OPTIMIZATION AND INVERSE DESIGN IN AERONAUTICS: HOW TO COUPLE GENETIC ALGORITHMS WITH RADIAL BASIS FUNCTION NETWORKS.

Kyriakos C. Giannakoglou

National Technical University of Athens,
9, Iroon Polytechniou Str., Athens 157 73, GREECE
Tel: (30)-1-772.16.36, Fax: (30)-1-772.37.89
kgianna@central.ntua.gr

Abstract. This text describes the outcome of modern research activities which serve to reduce the computing cost of any stochastic optimization method that works with populations of candidate solutions, rather than a single individual. The proposed technique is based on Computational Intelligence and its application will be demonstrated along with Genetic Algorithms (GAs). A particular class of Artificial Neural Networks ($ANNs$), the so–called Radial Basis Function (RBF) networks, will be used to pre–evaluate the entire population and indicate a few individuals that deserve to undergo costly, exact evaluations. Besides, the same network(s) will be used to estimate the cost function gradient at the current best solution(s), which may improve further these solutions using a simple descent scheme.

Key words: Genetic Algorithms, Artificial Neural Networks, Radial Basis Function Networks, Aerodynamic Shape Optimization.

1 INTRODUCTION

Genetic Algorithms (*GAs*, [1], [2]) are probably the most widely used stochastic optimization method, in various scientific and industrial areas. *GAs* operate on populations. Starting from a randomly selected initial population of candidate solutions, the population is free to evolve by means of stochastic transition rules and encoded natural processes (parent selection, crossover, mutation, etc.). During the evolution, individuals associated with their own payoff values, compete and the fitter are likely to contribute more offspring in the next generation.

The most prominent advantage of *GAs* is that they are difficult to fool even in complex, multimodal solution landscapes. On the other hand, their only drawback is that a great number of cost function evaluations is needed. Let us make clear that, roughly speaking, the computing cost of the genetic operations is zero. What costs is the evaluation of the would–be solutions. As a typical example, in the field of aeronautics or turbomachinery, the optimization or the inverse design of an airfoil calls for Computational Fluid Dynamics (*CFD*) tools. Unfortunately, although modern *CFD* tools, with sophisticated turbulence models, may enlight and quantify complex flow phenomena, they are very costly. Therefore, the need for minimizing the number of evaluations required by the *GA* for a given level of solution accuracy, is pressing. Through solving this problem, *GAs* with low computing cost and high competitiveness will be available.

In view of the above, enhanced versions of *GAs* have been devised in the past, [3], [4], [5], [6], based on cheap, inexact pre–evaluation (*IPE*) techniques. Working with *GAs*, the *IPE* technique is used in order to skip a great part of unnecessary evalutions in each generation. Using artificial intelligence and the previously seen solutions, individuals that are likely to die out are identified and treated without resorting to costly exact evaluations. Conceptually, the already examined solutions with their payoff values form a hypersurface, which is used to approximate the fitness of any new individual. The "interpolation" tool, which is the heart of the *IPE* technique is based on trained Radial Basis Function (*RBF*) networks. Recent activities and tests demonstrated that the *RBF* networks, with "local" training, outperform the standard multilayer perceptron employed in the aforementioned papers. In the present method, the role of the *RBF* networks is dual. First, to (inexactly) evaluate a new individual, without resorting to the exact but costly evaluation tool if this individual does not seem to be a promising solution. Second, to guess cost function derivatives, that will be used to further improve the best individuals in the current generation. We give emphasis to the second role of the *RBF* networks, which is the novel point in this work.

2 ABOUT GENETIC ALGORITHMS

GAs are optimization methods based on the mechanics of natural selection and natural genetics. *GAs* seek global optimal solutions, by making use of previous information from already examined search points to speculate on new ones with improved performance, [1]. There are good reasons for the widespread use of *GAs* in various application domains. The most important are listed below:

(a) they are robust and may capture the global optimum solution, without being trapped to local optima,

(b) the only information they require is one payoff value per objective for each candidate solution, according to a predefined cost function,

(c) they may readily incorporate any existing evaluation software, like *CFD*, *CEM*, etc solvers, with the minimum user effort,

(d) they can be easily parallelized, so that different members of the current population may run concurrently on different networked processors and

(e) they may handle either single- or multi-objective problems.

