Recent Enhancements To The FUN3D Flow Solver For Moving-Mesh Applications

Robert T. Biedron*

James L. Thomas †

NASA Langley Research Center, Hampton, VA 23681

Abstract

An unsteady Reynolds-averaged Navier-Stokes solver for unstructured grids has been extended to handle general mesh movement involving rigid, deforming, and overset meshes. Mesh deformation is achieved through analogy to elastic media by solving the linear elasticity equations. A general method for specifying the motion of moving bodies within the mesh has been implemented that allows for inherited motion through parent-child relationships, enabling simulations involving multiple moving bodies. Several example calculations are shown to illustrate the range of potential applications. For problems in which an isolated body is rotating with a fixed rate, a noninertial reference-frame formulation is available. An example calculation for a tilt-wing rotor is used to demonstrate that the time-dependent moving grid and noninertial formulations produce the same results in the limit of zero time-step size.

Nomenclature

$[\Phi]$	state-transition matrix
$[\Theta]$	convolution integral of $[\Phi]$
α	angle of attack
$\bar{\bar{\tau}}$	stress tensor
$\bar{\bar{\mathbf{F}}}$	flux tensor
$\bar{\bar{\mathbf{I}}}$	identity tensor
δ	structural displacement
$\hat{\mathbf{F}}_a$	generalized aerodynamic force
$\hat{\mathbf{n}}$	unit (normal) vector
$\mathbf{\Omega}$	rotation vector
$\mathbf{Q}$	control-volume average of $\mathbf{q}$
$\mathbf{q}$	vector of conserved variables or generalized coordinates
$\mathbf{R}$	radial vector
$\mathbf{u}$	mesh point displacement
$\mathbf{W}$	control volume face velocity
$\mathbf{x}_i = (\mathbf{q}, \dot{\mathbf{q}})^T$	vector of generalized displacements and velocities, ith mode
ν	Poisson's ratio
ω_i	natural frequency of ith mode
$\phi_i(\mathbf{x})$	ith mode shape
ϕ_n	coefficients for backward time differencing
E	Young's modulus
R	residual
t	nondimensional time
V	control volume
$[\mathbf{T_m}]$	4x4 transform matrix
CG	center of gravity
GCL	Geometric Conservation Law

Introduction

The simulation of flows involving moving geometries is becoming increasingly practical as computer performance increases. The range of applications in which moving geometries are essential is quite broad; a very short list includes store separation,[1] rotorcraft,[2] flutter analysis,[3] and maneuvering aircraft.[4] Depending on the particular application, different types of mesh-motion schemes within the CFD solver may be needed. For example, analysis of a rigid aircraft in spin is most efficiently done with a mesh in which all points move as a rigid body along with the aircraft. For flutter analysis however, the geometry must be allowed to deform, and this in turn requires that the surrounding mesh deform to accommodate the changes in the surface. Store separation analysis usually involves large relative motion; without remeshing, mesh

*Senior Research Scientist, Member AIAA

†Senior Research Scientist, Fellow AIAA

deformation schemes typically fail (resulting in collapsed cells) when the motion is too large. Remeshing can eliminate the collapsed cells, but this adds extra computational cost, and remeshing is not easily made compatible with the backward-time integration schemes used in many flow solvers. Overset meshes pose no difficulties for backward-time integration and enable large relative motion between components. Rotorcraft simulations present a particularly challenging problem, because the rotor blades, in addition to rotational motion and pitch changes, are inherently aeroelastic. In addition, there is large relative motion between the blades and the fuselage. For these simulations, deforming overset meshes are used.

This paper describes recent work to extend the mesh motion capabilities in the FUN3D unstructured-mesh, Navier-Stokes flow solver to allow a broad range of moving-geometry applications. Previously, FUN3D only addressed rigid mesh motion of the entire domain,[5] which limited the range of applications. The general formulation for temporal integration on moving and deforming meshes is presented in detail, as is the linear elasticity analogy used to deform meshes when required. The transform matrix approach used for moving either a rigid grid or a rigid surface immersed in a deforming grid is described. For a specialized class of applications, a noninertial reference frame in which the body is stationary leads to a more efficient solution than the general mesh motion approach. The similarity of this particular noninertial formulation to the moving-mesh formulation is discussed. Finally, the mode-shape based, linear-structures model implemented for aeroelastic problems is briefly described. Several example calculations are shown to illustrate some of the applications now possible with the FUN3D solver, including dynamic pitching of a blended-wing aircraft, flutter of a thin wing, and store separation with overset grids. An isolated rotor in hover is used to illustrate the noninertial formulation. The same example is used to verify that the noninertial frame and the inertial-frame, moving mesh formulations produce the same result as the time step tends to zero in the inertial frame.

Governing Equations

The unsteady Navier-Stokes equations, representing the conservation laws as seen by an observer in an inertial reference frame, may be written in integral form for either a moving or stationary control volume as

$$\frac{\partial}{\partial t}\int_V \mathbf{q}\,dV + \oint_{\partial V} (\bar{\bar{\mathbf{F}}}^* - \bar{\bar{\mathbf{F}}}_\mathbf{v})\cdot\hat{\mathbf{n}}\,dS = 0$$

where V is the control volume, bounded by control surface ∂V, with local control volume face velocity $\mathbf{W}$. The vector $\mathbf{q}$ represents the conserved variables; the tensors $\bar{\bar{\mathbf{F}}}^*$ and $\bar{\bar{\mathbf{F}}}_\mathbf{v}$ represent the convective and diffusive fluxes of the conserved variables, respectively. For the case of a moving control volume, the convective fluxes must account for the relative reduction or enhancement of the flux through the control surface owing to the local control surface speed. The flux through the moving control volume is therefore $\bar{\bar{\mathbf{F}}}^* = \bar{\bar{\mathbf{F}}} - \mathbf{q}\mathbf{W}^\mathbf{T}$, where $\bar{\bar{\mathbf{F}}}$ is the usual flux tensor associated with a stationary control volume. This is often referred to as the Arbitrary Lagrangian-Eulerian (ALE)[6] formulation; the special cases of $\mathbf{W} = 0$ and $\mathbf{W} = (u, v, w)^T$ (fluid velocity) lead to the classical Eulerian and Lagrangian descriptions, respectively.

Introducing a volume average of $\mathbf{q}$

$$\mathbf{Q} = \frac{\int_V \mathbf{q}\,dV}{V}$$

the conservation equations are

$$\frac{\partial(\mathbf{Q}V)}{\partial t} + \oint_{\partial V} (\bar{\bar{\mathbf{F}}}^* - \bar{\bar{\mathbf{F}}}_\mathbf{v})\cdot\hat{\mathbf{n}}\,dS = 0 \tag{1}$$

where for compressible flow the conserved variables, inviscid flux vectors, and viscous flux vectors are, respectively,

$$\mathbf{q} = \begin{bmatrix} \rho \\ \rho u \\ \rho v \\ \rho w \\ E \end{bmatrix}$$

$$\bar{\bar{\mathbf{F}}}^* = \begin{bmatrix} \rho(u-W_x) \\ \rho u(u-W_x)+p \\ \rho v(u-W_x) \\ \rho w(u-W_x) \\ (E+p)(u-W_x)+W_x p \end{bmatrix}\hat{\mathbf{i}} + \begin{bmatrix} \rho(v-W_y) \\ \rho u(v-W_y) \\ \rho v(v-W_y)+p \\ \rho w(v-W_y) \\ (E+p)(v-W_y)+W_y p \end{bmatrix}\hat{\mathbf{j}} + \begin{bmatrix} \rho(w-W_z) \\ \rho u(w-W_z) \\ \rho v(w-W_z) \\ \rho w(w-W_z)+p \\ (E+p)(w-W_z)+W_z p \end{bmatrix}\hat{\mathbf{k}}$$

$$\bar{\bar{\mathbf{F}}}_{\mathbf{v}} = \begin{bmatrix} 0 \\ \tau_{xx} \\ \tau_{yx} \\ \tau_{zx} \\ u\tau_{xx} + v\tau_{xy} + w\tau_{xz} - kT_x \end{bmatrix} \hat{\mathbf{i}} + \begin{bmatrix} 0 \\ \tau_{xy} \\ \tau_{yy} \\ \tau_{zy} \\ u\tau_{yx} + v\tau_{yy} + w\tau_{yz} - kT_y \end{bmatrix} \hat{\mathbf{j}} + \begin{bmatrix} 0 \\ \tau_{xz} \\ \tau_{yz} \\ \tau_{zz} \\ u\tau_{zx} + v\tau_{zy} + w\tau_{zz} - kT_z \end{bmatrix} \hat{\mathbf{k}}$$

In the above equations, the velocity field given by $\mathbf{u} = (u, v, w)^T$ is measured relative to the inertial frame. The mean flow equations are closed by the equation of state for a perfect gas

$$p = (\gamma - 1)(E - \rho\frac{(u^2 + v^2 + w^2)}{2})$$

The detailed components of the stress tensor, heat flux vector, and auxiliary closure equations are given in Reference 5 and are not repeated here. The effects of turbulence are incorporated via a turbulent eddy viscosity determined from a suitable turbulence model.

It is worth noting at this point that for the special case of a spatially and temporally constant state vector, e.g. $\mathbf{q} = (1, 0, 0, 0, 0)^T$, the governing equations reduce to the "Geometric Conservation Law" (GCL)[7]

$$\frac{\partial V}{\partial t} = \oint_{\partial V} \mathbf{W}^T \cdot \hat{\mathbf{n}}\, dS \tag{2}$$

Solution Algorithm

The unstructured mesh solver used for this study is FUN3D.[8] Within the code, Equation 1 is discretized over the median dual volume surrounding each mesh point, balancing the time rate of change of the averaged conserved variables in each dual volume with the flux of mass, momentum, and energy through the instantaneous surface of the control volume.

Spatial Discretization

Details of the spatial discretization have been presented in Reference 8, and will only be summarized here. Within FUN3D, the convective fluxes are computed with a flux-splitting scheme, and for second order accuracy the values at dual-cell interfaces are reconstructed using gradients at mesh nodes computed with a least-squares technique. Limiting of the reconstructed values may be employed for flows with strong shocks. For all results presented in this paper, the convective flux scheme used is Roe's flux difference splitting[9] with second order unlimited reconstruction. For tetrahedral meshes, the full viscous fluxes are discretized using a finite-volume formulation in which the required velocity gradients on the dual faces are computed using the Green-Gauss theorem. On tetrahedral meshes this is equivalent to a Galerkin type approximation. For non-tetrahedral meshes, the same Green-Gauss approach can lead to odd-even decoupling. A pure edge-based approach can be used to circumvent the odd-even decoupling issue, but yields only approximate viscous terms. Thus for non-tetrahedral meshes, the edge-based gradients are combined with Green-Gauss gradients, which improves the h-ellipticity of the operator, and allows the complete viscous stresses to be evaluated. For turbulent flows, both the one-equation model of Spalart and Allmaras[10] (SA) and the two-equation SST model of Menter[11] are available. The SA model may be solved loosely coupled to the mean-flow equations or tightly coupled to the mean-flow equations. For all results presented in this paper, the one equation SA model is employed, solved in a loosely coupled fashion.

Few modifications are needed to the spatial discretization summarized above to handle moving meshes. Specifically, the dual face speeds need to be included in the convective flux routines for both the mean flow and turbulence equations (including boundary routines); the diffusive flux routines remain unaltered. For viscous flows, the no-slip condition at the wall is satisfied by setting the fluid velocity equal to the velocity of the wall itself.

Temporal Discretization

The time-advancement scheme in FUN3D is described in some detail below. The basic steps follow the derivation given by Reference 5 for the rigid-mesh case, with extensions to allow for the more general deforming mesh case.

