51st AIAA Aerospace Sciences Meeting including the New Horizons Forum and Aerospace Exposition
07 - 10 January 2013, Grapevine (Dallas/Ft. Worth Region), Texas

AIAA 2013-0662

Geometry Optimization in Three-Dimensional Unsteady Flow Problems using the Discrete Adjoint

Karthik Mani *and Dimitri J. Mavriplis †

Department of Mechanical Engineering, University of Wyoming, Laramie, Wyoming 82071-3295

The unsteady discrete adjoint equations are utilized to construct objective function gradients for use in geometry optimization in large scale three-dimensional unsteady flow problems. An existing massively parallel unstructured Reynolds-averaged Navier-Stokes equations solver forms the basis for the framework used to solve both the primal and dual (adjoint) problems. The framework is also capable of handling dynamic mesh deformation using the linear elasticity model. Turbulence modeling options using the one-equation Spalart-Allmaras model and the two-equation k-ω model are available. The adjoint equations are formulated such that all aspects of the primal problem are exactly linearized and no assumptions such as freezing certain components are made. This leads to discretely exact gradients which can be used for optimization purposes. The adjoint-based gradients are verified against those computed by the complex-step method and agreement to machine precision is observed. An agglomeration multigrid solution strategy is employed to accelerate the convergence of the primal problem while GMRES preconditioned by linear agglomeration multigrid is used to solve the dual problem. The method is applied to minimizing the time-integrated torque coefficient of the HART2 rotor in hover while constraining the time-integrated thrust coefficient to the baseline value.

I. Introduction

In the recent past, the use of adjoint equations has become a popular approach for solving aerodynamic design optimization problems based on computational fluid dynamics (CFD).[1–6] Adjoint equations are a very powerful tool in the sense that they allow the computation of sensitivity derivatives of an objective function to a set of given inputs at a cost which is essentially independent of the number of inputs. This is in contrast to the brute-force finite-difference method, where each input or design variable has to be perturbed individually to obtain a corresponding effect on the output. This is a tedious and costly process which is of little use when there a large number of design variables or inputs. Another major shortcoming of the finite-difference method is that it suffers from step-size limitations which affect the accuracy of the computed gradients.

While the use of adjoint equations is now fairly well established in steady-state shape optimization, only recently have inroads been made into extending them to unsteady flow problems. Unsteady discrete adjoint-based shape optimization was initially demonstrated in the context of two-dimensional problems by Mani and Mavriplis[7] and also by Rumpfkeil and Zingg.[8] Preliminary demonstration of the method's feasibility in three-dimensional problems was done by Mavriplis.[9] Full implementation in a general sense and application to large scale problems involving helicopter rotors was then carried out by Nielsen et.al. in the NASA FUN3D code.[10,11]

Unsteady solutions are inherently expensive and this poses serious challenges for their use in the design process. Complementary methods, such as frequency domain techniques mitigate the expensive nature of unsteady problems but are generally limited to problems that exhibit strong periodicity. The prevalence of arbitrary unsteady flow phenomena in engineering problems of interest necessitates the development of methods that operate directly in the time domain. Improvements in computer performance and massive parallelization on compute clusters have dramatically reduced the time required for unsteady solutions to

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

the point where their use as a design tool is gradually becoming commonplace. Large scale unsteady problems such as rotorcraft aerodynamics, aeroelastic flutter and maneuvering aircraft are few examples of successful applications of unsteady CFD methods. Given these advances, it is only natural that adjoint methods be extended to such unsteady applications for shape optimization. However, serious challenges remain to be addressed for large-scale problems. One of the primary challenges in unsteady adjoint methods is the efficient storage of (and I/O operations on) the solution time history. Computing the unsteady adjoint involves a backward integration in time and requires the storage of the entire solution history. For large three-dimensional problems this may result in hundreds of gigabytes of solution data being stored and operated on. It becomes essential to solve unsteady adjoint problems on parallel compute clusters with data being written locally at each node. Achieving efficient parallel scalability in both unsteady analysis and adjoint problems is another crucial factor in developing solvers for optimization purposes. Generally, optimization algorithms require anywhere between 10 and 25 optimization cycles to converge to a reasonable optimum. Each optimization cycle consists of approximately one forward time integration for the analysis problem and one backward integration in time for the adjoint gradient. These factors combined with the expensive nature of each unsteady solution necessitates massive parallelism to reduce overall cycle times. The goal of this paper is to present an efficient parallel scalable framework for computing adjoint-based gradients in large-scale three-dimensional unsteady flow problems and to apply these gradients to design optimization in realistic scenarios.

II. Analysis Problem Formulation

A. Governing Equations of the Flow Problem in ALE Form

The conservative form of the Navier-Stokes equations are used in solving the flow problem. These may be written as:

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

Applying the divergence theorem and integrating over a moving control volume $V(t)$ yields:

$$\int_{V(t)} \frac{\partial \mathbf{U}}{\partial t} dV + \int_{dB(t)} \mathbf{F}(\mathbf{U}) \cdot \mathbf{n} dB = 0 \tag{2}$$

Using the differential identity:

$$\frac{\partial}{\partial t} \int_{V(t)} \mathbf{U} dV = \int_{V(t)} \frac{\partial \mathbf{U}}{\partial t} dV + \int_{dB(t)} (\dot{\mathbf{x}} \cdot \mathbf{n}) dB \tag{3}$$

equation (2) is rewritten as:

$$\frac{\partial}{\partial t} \int_{V(t)} \mathbf{U} dV + \int_{dB(t)} [\mathbf{F}(\mathbf{U}) - \dot{\mathbf{x}}\mathbf{U}] \cdot \mathbf{n} dB = 0 \tag{4}$$

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

$$\frac{\partial V\mathbf{U}}{\partial t} + \int_{dB(t)} [\mathbf{F}(\mathbf{U}) - \dot{\mathbf{x}}\mathbf{U}] \cdot \mathbf{n} dB = 0 \tag{5}$$

This is the Arbitrary-Lagrangian-Eulerian (ALE) finite-volume form of the Euler equations. The equations are required in ALE form since the problem involves deforming meshes where mesh elements change in shape and size at each time-step. Here V refers to the area of the control volume, $\dot{\mathbf{x}}$ is the vector of mesh face or edge velocities, and $\mathbf{n}$ is the unit normal of the face or edge. The state vector $\mathbf{U}$ consists of the conserved variables and the cartesian flux vector $\mathbf{F} = (\mathbf{F}_x, \mathbf{F}_y, \mathbf{F}_z)$ contains both inviscid and viscous fluxes. The equations are closed with the perfect gas equation of state and an appropriate turbulent viscosity model.

B. Temporal Discretization

The time derivative term is discretized using either a first-order accurate backward-difference-formula (BDF1), or a second-order accurate BDF2 scheme. These discretizations are shown in equations (6) and (7) respectively. The index n is used to indicate the current time-level as the convention throughout the paper. The discretization of the BDF2 scheme shown in equation (7) is based on a uniform time-step size.

$$\frac{\partial V\mathbf{U}}{\partial t} = \frac{V^n\mathbf{U}^n - V^{n-1}\mathbf{U}^{n-1}}{\Delta t} \tag{6}$$

$$\frac{\partial V\mathbf{U}}{\partial t} = \frac{\frac{3}{2}V^n\mathbf{U}^n - 2V^{n-1}\mathbf{U}^{n-1} + \frac{1}{2}V^{n-2}\mathbf{U}^{n-2}}{\Delta t} \tag{7}$$

The second-order accurate discretization is used exclusively for the results presented in the paper.

