Aerospace Science and Technology 23 (2012) 330–344

Contents lists available at SciVerse ScienceDirect

Aerospace Science and Technology

www.elsevier.com/locate/aescte

Neural networks based airfoil generation for a given C_p using Bezier–PARSEC parameterization

Athar Kharal, Ayman Saleem [*]

National University of Sciences and Technology (NUST), Islamabad 44000, Pakistan

ARTICLE INFO

Article history:
Received 3 June 2011
Received in revised form 22 August 2011
Accepted 29 August 2011
Available online 10 September 2011

Keywords:
Neural networks
Airfoil design
Aerodynamic optimization
Airfoil parameterization
Genetic algorithms
Bezier curves
Airfoil profiles
Bezier–PARSEC

ABSTRACT

Determining the airfoil geometry from a given C_p-distribution is an inverse problem of paramount importance specially in the context of variable geometry aerodynamic platforms. This work describes the implementation of artificial neural nets for the airfoil geometry determination. Instead of using full coordinates of the airfoil, Bezier–PARSEC 3434 parameters have been used to describe an airfoil. Some of these parameters have been determined using a Genetic Algorithm. In the second stage C_p-distribution in terms of c_l, c_d and c_m for 10 angles of attack has been input into three different neural nets for learning and then estimating the corresponding BP3434 parameters. Feed-forward backpropagation, Generalized regression and Radial basis neural nets have been trained and then compared in terms of performance and regression statistics. The work establishes the superiority of feed-forward backpropagation neural nets. The result is partly due to good function approximation properties of the neural architecture and partly due to the use of Bezier–PARSEC 3434 parameterization scheme.

1. Introduction

The airfoil, in many aspects, is the heart of the airplane [9]; through its pressure distribution it affects the stall speed, turning performance and cruise speed. The actual pressure distribution over an airfoil is governed by the non-linear Navier–Stokes equations.

In many cases, known airfoils described by a vector of 50–80 length as listed in the UIUC Airfoil Coordinates Database [1] and in [2], are selected which most nearly approximate the required lift-drag specification. This approach does not always yield efficient performance and sometimes even restricts the airplane's performance due to airfoil's intrinsic aerodynamic limitations. Reduced degrees-of-freedom are required for faster and efficient optimization of airfoils but such reduction of DoF results in computational error of shape as well as of the C_p-distribution. Such a vicious circle of error propagation is critical in almost all cases specially in scenarios like Structure Fluid Interaction (SFI). Ideally, the process of airfoil shape determination should be reversed. The airfoil should be tuned to the airplanes' performance demands to enhance the aircraft's efficiency. To this end different techniques have been reported in literature. Techniques of soft computing e.g. fuzzy logic, neural networks, genetic algorithms and simulated annealing have been implemented due to their robustness in approximating non-linear functions. Such techniques have been used either isolatedly or in tandem with others.

In this work, for a given C_p-distribution, three different neural architectures have been used to generate an optimized airfoil shape. Organization of the paper is as follows: Section 2 briefly reviews the relevant literature. Section 4 presents the Bezier PARSEC parametrization in detail. Next Section 5 gives the scheme of our work. Results and discussion are presented in Section 6. Finally Section 7 concludes the work and presents possible future directions to extend this research.

2. Related work

Many researchers explored different possibilities to exploit the benefits of soft computing to the advantage of aerodynamics. By far Neural Nets combined with Fuzzy Inference Systems and/or Genetic Algorithms has been the most active approaches. This section briefly reviews the relevant literature.

In [7] Ju addressed an optimization design problem and the corresponding numerical method for wind turbine airfoils with the maximum lift-drag ratios at multiple working conditions as the optimal objective. The airfoil profile is parameterized by Bezier curves and the design space for airfoil family is specified by the full factorial design of experiment method. The aerodynamic parameters for each sample airfoil have been calculated by the computational fluid dynamics method. The optimal problem is numerically solved

[*] Corresponding author. Tel.: +92 342 855 3259.
E-mail addresses: atharkharal@gmail.com (A. Kharal), aiman1173@hotmail.com (A. Saleem).

1270-9638/$ – see front matter © 2011 Elsevier Masson SAS. All rights reserved.
doi:10.1016/j.ast.2011.08.010

by combining the genetic algorithm with the artificial neural network model. The work demonstrated that the optimized airfoil has better aerodynamic performance.

Venkataraman [14] proposed the use of four curves to define an airfoil: two for the upper surface and two for the lower surface. Derksen and Rogalsky [6] have presented a slightly different method than Venkataraman [14] for airfoil parameterization. This method is an extension of the Bezier parameterization used in [10] and is an amalgamation of PARSEC and Bezier parameterization. Using these curves the method generates the Bezier control points for a few airfoils and the results are compared with respect to their accuracy and convergence speed using Differential Evolution. The method fits a wide variety of airfoils and has resulted in accelerated convergence for aerodynamic optimization.

Shahrokhi and Jahangirian in [11] took account of various characteristics of viscous transonic flow particularly around the trailing edge. The work then applied to airfoil shape optimization at high Reynolds number turbulent flow conditions using a Genetic Algorithm. An unstructured grid Navier–Stokes flow solver with a two-equation K–ε turbulence model has been used to evaluate the fitness function. The aerodynamic characteristics of the optimum airfoil obtained from the proposed parametric method have been compared with those from alternative methods. They concluded that the new method is capable of finding efficient and optimum airfoils in fewer number of generations.

Bellman et al. [5] implements a GA for shape optimization to maximize the lift of low Reynold number airfoils for micro air vehicle (MAV) applications. It has been shown that the combined GA-ANN technique can be employed efficiently and accurately to produce globally optimal airfoil designs for a desired objective value. The authors have used Joukowski transformation [3] to define an airfoil in the zeta-plane. Thus, the design space is limited to Joukowski airfoils only. The ANN used in this work had only three inputs and one output.

Rai et al. [8] incorporated the advantages of traditional response surface methodology into neural networks. The procedure employs a strategy called parameter-based partitioning of the design space and uses a sequence of response surfaces based on both neural networks and polynomial fits to traverse the design space in search of the optimal solution. This results in a response surfaces that has both the power of neural networks and the economy of low-order polynomials (in terms of number of simulations needed and network training requirements). The results obtained are closer to the target design than those obtained using an earlier method with only three design variables. The capability of the method in transforming generic shapes, such as simple curved plates, into optimal airfoils has also been reported.

A number of numerical experiments have been reported in [15], in which different Bezier parameterization of an airfoil have been constructed and compared. The authors have concluded that the quality of a Bezier fit depends on the way the abscissas of the control points are prescribed. The Bezier curve is improved by a regularization technique. Finally, the work reported in [4] first defines a target pressure distribution. An ANN is trained using data from several flows in the vicinity of the target pressure distribution. The pressure distribution closest to the target pressure distribution is selected.

3. Preliminaries

3.1. Panel method

Panel methods are used to evaluate the pressure field around an object in a flow field. In panel methods, the surface of the body is discretized into a series of segments or panels. The velocity potential integral equations are then applied to each panel instead of the complete surface. This reduces the integral equations into a set of simultaneous algebraic equations.

Compared to the classical thin airfoil theory, the panel methods are more suited to aerodynamic analysis of airfoils with considerable thickness and at high angles of attack. Thus, these methods are in between, the thin airfoil theory solution and a rigorous Navier–Stokes flow solver, in terms of accuracy and robustness. This is one of the reasons for the wide applicability of panel methods.

Hess and Smith, introduced a technique for implementation of Panel Method in 1966, which is now a classical one. The technique is based upon the distribution of sources and vortices on the airfoil surface and the assumption that the vortex strength remains constant over the entire aircraft and its value is fixed by the Kutta condition. However, the source strength is varied from panel to panel to satisfy the flow tangency boundary condition everywhere. Moreover these conditions are satisfied at the mid-points of the nodes. Though this formulation slightly affects the airfoil surface geometry, it is, however, simple to implement and yields reasonably accurate results for a fair number of panels with the panels concentrated more on the leading- and trailing-edges. Now this Panel Method based solver generates the tangential velocity (at the node mid-points) and since the normal velocity component is zero, the pressure coefficient at the mid-points is simply evaluated using:

$$C_p = 1 - \frac{V_t^2}{V_\infty^2}$$

where; $V_t^2 =$ tangential velocity at node mid-points
$V_\infty^2 =$ free-stream velocity

It must be noted that if the design space for airfoil C_p-distribution is limited to incompressible flow the results by Panel Method are considered to be reasonably acceptable.

3.2. Artificial neural networks

Artificial neural networks are computational models based on parallel processing and having the ability to adapt. A neural net essentially comprises a set of processing units called neurons. For every unit there is a state of activation y_k, which is equivalent to the output of the unit. Connection between different neurons is defined by a weight w_{jk} which determines the effect that the signal of neuron j has on neuron k. For every neuron an external input θ_k is also present. A propagation rule determines the effective input s_k of a neuron from its external inputs. Most frequently used form of the propagation rule is given as:

$$s_k(t) = \sum_j w_{jk}(t) y_j(t) + \theta_k(t)$$

An activation function F_k is given which determines the new level of activation based on the effective input $s_k(t)$ (see Fig. 1):

$$y_k(t) = F_k\big(s_k(t)\big)$$

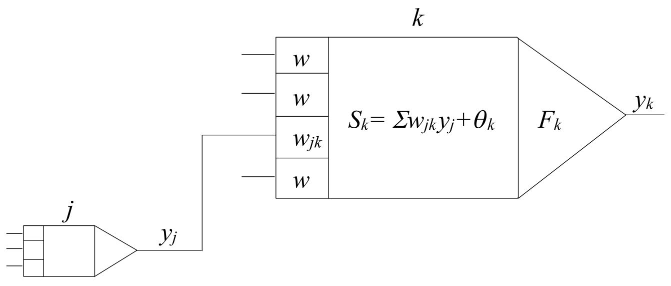


Fig. 1. Net-structure of an ANN.