Equation 1 may be written as

$$\frac{\partial(\mathbf{Q}V)}{\partial t} = \mathbf{R}$$

Evaluating this equation at time level n+1, and writing the time derivative as a series expansion of successive levels backward in time gives

$$\frac{1}{\Delta t}[\phi_{n+1}(\mathbf{Q}V)^{n+1} + \phi_n(\mathbf{Q}V)^n + \phi_{n-1}(\mathbf{Q}V)^{n-1} + \phi_{n-2}(\mathbf{Q}V)^{n-2} + \ldots] = \mathbf{R}(\mathbf{Q}^{n+1}) \equiv \mathbf{R}^{n+1} \tag{3}$$

Similarly, the Geometric Conservation Law may be discretized as

$$\frac{1}{\Delta t}[\phi_{n+1}V^{n+1} + \phi_n V^n + \phi_{n-1}V^{n-1} + \phi_{n-2}V^{n-2} + \ldots] = R_{GCL}^{n+1} \tag{4}$$

The sequence $\{\ \phi_n\ \}$ defines a family of backward difference formulae (BDF). The particular choice of $\{\ \phi_n\ \}$ governs the accuracy of the temporal discretization; several choices are described in Reference 5. For consistency, the sequence must satisfy the requirement $\Sigma\phi_n = 0$, and the discretization of the time derivative term in the GCL must use the same sequence as the conservation equations. Equation 3 may be written as

$$\begin{aligned}\frac{1}{\Delta t}[\mathbf{Q}^n(\phi_{n+1}V^{n+1} + \phi_n V^n + \phi_{n-1}V^{n-1} + \phi_{n-2}V^{n-2} + \ldots)\\ +\phi_{n+1}(\mathbf{Q}^{n+1} - \mathbf{Q}^n)V^{n+1} + \phi_{n-1}(\mathbf{Q}^{n-1} - \mathbf{Q}^n)V^{n-1} + \phi_{n-2}(\mathbf{Q}^{n-2} - \mathbf{Q}^n)V^{n-2} + \ldots] = \mathbf{R}^{n+1}\end{aligned} \tag{5}$$

Following Morton et al.,[12] the discrete GCL (Equation 4), can be used to rewrite Equation 5 as

$$\mathbf{Q}^n R_{GCL}^{n+1} + \frac{1}{\Delta t}[\phi_{n+1}(\mathbf{Q}^{n+1} - \mathbf{Q}^n)V^{n+1} + \phi_{n-1}(\mathbf{Q}^{n-1} - \mathbf{Q}^n)V^{n-1} + \phi_{n-2}(\mathbf{Q}^{n-2} - \mathbf{Q}^n)V^{n-2} + \ldots] = \mathbf{R}^{n+1} \tag{6}$$

At this stage the right-hand side of Equation 6 could be linearized about time level n, resulting in an implicit scheme for $\Delta\mathbf{Q}^n = \mathbf{Q}^{n+1} - \mathbf{Q}^n$. However, the linearization introduces an additional error into the result that is time-step dependent. Furthermore, the nonlinear residual $\mathbf{R}(\mathbf{Q})$ is difficult to linearize exactly, so that in practice approximate linearizations are often used, introducing another level of error.

To mitigate these linearization errors, as in Reference 13, a pseudo-time term is added to Equation 6

$$\begin{aligned}(V\frac{\partial \mathbf{Q}}{\partial \tau})^{n+1} + \mathbf{Q}^n R_{GCL}^{n+1} + \frac{1}{\Delta t}[\phi_{n+1}(\mathbf{Q}^{n+1} - \mathbf{Q}^n)V^{n+1} + \phi_{n-1}(\mathbf{Q}^{n-1} - \mathbf{Q}^n)V^{n-1}\\ +\phi_{n-2}(\mathbf{Q}^{n-2} - \mathbf{Q}^n)V^{n-2} + \ldots] = \mathbf{R}^{n+1}\end{aligned}$$

where the pseudo time is denoted by τ and $0 < \tau < \infty$ during each physical time step. By construction, $\mathbf{Q}$ is now a function of t and τ while V remains a function only of t. Provided that $\partial\mathbf{Q}/\partial\tau$ vanishes for large τ, during each time step, Equation 6 is recovered. Discretizing this term with a first-order backward difference about pseudo-time level $m+1$, and noting that previous values of the solution ($\mathbf{Q}^n$, $\mathbf{Q}^{n-1}$, etc) do not depend on the current pseudo-time level gives

$$\begin{aligned}V^{n+1}\frac{(\mathbf{Q}^{n+1,m+1} - \mathbf{Q}^{n+1,m})}{\Delta\tau} + \mathbf{Q}^n R_{GCL}^{n+1} + \frac{1}{\Delta t}[\phi_{n+1}(\mathbf{Q}^{n+1,m+1} - \mathbf{Q}^n)V^{n+1} + \phi_{n-1}(\mathbf{Q}^{n-1} - \mathbf{Q}^n)V^{n-1}\\ +\phi_{n-2}(\mathbf{Q}^{n-2} - \mathbf{Q}^n)V^{n-2} + \ldots] = \mathbf{R}^{n+1,m+1}\end{aligned}$$

It should be noted that there is no particular advantage to using a higher-order discretization of the pseudo-time term, just as first-order temporal discretization is typically used with local-time stepping methods to advance time-independent problems to a steady state. In fact, the use of pseudo time is completely analogous to local-time stepping for steady flows.

The nonlinear residual at the $m+1$ pseudo-time level may be linearized about the m^{th} level as

$$\mathbf{R}^{n+1,m+1} = \mathbf{R}^{n+1,m} + (\mathbf{Q}^{n+1,m+1} - \mathbf{Q}^{n+1,m})\frac{\partial \mathbf{R}^{n+1,m}}{\partial \mathbf{Q}}$$

Subtracting $V^{n+1}\phi_{n+1}\mathbf{Q}^{n+1,m}/\Delta t$ from both sides and defining $\Delta\mathbf{Q}^{n+1,m} = \mathbf{Q}^{n+1,m+1} - \mathbf{Q}^{n+1,m}$, gives, after rearrangement, the final form

$$\begin{aligned}\left[\left(\frac{V^{n+1}}{\Delta\tau} + \frac{V^{n+1}\phi_{n+1}}{\Delta t}\right)\mathbf{I} - \frac{\partial \mathbf{R}^{n+1,m}}{\partial \mathbf{Q}}\right]\Delta\mathbf{Q}^{n+1,m} = \mathbf{R}^{n+1,m} - \mathbf{Q}^n R_{GCL}^{n+1} - V^{n+1}\frac{\phi_{n+1}}{\Delta t}(\mathbf{Q}^{n+1,m} - \mathbf{Q}^n)\\ -V^{n-1}\frac{\phi_{n-1}}{\Delta t}(\mathbf{Q}^{n-1} - \mathbf{Q}^n) - V^{n-2}\frac{\phi_{n-2}}{\Delta t}(\mathbf{Q}^{n-2} - \mathbf{Q}^n) - \ldots\end{aligned} \tag{7}$$

Equation 7 is the means by which the solution is advanced in time in FUN3D; with the physical time step Δt set to ∞, and for a stationary mesh, Equation 7 reverts to the standard "steady state" scheme described in Reference 8. At each subiteration m, the linear system represented by Equation 7 is iteratively solved using a user-specified number of Gauss-Seidel sweeps with multi-color ordering. Within the non-linear iteration process between time steps, the equations are advanced in pseudo time with local time stepping (spatially varying time step $\Delta\tau$ based on a specified constant CFL

number) to accelerate the solution to a steady state in pseudo time, with the physical time step Δt being held constant over the entire mesh. The CFL number may be ramped during the subiterations to accelerate the convergence in pseudo time. To reduce computational time, the evaluation of the flux Jacobian $\partial \mathbf{R}^m / \partial \mathbf{Q}$ is periodically frozen during pseudo-time stepping.

The R_{GCL}^{n+1} term appearing in Equation 7 is evaluated as follows. Given coordinates for each point in the grid at time levels $n+1, n, n-1, \ldots$, velocities at each point are calculated using the same backward difference formula as used to discretize the time term in Equation 7 (i.e. the same sequence $\{ \phi_n \}$ is used). These grid-point velocities are then averaged to the locations that define the dual face, and the required integration over each face of the dual cell is carried out via the midpoint rule

$$R_{GCL} = \oint_{\partial V} \mathbf{W}^T \cdot \hat{\mathbf{n}} \, dS \cong \sum_{i=1}^{nfaces} (W_x^{avg} S_x + W_y^{avg} S_y + W_z^{avg} S_z)$$

where S_x, S_y, S_z are the directed areas of the dual face. The procedure is analogous to the one used to define the dual volumes. Note that if $\mathbf{W}$ was specified exactly, then evaluating the surface integral with the midpoint rule would result in a local error of order $\Delta h \Delta S \sim \Delta h^3$, where Δh is a characteristic dimension of the control-surface face. However, within the flow solver, mesh-point velocities are evaluated using the same N^{th}-order backward-difference formula that is used to discretize the flow equations in time. Note that mesh-point velocities could be evaluated analytically for rigid meshes, but finite differences are used for consistency with deforming meshes. Thus the midpoint velocities are known only to $O(\Delta t^N)$, which results in a additional local error of $O(\Delta t^N \Delta S) = O(\Delta t^N \Delta h^2)$. Consequently the local error in evaluating R_{GCL} is $O(\Delta h^3, \Delta t^N \Delta h^2)$. In particular, when evaluated on rigidly moving meshes (for which the cell volumes remain unchanged), R_{GCL} will be zero only to within local truncation error.

Temporal Error Control

The subiterative scheme outlined above has been implemented in a number of flow solvers. One issue that consistently arises is the choice of the number of subiterations to use. Converging the subiteration residual (given by the right-hand side of Equation 7) at each time step to machine zero is generally prohibitive in terms of computational cost. So the issue becomes one of striking a balance between computational cost and accuracy, i.e. to perform just enough subiterations to obtain a result that is essentially unchanged by additional subiterations. For the BDF schemes, one can estimate the temporal error incurred at each time step by examining the difference in residual contribution with two different levels of approximations of time derivatives.[14] The temporal error norm can be used as an exit criteria for terminating the subiteration loop of the dual time stepping process. Basically, the subiteration process is terminated when the residuals drop below a specified fraction of the temporal error norm. Such a strategy results in uniform temporal accuracy for all time steps, and eliminates the guess work for selecting iteration count or preselected residual reduction.

Mesh Deformation

A number of methods for deforming a given mesh to accommodate a changing boundary shape/orientation have been proposed. An excellent comparison of the various methods is given in Reference 15. The difficulty with mesh deformation is that negative volumes within the field may be generated for arbitrary displacements of the boundary. Treating the mesh-deformation problem as analogous to a linear elasticity problem has been found in practice to be the most robust method, and is the approach used in FUN3D. In the absence of body forces, the linear elasticity equations in differential form are

$$\nabla \cdot \bar{\bar{\sigma}} = 0 \tag{8}$$

where $\bar{\bar{\sigma}}$ is the stress tensor given by

$$\bar{\bar{\sigma}} = \lambda Tr(\bar{\bar{\epsilon}}) \bar{\bar{\mathbf{I}}} + 2\mu \bar{\bar{\epsilon}}$$

where Tr is the trace, $\bar{\bar{\mathbf{I}}}$ the identity tensor, λ and μ are material properties of the elastic material (Lamè constants), and $\bar{\bar{\epsilon}}$ is the strain tensor

$$\bar{\bar{\epsilon}} = \frac{1}{2} \left(\frac{\partial u_i}{\partial x_j} + \frac{\partial u_j}{\partial x_i} \right)$$

with $\mathbf{u} = (u_1, u_2, u_3)^T$ being the displacement vector.

