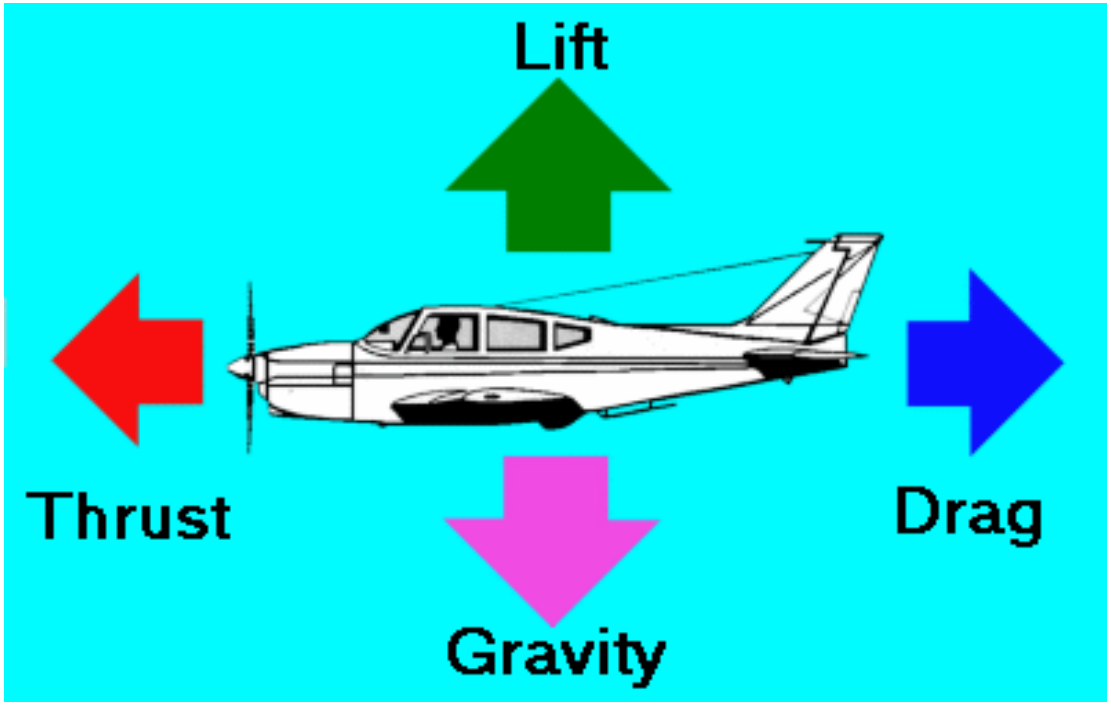
Numerical Shape Optimization

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

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

IMI Workshop on Control and Inverse Problems
IISc, Bangalore
1-15 December, 2009

Aerodynamic shape optimization

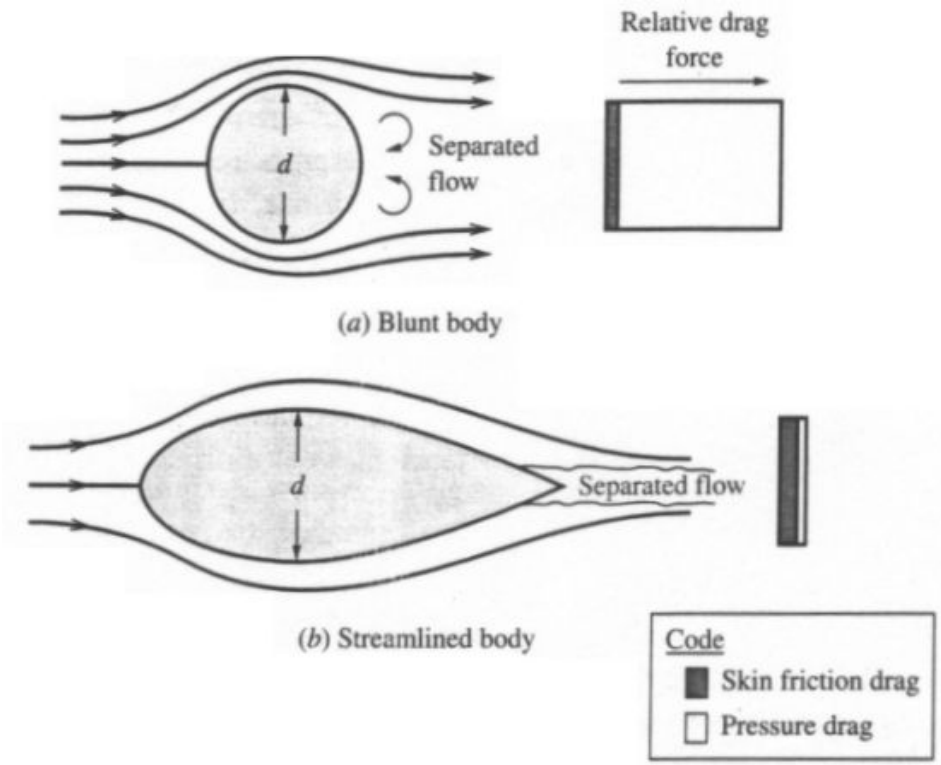


http://www.aviation-history.com

$$\begin{aligned} Lift &= Weight \\ Drag &= Thrust \end{aligned}$$

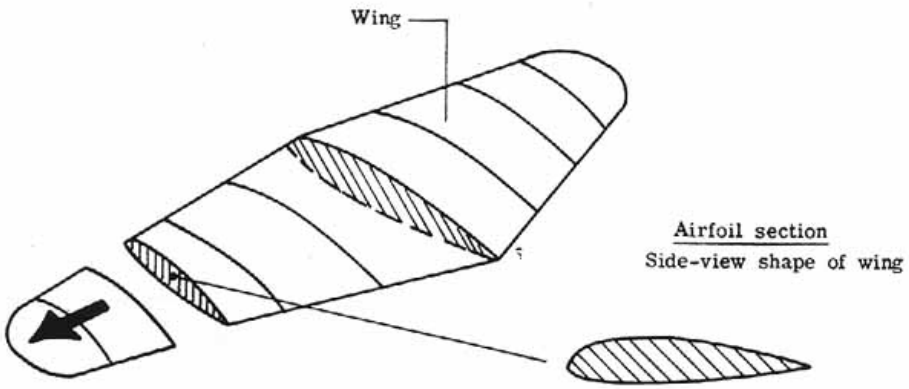
$$\boxed{\min_{\beta} Drag(\beta), \quad \text{such that} \quad Lift(\beta) = W}$$

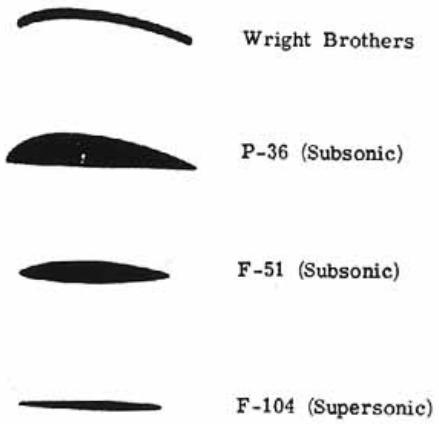
Effect of shape on flow



http://www.aerospaceweb.org

Wing and Airfoils

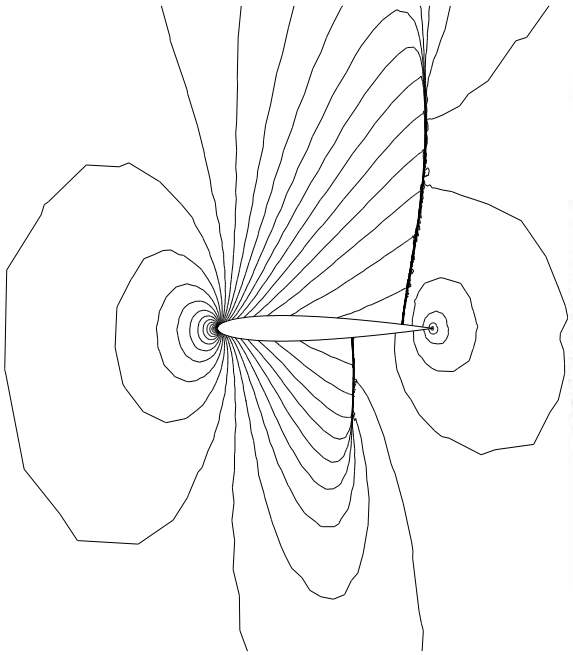


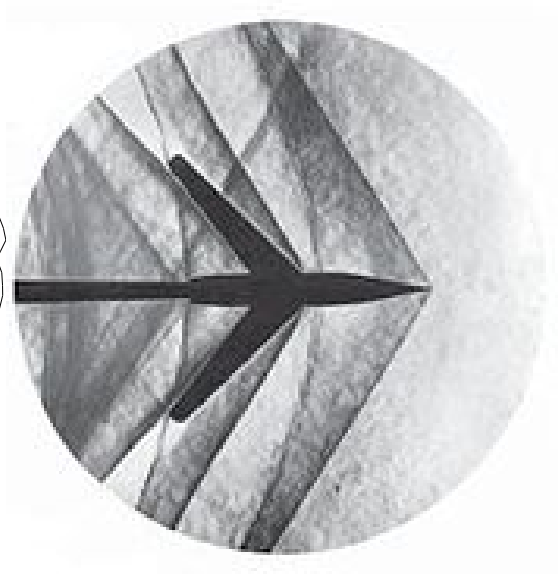


http://www.centennialofflight.gov

Compressible flow and shocks

$$\text{Mach number, } M = \frac{\text{speed of air}}{\text{speed of sound}}$$





$$\text{Range, } R = M \frac{a}{c_T} \frac{C_L}{C_D} \log \frac{W_i}{W_f}$$

Objectives and controls

- Objective function $\mathfrak{I}(\beta)$
  mathematical representation of system performance

$$\mathfrak{I}(\beta) = \mathfrak{I}(\beta, Q(\beta))$$

- Control variables β

$$\beta \text{ represents the shape}$$

  We want to minimize/maximize $\mathfrak{I}$ wrt β
- State variable Q: solution of an ODE or PDE

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

Basic problem

How to find the derivative of the objective function $\mathfrak{I}$ wrt the design variables (shape) β ?

Gradient-based minimization: blackbox/FD approach

$$\boxed{\min_{\beta \in \mathbb{R}^N} \mathfrak{I}(\beta, Q(\beta))}$$

- Initialize β^0, $n = 0$
- For $n = 0, \ldots, N_{iter}$
  - Solve $R(\beta^n, Q^n) = 0$
  - For $j = 1, \ldots, N$
    - ★ $\beta^n_{(j)} = [\beta^n_1, \ldots, \beta^n_j + \Delta\beta_j, \ldots, \beta^n_N]^\top$
    - ★ Solve $R(\beta^n_{(j)}, Q^n_{(j)}) = 0$
    - ★ $\frac{\mathrm{d}\mathfrak{I}}{\mathrm{d}\beta_j} \approx \frac{\mathfrak{I}(\beta^n_{(j)}, Q^n_{(j)}) - \mathfrak{I}(\beta^n, Q^n)}{\Delta\beta_j}$
  - Steepest descent step

$$\beta^{n+1} = \beta^n - s^n \frac{\mathrm{d}\mathfrak{I}}{\mathrm{d}\beta}(\beta^n)$$

Cost of FD-based steepest-descent

$$\text{Cost} = O(N+1)N_{iter} = O(N+1)O(N) = O(N^2)$$

Accuracy of FD: Choice of step size

$$\frac{\mathrm{d}}{\mathrm{d}x} f(x_0) = \frac{f(x_0+\delta) - f(x_0)}{\delta} + O(\delta)$$

In principle, if we choose a small δ, we reduce the error.

But computers have <u>finite precision</u>. Instead of $f(x_0)$ the computers gives $f(x_0) + O(\epsilon)$ where ϵ = machine precision.

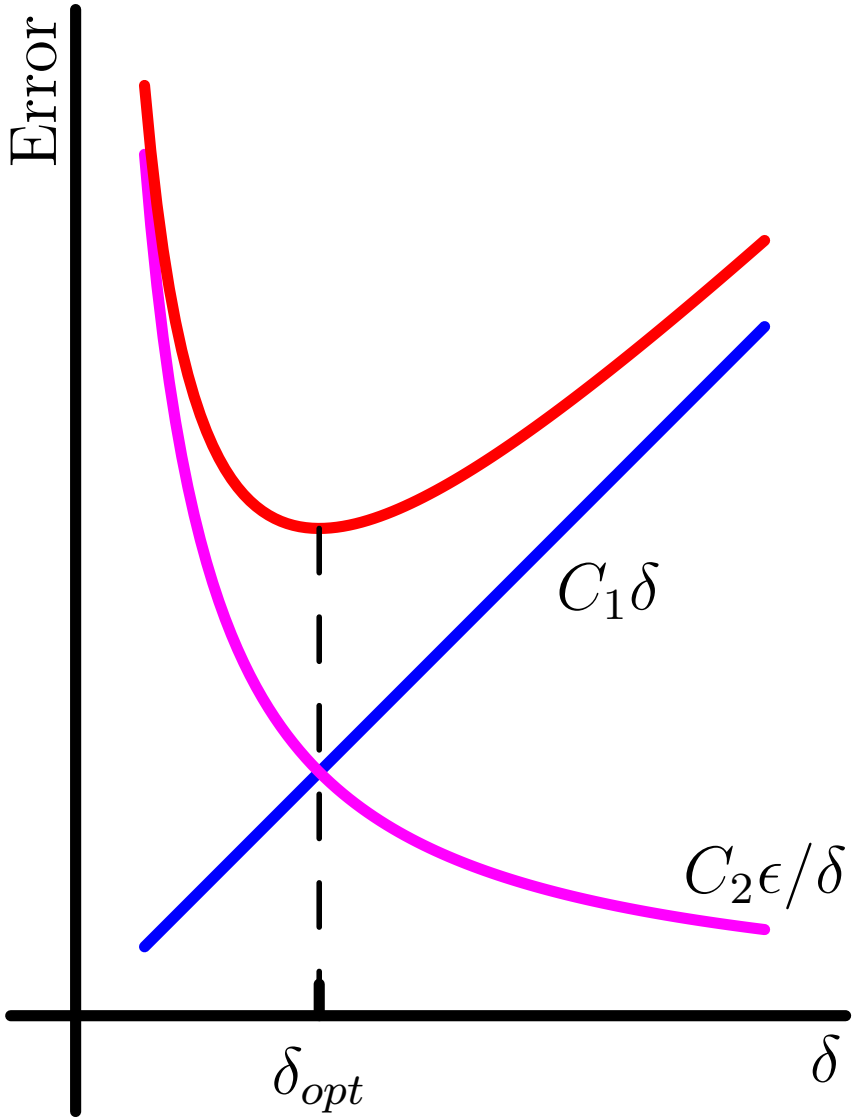
$$\frac{[f(x_0+\delta) + O(\epsilon)] - [f(x_0) + O(\epsilon)]}{\delta} = \frac{f(x_0+\delta) - f(x_0)}{\delta} + C_1 \frac{\epsilon}{\delta}$$

$$= \frac{\mathrm{d}}{\mathrm{d}x} f(x_0) + \underbrace{C_2\delta + C_1\frac{\epsilon}{\delta}}_{\text{Total error}}, \qquad C_1, C_2 \text{ depend on } f, x_0, \text{precision}$$

Total error is least when

$$\delta = \delta_{\text{opt}} = C_3\sqrt{\epsilon}, \quad C_3 = \sqrt{C_1/C_2}$$

Accuracy of FD: Choice of step size

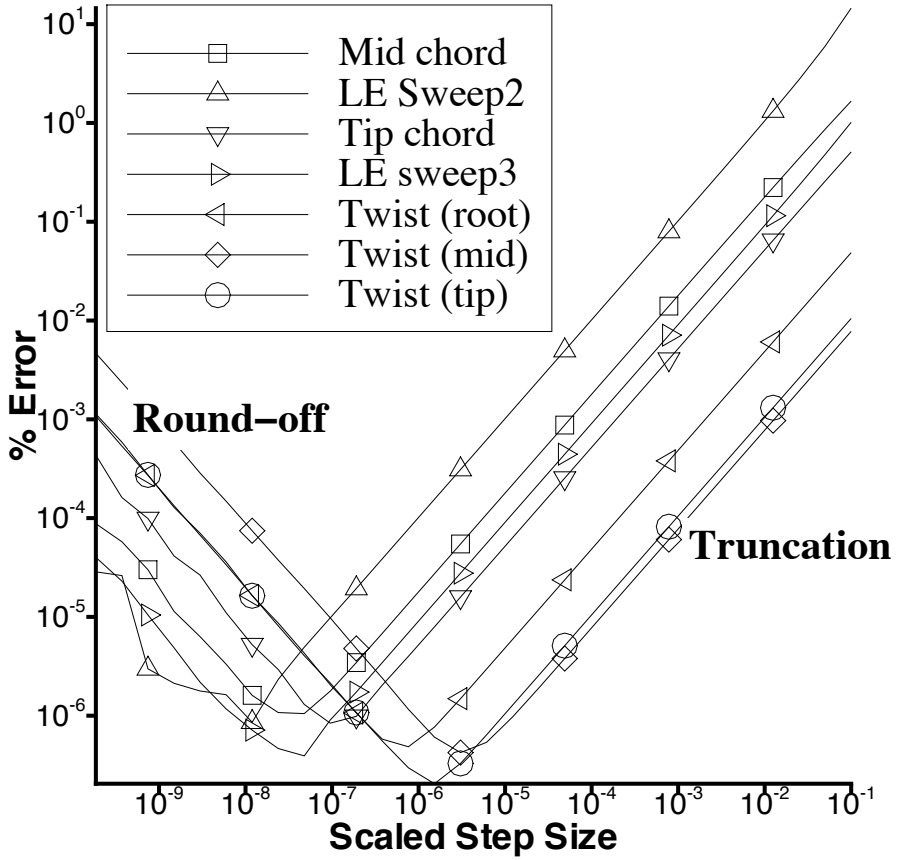


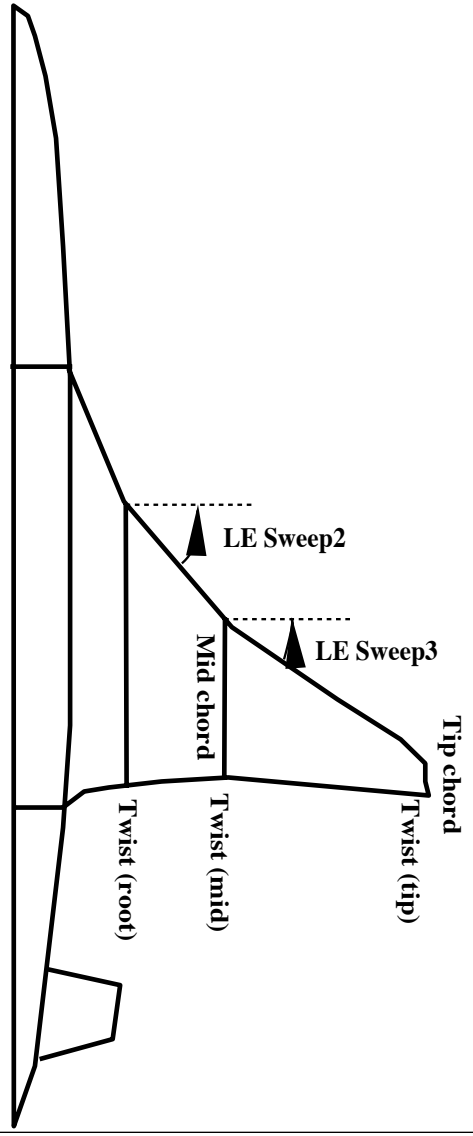
Precision	ϵ	δ_{opt}
Single	10^{-8}	10^{-4}
Double	10^{-16}	10^{-8}

See:
Brian J. McCartin, *Seven Deadly Sins of Numerical Computation* The American Mathematical Monthly, Vol. 105, No. 10 (Dec., 1998), pp. 929-941

Drag gradient using FD (Samareh)

Errors in Finite Difference Approximations





Iterative problems

$$\mathfrak{I}(\beta, Q), \quad \text{where} \quad R(\beta, Q) = 0$$

- Q is implicitly defined, require an iterative solution method.
- Assume a Q^0 and iterate $Q^n \longrightarrow Q^{n+1}$ until $||R(\beta, Q^n)|| \leq \texttt{TOL}$
- If `TOL` is too small, need too many iterations
- Many problems, we cannot reduce $||R(\beta, Q^n)||$ to small values
- This means that numerical value of $\mathfrak{I}$ will be noisy
  Finite difference will contain too much error, and is useless

RAE5243 airfoil, Mach=0.68, Re=19 million, AOA=2.5 deg.

iter	Lift	Drag
41496	0.824485788042416	1.627593747613790E-002
41497	0.824485782714867	1.627593516695762E-002
41498	0.824485777387834	1.627593285794193E-002
41499	0.824485772061306	1.627593054909022E-002
41500	0.824485766735297	1.627592824040324E-002

Complex variable method

$$f(x_0 + i\delta) = f(x_0) + i\delta f'(x_0) + O(\delta^2) + iO(\delta^3)$$
$$f'(x_0) = \frac{1}{\delta}\text{imag}[f(x_0 + i\delta)] + O(\delta^2)$$

- No roundoff error
- We can take δ to be very small, $\delta = 10^{-20}$
- Can be easily implemented
  - fortran: redeclare real variables as complex
  - matlab: no change
- Iterative problems: $\beta \longrightarrow \beta + i\Delta\beta$
  - Obtain $\tilde{Q} = Q(\beta + i\Delta\beta)$ by solving $R(\beta + i\Delta\beta, \tilde{Q}) = 0$
  - Then gradient

$$\mathfrak{I}'(\beta) \approx \frac{1}{\Delta\beta}\text{imag}[\mathfrak{I}(\beta + i\Delta\beta, Q(\beta + i\Delta\beta)]$$

- Computational cost is $O(N^2)$ or higher (due to complex arithmetic)

Sensitivity equation approach

- PDE constrained optimization problem

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

- Derivative of $J(\beta) = J(\beta, Q(\beta))$

$$J'(\beta)\tilde{\beta} = \lim_{\epsilon \to 0} \frac{J(\beta + \epsilon\tilde{\beta}) - J(\beta)}{\epsilon} = J_\beta \tilde{\beta} + J_Q \tilde{Q}$$

- Differentiating the state equation

$$R_\beta \tilde{\beta} + R_Q \tilde{Q} = 0$$

- Suppose $\beta \in \mathbb{R}^N$. To compute $\frac{\mathrm{d}J}{\mathrm{d}\beta_i}$
  - Set $\tilde{\beta}_{(i)} = [0, \ldots, 0, 1, 0, \ldots, 0]$
  - Solve for $\tilde{Q}_{(i)}$ from $R_\beta \tilde{\beta}_{(i)} + R_Q \tilde{Q}_{(i)} = 0$
  - Compute $\frac{\mathrm{d}J}{\mathrm{d}\beta_i} = J_\beta \tilde{\beta}_{(i)} + J_Q \tilde{Q}_{(i)}$
  - No problem of roundoff error
  - Cost of gradient is $O(N^2)$

Adjoint equation approach

$$J'(\beta)\tilde{\beta} = J_\beta \tilde{\beta} + J_Q \tilde{Q}, \quad R_\beta \tilde{\beta} + R_Q \tilde{Q} = 0$$

Introduce the adjoint variable Ψ

$$\begin{aligned} J'(\beta)\tilde{\beta} &= J_\beta \tilde{\beta} + J_Q \tilde{Q} + (R_\beta \tilde{\beta} + R_Q \tilde{Q}, \Psi) \\ &= (J_\beta \tilde{\beta}, 1) + (J_Q \tilde{Q}, 1) + (R_\beta \tilde{\beta} + R_Q \tilde{Q}, \Psi) \\ &= (\tilde{\beta}, J_\beta^*) + (\tilde{Q}, J_Q^*) + (\tilde{\beta}, R_\beta^* \Psi) + (\tilde{Q}, R_Q^* \Psi) \\ &= (\tilde{\beta}, J_\beta^* + R_\beta^* \Psi) + (\tilde{Q}, J_Q^* + R_Q^* \Psi) \end{aligned}$$

Since Ψ was arbitrary, and we want to eliminate $\tilde{Q}$, we can set

$$\boxed{J_Q^* + R_Q^* \Psi = 0}$$

Then the derivative is given by

$$\boxed{J'(\beta)\tilde{\beta} = (\tilde{\beta}, J_\beta^* + R_\beta^* \Psi)}$$

Adjoint approach

The adjoint equation

$$\boxed{R_Q^* \Psi + J_Q^* = 0}$$

does not depend on the design variables β. Irrespective of the number of design variables, we have to solve just one adjoint equation.

The gradient can be computed cheaply from

$$\boxed{J'(\beta)\tilde{\beta} = (J_\beta^* + R_\beta^* \Psi, \tilde{\beta})}$$

which only requires scalar products.

Thus the cost of adjoint approach to compute gradient is $O(1)$, i.e., it is <u>independent</u> of the number of design variables.

