Toward a Pseudo-Time Accurate Formulation of the Adjoint and Tangent Systems

Emmett Padway *
Dimitri Mavriplis †
Department of Mechanical Engineering, University of Wyoming, Laramie, WY 82071

This paper presents efforts to better compute sensitivities using adjoint and tangent systems for primal problems with partial or incomplete convergence. First we apply the unsteady adjoint to the steady equations to obtain a pseudo-time accurate formulation of the tangent and adjoint systems. This approach is demonstrated for the tangent and adjoint systems for three different solution algorithms: forward Euler and 5-stage Runge-Kutta pseudo-time evolution, and a quasi-Newton method. Proofs for the forward Euler and quasi-Newton method systems are shown. Those for the 5-stage Runge-Kutta scheme are found in the Appendix. Verification results for the derived schemes are shown followed by three test cases. The first test case compares results of the steady-state adjoint to the pseudo-time accurate formulation, showing magnitude differences and evaluating the angle between the two computed sensitivity vectors. The second case compares the steady-state method to the pseudo-time accurate formulation in a subsonic case for a non-converging primal problem. This case shows that while averaging the state decreases the angle between the two vectors, there are still significant magnitude differences between the vectors. The final test case is a transonic case with a non-converging primal and shows that the sensitivity vector computed from an averaged state still has a significant angle difference with the vector computed from the pseudo-time accurate result. Finally, an investigation into the effect of averaging the pseudo-time accurate adjoint computes sensitivities is presented.

I. Introduction

As computers and algorithms have developed, the field of Computational Fluid Dynamics (CFD) has expanded to include a larger role for Multidisciplinary Design Optimization (MDO) and Adaptive Mesh Refinement (AMR). Much MDO is done using gradient driven optimization as this allows for far fewer function evaluations when compared to global methods, such as genetic algorithms. This is necessary when the function evaluations are as expensive as they are for many CFD simulations. The optimization toolboxes used for these purposes (SNopt,[1,2] DAKOTA,[3] etc.) are indifferent to the source of the gradient and this has allowed people to provide sensitivities through finite-difference (real or complex-step), tangent, or adjoint methods. The last two methods[4] are more accurate than the traditional finite-difference method (providing they are properly implemented), and are developed through conditions on convergence to generate mathematical equations to solve for the sensitivities. The main benefit of the complex-step finite-difference method is that for small complex steps it provides the exact sensitivities of the computational process, with the downsides being the slowdown of the process due to the inclusion of complex arithmetic, as well as the fact that this method scales with the number of design variables. The tangent formulation scales similarly to the finite-difference method, in that it generates a linear system which scales with the number of design variables. However the formulation is based on the requirement that the primal problem be fully converged to generate its linear system. The adjoint system uses that same residual requirement, but is different in that it transposes the system and scales independently of the number of design variables; it scales with the number of objective functions. For many aerospace applications the number of design variables is one to two orders of magnitude higher than the number of objective functions and the adjoint method is a powerful

*Graduate Student, Member AIAA; email: epadway@uwyo.edu
†Professor, AIAA Associate Fellow; email: mavripl@uwyo.edu

and preferred technique to obtain sensitivities. Additionally the adjoint solution is well suited for AMR as a tool to refine a computational mesh to increase accuracy in an output of interest.[5,6]

As was mentioned previously, both the adjoint and tangent systems require that the primal problem be fully converged, but as the field of CFD has tried more difficult problems (higher-order formulations, blunt geometries, or time-accurate simulations) this constraint has become difficult or impractical to obey, and numerous design or mesh refinement cycles have been done using adjoint systems linearized about partially converged primal solutions. This defect shows itself in less robust and more difficult to solve adjoint systems that are also sensitive to the specific state of the primal problem about which they are linearized when the primal simulation is terminated.[7] For AMR, this can lead to obtaining different AMR patterns which can lead to refining the mesh in areas that may not contribute to the output of interest, and not refining in areas where the output of interest is affected. This can impact the accuracy of the primal problem as the mesh adaptation proceeds. In the realm of design optimization, inaccurate adjoints can lead to inaccurate sensitivities, which can change the course of the design cycle and lead to stagnation (as the Karush-Kuhn-Tucker (KKT) conditions, which govern termination, require that the gradient vanish at a local extremum).[8]

As mentioned above, Krakos and Darmofal[7] illustrate that for a nonconvergent case, the state about which the adjoint is linearized can notably affect the sensitivity calculations. The authors then show that by running the non-convergent steady-state case as a time-accurate case and applying the unsteady adjoint to the time-accurate case returns useful and accurate adjoint computed sensitivities for the time-averaged lift. The authors suggest that, for steady-state cases which can be solved by strong solvers, but which may show physical unsteadiness, the time-accurate approach is in fact the proper analysis framework to use as otherwise CFD practitioners may risk obtaining unphysical and non-useful sensitivity vectors. Krakos et al. follow their previous work with an investigation of statistical and windowing techniques to allow only partial time integration for periodic primal flows with time-averaged outputs of interest.[9] The authors demonstrate that with proper windowing techniques only partial time integration is required to obtain accurate sensitivities; something that will be investigated later in this paper. However even for time-accurate formulations, Mishra et al.[10] have demonstrated growing sensitivity error through the adjoint reverse time-integration due to partial convergence of the primal problem at each implicit time step, which is a common practice in applied CFD problems.

Luers et al.[11] illustrate the importance of accurate gradients in well converging cases that have not reached deep convergence. Their paper shows steady-state optimizations for a CRESCENDO turbine cascade, and contrasts the optimization for finite-difference computed gradients to that driven by adjoint computed gradients in a case with a 5 order drop in the primal residual. It is shown that that by using the finite-difference provided gradients the optimized efficiency increases by a greater percentage than by using the adjoint provided gradients, while still obeying the constraint of the primal problem.

In an effort to address this problem, Brown and Nadarajah[12] investigate and bound for the error in the adjoint computed sensitivities arising from partial convergence of the primal problem. From these error estimates, the authors feed the calculated error bounds into the optimization toolboxes, and show improved convergence properties in the design cycle. In a simpler approach, Shimizu et al., rather than computing the error directly, have worked to lessen it by applying least squares shadowing and windowing techniques towards better behaved adjoint systems and more accurate adjoint computed sensitivities with an application to chaotic adjoint problems.[13,14]

This paper does not address chaotic flows, but instead seeks to address the effects of incomplete primal problem convergence in steady state problems by developing a different constraint, one that is accurate at every step in the pseudo-time history. This paper shows verification of the methods through a comparison of the computed sensitivities with complex-step finite-difference sensitivities, and proceeds to examine the accuracy of sensitivities obtained in this manner as a function of the pseudo-time integration space and averaging windows

II. Background and In-House Solver

II.A. Governing Equations

We developed an in-house flow solver to solve the steady-state Euler equations on unstructured meshes. The steady-state compressible Euler equations (which may also be referred to as the primal problem) can be written as follows.

$$\nabla \cdot F(u(D)) = 0 \tag{1}$$

This can also be written as:

$$R(u(D), D) = 0 \tag{2}$$

where u is the conservative variable vector, is the mesh coordinate vector, D is the design variable vector, and $F(u)$ is the conservative variable flux.

II.B. Spatial Discretization

The residual about the closed control volume is given as:

$$R = \int_{dB} [F(u(D))] \cdot n(x(D)), \mathrm{dB} = \sum_{i=1}^{n_{edge}} F_{e_i}^{\perp}(u(D), n_{e_i}(x(D)))B_{e_i}(x(D)) \tag{3}$$

This equation gives the operator in the aforementioned requirement for the adjoint and tangent systems.

The solver used in this work is a steady-state finite-volume cell-centered Euler solver with second-order spatial accuracy implemented for triangular elements. Second-order accuracy is implemented through weighted least squares gradient reconstruction.[15] The solver has three different flux calculations implemented and linearized, these are the Lax-Friedrichs,[16] Roe,[17] and van Leer[18] schemes. In this work, only the Lax-Friedrichs and Van-Leer schemes are used.

II.C. Steady-State Solver

The solver technology for this code uses either explicit time stepping through pseudo time using a forward Euler time discretization, or a low storage five stage Runge-Kutta scheme, or a quasi-Newton method. The forward Euler scheme is written as follows.

$$u^k = u^{k-1} + CFL\Delta t(u^{k-1}, D)R(u(D), D) \tag{4}$$

The five stage Runge-Kutta scheme is given as follows, iterating n through pseudo time:

$$\begin{aligned}
u^{k,0} &= u^{k-1,5} \\
u^{k,1} &= u^{k,0} + CFL\alpha^0\Delta tR(u^{k,0}) \\
u^{k,2} &= u^{k,0} + CFL\alpha^1\Delta tR(u^{k,1}) \\
u^{k,3} &= u^{k,0} + CFL\alpha^2\Delta tR(u^{k,2}) \\
u^{k,4} &= u^{k,0} + CFL\alpha^3\Delta tR(u^{k,3}) \\
u^{k,5} &= u^{k,0} + CFL\alpha^4\Delta tR(u^{k,4}) \\
u^{k+1,0} &= u^{k,5}
\end{aligned} \tag{5}$$

where α_l are the Runge-Kutta stage coefficients used in this work.

$$\begin{aligned}
\alpha_0 &= 0.0695 \\
\alpha_1 &= 0.1602 \\
\alpha_2 &= 0.2898 \\
\alpha_3 &= 0.5060 \\
\alpha_4 &= 1.0
\end{aligned} \tag{6}$$

The equation for the local explicit time step limit Δt is given as:

$$\Delta t_i = \frac{r_i}{\sqrt{(u^2 + v^2)} + c} \tag{7}$$

where r_i is the circumference of the inscribed circle for mesh cell i, u and v are the horizontal and vertical velocity components respectively, and c is the speed of sound in the triangular element.

Alternatively, a quasi-Newton method is implemented using pseudo-transient continuation (PTC) with a BDF1 pseudo temporal discretization scheme. For Newton's method the time-stepping procedure is written as:

$$u^k = u^{k-1} + \Delta u \tag{8}$$

where we compute Δu by solving the following system of linear equations.

$$[P]\,\Delta u = -R(u) \tag{9}$$

We can substitute our expression for Δu into the time-stepping equation (8) to obtain our final form of this equation.

$$u^k = u^{k-1} - [P_{k-1}]^{-1}\,R \tag{10}$$

Here $[P_{k-1}]$ is a first-order spatially accurate Jacobian augmented with a diagonal term to ensure that it is diagonally dominant.

$$[P_{k-1}] = \left[\frac{\partial R}{\partial u^{k-1}}\right] + \frac{I}{\Delta t CFL} \tag{11}$$

In order to solve the linear system we use a block Gauss-Seidel solver. This is done by lagging the off-diagonal components, with the right hand side being the linear residual of the original system. In this work the residual is the second-order accurate spatial residual operator, and $[P_{k-1}]$ is the term outlined in equation (11). Equation (9) is then solved iteratively as

$$[D]\,\Delta(\Delta u)^l = -R(u) - [P_{k-1}]\,\Delta u^l \tag{12}$$

where the matrix $[D]$ is the element block diagonal entry in the Jacobian matrix.

$$\Delta u^{l+1} = \Delta u^l + \omega(\Delta(\Delta u))^l \tag{13}$$

This linear solver is also applied in a similar fashion to the tangent and adjoint solvers, to be described subsequently.

II.D. Mesh Deformation

Mesh deformation is required for the design problem to deform the computational mesh in response to shape design changes, and although we do not show any design cycles in this work, the mesh deformation equations are required to compute the mesh sensitivities which are in turn used to compute the objective function sensitivities. There are two mesh deformation schemes used in this work to obtain mesh sensitivities and for future use in design optimization. The first is a local method based on a spring analogy,[19] where a system of linear equations is solved for interior mesh deformations based on the surface deformations as:

$$[K]\,\Delta x_v = \Delta x_s \tag{14}$$

where $[K]$ is the sparse spring stiffness matrix, Δx_v represents the interior mesh deformations, and Δx_s represents the surface mesh deformations. The off-diagonal elements of this matrix are only non-zero where the mesh nodes corresponding to the off-diagonal elements are connected by an edge. In those entries the stiffness value is taken as:

$$k_{ik} = -\frac{1}{r_{ik}^2} \tag{15}$$

where r_{ik} is the distance between nodes i and k. The diagonal element is the sum of the edge stiffnesses incident onto each node:

$$k_{ii} = \sum_{k=1}^{n_{con}} \frac{1}{r_{ik}^2} \tag{16}$$

This equation is a linear problem and is solved using a point Gauss-Seidel method.

The second mesh deformation scheme is a global inverse distance weighted method.[20] The displacement at each node i in the mesh is determined as follows:

$$\Delta x_{v_i} = \frac{\sum w_{ik}(\vec{r}_{ik})\Delta x_{s_k}}{\sum w_{ik}(\vec{r}_{ik})} \tag{17}$$

The w function is the weight of surface point k on interior node i, the delta function is its displacement, and $\vec{r}_{ik}$ is the distance between the interior mesh point and the surface mesh point. The weight function is calculated as follows:

$$w_{ik}(\vec{r}_{ik}) = A_i\left[\left(\frac{L_{def}}{\|\vec{r}_{ik}\|}\right)^a + \left(\frac{\alpha L_{def}}{\|\vec{r}_{ik}\|}\right)^b\right] \tag{18}$$

L_{def} is the characteristic length of the body, a and b are experimentally derived constants, α is a a fraction of L_{def} to reserve for stiffer deformation. This method results in an explicit update and is cheaper to evaluate than the spring analogy method outlined first.

II.E. Shape Design Variables

The shape design parameterization is implemented using Hicks-Henne bump functions[19] to smoothly perturb the airfoil coordinates. The equation given as:

$$\delta x_s(x) = a \cdot sin^4(\pi x^{m_i}) \tag{19}$$

where the variable m_i is defined as:

$$m_i = \frac{ln(.5)}{ln(x_{M_i})} \tag{20}$$

where x_{M_i} is the bump center, δx_s represents the surface nodal displacements by the bump function, x is the location of the surface node in question, and a is the amplitude of the bump function, which serves as the design variable D_i. In this code, the bump functions are equally spaced along the airfoil and act on both the top and bottom surfaces, forcing symmetric changes in the airfoil.