Equation 8 may be cast in finite-volume form by integrating around a control volume and using Gauss' Theorem to give

$$\mathbf{R_{el}}(\mathbf{u}) = \oint_{\partial V} \lambda \frac{\partial u_i}{\partial x_i} \bar{\bar{\mathbf{I}}} \cdot \hat{\mathbf{n}} \, dS + \oint_{\partial V} \mu \left(\frac{\partial u_i}{\partial x_j} + \frac{\partial u_j}{\partial x_i} \right) \cdot \hat{\mathbf{n}} \, dS = 0 \tag{9}$$

The terms appearing in Equation 9 are evaluated in the same manner as the viscous terms in Equation 1.

Given an initial guess, $\mathbf{u_0}$, write the displacement as $\mathbf{u} = \mathbf{u_0} + \Delta\mathbf{u}$. Substitution into Equation 9 gives rise to the following system of linear equations for $\Delta\mathbf{u}$

$$\frac{\partial \mathbf{R_{el}}}{\partial \mathbf{u}} \Delta\mathbf{u} = -\mathbf{R_{el}}(\mathbf{u_0}) \tag{10}$$

Since Equation 9 is linear, the Jacobian $\partial\mathbf{R_{el}}/\partial\mathbf{u}$ does not depend on $\mathbf{u}$, and need be evaluated only once. Equation 10 is solved using the generalized minimum residual (GMRES) method.[16] The initial guess $\mathbf{u_0}$ for the displacements is by default taken as zero; optionally $\mathbf{u_0}$ may be taken from the mesh at the previous time step. For periodic motion of the boundary, starting from $\mathbf{u_0}$ = 0 each time step guarantees the deformed mesh will be periodic as well. However, starting from the displacement field from the previous time step typically results in faster convergence of the elasticity equations. Dirchlet boundary conditions are specified on non-symmetry boundaries to reflect the current boundary position/shape. On symmetry boundaries, the mesh motion is constrained to the plane.

The material properties λ and μ are related to Young's modulus E and Poisson's ratio ν by

$$\lambda = \frac{\nu E}{(1+\nu)(1-2\nu)}, \quad \mu = \frac{E}{2(1+\nu)}$$

Young's modulus is a positive quantity that indicates the stiffness of the material; larger values correspond to stiffer materials. Poisson's ratio indicates the relative amount that a material shrinks in transverse directions as it is extended in the axial direction. Physical materials have $-1 < \nu < \frac{1}{2}$, with cork having a value of approximately 0. To close the elasticity equations, two of the four parameters λ, μ, E, ν must be specified. Following Reference 15, E is taken as either inversely proportional to the dual-cell volume or inversely proportional to the distance from the nearest solid boundary. As volumes in a typical CFD mesh are smallest near the surface, both means of specifying E have the effect that cells near a boundary are stiffer than those further away, and thus move without significant distortion as the boundary is moved. Larger/more distant cells are thus relegated to accommodating larger deformations, but since they are larger, they are better able to tolerate the deformation without negative volumes. The Poisson ratio is given a uniform value of zero; limited numerical experiments showed little sensitivity to other constant values of ν. It should be noted that an earlier implementation of the elasticity equations in the FUN3D suite, aimed at design applications,[17] used a different paradigm wherein E was taken as a constant and Poisson's ratio was taken as inversely proportional to the cell aspect ratio. The current formulation has proven to be considerably more robust.

Motion Specification

The initial implementation of time-varying meshes within the FUN3D code was restricted to translation or rotation of the entire mesh as a solid body, with relatively limited options for motion specification.[5] The motion driver has been substantially revamped in order to increase generality, as well as to be able to handle overset and deforming meshes. Towards this end, the focus of motion specification has shifted from mesh motion to body motion; mesh motion is now a secondary specification tied to body motion. A body is defined by one or more solid boundaries within the mesh. Multiple bodies in relative motion can be accommodated, although some types of associated mesh motion preclude this. For example, a non-overset rigid mesh cannot support bodies in relative motion; however, a non-overset deforming mesh can support relative motion provided the motion is not too large. Overset meshes (rigid or deforming) can support arbitrarily large relative motion.

Within the code, motion is handled via application of a 4x4 transform matrix that contains both translation and orthonormal rotation components. Given a point at an initial position $(x, y, z)^T$, application of the transform matrix moves the point to its new position $(x', y', z')^T$

$$\begin{bmatrix} x' \\ y' \\ z' \\ 1 \end{bmatrix} = \begin{bmatrix} R_{11} & R_{12} & R_{13} & T_x \\ R_{21} & R_{22} & R_{23} & T_y \\ R_{31} & R_{32} & R_{33} & T_z \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}$$

Application of the inverse transform matrix moves the point back to the initial position; the inverse is computed using the method outlined in Reference 18, which has been found to provide an accurate inverse of the transform matrix.
Two often encountered transforms are a pure translation from the origin to a point $(x_0, y_0, z_0)^T$ and a pure rotation θ in the direction $\hat{\mathbf{n}}$ (unit vector) about the origin

$$[\mathbf{T_0}] = \begin{bmatrix} 1 & 0 & 0 & x_0 \\ 0 & 1 & 0 & y_0 \\ 0 & 0 & 1 & z_0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$[\mathbf{R_0}] = \begin{bmatrix} (1-cos\theta){n_x}^2 + cos\theta & (1-cos\theta)n_x n_y - n_z sin\theta & (1-cos\theta)n_x n_z + n_y sin\theta & 0 \\ (1-cos\theta)n_x n_y + n_z sin\theta & (1-cos\theta){n_y}^2 + cos\theta & (1-cos\theta)n_y n_z - n_x sin\theta & 0 \\ (1-cos\theta)n_x n_z - n_y sin\theta & (1-cos\theta)n_y n_z + n_x sin\theta & (1-cos\theta){n_z}^2 + cos\theta & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

where the right-hand rule in the direction of $\hat{\mathbf{n}}$ has been used to define the sense of θ. Other rotation matrices may arise depending on how the motion is specified, e.g. rotations specified terms of three Euler angles rather than a single rotation angle and direction (Euler axis) as above. An extremely useful feature of the transform matrix approach is that multiple transformations telescope via matrix multiplication. Thus, rotation about a point $(x_0, y_0, z_0)^T$, by an angle θ, in the direction $\hat{\mathbf{n}}$ is effected by first translating to the origin, performing the rotation, and then translating back to $(x_0, y_0, z_0)^T$, which is accomplished via three matrix multiplications to give the complete transform matrix $[\mathbf{T_m}]$

$$[\mathbf{T_m}] = [\mathbf{T_0}][\mathbf{R_0}][\mathbf{T_0}]^{-1}$$

The telescoping property of the transform matrices is also useful for tracking motions of one body relative to another. For example, a wing may be translating up and down, while at the same time, a flap may be pitching about a hinge line fixed on the wing. It is natural to describe the pitching motion of the flap in its own coordinate system, which at t=0 (say) is the same as the wing coordinate system. If the transform matrix describing the plunging of the wing relative to its initial position in an inertial reference frame is given by $[\mathbf{T_m}]_{\mathbf{wing}}$, and the transform matrix of the pitching motion of the flap relative to a hinge line defined in the flap coordinate system is given by $[\mathbf{T_m}]_{\mathbf{flap}}$, then the position of the flap, relative to the inertial frame, may by computed using the composite transform

$$[\mathbf{T_m}] = [\mathbf{T_m}]_{\mathbf{wing}}[\mathbf{T_m}]_{\mathbf{flap}}$$

The order of matrix multiplication is important: post multiplication of the parent transform takes coordinates from the child system into the parent system, which is then moved relative to the inertial frame according to the parent transform. The example above is for a simple, one-generation (parent-child) composite motion, but the concept may be extended to any number of generations.

Rotating Noninertial Reference Frame

The moving-body infrastructure described above is quite general and can be applied to a large class of problems. There are however, specialized situations where the general approach is not the most efficient approach. A very common scenario is the analysis of a single body rotating in isolation. Examples include helicopter rotors in hover and propellers in forward flight, assuming fuselage interactions can be neglected. In such situations, the problem can be formulated in a noninertial reference frame rotating with the body. If the problem is steady in the noninertial frame, then a significant reduction in computation time can be achieved over the time required to solve the problem in the inertial frame, where a time-dependent problem must be solved.

In the following, it is assumed that the body angular velocity $\mathbf{\Omega}$ is constant and that the mesh is rigid. Furthermore, in the FUN3D implementation, it is assumed that the reference frame undergoes no motion other than rotation. Although the noninertial-frame formulation could be generalized to accommodate arbitrary motions and deforming meshes, any significant advantage over the time-dependent inertial formulation would disappear. A more general treatment of noninertial frames is given in Reference 19.

The form of the equations for the noninertial formulation depends on whether the equations are written in terms of absolute velocity (velocity relative to the inertial frame) or velocity relative to the noninertial frame. Let $\mathbf{u_r}$ be the fluid velocity relative to the rotating frame, and $\mathbf{u}$ be the absolute velocity. The two are related by

$$\mathbf{u} = \mathbf{u_r} + \mathbf{\Omega} \times \mathbf{R} \tag{11}$$

where $\mathbf{R}$ is the distance from the axis of rotation.

The unsteady Navier-Stokes equations for the relative velocity in a frame rotating with constant angular velocity, in the absence of eternally-applied forces or heat addition, are given by (see for example, Hirsch[20])

$$\frac{\partial \rho}{\partial t} + \nabla \cdot [\rho \mathbf{u_r}] = 0$$

$$\frac{\partial (\rho \mathbf{u_r})}{\partial t} + \nabla \cdot \left[\rho \mathbf{u_r}\mathbf{u_r} + p\bar{\bar{\mathbf{I}}} - \bar{\bar{\tau}}\right] = -2\rho(\mathbf{\Omega} \times \mathbf{u_r}) - \rho\mathbf{\Omega} \times (\mathbf{\Omega} \times \mathbf{R}) \tag{12}$$

$$\frac{\partial E_r}{\partial t} + \nabla \cdot [\mathbf{u_r}(E_r + p) + k\nabla T - \bar{\bar{\tau}} \cdot \mathbf{u_r}] = 0$$

where the first term on the right-hand side of the momentum equations is the Coriolis force per unit volume, the second term is the centrifugal force per unit volume. In the energy equation, $E_r = p/(\gamma-1) + \frac{1}{2}\rho\left|\mathbf{u_r}\right|^2 - \frac{1}{2}\rho\left|(\mathbf{\Omega}\times\mathbf{R})\right|^2 = E - \rho\mathbf{u_r}\cdot(\mathbf{\Omega}\times\mathbf{R})$ is the total energy per unit volume in the rotating frame. These are the conservation laws seen by an observer in the rotating frame, and $\mathbf{u_r}$ is the velocity field observed relative to the rotating frame. An alternate set of equations is obtained by substituting the expression for $\mathbf{u_r}$ from Equation 11 into Equations 12 and using the identity $\nabla\cdot(\mathbf{\Omega}\times\mathbf{R}) = 0$ to give

$$\frac{\partial\rho}{\partial t} + \nabla\cdot[\rho(\mathbf{u}-\mathbf{\Omega}\times\mathbf{R})] = 0$$

$$\frac{\partial(\rho\mathbf{u})}{\partial t} + \nabla\cdot\left[\rho\mathbf{u}(\mathbf{u}-\mathbf{\Omega}\times\mathbf{R}) + p\bar{\bar{\mathbf{I}}} - \bar{\bar{\tau}}\right] = -\rho(\mathbf{\Omega}\times\mathbf{u}) \tag{13}$$

$$\frac{\partial E}{\partial t} + \nabla\cdot[(\mathbf{u}-\mathbf{\Omega}\times\mathbf{R})(E+p) + (\mathbf{\Omega}\times\mathbf{R})p + k\nabla T - \bar{\bar{\tau}}\cdot\mathbf{u}] = 0$$

Equations 13 are the governing equations, in terms of absolute velocity, in a reference frame rotating with a constant angular velocity $\mathbf{\Omega}$. It is easily verified that stress tensor $\bar{\bar{\tau}}$ is invariant with the change from relative to absolute velocities. Reference 21 states that using the absolute-velocity formulation allows for more accurate evaluation of the fluxes in a finite-volume scheme. Limited experience with the relative-velocity formulation in FUN3D prior to adopting the absolute-velocity formulation suggests that is indeed the case.

