Tangent and Adjoint Problems in Partially Converged Flows

Emmett Padway

September 11th, 2020

University of Wyoming CFD Lab

Outline

- Introduction

Outline

- Introduction
- Motivation and approach

Outline

- Introduction
- Motivation and approach
- Pseudo-time accurate formulation of tangent and adjoint systems

Outline

- Introduction
- Motivation and approach
- Pseudo-time accurate formulation of tangent and adjoint systems
- Error due to approximate linearization of the fixed point

Outline

- Introduction
- Motivation and approach
- Pseudo-time accurate formulation of tangent and adjoint systems
- Error due to approximate linearization of the fixed point
- Effect of averaging the objective function on the sensitivity behavior

Outline

- Introduction
- Motivation and approach
- Pseudo-time accurate formulation of tangent and adjoint systems
- Error due to approximate linearization of the fixed point
- Effect of averaging the objective function on the sensitivity behavior
- Application of pseudo-time accurate approach to aerodynamic shape optimization

Outline

- Introduction
- Motivation and approach
- Pseudo-time accurate formulation of tangent and adjoint systems
- Error due to approximate linearization of the fixed point
- Effect of averaging the objective function on the sensitivity behavior
- Application of pseudo-time accurate approach to aerodynamic shape optimization
- Application of dual weighted constraint/fixed-point method

Outline

- Introduction
- Motivation and approach
- Pseudo-time accurate formulation of tangent and adjoint systems
- Error due to approximate linearization of the fixed point
- Effect of averaging the objective function on the sensitivity behavior
- Application of pseudo-time accurate approach to aerodynamic shape optimization
- Application of dual weighted constraint/fixed-point method
- Conclusions/future work

Introduction

Introduction: CFD and Optimization

- CFD has become more widespread and practical as it has become more automated.

Introduction: CFD and Optimization

- CFD has become more widespread and practical as it has become more automated.
- This has lead to a focus on automated design, mostly with gradient based approaches.

Introduction: CFD and Optimization

- CFD has become more widespread and practical as it has become more automated.
- This has lead to a focus on automated design, mostly with gradient based approaches.
- With an increased role in design has come greater coupling between analysis and optimization codes.

Introduction: CFD and Optimization

- CFD has become more widespread and practical as it has become more automated.
- This has lead to a focus on automated design, mostly with gradient based approaches.
- With an increased role in design has come greater coupling between analysis and optimization codes.
- Due to decades of engineering we can assume the baseline design will be optimized to the global minimum.

Introduction: the Design Problem

$$L = L(U(x_v(x_s(D))), x_v(x_s(D))) \tag{1}$$

where:

- L is the objective function to be optimized
- u is the conservative variable vector
- x_v is the volume mesh coordinate vector
- x_s is the surface mesh coordinate vector
- D is the design variable vector

Introduction: the Design Problem

$$L = L(U(x_v(x_s(D))), x_v(x_s(D))) \tag{1}$$

where:

- L is the objective function to be optimized
- u is the conservative variable vector
- x_v is the volume mesh coordinate vector
- x_s is the surface mesh coordinate vector
- D is the design variable vector

Introduction: the Design Problem

$$L = L(U(x_v(x_s(D))), x_v(x_s(D))) \tag{1}$$

where:

- L is the objective function to be optimized
- u is the conservative variable vector
- x_v is the volume mesh coordinate vector
- x_s is the surface mesh coordinate vector
- D is the design variable vector

Introduction: the Design Problem

$$L = L(U(x_v(x_s(D))), x_v(x_s(D))) \tag{1}$$

where:

- L is the objective function to be optimized
- u is the conservative variable vector
- x_v is the volume mesh coordinate vector
- x_s is the surface mesh coordinate vector
- D is the design variable vector

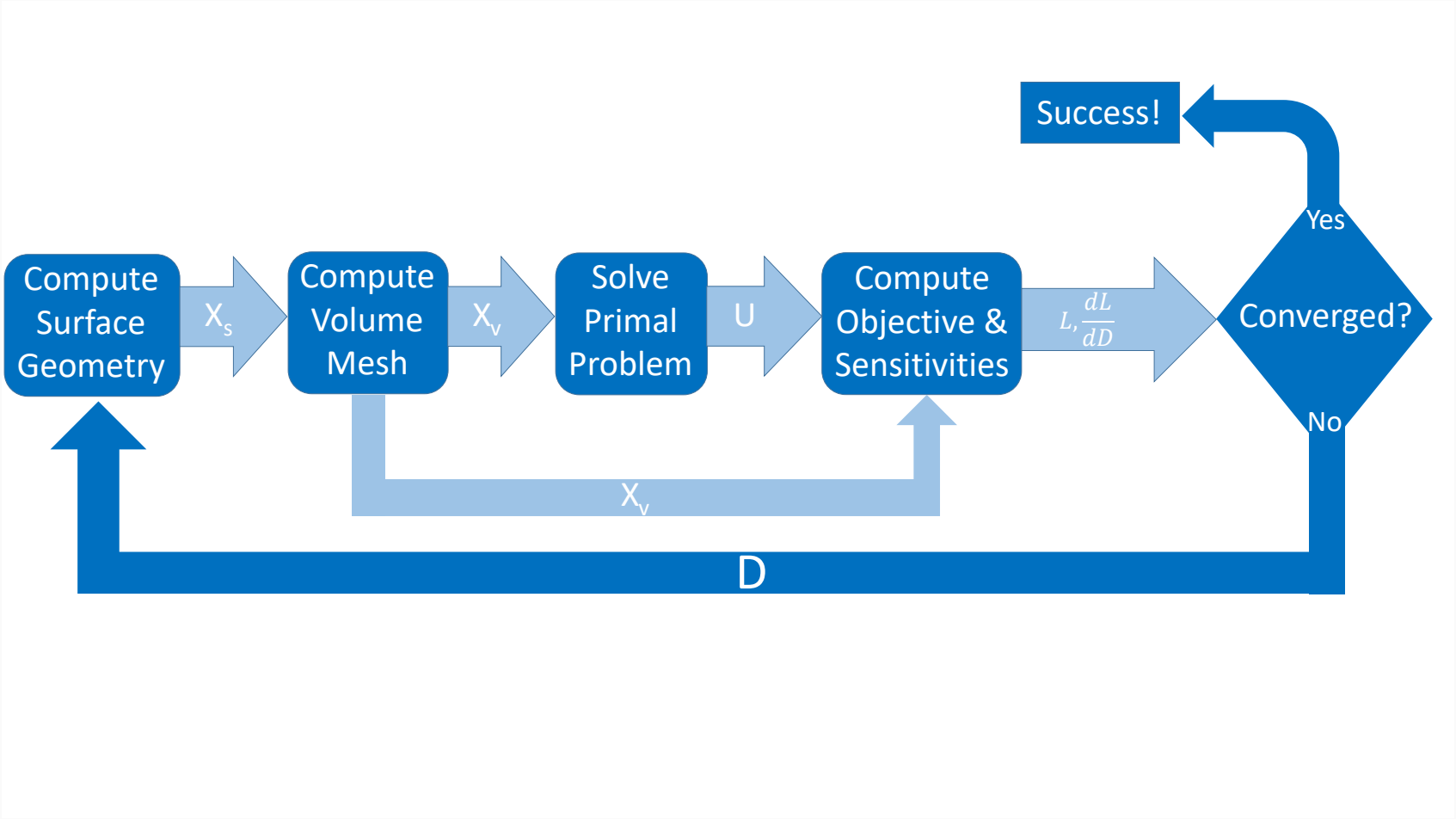
Introduction: the Design Problem

$$L = L(U(x_v(x_s(D))), x_v(x_s(D))) \tag{1}$$

where:

- L is the objective function to be optimized
- u is the conservative variable vector
- x_v is the volume mesh coordinate vector
- x_s is the surface mesh coordinate vector
- D is the design variable vector

Introduction: the Design Problem



Design Process Flow Chart

Introduction: Tangent/Forward Linearization

$$\frac{dL}{dD} = \frac{\partial L}{\partial x_v}\frac{dx_v}{dx_s}\frac{dx_s}{dD} + \frac{\partial L}{\partial u}\frac{du}{dx_v}\frac{dx_v}{dx_s}\frac{dx_s}{dD} \tag{2}$$

Introduction: Tangent/Forward Linearization

For a converged flow the discretized form of the governing equations is satisfied – $R = 0$ – and therefore the total derivative is equal to zero as well:

$$\left[\frac{\partial R}{\partial x_v}\right] \frac{dx_v}{dx_s} \frac{dx_s}{dD} + \left[\frac{\partial R}{\partial u}\right] \frac{du}{dx_v} \frac{dx_v}{dx_s} \frac{dx_s}{dD} = 0 \tag{3}$$

Introduction: Tangent/Forward Linearization

For a converged flow the discretized form of the governing equations is satisfied – $R = 0$ – and therefore the total derivative is equal to zero as well:

$$\left[\frac{\partial R}{\partial x_v}\right] \frac{dx_v}{dx_s} \frac{dx_s}{dD} + \left[\frac{\partial R}{\partial u}\right] \frac{du}{dx_v} \frac{dx_v}{dx_s} \frac{dx_s}{dD} = 0 \tag{3}$$

Isolating the right-hand side:

$$\left[\frac{\partial R}{\partial u}\right] \frac{du}{dx_v} \frac{dx_v}{dx_s} \frac{dx_s}{dD} = -\left[\frac{\partial R}{\partial x_v}\right] \frac{dx_v}{dx_s} \frac{dx_s}{dD} \tag{4}$$

Introduction: Tangent/Forward Linearization

We could solve for $\frac{du}{dx_v}$ by dividing both sides by $\frac{dx_v}{dx_s}\frac{dx_s}{dD}$.

$$\left[\frac{\partial R}{\partial u}\right]\frac{du}{dx_v} = -\left[\frac{\partial R}{\partial x_v}\right] \tag{5}$$

Introduction: Tangent/Forward Linearization

We could solve for $\frac{du}{dx_v}$ by dividing both sides by $\frac{dx_v}{dx_s}\frac{dx_s}{dD}$.

$$\left[\frac{\partial R}{\partial u}\right]\frac{du}{dx_v} = -\left[\frac{\partial R}{\partial x_v}\right] \tag{5}$$

but this would scale with the mesh, and so we don't divide both sides and rewrite the equation in simplified notation as below:

$$\left[\frac{\partial R}{\partial u}\right]\frac{du}{dD} = -\left[\frac{\partial R}{\partial x}\right]\frac{dx}{dD} \tag{6}$$

Introduction: Tangent/Forward Linearization

The sensitivity equation becomes in reduced notation:

$$\frac{dL}{dD} = \frac{\partial L}{\partial x}\frac{dx}{dD} - \frac{\partial L}{\partial u}\left[\frac{\partial R}{\partial u}\right]^{-1}\left[\frac{\partial R}{\partial x}\right]\frac{dx}{dD} \tag{7}$$

Introduction: Adjoint/Reverse Linearization

The adjoint equation begins from the objective function augmented with a Lagrangian vector and constraint equal to zero:

$$J = L(U(x_v(x_s(D))), x_v(x_s(D))) + \Lambda^T R(U(x_v(x_s(D))), x_v(x_s(D))) \quad (8)$$

Introduction: Adjoint/Reverse Linearization

The adjoint equation begins from the objective function augmented with a Lagrangian vector and constraint equal to zero:

$$J = L(U(x_v(x_s(D))), x_v(x_s(D))) + \Lambda^T R(U(x_v(x_s(D))), x_v(x_s(D))) \quad (8)$$

Taking the total derivative:

$$\frac{dJ}{dD} = \left(\frac{\partial L}{\partial x_v} + \frac{\partial L}{\partial u}\frac{du}{dx_v}\right)\frac{dx_v}{dx_s}\frac{dx_s}{dD} + \Lambda^T\left[\left(\frac{\partial R}{\partial x_v} + \frac{\partial R}{\partial u}\frac{du}{dx_v}\right)\frac{dx_v}{dx_s}\frac{dx_s}{dD}\right] \quad (9)$$

Introduction: Adjoint/Reverse Linearization

By grouping like terms we can obtain the below expression:

$$\frac{dJ}{dD} = \left(\frac{\partial L}{\partial x_v} + \Lambda^T \frac{\partial R}{\partial x_v}\right) \frac{dx_v}{dx_s} \frac{dx_s}{dD} + \left(\frac{\partial L}{\partial u} + \Lambda^T \frac{\partial R}{\partial u}\right) \frac{du}{dx_v} \frac{dx_v}{dx_s} \frac{dx_s}{dD} \quad (10)$$

Introduction: Adjoint/Reverse Linearization

By grouping like terms we can obtain the below expression:

$$\frac{dJ}{dD} = \left(\frac{\partial L}{\partial x_v} + \Lambda^T \frac{\partial R}{\partial x_v}\right) \frac{dx_v}{dx_s} \frac{dx_s}{dD} + \left(\frac{\partial L}{\partial u} + \Lambda^T \frac{\partial R}{\partial u}\right) \frac{du}{dx_v} \frac{dx_v}{dx_s} \frac{dx_s}{dD} \tag{10}$$

To avoid computing $\frac{du}{dx_v} \frac{dx_v}{dx_s} \frac{dx_s}{dD}$ we define an adjoint equation:

$$\left[\frac{\partial R}{\partial u}\right]^T \Lambda = -\left[\frac{\partial L}{\partial u}\right]^T \tag{11}$$

Introduction: Adjoint/Reverse Linearization

This gives the following expression for the sensitivities

$$\frac{dL}{dD} = \left(\frac{\partial L}{\partial x_v} + \Lambda^T \frac{\partial R}{\partial x_v} \right) \frac{dx_v}{dx_s} \frac{dx_s}{dD} \tag{12}$$

Introduction: Adjoint/Reverse Linearization

This gives the following expression for the sensitivities

$$\frac{dL}{dD} = \left(\frac{\partial L}{\partial x_v} + \Lambda^T \frac{\partial R}{\partial x_v} \right) \frac{dx_v}{dx_s} \frac{dx_s}{dD} \tag{12}$$

We then define a mesh adjoint:

$$\frac{dL}{dD} = \Lambda_{x_s}^T \frac{dx_s}{dD} \tag{13}$$

Introduction: Adjoint/Reverse Linearization

This gives the following expression for the sensitivities

$$\frac{dL}{dD} = \left(\frac{\partial L}{\partial x_v} + \Lambda^T \frac{\partial R}{\partial x_v} \right) \frac{dx_v}{dx_s} \frac{dx_s}{dD} \tag{12}$$

We then define a mesh adjoint:

$$\frac{dL}{dD} = \Lambda_{x_s}^T \frac{dx_s}{dD} \tag{13}$$

For simplicity, the sensitivity equation used is written as:

$$\frac{dL}{dD} = \frac{\partial L}{\partial D} + \Lambda^T \frac{\partial R}{\partial D} \tag{14}$$

Introduction: Tangent and Adjoint Method Summary

- The tangent is called the forward linearization because it begins with a perturbation to the design variables and is propagated through to the objective.

Introduction: Tangent and Adjoint Method Summary

- The tangent is called the forward linearization because it begins with a perturbation to the design variables and is propagated through to the objective.
- In contrast, the adjoint begins with the perturbation to the objective function and back propagates through the objective function evaluation to the design variables.

Introduction: Tangent and Adjoint Method Summary

- The tangent is called the forward linearization because it begins with a perturbation to the design variables and is propagated through to the objective.
- In contrast, the adjoint begins with the perturbation to the objective function and back propagates through the objective function evaluation to the design variables.
- Both also require solution of a linear system to provide sensitivities.

Introduction: Tangent and Adjoint Method Summary

- The tangent is called the forward linearization because it begins with a perturbation to the design variables and is propagated through to the objective.
- In contrast, the adjoint begins with the perturbation to the objective function and back propagates through the objective function evaluation to the design variables.
- Both also require solution of a linear system to provide sensitivities.
- The tangent system has as many right hand-sides as design variables.

Introduction: Tangent and Adjoint Method Summary

- The tangent is called the forward linearization because it begins with a perturbation to the design variables and is propagated through to the objective.
- In contrast, the adjoint begins with the perturbation to the objective function and back propagates through the objective function evaluation to the design variables.
- Both also require solution of a linear system to provide sensitivities.
- The tangent system has as many right hand-sides as design variables.
- The adjoint system scales with objective functions (or quantities of interest).

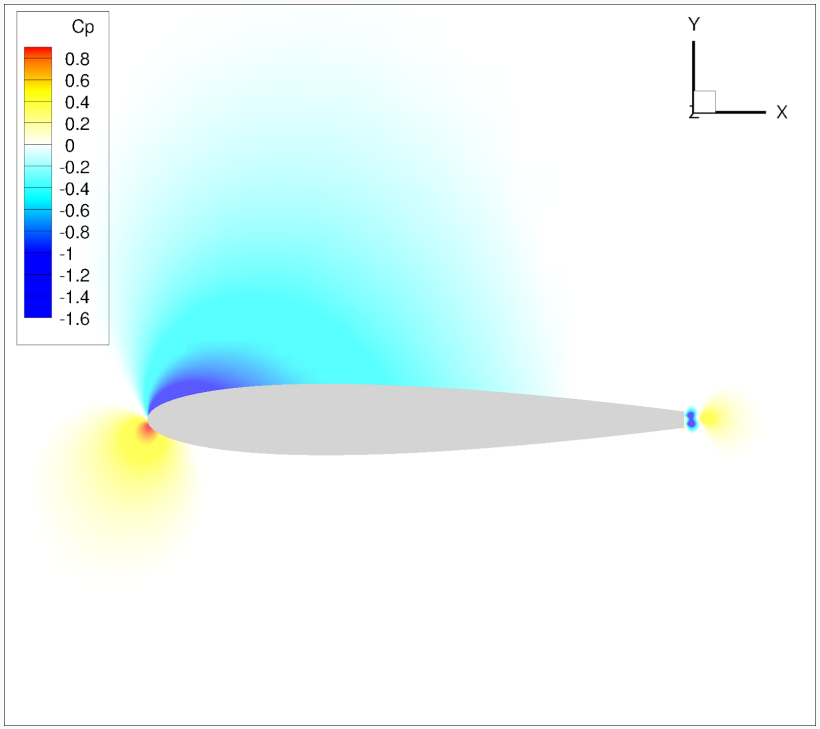
Introduction: Tangent and Adjoint Method Summary

- The tangent is called the forward linearization because it begins with a perturbation to the design variables and is propagated through to the objective.
- In contrast, the adjoint begins with the perturbation to the objective function and back propagates through the objective function evaluation to the design variables.
- Both also require solution of a linear system to provide sensitivities.
- The tangent system has as many right hand-sides as design variables.
- The adjoint system scales with objective functions (or quantities of interest).
- Both methods require that the discretized governing equations be satisfied.

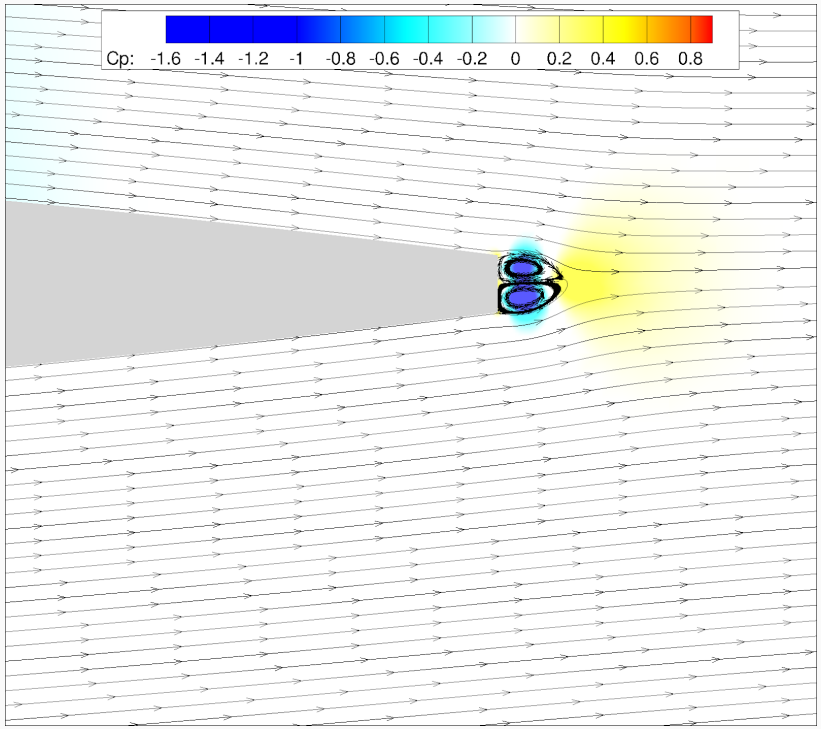
Motivation and Approach

Motivation: Non-Zero Residual?

As the field has moved to higher order and more complex geometries convergence issues have become more common. A simple example is that of the truncated NACA0012 airfoil:



(a) Full Airfoil

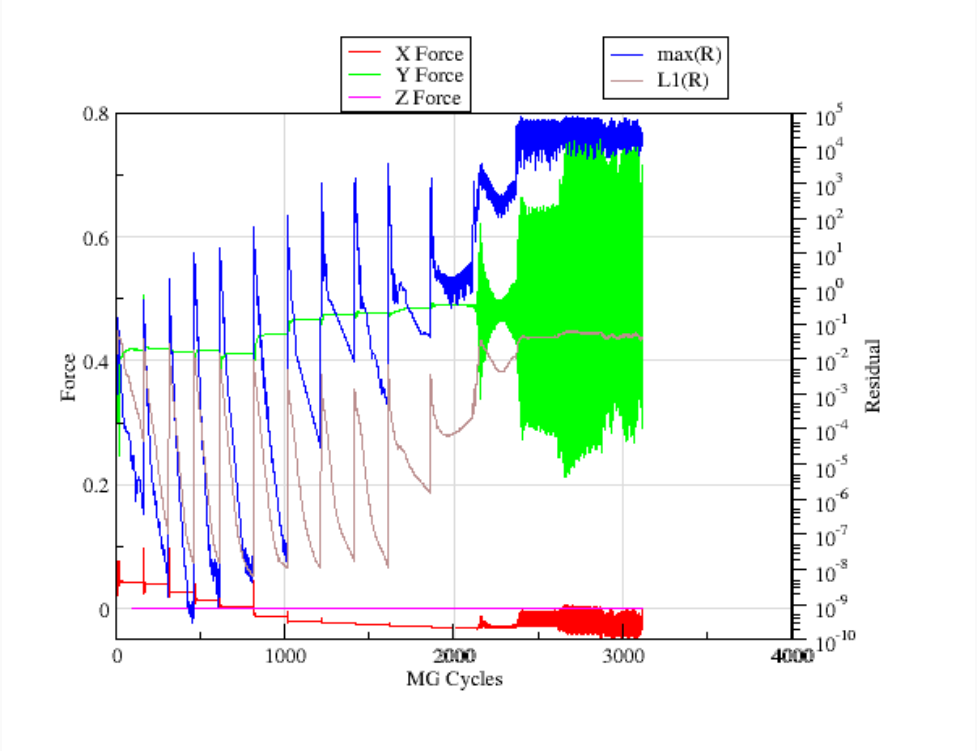


(b) Trailing Edge with Streamlines

Non-converging simulation for NACA0012 airfoil with blunt trailing edge

Motivation: Non-Zero Residual?

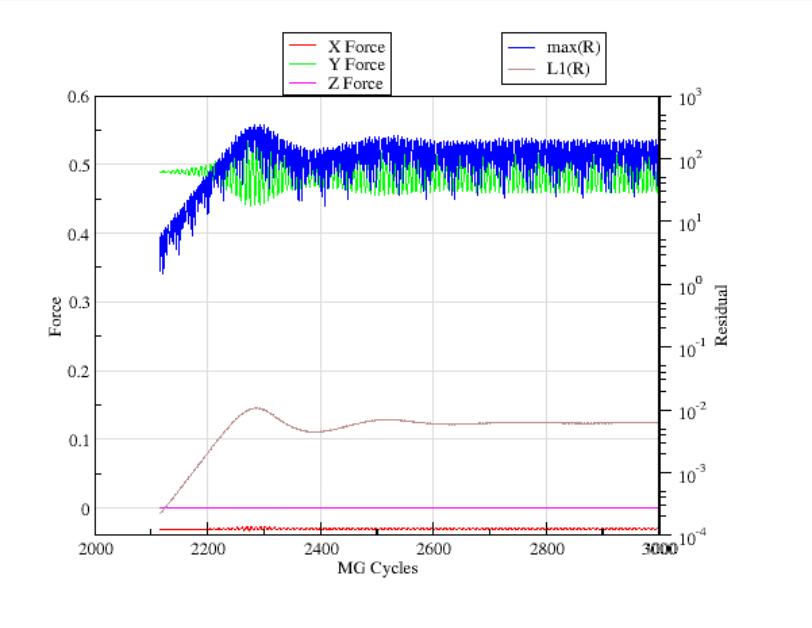
The lack of convergence shows itself in oscillatory behavior of the nonlinear problem and this figure shows good convergence for the early meshes and poor behavior on the finer meshes that are less dissipative



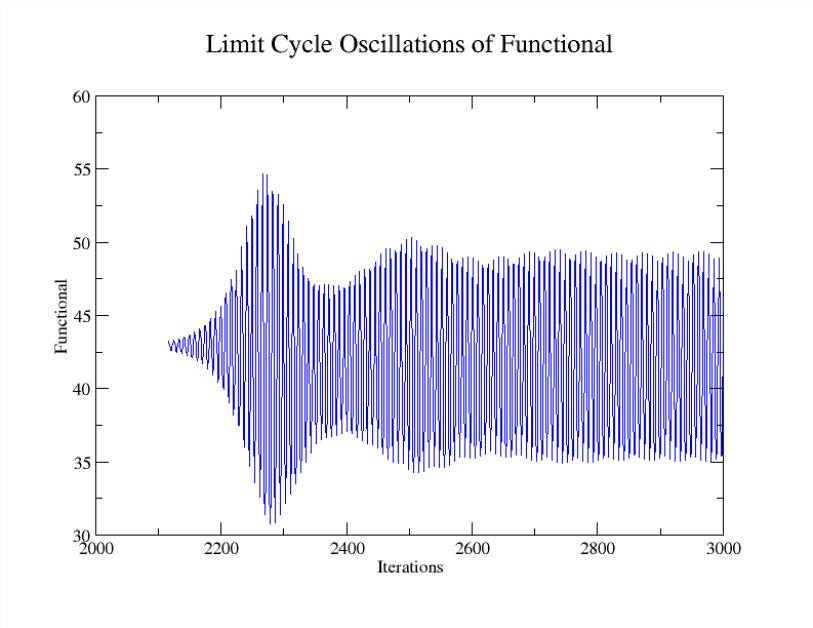
Convergence history for adaptive simulation with Cart3D

Motivation: Non-Zero Residual?

