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

\[ \text{Lift} = \text{Weight} \]
\[ \text{Drag} = \text{Thrust} \]

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

http://www.aviation-history.com
Effect of shape on flow

(a) Blunt body

(b) Streamlined body

http://www.aerospaceweb.org
Wing and Airfoils

http://www.centennialofflight.gov
Compressible flow and shocks

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

Range, \( R = M \frac{a}{c_T} \frac{C_L}{C_D} \log \frac{W_i}{W_f} \)
Objectives and controls

- **Objective function** $\mathcal{J}(\beta)$
  mathematical representation of system performance

  $\mathcal{J}(\beta) = \mathcal{J}(\beta, Q(\beta))$

- **Control variables** $\beta$

  $\beta$ represents the shape

  We want to minimize/maximize $\mathcal{J}$ wrt $\beta$

- **State variable** $Q$: solution of an ODE or PDE

  $R(\beta, Q) = 0 \implies Q = Q(\beta)$

---

**Basic problem**

How to find the derivative of the objective function $\mathcal{J}$ wrt the design variables (shape) $\beta$?
Gradient-based minimization: blackbox/FD approach

\[ \min_{\beta \in \mathbb{R}^N} I(\beta, Q(\beta)) \]

- Initialize \( \beta^0, n = 0 \)
- For \( n = 0, \ldots, N_{\text{iter}} \)
  - Solve \( R(\beta^n, Q^n) = 0 \)
  - For \( j = 1, \ldots, N \)
    - \( \beta_{(j)}^n = [\beta_1^n, \ldots, \beta_j^n + \Delta \beta_j, \ldots, \beta_N^n]^\top \)
    - Solve \( R(\beta_{(j)}^n, Q_{(j)}^n) = 0 \)
    - \( \frac{dI}{d\beta_j} \approx \frac{I(\beta_{(j)}^n, Q_{(j)}^n) - I(\beta^n, Q^n)}{\Delta \beta_j} \)
  - Steepest descent step
    \[ \beta^{n+1} = \beta^n - s^n \frac{dI}{d\beta}(\beta^n) \]

Cost of FD-based steepest-descent

\[ \text{Cost} = O(N + 1) N_{\text{iter}} = O(N + 1) O(N) = O(N^2) \]
Accuracy of FD: Choice of step size

\[ \frac{d}{dx} f(x_0) = \frac{f(x_0 + \delta) - f(x_0)}{\delta} + O(\delta) \]

In principle, if we choose a small \( \delta \), we reduce the error.

But computers have finite precision. Instead of \( f(x_0) \) the computers gives \( f(x_0) + O(\epsilon) \) where \( \epsilon = \) machine precision.

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

\[ = \frac{d}{dx} f(x_0) + C_2 \delta + C_1 \frac{\epsilon}{\delta}, \quad \text{Total error} \]

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

<table>
<thead>
<tr>
<th>Precision</th>
<th>$\epsilon$</th>
<th>$\delta_{opt}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>Single</td>
<td>$10^{-8}$</td>
<td>$10^{-4}$</td>
</tr>
<tr>
<td>Double</td>
<td>$10^{-16}$</td>
<td>$10^{-8}$</td>
</tr>
</tbody>
</table>

See:
Brian J. McCartin, *Seven Deadly Sins of Numerical Computation*
Drag gradient using FD (Samareh)

Errors in Finite Difference Approximations

- Mid chord
- LE Sweep2
- Tip chord
- LE sweep3
- Twist (root)
- Twist (mid)
- Twist (tip)

% Error vs Scaled Step Size

Round-off
Truncation

LE Sweep2
LE Sweep3
Mid chord
Twist (root)
Twist (mid)
Twist (tip)
Iterative problems

\[ \mathcal{I}(\beta, Q), \quad \text{where} \quad R(\beta, Q) = 0 \]

- \( Q \) is implicitly defined, require an iterative solution method.
- Assume a \( Q^0 \) and iterate \( Q^n \rightarrow Q^{n+1} \) until \( \| R(\beta, Q^n) \| \leq \text{TOL} \)
- If \( \text{TOL} \) 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 \( \mathcal{I} \) 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.](image)

<table>
<thead>
<tr>
<th>iter</th>
<th>Lift</th>
<th>Drag</th>
</tr>
</thead>
<tbody>
<tr>
<td>41496</td>
<td>0.824485788042416</td>
<td>1.627593747613790E-002</td>
</tr>
<tr>
<td>41497</td>
<td>0.824485782714867</td>
<td>1.627593516695762E-002</td>
</tr>
<tr>
<td>41498</td>
<td>0.824485777387834</td>
<td>1.627593285794193E-002</td>
</tr>
<tr>
<td>41499</td>
<td>0.824485772061306</td>
<td>1.627593054909022E-002</td>
</tr>
<tr>
<td>41500</td>
<td>0.824485766735297</td>
<td>1.627592824040324E-002</td>
</tr>
</tbody>
</table>
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 \( \delta \) to be very small, \( \delta = 10^{-20} \)
- Can be easily implemented
  - fortran: redefine real variables as complex
  - matlab: no change
- Iterative problems: \( \beta \rightarrow \beta + i\Delta \beta \)
  - Obtain \( \tilde{Q} = Q(\beta + i\Delta \beta) \) by solving \( R(\beta + i\Delta \beta, \tilde{Q}) = 0 \)
  - Then gradient

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

- 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 \( \frac{dJ}{d\beta_i} \)

  - 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 \( \frac{dJ}{d\beta_i} = J_\beta \tilde{\beta}_{(i)} + J_Q \tilde{Q}_{(i)} \)

