Aerodynamic shape optimization using particle swarms and metamodels

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

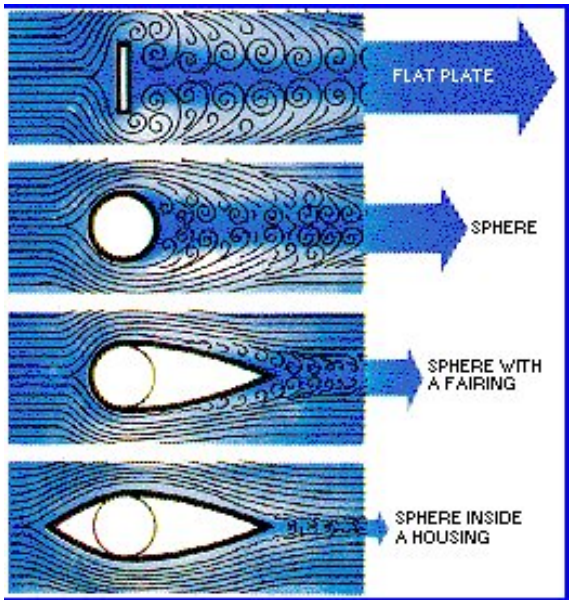
Tata Institute of Fundamental Research
Center for Applicable Mathematics
Bangalore 560065
http://math.tifrbng.res.in

Indo-German Workshop
IIT Madras
November 2008

Themes

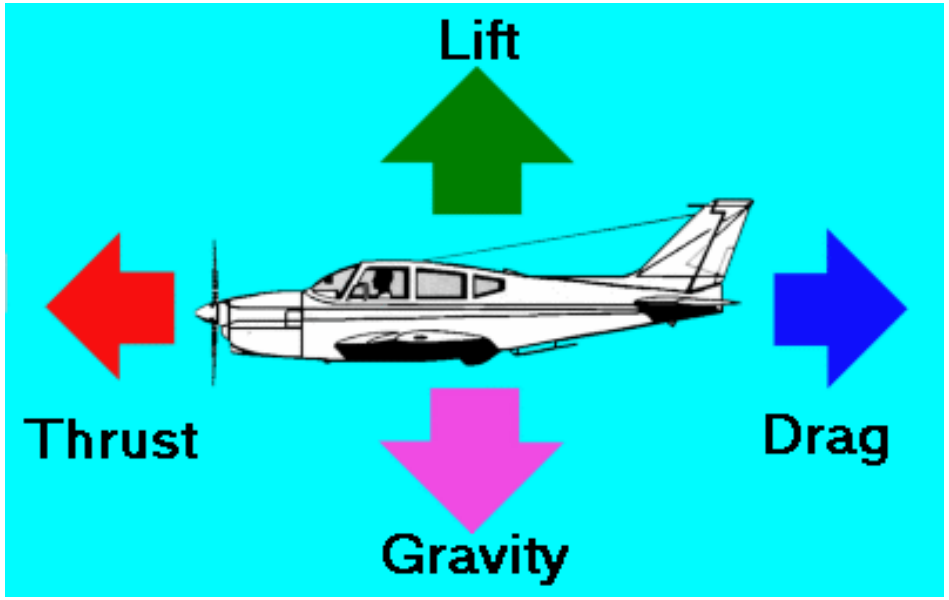
- Shape optimization
- Global optimization
- Optimization of system with expensive cost function
- Robust optimization

Effect of shape on flow quality



http://www.centennialofflight.gov

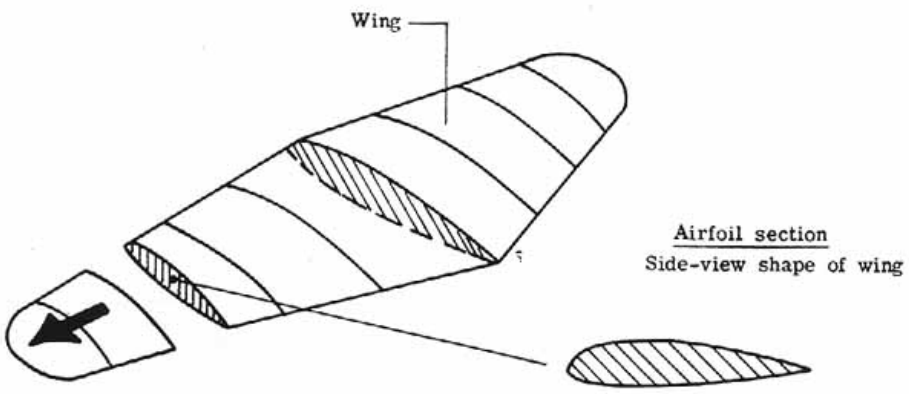
Forces on an airplane

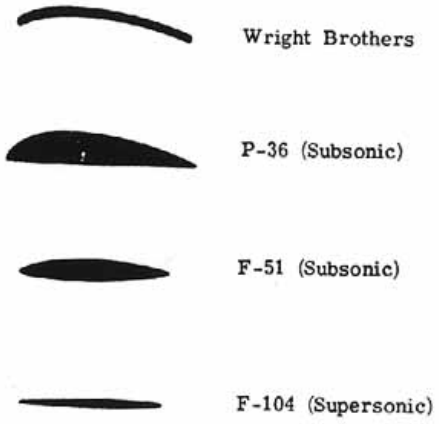


http://www.aviation-history.com

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

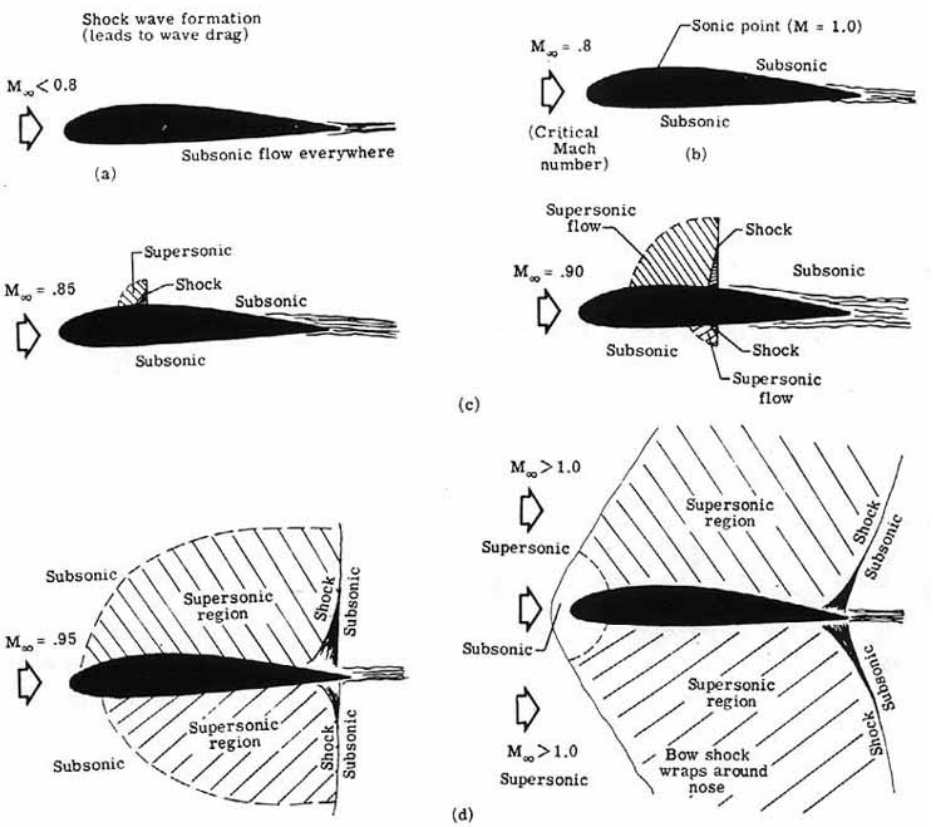
Wing and Airfoil





http://www.centennialofflight.gov

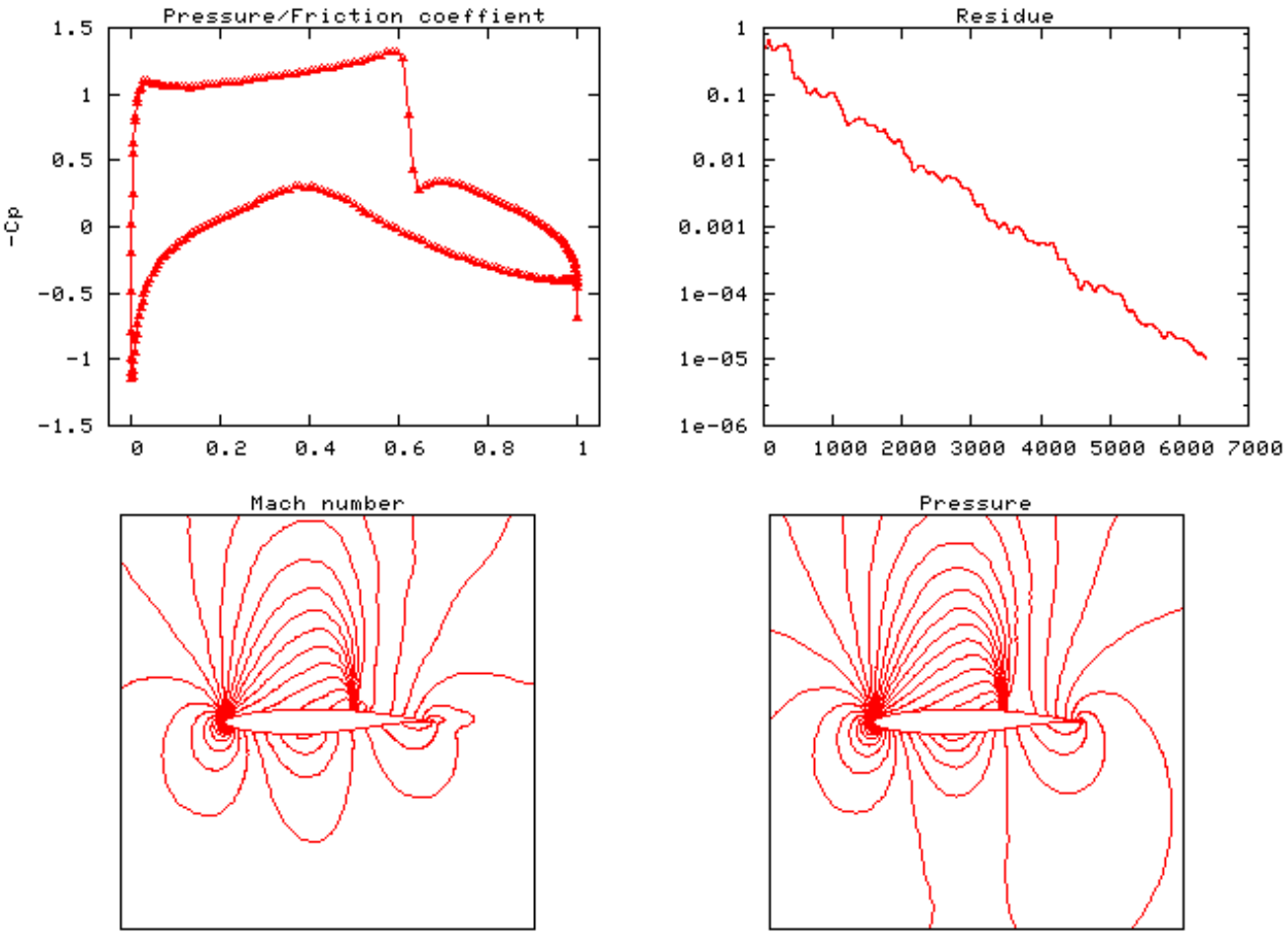
Effect of Mach number: M = fluid speed/sound speed



