JOURNAL OF AIRCRAFT
Vol. 43, No. 5, September–October 2006

Validation of Implicit Algorithms for Unsteady Flows Including Moving and Deforming Grids

R. H. Nichols*
University of Alabama at Birmingham, Birmingham, Alabama 35294
and
B. D. Heikkinen†
Aerospace Testing Alliance, Arnold Air Force Base, Tennessee 37389

Time-accurate Navier–Stokes flow solvers utilizing Newton and dual-time-step implicit subiteration algorithms have been investigated for both moving-body and deforming-grid applications. A set of relatively simple two-dimensional validation cases has been identified to assess the performance of these unsteady computational fluid dynamics (CFD) solvers with varying time-step size and number of subiterations. These cases demonstrate the advantages of second-order time derivatives and subiterations for unsteady-flow simulations. This investigation also indicates subiteration algorithms and large time steps can be used to reduce the computational cost of a given unsteady-flow simulation.

Nomenclature

a	=	speed of sound
$\boldsymbol{c}$	=	grid-speed vector
E	=	total energy
$\boldsymbol{E}, \boldsymbol{F}, \boldsymbol{G}$	=	inviscid flux vectors [Eq. (3)]
f	=	frequency
I	=	identity matrix
J	=	Jacobian
M	=	Mach number
m	=	subiteration counter
n	=	time-level counter
$\boldsymbol{n}$	=	surface normal vector
p	=	pressure
$\boldsymbol{Q}$	=	nonconserved variable vector [Eq. (2)]
$\boldsymbol{q}$	=	conserved variable vector [Eq. (2)]
Re	=	Reynolds number
RHS_{GCL}	=	defined in Eq. (15)
T_{ref}	=	reference temperature
t	=	time
u, v, w	=	fluid velocity
u_g, v_g, w_g	=	grid velocity
V	=	volume
$\delta\Omega$	=	control surface
ζ, η, ξ	=	computational coordinates
ζ_t, η_t, ξ_t	=	time metrics
λ	=	eigenvector
ξ_x, ξ_y, ξ_z	=	spatial metrics
ρ	=	density
τ	=	pseudotime variable
Ω	=	control volume
ω	=	vorticity

Introduction

THERE currently exists a great interest in performing calculations using CFD for high-Reynolds-number unsteady flows. Research are beginning to apply the new hybrid Reynolds averaged Navier–Stokes (RANS)/large eddy simulation (LES) class of turbulence models to a host of unsteady high-Reynolds flows containing large-scale coherent turbulent structures. Applications involving moving and deforming bodies are also becoming more common. Validation and verification of CFD codes become more difficult for these unsteady flows than for traditional steady-state problems. Here validation and verification are used in the traditional sense; that is, verification deals with removing coding errors from an algorithm and validation deals with the ability of an algorithm to simulate the physics of a flow. Grid-convergence studies may not address all of the relevant dimensions of a problem because turbulent flows display a cascade of both length and time scales. Often refining the grid results in resolution of smaller-scale structure in the unsteady flow, and hence a "grid-resolved" solution may exist except in the direct numerical simulation (DNS) limit. We hope the relevant physics of the flow can be captured at some level of grid refinement well above the DNS limit. In many unsteady-flow cases convergence can only be judged in a statistical sense. A large number of time steps may be required to reduce the error in computing the statistical parameters of interest.

The choice of time step can have a tremendous effect on the time accuracy of a solution. The high-frequency spectral regime will be underresolved if the chosen time step is too large. Large time steps can introduce error in the solution if no means are provided to local convergence of the solution (i.e., convergence in both time and space at each time step). If the time step is too small a tremendous number of iterations will be required to adequately resolve the low-frequency spectral regime. Hence, the selection of a time step for a typical unsteady CFD problem is an exercise in the art of compromise and often requires some a priori knowledge of the unsteady nature of the flow.

Much of the numerical technology for the solution of the Navier–Stokes equations over the past three decades has been focused on obtaining steady-state solutions. These algorithms generally provide large amounts of numerical dissipation to rapidly damp out spurious numerical fluctuations. Excessive numerical dissipation can cause the unsteady structures in the flow to be overdamped. This study focuses on subiteration strategies that allow for large time steps and local convergence at each step. Large time steps are useful in problems that require grid assembly or grid remeshing at each time step because they minimize the number of these operations required for a simulation. The grid-assembly or grid-remeshing operation is often as computationally expensive as a single time step of the CFD solver. Numerical stability issues can often limit the maximum attainable time step for a given algorithm and has led many developers to investigate the dual-time-stepping approach to time-accurate simulations.

Received 11 April 2005; revision received 31 May 2005; accepted for publication 8 June 2005.

*Research Associate Professor, Department of Mechanical Engineering. Senior Member AIAA.
†Senior Engineer. Senior Member AIAA.

This effort outlines the basic implicit numerical subiteration algorithms required for unsteady-flow applications using large time steps including moving- and deforming-body applications. Simple two-dimensional examples are provided that allow these numerical algorithms to be assessed for unsteady-flow applications. The test cases were chosen based on the availability of analytical solutions to the Navier–Stokes equations and/or a regular periodic behavior of the flow. This allows the user to evaluate the ability of a flow solver to provide a locally converged solution in time and to capture the relevant physics of the flow. These cases also provide insight into the level of dissipation in a given numerical scheme.

Theory

The Navier–Stokes equations may be written in generalized curvilinear coordinates as

$$\frac{\partial \boldsymbol{q}}{\partial t}+\frac{\partial \boldsymbol{E}}{\partial \xi}+\frac{\partial \boldsymbol{F}}{\partial \eta}+\frac{\partial \boldsymbol{G}}{\partial \zeta}=0 \tag{1}$$

where $\boldsymbol{q}$ is the vector of conserved variables

$$\boldsymbol{q}=\boldsymbol{Q}\forall=\boldsymbol{Q}/J=\forall\begin{bmatrix}\rho\\ \rho u\\ \rho v\\ \rho w\\ E\end{bmatrix} \tag{2}$$

The inviscid flux across each face can be written as

$$\boldsymbol{E}=\begin{bmatrix}\rho(u\xi_x+v\xi_y+w\xi_z+\xi_t)\\ \rho u(u\xi_x+v\xi_y+w\xi_z+\xi_t)+p\xi_x\\ \rho v(u\xi_x+v\xi_y+w\xi_z+\xi_t)+p\xi_y\\ \rho w(u\xi_x+v\xi_y+w\xi_z+\xi_t)+p\xi_z\\ (E+p)(u\xi_x+v\xi_y+w\xi_z+\xi_t)-p\xi_t\end{bmatrix} \tag{3}$$

The time metric is given by

$$\xi_t=(u_g\xi_x+v_g\xi_y+w_g\xi_z) \tag{4}$$

The $\boldsymbol{F}$ and $\boldsymbol{G}$ vectors can be constructed from the $\boldsymbol{E}$ vector by replacing ξ with η and ζ respectively.

Geometric Conservation Law (GCL)

If the computational grid is deforming, then the elemental volumes are changing as a function of time. Then, Eq. (1) should be written as

$$\forall\frac{\partial \boldsymbol{Q}}{\partial t}+\boldsymbol{Q}\frac{\partial \forall}{\partial t}+\frac{\partial \boldsymbol{E}}{\partial \xi}+\frac{\partial \boldsymbol{F}}{\partial \eta}+\frac{\partial \boldsymbol{G}}{\partial \zeta}=0 \tag{5}$$

The geometric conservation law (GCL) was first introduced by Thomas and Lombard[1] to address the volume derivative in the second term in Eq. (5). The GCL relates the rate of change of a physical volume to the motion of the volume faces. The GCL is derived by considering a control volume Ω with a control surface $\partial\Omega$ that moves with a velocity $\boldsymbol{c}$ with respect to a stationary inertial reference frame. The continuity equation can be written in integral form as

$$\frac{\mathrm{d}}{\mathrm{d}t}\int_\Omega \rho\,\mathrm{d}\Omega+\oint_{\partial\Omega}\rho(\boldsymbol{V}+\boldsymbol{c})\cdot\boldsymbol{n}\,\mathrm{d}s=0 \tag{6}$$

Assuming uniform conditions everywhere in the field, Eq. (7) reduces to

$$\frac{\mathrm{d}}{\mathrm{d}t}\int_\Omega \mathrm{d}\Omega=-\oint_{\partial\Omega}\boldsymbol{c}\cdot\boldsymbol{n}\,\mathrm{d}s \tag{7}$$

Eq. (7) represents the GCL in integral form. The differential form of the equation would be

$$\frac{\partial \forall}{\partial t}+\frac{\partial \xi_t}{\partial \xi}+\frac{\partial \eta_t}{\partial \eta}+\frac{\partial \zeta_t}{\partial \zeta}=0 \tag{8}$$

Discretizing the time derivative in Eq. (1) using first- or second-order backward differences gives

$$\frac{\Delta\boldsymbol{q}^{n+1}-[\theta_2/(1+\theta_2)]\Delta\boldsymbol{q}^n}{\Delta t}+\frac{1}{1+\theta_2}RHS^{n+1}=0 \tag{9}$$

where

$$RHS=\frac{\partial \boldsymbol{E}}{\partial \xi}+\frac{\partial \boldsymbol{F}}{\partial \eta}+\frac{\partial \boldsymbol{G}}{\partial \zeta} \tag{10}$$

and $\theta_2=0$ for first-order time or $\theta_2=\frac{1}{2}$ for second-order time. The delta quantities can be redefined as shown by Hyams[2] using Eq. (2) to yield

$$\Delta\boldsymbol{q}^{n+1}=(\boldsymbol{Q}\forall)^{n+1}-(\boldsymbol{Q}\forall)^n=\forall^{n+1}\Delta\boldsymbol{Q}^{n+1}+\boldsymbol{Q}^n\Delta\forall^{n+1} \tag{11}$$

$$\Delta\boldsymbol{q}^{n}=(\boldsymbol{Q}\forall)^{n}-(\boldsymbol{Q}\forall)^{n-1}=\forall^{n-1}\Delta\boldsymbol{Q}^{n}+\boldsymbol{Q}^n\Delta\forall^{n} \tag{12}$$

