An evolutionary geometric primitive for automatic design synthesis of functional shapes: The case of airfoils

L. Di Angelo, P. Di Stefano *
Department of Industrial and Information Engineering and Economics, University of L’Aquila, Italy

A R T I C L E   I N F O
Article history:
Received 7 June 2013
Received in revised form 18 September 2013
Accepted 6 October 2013

Keywords:
Geometric primitive
Automatic design synthesis
Artificial neural networks
Airfoils design
Self-adaptive geometric primitive
Functional shape synthesis

A B S T R A C T
A novel self-adaptive geometric primitive for functional geometric shape synthesis is presented. This novel geometric primitive, for CAD use, is specifically designed to reproduce geometric shapes with functional requirements, such as the aerodynamic and hydrodynamic ones, once the functional parameters are furnished. It produces a typical CAD representation of a functional profile: a set of Bézier curves. The proposed primitive follows a generate-and-test approach and takes advantage of the use of a properly designed artificial neural network (BNN). It combines the properties of a geometric primitive and the capability to manage the engineering knowledge in a specific field of application. The proposed evolutionary primitive is applied to a real engineering application: the automatic synthesis of airfoils. Some examples are simulated in order to test the effectiveness of the proposed method. The results obtained by an original prototypal software are presented and critically discussed.

© 2013 Elsevier Ltd. All rights reserved.

1. Introduction

The synthesis of new functional shapes is the scope of many design activities. Typical domains are all those where functional constraints qualify the geometric shape, such as the aerodynamics and the hydrodynamics fields. The design task is to obtain a geometric shape that satisfies some predefined functional requirements.

Generally, for this kind of engineering problem, the geometric shape, which satisfies functional requirements, might allow arbitrary alternatives; in other words, there does not exist a unique association between functional requirements and geometric shape. Furthermore, the typical geometric primitives, used to design the geometric shape, require input geometric coefficients (the control points of the shape), which, in many practical cases, are not directly associated with the characteristic functional parameters; such is the case of airfoil design. For these reasons, the design synthesis of a functional geometric shape is a complex task for the designer to accomplish; he can easily define a geometric shape that satisfies geometric requirements, but he cannot design as easily a geometric shape that satisfies some specific functional requirements (for example, aerodynamic requirements).

With a view to proposing a solution to this design problem, a new geometric primitive is suggested in this paper. This geometric primitive is a computational artefact, which autonomously performs the synthesis of a functional geometric shape from the input of its functional parameters. For this purpose the new type of geometric primitive needs to incorporate the knowledge concerning the specific design problem, so that a solution between the many alternatives can be synthesized. The geometric shape that can be generated must be rigorously compatible with the available knowledge, which is integrated into the geometric primitive. Generally, for this kind of engineering problem there is not a known and well-defined mathematical map between functional requirements and geometric data that represent any geometric shape, not even when the required knowledge concerns a specific design domain. The geometric primitive here proposed is therefore conceived as an evolutionary artefact, which can not only use the old knowledge but also produce and reuse the new knowledge which concerns the specific design problem and which has been obtained while generating new geometric shapes.

The proposed geometric primitive benefits from the evolutionary strategies so that, for each generation item, its capability to synthesize a functional geometric shape is improved and may offer a wider set of engineering solutions. The generative capability comes from prior similar designs and is improved thanks to the knowledge acquired during designing.

The automatic synthesis of a geometric shape cannot be performed without the capability of using the previous knowledge concerning the design problem. There is an initial knowledge that represents the imprinting of the automatic process. The starting knowledge must be based on some well-defined patches of valid
knowledge which covers all the essential aspects of the design problem. The functionality of the proposed primitive evolves during its lifetime, based on the experience acquired, and cannot be known in advance. The design solutions generated in the life cycle and their generation history affect the behaviour of the primitive. In order to reuse the knowledge acquired during the design process, it is essential to define efficient and dynamic ways to store and validate it. During the generative process it is crucial to monitor new solutions which may be generated so as to accept or refuse them. The generation of unexpected results may lead to new design solutions if they can be analysed and understood. That is why an important role is played not only by the monitoring of the generative process, but also by its capability both to evaluate the performance of the generated solution and to recognise new cases. Typically, the automatic design process is monitored by a mathematical model that simulates the physical principle that governs the technical solution. This knowledge is valid if the analysis returns correct inferences about the design solution.

