Inverse Problems, Design and Optimization Symposium
Rio de Janeiro, Brazil, 2004

EVOLUTIONARY OPTIMIZATION USING A NEW RADIAL BASIS FUNCTION NETWORK AND THE ADJOINT FORMULATION

Ioannis C. Kampolis, Dimitrios I. Papadimitriou, Kyriakos C. Giannakoglou
Lab. of Thermal Turbomachines,
National Technical University of Athens,
P.O. Box 64069, 15710, Athens, Greece,
e-mail: kgianna@central.ntua.gr

ABSTRACT

Control theory, computational intelligence and stochastic optimization methods are put together to build a new search algorithm for use in optimization problems involving computationally expensive evaluation tools, such as the design of optimal aerodynamic shapes. The search for optimal solutions is carried out through Evolutionary Algorithms (*EAs*). Instead of using the exact and costly evaluation tool, *EAs* rely on a low–cost, regularly (re)trained new surrogate evaluation model. Compared to some previous works by the same group, where on–line trained surrogate models were used, the new method has two noticeable differences. The first one is that the surrogate model is built off–line, i.e. separately from the optimization method; thus, a repetitive scheme is established which converges to the optimal solution within a small number of external cycles. The second and most important difference is that the surrogate model is trained on both objective function values and its derivatives with respect to the design variables. For this purpose, a new Radial Basis Function (*RBF*) neural network is proposed, with extra terms and adjustable coefficients. Single– and multi–objective mathematical problems as well as the inverse design of a peripheral compressor cascade will be presented. For the latter, the objective function gradient is computed through solving the adjoint flow equations.

EVOLUTIONARY ALGORITHMS WITH SURROGATE MODELS

The computational cost of shape optimization methods in aeronautics or turbomachinery is proportional to the number of candidate solutions that should be evaluated by *CFD* software, before reaching the optimal solution. In population–based *EAs*, this number is usually high enough. To alleviate this problem, surrogate evaluation models can be used instead; in the past, several relevant algorithmic variants have been proposed by our research group, see [1–9] .

In those works, various surrogate evaluation models (multilayer perceptron, *RBF* networks, kriging model) have been incorporated into *EAs*, either in the global (in the earlier works) or in the local sense (in the most recent works).

A surrogate evaluation model will be referred to as "global" if this covers the entire search space. During the evolution, the global model needs to be updated regularly, using data from preceding evaluations. Its training is carried out off–line, whenever "enough" new data have been recorded. On the other hand, a "local" surrogate evaluation model needs to be trained for each and every individual anew, using the available neighboring data. This will be referred to as "on–line" training.

In general, global and local surrogate models can be used in the same manner: within

each generation of the *EA*, the surrogate models pre–evaluate the entire population (inexact pre–evaluation phase, *IPE* see [6], [9]). Then, only the top individuals needs to be evaluated using the costly *CFD* tool, before proceeding to the selection process in order to define the parent population for the next generation. We thus prevent a great number of non–promising population members from being exactly evaluated.

Even with the *IPE* technique, where the computational cost is reduced by almost one order of magnitude, gradient–based optimization algorithms usually make this faster than *EAs*! Among other, control theory is often employed to derive the adjoint to the flow equations whose solution yields the objective function gradient values, [10]. This can be used along with any gradient descent–like algorithm to get the optimal solution. However, by doing so, local instead of global optimal solutions could be found.

Despite the successful use of local surrogate models in evolutionary optimization, as shown in [7], [9], etc., we herein propose and assess a different optimization algorithm. All of the previously mentioned tools are used, though in a different way: *EAs* undertake the search for the optimal solution; an enhanced surrogate model is regularly rebuilt (off–line training) and used; the adjoint equations are used to support the surrogate model rather than the optimization method itself. The proposed method is described below, in detail.

