Tangent and Adjoint Problems in Partially Converged Flows

Emmett Padway

August 21st, 2019

University of Wyoming CFD Lab

Outline

- Motivation/introduction

Outline

- Motivation/introduction
- Explanation of steady-state adjoint and tangent problems

Outline

- Motivation/introduction
- Explanation of steady-state adjoint and tangent problems
- Derivation and investigation of the pseudo-time accurate adjoint and tangent problems

Outline

- Motivation/introduction
- Explanation of steady-state adjoint and tangent problems
- Derivation and investigation of the pseudo-time accurate adjoint and tangent problems
    - Derivation, verification, and investigation of Forward Euler

Outline

- Motivation/introduction
- Explanation of steady-state adjoint and tangent problems
- Derivation and investigation of the pseudo-time accurate adjoint and tangent problems
  - Derivation, verification, and investigation of Forward Euler
  - Derivation, verification, and investigation for Newton's Method

Outline

- Motivation/introduction
- Explanation of steady-state adjoint and tangent problems
- Derivation and investigation of the pseudo-time accurate adjoint and tangent problems
  - Derivation, verification, and investigation of Forward Euler
  - Derivation, verification, and investigation for Newton's Method
- Application of pseudo-time accurate approach to a well converging but truncated simulation

Outline

- Motivation/introduction
- Explanation of steady-state adjoint and tangent problems
- Derivation and investigation of the pseudo-time accurate adjoint and tangent problems
  - Derivation, verification, and investigation of Forward Euler
  - Derivation, verification, and investigation for Newton's Method
- Application of pseudo-time accurate approach to a well converging but truncated simulation
- Application of pseudo-time accurate approach to a non-converging primal problem in transonic flow

Outline

- Motivation/introduction
- Explanation of steady-state adjoint and tangent problems
- Derivation and investigation of the pseudo-time accurate adjoint and tangent problems
  - Derivation, verification, and investigation of Forward Euler
  - Derivation, verification, and investigation for Newton's Method
- Application of pseudo-time accurate approach to a well converging but truncated simulation
- Application of pseudo-time accurate approach to a non-converging primal problem in transonic flow
- Conclusions/future work

Motivation/Introduction

Motivation/Introduction

- The adjoint system is well suited to design optimization and adaptive mesh refinement.

Motivation/Introduction

- The adjoint system is well suited to design optimization and adaptive mesh refinement.
- The adjoint and tangent problems are derived from $R = 0$.

Motivation/Introduction

- The adjoint system is well suited to design optimization and adaptive mesh refinement.
- The adjoint and tangent problems are derived from $R = 0$.
- When simulating problems with blunt or complex geometries or higher order spatial discretizations this constraint can be difficult to satisfy.

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

- 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

- 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
- 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

- Volumetric optimization of a CRESCENDO turbine with objective function and constraint

Motivation: Luers et al. 2018

- Volumetric optimization of a CRESCENDO turbine with objective function and constraint
- four order of magnitude residual decrease, and seemingly converged objective function and constraint values

Motivation: Luers et al. 2018

- Volumetric optimization of a CRESCENDO turbine with objective function and constraint
- four order of magnitude residual decrease, and seemingly converged objective function and constraint values
- close qualitative agreement between adjoint and finite-difference computed sensitivities

Motivation: Luers et al. 2018

- Volumetric optimization of a CRESCENDO turbine with objective function and constraint
- four order of magnitude residual decrease, and seemingly converged objective function and constraint values
- close qualitative agreement between adjoint and finite-difference computed sensitivities
- better final design from finite-difference computed sensitivities than the adjoint-computed ones

Introduction: Pseudo-Time Accurate Approach

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

Introduction: Pseudo-Time Accurate Approach

- We employ a pseudo-time accurate approach that applies the unsteady adjoint to the steady state problem.
- In the tangent problem we take the derivative of every step used to solve the primal to march forward in pseudo-time

Introduction: Pseudo-Time Accurate Approach

- We employ a pseudo-time accurate approach that applies the unsteady adjoint to the steady state problem.
- In the tangent problem we take the derivative of every step used to solve the primal to march forward in pseudo-time
- In the adjoint problem, we transpose those derivatives to march back in pseudo-time.

Introduction: Pseudo-Time Accurate Approach

- We employ a pseudo-time accurate approach that applies the unsteady adjoint to the steady state problem.
- In the tangent problem we take the derivative of every step used to solve the primal 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.

Introduction: Pseudo-Time Accurate Approach

- We employ a pseudo-time accurate approach that applies the unsteady adjoint to the steady state problem.
- In the tangent problem we take the derivative of every step used to solve the primal 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.

Introduction: Pseudo-Time Accurate Approach

- We employ a pseudo-time accurate approach that applies the unsteady adjoint to the steady state problem.
- In the tangent problem we take the derivative of every step used to solve the primal 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.
    - These are not chaotic flows, as is shown in the results section the sensitivities do not diverge.

Steady-State Tangent and Adjoint Problems

Steady-State Tangent System

We begin with an objective functional $L(u(D), x(D))$, whose derivative is the sensitivity equation and is expressed as:

$$\frac{dL}{dD} = \frac{\partial L}{\partial x}\frac{dx}{dD} + \frac{\partial L}{\partial u}\frac{du}{dD} \tag{1}$$

We can calculate everything except for $\frac{du}{dD}$ by differentiating subroutines in the code. $\frac{du}{dD}$ is obtained by solving the tangent system.

Steady-State Tangent System

To develop the tangent system, we begin with our zero residual constraint:

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

Steady-State Tangent System

To develop the tangent system, we begin with our zero residual constraint:

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

Then we take the total derivative, which must be equal to zero

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

Steady-State Tangent System

To develop the tangent system, we begin with our zero residual constraint:

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

Then we take the total derivative, which must be equal to zero

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

We have a system of linear equations that scales with the number of design variables:

$$\left[\frac{\partial R}{\partial u}\right] \frac{du}{dD_i} = -\left[\frac{\partial R}{\partial x}\right] \frac{dx}{dD_i} \tag{4}$$

The calculated values of $\frac{du}{dD}$ can be substituted into the objective functional derivative to return the sensitivities

Steady-State Adjoint System

Beginning from the objective function, we add a constraint and a lagrange multiplier:

$$J(u(D), x(D)) = L(u(D), x(D)) + \Lambda^T R(u(D), x(D)) \tag{5}$$

Steady-State Adjoint System

Beginning from the objective function, we add a constraint and a lagrange multiplier:

$$J(u(D), x(D)) = L(u(D), x(D)) + \Lambda^T R(u(D), x(D)) \tag{5}$$

we can take the derivative of both sides to obtain the sensitivity equation:

$$\frac{dJ}{dD} = \left(\frac{\partial L}{\partial x} + \Lambda^T \frac{\partial R}{\partial x}\right) \frac{dx}{dD} + \left(\frac{\partial L}{\partial u} + \Lambda^T \frac{\partial R}{\partial u}\right) \frac{du}{dD} \tag{6}$$

Steady-State Adjoint System

Beginning from the objective function, we add a constraint and a lagrange multiplier:

$$J(u(D), x(D)) = L(u(D), x(D)) + \Lambda^T R(u(D), x(D)) \tag{5}$$

we can take the derivative of both sides to obtain the sensitivity equation:

$$\frac{dJ}{dD} = \left(\frac{\partial L}{\partial x} + \Lambda^T \frac{\partial R}{\partial x}\right) \frac{dx}{dD} + \left(\frac{\partial L}{\partial u} + \Lambda^T \frac{\partial R}{\partial u}\right) \frac{du}{dD} \tag{6}$$

We choose Λ such that we do not have to calculate $\frac{du}{dD}$:

$$\left[\frac{\partial R}{\partial u}\right]^T \Lambda = -\left[\frac{\partial L}{\partial u}\right]^T \tag{7}$$

Pseudo-Time Accurate Tangent and Adjoint Formulation Derivation and Investigation

Derivation for Forward Euler

PTA Tangent System for Forward Euler (Derivation)

As in the steady-state tangent system we want to solve for $\frac{du}{dD}$. To that end we look at our pseudo-time evolution equation:

$$u^k = u^{k-1} + CFL\Delta t(u^{k-1}(D), x(D))R(u^{k-1}(D), x(D)) \tag{8}$$

where Δt is the local time step.

PTA Tangent System for Forward Euler (Derivation)

As in the steady-state tangent system we want to solve for $\frac{du}{dD}$. To that end we look at our pseudo-time evolution equation:

$$u^k = u^{k-1} + CFL\Delta t(u^{k-1}(D), x(D))R(u^{k-1}(D), x(D)) \tag{8}$$

where Δt is the local time step.
We take the derivative and obtain a pseudo-time evolution equation:

$$\begin{aligned} \frac{du^k}{dD} = \frac{du^{k-1}}{dD} &+ CFL\Delta t \left[\frac{\partial R}{\partial x}\frac{dx}{dD} + \frac{\partial R}{\partial u}\frac{du}{dD}^{k-1} \right] \\ &+ CFL \left[\frac{\partial \Delta t}{\partial x}\frac{dx}{dD} + \frac{\partial \Delta t}{\partial u^{k-1}}\frac{du^{k-1}}{dD} \right] R(u^{k-1}) \end{aligned} \tag{9}$$

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

PTA Tangent System for Forward Euler (Derivation)

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

$$L = L(u^n, u^{n-1}, ..., u^{n-m}, D) \tag{10}$$

PTA Tangent System for Forward Euler (Derivation)

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

$$L = L(u^n, u^{n-1}, ..., u^{n-m}, D) \tag{10}$$

the sensitivity equation is:

$$\frac{dL}{dD} = \frac{\partial L}{\partial x}\frac{dx}{dD} + \frac{\partial L}{\partial u^n}\frac{du^n}{dD} + \frac{\partial L}{\partial u^{n-1}}\frac{du^{n-1}}{dD} + ... + \frac{\partial L}{\partial u^{n-m}}\frac{du^{n-m}}{dD} \tag{11}$$

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

PTA Adjoint System for Forward Euler (Derivation)

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

$$L = L(u^n, u^{n-1}, ..., u^{n-m}, D) \tag{12}$$

PTA Adjoint System for Forward Euler (Derivation)

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

$$L = L(u^n, u^{n-1}, ..., u^{n-m}, D) \tag{12}$$

with the pseudo-time constraint of:

$$G^k(u^k(D), u^{k-1}(D), D) = u^k - u^{k-1} - CFL\Delta t R(u^{k-1}) = 0 \tag{13}$$

PTA Adjoint System for Forward Euler (Derivation)

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

$$L = L(u^n, u^{n-1}, ..., u^{n-m}, D) \tag{12}$$

with the pseudo-time constraint of:

$$G^k(u^k(D), u^{k-1}(D), D) = u^k - u^{k-1} - CFL\Delta t R(u^{k-1}) = 0 \tag{13}$$

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

$$\begin{aligned} J(D, u^n, u^{n-1}, ..., \Lambda^n, \Lambda^{n-1}, ...) = {} & L(u^n, u^{n-1}, ..., u^{n-m}, D) \\ & + \Lambda^{n\,T} G^n(u^n(D), u^{n-1}(D), D) \\ & + \Lambda^{n-1\,T} G^{n-1}(u^{n-1}(D), u^{n-2}(D), D) \\ & + ... \\ & + \Lambda^{1\,T} G^1(u^1(D), u^0(D), D) \end{aligned} \tag{14}$$

PTA Adjoint System for Forward Euler (Derivation)

Taking the derivative of the augmented objective functional yields:

$$
\begin{aligned}
\frac{dJ}{dD} = & \frac{\partial L}{\partial x}\frac{dx}{dD} + \frac{\partial L}{\partial u^n}\frac{du^n}{dD} + \frac{\partial L}{\partial u^{n-1}}\frac{du^{n-1}}{dD} + \ldots + \frac{\partial L}{\partial u^{n-m}}\frac{du^{n-m}}{dD} \\
& + \Lambda^{nT}\left(\frac{\partial G^n}{\partial x}\frac{dx}{dD} + \frac{\partial G^n}{\partial u^n}\frac{du^n}{dD} + \frac{\partial G^n}{\partial u^{n-1}}\frac{du^{n-1}}{dD}\right) \\
& + \Lambda^{n-1T}\left(\frac{\partial G^{n-1}}{\partial x}\frac{dx}{dD} + \frac{\partial G^{n-1}}{\partial u^{n-1}}\frac{du^{n-1}}{dD} + \frac{\partial G^{n-1}}{\partial u^{n-2}}\frac{du^{n-2}}{dD}\right) \\
& + \ldots \\
& + \Lambda^{1T}\left(\frac{\partial G^1}{\partial x}\frac{dx}{dD} + \frac{\partial G^1}{\partial u^1}\frac{du^1}{dD} + \frac{\partial G^1}{\partial u^0}\frac{du^0}{dD}\right)
\end{aligned}
\tag{15}
$$

PTA Adjoint System for Forward Euler (Derivation)

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

$$\frac{\partial L}{\partial u^{k-1}} + \Lambda^{kT} \frac{\partial G^k}{\partial u^{k-1}} + \Lambda^{k-1T} \frac{\partial G^{k-1}}{\partial u^{k-1}} = 0 \tag{16}$$

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

PTA Adjoint System for Forward Euler (Derivation)

