Numerical Algorithms manuscript No.
(will be inserted by the editor)

Approximate Linearization of Fixed Point Iterations

Error Analysis of Tangent and Adjoint Problems Linearized about Non-Stationary Points

Emmett Padway · Dimitri Mavriplis

Received: date / Accepted: date

Abstract Previous papers have shown the impact of partial convergence of discretized PDE on the accuracy of tangent and adjoint linearizations. A series of papers suggested linearization of the fixed point iteration used in the solution as a mean of computing the sensitivities rather than linearizing the discretized PDE as the lack of convergence of the nonlinear problem indicates that the discretized form of the governing equations has not been satisfied. These works showed that the accuracy of an approximate linearization depends in part on the convergence of the nonlinear system. This work shows an error analysis of the impact of the approximate linearization for both the tangent and adjoint modes and shows a series of results for an exact Newton solver, an inexact Newton solver, and a low storage explicit Runge-Kutta scheme to confirm the error analyses.

1 Introduction

Tangent and Adjoint methods have become a larger part of the field of Computational Fluid Dynamics (CFD) as computers and algorithms have developed and a concurrent push for automatic design has strengthened. Both of these methods are derived off the requirement that the discretized form of the governing Partial Differential equation (PDE) is satisfied; which is denoted by the residual being equal to 0. The adjoint linearization (either through the discrete or continuous methods) is more commonly used as it scales independently of the number of design variables and its use for error estimation [12,18]. These methods are highly accurate and are often verified through use of the complex-step finite differentiation as this does not suffer from round off error [11]. However, for cases where the discretized governing equation is not satisfied, the finite-differentiation gives the

E. Padway
University of Wyoming, Department of Mechanical Engineering, 1000 E. University Ave, Laramie, WY, 82071
E-mail: epadway@uwyo.edu

D. Mavriplis
University of Wyoming, Department of Mechanical Engineering, 1000 E. University Ave, Laramie, WY, 82071

sensitivity of the solution process and the adjoint and tangent are no longer exact linearizations and the adjoint loses duality to the primal problem. Such cases have been shown to be very common in CFD simulations [4,5,15] and in such cases the sensitivities suffer greatly in terms of accuracy and exhibit a high dependence on the state of the simulation at termination.
To address these cases Padway and Mavriplis [15,13] look into linearizing the fixed point iteration used to solve the nonlinear problem and obtain the sensitivities of the solution process and they show better sensitivity behavior and successful optimizations with this method; they also applied this method to error estimation and adaptive mesh refinement [14]. This method inherits some of the characteristics shown in the work on one shot optimzation methods [3] including the guaranteed convergence of the tangent and adjoint problems. Padway and Mavriplis[13] showed by numerical experiment that for an approximately linearized fixed point iteration, convergence of the non-linear problem led to a decrease in error from the approximate linearization. This paper shows that providing the residual operator is properly linearized, the error from inexact linearization of the fixed point iteration vanishes as the nonlinear problem is converged through a series of proofs and numerical experiments. Additionally it shows that for nonconverged problems, the error in the sensitivity has two terms, one of which scales with the residual, and the other of which scales with the error in the approximate linearization; the second of which is multiplied by the derivative of the contractive fixed point iteration. As problems with nonconvergent fixed points have become more prevalent as the field has attempted more difficult problems, methods like those mentioned above show more promise. In the realm of design optimization, which is the focus of this paper, 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 [2]. As such having a good estimate on the error of the sensitivities for such a method (when compared to the complex-step finite-differentiation) that can provide better designs than those provided by the classical steady state adjoint is highly desirable. In a similar vein, Brown and Nadarajah[1] investigate and bound for the error in the adjoint computed sensitivities arising from partial convergence of the primal problem for problem that show smooth convergence behavior.

2 Background and In-House Solver

2.1 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 analysis problem) can be written as follows.

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

Which can also be written as:

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

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

2.2 Spatial Discretization

The residual about the closed control volume is given as:

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

This equation gives the operator in the aforementioned requirement for the adjoint and tangent systems. In the discretized form of the residual operator, x is the mesh points, F is the numerical flux across the element boundary, B is the element boundary, and n is the edge normal on the element boundary. 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 [8]. The solver has three different flux calculations implemented and linearized, these are the Lax-Friedrichs[7], Roe[16], and van Leer[6] schemes. In this work, only the Lax-Friedrichs and Van-Leer schemes are used.

2.3 Steady-State Solver

To solve the discretized PDE shown above we define a fixed point iteration, where u is the conservative variable vector and D is fixed for each design iteration:

$$u^{k+1} = N(u^k, D) = u^k + H(u^k, D) = u^k + A(u^k, D)R(u^k, D) \quad (4)$$

The solver technology for this code uses either explicit pseudo-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 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:

$$u^k = u^{k-1} + \Delta u \quad (5)$$

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

$$[P]\,\Delta u = -R(u) \quad (6)$$

We can substitute our expression for Δu into the time-stepping equation (5) to obtain our final form of this equation.

$$u^k = u^{k-1} - [P_{k-1}]^{-1} R \quad (7)$$

Here $[P_{k-1}]$ is a first-order spatially accurate Jacobian augmented with a diagonal term to ensure that it is diagonally dominant, shown in equation (8).

$$[P_{k-1}] = \left[\frac{\partial R}{\partial u^{k-1}}\right]_1 + \frac{vol}{\Delta t CFL} \quad (8)$$

note that for the above definition this would imply:

$$A(u^{k-1}, D) = -\left[P_{k-1}\right]^{-1} \tag{9}$$

Please note the subscript on the jacobian above denotes that it is a 1st order jacobian; in this work the subscripts of 1 and 2 will denote first and second order spatially accurate jacobians respectively. The equation for the local explicit time step limit Δt is given as:

$$\Delta t_i = \frac{r_i}{\sqrt{(u^2 + v^2)} + c} \tag{10}$$

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.
Furthermore, the CFL is scaled either with a simple ramping coefficient (β) and cap criterion:

$$CFL^{k+1} = min(\beta \cdot CFL, CFL_{max}) \tag{11}$$

or with a linesearch and CFL controller [9], which seeks to minimize the L2 norm of the pseudo-temporal residual, defined as:

$$R_t(u + \alpha\Delta u) = \frac{vol}{\Delta t}\alpha\Delta u^n + R(u + \alpha\Delta u^n) \tag{12}$$

In order to solve the linear system we use either of two smoothers: a point-implicit Jacobi or point-implicit Gauss-Seidel solver, or a flexible GMRES solver [17] preconditioned by the gauss-seidel smoother. These solvers are also implemented to solve the stiff steady state tangent and adjoint systems as well.

2.4 A Review of Tangent and Adjoint Systems

2.4.1 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 vector of point coordinates. In order to obtain an expression for the sensitivities we take the derivative of the objective functional [10]:

$$\frac{dL}{dD} = \frac{\partial L}{\partial x}\frac{\partial x}{\partial D} + \frac{\partial L}{\partial u}\frac{\partial u}{\partial D} \tag{13}$$

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 dependent on the choice of mesh motion scheme and design variables. It is not possible to obtain $\frac{\partial u}{\partial D}$ through linearization of the subroutines in the code without linearizing the entire analysis 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.

$$\left[\frac{\partial R}{\partial x}\right]\frac{\partial x}{\partial D} + \left[\frac{\partial R}{\partial u}\right]_2\frac{\partial u}{\partial D} = 0 \tag{14}$$

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

$$\left[\frac{\partial R}{\partial u}\right]_2 \frac{\partial u}{\partial D} = -\left[\frac{\partial R}{\partial x}\right] \frac{\partial x}{\partial D} \tag{15}$$

We solve this linear system, using hand differentiated subroutines to provide the left hand matrix $\left[\frac{\partial R}{\partial u}\right]_2$, 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 (13) to obtain the final sensitivities.

2.4.2 Discrete Adjoint Formulation

The adjoint formulation begins with the same sensitivity equation:

$$\frac{dL}{dD} = \frac{\partial L}{\partial x}\frac{\partial x}{\partial D} + \frac{\partial L}{\partial u}\frac{\partial u}{\partial D} \tag{16}$$

Using the condition $R(u(D), D) = 0$, we return to equation (15) and pre-multiply both sides of the equation by the inverse Jacobian matrix to obtain:

$$\frac{\partial u}{\partial D} = -\left[\frac{\partial R}{\partial u}\right]_2^{-1} \left[\frac{\partial R}{\partial x}\right] \frac{\partial x}{\partial D} \tag{17}$$

Substituting the above expression into the sensitivity equation yields:

$$\frac{dL}{dD} = \frac{\partial L}{\partial x}\frac{\partial x}{\partial D} - \frac{\partial L}{\partial u}\left[\frac{\partial R}{\partial u}\right]_2^{-1} \left[\frac{\partial R}{\partial x}\right] \frac{\partial x}{\partial D} \tag{18}$$

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

$$\mathbf{\Lambda}^T = -\frac{\partial L}{\partial u}\left[\frac{\partial R}{\partial u}\right]_2^{-1} \tag{19}$$

which gives an equation for the adjoint variable:

$$\left[\frac{\partial R}{\partial u}\right]_2^T \mathbf{\Lambda} = -\left[\frac{\partial L}{\partial u}\right]^T \tag{20}$$

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

$$\frac{dL}{dD} = \left[\frac{\partial L}{\partial x} + \mathbf{\Lambda}^T \frac{\partial R}{\partial x}\right] \frac{\partial x}{\partial D} \tag{21}$$

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

$$[K]^T \mathbf{\Lambda_x} = \left[\frac{\partial L}{\partial x}\right]^T + \left[\frac{\partial R}{\partial x}\right]^T \mathbf{\Lambda} \tag{22}$$

and the final expression for sensitivity is given as:

$$\frac{dL}{dD} = \mathbf{\Lambda_x}^T \frac{\partial x_s}{\partial D} \tag{23}$$

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.

