Automatic Design Optimization Applied to Lean Premixed Combustor Cooling

F. Duchaine[1]*, L. Y. M. Gicquel[1], D. Bissières[2], C. Bérat[2] and T. Poinsot[3]

[1]*CERFACS,* [3]*IMFT - Toulouse France,* [2]*TURBOMECA - Bordes France*

The design of gas turbine engines is a complex and time consuming engineering task involving numerous iterative processes. Among all the devices composing the engine, the combustion chamber benefits from recent progress in High Performance Computing for Computational Fluid Dynamics (CFD) which is intensively used. Although CFD offers substential improvements in flexibility and costs when compared to experimental approaches, the number of CFD computations still remains large and engineering intensive. A strategy to alleviate the CFD specialist involvment in reaching an optimal solution is desired to fully capitalize on the method. One solution aims at automatizing the optimization performed with CFD.

From the optimization point of view, the expensive step is the CFD run necessary to evaluate the objective function value for a set of control parameters. The promising method developed in this work consists in performing optimization with classical techniques based on an inexpensive metamodeling of the objective function. The idea is to reduce the number of CFD computations while proposing a rigorous framework to converge robustly to an optimal design. An adaptive Gaussian Process model[1] is used to approximate the fitness. The model is continuously updated by increasing the reference database as the search progresses. Implementation issues such as efficiency, numerical stability of the model, techniques to accelerate the optimization process, exploration / exploitation conflict and the different levels of parallelism involved in CFD optimization are addressed. Practicallity of the optimization tool is here inherited from the powerful coupling device PALM.[2] PALM allows independent developments of the different units composing the application offering a high level of flexibility in terms of evolution and enhancement.

Illustration of the approach when applied to a simple engine cooling system is presented: Validations steps and comparisons of the proposed surrogate approach with the very well known optimization method Simplex[3] are shown.

Introduction

Due to drastic norms on pollution, aeronautical engine manufacturers need to propose new technological solutions for the next generation of aero-engine devices. Indeed, ambitious NOx reduction targets of 80% are set for 2020. These strict objectives yield many research projects aiming at the definitions and studies of innovative combustion systems. Among all the initiatives from the European research community, the project INTELLECT D.M. number FP6-502961 (INTEgrated Lean Low Emission CombusTor Design Methodology) investigates new design rules and methodologies for the definition and validation of low emission combustors. The specificity of the retained technology for NOx reduction is that the combustion chambers will more likely operate with an excess of air; the primary reason being to reduce significantly the flame temperature and thereof the NOx production. With this approach, up to 70% of the total air flow through the chamber is premixed with the fuel before entering the reaction zone. As a consequence, the design of the pre-diffuser and cooling systems become crucial as they are key elements to provide good air distribution throughout the chamber and to avoid large values of the temperature peaks. Another characteristic of this technology is the narrowing of the operating range of the new burners. Indeed, combustion instabilities arise rapidly even with small deviations from the initially defined operating point.

This work, initiated by Turbomeca (SAFRAN Group), focuses on the first issue and analyzes the suitability of optimization techniques for the design of an areo-engine cooling system. The main idea is to assess and construct an automatic optimizer based on a CFD solver so as to obtain design solutions satisfying given chamber outlet temperature profiles. Nowadays, optimization takes a very large place in the scientific community and real world offers numerous examples where one can use optimization. However, optimization method performances depend on the type of problem to be treated. The coupling of an optimizer with a CFD solver imposes obvious restrictions such as computer power, time and memory... Among existing methods, the most performant optimization processes are the gradient-based techniques. They are nonetheless not accessible with standard CFD codes and adjoint solvers are often mandatory to obtain good evaluations of the gradients. More recent methods, such as evolutionary algorithms, also known as genetic algorithms,[4] are very often used in CFD optimization. They give good results and coincide with the philosophy of CFD, that is they allow several computations of differents design points at the same time. The number of objective function computations (CFD runs) remains large with this approach and the overall response-time of the process is large. An original approach to avoid such problems and adopted in this work, consists in using a low order model, or surrogate model,[5] to substitute the CFD runs. Standard optimization algorithms are then applied on this model. The overall process requires less computing time.

* Florent.Duchaine@cerfacs.fr

The paper is organized as follow: First, we present the surrogate model used to approximate the objective function. Then, a general presentation of the coupler PALM is done. Details and specifications of the target geometry to be optimized are then given. Finally, numerical results are presented for two a decision variable test case.

Optimization, Surrogate Models and Gaussian Processes

There are many ways to implement surrogate models to accelerate the convergence of an optimization precedure by reducing the number of time consuming CFD computations. Some authors use them in a local way to find the value of the approximated function based on a restricted number of sample points. Others, as in this article, use surrogate models as a global approximation of the fitness for the entire domain of optimization parameters. The resulted global surface gives a tendency of the comportement of the fitness versus each variables. Other variations in the existing methods appears in the alternate use of the true fitness values or the approximations given via the surrogate model. For example, when evolutionnary programming is concerned, authors assign values to a part of a generation based on CFD computations while the other part inherit their value from the low order model. On the other hand all values can alternatively be obtained from the same model that is CFD and surrogate.

Surrogate Models

A wide variety of surrogate models are used in the literature to approximate fitness functions in the context of optimization. The most prominent among all approaches are polynomial models,[6] artificial neural networks,[7] radial basis function networks[8] and Gaussian process.[9]

Polynomial models, also referred as response surfaces, are easy to use and implement. They provide good results in the case of unimodal problems but tend to oscillate too much when fitting high order multimodal polynomials. Artificial Neural Networks consist of a large number of highly interconnected processing units, each aggregating information from a large number of connected peers. Given a certain number of units, it can approximate any function. The main drawback of this technique is the difficulty of finding an appropriate network topology and size. The output of the Radial Basis Function Network is a weighted sum of radial basis function characterized by a radial basis Kernel. Typical choices for the Kernel include linear splines, cubic splines, multiquadrics, thinplate spline and Gaussian functions. The major difficulty is to set the different parameters of the Kernel function which have a large impact on the approximations. Gaussian processes specify a probabilistic model over a given set of data points. It is constructed such that the likelihood of the function value given the decision variable values is maximized for all the data. Like Artificial Neural Networks, Gaussian processes can approximate any function. One common drawback to all of these methods is the associated computational cost.