http://www.centennialofflight.gov

Example of optimization: RAE2822

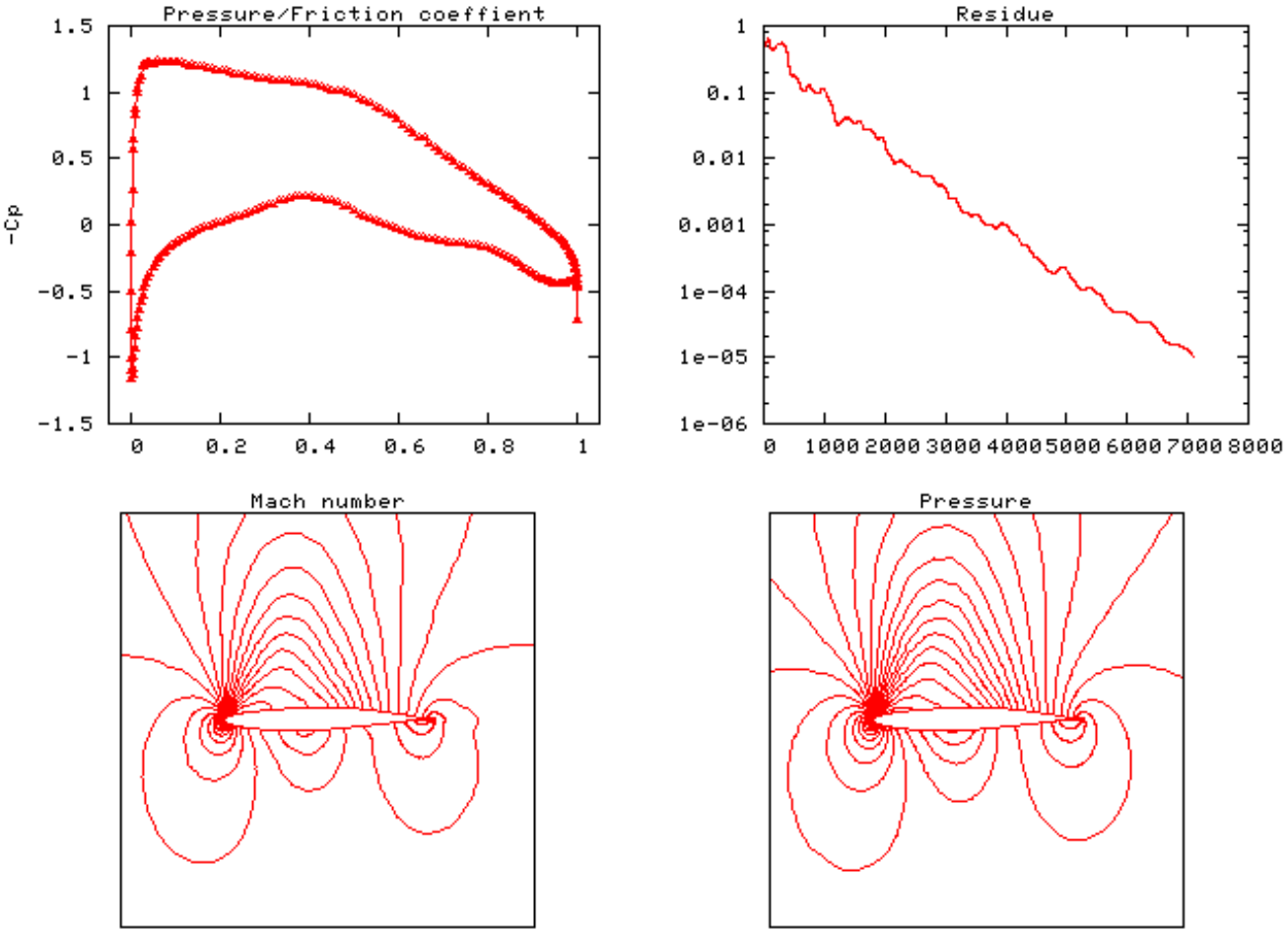
Initial shape



Solver: euler2d

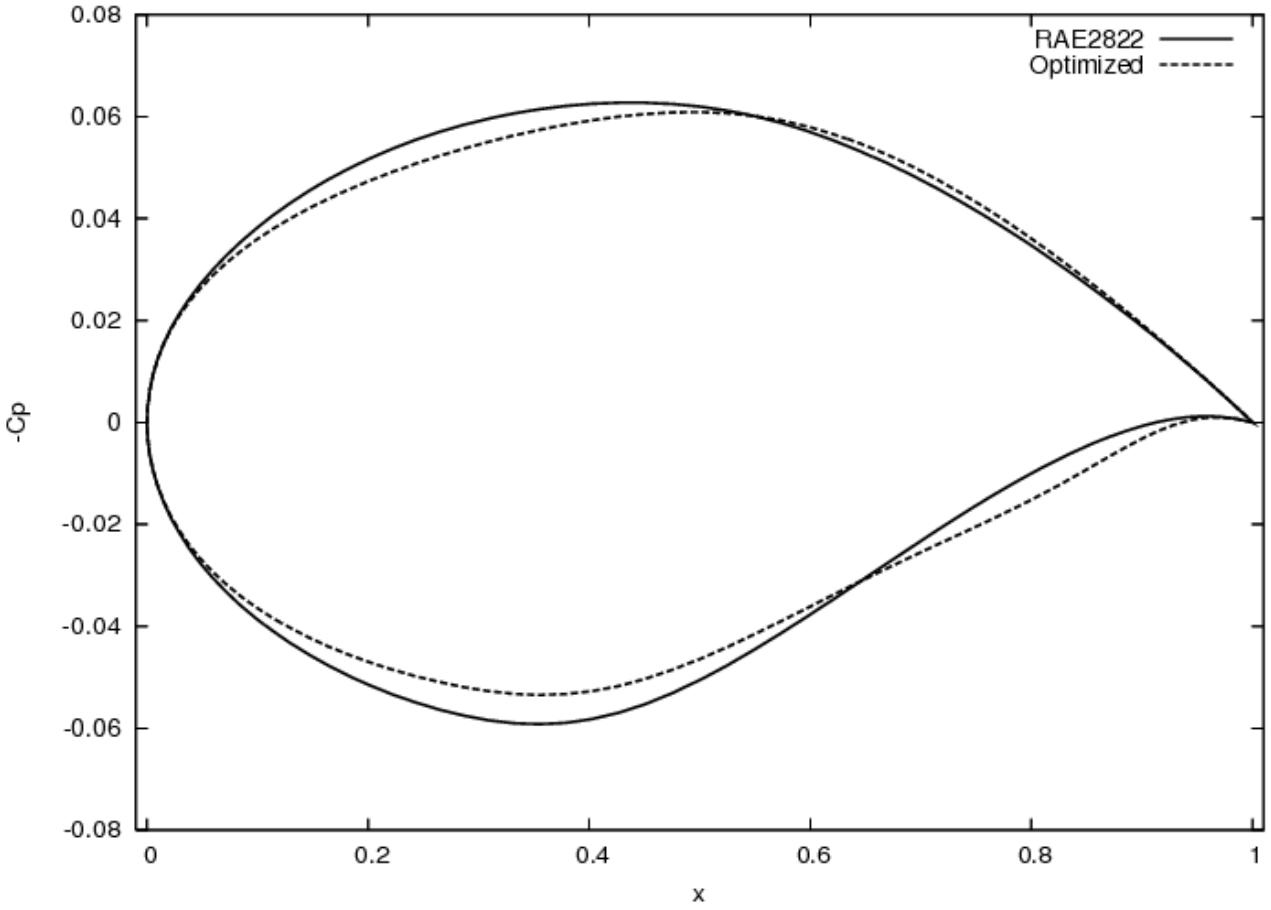
Example of optimization: RAE2822

Optimized shape



Optimizer: Torczon Simplex, 20 Hicks-Henne parameters

Example of optimization: RAE2822



Shape optimization

- Very small shape modifications:
  cannot be obtained using trial and error methods
- Transonic wings: possible to obtain shock-free flow with small shape changes
- 1% - 3% reduction in drag $\Longrightarrow$ significant fuel savings
- Many other applications
  - Turbomachinery
  - Ship hull design
  - Automobiles
  - Chemical industry

Shape optimization problem

- Cost function

$$J = J(S, u)$$

- u satisfies the flow equations (Euler or Navier-Stokes)

$$N(S, u) = 0$$

  u depends on shape S

- Minimization problem

$$\min_{S \in \mathcal{S}_{ad}} J(S, u), \quad \text{s.t.} \quad N(S, u) = 0$$

$$\mathcal{S}_{ad} = \text{Set of admissible shapes}$$

Classical method

- Let the shape S be given by a function $F(x)$.
- Variation in cost function: $F \to F + \delta F$

$$\delta J = \int G(x)\delta F(x)\mathrm{d}x$$

- G: Shape gradient
- Steepest descent: choose

$$\delta F = -\lambda G, \quad \lambda > 0$$

$$\delta J = -\lambda \int G^2(x)\mathrm{d}x \leq 0$$

Steepest descent method

1. Choose an initial shape $S^o = S(F^o)$
2. Set $n = 0$
3. Solve the flow equations

$$N(S^n, u^n) = 0$$

4. Compute shape gradient G^n
5. Change the shape

$$S^{n+1} = S(F^n + \delta F^n), \quad \delta F^n = -\lambda G^n$$

6. If $\|G^n\| < \mathrm{TOL}$, then STOP,
   else
   $n = n + 1$, go to step 3

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

Modern methods

- Global search methods
  - Genetic algorithm
  - Particle swarm optimization
  - Ant colony optimization
  - ...
- Do not require gradient
- Collection of M solutions at any iteration n

$$P^n = \{x_1^n, x_2^n, \ldots, x_M^n\}$$

- Solutions evolve according to some rules

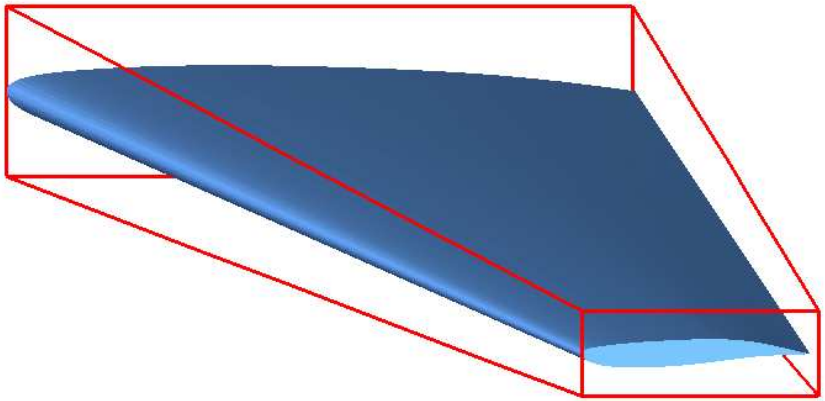
$$P^{n+1} = E(P^n)$$

Elements of shape optimization

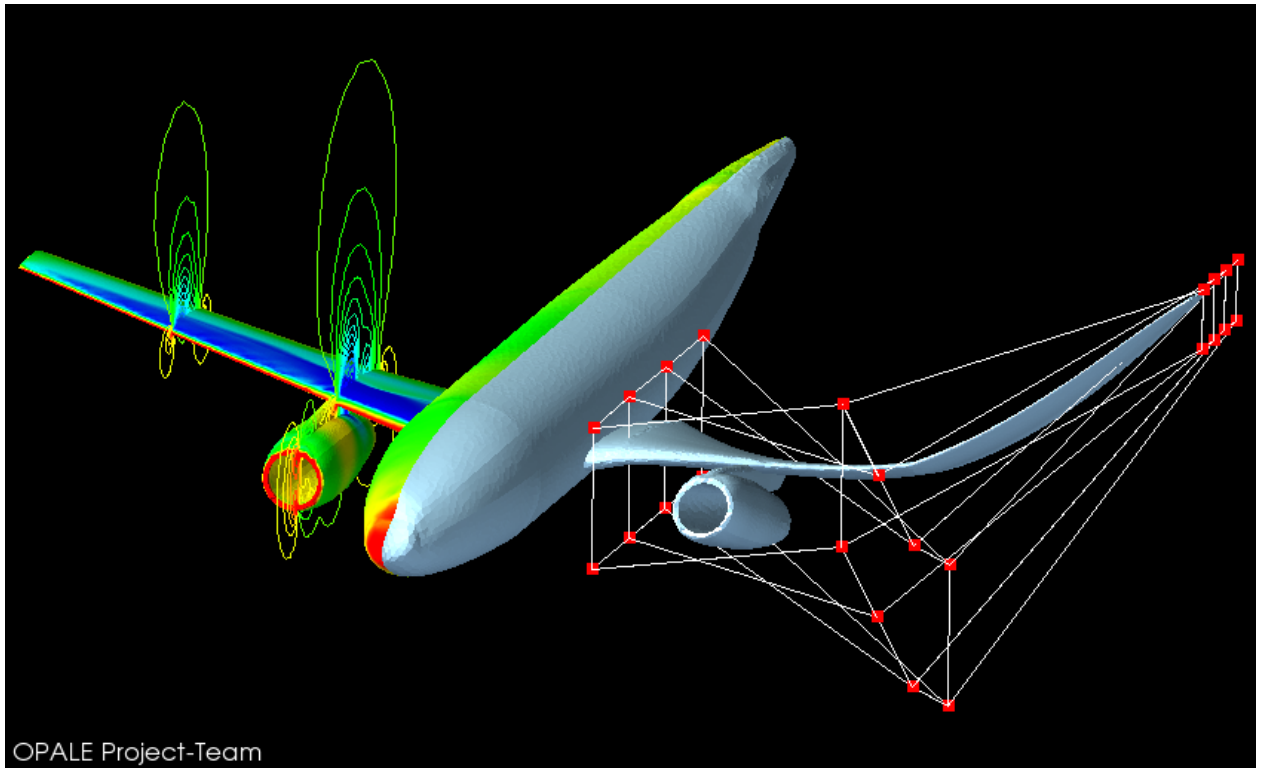
1. Shape parameterization
2. Surface grid generation/deformation
3. Domain grid generation/deformation
4. Flow solution (Euler/Navier-Stokes solver)
5. Adjoint flow solution (for gradient-based methods only)
6. Optimization method

