INTERNATIONAL JOURNAL FOR NUMERICAL METHODS IN FLUIDS
Int. J. Numer. Meth. Fluids 2004; **44**:1257–1278 (DOI: 10.1002/fld.688)

Hybrid genetic algorithms and artificial neural networks for complex design optimization in CFD

R. Duvigneau and M. Visonneau*,†

Laboratoire de Mécanique des Fluides CNRS UMR 6598, Ecole Centrale de Nantes, 1, rue de la Noe 44321 Nantes, France

SUMMARY

The present paper is devoted to the study of design optimization strategies in the particular framework of complex computational fluid dynamics. Genetic algorithms are chosen as the optimization strategy, thanks to their robustness and flexibility. Two ways are explored to improve the behaviour of genetic algorithms in order to increase the efficiency of the search. First, approximated pre-evaluations based on artificial neural networks are used to benefit from the knowledge acquired from the problem and to reduce the number of expensive evaluations by the flow solver required at each generation. Then, a hybridization technique is proposed for the final local search, which is performed by a deterministic method. These approaches are validated and applied on two- and three-dimensional problems, involving Reynolds-averaged Navier–Stokes computations with near-wall turbulence modeling. Copyright © 2004 John Wiley & Sons, Ltd.

KEY WORDS: shape optimization; Navier–Stokes; genetic algorithms; artificial neural networks

1. INTRODUCTION

For several years, flow computations have been taken into account by engineers to improve the designs, for example in aerodynamics or hydrodynamics. Modifications were performed manually first, and then using automated tools to lead the search towards optimality. Thanks to the progress in computational fluid dynamics (CFD) and computers hardware during the last few years, the automated design optimization procedures are now expected to solve problems including complicated flows and realistic configurations. Consequently, the physical and geometrical configurations encountered in industrial applications make necessary the recourse to viscous flow solvers based on sophisticated turbulence modelling, dealing with realistic geometries. In this framework, the naive use of a standardized toolbox optimization software

*Correspondence to: M. Visonneau, Laboratoire de Mécanique des Fluides, CNRS UMR 6598, Ecole Centrale de Nantes, 1, Rue de la Noe, F-44321 Nantes, France.
†E-mail: michel.visonneau@ec-nantes.fr

Contract/grant sponsor: Scientific Committee of CINES; contract/grant number dmn 2050
Contract/grant sponsor: IDRIS; contract/grant number 1308

Received 10 April 2003
Revised 22 October 2003

connected to a flow solver and an automated grid generator cannot be a sensible strategy, since the peculiarities of the flow solver should be taken into account. Otherwise, some limitations may be quickly encountered. For instance, the constraints on the grid ($y^+ = O(1)$ near the wall) linked to the use of near-wall turbulence models will have serious consequences on the mesh update procedure. If a particular strategy taking into account the high stretching of the volumes near the wall is not included, the mesh update will fail. This observation explains why the recourse to an automatic grid generation software may not be wise in such a context. Another reason is related to the mandatory use of parallelization strategies, such as domain decomposition, as soon as three-dimensional problems are considered. The mesh update, as well as the parameterization of the shape, should be adapted to this multi-block partition to work within each block independently and to send information for updating the overall domain. Otherwise, the parallelization becomes useless. Concerning the optimization methods, the high computational costs implied by three-dimensional calculations, as well as the possible occurrence of numerical noise of various origins during the evaluations, should be taken into account for the choice of an optimization strategy. Finally, if differently connected software are employed in the design loop, some practical difficulties may arise, when distant computers are involved in the optimization process. All these remarks justify the development of an optimization procedure, in which all numerical tools are adapted to the flow solver and integrated into a single code.

In the past, considerable research efforts were focused on the development of techniques for evaluating the sensitivity of the cost function with respect to the shape, in order to use gradient-based optimization methods [1, 2]. In that way, the coupling between the optimizer and the flow solver is strong and a low number of evaluations is required to reach an optimal design. This approach was successful and cheap when rather simple flows were considered, but many limitations were noted when this approach was applied to more complex and realistic problems. First, the evaluation of the derivatives of the cost function with respect to the design variables is cumbersome when sophisticated flow solvers and highly non-linear turbulence models are considered [3]. Then, the presence of a numerical noise related to the complexity of the flow was reported [4–6], which generates spurious local minima and inhibits the capabilities of gradient-based strategies. Moreover, using such methods, a local optimization is performed, involving only one criterion, which is not satisfactory in an industrial framework. Lastly, one may think that the difficulty in evaluating the derivatives when different physical fields of applications are coupled, makes quite unlikely the development of gradient-based multi-disciplinary optimization strategies.

To overcome these limitations, some authors proposed the employment of more powerful optimization strategies, such as genetic algorithms (GAs). These stochastic methods [7] are known for their robustness, even when the cost function is noisy or discontinuous, and their ability to perform global optimization. Moreover, they have the capability to solve multi-criteria problems. However, GAs do not use derivative information to lead the search. Therefore, a weak coupling between the optimizer and the flow solver is observed, yielding an expensive strategy which requires a high number of evaluations. If this approach was successful for two-dimensional inviscid flows, its use for three-dimensional viscous problems cannot be advised when industrial applications are considered.

Therefore, the present paper is devoted to the study of acceleration techniques for GAs, in the particular framework of complex CFD applications. Two approaches are considered here. First, artificial neural networks (ANNs) are employed to benefit from the flow com-

putations performed during the design process and to provide more informations to the GA in order to prevent it from performing numerous evaluations. Typically, ANNs are expected to approximate the cost function in a pre-evaluation procedure in order to reduce the number of expensive evaluations by the flow solver required by GAs. Then, a second way is explored, which concerns the use of a deterministic method to perform quickly the final local search in order to remedy the poor local convergence properties of GAs. Such a hybrid methodology is expected to provide successively an efficient global and local search, taking the best part of both methods without losing their fundamental advantages. A similar approach was reported in Reference [8] employing classical GAs first, and then a conjugate gradient method, the derivatives being evaluated using ANNs approximations.

These approaches are validated and tested on two- and three-dimensional problems with the help of the ISIS flow solver developed in our laboratory. This flow solver deals with the incompressible Reynolds-averaged Navier–Stokes (RANS) equations on unstructured grids, the modelization of the turbulence being achieved by near-wall low- Reynolds number models.

The shape of a two-dimensional airfoil is first optimized to validate the above-mentioned procedures. Then, the three-dimensional design of a wing is considered, to show the capability of the resulting approach to deal with real-life problems.

2. FLOW SOLVER

The multi-purpose flow solver ISIS, developed in our laboratory, is implemented in the optimization procedure, providing the capability to solve realistic problems. Pseudo-steady incompressible RANS equations are solved with a strongly conservative formulation. For each control volume, the momentum and mass conservation laws are written as

$$\frac{\partial}{\partial \tau}\int_V \rho U_i \,\mathrm{d}V + \int_S \rho U_i(\boldsymbol{U}-\boldsymbol{U}_m).\boldsymbol{n}\,\mathrm{d}S = \int_S (T_{ij}I_j - pI_i).\boldsymbol{n}\,\mathrm{d}S + \int_V \rho g_i \,\mathrm{d}V \quad i=1,2,3 \tag{1}$$

$$\int_S \boldsymbol{U}.\boldsymbol{n}\,\mathrm{d}S = 0 \tag{2}$$

$\boldsymbol{U}_m$ representing the velocity of the moving faces S of the control volume V, T_{ij} the stress tensor, g_i are the components of the gravity force and $\boldsymbol{n}$ is the normal to the control volume faces. I_j is a vector whose components vanish, except for the component j which is equal to unity. The discretization scheme is based on a finite-volume method, generalized to unstructured meshes composed of arbitrary control volume shapes. The flow variables are stored at the centre of the control volumes; surface and volume integrals being evaluated by second-order accurate approximations. The pressure–velocity coupling is performed by a SIMPLE-like algorithm. Several near-wall low- Reynolds number turbulence models, ranging from one-equation Spalart–Allmaras [9] model, two-equation k–ω closures [10], to a full Reynolds stress transport R_{ij}–ω model [11] are implemented in the flow solver.

