AIAA JOURNAL
Vol. 48, No. 5, May 2010

Comparison of Gradient-Based and Gradient-Enhanced Response-Surface-Based Optimizers

J. Laurenceau[*]
Centre Européen de Recherche et Formation Avancées en Calcul Scientifique, 31057 Toulouse, France
M. Meaux[†]
Airbus Industries, 31060 Toulouse, France
M. Montagnac[‡]
Centre Européen de Recherche et Formation Avancées en Calcul Scientifique, 31057 Toulouse, France
and
P. Sagaut[§]
Université Pierre et Marie Curie, 75252 Paris cedex 05, France

DOI: 10.2514/1.45331

This paper deals with aerodynamic shape optimization using a high-fidelity solver. Because of the computational cost needed to solve the Reynolds-averaged Navier–Stokes equations, the performance of the shape must be improved using very few objective function evaluations, despite the high number of design variables. In our framework, the reference algorithm is a quasi-Newton gradient optimizer. An adjoint method inexpensively computes the sensitivities of the functions, with respect to design variables, to build the gradient of the objective function. As usual, aerodynamic functions show numerous local optima when the shape varies, and a more global optimizer is expected to be beneficial. Consequently, a kriging-based optimizer is set up and described. It uses an original sampling refinement process that adds up to three points per iteration by using a balancing between function minimization and error minimization. To efficiently apply this algorithm to high-dimensional problems, the same sampling process is reused to form a cokriging (gradient-enhanced model) based optimizer. A comparative study is then described on two drag-minimization problems depending on 6 and 45 design variables. This study was conducted using an original set of performance criteria, characterizing the strength and weakness of each optimizer in terms of improvement, cost, exploration, and exploitation.

Nomenclature

A	=	amplitude of the Hicks–Henne bump function
$C(x)$	=	sampling refinement criterion
Cd	=	drag coefficient
Cd_f	=	friction drag coefficient
Cd_i	=	induced drag coefficient
Cd_p	=	pressure drag coefficient
Cd_{vp}	=	viscous pressure drag coefficient
Cd_w	=	wave drag coefficient
Cl	=	lift coefficient
c	=	mean chord length
$\mathcal{D}$	=	domain of design variables
F_s	=	exact function at samples, N
$F(x)$	=	objective function
$H(x)$	=	Hessian matrix of the objective function
$HH(x)$	=	Hicks–Henne deformation function
$I(f)$	=	improvement of the objective function, %
N	=	order of the correlation matrix/quantity of information
n_{aug}	=	number of gradient-augmented samples
n_{dv}	=	number of design variables
n_{eval}	=	number of function evaluations
n_{grad}	=	number of gradient evaluations
n_{iter}	=	number of iterations of the optimization process
n_{pop}	=	optimizer population size
n_s	=	number of samples
$\mathcal{P}$	=	set of possible optima on the response surface
R	=	correlation matrix, $N \times N$
$r(x)$	=	correlation vector, N
$\mathcal{S}$	=	sample data set
$\mathrm{scf}(.)$	=	spatial correlation function
$S(x)$	=	standard error
s^i	=	ith sample
x	=	vector of design variables, n_{dv}
β	=	zero-order regression model
θ	=	spatial correlation function coefficients, n_{dv}
σ^2	=	model variance
$\lVert \cdot \rVert$	=	two norm

Subscripts

c	=	$\in [1,\ n_{\mathrm{eval}}]$
i	=	$\in [1,\ n_s]$
j	=	$\in [1,\ n_s]$
k	=	$\in [1,\ n_{\mathrm{iter}}]$
r	=	$\in [1,\ n_{\mathrm{pop}}]$
v	=	$\in [1,\ n_{\mathrm{dv}}]$

Superscripts

ini	=	initial sampling database
ref	=	best known value at current iteration (iteration k)

Presented as Paper 1889 at the 49th AIAA/ASME/ASCE/AHS/ASC Structures, Structural Dynamics, and Materials Conference and 16th AIAA/ASME/AHS Adaptive Structures Conference 10t, Schaumburg, IL, 7–10 April 2008; received 7 May 2009; revision received 4 February 2010; accepted for publication 6 February 2010. Copyright © 2010 by the American Institute of Aeronautics and Astronautics, Inc. All rights reserved. Copies of this paper may be made for personal or internal use, on condition that the copier pay the $10.00 per-copy fee to the Copyright Clearance Center, Inc., 222 Rosewood Drive, Danvers, MA 01923; include the code 0001-1452/10 and $10.00 in correspondence with the CCC.

[*]Postdoctoral Research Fellow, Computational Fluid Dynamics Department, 42 Avenue Gaspard Coriolis; julien.laurenceau@cerfacs.fr.

[†]Engineer, Aerodynamics Department, 316 Route de Bayonne.

[‡]Research Engineer, Computational Fluid Dynamics Department, 42 Avenue Gaspard Coriolis.

[§]Professor, Institut Jean Le Rond d'Alembert, 4 place Jussieu–Case 162.

$\hat{\cdot}$	=	approximated value
$\bar{\cdot}$	=	scaled value

I. Introduction

IN THE field of aerodynamic aircraft design, the studied functions are very sensitive to small changes on the shape, and it is particularly hard for designers to reach an optimal solution by trial and error. Numerical shape-optimization tools are thus particularly favored by aerodynamicists. These tools form a completely automatic process capable of handling expensive numerical computer simulations, given some degrees of freedom on the geometry and a figure of merit qualifying the performance of the shape. In the context of detailed design, a single optimization involves a few hundred resolutions of the Reynolds-Averaged Navier–Stokes (RANS) equations on three-dimensional meshes. To limit the computational cost and wall-clock time of the process, the optimization algorithm must then be carefully chosen. It should improve the objective function as much as possible, given a limited number of function evaluations, in spite of a high number of design variables. The choice of the optimizer reflects a compromise between the amount of improvement of the objective function and the computational cost (or time).

The quickest optimization algorithms use gradient information to converge along a descent path, departing from a baseline shape to a local optimum. The first optimizations conducted by Hicks and Henne [1] demonstrated the interest of gradient-based optimizers in the field of numerical aerodynamic shape design. This type of process is still currently in use [2–5] with the addition of efficient gradient calculation techniques, such as the adjoint method that enables the computation of the sensitivity of the objective function with respect to the design variables at a computational cost independent of the number of design variables. Despite their speed of convergence, gradient-based algorithms are known to be easily trapped by local optima.

The nonlinear physical phenomena occurring in transonic flows imply numerous local optima on aerodynamic functions (e.g., drag, lift, and momentum). Because of this fact, the local optimum obtained with gradient-based optimizers could certainly be further improved by using more global optimizers, such as genetic algorithms. However, this type of algorithm would require thousands of function evaluations, and the related computational cost is not sustainable when dealing with three-dimensional Navier–Stokes analysis. The use of global optimizers was made possible more recently, thanks to the use of surrogate models [6–10] that approximated the expensive computational-fluid-dynamics (CFD) function by a black-box model, which is inexpensive to evaluate. Numerous surrogate-model management methods exist (variable fidelity, trust region, sampling refinement, etc.). In the simplest method, a single surrogate model is used during the entire process. The model is built once at the beginning of the process, and then it completely replaces the CFD solver [11]. The success of this strategy depends largely on the accuracy of the global response surface (RS). It is only applicable to low-dimensional problems, because reaching a high density of sampling points becomes too computationally expensive when the dimension increases (curse of dimensionality). When dealing with high-dimensional problems, it is preferable to use a hybrid algorithm to perform some of the evaluations with the CFD solver and the remaining evaluations on the surrogate model. In this paper, the hybridization technique retained is the iterative sampling refinement. At each iteration of the process, a global optimum is determined on the surrogate model and is then validated through a CFD evaluation. The new CFD evaluation is then added to the sample database, and the surrogate model is updated.

The main goal of this paper is to present an original optimizer based on iterative sampling refinement and to assess its efficiency for aerodynamic design. This algorithm is original for several reasons. First, it combines multiple sampling refinement criteria within the same process iteration. Second, it uses a variant of the cokriging (gradient-enhanced kriging) method [12], referred to as sample-limited cokriging, to build the RSs. This formulation was set up to overcome the large computational cost needed to build surrogate-model-interpolating high-dimensional gradient vectors. Many papers present new optimizers through a description followed by applications to some aerodynamic optimization cases. This effectively shows that the new optimizer works well within a specific optimization framework (specific shape parameterization and specific analysis code), but this does not give insights into the efficiency of the optimizer within another optimization framework. It is then impossible to determine if the new optimizer will be more efficient than your favorite optimizer. Indeed, no reliable validation test cases exist in the field of aerodynamic shape optimization. Recently, a move toward an aerodynamic optimization workshop was initiated by Epstein et al. [13] by comparing three different optimization frameworks on a wing-design problem (same initial geometry but different shape parameterizations and analysis codes). Because it is not yet available, the performance of the optimizer is assessed by comparison with those of a well-known optimizer, namely the design optimization tools/Broyden–Fletcher–Goldfarb–Shanno (DOT–BFGS) [14]. All optimizers were implemented within the same aerodynamic optimization framework (same analysis code and shape parameterization); therefore, it was possible to accurately benchmark their relative performance. Additionally, this paper proposes some original criteria to analyze and describe the performance of optimizers. Currently, optimization tools are being commonly used within design departments. These tools generally propose a wide variety of optimization algorithms, and designers need measures in order to clearly identify advantages and drawbacks of each optimizer and to be able to choose the optimizer best suited to their problems.

This paper is organized as follows. Section II describes the criteria used to assess the performance of the optimizers. It also presents the high-fidelity optimization suite OPTALIA (optimisation en aérodynamique) and the gradient-based optimizer DOT–BFGS. The latter algorithm uses a classical quasi-Newton method and is taken as the reference optimizer. Numerous applications were successfully conducted using this optimization framework [4,11], but to tackle the limitations underlined previously, two RS-based optimizers have been developed using a kriging or cokriging method, which are presented in Sec. III. They rely on a multicriteria sampling refinement process and are able to run multiple CFD simulations in parallel in order to reduce the total-clock time of the process. Finally, Sec. IV compares the RS-based optimizers to the gradient-based reference optimizer on a low-dimensional drag-reduction problem considering six design variables on a RAE2822 airfoil and a high-dimensional test problem considering 45 design variables on a wing. These problems were chosen to assess the effect of dimensionality on the efficiency of optimizers.

II. Optimization Suite

The in-house software OPTALIA, developed at Airbus, is used to perform aerodynamic shape optimization. This high-fidelity

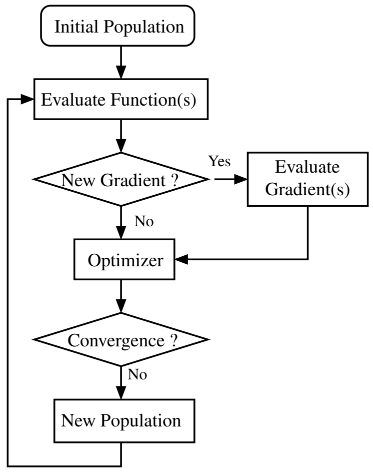


Fig. 1 Description of the optimization process implemented to accommodate various types of optimizers.

optimization suite can improve aerodynamic performance of an aircraft by changes in the shape (planform variables fixed) and is adapted to the work done during the detailed design phase.

A. Common Optimization Framework

A general optimization framework, represented in Fig. 1, has been set up in OPTALIA in order to implement various type of optimizers (gradient, genetic, and RS). From a global point of view, the process can be described as a succession of two main tasks: evaluation and optimization. The evaluator is in charge of computing the function value and, if needed, the gradient value of all elements of the population. The population size n_{pop} depends on the optimization algorithm chosen, but if it is greater than unity, the evaluator performs the simulations of all shapes simultaneously by running multiple jobs on high-performance computers. A large population can reduce the wall-clock time of the optimization process at the risk of saturating computational resources. Once all shapes have been evaluated, the optimizer uses the new information on the functions to propose a new population of shapes, and the next iteration begins. In addition to the internal stopping criteria of the optimizer, the convergence is forced at the process level when the number of iterations or the number of function evaluations exceeds a given threshold, max $n_{\mathrm{iter}} = 100$ and max $n_{\mathrm{eval}} = 200$.

