Numerical Algorithms for Optimization and Control of PDE's Systems

Application to Fluid Dynamics

R. Duvigneau

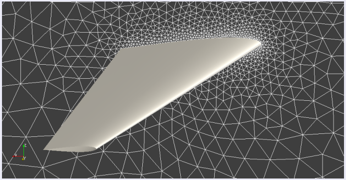
INRIA Sophia Antipolis-Méditerranée, OPALE Project-Team

Some examples and illustrations ...

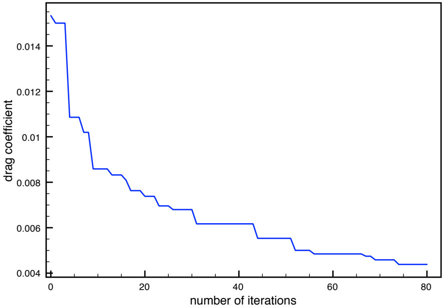
- Global optimization of the wing shape of a business aircraft
- Local optimization of the shape of an airfoil
- Identification of optimal parameters for flow control

Problem description (Piaggio-Aero P180)

- Drag minimization
- Lift constraint
- Global variables : span, root/tip chord ratio, sweep, twist and incidence angles
- Local variables : airfoil section (10 variables)
- Cruise regime $M_\infty = 0.83$
- Euler solver (Finite-Volume)
- Mesh 56,512 nodes 311,820 cells

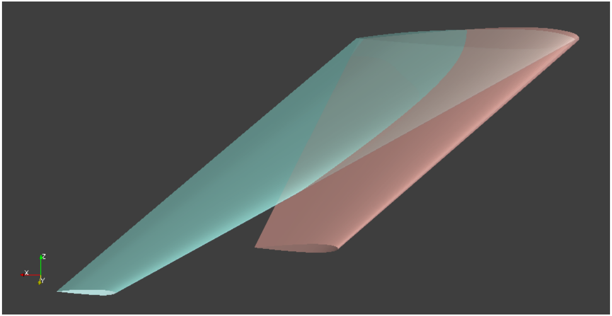


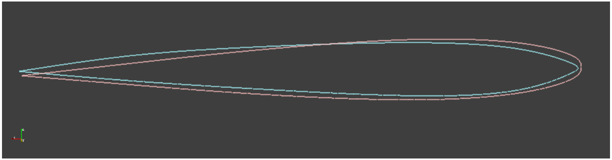
Wing design



→ Convergence in about 80 iterations ∼ 960 simulations

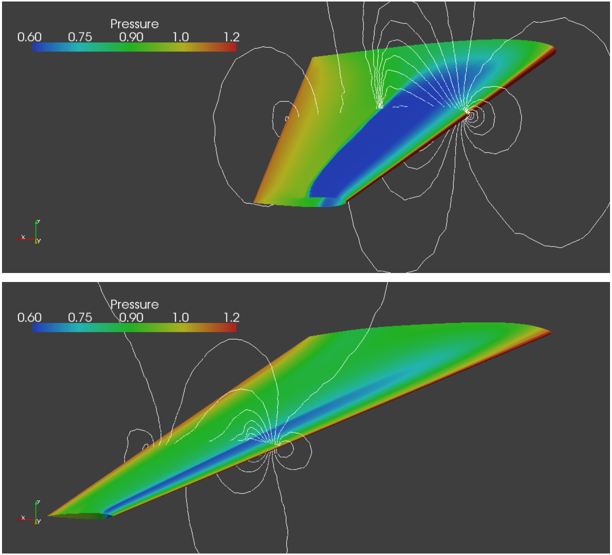
Wing design





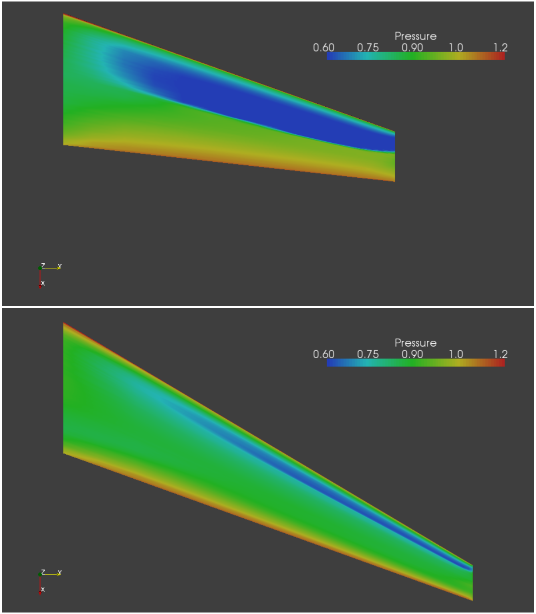
Comparison of initial (red) and final (blue) wing shapes

Wing design



Comparison of initial (top) and final (bottom) pressure fields

Wing design

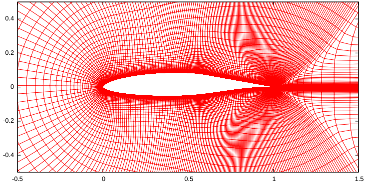


Comparison of initial (top) and final (bottom) planforms

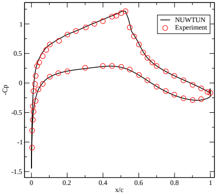
Airfoil modification

Problem description

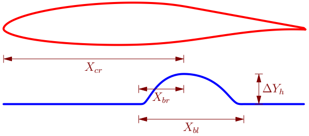
- Navier-Stokes, $k - \omega$ turbulence modeling
- Reynolds number $Re = 19\ 10^6$, Mach number 0.68
- Drag reduction



Computational mesh: 353×97 nodes

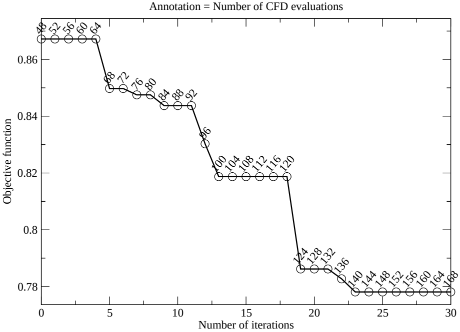


Comparison with experiments



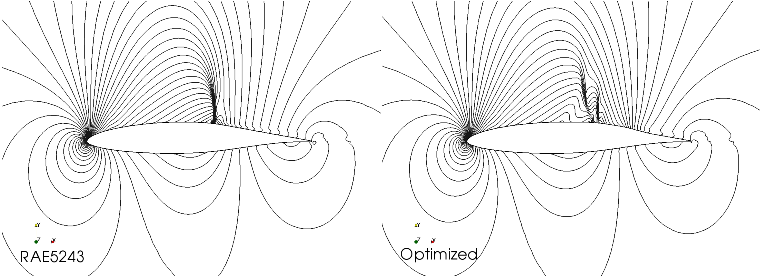
Definition of shape modification: four parameters

Airfoil modification

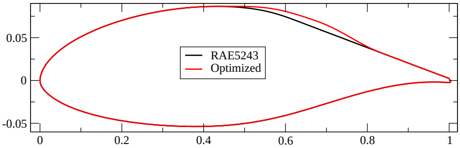


→ Convergence in about 30 iterations ∼ 150 simulations

Airfoil modification



Comparison of initial (left) and final (right) pressure fields

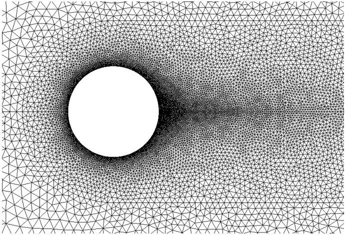


Comparison of initial (black) and final (red) airfoil shapes

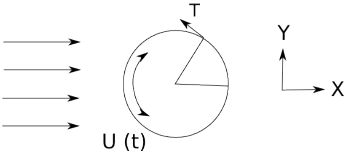
Flow control

Problem description

- Navier-Stokes modeling (laminar flow)
- Reynolds number $Re = 200$
- Oscillatory rotating cylinder
- Objective : reduce the time-averaged drag
- Two control parameters : amplitude and frequency

