Two (Quasi-)Differentiable OSD Methods

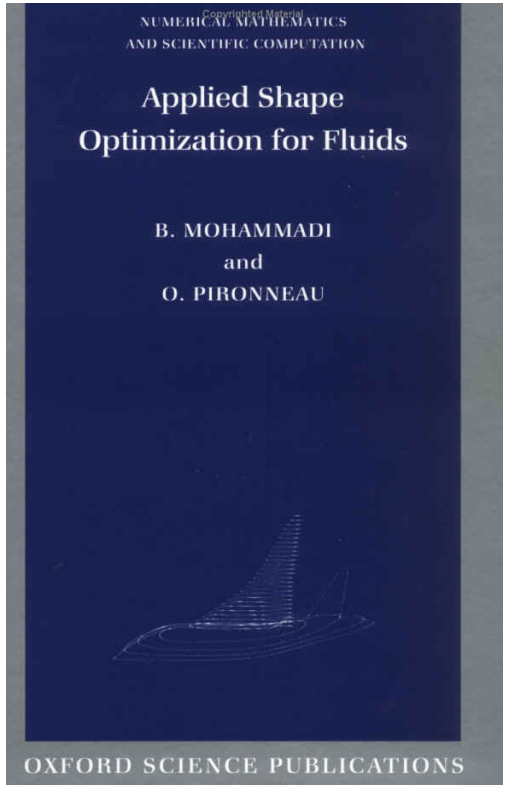
http://www.ann.jussieu.fr/pironneau

Olivier Pironneau[1]

[1]University of Paris VI, Laboratoire J.-L. Lions, **Olivier.Pironneau@upmc.fr**

MDO-SCOMA course, March 09

Outline



More details in

B. Mohammadi & O.P. *Applied Optimal Shape Design*, Oxford U. Press (2001). Second edition 2009.

O.P. *Optimal shape design for elliptic systems*. Springer, (1984)

The Topic of Today

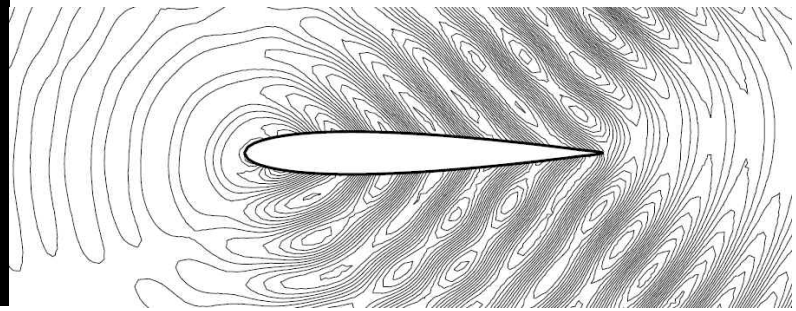
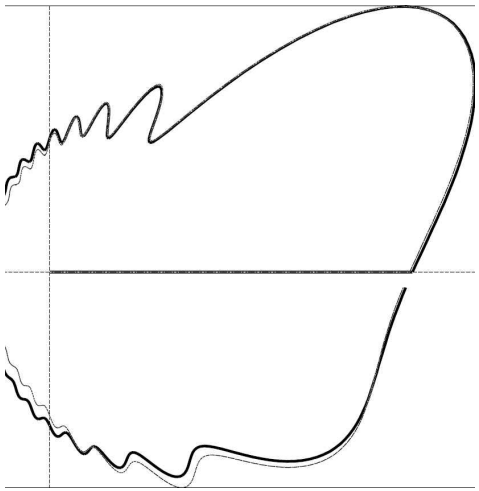
Present the continuous case.

Books

- **J. Haslinger and R. A. E. Makinen** Introduction to Shape Optimization: Theory, Approximation, and Computation.SIAM series 2003.
- **Jasbir S. Arora**: Introduction to Optimum Design. Elsevier 2004
- **E. Laporte, P. Letallec**: Numerical Methods in Sensitivity Analysis and Shape Optimization. Birkhauser, 2003.
- **M. Bendsoe and O. Sigmund.** *Topology Optimization. Theory, Methods, and Applications*. Springer 2003.
- **M. Delfour, J.P. Zolezio**: shape and geometries SIAM 2001.
- **G. Allaire**: *Shape Optimization by Homogenization* Springer 2001.
- **A. Cherkaev.** *Variational Methods for Structural Opt.* Springer 2000.
- **G.W. Litvinov.** *Optimization in elliptic pb with appli. to mech. of structures and fluid mech.*, vol 119 Operator Theory. Birkhauser, 2000.
- **J. Haslinger, P. Neittanmakki**: Finite Element Approximation for Optimal Shape. J. Wiley 1996.
- **M. Bendsoe** *Methods for optimization of structural topology, shape and material*. Springer 1995.

Important Applications

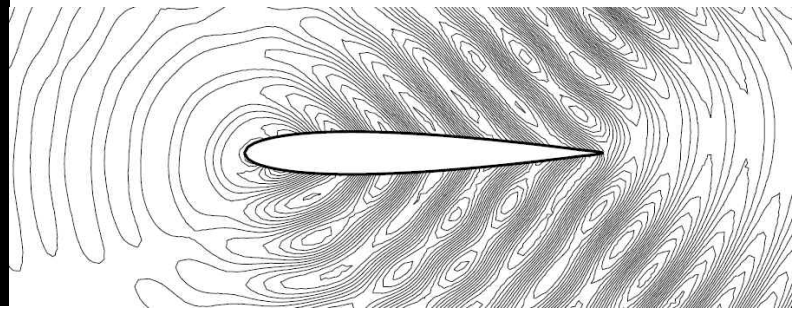
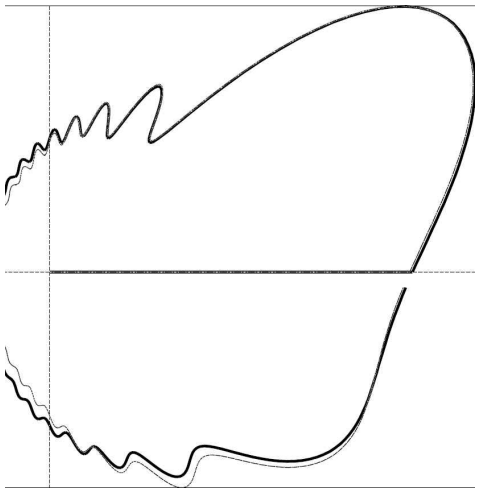
- Aerodynamics: Shape optimization to improve airplanes, cars, ventilators, turbines...
- Hydrodynamics: wave drag of boats, pipes, by-pass, harbors...

- Electromagnetics: Stealth airplane, antenna, missiles...
- Combustion: Car and airplane engines, scramjets...
- Turbulence: delay the separation of boundary layers, reduce turbulent drag (active control, deformable airplane...)

Important Applications

- Aerodynamics: Shape optimization to improve airplanes, cars, ventilators, turbines...
- Hydrodynamics: wave drag of boats, pipes, by-pass, harbors...

- Electromagnetics: Stealth airplane, antenna, missiles...
- Combustion: Car and airplane engines, scramjets...
- Turbulence: delay the separation of boundary layers, reduce turbulent drag (active control, deformable airplane...)

Main Topics for Shape Optimization

$$\min_{v \in V \subset \mathcal{R}^d} E(v)$$

- Black Box Optimization: use only $v \rightarrow E(v)$
- Differentiable Optimization: use also $\text{grad}_v E(v)$

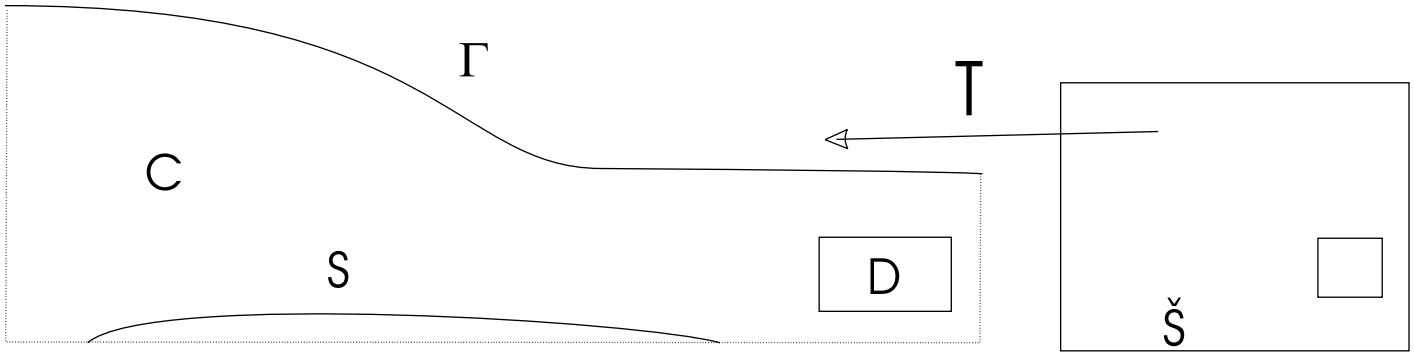
$$E(v+\delta v) = E(v) + < \text{grad}E(v), \delta v > + o(\|\delta v\|)$$
$$\delta v = -\rho \text{grad}E(v) \;\Rightarrow\; E(v+\delta v) - E(v) \approx -\rho \|\text{grad}E(v)\|^2$$

- Constrained Optimization: $V = \{v \in H \;:\; f(v) = 0, \;\; g(v) \leq 0\}$
- Multi-criteria and Pareto optimality:

$$E(v) = \sum_i \alpha_i E_i(v) \;\Leftrightarrow\; ? \;\nexists w :\; E_i(w) \leq E_i(v) \;\forall i$$

- Topological Optimization: Embed the problem into a larger class

An Academic Problem



$$\min_{\mathbf{s}\in\mathcal{S}_d}\{\int_D |\psi-\psi_d|^2 \ : \ -\Delta\psi=0, \text{ in } C-\dot{S}, \quad \psi|_{\mathbf{S}}=0 \ \ \psi|_{\partial C}=\psi_d\}$$

Wind tunnel Design by adapting S so that flow is uniform in D. Flow is irrotational inviscid and 2D.

Existence of Solution

Theorem

$$\min_{v \in V \subset \mathcal{R}^d} E(v)$$

has a solution if V is closed, E is bounded from below, l.s.c. and either V is bounded or $\lim_{||x|| \to \infty} E(x) = +\infty$

Thus if one can show that the criteria of the OSD problem is l.s.c. a solution will exist. It has been shown by Sverak that this is so if the number of connected component of Ω is bounded.

In Allaire, Bucur, Delfour et al, it is shown that a penalization of the perimeter of the unknown surface also induce existence in 2D.

Theorem The following problem has at least one solution:

$$\min_{\mathbf{s} \in \mathcal{S}_d} \{ \int_D |\psi - \psi_d|^2 + \epsilon |S|^2 \ : \ -\Delta \psi = 0, \text{ in } C - \dot{S}, \quad \psi|_{\mathbf{S}} = 0 \ \ \psi|_{\partial C} = \psi_d \}$$

Uniqueness is almost impossible to prove;

Existence of Solution

Theorem

$$\min_{v \in V \subset \mathcal{R}^d} E(v)$$

has a solution if V is closed, E is bounded from below, l.s.c. and either V is bounded or $\lim_{||x|| \to \infty} E(x) = +\infty$

Thus if one can show that the criteria of the OSD problem is l.s.c. a solution will exist. It has been shown by Sverak that this is so if the number of connected component of Ω is bounded.

In Allaire, Bucur, Delfour et al, it is shown that a penalization of the perimeter of the unknown surface also induce existence in 2D.