Although Equations 13 are in differential form while Equations 1 are in control-volume form, inspection shows that apart from the source term on the right-hand side of the momentum equations in the noninertial formulation, the individual terms are nearly identical, with the frame velocity $\mathbf{\Omega}\times\mathbf{R}$ replacing the grid velocity $\mathbf{W}$. Thus, the coding infrastructure to evaluate the relative flux through a control volume for moving meshes can be utilized for noninertial reference-frame problems. Using Equations 13 for the absolute velocity requires no modification in the flow solver to either the far-field reference state or the relation between conserved energy and pressure. A single initial evaluation of $\mathbf{\Omega}\times\mathbf{R}$ throughout the field is performed, without having to recompute face speeds, grid normals or volumes at each time step as required in the moving-grid formulation. However, since the source terms are a function of the velocity field, they must be evaluated each time step/iteration. Although the noninertial-frame formulation is anticipated to be applied to steady (in the noninertial frame) problems, the governing equations are implemented in time-dependent form, since the baseline solver marches in (pseudo) time to a steady state. Furthermore, this allows for a time-accurate simulation in the noninertial frame if desired/required.

It should be noted that within the assumptions of the specific noninertial frame considered here (pure rotation with constant $\mathbf{\Omega}$), the far field conditions are not completely arbitrary. For example, consider a typical situation in which the absolute velocity in the far field is constant, $\mathbf{u}_\infty$, with corresponding constant conditions ρ_∞ and p_∞. In that case the continuity and energy equations are satisfied identically in the far field, but the momentum equation requires

$$\mathbf{\Omega}\times\mathbf{u}_\infty = 0$$

which requires either 1) $\mathbf{u}_\infty = 0$ or 2) $\mathbf{u}_\infty$ parallel to $\mathbf{\Omega}$. In terms of an application scenario, if a helicopter blade rotates about (say) the vertical axis, then the noninertial formulation above is only applicable for hover (condition 1) or ascending/descending flight (condition 2).

Overset Grids

Deforming meshes can accommodate moderately large surface motions within a single mesh system, but if the motion becomes too large, negative cell volumes can result. To overcome this, overset meshes can be used for applications involving large motions; rotorcraft and store separation are two examples. The overset method was first implemented in the FUN3D solver by O'Brien.[22] The implementation uses the Donor interpolation/Receptor Transaction library[23] (DiRTlib) to facilitate the use of overset grids in a parallel environment without extensive modification to the flow solver. As for non-overset meshes, the flow solver continues to operate on a single mesh (partitioned for multiple processors). For overset meshes, points are flagged with an identifier for the particular component mesh with which they are associated. With a few simple calls within the flow solver, DiRTlib handles the equation blanking and solution interpolation required for the overset method. Linear interpolation of the solution between points associated with different component meshes is used with two layers of donor points, consistent with the underlying second-order spatial accuracy of the baseline solver. Points within holes (blanked regions) are assigned solution values by averaging the solution at neighboring points. This also helps ensure that for moving mesh problems, points that were blanked at previous time steps do not suddenly become unblanked with initial freestream values. Orphan points (points without valid donors), if any, are assigned solution values in the same manner as hole points.

DiRTlib does not perform composite grid assembly, cut holes (establish blanking), or determine the requisite interpolation coefficients. For that, the Structured, Unstructured, and Generalized overset Grid AssembleR[24] (SUGGAR) program is employed. SUGGAR may be compiled as a stand-alone executable or as a callable library. In a preprocessing step prior to the initial flow-solver execution, SUGGAR reads two or more component meshes and creates a single composite mesh with the configuration in the initial position, along with a file (a DCI file in SUGGAR/DiRTlib parlance) identifying points corresponding to each component mesh, blanked points, and interpolation coefficients. This composite mesh is then partitioned for execution on multiple processors, after which the flow-solver execution may begin. The current usage of SUGGAR within the FUN3D solver is as a library,[25] with one processor devoted to the SUGGAR task. When called upon, this processor receives updated grid information from the other processors, computes the new overset connectivity data, and then sends that data back to the flow-solve processors. A new DCI file containing the updated connectivity data is also written out for subsequent reuse, if desired. Currently, only a single processor runs the SUGGAR task, since SUGGAR does not parallelize well. An updated version of SUGGAR, SUGGAR++, is due to be released soon and is expected to offer significantly better parallel performance than its predecessor. The current FUN3D parallel-processing implementation supports multiple MPI (Message Passing Interface) communicators, and will be able to take advantage of multiple processors running SUGGAR++ when that code is available.

In applications involving overset deforming meshes, the situation may occur where large parts of the mesh undergo no deformation. In this case some efficiency is lost because the linear elasticity solver, by default, operates on all points of the mesh. Thus, for overset deforming meshes, the non-deforming points are masked from solution in the linear elasticity equations. In rotorcraft applications, this means that points in the fuselage/background mesh are masked; these points may in fact be the majority of the points in the mesh. Simply masking the points is typically not sufficient to achieve increased efficiency. During the preprocessing step of mesh partitioning for parallel execution, the deforming points need to be weighted so that an equal number of deforming points are given to each processor.

Structural Dynamics

For applications in which motion of the structure is an important component of the simulation, for example, the prediction of wing flutter, a linear structural dynamics model has been added. The formulation and implementation in FUN3D is the same as that used in both the CAP-TSD[26] transonic small disturbance solver and the CFL3D[27] structured-grid Navier-Stokes solver. The formulation is briefly described below.

The coupled linear structural dynamics equations can be written as

$$[\mathbf{M}]\ddot{\delta} + [\mathbf{D}]\dot{\delta} + [\mathbf{K}]\delta = \mathbf{F}_a \tag{14}$$

where $[\mathbf{M}]$ is the mass matrix, $[\mathbf{D}]$ the damping matrix, $[\mathbf{K}]$ the stiffness matrix, $\delta(\mathbf{x}, t)$ the displacement, and $[\mathbf{F}_a(t)]$ the loading, here assumed to be from aerodynamic forces only. The displacements are written as an expansion in terms of natural vibration modes $\{\ \phi_i(\mathbf{x})\ \}$

$$\delta = \sum_{i=1}^{N_{modes}} \mathbf{q}_i(t)\phi_i(\mathbf{x})$$

where the coefficients of the series, $\{\mathbf{q}_i\}$, are referred to as the generalized coordinates. The vibration modes have associated natural frequencies $\{\omega_i\}$ and are orthonormalized with the mass matrix, so that $\phi^T[\mathbf{M}]\phi = [\mathbf{I}]$. Substitution of the series representation into Equation 14 and multiplying by ϕ^T yields

$$\ddot{\mathbf{q}} + [\zeta]\dot{\mathbf{q}} + [\omega]\mathbf{q} = \phi^T\mathbf{F}_a = \hat{\mathbf{F}}_a \tag{15}$$

where $\hat{\mathbf{F}}_a$ is the generalized aerodynamic force and $[\zeta] = \phi^T[\mathbf{D}]\phi$ and $[\omega] = \phi^T[\mathbf{K}]\phi$ both are diagonal matrices, so that Equation 15 is an uncoupled system. For the i^{th} mode, the element of $[\omega]$ is ${\omega_i}^2$ and the element of $[\zeta]$ is $2\omega_i\zeta_i$. Since the equations are now decoupled, they can be written as a sequence of scalar equations, and it is convenient to transform these scalar second-order equations to a system of first-order equations through the substitution $\mathbf{x}_i = (\mathbf{q}_i, \dot{\mathbf{q}}_i)^T$ to yield

$$\begin{bmatrix} \dot{x}_{i1} \\ \dot{x}_{i2} \end{bmatrix} + \begin{bmatrix} 0 & 1 \\ {\omega_i}^2 & 2\omega_i\zeta_i \end{bmatrix} \begin{bmatrix} x_{i1} \\ x_{i2} \end{bmatrix} = \begin{bmatrix} 0 \\ \hat{F}_{ai} \end{bmatrix}$$

The structural equations are integrated in time following Reference 28. In this approach, the system of first-order equations is integrated via a predictor-corrector scheme via

$$predictor : \tilde{\mathbf{x}}^{n+1} = [\Phi]\mathbf{x}^n + \frac{1}{2}[\Theta](3\hat{\mathbf{F}_a}^n - \hat{\mathbf{F}_a}^{n-1})$$

$$corrector : \mathbf{x}^{n+1} = [\Phi]\mathbf{x}^n + \frac{1}{2}[\Theta](\hat{\mathbf{F}_a}^{n+1} + \hat{\mathbf{F}_a}^n)$$

where $[\Phi]$ is the state-transition matrix and $[\Theta]$ is a convolution integral of $[\Phi]$.

The linear structural model described above is adequate for applications with relatively small deflections, such as the prediction of the onset of flutter (as opposed to large deflections that occur as flutter progresses). For cases in which deflections are large enough to require a nonlinear structural model, provision is made in the flow solver to read in new surface shapes as the solution evolves, and to output distributed normal force and shear force distributions on the surface. The new surface shape could be provided by a nonlinear finite element model (FEM) together with appropriate middleware (not part of the FUN3D software suite) to transfer the deflections from the FEM grid points to the CFD grid points, and to transfer the normal and shear forces from the CFD grid points to the FEM grid points. An example of coupling FUN3D to a nonlinear structural model, for the specialized case of flexible helicopter rotor blades, is given in Reference 29.

6DOF Motion

The mesh motion in FUN3D has been coupled with the six-degree-of-freedom (6DOF) library described in Reference 30. Together with the overset methodology, this greatly expands the range of problems able to be addressed by the flow solver. An extensive range of subroutine calls are available in the library for initialization, addition and deletion of both aerodynamic forces and imposed forces, and time advancement. The library has the ability to track parent/child motion. Currently the motion description in the library is limited to rigid bodies.

Results

Freestream Preservation

To demonstrate the necessity of including the GCL in deforming mesh solutions, the 2D airfoil shown in Figure 1 is pitched up and down with an amplitude of 40°. The flow field is initialized as uniform, and Riemann boundary conditions are set on all boundaries, with the external state taken as the same uniform flow. Thus, the solution should remain as the initial uniform state for all time, resulting in zero residual for all time, regardless of the mesh deformation. As seen in Figure 2, if the GCL is included in Equation 7, the residual remains zero to machine precision. However, if the GCL term is not included in Equation 7, the residual does not remain zero (although it is small), indicating that the solution departs from freestream as the mesh is deformed.

Order Property Verification

Verification that the temporal discretization of Equation 7 achieves design order is carried out for both rigid and deforming meshes. In particular, the 2^{nd}-order accurate integration scheme referred to as BDF2opt[14] is verified. The BDF2opt scheme results from a blend of 2^{nd} and 3^{rd} order BDF schemes. Although formally second-order accurate, BDF2opt has a smaller coefficient for the error term than the standard 2^{nd}-order BDF scheme, and retains both A- and L-stability properties of the standard 2^{nd}-order BDF scheme. In contrast, the standard 3^{rd}-order BDF scheme exhibits a small region of instability and so can be unreliable in use. Three-dimensional laminar flow over a pitching wing is computed on very coarse mesh consisting of approximately 16,000 nodes and 91,000 tetrahedra. The wing pitched about the root quarter chord according to $\theta = \theta_m sin(2\pi kt)$ where θ_m = 4°, k = 0.01, and t is the nondimensional time. Figure 3 depicts the wing surface and symmetry-plane mesh for the wing at the minimum and maximum pitch angles for the deforming-mesh case. In both cases the "exact" solution is taken as the solution obtained using 3200 time steps per pitch cycle. For the deforming-mesh case the elasticity equations are solved to machine zero at each time step. In both cases the temporal error controller is used to insure the residual is converged to two orders of magnitude below the estimated temporal error. Errors in normal force and pitching moment relative to the "exact" solutions are plotted in Figure 4 for time steps corresponding to 100, 200, 400, 800, and 1600 steps per pitch cycle. For both rigid and deforming mesh motion, the design order of 2 is achieved consistent with the BDF2opt scheme.