One of the challenges in aerodynamic shape optimization is to efficiently manage running the evaluator and the optimizer automatically in batch mode. More particularly, the evaluator itself is a complex process requiring large computational resources. It is described next.

B. Evaluator for Computational Fluid Dynamics Functions

1. *Shape Parameterization and Mesh Deformation*

The shape parameterization consists of applying the Hicks–Henne sinusoidal bump functions on the surface skin of a block-structured mesh. Each bump function is defined by three variables driving the amplitude, the position, and the width expansion. The direction of the deformation can either be along the vector locally normal to the surface or along a fixed vector (vertical axis). This type of deformation was initially developed by Hicks and Henne [1] for numerical optimization of airfoils. The original Hicks–Henne bump formulation gives deformation along a curve. When applied on a two-dimensional surface, a linear propagation of the bump is done in the second direction, using fixed propagation distances.

Once computed, the field of deformation on the surface skin is spread in the volume mesh, using a mixed integral/transfinite interpolation method. The integral method is used to compute the deformation of nodes defining the boundaries between blocks, and then the transfinite interpolation computes deformation inside each block in parallel.

An analytical linearization of the shape parameterization and of the mesh-deformation modules enables the inexpensive computation (in terms of CPU time) of the sensitivity of the surface mesh and the sensitivity of the volume mesh with respect to design variables.

A description of the mesh-deformation technique and a validation case can be found in Meaux et al. [4].

2. *Flow Simulation, Mean-Flow, and Adjoint Codes*

Flow analyses were performed with the ELSA (ensemble logiciel pour la simulation en aérodynamique) [15] software developed by ONERA and the Centre Européen de Recherche et de Formation Avancée en Calcul Scientifique. The flow is simulated by solving the RANS equations associated with the one-equation Spalart–Allmaras turbulence model on block-structured meshes, using a cell-centered finite-volume approach. The second-order Roe's upwind scheme with the van Albada limiter is used as spatial scheme coupled with an implicit time resolution. Diffusive terms are discretized with a second-order centered scheme. Multigrid and local time-stepping techniques are used to increase the converge rate.

An overview of various results obtained with ELSA was described by Cambier and Veuillot [15]. Industrial users, such as Airbus, participate in the validation of the software. The results obtained by Airbus engineers from the second AIAA-drag-prediction-workshop test case were described by Brodersen et al. [16].

One of the main requirements of designers is to obtain the same results when using the CFD solver, inside or outside the automatic optimization tool. As hysteresis phenomena are common when dealing with transonic flows, the same initial flow condition (uniform flow) was used for all the simulations performed during the optimization. Therefore, the computational cost of CFD simulations cannot be reduced by using a restart strategy, using the flow solution corresponding to the previous shape. The computational cost of the optimization grows linearly with the number of function evaluations.

The sensitivity of the objective function with respect to the design variables is computed using the discrete adjoint method [17]. For an explicit presentation of the adjoint system solved within ELSA and an evaluation of the accuracy of the sensitivities, the reader is referred to Peter et al. [18] and Meaux et al. [4]. This method enables the computation of the sensitivity of a single function with respect to n_{dv} design variables at the cost of one linear system resolution (same size as the RANS system). The gradient vector is computed using approximately the same computational time (factor ≈ 1.5) as one mean-flow simulation. For typical aerodynamic problems considering hundreds of design variables and a few functions (lift and drag), this is a considerable improvement over the classical method of forward finite differences requiring as many flow solutions as design variables.

3. *Aerodynamic Function Computation*

The objective function chosen is the far-field pressure drag,

$$F = Cd_p = Cd_{\mathrm{vp}} + Cd_w + Cd_i \tag{1}$$

where Cd_{vp}, Cd_w, and Cd_i denote the viscous pressure drag, the wave drag, and the induced drag, respectively. The friction drag Cd_f is excluded from the objective function, as it does not significantly change with the amplitude of deformation considered. The wetted surface is almost unaffected by the shape deformation. The postprocessing code used is FFD41 [19], developed by ONERA. It implements a far-field drag breakdown method. The two main advantages of this approach are its ability to decompose pressure drag into physical components (wave drag, induced drag, and viscous pressure drag) and its accuracy through a filtering of nonphysical drag (spurious drag). The accuracy of this software was assessed during the Second AIAA Drag Prediction Workshop [16].

The postprocessing module can also compute the sensitivities of the drag with respect to the flow variables and with respect to the mesh, with an analytical formulation.

C. Gradient-Based Optimizer, Reference Optimizer Design-Optimization-Tools–Broyden–Fletcher–Goldfarb–Shanno

The reference optimizer is gradient based and uses the classical quasi-Newton BFGS method from the DOT [14] library. This algorithm is similar to the very well-known algorithm CONMIN. The description of one internal step of this optimizer is given in Fig. 2. First, the algorithm determines a descent direction d_k, using the evolution of the gradient vector during the last two internal iterations. Once the direction is computed, a line search aimed at computing the norm of displacement giving the best improvement is performed. The line search is driven by a monodimensional polynomial interpolation and requires successive function evaluations. This type of optimizer is intrinsically sequential, as it follows a single descent path. The optimizer stops when no improvement is achieved during two optimizer internal iterations, or if the gradient norm is null (falls under a given threshold). This type of algorithm is quick to converge, but it can only find a local optimum depending on the starting location. It is possible to use a multistart strategy in order to try to avoid some local optima, but the computational cost is increased because of the multiple optimizations to be performed. As the main advantage of this algorithm was its speed, the multistart strategy was not used. For the optimization framework, only one vector of design variables is

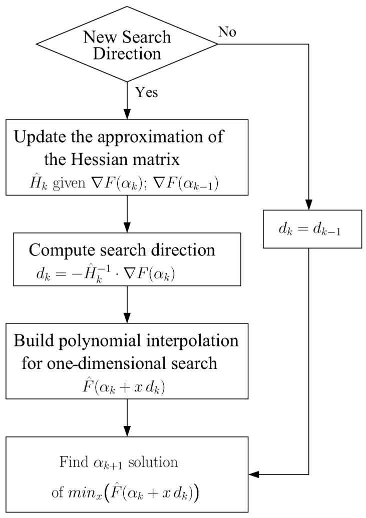

New Search Direction
No
Yes
Update the approximation of the Hessian matrix $\hat{H}_k$ given $\nabla F(\alpha_k)$; $\nabla F(\alpha_{k-1})$
$d_k = d_{k-1}$
Compute search direction $d_k = -\hat{H}_k^{-1} \cdot \nabla F(\alpha_k)$
Build polynomial interpolation for one-dimensional search $\hat{F}(\alpha_k + x\, d_k)$
Find α_{k+1} solution of $min_x\big(\hat{F}(\alpha_k + x\, d_k)\big)$

Fig. 2 Description of the gradient-based optimizer module DOT–BFGS.

handled by process iteration ($n_{\text{pop}} = 1$), whereas an internal iteration of the optimizer comprises multiple function evaluations (a search direction update and a complete line search).

In terms of the quantity of information, the quasi-Newton gradient algorithm proposes the next set of variables by using only information about the current internal iteration. The internal iteration contains information about the search direction plus some function evaluations. The number of function evaluations performed during the line search is not constant. It generally does not exceed 10, and it is assumed to be equal to 10 in this paper. The line-search step involves the computation of $N = 10$ scalar information concerning the unknown function. The computation of the search direction requires two gradient vectors giving $N = 2n_{\text{dv}}$ scalar information on the function to the optimizer. This optimization algorithm proposes a new shape based on $N = 2n_{\text{dv}} + 10$ scalar information on the unknown function. Even if the approximated Hessian matrix $\hat{H}_k$ is more and more accurate as the number of internal iterations increases, the algorithm does not retain all the information known about the function, but it focuses on the information in the vicinity of the current shape.

D. Performance Measurement of Optimization Algorithms

The properties of the optimization algorithms are commonly described by two opposite notions: exploration and exploitation. Exploration represents global search for promising solutions within the entire domain of variables, whereas exploitation represents local search of promising solutions given some information on the function. Optimizers that perform exploration to the exclusion of exploitation (random search) are likely to discover a variety of interesting solutions at the cost of many trials without gaining the full potential represented by each solution. Conversely, optimizers that perform exploitation to the exclusion of exploration are likely to find themselves trapped by the nearest local optimum.

The figure of merit measuring the wall-clock time needed for an optimization to converge is the number of process iterations n_{iter}. The figure of merit, measuring the computational cost of an optimization, is the total number of function and gradient evaluations, $n_{\text{eval}} + n_{\text{grad}}$. Both of these quantities qualify the exploitation properties of the optimizer. One has to notice that a distinction is made between iteration at the process level (Fig. 1) and iteration at the optimizer level (Fig. 2). The number of internal iterations of the optimizer does not directly intervene to assess performance and is not reported (for DOT–BFGS, it corresponds to the number of gradient calculations).

Values denoted by the superscript of ref are reference values obtained by comparing the value of the objective function at a given process iteration to all previous evaluations, so that the quantity $F_c - F_c^{\text{ref}}$ is always positive or null. This reference value corresponds to the current best function value known by the optimizer: the current optimum. The reference value at the last iteration of the gradient algorithm corresponds to the last computation, because it converges by following a descent path. That is not necessarily true for RS-based optimizers. The evolution of the reference value with respect to the number of iterations represents the convergence history.

Exploration denotes the ability of the optimizer to extensively poll the domain of variables. It is measured in terms of the variation of the design variables (x_c) with respect to the current best known variables (x_c^{ref}) during the optimization, $\sum ||\bar{x}_c - \bar{x}_c^{ref}||/n_{dv}$. To properly measure this quantity, the design variables are scaled between zero and one, and the distance between vectors of the design variables is divided by the number of design variables. Alternatively, the exploration can also be measured in terms of the variation of the function value during the optimization, $\sum(F_c - F_c^{\text{ref}})$, but as the functions studied are nonlinear, it is better to measure the distances in the space of scaled variables. The reference point (x_c^{ref}) chosen to measure exploration is important. It does not seem logical to choose the baseline shape (null deformation) as the reference point, because lots of algorithms do not require a starting point. An optimum found by one of the optimizers used could have been chosen a posteriori, but it would have introduced a bias, as it is not ensured that all optimizers would poll this location. By choosing the current best variables known by the optimizer as a reference point, the exploration criterion effectively measures the ability of the optimizer to ignore its current best value for the sake of other promising locations.

The performance of an optimizer is expected to vary with respect to the complexity of the optimization problem considered. This complexity cannot be precisely assessed, as it depends on numerous parameters: the number of variables, the range of the variables, the number of functions, and the behavior of the functions on the domain. Despite that, it is generally admitted that the complexity of an optimization problem is the product of the number of variables by the number of functions (objectives and constraints).

At each iteration, the optimizer reads and exploits a database containing information on the function in order to find a promising location. This database contains function and, sometimes, gradient values at the locations already explored during the previous iterations. Some optimizers take into account the whole database (for example, RS-based methods), and others use only a part of the database (for example, DOT–BFGS uses only the two last locations). However, the total quantity of information N contained in this database will be used as a figure of merit to compare optimizers. A function value will be counted as one scalar value, and a gradient value will be counted as n_{dv} scalar values, so that $N = n_{\text{eval}} + n_{\text{grad}} n_{\text{dv}}$.

As a summary, the criteria retained to characterize optimizers are as follows:

1) The main criterion is the relative improvement of the function ($I(f) = [(F^{\text{ini}} - F^{\text{ref}})/F^{\text{ini}}] \cdot 100$).

2) The cost criteria are the computational time and the cost of the optimization represented by the total number of process iterations n_{iter}, function evaluations n_{eval}, gradient evaluations n_{grad}, and the total quantity of scalar information required by the optimizer to reach the optimum N.