We do not intend to present a detailed analysis of *GAs* (see [1], [2], etc.) but we will rather stick with their enhancement using the *IPE* technique and especially with the descent of the current best solution using inexactly computed gradients. We recall that the *ANN*–based *IPE* acts as an auxiliary tool supporting the genetic search.

As stated in the Introduction, *GAs* handle populations of candidate solutions. Let N_{pop} be the population size, i.e. the number of candidate solutions examined within each generation. Regardless of the coding procedure (binary or real coding can be used), the i-th candidate solution $x_m^{(i)}$, $m = 1, M$, $i = 1, N_{pop}$ is an array formed by the values of the M free-parameters. In particular, if the decision variables are coded as some finite-length binary strings, formed by 0s and 1s, the *GA* operates on (head-to-tail) concatenated strings. The formation of the mating pool for the next generation, along with two important genetic operators, crossover and mutation, are the steps that help guide a directed search for improvement. During these operations, only payoff values, i.e. the cost or fitness function values, are necessary.

In order to form the mating pool and create new offspring, the members of each generation compete. In the *GA* variant used herein, parent selection is based on linear ranking (for the 15% of the population) and on probabilistic tournament (for the rest of it). One–point crossover per variable is used with possibility 85%. Binary coding is employed, where bits are allowed to mutate with a small probability (about or less than 0.1%).

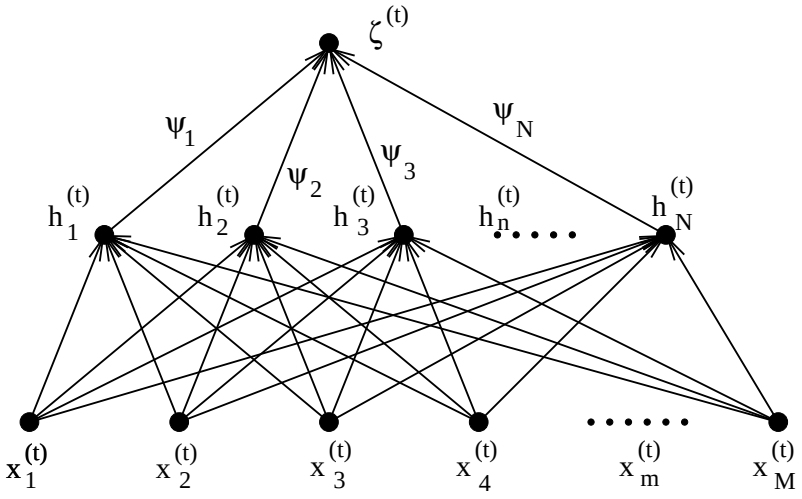

Figure 1: An *RBF* network with a single output unit.

3 ABOUT ARTIFICIAL NEURAL NETWORKS

3.1 The *RBF* Networks – Definitions

An *RBF* network, [7], [8], can safely be used as a surface approximation method in a high-dimensional space, and so does an *ANN* of any other kind. The reasons for selecting the *RBF* networks for this purpose will become clear as the paper develops. An *RBF* network, after being trained on a number of paired inputs-outputs, is capable of guessing the output value of any new input which does not belong to the training set. The accuracy of the *RBF* outputs depends on several parameters: the shape of the hypersurface that is to be approximated, the availability of a representative number of data for the network training and the basic characteristics of the network itself. In general, *ANNs* used for this purpose may consist of multiple layers (at least one, over and above the input and output layers) with varying numbers of processing units on the so–called hidden layers. On the contrary, an *RBF* network possesses three layers in total. These are the input layer with M sensory units, the hidden layer with N processing units and the output layer with a single unit, if a single function is to be approximated. Such a typical network is illustrated in fig. 1. According to this structure, a nonlinear mapping ($\mathcal{R}^M \rightarrow \mathcal{R}^N$) from the input units to the hidden ones is then followed by a linear one ($\mathcal{R}^N \rightarrow \mathcal{R}^1$) to the output unit. Crucial parameters affecting its performance are the number N of hidden units and the so-called *RBF* centers associated with them. No weight factors are associated with the connections between the input and the hidden units. On the other hand, N weight factors ($\psi_n, n = 1, N$) are given to the links that end to the output units and these should be computed during the training.