Substituting Eqs. (11) and (12) into Eq. (10) yields

$$\frac{\forall^{n+1}\Delta\boldsymbol{Q}^{n+1}-[\theta_2/(1+\theta_2)]\forall^{n-1}\Delta\boldsymbol{Q}^n}{\Delta t}+\boldsymbol{Q}^n\left[\frac{\Delta\forall^{n+1}-[\theta_2/(1+\theta_2)]\Delta\forall^n}{\Delta t}\right]+\frac{1}{1+\theta_2}RHS^{n+1}=0 \tag{13}$$

Using the differencing expression from Eq. (13) to approximate the time derivative in Eq. (8) yields

$$\frac{\Delta\forall^{n+1}-[\theta_2/(1+\theta_2)]\Delta\forall^n}{\Delta t}=\frac{-1}{1+\theta_2}RHS_{\mathrm{GCL}}^{n+1} \tag{14}$$

where

$$RHS_{\mathrm{GCL}}=\frac{\partial \xi_t}{\partial \xi}+\frac{\partial \eta_t}{\partial \eta}+\frac{\partial \zeta_t}{\partial \zeta} \tag{15}$$

Substituting the right-hand side of Eq. (14) into Eq. (13) yields

$$\frac{(1+\theta_2)\forall^{n+1}\Delta\boldsymbol{Q}^{n+1}-\theta_2\forall^{n-1}\Delta\boldsymbol{Q}^n}{\Delta t}+\boldsymbol{Q}^nRHS_{\mathrm{GCL}}^{n+1}+RHS^{n+1}=0 \tag{16}$$

The flux-like terms in the RHS_{GCL}^{n+1} function [Eq. (15)] need to be constructed in a similar manner and to the same spatial accuracy as the inviscid fluxes in the flow equations to ensure that the GCL is satisfied and that spurious source terms caused by the volume changes are eliminated. The temporal order of accuracy is given by the time term in Eq. (16). Note that this implicit formulation requires that the volumes from two time levels back in the computation be used for second-order time accuracy. This requires that three time levels of grid be stored. The GCL terms must also be included in all transport equations (i.e., species, turbulence, etc.).

Newton Iteration

Equation (16) can be solved using Newton's method following the work of Hyams.[2] First, a function F is defined as

$$F_0(\boldsymbol{Q}^{n+1})=\frac{(1+\theta_2)\forall_0^{n+1}\Delta\boldsymbol{Q}_0^{n+1}-\theta_2\forall_0^{n+1}\Delta\boldsymbol{Q}_0^{n+1}}{\Delta t}+\boldsymbol{Q}^{n+1}RHS_{0,\mathrm{GCL}}^{n+1}+RHS_0^{n+1} \tag{17}$$

Here the 0 subscript denotes the vertex for node-centered schemes or the cell center for cell-centered schemes. The quantity F^{n+1} is the function that should be driven to zero in the Newton iteration. Expanding F^{n+1} in a Taylor series from a known level $n+1, m$ yields

$$F_0^{n+1,m+1}=F_0^{n+1,m}+\frac{\partial F_0^{n+1,m}}{\partial t}\Delta t+O(\Delta t^2) \tag{18}$$

Dropping the $O(\Delta t^2)$ error term and linearizing the time derivative yields

$$F_0^{n+1,m+1} = F_0^{n+1,m} + \frac{\partial F_0^{n+1,m}}{\partial \boldsymbol{Q}} \Delta \boldsymbol{Q}^{n+1,m+1} \tag{19}$$

Because the LHS of Eq. (19) is zero at Newton convergence,

$$-F_0^{n+1,m} = \frac{\partial F_0^{n+1,m}}{\partial \boldsymbol{Q}} \Delta \boldsymbol{Q}^{n+1,m+1} \tag{20}$$

where

$$\Delta \boldsymbol{Q}^{n+1,m+1} = \boldsymbol{Q}^{n+1,m+1} - \boldsymbol{Q}^{n+1,m} \tag{21}$$

Equation (20) can then be expanded to

$$\begin{aligned} -\Bigg[& \frac{(1+\theta_2)\forall_0^{n+1}\left(\boldsymbol{Q}_0^{n+1,m} - \boldsymbol{Q}_0^{n}\right) - \theta_2 \forall_0^{n-1} \Delta \boldsymbol{Q}_0^{n-1}}{\Delta t} \\ & + \boldsymbol{Q}_0^n RHS_{0,\mathrm{GCL}}^{n+1} + RHS_0^{n+1,m} \Bigg] \\ = & \left[\frac{(1+\theta_2)\forall_0^{n+1} I}{\Delta t} + \frac{\partial RHS_0^{n+1,m}}{\partial \boldsymbol{Q}_0} \right] \Delta \boldsymbol{Q}_0^{n+1,m+1} \\ & + \left[\frac{\partial RHS_0^{n+1,m}}{\partial \boldsymbol{Q}_i} \right] \Delta \boldsymbol{Q}_i^{n+1,m+1} \end{aligned} \tag{22}$$

In terms of the general matrix form, $\boldsymbol{Ax} = \boldsymbol{b}$, the first bracketed term in Eq. (22) is the right-hand side vector $\boldsymbol{b}$. The second bracketed term in Eq. (22) represents the diagonal terms of matrix $\boldsymbol{A}$, and the third bracketed term represents the off-diagonal terms of matrix $\boldsymbol{A}$. The computational work required to complete one subiteration is equivalent to the work required for one time step without the Newton scheme.

Dual-Time Stepping

Equation (16) can also be solved using a dual-time-stepping method following the work of Pandya et al.[3] An artificial time (τ) term is explicitly added to Eq. (1) to yield

$$\frac{\partial \boldsymbol{q}}{\partial \tau} + \frac{\partial \boldsymbol{q}}{\partial t} + \frac{\partial \boldsymbol{E}}{\partial \xi} + \frac{\partial \boldsymbol{F}}{\partial \eta} + \frac{\partial \boldsymbol{G}}{\partial \zeta} = 0 \tag{23}$$

The pseudotime iteration (subiteration) is performed at each physical time step. The pseudotime iteration must be converged (i.e., $\partial \boldsymbol{q}/\partial \tau = 0$) at each physical time step to assure time accuracy. Local time stepping, multigrid methods, or other stability-enhancing techniques can be used in conjunction with the pseudotime iteration as long as a local convergence is obtained at each time step. Taking into account grid deformation, Eq. (23) can be rewritten as

$$\frac{\partial \boldsymbol{q}}{\partial \tau} + \forall \frac{\partial \boldsymbol{Q}}{\partial t} + \boldsymbol{Q} \frac{\partial \forall}{\partial t} + \frac{\partial \boldsymbol{E}}{\partial \xi} + \frac{\partial \boldsymbol{F}}{\partial \eta} + \frac{\partial \boldsymbol{G}}{\partial \zeta} = 0 \tag{24}$$

Discretizing Eq. (24) using first-order differences for the artificial time derivative and second-order time for the physical time derivative yields

$$\begin{aligned} -\Bigg[& \frac{(1+\theta_2) V_0^{n+1}\left(\boldsymbol{Q}_0^{n+1,m} - \boldsymbol{Q}_0^{n}\right) - \theta_2 V_0^{n-1} \Delta \boldsymbol{Q}_0^{n-1}}{\Delta t} \\ & + \boldsymbol{Q}_0^n RHS_{\mathrm{GCL}}^{n+1} + RHS_0^{n+1,m} \Bigg] \\ = & \left[\frac{\forall_0^{n+1} I}{\Delta \tau} + \frac{(1+\theta_2)\forall_0^{n+1} I}{\Delta t} + \frac{\partial RHS_0^{n+1,m}}{\partial \boldsymbol{Q}_0} \right] \Delta \boldsymbol{Q}_0^{n+1,m+1} \\ & + \left(\frac{\partial RHS_i^{n+1,m}}{\partial \boldsymbol{Q}_i} \right) \Delta \boldsymbol{Q}_i^{n+1,m+1} \end{aligned} \tag{25}$$

Here m is the pseudoiteration counter and n is the time-step counter. In terms of the general matrix form, $\boldsymbol{Ax} = \boldsymbol{b}$, the first bracketed term in Eq. (25) is the right-hand-side vector $\boldsymbol{b}$. The second bracketed term in Eq. (25) represents the diagonal terms of matrix $\boldsymbol{A}$, and the third bracketed term represents the off-diagonal terms of matrix $\boldsymbol{A}$. If the pseudotime step ($\Delta \tau$) is set to the physical time step (Δt), then the dual-time-stepping algorithm is equivalent to the Newton algorithm [Eq. (22)] plus an additional stabilizing term on the diagonal of the $\boldsymbol{A}$ matrix. As with the Newton method, the computational work required to complete one subiteration is equivalent to the work required for one time step without the dual-time-stepping scheme.

Boundary Conditions for Moving Grids

The grid velocities (time metrics for structured-grid flow solvers) must be included in the inviscid flux calculations for all transport equations included in a simulation (including turbulence models and species equations). The grid velocities must also be included for solid-wall boundary conditions (slip and no-slip walls) and in the calculation of the invariants for characteristic boundary conditions.

