J. Peter, G. Carrier, D. Bailly,
P. Klotz,.M. Marcelet, F. Renac

(Onera)

E-mail: jacques.peter@onera.fr

Local and Global Search Methods for Design in Aeronautics

Optimization is used at all stages of aircraft design. In the early phases (also called preliminary or conceptual design) multicriteria Pareto optimization [11] based on heuristic laws is carried out. Conversely, in the late design phases, accurate structural and aerodynamics predictions are required to steer small changes of a baseline shape through mono-objective optimization. This article is devoted to the local and global optimization methods actually used during this second part of the design process.

Through discretization, parametrization, and numerical simulation, the design problem can be formulated mathematically as a finite-dimensional optimization. Hence a good knowledge of global and local optimization algorithms is important to aerodynamic design engineers. The classical algorithms which are currently used at Onera for shape design are presented. Besides, many local optimization algorithms require the gradient of the functions of interest with respect to the design parameters. The different ways to compute those derivatives - often called “sensitivities” - are also described. Numerous 2D and 3D applications were dealt with at Onera using the methods described; they are presented in a companion article [14] and also briefly described in a course [1].

Introduction

Numerical optimization aims at locating the minima of a regular function (called objective function) on a finite-dimensional design space, while satisfying a certain number of constraints (expressed as inequality verified by the so-called constraint functions). More precisely, local optimization aims to find a local optimum in the region of an initial guess, whereas global optimization aims to find the global optimum on the whole design space. These problems are, of course, the mathematical counterparts of mechanical optimization problems - like drag minimization of an aircraft or maximization of the total pressure recovery of a supersonic aircraft air intake - as soon as (a) a mesh and a numerical simulation tool are available; (b) the shape of the object to be optimized has been parametrized/a mesh deformation technique is available to propagate its deformation to the whole mesh; (c) the objective and constraints of the mathematical optimization problem have been expressed as functions of the geometry and state variables, which are usually the results of dedicated post-processing tools.

Numerical optimization for aircraft design was introduced almost as soon as mature simulation codes appeared. The aerodynamic optimizations carried out by G.N. Van der Plaats at Nasa in the mid 70’s illustrate this early interest in optimization [18]. At that time, 2D and simple 3D configurations were considered, simplex or descent methods were used and the gradients required by descent methods were estimated by the finite-differences.

Since then, the framework of aerospace optimization has known at least three drastic extensions: (1) several global optimization methods have been defined and intensively used (evolutionary algorithm, particle swarm, ant colony, simulated annealing,...); (2) surrogate functions (neural network, Kriging, polynomial regression, support vector machine,...) have been used for a part of the evaluations of the global optimization methods leading to significant cost reductions; (3) adjoint vector and direct differentiation methods have been defined, studied and increasingly frequently used to compute the gradients necessary for descent algorithms.

The next section presents the basic definitions and theorems. The third section is dedicated to global search methods. The fourth section is devoted to descent methods. The fifth section describes the adjoint and direct methods that can efficiently compute the gradient of the functions of interest with respect to the design parameters. All

of the last three sections concentrate on the methods which were found to be among the most efficient and are actually used at Onera for aerodynamic design.

Notations, mathematical problem and properties

Mathematical optimization problem

In this first subsection, the classical notations of a mathematical finite-dimensional optimization problem are defined. Let α be the current vector of the input space (design vector). Let us denote n_f its dimension. The vector α is supposed to vary in D_α (the design space), a parallelepiped of $\mathbb{R}^{nf}$. The objective function to minimize on $D_\alpha \subset \mathbb{R}^{nf}$ is denoted $J(\alpha)$. The constraints of the problem are supposed to be formulated through n_c functions $G_j(\alpha)$, that are negative at admissible design points. Only inequality constraints are considered here, as for most practical design problems an adequate choice of the design parameters enables us to avoid equality constraints. Obviously the local and global optimization problems read

Global [resp. local] optimum search
Seek for $\alpha^* \in D_\alpha$ such that $J(\alpha^*) = Min\, J(\alpha) over\ D_\alpha$ design space [resp. on V_{α^*} neighborhood of α^*]
and $\forall j \in [1, n_c] \quad G_j(\alpha^*) \le 0$

In most common situations, the objective and constraint functions are at least continuous. In this presentation, for the sake of simplicity, the functions and state equations are supposed to have C^1 regularity and, in some sections, also C^2 regularity.

Optimization problem stemming from a numerical simulation

In the framework of aircraft or turbomachinery design, the functions of interest depend on a distributed state field and geometrical variables that are linked by a system of non-linear equations discretizing the governing equations of fluid dynamics. Let us note $S(\alpha)$, the coordinates of the surface mesh of a solid body. From any surface mesh, $S(\alpha)$, a volumic mesh $X(\alpha)$ is built. Both S and X are also supposed to have C^1 regularity. The Jacobian $dX/d\alpha$ can always be estimated by finite differences and in some cases by the following product of Jacobians $\frac{dX}{d\alpha} = \frac{dX}{dS}\frac{dS}{d\alpha}$. The state variables (aerodynamic conservative variables) are noted W (vector of size n_W). State variables and mesh satisfy the discrete equations of fluid mechanics – a discrete form of Reynolds averaged Navier-Stokes (RaNS) equations, for example. In the framework of finite-difference/finite-volume methods, these equations read: $R(W, X) = 0$ (in general, nonlinear set of n_W equations). These equations are also supposed to have C^1 regularity with respect to (w.r.t.) their two vector arguments. It is also assumed that, all over the design space the flow is perfectly converged and the hypothesis of the implicit function theorem is satisfied ($\forall((W_i, X_i) / R(W_i, X_i) = 0 \quad \det(W_i, X_i) \neq 0$) which defines the flow field W as a C^1 function of the mesh X, and then as a C^1 regular function of α over D_α. The discrete state equations may be rewritten $R(W(\alpha), X(\alpha)) = 0$.

In case of an optimization problem associated with a framework of numerical simulation, the objective function is actually computed as $J(\alpha) = \overline{J}(W(\alpha), X(\alpha))$. The constraint functions $G_j(\alpha)$ have the same dependencies $G_j(\alpha) = \overline{G_j}(W(\alpha), X(\alpha))$. The definitions of global optimum search and local optimum search may be rewritten using $\overline{J}$ and $\overline{G_j}$.

The Karush-Kuhn-Tucker condition

For the unconstrained optimization of a C^2 function of $\mathbb{R}^{n_f}$, classical conditions of existence for minima read:

- Local optimum located in α^* - $\nabla J(\alpha^*) = 0$ is a necessary condition. $\nabla J(\alpha^*) = 0$ and $H(\alpha^*)$ positive definite (H hessian matrix of J) is a sufficient condition.
- Global optimum located in α^* - $\nabla J(\alpha^*) = 0$ is a necessary condition. $\nabla J(\alpha^*) = 0$ and $H(\alpha) = 0$ positive definite on D_α is a sufficient condition.

