Eleventh International Conference on Computational Fluid Dynamics (ICCFD11), Maui, HI, USA, July 11-15, 2022

ICCFD11-2022-1701

Progress in the Usage of Inexact Linearizations in Piggy-back Iterations for Adjoint Computation

E. Padway
Corresponding author: emmett.padway@nianet.org

National Institute of Aerospace, Hampton, VA, USA.

July 12th, 2022

Abstract: This paper contains improvements in computing steady-state adjoint-based sensitivities using inexact linearizations of fixed-point iterations used to solve the nonlinear problem. This method guarantees convergence of the adjoint solution, provided that the solution of the nonlinear primal problem is fully converged. The method is enhanced by introducing an adjoint iteration tolerance and a nonlinear constraint tolerance to allow for cheaper and more robust sensitivity computation as well as cheaper design algorithms. Investigations in the adjoint iteration tolerance show it is possible to get better convergence and less expensive adjoint solvers by selectively skipping iterations with the "piggy-back" approach and that these can be used effectively as parts of design cycles especially when coupled with the nonlinear constraint controller which further decreases design cost.

Keywords: Optimization, Adjoint Methods, Fixed-point Iterations.

1 Introduction

In aerodynamic shape optimization, gradient based approaches are used to solve the minimization problem:

$$\min_D L(u(D), D), \text{ s.t. } R(u(D), D) = 0, \tag{1}$$

where L is the objective function (such as lift or drag), R is the residual operator (the error in the discretized form of the governing equations), u is the conservative variable vector, and D is the design variable vector. Gradient based methods are prefered because 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 classic approach is that of partial differential equation (PDE) constrained optimization –also known as the nested approach – in which the PDE is solved at each iteration and the sensitivities about the converged state are used to change the design variables to generate a new design. The sensitivity equation is

$$\frac{dL}{dD} = \frac{\partial L}{\partial D} + \frac{\partial L}{\partial u}\frac{du}{dD}, \tag{2}$$

where an expression for $\frac{du}{dD}$ is required for the sensitivity to be computed. Since the nested approach proceeds off the assumption that $R = 0$:

$$\left[\frac{\partial R}{\partial u}\right] \frac{du}{dD} + \frac{\partial R}{\partial D} = 0. \tag{3}$$

Substituting equation for $\frac{du}{dD}$ into the sensitivity equation yields

$$\frac{dL}{dD} = \frac{\partial L}{\partial D} - \frac{\partial L}{\partial u}\left[\frac{\partial R}{\partial u}\right]^{-1} \frac{\partial R}{\partial D}, \tag{4}$$

One can then define an adjoint variable $\mathbf{\Lambda}$ according to the equation below

$$\left[\frac{\partial R}{\partial u}\right]^{T} \mathbf{\Lambda} = -\left[\frac{\partial L}{\partial u}\right]^{T}, \tag{5}$$

which scales with the number of objective functions rather than the number of design variables.
The other approach to the optimization problem is often referred to as the one-shot approach [5], in which the PDE, the adjoint system and the design problem are solved in tandem:

$$\begin{aligned} u^{k+1} &= N(u^k, D), \\ \Lambda^{k+1} &= B(u^k, \Lambda^k, D), \\ D^{k+1} &= D^k + F(u^k, \Lambda^k, D^k). \end{aligned} \tag{6}$$

Where $N(u^k, D)$ is a fixed-point iteration meant to drive $R(u^k, D)$ to 0, B is an operator which in the context of "piggy-back" iterations is defined to be $L_u + N_u\Lambda^k$, and F is a preconditioner to guarantee convergence for the coupled iterations. The one-shot adjoint method was pioneered by Ta'asan [6], and used to develop a unified multigrid solver to the nonlinear, adjoint and design problems all at once, greatly decreasing the cost of the design process 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. [7] 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, as well as preconditioners that work on all three problems. Gunther et al. [8] 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 method 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. As a result, much of the previous work on these piggy-back iterations used explicit iterations of the form

$$N(u^k, D) = u^k + \frac{\Delta t}{vol} R(u^k), \tag{7}$$

where Δt and vol were the local time step and volume respectively. The differentiation of such a fixed-point iteration is straight-forward and work on implicit iterations of the form

$$N(u^k, D) = u^k - [P_k]^{-1} R(u^k), \tag{8}$$

where P_k is an approximate residual Jacobian is less common due to the need to differentiate the approximate inversion of the P_k matrix.
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 the best way to differentiate the fixed-point iterations that include a linear solve [9], in the meantime this is a barrier to applications of the

piggy-back iterations to more robust and scalable fixed-point iterations; Padway and Mavriplis showed a possible approach to solve this problem [10].
This work is an outgrowth of the work of Padway and Mavriplis [11, 12] on the pseudo-time accurate adjoint, also referred to as the black-box adjoint [13]. 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 [12] 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 [14] 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. Subsequent work of Padway and Mavriplis [10] proved similar behavior for the piggy-back iterations, they showed desirable convergence behaviors in both mathematical proof and numerical experiment, in that the error due to such approximations decreases at the same rate as the solution of the PDE itself. They used these inexact linearizations in design problems for non-linear problems with stalling convergence. This work will show the application of the piggy-back iterations to convergent iterations only in order to decrease the computational expense of such techniques and more throughly evaluate their behavior with non-linear controllers on the adjoint iterations and the design process itself.

2 Background and In-House Solver

2.1 Governing Equations

This work uses 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.

$$\nabla \cdot F(u(D)) = 0 \tag{9}$$

Which can also be written as:

$$R(u(D), D) = 0 \tag{10}$$

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:

$$R = \int_{dB} [F(u(D))] \cdot n(x(D)), \mathrm{dB} = \sum_{i=1}^{n_{edge}} F_{e_i}^{\perp}(u(D), n_{e_i}(x(D)))B_{e_i}(x(D)) \tag{11}$$

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 [15]. The work presents results in which Van Leer's inviscid flux vector splitting as the euler flux. In order to slope limit the solution reconstruction near shock discontinuties, a modified Venkatakrishnan limiter is used; Venkatakrishnan's limiter[16] is modified in a few important ways as detailed in previous works[12].

2.3 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:

$$u^k = u^{k-1} + \Delta u \tag{12}$$

where Δu is computed by solving the following system of linear equations.

$$[P]\,\Delta u = -R(u) \tag{13}$$

One can substitute the expression for Δu into the time-stepping equation (12) to obtain the final form of this equation.

$$u^{k+1} = u^k - [P_k]^{-1} R \tag{14}$$

Here $[P_k]$ is a first-order spatially accurate Jacobian augmented with a diagonal term to ensure that it is diagonally dominant, shown in equation (15).

$$[P_k] = \left[\frac{\partial R}{\partial u^k}\right]_1 + \frac{vol}{\Delta t CFL} \tag{15}$$

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:

$$\Delta t_i = \frac{r_i}{\sqrt{(u^2 + v^2)} + c} \tag{16}$$

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:

$$CFL = min(\beta \cdot CFL, CFL_{max}) \tag{17}$$

or with a linesearch and CFL controller [17, 18], which seeks to minimize the L_2 norm of the pseudo-temporal residual, defined as below.

$$R_t(u + \alpha\Delta u) = \frac{vol}{\Delta t}\alpha\Delta u^n + R(u + \alpha\Delta u^n) \tag{18}$$

When the pseudo-temporal residual decreases, this is considered to be a satisfactory value for α and the CFL is changed 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 there is no need to differentiate it, and one 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

procedure LINE SEARCH AND CFL CONTROLLER
  $r_{t0} = \|R_t(u + \Delta u)\|_2$
  $r_{s0} = \|R(u)\|_2$
  $r_{s1} = \|R(u + \Delta u)\|_2$
  if $r_{s0} < r_{s1}$ **then**
    for $k = 1, ..., iter_{max}$ **do**
      $\alpha = c\alpha$
      $r_{t1} = \|R_t(u + \alpha\Delta u)\|_2$
      if $r_{t1} < r_{t0}$) **then** exit
  if $\alpha < \alpha_{l_1}$ **then**
    $\alpha = 0$
    $CFL = \beta_1 CFL$
  else if $\alpha_{l_1} < \alpha < \alpha_{l_2}$ **then**
    $CFL = CFL$
  else if $\alpha < \alpha_{l_2}$ **then**
    $CFL = min(\beta_2 CFL, CFL_{max})$

This line-search controller is augmented with a relizability check. This realizability check keeps the change in density, energy and pressure below a user defined maximum change depending on the freestream values and above a minimum threshold. Since pressure is a nonlinear function of the conservative variables, this is done using a linearization of the pressure with respect to the conservative variables, combined with a fixed-iteration back-tracking line search. The minimum update between the three thresholds is then applied to the cell as a whole and stored.