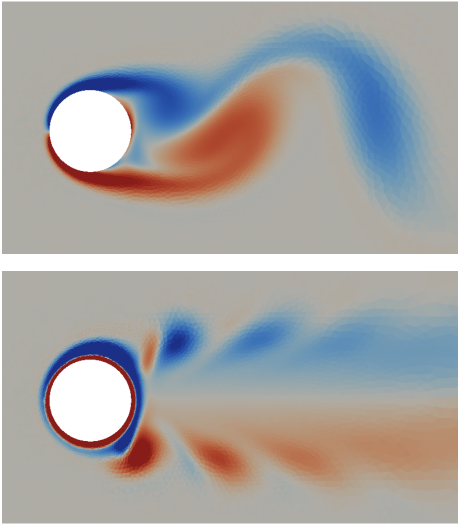


Mesh: 70,000 nodes 330,000 cell



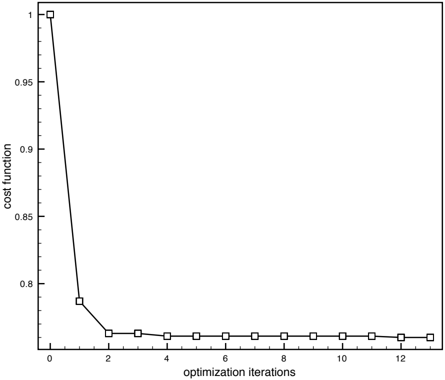
control system

Flow control



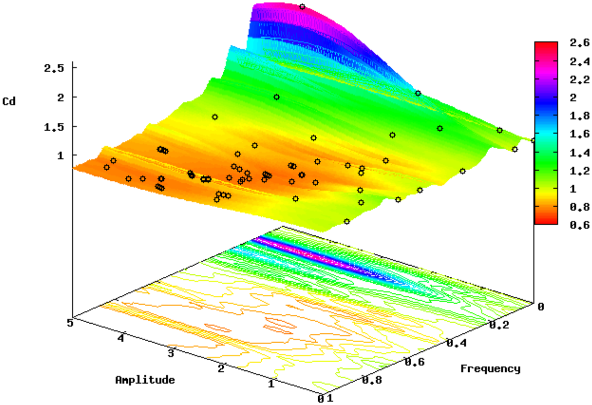
Comparison of vorticity fields without (top) and with control (bottom)

Flow control



$\rightarrow$ Convergence in about 5 iterations $\sim$ 35 simulations

Flow control



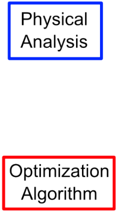
Surrogate model of drag coefficient

Introduction

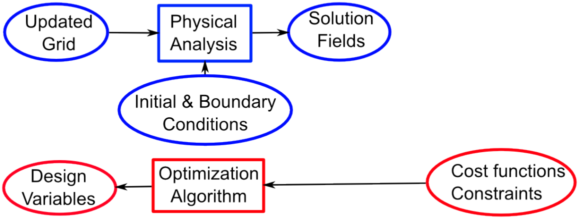
How to carry out such an optimization process ?

- What are the tools involved ?
- How are they organized ?

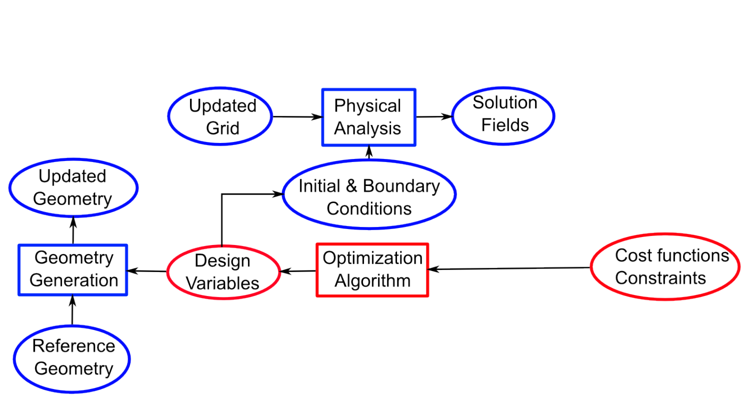
Design loop



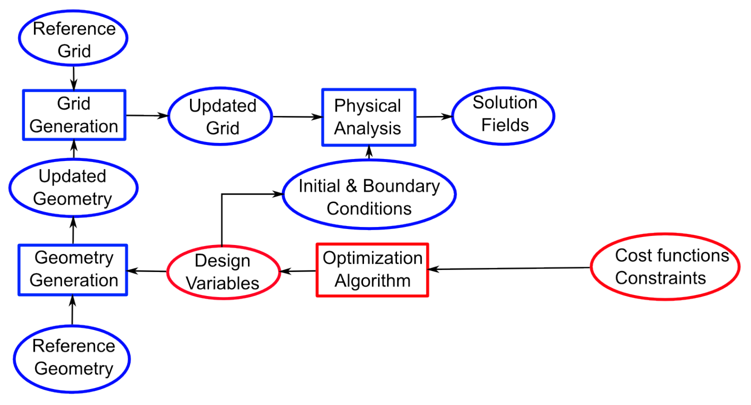
Design loop


Updated Grid
Physical Analysis
Solution Fields
Initial & Boundary Conditions
Design Variables
Optimization Algorithm
Cost functions Constraints

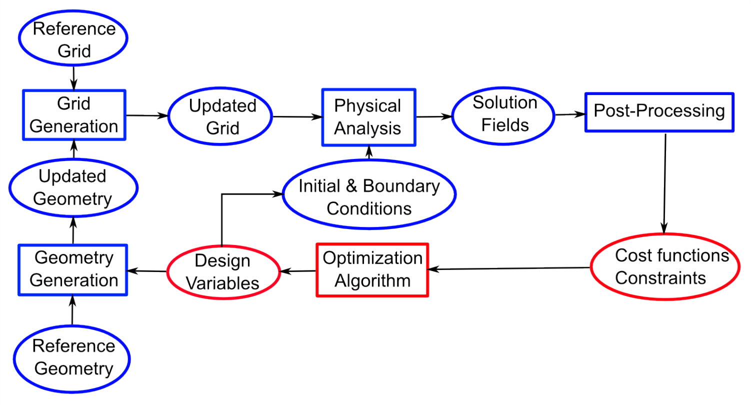
Design loop


Updated Grid
Physical Analysis
Solution Fields
Updated Geometry
Initial & Boundary Conditions
Geometry Generation
Design Variables
Optimization Algorithm
Cost functions Constraints
Reference Geometry

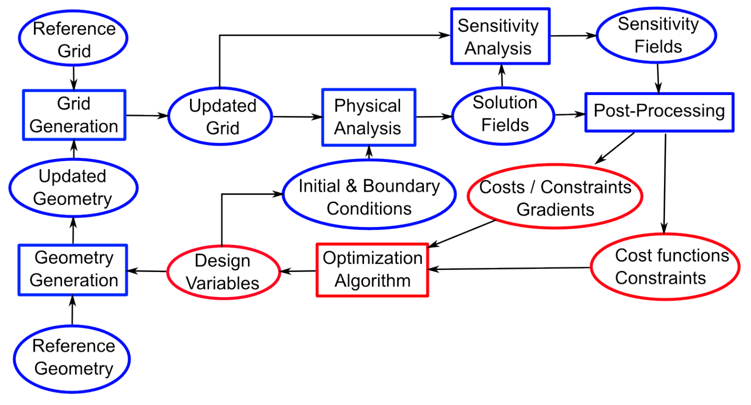
Design loop


Reference Grid
Grid Generation
Updated Grid
Physical Analysis
Solution Fields
Updated Geometry
Initial & Boundary Conditions
Geometry Generation
Design Variables
Optimization Algorithm
Cost functions Constraints
Reference Geometry

