Extended abstract of a paper submitted for review toward presentation at the AIAA 21st Applied Aerodynamics Conference, Orlando, FL, June 23-26, 2003.

AERODYNAMIC SHAPE OPTIMIZATION USING HYBRIDIZED DIFFERENTIAL EVOLUTION

Nateri K. Madavan*

NASA Ames Research Center, Moffett Field, California

ABSTRACT

An aerodynamic shape optimization method that uses an evolutionary algorithm known at Differential Evolution (DE) in conjuction with various hybridization strategies is described. DE is a simple and robust evolutionary strategy that has been proven effective in determining the global optimum for several difficult optimization problems. Various hybridization strategies for DE are explored, including the use of neural networks as well as traditional local search methods. A Navier-Stokes solver is used to evaluate the various intermediate designs and provide inputs to the hybrid DE optimizer. The method is implemented on distributed parallel computers so that new designs can be obtained within reasonable turnaround times. Results are presented for the inverse design of a turbine airfoil from a modern jet engine. (The final paper will include at least one other aerodynamic design application). The capability of the method to search large design spaces and obtain the optimal airfoils in an automatic fashion is demonstrated.

INTRODUCTION

Remarkable progress has been made in recent years in the ability to design airfoil shapes that are optimal with regard to certain desired characteristics. This progress has been achieved by combining improved methods for the simulation of complicated flow fields with efficient numerical optimization techniques and by harnessing the powerful capabilities of modern computers. Both steady and unsteady Navier-Stokes and Euler solvers have been combined with various traditional optimization techniques (gradient-based methods,[1,2] response surfaces, etc.) to obtain optimal airfoil designs.

More recently, there has been considerable interest in the development of airfoil shape optimization techniques that are based on nontraditional approaches such as evolutionary algorithms and neural networks. Various approaces based on neural networks,[3,4] neural networks in conjunction with response surfaces,[5,6,7,8,9] genetic algorithms,[10,11,12] and genetic algorithms in conjunction with neural networks,[13,14] among others, have been reported in the literature. These techniques offer several advantages over traditional optimization methods. The references cited above primarily deal with airfoil shape optimization for turbomachinery applications. A complete review of the literature and the comparative merits of all these different optimization techniques is beyond the scope of this article. The reader is refered to the references cited as the starting point for a more exhaustive literature search.

This paper deals with an aerodynamic shape optimization method that uses an evolutionary algorithm known at Differential Evolution.[15] DE is a simple and robust evolutionary strategy that has been proven effective in determining the global optimum for several difficult optimization problems.[16] Its application in aeronautics, however, has been rather limited. It has been used in the predictive control of aircraft dynamics.[17] It has been used in conjunction with a potential flow solver in the inverse design of turbomachinery airfoils;[18] the same authors have also presented a hybridized version[19] that combines DE with a local direct simplex search method to minimize the number of objective function evaluations using the potential flow solver.

While population-based approaches such as DE are robust and capable of locating the global optimum in difficult objective function landscapes, they often suffer from slow convergence once in the vicinity of the global optimum and require many objective function evaluations. This limits their applicability in aerodynamic design optimization where the objective function evaluations typically are obtained from high-fidelity simulations (e.g., Navier-Stokes simulations) that are computationally expensive. This paper deals with hybridization strategies that use rapidly convergent, but less robust, local optimization methods in conjunction with the global and robust DE algorithm to effectively reduce overall computing time requirements in aerodynamic shape optimization

* Research Scientist, NASA Advanced Supercomputing Division. Senior Member, AIAA.

Various hybridization strategies for DE are explored, including the use of neural networks as well as traditional local search methods. A Navier-Stokes solver is used to evaluate the various intermediate designs and provide inputs to the hybrid DE optimizer. One hybridization strategy that is explored uses a neural network that is trained on the computational fluid dynamics (CFD) simulation data and acts as a surrogate for the Navier-Stokes solver in the objective function evaluations. Other strategies incorporate various local search methods and will be described in the final paper. The method is implemented on distributed parallel computers so that new designs can be obtained within reasonable turnaround times. Results are presented for the inverse design of a turbine airfoil from a modern jet engine. (The final paper will include at least one other aerodynamic design application). The capability of the method to search large design spaces and obtain the optimal airfoils in an automatic fashion is demonstrated.