  - 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 \( \Psi \)

\[
J'(\beta) \tilde{\beta} = J_\beta \tilde{\beta} + J_Q \tilde{Q} + (R_\beta \tilde{\beta} + R_Q \tilde{Q}, \Psi) \\
= (J_\beta \tilde{\beta}, 1) + (J_Q \tilde{Q}, 1) + (R_\beta \tilde{\beta} + R_Q \tilde{Q}, \Psi) \\
= (\tilde{\beta}, J_\beta^*) + (\tilde{Q}, J_Q^*) + (\tilde{\beta}, R_\beta^* \Psi) + (\tilde{Q}, R_Q^* \Psi) \\
= (\tilde{\beta}, J_\beta^* + R_\beta^* \Psi) + (\tilde{Q}, J_Q^* + R_Q^* \Psi)
\]

Since \( \Psi \) was arbitrary, and we want to eliminate \( \tilde{Q} \), we can set

\[
J_Q^* + R_Q^* \Psi = 0
\]

Then the derivative is given by

\[
J'(\beta) \tilde{\beta} = (\tilde{\beta}, J_\beta^* + R_\beta^* \Psi)
\]
Adjoint approach

The adjoint equation

\[ R_Q^* \Psi + J_Q^* = 0 \]

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

The gradient can be computed cheaply from

\[ J'(\beta)\tilde{\beta} = (J^*_\beta + R^*_\beta \Psi, \tilde{\beta}) \]

which only requires scalar products.

Thus the cost of adjoint approach to compute gradient is \( O(1) \), i.e., it is independent of the number of design variables.
One-shot versus iterative methods

- **One-shot method:** solve for \((Q, \Psi, \beta)\) simultaneously\(^1\)

\[
\begin{align*}
R(\beta, Q) &= 0 \\
R^*_Q \Psi + J^*_Q &= 0 \\
J^*_\beta + R^*_\beta \Psi &= 0
\end{align*}
\]

- **Iterative method:** steepest descent
  1. Initial guess for \(\beta\)
  2. Solve for \(Q\) from \(R(\beta, Q) = 0\)
  3. Solve for \(\Psi\) from \(R^*_Q \Psi + J^*_Q = 0\)
  4. Compute gradient \(J'(\beta) = J^*_\beta + R^*_\beta \Psi\)
  5. Update \(\beta \leftarrow \beta - \varepsilon J'(\beta)\)
  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

PDE

Adjoint PDE

Discrete adjoint
Shape parameterization for gradient computation

- Mapping a reference domain \( \hat{\Omega} \subset \mathbb{R}^d \)

\[ T : \mathbb{R}^d \rightarrow \mathbb{R}^d, \quad T \in H^m(\mathbb{R}^d) \]

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

- Normal displacements
  - reference domain \( \hat{\Omega} \)
  - \( \Gamma \subset \partial \hat{\Omega} \) to be optimized
  - perturb \( \Gamma \) 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

\[ J(\Omega, u) = \int_{\Omega} F(u) dx, \quad u = u(\Omega) \]

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

\[ J(\Omega_\epsilon, u^\epsilon) - J(\Omega, u) = \int_{\Omega_\epsilon - \Omega} F(u^\epsilon) dx - \int_{\Omega - \Omega_\epsilon} F(u) dx + \int_{\Omega \cap \Omega_\epsilon} [F(u^\epsilon) - F(u)] dx \]

Define the following boundary parts

\[ \Gamma^+ = \{ s \in \Gamma : \alpha(s) > 0 \} \]
\[ \Gamma^- = \{ s \in \Gamma : \alpha(s) \leq 0 \} \]
Then we approximate the integrals as follows:

\[
\int_{\Omega_\epsilon - \Omega} F(u^\epsilon) \, dx = +\epsilon \int_{\Gamma^+} \alpha(s) F(u) \, ds + O(\epsilon^2)
\]

\[
\int_{\Omega - \Omega_\epsilon} F(u) \, dx = -\epsilon \int_{\Gamma^-} \alpha(s) F(u) \, ds + O(\epsilon^2)
\]

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

\[
\int_{\Omega \cap \Omega_\epsilon} [F(u^\epsilon) - F(u)] \, dx = \epsilon \int_{\Omega} F'(u) \tilde{u} \, dx + O(\epsilon^2)
\]

Shape derivative is

\[
\frac{1}{\epsilon} [J(\Omega_\epsilon) - J(\Omega)] = \int_{\Gamma} \alpha(s) F(u) \, ds + \int_{\Omega} F'(u) \tilde{u} \, dx + O(\epsilon)
\]
Derivative of boundary functional

\[ J(\Gamma, u) = \int_{\Gamma} f(u) \, ds \]

We perturb \( \Gamma \) by normal displacements

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

We must consider change in area element \( ds \) due to this perturbation. First look at a line element \( ds \) with radius of curvature \( R \), \( ds = Rd\theta \)

Due to boundary perturbation, line element changes to

\[ ds^\epsilon = \left( 1 - \epsilon \frac{\alpha}{R} \right) ds \]

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 $\int_{\Gamma_\epsilon} f(u^\epsilon)ds$ in terms of integral and quantities on $\Gamma$