3. The exploitation criteria, representing the ability of the optimizer to translate information on the function into an improvement, are $[I(f)]/N$ or $[I(f)]/n_{\text{eval}}$.

4. The exploration criteria, representing the ability of the optimizer to investigate locations far from the current best known location, are the total exploration $\sum ||\bar{x}_c - \bar{x}_c^{ref}||/n_{dv}$, the exploration per iteration $\sum ||\bar{x}_c - \bar{x}_c^{ref}||/n_{dv}/n_{iter}$, and the exploration per quantity of information $\sum ||\bar{x}_c - \bar{x}_c^{ref}||/n_{dv}/N$.

The choice of optimizer comes from a compromise between four competing criteria: improvement, computational cost (or time), exploitation, and exploration.

III. Response-Surface-Based Optimizer: Response-Surface Kriging and Response-Surface Cokriging

Surrogate modeling tools are now widely used in engineering. Basically, these tools enable the inexpensive approximation of a function on a continuous domain $\mathcal{D}$, given a database of n_s samples, $\mathcal{S}$. Once built, the surrogate model can predict numerous information on the true function through graphical plots giving trends, sensitivity of the function with respect to each variable, or the value and location of the minimum when coupled with an optimizer.

A. Response-Surface-Based Optimization

Although it is computationally affordable to build a globally reliable RS when considering three or four design variables, a phenomenon described by Bellman [20] as the curse of dimensionality prevents the use of global RSs on high-dimensional spaces [21]. In fact, this phenomenon makes the global optimum almost impossible to reach on high-dimensional problems, because the number of evaluations required increases exponentially with the number of variables. It means that the design predicted by the RS has to be confirmed with calls to the true function. The RS is only trusted locally at sample locations (local RSs). These additional calls to the true function enable the validation of the prediction made by the surrogate and also enhance its accuracy.

A previous study [22] showed that when increasing dimension from one to six, the complexity of aerodynamic functions cannot be represented with as few as 200 function evaluations, and it recommends the use of local RSs. Local RS-based optimizers are initialized with a space-filling sampling, giving rough trends of the function and internally using an expensive global optimizer to optimize one or multiple sampling refinement criteria, $C_r(x)$, in order to iteratively increase the surface accuracy around locations of possible optima. At each iteration, multiple locations can be explored. Once the evaluator has computed the function values at these locations, the RS is updated by adding these new samples.

This type of optimizer relies on the assumption that an efficient surrogate model can be built in high-dimensional search spaces (several tens of variables).

B. Building Response Surfaces: From Kriging to Samples-Limited Indirect Cokriging

Kriging (design and analysis of computer experiments formulation) was chosen to build the RS $\hat{F}(x)$ for its ability to accurately approximate nonlinear functions and guarantee a null error at samples. This method is very well known in the field of numerical optimization and CFD. The reader is referred to papers by Sacks et al. [23] and Jones et al. [6] for a detailed description of the method. The next section presents only its main features.

1. *Kriging*

The kriging method is formed by a constant term $\hat{\beta}$, representing a mean of the function at samples F_s, plus a linear combination of the basis function interpolating each sample built up as a stochastic process $Z(x)$:

$$\hat{F}(x) = \hat{\beta} + Z(x) \approx F(x) \tag{2}$$

The basic assumption underlying kriging is that the covariance of the function is linked to the spatial correlation, and this correlation is at the maximum when the distance between the points is null and decreases with the distance:

$$\mathrm{cov}\,[Z(s^i),\ Z(s^j)] = \hat{\sigma}^2 R_{ij} \tag{3}$$

The correlation matrix R (order $N = n_s$) results from evaluations of a spatial correlation function linking the distance to the correlation between two points:

$$R_{ij} = \prod_v \mathrm{scf}_v(|s^i_v - s^j_v|) \tag{4}$$

The correlation function tends toward one when distance decreases toward zero. A cubic spline correlation function, depending on one parameter driving the directional strength of the correlation θ_v, was chosen:

$$\mathrm{scf}_v(x) = \begin{cases} 1 - 6(x\theta_v)^2 + 6(x\theta_v)^3 & x < \frac{1}{2\theta_v} \\ 2(1 - x\theta_v)^3 & \frac{1}{2\theta_v} \le x < \frac{1}{\theta_v} \\ 0 & x \ge \frac{1}{\theta_v} \end{cases} \tag{5}$$

If the distance between two samples is too small, their corresponding columns in the correlation matrix will be almost the same, implying an ill-conditioned matrix and an unstable kriging model. That is why space-filling sampling methods, like Latin hypercube sampling are recommended when using kriging.

To evaluate kriging at an unknown location x, a vector of correlation r between sample points and this unknown point is computed:

$$r_i(x) = \prod_v \mathrm{scf}_v(|x_v - s^i_v|) \tag{6}$$

Once the correlation matrix has been inverted, the prediction $\hat{F}$ can finally be computed at the point x:

$$\hat{F}(x) = \hat{\beta} + r^t(x)R^{-1}(F_s - 1\hat{\beta}) \tag{7}$$

One should notice that the kriging method does not require a minimum number of samples to be built, whereas a linear polynomial interpolator would require at least $n_{\mathrm{dv}} + 1$ samples. In addition, the predicted uncertainty of the kriging function, or standard error $\hat{S}(x)$, can be computed:

$$\hat{S}(x) = \hat{\sigma}\left[1 - r^t(x)R^{-1}r(x) + \frac{[1 - 1R^{-1}r(x)]^2}{1^tR^{-1}1}\right]^{1/2} \ge 0 \tag{8}$$

The standard error of kriging is null at samples and increases with distance.

2. *Kriging Fit*

Kriging models are fitted by maximizing the logarithm of their likelihood estimates [maximum likelihood estimate (MLE) problem]. For the parameters β and σ^2, analytical expressions maximizing this value are known:

$$\hat{\beta} = (1^tR^{-1}1)^{-1}1^tR^{-1}F_s \tag{9}$$

$$\hat{\sigma}^2 = \frac{1}{n_s}(F_s - 1\hat{\beta})^tR^{-1}(F_s - 1\hat{\beta}) \tag{10}$$

The correlation parameters θ_v are the solution of the following MLE optimization problem:

$$\mathrm{MLE} = \max_\theta\left(-\frac{1}{2}\{n_s[\ln(\hat{\sigma}^2) + \ln(2\pi) + 1] + \ln|R|\}\right) \tag{11}$$

It is solved using a gradient-based optimization algorithm initialized by an appropriate guess, as described by Laurenceau and Sagaut [22]. Each likelihood evaluation requires the computation of the determinant of the correlation matrix, and the resolution of the optimization problems requires several hundred likelihood evaluations.

During the sampling refinement process, the parameters are refitted at each update of the sample database.

3. *Sample-Limited Indirect Cokriging and Gradient-Enhanced Kriging*

Gradient-enhanced RSs interpolate the function and the gradient at each sample location. Such RSs built using the kriging method are commonly called the cokriging model. The benefits of gradient-enhanced surrogates were described by Chung and Alonso [24], Liu [12], and Sobester et al. [25]. As cokriging models include more information on the true function than kriging models, they need fewer samples to achieve a given level of accuracy. Moreover, a comparison varying the dimension of the problem and the number of samples on an aerodynamic test case [22] has proven that the vectorial information provided by the gradient is more beneficial for high-dimensional problems.

For its flexibility, the formulation retained to build the gradient-enhanced kriging is the indirect cokriging formulation [12]. It does not change the kriging formulation described previously, because the gradient interpolation is performed through a sample database augmentation scheme. Instead of directly using the gradient information, it is used to add one point per direction at each sample, using a first-order Taylor approximation:

$$F(s^{n_s+iv}) = F(s^i) + \frac{\partial F(s^i)}{\partial x_v} 10^{-4} \mathrm{range}_v(\mathcal{S}) \qquad (12)$$

After using the gradient information at each sample, the augmented database contains $N = n_s(n_{\mathrm{dv}} + 1)$ samples.

The computational cost of the RS (fit and evaluation) is considered negligible when compared with CFD function evaluations for a matrix order of up to $N \approx 400$. However, as the computational cost of cokriging depends on the number of variables, it becomes impractical for high-dimensional problems. It is then necessary to use altered augmentation schemes.

Liu [12] proposed a direction-limited augmentation scheme using only one augmentation point per sample, $N = 2n_s$. Practically, this strategy implies a loss of information, because only one element of the gradient vector is used. In the context of the OPTALIA optimization software, gradients of CFD functions are computed with an adjoint method, and the complete gradient vector is computed, even if only one term of the vector is included in the database.

For this reason, a sample-limited augmentation scheme is preferred. It consists of using the complete gradient information for only a limited subset of samples: the first n_{aug} in the database. The augmented matrix order is then $N = n_s + n_{\mathrm{aug}} n_{\mathrm{dv}}$. Practically, the parameter n_{aug} is used to adapt the cokriging computational cost to the number of variables. For the problem considering six design variables ($n_{\mathrm{dv}} = 6$), ten augmented samples were used ($n_{\mathrm{aug}} = 10$), whereas for the 45-design-variable test case ($n_{\mathrm{dv}} = 45$), five augmented samples were used ($n_{\mathrm{aug}} = 5$).

C. Global Optimization on the Response Surface for Sampling Refinement

As stated in Sec. III.A, another optimization problem has to be solved inside the RS-based optimizer module. Some sampling refinement criteria $C_r(x)$ (presented in the next section) must be optimized to exploit the information contained in the RS and find a suitable new set of design variables. As the criteria are inexpensively evaluated, a global optimization strategy was adopted for their minimization. To effectively find the global minimum, even for high-dimensional problems (45 variables here), a strategy using multiple optimizers is used. This strategy proceeds in building a set $\mathcal{P}$ of possible optima of the criterion, given by the various methods, and finally retains only the optimum of this set of limited size.

First, the set of possible optima $\mathcal{P}$ is initialized by the minimum coming from the sample database.

Second, a binary-coded genetic algorithm¶ is used with a population size of 100 individuals and a maximum number of generations equal to 1000. A convergence test was implemented to stop this algorithm when no improvement on the function is made during 10 consecutive iterations. This algorithm is run 100 times, using different seed values. Each run enables the insertion of a new point in the set of possible optima $\mathcal{P}$. At this stage, the set $\mathcal{P}$ contains 101 elements.

Third, a pseudorandom exploration is performed by evaluating the function at 10^5 locations. The minimum of the exploration phase is used to initialize a gradient descent (quasi-Newton BFGS method). Once converged, the minimum of the gradient algorithm is inserted in the set $\mathcal{P}$. This step is repeated 10 times, with different seed values for the exploration. At this stage, the set $\mathcal{P}$ contains 111 elements.

¶Data available online at <http://www.cuaerospace.com/carroll/ga.html> [retrieved 01 March 2007].

Finally, a last gradient descent is performed, departing from the minimum point of the set of possible optima $\mathcal{P}$.

This strategy may not be the most efficient in terms of the number of evaluations, but the authors are confident that it enables the discovery of the global optimum on the RS with as much accuracy on the six-design-variables test case than on the 45-design-variables test case. As no bias can be attributed to this internal optimization step, any loss of performance of the RS-based optimizer when increasing dimension will be attributed to a loss of accuracy of the RS.

D. Sampling Refinement Strategy

In this paper's context, the sampling database is very sparse, due to the relatively high number of variables (45 variables) and the limited number of samples (less than 200 samples). The RSs are thus globally inaccurate, and the success of the optimization strategy lies on the sampling refinement method. This updating method must represent a compromise between exploitation and exploration. It must perform some exploration in order to avoid being trapped by one of the multiple local optima and some exploitation in order to decrease the computational cost as much as possible. Moreover, it must prevent a premature convergence in case of ill-fitted models. These ill-fitted models correspond to models built with incorrect correlation parameters. When dealing with very sparse sampling databases, the MLE optimization sometimes gives overestimated correlation parameters corresponding to completely uncorrelated samples, which is unrealistic.