II.F. A Review of Tangent and Adjoint Systems

II.F.1. Tangent Formulation

For an aerodynamic optimization problem, we consider an objective functional $L(u(D), x(D))$, for example lift or drag, where u is the conservative variable vector, and x is the node coordinate vector. In order to obtain an expression for the sensitivities we take the derivative of the objective functional:[21]

$$\frac{dL}{dD} = \frac{\partial L}{\partial x}\frac{\partial x}{\partial D} + \frac{\partial L}{\partial u}\frac{\partial u}{\partial D} \tag{21}$$

For the above expression $\frac{\partial L}{\partial x}$ and $\frac{\partial L}{\partial u}$ can be directly obtained by differentiating the corresponding subroutines in the code. $\frac{\partial x}{\partial D}$ is calculated by solving the spring analogy mesh deformation equation:

$$[K]\frac{\partial x_v}{\partial D_i} = \frac{\partial x_s}{\partial D_i} \tag{22}$$

and we calculate $\frac{\partial x_s}{\partial D_j}$ through differentiating the shape design variables. For the global inverse distance weighted method, the mesh sensitivities are computed directly as a function of the surface coordinate sensitivities:

$$\frac{\partial x_{v_i}}{\partial D_j} = \frac{\sum w_{ik}(\vec{r}_{ik})\frac{\partial x_{s_k}}{\partial D_j}}{\sum w_{ik}(\vec{r}_{ik})} \tag{23}$$

It is not possible to obtain $\frac{\partial u}{\partial D}$ through linearization of the subroutines in the code without linearizing the entire primal solution process, as will be covered in later sections. In order to solve for this term we use the constraint that for a fully converged flow $R(u(D), x(D)) = 0$. By taking the derivative of the residual operator we obtain the equation below.

$$\left[\frac{\partial R}{\partial x}\right]\frac{\partial x}{\partial D} + \left[\frac{\partial R}{\partial u}\right]\frac{\partial u}{\partial D} = 0 \tag{24}$$

We can isolate the sensitivity of the residual to the design variables to obtain the tangent system.

$$\left[\frac{\partial R}{\partial u}\right]\frac{\partial u}{\partial D} = -\left[\frac{\partial R}{\partial x}\right]\frac{\partial x}{\partial D} \tag{25}$$

We solve this linear system, using hand differentiated subroutines to provide the left hand matrix $\left[\frac{\partial R}{\partial u}\right]$, the right hand side $\left[\frac{\partial R}{\partial x}\right]\frac{\partial x}{\partial D}$ (which scales with the design variables), and obtain $\frac{\partial u}{\partial D}$. We then substitute $\frac{\partial u}{\partial D}$ into equation (21) to obtain the final sensitivities.

II.F.2. Discrete Adjoint Formulation

The adjoint formulation begins with the same sensitivity equation:

$$\frac{dL}{dD} = \frac{\partial L}{\partial x}\frac{\partial x}{\partial D} + \frac{\partial L}{\partial u}\frac{\partial u}{\partial D} \tag{26}$$

Using the condition $R(u(D), D) = 0$, we return to equation (25) and pre-multiply both sides of the equation by the inverse Jacobian matrix to obtain:

$$\frac{\partial u}{\partial D} = -\left[\frac{\partial R}{\partial u}\right]^{-1}\left[\frac{\partial R}{\partial x}\right]\frac{\partial x}{\partial D} \tag{27}$$

Substituting the above expression into the sensitivity equation yields:

$$\frac{dL}{dD} = \frac{\partial L}{\partial x}\frac{\partial x}{\partial D} - \frac{\partial L}{\partial u}\left[\frac{\partial R}{\partial u}\right]^{-1}\left[\frac{\partial R}{\partial x}\right]\frac{\partial x}{\partial D} \tag{28}$$

We then define an adjoint variable $\mathbf{\Lambda}$ such that:

$$\mathbf{\Lambda}^T = -\frac{\partial L}{\partial u}\left[\frac{\partial R}{\partial u}\right]^{-1} \tag{29}$$

which gives an equation for the adjoint variable:

$$\left[\frac{\partial R}{\partial u}\right]^T \mathbf{\Lambda} = -\left[\frac{\partial L}{\partial u}\right]^T \tag{30}$$

We can solve this linear system and obtain the sensitivities for the objective function as follows:

$$\frac{dL}{dD} = \left[\frac{\partial L}{\partial x} + \mathbf{\Lambda}^T\frac{\partial R}{\partial x}\right]\frac{\partial x}{\partial D} \tag{31}$$

We can then define a mesh adjoint variable $\mathbf{\Lambda_x}$:

$$[K]^T\,\mathbf{\Lambda_x} = \left[\frac{\partial L}{\partial x}\right]^T + \left[\frac{\partial R}{\partial x}\right]^T\mathbf{\Lambda} \tag{32}$$

and the final expression for sensitivity is given as:

$$\frac{dL}{dD} = \mathbf{\Lambda_x}^T\frac{\partial x_s}{\partial D} \tag{33}$$

The adjoint system is of interest, because as mentioned in the introduction, it results in an equation for the sensitivity that does not scale with the number of design variables.

III. Development of the Pseudo-Time Accurate Tangent

To show the development and utility of the pseudo-time accurate tangent approach, we begin with the simplest time-evolution process, forward Euler time-stepping, after which we demonstrate this approach for a more complex time-stepping algorithm. This will be done by taking the derivatives of the solution process, with the goal being to get an exact derivative of the solution process itself, one that will correspond exactly to the sensitivities obtained by the complex step method at each step of the solution process. The idea is that a sensitivity obtained from the exact differentiation of the solution process is preferable to one obtained by linearizing the adjoint system about an unconverged state, because, as stated above, adjoint systems of unconverged primals are very sensitive to the specific state about which they are linearized. We note that in this section and the derivation of the adjoint section, when taking the derivative of an operator with respect to the design variables, $\frac{\partial}{\partial D}$ is an abbreviation of $\frac{\partial}{\partial x}\frac{\partial x}{\partial D}$.

III.A. Tangent System for Forward Euler Pseudo-Time Evolution

The forward Euler time-step is written as:

$$u^k = u^{k-1} + CFL\Delta t(u^{k-1}, D)R(u(D), D) \tag{34}$$

We take the derivative of each side of the equation and ob tain the following evolution equation.

$$\frac{du^k}{dD} = \frac{du^k}{dD} + CFL\Delta t\left[\frac{\partial R}{\partial D} + \frac{\partial R}{\partial u}\frac{du}{dD}^{k-1}\right] + CFL\left[\frac{\partial \Delta t}{\partial D} + \frac{\partial \Delta t}{\partial u^{k-1}}\frac{du^{k-1}}{dD}\right]R(u^{k-1}) \tag{35}$$

This equation gives a simple expression for the solution of the tangent system. In order to write the sensitivity of an objective function which depends on the m final states $L(u^n, u^{n-1}, ..., u^{n-m}, x(D))$ for a simulation with n time steps, we express it as follows:

$$\frac{dL}{dD} = \frac{\partial L}{\partial D} + \frac{\partial L}{\partial u^n}\frac{du^n}{dD} + \frac{\partial L}{\partial u^{n-1}}\frac{du^{n-1}}{dD} + ... + \frac{\partial L}{\partial u^{n-m}}\frac{du^{n-m}}{dD} \tag{36}$$

An example of such an objective functional would be the average lift over the final m pseudo time steps of the simulation.

III.B. Tangent System for Quasi-Newton Method

For this section, Newton's method was implemented using pseudo-transient continuation (PTC) with a BDF1 scheme in the context of a quasi-Newton method. This is specifically not a full Newton method as it uses an approximation to the Jacobian matrix which is only first-order accurate, and the resulting linear system is only approximately solved. For Newton's method the time-stepping procedure is written as in equation (8). We can substitute our expression for Δu into the time-stepping equation to obtain our final form of this equation.

$$u^k = u^{k-1} - [P_{k-1}]^{-1} R \tag{37}$$

where $[P_{k-1}]$ is a first-order accurate Jacobian augmented with a diagonal term to ensure that it is diagonally dominant.

$$[P_{k-1}] = \frac{\partial R}{\partial u^{k-1}} + \frac{I}{\Delta t CFL} \tag{38}$$

If we take the derivative of each side of equation (38) we obtain:

$$\frac{du}{dD}^k = \frac{du}{dD}^{k-1} - [P_{k-1}]^{-1}\left[\frac{\partial R}{\partial D} + \frac{\partial R}{\partial u^{k-1}}\frac{du}{dD}^k\right] - R(u^{k-1})\left[\frac{\partial [P_{k-1}]^{-1}}{\partial D} + \frac{\partial [P_{k-1}]^{-1}}{\partial u}\frac{du}{dD}^{k-1}\right] \tag{39}$$

If we choose to neglect the change in the preconditioner we can simplify the above equation a great deal, as is shown below, by avoiding taking derivatives of inverse Jacobians. This can be done either by using a frozen Jacobian, which is also known as a Newton-Chord method, or by arguing by egodicity that for partially converged cases, oscillations about some mean are independent of the initial state. It should be noted that although the derivative of the the inverse preconditioner is nonzero, the argument used is based on the assumption that the residual at this point is very small, and will be multiplying this derivative term, as the last term on the right hand side of equation (39), will lead to negligible contributions from this term overall. Consequently, we will investigate whether running for a long enough period of pseudo-time in the limit cycle oscillation zone gives sufficient information to provide a suitable approximation of the tangent system and allows a close approximation of the sensitivities obtained using the complex-step method, which are considered to be the exact sensitivities of the entire process. Proceeding with the simplified equation:

$$\frac{du}{dD}^k = \frac{du}{dD}^{k-1} - [P_{k-1}]^{-1}\left[\frac{\partial R}{\partial D} + \frac{\partial R}{\partial u}\frac{du}{dD}^{k-1}\right] \tag{40}$$

This can then be rewritten as:

$$\frac{du}{dD}^k = \frac{du}{dD}^{k-1} + \Delta\left(\frac{du}{dD}\right) \tag{41}$$

and we solve the following linear system:

$$[P_{k-1}]\,\Delta\left(\frac{du}{dD}\right) = -\left[\frac{\partial R}{\partial D} + \frac{\partial R}{\partial u}\frac{du}{dD}^{k-1}\right] \tag{42}$$

It is important to note that we are not taking the exact inverse of the preconditioner matrix in the iterative equation above, rather we are performing the exact number of steps to invert it as we did in the primal problem, as we are drawing directly off the primal solution process. If this iterative process is run at the converged state then it corresponds to the dual direct differentiation; its adjoint presented later is the exact dual of that process.[22,23]

IV. Development of the Pesudo-Time Accurate Adjoint

As stated above, the pseudo-time accurate adjoint method is drawn from the derivation of the unsteady adjoint. In this method we look at each pseudo-time step and work backwards through pseudo-time to get the pseudo-time accurate adjoint solution. In this derivation the objective function is a pseudo-time averaged functional, averaged over the last m steps. For this derivation we will say that we have a program that runs through n pseudo-time steps. From this we define our objective function L as:

$$L = L(u^n, u^{n-1}, ..., u^{n-m}, D) \tag{43}$$

where u^n is the conservative variable vector at the final time step and D is the design variable vector. For the constraint we cannot select $R(u, D) = 0$, as this is not true at each pseudo-time step; instead, we select a constraint based on the pseudo-time evolution of the solution, for which, the kth constraint will be referred to as G^k. We know because we are using first order time-stepping that the constraint is dependent only on the old time-step, the new time-step, and the design variables, expressed as follows.

$$G^k = G^k(u^k, u^{k-1}, D) = 0 \tag{44}$$

We define an augmented objective function with n constraints and n Lagrange multipliers:

$$\begin{aligned} J(D, u^n, u^{n-1}, u^{n-2}, ..., \Lambda^n, \Lambda^{n-1}, ...) = {} & L(u^n, u^{n-1}, ..., u^{n-m}, D) \\ & + \Lambda^{nT} G^n(u^n(D), u^{n-1}(D), D) \\ & + \Lambda^{n-1T} G^{n-1}(u^{n-1}(D), u^{n-2}(D), D) \\ & + ... \\ & + \Lambda^{1T} G^1(u^1(D), u^0(D), D) \end{aligned} \tag{45}$$

In order to get an expression for the adjoint we take the derivative of the augmented objective function with respect to the conservative variables at different pseudo-time steps, and choose our lagrange multiplier such that these partial derivatives are equal to 0.

$$\begin{aligned} \frac{\partial J}{\partial u^n} &= \frac{\partial L}{\partial u^n} + \Lambda^{nT}\frac{\partial G^n}{\partial u^n} = 0 \\ \frac{\partial J}{\partial u^{n-1}} &= \frac{\partial L}{\partial u^{n-1}} + \Lambda^{nT}\frac{\partial G^n}{\partial u^{n-1}} + \Lambda^{n-1T}\frac{\partial G^{n-1}}{\partial u^{n-1}} = 0 \\ \frac{\partial J}{\partial u^{n-2}} &= \frac{\partial L}{\partial u^{n-2}} + \Lambda^{n-1T}\frac{\partial G^{n-1}}{\partial u^{n-2}} + \Lambda^{n-2T}\frac{\partial G^{n-2}}{\partial u^{n-2}} = 0 \\ &... \\ \frac{\partial J}{\partial u^1} &= \frac{\partial L}{\partial u^1} + \Lambda^{2T}\frac{\partial G^2}{\partial u^1} + \Lambda^{1T}\frac{\partial G^1}{\partial u^1} = 0 \end{aligned} \tag{46}$$

Using $L = L(u^n, u^{n-1}, ..., u^{n-m}, D)$, we get the following equation.