Forced Pitching

Next, the mesh deformation capability is employed for the simulation of a Blended Wing Body (BWB) configuration undergoing forced pitching. Figure 5 shows a 3 percent-scale model of this configuration mounted for testing in the NASA Langley 14- $\times$ 22-Foot Subsonic Wind Tunnel. This configuration was previously analyzed in References 5 and 31 using rigid mesh motion. Here, results for both rigid mesh motion and mesh deformation are compared for the case of a mean angle of attack of 8° and a pitch amplitude of 5°. In both cases the initial mesh is the same, consisting of approximately 4.9 million nodes and 29.2 million tetrahedral cells. The computational grid shown in Figure 6 and used for the pitch analysis was a "free-air" mesh that did not include either the wind-tunnel walls or the mounting apparatus for the test article. Time steps corresponding to 1600 steps per pitch cycle were used. The results for pitching-moment and normal-force variation with angle of attack are shown in Figures 7 and 8, respectively. In both cases the experimentally-measured values are shown for reference. The two mesh-motion schemes are seen to give virtually identical results. Compared to the experimentally-measured moment and force data, the computations reproduce the overall shapes of the response curves, though there is an offset from the data. The offset is especially pronounced in pitching moment. In Reference 31 a series of static-grid computations were performed on this configuration with a mesh that included the mounting apparatus (but not the tunnel

walls). The computed static pitching moment and normal force coefficients from Reference 31, with the mounting post present in the grid, are included in Figures 7 and 8. Also included are the corresponding static data computed using the free-air mesh and the static data from the experiment. Including the mounting post results in considerably better agreement between computation and experiment for the static pitching-moment. The predicted static normal-force coefficient is also slightly improved compared to the data when the experimental setup is more faithfully reproduced. Although the post was not modeled in the dynamic pitching computations, the static force and moment results suggest that doing so would reduce the offset between computation and experiment.

Noninertial Frame vs. Inertial Frame

The noninertial and moving grid capabilities are demonstrated and compared by simulating an isolated rotor in hover. The rotor used for the computations is the Tilt Rotor Aeroacoustic Model (TRAM),[32,33] and shown installed for testing in the Duits-Nederlandse Windtunnel Large Low-speed Facility (DNW-LLF) in Figure 9. The TRAM is a quarter-scale model of the V-22 rotor and nacelle, although the blade root fairing is different than that on the V-22 rotor. For the results presented here, the collective pitch is 14°, with a tip Mach number of 0.62. The grid employed for these calculations was a coarse mesh consisting of approximately 1.2 million nodes and 7.2 million tetrahedra; the nacelle is not modeled. Computations using finer meshes, together with detailed comparison with experimental data are the subject of a future paper. A relatively coarse mesh is sufficient here to illustrate the consistency of the rotating, noninertial reference-frame formulation and the inertial frame, moving-grid formulation.

First a converged, steady-state solution using the noninertial formulation was obtained. For expediency, time-accurate, moving-grid computations using the coarsest time step were restarted from the noninertial results, and the solution was continued until fully converged thrust and torque coefficients were again obtained, after approximately two rotor revolutions. Subsequent time-step refinements were restarted from the preceding time-step result. Time-accurate results were obtained using a 2^{nd}-order accurate (in time) integration scheme referred to as BDF2opt,[14] with time steps corresponding to 1°, 1/2°, 1/4°, 1/8° and 1/16° azimuth change of the rotor per time step. The results for the computed thrust and torque coefficients are shown in Table 1. For the purposes of this time-step refinement study, the noninertial values are taken as the "exact" values on the given grid. Time-step refinement for the time-accurate, moving-grid formulation shows that indeed the time-accurate results approach the noninertial results as the step size tends toward zero.

	$C_Q \times 10^2$	$C_T \times 10^1$
Noninertial	0.1842731	0.1464749
1° / step	0.1848883 (0.334)	0.1469585 (0.330)
1/2° / step	0.1844665 (0.105)	0.1466321 (0.107)
1/4° / step	0.1843304 (0.031)	0.1465230 (0.033)
1/8° / step	0.1842875 (0.0078)	0.1464866 (0.0080)
1/16° / step	0.1842745 (0.0008)	0.1464772 (0.0016)

Table 1 Convergence of thrust and torque coefficients toward noninertial frame values with time-step refinement. Numbers in parentheses indicate per-cent error compared to the noninertial values.

The results tabulated above are also plotted in Figure 11. The BDF2opt scheme is formally second order, and it is observed that this order property is indeed achieved for the smaller time steps; for the smallest time step the torque coefficient appears to converge with a higher order. The errors in the moving grid results for engineering quantities such as the thrust and torque coefficients, even for the largest time steps, are fairly small - less than one-half percent. A more discriminating indicator of the convergence of the time-accurate, moving-grid formulation toward the noninertial formulation is indicated by surface-restricted streamlines, as shown in Figure 12. The streamlines are based on the relative velocity field near the surface as seen by an observer moving with the rotor, i.e. Equation 11 has been applied to the absolute velocities obtained from both the noninertial and moving-grid solutions. The gross features of the inboard separation region that extends aft from roughly the 50% chord position are captured using the largest time step corresponding to 1°. However, some details are noticeably different from the noninertial streamlines, particularly the outward turning of the streamlines outboard of the separation region. This suggests that the largest time steps are not adequate to resolve the balance between the centrifugal, centripetal, pressure and viscous forces in this region, and to keep the streamlines moving in the chordwise direction. As the time step is reduced, all details of the streamlines from the two formulations coalesce as expected.

Flutter Onset

To demonstrate the structural dynamics model, transonic flow over the AGARD 445.6 Aeroelastic Wing[34]is examined. This geometry is widely used to validate computational aeroelastic tools. The 445.6 wing has a 45° swept planform with a taper ratio of 0.658 and an aspect ratio of 1.65, with a NACA 65A004 airfoil section. The wing was structurally weakened to enhance its tendency to flutter. The first four natural vibration modes are used in the structural-dynamics model for the results presented here, and are depicted in Figure 13. The modes have natural frequencies of 9.60, 38.2, 48.35 and 91.54 Hz. Inviscid results for Mach 0.9 at an angle of attack of 0° are presented here; zero structural damping was assumed. A tetrahedral mesh containing 430,000 nodes was used, and is shown in Figure 14. The aeroelastic simulations are begun from a steady-state, rigid-wing solution. The solution is then continued in a time-accurate manner, with the modal velocities being given a small initial value. The structural response is then observed to see if the initial transient grows, decays, or is undamped. Normally, for aeroelastic simulations a steady, static-aeroelastic solution is first obtained, but since the 445.6 wing has a symmetrical section, the static-aeroelastic deflection at $\alpha = 0$ is zero.

Figure 15 shows the computed response for the four modes at a dynamic pressure of 75.0 psf. The response is essentially undamped following the initial transient, indicating this dynamic pressure is at or very close to the neutral stability point. In contrast, Figure 16 shows the computed response for a dynamic pressure of 89.3 psf. The response of mode 1 (first bending) is clearly divergent, indicating this dynamic pressure is above the flutter boundary. These responses agree well with the corresponding results presented in Reference 3, which were obtained using CFL3D and a structured mesh of comparable size.

Overset/6DOF

The final demonstration problem is the 6DOF motion of an under-wing store. Overset grids are used to handle the relative motion between two rigid bodies. The mass properties of the store and the test conditions are chosen to match the test described in Reference 35. The grid used here for illustrative purposes is a very coarse mesh taken from SUGGAR training materials and is intended for demonstrating the basic overset mesh techniques, not for accurate CFD simulation. The geometry in the grid is significantly modified in two respects compared to the test article: 1) the large-diameter mounting sting projecting aft behind the store is removed and replaced with an aerodynamically-shaped boattail and 2) the under-wing pylon is removed entirely.

Because of the aforementioned geometry compromises, computed forces and moments on the store, even in steady state at the stowed position, are significantly different from the measured values. Therefore, for the current purpose of demonstrating the mechanics of linking the flow solver and 6DOF module, the force and moment coefficients computed by the solver are replaced by the experimentally-measured values before the 6DOF motion equations are integrated. In all other aspects this demonstration is of the complete CFD/6DOF system: given force and moment coefficients, the equations of motion for a rigid body are integrated forward in time via a call to the 6DOF library, the resulting CG position and body-orientation data are used within the flow solver to move the store and its surrounding mesh to the new position, and new overset connectivity information is then computed by a call to SUGGAR, a new flow solution is computed, and the process repeated. Figure 17 shows the configuration at t = 0 together with a slice through the overset mesh, illustrating the holes cut in the meshes, as well as the overlap regions. The coarseness of the mesh is evident. Figure 18 shows the corresponding configuration at t = 0.3 seconds, positioned via the computed 6DOF motion. Figures 19 and 20 show the computed CG velocity and store rotational velocity as functions of time along with the experimentally measured data. The computed linear velocities are in excellent agreement with the measured data. There are small differences between the computed and measured angular velocities, but these quantities are known to be quite sensitive to uncertainties in ejector forces and inertia properties.

Summary

A Navier-Stokes solver for unstructured grids has been extended to allow a wide range of applications involving moving geometries and unsteady flows. Rigid, deforming, or overset grids may be used, allowing the most appropriate/efficient grid motion scheme to be employed for the problem. The importance of satisfying the Geometric Conservation Law has been demonstrated for a deforming-mesh problem. Design-order properties of the modified solver have been verified for one particular choice of the time-advancement scheme (BDF2opt). Example cases have been presented to show some of the potential applications, including aeroelastic analysis and 6DOF motion; many more are possible.

Acknowledgments

The authors would like to thank Dr. Robert Bartels of the Aeroelasticity Branch at NASA Langley Research Center for providing the mode shapes of the AGARD 446.5 Aeroelastic Wing. The helpful suggestions of Ms. Elizabeth Lee-Rausch regarding the TRAM computations are gratefully acknowledged. The authors would also like to thank Dr. Ralph Noack of The Pennsylvania State University for the many helpful discussions on the use of SUGGAR and DiRTlib, as well as for the mesh used for the wing-store problem.

References

[1]Cenko, A., Gowanlock, D., Lutton, M., and Tutty, M., "F/A-18C/JDAM Applied Computational Fluid DynamicsChallenge II Results," AIAA Paper 2007-0795, Jan. 2000.

[2]Potsdam, M., Yeo, H., and Johnson, W., "Rotor Airloads Prediction Using Aerodynamic/Structural Coupling," *American Helicopter Society 60th Annual Forum Proceedings*, 2004.

[3]Silva, W. A. and Bartels, R. E., "Development of reduced-order models for aeroelastic analysis and flutter prediction using the CFL3Dv6.0 code," *Journal of Fluids and Structures*, Vol. 19, 2004, pp. 729–745.

[4]Schutte, A., Einarsson, G., Schoning, B., Raichle, A., Monnich, W., Alrutz, T., Neumann, J., and Heinecke, J., "Numerical Simulation of Maneuvering Combat Aircraft," *High Performance Computing in Science and Engineering '05*, Springer Berlin Heidelberg, 2006.

[5]Biedron, R. T., Vatsa, V. N., and Atkins, H. L., "Simulation of Unsteady Flows Using an Unstructured Navier-Stokes Solver on Moving and Stationary Grids," AIAA Paper 2005-5093, June 2005.

[6]Hirt, C. W., Amsden, A. A., and Cook, J. L., "An Arbitrary Lagrangian-Eulerian Computing Method for All Flow Speeds," *Journal of Computational Physics*, Vol. 14, No. 3, 1974, pp. 227–257.