The grid velocity must also be taken into account when using ghost cells for implicit boundaries. The following subscripts are used in the derivation of the ghost-cell properties:

2—index of the first point off the wall,
1—index of the wall point, and
ghost—index of the ghost cell (mirrored from 2).

For a slip wall, the velocity at 2 is given by $u_{i,2}$. This can be decomposed into

$$u_{i,2} = u'_{i,2} + u_{i,g} \tag{26}$$

where the grid velocity $u_{i,g}$ is assumed to be constant for all the nodes for this derivation. The magnitude of the normal velocity can be defined as

$$u_{\mathrm{norm}} = u'_{i,2} \cdot n_i = u_{i,2} \cdot n_i - u_{i,g} \cdot n_i \tag{27}$$

where n_i are the components of the unit vector normal to the surface ($\boldsymbol{n}$). The velocity at the wall can be found by extrapolating the velocity from 2 and subtracting the normal component at the wall:

$$u_{i,1} = u'_{i,2} - u_{\mathrm{norm}} n_i + u_{i,g} = u_{i,2} - u_{\mathrm{norm}} n_i \tag{28}$$

The velocities at the ghost cell are given by

$$u_{i,\mathrm{ghost}} = u'_{i,2} - 2u_{\mathrm{norm}} n_i + u_{i,g} = u_{i,2} - 2u_{\mathrm{norm}} n_i \tag{29}$$

The delta quantities are given by

$$\Delta u_{i,21} = u_{i,2} - u_{i,1} = u_{\mathrm{norm}} n_i \tag{30}$$

$$\Delta u_{i,1\mathrm{ghost}} = u_{i,1} - u_{i,\mathrm{ghost}} = u_{\mathrm{norm}} n_{i1} = \Delta u_{i,21} \tag{31}$$

For a no-slip wall, the velocity at 2 is given by $u_{i,2}$. This can be decomposed into

$$u_{i,2} = u'_{i,2} + u_{i,g} \tag{32}$$

The velocity at the wall is equal to the grid velocity. The velocity at the ghost cell is defined as

$$u_{i,\mathrm{ghost}} = -u'_{i,2} + u_{i,g} = -u_{i,2} + 2u_{i,g} \tag{33}$$

The delta quantities required by some codes are given by

$$\Delta u_{i,21} = u_{i,2} - u_{i,1} = u_{i,2} - u_{i,g} \tag{34}$$

$$\Delta u_{i,1\mathrm{ghost}} = u_{i,1} - u_{i,\mathrm{ghost}} = u_{i,2} - u_{i,g} = \Delta u_{i,21} \tag{35}$$

The delta formulation is a convenient way to specify the ghost-cell values of velocity for moving-body problems.

Time-Accurate Validation

The following are some example validation cases for examining the time accuracy of a Navier–Stokes flow solver. The test cases presented here were run using the NXAIR[4] and OVERFLOW2[5] structured-grid overset Navier–Stokes flow solvers. The NXAIR flow solver uses a third-order HLLEM[6] upwind algorithm for the inviscid fluxes and second-order discretization in time. The implicit algorithm solves the nonfactored matrix system using a symmetric successive over relaxation (SSOR) algorithm that may be used in conjunction with a multigrid method. NXAIR uses a Newton subiteration strategy at each time step and includes both first- and second-order GCL in time and second-order GCL in space for deforming grids. OVERFLOW2 has a number of choices for the inviscid fluxes and the implicit algorithm. The choices for the inviscid flux calculation include second- and fourth-order central difference algorithms and several upwind algorithms including the Roe upwind algorithm. The alternating direction implicit (ADI) algorithm choices include a block tridiagonal solver and a diagonalized solver. A multigrid method can be used with either ADI solution algorithm. OVERFLOW2 can also use Newton subiterations or dual-time stepping at each time step. The solutions presented here for OVERFLOW2 used the default diagonalized implicit solver with the dual-time-stepping algorithm because this is the current recommended approach for unsteady applications. OVERFLOW2 does not presently include the formal GCL terms but does include a free-stream correction flux term that is similar to a first-order GCL. This effort is not intended to be a comparison of the two codes but a demonstration of the application of these test cases in the evaluation of the unsteady performance of the two production codes individually.

Deforming-Grid Free-Stream Test Case

A 41×41 Cartesian grid and a deformed grid of the same dimensions were constructed. The two grids are shown overlaid in Fig. 1. Solutions were computed using NXAIR with three Newton subiterations. The free-stream Mach number was set to 0.5 and the boundaries' values were held fixed for these calculations so that any deviation from free stream would be due to the grid deformation. The CFL number was set to 1000, and the computations were performed in a time-accurate manner. The solution was then calculated on a deforming grid that oscillates between the Cartesian grid and the deformed grid. An interpolation function given by Eq. (36) is used to determine the computational grid at each time step as is shown in Eq. (37) and (38):

$$f = \left[\sin\left(\frac{n\pi}{100}\right)\right]^2 \tag{36}$$

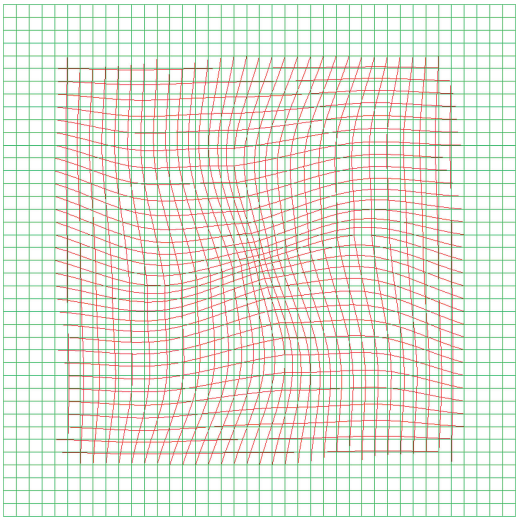

Fig. 1 Grids used in the free-stream deforming-grid study. The red grid is designated "grid 1" and the green is "grid 2."

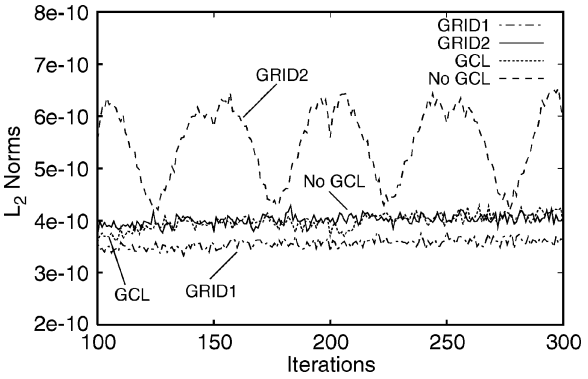

Fig. 2 Free-stream solutions with and without GCL.

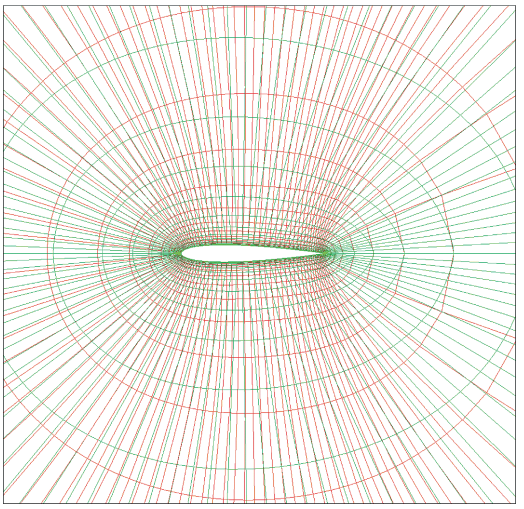

Fig. 3 NACA0012 grids used in the deforming-grid study. The red grid is designated "grid 1" and the green grid is "grid 2." Every other point is plotted for clarity.

$$x_{\text{comp}} = x_1 + f(x_2 - x_1) \tag{37}$$

$$y_{\text{comp}} = y_1 + f(y_2 - y_1) \tag{38}$$

where n is the iteration count for the flow-solver time step. This example represents a large amount of grid deformation per time step. The L2 norms for these computations are shown in Fig. 2. The grid 1 and grid 2 solutions were computed on the individual grids without any deformation. The GCL and no-GCL solutions were computed on the deforming grids.

The deforming-grid solution with the GCL is seen to produce the same level of L2 norm as the steady-state solutions on the Cartesian and deformed grids. The deforming-grid solution without the GCL produces an L2 norm that is about five times larger than the steady-state results. This indicates that the GCL is eliminating the error associated with the grid deformation.

Deforming-Grid NACA0012 Test Case

Two two-dimensional O-type Euler grids were constructed for a NACA0012 airfoil. Both grids have 221×41 points. The point distribution along the airfoil is the same for both grids. The two grids are shown overlaid in Fig. 3. The flow conditions chosen for this test case were a free-stream Mach number of 0.5 and an angle of attack of 2 deg.

The computations were performed using NXAIR with 1 (no subiterations), 3, 5, and 10 Newton subiterations. Unsteady calculations were performed as the grid was allowed to deform between grid 1 and grid 2 as described by Eqs. (36)–(38). The solutions were run time accurately with a CFL number of 2.7×10^4. The grid

deformations were begun after 1000 time steps on grid 1. The predicted force coefficients are shown in Fig. 4. The steady-state values for each grid are also shown in these figures.

The solutions become periodic as the grid is deformed for all combinations of Newton subiteration and GCL or no GCL. The forces oscillate between the steady-state values and converge with increasing Newton iterations when the GCL is used. The oscillating forces greatly exceed the steady-state force levels and do not seem to converge with increasing Newton iterations when the GCL is not used. It is obvious that failure to use the GCL in deforming-grid calculations can result in large errors in the calculation of forces. This example also indicates that the subiteration method alone is not sufficient to remove the errors introduced into the solution by the grid deformation.