3 Development of the Pseudo-Time Accurate Tangent

In this section we show the previous pseudo-time accurate tangent formulations [15, **?**]. We also discuss the impacts on implementation and accuracy of some of the assumptions made in the previous formulation and the one presented here. As stated above, the assumptions made are exact when the linear system at each iteration is solved to machine precision. In such cases the tangent sensitivities would exactly correspond to the sensitivities of solution process itself, and will correspond exactly to the sensitivities obtained by the complex-step finite difference method at each step of the solution process. The idea is that a sensitivity vector obtained from the 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 analyses are very sensitive to the specific state about which they are linearized and we cannot say with certainty what they correspond to mathematically. This formulation is the equivalent of using the derivative of the solution process to drive the optimization. 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 $\frac{\partial}{\partial x}\frac{\partial x}{\partial D}$.

3.1 Tangent System for quasi-Newton Method

For this section, we begin from equation (7)

$$u^k = u^{k-1} - [P_{k-1}]^{-1} R \tag{24}$$

where $[P_{k-1}]$ is again a first-order accurate Jacobian augmented with a diagonal term to ensure that it is diagonally dominant, as described in equation (8).

$$[P_{k-1}] = \left[\frac{\partial R}{\partial u^{k-1}}\right]_1 + \frac{vol}{\Delta t CFL} \tag{25}$$

If we take the derivative of each side of equation (25) we obtain:

$$\frac{du}{dD}^k = \frac{du^{k-1}}{dD} - [P_{k-1}]^{-1}\left[\frac{\partial R}{\partial D} + \left[\frac{\partial R}{\partial u^{k-1}}\right]_2 \frac{du}{dD}^k\right] - \left[\frac{d\,[P_{k-1}]^{-1}}{dD}\right] R(u^{k-1}) \tag{26}$$

Here, rather than neglecting the change in the preconditioner we use a definition of the derivative of the matrix inverse defined by:

$$\frac{d\,[K]^{-1}}{dx} = -\,[K]^{-1}\left[\frac{dK}{dx}\right][K]^{-1} \tag{27}$$

This assumption will not be exact for any case in which the linear system solve is not exact to machine precision. However, it will allow us to have a clearly defined source of error in our computation and evaluate how much the linear tolerance of the system affects the sensitivity computation. Furthermore, we can argue that the error from this sensitivity will go to 0 as we approach full convergence, and we can investigate the behavior in near-ergodic limit cycle oscillations. By computing

the total derivative of the nonlinear solver– as in equation (26)– and substituting in the expression from (27) into equation (26), the below expression is generated.

$$\begin{aligned}\frac{du^k}{dD} = \frac{du^{k-1}}{dD} - [P_{k-1}]^{-1}\left[\left[\frac{\partial R}{\partial u^{k-1}}\right]_2 \frac{du^{k-1}}{dD} + \left[\frac{\partial R}{\partial x}\right]\frac{dx}{dD}\right] \\ + [P_{k-1}]^{-1}\frac{d\,[P_{k-1}]}{dD}\,[P_{k-1}]^{-1}\,R(u^{k-1}, D)\end{aligned} \tag{28}$$

Expanding the total derivative shows that this is in fact the sum of two matrix vector products:

$$\begin{aligned}\frac{du^k}{dD} = \frac{du^{k-1}}{dD} - [P_{k-1}]^{-1}\left[\left[\frac{\partial R}{\partial u^{k-1}}\right]_2 \frac{du^{k-1}}{dD} + \left[\frac{\partial R}{\partial x}\right]\frac{dx}{dD}\right] \\ + [P_{k-1}]^{-1}\left[\frac{\partial P_{k-1}}{\partial u^{k-1}}\frac{du^{k-1}}{dD} + \frac{\partial P_{k-1}}{\partial x}\frac{dx}{dD}\right][P_{k-1}]^{-1}\,R(u^{k-1}, D)\end{aligned} \tag{29}$$

While the full hessian is an undesireable item to compute, this formulation requires two hessian vector products that can be obtained through complex perturbations to the conservative variables and nodal coordinate vectors, and subsequent evaluation of the jacobian. This avoids the need for the full hessian computation. Furthermore, one of the matrix inverses can be removed by reusing the computation of Δu and rewriting the equation as:

$$\begin{aligned}\frac{du^k}{dD} = \frac{du^{k-1}}{dD} - [P_{k-1}]^{-1}\left[\left[\frac{\partial R}{\partial u^{k-1}}\right]_2 \frac{du^{k-1}}{dD} + \left[\frac{\partial R}{\partial x}\right]\frac{dx}{dD}\right] \\ + [P_{k-1}]^{-1}\left[\frac{\partial P_{k-1}}{\partial u^{k-1}}\frac{du^{k-1}}{dD} + \frac{\partial P_{k-1}}{\partial x}\frac{dx}{dD}\right]\Delta u\end{aligned} \tag{30}$$

This can then be rewritten as:

$$\frac{du}{dD}^{k} = \frac{du}{dD}^{k-1} + \Delta\left(\frac{du}{dD}\right) \tag{31}$$

and we solve the following linear system:

$$\begin{aligned}[P_{k-1}]\,\Delta\left(\frac{du}{dD}\right) = -\left[\left[\frac{\partial R}{\partial u^{k-1}}\right]_2 \frac{du^{k-1}}{dD} + \left[\frac{\partial R}{\partial x}\right]\frac{dx}{dD}\right] \\ + \left[\frac{\partial P_{k-1}}{\partial u^{k-1}}\frac{du^{k-1}}{dD} + \frac{\partial P_{k-1}}{\partial x}\frac{dx}{dD}\right]\Delta u\end{aligned} \tag{32}$$

It is important to note that this expression is only exact for machine-zero solution of the linear system at every iteration, but for those cases we will have exact correspondence between this and the complex-step computed sensitivities. This saves us from needing to differentiate the entire linear solution process, which is intractable for many cases for the forward mode, and even more difficult for the reverse mode. The reverse differentiation of a Krylov solver would be an onerous task that would yield little gain. One additional point is that with this method there are no conditions on exact duals of the linear solver, and one can use different solvers for the forward and the reverse linearizations. We can also note that, where

in the initial formulation– in equation (29)– we see 3 approximate linear solves; when we group terms and use the already stored information from the analysis solve we are left with only one approximate linear solve, the same as in the analysis problem's nonlinear solver. Although we are not limited to using the same linear solver for the analysis and tangent problems and its dual for the adjoint problem, by doing so we gain in that we are guaranteed to have a converging tangent and adjoint solver. We would then be solving using the same algorithm that ran without divergence through the analysis portion, when we apply this algorithm to the tangent and adjoint problem we have the same eigenvalues, same condition numbers, and same convergence properties. Therefore, if our analysis problem did not diverge, neither will the tangent or the adjoint.