$$\begin{aligned}
\frac{\partial J}{\partial u^n} &= \frac{\partial L}{\partial u^n} + \Lambda^{nT}\frac{\partial G^n}{\partial u^n} = 0 \\
\frac{\partial J}{\partial u^{n-1}} &= \frac{\partial L}{\partial u^{n-1}} + \Lambda^{nT}\frac{\partial G^n}{\partial u^{n-1}} + \Lambda^{n-1T}\frac{\partial G^{n-1}}{\partial u^{n-1}} = 0 \\
&... \\
\frac{\partial J}{\partial u^{n-m}} &= \frac{\partial L}{\partial u^{n-m}} + \Lambda^{n-(m-1)T}\frac{\partial G^{n-(m-1)}}{\partial u^{n-m}} + \Lambda^{n-mT}\frac{\partial G^{n-m}}{\partial u^{n-m}} = 0 \\
\frac{\partial J}{\partial u^{n-(m+1)}} &= \Lambda^{n-mT}\frac{\partial G^{n-m}}{\partial u^{n-(m+1)}} + \Lambda^{n-(m+1)T}\frac{\partial G^{n-(m+1)}}{\partial u^{n-(m+1)}} = 0 \\
&... \\
\frac{\partial J}{\partial u^1} &= \Lambda^{2T}\frac{\partial G^2}{\partial u^1} + \Lambda^{1T}\frac{\partial G^1}{\partial u^1} = 0
\end{aligned} \tag{47}$$

Using the equation for the adjoint at the final pseudo-time step we obtain:

$$\left[\frac{\partial G^n}{\partial u^n}\right]^T \Lambda^n = -\left[\frac{\partial L}{\partial u^n}\right]^T \tag{48}$$

using the other adjoint equations we get an adjoint recurrence relation for k = 2, 3, ...,m:

$$\frac{\partial G^{k-1}}{\partial u^{k-1}}^T \Lambda^{k-1} = -\frac{\partial G^k}{\partial u^{k-1}}^T \Lambda^k \tag{49}$$

with the adjoint recurrence relation for k = m+1, ..., n-1 as follows.

$$\frac{\partial G^{k-1}}{\partial u^{k-1}}^T \Lambda^{k-1} = -\frac{\partial G^k}{\partial u^{k-1}}^T \Lambda^k - \left[\frac{\partial L}{\partial u^{k-1}}\right]^T \tag{50}$$

If the objective function is only dependent on the final time-step we only have one recurrence relation for all k = 2, 3, ..., n-1:

$$\frac{\partial G^{k-1}}{\partial u^{k-1}}^T \Lambda^{k-1} = -\frac{\partial G^k}{\partial u^{k-1}}^T \Lambda^k \tag{51}$$

Lastly, we can take the derivative of equation (45) with respect to the design variables to get the sensitivity equation.

$$\frac{dJ}{dD} = \frac{\partial L}{\partial D} + \Lambda^{nT}\frac{\partial G^n}{\partial D} + \Lambda^{n-1T}\frac{\partial G^{n-1}}{\partial D} + \Lambda^{n-2T}\frac{\partial G^{n-2}}{\partial D} + ... \tag{52}$$

IV.A. Adjoint Computed Sensitivities for Forward Euler Pseudo-Time Evolution

As stated above, our constraint is based on the time-step. For forward Euler pseudo-time evolution, we set our time-stepping equation equal to 0 and write the constraint.

$$G^k(u^k(D), u^{k-1}(D), D) = u^k - u^{k-1} - CFL\Delta t R(u^{k-1}) = 0 \tag{53}$$

We take our constraint derivatives in order to substitute them into the adjoint recurrence relations:

$$\begin{aligned}
\frac{\partial G^k}{\partial u^k} &= I \\
\frac{\partial G^k}{\partial u^{k-1}} &= -I - CFL\Delta t^{k-1}\frac{\partial R(u^{k-1})}{\partial u^{k-1}} - CFL\frac{\partial \Delta t^{k-1}}{\partial u^{k-1}}R \\
\frac{\partial G^k}{\partial D} &= -CFL\Delta t^{k-1}\frac{\partial R(u^{k-1})}{\partial D} - CFL\frac{\partial \Delta t^{k-1}}{\partial D}R
\end{aligned} \tag{54}$$

Using the equation for the adjoint at the final pseudo-time step with our constraint derivatives we get the following expression:

$$[I]\,\Lambda^n = -\left[\frac{\partial L}{\partial u^n}\right]^T \tag{55}$$

we then substitute the constraint derivatives into equation (51) and get the following equation as our recurrence relation.

$$[I]\,\Lambda^{k-1} = -\left[-I - CFL\Delta t^{k-1}\frac{\partial R(u^{k-1})}{\partial u^{k-1}} - CFL\frac{\partial \Delta t}{\partial u^{k-1}}R(u^{k-1})\right]^T \Lambda^k \tag{56}$$

We can apply this adjoint and substitute the derivatives of the constraints of the augmented functional to get an expression of equation (52) suitable for the sensitivities of the forward Euler time-stepping process.

$$\begin{aligned}\frac{dJ}{dD} = \frac{\partial L}{\partial D} &- \Lambda^{nT}\left[CFL\Delta t^{k-1}\frac{\partial R(u^{k-1})}{\partial D} + CFL\frac{\partial \Delta t}{\partial D}R(u^{k-1})\right] \\ &- \Lambda^{n-1T}\left[CFL\Delta t^{k-2}\frac{\partial R(u^{k-2})}{\partial D} + CFL\frac{\partial \Delta t}{\partial D}R(u^{k-2})\right] \\ &- \ldots \\ &- \Lambda^{1T}\left[CFL\Delta t^{0}\frac{\partial R(u^{0})}{\partial D} + CFL\frac{\partial \Delta t}{\partial D}R(u^{0})\right]\end{aligned} \tag{57}$$

It is important to note that, for explicit time stepping methods, we obtain tangent and adjoint systems that do not require solving a system of linear equations, and instead only require time marching in the dual of the forward problem.

IV.B. Adjoint Computed Sensitivites for Quasi-Newton Method

As shown in the tangent system derivation our time-stepping equation is:

$$u^k = u^{k-1} - [P_{k-1}]^{-1} R \tag{58}$$

as in the forward Euler tangent system, we move all terms to one side and obtain the following equation as the constraint.

$$G^k(u^k(D), u^{k-1}(D), D) = u^k - u^{k-1} + [P_{k-1}]^{-1} R(u^{k-1}) = 0 \tag{59}$$

We have the preconditioner matrix defined as seen previously in (11):

$$[P_k] = \frac{\partial R(u^k)}{\partial u^k} + \frac{1}{\Delta t^k CFL^k} \tag{60}$$

As in the adjoint derivation for forward Euler pseudo-time evolution, we then take the derivatives of our constraint equations. These are written as follows.

$$\begin{aligned}\frac{\partial G^k}{\partial u^k} &= I \\ \frac{\partial G^k}{\partial u^{k-1}} &= -I + [P_{k-1}]^{-1}\frac{\partial R(u^{k-1})}{\partial u^{k-1}} + \frac{\partial [P_{k-1}]^{-1}}{\partial u^{k-1}}R(u^{k-1}) \\ \frac{\partial G^k}{\partial D} &= [P_{k-1}]^{-1}\frac{\partial R(u^{k-1})}{\partial D} + \frac{\partial [P_{k-1}]^{-1}}{\partial D}R(u^{k-1})\end{aligned} \tag{61}$$

By using the same logic as in the pseudo-time accurate tangent method, we choose to neglect terms multiplied by the residual and simplify the constraint derivatives to those below. As in the tangent system we are now making only an approximation of the exact sensitivity and error equations, with the goal being to run long enough in the oscillatory portion of the convergence history to get a good enough adjoint approximation to compute sensitivities.

$$\begin{aligned}\frac{\partial G^k}{\partial u^k} &= I \\ \frac{\partial G^k}{\partial u^{k-1}} &= -I + [P_{k-1}]^{-1}\frac{\partial R(u^{k-1})}{\partial u^{k-1}} \\ \frac{\partial G^k}{\partial D} &= [P_{k-1}]^{-1}\frac{\partial R(u^{k-1})}{\partial D}\end{aligned} \tag{62}$$

Using the equation for the adjoint at the final pseudo-time step with our constraint derivatives we get the same expression as in the forward Euler pseudo-time accurate adjoint.

$$[I]\,\Lambda^n = -\left[\frac{\partial L}{\partial u^n}\right]^T \tag{63}$$

Substituting in the constraint derivatives into equation (51) returns:

$$[I]\,\Lambda^{k-1} = -\left[-I + [P_{k-1}]^{-1}\,\frac{\partial R(u^{k-1})}{\partial u^{k-1}}\right]^T \Lambda^k \tag{64}$$

we can also write this recurrence relation in delta form so that it more closely follows the primal problem. To this end, we define a $\Delta\Lambda$ such that $\Lambda^{k-1} = \Lambda^k + \Delta\Lambda$. Which gives the recurrence relation as:

$$\Delta\Lambda = -\left[[P_{k-1}]^{-1}\,\frac{\partial R(u^{k-1})}{\partial u^{k-1}}\right]^T \Lambda^k \tag{65}$$

distributing the transpose allows us to rewrite the equation.

$$\Delta\Lambda = -\left[\frac{\partial R(u^{k-1})}{\partial u^{k-1}}\right]^T [P_{k-1}]^{-T}\,\Lambda^k \tag{66}$$

This motivates us to define a secondary adjoint variable for each recurrence relation:

$$[P_{k-1}]^T\,\psi^k = \Lambda^k \tag{67}$$

we then rewrite the delta form of the adjoint recurrence relation as follows.

$$\Delta\Lambda = \left[\frac{\partial R(u^{k-1})}{\partial u^{k-1}}\right]^T \psi^k \tag{68}$$

It is important to note that, as in the tangent mode, these linear systems must be solved as the exact dual of the primal solve, as we are attempting to transpose exactly the primal solve. This secondary adjoint variable will then be used in the sensitivity equation (52) and we obtain:

$$\frac{dJ}{dD} = \frac{\partial L}{\partial D} + \psi^{nT}\frac{\partial R(u^{n-1})}{\partial D} + \psi^{n-1T}\frac{\partial R(u^{n-2})}{\partial D} + ... + \psi^{1T}\frac{\partial R(u^0)}{\partial D} \tag{69}$$

V. Verification

In this section we use the tangent and adjoint computed sensitivities of the lift objective functional as proxies for the behavior of the tangent and adjoint systems, and as a means to examine their behavior, convergence, and verify their correctness. The design variables are two Hicks-Henne bump functions located at one third and two thirds of the chord length respectively. The mesh sensitivities were calculated with the spring analogy outlined previously. The mesh is unstructured and consists of 4212 triangular elements shown in Figure 1. All verification cases were run with $Mach = .6$ and $\alpha = 1^o$.

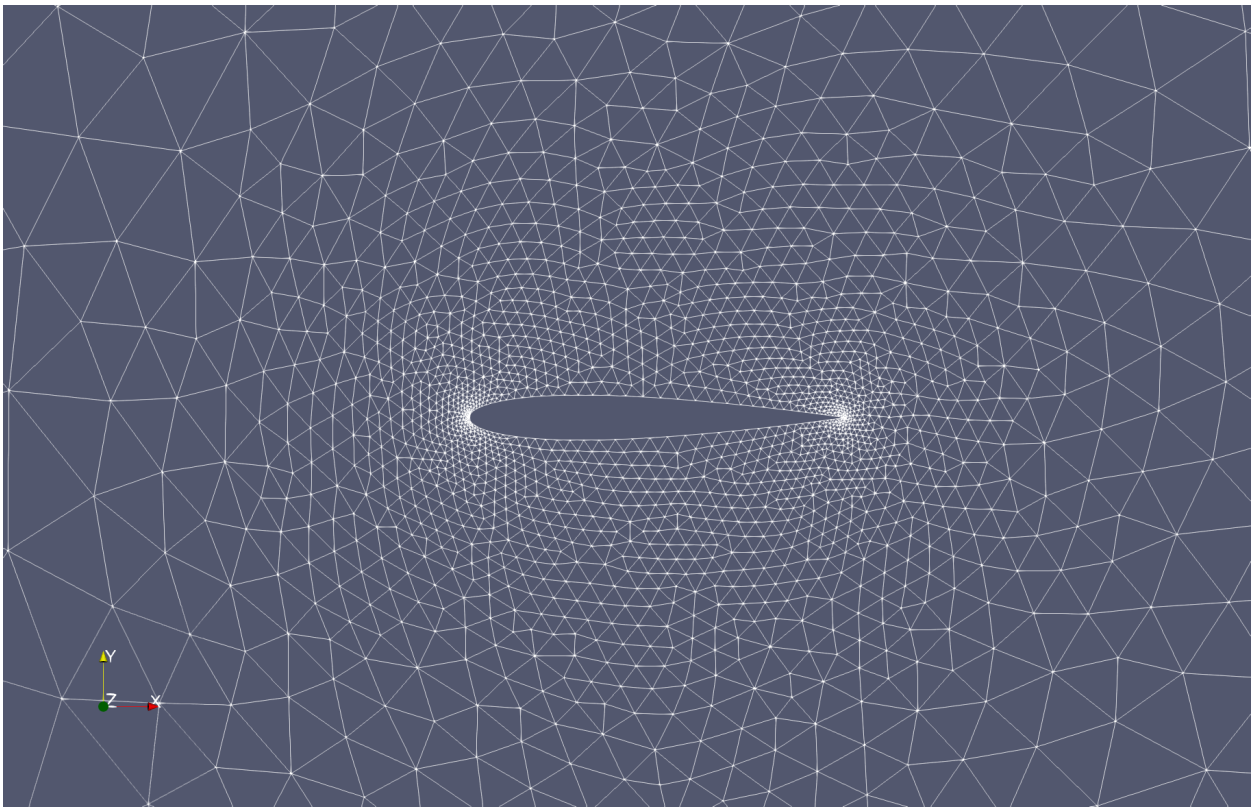


Figure 1. Computational mesh for NACA0012 airfoil

V.A. Tangent Verification

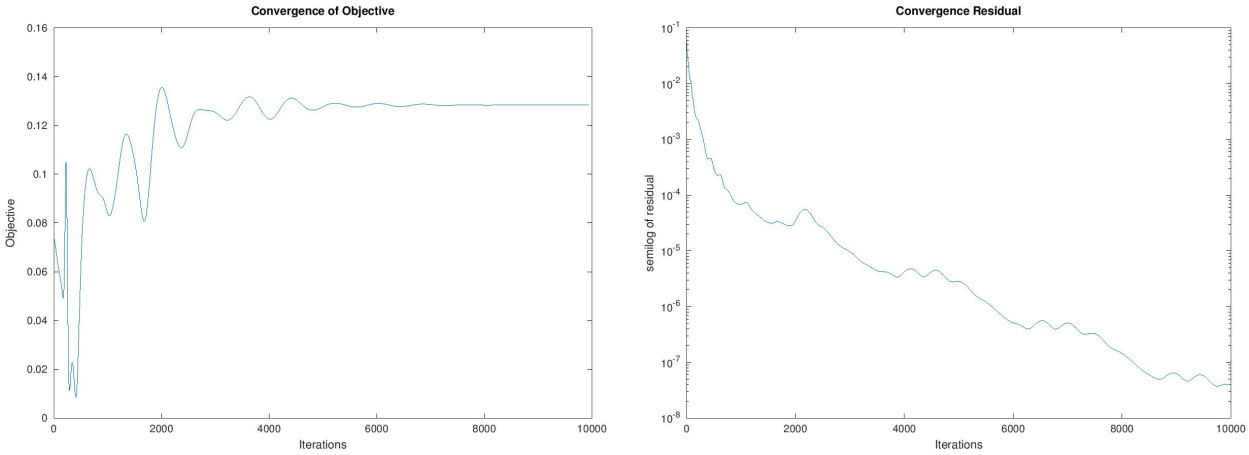
For the pseudo-time accurate tangent system verification we compare the complex step computed sensitivities to those provided by the tangent system. Since the pseudo-time accurate tangent formulation is designed to compute the exact derivative we expect to see exact correspondence between the two methods, i.e. differences near the order of machine zero.