DESIGN OPTIMIZATION METHOD

Differential Evolution

Differential Evolution is an evolutionary strategy (ES) developed for single-objective optimization in continuous search spaces. It is conceptually simple and possesses good convergence properties that have been demonstrated in a variety of applications.[15] Details of the algorithm can be found elsewhere;[15,20] only its main features are summarized here.

The approach uses a population P that contains m n-dimensional real-valued parameter vectors, where n is the number of parameters or decision variables:

$$P = \{\bar{x}_1, \ldots, \bar{x}_m\}$$

The population is usually initialized at generation $g = 0$ in a random fashion:

$$P(0) = \{\bar{x}_1(0), \ldots, \bar{x}_m(0)\}, \quad g = 0$$

The population size m is maintained constant throughout the optimization process. Differential evolution is thus similar to a (μ, λ) ES[21] with μ and λ equal to m.[22] The method however differs from standard ES approaches in several respects as described below.

As with all ES-based approaches, mutation is the key ingredient of differential evolution. The basic idea is to generate new parameter vectors for the subsequent generation by using weighted differences between two (or more) parameter vectors selected randomly from the current population to provide appropriately scaled perturbations that modify another parameter vector (or, comparison vector) selected from the same population. This idea has been implemented in various forms but the form discussed and used here is the classical implementation where new trial parameter vectors $\{\bar{y}_1, \ldots, \bar{y}_m\}$ for the next generation $g + 1$ are generated according to the following mutation scheme:

For $l = 1, m$; $i = 1, n$ generate

$$y_l^i = x^i_{\alpha_1^l}(g) + F.\left(x^i_{\alpha_2^l}(g) - x^i_{\alpha_3^l}(g)\right)$$

In the above α_1^l, α_2^l, α_3^l are distinct elements of $\{1, 2, \ldots, m\}$ randomly selected for each l, and $F \in [0,2]$ is a parameter that controls the amplification of the differential variation. Other variants that either use the difference between more than two parameter vectors or keep track of the best parameter vector at each generation and use it in the mutation scheme have also been developed[15] and used with varying success in specific applications.

DE is similar to other recombinative ES approaches in that it also uses discrete recombination. The strategy adopted in differential evolution is to modify the trial parameter vectors $\{\bar{y}_1, \ldots, \bar{y}_m\}$ to generate parameter vectors $\{\bar{z}_1, \ldots, \bar{z}_m\}$ as follows:

For $l = 1, m$; $i = 1, n$ generate

$$z_l^i = \begin{cases} y_l^i \text{ with probability } & p_c \\ x_l^i(g) \text{ with probability } & 1 - p_c \end{cases}$$

where p_c is a parameter that controls the proportion of perturbed elements in the new population. Note that the mutation and recombination operations described above can lead to new vectors that may fall outside the boundaries of the variables. Various repair rules can be used to ensure that these inadmissible vectors do not enter the population. A simple strategy, which is the one adopted here, is to delete these inadmissible vectors and form new ones until the population is filled.

The selection scheme used in DE is deterministic but differs from methods usually employed in standard ES approaches. Selection is based on local competition only, with the modified trial parameter vector competing against one population member (the comparison vector) and the survivor entering the new population $g + 1$ as follows:

For $l = 1, m$

$$\bar{x}_l(g+1) = \begin{cases} \bar{z}_l & \text{if} \quad f(z_l) < f(x_l(g)) \\ \bar{x}_l(g) & \text{else} \end{cases}$$

where f denotes the corresponding objective function (or fitness) value. This greedy selection criterion results in fast convergence; the adaptive nature of the mutation operator, in general, helps safeguard against premature convergence and allows the process to extricate itself from local optima. The generation counter is incre-

mented and the process is repeated until some stopping criteria are satisfied.