C. Spatial discretization

The solver uses a vertex-centered median dual control volume formulation that is second-order accurate, where the inviscid flux integral S around a closed control volume is discretized as:

$$S = \int_{dB(t)} [\mathbf{F}(\mathbf{U}) - \dot{\mathbf{x}}\mathbf{U}] \cdot \mathbf{n} dB = \sum_{i=1}^{n_{edge}} \mathbf{F}^{\perp}_{e_i}(V_{e_i}, \mathbf{U}, \mathbf{n}_{ei})B_{e_i} \tag{8}$$

where B_e is the face area, V_e is the normal face velocity, $\mathbf{n}_e$ is the unit normal of the face, and $F^{\perp}_e$ is the normal flux across the face. The normal flux across the face is computed using the second-order accurate matrix dissipation scheme[12] as the sum of a central difference and an artificial dissipation term as shown below,

$$\mathbf{F}^{\perp}_e = \frac{1}{2}\left\{\mathbf{F}^{\perp}_L(\mathbf{U}_L, V_e, \mathbf{n}_e) + \mathbf{F}^{\perp}_R(\mathbf{U}_R, V_e, \mathbf{n}_e) + \kappa^{(4)}[T]|[\lambda]|[T]^{-1}\left\{(\nabla^2\mathbf{U})_L - (\nabla^2\mathbf{U})_R\right\}\right\} \tag{9}$$

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

$$(\nabla^2\mathbf{U})_i = \sum_{k=1}^{neighbors} (\mathbf{U}_k - \mathbf{U}_i) \tag{10}$$

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

$$\mathbf{F}^{\perp} = \begin{pmatrix} \rho(V^{\perp} - V_e) \\ \rho(V^{\perp} - V_e)u + \hat{n}_x p \\ \rho(V^{\perp} - V_e)v + \hat{n}_y p \\ \rho(V^{\perp} - V_e)w + \hat{n}_z p \\ E_t(V^{\perp} - V_e) + pV^{\perp} \end{pmatrix} \tag{11}$$

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

D. Mesh Deformation Strategy

In order to deform the mesh in response to design shape changes, a spring analogy and a linear elastic analogy mesh deformation approach have been implemented. Even for small design changes, the spring analogy has been found to result in invalid meshes with negative volumes, particularly for viscous meshes with high stretching in near wall regions. The linear elasticity approach has proven to be much more robust. In this approach, the mesh is modeled as a linear elastic solid with a variable modulus of elasticity that can be prescribed either as inversely proportional to cell volume or to the distance of each cell from the nearest wall.[13] The resulting equations are discretized and solved on the mesh in its original undeformed configuration in response to surface displacements generated by design shape changes using a line-implicit multigrid algorithm analogous to that used for the flow equations. The adjoint of the linear elastic mesh motion equations is also implemented, by transposing the stiffness matrix of the mesh motion formulation and is solved in a similar manner. Figure (1) illustrates a typical convergence history for both the analysis and adjoint version of the linear elastic mesh motion problem applied to a 2.32 million point mesh about the four-bladed HART2 rotor.

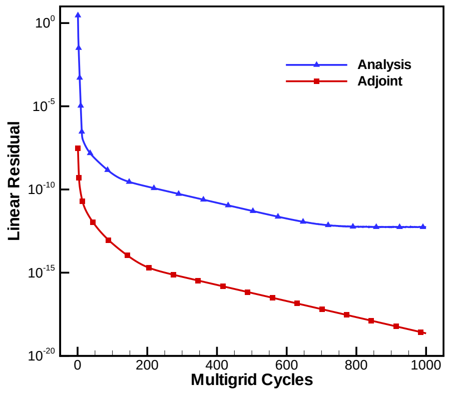


Figure 1. Convergence of the linear elasticity-based mesh deformation equations in both analysis and adjoint modes. This particular example is for shape changes on the 4-bladed HART2 rotor discretized using 2.32 million mesh points.

E. The Discrete Geometric Conservation Law (GCL)

The discrete geometric conservation law (GCL) requires that a uniform flow field be preserved when equation (5) is integrated in time. In other words the deformation of the computational mesh should not introduce conservation errors in the solution of the flow problem. This translates into $\mathbf{U} = constant$ being an exact solution of equation (5). For a conservative scheme, the integral of the fluxes around a closed volume goes to zero when $\mathbf{U} = constant$. Applying these conditions to equation (5) results in the mathematical description of the GCL as stated below.

$$\frac{\partial V}{\partial t} - \int_{dB(t)} \dot{\mathbf{x}} \cdot \mathbf{n} dB = 0 \tag{12}$$

Equation (12) implies that the change in volume of a control volume should be discretely equal to the volume swept by the boundary of the control volume. The vector of cartesian face velocities $\dot{x}$ for each of the faces encompassing the control volume must therefore be chosen such that equation (12) is satisfied. The inner product of the cartesian face velocities and the face normal vector is represented by the term $\mathbf{V_e}$. There are several methods to compute the face velocities while satisfying the GCL, but in previous work it has been shown that satisfaction of the GCL is not a sufficient condition to achieve the underlying accuracy of the chosen time-integration scheme.[14,15] The GCL has been specifically derived for first, second and third-order backward difference (BDF) time-integration schemes and also for the fourth-order accurate IRK64 implicit Runge-Kutta scheme.[15] The form of the GCL described in Ref.[15] is used exclusively in this work.

III. Unsteady Adjoint Formulation

The basic adjoint implementation follows the strategy developed in references.[9,16] Consider an arbitrary objective function L that is evaluated using the unsteady flow solution set $\mathbf{U}$ and unsteady mesh solution set $\mathbf{x}$ expressed as:

$$L = L(\mathbf{U}, \mathbf{x}) \tag{13}$$

Assuming that the state variables (i.e.$\mathbf{U}, \mathbf{x}$) are dependent on some input design parameters $\mathbf{D}$, the total sensitivity of the objective function L to the set of design inputs can expressed as the inner product between the vector of state sensitivities to design inputs and the vector of objective sensitivities to the state variables as:

$$\frac{dL}{d\mathbf{D}}^T = \begin{bmatrix} \frac{\partial \mathbf{U}}{\partial \mathbf{D}}^T & \frac{\partial \mathbf{x}}{\partial \mathbf{D}}^T \end{bmatrix} \begin{bmatrix} \frac{\partial L}{\partial \mathbf{U}}^T \\ \frac{\partial L}{\partial \mathbf{x}}^T \end{bmatrix} \tag{14}$$

where the equation has been transposed to enable the derivation of the adjoint equations. The non-linear flow residual operator and the linear elasticity mesh residual operator as earlier provide the constraints to the optimization and can be expressed in general form over the whole space and time domains as:

$$\mathbf{R}(\mathbf{U}, \mathbf{x}) = 0 \tag{15}$$
$$\mathbf{G}(\mathbf{x}, \mathbf{x}_{surf}) = 0 \tag{16}$$

which when linearized with respect to the design inputs yields:

$$\frac{\partial \mathbf{R}}{\partial \mathbf{U}}\frac{\partial \mathbf{U}}{\partial \mathbf{D}} + \frac{\partial \mathbf{R}}{\partial \mathbf{x}}\frac{\partial \mathbf{x}}{\partial \mathbf{D}} = 0 \tag{17}$$
$$\frac{\partial \mathbf{G}}{\partial \mathbf{x}}\frac{\partial \mathbf{x}}{\partial \mathbf{D}} + \frac{\partial \mathbf{G}}{\partial \mathbf{x}_{surf}}\frac{\partial \mathbf{x}_{surf}}{\partial \mathbf{D}} = 0 \tag{18}$$

When combined and expressed in matrix form becomes:

$$\begin{bmatrix} \frac{\partial \mathbf{R}}{\partial \mathbf{U}} & \frac{\partial \mathbf{R}}{\partial \mathbf{x}} \\ 0 & \frac{\partial \mathbf{G}}{\partial \mathbf{x}} \end{bmatrix} \begin{bmatrix} \frac{\partial \mathbf{U}}{\partial \mathbf{D}} \\ \frac{\partial \mathbf{x}}{\partial \mathbf{D}} \end{bmatrix} = \begin{bmatrix} 0 \\ -\frac{\partial \mathbf{G}}{\partial \mathbf{x}_{surf}}\frac{\partial \mathbf{x}_{surf}}{\partial \mathbf{D}} \end{bmatrix} \tag{19}$$