Consider $x \in \Gamma$ and corresponding $x^\epsilon \in \Gamma_\epsilon$ given by $x^\epsilon = x + \epsilon \alpha n$

\begin{align*}
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)ds^\epsilon &= f(u)ds + \epsilon \left[\alpha f'(u) \frac{\partial u}{\partial n} - \frac{\alpha}{R} f(u)\right] ds + \epsilon f'(u) \tilde{u} + O(\epsilon^2)
\end{align*}
Finally the sensitivity is given by

\[
\frac{1}{\epsilon} [J(\Gamma_\epsilon) - J(\Gamma)] = \int_\Gamma \alpha \left[ f'(u) \frac{\partial u}{\partial n} - \frac{f(u)}{R} \right] ds + \int_\Gamma f'(u) \tilde{u} ds + O(\epsilon)
\]
Euler equations: compressible, inviscid flow

\[ \rho = \text{density} \]
\[ u = (u_1, u_2, u_3) = \text{velocity} \]
\[ p = \text{pressure} \]

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,

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

\[ \gamma = \frac{C_p}{C_v} \quad \text{For air } \gamma = 1.4 \]
Conservation of mass

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

$$\frac{d}{dt} \int_{\Omega} \rho \, dx = - \oint_{\partial \Omega} \rho (u \cdot n) \, ds$$

Using divergence theorem

$$\frac{d}{dt} \int_{\Omega} \rho \, dx = - \int_{\Omega} \text{div} (\rho u) \, dx$$

Since this should hold for any control volume $\Omega$

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

$$\frac{d}{dt} \int_{\Omega(t)} \rho u_i \, dx = \int_{\partial \Omega(t)} R_i \, ds + \int_{\Omega(t)} \rho f_i \, dx$$

For an inviscid fluid, $R = -\rho n$. The left hand side is

$$\frac{d}{dt} \int_{\Omega(t)} \rho u_i \, dx = \int_{\Omega(t)} \frac{\partial}{\partial t} (\rho u_i) \, dx + \int_{\partial \Omega(t)} (\rho u_i)(u \cdot n) \, ds$$

$$= \int_{\Omega(t)} \frac{\partial}{\partial t} (\rho u_i) \, dx + \int_{\Omega(t)} \text{div}(\rho u_i u) \, dx$$

$$\int_{\Omega(t)} \frac{\partial}{\partial t} (\rho u_i) \, dx + \int_{\Omega(t)} \text{div}(\rho u_i u) \, dx = -\int_{\Omega} \frac{\partial p}{\partial x_i} \, dx + \int_{\Omega} \rho f_i \, dx$$

$$\frac{\partial}{\partial t} (\rho u_i) + \text{div}(\rho u_i u) + \frac{\partial p}{\partial x_i} = \rho f_i$$
Conservation of energy

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

\[ \frac{d}{dt} \int_{\Omega(t)} E \, dx = \int_{\partial \Omega(t)} (R \cdot u) \, ds + \int_{\Omega(t)} \rho (f \cdot u) \, dx \]

Using divergence theorem

\[ \int_{\Omega} \frac{\partial E}{\partial t} \, dx + \int_{\Omega} \text{div}(Eu) \, dx = - \int_{\Omega} \text{div}(pu) \, dx + \int_{\Omega(t)} \rho (f \cdot u) \, dx \]

\[ \frac{\partial E}{\partial t} + \text{div}\left[ (E + p)u \right] = \rho f \cdot u \]
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**
  \[
  \frac{\partial}{\partial t} (\rho u_1) + \frac{\partial}{\partial x_1} (p + \rho u_1^2) + \frac{\partial}{\partial x_2} (\rho u_1 u_2) + \frac{\partial}{\partial x_3} (\rho u_1 u_3) = 0
  \]
  \[
  \frac{\partial}{\partial t} (\rho u_2) + \frac{\partial}{\partial x_1} (\rho u_1 u_2) + \frac{\partial}{\partial x_2} (p + \rho u_2^2) + \frac{\partial}{\partial x_3} (\rho u_2 u_3) = 0
  \]
  \[
  \frac{\partial}{\partial t} (\rho u_3) + \frac{\partial}{\partial x_1} (\rho u_1 u_3) + \frac{\partial}{\partial x_2} (\rho u_2 u_3) + \frac{\partial}{\partial x_3} (p + \rho u_3^2) = 0
  \]

- **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

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

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 $\Gamma$ such that the pressure on the airfoil is close to a specified pressure $p^*$, i.e.,

$$\min_{\Gamma} \frac{1}{2} \int_{\Gamma} (p - p^*)^2 \, ds$$

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 $\Gamma$, the solution changes from $U$ to $U + \epsilon \tilde{U}$

\[
F(U + \epsilon \tilde{U}) = F(U) + \epsilon \frac{\partial F}{\partial U}(U)\tilde{U} + O(\epsilon^2)
= F(U) + \epsilon A\tilde{U} + O(\epsilon^2)
\]

Equation governing first order perturbation

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

For an arbitrary function $V$

\[
\int_{\Omega} V \cdot (A\tilde{U})_x = - \int_{\Omega} V_x \cdot A\tilde{U} + \int_{\partial\Omega} V \cdot A\tilde{U} n_x
= - \int_{\Omega} A^t V_x \cdot \tilde{U} + \int_{\partial\Omega} V \cdot A\tilde{U} n_x
\]

Then

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

\[-\int_{\Omega} (A^t V_x + B^t V_y + C^t V_z) \cdot \tilde{U} + \int_{\partial \Omega} V \cdot (An_x + Bn_y + Cn_z) \tilde{U} = 0\]

Define

\[D = An_x + Bn_y + Cn_z\]

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 \(\Gamma\), \(u \cdot n = 0\)

\[DU|_{\Gamma} = \begin{bmatrix} 0 \\ pn \\ 0 \end{bmatrix} \]
Since $DU = Fn_x + Gn_y + Hn_z$