For a constrained problem on a finite size domain, the necessary condition for optimality is more complex to express. Actually, we first have to introduce more explicit notations for the parallelepiped design space D_α :

$$D_\alpha = [\alpha_{1,l}, \alpha_{1,u}] \times [\alpha_{2,l}, \alpha_{2,u}] \times [\alpha_{3,l}, \alpha_{3,u}] ... \times [\alpha_{n_f,l}, \alpha_{n_f,u}]$$

Then the domain bounds are rewritten as $2n_f$ additional constraints
$G_{n_c+1}(\alpha) = \alpha_{1,l} - \alpha_1$, $G_{n_c+2}(\alpha) = \alpha_1 - \alpha_{1,u}$, …
…, $G_{n_c+2n_f}(\alpha) = \alpha_{n_f} - \alpha_{n_f,u}$.
For an optimization problem with inequality constraint the classical necessary conditions for optimality in α^* are the so-called "Karush-Kuhn-Tucker" conditions: α^* is an admissible state

$$\nabla J(\alpha^*) + \sum_j \lambda_j^* \nabla G_j(\alpha^*) = 0 \quad \forall j \in [1, n_c + 2n_f] \quad \lambda_j^* G_j(\alpha^*) = 0$$

(the last proposition means that only the constraints attaining the limit value zero may have their gradient included in the linear combination of gradients)

An illustration of the KKT condition for a 2D problem is presented by figure 1. The functions are:
$J(\alpha) = \alpha_1^2 + 3\alpha_2^2$, $G_1(\alpha) = 1 - \alpha_1 + \alpha_2^2$, $G_2(\alpha) = 4 - \alpha_1 - 2\alpha_2$.

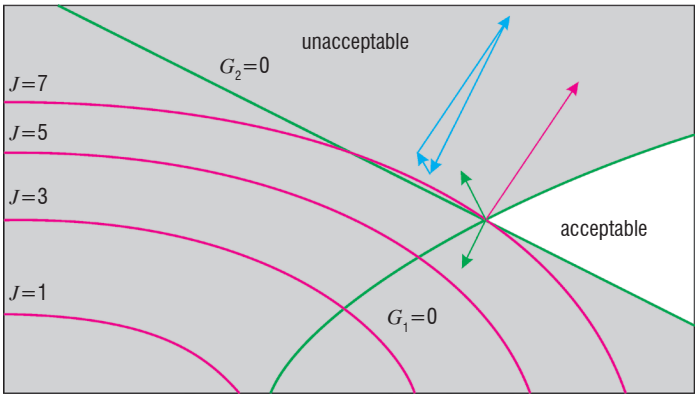

Figure 1 - KKT condition for a 2D example

Global search methods

Stochastic methods

Stochastic optimization methods refer to non deterministic, direct search algorithms, and belong to the family of global search methods. As a direct search method, a stochastic algorithm is suited for optimization problems presenting non-differentiable or even non-continuous functions. However, the main advantage of stochastic methods, which

have a common peculiarity of introducing some randomness in their search process, is their ability to deal with multimodal optimization problems since they (can) provide the global minimum for problems with several local minima. This interesting feature, directly inherited from the stochastic nature of their search process, is achieved at the price of a significant increase of the number of function evaluations during the optimization process, compared to local methods.

Stochastic methods cover a wide variety of algorithms. The most popular stochastic algorithms which have been concretely applied to engineering design optimization problems are (not an exhaustive list): evolutionary algorithms, a category of methods which includes Genetic Algorithms (GA) [5][6] and Differential Evolution (DE) [15][16] algorithms, the Simulated Annealing (SA) [7] method, the Particle Swarm Optimization (PSO) [8] method and the Ant Colony Optimization technique [3]. Compared to a pure randomized search (which can be considered as a stochastic method), the previously cited methods are all grounded on some simple heuristics or more complex biologically or nature inspired metaheuristics which provide them with better performances, at least for some classes of optimization problems.

Over the past ten years, several applications of stochastic optimization methods to aerodynamic and multidisciplinary design optimization problems have been performed at Onera [1][3]. Several of these applications relied on GADO [12], a GA developed at Rutgers University.

Surrogate based methods

Stochastic methods are effective search algorithms and can reach the optimal solution without getting trapped into local minima. However a large number of evaluations are usually required to obtain this optimal point. Typically, even the "greediest" (most efficient, possibly less robust) stochastic methods will require several thousand evaluations to provide an optimum design. The aerodynamic-related experiences conducted at Onera with such optimization techniques involved analyses requiring less than 10 minutes (wall-clock) on super-computers. Such analysis time was achieved typically for 3D Euler simulation (in meshes of about 500.000 nodes) or 2D RANS simulation. This yielded a total time of one to two weeks for an optimization, while the target time for actual design applications in industry would be 1 day, or even half a day (one night). For many years now, alternative global search methods have been studied to reduce the CPU cost. One of their features is that they use surrogate model techniques in order to strongly reduce the number of CFD computations.

A first approach consists of building a sufficiently accurate model that presents the same optimal point than the objective function. Current response surfaces based methods used for global optimization can be categorized on the basis of the type of response surface and method used to determine the new design points with which to enrich the surrogate model. A first approach is based on surrogate model construction from an initial sampling. The model's accuracy depends on the sampling size and the location of the data points. So space filling techniques are better than the classical ones, because they cover the design space in a uniform way. A second approach consists in using infill criteria. A first model is build from a small size initial sampling. The predictor and an estimate of the model's error are used to determine new search points. Finally, a third approach consists of using a stochastic optimization algorithm and evaluating the different individuals of the population with the surrogate model.

Surrogate model techniques

Artificial Neural Network (ANN)

Multi-layer perceptron is the most popular neural network architecture, typically consisting in a hidden layer placed between the input and the output of the neural network. The number of neurons of the hidden layer (Nc) is a given parameter defining the degrees of freedom of the model. The internal connections between neurons of consecutive layers are characterized by unknown weights and an activation function $\Phi(z)=\frac{1}{1+e^{-z}}$.

The ANN model is $\hat{J}_{ANN}(\alpha)=\sum_{j=1}^{Nc} B_j \Phi\left(\sum_{k=1}^{n_f} A_{jk}\alpha_k + A_{j0}\right)+B_0$

where α_k is the k-th component of α. The weights A, B are estimated with a gradient-based optimization procedure by solving the system for a database $(\alpha^l, J(\alpha^l))\, l \in [1, n_{db}]$ $(\alpha^l \in \mathbb{R}^{n_f})$. Because the weights are not unique, more than one network must be built in order to choose the one providing the best prediction.

Model complexity increases with neuron number. Too few neurons can lead to under fitting. Too many neurons can contribute to over fitting, in which all training points are well fitted, but the fitting curve oscillates widely between these points.

Radial Basis Functions (RBF)

This surrogate model has been developed for the interpolation of scattered multivariate data. It appears as a linear combination of N radial basis $\psi(r)$, of the form:

$$\hat{J}_{RBF}(\alpha)=\sum_{i=1,N} w_i \psi_i(\|\alpha - c^i\|)$$

where w_i represents the weights of the linear combination, c^i is the ith of the N basis function centers. A strong simplification consists in imposing N to be equal to n_{db}, the number of data points and the centers $c^i=\alpha^i$. This leads to the matrix equation $\Psi w = y$, which gives the weights by inversion (Ψ being the symmetric matrix defined by $\Psi_{ij}=\psi(\|\alpha^j-\alpha^i\|)$). The choice of the radial basis function can have an important effect, since Gaussian and inverse multi-quadric basis functions lead to a symmetric positive definite matrix. Estimating the other parameters, like a (see table 1), is an additional task.

Thin plate spline	$\psi(r)=r^2 \ln r$
Gaussian	$\psi(r)=e^{-r^2/2a^2}$
Multi-quadric	$\psi(r)=(r^2+a^2)^{1/2}$
Inverse multiquadric	$\psi(r)=(r^2+a^2)^{-1/2}$

Table 1 - Examples of radial basis functions

Kriging

While building this surrogate model, one assumes that the output function is the sum of a regression and a stochastic part. The last one is supposed to have zero mean and follow second order stationary process described by a correlation model $\Re(\theta,d)$ (see table 2).