This paper actually analyses the possibility of using a new kind of self-adaptive geometric primitive; this primitive excels the traditional approaches to functional geometry design by performing an automatic synthesis of the geometric shape directly from the functional parameters that characterise it. This primitive is designed to autonomously perform the generate-and-test process, which typically characterises the geometric shape synthesis, and to reuse the knowledge acquired in that process.

The proposed method has found its generative capability both in the knowledge that is transferred to the geometric primitive and in that knowledge which the primitive generates while testing the attempted solutions. Thus, the method extends its generative capability by generalising from the available knowledge. The present approach benefits from a dedicated artificial neural network that directly performs the synthesis of the geometric shape in the form of Bézier curves.

2. Related works

The approach typically used in airfoil design solves an optimisation problem, in which a shape needs to be computed in order to fit assigned aerodynamic parameters. The optimisation approach requires the integration of different modules; each of them plays an important role in the airfoil design process:

- geometric shape function: it serves to define the geometric shape of the airfoil;
- computational solver: it performs a functionality of the geometric shape;
- search model for shape optimisation.

2.1. Geometric shape function

In airfoil optimisation the parametric curves typically used are spline [1], B-spline [2–6] and Bézier [7–14]. These curves can directly represent the geometric shape of the airfoil or the mean camber line and airfoil thickness in the way proposed by Abbott et al. [15]. As pointed out by Song and Keane [16], the first approach has a greater capability than the second one to generate radical new shapes but it is nevertheless less suitable to produce efficient designs because of the larger space in terms of alternative shapes. Marinus [17] states that the B-spline representation delivers better aeronautics results and it is more efficient than the Bézier formulations in terms of convergence. Quite frequently, as far as the use of the Bézier curves is concerned, the uniform parameterisation is used to approximate the airfoil geometry [14]. Tang and Desideri [18] demonstrate that the best Bézier approximations are obtained by using a self-adaptive parameterisation but the convergence of the method proves to be more difficult to achieve.

PARSEC parameterisation [19] has been used by [20–22] to represent an airfoil shape by using geometric parameters having a physical relevance for the aerodynamic problem. It is based on a polynomial expression for both sides of the airfoil with a reduced number of design parameters (airfoil slope at the trailing edge, coordinates and second derivatives of the ordinate of the point at the maximum thickness, radius of curvature at the leading edge). Oyama et al. [23] show that this type of parameterisation improves the robustness and convergence speed for an aerodynamic optimisation involving genetic algorithms. In order to combine the advantages of both the Bézier and the PARSEC parameterisation, some authors [24,25] introduce the Bézier–PARSEC parameterisation. By this approach the mean camber line and airfoil thickness are expressed in the form of Bézier curves by transforming the assigned PARSEC parameters.

2.2. Computational solver

This module typically consists of a flow solver that, starting from the airfoil coordinates, Reynolds and Mach numbers, calculates the corresponding aerodynamic coefficients (pressure distribution on the airfoil and lift and drag characteristics). In the airfoil design process the CFD solver is very time-consuming though. It is estimated that it consumes 90% of the computational time required [26]. The quality of the practical results largely depends on the validity of the results yielded by the computational solver which performs the monitoring of the airfoil shape optimisation. The computational solver is a very important component of the optimisation module; a detailed analysis of its related literature, which is both wide and voluminous, is outside the scope of this work. For the implementation of the evolutionary geometric primitive described in this work the XFOIL [27] has been used as the computational solver.

2.3. Search model for shape optimisation

Typically, the most important optimisation algorithms fall within two categories: gradient-based methods and global-based methods.

The gradient-based methods are iterative methods that extensively use the information of the gradient of the function to be minimised during iterations. Conventional gradient-based methods are used for single-point airfoil design for which one regime of flow condition is analysed [28,29]. These methods are fast but not robust since the search process is very dependent on the starting point and rarely converges to a global minimum. The gradient-based methods require that the mathematical function that links functional requirements and geometric shape should be known, and that continuity and derivability conditions should be met. Such a function is very difficult to define in the fields here considered.