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



Free Form Deformation

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

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

- Bernstein polynomials

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

- Design variables

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

Flow model

- Euler equations in conservation form

$$\frac{\partial U}{\partial t} + \frac{\partial E}{\partial x} + \frac{\partial F}{\partial y} + \frac{\partial G}{\partial z} = 0, \quad \text{in} \quad \Omega$$

- Conserved variables

$$U = \begin{bmatrix} \rho \\ \rho u \\ \rho v \\ \rho w \\ E \end{bmatrix}$$

ρ : fluid density
(u, v, w) : fluid velocity
E : energy per unit volume

- (E, F, G) = flux vector

Integral form

- For any fixed volume $V \subset \mathbb{R}^3$

$$\frac{\mathrm{d}}{\mathrm{d}t}\int_V U \mathrm{d}x + \oint_{\partial V} (En_x + Fn_y + Gn_z)\mathrm{d}S = 0$$

- Divide Ω into disjoint polygonal *finite volumes*

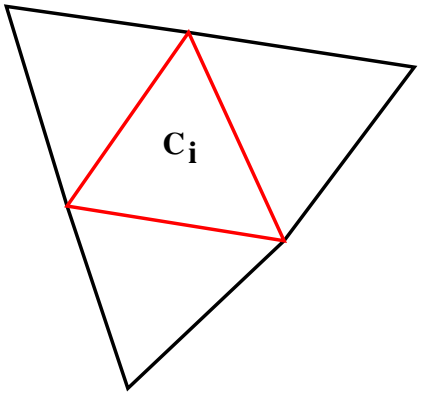
$$\Omega = \cup_{i=1}^{N_c} C_i$$

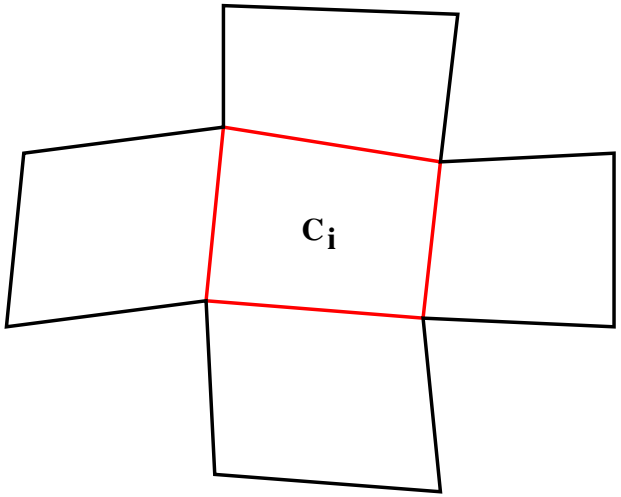
- Cell average value

$$U_i(t) = \frac{1}{|C_i|}\int_{C_i} U(x,t)\mathrm{d}x$$

Finite volume method

- Cell connectivity: $N(i) = \{j : C_j \text{ and } C_i \text{ share a common face}\}$

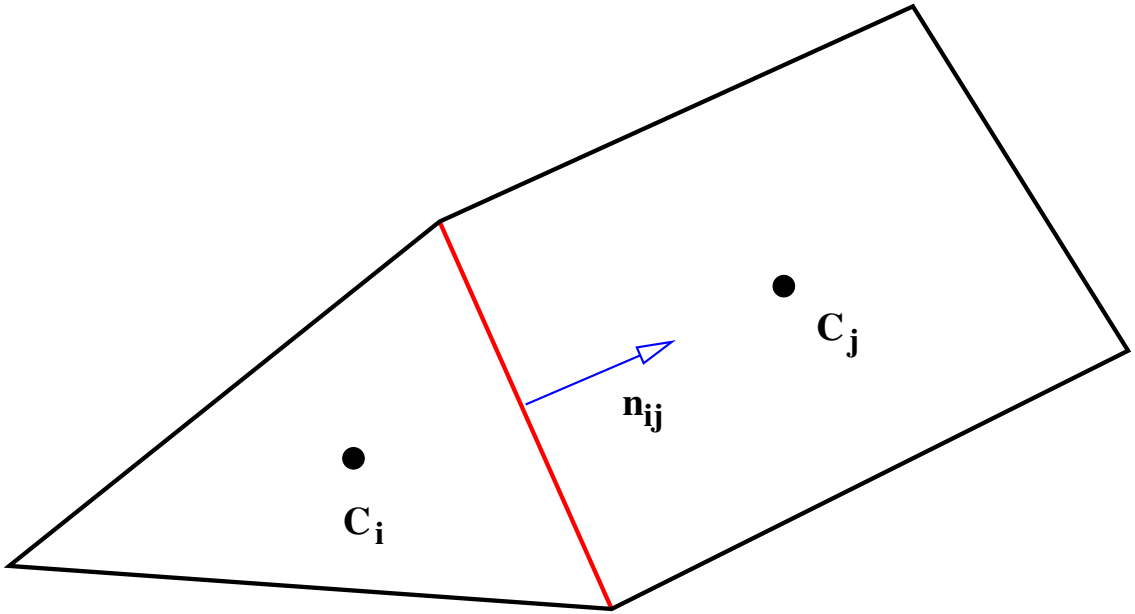




$$\oint_{\partial C_i} (En_x + Fn_y + Gn_z)\mathrm{d}S = \sum_{j \in N(i)} \int_{C_i \cap C_j} (En_x + Fn_y + Gn_z)\mathrm{d}S$$

Finite volume method

- Approximate flux integral by quadrature



$$\int_{C_i \cap C_j} (En_x + Fn_y + Gn_z)\mathrm{d}S \approx H_{ij}\Delta S_{ij}$$

- Semi-discrete update equation (ODE)

$$|C_i|\frac{\mathrm{d}U_i}{\mathrm{d}t} + \sum_{j \in N(i)} H_{ij}\Delta S_{ij} = 0, \quad H_{ij} = H(U_i, U_j, \hat{n}_{ij})$$

FVM Algorithm: Explicit, steady-state

$$\text{Find } U \text{ such that} \quad \frac{\partial E}{\partial x} + \frac{\partial F}{\partial y} + \frac{\partial G}{\partial z} = 0$$

1. Set $n = 0$, initial condition U^0, choose time step Δt
2. Update solution to next time level

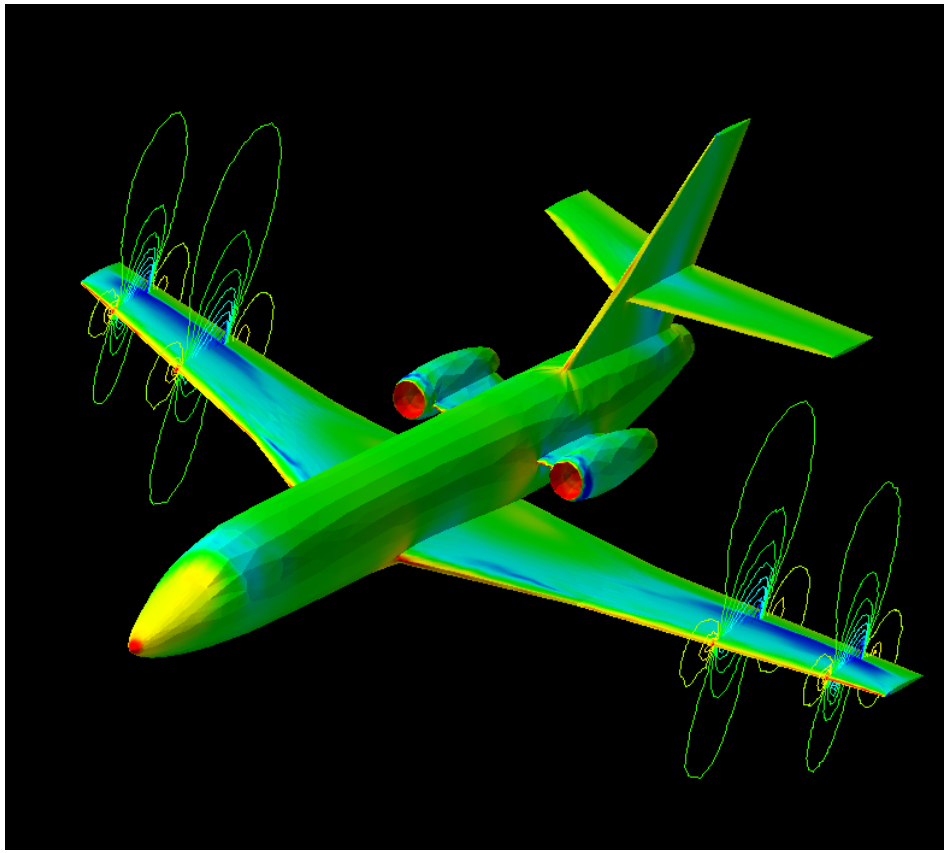
$$U_i^{n+1} = U_i^n - \frac{\Delta t}{|C_i|} \sum_{j \in N(i)} H_{ij}^n \Delta S_{ij}, \quad i = 1, \ldots, N_c$$

3. If $\|(U^{n+1} - U^n)/\Delta t\| > \mathrm{TOL}$, then $n = n + 1$, go to step 2
4. Compute cost function

$$J = J(x, U^{n+1})$$

CFD example: Falcon

Mach = 0.85



Grid: 455160 nodes, Solver: Num3sis

Particle swarm optimization

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

Optimization problem

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

Particle swarm optimization

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

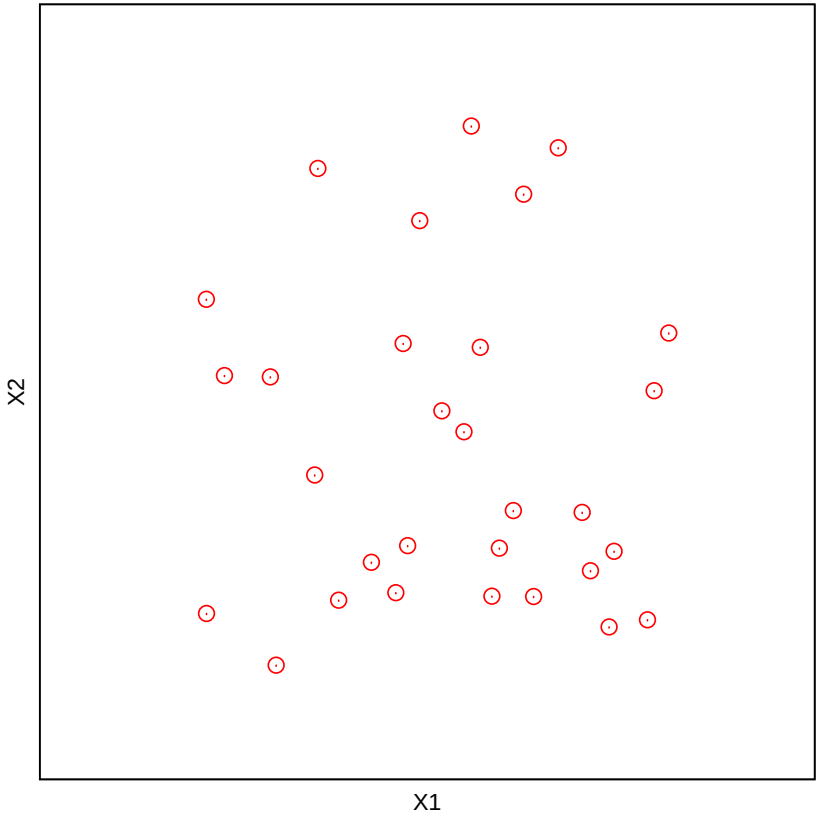
Optimization problem

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

Particle swarm optimization

Particles distributed in design space

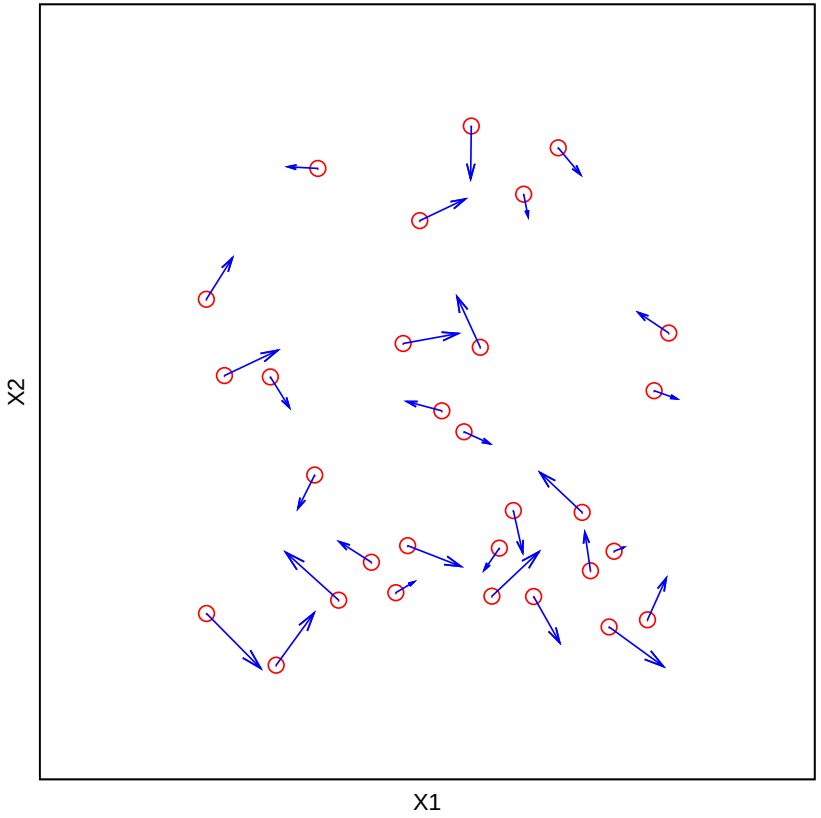
$$x_i \in D, \quad i = 1, ..., N_p$$



Particle swarm optimization

Each particle has a velocity

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



Particle swarm optimization

- Particles have memory (t = iteration number)