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

$$\frac{\partial L}{\partial u^{k-1}} + \Lambda^{kT}\frac{\partial G^k}{\partial u^{k-1}} + \Lambda^{k-1T}\frac{\partial G^{k-1}}{\partial u^{k-1}} = 0 \tag{16}$$

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

We can plug in the constraint derivative to the recurrence relation and obtain:

$$\begin{aligned}[I]\,\Lambda^{k-1} = &-\left[-I - CFL\Delta t^{k-1}\frac{\partial R(u^{k-1})}{\partial u^{k-1}} - CFL\frac{\partial \Delta t}{\partial u^{k-1}}R(u^{k-1})\right]^T \Lambda^k \\ &-\left[\frac{\partial L}{\partial u^{k-1}}\right]^T\end{aligned} \tag{17}$$

PTA Adjoint System for Forward Euler (Derivation)

The sensitivity equation with the terms that vanish due to the choice of adjoint variable is expressed as:

$$\frac{dJ}{dD} = \frac{\partial L}{\partial D} + \Lambda^{n\,T}\frac{\partial G^{n}}{\partial D} + \Lambda^{n-1\,T}\frac{\partial G^{n-1}}{\partial D} + \Lambda^{n-2\,T}\frac{\partial G^{n-2}}{\partial D} + \ldots \tag{18}$$

PTA Adjoint System for Forward Euler (Derivation)

The sensitivity equation with the terms that vanish due to the choice of adjoint variable is expressed as:

$$\frac{dJ}{dD} = \frac{\partial L}{\partial D} + \Lambda^{nT}\frac{\partial G^n}{\partial D} + \Lambda^{n-1T}\frac{\partial G^{n-1}}{\partial D} + \Lambda^{n-2T}\frac{\partial G^{n-2}}{\partial D} + \ldots \tag{18}$$

Plugging in the constraint derivative values we get:

$$\begin{aligned}\frac{dJ}{dD} = \frac{\partial L}{\partial D} &- \Lambda^{nT}\left[CFL\Delta t^{n-1}\frac{\partial R(u^{n-1})}{\partial D} + CFL\frac{\partial \Delta t}{\partial D}R(u^{n-1})\right] \\ &- \Lambda^{n-1T}\left[CFL\Delta t^{n-2}\frac{\partial R(u^{n-2})}{\partial D} + CFL\frac{\partial \Delta t}{\partial D}R(u^{k-2})\right] \\ &- \ldots \\ &- \Lambda^{1T}\left[CFL\Delta t^{0}\frac{\partial R(u^{0})}{\partial D} + CFL\frac{\partial \Delta t}{\partial D}R(u^{0})\right]\end{aligned} \tag{19}$$

PTA Adjoint System for Forward Euler (Derivation)

- Tangent equations march forward in time similarly to the primal problem.

PTA Adjoint System for Forward Euler (Derivation)

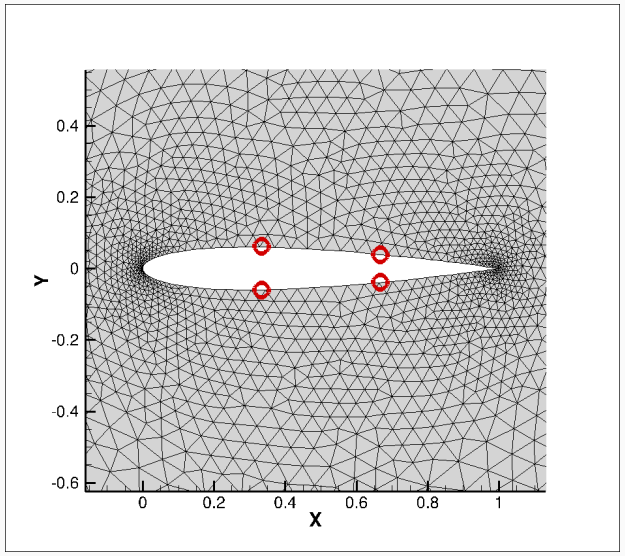
- Tangent equations march forward in time similarly to the primal problem.
- The adjoint equations march backwards in time in the reverse of the primal problem, accumulating the sensitivities through the backwards in time integration.

PTA Adjoint System for Forward Euler (Derivation)

- Tangent equations march forward in time similarly to the primal problem.
- The adjoint equations march backwards in time in the reverse of the primal problem, accumulating the sensitivities through the backwards in time integration.
- Each tangent/adjoint pseudo-time step equation is cheap, and is the result of a few matrix vector multiplications and additions, rather than a full linear system solution.

Verification and Investigation of Forward Euler and RK5

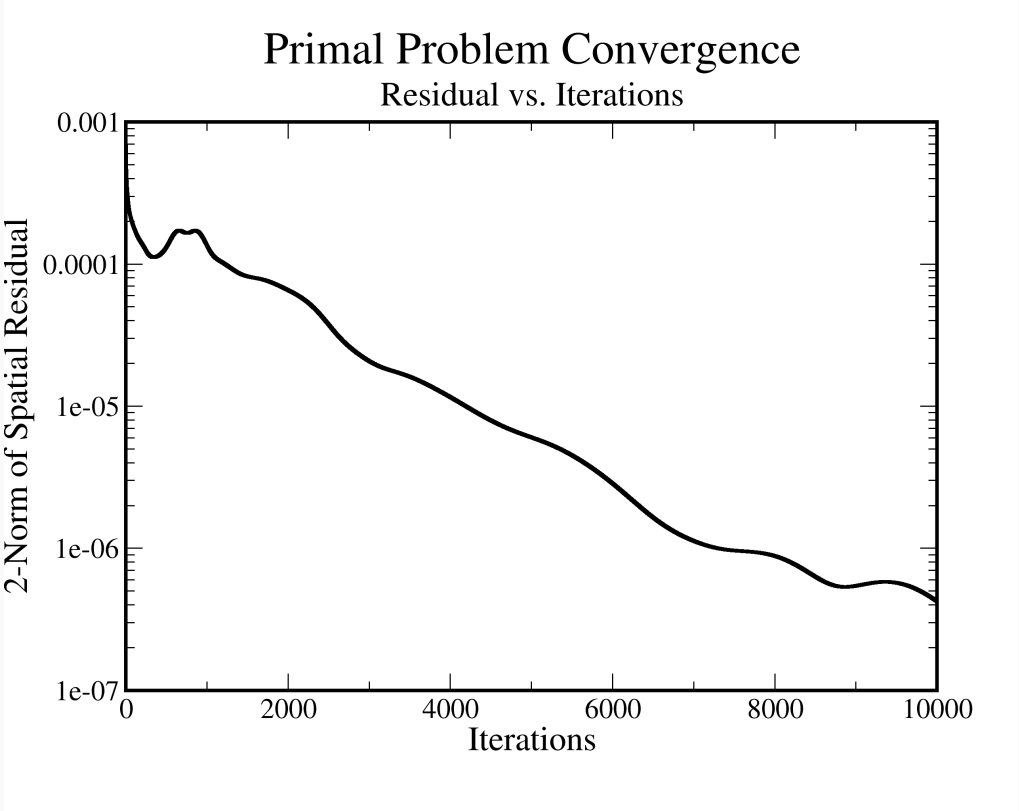
PTA Tangent System for Forward Euler (Verification)



Computational mesh for NACA0012 airfoil

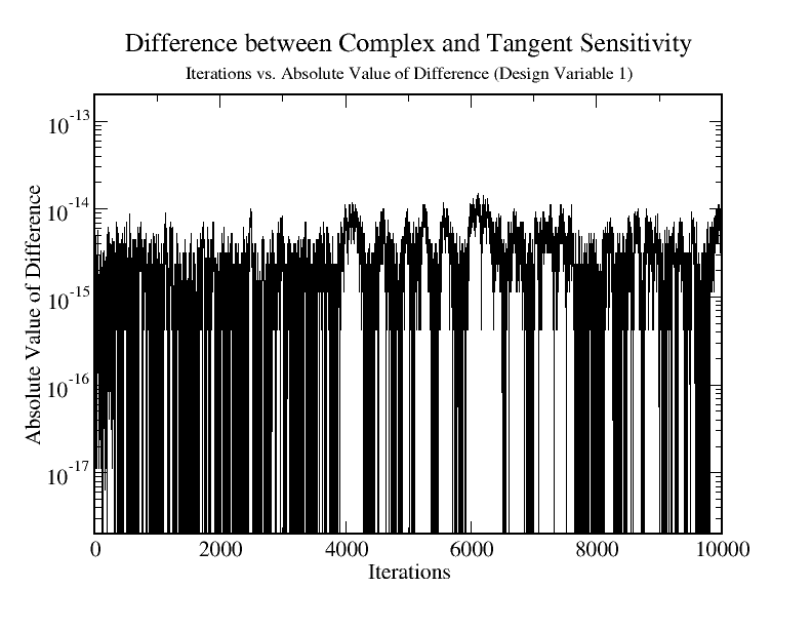
- Design variables are two equally spaced Hicks-Henne bump functions.
- $Mach = .6$, $\alpha = 1^o$
- Spring analogy used for mesh motion and mesh sensitivities
- Objective function is a weighted combination of lift and drag

Tangent System for Forward Euler (Verification)

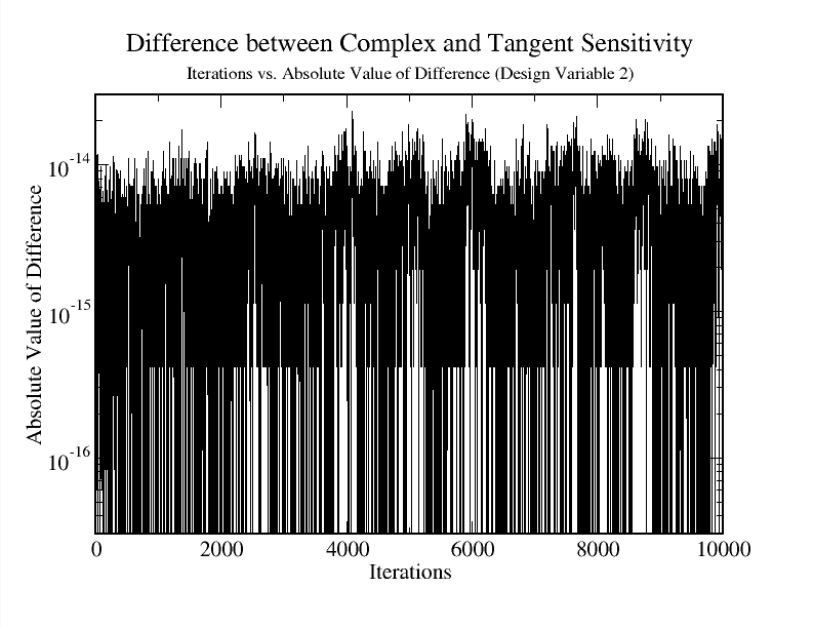


Residual convergence

Tangent System for Forward Euler (Verification)



(a) Design Variable 1



(b) Design Variable 2

Difference between pseudo-time accurate tangent sensitivities and complex sensitivities for forward Euler pseudo-time evolution with linearized time-step

Differences are of the order of machine zero for all iterations in pseudo-time. (Implementation is verified)

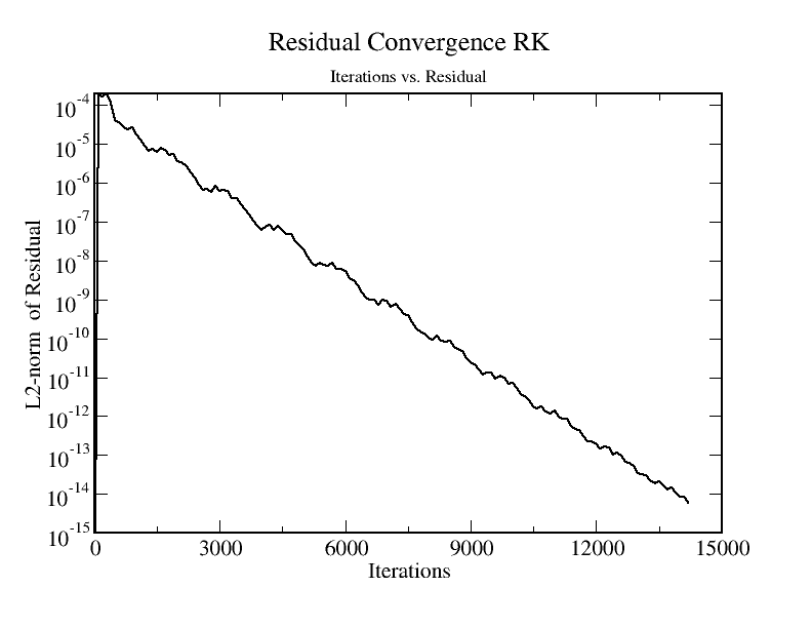
PTA Adjoint System (Verification)

Scheme	Steps	Complex Sensitivity	Adjoint Sensitivity	$\|\|Residual\|\|_2$
RK5	50	-0.6781697638433**92**	-0.6781697638433**77**	0.1984106666029376E-03
RK5	200	0.4319604250392554	0.4319604250392554	0.1329585951240718E-03
RK5	500	3.729554115544452	3.729554115544451	0.1768186072970460E-03

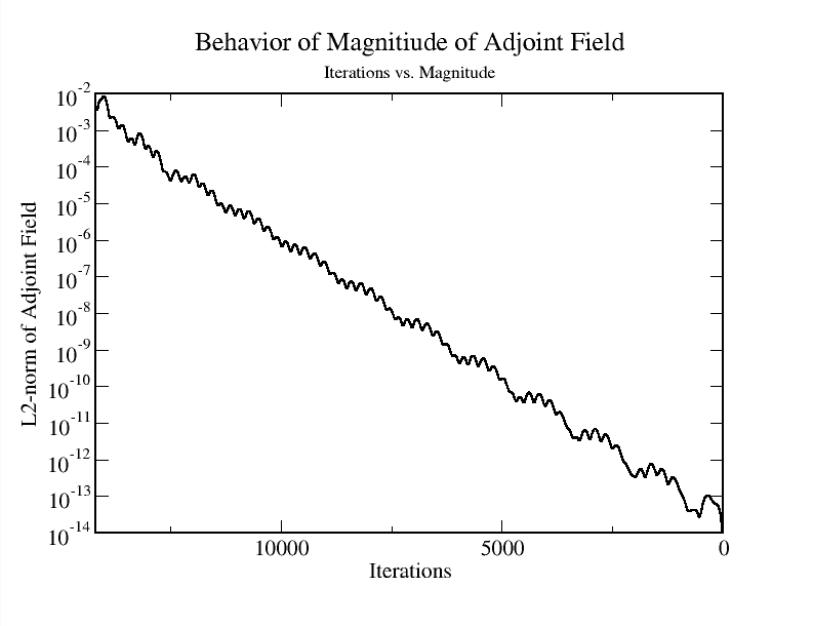
Comparison of adjoint and complex-step computed sensitivities

- Here we show the verification for a low storage five stage explicit RK scheme looking at the first design variable
  - The derivation for the RK5 method can be found in our paper: "Toward a Pseudo-Time Accurate Formulation of the Adjoint and Tangent Systems".