One-shot versus iterative methods

- <u>One-shot method</u>: solve for (Q, Ψ, β) simultaneously[1]

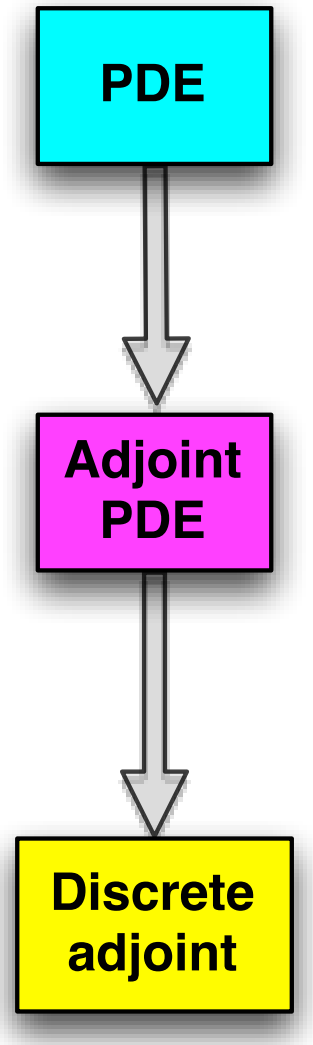
$$\begin{aligned} R(\beta, Q) &= 0 \\ R_Q^* \Psi + J_Q^* &= 0 \\ J_\beta^* + R_\beta^* \Psi &= 0 \end{aligned}$$

- <u>Iterative method</u>: steepest descent
  1. Initial guess for β
  2. Solve for Q from $R(\beta, Q) = 0$
  3. Solve for Ψ from $R_Q^* \Psi + J_Q^* = 0$
  4. Compute gradient $J'(\beta) = J_\beta^* + R_\beta^* \Psi$
  5. Update $\beta \longleftarrow \beta - \varepsilon J'(\beta)$
  6. If not converged, goto step 2

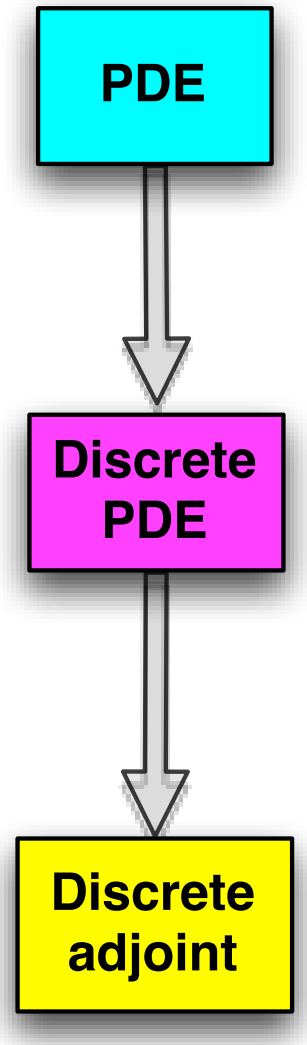
[1] Hazra and Schulz, Simultaneous pseudo-timestepping for aerodynamic shape optimization

Adjoint: Two approaches

Continuous or differentiate-discretize



Discrete or discretize-differentiate



Shape parameterization for gradient computation

- Mapping[2] a reference domain $\hat{\Omega} \subset \mathbb{R}^d$

$$T : \mathbb{R}^d \to \mathbb{R}^d, \quad T \in H^m(\mathbb{R}^d)$$

$$\text{Admissible shapes} = \{\Omega : \Omega = T(\hat{\Omega})\}$$

- Normal displacements
  - reference domain $\hat{\Omega}$
  - $\Gamma \subset \partial\hat{\Omega}$ to be optimized
  - perturb Γ by normal displacements

$$\Gamma^\epsilon = \{x + \epsilon\alpha(x)n_\Gamma(x) : x \in \Gamma\}, \quad \alpha \in H^m(\Gamma)$$

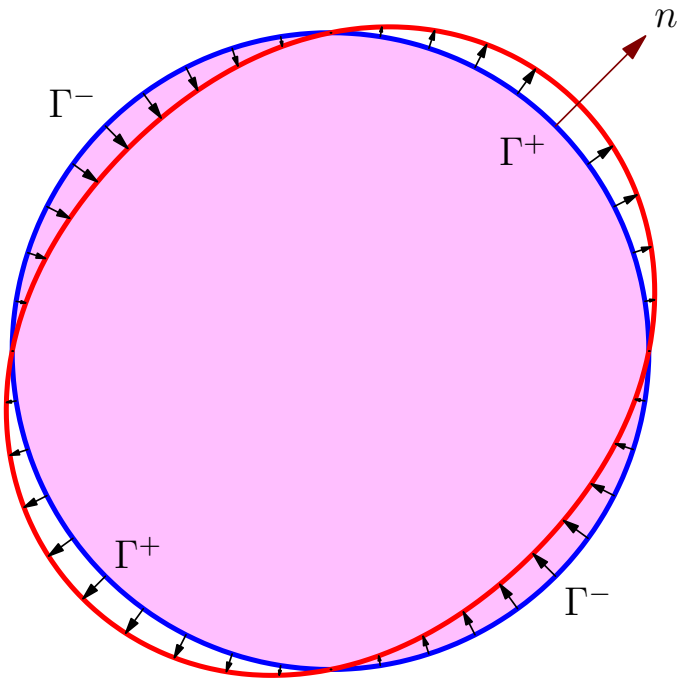
[2] Delfour & Zolesio, Shapes and geometries: Analysis, Differential Calculus and Optimization, SIAM

Derivative of volume functional

$$J(\Omega, u) = \int_{\Omega} F(u)\mathrm{d}x, \quad u = u(\Omega)$$

If A, B are two sets then $A = (A - B) \cup (A \cap B)$

$$J(\Omega_\epsilon, u^\epsilon) - J(\Omega, u) = \int\limits_{\Omega_\epsilon - \Omega} F(u^\epsilon)\mathrm{d}x - \int\limits_{\Omega - \Omega_\epsilon} F(u)\mathrm{d}x + \int\limits_{\Omega \cap \Omega_\epsilon} [F(u^\epsilon) - F(u)]\mathrm{d}x$$



Define the following boundary parts

$$\begin{aligned} \Gamma^+ &= \{s \in \Gamma : \alpha(s) > 0\} \\ \Gamma^- &= \{s \in \Gamma : \alpha(s) \leq 0\} \end{aligned}$$

Then we approximate the integrals as follows:

$$\int\limits_{\Omega_\epsilon - \Omega} F(u^\epsilon)\mathrm{d}x = +\epsilon \int_{\Gamma^+} \alpha(s) F(u)\mathrm{d}s + O(\epsilon^2)$$

$$\int\limits_{\Omega - \Omega_\epsilon} F(u)\mathrm{d}x = -\epsilon \int_{\Gamma^-} \alpha(s) F(u)\mathrm{d}s + O(\epsilon^2)$$

Define the sensitivity of u

$$\tilde{u} = \lim_{\epsilon \to 0} \frac{1}{\epsilon}(u^\epsilon - u) \quad \text{or} \quad u^\epsilon - u = \epsilon \tilde{u} + O(\epsilon^2)$$

Then

$$\int\limits_{\Omega \cap \Omega_\epsilon} [F(u^\epsilon) - F(u)]\mathrm{d}x = \epsilon \int_\Omega F'(u)\tilde{u}\mathrm{d}x + O(\epsilon^2)$$

Shape derivative is

$$\frac{1}{\epsilon}[J(\Omega_\epsilon) - J(\Omega)] = \int_\Gamma \alpha(s) F(u)\mathrm{d}s + \int_\Omega F'(u)\tilde{u}\mathrm{d}x + O(\epsilon)$$

Derivative of boundary functional

$$J(\Gamma, u) = \int_\Gamma f(u) \mathrm{d}s$$

We perturb Γ by normal displacements

$$\Gamma_\epsilon = \{x + \epsilon\alpha(s(x))n : x \in \Gamma\}$$

We must consider change in area element $\mathrm{d}s$ due to this perturbation. First look at a line element $\mathrm{d}s$ with radius of curvature R, $\mathrm{d}s = R\mathrm{d}\theta$ Due to boundary perturbation, line element changes to

$$\mathrm{d}s^\epsilon = \left(1 - \epsilon\frac{\alpha}{R}\right)\mathrm{d}s$$

Same formula holds in 3-D but R is mean curvature defined using two orthogonal tangential coordinates

$$\frac{1}{R} = \frac{1}{R_1} + \frac{1}{R_2}$$

First express $\int_{\Gamma_\epsilon} f(u^\epsilon)\mathrm{d}s$ in terms of integral and quantities on Γ
Consider $x \in \Gamma$ and corresponding $x^\epsilon \in \Gamma_\epsilon$ given by $x^\epsilon = x + \epsilon\alpha n$

$$
\begin{aligned}
u(x^\epsilon) &= u(x) + \epsilon\alpha\frac{\partial u}{\partial n}(x) + O(\epsilon^2) \\
u^\epsilon(x^\epsilon) &= u(x^\epsilon) + \epsilon\tilde{u}(x^\epsilon) + O(\epsilon^2) \\
&= u(x) + \epsilon\alpha\frac{\partial u}{\partial n}(x) + \epsilon\tilde{u}(x) + O(\epsilon^2) \\
f(u^\epsilon(x^\epsilon)) &= f\left(u(x) + \epsilon\alpha\frac{\partial u}{\partial n}(x) + \epsilon\tilde{u}(x)\right) + O(\epsilon^2) \\
&= f(u) + \epsilon\left(\alpha\frac{\partial u}{\partial n} + \tilde{u}\right)f'(u) + O(\epsilon^2) \\
f(u^\epsilon)\mathrm{d}s^\epsilon &= f(u)\mathrm{d}s + \epsilon\left[\alpha f'(u)\frac{\partial u}{\partial n} - \frac{\alpha}{R}f(u)\right]\mathrm{d}s + \epsilon f'(u)\tilde{u} + O(\epsilon^2)
\end{aligned}
$$

Finally the sensitivity is given by

$$\frac{1}{\epsilon}[J(\Gamma_\epsilon) - J(\Gamma)] = \int_\Gamma \alpha \left[f'(u)\frac{\partial u}{\partial n} - \frac{f(u)}{R} \right] \mathrm{d}s + \int_\Gamma f'(u)\tilde{u}\mathrm{d}s + O(\epsilon)$$

Euler equations: compressible, inviscid flow

$$\begin{aligned} \rho &= \text{density} \\ u &= (u_1, u_2, u_3) = \text{velocity} \\ p &= \text{pressure} \end{aligned}$$

Total energy per unit volume

$$E = \rho e + \frac{1}{2}\rho |u|^2$$

$$e = \text{internal energy per unit mass}$$

For an ideal gas,

$$e = \frac{p}{(\gamma - 1)\rho} \quad \Longrightarrow \quad E = \frac{p}{\gamma - 1} + \frac{1}{2}\rho |u|^2$$

$$\gamma = \frac{C_p}{C_v} \qquad \text{For air } \gamma = 1.4$$

Conservation of mass

Rate of change of mass inside volume Ω = − flux of mass going out through $\partial\Omega$

$$\frac{\mathrm{d}}{\mathrm{d}t}\int_\Omega \rho \mathrm{d}x = -\oint_{\partial\Omega} \rho(u\cdot n)\mathrm{d}s$$

Using divergence theorem

$$\frac{\mathrm{d}}{\mathrm{d}t}\int_\Omega \rho \mathrm{d}x = -\int_\Omega \mathrm{div}(\rho u)\mathrm{d}x$$

Since this should hold for any control volume Ω

$$\boxed{\frac{\partial \rho}{\partial t} + \mathrm{div}(\rho u) = 0}$$

Conservation of momentum: Newton's law

We consider a material volume $\Omega(t)$, i.e, $\Omega(t)$ moves with the fluid

Rate of change of momentum = Net force acting on the body

$$\frac{\mathrm{d}}{\mathrm{d}t}\int_{\Omega(t)} \rho u_i \mathrm{d}x = \int_{\partial\Omega(t)} R_i \mathrm{d}s + \int_{\Omega(t)} \rho f_i \mathrm{d}x$$

For an inviscid fluid, $R = -pn$. The left hand side is

$$\begin{aligned}\frac{\mathrm{d}}{\mathrm{d}t}\int_{\Omega(t)} \rho u_i \mathrm{d}x &= \int_{\Omega(t)} \frac{\partial}{\partial t}(\rho u_i)\mathrm{d}x + \oint_{\partial\Omega(t)} (\rho u_i)(u\cdot n)\mathrm{d}s \\ &= \int_{\Omega(t)} \frac{\partial}{\partial t}(\rho u_i)\mathrm{d}x + \int_{\Omega(t)} \mathrm{div}(\rho u_i u)\mathrm{d}x\end{aligned}$$

$$\int_{\Omega(t)} \frac{\partial}{\partial t}(\rho u_i)\mathrm{d}x + \int_{\Omega(t)} \mathrm{div}(\rho u_i u)\mathrm{d}x = -\int_{\Omega} \frac{\partial p}{\partial x_i}\mathrm{d}x + \int_{\Omega} \rho f_i \mathrm{d}x$$

$$\boxed{\frac{\partial}{\partial t}(\rho u_i) + \mathrm{div}(\rho u_i u) + \frac{\partial p}{\partial x_i} = \rho f_i}$$

Conservation of energy

Rate of change of energy = rate of work done by all forces

$$\frac{\mathrm{d}}{\mathrm{d}t}\int_{\Omega(t)} E\mathrm{d}x = \oint_{\partial\Omega(t)} (R\cdot u)\mathrm{d}s + \int_{\Omega(t)} \rho(f\cdot u)\mathrm{d}x$$

Using divergence theorem

$$\int_{\Omega} \frac{\partial E}{\partial t}\mathrm{d}x + \int_{\Omega} \mathrm{div}(Eu)\mathrm{d}x = -\int_{\Omega} \mathrm{div}(pu)\mathrm{d}x + \int_{\Omega(t)} \rho(f\cdot u)\mathrm{d}x$$

$$\boxed{\frac{\partial E}{\partial t} + \mathrm{div}[(E+p)u] = \rho f\cdot u}$$

Euler equations

- Mass conservation

$$\frac{\partial \rho}{\partial t} + \frac{\partial}{\partial x_1}(\rho u_1) + \frac{\partial}{\partial x_2}(\rho u_2) + \frac{\partial}{\partial x_3}(\rho u_3) = 0$$

- Momentum conservation

$$\begin{aligned}
\frac{\partial}{\partial t}(\rho u_1) + \frac{\partial}{\partial x_1}(p + \rho u_1^2) + \frac{\partial}{\partial x_2}(\rho u_1 u_2) + \frac{\partial}{\partial x_3}(\rho u_1 u_3) &= 0 \\
\frac{\partial}{\partial t}(\rho u_2) + \frac{\partial}{\partial x_1}(\rho u_1 u_2) + \frac{\partial}{\partial x_2}(p + \rho u_2^2) + \frac{\partial}{\partial x_3}(\rho u_2 u_3) &= 0 \\
\frac{\partial}{\partial t}(\rho u_3) + \frac{\partial}{\partial x_1}(\rho u_1 u_3) + \frac{\partial}{\partial x_2}(\rho u_2 u_3) + \frac{\partial}{\partial x_3}(p + \rho u_3^2) &= 0
\end{aligned}$$

- Energy conservation

$$\frac{\partial E}{\partial t} + \frac{\partial}{\partial x_1}[(E + p)u_1] + \frac{\partial}{\partial x_2}[(E + p)u_2] + \frac{\partial}{\partial x_3}[(E + p)u_3] = 0$$

Euler equations: vector conservation form

$$\boxed{\frac{\partial U}{\partial t} + \frac{\partial F_1}{\partial x_1} + \frac{\partial F_2}{\partial x_2} + \frac{\partial F_3}{\partial x_3} = 0}$$

where U is the conserved variables

$$U = \begin{bmatrix} \rho \\ \rho u_1 \\ \rho u_2 \\ \rho u_3 \\ E \end{bmatrix}$$

and (F_1, F_2, F_3) are the flux vectors

$$F_1 = \begin{bmatrix} \rho u_1 \\ p + \rho u_1^2 \\ \rho u_1 u_2 \\ \rho u_1 u_3 \\ (E+p)u_1 \end{bmatrix}, \quad F_2 = \begin{bmatrix} \rho u_2 \\ \rho u_1 u_2 \\ p + \rho u_2^2 \\ \rho u_2 u_3 \\ (E+p)u_2 \end{bmatrix}, \quad F_3 = \begin{bmatrix} \rho u_3 \\ \rho u_1 u_3 \\ \rho u_2 u_3 \\ p + \rho u_3^2 \\ (E+p)u_3 \end{bmatrix}$$

Euler equations: Hyperbolic property

- Flux jacobians

$$A = \frac{\partial F_1}{\partial U}, \quad B = \frac{\partial F_2}{\partial U}, \quad C = \frac{\partial F_3}{\partial U}$$

- Quasi-linear form

$$\frac{\partial U}{\partial t} + A\frac{\partial U}{\partial x_1} + B\frac{\partial U}{\partial x_2} + C\frac{\partial U}{\partial x_3} = 0$$

- Hyperbolic property: $A(U,\omega) = \omega_1 A + \omega_2 B + \omega_3 C, \quad \omega \in \mathbb{R}^3$
  - All eigenvalues are real
  - Eigenvectors are linearly independent
- Eigenvalues of $A(U,\omega)$ with $|\omega| = 1$ are

$$u \cdot \omega - a, \quad u \cdot \omega, \quad u \cdot \omega, \quad u \cdot \omega, \quad u \cdot \omega + a$$

$$a = \text{speed of sound} = \sqrt{\frac{\gamma p}{\rho}}$$

Shape design using Euler equations

Pressure matching problem

Find the shape of the airfoil Γ such that the pressure on the airfoil is close to a specified pressure p^*, i.e.,

$$\min_{\Gamma} \frac{1}{2} \int_{\Gamma} (p - p^*)^2 \mathrm{d}s$$

The pressure $p = p(U)$ is obtained from the solution of the steady Euler equations

$$F_x + G_y + H_z = 0$$

The fluxes also satisfy the homogeneity property

$$F(U) = A(U)U, \quad G(U) = B(U)U, \quad H(U) = C(U)U$$

Due to change in shape Γ, the solution changes from U to $U + \epsilon\tilde{U}$

$$\begin{aligned} F(U + \epsilon\tilde{U}) &= F(U) + \epsilon\frac{\partial F}{\partial U}(U)\tilde{U} + O(\epsilon^2) \\ &= F(U) + \epsilon A\tilde{U} + O(\epsilon^2) \end{aligned}$$

Equation governing first order perturbation

$$(A\tilde{U})_x + (B\tilde{U})_y + (C\tilde{U})_z = 0$$

For an arbitrary function V

$$\begin{aligned} \int_\Omega V \cdot (A\tilde{U})_x &= -\int_\Omega V_x \cdot A\tilde{U} + \int_{\partial\Omega} V \cdot A\tilde{U} n_x \\ &= -\int_\Omega A^{\mathrm{t}} V_x \cdot \tilde{U} + \int_{\partial\Omega} V \cdot A\tilde{U} n_x \end{aligned}$$

Then

$$\int_\Omega V \cdot [(A\tilde{U})_x + (B\tilde{U})_y + (C\tilde{U})_z] = 0$$

becomes

$$
\begin{aligned}
& -\int_{\Omega} (A^{\mathrm{t}} V_x + B^{\mathrm{t}} V_y + C^{\mathrm{t}} V_z) \cdot \tilde{U} \\
+ & \int_{\partial\Omega} V \cdot (A n_x + B n_y + C n_z) \tilde{U} = 0
\end{aligned}
$$

Define

$$D = A n_x + B n_y + C n_z$$

we note that

$$DU = \begin{bmatrix} \rho(u \cdot n) \\ pn + \rho u (u \cdot n) \\ (E + p)(u \cdot n) \end{bmatrix} = \text{normal flux}$$

and on solid wall Γ, $u \cdot n = 0$

$$DU|_{\Gamma} = \begin{bmatrix} 0 \\ pn \\ 0 \end{bmatrix}$$

Since $DU = Fn_x + Gn_y + Hn_z$

$$\begin{aligned}\widetilde{DU} &= \tilde{F}n_x + \tilde{G}n_y + \tilde{H}n_z \\ &= A\tilde{U}n_x + B\tilde{U}n_y + C\tilde{U}n_z \\ &= D\tilde{U}\end{aligned}$$

Also

$$\widetilde{pn} = \tilde{p}n + p\tilde{n}, \quad \tilde{n} = -\sum_{j=1}^{d-1} \frac{\partial \alpha}{\partial t_j} t_j$$

Denoting $V = [v_1, v, v_5]$ with $v = (v_2, v_3, v_4)$

$$\begin{aligned}\int_\Gamma V \cdot D\tilde{U} &= \int_\Gamma V \cdot \widetilde{DU} \\ &= \int_\Gamma V \cdot [0 \quad \widetilde{pn} \quad 0] = \int_\Gamma v \cdot \widetilde{pn} \\ &= \int_\Gamma \tilde{p}(v \cdot n) - \int_\Gamma p \sum_{j=1}^{d-1} \frac{\partial \alpha}{\partial t_j}(v \cdot t_j)\end{aligned}$$

First variation in objective function

$$\frac{1}{\epsilon}\tilde{J} = \int_\Gamma \left[(p - p^*)\tilde{p} + \alpha(p - p^*)\frac{\partial p}{\partial n} - \alpha\frac{(p - p^*)^2}{2R} \right]$$

To obtain gradient of J wrt α we must eliminate $\tilde{p}$. We add the first order perturbation equation to the above equation

$$\begin{aligned} \frac{1}{\epsilon}\tilde{J} &= RHS - \int_\Omega (A^{\mathrm{t}}V_x + B^{\mathrm{t}}V_y + C^{\mathrm{t}}V_z) \cdot \tilde{U} \\ &\quad + \int_{\partial\Omega} V \cdot D\tilde{U} \end{aligned}$$

To eliminate $\tilde{U}$ from the volume integral, we choose V to satisfy

$$\boxed{A^{\mathrm{t}}V_x + B^{\mathrm{t}}V_y + C^{\mathrm{t}}V_z = 0, \quad \text{in} \quad \Omega}$$

$$
\begin{aligned}
\frac{1}{\epsilon}\tilde{J} &= RHS + \int_\Gamma V \cdot D\tilde{U} + \int_{\Gamma_o} V \cdot D\tilde{U} \\
&= \int_\Gamma (p - p^* + v \cdot n)\tilde{p} \\
&+ \int_\Gamma \alpha \left[(p - p^*)\frac{\partial p}{\partial n} - \frac{(p - p^*)^2}{2R} \right] \\
&- \int_\Gamma p \sum_{j=1}^{d-1} \frac{\partial \alpha}{\partial t_j}(v \cdot t_j) \\
&+ \int_{\Gamma_o} D^{\mathrm{t}} V \cdot \tilde{U}
\end{aligned}
$$

To eliminate $\tilde{p}$ we choose the adjoint boundary conditions

$$
\boxed{p - p^* + v \cdot n = 0, \quad \text{on} \quad \Gamma}
$$

We choose V on Γ_o to eliminate the integral over Γ_o

- Supersonic inflow: U is specified so $\tilde{U} = 0$.
  No b.c. are imposed on V
- Supersonic outflow: No b.c. for U, hence $\tilde{U} \neq 0$.
  Choose $V = 0$
- Subsonic inflow:

$$\int_{\Gamma_o} D^{\mathrm{t}} V \cdot \tilde{U} = \int_{\Gamma_o} D^{\mathrm{t}} V \cdot T^{-1} T \tilde{U} = \int_{\Gamma_o} T^{-\mathrm{t}} D^{\mathrm{t}} V \cdot T \tilde{U}$$

  We have $(T\tilde{U})_{1,2,3,4} = 0$ while $(T\tilde{U})_5$ is arbitrary, so we choose $(T^{-\mathrm{t}} D^{\mathrm{t}} V)_5 = 0$
- Subsonic outflow: $(T\tilde{U})_1$ is specified while $(T\tilde{U})_{2,3,4,5}$ is arbitrary.
  Hence we choose $(T^{-\mathrm{t}} D^{\mathrm{t}} V)_{2,3,4,5} = 0$

Adjoint Euler equations

$$\boxed{A^{\mathrm{t}} V_x + B^{\mathrm{t}} V_y + C^{\mathrm{t}} V_z = 0, \quad \text{in} \quad \Omega}$$

with boundary conditions

- supersonic inflow

$$\text{none}$$

- supersonic outflow

$$V = 0$$

- subsonic inflow

$$(T^{-\mathrm{t}} D^{\mathrm{t}} V)_5 = 0$$

- subsonic outflow

$$(T^{-\mathrm{t}} D^{\mathrm{t}} V)_{2,3,4,5} = 0$$

- solid wall

$$p - p^* + v \cdot n = 0$$

Shape derivative

$$\begin{aligned}\frac{1}{\epsilon}\tilde{J} &= \int_\Gamma \alpha\left[(p-p^*)\frac{\partial p}{\partial n} - \frac{(p-p^*)^2}{2R}\right] - \int_\Gamma p\sum_{j=1}^{d-1}\frac{\partial \alpha}{\partial t_j}(v\cdot t_j) \\ &= \int_\Gamma \alpha\left[(p-p^*)\frac{\partial p}{\partial n} - \frac{(p-p^*)^2}{2R} - \mathrm{div}_\Gamma(pv)\right]\end{aligned}$$

Hence

$$\nabla_\alpha J = (p-p^*)\frac{\partial p}{\partial n} - \frac{(p-p^*)^2}{2R} - \mathrm{div}_\Gamma(pv)$$

Steepest descent update

$$\alpha^{n+1} = \alpha^n - s^n \nabla_\alpha J(\alpha^n)$$

Discrete adjoint approach

- Approximate PDE using <u>finite volume method</u>

$$R(X, Q) = 0$$

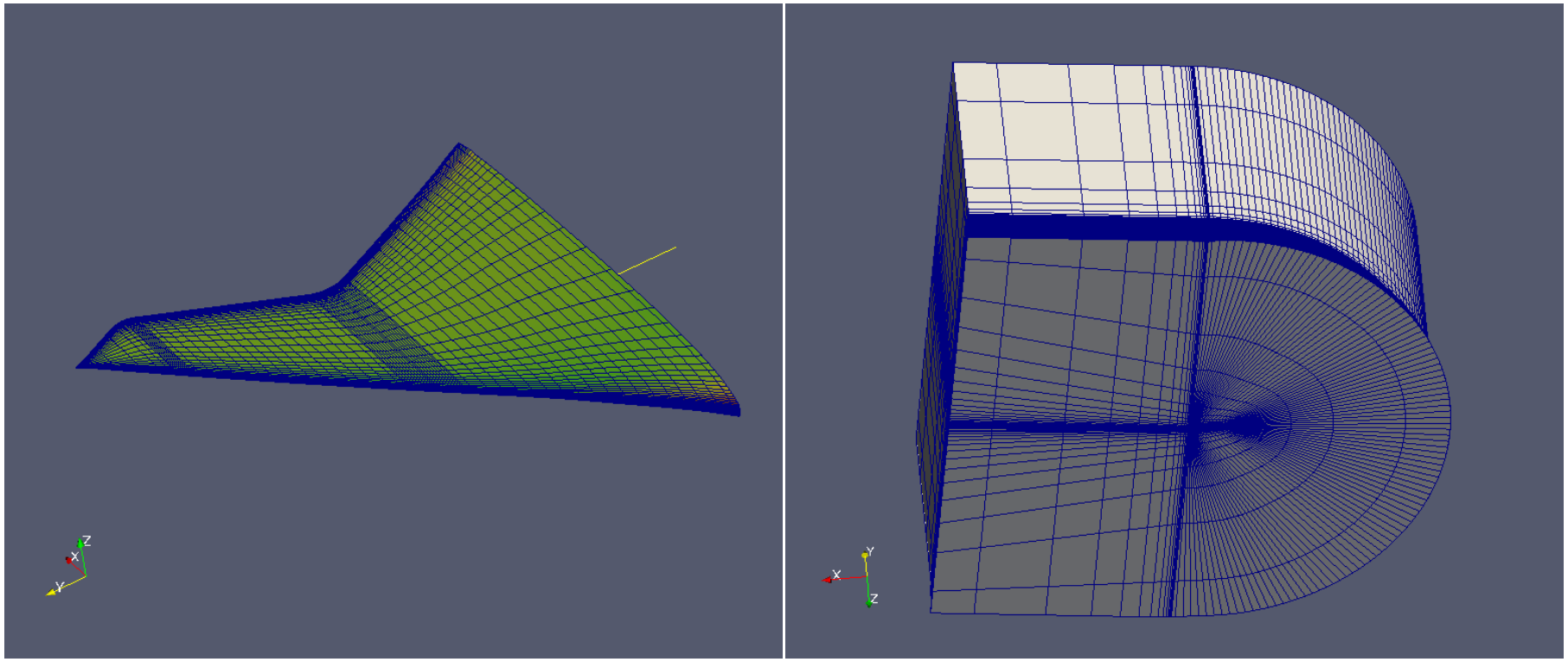
where

$$\begin{aligned} X = &\quad \text{grid coordinates}, \quad X \in \mathbb{R}^m \\ Q = &\quad \text{solution vector}, \quad Q \in \mathbb{R}^n \\ & R : \mathbb{R}^m \times \mathbb{R}^n \to \mathbb{R}^n \end{aligned}$$

