Airfoil shape optimization using adjoint method and automatic differentiation

Praveen. C
praveen@math.tifrbng.res.in

Tata Institute of Fundamental Research
Center for Applicable Mathematics
Bangalore 560065
http://math.tifrbng.res.in

AeSI CFD Symposium
11-12 August, 2009

Objectives and controls

- Objective function $\mathfrak{I}(\beta) = \mathfrak{I}(\beta, Q)$
  mathematical representation of system performance
- Control variables β
  - Parametric controls $\beta \in \mathbb{R}^n$
  - Infinite dimensional controls $\beta : X \to Y$
  - Shape $\beta \in$ set of admissible shapes
- State variable Q: solution of an ODE or PDE

$$R(\beta, Q) = 0 \quad \Longrightarrow \quad Q = Q(\beta)$$

Mathematical formulation

- Constrained minimization problem

$$\boxed{\min_{\beta} \mathfrak{I}(\beta, Q) \quad \text{subject to} \quad R(\beta, Q) = 0}$$

- Find $\delta\beta$ such that $\delta\mathfrak{I} < 0$ (to first order)

$$\delta\mathfrak{I} = \frac{\partial \mathfrak{I}}{\partial \beta}\delta\beta + \frac{\partial \mathfrak{I}}{\partial Q}\delta Q = \underbrace{\left[\frac{\partial \mathfrak{I}}{\partial \beta} + \frac{\partial \mathfrak{I}}{\partial Q}\frac{\partial Q}{\partial \beta}\right]}_{G}\delta\beta$$

- Steepest descent

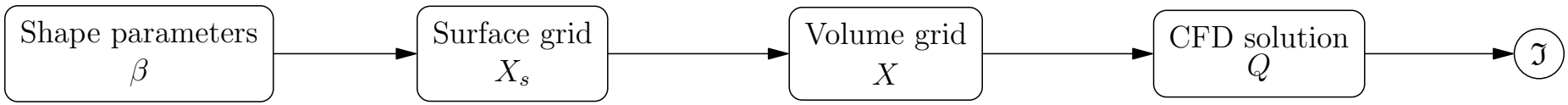
$$\delta\beta = -\epsilon G^{\top}, \quad \epsilon > 0$$

$$\delta\mathfrak{I} = -\epsilon G G^{\top} = -\epsilon \|G\|^2 < 0$$

How to compute gradient G cheaply and accurately ?

Elements of shape optimization

1. Shape parameterization
2. Surface grid generation/deformation
3. Domain grid generation/deformation
4. Flow solution (Euler/Navier-Stokes solver)
5. Adjoint flow solution
6. Optimization method



$$\frac{\mathrm{d}\mathfrak{I}}{\mathrm{d}\beta} = \frac{\mathrm{d}\mathfrak{I}}{\mathrm{d}X}\frac{\mathrm{d}X}{\mathrm{d}X_s}\frac{\mathrm{d}X_s}{\mathrm{d}\beta}$$

Adjoint approach

- For shape optimization: $\mathfrak{I} = \mathfrak{I}(X, Q)$

$$\frac{\mathrm{d}\mathfrak{I}}{\mathrm{d}X} = \frac{\partial \mathfrak{I}}{\partial X} + \frac{\partial \mathfrak{I}}{\partial Q}\frac{\partial Q}{\partial X}$$

- Flow sensitivity $\frac{\partial Q}{\partial X}$; costly to evaluate
- Differentiate state equation $R(X, Q) = 0$

$$\frac{\partial R}{\partial X} + \frac{\partial R}{\partial Q}\frac{\partial Q}{\partial X} = 0$$

- Introducing an *adjoint variable* Ψ, we can write

$$\frac{\mathrm{d}\mathfrak{I}}{\mathrm{d}X} = \frac{\partial \mathfrak{I}}{\partial X} + \frac{\partial \mathfrak{I}}{\partial Q}\frac{\partial Q}{\partial X} + \Psi^{\top}\left[\frac{\partial R}{\partial X} + \frac{\partial R}{\partial Q}\frac{\partial Q}{\partial X}\right]$$

Adjoint approach

- Collect terms involving the flow sensitivity

$$\frac{\mathrm{d}\mathfrak{I}}{\mathrm{d}X} = \frac{\partial \mathfrak{I}}{\partial X} + \Psi^{\top} \frac{\partial R}{\partial X} + \left[\frac{\partial \mathfrak{I}}{\partial Q} + \Psi^{\top} \frac{\partial R}{\partial Q} \right] \frac{\partial Q}{\partial X}$$

- Choose Ψ so that flow sensitivity vanishes

$$\frac{\partial \mathfrak{I}}{\partial Q} + \Psi^{\top} \frac{\partial R}{\partial Q} = 0 \quad \text{or} \quad \left(\frac{\partial R}{\partial Q} \right)^{\top} \Psi + \left(\frac{\partial \mathfrak{I}}{\partial Q} \right)^{\top} = 0$$

- Gradient

$$\frac{\mathrm{d}\mathfrak{I}}{\mathrm{d}X} = \frac{\partial \mathfrak{I}}{\partial X} + \Psi^{\top} \frac{\partial R}{\partial X}$$

Optimization steps

- $\beta \Longrightarrow X_s \Longrightarrow X$
- Solve the flow equations to steady-state

$$\boxed{\frac{\mathrm{d}Q}{\mathrm{d}t} + R(X,Q) = 0} \quad \Longrightarrow \quad Q, \quad \mathfrak{I}(X,Q)$$

- Solve adjoint equations to steady-state

$$\boxed{\frac{\mathrm{d}\Psi}{\mathrm{d}t} + \left(\frac{\partial R}{\partial Q}\right)^\top \Psi + \left(\frac{\partial \mathfrak{I}}{\partial Q}\right)^\top = 0} \quad \Longrightarrow \quad \Psi$$