In order to solve the linear system a point-implicit Jacobi or point-implicit Gauss-Seidel solver is used. 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 (15). Equation (13) is then solved iteratively as:

$$[D]\,\Delta(\Delta u)^l = -R(u) - [P_{k-1}]\,\Delta u^l \tag{19}$$

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

$$\Delta u^{l+1} = \Delta u^l + \omega(\Delta(\Delta u))^l \tag{20}$$

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)[19] is 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, 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_s is the geometry coordinate vector and x_v is the volume mesh point coordinate vector. In order to obtain an expression for the sensitivities take the derivative of the objective functional [20]:

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

Algorithm 2 Flexible Restarted GMRES

procedure FLEXIBLE GMRES
  for $k = 1, ..., ncycles$ **do**
    $r_0 = b - Ax_0, \beta = \|r_0\|, v_1 = r_0/\beta$
    for $j = 1, ..., m$ **do**
      $z_j = M^{-1} v_j$
      $v_{j+1} = Az_j$
      for $i = 1, ..., j$ **do**
        $h_{i,j} = (v_{j+1}, v_i)$
        $v_{j+1} = v_{j+1} - h_{i,j} v_i$
      $h_{j+1,j} = \|v_{j+1}\|, v_{j+1} = v_{j+1}/h_{j+1,j}$
      Define $Z_m = [z_1, ..., z_m], \bar{H}_m = [h_{i,j}]_{1<i<j+1, 1<j<m}$
    Solve least squares problem for y_m
    $x_0 = x_0 + Z_m y_m$

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:

$$[K] \frac{dx_v}{dD_i} = \frac{dx_s}{dD_i} \tag{22}$$

and one can calculate $\frac{dx_s}{dD_j}$ through differentiating the shape design variables. For the global inverse distance weighted method[21] surface changes in the geometry can be smoothly and explicitly propogated onto the volume mesh, and for these the mesh sensitivities are computed directly as a function of the surface coordinate sensitivities:

$$\frac{dx_{v_i}}{dD_j} = \frac{\sum w_{ik}(\vec{r}_{ik}) \frac{dx_{s_k}}{dD_j}}{\sum w_{ik}(\vec{r}_{ik})} \tag{23}$$

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 one uses the constraint that for a fully converged flow $R(u(D), x(D)) = 0$. By taking the derivative of the residual operator one can obtain the equation below.

$$\left[\frac{\partial R}{\partial x}\right] \frac{dx}{dD} + \left[\frac{\partial R}{\partial u}\right]_2 \frac{du}{dD} = 0 \tag{24}$$

The tangent system is obtained by isolating the sensitivity of the residual to the design variables.

$$\left[\frac{\partial R}{\partial u}\right]_2 \frac{du}{dD} = -\left[\frac{\partial R}{\partial x}\right] \frac{dx}{dD} \tag{25}$$

This linear system is then solved, 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 $\frac{du}{dD}$. It is then possible to substitute $\frac{du}{dD}$ into equation (21) to obtain the final sensitivities.

2.4.2 Discrete Adjoint Formulation

The adjoint formulation begins with the same sensitivity equation:

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

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

$$\frac{du}{dD} = -\left[\frac{\partial R}{\partial u}\right]_2^{-1}\left[\frac{\partial R}{\partial x}\right]\frac{dx}{dD} \tag{27}$$

Substituting the above expression into the sensitivity equation yields:

$$\frac{dL}{dD} = \frac{\partial L}{\partial x}\frac{dx}{dD} - \frac{\partial L}{\partial u}\left[\frac{\partial R}{\partial u}\right]_2^{-1}\left[\frac{\partial R}{\partial x}\right]\frac{dx}{dD} \tag{28}$$

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

$$\mathbf{\Lambda}^T = -\frac{\partial L}{\partial u}\left[\frac{\partial R}{\partial u}\right]_2^{-1} \tag{29}$$

which gives an equation for the adjoint variable:

$$\left[\frac{\partial R}{\partial u}\right]_2^T \mathbf{\Lambda} = -\left[\frac{\partial L}{\partial u}\right]^T \tag{30}$$

This linear system can be solved to obtain the sensitivities for the objective function as follows:

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

For further use of adjoint one can then define a mesh adjoint variable $\mathbf{\Lambda_x}$:

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

and the final expression for sensitivity is given as:

$$\frac{dL}{dD} = \mathbf{\Lambda_x}^T\frac{dx_s}{dD} \tag{33}$$

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

3 Development of the Inexactly Linearized Piggy-Back Iterations

This section shows the piggy-back iterations that are similar to those used 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. This begins from the understanding as in the steady state adjoint that the goal is for 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:

$$u^{k+1}(D) = N(u^k(D), D) = u^k(D) + H(u^k(D), D) \tag{34}$$

where:

$$H(u^k(D), D) = A(u^k, D)R(u^k(D), D) \tag{35}$$

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 $G(u^*(D), D) - u^*(D) = 0$, allows for a similar proof to that of the steady state adjoint shown previously. One must begin from the objective function (L) augmented by a Lagrangian

vector (the adjoint variable vector) multiplying the constraint as shown below.

$$J(u^*(D), D) = L(u^*(D), D) + \Lambda^T (N(u^*(D), D) - u^*) \tag{36}$$

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

$$\begin{array}{ll} N(u^*(D), D) - u^* = 0 & \textit{State equation} \\ \frac{\partial L(u^*(D), D)}{\partial u} + \Lambda^T \frac{\partial (N(u^*(D), D) - u^*)}{\partial u} = 0 & \textit{Adjoint equation} \\ \frac{\partial L(u^*(D), D)}{\partial D} + \Lambda^T \frac{\partial (N(u^*(D), D))}{\partial D} = 0 & \textit{Design equation} \end{array} \tag{37}$$

The adjoint and sensitivity equations can be evolved through iteration-space as shown below

$$\begin{aligned} \Lambda^{k+1T} &= \frac{\partial L(u(D), D)}{\partial u} + \Lambda^{kT} \left(\frac{\partial N(u^k(D), D)}{\partial u} \right) \\ \frac{dL(u(D), D)}{dD} &= \frac{\partial L(u(D), D)}{\partial D} + \Lambda^{kT} \left(\frac{\partial N(u^k(D), D)}{\partial D} \right) \end{aligned} \tag{38}$$

and use the definition of the fixed-point iteration allows for the expansion of the above expressions.

$$\begin{aligned} \Lambda^{k+1T} &= \frac{\partial L(u(D), D)}{\partial u} + \Lambda^{kT} \left(\frac{\partial u^k + A(u^k(D), D) R(u^k(D), D)}{\partial u} \right) \\ \frac{dL(u(D), D)}{dD} &= \frac{\partial L(u(D), D)}{\partial D} + \Lambda^{kT} \left(\frac{\partial A(u^k(D), D) R(u^k(D), D)}{\partial D} \right) \end{aligned} \tag{39}$$

This returns the equations below:

$$\begin{aligned} \Lambda^{k+1T} &= \frac{\partial L(u(D), D)}{\partial u} + \Lambda^{kT} \left(\frac{\partial A(u^k(D), D) R(u^k(D), D)}{\partial u} \right) + \Lambda^{kT} \\ \frac{dL(u(D), D)}{dD} &= \frac{\partial L(u(D), D)}{\partial D} + \Lambda^{kT} \left(\frac{\partial A(u^k(D), D) R(u^k(D), D)}{\partial D} \right) \end{aligned} \tag{40}$$

and after expanding the derivatives, and dropping the dependencies in the notation returns the below equation.

$$\begin{aligned} \Lambda^{k+1T} &= \frac{\partial L}{\partial u} + \Lambda^{kT} \left(\frac{\partial A}{\partial u} R(u^k, D) + A(u^k, D) \frac{\partial R}{\partial u} \right) + \Lambda^{kT} \\ \frac{dL}{dD} &= \frac{\partial L}{\partial D} + \Lambda^{kT} \left(\frac{\partial A}{\partial D} R(u^k(D), D) + A(u^k, D) \frac{\partial R}{\partial D} \right) \end{aligned} \tag{41}$$

Previous work [10] showed that for correct linearization of the residual operator the error converges with the non-linear problem's convergence. This showed 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. By this definition one can then obtain this expression for the error where through grouping the terms in the parenthesis into the operator B where $B = \frac{\partial N}{\partial u}$ and by Cauchy-Schwarz inequality it is clear that:

$$\left\| \epsilon_\Lambda^{kT} \left(\frac{\partial A}{\partial u} R(u^k, D) + A(u^k, D) \frac{\partial R}{\partial u} + I \right) \right\| < \|B\| \left\| \epsilon_\Lambda^k \right\| \tag{42}$$

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.

$$\left\| \epsilon_\Lambda^{k+1} \right\| < \|B\| \left\| \epsilon_\Lambda^k \right\| + \left\| \tilde{\Lambda}^{kT} \epsilon_u^k \right\| \|R\| \tag{43}$$

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. The equation for the error in the sensitivities reproduced below shows 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 iteration, the second by the proof outlined above.

$$\frac{dL}{dD} - \widetilde{\frac{dL}{dD}} = \tilde{\Lambda}^{kT}\epsilon_D^k R(u^k, D) + \epsilon_\Lambda^{kT}\left(\frac{\partial A}{\partial D}R(u^k, D) + A(u^k, D)\frac{\partial R}{\partial D}\right) \tag{44}$$

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 [14], and then confirmed to be correct for the piggy-back approach [10]. Importantly for this work, the rate of convergence of the error mimics the rate of convergence of the non-linear problem, something that will be used in this work.

