Hessian Calculation using AD

Devendra Ghate and Mike Giles

devg@comlab.ox.ac.uk, giles@comlab.ox.ac.uk

Oxford University Computing Laboratory

Motivation

Hessian of a functional of interest (J) with respect to design variables α_i

$$\frac{\partial^2 J}{\partial \alpha_i \partial \alpha_j},$$

is used in various applications including optimisation and uncertainty propagation.

Background

Hessian capability in some AD packages using forward-on-reverse mode

- ADOL-C provides driver routines for calculating
  - Entire hessian
  - Hessian-vector product
- ADIFOR & TAPENADE
  - forward-on-forward
  - forward-on-reverse (?)
- `www.autodiff.org`
  - Very few applications of AD tools for Hessian calculation listed
  - Large number of publications on Hessian calculation in early 90s

Background

Research by AD community in

- parallel implementation of Hessian calculation (Bücker *et.al.*, '06) (forward-on-forward)
- efficient sparse Hessian calculation (Verma, '99)
- efficient calculation of Hessian-Vector products for optimisation applications (many publications)

Background

AD community strives to provide a generic Hessian capability, but “BLACK-BOX” application of AD for Hessian is too expensive for applications based on fixed-point iteration.

Since we already have a neatly structured nonlinear code (HYDRA) with linear and adjoint capabilities, we look for an algorithm suited for our application.

Gradient Calculation

Consider the functional of interest $j(\alpha) = J(\alpha, w(\alpha))$ where w is defined by $R(\alpha, w) = 0$. Here $w = \{x, u\}$. Gradient of J is given by

$$\frac{\partial j}{\partial \alpha_i} = \frac{\partial J}{\partial \alpha_i} + \frac{\partial J}{\partial w}\,\frac{\partial w}{\partial \alpha_i}.$$

Gradient Calculation

Consider the functional of interest $j(\alpha) = J(\alpha, w(\alpha))$ where w is defined by $R(\alpha, w) = 0$. Here $w = \{x, u\}$. Gradient of J is given by

$$\frac{\partial j}{\partial \alpha_i} = \frac{\partial J}{\partial \alpha_i} + \frac{\partial J}{\partial w}\,\frac{\partial w}{\partial \alpha_i}.$$

Continuing similarly, the Hessian is

$$\begin{aligned}\frac{\partial^2 j}{\partial \alpha_i\,\partial \alpha_j} &= \frac{\partial^2 J}{\partial \alpha_i\,\partial \alpha_j} + \frac{\partial^2 J}{\partial \alpha_i \partial w}\left(\frac{\partial w}{\partial \alpha_j}\right) + \frac{\partial^2 J}{\partial \alpha_j \partial w}\left(\frac{\partial w}{\partial \alpha_i}\right) \\ &+ \frac{\partial^2 J}{\partial w^2}\left(\frac{\partial w}{\partial \alpha_i}\,\frac{\partial w}{\partial \alpha_j}\right) + \frac{\partial J}{\partial w}\left(\frac{\partial^2 w}{\partial \alpha_i\,\partial \alpha_j}\right)\end{aligned}$$

Gradient Calculation

Consider the functional of interest $j(\alpha) = J(\alpha, w(\alpha))$ where w is defined by $R(\alpha, w) = 0$. Here $w = \{x, u\}$. Gradient of J is given by

$$\frac{\partial j}{\partial \alpha_i} = \frac{\partial J}{\partial \alpha_i} + \frac{\partial J}{\partial w}\,\frac{\partial w}{\partial \alpha_i}.$$

Continuing similarly, the Hessian is

$$\begin{aligned}\frac{\partial^2 j}{\partial \alpha_i\,\partial \alpha_j} \;=\;& \frac{\partial^2 J}{\partial \alpha_i\,\partial \alpha_j} + \frac{\partial^2 J}{\partial \alpha_i \partial w}\left(\frac{\partial w}{\partial \alpha_j}\right) + \frac{\partial^2 J}{\partial \alpha_j \partial w}\left(\frac{\partial w}{\partial \alpha_i}\right)\\ +\;& \frac{\partial^2 J}{\partial w^2}\left(\frac{\partial w}{\partial \alpha_i}\,\frac{\partial w}{\partial \alpha_j}\right) + \frac{\partial J}{\partial w}\left(\frac{\partial^2 w}{\partial \alpha_i\,\partial \alpha_j}\right)\end{aligned}$$

Explicit Calculation

If $\alpha \in R^n$ then (forward mode) Hessian calculation requires

- Baseline Nonlinear solution
- $O(n)$ Linear solutions $\frac{\partial w}{\partial \alpha_i}$
- $O(n^2)$ Second derivative solutions $\frac{\partial^2 w}{\partial \alpha_i \, \partial \alpha_j}$

For explicit functions, this is straightforward.