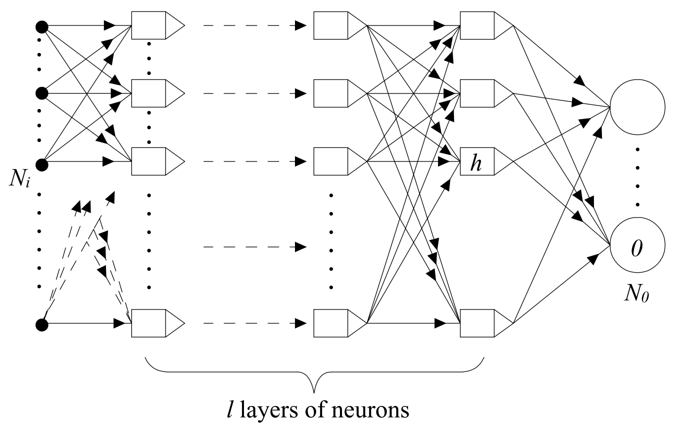


Fig. 2. A multi-layer feed-forward ANN with l layers of neurons.

Thus each unit performs a relatively simple job: receive input from neighbors or external sources and use this to compute an output signal which is propagated to other units. Apart from this processing, a second task is the adjustment of the weights. Learning rule is the method for information gathering about the error made by the neural net as a whole and consequently adjusting the value of individual weights w_{jk}. The system is inherently parallel in the sense that many units can carry out their computations at the same time.

Although activation functions are not restricted to nondecreasing functions. Generally, some sort of threshold function is used e.g. a hard limiting threshold function (a sgn function), a linear or semi-linear function, or a smoothly limiting threshold e.g. sigmoid function given as:

$$F(s_k) = \frac{1}{1+e^{-s_k}}$$

Many neurons may be collected and connected in parallel. Such an assembly of neurons is termed as 'layer'.

3.2.1. *Feed-forward backpropagation neural net*

A feed-forward network has a layered structure. Each layer consists of units which receive their input from units from a layer directly below and send their output to units in a layer directly above the unit. There are no connections within a layer. The N_i inputs are fed into the first layer of $N_{h,1}$ hidden units. No processing takes place in input units. The output of the hidden units is distributed over the next layer of $N_{h,2}$ hidden units, until the last layer of hidden units, of which the outputs are fed into a layer of N_o output units (see Fig. 2).

Backpropagation is a learning rule. It involves two phases: During the first phase the input x is presented and propagated forward through the network to compute the output values y_o^p for each output unit in pth layer. This output is compared with its desired value d_o, resulting in an error signal δ_o^p for each output unit. The second phase involves a backward pass through the network during which the error signal is passed to each unit in the network and appropriate weight changes are calculated.

Although backpropagation can be applied to networks with any number of layers, it has been shown through the Universal Approximation Theorem that only one layer of hidden units suffices to approximate any function with finitely many discontinuities to arbitrary precision, provided the activation functions of the hidden units are non-linear. In most applications a feed-forward network with a single layer of hidden units is used with a sigmoid activation function for the units.

3.2.2. *Radial basis neural net*

For a single neuron, the propagation rule of a radial basis neuron is given as:

$$s_k(t) = \| w_{jk}(t) - y_j(t) \| \theta_k(t)$$

and the activation function is given as

$$y_k(t) = \widetilde{F}_k\big(s_k(t)\big) = e^{-(s_k(t))^2}$$

Each neuron's weighted input is the distance between the input vector and its weight vector and net input is the element-by-element product of its weighted input with its bias. Each neuron's output is its net input passed through $\widetilde{F}_k$.

The neurons in RBF networks have localized receptive fields because they only respond to inputs that are close to their centers. This is in contrast to the standard multi-layer networks, where the sigmoid function creates a global response. The RBF network trains faster than other multi-layer networks, but requires many neurons for high-dimensional input spaces.

Main advantage of RBF networks is the phenomenally less time taken for their building and learning. They are also credible for moderately difficult regression or classification problems.

3.2.3. *Generalized regression neural net*

General Regression Neural Networks (GRNN) is one of the type of neural networks with a one pass learning algorithm. GRNNs are used basically for estimation of continuous variables, as in standard regression techniques and do not require an iterative procedure. These nets are similar to the radial basis network, but have a peculiar linear second layer. The net is based upon the technique of kernel regression. GRNNs provide estimates of continuous variables and converge to the underlying (linear or non-linear) regression surface. GRNN do not require an iterative training, instead they are one pass learning algorithms with highly parallel structure.

The main advantages of GRNN are fast learning and convergence to the optimal regression surface as the number of samples becomes very large. GRNNs have been shown to perform well in noisy environments given enough data. As the training set size becomes large, the estimation error approaches zero. GRNN is particularly advantageous with sparse data, because the regression surface is instantly defined every where, even with just one sample.

3.3. *Genetic algorithms (GAs)*

Genetic algorithm is a method for solving both constrained and unconstrained optimization problems that is based on Natural Selection. A GA repeatedly modifies a population of individual solutions. At each step, the genetic algorithm selects individuals at random from the current population to be parents and uses them to produce the children for the next generation. Over successive generations, the population "evolves" towards an optimal solution. For a typical computing process, a population of individuals is created and initialized randomly at first. Each individual comprises of the three key operation variables. The evolution operations of selection, crossover and mutation are employed in the evolving process. The fitness of each individual in the population is estimated by a suitable fitness function. The fitness of each individual indicates the living ability of itself in the evolution process. The genetic algorithm uses three main types of rules at each step to create the next generation from the current population:

- Selection rules select the individuals, called parents, that contribute to the population at the next generation.
- Crossover rules combine two parents to form children for the next generation.

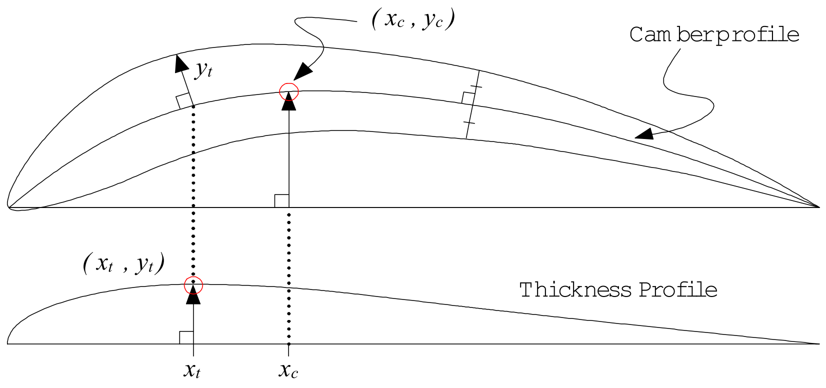


Fig. 3. Airfoil shape generation by superposition of the half thickness profile normal to the camber curve.

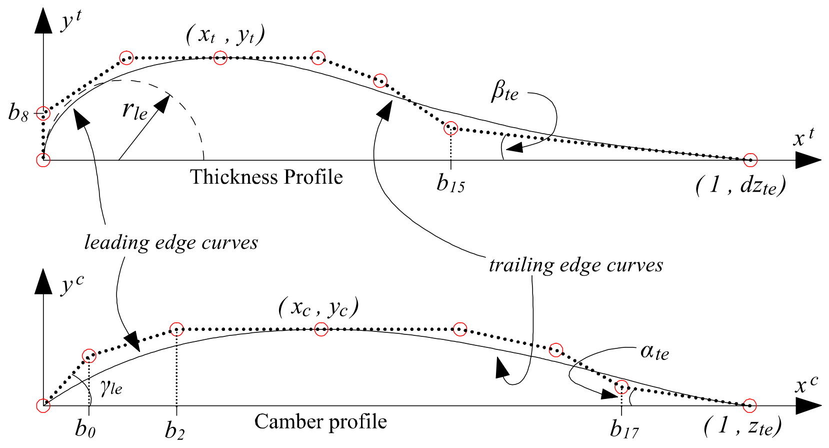


Fig. 4. Thickness and camber curves as defined by BP3434 parameterization.

- Mutation rules apply random changes to individual parents to form children.

The genetic algorithm differs from a classical, derivative-based, optimization algorithm in two main ways: Classical Algorithm generates a single point at each iteration. The sequence of points approaches an optimal solution. Classical Algorithm selects the next point in the sequence by a deterministic computation. On the other hand GA generates a population of points at each iteration. The best point in the population approaches an optimal solution. GA selects the next population by computation which uses random number generators.

GAs are robust with initial values and parallel in the computation, which is a promising tool to search for the global optimization. GAs are applied to solve a variety of optimization problems that are not well suited for standard optimization algorithms, including problems in which the objective function is discontinuous, non-differentiable, stochastic, or highly non-linear.

3.4. PARSEC parameters

PARSEC parameterization is demonstrated in [13,12] by Sobieczky. This parameterization method is particularly interesting as it incorporates parameters which have a physical relevance to airfoil shape and flow.

3.5. Bezier curves

A Bezier curve of degree n is uniquely defined by $n+1$ vertex points of a polygon. These vertices are called the control points of the nth order Bezier curve. The general expression for an nth order Bezier curve is given below:

$$P(u) = \sum_{i=0}^{n} P_i \frac{n!}{i!(n-i)!} u^i (1-u)^{n-i}$$

where $P_i = i$th control point

The parameter u goes from 0 to 1; with 0 at the zeroth control point and unity at the nth control point.

A third order Bezier curve is given by:

$$x(u) = x_0(1-u)^3 + 3x_1 u(1-u)^2 + 3x_2 u^2(1-u) + x_3 u^3$$
$$y(u) = y_0(1-u)^3 + 3y_1 u(1-u)^2 + 3y_2 u^2(1-u) + y_3 u^3$$

A fourth order Bezier curve is parametrically given by:

$$x(u) = x_0(1-u)^4 + 4x_1 u(1-u)^3 + 6x_2 u^2(1-u)^2 + 4x_3 u^3(1-u) + x_4 u^4$$
$$y(u) = y_0(1-u)^4 + 4y_1 u(1-u)^3 + 6y_2 u^2(1-u)^2 + 4y_3 u^3(1-u) + y_4 u^4$$

4. Bezier–PARSEC parameterization

Derksen and Rogalsky [6] have introduced the Bezier–PARSEC parameterization. Their approach is further subdivided into two parameterization methods viz. BP3333 and BP3434. In both the methods, Bezier control points are determined in terms of the PARSEC parameters of an airfoil (see Figs. 3 and 4).

4.1. BP3333

In the BP3333 parameterization, four third order Bezier curves are used to represent an airfoil; two for the camber shape and

Table 1
Thickness profile ($0 < b_8 < \min(y_t, \sqrt{\frac{-2r_{le}x_t}{3}})$).