THE PROPOSED OPTIMIZATION METHOD

As stated before, the new method is based on *EAs*, surrogate models and adjoint techniques but with a number of noticeable differences compared to our previous works. These are:

- The surrogate model is a new *RBF* network, first proposed in this paper. Since the network will be trained on both responses and their gradients with respect to the network input parameters, it should incorporate additional tuning parameters. The use of gradient information during training increases the predictive capabilities of the network or, alternatively, the same accuracy can be achieved using much smaller training datasets. The additional cost for training the network is almost negligible for small-sized networks. The new *RBF* network will be described in the next section.

- The *EA* is used to compute the "optimal" solution, using evaluations based on the surrogate model (this explains the use of quotes). According to the previous discussion, the surrogate model should be referred to as global. Upon convergence of the *EA*, the current "optimal" solution is exactly evaluated. The process is terminated or a new cycle starts, depending on the deviation between its exact fitness value and that computed via the surrogate model.

- Computation of objective function derivatives is made possible through the coupling of the flow equations solver with the numerical solution of the adjoint system of equations. The latter is discussed in a separate section. The *CPU* cost of the adjoint equations solver does not exceed the cost of the numerical solution of the flow equations and does not depend on the number of design variables. So, practically, the *CPU* cost for the analysis of a candidate solution (solution of the flow and adjoint equations) is about twice the cost of computing only its response (solution of the flow equations).

- In multi–objective problems, as many *RBF* networks as the number of objectives should be trained. The multi–objective algorithm differs from the single–objective one in the manner new training datasets are defined.

The single–objective optimization algorithm is described below:

Step 1: The starting training dataset for the surrogate model is created. The training patterns can be defined using any *Design of Experiment* (*DoE*) technique, to attain maximum information for the response surface through the minimum number of patterns. Regular grids, random

sampling, (full or fractional) factorial designs, orthogonal arrays etc. can be used. In this paper, the training patterns are either chosen at random or defined as the nodal points of a grid fitted to the search space.

Step 2: For the previously selected individuals, the flow solver and the adjoint equations are solved. The so–computed responses and derivatives are stored in the training database.

Step 3: The surrogate model (*RBF* network) is trained.

Step 4: The *EA* software is used to get the "optimal" solution, using only the surrogate evaluation model.

Step 5: The "optimal" solution is re–evaluated separately, using the exact evaluation tool. If the deviation between the approximate and exact fitness is less than a user defined threshold, the algorithm terminates here.

Step 6: The training set is redefined by adding new entries or eliminating some of the existing ones. The most recent "optimal" solution which has been evaluated exactly in step 5 is added to the training set; at the same time, the closest to the "optimal" solution training pattern should be eliminated if their distance (nondimensional, measured in the parametric space) is less than a user–defined value. Over and above, κ new training patterns (κ is a user–defined small integer) are evaluated and then added to the dataset. These are selected by running κ minor optimization problems, seeking for points with maximum average distance from the existing training patterns. This search is also carried out by *EAs* and its computing cost is negligible. Finally, windowing, i.e. the reduction of the search space is possible (though optional) and, by doing so, some other entries are eliminated from the new training set. Return to step 2.

With some modifications, the same algorithm applies to multi–objective problems as well. In this case, instead of adapting the search space (*Step 6*) around a single "optimal" solution, a more complicated algorithm is used to carry out the adaptation around the Pareto front members. A thinning process is used to reduce the number of entries in the Pareto front and control the number of additional entries into the training dataset.

THE NEW *RBF* NETWORK

For the new *RBF* network description, we will assume M inputs (corresponding to the M design variables) and a single output. A typical *RBF* network is shown in fig. 1; it involves N hidden units and, for its training, N patterns are used. The response is given by

$$y = \sum_{i=1}^{N} \psi_i \, exp(-\sigma_i) \qquad (1)$$

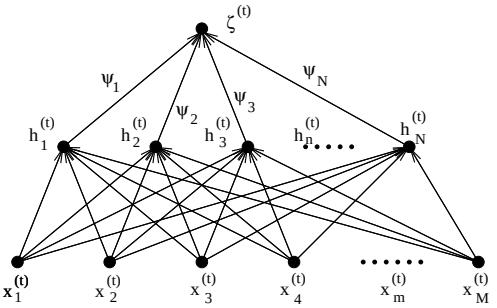

Figure 1: An *RBF* network with a single output unit.