Exponential	$\Re(\theta,d)=e^{-\theta\left\|d\right\|^p}$
Gaussian	$\Re(\theta,d)=e^{-\theta d^2}$
Spline cubic $\xi=\theta\left\|d\right\|$ $0<a<1$	$\Re(\theta,d)=\begin{cases}(1-3/a)\xi^2+(1+a)/a^2\xi^3 & 0\le\xi\le a\\(1-\xi)^3/(1-a) & a\le\xi\le 1\\0 & 1\le\xi\end{cases}$

Table 2 - Examples of correlation models

The likelihood is maximised in order to estimate the unknown parameters θ and p. In the case of ordinary kriging, the surrogate model relations are:

$$\hat{J}_{OK}(\alpha)=\hat{\mu}+r^T\mathsf{R}^{-1}(J_s-1\hat{\mu})\qquad \hat{\mu}=\frac{1^T\mathsf{R}^{-1}J_s}{1^T\mathsf{R}^{-1}1},$$

where r is the correlation vector between the prediction point and the data points $(r_i=\Re(\theta,\|\alpha^i-\alpha\|))$, $J_s=(J(\alpha^1),J(\alpha^2),...J(\alpha^{n_{db}}))$ is the vector of the function values of the data points, R is the correlation matrix for the sample points $(\mathsf{R}^{ij}=\Re(\theta,\|\alpha^i-\alpha^j\|))$, 1 is a vector of ones of size n_{db}. The estimate of variance at unsampled points is given by:

$$s^2(\alpha)=\sigma^2[1-r^T\mathsf{R}^{-1}r+\frac{\left(1-1^T\mathsf{R}^{-1}r\right)}{1^T\mathsf{R}^{-1}1}]$$

Initial sampling techniques

An initial sampling is essential to obtain an accurate surrogate model, as all the parameters of the surrogate models are estimated from data points (location and function value). Those techniques can be divided into two categories:

- Classical samplings: Full factorial and Central Composite Design (CCD) are the more basic forms of sampling. They present the drawback that the size of the sampling increases exponentially with the number of design variables. D-optimal design is suitable for higher dimensional problems. In the case of systematic error, an appropriate design must fill the design space rather concentrate on the boundaries.
- Space filling samplings: Space filling samplings spread data sampling points throughout the entire design space. The most common methods are Orthogonal arrays, Latin Hypercube, Hammersley and uniform samplings. They cover the design space in a uniform way and in general are adapted to large scale problems.

Adaptive sampling techniques

The use of an initial sampling leads to a globally accurate model. However, it is unlikely that the model is sufficiently accurate in the region of the predicted optimum. Hence infill points have to be defined to improve the model's accuracy. The location of these new points is based on a criterion. Several classical criteria are presented in [16]. The most important is the Expected Improvement (EI). It consists of calculating the expected accuracy improvement, given the predictor $\hat{J}$ and the variance s^2 :

$$EI=\begin{cases}(J_{\min}-\hat{J}(\alpha))\Phi\left[\frac{J_{\min}-\hat{J}(\alpha)}{s(\alpha)}\right]+s\varphi\left[\frac{J_{\min}-\hat{J}(\alpha)}{s(\alpha)}\right] & \text{if } s>0\\0 & \text{if } s=0\end{cases}$$

where Φ and φ are, respectively, the normal cumulative distribution function and probability density function.

Acceleration of stochastic algorithms using surrogate models

The acceleration of a GA is obtained from the reduction of the number of exact function evaluations by the costly numerical simulation codes (CFD code in the case of aerodynamics). This involves a surrogate model which provides more or less accurate evaluations of $J(\alpha)$ and $G_l(\alpha)$, at negligible CPU cost. The use of a surrogate model instead of the numerical simulation code usually preserves the robustness of the complete optimization process. Actually, the dynamic improvement of the surrogate model is more appropriate for converging to the optimum value.

Several approaches have been defined to partly replace the numerical simulation codes by a global surrogate model. The sampling points are generally obtained after one or several generations of a classical stochastic research. Then, the model is updated online, from the new generated points, when the optimization process is going on. Due to the curse of dimensionality, the accuracy of the global surrogate models becomes more and more difficult to insure for high number of variables (high n_f). In order to overcome this drawback, another approach involving local surrogate models can be used during the search process. Finally, the two types of models (local and global) can be used to improve the speeding up of the optimization process

Local Search Using Derivatives – Descent Methods

A descent method is an iterative method for the solution of a local optimization problem. Starting from an initial point, it attempts to converge to an optimum, using local information (most often function value and gradient) at the current point to compute the next iterate.
In the following pages, we do not present the popular algorithms for 1D minimization (dichotomy, golden number, polynomial interpolation...) which are well described in many classical books [19]. On the contrary, we focus on the multiple dimension case, without and with constraints. The main point for each algorithm presentation is the definition of the descent direction. Obviously for optimization based on complex flow simulation, the search of the descent step is always based on a cheap algorithm like polynomial interpolation.

Descent methods for unconstrained optimization

In this part, we focus on the search of an optimum of the objective function on the design space $D_\alpha=\mathbb{R}^{nf}$. The straight forward algorithm (steepest descent) that simply goes down the direction opposite the gradient of J is not described here, as it is known to often be inefficient. The conjugate gradient and BFGS methods are described.

Conjugate gradient methods

The idea behind Conjugate gradient methods comes from the analysis of the behavior of the steepest descent for the specific case of positive definite quadratic forms ($J(\alpha)=\frac{1}{2}\alpha^T H\alpha+b^T\alpha$).

In this case, the conditioning of the positive definite matrix H strongly affects the robustness and convergence speed of the steepest descent. To improve robustness, at step k Conjugate Gradient methods use a descent direction d^k orthogonal to d^{k-1} in the sense of H - $(d^{k-1})^T Hd^k=0$

$$d^k=-\nabla J(\alpha^k)+\beta^k d^{k-1} \qquad \beta^k=\frac{(\nabla J(\alpha^k))^T Hd^{k-1}}{(d^{k-1})^T Hd^{k-1}}.$$

Conjugate gradient algorithm

Set $k=0$ and $d^{-1}=0$, an initial point α^0 and a stopping tolerance ε
While $||\nabla J(\alpha^k)||>\varepsilon$ **do**
Compute $d^k=-\nabla J(\alpha^k)+\beta^k d^{k-1}$
Find t^* by line search on $q(t)=J(\alpha^k+td^k)$
Update current iterate $\alpha^{k+1}=\alpha^k+t^*d^k$ and set $k=k+1$
End while

Two formulas have been proposed for computation of β^k for extension to non quadratic cases. The first one is based on another formula of β^k in the quadratic positive definite case $\beta^k=\frac{||\nabla J(\alpha^k)||^2}{||\nabla J(\alpha^{k-1})||^2}$ which can be directly applied to a non-quadratic function as it does not refer to matrix H anymore. The second extension, proposed by Polak and Ribière in 1969, also reduces to the same algorithm in the quadratic positive definite case (see below, as then $\nabla J(\alpha^k)^T\nabla J(\alpha^{k-1})=0$). It is known to lead to a more efficient algorithm for specific applications.

$$\beta^k_{FR}=\frac{||\nabla J(\alpha^k)||^2}{||\nabla J(\alpha^{k-1})||^2} \qquad \beta^k_{PR}=\frac{||\nabla J(\alpha^k)||^2}{||\nabla J(\alpha^{k-1})||^2}-\frac{\nabla J(\alpha^k)^T\nabla J(\alpha^{k-1})}{||\nabla J(\alpha^{k-1})||^2}$$

A more efficient algorithm requires more information, in particular information about the second derivatives of the function to minimize. Newton methods have been devised for this reason.