4 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 a program that runs through n pseudo-time steps. From this we define our objective function L as:

$$L = L(u^n, u^{n-1}, ..., u^{n-m}, D) \tag{33}$$

where u^n is the conservative variable vector at the final time step n 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.

$$G^k = G^k(u^k, u^{k-1}, D) = 0 \tag{34}$$

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

$$\begin{aligned} J(D, u^n, u^{n-1}, u^{n-2}, ..., \Lambda^n, \Lambda^{n-1}, ...) = {} & L(u^n, u^{n-1}, ..., u^{n-m}, D) \\ & + \Lambda^{nT} G^n(u^n(D), u^{n-1}(D), D) \\ & + ... \\ & + \Lambda^{1T} G^1(u^1(D), u^0(D), D) \end{aligned} \tag{35}$$

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.

$$\begin{aligned}
\frac{\partial J}{\partial u^n} &= \frac{\partial L}{\partial u^n} + \Lambda^{nT}\frac{\partial G^n}{\partial u^n} = 0 \\
\frac{\partial J}{\partial u^{n-1}} &= \frac{\partial L}{\partial u^{n-1}} + \Lambda^{nT}\frac{\partial G^n}{\partial u^{n-1}} + \Lambda^{n-1T}\frac{\partial G^{n-1}}{\partial u^{n-1}} = 0 \\
\frac{\partial J}{\partial u^{n-2}} &= \frac{\partial L}{\partial u^{n-2}} + \Lambda^{n-1T}\frac{\partial G^{n-1}}{\partial u^{n-2}} + \Lambda^{n-2T}\frac{\partial G^{n-2}}{\partial u^{n-2}} = 0 \\
&\ldots \\
\frac{\partial J}{\partial u^1} &= \frac{\partial L}{\partial u^1} + \Lambda^{2T}\frac{\partial G^2}{\partial u^1} + \Lambda^{1T}\frac{\partial G^1}{\partial u^1} = 0
\end{aligned} \tag{36}$$

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

$$\begin{aligned}
\frac{\partial J}{\partial u^n} &= \frac{\partial L}{\partial u^n} + \Lambda^{nT}\frac{\partial G^n}{\partial u^n} = 0 \\
\frac{\partial J}{\partial u^{n-1}} &= \frac{\partial L}{\partial u^{n-1}} + \Lambda^{nT}\frac{\partial G^n}{\partial u^{n-1}} + \Lambda^{n-1T}\frac{\partial G^{n-1}}{\partial u^{n-1}} = 0 \\
&\ldots \\
\frac{\partial J}{\partial u^{n-m}} &= \frac{\partial L}{\partial u^{n-m}} + \Lambda^{n-(m-1)T}\frac{\partial G^{n-(m-1)}}{\partial u^{n-m}} + \Lambda^{n-mT}\frac{\partial G^{n-m}}{\partial u^{n-m}} = 0 \\
\frac{\partial J}{\partial u^{n-(m+1)}} &= \Lambda^{n-mT}\frac{\partial G^{n-m}}{\partial u^{n-(m+1)}} + \Lambda^{n-(m+1)T}\frac{\partial G^{n-(m+1)}}{\partial u^{n-(m+1)}} = 0 \\
&\ldots \\
\frac{\partial J}{\partial u^1} &= \Lambda^{2T}\frac{\partial G^2}{\partial u^1} + \Lambda^{1T}\frac{\partial G^1}{\partial u^1} = 0
\end{aligned} \tag{37}$$

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

$$\left[\frac{\partial G^n}{\partial u^n}\right]^T \Lambda^n = -\left[\frac{\partial L}{\partial u^n}\right]^T \tag{38}$$

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

$$\frac{\partial G^{k-1}}{\partial u^{k-1}}^T \Lambda^{k-1} = -\frac{\partial G^k}{\partial u^{k-1}}^T \Lambda^k \tag{39}$$

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

$$\frac{\partial G^{k-1}}{\partial u^{k-1}}^T \Lambda^{k-1} = -\frac{\partial G^k}{\partial u^{k-1}}^T \Lambda^k - \left[\frac{\partial L}{\partial u^{k-1}}\right]^T \tag{40}$$

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:

$$\frac{\partial G^{k-1}}{\partial u^{k-1}}^T \Lambda^{k-1} = -\frac{\partial G^k}{\partial u^{k-1}}^T \Lambda^k \tag{41}$$

Lastly, we can take the derivative of equation (35) with respect to the design variables to get the sensitivity equation.

$$\frac{dJ}{dD} = \frac{\partial L}{\partial D} + \Lambda^{nT}\frac{\partial G^n}{\partial D} + \Lambda^{n-1T}\frac{\partial G^{n-1}}{\partial D} + \Lambda^{n-2T}\frac{\partial G^{n-2}}{\partial D} + ... \tag{42}$$

4.1 Adjoint Computed Sensitivites for quasi-Newton Method

For this section we begin with the derivatives of our constraint equations below:

$$\begin{aligned}
\frac{\partial G^k}{\partial u^k} &= I \\
\frac{\partial G^k}{\partial u^{k-1}} &= -I + \left[P_{k-1}\right]^{-1} \left[\frac{\partial R(u^{k-1})}{\partial u^{k-1}}\right]_2 + \frac{\partial \left[P_{k-1}\right]^{-1}}{\partial u^{k-1}} R(u^{k-1}) \\
\frac{\partial G^k}{\partial D} &= \left[P_{k-1}\right]^{-1} \frac{\partial R(u^{k-1})}{\partial D} + \frac{\partial \left[P_{k-1}\right]^{-1}}{\partial D} R(u^{k-1})
\end{aligned} \tag{43}$$

By using the differentiation of a matrix inverse shown in equation (27) we can obtain the derivative of the constraint term shown below:

$$\begin{aligned}
\frac{\partial G^k}{\partial u^k} &= I \\
\frac{\partial G^k}{\partial u^{k-1}} &= -I + \left[P_{k-1}\right]^{-1} \left[\frac{\partial R(u^{k-1})}{\partial u^{k-1}}\right]_2 \\
&\quad - \left[P_{k-1}\right]^{-1} \frac{\partial \left[P_{k-1}\right]}{\partial u^{k-1}} \left[P_{k-1}\right]^{-1} R(u^{k-1}) \\
\frac{\partial G^k}{\partial D} &= \left[P_{k-1}\right]^{-1} \frac{\partial R(u^{k-1})}{\partial D} - \left[P_{k-1}\right]^{-1} \frac{\partial \left[P_{k-1}\right]}{\partial D} \left[P_{k-1}\right]^{-1} R(u^{k-1})
\end{aligned} \tag{44}$$

Using the definition of the nonlinear solver increment we can simplify the above equation with:

$$\begin{aligned}
\frac{\partial G^k}{\partial u^k} &= I \\
\frac{\partial G^k}{\partial u^{k-1}} &= -I + \left[P_{k-1}\right]^{-1} \left[\left[\frac{\partial R(u^{k-1})}{\partial u^{k-1}}\right]_2 - \frac{\partial \left[P_{k-1}\right]}{\partial u^{k-1}} \Delta u\right] \\
\frac{\partial G^k}{\partial D} &= \left[P_{k-1}\right]^{-1} \left[\frac{\partial R(u^{k-1})}{\partial D} - \frac{\partial \left[P_{k-1}\right]}{\partial x} \frac{dx}{dD} \Delta u\right]
\end{aligned} \tag{45}$$

Please note that we can compute these hessian vector products using complex frechet derivatives, rather than hand differentiating the residual operator twice to obtain the Hessian operator; even though we are in the adjoint mode, the hessian vector products are not transpose matrix vector products and can therefore be computed using frechet derivatives. Using the equation for the adjoint at the final pseudo-time step with our constraint derivatives we get the following initial source term.

$$\left[I\right] \Lambda^n = -\left[\frac{\partial L}{\partial u^n}\right]^T \tag{46}$$

Substituting in the constraint derivatives from equation (45) into equation (41) returns:

$$\left[I\right] \Lambda^{k-1} = -\left[-I + \left[P_{k-1}\right]^{-1} \left[\left[\frac{\partial R(u^{k-1})}{\partial u^{k-1}}\right]_2 - \frac{\partial \left[P_{k-1}\right]}{\partial u^{k-1}} \Delta u\right]\right]^T \Lambda^k \tag{47}$$

we can write this recurrence relation in delta form as in the analysis solver.

$$\Delta\Lambda = -\left[[P_{k-1}]^{-1}\left[\left[\frac{\partial R(u^{k-1})}{\partial u^{k-1}}\right]_2 - \frac{\partial\left[P_{k-1}\right]}{\partial u^{k-1}}\Delta u\right]\right]^T \Lambda^k \tag{48}$$

Distributing the transpose returns:

$$\Delta\Lambda = -\left[\left[\frac{\partial R(u^{k-1})}{\partial u^{k-1}}\right]_2 - \frac{\partial\left[P_{k-1}\right]}{\partial u^{k-1}}\Delta u\right]^T [P_{k-1}]^{-T}\,\Lambda^k \tag{49}$$

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

$$[P_{k-1}]^T\,\psi^k = \Lambda^k \tag{50}$$

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

$$\Delta\Lambda = -\left[\left[\frac{\partial R(u^{k-1})}{\partial u^{k-1}}\right]_2 - \frac{\partial\left[P_{k-1}\right]}{\partial u^{k-1}}\Delta u\right]^T \psi^k \tag{51}$$

It is important to note that, as noted before we do not need exact dual correspondence here between the adjoint solver and the tangent one. Regardless of duality this formulation will recover machine precision accuracy in the sensitivities in the limit of the linear system being solved to machine precision. We substitute the constraint derivatives from equation (45) into the sensitivity equation (42) and we obtain:

$$\begin{aligned}\frac{dJ}{dD} = \frac{\partial L}{\partial D} &+ \Lambda^{nT}\,[P_{n-1}]^{-1}\left[\frac{\partial R(u^{n-1})}{\partial D} - \frac{\partial\left[P_{n-1}\right]}{\partial x}\frac{dx}{dD}\Delta u\right] \\ &+ \ldots \\ &+ \Lambda^{1T}\,[P_0]^{-1}\left[\frac{\partial R(u^0)}{\partial D} - \frac{\partial\left[P_0\right]}{\partial x}\frac{dx}{dD}\Delta u\right]\end{aligned} \tag{52}$$

We can refer back to our definition of the secondary adjoint variable to simplify this equation and remove an additional linear solve and get the equation below:

$$\begin{aligned}\frac{dJ}{dD} = \frac{\partial L}{\partial D} &+ \psi^{nT}\left[\frac{\partial R(u^{n-1})}{\partial D} - \frac{\partial\left[P_{n-1}\right]}{\partial x}\frac{dx}{dD}\Delta u\right] \\ &+ \ldots \\ &+ \psi^{1T}\left[\frac{\partial R(u^0)}{\partial D} - \frac{\partial\left[P_0\right]}{\partial x}\frac{dx}{dD}\Delta u\right]\end{aligned} \tag{53}$$

We can note that the adjoint sensitivity formulation uses only one approximate linear solver per nonlinear step, like the forward and tangent solvers. Should we want to guarantee convergence of the adjoint problem, we can use the dual solver of the primal linear solver used at each nonlinear iteration. Please note that these linearizations are only exact for cases in which the linear system is sovled to machine precision at each non-linear iteration. This is not a feasible requirement for realistic CFD solvers, so we have to investigate the impact of partial convergence of the linear system on the error in sensitivities.