Leading edge		Trailing edge	
$x_0 = 0$	$y_0 = 0$	$x_0 = x_t$	$y_0 = y_t$
$x_1 = 0$	$y_1 = b_8$	$x_1 = \frac{7x_t + \frac{9b_8^2}{2r_{le}}}{4}$	$y_1 = y_t$
$x_2 = \frac{-3b_8^2}{2r_{le}}$	$y_2 = y_t$	$x_2 = 3x_t + \frac{15b_8^2}{4r_{le}}$	$y_2 = \frac{1}{2}(y_t + b_8)$
$x_3 = x_t$	$y_3 = y_t$	$x_3 = b_{15}$	$y_3 = dz_{te} + (1 - b_{15})\tan(\beta_{te})$
		$x_4 = 1$	$y_4 = dz_{te}$

Table 2
Camber profile.

Leading edge		Trailing edge	
$x_0 = 0$	$y_0 = 0$	$x_0 = x_c$	$y_0 = y_c$
$x_1 = b_0$	$y_1 = b_0\tan(\gamma_{le})$	$x_1 = \frac{1}{2}(3x_c - y_c\cot(\gamma_{le}))$	$y_1 = y_c$
$x_2 = b_2$	$y_2 = y_c$	$x_2 = \frac{1}{6}(-8y_c\cot(\gamma_{le}) + 13x_c)$	$y_2 = \frac{5}{6}y_c$
$x_3 = x_c$	$y_3 = y_c$	$x_3 = b_{17}$	$y_3 = z_{te} + (1 - b_{17})\tan(\alpha_{te})$
		$x_4 = 1$	$y_4 = z_{te}$

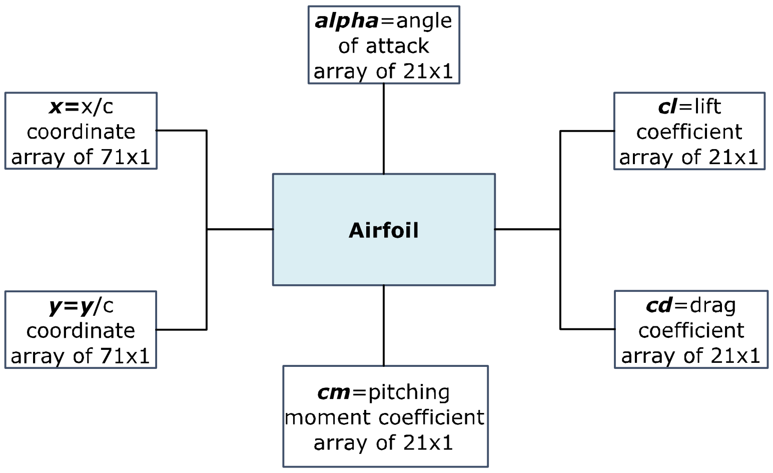

alpha=angle of attack array of 21x1
x=x/c coordinate array of 71x1
cl=lift coefficient array of 21x1
Airfoil
y=y/c coordinate array of 71x1
cd=drag coefficient array of 21x1
cm=pitching moment coefficient array of 21x1

Fig. 5. The structured array composition.

two for the thickness shape. The expressions for evaluating all the Bezier control points are given in terms of the 12 PARSEC parameters. As shown in [6] the advantages of BP3333 parametrization are several. Some are listed below:

1. The parametrization is more related to aerodynamic parameters of an airfoil since all the control points are defined by 12 PARSEC parameters.
2. Shape optimization is faster for BP3333 due to the relatively small number of parameters.
3. Continuity characteristics are far better as compared to Bezier parameterization.
4. As all the variables have a physical significance, constraints are easy to apply. This prevents the design process from significant deviation.
5. Sharp leading edges are easily avoided due to the leading edge radius parameter.

Disadvantage of BP3333 is that it has reduced degree of freedom, especially at the trailing edge. Thus, it is not able to parameterize many airfoils. Derksen in [6] has highlighted that BP3333 fails in representing airfoils which have a radical camber trailing edge curve, where the camber curve dips below the x-axis and where the trailing edge thickness is finite. The authors have also given a sample aerodynamic design problem to compare the convergence of BP3333 and Bezier parameterization for a given target C_p-distribution.

Table 3

Camber curve	Unknown Bezier control points
Leading edge Bezier curve of 3rd order	b_0, b_2
Trailing edge Bezier curve of 4th order	b_{17}
Thickness curve	
Leading edge Bezier curve of 3rd order	b_8
Trailing edge Bezier curve of 4th order	b_{15}, b_8

4.2. BP3434

In the BP3434 scheme, 10 PARSEC parameters and 5 Bezier parameters are used to define the four Bezier curves. In BP3434, third order Bezier curves are used to describe the airfoil camber- and thickness-leading edge curves while fourth order curves describe the shape of the airfoil camber- and thickness-trailing edge curves. Fourth order Bezier curves are used to increase the degree of freedom for airfoil parameterization at the trailing edge. For many airfoils, the trailing edge curve has a sudden cusp which is difficult to approximate by BP3333. Also, if the camber is negative for any portion of the chord length, then BP3434 outperforms BP3333 due to the better freedom at the trailing edge.

The curvature of the camber and thickness crests (κ_c and κ_t respectively) have been left out in BP3434 parameterization. By

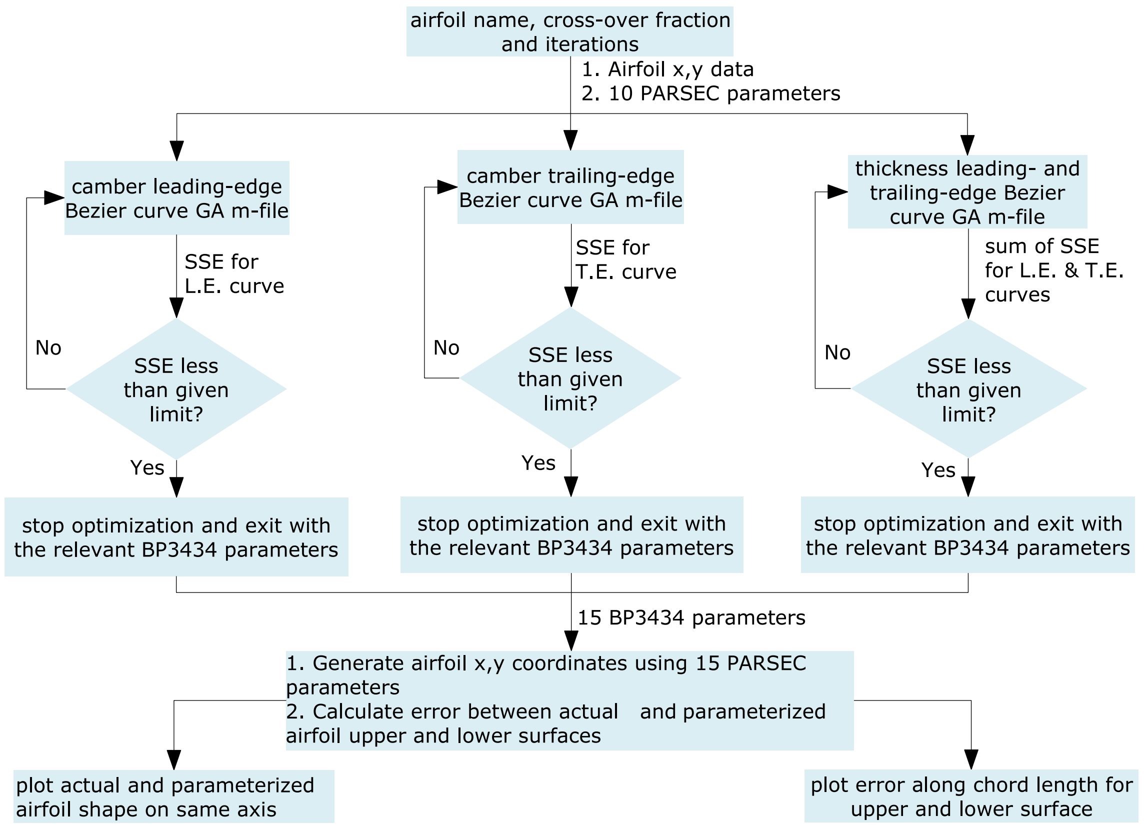

airfoil name, cross-over fraction and iterations
1. Airfoil x,y data
2. 10 PARSEC parameters
camber leading-edge Bezier curve GA m-file
SSE for L.E. curve
SSE less than given limit?
No
Yes
stop optimization and exit with the relevant BP3434 parameters
camber trailing-edge Bezier curve GA m-file
SSE for T.E. curve
SSE less than given limit?
No
Yes
stop optimization and exit with the relevant BP3434 parameters
thickness leading- and trailing-edge Bezier curve GA m-file
sum of SSE for L.E. & T.E. curves
SSE less than given limit?
No
Yes
stop optimization and exit with the relevant BP3434 parameters
15 BP3434 parameters
1. Generate airfoil x,y coordinates using 15 PARSEC parameters
2. Calculate error between actual and parameterized airfoil upper and lower surfaces
plot actual and parameterized airfoil shape on same axis
plot error along chord length for upper and lower surface

Fig. 6. BP3434 parameter determination using GA.