The training of the *RBF* network on a complex input-output mapping, like that between the M geometrical parameters defining an aerodynamic shape (according to a parametric modelling that relies often on Bezier or other polynomial curves, [3]) and its performance (drag, lift, deviation from a prescribed pressure or velocity distribution along its walls, etc.), allows new shapes to be evaluated at no cost at all. This evaluation is, of course, approximate but it should be considered as sufficient for the less important individuals in a generation.

3.2 Training the *RBF* Networks

Let $x_m^{(t)}, m = 1, M, t = 1, T$ be the components of the T input patterns which have been selected for training the network. For each one of them, the corresponding output value $y^{(t)}, t = 1, T$ is also known. In a genetic optimization, the input–output pairs are available from evaluations that took place during the preceding generations. Without loss in generality, we may assume that the number of hidden units equals the size of the training set ($N = T$). As it will be analyzed below, in the proposed method this is feasible, since the training examples are a subset only of the available information, selected through proximity criteria, [5]. Once T of them are selected, the number of hidden units N is fixed ($N = T$).

Each one of the $N = T$ hidden units is given a M–dimensional array, which is called the *RBF* center. Let $\vec{c}^{(n)}, n = 1, N$ be these centers. With $N = T$, the evident choice is

$$\vec{c}_m^{(t)} = \vec{x}_m^{(t)} \quad , \quad t = 1, T, \ m = 1, M \tag{1}$$

The mapping ($\mathcal{R}^M \rightarrow \mathcal{R}^N$) to the hidden units computes the n^{th} hidden unit value $h_n^{(t)}$ using a nonlinear activation function Φ, as follows

$$h_n^{(t)} = \Phi\left(\left\|\vec{x}^{(t)} - \vec{c}^{(n)}\right\|_2, r_n\right) \tag{2}$$

Various activation functions can be used, which in general perform differently. Here, the

$$\Phi(u, r) = \exp(-u^2/r^2) \tag{3}$$

has been used with constant $r_n = r$. This function is plotted in fig. 2, for two different values of r. Then, the output of the network is computed using the (unknown, during the training) weights $\psi_n, n = 1, N$, as follows

$$\zeta^{(t)} = \psi_n h_n^{(t)} \tag{4}$$

where summation applies to the repeated index n.

The weights $\psi_n, n = 1, N$ are computed by requiring

$$\zeta^{(t)} = y^{(t)} \quad , \quad t = 1, T \tag{5}$$

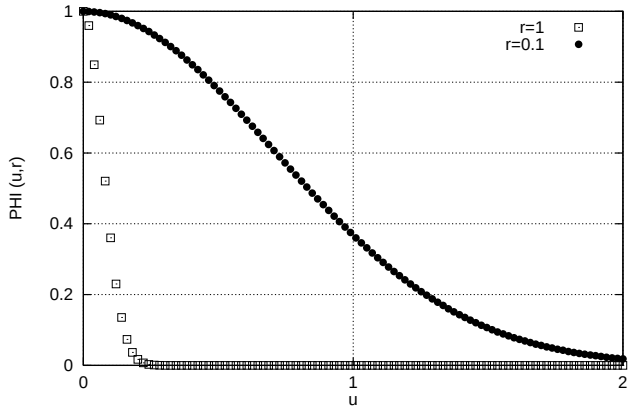

Figure 2: The activation function of eq. 3, for two values of r.