3.1 Newton-Chord Approximate Linearization

This methods begins by referring back to equation (14) as the fixed-point iteration that must be linearized as part of the piggy-back iterations. If it is rewritten it as follows:

$$u^{k+1} = N(u) = u^k - [P_k]^{-1} R \tag{45}$$

it can then be substituted as the fixed-point iteration definition into the piggyback iterations.

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

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

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

by expanding the derivatives one can obtain better insight to the appropriate way to calculate this linearization.

$$\begin{aligned}\Lambda^{k+1} &= \left(I - [P_k]^{-1}\left[\frac{\partial R(u^k)}{\partial u^k}\right]_2 - \frac{\partial [P_k]^{-1}}{\partial u^k}R(u^k)\right)^T \Lambda^k + \frac{\partial L}{\partial u}^T\\ \frac{dL}{dD} &= \frac{\partial L}{\partial D} - \Lambda^{kT}\left([P_k]^{-1}\frac{\partial R(u^k)}{\partial D} + \frac{\partial [P_k]^{-1}}{\partial D}R(u^k)\right)\end{aligned} \tag{48}$$

It is then clear that this equation requires the derivative of the preconditioner matrix, which is not trivial to compute. This section proceeds 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. This will be referred to as the Newton-Chord linearization (NC). Note this corresponds to the backward-Euler Lagrange-based

formulation found in the literature [13].

$$\begin{aligned}\Lambda^{k+1} &= \left(I - [P_k]^{-1}\left[\frac{\partial R(u^k)}{\partial u^k}\right]_2\right)^T \Lambda^k + \frac{\partial L}{\partial u}^T \\ \frac{dL}{dD} &= \frac{\partial L}{\partial D} - \Lambda^{kT}\left([P_k]^{-1}\frac{\partial R(u^k)}{\partial D}\right)\end{aligned} \tag{49}$$

It appears that one would have to solve for as many linear systems as design variables at each nonlinear iteration, which is clearly undesirable, but if the equations are rewritten in a delta formulation and define a secondary adjoint variable the following is obtained.

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

As a result of this rearrangement of the terms it is clear that there are 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.

3.2 Inexact-quasi-Newton Approximate Linearization

This linearization begins from equation (48) and is an attempt to linearize the approximate inversion of the left hand matrix. One can refer to the expression for the analytic derivative of a matrix inverse shown below for a matrix K.

$$\frac{dK^{-1}}{dx} = -K^{-1}\frac{dK}{dx}K^{-1} \tag{51}$$

Substituting this into equation (48) returns the below expression:

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

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

$$\begin{aligned}\Lambda^{k+1} &= \left(I - [P_k]^{-1}\left[\frac{\partial R(u^k)}{\partial u^k}\right]_2 + [P_k]^{-1}\frac{d[P_k]}{du^k}\Delta u\right)^T \Lambda^k + \frac{\partial L}{\partial u}^T \\ \frac{dL}{dD} &= \frac{\partial L}{\partial D} - \Lambda^{kT}\left([P_k]^{-1}\frac{\partial R(u^k)}{\partial D} + [P_k]^{-1}\frac{d[P_k]}{dD}\Delta u\right)\end{aligned} \tag{53}$$

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

$$
\begin{aligned}
\left[P_k\right]^T \Psi^k &= -\Lambda^k \\
\Delta\Lambda^k &= \left(\left[\frac{\partial R(u^k)}{\partial u^k}\right]_2 - \frac{d\left[P_k\right]}{du^k}\Delta u\right)^T \Psi^k + \frac{\partial L}{\partial u}^T \\
\frac{dL}{dD} &= \frac{\partial L}{\partial D} + \Psi^{kT}\left(\frac{\partial R(u^k)}{\partial D} - \frac{d\left[P_k\right]}{dD}\Delta u\right)
\end{aligned}
\tag{54}
$$

As a result of this rearranging of the terms it is clear – once again – that there are 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.

3.3 Adjoint Iteration and Nonlinear Constraint Controllers

This section contains two controller schemes, the first an adjoint iteration controller to control whether the adjoint update is calculated, and the second a nonlinear constraint controller to select the tolerance of the PDE constraint. A simple implementation of the adjoint iteration controller would entail using the convergence of the adjoint and the nonlinear problems. Since the expressions for the adjoint and sensitivity show convergence behavior similar to that of the fixed point iteration of the nonlinear problem, it is useful look to the convergence of the fixed-point iteration as being a quantity by which to guide the adjoint solver. Specificially one can define a skipping threshold (η_Λ) and tolerance (τ_{adj}) and say that iterations with a contractivity (c) less than η and a nonlinear residual greater than the τ_{adj} will not be used to compute the adjoint. The motivation in developing and using an adaptive η_Λ based off the nonlinear residual is that due to the approximations made in the derivation of the approximate linearizations of the piggy-back iterations, contractivity of the nonlinear residual is a less accurate guide to the contractivity of the adjoint iteration early in the solution process when the residual is large. As such it is desireable to use only the highly contractive iterations (low values of c) to hedge against the larger error in the linearization in the less converged iterations. Note that in the rest of this paper, the fixed skipping threshold method is denoted by $\eta_\Lambda = a$ (where $a = \eta_{base}$) and the adaptive method is $\eta_\Lambda = f(\|R(u)\|)$.

Algorithm 3 Adjoint Iteration Controller

procedure ADJOINT ITERATION CONTROLLERS
  user inputs $\eta_{max}, \eta_{base}, \tau_{adj}, \tau_u$
  $adj_{eval} = .false.$
  $\rho_{i-1} = R(u^{i-1})$
  $\rho_i = R(u^i)$
  $c = \frac{\rho_i}{\rho_{i-1}}$
  if $opt = 1$ **then**
    $\eta_\Lambda = \eta_{base}$
  else if $opt = 2$ **then**
    $\eta_\Lambda = \eta_{base} + (\eta_{max} - \eta_{base})(log(\frac{\rho_i}{\tau_u}) - 1.0)$
  if $(c < \eta_\Lambda$ .or. $\rho_i < \tau_{adj})$ **then** $adj_{eval} = .true.$

In order to have a useful adjoint computation method one needs to obtain reasonable results when the linearization of the fixed point iteration has not fully converged the adjoint problem. Previous work [10] has dealt with this to a limited extent, here it is used in the context of inexactly constrained adjoint algorithms [22]. The methods in Algorithm 4 set the tolerance of the nonlinear problem depending on the gradient at a design point, and the change in the gradient and the converged solution between subsequent updates. Note that this has the inherent effect of also adaptively tuning the adjoint as in previous work by Brown and Nadarajah [23], and the the adjoint problem will converge similarly to the primal – with some lag [24]. Similarly to the previous section where two options of selecting the skipping threshold were presented, here too are two methods of selecting the nonlinear tolerance presented.

Algorithm 4 Nonlinear Constraint Controller

procedure SELECTION OF NONLINEAR TOLERANCE
    user inputs η, p, τ_{max}
    $S_0 = 0.0$
    if $opt = 1$ **then**
        $\tau_u = min(\tau_{max}, \eta \|G\|^p))$
    else if $opt = 2$ **then**
        $S_k = max(S_{k-1}, \frac{\Delta G}{\Delta w})$
        $\tau_u = \eta \|G\|^p / S_k$

Note that in the work by Brown and Nadarajah, the steepest descent method is used to ensure that the only thing impacting convergence of the optimization problem was the gradient at that design iteration, whereas in this work SNopt's quasi-Newton solver was used, which uses gradient values at previous iterations to calculate an approximate Hessian. The outcome of this is that some of the heuristics they came up with regarding the tuning of certain parameters (η, p, τ_{max}) may not be optimal for the design cases presented in the next section. Additionally it is possible that there are better algorithms for nonlinear constraint control for quasi-Newton algorithms than the ones that were obtained by applying the steepest descent controllers to the quasi-Newton optimizer.

4 Verification and Investigation

The case of a NACA0012 airfoil (shown in Figure 1) in $Mach = .7$ and $\alpha = 2.0^o$ compressible inviscid flow is used for verification and the investigation of the behavior of the nonlinear iterative controllers. The sensitivities, expense (measured by time taken to solve the adjoint problem), and the number of nonlinear iterations (out of 300) skipped for the adjoint solution methods are compared in Table 1. It is clear that the choice to skip certain iterations is not harmful to the accuracy of the final sensitivities and decreases the cost with both inexact linearizations becoming more economical than the steady-state adjoint solution method. The value of .96 was chosen as that was approximately the average contractivity of the iterative process.