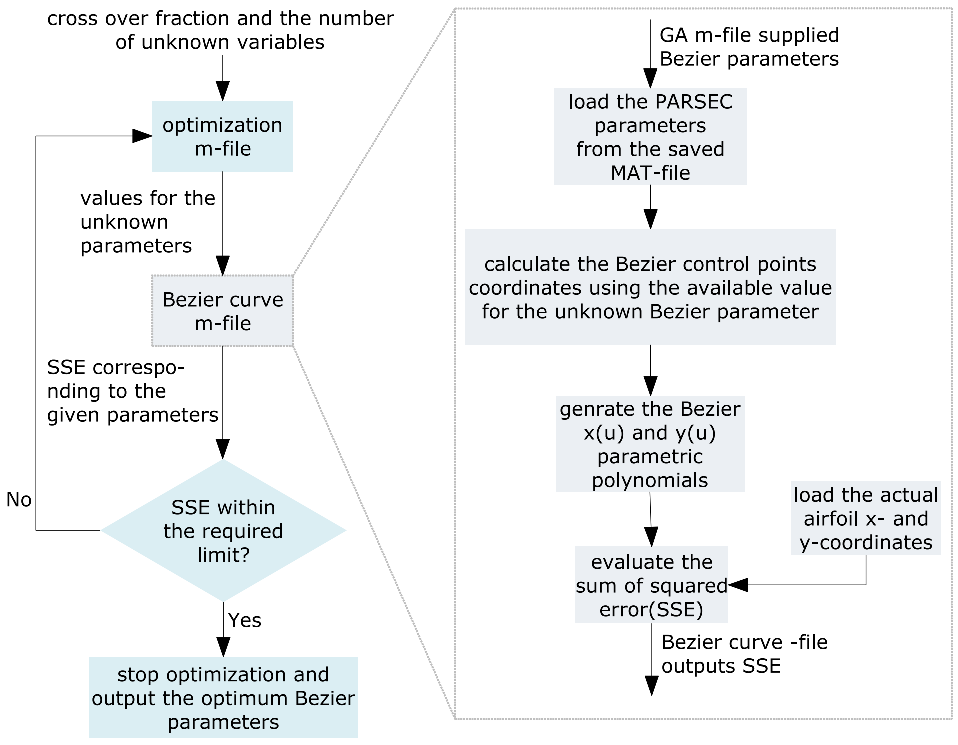

cross over fraction and the number of unknown variables
optimization m-file
values for the unknown parameters
Bezier curve m-file
SSE correspo-nding to the given parameters
SSE within the required limit?
No
Yes
stop optimization and output the optimum Bezier parameters
GA m-file supplied Bezier parameters
load the PARSEC parameters from the saved MAT-file
calculate the Bezier control points coordinates using the available value for the unknown Bezier parameter
genrate the Bezier x(u) and y(u) parametric polynomials
load the actual airfoil x- and y-coordinates
evaluate the sum of squared error(SSE)
Bezier curve -file outputs SSE

Fig. 7. GA optimization of unknown variables.

ignoring the crest curvature parameters and introducing 5 Bezier parameters, the relations for the Bezier control points are far more simple.

The control points for the leading and trailing edge Bezier curves of thickness and camber profiles are respectively given in Tables 1 and 2.

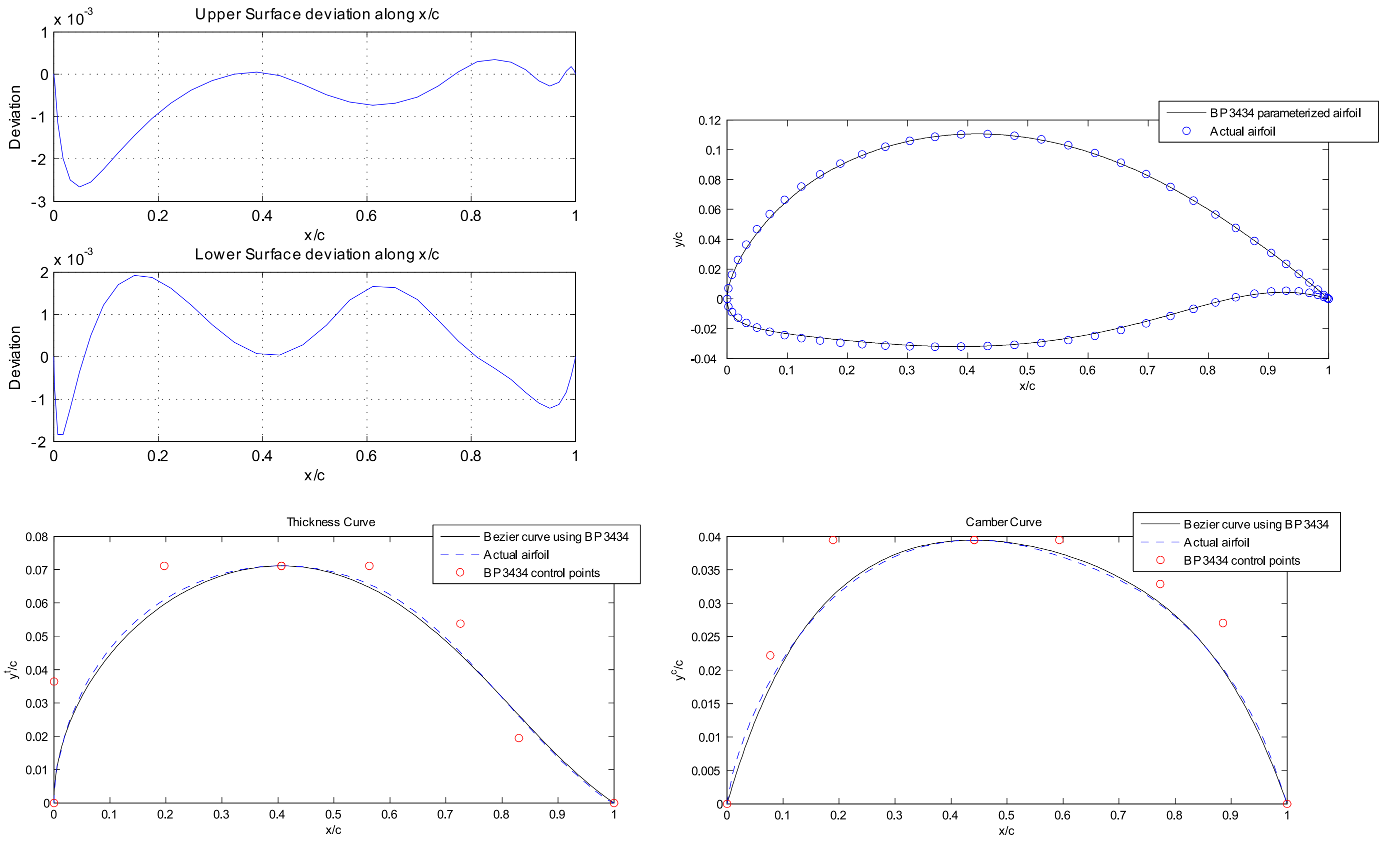


Fig. 8. Parameterization results for Eppler E433.

5. Scheme of main work

xy-coordinates of an airfoil are represented as a vector of 71 points. The vector is arranged as

$$\text{upper surface:}\quad \langle x_i, y_i\rangle;\quad i=1,\dots,36$$

$$\text{lower surface:}\quad \langle x_j, y_j\rangle;\quad j=36,\dots,71$$

where $x_i = x_j$

The 71 x- and y-coordinates are separated into the upper and lower surfaces. Here x_i moves from 1 to 0 for upper and lower surface, i.e. from trailing edge to leading edge. Thus, both the surfaces are defined by exactly the same x-values and only the y-values change.

The mean camber line is defined as the locus of all the points measured perpendicular to the chord line midway between the upper and lower surfaces. Therefore, y_c (camber curve y points) was obtained by adding the coordinates for the upper and lower surfaces and dividing by 2. The upper and lower y-values which are added and divided by 2 correspond to the same x-value. To non-dimensionalize the coordinates, the upper and lower xy-coordinates are divided by the chord length c. Symbolically, let (x_i, y_i); $i=1,\dots,71$ be the coordinates of an airfoil. Camber profile is given as (x_i^c, y_i^c), where

$$c = |x_1 - x_{36}|$$

$$y_i^u = \frac{y_i}{c} \quad\text{and}\quad y_j^l = \frac{y_j}{c} \quad \text{where } i=1,\dots,36,\ j=36,\dots,71$$

$$x_i^c = \frac{x_i}{c} \quad\text{and}\quad y_i^c = \frac{y_i^u + y_j^l}{2}$$

Similarly, the thickness curve is obtained. The thickness curve required is defined by Derksen and Rogalsky [10] as the half-thickness profile. The half thickness profile (x_i^t, y_i^t) is obtained by subtracting the camber curve from the upper surface i.e.

$$x_i^t = x_i^c \quad\text{and}\quad y_i^t = y_i^u - y_i^c$$

Two-dimensional aerodynamic analysis is to be carried out on the airfoil using the vectors of upper and lower surface values. But the solution time for a Navier–Stokes flow solver, even with assumptions of incompressible and inviscid flow, are on the order of magnitude higher than those for panel methods. Moreover, as C_p for a large number of airfoils, 500 in total, is required, time is an important factor. For example we discretized a NACA 6-series airfoil surface into 60 elements and solved using a Navier–Stokes solver assuming incompressible and inviscid flow at Reynolds number of 300 000, the solution convergence took more than 20 minutes. Whereas the same problem was solved using Panel Method in a fraction of time. Furthermore, since the design space for airfoil C_p-distribution is limited to incompressible flow therefore the results by Panel Method are reasonably acceptable. Hence we opted to use Panel Method with 60 panels, flow Reynold's number of 300 000 and assuming an incompressible flow.

This 2-D analysis provides lift coefficient (c_l), drag coefficient (c_d) and the quarter-chord pitching moment coefficient (c_m) at 10 angles of attack (AoA) viz. $1^\circ : 1 : 10^\circ$. For computer storage of the data, obtained so far, a structured array has been used due to the ease of referencing and accessing the different fields within the structure. The structured array is shown in Fig. 5 with the corresponding variables.

Table 4
Details for the three neural networks.

	FFBP	GRNN	RBF
Layers	2	2	2
Hidden layer	Tan-sigmoid	Radial basis	Radial basis
Output layer	Linear	Special linear	Special linear
Data standardization	zero mean 1 standard deviation	zero mean 1 standard deviation	zero mean standard deviation
PCA	99.99% retained	99.99% retained	99.99% retained
Data division	Training 71.4% Validation 14.3% Test 14.3%	Training 71.4% Validation 14.3% Test 14.3%	Training 71.4% Validation 14.3% Test 14.3%
Training algorithm	Levenberg–Marquardt	–	–

The required PARSEC parameters for BP3434 are:

Maximum thickness point $= (x_t, y_t)$
Maximum camber point $= (x_c, y_c)$
Trailing edge vertical displacement $= z_{te}$
Half thickness of the trailing edge $= dz_{te}$
Trailing camber line angle $= \alpha_{te}$
Trailing wedge angle $= \beta_{te}$
Leading edge direction $= \gamma_{le}$
Leading edge radius $= r_{le}$

We first obtain PARSEC parameters using the thickness and camber profiles. For this, $C^c(x_i)$ and $C^t(x_i)$ cubic spline interpolants are fitted to the camber and thickness curves, respectively.