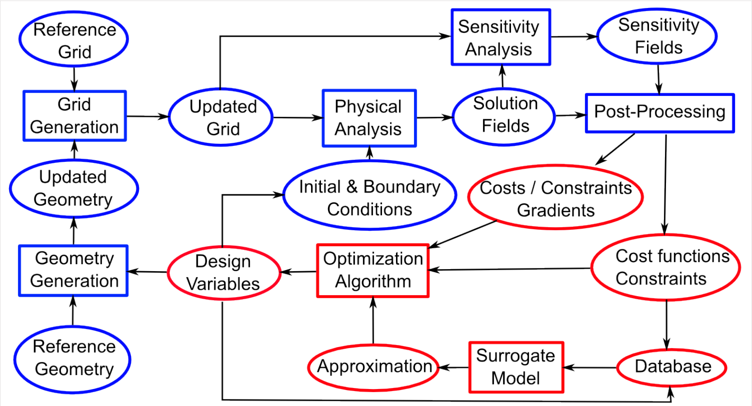
Design loop


Reference Grid
Grid Generation
Updated Grid
Physical Analysis
Solution Fields
Post-Processing
Updated Geometry
Initial & Boundary Conditions
Geometry Generation
Design Variables
Optimization Algorithm
Cost functions Constraints
Reference Geometry

Design loop


Reference Grid
Grid Generation
Updated Grid
Physical Analysis
Solution Fields
Sensitivity Analysis
Sensitivity Fields
Post-Processing
Updated Geometry
Initial & Boundary Conditions
Costs / Constraints Gradients
Geometry Generation
Design Variables
Optimization Algorithm
Cost functions Constraints
Reference Geometry

Design loop


Reference Grid
Grid Generation
Updated Grid
Physical Analysis
Solution Fields
Post-Processing
Sensitivity Analysis
Sensitivity Fields
Updated Geometry
Initial & Boundary Conditions
Costs / Constraints Gradients
Geometry Generation
Design Variables
Optimization Algorithm
Cost functions Constraints
Reference Geometry
Approximation
Surrogate Model
Database

Definition of an optimization problem

Minimize	$f(x)$	$x \in \mathbb{R}^n$	*cost function*
Subject to	$g_i(x) = 0$	$i = 1, \cdots, l$	*equality constraints*
	$h_j(x) \geqslant 0$	$j = 1, \cdots, m$	*inequality constraints*

Presentation

Definition of an optimization problem

Minimize	$f(x)$	$x \in \mathbb{R}^n$	*cost function*
Subject to	$g_i(x) = 0$	$i = 1, \cdots, l$	*equality constraints*
	$h_j(x) \geqslant 0$	$j = 1, \cdots, m$	*inequality constraints*

Role of the optimizer

Provide a candidate design vector $x \in \mathbb{R}^n$

Definition of an optimization problem

Minimize	$f(x)$	$x \in \mathbb{R}^n$	*cost function*
Subject to	$g_i(x) = 0$	$i = 1, \cdots, l$	*equality constraints*
	$h_j(x) \geqslant 0$	$j = 1, \cdots, m$	*inequality constraints*

Role of the optimizer

Provide a candidate design vector $x \in \mathbb{R}^n$

Some approaches

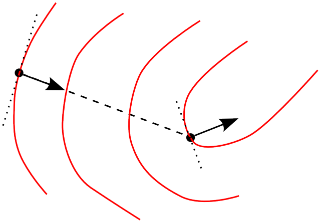
- Descent methods : suitable for smooth convex functions
  steepest descent, conjugate gradient, quasi-Newton, Newton methods
- Pattern search methods : suitable for noisy convex functions
  Nelder-Mead simplex, Torczon's multidirectional search algorithms
- Evolutionary methods : suitable for multimodal functions
  genetic algorithms, evolution strategies, particle swarm optimization, ant colony methods, simulated annealing

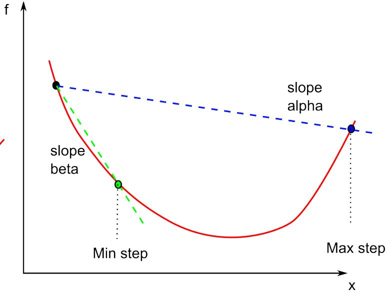
Steepest descent

Algorithm

For each iteration k:

- Estimation of gradient $\nabla f(x_k)$
- Define search direction $d_k = -\nabla f(x_k)$
- Line search : find the step length ρ such as :
  - $f(x_k + \rho d_k) < f(x_k) + \alpha \nabla f(x_k) \cdot \rho d_k$ (Armijo rule)
  - $f(x_k + \rho d_k) > f(x_k) + \beta \nabla f(x_k) \cdot \rho d_k$ (Goldstein rule)





Steepest descent

Summary

Properties of steepest descent method:

- Proof of convergence to a local optimum
- Linear convergence rate :

$$\lim_{k \to \infty} \frac{\|x_{k+1} - x^\star\|}{\|x_k - x^\star\|} = a > 0$$

In practice:

- Unable to take into account function curvature
- Oscillatory optimization path

Quasi-Newton methods

Algorithm

Improvement of the search direction:

- Definition according to the function curvature:

$$\widetilde{H}_k \, d_k = -\nabla f(x_k),$$

  with $\widetilde{H}$ approximation of the Hessian matrix (second-order derivative)

- Iterative construction of the Hessian matrix (BFGS formula) :

$$\widetilde{H}_0 = Id$$

$$\widetilde{H}_{k+1} = \widetilde{H}_k - \frac{1}{s_k^T \widetilde{H}_k s_k} \widetilde{H}_k s_k s_k^T \widetilde{H}_k^T + \frac{1}{y_k^T s_k} y_k y_k^T,$$

  with $s_k = x_{k+1} - x_k$ and $y_k = \nabla f(x_{k+1}) - \nabla f(x_k)$

Quasi-Newton methods

Summary

Properties of quasi-Newton methods:

- Take into account the curvature of the past iteration:

$$\widetilde{H}_{k+1}(x_{k+1} - x_k) = \nabla f(x_{k+1}) - \nabla f(x_k)$$

- $\widetilde{H}_k$ positive definite (if line search efficient enough)
- Proof of convergence to local optimum
- Super-linear convergence rate :

$$\lim_{k \to \infty} \frac{\|x_{k+1} - x^\star\|}{\|x_k - x^\star\|} = 0$$

In practice:

- Very efficient algorithm for convex problems (close to Newton method)
- Gradient should be available

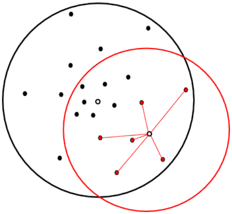
$(\lambda, \mu)-$Evolution Strategy

Algorithm

For each iteration k, according to $\bar{x}_k$ and $\bar{\sigma}_k$:

- Generate a set of λ perturbation lengths $\sigma_i = \bar{\sigma}_k e^{\tau\ N(0,1)}$
- Generate a sample of λ individuals $x_i = \bar{x}_k + \sigma_i\ N(0, Id)$ (mutation) where $N(0, Id)$ multivariate normal distribution with 0 mean and Id covariance matrix
- Evaluate the performance of λ individuals
- Choose the best μ parents among λ individuals (selection)
- Update distribution properties (crossover and self-adaption) :

$$\bar{x}_{k+1} = \frac{1}{\mu}\sum_{i=1}^{\mu} x^i \qquad \bar{\sigma}_{k+1} = \frac{1}{\mu}\sum_{i=1}^{\mu} \sigma^i$$