The global-based methods can achieve a global optimum solution without a mathematical map between functional requirements and geometric shape; they iteratively choose a new solution, trying to reduce the value of the objective function (or functions) by changing the characteristic parameters within the admissible region. The most popular global-based methods fall within two categories: evolutionary algorithm (EA) and Artificial Neural Network (ANN) based algorithm. Under the category of EA, mention should be made of the Genetic Algorithms (GAs) [30]. GAs are considered reliable (in the sense of finding solutions which are very close to the global optimum), robust (in the sense of avoiding local optima) and capable to handle multiple objectives. For these reasons, GAs are used for many shape optimisation
problems [13,26,31–39]. The main disadvantage of GAs is that they are very time-consuming. Among the evolutionary algorithms, we also find the Particle Swarm Optimiser (PSO) proposed by Kennedy and Eberhart [40]. This methodology has been applied for airfoil design by some authors [20,21,41].

Artificial Neural Networks have been used in the inverse airfoil design because they can emulate, to a great extent, the nonlinear relationships existing between airfoil configurations and the corresponding functional parameters [42–45]. More recently, Kharal and Saleem [25] have compared the performance of three different neural networks in the generation of an optimised airfoil shape for a given pressure distribution. Feed-forward back-propagation, generalised regression and radial basis neural nets have been trained with 500 heterogeneous airfoils and tested using 200 airfoils unknown to the network. The obtained results show the superiority of feed-forward back-propagation neural nets over the other two architectures.

In order to reduce optimisation costs, some hybrid optimisation algorithms are proposed. These methods can be grouped into multilevel search [46–53] and multilevel parameterisation [54] algorithms. In the multilevel search algorithms each level is associated with different search techniques. Kampolis and Giannakoglou [11] propose an optimisation platform which combines both these hybrid techniques. In order to solve the inverse airfoil design problem, Hacigolu [14] puts forth a hybridisation technique (AGANN) that uses an Artificial Neural Network to drive the generation of the candidate solutions of the genetic algorithm.

3. Bézier neural network

An artificial associative memory is a means capable of generating a complete representation when only a part of the whole knowledge is given. An associative memory can be typically obtained in two ways. The first way is based on an algorithmic reconstruction of the knowledge. Knowledge is defined at the stage of software development as a series of sequences of predefined “if else end” operations. Another way is to create an associative memory by using an artificial neural network. Artificial neural networks, like biological neural networks, contain a collection of neuron units communicating with each other via axon connections. The memory of a neural network is included in the synaptic weights that are adaptively trained by a learning mechanism. In this case, knowledge is defined in a flexible way and can be updated when new knowledge is generated. This characteristic makes the artificial neural network a dynamic artefact suitable to be used in an adaptive process where new knowledge is generated and reused in a self-training process. The neural network has an intrinsic parallel structure and its execution is very rapid. The capability of a neural network to describe knowledge strictly depends on its structure, number of nodes and synaptic connections [55].

A multi-layer feed-forward neural network with an arbitrary large number of units in hidden layers can approximate any real continuous function. It is for this reason that neural networks are also proposed to be used in applications such as those for airfoil profile reconstruction [25,42–45,56–58]. Although the neural network exerts the capability to predict the trend of the lift and drag coefficient, the output of traditional neural networks is not useful in a CAD environment.

With a view to performing a functional profile generation by using specific knowledge, this work presents Bézier Neural Network (BNN). BNN is an artificial neural network properly designed to reproduce geometry in a typical CAD format (a Bézier curve), whose input consists of some functional parameters and not geometric parameters. In BNN the capability of a neural network to approximate very complex and non-linear functions has been combined with the capability of Bézier functions to reproduce geometric shapes.

BNN shows a good capability to reproduce the geometry for which it has been previously trained, and whose output is a Bézier curve. Another property of the neural network is the capability to yield a correct output from an input stimulus for which it has not been trained before. It is a characteristic that may be checked in artificial neural networks and depends on the specific neural model which is chosen [55]. For these reasons, another scope of the neural network in the method being proposed is the generation of a functional profile which is an extension of the trained knowledge. This property has been used to perform the evolutionary strategy of the geometric primitive.

3.1. Configuration of the neural network

BNN is an artificial neural network composed of five layers and three weight layers (Fig. 1).