- Q depends implicitly on X through $R(X, Q) = 0$
- Discrete approximation to objective function

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

Elements of practical shape optimization



Shape parameters β → Surface grid X_s → Volume grid X → CFD solution Q → $\mathfrak{I}$

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

Adjoint approach: $\frac{\mathrm{d}\mathfrak{I}}{\mathrm{d}X}$

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

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

- Differentiate state equation $R(X, Q) = 0$

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

- Flow sensitivity $\frac{\partial Q}{\partial X}$; costly to evaluate
- Introducing an *adjoint variable* $\Psi \in \mathbb{R}^n$, we can write

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

Adjoint approach

- Collect terms involving the flow sensitivity

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

- Choose Ψ so that flow sensitivity vanishes

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

- Gradient

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

Cost of Adjoint-based steepest-descent

$$\text{Cost} = O(1)N_{iter} = O(N)$$

Optimization steps

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

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

- Solve adjoint equations to steady-state

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

- Compute gradient wrt grid X

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

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

Automatic Differentiation

- Discrete adjoint method requires

$$\frac{\partial J}{\partial X}, \quad \frac{\partial J}{\partial Q}, \quad \left(\frac{\partial R}{\partial X}\right)^t \Psi, \quad \left(\frac{\partial R}{\partial Q}\right)^t \Psi$$

- Differentiate J and R by hand and write new code to compute above derivatives
- J and R are already available as a computer code
  - elementary operations: $*, /, +, -$
  - elementary functions: sin, exp, log, etc.
  - chain rule of differentiation
  - use Automatic Differentiation
- AD tool: automates the application of chain rule of differentiation

Automatic Differentiation: forward and backward mode

$$X \in \mathbb{R}^m, \quad Y \in \mathbb{R}^n, \quad Y = F(X)$$

- Forward mode

$$X \in \mathbb{R}^m, \quad \dot{X} \in \mathbb{R}^m \quad \rightarrow \quad \left(\frac{\partial F}{\partial X}\right) \dot{X} \in \mathbb{R}^n$$

- Backward mode

$$X \in \mathbb{R}^m, \quad \bar{Y} \in \mathbb{R}^n \quad \rightarrow \quad \left(\frac{\partial F}{\partial X}\right)^{\mathrm{t}} \bar{Y} \in \mathbb{R}^m$$

Automatic Differentiation: TAPENADE

Code `residue.f` to compute $X, Q \rightarrow R(X, Q) = \texttt{RES}$

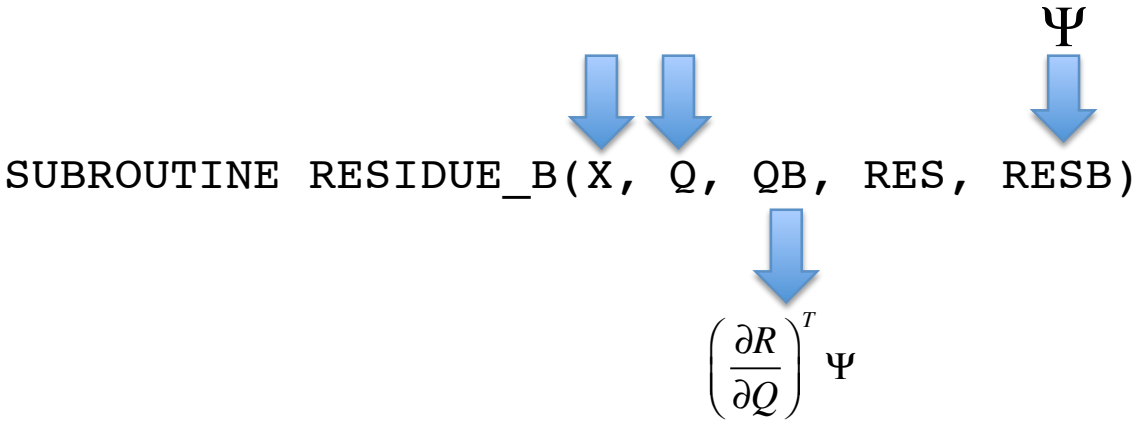
```
REAL X(M), Q(N), RES(N)
CALL RESIDUE(X, Q, RES)
```

Differentiate `RES` wrt `Q`

```
tapenade -backward -vars Q -outvars RES residue.f
```

Generates new code `residue_b.f` : $X, Q, \Psi \rightarrow \left(\frac{\partial R}{\partial Q}\right)^t \Psi$

```
REAL X(M), Q(N), RES(N)
REAL QB(N), RESB(N)
CALL RESIDUE_B(X, Q, QB, RES, RESB)
```



Reverse mode

Consider a subroutine which takes in a vector $x(n)$ and returns a scalar function $f = f(x)$. A computer program to compute f contains a sequence of computations. Each line takes some set of previously computed values and returns a new value.

$$\begin{aligned} t_1 &= L_1(T_0) \\ t_2 &= L_2(T_1) \\ \vdots &= \vdots \\ t_p &= L_p(T_{p-1}) \\ f &= L_{p+1}(T_p) \end{aligned}$$

$$x \in \mathbb{R}^n, \quad T_0 = x$$

$$t_r \in \mathbb{R}$$

$$T_r = \begin{bmatrix} T_{r-1} \\ t_r \end{bmatrix}$$

The function f as coded, in general, depends on all of these variables $x, t_1, t_2, \ldots, t_p$

AD tools

- Source transformation and operator overloading
- See `http://www.autodiff.org`

Tool	Modes	Method	Lang
ADIFOR	F	ST	Fortran
ADOLC	F/B	OO	C
TAPENADE	F/B	ST	Fortran/C
TAF/TAMC	F/B	ST	Fortran/C
MAD	F	OO	Matlab

Important criteria for MDO applications of complex, 3D models

Consistent: Is the parameterization consistent across multiple disciplines?

Airplane shape design variables: Are the design variables directly related to the airplane shape design variables such as camber, thickness, twist, shear, and planform?

Compact: Does the parameterization provide a compact set of design variables?
10s vs 1000s

Smooth: Does the shape perturbation maintain a smooth geometry?

Local control: Is there any local control on shape changes?

Analytical sensitivity: Is it feasible to calculate the sensitivity analytically?

Grid deformation: Does the parameterization allow the grid to be deformed?

Setup time: Can a shape optimization application be set up quickly?
hours, days, weeks, months?

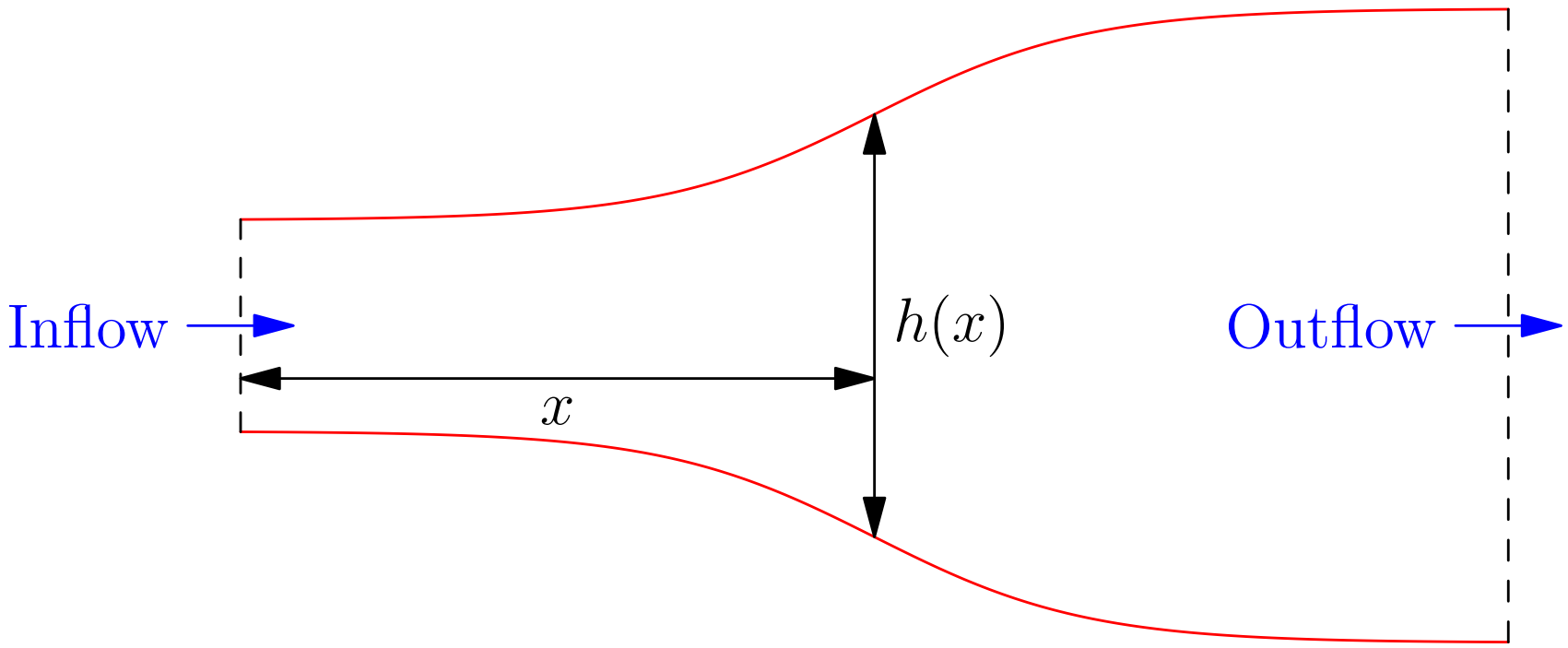
Existing grid: Does the parameterization allow the existing grid to be reused?
Does it require to reverse engineer the baseline design parameters?

CAD: Is there a direct connection to the CAD system?

Samareh: Survey of shape parameterization techniques

Examples of shape optimization

Quasi 1-D flow



Quasi 1-D flow

- Quasi 1-D flow in a duct

$$\frac{\partial}{\partial t}(hU) + \frac{\partial}{\partial x}(hf) = \frac{\mathrm{d}h}{\mathrm{d}x}P, \quad x \in (a,b) \quad t > 0$$

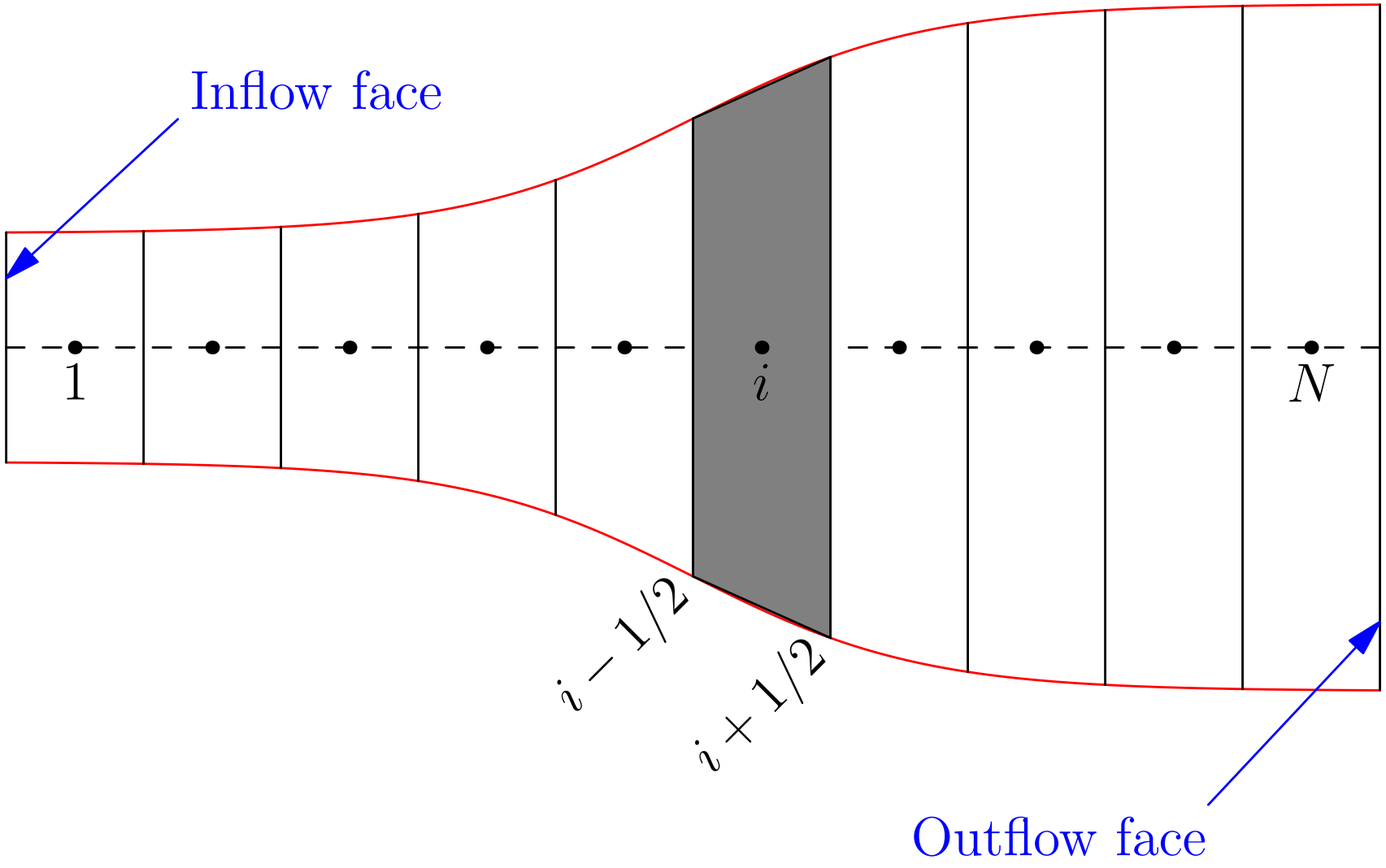
$$U = \begin{bmatrix} \rho \\ \rho u \\ E \end{bmatrix}, \quad f = \begin{bmatrix} \rho u \\ p + \rho u^2 \\ (E+p)u \end{bmatrix}, \quad P = \begin{bmatrix} 0 \\ p \\ 0 \end{bmatrix}$$

$$h(x) = \text{cross-section height of duct}$$

- Inverse design: find shape h to get pressure distribution p^*
- Optimization problem: find the shape h which minimizes

$$J = \int_a^b (p - p^*)^2 \mathrm{d}x$$

Quasi 1-D flow



Quasi 1-D flow

- Finite volume scheme

$$h_i \frac{\mathrm{d}U_i}{\mathrm{d}t} + \frac{h_{i+1/2}F_{i+1/2} - h_{i-1/2}F_{i-1/2}}{\Delta x} = \frac{(h_{i+1/2} - h_{i-1/2})}{\Delta x} P_i$$

- Discrete cost function

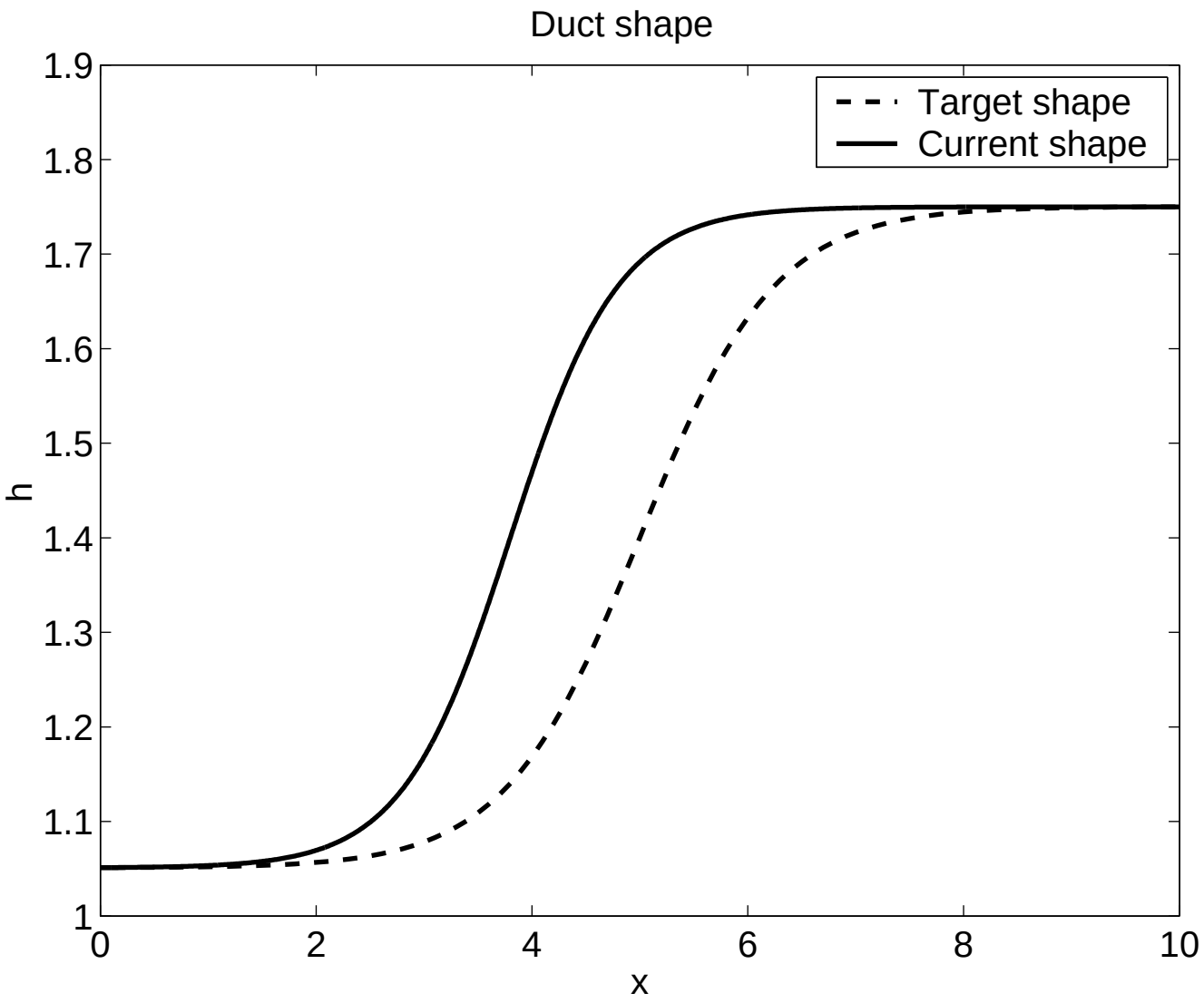
$$J = \sum_{i=1}^{N} (p_i - p_i^*)^2$$

- Control variables

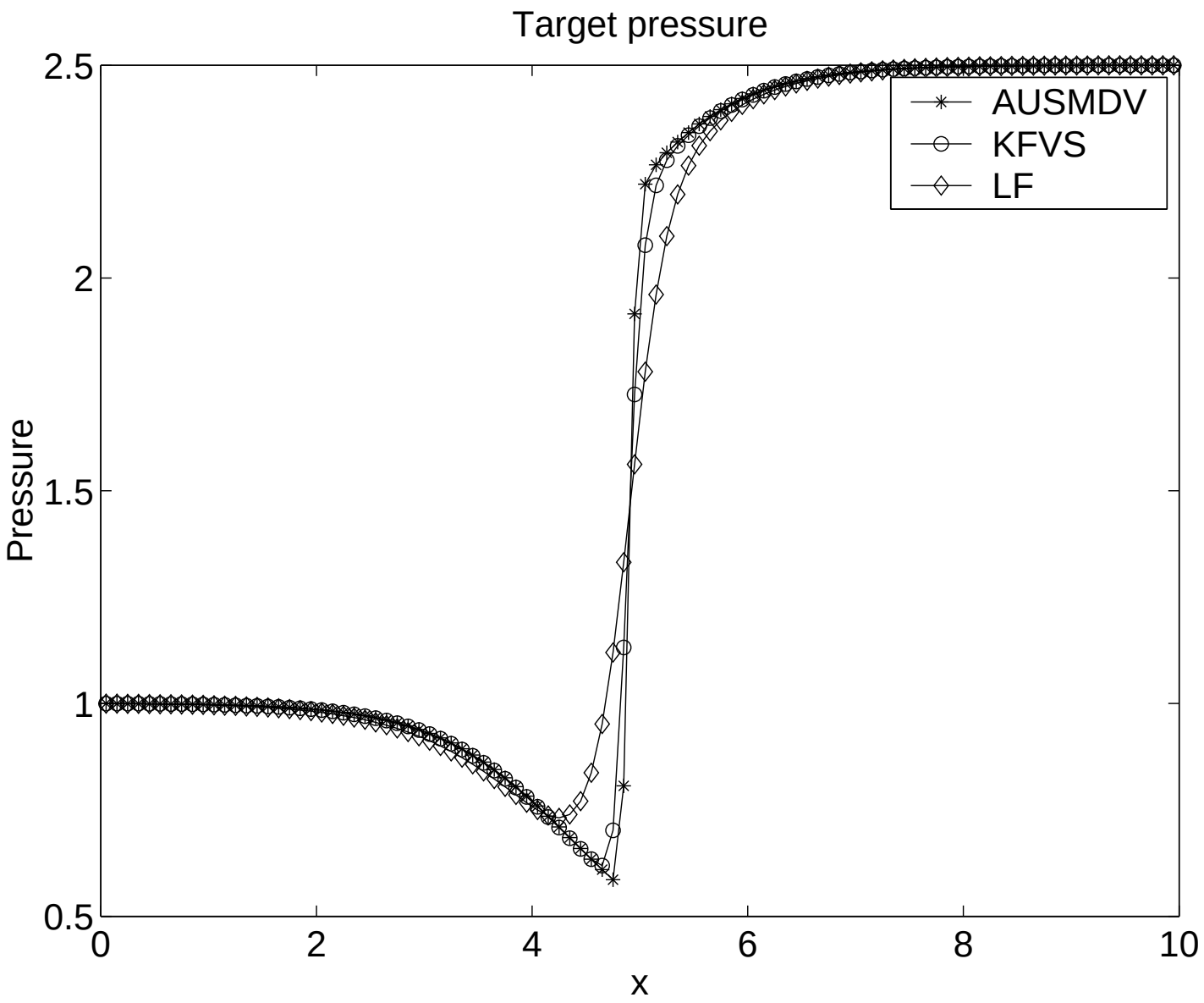
$$h_{1/2}, h_{1+1/2}, \ldots, h_{i+1/2}, \ldots, h_{N+1/2}$$

- $N = 100$

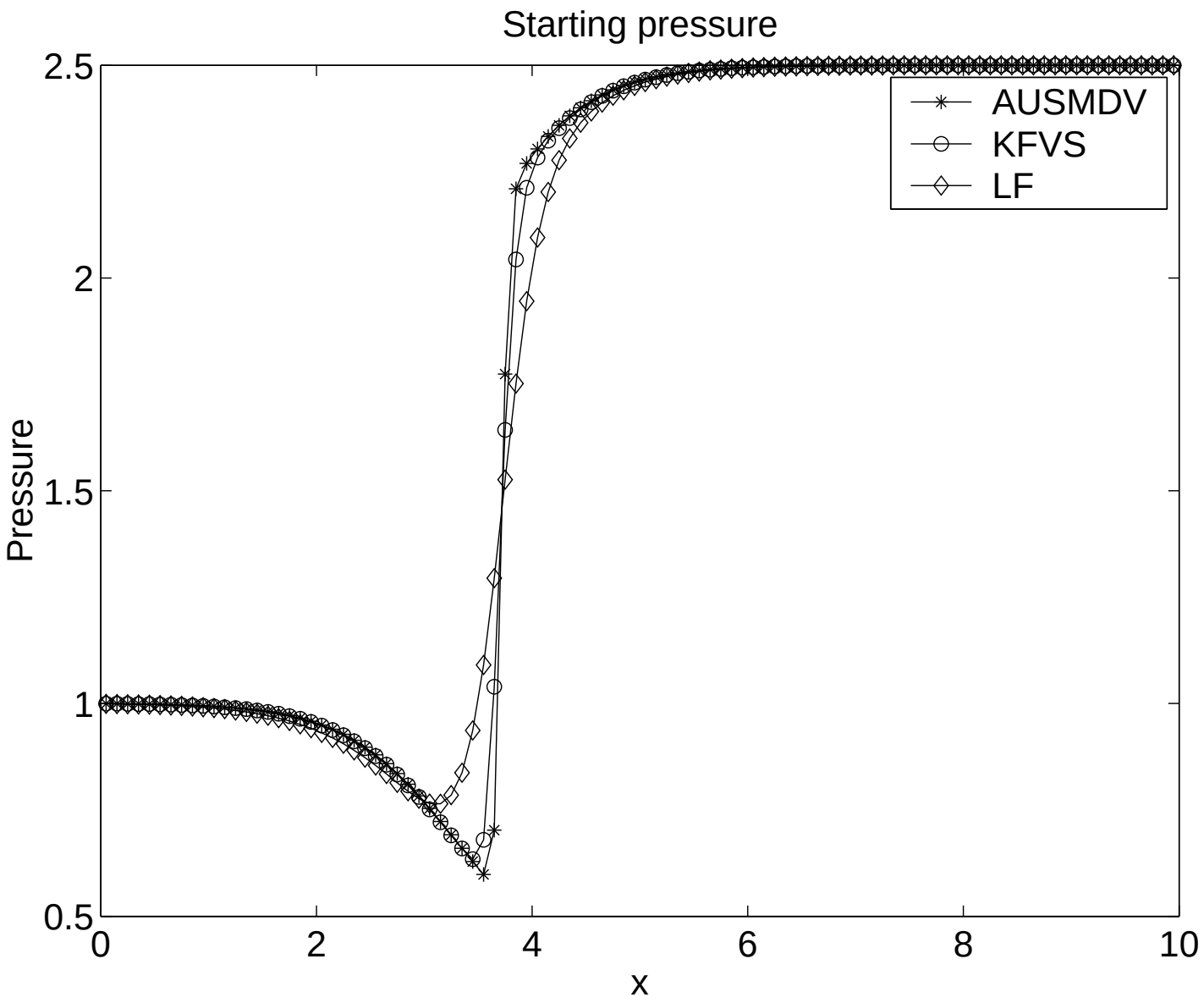
Duct shape


Duct shape
Target shape
Current shape
h
x
1.9
1.8
1.7
1.6
1.5
1.4
1.3
1.2
1.1
1
0
2
4
6
8
10