Transposing and rearranging the above equation provides an expression for the state sensitivities that are unknown in equation (14) as:

$$\begin{bmatrix} \frac{\partial \mathbf{U}}{\partial \mathbf{D}}^T & \frac{\partial \mathbf{x}}{\partial \mathbf{D}}^T \end{bmatrix} = \begin{bmatrix} 0 & -\frac{\partial \mathbf{x}_{surf}}{\partial \mathbf{D}}^T \frac{\partial \mathbf{G}}{\partial \mathbf{x}_{surf}}^T \end{bmatrix} \begin{bmatrix} \frac{\partial \mathbf{R}}{\partial \mathbf{U}}^T & 0 \\ \frac{\partial \mathbf{R}}{\partial \mathbf{x}}^T & \frac{\partial \mathbf{G}}{\partial \mathbf{x}}^T \end{bmatrix}^{-1} \tag{20}$$

Substituting the expression for the state sensitivities back into equation (14), a system of adjoint equations can be now defined as:

$$\begin{bmatrix} \frac{\partial \mathbf{R}}{\partial \mathbf{U}}^T & 0 \\ \frac{\partial \mathbf{R}}{\partial \mathbf{x}}^T & \frac{\partial \mathbf{G}}{\partial \mathbf{x}}^T \end{bmatrix} \begin{bmatrix} \Lambda_{\mathbf{U}} \\ \Lambda_{\mathbf{x}} \end{bmatrix} = \begin{bmatrix} \frac{\partial L}{\partial \mathbf{U}}^T \\ \frac{\partial L}{\partial \mathbf{x}}^T \end{bmatrix} \tag{21}$$

where $\Lambda_{\mathbf{U}}$ and $\Lambda_{\mathbf{x}}$ are the flow and mesh adjoint variables respectively. Noting that the linearization of the mesh residual with respect to the mesh coordinates $\mathbf{x}$ at an arbitrary time level n is simply the constant stiffness matrix $[\mathbf{K}]$, and taking into account the functional dependencies of the flow residual operator $\mathbf{R}$ from equation (30), the adjoint system can be expanded discretely in time as (shown for a time domain with three steps):

$$\left[\begin{array}{ccc|ccc}
\frac{\partial \mathbf{R}^{n-2}}{\partial \mathbf{U}^{n-2}}^T & \frac{\partial \mathbf{R}^{n-1}}{\partial \mathbf{U}^{n-2}}^T & \frac{\partial \mathbf{R}^{n}}{\partial \mathbf{U}^{n-2}}^T & 0 & 0 & 0 \\
0 & \frac{\partial \mathbf{R}^{n-1}}{\partial \mathbf{U}^{n-1}}^T & \frac{\partial \mathbf{R}^{n}}{\partial \mathbf{U}^{n-1}}^T & 0 & 0 & 0 \\
0 & 0 & \frac{\partial \mathbf{R}^{n}}{\partial \mathbf{U}^{n}}^T & 0 & 0 & 0 \\
\hline
\frac{\partial \mathbf{R}^{n-2}}{\partial \mathbf{x}^{n-2}}^T & \frac{\partial \mathbf{R}^{n-1}}{\partial \mathbf{x}^{n-2}}^T & \frac{\partial \mathbf{R}^{n}}{\partial \mathbf{x}^{n-2}}^T & [\mathbf{K}]^T & 0 & 0 \\
0 & \frac{\partial \mathbf{R}^{n-1}}{\partial \mathbf{x}^{n-1}}^T & \frac{\partial \mathbf{R}^{n}}{\partial \mathbf{x}^{n-1}}^T & 0 & [\mathbf{K}]^T & 0 \\
0 & 0 & \frac{\partial \mathbf{R}^{n}}{\partial \mathbf{x}^{n}}^T & 0 & 0 & [\mathbf{K}]^T
\end{array}\right]
\left[\begin{array}{c}
\Lambda_{\mathbf{U}}^{n-2} \\ \Lambda_{\mathbf{U}}^{n-1} \\ \Lambda_{\mathbf{U}}^{n} \\ \hline \Lambda_{\mathbf{x}}^{n-2} \\ \Lambda_{\mathbf{x}}^{n-1} \\ \Lambda_{\mathbf{x}}^{n}
\end{array}\right]
=
\left[\begin{array}{c}
\frac{\partial L}{\partial \mathbf{U}^{n-2}}^T \\ \frac{\partial L}{\partial \mathbf{U}^{n-1}}^T \\ \frac{\partial L}{\partial \mathbf{U}^{n}}^T \\ \hline \frac{\partial L}{\partial \mathbf{x}^{n-2}}^T \\ \frac{\partial L}{\partial \mathbf{x}^{n-1}}^T \\ \frac{\partial L}{\partial \mathbf{x}^{n}}^T
\end{array}\right] \tag{22}$$

which when rearranged through row and column swapping results in an upper triangular system of linear equations as:

$$\left[\begin{array}{cc|cc|cc}
[\mathbf{K}]^T & \frac{\partial \mathbf{R}^{n-2}}{\partial \mathbf{x}^{n-2}}^T & 0 & \frac{\partial \mathbf{R}^{n-1}}{\partial \mathbf{x}^{n-2}}^T & 0 & \frac{\partial \mathbf{R}^{n}}{\partial \mathbf{x}^{n-2}}^T \\
0 & \frac{\partial \mathbf{R}^{n-2}}{\partial \mathbf{U}^{n-2}}^T & 0 & \frac{\partial \mathbf{R}^{n-1}}{\partial \mathbf{U}^{n-2}}^T & 0 & \frac{\partial \mathbf{R}^{n}}{\partial \mathbf{U}^{n-2}}^T \\
\hline
0 & 0 & [\mathbf{K}]^T & \frac{\partial \mathbf{R}^{n-1}}{\partial \mathbf{x}^{n-1}}^T & 0 & \frac{\partial \mathbf{R}^{n}}{\partial \mathbf{x}^{n-1}}^T \\
0 & 0 & 0 & \frac{\partial \mathbf{R}^{n-1}}{\partial \mathbf{U}^{n-1}}^T & 0 & \frac{\partial \mathbf{R}^{n}}{\partial \mathbf{U}^{n-1}}^T \\
\hline
0 & 0 & 0 & 0 & [\mathbf{K}]^T & \frac{\partial \mathbf{R}^{n}}{\partial \mathbf{x}^{n}}^T \\
0 & 0 & 0 & 0 & 0 & \frac{\partial \mathbf{R}^{n}}{\partial \mathbf{U}^{n}}^T
\end{array}\right]
\left[\begin{array}{c}
\Lambda_{\mathbf{x}}^{n-2} \\ \Lambda_{\mathbf{U}}^{n-2} \\ \hline \Lambda_{\mathbf{x}}^{n-1} \\ \Lambda_{\mathbf{U}}^{n-1} \\ \hline \Lambda_{\mathbf{x}}^{n} \\ \Lambda_{\mathbf{U}}^{n}
\end{array}\right]
=
\left[\begin{array}{c}
\frac{\partial L}{\partial \mathbf{x}^{n-2}}^T \\ \frac{\partial L}{\partial \mathbf{U}^{n-2}}^T \\ \hline \frac{\partial L}{\partial \mathbf{x}^{n-1}}^T \\ \frac{\partial L}{\partial \mathbf{U}^{n-1}}^T \\ \hline \frac{\partial L}{\partial \mathbf{x}^{n}}^T \\ \frac{\partial L}{\partial \mathbf{U}^{n}}^T
\end{array}\right] \tag{23}$$

The system being upper triangular in nature, the solution to the adjoint equations can be obtained through back-substitution, starting at the final time level n and sweeping back in time. At any arbitrary time-step n (except the final two time-steps), the linear flow and mesh adjoint equations take up the form shown below.

$$\left[\frac{\partial \mathbf{R}^{n}}{\partial \mathbf{U}^{n}}\right]^T \Lambda_{\mathbf{U}}^{n} = \frac{\partial L}{\partial \mathbf{U}^{n}}^T - \frac{\partial \mathbf{R}^{n+1}}{\partial \mathbf{U}^{n}}^T \Lambda_{\mathbf{U}}^{n+1} - \frac{\partial \mathbf{R}^{n+2}}{\partial \mathbf{U}^{n}}^T \Lambda_{\mathbf{U}}^{n+2} \tag{24}$$

$$[\mathbf{K}]^T \Lambda_{\mathbf{x}}^{n} = \frac{\partial L}{\partial \mathbf{x}^{n}}^T - \frac{\partial \mathbf{R}^{n}}{\partial \mathbf{x}^{n}}^T \Lambda_{\mathbf{U}}^{n+2} - \frac{\partial \mathbf{R}^{n+1}}{\partial \mathbf{x}^{n}}^T \Lambda_{\mathbf{U}}^{n+1} - \frac{\partial \mathbf{R}^{n+2}}{\partial \mathbf{x}^{n}}^T \Lambda_{\mathbf{U}}^{n+2} \tag{25}$$

