AIAA SciTech Forum
January 3-7, 2022, San Diego, CA & Virtual
AIAA SCITECH 2022 Forum

10.2514/6.2022-1613

Inexact Linearization of Fixed Point Iterations in the Piggy-Back Iterations of the One-Shot Adjoint

Emmett Padway *
National Institute of Aerospace, Hampton, VA 23606
Dimitri Mavriplis †
Department of Mechanical Engineering, University of Wyoming, Laramie, WY 82071

This paper builds on previous work on inexact linearizations in the context of the pseudo-time accurate (or black-box) adjoint and applies it to the "piggy-back" iterations used in the "one-shot" adjoint. We present proofs that inexact linearization of the fixed point iterations used to solve the primal problem still allow for exact computation of the adjoint computed sensitivities for converged nonlinear problems. We verify the implementation of the sensitivity computation and then show optimizations using these methods for converging and nonconverging simulations.

I. Introduction

Multidisciplinary Design Optimization (MDO) has become a larger part of the expanding field of Computational Fluid Dynamics (CFD) as computers and algorithms have developed. 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 researchers to provide sensitivities through a selection of commonly used methods: 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. As the adjoint method has been more widely adopted, research has been conducted into more intricate formulations that would allow for cheaper designs or more robust adjoint computation. The One-Shot adjoint method was pioneered by Ta'asan,[5] and used to develop a unified multigrid solver to the nonlinear, adjoint and design problems all at once, greatly decreasing the cost of design by using an optimal solver for all problems and by not having to converge the nonlinear and adjoint problems completely for each design cycle iteration in contrast to the typical or "nested" approach. This method demands considerable difficulty in implementation and since a multigrid solver is used, it can be very difficult to maintain robustness. Shi-Dong et al.[6] develop a scalable method for the full-space optimization problem, solving the constraint, adjoint, and design problems all at once using a Newton algorithm for all three; this is highly desirable, but does require the second derivatives of the residual operator and the objective function and requires significant implementation efforts and as well as preconditioners that work on all three problems. Gunther et al.[7] developed the piggy-back iterations for the one-shot adjoint, which use the linearization of the fixed-point iteration to solve the adjoint problem. This leverages the Banach Fixed-Point Theorem to prove that the adjoint problem will converge at the same rate as the nonlinear problem; but it suffers from having to linearize the fixed-point iteration used for the nonlinear problem, either by hand- or automatic-differentiation. These guarantees on convergence are desirable but the differentiation of Newton-type solvers is very difficult due to the difficulties of differentating the linear solver which is path-dependent (i.e. Krylov solvers) and often not solved to machine precision. While research is underway into

*Research Engineer, Member AIAA; email: emmett.padway@nianet.org
†Professor, AIAA Associate Fellow; email: mavripl@uwyo.edu

the best way to differentiate the fixed-point iterations that include a linear solve,[8] in the meantime this is a barrier to applications of the piggy-back iterations to more robust and scalable fixed-point iterations.
This work is an outgrowth of the work of Padway and Mavriplis[9,10] on the pseudo-time accurate adjoint, also referred to as the black-box adjoint.[11] These works looked at calculating the adjoint sensitivities by linearizing the full iterative history of the nonlinear solver to get reliable adjoint sensitivities even when the constraint equation could not be solved. The work of Padway and Mavriplis[10] looked specifically at the behavior of inexact quasi-Newton algorithms and possible approximations that could be made to allow for differentiation of the linear solver algorithm, and they found (through numerical experiment) that by converging the nonlinear problem the error in the sensitivities due to these approximations vanished. Later work by Padway and Mavriplis[12] proved why this was the case and showed that in general for fixed-point iterations that could be viewed as an operator acting on the residual that, providing the residual was appropriately linearized, the error in the sensitivities would vanish as the nonlinear problem converges. This paper will show a similar proof to show that by converging the constraint equations the error in the sensitivities calculated by the piggy-back iterations using inexact linearization will vanish, thus providing a feasible way to extend piggy-back iterations to Newton-type solvers. We will then verify the implementation by comparing the adjoint sensitivities computed by the piggy-back iterations to the steady-state adjoint computed sensitivities and complex-step finite difference computed sensitivities. Finally we apply this method to transonic and supersonic optimization test cases to demonstrate the capabilities of these techniques, in well-converging, stalled, and oscillatory nonlinear problems.

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 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.

II.B. Spatial Discretization

The residual about the closed control volume is given as:

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

This equation gives the operator in the requirement for the adjoint and tangent systems, namely that $R = 0$. In the discretized form of the residual operator, x denotes the vector of 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.[13] The solver has three different flux calculations implemented and linearized, these are the Lax-Friedrichs,[14] Roe,[15] and van Leer[16] schemes. In this work, only the Lax-Friedrichs and Van-Leer schemes are used. In order to slope limit the solution reconstruction near shock discontinuties, a modified Venkatakrishnan limiter is used; Venkatakrishnan's limiter[17] is modified in a few important ways as detailed in previous works.[10]

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 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:

$$u^k = u^{k-1} + \Delta u \tag{4}$$

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

$$[P]\,\Delta u = -R(u) \tag{5}$$

We can substitute our expression for Δu into the time-stepping equation (4) to obtain our final form of this equation.

$$u^{k+1} = u^k - [P_k]^{-1}\,R \tag{6}$$

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 (7).

$$[P_k] = \left[\frac{\partial R}{\partial u^k}\right]_1 + \frac{vol}{\Delta t CFL} \tag{7}$$

Where vol is the area of the cell and CFL is a coefficient used to scale the time-step for explicit problems to satisfy the CFL (Courant-Friedrichs-Lewy) condtion, or to increase the time-step in implicit iterations to improve robustness and convergence of Newton-type solvers. Please note the subscript on the Jacobian above denotes that it is the Jacobian of the first-order spatially accurate residual operator; 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:

$$\Delta t_i = \frac{r_i}{\sqrt{(u^2+v^2)}+c} \tag{8}$$

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:

$$CFL = min(\beta \cdot CFL, CFL_{max}) \tag{9}$$

or with a linesearch and CFL controller,[18,19] which seeks to minimize the L2 norm of the pseudo-temporal residual, defined as:

$$R_t(u + \alpha\Delta u) = \frac{vol}{\Delta t}\alpha\Delta u^n + R(u + \alpha\Delta u^n) \tag{10}$$

When the pseudo-temporal residual decreases, we consider this to be a satisfactory value for α and we change the CFL accordingly. The pseudocode explains the actual process for the linesearch and CFL controller. The parameters $iter_{max}$, c, α_{l_1}, α_{l_2}, β_1, and β_2 are all user defined input values, defaulted to 30, .9, .1, .75, .1, and 1.5 respectively. One benefit of this combined line-search and CFL controller is that we do not have to differentiate it, and can simply store the values of CFL and α and use these fixed values in the forward and reverse linearizations and still obtain machine-level correspondence when comparing those sensitivity values to those of the complex-step differentiated solution process. This is possible because for small enough perturbations that the CFL controller does not take another path, the combined CFL controller and line-search is a piece-wise constant function with a zero derivative.

Algorithm 1 CFL controller

```
1: procedure Line Search and CFL Controller
2:     $r_{t0} = \|R_t(u + \Delta u)\|_2$
3:     $r_{s0} = \|R(u)\|_2$
4:     $r_{s1} = \|R(u + \Delta u)\|_2$
5:     if $r_{s0} < r_{s1}$ then
6:         for $k = 1, ..., iter_{max}$ do
7:             $\alpha = c\alpha$
8:             $r_{t1} = \|R_t(u + \alpha\Delta u)\|_2$
9:             if $r_{t1} < r_{t0}$) then exit
10:    if $\alpha < \alpha_{l_1}$ then
11:        $\alpha = 0$
12:        $CFL = \beta_1 CFL$
13:    else if $\alpha_{l_1} < \alpha < \alpha_{l_2}$ then
14:        $CFL = CFL$
15:    else if $\alpha < \alpha_{l_2}$ then
16:        $CFL = min(\beta_2 CFL, CFL_{max})$
```

In order to solve the linear system we use a point-implicit Jacobi or point-implicit 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 based off the first-order accurate residual operator outlined in equation (7). Equation (5) is then solved iteratively as:

$$[D]\,\Delta(\Delta u)^l = -R(u) - [P_{k-1}]\,\Delta u^l \tag{11}$$

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

$$\Delta u^{l+1} = \Delta u^l + \omega(\Delta(\Delta u))^l \tag{12}$$

These linear solvers can also be applied as smoothers either to a BiCGStab or a GMRES linear solver. The flexible GMRES linear solver is the more commonly used one in this Newton-Krylov nonlinear solver, and the algorithm below shows its implementation, where the operator M^{-1} is the preconditioning matrix using the block Jacobi or block Gauss-Seidel relaxation schemes outlined above. The FGMRES solver outlined in algorithm (2)[20] is implemented to solve the stiff steady-state tangent and adjoint systems as well.

Algorithm 2 Flexible Restarted GMRES

```
1: procedure Flexible GMRES
2:     for $k = 1, ..., ncycles$ do
3:         $r_0 = b - Ax_0, \beta = \|r_0\|, v_1 = r_0/\beta$
4:         for $j = 1, ..., m$ do
5:             $z_j = M^{-1}v_j$
6:             $v_{j+1} = Az_j$
7:             for $i = 1, ..., j$ do
8:                 $h_{i,j} = (v_{j+1}, v_i)$
9:                 $v_{j+1} = v_{j+1} - h_{i,j}v_i$
10:            $h_{j+1,j} = \|v_{j+1}\|, v_{j+1} = v_{j+1}/h_{j+1,j}$
11:            Define $Z_m = [z_1, ..., z_m], \bar{H}_m = [h_{i,j}]_{1<i<j+1,1<j<m}$
12:        Solve least squares problem for $y_m$
13:        $x_0 = x_0 + Z_m y_m$
```

II.D. A Review of Tangent and Adjoint Systems

II.D.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 mesh nodal coordinates and specifically x_{surf} is the geometry coordinate vector and x_v is the volume mesh point coordinate vector. In order to obtain an expression for the sensitivities we take the derivative of the objective functional:[21]

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

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 calculated by solving the spring analogy mesh deformation equation:

$$[K]\frac{dx_v}{dD_i} = \frac{dx_s}{dD_i} \tag{14}$$

and we calculate $\frac{dx_s}{dD_j}$ through differentiating the shape design variables. For the global inverse distance weighted method[22] we can smoothly and explicitly propogate surface changes in the geometry onto the volume mesh, and for these the mesh sensitivities are computed directly as a function of the surface coordinate sensitivities:

$$\frac{dx_{v_i}}{dD_j} = \frac{\sum w_{ik}(\vec{r}_{ik})\frac{dx_{s_k}}{dD_j}}{\sum w_{ik}(\vec{r}_{ik})} \tag{15}$$

It is not possible to obtain $\frac{du}{dD}$ 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.

$$\left[\frac{\partial R}{\partial x}\right]\frac{dx}{dD} + \left[\frac{\partial R}{\partial u}\right]_2\frac{du}{dD} = 0 \tag{16}$$

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

$$\left[\frac{\partial R}{\partial u}\right]_2\frac{du}{dD} = -\left[\frac{\partial R}{\partial x}\right]\frac{dx}{dD} \tag{17}$$

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 $\left[\frac{\partial R}{\partial x}\right]\frac{dx}{dD}$ (which scales with the design variables), and obtain $\frac{du}{dD}$. We then substitute $\frac{du}{dD}$ into equation (13) to obtain the final sensitivities.