Hybridization Strategies

1. Using Neural Networks

While the DE algorithm is quite efficient and effective in exploring the entire design space in search of the optimal solution, the algorithm has a tendency to slow down as it approaches the vicinity of the optimal solution. Many subsequent function evaluations are then required to obtain the exact optimum. The computational effort required by these function evaluations in the vicinity of the optimum can be nearly eliminated using a hybrid approach that combines the DE algorithm with a neural network that is trained on the CFD simulation data.[23] The trained neural network is then used to evaluate the objective function with negligible computational effort and expense instead of using the CFD solver. While hybridization is always useful, it must be done with caution. Here the neural network is used only in the latter stages after the entire population has evolved to the general vicinity of the optimal solution. Thus the neural network is used as a "local" response surface with validity only in a small region of the design space. This makes it easier to train the neural network and improves its generalization abilities. For the inverse shape optimization problem, the neural network is trained on the sum-of-squares error between the actual pressure computed by the CFD solver and the target pressure at various points on the airfoil. A three-layer feed-forward neural network as shown in Fig. 1 is used here.

2. Using Local Search Methods

The final paper will include results using hybridization strategies based on local search methods.

Constraint Handling

For use in aerodynamic design we have incorporated an efficient constraint handling mechanism into the DE algorithm. It is a parameter-less approach that helps steer the algorithm away from infeasible regions of the design space. Physical constraints (e.g. maximum airfoil thickness) are imposed, as well as aerodynamic constraints (e.g. wavy surfaces). Airfoil geometries that do not violate the constraints but for which the CFD solver fails to converge are also deemed infeasible and treated accordingly by the constraint handling mechanism.

Airfoil Geometry Parametrization

Geometry parameterization and prudent selection of design variables are critical aspects of any shape optimization procedure. Since this study focuses on airfoil redesign, the ability to represent various airfoil geometries with a common set of geometrical parameters is essential. Variations of the airfoil geometry can be obtained then by smoothly varying these parameters. Geometrical constraints imposed for various reasons, such as structural, aerodynamic (e.g., to eliminate flow separation), etc., should be included in this parametric representation as much as possible. Additionally, the smallest number of parameters should be used to represent the family of airfoils.

The airfoil geometry parametrization method described in Rai and Madavan[7] is used here. A total of 13 geometric parameters were used to define the airfoil geometry in the current study. These parameters are listed below (see Fig. 2 for illustration):

1. Leading edge and trailing edge airfoil metal angles (2 parameters).
2. Eccentricity of upper leading edge ellipse (1 parameter).
3. Angles defining the extent of the leading edge ellipses (2 parameters).
4. Semi-minor axes values at the leading edge (2 parameters).
5. Angles defining the extent of the trailing edge circle (2 parameters).
6. Airfoil y-coordinate values at about 50% chord on the upper and lower surfaces (2 parameters).
7. Airfoil y-coordinate values at about 75% chord on the upper surface (1 parameter).
8. Airfoil stagger angle (1 parameter).

This method of generating the airfoil surface provided the necessary variations in airfoil geometry required by the optimization procedure.

CFD Simulation Methodology

A Navier-Stokes solver was used to perform the flow simulations (direct function evaluations) that serve as inputs to the optimization process. The solver used is a modified version of the ROTOR-2 computer code[24] and solves the two-dimensional, Navier-Stokes equations around a single airfoil in a cascade (with spanwise periodic boundary conditions) for a given set of inlet and exit conditions. Multiple grids are used to discretize the flow domain; an inner "0" grid that contains the airfoil and an outer "H" grid that conforms to the external boundaries as shown in Fig. 3.

Figure 3 also shows the grid system used to discretize the flow domain. Each airfoil has two grids associated with it: an inner "O" grid that contains the airfoil and an outer "H" grid that conforms to the external boundaries. For the analyses performed here, each inner O grid has 151 points in the circumferential direction and 41 points

in the wall-normal direction. Each outer H grid has 101 points in the axial direction and 41 points in the transverse direction. For the sake of clarity, only some of the grid points are shown in Fig. 3.

The dependent variables are initialized to freestream values and the equations of motion are then integrated subject to the boundary conditions. The flow parameters that are specified are the pressure ratio across the turbine (ratio of exit static pressure to inlet total pressure), inlet temperature and flow angle, flow coefficient, and unit Reynolds number based on inlet conditions.