PTA Adjoint System RK5 Investigation



(a) Residual Convergence

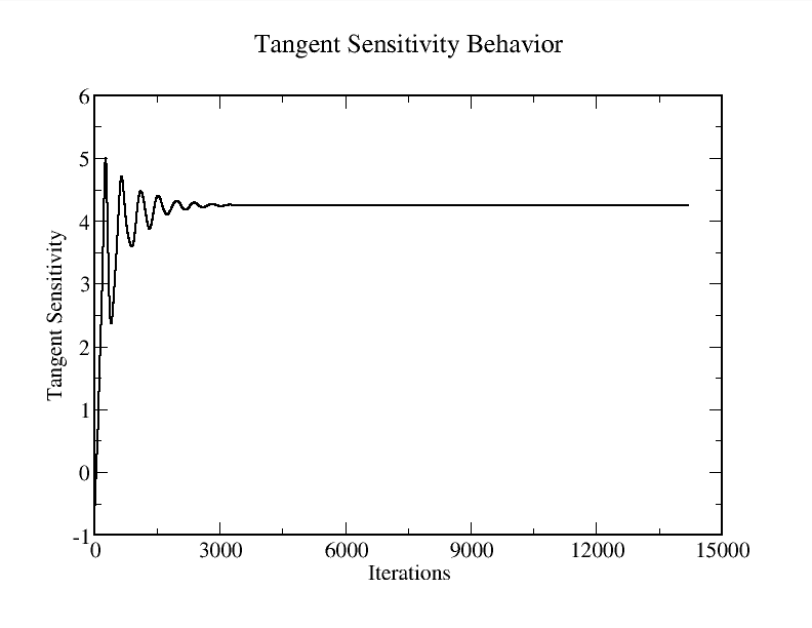


(b) Adjoint Magnitude Behavior

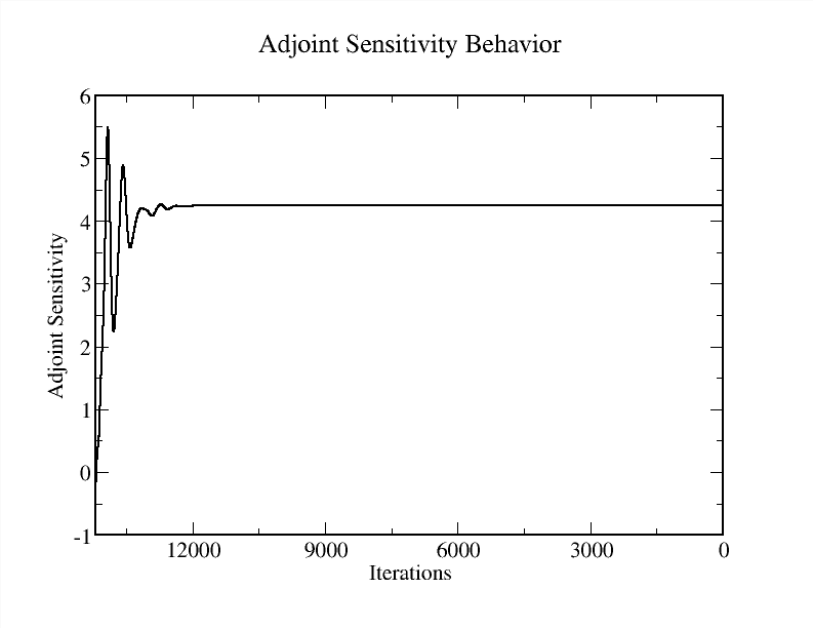
Residual convergence and adjoint magnitude behavior for RK5 scheme

The adjoint at a pseudo-time step relates a perturbation in the constraint of that pseudo-time step to a change in the objective function

PTA Adjoint System RK5 Investigation



(a) Tangent Sensitivity Convergence



(b) Adjoint Sensitivity Convergence

Sensitivity convergence for RK5 scheme

Derivation for Newton's Method

PTA Tangent system for Newton's Method

As previously, we try to solve for $\frac{dU}{dD}$. Looking at our primal solver:

$$u^k = u^{k-1} + [P_{k-1}]^{-1} R(u^{k-1}(D), X(D)) \tag{20}$$

where P_k is defined as:

$$P_k = \frac{\partial R}{\partial u} + M \tag{21}$$

Where $\frac{\partial R}{\partial u}$ is an approximation to the jacobian and M is a suitable mass matrix:

$$M = \frac{vol}{\Delta t} \tag{22}$$

PTA Tangent system for Newton's Method

Taking the derivative of the nonlinear solver gives:

$$\frac{du^k}{dD} = \frac{du^{k-1}}{dD} + \frac{d\left[P_{k-1}\right]^{-1} R(u^{k-1}, D)}{dD} \tag{23}$$

PTA Tangent system for Newton's Method

Taking the derivative of the nonlinear solver gives:

$$\frac{du^k}{dD} = \frac{du^{k-1}}{dD} + \frac{d\left[P_{k-1}\right]^{-1} R(u^{k-1}, D)}{dD} \tag{23}$$

If a Newton-Chord method is used, the derivative of the augmented jacobian inverse can be neglected:

$$\frac{du^k}{dD} = \frac{du^{k-1}}{dD} + \left[P_{k-1}\right]^{-1} \left[\frac{dR(u^{k-1}, D)}{dD}\right] \tag{24}$$

PTA Tangent system for Newton's Method

Taking the derivative of the nonlinear solver gives:

$$\frac{du^k}{dD} = \frac{du^{k-1}}{dD} + \frac{d\left[P_{k-1}\right]^{-1} R(u^{k-1}, D)}{dD} \tag{23}$$

If a Newton-Chord method is used, the derivative of the augmented jacobian inverse can be neglected:

$$\frac{du^k}{dD} = \frac{du^{k-1}}{dD} + \left[P_{k-1}\right]^{-1} \left[\frac{dR(u^{k-1}, D)}{dD}\right] \tag{24}$$

Otherwise, the derivative must be computed:

$$\begin{aligned} \frac{du^k}{dD} = \frac{du^{k-1}}{dD} &+ \left[P_{k-1}\right]^{-1} \left[\frac{dR(u^{k-1}, D)}{dD}\right] \\ &+ \frac{d\left[P_{k-1}\right]^{-1}}{dD} R(u^{k-1}, D) \end{aligned} \tag{25}$$

PTA Tangent system for Newton's Method

- fully linearizing the linear system solve is not feasible, especially for more complicated linear solvers, e.g. GMRES

PTA Tangent system for Newton's Method

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

PTA Tangent system for Newton's Method

- fully linearizing the linear system solve is not feasible, especially for more complicated linear solvers, e.g. GMRES
- There are two ways to tackle the linearization of the matrix inverse
- the first way is to use complex perturbations in the conservative variable and the nodal coordinate vectors to obtain the frechet derivative
- the second way is to use an identity for the derivative of the matrix inverse,this will not be exact, 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

Newton-Chord:

$$\frac{du^k}{dD} = \frac{du^{k-1}}{dD} + [P_{k-1}]^{-1} \left[\frac{\partial R}{\partial u^{k-1}} \frac{du^{k-1}}{dD} + \left[\frac{\partial R}{\partial x} \right] \frac{dx}{dD} \right] \tag{26}$$

PTA Tangent system for Newton's Method

Newton-Chord:

$$\frac{du^k}{dD} = \frac{du^{k-1}}{dD} + [P_{k-1}]^{-1} \left[\frac{\partial R}{\partial u^{k-1}} \frac{du^{k-1}}{dD} + \left[\frac{\partial R}{\partial x} \right] \frac{dx}{dD} \right] \tag{26}$$

Inverse Identity:

$$\begin{aligned} \frac{du^k}{dD} = \frac{du^{k-1}}{dD} &+ [P_{k-1}]^{-1} \left[\frac{\partial R}{\partial u^{k-1}} \frac{du^{k-1}}{dD} + \left[\frac{\partial R}{\partial x} \right] \frac{dx}{dD} \right] \\ &- [P_{k-1}]^{-1} \frac{d\,[P_{k-1}]}{dD} [P_{k-1}]^{-1} R(u^{k-1}, D) \end{aligned} \tag{27}$$

PTA Tangent system for Newton's Method

Newton-Chord:

$$\frac{du^k}{dD} = \frac{du^{k-1}}{dD} + [P_{k-1}]^{-1} \left[\frac{\partial R}{\partial u^{k-1}} \frac{du^{k-1}}{dD} + \left[\frac{\partial R}{\partial x} \right] \frac{dx}{dD} \right] \tag{26}$$

Inverse Identity:

$$\begin{aligned} \frac{du^k}{dD} = \frac{du^{k-1}}{dD} &+ [P_{k-1}]^{-1} \left[\frac{\partial R}{\partial u^{k-1}} \frac{du^{k-1}}{dD} + \left[\frac{\partial R}{\partial x} \right] \frac{dx}{dD} \right] \\ &- [P_{k-1}]^{-1} \left[\frac{\partial P_{k-1}}{\partial u^{k-1}} \frac{du^{k-1}}{dD} + \frac{\partial P_{k-1}}{\partial x} \frac{dx}{dD} \right] \Delta u \end{aligned} \tag{27}$$

PTA Adjoint System for Newton's Method (Derivation)

As in the forward euler adjoint there is a series of adjoint recurrence relations:

$$\frac{\partial L}{\partial u^{k-1}} + \Lambda^{kT} \frac{\partial G^k}{\partial u^{k-1}} + \Lambda^{k-1T} \frac{\partial G^{k-1}}{\partial u^{k-1}} = 0 \tag{28}$$

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

PTA Adjoint System for Newton's Method (Derivation)

As in the forward euler adjoint there is a series of adjoint recurrence relations:

$$\frac{\partial L}{\partial u^{k-1}} + \Lambda^{kT} \frac{\partial G^k}{\partial u^{k-1}} + \Lambda^{k-1T} \frac{\partial G^{k-1}}{\partial u^{k-1}} = 0 \tag{28}$$

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

Plugging in the constraint derivatives from Newton's method:

$$\begin{aligned} [I]\,\Lambda^{k-1} = &- \left[-I - \frac{\partial \left[P_{k-1}\right]^{-1} R(u^{k-1})}{\partial u^{k-1}} \right]^T \Lambda^k \\ &- \left[\frac{\partial L}{\partial u^{k-1}} \right]^T \end{aligned} \tag{29}$$

PTA Adjoint System for Newton's Method (Derivation)

Newton-Chord:

$$[I]\,\Lambda^{k-1} = -\left[-I - [P_{k-1}]^{-1}\,\frac{\partial R(u^{k-1})}{\partial u^{k-1}}\right]^T \Lambda^k - \left[\frac{\partial L}{\partial u^{k-1}}\right]^T \tag{30}$$

PTA Adjoint System for Newton's Method (Derivation)

Newton-Chord:

$$[I]\,\Lambda^{k-1} = -\left[-I - [P_{k-1}]^{-1}\,\frac{\partial R(u^{k-1})}{\partial u^{k-1}}\right]^T \Lambda^k - \left[\frac{\partial L}{\partial u^{k-1}}\right]^T \tag{30}$$