II.D.2. Discrete Adjoint Formulation

The adjoint formulation begins with the same sensitivity equation:

$$\frac{dL}{dD} = \frac{\partial L}{\partial x}\frac{dx}{dD} + \frac{\partial L}{\partial u}\frac{du}{dD} \tag{18}$$

Using the condition $R(u(D), D) = 0$, we return to equation (17) and pre-multiply both sides of the equation by the inverse Jacobian matrix to obtain:

$$\frac{du}{dD} = -\left[\frac{\partial R}{\partial u}\right]_2^{-1}\left[\frac{\partial R}{\partial x}\right]\frac{dx}{dD} \tag{19}$$

Substituting the above expression into the sensitivity equation yields:

$$\frac{dL}{dD} = \frac{\partial L}{\partial x}\frac{dx}{dD} - \frac{\partial L}{\partial u}\left[\frac{\partial R}{\partial u}\right]_2^{-1}\left[\frac{\partial R}{\partial x}\right]\frac{dx}{dD} \tag{20}$$

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

$$\mathbf{\Lambda}^T = -\frac{\partial L}{\partial u}\left[\frac{\partial R}{\partial u}\right]_2^{-1} \tag{21}$$

which gives an equation for the adjoint variable:

$$\left[\frac{\partial R}{\partial u}\right]_2^T \mathbf{\Lambda} = -\left[\frac{\partial L}{\partial u}\right]^T \tag{22}$$

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

$$\frac{dL}{dD} = \left[\frac{\partial L}{\partial x} + \mathbf{\Lambda}^T \frac{\partial R}{\partial x}\right]\frac{dx}{dD} \tag{23}$$

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

$$[K]^T \mathbf{\Lambda_x} = \left[\frac{\partial L}{\partial x}\right]^T + \left[\frac{\partial R}{\partial x}\right]^T \mathbf{\Lambda} \tag{24}$$

and the final expression for sensitivity is given as:

$$\frac{dL}{dD} = \mathbf{\Lambda_x}^T \frac{dx_s}{dD} \tag{25}$$

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

III. Development of the Inexactly Linearized Piggy-Back Iterations

In this section we develop the piggy-back iterations for the one shot adjoint. Note that this is a coupling of only the constraint and adjoint problem, it does not include the design problem. We begin from the understanding as in the steady state adjoint that we want the residual to be equal to 0, and this is the stationary point of the fixed-point iteration. Beginning from a general fixed point iteration defined as:

$$u^{k+1}(D) = N(u^k(D), D) = u^k(D) + H(u^k(D), D) \tag{26}$$

where:

$$H(u^k(D), D) = A(u^k, D)R(u^k(D), D) \tag{27}$$

and $A(u^k, D)$ is based on the choice of fixed point iteration. Using the constraint that for a stationary point (u^*) of the fixed-point iteration $G(u^*(D), D) - u^*(D) = 0$, we can proceed with a similar proof to that of the steady state adjoint shown previously. We begin from the objective function (L) augmented by a lagrangian vector (the adjoint variable vector) multiplying the constraint as shown below.

$$J(u^*(D), D) = L(u^*(D), D) + \Lambda^T(N(u^*(D), D) - u^*) \tag{28}$$

In order to find an optimal design the KKT conditions must be satisfied:

$$\begin{array}{ll} N(u^*(D), D) - u^* = 0 & \textit{State equation} \\ \frac{\partial L(u^*(D),D)}{\partial u} + \Lambda^T \frac{\partial(N(u^*(D),D)-u^*)}{\partial u} = 0 & \textit{Adjoint equation} \\ \frac{\partial L(u^*(D),D)}{\partial D} + \Lambda^T \frac{\partial(N(u^*(D),D))}{\partial D} = 0 & \textit{Design equation} \end{array} \tag{29}$$

We can evolve the adjoint and sensitivity equations through iteration-space as shown below

$$\begin{aligned} \Lambda^{k+1T} &= \frac{\partial L(u(D), D)}{\partial u} + \Lambda^{kT}\left(\frac{\partial N(u^k(D), D)}{\partial u}\right) \\ \frac{dL(u(D), D)}{dD} &= \frac{\partial L(u(D), D)}{\partial D} + \Lambda^{kT}\left(\frac{\partial N(u^k(D), D)}{\partial D}\right) \end{aligned} \tag{30}$$

and use our definition of the fixed point we can expand the above expressions.

$$\begin{aligned}\Lambda^{k+1T} &= \frac{\partial L(u(D),D)}{\partial u} + \Lambda^{kT}\left(\frac{\partial u^k + A(u^k(D),D)R(u^k(D),D)}{\partial u}\right)\\ \frac{dL(u(D),D)}{dD} &= \frac{\partial L(u(D),D)}{\partial D} + \Lambda^{kT}\left(\frac{\partial A(u^k(D),D)R(u^k(D),D)}{\partial D}\right)\end{aligned} \tag{31}$$

We can then obtain the equations below:

$$\begin{aligned}\Lambda^{k+1T} &= \frac{\partial L(u(D),D)}{\partial u} + \Lambda^{kT}\left(\frac{\partial A(u^k(D),D)R(u^k(D),D)}{\partial u}\right) + \Lambda^{kT}\\ \frac{dL(u(D),D)}{dD} &= \frac{\partial L(u(D),D)}{\partial D} + \Lambda^{kT}\left(\frac{\partial A(u^k(D),D)R(u^k(D),D)}{\partial D}\right)\end{aligned} \tag{32}$$

and expand the derivatives, we also drop the dependencies in the notation which returns the below equation.

$$\begin{aligned}\Lambda^{k+1T} &= \frac{\partial L}{\partial u} + \Lambda^{kT}\left(\frac{\partial A}{\partial u}R(u^k,D) + A(u^k,D)\frac{\partial R}{\partial u}\right) + \Lambda^{kT}\\ \frac{dL}{dD} &= \frac{\partial L}{\partial D} + \Lambda^{kT}\left(\frac{\partial A}{\partial D}R(u^k(D),D) + A(u^k,D)\frac{\partial R}{\partial D}\right)\end{aligned} \tag{33}$$

We assume that the linearization of the residual operator is correct, but that the linearization of the A operator is an approximation, denoted by tilde, and we get the following equation.

$$\begin{aligned}\tilde{\Lambda}^{k+1T} &= \frac{\partial L}{\partial u} + \tilde{\Lambda}^{kT}\left(\widetilde{\frac{\partial A}{\partial u}}R(u^k,D) + A(u^k(D),D)\frac{\partial R}{\partial u}\right) + \widetilde{\Lambda^{kT}}\\ \widetilde{\frac{dL}{dD}} &= \frac{\partial L}{\partial D} + \widetilde{\Lambda^{kT}}\left(\widetilde{\frac{\partial A}{\partial D}}R(u^k,D) + A(u^k(D),D)\frac{\partial R}{\partial D}\right)\end{aligned} \tag{34}$$

If we compare the exact and the approximate expressions for the sensitivities and subtract the two we obtain the following.

$$\begin{aligned}\epsilon_\Lambda^{k+1T} &= \Lambda^{kT}\left(\frac{\partial A}{\partial u}R(u^k,D) + A(u^k,D)\frac{\partial R}{\partial u}\right) + \Lambda^{kT} - \tilde{\Lambda}^{kT}\left(\widetilde{\frac{\partial A}{\partial u}}R(u^k,D) + A(u^k(D),D)\frac{\partial R}{\partial u}\right) - \widetilde{\Lambda^{kT}}\\ \epsilon_D^{k+1} &= \Lambda^{kT}\left(\frac{\partial A}{\partial D}R(u^k(D),D) + A(u^k,D)\frac{\partial R}{\partial D}\right) - \widetilde{\Lambda^{kT}}\left(\widetilde{\frac{\partial A}{\partial D}}R(u^k,D) + A(u^k(D),D)\frac{\partial R}{\partial D}\right)\end{aligned} \tag{35}$$

To proceed further we first must define three error terms for the errors at each nonlinear iteration in the piggy-back iterations of the adjoint, the linearization of the nonlinear iteration with respect to the state variable and the linearization with respect to the design variables, denoted by $\epsilon_\Lambda^k, \epsilon_u^k, \epsilon_D^k$ respectively.

$$\begin{aligned}\epsilon_\Lambda^k &= \Lambda^k - \tilde{\Lambda}^k\\ \epsilon_u^k &= \frac{\partial A^k}{\partial u^{k-1}} - \widetilde{\frac{\partial A^k}{\partial u^{k-1}}}\\ \epsilon_D^k &= \frac{dA^k}{dD} - \widetilde{\frac{\partial A^k}{\partial D}}\end{aligned} \tag{36}$$

We group like terms and use the ϵ_Λ^k terms and get the following expression:

$$\begin{aligned}\epsilon_\Lambda^{k+1T} &= \Lambda^{kT}\left(\frac{\partial A}{\partial u}R(u^k,D)\right) - \tilde{\Lambda}^{kT}\left(\widetilde{\frac{\partial A}{\partial u}}R(u^k,D)\right) + \epsilon_\Lambda^{kT}\left(A(u^k,D)\frac{\partial R}{\partial u} + I\right)\\ \frac{dL}{dD} - \widetilde{\frac{dL}{dD}} &= \Lambda^{kT}\left(\frac{\partial A}{\partial D}R(u^k,D)\right) - \tilde{\Lambda}^{kT}\left(\widetilde{\frac{\partial A}{\partial D}}R(u^k,D)\right) + \epsilon_\Lambda^{kT}\left(A(u^k,D)\frac{\partial R}{\partial D}\right)\end{aligned} \tag{37}$$

we can then use the definition of ϵ_u^k and ϵ_D^k and get the below expression which is only in terms of exact derivatives and error.

$$\begin{aligned} \epsilon_\Lambda^{k+1T} &= \Lambda^{kT}\left(\frac{\partial A}{\partial u}R(u^k,D)\right) - \tilde{\Lambda}^{kT}\left(\frac{\partial A}{\partial u} - \epsilon_u^k\right)R(u^k,D) + \epsilon_\Lambda^{kT}\left(A(u^k,D)\frac{\partial R}{\partial u} + I\right) \\ \frac{dL}{dD} - \widetilde{\frac{dL}{dD}} &= \Lambda^{kT}\left(\frac{\partial A}{\partial D}R(u^k,D)\right) - \tilde{\Lambda}^{kT}\left(\frac{\partial A}{\partial D} - \epsilon_D^k\right)R(u^k,D) + \epsilon_\Lambda^{kT}\left(A(u^k,D)\frac{\partial R}{\partial D}\right) \end{aligned} \tag{38}$$

We then rearrange one last time to get the following expression:

$$\begin{aligned} \epsilon_\Lambda^{k+1T} &= \tilde{\Lambda}^{kT}\epsilon_u^k R(u^k,D) + \epsilon_\Lambda^{kT}\left(\frac{\partial A}{\partial u}R(u^k,D) + A(u^k,D)\frac{\partial R}{\partial u} + I\right) \\ \frac{dL}{dD} - \widetilde{\frac{dL}{dD}} &= \tilde{\Lambda}^{kT}\epsilon_D^k R(u^k,D) + \epsilon_\Lambda^{kT}\left(\frac{\partial A}{\partial D}R(u^k,D) + A(u^k,D)\frac{\partial R}{\partial D}\right) \end{aligned} \tag{39}$$

and we note that the term multiplying ϵ_Λ^k is in fact the derivative of the fixed point iteration. We can group the terms in the parenthesis into the operator B where $B = \frac{\partial N}{\partial u}$ and by cauchy-schwarz inequality we know that:

$$\left\|\epsilon_\Lambda^{kT}\left(\frac{\partial A}{\partial u}R(u^k,D) + A(u^k,D)\frac{\partial R}{\partial u} + I\right)\right\| < \|B\|\left\|\epsilon_\Lambda^k\right\| \tag{40}$$

B is the derivative of the contractive fixed point iteration, therefore $\|B\| < 1$. As a result, the error in adjoint sensitivities expressed by ϵ_Λ decreases as the residual decreases and the contractivity of the fixed point iteration progresses through the primal solution process. The triangle inequality is used to show the pseudo-temporal evolution of the error as governed by the equation below.

$$\left\|\epsilon_\Lambda^{k+1}\right\| < \|B\|\left\|\epsilon_\Lambda^k\right\| + \left\|\tilde{\Lambda}^{kT}\epsilon_u^k\right\|\|R\| \tag{41}$$

We know by the contractivity of the fixed point iteration that R goes to 0, and therefore the error in the adjoint, ϵ_Λ^k, goes to 0. If we refer to the equation for the error in the sensitivities reproduced below we see that the error has one term multiplied by the residual and another multiplied by the error in the adjoint. The first goes to machine zero by definition of satisfaction of the fixed point, the second by the proof outlined above.

$$\frac{dL}{dD} - \widetilde{\frac{dL}{dD}} = \tilde{\Lambda}^{kT}\epsilon_D^k R(u^k,D) + \epsilon_\Lambda^{kT}\left(\frac{\partial A}{\partial D}R(u^k,D) + A(u^k,D)\frac{\partial R}{\partial D}\right) \tag{42}$$

Therefore inexact linearizations of the fixed point do not prevent accurate adjoint sensitivity computations at convergence of the nonlinear problem and should the nonlinear problem not be solved to machine precision the error depends on the level of convergence of the nonlinear problem. These conclusions had been demonstrated for the pseudo-time accurate formulation of the adjoint system,[12] and hypothesized there to apply to these cases and this analysis confirms that hypothesis. The motivation for this analysis was the difficulty in using the piggy-back iterations of the adjoint with implicit solvers due to issues with linearizing the fixed point iterations for these implicit solvers and so we must move to showing two approximate linearizations that are used in this work.