Gaussian Process Model

Among the presented empirical models, Gaussian processes (GPs) appear to be the most promising for fitness function approximation. Indeed, a GP combines the following decisive properties:

- the implentation of GPs is independent of the number of decision variables,
- GPs can approximate arbitrary functions including multimodality and discontinuities,
- GP contains meaningful hyperparameters that can be obtained theoretically with an optimization procedure,
- GP yields an uncertainty measure of the predicted value (in the form of a standard deviation).

To expose the theorical framework of GPs, we follow the notations of MacKay.[1] The analysis makes use of some noisy data set D consisting of N pairs of L-dimensional input vectors and scalar outputs $\{\mathbf{x}_n, t_n\}_{n=1}^{N}$. Based on this set, the aim of the process is to find a prediction t_{N+1} at a new point $\mathbf{x}_{N+1} \notin D$. Denoting the set of input vectors by $\mathbf{X}_N = \{\mathbf{x}_1, \mathbf{x}_2, ..., \mathbf{x}_N\}$ and the set of corresponding function values by the vector $\mathbf{t}_N = \{t_1, t_2, ..., t_N\}$, the GP supposes a probabilistic model for a given data set. That is, the vector of known function values $\mathbf{t}_N$ is regarded as one sample of a multivariate Gaussian distribution with joint probability density $p(\mathbf{t}_N|\mathbf{X}_N)$. Addition of a new point $\mathbf{x}_{N+1}$ to the data set infers that the resulting vector of function values $\mathbf{t}_{N+1}$ is also a sample of the Gaussian joint probability density $p(\mathbf{t}_{N+1}|\mathbf{X}_{N+1}) \equiv p(\mathbf{t}_N, t_{N+1}|\mathbf{X}_N, \mathbf{x}_{N+1})$. Consequently, the dimension of the probability density and the process itself depend only on the number of data points and not on the decision space L.

Using the rule of conditional probabilities $p(A|B) = p(A, B)/p(B)$, the probability density for the function value t_{N+1} at a new point $\mathbf{x}_{N+1}$, given the N known data points $\mathbf{X}_N$ and their associated function values $\mathbf{t}_N$, can be expressed in term of a univariate Gaussian distribution,

$$p(t_{N+1}|\mathbf{X}_{N+1}, \mathbf{t}_N) = \frac{p(\mathbf{t}_{N+1}|\mathbf{X}_{N+1})}{p(\mathbf{t}_N|\mathbf{X}_N)}. \tag{1}$$

The multivariate Gaussian density appearing in the denominator of Eq. 1 is assumed to follow,

$$p(\mathbf{t}_N|\mathbf{X}_N) = \frac{1}{Z_N} exp\left(-\frac{1}{2}\mathbf{t}_N^T C_N^{-1}\mathbf{t}_N\right), \tag{2}$$

where C_N denotes the covariance matrix of the Gaussian distribution with zero mean for the sample points and $Z_N = \left((2\pi)^N \, det(C_N)\right)^{1/2}$ is an appropriate normalization constant. By extention, the $N+1$ data points satisfies,

$$p(\mathbf{t}_{N+1}|\mathbf{X}_{N+1}) = \frac{1}{Z_{N+1}} exp\left(-\frac{1}{2}\mathbf{t}_{N+1}^T C_{N+1}^{-1}\mathbf{t}_{N+1}\right), \tag{3}$$

where Z_{N+1} is again an appropriate normalization constant and the new covariance matrix C_{N+1} that can be written as,

$$C_{N+1} = \begin{pmatrix} C_N & \mathbf{k} \\ \mathbf{k}^T & \kappa \end{pmatrix}. \tag{4}$$

In Eq. 4, $\mathbf{k}$ is a vector containing the covariances between the new point $\mathbf{x}_{N+1}$ and the N known points while κ stands for the variance of the new point.
Substituting Eq. 2 and 3 in the right-hand side of Eq. 1 yields,

$$p(t_{N+1}|\mathbf{X}_{N+1}, \mathbf{t}_N) = \frac{1}{Z} exp\left(-\frac{1}{2}\frac{(t_{N+1} - \hat{t}_{N+1})^2}{\sigma^2_{\hat{t}_{N+1}}}\right). \tag{5}$$

Eq 5 is a univariate Gaussian density for t_{N+1} with mean and variance given by,

$$\hat{t}_{N+1} = \mathbf{k}^T C_N^{-1} t_N, \tag{6}$$

$$\sigma^2_{\hat{t}_{N+1}} = \kappa - \mathbf{k}^T C_N^{-1}\mathbf{k}. \tag{7}$$

The covariance matrices C_N and C_{N+1} are defined in order to bypass the step of expressing priors on the modelling function and by combining them into a covariance function C. Their are many ways to impose C with the only constraint that it must generate a non-negative definite covariance matrix for any set of points $\mathbf{X}_N$. In the context of this study, the stationary covariance function proposed by MacKay[1] is implemented and reads for a two data point set $\mathbf{x}_p$ and $\mathbf{x}_q$:

$$C(\mathbf{x}_p, \mathbf{x}_q, \Theta) = \theta_1 exp\left(-\frac{1}{2}\sum_{l=1}^{L}\frac{(x_{p,l} - x_{q,l})^2}{r_l^2}\right) + \theta_2 + \delta_{pq}\theta_3 \tag{8}$$

where l represents the l^{th} component of the L-dimensional vectors $\mathbf{x}_p$ and $\mathbf{x}_q$, and $\Theta = (\theta_1, \theta_2, \theta_3, r_l)$ are the hyperparameters of the covariance function. r_l corresponds to a length scale characterising the direction l. A large length scale means that the ouput value $\hat{t}_{N+1}$ is expected to be essentially a constant function of that input (smoothing effect). The θ_1 hyperparameter scale the correlation. θ_2 allows the whole function to be offset away from zero by an unknown constant. Finally, θ_3 represents a white input-independant noise applied only on to the diagonal terms of the covariance matrix. The covariance matrix C_N, the covariance vector $\mathbf{k}$ and the variance κ of Eq. 4 can be expressed in terms of the covariance function as:

$$C_{N_{ij}} = C(\mathbf{x}_i, \mathbf{x}_j),\ (\mathbf{x}_i, \mathbf{x}_j) \in D^2 \tag{9}$$

$$k_i = C(\mathbf{x}_i, \mathbf{x}_{N+1}),\ \mathbf{x}_i \in D \tag{10}$$

$$\kappa = C(\mathbf{x}_{N+1}, \mathbf{x}_{N+1}) = \theta_1 + \theta_2 + \theta_3 \tag{11}$$