Newton and quasi-Newton methods - Principle

The classical Newton method is originally a method to find the roots of the equation $z(\alpha)=0$ by approximating the function z by successive linear expansions. Starting from the current iterate α^k, substituting z by its linear approximation leads to $z(\alpha^k+d^k)=z(\alpha^k)+\nabla z(\alpha^k).d^k+o(||d^k||)$. Neglecting the term $o(||d^k||)$ yields $d^k=-[\nabla z(\alpha^k)]^{-1}.z(\alpha^k)$ and is used to calculate the next iterate $\alpha^{k+1}=\alpha^k+d^k$. The same method can be used as an optimization algorithm, by solving the Karush-Kuhn-Tucker optimality condition. In this case, the function $z(\alpha)$ becomes the gradient ∇J of the objective function J and the gradient ∇z, its Hessian $\nabla^2 J$. At each iteration k, the descent direction has to be computed by the formula $d^k=-[\nabla^2 J(\alpha^k)]^{-1}.\nabla J(\alpha^k)$

Newton algorithm

Set $k=0$, an initial point α^0 and a stopping tolerance ε
While $||\nabla J(\alpha^k)||>\varepsilon$ **do**
Compute $d^k=-[\nabla^2 J(\alpha^k)]^{-1}.\nabla J(\alpha^k)$
Update current iterate $\alpha^{k+1}=\alpha^k+t^*d^k$ and set $k=k+1$
End while

The important advantage of this method resides in its convergence in the region of the solution, which is superlinear in general and quadratic (at each iteration, the number of exact digits is doubled) if J has C^3 regularity.

Besides, the drawbacks of Newton's method are also well-known: (1) the Hessian is required: in most engineering problems, an explicit form of the objective function is unavailable. The Hessian must be computed numerically which requires a large number of the objective function evaluations; (2) in high dimensional spaces, the solution of the linear system at each iteration is very CPU demanding; (3) Newton methods diverge violently far from the optimal point; (4) this algorithm converges on the closest stationary point, not necessarily the global minimum. Quasi-Newton methods were developed to circumvent these drawbacks.

Considering this list of drawbacks, the quasi-Newton realizes an improvement over the above Newton method based on two main ideas:

- First, the stability problems of the method can be avoided by adding a line-search process in the algorithm. Actually, noting that the requirement $J(\alpha^{k+1})<J(\alpha^k)$ enforces stability, the Newton increment d^k is considered as a direction, along which a line-search is performed to reduce the function $q(t)=J(\alpha^k+td^k)$. It can be proved that the line-search is possible if and only if the Hessian of the objective function is positive definite.
- Secondly, rather than computing the Hessian matrix, its inverse is approximated by a matrix H which evolves during the iterations. This matrix can be chosen symmetric positive definite. An algorithm following this approach will be presented in the next section. Next algorithm describes the steps of a generic quasi-Newton algorithm.

Generic quasi-Newton algorithm

Set $k=0$, an initial point α^0, a stopping tolerance ε, an initial matrix $\overline{H}^0$ positive definite (generally, the identity matrix)
While $||\nabla J(\alpha^k)||>\varepsilon$ **do**
Compute $d^k=-\overline{H}^k\nabla J(\alpha^k)$
Make a line search for $q(t)=J(\alpha^k+td^k)$ initialized by $t=1$
Update current iterate $\alpha^{k+1}=\alpha^k+t^*d^k$
Compute new matrix $\overline{H}^{k+1}$
set $k=k+1$
End while

BFGS method

Let us now explain how the approximate invert of the Hessian, matrix $\overline{H}^k$ evolves during the iterations. The method presented here was introduced by C. Broyden, R. Fletcher, D. Goldfarb and D. Shanno. It is certainly the most popular quasi-Newton algorithm. The matrix

$\overline{H}^{k+1}$ is computed from $\overline{H}^{k}$ and some vectors attached to iteration k and $k+1$ computations according to following formulas

$$s^k = \alpha^{k+1} - \alpha^k$$
$$y^k = \nabla J(\alpha^{k+1}) - \nabla J(\alpha^k)$$
$$\overline{H}^{k+1} = \overline{H}^{k} - \frac{s^k (y^k)^T \overline{H}^k + \overline{H}^k y^k (s^k)^T}{(y^k)^T s^k} + \left[1 + \frac{(y^k)^T \overline{H}^k y^k}{(y^k)^T s^k}\right] \frac{s^k (s^k)^T}{(y^k)^T s^k}$$

Constrained optimization

Actually, most complex industrial problems lead to multi-dimensional constrained optimization. Two efficient algorithms for this task (feasible direction method and sequential quadratic programming) are described here. The presentation of classical methods that are seldom used for design in aeronautics (method of centers, sequential linear programming) can be found in [19]

Feasible direction method

The goal of feasible direction method is to build a sequence of points α^k such that $\alpha^{k+1} = \alpha^k + td^k$ where the displacement along direction d^k leads to lower values of both, the objective and the active constraints (all constraints satisfying $G_j(\alpha^k) = 0$). After d^k has been defined, the factor t is determined by a mono-dimensional optimization. Let us now derive the definition of d^k. As before, the indexes are the ones of the iteration (k) of the algorithm. The vector d^k must satisfy $\nabla J(\alpha^k).d^k \le 0$ and $\forall j / G_j(\alpha^k) = 0 \quad \nabla G_j(\alpha^k).d^k \le 0.$

The tricky point is determining the vector d^k ensuring the best descent. For a simple two dimensional problem ($n_f = 2$) with one active convex constraint G_1, it is easy to check that the minimization $[Min \nabla J(\alpha^k).d^k \le 0$ with $\nabla G_j(\alpha^k).d^k \le 0 \quad \forall j / G_j(\alpha^k) = 0]$ will lead to a vector d that points towards inadmissible states (see figure 2). To tackle this issue, scalar factors θ_j are included in the problem $[\nabla G_j(\alpha^k).d^k + \theta_j \le 0 \quad \theta_j > 0 \quad \forall j / G_j(\alpha^k) = 0]$

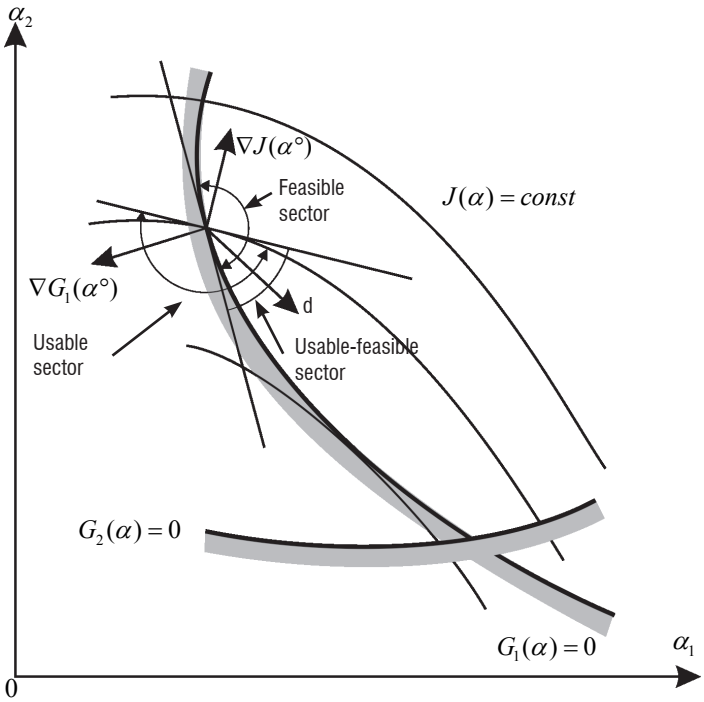

Figure 2 - Principle of feasible direction search

One wants to link the value of θ_j with the one of $\nabla J(\alpha^k).d$ Eventually, the search for the best direction for descent is reformulated as follows

Maximize β find d^k bounded
$\nabla J(\alpha^k).d^k + \beta \le 0$ and
$\nabla G_j(\alpha^k).d^k + \theta_j \beta \le 0 \quad \theta_j > 0 \quad \forall j / G_j(\alpha^k) = 0$

Obviously, if G_j is a linear constraint then $\theta_j = 0$ is suitable. For non linear constraints the simplest choice is $\theta_j = 1$. More complex formulas are presented in [19].