III.A. Newton-Chord Approximate Linearization

We refer back to equation (6) as the fixed point iteration that must be linearized as part of the piggy-back iterations. If we rewrite it as follows:

$$u^{k+1} = N(u) = u^k - [P_k]^{-1}R \tag{43}$$

and we can substitute the fixed point definition into the piggyback iterations.

$$\begin{aligned} \Lambda^{k+1T} &= \frac{\partial L}{\partial u} + \Lambda^{kT}\left(\frac{\partial u^k + [P_k]^{-1}R}{\partial u^k}\right) \\ \frac{dL}{dD} &= \frac{\partial L}{\partial D} + \Lambda^{kT}\left(\frac{\partial u^k + [P_k]^{-1}R}{\partial D}\right) \end{aligned} \tag{44}$$

For clarity, we transpose the first equation and eliminate $\frac{\partial u}{\partial D}$ as it is 0:

$$\begin{aligned}\Lambda^{k+1} &= \left(\frac{\partial u^k + [P_k]^{-1} R}{\partial u^k}\right)^T \Lambda^k + \frac{\partial L}{\partial u}^T \\ \frac{dL}{dD} &= \frac{\partial L}{\partial D} + \Lambda^{kT}\left(\frac{\partial [P_k]^{-1} R}{\partial D}\right)\end{aligned} \tag{45}$$

if we then expand the derivatives we can obtain better insight to the appropriate way to calculate this linearization.

$$\begin{aligned}\Lambda^{k+1} &= \left(I - [P_k]^{-1}\left[\frac{\partial R(u^k)}{\partial u^k}\right]_2 - \frac{\partial [P_k]^{-1}}{\partial u^k} R(u^k)\right)^T \Lambda^k + \frac{\partial L}{\partial u}^T \\ \frac{dL}{dD} &= \frac{\partial L}{\partial D} - \Lambda^{kT}\left([P_k]^{-1}\frac{\partial R(u^k)}{\partial D} + \frac{\partial [P_k]^{-1}}{\partial D} R(u^k)\right)\end{aligned} \tag{46}$$

We can then see the derivative of the preconditioner matrix, which is not trivial to compute. In this section we will proceed as though the preconditioner is fixed, making this the exact linearization of a Newton-Chord method, and an inexact linearization of an inexact-quasi-Newton solver that we will refer to as the Newton-Chord linearization (NC). Note this corresponds to the backward-Euler Lagrange-based formulation found in the literature.[11]

$$\begin{aligned}\Lambda^{k+1} &= \left(I - [P_k]^{-1}\left[\frac{\partial R(u^k)}{\partial u^k}\right]_2\right)^T \Lambda^k + \frac{\partial L}{\partial u}^T \\ \frac{dL}{dD} &= \frac{\partial L}{\partial D} - \Lambda^{kT}\left([P_k]^{-1}\frac{\partial R(u^k)}{\partial D}\right)\end{aligned} \tag{47}$$

It appears that we would have to solve for as many linear systems as design variables at each non-linear iteration, which is clearly undesirable, but if we rewrite the equations in a delta formulation and define a secondary adjoint variable we obtain:

$$\begin{aligned}[P_k]^T \Psi^k &= -\Lambda^k \\ \Delta\Lambda^k &= \left[\frac{\partial R(u^k)}{\partial u^k}\right]_2^T \Psi^k + \frac{\partial L}{\partial u}^T \\ \frac{dL}{dD} &= \frac{\partial L}{\partial D} + \Psi^{kT}\frac{\partial R(u^k)}{\partial D}\end{aligned} \tag{48}$$

As a result of this rearrangement of the terms we can see that we have as many linear system solver calls in the adjoint method as in the primal problem. It is important to note that this linearization is only exact for Newton-Chord solvers that solve the linear system using a relaxation scheme and that the same number of linear iterations must be done in the primal and adjoint problems and that the adjoint relaxation scheme must be the exact dual of the relaxation scheme used in the primal problem.

III.B. Inexact-quasi-Newton Approximate Linearization

We begin from equation (46) and here consider an attempt to linearize the preconditioner matrix. We use the expression for the analytic derivative of a matrix inverse shown below for a matrix K.

$$\frac{dK^{-1}}{dx} = -K^{-1}\frac{dK}{dx}K^{-1} \tag{49}$$

Substituting this into equation (46) we obtain the below expression:

$$\begin{aligned}\Lambda^{k+1} &= \left(I - [P_k]^{-1}\left[\frac{\partial R(u^k)}{\partial u^k}\right]_2 + [P_k]^{-1}\frac{d[P_k]}{du^k}[P_k]^{-1} R(u^k)\right)^T \Lambda^k + \frac{\partial L}{\partial u}^T \\ \frac{dL}{dD} &= \frac{\partial L}{\partial D} - \Lambda^{kT}\left([P_k]^{-1}\frac{\partial R(u^k)}{\partial D} + [P_k]^{-1}\frac{d[P_k]}{dD}[P_k]^{-1} R(u^k)\right)\end{aligned} \tag{50}$$

At first it would seem that we have four linear system solve per design variable per nonlinear iteration, but we can make judicious use of previously computed quantities and use some tricks to deal with this increased expense. First we use the fixed point iteration definition that $\Delta u = [P_k]^{-1} R(u^k)$ to simplify the above equation.

$$\begin{aligned}\Lambda^{k+1} &= \left(I - [P_k]^{-1}\left[\frac{\partial R(u^k)}{\partial u^k}\right]_2 + [P_k]^{-1}\frac{d\,[P_k]}{du^k}\Delta u\right)^T \Lambda^k + \frac{\partial L}{\partial u}^T \\ \frac{dL}{dD} &= \frac{\partial L}{\partial D} - \Lambda^{kT}\left([P_k]^{-1}\frac{\partial R(u^k)}{\partial D} + [P_k]^{-1}\frac{d\,[P_k]}{dD}\Delta u\right)\end{aligned} \tag{51}$$

Then, as in the Newton-Chord linearization we define a secondary adjoint variable and use a delta form of the problem.

$$\begin{aligned}[P_{k-1}]^T\,\Psi^k &= -\Lambda^k \\ \Delta\Lambda^k &= \left(\left[\frac{\partial R(u^k)}{\partial u^k}\right]_2 - \frac{d\,[P_k]}{du^k}\Delta u\right)^T \Psi^k + \frac{\partial L}{\partial u}^T \\ \frac{dL}{dD} &= \frac{\partial L}{\partial D} + \Psi^{kT}\left(\frac{\partial R(u^k)}{\partial D} - \frac{d\,[P_k]}{dD}\Delta u\right)\end{aligned} \tag{52}$$

As a result of this rearranging of the terms we can see that we have as many linear system solver calls in the adjoint method as in the primal problem. It is important to note that this linearization is only exact for exact solution of the linear system at each nonlinear step, but has no requirement on the primal, forward, and reverse linear solvers.

IV. Verification

In this section we will compare the sensitivities calculated using the "piggy-back" iterations with inexact linearizations to those computed by the steady-state adjoint and complex sensitivities for both inexact linearizations: the Newton-Chord and the inexact-quasi-Newton approximate linearizations. We tested these implementations on an unstructured mesh that consists of 4212 triangular elements shown in Figure 1. All verification cases were run with $Mach = .7$ and $\alpha = 2^o$. The design variables are two Hicks-Henne bump functions located at one third and two thirds of the chord length respectively and the volume mesh is moved using inverse distance weighting as outlined in previous works.[9]

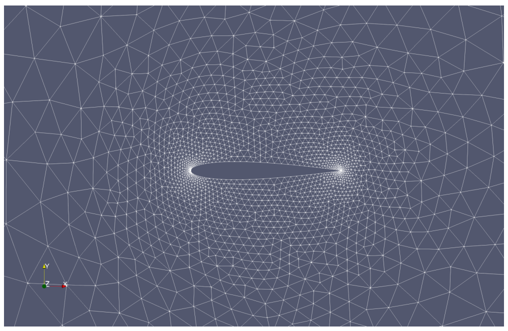


Figure 1. Computational mesh for NACA0012 airfoil

	Design Variable 1	Design Variable 2
Complex	-0.4507974688123841	0.4465241035044418
Newton-Chord	-0.450797468**9103023**	0.4465241035**326785**
Inexact-Quasi-Newton	-0.450797468**8999744**	0.4465241035**301284**
Steady State	-0.450797468**9185431**	0.4465241035**356137**

Table 1. Comparison of steady-state adjoint, piggy-back adjoint (inexact-quas-Newton and Newton-Chord) and complex-step computed sensitivities

We can see that the accuracy is comparable between the piggy-back and steady-state adjoint computation methods as compared to the complex-step finite-difference computed sensitivities; this is validation of this approach.

V. Results

The results section shows the results of three optimization cases, a weak shock transonic case, a standard transonic case and a supersonic case with a strong shock. The goal being to first show the desired behavior in cases where the nonlinear problem is solved easily. After which we show the more common use cases of transonic and supersonic optimization where we show the piggy-back iterations still manage to improve the performance of the airfoil even lacking convergence of the primal problem (and therefore a lack of convergence in the adjoint). All cases are run for a NACA0012 geometry using the mesh shown in Figure 1.

V.A. Weak Shock Transonic Case

This case was run at $M = .7$, $\alpha = 3^o$, where the goal is to minimize drag and increase lift. The objective function here is a composite objective function of lift and drag.

$$L = \omega_L (C_L - C_{L_T})^2 + \omega_D (C_D - C_{D_T})^2 \tag{53}$$

The targets for lift and drag are denoted by C_{L_T}, and C_{D_T} respectively. The target lift coefficient is set to .211 as it is larger than the value in the baseline simulation, the target drag coefficient is set to 0. The respective weights are set to 1.0 for all terms. This objective function will make the optimizer try to increase lift and minimize the drag. The design variables are 30 Hicks-Henne bump functions evenly distributed to the top and bottom surfaces of the airfoil facilitating asymmetric optimizations. We can see in Figure 2 that the nonlinear problem converges for the initial geometry, further investigations showed good convergence behavior for perturbed geometries as well. Figure 3 shows the behavior of the design cycle for all three methods, showing the convergence for optimality and objective functions as a function of both the number of major iterations and the number of function evaluations – where the optimality is computed as the norm of the sensitivities for design variables not at the limits of their values, if the sensitivities seek to push the design variable past its constraint limit it no longer contributes to the optimality. We can see very similar behavior between all three methods, with the two approximate linearizations being nearly identical to one another and close in behavior to the typical steady-state adjoint, which can be attributed to the convergence of the nonlinear problem, and therefore the error in the approximate linearization has shrunk significantly. We see excellent behavior in the convergence of the optimality and the objective function. Furthermore we can see the final geometries are very close in shape as well, with no noticeable differences between the two approximate linearizations and their close correspondance to the steady-state adjoint design, as shown in Figure 4. The contour plots in Figures 5 and 6 depict the density and Mach number plots for the baseline and optimized geometries, showing very similar behaviors for the optimized geometries with the shock strength decreasing and nearly vanishing due to the optimization process, which designs an asymmetric airfoil.