Unlike conventional *RBF* networks, the weights ψ_i are not the unknown parameters to be computed during the training. Instead, they are expressed as linear combination of the real unknown quantities b_i and $a_{i,m}$, namely

$$\psi_i = b_i + \sum_{m=1}^{M} a_{i,m}[1 + (x_m - c_{i,m})] \qquad (2)$$

where x_m and $c_{i,m}$ denote the m-th component of a training pattern and the i-th *RBF* center, respectively. The *RBF* centers coincide with the training patterns. The σ_i quantitites are given by

$$\sigma_i = M \sum_{m=1}^{M} I_{i,m}(x_m - c_{i,m})^2 \qquad (3)$$

As previously stated, the b_i and $a_{i,m}$ coefficients are computed during the training process. A linear system is formulated by setting responses and derivatives computed by the network equal to the corresponding archived values for each training pattern, separately. The resulting system is solved using a direct inversion method. The iterative *Back–Error–Propagation* (*BEP*) method can be used instead; *BEP* is cheaper to run with huge training datasets and/or high–dimensional multivariate problems.

THE ADJOINT FORMULATION OF THE FLOW EQUATIONS

The *CFD* evaluation sofware used to evaluate candidate solutions is a primitive variable, implicit solver of the compressible, inviscid fluid flow equations, written as a hyperbolic system of equations

$$\overrightarrow{R}(\overrightarrow{U}) \;=\; \frac{\partial \overrightarrow{U}}{\partial t} + \frac{\partial \overrightarrow{F}}{\partial x} + \frac{\partial \overrightarrow{G}}{\partial y} + \frac{\partial \overrightarrow{E}}{\partial z} = \overrightarrow{0} \quad (4)$$

where $\overrightarrow{U}$ is the array of the conservative flow variables. The convection terms are discretized using second–order upwind schemes. The target is to reproduce a given pressure distribution p_t over the blade surface. The objective function is defined as

$$I = \frac{1}{2} \iint_{S_w} (p - p_t)^2 dS \quad (5)$$

and, through adding the Euler equations multiplied by the costate variables $\overrightarrow{\Psi}$, we come up with the augmented objective function, [10]

$$I = \frac{1}{2} \iint_{S_w} (p - p_t)^2 dS_w + \iiint_V \overrightarrow{\Psi}^T \overrightarrow{R} dV \quad (6)$$

or, symbolically,

$$I = I(\overrightarrow{U}, \overrightarrow{\Psi}, \overrightarrow{X}) \quad (7)$$

where V is the flow domain and $\overrightarrow{X}$ stands for the design variables' array. The continuous adjoint formulation is set up to compute derivatives

$$\frac{dI}{d\overrightarrow{X}} = \frac{\partial I}{\partial \overrightarrow{U}} \frac{\partial \overrightarrow{U}}{\partial \overrightarrow{X}} + \frac{\partial I}{\partial \overrightarrow{X}} \quad (8)$$

and leads to the linear adjoint equation

$$\frac{\partial \overrightarrow{\Psi}}{\partial t} - A^T \frac{\partial \overrightarrow{\Psi}}{\partial x} - B^T \frac{\partial \overrightarrow{\Psi}}{\partial y} - C^T \frac{\partial \overrightarrow{\Psi}}{\partial z} = \overrightarrow{0} \quad (9)$$

which is solved through the same numerical kernel with the Euler equations. Here $A = \frac{\partial \overrightarrow{F}}{\partial \overrightarrow{U}}$, etc.