But flow equations are implicit, which would require computationally expensive iterative solutions for $\frac{\partial w}{\partial \alpha_i}$ and $\frac{\partial^2 w}{\partial \alpha_i \, \partial \alpha_j}$.

(The linearisation of an iterative process also affects the cost of forward-on-reverse calculations.)

Hessian Calculation

Rearranging, the Hessian of the functional of interest J is

$$\frac{\partial^2 j}{\partial \alpha_i \, \partial \alpha_j} = \frac{\partial J}{\partial w} \, \frac{\partial^2 w}{\partial \alpha_i \partial \alpha_j} + D^2_{i,j} J$$

where,

$$D^2_{i,j} J = \frac{\partial^2 J}{\partial \alpha_i \, \partial \alpha_j} + \frac{\partial^2 J}{\partial \alpha_i \partial w} \left(\frac{\partial w}{\partial \alpha_j} \right) + \frac{\partial^2 J}{\partial \alpha_j \partial w} \left(\frac{\partial w}{\partial \alpha_i} \right) + \frac{\partial^2 J}{\partial w^2} \left(\frac{\partial w}{\partial \alpha_i} \, \frac{\partial w}{\partial \alpha_j} \right)$$

Hessian Calculation

Similarly, for the state equation $R(\alpha, w) = 0$,

$$\frac{\partial R}{\partial w} \, \frac{\partial^2 w}{\partial \alpha_i \partial \alpha_j} + D^2_{i,j} R = 0$$

Substituting,

$$\begin{aligned}\frac{\partial^2 j}{\partial \alpha_i \partial \alpha_j} &= -\frac{\partial J}{\partial w} \left(\frac{\partial R}{\partial w} \right)^{-1} D^2_{i,j} R + D^2_{i,j} J \\ &= v^T D^2_{i,j} R + D^2_{i,j} J.\end{aligned}$$

where v is the usual adjoint variable associated with J.

Computational Cost

Suppose we have n design variables and m functions of interest then

- Baseline nonlinear solution (iterative)
- n linear solutions (iterative)
- m adjoint solutions (iterative)
- $\frac{1}{2}n(n+1)$ cheap evaluations of $D^2_{i,j}R$
- $m \times \frac{1}{2}n(n+1)$ really cheap dot products $v^T D^2_{i,j}R$
- $m \times \frac{1}{2}n(n+1)$ really cheap evaluations of $D^2_{i,j}J$

Computational Cost

If cost dominated by iterative solver, then

Method	Cost
iterative forward-on-forward	$O(n^2)$
iterative forward-on-reverse	$O(n \times m)$
iterative forward/reverse + residual forward-on-forward	$O(n + m)$

n - # design variables
m - # functions of interest

Implementation

- Tapenade used twice in forward mode to generate double differentiated routines
- Second order perturbations propagated through various nonlinear routines
- No extra code structuring required
- Separate functions written to evaluate $D^2_{i,j}R$ and $D^2_{i,j}J$
- A `makefile` written to generate all the double differentiated routines

Implementation

```
lift_wall   (w,J)
```

```
lift_wall_d (w,wd,J,Jd)
```

wd $= \frac{\partial w}{\partial \alpha_i}$, Jd $= \frac{\partial j}{\partial \alpha_i}$.

```
lift_wall_dd(w,wd0,wd,wdd,J,Jd0,Jd,Jdd)
```

wd $= \frac{\partial w}{\partial \alpha_i}$, Jd $= \frac{\partial j}{\partial \alpha_i}$,

wd0 $= \frac{\partial w}{\partial \alpha_j}$, Jd0 $= \frac{\partial j}{\partial \alpha_j}$,

wdd $= \frac{\partial^2 w}{\partial \alpha_i \partial \alpha_j}$, Jdd $= \frac{\partial^2 j}{\partial \alpha_i \partial \alpha_j}$.

Setting wdd $= 0$ gives Jdd $= D^2_{i,j} J$.

