Adaptive Neuro-Fuzzy Inference Controllers for Smart Material Actuators

Teodor Lucian Grigorie\textsuperscript{1} and Ruxandra Mihaela Botez\textsuperscript{2}
École de Technologie Supérieure, Montréal, Quebec H3C 1K3 Canada

An intelligent approach for smart material actuator modeling of the actuation lines in a morphing wing system is presented, based on adaptive neuro-fuzzy inference systems. Four independent neuro-fuzzy controllers are created from the experimental data using a hybrid method -- a combination of back-propagation and Least-Mean-Square (LMS) methods -- to train the fuzzy inference systems. The controllers’ objective is to correlate each set of forces and electrical currents applied on the smart material actuator to the actuator’s elongation. The actuator experimental testing is performed for five force cases, using a variable electrical current. An integrated controller is created from four neuro-fuzzy controllers, developed with Matlab/Simulink software for electrical current increases, constant electrical current, electrical current decreases, and for null electrical current in the cooling phase of the actuator, and is then validated by comparison with the experimentally obtained data.

Nomenclature

\begin{align*}
A_i^q & = \text{associated individual antecedent fuzzy sets of each input variable } \ (i=1, N) \\
a, b & = \text{parameters of the generalized bell membership function} \\
a_i^k & = \text{parameters of the linear function } \ (k=1, 2, i=1, N) \\
b_i^0 & = \text{scalar offset } \ (i=1, N) \\
C_p & = \text{pressure coefficient} \\
c & = \text{cluster center} \\
c_i^q & = \text{cluster center } \ (q=1, 2) \\
\end{align*}

\textsuperscript{1}Postdoctoral fellow, Laboratory of Research in Active Controls, Avionics and AeroServoElasticity LARCASE, 1100 Notre-Dame West Street, Montreal, Quebec, H3C 1K3, Canada, AIAA Member

\textsuperscript{2}Professor, Laboratory of Research in Active Controls, Avionics and AeroServoElasticity LARCASE, 1100 Notre-Dame West Street, Montreal, Quebec, H3C 1K3, Canada, AIAA Member
\( F \) = force
\( i \) = electrical current
\( k \) = variable for neuro-fuzzy controller selection
\( k_y \) = step size
\( l \) = dimension of the data vectors
\( M \) = number of data points
\( N \) = number of rules
\( r_m \) = radius within the fuzzy neighborhood, contributes to the density measure
\( Re \) = Reynolds number
\( t \) = time
\( T \) = temperature of the smart material actuator
\( T_{amb} \) = ambient temperature
\( u_{ij} \) = center of the \( j^{th} \) cluster
\( u_k \) = data vectors
\( V \) = speed
\( \omega^i \) = degree of fulfillment of the antecedent, i.e., the level of firing of the \( i^{th} \) rule
\( x \) = input vector
\( x_q \) = individual input variables \( (q=1,2) \)
\( y \) = output of the fuzzy model
\( y^i \) = first-order polynomial function in the consequent \( (i=1,N) \)
\( \alpha \) = angle of attack
\( \epsilon \) = cost function
\( \rho \) = density measure
\( \sigma \) = dispersion
\( \sigma_q^i \) = cluster dispersion
\( \Delta t \) = time variation
\( \Delta \delta \) = actuator elongation
I. Introduction

The aim of this paper is to obtain a reliable, easy-to-implement model for Smart Material Actuators (SMAs), with direct applications in the morphing wing project. Based on adaptive neuro-fuzzy inference systems, an integrated controller is built to model the smart material actuators used in the actuation lines of a wing.

As shown in Fig. 1, a complex system was obtained, which modifies the airfoil shape in order to optimize it from the perspective of the laminar flow region. For various flight conditions (angles of attack $\alpha$, speeds $V$ and Reynolds numbers $Re$), the loop controller would receive the airfoil upper surface $C_p$ distribution, determined from the surface pressure distribution measured by the kulite sensors. The $C_p$ distribution is compared with a computational fluid dynamics (CFD) database, which is generated so that for different airfoil types, the transition point is given as a function of the $C_p$ distribution. Once a match is found, a transition point is offered to the loop controller by the CFD database; the controller will be able to decide if the airfoil shape must be adjusted or not. The adjustment of the airfoil shape is made in real time using the SMA actuators. The loop is closed by the airfoil shape, which offers the optical sensors another surface pressure distribution.