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

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

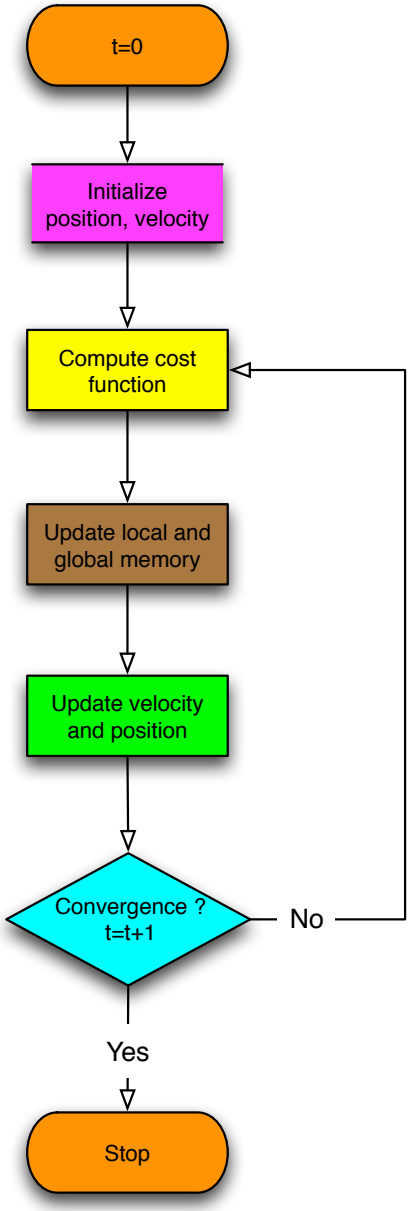
- Velocity update

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

- Position update

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

Basic PSO algorithm



PSO: Parallelizability

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

Parallel evaluation of cost functions using MPI: Message Passing Interface

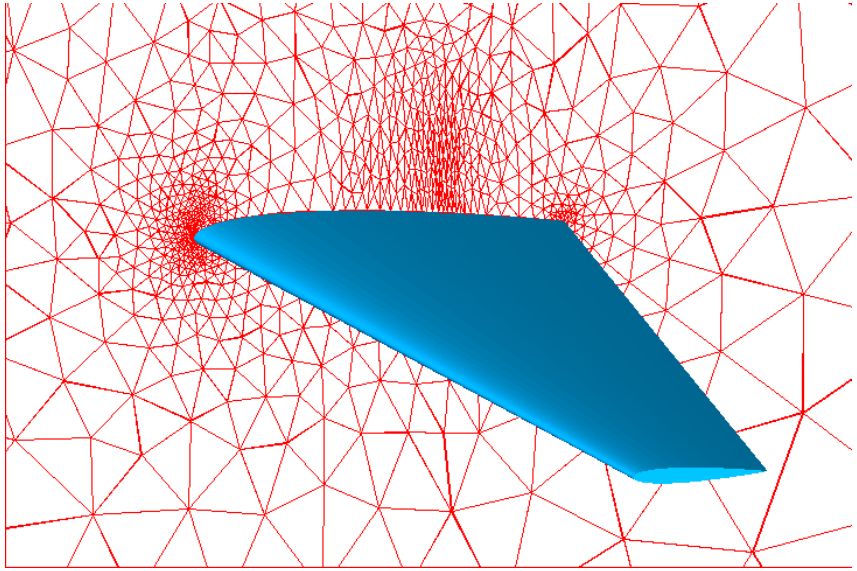
Test case: Wing shape optimization

- 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

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



(Piaggio Aero. Ind.)
Grid: 31124 nodes

Cost function

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

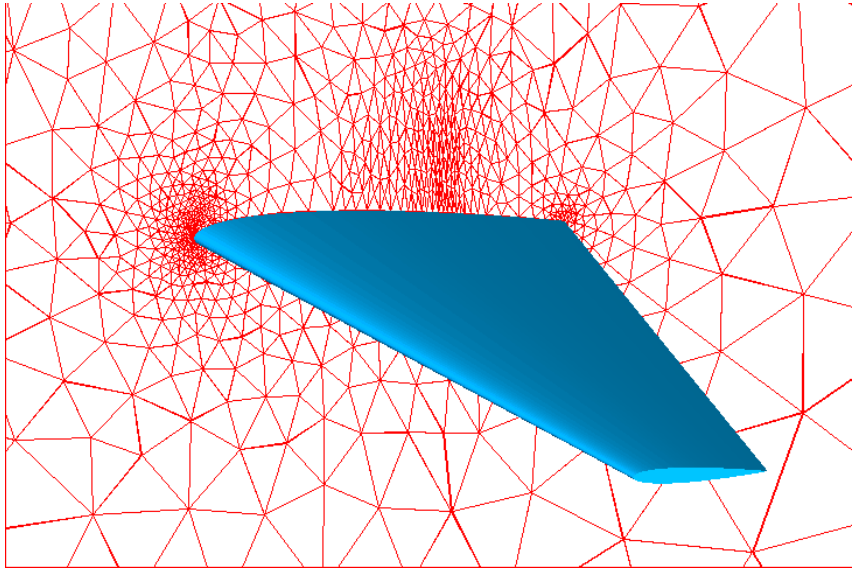
Test case: Wing shape optimization

- 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

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



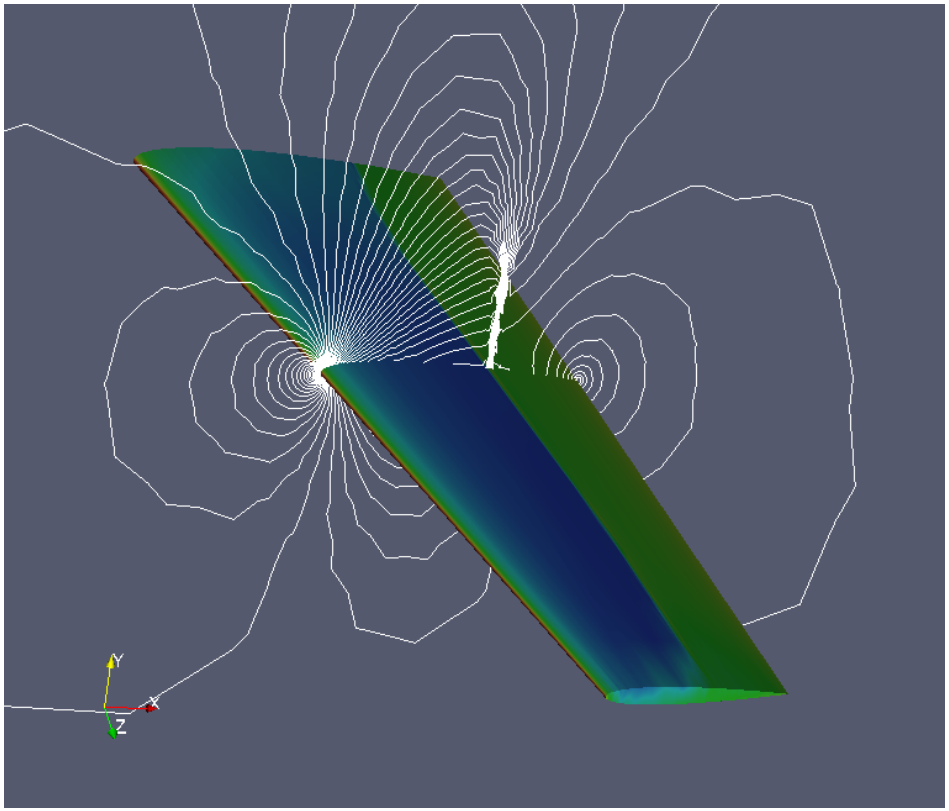
(Piaggio Aero. Ind.)
Grid: 31124 nodes

Cost function

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

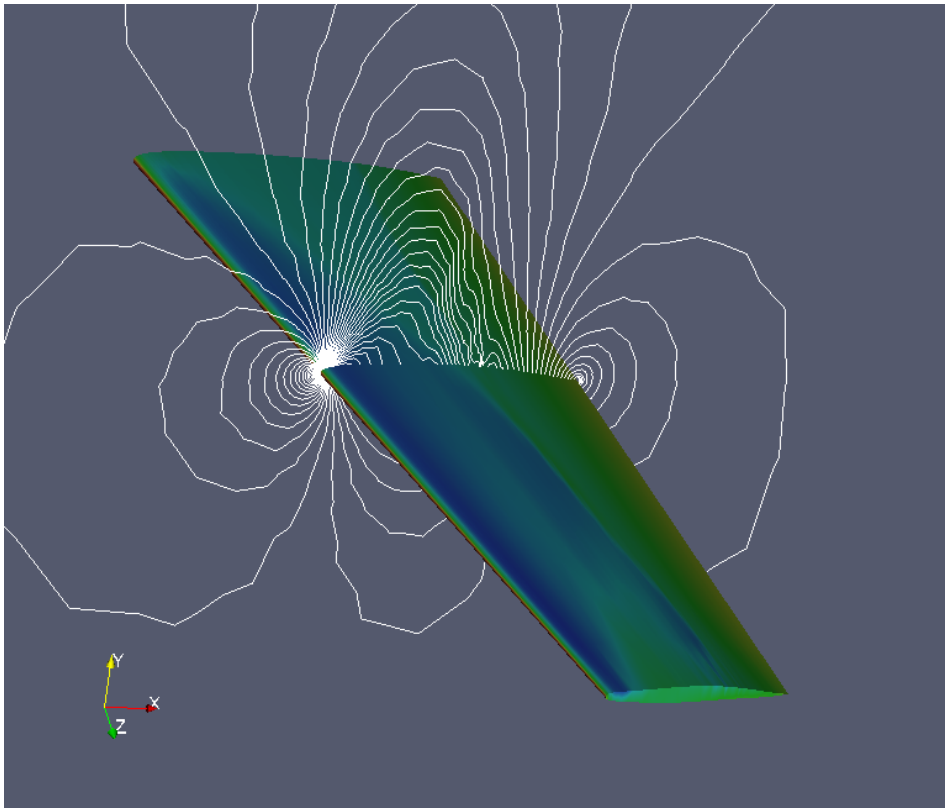
Wing optimization

Initial shape



Wing optimization

Optimized shape



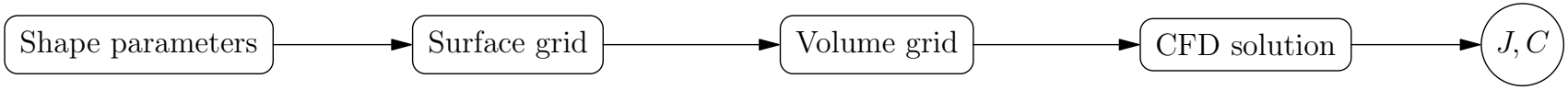
PSO computational cost

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

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

Metamodels

- Expensive model



- Replace costly model with cheap model:
  metamodel or surrogate model

- Approximation of cost function and constraint function(s)
  - Response surfaces (polynomial model)
  - Neural networks
  - Radial basis functions
  - Kriging

Metamodels

- Population based methods
  - Large amount of information
  - Information lost in subsequent iterations
- Save information in a database
- Use database to construct (meta) models
- Inexact Pre-Evaluation (IPE) (Giannakoglou et al.) with Genetic Algorithm

In this work

Combine IPE with PSO for aerodynamic shape design

Metamodels

- Population based methods
  - Large amount of information
  - Information lost in subsequent iterations
- Save information in a database
- Use database to construct (meta) models
- Inexact Pre-Evaluation (IPE) (Giannakoglou et al.) with Genetic Algorithm