Actual Implementation

$w = \{x, u\}$, where

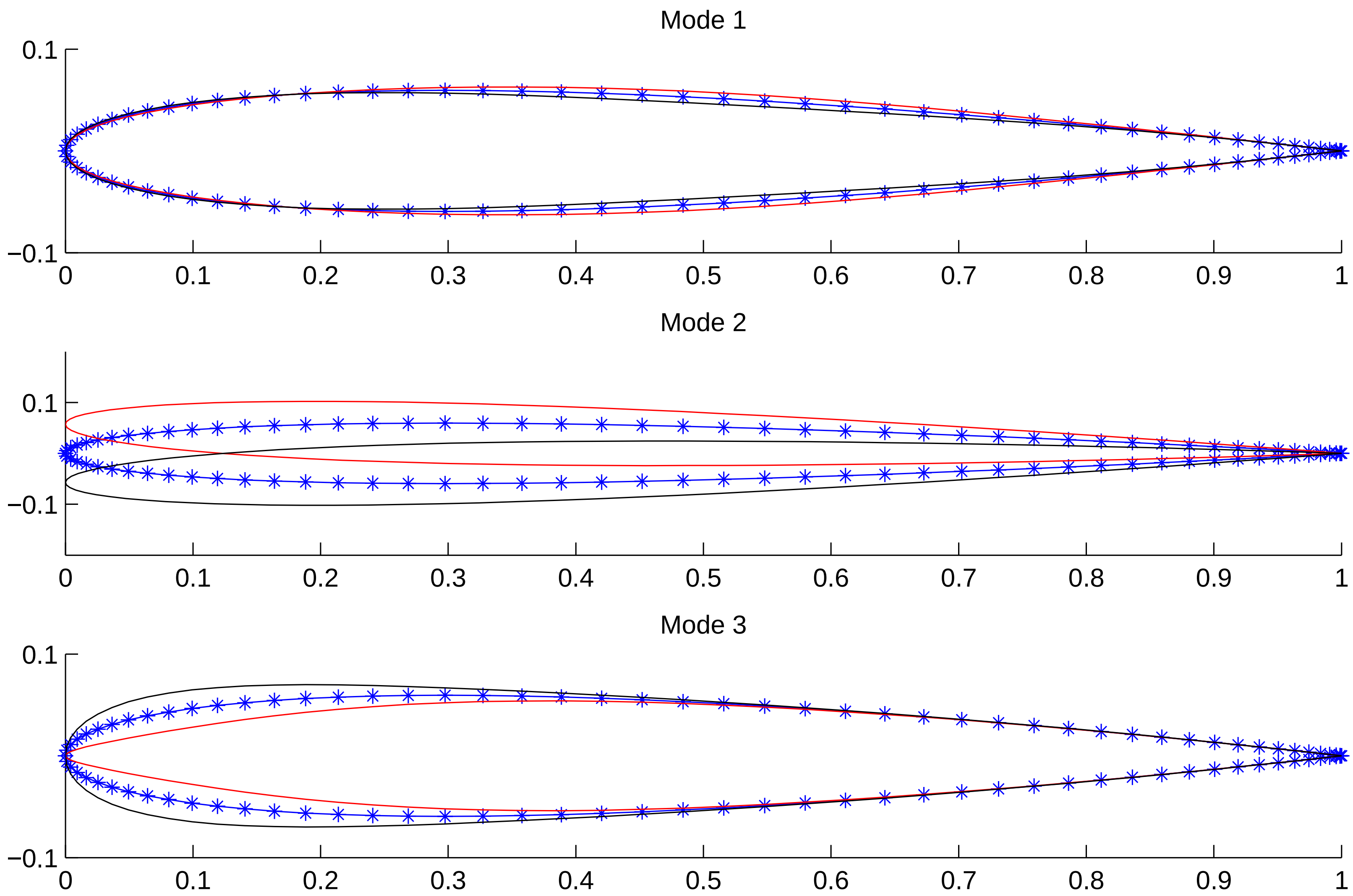
x - grid variables, and

u - flow variables

- $\frac{\partial^2 x}{\partial \alpha_i \partial \alpha_j}$ available
- No need for adjoints w.r.t x
- Only $\frac{\partial^2 u}{\partial \alpha_i \partial \alpha_j}$ replaced by the adjoint term
- Rest of the analysis remains same

Test Case

Three modes of perturbation to NACA0012 airfoil



Test Case

- 2D Euler Solver
- Freestream mach = 0.4, angle of attack = 3^o

Modes	Finite Difference	Direct
1 - 1	$-\mathbf{3.111203}625702499E-07$	$-\mathbf{3.111203}862791910E-07$
1 - 2	$-\mathbf{2.097600}811599999E-06$	$-\mathbf{2.097600}748629300E-06$
1 - 3	$-\mathbf{9.59223}201885120E-07$	$-\mathbf{9.59223}212828186E-07$
2 - 1	$-\mathbf{2.09760074}7610895E-06$	$-\mathbf{2.09760074}8629318E-06$
2 - 2	$-\mathbf{2.15968742}3428786E-04$	$-\mathbf{2.15968742}4802269E-04$
2 - 3	$-\mathbf{1.74653785}7481162E-04$	$-\mathbf{1.74653785}9860203E-04$
3 - 1	$-\mathbf{9.59222}904915369E-07$	$-\mathbf{9.59223}212828262E-07$
3 - 2	$-\mathbf{1.7465378}61210817E-04$	$-\mathbf{1.7465378}59860204E-04$
3 - 3	$-\mathbf{1.97093703}6569875E-05$	$-\mathbf{1.97093703}4187627E-05$