(λ, μ)−Evolution Strategy

Summary

Properties of evolution strategies:

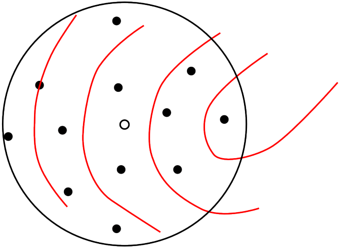
- Proof of convergence to the global optimum in a statistical sense

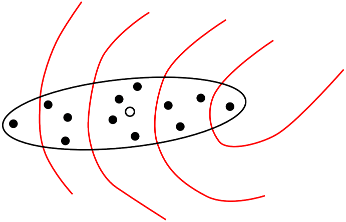
$$\text{e.g. } \forall \epsilon > 0 \quad \lim_{k\to\infty} P(|f(x_k) - f(x^\star)| \leqslant \epsilon) = 1$$

- linear convergence rate

In practice:

- Able to avoid local optima
- Curse of dimensionality
- Low local convergence rate due to isotropic search





Covariance Matrix Adaption (CMA-ES)

Algorithm

Improvement by using an anisotropic distribution:

- Use of a covariance matrix C_k to generate the population:

$$
\begin{aligned}
x_i &= \bar{x}_k + \bar{\sigma}_k \; N(0, C_k) \\
&= \bar{x}_k + \bar{\sigma}_k \; B_k D_k N(0, Id)
\end{aligned}
$$

  with B_k matrix of eigenvectors of $C_k^{1/2}$
  and D_k diagonal matrix of eigenvalues

- Iterative construction of the covariance matrix :

$$C_0 = Id$$

$$C_{k+1} = \underbrace{(1-c)C_k}_{\text{previous estimate}} + \underbrace{\frac{c}{m} p_k p_k^T}_{\text{1D update}} + \underbrace{c(1-\frac{1}{m}) \sum_{i=1}^{\mu} \omega^i (y^i)(y^i)^T}_{\text{covariance of parents}}$$

  with :

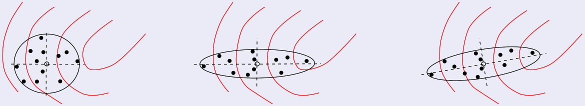
  - p_k evolution path (moves performed during last iterations)
  - $y^i = (x^i - \bar{x}_k)/\sigma_k$

Covariance Matrix Adaption (CMA-ES)

Summary

Properties:

- C_k is an approximation of the inverse of the Hessian matrix



$\bar{x}_k + \bar{\sigma}_k \; N(0, Id)$ $\quad$ $\bar{x}_k + \bar{\sigma}_k \; N(0, D^2)$ $\quad$ $\bar{x}_k + \bar{\sigma}_k \; N(0, BD^2B^T)$

- Several invariance properties:
  - Invariance under order-preserving transformations of the function
  - Invariance under scaling of the search space
  - Invariance under rotation of the search space

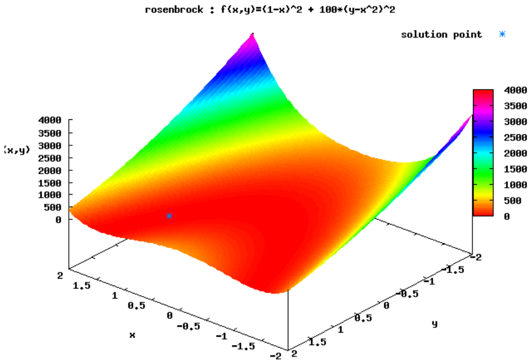
In practice:

- Outperforms most evolutionary methods
- Only a few parameters defined by the user

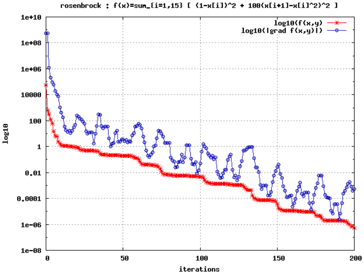
Analytical example

Rosenbrock function

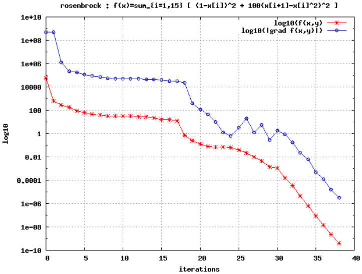
- Non-convex function "Banana valley"
- Dimension $n = 16$



Analytical example

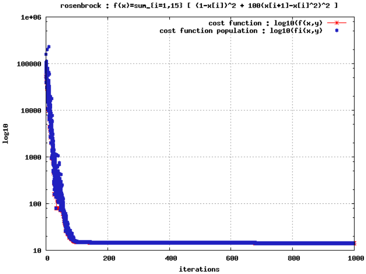


steepest-descent

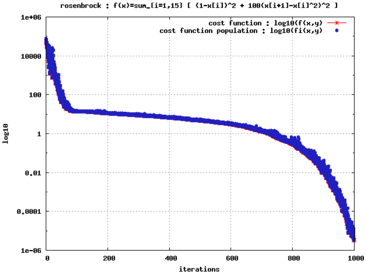


quasi-Newton

Analytical example



ES

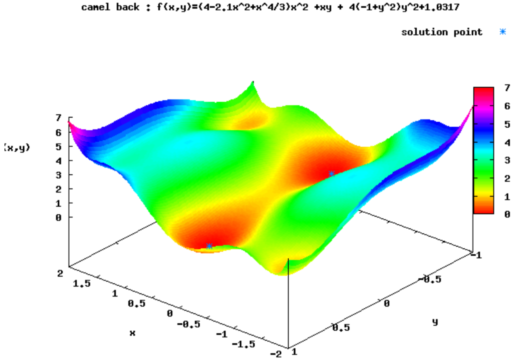


CMA-ES

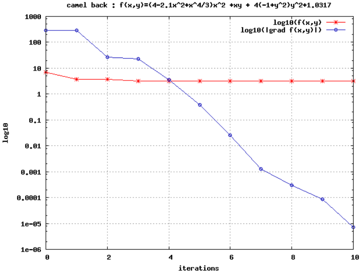
Analytical example

Camel back function

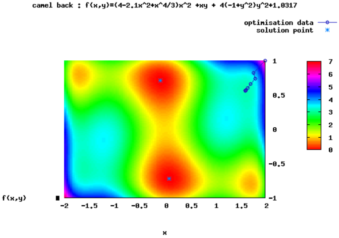
- Dimension $n = 2$
- Six local optima
- Two global optima



Analytical example

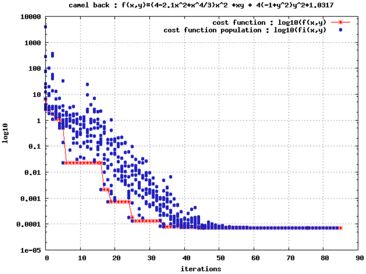


Quasi-Newton

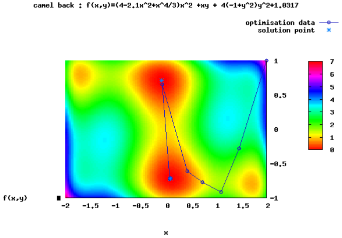


Optimization path

Analytical example

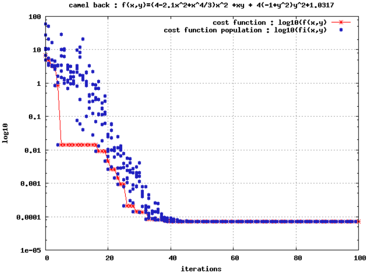


ES

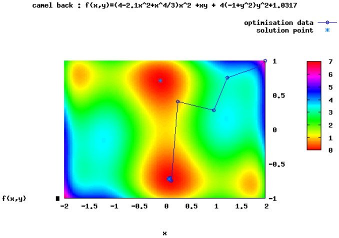


Optimization path

Analytical example



CMA-ES



Optimization path

Presentation

Role of the parameterization tool

- Describe the system with a few variables
- Transform a given design vector to a new engineering system