In this work

Combine IPE with PSO for aerodynamic shape design

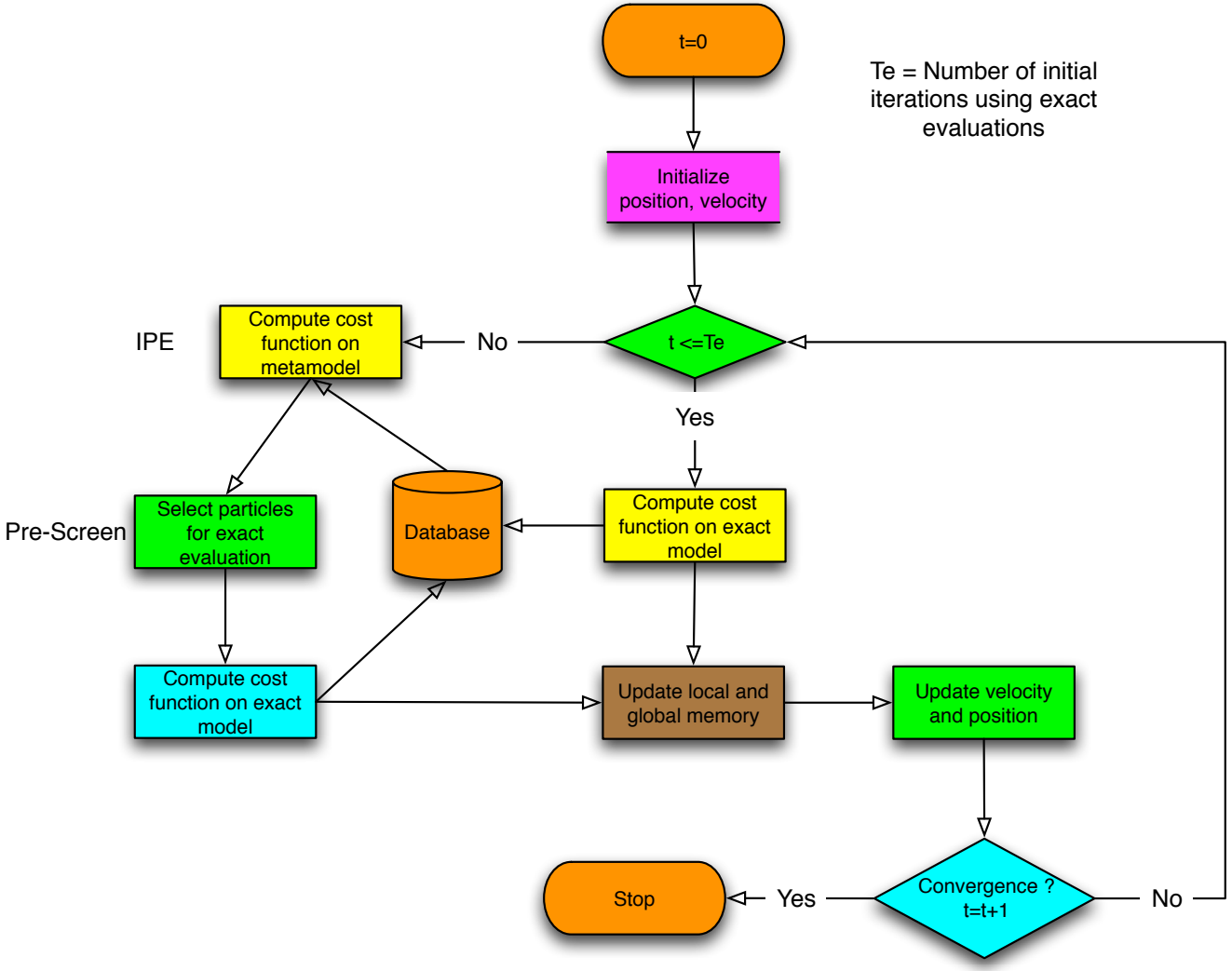
Metamodels

- Population based methods
  - Large amount of information
  - Information lost in subsequent iterations
- Save information in a database
- Use database to construct (meta) models
- Inexact Pre-Evaluation (IPE) (Giannakoglou et al.) with Genetic Algorithm

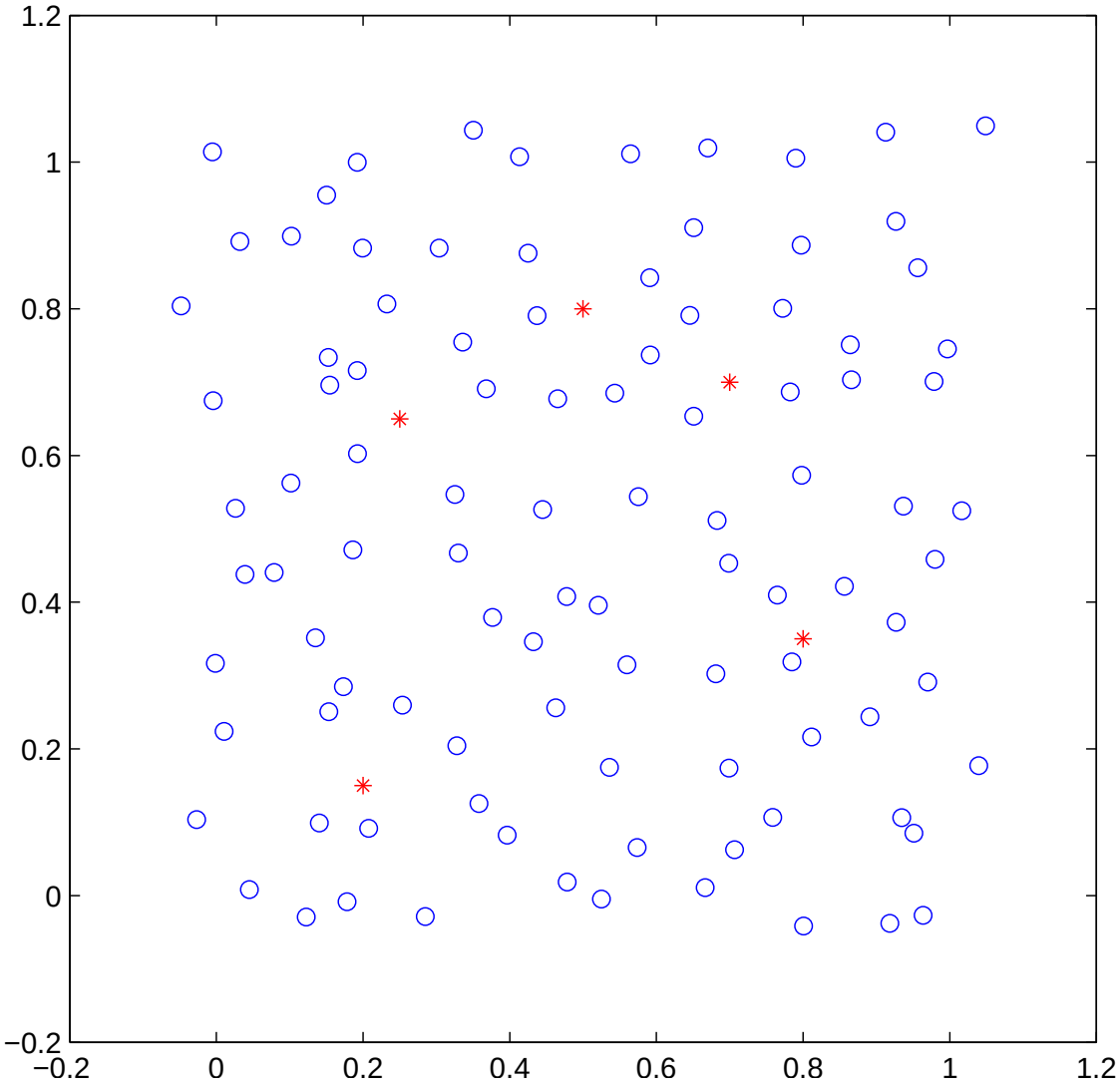
In this work

Combine IPE with PSO for aerodynamic shape design

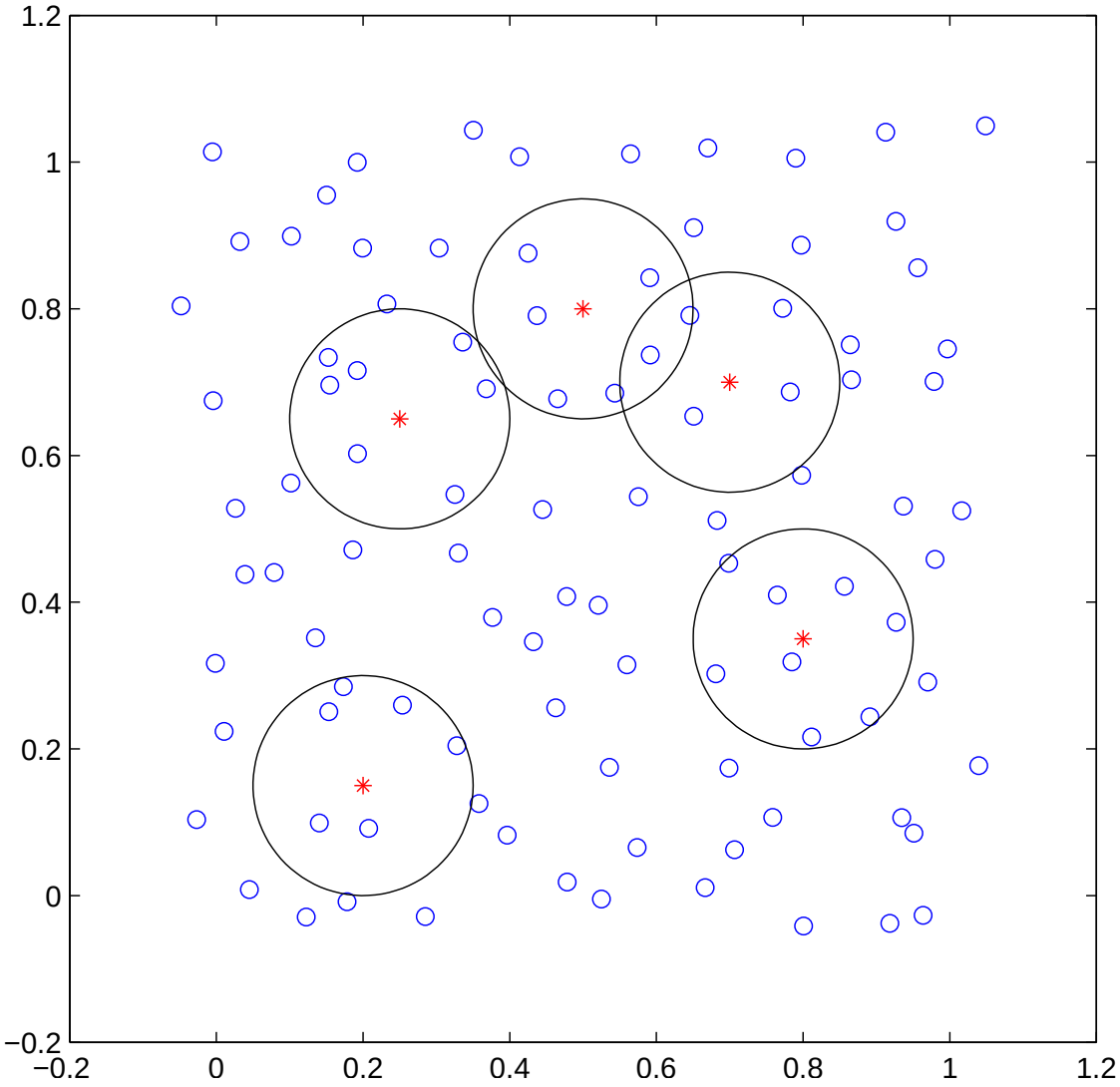
Metamodel-assisted PSO



Database and IPE



Local dataset



PSO with IPE

$$
\begin{array}{cccccccc}
x_1^n & x_2^n & x_3^n & x_4^n & x_5^n & x_6^n & x_7^n & x_8^n \\
\downarrow & \downarrow & \downarrow & \downarrow & \downarrow & \downarrow & \downarrow & \downarrow \\
\tilde{\mathcal{J}}(x_1^n) & \tilde{\mathcal{J}}(x_2^n) & \tilde{\mathcal{J}}(x_3^n) & \tilde{\mathcal{J}}(x_4^n) & \tilde{\mathcal{J}}(x_5^n) & \tilde{\mathcal{J}}(x_6^n) & \tilde{\mathcal{J}}(x_7^n) & \tilde{\mathcal{J}}(x_8^n) \\
\downarrow & \downarrow & \downarrow & \downarrow & \downarrow & \downarrow & \downarrow & \downarrow \\
\tilde{\mathcal{J}}(x_1^n) & \mathcal{J}(x_2^n) & \tilde{\mathcal{J}}(x_3^n) & \tilde{\mathcal{J}}(x_4^n) & \mathcal{J}(x_5^n) & \tilde{\mathcal{J}}(x_6^n) & \tilde{\mathcal{J}}(x_7^n) & \tilde{\mathcal{J}}(x_8^n) \\
\downarrow & \downarrow & \downarrow & \downarrow & \downarrow & \downarrow & \downarrow & \downarrow \\
\hline
v_1^{n+1} & v_2^{n+1} & v_3^{n+1} & v_4^{n+1} & v_5^{n+1} & v_6^{n+1} & v_7^{n+1} & v_8^{n+1} \\
\downarrow & \downarrow & \downarrow & \downarrow & \downarrow & \downarrow & \downarrow & \downarrow \\
x_1^{n+1} & x_2^{n+1} & x_3^{n+1} & x_4^{n+1} & x_5^{n+1} & x_6^{n+1} & x_7^{n+1} & x_8^{n+1}
\end{array}
$$

