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

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

http://www.aviation-history.com
http://www.centennialofflight.gov
Effect of Mach number: \( M = \text{fluid speed/sound speed} \)

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

![Graph showing the comparison between RAE2822 and its optimized version. The graph plots the coefficient of pressure ($C_p$) against the nondimensional streamwise coordinate ($x$). The solid line represents RAE2822, and the dashed line represents the optimized version.]
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 $\Rightarrow$ 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
  \[ \min_{S \in S_{ad}} J(S, u), \quad \text{s.t.} \quad N(S, u) = 0 \]

\( S_{ad} = \) Set of admissible shapes
Let the shape $S$ be given by a function $F(x)$.

Variation in cost function: $F \rightarrow F + \delta F$

$$\delta J = \int G(x) \delta F(x) dx$$

$G$: Shape gradient

Steepest descent: choose

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

$$\delta J = -\lambda \int G^2(x) dx \leq 0$$
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 $\|G^n\| < \text{TOL}$, 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

\[
X(P) = X^o(P) + \sum_{i=0}^{n_i} \sum_{j=0}^{n_j} \sum_{k=0}^{n_k} B^n_i(\xi_p)B^n_j(\eta_p)B^n_k(\zeta_p)Y_{ijk}
\]

- Bernstein polynomials

\[
B^n_p(t) = C^n_p t^p (1 - t)^{n-p}
\]

- Design variables

\[
\{ Y_{ijk} \}, \quad 0 \leq i \leq n_i, \quad 0 \leq j \leq n_j, \quad 0 \leq k \leq n_k
\]
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

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

- \(\rho\) : fluid density
- \((u, v, w)\) : fluid velocity
- \(E\) : energy per unit volume

- \((E, F, G) = \text{flux vector}\)
Integral form

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

$$\frac{d}{dt} \int_V U \, dx + \int_{\partial V} (E_{nx} + F_{ny} + G_{nz}) \, dS = 0$$

- Divide $\Omega$ into disjoint polygonal *finite volumes*

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

- Cell average value

$$U_i(t) = \frac{1}{|C_i|} \int_{C_i} U(x, t) \, dx$$
Cell connectivity: $N(i) = \{ j : C_j \text{ and } C_i \text{ share a common face} \}$

\[
\oint_{\partial C_i} (E_n x + F_n y + G_n z) dS = \sum_{j \in N(i)} \int_{C_i \cap C_j} (E_n x + F_n y + G_n z) dS
\]
Finite volume method

- Approximate flux integral by quadrature

\[ \int_{C_i \cap C_j} (E_{n_x} + F_{n_y} + G_{n_z})dS \approx H_{ij}\Delta S_{ij} \]

- Semi-discrete update equation (ODE)

\[ |C_i| \frac{dU_i}{dt} + \sum_{j \in N(i)} H_{ij}\Delta S_{ij} = 0, \quad H_{ij} = H(U_i, U_j, \hat{n}_{ij}) \]
FVM Algorithm: Explicit, steady-state

Find $U$ such that

$$\frac{\partial E}{\partial x} + \frac{\partial F}{\partial y} + \frac{\partial G}{\partial z} = 0$$

1. Set $n = 0$, initial condition $U^0$, choose time step $\Delta t$
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 $\|(U^{n+1} - U^n)/\Delta t\| > \text{TOL}$, 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

- 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 : $p^t_i = \arg\min_{0 \leq s \leq t} J(x^s_i)$

  Global memory : $p^t = \arg\min_i J(p^t_i)$

- Velocity update

  $v^{t+1}_i = \omega v^t_i + c_1 r_1(p^t_i - x^t_i) + c_2 r_2(p^t - x^t_i)$

- Position update

  $x^{t+1}_i = x^t_i + v^{t+1}_i$
Basic PSO algorithm

1. Initialize position, velocity
2. Compute cost function
3. Update local and global memory
4. Update velocity and position
5. Convergence?
   - Yes: Stop
   - No: t = t + 1
PSO: Parallelizability

Parallel evaluation of cost functions using MPI: Message Passing Interface
Test case: Wing shape optimization

- Minimize drag under lift constraint
  \[
  \min \frac{C_d}{C_{d_0}} \quad \text{s.t.} \quad \frac{C_l}{C_{l_0}} \geq 0.999
  \]

- 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

\[
J = \frac{C_d}{C_{d_0}} + 10^4 \max \left( 0, 0.999 - \frac{C_l}{C_{l_0}} \right)
\]
Test case: Wing shape optimization

- Minimize drag under lift constraint
  \[
  \min \frac{C_d}{C_{d_0}} \quad \text{s.t.} \quad \frac{C_l}{C_{l_0}} \geq 0.999
  \]

- 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

\[
J = \frac{C_d}{C_{d_0}} + 10^4 \max \left( 0, 0.999 - \frac{C_l}{C_{l_0}} \right)
\]
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)

$$(10 \text{ minutes/CFD}) (120 \text{ CFD/iteration}) (200 \text{ iterations}) = 4000 \text{ hours}$$
Metamodels