For the maximum thickness point (x_t, y_t), the thickness cubic spline interpolant is searched from $x = 0$ to $x = 1$ for any first derivative stationary points. Once x_t is known, it is used to determine y_t-value using the thickness cubic interpolant evaluated at x_t. Symbolically

$$y_t = C^t\left(\left\{x_t \mid \left.\frac{dC^t}{dx_i}\right|_{x_i=x_t} = 0\right\}\right)$$

Similarly, the maximum camber point (x_c, y_c) is found:

$$y_c = C^c\left(\left\{x_c \mid \left.\frac{dC^c}{dx_i}\right|_{x_i=x_c} = 0\right\}\right)$$

The trailing edge vertical displacement (z_{te}) is calculated by evaluating the camber cubic spline at $x = 1$. The half-thickness of the trailing edge (dz_{te}) is found by evaluating the thickness cubic spline at $x = 1$.

$$z_{te} = C^c(x)|_{x=1} \quad \text{and} \quad dz_{te} = C^t(x)|_{x=1}$$

Trailing camber line angle (α_{te}) is the negative of the arctan of the slope of camber curve at $x = 1$. Similarly, the trailing wedge angle (β_{te}) is the negative of the $\tan^{-1}$ of the gradient of thickness cubic spline at $x = 1$. Both the quantities are negative due to the angle convention which is negative clockwise and positive anti-clockwise.

$$\alpha_{te} = -\tan^{-1}\left(\left.\frac{dC^c}{dx}\right|_{x=1}\right) \quad \text{and} \quad \beta_{te} = -\tan^{-1}\left(\left.\frac{dC^t}{dx}\right|_{x=1}\right)$$

Leading edge direction (γ_{le}) is simply the arctan of the slope of the camber curve at $x = 0$.

With the ten PARSEC parameters generated, the remaining parameters to be found are the five Bezier parameters viz. b_0, b_2, b_8, b_{15} and b_{17}. There is no definite formula for Bezier parameters; therefore, we evaluate them by curve fitting. The actual airfoil shape (in terms of x- and y-coordinates) is available, all we require is a set of parameterized Bezier curves with the correct five Bezier control points such that the sum-of-least-square error between the actual and parameterized curve is the smallest.

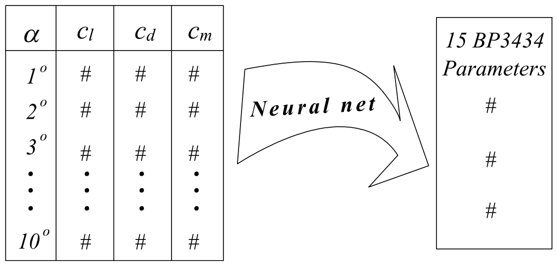


Fig. 9. Neural net implementation schema.

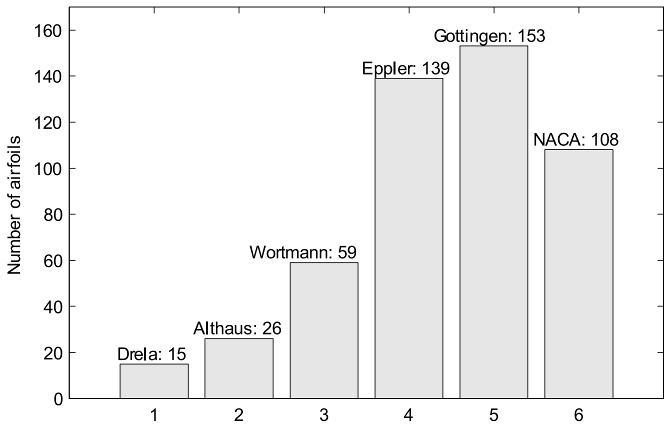


Fig. 10. Airfoil data composition.

Each of the four Bezier curves which form the airfoil shape have either one or two of the unknown Bezier control points stated above. The Bezier curves and the unknown points are shown in Table 3.

Note that b_8 is common to both the thickness leading and trailing edge curves.

Two approaches are available to determine the Bezier parameters. First approach uses the simplest method in which the unknown Bezier variable is varied from an appropriate initial value to its maximum possible value. The number of iterations or loops in this method depend on the accuracy desired for final result. For each iteration, depending upon the curve being considered, the squared error between the actual airfoil y-value, and the Bezier generated airfoil y-value is evaluated at 101 x-values. The squared errors are then summed and checked against a particular error limit. If the SSE (sum-of-squared-errors) is less than the acceptable limit then the particular Bezier value is extracted. Since some of the control points are very sensitive, a very small step size is required, typically 0.00001 or even lesser. Due to this, a very long computation time is required. Even though this approach is simple, the time and efficiency of the process are not appealing.

Second approach, which we actually adopted, uses Genetic Algorithms to find the most optimum value for the unknown Bezier curves. Four fitness functions are needed, one for each Bezier

Table 5
Test results for 200 airfoils

	FFBP	GRNN	RBF
Airfoils with MSE $< 1 \times 10^{-4}$	97	59	36
Airfoils with MSE $\geqslant 1 \times 10^{-4}$	103	141	164
MSE	4.552×10^{-4}	6.400×10^{-4}	1.070×10^{-2}

Table 6
Test results for 500 airfoils.

	FFBP	GRNN	RBF
Airfoils with MSE $< 1 \times 10^{-4}$	247	327	371
Airfoils with MSE $\geqslant 1 \times 10^{-4}$	253	173	129
MSE	1.820×10^{-4}	7.968×10^{-5}	1.158×10^{-4}

Table 7
Test results for 700 airfoils.

	FFBP	GRNN	RBF
Airfoils with MSE $< 1 \times 10^{-4}$	464	415	436
Airfoils with MSE $\geqslant 1 \times 10^{-4}$	236	285	264
MSE	4.155×10^{-4}	2.544×10^{-4}	3.946×10^{-4}

curve. Consider the leading-edge Bezier curve of the camber curve as an example to illustrate the process. The objective function is simply the difference between the actual airfoil shape and the Bezier generated airfoil shape. MATLAB's Genetic Algorithm (GA) toolbox has been used to minimize the SSE for each Bezier curve. A crossover fraction of 0.75 is used while the number of generations is set at 100. The default elite count of 2 is retained with a stochastic uniform selection function. The SSE is to be minimized with two variables viz. b_0 and b_2. Using the GA tool an m-file is generated for the curve. This script is used to obtain the values of b_0 and b_2 which minimize the SSE for the airfoil.

For the thickness curves one fitness function is constructed as both the curves (leading and trailing edge) depend on b_8. Apart from this difference, all the fitness functions for the curves are similar in working. Flow charts for the used method are shown in Figs. 6 and 7.

In BP3434 parameterization, only 15 variables are required to completely describe the airfoil geometry. After the BP3434 parameters are generated, they are used to generate the required airfoil shape. Also, the error between the actual airfoil and the parameterized airfoil is compared.

For this error to be calculated, the x-coordinates of the actual and parameterized airfoil must be the same. For example, to calculate the error at an x-value of 0.25, the y-value from the actual and parameterized airfoil must correspond to $x = 0.25$. To accomplish this uniformity in error calculation, the x-coordinates from the actual airfoil data are used, that is, the first 36 array values ranging from 1 to 0.

The issue is to generate the Bezier parameterized airfoil y-values at these x-coordinates. For this, the following technique is employed: For the camber curve, once the leading- and trailing-edge x- and y-values have been generated, they are combined together to form a single set of xy arrays which describe the camber curve. This was done by creating new x and y column vectors in which the first elements correspond to the leading-edge curve followed by the trailing edge elements. This vector of x- and y-values is then fitted with a cubic spline interpolant. The cubic spline is then evaluated at the 36 x-values available in the structured array for the actual airfoil data. The same procedure is applied for the thickness curve.

Making camber and thickness curves available with the same x-coordinates, the airfoil shape can be found. For the upper surface, the thickness curve is added to the camber curve. For the lower surface, the thickness curve is subtracted from the camber curve.

Table 8
Comparison of results for the three ANN.

FFBP	$\geqslant 1 \times 10^{-4}$	$< 1 \times 10^{-4}$
Known airfoils	50.6%	49.4%
Unknown airfoils	51.5%	48.5%
GRNN	$\geqslant 1 \times 10^{-4}$	$< 1 \times 10^{-4}$
Known airfoils	34.6%	65.4%
Unknown airfoils	70.5%	29.5%
RBF	$\geqslant 1 \times 10^{-4}$	$< 1 \times 10^{-4}$
Known airfoils	25.8%	74.2%
Unknown airfoils	82.0%	18.0%

The error for both the upper and lower surfaces is calculated by simply taking the difference between the y-values of the original and parameterized airfoil. This error is then plotted against the x-values from 0 to 1 for the upper and lower surfaces. The airfoil geometries, both actual and parameterized, are also plotted on the same axes for comparison. The parameterization results for Eppler E433 sailplane airfoil are shown in Fig. 8.

5.1. Neural network implementation

Three types of neural networks have been implemented: a feed-forward backpropagation neural network, generalized regression network and a radial basis network. The architectural specifications of the three networks are given in Table 4.

The schema for neural net implementation is shown in Fig. 9, i.e. inputs of the neural nets comprised a matrix of 10×4 dimension providing the values of c_l, c_d, c_m at 10 angles of attack for each airfoil. Target of the neural net is a vector of 15 BP3434 parameters. Training dataset comprises 500 heterogeneous airfoils and its composition is as shown in Fig. 10.

6. Discussion

For an inverse problem of airfoil determination like the present one, we have considered following aspects important:

1. the training data presented to the neural nets must cover a wide range of airfoils,
2. the trained network must yield reasonable results for a given C_p-distribution and
3. the number of parameters describing the airfoil shape must be kept to a minimum.