5 General Sensitivity Convergence Proof for Approximate Tangent Linearization of the Fixed Point Iteration

We begin with a fixed point iteration:

$$u^{k+1} = N(u^k, D) = u^k + H(u^k, D) = u^k + A(u^k, D)R(u^k, D) \tag{54}$$

where A is some operator dependent on the pseudo-temporal discretization and R is the appropriate residual operator for the spatial discretization of the governing equations. For an exact linearization of the solution process we have the expression below:

$$\frac{du^{k+1}}{dD} = \frac{dN(u^k)}{dD} = \frac{du^k}{dD} + \frac{dA}{dD}R + A\frac{dR}{dD} \tag{55}$$

For an inexact linearization we inexactly linearize A as $\widetilde{\frac{dA}{dD}}$ and obtain:

$$\frac{\widetilde{du^{k+1}}}{dD} = \frac{d\widetilde{N(u^k)}}{dD} = \frac{\widetilde{du^k}}{dD} + \frac{\widetilde{dA}}{dD}R + A\frac{\widetilde{dR}}{dD} \tag{56}$$

To get the error in the conservative variable sensitivities we subtract the two expressions from one another:

$$\frac{du^{k+1}}{dD} - \frac{\widetilde{du^{k+1}}}{dD} = \frac{du^k}{dD} - \frac{\widetilde{du^k}}{dD} + \frac{dA}{dD}R - \frac{\widetilde{dA}}{dD}R + A\frac{dR}{dD} - A\frac{\widetilde{dR}}{dD} \tag{57}$$

We define $\frac{d\epsilon}{dD}$ as the error in $\frac{du}{dD}$ and group like terms:

$$\frac{d\epsilon^{k+1}}{dD} = \frac{d\epsilon^k}{dD} + \left[\frac{dA}{dD} - \frac{\widetilde{dA}}{dD}\right]R + A\left[\frac{dR}{dD} - \frac{\widetilde{dR}}{dD}\right] \tag{58}$$

$$\begin{aligned}\frac{d\epsilon^{k+1}}{dD} = \frac{d\epsilon^k}{dD} &+ \left[\frac{\partial A}{\partial x}\frac{dx}{dD} + \frac{\partial A}{\partial u^k}\frac{du^k}{dD} - \frac{\partial \widetilde{A}}{\partial x}\frac{dx}{dD} - \frac{\widetilde{\partial A}}{\partial u^k}\frac{\widetilde{du^k}}{dD}\right]R \\ &+ A\left[\frac{\partial R}{\partial x}\frac{dx}{dD} + \frac{\partial R}{\partial u^k}\frac{du^k}{dD} - \frac{\partial R}{\partial x}\frac{dx}{dD} - \frac{\partial R}{\partial u^k}\frac{\widetilde{du^k}}{dD}\right]\end{aligned} \tag{59}$$

We can cancel out like terms to obtain:

$$\begin{aligned}\frac{d\epsilon^{k+1}}{dD} = \frac{d\epsilon^k}{dD} &+ \left[\frac{\partial A}{\partial x}\frac{dx}{dD} + \frac{\partial A}{\partial u^k}\frac{du^k}{dD} - \frac{\partial \widetilde{A}}{\partial x}\frac{dx}{dD} - \frac{\widetilde{\partial A}}{\partial u^k}\frac{\widetilde{du^k}}{dD}\right]R \\ &+ A\left[\frac{\partial R}{\partial u^k}\frac{du^k}{dD} - \frac{\partial R}{\partial u^k}\frac{\widetilde{du^k}}{dD}\right]\end{aligned} \tag{60}$$

We can then group like terms and use the definition of $\frac{d\epsilon}{dD}$:

$$\begin{aligned}\frac{d\epsilon^{k+1}}{dD} = \frac{d\epsilon^k}{dD} &+ \left[\left(\frac{\partial A}{\partial x} - \frac{\widetilde{\partial A}}{\partial x}\right)\frac{dx}{dD} + \left(\frac{\partial A}{\partial u} - \frac{\widetilde{\partial A}}{\partial u}\right)\frac{du^k}{dD} + \frac{\partial A}{\partial u^k}\frac{d\epsilon^k}{dD}\right]R \\ &+ A\left[\frac{\partial R}{\partial u^k}\frac{d\epsilon^k}{dD}\right]\end{aligned} \tag{61}$$

We can rearrange terms:

$$\begin{aligned}\frac{d\epsilon^{k+1}}{dD} = \frac{d\epsilon^k}{dD} + A\left[\frac{\partial R}{\partial u^k}\frac{d\epsilon^k}{dD}\right] + \frac{\partial A}{\partial u^k}\frac{d\epsilon^k}{dD}R \\ + \left[\left(\frac{\partial A}{\partial x} - \widetilde{\frac{\partial A}{\partial x}}\right)\frac{dx}{dD} + \left(\frac{\partial A}{\partial u} - \widetilde{\frac{\partial A}{\partial u}}\right)\frac{du^k}{dD}\right]R\end{aligned} \tag{62}$$

We can group the first three terms on the right hand side into B where $B = \frac{\partial N}{\partial u}$, by cauchy-schwarz inequality we know that:

$$\left\|\frac{d\epsilon^k}{dD} + A\left[\frac{\partial R}{\partial u^k}\frac{d\epsilon^k}{dD}\right] + \frac{\partial A}{\partial u^k}\frac{d\epsilon^k}{dD}R\right\| < \|B\|\left\|\frac{d\epsilon^k}{dD}\right\| \tag{63}$$

and since B is the derivative of the contractive fixed point interation $\|B\| < 1$ and the error in the sensitivities expressed by $\frac{d\epsilon}{dD}$ decreases as the residual decreases and the contractivity of the fixed point iteration progresses through the primal solution process. One can also note, that in cases where A approaches $\frac{\partial R}{\partial u}^{-1}$ we have no explicit dependence on contractivity of the fixed point but the error will be multiplied by the residual in all terms.

6 General Sensitivity Convergence Proof for Approximate Adjoint Linearization of the Fixed Point Iteration

As in the tangent section, we wish to quantify the error introduced to the sensitivity calculation by the approximate linearization of the fixed point iteration. We know that for a properly implemented linearization and tranposition that the adjoint and tangent return the same sensitivity values (even for approximate linearizations and therefore that the error in the sensitivities goes to zero as the nonlinear problem is converged.Beginning from equation (42) first shown in the initial pseudo-time accurate adjoint derivations and reproduced below, where G is the shifted fixed point iteration, $G = u - N(u)$.

$$\frac{dL}{dD} = \frac{\partial L}{\partial D} + \Lambda^{nT}\frac{\partial G^n}{\partial D} + \Lambda^{n-1T}\frac{\partial G^{n-1}}{\partial D} + \Lambda^{n-2T}\frac{\partial G^{n-2}}{\partial D} + ... + \Lambda^{1T}\frac{\partial G^1}{\partial D} \tag{64}$$

As in the tangent proof we have the following identity for the fixed point iteration. It is important to note that by definition of the fixed point, A is not orthogonal to R and it is bounded away from 0, lest the fixed point terminate at a state that does not satisfy the discretized form of the governing equations (signifed by $R = 0$).

$$u^{k+1} = N(u^k, D) = u^k + H(u^k, D) = u^k + A(u^k, D)R(u^k, D) \tag{65}$$

Our fixed point iteration derivatives are expressed below, where the linearization of the residual is exact, but the linearization of the A matrix is approximate in the actual implementation:

$$\begin{aligned}\frac{\partial H^{k+1}}{\partial u^k} &= \frac{\partial A(u^k, D)}{\partial u^k}R(u^k, D) + A(u^k, D)\frac{\partial R(u^k, D)}{\partial u^k} \\ \frac{\partial u^{k+1}}{\partial D} &= \frac{\partial N(u^k, D)}{\partial D} = \frac{\partial A(u^k, D)}{\partial D}R(u^k, D) + A(u^k, D)\frac{\partial R(u^k, D)}{\partial D}\end{aligned} \tag{66}$$