If we run for many iterations on the first non-convergent mesh we can see the behavior of these limit cycle oscillations:



(a) Residual

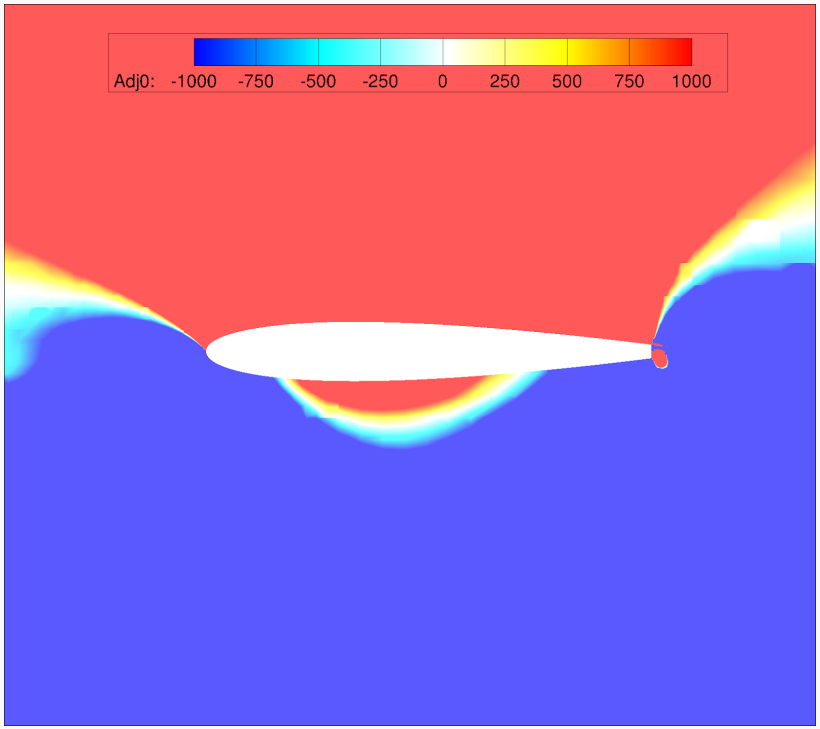


(b) Functional

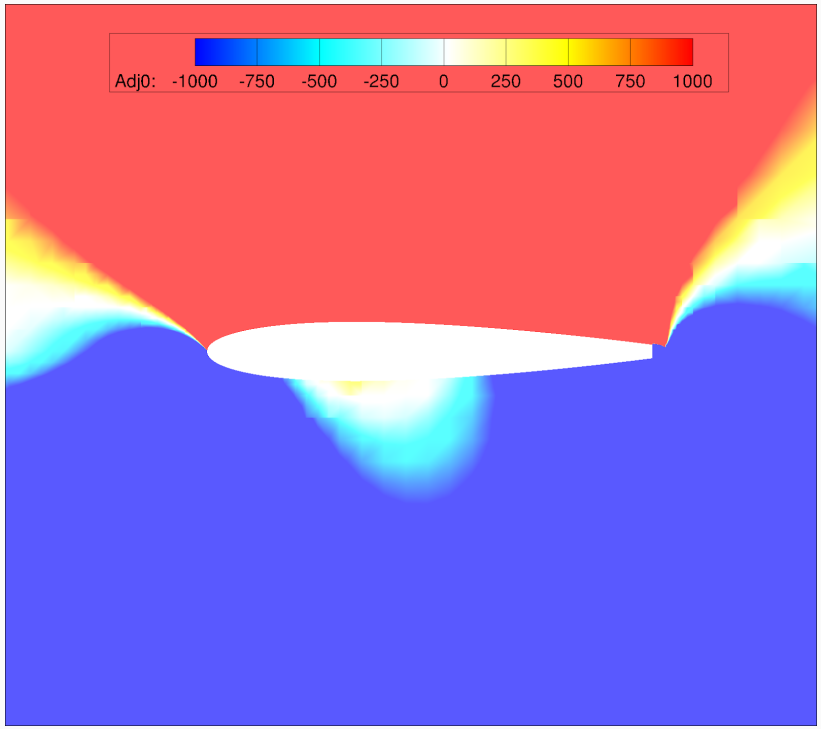
Limit cycle oscillations for first non-convergent mesh (10th mesh)

Motivation: Non-Zero Residual?

When we compare the adjoint fields between the mean residual state and the minimum residual state we see notable differences:



(a) Mean Residual State

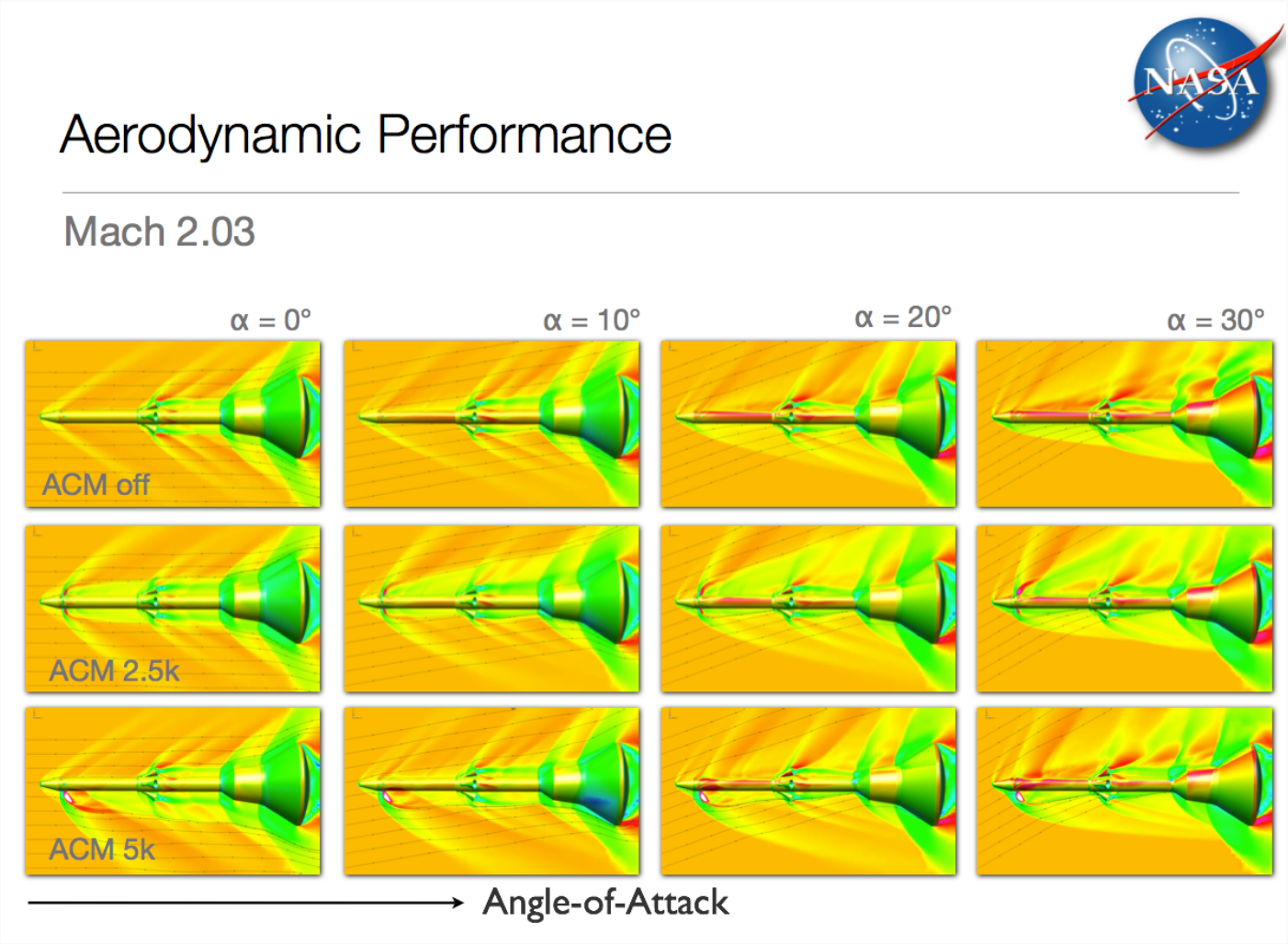


(b) Minimum Residual State

Adjoint linearized about states in limit cycle oscillations

Motivation: Non-Zero Residual?

If small scale unsteadiness leads to that sort of challenge, imagine more complex geometries with large scale unsteadiness.



Motivation: Krakos and Darmofal 2012

- Steady-state problems mimicking physical unsteadiness can lead to non-physical results.

Motivation: Krakos and Darmofal 2012

- Steady-state problems mimicking physical unsteadiness can lead to non-physical results.
- The adjoint system is difficult to solve and the results are sensitive to when the simulation is terminated.

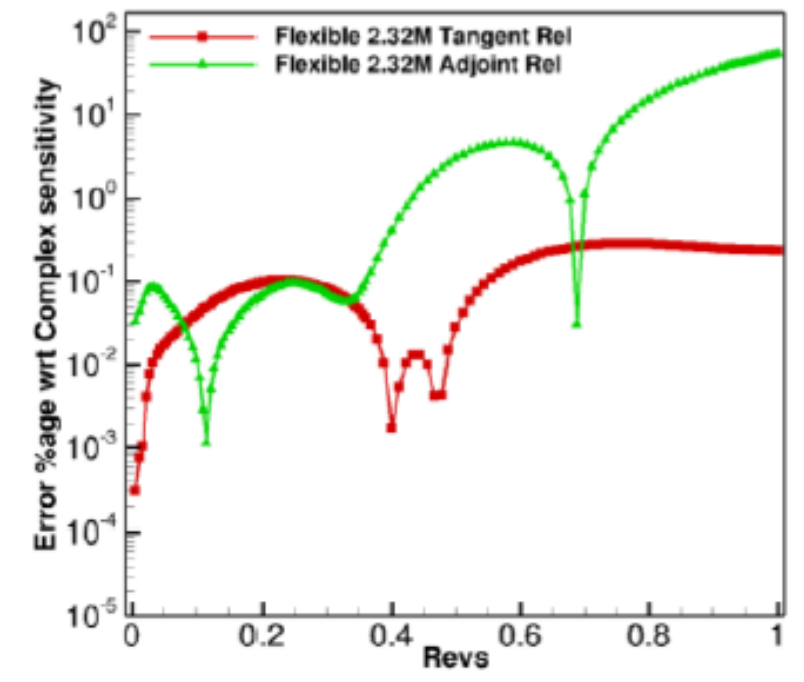
Motivation: Krakos and Darmofal 2012

- Steady-state problems mimicking physical unsteadiness can lead to non-physical results.
- The adjoint system is difficult to solve and the results are sensitive to when the simulation is terminated.
    - This is an issue, as when a simulation mimics the physical unsteadiness by entering limit-cycle oscillations, no state is more valid than another.

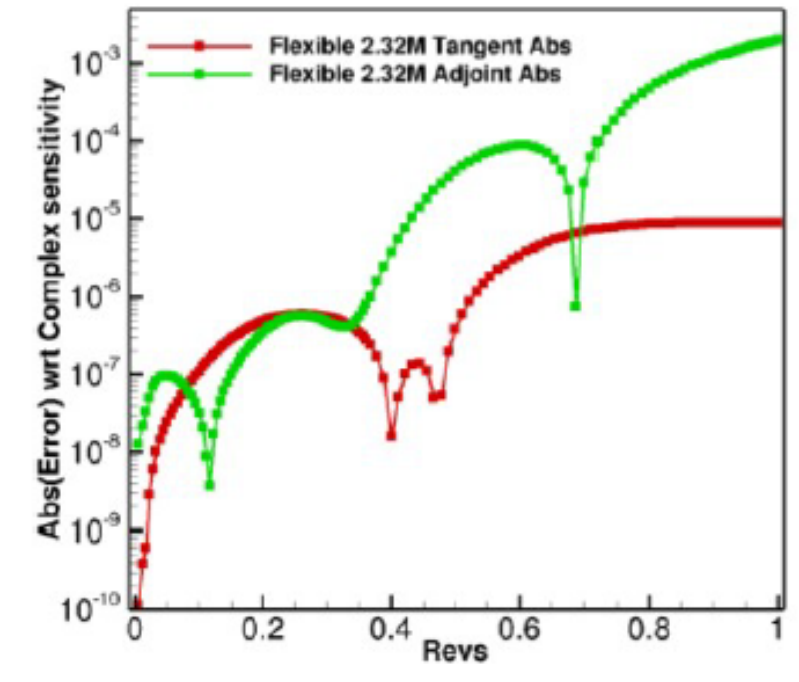
Motivation: Krakos and Darmofal 2012

- Steady-state problems mimicking physical unsteadiness can lead to non-physical results.
- The adjoint system is difficult to solve and the results are sensitive to when the simulation is terminated.
  - This is an issue, as when a simulation mimics the physical unsteadiness by entering limit-cycle oscillations, no state is more valid than another.
- The time accurate approach is the proper way to view these problems as the steady-state converged problem is unphysical and provides very different sensitivities than the time-accurate averaged case.

Motivation: Mishra et al. 2015



(a) Relative Error



(b) Absolute Error

Complex vs. adjoint sensitivities for partially converged unsteady rotorcraft flows

Mishra et al. show that for time-dependent rotorcraft optimization, partial convergence of the implicit system at each time-step leads to growing adjoint sensitivity error.

Motivation: Mishra et al. 2015

When taken in combination with Krakos and Darmofal, this shows that a time accurate approach is appropriate, but when we move to the time accurate approach, the cost of deep convergence is generally prohibitive.

Motivation: Luers et al. 2018

Steady-state volumetric optimization of a CRESCENDO turbine with objective function and constraint:

Motivation: Luers et al. 2018

Steady-state volumetric optimization of a CRESCENDO turbine with objective function and constraint:

- Simulation is run through a four order of magnitude residual decrease, and seemingly converged objective function and constraint values.

Motivation: Luers et al. 2018

Steady-state volumetric optimization of a CRESCENDO turbine with objective function and constraint:

- Simulation is run through a four order of magnitude residual decrease, and seemingly converged objective function and constraint values.
- They show close qualitative agreement between adjoint and finite-difference computed sensitivities.

Motivation: Luers et al. 2018

Steady-state volumetric optimization of a CRESCENDO turbine with objective function and constraint:

- Simulation is run through a four order of magnitude residual decrease, and seemingly converged objective function and constraint values.
- They show close qualitative agreement between adjoint and finite-difference computed sensitivities.
- They obtain a better final design from finite-difference computed sensitivities than the adjoint-computed ones.

Approach: Pseudo-Time Accurate Approach

- We employ a pseudo-time accurate approach that applies the unsteady adjoint to the steady state problem.

Approach: Pseudo-Time Accurate Approach

- We employ a pseudo-time accurate approach that applies the unsteady adjoint to the steady state problem.
- This can also be seen as an extension of the discrete adjoint approach (discretize, solve, linearize solution process).

Approach: Pseudo-Time Accurate Approach

- We employ a pseudo-time accurate approach that applies the unsteady adjoint to the steady state problem.
- This can also be seen as an extension of the discrete adjoint approach (discretize, solve, linearize solution process).
- In the tangent problem we take the derivative of every step used to solve the analysis to march forward in pseudo-time.

Approach: Pseudo-Time Accurate Approach

- We employ a pseudo-time accurate approach that applies the unsteady adjoint to the steady state problem.
- This can also be seen as an extension of the discrete adjoint approach (discretize, solve, linearize solution process).
- In the tangent problem we take the derivative of every step used to solve the analysis to march forward in pseudo-time.
- In the adjoint problem, we transpose those derivatives to march back in pseudo-time.

Approach: Pseudo-Time Accurate Approach

- We employ a pseudo-time accurate approach that applies the unsteady adjoint to the steady state problem.
- This can also be seen as an extension of the discrete adjoint approach (discretize, solve, linearize solution process).
- In the tangent problem we take the derivative of every step used to solve the analysis to march forward in pseudo-time.
- In the adjoint problem, we transpose those derivatives to march back in pseudo-time.
- This approach will correspond exactly to the complex-step finite-difference computed sensitivities and is developed for use in unconverged flows.

Approach: Pseudo-Time Accurate Approach

- We employ a pseudo-time accurate approach that applies the unsteady adjoint to the steady state problem.
- This can also be seen as an extension of the discrete adjoint approach (discretize, solve, linearize solution process).
- In the tangent problem we take the derivative of every step used to solve the analysis to march forward in pseudo-time.
- In the adjoint problem, we transpose those derivatives to march back in pseudo-time.
- This approach will correspond exactly to the complex-step finite-difference computed sensitivities and is developed for use in unconverged flows.
    - This assumes that the complex-step finite-difference computed sensitivities, which are the sensitivities of the solution process, are the appropriate sensitivities to use for optimization.

Code Overview/Capabilities

- steady-state Euler code
- second-order spatially accurate using gradient reconstruction
- van-leer flux splitting used predominantly
- Venkatakrishnan (VK) limiter with stagnation fix and temporary first order realizability check
- explicit and implicit solvers
- GMRES or BiCGStab Krylov linear solvers preconditioned with point implicit Jacobi or Gauss-Seidel
- line-search and CFL controller to assist implicit solver
- tangent, discrete adjoint and complex sensitivities for steady state sensitivity computation
- pseudo-time accurate tangent and adjoint capabilities for all solver schemes

Derivation for PTA Tangent and Adjoint Formulation

Derivation for PTA Tangent and Adjoint Formulation

- PTA tangent and adjoint derivations for a general fixed point iteration

Derivation for PTA Tangent and Adjoint Formulation

- PTA tangent and adjoint derivations for a general fixed point iteration
- PTA tangent derivation and adjoint formulation for forward Euler (simplest explicit solver)

Derivation for PTA Tangent and Adjoint Formulation

- PTA tangent and adjoint derivations for a general fixed point iteration
- PTA tangent derivation and adjoint formulation for forward Euler (simplest explicit solver)
- PTA tangent derivation and adjoint formulation for inexact-quasi-Newton method (workhorse implicit solver)

PTA Tangent System for General Fixed-Point Iteration

For a general fixed point iteration:

$$u^{k+1} = u^k + H(u^k(D), D) \tag{15}$$

the tangent sensitivity equation is:

$$\frac{du^{k+1}}{dD} = \frac{du^k}{dD} + \frac{dH}{dD} \tag{16}$$

PTA Tangent System for General Fixed-Point Iteration

For an objective functional dependent on the last m states for a simulation that runs n time steps:

$$L = L(u^n, u^{n-1}, ..., u^{n-m}, D) \tag{17}$$

PTA Tangent System for General Fixed-Point Iteration

For an objective functional dependent on the last m states for a simulation that runs n time steps:

$$L = L(u^n, u^{n-1}, ..., u^{n-m}, D) \tag{17}$$

the sensitivity equation is defined as below.

$$\frac{dL}{dD} = \frac{\partial L}{\partial x}\frac{dx}{dD} + \frac{\partial L}{\partial u^n}\frac{du^n}{dD} + \frac{\partial L}{\partial u^{n-1}}\frac{du^{n-1}}{dD} + ... + \frac{\partial L}{\partial u^{n-m}}\frac{du^{n-m}}{dD} \tag{18}$$

We can substitute the pseudo-time accurate tangent provided values for $\frac{du^k}{dD}$.

PTA Adjoint System for General Fixed-Point Iteration

Here we begin with an objective functional dependent on the last m states for a simulation that runs n time steps:

$$L = L(u^n, u^{n-1}, ..., u^{n-m}, D) \tag{19}$$

PTA Adjoint System for General Fixed-Point Iteration

Here we begin with an objective functional dependent on the last m states for a simulation that runs n time steps:

$$L = L(u^n, u^{n-1}, ..., u^{n-m}, D) \tag{19}$$

with the pseudo-time constraint resulting from shifting the fixed-point iteration by its output.

$$G^k(u^k(D), u^{k-1}(D), D) = u^k - u^{k-1} - H(u^{k-1}(D), D) = 0 \tag{20}$$

PTA Adjoint System for General Fixed-Point Iteration

Here we begin with an objective functional dependent on the last m states for a simulation that runs n time steps:

$$L = L(u^n, u^{n-1}, ..., u^{n-m}, D) \tag{19}$$

with the pseudo-time constraint resulting from shifting the fixed-point iteration by its output.

$$G^k(u^k(D), u^{k-1}(D), D) = u^k - u^{k-1} - H(u^{k-1}(D), D) = 0 \tag{20}$$

We then form our augmented objective functional with the above constraint and a corresponding adjoint for each pseudo-time step.

$$\begin{aligned} J(D, u^n, u^{n-1}, ..., \Lambda^n, \Lambda^{n-1}, ...) = {} & L(u^n, u^{n-1}, ..., u^{n-m}, D) \\ & + \Lambda^{nT} G^n(u^n(D), u^{n-1}(D), D) \\ & + \Lambda^{n-1T} G^{n-1}(u^{n-1}(D), u^{n-2}(D), D) \\ & + ... \\ & + \Lambda^{1T} G^1(u^1(D), u^0(D), D) \end{aligned} \tag{21}$$

PTA Adjoint System for General Fixed-Point Iteration

Taking the derivative of the augmented objective functional yields:

$$\begin{aligned}\frac{dJ}{dD} = &\frac{\partial L}{\partial x}\frac{dx}{dD} + \frac{\partial L}{\partial u^n}\frac{du^n}{dD} + \frac{\partial L}{\partial u^{n-1}}\frac{du^{n-1}}{dD} + \ldots + \frac{\partial L}{\partial u^{n-m}}\frac{du^{n-m}}{dD} \\ &+ \Lambda^{nT}\left(\frac{\partial G^n}{\partial x}\frac{dx}{dD} + \frac{\partial G^n}{\partial u^n}\frac{du^n}{dD} + \frac{\partial G^n}{\partial u^{n-1}}\frac{du^{n-1}}{dD}\right) \\ &+ \Lambda^{n-1T}\left(\frac{\partial G^{n-1}}{\partial x}\frac{dx}{dD} + \frac{\partial G^{n-1}}{\partial u^{n-1}}\frac{du^{n-1}}{dD} + \frac{\partial G^{n-1}}{\partial u^{n-2}}\frac{du^{n-2}}{dD}\right) \\ &+ \ldots \\ &+ \Lambda^{1T}\left(\frac{\partial G^1}{\partial x}\frac{dx}{dD} + \frac{\partial G^1}{\partial u^1}\frac{du^1}{dD} + \frac{\partial G^1}{\partial u^0}\frac{du^0}{dD}\right)\end{aligned} \tag{22}$$

PTA Adjoint System for General Fixed-Point Iteration

As in the steady-state adjoint we choose the adjoint variable such that we do not have to calculate $\frac{du^k}{dD}$, which returns a series of adjoint recurrence relations:

$$\frac{\partial L}{\partial u^{k-1}} + \Lambda^{kT} \frac{\partial G^k}{\partial u^{k-1}} + \Lambda^{k-1T} \frac{\partial G^{k-1}}{\partial u^{k-1}} = 0 \tag{23}$$

with $\Lambda^{n+1T} = 0$.

PTA Adjoint System for General Fixed-Point Iteration

As in the steady-state adjoint we choose the adjoint variable such that we do not have to calculate $\frac{du^k}{dD}$, which returns a series of adjoint recurrence relations:

$$\frac{\partial L}{\partial u^{k-1}} + \Lambda^{kT} \frac{\partial G^k}{\partial u^{k-1}} + \Lambda^{k-1T} \frac{\partial G^{k-1}}{\partial u^{k-1}} = 0 \tag{23}$$

with $\Lambda^{n+1T} = 0$. The recurrence relation marches back in pseudo-time:

$$\Lambda^{k-1T} \frac{\partial G^{k-1}}{\partial u^{k-1}} = \Lambda^{kT} \frac{\partial G^k}{\partial u^{k-1}} - \frac{\partial L}{\partial u^{k-1}} \tag{24}$$

PTA Adjoint System for General Fixed-Point Iteration

As in the steady-state adjoint we choose the adjoint variable such that we do not have to calculate $\frac{du^k}{dD}$, which returns a series of adjoint recurrence relations:

$$\frac{\partial L}{\partial u^{k-1}} + \Lambda^{kT}\frac{\partial G^k}{\partial u^{k-1}} + \Lambda^{k-1T}\frac{\partial G^{k-1}}{\partial u^{k-1}} = 0 \tag{23}$$

with $\Lambda^{n+1T} = 0$. The recurrence relation marches back in pseudo-time:

$$\Lambda^{k-1T}\frac{\partial G^{k-1}}{\partial u^{k-1}} = \Lambda^{kT}\frac{\partial G^k}{\partial u^{k-1}} - \frac{\partial L}{\partial u^{k-1}} \tag{24}$$

which returns the below sensitivity equation.

$$\frac{dJ}{dD} = \frac{\partial L}{\partial D} + \Lambda^{nT}\frac{\partial G^n}{\partial D} + \Lambda^{n-1T}\frac{\partial G^{n-1}}{\partial D} + \Lambda^{n-2T}\frac{\partial G^{n-2}}{\partial D} + \ldots \tag{25}$$

PTA Tangent System for Forward Euler (Derivation)

We look at our pseudo-time evolution equation:

$$H(u^k, D) = CFL\Delta tR \tag{26}$$

where CFL is a parameter and Δt is a local time step.

PTA Tangent System for Forward Euler (Derivation)

$$u^k = u^{k-1} + CFL\Delta t(u^{k-1}(D), x(D))R(u^{k-1}(D), x(D)) \qquad (27)$$

PTA Tangent System for Forward Euler (Derivation)

$$u^k = u^{k-1} + CFL\Delta t(u^{k-1}(D), x(D))R(u^{k-1}(D), x(D)) \tag{27}$$

We take the derivative and obtain a pseudo-time evolution equation:

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

By running this relation through pseudo-time we can obtain the exact sensitivities at every pseudo-time step.