Once the vector of adjoint variables is known, the total sensitivity can be determined as:

$$\frac{dL}{d\mathbf{D}}^T = \left[\begin{array}{cc} 0 & -\left(\frac{\partial \mathbf{x}_{surf}}{\partial \mathbf{D}}^T \frac{\partial \mathbf{G}}{\partial \mathbf{x}_{surf}}^T\right)\end{array}\right]\left[\begin{array}{c}\Lambda_{\mathbf{U}} \\ \Lambda_{\mathbf{x}}\end{array}\right] = -\frac{\partial \mathbf{x}_{surf}}{\partial \mathbf{D}}^T [\beta]^T \Lambda_{\mathbf{x}} \tag{26}$$

where $[\beta]$ is the generalized mesh transformation matrix, which transforms the baseline surface mesh coordinates to the appropriate orientation at different time levels. The complete gradient vector is therefore simply a matrix-vector product between the sensitivity of the mesh coordinates of the deformed geometry to the design inputs and the sum in time of the transformed mesh adjoint variables.

IV. Adjoint Gradient Verification

The gradients computed using the unsteady adjoint method presented above are verified against values obtained through complex variable-based differentiation of the analysis solver. Any function $f(x)$ operating on a real variable x can be utilized to compute the derivative $f'(x)$ by redefining the input variable x and all intermediate variables used in the discrete evaluation of $f(x)$ as a complex variables. For a complex input, the function when redefined as described produces a complex output. The derivative of the real function $f(x)$ can be computed by expanding the complex operator $f(x + ih)$ as:

$$f(x + ih) = f(x) + ihf'(x) + \cdots \tag{27}$$

from which the derivative $f'(x)$ can be easily determined as:

$$f'(x) = \frac{Im\left[f\left(x + ih\right)\right]}{h} \tag{28}$$

As in the case of finite-differencing, the complex step-based differentiation also requires a step size. However, unlike finite-differencing the complex step method is insensitive to small step sizes since no differencing is required. In theory it is possible to verify adjoint-based gradients using the complex step method to machine precision. Verification of the adjoint implementation is a two fold process and is performed as follows. The forward or tangent linearization of the code is constructed using chain rule-based discrete differentiation of the analysis solver and naturally follows the sequence of discrete operations in the code. Hence, it is possible to incrementally verify each step in the forward linearization using the complex version of the analysis solver. A complex perturbation is introduced at the beginning of the code and derivatives may be extracted at any intermediate step in the sequence of discrete operations in the solution process to compare against derivatives obtained from the forward linearization of the code. Once the forward linearization of the code has been implemented and verified in this manner, it is then discretely transposed to obtain the adjoint linearization. The verification of the adjoint linearization is done by invoking of the principle of duality between transpose operations. Any step in the adjoint linearization can be verified against the corresponding step in the forward linearization using the principle of duality. It is possible to verify the complete adjoint implementation incrementally in this manner.

Table (1) shows the verification results for the adjoint implementation by comparing the unsteady forward and adjoint linearization sensitivities to complex step-based sensitivities. These results are based on a turbulent pitching ONERA M6 wing case integrated over 5 time-steps. A generic functional consisting of the sum of all force components at each time-step and over all 5 time-steps was used while the design variable is a single surface grid point at the intersection of the wing and the symmetry plane close to the leading edge. The mixed element mesh consisted of approximately 83,000 vertices and the verification was done in parallel on 16 processors. The equations were converged to machine precision at each time-step to eliminate any algebraic errors. The verification indicates the expected match up of all digits up to machine precision.

Table 1. Verification of linearization

Method	Sensitivity
Adjoint	0.171936619308811
Tangent	0.171936619308811
Complex	0.171936619308810

V. Analysis and Adjoint Solution Strategy

Denoting the spatially discretized terms at time level n by the operator $S^n(\mathbf{U}^n)$, the resulting system of non-linear equations to be solved for the analysis problem at each time step can be written as (shown for the case of the BDF2 scheme):

$$\mathbf{R}^n = \frac{\frac{3}{2}V^n\mathbf{U}^n - 2V^{n-1}\mathbf{U}^{n-1} - \frac{1}{2}V^{n-2}\mathbf{U}^{n-2}}{\Delta t} + S^n(\mathbf{U}^n) = 0 \tag{29}$$

which in simplified form exhibiting the functional dependencies on $\mathbf{U}$ and $\mathbf{x}$ at different time levels is given as:

$$\mathbf{R}^n(\mathbf{U}^n, \mathbf{U}^{n-1}, \mathbf{U}^{n-2}, \mathbf{x}^n, \mathbf{x}^{n-1}, \mathbf{x}^{n-2}) = 0 \tag{30}$$

The implicit residual is linearized with respect to the unknown solution vector $\mathbf{U}^n$ and solved for using Newton's method as:

$$\begin{aligned} \left[\frac{\partial \mathbf{R}^k}{\partial \mathbf{U}^k}\right]\delta\mathbf{U}^k &= -\mathbf{R}^k \\ \mathbf{U}^{k+1} &= \mathbf{U}^k + \delta\mathbf{U}^k \\ \delta\mathbf{U}^k \rightarrow 0, \mathbf{U}^n &= \mathbf{U}^k \end{aligned} \tag{31}$$

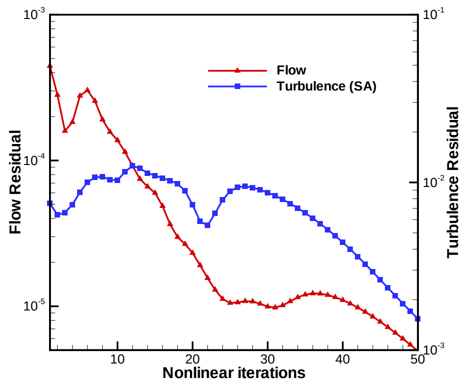
The linearization is approximate since the flow jacobian matrix only consists of the spatially first-order accurate terms. It should be noted that although the flow Jacobian matrix is restricted to first-order terms, it does contain all coupling terms between the flow equations and the turbulence model. At each time step, an agglomeration multigrid algorithm is used to converge the unsteady residual to small values. On each mesh level, a line-implicit block-tridiagonal algorithm is used to further accelerate the convergence of the multigrid algorithm for highly-stretched unstructured meshes in the boundary layer and wake regions.[17] Typical convergence of the flow and turbulence equations at an arbitrary time step is shown in Figure (2(a)). Preconditioned GMRES with line-implicit linear multigrid as the preconditioner may be invoked in order to obtain full convergence to machine precision at each time-step. Preconditioned GMRES only requires the product of the exact flow Jacobian matrix with a vector and does not require the storage of the the exact linearization in memory. Since the code has been linearized in a discretely exact sense for products with arbitrary vectors in both forward and adjoint modes for the computation of sensitivities, it is possible to use the forward linearization infrastructure to compute such a matrix-vector product in the GMRES algorithm for the analysis problem. Figure (2(b)) shows the convergence utilizing the preconditioned GMRES algorithm at an arbitrary time-step for the HART2 rotor in hover on a mesh consisting of 2.32 million grid points. In this example, 100 krylov vectors with no restarts were used for each of the last 9 nonlinear iterations. Each krylov vector was solved for using 3 multigrid cycles and 4 line-implicit smoothing cycles on each level.