Substituting in the approximate terms into 42 we get the expression below:

$$\widetilde{\frac{dL}{dD}} = \frac{\partial L}{\partial D} + \tilde{\Lambda}^{nT}\widetilde{\frac{\partial G^n}{\partial D}} + \tilde{\Lambda}^{n-1T}\widetilde{\frac{\partial G^{n-1}}{\partial D}} + \tilde{\Lambda}^{n-2T}\widetilde{\frac{\partial G^{n-2}}{\partial D}} + ... + \tilde{\Lambda}^{1T}\widetilde{\frac{\partial G^1}{\partial D}} \tag{67}$$

We can subtract the exact sensitivity equation from the approximate one:

$$\begin{aligned}
\frac{dL}{dD} - \widetilde{\frac{dL}{dD}} = &\left(\frac{\partial L}{\partial D} - \frac{\partial L}{\partial D}\right) \\
&+ \left(\Lambda^{nT}\frac{\partial G^n}{\partial D} - \tilde{\Lambda}^{nT}\widetilde{\frac{\partial G^n}{\partial D}}\right) \\
&+ \left(\Lambda^{n-1T}\frac{\partial G^{n-1}}{\partial D} - \tilde{\Lambda}^{n-1T}\widetilde{\frac{\partial G^{n-1}}{\partial D}}\right) \\
&+ \left(\Lambda^{n-2T}\frac{\partial G^{n-2}}{\partial D} - \tilde{\Lambda}^{n-2T}\widetilde{\frac{\partial G^{n-2}}{\partial D}}\right) \\
&+ ... \\
&+ \left(\Lambda^{1T}\frac{\partial G^1}{\partial D} - \tilde{\Lambda}^{1T}\widetilde{\frac{\partial G^1}{\partial D}}\right)
\end{aligned} \tag{68}$$

We define three error terms for the errors at each nonlinear iteration in the pseudo-time adjoint, the linearization of the nonlinear iteration with respect to the state variable and the linearization with respect to the design variables, denoted by $\epsilon^k_\Lambda, \epsilon^k_u, \epsilon^k_D$ respectively.

$$\begin{aligned}
\epsilon^k_\Lambda &= \Lambda^k - \tilde{\Lambda}^k \\
\epsilon^k_u &= \frac{\partial G^k}{\partial u^{k-1}} - \widetilde{\frac{\partial G^k}{\partial u^{k-1}}} \\
\epsilon^k_D &= \frac{\partial G^k}{\partial D} - \widetilde{\frac{\partial G^k}{\partial D}}
\end{aligned} \tag{69}$$

Using these epsilon definitions we can simplify the equations into the below equation with the epsilon terms as the unknowns:

$$\begin{aligned}
\frac{dL}{dD} - \widetilde{\frac{dL}{dD}} = &\epsilon^{nT}_\Lambda\frac{\partial G^n}{\partial D} + \Lambda^{nT}\epsilon^n_D - \epsilon^{nT}_\Lambda\epsilon^n_D \\
&+ \epsilon^{n-1T}_\Lambda\frac{\partial G^{n-1}}{\partial D} + \Lambda^{n-1T}\epsilon^{n-1}_D - \epsilon^{n-1T}_\Lambda\epsilon^{n-1}_D \\
&+ \epsilon^{n-2T}_\Lambda\frac{\partial G^{n-2}}{\partial D} + \Lambda^{n-2T}\epsilon^{n-2}_D - \epsilon^{n-2T}_\Lambda\epsilon^{n-2}_D \\
&+ ... \\
&+ \epsilon^{1T}_\Lambda\frac{\partial G^1}{\partial D} + \Lambda^{1T}\epsilon^1_D - \epsilon^{1T}_\Lambda\epsilon^1_D
\end{aligned} \tag{70}$$

It's clear that for the error to be 0 that ϵ^k_Λ and ϵ^k_D must go to 0 with convergence of the primal problem. As such, this proof cannot proceed any further without

expressions for the epsilon error terms, and we proceed using the fact that only the $\frac{\partial A}{\partial ()}$ term has an approximation to get the following two identities:

$$\begin{aligned}
\epsilon_u^k &= \frac{\partial G^k}{\partial u^{k-1}} - \widetilde{\frac{\partial G^k}{\partial u^{k-1}}} = \frac{\partial A}{\partial u}R + A\frac{\partial R}{\partial u} - \widetilde{\frac{\partial A}{\partial u}}R - A\frac{\partial R}{\partial u} = \left(\frac{\partial A}{\partial u} - \widetilde{\frac{\partial A}{\partial u}}\right)R \\
\epsilon_D^k &= \frac{\partial G^k}{\partial D} - \widetilde{\frac{\partial G^k}{\partial D}} = \frac{\partial A}{\partial D}R + A\frac{\partial R}{\partial D} - \widetilde{\frac{\partial A}{\partial D}}R - A\frac{\partial R}{\partial D} = \left(\frac{\partial A}{\partial D} - \widetilde{\frac{\partial A}{\partial D}}\right)R
\end{aligned} \tag{71}$$

The two error terms above have a scaling with the residual that is important to keep in mind. We use our definitions of the error terms to get an expression of the difference between the sensitivities with exact and approximate linearizations. Using equation 41 gives an error recurrence relationship:

$$\epsilon_\Lambda^{k-1} = -\frac{\partial G^k}{\partial u^{k-1}}^T \Lambda^k + \widetilde{\frac{\partial G^k}{\partial u^{k-1}}}^T \tilde{\Lambda}^k \tag{72}$$

We can substitute in the appropriate identities into the equation above:

$$\begin{aligned}
\epsilon_\Lambda^{k-1} &= \left(\frac{\partial G^k}{\partial u^{k-1}}^T - \epsilon_u^k\right)\left(\Lambda^k - \epsilon_\Lambda^k\right) - \frac{\partial G^k}{\partial u^{k-1}}^T \Lambda^k \\
&= -\epsilon_u^{kT}\Lambda^k + \epsilon_u^{kT}\epsilon_\Lambda^k - \frac{\partial G^k}{\partial u^{k-1}}\epsilon_\Lambda^k
\end{aligned} \tag{73}$$

Applying this equation to the second to last iteration:

$$\epsilon_\Lambda^{n-1} = -\epsilon_u^{nT}\Lambda^n + \epsilon_u^{nT}\epsilon_\Lambda^n - \frac{\partial G^n}{\partial u^{n-1}}\epsilon_\Lambda^n \tag{74}$$

Since $\Lambda^n = \frac{\partial L}{\partial u^n}$ which has no approximation error, $\epsilon_\Lambda^n = 0$. and substituting in the expression for ϵ_u^{kT} we get:

$$\epsilon_\Lambda^{n-1} = -\left(\frac{\partial A}{\partial u} - \widetilde{\frac{\partial A}{\partial u}}\right)R^n\Lambda^n \tag{75}$$

Should we proceed through the same steps for ϵ_Λ^{n-2}, we obtain:

$$\epsilon_\Lambda^{n-2} = -\left(\frac{\partial A}{\partial u} - \widetilde{\frac{\partial A}{\partial u}}\right)R^{n-1}\Lambda^{n-1} \tag{76}$$

It is therefore clear that in order to show that the error goes to zero it must be shown that $\left\|R^k\Lambda^k\right\| \approx 0$ for all iterations. It is important here to refer back to the definition of the fixed point iteration; A is bounded away from 0 and A is not orthogonal to R (lest the fixed point terminate at a state that does not satisfy the discretized form of the PDE). Therefore $\|A\| \neq 0$ and it is possible to move to an analysis of the $\Lambda^k R^k$ term. Since the pseudo-temporal adjoint converges at the reverse of the analysis process (through transpose and linearizing the fixed point iteration), $\left\|H^k\Lambda^k\right\| = \|H^n\Lambda^n\|$, which for a converged simulation is on the order of machine zero. Since A is bounded away from zero and not orthogonal to R, this means that $\left\|R^k\Lambda^k\right\| \approx 0$, and all the errors in the linearization of the A

operator (denoted by ϵ_u) do not contribute to the sensitivity error. The reverse convergence of the adjoint as compared to the analysis mode is confirmed by the results in [13], where the adjoint is shown to converge to its final value in the reverse of the analysis problem.

$$\epsilon_\Lambda^{k-1} = -\left(\frac{\partial A}{\partial u} - \widetilde{\frac{\partial A}{\partial u}}\right) R^k \Lambda^k \approx 0 \tag{77}$$