PTA Adjoint System for Forward Euler

Adjoint and sensitivity increment equations for forward Euler:

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

PTA Adjoint System for Forward Euler

Adjoint and sensitivity increment equations for forward Euler:

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

$$\Delta\Lambda^{k-1} = -\left[CFL\Delta t^{k-1}\left[\frac{\partial R(u^{k-1})}{\partial u^{k-1}}\right]_2 + CFL\frac{\partial \Delta t}{\partial u^{k-1}}R\right]^T \Lambda^k - \frac{\partial L}{\partial u^{k-1}} \tag{30}$$

PTA Adjoint System for Forward Euler

Adjoint and sensitivity increment equations for forward Euler:

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

$$\Delta\Lambda^{k-1} = -\left[CFL\Delta t^{k-1}\left[\frac{\partial R(u^{k-1})}{\partial u^{k-1}}\right]_2 + CFL\frac{\partial \Delta t}{\partial u^{k-1}}R\right]^T \Lambda^k - \frac{\partial L}{\partial u^{k-1}} \tag{30}$$

$$\Delta\frac{dJ}{dD}^k = \Lambda^{kT}\left[\frac{\partial H}{\partial D}\right] - \frac{\partial L}{\partial D}^{k-1} \tag{31}$$

PTA Adjoint System for Forward Euler

Adjoint and sensitivity increment equations for forward Euler:

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

$$\Delta\Lambda^{k-1} = -\left[CFL\Delta t^{k-1}\left[\frac{\partial R(u^{k-1})}{\partial u^{k-1}}\right]_2 + CFL\frac{\partial \Delta t}{\partial u^{k-1}}R\right]^T \Lambda^k - \frac{\partial L}{\partial u^{k-1}} \tag{30}$$

$$\Delta\frac{dJ}{dD}^k = \Lambda^{kT}\left[\frac{\partial H}{\partial D}\right] - \frac{\partial L}{\partial D}^{k-1} \tag{31}$$

$$\Delta\frac{dJ}{dD}^k = -\Lambda^{kT}\left[CFL\Delta t^{k-1}\frac{\partial R(u^{k-1})}{\partial D} + CFL\frac{\partial \Delta t^{k-1}}{\partial D}R\right] - \frac{\partial L}{\partial D}^{k-1} \tag{32}$$

Nonlinear Solver: Newton's Method

To solve the nonlinear problem we use a quasi-Newton method using pseudo-transient continuation (PTC) in a BDF1 pseudo temporal discretization scheme. The pseudo-time stepping procedure being:

$$u^k = u^{k-1} + \Delta u^{k-1} \tag{33}$$

where Δu is calculated by solving the linear system below.

$$[P_{k-1}]\,\Delta u^{k-1} = R(u^{k-1}(D), X(D)) \tag{34}$$

Nonlinear Solver: Newton's Method

The preconditioner matrix P_k is:

$$P_k = \left[\frac{\partial R}{\partial u}\right]_1 + M \tag{35}$$

and $\left[\frac{\partial R}{\partial u}\right]_1$ is an first order approximation to the Jacobian and M is a suitable mass matrix.

$$M = \frac{vol}{CFL\Delta t} \tag{36}$$

Nonlinear Solver: Newton's Method

The preconditioner matrix P_k is:

$$P_k = \left[\frac{\partial R}{\partial u}\right]_1 + M \tag{35}$$

and $\left[\frac{\partial R}{\partial u}\right]_1$ is an first order approximation to the Jacobian and M is a suitable mass matrix.

$$M = \frac{vol}{CFL\Delta t} \tag{36}$$

- Δt is the local time step.

Nonlinear Solver: Newton's Method

The preconditioner matrix P_k is:

$$P_k = \left[\frac{\partial R}{\partial u}\right]_1 + M \tag{35}$$

and $\left[\frac{\partial R}{\partial u}\right]_1$ is an first order approximation to the Jacobian and M is a suitable mass matrix.

$$M = \frac{vol}{CFL\Delta t} \tag{36}$$

- Δt is the local time step.
- We have added a combined line-search and CFL controller to assist with stability and non-linear convergence.

PTA Tangent system for Newton's Method

Taking the derivative of the nonlinear solver gives the following expression.

$$\frac{du^k}{dD} = \frac{du^{k-1}}{dD} + \frac{d\left[P_{k-1}\right]^{-1} R(u^{k-1}, D)}{dD} \tag{37}$$

PTA Tangent system for Newton's Method

Taking the derivative of the nonlinear solver gives the following expression.

$$\frac{du^k}{dD} = \frac{du^{k-1}}{dD} + \frac{d\left[P_{k-1}\right]^{-1} R(u^{k-1}, D)}{dD} \tag{37}$$

If you expand the derivative:

$$\begin{aligned}\frac{du^k}{dD} = \frac{du^{k-1}}{dD} &+ \left[P_{k-1}\right]^{-1}\left[\frac{dR(u^{k-1}, D)}{dD}\right] \\ &+ \frac{d\left[P_{k-1}\right]^{-1}}{dD} R(u^{k-1}, D)\end{aligned} \tag{38}$$

PTA Tangent system for Newton's Method

- Fully linearizing the matrix inverse is not feasible, especially for more complicated linear solvers, e.g. GMRES.

PTA Tangent system for Newton's Method

- Fully linearizing the matrix inverse is not feasible, especially for more complicated linear solvers, e.g. GMRES.
- There are two ways to tackle the linearization of the matrix inverse.

PTA Tangent system for Newton's Method

- Fully linearizing the matrix inverse is not feasible, especially for more complicated linear solvers, e.g. GMRES.
- There are two ways to tackle the linearization of the matrix inverse.
  1. Use complex perturbations in the conservative variable and the nodal coordinate vectors to obtain the Frechét derivative of the linear system solve.

PTA Tangent system for Newton's Method

- Fully linearizing the matrix inverse is not feasible, especially for more complicated linear solvers, e.g. GMRES.
- There are two ways to tackle the linearization of the matrix inverse.
  1. Use complex perturbations in the conservative variable and the nodal coordinate vectors to obtain the Frechét derivative of the linear system solve.
  2. Identity for the derivative of the matrix inverse, this will not be exact except in the limit of machine zero linear system tolerance, but it is less expensive and simpler: $\frac{dA^{-1}}{dx} = -A^{-1}\frac{dA}{dx}A^{-1}$.

PTA Tangent system for Newton's Method

Starting from the nonlinear solver:

$$u^k = u^{k-1} + \Delta u = u^{k-1} + [P_{k-1}]^{-1} R \tag{39}$$

PTA Tangent system for Newton's Method

Starting from the nonlinear solver:

$$u^k = u^{k-1} + \Delta u = u^{k-1} + [P_{k-1}]^{-1} R \tag{39}$$

We can take the derivative and substitute in with the inverse identity:

$$\begin{aligned} \frac{du^k}{dD} = \frac{du^{k-1}}{dD} + [P_{k-1}]^{-1} \left[\left[\frac{\partial R(u^{k-1})}{\partial u^{k-1}} \right]_2 \frac{du^{k-1}}{dD} + \left[\frac{\partial R}{\partial x} \right] \frac{dx}{dD} \right] \\ - [P_{k-1}]^{-1} \frac{d\,[P_{k-1}]}{dD} [P_{k-1}]^{-1} R(u^{k-1}, D) \end{aligned} \tag{40}$$

PTA Tangent system for Newton's Method

Starting from the nonlinear solver:

$$u^k = u^{k-1} + \Delta u = u^{k-1} + [P_{k-1}]^{-1} R \tag{39}$$

We can take the derivative and substitute in with the inverse identity:

$$\begin{aligned} \frac{du^k}{dD} = \frac{du^{k-1}}{dD} &+ [P_{k-1}]^{-1} \left[\left[\frac{\partial R(u^{k-1})}{\partial u^{k-1}} \right]_2 \frac{du^{k-1}}{dD} + \left[\frac{\partial R}{\partial x} \right] \frac{dx}{dD} \right] \\ &- [P_{k-1}]^{-1} \left[\frac{\partial P_{k-1}}{\partial u^{k-1}} \frac{du^{k-1}}{dD} + \frac{\partial P_{k-1}}{\partial x} \frac{dx}{dD} \right] \Delta u \end{aligned} \tag{40}$$

PTA Tangent system for Newton's Method

Pseudo-time evolution equation of conservative variable sensitivities:

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

PTA Tangent system for Newton's Method

Pseudo-time evolution equation of conservative variable sensitivities:

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

$$\begin{aligned}[P_{k-1}]\Delta\left(\frac{du^{k-1}}{dD}\right) = &\left[\left[\frac{\partial R}{\partial u^{k-1}}\right]_2 \frac{du^{k-1}}{dD} + \left[\frac{\partial R}{\partial x}\right]\frac{dx}{dD}\right] \\ &- \left[\frac{\partial P_{k-1}}{\partial u^{k-1}}\frac{du^{k-1}}{dD} + \frac{\partial P_{k-1}}{\partial x}\frac{dx}{dD}\right]\Delta u\end{aligned} \tag{42}$$

PTA Adjoint System for Newton's Method

Define a secondary adjoint variable:

$$[P_{k-1}]^T \psi^k = \Lambda^k \tag{43}$$

PTA Adjoint System for Newton's Method

Define a secondary adjoint variable:

$$[P_{k-1}]^T \psi^k = \Lambda^k \tag{43}$$

Adjoint and sensitivity increment equations:

$$\Delta\Lambda^{k-1} = \left[\left[\frac{\partial R(u^{k-1})}{\partial u^{k-1}} \right]_2 - \frac{\partial P_{k-1}}{\partial u^{k-1}} \Delta u \right]^T \psi^k - \left[\frac{\partial L}{\partial u^{k-1}} \right]^T \tag{44}$$

$$\Delta \frac{dJ}{dD}^k = -\psi^{kT} \left[\frac{\partial R(u^{k-1})}{\partial D} - \frac{\partial [P_{k-1}]}{\partial D} \Delta u^{k-1} \right] - \frac{\partial L}{\partial D}^{k-1} \tag{45}$$

Notes on the PTA Formulations for Newton's Method

Some things to note:

- The linear system solved for each tangent or adjoint system has the same (or transposed) LHS as in the analysis simulation at the given non-linear iteration.

Notes on the PTA Formulations for Newton's Method

Some things to note:

- The linear system solved for each tangent or adjoint system has the same (or transposed) LHS as in the analysis simulation at the given non-linear iteration.
  - Therefore, if the tangent uses the same iterative solver as the analysis and the adjoint uses the dual of the analysis solver we can guarantee convergence.

Notes on the PTA Formulations for Newton's Method

Some things to note:

- The linear system solved for each tangent or adjoint system has the same (or transposed) LHS as in the analysis simulation at the given non-linear iteration.
  - Therefore, if the tangent uses the same iterative solver as the analysis and the adjoint uses the dual of the analysis solver we can guarantee convergence.
- This formulation requires the second order accurate Jacobian, but depending on choice of non-linear solver won't require the approximate inversion of that matrix.

Notes on the PTA Formulations for Newton's Method

Some things to note:

- The linear system solved for each tangent or adjoint system has the same (or transposed) LHS as in the analysis simulation at the given non-linear iteration.
  - Therefore, if the tangent uses the same iterative solver as the analysis and the adjoint uses the dual of the analysis solver we can guarantee convergence.
- This formulation requires the second order accurate Jacobian, but depending on choice of non-linear solver won't require the approximate inversion of that matrix.
- The right hand side has derivatives of the preconditioner matrix, but in the context of matrix vector products. In this work they are computed with complex Frechét differentiation.

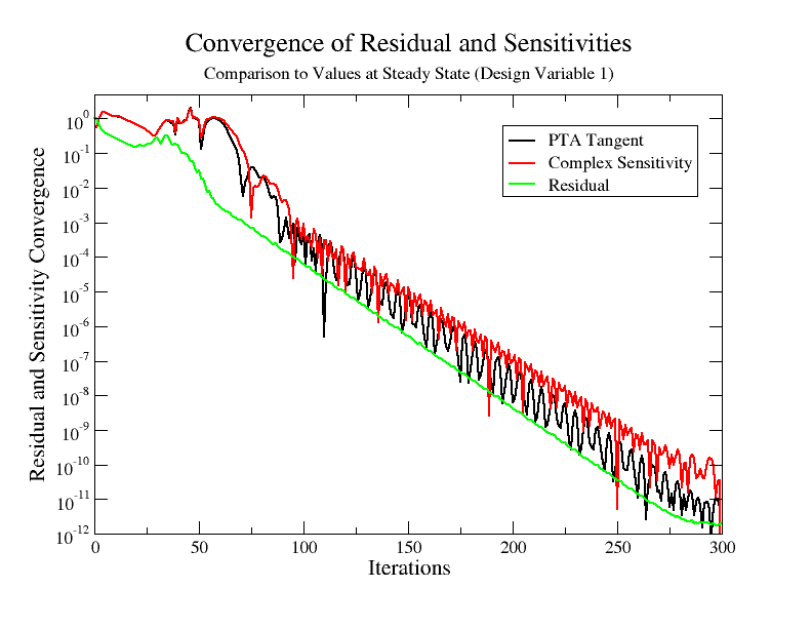
Notes on the PTA Formulations for Newton's Method

Some things to note:

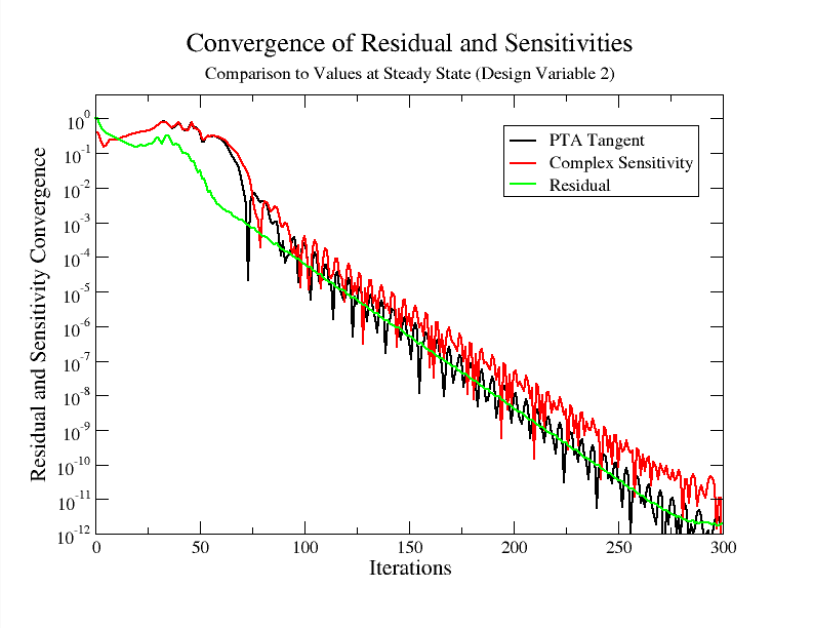
- The linear system solved for each tangent or adjoint system has the same (or transposed) LHS as in the analysis simulation at the given non-linear iteration.
  - Therefore, if the tangent uses the same iterative solver as the analysis and the adjoint uses the dual of the analysis solver we can guarantee convergence.
- This formulation requires the second order accurate Jacobian, but depending on choice of non-linear solver won't require the approximate inversion of that matrix.
- The right hand side has derivatives of the preconditioner matrix, but in the context of matrix vector products. In this work they are computed with complex Frechét differentiation.
- These formulations are exact for machine-zero convergence of the linear system.

Error Due to Approximate Linearization of the Fixed-Point

Sensitivity Convergence for Inexact Newton: 1e-1



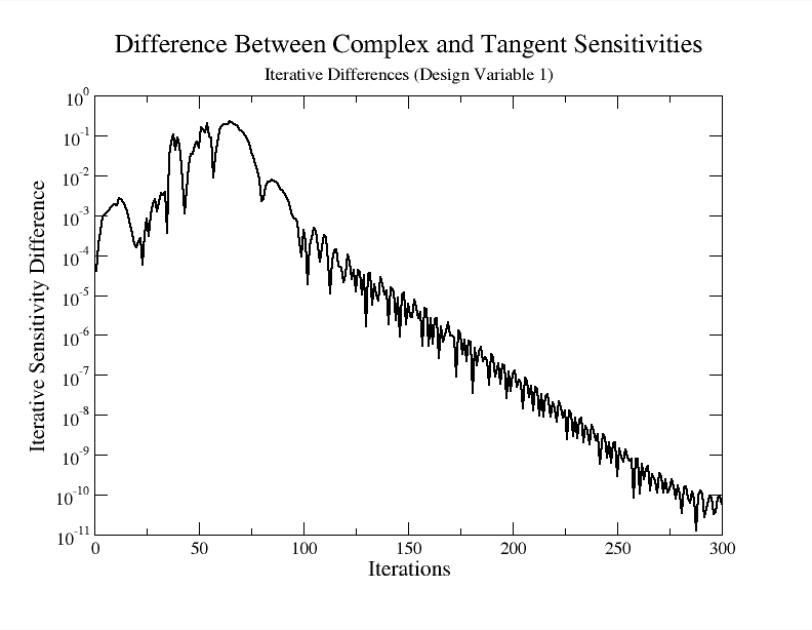
(a) Design Variable 1



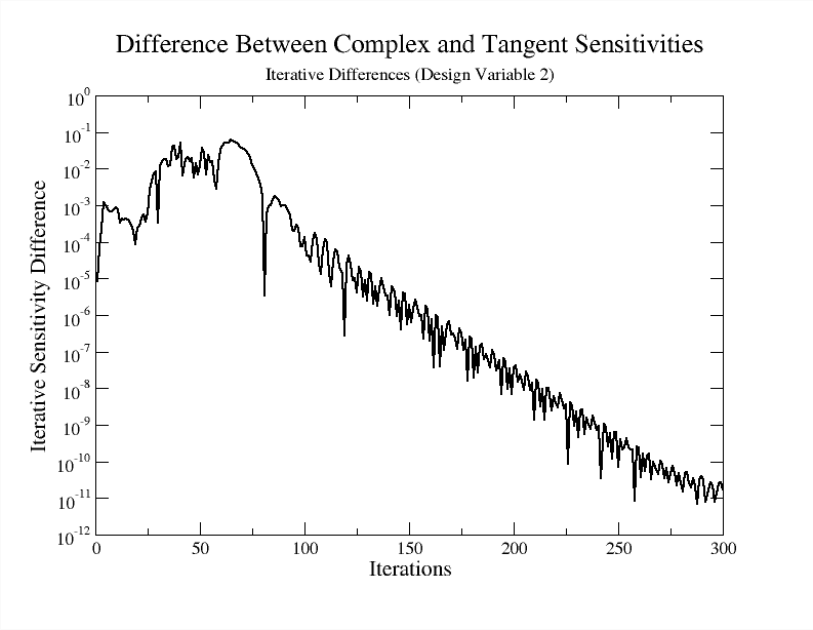
(b) Design Variable 2

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

Sensitivity Difference for Inexact Newton: 1e-1



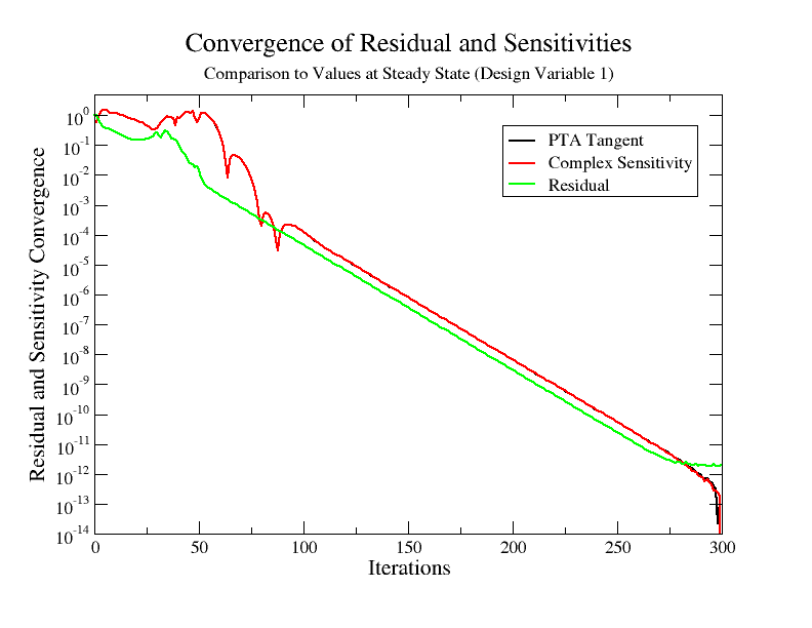
(a) Design Variable 1



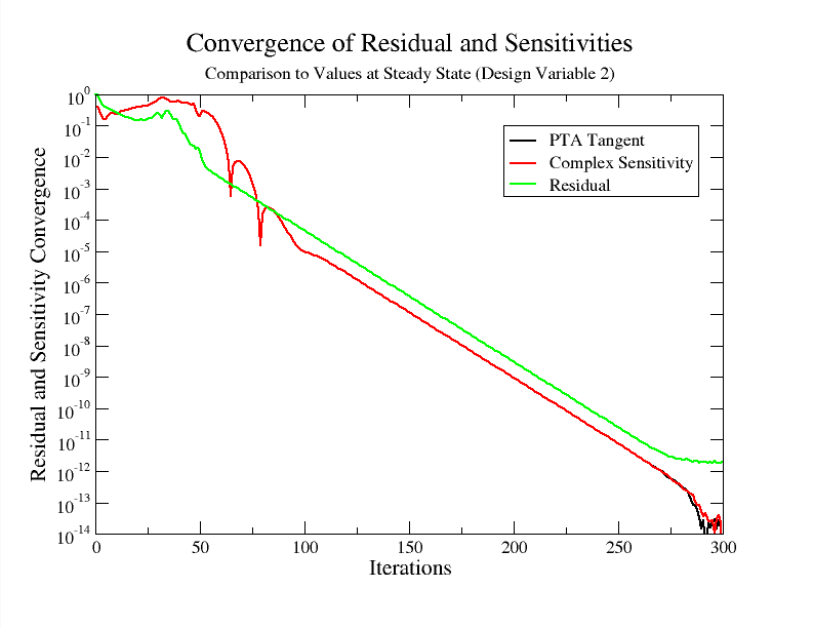
(b) Design Variable 2

Iterative sensitivity difference for linear tolerance, $1e-1$

Sensitivity Convergence for Inexact Newton: 1e-4



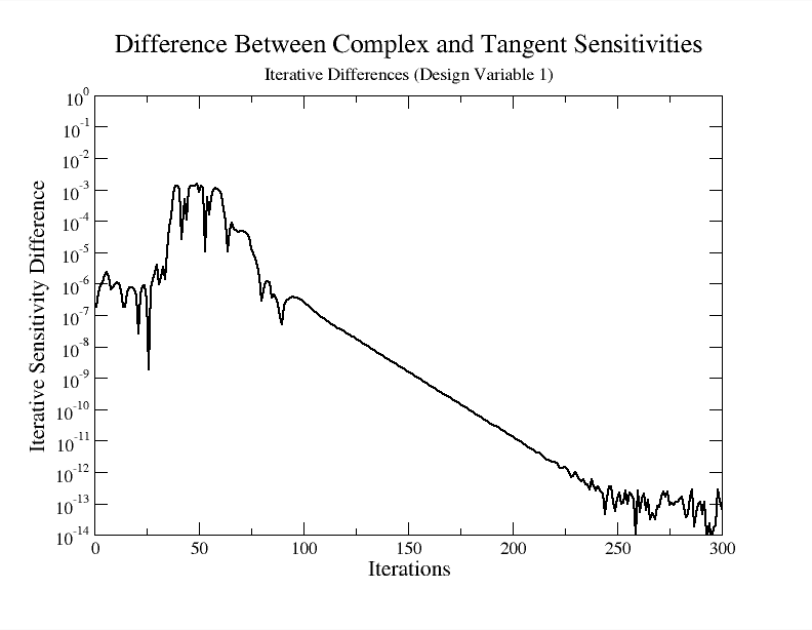
(a) Design Variable 1



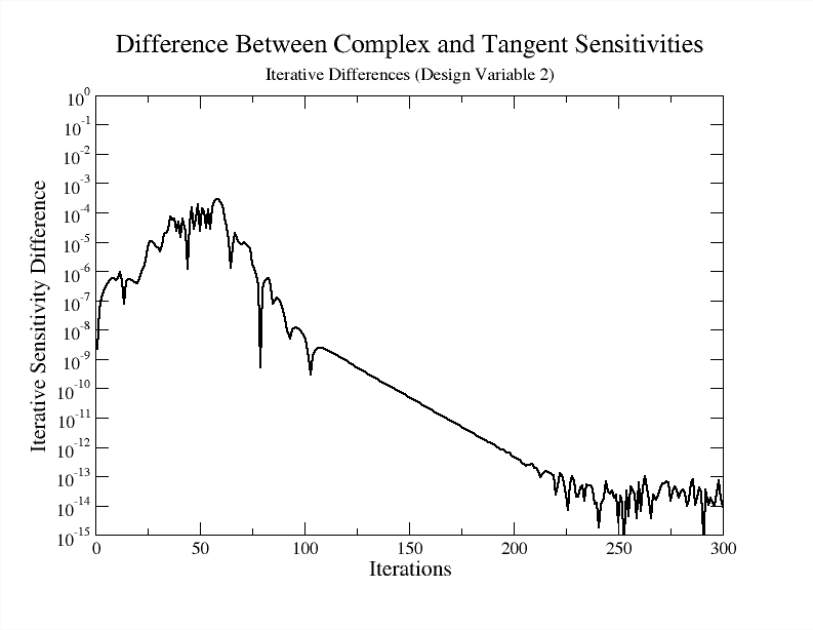
(b) Design Variable 2

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

Sensitivity Difference for Inexact Newton: 1e-4



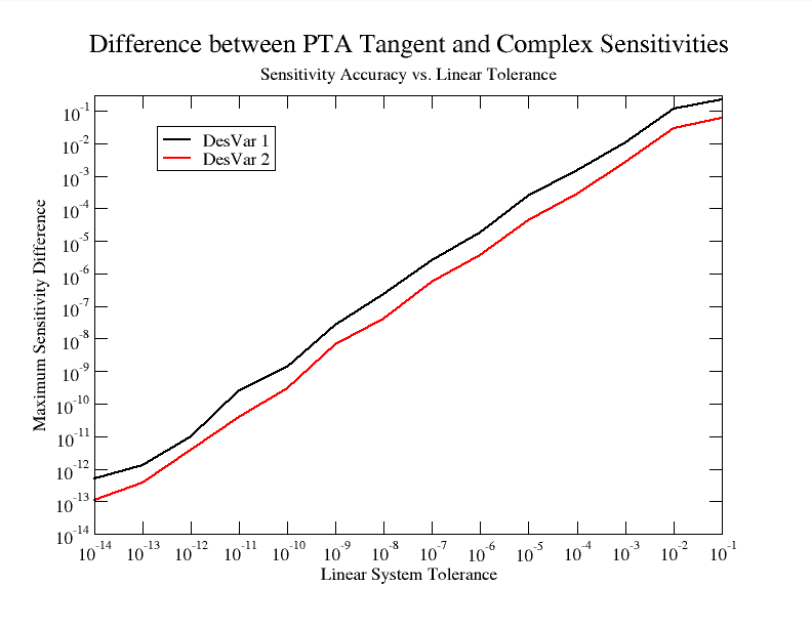
(a) Design Variable 1



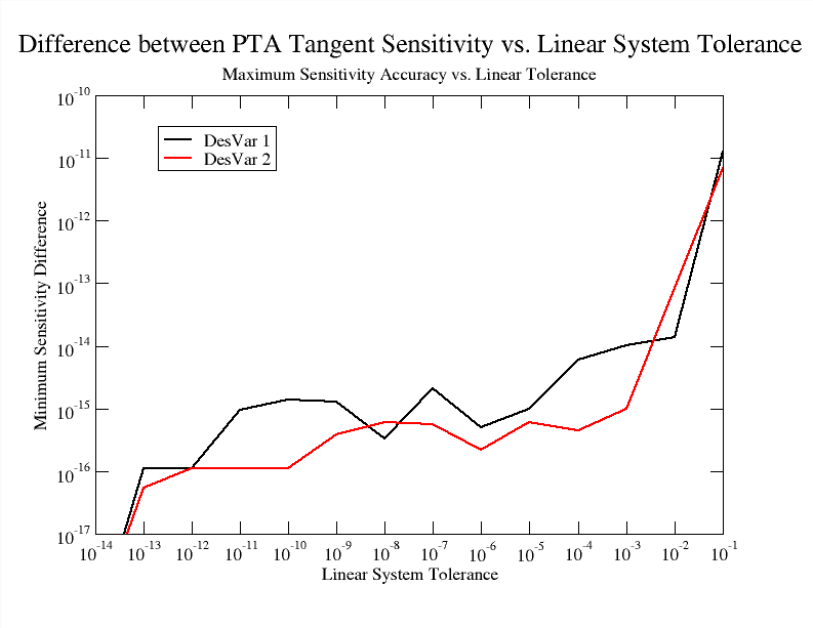
(b) Design Variable 2

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

Max Sensitivity Difference for Inexact Newton



(a) Maximum Iterative Difference



(b) Minimum Iterative Difference

Iterative difference vs. linear tolerance

Error Due to Approximate Linearization of the Fixed-Point

For a general fixed point iteration to solve the equation $R = 0$:

$$u^{k+1} = N(u^k, D) = u^k + H(u^k, D) = u^k + A(u^k, D)R(u^k, D) \qquad (46)$$

Error Due to Approximate Linearization of the Fixed-Point

For a general fixed point iteration to solve the equation $R = 0$:

$$u^{k+1} = N(u^k, D) = u^k + H(u^k, D) = u^k + A(u^k, D)R(u^k, D) \tag{46}$$

The exact linearization of the solution process:

$$\frac{du^{k+1}}{dD} = \frac{dN(u^k)}{dD} = \frac{du^k}{dD} + \frac{dA}{dD}R + A\frac{dR}{dD} \tag{47}$$