The hyperparameters can always either be set by the user or retrieved from the data. As the values of the hyperparameters have a large influence on the result of the GP, we opt for an optimal approximation such that the log-likelihood of the given function values $\mathbf{t}_N$ under multivariate Gaussian with zero mean and covariance $C_N = C(\mathbf{X}_N, \Theta)$ is maximal. This log-likelihood can be expressed as:

$$\begin{aligned} \lambda &= log(p(\mathbf{t}_N)|\mathbf{t}_N)) \\ &= -\tfrac{1}{2}\left(log\ detC_N + \mathbf{t}_N^T C_N^{-1}\mathbf{t}_N + N\ log(2\pi)\right) \end{aligned} \tag{12}$$

Even if it is straightforward to find the derivatives $\frac{\partial\lambda}{\partial\Theta}$, MacKay[1] shows that λ is often multimodal. For this reason, we use an evolutionary algorithm proposed by Michalewicz[10] to find the optimal hyperparameters.
To conclude on that part, we can note that most of CPU time requested by the Gaussian method is due to the inversion of the covariance matrix C_N in Eqs. 6, 7 and 12. To reduce the potential impact of this disadvantage, numerical tricks as LU decomposition can be used.

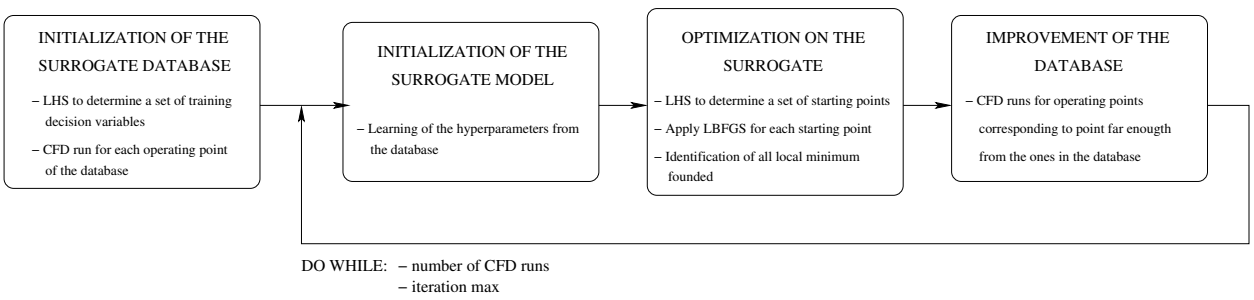

Fig. 1 Flowchart of the surrogate assisted optimization algorithm

The algorithm to be used is detailed on Fig. 1. Prior to the optimization loop, one requires the use of the Design Of Experiments (DOE) approach to initialize the surrogate database D. To extract as much information as possible from a limited set of computer experiments, the chosen method for space filling is based on the Latin Hypercube Sampling[14] (LHS). LHS allows to generate a satisfactory disbution of points in a given search domain.
The optimization loop then continues for a pre-definied number of iteration or when the maximum allowed CFD computations is done. Optimization of the surrogate model is obtained from a "Low memory Broyden Fletcher Goldfard Shanno Bounded" (LBFGS-B) which corresponds to the quasi-Newton algorithm proposed by Byrd *et al.*[15] One iteration of the surrogate-based method performs several LBFGS-B runs with different starting points spreaded over the decison domain. Hence, the gradients of the fitness function needed by the LBFGS are exactly computed by derivation of Eq. 6 and 7:

$$\frac{\partial \hat{t}_{N+1}}{\partial x_l} = \frac{\partial \mathbf{k}}{\partial x_l}^T C_N^{-1} t_N \tag{13}$$

$$\frac{\partial \sigma^2_{\hat{t}_{N+1}}}{\partial x_l} = -\frac{\partial \mathbf{k}}{\partial x_l}^T C_N^{-1}\mathbf{k} - \mathbf{k}^T C_N^{-1}\frac{\partial \mathbf{k}}{\partial x_l} \tag{14}$$

Using a gradient-based optimization algorithm coupled with a surrogate approach yields good exploitation of the disponible data which is essential for fast converge. The multi-start aspect of the gradient optimizer enforced by the use of a merite function instead of the fitness function allows a great exploration of the decision domain. The notion of exploration is needed in the case of multimodal function in order to find a global extremum. Merite function is a weighted sum of fitness function and estimation of the density ρ of points at a given place: $f_M = f_{obj} - \alpha\rho$, $\alpha > 0$. Gaussian processes give naturally a notion of the density through the variance of the estimation $\sigma^2_{\hat{t}}$ which is high where the prediction $\hat{t}$ is probably not accurate. So the optimization is performed on $f_M = \hat{t} - \alpha\sigma_{\hat{t}}$.

PALM: The Coupling Device (http://www.cerfacs.fr/∼palm)

The PALM[2] project aims at implementing a general tool to easily integrate high performance computing applications in a flexible and evolutive way. It is originally designed for oceanographic data assimilation algorithms, but its domain of application extends to every kind of scientific application. In the framework of PALM, applications are split into elementary components that can exchange data. Its main features are: The dynamic launching of the coupled components, the full independence of the components from the application algorithm, the parallel data exchanges with redistribution and the separation of the physics from the algebraic manipulations (performed by the PALM algebra toolbox).
To generate a PALM application, the first necessary step requires the existing codes (call units) to be compatible with PALM. Modifications are rapidly achieved by inserting in the source codes some PALM instructions such as *Get* and *Put* which will allow to exchange data from one code to another. An interface or ID card of the code describing the exchanged data have to be written. Once all the components of the application are available, PrePALM, the PALM graphical preprocessor, is used to develop the structure of the algorithm: Order the launch of the components either concurrently or successively, in loops or conditionally.
PALM is able to handle many parallel codes as well as several instances of the same code. This particularity is very usefull in the context of optimization with CFD codes. Indeed, most of the codes we use are parallel, and the possibility to make many computations at the same time will compress the whole clock time needed to complete the set of simulations prescribed by an optimization application. Finally, PALM automatically manages the processor distribution for an application between the different units and based on a user-defined priority fashion.