Such considerations have first of all affected the choice of parameterization scheme. The main shortcoming in BP3333, which

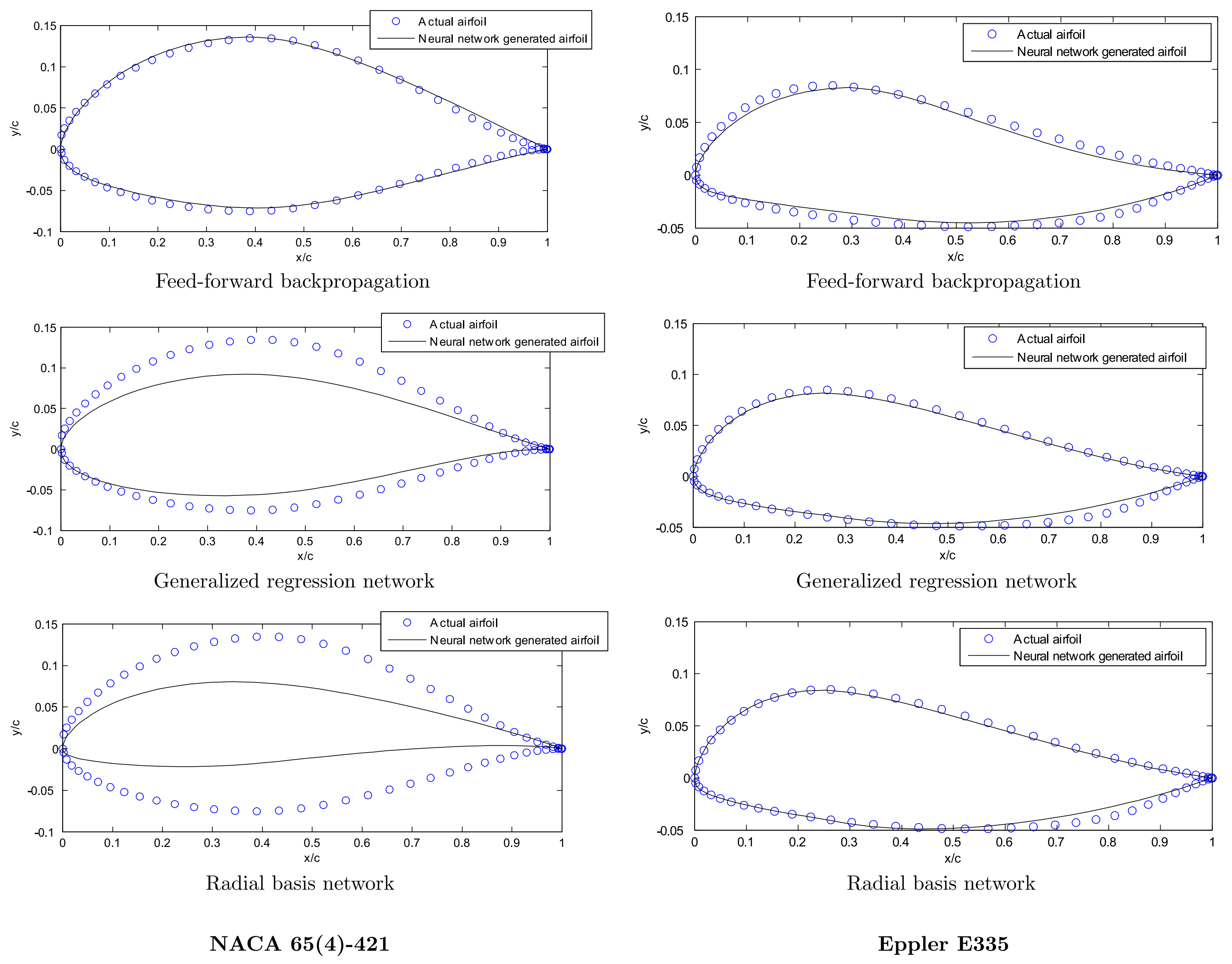


Fig. 11.

forced the use of BP3434 in our work is the lack of sufficient degree of freedom at the trailing edge. This prevents the parameterization of many airfoils. On the other hand, BP3434 has increased freedom to parameterize a given airfoil due to the fourth order Bezier curves at the trailing edges. Since this work is a reverse design process, significant number of airfoils are required to get acceptable results. Therefore, BP3434 was chosen so that as many airfoils as possible can be parameterized.

The trained neural nets were tested using 200 unknown-to-the-network airfoils. Results for these tests are shown in Table 5.

Test results against the 500 airfoils used for training (see Table 6).

Test results for total 700 (500 known + 200 unknown) airfoils are shown in Table 7.

These results may be further consolidated as shown in Table 8.

The three networks were tested for the 700 airfoils, with 500 of the airfoils included in the training set, the results again show the better performance of FFBP. Even though GRNN and radial basis show an improved performance with 700 airfoils. This is indicated by the reduced MSE and the increased fraction of airfoils with acceptable MSE value. The improved performance of GRNN and RB is due to the fact that 500 of the 700 test airfoils were used for training the networks. Thus, the network results for these 500 airfoils is obviously good. The performance of FFBP has remained almost the same as that for 200 airfoils tested earlier. This shows that FFBP network has adapted to the situation. As long as the input airfoil is within the learning domain of FFBP network, the network yields good results. This is demonstrated in the plot of NACA 65(4)-421 for the FFBP network.

The results by the three networks for an unknown-to-network airfoil, NACA 65(4)-421 airfoil, and an airfoil included in the training set, Eppler E335 flying wing airfoil, are shown in Fig. 11.

The plots for NACA 65(4)-421 and Eppler E335 airfoils clearly show that feed-forward network can be employed for this reverse design problem. The results of GRNN and radial basis depend on the similarity of the input data to the network training data. This is clearly shown for the NACA 65(4)-421 plot for GRNN and radial basis. This is also highlighted by the test results for 200 airfoils not included in the training data for the networks. The MSE for GRNN and radial basis is higher as compared to the FFBP network. Among the three networks, RBF net performs the worst with a high MSE of 1.07×10^{-2}.

The networks are analyzed in more detail through a post training regression analysis. The results for the networks for training, test and validation data is given below. In a regression analysis, the network output for known targets is compared. For an ideal network, the network outputs (Y) exactly match the target values (T). This is graphically illustrated by a straight line through the origin

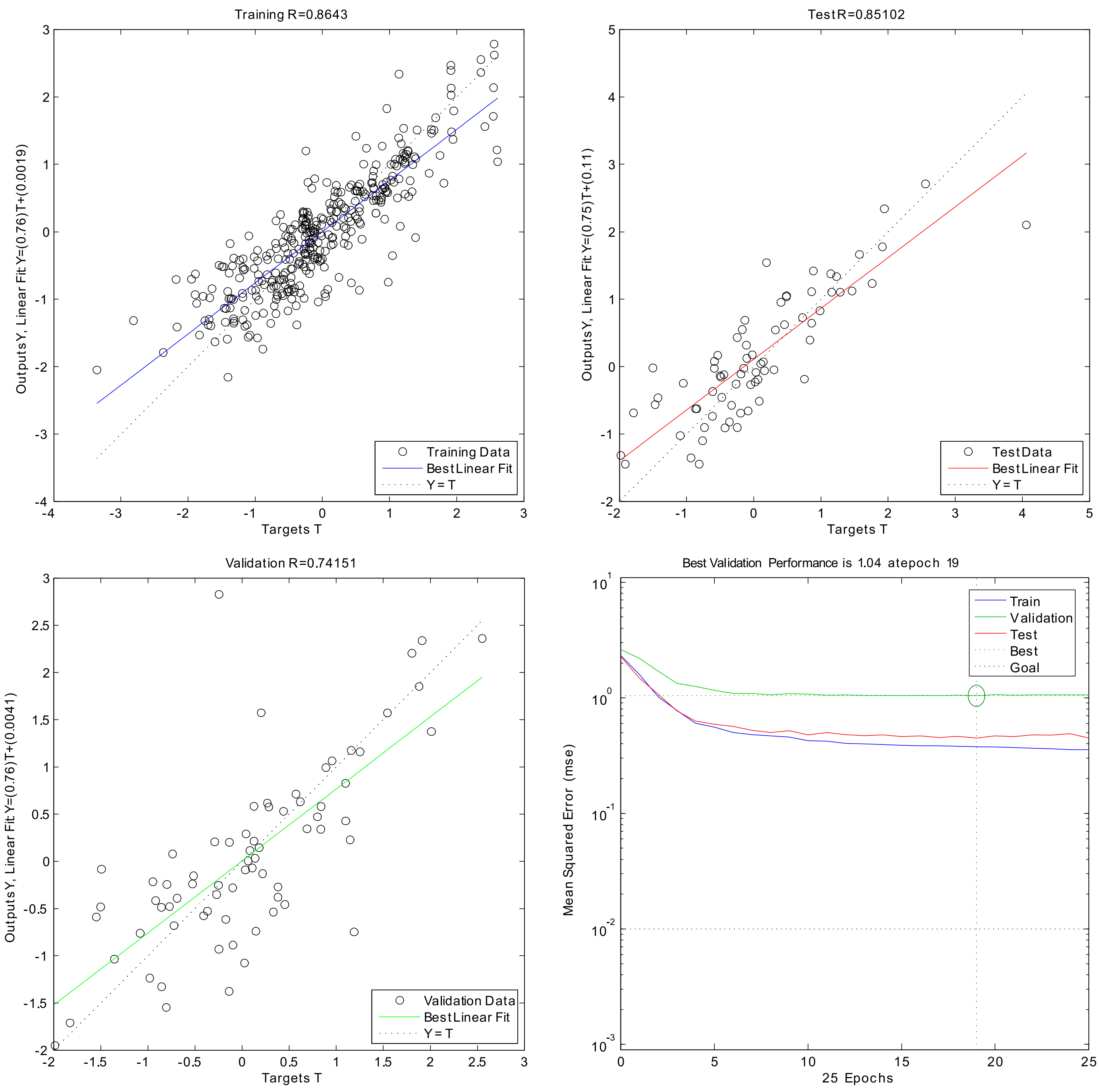


Fig. 12. Regression analysis of FFBP network.

with a unity slope on a plot of network outputs against target outputs (Y–T plot). This line is shown for comparison on Figs. 12–14 by a dotted line.

The performance of the feed-forward network is better than that of the generalized regression and radial basis networks as shown by the regression analysis of the test and validation data. This is highlighted by the highest regression values for the test and validation data for the FFBP network when compared with the other two networks. The training R-value for the FFBP is the lowest amongst the three networks, with the highest value for the radial-basis network. Both GRNN and radial basis networks have very good R-value for the training data. This can be attributed to their architecture. Both determine the distance between an input vector and the network weight vector. This distance is then multiplied element-by-element with the network bias vector in the linear output layer. Thus, an input close to the weight vector will produce an output, for the linear layer, close to unity, while an input different from the weight vector will yield a value close to zero. As the network weight and bias matrices are formed using the training data, both the networks give a very high R-value ($R > 0.90$) for the training data. However, for the test and validation data, the results are poor.