Error Due to Approximate Linearization of the Fixed-Point

For an inexact linearization of the fixed point, only A is inexactly linearized. The computed linearization of the fixed point iteration with the approximate linearization of A is below:

$$\widetilde{\frac{du^{k+1}}{dD}} = \widetilde{\frac{dN(u^k)}{dD}} = \widetilde{\frac{du^k}{dD}} + \widetilde{\frac{dA}{dD}} R + A\widetilde{\frac{dR}{dD}} \tag{48}$$

Error Due to Approximate Linearization of the Fixed-Point

For an inexact linearization of the fixed point, only A is inexactly linearized. The computed linearization of the fixed point iteration with the approximate linearization of A is below:

$$\widetilde{\frac{du^{k+1}}{dD}} = \widetilde{\frac{dN(u^k)}{dD}} = \widetilde{\frac{du^k}{dD}} + \widetilde{\frac{dA}{dD}}R + A\widetilde{\frac{dR}{dD}} \tag{48}$$

The error in the total derivative of the residual is due to the multiplication by an inexact linearization:

$$\widetilde{\frac{dR}{dD}} = \frac{\partial R}{\partial D} + \frac{\partial R}{\partial u}\widetilde{\frac{du}{dD}} \tag{49}$$

Error Due to Approximate Linearization of the Fixed-Point

To get the error in the conservative variable linearization it is necessary to subtract the two expressions from one another:

$$\frac{du^{k+1}}{dD} - \widetilde{\frac{du^{k+1}}{dD}} = \frac{du^{k}}{dD} - \widetilde{\frac{du^{k}}{dD}} + \frac{dA}{dD}R - \widetilde{\frac{dA}{dD}}R + A\frac{dR}{dD} - A\widetilde{\frac{dR}{dD}} \quad (50)$$

Error Due to Approximate Linearization of the Fixed-Point

$\frac{d\epsilon}{dD}$ is defined as the error in $\frac{du}{dD}$ and used to group like terms:

$$\frac{d\epsilon^{k+1}}{dD} = \frac{d\epsilon^{k}}{dD} + \left[\frac{dA}{dD} - \widetilde{\frac{dA}{dD}}\right] R + A \left[\frac{dR}{dD} - \widetilde{\frac{dR}{dD}}\right] \tag{51}$$

Error Due to Approximate Linearization of the Fixed-Point

By grouping like terms and rearranging we obtain this form of the expression for the error in the sensitivities:

$$\begin{aligned}\frac{d\epsilon^{k+1}}{dD} = &\left[\frac{d\epsilon^{k}}{dD} + A\left[\frac{\partial R}{\partial u^{k}}\frac{d\epsilon^{k}}{dD}\right] + \frac{\partial A}{\partial u^{k}}\frac{d\epsilon^{k}}{dD}R\right] \\ &+ \left[\left(\frac{\partial A}{\partial D} - \widetilde{\frac{\partial A}{\partial D}}\right) + \left(\frac{\partial A}{\partial u} - \widetilde{\frac{\partial A}{\partial u}}\right)\frac{du^{k}}{dD}\right]R\end{aligned} \tag{52}$$

Error Due to Approximate Linearization of the Fixed-Point

We know that the second bracketed term goes to zero as the residual does, but we don't know why this occurs for the first term, so we use the Cauchy-Schwarz inequality to return the inequality below.

$$\left\| \frac{d\epsilon^k}{dD} + A\frac{\partial R}{\partial u^k}\frac{d\epsilon^k}{dD} + \frac{\partial A}{\partial u^k}\frac{d\epsilon^k}{dD}R \right\| < \left\| I + A\frac{\partial R}{\partial u^k} + \frac{\partial A}{\partial u^k}R \right\| \left\| \frac{d\epsilon^k}{dD} \right\| \quad (53)$$

Error Due to Approximate Linearization of the Fixed-Point

We know that the second bracketed term goes to zero as the residual does, but we don't know why this occurs for the first term, so we use the Cauchy-Schwarz inequality to return the inequality below.

$$\left\| \frac{d\epsilon^k}{dD} + A\frac{\partial R}{\partial u^k}\frac{d\epsilon^k}{dD} + \frac{\partial A}{\partial u^k}\frac{d\epsilon^k}{dD}R \right\| < \left\| I + A\frac{\partial R}{\partial u^k} + \frac{\partial A}{\partial u^k}R \right\| \left\| \frac{d\epsilon^k}{dD} \right\| \quad (53)$$

We can define $B = I + A\frac{\partial R}{\partial u^k} + \frac{\partial A}{\partial u^k}R$ and we realize that $B = \frac{\partial N}{\partial u}$, since N is the contractive fixed point iteration, $\|B\| < 1$, and this explains why both terms go to zero as the nonlinear problem does

Error Due to Approximate Linearization of the Fixed-Point

We know that the second bracketed term goes to zero as the residual does, but we don't know why this occurs for the first term, so we use the Cauchy-Schwarz inequality to return the inequality below.

$$\left\| \frac{d\epsilon^k}{dD} + A\frac{\partial R}{\partial u^k}\frac{d\epsilon^k}{dD} + \frac{\partial A}{\partial u^k}\frac{d\epsilon^k}{dD}R \right\| < \left\| I + A\frac{\partial R}{\partial u^k} + \frac{\partial A}{\partial u^k}R \right\| \left\| \frac{d\epsilon^k}{dD} \right\| \quad (53)$$

We can define $B = I + A\frac{\partial R}{\partial u^k} + \frac{\partial A}{\partial u^k}R$ and we realize that $B = \frac{\partial N}{\partial u}$, since N is the contractive fixed point iteration, $\|B\| < 1$, and this explains why both terms go to zero as the nonlinear problem does

$$\left\| \frac{d\epsilon^{k+1}}{dD} \right\| < \|B\| \left\| \frac{d\epsilon^k}{dD} \right\| + \left\| \left[\frac{dA}{dD} - \widetilde{\frac{dA}{dD}} \right] R \right\| \quad (54)$$

Low Storage Explicit Runge-Kutta Formulation

$$u^{k,l} = u^{k,0} + CFL\alpha^{l-1}\Delta t R(u^{k,l-1}) \tag{55}$$

Low Storage Explicit Runge-Kutta Formulation

$$u^{k,l} = u^{k,0} + CFL\alpha^{l-1}\Delta t R(u^{k,l-1}) \tag{55}$$

$$u^{k+1} = u^{k} + A(u^{k}(D), D)R(u^{k}, D) \tag{56}$$

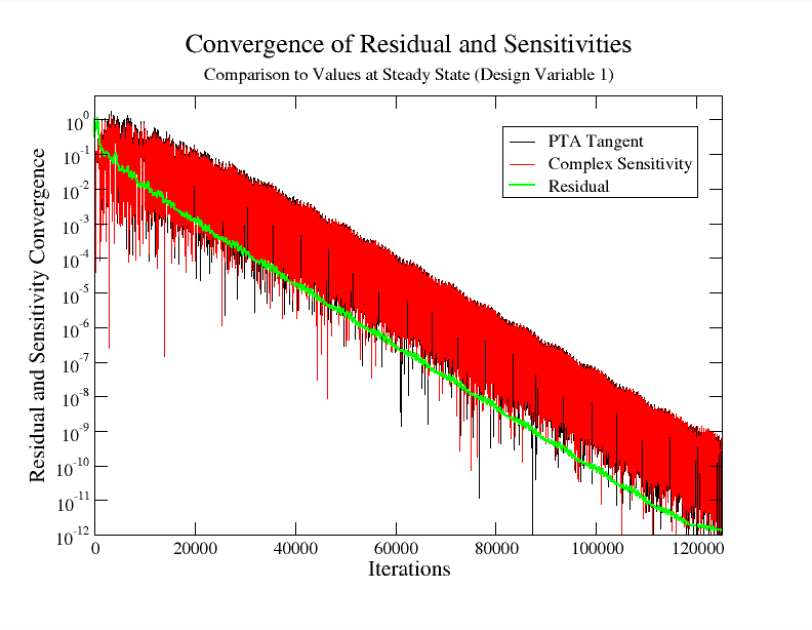
Low Storage Explicit Runge-Kutta Formulation

$$u^{k,l} = u^{k,0} + CFL\alpha^{l-1}\Delta t R(u^{k,l-1}) \tag{55}$$

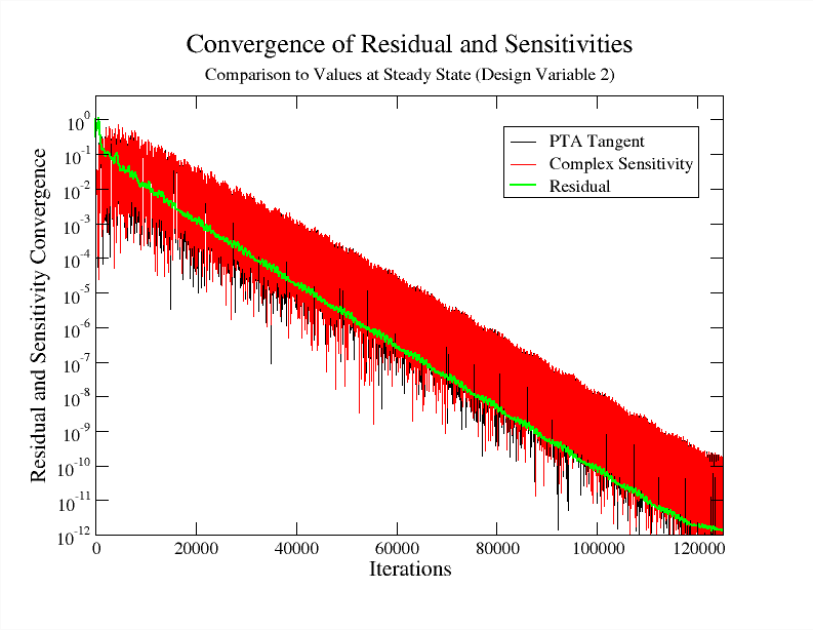
$$u^{k+1} = u^{k} + A(u^{k}(D), D)R(u^{k}, D) \tag{56}$$

$$\begin{aligned}\frac{du^{k,l+1}}{dD} = \frac{du^{k,0}}{dD} &+ CFL\alpha^{l}\left(\frac{\partial \Delta t}{\partial u^{k,0}}\frac{du^{k,0}}{dD} + \frac{\partial \Delta t}{\partial D}\right) R(u^{k,l}) \\ &+ CFL\alpha^{l}\Delta t\left(\frac{\partial R}{\partial D} + \left[\frac{\partial R(u^{k,l})}{\partial u^{k,l}}\right]_2 \frac{du^{k,l}}{dD}\right)\end{aligned} \tag{57}$$

Sensitivity Convergence for Non-Robust LSERK5



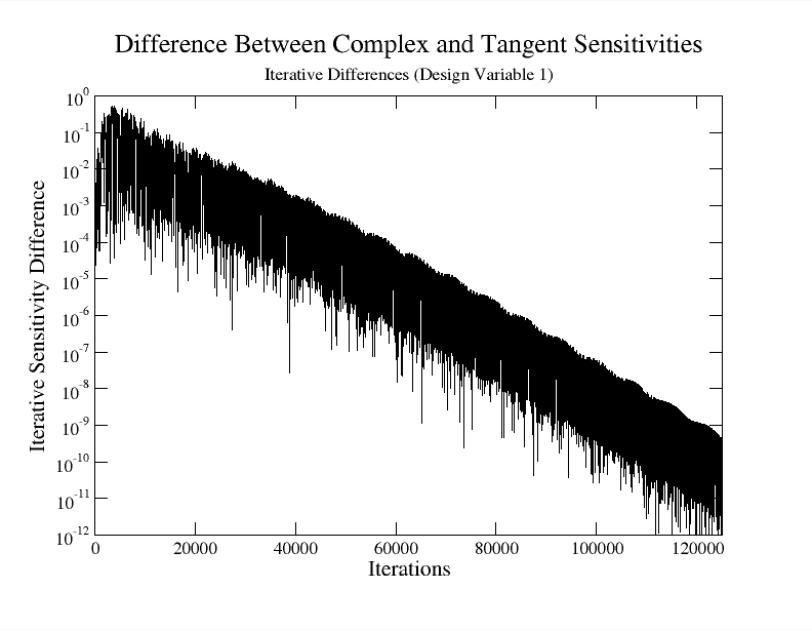
(a) Design Variable 1



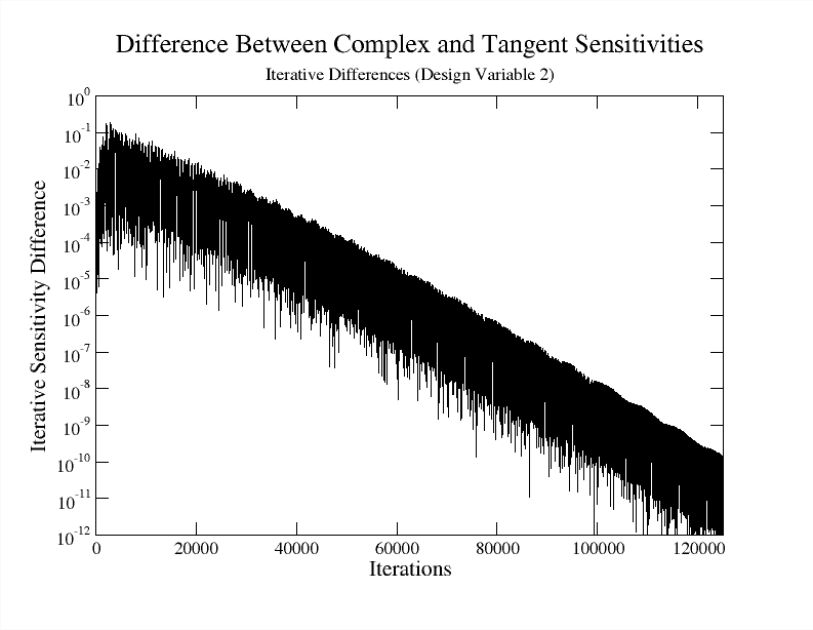
(b) Design Variable 2

Runge Kutta Sensitivity Convergence

Sensitivity Difference for Non-Robust LSERK45



(a) Design Variable 1



(b) Design Variable 2

Runge Kutta Sensitivity Difference

Inexact Linearization of Fixed-Point Iteration: Summary

- Convergence of the nonlinear problem converges the inexact linearization to the exact linearization, with the convergence occurring in the asymptotic convergence region primarily.

Inexact Linearization of Fixed-Point Iteration: Summary

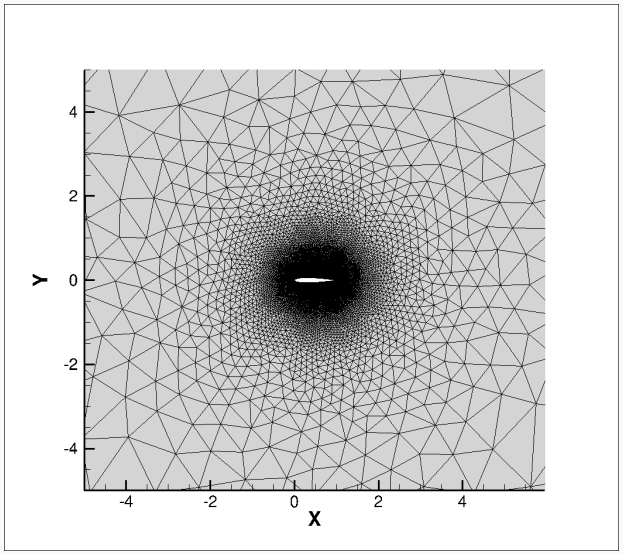
- Convergence of the nonlinear problem converges the inexact linearization to the exact linearization, with the convergence occurring in the asymptotic convergence region primarily.
- We were able to extend this proof to multistage schemes by looking at the effect of all the stages as an operator acting on the initial stage residual evaluation.

Inexact Linearization of Fixed-Point Iteration: Summary

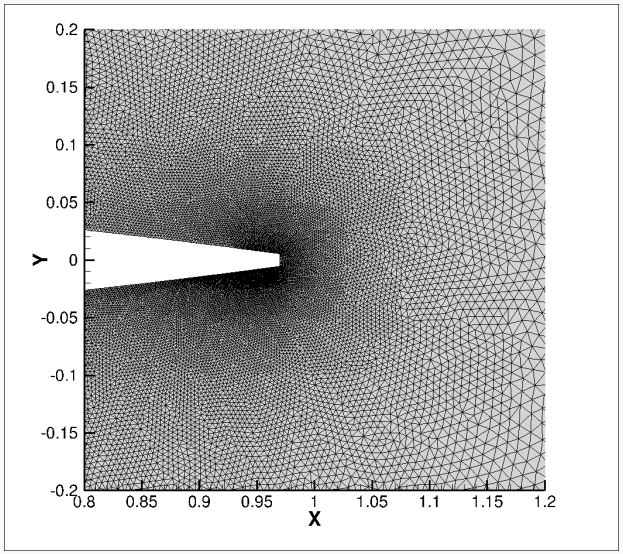
- Convergence of the nonlinear problem converges the inexact linearization to the exact linearization, with the convergence occurring in the asymptotic convergence region primarily.
- We were able to extend this proof to multistage schemes by looking at the effect of all the stages as an operator acting on the initial stage residual evaluation.
- Transitioning between exact and inexact linearizations would prevent the early error accumulation while still obtaining the asymptotic convergence of the Newton method (i.e. going from preconditioned RK to Newton with an RK smoothed residual for harder problems)

Effect of Averaging of the Objective Function on Sensitivity Behavior

Non-Converging Transonic Case Introduction



(a) Mesh

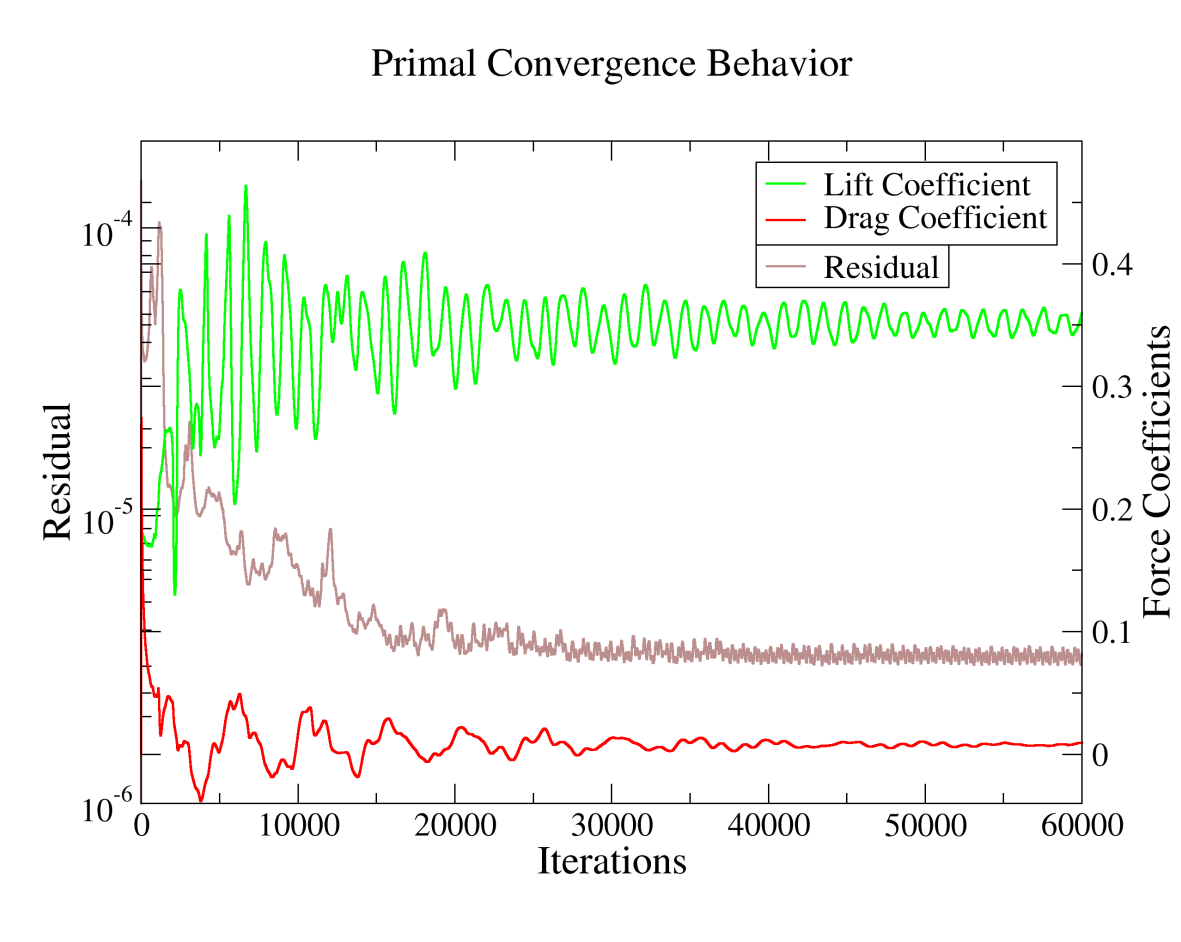


(b) Trailing Edge

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

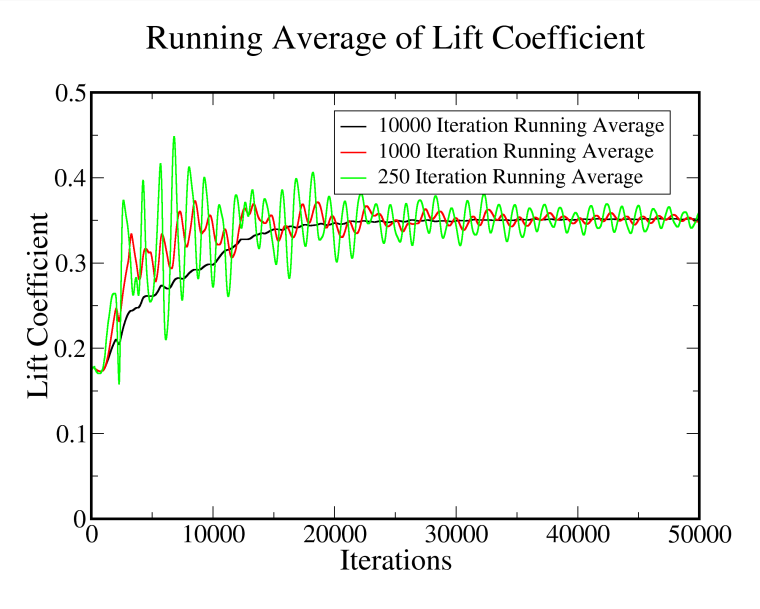
- Design variables are three Hicks-Henne bump functions.
- $Mach = .7$, $\alpha = 2^o$
- Explicit inverse distance weighting used for mesh motion and mesh sensitivities
- LSERK5 solver

Primal Behavior of Non-Converging Transonic Simulation

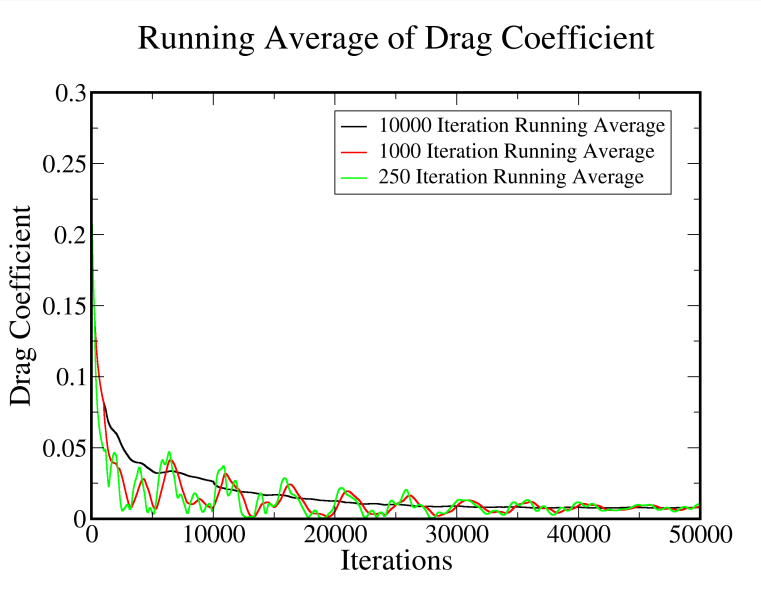


Primal convergence for cut-off NACA0012 airfoil at $Mach = .7$, $\alpha = 2^o$

Running Average of Force Coefficients



(a) C_L over different averaging windows



(b) C_D over different averaging windows

Force coefficient averaging over different windows

Why is Angle Important?