Theorem The following problem has at least one solution:

$$\min_{\mathbf{s} \in \mathcal{S}_d} \{ \int_D |\psi - \psi_d|^2 + \epsilon |S|^2 \ : \ -\Delta \psi = 0, \text{ in } C - \dot{S}, \quad \psi|_{\mathbf{S}} = 0 \ \ \psi|_{\partial C} = \psi_d \}$$

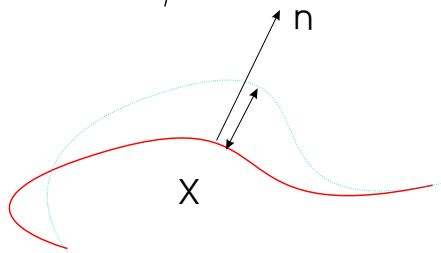
Uniqueness is almost impossible to prove;

Sensitivity Analysis

$$-\Delta\psi^\epsilon = f \quad \text{in } \Omega^\epsilon \qquad \psi^\epsilon = 0 \text{ on } \Gamma^\epsilon := \{x + \epsilon\alpha n \ : \ x \in \Gamma\}$$

Definition *If* $\psi'_\alpha := \lim \frac{1}{\epsilon}(\psi^\epsilon - \psi)$ *exists then* ψ *is Gateau differentiable with respect to* Γ *in the direction* α*. If* ψ'_α *is linear in* α *then* ψ *is Frechet differentiable.* Similarly

$$\psi^{\epsilon\alpha} = \psi + \epsilon\psi'_\alpha + \frac{\epsilon^2}{2}\psi''_\alpha$$



To compute ψ' and ψ'' notice that, by linearity, they satisfy the same PDE but with $f = 0$. By Taylor expansion, $x \in \Gamma$:

$$0 = \psi^{\epsilon\alpha}(x + \epsilon\alpha n) = \psi^{\epsilon\alpha}(x) + \epsilon\alpha\frac{\partial\psi^{\epsilon\alpha}}{\partial n}(x) + \frac{\epsilon^2\alpha^2}{2}\frac{\partial^2\psi}{\partial n^2}(x) + \ldots$$

Therefore

$$-\Delta\psi'_\alpha = 0 \quad \psi'_\alpha|_\Gamma = -\alpha\frac{\partial\psi}{\partial n}, \qquad -\Delta\psi''_\alpha = 0 \quad \psi''_\alpha|_\Gamma = -\alpha\frac{\partial\psi'_\alpha}{\partial n} - \frac{\alpha^2}{2}\frac{\partial^2\psi}{\partial n^2}$$

Optimality Conditions

Consider the Wind Tunnel Problem with $S^\epsilon = \{x + \epsilon\alpha n : \ x \in S\}$. Think of the PDE as the implicit definition of $S \to \psi(S)$. Then $J()$ is a function of S only:

$$J(S^\epsilon) = \int_D |\psi^\epsilon - \psi_d|^2 = \int_D |\psi - \psi_d|^2 + 2\epsilon \int_D (\psi^\epsilon - \psi_d)\psi'_\alpha + o(\epsilon)$$

with $\Delta\psi'_\alpha = 0, \ \ \psi'_\alpha|_S = -\alpha\frac{\partial\psi}{\partial n}, \ \ \psi'_\alpha|_{\Gamma - S} = 0.$
If J is Frechet differentiable there exists ξ such that $J'_\alpha = \int_S \xi\alpha$. To find ξ we must use the adjoint trick and introduce

$$-\Delta p = (\psi^\epsilon - \psi_d)I_D, \ \ p|_\Gamma = 0$$

Then

$$2\int_D (\psi^\epsilon - \psi_d)\psi'_\alpha = -2\int_\Omega \psi'_\alpha \Delta p = -2\int_\Omega \Delta\psi'_\alpha p + \int_\Gamma (\frac{\partial p}{\partial n}\psi'_\alpha + \frac{\partial\psi'_\alpha}{\partial n}p)$$

Corollary $\quad J'_\alpha = 2\int_S \frac{\partial p}{\partial n}\frac{\partial\psi}{\partial n}\alpha$

Conceptual Algorithm

- 0. Choose a shape S^0, a small number $\rho > 0$ and set m=0.
- 1. Compute ψ^m and p^m by solving

$$-\Delta\psi^m = 0, \quad \psi^m|_{S^m} = 0, \ \psi^m|_{\Gamma_d} = \psi_d$$
$$-\Delta p^m = (\psi^m - \psi_d) I_D, \quad p|_{\Gamma^m} = 0$$

- 2. Set

$$\alpha = -\rho \frac{\partial p^m}{\partial n} \frac{\partial \psi^m}{\partial n} \qquad \mathcal{S}^{m+1} = \{x + \alpha n : \ x \in S^m\}$$

- 3. Set $m \leftarrow m + 1$ and go to 1.

It works because

$$J(S^{m+1}) = J(S^m) + \int_{S^m} \xi\alpha = J(S^m) - 2\rho \int_{S^m} (\frac{\partial p^m}{\partial n} \frac{\partial \psi^m}{\partial n})^2 + o(\alpha)$$

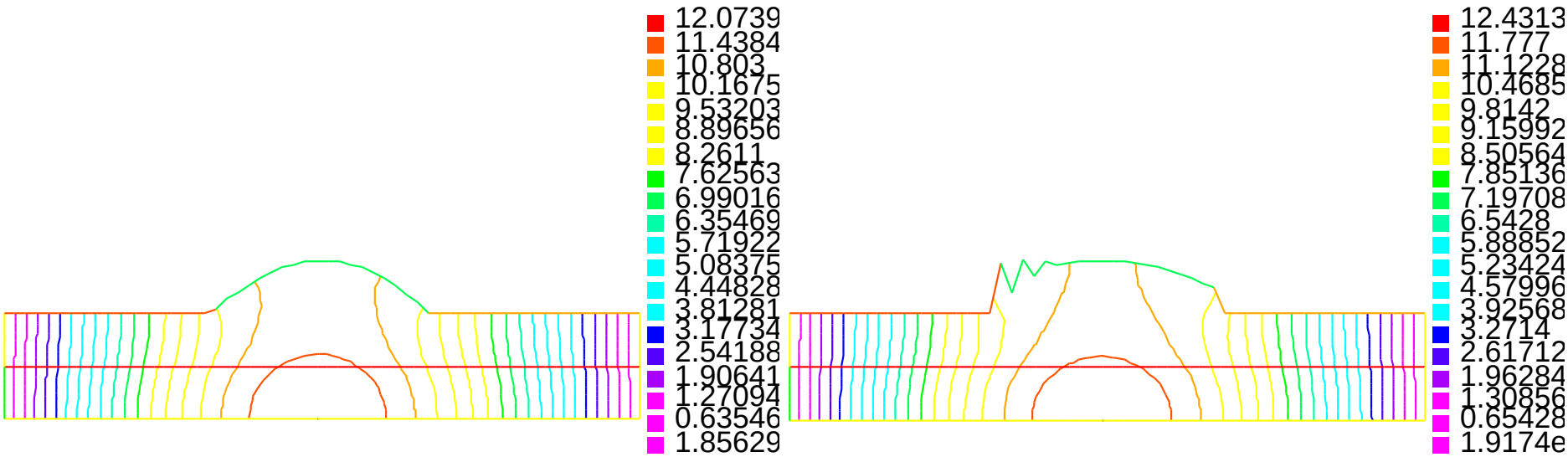
Notice that there is a loss of regularity from S^m to S^{m+1}!

Implementation with freefem++

```
real xl = 5, L=0.3;
mesh th = square(30,30,[x,y*(0.2+x/xl)]);
func D=(x>0.4+L && x<0.6+L)*(y<0.1);
func psid = 0.8*y;
fespace Vh(th,P1);
Vh psi,p,w;
problem streamf(psi,w)=int2d(th)(dx(psi)*dx(w) + dy(psi)*dy(w))
   +on(1,4,psi = y/0.2)+on(2,psi=y/(0.2+1.0/xl)) + on(3,psi=1);

problem adjoint(p,w)=int2d(th)(dx(p)*dx(w) + dy(p)*dy(w))
   - int2d(th)(D*(psi-psid)*w)+ on(1,2,3,4,p=0);
Vh a=0.2+x/xl, gradE;
for(int i=0;i<100;i++){
      streamf; adjoint;
      real E = int2d(th)(D*(psi-psid)^2)/2;
      gradE = dx(psi)*dx(p) + dy(psi)*dy(p);
      a=a(x,0)-50*gradE(x,a(x,0))*x*(1-x);
      th = square(30,30,[x,y*a(x,0)]);
}
```

Execute

Oscillations



Regularity Preserving Algorithms: Sobolev Gradients

$$\alpha = -\rho \frac{\partial p}{\partial n}\frac{\partial \psi}{\partial n} \qquad \mathcal{S}^{m+1} = \{x + \alpha n : \ x \in \mathcal{S}^m\}$$

can be replaced by

$$\frac{d^2\tilde{\alpha}}{ds^2} = \rho \frac{\partial p}{\partial n}\frac{\partial \psi}{\partial n} \ \ \tilde{\alpha}(s_0) = \tilde{\alpha}(s_1) = 0. \qquad \mathcal{S}^{m+1} = \{x + \tilde{\alpha} n : \ x \in \mathcal{S}^m\}$$

$$\Rightarrow \ \ J(\mathcal{S}^{m+1}) - J(\mathcal{S}^m) = \frac{2}{\rho}\int_{\mathcal{S}^m} \tilde{\alpha}\frac{d^2\tilde{\alpha}}{ds^2} = -\frac{2}{\rho}\int_{\mathcal{S}^m} (\frac{d\tilde{\alpha}}{ds})^2 + o(\rho)$$

Alternatively one may use a smoothing operator like

$$\beta \to \gamma(\beta) = v \text{ where } v \text{ is solution of } \ -\Delta v = 0 \quad \frac{\partial v}{\partial n}|_\Gamma = \beta.$$

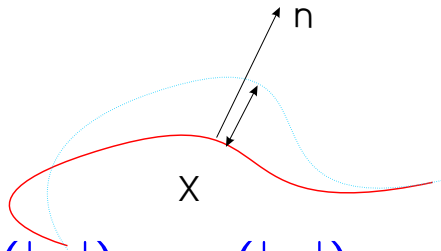
Let $\mathcal{S}^{m+1} = \{x + \gamma(\beta) n : \ x \in \mathcal{S}^m\}$ with $\beta = \frac{\partial p}{\partial n}\frac{\partial \psi}{\partial n}$

$$J(\mathcal{S}^{m+1}) - J(\mathcal{S}^m) = 2\rho\int_\Gamma \gamma(\beta)\beta = 2\rho\int_\Gamma v\frac{\partial v}{\partial n} = -2\rho\int_\Omega |\nabla v|^2$$

Geometric Constraints

- Projected Gradient: the case $\int_\Omega = 1$.

$$\Gamma' = \{x + \alpha n(x) : \ x \in \Gamma\} \Rightarrow \ \ \delta J = \int_\Gamma \chi \alpha ds + o(|\alpha|)$$



$$\Gamma' = \{x + (\alpha - \frac{1}{|\Gamma|}\int_\Gamma \alpha) n(x) : \ x \in \Gamma\}$$

$$\Rightarrow \ \ \delta \int_\Omega = \int_\Gamma (\alpha - \frac{1}{|\Gamma|}\int_\Gamma \alpha) + o(|\alpha|) = o(|\alpha|)$$

$$\delta J = \int_\Gamma (\chi - \frac{1}{|\Gamma|}\int_\Gamma \chi)(\alpha - \frac{1}{|\Gamma|}\int_\Gamma \alpha) ds + o(|\alpha|)$$

- Penalization: replace J by

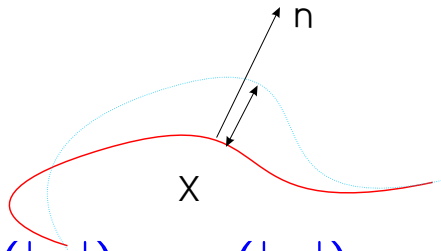
$$J + \frac{1}{\epsilon}|F(\Omega)^+|^2 + \frac{1}{\omega}|G(\Omega)|^2$$

to maintain $F(\Omega \leq 0), \ G(\Omega) = 0$

Geometric Constraints

- Projected Gradient: the case $\int_\Omega = 1$.

$$\Gamma' = \{x + \alpha n(x) : \ x \in \Gamma\} \Rightarrow \ \ \delta J = \int_\Gamma \chi \alpha ds + o(|\alpha|)$$



$$\Gamma' = \{x + (\alpha - \frac{1}{|\Gamma|}\int_\Gamma \alpha) n(x) : \ x \in \Gamma\}$$

$$\Rightarrow \ \ \delta \int_\Omega = \int_\Gamma (\alpha - \frac{1}{|\Gamma|}\int_\Gamma \alpha) + o(|\alpha|) = o(|\alpha|)$$

$$\delta J = \int_\Gamma (\chi - \frac{1}{|\Gamma|}\int_\Gamma \chi)(\alpha - \frac{1}{|\Gamma|}\int_\Gamma \alpha) ds + o(|\alpha|)$$

- Penalization: replace J by

$$J + \frac{1}{\epsilon}|F(\Omega)^+|^2 + \frac{1}{\omega}|G(\Omega)|^2$$

to maintain $F(\Omega \leq 0), \ G(\Omega) = 0$

State Constraints

$$\min_v \{J(u,v) : \ Au = g(v), \ \ F(u,v) \leq 0\}$$

where A is a linear invertible operator.