It is a simple matter to show that in order to compute the ψ values, the inversion of a square $(T \times N)$ (if $T = N$) matrix is required. This matrix, which will be referred to as H is formed by the $h_n^{(t)}$ coefficients. Each row of H corresponds to a learning example, whereas each column to an *RBF* center. We purposely choose T to be small (practically, between 10 and 20), so that the inversion of H through, for instance, the modified Gram-Schmidt technique, is non–costly. Also, a major advantage of the use of *RBF* networks is related to multi–objective optimization problems. Regardless of the number of outputs, a single H matrix is formed, so that a single inversion is needed. In practice, the training of an *RBF* network for multiple outputs costs as much as its training for a single output.

Trained *RBF* networks are capable of providing not only inexact cost function values but also their derivatives. During its exploitation, a trained network which is presented with $\vec{x}^{(*)}$ estimates an output

$$\zeta^{(*)} = \psi_n \Phi \left(\left\| \vec{x}^{(*)} - \vec{c}^{(n)} \right\|_2, r_n \right) \tag{6}$$

and, after a few mathematical manipulations, M partial derivatives, as follows

$$\frac{\partial \zeta^{(*)}}{\partial x_m} = -2 \sum_{n=1}^{N} \frac{\zeta^{(*)}}{r_n^2} \left(x_m^{(*)} - c_m^{(n)} \right) \quad , \quad m = 1, M \tag{7}$$

Eq. 7 is valid only for the activation function given by eq. 3. Particular treatment is needed if $\vec{x}^{(*)} = \vec{c}^{(n)}$.

3.3 Comments

The function

$$f(x) = e^{-0.5x} sin(4\pi x) + x^2 \tag{8}$$

will be used to demonstrate the use of *RBF* networks as an interpolation tool as well as a tool for forecasting derivatives. Two cases have been examined, both with the same training set size ($T = 21$). With $x \in [0, 1]$, the input patterns were first clustered toward $x = 0$ and $x = 1$, using a simple sinusoidal distribution. This case is presented on the left column of fig. 3. On the right column of the same figure, this problem is analyzed with T equidistant training examples. The value $r = 0.1$ was used. Regardless of the data–point distribution, the function is quite accurately captured.

Interesting conclusions can be drawn by examining the $\frac{\partial f}{\partial x}$ values computed by the trained network. In the first case, the clustered data–points near the edges provide accurate derivative values in these areas. On the contrary, slight discrepancies exist away from the two edges. The network which has been trained using equidistant x's fails to reproduce the analytical expressions for the derivatives close to $x = 0$ and $x = 1$. However, it performs very accurately in the middle.

It is interesting to comment on these different behaviors. At the bottom of fig. 3, the basis functions used in both cases have been plotted. These should be viewed as building elements which, scaled by the ψ factors and superimposed, create the final approximation curve. With equidistant basis functions (i.e. equidistant data) the two edges of the curves may benefit of a few basis function only, since the rest locally tend to zero. The low number of building elements make the approximation less flexible in this area. Any point close to the edges is described by almost half of the basis functions that are available in the interior. Even if differences are not so visible for the function itself, the derivatives show slight discrepancies (left column). The quality of predictions for both the function $f(x)$ and its derivative, improves if more data–points are used in this area (right column). Unfortunately, as expected, the improvement of quality at the edges deteriorates slightly the accuracy in the middle.

4 THE PROPOSED OPTIMIZATION ALGORITHM

As explained, in detail, in the preceding sections, the role of the *RBF* networks in the context of a *GA*–based optimization is to provide cheap (and hence inexact) fitness function values for the individuals along with approximate values for their derivatives with respect to the design variables.

In view of the above, the proposed optimization algorithm is given below. We recall that we handle minimization problems.

Phase 1: The starting population which consists of N_{pop} randomly selected individuals $\vec{x}_m^i$, $m = 1, M$, $i = 1, N_{pop}$ is formed.