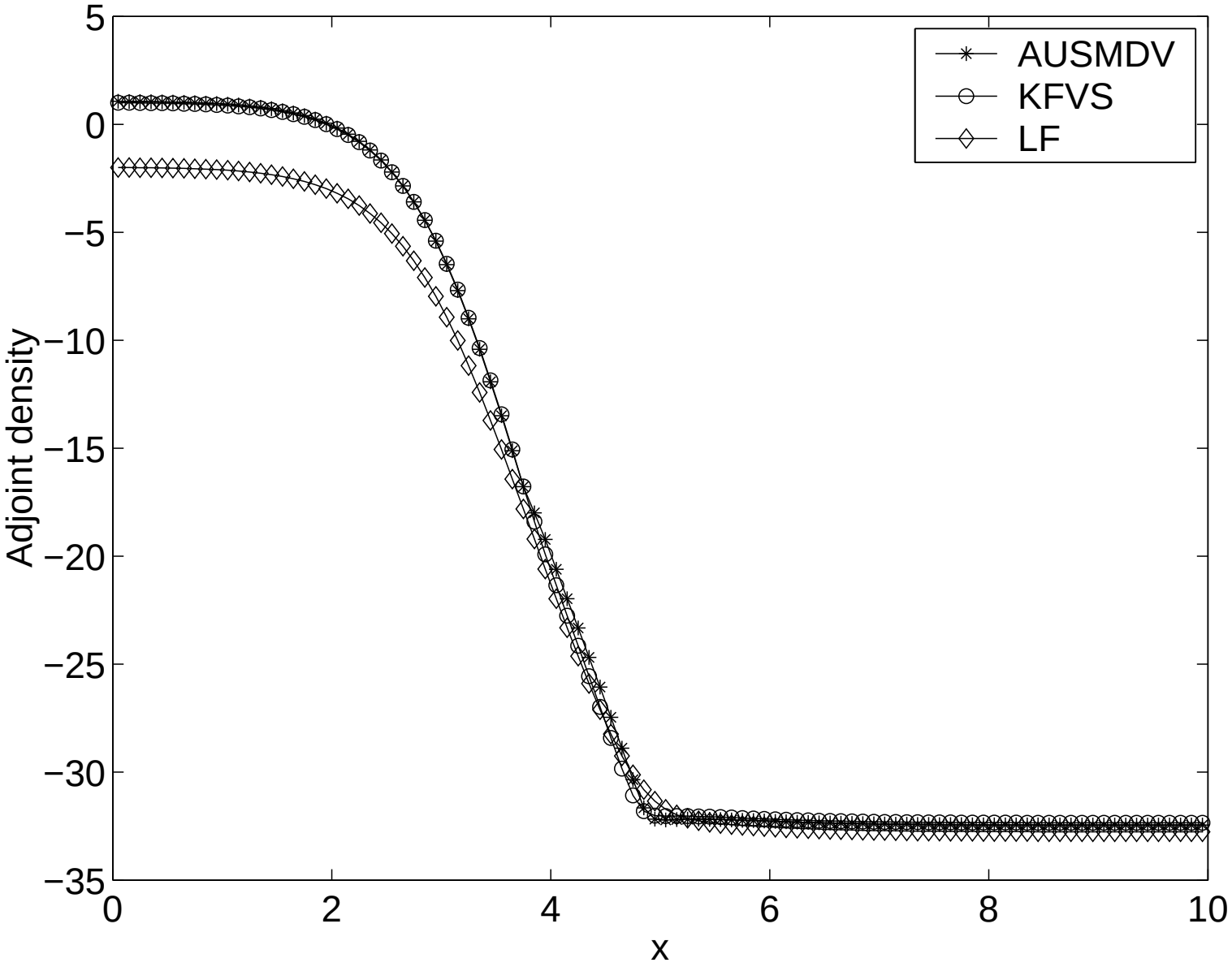
Target pressure distribution p^*



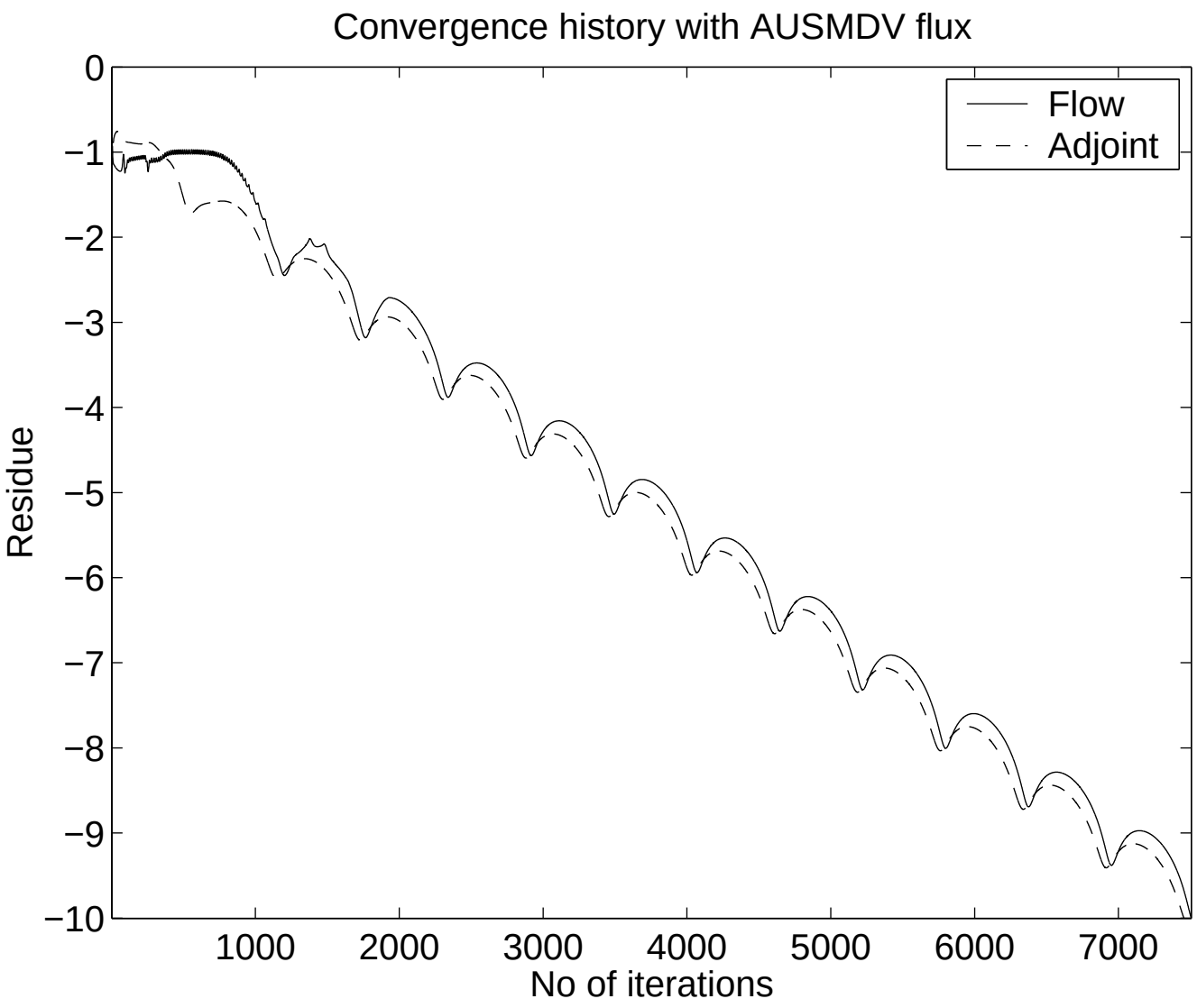
Current pressure distribution



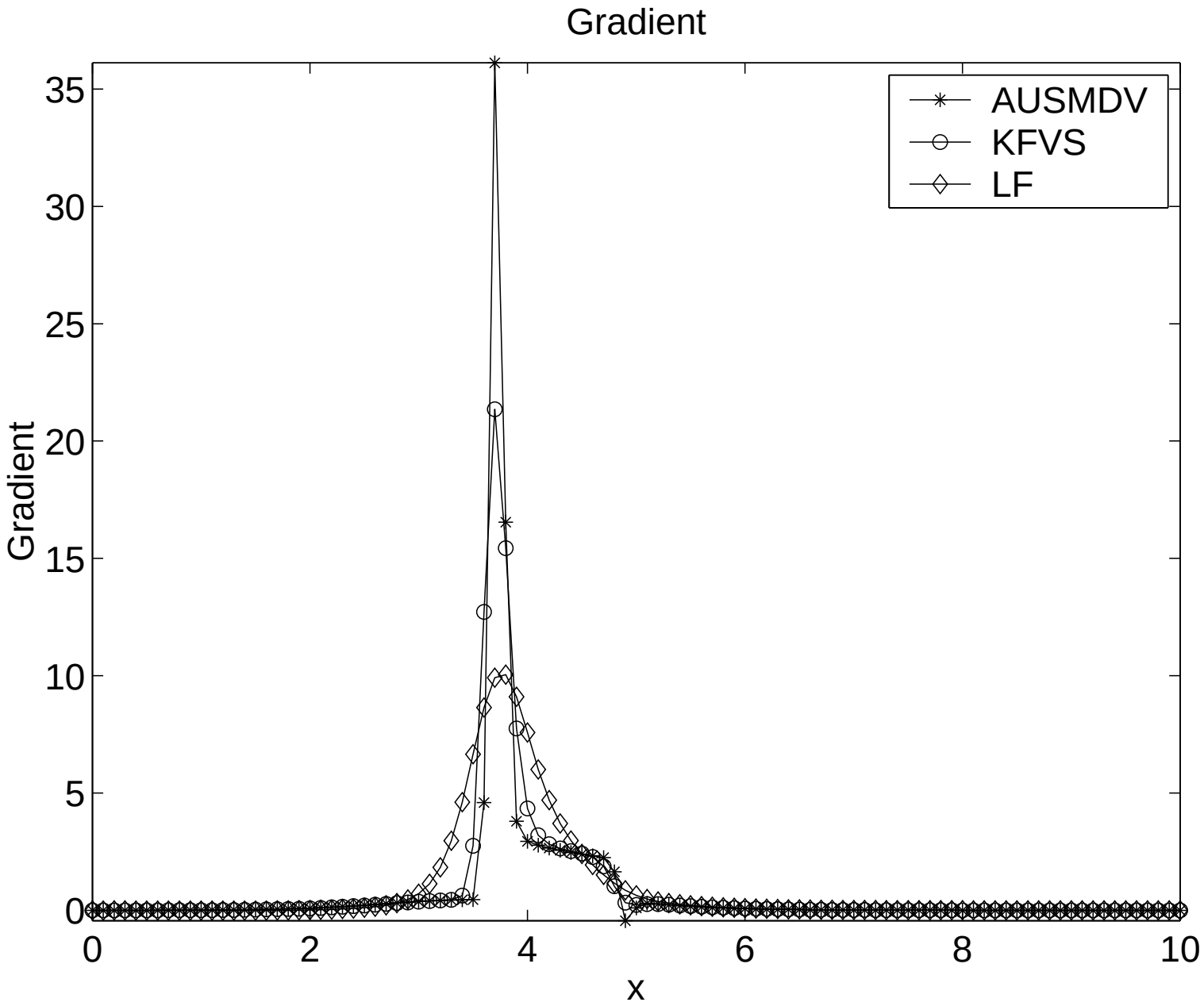
Adjoint density



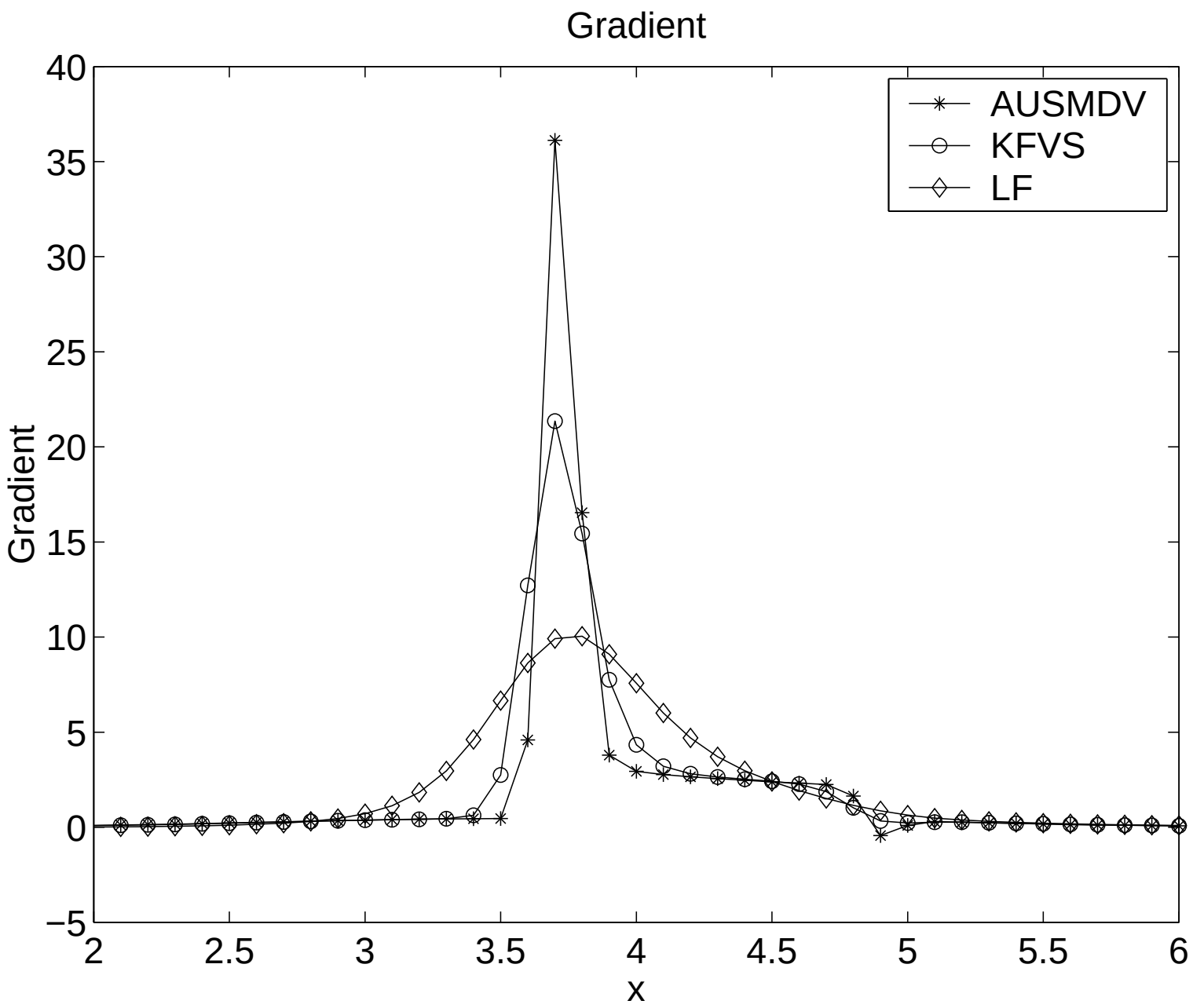
Convergence history: Explicit Euler



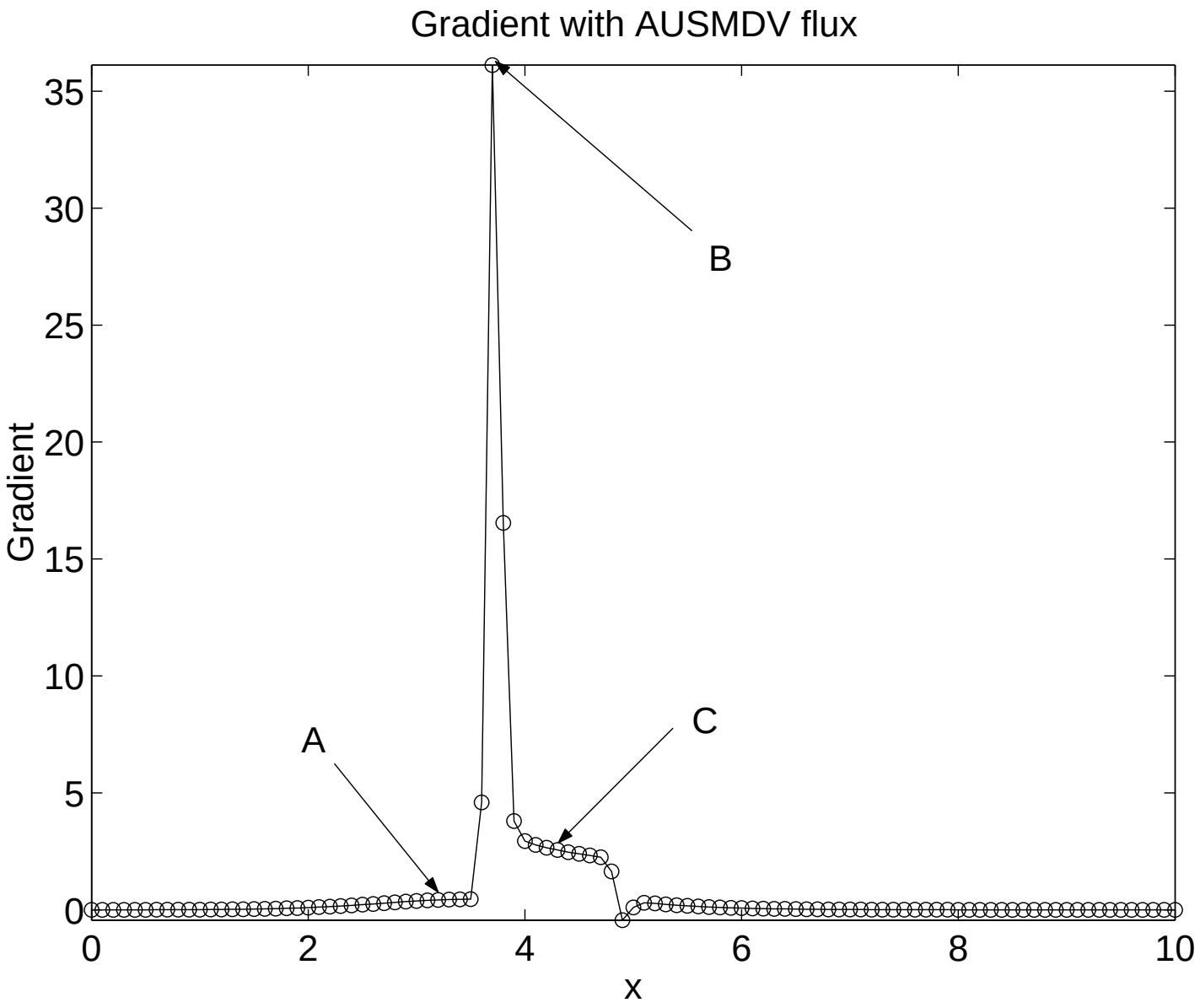
Shape gradient



Shape gradient



Validation of Shape gradient



Validation of shape gradient

$$\frac{\partial J}{\partial h} \approx \frac{J(h+\Delta h) - J(h-\Delta h)}{2\Delta h}$$

Δh	A	B	C
0.01	0.4191069499	35.18452823	2.545316345
0.001	0.4231223499	36.10982621	2.556461900
0.0001	0.4231624999	36.11933154	2.556573499
0.00001	0.4231599998	36.11942125	2.556575000
0.000001	0.4231999994	36.11942305	2.556550001
0.0000001	0.4229999817	36.11942329	2.556499971
AD	0.4231628330	36.11941951	2.556574450

Gradient smoothing

- Non-smooth gradients G especially in the presence of shocks
- Smooth using an elliptic equation

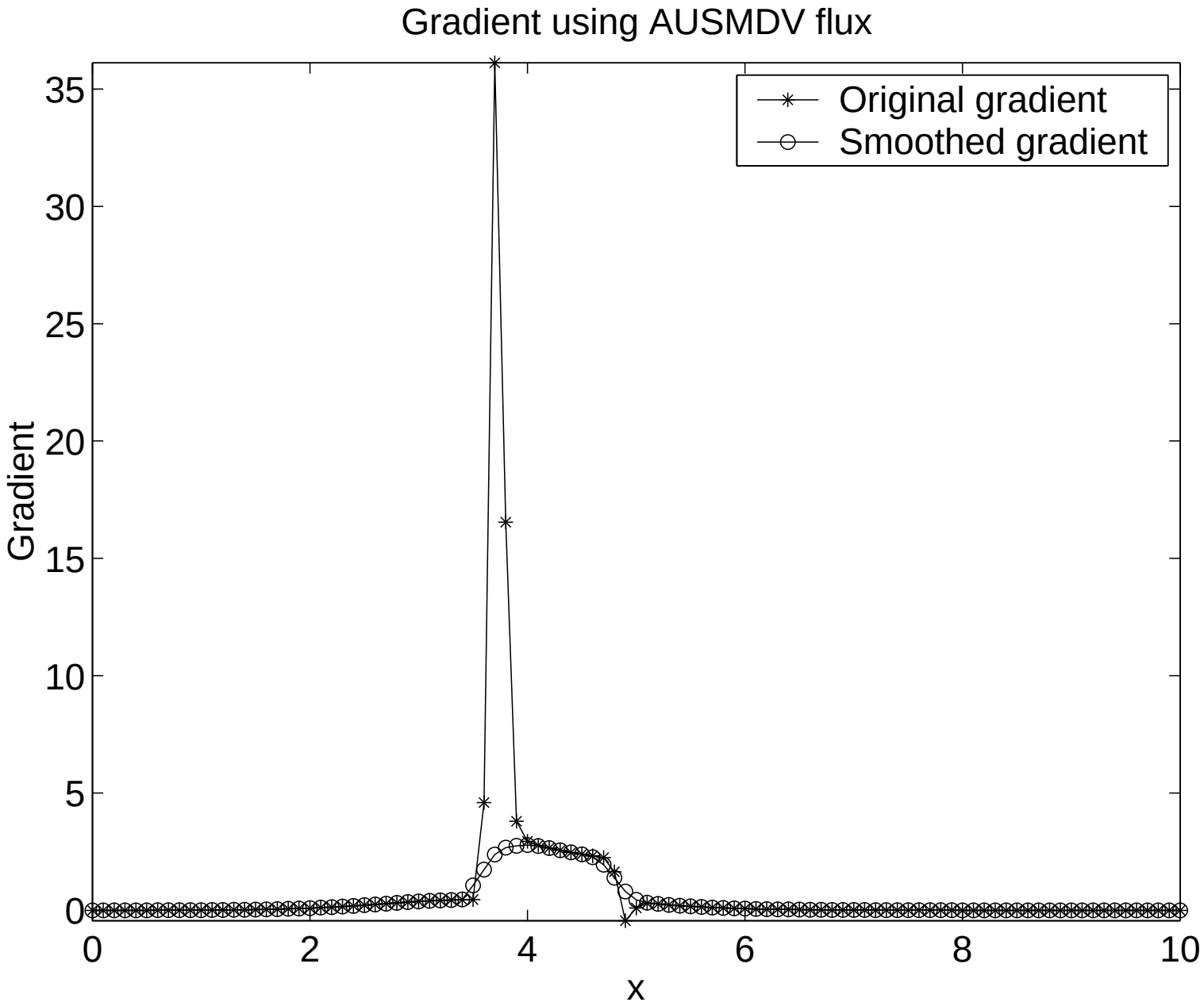
$$\left(1 - \epsilon(x)\frac{\mathrm{d}^2}{\mathrm{d}x^2}\right)\bar{G}(x) = G(x)$$

$$\epsilon_i = \{|G_{i+1} - G_i| + |G_i - G_{i-1}|\}L_i$$

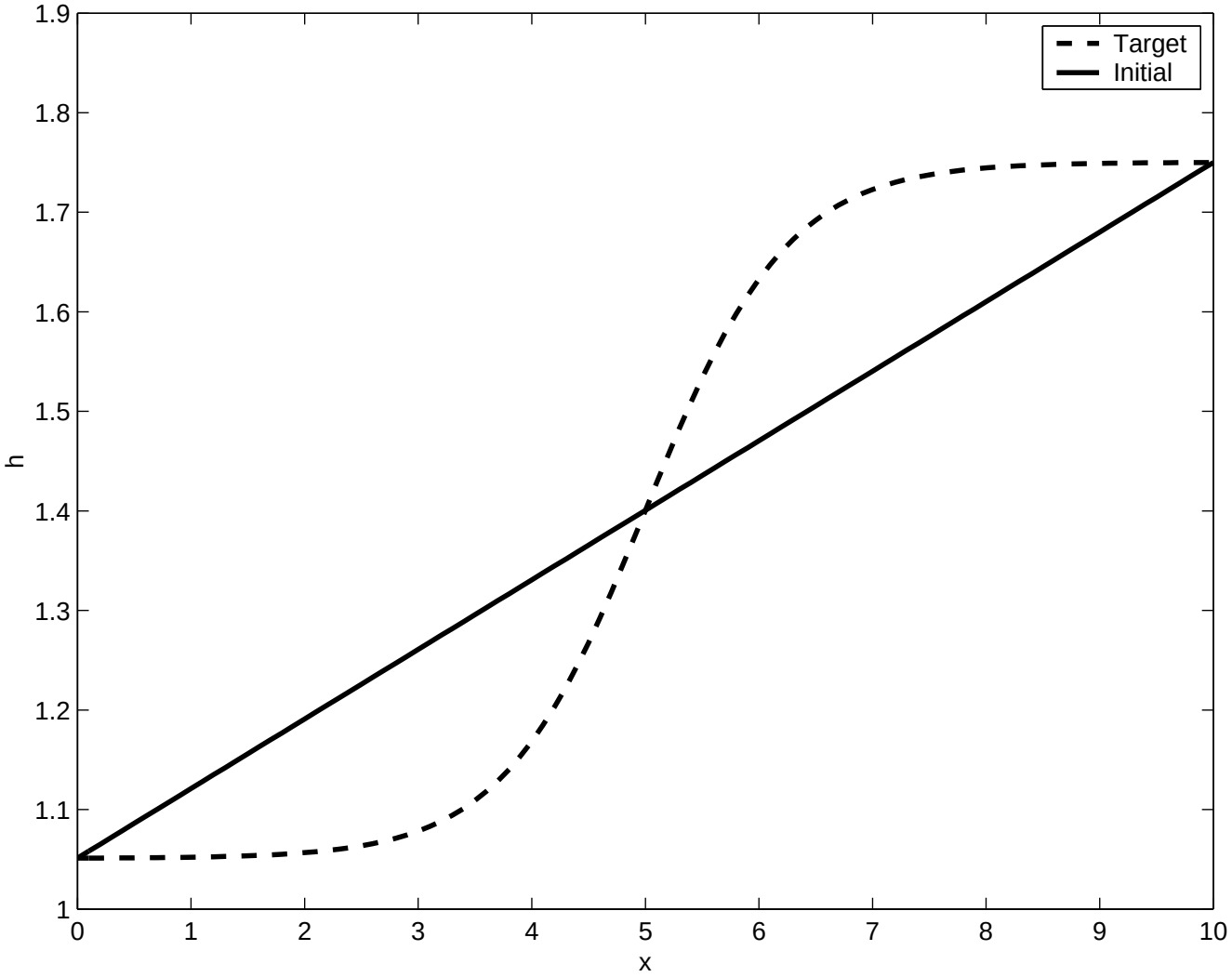
$$L_i = \frac{|G_{i+1} - 2G_i + G_{i-1}|}{\max(|G_{i+1} - G_i| + |G_i - G_{i-1}|, \mathrm{TOL})}$$