Design Objective Formulation

Various design objectives can be incorporated in the current procedure depending on the optimization problem being solved. For the inverse turbine airfoil design shown here, the design objective function was formulated The design objective function was formulated as the equally-weighted sum-of-squares error between the target and actual pressure obtained during the optimization process at various locations on the airfoil.

Implementation on Distributed Parallel Computers

In order to reduce overall design time, the procedure has been implemented on a distributed parallel computer. The results in this article were obtained on the SGI Origin 3000 and the Cray SV1 distributed parallel computers at NASA Ames Research Center. Parallel implementation of the method is quite straightforward and relies on the simultaneous computation of multiple, independent aerodynamic simulations on separate processors. A script-based procedure is used that invokes a variable number of processors depending on processor availability and the size of the population used. The number of processors can also be adjusted as the design proceeds. The current setup is based on a "master-slave" arrangement, with the master handling the tasks of setting up the simulations, neural network training for the hybrid method, and farming out of the aerodynamic computations to the other "slave" processors. Since the aerodynamic computations are independent of each other, no communication between the processors is required until the computations are completed. The slave processors then communicate their results to the master which then performs the necessary calculations to determine the members of the next population.

RESULTS

The design method was used in the inverse design of a turbine airfoil with a specified pressure distribution. The target pressure distribution was obtained at the midspan of a turbine vane from a modern Pratt and Whitney jet engine. Several flow and geometry parameters were also supplied and used in the design process. The design objective function was formulated as the equally-weighted sum-of-squares error between the target and actual pressure obtained during the optimization process at 45 locations on the airfoil. Some of these results have been presented earlier.[23]

The initial design space was chosen to be quite large to allow a wide range of airfoil shapes to be explored. A sampling of some of the initial airfoil geometries is shown in Fig. 4. In order to hold the CFD function evaluations to a reasonable number, 6 (instead of 13) design variables were used in the initial stages of the design. The population of 50 members were then evolved using the DE algorithm.

The DE algorithm without any hybridization strategy was considered first. Figure 5 shows the pressure distribution for the optimal airfoil that represents the best airfoil obtained after 13 generations using the DE optimization method with 6 design parameters. The algorithm is able to approach the target distribution within about 650 function evaluations. Note that in the early stages of evolution several of the airfoil geometries were infeasible and hence were not evaluated by the CFD solver. After 13 generations, the number of design variables is increased to 13 and the population evolved further. The optimal pressure distribution shown in Fig. 5 was obtained after an additional 13 generations and agrees well with the target distribution. The airfoil geometries corresponding to the optimal designs with both 6 and 13 design parameters are shown in Fig. 6.

The convergence history of the baseline DE algorithm[23] is shown in Fig. 7 which plots the population mean and variance as a function of the number of generations. Both the mean and the variance decrease with generation number except for the slight increase corresponding to the switch from 6 to 13 design parameters. The figure also shows that convergence of the algorithm is slower in the vicinity of the optimal solution. This is typical of many other evolutionary algorithms and highlights the need for a hybrid approach that combines the global search and exploration capabilities of the DE algorithm with other algorithms that can converge rapidly to an optimum when initialized in the local neighborhood. As described earlier, a different hybrid approach to reducing overall design time was adopted here in the DE-NN method where the populations after the first few generations (using 13 design parameters) were used to train a neural network. Figure 8 shows the envelope of pressure data around the target pressure distribution that was used to train the neural network. Note, however, that the network was trained directly on the sum-square-error and not on the

individual pressure data. The data envelope in Fig. 8 represents the variation of the pressure distributions for the range of airfoil geometries in the neural network training set and is meant to convey the local nature of the neural network response surface in the vicinity of the optimal solution. The pressure distribution for the optimal airfoil design obtained by the DE-NN approach is shown in Fig. 9 and compares well with the target and optimal DE design pressure distributions.

FINAL PAPER

Some preliminary results using one hybridization strategy for the DE algorithm are presented. The final paper will include results using alternative strategies. Additional demonstration computations illustrating the use of the method in direct (not inverse) design are also underway.

REFERENCES

[1]Lee, S. Y., and Kim, K. Y., "Design Optimization of Axial Flow Compressor Blades with a Three-Dimensional Navier-Stokes Solver,: Proceedings of the ASME Turbo Expo 2000, Munich, Germany, May 8-11, 2000.