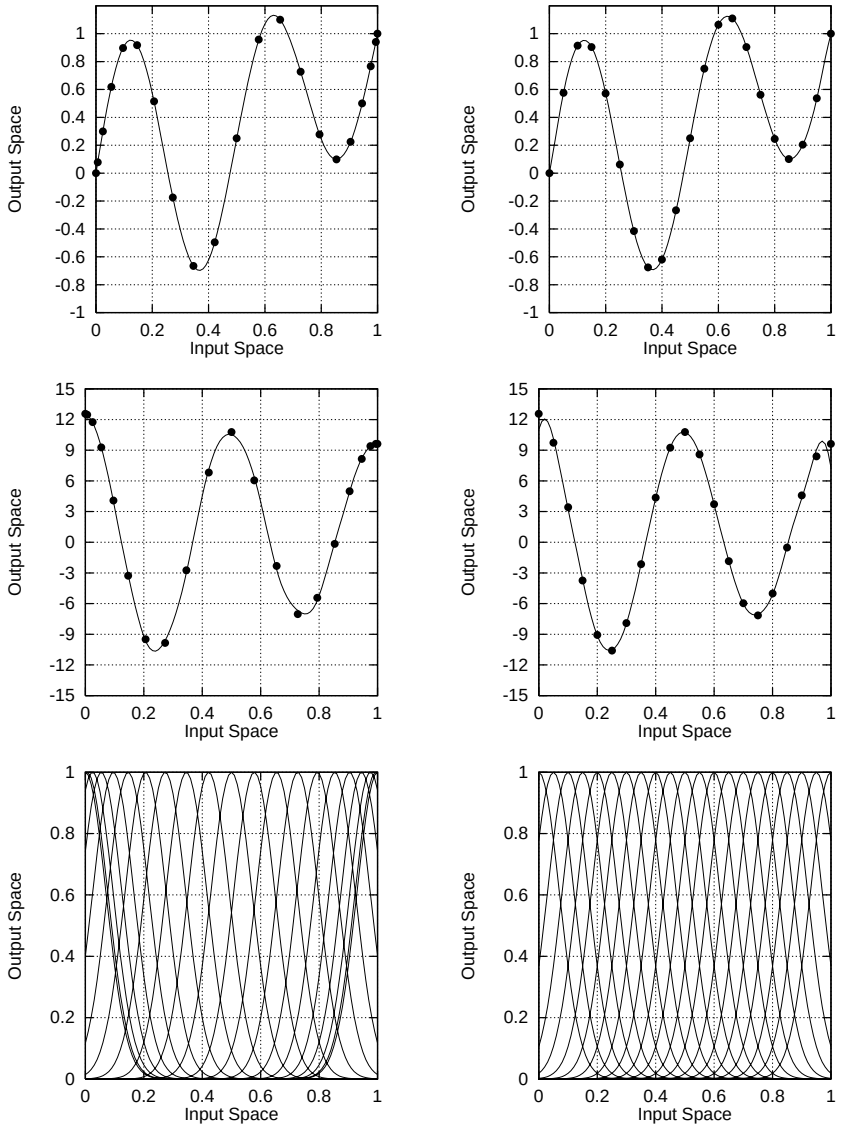

Figure 3: Example of the use of an *RBF* network to compute a function (upper) and its derivative (mid). On the lower part, the *RBF* functions are illustrated. Training set size $T = 21$. Left: Learning examples concentrated toward the edges. Right: Equidistant learning examples. (Points: Training set, continuous line: *RBF* network output)

Phase 2: The starting population keeps evolving for a few generations, during which the entire population undergoes exact evaluations. The so–evaluated individuals and their cost function values are stored in a database (*DB*). The number of generations recommended for Phase 2 depends on the population size N_{pop}. It

could be stated that, about $100/N_{pop}$ initial generations should be evaluated accurately prior to the implementation of the *IPE* scheme.