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

Also

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

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

$$
\int_{\Gamma} V \cdot D\tilde{U} = \int_{\Gamma} V \cdot \widetilde{DU}
= \int_{\Gamma} V \cdot [0 \quad \tilde{p}n \quad 0] = \int_{\Gamma} v \cdot \tilde{p}n
= \int_{\Gamma} \tilde{p}(v \cdot n) - \int_{\Gamma} p \sum_{j=1}^{d-1} \frac{\partial \alpha}{\partial t_j} (v \cdot t_j)
$$
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 \( \alpha \) we must eliminate \( \tilde{p} \). We add the first order perturbation equation to the above equation

\[
\frac{1}{\epsilon} \tilde{J} = RHS - \int_{\Omega} (A^t V_x + B^t V_y + C^t V_z) \cdot \tilde{U} + \int_{\partial \Omega} V \cdot D \tilde{U}
\]

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

\[
A^t V_x + B^t V_y + C^t V_z = 0, \quad \text{in} \quad \Omega
\]
\[
\frac{1}{\epsilon} \tilde{j} = RHS + \int_{\Gamma} V \cdot D\tilde{U} + \int_{\Gamma_o} V \cdot D\tilde{U}
\]

\[
= \int_{\Gamma} \left( p - p^* + v \cdot n \right) \tilde{p}
\]

\[
+ \int_{\Gamma} \alpha \left[ (p - p^*) \frac{\partial p}{\partial n} - \frac{(p - p^*)^2}{2R} \right]
\]

\[
- \int_{\Gamma} p \sum_{j=1}^{d-1} \frac{\partial \alpha}{\partial t_j} (v \cdot t_j)
\]

\[
+ \int_{\Gamma_o} D^t V \cdot \tilde{U}
\]

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

\[
p - p^* + v \cdot n = 0, \quad \text{on} \quad \Gamma
\]
We choose \( V \) on \( \Gamma_o \) to eliminate the integral over \( \Gamma_o \)

- **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:**

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

  We have \((T \tilde{U})_{1,2,3,4} = 0\) while \((T \tilde{U})_5\) is arbitrary, so we choose 
  \((T^{-t} D^t V)_5 = 0\)

- **Subsonic outflow:** \((T \tilde{U})_1\) is specified while \((T \tilde{U})_{2,3,4,5}\) is arbitrary.
  Hence we choose \((T^{-t} D^t V)_{2,3,4,5} = 0\)
Adjoint Euler equations

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

with boundary conditions

- supersonic inflow
  
  $$\text{none}$$

- supersonic outflow
  
  $$V = 0$$

- subsonic inflow
  
  $$(T^{-t} D^t V)_5 = 0$$

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

- solid wall
  
  $$p - p^* + v \cdot n = 0$$
Shape derivative

\[
\frac{1}{\epsilon} \tilde{J} = \int_{\Gamma} \alpha \left[ (p - p^*) \frac{\partial p}{\partial n} - \frac{(p - p^*)^2}{2R} \right] - \int_{\Gamma} p \sum_{j=1}^{d-1} \frac{\partial \alpha}{\partial t_j} (v \cdot t_j)
\]

\[
= \int_{\Gamma} \alpha \left[ (p - p^*) \frac{\partial p}{\partial n} - \frac{(p - p^*)^2}{2R} - \text{div}_{\Gamma}(pv) \right]
\]

Hence

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

Steepest descent update

\[
\alpha^{n+1} = \alpha^n - s^n \nabla_{\alpha} J(\alpha^n)
\]
Discrete adjoint approach

- Approximate PDE using **finite volume method**

\[ R(X, Q) = 0 \]

where

\[
\begin{align*}
X &= \text{grid coordinates, } X \in \mathbb{R}^m \\
Q &= \text{solution vector, } Q \in \mathbb{R}^n
\end{align*}
\]

\[ R : \mathbb{R}^m \times \mathbb{R}^n \rightarrow \mathbb{R}^n \]

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

\[ \mathcal{J}(X, Q) \]
Elements of practical shape optimization

\[ \frac{d\mathcal{J}}{d\beta} = \frac{d\mathcal{J}}{dX} \frac{dX}{dX_s} \frac{dX_s}{d\beta} \]
Adjoint approach: \( \frac{d\mathcal{J}}{dX} \)

- For shape optimization: \( \mathcal{J} = \mathcal{J}(X, Q) \)
  \[
  \frac{d\mathcal{J}}{dX} = \frac{\partial \mathcal{J}}{\partial X} + \frac{\partial \mathcal{J}}{\partial Q} \frac{\partial Q}{\partial X}
  \]

- 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
  \[
  \frac{d\mathcal{J}}{dX} = \left[ \frac{\partial \mathcal{J}}{\partial X} + \frac{\partial \mathcal{J}}{\partial Q} \frac{\partial Q}{\partial X} \right] + \Psi^t \left[ \frac{\partial R}{\partial X} + \frac{\partial R}{\partial Q} \frac{\partial Q}{\partial X} \right]
  \]
Adjoint approach

- Collect terms involving the flow sensitivity

\[
\frac{d\mathcal{J}}{dX} = \left[ \frac{\partial \mathcal{J}}{\partial X} + \Psi^t \frac{\partial R}{\partial X} \right] + \left[ \frac{\partial \mathcal{J}}{\partial Q} + \Psi^t \frac{\partial R}{\partial Q} \right] \frac{\partial Q}{\partial X}
\]