V.A.1. Forward Euler Pseudo-Time Evolution

Figure 2 shows convergence behavior of the primal system residual and the convergence of the objective functional to its final value. Figure 3 shows verification of the implementation of the pseudo-time accurate tangent. This figure shows the difference between the pseudo-time accurate tangent and complex sensitivities is on the order of machine zero regardless of the residual value at the given pseudo-time step. The error is of order 1e-15 for both design variables, and as such it is clear this method is verified.

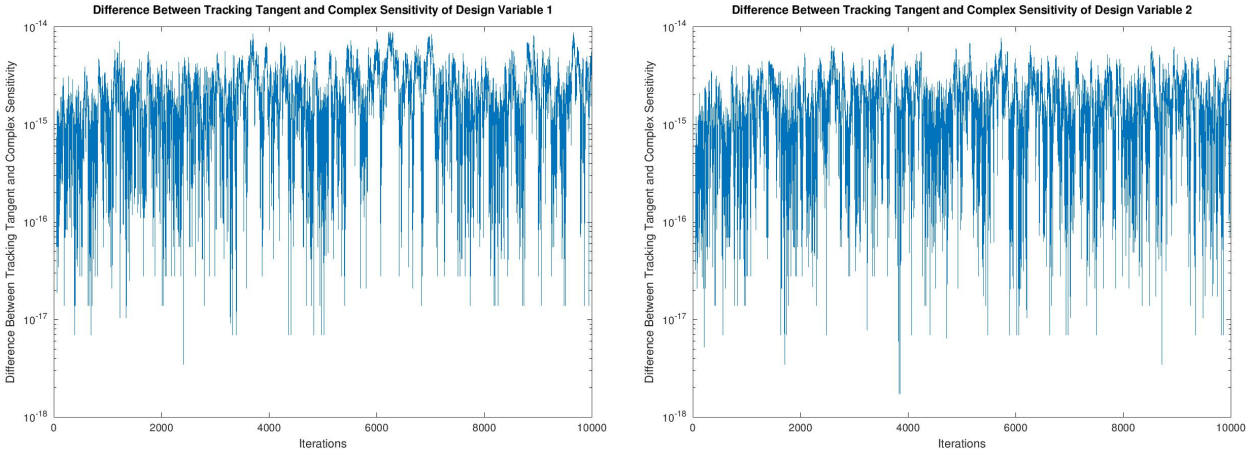
V.A.2. Newton-Chord Method

The verification in this section is done by implementing a Newton-Chord method restarted at the 26th iteration. This verification is accomplished by starting the primal problem at initial conditions and running a quasi-Newton algorithm for 25 steps, at which point we freeze the Jacobian and continue solving the primal problem with the frozen Jacobian. The complex-step finite-difference sensitivities were calculated by restarting at the 26th iteration with the frozen Jacobian and introducing a complex perturbation to the design variables, therefore ensuring the complex perturbation does not affect the frozen Jacobian. To run the pseudo-time accurate tangent, the tangent problem was restarted at the 26th iteration. To parallel the Newton-Chord solution the preconditioner matrix on the left hand side of equation (42) is the frozen Jacobian matrix, and the terms on the right hand side are the linearization of the spatial residual operator, which changes through pseudo-time. As we have taken the exact derivative of the process outlined in the primal solution process, we expect exact correspondence between the complex step derivative and the pseudo-time



(a) Residual (b) Objective Functional

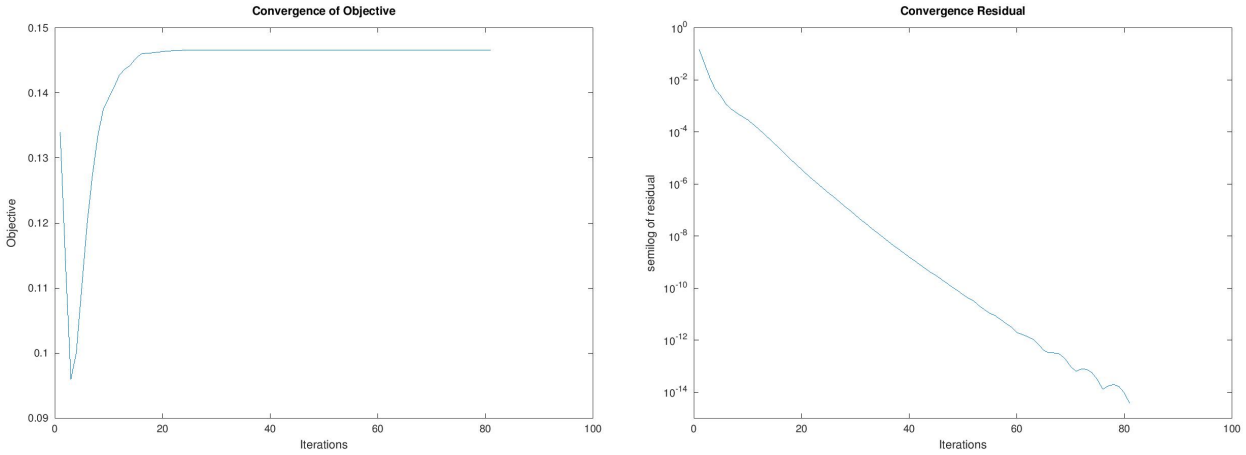
Figure 2. Convergence of spatial residual and objective function for forward Euler pseudo-time evolution



(a) Design Variable 1 (b) Design Variable 2

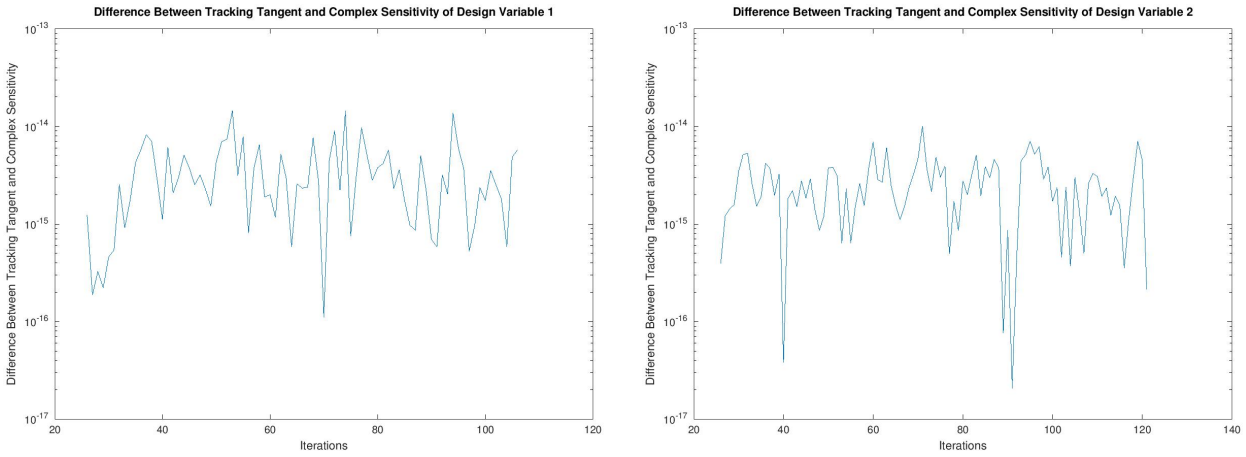
Figure 3. Difference between pseudo-time accurate tangent sensitivities and complex sensitivities for forward Euler pseudo-time evolution

accurate tangent sensitivity. The functional and residual behavior are depicted in Figure 4. Figure 5 plots the difference between the complex and pseudo-time accurate adjoint method sensitivities at each pseudo time step, and confirms that we obtain sensitivities which agree to machine precision.



(a) Functional (b) Residual

Figure 4. Convergence of objective function and residual for quasi-Newton method



(a) Design Variable 1 (b) Design Variable 2

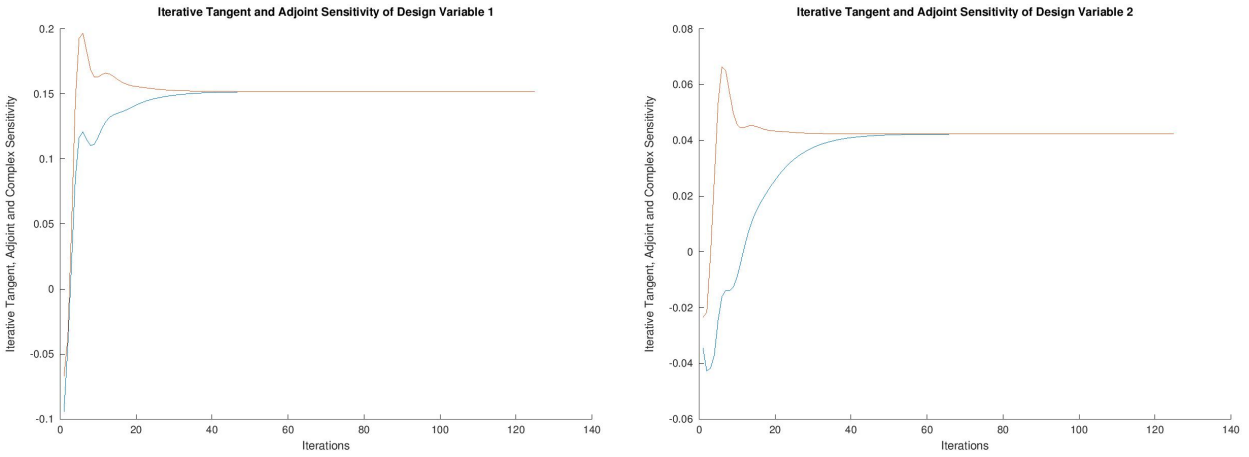
Figure 5. Difference between pseudo-time accurate tangent computed sensitivities and complex sensitivities for quasi-Newton method at each pseudo-time step

V.B. Adjoint Verification

Because the adjoint computed sensitivities are computed through reverse integration in pseudo-time, a comparison similar to the graphical representation in the tangent section is not practical. As such, we have provided Table 1 showing the difference between the complex and adjoint provided sensitivities on truncated runs for verification purposes. We can see that regardless of the convergence of the residual, which we marked by reporting the L_2-norm of the spatial residual in the final column of the table, we obtain exact correspondence between the sensitivities provided by the adjoint and complex-step methods. We also show comparisons of the convergence of the pseudo-time accurate tangent and adjoint sensitivities to their final values. It is important to note that the index of the tangent runs forward in pseudo-time and the time increment increases from left to right; in contrast, the adjoint runs backward in pseudo-time, and as such the time increment decreases from left to right. This is done to aid comparison between the two methods and contrast their respective convergence behaviors in Figure 6, which shows that the two converge to the final value of the sensitivity at comparable rates for the quasi-Newton method.

Scheme	Steps	Complex Sensitivity	Adjoint Sensitivity	$\|\|Residual\|\|_2$
RK5	50	2.582417731112833	2.582417731112833	0.2315064463096067E-03
RK5	200	1.8735681698763	1.8735681698761	0.1941186088514834E-03
RK5	25000	-36.6039235232360	-36.6039235232388	0.7237556735513925E-04
Newton	75	0.151782985164694	0.151782985164693	0.2539747805930525E-05

Table 1. Comparison of adjoint and complex-step computed sensitivities



(a) Design Variable 1

(b) Design Variable 2

Figure 6. Comparison of convergence through iteration space of tangent (in blue) and adjoint (in red) computed sensitivities

VI. Results

In this section we show several results as part of the comparison between the pseudo-time accurate adjoint and the classical – or steady-state – adjoint. First, we look at a well converging simulation that is truncated before the residual reaches machine zero, and compare the sensitivities from the steady-state adjoint linearized about the final state to those from a pseudo-time accurate adjoint. Then we consider two cases which fail to converge fully and show the effect of increasing the averaging window of the objective functional on the pseudo-time accurate adjoint derived sensitivities. Next, we show the sensitivities of the steady-state adjoint linearized about the final state of the simulation as well as about the averaged state, and compare these sensitivities to those provided by the pseudo-time accurate adjoint using the same functional averaging window as that used to generate the average state. Finally, we also consider the effect of averaging the computed sensitivities.

The objective functional here for a simulation run for n pseudo-time steps is either a windowed combination of the coefficients of lift and drag, in which case the lift and drag coefficients are averaged over the last n_{window} pseudo-time steps, or an instantaneous combination at the final state, which can be calculated by setting $n_{window} = 1$. The windowed objective function is written below, where ω is the weight, in this work $\omega = 8$.

$$L = \frac{1}{n_{window}} \sum_{i=n-n_{window}+1}^{n} C_{L_i} + \omega C_{D_i} \tag{70}$$

The comparisons in this section will compare the values of the sensitivity vectors computed from the different methods to one another, as well as the directions of those sensitivity vectors. The sensitivity vector direction is typically used in the line search methods found in most gradient-based optimizers, and errors in the search direction can lead to suboptimal designs or outright failure of the design process. All cases in this section were run with the five stage Runge-Kutta scheme.