- Finite difference with Jacobi iterations

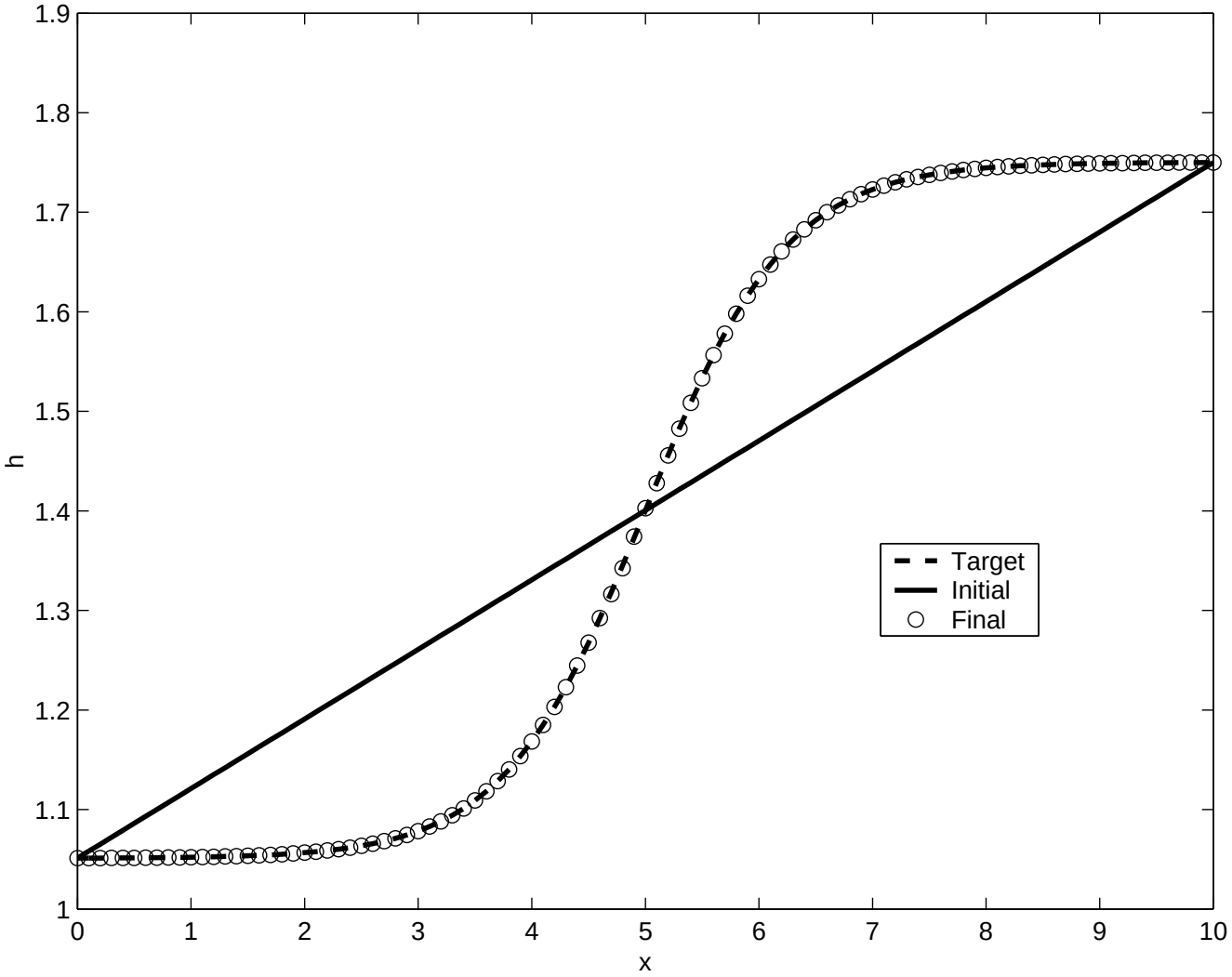
Gradient smoothing



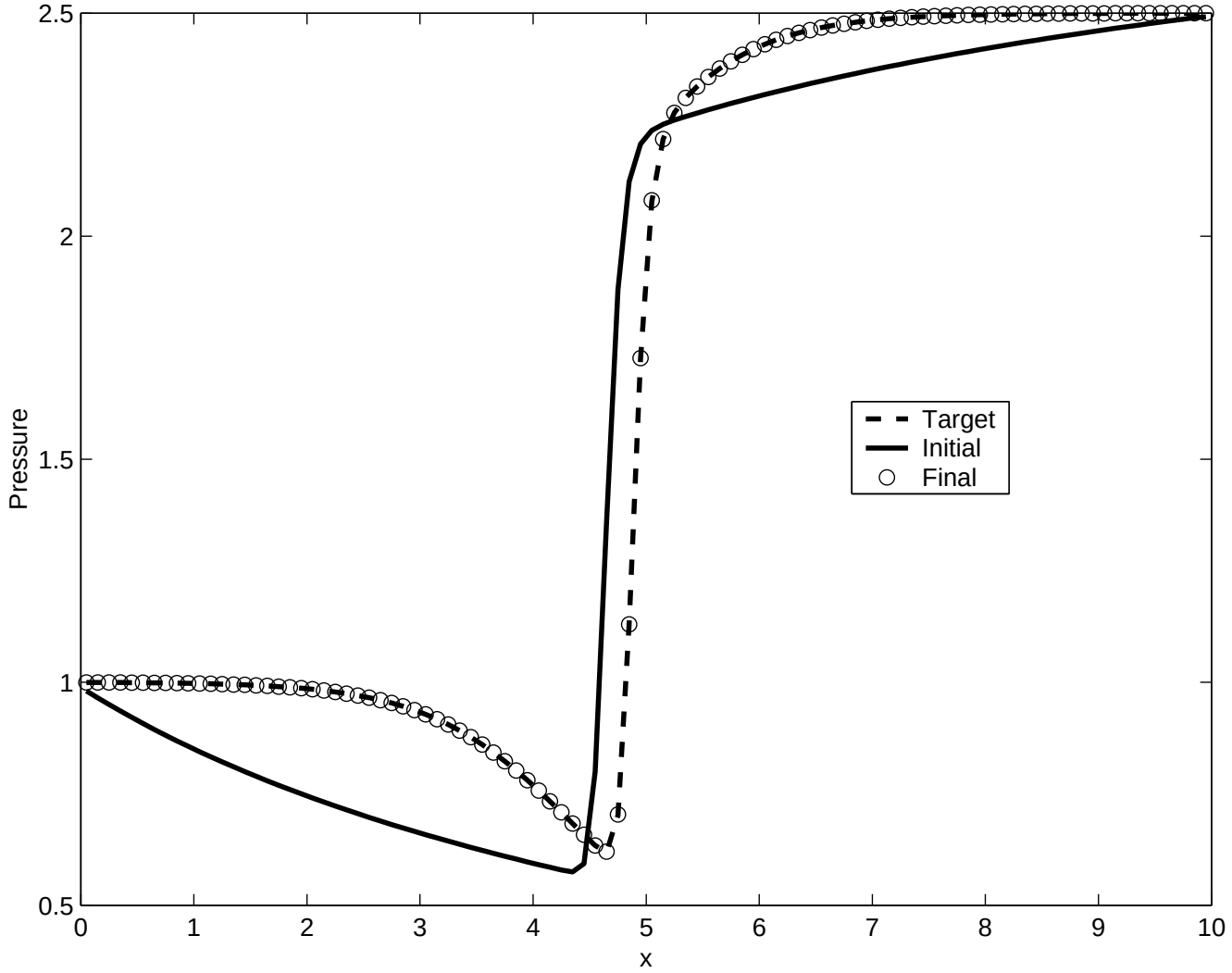
Quasi 1-D optimization: Shape



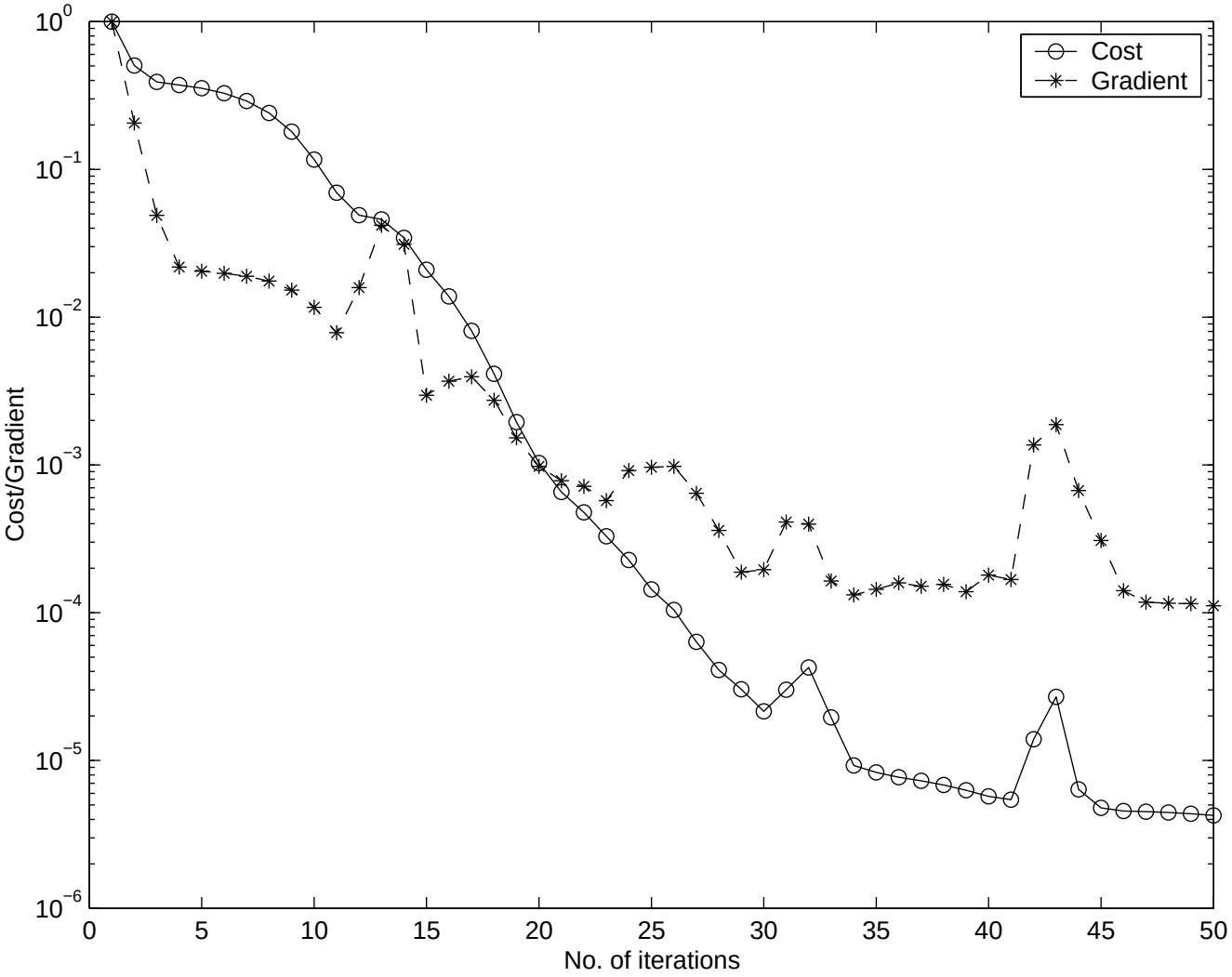
Quasi 1-D optimization: Final shape



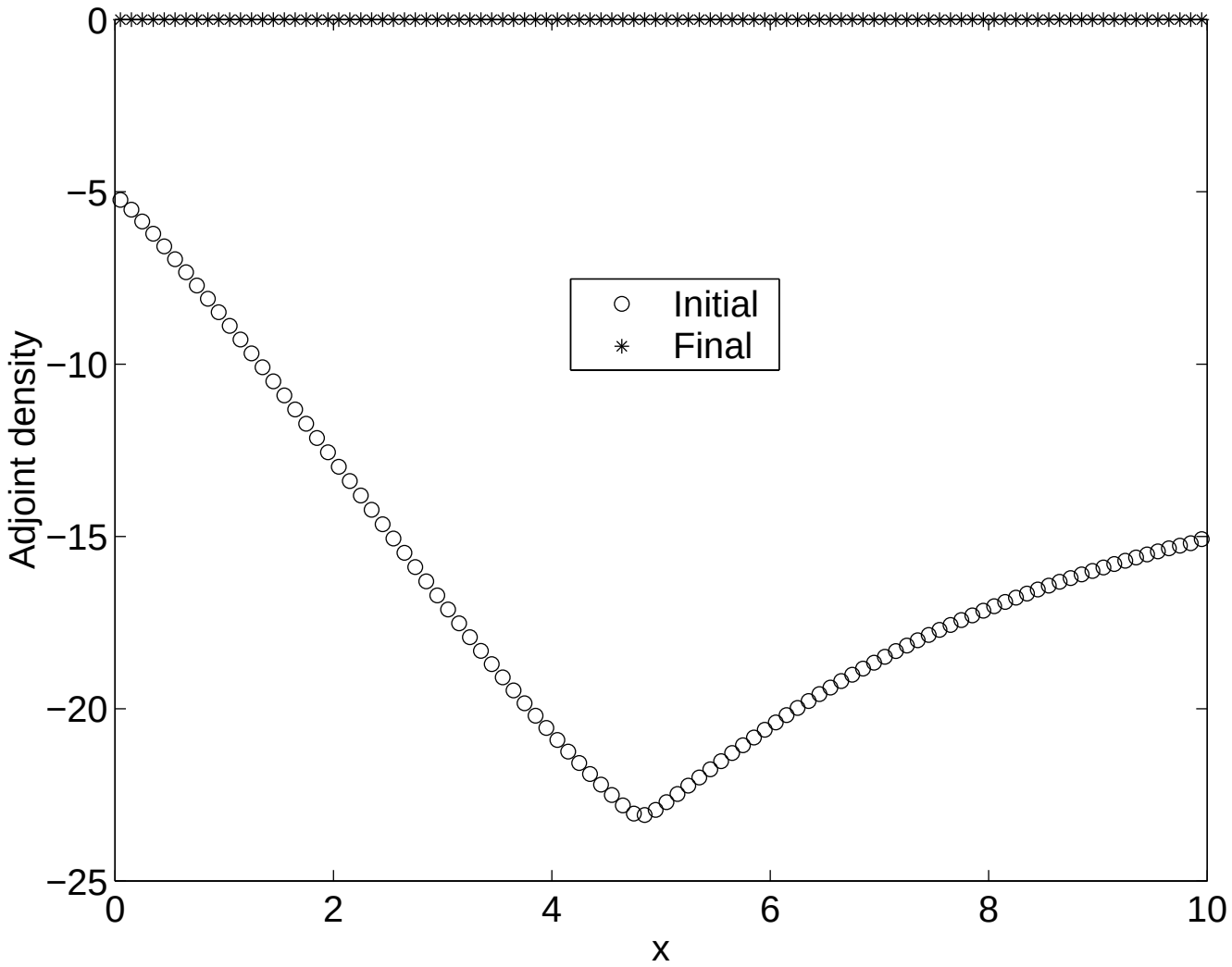
Quasi 1-D optimization: Pressure



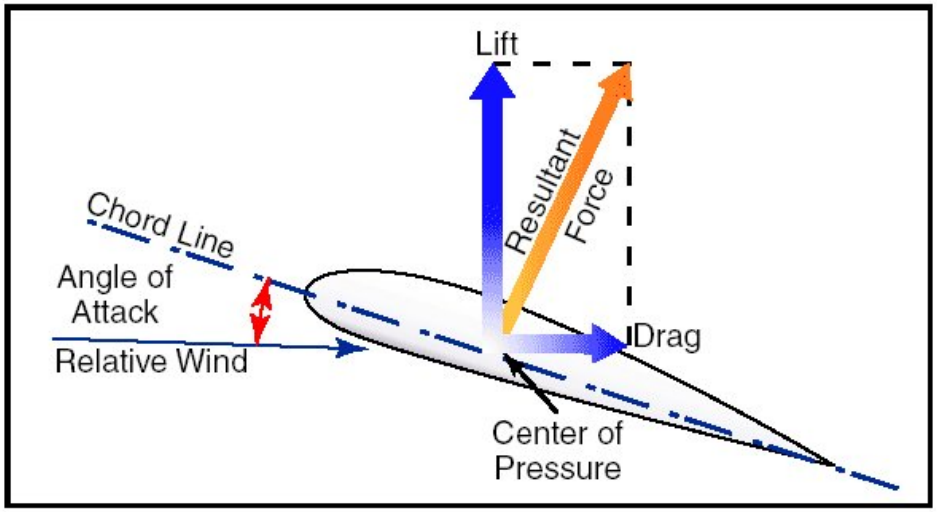
Quasi 1-D optimization: Convergence



Quasi 1-D optimization: Adjoint density



Airfoil shape optimization



Shape parameterization

- Parameterize the deformations

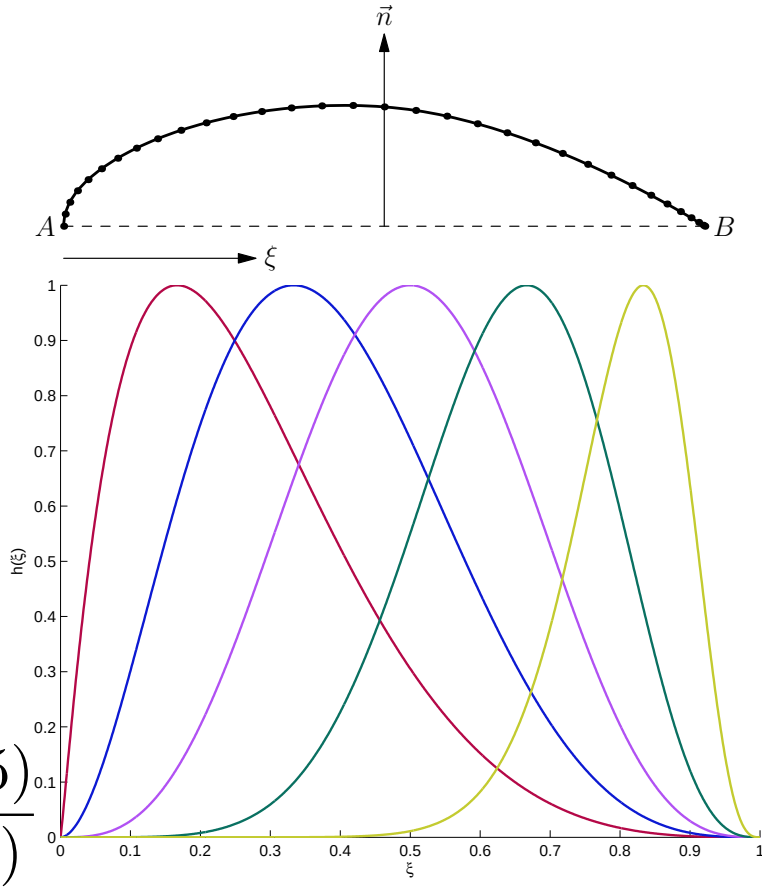
$$\begin{bmatrix} x_s \\ y_s \end{bmatrix} = \begin{bmatrix} x_s^{(0)} \\ y_s^{(0)} \end{bmatrix} + \begin{bmatrix} n_x \\ n_y \end{bmatrix} h(\xi)$$

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

- 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

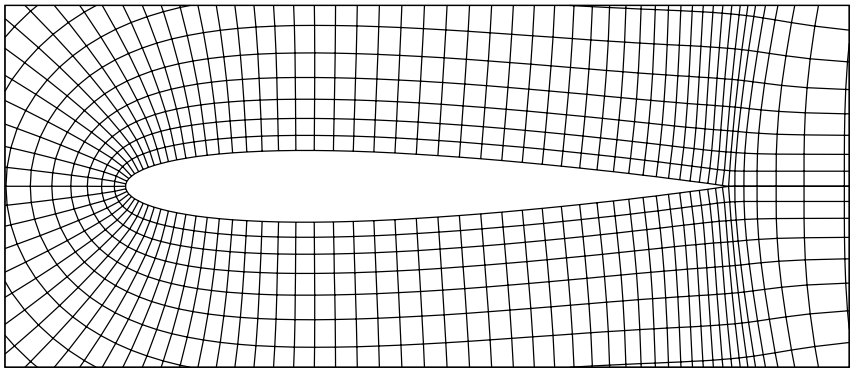
- Interpolate displacement of surface points to interior points using RBF

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

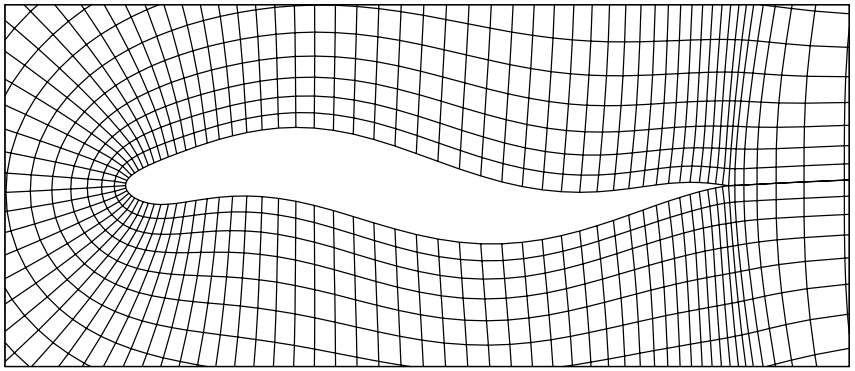
where $\vec{r} = (x, y)$

- 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`

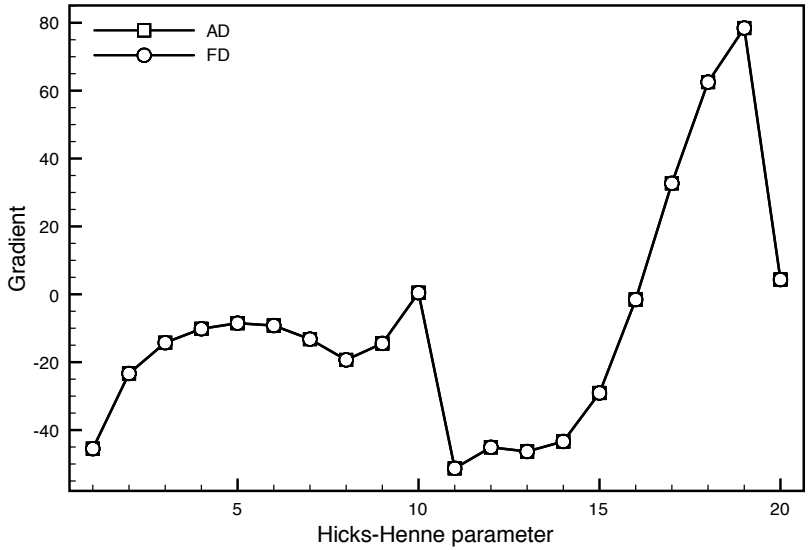
- Finite volume scheme
- Structured, multi-block grids
- Roe flux
- MUSCL reconstruction with Hemker-Koren limiter
- Implicit scheme

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

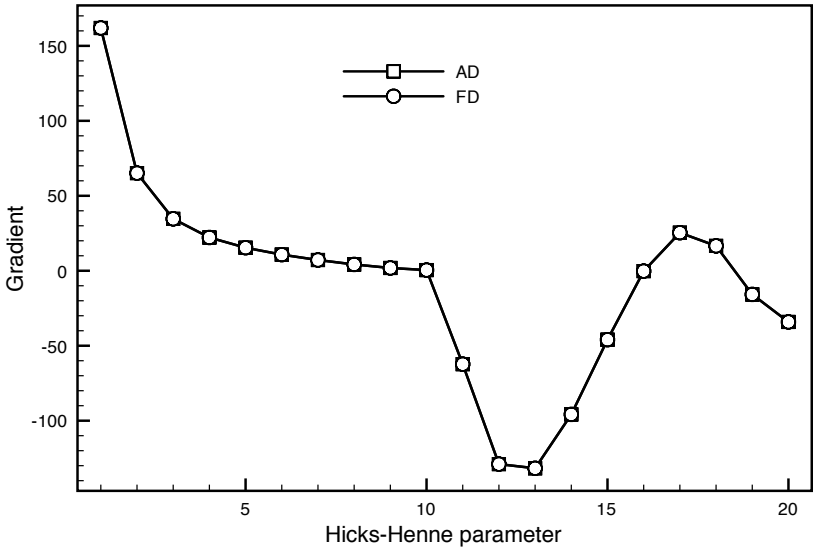
Validation of adjoint gradients

- Dot-product test

$$
\begin{aligned}
\dot{R} &:= R'(Q)\dot{Q} \\
&\approx \frac{R(Q+\epsilon\dot{Q}) - R(Q)}{\epsilon} \\
\bar{Q} &:= [R'(Q)]^{\top}\bar{R} \\
\dot{R}^{\top}\bar{R} &\stackrel{?}{=} \dot{Q}^{\top}\bar{Q}
\end{aligned}
$$



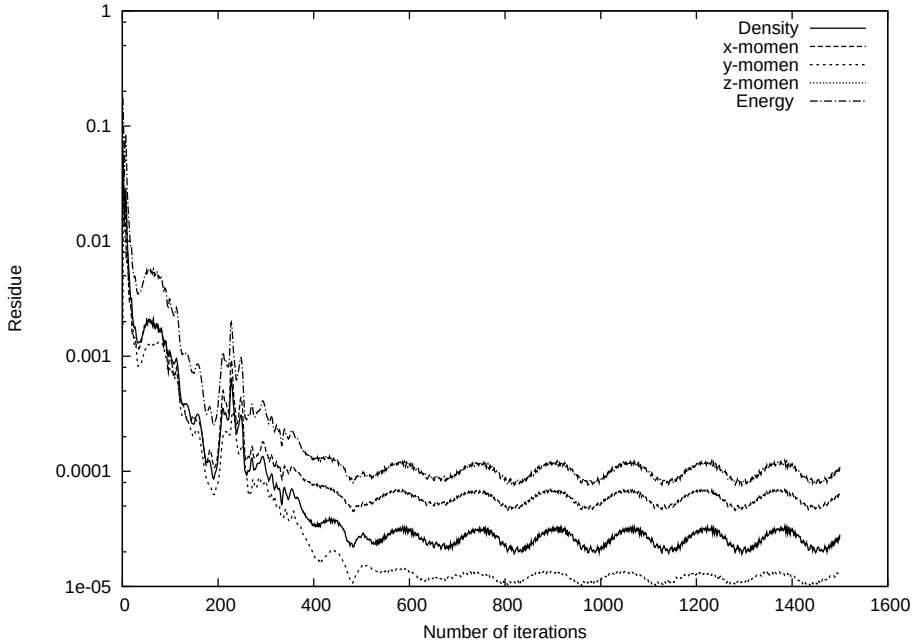
- Upwind schemes and limiters can cause problems due to non-differentiability
- Check adjoint derivatives against finite difference
  - NACA0012: C_d/C_l
  - RAE2822: C_d