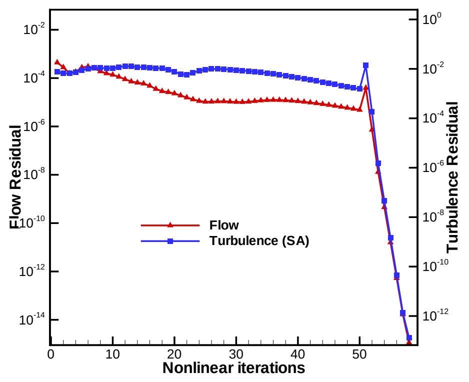
The adjoint equations form a linear system at each time-step and are solved using line-implicit linear multigrid preconditioned GMRES. Figure (3(a)) shows the typical convergence of the adjoint system at an arbitrary time-step using 100 krylov vectors and 2 GMRES restarts. Full convergence of the adjoint system at each time-step can be achieved through repeated restarts or by increasing the number of krylov vectors per restart. It is generally not possible to use a very large number of krylov vectors due to memory limitations and hence more restarts are used. Figure (3(b)) indicates that full convergence to machine precision can generally be achieved using 100 krylov vectors and 30 restarts. For similar levels of convergence, the cost of obtaining analysis and adjoint solutions at a given time-step are similar in terms of wall clock time.

VI. Optimization Example

A. Unsteady Test Problem

The HART2 rotor in hover solved in a time-accurate manner is employed as the test problem for the optimization example. The mixed element mesh made up of prizms, pyramids and tetrahedra consists of approximately 2.32 million grid points and is shown in Figures (4(a)) and (4(b)). The simulation is run for a single revolution using a 2 degree time-step for 180 time-steps starting from freestream initialization. The time-dependent mesh motion is determined by rotating the entire mesh as a solid body at each time step. The unsteady Reynolds-averaged Navier-Stokes equations are solved at each time step in ALE form, using the Spalart-Allmaras turbulence model. Figures (5(a)) and (5(b)) shows a snapshot of the pressure coefficient contours on the rotor at the end of a single revolution.

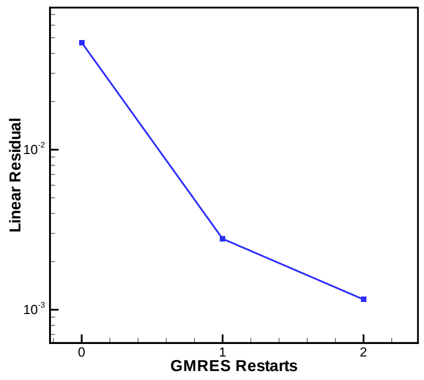


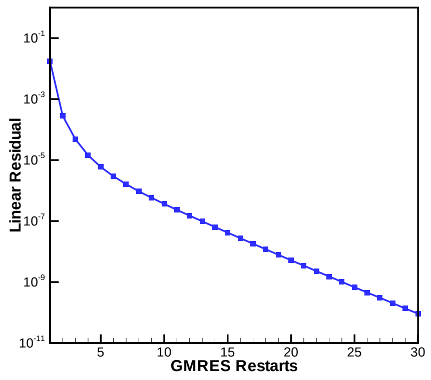


(a) Typical convergence of the flow and turbulence equations at an arbitrary time-step using line implicit linear multigrid.

(b) Typical convergence of the flow and turbulence equations at an arbitrary time-step using line implicit linear multigrid for the first 50 nonlinear iterations and line implicit linear multigrid preconditioned GMRES for the final 9 nonlinear iterations.

Figure 2. Convergence of unsteady flow and turbulence equations.





(a) Typical convergence of adjoint at an arbitrary time-step.

(b) Full convergence of adjoint at an arbitrary time-step.

Figure 3. Convergence of unsteady adjoint equations.

B. Geometry Parameterization

In order to obtain sensitivities with respect to a set of shape parameters that are well suited for design optimization purposes, a baseline blade is constructed by stacking 11 airfoil section along the span. Each airfoil contains 10 Hicks-Henne bump functions 5 on the upper surface and 5 on the lower surface that can be used to modify the airfoil shape. Additionally, the twist values of the blade at the root and tip airfoil sections are also used as design variables resulting in a total of 112 design variables. Figure (6(a)) provides an illustration of the baseline blade design setup. A high density structured mesh is generated about this blade geometry, which is then rotated and translated to match each individual blade in the CFD mesh, as shown in Figure (6(b)). Interpolation patterns between each unstructured mesh surface point and the baseline structured mesh are determined in a preprocessing phase. These interpolation patterns are then used to interpolate shape changes from the baseline blade to all four blades in the CFD mesh (as determined by changes in the design variables) and to transfer sensitivities from the surface CFD mesh points to the design variables using the chain rule of differentiation.

C. Unsteady Objective Function Formulation

A time-integrated objective function based on the time variation of the thrust (C_T) and torque (C_Q) coefficients is used for this test case. The goal of the optimization is to reduce the time-integrated torque coefficient while constraining the time-integrated thrust coefficient to the baseline rotor performance. The objective function is based on the summation of the differences between a target and a computed objective value at each time level n in the final $1/6^{th}$ of the first revolution of the rotor. Mathematically the local objective function at each time-step in the integration range is defined as:

$$L^n = (\delta C_T^n)^2 + 10(\delta C_Q^n)^2 \tag{32}$$

$$\delta C_T^n = (C_T^n - C_{T_{target}}^n) \tag{33}$$

$$\delta C_Q^n = (C_Q^n - C_{Q_{target}}^n) \tag{34}$$

where the target thrust coefficient values at each time-step in the integration range are set from the baseline HART2 rotor values and the target torque values are set to zero. The weight of 10 on the torque coefficient is necessary to equalize the difference in orders-of-magnitude between the thrust and torque coefficients. The global or time-integrated objective is then constructed using equal unit weights at each time-step as:

$$L^g = \sum_{n=120}^{n=180} L^n \tag{35}$$

D. Time-dependent Optimization

The optimization procedure uses the L-BFGS-B bounded reduced Hessian algorithm developed by Nocedal et.al.[18] Each request by the optimization driver for a function and gradient value results in a single forward time-integration of the analysis solver and a single backward integration in time of the adjoint solver. A bound of $\pm 5\%$ of the chord length for each defining airfoil section was set on the Hicks-Henne bump functions, while a bound of $\pm 2^o$ of twist was set on the root and tip twist definitions. The optimization was performed on the MtMoran IBM computer cluster recently installed at the University of Wyoming Advanced Research Computing Center with the analysis/adjoint solver running in parallel on 1024 cores. Each time-step in the analysis problem used 50 nonlinear iterations with each nonlinear iteration requiring 3 linear multigrid cycles, while each time-step in the adjoint solution used 120 krylov vectors with 3 linear multigrid cycles per krylov vector and no restarts in the GMRES algorithm. For the mesh used in the example problem each partition writes out approximately 1.5MB of data at the end of a time-step to local scratch space on each node. The MtMoran computer cluster is configured with 1TB of disk space per node which is shared by 16 processors for scratch files. Overall, the time required for file I/O was negligible compared to the run time of the solver.

A single function/gradient call required approximately 1 hour of wall clock time and the optimization was terminated after 12 wall clock hours. During this time the optimization algorithm requested approximately 10 function/gradient calls and performed 4 design iterations. Approximately 250 GB of data was written to and read from disk during each function/gradient call for this case.

Figures (7(a)) and (7(b)) show the convergence of the functional and the $L2$-norm of the gradient vector during the course of the optimization. Although the functional itself has not be reduced significantly, an order-of-magnitude reduction is observed in the norm of the gradient vector indicating that the optimization is approaching a local minimum. Figures (8(a)) and (8(b)) compare the time variation of the thrust and torque coefficients of the baseline HART2 rotor and the optimized rotor. An approximate reduction of 5% in the torque coefficient within the objective function integration range is observed while an approximate loss of 2% in the thrust coefficient is incurred. The resulting shape changes to the rotor are relatively minor, as shown in Figure (9). The most sensitive design variables are at the tip of the blade where a slight airfoil thinning takes place, while virtually no changes are observed at the sections closer to the root. The twist design variables were found to have low sensitivity values to this objective and were modified very little throughout the optimization procedure.