Inverse identity:

$$\begin{aligned}[I]\,\Lambda^{k-1} =& -\left[-I - [P_{k-1}]^{-1}\,\frac{\partial R(u^{k-1})}{\partial u^{k-1}} - \frac{\partial\,[P_{k-1}]^{-1}}{\partial u^{k-1}} R(u^{k-1})\right]^T \Lambda^k \\ & - \left[\frac{\partial L}{\partial u^{k-1}}\right]^T\end{aligned} \tag{31}$$

PTA Adjoint System for Newton's Method (Derivation)

Newton-Chord:

$$[I]\,\Lambda^{k-1} = -\left[-I - [P_{k-1}]^{-1}\,\frac{\partial R(u^{k-1})}{\partial u^{k-1}}\right]^T \Lambda^k - \left[\frac{\partial L}{\partial u^{k-1}}\right]^T \tag{30}$$

Inverse identity:

$$\begin{aligned}[I]\,\Lambda^{k-1} =& -\left[-I - [P_{k-1}]^{-1}\,\frac{\partial R(u^{k-1})}{\partial u^{k-1}} + [P_{k-1}]^{-1}\,\frac{\partial P_{k-1}}{\partial u^{k-1}}\Delta u\right]^T \Lambda^k \\ & - \left[\frac{\partial L}{\partial u^{k-1}}\right]^T \end{aligned} \tag{31}$$

PTA Adjoint System for Newton's Method (Derivation)

Define a secondary adjoint variable:

$$[P_{k-1}]^T \psi^k = \Lambda^k \tag{32}$$

PTA Adjoint System for Newton's Method (Derivation)

Define a secondary adjoint variable:

$$[P_{k-1}]^T \psi^k = \Lambda^k \tag{32}$$

Newton-Chord:

$$\Delta\Lambda^{k-1} = \left[\frac{\partial R(u^{k-1})}{\partial u^{k-1}}\right]^T \psi^k - \left[\frac{\partial L}{\partial u^{k-1}}\right]^T \tag{33}$$

PTA Adjoint System for Newton's Method (Derivation)

Define a secondary adjoint variable:

$$[P_{k-1}]^T \psi^k = \Lambda^k \tag{32}$$

Newton-Chord:

$$\Delta\Lambda^{k-1} = \left[\frac{\partial R(u^{k-1})}{\partial u^{k-1}}\right]^T \psi^k - \left[\frac{\partial L}{\partial u^{k-1}}\right]^T \tag{33}$$

Inverse identity:

$$\Delta\Lambda^{k-1} = \left[\frac{\partial R(u^{k-1})}{\partial u^{k-1}} - \frac{\partial P_{k-1}}{\partial u^{k-1}}\Delta u\right]^T \psi^k - \left[\frac{\partial L}{\partial u^{k-1}}\right]^T \tag{34}$$

PTA Adjoint System for Newton's Method (Derivation)

Beginning with the constraint based expression of the sensitivity equation as we did for forward euler:

$$\frac{dJ}{dD} = \frac{\partial L}{\partial D} + \Lambda^{nT}\frac{\partial G^n}{\partial D} + \Lambda^{n-1T}\frac{\partial G^{n-1}}{\partial D} + \Lambda^{n-2T}\frac{\partial G^{n-2}}{\partial D} + \ldots \tag{35}$$

PTA Adjoint System for Newton's Method (Derivation)

Beginning with the constraint based expression of the sensitivity equation as we did for forward euler:

$$\frac{dJ}{dD} = \frac{\partial L}{\partial D} + \Lambda^{nT}\frac{\partial G^n}{\partial D} + \Lambda^{n-1T}\frac{\partial G^{n-1}}{\partial D} + \Lambda^{n-2T}\frac{\partial G^{n-2}}{\partial D} + \ldots \tag{35}$$

Plugging in the constraint derivative values for the Newton-Chord method:

$$\begin{aligned}\frac{dJ}{dD} = \frac{\partial L}{\partial D} &- \Lambda^{nT}\left[[P_{n-1}]^{-1}\frac{\partial R(u^{n-1})}{\partial D}\right] \\ &- \Lambda^{n-1T}\left[[P_{n-2}]^{-1}\frac{\partial R(u^{n-2})}{\partial D}\right] \\ &- \ldots \\ &- \Lambda^{1T}\left[[P_0]^{-1}\frac{\partial R(u^0)}{\partial D}\right]\end{aligned} \tag{36}$$

PTA Adjoint System for Newton's Method (Derivation)

Beginning with the constraint based expression of the sensitivity equation as we did for forward euler:

$$\frac{dJ}{dD} = \frac{\partial L}{\partial D} + \Lambda^{nT}\frac{\partial G^n}{\partial D} + \Lambda^{n-1T}\frac{\partial G^{n-1}}{\partial D} + \Lambda^{n-2T}\frac{\partial G^{n-2}}{\partial D} + \ldots \tag{35}$$

Plugging in the constraint derivative values for the Newton-Chord method:

$$\begin{aligned}\frac{dJ}{dD} = \frac{\partial L}{\partial D} &- \psi^{nT}\left[\frac{\partial R(u^{n-1})}{\partial D}\right] \\ &- \psi^{n-1T}\left[\frac{\partial R(u^{n-2})}{\partial D}\right] \\ &- \ldots \\ &- \psi^{1T}\left[\frac{\partial R(u^{0})}{\partial D}\right]\end{aligned} \tag{36}$$

PTA Adjoint System for Newton's Method (Derivation)

Referencing the constraint based expression of the sensitivity equation as we did for forward euler:

$$\frac{dJ}{dD} = \frac{\partial L}{\partial D} + \Lambda^{nT}\frac{\partial G^n}{\partial D} + \Lambda^{n-1T}\frac{\partial G^{n-1}}{\partial D} + \Lambda^{n-2T}\frac{\partial G^{n-2}}{\partial D} + \ldots \tag{37}$$

Plugging in the constraint derivative values using the inverse identity:

$$\begin{aligned}\frac{dJ}{dD} = \frac{\partial L}{\partial D} &- \Lambda^{nT}\left[[P_{n-1}]^{-1}\frac{\partial R(u^{n-1})}{\partial D} - [P_{n-1}]^{-1}\frac{\partial [P_{n-1}]}{\partial D}\Delta u^{n-1}\right] \\ &- \Lambda^{n-1T}\left[[P_{n-2}]^{-1}\frac{\partial R(u^{n-2})}{\partial D} - [P_{n-2}]^{-1}\frac{\partial [P_{n-2}]}{\partial D}\Delta u^{n-2}\right] \\ &- \ldots \\ &- \Lambda^{1T}\left[[P_0]^{-1}\frac{\partial R(u^0)}{\partial D} - [P_0]^{-1}\frac{\partial [P_0]}{\partial D}\Delta u^0\right]\end{aligned} \tag{38}$$

PTA Adjoint System for Newton's Method (Derivation)

Referencing the constraint based expression of the sensitivity equation as we did for forward euler:

$$\frac{dJ}{dD} = \frac{\partial L}{\partial D} + \Lambda^{nT}\frac{\partial G^n}{\partial D} + \Lambda^{n-1T}\frac{\partial G^{n-1}}{\partial D} + \Lambda^{n-2T}\frac{\partial G^{n-2}}{\partial D} + \ldots \tag{37}$$

Plugging in the constraint derivative values using the inverse identity:

$$\begin{aligned}\frac{dJ}{dD} = \frac{\partial L}{\partial D} &-\psi^{nT}\left[\frac{\partial R(u^{n-1})}{\partial D} - \frac{\partial\left[P_{n-1}\right]}{\partial D}\Delta u^{n-1}\right] \\ &-\psi^{n-1T}\left[\frac{\partial R(u^{n-2})}{\partial D} - \frac{\partial\left[P_{n-2}\right]}{\partial D}\Delta u^{n-2}\right] \\ &-\ldots \\ &-\psi^{1T}\left[\frac{\partial R(u^{0})}{\partial D} - \frac{\partial\left[P_{0}\right]}{\partial D}\Delta u^{0}\right]\end{aligned} \tag{38}$$

Verification and Investigation of Newton's Method

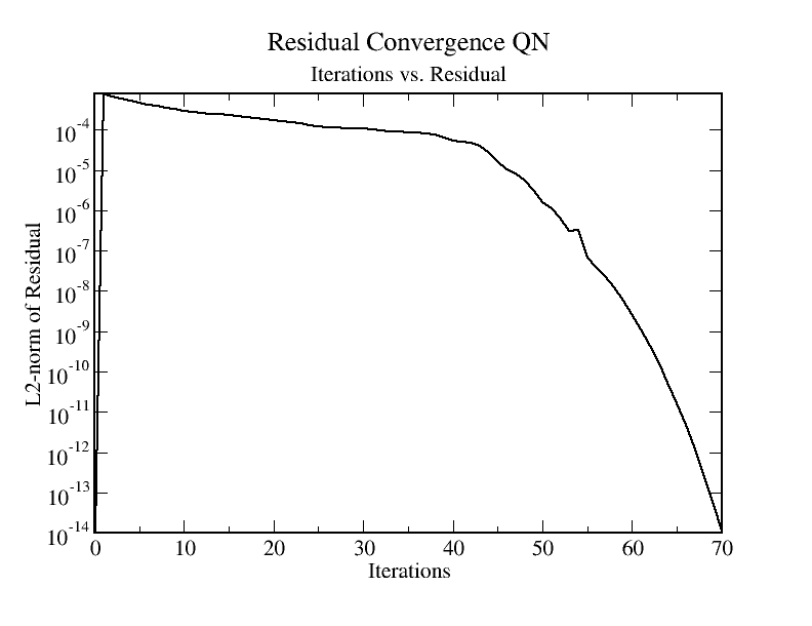
PTA Adjoint System (Verification)

Scheme	Steps	Complex Sensitivity	Adjoint Sensitivity	$\|\|Residual\|\|_2$
NC	10	-2.344315562026746	-2.344315562026735	0.3205858355993253E-03
NC	20	-5.117070610162804	-5.5117070610162807	0.2522158472255535E-03
QNII	10	-3.482196439075767	-3.482196439075786	0.2982633094835574E-03
QNII	20	-2.264493935071775	-2.264493935071703	0.1691766297621063E-03

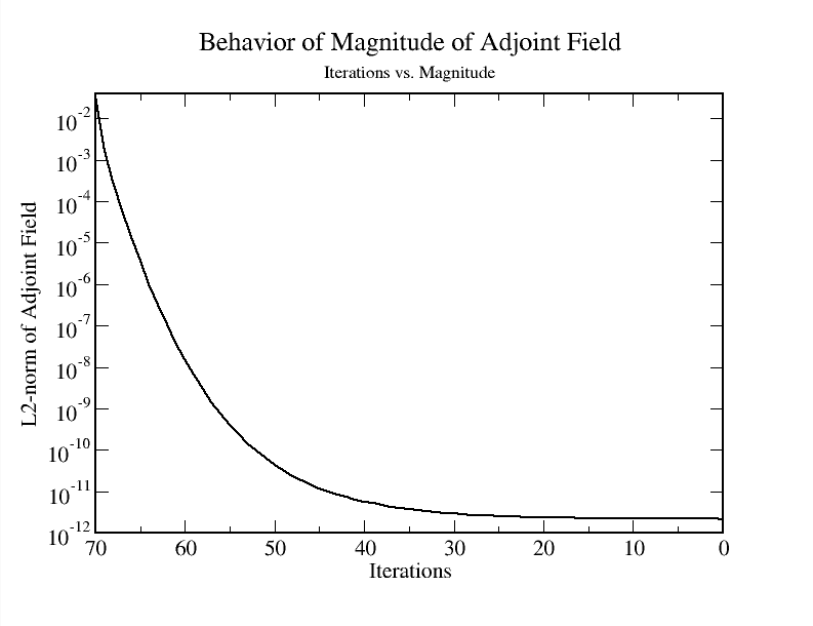
Comparison of adjoint and complex-step computed sensitivities

- Here we show the sensitivity comparison between the Newton-Chord (NC) and quasi-Newton inverse identity (QNII) adjoints and complex sensitivities to verify the implementation
- For the Newton-Chord method, the solver must be done with a dual solver, here with block Jacobi.
- For the Quasi-Newton method, the solver does not need to be a dual solver and is done primarily with block Gauss-Seidel