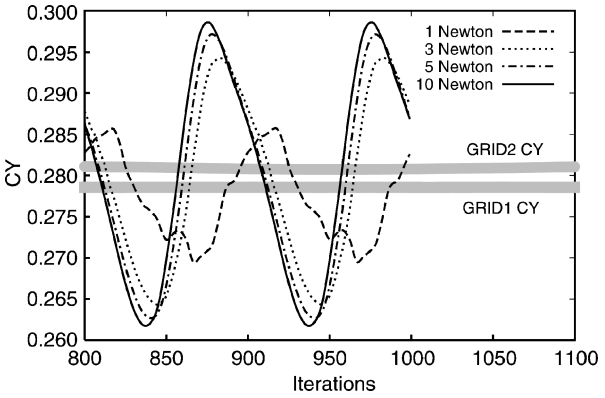

Fig. 4a Normal force coefficient on a deforming grid without GCL.

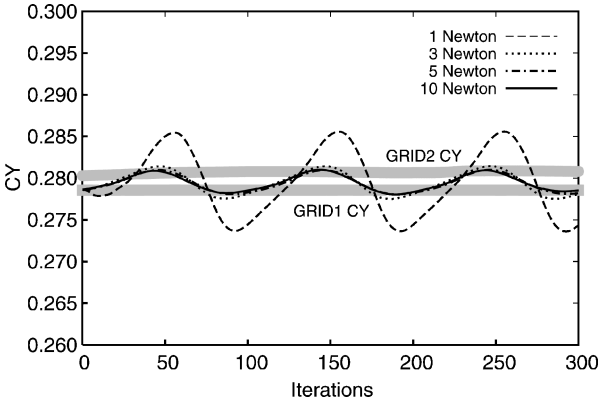

Fig. 4b Normal force coefficient on a deforming grid with GCL.

Inviscid Vortex Convection

A simple example of the value of subiterations and second-order time is the convection of an inviscid vortex. The ability to conserve the vortex shape and strength is important in many unsteady cases in which a shed vortex interacts with bodies well downstream of the vortex origin. This case can also be used to examine the level of numerical dissipation for a given flow solver. A vortex of strength $\Gamma = 5$ is centered on an 81×81 uniform grid in the x–z plane and allowed to convect downstream at $M = 0.5$ with a nondimensional time step $[dt\,(a_\infty / L_{Ref})]$ of 0.04. The grid spacing was set to 0.25 in both the x and z directions. The vortex is given by

$$\omega = [\Gamma(2 - \bar{R})/2\pi] \exp[0.5(1 - \bar{R})] \tag{39}$$

where $\bar{R}^2 = (x - x_0)^2 + (z - z_0)^2$ and x_0 and z_0 represent the location of the vortex center. The grid is given periodic boundary conditions in the flow direction. This allows the vortex to convect out of and back into the computational domain. The vortex should complete one cycle on the grid (i.e., return to its initial location) every 500 time steps. The vortex was allowed to convect for a nondimensional length of 100 (five cycles through the grid).

Figure 5 shows an example of a flow solver with relatively low numerical dissipation (OVERFLOW2 using third-order Roe spatial fluxes and second-order time with three Newton subiterations). The vorticity profiles and the vortex location are well preserved by this algorithm after five cycles through the grid. The vortex core is seen to be somewhat underresolved on this computational grid.

The minimum pressure in the vortex as a function of time is shown in Fig. 6 for several of the flux algorithms provided by OVERFLOW2. In theory this pressure should be preserved. The standard (STD) algorithm is the standard central difference algorithm with mixed second- and fourth-order smoothing where DIS4 is the fourth-order-smoothing coefficient. The results show the sensitivity of the solution to the numerical algorithm and smoothing level. The Yee algorithm and the STD algorithm with DIS4 = 0.04 are clearly inappropriate for this application.

Second-order time is seen to significantly lower the numerical dissipation as indicated by the better preservation of the physical quantities at the vortex core. The best results for this test case were obtained with the Roe third-order spatial algorithm using second-order time differencing and employing at least two Newton subiterations. Similar results to those shown with OVERFLOW2 were observed for the NXAIR flow solver using the default third-order spatial HLLEM algorithm.

Viscous Shock Tube

Shock-tube problems have long been used as test cases for unsteady flows. The position of the shock, expansion, and contact discontinuity in inviscid flow can be predicted theoretically. In this case a laminar viscous isothermal wall has been added so that numerical convergence in the boundary layer may also be addressed. The conditions chosen for this problem are left state: $\rho/\rho_{\text{ref}} = 1.0$,

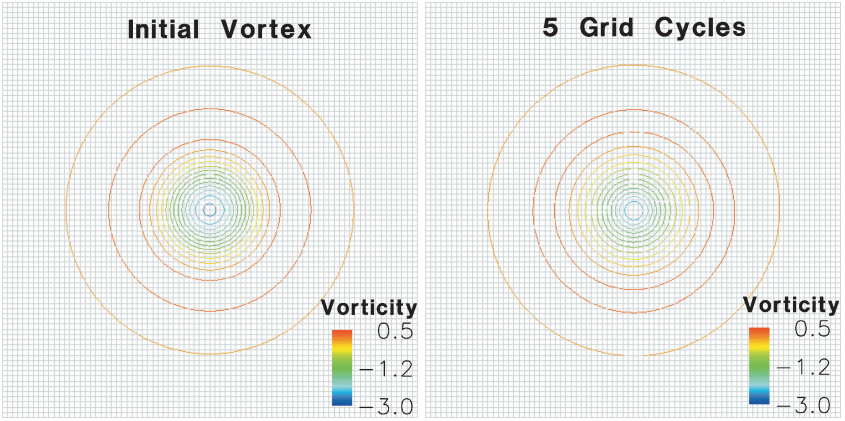

Fig. 5 Vorticity profiles for an inviscid convected vortex using OVERFLOW2.

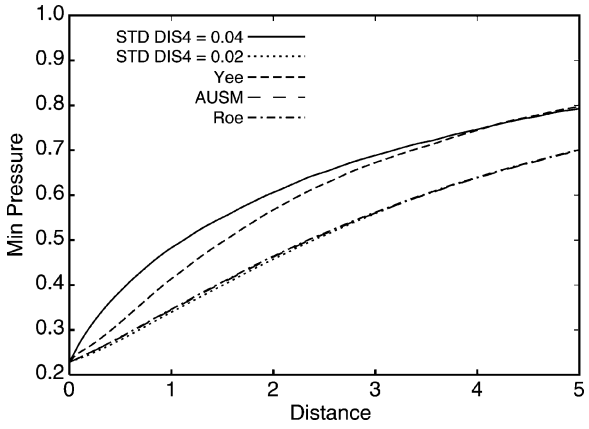

Fig. 6a Vortex minimum pressure for several algorithms within OVERFLOW2 using first-order time and three Newton subiterations.

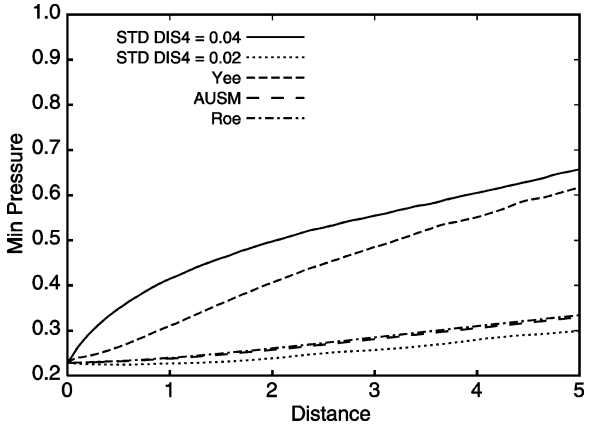

Fig. 6b Vortex minimum pressure for several algorithms within OVERFLOW2 using second-order time and three Newton subiterations.

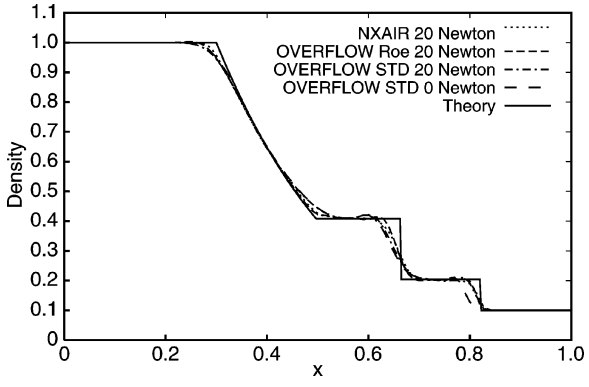

Fig. 7 Density on the centerline of the viscous shock tube at nondimensional time 0.2.

$p/p_{\text{ref}} = 1.0$, $T_{\text{wall}}/T_{\text{ref}} = 1.0$; right state: $\rho/\rho_{\text{ref}} = 0.1$, $p/p_{\text{ref}} = 0.1$, $T_{\text{wall}}/T_{\text{ref}} = 1.0$; Re (based on a_{inf}) $= 1.0 \times 10^5$; and nondimensional time step $= 0.00125$ and 0.000625. The results are evaluated at nondimensional times of 0.2 (160 iterations for a nondimensional time step of 0.00125 and 320 iterations for a nondimensional time step of 0.000625).