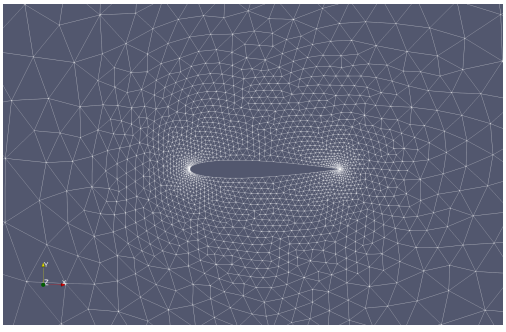
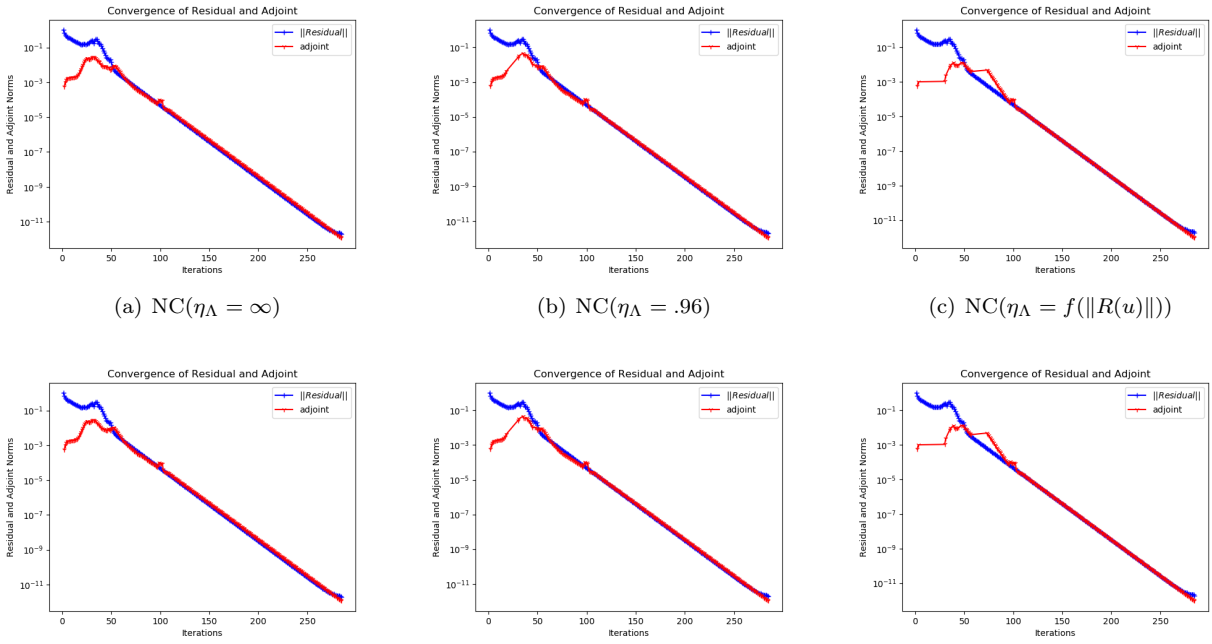


Figure 1: Computational mesh for NACA0012 airfoil

By examining the convergence behavior in the adjoint and comparing it to the convergence of the primal, it is possible to confirm the theoretical convergence rates and see how the selective iterations function depending on the choice of threshold and tolerance. Figure 4 shows the expected behavior, in that iterations that do not converge the primal problems are skipped. Additionally the adjoint converges similarly to the primal problem, finishing with a similar level of convergence in the adjoint problem as that of the normalized residual from the primal problem.
It is enlightening to compare the adjoint convergence plots –grouped by approximate linearization – to one another as in Figure 3. It is clear that the adaptive η iterations which skip the most iterations also converge the adjoint the best (by a small margin), this can be attributed to neglected the nonconvergent iterations which waste compute time and often increase the adjoint residual.

	Design Variable 1	Design Variable 2	n_{skip}
Complex	-0.4507974688123841	0.4465241035044418	N/A
NC($\eta_\Lambda = \infty$)	-0.450797468**9173912**	0.446524103**5369428**	0
NC($\eta_\Lambda = .96$)	-0.450797468**9173086**	0.446524103**5368919**	17
NC($\eta_\Lambda = f(\|R(u)\|)$)	-0.450797468**9172020**	0.446524103**5368239**	49
IQN($\eta_\Lambda = \infty$)	-0.450797468**9177968**	0.44652410357**0890**	0
IQN($\eta_\Lambda = .96$)	-0.450797468**9176973**	0.446524103**5370540**	17
IQN($\eta_\Lambda = f(\|R(u)\|)$)	-0.450797468**9176271**	0.446524103**5369982**	49
Steady State	-0.450797468**9156679**	0.446524103**5362271**	N/A

Table 1: Comparison of sensitivities computed by various methods .



(a) NC($\eta_\Lambda = \infty$) (b) NC($\eta_\Lambda = .96$) (c) NC($\eta_\Lambda = f(\|R(u)\|)$)

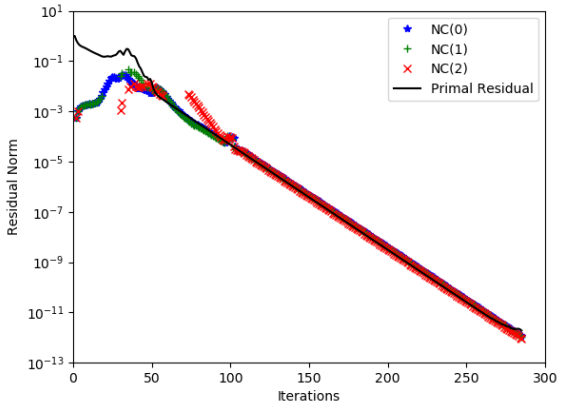
(d) IQN($\eta_\Lambda = \infty$) (e) IQN($\eta_\Lambda = .96$) (f) IQN($\eta_\Lambda = f(\|R(u)\|)$)

Figure 2: Adjoint computation convergence for given η_Λ and τ_Λ

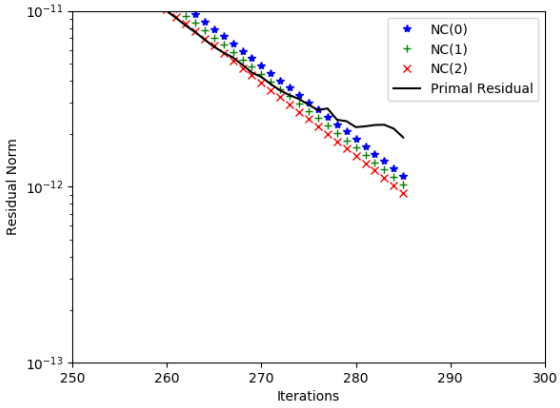
Having seen the desired behavior with the reasonable parameters in Figure , it is then useful to evaluate the behavior over a range of η_Λ and τ_Λ. Figure shows the accuracy as a function of these two parameters. It shows that when using the constant η algorithm a large portion of the domain shows low numbers of iterations skipped, this happens when the threshold is below the average contractivity of the solution process and leads to a more expensive method than the adaptive η method, however when one skips too many iterations the accuracy of the gradients. Wise choices in η_Λ and τ_Λ lead to a moderate number of iterations skipped without any degradation in accuracy. This region can be found by looking at 4 in conjuction with 5.
Additionally the totality of the results shows that the choice of inexact linearization does not matter much to the adjoint convergence for these piggy-back iterations, which is in accordance with both intuition and previous work [10].

5 Design cycle results

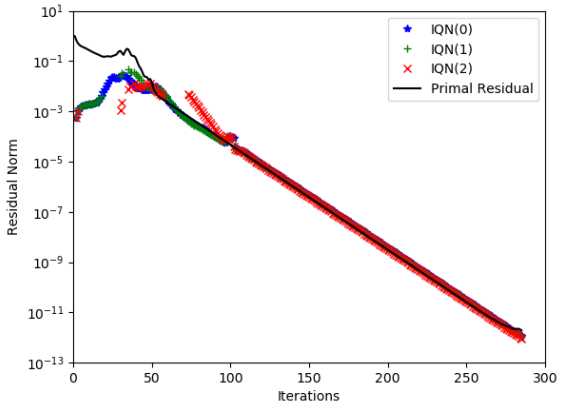
The case presented in this section is a NACA0012 airfoil shown in Figure (1) with asymmetric design variables in $Mach = .8$ and $\alpha = 1.25^o$ flow. The baseline geometry shows good convergence behavior so this seemed to be a suitable choice of test case. The objective function here is a composite objective function of lift and