![Fig. 1 Closed-loop morphing wing system](image)

In order to validate the morphing wing system (numerical simulation versus test results), good numerical models for each of the physical elements in the system must be obtained. The aim of this paper is to offer a good model for SMA actuators, with direct application to our morphing wing system.
This model uses the numerical values from the SMAs’ experimental testing and it takes advantage of the outstanding properties of fuzzy logic, which allow the signal’s empirical processing without the use of mathematical analytical models. Fuzzy logic systems can emulate human decision-making more closely than many other classifiers through the processing of expert system knowledge, formulated linguistically in fuzzy rules in an IF-THEN form. Fuzzy logic is recommended for very complex processes, when no simple mathematical model exists, for highly nonlinear processes, and for multi-dimensional systems.

The input variables in a fuzzy control system are usually mapped into place by sets of membership functions (mf) known as “fuzzy sets”; the mapping process is called “fuzzification”. The control system’s decisions are made on the basis of a fuzzy rules set, and are invoked using the membership functions and the truth values obtained from the inputs; a process called “inference”. These decisions are mapped into a membership function and truth value that controlling the output variable. The results are combined to give a specific answer in a procedure called “defuzzification”. Elaboration of the model thus requires a fuzzy rules set and the membership functions (mf) associated with each of the inputs ([1], [2]).

The ability and the experience of a designer in evaluating the rules and the membership functions of all of the inputs are decisive in obtaining a good fuzzy model. However, a relatively new design method allows a competitive model to be built using a combination of fuzzy logic and Neural-Network techniques. Moreover, this method allows the possibility to generate and optimize the fuzzy rules set and the parameters of the membership functions by means of fuzzy inference systems’ training. To this end, a hybrid method -- a combination of back-propagation and Least-Mean-Square (LMS) methods – is used, in which experimentally obtained data are considered. Already implemented in Matlab ([1], [3]), the method is easy to use, and gives excellent results in a very short time.

II. Actuator experimental testing

The SMA testing was performed using the bench test in Fig. 1 at \( T_{\text{amb}} = 24 \, ^\circ \text{C} \), for five load cases with forces of 120 N, 140 N, 150 N, 180 N and 190 N. The electrical currents following the increasing-constant-decreasing-zero values evolution were applied on the SMA actuator in each of the five cases considered for load forces. In each of the cases to be analyzed, the following parameters were recorded: time, the electrical current applied to the SMA, the load force, the material temperature and the actuator elongation (measured using a Linear Variable Differential Transformer (LVDT)).
To model the SMA we built an integrated controller based on Adaptive Neuro-Fuzzy Inference Systems. The experimental elongation-current curves obtained in the five load cases are shown in Fig. 2. One can observe that all five of the obtained curves have four distinct zones: electrical current increase, constant electrical current, electrical current decrease and null electrical current in the cooling phase of the SMA. Four Fuzzy Inference Systems (FIS’s) are used to obtain four neuro-fuzzy controllers: one for the current increase, one for the constant current, one for the current decrease, and one controller for the null current (after its decrease). For the first and the third controllers, inputs such as the force and the current are used, while for the second and the fourth, inputs such as the force and the time values reflecting the SMA’s thermal inertia are used (the time values required for the SMA to recover its initial temperature value (approximately 24°C) are used for the four controller). Finally, the four obtained controllers must be integrated into a single controller.

The reasoning behind the design of the first and the third controllers is that, from the available experimental data, two elongations for the same values of forces and currents are used (see Fig. 2). Due to the experimental data values, this data cannot be represented as algebraic functions; therefore, it is impossible to use the same FIS representation. Matlab produces an interpolation between the two elongation values obtained for the same values of forces and currents, which cannot be valid for our application.