- Compute gradient wrt grid X

$$\frac{\mathrm{d}\mathfrak{I}}{\mathrm{d}X} = \frac{\partial \mathfrak{I}}{\partial X} + \Psi^\top \frac{\partial R}{\partial X}$$

$$\frac{\mathrm{d}\mathfrak{I}}{\mathrm{d}\beta} = \frac{\mathrm{d}\mathfrak{I}}{\mathrm{d}X}\frac{\mathrm{d}X}{\mathrm{d}X_s}\frac{\mathrm{d}X_s}{\mathrm{d}\beta} \quad \Longrightarrow \quad \beta \longleftarrow \beta - \epsilon \frac{\mathrm{d}\mathfrak{I}}{\mathrm{d}\beta}$$

Continuous and discrete approaches

- Continuous approach (differentiate and discretize)
  PDE $\longrightarrow$ Adjoint PDE $\longrightarrow$ Discrete adjoint
- Discrete approach (discretize and differentiate)
  PDE $\longrightarrow$ Discrete PDE $\longrightarrow$ Discrete adjoint
- We use the discrete approach
  - $\boxed{R(X,Q)=0}$ represent the finite volume equations which are algebraic equations
  - Use ordinary calculus to differentiate
  - Need to compute

$$\frac{\partial \mathcal{I}}{\partial Q}, \quad \frac{\partial \mathcal{I}}{\partial X}, \quad \left(\frac{\partial R}{\partial Q}\right)^{\top} \Psi, \quad \left(\frac{\partial R}{\partial X}\right)^{\top} \Psi$$

Automatic differentiation

- Computer code available to compute

$$\mathfrak{I}(X, Q), \quad R(X, Q)$$

- Code is made of composition of elementary functions

$$T_0 = X, \quad T_r = F_r(T_{r-1})$$
$$Y = F(X) = F_p \circ F_{p-1} \circ \ldots \circ F_1(T_0)$$

  - Use differentiation by parts formula

$$\dot{Y} = F'(X)\dot{X} = F'_p(T_{p-1})F'_{p-1}(T_{p-2}) \ldots F'_1(T_0)\dot{X}$$

  - Automated using AD tools

Reverse differentiation

- Reverse mode computes transpose: $(X, \bar{Y}) \longrightarrow \bar{X}$

$$\bar{X} = [F'(X)]^\top \bar{Y} = [F_1'(T_0)]^\top [F_2'(T_1)]^\top \ldots [F_p'(T_{p-1})]^\top \bar{Y}$$

- Forward sweep and then reverse sweep

$$\begin{array}{lccccccc} \text{Func:} & T_0 & \longrightarrow & F_1 & \xrightarrow{T_1} & F_2 & \xrightarrow{T_2} \ldots \xrightarrow{T_{p-1}} & F_p \\ \text{Grad:} & T_{p-1}, \bar{Y} & \longrightarrow & [F_p']^T & \xrightarrow{\bar{T}_{p-2}} & [F_{p-1}']^T & \xrightarrow{\bar{T}_{p-3}} \ldots \xrightarrow{\bar{T}_0} & [F_1']^T \end{array}$$

Forward variables T_j required in reverse order: store or recompute

- Reverse mode useful to compute

$$\left(\frac{\partial \mathfrak{I}}{\partial Q}\right)^\top, \quad \left(\frac{\partial \mathfrak{I}}{\partial X}\right)^\top, \quad \left(\frac{\partial R}{\partial Q}\right)^\top \Psi, \quad \left(\frac{\partial R}{\partial X}\right)^\top \Psi$$

Differentiation: Example

- A simple example

$$f = (xy + \sin x + 4)(3y^2 + 6)$$

- Computer code, $f = t_{10}$

$$\begin{aligned}
t_1 &= x \\
t_2 &= y \\
t_3 &= t_1 t_2 \\
t_4 &= \sin t_1 \\
t_5 &= t_3 + t_4 \\
t_6 &= t_5 + 4 \\
t_7 &= t_2^2 \\
t_8 &= 3t_7 \\
t_9 &= t_8 + 6 \\
t_{10} &= t_6 t_9
\end{aligned}$$

F77 code: costfunc.f

```
      subroutine costfunc(x, y, f)
      t1  = x
      t2  = y
      t3  = t1*t2
      t4  = sin(t1)
      t5  = t3 + t4
      t6  = t5 + 4
      t7  = t2**2
      t8  = 3.0*t7
      t9  = t8 + 6.0
      t10 = t6*t9
      f   = t10
      end
```

Automatic Differentiation: Reverse mode

```
SUBROUTINE COSTFUNC_B(x, xb, y, yb, f, fb)
t1 = x
t2 = y
t3 = t1*t2
t4 = SIN(t1)
t5 = t3 + t4
t6 = t5 + 4
t7 = t2**2
t8 = 3.0*t7
t9 = t8 + 6.0
t10b = fb
t6b = t9*t10b
t9b = t6*t10b
t8b = t9b
t7b = 3.0*t8b
t5b = t6b
t3b = t5b
t2b = t1*t3b + 2*t2*t7b
t4b = t5b
t1b = t2*t3b + COS(t1)*t4b
yb = t2b
xb = t1b
fb = 0.0
END
```

Implementation of AD

- CFD code written with many subroutines
- Subroutines differentiated individually
- Then assembled together to form adjoint solver
- Only non-linear portions differentiated with AD
  - numerical flux (Roe)
  - Limiters
- Linear portions differentiated manually
- Leads to an efficient code with less memory requirements