[7]Thomas, P. D. and Lombard, C. K., "Geometrical Conservation Law and Its Application," *AIAA J.*, Vol. 17, No. 10, Oct. 1978, pp. 1030–1037.

[8]Anderson, W. K. and Bonhaus, D. L., "An Implicit Upwind Algorithm for Computing Turbulent Flows on Unstructured Grids," *Computers and Fluids*, Vol. 23, No. 1, 1994, pp. 1–22.

[9]Roe, P. L., "Approximate Riemann Solvers, Parameter Vectors, and Difference Schemes," *Journal of Computational Physics*, Vol. 43, 1981, pp. 357–372.

[10]Spalart, P. R. and Allmaras, S. R., "A One-Equation Turbulence Model for Aerodynamic Flows," *La Recherche Aerospatiale,* No. 1, 1994, pp. 5–21.

[11]Menter, F. R., "Two-Equation Eddy-Viscosity Turbulence Models for Engineering Applications," *AIAA Journal*, Vol. 32, No. 8, 1994, pp. 1598–1605.

[12]Morton, S. A., Melville, R. B., and Visbal, M. R., "Accuracy and Coupling Issues of Aeroelastic Navier-Stokes Solutions on Deforming Meshes," AIAA Paper 1997-1085, April 1997.

[13]Rumsey, C., Sanetrik, M., Biedron, R., Melson, N., and Parlette, E., "Efficiency and Accuracy of Time-Accurate Turbulent Navier-Stokes Computations," *Computers and Fluids*, Vol. 25, No. 2, 1996, pp. 217–236.

[14]Vatsa, V. and Carpenter, M. H., "Higher-Order Temporal Schemes with Error Controllers for Unsteady Navier-Stokes Equations," AIAA Paper 2005-5245, June 2005.

[15]Yang, Z. and Mavriplis, D. J., "Unstructured Dynamic Meshes with Higher-order TimeIntegration Schemes for the Unsteady Navier-Stokes Equations," AIAA Paper 2005-1222, Jan. 2005.

[16]Saad, Y. and Schultz, M. H., "GMRES: A Generalized Minimum Residual Algorithm for Solving Nonsymmetric Linear Systems," *SIAM Journal of Scientific and Statistical Computing*, Vol. 7, 1986, pp. 856–869.

[17]Nielsen, E. J. and Anderson, W. K., "Recent Improvements in Aerodynamic Design Optimizationon Unstructured Meshes," *AIAA Journal*, Vol. 40, No. 6, 2002, pp. 1155–1163.

[18]Shagam, J., "Fast Inversion of Orthonormal Transformation Matrices," New Mexico State University Report NMSU-CS-2003-003, April 2003, http://www.cs.nmsu.edu/CSWS/techRpt/2003-003.pdf.

[19]Limache, A. C. and Cliff, E. M., "Aerodynamic Sensitivity Theory for Rotary Stability Derivatives," *Journal of Aircraft*, Vol. 37, No. 4, 2000, pp. 676–683.

[20]Hirsch, C., Wiley New York, 1988.

[21]Agarwal, R. K. and Deese, J. E., "Euler Calculations For Flowfield of a Helicopter Rotor in Hover," AIAA Paper 1986-1782, 1986.

[22]O'Brien, D., *Analysis of Computational Modeling Techniques for Complete Rotorcraft Configurations*, Ph.D. thesis, Georgia Institute of Technology, 2006.

[23]Noack, R. W., "DiRTlib: A Library to Add an Overset Capability to Your Flow Solver," AIAA Paper 2005-5116, June 2005.

[24]Noack, R. W., "SUGGAR: a General Capability for Moving Body Overset Grid Assembly," AIAA Paper 2005-5117, June 2005.

[25]Noack, R. W., "Improvements to SUGGAR and DiRTlib for Overset Store Separation Simulations," AIAA Paper 2009-340, Jan. 2009.

[26]Bennett, R. M., Batina, J. T., and Cunningham, H. J., "Wing Flutter Calculations with the CAP-TSD Unsteady Transonic Small Disturbance Program," AIAA Paper 1988-2347, 1988.

[27]Bartels, R. E., Rumsey, C. L., and Biedron, R. T., "CFL3D Version 6.4 - General Usageand Aeroelastic Analysis," NASA TM TM-2006-214301, 2006.

[28]Edwards, J. W., Bennett, R. M., Whitlow W., J., and Seidel, D. A., "Time-Marching Transonic FlutterSolutions Including Angle-of-Attack Effects," *AIAA Journal*, Vol. 20, No. 11, 1983, pp. 899–906.

[29]Biedron, R. T. and Lee-Rausch, E. M., "Rotor Airloads Prediction Using Unstructured Meshes and LooseCFD/CSD Coupling," AIAA Paper 2008-7341, Aug. 2008.

[30]Koomullil, R. P. and Prewitt, N. C., "A Library Based Approach for Rigid Body Dynamics Simulation," AIAA Paper 2007-4476, June 2007.

[31]Hall, R. M., Biedron, R. T., Ball, D. N., Bogue, D. R., Chung, J., Green, B. E., Grismer, M. J., Brooks, G. P., and Chambers, J. R., "Computational Methods for Stability and Control (COMSAC): The Time Has Come," AIAA Paper 2005-6121, Aug. 2005.

[32]Young, L. A., Booth Jr., E. R., Yamauchi, G. K., Botha, G., and Dawson, S., "Overview of the Testing of a Small-Scale Proprotor," *American Helicopter Society 55th Annual Forum Proceedings*, 1999.

[33]Swanson, S. M., McCluer, M. S., Yamauchi, G. K., and Swanson, A. A., "Airloads Measurements from a 1/4-Scale Tiltrotor Wind Tunnel Test," *25th European Rotorcraft Forum, Rome, Italy*, 1999.

[34]Yates Jr., E. C., "AGARD Standard Aeroelastic Configurations for Dynamic Response. CandidateConfiguration 1 - Wing 445.6," NASA TM TM-100492, Aug. 1987.

[35]Heim, R. E., "CFD Wing/Pylon/Fined Store Mutual Interference Wind Tunnel Experiment," Arnold Engineering Development Center Report AEDC-TSR-91-P4, Feb. 1991.

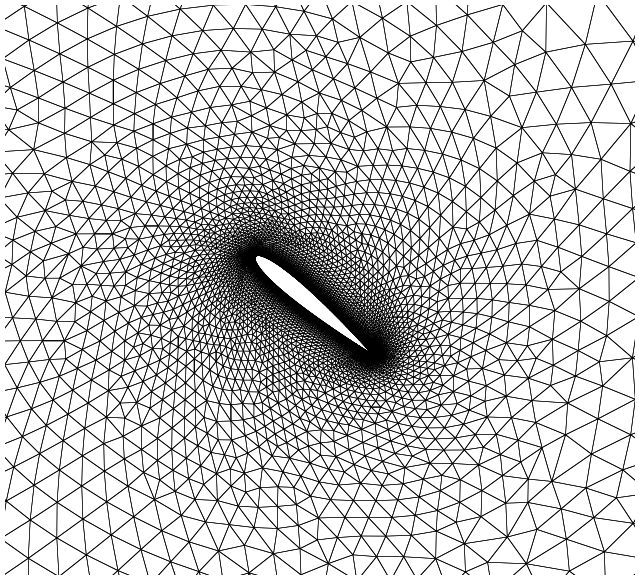

Fig. 1 Deforming mesh used in GCL test with airfoil at maximum pitch angle.

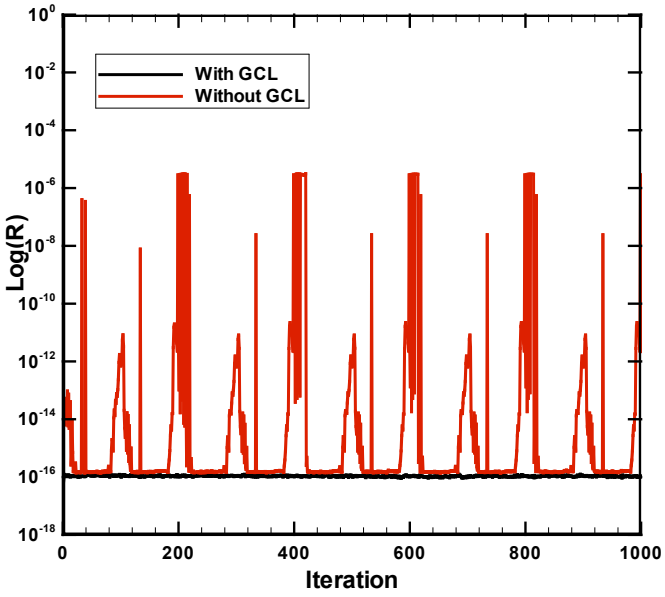

Fig. 2 Effect of GCL on the maintenance of uniform freestream flow with mesh deformation.

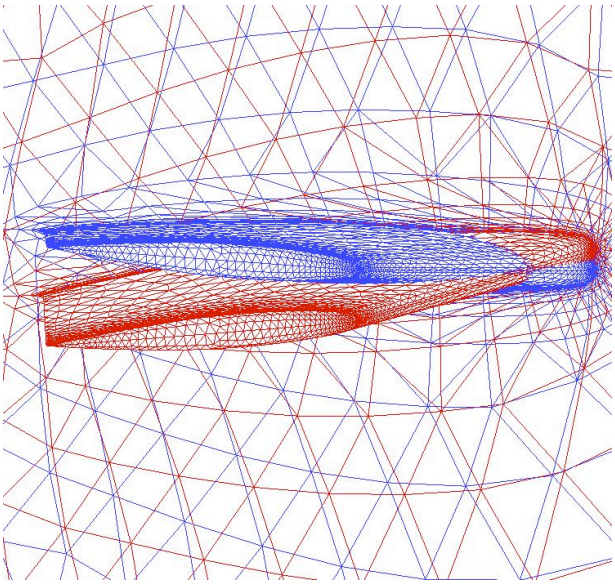

Fig. 3 Wing used for temporal-error verification at minimum and maximum pitch angles; deforming mesh.

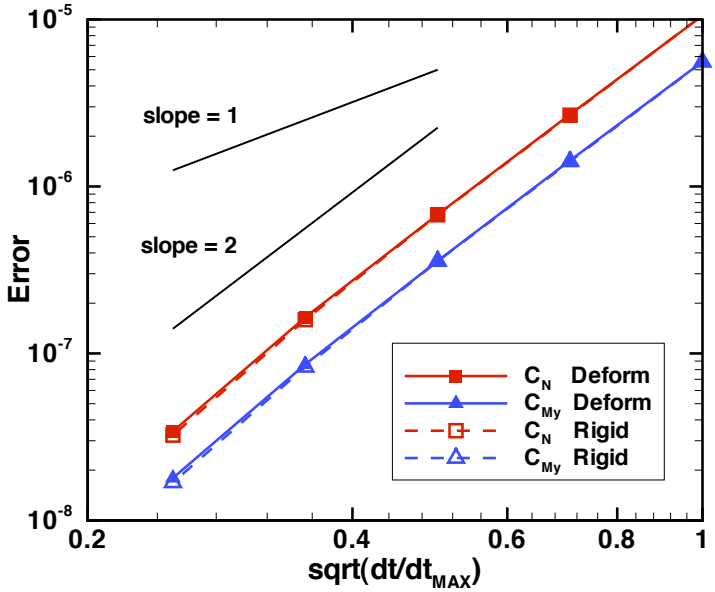

Fig. 4 Error reduction with decreasing time step on rigid and deforming moving meshes; reference slopes for 1st and 2nd order convergence are shown.

Fig. 5 Blended Wing Body model mounted in NASA-Langley 14- × 22-Foot Subsonic Wind Tunnel.