3. PARAMETERIZATION AND MESH UPDATE

A mesh deformation tool is developed in order to update the mesh during the design process, since the grid should move in accordance with the boundaries deformation. In order to treat realistic geometries, hybrid grids are usually employed, with a structured mesh near the wall boundaries and an unstructured mesh in the outer domain where lower gradients are usually observed. Because of the complexity of such grids, the generation of each new mesh cannot be performed automatically from scratch by a standard software, particularly when a multi-block decomposition is considered. Consequently, a deformation of the initial grid is preferred to generate a new mesh in accordance with the current shape.

For the two-dimensional cases, a linear and torsional spring analogy is employed to control the deformation of the grid [12]. The parameterization is realized by a B-spline curve, ensuring the smoothness of the shapes and reducing the number of the design variables. The linear spring analogy is commonly employed to deform the mesh in design strategies. A linear spring is attached along each edge connecting two nodes i and j, with a stiffness inversely proportional to the square of the length l_{ij} of the edge:

$$k_{ij} \propto \frac{1}{l_{ij}^2} \tag{3}$$

The displacement of the nodes q is the solution of a linear system, which represents the quasi-static equilibrium of the discrete mechanical problem

$$K_{\text{lin}} q = 0, \quad q = \bar{q} \quad \text{on boundaries} \tag{4}$$

where K_{lin} is the stiffness matrix and $\bar{q}$ the displacement of the nodes imposed on the boundaries. This method prevents two nodes from colliding, since in that case the stiffness tends towards an infinite value according to (3), but it does not prevent a node from colliding with the edge that faces it. This behaviour is particularly damaging for viscous calculations, since the high stretching of the control volumes is unavoidable in the boundary layer and may lead to the failure of the deformation process by generating negative volumes. Therefore, special treatments are usually applied near the wall, involving geometric considerations [1]. In this paper, another approach which associates a torsional spring analogy with a linear one [12] is employed. The method consists in attaching to each node i of each volume a torsional spring with a stiffness defined by

$$C_i^{ijk} \propto \frac{1}{\sin^2(\theta_i^{ijk})} \tag{5}$$

where θ_i^{ijk} is the angle $j\hat{i}k$ at the node i of the volume. System (4) is then updated in

$$(K_{\text{lin}} + K_{\text{tors}}) q = 0, \quad q = \bar{q} \quad \text{on boundaries} \tag{6}$$

using an energy equivalence to express the effects of the torsional springs in terms of displacements of the nodes. The torsional springs prevent the volumes from interpenetrating each other. The association of linear and torsional springs provides a powerful mesh deformation tool, which maintains the grid quality near the wall even for high deformations.

For three-dimensional configurations, the implementation of the previous strategy is tedious in practice, because it is difficult to define precisely the characteristics of the torsional springs.

Therefore, a free form deformation (FFD) technique [13] is used to control the shape perturbations during the design process. It consists in first embedding the object to be deformed in a box, then modifying the metrics of the space in the box and the object inside, by deforming the box, rather than modifying the object itself. In this way, the shape of the object can be modified without even identifying its nature. Actually, a local co-ordinate system is attached on a parallelepipedic volume including the object to be deformed. The co-ordinates of any point P in this reference system are

$$P = \xi \mathbf{e_1} + \eta \mathbf{e_2} + \zeta \mathbf{e_3} \tag{7}$$

where $\mathbf{e_1}$, $\mathbf{e_2}$ and $\mathbf{e_3}$ are vectors of unit length, parallel to the leading directions of the parallelepipedic volume. A mapping function is then applied to the space inside the volume, by the use of a trivariate B-spline product. The new position P_{FFD} of the previous point P is provided by the algebraic relation

$$P_{\mathrm{FFD}} = \sum_{i,j,k} N_i(\xi) N_j(\eta) N_k(\zeta) P_{\mathrm{BS}}^{ijk} \tag{8}$$

where N_i, N_j and N_k are the B-spline basis functions and P_{BS}^{ijk} are the cartesian co-ordinates of the control points. By moving some of these control points, the space inside the parallelepipedic volume is perturbed, yielding a smooth deformation of the shape inside. This approach provides an easy-to-use manipulation tool, since no curvilinear representation of the object is required.

4. CLASSICAL GENETIC ALGORITHMS

GAs belong to the evolutionary methods of optimization, which mimic the natural laws of evolution to improve the performance of a set of individuals. They are based on the Darwin principle of 'the survival of the fittest'. For the present approach, classical GA methods are implemented [7]. Once a binary encoding representation of the design variables is chosen, a population of n_{ind} individuals is randomly generated. The performance of the whole population is evaluated, requiring n_{ind} simulations. Then, the population is evoluting through genetic operators. For the selection operator, whose purpose is to eliminate the worst individuals, a tournament strategy is chosen, since its pressure selection is lower than the classic roulette-wheel operator which may cause premature convergence. In practice, it consists in picking randomly two individuals and comparing their fitness. A copy of the best individual is selected for the new population, while both individuals are put back in the current population. The process is repeated n_{ind} times to keep unchanged the population size. Then, the goal of the crossover operator is to exploit and develop the capacity of the new population by exchanging some characteristics between some individuals. The individuals are mated randomly and a one-point crossover is performed, with a probability $p_{\mathrm{c}} = o(1)$, by swapping a part of the strings of two parents to give two children. During this operation, new individuals are generated, but only based on existing strings. Finally, the mutation operator is applied to increase the diversity by permuting the value of some bits, with the probability $p_{\mathrm{m}} \ll o(1)$. Thus, new individuals are created by exploring the design space. The process goes on evaluating the performance of the new generation of individuals.

5. PRE-EVALUATIONS USING ARTIFICIAL NEURAL NETWORKS

5.1. Pre-evaluations process

The GAs described below are known for their robustness and their ability to perform a global optimization. Moreover, they are less sensitive to the numerical noise than gradient-based approaches. However, the search performed is not really efficient, because n_{ind} design points should be evaluated at each generation regardless the informations already acquired on the functional. Moreover, among these new trial points, some low fitted individuals will be eliminated as early as the selection step and their expensive evaluation by the flow solver may be considered as useless. Furthermore, it should be underlined that accurate evaluations are not required by GAs, which only need to know the ranging of individuals.

Considering all these remarks, a pre-evaluation procedure based on an approximated model of the cost function is introduced at each generation. By replacing some expensive evaluations performed by the flow solver by cheap approximated evaluations, the knowledge acquired from the problem is used to reduce the computational costs and some useless accurate evaluations are avoided, as proposed by Giotis and Giannakoglou [14, 15].