The first three layers identify a classical multilayer back-propagation network having a sigmoid activation function. The two remaining layers perform the Bézier function.

Layer A is composed of \( n_A \) nodes, which accept the input variables \( y^0 \), and a bias node. An additional input node accepts the parameteric variables \( t (0 \leq t \leq 1) \) of the Bézier function.

Layers B and C are two hidden layers composed respectively of \( n_B \) and \( n_C \) nodes plus a bias node.

Layer D is the third hidden layer. It is composed of \( n_D \) nodes. The input of this layer is transformed by a linear function \( AF_D(t,x^D) \):

\[
y^0 = AF_D(t,x^D) = x^0 z
\]

in which \( z \) is the slope coefficient of the linear function. It is not a critical factor and, in the implementation of the BNN, its value is fixed at 1.0.

The input \( y^0 \) of layer D is the output of the control points of a Bézier function, whose degree is \( n = n_D-1 \).

Layer E is the output of the network. Its value is:

\[
y = \sum_{h=1}^{n_B} y^h B^h_{n_B-1}(t)
\]

where the term \( B^h_{n_B-1} \) denotes the Bernstein polynomials and \( y^h \) is the component of the control points of the Bézier curve.

The weight vectors \( (w^A) \), \( (w^B) \) and \( (w^D) \) can be obtained by minimising the error function defined as follows:

\[
e^{(s)} = \frac{1}{2} \sum_{i=1}^{n} (y^i - y^D)^2
\]

where \( y^{(s)} \) is the output of the BNN in the \( s \)th iteration of the training process and \( y^D \) is the desired output. The weight vectors \( (w^A) \), \( (w^B) \) and \( (w^D) \) are updated in the direction opposite the gradient:

\[
w^{s+1} = w^{s} - \Delta w^{s} = w^{s} - \eta \frac{\partial e}{\partial w^{s}}
\]

where \( \eta \) is the learning factor. The synaptic connections of the BNN are updated by using the following equations:

\[
\Delta w_{k}^{s+1} = \eta z(y-y^D)B^s_{n_B-1} y^D_k
\]

\[
\Delta w_{k}^{a} = \eta z(y-y^D)y^D_k(1-y^D_k)\sum_{h=1}^{n_B} y^h B^h_{n_B-1} w_{h}
\]

In order to describe plane curves or curves in space, two or three BNNs need to be used, one for each co-ordinate. The learning rule for these cases is to minimise half the square distance between the output point \( P(t) \), and the desired point \( P'(t) \). The learning error is formulated by:
where \( m \) indicates the current co-ordinate.

The BNN trains directly by using the geometric shape of the profile and not by giving the control points like other methods in literature do. The BNN is oriented towards performing the generation of a shape profile rather than obtaining the control points. This makes it possible to train the BNN by using functional shapes of different sources; for example, the natural shapes which are directly acquired from the object.

4. Automatic design synthesis of airfoils

Fig. 2 shows the internal structure of the geometric primitive which is proposed. It is the typical structure of a generative system, which performs iterative cycles of generate and test phases. The BNN is the kernel of the generative component; it carries out the synthesis phase of tentative solutions by extending its generative capability beyond the training range related to the available knowledge. It shows the typical adaptive processing capability of neural networks that are based on adaptive learning and self-organisation rules. A salient feature of artificial neural networks is their associative memory.

The knowledge about the design problem is stored in a database as examples of design solutions. This knowledge is elaborated on and transferred to the BNN during its training process, and stays within the values of the synaptic weights. A self-training algorithm updates the values of the synaptic weights every time new knowledge is developed in the life cycle. This makes the proposed generative system an autonomous artefact that continuously improves its generative capability.

From a mathematical standpoint, the generative component performs a non-linear mapping between the domain of the design parameters and the domain of the related functional geometric shape.

\[
g^{i-1} = F(p_T) \tag{7}
\]

where \( g = \{g_x(t), g_y(t)\} \) is the parametric representation of the geometric shape, \( p_T = [p_1, ..., p_n] \) is the vector of the required functional parameters and \( F \) is the function that approximates the map of the geometric shape from functional parameters after the \( i \)th iteration of the generative process.