[2]Janus, J. M., and Newman, J. C., "Aerodynamic and Thermal Design Optimization for Turbine Airfois," AIAA Paper No. 2000-0840, 39th AIAA Aerospace Science Meeting and Exhibit, Reno, NV, Jan. 8-11, 2001.

[3]Rai, M. M., "A Rapid Aerodynamic Design Procedure Based on Artificial Neural Networks," AIAA Paper No. 2001-0315, Jan. 2001.

[4]Pierret, S., and Braembussche, R. A. V., "Turbomachinery Blade Design Using a Navier-Stokes Solver and Artificial Neural Network," Journal of Turbomachinery, Vol 121, No. 4, pp. 326-332, 1999.

[5]Papila, N., Shyy, W., Griffin, L. W., and Dorney, D. J., "Shape Optimization of Supersonic Turbines Using Response Surface and Neural Network Methods," AIAA Paper No. 2001-1065, Jan. 2001.

[6]Rai, M. M., and Madavan, N. K., "Application of Artificial Neural Networks to the Design of Turbomachinery Airfoils," AIAA Journal of Propulsion and Power, Vol. 17, No. 1, pp. 176-183, Jan. 2001.

[7]Rai, M. M., and Madavan, N. K., "Aerodynamic Design Using Neural Networks," AIAA Journal, Vol. 38, No. 1, pp. 173-182, Jan. 2000.

[8]Madavan, N. K., Rai, M. M., and Huber, F. W., "Neural Net-Based Redesign of a Gas Generator Turbine for Improved Unsteady Aerodynamic Performance," AIAA Journal of Propulsion and Power, to appear. Also, AIAA Paper No. 99-2522, 35th AIAA/ ASME/SAE/ASEE Joint Propulsion Conference and Exhibit, Los Angeles, CA, Jun. 20-24, 1999.

[9]Rai, M. M., Madavan, N. K., and Huber, F. W., "Improving the Unsteady Aerodynamic Performance of Transonic Turbines Using Neural Networks," AIAA Paper No. 2000-0169, 38thAerospace Sciences Meeting and Exhibit, Reno, NV, Jan. 10-13, 2000.

[10]Obayashi, S., and Takanashi, S., "Genetic Optimization of Target Pressure Distributions for Inverse Design Methods," AIAA Journal, Vol. 34, No. 5, pp. 881-886, 1996.

[11]Dennis, B. H., Egorov, I. N., Han, Z.-X., Dulikravich, G. S., and Poloni, C., "Multi-Objective Optimization of Turbomachinery Cascades for Minimum Loss, Maximum Loading, and Maximum Gap-to-Chord Ratio," AIAA Paper No. 2000-4876, Sept. 2000.

[12]Quagliarella, D., and Cioppa, A. D., "Genetic Algorithms Applied to the Aerodynamic Design of Transonic Airfoils," AIAA Paper 94-1896-CP, 1994.

[13]Uelschen, M., and Lawerenz, M., "Design of Axial Compressor Airfoils with Artificial Neural Networks and Genetic Algorithms," AIAA Paper No. 2000-2546, Fluids 2000, Denver, CO, Jun. 19-22, 2000.

[14]Poloni, C., Giurgevich, A., Onesti, L., and Pediroda, V., "Hybridization of a Multi-Objective Genetic Algorithm, a Neural Network, and a Classical Optimizer for a Complex Design Problem in Fluid Dynamics," Comp. Meth. Appl. Mech. Engr., Vol. 186, pp. 403-420, 2000.

[15]Storn, R. and Price, K., "Differential Evolution - A Simple Evolution Strategy for Fast Optimization," Dr. Dobb's Journal, Vol. 22, No. 4, pp. 18-24, April 1997.

[16]J. Lampinen: A Bibliography of Differential Evolution Algorithm. Technical Report. Lappeenranta University of Technology, Department of Information Technology, Laboratory of Information Processing, Lappeenranta, Finland (2001).

[17]Nho, K., and Agarwal, R. K., "Fuzzy Logic Model-Based Predictive Control of Aircraft Dynamics Using ANFIS," AIAA Paper 2001-0316, Jan., 2001.