Shape parameterization

- Parameterize the deformations

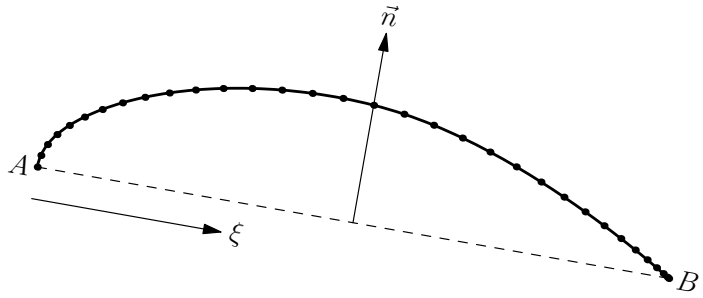
$$\begin{bmatrix} x_s \\ y_s \end{bmatrix} = \begin{bmatrix} x_s^{(0)} \\ y_s^{(0)} \end{bmatrix} + \begin{bmatrix} n_x \\ n_y \end{bmatrix} h(\xi)$$

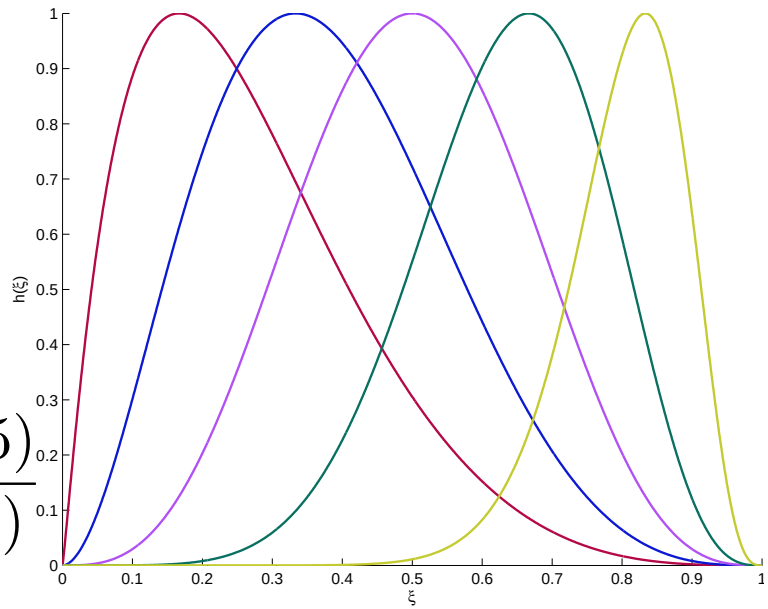
$$h(\xi) = \sum_{k=1}^{m} \beta_k B_k(\xi)$$

- Hicks-Henne bump functions

$$B_k(\xi) = \sin^p(\pi \xi^{q_k}), \quad q_k = \frac{\log(0.5)}{\log(\xi_k)}$$

- Move points along normal to reference line AB





Exact derivatives $\frac{\mathrm{d}X_s}{\mathrm{d}\beta}$ can be computed

Grid deformation

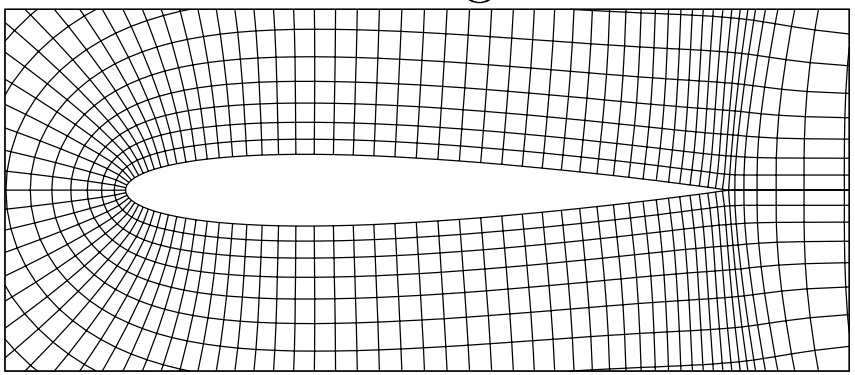
- Interpolate displacement of surface points to interior points using RBF

$$\tilde{f}(x,y) = a_0 + a_1 x + a_2 y + \sum_{j=1}^{N} b_j |\vec{r} - \vec{r}_j|^2 \log |\vec{r} - \vec{r}_j|$$

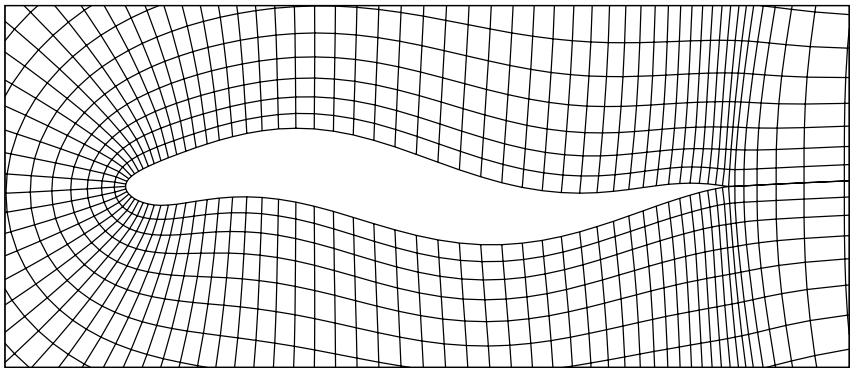
where $\vec{r} = (x, y)$

- Results in smooth grids
- Exact derivatives $\frac{\mathrm{d}X}{\mathrm{d}X_s}$ can be computed