The above function is dynamically defined and changes during the life cycle of the generative system. The function \( F \) is automatically updated in each training cycle by an adaptive process which modifies the values of the synaptic weight.

\[
p^{i+1}_k = A(g^{i+1}) \tag{8}
\]

In the generative cycle, the pairs of data \( \{p^{i+1}_k, g^{i+1}\} \) obtained after each iteration represent new pieces of knowledge that are stored in a training data file. The real functional parameter \( p^{i+1}_k \) of the tentative solution \( g^{i+1} \) is estimated by the analysis of the airfoil at the \((i + 1)\)th iteration of the design process.

In the domain of the functional parameters, the distance between the required functional parameters and the verified parameters of the generated shape geometry represents a measure of the error in the current iteration of the generative process. It is evaluated as follows:

\[
r^i = \sqrt{\sum_j \left( \frac{(p_{ij} - p^{*}_{ij})^2}{(p_{ij})^2} \right)} \tag{9}
\]

where \( p_{ij} \) is the \( j \)th component of the vector \( p_T \) of the required functional parameters. The iterative process is stopped when the distance \( r \) is less than a given reference value \( c \).

After each search iteration, a training cycle of the BNN is performed and the generative function \( F^{i+1} \) is updated.

In order to use the proposed evolutionary primitive in practical design activities, the whole system has been implemented in CATIA V5 (Dassault Systems) by using the VBA language and available functions in the CAA API. The monitoring component of the ADS, is
an external computational fluid dynamic solver [27] that has been interfaced with the VBA developed macro in CATIA. In Fig. 5a the dialog box is shown where the input parameters are reported.

5. Cases studied

Airfoils are typical objects whose performance strictly depends on their shape. For these objects there does not exist a structured mathematical map between the functional requirements and the geometric data which describe the geometric shape. The assigned performance, defined by some functional parameters, can be reached through alternative solutions. Practically speaking, the shape is designed through an iterative process by means of an experimental analysis of prototypes or by a computer-based simulation of a pre-designed shape. Sometimes the geometric data are tabulated in specific handbooks according to their functional property. Such is the case of the NACA airfoil profiles [15].

Typical functional parameters of airfoil profile are: Lift ($C_L$) and drag ($C_D$) coefficients, which are functions of the operative conditions defined by the Reynolds number ($Re$) and the angle of attack ($\alpha$).

Two BNNs are required to model a single airfoil profile: one for the upper profile and another for the lower profile of the airfoil.

The ADS has been initialised by a set of training data taken from selected NACA profiles, which are typically used in airfoil construction. In particular, the starting knowledge is based on five NACA profiles of the four-digit series (2418, 2420, 2422, 2424 and 2426). All the tests have been conducted by using XFOIL [27] as the monitoring component of the ADS. The airfoil has been analysed by assuming fixed values for the Reynolds number ($Re = 300,000$) and for the angle of attack ($\alpha = 2\degree$). The values of the coefficient $C_L$ and $C_D$ inferred from the CFD software, have been assumed to be the real functional parameters of the airfoil. Training data consisting of functional parameters and a shape geometry (described by 30 points for each part of the airfoil profile) have been submitted to the BNN. After the initial training, the mean square deviation of the output of the BNN, for all the points of the wing profiles used in the training process, is less than $10^{-4}$ and the maximum deviation is 0.0056 units. The error in reproducing functional parameters for the set of initial training airfoils is relatively low even when it is compared with the results reported in literature.

The ADS has been tested to reproduce an airfoil having functional parameters outside the initial training range: $C_L = 0.514$ and $C_D = 0.0078$ (Point A in Fig. 3). These values of the functional requirements characterise the NACA 2416 airfoil profile which is not included in the initial training data, although it belongs to the same family of profiles used for the BNN training. A valid solution, which provides the functional target point, has been obtained in four generate-and-test cycles. The generative process has been interrupted for an error ($\epsilon$) less than 0.075. The final airfoil has been generated in four attempts and the characteristic parameters are depicted in Fig. 3. The sequence of values of the functional parameters shows the disposition of the proposed ADS to benefit from the new knowledge generated in each iterative cycle and a capability to generalise from it.

In Fig. 4 the generated airfoil profile is compared with the NACA profile which performs the same functional parameters. The generated profile is similar to the NACA 2416. It evidences how the imprinting knowledge used to train the BNN affects the generated profile.