Otherwise, the error at every iteration is scaled by a magnitude of the final state residual for a stalled or truncated simulation. Using that $\epsilon_\Lambda^k = 0$ for a converged simulation we get the following expression when neglecting the terms in 70 that are multiplied by $\Lambda^k R^k \approx 0$:

$$\frac{dL}{dD} - \widetilde{\frac{dL}{dD}} = \Lambda^{nT}\epsilon_D^n + \Lambda^{n-1T}\epsilon_D^{n-1} + \Lambda^{n-2T}\epsilon_D^{n-2} + ... + \Lambda^{1T}\epsilon_D^1 \tag{78}$$

Substituting in the expression for the error in the linearization of the nonlinear iteration with respect to the design variable returns the following:

$$\begin{aligned}
\frac{dL}{dD} - \widetilde{\frac{dL}{dD}} = &-\Lambda^{nT}\left(\frac{\partial A}{\partial u}^n - \widetilde{\frac{\partial A}{\partial u}}^n\right) R^n \\
&- \Lambda^{n-1T}\left(\frac{\partial A}{\partial u}^{n-1} - \widetilde{\frac{\partial A}{\partial u}}^{n-1}\right) R^{n-1} \\
&- \Lambda^{n-2T}\left(\frac{\partial A}{\partial u}^{n-2} - \widetilde{\frac{\partial A}{\partial u}}^{n-2}\right) R^{n-2} \\
&- ... \\
&- \Lambda^{1T}\left(\frac{\partial A}{\partial u}^{1} - \widetilde{\frac{\partial A}{\partial u}}^{1}\right) R^{1}
\end{aligned} \tag{79}$$

We can rearrange the above expression:

$$\begin{aligned}
\frac{dL}{dD} - \widetilde{\frac{dL}{dD}} = &-\left(\frac{\partial A}{\partial u}^n - \widetilde{\frac{\partial A}{\partial u}}^n\right)^T \Lambda^n R^n \\
&- \left(\frac{\partial A}{\partial u}^{n-1} - \widetilde{\frac{\partial A}{\partial u}}^{n-1}\right)^T \Lambda^{n-1} R^{n-1} \\
&- \left(\frac{\partial A}{\partial u}^{n-2} - \widetilde{\frac{\partial A}{\partial u}}^{n-2}\right)^T \Lambda^{n-2} R^{n-2} \\
&- ... \\
&- \left(\frac{\partial A}{\partial u}^{1} - \widetilde{\frac{\partial A}{\partial u}}^{1}\right)^T \Lambda^{1} R^{1}
\end{aligned} \tag{80}$$

We can use the same argument as above that $\left\| R^k \Lambda^k \right\| \approx \| \Lambda^n R^n \| \approx 0$, and get that for a converged flow the error in the sensitivities goes to 0. Again we see that the convergence of the nonlinear problem leads to less error in the sensitivities

even in unconverged flows. From the expressions above we can see that the error in the sensitivities is dependent on the error in the A operator and the residual of the nonlinear problem. For quasi-Newton solvers like the ones that are the thrust of this work, this indicates the convergence is a multiple of the tolerance of the linear system and the convergence of the non-linear problem, and we will see this behavior borne out in the results section.

7 Impact of Appromixate Linearization on Sensitivity Accuracy

This section shows the effect of an approximate linearization on the accuracy of the sensitivity computation. The first two sets of results portray the effect of partial linear solves on the accuracy of the identity of the differentiation of the inverse matrix in the sensitivity computation shown in equation (27). The final set shows the impact of exactly linearizing the residual operator at every stage of an RK scheme when the gradients are only computed at the first stage and then frozen throughout the analysis. All cases were run on an unstructured triangular mesh consisting of 4212 elements shown in Figure (1), in $M = .7$ flow with $\alpha = 2^o$.

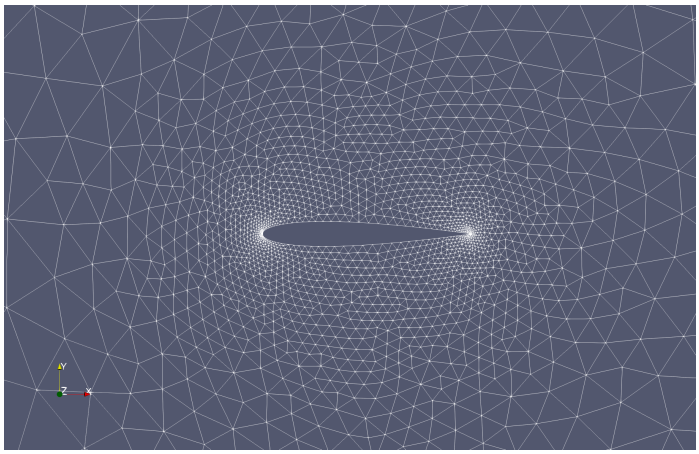
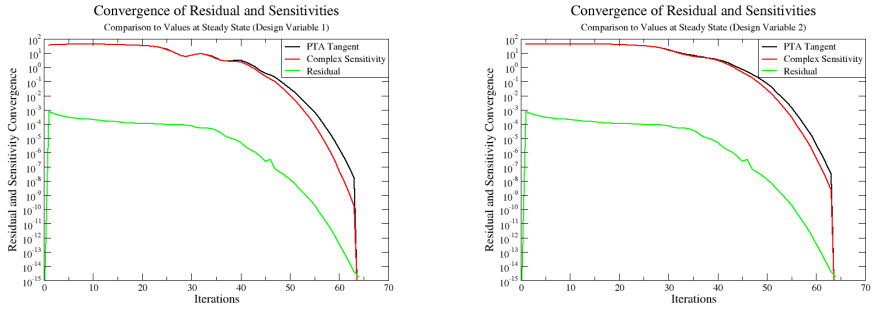


Fig. 1 Computational mesh for NACA0012 airfoil

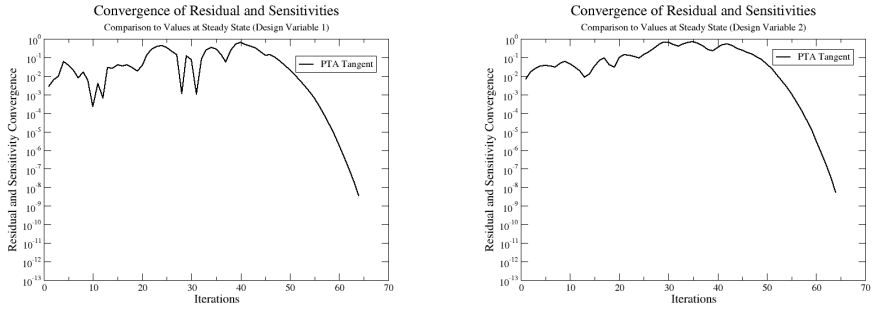
7.1 Results for an Exact Jacobian Augmented with a Mass Matrix

Here we look at the results for when the left hand side is the exact linearization of the residual operator augmented with a suitable mass matrix. We can see in Figure (2) that for the linear tolerance $1e-1$ that the complex and tangent sensitivities converge to their final values at the same rate as the analysis problem itself, which is expected based on the formulation. We can then see in Figure (3), which depicts the maximum difference between the complex and tangent sensitivities over the iteration history of the analysis solution process, that the maximum difference is of the order of the linear tolerance of the linear system. Furthermore, as the



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

Fig. 2 Sensitivity convergence for linear tolerance, $1e-1$: difference between current and final sensitivity values



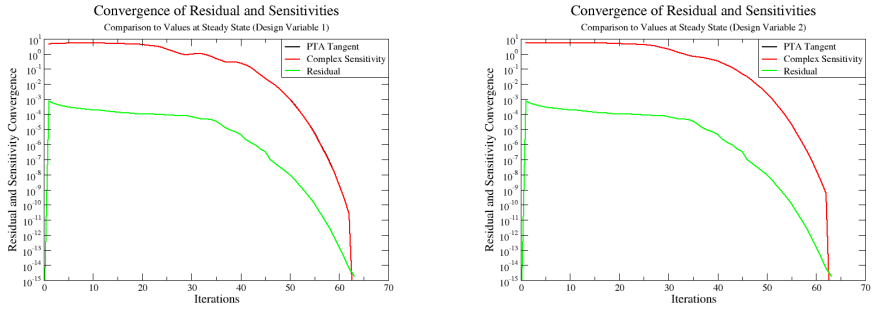
(a) Design Variable 1 (b) Design Variable 2

Fig. 3 Iterative sensitivity difference for linear tolerance, $1e-1$

analysis problem converges, so do the complex and tangent sensitivities to each other despite the inexact differentiation. We can see that at full convergence of the analysis problem, the tangent and complex-step sensitivities correspond to each other to a high degree of precision and these would also correspond to the steady-state tangent and adjoint computed sensitivities linearized about the converged analysis state.