During the first generations, a classical GA is running, all individuals evaluated at each generation being stored in a database. Then, from the generation k_{pe} to the end of the optimization process, one modifies the evaluation step at the beginning of each generation in the following way. The n_{ind} individuals of the current population are first pre-evaluated using an approximated functional. This approximation is built thanks to the database entries. These evaluations are called inexact pre-evaluations and provide a good estimate of the fitness of the individuals. Then, some individuals are chosen in the population to be evaluated by the flow solver. Since the high fitted individuals will be probably selected to generate the offsprings, one would like to know more accurately their fitness. Thus, the σn_{ind} most interesting individuals, according to the inexact pre-evaluations, are exactly evaluated by the flow solver. These evaluations are included in the database, updating it in interesting areas. Using this update strategy, the database is continuously updated during the optimization process and the inexact pre-evaluations become increasingly accurate, since the database entries are concentrated in the most promising areas, as most new optimization points. Once the evaluations are done, the genetic operators act as usual, according to the exact, if provided, or inexact evaluations of the individuals. The algorithm is summarized in the following lines:

1. Initialization
2. Evaluation of the generation k:
   if $k < k_{\text{pe}}$ then
   - n_{ind} evaluations by the flow solver
   - storage of n_{ind} points in the database

   if $k \geqslant k_{\text{pe}}$ then
   - n_{ind} inexact pre-evaluations according to the database
   - choice of the σn_{ind} best individuals according to the pre-evaluations
   - σn_{ind} evaluations by the flow solver
   - storage of σn_{ind} points in the database
3. Selection
4. Crossover

5. Mutation
6. $k \leftarrow k+1$ go to step 2

One must underline that this strategy is totally different from the method consisting in building first a global approximation of the cost function according to some evaluations, and then using only this approximation to perform the optimization cycles. Actually, in such an approach the flow solver is definitely forgotten after the construction of the approximation and numerous *a priori* evaluations must be performed in the whole design space to obtain an accurate global approximation. For the present strategy, a strong link is maintained between the flow solver, the representation process and the optimizer. Thanks to this coupling, evaluations are performed only in promising areas of the design space, which leads to a continuously updated and more accurate representation.

5.2. *Artificial neural networks*

The representation of the cost function used in the pre-evaluation process should be able to build a non-linear functional employing a low number of already known points. This is the reason why ANNs are preferred to classical polynomial representations. Different structures of ANNs exist and may be considered for these applications. The most commonly used ANNs are the multilayer perceptrons (MPs). They are composed of some layers of some neurones, the first layer having as many neurones as the number of design variables. The number of neurones on the last layer is equal to the number of objective functions. All the neurones are connected to the neurones of the previous and next layers. To evaluate the output value of MPs at the point $x \in \Re^n$, the design variables are considered as input signal, which is propagating through the network according to the connections ω_{ij} between each couple of neurones N_i and N_j, up to the output layer. The MPs are characterized by the connections ω_{ij}. Consequently, a first step, called training, consists in determining adequate values for these connections. For the MPs, the training is performed by minimizing the evaluation errors of the ANN for the known points of the database entries. This stage is crucial and sometimes tedious, since this optimization exercise involves a large number of variables. It is usually performed through gradient-based methods and may suffer from local minimization, resulting in a weak training which provides a poor approximation. Moreover, the results may depend strongly on the number of layers and neurones, which is a real disadvantage for the present application, since the training should be performed automatically.

Therefore, the radial basis functions (RBFs) networks, which are in fact particular MPs, are adopted. They are constituted by only three layers, the intermediate layer counting as many neurones as the number of database entries n_{ex}. A representation is given in Figure 1. For each neurone N_i of this layer, a vector C_i, called the RBFs centre, is associated and has the value of the ith database entry

$$C_i = x^i, \quad i = 1, \ldots, n_{\text{ex}} \tag{9}$$

It is necessary now to evaluate the ANNs output value for the input signal x. For each neurone N_i of the intermediate layer, the input signal is the distance between x and the RBFs centre C_i:

$$E_i = \|x - C_i\|_2 \tag{10}$$

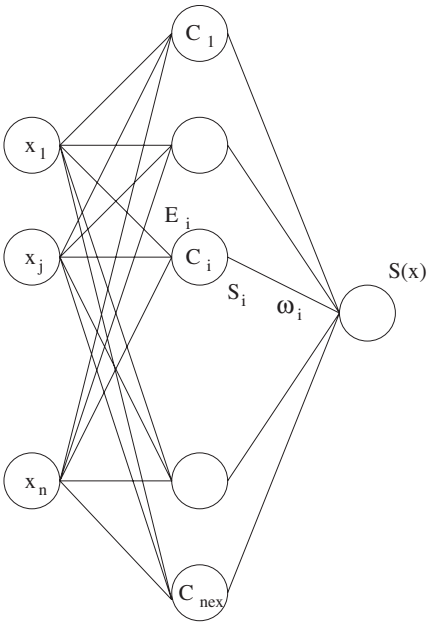

Figure 1. Structure of an RBF network.

The output value of the neurone is a non-linear function of the input signal given by

$$S_i = f_{\mathrm{RBF}}(E_i, e) \tag{11}$$

where e is an attenuation coefficient. Several functions may be considered [16]. For this study, the following expression is used:

$$f_{\mathrm{RBF}}(u, r) = \exp\left(-\left(\frac{u}{e}\right)^2\right) \tag{12}$$

The attenuation coefficient rules the domain of influence of each radial basis function. Finally, the ANNs output value is obtained through the linear relation

$$S(x) = \sum_{i=1}^{n_{\mathrm{ex}}} \omega_i S_i \tag{13}$$

where ω_i is the value of the connection between the neurone N_i of the intermediate layer and the output layer.

For the training of the RBFs network, only the connections ω_i between the intermediate layer and the output layer are taken in consideration. An interpolating condition for the points of the database, whose values are y^t, $t = 1, \ldots, n_{\mathrm{ex}}$, is written by

$$y^t = \sum_{i=1}^{n_{\mathrm{ex}}} \omega_i S_i, \quad t = 1, \ldots, n_{\mathrm{ex}} \tag{14}$$

Consequently, the training just consists in solving the previous linear interpolating system to determine the adequate weights ω_i. Contrary to classical MPs, no weak training is expected.

It could be shown that this system is invertible for several functions f_{RBF}, providing that the entries of the database are distinct.

5.3. *Validation*

Before checking the capabilities of the current approach to accelerate the optimization calculations, some exercises are performed in order to validate the method and determine the best set of parameters. Indeed, some questions still remain open. First, the size of the database used to train the ANNs should be determined. Actually, the whole database may not be useful during the training process, since the furthest points of the database may have a bad influence when evaluating a precise individual. Therefore, local databases may be employed to train as many ANNs as the number of individuals. The number of entries included in the local databases should be also determined. If it is too large, useless points may be included. If it is too narrow, the approximation may be poor. Then, the second parameter which has to be determined is the number of generations, for which all evaluations are performed exactly at the beginning. If the pre-evaluation procedure begins too early, the database will be too sparse in order ANNs to provide accurate enough informations. Finally, an adequate coefficient of attenuation should be determined. The sensitivity of the method with respect to these three parameters has to be explored.