PSO with IPE

$$
\begin{array}{cccccccc}
x_1^n & x_2^n & x_3^n & x_4^n & x_5^n & x_6^n & x_7^n & x_8^n \\
\downarrow & \downarrow & \downarrow & \downarrow & \downarrow & \downarrow & \downarrow & \downarrow \\
\tilde{\mathcal{J}}(x_1^n) & \tilde{\mathcal{J}}(x_2^n) & \tilde{\mathcal{J}}(x_3^n) & \tilde{\mathcal{J}}(x_4^n) & \tilde{\mathcal{J}}(x_5^n) & \tilde{\mathcal{J}}(x_6^n) & \tilde{\mathcal{J}}(x_7^n) & \tilde{\mathcal{J}}(x_8^n) \\
\downarrow & \downarrow & \downarrow & \downarrow & \downarrow & \downarrow & \downarrow & \downarrow \\
\tilde{\mathcal{J}}(x_1^n) & \mathcal{J}(x_2^n) & \tilde{\mathcal{J}}(x_3^n) & \tilde{\mathcal{J}}(x_4^n) & \mathcal{J}(x_5^n) & \tilde{\mathcal{J}}(x_6^n) & \tilde{\mathcal{J}}(x_7^n) & \tilde{\mathcal{J}}(x_8^n) \\
\downarrow & \downarrow & \downarrow & \downarrow & \downarrow & \downarrow & \downarrow & \downarrow \\
\hline
v_1^{n+1} & v_2^{n+1} & v_3^{n+1} & v_4^{n+1} & v_5^{n+1} & v_6^{n+1} & v_7^{n+1} & v_8^{n+1} \\
\downarrow & \downarrow & \downarrow & \downarrow & \downarrow & \downarrow & \downarrow & \downarrow \\
x_1^{n+1} & x_2^{n+1} & x_3^{n+1} & x_4^{n+1} & x_5^{n+1} & x_6^{n+1} & x_7^{n+1} & x_8^{n+1}
\end{array}
$$

PSO with IPE

$$
\begin{array}{cccccccc}
x_1^n & x_2^n & x_3^n & x_4^n & x_5^n & x_6^n & x_7^n & x_8^n \\
\downarrow & \downarrow & \downarrow & \downarrow & \downarrow & \downarrow & \downarrow & \downarrow \\
\tilde{\mathcal{J}}(x_1^n) & \tilde{\mathcal{J}}(x_2^n) & \tilde{\mathcal{J}}(x_3^n) & \tilde{\mathcal{J}}(x_4^n) & \tilde{\mathcal{J}}(x_5^n) & \tilde{\mathcal{J}}(x_6^n) & \tilde{\mathcal{J}}(x_7^n) & \tilde{\mathcal{J}}(x_8^n) \\
\downarrow & \downarrow & \downarrow & \downarrow & \downarrow & \downarrow & \downarrow & \downarrow \\
\tilde{\mathcal{J}}(x_1^n) & \mathcal{J}(x_2^n) & \tilde{\mathcal{J}}(x_3^n) & \tilde{\mathcal{J}}(x_4^n) & \mathcal{J}(x_5^n) & \tilde{\mathcal{J}}(x_6^n) & \tilde{\mathcal{J}}(x_7^n) & \tilde{\mathcal{J}}(x_8^n) \\
\downarrow & \downarrow & \downarrow & \downarrow & \downarrow & \downarrow & \downarrow & \downarrow \\
\hline
v_1^{n+1} & v_2^{n+1} & v_3^{n+1} & v_4^{n+1} & v_5^{n+1} & v_6^{n+1} & v_7^{n+1} & v_8^{n+1} \\
\downarrow & \downarrow & \downarrow & \downarrow & \downarrow & \downarrow & \downarrow & \downarrow \\
x_1^{n+1} & x_2^{n+1} & x_3^{n+1} & x_4^{n+1} & x_5^{n+1} & x_6^{n+1} & x_7^{n+1} & x_8^{n+1}
\end{array}
$$

Pre-screening criteria

- Best particles criterion

$$\underbrace{\tilde{J}(x_{i_1}^t) < \tilde{J}(x_{i_2}^t) < \ldots < \tilde{J}(x_{i_M}^t)}_{S^t = \text{Best M particles}} < \ldots$$

  M: specified by the user

- Adaptive pre-screening criterion
  - p_i = local memory of i'th particle
  - Evaluate $J(x_i^t)$ if local memory p_i is predicted to improve

$$\tilde{J}(x_i^t) < J(p_i^{t-1})$$

$$S^t = \{x_i^t : \tilde{J}(x_i^t) < J(p_i^{t-1})\}$$

Pre-screening criteria

- Best particles criterion

$$\underbrace{\tilde{J}(x_{i_1}^t) < \tilde{J}(x_{i_2}^t) < \ldots < \tilde{J}(x_{i_M}^t)}_{S^t = \text{Best M particles}} < \ldots$$

  M: specified by the user

- Adaptive pre-screening criterion
  - p_i = local memory of i'th particle
  - Evaluate $J(x_i^t)$ if local memory p_i is predicted to improve

$$\tilde{J}(x_i^t) < J(p_i^{t-1})$$

$$S^t = \{x_i^t : \tilde{J}(x_i^t) < J(p_i^{t-1})\}$$

Pre-screening criteria

- Best particles criterion

$$\underbrace{\tilde{J}(x_{i_1}^t) < \tilde{J}(x_{i_2}^t) < \ldots < \tilde{J}(x_{i_M}^t)}_{S^t = \text{Best M particles}} < \ldots$$

  M: specified by the user

- Adaptive pre-screening criterion
  - p_i = local memory of i'th particle
  - Evaluate $J(x_i^t)$ if local memory p_i is predicted to improve

$$\tilde{J}(x_i^t) < J(p_i^{t-1})$$

$$S^t = \{x_i^t : \tilde{J}(x_i^t) < J(p_i^{t-1})\}$$

RBF: Basic method

- Given data

$$(x_1, f_1), (x_2, f_2), \ldots, (x_N, f_N)$$

$$x_i \in \mathbb{R}^d$$
$$f_i = f(x_i) \in \mathbb{R}$$

- RBF approximation

$$\hat{f}(x) = \sum_{n=1}^{N} w_n \phi(\|x - x_n\|)$$

- Example: *Gaussian RBF*

$$\phi : \mathbb{R} \mapsto \mathbb{R}$$

$$\phi(r) = \exp(-r^2/a^2)$$

RBF: Basic method

- Given data

$$(x_1, f_1), (x_2, f_2), \ldots, (x_N, f_N)$$

$$x_i \in \mathbb{R}^d$$
$$f_i = f(x_i) \in \mathbb{R}$$

- RBF approximation

$$\hat{f}(x) = \sum_{n=1}^{N} w_n \phi(\|x - x_n\|)$$

- Example: *Gaussian RBF*

$$\phi : \mathbb{R} \mapsto \mathbb{R}$$

$$\phi(r) = \exp(-r^2/a^2)$$

RBF: Basic method

- Given data

$$(x_1, f_1), (x_2, f_2), \ldots, (x_N, f_N)$$

$$x_i \in \mathbb{R}^d$$
$$f_i = f(x_i) \in \mathbb{R}$$

- RBF approximation

$$\hat{f}(x) = \sum_{n=1}^{N} w_n \phi(\|x - x_n\|)$$

- Example: *Gaussian RBF*

$$\phi : \mathbb{R} \mapsto \mathbb{R}$$

$$\phi(r) = \exp(-r^2/a^2)$$

RBF interpolant

- Find unknown coefficients w_n in

$$\hat{f}(x) = \sum_{n=1}^{N} w_n \phi(\|x - x_n\|)$$

from interpolation conditions

$$\hat{f}(x_n) = f(x_n) = f_n, \quad 1 \leq n \leq N$$

Test case: Supersonic Business Jet

$$\min C_d$$

$$C_l \geq C_{l_o} \quad \text{Lift constraint}$$

$$V \geq V_o \quad \text{Volume constraint}$$

Thickness constraint

Test case: Supersonic Business Jet

- Mach = 1.7
- Angle of attack = 1 deg.
- Minimize drag subject to constraint on lift and volume

Cost function

$$\mathcal{J} = \frac{C_d}{C_{d_o}} + 10^4 \max\left(0, 0.999 - \frac{C_l}{C_{l_o}}\right) + 10^3 \max\left(0, \frac{V_o - V}{V_o}\right) + I_p$$

where

- C_d = drag coefficient
- C_l = lift coefficient
- V = volume of the wing
- I_p = a penalty term to control the thickness

Test case: Supersonic Business Jet

- Mach = 1.7
- Angle of attack = 1 deg.
- Minimize drag subject to constraint on lift and volume

Cost function

$$\mathcal{J} = \frac{C_d}{C_{d_o}} + 10^4 \max\left(0, 0.999 - \frac{C_l}{C_{l_o}}\right) + 10^3 \max\left(0, \frac{V_o - V}{V_o}\right) + I_p$$

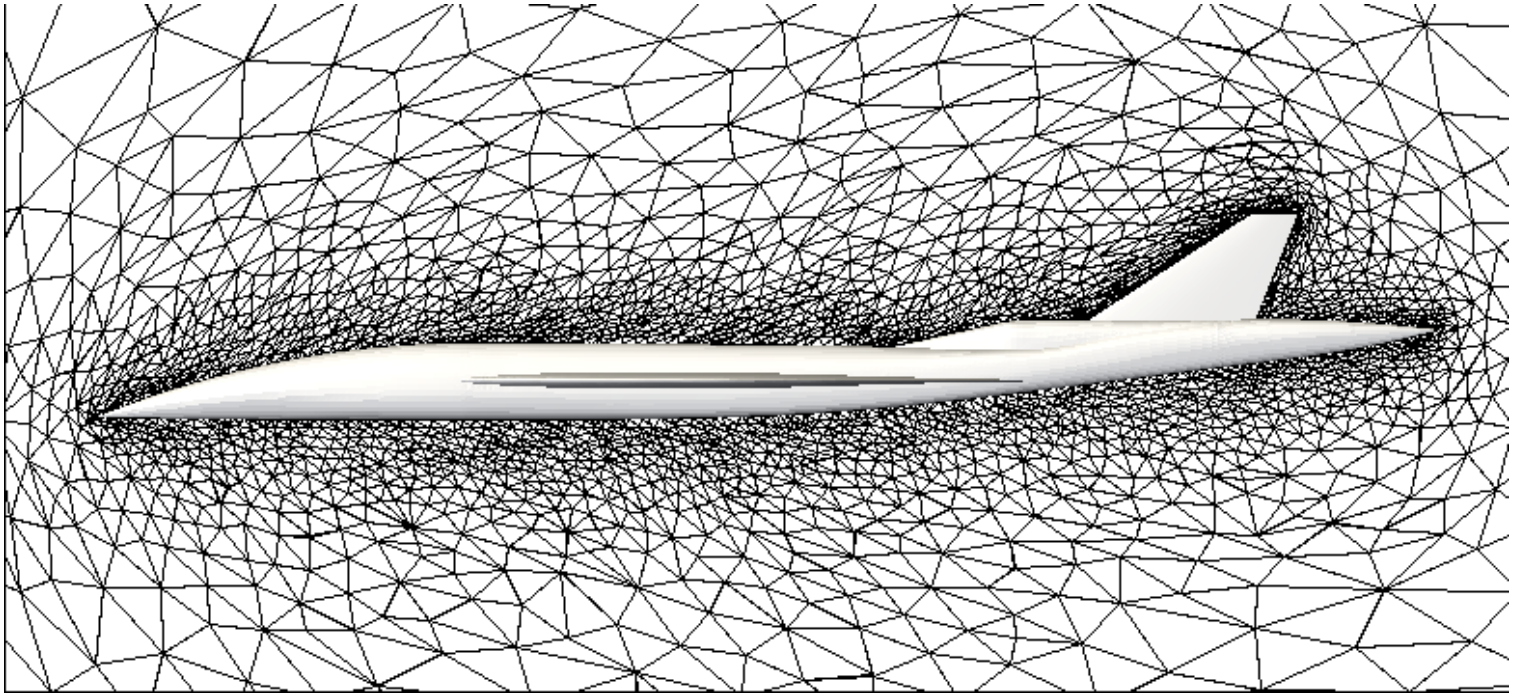
where

- C_d = drag coefficient
- C_l = lift coefficient
- V = volume of the wing
- I_p = a penalty term to control the thickness