[18]Rogalsky, T., Derksen, R.W. and Kocabiyik, S., "Differential Evolution in Aerodynamic Optimization," Canadian Aeronautics and Space Institute Journal, Vol. 46, No. 4, pp. 183-190, Dec. 2000.

[19]Rogalsky, T. and Derksen, R.W., "Hybridization of Differential Evolution for Aerodynamic Design," Proceedings of the 8th Annual Conference of the Computational Fluid Dynamics Society of Canada, pp. 729-736, June 11-13, 2000.

[20]K. V. Price: 'Differential Evolution: A Fast and Simple Numerical Optimizer'. In: Biennial Conference of the North American Fuzzy Information Processing

Society, (NAFIPS), June 1996, ed. by M. Smith, M. Lee, J. Keller, J. Yen (IEEE Press, New York 1996) pp. 524--527.

[21]Back. T, Hammel, U., and Schwefel, H.-P., "Evolutionary Computation: Comments on the History and Current State," IEEE Trans. on Evolutionary Computation, Vol. 1, pp. 3-17, 1997.

[22]M. A. Shokrollahi, R. Storn: 'Design of Efficient Erasure Codes with Differential Evolution'. In: Proceedings of ISIT 2000, International Symposium on Information Theory, Sorrento, Italy, June 25-30, 2000.

[23]Madavan, N. K., "Turbomachinery Airfoil Design Optimization Using Differential Evolution," 2nd International Conference on Computational Fluid Dynamics, Sydney, Australia, Jul. 2002.

[24]Rai, M. M., and Madavan, N. K., "Multi-Airfoil Navier-Stokes Simulations of Turbine Rotor-Stator Interaction," ASME Journal of Turbomachinery, Vol. 112, pp. 167-190, Jul. 1990.

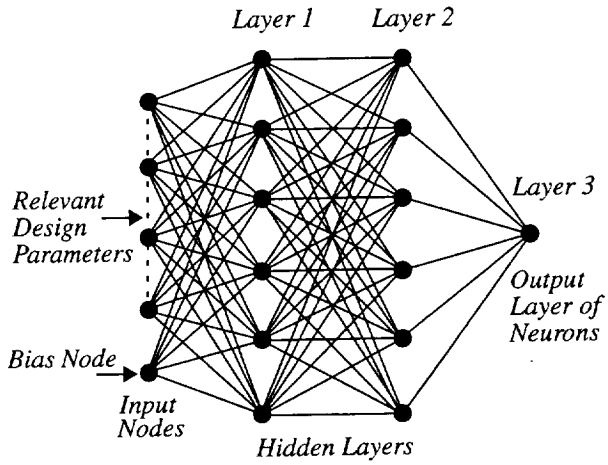

Figure 1. Schematic of the three-layer feed-forward neural network used in the hybrid DE-NN method.

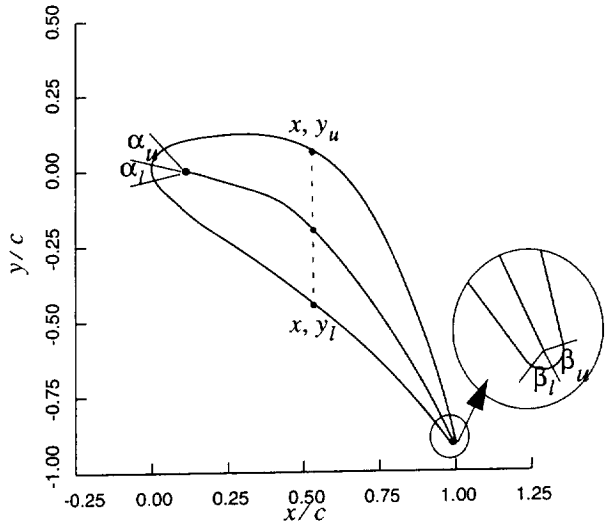

Figure 2. Schematic of a generic airfoil showing location of control points on the airfoil surface and the defining angles used in the parametrization of the airfoil geometry.