The design optimization of an airfoil is considered as exercise to give answers to the previous questions. The goal is the increase of the lift at the incidence of 5° for a Reynolds number $Re = 10^6$. A hybrid mesh of about 7000 nodes is employed, with a structured grid of 200×21 nodes around the airfoil and a triangular grid further, initially built around the NACA 0012 airfoil. The turbulence modelling is achieved by the near-wall Spalart–Allmaras model. The shape is parameterized by two Bezier curves, six control points being moved vertically during the optimization. For the genetic algorithm, the crossover and mutation probabilities are $p_c = 0.95$ and $p_m = 0.1$, the population counts 20 individuals with a coding length of 36. High probability values are used to promote the creation of new individuals, since the population size is quite small. First, the influence of the size of the local databases is studied for different attenuation coefficients. For this purpose, one supposes that until generation 5, all the evaluations are performed exactly by the flow solver and stored in the database. Then, for all the individuals of the sixth generation, a comparison between the inexact pre-evaluations and the exact evaluations is done. For each individual, an ANN is trained using a local database, including the five to 100 nearest entries, the last case corresponding actually to the full database. For each case, different values of the attenuation coefficient are also tested. Figures 2 and 3 show the average and maximum error for these tests. One can observe that neural approximations based only on five entries are poor. An increase in the number of database entries for the training leads to a reduction of the error. When the attenuation coefficient is too high and a large local database is used, the ANNs approximations become very bad. This phenomenon is due to the presence of furthest points in the local databases, whose influence is not neglected because of the high attenuation value. When the attenuation coefficient is low, the furthest points have less influence on the results and large local databases may be used with good results. However, when it is too low, some design points are poorly fitted by ANNs, since they are too far away from points in the local databases, particularly when the databases count a small number of entries. Finally, one can see that the results obtained using 10–20 entries are similar and satisfactory for a large domain of attenuation

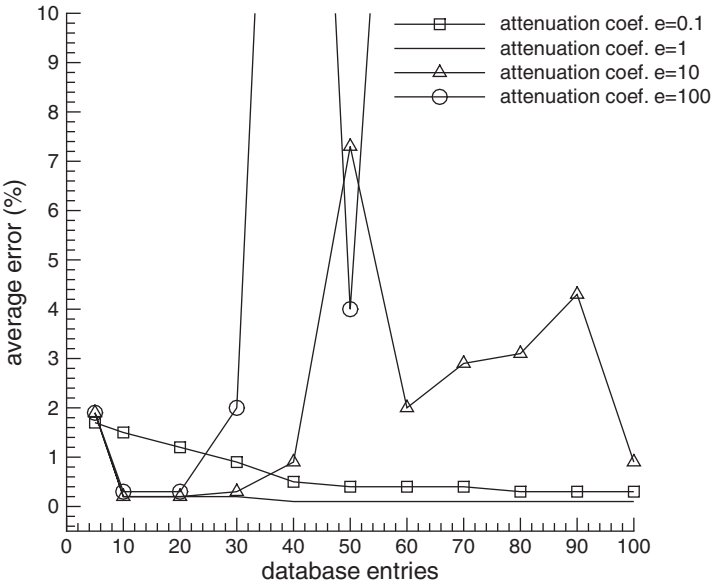

Figure 2. Average error versus the number of database entries.

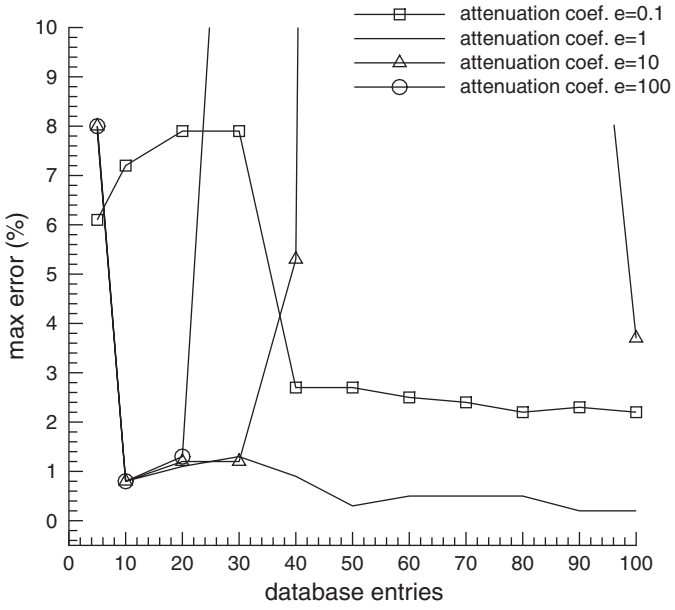

Figure 3. Max error versus the number of database entries.

values, which may be easily found by some a priori experiments. This conclusion is in agreement with Giannakoglou and Giotis' results [15], which are usually advising the use of such small local databases.

To study the influence of the last parameter, the same exercise is performed, including 10 entries in the local databases for the training and an attenuation coefficient $e = 1$. The choice of these values are justified by the previous studies. The comparison between the inexact evaluations and the evaluations by the flow solver are carried out for different generations. For all cases, it is supposed that the individuals at the previous generations were exactly evaluated. The results are presented in Figure 4. From the third generation, quite accurate neural approximations are observed, since the average error is less than 1 per cent. Even if the maximum error is higher, ANNs are able to provide the correct hierarchy between the individuals. This is illustrated by Figure 5 which shows a comparison of the cost function and ANNs values for all individuals of the fifth generation. For this generation, some individuals are not accurately evaluated by ANNs, but the hierarchy of individuals in the population is provided satisfactorily. Because of the role of the ANNs in the pre-evaluation process, this is only the hierarchy which is expected from ANNs and not an accurate evaluation. For the next generations, the average error becomes lower and lower, thanks to the update of the database, even when the gaps between the individuals are reduced. Sometimes, the error is larger for one individual, since it is far away from the database entries due to the random mutation operator. Finally, this satisfactory behaviour makes possible an early use of this pre-evaluation procedure in the optimization process, which leads to a strong reduction in the number of exact evaluations.

5.4. Application to two-dimensional airfoil optimization

This acceleration strategy is finally applied to the optimization of a two-dimensional airfoil in order to study the influence of the population rate σ exactly evaluated at each generation by the flow solver.

The optimization problem consists in increasing the lift coefficient Cl, for a Reynolds number $Re = 10^6$ and an incidence of 5°. Moreover, the drag coefficient Cd should not exceed a reference drag coefficient $\mathrm{Cd}_{\mathrm{ref}}$ computed for the NACA 0012. Thus, the cost function to be maximized is

$$f = \mathrm{Cl} - \max(O, \mathrm{Cd} - \mathrm{Cd}_{\mathrm{ref}}) \tag{15}$$

The mesh and the turbulence model employed are kept unchanged with respect to the previous computations. The shape is parameterized by two Bezier curves, whose 10 control points are moved vertically during the design process. For the GAs, a population of 30 individuals evoluting during 30 generations is used, with elitism. The probability of crossover and mutation are, respectively, $p_{\mathrm{c}} = 0.95$ and $p_{\mathrm{m}} = 0.02$, for a coding length of 76. The inexact pre-evaluations procedure begins as early as the second generation. Ten entries are employed to train the local ANNs, with an attenuation factor of value 10.

A first calculation is performed for which the whole population is exactly evaluated, corresponding to the case $\sigma = 1$. Then, a decreasing sequence of values for σ is tested. In Figure 6, a comparison of the evolutions of the cost function for the best individual of each generation with respect to the generation number is shown. One may observe that all computations reach the same level, except the case $\sigma = 0.03$, for which the number of exact evaluations per generation is too low to ensure an adequate update of the database. In this case, the ANNs pre-evaluations are not accurate enough to provide useful informations to

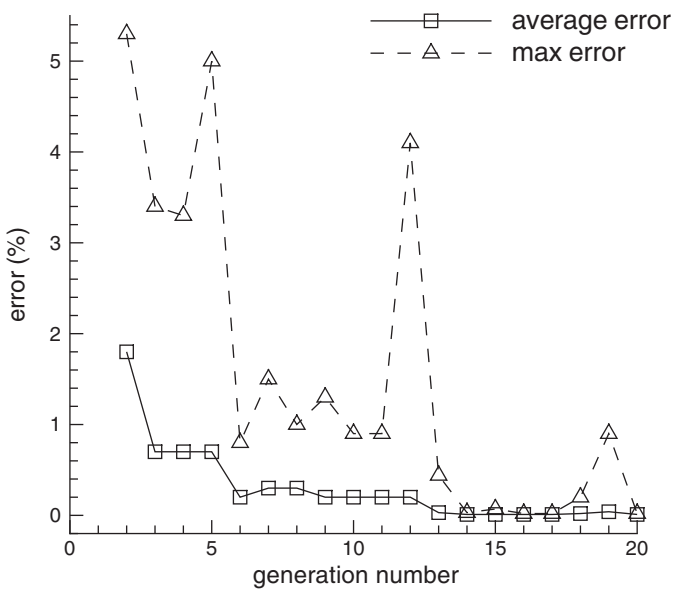

Figure 4. Error versus the generation number.