The constant values, namely the null values of the current before and after the current decrease phase, should not be considered as inputs in the second and fourth controllers because they are not suggestive for the characterization of the SMA elongation. The values of the actuator temperatures may appear to be very suggestive in these phases,
but the temperature must be a model output. For these phases the time values are very suggestive, as they represent a measure of the actuator thermal inertia. Time is the second input of the third controller, and so time is also the second input of the second and the fourth controllers – since force was considered as the first input (the time values must be considered when the current becomes constant or null).

III. The proposed method

Fuzzy controllers are very simple conceptually and are based on fuzzy inference systems (FIS’s). Three steps are considered in a fuzzy inference system design: an input, the processing, and then an output step. In the input step, the controller inputs are mapped into the appropriate membership functions ($mf$). Next, a collection of “IF-THEN” logic rules is created; the IF part is called the “antecedent” and the THEN part is called the “consequent”. In this step, each appropriate rule is invoked and a result is generated. The results of all of the rules are then combined. In the last step, the combined result is converted into a specific control output value.

Considering the numerical values resulting from the SMA experimental testing, an empirical model can be developed, which is based on a neuro-fuzzy network. The model can learn the process behavior based on the input-output process data by using an FIS, which should model the experimental data.

Using methods already implemented in commercial software, an FIS can be generated simply with the Matlab “genfis1” or “genfis2” functions. The “genfis1” function generates a single-output Sugeno-type fuzzy inference system (FIS) using a grid partition on the data (no clustering). This FIS is used to provide initial conditions for ANFIS training. The “genfis1” function uses generalized Bell-type membership functions for each input. Each rule
generated by the “genfis1” function has one output membership function, which is, by default, of a linear type. It is also possible to create an FIS using the Matlab “genfis2” function. This function generates an initial Sugeno-type FIS by decomposing the operation domain into different regions using the fuzzy subtractive clustering method. For each region, a low-order linear model can describe the local process parameters. Thus, the non-linear process is locally linearized around a functioning point by use of the Least Squares (LS) method. The obtained model is then considered valid in the entire region around this point. The limitation of the operating regions implies the existence of overlapping among these different regions; their definition is given in a fuzzy manner. Thus, for each model input, several fuzzy sets are associated to their membership functions’ corresponding definitions. By combining these fuzzy inputs, the input space is divided into fuzzy regions. A local linear model is used for each of these regions, while the global model is obtained by defuzzification with the gravity center method (Sugeno), which performs the interpolation of the local models’ outputs ([1], [3]).

Based on the goal of finding regions with a high density of data points in the featured space, the subtractive clustering method is used to divide the space into a number of clusters. All of the points with the highest number of neighbors are selected as centers of clusters. The clusters are identified one by one, as the data points within a prespecified fuzzy radius are removed (subtracted) for each cluster. Following the identification of each cluster, the algorithm locates a new cluster until all of the data points have been checked. If a collection of \( M \) data points, specified by \( l \)-dimensional vectors \( u_k, k = 1, 2 \ldots, M \), is considered, a density measure at data point \( u_k \) can be defined as follows:

\[
\rho_k = \sum_{j \neq k} \exp \left( -\frac{|u_k - u_j|^2}{(r_m / 2)^2} \right),
\]

where \( r_m \) is a positive constant that defines the radius within the fuzzy neighborhood and contributes to the density measure. The point with the highest density is selected as the first cluster center. Let \( u_{c1} \) be the selected point and \( \rho_{c1} \) its density measure. Next, the density measure for each data point \( u_k \) is revised by the formula:

\[
\rho'_k = \rho_k - \rho_{c1} \exp \left( -\frac{|u_k - u_{c1}|^2}{(r_n / 2)^2} \right),
\]

where \( r_n \) is a positive constant, greater than \( r_m \), that defines a neighborhood where density measures will be reduced in order to prevent closely spaced cluster centers. In this way, the data points near the first cluster center \( u_{c1} \) will have significantly reduced density measures, and therefore cannot be selected as subsequent cluster centers. After
the density measures for each point have been revised, then the next cluster center \( u_{c_2} \) is selected and all the density measures are again revised. The process is repeated until all the data points have been checked and a sufficient number of cluster centers generated. When the subtractive clustering method is applied to an input-output data set, each of the cluster centers are used as the centers for the premise sets in a singleton type of rule base ([4]).