PTA Adjoint System Quasi-Newton Investigation



(a) Residual Convergence

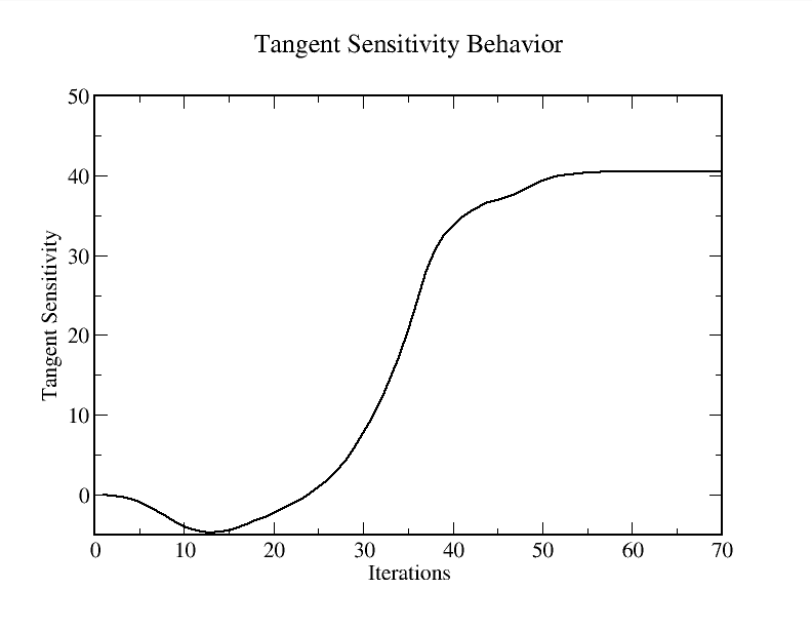


(b) Adjoint Magnitude Behavior

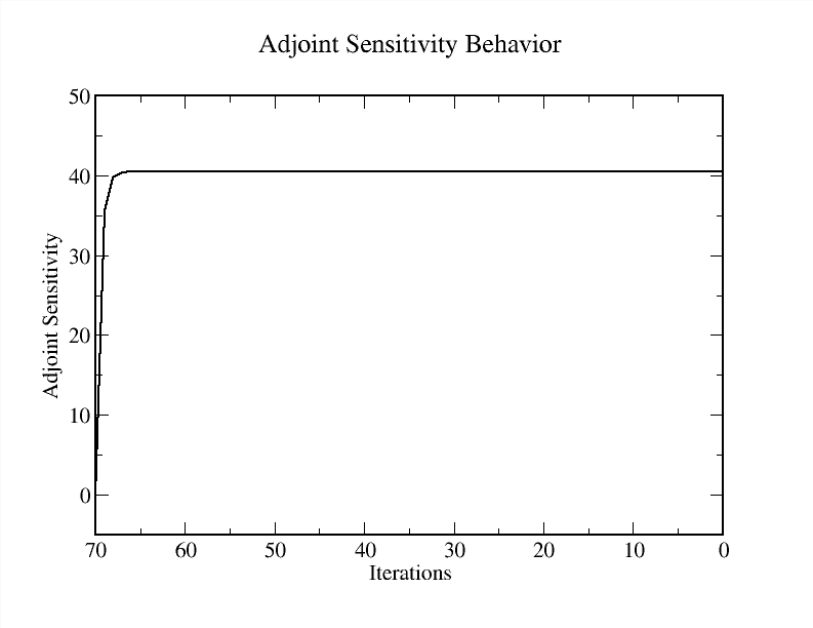
Residual convergence and adjoint magnitude behavior for Quasi-Newton scheme

The adjoint at a pseudo-time step relates a perturbation in the constraint of that pseudo-time step to a change in the objective function