Initial grid



Deformed grid



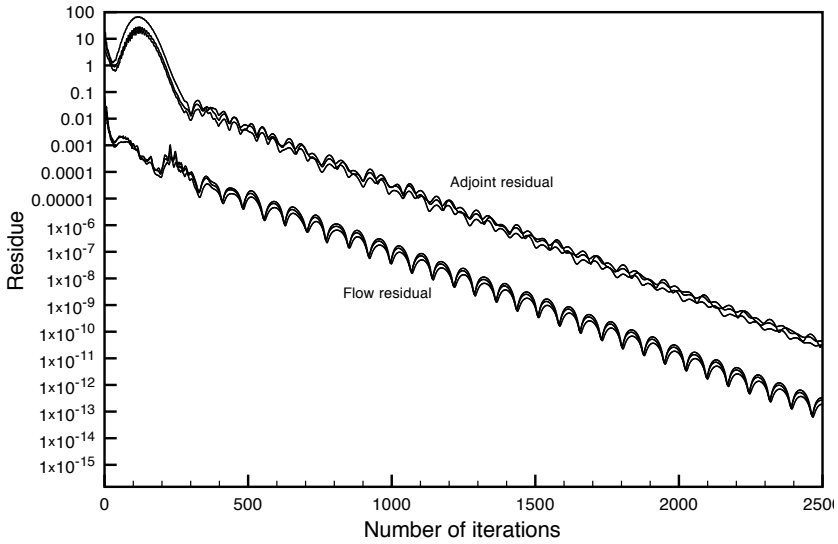
NUWTUN flow solver

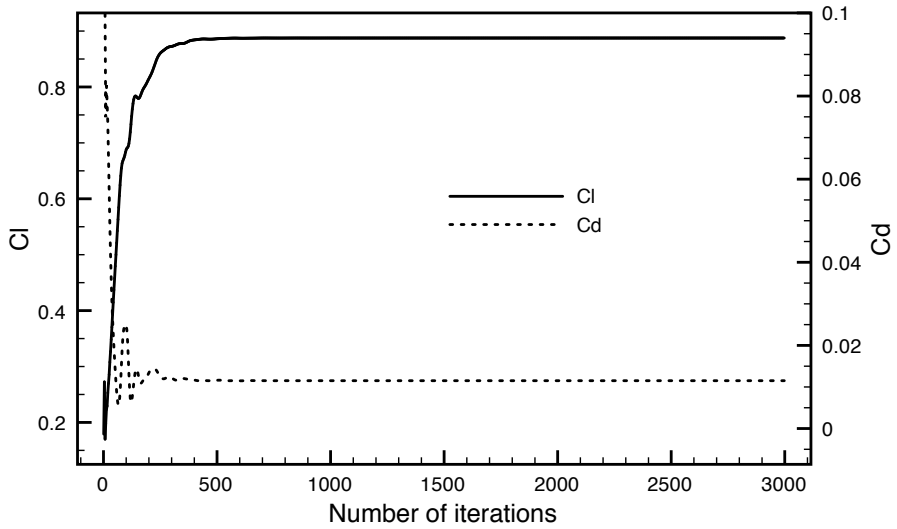
Based on the ISAAC code of Joseph Morrison
`http://isaac-cfd.sourceforge.net`

- Finite volume scheme
- Structured, multi-block grids
- Roe flux
- MUSCL reconstruction with Koren limiter
- Implicit scheme

Source code of NUWTUN available online
`http://nuwtun.berlios.de`

Convergence tests

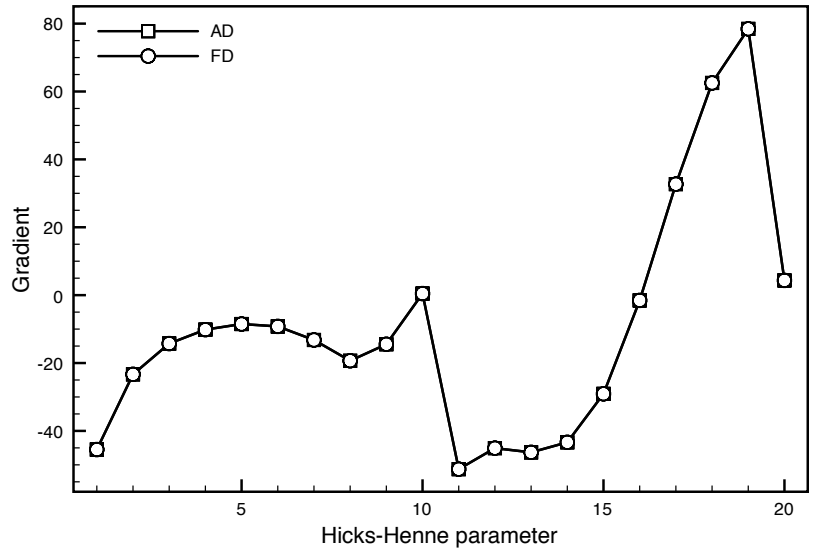


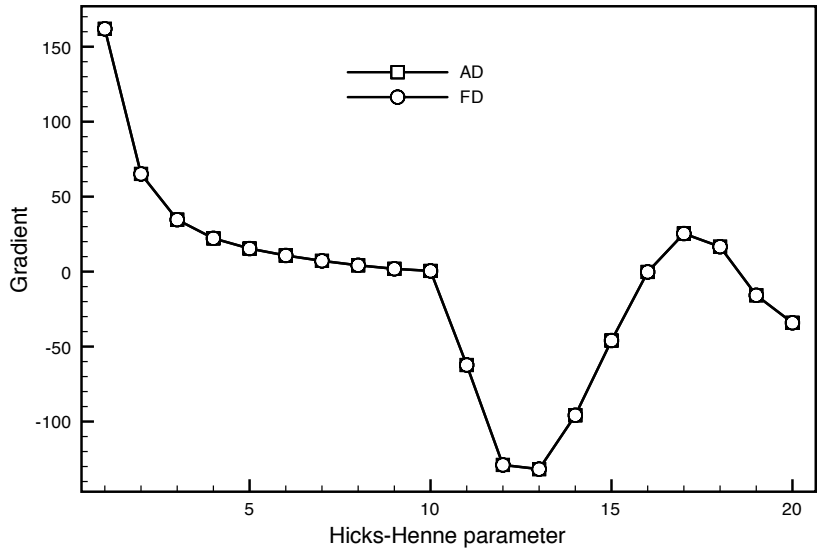


Convergence characteristics for the flow and adjoint solutions, and convergence of lift and drag coefficients, for RAE2822 airfoil at $M_\infty = 0.73$