Extrapolation

Comparison between

- Nonlinear Solution

$$L_\alpha = L(x(\alpha), u(\alpha))$$

- Quadratic extrapolation using Hessian

$$L_\alpha = L_{\alpha_0} + \frac{\partial L}{\partial \alpha}(\alpha - \alpha_0) + \frac{1}{2}\frac{\partial^2 L}{\partial \alpha^2}(\alpha - \alpha_0)^2$$

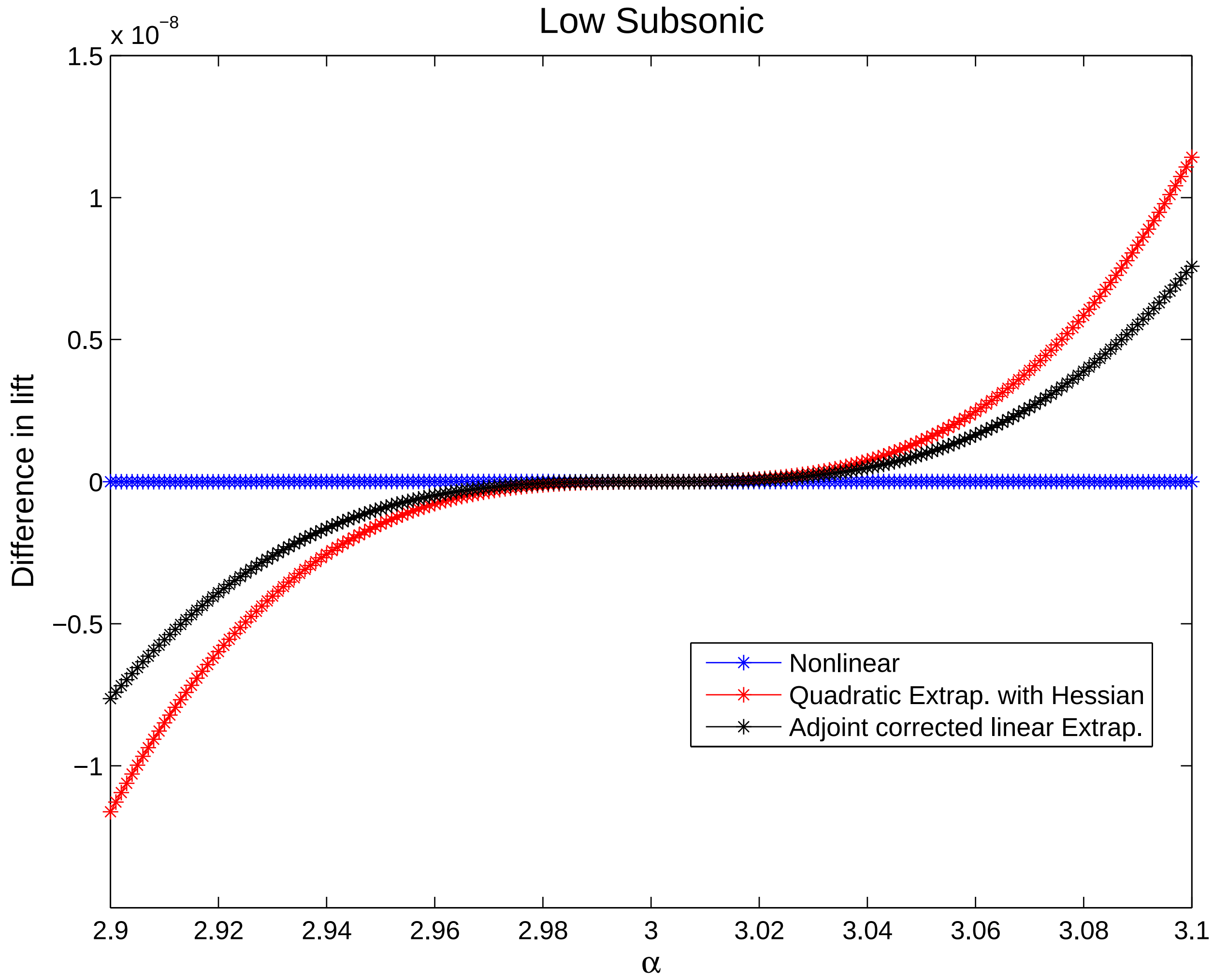
- Linear extrapolation with adjoint correction

$$L_\alpha = L_{\alpha_0} + \frac{\partial L}{\partial \alpha}(\alpha - \alpha_0) - v(\alpha_0)^T R(x(\alpha), u(\alpha_0) + \frac{\partial u}{\partial \alpha}(\alpha - \alpha_0))$$

The difference from a least-squares cubic fit of the nonlinear solution is plotted.