Developed tool

An overview of the optimization tool is presented on Fig. 2. Each rectangular box with thick lines (call branchs in the PALM language) can run independently on different processors. They coincide with the first level of parallelism taken into account by PALM. The second level corresponds to the capacity of handling parallel codes as already underlined in the description of PALM. MPI communications necessary for the exchange of data between processes (*i.e.* meshes and fields) are depicted by arrows. The surrogate assisted optimization algorithm is briefly detailed on the flowchart of Fig. 1. One can note that the optimization unit is able to decide whether the true function needs to be evaluated trought the CFD code or if the approximation is satisfactory. An interface manages and launches the CFD computations in a "multi-run" way: It aims at automatically distribute the tasks depending on the available ressources. In the next section, this algorithm is compared to the Simplex optimization approach. Note that the flexibility of PALM allows quick modifications in the application, as the use of another optimization algorithm or CFD code. In the case of the Simplex PALM application, the Simplex branch exchanges optimization parameters and objective function values directly with the CFD branch.

Dealing exclusively with the CFD computations, the optimization parameters are first transformed by the so-called "Pre-processing" unit: It handles the mesh as far as design optimization is concerned as well as the boundary conditions when controlling the operating point. The automatic mesh control when directly parameterized by optimization parameters is a central problem. Two methods are conceivable:

- Given an initial shape and mesh, the new mesh is adapted to the shape defined by the design parameters by deforming the previous grid. Several techniques exit and one can cite Laplacian, spring[11] and explicit[12] methods. These approaches have the advantage of conserving the total number of nodes and mesh connectivity garrantying homogeneous MPI communications between units during the optimization process. Mesh deformation is however limited to small geometrical changes otherwise leading to negative cells and poor quality meshes.

- The second method consists in implementing a fully automatic mesh generator in the loop. The main difficulties concern the parameterization of the shape and the prohibitive computational time issued by the generation of a new mesh.

In the context of this study, both methods are used to construct $2D$ meshes. In particular, the implemented approaches are *Ipol* and *Delaundo*, the $1D$-shape as well as the $2D$-domain meshers developed by Müller.[13] It is important to note that moving nodes and remeshing techniques often need smoothing steps to enhance cell quality. Smoothers such as Laplacian-based and angle or aspect ratio optimization-based are available in the tool.

Finally, at the end of the CFD computations, flow solutions are post-treated in order to evaluate the optimization variables necessary for objective function calculations. At the moment, only single objective function problems are possible but extension to multi-objective calculations can be done by summing the corresponding fitness.

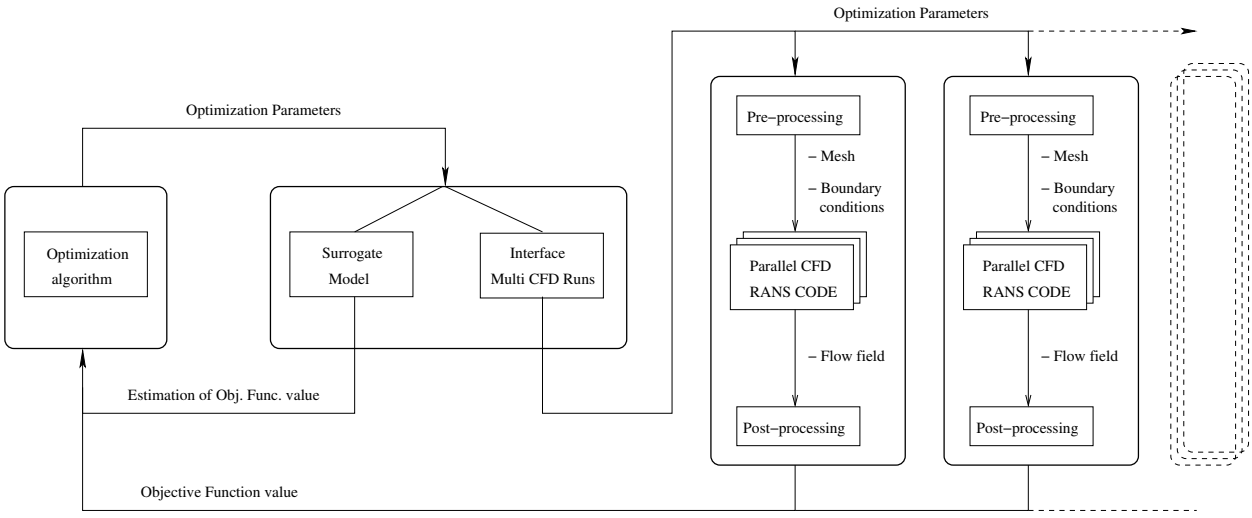

Fig. 2 Overview of the optimization tool

Numerical results

Presentation of the test case

The studied configuration consists in a 2D channel ($0.03m \times 0.25m$) in which hot gases flow ($T_h = 1500K$) while two dilution injectors aim at cooling the hot stream before it exits the pipe, Fig. 3. This configuration has the particularity of being representative of a cooling process found in the dilution region of a combustion chamber. The attempt of the presented computations is to find the optimal locations of the cooling injectors to reach a given output temperature profile. From the test case, it is expected that for large values of the positions of the dilution injectors, respectively noted L_{cu} and L_{cl}, the cooling will be less efficient due to poor mixing of the hot gas with the cold one. Likewise $L_{cu} \neq L_{cl}$ for the upper and lower injector positions should lead to assymetric exit temperature profile if the inflows are the same.