RESULTS AND DISCUSSION

To demonstrate the general capabilities of the method, single– and multi–objective mathematical optimization problems are first solved. In particular, problems with only two free parameters contribute a lot to the understanding on the method, through simple response surface plots. Of course, in these problems, the objective function gradient is computed analytically. As already stated above, the engineering problem with which this paper is dealing with is the inverse design of a $3D$ compressor peripheral cascade.

The Single–Objective Rastrigin Function

The first mathematical problem is the minimization of the Rastrigin function. Regardless of the number of free parameters (M), the Rastrigin function has many local stationary points and is a typical test problem for assessing the capabilities of optimization methods. It is defined by

$$F_R\ (\overrightarrow{x}) = \sum_{i=1}^{M} \left[x_i^2 - 10cos(2\pi\ x_i) + 10 \right] \quad (10)$$

First of all, the solution of the problem with $M = 2$ is demonstrated. Both independent variables $x_i,\ i = 1, 2$ were bounded in $[-1, 1.5]$. The search space was purposely defined to be asymmetric with respect to the global minimum $(0, 0)$. At the first cycle, the surrogate model was trained using $N = 30$ randomly chosen patterns. None of them was allowed to lie within a small circle (radius=0.30) centered at $(0, 0)$, i.e. too close to the final solution, fig.2. Two cycles were sufficient for the optimization process to converge to the global optimum. This was due to the excellent representation of the exact response surface that the new RBF network achieved from the first cycle, even with only 30 training patterns. It is evident that the second cycle contributes practically to the

refinement of the current solution. In order to demonstrate the prediction accuracy of the surrogate model (built in the first cycle), a dense grid was generated and the model responses over its nodal points were computed. The corresponding iso–areas are shown in fig.3. This figure also includes the exact responses as well as responses computed using a conventional RBF network (trained on responses, without accounting for gradients). The superiority of the new model is obvious. Moreover, it is interesting to note that some subareas in fig.2 (among them, the area close to the optimal solution) are point–free (no points within these areas were included in the database). This is where the conventional RBF network systematically fails; in contrast, the new surrogate model, driven also by gradient data, yields excellent predictions.

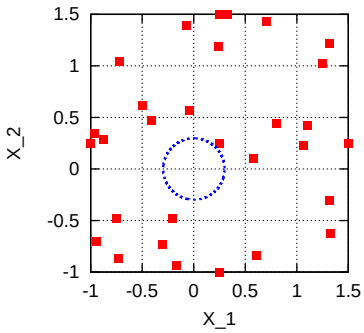

Figure 2: Rastrigin Function, $M = 2$: Distribution of the $N = 30$ randomly chosen training patterns and the point–free circle around the optimal solution.

The same optimization problem was also solved with $M = 10$ degrees of freedom, with the aforementioned bounds for all variables and the same training pattern selection technique. At the first cycle, the surrogate model was trained using $N = 100$ patterns selected at random and, after six optimization cycles, the global minimum was found. The convergence history is shown in fig.4. It is clear that, in the first three cycles, the EA which was supported by the surrogate model yielded "optimal" solutions far from the real one (the "optimal" solutions were temporarily approaching local stationary points). After the first three cycles, the EA was capable to yield an "optimal" solution close to the real one which was further

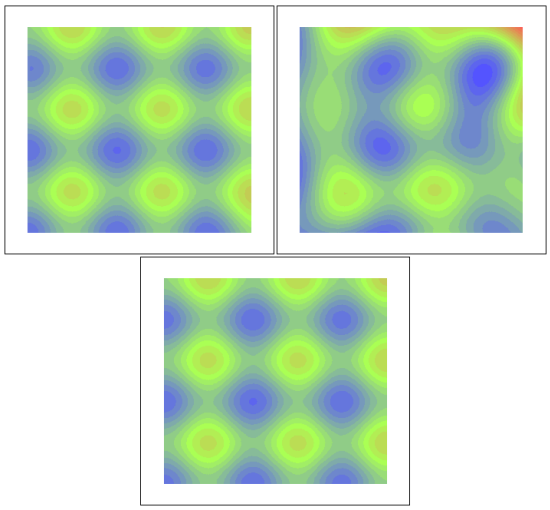

Figure 3: Rastrigin Function, $M = 2$: Response surfaces (a) predicted by the new RBF network (top–left), (b) predicted by a conventional RBF network, i.e. without taking into account gradient information (top–right) and (c) the exact one (analytically computed, bottom).

refined during the last two cycles.