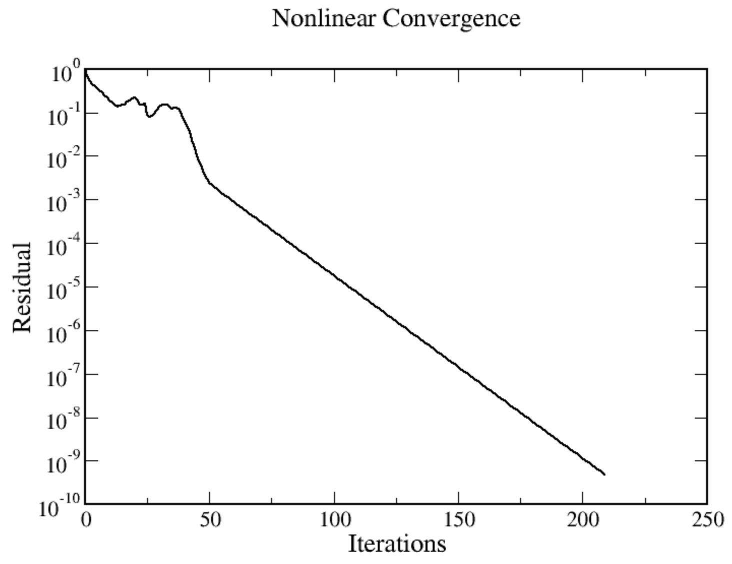
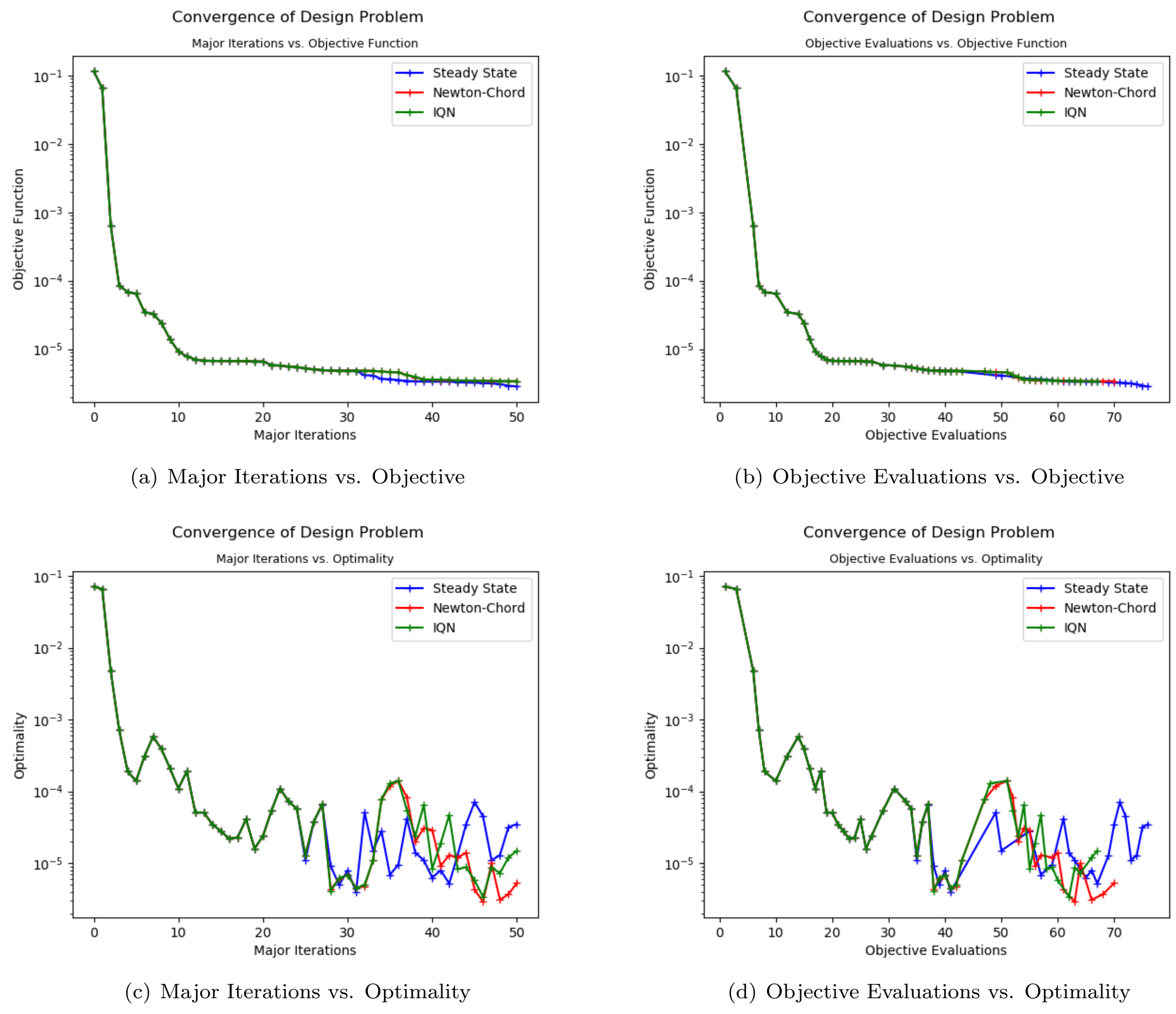


Figure 2. Nonlinear problem convergence for baseline geometry



(a) Major Iterations vs. Objective (b) Objective Evaluations vs. Objective

(c) Major Iterations vs. Optimality (d) Objective Evaluations vs. Optimality

Figure 3. Design cycle summary

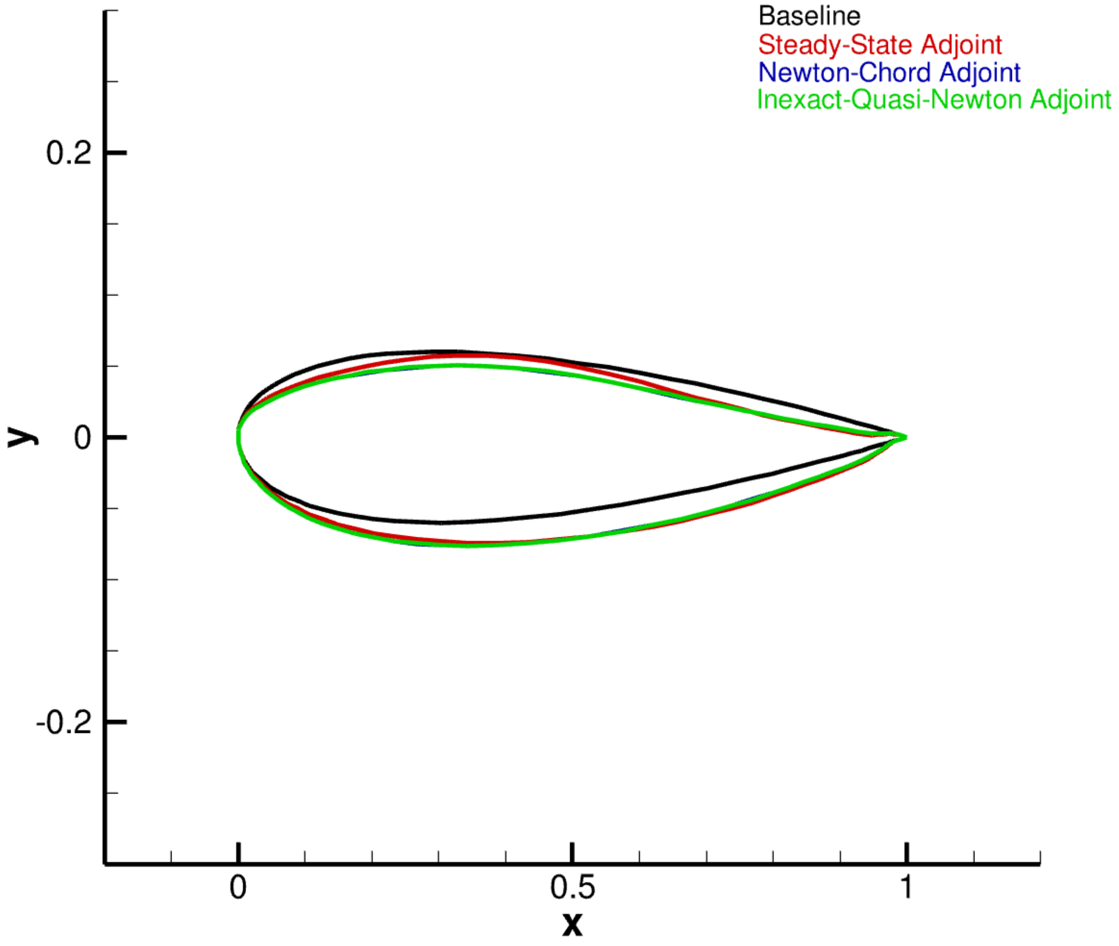
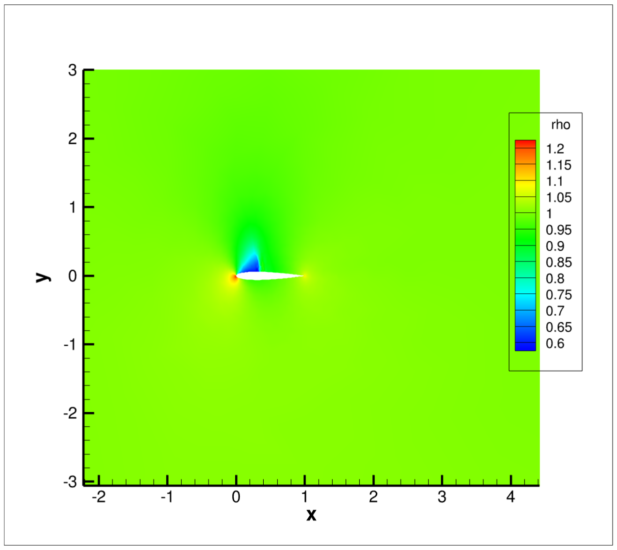
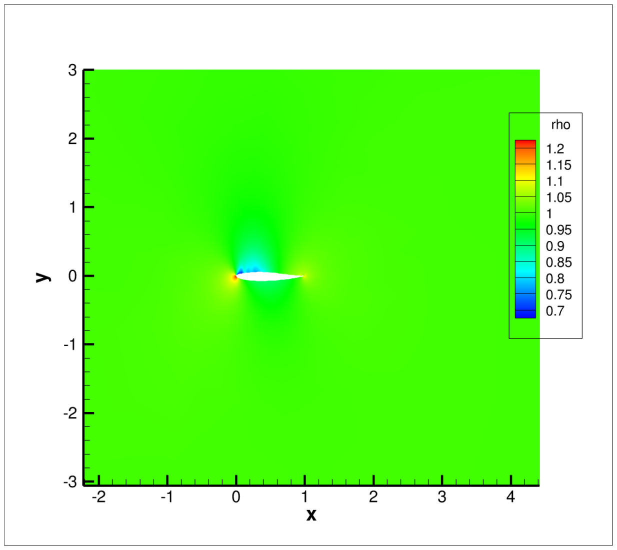