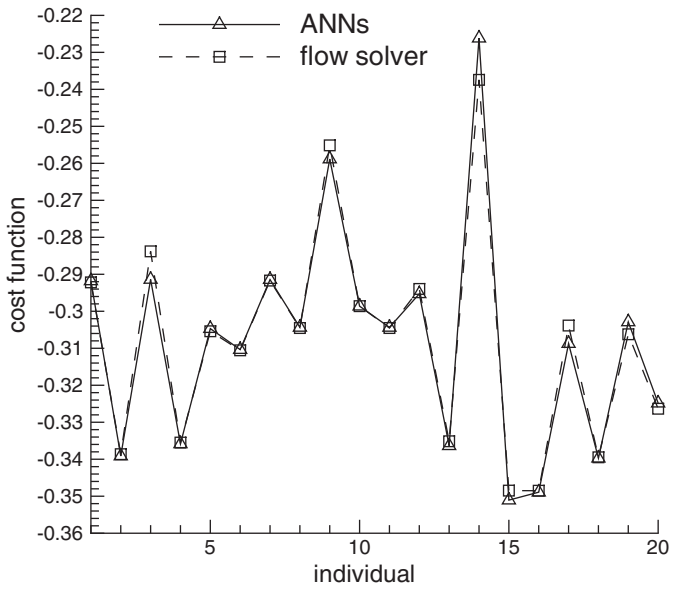

Figure 5. Error versus the individuals at generation 5.

GAs. Some oscillations may be noticed for the cases $\sigma = 0.03$ and 0.1, underlining the poor pre-evaluations provided to GAs. However, for the last case, GAs are robust enough to find the best combinations of genes, even when poor informations are sometimes used. The

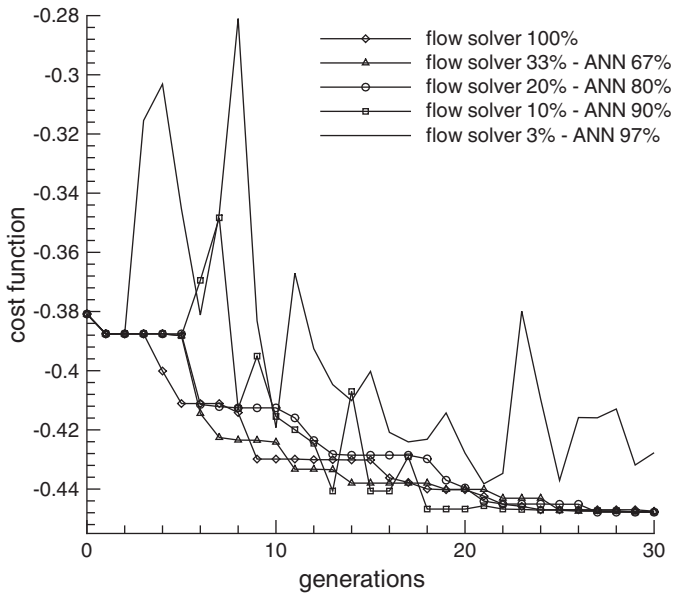

Figure 6. Evolution of the cost function versus the generation number.

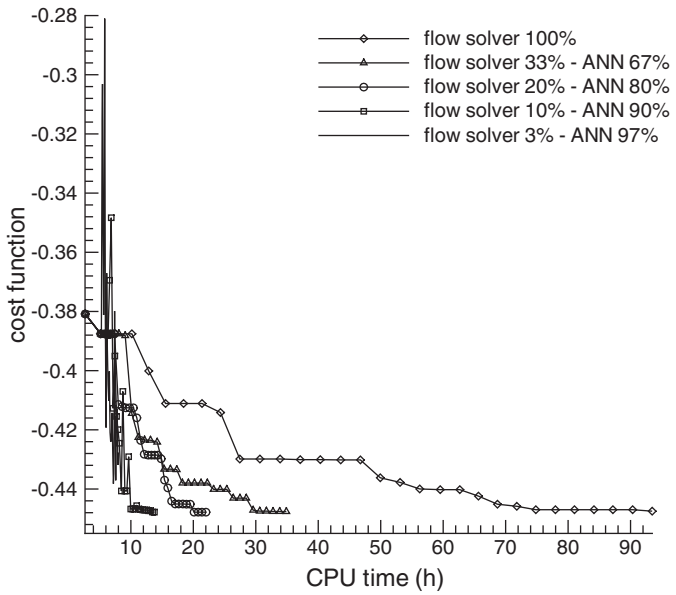

Figure 7. Evolution of the cost function versus CPU time.

acceleration effect is depicted in Figure 7, which provides the evolution of the cost function with respect to CPU time. The computations are performed using one SGI R10000 processor. For an advisable value $\sigma = 0.2$, a reduction ratio of value 5 for the CPU time is obtained.

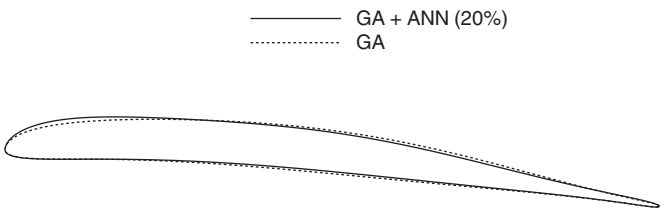

Figure 8. Final shapes for $\sigma = 1$ and 0.2.

The final shapes for the cases $\sigma = 1$ and 0.2 are shown in Figure 8. One may observe that they are very close to each other, although small gaps exist, due to a lack of local search. It should be underlined that the shape found by using inexact pre-evaluations is slightly better than the one determined with exact evaluations. Therefore, no lack of efficiency is observed.

6. HYBRIDIZATION

6.1. Method

Although the previous approach does not modify the nature of GAs, one may try now to substitute a more efficient algorithm to GAs for the final local search. Indeed, GAs have a poor local convergence rate, although they have the capability to determine quickly the most interesting areas of the design space. On the contrary, deterministic methods perform only local optimizations, but have a higher local convergence rate. Therefore, the hybrid method consists in using GAs to determine a good starting point for a deterministic algorithm.

In a first step, GAs are running as usually, during several generations. The previous acceleration approach may be used during this step, as well. A criterion is introduced to evaluate when the most interesting area is determined and GAs should be stopped. This criterion has a crucial influence. If it is too strict, too many GAs calculations are performed and no real acceleration effect is observed. If it is too tolerant, the starting point of the deterministic method may be too far away from the absolute minimum and may yield only a local optimum. In practice, GAs are stopped when the population rate σ_{hyb} is included in a sphere of radius ρ_{hyb} centred on the best individual.

Then, the second step begins by introducing a deterministic approach. A derivative-free trust-region method based on linear or quadratic interpolation [17] is used. A set of points is taken from the database to build a first interpolation model. Then, at each iteration, the furthest point from the best one is replaced by the minimum of the model found in the trust region, updating consequently the model. Only few iterations are expected to find the minimum of the cost function.