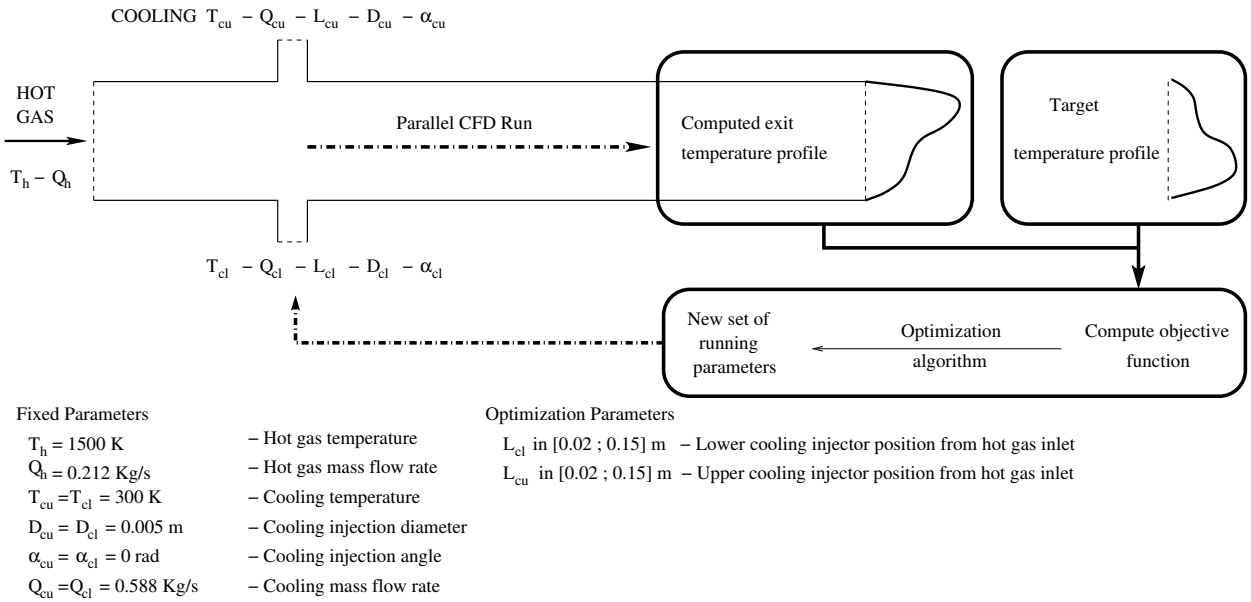

Fig. 3 Optimization configuration

The objective function is obtained by comparing the exit temperature profile computed by the CFD code and the target through the following expression:

$$f_{obj} = \frac{1}{D_h} \int \left(\frac{T^t(y) - T^c(y)}{T^t(y)} \right)^2 dy \tag{15}$$

where D_h is the diameter of the channel, T^t is the target temperature and T^c is the computed exit temperature. The expression of the objective function is a root mean square between target and computed temperature profiles. It is non-dimensionalized by the target temperature to yield relative importance to the standard RMS (note that the temperature goes from approximatively 300 K to 1500 K in the channel). Tests prove this objective function to yield better results than standard RMS in terms of solution quality and convergence rapidity (on this particular configuration). Further assessment of other objective function is needed for a better understanding of its impact on the solution. The target temperature profile comes from a CFD computation with $L_{cu} = 0.075\ m$ and $L_{cl} = 0.05\ m$ which stands for the global minimum of the search domain.

Since the shape deformations are important in the context of this study, remeshing techniques are adopted. The generated meshes have a mean of 9000 nodes and 17500 cells. CFD computations are obtained from N3S-Natur, a Reynolds Average Navier Stokes (RANS) solver developed by INCKA Simulog. Each CFD calculation takes about 15 minutes on 5 processors of the DEC ALPHA.

Optimization with the Simplex method

The Simplex algorithm belongs to the direct seach methods which are local optimizers. Simplex is easy to use and to implement but suffers from pre-mature convergence disease. The version proposed by Nelder and Mead[3] is used along with several initialisations in the search domain space, Fig. 6. Results are summarized in table 1 while Fig. 4 shows the evolution of the exit temperature profile during the runs $5, 6, 7$ and 8. One will note that the Simplex does not converge to the same optimum value even for closely localized initialisations. Runs $4, 5$ and 7 seem to yield the same point which turns out to be the global optimum ($L_{cu} = 0.075\ m$, $L_{cl} = 0.05\ m$). Figures. 4 confirms that 5

Run ID	Initialisation	Optimum	Number of CFD evaluations	Symbol in Fig. 6
1	$L_{cu} = 0.120\,m$ $L_{cl} = 0.030\,m$	$L_{cu} = 0.101\,m$ $L_{cl} = 0.045\,m$	37	$\times$
2	$L_{cu} = 0.030\,m$ $L_{cl} = 0.133\,m$	$L_{cu} = 0.030\,m$ $L_{cl} = 0.119\,m$	38	$\triangleleft$
3	$L_{cu} = 0.120\,m$ $L_{cl} = 0.133\,m$	$L_{cu} = 0.120\,m$ $L_{cl} = 0.107\,m$	35	$\diamond$
4	$L_{cu} = 0.103\,m$ $L_{cl} = 0.030\,m$	$L_{cu} = 0.070\,m$ $L_{cl} = 0.054\,m$	33	$\square$
5	$L_{cu} = 0.040\,m$ $L_{cl} = 0.073\,m$	$L_{cu} = 0.069\,m$ $L_{cl} = 0.037\,m$	40	$\star$
6	$L_{cu} = 0.080\,m$ $L_{cl} = 0.133\,m$	$L_{cu} = 0.104\,m$ $L_{cl} = 0.101\,m$	36	$+$
7	$L_{cu} = 0.030\,m$ $L_{cl} = 0.043\,m$	$L_{cu} = 0.071\,m$ $L_{cl} = 0.029\,m$	42	$\circ$
8	$L_{cu} = 0.058\,m$ $L_{cl} = 0.125\,m$	$L_{cu} = 0.037\,m$ $L_{cl} = 0.141\,m$	34	$\triangleright$

Table 1 Simplex results

and 7 solutions fit the target profile quite well. Runs 6 and 8 however result in convergened solutions which are local optima but where the obtained temperature profile does not match the target.
As a conclusion on the Simplex optimization, it is observed that this method requires a lot of CFD runs to converge to a local optimum. It is very sensitive to the level of noise present in CFD computations which are used to evaluate the fitness function. This noise increases pre-mature convergence of the algorithm as illustrated by run 6 where the optimum is not far from the target but does not seem to reach it.