VI.A. Application of the Pseudo-Time Accurate Adjoint to a Truncated Simulation

This section shows that for a well converging primal problem which is truncated before deep convergence and which has a well converging adjoint system, we see a significant difference in the values between the pseudo-time accurate adjoint and the steady-state adjoint computed sensitivities. For this case we solved the primal problem for a NACA0012 airfoil on the mesh shown for verification in Figure 1, at $Mach = .85$ and $\alpha = 3$. The simulation is terminated when the L_2-norm of the residual is less than 1e-6. The resulting density field is shown in Figure 7.

We then solved the steady-state adjoint problem; the convergence plots of the nonlinear residual in the primal problem and the linear residual in the adjoint problem are shown in Figure 8. The design variables consisted of 4 Hicks-Henne bump functions equally spaced at one-fifth, two-fifths, three-fifths and four-fifths of the chord length of the airfoil. The mesh sensitivities were calculated from the spring analogy method.

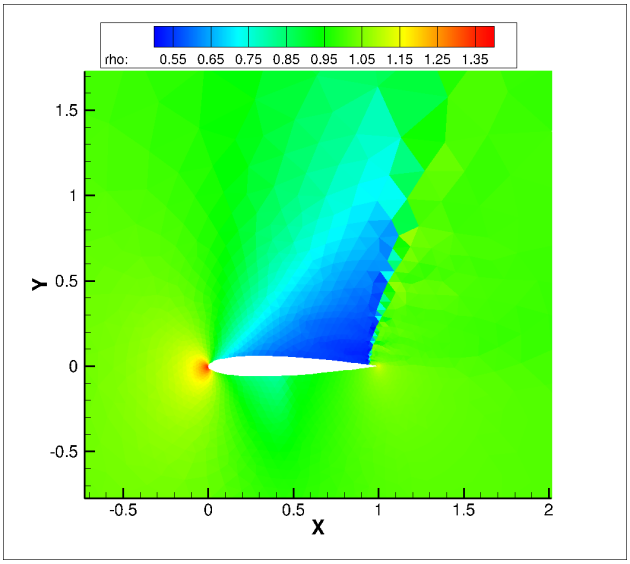
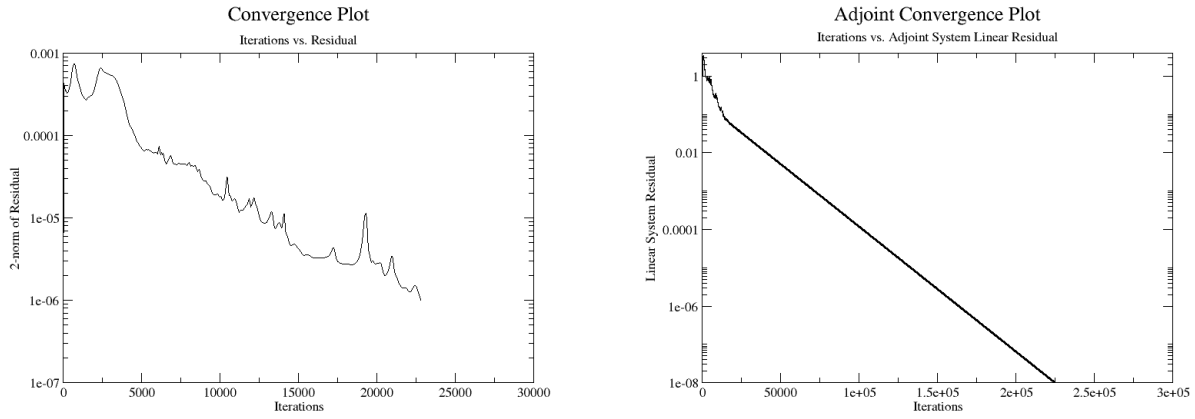


Figure 7. Density field for NACA0012 airfoil in $Mach = .85$, $\alpha = 3$



(a) Nonlinear Residual Convergence

(b) Adjoint Linear Convergence

Figure 8. Residual and steady-state adjoint convergence for truncated simulation

We then compare the steady-state adjoint computed sensitivities to the pseudo-time accurate adjoint computed sensitivitites, and evaluate the percentage difference and angle between the two sensitivity vectors. The angle θ is calculated as follows, with u and v being the sensitivity vectors:

$$a = \frac{u \cdot v}{\|u\| \, \|v\|} \tag{71}$$

Design Variable	Steady-State Value	Pseudo-Time Accurate Value	Percent Difference
1	-38.9168562768552	-34.07260014448312	14.2%
2	-58.8314866801896	-62.28918541979274	5.6%
3	-51.9374815863084	-70.15517817818902	26.0%
4	-69.3520141341324	-95.35571541527936	27.3%

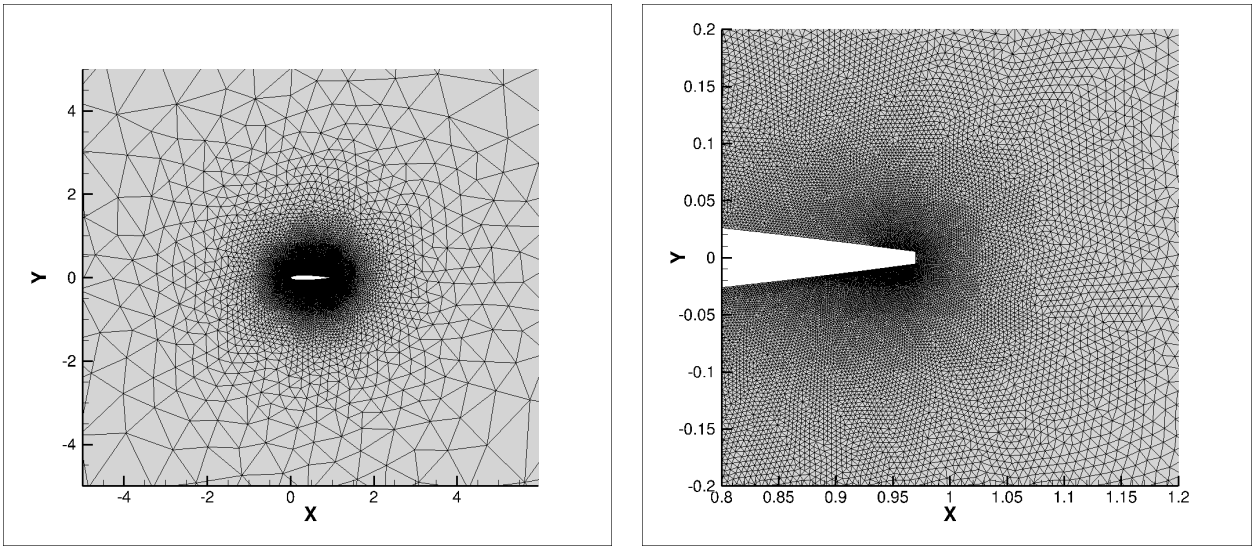
Table 2. Comparison of pseudo-time accurate adjoint and steady-state adjoint computed sensitivities for truncated primal simulation

$$\begin{aligned} \theta &= arccos(a), \theta \leq 180 \\ \theta &= 360 - arccos(a), \theta > 180 \end{aligned} \tag{72}$$

Using the values for the sensitivity vectors shown in Table 2 we obtain a value of the angle between the two sensitivity vectors of $\theta = 8.652^o$. Between the the angle value and difference in the magnitude values we can see significant discrepancies between the results of these two methods of sensitivity computation, and as such conclude even in cases of well converging primal problems that we gain significantly from the use of the pseudo-time accurate adjoint as we see a significant difference between the steady-state adjoint sensitivities and the exact sensitivities of the partially converged solution as obtained by the pseudo-time accurate adjoint formulation.

VI.B. Application of the Pseudo-Time Accurate Adjoint to Non-converging Primal Problem

A primal problem for a cutoff airfoil (NACA0012 with a blunt trailing edge at 97% of the chord length) was solved using a fine mesh with 64398 trianglular elements shown in Figure 13 in order to devise a non-convergent steady-state problem.



(a) Mesh (b) Trailing Edge

Figure 9. Fine mesh for NACA0012 airfoil cut off at 97% chord length

VI.B.1. Subsonic Case

The flow conditions for the test cases in this section are given by $Mach = .3$ and $\alpha = 3$. Figure 10 shows the convergence of the residual, lift coefficient and drag coefficient through 60000 iterations, at which point the residual stopped decreasing when it was run out to 80000 iterations.

Figure 11 shows the stagnation pressure values for the final state and the averaged state. The figure shows that for the instantaneous final state there is shedding at the trailing edge of the airfoil, and the oscillations in the flow field are far stronger than those at the averaged state. Figure 12 shows the behavior

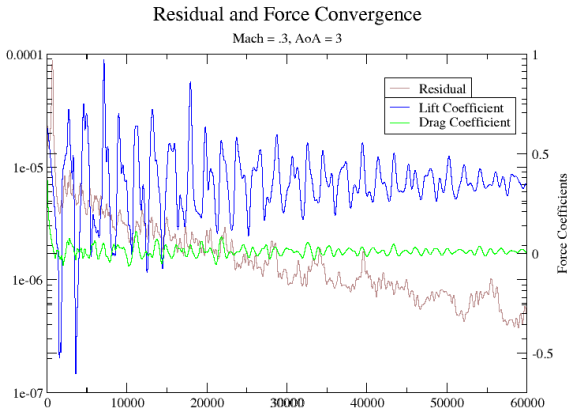
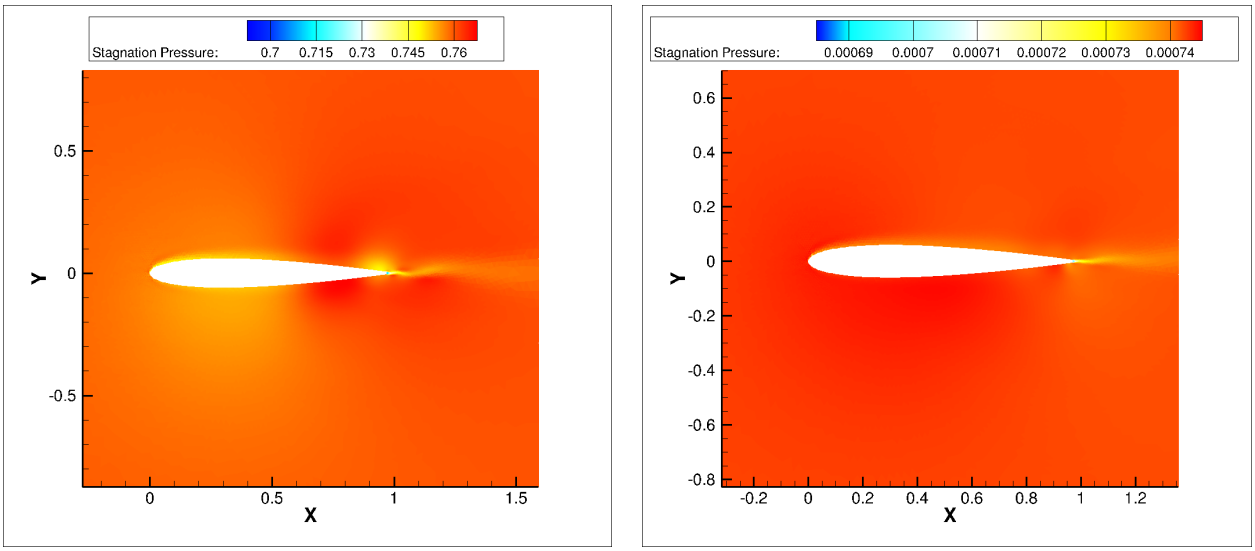


Figure 10. Primal convergence for $Mach = .3$, $\alpha = 3$



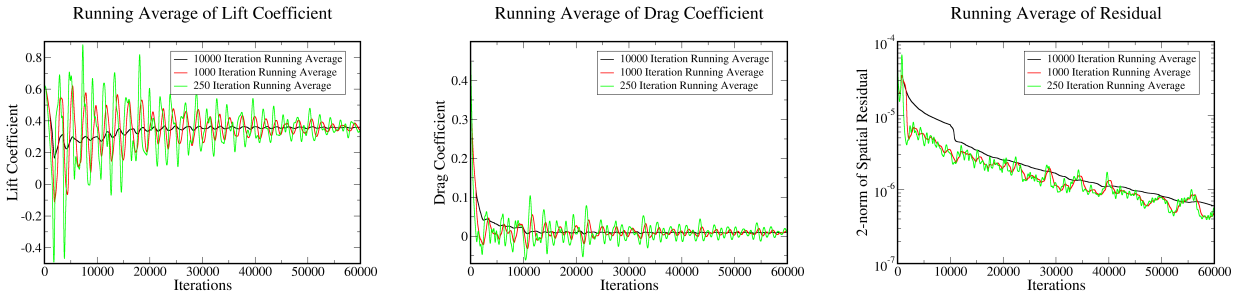
(a) Final State

(b) Average State

Figure 11. Stagnation pressure for final state and primal state

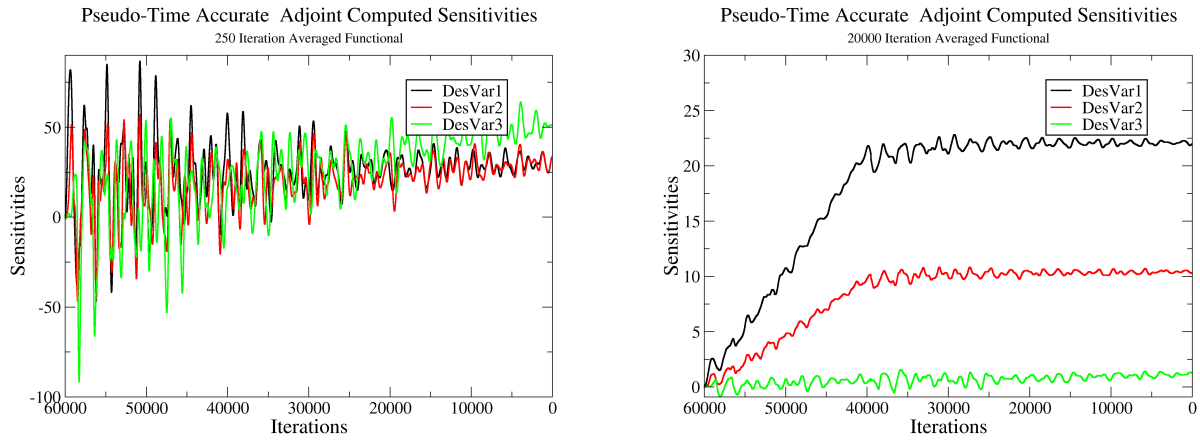
of the coefficient of lift and drag over different averaging windows. It is clear that increasing the averaging window damps the oscillations and gives a more convergent behavior of the force coefficients, as the two smaller windows show large oscillations, with the largest window showing negligible oscillations. We see that averaging the force coefficients leads to well behaved values over long intervals, which is a common practice when evaluating CFD results for engineering applications.