Fig. 5 shows two screenshots of the ADS implementation in CATIA V5: in Fig. 5a, the dialog box for the input of functional and geometric parameters; in Fig. 5b, the results of the ADS in terms of 3D geometric shape.
In order to verify the role of the history of the generated instances in the evolution of the ADS, other tests have been conducted. For this purpose, a new target point has been chosen far from the initial training data (B). The chosen functional parameters do not have a correspondent wing profile in the NACA series. The target profile has been generated in two different knowledge states of the ADS. In the first state, only the initial knowledge has been used (5 airfoils) (Fig. 6a). In the second state, both the initial knowledge and that developed in the generation of the earlier example (airfoil A) have been used (9 profiles, Fig. 6b). In the first case, the target point has been approximated in three steps whereas in the second case, the same point has been approximated in four steps, as shown in Fig. 6. Although more attempts have been required to obtain an acceptable solution in the second case, the generated profile in the first attempt presents functional parameters not very far from the target.

Fig. 7 depicts the two generated profiles. The dimensional differences between the profiles are reported respectively for inferior and superior profiles of the airfoil. With a difference of 5% in thickness, the two profiles are assumed to show the same functional performance. The differences in shape are due to the knowledge that has been used to generate each profile.

The proposed geometric primitive has been tested in reproducing airfoils derived from the NACA 14XX and 34XX and similar results have been obtained.

6. Comparison and discussion

The performance of BNN is compared with a different approach, used in the literature [14], to generate airfoil profile. It is based on a typical back-propagation artificial neural network (BPNN) which, starting from the airfoil functional parameters, furnishes the Bézier control points of the related geometric profile. Both BNN and BPNN generate airfoils in the form of two Bézier curves. BPNN uses a
general purpose artificial neural network which develops an associative memory between the functional parameters and the control points of a Bézier curve. BNN is an artificial neural network especially conceived as a geometric primitive which develops an associative memory between the functional parameters and the shape geometry. The training data for BPNN consist in a set of control points of Bézier curves, whereas the training data for BNN are a set of points of the geometric profile. Furthermore, the BPNN training process is performed by means of a numerical optimisation while BNN is trained by using analytical functions (Eq. (5)) which make it specifically suitable to generate Bézier curves. BPNN requires that the geometric shape should be preventively approximated by a Bézier curve whereas BNN performs the training process and the profile approximation in an only process.

A BPNN with 4 nodes in the first layer, 40 nodes in each of the two hidden layers, sigmoid–tangent activation functions and 8 output nodes (it is the same structure as the first four layers of the BNN) has been implemented in MatLAB®. The training process is carried out according to the Levenberg – Marquardt (trainlm) training function. BPNN has been trained by using the same set of NACA profiles used to train the BNN whose Bézier control points have been obtained by approximating each NACA profile by the least square method and the centripetal parameterisation. The training process of BPNN has been stopped when the errors stabilise at a constant value.

At the end of the training process, BPNN reproduces the NACA profiles with errors which are comparable to those produced by BNN (Table 1). Then, BPNN, thus trained, has been used to generate a profile which has the same functional parameters as the NACA 2416, by the same generative approach used to test BNN, described in Section 4. Errors produced by both neural networks at each step of the generative process are quoted in Table 2. From the first generative step, BPNN produces a greater percentage of error and it does not converge to an acceptable solution. Analogous results have been obtained using the same number of control points (9 and 13) used by Hacioglu in his work [14]. Even when using a different parameterisation criterion to generate the control points necessary to train BPNN, it does not yield different results.