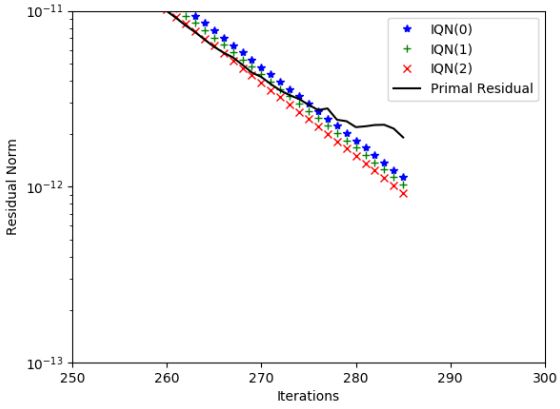
(a) NC full history



(b) NC zoomed



(c) IQN full history



(d) IQN zoomed

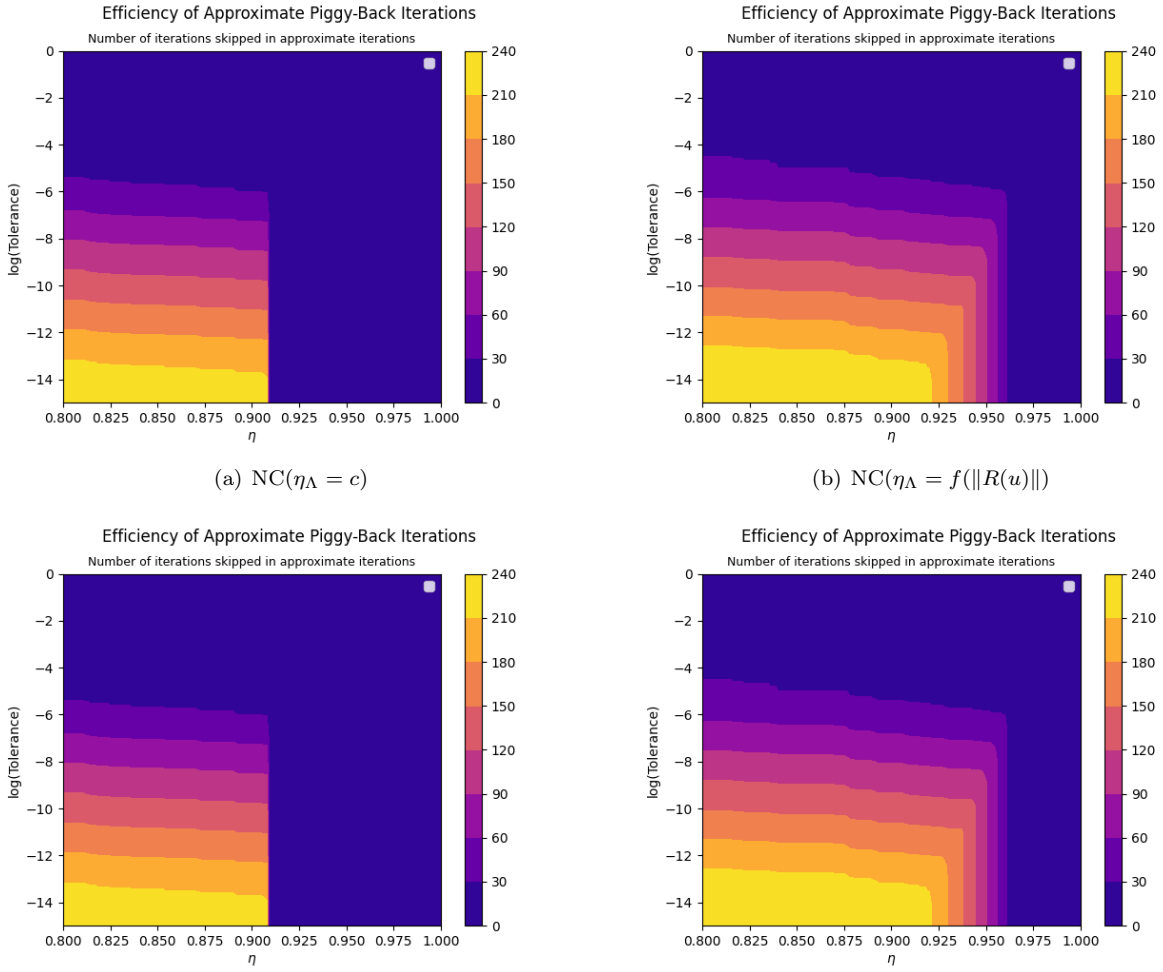
Figure 3: Adjoint computation comparison for given η_Λ and τ_Λ

drag. Where we select p and η_{opt} as 1.2 and $1e-2$ respectively. We choose tau_{max} to guarantee convergence of 3 orders in the primal problem.

$$L = \omega_L(C_L - C_{L_T})^2 + \omega_D(C_D - C_{D_T})^2 \tag{55}$$

The targets for lift and drag are denoted by C_{L_T}, and C_{D_T} respectively. The target lift coefficient is set to 1.0 and the target drag is set to .0223 which is the baseline value. The weights are set to $1e-2$ and 100 respectively, weighting the drag so heavily was done to ensure constraint of the optimization. We can compare the optimization behavior for the steady state adjoint to that of the approximate linearizations with varying iteration controllers to get a sense as to how the design cycle is impacted by the adjoint iteration controllers for each nonlinear controller.

The design comparison shown in Figure 10 show that the optimizer acheives the objective by slimming the airfoil and introducing a slight bend in the back to make a flap. Only one final design is shown because the different adjoint iteration and nonlinear constraint controllers return very similar shapes and optimal points, showing that the inexact linearization algorithms are effective. In fact it appears that while the inexact linearizations without adjoint iteration controllers show ill behaved optimality behavior when compared to the steady-state optimizations, this problem is solved by the inclusion of the adjoint iteration controllers; in fact these adjoint iteration controllers make the adjoint computation through inexact linearization better behaved than the steady state method when measured by optimality with Figure 8 showing a final steady-



(a) NC($\eta_\Lambda = c$)

(b) NC($\eta_\Lambda = f(\|R(u)\|)$)

(c) IQN($\eta_\Lambda = c$)

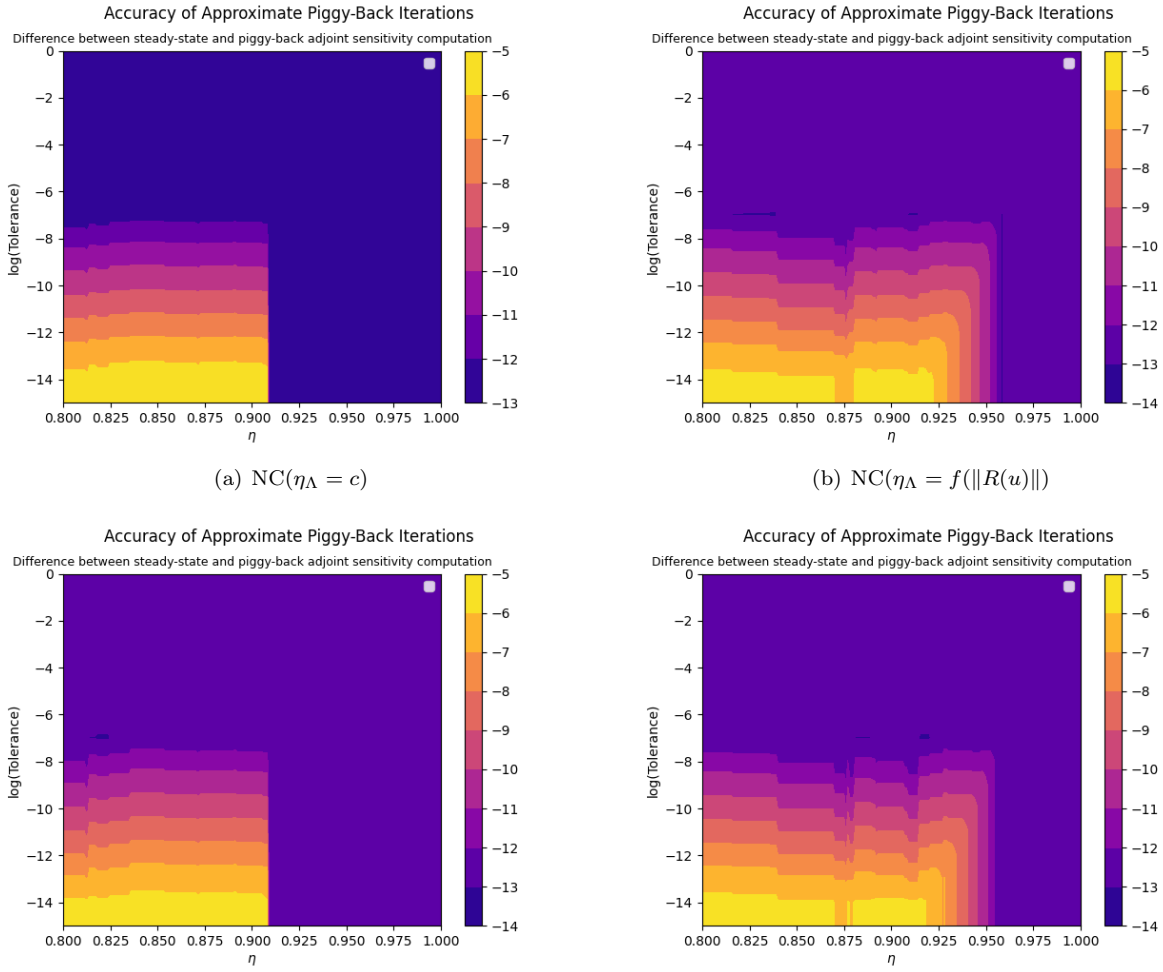
(d) IQN($\eta_\Lambda = f(\|R(u)\|)$)

Figure 4: Parametric study of adjoint iterations skipped

state adjoint design with a slightly higher objective function when it terminates due to numerical difficulty. This is an encouraging result, as it indicates that we gain noticeable benefits despite making our adjoint system less expensive to solve. As an example the final design variables in the case that reaches optimality (using $\eta_\Lambda = c$) and the one with functionally infinite optimality are nearly identical as shown in Table 2, with the sensitivity comparison being shown in Table 3 which shows that the sensitivities on the upper surface (the first four design variables) are notably larger for the "diverged" optimization hence the lack of convergence in the optimality problem. This result does show the need for more robust theory for the inexact constraints as applied to quasi-Newton solvers as well as possible modifications for bound constrained problems.