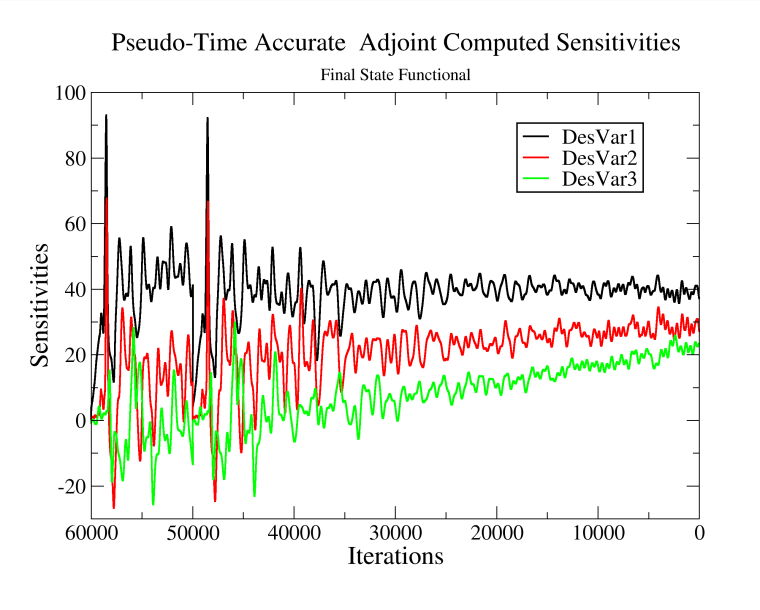
- The important thing about the sensitivity vector is the direction.
- If the direction is off, then the optimizer takes a suboptimal path and this leads to greater expense and possibly suboptimal designs
- If the magnitude is off, then through the line search maybe the step isn't as large, but that's not a large issue for an optimizer

The angle θ is calculated as follows, with u and v being the sensitivity vectors:

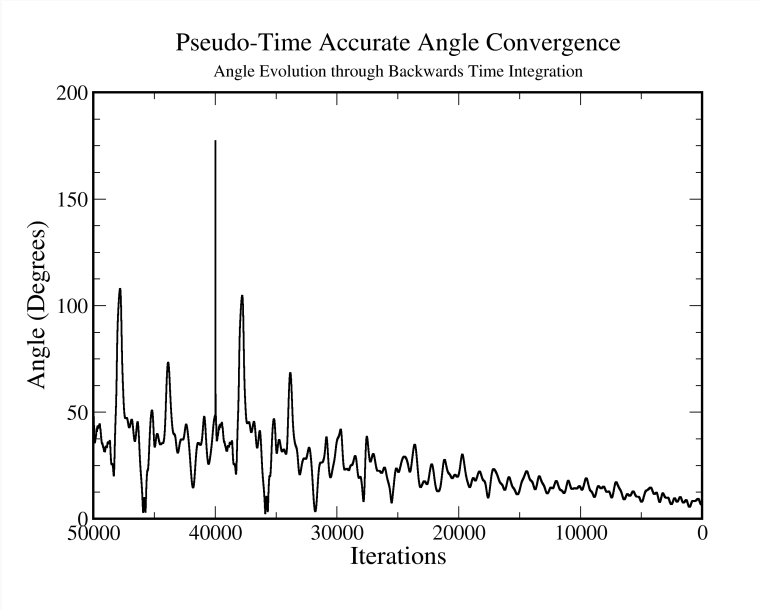
$$a = \frac{u \cdot v}{\|u\| \, \|v\|} \tag{58}$$

$$\begin{aligned} \theta &= arccos(a), \theta \leq 180 \\ \theta &= 360 - arccos(a), \theta > 180 \end{aligned} \tag{59}$$

Sensitivities for Final State Objective Functional



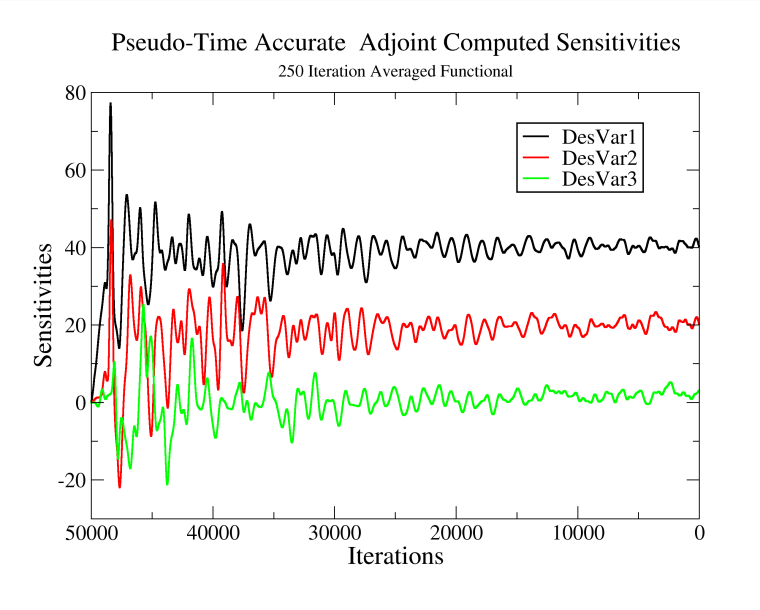
(a) Sensitivities



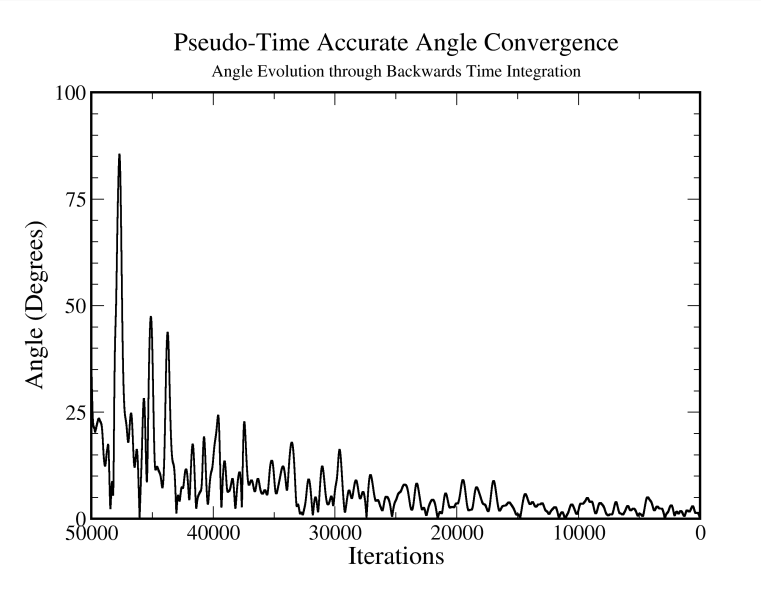
(b) Angle Convergence

Pseudo-Time accurate adjoint sensitivities for objective functional calculated at the final state at $Mach = .7$, $\alpha = 2^o$

Sensitivities for 250 Iteration Window Averaged Objective Functional



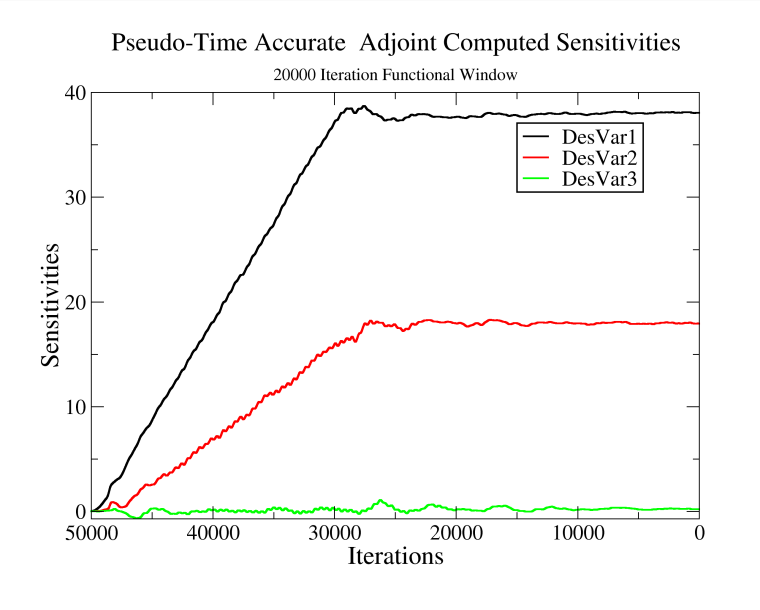
(a) Sensitivities



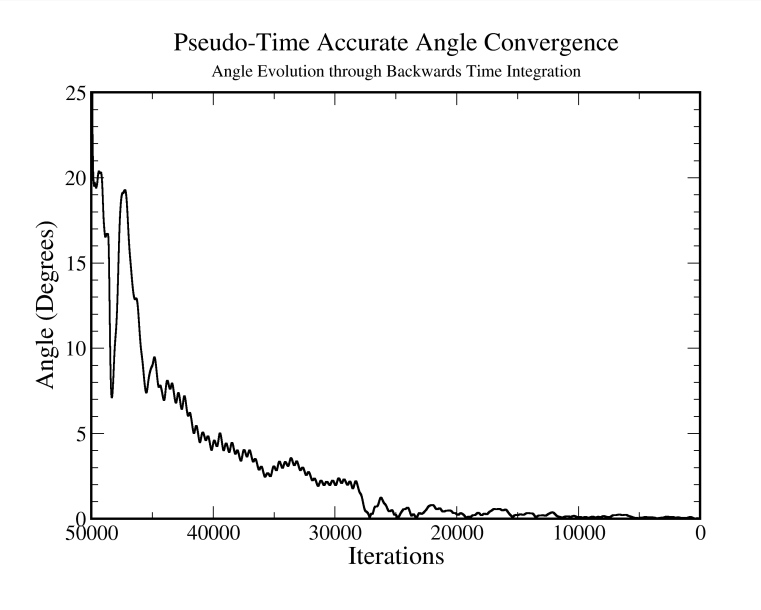
(b) Angle Convergence

Pseudo-Time accurate adjoint sensitivities for 250 iteration average objective functional at $Mach = .7$, $\alpha = 2^o$

Sensitivities for 20000 Iteration Window Averaged Objective Functional



(a) Sensitivities



(b) Angle Convergence

Pseudo-Time accurate adjoint sensitivities for 20000 iteration average objective functional at $Mach = .7$, $\alpha = 2^o$

Sensitivity Vector Comparison of Final State Steady State Adjoint to PTA Adjoint

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

Pseudo-Time accurate adjoint for 20000 iteration averaging window for objective functional and steady-state sensitivities

Sensitivity Vector Comparison of Final State Steady State Adjoint to PTA Adjoint

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

Pseudo-Time accurate adjoint for 20000 iteration averaging window for objective functional and steady-state sensitivities

The angle between the two sensitivity vectors is $\theta = 7.98972^o$

Sensitivity Vector Comparison of Averaged State Steady State Adjoint to PTA Adjoint

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

Pseudo-Time accurate adjoint for 20000 iteration averaging window for objective functional and steady-state sensitivities

Sensitivity Vector Comparison of Averaged State Steady State Adjoint to PTA Adjoint

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

Pseudo-Time accurate adjoint for 20000 iteration averaging window for objective functional and steady-state sensitivities

The angle between the two sensitivity vectors is $\theta = 3.93855^o$

Averaging of Objective Function: Summary

- Without averaging the objective function in pseudo-time the sensitivities are too dependent on the number of iterations and the initial conditions.

Averaging of Objective Function: Summary

- Without averaging the objective function in pseudo-time the sensitivities are too dependent on the number of iterations and the initial conditions.
- Averaging the objective removes these dependencies and allows for partial backwards-in-pseudo-time integration.

Averaging of Objective Function: Summary

- Without averaging the objective function in pseudo-time the sensitivities are too dependent on the number of iterations and the initial conditions.
- Averaging the objective removes these dependencies and allows for partial backwards-in-pseudo-time integration.
- The steady-state adjoint sensitivities are notably different than the PTA adjoint ones in magnitude, direction, and sign.

Aerodynamic Shape Optimizations

Aerodynamic Shape Optimization Overview

- NACA0012 with detached bow shock and composite objective function of lift, drag and entropy (symmetric)

Aerodynamic Shape Optimization Overview

- NACA0012 with detached bow shock and composite objective function of lift, drag and entropy (symmetric)
- NACA0012 with blunt trailing edge in transonic flow with large angle of incidence with composite objective function of lift and drag (asymmetric)

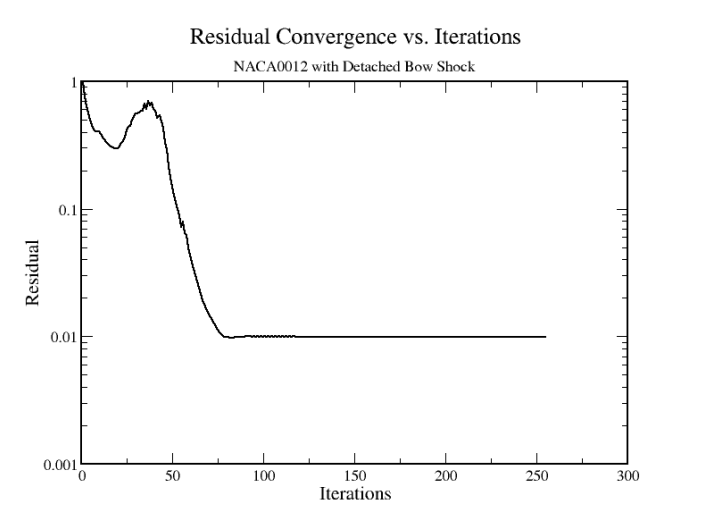
Aerodynamic Shape Optimization Overview

- NACA0012 with detached bow shock and composite objective function of lift, drag and entropy (symmetric)
- NACA0012 with blunt trailing edge in transonic flow with large angle of incidence with composite objective function of lift and drag (asymmetric)
- inverse distance weighting (with rotation term) for mesh motion and sensitivities

Aerodynamic Shape Optimization Overview

- NACA0012 with detached bow shock and composite objective function of lift, drag and entropy (symmetric)
- NACA0012 with blunt trailing edge in transonic flow with large angle of incidence with composite objective function of lift and drag (asymmetric)
- inverse distance weighting (with rotation term) for mesh motion and sensitivities
- objective function calculated over an averaging window and the sensitivities are integrated only back through that window

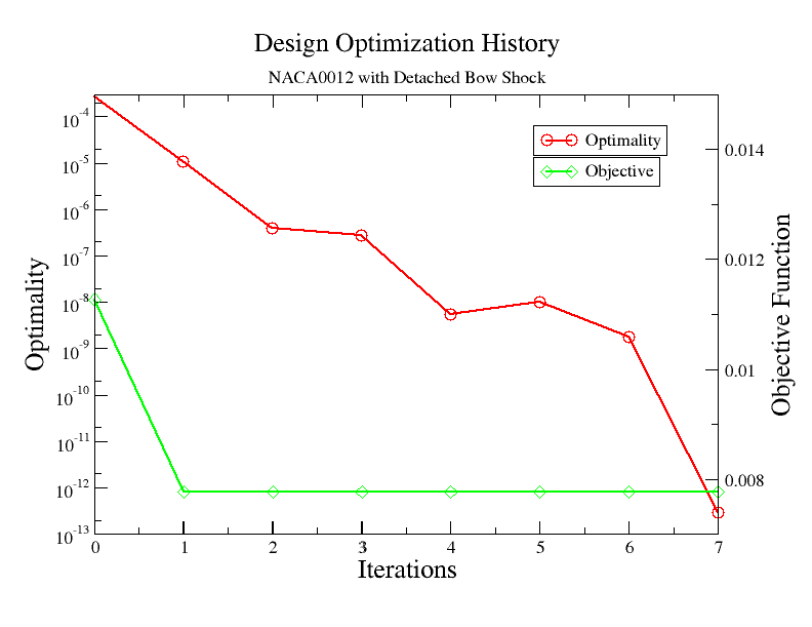
NACA0012 with Detached Bow Shock



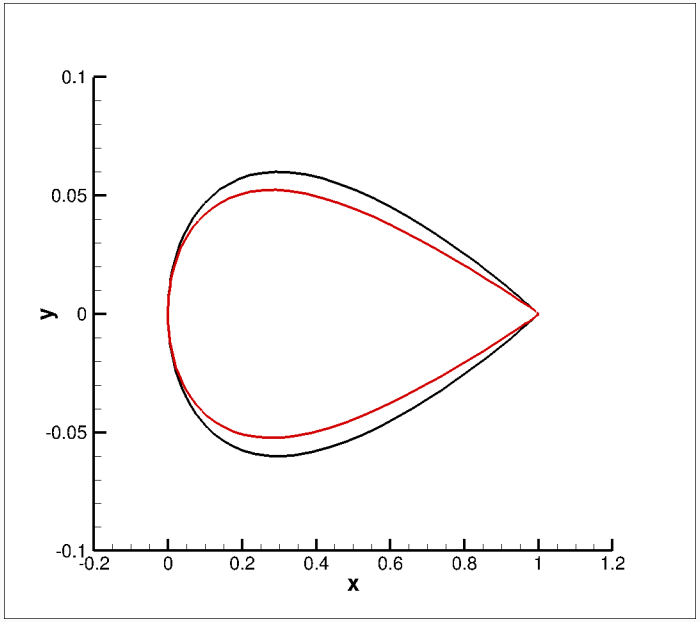
Analysis convergence plot

- $Mach = 1.25$, $\alpha = 3^o$
- objective function penalty method to keep lift constant and minimize drag and entropy
- analysis code has stalled due to limiter behavior near the shock