PTA Adjoint System Quasi-Newton Investigation



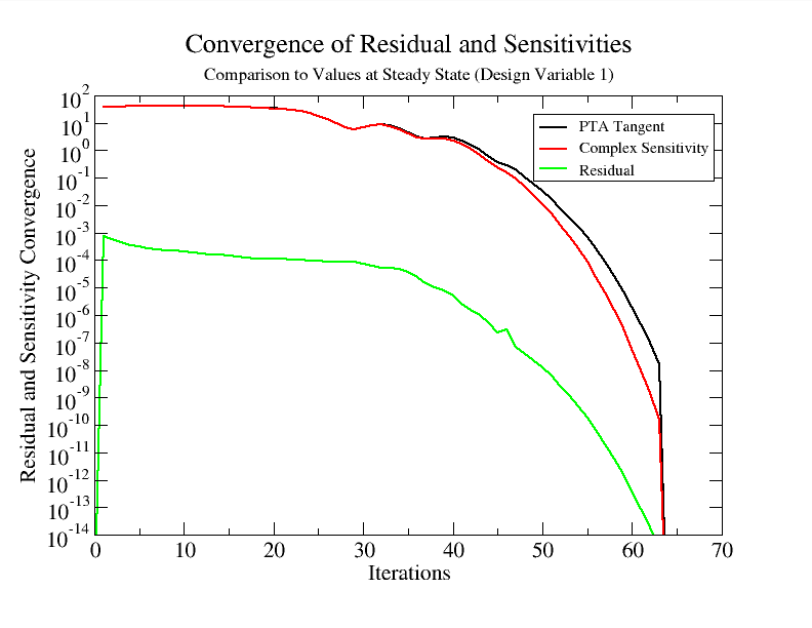
(a) Tangent Sensitivity Convergence



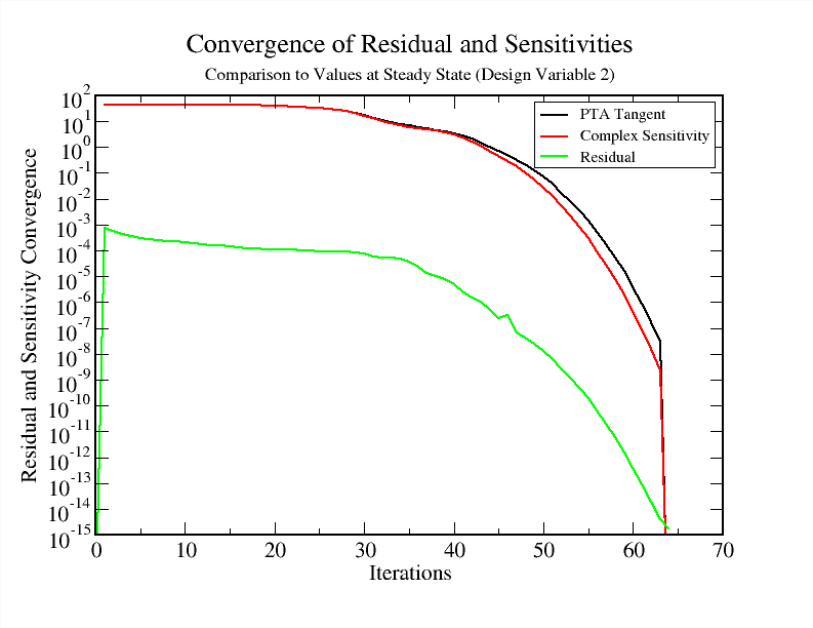
(b) Adjoint Sensitivity Convergence

Sensitivity convergence for Quasi-Newton scheme

PTA Tangent Behavior for Linear Tolerance = 1e-1



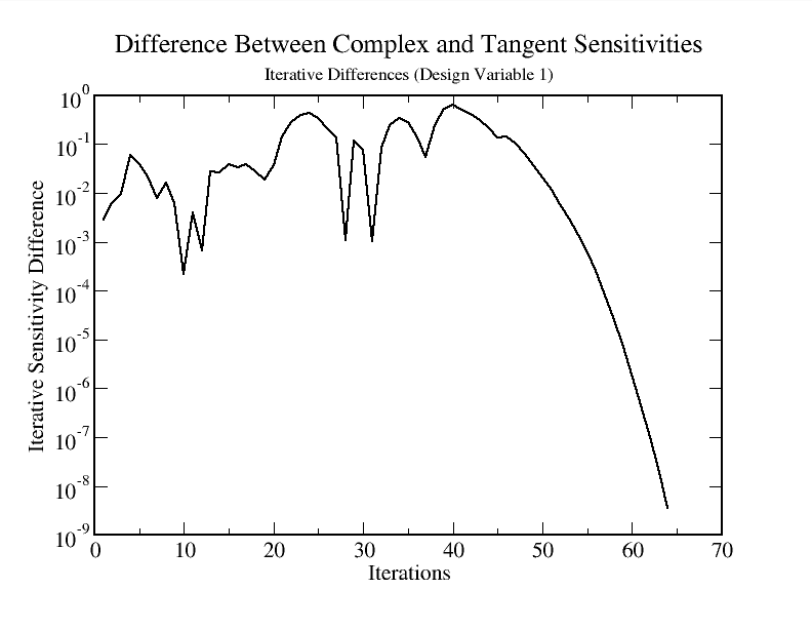
(a) Design Variable 1



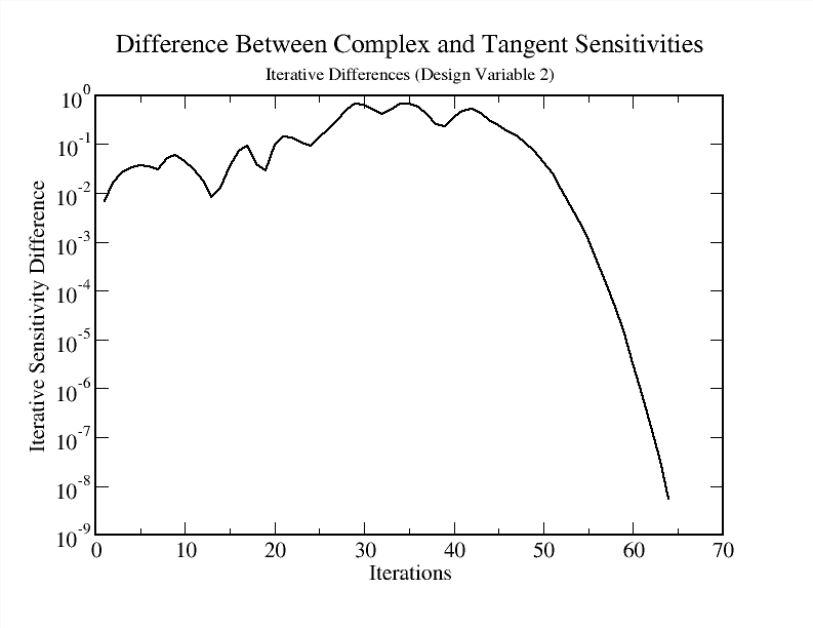
(b) Design Variable 2

Sensitivity convergence for linear tolerance, 1e-1

PTA Tangent Behavior for Linear Tolerance = 1e-1



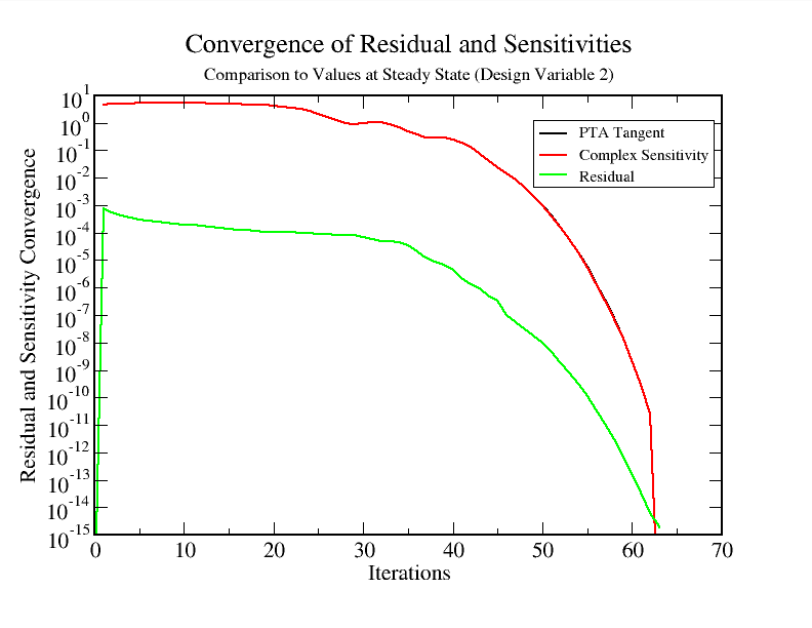
(a) Design Variable 1



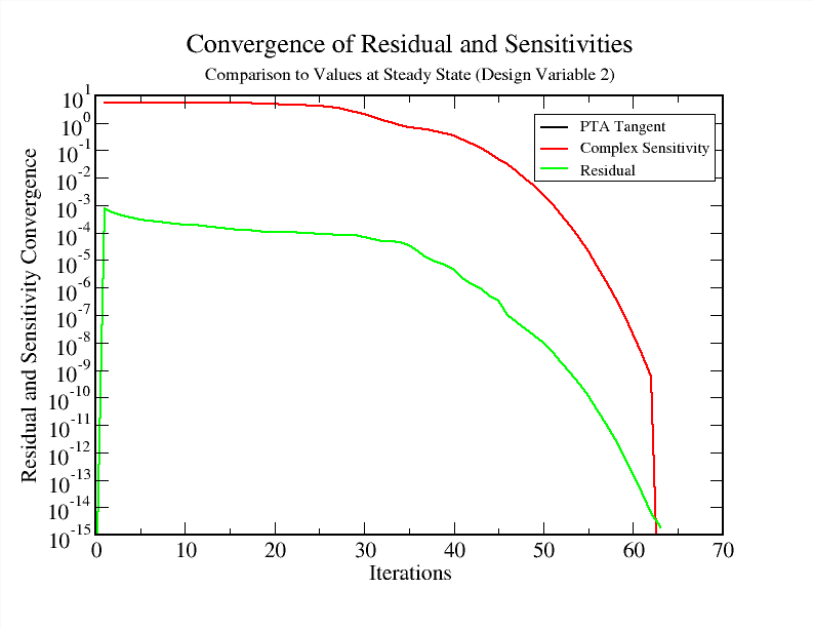
(b) Design Variable 2

Iterative sensitivity difference for linear tolerance, 1e-1

PTA Tangent Behavior for Linear Tolerance = 1e-4



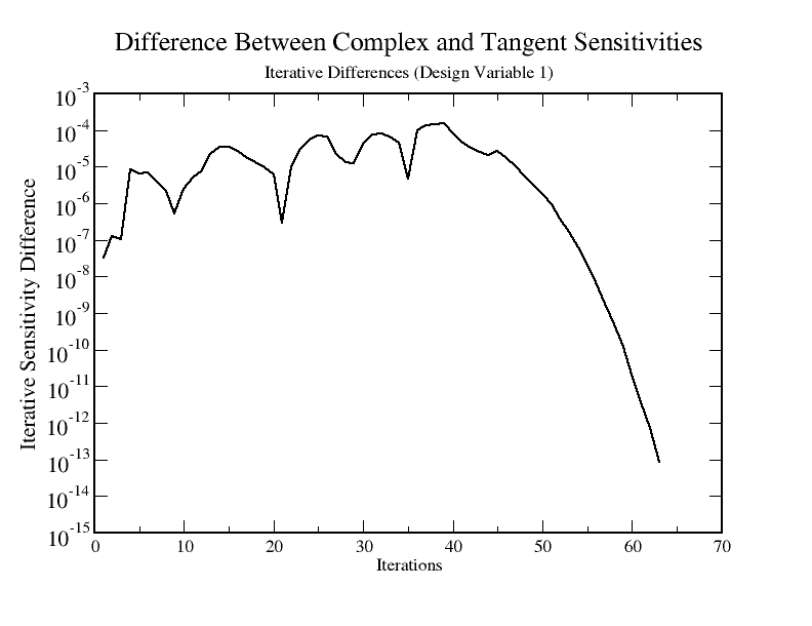
(a) Design Variable 1



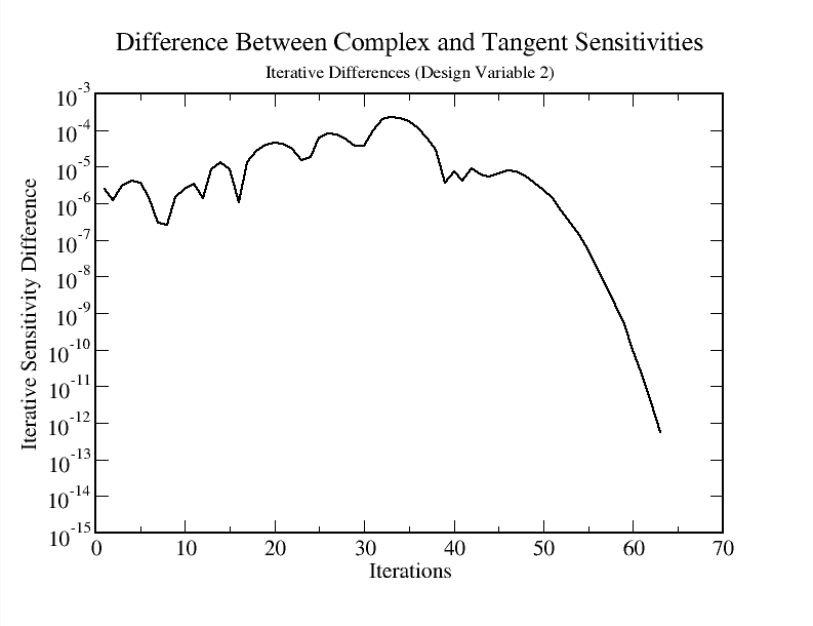
(b) Design Variable 2

Sensitivity convergence for linear tolerance, 1e-4

PTA Tangent Behavior for Linear Tolerance = 1e-4



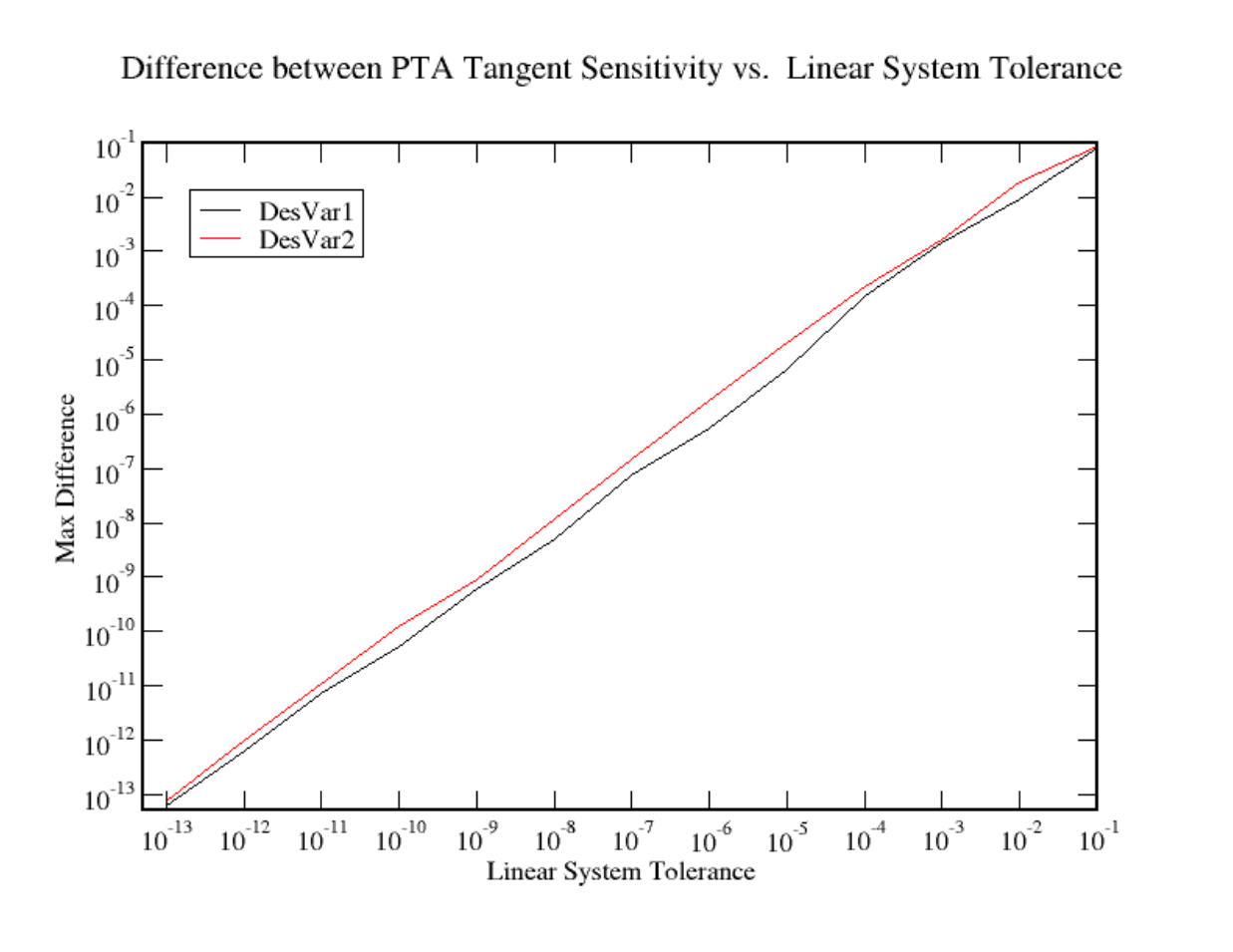
(a) Design Variable 1



(b) Design Variable 2

Iterative sensitivity difference for linear tolerance, 1e-4

PTA Tangent Behavior as a function of linear tolerance

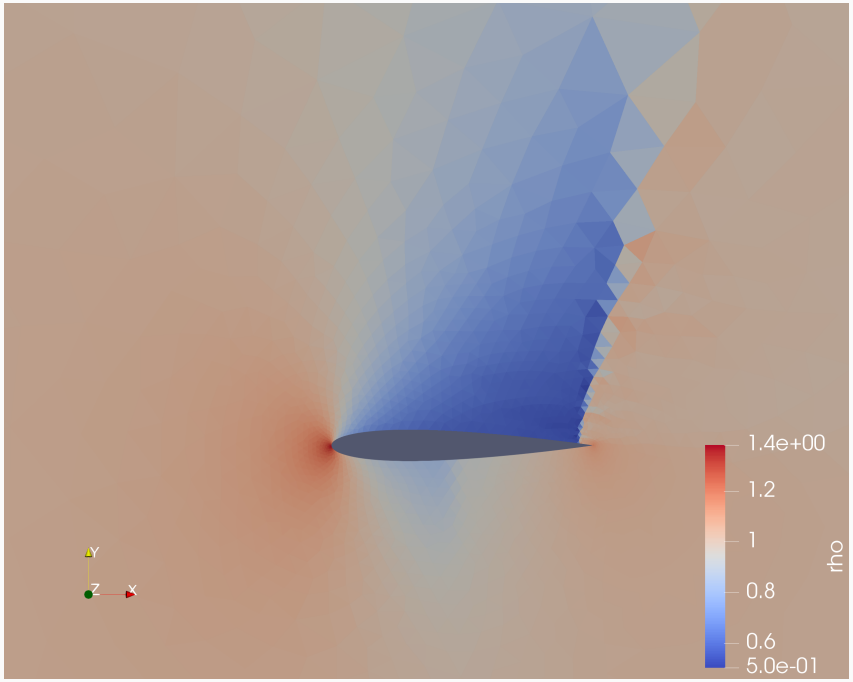


Maximum iterative difference vs. linear tolerance

Results

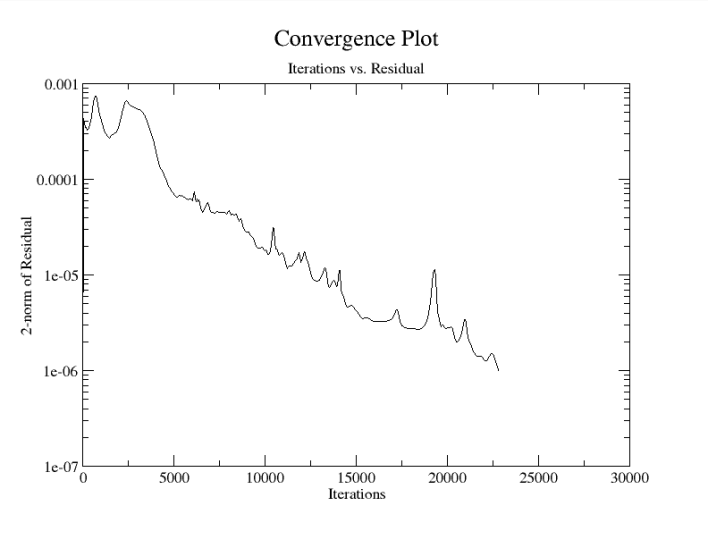
Results for a Truncated Simulation

Results: Truncated Simulation

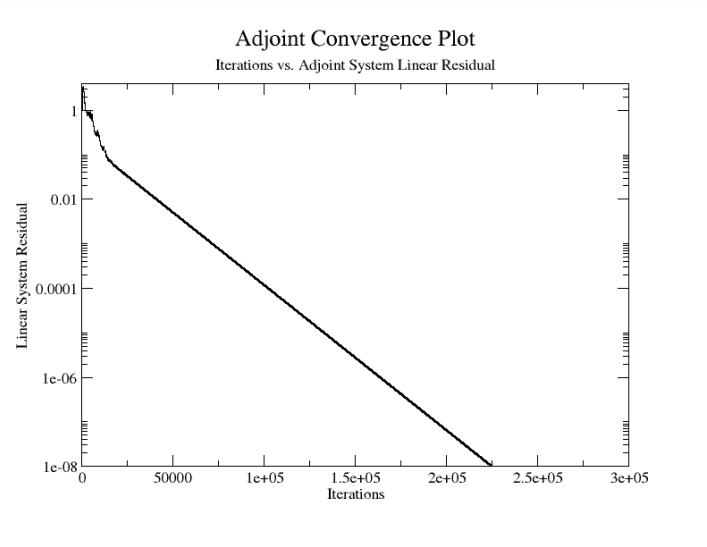


Density field for NACA0012 airfoil in $Mach = .85$, $\alpha = 3$

Results: Steady-State Residual and Steady-State Adjoint Convergence



(a) Primal Convergence



(b) Adjoint Linear Convergence

Residual and steady-state adjoint convergence

- The primal problem residual has decreased by three orders of magnitude.
- The adjoint problem has converged by 8 orders of magnitude, and is sufficiently accurate for this comparison.

Why is Angle Important?

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

$$a = \frac{u \cdot v}{\|u\| \, \|v\|} \tag{39}$$

$$\begin{aligned} \theta &= arccos(a), \theta \leq 180 \\ \theta &= 360 - arccos(a), \theta > 180 \end{aligned} \tag{40}$$

Why is Angle Important?

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

$$a = \frac{u \cdot v}{\|u\| \, \|v\|} \tag{39}$$

$$\begin{aligned} \theta &= arccos(a), \theta \leq 180 \\ \theta &= 360 - arccos(a), \theta > 180 \end{aligned} \tag{40}$$

- The important thing about the sensitivity vector is the direction.

Why is Angle Important?

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

$$a = \frac{u \cdot v}{\|u\| \, \|v\|} \tag{39}$$

$$\begin{aligned} \theta &= arccos(a), \theta \leq 180 \\ \theta &= 360 - arccos(a), \theta > 180 \end{aligned} \tag{40}$$

- 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.