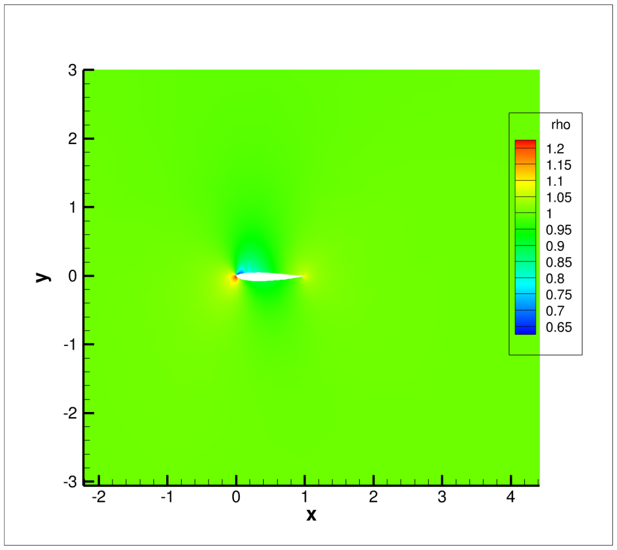
Figure 4. Geometry comparison



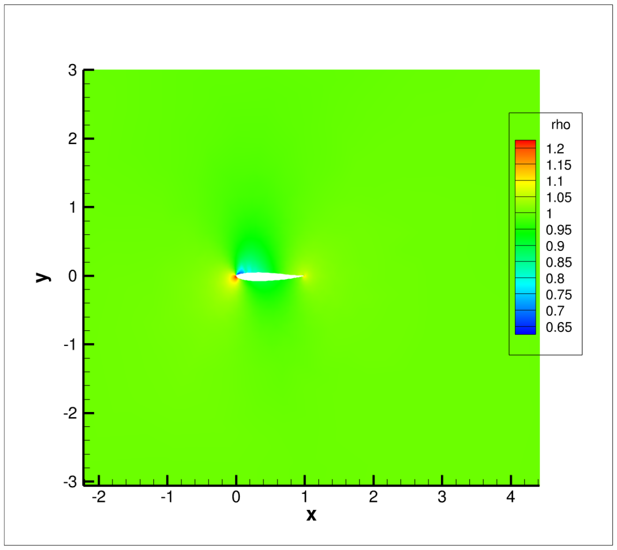
(a) Baseline Design



(b) Steady-State Optimized Design

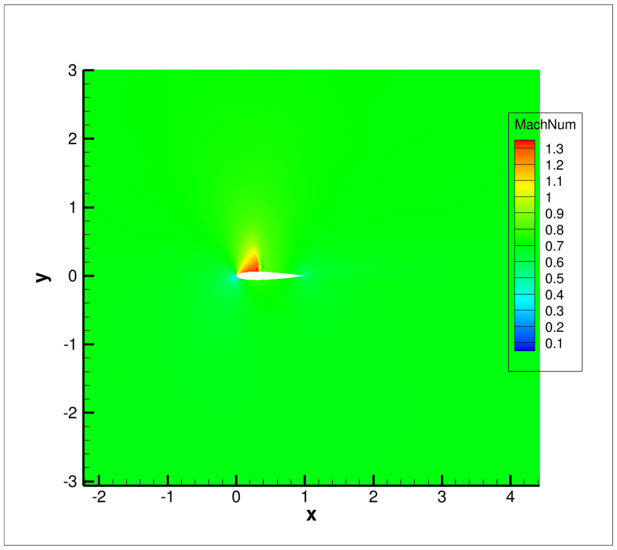


(c) NC Optimized Design

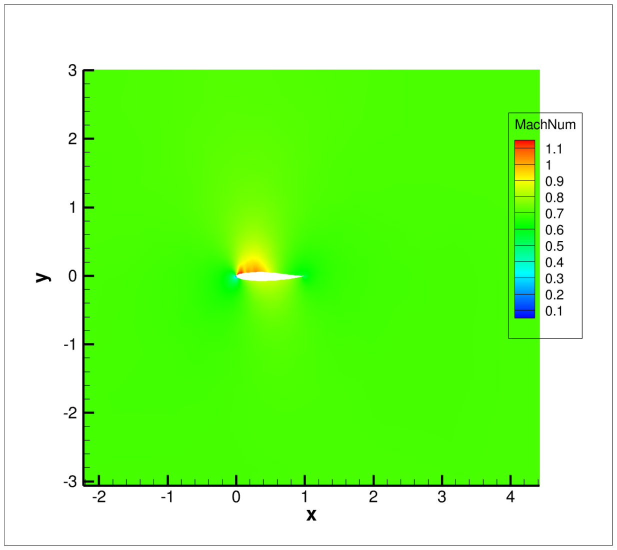


(d) IQN Optimized Design

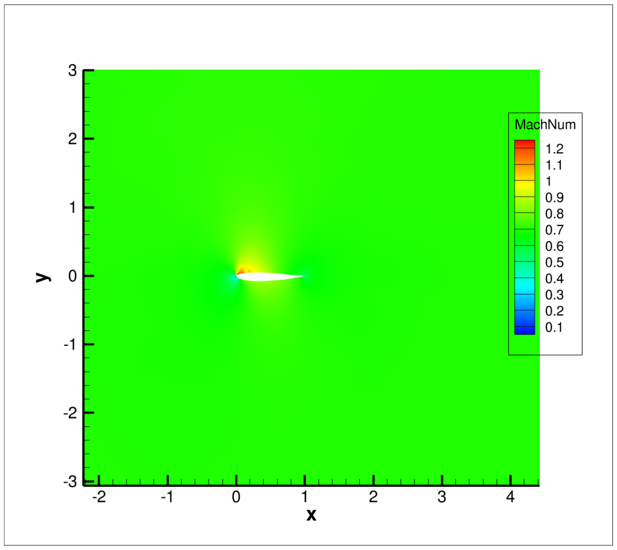
Figure 5. Density comparison



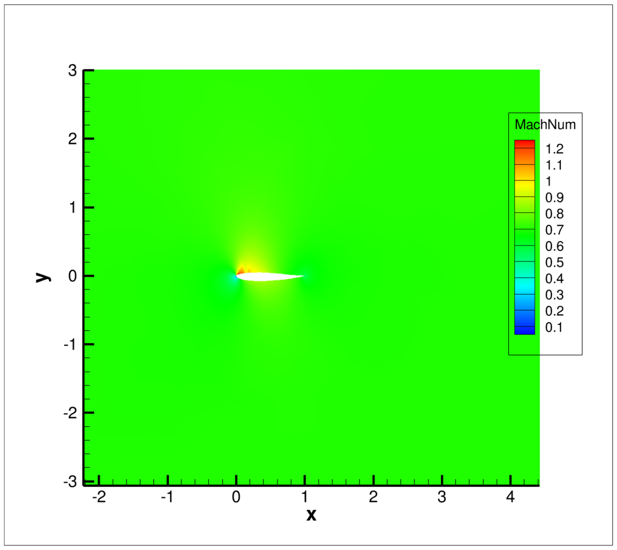
(a) Baseline Design



(b) Steady-State Optimized Design



(c) NC Optimized Design

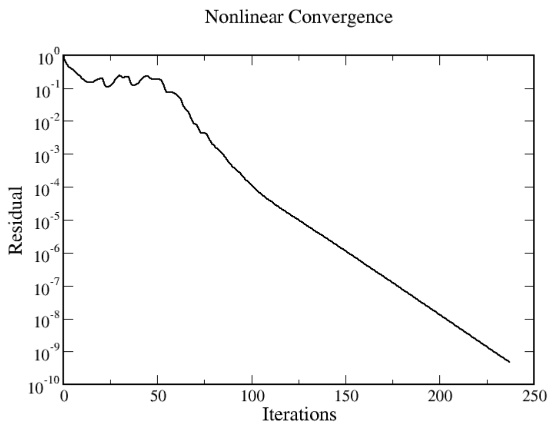


(d) IQN Optimized Design

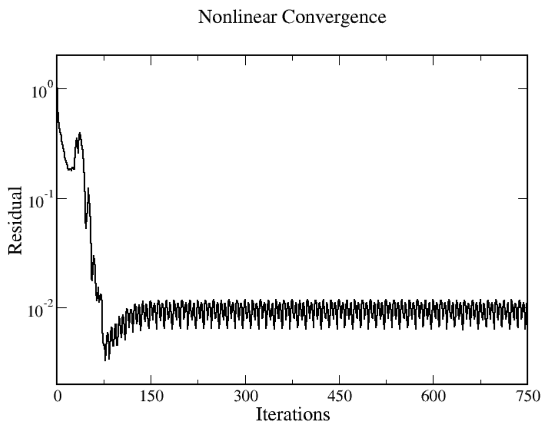
Figure 6. Mach number comparison

V.B. Standard Transonic Case

This case was run at $M = .8$, $\alpha = 1.25^o$, which is an oft-simulated transonic case, where the goal is to minimize drag with an objective function is C_D^2. The design variables are 15 Hicks-Henne bump functions used to symmetrically perturb the top and bottom of the airfoil, where the airfoil thickness can only be increased. We can see in Figure 7 that while the nonlinear problem converges for the initial geometry, it does not for a perturbed geometry and instead enters pronounced limit cycle oscillations. This case shows very interesting behavior in the optimality and the design itself. Figure 8 shows the behavior of the design cycle for all three methods, showing the convergence for optimality and objective functions as a function of both the number of major iterations and the number of function evaluations. We can see very similar behavior between the inexact linearizations in that they both show good convergence in the objective function and show very poor convergence in the optimality value; this can be attributed to the convergence issues of the nonlinear problem, which prevents the error in the inexact linearization from disappearing and combines with the large oscillations gives a great deal of dependence on the specific iterative behavior as demonstrated in previous works for the pseudo-time accurate adjoint.[9,10] The steady-state adjoint shows good objective function convergence, but very little convergence in optimality. We are seeing that the inexact linearizations in the piggy-back iterations combined with a poorly converging problem does lead to detrimental behavior in the optimality problem. This is interesting to note because it persists even as the objective function is optimized appropriately – faster even than the optimization driven by the steady-state adjoint. When we turn to the optimized geometries shown in Figure 9, we see very interesting results, where the steady-state adjoint and inexact-quasi-Newton linearization led to one design, and the Newton-Chord linearization led to another one; this is similar to previous results found in the ADODG workshop[23] which found different solutions for the same simulation leading to differing optimization behavior. The Newton-Chord method leads to a much thicker mid-chord profile as opposed to the other two methods; the contour plots in Figures 10 and 11 depict the density and Mach number plots for the baseline and optimized geometries, showing the varying behaviors. Despite the notably different designs, all three optimized geometries have weakened the shock by pushing it further aft along the chord by widening the tail of the airfoil, thereby decreasing the shock, and lowering drag.

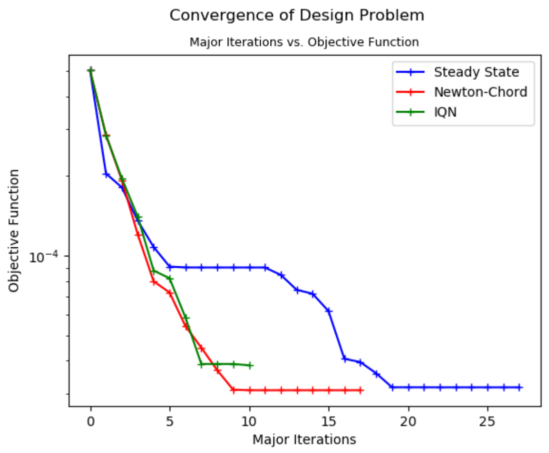


(a) Baseline Geometry Convergence

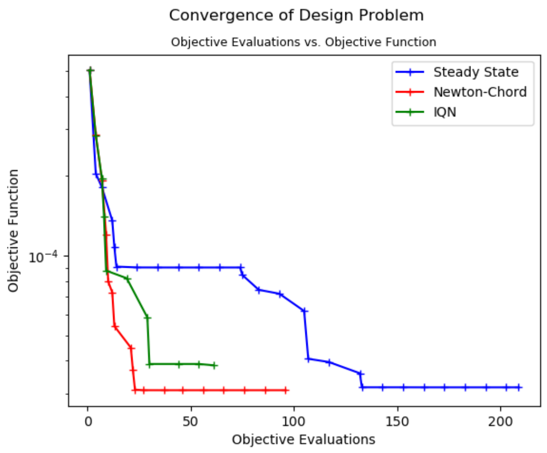


(b) Perturbed Geometry Convergence

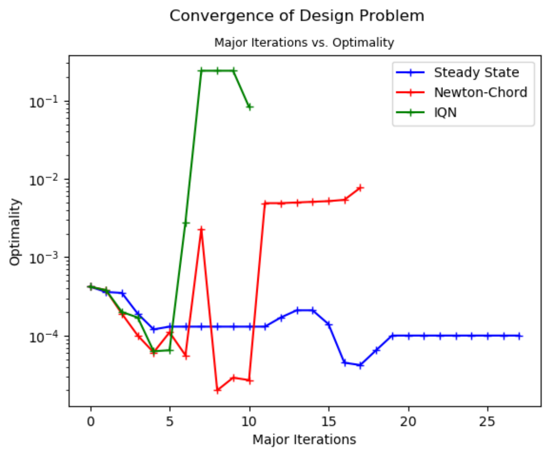
Figure 7. Nonlinear problem convergence for baseline and perturbed geometries