Figure 13 shows the convergence of the sensitivities back through pseudo-time on reversed x-axes when computed by the pseudo-time accurate adjoint. It is clear that by increasing the size of the objective functional averaging window, the magnitude of the sensitivities grow smaller. It is also clear that the sensitivities provided by the adjoint for an objective functional averaged over a long window approach the final value after the adjoint steps back through the averaging window, as opposed to the adjoint for a smaller averaging window objective functional which does not show this behavior. We can also see that for the longer averaging window the oscillations about this fully pseudo-time integrated value in the sensitivities are smaller. In order to evaluate the use of the computed sensitivities for a gradient-based optimizer, we evaluate the evolution of the angle between the partially (backwards-in-time) integrated adjoint sensitivity vector to the fully integrated adjoint sensitivity vector as the adjoint steps back through pseudo-time. This is done to measure the feasibility of early termination of the time integration. In fact for the longer window



(a) Lift Coefficient (b) Drag Coefficient (c) Residual

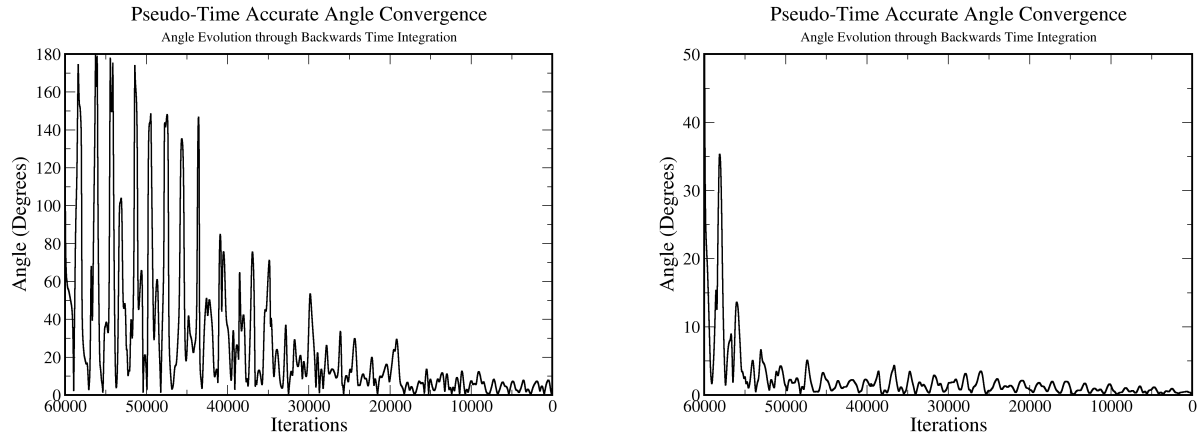
Figure 12. Behavior of primal for different averaging windows



(a) 250 Iteration Window (b) 20000 Iteration Window

Figure 13. Pseudo-Time accurate adjoint sensitivities for varying objective windows

objective function, the angle between the partially integrated versus the fully integrated adjoint computed sensitivities is well behaved as is demonstrated in Figure 14. We use the fully pseudo-time integrated adjoint sensitivities as the comparison point as they represent the exact sensitivities and correspond to the complex-step finite-difference computed sensitivities. We can see that when we have integrated to roughly twice the averaging window for well behaved cases, we are very close to the exact sensitivities.

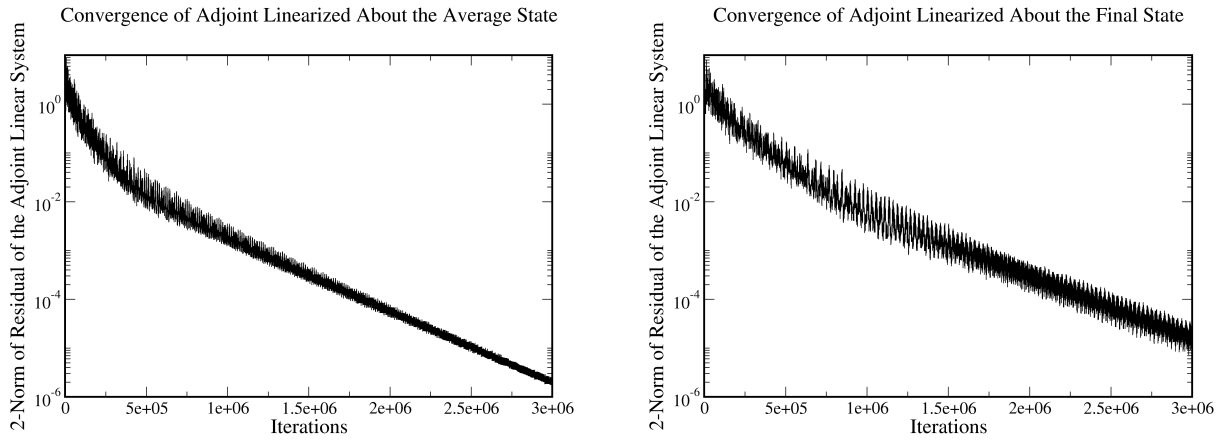


(a) 250 Iteration Window

(b) 20000 Iteration Window

Figure 14. Angle between partially time-integrated and fully time-integrated sensitivities over iteration space to final sensitivity

We then compare how pseudo-time accurate adjoint computed sensitivities compare to the values obtained by the steady-state adjoint linearized about different states. We examine the pseudo-time accurate adjoint using as the objective functional the average objective functional over the last 20000 iterations. We compare this to a steady-state adjoint evaluation at both the final state and an averaged state produced by averaging the conservative variables over the last 20000 iterations, the same window as the functional driving the pseudo-time accurate adjoint. Figure 15 depicts the plots of the convergence of the two steady-state adjoint systems. We note that it takes many iterations to converge these systems, as the adjoint linearized about a partially converged state has been found to be hard to converge.[7] The pseudo-time accurate adjoint does not suffer from this problem as it transposes the derivatives of the operations used in the primal problem, and therefore does not require more advanced solver technology than the primal problem.



(a) Average State

(b) Final State

Figure 15. Steady-State adjoint convergence for $Mach = .3, \alpha = 3$

Table 3 provides the computed sensitivity vectors and from this information we evaluate the angle between

Design Variable	Steady-State Value (Final State)	Pseudo-Time Accurate Value	Percent Difference
1	21.3443558908559	21.96809430870143	2.83%
2	9.15634414104837	10.26553487597339	10.8%
3	-2.88279274401136	1.239597303802120	332.56%

Table 3. Pseudo-Time accurate adjoint and steady-state sensitivities computed at final state for non-converging primal problem

the two sensitivity vectors. The first vector is computed by the steady-state adjoint linearized about the final state and the second is computed by the pseudo-time accurate adjoint. In this case we obtain an angle between the two vectors of $\theta = 10.167^o$. This is a significant angle difference, and could cause problems for optimization with a gradient-based optimizer.

Design Variable	Steady-State Value (Avg State)	Pseudo-Time Accurate Value	Percent Difference
1	22.6224747115315	21.96809430870143	2.97%
2	10.5354461565917	10.26553487597339	2.63%
3	0.430506286872907	1.239597303802120	65.27%

Table 4. Pseudo-Time accurate adjoint and steady-state sensitivities computed at averaged state for non-converging primal problem

Table 4 compares two sensitivity vectors, calculated by the steady-state adjoint about the averaged state and the pseudo-time accurate adjoint respectively. In this case, the angle between these vectors is $\theta = 1.9396^o$. We can see that when we have entered limit cycle oscillations that averaging the state greatly improves the performance of the steady-state adjoint. However, although the angle is relatively small in this case the magnitudes of the individual sensitivities still differ substantially.

VI.B.2. Transonic Case

In this section the flow conditions are changed to $Mach = .7$, $\alpha = 2^o$. Figure 16 shows the convergence of the residual, lift coefficient and drag coefficient, and the plots extend far into the limit-cycle oscillation region.

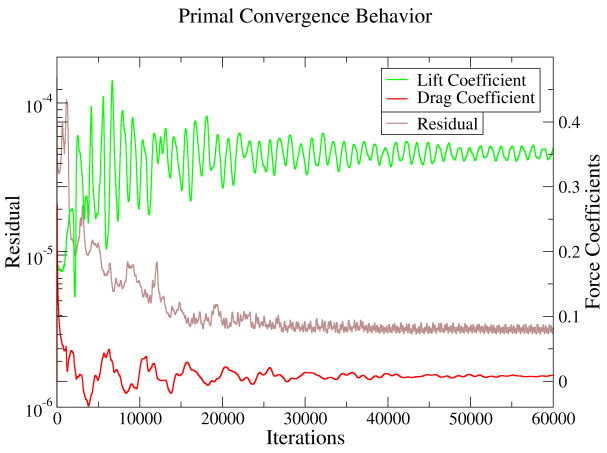
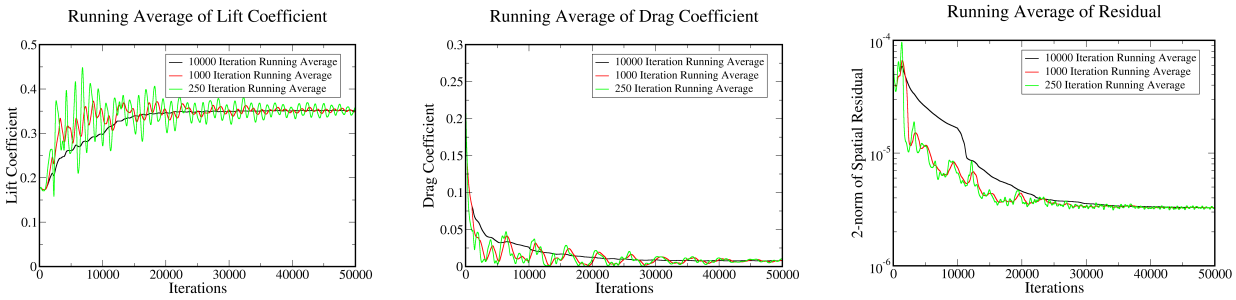


Figure 16. Primal convergence for cut-off NACA0012 airfoil at $Mach = .7$, $\alpha = 2^o$

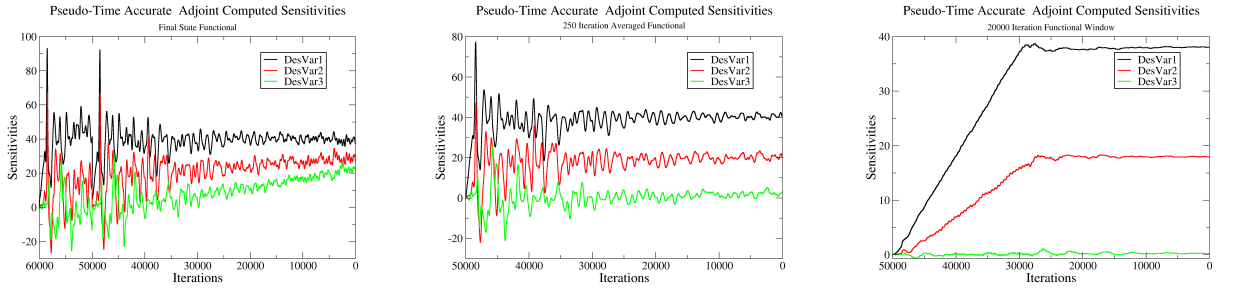
Figure 17 shows the behavior of the residual and the coefficients of lift and drag over different averaging windows. It is clear that the convergence behavior with averaging is better here than in the subsonic case, and that for a 10000 iteration averaging window the average converges well. In Figure 18, we show the convergence of the sensitivities back through time on reversed x-axes when computed by the pseudo-time accurate adjoint. As in the subsonic case increasing the size of the averaging window for the objective



(a) Lift Coefficient (b) Drag Coefficient (c) Residual

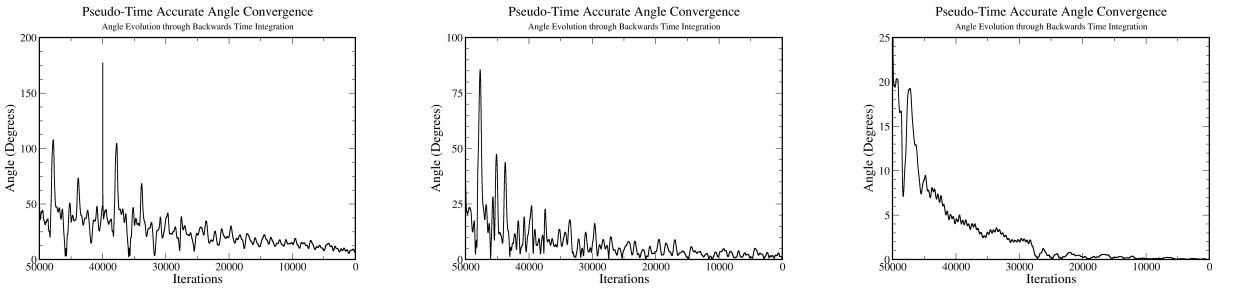
Figure 17. Behavior of primal problem for different averaging windows for $Mach = .7$, $\alpha = 2^o$

functional gives smaller magnitude sensitivities. Additionally, by extending the simulation far into the region of limit cycle oscillations, we have very small oscillations in the values of the sensitivities. As before, we then look to the angle behavior in order to get a better sense of how truncating the backwards pseudo-time integration may affect the line-search of a gradient-based optimizer.