Feasible direction method

Set $k = 0$, an initial point α^0, a stopping tolerance ε
While KKT conditions not satisfied **do**
  Find d^k (bounded) which maximizes β subject to
  $\nabla J(\alpha^k).d^k + \beta \le 0$ and
  $\forall j / G_j(\alpha^k) = 0 \quad \nabla G_j(\alpha^k).d^k + \theta_j \beta \le 0.$
  Make a line search for $q(t) = J(\alpha^k + td^k)$
  Update current iterate $\alpha^{k+1} = \alpha^k + t^* d^k$ and set $k = k+1$
End while

Sequential Quadratic Programming (SQP)

In this method, an auxiliary function of the descent vector d is introduced. It is a quadratic approximation of $J(\alpha + d)$. The heart of the algorithm reads

$$\text{Minimize } Q(d) = J(\alpha^k) + \nabla J(\alpha).d + \frac{1}{2} d^T B^k d$$
$$\text{Subject to } \nabla G_j(\alpha^k).d^k + \delta_j G_j(\alpha^k) \le 0.$$

where B is a positive definite matrix equal to the identity matrix I at the first step and to an approximation of the hessian for the next iterations. Parameter δ_j is taken equal to 1 if the constraint is strictly respected ($G_j(\alpha^k) \le 0.$) and to a value in [0,1] if the current design point α^k violates constraint $G_j(\alpha)$. See reference [19], for more information about hessian estimation, and monodimensional search after determination of d.

Sequential quadratic programming

Set $k = 0$, an initial point α^0, a stopping tolerance ε, an initial approximate Hessian matrix B^0
While KKT conditions not satisfied **do**
  Find d^k which minimizes
  $Q(d) = J(\alpha^k) + \nabla J(\alpha).d + 0.5 d^T B^k d$
  Subject to $\nabla G_j(\alpha^k).d^k + \delta_j G_j(\alpha^k) \le 0.$
  Update current iterate $\alpha^{k+1} = \alpha^k + d^k$ and set $k = k+1$
  Build B^{k+1}, for example from BFGS formula
End while

Sensitivity evaluations for descent methods

This section will describe in some detail the main methods of gradient evaluation, firstly finite dif-ferences, followed by discrete versions of the direct and adjoint methods. A detailed bibliography of all of the methods presented and more material (continuous adjoint, exact

duality, second-order derivatives, frozen turbulent-viscosity assumption…) can be found in [12].

Finite differences

The application of finite differences to an entire flow solver is by far the simplest means of obtaining solution gradients, as it requires no modification of the solver itself. As a result it was one of the first sensitivity evaluation methods to be used. To start, the numerical fiow solution corresponding not only to α but also to perturbed states $\alpha+\delta\alpha$ and possibly $\alpha-\delta\alpha$ is calculated. For the typical case of $\delta\alpha$ representing a geometry modification, this implies a mesh deformation $X(\alpha+\delta\alpha)$ and a new flow solution on the modified mesh satisfying $R(W(\alpha+\delta\alpha),X(\alpha+\delta\alpha))=0$. An approximation of objective functions derivatives in the direction $\delta\alpha$ can then be computed using a finite difference formula, such as the second-order accurate formula

$$\frac{dJ(\alpha)}{d\alpha}=\frac{\overline{J}(W(\alpha+\delta\alpha),X(\alpha+\delta\alpha))-\overline{J}(W(\alpha+\delta\alpha),X(\alpha+\delta\alpha))}{2\delta\alpha}$$

The entire matrix $(dJ(\alpha)/d\alpha\,,dG_1(\alpha)/d\alpha\,\ldots)$ may be evaluated at a cost of $2n_f$ flow solutions, or if a first-order difference is used n_f+1 flow solutions, making the method impractical for large design spaces. Another serious disadvantage is that the choice of the step size $\delta\alpha$ is critical to the accuracy of the result. The fundamental limitations of finite differences have lead to the investigation of alternative means of gradient evaluation.

The discrete direct method

Under the assumption that the discrete residual R is once continuously differentiable with respect to the flow field and mesh in a neighborhood of $W(\alpha)$ and $X(\alpha)$, the discrete form of the governing equations $R(W,X)=0$ may be differentiated with respect to each component of α to give

$$\left(\frac{\partial R}{\partial W}\right)\frac{dW}{d\alpha_i}=-\left(\frac{\partial R}{\partial X}\frac{dX}{d\alpha_i}\right) \qquad (1)$$

This may be regarded as a linear system in unknowns $\frac{dW}{d\alpha_i}$, where $\frac{dX}{d\alpha_i}$ (or even) may be evaluated by finite differences as $\left(\frac{\partial R}{\partial X}\frac{dX}{d\alpha_i}\right)$

in the previous section, and the partial derivatives could be evaluated for example by hand. The dimension of the system is the number of degrees of freedom in the non-linear equations n_W, and it can be regarded as a linearization of those equations. Given the n_f solutions $\frac{dW}{d\alpha_i}$ the derivatives of the functions of interest are

$$\frac{dJ}{d\alpha_i}=\frac{\partial\overline{J}}{\partial X}\frac{dX}{d\alpha_i}+\frac{\partial\overline{J}}{\partial W}\frac{dW}{d\alpha_i} \quad\text{and}\quad \frac{dG_k}{d\alpha_i}=\frac{\partial\overline{G_k}}{\partial X}\frac{dX}{d\alpha_i}+\frac{\partial\overline{G_k}}{\partial W}\frac{dW}{d\alpha_i}$$

where again the partial derivatives are in principle easy to evaluate, as J is a known, explicit function of W and X. Hence the 2 n_f non-linear solutions required for second-order flow finite differences have been replaced by one non-linear and n_f linear solutions, all of dimension n_W, and the dependence on finite differences has been confined to the relatively cheap mesh update procedure.

The discrete adjoint method

There are many ways to derive the discrete adjoint equations, the one given here is chosen for its similarity to the derivation of the continuous adjoint. Let the direct linearization (1) be premultiplied by an arbitrary line vector λ of dimension n_W, so that

$$\forall\,\lambda\in\mathbb{R}^{n_W}\quad \lambda^T\left(\frac{\partial R}{\partial W}\right)\frac{dW}{d\alpha_i}+\lambda^T\left(\frac{\partial R}{\partial X}\frac{dX}{d\alpha_i}\right)=0.$$

Adding this expression to last but one equation of previous subsection,

$$\forall\,\lambda\in\mathbb{R}^{n_W}\quad \frac{dJ}{d\alpha_i}=\frac{\partial\overline{J}}{\partial X}\frac{dX}{d\alpha_i}+\frac{\partial\overline{J}}{\partial W}\frac{dW}{d\alpha_i}+\lambda^T\left(\frac{\partial R}{\partial W}\right)\frac{dW}{d\alpha_i}+\lambda^T\left(\frac{\partial R}{\partial X}\frac{dX}{d\alpha_i}\right),$$

Hence the term $\frac{dW}{d\alpha_i}$ may be eliminated by choosing the arbitrary vector λ to satisfy

$$\lambda^T\left(\frac{\partial R}{\partial W}\right)+\left(\frac{\partial\overline{J}}{\partial W}\right)=0 \quad\text{or}\quad \left(\frac{\partial R}{\partial W}\right)^T\lambda=-\left(\frac{\partial\overline{J}}{\partial W}\right)^T \qquad (2)$$

the adjoint equation, a linear system in unknowns λ the adjoint solution, with respect to the objective function J. Given λ the sensitivities may be written

$$\frac{dJ}{d\alpha_i}=\frac{\partial\overline{J}}{\partial X}\frac{dX}{d\alpha_i}+\lambda^T\left(\frac{\partial R}{\partial X}\frac{dX}{d\alpha_i}\right).$$

The critical point is that, because α does not appear in (A), that linear system must only be solved once for each function to be differentiated. Hence the full matrix $(dJ(\alpha)/d\alpha\,,\ dG_1(\alpha)/d\alpha\,\ldots)$ may be evaluated at a cost of $1+n'_c$ linear system solutions of size n_W, substantially independent of n_f (where n'_c is the number of active constraints at a specific step of the optimization process ; only the gradients of those constraints are then needed).