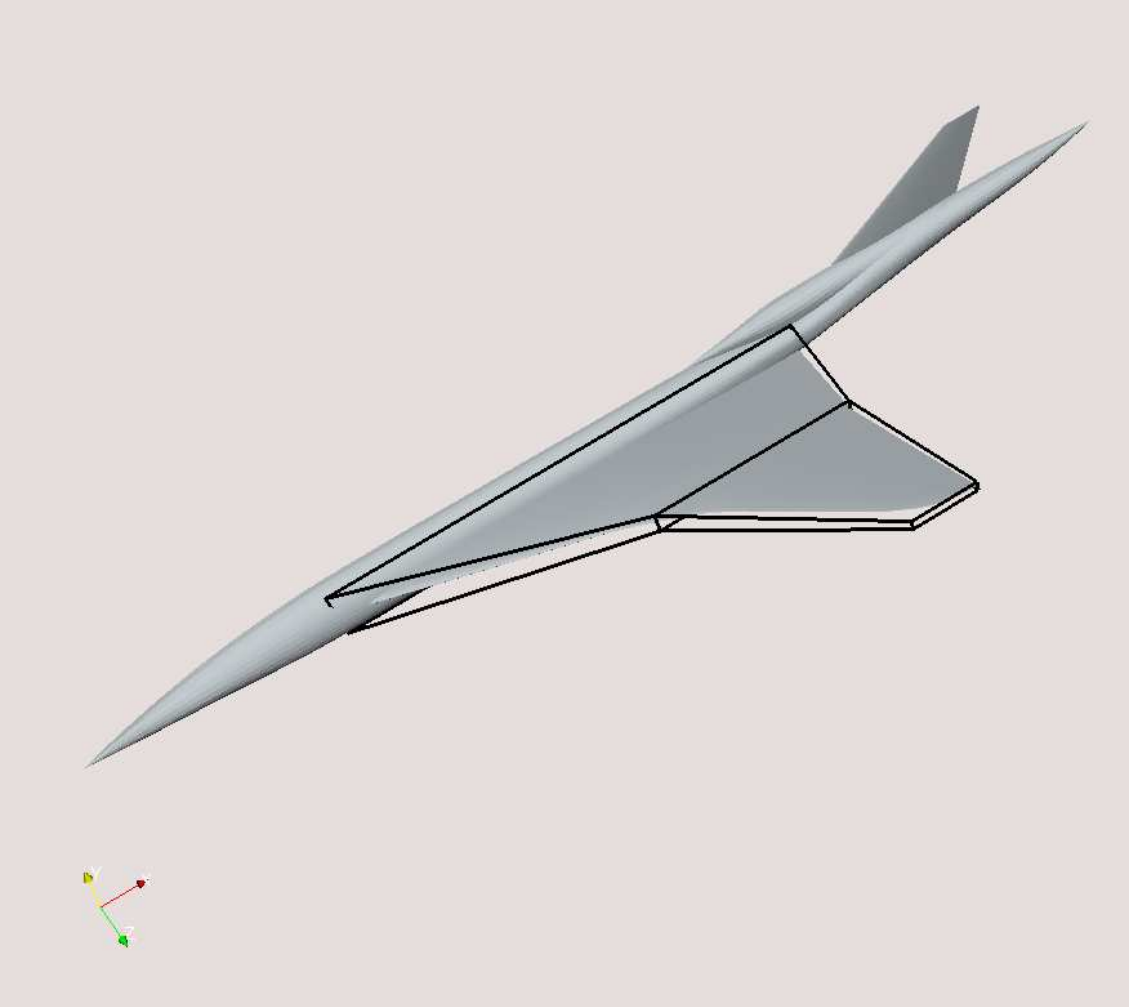
CFD solver: NS3D

- Inviscid flow model
- Unstructured finite volume solver
- Vertex-based scheme
- Roe's Riemann solver

- Second order through MUSCL-type reconstruction
- Van Albada limiter
- Supersonic Business Jet
  - 37375 nodes
  - 184249 tetrahedra



SSBJ: FFD parameterization

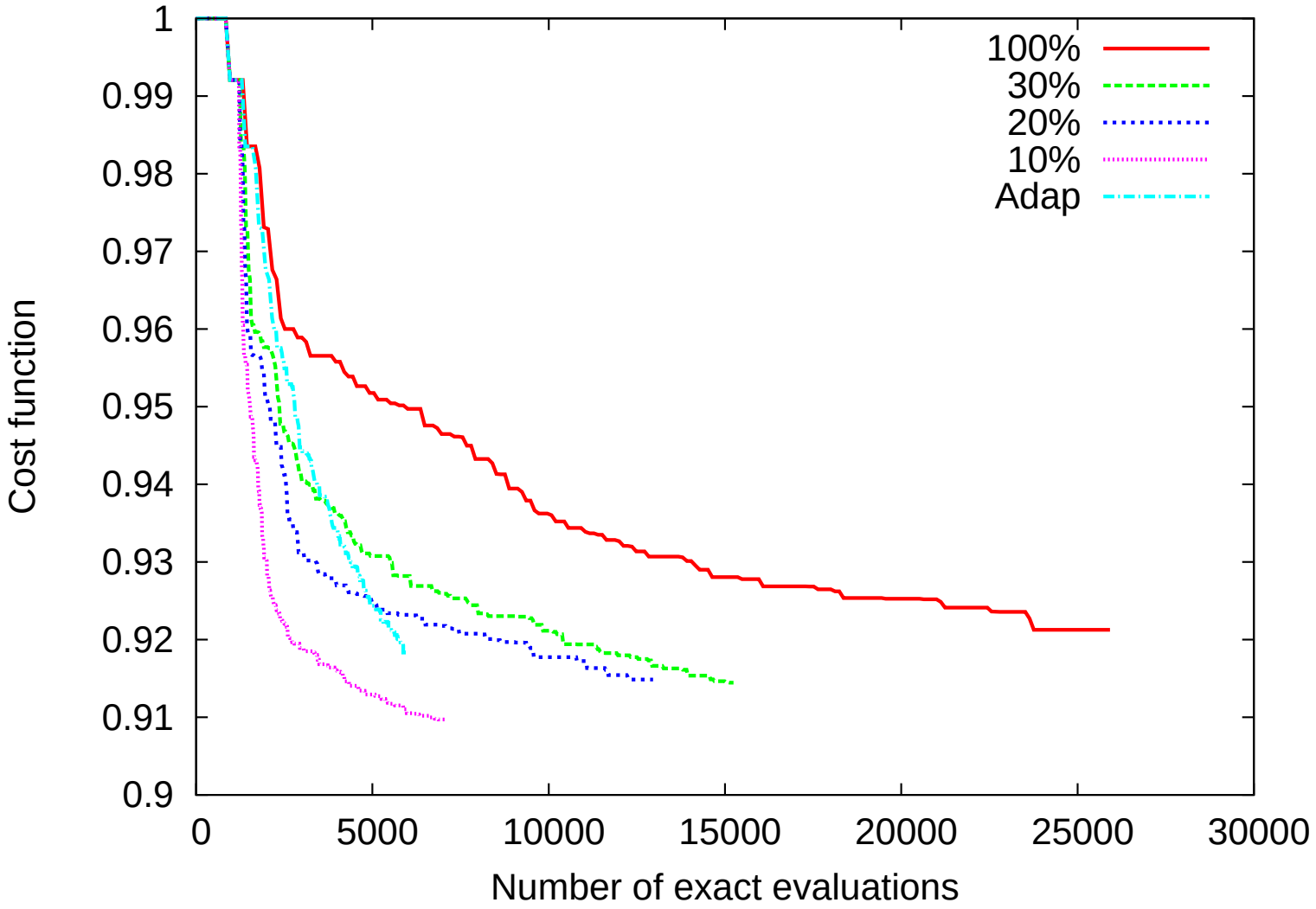


$$n_i \times n_j \times n_k = 6 \times 1 \times 2 \Longrightarrow 20 \text{ variables}$$

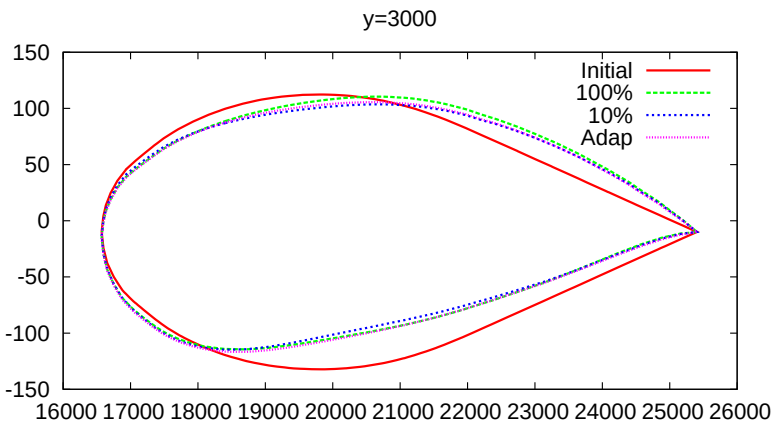
SSBJ: Metamodel-assisted PSO

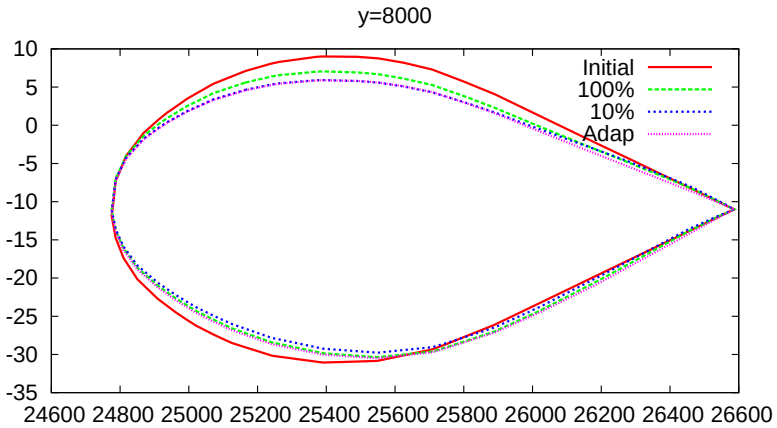
	Cost	# CFD	Effort
Initial	1.0	-	-
100% CFD	0.9212	25920	100%
30% CFD	0.9144	15240	59%
20% CFD	0.9148	12960	50%
10% CFD	0.9097	7080	27%
Adaptive	0.9183	6002	23%