Convergence and limiter: RAE2822 airfoil

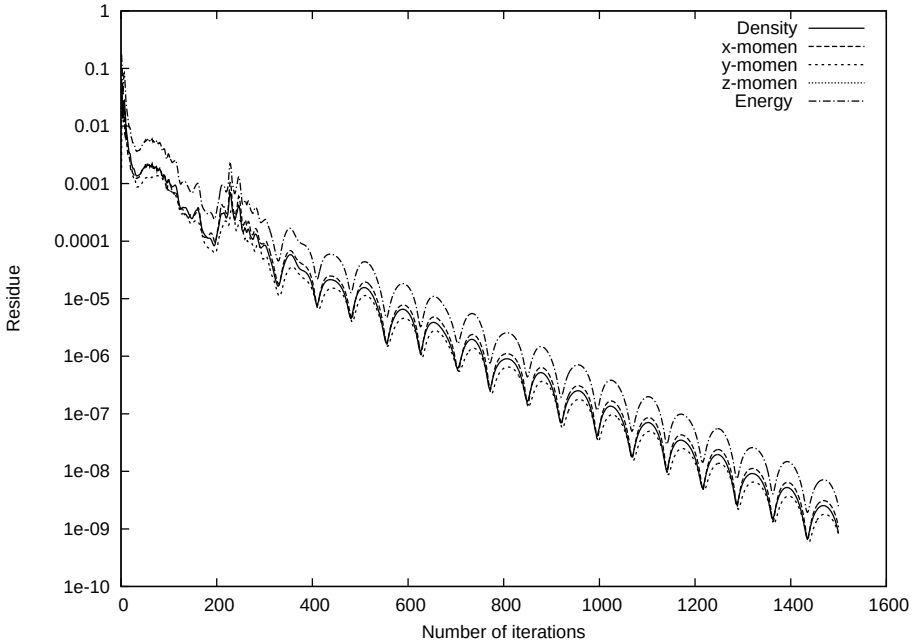
$$L(a,b) = \frac{a(b^2 + 2\epsilon) + b(2a^2 + \epsilon)}{2a^2 - ab + 2b^2 + 3\epsilon}, \quad 0 < \epsilon \ll 1$$

$\epsilon = 10^{-8}$



Adjoint iterations blow-up

$\epsilon = 10^{-4}$

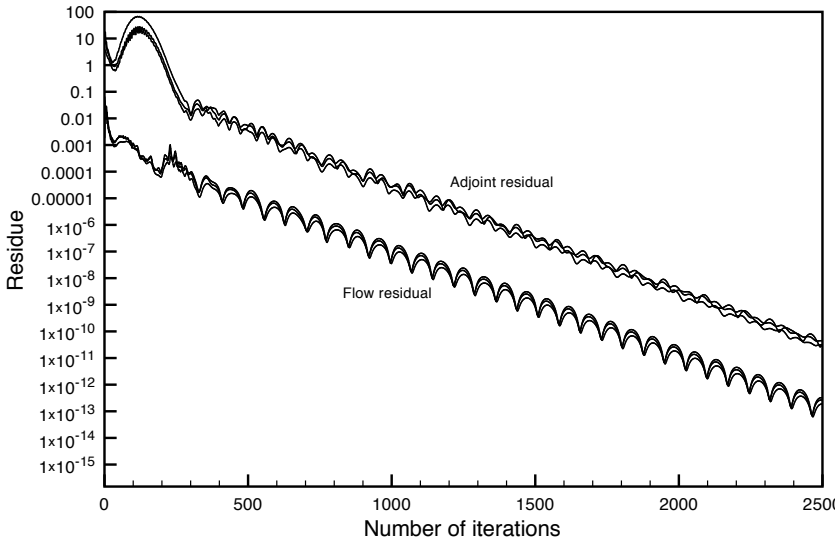


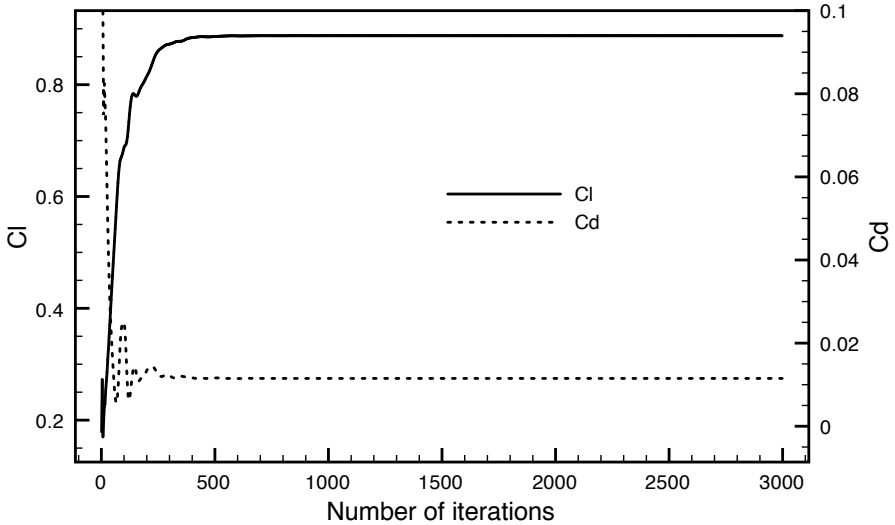
No blow-up

Iterative convergence tests

Primal residual
$R^n = ||R(Q^n)||$

Adjoint residual
$R^n = ||[R'(Q^\infty)]^\top \Psi^n + \mathfrak{I}'(Q^\infty)||$





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

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: $M_\infty = 0.729$, $\alpha = 2.31^o$

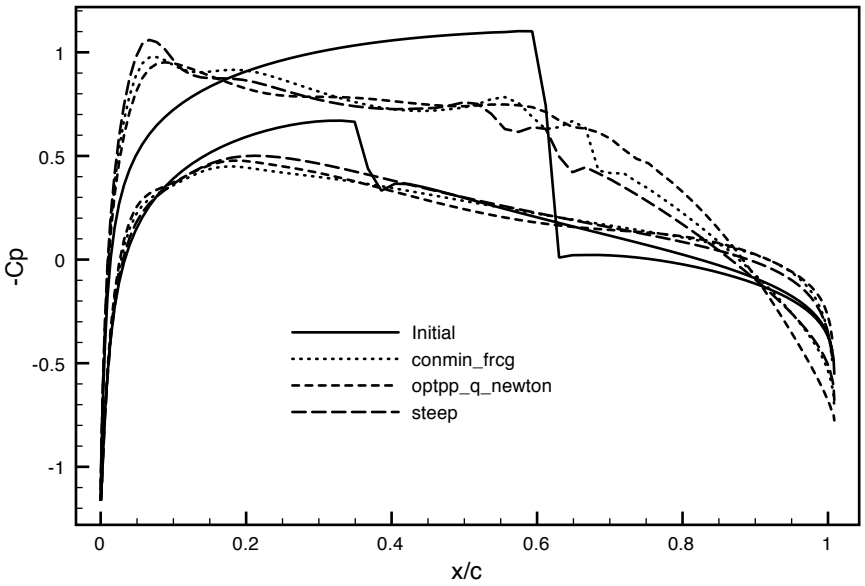
- Penalty approach

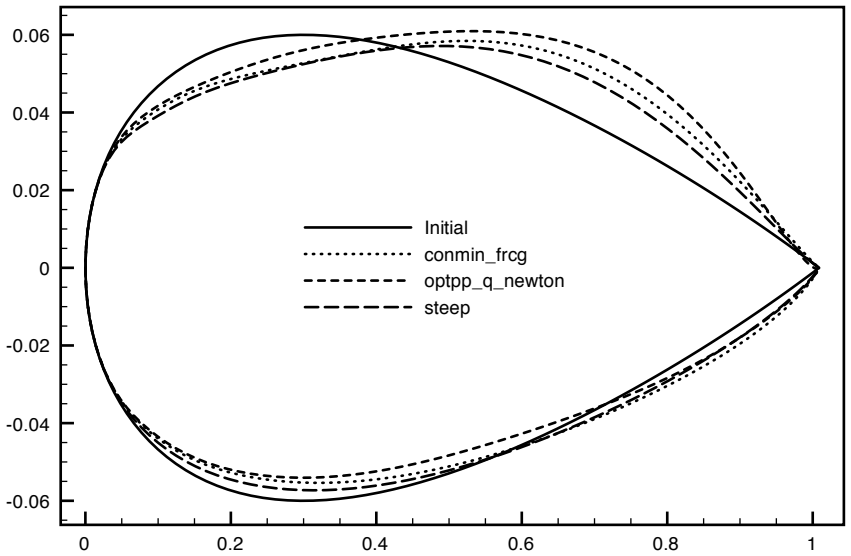
$$\mathfrak{I} = \frac{C_d}{C_{d_0}} + \omega \left| 1 - \frac{C_l}{C_{l_0}} \right|$$

- Constrained minimization

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

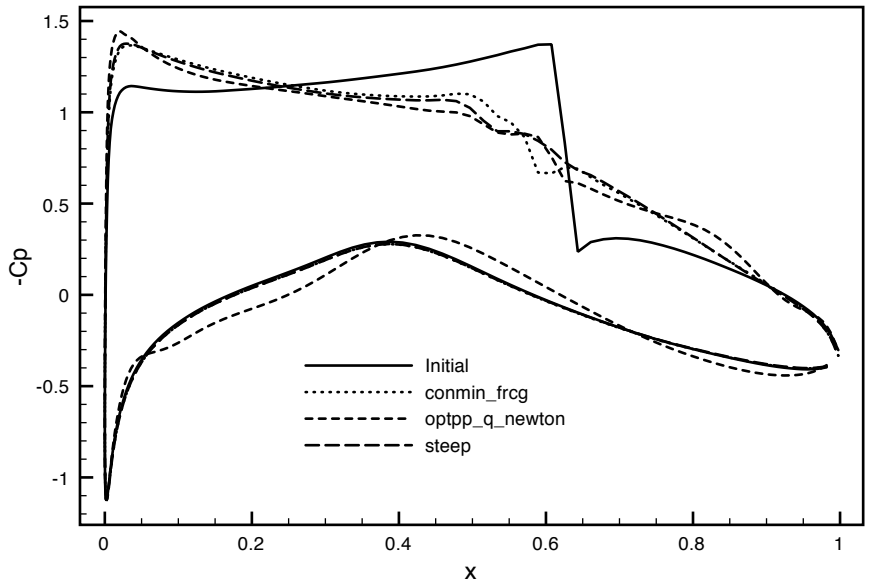
NACA0012: Maximize L/D

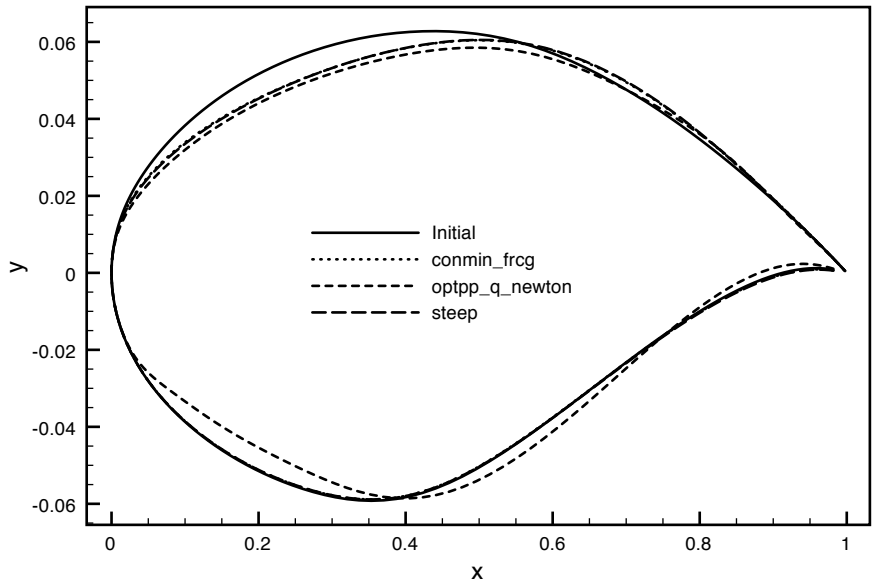




Method	$\mathfrak{J}$	$100C_d$	C_l	N_{fun}	N_{grad}
Initial	1.000	2.072	0.295	-	-
`conmin_frcg`	0.170	0.389	0.325	50	13
`optpp_q_newton`	0.142	0.329	0.329	50	39
`steep`	0.166	0.335	0.287	50	45

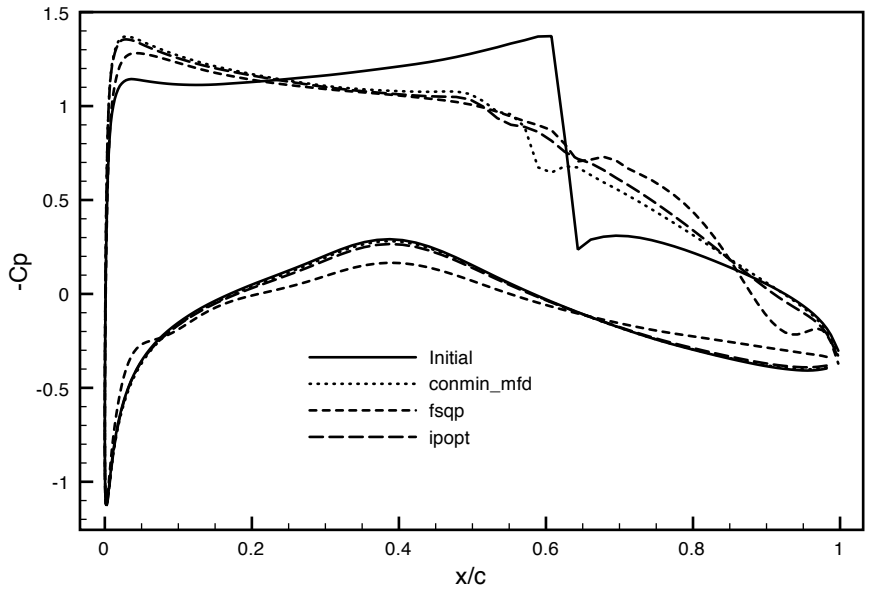
RAE2822: Drag minimization, penalty approach

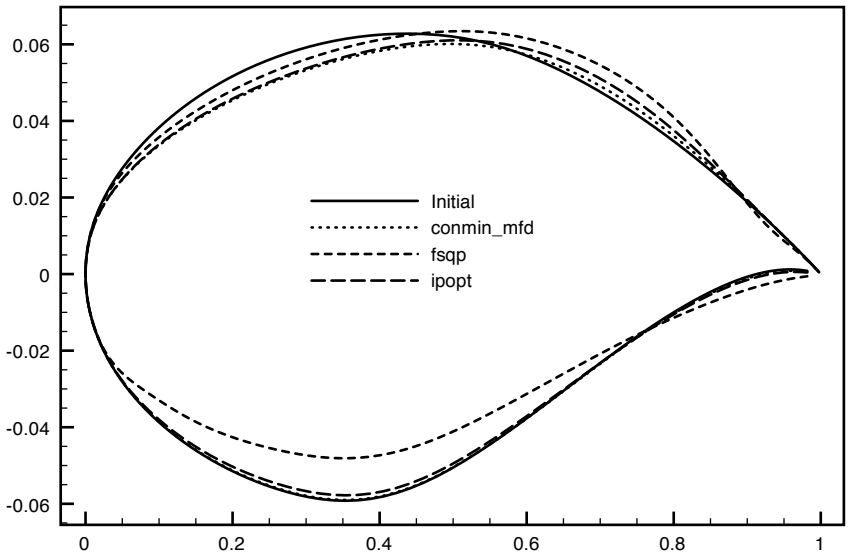




Method	$\mathfrak{J}$	$100C_d$	C_l	N_{fun}	N_{grad}
Initial	1.000	1.150	0.887	-	-
`conmin_frcg`	0.355	0.405	0.890	50	13
`optpp_q_newton`	0.351	0.400	0.884	50	51
`steep`	0.341	0.388	0.884	50	47

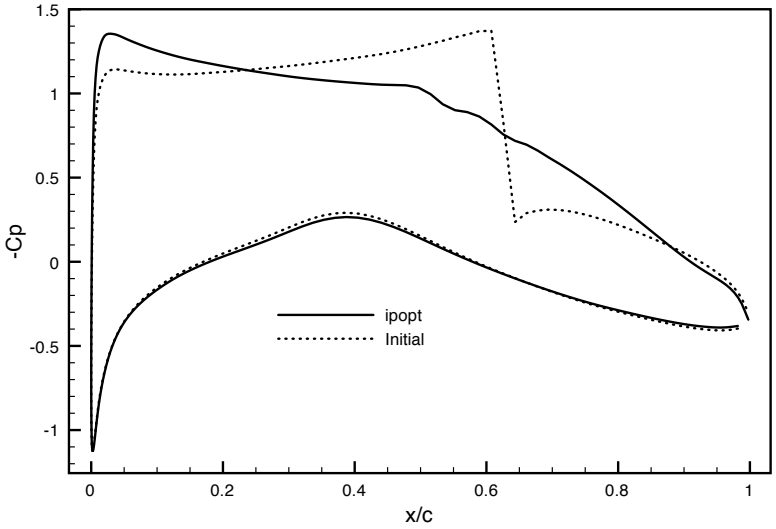
RAE2822: Lift-constrained drag minimization

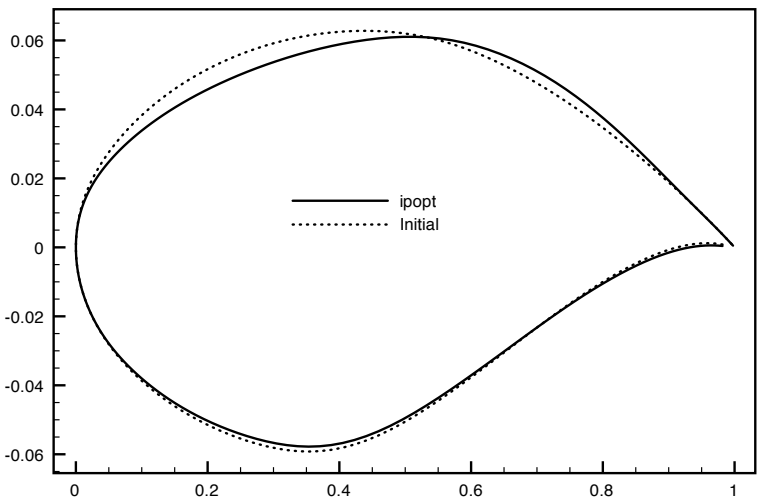


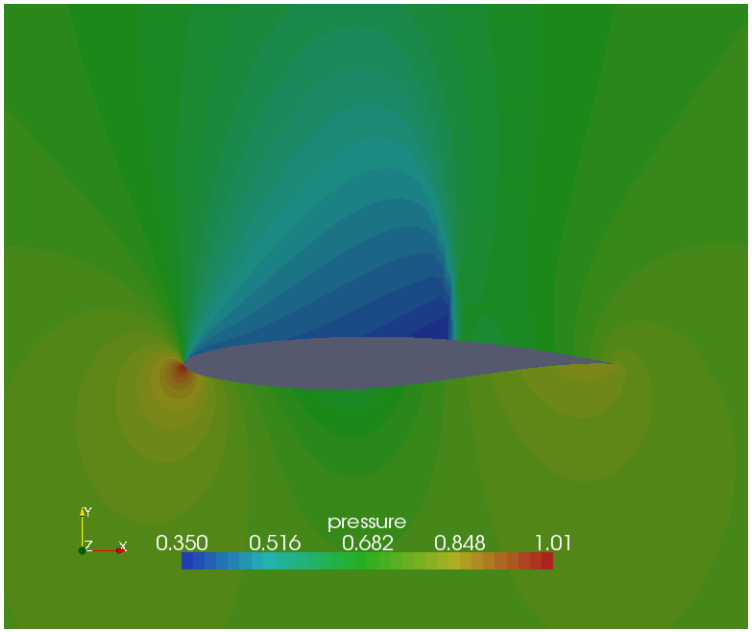


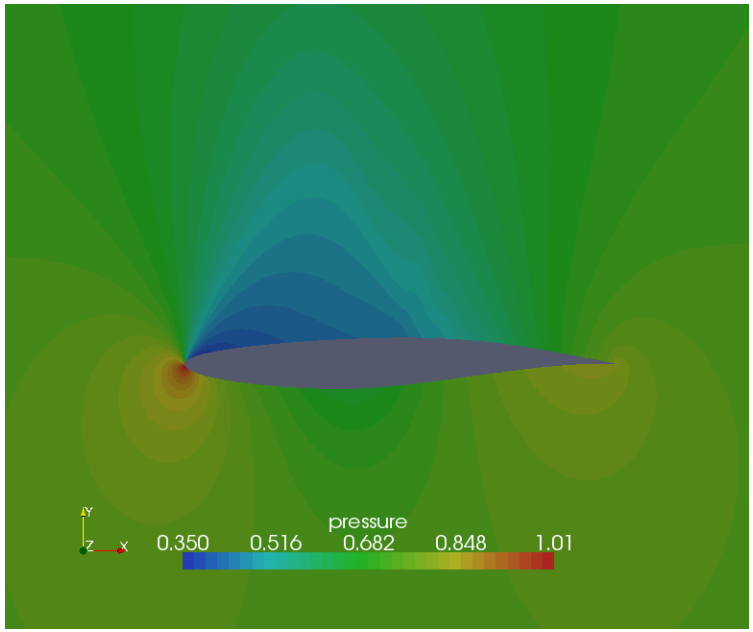
Method	$\mathfrak{J}$	$100C_d$	C_l	N_{fun}	N_{grad}
Initial	1.000	1.150	0.887	-	-
`conmin_mfd`	0.342	0.394	0.881	50	22
`fsqp`	0.325	0.374	0.887	50	32
`ipopt`	0.335	0.385	0.887	45	62

RAE2822: Lift-constrained drag minimization

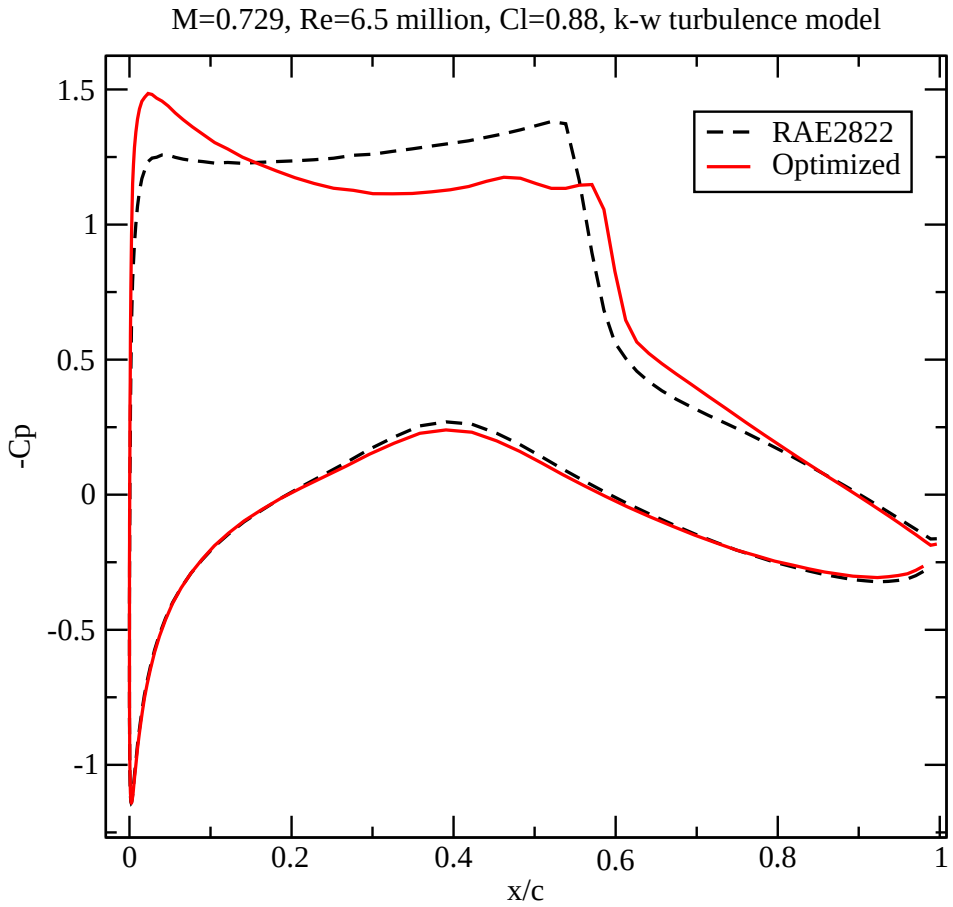






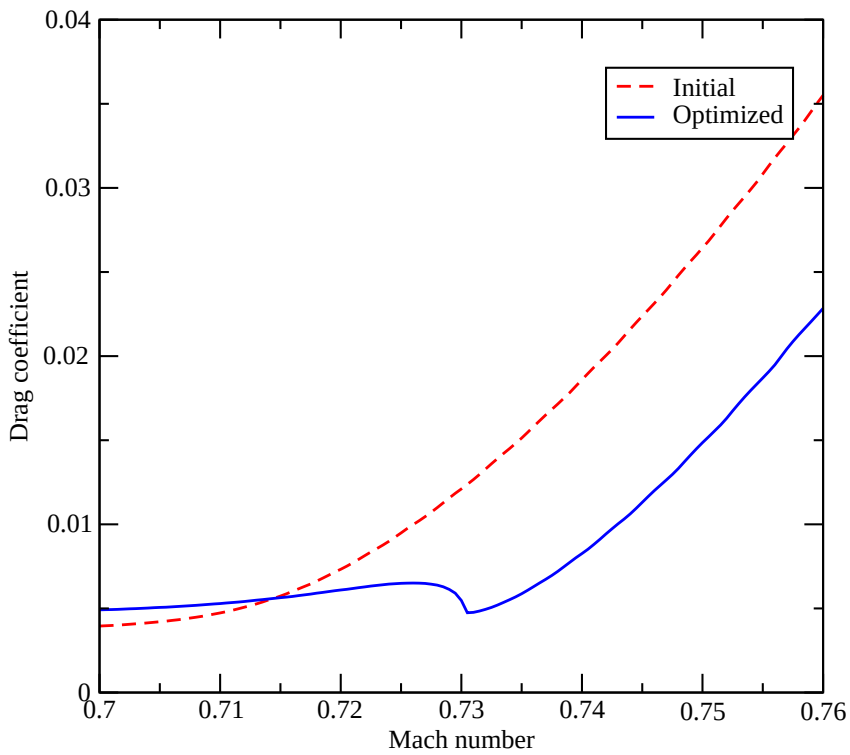


RANS computation

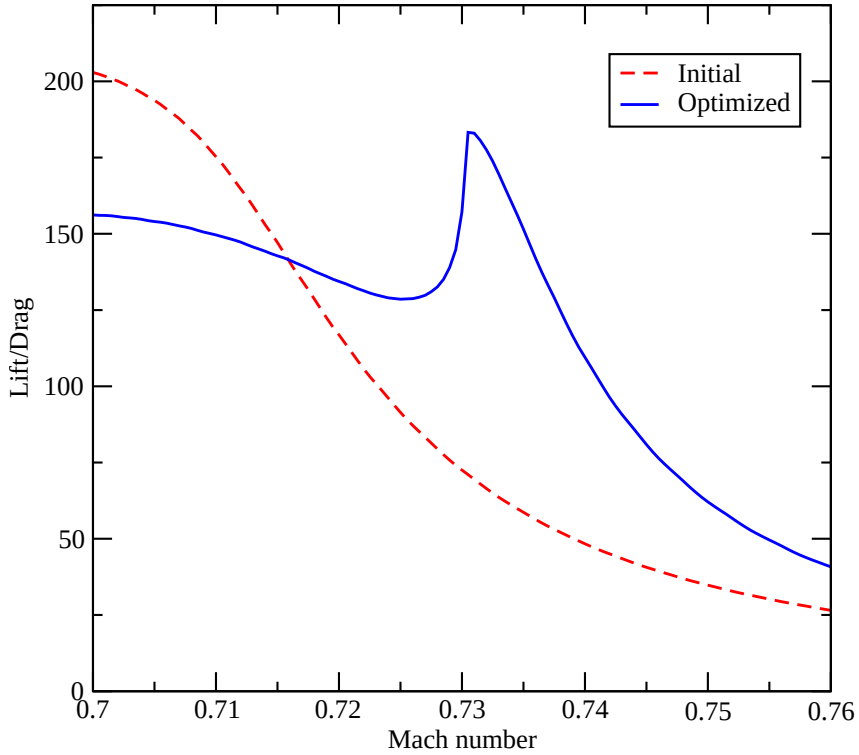


Shape	Inviscid			RANS		
	AOA	C_l	$100C_d$	AOA	C_l	$100C_d$
RAE2822	2.31	0.887	1.150	3.28	0.887	2.558
Optimized	2.31	0.887	0.386	3.22	0.885	1.692

