td>0.0021</td>
<td>-0.14%</td>
</tr>
<tr>
<td>2</td>
<td>-1.5116103</td>
<td>-1.5204085</td>
<td>0.0088</td>
<td>-0.58%</td>
</tr>
<tr>
<td>3</td>
<td>-1.5326331</td>
<td>-1.5531322</td>
<td>0.0205</td>
<td>-1.32%</td>
</tr>
<tr>
<td>4</td>
<td>-1.5674002</td>
<td>-1.6063816</td>
<td>0.0390</td>
<td>-2.43%</td>
</tr>
<tr>
<td>5</td>
<td>-1.6402228</td>
<td>-1.7041819</td>
<td>0.0640</td>
<td>-3.75%</td>
</tr>
</tbody>
</table>

Table 6 Comparison of $C_n$ for wind-axis and body-axis computations

<table>
<thead>
<tr>
<th>$\alpha$</th>
<th>Wind</th>
<th>Body</th>
<th>Error</th>
<th>% error</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>2.9516551 x 10^-2</td>
<td>2.9516551 x 10^-2</td>
<td>0.0000</td>
<td>0.00%</td>
</tr>
<tr>
<td>1</td>
<td>2.8696904 x 10^-2</td>
<td>2.719394 x 10^-2</td>
<td>0.0015</td>
<td>5.51%</td>
</tr>
<tr>
<td>2</td>
<td>2.8777435 x 10^-2</td>
<td>2.3128088 x 10^-2</td>
<td>0.0055</td>
<td>23.99%</td>
</tr>
<tr>
<td>3</td>
<td>2.2586010 x 10^-2</td>
<td>1.2211285 x 10^-2</td>
<td>0.0104</td>
<td>84.96%</td>
</tr>
<tr>
<td>4</td>
<td>5.8047916 x 10^-3</td>
<td>-8.7061377 x 10^-3</td>
<td>0.0145</td>
<td>-166.67%</td>
</tr>
<tr>
<td>5</td>
<td>-4.2725415 x 10^-2</td>
<td>-5.6996461 x 10^-2</td>
<td>0.0143</td>
<td>-25.04%</td>
</tr>
</tbody>
</table>

C. Low-Speed Verification: Lateral Derivatives

Next, we examine the lateral coefficients ($C_y$, $C_l$, and $C_n$) and their derivatives. The lateral coefficients are all zero for the symmetric flight condition at $\beta = 0$, and therefore we show the variation of these coefficients for a variety of sideslip angles $\beta$. The variations for $C_y$, $C_l$, and $C_n$ are plotted in Figs. 10a, 11a, and 12a, respectively.

While the values of the coefficients themselves are zero at the symmetric flight condition, the values of their derivatives are not. Therefore, to evaluate the derivatives of the lateral coefficients, we examine the derivatives for a range of $\alpha$ values. In this section, the figures are split into groups by coefficient. Each group shows the variation of the coefficient for a variety of values of $\beta$ at $\alpha$ = 5 degrees, as well as the variation of the coefficient derivatives with respect to $\beta$, $p$, and $r$ for a range of $\alpha$ values. Figure 10 shows the values for $C_y$, Fig. 11 shows the values for $C_l$, and Fig. 12 shows the values for $C_n$.

The variation in the slopes between the SUMb computed coefficients and the PMARC computed coefficients is larger for the lateral derivatives than for the longitudinal derivatives. However, the dominant trends are the same for both sets of results. $C_{yl}$ is essentially constant with $\beta$ for both cases, $C_{yl}$ decreases with $\alpha$, and $C_{yl}$ has a nonlinear dependence on $\alpha$. Similar trends are exhibited by the $C_l$ derivatives. $C_{yn}$ is constant with $\beta$, $C_{yn}$ is constant with $\alpha$, and $C_{yn}$ increases with increasing $\alpha$. Finally, for the $C_n$ derivatives, we see that for both cases $C_{yn}$ is essentially constant with $\beta$, $C_{yn}$ decreases with $\alpha$, and $C_{yn}$ has a curved shape.