Apart from the initial 100 patterns, 25 more were exactly evaluated during the subsequent cycles. Thus, 125 objective function and gradient evaluations were necessary which, for an equivalent flow problem (where the adjoint equations should be solved to compute the gradient) means that the total computing cost would be equal to 250 equivalent flow solutions.

Two–Objective Minimization Problem

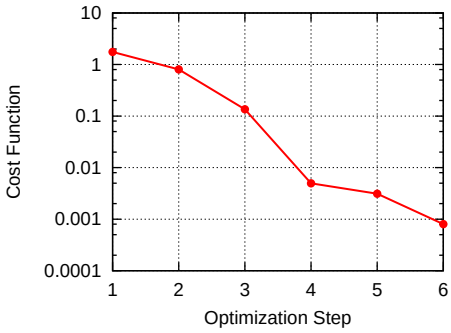

Figure 4: Rastrigin Function, $M = 10$: Convergence history of the optimization algorithm.

The second problem involves two mathematical functions with $M = 2$ degrees of freedom each. Both should be minimized, leading thus to the Pareto front of optimal solutions. The two functions are

$$F_k(x) = \left(\frac{\sum_{i=1}^{N} \left[(x_i - a_k)^2 - 10 \ cos(2\pi(x_i - a_k)) + 10 \right]}{2} \right)^{0.25} \tag{11}$$

where $k = 1, 2$, $a_1 = 0$, $a_2 = 1.5$, $x_1 \in [-2.12, 1.12]$ and $x_2 \in [-1.12, 2.12]$.

The starting training set was formed by $N = 100$ randomly selected patterns, the same for each objective function. By the end of each cycle, the Pareto optimal solutions were exactly evaluated and added to the training dataset. In order to avoid extra (unnecessary) computing charge that might occur whenever the Pareto front is overcrowded, a thinning technique was employed. The role of front thinning is to identify a subset of the Pareto members (its size is user–defined), based on distance considerations. In the literature, several thinning techniques for the Pareto front are proposed, but more comments on them is beyond the scope of this paper.

At the end of the optimization process the training dataset consisted of 133 patterns. The Pareto front finally computed using the surrogate model is compared to the exact front, as shown in fig.5. The response surfaces predicted by the proposed surrogate model, the conventional RBF network and the exact response surface, over the nodal points of a dense grid, are shown in fig. 6. The new RBF networks is much better than the conventional one, especially as far as the first function is of concern.

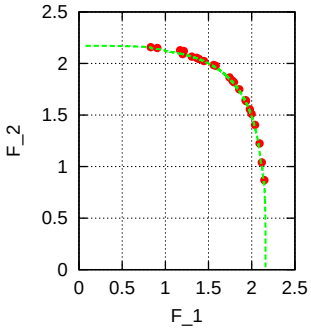

Figure 5: Two Objective Function: The exact Pareto front (dashed line) and that computed using the surrogate model (marks).