- Choose \( \Psi \) so that flow sensitivity vanishes

\[
\frac{\partial \mathcal{J}}{\partial Q} + \Psi^t \frac{\partial R}{\partial Q} = 0 \quad \Rightarrow \quad \left( \frac{\partial R}{\partial Q} \right)^t \Psi + \left( \frac{\partial \mathcal{J}}{\partial Q} \right)^t = 0
\]

- Gradient

\[
\frac{d\mathcal{J}}{dX} = \frac{\partial \mathcal{J}}{\partial X} + \Psi^t \frac{\partial R}{\partial X}
\]

Cost of Adjoint-based steepest-descent

\[
\text{Cost} = O(1)N_{iter} = O(N)
\]
Optimization steps

• $\beta \implies X_s \implies X$

• Solve the flow (primal) equations to steady-state

\[
X \implies \frac{dQ}{dt} + R(X, Q) = 0 \implies Q, \quad \mathcal{J}
\]

• Solve adjoint equations to steady-state

\[
X, Q \implies \frac{d\Psi}{dt} + \left(\frac{\partial R}{\partial Q}\right)^t \Psi + \left(\frac{\partial \mathcal{J}}{\partial Q}\right)^t = 0 \implies \Psi
\]

• Compute gradient wrt grid $X$

\[
\frac{d\mathcal{J}}{dX} = \frac{\partial \mathcal{J}}{\partial X} + \Psi^t \frac{\partial R}{\partial X}
\]

\[
\frac{d\mathcal{J}}{d\beta} = \frac{d\mathcal{J}}{dX} \frac{dX}{dX_s} \frac{dX_s}{d\beta} \implies \beta \leftarrow \beta - \epsilon \frac{d\mathcal{J}}{d\beta}
\]
Automatic Differentiation

- Discrete adjoint method requires

\[ \frac{\partial J}{\partial X}, \frac{\partial J}{\partial Q}, \left( \frac{\partial R}{\partial X} \right)^t \psi, \left( \frac{\partial R}{\partial Q} \right)^t \psi \]

- 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**

\[ X \in \mathbb{R}^m, \quad \bar{Y} \in \mathbb{R}^n \quad \rightarrow \quad \left( \frac{\partial F}{\partial X} \right)^t \bar{Y} \in \mathbb{R}^m \]
Automatic Differentiation: TAPENADE
Code residue.f to compute $X, Q \rightarrow R(X, Q) = \text{RES}$

REAL $X(M), Q(N), \text{RES}(N)$
CALL RESIDUE($X, Q, \text{RES}$)

Differentiate $\text{RES}$ wrt $Q$

tapenade -backward -vars $Q$ -outvars $\text{RES}$ residue.f

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

REAL $X(M), Q(N), \text{RES}(N)$
REAL $QB(N), \text{RESB}(N)$
CALL RESIDUE_B($X, Q, QB, \text{RES}, \text{RESB}$)

\[
\Psi \\
\downarrow \\
\downarrow \\
\downarrow \\
\text{SUBROUTINE RESIDUE_B($X, Q, QB, \text{RES}, \text{RESB}$)} \\
\left( \frac{\partial R}{\partial Q} \right)^t \Psi
\]
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.

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

$x \in \mathbb{R}^n, \quad T_0 = x \quad t_r \in \mathbb{R} \\
T_r = \begin{bmatrix} T_{r-1} \\ t_r \end{bmatrix}$

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](http://www.autodiff.org)

<table>
<thead>
<tr>
<th>Tool</th>
<th>Modes</th>
<th>Method</th>
<th>Lang</th>
</tr>
</thead>
<tbody>
<tr>
<td>ADIFOR</td>
<td>F</td>
<td>ST</td>
<td>Fortran</td>
</tr>
<tr>
<td>ADOLC</td>
<td>F/B</td>
<td>OO</td>
<td>C</td>
</tr>
<tr>
<td>TAPENADE</td>
<td>F/B</td>
<td>ST</td>
<td>Fortran/C</td>
</tr>
<tr>
<td>TAF/TAMC</td>
<td>F/B</td>
<td>ST</td>
<td>Fortran/C</td>
</tr>
<tr>
<td>MAD</td>
<td>F</td>
<td>OO</td>
<td>Matlab</td>
</tr>
</tbody>
</table>
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

\[ h(x) \]
Quasi 1-D flow

- Quasi 1-D flow in a duct

\[ \frac{\partial}{\partial t}(hU) + \frac{\partial}{\partial x}(hf) = \frac{dh}{dx}P, \quad x \in (a, b) \quad t > 0 \]

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

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

- Inverse design: find shape \( h \) to get pressure distribution \( p^* \)

- Optimization problem: find the shape \( h \) which minimizes

\[ J = \int_a^b (p - p^*)^2 dx \]
Quasi 1-D flow

Inflow face

Outflow face

\[ i - \frac{1}{2} \to i + \frac{1}{2} \]
Quasi 1-D flow

• Finite volume scheme

\[ h_i \frac{dU_i}{dt} + \frac{h_{i+1/2} F_{i+1/2} - h_{i-1/2} F_{i-1/2}}{\Delta x} = \frac{(h_{i+1/2} - h_{i-1/2})}{\Delta x} P_i \]

• 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

The diagram illustrates the comparison between the target shape and the current shape of a duct. The x-axis represents the position along the duct, while the y-axis represents the height (h). The dashed line represents the target shape, and the solid line represents the current shape. The current shape closely follows the target shape, indicating effective optimization.
Target pressure distribution $p^*$