SSBJ: Cost function



SSBJ: Wing shape

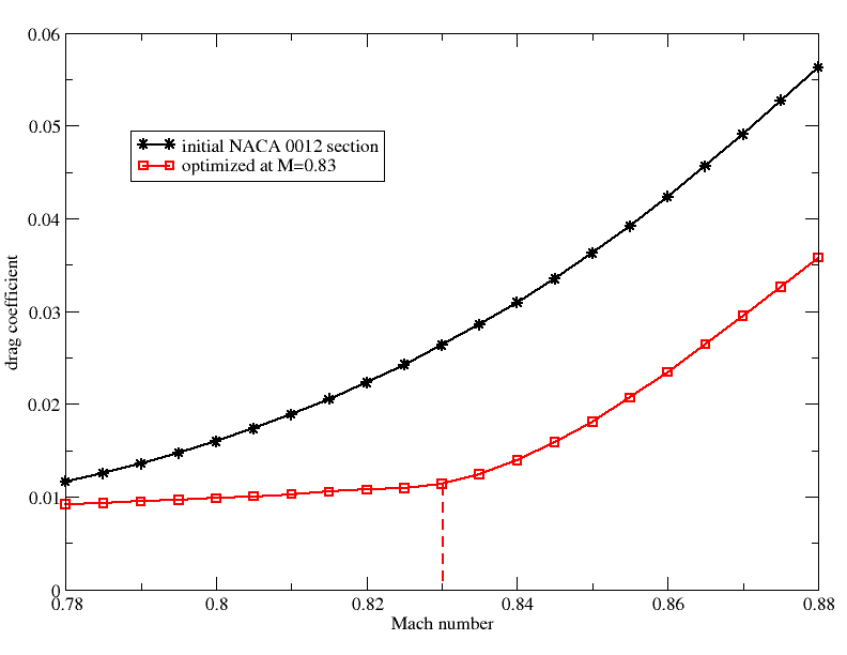


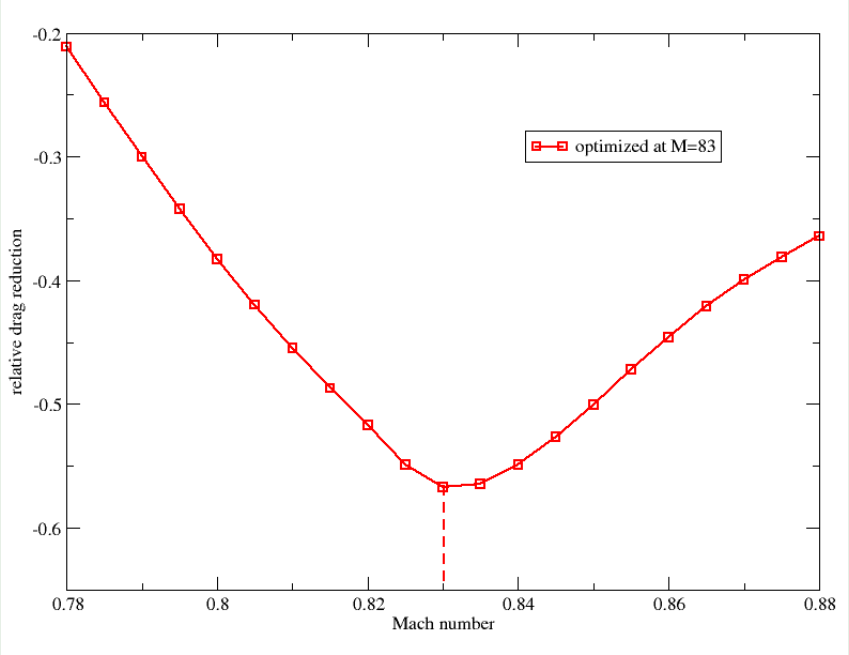


Robust design

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

Example : effect of Mach number fluctuations





Optimization problem

Optimization

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

Robust optimization

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

Optimization problem

Optimization

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

Robust optimization

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

Monte-Carlo estimation using meta-models

- For each design x: compute

$$\mathcal{J}(x, a_1), \mathcal{J}(x, a_2), \ldots, \mathcal{J}(x, a_P)$$

$$\Downarrow$$

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

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

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

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

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

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

Monte-Carlo estimation using meta-models

- For each design x: compute

$$\mathcal{J}(x, a_1), \mathcal{J}(x, a_2), \ldots, \mathcal{J}(x, a_P)$$

$$\Downarrow$$

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

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

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

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

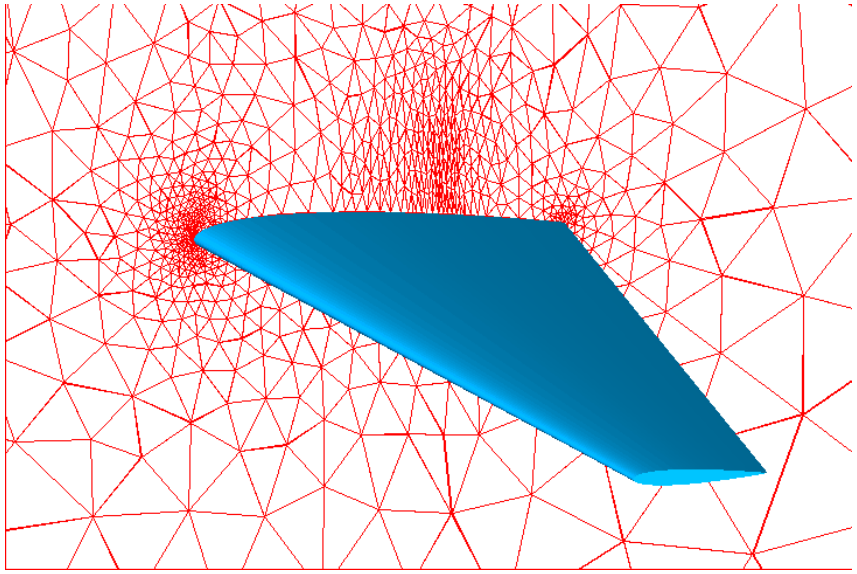
Robust optimization problem: $0 \leq \omega \leq 1$

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

Test case: Wing shape optimization

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

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

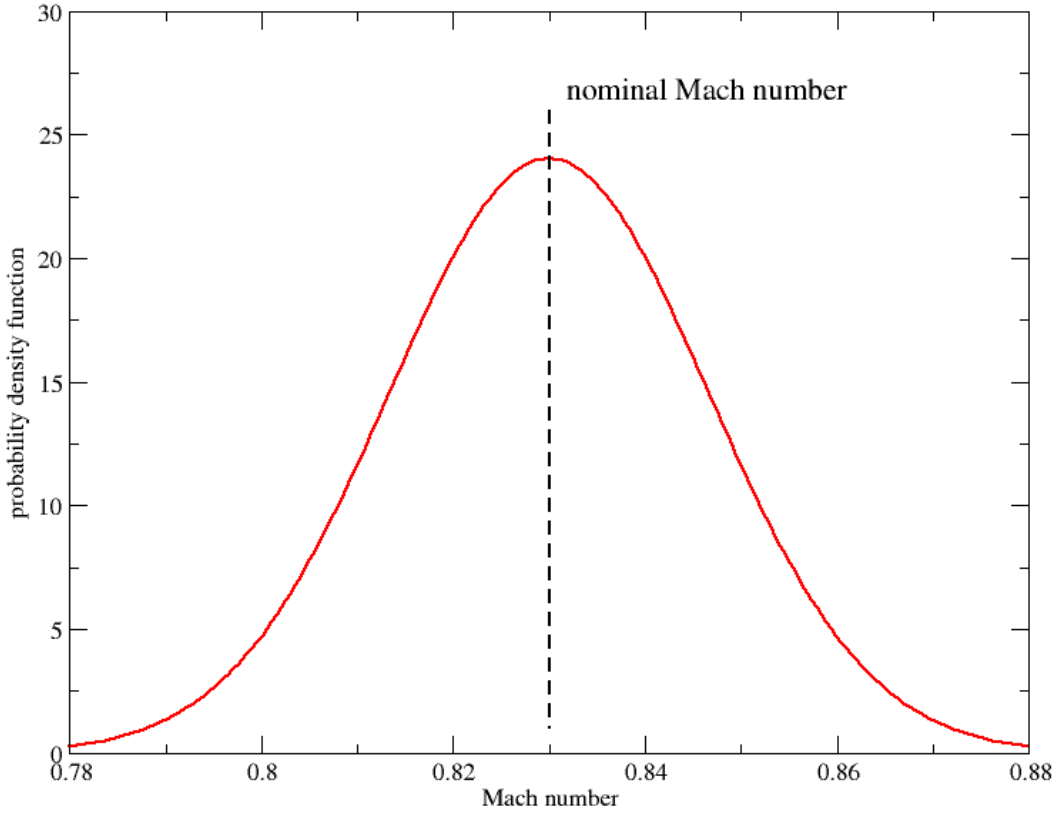


(Piaggio Aero. Ind.)
Grid: 31124 nodes

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

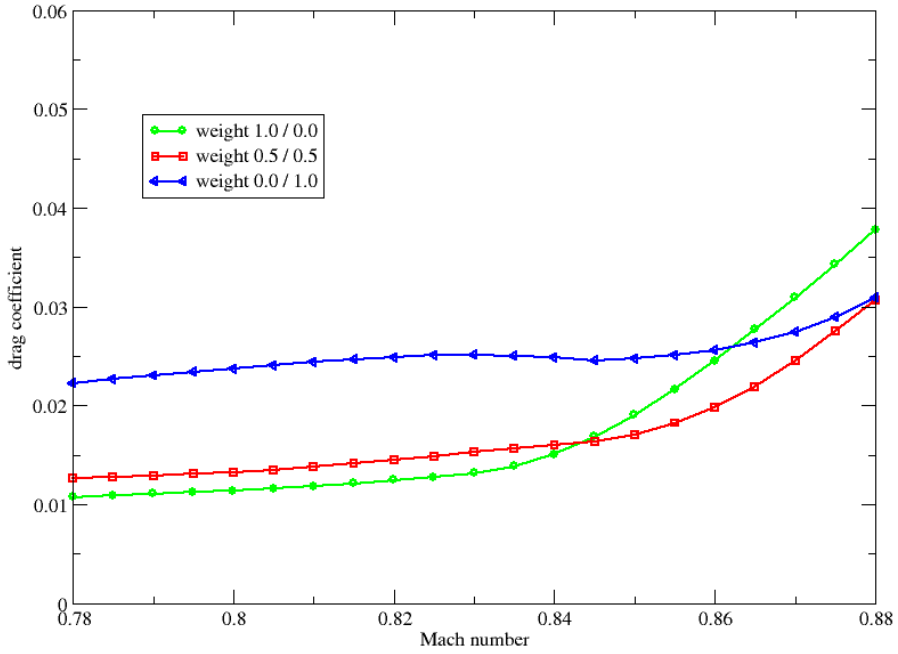
PDF of Mach number

Mean = 0.83, std. dev. = 0.0166

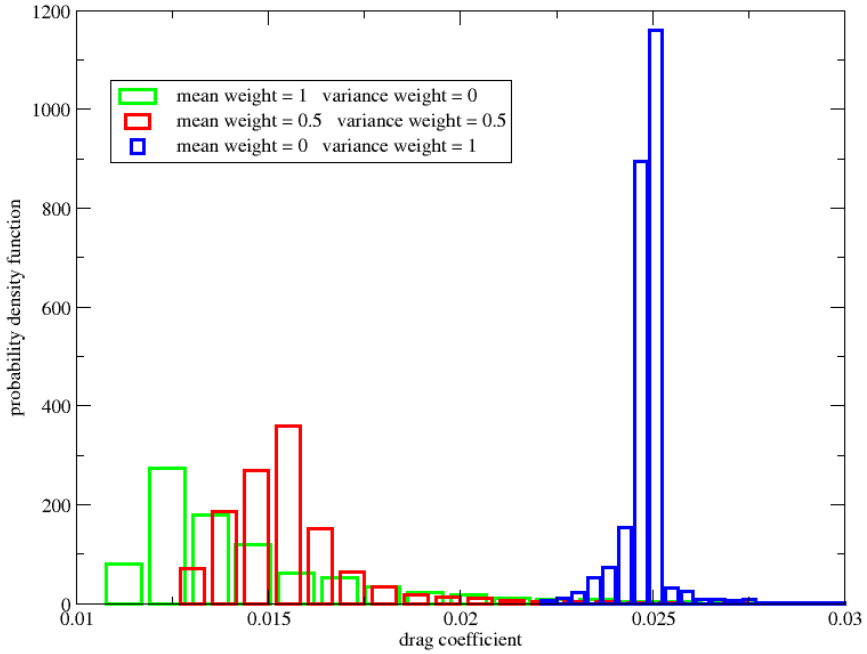


Wing optimization: result

Drag vs Mach number

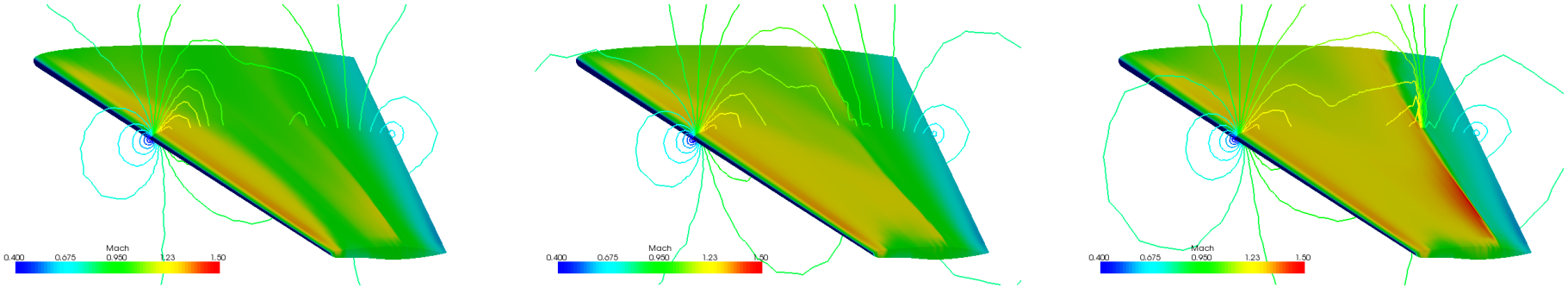


PDF of drag

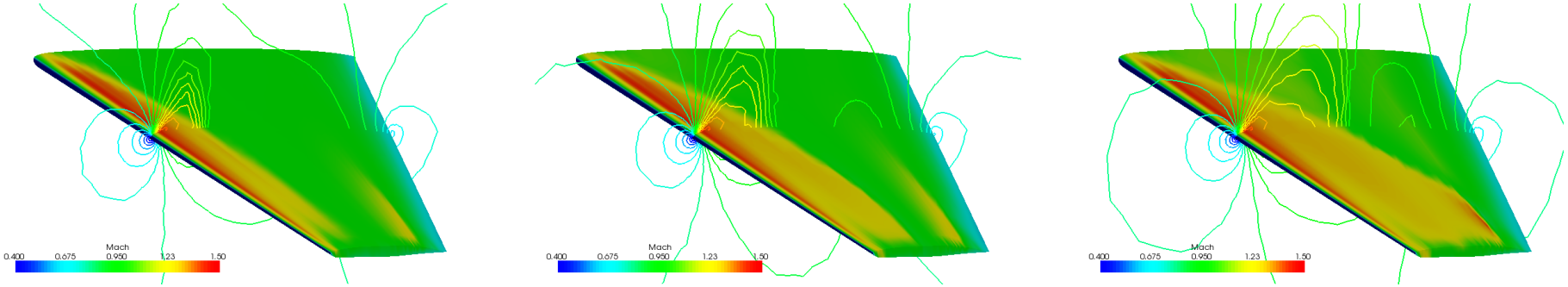


Wing optimization: flow solutions

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)

End

Joint work with
Regis Duvigneau
Projet OPALE
INRIA, Sophia Antipolis

Email:
praveen@math.tifrbng.res.in