$\delta J = J'_u \delta u + J'_v \delta v$ with $A\delta u = g'_v \delta v, \ \ F'_u \delta u + F'_v \delta v \leq 0$ if $F(u,v) = 0$

Introducing $A^T p = J'_u, \ \ A^T q = F'_u$ leads to

$J'_u \delta u = \delta u \cdot A^T p = p \cdot A\delta u = p \cdot g'_v \delta v \qquad F'_u \delta u = \delta u \cdot A^T p = q \cdot A\delta u = q \cdot g'_v \delta v$

$$\delta J = (p \cdot g'_v + J'_v)\delta v \text{ with } (q \cdot g'_v + F'_v)\delta v \leq 0 \text{ if } F(u,v) = 0$$

A direction of descent is built from this.
Notice that two adjoint vectors are needed

Example

Build a stealth airfoil with "good" aerodynamic properties

$$\min_S J := \int_D |u|^2 \ : \ \int_S \frac{\partial \psi}{\partial n} = a$$

$$\omega^2 u + \Delta u = 0, \text{ in } \Omega \ \ u|_\Gamma = g$$
$$-\Delta \psi = 0, \text{ in } \Omega \ \ \psi|_\Gamma = \psi_d$$

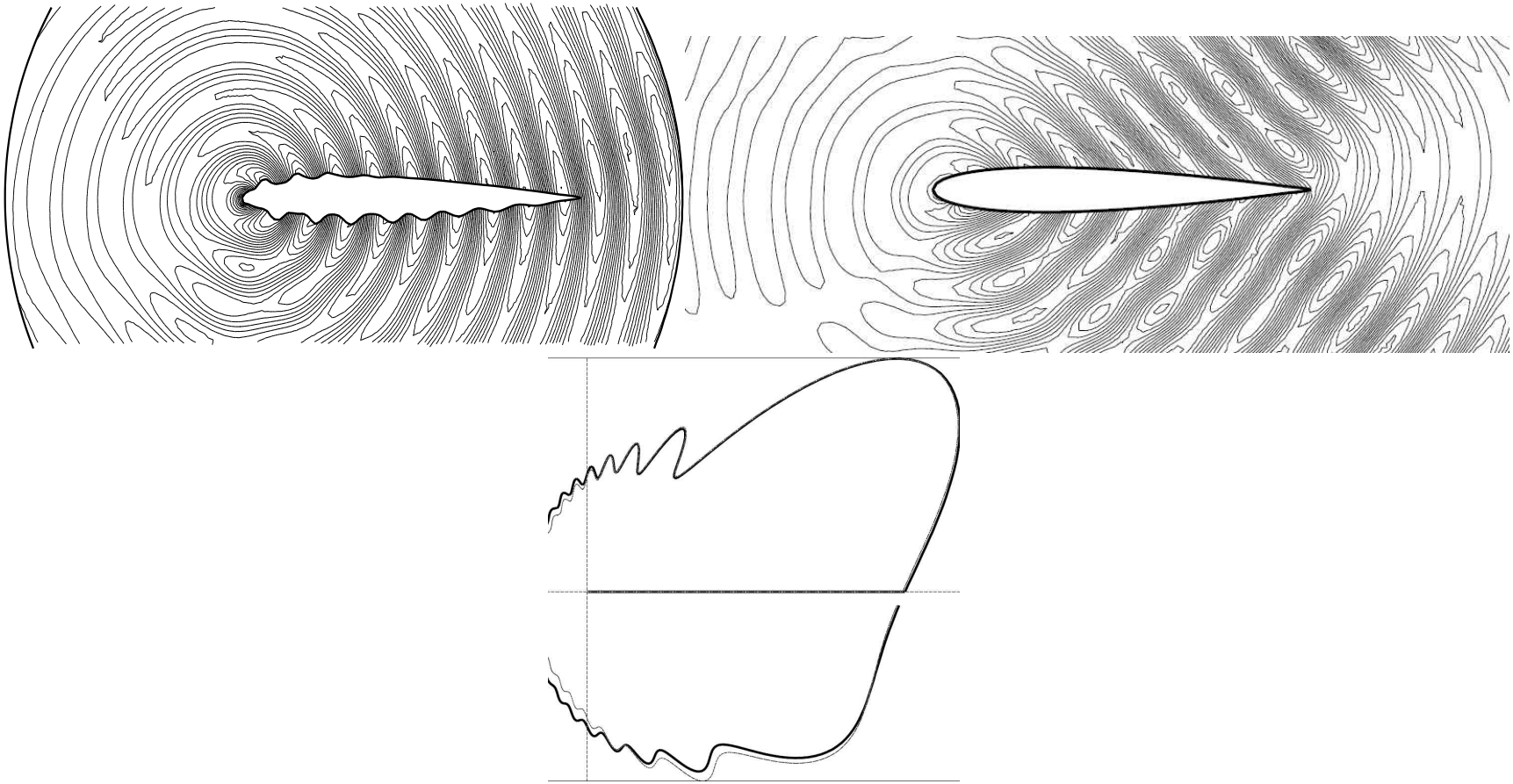
Requires the following

Lemma

$$\Gamma' = \{x + \alpha n : \ x \in \Gamma\} \Rightarrow \delta \int_\Gamma f = \int_\Gamma \alpha (\frac{\partial f}{\partial n} - \frac{f}{R})$$

where R is the mean radius of curvature.

Example



Computed by A. Baron

The Minimum Drag Problem

$$J(\Omega) \equiv \min_{\Omega \in C, vol(C)=1} \int_\Omega \frac{1}{2}||\nabla u||^2 dx \ : \qquad u|_{\partial\Omega} = g$$
$$u\nabla u + \nabla p - \nu\Delta u = 0, \quad \nabla \cdot u = 0,$$

The solution exists in 2D if the nb of connected component is bounded. In 3D ? but probably yes because the criteria is the energy of the system. For safety regularize by adding $\epsilon call(C)$.

Proposition

$$\partial\Omega' = \{x + \alpha n(x) : \ x \in \partial\Omega\} \Rightarrow ? \ \ \delta J = \int_{\partial\Omega} \chi\alpha ds + o(|\alpha|)$$

$$\delta J = \int_{\partial\Omega} \alpha \frac{\partial u}{\partial n} \cdot (\frac{1}{2}\frac{\partial u}{\partial n} + \frac{\partial w}{\partial n}) + o(|\alpha|)$$

where $-u\nabla w + w\nabla u^T + \nabla q - \nu\Delta w = \nu\Delta u, \quad \nabla \cdot w = 0, \ w|_{\partial\Omega} = 0$

From JFM 73. See also, Modi, Gunzberger, Tasan, Jameson... Q? What minimal norm on α?

The Minimum Drag Problem

$$J(\Omega) \equiv \min_{\Omega \in C, vol(C)=1} \int_\Omega \frac{1}{2}||\nabla u||^2 dx \ : \qquad u|_{\partial\Omega} = g$$
$$u\nabla u + \nabla p - \nu\Delta u = 0, \quad \nabla \cdot u = 0,$$

The solution exists in 2D if the nb of connected component is bounded. In 3D ? but probably yes because the criteria is the energy of the system. For safety regularize by adding $\epsilon call(C)$.

Proposition

$$\partial\Omega' = \{x + \alpha n(x) : \ x \in \partial\Omega\} \Rightarrow ? \ \ \delta J = \int_{\partial\Omega} \chi\alpha ds + o(|\alpha|)$$

$$\delta J = \int_{\partial\Omega} \alpha \frac{\partial u}{\partial n} \cdot (\frac{1}{2}\frac{\partial u}{\partial n} + \frac{\partial w}{\partial n}) + o(|\alpha|)$$

where $-u\nabla w + w\nabla u^T + \nabla q - \nu\Delta w = \nu\Delta u, \quad \nabla \cdot w = 0, \ w|_{\partial\Omega} = 0$

From JFM 73. See also, Modi, Gunzberger, Tasan, Jameson... Q? What minimal norm on α?

Proof

Recall that $\delta \int_\Omega f = \int_\Gamma \alpha f$. Then

$$\delta J = \int_\Omega \nabla u \cdot \nabla \delta u + \frac{1}{2} \int_{\partial\Omega} \alpha |\nabla u|^2 + o(|\alpha|)$$

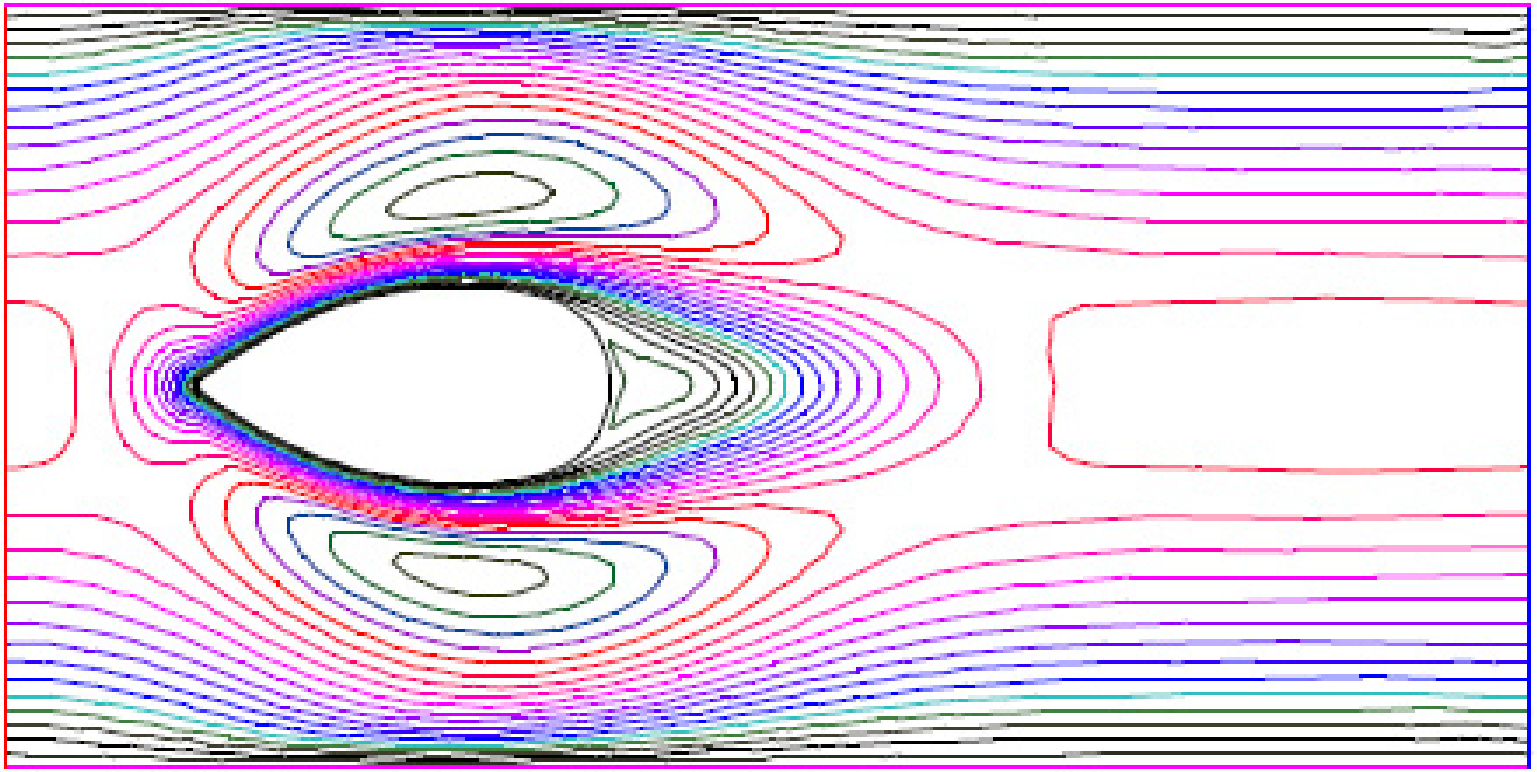
and $\delta u \nabla u + u \nabla \delta u + \nabla \delta p - \nu \Delta \delta u = 0, \quad \nabla \cdot \delta u = 0, \quad \delta u|_\Gamma = -\alpha \frac{\partial u}{\partial n}$

So $\int_\Omega \nabla u \cdot \nabla \delta u = -\int_\Omega \delta u \Delta u$

$$= -\frac{1}{\nu} \int_\Omega (-u \nabla w + w \nabla u^T + \nabla q - \nu \Delta w) \delta u$$

$$= -\frac{1}{\nu} \int_\Omega (\nabla \cdot (u \otimes \delta u + \delta u \otimes u) - \nu \Delta \delta u) w - \int_\Gamma \nu \frac{\partial w}{\partial n} \delta u + o(|\alpha|)$$

Example



Minimum drag object of given area at Reynold 50 (Courtesy of Kawahara et al.).

Compressible Flows

Euler or Navier-Stokes equations

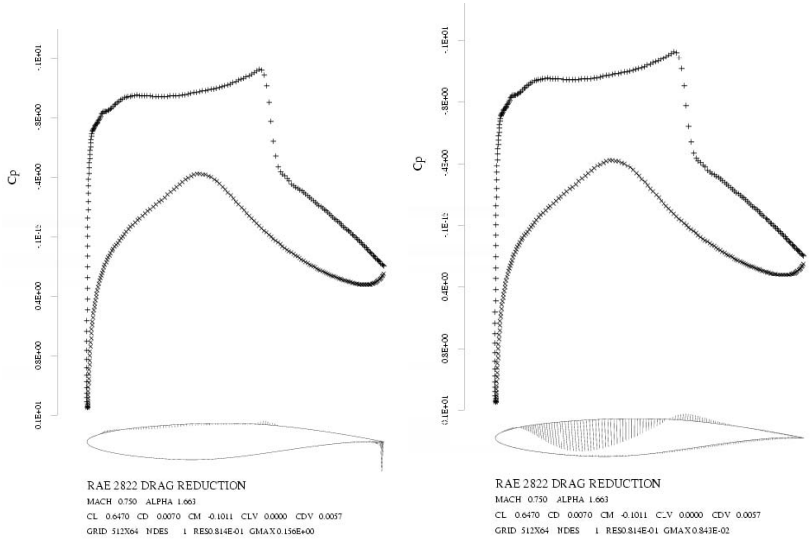
$$W = \begin{pmatrix} \rho \\ \rho u \\ \rho E \end{pmatrix} \qquad \partial_t W + \nabla \cdot F(W) - \nabla \cdot G(W, \nabla W) = 0$$

$$W(0, x) = 0, \quad + \text{B.C.}$$

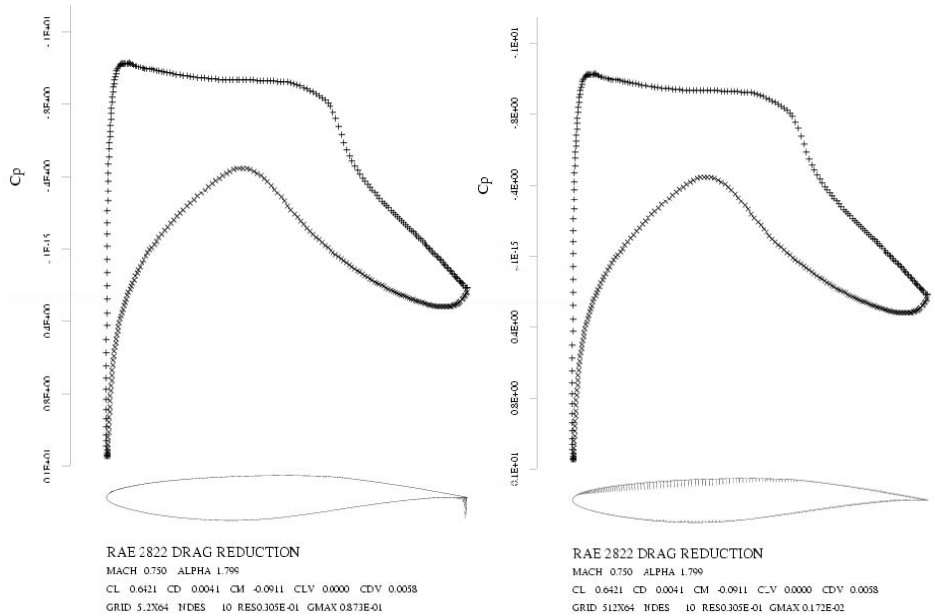
Involves an adjoint equation

$$\partial_t P + (F'(W) - G'_{,1}(W, \nabla W)^T \nabla P - \nabla \cdot (G'_{,2}(W, \nabla W)^T \nabla P) = 0$$