![Graph showing target pressure distribution with different markers for AUSMDV, KFVS, and LF]
Current pressure distribution

Starting pressure

- AUSMDV
- KFVS
- LF
Adjoint density

**Shape Optimization**
Convergence history: Explicit Euler

Convergence history with AUSMDV flux

- Plot shows the convergence history with AUSMDV flux.
- The x-axis represents the number of iterations.
- The y-axis represents the residue.
- Two lines are plotted: one for the Flow and another for the Adjoint.
- The residue decreases significantly with increasing iterations.
Shape gradient

Gradient

AUSMDV
KFVS
LF
Validation of Shape gradient

Gradient with AUSMDV flux

---

Praveen. C (TIFR-CAM)
Validation of shape gradient

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

<table>
<thead>
<tr>
<th>$\Delta h$</th>
<th>A</th>
<th>B</th>
<th>C</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.01</td>
<td>0.4191069499</td>
<td>35.18452823</td>
<td>2.545316345</td>
</tr>
<tr>
<td>0.001</td>
<td>0.4231223499</td>
<td>36.10982621</td>
<td>2.556461900</td>
</tr>
<tr>
<td>0.00001</td>
<td>0.4231624999</td>
<td>36.11933154</td>
<td>2.556573499</td>
</tr>
<tr>
<td>0.000001</td>
<td>0.4231599998</td>
<td>36.11942125</td>
<td>2.556575000</td>
</tr>
<tr>
<td>0.0000001</td>
<td>0.4231999994</td>
<td>36.11942305</td>
<td>2.556550001</td>
</tr>
<tr>
<td>0.00000001</td>
<td>0.4229999817</td>
<td>36.11942329</td>
<td>2.556499971</td>
</tr>
<tr>
<td>AD</td>
<td>0.4231628330</td>
<td>36.11941951</td>
<td>2.556574450</td>
</tr>
</tbody>
</table>
Gradient smoothing

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

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

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

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

- Finite difference with Jacobi iterations
Gradient smoothing

Gradient using AUSMDV flux

- - - Original gradient
- - - Smoothed gradient
Quasi 1-D optimization: Shape
Quasi 1-D optimization: Final shape
Quasi 1-D optimization: Pressure
Quasi 1-D optimization: Convergence

![Graph showing the convergence of cost and gradient over iterations. The graph plots the cost and gradient against the number of iterations, with both values on a logarithmic scale. The cost consistently decreases, while the gradient shows fluctuations before stabilizing.]
Quasi 1-D optimization: Adjoint density

![Graph showing Adjoint density over the range of x from 0 to 10, with Initial and Final markers.](image)
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 \( \frac{dX_s}{d\beta} \) 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 \( \frac{dX}{dX_s} \) can be computed
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
  \[
  \dot{R} := R'(Q)\dot{Q} \\
  \approx \frac{R(Q + \epsilon\dot{Q}) - R(Q)}{\epsilon} \\
  \bar{Q} := [R'(Q)]^\top \tilde{R} \\
  \dot{R}^\top \tilde{R} \ ? \dot{Q}^\top \bar{Q}
  \]

- 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

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

\[ \epsilon = 10^{-8} \]

\[ \epsilon = 10^{-4} \]

Adjoint iterations blow-up

No blow-up
Iterative convergence tests

**Primal residual**

\[ R^n = \| R(Q^n) \| \]

**Adjoint residual**

\[ R^n = \| [R'(Q^\infty)]^\top \Psi^n + \mathcal{J}'(Q^\infty) \| \]

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** \( M_\infty = 0.8, \alpha = 1.25^\circ \)

\[
\mathcal{J} = \frac{C_d}{C_l} \frac{C_{l_0}}{C_{d_0}}
\]

**RAE2822** \( M_\infty = 0.729, \alpha = 2.31^\circ \)

- Penalty approach
  \[
  \mathcal{J} = \frac{C_d}{C_{d_0}} + \omega \left| 1 - \frac{C_l}{C_{l_0}} \right|
  \]
  - Constrained minimization

\[
\min \mathcal{J} = \frac{C_d}{C_{d_0}} \quad s.t. \quad C_l = C_{l_0}
\]
NACA0012: Maximize L/D

![Graph showing Cp vs x/c for different methods]

<table>
<thead>
<tr>
<th>Method</th>
<th>$I$</th>
<th>$100C_d$</th>
<th>$C_l$</th>
<th>$N_{fun}$</th>
<th>$N_{grad}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>Initial</td>
<td>1.000</td>
<td>2.072</td>
<td>0.295</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>conmin_frcg</td>
<td>0.170</td>
<td>0.389</td>
<td>0.325</td>
<td>50</td>
<td>13</td>
</tr>
<tr>
<td>optpp_q_newton</td>
<td>0.142</td>
<td>0.329</td>
<td>0.329</td>
<td>50</td>
<td>39</td>
</tr>
<tr>
<td>steep</td>
<td>0.166</td>
<td>0.335</td>
<td>0.287</td>
<td>50</td>
<td>45</td>
</tr>
</tbody>
</table>
RAE2822: Drag minimization, penalty approach

Method | \( \mathcal{J} \) | 100\( C_d \) | \( C_l \) | \( N_{\text{fun}} \) | \( N_{\text{grad}} \)  
------- | -------- | -------- | -------- | -------- | --------  
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  

Praveen. C (TIFR-CAM)  
Shape Optimization  
IISc, Dec 2009
RAE2822: Lift-constrained drag minimization

![Graphs showing lift-constrained drag minimization results](image)