Phase 3: During the subsequent generations and for each individual, a "local" *RBF* network is formed and trained. For the training, the T *DB*–entries which lie closer to the current individual are selected and used. The trained "local" network is used to pre–evaluate the candidate solution, by practically performing a multivariate interpolation. The approximate cost function value as well as its derivatives, both computed by the *RBF* network, are associated with the individual.

Phase 4: Using the previously computed N_{pop} inexact cost function values, the σN_{pop} inividuals with the best scores are selected in order to be re–evaluated exactly, using the *CFD* tool. The inexact cost function values for these individuals are replaced with exact ones. The *DB* is enriched by individuals which have undergone exact evaluations.

Phase 5: The standard genetic operators are applied to the entire population, by merging exact and inexact scores. N_{pop} offspring are thus created. Then, the best K of them, are further modified through the steepest descent scheme, based on inexactly computed gradient components. The updated individuals are

$$\vec{x}^{new} = \vec{x}^{current} - \eta \nabla \zeta^{current} \tag{9}$$

where $\eta > 0$. Replace $\vec{x}^{current}$ with $\vec{x}^{new}$ if

$$f(\vec{x}^{new}) < f(\vec{x}^{current})$$

Phases 2 to 5 are repeated up to the final convergence.

5 METHOD EVALUATION – RESULTS

The proposed method will be used for the re–design of two airfoils, using prespecified pressure coefficient distributions along their solid walls. The first test geometry was the well–known *NACA12* isolated profile, whereas the second was a *NACA66-010* airfoil used in turbomachinery cascades. In both studies, a simple flow model, namely the panel method [9] for incompressible, irrotational airfoil flows was used. The reason for selecting such a simple model is that this was indeed a very fast evaluation tool; so, within the same wall clock time, the same problem was examined several times and the results presented herein are "average" results and "average"

code behaviour, i.e. the outcome of a great number of runs. It should be stated, of course, that the economy in computing time (number of evaluations required) offered by the present method does not depend on the flow model approximation; so, similar results are expected using more sophisticated flow models.

The isolated *NACA12* airfoil was examined at zero incidence. As it is symmetric, the pressure coefficient was symmetric too. The target distribution was computed using the same flow model and the conventional *NACA12* profile. The target and best computed pressure coefficient distributions using (any variant of) the proposed optimization method are shown in fig. 4. In the same figure, one may also see the standard *NACA12* profile shape and best computed one. Pressure distributions and geometries are in excellent agreement.

The convergence history of the conventional *GA*, the *GA* with *IPE* and the present enhanced method (*GA* with *IPE* and descent of the better individual(s) using gradient computed by means of the trained *RBF* networks) are all shown in in fig. 5. The *GA* with *IPE* is much faster than the conventional *GA*. In this case, $N_{pop} = 50$, $\sigma = 0.20$, and the *IPE* technique was first employed at the third generation. Additional speed–up can be obtained through the steepest descent improvement of the best individual ($K = 1$) in each generation. The value $\eta = 0.02$ was used. The training set size was constant and equal to $T = 20$.

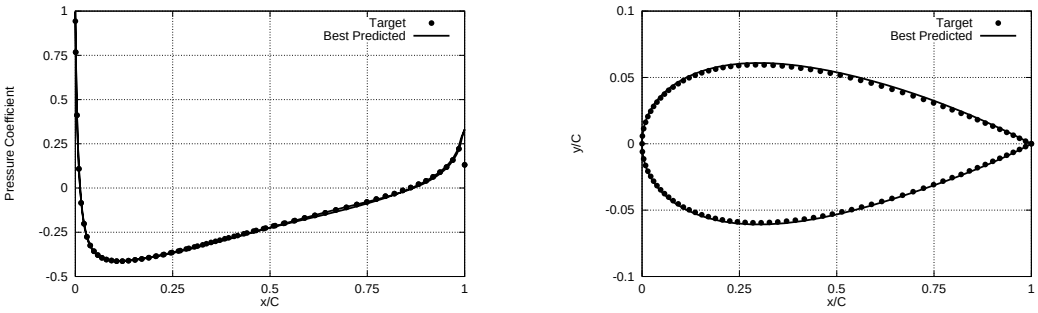

Figure 4: Re–design of *NACA12*, using the pressure coefficient distribution along the solid walls as target. Left: Target and best computed pressure coefficient distributions. Right: Target (standard *NACA12*) and best computed airfoil shape.