Presentation

Role of the parameterization tool

- Describe the system with a few variables
- Transform a given design vector to a new engineering system

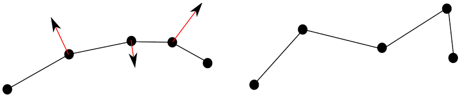
A particular case: shape parameterization

- Discrete shape : *set of moving nodes*
- Parametric surface : *Bézier, B-Splines, NURBS with control points*
- Composite approach : *set of baseline components with parameters*
- Free-form deformation : *deform a lattice that embeds the system*

Discrete shape representation

Principle

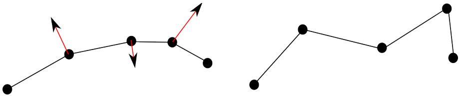
- Shape described by a surface grid
- Each node can be moved independently



Discrete shape representation

Principle

- Shape described by a surface grid
- Each node can be moved independently



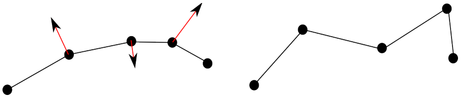
Advantages

- No strong assumption concerning the search space
- No need for CAD software (*CAD-free*)

Discrete shape representation

Principle

- Shape described by a surface grid
- Each node can be moved independently



Advantages

- No strong assumption concerning the search space
- No need for CAD software (*CAD-free*)

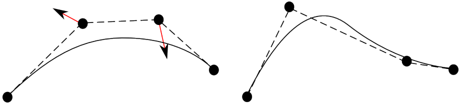
Drawbacks

- Smoothing required at each optimization step
- Very large number of design variables ($\sim$ 10,000)
- Optimized shape $=$ surface grid

Parametric shape representation

Principle

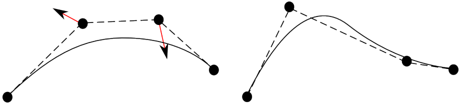
- Shape described by a parametric surface: $x(\xi) = \sum_{i=0}^{n} N_i(\xi) X_i$
- Choice of basis functions N_i:
  - Bézier : global influence, C^{∞}
  - B-Spline : compact support, C^{k-2} (k order)
  - NURBS : conic surfaces possible



Parametric shape representation

Principle

- Shape described by a parametric surface: $x(\xi) = \sum_{i=0}^{n} N_i(\xi) X_i$
- Choice of basis functions N_i:
  - Bézier : global influence, C^∞
  - B-Spline : compact support, C^{k-2} (k order)
  - NURBS : conic surfaces possible



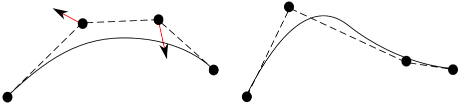
Advantages

- Low number of design variables (~ 10)
- Optimal shape is parametric and smooth
- Hierarchical representation

Parametric shape representation

Principle

- Shape described by a parametric surface: $x(\xi) = \sum_{i=0}^{n} N_i(\xi) X_i$
- Choice of basis functions N_i:
  - Bézier : global influence, C^∞
  - B-Spline : compact support, C^{k-2} (k order)
  - NURBS : conic surfaces possible



Advantages

- Low number of design variables (~ 10)
- Optimal shape is parametric and smooth
- Hierarchical representation

Drawbacks

- Mild assumption concerning the design space

Composite shape representation

Principle

- Use a set of baseline components (edges, circles, etc) with parameters (lengths, angles, positions)

Composite shape representation

Principle

- Use a set of baseline components (edges, circles, etc) with parameters (lengths, angles, positions)

Advantages

- Optimal shape obtained from baseline components (manufacturing constraint)
- Low number of design variables
- Variables chosen according to engineering experience

Composite shape representation

Principle

- Use a set of baseline components (edges, circles, etc) with parameters (lengths, angles, positions)

Advantages

- Optimal shape obtained from baseline components (manufacturing constraint)
- Low number of design variables
- Variables chosen according to engineering experience

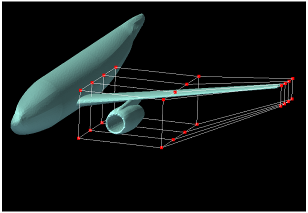
Drawbacks

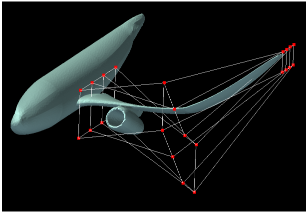
- Strong assumption concerning the design space

Free-Form Deformation (FFD)

Principle

- Represent deformation instead of the shape
- Embed the the system in a lattice, described as parametric volume

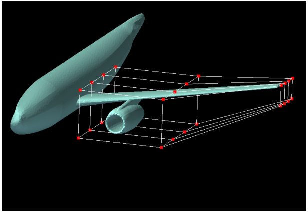


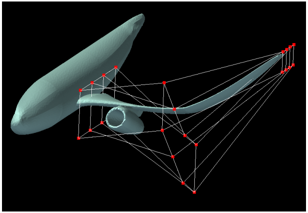


Free-Form Deformation (FFD)

Principle

- Represent deformation instead of the shape
- Embed the the system in a lattice, described as parametric volume





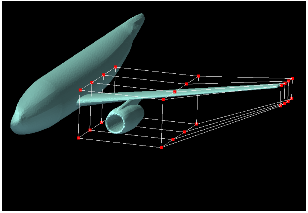
Advantages

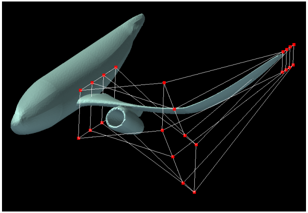
- Low number of design variables, whatever the system complexity

Free-Form Deformation (FFD)

Principle

- Represent deformation instead of the shape
- Embed the the system in a lattice, described as parametric volume





Advantages

- Low number of design variables, whatever the system complexity

Drawbacks

- Not easy to handle
- No representation of the optimal shape

Presentation

Role of the grid generation tool

- Generate a grid in accordance with the parameterized geometry
- Requirement: suitable for computations

Presentation

Role of the grid generation tool

- Generate a grid in accordance with the parameterized geometry
- Requirement: suitable for computations

Difficulties

- Automated task
- Need for robustness (large variations of geometry)
- Need for accuracy (mesh quality maintained)

Presentation

Role of the grid generation tool

- Generate a grid in accordance with the parameterized geometry
- Requirement: suitable for computations

Difficulties

- Automated task
- Need for robustness (large variations of geometry)
- Need for accuracy (mesh quality maintained)

Possible Approaches

- Generate completely a new grid
- Deform an existing reference grid

Complete mesh generation

Principle

- Store all the steps of the mesh construction (usually in a script) with some parameters as input (geometry)
- Run the grid generation software automatically for each new geometry

Complete mesh generation

Principle

- Store all the steps of the mesh construction (usually in a script) with some parameters as input (geometry)
- Run the grid generation software automatically for each new geometry

Advantages

- Very large geometry changes are possible

Complete mesh generation

Principle

- Store all the steps of the mesh construction (usually in a script) with some parameters as input (geometry)
- Run the grid generation software automatically for each new geometry

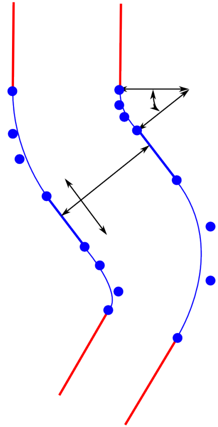
Advantages

- Very large geometry changes are possible

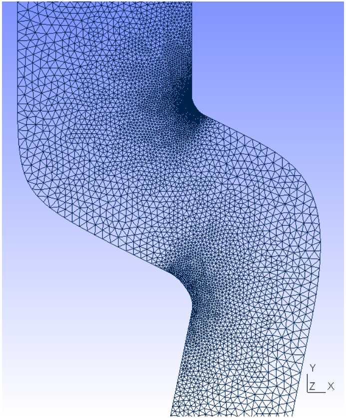
Drawbacks

- Regularity of the mesh w.r.t. geometry change ?
- treatment of boundary layers tedious