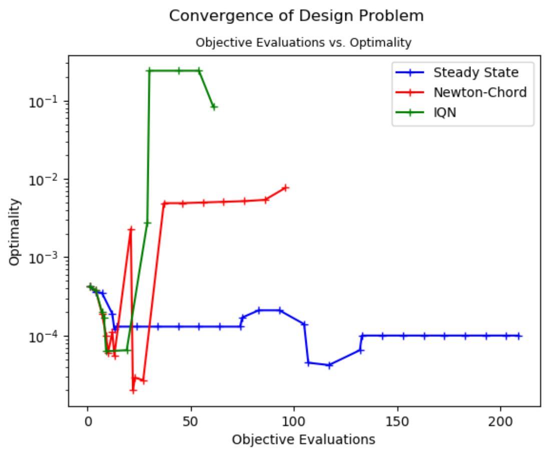
(a) Major Iterations vs. Objective



(b) Objective Evaluations vs. Objective



(c) Major Iterations vs. Optimality



(d) Objective Evaluations vs. Optimality

Figure 8. Design cycle summary

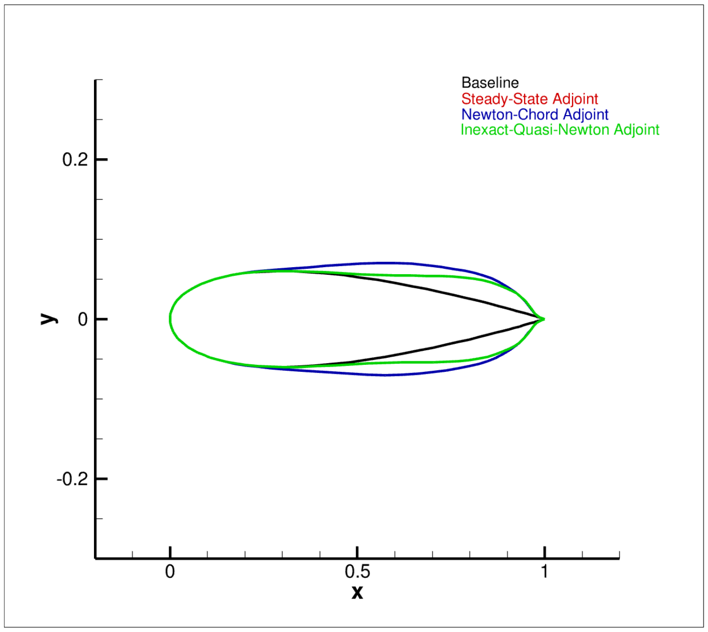
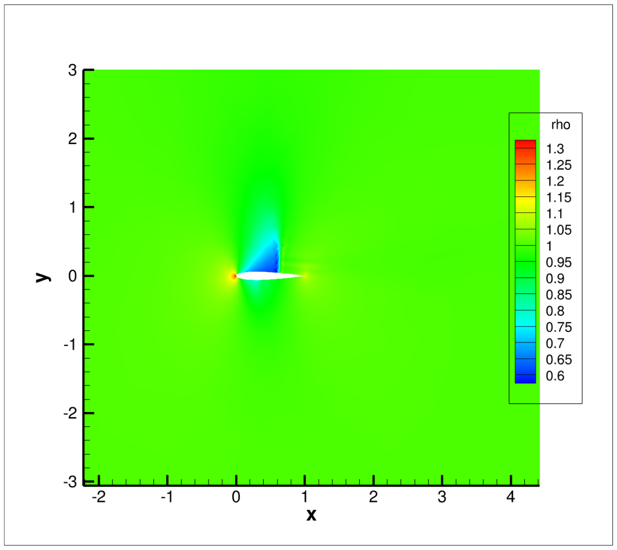
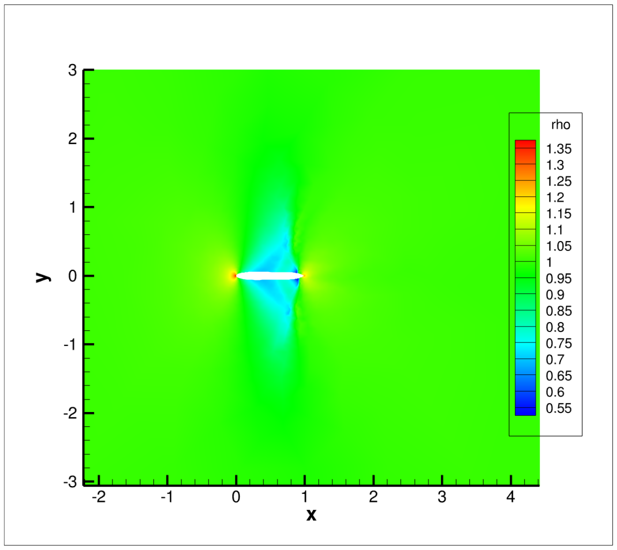


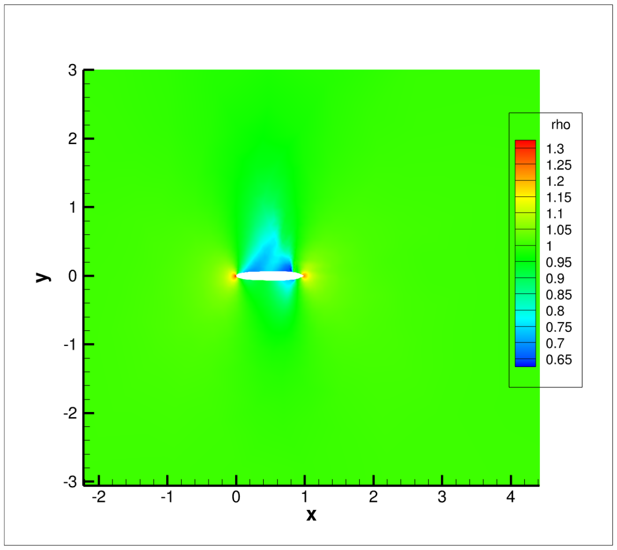
Figure 9. Geometry comparison



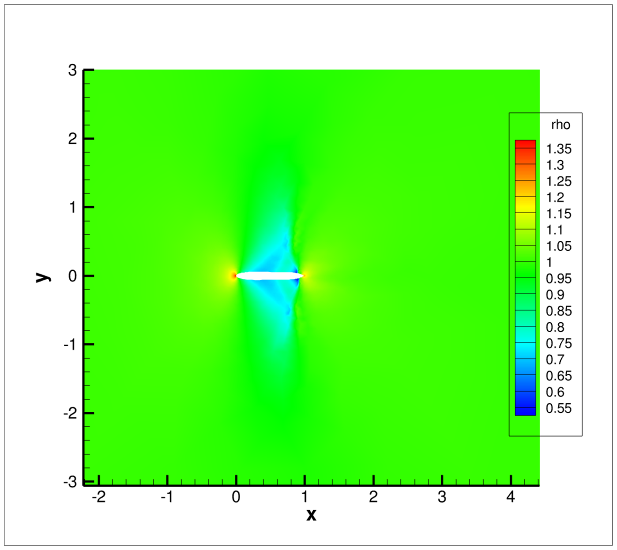
(a) Baseline Design



(b) Steady-State Optimized Design

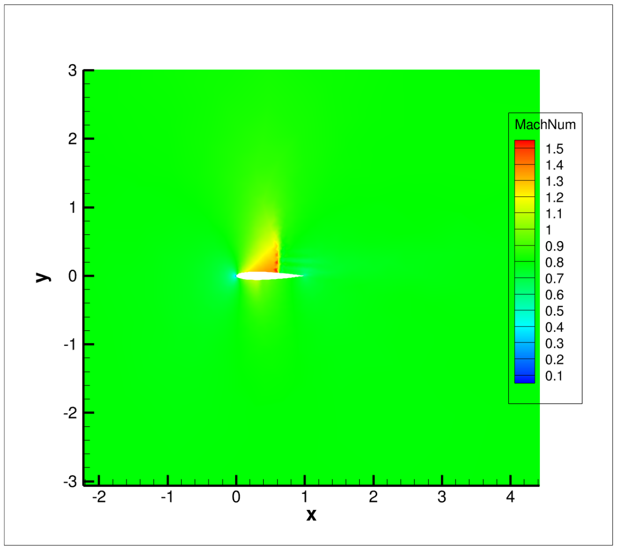


(c) NC Optimized Design

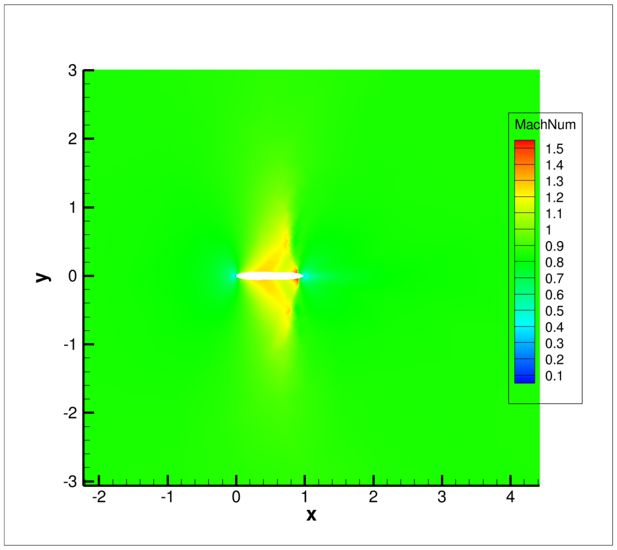


(d) IQN Optimized Design

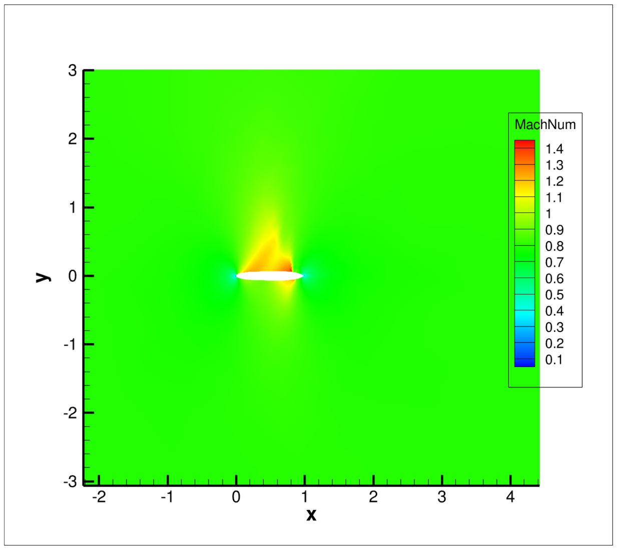
Figure 10. Density comparison



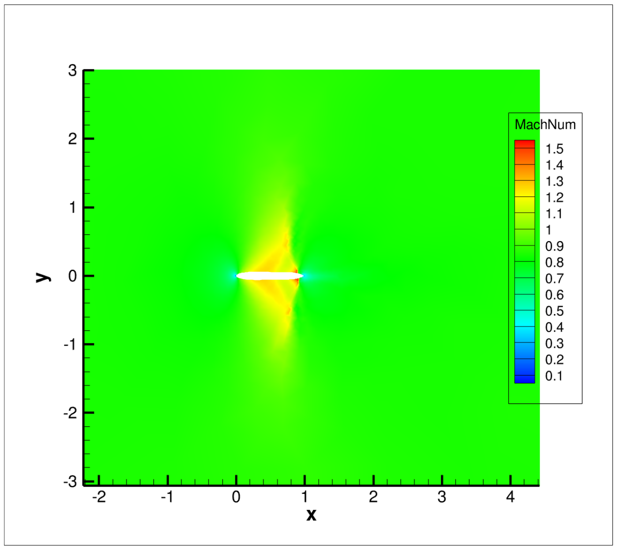
(a) Baseline Design



(b) Steady-State Optimized Design



(c) NC Optimized Design



(d) IQN Optimized Design

Figure 11. Mach number comparison

V.C. Supersonic Case

The case presented in this section is a NACA0012 airfoil shown in Figure (1) with symmetric design variables in $M = 1.25$ flow with $\alpha = 3^o$. The objective function here is a composite objective function of lift and drag.