Validation of adjoint gradients

- Dot-product test
- Limiters can cause non-differentiability
- Koren limiter: dependance on parameter
- Check adjoint derivatives against finite difference





Test cases

- NACA0012: $M_\infty = 0.8$, $\alpha = 1.25^o$

$$\mathfrak{I} = \frac{C_d}{C_l}\frac{C_{l_0}}{C_{d_0}}$$

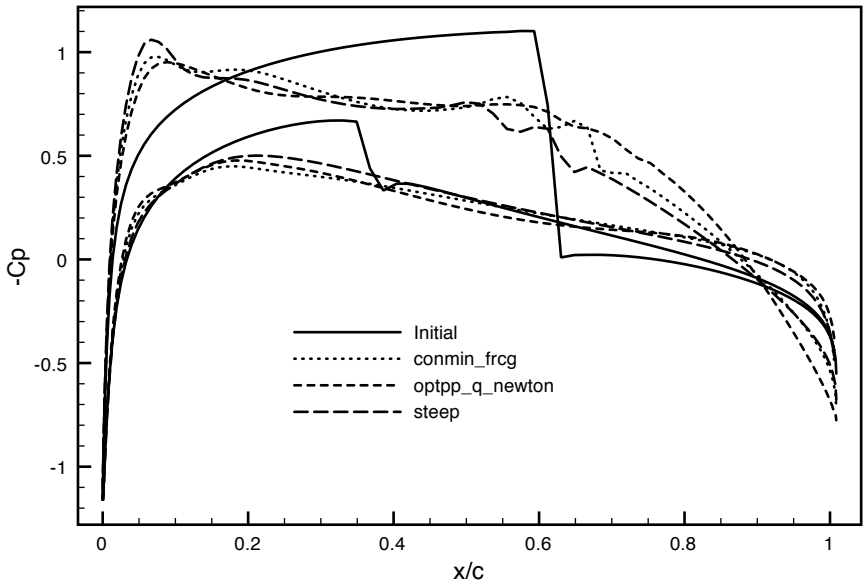
- RAE2822: $M_\infty = 0.729$, $\alpha = 2.31^o$
  - Penalty approach

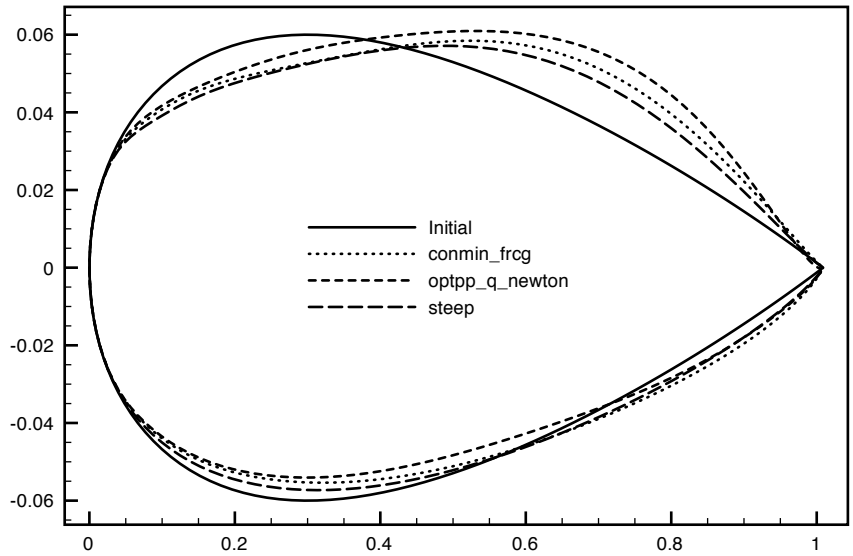
$$\mathfrak{I} = \frac{C_d}{C_{d_0}} + \left|1 - \frac{C_l}{C_{l_0}}\right|$$