Optimization of NACA0012 with Detached Bow Shock: Summary



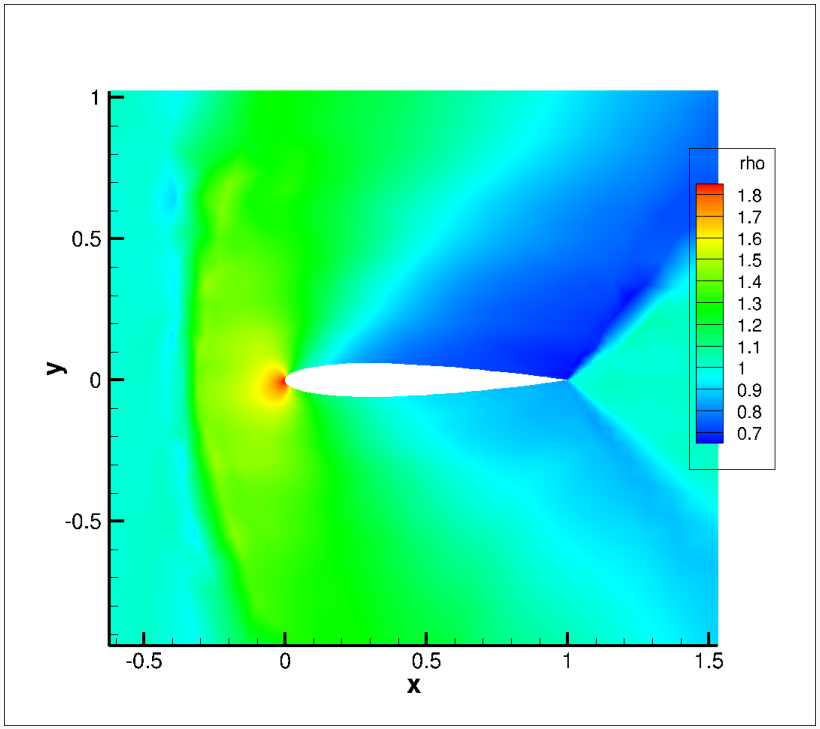
(a) Design Cycle Convergence



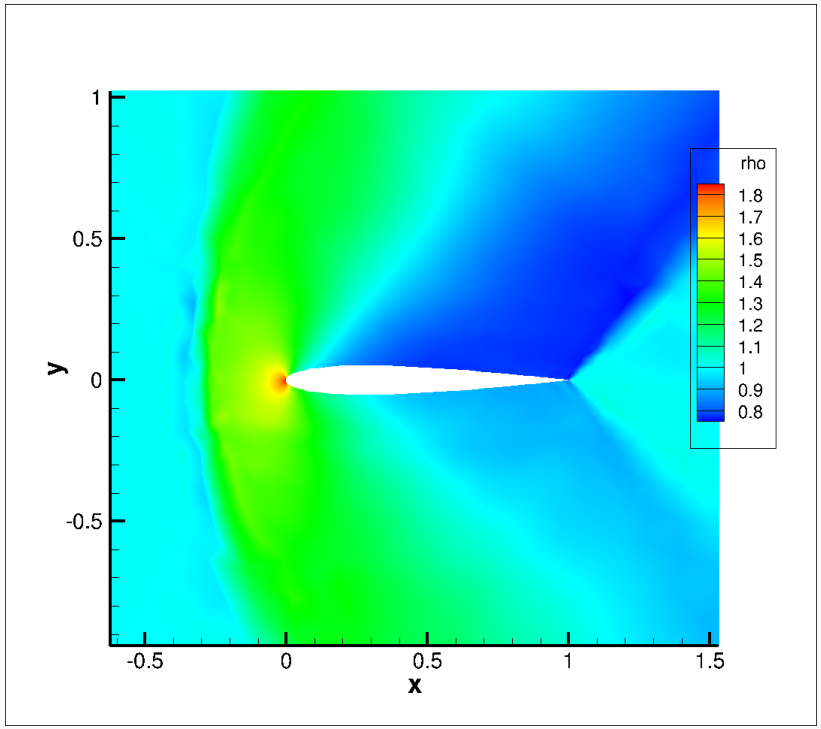
(b) Airfoil Comparison

Design cycle summary

Optimization of NACA0012 with Detached Bow Shock: Density



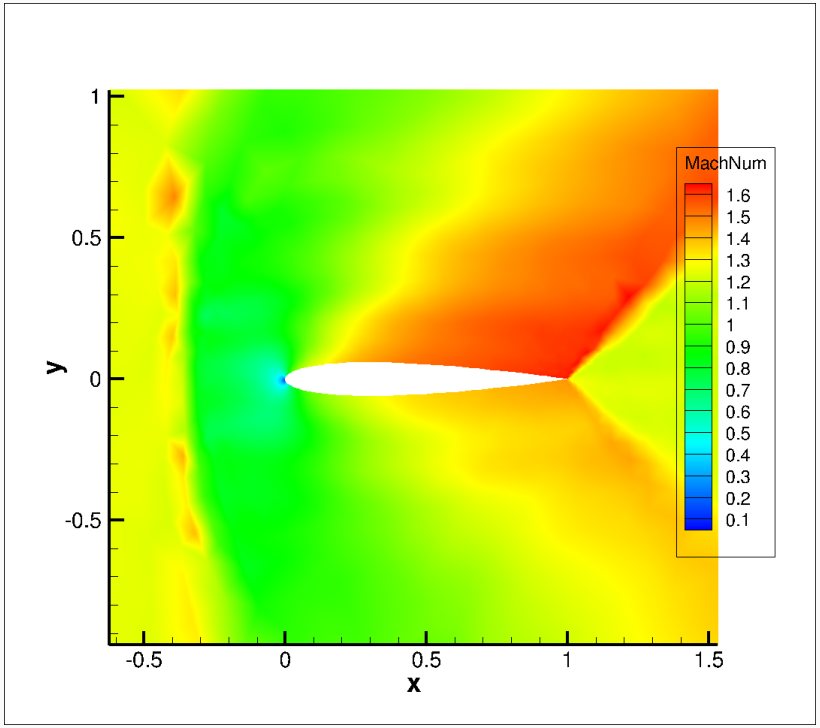
(a) Baseline Design



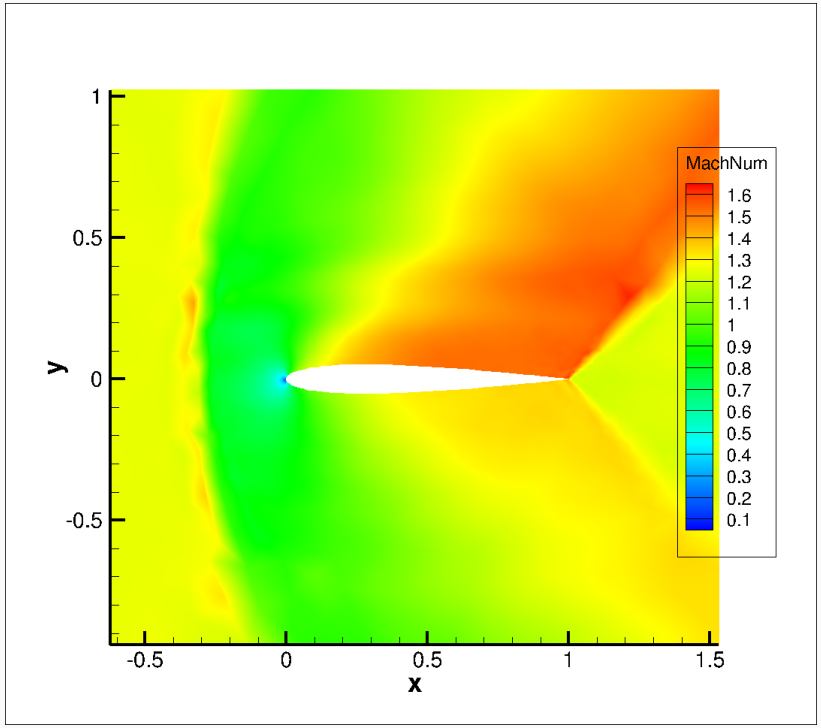
(b) Optimized Design

Density field comparison

Optimization of NACA0012 with Detached Bow Shock: Mach



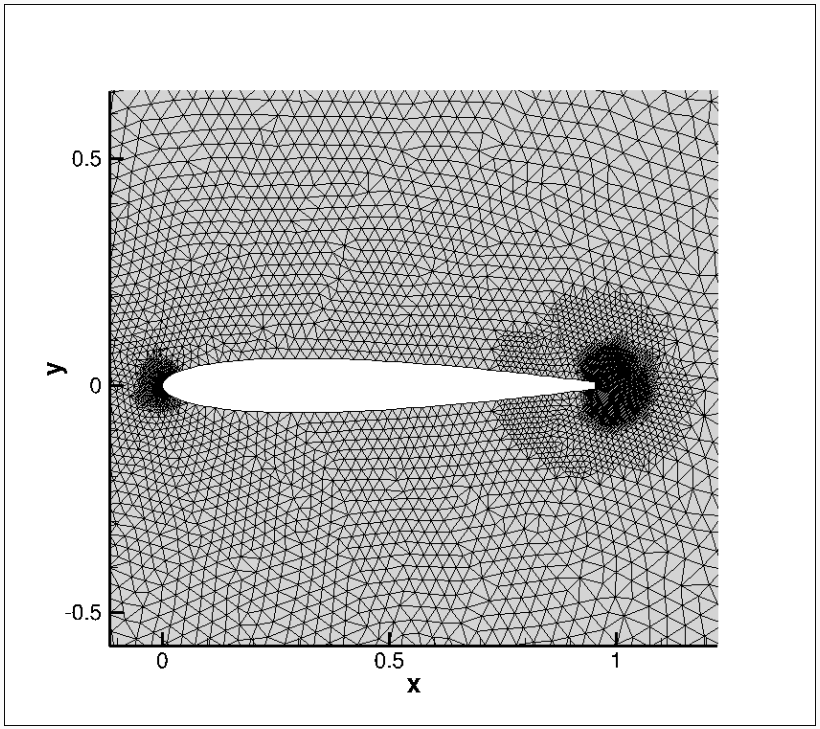
(a) Baseline Design



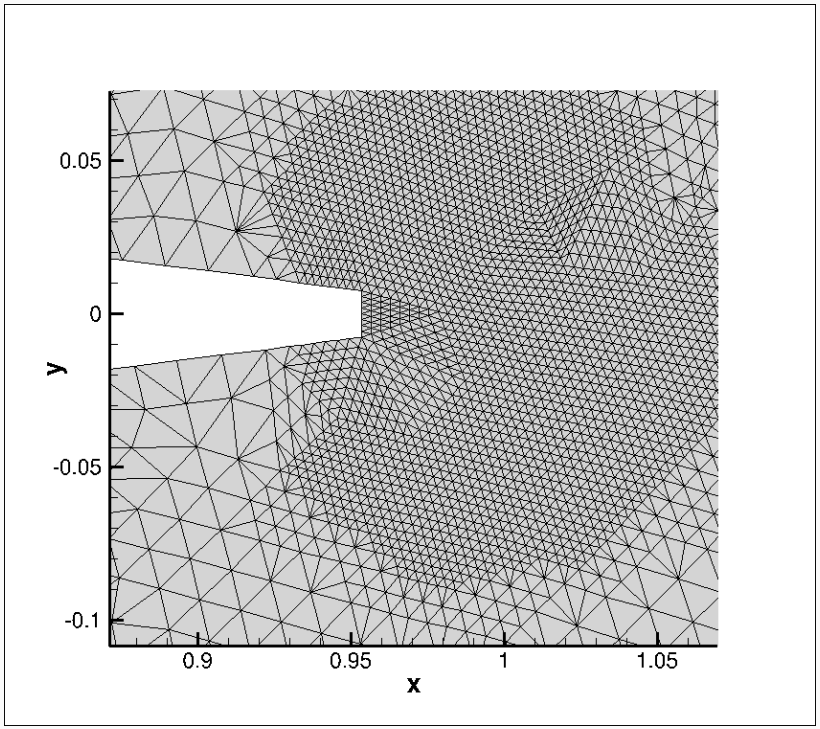
(b) Optimized Design

Mach field comparison

Optimization of NACA0012 Airfoil with Trailing Edge Unsteadiness: Mesh



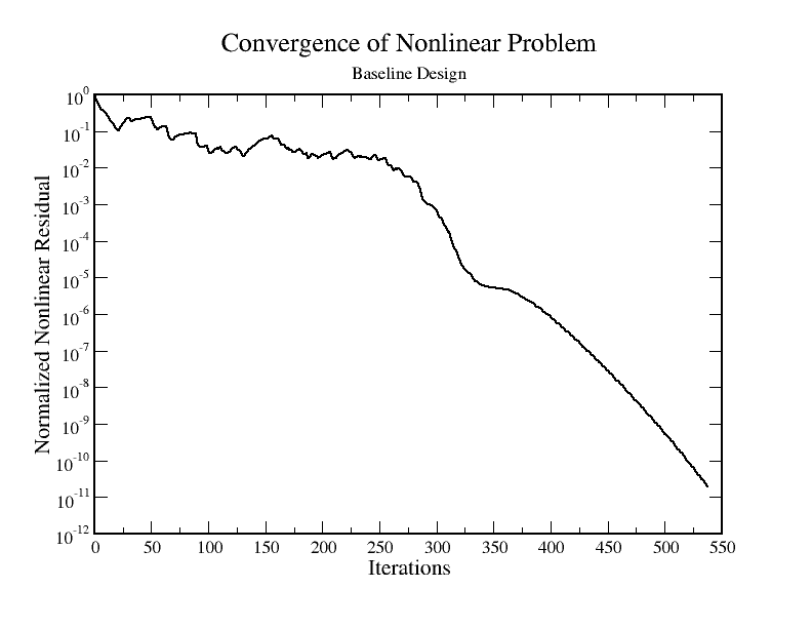
(a) Airfoil Mesh



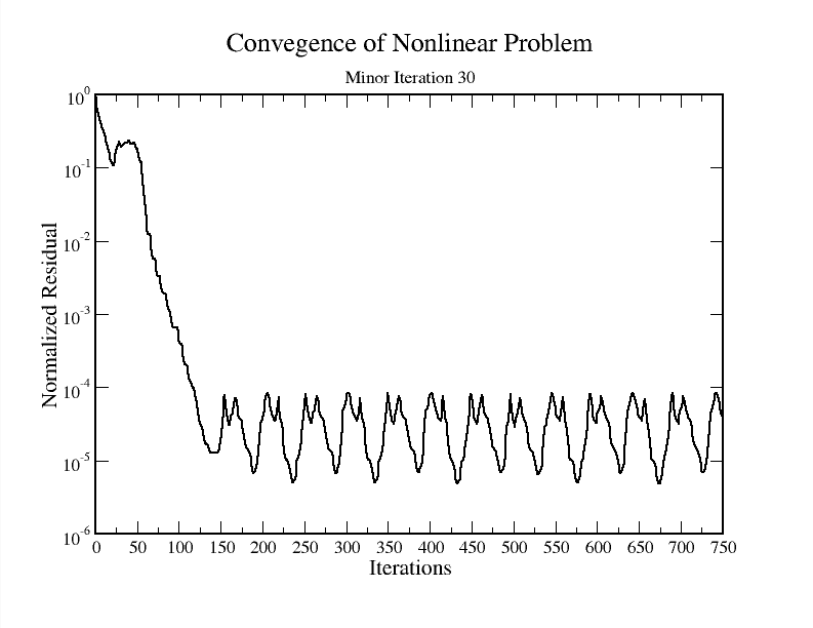
(b) Airfoil Trailing Edge

Computational mesh

Optimization of NACA0012 Airfoil in High Angle of Attack



(a) Baseline Nonlinear Convergence

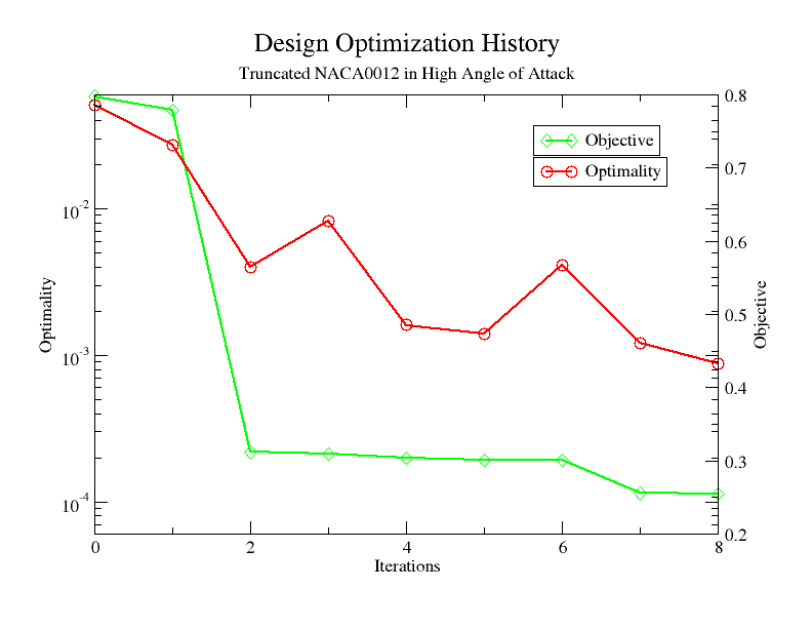


(b) Perturbed Nonlinear Convergence

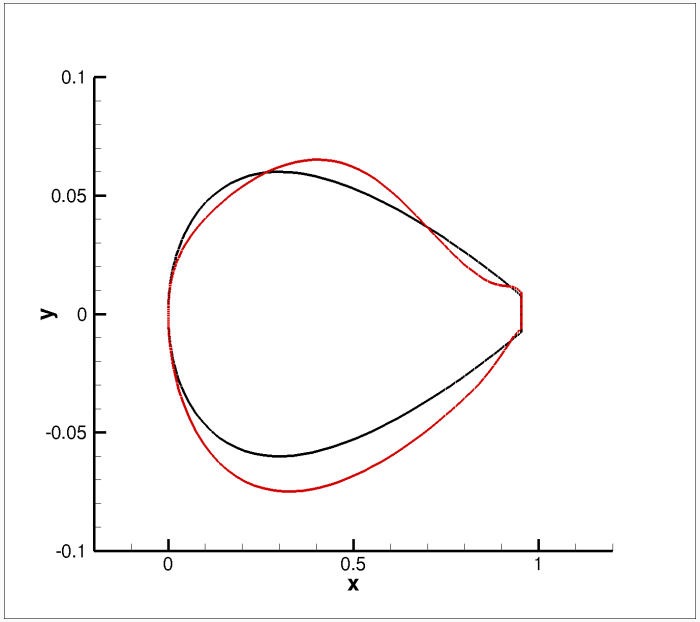
Comparison of analysis convergence plot for NACA0012 with a blunt trailing edge in transonic flow with high angle of attack

- $Mach = .75$, $\alpha = 6.0^o$
- objective function penalty method with target lift and drag
- analysis code has stalled due to trailing edge unsteadiness

Optimization of NACA0012 Airfoil in High Angle of Attack: Summary



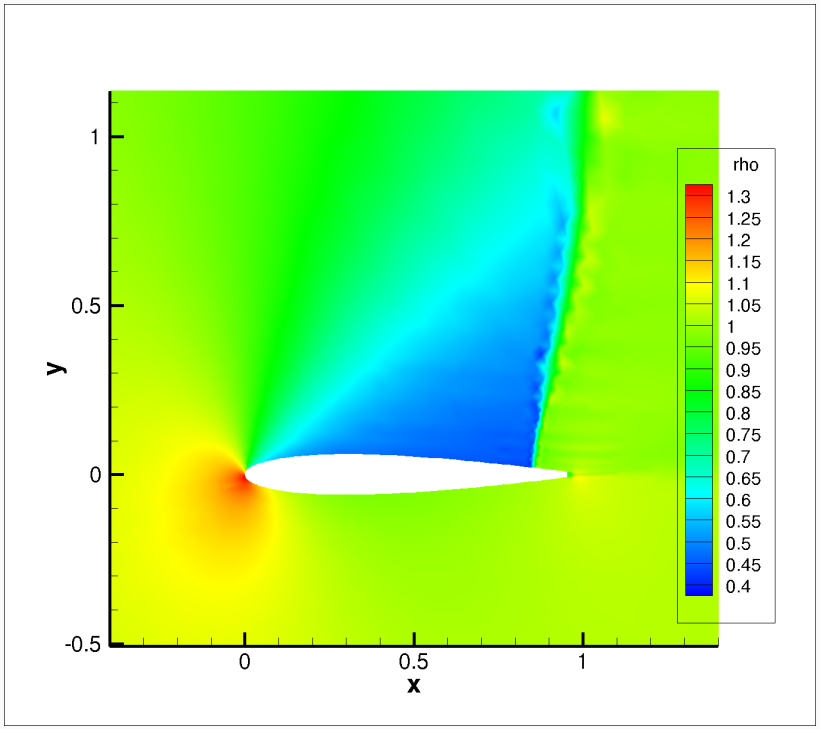
(a) Design Cycle Convergence



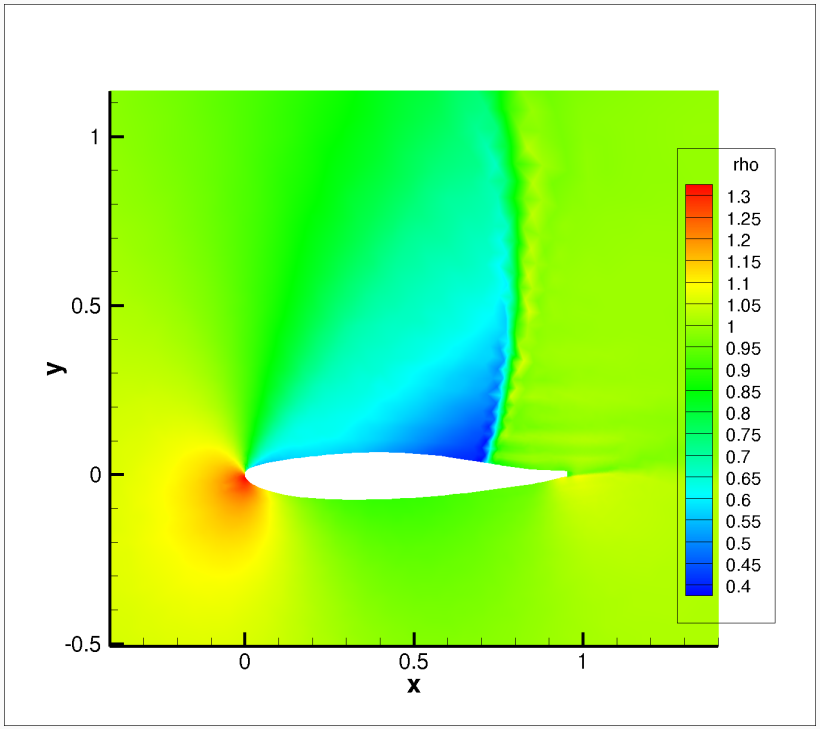
(b) Airfoil Comparison

Design cycle summary

Optimization of NACA0012 Airfoil in High Angle of Attack: Density



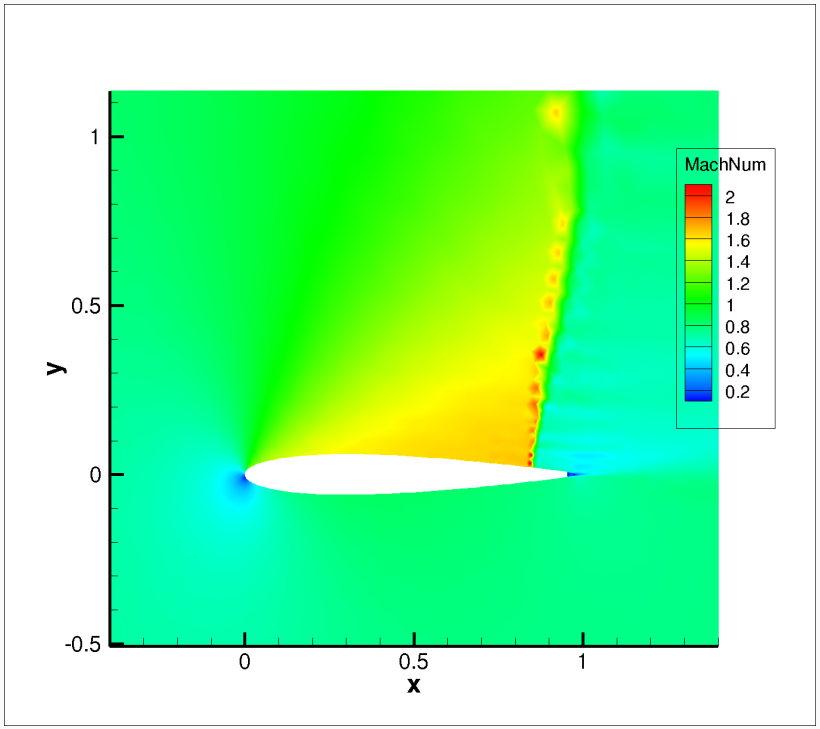
(a) Baseline Design



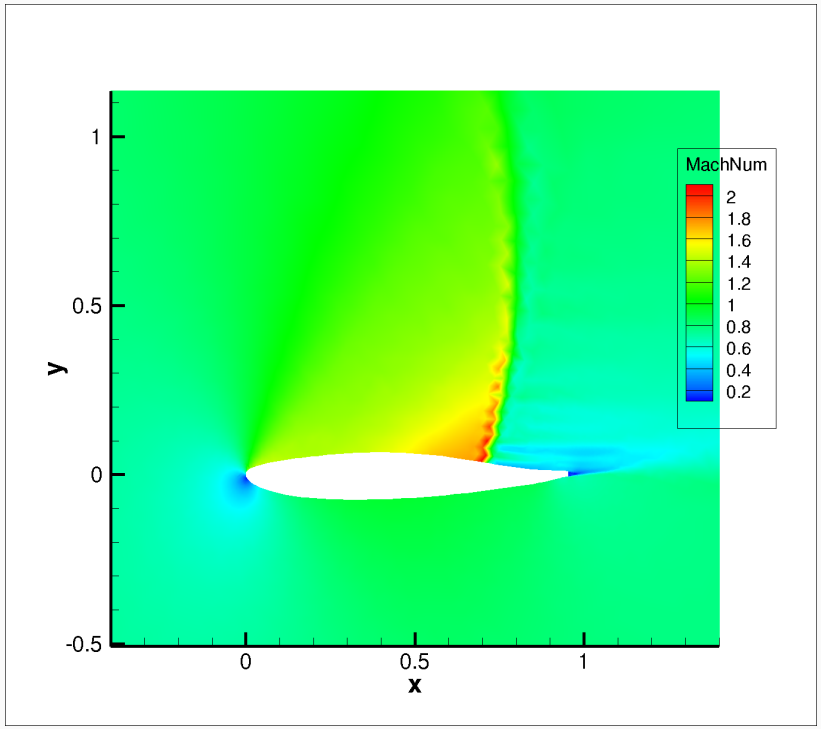
(b) Optimized Design

Density field comparison

Optimization of NACA0012 Airfoil in High Angle of Attack: Mach



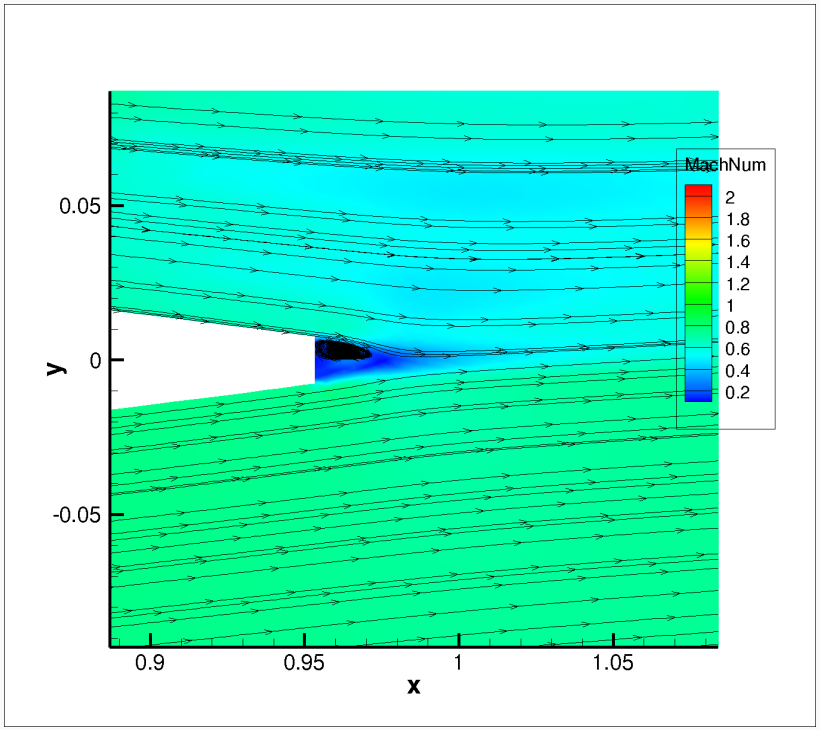
(a) Baseline Design



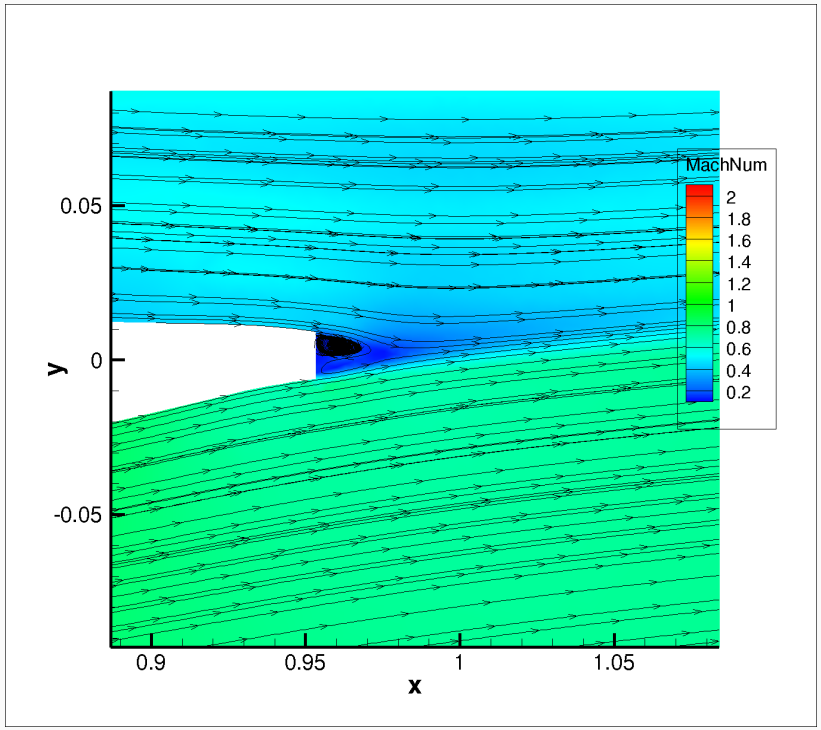
(b) Optimized Design

Mach field comparison

Optimization of NACA0012 Airfoil in High Angle of Attack: Trailing Edge



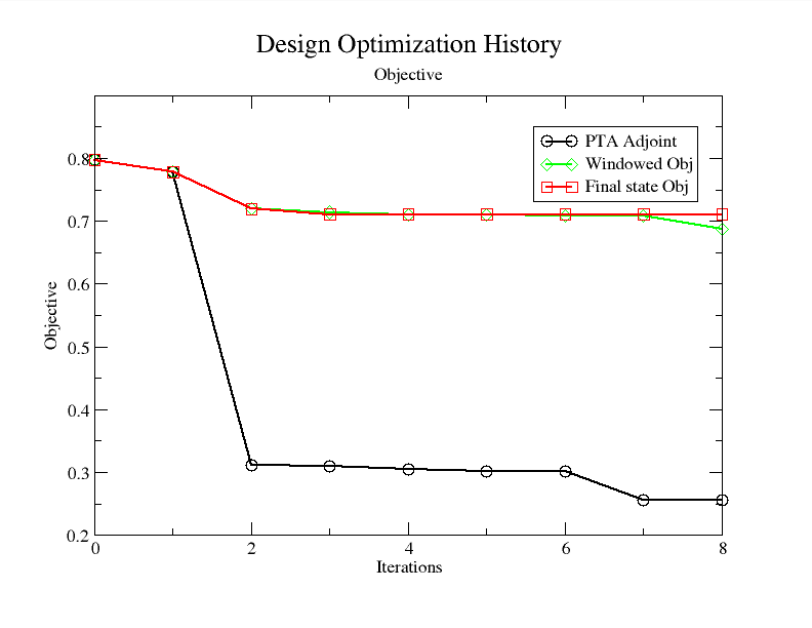
(a) Baseline Trailing Edge with Streamlines



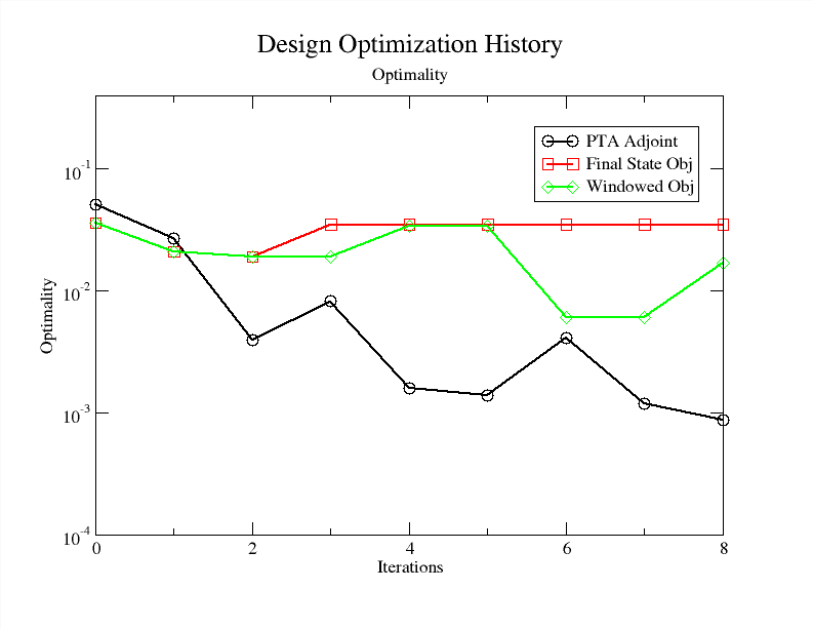
(b) Optimized Trailing Edge with Streamlines

Mach field comparison with streamlines

Optimization of NACA0012 Airfoil in High Angle of Attack: Comparison of Design Cycle to Steady-State Adjoint



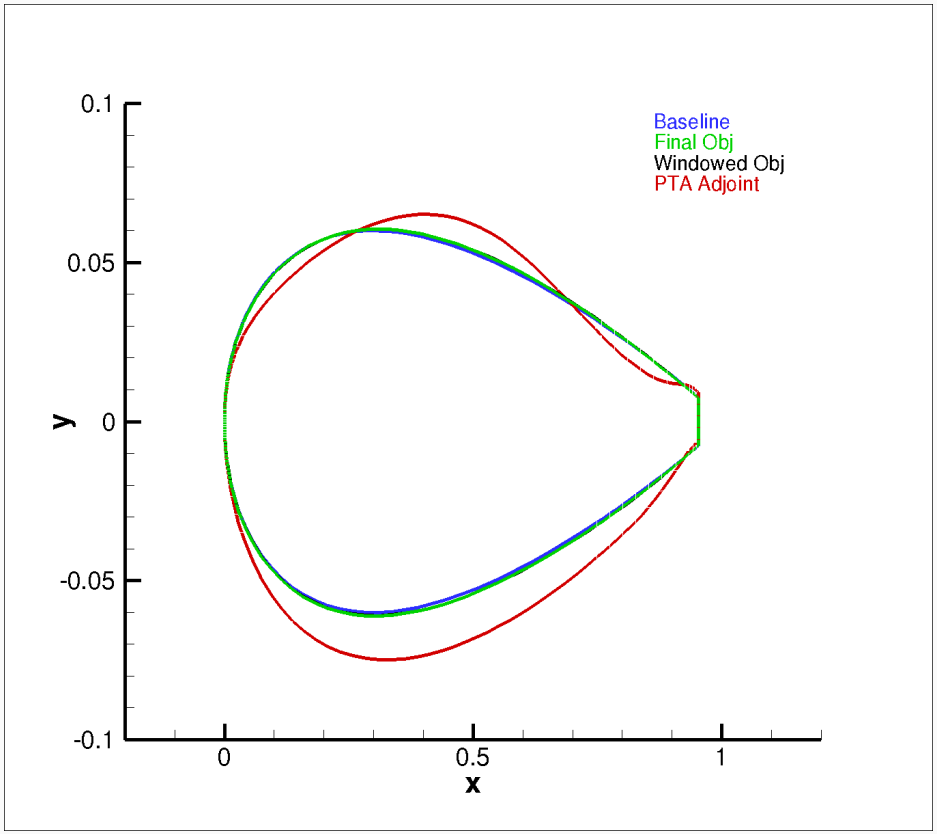
(a) Objective Function Convergence



(b) Optimality Convergence

Design cycle summary for NACA0012 truncated at 95% of the chord in transonic flow with high angle attack: comparison to steady state adjoint results

Optimization of NACA0012 Airfoil in High Angle of Attack: Comparison of Airfoil Geometry



Baseline and optimized airfoils for high angle of attack optimization

Optimization of Non-Converging Problems using PTA Adjoint: Summary

- We perform optimizations using sensitivities of averaged objective functions through inexactly linearizing the fixed-point iterations.