The adaptive sampling process can be designed to produce one sample or multiple samples by iteration. As stated in the surrogate management framework [26], the optimizer has to efficiently exploit, at each iteration, the current RS (SEARCH step). It is then recommended that the RS be globally explored for multiple prospective pools instead of restricting the search to the minimum of the RS. Consequently, it was decided to implement a process handling multiple refinement locations per iteration. One of the advantages of this type of process is that the calls to the true function (the expensive CFD computation) can be parallelized and the total-clock time decreased. The classical approach to obtain multiple samples by iteration is to use a single sampling criteria and take multiple local optima or multiple individuals of the last generation of a genetic algorithm as refinement locations [25]. In this paper, a different approach is developed. The authors propose the combination of multiple sampling refinement criteria within the same iteration in order to form an original and efficient sampling refinement process. To maintain the efficiency of the optimizer and limit the total computational cost, the number of combined criteria was limited to three.

1. *Combination of Three Sampling Criteria*

The most obvious criterion exploits only information on the function and search for the location of the minimum on the model [27]:

$$C_1(x) = \hat{F}(x) \qquad (13)$$

This criterion ensures that, in the case of a globally accurate RS, the optimum of the function will be sampled. That is why it was included within our combination of sampling criteria. This criterion is devoted to exploitation, but it is not exempt from exploration, because the model is inaccurate. This paper will not present a complete comparison of sampling criteria, but from the authors' experience, this criterion used on its own within an optimization process leads to a premature convergence and gives only a moderate improvement of the function. This fact has led us to couple this criterion with other criteria, exploiting the information given by the standard error [Eq. (8)] of the model in order to enhance the exploration capabilities of the algorithm.

It is possible to directly place the new sample at the location of the maximum predicted error. It would enable the achievement of a globally accurate model, but it would certainly waste a lot of samples to find the optimum of the function. This criterion is devoted to pure exploration and was not retained here. Several techniques exist to balance the exploitation of zones of low function values with the exploration of zones of high-error values (locations far from samples).

The expected improvement (EI) [6] is the most favored criterion when using kriging. This criterion represents a balancing between exploitation and exploration, which has proven to be efficient for, at least, low-dimensional problems [6,28–30] when the number of variables does not exceed 10. It maximizes the probability of improving the function over the current best known sample, given a model and its standard error. However, the EI criterion was not retained here, because its maximization is very difficult on high-dimensional problems. The EI function is highly multimodal and exhibits large flat areas with null gradients [6].

The lower confidence bounding (LCB) criterion of Cox and John [31] was preferred, as it tends to give a more regular function (a regularity similar to one of the original function) and is then easier to optimize. It directly uses a balancing coefficient b between function and error to perform a linear combination:

$$\mathrm{LCB}_b(x) = \hat{F}(x) - b\hat{S}(x); \qquad b \in \mathcal{R} \tag{14}$$

The minimum of the LCB criterion indicates the location of the sample. With the standard error of kriging always being positive, the error balancing coefficient b can be chosen positive or negative. An increase of the balancing coefficient enhances the exploration of the criterion, and vice versa. Cox and John [31] used the values $b = 2.0$ and $b = 2.5$ to solve several low-dimensional mathematical test problems, but they specified that their choice for b needed further investigation. This parameter depends on the optimization problem studied, and the user has to define an appropriate value. The authors made some attempts (not described here) to determine this coefficient dynamically, thanks to heuristic methods, but no significant gain on the efficiency of the optimization were achieved over the most obvious choice of $b = 1.0$. Therefore, the value $b = +1.0$ was retained:

$$C_2(x) = \mathrm{LCB}_1(x) = \hat{F}(x) - \hat{S}(x) \tag{15}$$

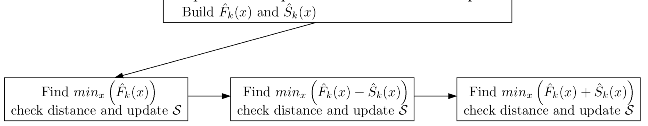
This $\mathrm{LCB}_1(x)$ criterion is dedicated to exploration, as it tends to explore locations distant from samples (for which the standard error is high). The low value for b enables moderate exploration that is coherent with the fact that the samples database is very sparse.

The first criterion can be related to a LCB criterion with a null balancing coefficient, $C_1(x) = \mathrm{LCB}_0(x)$. In this work, it was decided to use a LCB criterion with a negative coefficient of $b = -1.0$:

$$C_3(x) = \mathrm{LCB}_{-1}(x) = \hat{F}(x) + \hat{S}(x) \tag{16}$$

With a negative balancing coefficient, the criterion LCB_{-1} performs less exploration than the first criterion, directly minimizing the function (C_1). It is dedicated to exploitation. Most of the time, $C_1(x) = \mathrm{LCB}_0(x)$ and $C_3(x) = \mathrm{LCB}_{-1}(x)$ produce the same refinement location, so that only $C_1(x) = \mathrm{LCB}_0(x)$ is activated. Despite that, from the authors' experience, this criterion is crucial when dealing with very sparse sampling. Its role is to enhance the robustness of the process and to avoid a premature convergence (for example, in case of ill-fitted models).

The new kriging/cokriging-based optimizers (Fig. 3) are initialized using a Latin hypercube space-filling sampling method during the first iteration. The refinement process begins at the second iteration, after n_s^{ini} function evaluations. Once initialized, this framework simultaneously runs up to three CFD runs ($n_{\mathrm{pop}} \leq 3$) per iteration. Because of the fact that the sampling refinement process tends to explore locations of high-standard-error values (minimum of $C_2(x) = \mathrm{LCB}_1(x)$), it is able to explore undersampled regions. Because of its robustness, this optimizer is able to manage very coarse initial sample database (10 samples for 45 variables) at the cost of more iteration of the process. However, a large initial sampling database is beneficial in terms of total-clock time, because the first n_s^{ini} evaluations are performed simultaneously.


Update the Response Surface with new samples
Build $\hat{F}_k(x)$ and $\hat{S}_k(x)$
Find $min_x\left(\hat{F}_k(x)\right)$ check distance and update $\mathcal{S}$
Find $min_x\left(\hat{F}_k(x) - \hat{S}_k(x)\right)$ check distance and update $\mathcal{S}$
Find $min_x\left(\hat{F}_k(x) + \hat{S}_k(x)\right)$ check distance and update $\mathcal{S}$
Fig. 3 The sampling refinement process of the new RS-based optimizers.

For a wider point of view on surrogate-based optimization and sampling refinement techniques, the reader is referred to the review paper of Forrester and Keane [32] and the thesis of Sasena [28].

2. *Distance Check and Convergence*

Each location indicated by the minimization of a sampling refinement criterion must be validated by performing a distance computation with respect to already computed locations. The main purpose of this validation is to avoid performing expensive CFD computations on quasi-identical shapes. The secondary purpose is to ensure a good conditioning of the kriging correlation matrix. The point is rejected if its minimal scaled distance, with respect to already computed samples, falls below a fixed threshold:

$$x^{\mathrm{ref}} \tag{17}$$

accepted if

$$\min_i\left(\frac{1}{n_{\mathrm{dv}}}\sum_v |\hat{x}_v^{\mathrm{ref}} - \hat{s}_v^i|\right) > \epsilon$$

The value of the threshold ϵ must be adapted to the level of convergence expected from the optimizer (i.e., the number of iterations allowed). The value chosen here is $\epsilon = 10^{-6}$.

Convergence is reached when all three proposed refinement locations are rejected [too close from the already sampled locations in Eq. (17)] or if the function value was not improved during the last 20 iterations.

Figure 3 describes the sampling refinement process implemented within our new RS-based optimizers.

IV. Results

The no-free-lunch theorem for optimization by Wolpert and Macready [33] states that the ultimate optimization algorithm does not exist. If optimizer A performs better than optimizer B for a given set of problems, then it must perform worse on the remaining problems. It is theoretically possible to find another set of problems for which the optimizer will not be able to contradict this fact. Their recommendations are then to use a wide variety of test problems in order to compare general-purpose optimization algorithms and to incorporate problem-specific knowledge into the optimizer as much as possible. As the aim of the present paper is to determine optimizers suited to aerodynamic design problems, the comparison of optimizers was not made by using a large-spectrum testing environment (such as the constrained and unconstrained testing environment), as would have been done for a general-purpose optimizer. In the following section, a comparison between two new kriging- and cokriging-based optimizers and the reference gradient optimizer (DOT–BFGS) is made by studying two aerodynamic test cases. The test cases are directly performed within OPTALIA on functions evaluated through CFD computations (including numerical noise). Moreover, the problems are defined as representative simplifications (no constraint functions, less design variables, and coarser meshes) of real industrial problems handled by aircraft designers.

A. Airfoil Drag Minimization, Considering Six Design Variables and One Function

The first test problem concerns a RAE2822 airfoil at a Mach number of $M = 0.729$ and an angle of attack (AOA) of 2.31 deg. The chord length is 1 m, and the Reynolds number value is $Re = 7.10^6$.

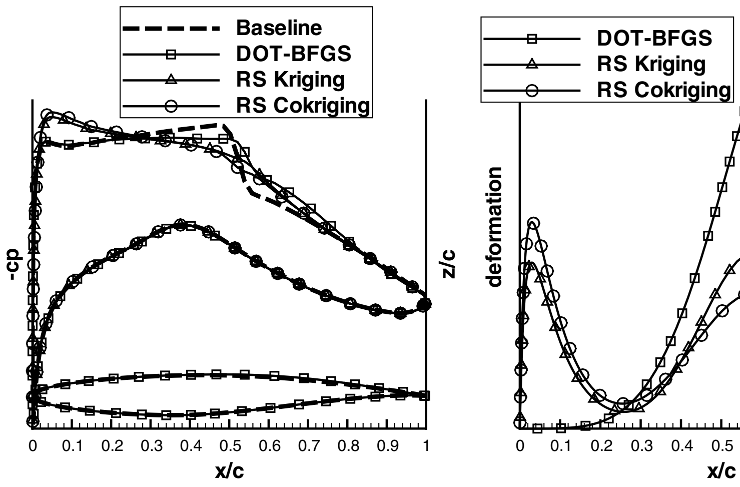


Fig. 4 Optimization of the RAE2822 airfoil: pressure coefficient distributions (left) and optimal deformations of the upper surface (right) given by DOT–BFGS, RS–kriging, and RS–cokriging optimizers.

The C mesh is formed of 73×458 nodes with its boundary-layer refinement. The wall-clock time for one steady flow simulation is 5 min., using two Advanced Micro Devices, Inc. (AMD) Opteron 275 (2.2 Ghz) processors to perform 400 iterations.

The objective function considered is the far-field pressure drag. Two bumps ($n_{\mathrm{dv}} = 6$) are applied to deform the upper surface in the direction of the vertical axis. Only positive deformations are allowed, and the maximum amplitude of one bump is 5 mm ($A^{\max}/c = 0.5\%$). The domain of variation for the positions of the bumps allows bump overlapping. The complexity of the optimization problem is low, as only one function and six design variables are considered. Moreover, the extent of the domain of design variables can be considered relatively small, as it is restricted to combinations of two positive bumps.

The same initial samples distribution is used for kriging and cokriging RSs. It contains 10 samples, $n_s^{\mathrm{ini}} = 10$ and $N^{\mathrm{ini}} = 10$ (including the baseline shape), and nine space-filling samples distributed by a Sobol method. The initial sampling density is low, and its value is $10^{1/6} \approx 1.5$ points per direction. The size of the initial database was minimized in order to enable the exploration of the design space through the iterative sampling refinement process. The sample database augmentation for the sample-limited cokriging includes the information of 10 gradient vectors, $n_{\mathrm{aug}} = 10$, providing a quantity of information equivalent to 60 additional samples, $N^{\mathrm{ini}}_{\mathrm{RS\ cokriging}} = 70$ ($70^{1/6} \approx 2.1$). The computational cost of the gradient-augmented database is higher (10 adjoint computations). This additional cost is not very significant, because the 10 initial samples are evaluated in parallel during the first iteration. However, it is taken into account in the following results.