6.1. Performance of feed-forward backpropagation network for an Althaus and Gottingen airfoil

A 2D CFD analysis for two airfoils: Althaus AH93-W-215 and Gottingen 300, is performed using the commercial CFD software

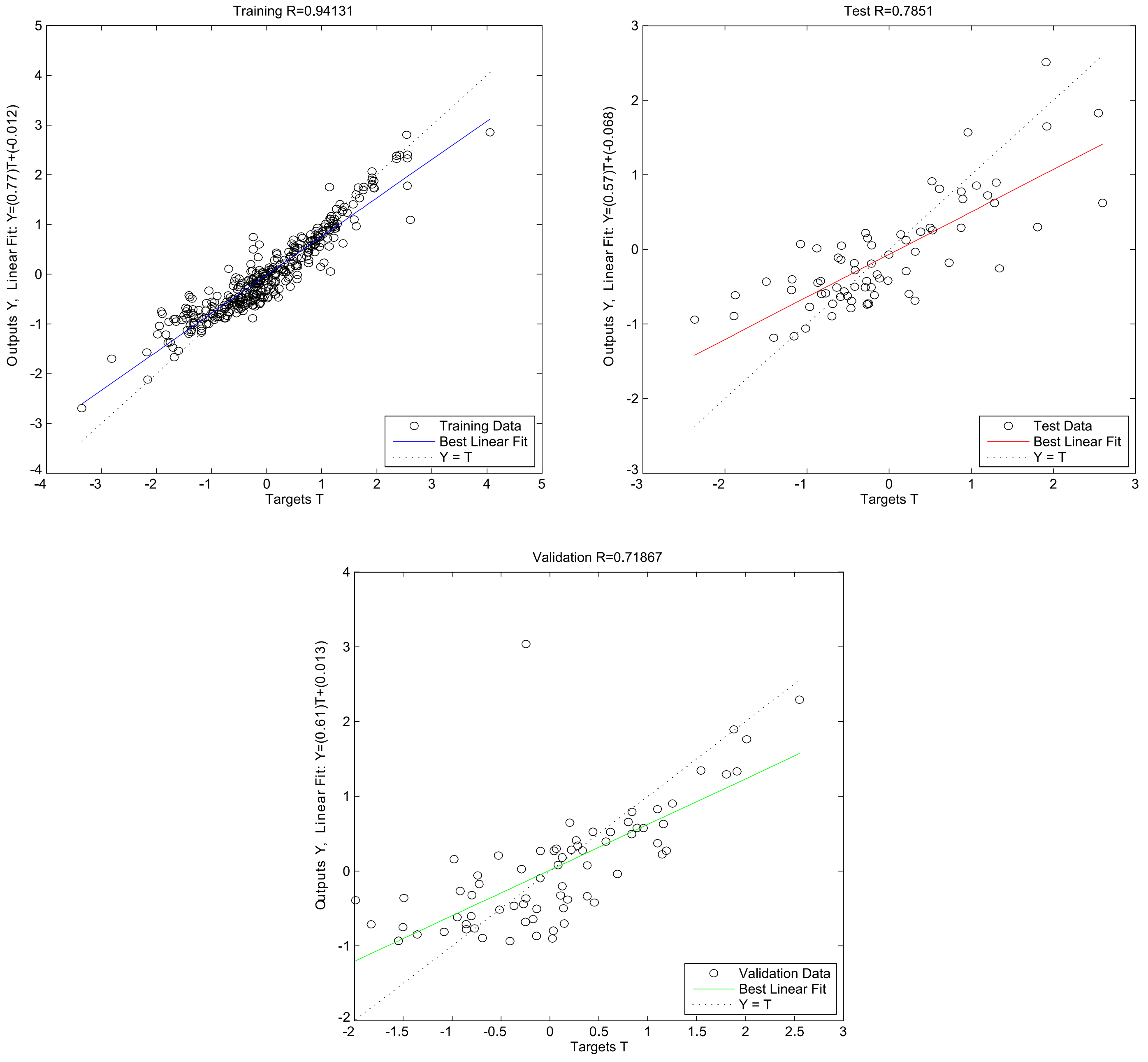


Fig. 13. Regression analysis of GRNNs.

FLUENT. The analysis is carried out at the same flow conditions as that for the panel method given above. The C_p-plots highlight the sensitivity of low Reynolds number flow for minor geometric changes. Consider the Gottingen 300 airfoil as an example. The Neural network airfoil has a marginally larger thickness (the crest of the upper surface) than the actual airfoil. This small increase in thickness has a more than proportional increase in the C_p as shown.

The results are given for both the actual airfoil and the neural network generated airfoil for comparison (see Figs. 15 and 16).

7. Conclusion

This work pertains to determination of airfoil geometry from a given C_p-distribution using neural nets and Bezier–PARSEC parametrization. For variable geometry platforms such determination in real time is of importance. The main contribution of the paper is to use Bezier–PARSEC 3434 parameters instead of full coordinates of the airfoil in combination with three different neural architectures. Secondly, the sample of airfoils considered for this work is considerably larger than other works available in literature. We employed Genetic Algorithms for fine tuning of certain BP3434 parameters. Feed-forward backpropagation, Generalized regression and Radial basis neural nets have been trained and then compared in terms of performance and regression statistics. For this particular problem, we conclude the superiority of feed-forward backpropagation neural nets over the other two architectures.

We are presently working to implement this serial code onto a parallel/distributed computing environment. In such a scenario many of the compromises made in present work may be done away with. For example instead of panel method a full Navier–Stokes solver may be employed on a Graphical Processing Unit (GPU) setup. Sample size of the airfoils may also be enlarged

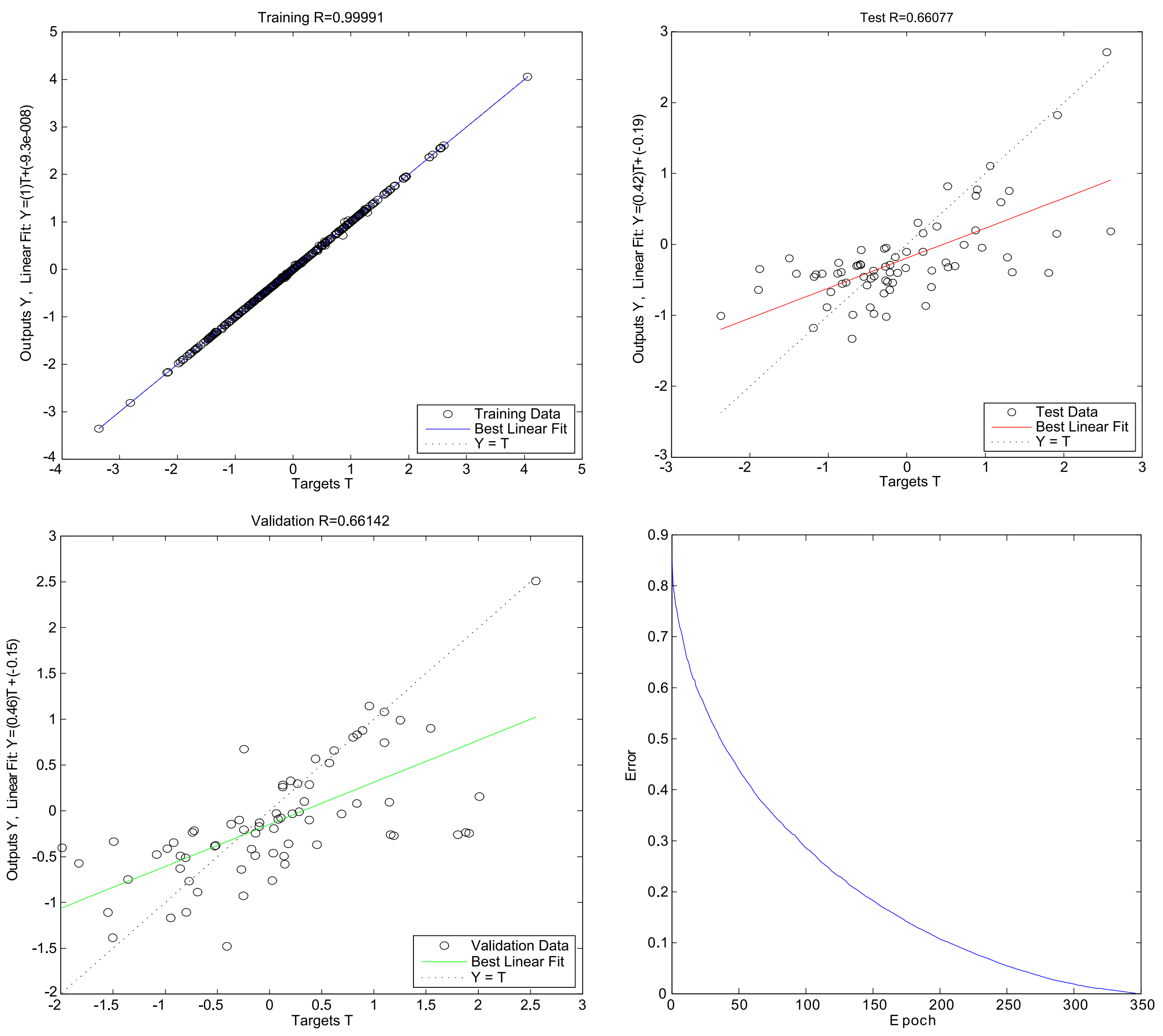
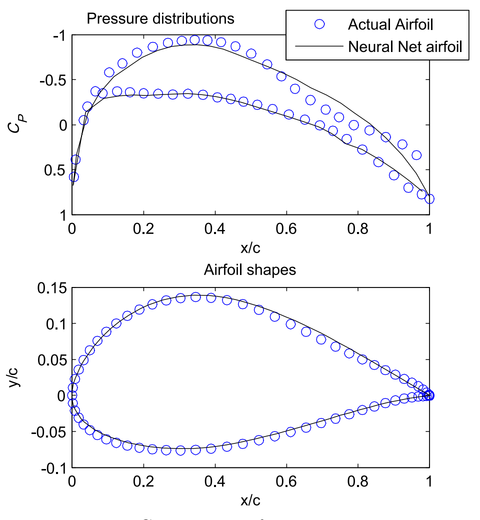
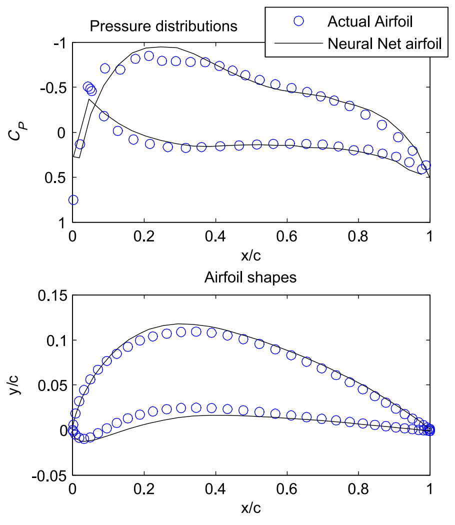


Fig. 14. Regression analysis of RBF networks.