<table>
<thead>
<tr>
<th>Method</th>
<th>$J$</th>
<th>$100C_d$</th>
<th>$C_l$</th>
<th>$N_{\text{fun}}$</th>
<th>$N_{\text{grad}}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>Initial</td>
<td>1.000</td>
<td>1.150</td>
<td>0.887</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>conmin_mfd</td>
<td>0.342</td>
<td>0.394</td>
<td>0.881</td>
<td>50</td>
<td>22</td>
</tr>
<tr>
<td>fsqp</td>
<td>0.325</td>
<td>0.374</td>
<td>0.887</td>
<td>50</td>
<td>32</td>
</tr>
<tr>
<td>ipopt</td>
<td>0.335</td>
<td>0.385</td>
<td>0.887</td>
<td>45</td>
<td>62</td>
</tr>
</tbody>
</table>
RAE2822: Lift-constrained drag minimization
RANS computation

M=0.729, Re=6.5 million, Cl=0.88, k-w turbulence model

<table>
<thead>
<tr>
<th>Shape</th>
<th>Inviscid</th>
<th>RANS</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>AOA</td>
<td>$C_l$</td>
</tr>
<tr>
<td>RAE2822</td>
<td>2.31</td>
<td>0.887</td>
</tr>
<tr>
<td>Optimized</td>
<td>2.31</td>
<td>0.887</td>
</tr>
</tbody>
</table>
Sensitivity to perturbations

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

Need for robust aerodynamic optimization
Consider the functional

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

where $u$ is the solution of the conservation law

$$ u_t + f(u)_x = 0, \quad x \in (0, 1), \quad t \in (0, T) $$

$$ u(x, 0) = u_0(x) $$

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

$$ \tilde{J} = \int g(u(x, T))\tilde{u}dx, \quad g(u) = \frac{dG}{du} $$

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

$$ (\tilde{u})_t + (f'(u)\tilde{u})_x = 0 $$

$$ \tilde{u}(x, 0) = \tilde{u}_0 $$
To write $\tilde{J}$ in terms of $\tilde{u}_0$, we introduce the adjoint variable $v(x, t)$

$$\tilde{J} = \int g(u(x, T))\tilde{u}(x, T)dx - \int_0^T \int v [(\tilde{u})_t + (f'(u)\tilde{u})_x] \, dx\,dt$$

Integrating by parts, we get

$$\tilde{J} = \int v(x, 0)\tilde{u}_0(x)\,dx + \int (g - v)\tilde{u}(x, T)\,dx$$
$$- \int_0^T \int [v_t + f'(u)v_x] \tilde{u}(x, t)\,dx\,dt$$

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

$$v_t + f'(u)v_x = 0$$
$$v(x, T) = g(u(x, T))$$

The derivative is given by

$$\tilde{J} = \int v(x, 0)\tilde{u}_0(x)\,dx$$
Note that the adjoint equation must be solved backward in time, starting from $t = T$ to $t = 0$.

**Example: Inviscid Burger’s equation**

$$u_t + \left(\frac{u^2}{2}\right)_x = 0, \quad x \in \mathbb{R}, \quad t \in (0, T)$$

$$u(x, 0) = \begin{cases} u_l & x < 0 \\ u_r & x > 0 \end{cases}$$

Exact solution:

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

Consider the objective functional

$$J = \frac{1}{2} \int u^2(x, T)dx$$
Then the corresponding adjoint equation is

\[ v_t + uv_x = 0 \]

\[ v(x, T) = u(x, T) = \begin{cases} 
    u_l & x < ST \\
    u_r & x > ST 
\end{cases} \]

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 $S = \frac{dx_s}{dt}$ is the shock speed.

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

$$x_s \rightarrow x_s + \tilde{x}_s, \quad \frac{dx_s}{dt} \rightarrow \frac{dx_s}{dt} + \frac{d\tilde{x}_s}{dt}$$

$$\frac{d\tilde{x}_s}{dt} [u] + \frac{dx_s}{dt} [\tilde{u}] + \frac{dx_s}{dt} \tilde{x}_s \left[ \frac{\partial u}{\partial x} \right] = \left[ \frac{df}{du} \tilde{u} \right] + \tilde{x}_s \left[ \frac{df}{du} \frac{\partial u}{\partial x} \right]$$
\[ J = \int_{-1}^{x_s(T)} G(u(x, T))\,dx + \int_{x_s(T)}^{+1} G(u(x, T))\,dx \]

Perturbation in \( J \) is

\[ \tilde{J} = \int_{-1}^{x_s(T)} g(u(x, T))\tilde{u}(x, T)\,dx + \int_{x_s(T)}^{+1} g(u(x, T))\tilde{u}(x, T)\,dx - [G]_{x_s(T)} \tilde{x}_s(T) \]

\[ - \int_{-1}^{T} \int_{x_s(t)}^{T} \nu \left[ \tilde{u}_t + (f'(u)\tilde{u})_x \right] \,dx\,dt - \int_{x_s(t)}^{T} \int_{-1}^{+1} \nu \left[ \tilde{u}_t + (f'(u)\tilde{u})_x \right] \,dx\,dt \]

\[ - \int_{0}^{T} \nu_s(t) \left\{ \frac{d\tilde{x}_s}{dt}[u] + \frac{dx_s}{dt}[\tilde{u}] + \frac{dx_s}{dt} \tilde{x}_s \left[ \frac{\partial u}{\partial x} \right] - \left[ \frac{df}{du}\tilde{u} \right] - \tilde{x}_s \left[ \frac{df}{du} \frac{\partial u}{\partial x} \right] \right\} \,dt \]
After some work we get

\[ \tilde{J} = \int_{-1}^{+1} v(x,0) \tilde{u}_0(x) dx \]