The starting point for the gradient algorithm corresponds to large equidistributed bumps of null amplitudes.

On Fig. 4 the baseline and the optimized shapes and the associated pressure coefficient values are represented. The pressure distribution on the baseline shape shows the presence of a shock (drag production) at the upper surface of the airfoil at 55% of the chord length. This shock is attenuated (suppressed) by the deformations proposed by the different optimizers. The noteworthy differences between each pressure distribution show that all optimizers converged to different solutions. Both RS optimizers found an optimal shape by distributing the two bumps near the leading edge and the shock location, whereas the gradient optimizer was trapped by the high sensitivity of the drag to deformations near the shock location and tried to superpose both bumps near the shock. This illustrates the fact that functions studied in numerical aerodynamic shape optimization exhibit multimodal behavior and strong trends (high-gradient values). The low complexity of this problem does not prevent the gradient optimizer from being trapped in a local optimum.

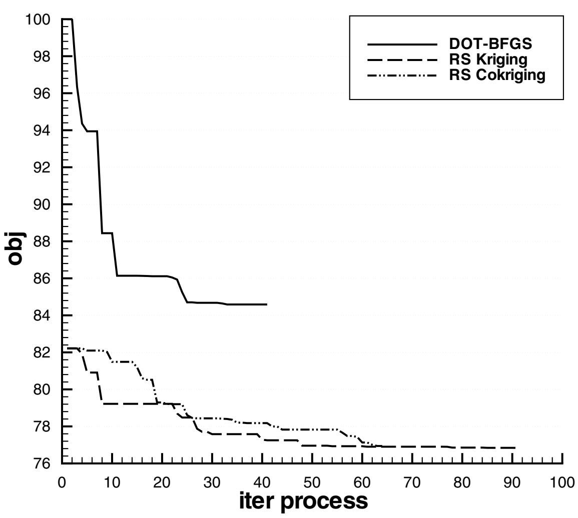
In terms of improvement of the objective function (Table 1), the RS optimizers largely outperform the gradient optimizer. Both RSs algorithms give very close results in terms of final shape deformation

Table 1 Optimization of the RAE2822 airfoil: aerodynamic performance of the baseline and optimized shapes

	Baseline	DOT–BFGS	RS–kriging	RS–cokriging
$Cd_p = F$	100	84.6 (−15.4%)	76.7 (−23.3%)	76.9 (−23.1%)
Cd_w	100	16.5	0.0	3.0
Cd_{vp}	100	100.7	95.3	94.5
Cl/Cd	100	110	114	114
AOA for $Cl = Cl_0$	2.31	2.19	2.24	2.243
Cd_p for $Cl = Cl_0$	100	79.7 (−20.2%)	79.0 (−21.0%)	76.9 (−23.1%)
Cl/Cd for $Cl = Cl_0$	100	112	113	114

Table 2 Optimization of the RAE2822 airfoil: performance of the optimizers DOT–BFGS, RS–kriging, and RS–cokriging (the last and next-to-last set of lines show, respectively, the performance in terms of exploration and exploitation)

	DOT–BFGS	RS–kriging	RS–cokriging
Improvement $I(f)$	15.4%	23.3%	23.1%
n_{iter}; $n_{\mathrm{eval}} + n_{\mathrm{grad}}$; n_s^{ini}; N	41; 41 + 6; 0; 77	90; 172 + 0; 10; 172	63; 131 + 10; 10; 191
Stopping criterion	No improvement	No new point	No new point
$I(f)$ at $n_{\mathrm{eval}} = 41$; $n_{\mathrm{iter}} = 41$	15.4%; 15.4%	20.8%; 22.7%	18.5%; 21.8%
$I(f)/n_{\mathrm{eval}}$	$37.6 \cdot 10^{-2}$	$13.5 \cdot 10^{-2}$	$17.6 \cdot 10^{-2}$
$I(f)/N$	$20.0 \cdot 10^{-2}$	$13.5 \cdot 10^{-2}$	$12.1 \cdot 10^{-2}$
$\sum \lvert F_c - F_c^{\mathrm{ref}} \rvert$	56.2	1503	953
$(1/n_{\mathrm{dv}}) \sum \lVert \bar{x}_c - \bar{x}_c^{\mathrm{ref}} \rVert$	0.200	15.6	10.0
$(1/n_{\mathrm{dv}}) \sum \lVert \bar{x}_c - \bar{x}_c^{\mathrm{ref}} \rVert / n_{\mathrm{eval}}$	$0.48 \cdot 10^{-2}$	$9.07 \cdot 10^{-2}$	$7.63 \cdot 10^{-2}$
$(1/n_{\mathrm{dv}}) \sum \lVert \bar{x}_c - \bar{x}_c^{\mathrm{ref}} \rVert / N$	$2.60 \cdot 10^{-3}$	$90.7 \cdot 10^{-3}$	$52.3 \cdot 10^{-3}$



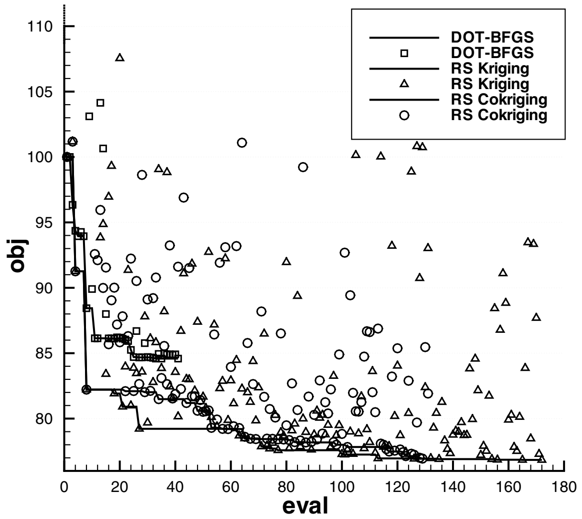


Fig. 5 Optimization of the RAE2822 airfoil: evolution of the objective function F during the optimizations. (The lines represent the best function value explored by the optimizer F^{ref} at n_{iter} for the left figure and n_{eval} for the right figure. On the right, the symbols represent the function values of all shapes explored during the optimizations at evaluation n_{eval}.)

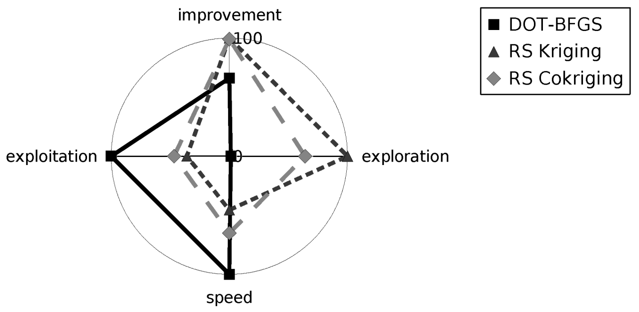


Fig. 6 Optimization of the RAE2822 airfoil: polar diagram summarizing the performance of the optimizers. [Improvement corresponds to $I(f)$, exploration corresponds to $(1/n_{\mathrm{dv}})\sum\|\bar{x}_c - \bar{x}_c^{\mathrm{ref}}\|$, speed corresponds to $1/n_{\mathrm{iter}}$, and exploitation corresponds to $I(f)/n_{\mathrm{eval}}$.]

and function value. This fact seems to indicate that both methods tend to the same solution, which is probably the global optimum of the problem. It confirms their enhanced exploration capabilities. One can see from Table 1 that, contrary to the RS optimizers, DOT–BFGS does not manage to completely suppress the wave drag Cd_p and slightly increases the viscous pressure drag Cd_{vp}.

Despite the fact that an improvement is achieved within the initial sample database of the RSs, it appears that a multistart strategy involving DOT–BFGS departing from each one of these 10 samples does not manage to outperform either of the RS optimizers. The multistart gradient strategy departing from 10 space filling (including the baseline shape) achieved an improvement of $I(f) = 21.3\%$. This is 2% worse than the RS optimizers, but it is 6% better than DOT–BFGS departing from the baseline shape. The computational cost involved largely exceeded our initial bounds (414 function evaluations and 73 gradient evaluations), which is why this approach was not extensively studied. Moreover, it appeared that all 11 optimizations run on this problem with DOT–BFGS were trapped by different local optima, whereas the problem considered only six variables.

Even if the purpose of this paper is not to determine a realistic design but to compare the optimizers, a short assessment of the aerodynamic performance of optimized shapes was made. It appears that the increased thickness (positive amplitude of the bumps) of the optimized airfoils induces an increase of the lift coefficient. The aerodynamicists want to decrease drag for a given lift. To confirm the interest of the optimized shapes, the AOA of the optimized airfoils was recalibrated to achieve the same lift as on the baseline shape ($Cl = Cl_0$ on Table 1). In these conditions, the ranking between the different shapes remains unchanged.

Figure 5 represents the convergence history of each optimizer. On the right-hand side, the symbols represent the points being currently evaluated (F_c), and the lines represent the current best-known value of the optimizer (F_c^{ref}). The exploration in terms of function value, $\sum |F_c - F_c^{\mathrm{ref}}|$, corresponds to the sum of the differences between symbols and lines, so that a distribution of symbols widely spread indicates more exploration. On the left-hand side is represented the evolution of F_c^{ref} with respect to the iteration number for each optimizer. It appears that even the purely space-filling samples (computed during the first iteration of RS optimizers) managed to improve the baseline shape and outperform the optimum found by DOT–BFGS. Despite that, the worst algorithm in terms of final function value, DOT–BFGS, is also the quickest to converge (lowest n_{iter}) and less expensive. Its full performance characteristics will now be drawn up by using the criteria described previously (Sec. II.D).

These performance data are reported in Table 2 and summed up in a polar diagram in Fig. 6. In this diagram, a bigger value represents a better performance, and the values increase with the distance from the center of the circle. The upper axis shows the improvement of the function, the bottom one represents the total time of the optimization, the right side of the diagram shows the exploration criterion, and the left side displays the exploitation criterion. DOT–BFGS is quicker than RS-based optimizers, because it performs very few explorations of the domain. It performs around 50 times less exploration than the RS-based optimizers. The RS–kriging optimizer achieves higher total exploration (+50%) and exploration per evaluation values than the RS–cokriging optimizer. It is important to note that the RS–kriging reaches a plateau of convergence in 48 iterations (DOT–BFGS needs 41 iterations to converge), but it needs around twice this number of iterations (90) to converge. The triggered stopping criterion (no new point) corresponds to the fact that RS–kriging fails

Table 3 Optimization of the AS28 wing: aerodynamic performance of the baseline and optimized shapes

	Baseline	DOT–BFGS	RS–kriging	RS–cokriging
$Cd_p = F$	100	95.1 (−4.9%)	94.6 (−5.4%)	93.5 (−6.5%)
Cd_w	100	28.3	28.3	11.2
Cd_{vp}	100	99.2	98.3	98.7
Cd_i	100	103.2	102.9	103.7
Cl/Cd	100	106	106	108
AOA for $Cl = Cl_0$	2.2	2.07	2.09	2.07
Cd_p for $Cl = Cl_0$	100	90.9 (−9.1%)	93.3 (−6.7%)	90.2 (−9.8%)
Cl/Cd for $Cl = Cl_0$	100	107	108	108

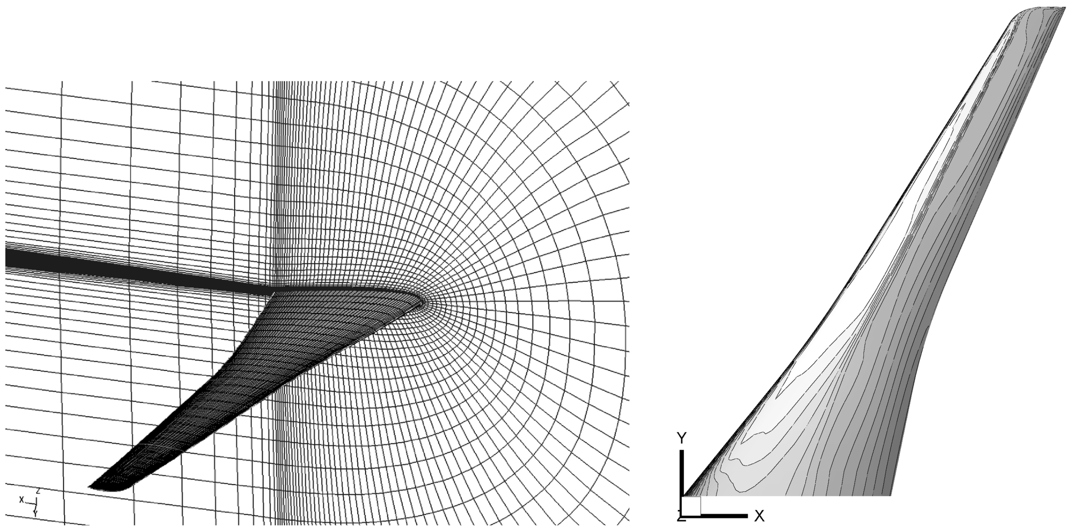


Fig. 7 Optimization of the AS28 wing: view of the mesh (left) and pressure coefficient distribution on the upper surface (right) for the baseline shape.

to find unsampled locations to explore with respect to the condition stated in Eq. (17). In terms of exploitation, the best choice is definitely DOT–BFGS, followed by RS–cokriging (−55%) and RS–kriging (−65%).