Some Realizations - A. Jameson (I)

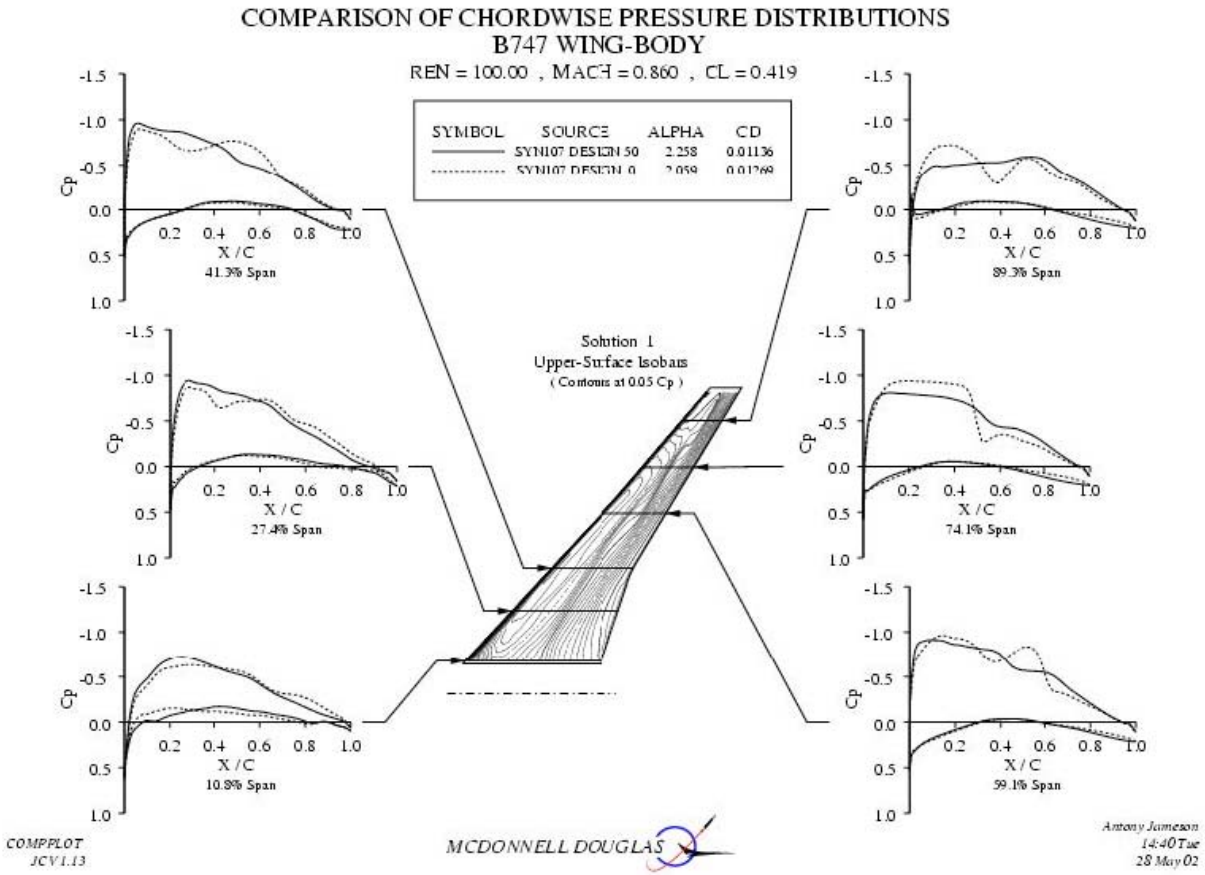


Plain vs Sobolev Gradients



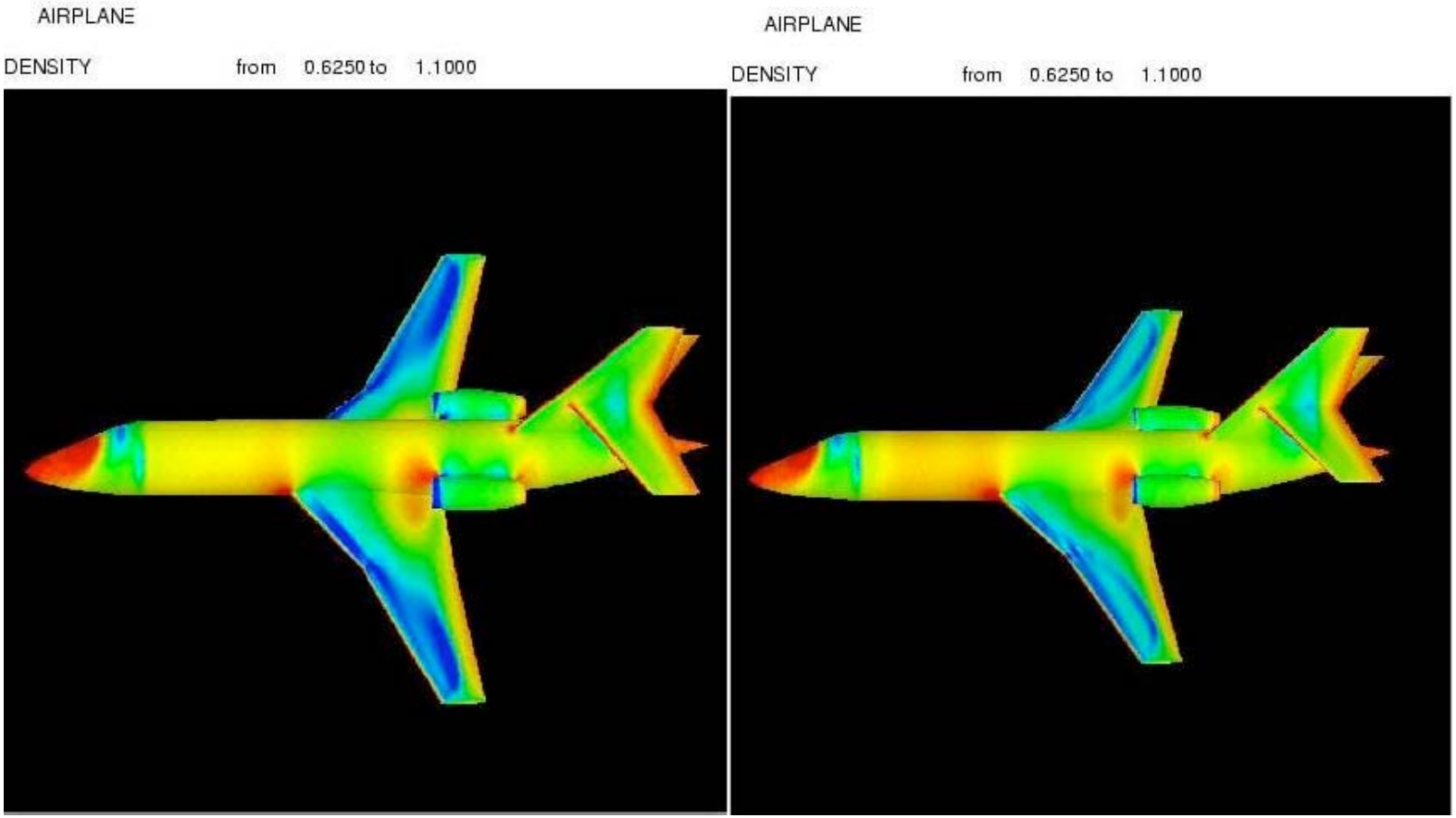
Before & after optimization

Some Realizations - A. Jameson (II)



Optimization of the Boeing 747: 10% wing drag saving (5% aircraft drag)

Some Realizations - A. Jameson (III)



Falcon jet: C_D decreases from 234 to 216

Outline

Summary

- Optimal Shape Design of S relies on Optimization

$$\min_S J(u,S) \ : \ A(S)u = f$$

- The Continuous problem is well posed after regularization

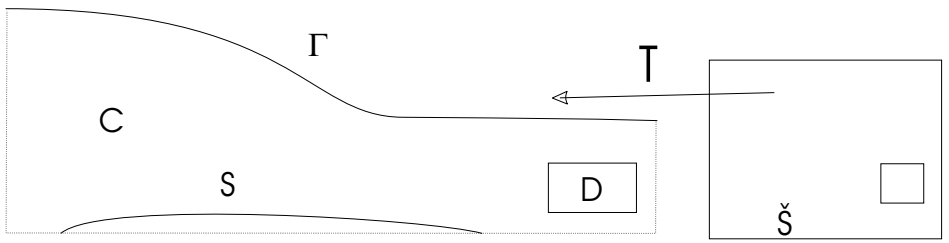
$$\min_S J(u,S) + \epsilon |S|^2 \ : \ A(S)u = f$$

- The L^2 local gradient χ is computable by calculus of variation:

$$\delta J = \int_S \chi\alpha + o(|\alpha|), \quad S(\alpha) = \{x + \alpha(x)n(x) \ : \ x \in S\}$$

- The Sobolev gradient is the right tool for gradient methods:

$$-\Delta_S \beta = \chi, \quad S^{n+1} = \{x - \rho\beta(x)n(x) \ : \ x \in S^n\}$$



$$\min_{\mathbf{s}\in\mathcal{S}_d} J(S) := \{ \int_D |\psi - \psi_d|^2 \ : \ -\Delta\psi = 0, \text{ in } C - \dot{S}, \quad \psi|_{\mathbf{S}} = 0 \ \ \psi|_{\partial C} = \psi_d \}$$

Discretization of gradients $J'_\alpha = \nabla\psi\nabla p$ where $-\Delta p = 2I_D(\psi - \psi_d)$, $p|_\Gamma = 0$ or derivation of gradient for the discrete problem?

Optimization of the Discrete Problem

- The Finite Element Method,
- Discrete Gradients
- Finite Volume Methods

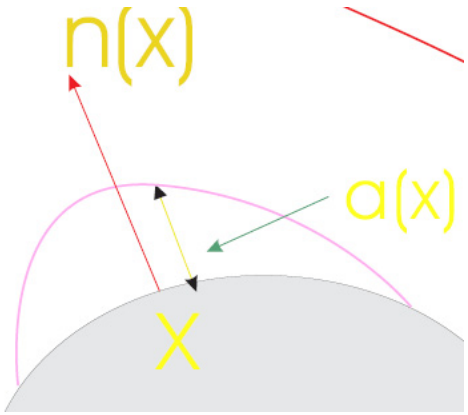
The Finite Element Method

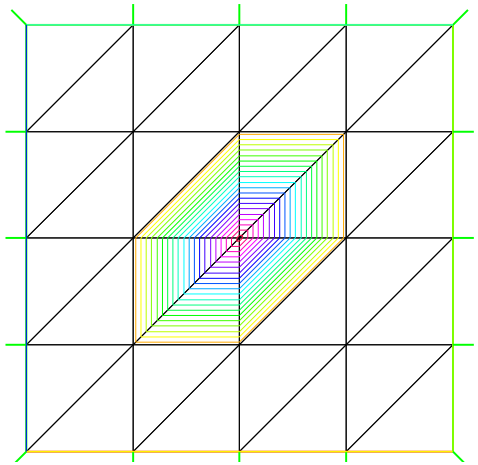
Ω is covered with triangles T_k and q^j are the vertices. The PDE of the wind tunnel problem is approximated by

$$\int_\Omega \nabla\psi_h \nabla w_h = 0, \quad \psi_h|_S = 0, \quad \psi_h|_\Gamma = \psi_d$$

for all w_h continuous and affine on each T_k and zero on $\partial\Omega$.

$$J = \int_D ||\psi_h - \psi_d||^2 + \epsilon \int_S |\frac{d^2\alpha}{dn^2}|^2$$





Let $\delta q_h(x) = \sum_i \delta q_i w(x)$, the basis $\{w^j\}$, the hat function of q_j.