The calculations were performed on a 101×81 grid with a wall-spacing equivalent to a y^+ of 1. The streamwise spacing for the grid was 0.01. Results are presented for the NXAIR and OVERFLOW2 (Roe algorithm and second-order spatial central difference algorithm) codes. All calculations were performed with second-order time. The density on the tube centerline at nondimesional time 0.2 for the larger time step is shown in Fig. 7. The Newton subiteration

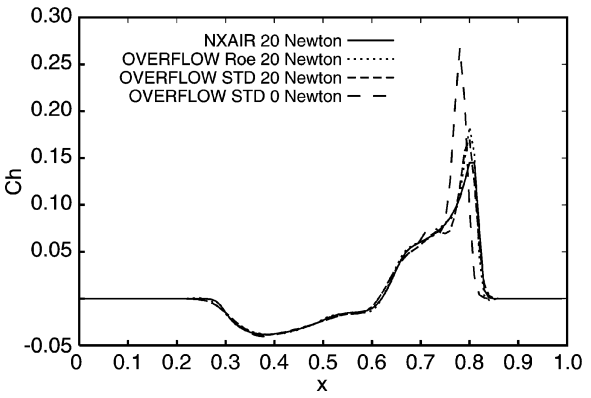

Fig. 8 Wall heat transfer for the viscous shock tube at nondimensional time 0.2.

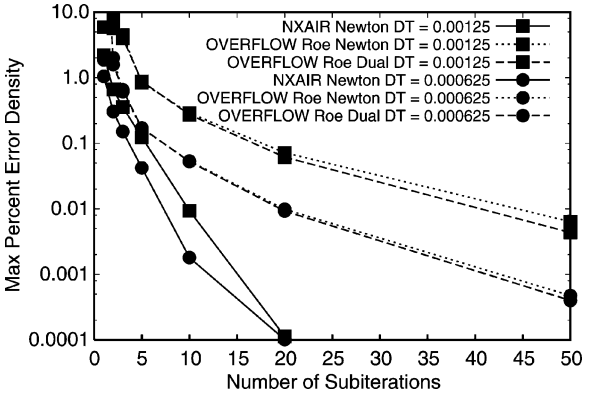

Fig. 9 Maximum error in centerline density for the shock tube.

is shown to significantly improve the results for the OVERFLOW2 STD (second-order spatial central difference) algorithm. The wall heat transfer at the same instant in time is shown in Fig. 8. The NXAIR and OVERFLOW2 solutions with 10 Newton subiterations are in good agreement. The OVERFLOW2 solution with no Newton subiterations differs significantly from the solutions with the Newton subiteration.

This test case offers the opportunity to compare solutions using the Newton subiteration algorithm and dual-time-stepping subiteration algorithm. The level of convergence is assessed by taking the difference between the solution with a given number of subiterations and a reference solution with enough subiterations to obtain local convergence. For NXAIR, the reference solution was obtained with 50 Newton subiterations. For OVERFLOW2 (Roe algorithm), the reference solution was obtained with 100 Newton subiterations. The dual-time-stepping solutions were obtained using local time stepping in the pseudotime step with a Courant-Friedrichs-Lewy (CFL) of one, and the maximum CFL was held to 5.

The maximum error for the various subiteration processes is shown as a function of the number of subiterations in Figs. 9 and 10. Increasing the number of subiterations drives the solutions to a local level of convergence for all the schemes tested. The NXAIR Newton scheme requires fewer subiterations to reach a level of local convergence than either of the subiteration schemes employed in OVERFLOW2. This may in part be due to the diagonalized algorithm used in OVERFLOW2. The error in the heat transfer coefficient seems to converge more slowly than does the error in density for the OVERFLOW2 algorithms, whereas they seem to converge at about the same rate for the NXAIR algorithm.

The computational cost for the algorithms is investigated in Figs. 11 and 12. An equivalent computational unit is defined as the time to complete one time step at the largest time step investigated. The computational unit (CU) is defined for each individual algorithm, and hence this is not an algorithm-to-algorithm comparison. Running with a larger time step and more Newton subiterations

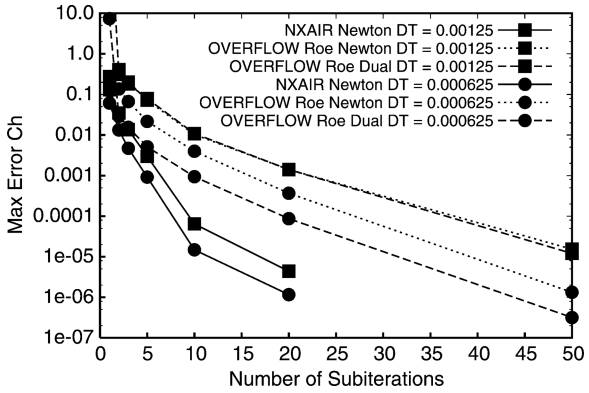

Fig. 10 Maximum error in wall heat transfer for the shock tube.

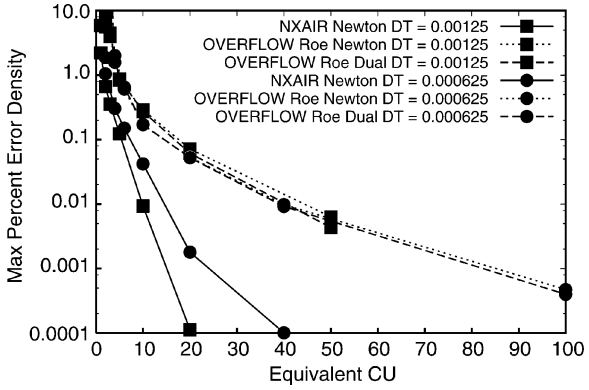

Fig. 11 Maximum error in centerline density vs computational units (CU) for the shock tube.

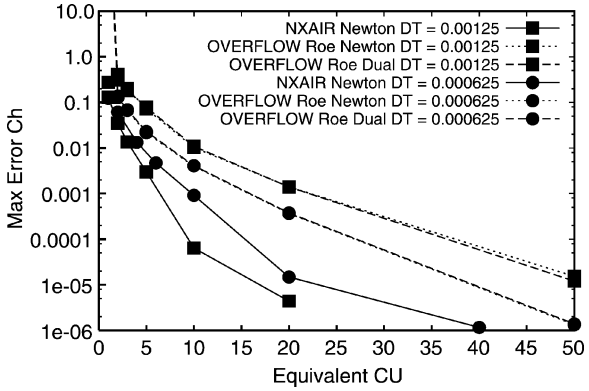

Fig. 12 Maximum error in wall heat transfer vs computational units (CU) for the shock tube.

drives NXAIR to the minimum error for density and heat transfer in the least amount of computational work. A constant product of the time step and number of subiterations produces the same density error level with OVERFLOW2. The heat transfer error is reduced more rapidly with the smaller time step with OVERFLOW2. This may indicate that the OVERFLOW2 algorithm is more sensitive to grid stretching or to the higher CFL, both of which are at a maximum near the walls of the shock tube.

Laminar Vortex Shedding from a Circular Cylinder

The vortex shedding from a circular cylinder is essentially two-dimensional for $Re_D < 180$ (Ref. 7). This example offers a simple geometry and a highly periodic flow to investigate the performance of a flow solver for a periodic unsteady viscous flow. The conditions chosen for the simulation here are a free-stream Mach number of 0.2, Reynolds number (Re_D) of 150, and a reference temperature of 500°R. Two time steps were selected for this study: $\Delta t = 9.12 \times 10^{-5}$ s (CFL = 92) and $\Delta t = 2.28 \times 10^{-5}$ s (CFL = 23).

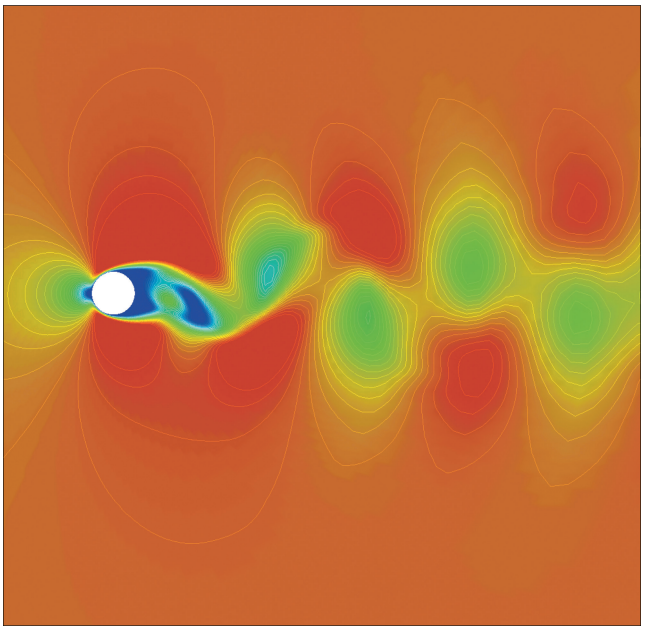

Fig. 13 Mach-number contours for a circular cylinder at $M = 0.2$ and $Re_D = 150$.

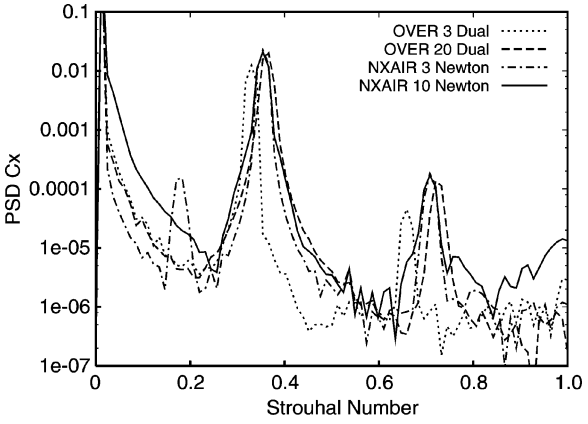

Fig. 14 Power spectral density of drag for the circular cylinder.