In the second problem, a two–dimensional cascade airfoil, the *NACA66-010* one with 30^o stagger angle, 60^o camber angle and solidity equal to 1 was re–designed. The flow angle was equal to 45^o and, as in the previous case, the target distribution was obtained using the same flow model (the evaluation tool). It should be pointed out that the geometrical modelling was done at zero stagger angle and the airfoil was rotated to the proper stagger angle at a pre–processing level within the evaluation software. The target and best computed pressure coefficient distributions are shown in fig. 6. The same figure illustrates also the standard *NACA66-010* profile

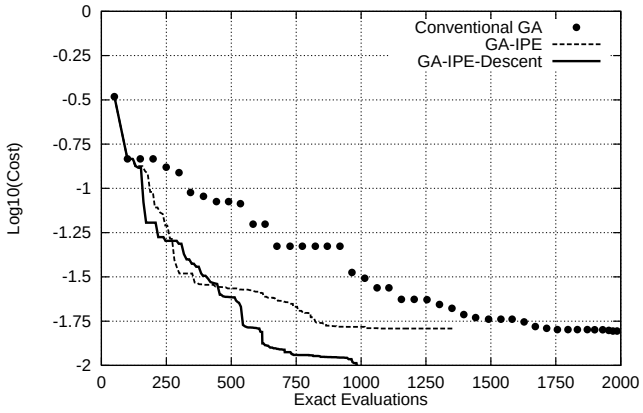

Figure 5: Re–design of *NACA12*: Convergence history of the conventional *GA*, the *GA* with *IPE* and the present method (*GA* with *IPE* and descent of the better individual(s) using the *RBF* network–computed gradient.

shape and best computed one. Pressure distributions and geometries are in perfect agreement.

As in the previous case, fig. 7 illustrates the convergence history of the conventional *GA*, the *GA* with *IPE* and the *GA* with *IPE* and steepest descent of the best individual in each generation. The *GA* with *IPE* is much faster than the conventional *GA*, and the present methods outperforms both of them, especially toward convergence, where the training set is adequate for a satisfactory training of the *RBF* network. The same parameters, as in the previous run, have been used.

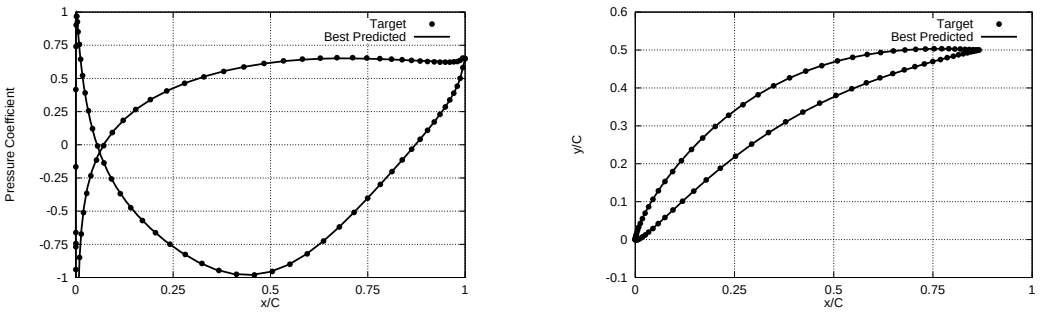

Figure 6: Re–design of *NACA66-010*, using the pressure coefficient distribution along the solid walls as target. Left: Target and best computed pressure coefficient distributions. Right: Target (standard *NACA66-010*) and best computed airfoil shape.