The Matlab “genfis1” function generates membership functions of the generalized Bell type, defined as follows ([2], [5]):

\[
A'_i(x) = \left(1 + \frac{x - c'_i}{a}ight)^{-b},
\]

where \( c'_i \) is the cluster center defining the position of the membership function, \( a \) and \( b \) are two parameters which define the membership function shape, and \( A'_i \) \( (i=1,N) \) are the associated individual antecedent fuzzy sets of each input variable \( (N = \text{number of rules}) \). Matlab’s “genfis2” function generates Gaussian-type membership functions, defined with the following expression ([2], [5]):

\[
A'_i(x) = \exp\left(-0.5 \frac{(x - c'_i)^2}{\sigma'_i}\right),
\]

where \( c'_i \) is the cluster center and \( \sigma'_i \) is the dispersion of the cluster.

The Sugeno fuzzy model was proposed by Takagi, Sugeno and Kang to generate fuzzy rules from a given input-output data set ([6]). In our system, for each of the four FIS’s (two inputs and one output), a first-order model is considered, which for \( N \) rules is given by ([5], [6]):

\[
\text{Rule } 1: \quad \text{If } x_i \text{ is } A^i_1 \text{ and } x_j \text{ is } A^j_1, \quad \text{then } y^i(x_i, x_j) = b^i_0 + a^i_1 x_i + a^i_2 x_j, \\
\vdots \\
\text{Rule } i: \quad \text{If } x_i \text{ is } A^i_i \text{ and } x_j \text{ is } A^j_i, \quad \text{then } y^i(x_i, x_j) = b^i_0 + a^i_1 x_i + a^i_2 x_j, \\
\vdots \\
\text{Rule } N: \quad \text{If } x_i \text{ is } A^i_N \text{ and } x_j \text{ is } A^j_N, \quad \text{then } y^i(x_i, x_j) = b^i_0 + a^i_1 x_i + a^i_2 x_j,
\]

where \( x_i (q=1,2) \) are the individual input variables and \( y^i (i=1,N) \) is the first-order polynomial function in the consequent. \( a^i_k (k=1,2, i=1,N) \) are parameters of the linear function and \( b^i_0 (i=1,N) \) denotes a scalar offset. The parameters \( a^i_k, b^i_k (k=1,2, i=1,N) \) are optimized by the Least Square method.

For any input vector, \( \mathbf{x} = [x_i, x_j]^T \), if the singleton fuzzifier, the product fuzzy inference and the center average
defuzzifier are applied, then the output of the fuzzy model $y$ is inferred as follows (weighted average):

$$y = \left( \sum_{i=1}^{N} \frac{w'_i(x) \cdot y_i}{\sum_{i=1}^{N} w'_i(x)} \right), \quad (6)$$

where

$$w'_i(x) = A'_i(x_i) \times A'_i(x_i). \quad (7)$$

$w'_i(x)$ represents the degree of fulfillment of the antecedent, i.e., the level of firing of the $i$th rule.

The adaptive neuro-fuzzy inference system calculates the Sugeno-type fuzzy inference system parameters using Neural Networks. A very simple way to train these FISs is to use Matlab’s “ANFIS” function, which uses a learning algorithm to identify the membership function parameters of a Sugeno-type fuzzy inference system with two outputs and one input. As a starting point, the input-output data and the FIS models generated with the “genfis1” or “genfis2” functions are considered. “ANFIS” optimizes the membership functions’ parameters for a number of training epochs, determined by the user. With this optimization, the neuro-fuzzy model can produce a better process approximation by means of a quality parameter in the training algorithm ([3]). After this training, the models may be used to generate the elongation values corresponding to the input parameters.