The time steps were chosen to allow about 280 and 1120 time steps per cycle of the primary vortex-shedding frequency as seen in the normal force. The simulations were performed on the 401 × 201 grid. Each simulation was run 10,000 iterations and the last 4096 were statistically analyzed. The unsteady motion was allowed to develop naturally for both codes (i.e., the flow was not artificially forced). The OVERFLOW2 results used the third-order Roe flux algorithm, and the NXAIR results were obtained with the third-order HLLEM flux algorithm.

Mach-number contours at one time instant are shown in Fig. 13. The vortex street is clearly evident. The dual-time-stepping results presented using OVERFLOW2 correspond to a local time step with a CFL of 1.0 and a minimum CFL of 5 in the inner iteration. Some spectral results for drag and lift are shown in Figs. 14 and 15. The spectra are seen to converge with increasing number of subiterations. The effect of increasing the number of subiterations and reducing the time step for the various subiteration strategies are shown in Figs. 16–21. Again, both codes demonstrate convergence for all the parameters examined with increasing number of subiterations. The predicted average-drag coefficient for both codes agrees well with experimental value[8] of 1.34. The lift Strouhal number agrees

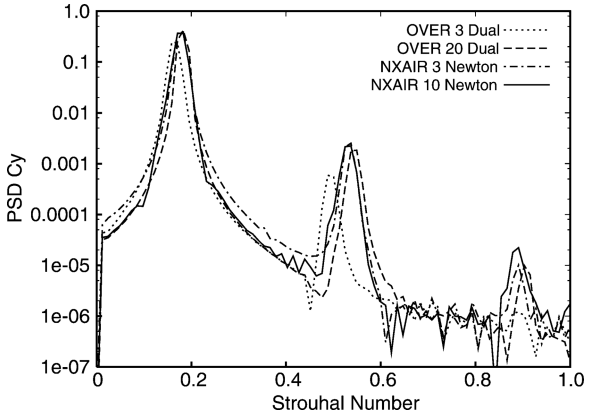

Fig. 15 Power spectral density of lift for the circular cylinder.

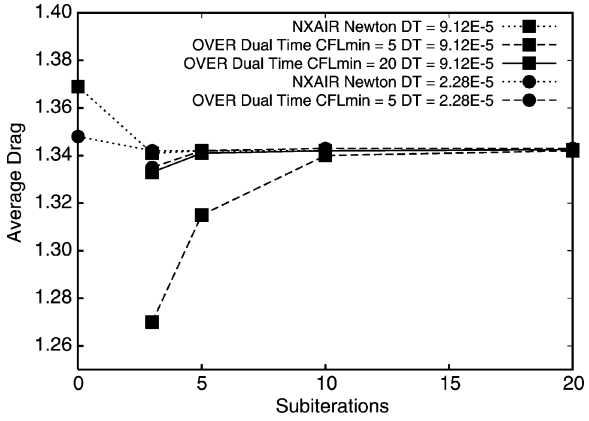

Fig. 16 Variation in the average drag with number of subiterations.

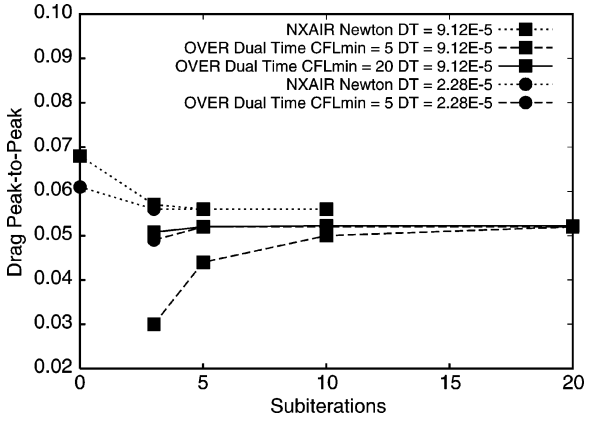

Fig. 17 Variation in the peak-to-peak drag with number of subiterations.

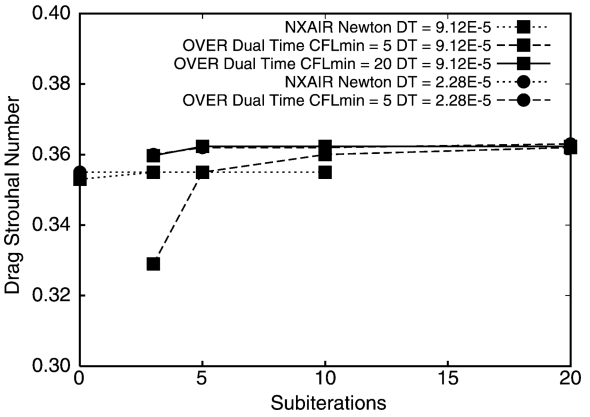

Fig. 18 Variation in the drag primary peak Strouhal number with number of subiterations.

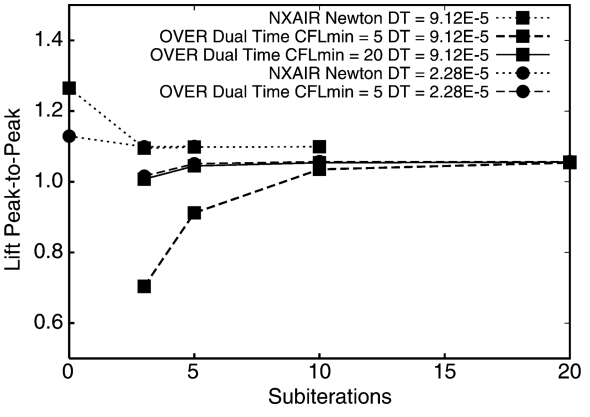

Fig. 19 Variation in the peak-to-peak lift with number of subiterations.

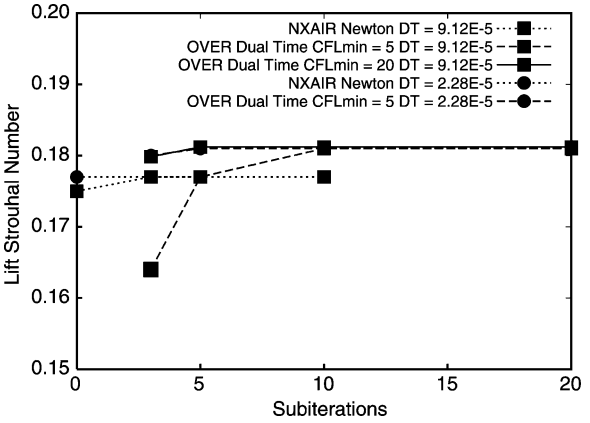

Fig. 20 Variation in the lift primary peak Strouhal number with number of subiterations.

well with correlations of available data[8–10] that range from 0.179 to 0.182.

The predicted Strouhal number converges with fewer subiterations than does the peak-to-peak value. This indicates that more subiterations are required to accurately predict the magnitude of the unsteady flow than to predict the frequency. A minimum of three Newton subiterations is required to achieve local convergence of the solution for NXAIR using a time step of 9.12×10^{-5} s. The dual-time-stepping algorithm in OVERFLOW2 requires about 20 subiterations to reach a similar level of convergence. Increasing the value of CFLMIN to 20 for OVERFLOW2 reduces the number of dual-time-step subiterations for convergence to three. This shows the importance of a rapidly converging algorithm in the inner iteration for dual-time stepping. Reducing the time step to 2.28×10^{-5} s improves the convergence and reduces the number of subiterations required.

The amount of computational work required to obtain a level of error for the lift peak to peak is shown in Fig. 22. An equivalent computational unit is defined as the time to complete one time step at the largest time step investigated. The CU is defined for each individual algorithm, and hence this is not an algorithm-to-algorithm comparison. Running with a larger time step and more Newton subiterations is seen to drive all the algorithms to the minimum error in the least amount of time.

Oscillating Airfoil

The low-angle-of-attack NACA0015 oscillating-airfoil case of Ref. 11 is a simple two-dimensional moving-body problem for

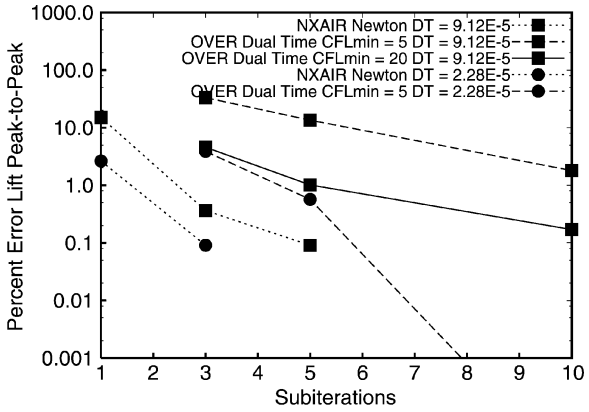

Fig. 21 Percent change in peak-to-peak lift with time step and subiterations.