Illustration

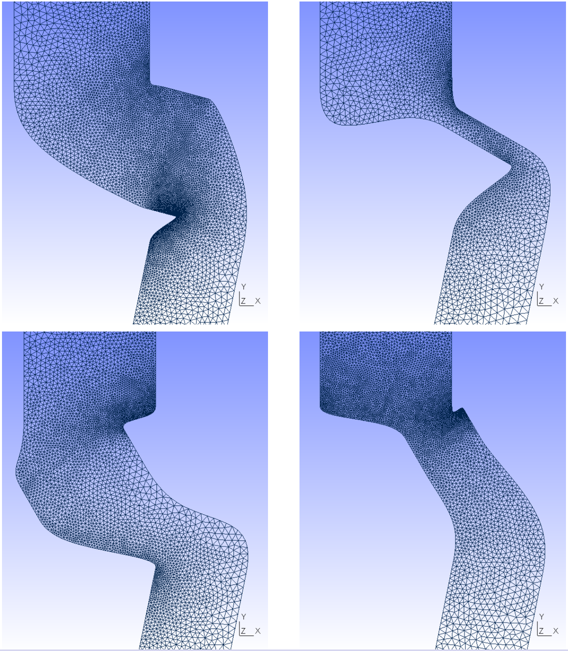


parameterization



initial mesh

Illustration



Reference mesh deformation

Principle

- Construct a reference mesh (initial geometry)
- Move the nodes according to geometry change (same topology)
- Need to solve an extra problem for the displacement field (e.g. structural analogy)

Reference mesh deformation

Principle

- Construct a reference mesh (initial geometry)
- Move the nodes according to geometry change (same topology)
- Need to solve an extra problem for the displacement field (e.g. structural analogy)

Advantages

- Control of the grid quality
- Smoothness of the deformation

Reference mesh deformation

Principle

- Construct a reference mesh (initial geometry)
- Move the nodes according to geometry change (same topology)
- Need to solve an extra problem for the displacement field (e.g. structural analogy)

Advantages

- Control of the grid quality
- Smoothness of the deformation

Drawbacks

- Moderate geometry change

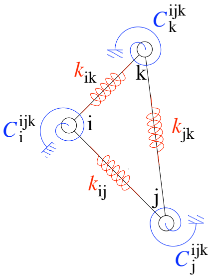
Example: discrete mechanical analogy

Method

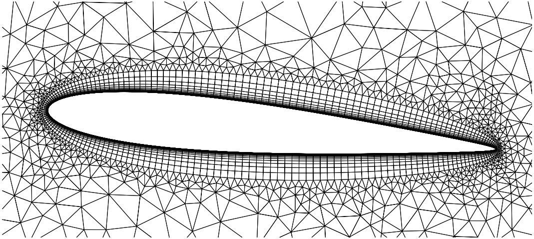
- Consider all edges as lineal springs and all angles as torsional springs
- Solve a linear system representing the mechanical equilibrium:

$$(K_{lin} + K_{tors}|Id)\, q = \begin{pmatrix} 0 \\ \overline{q} \end{pmatrix}$$

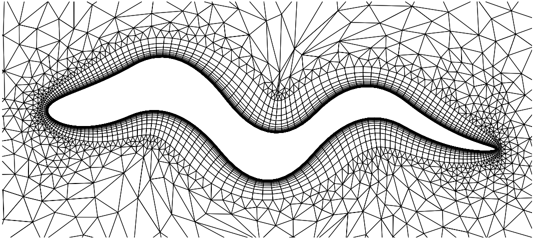
with:
K stiffness matrices
q displacement
$\overline{q}$ imposed boundary displacement



Example: discrete mechanical analogy

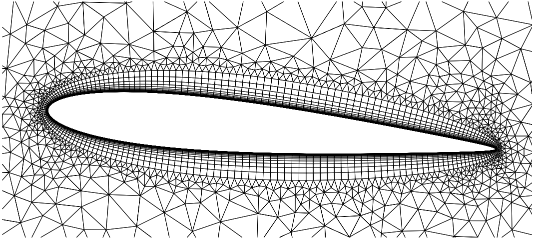


Initial grid

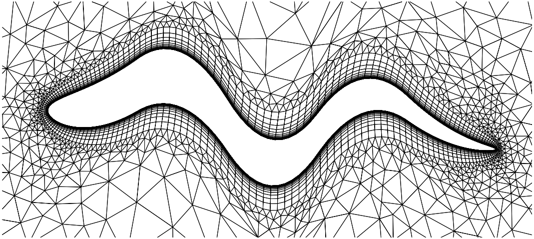


Deformed grid

Example: use of Free-Form Deformation



Initial grid



Deformed grid

Role of sensitivity analysis

- Compute the gradient of the cost function (e.g. drag) with respect to design variables

Presentation

Role of sensitivity analysis

- Compute the gradient of the cost function (e.g. drag) with respect to design variables

Possible Approaches

- Finite-difference approximation
- Complex estimation
- Adjoint approach

Finite-difference approximation of the gradient

Method

- Perform $n+1$ or $2n$ simulations for perturbed design variable values $x \pm \epsilon e_i$
- Compute approximated gradient:

$$\nabla f(x)|_i \simeq \frac{f(x+\epsilon e_i) - f(x)}{\epsilon} \qquad \nabla f(x)|_i \simeq \frac{f(x+\epsilon e_i) - f(x-\epsilon e_i)}{2\epsilon}$$

Finite-difference approximation of the gradient

Method

- Perform $n+1$ or $2n$ simulations for perturbed design variable values $x \pm \epsilon e_i$
- Compute approximated gradient:

$$\nabla f(x)|_i \simeq \frac{f(x+\epsilon e_i) - f(x)}{\epsilon} \qquad \nabla f(x)|_i \simeq \frac{f(x+\epsilon e_i) - f(x-\epsilon e_i)}{2\epsilon}$$

Drawbacks

- Choice of perturbation coefficient ϵ
- Low accuracy $\rightarrow$ noisy cost function
- Very expensive for large n

Finite-difference approximation of the gradient

Method

- Perform $n+1$ or $2n$ simulations for perturbed design variable values $x \pm \epsilon e_i$
- Compute approximated gradient:

$$\nabla f(x)|_i \simeq \frac{f(x+\epsilon e_i) - f(x)}{\epsilon} \qquad \nabla f(x)|_i \simeq \frac{f(x+\epsilon e_i) - f(x-\epsilon e_i)}{2\epsilon}$$

Drawbacks

- Choice of perturbation coefficient ϵ
- Low accuracy $\rightarrow$ noisy cost function
- Very expensive for large n

Advantages

- Non-intrusive approach

Complex estimation

Method

- Move all real variables x of the code to complex variables: $z = x + iy$
- The cost function (output) is also complex: $f = u + iv$
- The Cauchy-Riemann condition : $\frac{\partial u}{\partial x} = \frac{\partial v}{\partial y} \simeq \frac{v(x+i(y+h))-v(x+iy)}{h}$
- But values are real : $y = 0$ $v(x) = 0$ and $u = f$
- Then, the following approximation can be used:

$$\nabla f(x) \simeq \frac{Im[f(x+ih)]}{h}$$

- Propagate imaginary part of input variable to estimate gradient

Complex estimation

Method

- Move all real variables x of the code to complex variables: $z = x + iy$
- The cost function (output) is also complex: $f = u + iv$
- The Cauchy-Riemann condition : $\frac{\partial u}{\partial x} = \frac{\partial v}{\partial y} \simeq \frac{v(x+i(y+h))-v(x+iy)}{h}$
- But values are real : $y = 0$ $v(x) = 0$ and $u = f$
- Then, the following approximation can be used:

$$\nabla f(x) \simeq \frac{Im[f(x+ih)]}{h}$$

- Propagate imaginary part of input variable to estimate gradient

Advantages

- Accurate gradient estimation (no difference)