Summary: Continuous versus Discrete Gradient

$$\min_{\mathbf{s}\in\mathcal{S}_d} J(S) := \{\int_D |\psi-\psi_d|^2 \ : \ -\Delta\psi = 0, \text{ in } C-\dot{S}, \quad \psi|_{\mathbf{s}} = 0 \ \ \psi|_{\partial C} = \psi_d\}$$

$$\delta J = \int_S \frac{\partial p}{\partial n}\frac{\partial \psi}{\partial n} \text{ with } -\Delta p = 2(\psi-\psi_d)I_D$$

use normal displacement $\approx v$: $-\Delta v = 0, \quad \frac{\partial v}{\partial n}|_\Gamma = \frac{\partial p}{\partial n}\frac{\partial \psi}{\partial n}$

For the discrete system

$$\min_{\mathbf{q}^i\in\mathcal{Q}_d} J(S_h) = \{\int_D |\psi_h-\psi_d|^2 : \int_\Omega \nabla\psi_h\cdot\nabla w^j = 0, \ \forall j \ \psi_h|_{\mathbf{s}} = 0 \ \ \psi_h|_{\partial C} = \psi_d\}$$

$$\delta J = \int_\Omega (\nabla\psi_h(\nabla\delta q_h + \nabla\delta q_h^T)\nabla p_h - \nabla\psi_h\cdot\nabla p_h \nabla\cdot\delta q_h) = \sum \chi_j \delta q^j$$

with $\int_\Omega \nabla p_h \nabla w^j = 2\int_D (\psi_h-\psi_d)w^j, \quad p_h \in V_{0h}$

And use a smoothed version of χ_j to move the vertices and find the new shape (and triangulation).

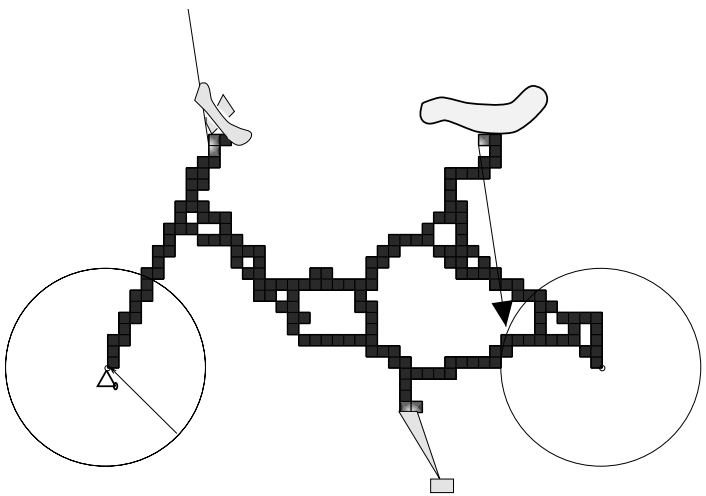
Part II

- Topological Optimization
- Shocks: extending Calculus of Variations

Topological Optimization

- Applies when the topology is not known
- Black-box favors Genetic algorithm (yet slow)
- Combine topological and geometrical shape design?

From T. Borrval and J. Petterson



From Schoenauer et al

Topological Derivatives

Following the work of L. Tartar and N. Kikuchi, J. Sokolowski came with the following idea (f has zero mean, $B(0,1)$ the unit ball)

$$-\Delta u = f \text{ in } \Omega, \qquad u|_\Gamma = 0,$$
$$-\Delta u^\epsilon = f \text{ in } \Omega \backslash B(x_0,\epsilon), \qquad u^\epsilon|_\Gamma = 0,$$
$$\text{Neumann or Dirichlet on } \partial B(x_0,\epsilon) = 0,$$
$$u'_{x_0}(x) = \lim_{\epsilon\to 0} \frac{1}{\epsilon^\gamma}(u^\epsilon - u)$$

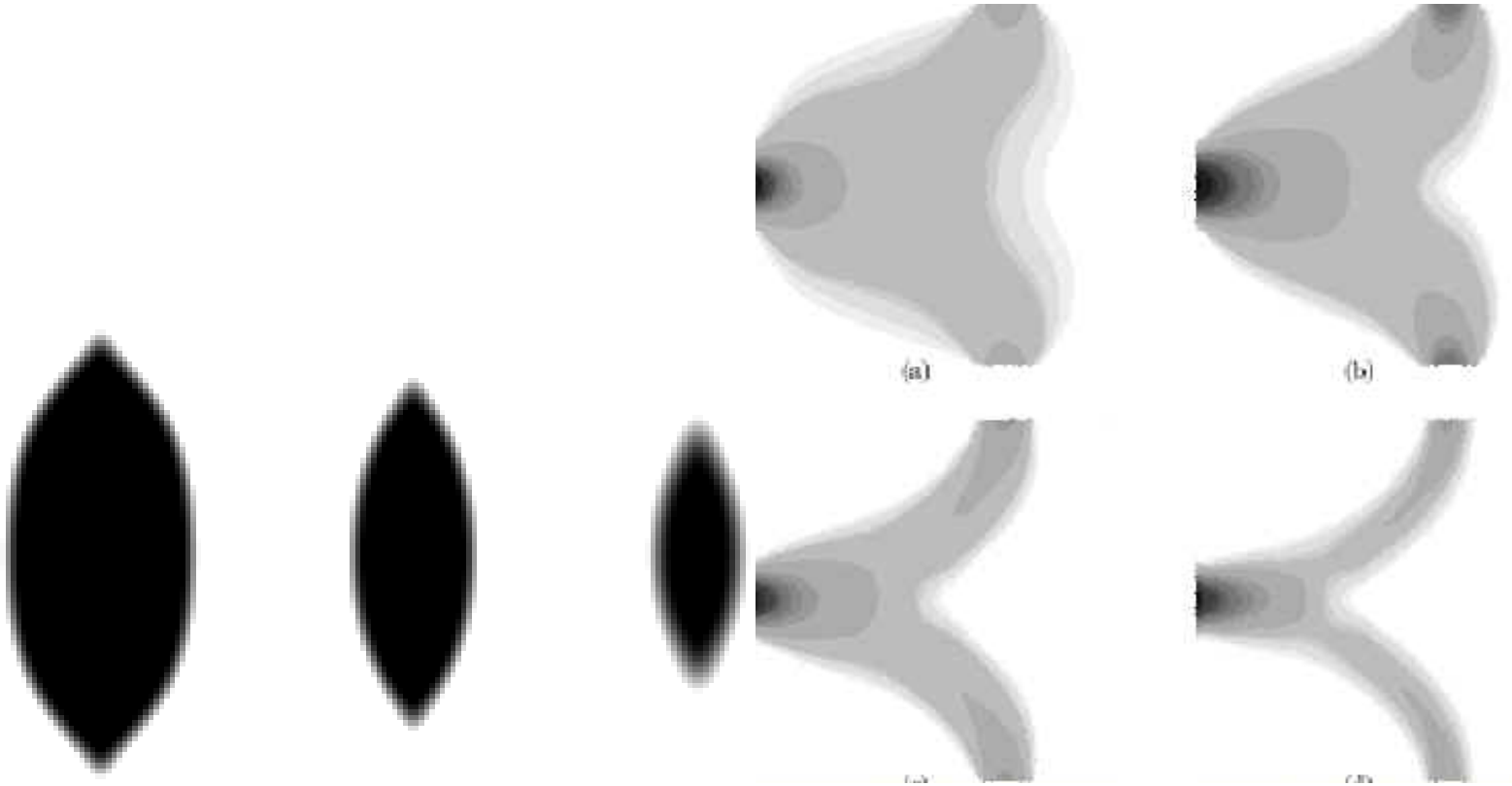
exists and is not identically 0 or $+\infty$ for some value of γ.

Theorem For the Neumann (resp Dirichlet) problem $\gamma = 2$ (resp $log\epsilon$) in 2D and u' solves

$$\int_\Omega \nabla u \cdot \nabla w = c \nabla u \nabla w|_{x_0}$$

This is sufficient for gradient type algorithm, but convergence is usually a problem.

Applications of Topological Optimization



Stokes flow drag optimization (courtesy of M. Masmoudi)

Micro Channel flow (Borrval and Petterson)

Optimization of a micro channel flow averaged vertically gives

$$\min_{z(x)\in Z} j(u) := \int_D (u-u_d)^2 \ : \ \frac{5}{2z^2}u - \Delta u + \nabla p = 0, \quad \nabla\cdot u = 0, \quad u|_\Gamma = g$$

where the pointwise values of functions of Z are equal to ϵ or h. Let $\rho = 2.5z^{-2}$; notice that

$$[\rho u] = \bar{\rho}[u] + [\rho]\bar{u} \qquad \bar{a} = \frac{a_1+a_2}{2} \qquad [a] = a_1 - a_2$$

Therefore if u' exists, the derivative w/r "ρ_2 becoming ρ_1" at x_0, it must be

$$\bar{\rho}u' + \rho'\bar{u} - \Delta u' + \nabla p' = 0 \quad \nabla\cdot u' = 0 \text{ with } \rho' = [\rho]\delta(x-x_0), \quad u'|\,\Gamma = 0$$

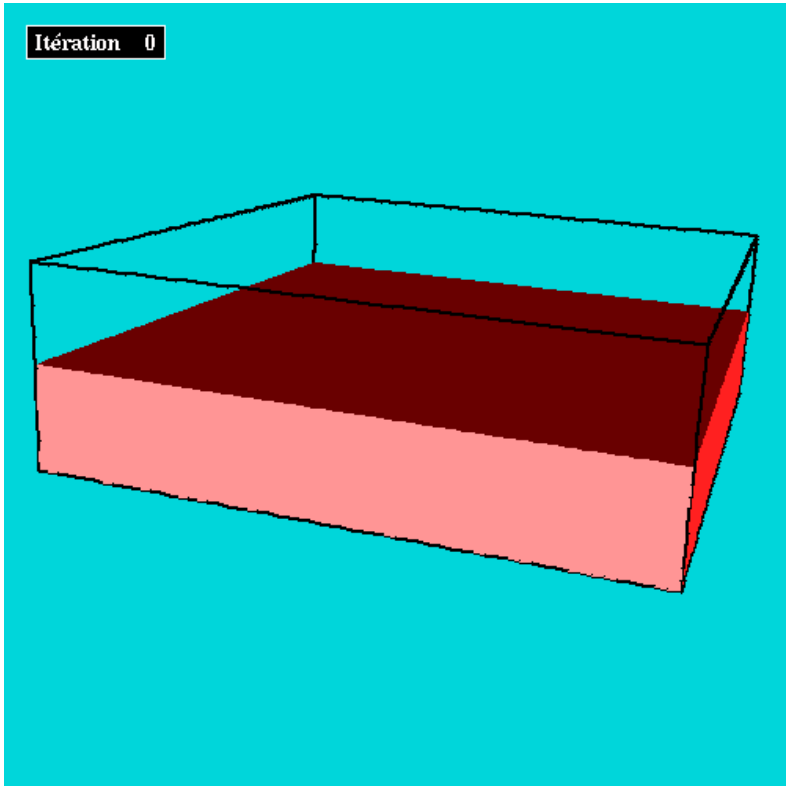
But u is continuous so $\bar{u} = u$. Introduce the adjoint state v, q

$$\bar{\rho}v - \Delta v + \nabla q = 0 \quad \nabla\cdot q = 2(u-u_d)\chi_D, \quad v|\,\Gamma = 0$$
$$\Rightarrow \quad j' = -[\rho]u(x_0)p(x_0)$$

Replace : ρ_2 by ρ_1 at x_0 when $[\rho]u(x_0)p(x_0) > 0$

Important Applications

Solid mechanics: Weight optimization of airplanes, cars, parts...



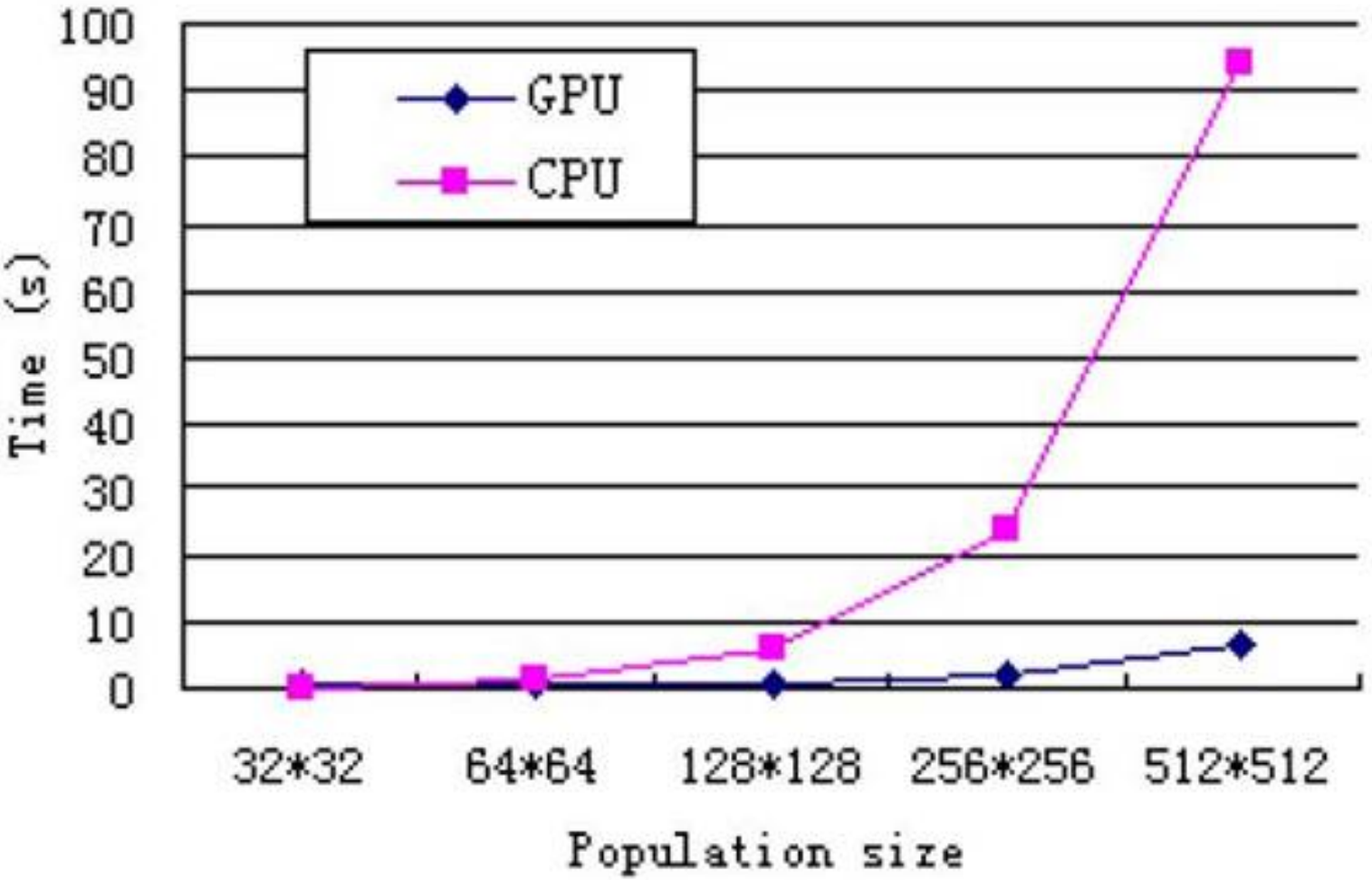
Topological optimization of the weight of a stool for a given strength (courtesy of F. Jouve et al)