Robustness enhancement using recursive projection method

Both linearized and adjoint equations (equations (1) and (2)) are large sparse linear systems which can not be directly inverted for large size simulations. Most often an iterative procedure involving an approximate Jacobian $(\partial R/\partial W)^{(APP)}$ is used. The corresponding algorithm reads (for the adjoint method):

$$\left(\frac{\partial R}{\partial W}\right)^{(APP)T}(\lambda^{(l+1)}-\lambda^{(l)})=-\left(\frac{\partial\overline{J}}{\partial W}\right)^T-\left(\frac{\partial R}{\partial W}\right)^T\lambda^{(l)}$$

Hence linear systems that appear in discrete gradient computation are very often solved by an iteration procedure of the form:

$$x^{(l+1)}=F\left(x^{(l)}\right)=\Phi x^{(l)}+M^{-1}b \qquad (3)$$

Algorithm (3) aims at solving $Ax=b$. The matrix $\Phi=I-M^{-1}A$ represents the iteration matrix of the numerical scheme whereas M denotes a preconditioning matrix.

Whether or not the iteration procedure converges to the solution $x = A^{-1}b$ depends upon the eigenvalues of Φ. Here, we follow the Recursive Projection Method (RPM) introduced by Shroff and Keller [15] for the stabilization of unstable recursive fixed point iteration procedures.

Suppose the iteration diverges thus implying that there are $m \geq 1$ eigenvalues of Φ with a modulus greater than unity:

$$|\lambda_1| \geq \ldots \geq |\lambda_m| \geq 1 .$$

Define the subspace $\mathbf{P} = span\{e_1, \ldots, e_m\}$ spanned by the eigenvectors associated to these eigenvalues and $\mathbf{Q} = \mathbf{P}^{\perp}$ its orthogonal complement in $\mathbb{R}^N$. These subspaces form a direct sum in $\mathbb{R}^N$, hence every vector can be decomposed in a unique way as the sum

$$\forall x \in \mathbb{R}^N, \quad \exists x_P \in \mathbf{P},\ x_Q \in \mathbf{Q} : x = x_P + x_Q .$$

The orthogonal projectors onto the subspaces $\mathbf{P}$ and $\mathbf{Q}$ are denoted P and Q respectively. Let $V \in \mathbb{R}^N$ be a matrix whose columns constitute an orthonormal basis for P, then the projectors are defined by $\mathbf{P} = VV^T, \mathbf{Q} = I - P$ where I denotes the identity matrix and exponent T is the transpose operator. Note that since $\mathbf{P}$ is an invariant subspace of Φ we have $Q\Phi P = 0$. Then, the RPM iteration reads

$$\begin{aligned} x_Q^{(l+1)} &= QF\left(x^{(l)}\right) \\ x_P^{(l+1)} &= x_P^{(l)} + (I - P\Phi P)^{-1}\left(PF\left(x^{(l)}\right) - x_P^{(l)}\right) \\ x^{(l+1)} &= x_Q^{(l+1)} + x_P^{(l+1)} \end{aligned} \qquad (4)$$

According to Shroff and Keller, the RPM iteration converges even when the original iteration (3) diverges [15]. The implementation of the algorithm (4) requires the construction of the projectors P and Q and hence the evaluation of the orthonormal basis.

The method for computing V is the following: Consider the vector $v_1 = \Delta x_Q^{(l-k+1)}$ where $\Delta x_Q^{(j)} = x_Q^{(j+1)} - x_Q^{(j)}$ and the matrix $\hat{A} = Q\Phi Q$. Define the Krylov subspace of dimension k generated by v_1 and $\hat{A} : \mathbf{K}_k = span\{v_1, \hat{A}v_1, \ldots, \hat{A}^{k-1}v_1\}$. Let $K_k = (v_1, \hat{A}v_1, \ldots, \hat{A}^{k-1}v_1)$ a matrix whose columns span the subspace $\mathbf{K}_k$. Compute the QR factorization $K_k = Q_k R_k$ where the columns of Q_k form an orthonormal basis of $\mathbf{K}_k$ and where the absolute values of the diagonal elements of the upper triangular matrix R_k are sorted in a decreasing order.

If the algorithm (3) fails to converge, vectors for V are chosen on the basis of the following condition. For the largest $1 \leq j \leq k-1$ such that

$$\left| \frac{R_{j,j}}{R_{j+1,j+1}} \right| > \kappa$$

the first j columns of Q_k are added to V. The parameter κ stands for the Krylov acceptance ratio and its inverse represents a bound for the residual of the computed eigenspace of $\hat{A}$.

Adjoint method for aeroelasticity

When gradient computation is carried out in an aeroelastic framework, the state variables can be divided into two parts: W and D, where W stands for the aerodynamic conservative variables at the center of the cells of the volume mesh $X(\alpha)$ and D represents the structural displacements of the nodes of the structural model associated with the solid body considered. In addition, a new mesh $Z(\alpha)$ must be introduced for the structural model. The structural mesh can depend a priori on the vector of shape parameters. However, if the solid body planform remains unchanged when α varies within the design space, this dependency vanishes. The state variables and meshes satisfy the discrete equations of both fluid mechanics (R_a) and structural mechanics (R_s),

$$\begin{cases} R_a(W, D, X(\alpha), Z(\alpha)) = 0 \\ R_s(W, D, X(\alpha), Z(\alpha)) = 0 \end{cases} .$$

In general, R_a is a set of nonlinear equations (of size n_a) but R_s is a set of linear equations (of size n_s). These equations are coupled through the aerodynamic loads L_s which stimulate the structural model:

$$L_s = L_s(W, D, X, Z),$$

and through D_a the displacements of which are transmitted backwards to the volume mesh so as to follow the structural displacements (this can be done in a two-step manner using surface mesh transfer intermediately):

$$D_a = D_a(D, Z).$$

The current volume mesh $X(\alpha)$ is actually a function of the initial volume mesh $X_0(\alpha)$ on which the structural displacements are transferred, and which, incidentally, is actually directly controlled by α,

$$X = X(X_0, D_a(D, Z)).$$

These couplings introduce a number of issues regarding load transfer consistency and conservativeness first, and mesh deformation second, which have been extensively dealt with in literature. The reader can, for instance, refer to the articles of Farhat et al., Maman et al., Arian and Smith et al[1].