Optimization with surrogate assisted algorithm

The initialization step of the surrogate based optimization process is designed so as to perform 20 objective function evaluations using the CFD code. Then, the improvement of the model is done thougth 15 iterations of the method (Fig. 1) requiring a total of 35 additional CFD computations. After this refinement phase of the surrogate, the LBFGS-B algorithm does not find any other local minimum for the surrogate model.
Figure 5 illustrates the impact of improving the model by comparing $\hat{t}$, $-\rho\sigma_{\hat{t}}$ and $f_M = \hat{t} - \rho\sigma_{\hat{t}}$ after initialisation and after 15 iterations of the algorithm. We first note that many points are computed on the bound of the variables. This is due to the fact that the model often gives poor predictions at these locations because their is an insufficient number of sample points outside the bounds of the search domain. Figure 5 (b) clearly illustrates the great improvement of the model through the important reduction of the $-\rho\sigma_{\hat{t}}$ within the search domain. As expected, the procedure is able to find two minima:

- a local one at $L_{cu} = 0.043\,m$, $L_{cl} = 0.15\,m$
- the global one at $L_{cu} = 0.0771\,m$, $L_{cl} = 0.0429\,m$

Finally, one will note the existence in the surface of a large valley coinciding with the attraction bassin of the global minimum. This valley is oriented along the straight line $L_{cl} = L_{cu} - 0.025$ which corresponds to the set of parameters leading the same degree of dissymetry in exit temperature profile as defined by the target temperature profile but with different mean exit temperatures. An important aspect of the surrogate approach is underlined here: The observation of the obtained map leads to a reduced number of decision parameters by applying proper constraints or by linking them with the objective.
To conclude on the surrogate optimization, this method gives accurate results with few CFD evaluations. It also provides a map of local extrema along with a clear vizualsation of the degree of correlation of the optimization variables on the fitness function.

Comparison between Simplex and surrogate method

Figure 6 shows the behaviour of the Simplex optimizations previously presented and fronted with the surface obtained from the surrogate based optimization (after 55 CFD computations). The chaotic behaviour of the Simplex convergence history linked to the drastic valley $L_{cl} = L_{cu} - 0.025$ is clearly illustrated. Indeed, this type of configuration often leads to convergence disease of the Simplex method. More specifically, the path followed by Run 8 tends to enforce the existence of the local minimum found by the surrogate approach as well as the stationnary point evidenced

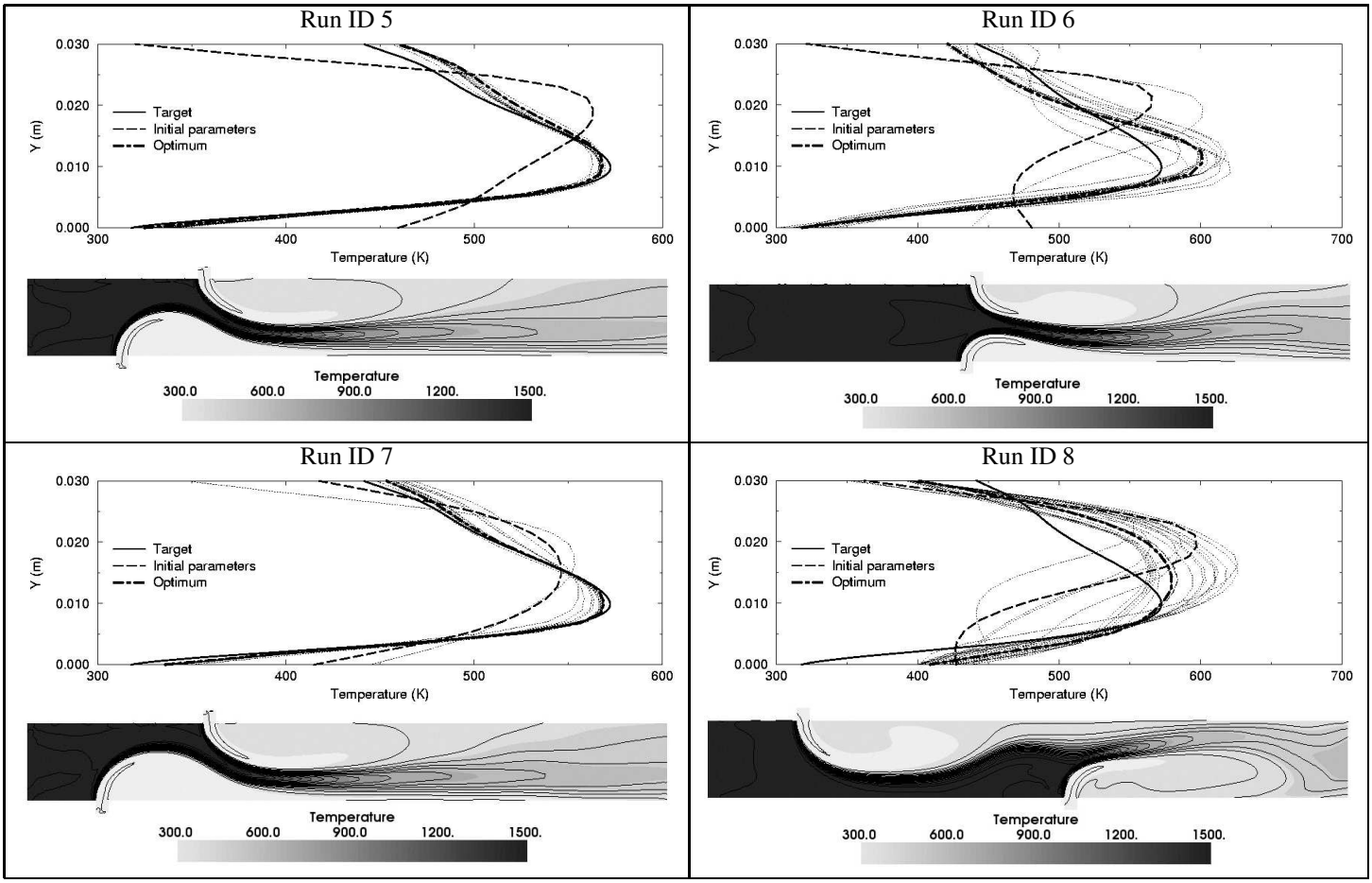

Fig. 4 Simplex results: Evolution of the exit temperature profile during run 5, 6, 7 and 8.

by Run 2. These two Simplex runs underline the importance of the initialisations for this approach since two very nearby initial guesses may not yield the same result.