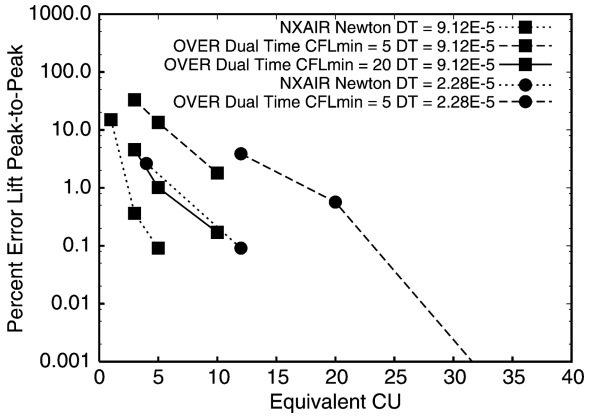

Fig. 22 Percent change in peak-to-peak lift vs equivalent computational units (CU).

turbulent flows. The flow remains fully attached to the airfoil for most of the cycle, allowing two-dimensional simulations to adequately resolve the physics of the problem. Simulations at higher angle of attack would require three-dimensional simulations to capture the large-scale flow separation. The case chosen for this simulation used the angle-of-attack variation given by

$$\alpha = 4\,\text{deg} + 4.2\,\text{deg}\sin(2\pi f t) \tag{40}$$

where the frequency of oscillation was specified as 10 cycles per second. The conditions of the simulation were a Mach number of 0.29 and a chord Reynolds number of 1.95×10^6. The airfoil is rotated about the quarter-chord point.

A 441×71 C grid was used to discretize the NACA0015 airfoil. The grid included 241 points along the airfoil and was packed at both the leading and trailing edges. The wall spacing was set for $y^+ = 1$ along the airfoil. The Spalart–Allmaras[12] one-equation turbulence model was used in this simulation for both NXAIR and OVERFLOW2. Both codes solve the turbulent transport equations loosely coupled with the mean-flow equations within the subiteration loop. The OVERFLOW2 results used the fourth-order central difference flux algorithm. Similar results were obtained with the default second-order central difference flux algorithm and the Roe flux algorithm in OVERFLOW2. Two time steps corresponding to 512 (1/5120 s) and 1024 (1/10240 s) steps per pitch cycle were run. These time steps were chosen to guarantee that the solution would contain points at the maximum and minimum angles of attack (−0.2 and 8.2 deg), and also at the mid–angle of attack (4 deg) on both the down and up stroke.

The NXAIR calculations were initialized to free-stream conditions and the pitching motion was included from the first time step. The second pitch cycle was used in the comparisons shown here to remove the initial transients from the free-stream conditions. To overcome numerical-stability problems, a steady-state solution was initially obtained for each of the OVERFLOW2 examples. This solution was used to initialize the moving-body calculations. The second pitch cycle was used in the comparisons shown here.

The flow is attached to the airfoil over the entire angle-of-attack range. Force and moment coefficients as a function of airfoil angle of attack are shown in Figs. 23 and 24 for NXAIR and OVERFLOW2 respectively. The two-dimensional simulation for both flow solvers is in good agreement with the data and with previous CFD

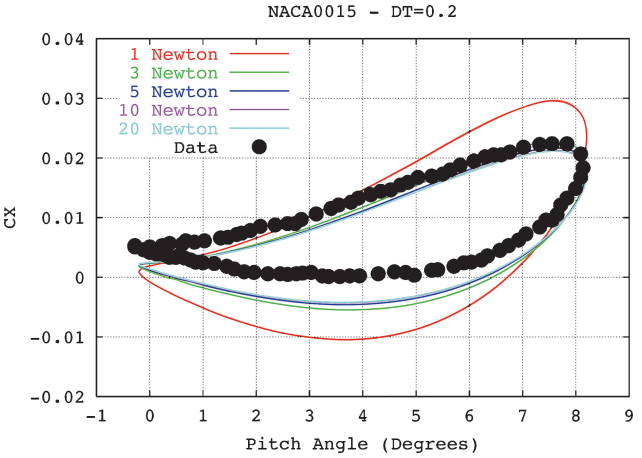

Fig. 23a NXAIR axial-force coefficient vs pitch angle for a pitching NACA0015 airfoil for $\Delta t = 1/5120$ s.

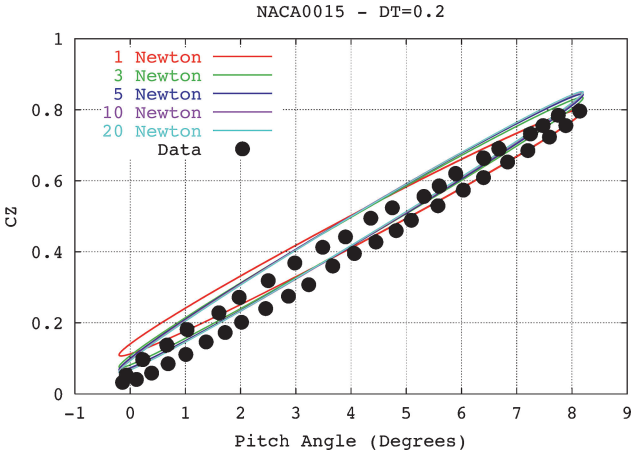

Fig. 23b NXAIR normal-force coefficient vs pitch angle for a pitching NACA0015 airfoil for $\Delta t = 1/5120$ s.

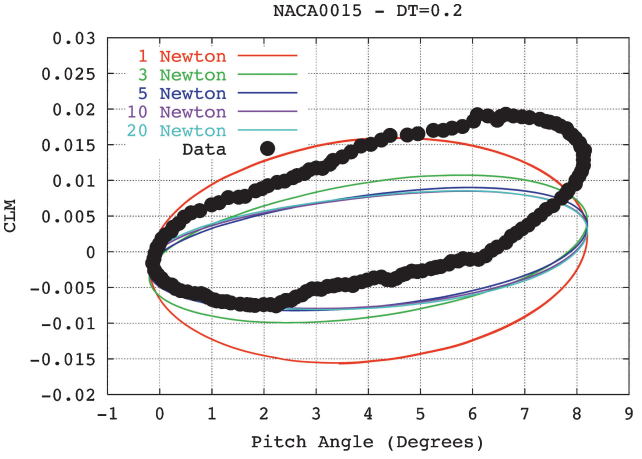

Fig. 23c NXAIR pitching-moment coefficient vs pitch angle for a pitching NACA0015 airfoil $\Delta t = 1/5120$ s.

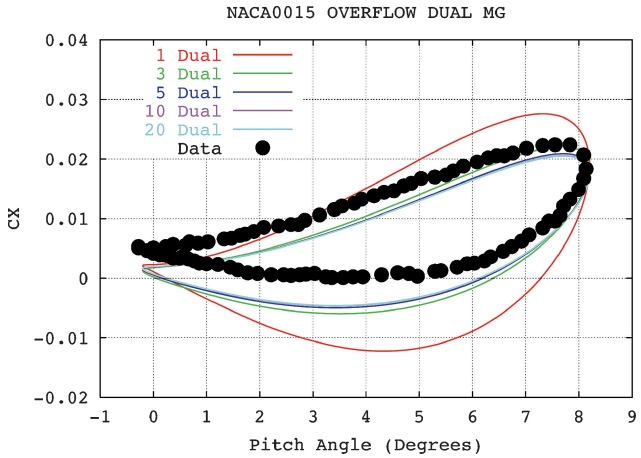

Fig. 24a OVERFLOW2 axial-force coefficient vs pitch angle for a pitching NACA0015 airfoil for $\Delta t = 1/5120$ s using dual-time-stepping and multigrid methods.

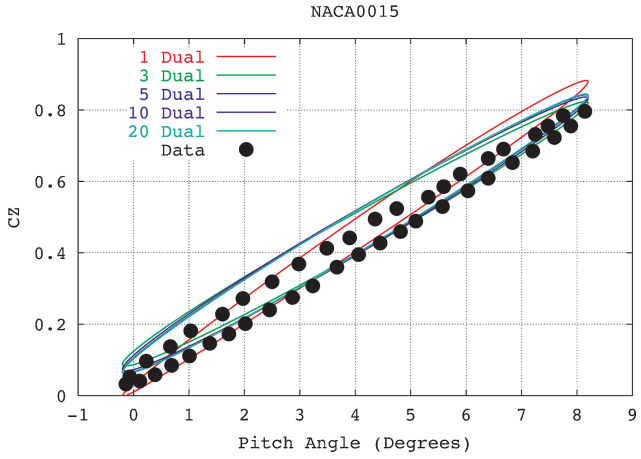

Fig. 24b OVERFLOW2 normal-force coefficient vs pitch angle for a pitching NACA0015 airfoil for $\Delta t = 1/5120$ s using dual-time-stepping and multigrid methods.

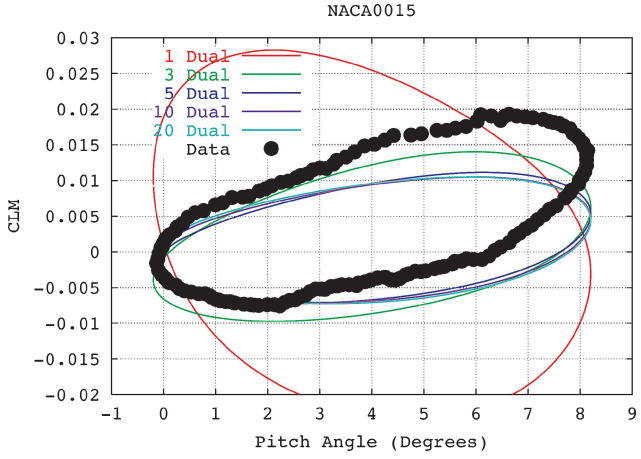

Fig. 24c OVERFLOW2 pitching-moment coefficient vs pitch angle for a pitching NACA0015 airfoil for $\Delta t = 1/5120$ s using dual-time-stepping and multigrid methods.

predictions for this case.[13,14] The differences between the CFD and the data can be attributed to boundary-layer-transition location and to three-dimensional effects, both of which are ignored in the computations.