VII. Concluding Remarks

A time-dependent adjoint capability has been implemented and verified in a production level 3D Reynolds-averaged Navier Stokes solver and used to perform a shape optimization for a rotor in hover. Although the analysis and adjoint capabilities for time-dependent mesh deformation in response to cyclic blade motion has been implemented and verified, the current initial optimization problem that was chosen involved only solid body rotation of the mesh at each time step, and future work will include cases with time-dependent mesh deformation. The analysis problem used for the optimization was devised to provide a realistic optimization setting while requiring manageable computer resources. Thus, a relatively large time step of 2 degrees was used and only one rotor revolution was simulated. One of the principal constraints was the availability of sufficient computer resources, and a reluctance to reduce the level of convergence at each time step in order to save computer time, since this entails the risk of obtaining inaccurate design sensitivities. Future work will focus on demonstrating optimizations for larger problems involving smaller time steps for longer time integration periods and using finer meshes. Additionally, this approach will be extended to combined aero-structural optimizations following our previous work in two dimensions.[19]

VIII. Acknowledgements

This work was partially funded under ONR Grant number N00014-09-1-1060 and NASA Grant NNX10AL54G. Computer resources were provided by the University of Wyoming Advanced Research Computing Center.

References

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

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

[3]Nadarajah, S. and Jameson, A., "A Comparison of the Continuous and Discrete Adjoint Approach to Automatic Aerodynamic Optimization," *Proceedings of the 38th Aerospace Sciences Meeting and Exhibit, Reno NV*, 2000, AIAA Paper 2000–0667.

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

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

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

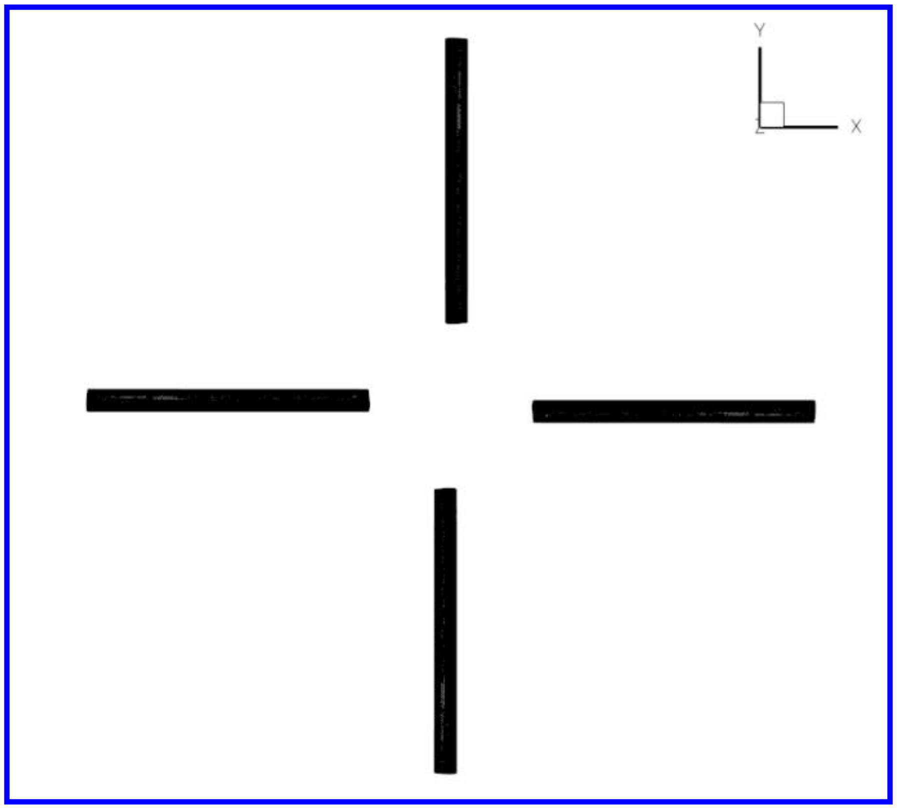
[7]Mani, K. and Mavriplis, D. J., "Unsteady Discrete Adjoint Formulation for Two-Dimensional Flow Problems with Deforming Meshes," *AIAA Journal*, Vol. 46-6, June 2008, pp. 1351–1364.

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

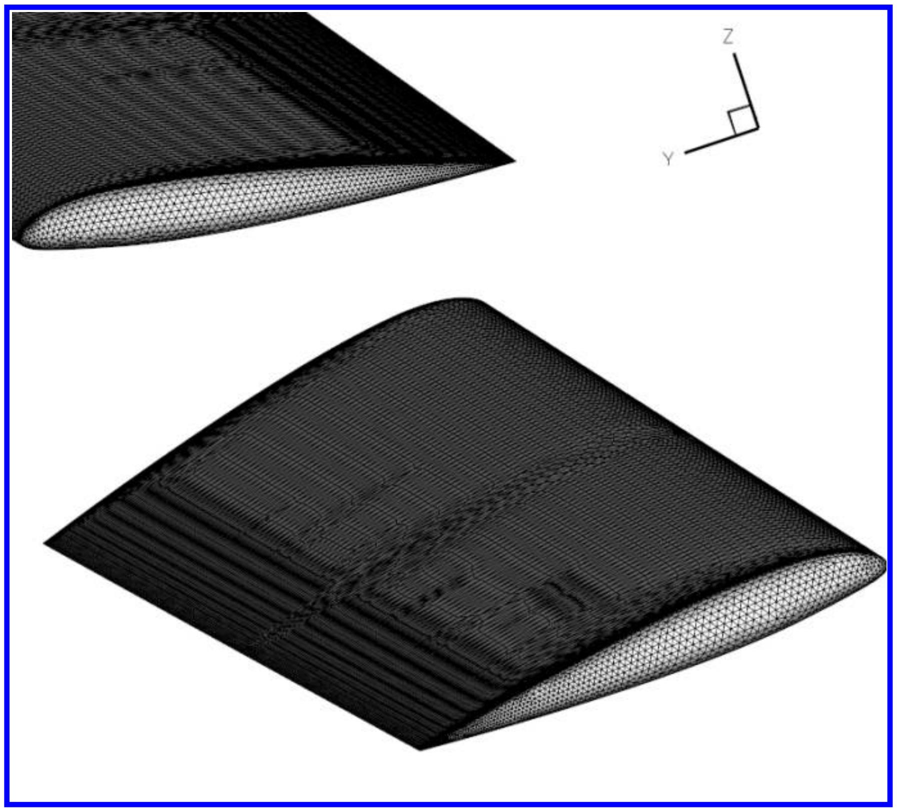
[9]Mavriplis, D. J., "Solution of the Unsteady Discrete Adjoint for Three-Dimensional Problems on Dynamically Deforming Unstructured Meshes," *Proceedings of the 46th Aerospace Sciences Meeting and Exhibit, Reno NV*, 2008, AIAA Paper 2008–0727.

[10]Nielsen, E. J., Diskin, B., and Yamaleev, N., "Discrete Adjoint-Based Design Optimization of Unsteady Turbulent Flows on Dynamic Unstructured Grids," *AIAA Journal*, Vol. 48-6, June 2010, pp. 1195–1206.

[11]Nielsen, E. J., Lee-Rausch, E. M., and Jones, W. T., "Adjoint-Based Design of Rotors in a Noninertial Reference Frame," *Journal of Aircraft*, Vol. 47-2, March-April 2010, pp. 638–646.