Table 2 presents estimated CPU times as well as the whole clock time for a Simplex optimization (average of the 8 runs) and a surrogate based search. CPU times take into account all the involved devices including the PALM driver processor which maintains the MPI communications between units, the optimization branch and the so-called "interpretation branch" in the case of the surrogate approach. For this specific study, two CFD branches were used in parallel. Since only two optimization parameters are needed for the problem considered, the total number of CFD computations for convergence is not very important. Consequently, the surrogate evaluations obtained with the gaussian process has a negligible impact on the total cost.

Method	CPU time	Number of processors	Clock time
Simplex	3885 minutes	7	555 minutes
Surrogate	5360 minutes	13	412 minutes

Table 2 CPU times

Nonetheless, the metamodel assisted optimization procedure proves to give more accurate results than the Simplex approach with fewer evaluations of fitness values throught the expensive CFD code as global optimization is concerned. Moreover, and contrary to Simplex optimization, the general tendency of the fitness against optimization parameters can be directly observed on the results and global optima is garantied if existing in the pre-defined search domain.

Conclusions and perspectives

In this paper, we have presented a global optimization method for expensive fitness function based on an approximation issued from a Gaussian process. This model is constructed with an adaptive database which is improved during the optimization process. The proposed algorithm allows a reduced number of objective function evaluations reducing

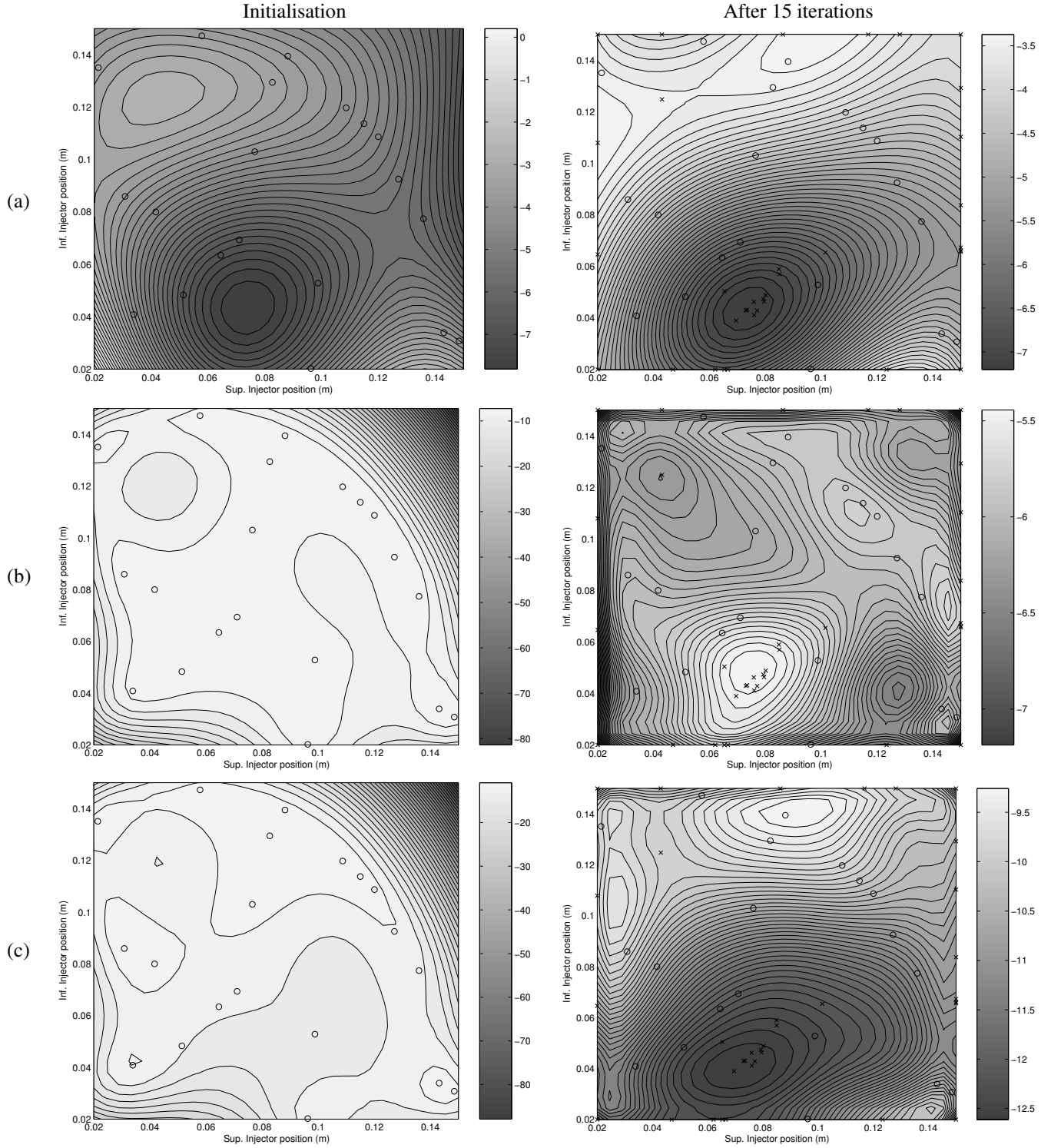

Fig. 5 Left, surrogate at the initialisation (20 CFD computations ∘) - Right, enrichement of the surrogate after 15 iterations (35 CFD computations ×). (a) $\hat{t}$ (b) $-\rho\sigma_{\hat{t}}$ (c) $f_M = \hat{t} - \rho\sigma_{\hat{t}}$

the total CPU time usually necessary with conventional approaches. Comparison between this method and the very popular Simplex algorithm leads to encouraging results and validates the choices for further developments.