The additional gradient information used by RS–cokriging does not seem absolutely necessary, certainly due to the low dimensionality of the problem. It seems to effectively produce a more accurate surrogate model, as can be deduced from the lower values of exploration when compared with RS–kriging, but it only slightly accelerates the convergence of the optimization process.

The experience of numerous airfoil optimizations with OPTALIA demonstrated that it is impossible to indefinitely increase the number of design variables on two-dimensional airfoils without having highly correlated design variables. By considering only two bumps, one can expect that the correlation between design variables is low and that the complexity of the optimization problem is effectively represented by the number of design variables.

B. Wing-Drag Minimization, Considering 45 Design Variables and One Function

To increase the number of design variables with as little redundancy between variables as possible, a three-dimensional shape is considered: the AS28 wing in cruise condition, at a Mach number of 0.8 and an AOA of 2.2 deg. The wing span is 25 m, and the mean chord value is $c = 5.4$ m, giving a Reynolds number of $Re = 40 \cdot 10^6$. The structured mesh (Fig. 7) is formed of four blocks, containing a total of $500 \cdot 10^3$ nodes with its boundary-layer refinement. The wall-clock time for one steady flow simulation is more than 1 h, using two AMD Opteron 275 (2.2 Ghz) processors to perform 500 iterations.

The 45 shape parameters correspond to 15 Hicks–Henne bump functions on the upper surface. The bumps are distributed by group of three, in five spanwise sections, and are oriented along the z axis. At each section, the three bumps are equidistributed, and an overlapping is allowed between two neighboring bumps. To ensure a wide variety of possible shapes (maximum degrees of freedom) with this parameterization, the linear spanwise expansion of each sectional deformation is stopped at either the closest boundary or at the closest deformed section. Only positive deformations are allowed, so that the internal volume of the wing can only increase and the maximum amplitude of one bump is 50 mm. Compared with the previous RAE2822 airfoil optimization case, the maximum amplitude scaled with respect to the mean chord value $A^{\max}/c$ has increased from 0.5 to 1.0%. The complexity of this problem is thus increased by considering more independent design variables n_{dv} and also by increasing the relative range of possible deformation $A^{\max}/c$.

The RS-based optimizers were initialized with only 11 samples, $n_s^{\text{ini}} = 11$ and $N^{\text{ini}} = 11$, and the number of samples is even less than the number of variables. The initial database was formed from the baseline configuration plus 10 space-filling samples (LHS method). The cokriging sampling was augmented by including the information of five gradient vectors, $n_{\text{aug}} = 5$, giving 225 additional scalar values ($N^{\text{ini}}_{\text{RS Cokriging}} = 236$). The starting point for the gradient algorithm corresponds to large equidistributed bumps of null amplitude.

In cruise condition, a shock wave appears on the upper surface of the baseline configuration. Table 3 shows that the drag reduction of most of the optimizers comes from the minimization of the wave drag and leads to shapes with very weak shocks, as can be verified on the pressure distribution on the wing skin (cf., Fig. 8–11). Here again, all optimizers find different shapes (cf., Figs. 9–12). DOT–BFGS converges to the most regular deformation, whereas RS-based optimizers find both more original and detailed shape deformations (Fig. 8).

Table 4 Optimization of the AS28 wing: performance of the optimizers DOT–BFGS, RS–kriging, and RS–cokriging (the last and next-to-last set of lines show, respectively, the performance in terms of exploration and exploitation)

	DOT–BFGS	RS–kriging	RS–cokriging
Improvement $I(f)$	4.9%	5.4%	6.5%
n_{iter}; $n_{\text{eval}} + n_{\text{grad}}$; n_s^{ini}; N	26; 26 + 4; 0; 206	66; 209 + 0; 11; 209	87; 208 + 5; 11; 433
Stopping criterion	No improvement	Max n_{eval}	Max n_{eval}
$I(f)$ at $n_{\text{eval}} = 26$; $n_{\text{iter}} = 26$	4.9%; 4.9%	3.1%; 4.1%	4.7%; 5.7%
$I(f)/n_{\text{eval}}$	$18.8 \cdot 10^{-2}$	$2.58 \cdot 10^{-2}$	$3.12 \cdot 10^{-2}$
$I(f)/N$	$2.38 \cdot 10^{-2}$	$2.58 \cdot 10^{-2}$	$1.50 \cdot 10^{-2}$
$\sum \lvert F_c - F_c^{\text{ref}} \rvert$	26.8	534.4	205.5
$(1/n_{\text{dv}}) \sum \lVert \bar{x}_c - \bar{x}_c^{\text{ref}} \rVert$	0.149	8.83	3.03
$(1/n_{\text{dv}}) \sum \lVert \bar{x}_c - \bar{x}_c^{\text{ref}} \rVert / n_{\text{eval}}$	$0.57 \cdot 10^{-2}$	$4.22 \cdot 10^{-2}$	$1.46 \cdot 10^{-2}$
$(1/n_{\text{dv}}) \sum \lVert \bar{x}_c - \bar{x}_c^{\text{ref}} \rVert / N$	$0.723 \cdot 10^{-3}$	$42.2 \cdot 10^{-3}$	$7.00 \cdot 10^{-3}$

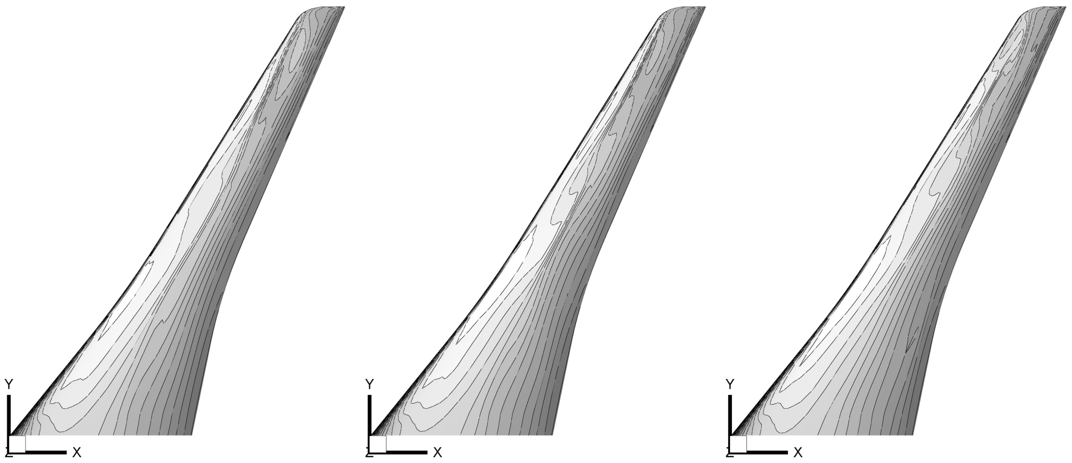


Fig. 8 Optimization of the AS28 wing: pressure coefficient contours on the upper surfaces of the optimal shapes given by DOT–BFGS (left), RS–kriging (middle), and RS–cokriging (right) optimizers.

Figure 13 presents the convergence history of the optimizations. The gradient-based algorithm keeps the same properties, as with the previous test problem. It converges using few process iterations and very few objective function evaluations (even fewer than on the six-dimensional problem). Despite its speed, it gives the lowest improvement of the objective function due its lack of domain exploration (Table 4). Contrary to the previous case, the first iteration of the RS-based optimizers (corresponding to space-filling samples) does not bring any improvement of the function, but it appears that RS–cokriging and DOT–BFGS have the same improvement speed during the first 20 iterations. When considering the last process iteration of DOT–BFGS ($n_{\text{iter}} = 26$), the function value given by the gradient optimizer is larger than the function value given by the cokriging-based optimizer. One could say that RS–cokriging has similar exploitation performance (improvement per iteration and improvement per evaluation) to DOT–BFGS when considering only the first 26 iterations. However, the RS–cokriging optimizer has not yet reached a plateau at this iteration and needs three times more

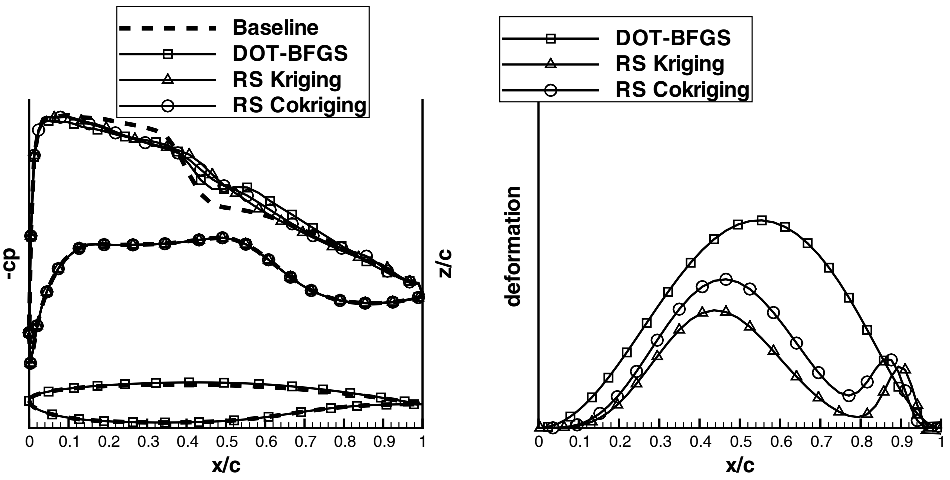


Fig. 9 Optimization of the AS28 wing: pressure coefficient distributions (left) and optimal deformations of the upper surface (right) given by DOT–BFGS, RS–kriging, and RS–cokriging optimizers at the wing section located at 35% of the wing span.

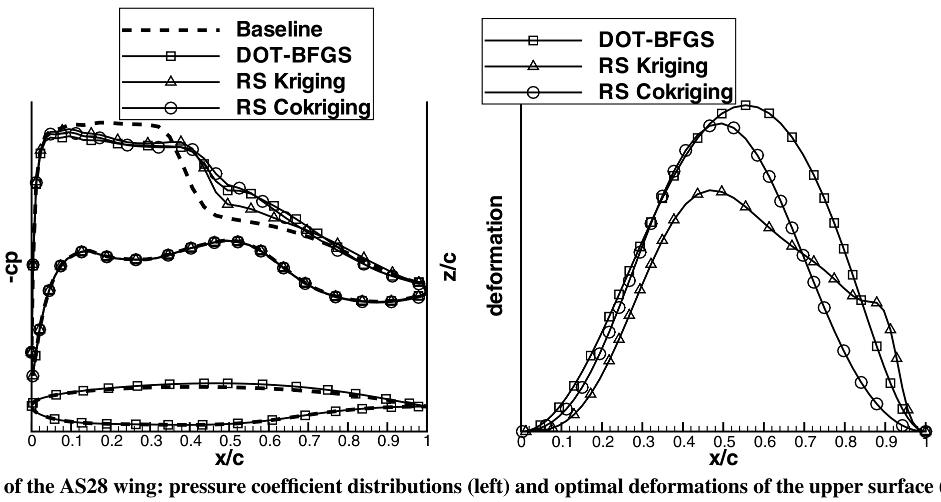


Fig. 10 Optimization of the AS28 wing: pressure coefficient distributions (left) and optimal deformations of the upper surface (right) given by DOT–BFGS, RS–kriging, and RS–cokriging optimizers at the wing section located at 50% of the wing span.