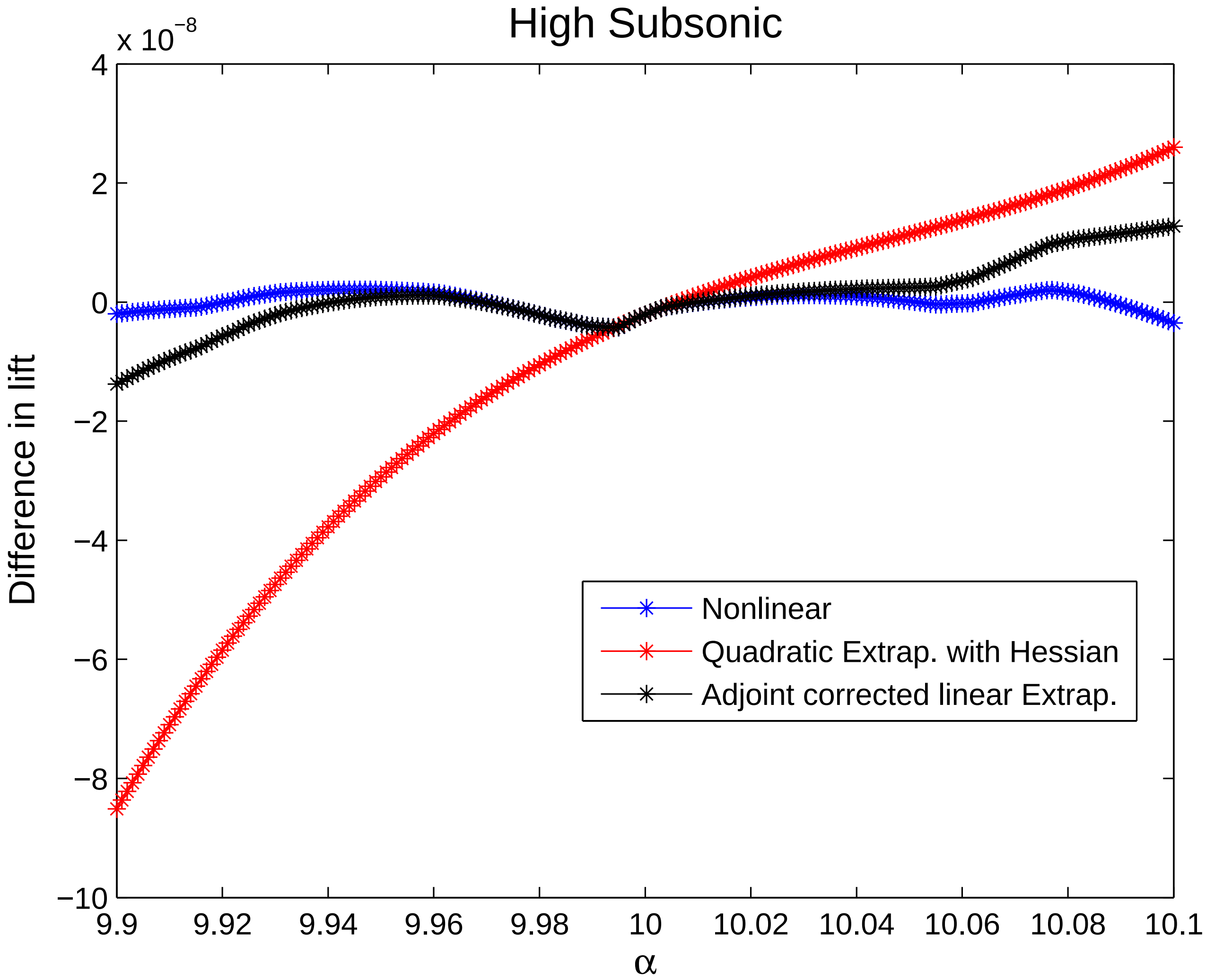
Extrapolation

Low subsonic test case (Freestream mach = 0.4, AOA = 3^o)