(a) Final State (b) 250 Iteration Window (c) 20000 Iteration Window

Figure 18. Pseudo-Time accurate adjoint sensitivities for varying objective windows for $Mach = .7$, $\alpha = 2^o$



(a) Final State (b) 250 Iteration Window (c) 20000 Iteration Window

Figure 19. Angle convergence over iteration space to final sensitivity for $Mach = .7$, $\alpha = 2^o$

Figure 19 depicts the angle between the partially and fully integrated sensitivities for different objective functional average windows. It is clear that as we increase the functional averaging window we see better angular behavior when comparing the partially pseudo-time integrated sensitivity vector to the fully pseudo-time integrated sensitivity vector. Looking at the angle between the partially time integrated and fully time integrated sensitivity vectors as we exit one and half times the averaging window, we see angles on the order of $.1^o$ or $.01^o$ for the largest functional averaging window.

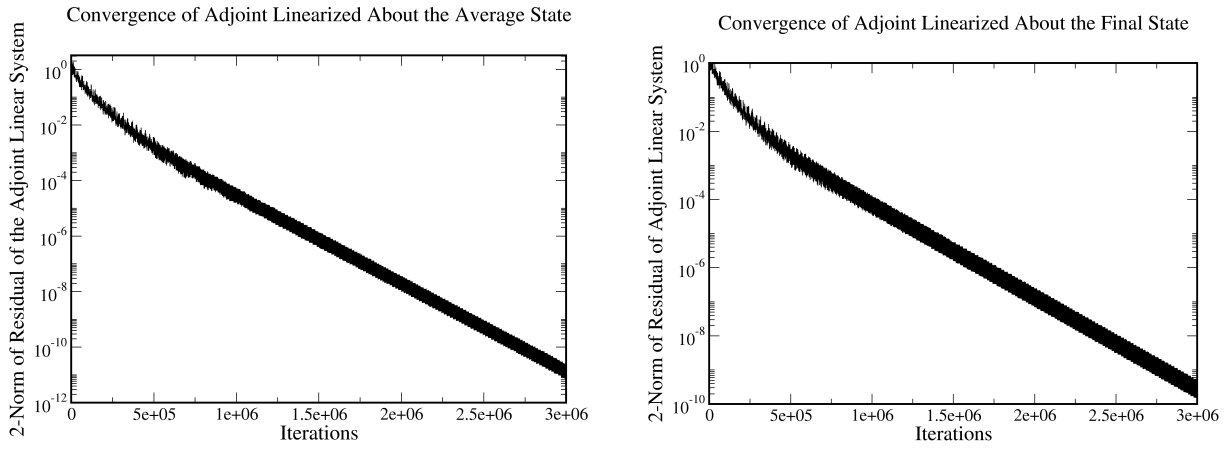
We then compare the pseudo-time accurate adjoint sensitivities to those provided by the steady-state adjoint. Figure 20 shows the convergence of the linear residual of the steady-state adjoint systems, linearized about the final state and averaged state respectively. These plots show, as before, that computing these

adjoint systems is very expensive, more expensive than the pseudo-time accurate adjoint calculation.

Design Variable	Steady-State Value (Final State)	Pseudo-Time Accurate Value	Percent Difference
1	44.5805974889098	38.03996816573513	17.19%
2	14.0224144011606	17.92859539270058	21.79%
3	-1.27443407446669	0.2080948770865852	712.43%

Table 5. Pseudo-Time accurate adjoint and steady-state sensitivities for non-converging primal problem $Mach = .7$, $\alpha = 2^o$

As part of our comparison between the pseudo-time accurate adjoint computed sensitivities and the final state steady-state adjoint computed sensitivities, we compute the angle between the vectors from the values in Table 5 and we obtain an angle of $\theta = 7.98972^o$. As before the sign of the sensitivity of the third design variable in the steady-state adjoint sensitivity vector is negative as opposed to the pseudo-time accurate sensitivity vector which has positive sign. This being an undesirable result we move onto the steady-state adjoint linearized about the averaged state, created by averaging the last 20000 iterations.



(a) Average State

(b) Final State

Figure 20. Steady-State adjoint convergence for $Mach = .7$, $\alpha = 2^o$

Design Variable	Steady-State Value (Avg State)	Pseudo-Time Accurate Value	Percent Difference
1	43.3642585762660	38.03996816573513	14.00%
2	16.9218981680917	17.92859539270058	5.62%
3	-9.717943674789181E-002	0.2080948770865852	146.70%

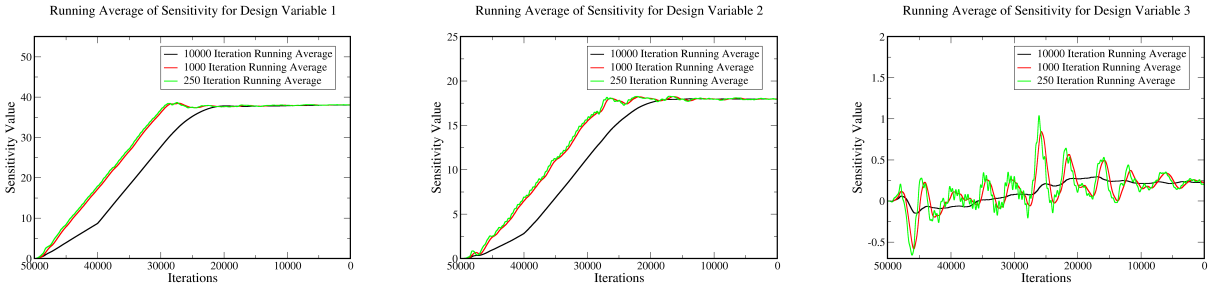
Table 6. Pseudo-Time accurate adjoint and steady-state sensitivities for non-converging primal problem for $Mach = .7$, $\alpha = 2^o$

Table 6 shows a comparison between the pseudo-time accurate adjoint computed sensitivities and the averaged state steady-state adjoint computed sensitivities. From these values we obtain a value of $\theta = 3.93855^o$, and we once more see opposing signs for the sensitivity of the third design variable.

Given the significant magnitude differences and the angle between the steady-state adjoint computed sensitivity vectors and the pseudo-time accurate adjoint computed sensitivity vectors, we seek additional techniques for improving the accuracy of the partially integrated pseudo-time accurate adjoint. Since the pseudo-time accurate adjoint provided sensitivities are well behaved except for oscillations about the mean after the backwards-in-time integration is complete through the functional averaging window, we look at the effects of averaging the sensitivity values themselves for the pseudo-time accurate adjoint problem when driven by a large functional averaging window.

Figure 21 depicts running averages of the sensitivities computed by the pseudo-time accurate adjoint for the largest objective functional averaging window. The instantaneous values of the sensitivities can be found for comparison in Figure 18c. Referencing the value of the running average of the sensitivity when

the sensitivity averaging window is outside the functional averaging window shows a dampening of the oscillations. In this case, where we average the functional over the last 20000 iterations and the sensitivity over the last 10000, we look at the value 30000 iterations through the backwards time integration and compare it to the final sensitivity calculation for the non averaged – or instantaneous – sensitivities. We see that the averaging over the largest window clearly damps out the oscillations, but for smaller windows it is not as effective, especially in the averaging of the highly oscillatory third design variable sensitivity. Then comparing the partially integrated and averaged sensitivity vector to the fully time integrated and instantaneous sensitivity vector in Table 7 gives an angle of $.34481^o$ between the two vectors. This value is an order of magnitude smaller than the angle obtained when comparing the fully time integrated pseudo-time accurate adjoint computed sensitivities to the steady-state adjoint computed sensitivities when linearized about the averaged state. Additionally, we can see that the magnitude differences are far smaller. Further research was done into smaller sensitivity averaging windows, but these smaller windows led to more oscillatory average gradients. The results show that the use of a sensitivity averaging window of approximately 50% of the functional averaging window is a reasonable choice for these test cases.



(a) Design Variable 1 (b) Design Variable 2 (c) Design Variable 3

Figure 21. Effect of averaging sensitivities for pseudo-time accurate adjoint for $Mach = .7$, $\alpha = 2^o$

Design Variable	10000 Iteration Average Value	Instantaneous Value	Percent Difference
1	37.815963808090757	38.03996816573513	.5%
2	17.554182903957241	17.92859539270058	2%
3	.26808010969592244	0.2080948770865852	28%

Table 7. Comparison of partially integrated averaged pseudo-time accurate averaged sensitivities (computed at iteration 30000) to fully backwards integrated instantaneous pseudo-time accurate sensitivities for $Mach = .7$, $\alpha = 2^o$

VII. Conclusions

We developed pseudo-time accurate formulations of the tangent and adjoint systems to provide exact sensitivities of the primal problem solution process. We then verified that these sensitivities correspond to the complex-step finite-difference provided sensitivities to machine precision. We applied this formulation first to a well converging, but truncated, transonic simulation and showed significant differences between the steady-state computed sensitivities and the pseudo-time accurate adjoint computed sensitivities. We then applied the pseudo-time accurate adjoint to a subsonic non-converging simulation and showed that by averaging the objective function in pseudo-time, the pseudo-time accurate adjoint provides well behaved sensitivities that differ notably from the steady-state adjoint computed sensitivities. Subsequently, the pseudo-time accurate adjoint was applied to a non-converging transonic simulation, which showed larger differences between the pseudo-time accurate adjoint computed sensitivities and the steady-state adjoint computed sensitivities. This case also showed well behaved sensitivities for large functional averaging windows. Finally, an investigation was done into averaging the sensitivities with the purpose of investigating the feasibility of partial backwards-in-time integration of the pseudo-time accurate adjoint. It was shown that by selecting a suitably large objective functional window, one that has a statistically converged objective functional, a sensitivity averaging window of half the objective functional window is sufficient for good agreement between the fully-integrated sensitivities and the partially integrated averaged sensitivities in terms of direction and

magnitude of the sensitivity vectors.

VIII. Acknowledgments

This work was supported in part by NASA Grant NNX16AT23H and the NASA Graduate Aeronautics Scholars Program. Computing time was provided by NASA Advanced Supercomputing on Pleiades and Electra. The first author would like to thank Michael Aftosmis and Marian Nemec for their input and advice.

References

[1]P. E. Gill, W. Murray, M. A. Saunders, Elizabeth Wong.SNOPT 7.7 User's Manual. CCoM Technical Report 18-1, Center for Computational Mathematics, University of California, San Diego.

[2]P. E. Gill, W. Murray and M. A. Saunders. SNOPT: An SQP algorithm for large-scale constrained optimization. SIAM Review 47 (2005), 99-131.

[3]Adams, B.M., Bauman, L.E., Bohnhoff, W.J., Dalbey, K.R., Ebeida, M.S., Eddy, J.P., Eldred, M.S., Hough, P.D., Hu, K.T., Jakeman, J.D., Stephens, J.A., Swiler, L.P., Vigil, D.M., and Wildey, T.M., "Dakota, A Multilevel Parallel Object-Oriented Framework for Design Optimization, Parameter Estimation, Uncertainty Quantification, and Sensitivity Analysis: Version 6.0 User's Manual," Sandia Technical Report SAND2014-4633, July 2014. Updated November 2015

[4]Siva Nadarajah. The Discrete Adjoint Approach to Aerodynamic Shape Optimization. Ph.d. dissertation, Department of Aeronautics and Astronautics, Stanford University, Stanford, CA, January 2003

[5]Nemec, M. and Aftosmis, M. J., "Toward Automatic Verification of Goal-Oriented Flow Simulations," Tech. Rep. TM-2014-218386, NASA, 2014

[6]David A. Venditti , David L. Darmofal, Anisotropic grid adaptation for functional outputs: application to two-dimensional viscous flows, Journal of Computational Physics, v.187 n.1, p.22-46, May 2003. https://doi.org/10.1016/S0021-9991(03)00074-3

[7]Krakos, J. A., and Darmofal, D. L. (2010). Effect of small-scale output unsteadiness on adjoint-based sensitivity. AIAA Journal, 48(11), 2611-2623. doi:10.2514/1.J050412

[8]Gill, P. E., Murray, W., and Saunders, M. A., "SNOPT: An SQP algorithm for large-scale constrained optimization," SIAM journal on optimization, Vol. 12, No. 4, 2002, pp. 979–100

[9]Krakos, J. A., Wang, Q., Hall, S. R., and Darmofal, D. L. (2012). Sensitivity analysis of limit cycle oscillations. Journal of Computational Physics, 231(8), 3228-3245. doi:10.1016/j.jcp.2012.01.001

[10]A. Mishra, D. Mavriplis and J. Sitaraman, "Multipoint Time-Dependent Aero-elastic Adjoint-based Aerodynamic Shape Optimization of Helicopter Rotors" AHS Forum 71,Virginia Beach VA, May 2015, pp 828 - 844

[11]Mathias Luers, Max Sagebaum, Sebastian Mann, Jan Backhaus, David Grossmann, and Nicolas R. Gauger. "Adjoint-based Volumetric Shape Optimization of Turbine Blades", 2018 Multidisciplinary Analysis and Optimization Conference, AIAA AVIATION Forum, AIAA Paper 2018-3638, Atlanta, Georgia, 06/2018. https://doi.org/10.2514/6.2018-3638

[12]David A. Brown and Sivakumaran Nadarajah. "An Adaptive Constraint Tolerance Method for Optimization Algorithms Based on the Discrete Adjoint Method", 2018 AIAA/ASCE/AHS/ASC Structures, Structural Dynamics, and Materials Conference, AIAA SciTech Forum, AIAA Paper 2018-0414, Kissimmee, Florida, 01/2018, https://doi.org/10.2514/6.2018-0414

[13]Yukiko S. Shimizu and Krzysztof Fidkowski. "Output Error Estimation for Chaotic Flows", 46th AIAA Fluid Dynamics Conference, AIAA AVIATION Forum, AIAA Paper, 2016-3806, Washington D.C., 06/2016. https://doi.org/10.2514/6.2016-3806

[14]Yukiko S. Shimizu and Krzysztof Fidkowski. "Output-Based Error Estimation for Chaotic Flows Using Reduced-Order Modeling", 2018 AIAA Aerospace Sciences Meeting, AIAA SciTech Forum, (AIAA Paper 2018-0826), Kissimmee, Florida, 01/2018. https://doi.org/10.2514/6.2018-0826

[15]Dimitri Mavriplis. "Revisiting the Least-Squares Procedure for Gradient Reconstruction on Unstructured Meshes", 16th AIAA Computational Fluid Dynamics Conference, Fluid Dynamics and Co-located Conferences, AIAA Paper 2003-3986, Orlando, Florida, 06/2003. https://doi.org/10.2514/6.2003-3986

[16]LeVeque, Randall J., Numerical Methods for Conservation Laws", Birkhauser Verlag, 1992, p. 125.