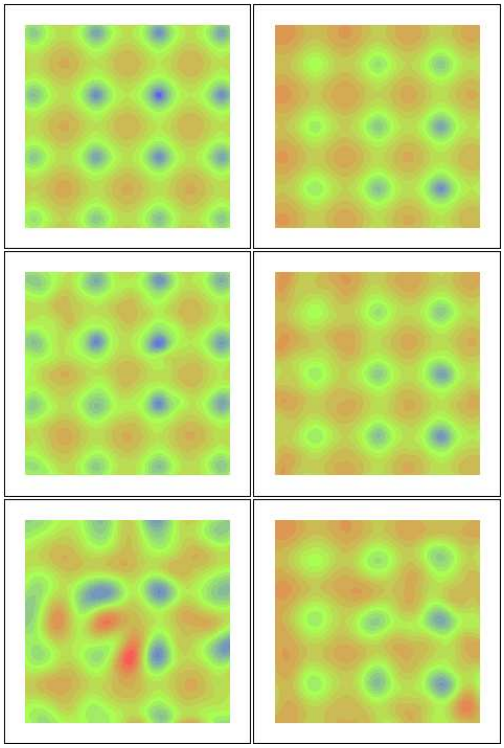

Figure 6: Two Objective Function: Response surfaces for the first (left column) and second (right column) objective functions. The exact shapes (top), the ones computed using the new RBF network (mid) and thoses produced using a conventional RBF network (bottom) are shown.

Inverse Design of a $3D$ Peripheral Cascade

The last problem is concerned with the inverse design of a peripheral compressor cascade, based on given pressure distributions along its surfaces. The analysis tool was a finite–volume solution method for the $3D$ compressible, inviscid flow equations. A similar technique was used to numerically solve the adjoint equations, yielding thus the sensitivity derivatives required for the network training. The isentropic Mach number at outlet hub was equal to $M_{2,is} = 0.4$ and the peripheral and radial inlet flow angles were $a_{per} = 58^o, a_{radial} =$

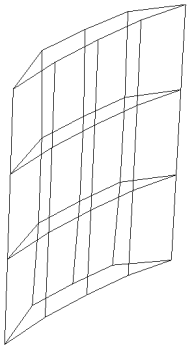

Figure 7: 3D Peripheral Cascade: Parameterization of the pressure and suction blade surfaces using Bezier surfaces. A total number of $5 \times 4 = 20$ control points were defined on each surface.

0^o.

The blade pressure and suction sides were parameterized using two Bezier surfaces, which met along the leading and training edges. Each side of the blade (pressure and suction), was parameterized using 20 control points, 4 of them to the radial direction, times 5 to the axial one. fig. 7. Among them, 14 were fixed and only 6 points per blade side were allowed to vary. So, the total number of degrees of freedom was 12.

The Mach number distribution over the blade surfaces is shown in fig.8. This was the preset target (though the target was expressed in terms of pressure coefficient rather than Mach number) but it is also too close to the corresponding distribution over the computed optimal blade.

The starting surrogate model is built upon the responses and gradients for 100 random training patterns. The optimization process converged within six cycles to a blade geometry that reproduces the target pressure distribution. The convergence history is shown in fig. 9. The CPU cost for this test case was $100 + 6$ direct evaluations of the flow field and the same number of adjoint evaluations, corresponding to 212 Euler calls.

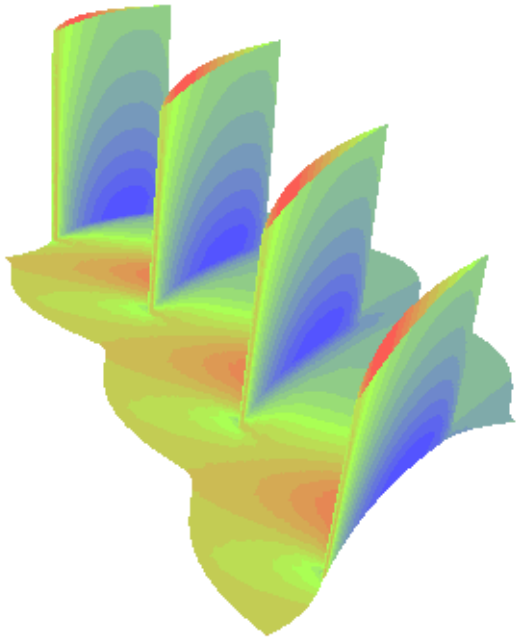

Figure 8: 3D Peripheral Cascade: Mach number distribution over the blade surfaces and the hub.