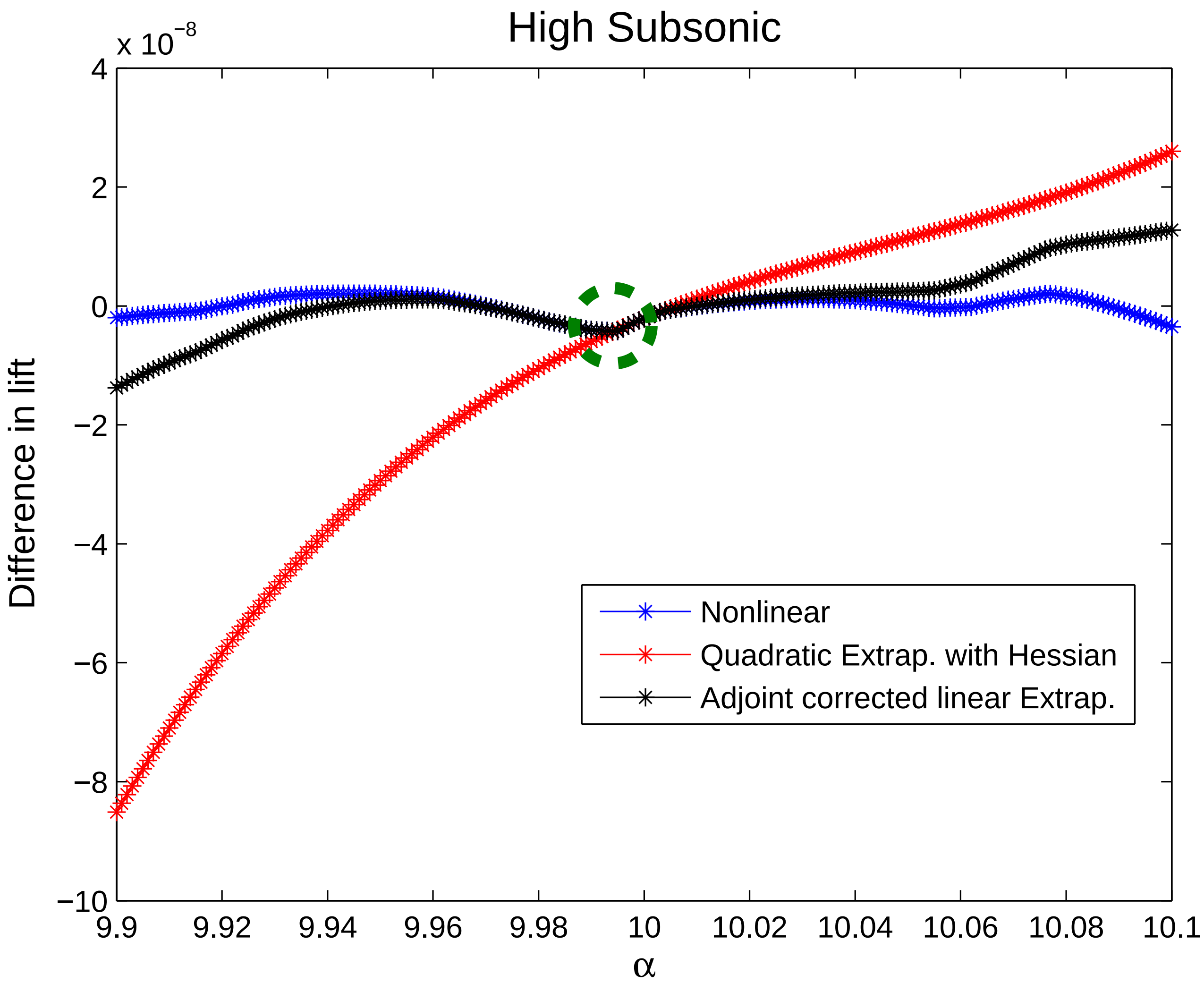
Extrapolation

High subsonic test case (Freestream mach = 0.65, AOA = 10^o)