6 Conclusions

This work contains the development and experimentation of two methods of adaptive iterative control on piggy-back iterations of the one-shot adjoint. These two methods are introduced and then verified by comparison (in terms of time to solution and accuracy) to the complex-step finite-difference sensitivities and the steady-state adjoint computed ones. This is followed by a comparison (along the previous criteria) through a parametric study to show the behavior of these iterations for varied η_Λ and τ_Λ for both adaptive iterative controls and approximate linearizations. This showed that it is possible to skip many iterations before seeing noticeable degradation in gradient accuracy, allowing for accurate adjoint computation with the same



(a) NC($\eta_\Lambda = c$)

(b) NC($\eta_\Lambda = f(\|R(u)\|)$)

(c) IQN($\eta_\Lambda = c$)

(d) IQN($\eta_\Lambda = f(\|R(u)\|)$)

Figure 5: Parametric study of adjoint accuracy measured by the log_{10} of the norm of the difference

memory footprint as the primal problem. Finally it was shown that these methods can be combined with adaptive constraint tolerance method to get less expensive and more successful design cycles. This is an encouraging result which motivates development of further theory and inexact constraint controllers for bound constrained quasi-Newton optimization algorithms. Furthermore, the utility of these adjoint systems for unconverged primal states in the intermediate design cycles encourages further use the inexact linearizations and adaptive iterative controls to the one-shot method for optimization, also known as simulation analysis and design.

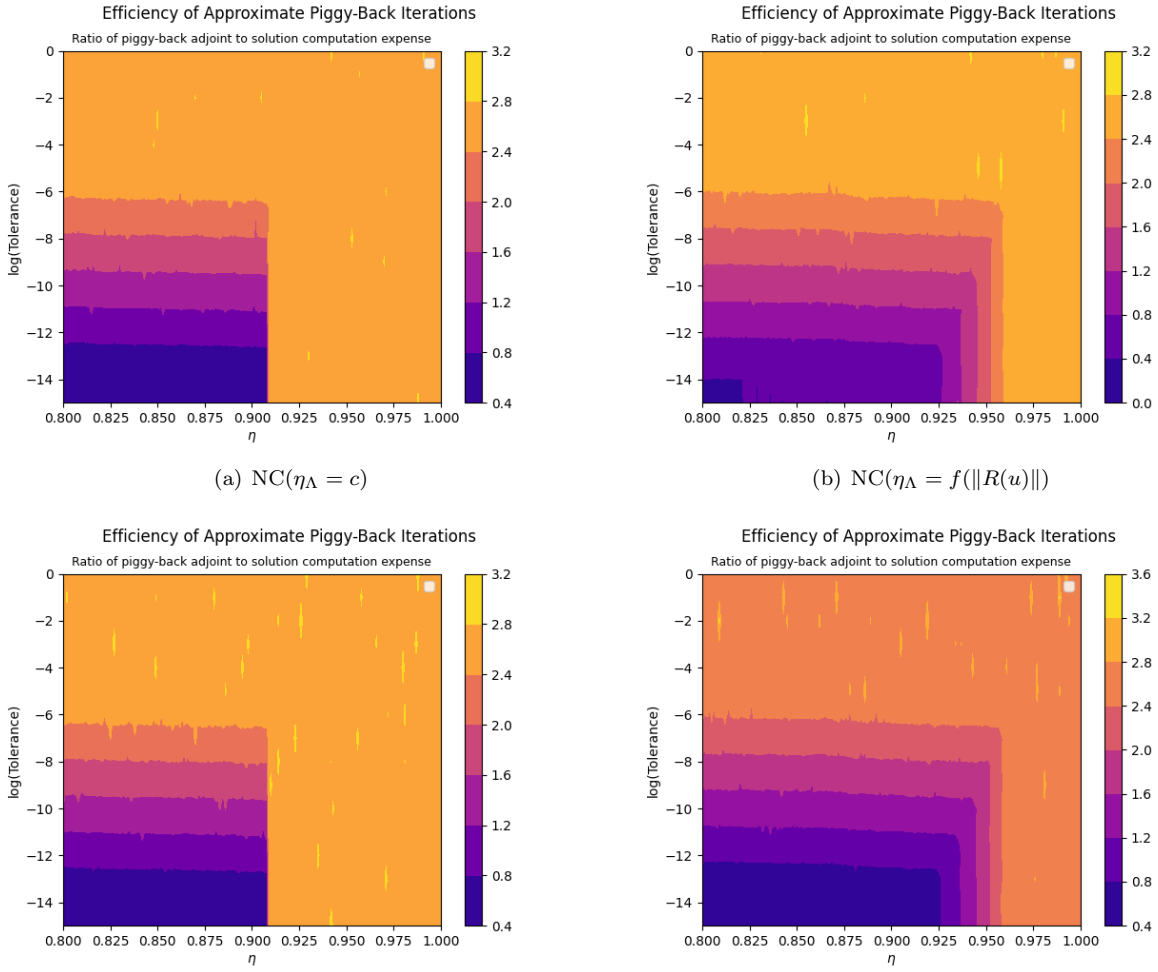
7 Acknowledgments

Computing time was provided by ARCC on the Teton supercomputer.

References

[1] Philip E. Gill, Walter Murray, and Michael A. Saunders. UserâĂŹs guide for snopt version 7: Software for large-scale nonlinear programming.

[2] Philip E Gill, Walter Murray, and Michael A Saunders. Snopt: An sqp algorithm for large-scale constrained optimization. *SIAM review*, 47(1):99–131, 2005.



(a) NC($\eta_\Lambda = c$)

(b) NC($\eta_\Lambda = f(\|R(u)\|)$)

(c) IQN($\eta_\Lambda = c$)

(d) IQN($\eta_\Lambda = f(\|R(u)\|)$)

Figure 6: Parametric study of adjoint computation time divided by the nonlinear-problem time to solution

[3] B.M. Adams, L.E. Bauman, W.J. Bohnhoff, and et al. Dakota, a multilevel parallel object-oriented framework for design optimization, parameter estimation, uncertainty quantification, and sensitivity analysis: Version 6.0 userâĂŹs manual, 2015.

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

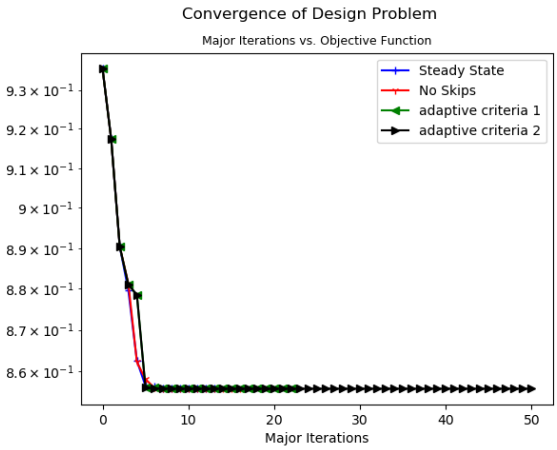
[5] Stefanie Günther, Nicolas R. Gauger, and Qiqi Wang. Simultaneous single-step one-shot optimization with unsteady pdes. *J. Comput. Appl. Math.*, 294:12–22, 2016.

[6] Eyal Arian and Shlomo Ta'asan. Shape optimization in one shot. In Jeffrey Borggaard, John Burkardt, Max Gunzburger, and Janet Peterson, editors, *Optimal Design and Control*, pages 23–40, Boston, MA, 1995. Birkhäuser Boston.

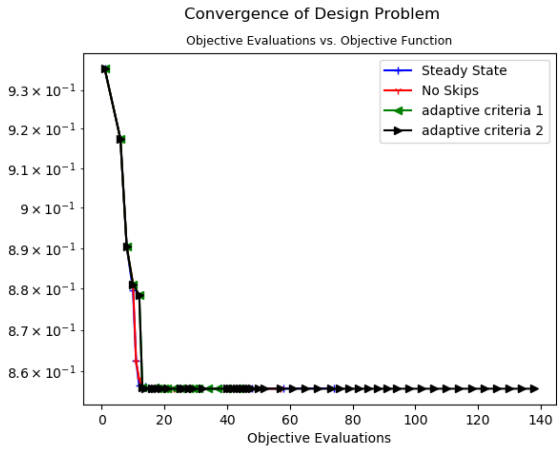
[7] Doug Shi-Dong and Sivakumaran Nadarajah. Newton-krylov full-space aerodynamic shape optimization. 59th AIAA Aerospace Sciences Meeting, AIAA Paper 2021-0281, Virtual Event, January 2021. https://doi.org/10.2514/6.2021-0281.