Why is Angle Important?

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

$$a = \frac{u \cdot v}{\|u\| \, \|v\|} \tag{39}$$

$$\begin{aligned} \theta &= arccos(a), \theta \leq 180 \\ \theta &= 360 - arccos(a), \theta > 180 \end{aligned} \tag{40}$$

- 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.

Results: Truncated Simulation Sensitivity Vectors

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

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

Results: Truncated Simulation Sensitivity Vectors

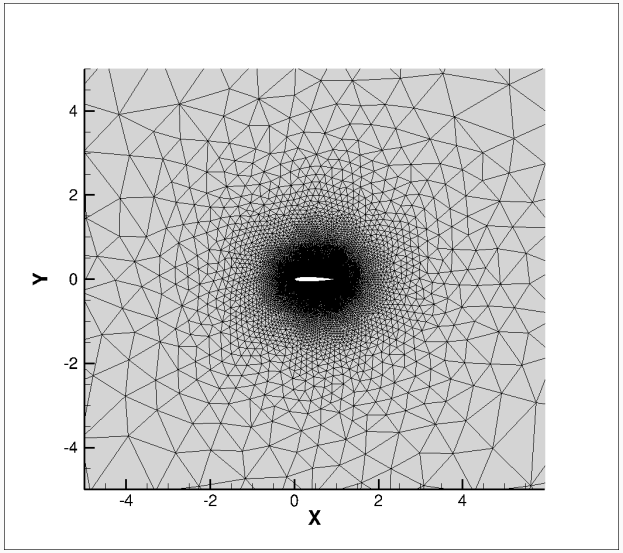
Design Variable	Steady-State Value	Pseudo-Time Accurate Value	Percent Difference
1	-38.9168562768552	-34.07260014448312	14.2%
2	-58.8314866801896	-62.28918541979274	5.6%
3	-51.9374815863084	-70.15517817818902	26.0%
4	-69.3520141341324	-95.35571541527936	27.3%

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

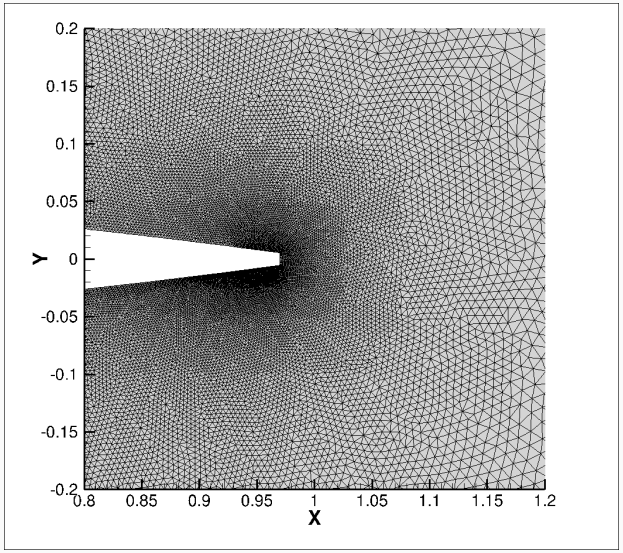
The angle between the two sensitivity vectors is $\theta = 8.652^o$.

Results for Non-Converging Transonic Simulation

Results: Non-Converging Transonic Case Introduction



(a) Mesh

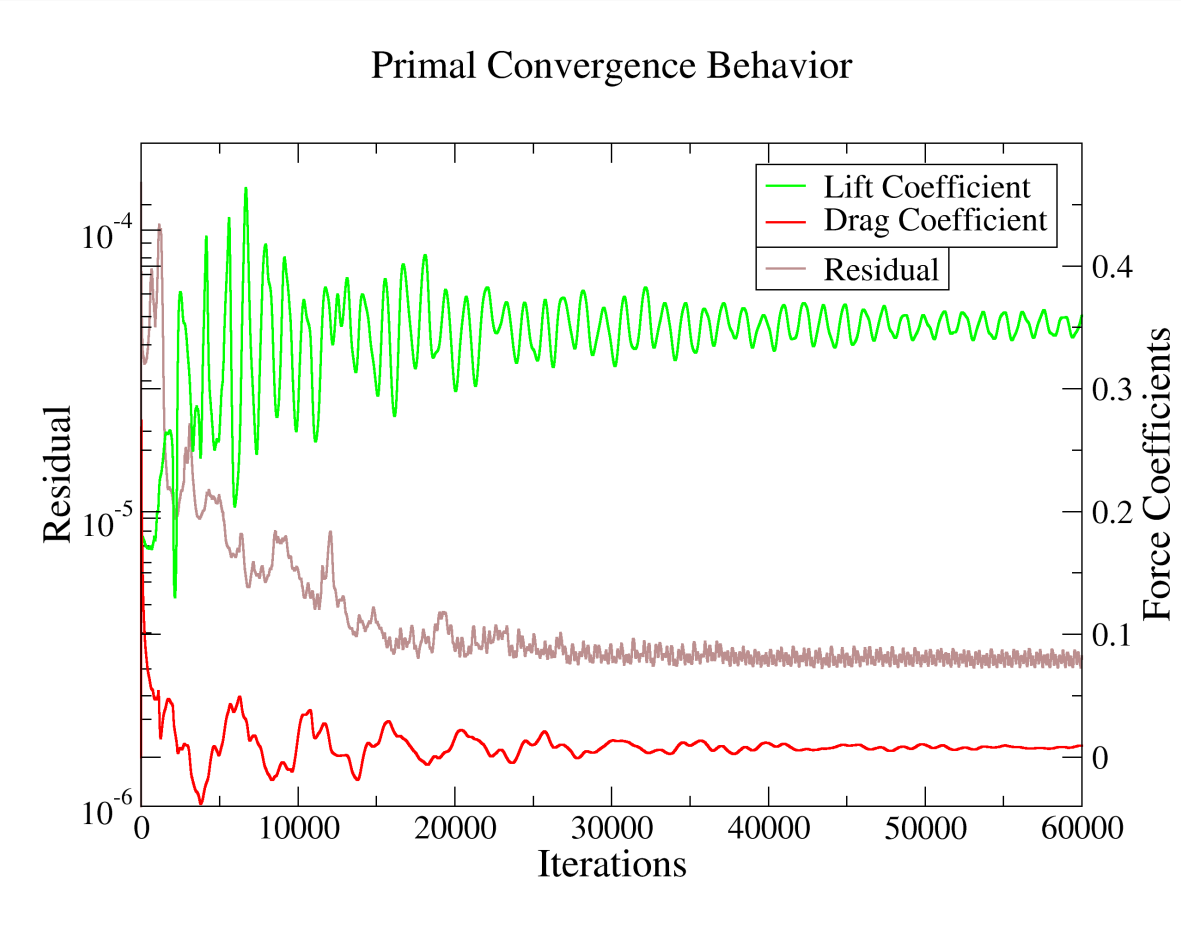


(b) Trailing Edge

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

Results: Primal Behavior of Non-Converging Transonic Simulation



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

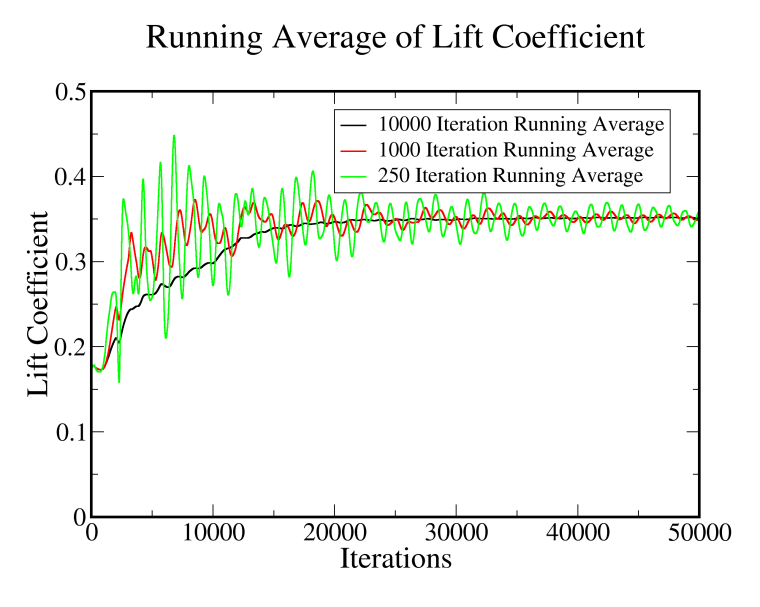
Results: Running Average of Force Coefficients

- Using instantaneous objective functionals in oscillatory flows can provide sensitivities with a high dependence on the initial conditions and the design variables, this is not true for the averaged objective functional.

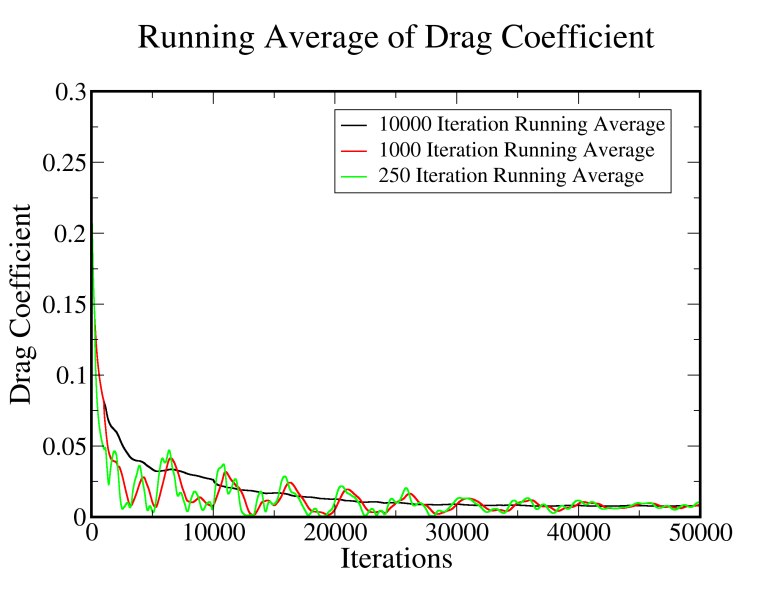
Results: Running Average of Force Coefficients

- Using instantaneous objective functionals in oscillatory flows can provide sensitivities with a high dependence on the initial conditions and the design variables, this is not true for the averaged objective functional.
- The averaged objective functional, for a statistically converged average shows low dependence on the initial condition, which could allow for partial backwards in time integration for cheaper evaluation of the sensitivities.

Results: Running Average of Force Coefficients



(a) C_L over different averaging windows

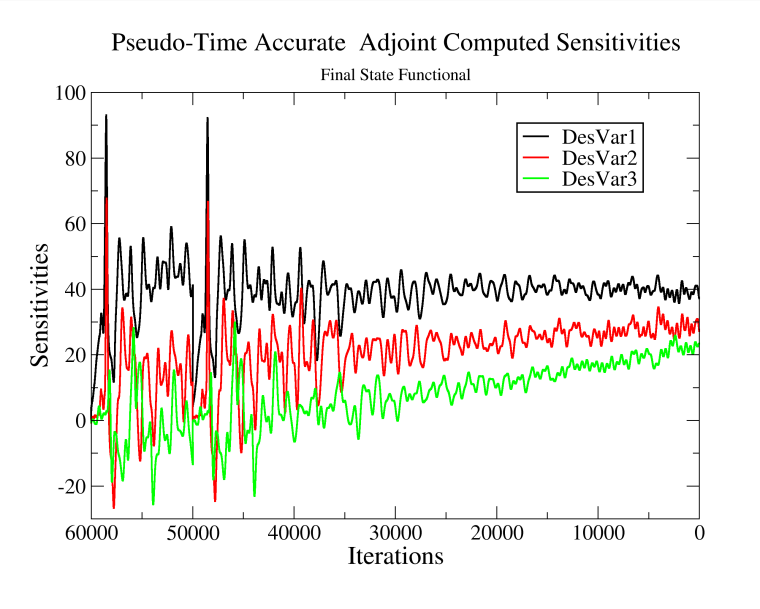


(b) C_D over different averaging windows

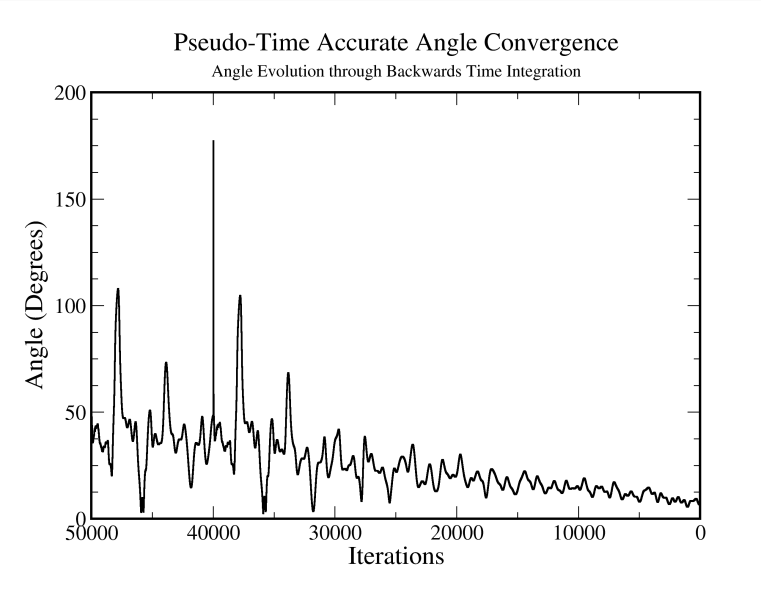
Force coefficient averaging over different windows

- 250 and 1000 iteration averaging windows do not damp out oscillations sufficiently
- The 10000 averaging window appears to be statistically converged. We will use 20000 iteration averaging window going forward.