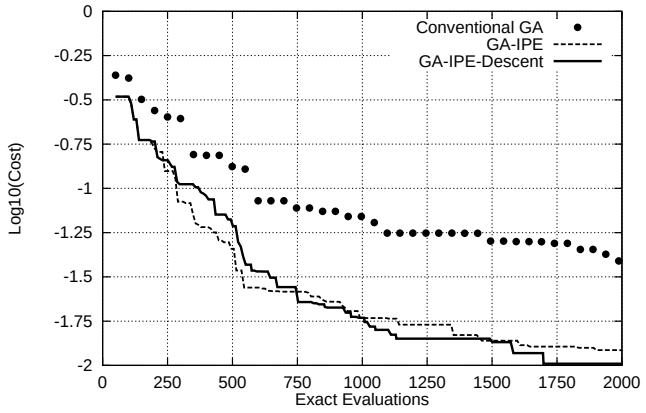

Figure 7: Re–design of *NACA66-010*: Convergence history of the conventional *GA*, the *GA* with *IPE* and the present method (*GA* with *IPE* and descent of the better individual(s) using the *RBF* network–computed gradient.

6 CONCLUSIONS

A technique has been proposed which can be incorporated in any conventional genetic optimization method in order to reduce the number of evaluations required to reach the optimum solution. The so–called *IPE* technique consists of a "cheap" pre–evaluation phase, based on "locally" trained *RBF* networks and has been presented in the past by the same author. The inexact and non–costly evaluation method distinguishes the most promising individuals which undergo costly exact evaluations. The contribution of this paper is that introduces a new speed–up mechanism in the aforementioned *IPE* technique. The *RBF* networks are used also to provide inexact values of the cost function gradient. Using this approximate gradient, the most promising individuals can be "pushed" toward the optimum solution, using the steepest descent technique. The analyzed test problems demonstrated the superiority of the proposed method. The profit from the use of this method is visible toward convergence, where the *RBF* networks become ready to provide "less inexact" gradient information.

Acknowledgement

This paper has been prepared in memory of Bertrand Mantel who has been involved in the early stages of this work. Part of this work was funded by Dassault Aviation. Support from Thematic Network INGENET is also acknowledged. The author would like to acknowledge Dr. T. Pappou and Mr. A. Giotis for their useful discussions.

References

[1] D.E. Goldberg. *Genetic Algorithms in search, optimization & machine learning.* Addison-Wesley, 1989.

[2] Z. Michalewicz. *Genetic Algorithms + Data Structures = Evolution Programs.* Springer-Verlag, Berlin Heidelberg, 2nd edition, 1994.

[3] K.C. Giannakoglou. Designing turbomachinery blades using evolutionary methods. ASME Paper 99-GT-181, 1999.

[4] A.P. Giotis and K.C. Giannakoglou. Single- and multi-objective airfoil design using genetic algorithms and artificial intelligence. EUROGEN 99, Evolutionary Algorithms in Engineering and Computer Science, Jyvaskyla, Miettinen K. et al (Eds.), John Wiley & Sons, Finland, May 1999.

[5] K.C. Giannakoglou. Acceleration of genetic algorithms using artificial neural networks - theoretical background. Von-Karman Institute LS 2000-07, May 2000.

[6] K.C. Giannakoglou and A.P. Giotis. Acceleration of genetic algorithms using artificial neural networks - application of the method. Von-Karman Institute LS 2000-07, May 2000.

[7] S. Haykin. *Neural Networks.* Prentice Hall International, Inc., 2nd edition, 1998.

[8] A. Cichocki and R. Unbenhauen. *Neural Networks For Optimization and Signal Processing.* John Wiley & Sons, 1993.

[9] V. Dedoussis. *Calculation of Inviscid Flow through Aerofoil Cascade and between Wind Tunnel Wall.* M.Sc. Thesis, Dept. Aeronautics, Imperial College, London, 1983.