6.2. Application to two-dimensional airfoil optimization

The previous optimization exercise, including inexact pre-evaluations based on $\sigma = 0.2$, is chosen to validate the hybridization technique and to determine the influence of the

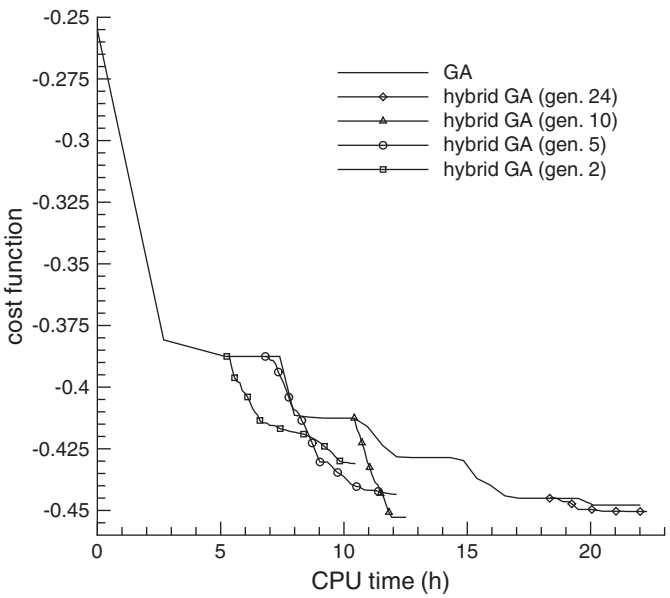

Figure 9. Evolution of the cost function for the hybridization.

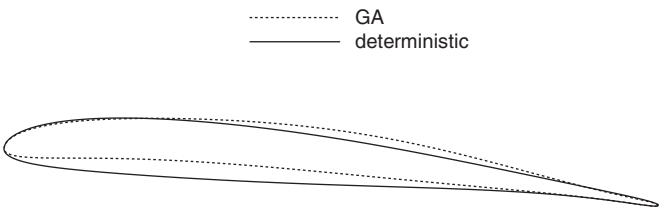

Figure 10. Final shapes for GAs and deterministic approaches.

stopping criterion. The population rate is fixed to $\sigma_{\text{hyb}} = 0.66$, since some individuals are always far away from the remaining population due to the random mutation operator. Four radii are tested $\rho_{\text{hyb}} = 0.1875$, 0.225, 0.375 and 0.625, these dimensionless values being related to the largest variation domain of the design variables. Using these parameters, GAs are stopped, respectively, at generations 24, 10, 5 and 2. The evolution of the cost function is presented in Figure 9. As expected, the choice of the radius is crucial. For an adequate value $\rho_{\text{hyb}} = 0.225$, the computational time is divided by 2. If the CPU time for classical GAs is considered as reference, the use of inexact pre-evaluations and hybridization provides a reduction ratio of 10. However, when the stopping criterion is too strict, a poor acceleration is observed, even if the performance of the final shape is slightly improved. When it is too tolerant, the minimum is not reached in a few iterations. When looking at the final shapes, one may underline that the deterministic method starting from the NACA 0012 airfoil does not reach the optimal shape (Figure 10). The shape obtained

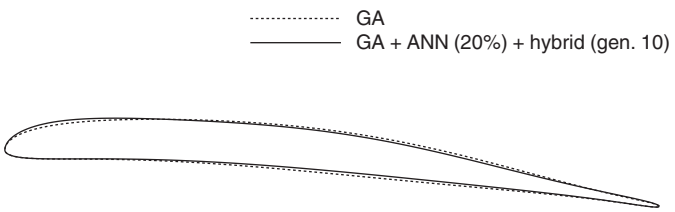

Figure 11. Final shapes for GAs and hybrid method.

with the inexact pre-evaluations and hybridization technique (Figure 11) is very close to the one found with only ANNs acceleration (Figure 8), its performance being slightly better thanks to a more efficient local search. Therefore, a real synergy between the algorithms is demonstrated.

The hybridization approach yields interesting results, but the determination of the stopping criterion seems to be crucial as well as dependent on the problem. Therefore, its use as acceleration technique looks tricky in practice. Nevertheless, this strategy seems to provide a way for refining the shape determined by GAs and we recommend its use in this spirit.

7. APPLICATION TO THREE-DIMENSIONAL DESIGN OPTIMIZATION

Finally, the techniques presented previously are used to perform the optimization of a three-dimensional wing. The airfoil considered is an extruded NACA 0015 airfoil with a span of 3.3, with a rounded salmon, at an incidence of 8°. The Reynolds number is $Re = 2 \times 10^6$, the turbulence modelling being achieved by the near-wall SST k–ω model of Menter. An unstructured mesh is employed, the airfoil skin being mapped with about 14 000 triangles (Figure 12). This skin grid is extruded using prismatic volumes to fulfill the criterion $y^+ = 1$ for viscous flows, tetrahedral volumes being used far away from the wall. The whole mesh is comprised of about 800 000 volumes. The goal of the optimization is to reduce the intensity of the vortex generated at the salmon by modifying only the airfoil shape close to this location. More precisely, the maximum of the vorticity modulus $|\Omega|$ in the half space ($\mathscr{D}$) downwind the wing is chosen as optimization criterion to be minimized:

$$f = \max_{X \in \mathscr{D}}(|\Omega(X)|) \tag{16}$$

This cost function is obviously more complex than lift or drag, used as cost function for the validations, and one may fear that ANNs accuracy will decrease in such a context. However, ANNs are only used for ranging purpose during the pre-evaluations, contrary to the method presented in Reference [8], where they are employed to evaluate sensitivities. Therefore, highly accurate ANNs approximations are not necessary in the present approach and one hopes that the overall procedure remains robust, even when the cost function becomes highly non-linear.

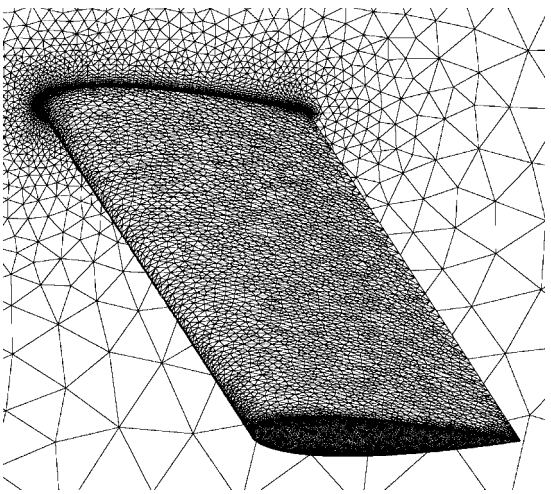

Figure 12. View of the skin mesh.

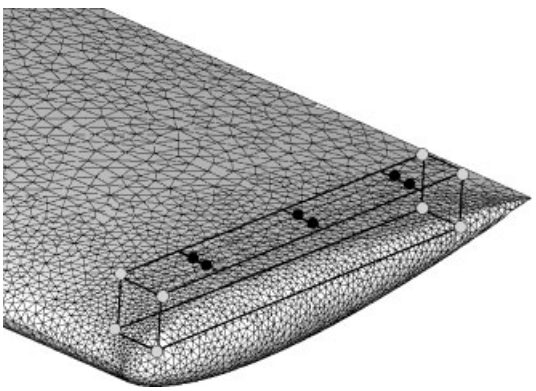

Figure 13. Parameterization.

The problem is parameterized by the FFD method. Six control points are moved vertically during the optimization, located on the succion face at two sections near the salmon, as shown in Figure 13. All calculations are performed using a multi-block parallelization approach involving 64 R14000 processors.