Results: Sensitivities for Final State Objective Functional



(a) Sensitivities

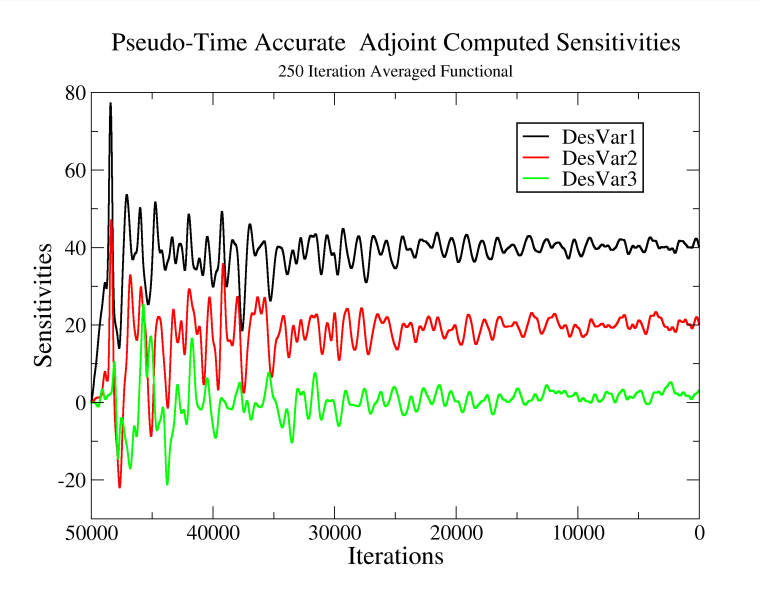


(b) Angle Convergence

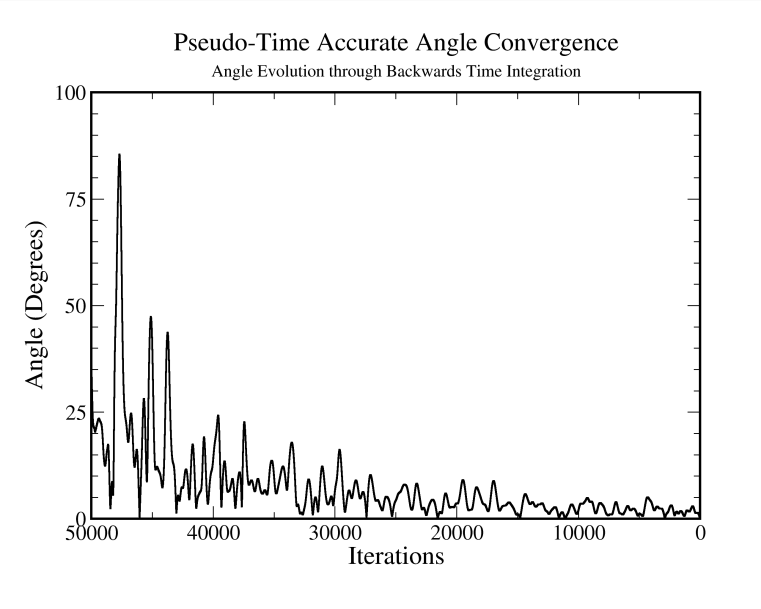
Sensitivities integrated through pseudo-time

- The sensitivities are highly oscillatory.
- The angle fails to converge well as we integrate back.

Results: Sensitivities for 250 Iteration Window Averaged Objective Functional



(a) Sensitivities

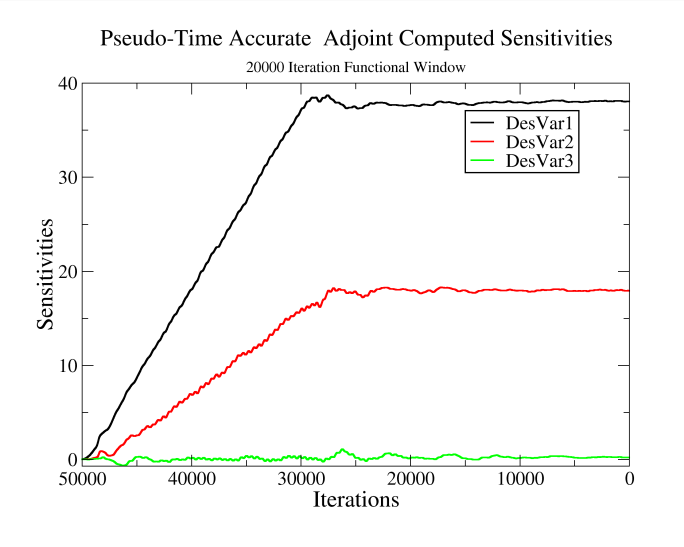


(b) Angle Convergence

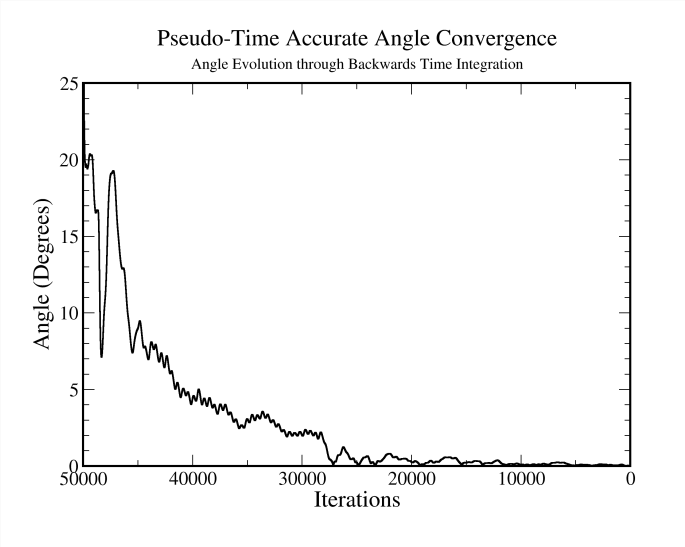
Sensitivities integrated through pseudo-time

- The sensitivities are highly oscillatory, but less oscillatory than the instantaneous objective case.
- The angle is better behaved, but fails to converge.

Results: Sensitivities for 20000 Iteration Window Averaged Objective Functional



(a) Sensitivities

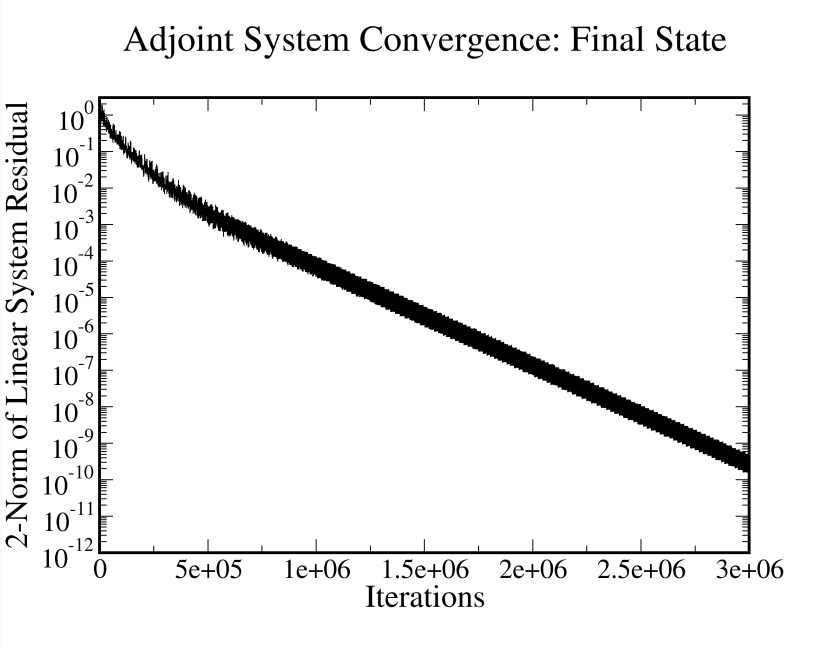


(b) Angle Convergence

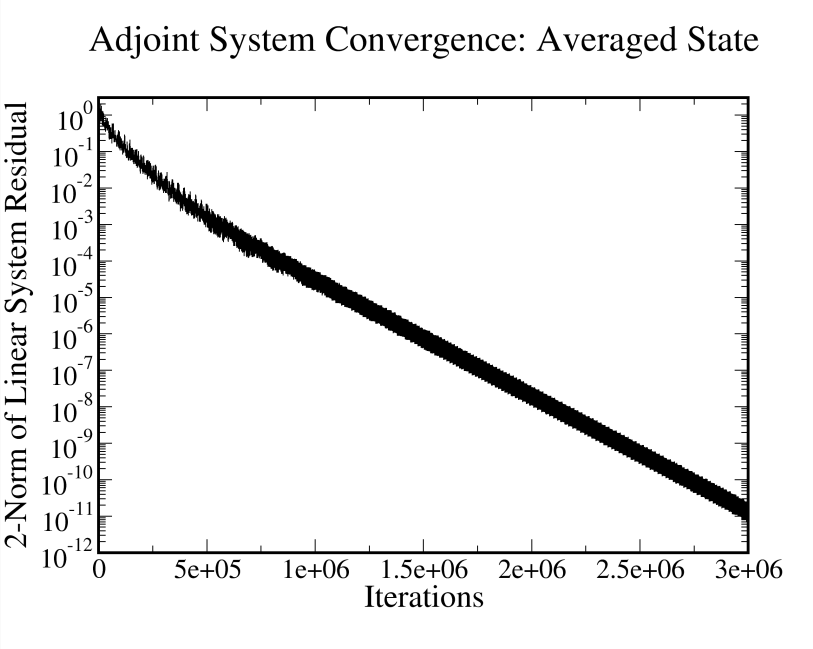
Sensitivities integrated through pseudo-time

- The sensitivities are well behaved and converge to a final value outside of the averaging window
- The angle is well behaved, lending credence to partial backwards-in-psuedo-time integration, as was previously theorized.

Results: Convergence for Steady-State Adjoint systems



(a) Final State



(b) Average State

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

- Both systems take many iterations to converge, this follows the conclusions of Krakos and Darmofal.
- The average state converges slightly faster.

Results: 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%

Comparison of sensitivity vectors for steady-state and pseudo-time accurate (20000 iteration averaging window) adjoints

Results: 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%

Comparison of sensitivity vectors for steady-state and pseudo-time accurate (20000 iteration averaging window) adjoints

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

Results: 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%

Comparison of sensitivity vectors for steady-state and pseudo-time accurate (20000 iteration averaging window) adjoints

Results: 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%

Comparison of sensitivity vectors for steady-state and pseudo-time accurate (20000 iteration averaging window) adjoints

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

Conclusions and Future Work

Conclusions

- For Newton's Method we can compute the adjoint and tangent systems without full solution of the linear system and obtain error estimates on the sensitivity

Conclusions

- For Newton's Method we can compute the adjoint and tangent systems without full solution of the linear system and obtain error estimates on the sensitivity
- For non-converging or truncated simulations, the pseudo-time accurate sensitivities are significantly different than the steady-state sensitivities when linearized about the final state and the pseudo-time averaged state

Conclusions

- For Newton's Method we can compute the adjoint and tangent systems without full solution of the linear system and obtain error estimates on the sensitivity
- For non-converging or truncated simulations, the pseudo-time accurate sensitivities are significantly different than the steady-state sensitivities when linearized about the final state and the pseudo-time averaged state
- Increasing the functional averaging window provides better sensitivity behavior

Conclusions

- For Newton's Method we can compute the adjoint and tangent systems without full solution of the linear system and obtain error estimates on the sensitivity
- For non-converging or truncated simulations, the pseudo-time accurate sensitivities are significantly different than the steady-state sensitivities when linearized about the final state and the pseudo-time averaged state
- Increasing the functional averaging window provides better sensitivity behavior
- The pseudo-time accurate sensitivities converge to approximately the final values after integrating through the functional averaging window (for long windows)

Conclusions

- For Newton's Method we can compute the adjoint and tangent systems without full solution of the linear system and obtain error estimates on the sensitivity
- For non-converging or truncated simulations, the pseudo-time accurate sensitivities are significantly different than the steady-state sensitivities when linearized about the final state and the pseudo-time averaged state
- Increasing the functional averaging window provides better sensitivity behavior
- The pseudo-time accurate sensitivities converge to approximately the final values after integrating through the functional averaging window (for long windows)
- We do not address chaotic flows where the sensitivity computation diverges.

Future Work

- Application to error estimation (underway)

Future Work

- Application to error estimation (underway)
- Test cases with design cycles

Future Work

- Application to error estimation (underway)
- Test cases with design cycles
- Applications to unsteady simulations and adjoints

Thank you!