[17]Roe, P., "Approximate Riemann Solvers, Parameter Vectors, and Difference Schemes," Journal of Computational Physics, Vol. 43-2, 1981, pp. 357–372.

[18]van Leer, B. (1982). Flux-vector splitting for the euler equations. Paper presented at the 8th International Conference on Numerical Methods in Fluid Dynamics, Aachen, West Germany; Germany; 28 June-2 July 1982.

[19]Karthik Mani and Dimitri Mavriplis. "An Unsteady Discrete Adjoint Formulation for Two-Dimensional Flow Problems with Deforming Meshes", 45th AIAA Aerospace Sciences Meeting and Exhibit, Aerospace Sciences Meetings, AIAA Paper 2007-60, Reno, Nevada, 01/2007. https://doi.org/10.2514/6.2007-60

[20]Jeroen Witteveen and Hester Bijl. "Explicit Mesh Deformation Using Inverse Distance Weighting Interpolation", 19th AIAA Computational Fluid Dynamics, Fluid Dynamics and Co-located Conferences, AIAA Paper 2009-3996, San Antonio, Texas, 06/2009. https://doi.org/10.2514/6.2009-3996

[21]D. Mavriplis, "Time dependent adjoint methods for single and multi-disciplinary problems" VKI Lecture notes, 38th Advanced Computational Fluid Dynamics Lecture Series of the von Karman Institute (VKI) for Fluid Dynamics, Rhode-Saint-Genese, Belgium, September 14-17, 2015.

[22]Eric Nielsen, James Lu, Michael Park, and David Darmofal. "An Exact Dual Adjoint Solution Method for Turbulent Flows

on Unstructured Grids", 41st Aerospace Sciences Meeting and Exhibit, Aerospace Sciences Meetings, AIAA Paper 2003-272, Reno, Nevada, 01/2009. https://doi.org/10.2514/6.2003-272

[23]Dimitri J. Mavriplis. "Multigrid Solution of the Discrete Adjoint for Optimization Problems on Unstructured Meshes", AIAA Journal, Vol. 44, No. 1 (2006), pp. 42-50.

IX. Appendix

In this appendix we show the derivation of the tangent and adjoint systems for a low storage Runge-Kutta scheme, this derivation is shown specifically for a 5-stage, but changing the number of stages leads to only trivial differences in the derivation. The notation used below uses k for the pseudo-time index, and l for the sub-stage index.

IX.A. Tangent Formulation

Low storage Runge-Kutta schemes have the following sub-stage evolution:

$$u^{k,l} = u^{k,0} + CFL\alpha^{l-1}\Delta t R(u^{k,l-1}) \tag{73}$$

with the end of the sub-stage time-stepping being governed as follows.

$$u^{k,0} = u^{k-1,5} \tag{74}$$

As before, the tangent formulation is straightforward and requires only the differentiation of the forward time-stepping, which provides the following expressions:

$$\frac{du^{k,l}}{dD} = \frac{du^{k,0}}{dD} + CFL\alpha^{l-1}\left[\left(\frac{\partial \Delta t}{\partial u^{k,0}}\frac{du^{k,0}}{dD} + \frac{\partial \Delta t}{\partial D}\right)R(u^{k,l-1}) + \Delta t\left(\frac{\partial R}{\partial D} + \frac{\partial R}{\partial u}\frac{du^{k,l-1}}{dD}\right)\right] \tag{75}$$

with the end of the substage time-stepping being differentiated as below.

$$\frac{du^{k,0}}{dD} = \frac{du^{k-1,5}}{dD} \tag{76}$$

IX.B. Adjoint Formulation

From the time-stepping above we can generate two new constraint equations with new constraint derivatives. The first constraint is modelled off the sub-stage evolution.

$$G^{k,l}(u^{k,l}, u^{k,l-1}, u^{k,0}, D) = u^{k,l} - u^{k,0} - CFL\alpha^{l}\Delta t R(u^{k,l-1}) = 0 \tag{77}$$

The first constraint has the following derivatives.

$$\begin{aligned}
\frac{\partial G^{k,l}}{\partial u^{k,l}} &= I \\
\frac{\partial G^{k,l}}{\partial u^{k,l-1}} &= -CFL\alpha^{l}\Delta t\frac{\partial R(u^{k,l-1})}{\partial u^{k,l-1}} \\
\frac{\partial G^{k,l}}{\partial u^{k,0}} &= -I - CFL\alpha^{l}\frac{\partial \Delta t}{\partial u^{k,0}}R(u^{k,l-1}) \\
\frac{\partial G^{k,l}}{\partial D} &= -CFL\alpha^{l}\left[\frac{\partial \Delta t}{\partial D}R + \Delta t\frac{\partial R}{\partial D}\right]
\end{aligned} \tag{78}$$

The second constraint is simpler:

$$G^{k,0}(u^{k,0}(D), u^{k-1,5}(D)) = u^{k,0} - u^{k-1,5} = 0 \tag{79}$$

the second constraint when differentiated provides as it derivatives the simple expressions below.

$$\begin{aligned}\frac{\partial G^{k,0}}{\partial u^{k,0}} &= I \\ \frac{\partial G^{k,0}}{\partial u^{k-1,5}} &= -I \\ \frac{\partial G^{k,0}}{\partial D} &= 0\end{aligned} \tag{80}$$

For the augmented objective we obtain the following equation, with constraints and adjoint variable vectors for each sub-stage.

$$\begin{aligned}&J(D, u^n, u^{n-1,5}, ..., u^{n-1,0}, ..., \Lambda^n, \Lambda^{n-1,5}, ..., \Lambda^{n-1,0}, ...) \\ &= \\ &L(u^n, D) + \Lambda^{nT} G^{n,0}(u^n(D), u^{n-1,5}(D), D) \\ &+ \Lambda^{n-1,5T} G^{n-1,5}(u^{n-1,5}(D), u^{n-1,4}(D), u^{n-1,0}(D), D) \\ &+ ... \\ &+ \Lambda^{n-1,1T} G^{n-1,1}(u^{n-1,1}(D), u^{n-1,0}(D), D) \\ &+ \Lambda^{n-1,0T} G^{n-1,0}(u^{n-1,0}(D), u^{n-2,5}(D), D) \\ &+ ... \\ &+ \Lambda^{0,1T} G^{0,1}(u^{0,1}(D), u^0(D), D)\end{aligned} \tag{81}$$

We take the derivatives with respect to the states to obtain the recurrence relations below, please note that this is for an objective function dependent only on the mesh and the final state.

$$\begin{aligned}\frac{\partial J}{\partial u^n} &= \frac{\partial L}{\partial u^n} + \Lambda^{nT} \frac{\partial G^n}{\partial u^n} = 0 \\ \frac{\partial J}{\partial u^{n-1,5}} &= \Lambda^{nT} \frac{\partial G^{n,0}}{\partial u^{n-1,5}} + \Lambda^{n-1,5T} \frac{\partial G^{n-1,5}}{\partial u^{n-1,5}} = 0 \\ \frac{\partial J}{\partial u^{n-1,4}} &= \Lambda^{n-1,5T} \frac{\partial G^{n-1,5}}{\partial u^{n-1,4}} + \Lambda^{n-1,4T} \frac{\partial G^{n-1,4}}{\partial u^{n-1,4}} = 0 \\ &... \\ \frac{\partial J}{\partial u^{n-1,0}} &= \Lambda^{n-1,5T} \frac{\partial G^{n-1,5}}{\partial u^{n-1,0}} + \Lambda^{n-1,4T} \frac{\partial G^{n-1,4}}{\partial u^{n-1,0}} + \Lambda^{n-1,3T} \frac{\partial G^{n-1,3}}{\partial u^{n-1,0}} \\ &\quad + \Lambda^{n-1,2T} \frac{\partial G^{n-1,2}}{\partial u^{n-1,0}} + \Lambda^{n-1,1T} \frac{\partial G^{n-1,1}}{\partial u^{n-1,0}} + \Lambda^{n-1,0T} \frac{\partial G^{n-1,0}}{\partial u^{n-1,0}} = 0 \\ &... \\ \frac{\partial J}{\partial u^1} &= \Lambda^{2T} \frac{\partial G^2}{\partial u^1} + \Lambda^{1T} \frac{\partial G^1}{\partial u^1} = 0\end{aligned} \tag{82}$$

Using the equation for the adjoint at the final pseudo-time step we calculate our initial adjoint variable in the backwards-in-time integration:

$$\left[\frac{\partial G^n}{\partial u^n}\right]^T \Lambda^n = -\left[\frac{\partial L}{\partial u^n}\right]^T \tag{83}$$

we then use the second adjoint equation listed to get the adjoint equation for $k = 1, 2, ..., n-1$ and $l = 5$ and obtain the following relation:

$$\left[\frac{\partial G^{k-1,5}}{\partial u^{k-1,5}}\right]^T \Lambda^{k-1,5} = -\left[\frac{\partial G^{k,0}}{\partial u^{k-1,5}}\right]^T \Lambda^{k,0} \tag{84}$$

using the other adjoint equations we get an expression for $k = 1, 2, ..., n-1$ and $l = 1, 2, 3, 4$.

$$\left[\frac{\partial G^{k,l-1}}{\partial u^{k,l-1}}\right]^T \Lambda^{k,l-1} = -\left[\frac{\partial G^{k,l}}{\partial u^{k,l-1}}\right]^T \Lambda^{k,l} \tag{85}$$

Our last constraint equation is applied for $k = 1, 2, ..., n-1$ and $l = 0$, and is complicated due to the presence of the 0 state in all 5 steps of the Runga-Kutta iteration.

$$\begin{aligned}\left[\frac{\partial G^{k,0}}{\partial u^{k,0}}\right]^T \Lambda^{k,0} = &- \left[\frac{\partial G^{k,1}}{\partial u^{k,0}}\right]^T \Lambda^{k,1} - \left[\frac{\partial G^{k,2}}{\partial u^{k,0}}\right]^T \Lambda^{k,2} - \left[\frac{\partial G^{k,3}}{\partial u^{k,0}}\right]^T \Lambda^{k,3} \\ &- \left[\frac{\partial G^{k,4}}{\partial u^{k,0}}\right]^T \Lambda^{k,4} - \left[\frac{\partial G^{k,5}}{\partial u^{k,0}}\right]^T \Lambda^{k,5}\end{aligned} \tag{86}$$

We can apply the specific definitions of the derivatives to the recurrence relations to get the following equations. The adjoint for the first step in the backwards-in-time integration is below.

$$[I]^T \Lambda^n = -\left[\frac{\partial L}{\partial u^n}\right]^T \tag{87}$$

The adjoint equation to switch from the 5 index to the next state is as follows.

$$\Lambda^{k-1,5} = \Lambda^{k,0} \tag{88}$$

We can see that the iterative equation for $k \neq 0$ has no dependence on the time step, because that depends solely on the mesh and the 0 sub-stage state.

$$[I]^T \Lambda^{k,l-1} = -\left[CFL\Delta t^k \alpha^{l-1} \frac{\partial R(u^{k,l-1})}{\partial u^{k,l-1}}\right]^T \Lambda^{k,l} \tag{89}$$

The constraint for the adjoint for the 0 sub-stage is:

$$\begin{aligned}[I]^T \Lambda^{k,0} = &- \left[-I - CFL\alpha^0 \frac{\partial \Delta t}{\partial u^{k,0}} R(u^{k,0}) - CFL\alpha^0 \Delta t \frac{\partial R(u^{k,0})}{\partial u^{k,0}}\right]^T \Lambda^{k,1} \\ &- \left[-I - CFL\alpha^1 \frac{\partial \Delta t}{\partial u^{k,0}} R(u^{k,1})\right]^T \Lambda^{k,2} \\ &- \left[-I - CFL\alpha^2 \frac{\partial \Delta t}{\partial u^{k,0}} R(u^{k,2})\right]^T \Lambda^{k,3} \\ &- \left[-I - CFL\alpha^3 \frac{\partial \Delta t}{\partial u^{k,0}} R(u^{k,3})\right]^T \Lambda^{k,4} \\ &- \left[-I - CFL\alpha^4 \frac{\partial \Delta t}{\partial u^{k,0}} R(u^{k,4})\right]^T \Lambda^{k,5}\end{aligned} \tag{90}$$

Lastly, our sensitivity equation comes from substituting our constraint derivatives into equation (52):

$$\begin{aligned}\frac{dJ}{dD} = L(u^n, D) &- \Lambda^{n-1,5T} CFL\alpha^k \left[\frac{\partial \Delta t^{n-1}}{\partial D} R(u^{n-1,4}) + \Delta t^{n-1} \frac{\partial R(u^{n-1,4})}{\partial D}\right] \\ &- ... \\ &- \Lambda^{n-1,1T} CFL\alpha^k \left[\frac{\partial \Delta t^{n-1}}{\partial D} R(u^{n-1,0}) + \Delta t^{n-1} \frac{\partial R(u^{n-1,0})}{\partial D}\right] \\ &- ... \\ &- \Lambda^{0,5T} CFL\alpha^k \left[\frac{\partial \Delta t^0}{\partial D} R(u^{0,4}) + \Delta t \frac{\partial R(u^{0,4})}{\partial D}\right] \\ &- ... \\ &- \Lambda^{0,1T} CFL\alpha^k \left[\frac{\partial \Delta t^0}{\partial D} R(u^{0,0}) + \Delta t \frac{\partial R(u^{0,0})}{\partial D}\right]\end{aligned} \tag{91}$$