Actually, it would be unrealistic to solve this problem using classical GAs, because of the cost of a single evaluation. For the present calculation, a population of 20 individuals is considered, evoluting during 23 generations with the probability $p_c = 0.95$ and $p_m = 0.02$, and a binary coding on 24 genes. For the first and second generations, all evaluations are performed by the flow solver, which represents almost a quarter of the global computational cost. Then, inexact pre-evaluations are introduced in the calculations. The ANNs are trained using 10 entries in the database. An attenuation factor of 10 is employed for the evaluations,

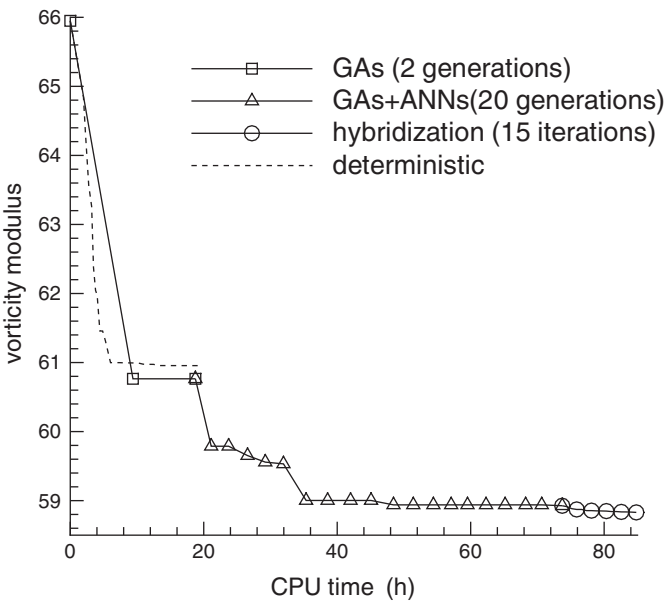

Figure 14. Evolution of the cost function.

which seems to be a safe value, considering the number of entries used for the training, according to the previous tests performed. At each generation, four individuals are exactly evaluated by the flow solver. At generation 23, one considers that a good approximation of the optimal design is reached by GAs and an hybridization process is started to refine the design in a few iterations of a deterministic method. The evolution of the cost function during the whole optimization procedure is shown in Figure 14. A reduction of 12% of the vorticity modulus is observed. It should be underlined that the use of the deterministic method from the initial shape would lead to a poor improvement, as depicted in Figure 14, which justifies the use of the present more sophisticated approach. The efforts on the wing are not modified significantly during the optimization procedure, since the modifications of the shape are local. The drag and the lift are increased of 0.6 and 0.4 per cent respectively.

The final shape is characterized by a thin bump along the junction between the wing and the salmon (Figures 15(a) and 16(a)). One can notice that the final shape found by using the deterministic method alone has a smaller bump and its characteristics are obviously different, as shown by Figures 15(b) and 16(b). The need of a more sophisticated optimization strategy is therefore demonstrated for this case.

If one examines the pressure fields in a vertical plane located just downwind of the wing (Figures 17 and 18), one may observe a decrease of the low-pressure area at the centre of the vortex, underlining the reduction of vorticity. The wall streamlines on the wing show a modification of the convergence line, which is longer for the final shape. The process by which the vorticity is reduced may be easily understood by looking at the total pressure fields at a location close to the middle of the chord of length l (Figures 19 and 20). For the initial shape, the detachment has not yet begun at this location, although for the optimized shape a

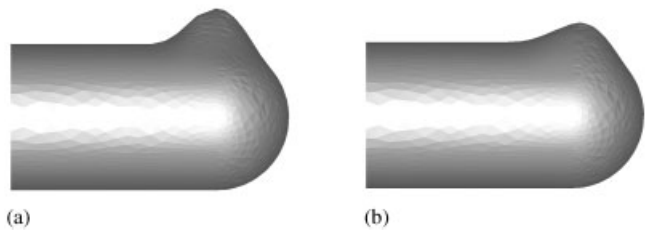

Figure 15. Front view of final shapes.

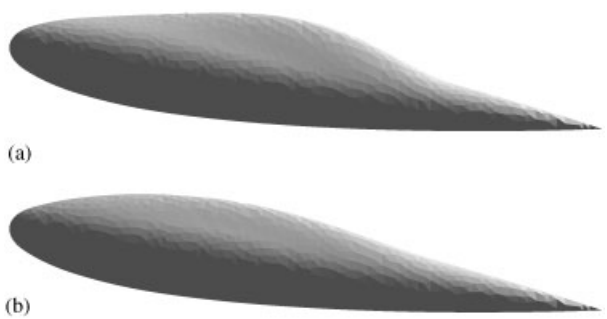

Figure 16. Transversal view of final shapes.

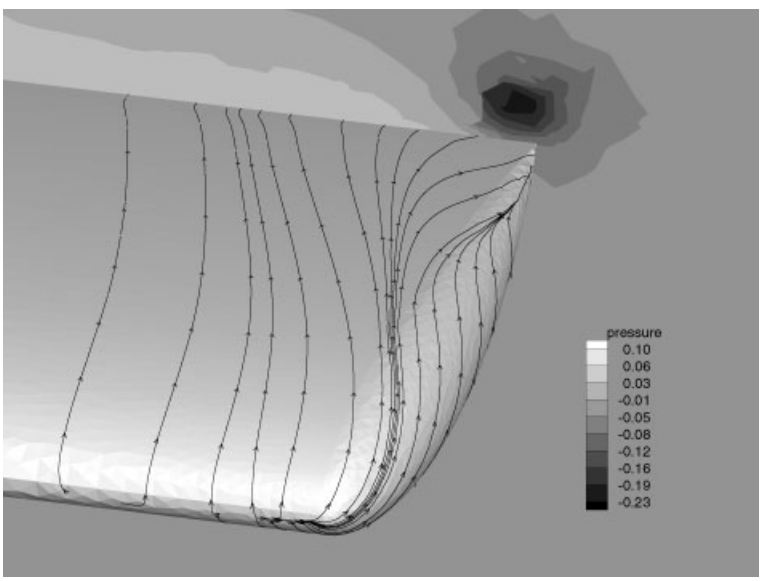

Figure 17. Wall streamlines and pressure field—initial shape.

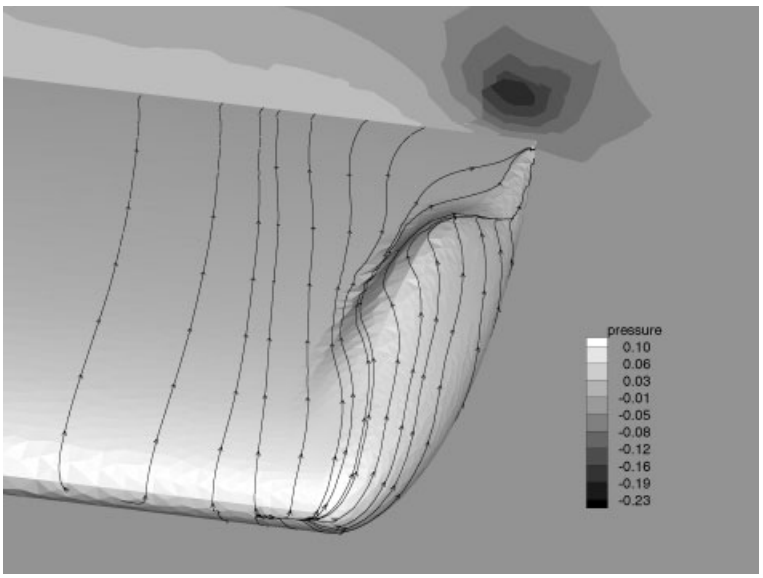

Figure 18. Wall streamlines and pressure field—final shape.