In this particular case, the discrepancies between the PMARC derivatives and the SUMb derivatives are largest for the $C_n$ and $C_y$ derivatives. This is due to the fact that both the coefficients and the derivatives are very small, on the order of $10^{-2}$, making it difficult to compute them accurately. Furthermore, the lateral forces are more difficult to predict due to the fact that the ONERA M6 wing has no dihedral. In this case, the lateral forces in inviscid flow are dominated by the flow at the wing tips, which is complicated in asymmetric flow. However, as noted above, even though there are differences in the value predicted by the methods, the trends in the two methods match, indicating that the formulation is capturing the necessary dependencies.

D. Wind-Axis Versus Body-Axis Derivatives

As mentioned in Sec. III A 2, it is not possible to physically isolate the lateral body-axis derivatives of the aircraft for nonzero angles of attack. However, because the ADjoint method calculates the derivatives from a single reference flow solution, we are able to compute derivatives about either the wind axis or the body axis at that reference solution. As long as the computation of the reference solution is possible, the ADjoint approach can linearize about that point, even if the lateral rotation about the body axis would not yield a steady flow solution.

In this section, we compare the wind-axis derivatives to the body-axis derivatives. From this comparison, we can assess the validity of using wind-axis derivatives in place of body-axis derivatives for small angles of attack. In Figs. 13a and 13b, we plot the variation of $C_{yi}$ and $C_{ni}$ versus $\alpha$ for both the wind-axis case and the body-axis case. Tables 5 and 6 show the values of the same two derivatives for $\alpha$ values from 0 to 5 deg for both the body- and wind-axis derivatives at Mach = 0.8395.

From these results, we can see a clear divergence of the two predicted values as the value of $\alpha$ increases. However, for the low values of $\alpha$ seen here, the variation is relatively small, staying under 5% for the $C_{yi}$ derivatives. Based on these results, we conclude that the values of the wind-axis derivatives are accurate enough to be useful. However, for larger angles of attack, a correction would likely be needed to produce useful results.

E. High-Speed Derivative Values

To provide a benchmark for future work, we compute the values of the wind-axis and body-axis stability derivatives for a transonic flight condition. Figures 14a and 14b show the chordwise pressure profiles for the 20 and 90% span positions on the wing, both from experimental data [30] and from the 14.7-million-cell mesh used in
The adjoint system are computed using the PETSc package 2.5 GHz Intel Nehalem Processors with eight cores and 16 GB RAM per node. The timings shown in Table 9 are run on four nodes connected by Infiniband for a total of 32 processors. As we see from the table, the flow solution takes approximately 5 min, while a single ADjoint solution takes only 8 min. The total time for computing all six ADjoint solutions required to evaluate the necessary stability derivatives is only 31 min. At this level of efficiency, the computational cost of computing the derivatives is low enough to be used for design.

VI. Conclusions

In this paper, a method for the computation of static and dynamic stability derivatives of arbitrary configurations was presented. The method combines the ADjoint sensitivity analysis technique with a moving-grid CFD solver to allow for the efficient computation of static and dynamic derivatives. For the longitudinal derivatives, the method was shown to match existing methods to within 1% for subsonic cases and to within 1–5% for transonic cases.

For a more general three-dimensional case, the method compared well with an existing panel code method, PMARC. Both lateral and longitudinal derivatives were examined at Mach = 0.1 and the results showed similar trends for both methods.

Finally, the cost of computing a full set of stability derivatives using this new method was measured. For a 1.8-million-cell case, the flow solution and the six ADjoint solutions required about 30 min on 32 processors (2.5 GHz Intel Nehalem). At this level of computational cost, it is certainly possible to consider using stability derivative data in the design process.

Acknowledgments

The authors are grateful for the funding provided by the Canada Research Chairs program and the Natural Sciences and Engineering Research Chairs program and the Natural Sciences and Engineering Research Chairs program.