Complex estimation

Method

- Move all real variables x of the code to complex variables: $z = x + iy$
- The cost function (output) is also complex: $f = u + iv$
- The Cauchy-Riemann condition : $\frac{\partial u}{\partial x} = \frac{\partial v}{\partial y} \simeq \frac{v(x+i(y+h))-v(x+iy)}{h}$
- But values are real : $y = 0$ $v(x) = 0$ and $u = f$
- Then, the following approximation can be used:

$$\nabla f(x) \simeq \frac{Im[f(x+ih)]}{h}$$

- Propagate imaginary part of input variable to estimate gradient

Advantages

- Accurate gradient estimation (no difference)

Drawbacks

- Move CFD code to complex !
- n simulations required

Adjoint approach

Method

- Let consider the cost function as a constrained functional:

$$f : x \mapsto f(x) = F(x, W) \in \mathbb{R}$$

with $W = W(x) \in \mathbb{R}^N$ is the solution of the state equation:

$$\Psi(x, W) = 0$$

- Apply chain rule:

$$\frac{df}{dx_i} = \frac{\partial F}{\partial x_i} + \frac{\partial F}{\partial W}\frac{dW}{dx_i}$$

$$\frac{\partial \Psi}{\partial x_i} + \frac{\partial \Psi}{\partial W}\frac{dW}{dx_i} = 0$$

- by combining equations:

$$\frac{df}{dx_i} = \frac{\partial F}{\partial x_i} - \frac{\partial F}{\partial W}\left(\frac{\partial \Psi}{\partial W}\right)^{-1}\frac{\partial \Psi}{\partial x_i}$$

Adjoint approach

Method

- Finally, gradient can be computed by solving the (linear) system first:

$$\left(\frac{\partial \Psi}{\partial W}\right)^T \Pi = \left(\frac{\partial F}{\partial W}\right)^T$$

where Π are the adjoint variables, and then by computing:

$$\left(\frac{df}{dx}\right)^T = \left(\frac{\partial F}{\partial x}\right)^T - \left(\frac{\partial \Psi}{\partial x}\right)^T \Pi$$

Adjoint approach

Method

- Finally, gradient can be computed by solving the (linear) system first:

$$\left(\frac{\partial \Psi}{\partial W}\right)^T \Pi = \left(\frac{\partial F}{\partial W}\right)^T$$

where Π are the adjoint variables, and then by computing:

$$\left(\frac{df}{dx}\right)^T = \left(\frac{\partial F}{\partial x}\right)^T - \left(\frac{\partial \Psi}{\partial x}\right)^T \Pi$$

Advantages

- Adjoint system do not depend on x
- Cost rather independent from the size of x

Adjoint approach

Method

- Finally, gradient can be computed by solving the (linear) system first:

$$\left(\frac{\partial \Psi}{\partial W}\right)^T \Pi = \left(\frac{\partial F}{\partial W}\right)^T$$

where Π are the adjoint variables, and then by computing:

$$\left(\frac{df}{dx}\right)^T = \left(\frac{\partial F}{\partial x}\right)^T - \left(\frac{\partial \Psi}{\partial x}\right)^T \Pi$$

Advantages

- Adjoint system do not depend on x
- Cost rather independent from the size of x

Drawbacks

- The adjoint system has to be constructed (intrusive approach)
- Inversion of adjoint system difficult (badly conditionned)

Some reasons to use surrogate models

- Cost function evaluations are very expensive
- Some results are just discarded (in particular with ES)
- Weak use of past results (iterative process without history)

Some reasons to use surrogate models

- Cost function evaluations are very expensive
- Some results are just discarded (in particular with ES)
- Weak use of past results (iterative process without history)

Principles

- Store all cost function values into a database
- Use the database to build a surrogate model
- Use the surrogate model as cheap and approximate evaluation

Some reasons to use surrogate models

- Cost function evaluations are very expensive
- Some results are just discarded (in particular with ES)
- Weak use of past results (iterative process without history)

Principles

- Store all cost function values into a database
- Use the database to build a surrogate model
- Use the surrogate model as cheap and approximate evaluation

Some possible surrogate models

- Radial basis functions (RBFs)
- Artificial Neural Networks (ANNs)
- Gaussian processes (kriging)

Radial Basis Functions (RBF)

Evaluation

Approximation of the function $\mathcal{J}(\mathbf{x})$, $\mathbf{x} \in \Re^n$ of the form:

$$f(\mathbf{x}) = \sum_{j=1}^{N_c} \omega_j \ \phi_j(\mathbf{x}) \tag{1}$$

where ϕ_j are radial functions:

$$\phi_j(\mathbf{x}) = \Phi(\|\mathbf{x} - \mathbf{x}_j\|) \quad \Phi(r) = e^{-\frac{r^2}{s^2}} \tag{2}$$

$(\mathbf{x}_j)_{j=1,\dots,N_c}$	points stored in the database
s	attenuation factor
$(\omega_j)_{j=1,\dots,N_c}$	weights adjusted to fit the data

Radial Basis Functions (RBF)

Training

$(\omega_j)_{j=1,\dots,N_c}$ are determined from interpolation conditions:

$$f(\mathbf{x_i}) = \sum_{j=1}^{N_c} \omega_j \, \phi_j(\mathbf{x}_i) \quad i = 1, \dots, N_c \tag{3}$$

$(\omega_j)_{j=1,\dots,N_c}$ is the solution of the linear system:

$$\begin{pmatrix} \phi_1(\mathbf{x}_1) & \dots & \phi_{N_c}(\mathbf{x}_1) \\ \phi_1(\mathbf{x}_2) & \dots & \phi_{N_c}(\mathbf{x}_2) \\ \dots & \dots & \dots \\ \phi_1(\mathbf{x}_{N_c}) & \dots & \phi_{N_c}(\mathbf{x}_{N_c}) \end{pmatrix} \begin{Bmatrix} \omega_1 \\ \omega_2 \\ \dots \\ \omega_{N_c} \end{Bmatrix} = \begin{Bmatrix} \mathcal{J}(\mathbf{x}_1) \\ \mathcal{J}(\mathbf{x}_2) \\ \dots \\ \mathcal{J}(\mathbf{x}_{N_c}) \end{Bmatrix} \tag{4}$$

s set by the user or optimized by internal algorithm

Gaussian processes (Kriging)

Principle

- The vector of known function values F_N is assumed to be one realization of a Gaussian process:

$$p(F_N) = \frac{\exp\left(-\frac{1}{2} F_N^\top C_N^{-1} F_N\right)}{\sqrt{(2\pi)^N \det(C_N)}}$$

  with a given covariance matrix $C_{mn} = c(x_m, x_n)$.