$$L = \omega_L(C_L - C_{L_T})^2 + \omega_D(C_D - C_{D_T})^2 \tag{54}$$

The targets for lift and drag are denoted by C_{L_T}, and C_{D_T} respectively. The target lift coefficient is set to .211 which is the objective function value in this baseline simulation, the target drag coefficient is set to 0. The respective weights are set to 1.0 for all terms. This objective function will make the optimizer try to keep the lift constant and minimize the drag.
The design variables are 15 Hicks-Henne bump functions used to symmetrically perturb the top and bottom of the airfoil, where each design variable has a maximum magnitude of $1e-3$. The nonlinear convergence plot in Figure 12 shows that the convergence has stalled; for this case, the linear system in the Newton-Krylov solver is converged 5 orders of magnitude at each nonlinear iteration. Figure 13 shows the behavior of the design cycle for all three methods, showing the convergence for optimality and objective functions as a function of both the number of major iterations and the number of function evaluations. We can see very similar behavior between the all three methods, with the two approximate linearizations being nearly identical to one another and very close to the typical steady-state adjoint even though the nonlinear problem is not converging, and therefore the error in the approximate linearization has not vanished. We do see some pathological behavior in the measuring of the optimality, once again, but the objective function decrease and the design cycle appears to be identical between all the linearizations used, even if the piggy-back iterations with approximate linearizations do spend extra design iterations at the optimum before the design process is halted by the optimizer. This indicates that the effects of the incomplete linearization seem to be mostly affecting the measurement of the optimality rather than the design for better-behaved optimization problems. Furthermore, we can see the final geometries are very close in shape as well, with no noticeable differences, as shown in Figure 14. The contour plots in Figures 15 and 16 depict the density and Mach number plots for the baseline and optimized geometries, showing very similar behaviors for the optimized geometries with the shock strength decreasing due to the optimization process.

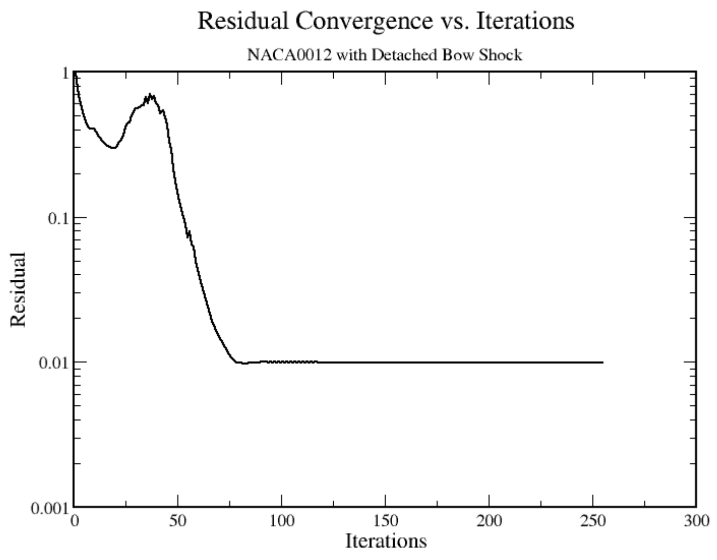
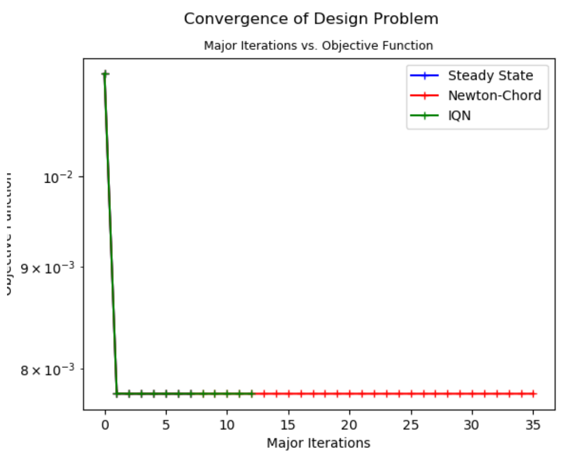


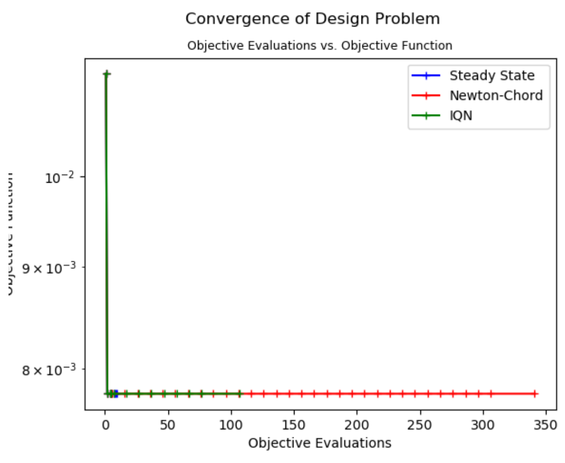
Figure 12. Analysis convergence plot

VI. Conclusion and Future Work

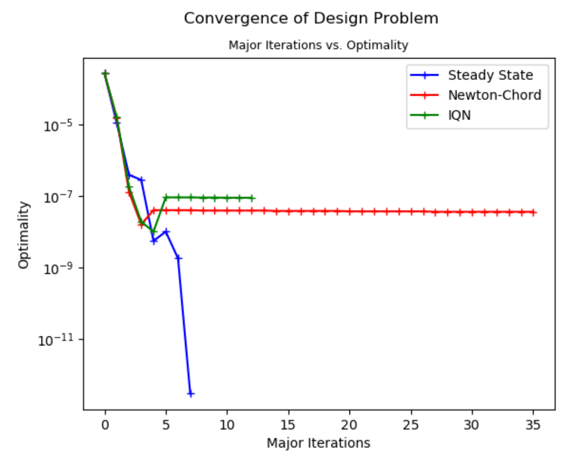
We showed the derivation for a novel approach to piggy-back iterations to solve the adjoint problem and proved that this converges to the correct result at the same rate as the convergence as the primal problem.



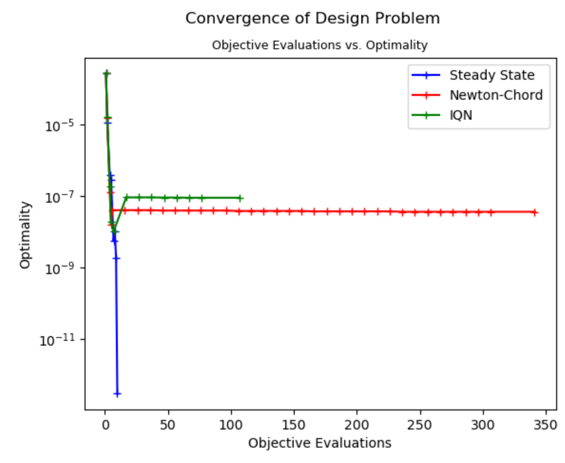
(a) Major Iterations vs. Objective



(b) Objective Evaluations vs. Objective



(c) Major Iterations vs. Optimality



(d) Objective Evaluations vs. Optimality

Figure 13. Design cycle summary

We then verified proper implementation of this method by comparing to the complex-step finite-difference computed sensitivities and the steady state adjoint computed sensitivities. Lastly we applied this method to steady state optimizations to show its utility for both convergent and nonconvergent simulations. We showed successful design cycles with both approaches even for nonconvergent simulations, but found that these applications showed issues with the optimality problem. We also found an example, where for a strongly nonconvergent simulation, the small errors due to the inexact linearizations led to drastically different designs with very similar objective functions, for this case with the large oscillations we also note that the inexact linearizations optimize the geometry in much fewer iterations than the steady-state adjoint, indicating that the issues with optimality are not as significant when compared to a faster design process for these nonconvergend simulations. The next areas of inquiry are using adaptive constraint equations with the piggy-back iterations to get cheaper design cycles with similar convergence rates, and efforts at computing the piggy-back iterations only for iterations that showed a decrease in the residual of the nonlinear problem. Additionally, work will be done to apply the inexact linearizations to the one-shot method for optimization, also known as simulation analysis and design.

VII. Acknowledgments

Computing time was provided by ARCC on the Teton supercomputer.

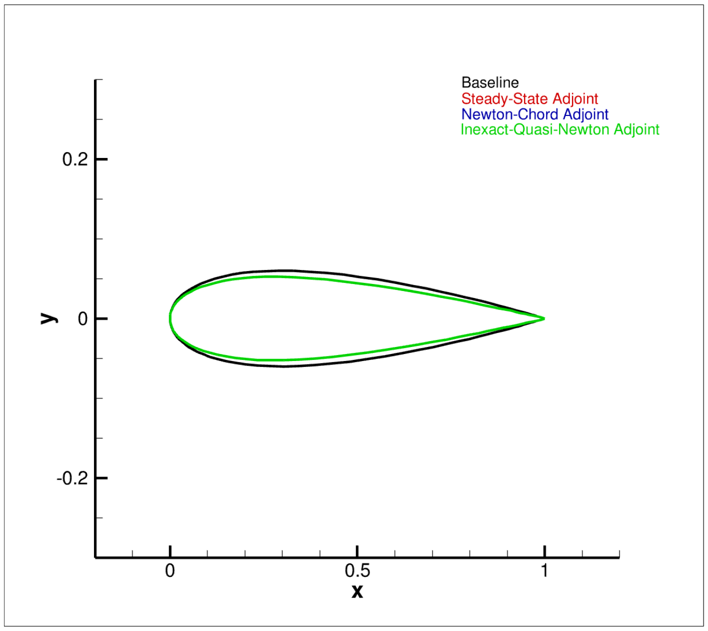


Figure 14. Geometry comparison

References

[1]Gill, P. E., Murray, W., and Saunders, M. A., "User's Guide for SNOPT Version 7: Software for Large-Scale Nonlinear Programming," .

[2]Gill, P. E., Murray, W., and Saunders, M. A., "SNOPT: An SQP algorithm for large-scale constrained optimization," *SIAM review*, Vol. 47, No. 1, 2005, pp. 99–131.

[3]Adams, B., Bauman, L., Bohnhoff, W., and et al., "Dakota, A Multilevel Parallel Object-Oriented Framework for Design Optimization, Parameter Estimation, Uncertainty Quantification, and Sensitivity Analysis: Version 6.0 User's Manual," 2015.

[4]Nadarajah, S. K., *The Discrete Adjoint Approach to Aerodynamic Shape Optimization, Ph.D. Dissertation*, Department of Aeronautics and Astronautics, Stanford University, USA, 2003.

[5]Arian, E. and Ta'asan, S., "Shape Optimization in One Shot," *Optimal Design and Control*, edited by J. Borggaard, J. Burkardt, M. Gunzburger, and J. Peterson, Birkhäuser Boston, Boston, MA, 1995, pp. 23–40.

[6]Shi-Dong, D. and Nadarajah, S., "Newton-Krylov Full-Space Aerodynamic Shape Optimization," 59th AIAA Aerospace Sciences Meeting, AIAA Paper 2021-0281, Virtual Event, January 2021. https://doi.org/10.2514/6.2021-0281.

[7]Gunther, S., *Simultaneous Optimization with Unsteady Partial Differential Equations*, Doctoral thesis, RWTH Aachen, 2017.

[8]Akbarzadeh, S., Huckelheim, J., and Muller, J.-D., "Consistent treatment of incompletely converged iterative linear solvers in reverse-mode algorithmic differentiation," Computational Optimization and Applications, 10.1007/s10589-020-00214-x.

[9]Padway, E. and Mavriplis, D. J., "Toward a Pseudo-Time Accurate Formulation of the Adjoint and Tangent Systems," 57th AIAA Aerospace Sciences Meeting, AIAA Paper 2019-0699, San Diego CA, January 2019. https://doi.org/10.2514/6.2019-0699.

[10]Padway, E. and Mavriplis, D. J., "Advances in the Pseudo-Time Accurate Formulation of the Adjoint and Tangent Systems for Sensitivity Computation and Design," 59th AIAA Aerospace Sciences Meeting, AIAA Paper 2020-3136, Virtual Event, June 2020. https://doi.org/10.2514/6.2020-3136.