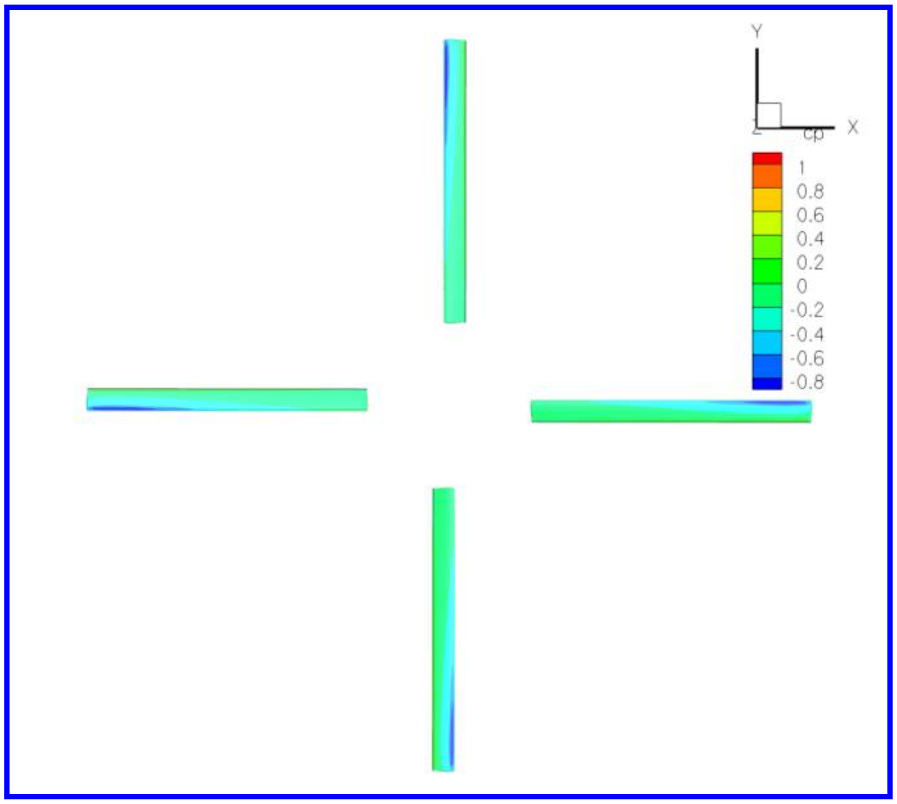


(a) Planform view

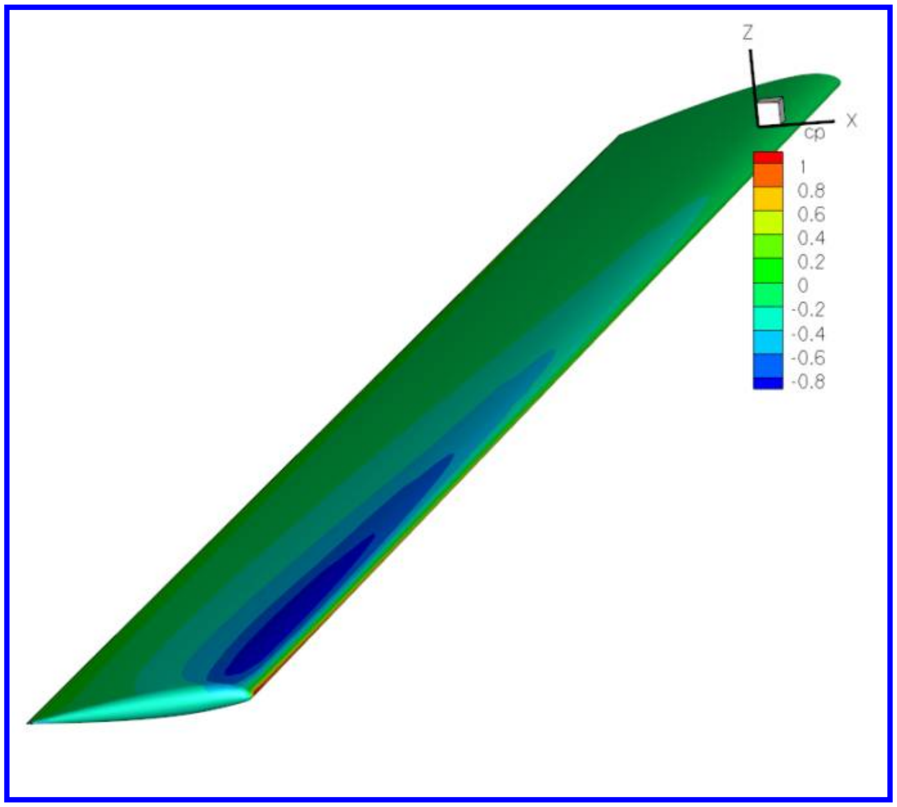


(b) Zoomed in view

Figure 4. HART2 rotor mesh consisting of 2.32 million points used in the optimization example.

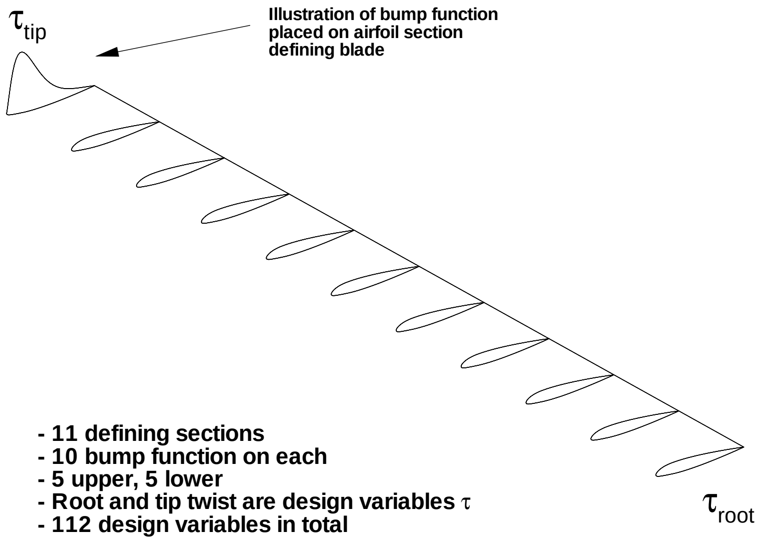


(a) Planform view

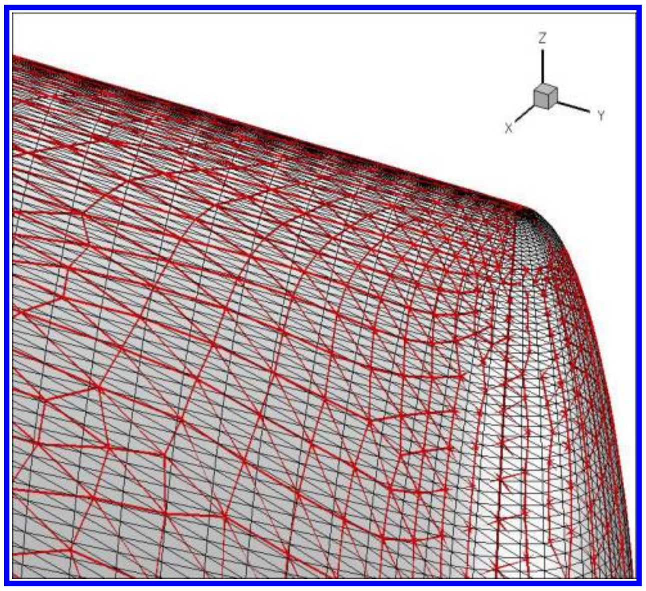


(b) Zoomed in view

Figure 5. C_P contours for the baseline HART2 rotor in hover after one revolution. The mesh consists of 2.32 million vertices.

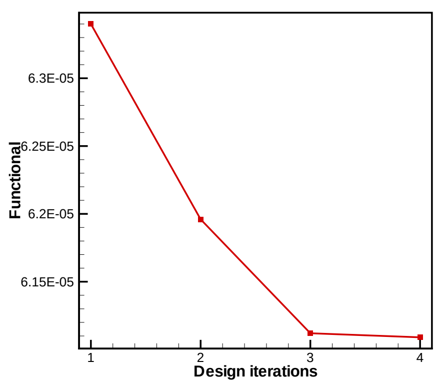

τ_{tip}
Illustration of bump function placed on airfoil section defining blade
- 11 defining sections
- 10 bump function on each
- 5 upper, 5 lower
- Root and tip twist are design variables τ
- 112 design variables in total
τ_{root}

(a) Blade design parameters

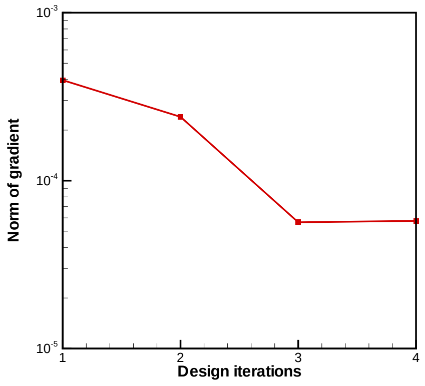


(b) Baseline structured blade mesh overlap with CFD mesh

Figure 6. Illustration of (a) baseline blade with design parameters and (b) overlap in tip region between baseline blade structured mesh and CFD surface unstructured mesh.