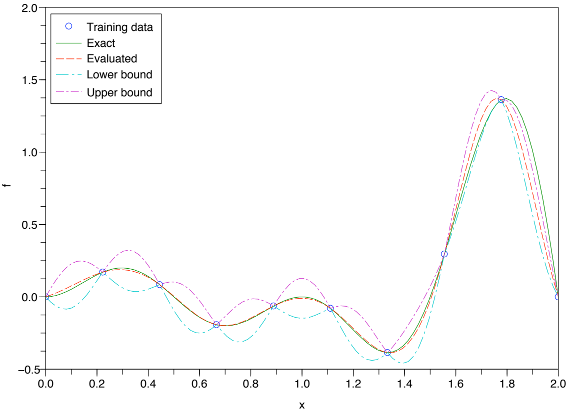
- It can be shown that (conditional probabilities):

$$p(f_{N+1}|F_N) \propto \exp\left[-\frac{(f_{N+1} - \hat{f}_{N+1})^2}{2\sigma^2_{f_{N+1}}}\right]$$

  where:

$$\hat{f}_{N+1} = k^\top C_N^{-1} F_N, \qquad \sigma^2_{f_{N+1}} = \kappa - k^\top C_N^{-1} k$$

Illustration for 1D problem



Gaussian Processes (Kriging)

Choice of the covariance function

- Distance dependent correlation, scaling, offset

$$c(x,y) = \theta_1 \exp\left[-\frac{1}{2}\sum_{i=1}^{d}\frac{(x_i-y_i)^2}{r_i^2}\right] + \theta_2,$$

- Parameters to be optimized $\Theta = (\theta_1, \theta_2, r_1, r_2, \ldots, r_d)$

Choice of the parameters (training phase)

- Choose Θ to maximize the likelihood of the known function values
- Internal optimization

Application to Inexact Pre-Evaluation (IPE)

Principle

- For evolutionary optimizers several evaluations are just not used
- Use surrogate models as pre-screening criterion for CFD evaluation

Application to Inexact Pre-Evaluation (IPE)

Principle

- For evolutionary optimizers several evaluations are just not used
- Use surrogate models as pre-screening criterion for CFD evaluation

Expected benefits

- Avoid useless evaluations for evolutionary optimizers
- Reduce drastically the cost of evolutionary optmizers
- Low coupling between optimizer and metamodel

Inexact Pre-Evaluation (IPE) approach

1. Set $n = 0$
2. If $n \leq N_e$ compute cost function $f(x_k^n), k = 1, \ldots, K$ using the exact model, else using metamodel $\tilde{f}(x_k^n), k = 1, \ldots, K$.
3. If $n > N_e$, then select a subset of points S^n for exact evaluation.
4. Update the optimizer parameters (mean, standard deviation) *using only the exactly evaluated cost functions*
5. Store exactly evaluated function values into a database
6. If $n < N_{\text{max}}$, then $n = n + 1$ and go to step (iii), else STOP.

Illustration for business jet

- CFD evaluations for the 10% best points or points with predicted improvement
- 500 iterations
- RBF model with local database (40 pts)

Number of exact evaluations

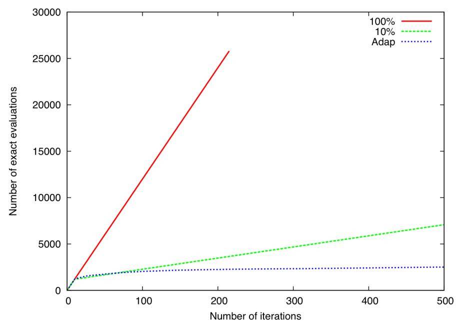
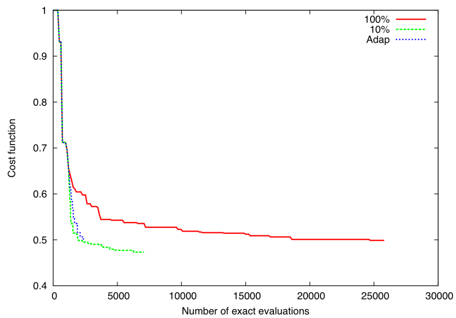
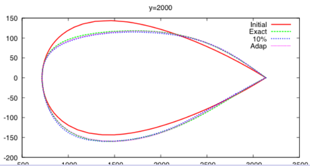


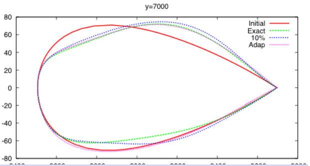
Illustration for business jet

Cost function



Final shapes





Application to Efficient Global Optimization (EGO)

Main ideas

- Use a metamodel to drive the search
- Use a Gaussian Process model to take into account the probability of obtaining a better design

Application to Efficient Global Optimization (EGO)

Main ideas

- Use a metamodel to drive the search
- Use a Gaussian Process model to take into account the probability of obtaining a better design

Expected benefits

- Global optimization (proof of convergence)
- Deterministic approach
- Measure of the probability of improvement

EGO algorithm

Basics

- Build iteratively a database and corresponding Gaussian process
- Enrichement chosen in order to:
  - Minimize the current Gaussian process model
  - Explore where the probability of improvement is high

EGO algorithm

Basics

- Build iteratively a database and corresponding Gaussian process
- Enrichement chosen in order to:
  - Minimize the current Gaussian process model
  - Explore where the probability of improvement is high

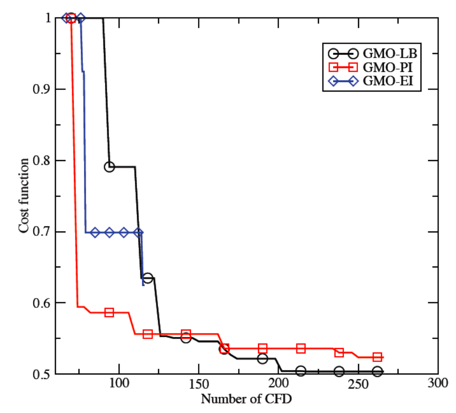
Algorithm

1. Build an *a priori* database (Latin Hypercube Sampling)
2. Construct a global Gaussian process
3. Find the points $x_i^\star$ that minimize / maximize a merit function :
   - Statistical lower bound
   - Probability of improvement
   - Expected improvement
4. Evaluate the p points $(x_i^\star)_{i=1,\ldots,p}$ and add them in the database
5. Return to step 2 until convergence

Illustration for business jet

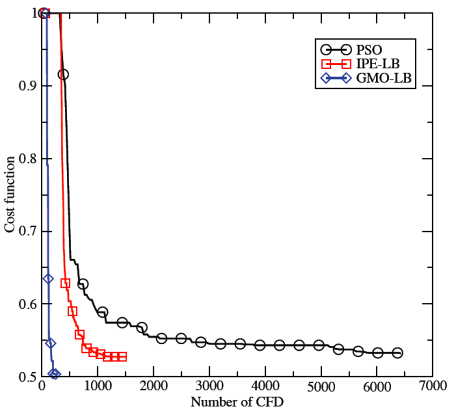
- EGO algorithm with three merit functions
- 150 iterations
- Initial database size : 60 points

Cost function



Comparison IPE - EGO

Cost function



Use of surrogate models

Main ideas

- Use all evaluations already performed
- Use expensive simulation only if required

Use of surrogate models

Main ideas

- Use all evaluations already performed
- Use expensive simulation only if required

Advantages

- Drastic reduction of CPU cost
- Interesting for post-processing / interactive study

Use of surrogate models

Main ideas

- Use all evaluations already performed
- Use expensive simulation only if required

Advantages

- Drastic reduction of CPU cost
- Interesting for post-processing / interactive study

Drawbacks

- More sophisticated approach
- Restricted to problems of low dimension

Conclusion

Optimization and control in CFD:

- A multi-disciplinary field:
  - Applied mathematics
  - Numerical methods
  - Physical phenomena
  - Geometry
  - Computer sciences
- Efficiency of methods are strongly problem dependent
- Efficiency depends strongly of the global coherency of the loop

Conclusion

Several topics have not been discussed:

- Multi-objective optimization: *How to minimise several criteria ?*
- Multi-disciplinary optimization: *How to couple several disciplines ?*
- Constrained optimization: *How to take into account constraints ?*
- Robust optimization: *How to take into account uncertainties ?*
- Hierarchical optimization: *How to develop multi-level strategies ?*
- Distributed optimization: *How to use parallel computing ?*
- Automatic differentiation: *How to use AD softwares ?*
- ...