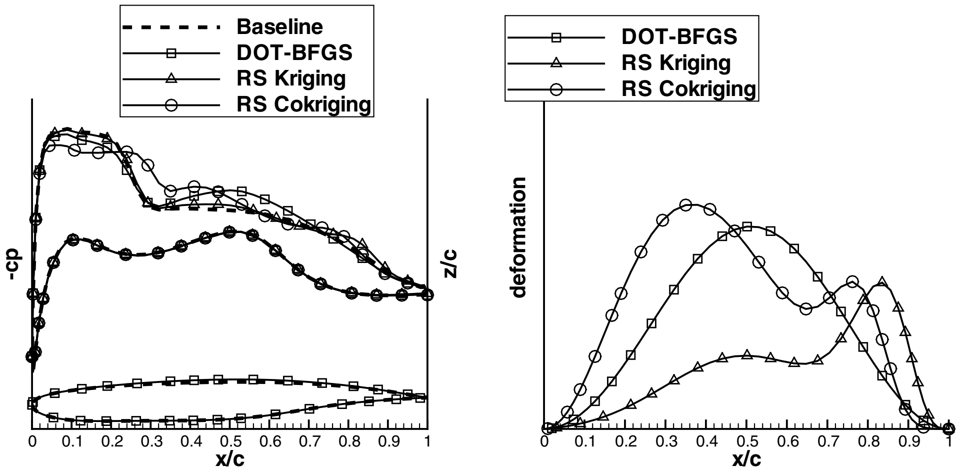


Fig. 11 Optimization of the AS28 wing: pressure coefficient distributions (left) and optimal deformations of the upper surface (right) given by DOT–BFGS, RS–kriging, and RS–cokriging optimizers at the wing section located at 85% of the wing span.

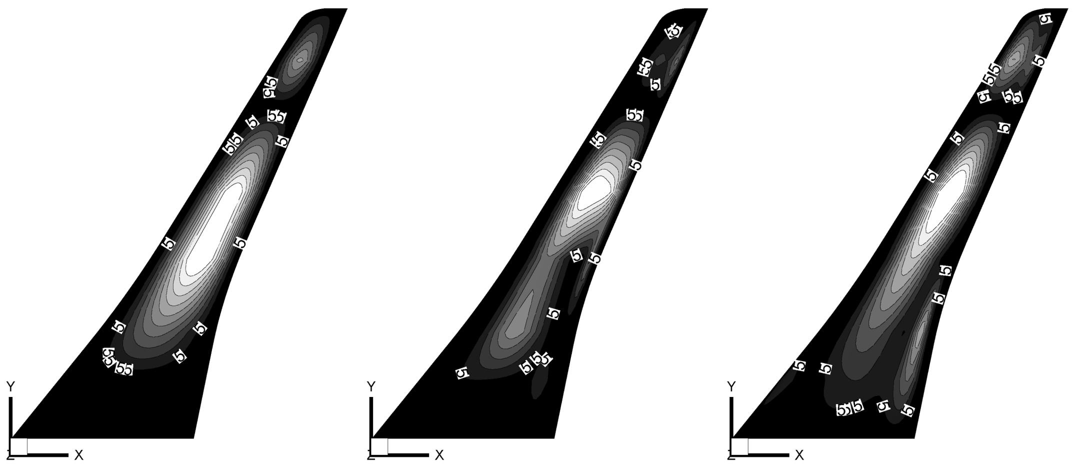


Fig. 12 Optimization of the AS28 wing: optimal deformations of the upper surface given by DOT–BFGS (left), RS–kriging (middle), and RS–cokriging (right) optimizers.

iterations to converge. Overall, DOT–BFGS is largely better in terms of exploitation.

For the RS-based optimizations, the influence of the very coarse initial sampling can be directly observed by comparing exploration values of kriging and cokriging optimizers in Table 4 and Fig. 14. As the kriging RS is less accurate, RS–kriging performs three times more exploration and needs more iterations to converge to its final value. Despite that, a significant improvement of the function is achieved, and RS–kriging slightly outperforms the gradient reference. Somehow, its convergence is stopped because the maximum number of authorized function evaluations is reached (200), whereas the optimizer continues its domain exploration,

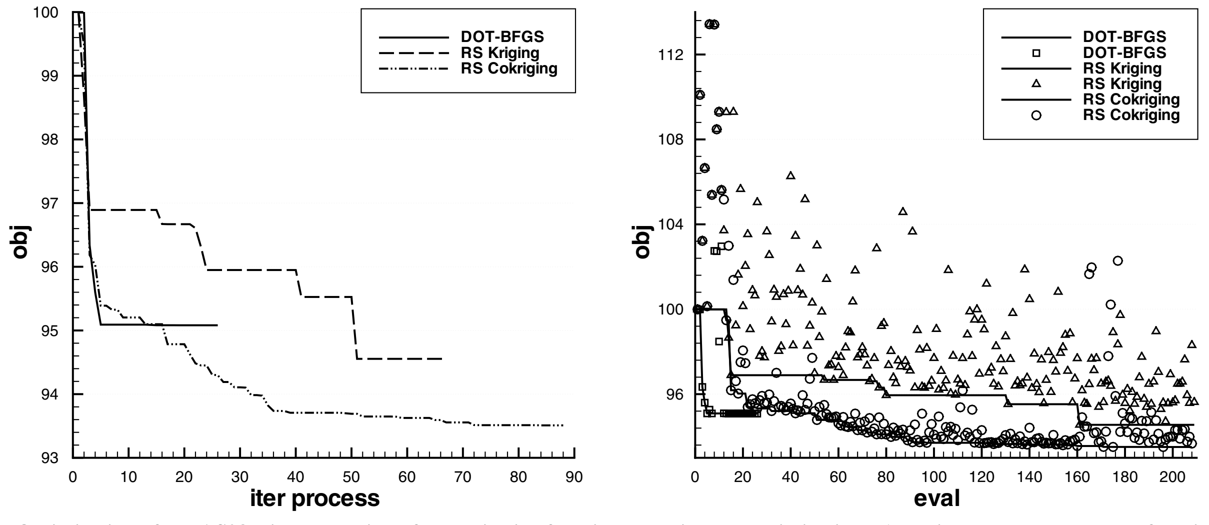


Fig. 13 Optimization of the AS28 wing: evolution of the objective function F during the optimizations. (The lines represent the best function value explored by the optimizer F^{ref} at n_{iter} for the left figure and n_{eval} for the right figure. On the right, the symbols represent the function values of all shapes explored during the optimizations at evaluation n_{eval}.

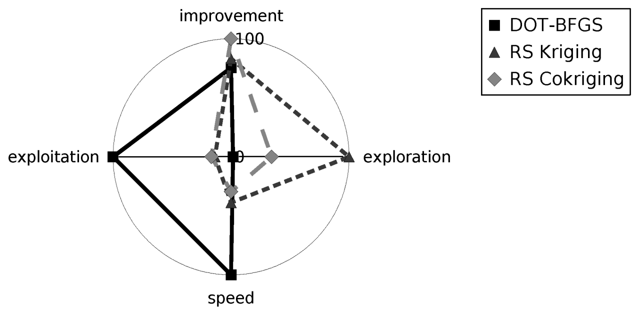


Fig. 14 Optimization of the AS28 wing: polar diagram summarizing the performance of the optimizers. [Improvement corresponds to $I(f)$, exploration corresponds to $(1/n_{\mathrm{dv}})\sum\|\bar{x}_c - \bar{x}_c^{\mathrm{ref}}\|$, speed corresponds to $1/n_{\mathrm{iter}}$, and exploitation corresponds to $I(f)/n_{\mathrm{eval}}$].

characterized by the large wide spreading of symbols on Fig. 13. The kriging-based optimizer seems to need more iterations to converge to a minimum of the function from the mathematical point of view (respecting the Karush–Kuhn–Tucker condition). In other words, when the increase in the number of design variables is not taken into account in the size of the initial RS database, the optimizer needs more iterations to converge. The RS–cokriging optimizer is also stopped by the same criterion, but this premature convergence seems less problematic in this case. In fact, the optimizer seems close to satisfying the usual convergence criterion (no improvement during 20 consecutive iterations), because very few improvements are achieved during the last 50 iterations, despite the continuous exploration of the domain.

It is interesting to compare DOT–BFGS and RS–kriging optimizers. Their results in terms of function improvement and improvement per quantity of information are very close, whereas the methods are completely different. The quantity of scalar values needed to converge is the same for both optimizers, but DOT–BFGS converges much quicker than RS–kriging. Contrary to RS–kriging, it performs very little exploration.

The sample-limited cokriging-based optimizer reaches the best objective function improvement. The relative gap between RS–kriging and RS–cokriging improvement is more than twice the gap between RS–kriging and DOT–BFGS improvement (Fig. 14). However, it appears that RS–cokriging performs less exploration than RS–kriging ($\approx -70\%$). This means that the exploration conducted during the RS–kriging optimization is considerably less efficient. This is due to the fact that the RS is less accurate, as it does not include any gradient information. The interest of the adjoint method coupled with a cokriging model is thus demonstrated on this high-dimensional problem.

V. Conclusions

A general framework for optimization enabling the use of various optimizers was set up. Two algorithms based on RSs built with kriging and cokriging were then implemented and compared with the quasi-Newton reference algorithm DOT–BFGS. The cokriging model was built using an original sample-limited strategy that enable the limitation of the computational cost of the model needed to build the model. The RS-based algorithms use an original multicriteria sampling refinement process and require, at each iteration, a global minimization of three carefully chosen criteria based on combinations of the predicted function and error ($C_1 = F(x)$, $C_2 = F(x) - S(x)$, and $C_3 = F(x) + S(x)$). Despite its efficiency, the global optimization strategy used for this purpose needs to be simplified in the near future.

Because RS and gradient-based optimizers are very different in nature, a set of measures was defined in order to characterize the general performance of optimizers. From these criteria, it appears relevant to retain four competing measures: the improvement of the function, the time (cost) of the optimization, the exploration value, and the exploitation value. The comparison of optimizers using these performance measures effectively enables the description of their strengths and weaknesses, and the authors would like to promote the use of such criteria in order to assess the comparative performance profiles of new optimizers. In all cases, the reference gradient optimizer DOT–BFGS is the quickest method and achieves the best exploitation values, but it gives the lowest improvements of the function. This is linked to the fact that it performs very few explorations and converges along a descent path, whereas the problems treated in numerical aerodynamic shape optimization are multimodal. The RS-based optimizers seem very promising, as they achieve better function improvements and exploration values and are complementary to the reference gradient optimizer. Best practices can be inferred from the results obtained on the two drag-reduction test problems presented.

For the low-dimensional problem ($n_{\mathrm{dv}} = 6$), both RS-based optimizers largely outperform the gradient-based algorithm in terms of total improvement of the function, but they also outperform it at equivalent computational cost and wall-clock time. Despite the fact that the gradient algorithm converges to a local optimum, it needs fewer function evaluations to reach its final optimum. The interpolation of gradient information does not significantly improve the performance of the cokriging-based optimizer, which is why, in conclusion, the kriging-based optimizer should be preferred for low-dimensional problems.