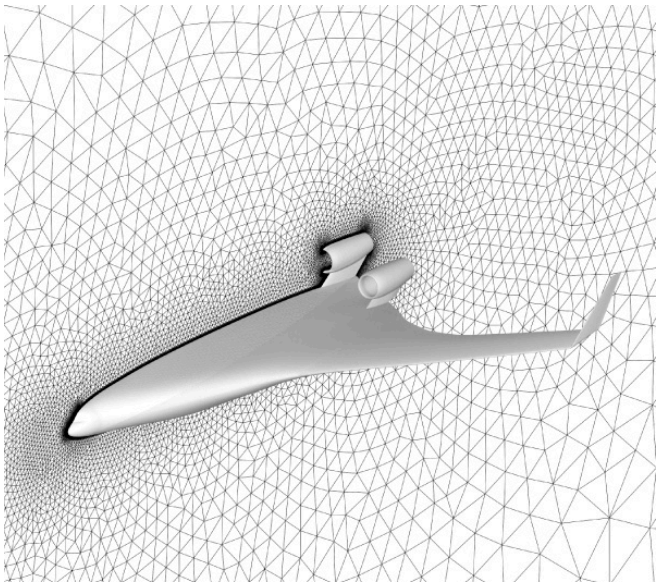

Fig. 6 Symmetry plane of grid used for dynamic Blended Wing Body calculations

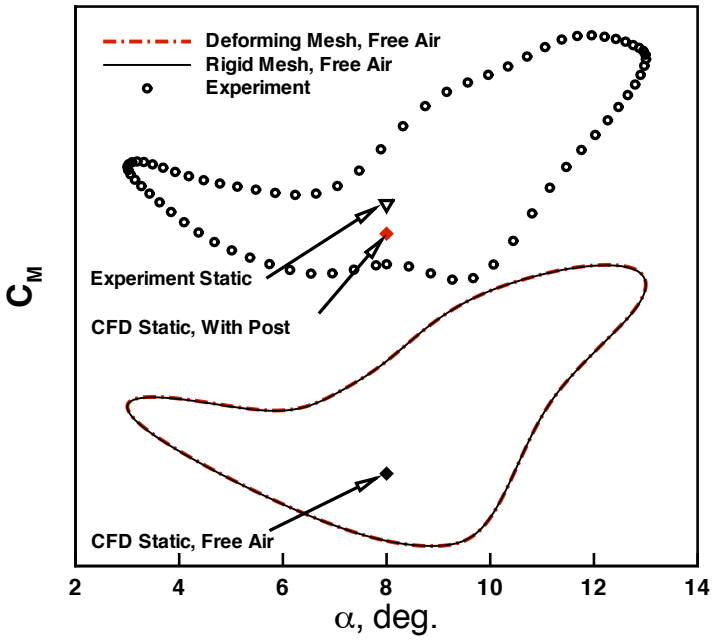

Fig. 7 Comparison of measured and predicted pitching-moment coefficient variation during forced pitch oscillation of a Blended Wing Body model using rigid and deforming meshes.

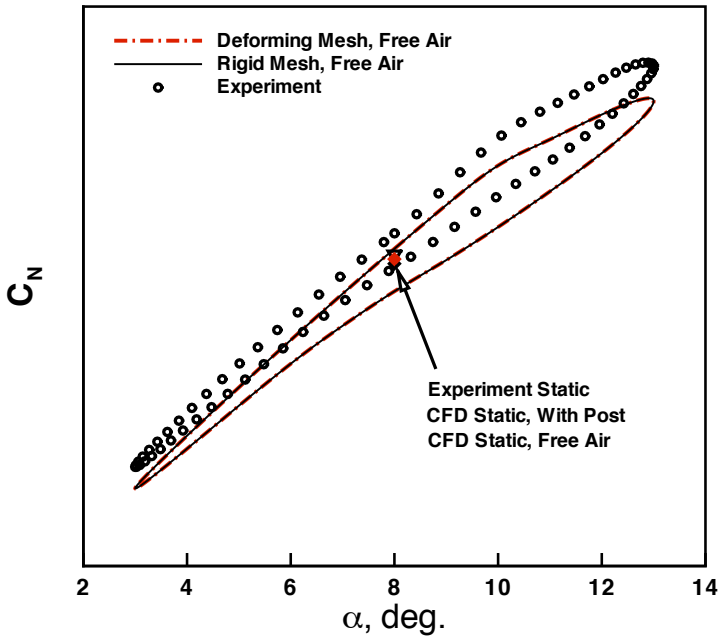

Fig. 8 Comparison of measured and predicted normal force coefficient variation during forced pitch oscillation of a Blended Wing Body model using rigid and deforming meshes.

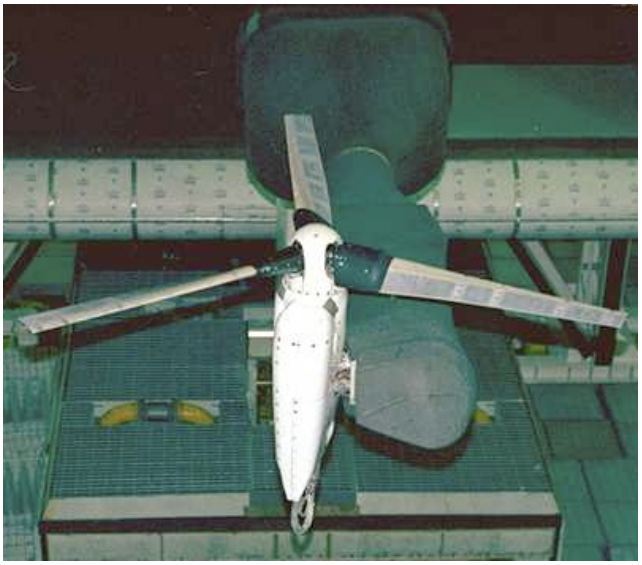

Fig. 9 Tilt Rotor Aeroacoustic Model mounted in the Duits-Nederlandse Windtunnel Large Low-speed Facility; helicopter mode.

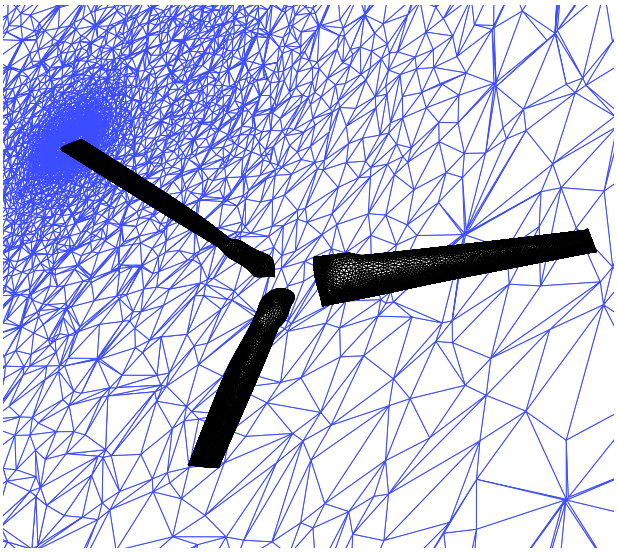

Fig. 10 Mesh used for the isolated TRAM rotor; surface grid in black, slice through field grid near tip of rotor in blue.

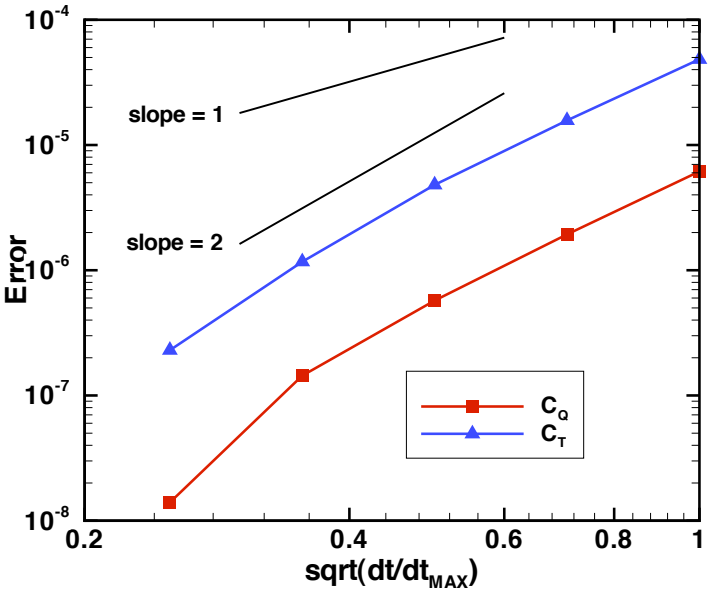

Fig. 11 Error reduction (relative to the noninertial formulation) in torque and thrust with decreasing time-step size for the isolated TRAM rotor; reference slopes for 1st and 2nd order convergence are shown.

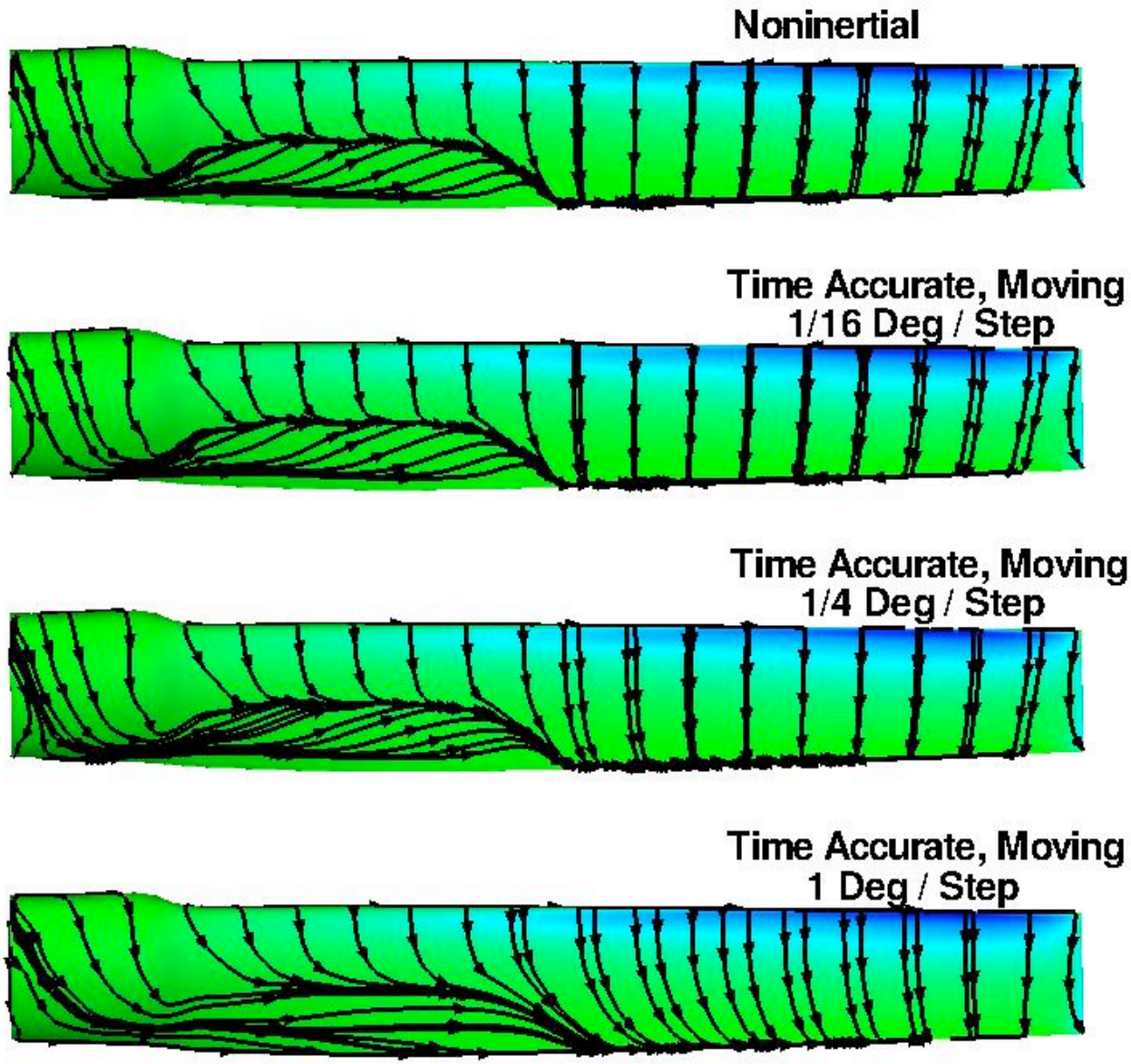

Fig. 12 Convergence, with decreasing time step, of surface-restricted streamlines from the time-accurate, moving grid formulation toward streamlines from the noninertial formulation. Streamlines are as viewed by an observer in a reference frame rotating with the blades; color contours indicate pressure.