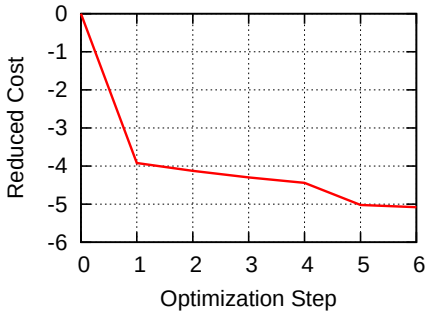

Figure 9: 3D Peripheral Cascade: Convergence history.

CONCLUSIONS

A new variant of *EA*–based optimization, using a new *RBF* network, was proposed.

The method seems promising for use in optimization problems with expensive fitness evaluations. The concept of the new method is to allow a direct and an adjoint flow solver to compute objective function values and gradients at a number of candidate solutions which will then be used to build a dependable surrogate evaluation model for the evolutionary search. Compared to conventional *RBF* networks, the new network has a number of additional tuning parameters. The new model yields systematically better predictions than those of the conventional model. It, thus, ensures that the optimization algorithm may converge to the global optimal solution within a limited number of cycles, each of which involves the *RBF* network retraining.

For the design of optimal aerodynamic shapes, the adjoint formulation is used to compute objective function derivatives. To extend this method in application areas other than those supported by *CFD* tools, a similar method that is capable of computing gradients, with reasonable computing cost, should be devised.

The training set is updated at the end of each cycle, prior to building the new *RBF* model. Each time, the training patterns are redefined through (a) adding new patterns (the most recent "optimal" solution should be among them; but, other points selected in less explored areas should be added) so as to protect the network from overfitting and the whole algorithm from being trapped into local stationary points, (b) eliminating some patterns, by employing distance–based criteria. The redefinition of the training set is crucial and research in this area is ongoing.

ACKNOWLEDGMENT
The second author was supported by grant from the Program Thales of the National Technical University of Athens

References

[1] K.C. Giannakoglou. A design method for turbine blades using genetic algorithms on parallel computers. ECCOMAS 98, John Wiley & Sons, 1998.

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

[3] 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, Miettinen K. et al (Eds.), John Wiley & Sons, Jyvaskyla, Finland, 1999.

[4] A.P. Giotis, K.C. Giannakoglou, and P. Periaux. A reduced cost multi-objective optimization method based on the pareto front technique, neural networks and PVM. ECCOMAS 2000, Barcelona, 2000.

[5] A.P. Giotis, M. Emmerich, B. Naujoks, K.C. Giannakoglou, and T. Bäck. Low-cost stochastic optimization for engineering applications. EUROGEN 2001 - Evolutionary Methods for Design, Optimisation and Control with Applications to Industrial Problems, K. Giannakoglou et. al. (Eds.), Athens, Greece, Sept 2001.

[6] K.C. Giannakoglou, A.P. Giotis, and M.K. Karakasis. Low–cost genetic optimization based on inexact pre–evaluations and the sensitivity analysis of design parameters. *Journal of Inverse Problems in Engineering*, 9:389–412, 2001.

[7] M. Karakasis, A.P. Giotis, and K.C. Giannakoglou. Inexact information aided, low-cost, distributed genetic algorithms for aerodynamic shape optimization. *Int. J. for Numerical Methods in Fluids*, 2004 (to appear).

[8] M. Emmerich, A. Giotis, M. Ozdemir, T. Bäck, and K.C. Giannakoglou. Metamodel–assisted evolution strategies. PPSN *VII*, Granada, Spain, 2002.

[9] K.C. Giannakoglou. Design of optimal aerodynamic shapes using stochastic optimization methods and computational intelligence. *Progress in Aerospace Sciences*, 38:43–76, 2002.

[10] A. Jameson. Essential elements of computational algorithms for aerodynamic analysis and design. NASA/CR-97-206268, ICASE Report No.97-68, 1997.