Optimization of Non-Converging Problems using PTA Adjoint: Summary

- We perform optimizations using sensitivities of averaged objective functions through inexactly linearizing the fixed-point iterations.
- The small-scale unsteadiness optimization driven by the PTA adjoint computed sensitivities satisfies the optimality condition.

Optimization of Non-Converging Problems using PTA Adjoint: Summary

- We perform optimizations using sensitivities of averaged objective functions through inexactly linearizing the fixed-point iterations.
- The small-scale unsteadiness optimization driven by the PTA adjoint computed sensitivities satisfies the optimality condition.
- The PTA adjoint optimization is far more successful than the steady state adjoint optimizations for the large-scale unsteadiness case.

Optimization of Non-Converging Problems using PTA Adjoint: Summary

- We perform optimizations using sensitivities of averaged objective functions through inexactly linearizing the fixed-point iterations.
- The small-scale unsteadiness optimization driven by the PTA adjoint computed sensitivities satisfies the optimality condition.
- The PTA adjoint optimization is far more successful than the steady state adjoint optimizations for the large-scale unsteadiness case.
- Use of windowing regularization techniques could help with convergence of the optimality and the general design behavior.

Conclusions and Future Work

Conclusions

- We developed a novel formulation of the tangent and adjoint systems through linearization of the fixed-point iterations.

Conclusions

- We developed a novel formulation of the tangent and adjoint systems through linearization of the fixed-point iterations.
- We can develop error estimates for inexact linearization of the fixed-point iteration and recover useful sensitivities.

Conclusions

- We developed a novel formulation of the tangent and adjoint systems through linearization of the fixed-point iterations.
- We can develop error estimates for inexact linearization of the fixed-point iteration and recover useful sensitivities.
- We can average the objective functions in pseudo-time to lower dependence on the initial conditions and allow for partial backwards-in-pseudo-time integration.

Conclusions

- We developed a novel formulation of the tangent and adjoint systems through linearization of the fixed-point iterations.
- We can develop error estimates for inexact linearization of the fixed-point iteration and recover useful sensitivities.
- We can average the objective functions in pseudo-time to lower dependence on the initial conditions and allow for partial backwards-in-pseudo-time integration.
- We can use these inexactly linearized fixed point and averaged objective functions to drive design optimizations whose analysis doesn't converge and outperform the steady-state adjoint driven design optimizations.

Conclusions

- We developed a novel formulation of the tangent and adjoint systems through linearization of the fixed-point iterations.
- We can develop error estimates for inexact linearization of the fixed-point iteration and recover useful sensitivities.
- We can average the objective functions in pseudo-time to lower dependence on the initial conditions and allow for partial backwards-in-pseudo-time integration.
- We can use these inexactly linearized fixed point and averaged objective functions to drive design optimizations whose analysis doesn't converge and outperform the steady-state adjoint driven design optimizations.
- We can use the pseudo-time accurate adjoint approach to compute error estimates and refinement criteria.

Future Work

Algorithmic improvements/goals:

- windowing regularization

Future Work

Algorithmic improvements/goals:

- windowing regularization
- increased robustness of error estimation through improved interpolation/limiting mechanics

Future Work

Algorithmic improvements/goals:

- windowing regularization
- increased robustness of error estimation through improved interpolation/limiting mechanics
- coupled AMR and design

Thanks

Committee:

Professor Mavriplis (chair)

Professor Stoellinger

Professor Fertig

Professor Ginting (outside member)

Professor Nadarajah (external member)

Thanks for listening!

Dual-Weighted Constraint for Error Estimation

Dual-Weighted Constraint for Error Estimation

- Applies the pseudo-time approach to error estimation

Dual-Weighted Constraint for Error Estimation

- Applies the pseudo-time approach to error estimation
- Seeks to quantify the error in the objective function due to the mesh

Dual-Weighted Constraint for Error Estimation

- Applies the pseudo-time approach to error estimation
- Seeks to quantify the error in the objective function due to the mesh
- Agnostic as to the impact of the objective function windowing

Dual-Weighted Constraint: Notes

- On its face this method would require computing the adjoint and the constraint on the fine mesh

Dual-Weighted Constraint: Notes

- On its face this method would require computing the adjoint and the constraint on the fine mesh
- This is not practical, so the adjoint is computed on the coarse mesh and interpolated onto the fine mesh

Dual-Weighted Constraint: Notes

- On its face this method would require computing the adjoint and the constraint on the fine mesh
- This is not practical, so the adjoint is computed on the coarse mesh and interpolated onto the fine mesh
- This does require weighting the fine-mesh fixed point by the adjoint similar to how the sensitivities weight the constraint derivative by the adjoint

Dual-Weighted Constraint: Notes

- On its face this method would require computing the adjoint and the constraint on the fine mesh
- This is not practical, so the adjoint is computed on the coarse mesh and interpolated onto the fine mesh
- This does require weighting the fine-mesh fixed point by the adjoint similar to how the sensitivities weight the constraint derivative by the adjoint
- Partial backwards in time integration and curvature correction

Dual-Weighted Constraint: Notes

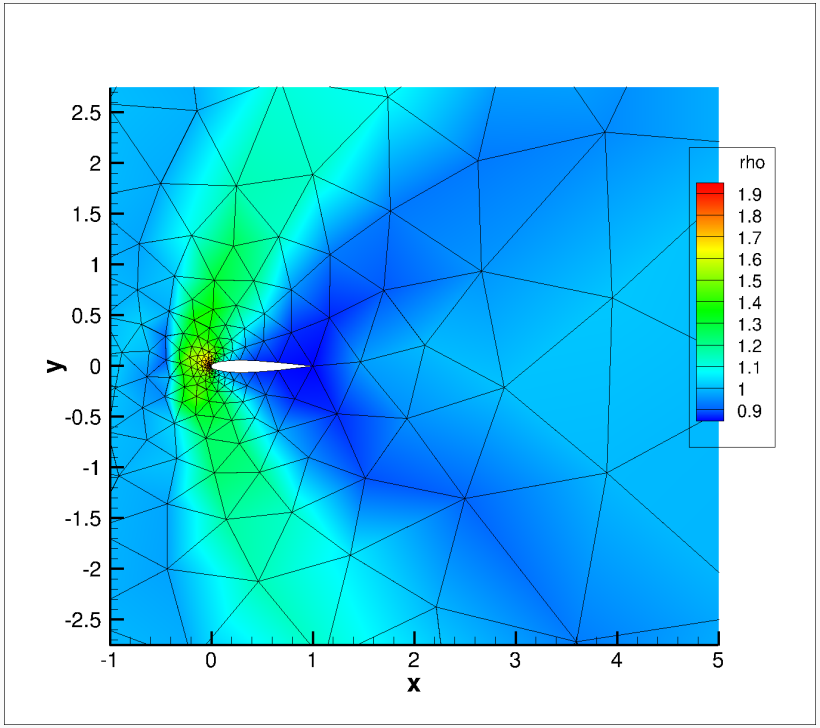
- On its face this method would require computing the adjoint and the constraint on the fine mesh
- This is not practical, so the adjoint is computed on the coarse mesh and interpolated onto the fine mesh
- This does require weighting the fine-mesh fixed point by the adjoint similar to how the sensitivities weight the constraint derivative by the adjoint
- Partial backwards in time integration and curvature correction
- The computation of the constraint on the fine mesh is still impractical, so virtual mesh methods are attempted

PTA Error Estimation and AMR Results

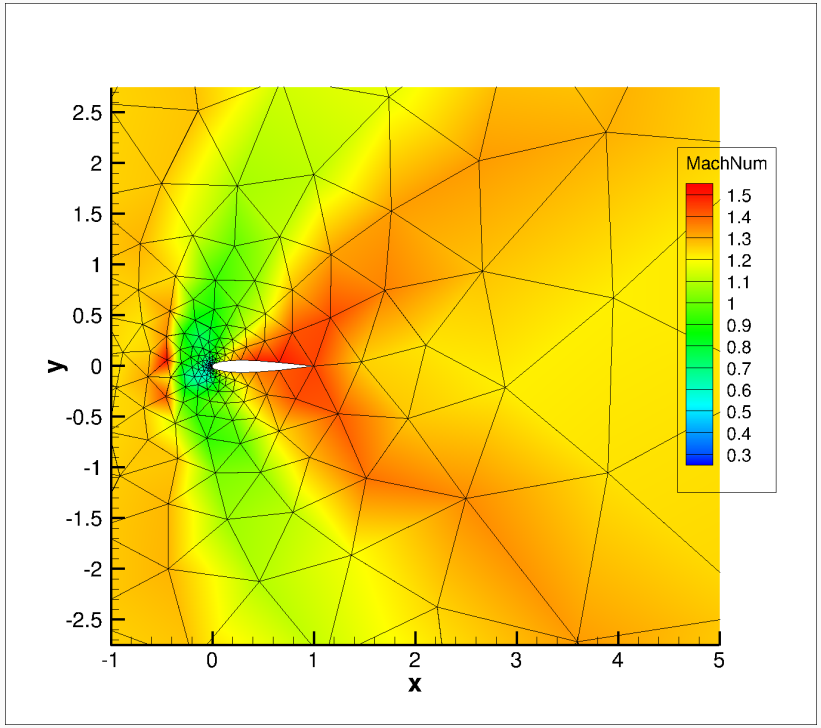
Supersonic Detached Bow Shock Case

Supersonic Detached Bow Shock Case: Initial Mesh

NACA0012 in $M = 1.25$, $\alpha = 0^o$ flow:



(a) Coarse Mesh Density

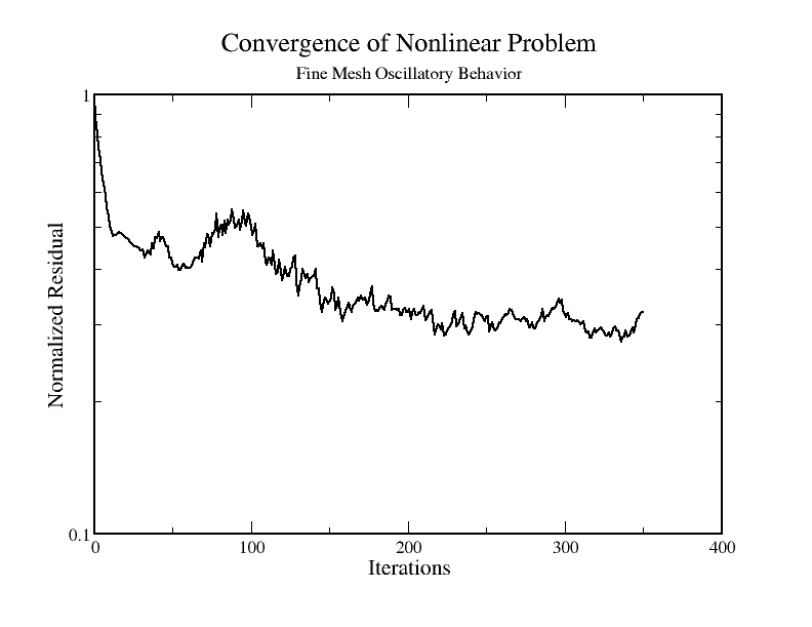


(b) Coarse Mesh Mach Number

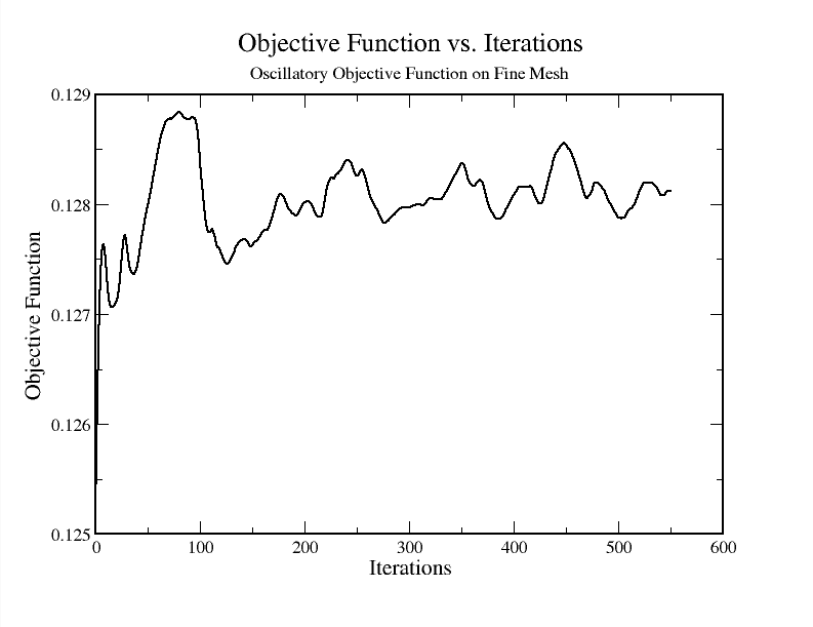
Coarse mesh for detached bow shock error estimation case

Quantity of interest is lift, drag, and entropy

Supersonic Detached Bow Shock Case: Fine Mesh Convergence and Objective



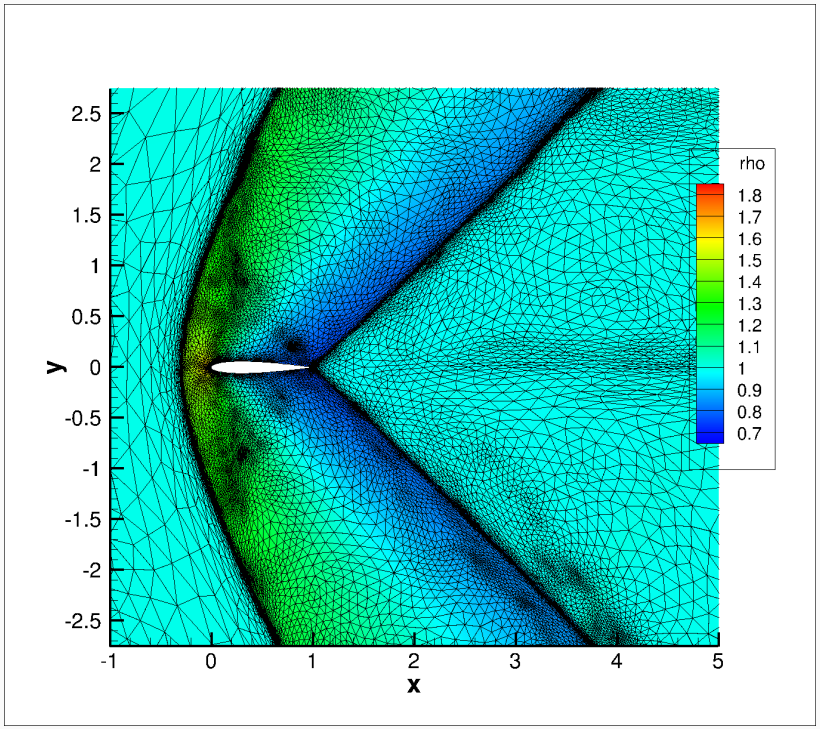
(a) Fine Mesh Convergence



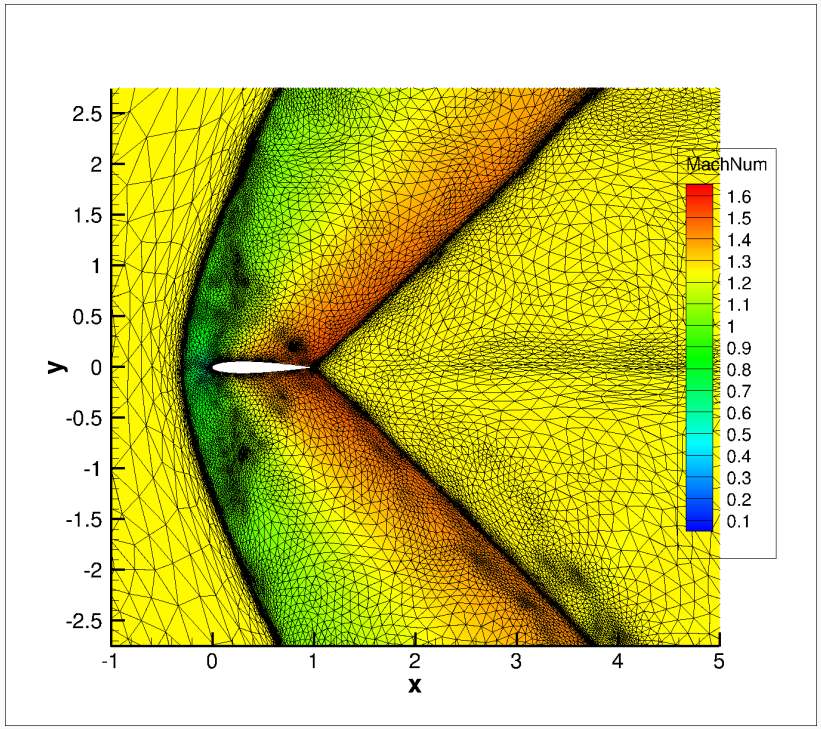
(b) Fine Mesh Objective

Objective and convergence behavior on a fine mesh

Embedded Mesh Error Estimation with Functional Correction: Final Adaptation Cycle



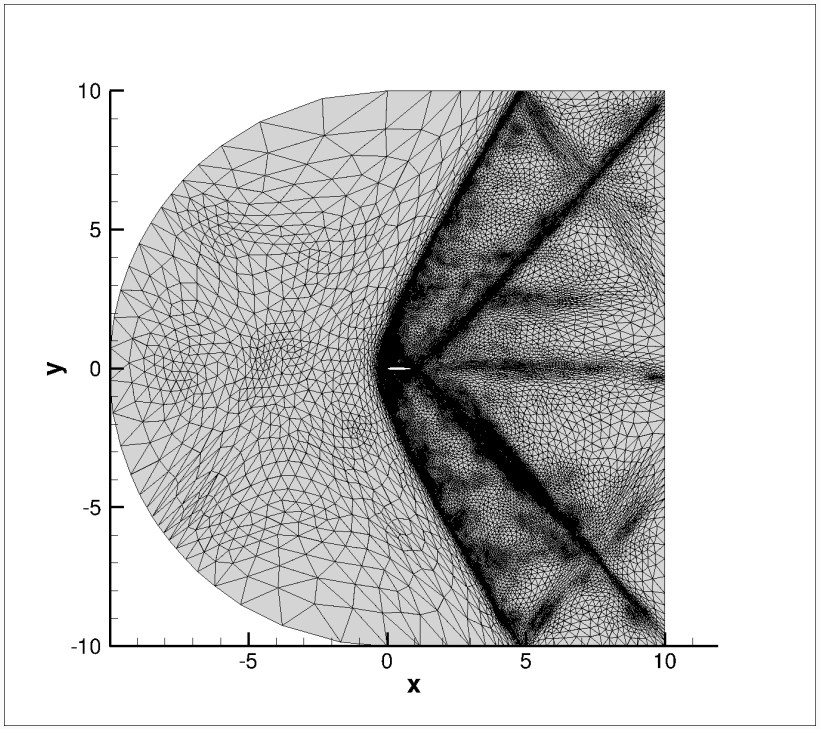
(a) Density with mesh



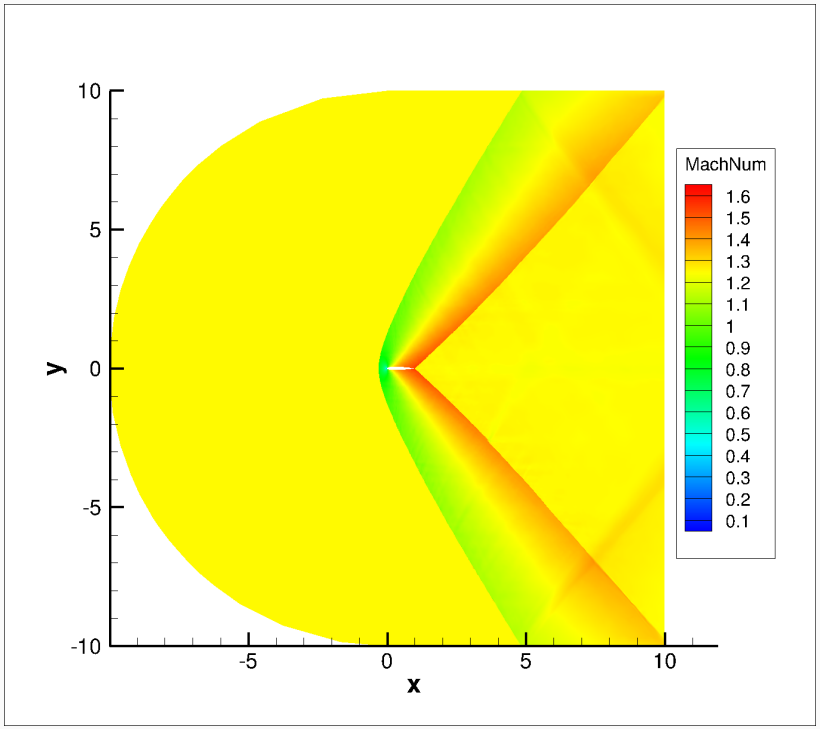
(b) Mach number with mesh

18th and final adaptation cycle for detached bow shock with error estimation

Embedded Mesh Error Estimation with Functional Correction: Final Adaptation Cycle



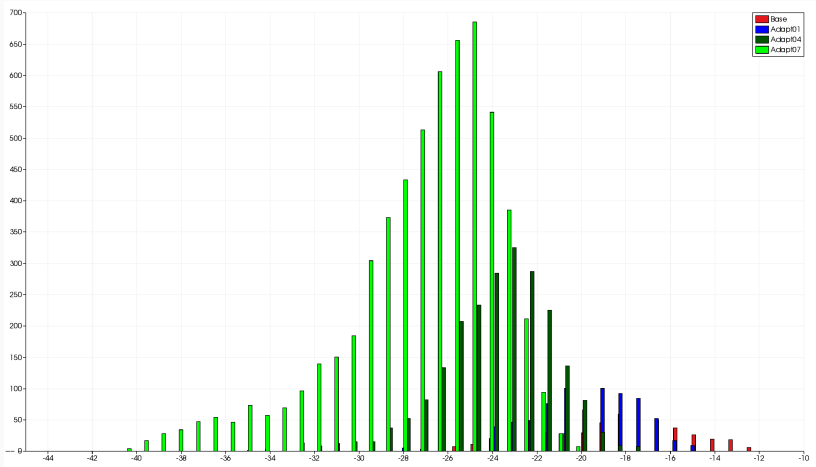
(a) Mesh in full domain



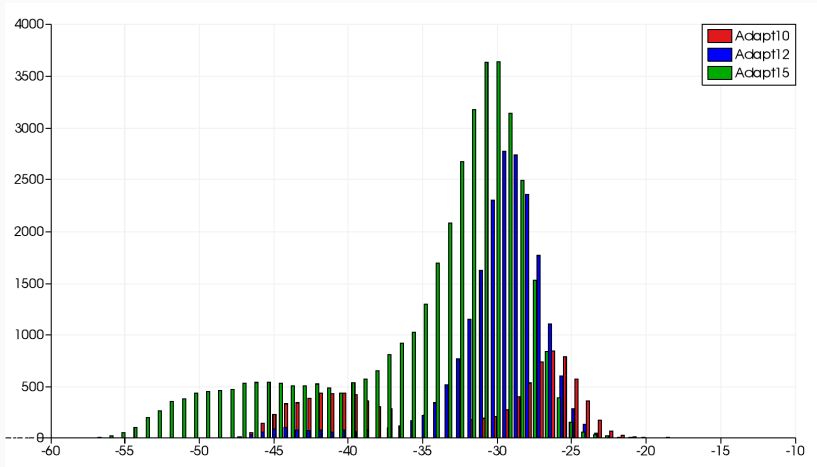
(b) Mach number in full domain

18th and final adaptation cycle for detached bow shock with error estimation

Embedded Mesh Error Estimation with Functional Correction: Error Histograms



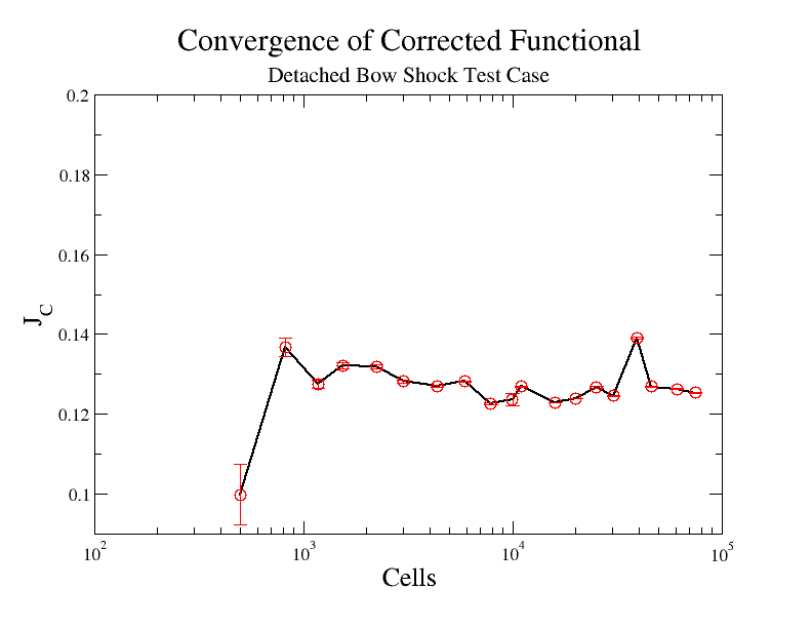
(a) Baseline to seventh adaptation cycle