The effectiveness of increasing the number of subiterations for the Newton and dual-time-stepping algorithms at two time-step increments are shown in Figs. 25–27. These figures show the normal force at the maximum and minimum angles of the oscillation and

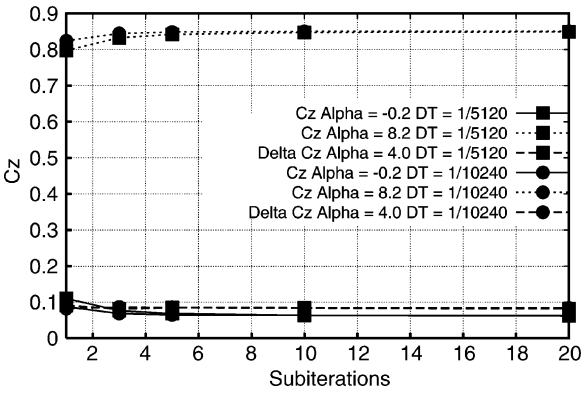

Fig. 25 NXAIR normal force for varying time step and number of subiterations.

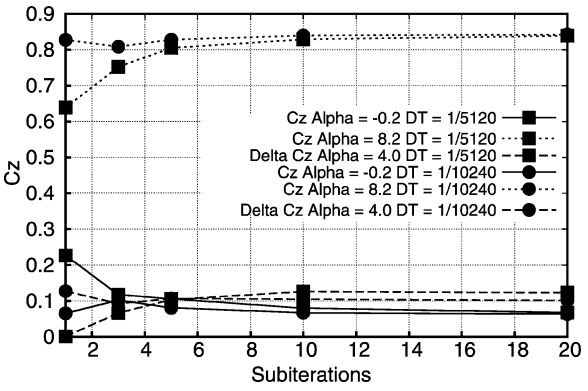

Fig. 26 OVERFLOW2 normal force for varying time step and number of subiterations using dual-time stepping.

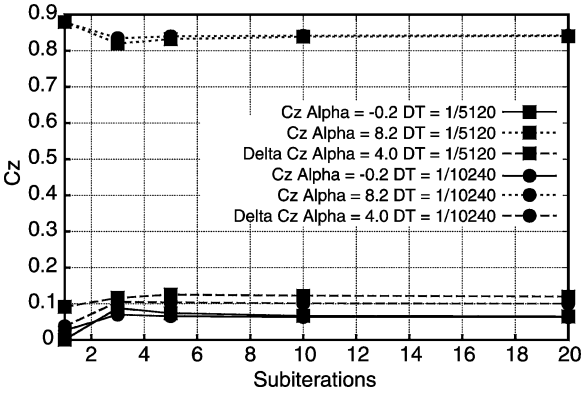

Fig. 27 OVERFLOW2 normal force for varying time step and number of subiterations using dual-time-stepping and multigrid methods.

the delta normal force (hysterisis in normal force) at the midpoint of the oscillation. The error presented in Fig. 28 is calculated relative to the solution with the most subiterations at a given time step for each code and algorithm.

The solutions are seen to locally converge with increasing number of subiterations. The NXAIR results also converge to the same value when the time step is reduced. The OVERFLOW2 results converge to a different value of normal force increment (delta Cz) with time-step variation. The cause of this anomaly is still under investigation. The multigrid algorithm is seen to provide improved convergence for the OVERFLOW2 results. This again indicates the importance of a rapidly converging inner algorithm when using dual-time stepping.

The amount of computational work required to obtain a level of error for the lift peak to peak is shown in Fig. 29. An equivalent computational unit is defined as the time to complete one time step at the largest time step investigated. The CU is defined for each individual

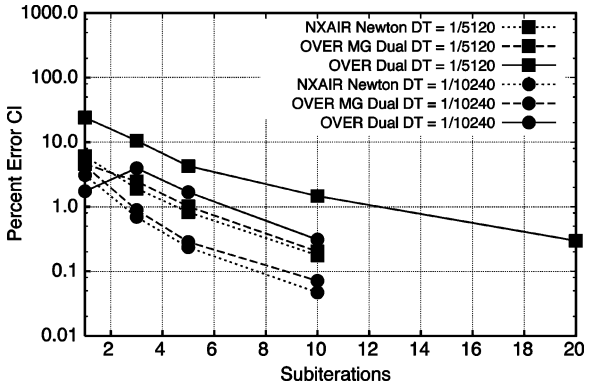

Fig. 28 Percent error in normal force at $\alpha = 8.2$ deg.

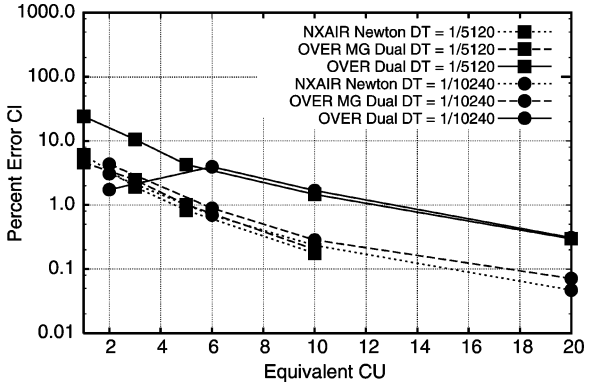

Fig. 29 Percent error in normal force vs equivalent computational units (CU) at $\alpha = 8.2$ deg.

algorithm, and hence this is not an algorithm-to-algorithm comparison. Running with a larger time step and more subiterations provides some advantage in driving all the algorithms to the minimum error in the least amount of time.

Conclusion

Time-accurate Navier–Stokes flow solvers utilizing Newton and dual-time-step implicit subiteration algorithms have been investigated for both moving-body and deforming-grid applications. A set of relatively simple two-dimensional validation cases has been identified to assess the performance of these unsteady CFD solvers with varying time-step size and number of subiterations. A set of two-dimensional validation cases has been identified to assess the performance of unsteady CFD solvers with varying time-step size and number of subiterations. These cases demonstrate the advantages of second-order time derivatives and subiterations for unsteady-flow simulations. This investigation also indicates subiteration algorithms and large time steps can be used to reduce the cost of a given unsteady-flow simulation.

Based on these investigations, it is recommended that an implicit flow solver for unsteady flows include the following capabilities:

1) second-order time derivatives,
2) numerical flux algorithms with low dissipation,
3) subiteration scheme to improve local convergence,
4) a rapidly converging inner algorithm, and
5) GCL terms when deforming grids are present.

Acknowledgments

This publication was made possible through support provided by the U.S. Department of Defense High Performance Computing Modernization Program (HPCMP) Programming Environment and Training (PET) activities through Mississippi State University under terms of Contract No. N62306-01-D-7110. The authors thank Pieter Buning (NASA LaRC) for his assistance with the OVERFLOW2 cases.

References

[1]Thomas, P. D., and Lombard, C. K., "Geometric Conservation Law and Its Application to Flow Computations on Moving Grids," *AIAA Journal*, Vol. 17, No. 10, 1978, pp. 1030–1037.

[2]Hyams, D., "An Investigation of Parallel Implicit Solution Algorithms for Incompressible Flows on Unstructured Topologies," Ph.D. Dissertation, Mechanical Engineering Dept., Mississippi State Univ., May 2000.

[3]Pandya, S. A., Venkateswaran, S., and Pulliam, T. H., "Implementation of Preconditioned Dual-Time Procedures in OVERFLOW," AIAA Paper 2003-0072, Jan. 2003.

[4]Tramel, R. W., and Nichols, R. H., "A Highly Efficient Numerical Method for Overset-Mesh Moving-Body Problems," AIAA Paper 97-2040, June 1997.

[5]Murphy, K. J., Buning, P. G., Pamadi, B. N., Scallion, W. I., and Jones, K. M., "Status of Stage Separation Tool Development for Next Generation Launch Technologies," AIAA Paper 2004-2595, June 2004.

[6]Einfeldt, B., Munz, C. D., Roe, P. L., and Sjogreen, B., "On Godunov-Type Methods near Low Densities," *Journal of Computational Physics*, Vol. 92, No. 2, 1991, pp. 273–295.

[7]Ma, X., and Karniadakas, G. E., "A Low-Dimensional Model for Simulating Three-Dimensional Cylinder Flow," *Journal of Fluid Mechanics*, Vol. 458, 2002, pp. 181–190.

[8]Tritton, D. J., "Experiments on the Flow past a Circular Cylinder at Low Reynolds Numbers," *Journal of Fluid Mechanics*, Vol. 6, 1959, pp. 547–567.

[9]Roshko, A., "On the Development of Turbulent Wakes from Vortex Streets," NACA Report 1191, U.S. GPO, Washington, DC, 1954.

[10]Friehe, C. A., "Vortex Shedding Form Cylinders at Low Reynolds Numbers," *Journal of Fluid Mechanics*, Vol. 100, Part 2, 1980, pp. 237–241.

[11]Piziali, R. A., "2-D and 3-D Oscillating Wing Aerodynamics for a Range of Angles of Attack Including Stall," NASA TM 4632, Sept. 1994.

[12]Spalart, P. R., and Allmaras, S. R., "A One-Equation Turbulence Model for Aerodynamic Flows," AIAA Paper 92-0439, Jan. 1992.

[13]Ko, S., and McCroskey, W. J., "Computations of Unsteady Separating Flows over an Oscillating Airfoil," *AIAA Journal*, Vol. 35, No. 7, 1997, pp. 1235–1238.

[14]Barakos, G., and Drikakis, D., "An Implicit Unfactored Method for Unsteady Turbulent Compressible Flows and Moving Boundaries," *Computers and Fluids*, Vol. 28, 1999, pp. 899–922.