The training process performed by using control points can be affected by some contradictions, which give rise to an incoherent neural network generation. In order to clarify the concept of contradiction, it should be observed that different sets of control
points can represent the same geometric profile characterised by the same functional parameters. Thus, a geometric profile in the training data set can be replaced by another one which is identical but has different control points. Typically, the incoherence in the training data set can generate training problems or limit the capability of the neural network to generate profiles which fall outside the training range. The geometric profile generated outside the training range is a mathematical extension of the associative memory retained in the weights of the neural network synaptic connections. This extension is valid if the neural network makes a coherent continuing shift from a geometric profile to another as well as if functional input parameters vary. This form of incoherence in the training data set can be replaced by another one which is identical but has different control points. Typically, the incoherence in the training data set can generate training problems or limit the capability of the neural network to generate profiles which fall outside the training range. The geometric profile generated outside the training range is a mathematical extension of the associative memory retained in the weights of the neural network synaptic connections. This extension is valid if the neural network makes a coherent continuing shift from a geometric profile to another as well as if functional input parameters vary. This form of incoherence in the training data set can be replaced by another one which is identical but has different control points. Typically, the incoherence in the training data set can generate training problems or limit the capability of the neural network to generate profiles which fall outside the training range.

Table 2

<table>
<thead>
<tr>
<th>Step</th>
<th>( r ) ( ^{a} ) BNN</th>
<th>( r ) ( ^{a} ) BPNN</th>
</tr>
</thead>
<tbody>
<tr>
<td>N#1</td>
<td>( 1.557 \times 10^{-2} )</td>
<td>( 2.181 \times 10^{-1} )</td>
</tr>
<tr>
<td>N#2</td>
<td>( 1.556 \times 10^{-2} )</td>
<td>( 2.010 \times 10^{-1} )</td>
</tr>
<tr>
<td>N#3</td>
<td>( 3.891 \times 10^{-4} )</td>
<td>( 2.020 \times 10^{-1} )</td>
</tr>
<tr>
<td>N#4</td>
<td>( 1.945 \times 10^{-4} )</td>
<td>( 2.239 \times 10^{-1} )</td>
</tr>
</tbody>
</table>

\( ^{a} \) \( r \) is evaluated according to Eq. (9).

Table 1

<table>
<thead>
<tr>
<th>NACA</th>
<th>( \text{Error}_{C_0} ) (%)</th>
<th>( \text{Error}_{C_2} ) (%)</th>
<th>( \text{Error}_{C_0} ) (%)</th>
<th>( \text{Error}_{C_2} ) (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>2418</td>
<td>1.15</td>
<td>0.00</td>
<td>0.77</td>
<td>5.45</td>
</tr>
<tr>
<td>2420</td>
<td>0.56</td>
<td>1.22</td>
<td>1.69</td>
<td>0.89</td>
</tr>
<tr>
<td>2422</td>
<td>0.00</td>
<td>3.66</td>
<td>10.58</td>
<td>1.75</td>
</tr>
<tr>
<td>2424</td>
<td>0.72</td>
<td>4.76</td>
<td>5.81</td>
<td>0.85</td>
</tr>
<tr>
<td>2426</td>
<td>3.74</td>
<td>9.64</td>
<td>5.19</td>
<td>0.83</td>
</tr>
<tr>
<td>Mean</td>
<td>1.23</td>
<td>3.62</td>
<td>4.81</td>
<td>1.96</td>
</tr>
</tbody>
</table>

The proposed geometric primitive benefits from the available knowledge about the engineering problem and from the knowledge that it itself produces during the design process. This CAD primitive evolves during its life cycle improving its capability to generate functional shapes by using results coming from the analysis of the trial cases which have been generated during the airfoil design synthesis. This primitive generates new shape geometries coherently with the knowledge imprinted in the initial training. In the test cases here presented it shows a capability to generate airfoils having a series of functional parameters which are very different from those of the initial training. This geometric primitive builds its design capabilities on the imprinting knowledge and on all the new knowledge generated in its operative life. It is able to manage and reuse the knowledge produced in the design generation cycles which can be difficult to memorise and is generally lost in the traditional approach to design of functional geometric shapes. The initial imprinting is very important since it furnishes the conceptual scheme of the geometric shape to be generated. The BNN evolves during its life cycle depending on the history of the analysed cases but always coherently with the imprinting setup.

The obtained results are promising for other kind of uses in which functional profiles have to be generated, but the following conditions must be verified:

- the training data have to represent a coherent set of geometric profiles;
- each geometric profile, in the training data, must be different for the other, so that ambiguity are not introduced;
- according to the universal approximation theorem [59,60], an adequate number of nodes in hidden layers must be used;
- in order to perform the self-adaptive capability, a valid computational analysis of the generated profiles must be implemented inside the primitive.

### References