- **Expensive model**
  - Shape parameters → Surface grid → Volume grid → CFD solution → $J, C$

- Replace costly model with cheap model:
  - **metamodel** or **surrogate model**
  - Shape parameters → $\tilde{J}, \tilde{C}$

- **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

$t=0$
- Initialize position, velocity

Compute cost function on exact model
- Update local and global memory

Update velocity and position

Convergence?
- $t=t+1$
- Yes
- Stop

$IPE$
- Select particles for exact evaluation

Database
- Compute cost function on exact model

Compute cost function on metamodel
- IPE
- No

$t <= Te$
- Yes

$t=0$
- Initialize position, velocity

Te = Number of initial iterations using exact evaluations

Pre-Screen
- Compute cost function on exact model

Update local and global memory
Local dataset
PSO with IPE

\[
\begin{align*}
&x_1^n \\
&\downarrow \\
&\tilde{J}(x_1^n) \\
&\downarrow \\
&\tilde{J}(x_1^n) \\
&\downarrow \\
&v_1^{n+1} \\
&\downarrow \\
&x_1^{n+1} \\

&x_2^n \\
&\downarrow \\
&\tilde{J}(x_2^n) \\
&\downarrow \\
&J(x_2^n) \\
&\downarrow \\
&v_2^{n+1} \\
&\downarrow \\
&x_2^{n+1} \\

&x_3^n \\
&\downarrow \\
&\tilde{J}(x_3^n) \\
&\downarrow \\
&J(x_3^n) \\
&\downarrow \\
&v_3^{n+1} \\
&\downarrow \\
&x_3^{n+1} \\

&x_4^n \\
&\downarrow \\
&\tilde{J}(x_4^n) \\
&\downarrow \\
&J(x_4^n) \\
&\downarrow \\
&v_4^{n+1} \\
&\downarrow \\
&x_4^{n+1} \\

&x_5^n \\
&\downarrow \\
&\tilde{J}(x_5^n) \\
&\downarrow \\
&J(x_5^n) \\
&\downarrow \\
&v_5^{n+1} \\
&\downarrow \\
&x_5^{n+1} \\

&x_6^n \\
&\downarrow \\
&\tilde{J}(x_6^n) \\
&\downarrow \\
&J(x_6^n) \\
&\downarrow \\
&v_6^{n+1} \\
&\downarrow \\
&x_6^{n+1} \\

&x_7^n \\
&\downarrow \\
&\tilde{J}(x_7^n) \\
&\downarrow \\
&J(x_7^n) \\
&\downarrow \\
&v_7^{n+1} \\
&\downarrow \\
&x_7^{n+1} \\

&x_8^n \\
&\downarrow \\
&\tilde{J}(x_8^n) \\
&\downarrow \\
&J(x_8^n) \\
&\downarrow \\
&v_8^{n+1} \\
&\downarrow \\
&x_8^{n+1}
\end{align*}
\]
PSO with IPE
PSO with IPE

<table>
<thead>
<tr>
<th>$x_1^n$</th>
<th>$x_2^n$</th>
<th>$x_3^n$</th>
<th>$x_4^n$</th>
<th>$x_5^n$</th>
<th>$x_6^n$</th>
<th>$x_7^n$</th>
<th>$x_8^n$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\tilde{J}(x_1^n)$</td>
<td>$\tilde{J}(x_2^n)$</td>
<td>$\tilde{J}(x_3^n)$</td>
<td>$\tilde{J}(x_4^n)$</td>
<td>$\tilde{J}(x_5^n)$</td>
<td>$\tilde{J}(x_6^n)$</td>
<td>$\tilde{J}(x_7^n)$</td>
<td>$\tilde{J}(x_8^n)$</td>
</tr>
<tr>
<td>$\tilde{J}(x_1^n)$</td>
<td>$\tilde{J}(x_2^n)$</td>
<td>$\tilde{J}(x_3^n)$</td>
<td>$\tilde{J}(x_4^n)$</td>
<td>$\tilde{J}(x_5^n)$</td>
<td>$\tilde{J}(x_6^n)$</td>
<td>$\tilde{J}(x_7^n)$</td>
<td>$\tilde{J}(x_8^n)$</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>$v_1^{n+1}$</th>
<th>$v_2^{n+1}$</th>
<th>$v_3^{n+1}$</th>
<th>$v_4^{n+1}$</th>
<th>$v_5^{n+1}$</th>
<th>$v_6^{n+1}$</th>
<th>$v_7^{n+1}$</th>
<th>$v_8^{n+1}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$x_1^{n+1}$</td>
<td>$x_2^{n+1}$</td>
<td>$x_3^{n+1}$</td>
<td>$x_4^{n+1}$</td>
<td>$x_5^{n+1}$</td>
<td>$x_6^{n+1}$</td>
<td>$x_7^{n+1}$</td>
<td>$x_8^{n+1}$</td>
</tr>
</tbody>
</table>
Pre-screening criteria