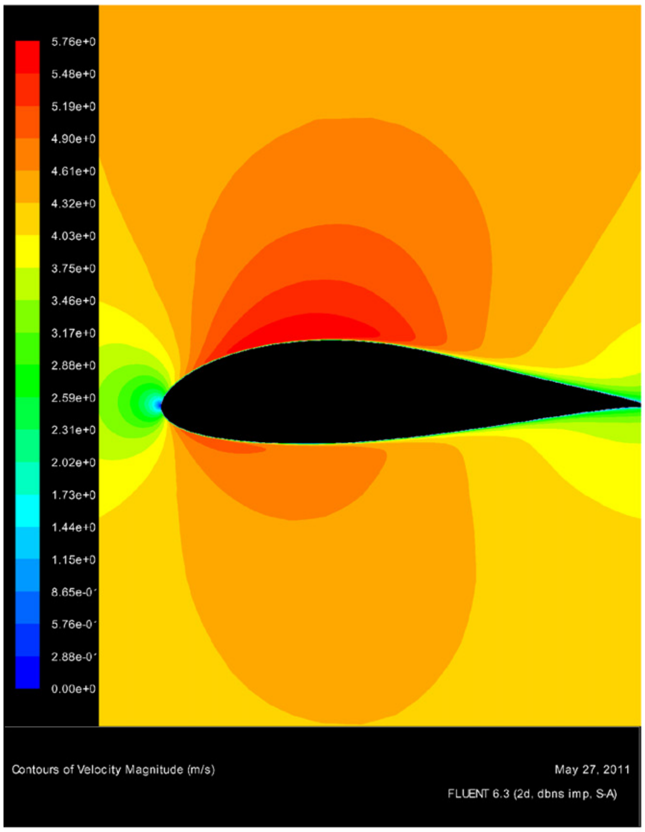


Comparison of pressure distribution and airfoil shape for actual and neural net Althaus $AH93-W-215$

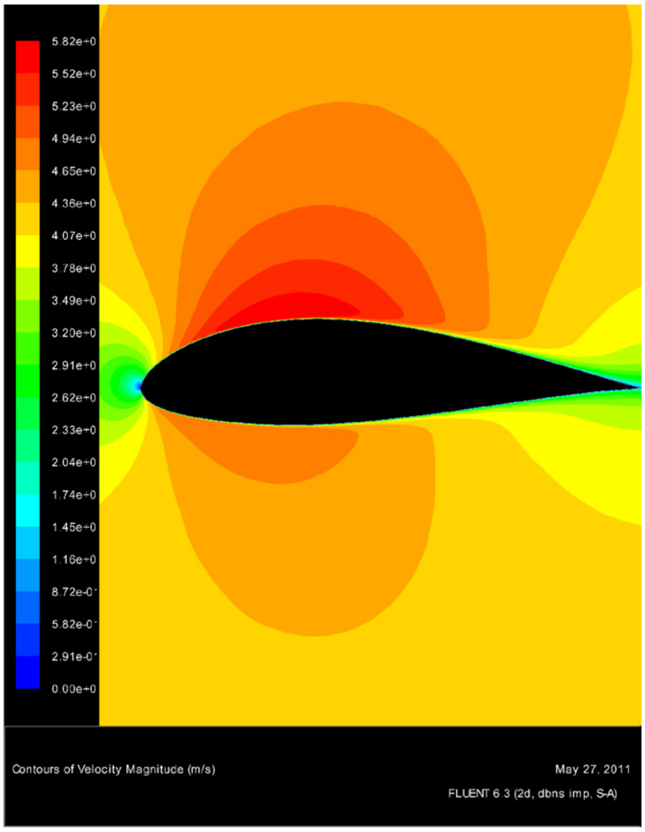


Comparison of pressure distribution and airfoil shape for actual and neural net Gottingen 300

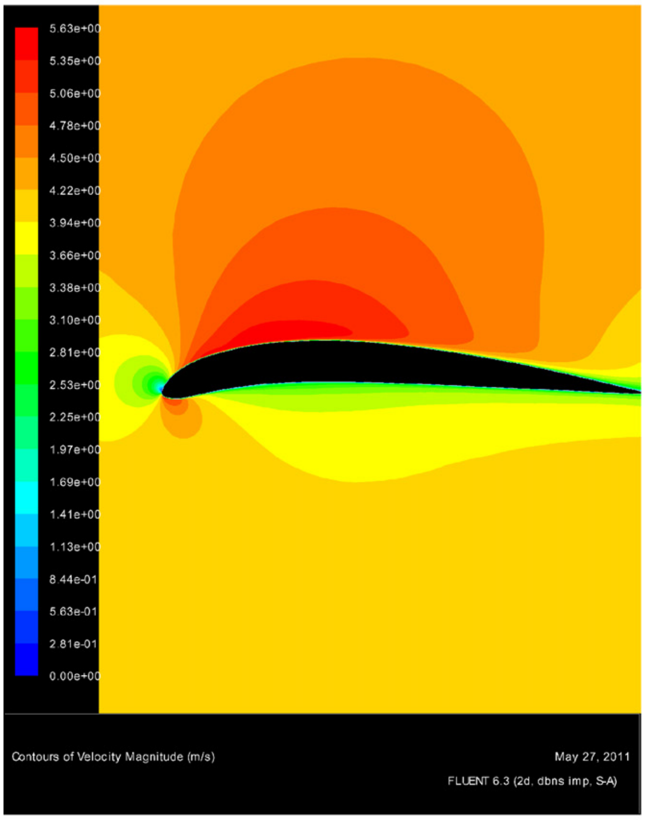
Fig. 15.



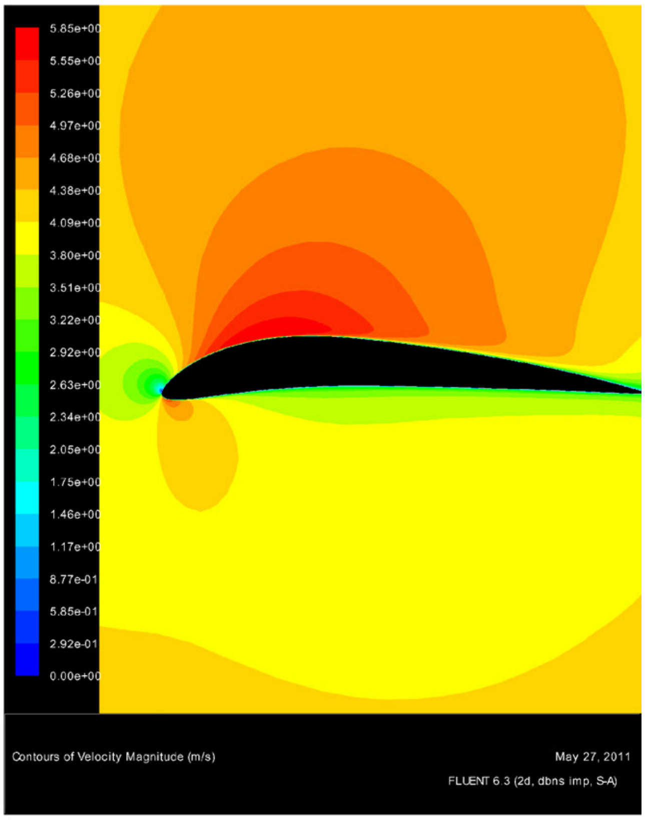
Actual Althaus $AH93 - W - 215$ airfoil



Neural Network Althaus $AH93 - W - 215$ airfoil



Actual Gottingen 300 airfoil



Neural Network Gottingen 300 airfoil

Fig. 16. Velocity magnitude contours.

considerably. Comparison of performance gain between BP3434 and full airfoil coordinates based scheme is also envisaged.

References

[1] http://www.ae.illinois.edu/m-selig/ads/coord_database.html (accessed on 21 August 2011).

[2] I.H. Abbot, A.E. von Doenhoff, Theory of Wing Sections, Dover Publications, New York, 1959.

[3] J. Anderson, Fundamentals of Aerodynamics, 4th edn., McGraw-Hill, NY, 2007.

[4] Application of artificial neural networks to the design of turbomachinery airfoils, AIAA paper 1998-1003.

[5] M. Bellman, J. Straccia, B. Morgan, K. Maschmeyer, R. Agarwal, Improving genetic algorithm efficiency with an artificial neural network for optimization of low Reynolds number airfoils, in: 47th AIAA Aerospace Sciences Meeting, 2009-1096.

[6] R.W. Derksen, T. Rogalsky, Bezier–PARSEC: An optimized aerofoil parameterization for design, Advances in Engineering Software 41 (2010) 923–930.

[7] Ju Ya-ping, Optimal design method for wind turbine airfoil based on artificial neural network model and genetic algorithm, Proceedings of the CSEE 29 (20) (July 15, 2009).

[8] M.M. Rai, K. Nateri, Madavan, Aerodynamic design using neural networks, AIAA Journal 38 (2000).

[9] D.P. Raymer, Aircraft Design: A Conceptual Approach, 4th edn., AIAA Education Series, AIAA, 2006.

[10] T. Rogalsky, R.W. Derksen, Kocabiyik, Differential evolution in aerodynamic optimization, Can. Aeronaut. Space J. 46 (4) (2000) 183–190.

[11] A. Shahrokhi, A. Jahangirian, Airfoil shape parameterization for optimum Navier–Stokes design with genetic algorithm, Aerospace Science and Technology 11 (2007) 443–450.

[12] H. Sobieczky, Parametric airfoils and wings, in: Recent Developments of Aerodynamic Design Methodologies, Friedr. Vieweg & Sohn Verlagsgesellschaft mbH, Braunschweig/Wiesbaden, 1999, pp. 71–87.

[13] H. Sobieczky, Manual aerodynamic optimization of an oblique flying wing, in: 36th Aerospace Sciences Meeting and Exhibit, Reno, NV, 1998, AIAA paper 98-0598.

[14] P. Venkataraman, A new procedure for airfoil definition, in: 13th Applied Aerodynamics Conference, AIAA paper 1995-1875-CP.

[15] Zhi Li Tang, Jean-Antoine Desideri, Towards self-adaptive parameterization of Bezier curves for airfoil aerodynamic design, Rapport de recherché, September 2002.