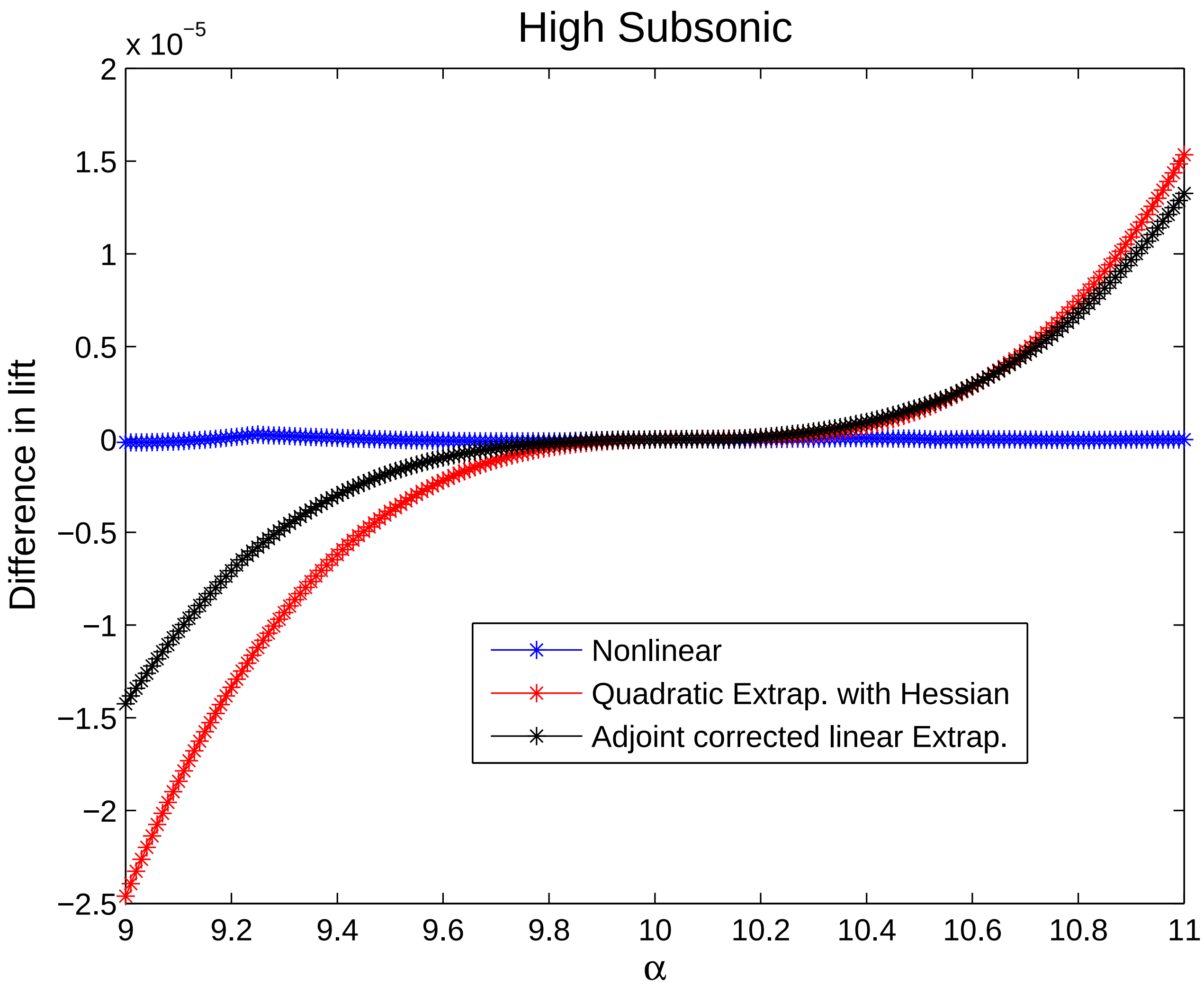
Extrapolation

High subsonic test case (Freestream mach = 0.65, AOA = 10^o)

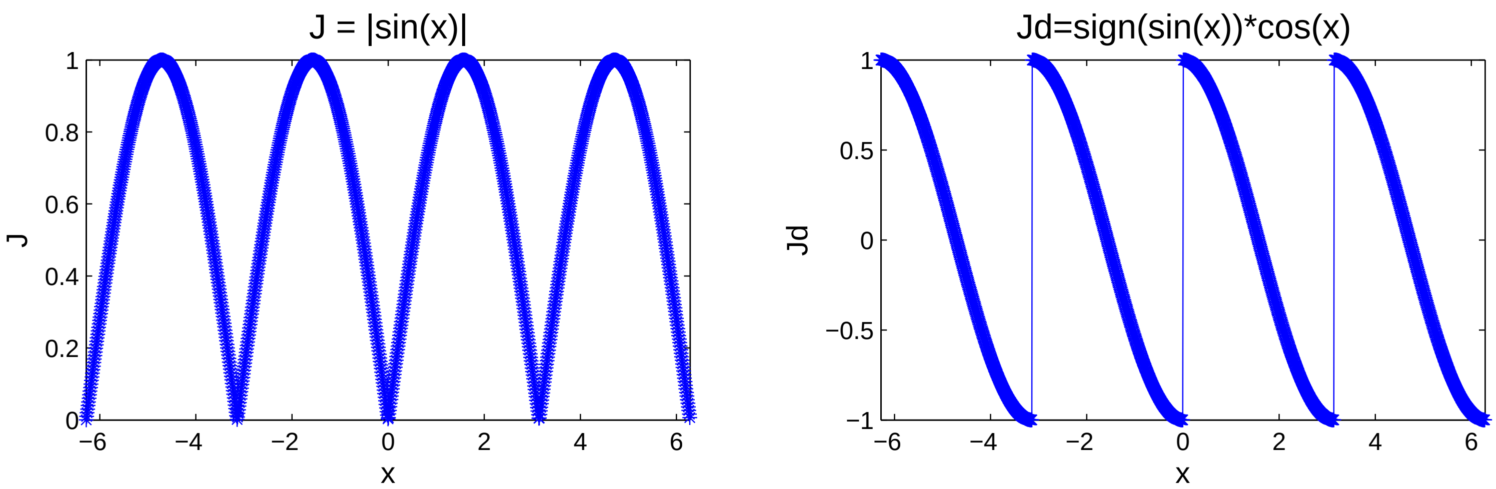


Extrapolation

High subsonic test case (Freestream mach = 0.65, AOA = 10^o)