  - Constrained minimization

$$\min \mathfrak{I} = \frac{C_d}{C_{d_0}} \quad s.t. \quad C_l = C_{l_0}$$

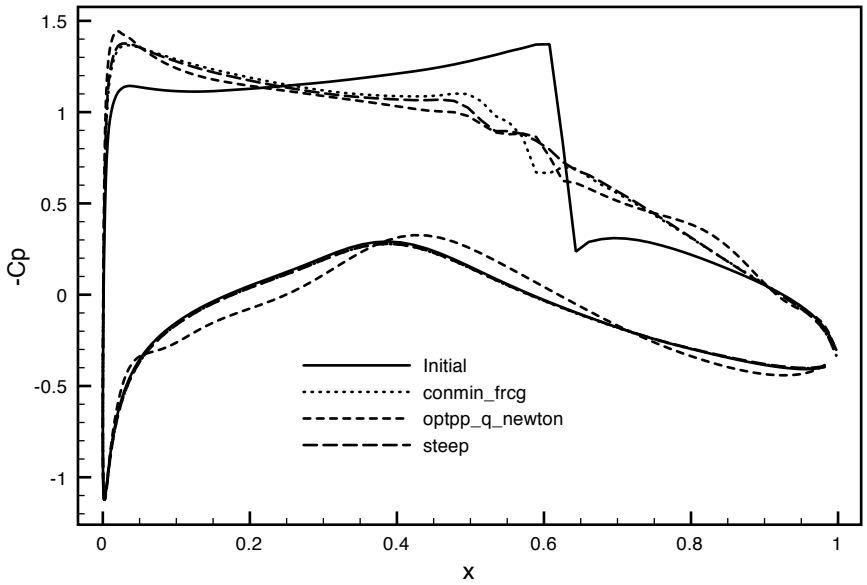
NACA0012: Maximize L/D

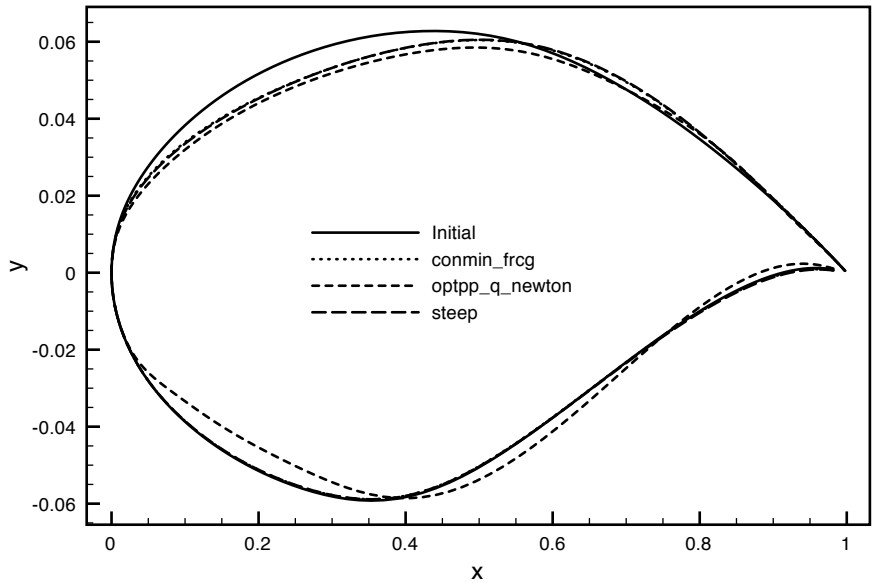




Method	$\mathfrak{I}$	$100C_d$	C_l	N_{fun}	N_{grad}
Initial	1.000	2.072	0.295	-	-
`conmin_frcg`	0.170	0.389	0.325	50	13
`optpp_q_newton`	0.142	0.329	0.329	50	39
`steep`	0.166	0.335	0.287	50	45

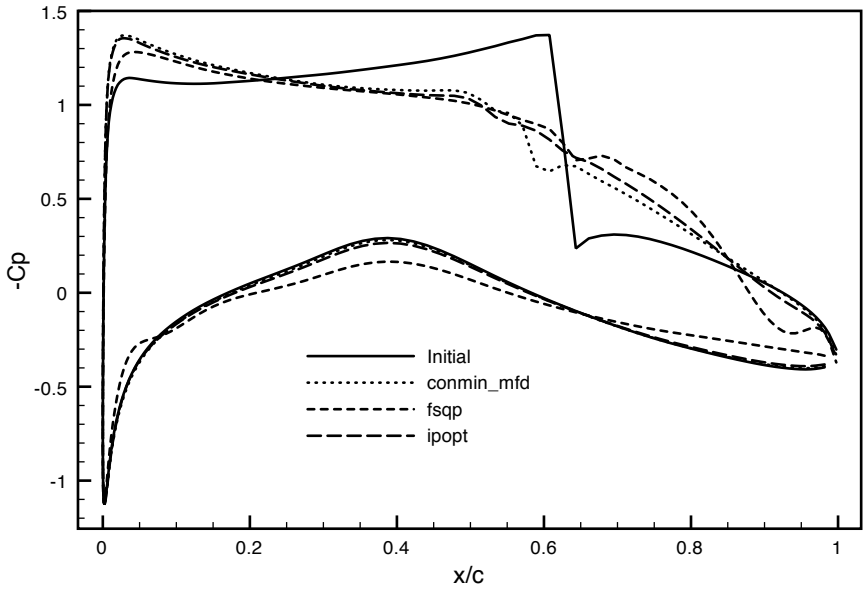
RAE2822: Drag minimization, penalty approach

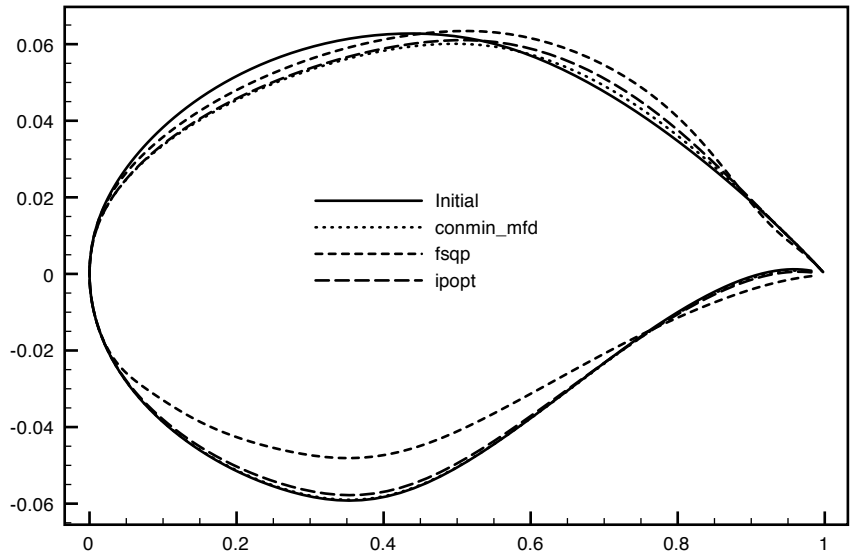




Method	$\mathfrak{I}$	$100C_d$	C_l	N_{fun}	N_{grad}
Initial	1.000	1.150	0.887	-	-
`conmin_frcg`	0.355	0.405	0.890	50	13
`optpp_q_newton`	0.351	0.400	0.884	50	51
`steep`	0.341	0.388	0.884	50	47