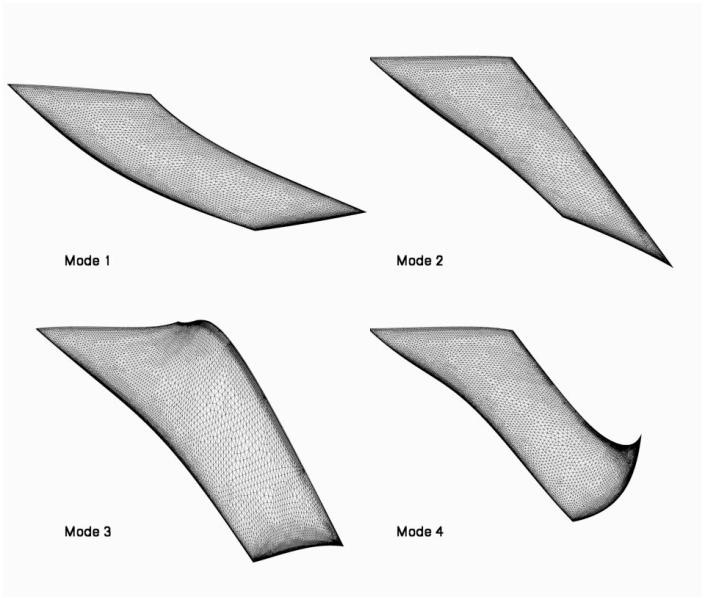

Fig. 13 First four natural mode shapes for AGARD 445.6 wing.

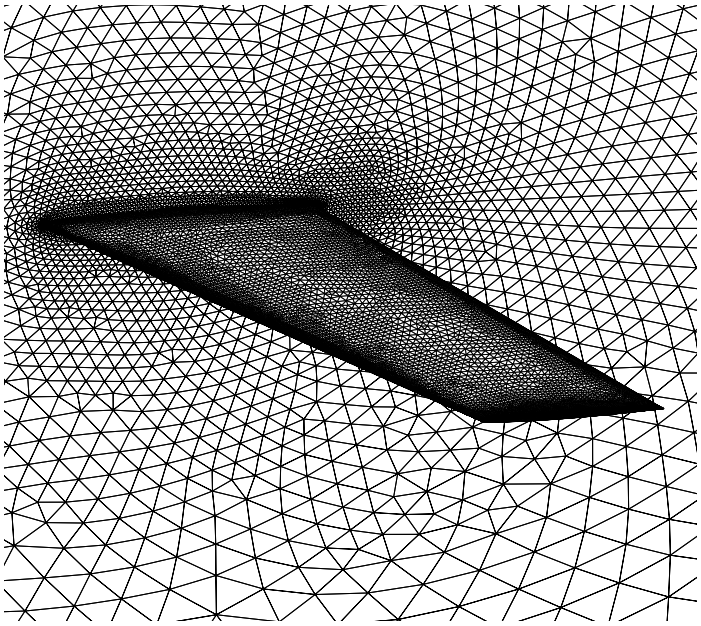

Fig. 14 Mesh used for AGARD 445.6 wing aeroelastic calculations.

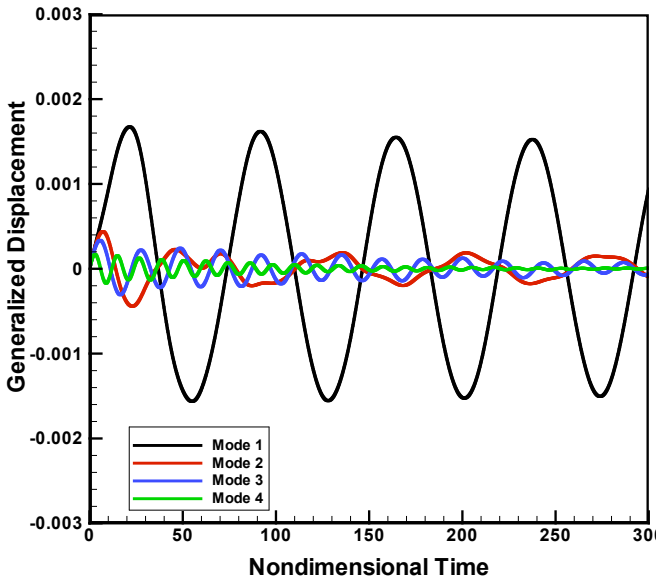

Fig. 15 Aeroelastic response at $\mathrm{M} = 0.9$, $\mathrm{q}_\infty = 75.0$ psf.

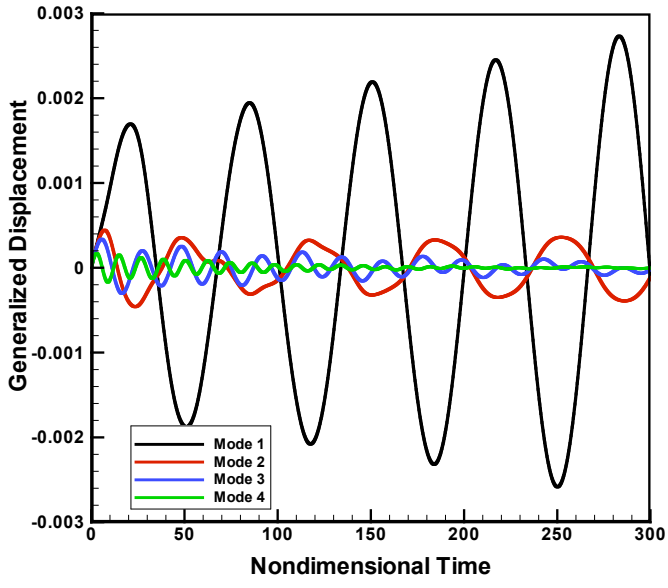

Fig. 16 Aeroelastic response at $\mathrm{M} = 0.9$, $\mathrm{q}_\infty = 89.3$ psf.

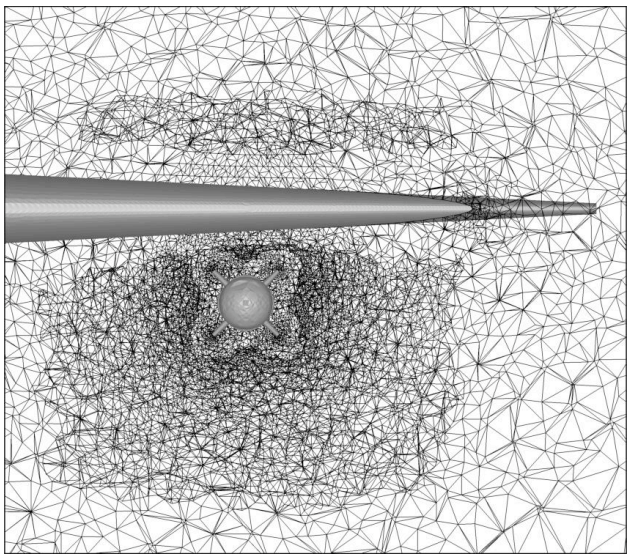

Fig. 17 Wing-store configuration at t = 0, with slice through overset mesh.

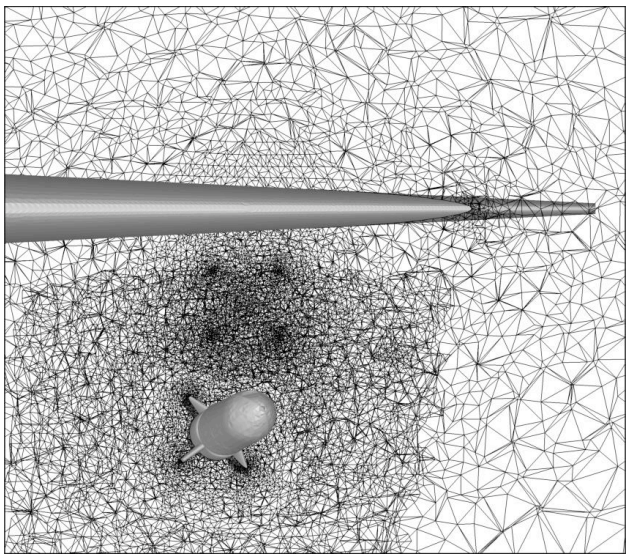

Fig. 18 Wing-store configuration at t = 0.3 seconds, with slice through overset mesh.

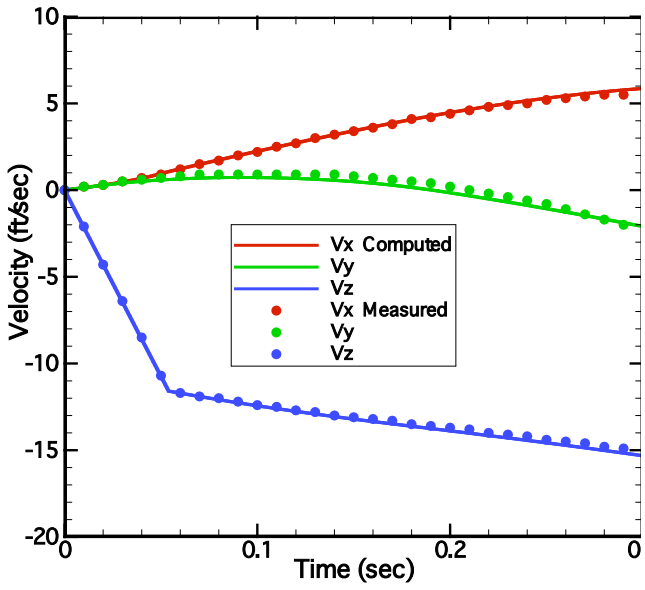

Fig. 19 Computed and measured velocity components of the store center-of-gravity.

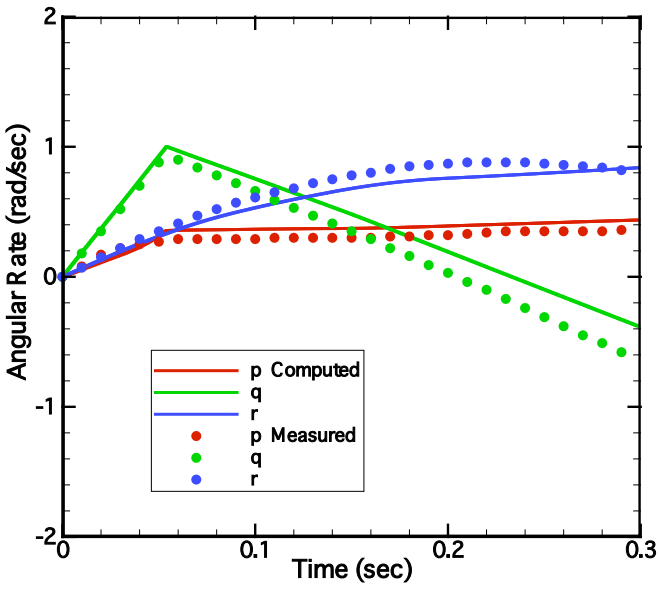

Fig. 20 Computed and measured angular rates of the store body-axis system.