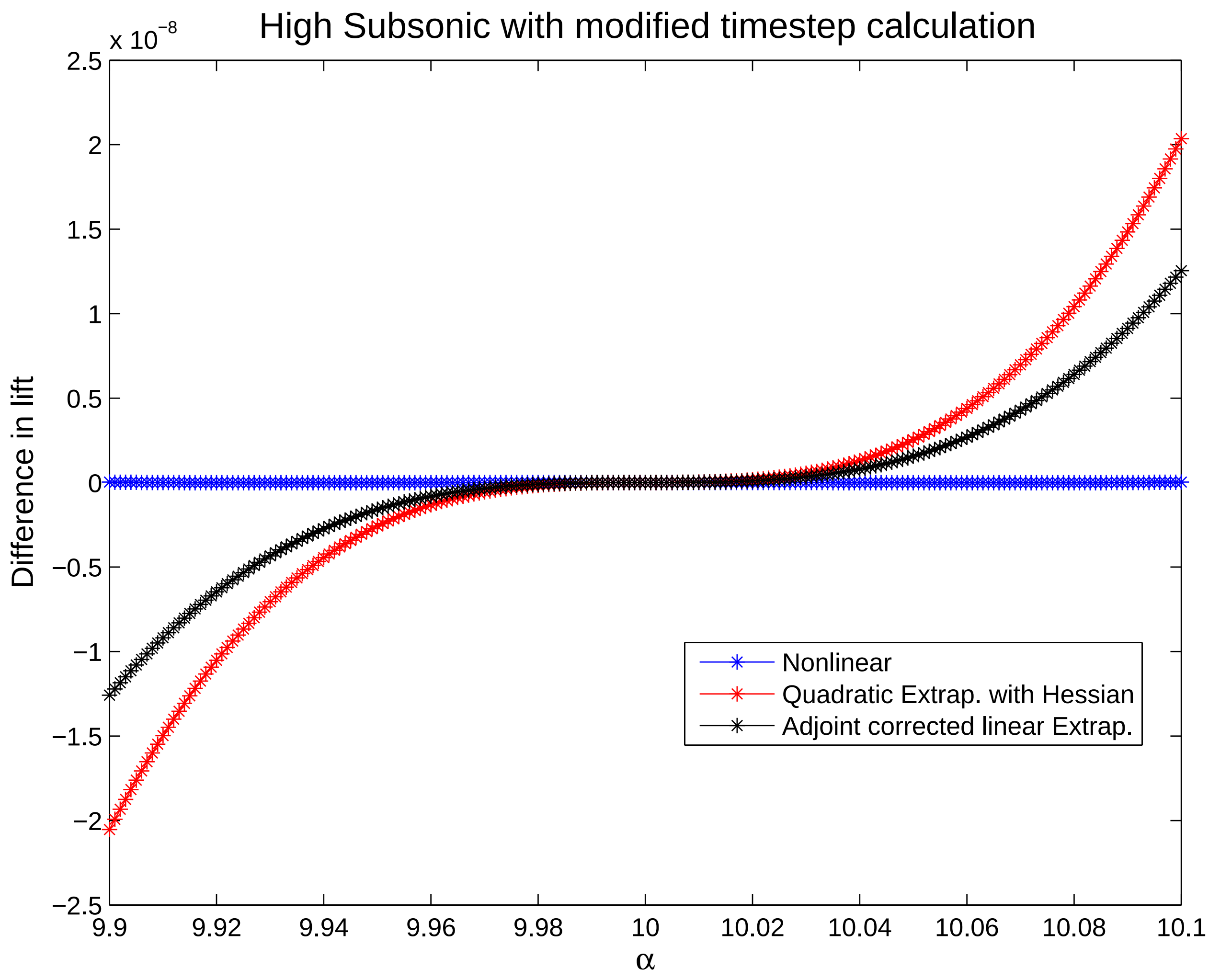
Extrapolation



- Time-step calculation included term $abs(v_x dy - v_y dx)$
- Sign change in this term at $\alpha = 9.995^o$ at one or more cells
- The term modified to $\sqrt{(v_x\ dy - v_y\ dx)^2 + \epsilon^2}$, where $\epsilon = 0.1\ c\ ds$

Extrapolation

High subsonic test case (Freestream mach = 0.65, AOA = 10^o)



Conclusion

- A computationally cheap and accurate method for Hessian calculation is demonstrated
  - For applications with expensive iterations might be more efficient than forward-on-reverse calculations, and simpler to implement
- Extrapolation
  - Linear extrapolation with adjoint correction is more accurate and robust than the quadratic extrapolation using Hessian