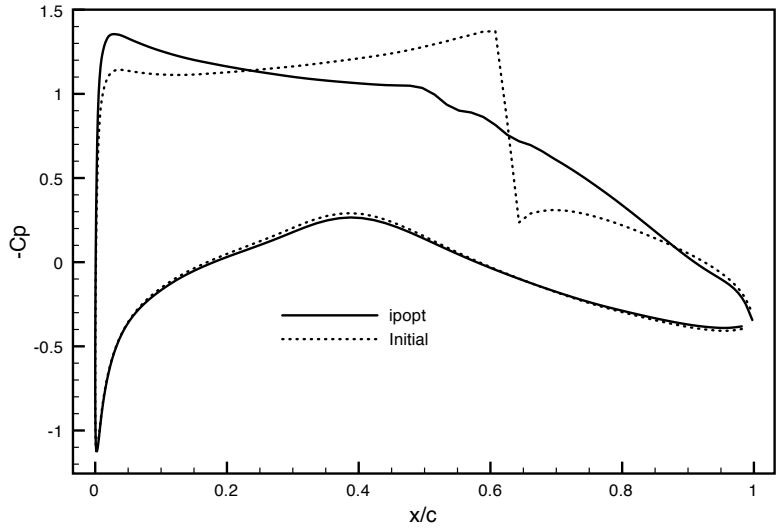
RAE2822: Lift-constrained drag minimization