Although the presented tool is effective, further improvements are possible. First, the way of coupling the surrogate model with the optimization procedure can be made better. Indeed, the enrichment process of the database has to be clarified when new points are found in the neighborhood of the known data or when no new point is located. Second, it is usefull to pursue the studies on the optimization algorithms and their interaction with surrogate modelling to

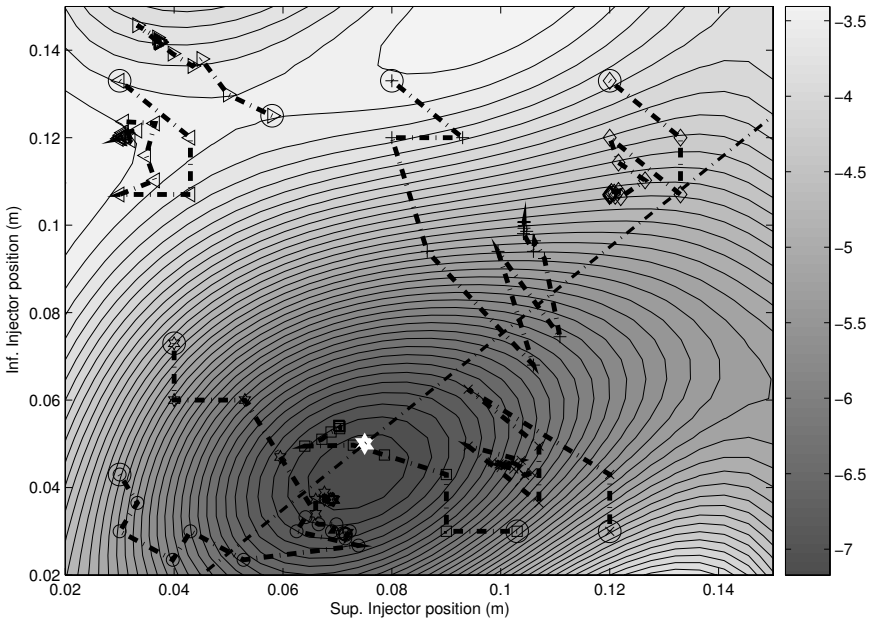

Fig. 6 Comparison between Simplex and surrogate method - Big circles: Starting points for the Simplex methods - Dotted line: $L_{cl} = L_{cu} - 0.025$ valley - White star: Global optimum $(L_{cu} = 0.075\ m\ ,\ L_{cl} = 0.05\ m)$

increase the accuracy of optimum prediction. Finally, an important part of the research concerns the possibility to undertake the optimization process with complex $3D$ geometry. This latter aspect imposes to control $3D$ shapes and the corresponding mesh using moving mesh technics or re-meshing strategies.

Acknowledgements

The authors thank D. Bissières and C. Bérat (Turbomeca, 64511 Bordes - France) for the support and the industrial context of the study, as well as the valued support from T. Morel and S. Buis (CERFACS, 31057 Toulouse - France) for their help concerning PALM. The contribution from the INCKA team is acknowledge when dealing with the N3S-Natur grid management tools.

References

[1] D.J.C. MacKay. Gaussian processes - a replacement for supervised neural networks? *Lecture notes for a tutorial at NIPS*, 1997.

[2] S. Buis, A. Piacentini, and D. Déclat. Palm: A computational framework for assembling high performance computing applications. *CONCURRENCY AND COMPUTATION: PRACTICE AND EXPERIENCE*, 2005.

[3] J.A. Nelder and R. Mead. A simplex method for function minimization. *The Computer Journal*, 7:308–313, 1965.

[4] S. Izquierdo. *Reducción de contaminantes en combustión aplicando técnicas de CFD y Algoritmos Genéticos*. PhD thesis, University of Zaragora, 2003.

[5] A.L. Marsden. *Aerodynamic Noise Control by Optimal Shape Design*. PhD thesis, Stanford University, 2004.

[6] T.W. Simpson, T.M. Mauery, J.J. Korte, and F. Misree. Comparison of response surface and kriging models for multidisciplinary optimization. *AIAA*, 98-4755, 1998.

[7] Y. Jin, M. Olhofer, and B. Sendhoff. On evolutionary optimisation with approximate fitness functions. In *Proceedings of the Genetic and Evolutionary Computation Conference GECCO*, pages 786–793, July 2000.

[8] Y.S. Ong, P.B. Nair, and A.J. Keane. Evolutionary optimization of computationally expensive problems via surrogate modeling. *AIAA*, 41(4):687–696, 2003.

[9] D. Büche, N.N. Schraudolph, and P. Koumoutsakos. Accelerating evolutionary algorithms with gaussian process fitness function models. *IEEE Transactions on Systems, Man and Cybernetics*, 35:183–194, 2005.

[10] Z. Michalewicz. *Genetic Algorithms + Data Structure = Evolution Programs*. Springer, 1996.

[11] C.O.E. Burg. A robust unstructured grid movement strategy using three-dimensional torsional spring. In *AIAA Paper 2004-2529*, Portland, Oregon, June, 2004. 34th AIAA Fluid Dynamics Conference.

[12] B. Mohammadi and O. Pironneau. *Applied Shape Optimization for Fluids*. Oxford Science Publications, 2001.

[13] J.D. Müller. *On Triangles and Flow*. PhD thesis, The University of Michigan, 1996.

[14] M. McKay, R. Beckman, and W. Conover. A comparison of three methods for selecting values of input variables in the analysis of output from a computer code. *Technometrics*, 21(2):239–245, 1979.

[15] R. H. Byrd, P. Lu, J. Nocedal, and C. Zhu. A limited memory algorithm for bound constrained optimization. *SIAM J. Scientific Computing*, 5(16):1190–1208, 1995.