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

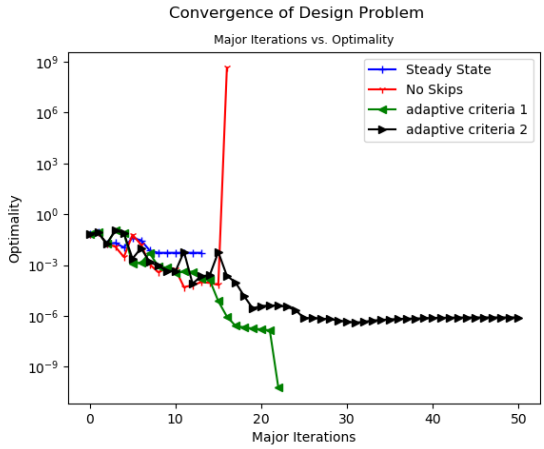
[9] Siamak Akbarzadeh, Jan Huckelheim, and Jens-Dominik Muller. Consistent treatment of incompletely converged iterative linear solvers in reverse-mode algorithmic differentiation. Computational Optimization and Applications, 10.1007/s10589-020-00214-x.



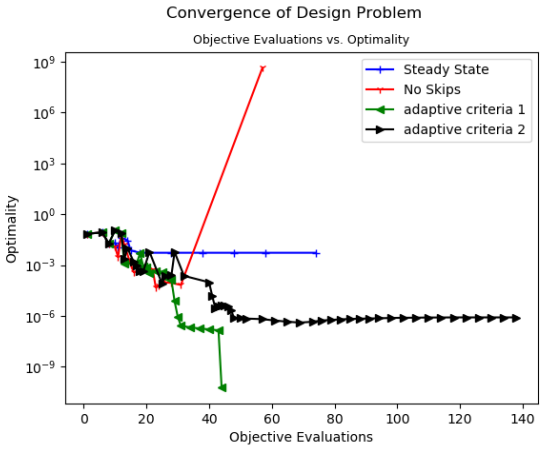
(a) Major Iterations vs. Objective



(b) Objective Evaluations vs. Objective



(c) Major Iterations vs. Optimality



(d) Objective Evaluations vs. Optimality

Figure 7: Design cycle summary for default nested optimization

[10] Emmett Padway and Dimitri J. Mavriplis. Inexact linearization of fixed point iterations in the piggyback iterations of the one-shot adjoint. AIAA SciTech 2022 Forum, AIAA Paper 2022-1613, San Diego, CA, January 2022, https://arc.aiaa.org/doi/abs/10.2514/6.2022-1613.

[11] Emmett Padway and Dimitri J. Mavriplis. 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.

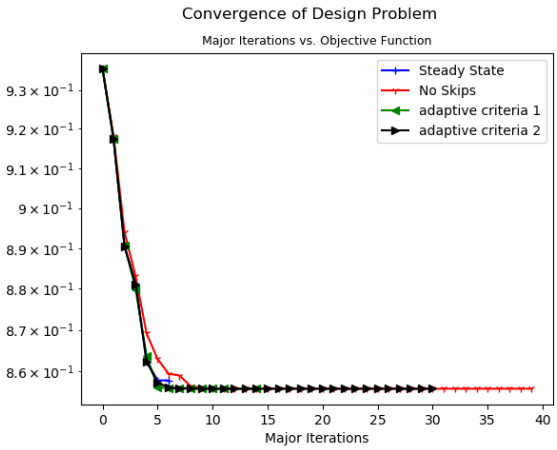
[12] Emmett Padway and Dimitri J. Mavriplis. Advances in the pseudo-time accurate formulation of the adjoint and tangent systems for sensitivity computation and design. 59th AIAA Aerospace Sciences Meeting, AIAA Paper 2020-3136, Virtual Event, June 2020. https://doi.org/10.2514/6.2020-3136.

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

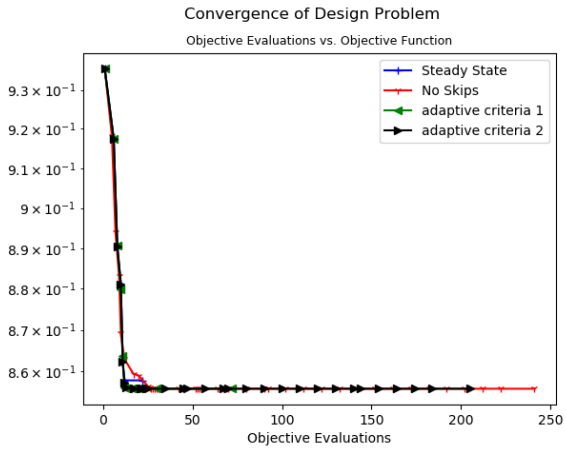
[14] Emmett Padway and Dimitri Mavriplis. Approximate linearization of fixed point iterations: Error analysis of tangent and adjoint problems linearized about non-stationary points. Arxiv, https://arxiv.org/abs/2104.02826.

[15] Dimitri Mavriplis. Revisiting the least-squares procedure for gradient reconstruction on unstructured meshes. 16th AIAA Computational Fluid Dynamics Conference, Fluid Dynamics and Co-located Conferences, AIAA Paper 2003-3986, Orlando, Florida, 06/2003. https://doi.org/10.2514/6.2003-3986.

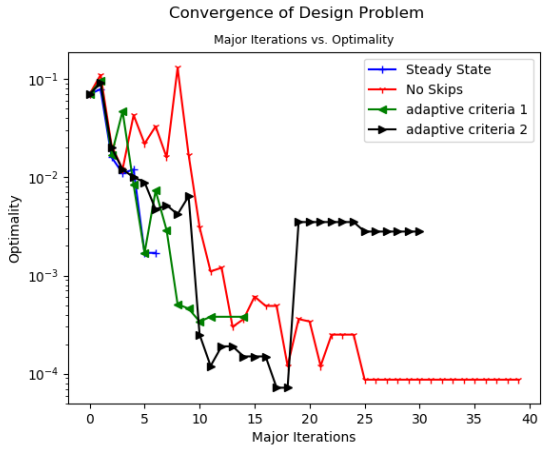
[16] Venkat Venkatakrishnan. On the accuracy of limiters and convergence to steady state so-



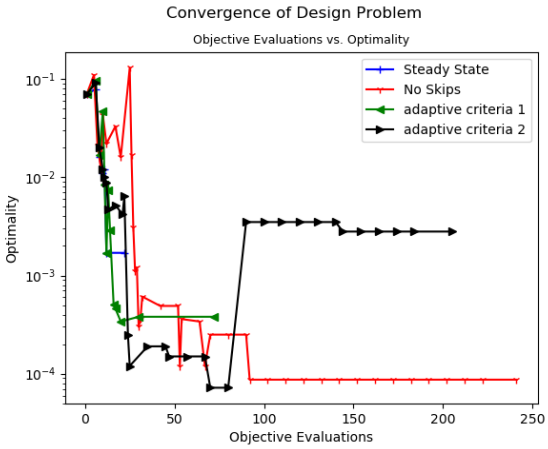
(a) Major Iterations vs. Objective



(b) Objective Evaluations vs. Objective



(c) Major Iterations vs. Optimality



(d) Objective Evaluations vs. Optimality

Figure 8: Design cycle summary for first nonlinear constraint controller

lutions. 31st Aerospace Sciences Meeting, AIAA Paper 1993-880, Reno NV, January 1993. https://doi.org/10.2514/6.1993-880.

[17] Dimitri J. Mavriplis. A residual smoothing strategy for accelerating newton method continuation. *Computers and Fluids*, 220:104859, 2021.

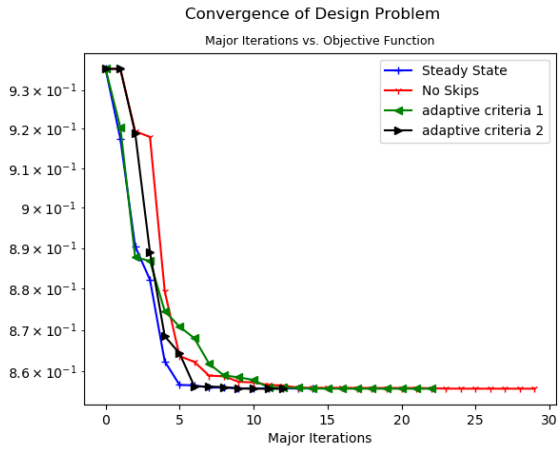
[18] Behzad R. Ahrabi and Dimitri J. Mavriplis. An implicit block ilu smoother for preconditioning of newtonâĂŞkrylov solvers with application in high-order stabilized finite-element methods. *Computer Methods in Applied Mechanics and Engineering*, 358:112637, 2020.

[19] Yousef Saad. *Iterative methods for sparse linear systems*, volume 82. Society for Industrial and Applied Mathematics, 2003.