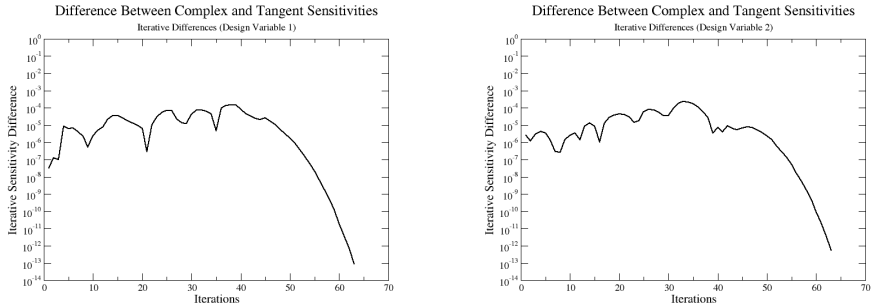
Figure (4) contains the same information as Figure (2), and Figure (3) is the sister plot of Figure (5) but the two later plots show resuls with a tighter linear system tolerance of $1e-4$. We can see that as before the maximum iterative difference is again on the order of the linear system tolerance, and that as the analysis converges so do the tangent and complex sensitivities to each other, down to nearly machine precision.

Having seen the impact of the tighter linear system tolerance on the maximum iterative difference between the tangent and complex sensitivities, we simulate the analysis problem with linear tolerances at every order from $1e-1$ to $1e-13$, and plot the maximum iterative difference in Figure (6). The maximum iterative difference is directly related to the linear system tolerance. This allows for good



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

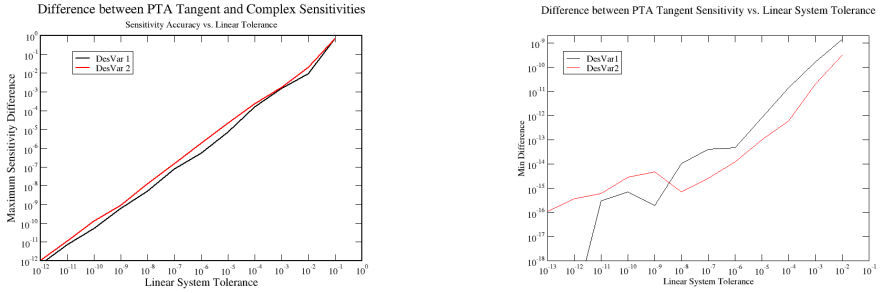
Fig. 4 Sensitivity convergence for linear tolerance, $1e-4$: difference between current and final sensitivity values



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

Fig. 5 Iterative sensitivity difference for linear tolerance, $1e-4$

estimates of the maximum iterative error as a function of the linear system tolerance. The minimum iterative difference shows similar behavior, but it is a function of the linear tolerance and the convergence of the non-linear problem.

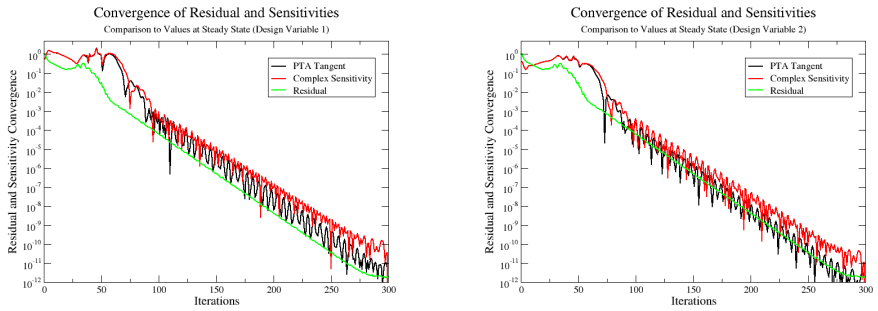


(a) Maximum Iterative Difference (b) Minimum Iterative Difference

Fig. 6 Iterative difference vs. linear tolerance

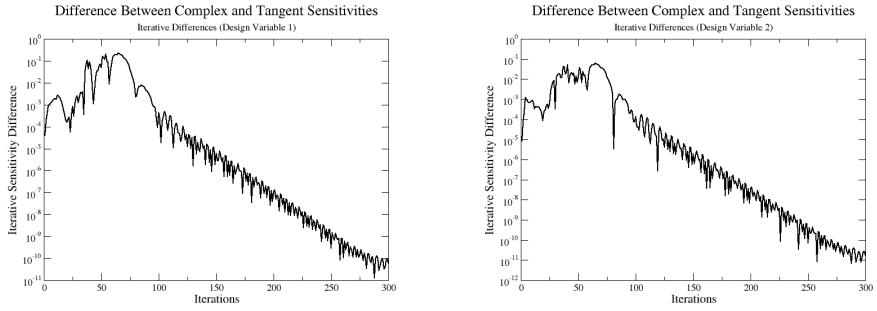
7.2 Results for an Inexact Jacobian Augmented with a Mass Matrix

Here we look at the results for when the left hand side is the first order linearization of the 2nd order residual operator augmented with a suitable mass matrix. The expectation is that we see similar behavior to that seen in the previous section. The difference being, that rather than have quadratic convergence in the primal and quadratic convergence of the inexact linearization to the complex linearization, we'd expect to see linear convergence similar to that of the analysis for the inexact newton solver. To demonstrate this we show sister plots to those of the previous section, but for the inexact newton runs. Figures (7, 8) show the behavior for a linear tolerance of $1e-1$ and Figures (9, 10) show the behavior for a tolerance of $1e-4$. These are the same tolerances we portrayed in the previous section, and we can see the same expected convergence of the inexactly linearized tangent to the complex linearization. We can see again, and as hypothesized by the error bounds in this work, that the maximum iterative difference is again on the order of the linear system tolerance, and that as the analysis converges so do the tangent and complex sensitivities to each other, down to nearly machine precision.



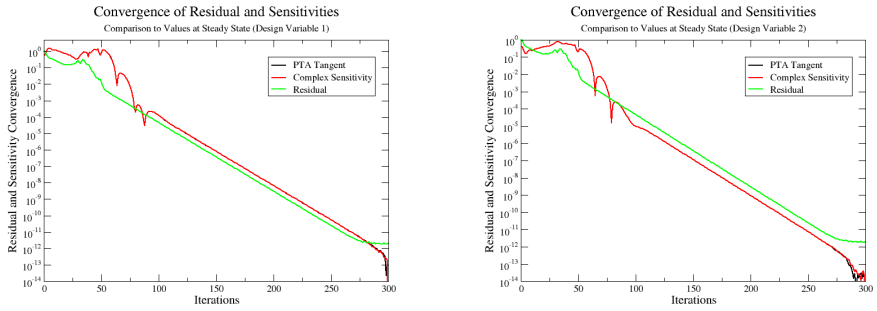
(a) Design Variable 1 (b) Design Variable 2

Fig. 7 Sensitivity convergence for linear tolerance, $1e-1$: difference between current and final sensitivity values



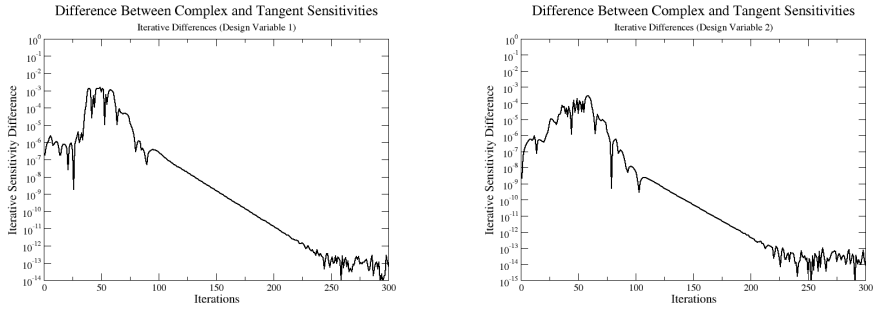
(a) Design Variable 1 (b) Design Variable 2

Fig. 8 Iterative sensitivity difference for linear tolerance, $1e-1$



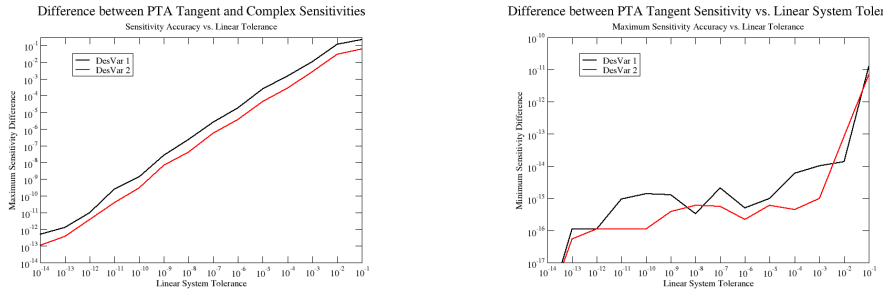
(a) Design Variable 1 (b) Design Variable 2

Fig. 9 Sensitivity convergence for linear tolerance, $1e-4$: difference between current and final sensitivity values



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

Fig. 10 Iterative sensitivity difference for linear tolerance, $1e-4$



(a) Maximum Iterative Difference

(b) Minimum Iterative Difference

Fig. 11 Iterative difference vs. linear tolerance

Having seen the impact of the tighter linear system tolerance on the maximum iterative difference between the tangent and complex sensitivities, and seeing the expected behavior as shown in the theoretical error bound and the previous section, we then simulate to check the impact of linear tolerances from $1e-1$ to $1e-14$. When we plot the maximum iterative difference in Figure (11) we see the expected behavior over that parameter sweep. This confirms the theoretical bound for a more general solver, one in which the left hand side isn't an exact linearization of the right hand side.