To train the fuzzy systems, ANFIS employs a back-propagation algorithm for the parameters associated with the input membership functions, and Least-Mean-Square estimations for the parameters associated with the output membership functions. For the FIS’s generated using the “genfis1” or “genfis2” functions, the membership functions are generalized Bell type or Gaussian type, respectively. According to equations (3) and (4), in these types of membership functions, $a$, $b$ and $c$, respectively $\sigma$, and $c$, are considered variables and must be adjusted. The back-propagation algorithm may, therefore, be used to train these parameters. The goal is to minimize a cost function of the following form

$$\varepsilon = \frac{1}{2} (y_{des} - y)^2, \quad (8)$$

where $y_{des}$ is the desired output. The output of each rule $y'(x_i, x_i)$ is defined by:

$$y'(t + 1) = y'(t) - k \cdot \frac{\partial \varepsilon}{\partial y'}, \quad (9)$$

where $k_i$ is the step size.

Starting from the Sugeno-system’s output (eq. (6)), modifying with eq. (9) results in:
\[
\frac{\partial E}{\partial y'} = \frac{\partial E}{\partial y} \cdot \frac{\partial y}{\partial y'},
\]

with

\[
\frac{\partial E}{\partial y} = y_{\text{in}} - y', \quad \frac{\partial y}{\partial y'} = \frac{w(x)}{\sum_{i=1}^{N} w'(x)}.
\]

Therefore, the output of each rule is obtained with the equation:

\[
y'(t+1) = y'(t) - k \cdot \frac{(y_{\text{in}} - y') \cdot w(x)}{\sum_{i=1}^{N} w'(x)}. \tag{12}
\]

If a generalized bell-type membership function is used, the parameters for the \( j \text{th} \) membership function of the \( i \text{th} \) fuzzy rule are determined with the following equations:

\[
a'_i'(t+1) = a'_i(t) - k \cdot \frac{\partial E}{\partial a'_i},
\]

\[
b'_i'(t+1) = b'_i(t) - k \cdot \frac{\partial E}{\partial b'_i},
\]

\[
c'_i'(t+1) = c'_i(t) - k \cdot \frac{\partial E}{\partial c'_i}. \tag{13}
\]

For a Gaussian-type membership function, the parameters of the \( j \text{th} \) membership function of the \( i \text{th} \) fuzzy rule are calculated with:

\[
\sigma'_i'(t+1) = \sigma'_i(t) - k \cdot \frac{\partial E}{\partial \sigma'_i},
\]

\[
c'_i'(t+1) = c'_i(t) - k \cdot \frac{\partial E}{\partial c'_i}. \tag{14}
\]

After the four controllers (Controller 1 for increasing current, Controller 2 for constant current, Controller 3 for decreasing current and Controller 4 for null current) have been obtained, they must be integrated, resulting in the logical scheme in Fig. 3.

The decision to use one of the four controllers depends on the current vector type (increasing, decreasing, constant or zero) and on the “k” variable value. Depending on the value of “k”, we may decide if a constant current value is part of an increasing vector or part of a decreasing vector. The initial “k” value is equal to 1 when Controller 1 is used, and is equal to 0 when Controllers 2, 3 or 4 are used.
IV. The integrated controller design and evaluation

In the first phase, the “genfis2” Matlab function ([3]) was used to generate and train the FISs associated with the four controllers in Fig. 3: “ElongationFis” (for the current increase phase), “cElongationFis” (for the constant phase of the current), “dElongationFis” (for the decrease phase of the current) and “d0ElongationFis” (for the null values of the current obtained after the decrease phase). The FISs are trained for different epochs (100,000 for the first FIS, 200,000 epochs for the second and the last FISs, and 10,000 epochs for the third) using the “ANFIS” Matlab function. Figure 4 displays the deviation between the neuro-fuzzy models and the experimentally obtained data for different training epochs, defining the quality parameter from the training algorithm. A rapid decrease in the deviation between the experimental data and the neuro-fuzzy model is apparent for all four FISs in terms of the quality parameter within the training algorithm over the first $10^3$ training epochs. Evaluating each of the four FISs for the experimental data using the “evalfis” command, the characteristics shown in Fig. 5 were obtained. The means of the relative absolute values of the errors for all four FISs are: 0.41085%, 5.59708%, 0.00347% and 3.52328% for ElongationFis, cElongationFis, dElongationFis and d0ElongationFis, respectively. The error obtained for the third FIS (“dElongationFis”) is very good, and so this FIS will be considered for implementation in the Simulink integrated controller. The first, second and fourth FISs have large error values and so the generating method must be changed.
Fig. 4 Training errors for the FISs generated and trained in the first phase