Table 7 Body-axis derivatives at Mach = 0.8395, $\alpha = 3.06$, and $x_{ref} = 0.0$ m

<table>
<thead>
<tr>
<th>Parameter</th>
<th>$C_L$</th>
<th>$C_D$</th>
<th>$C_Y$</th>
<th>$C_I$</th>
<th>$C_m$</th>
<th>$C_n$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\alpha$</td>
<td>5.5772</td>
<td>4.5422</td>
<td>$\times 10^{-01}$</td>
<td>$\sim0$</td>
<td>$\sim0$</td>
<td>$-4.0932$</td>
</tr>
<tr>
<td>$\beta$</td>
<td>$-1.5168 \times 10^{-05}$</td>
<td>4.0961</td>
<td>$\times 10^{-06}$</td>
<td>$-6.8243 \times 10^{-03}$</td>
<td>$-1.2667 \times 10^{-01}$</td>
<td>1.4722</td>
</tr>
<tr>
<td>Mach</td>
<td>$7.7872 \times 10^{-01}$</td>
<td>1.3225</td>
<td>$\times 10^{-01}$</td>
<td>$\sim0$</td>
<td>$\sim0$</td>
<td>$-8.3126 \times 10^{-01}$</td>
</tr>
<tr>
<td>$\dot{\rho}$</td>
<td>$-2.3528 \times 10^{-06}$</td>
<td>3.5867</td>
<td>$\times 10^{-07}$</td>
<td>2.4044</td>
<td>$\times 10^{-01}$</td>
<td>$-1.5185$</td>
</tr>
<tr>
<td>$\dot{q}$</td>
<td>1.3474</td>
<td>$\times 10^{01}$</td>
<td>6.0271</td>
<td>$\times 10^{-01}$</td>
<td>$\sim0$</td>
<td>$\sim0$</td>
</tr>
<tr>
<td>$\dot{r}$</td>
<td>1.3981</td>
<td>$\times 10^{-06}$</td>
<td>$-4.2876 \times 10^{-07}$</td>
<td>$-3.3968 \times 10^{-02}$</td>
<td>3.6030</td>
<td>$\times 10^{-01}$</td>
</tr>
</tbody>
</table>

Table 8 Wind-axis derivatives at Mach = 0.8395, $\alpha = 3.06$, and $x_{ref} = 0.00$ m

<table>
<thead>
<tr>
<th>Parameter</th>
<th>$C_L$</th>
<th>$C_D$</th>
<th>$C_Y$</th>
<th>$C_I$</th>
<th>$C_m$</th>
<th>$C_n$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\alpha$</td>
<td>5.5772</td>
<td>4.5422</td>
<td>$\times 10^{-01}$</td>
<td>$\sim0$</td>
<td>$\sim0$</td>
<td>$-4.0932$</td>
</tr>
<tr>
<td>$\beta$</td>
<td>$-1.5168 \times 10^{-05}$</td>
<td>4.0961</td>
<td>$\times 10^{-06}$</td>
<td>$-6.8243 \times 10^{-03}$</td>
<td>$-1.2667 \times 10^{-01}$</td>
<td>1.4722</td>
</tr>
<tr>
<td>Mach</td>
<td>$7.7872 \times 10^{-01}$</td>
<td>1.3225</td>
<td>$\times 10^{-01}$</td>
<td>$\sim0$</td>
<td>$\sim0$</td>
<td>$-8.3126 \times 10^{-01}$</td>
</tr>
<tr>
<td>$\dot{\rho}$</td>
<td>$-2.3528 \times 10^{-06}$</td>
<td>3.5867</td>
<td>$\times 10^{-07}$</td>
<td>2.4044</td>
<td>$\times 10^{-01}$</td>
<td>$-1.5185$</td>
</tr>
<tr>
<td>$\dot{q}$</td>
<td>1.3474</td>
<td>$\times 10^{01}$</td>
<td>6.0271</td>
<td>$\times 10^{-01}$</td>
<td>$\sim0$</td>
<td>$\sim0$</td>
</tr>
<tr>
<td>$\dot{r}$</td>
<td>1.5217</td>
<td>$\times 10^{-06}$</td>
<td>$-4.4730 \times 10^{-07}$</td>
<td>$-4.6747 \times 10^{-02}$</td>
<td>4.4085</td>
<td>$\times 10^{-01}$</td>
</tr>
</tbody>
</table>