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

Adjoints with shocks: Scalar conservation law

Consider the functional

$$J = \int_{-1}^{+1} G(u(x,T))\mathrm{d}x$$

where u is the solution of the conservation law

$$\begin{aligned} u_t + f(u)_x &= 0, \quad x \in (0,1), \quad t \in (0,T) \\ u(x,0) &= u_0(x) \end{aligned}$$

To compute the derivative of J wrt u_0 we make a small perturbation $u_0 \longrightarrow u_0 + \tilde{u}_0$. Then solution changes to $u + \tilde{u}$ and

$$\tilde{J} = \int g(u(x,T))\tilde{u}\mathrm{d}x, \quad g(u) = \frac{\mathrm{d}G}{\mathrm{d}u}$$

The equation for $\tilde{u}$ is obtained by linearizing the conservation law

$$\begin{aligned} (\tilde{u})_t + (f'(u)\tilde{u})_x &= 0 \\ \tilde{u}(x,0) &= \tilde{u}_0 \end{aligned}$$

To write $\tilde{J}$ in terms of $\tilde{u}_0$, we introduce the adjoint variable $v(x,t)$

$$\tilde{J} = \int g(u(x,T))\tilde{u}(x,T)\mathrm{d}x - \int_0^T \int v\left[(\tilde{u})_t + (f'(u)\tilde{u})_x\right]\mathrm{d}x\mathrm{d}t$$

Integrating by parts, we get

$$\begin{aligned}\tilde{J} &= \int v(x,0)\tilde{u}_0(x)\mathrm{d}x + \int (g-v)\tilde{u}(x,T)\mathrm{d}x \\ &\quad - \int_0^T \int \left[v_t + f'(u)v_x\right]\tilde{u}(x,t)\mathrm{d}x\mathrm{d}t\end{aligned}$$

To eliminate $\tilde{u}(x,t)$ and $\tilde{u}(x,T)$ from above equation, choose

$$\begin{aligned}v_t + f'(u)v_x &= 0 \\ v(x,T) &= g(u(x,T))\end{aligned}$$

The derivative is given by

$$\tilde{J} = \int v(x,0)\tilde{u}_0(x)\mathrm{d}x$$

Note that the adjoint equation must be solved backward in time, starting from $t = T$ to $t = 0$.

Example: Inviscid Burger's equation

$$
\begin{aligned}
u_t + (u^2/2)_x &= 0, \quad x \in \mathbb{R}, \quad t \in (0, T) \\
u(x, 0) &= \begin{cases} u_l & x < 0 \\ u_r & x > 0 \end{cases}
\end{aligned}
$$

Exact solution:

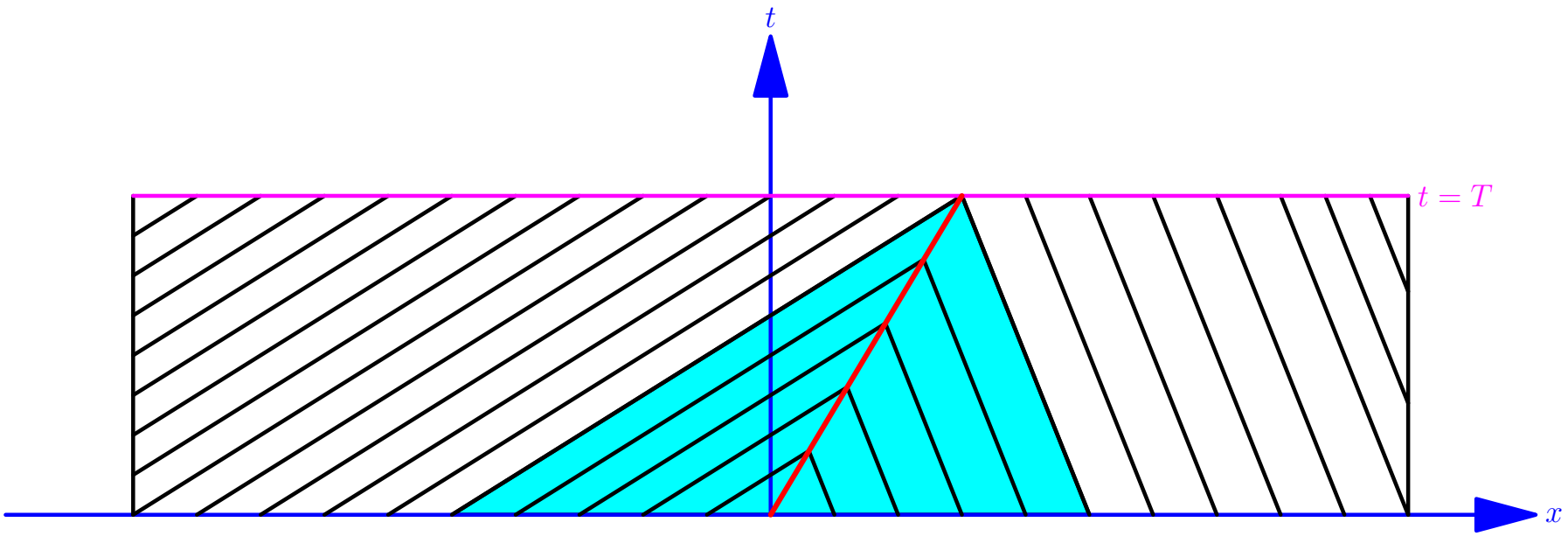
$$
u(x,t) = \begin{cases} u_l & x < St \\ u_r & x > St \end{cases} \qquad S = \frac{1}{2}(u_l + u_r)
$$

Consider the objective functional

$$
J = \frac{1}{2} \int u^2(x, T) \mathrm{d}x
$$

Then the corresponding adjoint equation is

$$
\begin{aligned}
v_t + u v_x &= 0 \\
v(x,T) &= u(x,T) = \begin{cases} u_l & x < ST \\ u_r & x > ST \end{cases}
\end{aligned}
$$



Adjoint solution cannot be determined in the shaded region. In other regions, solution can be determined by going forward in time along the characteristic until you hit $t = T$ line.

Back to scalar conservation law

Conservation laws can have discontinuous solutions, but the discontinuity must satisfy the Rankine-Hugoniot jump conditions.

If there is a discontinuity at $x = x_s$, then it must satisfy

$$f(u(x_s^+, t)) - f(u(x_s^-, t)) = S \cdot (u(x_s^+, t) - u(x_s^-, t))$$

where $S = \frac{\mathrm{d}x_s}{\mathrm{d}t}$ is the shock speed.

The linearization of the conservation law must also include the linearization of the jump conditions.

$$x_s \to x_s + \tilde{x}_s, \quad \frac{\mathrm{d}x_s}{\mathrm{d}t} \to \frac{\mathrm{d}x_s}{\mathrm{d}t} + \frac{\mathrm{d}\tilde{x}_s}{\mathrm{d}t}$$

$$\frac{\mathrm{d}\tilde{x}_s}{\mathrm{d}t}[u] + \frac{\mathrm{d}x_s}{\mathrm{d}t}[\tilde{u}] + \frac{\mathrm{d}x_s}{\mathrm{d}t}\tilde{x}_s \left[\frac{\partial u}{\partial x}\right] = \left[\frac{\mathrm{d}f}{\mathrm{d}u}\tilde{u}\right] + \tilde{x}_s \left[\frac{\mathrm{d}f}{\mathrm{d}u}\frac{\partial u}{\partial x}\right]$$

$$J = \int_{-1}^{x_s(T)} G(u(x,T))\mathrm{d}x + \int_{x_s(T)}^{+1} G(u(x,T))\mathrm{d}x$$

Perturbation in J is

$$\tilde{J} = \int\limits_{-1}^{x_s(T)} g(u(x,T))\tilde{u}(x,T)\mathrm{d}x + \int\limits_{x_s(T)}^{+1} g(u(x,T))\tilde{u}(x,T)\mathrm{d}x - [G]_{x_s(T)}\tilde{x}_s(T)$$

$$-\int\limits_{0}^{T}\int\limits_{-1}^{x_s(t)} v\left[\tilde{u}_t + (f'(u)\tilde{u})_x\right]\mathrm{d}x\mathrm{d}t - \int\limits_{0}^{T}\int\limits_{x_s(t)}^{+1} v\left[\tilde{u}_t + (f'(u)\tilde{u})_x\right]\mathrm{d}x\mathrm{d}t$$

$$-\int\limits_{0}^{T} v_s(t)\left\{\frac{\mathrm{d}\tilde{x}_s}{\mathrm{d}t}[u] + \frac{\mathrm{d}x_s}{\mathrm{d}t}[\tilde{u}] + \frac{\mathrm{d}x_s}{\mathrm{d}t}\tilde{x}_s\left[\frac{\partial u}{\partial x}\right] - \left[\frac{\mathrm{d}f}{\mathrm{d}u}\tilde{u}\right] - \tilde{x}_s\left[\frac{\mathrm{d}f}{\mathrm{d}u}\frac{\partial u}{\partial x}\right]\right\}\mathrm{d}t$$

After some work we get

$$\tilde{J} = \int_{-1}^{+1} v(x,0)\tilde{u}_0(x)\mathrm{d}x$$

provided the adjoint variables $v(x,t)$ and $v_s(t)$ satisfy the following equations

$$\begin{aligned}
v_t + f'(u)v_x &= 0 \\
v(x,T) &= g(u(x,T)) \\
v_s(t) &= v(x_s^+(t),t) = v(x_s^-(t),t) \\
\frac{\mathrm{d}}{\mathrm{d}t}v_s(t) &= 0 \\
v_s(T) &= \frac{[G]}{[u]}
\end{aligned}$$

Adjoint variable requires an interior boundary condition along the shock $(x_s(t),t)$. This uniquely determines the adjoint solution.

Steepest descent: limitations, problems

- Multiple optima are common
- Convergence towards local optimum:
  dependance on starting guess S^o
- Shape gradient: difficult to compute, or impossible
- Noisy cost function J

Gradient-free methods

- Simplex method (Nelder-Mead, Torczon)
- Global search methods
  - Genetic algorithm
  - Particle swarm optimization
  - Ant colony optimization
  - ...
- Do not require gradient
- Collection of M solutions at any iteration n

$$P^n = \{x_1^n, x_2^n, \ldots, x_M^n\} \subset \mathbb{R}^d$$

- Solutions evolve according to some rules

$$P^{n+1} = E(P^n)$$

Particle swarm optimization

- Kennedy and Eberhart (1995)
- Modeled on behaviour of animal swarms: ants, bees, birds
- Swarm intelligence

Optimization problem

$$\min_{x \in D} J(x), \quad D \subset \mathbb{R}^d$$

Particle swarm optimization

- Kennedy and Eberhart (1995)
- Modeled on behaviour of animal swarms: ants, bees, birds
- Swarm intelligence

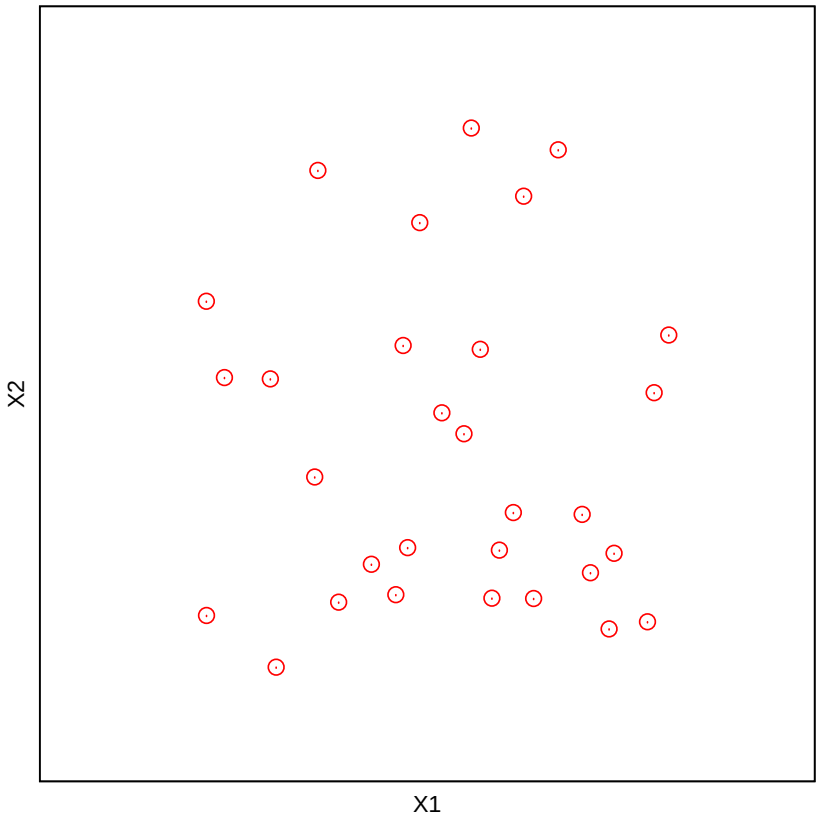
Optimization problem

$$\min_{x \in D} J(x), \quad D \subset \mathbb{R}^d$$

Particle swarm optimization

Particles distributed in design space

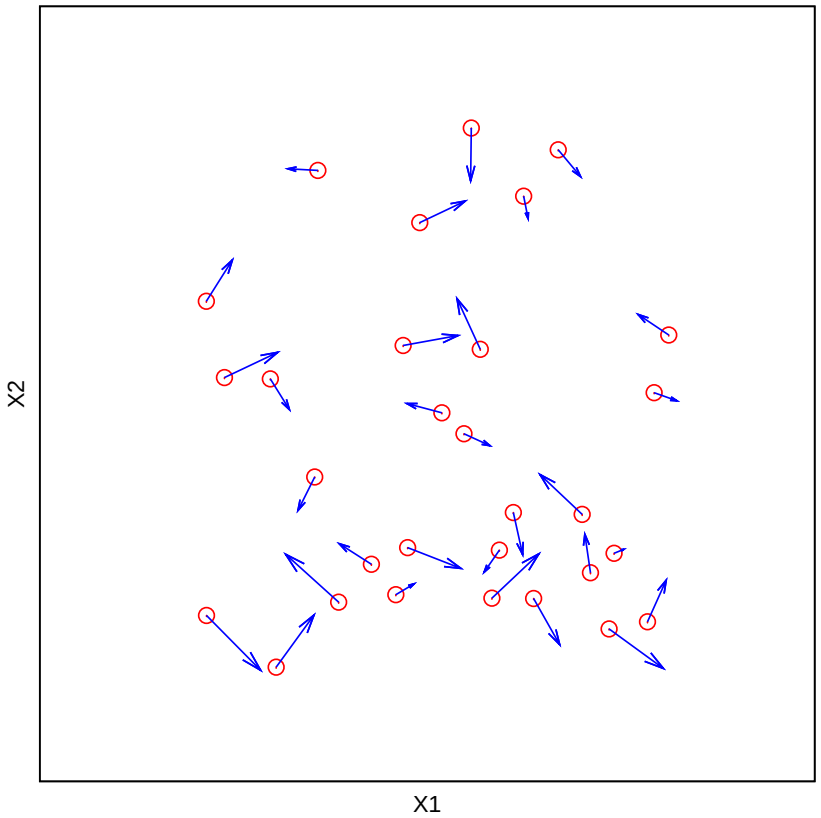
$$x_i \in D, \quad i = 1, ..., N_p$$



Particle swarm optimization

Each particle has a velocity

$$v_i \in \mathbb{R}^d, \quad i = 1, ..., N_p$$



Particle swarm optimization

- Particles have memory (t = iteration number)

$$\text{Local memory} \quad : p_i^t = \underset{0 \le s \le t}{\operatorname{argmin}} J(x_i^s)$$

$$\text{Global memory} \quad : p^t = \underset{i}{\operatorname{argmin}} J(p_i^t)$$

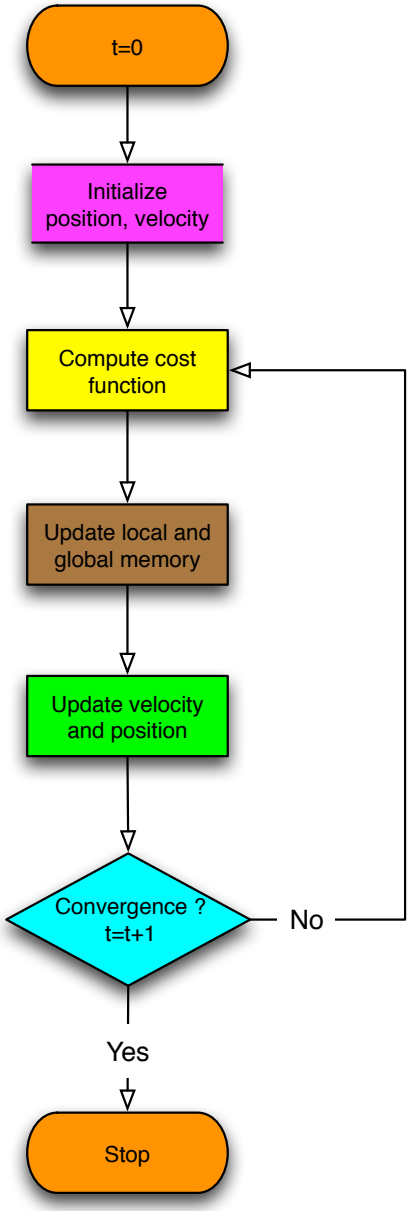
- Velocity update

$$v_i^{t+1} = \omega v_i^t + \underbrace{c_1 r_1^t (p_i^t - x_i^t)}_{Local} + \underbrace{c_2 r_2^t (p^t - x_i^t)}_{Global}$$

- Position update

$$x_i^{t+1} = x_i^t + v_i^{t+1}$$

Basic PSO algorithm



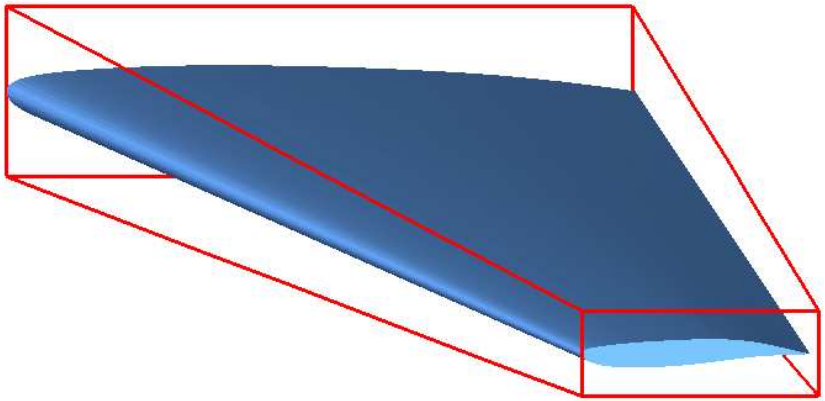
PSO: Parallelizability

$$
\begin{array}{cccc}
x_1^n & x_2^n & \dots & x_{N_p}^n \\
\downarrow & \downarrow & & \downarrow \\
J(x_1^n) & J(x_2^n) & \dots & J(x_{N_p}^n) \\
\downarrow & \downarrow & & \downarrow \\
\hline
v_1^{n+1} & v_2^{n+1} & \dots & v_{N_p}^{n+1} \\
\downarrow & \downarrow & & \downarrow \\
x_1^{n+1} & x_2^{n+1} & \dots & x_{N_p}^{n+1}
\end{array}
$$

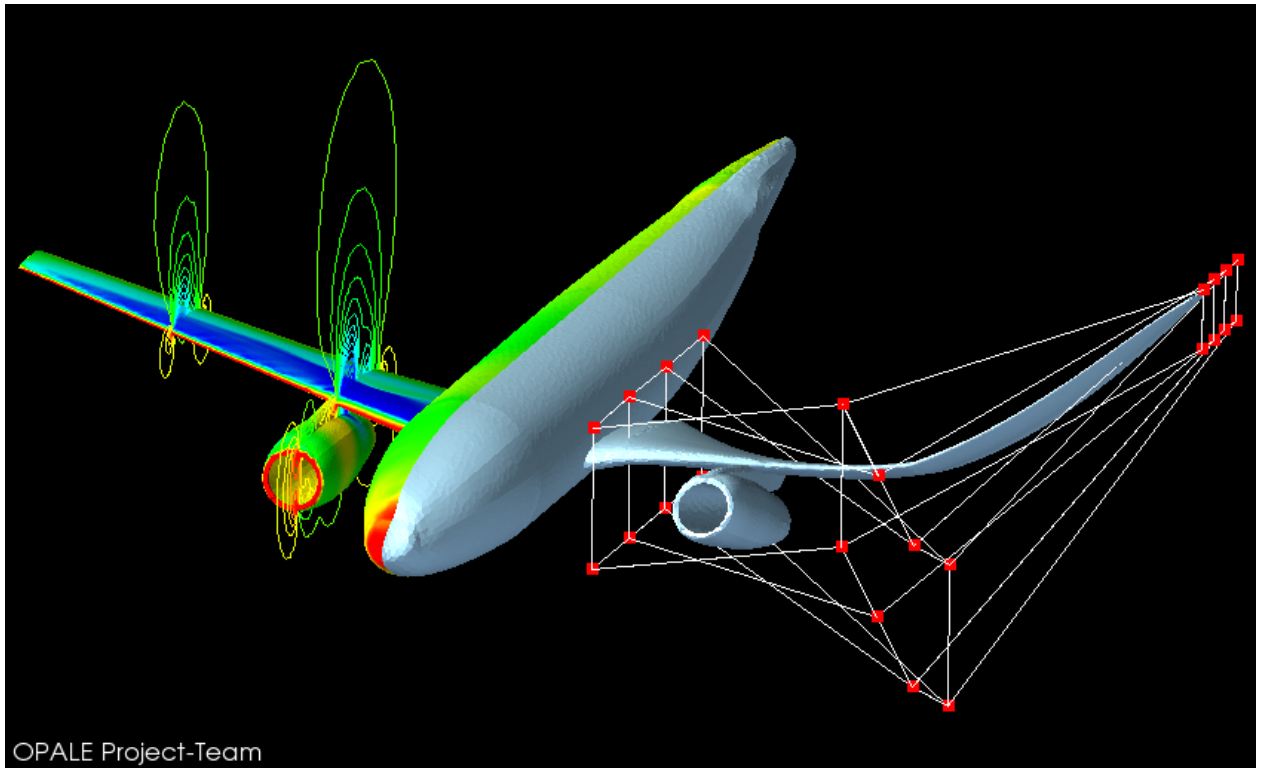
Parallel evaluation of cost functions using MPI: Message Passing Interface

Free Form Deformation

- Originated in computer graphics field
- Embed the object inside a box and deform the box
- Independent of the representation of the object



Free Form Deformation: Example



(Duvigneau, OPALE, INRIA)

Free Form Deformation

- $X^o(P)$ = coordinate of point P wrt reference shape
- Movement of point P under the deformation

$$X(P) = X^o(P) + \sum_{i=0}^{n_i} \sum_{j=0}^{n_j} \sum_{k=0}^{n_k} B_i^{n_i}(\xi_P) B_j^{n_j}(\eta_P) B_k^{n_k}(\zeta_P) Y_{ijk}$$

- Bernstein polynomials

$$B_p^n(t) = C_p^n t^p (1-t)^{n-p}$$

- Design variables

$$\{Y_{ijk}\}, \quad 0 \le i \le n_i, \quad 0 \le j \le n_j, \quad 0 \le k \le n_k$$

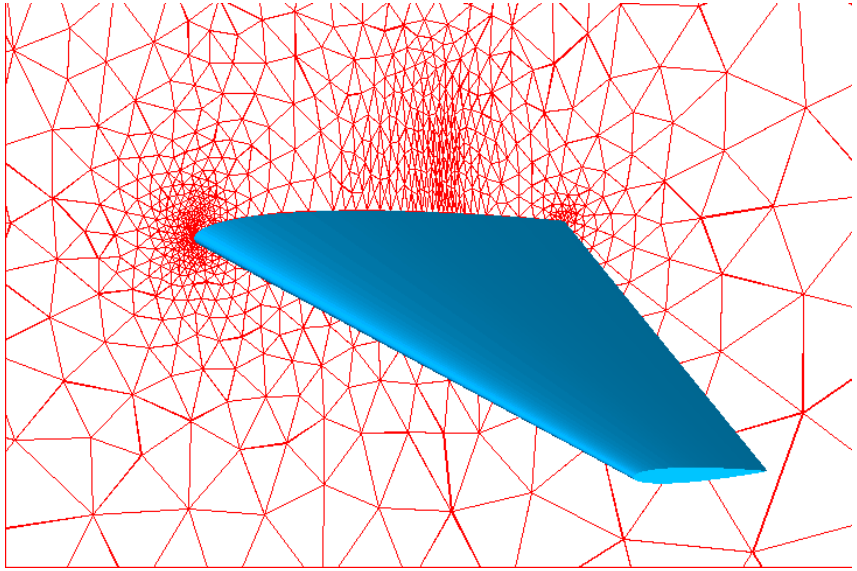
Test case: Wing shape optimization[3]