[11]Sagebaum, M., *Advanced Techniques for the Semi Automatic Transition from Simulation to Design Software*, Doctoral thesis, Technische Universität Kaiserslautern, 2018.

[12]Padway, E. and Mavriplis, D., "Approximate Linearization of Fixed Point Iterations: Error Analysis of Tangent and Adjoint Problems Linearized about Non-Stationary Points," Arxiv, https://arxiv.org/abs/2104.02826.

[13]Mavriplis, D., "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.

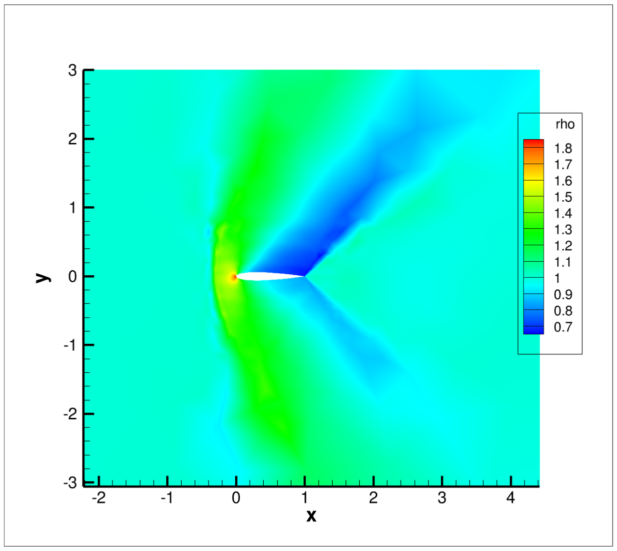
[14]LeVeque, R. J., *Numerical Methods for Conservation Laws*, Vol. 3, Springer, 1992.

[15]Roe, P., "Approximate Riemann Solvers, Parameter Vectors, and Difference Schemes," *Journal of Computational Physics*, Vol. 135, No. 2, 1997, pp. 250 – 258.

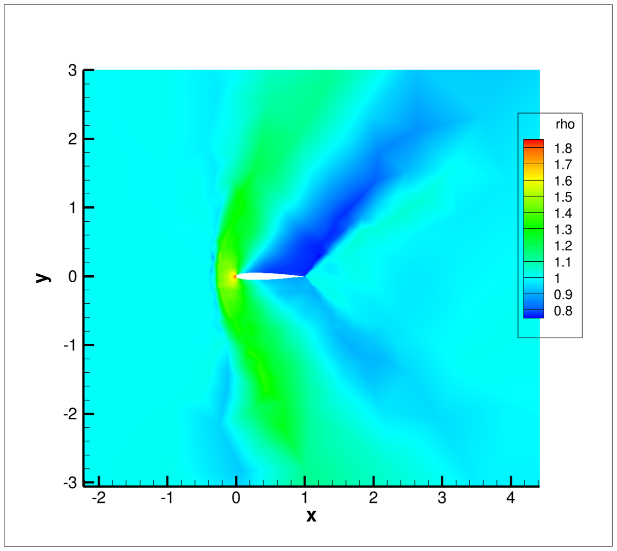
[16]van Leer, B., "Flux-vector splitting for the Euler equations," *Eighth International Conference on Numerical Methods in Fluid Dynamics*, edited by E. Krause, Springer Berlin Heidelberg, Berlin, Heidelberg, 1982, pp. 507–512.

[17]Venkatakrishnan, V., "On the accuracy of limiters and convergence to steady state solutions," 31st Aerospace Sciences Meeting, AIAA Paper 1993-880, Reno NV, January 1993. https://doi.org/10.2514/6.1993-880.

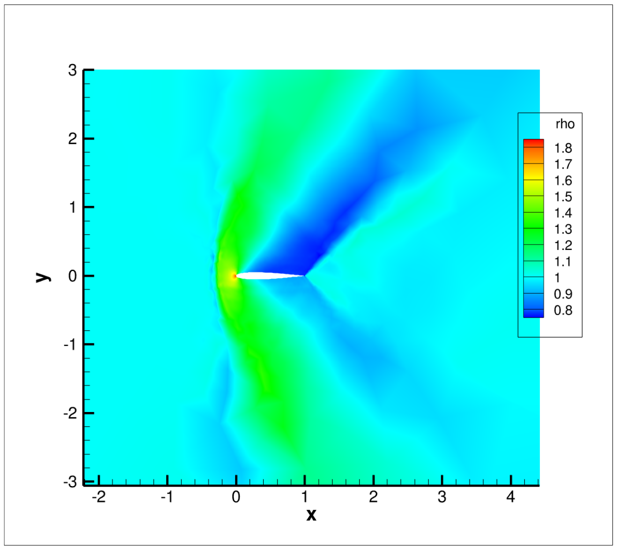
[18]Mavriplis, D. J., "A residual smoothing strategy for accelerating Newton method continuation," *Computers and Fluids*, Vol. 220, 2021, pp. 104859.



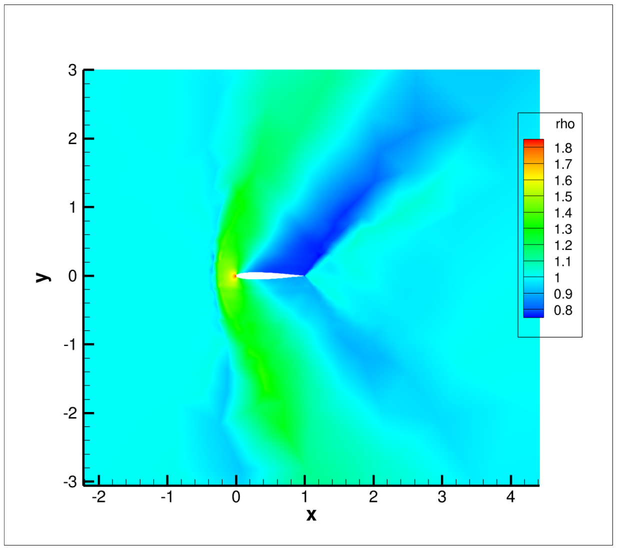
(a) Baseline Design



(b) Steady-State Optimized Design



(c) NC Optimized Design



(d) IQN Optimized Design

Figure 15. Density comparison

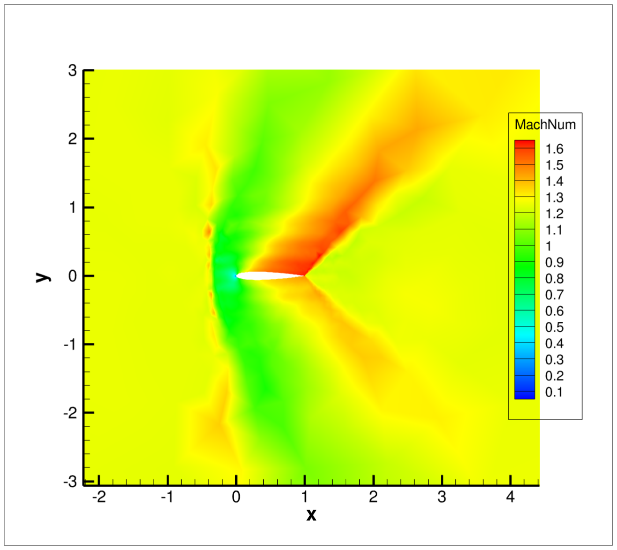
[19]Ahrabi, B. R. and Mavriplis, D. J., "An implicit block ILU smoother for preconditioning of Newton–Krylov solvers with application in high-order stabilized finite-element methods," *Computer Methods in Applied Mechanics and Engineering*, Vol. 358, 2020, pp. 112637.

[20]Saad, Y., *Iterative methods for sparse linear systems*, Vol. 82, Society for Industrial and Applied Mathematics, 2003.

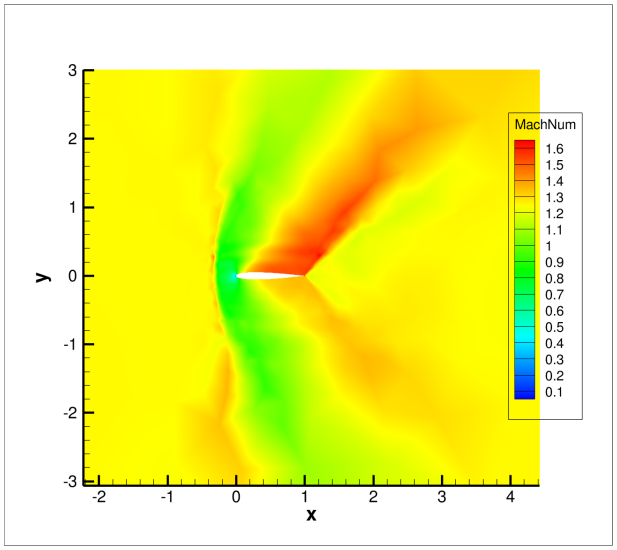
[21]Mavriplis, D. J., "VKI Lecture Series: 38th Advanced Computational Fluid Dynamics. Adjoint methods and their application in CFD, Time Dependent Adjoint Methods for Single and Multi-disciplinary Problems," *VKI Lecture Notes, von Karman Institute for Fluid Dynamics, Rhode St-Genese, Belgium*, Sep 2015.

[22]Luke, E., Collins, E., and Blades, E., "A fast mesh deformation method using explicit interpolation," *J. Comput. Physics*, Vol. 231, 01 2012, pp. 586–601.

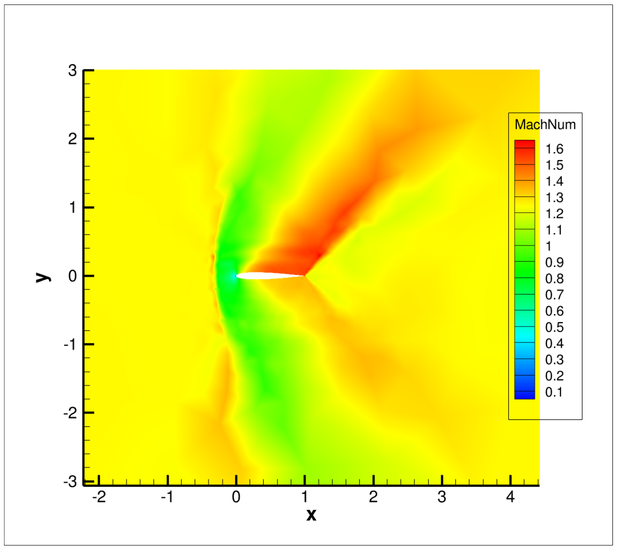
[23]Destarac, D., Carrier, G., Anderson, G. R., Nadarajah, S., Poole, D. J., Vassberg, J. C., and Zingg, D. W., "Example of a Pitfall in Aerodynamic Shape Optimization," *AIAA Journal*, Vol. 56, No. 4, 2018, pp. 1532–1540.



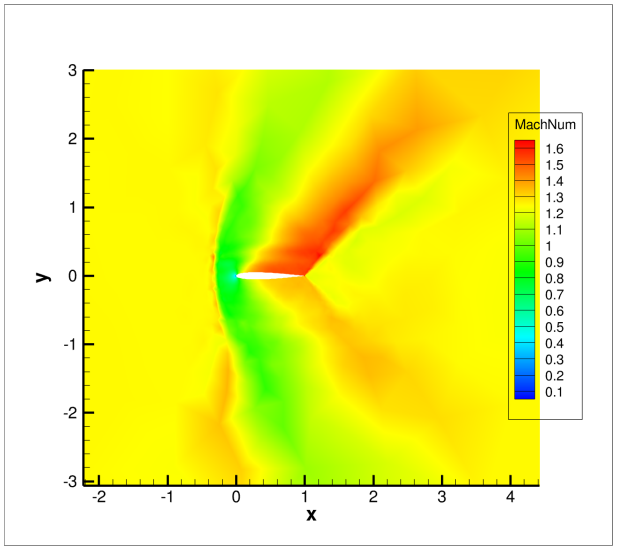
(a) Baseline Design



(b) Steady-State Optimized Design



(c) NC Optimized Design



(d) IQN Optimized Design

Figure 16. Mach number comparison