Table 9 ADjoint timing for the ONERA M6 case at Mach = 0.8395 and $\alpha = 3.06$ deg

<table>
<thead>
<tr>
<th>Number of processors</th>
<th>267</th>
<th>279</th>
<th>272</th>
<th>258</th>
<th>276</th>
<th>308</th>
<th>1859</th>
</tr>
</thead>
<tbody>
<tr>
<td>Flow solution</td>
<td>267</td>
<td>279</td>
<td>272</td>
<td>258</td>
<td>276</td>
<td>308</td>
<td>1859</td>
</tr>
<tr>
<td>ADjoint solution</td>
<td>466</td>
<td>279</td>
<td>272</td>
<td>258</td>
<td>276</td>
<td>308</td>
<td>1859</td>
</tr>
<tr>
<td>Breakdown:</td>
<td>117</td>
<td>117</td>
<td>117</td>
<td>117</td>
<td>117</td>
<td>117</td>
<td>117</td>
</tr>
<tr>
<td>Computation of residual matrices</td>
<td>26</td>
<td>26</td>
<td>26</td>
<td>26</td>
<td>26</td>
<td>26</td>
<td>26</td>
</tr>
<tr>
<td>Computation of right-hand side</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>6</td>
</tr>
<tr>
<td>Solution of the adjoint equations</td>
<td>321</td>
<td>277</td>
<td>270</td>
<td>256</td>
<td>274</td>
<td>306</td>
<td>1704</td>
</tr>
<tr>
<td>Computation of the total sensitivities</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>6</td>
</tr>
</tbody>
</table>

1With ~1.8 million cells and $10^{-10}$ relative convergence; all times are in seconds.

This study. While there are slight errors in the predicted location of the shock wave in the solution, on the whole, the solutions match well. This comparison shows that the Euler model used in this work is representative of the flow in this case. Furthermore, based on the comparisons shown in the previous section, we conclude that the stability derivative formulation presented in this work is correct and applicable to any flows for which the flow solver produces valid results.

The body-axis derivatives for the transonic test case are listed in Table 7, and the wind-axis derivatives for the same case are listed in Table 8.

V. Computational Performance

In addition to demonstrating the accuracy of the code, it is important to show its computational efficiency. To this end, we examined the time required to compute the various derivatives required in the generation of a simple linear flight dynamic model. The results in Table 9 show the computational time needed to evaluate the six consecutive adjoint problems ($C_L$, $C_D$, $C_Y$, $C_I$, $C_m$, and $C_n$) required to generate all of the static and dynamic derivatives for a given configuration. The linear system solutions required for the adjoint system are computed using the PETSc package, developed at Argonne National Labs [32]. This is a broadly applicable scientific computing package that contains a variety of linear and nonlinear solution methods, as well as a variety of preconditioning options. In this work, we are using the restarted GMRES solver with an additive Schwartz parallel preconditioner. Local preconditioning is accomplished with incomplete lower/upper factorization and a reverse Cuthill–McKee reordering. The computations for this work were performed on the SciNet general-purpose cluster, which uses

1Data available online at http://www.mcs.anl.gov/petsc [retrieved 24 June 2011].
Research Council. Computations were performed on the general-purpose cluster supercomputer at the SciNet high-performance computing consortium, which is funded by the Canada Foundation for Innovation under the auspices of Compute Canada, the Government of Ontario, Ontario Research Fund—Research Excellence, and the University of Toronto. The authors would also like to thank Edwin van der Weide and Juan J. Alonso for their assistance in the early stages of this project and, in particular, with the moving-grid formulation in the SUmb flow solver.

References


W. Anderson
Associate Editor