Assuming that this coupled system has C^1 regularity too with respect to its four vector arguments, that $W(\alpha)$ has $D(\alpha)$ regularity, and that,

$$\begin{cases} R_a(W(\alpha_i), D(\alpha_i), X(\alpha_i), Z(\alpha_i)) = 0 \\ R_s(W(\alpha_i), D(\alpha_i), X(\alpha_i), Z(\alpha_i)) = 0 \end{cases},$$

$$\det\left[\begin{pmatrix} \partial R_a / \partial W & \partial R_a / \partial D \\ \partial R_s / \partial W & \partial R_s / \partial D \end{pmatrix} (W(\alpha_i), D(\alpha_i), X(\alpha_i), Z(\alpha_i)) \right] \neq 0$$

the implicit function theorem allows us to define W and D as C^1 functions of X and Z in a neighborhood of $X(\alpha_i)$ and $Z(\alpha_i)$, and therefore thanks to the assumed regularity of $X(\alpha)$ and $Z(\alpha)$, as functions of α in a neighborhood of α_i. We suppose this property to be true on the entire design domain D_α, so that we may use $W(\alpha)$ and $D(\alpha)$ notation, and rewrite the discrete coupled equations

$$\begin{cases} R_a(W(\alpha), D(\alpha), X(\alpha), Z(\alpha)) = 0 \\ R_s(W(\alpha), D(\alpha), X(\alpha), Z(\alpha)) = 0 \end{cases} .$$

We assume that the discrete residual (R_a, R_s) is C^1 differentiable with respect to the state variables (W, D) and to the meshes (X, Z) in a neighborhood of $(W(\alpha_i), D(\alpha_i))$ and $(X(\alpha_i), Z(\alpha_i))$, so that the discrete form of the discrete coupled equations can be differentiated with respect to α:

[1] A detailed bibliography including these authors can be found in [9]

$$\begin{cases} \dfrac{\partial R_a}{\partial W}\dfrac{dW}{d\alpha} = -\dfrac{\partial R_a}{\partial D}\dfrac{dD}{d\alpha} - \dfrac{\partial R_a}{\partial X}\dfrac{dX}{d\alpha} - \dfrac{\partial R_a}{\partial Z}\dfrac{dZ}{d\alpha} \\ \dfrac{\partial R_s}{\partial D}\dfrac{dD}{d\alpha} = -\dfrac{\partial R_s}{\partial W}\dfrac{dW}{d\alpha} - \dfrac{\partial R_s}{\partial X}\dfrac{dX}{d\alpha} - \dfrac{\partial R_s}{\partial Z}\dfrac{dZ}{d\alpha} \end{cases}$$

Given the n_f solutions $(dW/d\alpha, dD/d\alpha)$ of the linear coupled system above, the derivatives of J, are:

$$\frac{dJ}{d\alpha} = \frac{\partial J}{\partial W}\frac{dW}{d\alpha} + \frac{\partial J}{\partial X}\left(\frac{\partial X}{\partial X_0}\frac{dX_0}{d\alpha} + \frac{\partial X}{\partial D_a}\left(\frac{dD_a}{dD}\frac{dD}{d\alpha} + \frac{dD_a}{dZ}\frac{dZ}{d\alpha}\right)\right).$$

As for the aerodynamic adjoint method, the discrete coupled equations are multiplied by two arbitrary line vectors Λ_a^T (of size n_a) and Λ_s^T (of size n_s), so that:

$$\begin{cases} \Lambda_a^T\dfrac{\partial R_a}{\partial W}\dfrac{dW}{d\alpha} + \Lambda_a^T\dfrac{\partial R_a}{\partial D}\dfrac{dD}{d\alpha} + \Lambda_a^T\dfrac{\partial R_a}{\partial X}\dfrac{dX}{d\alpha} + \Lambda_a^T\dfrac{\partial R_a}{\partial Z}\dfrac{dZ}{d\alpha} = 0 \\ \Lambda_s^T\dfrac{\partial R_s}{\partial D}\dfrac{dD}{d\alpha} + \Lambda_s^T\dfrac{\partial R_s}{\partial W}\dfrac{dW}{d\alpha} + \Lambda_s^T\dfrac{\partial R_s}{\partial X}\dfrac{dX}{d\alpha} + \Lambda_s^T\dfrac{\partial R_s}{\partial Z}\dfrac{dZ}{d\alpha} = 0 \end{cases},$$
$$\forall (\Lambda_a, \Lambda_s) \in (\mathbb{R}^{n_a}, \mathbb{R}^{n_s}).$$

Adding both these equations to the derivatives of J,

$$\begin{aligned} \frac{dJ}{d\alpha} &= \left(\frac{\partial J}{\partial W} + \Lambda_a^T\frac{\partial R_a}{\partial W} + \Lambda_s^T\frac{\partial R_s}{\partial W}\right)\frac{dW}{d\alpha} \\ &+ \left(A\frac{\partial X}{\partial D_a}\frac{dD_a}{dD} + \Lambda_a^T\frac{\partial R_a}{\partial D} + \Lambda_s^T\frac{\partial R_s}{\partial D}\right)\frac{dD}{d\alpha} \\ &+ A\frac{\partial X}{\partial X_0}\frac{dX_0}{d\alpha} + A\frac{\partial X}{\partial D_a}\frac{dD_a}{dZ}\frac{dZ}{d\alpha} \\ &+ \left(\Lambda_a^T\frac{\partial R_a}{\partial Z} + \Lambda_s^T\frac{\partial R_s}{\partial X}\frac{\partial X}{\partial D_a}\frac{dD_a}{dZ} + \Lambda_s^T\frac{\partial R_s}{\partial Z}\right)\frac{dZ}{d\alpha}, \\ &\forall (\Lambda_a, \Lambda_s) \in (\mathbb{R}^{n_a}, \mathbb{R}^{n_s}), \end{aligned}$$

with $A = \left(\dfrac{\partial J}{\partial X} + \Lambda_a^T\dfrac{\partial R_a}{\partial X} + \Lambda_s^T\dfrac{\partial R_s}{\partial X}\right)$.

The so-called adjoint vectors Λ_a and Λ_s are chosen such that each term containing $dW/d\alpha$ or $dD/d\alpha$ disappears from the expression of $dJ/d\alpha$. This leads to the adjoint coupled equations:

$$\begin{cases} \left(\dfrac{\partial R_a}{\partial W}\right)^T\Lambda_a = -\left(\dfrac{\partial J}{\partial W}\right)^T - \left(\dfrac{\partial R_s}{\partial W}\right)^T\Lambda_s \\ \left(\dfrac{\partial R_s}{\partial X}\dfrac{\partial X}{\partial D_a}\dfrac{dD_a}{dD} + \dfrac{\partial R_s}{\partial D}\right)^T\Lambda_s = -\left(\dfrac{\partial J}{\partial X}\right)^T \\ \qquad\qquad -\left(\dfrac{\partial R_a}{\partial X}\dfrac{\partial X}{\partial D_a}\dfrac{dD_a}{dD} + \dfrac{\partial R_a}{\partial D}\right)^T\Lambda_a \end{cases}$$

Once the adjoint vectors Λ_a and Λ_s have been solved from the linear coupled system above, the derivatives of J are:

$$\begin{aligned} \frac{dJ}{d\alpha} &= \left(\frac{\partial J}{\partial X} + \Lambda_a^T\frac{\partial R_a}{\partial X} + \Lambda_s^T\frac{\partial R_s}{\partial X}\right)\frac{\partial X}{\partial X_0}\frac{dX_0}{d\alpha} \\ &+ \left(\frac{\partial J}{\partial X} + \Lambda_a^T\frac{\partial R_a}{\partial X} + \Lambda_s^T\frac{\partial R_s}{\partial X}\right)\frac{\partial X}{\partial D_a}\frac{dD_a}{dZ}\frac{dZ}{d\alpha} \\ &+ \left(\Lambda_a^T\frac{\partial R_a}{\partial Z} + \Lambda_s^T\frac{\partial R_s}{\partial X}\frac{\partial X}{\partial D_a}\frac{dD_a}{dZ} + \Lambda_s^T\frac{\partial R_s}{\partial Z}\right)\frac{dZ}{d\alpha}. \end{aligned}$$

Function J has been assumed to depend only on the aerodynamic variables W and X. This does not reduce the generality of the method; indeed, one can easily introduce a direct dependence of J on D and Z, as long as the partial derivatives $\partial J/\partial D$ and $\partial J/\partial Z$ are available.