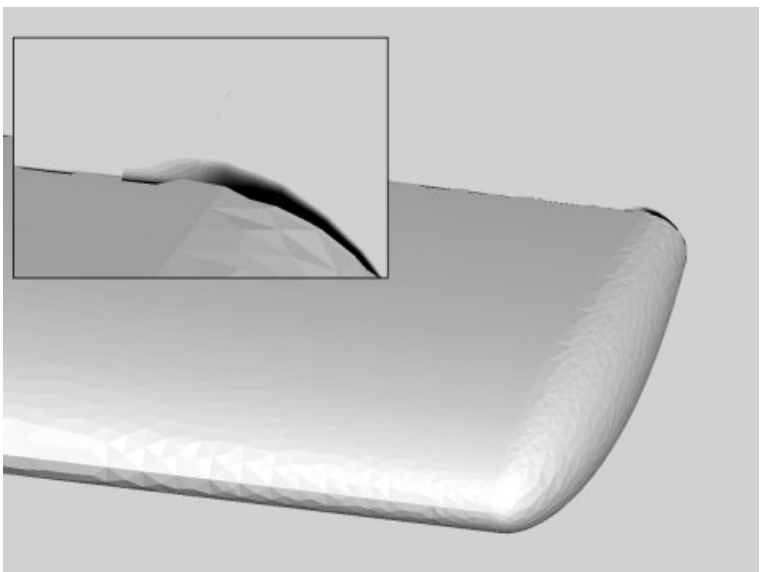

Figure 19. Total pressure field at $x/l = 0.7$—initial shape.

vortex may be already observed. This comparison shows that the shape modification yields a detachment upwind. Therefore, the vortex is dissipated earlier and the vorticity downwind of the wing is consequently reduced.

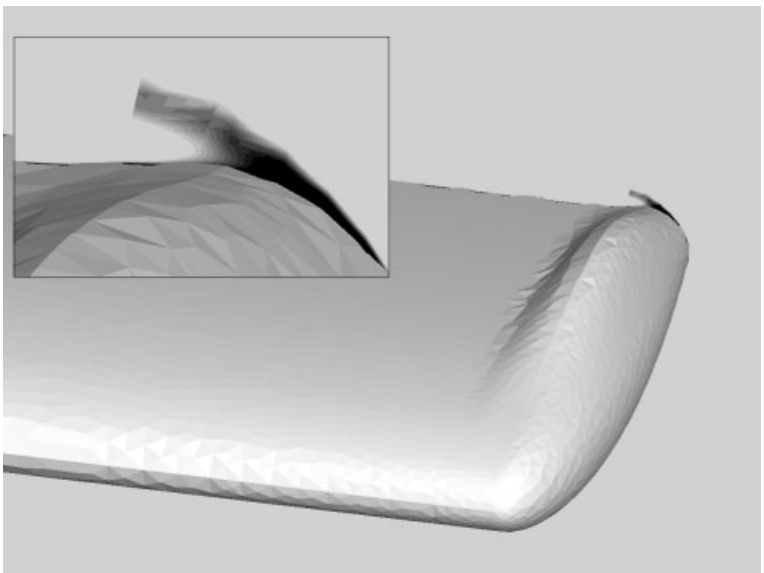

Figure 20. Total pressure field at $x/l = 0.7$—final shape.

8. CONCLUSION

Two acceleration techniques for GAs were evaluated in the particular framework of complex CFD applications. First, inexact pre-evaluations through ANNs were introduced to reduce the number of evaluations by the flow solver at each generation. Actually, one expects to evaluate through the flow solver only promising individuals. Then, the hybridization of GAs was proposed, to reduce the costs needed by the local search, this final step being performed by a deterministic approach. A complex flow solver, involving RANS computations and near-wall turbulence modelling, was used for the applications.

The use of ANNs was successfully applied to the optimization of a two-dimensional airfoil, reducing strongly the CPU time required and maintaining the capacity of GAs to determine the global optimum. The hybridization provided interesting results as well, but it seemed relatively tricky to obtain a real acceleration effect for practical applications. However, this strategy is advised to refine the shape determined by GAs.

Finally, the complex optimization of a three-dimensional airfoil was studied. Although this kind of calculations is still expensive, it was shown that it is now possible and realistic to solve optimization problems with high-fidelity CFD codes, using accelerated GAs and then local shape refinement by a deterministic approach. These methods are still more expensive than gradient-based algorithms, but they give the capability of performing a cheap global exploration first, and then an accurate local search of the optimal design. The main limitation of this approach is related to the low number of design variables used, which is not very restrictive as long as local shape modifications are considered.

ACKNOWLEDGEMENTS

The authors gratefully acknowledge the Scientific Committee of CINES (Project dmn2050) and IDRIS (Project 1308) for the attribution of CPU time.

REFERENCES

1. Anderson WK, Venkatakrishnan V. Aerodynamic design optimization on unstructured grids with a continuous adjoint formulation. *Computers and Fluids* 1999; **28**:443–480.
2. Jameson A, Martinelli L, Pierce NA. Optimum aerodynamic design using the Navier–Stokes equation. *Theorical and Computational Fluid Dynamics* 1998; **10**:213–237.
3. Anderson WK, Nielsen E. Aerodynamic design optimization on unstructured meshes using the Navier–Stokes equations. *AIAA Journal* 1999; **37**:1411–1419.
4. Giunta A, Dudley J, Narducci R, Grossman B, Haftka R, Mason W, Watson L. Noisy aerodynamic response and smooth approximations in HSCT design. *AIAA Paper* 94-4316, 1994.
5. Hosder S, Grossman B, Haftka R, Mason W, Watson L. Observations on CFD simulation uncertainties. *AIAA Paper* 2002-5531, 2002.
6. Madsen J. Response surface techniques for diffuser shape optimization. *AIAA Journal* 2000; **38**:1512–1518.
7. Goldberg D. *Genetic Algorithms in Search, Optimization and Machine Learning*. Addison Wesley Company Inc: Reading, MA, 1989.
8. Poloni C, Giurgevich A, Onesti L, Pediroda V. Hybridization of a multi-objective genetic algorithm, a neural network and a classical optimizer for a complex design problem in fluid dynamics. *Computer Methods in Applied Mechanics and Engineering* 2000; **186**:403–420.
9. Spalart PR, Allmaras SR. A one-equation turbulence model for aerodynamic flows. *AIAA Paper* 92-0439, 1991.
10. Menter F. Zonal two-equations k–ω turbulence models for aerodynamic flows. *AIAA Paper* 93-2906, 1993.
11. Deng G, Visonneau M. Comparison of explicit algebraic stress models and second-order turbulence closures for steady flows around ships. In *Proceedings of the Seventh International Conference on Numerical Ship Hydrodynamics*, Piquet J (ed). Ecole Centrale de Nantes: Nantes, France, 1999.
12. Farhat C, Degand C, Koobus B, Lesoinne M. Torsional springs for two dimensional dynamic unstructured fluid meshes. *Computational Methods in Applied Mechanics and Engineering* 1998; **163**:231–245.
13. Sederberg T, Parry S. Free-from deformation of solid geometric models. *Computer Graphics* 1986; **20**:151–160.
14. Giotis A, Giannakoglou K. *Eurogen* 99, *Evolutionary Algorithms in Engineering and Computer Science*. John Wiley & Sons: New York, 1999.
15. Giotis A, Giannakoglou K. *Genetic Algorithms for Optimization in Aeronautics and Turbomachinery*, VKI Lecture Series, VKI Publications: Brussels.
16. Powell M. Radial basis function methods for interpolation to functions of many variables. In *Fifth Hellenic-European Conference on Computer Mathematics and its Applications*, Athens, 2001.
17. Marazzi M, Nocedal J. Wedge trust region methods for derivative free optimization. *Mathematical Programming* 2002; **91**:289–305.