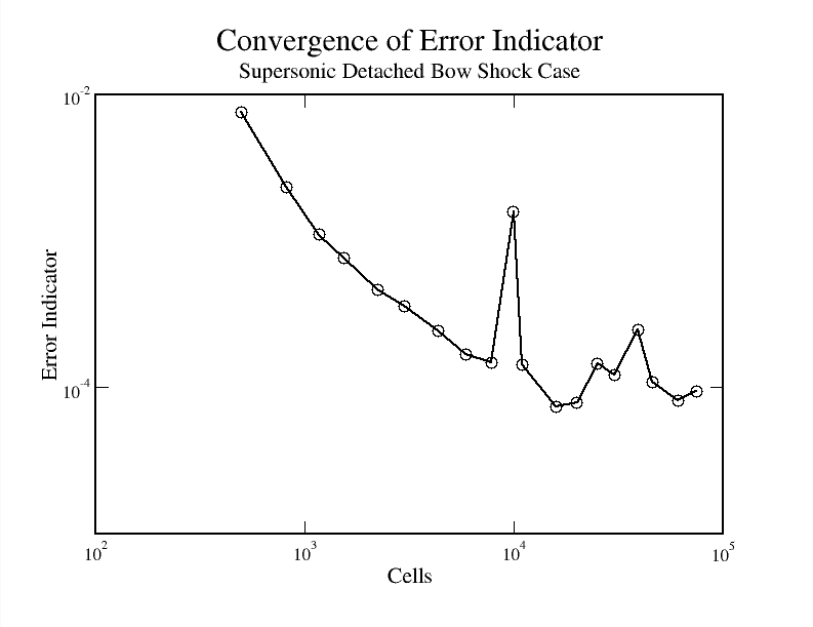
(b) Tenth to 15th adaptation cycle

Error histograms for detached bow shock case

Embedded Mesh Error Estimation with Functional Correction: Refinement Behavior



(a) Corrected functional with error bars



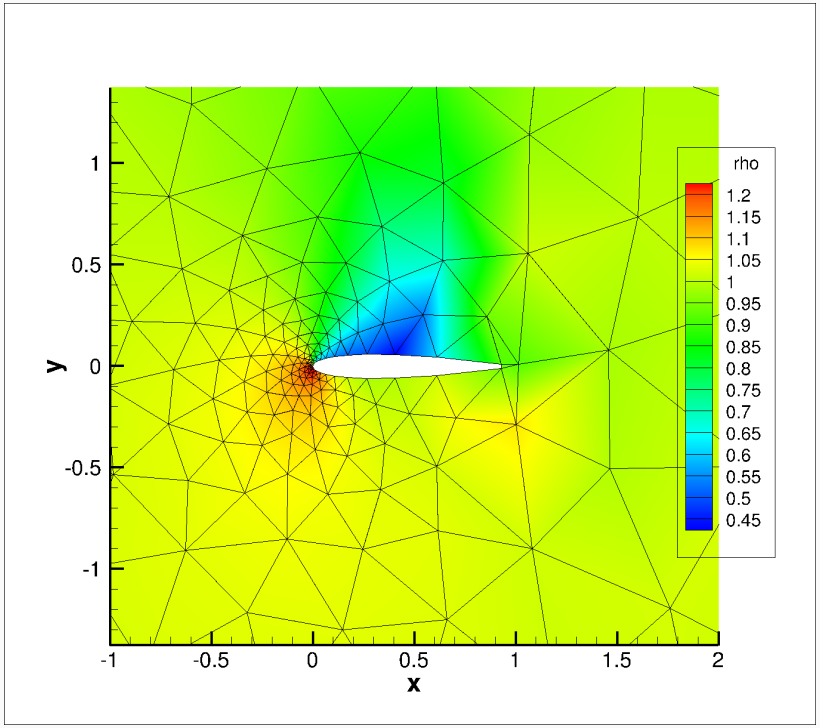
(b) Error convergence

Functional and error estimate convergence for supersonic detached bow shock

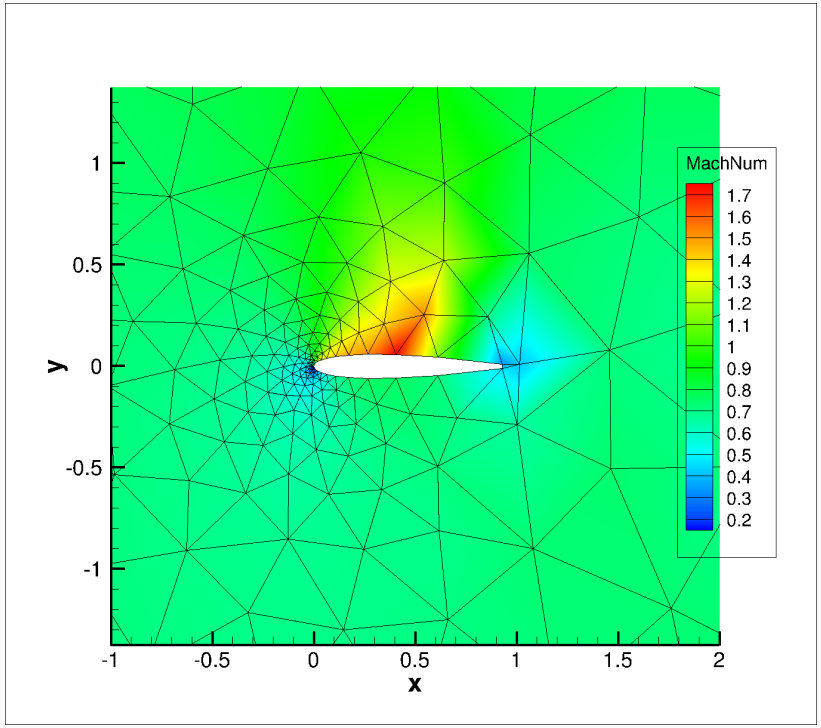
Transonic Blunt Trailing Edge Case

Transonic Blunt Trailing Edge Case: Initial Mesh

NACA0012 truncated at 93% of the chord in $M = 0.75$, $\alpha = 5^o$ flow:



(a) Coarse Mesh Density

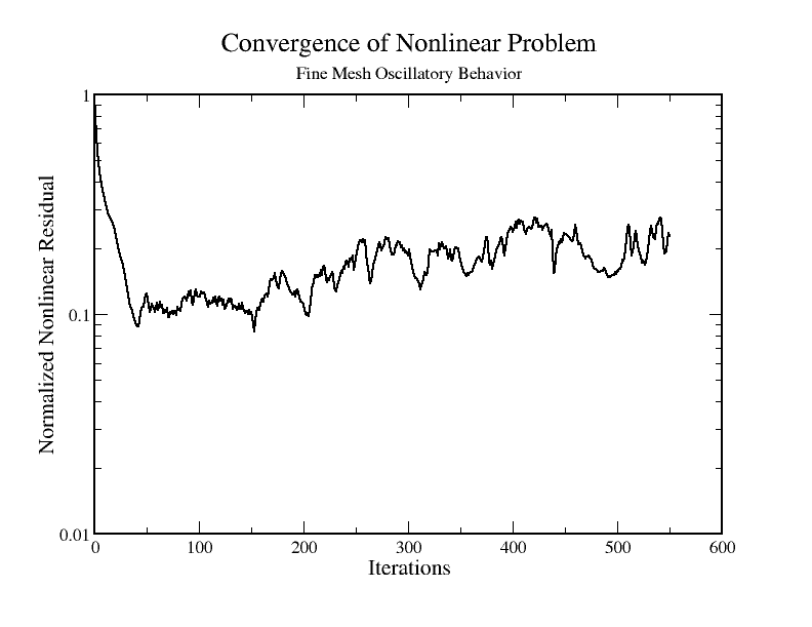


(b) Coarse Mesh Mach Number

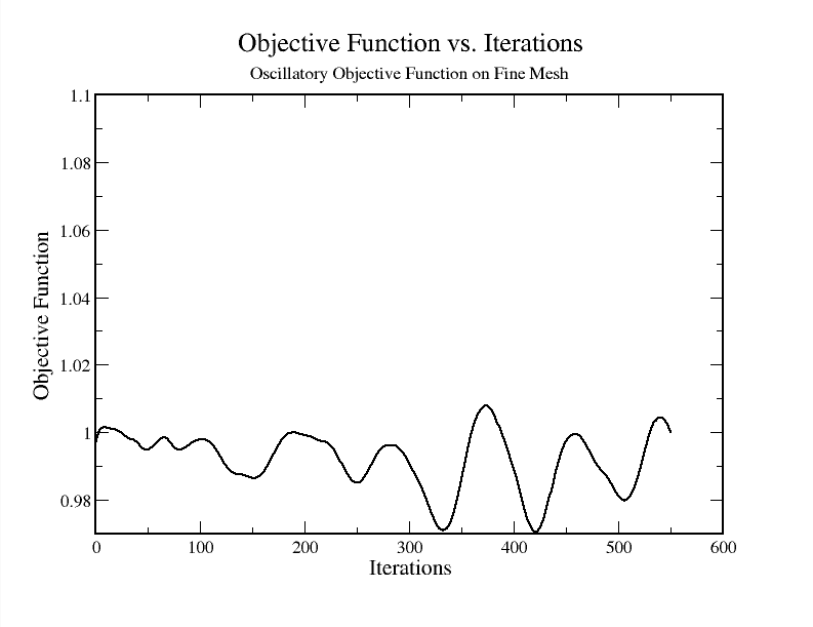
Coarse mesh for detached bow shock error estimation case

Quantity of interest is lift and drag

Transonic Blunt Trailing Edge: Fine Mesh Convergence and Objective



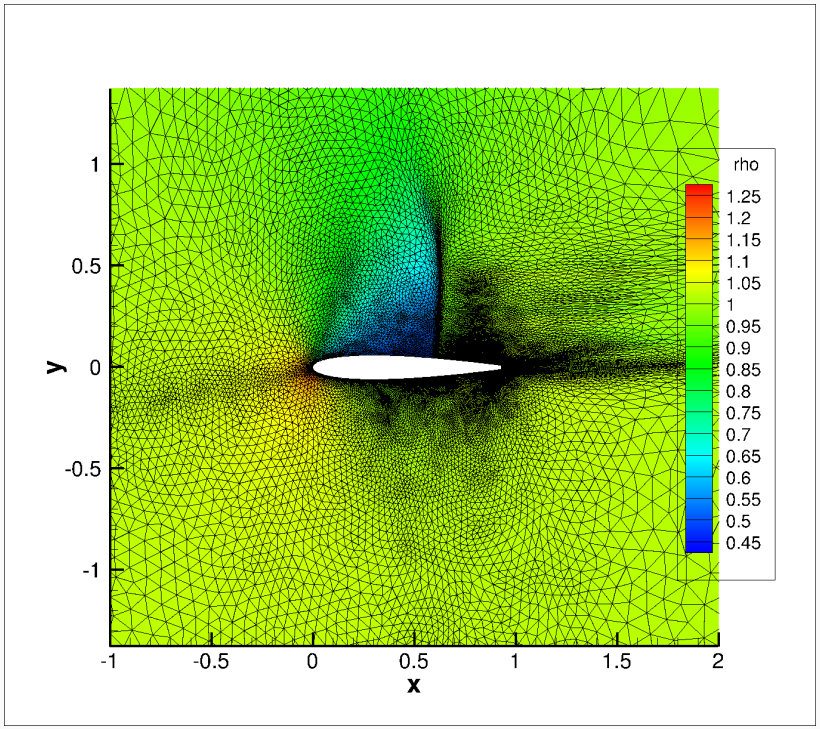
(a) Fine Mesh Convergence



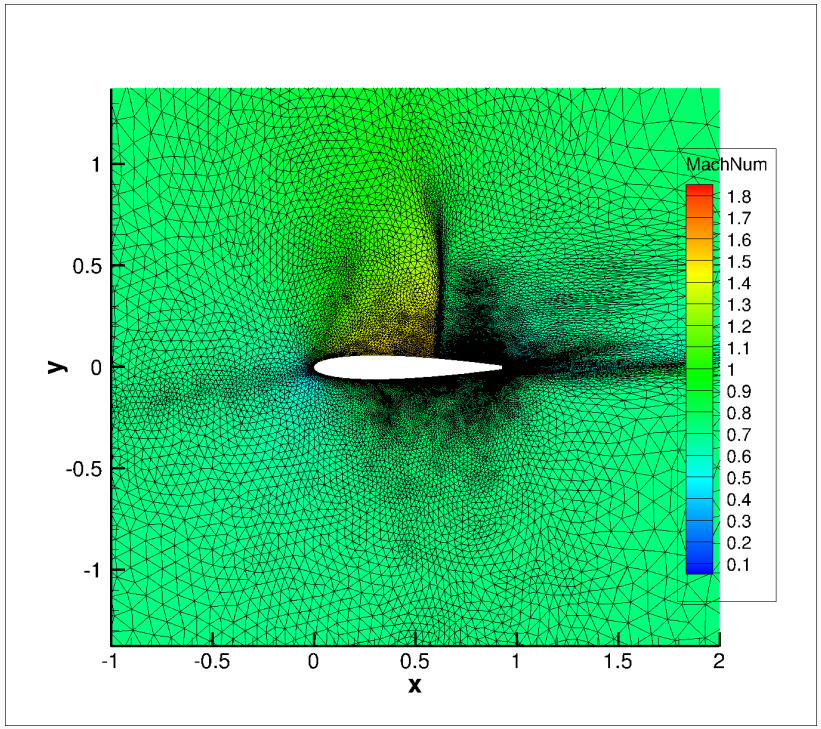
(b) Fine Mesh Objective

Objective and convergence behavior on a fine mesh

Embedded Mesh Error Estimation with Functional Correction: Final Adaptation Cycle



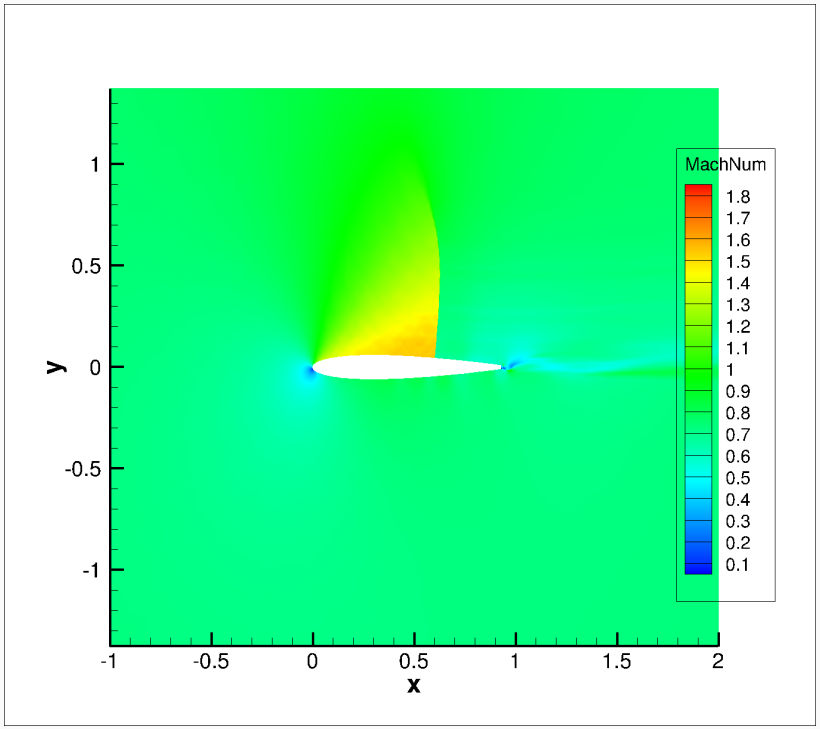
(a) Density with mesh



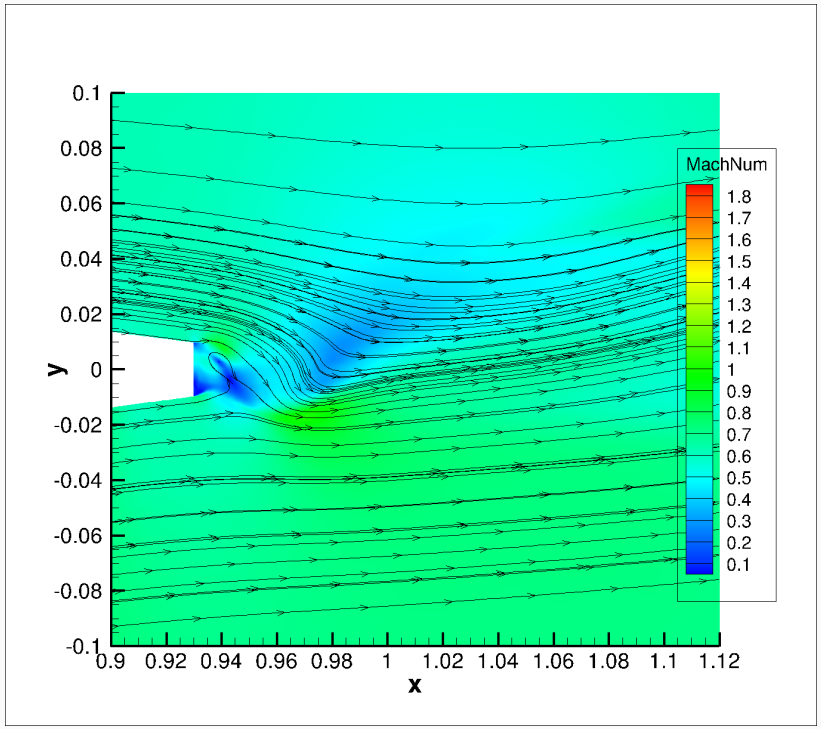
(b) Mach number with mesh

16th and final adaptation cycle for transonic blunt trailing edge with error estimation

Embedded Mesh Error Estimation with Functional Correction: Final Adaptation Cycle



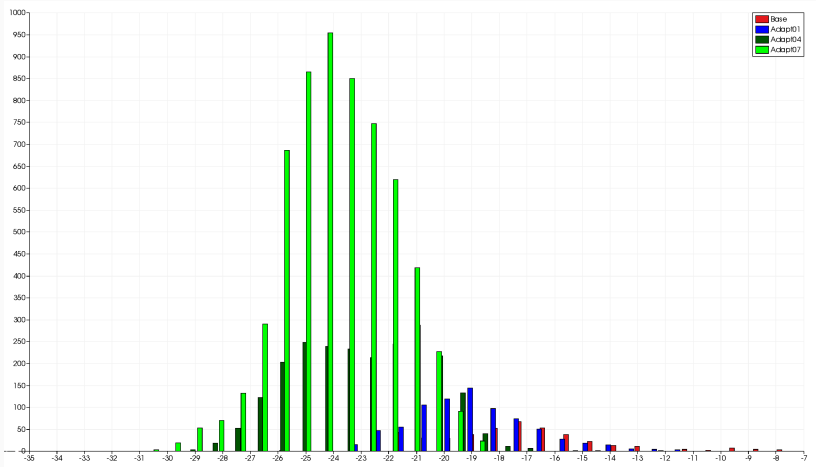
(a) Mach number without mesh



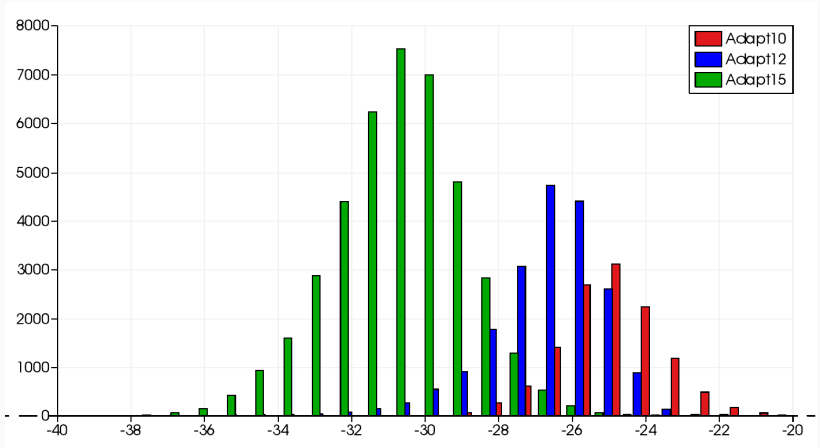
(b) Mach number with streamlines

16th and final adaptation cycle for transonic blunt trailing edge with error estimation

Embedded Mesh Error Estimation with Functional Correction: Error Histograms



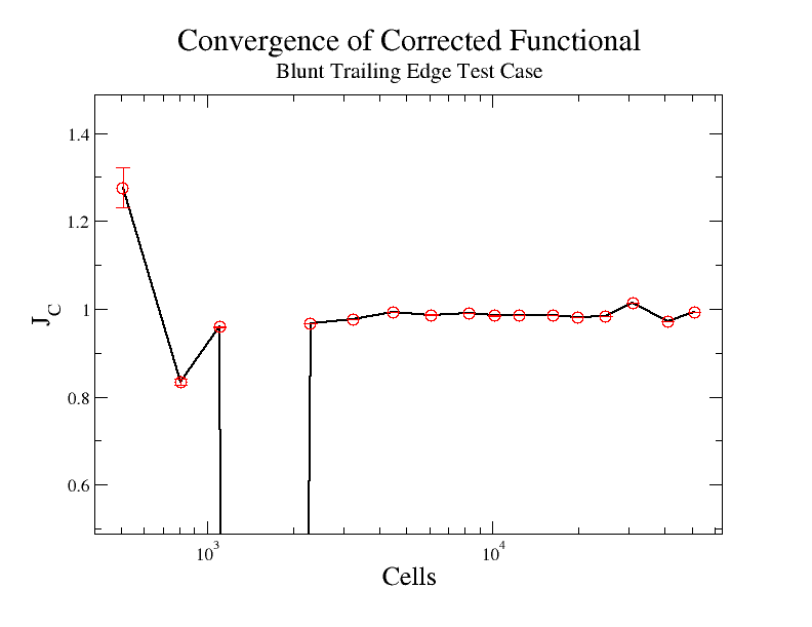
(a) Baseline to seventh adaptation cycle



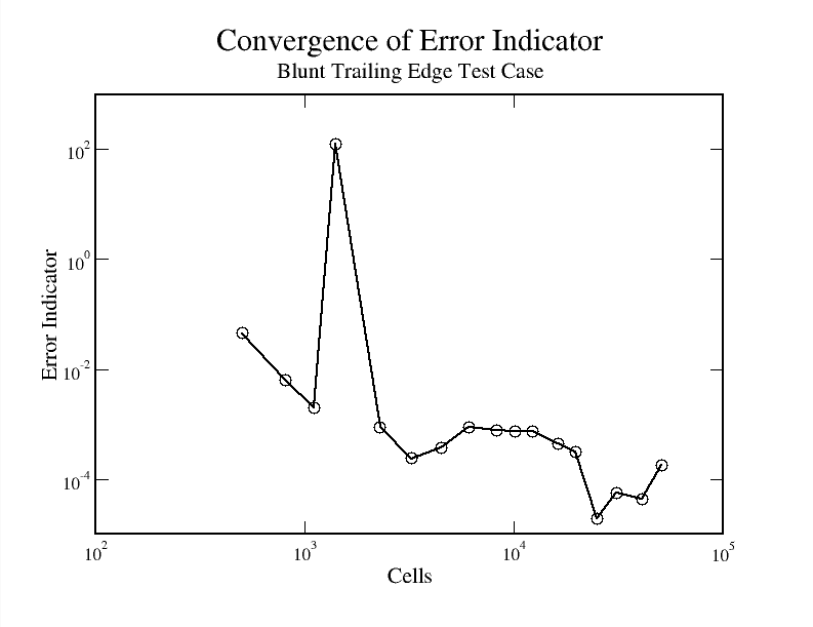
(b) Tenth to 15th adaptation cycle

Error histograms for transonic blunt trailing edge case

Embedded Mesh Error Estimation with Functional Correction: Refinement Behavior



(a) Corrected functional with error bars



(b) Error convergence

Functional and error estimate convergence for transonic blunt trailing edge

Application of the PTA Adjoint to Error Estimation: Summary

- The error estimate appears to appropriately refine the mesh in areas of high importance to the objective.

Application of the PTA Adjoint to Error Estimation: Summary

- The error estimate appears to appropriately refine the mesh in areas of high importance to the objective.
- The small-scale unsteadiness case shows consistent error estimates even on the finest meshes.

Application of the PTA Adjoint to Error Estimation: Summary

- The error estimate appears to appropriately refine the mesh in areas of high importance to the objective.
- The small-scale unsteadiness case shows consistent error estimates even on the finest meshes.
- The large-scale unsteadiness case shows overall consistent error estimates, but these could be improved by better interpolation methods.

Application of the PTA Adjoint to Error Estimation: Summary

- The error estimate appears to appropriately refine the mesh in areas of high importance to the objective.
- The small-scale unsteadiness case shows consistent error estimates even on the finest meshes.
- The large-scale unsteadiness case shows overall consistent error estimates, but these could be improved by better interpolation methods.
- Use of windowing regularization techniques could help with the functional convergence as well as we use a simple averaging approach that does not guarantee convergence.

Application of the PTA Adjoint to Error Estimation: Summary

- The error estimate appears to appropriately refine the mesh in areas of high importance to the objective.
- The small-scale unsteadiness case shows consistent error estimates even on the finest meshes.
- The large-scale unsteadiness case shows overall consistent error estimates, but these could be improved by better interpolation methods.
- Use of windowing regularization techniques could help with the functional convergence as well as we use a simple averaging approach that does not guarantee convergence.
- It would be interesting to pair this work with error estimates that capture error due to partial convergence of the nonlinear problem.