(a) Convergence of the functional in the optimization example



(b) Convergence of the norm of the gradient vector in the optimization example

Figure 7. Design optimization convergence.

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

[13]Yang, Z. and Mavriplis, D. J., "A Mesh Deformation Strategy Optimized by the Adjoint Method on Unstructured Meshes," *AIAA Journal*, Vol. 45, No. 12, 2007, pp. 2885–2896.

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

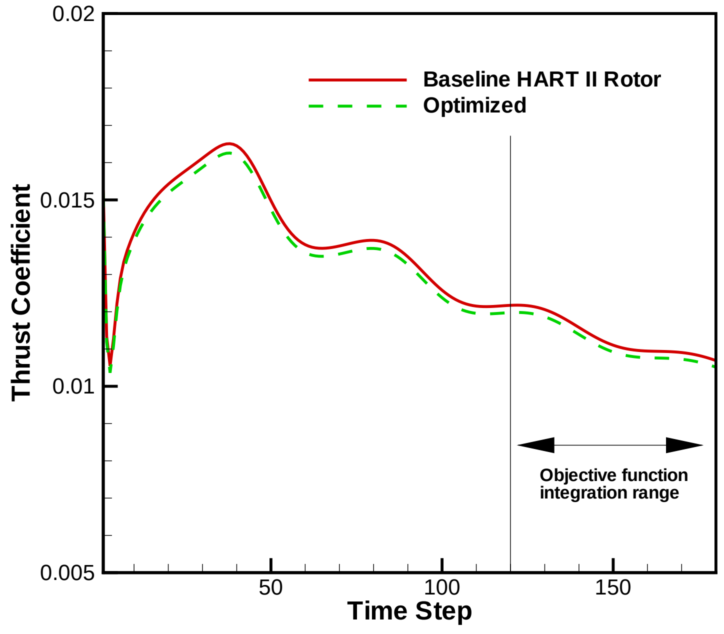
[15]Mavriplis, D. and Yang, Z., "Construction of the discrete geometric conservation law for high-order time-accurate simulations on dynamic meshes," *Journal of Computational Physics*, Vol. 213, 2006, pp. 557–573.

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

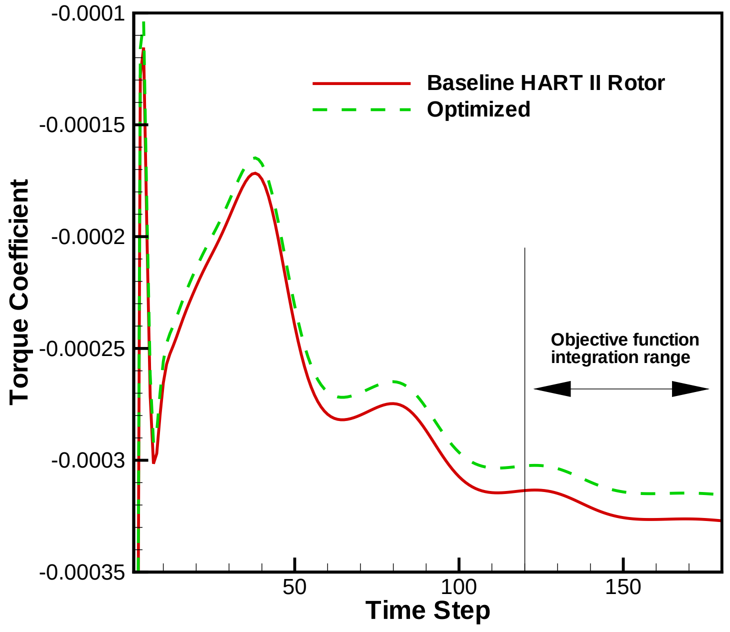
[17]Mavriplis, D. J., "Multigrid Strategies for Viscous Flow Solvers on Anisotropic Unstructured Meshes," *Journal of Computational Physics*, Vol. 145, No. 1, Sept. 1998, pp. 141–165.

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

[19]Mani, K. and Mavriplis, D. J., "Adjoint Based Sensitivity Formulation for Fully Coupled Unsteady Aeroelasticity Problems," *AIAA Journal*, Vol. 47, No. 8, 2009, pp. 1902–1915.

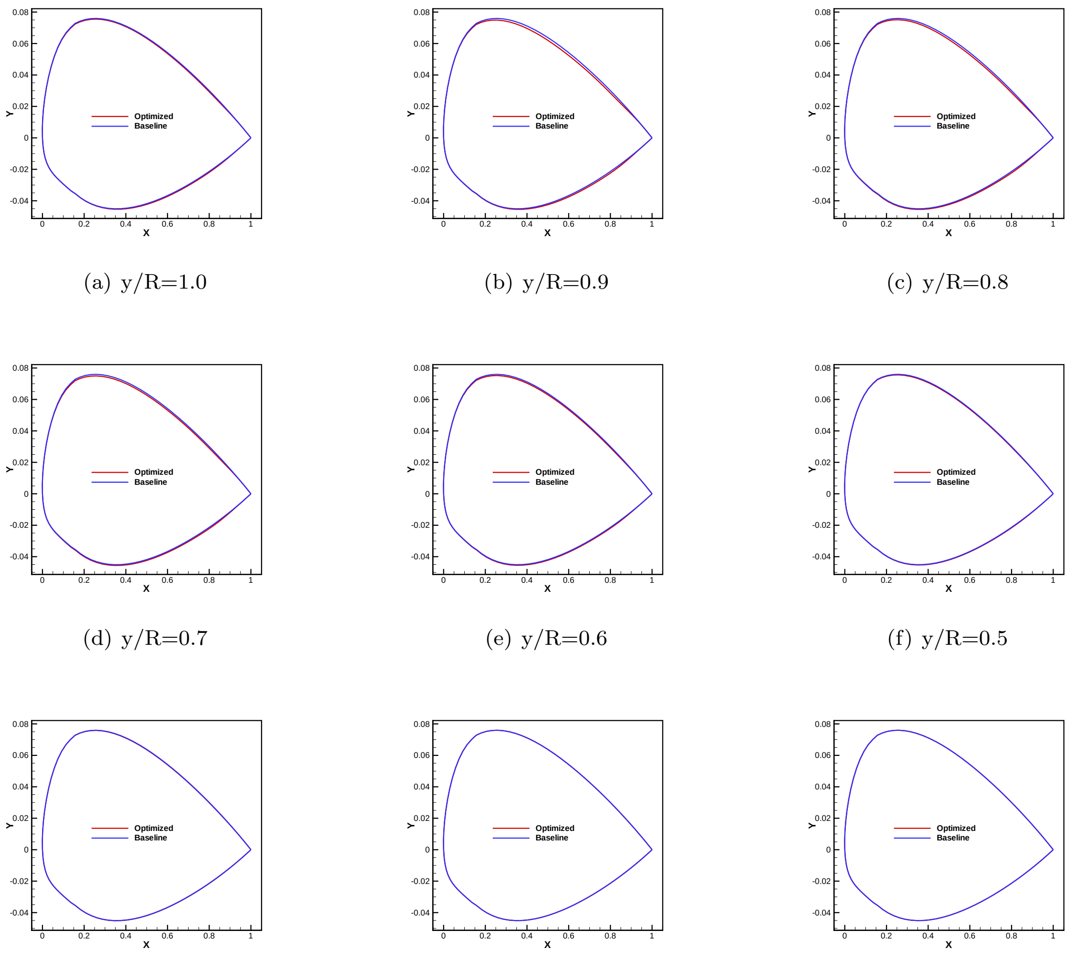


(a) Thrust coefficient



(b) Torque coefficient

Figure 8. Comparison of the time varying thrust and torque coefficients for the baseline HART2 rotor and the optimized rotor.



(a) y/R=1.0 (b) y/R=0.9 (c) y/R=0.8

(d) y/R=0.7 (e) y/R=0.6 (f) y/R=0.5

(g) y/R=0.4 (h) y/R=0.3 (i) y/R=0.2

Figure 9. Comparison of baseline airfoils defining the HART2 blade and the optimized airfoils sections (sections between y/R=0.2 and y/R=1.0 shown).