7.3 Results for Inexactly Linearized Explicit Runge-Kutta Solver

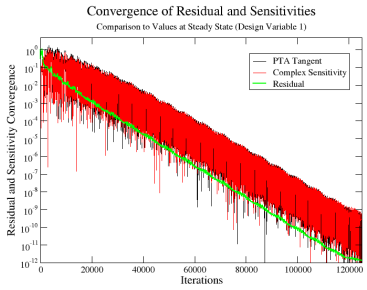
In this section we look at an explicit runge-kutta solver expressed below, iterating n through pseudo time and k through stages 1 to 5:

$$u^{n,k} = u^{n,0} + CFL\alpha^{k-1}\Delta t R(u^{n,k-1}) \tag{81}$$

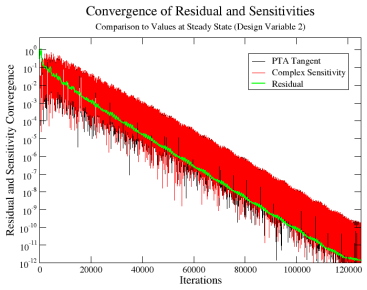
with the end of the sub-stage time-stepping being governed as follows.

$$u^{n,0} = u^{n-1,5} \tag{82}$$

To check the effect of inexact linearization of the fixed point iterations, we calculate the flow gradients for the 0 stage, and then hold them frozen through the fixed point iteration that solves the primal problem. For the linearization of the fixed point iteration we still update the gradients at each step and compute the jacobian of the flow accordingly. This allows us to look at the runge-kutta scheme as some A operator depending on the flow state and design variables that multiplies the 0 stage residual to get the final stage δu, where the linearization of A is inexact due to the inconsistent handling of the gradients. When we look at the convergence of the flow sensitivities to their respective final values as we did in the two previous sections, Figure (12) shows them converging to their respective final values as the primal problem converges.
Looking at the difference between the complex (exact) linearization and the approximate tangent linearization portrayed in Figure (13) shows the expected behavior. We see the error due to the inexact linearization of the Runge-Kutta

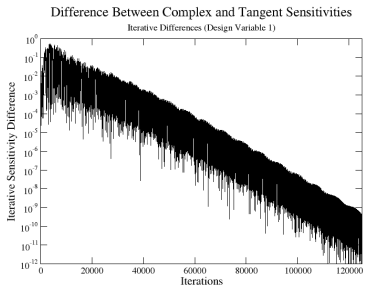


(a) Maximum Iterative Difference

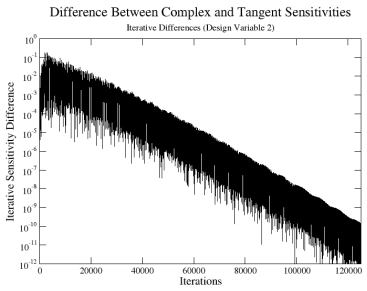


(b) Minimum Iterative Difference

Fig. 12 Runge Kutta Sensitivity Convergence



(a) Maximum Iterative Difference



(b) Minimum Iterative Difference

Fig. 13 Runge Kutta Sensitivity Difference

Scheme goes away at the rate of the primal problem convergence as shown in the proof section and the previous results.

8 Conclusion

We showed that the error between the approximate linearization and the complex (exact) linearization of a fixed point iteration is a function of both the satisfaction of the discretized PDE denoted by the residual and the accuracy of the approximation itself. We proved that in the limit of machine zero this approximation error vanishes and that we can obtain reasonable sensitivities for optimization without exact linearization of the fixed point iteration or full convergence of the non-linear problem, which will scale with the error in the discretized PDE. Finally, we demonstrated this behavior in a CFD code for an exact quasi-Newton solver, an inexact quasi-Newton solver and a five stage low-storage explicit Runge-Kutta solver.

9 Acknowledgments

This work was supported in part by NASA Grant NNX16AT23H and the NASA Graduate Aeronautics Scholars Program. Computing time was provided by ARCC on the Teton supercomputer.

References

1. Brown, D.A., Nadarajah, S.: An adaptive constraint tolerance method for optimization algorithms based on the discrete adjoint method DOI 10.2514/6.2018-0414. URL https://arc.aiaa.org/doi/abs/10.2514/6.2018-0414. 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
2. Gill, P.E., Murray, W., Saunders, M.A.: Snopt: An sqp algorithm for large-scale constrained optimization. SIAM review **47**(1), 99–131 (2005)
3. Günther, S., Gauger, N.R., Wang, Q.: Simultaneous single-step one-shot optimization with unsteady pdes. J. Comput. Appl. Math. **294**, 12–22 (2016)
4. Krakos, J.A., Darmofal, D.L.: Effect of small-scale output unsteadiness on adjoint-based sensitivity. AIAA Journal **48**(11), 2611–2623 (2010). DOI 10.2514/1.J050412. URL https://doi.org/10.2514/1.J050412
5. Krakos, J.A., Wang, Q., Hall, S.R., Darmofal, D.L.: Sensitivity analysis of limit cycle oscillations. Journal of Computational Physics **231**(8), 3228 – 3245 (2012). DOI https://doi.org/10.1016/j.jcp.2012.01.001. URL http://www.sciencedirect.com/science/article/pii/S0021999112000071
6. van Leer, B.: Flux-vector splitting for the euler equations. In: E. Krause (ed.) Eighth International Conference on Numerical Methods in Fluid Dynamics, pp. 507–512. Springer Berlin Heidelberg, Berlin, Heidelberg (1982)
7. LeVeque, R.J.: Numerical Methods for Conservation Laws, vol. 3. Springer (1992)
8. Mavriplis, D.: Revisiting the least-squares procedure for gradient reconstruction on unstructured meshes DOI 10.2514/6.2003-3986. URL https://arc.aiaa.org/doi/abs/10.2514/6.2003-3986. 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
9. Mavriplis, D.: A residual smoothing strategy for accelerating newton method continuation. arXiv preprint arXiv:1805.03756 (2018)
10. Mavriplis, D.J.: Vki lecture series: 38th advanced computational fluid dynamics. adjoint methods and their application in cfd, time dependent adjoint methods for single and multi-disciplinary problems (2015)
11. Nadarajah, S.K.: The Discrete Adjoint Approach to Aerodynamic Shape Optimization, Ph.D. Dissertation. Department of Aeronautics and Astronautics, Stanford University, USA (2003)
12. Nemec, M., Aftosmis, M.J.: Toward automatic verification of goal-oriented flow simulations. Tech. Rep. Tech. Rep. TM-2014-218386, NASA Ames Research Center (2014). URL https://ntrs.nasa.gov/search.jsp?R=20150000864
13. Padway, E., Mavriplis, D.J.: Advances in the pseudo-time accurate formulation of the adjoint and tangent systems for sensitivity computation and design DOI doi:10.2514/6.2020-3136. URL https://arc.aiaa.org/doi/abs/10.2514/6.2020-3136. 59th AIAA Aerospace Sciences Meeting, AIAA Paper 2020-3136, Virtual Event, June 2020. https://doi.org/10.2514/6.2020-3136
14. Padway, E., Mavriplis, D.J.: Application of the pseudo-time accurate formulation of the adjoint to output-based adaptive mesh refinement DOI doi:10.2514/6.2021-?? URL https://arc.aiaa.org/doi/abs/10.2514/6.2021-?? 59th AIAA Aerospace Sciences Meeting, AIAA Paper 2021-??, Virtual Event, January 2021. https://doi.org/10.2514/6.2020-3136
15. Padway, E., Mavriplis, D.J.: Toward a pseudo-time accurate formulation of the adjoint and tangent systems DOI 10.2514/6.2019-0699. URL https://arc.aiaa.org/doi/abs/10.2514/6.2019-0699. 57th AIAA Aerospace Sciences Meeting, AIAA Paper 2019-0699, San Diego CA, January 2019. https://doi.org/10.2514/6.2019-0699

16. Roe, P.: Approximate riemann solvers, parameter vectors, and difference schemes. Journal of Computational Physics **135**(2), 250 – 258 (1997). DOI https://doi.org/10.1006/jcph.1997.5705. URL http://www.sciencedirect.com/science/article/pii/S0021999197957053
17. Saad, Y.: Iterative methods for sparse linear systems, vol. 82. Society for Industrial and Applied Mathematics (2003). DOI 10.1137/1.9780898718003
18. Venditti, D.A., Darmofal, D.L.: Anisotropic grid adaptation for functional outputs: Application to two-dimensional viscous flows. J. Comput. Phys. **187**(1), 22–46 (2003). DOI 10.1016/S0021-9991(03)00074-3. URL https://doi.org/10.1016/S0021-9991(03)00074-3