When increasing the complexity of the problem by considering 45 design variables on a wing, the kriging-based optimizer requires twice as many iterations after gradient convergence before outmatching DOT–BFGS. It appears that, due to the lack of accuracy of the kriging model (insufficient size of the sample database), this optimizer wastes computational resources in an excessive exploration of the domain. The sample-limited cokriging-based optimizer does not suffer from this problem. It largely gives the best improvement and also manages to outmatch the gradient reference at equivalent wall-clock time, even if it needs three times as many iterations to converge. The total additional cost needed is moderate in terms of wall-clock time, due to the efficiency of the parallel framework, but it is large in terms of total number of function evaluations. However, it remains in the imposed limit of a maximum of 200 CFD runs per optimization. The cokriging-based optimizer should be preferred for this type of problems.

Overall, the gradient optimizer DOT–BFGS converges very quickly, but it does not reach the best function improvement (it converges toward a local optimum). It should be preferred when the computational cost is severely limited (several ten CFD runs) or when the number of variables is too high to compute a sample-limited cokriging (several hundred variables) inexpensively. One can again see here that there is no free lunch in optimization and that, in order to obtain a better solution, one has to pay the computational price.

These behaviors can be explained by considering the quantity of information N used by the optimizers to propose a new vector of design variables. At a given iteration, DOT–BFGS analyzes the information of two gradient vectors to determine a search direction plus a few function evaluations to perform a line search. It proposes a new shape based on $N_{\mathrm{Gradient}} \approx 10 + 2 \cdot n_{\mathrm{dv}}$ scalar information on the unknown function. The use of gradient vectors implies that this quantity depends on the number of variables, which explains why the gradient algorithm is not much affected by the dimensionality of the problem. However, this quantity of information does not depend on the number of evaluations or the number of iterations. This means that the gradient algorithm ignores some previously computed information on the function. It only uses the information in the vicinity of the current shape. Conversely, RSs exploit all known information on the function through the sample database, and the sampling refinement process aims at exploiting this database effectively in order to find the optimum. For the kriging-based algorithm, the quantity of information is independent of the number of variables $N_{\mathrm{RS\ Kriging}} = n_{\mathrm{eval}}$, explaining its domain of applicability (low-dimensional problems). The cokriging-based algorithm interpolates a fixed number n_{aug} of gradient vectors, $N_{\mathrm{RS\ Kriging}} = n_{\mathrm{eval}} + n_{\mathrm{aug}} \cdot n_{\mathrm{dv}}$, giving an efficient algorithm for even high-dimensional problems.

Further improvement of these RS-based algorithms could be made regarding the possibilities offered by the multicriteria sampling refinement process. It would be interesting to increase the size of the population by using, at each iteration, multiple local optima of the sampling criteria as candidates. It could be done by keeping the three criteria unchanged but limiting their optimizations to different subzones of the domain.

Acknowledgments

The computer implementation of J. Burkardt was used to generate the initial space-filling sampling, and the genetic algorithm of D. Caroll was used for optimization on response surfaces, and so the authors would like to thank them for sharing their work. The authors also thank H. Moet for helping them correct this paper.

References

[1] Hicks, R. M., and Henne, P., "Wing Design by Numerical Optimization," *Journal of Aircraft*, Vol. 15, No. 7, 1978, pp. 407–412. doi:10.2514/3.58379

[2] Kim, S., Alonso, J. J., and Jameson, A., "Multi-Element High-Lift Configuration Design Optimization Using Viscous Continuous Adjoint Method," *Journal of Aircraft*, Vol. 41, No. 5, 2004, pp. 1082–1097. doi:10.2514/1.17

[3] Le Moigne, A., and Qin, N., "Variable-Fidelity Aerodynamic Optimization for Turbulent Flows Using a Discrete Adjoint Formulation," *AIAA Journal*, Vol. 42, No. 7, 2004, pp. 1281–1292. doi:10.2514/1.2109

[4] Meaux, M., Cormery, M., and Voizard, G., "Viscous Aerodynamic Shape Optimization Based on the Discrete Adjoint State for 3D Industrial Configurations," *European Congress on Computational Methods in Applied Sciences and Engineering* [CD-ROM], University of Jyväskylä, Jyväskylä, Finland, July 2004.

[5] Widhalm, M., Ronzheimer, A., and Hepperle, M., "Comparison Between Gradient-Free and Adjoint Based Aerodynamic Optimization of a Flying Wing Transport Aircraft in the Preliminary Design," 25th Applied Aerodynamics Conference, AIAA, Paper 07-4060, 2007.

[6] Jones, D. R., Schonlau, M., and Welch, W. J., "Efficient Global Optimization of Expensive Black-Box Functions," *Journal of Global Optimization*, Vol. 13, No. 4, 1998, pp. 455–492. doi:10.1023/A:1008306431147

[7] Torczon, V., and Trosset, M. W., "Using Approximations to Accelerate Engineering Design Optimization," 7th AIAA/USAF/NASA/ISSMO Symposium on Multidisciplinary Analysis and Optimization, AIAA, Paper 1998-4800, 1998; also *Collection of Technical Papers*, Pt. 2, A98-39701 10-31.

[8] Alexandrov, N. M., Lewis, R. M., Gumbert, R. C., Lawrence, L. L., and Newman, P. A., "Approximation and Model Management in Aerodynamic Optimization with Variable-Fidelity Models," *Journal of Aircraft*, Vol. 38, No. 6, 2001, pp. 1093–1101. doi:10.2514/2.2877

[9] Ong, Y. S., Nair, P. B., and Keane, A. J., "Evolutionary Optimization of Computationally Expensive Problems via Surrogate Modelling," *AIAA Journal*, Vol. 41, No. 4, 2003, pp. 687–696. doi:10.2514/2.1999

[10] Peigin, S., and Epstein, B., "Robust Optimization of 2D Airfoils Driven by Full Navier–Stokes Computations," *Computers and Fluids*, Vol. 33, No. 9, 2004, pp. 1175–1200. doi:10.1016/j.compfluid.2003.11.001

[11] Mouton, S., Laurenceau, J., and Carrier, G., "Aerodynamic and Structural Optimisation of Powerplant Integration Under the Wing of a Transonic Transport Aircraft," *42ème Colloque d'Aérodynamique Appliquée AAAF*, Association Aeronautique et Astronautique de France, Paris, 2007.

[12] Liu, W., "Development of Gradient-Enhanced Kriging Approximations for Multidisciplinary Design Optimization," Ph.D. Thesis, Univ. of Notre Dame, Notre Dame, IN, 2003.

[13] Epstein, B., Jameson, A., Peigin, S., Roman, D., Harrison, H., and Vassberg, J., "Comparative Study of Three- Dimensional Wing Drag Minimization by Different Optimization Techniques," *Journal of Aircraft*, Vol. 46, No. 2, 2009, pp. 526–545. doi:10.2514/1.38216

[14] Vanderplaats, G. N., "D.O.T. Users Manual," Vanderplaats Research and Development, Colorado Springs, CO, 1995.

[15] Cambier, L., and Veuillot, J.-P., "Status of the ELSA CFD Software for Flow Simulation and Multi-Disciplinary Applications," 46th Aerospace Sciences Meeting and Exhibit, AIAA, Paper 2008-664, 2008.

[16] Brodersen, O., Rakowitz, M., Amant, S., Larrieu, P., Destarac, D., and Sutcliffe, M., "Airbus, ONERA, and DLR Results from the Second AIAA Drag Prediction Workshop," *Journal of Aircraft*, Vol. 42, No. 4, 2005, pp. 932–940; also AIAA, Paper 2004-0391, 2004. doi:10.2514/1.8662

[17] Jameson, A., "Aerodynamic Design Via Control Theory," *Journal of Scientific Computing*, Vol. 3, No. 3, 1988, pp. 233–260. doi:10.1007/BF01061285

[18] Peter, J., Pham, C.-T., and Drullion, F., "Contribution to Discrete Implicit Gradient and Discrete Adjoint Method for Aerodynamic Shape Optimisation," *European Conference on Computational Fluid Dynamics* [CD-ROM], University of Jyväskylä, Jyväskylä, Finland, 2004.

[19] Destarac, D., and van der Vooren, J., "Drag/Thrust Analysis of Jet-Propelled Transonic Transport Aircraft; Definition of Physical Drag Components," *Aerospace Science and Technology*, Vol. 8, No. 6, 2004, pp. 545–556. doi:10.1016/j.ast.2004.03.004

[20] Bellman, R., *Adaptive Control Processes: A Guided Tour*, Princeton Univ. Press, Princeton, NJ, 1961.

[21] Barthelemy, J.-F. M., and Haftka, R. T., "Approximation Concepts for Optimum Structural Design: A Review," *Structural optimization*, Vol. 5, No. 3, 1993, pp. 129–144. doi:10.1007/BF01743349

[22] Laurenceau, J., and Sagaut, P., "Building Efficient Response Surfaces of Aerodynamic Functions with Kriging and Cokriging," *AIAA Journal*, Vol. 46, No. 2, 2008, pp. 498–507. doi:10.2514/1.32308

[23] Sacks, J., Welch, W. J., Mitchell, T. J., and Wynn, H. P., "Design and Analysis of Computer Experiments," *Statistical Science*, Vol. 4, No. 4, 1989, pp. 409–435. doi:10.1214/ss/1177012413

[24] Chung, H.-S., and Alonso, J. J., "Using Gradients to Construct Cokriging Approximation Models for High Dimensional Design Optimization Problems," 40th Aerospace Science Meeting and Exhibit, AIAA, Paper 02-0317, 2002.

[25] Sobester, A., Leary, S. J., and Keane, A. J., "A Parallel Updating Scheme for Approximating and Optimizing High Fidelity Computer Simulations," *Structural and Multidisciplinary Optimization*, Vol. 27, No. 5, 2004, pp. 371–383. doi:10.1007/s00158-004-0397-9

[26] Booker, A. J., Dennis, J. E., Frank, P. D., Serafini, D. B., Torczon, V., and Trosset, M. W., "A Rigorous Framework for Optimization of Expensive Functions by Surrogates," *Structural Optimization*, Vol. 17, No. 1, 1999, pp. 1–13. doi:10.1007/BF01197708

[27] Leary, S. J., Bhaskar, A., and Keane, A. J., "Global Approximation and Optimization Using Adjoint Computational Fluid Dynamics Codes," *AIAA Journal*, Vol. 42, No. 3, 2004, pp. 631–641. doi:10.2514/1.9114

[28] Sasena, M. J., "Flexibility and Efficiency Enhancements for Constrained Global Design Optimization with Kriging Approximations," Ph.D. Thesis, Dept. of Mechanical Engineering, Univ. of Michigan, Ann Arbor, MI, 2002.

[29] Jeong, S., Murayama, M., and Yamamoto, K., "Efficient Optimization Design Method Using Kriging Model," *Journal of Aircraft*, Vol. 42, No. 2, 2005, pp. 413–419. doi:10.2514/1.6386

[30] Meunier, M., "Simulation and Optimization of Flow Control Strategies for Novel High-Lift Configurations," *AIAA Journal*, Vol. 47, No. 5, 2009, pp. 1145–1157. doi:10.2514/1.38245

[31] Cox, D., and John, S., "SDO: A Statistical Method for Global Optimization," *Multidisciplinary Design Optimization: State-of-the-Art*, edited by N. M. Alexandrov, and M. Y. Hussaini, Society for Industrial and Applied Mathematics, Philadelphia, 1997, pp. 315–329.

[32] Forrester, A. I. J., and Keane, A. J., "Recent Advances in Surrogate-Based Optimization," *Progress in Aerospace Sciences*, Vol. 45, No. 1, 2009, pp. 50–79. doi:10.1016/j.paerosci.2008.11.001

[33] Wolpert, D. H., and Macready, W. G., "No Free Lunch Theorems for Optimization," *IEEE Transactions on Evolutionary Computation*, Vol. 1, No. 1, April 1997, pp. 67–82. doi:10.1109/4235.585893

T. Zang
Associate Editor