- Best particles criterion

\[
\tilde{J}(x_{i_1}^t) < \tilde{J}(x_{i_2}^t) < \ldots < \tilde{J}(x_{i_M}^t) < \ldots
\]

\[S^t = \text{Best } M \text{ particles}\]

- \(M\): 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**

  \[ \tilde{J}(x_{i_1}^t) < \tilde{J}(x_{i_2}^t) < \ldots < \tilde{J}(x_{i_M}^t) < \ldots \]

  \[ S^t = \text{Best M particles} \]

  \( M \): 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**

\[ \tilde{J}(x_{i_1}^t) < \tilde{J}(x_{i_2}^t) < \ldots < \tilde{J}(x_{i_M}^t) < \ldots \]

\[ S^t = \text{Best M particles} \]

**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

\[\phi : \mathbb{R} \mapsto \mathbb{R}\]
\[\phi(r) = \exp(-r^2/\alpha^2)\]
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
  \[\phi : \mathbb{R} \mapsto \mathbb{R}\]
  \[\phi(r) = \exp(-r^2/a^2)\]
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

\[
\begin{align*}
\min C_d \\
C_l & \geq C_{l_0} \quad \text{Lift constraint} \\
V & \geq V_o \quad \text{Volume constraint} \\
\text{Thickness constraint}
\end{align*}
\]
Test case: Supersonic Business Jet

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

Cost function

\[ J = \frac{C_d}{C_{d_0}} + 10^4 \max \left( 0, 0.999 - \frac{C_l}{C_{l_0}} \right) + 10^3 \max \left( 0, \frac{V_o - V}{V_o} \right) + I_p \]

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

\[ J = \frac{C_d}{C_{d_0}} + 10^4 \max \left(0, 0.999 - \frac{C_l}{C_{l_0}}\right) + 10^3 \max \left(0, \frac{V_o - V}{V_o}\right) + I_p \]

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

\[ n_i \times n_j \times n_k = 6 \times 1 \times 2 \implies 20 \text{ variables} \]
<table>
<thead>
<tr>
<th></th>
<th>Cost</th>
<th># CFD</th>
<th>Effort</th>
</tr>
</thead>
<tbody>
<tr>
<td>Initial</td>
<td>1.0</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>100% CFD</td>
<td>0.9212</td>
<td>25920</td>
<td>100%</td>
</tr>
<tr>
<td>30% CFD</td>
<td>0.9144</td>
<td>15240</td>
<td>59%</td>
</tr>
<tr>
<td>20% CFD</td>
<td>0.9148</td>
<td>12960</td>
<td>50%</td>
</tr>
<tr>
<td>10% CFD</td>
<td>0.9097</td>
<td>7080</td>
<td>27%</td>
</tr>
<tr>
<td>Adaptive</td>
<td>0.9183</td>
<td>6002</td>
<td>23%</td>
</tr>
</tbody>
</table>
SSBJ: Cost function

Cost function

Number of exact evaluations

100% 30% 20% 10% Adap

Praveen. C (TIFR-CAM)  Shape optimization  IITM, Nov 2008 52 / 61
SSBJ: Wing shape

![Graph for y=3000](image)

- Initial
- 100%
- 10%
- Adap

![Graph for y=8000](image)

- Initial
- 100%
- 10%
- Adap

Praveen. C (TIFR-CAM)
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

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

Optimization

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

Robust optimization

\[
\begin{align*}
\mu_J(x) &= \int_{\Omega(A)} \mathcal{J}(x, a) \rho_A(a) \, da \\
\sigma_J^2(x) &= \int_{\Omega(A)} [\mathcal{J}(x, a) - \mu_J]^2 \rho_A(a) \, da
\end{align*}
\]
\[
\text{Prob}[\mathcal{C}(x, A) \leq 0] \geq p
\]
Optimization problem

Optimization

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

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

Robust optimization

\[
\min_{x \in \mathbb{R}^d} \left\{ \begin{array}{l}
\mu_J(x) = \int_{\Omega(A)} \mathcal{J}(x, a) \rho_A(a) \, da \\
\sigma_J^2(x) = \int_{\Omega(A)} \left[ \mathcal{J}(x, a) - \mu_J \right]^2 \rho_A(a) \, da
\end{array} \right.
\]

\[
\text{Prob}[C(x, A) \leq 0] \geq p
\]
Monte-Carlo estimation using meta-models

- For each design \( x \): compute
  \[
  J(x, a_1), J(x, a_2), \ldots, J(x, a_P)
  \]
  ↓
  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: \( 0 \leq \omega \leq 1 \)

\[
\min_{x \in \mathbb{R}^d} \omega M_J + (1 - \omega) S_J + \text{constraint penalty}
\]
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: $0 \leq \omega \leq 1$

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

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

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 $M_\infty = 0.83$

Robust design, weights (0.5, 0.5)
Joint work with
Regis Duvigneau
Projet OPALE
INRIA, Sophia Antipolis

Email:
praveen@math.tifrbng.res.in