Lesson II

Optimization of the continuous problem by discretization of the gradient is not allowed by the theory. Optimization of the discrete problem is possible but the formulas are complex. There is a middle way and there are acceleration methods

- Parallel and Stream Computing
- Mesh Refinement and Optimization
- Link with CAD
- Enormous systems: automatic Differentiations
- Some Results

GPGPU, CUDA, Nvidia and the CELL



Qizhi Yu et al. Parallel Genetic Algorithms on Programmable Graphics Hardware.

What to do To Use CUDA

You need Windows-Mac-Linux machine with Nvidia graphic card

- Download the compiler+Toolkit (+ graphic driver) $\Rightarrow$ `/usr/local/bin`
- To use the compiler easiest is to

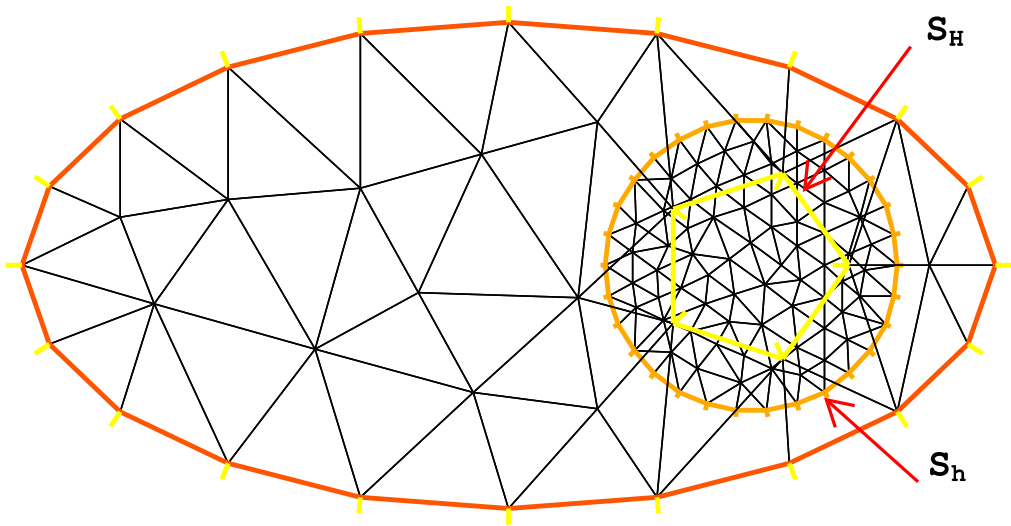
```
export PATH=/usr/local/cuda/bin:$PATH
export DYLD_LIBRARY_PATH
          =/usr/local/cuda/lib:$DYLD_LIBRARY_PATH
```

- To compile a *.cu program do `nvcc BSCuda.cu`
  also available in emulation mode: emul: `nvcc -deviceemu BSCuda.cu`
- Then launch by `./a.out`
- **Function Type Qualifiers**
  - `__device__` called by the GPU and run by the GPU
  - `__global__` called by the host and run by the GPU
  - `__host__` called by the host and run by the host
- Variables qual.: `__device__`, `__shared__`, `__constant__`

```
#include <math.h>
#define PI 3.14159265358979323846264338327950288f
const int NbBlocs = 20000, NbThreads = 500, N  = NbBlocs*NbThreads;
const float  K = 110, S0=100, r=0.02, sig=0.3, T=1.0,R = (r-sig*sig/2)*T;

__host__ __device__ void BS(float *x, float *y){
   float z = sqrtf(-2.0f * logf(*x)) * cosf(2.0f * PI * (*y));
   *y= S0*expf(R+sig*sqrtf(T)*z);
}
__global__ void BSgpu(float *a1, float *a2, int l) {
   int i = blockDim.x*blockIdx.x + threadIdx.x;
   if ( i < l ) BS(a1+i, a2+i);
}
void BScpu(float *a1, float *a2, int l) {
   for ( int i = 0 ; i < l ; ++i )  BS(a1+i, a2+i);
}

int main() {
   const int taille = N*sizeof(float);
   float *A1 = (float *) malloc(taille); // allocate A in CPU RAM
   float *A2 = (float *) malloc(taille);
   float *A3 = (float *) malloc(taille);
   float *B1, *B2;                       // will be allocated in GPU RAM
   srandom(time(NULL));
    for ( int n = 0 ; n < N ; ++n ){  // fill vector A with random nb
      A1[n] = (rand() + 0.5f)/(RAND_MAX + 1.0f);
      A2[n] = (rand() + 0.5f)/(RAND_MAX + 1.0f);
   }
   cudaMalloc( (void **) &B1, taille);           // allocate B1 in GPU RAM
   cudaMemcpy(B1, A1, taille, cudaMemcpyHostToDevice); // transfer A1 into B1
   cudaMalloc( (void **) &B2, taille);
   cudaMemcpy(B2, A2, taille, cudaMemcpyHostToDevice);

   BSgpu<<<NbBlocs, NbThreads>>>(B1,B2,N);

   cudaMemcpy(A3, B2, taille, cudaMemcpyDeviceToHost); // transfer results in A3

   BScpu(A1,A2,N);

   float put=0, err = 0;
   for ( int n = 0 ; n < N ; ++n ){
      err += fabs(A3[n] - A2[n]);
      put += fmax(K-A2[n],0.0f);
   }
   return 0;
```

The Schwarz-Zoom Method

γ_H (resp γ_h) is the interpolation operator on V_H (resp V_h)



Find $u_H^{m+1} \in V_H$, $u_h^{m+1} \in V_h$, such that $\forall w_H \in V_{0H}$, $\forall w_h \in V_{0h}$

$$a_H(u_H^{m+1}, w_H) = (f, w_H), \quad u_H^{m+1}|_{S_H} = \gamma_H u_h^m, \quad u_H^{m+1}|_{\Gamma_H} = g_H,$$
$$a_h(u_h^{m+1}, w_h) = (f, w_h), \quad u_h^{m+1}|_{S_h} = \gamma_h u_H^m, \quad u_h^{m+1}|_{\Gamma_h} = g_h$$

Convergence of Discrete Schwarz-Zoom Method

Hypothesis H1 Assume that the maximum principle holds for the 2 FEM-PDE and assume $\nu_H \in V_H$ solution of

$$a_H(\nu_H, w_H) = 0, \quad \forall w_H \in V_{0H}, \quad \nu_H|_{S_H} = 1, \quad \nu_H|_{\Gamma_H} = 0$$

satisfies $\lambda := |\nu_H|_{\infty,S_h} < 1$.

Theorem Under H1 Schwarz' algorithm converges towards u_H^*, u_h^*

$$a_H(u_H^*, w_H) = (f, w_H), \quad \forall w_H \in V_{0H}, \quad u_H^*|_{S_H} = \gamma_H u_h^*, \quad u_H^*|_{\Gamma_H} = g_H$$
$$a_h(u_h^*, w_h) = (f, w_h), \quad \forall w_h \in V_{0h}, \quad u_h^*|_{S_h} = \gamma_h u_H^*$$

and one has

$$\|u - u_H\|_{\infty,\Omega_H} + \|u - u_h\|_{\infty,\Omega_h} \leq C(H^2 \log \frac{1}{H} \|u\|_{2,\infty,\Omega_H} + h^2 \log \frac{1}{h} \|u\|_{2,\infty,\Omega_h})$$

Part of the Proof

Proposition Assume H1. Then the discrete Schwarz algorithm converges to the unique solution $(u_h^*, u_H^*) \in V_h \times V_H$ of the following system

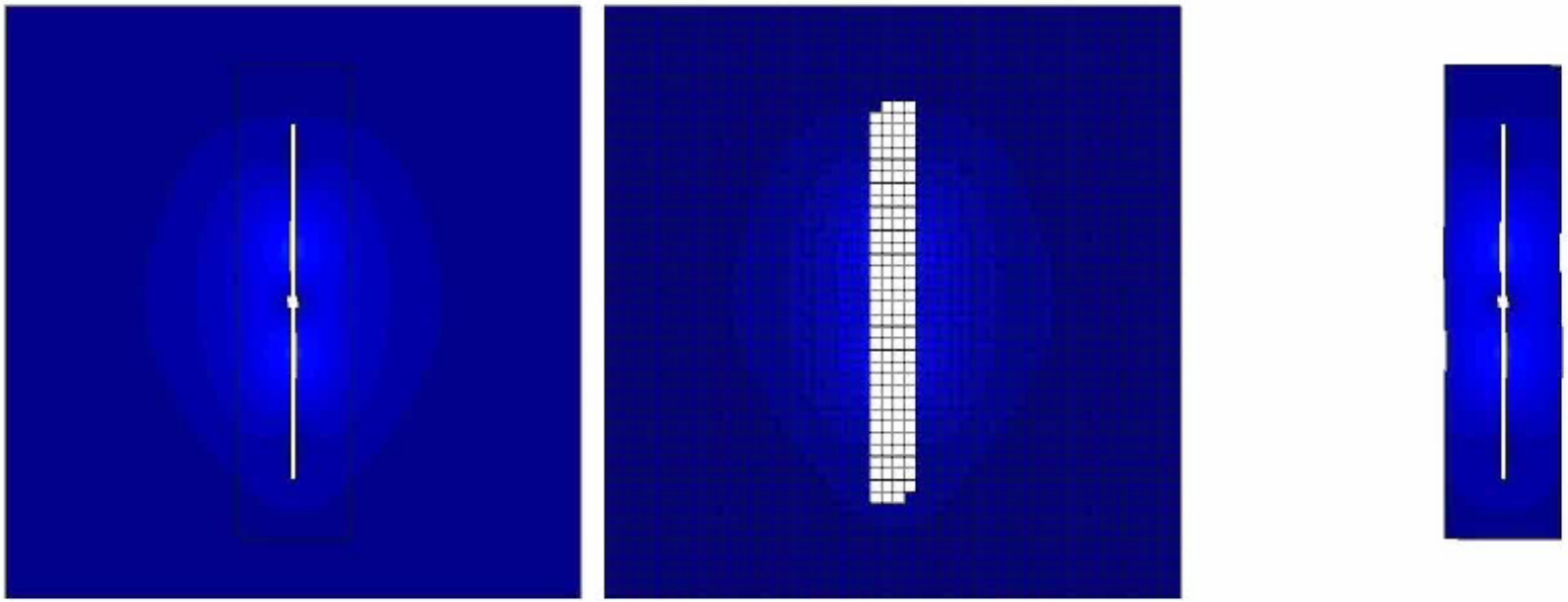
$$a_H(u_H^*, w_H) = (f, w_H), \quad \forall w_H \in V_{0H}, \quad u_H^*|_{S_H} = \gamma_H u_h^*, \quad u_H^*|_{\Gamma_H} = g_H$$
$$a_h(u_h^*, w_h) = (f, w_h), \quad \forall w_h \in V_{0h}, \quad u_h^*|_{S_h} = \gamma_h u_H^*$$

Proof By the maximum principle and as γ_H and γ_h decrease the L^∞ norms, $v_H \in V_H$, $v_h \in V_h$ defined by

$$a_H(v_H, w_H) = 0, \quad \forall w_H \in V_{0H}, \quad v_H|_{S_H} = \gamma_H u_h, \quad v_H|_{\Gamma_H} = 0$$
$$a_h(v_h, w_h) = 0, \quad \forall w_h \in V_{0h}, \quad v_h|_{S_h} = \gamma_h v_H$$
satisfy : $\|v_H\|_\infty \leq \|u_h\|_{\infty, S_H}, \quad \|v_h\|_\infty \leq \|v_H\|_{\infty, S_h}.$
Hence : $\|v_h\|_\infty \leq \|v_H\|_{\infty, S_h} \leq \lambda \|v_H\|_\infty \leq \lambda \|u_h\|_\infty.$

Let $T : V_h \to V_h$ define by $u_h^m = T(u_h^{m-1})$. Since T is affine, and contractant, by Banach's contraction theorem there is a unique fixed point $u_h^m \to u_h^*$ of T.