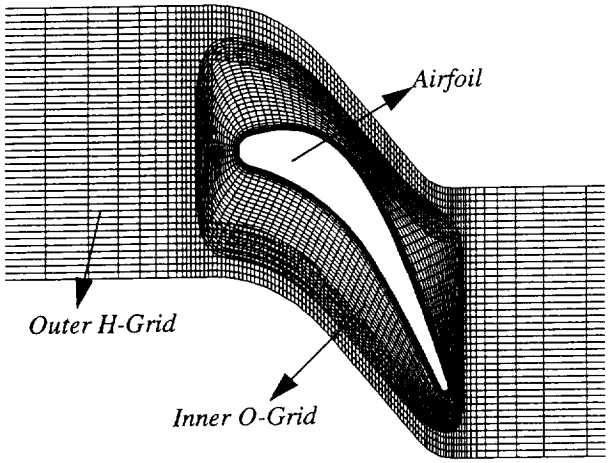

Figure 3. Representative turbine airfoil geometry and computational grid used in the CFD simulations.

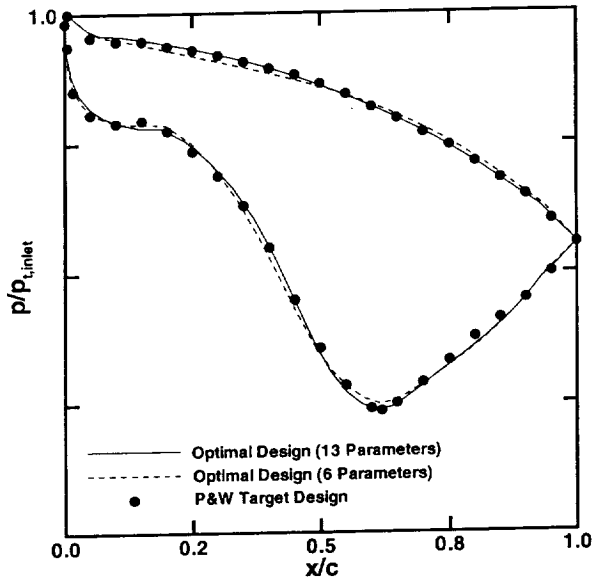

Figure 5. Airfoil pressure loading for the optimal design.

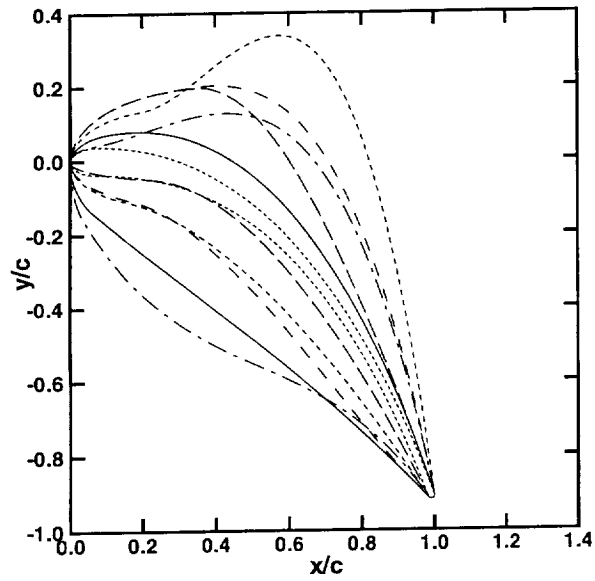

Figure 4. Sampling of initial airfoil geometries.

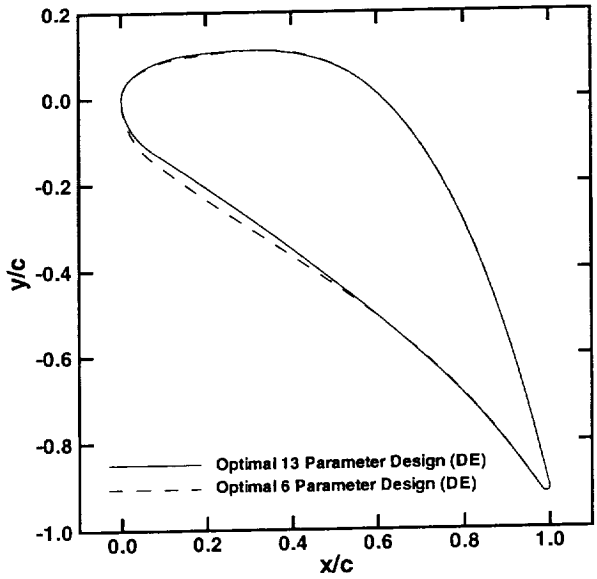

Figure 6. Airfoil geometry for the optimal design.