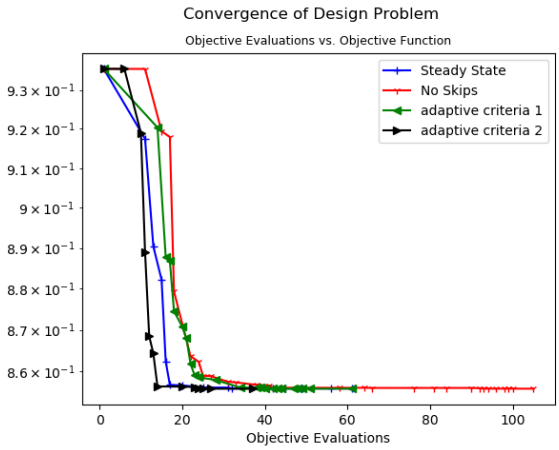
[20] Dimitri J. Mavriplis. Vki lecture series: 38th advanced computational fluid dynamics. adjoint methods and their application in cfd, time dependent adjoint methods for single and multi-disciplinary problems. *VKI Lecture Notes, von Karman Institute for Fluid Dynamics, Rhode St-Genese, Belgium*, Sep 2015.

[21] Edward Luke, Eric Collins, and Eric Blades. A fast mesh deformation method using explicit interpolation. *J. Comput. Physics*, 231:586–601, 01 2012.

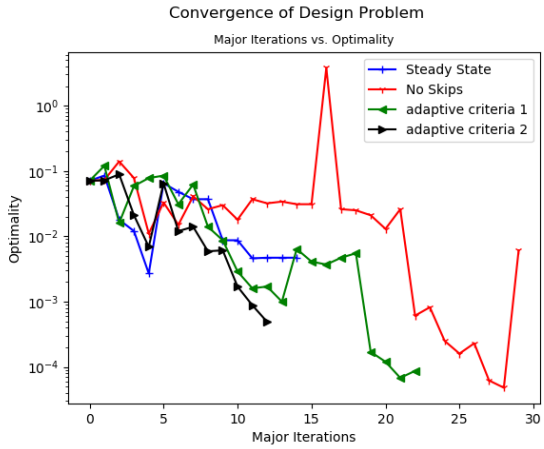
[22] David A. Brown and Sivakumaran Nadarajah. An adaptive constraint tolerance method for optimization algorithms based on the discrete adjoint method. 2018 AIAA/ASCE/AHS/ASC Structures, Structural Dynamics, and Materials Conference, AIAA SciTech Forum, AIAA Paper 2018-0414, Kissimmee, Florida, 01/2018, https://doi.org/10.2514/6.2018-0414.



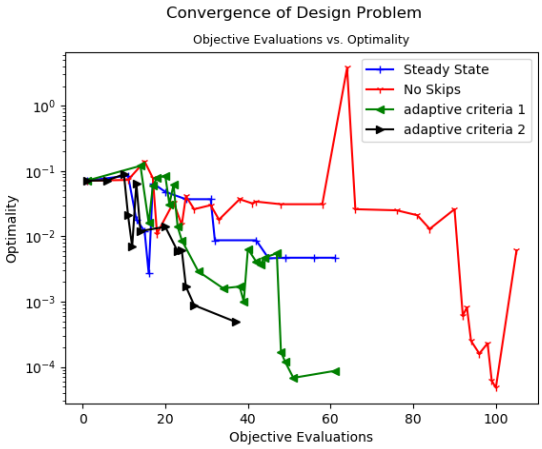
(a) Major Iterations vs. Objective



(b) Objective Evaluations vs. Objective



(c) Major Iterations vs. Optimality

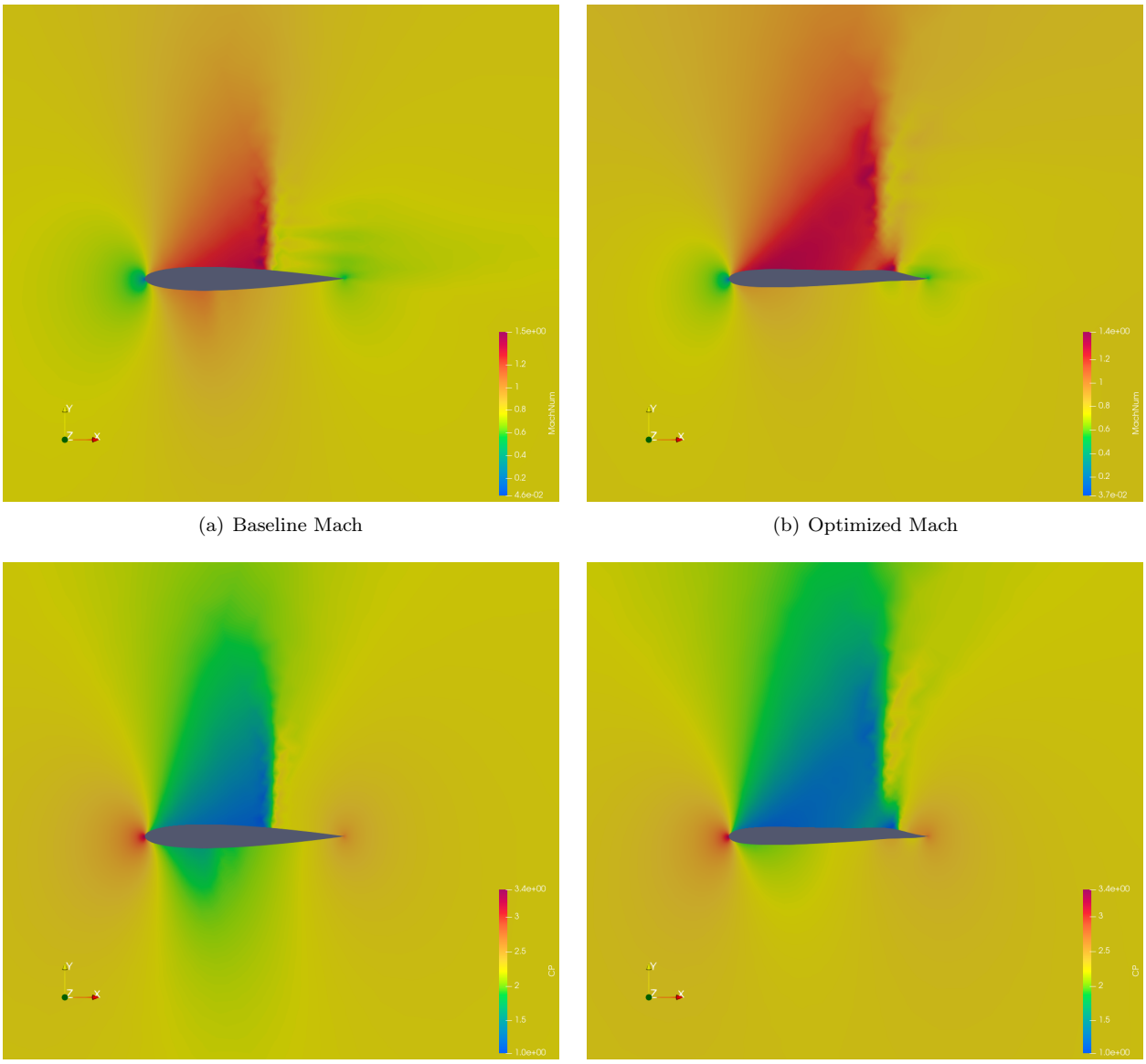


(d) Objective Evaluations vs. Optimality

Figure 9: Design cycle summary for second nonlinear constraint controller

[23] David A. Brown and Siva K. Nadarajah. Effect of inexact adjoint solutions on the discrete-adjoint approach to gradient-based optimization. *Optimization in Engineering*, 56(4):1532–1540, 2021.

[24] A. Griewank and A. Ponomarenko. Time-lag in derivative convergence for fixed point iterations. In *Proceedings of CARI'04, 7th African Conference on Research in Computer Science*, pages 295–304, 2004.



(a) Baseline Mach

(b) Optimized Mach

(c) Baseline Pressure Coefficient

(d) Optimized Pressure Coefficient

Figure 10: Comparison of baseline and final design

	Converged	Diverged
Design Variable 1	-5.7241083073080E-03	-5.7260097131611E-03
Design Variable 2	-9.2672623516420E-03	-9.2444716268683E-03
Design Variable 3	-4.6663841085040E-04	-4.8372530029862E-04
Design Variable 4	1.6279355919850E-02	1.6271010797993E-02
Design Variable 5	-1.0000000000000E-02	-9.9999999996993E-03
Design Variable 6	-1.0000000000000E-02	-9.9999999996993E-03
Design Variable 7	-1.0000000000000E-02	-1.0000000000000E-02
Design Variable 8	-1.0000000000000E-02	-1.0000000000000E-02

Table 2: Comparison of design variables for design cycles with converged and diverged optimality .

	Converged	Diverged
Design Variable 1	-1.078741346249856E-009	3.916765831407515E-004
Design Variable 2	-1.310834427115104E-010	6.641176189499734E-004
Design Variable 3	-1.551630247897950E-010	-2.289093542261270E-004
Design Variable 4	5.786487955461439E-011	-1.545606830304025E-003
Design Variable 5	0.379856367365289	0.379834642187345
Design Variable 6	0.444008637643953	0.444056093139126
Design Variable 7	0.489461963395485	0.489534638041718
Design Variable 8	0.632743674851339	0.632855810802217

Table 3: Comparison of sensitivities for design cycles with converged and diverged optimality .