An example of time- Chimera/Schwarz



Courtesy of W. A. Wall

Steepest Descent Method

Objective: **Use inexact gradient in the context of mesh refinement**

$$\min_{z \in Z} J(z) \quad \text{approximated by} \quad \min_{z \in Z_h} J_h(z).$$

Algorithm

(Steepest descent with Goldstein's rule)

while $\| \, grad_z J_h(z^m) \| > \epsilon$ ***do***

{

$z^{m+1} = z^m - \rho \, grad_z J_h(z^m)$ *where* ρ *is any number satisfying,*

$$-\beta\rho\|w\|^2 < J_h(z^m - \rho w) - J_h(z^m) < -\alpha\rho\|w\|^2$$

with $w = grad_z J_h(z^m)$ *Set* $m := m + 1$*;*

}

Steepest Descent with Mesh Refinement

Now consider the same algorithm with parameter refinement

Algorithm

(Steepest descent with refinement)

while $h > h_{min}$ *do*

{

while $\| \, grad_z J_h(z^m) \| > \epsilon h^\gamma$ *do*

{

$z^{m+1} = z^m - \rho \, grad_z J_h(z^m)$ *where* ρ *such that,*

$$-\beta\rho\|w\|^2 < J_h(z^m - \rho w) - J_h(z^m) < -\alpha\rho\|w\|^2$$

with $w = grad_z J_h(z^m)$ *Set* $m := m + 1$;

}

$h := h/2$;

}

Steepest Descent and Inexact Gradients

- Convergence obvious : it is either S.Descent or $\text{grad}J_h \to 0$ because $h \to h/2$.
- Gain in speed : we do not need the exact gradient $\text{grad}_z J_h$!
- Let N be an iteration parameter and $J_{h,N} \approx J_h$ and $\text{grad}_z J_{h,N} \approx \text{grad}_z J_h$ in the sense that

$$\lim_{N\to\infty} J_{h,N}(z) = J_h(z) \qquad \lim_{N\to\infty} \text{grad}_{zN} J_{h,N}(z) = \text{grad}_z J_h(z)$$

Add K and $N(h)$ with $N(h) \to \infty$ when $h \to 0$:

Inexact Gradient (II)

Algorithm

(E. Polak et al)(Steepest descent with Goldstein's rule mesh refinement and approximate gradients)

while $h > h_{min}\{$

while $|\, grad_{z^N} J^m| > \epsilon h^\gamma\{$

try to find a step size ρ *with* $w = grad_{z^N} J(z^m)$

$$-\beta\rho\|w\|^2 < J(z^m - \rho w) - J(z^m) < -\alpha\rho\|w\|^2$$

if success then

$\{z^{m+1} = z^m - \rho\, grad_{z^N} J^m;\quad m := m+1;\}$

else $N := N + K;$

$\}$

$h := h/2;\quad N := N(h);$

$\}$

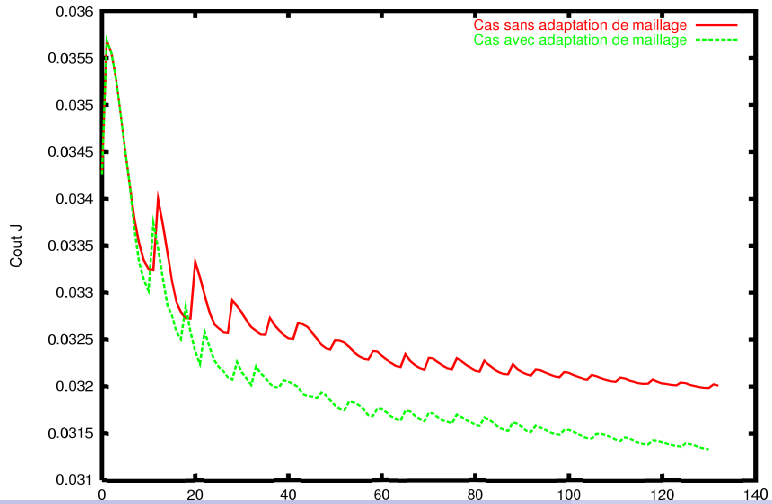
algorithm

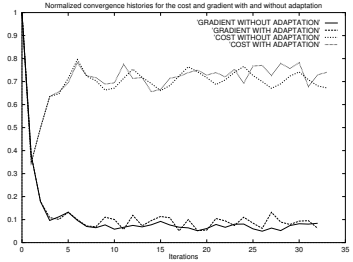
The convergence could be established from the observation that Goldstein's rule gives a bound on the step size:

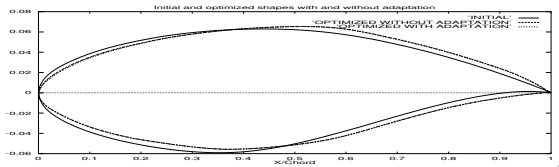
$$-\beta\rho\,\mathrm{grad}_z J\cdot h < J(z+\rho h)-J(z) = \rho\,\mathrm{grad}_z J\cdot h + \frac{\rho^2}{2}J''hh$$

$$\Rightarrow \quad \rho > 2(\beta-1)\frac{\mathrm{grad}_z J\cdot h}{J''(\xi)hh} \quad \text{so } J^{m+1}-J^m < -2\frac{\alpha(1-\beta)}{\|J''\|}|\,\mathrm{grad}_z J|^2$$

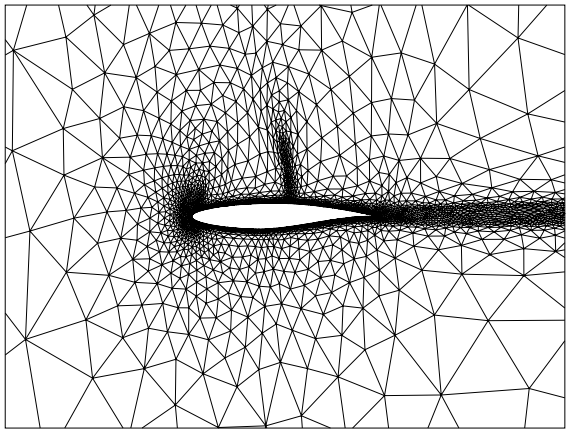
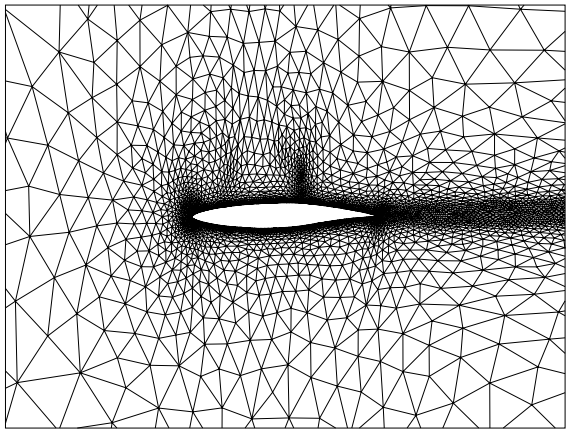
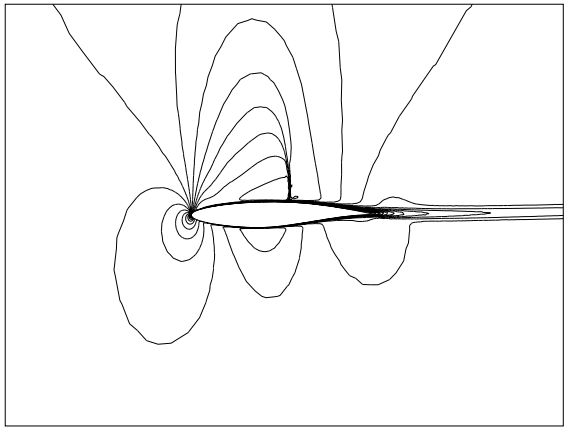
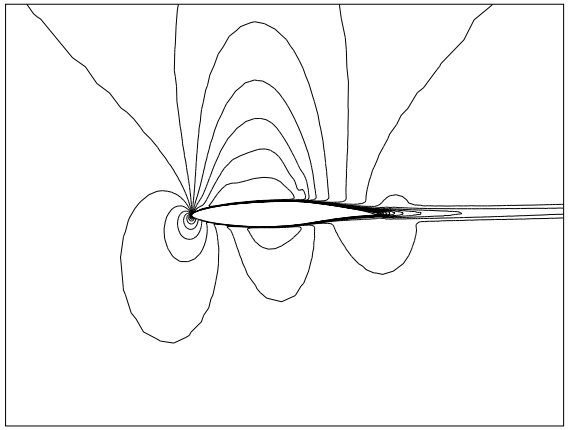
Thus at each grid level the number of gradient iterations is bounded by $O(h^{-2\gamma})$. Therefore the algorithm does not jam hence convergence.







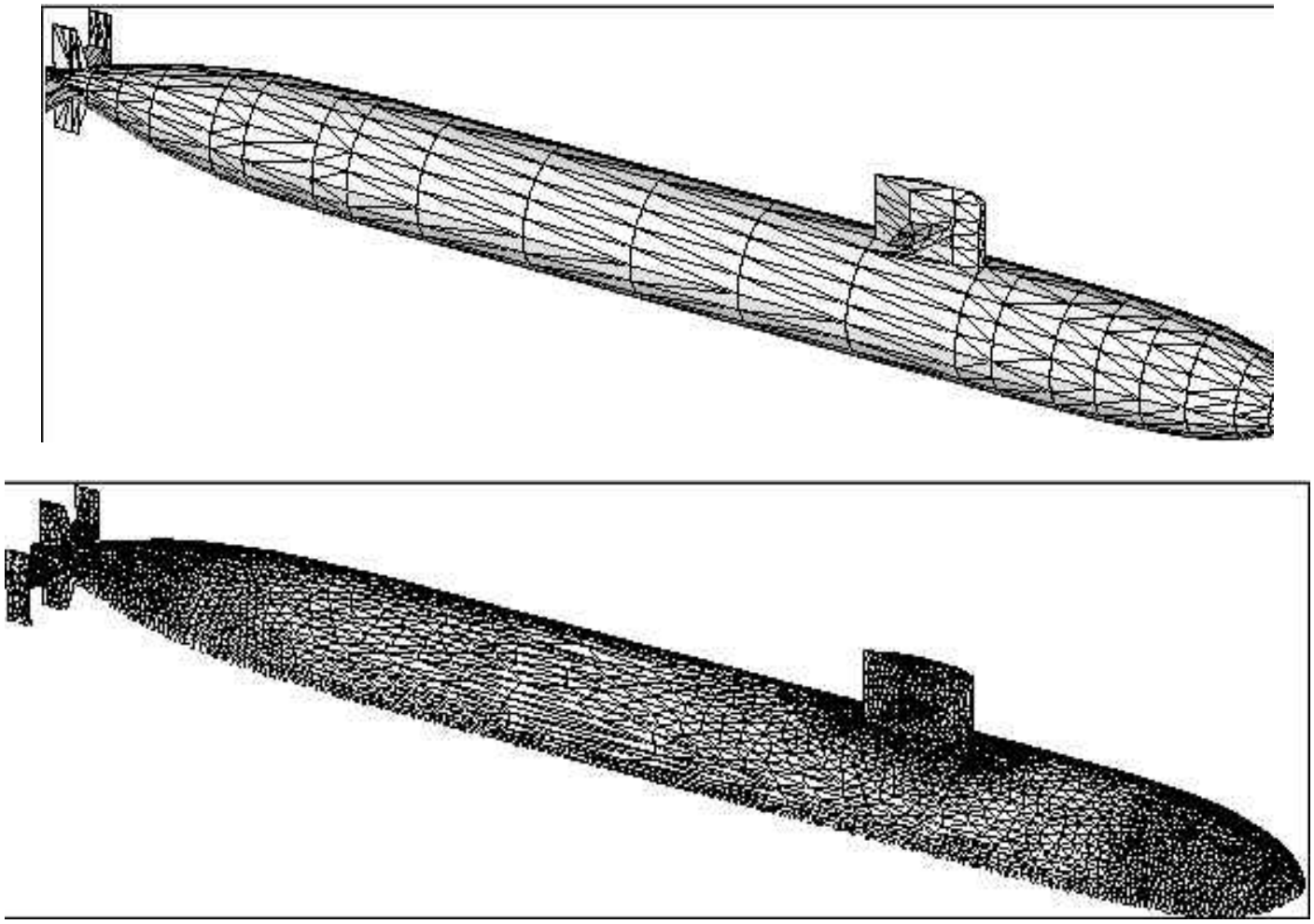
Mesh Refinements

Link with CAD

- CAD = Beziers patches or other with infinite details PROPRIETARY.
- 1. Generate a surface mesh
- 2. Get a C^1 + edges software for mesh refinement
- 3. Get a 3d volumic mesh generator (George-Hecht-Saltel...)
- 4. Do the optimization
- 5. Feed back into CAD (?)

$\Rightarrow$ CAD-free platform.