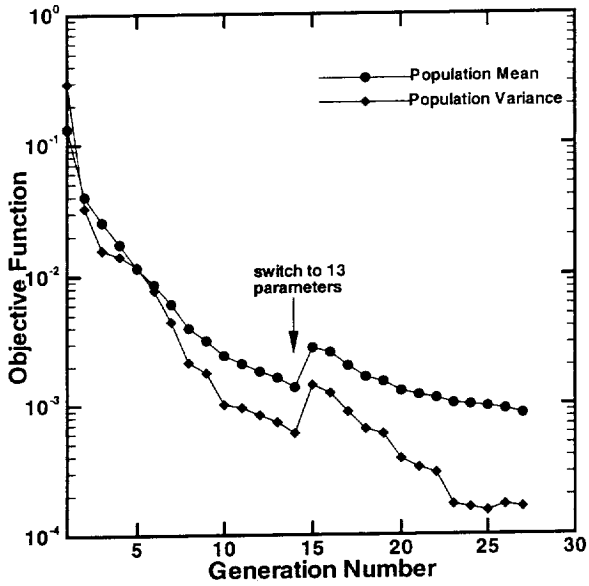

Figure 7. Convergence history for the baseline DE algorithm (no hybridization strategy included).

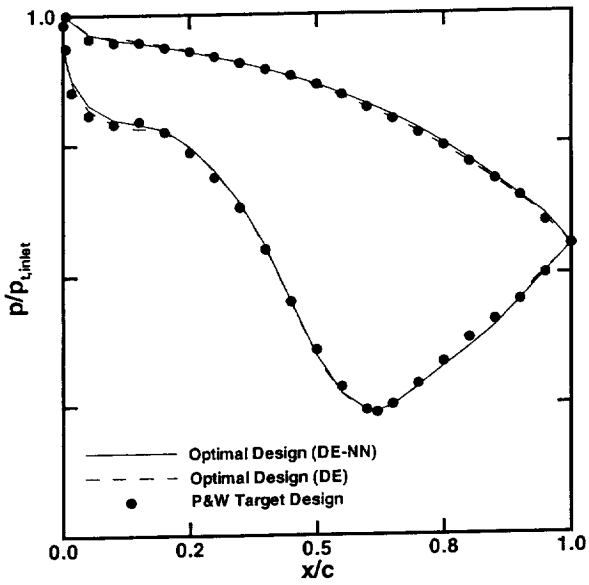

Figure 9. Airfoil pressure loading for the optimal design using the hybrid DE-NN approach.

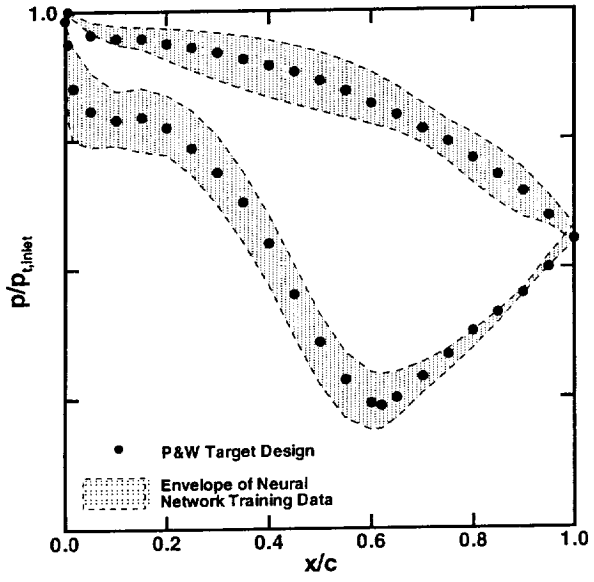

Figure 8. Envelope of airfoil loadings used to train the neural network for the hybrid DE-NN algorithm.