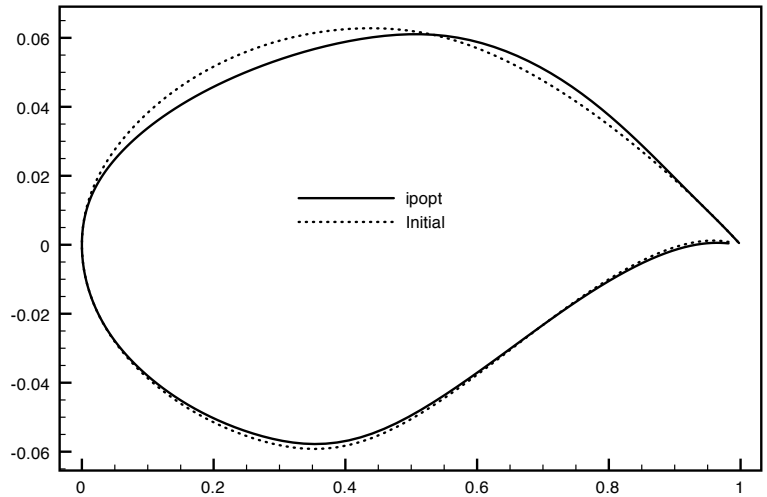


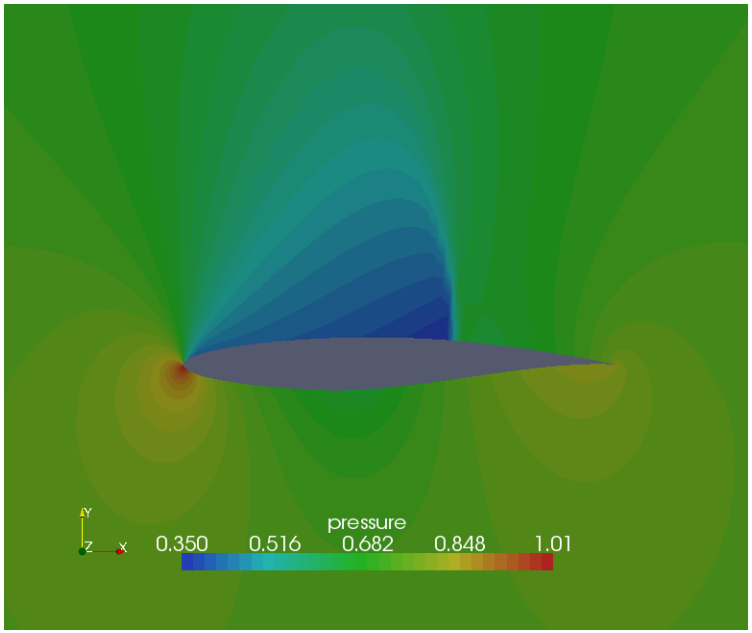


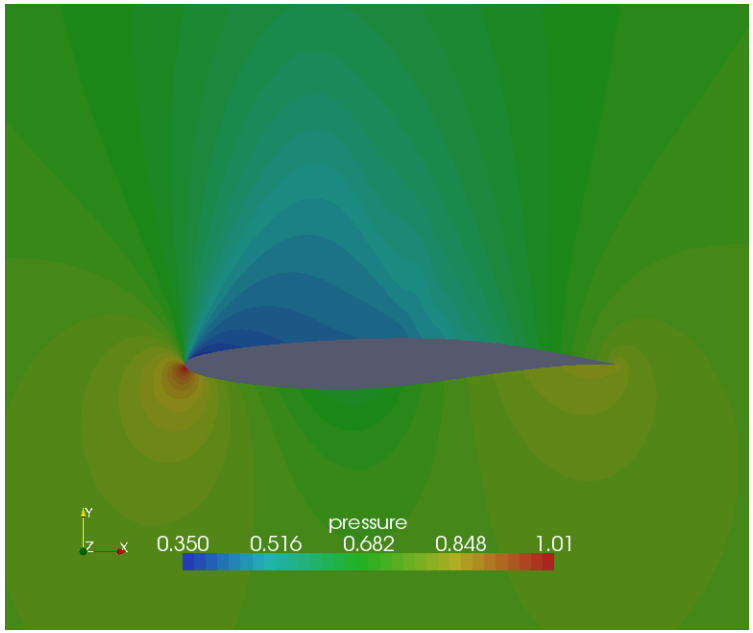
Method	$\mathfrak{I}$	$100C_d$	C_l	N_{fun}	N_{grad}
Initial	1.000	1.150	0.887	-	-
`conmin_mfd`	0.342	0.394	0.881	50	22
`fsqp`	0.325	0.374	0.887	50	32
`ipopt`	0.335	0.385	0.887	45	62

RAE2822: Lift-constrained drag minimization

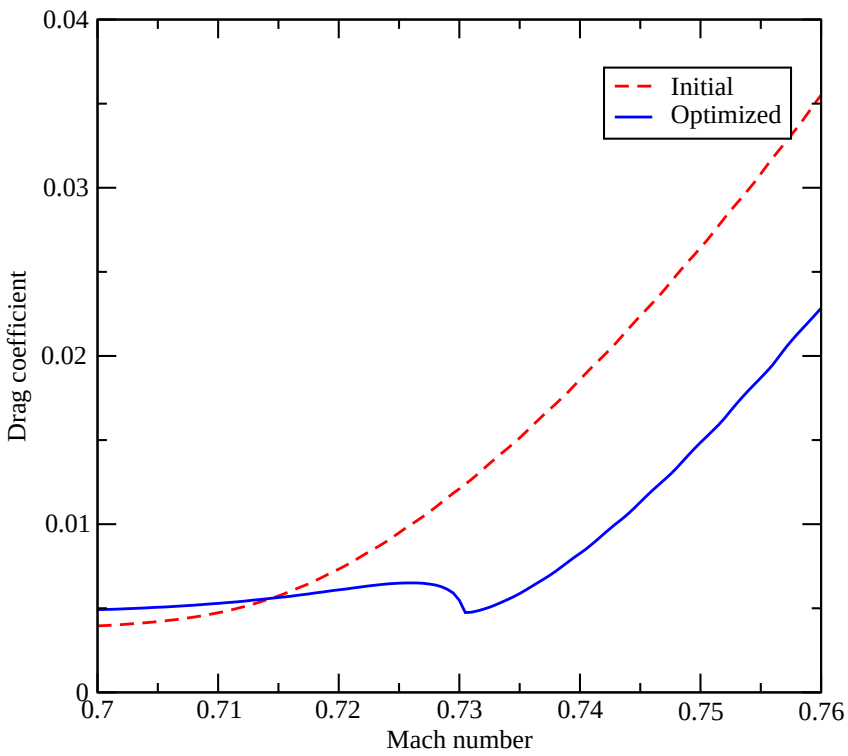




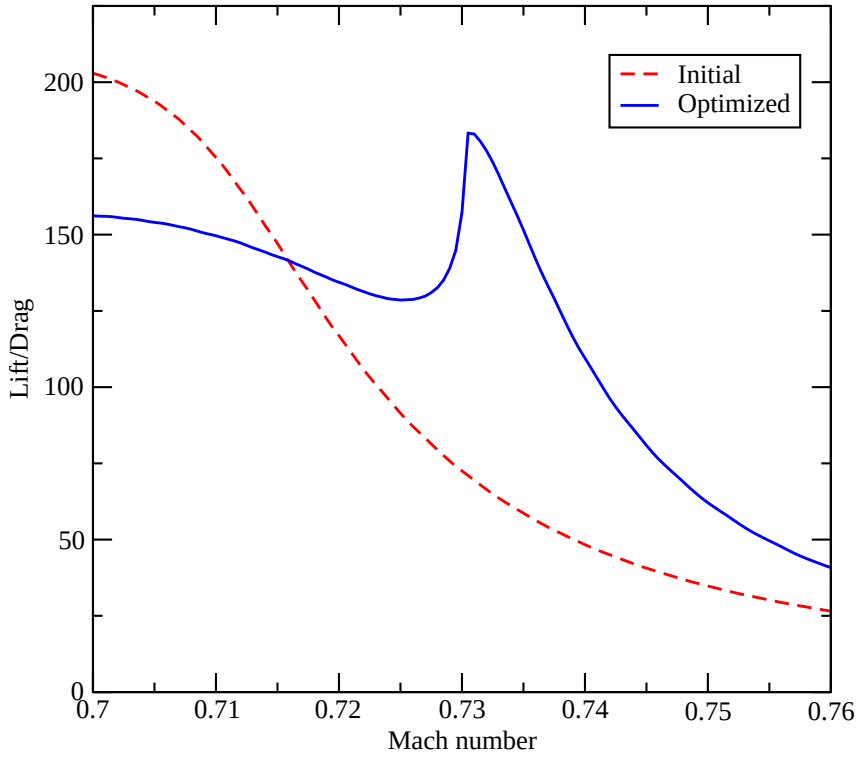




Sensitivity to perturbations



(a)



(b)

Variation of (a) drag coefficient and (b) L/D with Mach number for RAE2822 airfoil and optimized airfoil

Need for robust aerodynamic optimization