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

\[
\begin{align*}
    v_t + f'(u) v_x &= 0 \\
    v(x,T) &= g(u(x,T)) \\
    v_s(t) &= v(x_t^+(t), t) = v(x_t^-(t), t) \\
    \frac{d}{dt} v_s(t) &= 0 \\
    v_s(T) &= \frac{[G]}{[u]}
\end{align*}
\]

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 $n$

$$P^n = \{x^n_1, x^n_2, \ldots, x^n_M\} \subset \mathbb{R}^d$$

- Solutions evolve according to some rules

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

- 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

- 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

\[ x_i \in D, \quad i = 1, \ldots, N_p \]
Particle swarm optimization

Each particle has a velocity

\[ v_i \in \mathbb{R}^d, \quad i = 1, \ldots, N_p \]
Particle swarm optimization

- Particles have memory ($t =$ iteration number)

  **Local memory** : $p_i^t = \arg\min_{0 \leq s \leq t} J(x_i^s)$

  **Global memory** : $p_t = \arg\min_i J(p_i^t)$

- Velocity update

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

- Position update

  $$x_i^{t+1} = x_i^t + v_i^{t+1}$$
Basic PSO algorithm

$t=0$

Initialize position, velocity

Compute cost function

Update local and global memory

Update velocity and position

Convergence?

$t=t+1$

No

Yes

Stop
PSO: Parallelizability

\[
\begin{align*}
\begin{array}{cccc}
  x_1^n & x_2^n & \ldots & x_{Np}^n \\
  \downarrow & \downarrow & \ldots & \downarrow \\
  J(x_1^n) & J(x_2^n) & \ldots & J(x_{Np}^n) \\
  \downarrow & \downarrow & \ldots & \downarrow \\
  v_1^{n+1} & v_2^{n+1} & \ldots & v_{Np}^{n+1} \\
  \downarrow & \downarrow & \ldots & \downarrow \\
  x_1^{n+1} & x_2^{n+1} & \ldots & x_{Np}^{n+1}
\end{array}
\end{align*}
\]

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

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

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

- Bernstein polynomials

$$B^n_p(t) = C^n_{p} t^p (1 - t)^{n-p}$$

- Design variables

$$\{Y_{ijk}\}, \quad 0 \leq i \leq n_i, \quad 0 \leq j \leq n_j, \quad 0 \leq k \leq n_k$$
Test case: Wing shape optimization

- 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

\( M_\infty = 0.83, \ \alpha = 2^\circ \)

(Piaggio Aero. Ind.)

Grid: 31124 nodes

Cost function

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

\(^3\) Joint work with R. Duvigneau, OPALE, INRIA
Test case: Wing shape optimization

- 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

\[ M_\infty = 0.83, \alpha = 2^\circ \]

(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)

\[ (10 \text{ minutes/CFD}) \times (120 \text{ CFD/iteration}) \times (200 \text{ iterations}) = 4000 \text{ hours} \]
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

![Graph showing effect of Mach number fluctuations on drag coefficient and relative drag reduction.](image)
Optimization problem

Optimization

\[
\min_{x \in \mathbb{R}^d} J(x, a_o)
\]

\[C(x, a_o) \leq 0\]

Robust optimization

\[
\mu_J(x) = \int_{\Omega(A)} J(x, a) \rho_A(a) \, da
\]

\[
\sigma_J^2(x) = \int_{\Omega(A)} [J(x, a) - \mu_J]^2 \rho_A(a) \, da
\]

\[
\text{Prob}[C(x, A) \leq 0] \geq p
\]
Optimization problem

Optimization

\[
\min_{x \in \mathbb{R}^d} J(x, a_0)
\]
\[C(x, a_0) \leq 0\]

Robust optimization

\[
\min_{x \in \mathbb{R}^d} \left\{ \begin{array}{c}
\mu_J(x) = \int_{\Omega(A)} J(x, a) \rho_A(a) \, da \\
\sigma_J^2(x) = \int_{\Omega(A)} [J(x, a) - \mu_J]^2 \rho_A(a) \, da
\end{array} \right. 
\]
\[\text{Prob}[C(x, A) \leq 0] \geq p\]
Monte-Carlo estimation using meta-models

- For each design $x$: compute
  \[
  J(x, a_1), J(x, a_2), \ldots, J(x, a_P)
  \]
  \[
  \downarrow
  \]
  meta-model $\tilde{J}_x(a)$

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

  \[
  M_J(x) = \frac{1}{N_s} \sum_{i=1}^{N_s} \tilde{J}_x(a_i)
  \]

  \[
  S_J^2(x) = \frac{1}{N_s - 1} \sum_{i=1}^{N_s} [\tilde{J}_x(a_i) - M_J]^2
  \]

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

\[
\min_{x \in \mathbb{R}^d} \omega M_J + (1 - \omega) S_J + \text{constraint penalty}
\]
Monte-Carlo estimation using meta-models

- For each design $x$: compute

  \[ J(x, a_1), J(x, a_2), \ldots, J(x, a_P) \]

  \[ \downarrow \]

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

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

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

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

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

\[ M_\infty = 0.83, \; \alpha = 2^\circ \]

(Piaggio Aero. Ind.)
Grid: 31124 nodes

Number of CFD computations/iteration = 32 \( \times 4 = 128 \)
Mean = 0.83, 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)