- Minimize drag under lift constraint

$$\min \frac{C_d}{C_{d_o}} \quad \text{s.t.} \quad \frac{C_l}{C_{l_o}} \geq 0.999$$

- FFD parameterization, $n = 20$ design variables
- Particle swarm optimization: 120 particles

$M_\infty = 0.83$, $\alpha = 2^o$



(Piaggio Aero. Ind.)
Grid: 31124 nodes

Cost function

$$\mathcal{J} = \frac{C_d}{C_{d_o}} + 10^4 \max\left(0, 0.999 - \frac{C_l}{C_{l_o}}\right)$$

[3] Joint work with R. Duvigneau, OPALE, INRIA

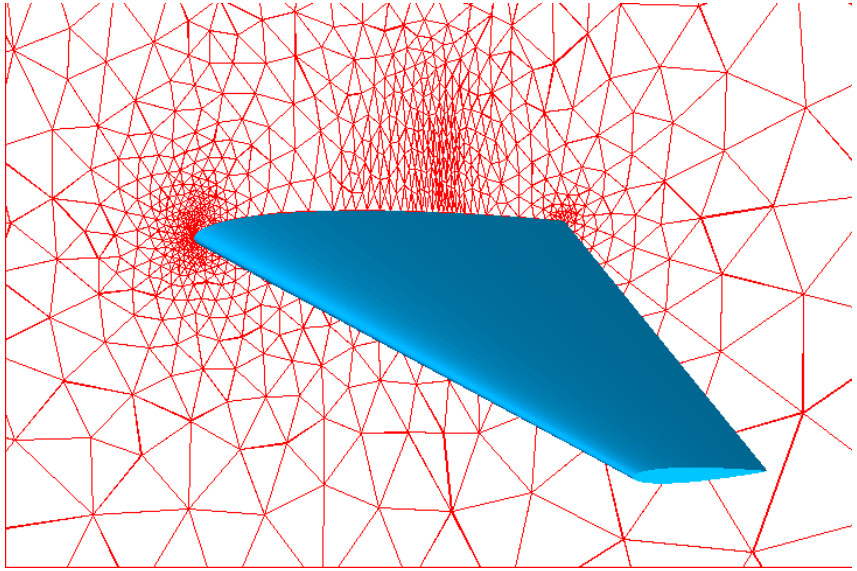
Test case: Wing shape optimization[3]

- Minimize drag under lift constraint

$$\min \frac{C_d}{C_{d_o}} \quad \text{s.t.} \quad \frac{C_l}{C_{l_o}} \geq 0.999$$

- FFD parameterization, $n = 20$ design variables
- Particle swarm optimization: 120 particles

$M_\infty = 0.83$, $\alpha = 2^o$



(Piaggio Aero. Ind.)
Grid: 31124 nodes

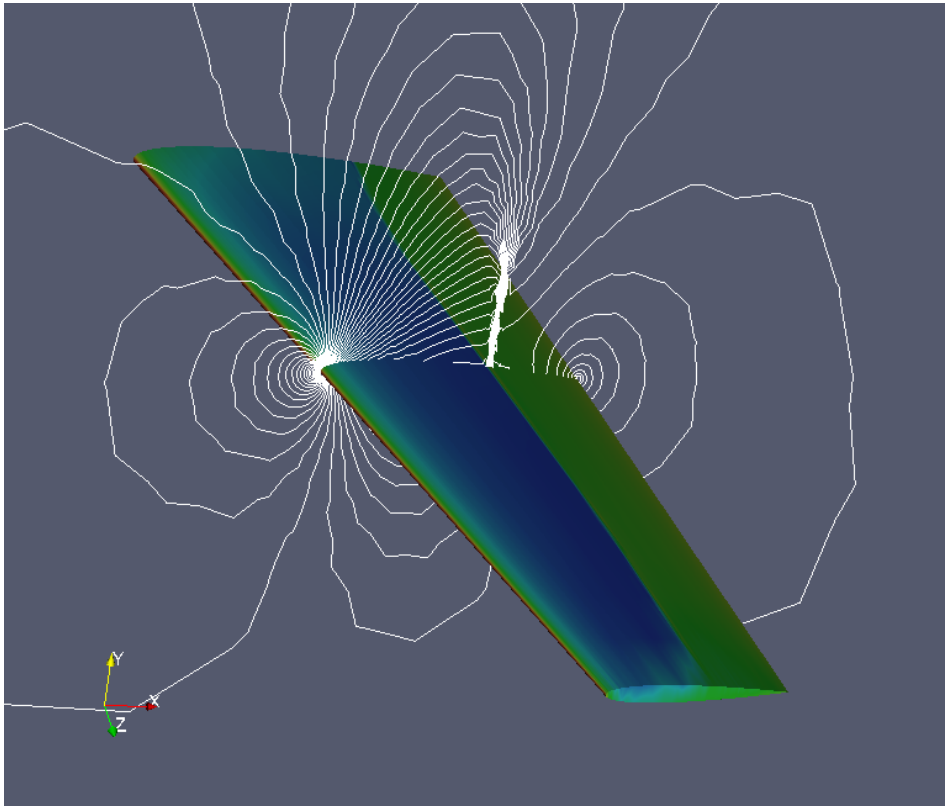
Cost function

$$\mathcal{J} = \frac{C_d}{C_{d_o}} + 10^4 \max\left(0, 0.999 - \frac{C_l}{C_{l_o}}\right)$$

[3] Joint work with R. Duvigneau, OPALE, INRIA

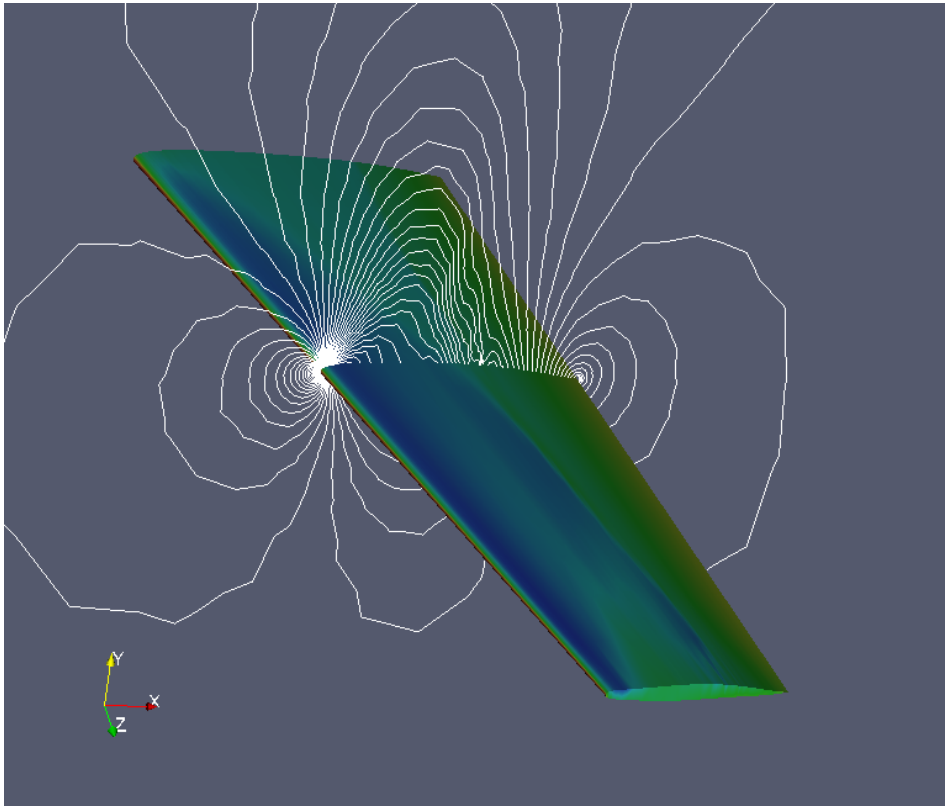
Wing optimization

Initial shape



Wing optimization

Optimized shape



PSO computational cost

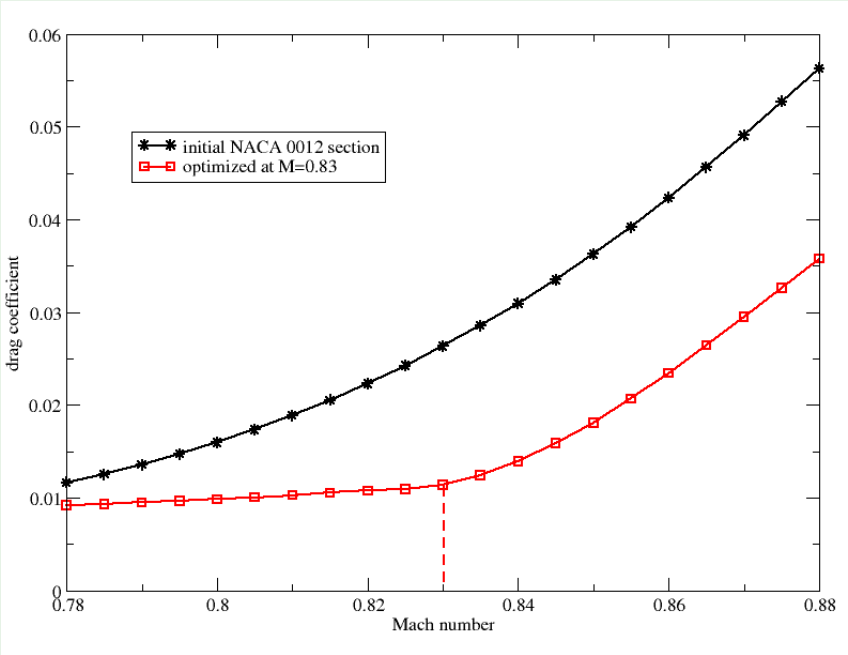
- Slow convergence: a few hundred iterations
- Require large swarm size: O(100)
- CFD is expensive: few minutes to hours
- Example: Wing optimization (coarse CFD grid)

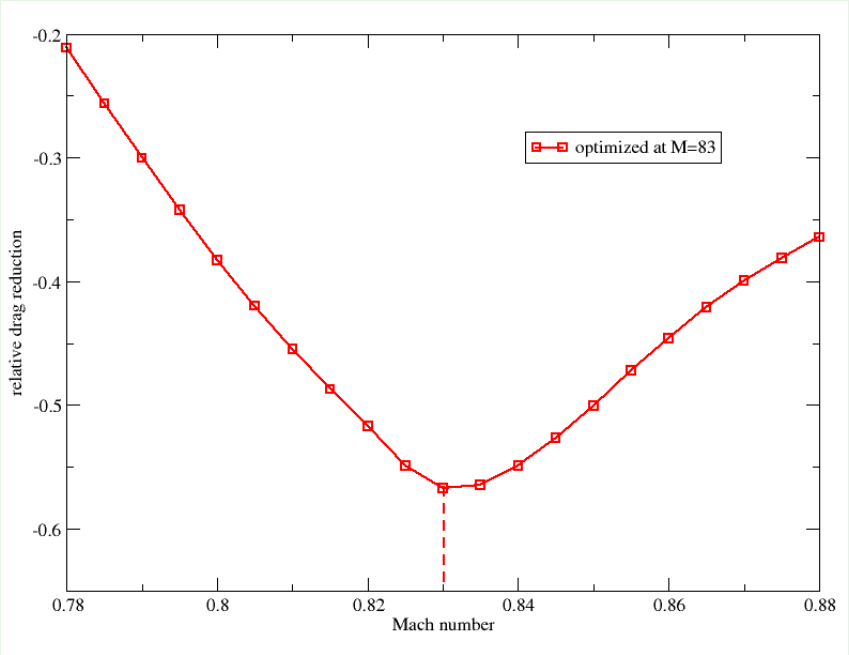
  (10 minutes/CFD) (120 CFD/iteration) (200 iterations) = 4000 hours

Robust design

Variability of the fitness $J(x, A)$ due to uncertain parameters A (mach number, angle of attack, shape, etc.)

Example : effect of Mach number fluctuations





Optimization problem

Optimization

$$\min_{x \in \mathbb{R}^d} \quad \mathcal{J}(x, a_o)$$
$$\mathcal{C}(x, a_o) \leq 0$$

Robust optimization

$$\min_{x \in \mathbb{R}^d} \begin{cases} \mu_J(x) = \displaystyle\int_{\Omega(A)} \mathcal{J}(x, a)\, \rho_A(a)\, da \\ \sigma_J^2(x) = \displaystyle\int_{\Omega(A)} [\mathcal{J}(x, a) - \mu_J]^2\, \rho_A(a)\, da \end{cases}$$
$$\text{Prob}[\mathcal{C}(x, A) \leq 0] \geq p$$

Optimization problem

Optimization

$$\min_{x \in \mathbb{R}^d} \quad \mathcal{J}(x, a_o)$$
$$\mathcal{C}(x, a_o) \leq 0$$

Robust optimization

$$\min_{x \in \mathbb{R}^d} \begin{cases} \mu_J(x) = \displaystyle\int_{\Omega(A)} \mathcal{J}(x, a)\, \rho_A(a)\, da \\ \sigma_J^2(x) = \displaystyle\int_{\Omega(A)} [\mathcal{J}(x, a) - \mu_J]^2\, \rho_A(a)\, da \end{cases}$$
$$\text{Prob}[\mathcal{C}(x, A) \leq 0] \geq p$$

Monte-Carlo estimation using meta-models

- For each design x: compute

$$J(x, a_1), J(x, a_2), \ldots, J(x, a_P)$$

$$\Downarrow$$

$$\text{meta-model } \tilde{\mathcal{J}}_x(a)$$

- Monte-Carlo estimation of the statistics: $N_s \gg 1$

$$\mathcal{M}_J(x) = \frac{1}{N_s} \sum_{i=1}^{N_s} \tilde{\mathcal{J}}_x(a_i)$$

$$\mathcal{S}_J^2(x) = \frac{1}{N_s - 1} \sum_{i=1}^{N_s} [\tilde{\mathcal{J}}_x(a_i) - \mathcal{M}_J]^2$$

Robust optimization problem: $0 \leq \omega \leq 1$

$$\min_{x \in \mathbb{R}^d} \quad \omega M_J + (1 - \omega) S_J + \text{constraint penalty}$$

Monte-Carlo estimation using meta-models

- For each design x: compute

$$J(x, a_1), J(x, a_2), \ldots, J(x, a_P)$$

$$\Downarrow$$

$$\text{meta-model } \tilde{\mathcal{J}}_x(a)$$

- Monte-Carlo estimation of the statistics: $N_s \gg 1$

$$\mathcal{M}_J(x) = \frac{1}{N_s} \sum_{i=1}^{N_s} \tilde{\mathcal{J}}_x(a_i)$$

$$\mathcal{S}_J^2(x) = \frac{1}{N_s - 1} \sum_{i=1}^{N_s} [\tilde{\mathcal{J}}_x(a_i) - \mathcal{M}_J]^2$$

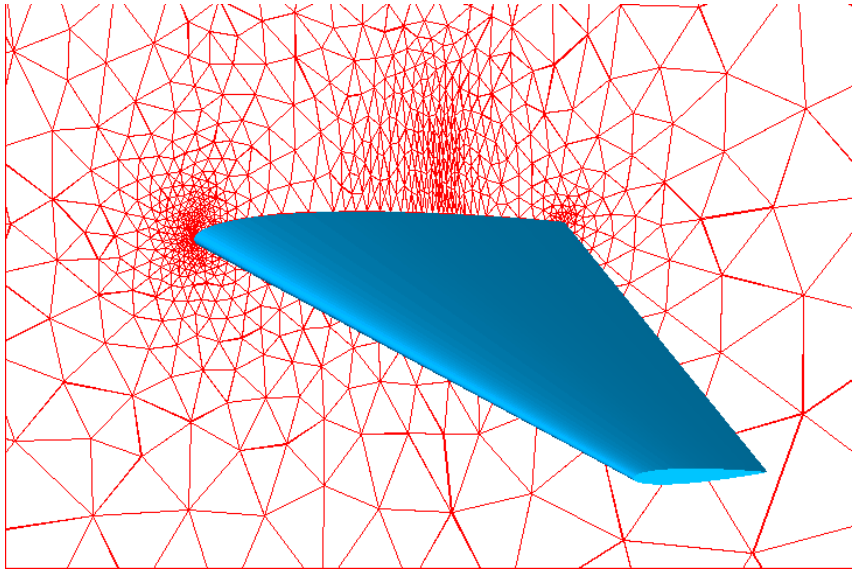
Robust optimization problem: $0 \leq \omega \leq 1$

$$\min_{x \in \mathbb{R}^d} \quad \omega M_J + (1 - \omega) S_J + \text{constraint penalty}$$

Test case: Wing shape optimization

- Minimize the drag mean and the drag variance
- Probabilistic lift constraint $p = 0.95$
- FFD parameterization, $n = 32$ design variables
- Particle swarm optimization: 32 particles
- Uncertain Mach number

$M_\infty = 0.83$, $\alpha = 2^o$

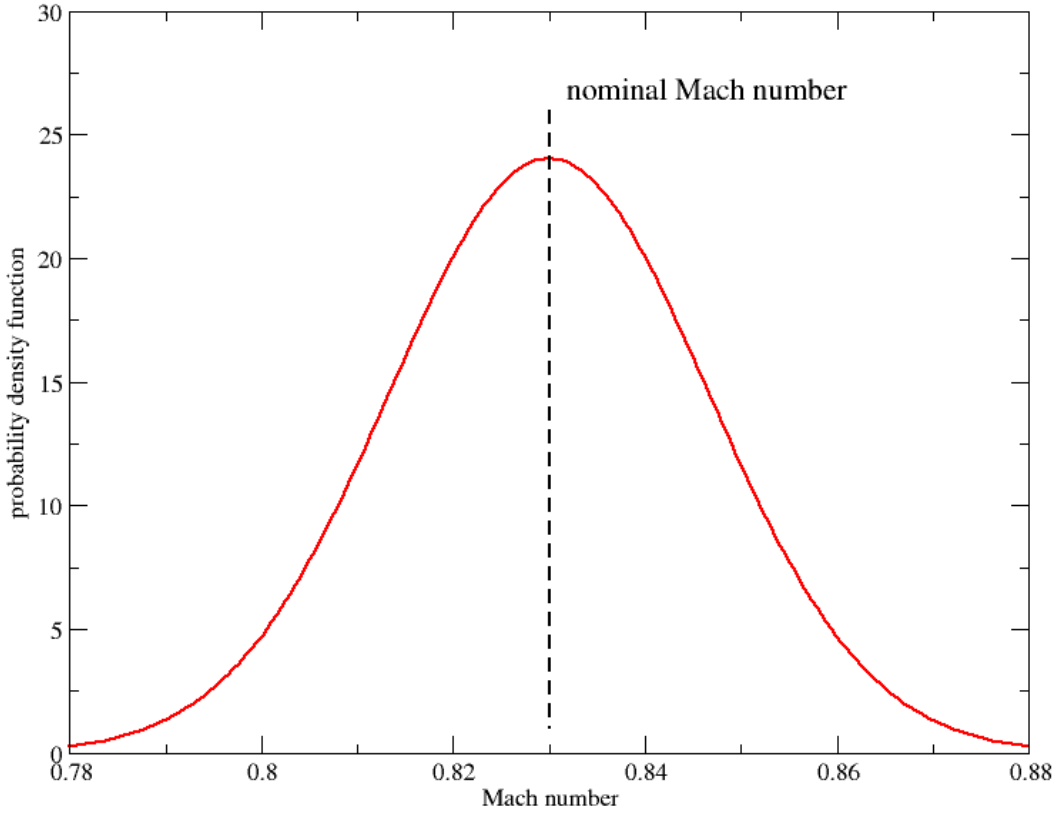


(Piaggio Aero. Ind.)
Grid: 31124 nodes

Number of CFD computations/iteration $= 32 \times 4 = 128$

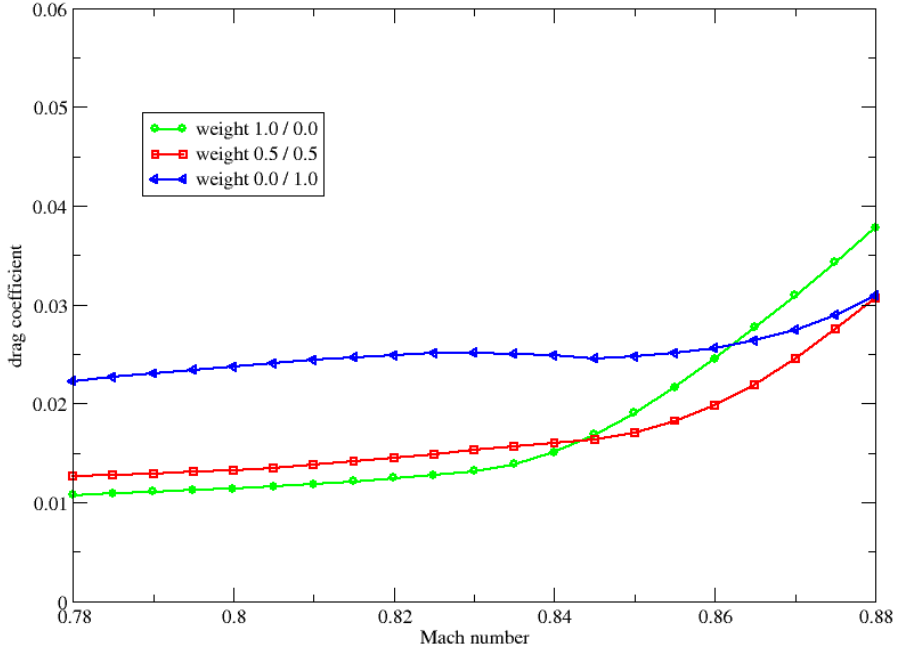
PDF of Mach number

$$\text{Mean} = 0.83, \text{ std. dev.} = 0.0166$$

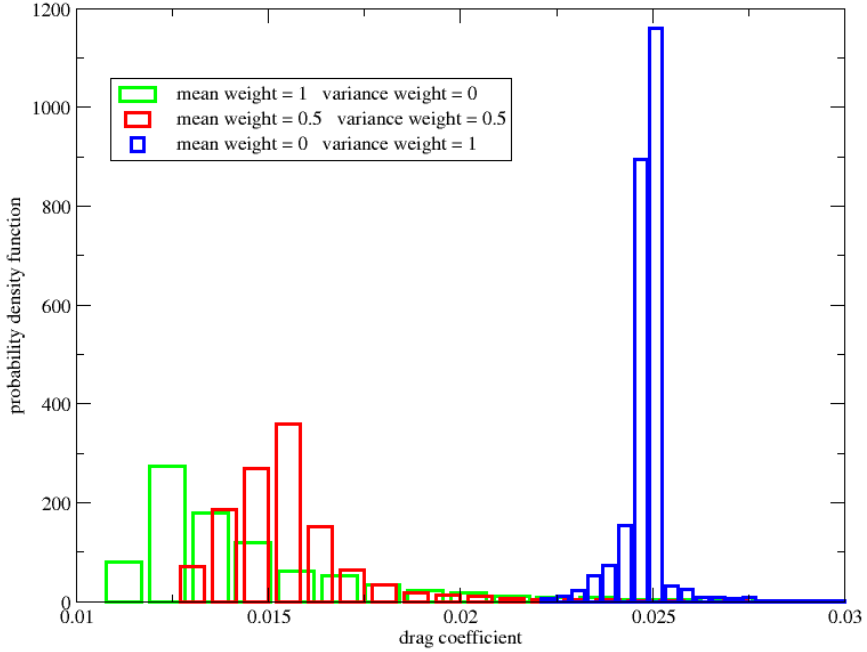


Wing optimization: result

Drag vs Mach number

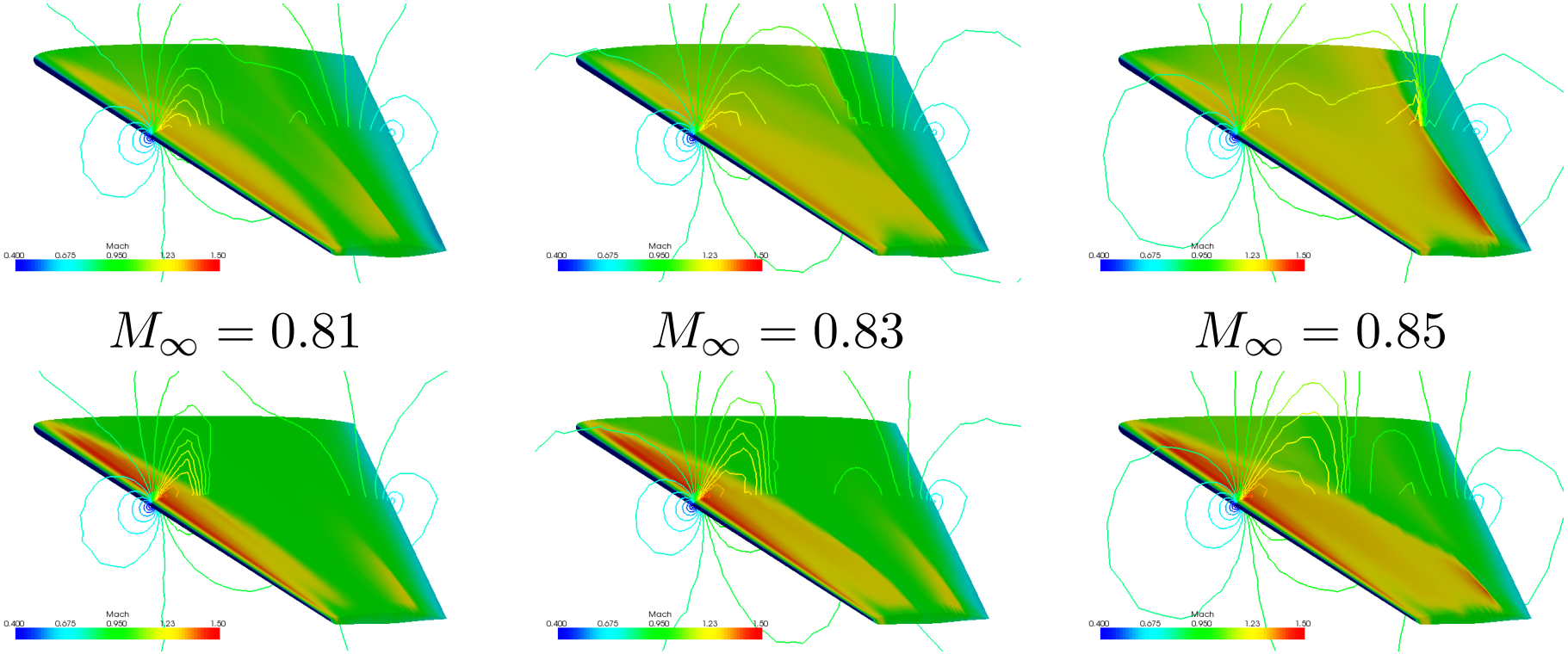


PDF of drag



Wing optimization: flow solutions

Optimal design for $M_\infty = 0.83$



$M_\infty = 0.81$ $M_\infty = 0.83$ $M_\infty = 0.85$

Robust design, weights (0.5, 0.5)