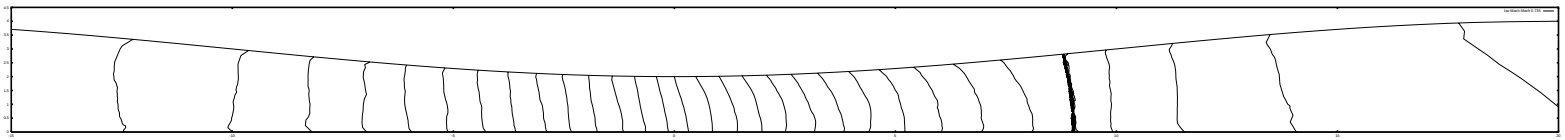


Mesh adaptation for the flow around a submarine

Huge Systems

$$\partial_t \rho + \nabla \cdot (\rho u) = 0$$
$$\partial_t (\rho u) + \nabla \cdot (\rho u \otimes u) + \nabla (p + \frac{2}{3}\rho k) = \nabla \cdot ((\mu + \mu_t) S)$$
$$\partial_t (\rho E) + \nabla \cdot ((\rho E + p + \frac{5}{3}\rho k) u) = \nabla \cdot ((\mu + \mu_t) S u) + \nabla ((\chi + \chi_t) \nabla T)$$
$$\partial_t \rho k + \nabla . (\rho u k) - \nabla ((\mu + \mu_t) \nabla k) = S_k$$
$$\partial_t \rho \varepsilon + \nabla . (\rho u \varepsilon) - \nabla ((\mu + c_\varepsilon \mu_t) \nabla \varepsilon) = S_\varepsilon .$$

Adjoint... use Automatic Differentiation of Programs (Adol-C, Tapenade)



in C++ by operator overloading $->$ OK up to 50 variables

Principle of Automatic Differentiation

Let $J(u) = |u - u_d|^2$, then its differential is

$$\delta J = 2(u - u_d)(\delta u - \delta u_d) \qquad \frac{\partial J}{\partial u} = 2(u - u_d)(1.0 - 0.0)$$

Obviously the derivative of J with respect to u is obtained by putting $\delta u = 1, \ \delta u_d = 0$. Now suppose that J is programmed in C/C++ by

```
double J(double u, double u_d){
    double z = u-u_d;
    z = z*(u-u_d);
    return z;
 }
int main(){  double u=2,u_d = 0.1;
    cout << J(u,u_d) << endl;
 }
```

A program which computes J and its differential can be obtained by writing above each differentiable line its differentiated form:

A simple example (cont)

```
class ddouble  {public: double v,d;
ddouble(double a, double b=0){ v = a; d=b;}
};

ddouble JandDJ(ddouble u, ddouble u_d)
{   ddouble z;
        z.d = u.d - u_d.d;
        z.v = u.v-u_d.v;
        z.d= z.d*(u.v-u_d.v) + z.v*(u.d - u_d.d);
        z = z*(u-u_d);
        return z;
}
int main()
 {
      ddouble u(2.,1.),   u_d= 0.1, J = JandDJ(u,u_d);
       cout << J << "  dJ="<<dJ<<endl;
 }
```

The class ddouble

```
class ddouble{ public: double v[2];
ddouble(double a, double b=0){ v[0] = a; v[1]=b;}
ddouble operator=(const ddouble& a)
  { val[1] = a.v[1]; val[0]=a.v[0];
    return *this;
  }
friend ddouble operator-(const ddouble& a,const ddouble& b)
       {    ddouble c;
            c.v[1] = a.v[1] - b.v[1];   // (a-b)'=a'-b'
            c.v[0] = a.v[0] - b.v[0];
            return c;
        }
friend ddouble operator*(const ddouble& a,const ddouble& b)
       {    ddouble c;
            c.v[1] = a.v[1]*b.v[0] + a.v[0]* b.v[1];
            c.v[0] = a.v[0] * b.v[0];
            return c;}
};
```

A Simple Example (final)

```
#include "ddouble.hpp"

ddouble J(ddouble u, ddouble u_d){
    ddouble z = u-u_d;
    z = z*(u-u_d);
    return z;
 }
int main(){
   ddouble u=2, u_d = 0.1;
   u.v[1]=1;
    cout << J(u,u_d).v[1] << endl;
 }
```

Simply replace all double by ddouble and link with the class lib.
A few pitfalls: e.g.

```
ddouble sqrt(ddouble x){     ddouble y;
 y.v[1]=x.v[1]/sqrt(fabs{x.v[0])+eps); y.v[0]=sqrt(x.v[0]);
 return y;   }
```

Limitations

```
program newtontest
x=0.0;
al=0.5
call newton(x,10,al)
write(*,*) x
end
```

```
subroutine newton(x,n,al)
do i=1,n
   f = x-alpha*cos(x)
   fp= 1+alpha*sin(x)
   x=x-f/fp
 enddo
 return
 end
```

2n adjoint variables are needed! while the theory is

$$f(x,\alpha)=0 \quad \Rightarrow \quad x' f'_x + f'_\alpha = 0 \quad \Rightarrow \quad x' = -\frac{f'_\alpha}{f'_x}$$

So it is better to understand the output of AD-reverse and clean it. see www.autodiff.org

Tapenade

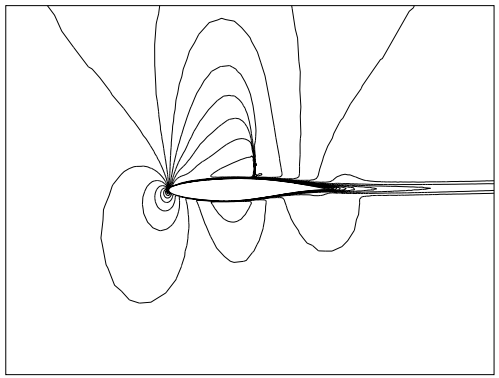
```
program newtontest
x=0.0
xb=2
al=0.5
call newton_b(x,xb,5,al,alb)
write(*,*) x,xb
end
SUBROUTINE NEWTON_B(x,xb,n,al,alb)
   DO i=1,n
     CALL PUSHREAL4(f)
     f = x - al*COS(x)
     CALL PUSHREAL4(fp)
     fp = 1 + al*SIN(x)
     CALL PUSHREAL4(x)
     x = x - f/fp
   ENDDO
   CALL PUSHINTEGER4(i-1)
   alb = 0.0
   CALL POPINTEGER4(nb)
   DO i=nb,1,-1
       CALL POPREAL4(x)
       fb = -(xb/fp)
       fpb = f*xb/fp**2
       CALL POPREAL4(fp)
       alb = alb+SIN(x)*fpb-COS(x)*fb
       xb = xb + al*COS(x)*fpb
     &   + (al*SIN(x)+1.0)*fb
       CALL POPREAL4(f)
       ENDDO
       END
```

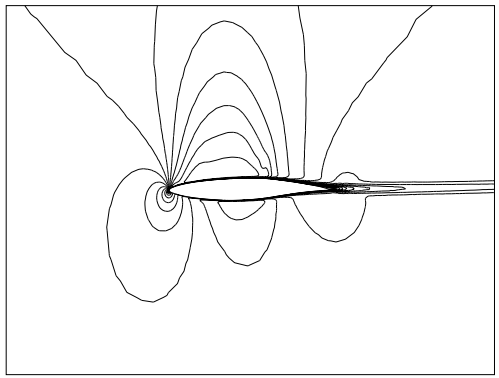
Optimization of a wing profile

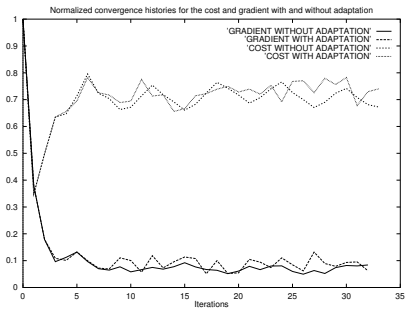
Drag is mostly pressure by the shock. The lift & area are imposed

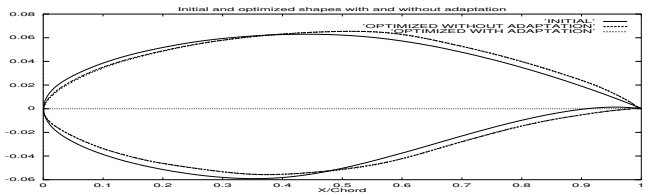
$$J(u,p,\theta) = F \cdot u_\infty + \frac{1}{\epsilon}|F \times u_\infty - C_l|^2 + \frac{1}{\beta}(\int_S dx - a)^2$$

with $F = \int_S (p\mathbf{n} + (\mu \nabla u + \nabla u^T))$ and Navier-Stokes + $k - \epsilon$ + wall laws

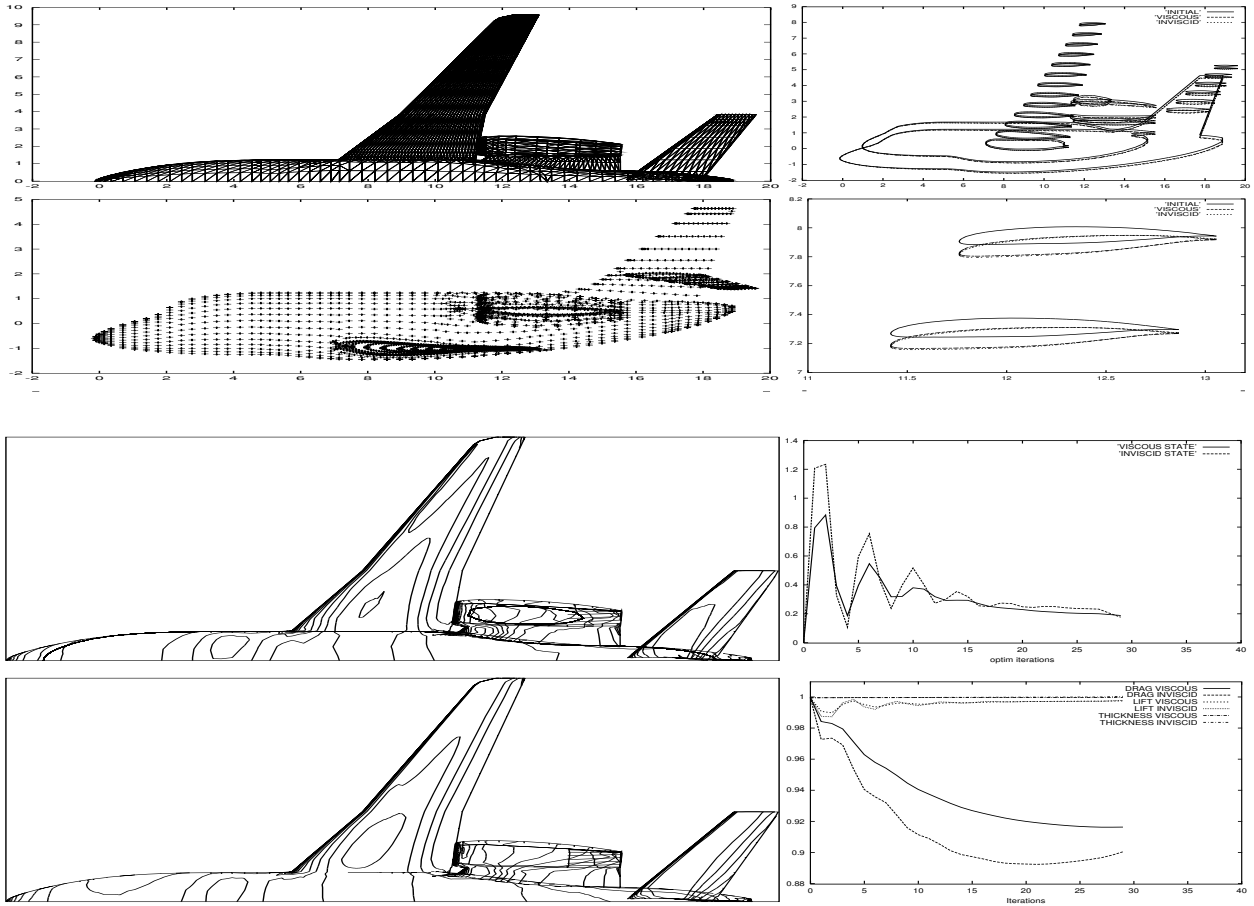








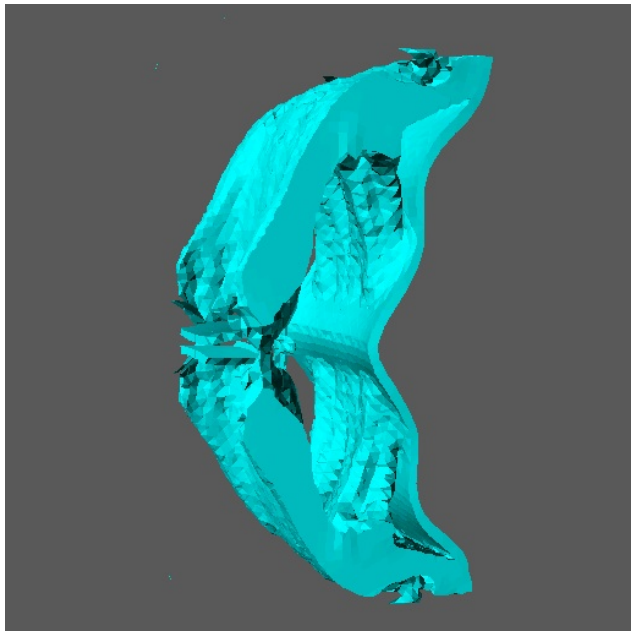
Optimization of a 3D Business Jet



Done by B. Mohammadi in a few hours on a workstation

Perspectives

- Differentiable Shape Optimization is complimentary to Evolutionary Methods
- It is the work of a specialist and it cannot be done overnight
- Yet 3D problems fit on a Workstation if care is applied
- It is an area where analysis really pays: a real meeting ground between good theory and practice



From G. Allaire - F. Jouve