The interested reader is invited to consult the classical literature on the subject, including in particular the articles of Sobieszcsanski-Sobieski, Farhat, Maute, Martins and Haftka[1].

Conclusion

Optimization in aeronautics has been a very active topic since the 70's. Several engineering disciplines, like structure and aerodynamics, have benefited from the progress accomplished in this field, at the crossroad of mathematics, numerics and computer science. The purpose of this article was not to present all of the methods that can be used for design in aeronautics, but only those that were found efficient and actually used at Onera for design with a special focus on aerodynamic design applications [14].

Huge advances have been made during the last twenty years, in particular in the fields of global optimization (see § Global search methods) and gradient computation (see § Sensitivity evaluations for descent methods), paving the way for the first concrete and conclusive applications of these methods for solving design problems, capturing, at least partially, the complexity of reallife industrial design problems. Future advances may concern the efficient coupling of flow analysis and shape optimization, also of global and local optimization algorithms ■

[1] A detailed bibliography including these authors can be found in [9]

Acknowledgements

The authors are deeply grateful to their colleagues Manuel Bompard (Onera/DSNA), S. Burguburu (Onera/DAAP) and R.P. Dwight (TU-Delft) for their contributions, encouragements and the sharing of ideas.

References

[1] M. BOMPARD, J. PETER - *Local Search Methods for Design in Aeronautics.* In RTO-AVT-167 Lecture Series "Strategies for optimization and automated design of gas turbine engines". 2010.

[2] G. CARRIER - *Multi-disciplinary Optimization of a Supersonic Transport Aircraft Wing Plan-form.* ECCOMAS 2004, European Congress on Computational Methods in Applied Sciences and Engineering, Jyväskylä, 24-28 July 2004.

[3] G. CARRIER - *Single and Multipoint Aerodynamic Optimizations of a Supersonic Transport Aircraft Wing Using Optimization Strategies Involving Adjoint Method and Genetic Algorithm.* ERCOFTAC Conference, Las Palmas, 5-7 April 2006.

[4] J.-L. DENEUBOURG, J.-M. PASTEELS, J.-C. VERHAEGHE - *Probabilistic Behaviour in Ants: a Strategy of Errors?* Journal of Theoretical Biology, 105, 1983.

[5] D.E. GOLDBERG - *Computeraided Gas Pipeline Operation Using Genetic Algorithms and Rule Learning.* Doctoral dissertation, University of Michigan, Ann Arbor, MI. (University Micro-films No. 8402282), 1983.

[6] J. H. HOLLAND - *Adaptation in Natural and Artificial Systems.* Ann Arbor, MI: University of Michigan Press, 1975.

[7] S. KIRKPATRICK, C.D. GELLAT, M.P. VECCHI - *Optimization by Simulated Annealing.* Science, 220(4598), 671-680, 1983.

[8] J. KENNEDY, R. EBERHART - *Particle Swarm Optimization.* In Proceedings IEEE international conference on neural networks Vol. 4, p.1942-1948, 1995.

[9] M. MARCELET J. PETER, G. CARRIER - *Sensitivity Analysis of a Strongly Coupled System Using the Discrete Direct and Adjoint Approach.* Revue Européenne de mécanique numérique. Vol. 17, 1077-1106, 2008.

[10] J. MÖLLER - *Studies of Recursive Projection Methods for Convergence Acceleration of Steady State Calculations.* Licenciate Thesis, Royal Institute of Technology (KTH), TRITA-NA-0119, 2001.

[11] V. PARETO - *Cours d'économie politique*. Lausanne. F. Rouge. 1896/97.

[12] J. PETER, R.P. DWIGHT - *Numerical Sensitivity Analysis: a Survey of Approaches.* Computers and Fluids 39 (2010) 373–391.

[13] K. RASHEED - *GADO: A Genetic Algorithm for Continuous Design Optimization*. Tech. Report DCS-TR-352, Dept. of Computer Science, Rutgers, The State University of New-Jersey, 1988, Ph D. Thesis.

[14] J. RENEAUX, Ph. BEAUMIER, Ph. GIRODROUX-LAVIGNE - *Advanced Aerodynamic Applications with the elsA Software.* Aerospace Lab. Issue 2. 2011.

[15] G.M. SHROFF, H.B. KELLER - *Stabilisation of Unstable Procedures: The Recursive Projection Method.* SIAM Journal of Num. analysis. Vol 30, 1099-1120, 1993.

[16] R. STORN, K. PRICE - *Differential Evolution - a Simple and Efficient Heuristic for Global Optimization Over Continuous Spaces.* Journal of Global Optimization 11: 341-359, 1997.

[17] R. STORN - *On the Usage of Differential Evolution for Function Optimization.* Biennial Conference of the North American Fuzzy Information Processing Society (NAFIPS). pp. 519-523, 1996.

[18] G. VANDERPLAATS, R. HICKS - *Numerical Airfoil Optimization Using a Reduced Number of Design Coordinates.* Tech. Rep. TMX 73151, NASA (1976).

[19] G. VANDERPLAATS - *Numerical Optimization Techniques for Engineering Design: with Applications*. McGraw Hill, 1984.

Acronyms

ANN	(Artificial Neural Network)
BFGS	(Broyden-Fletcher-Goldfarb-Shanno method)
CCD	(Central Composite Design)
CFD	(Computational Fluid Dynamics)
CPU	(Central Processing Unit)
EI	(Expected Improvement)
GA	(Genetic Algorithm)
PSO	(Particle Swarm Optimization)
RaNS	(Reynolds averaged Navier-Stokes)
RPM	(Recursive Projection Method)
SA	(Simulated Annealing)
SQP	(Sequential Quadratic Programming)

Jacques Peter After graduating from Ecole Polytechnique (1992), he received a PhD in mechanics from Ecole des Arts et Métiers (1996). He has has been working in the CFD Department of Onera since 1998. He is presently in charge of numerical methods for optimization and control for this department.

Gérald Carrier is currently working as a research engineer in the Applied Aero-dynamic department of Onera. In charge of several research projects related to numerical optimisation, he has an experience of more than 10 years in applying local and global optimization methods for aerodynamics and multi-disciplinary design problems.

Didier Bailly graduated from Estaca in 1985, he was awarded a PhD in 1996 from the University PARIS VI in the field of near-critical point fluid hydrodynamics. His current activities are in the field of drag extraction methods from numerical or experimental data and the use of surrogate models in optimization.

Patricia Klotz obtained a PhD from SupAero in 1991. She joined the Onera Mathematical Modelling and Numerical Simulation group in 1995 where she focused on surrogate based optimization applied to combustion chambers design. Currently she is in charge of an internal research project on surrogate modeling.

Meryem Marcelet After graduating from SupAero (2005) and the Georgia Institute of Technology of Atlanta (2005), she received a PhD in mechanics from Ecole des Arts et Metiers (2008). She has been working since 2008 in the CAE Department of Onera in the domain of analytical gradient evaluation for aeroelastic problems and multidisciplinary optimization.

Florent Renac Florent Renac is currently research scientist at the Computational Fluid Dynamics and Aeroacoustics Department of Onera. He graduated from the École Normale Supérieure de Cachan in 1999 and received a PhD degree from Univeristy Paris VI in 2004 (in collaboration with Fundamental and Experimental Aerodynamics Department of Onera). His research activity includes aerodynamic sensitivity analysis, high-order discretization methods and robust time discretization.