Fig. 5 FISs’ evaluation as a function of the number of experimental data points in the first phase
During the second phase, the “genfis1” Matlab function ([3]) can be used to build and train the remaining three fuzzy inference systems: “ElongationFis”, “cElongationFis”, and “d0ElongationFis”. The number of membership functions considered for each of these is 6 for the first input and 12 for the second input. The number of the training epochs considered for the three FISs are 10,000 for the first and second FISs, and 1,000 for “d0ElongationFis”. Following the evaluation of these three trained FISs for experimental data, the characteristics depicted in Fig. 6 were obtained. The evolution of the training errors is represented in Fig. 7. Evaluation of these three FISs gives the following values of the mean of the relative absolute errors: 0.26909%, 0.26595%, and 1.66242% for “ElongationFis”, “cElongationFis”, and “d0ElongationFis”, respectively.

![Fig. 6 FISs’ evaluation as a function of the number of experimental data points in the second phase](image)

The errors obtained in the second phase for the first and the second FISs are very good, and so these FISs can be implemented in the Simulink integrated controller. For the last FIS (“d0ElongationFis”), the error values are still too large, and so the number of the membership functions used to generate it must be adjusted. Therefore, a third phase of FISs building and training is reserved to obtain a better solution for the “d0ElongationFis” fuzzy inference system. In this phase, two cases were considered for the numbers of the membership functions. In the first case, the
mf numbers are 12 for the first input and 12 for the second input, and in the second case the mf number is 12 for the first input and 14 for the second. A number of 4000 training epochs were considered in the first case, and 1000 in the second. The training errors for both cases, after training with the “ANFIS” function, are presented in Fig. 8, and the evaluation as a function of the number of experimental data points is shown in Fig. 9. The means of the relative absolute error values for the two cases are: 0.74844% and 0.60746 %, respectively. Since the errors in the second case are lower, that is the configuration that was chosen to be implemented in a Simulink integrated controller.

Fig. 7 Training errors for the three FISs generated and trained in the second phase

The final values of the relative absolute errors for the four generated and trained FISs are: 0.26909% for “ElongationFis”, 0.26595% for “cElongationFis”, 0.00347% for “dElongationFis”, and 0.60746% for “d0ElongationFis”.

Representing the elongations (those obtained experimentally and by using the four FIS models) as functions of electrical current for the first and third FISs, and as a function of time for the other two FISs, produces the graphics in Fig. 10. The curves are represented for all five cases of the SMA load. One can easily observe that, through training, the FISs’ model the experimental data very well, and the SMA has different thermal constants, depending on the force value.
A good overlapping of the FIS models’ elongations with the elongation experimental data is clearly visible in Fig. 10. This superposition is dependent on the number of training epochs, and improves as the number of training epochs is higher. Because the training errors of all of the trained FISs ultimately take constant values, an improved approximation of the real model can be achieved with the neuro-fuzzy methods only when a higher quantity of experimental data is used.

To visualize the FIS’s features, the Matlab “anfisedit” command ([3]) is used, followed by the FIS’s importation on the interface level. The resulting surfaces for all four final, trained FISs are presented in Fig. 11. The parameters of the input’s membership functions for each of the four FIS’s before and after training are shown in Tables 1 and 2, respectively. For the generalized bell-type membership functions, produced with the “genfis1” function, the parameters are the membership function center (c) defining their position, and a, b which define their shape. For the Gaussian-type membership functions, generated with the “genfis2” function, the parameters are one-half of the dispersion (σ/2) and the center of the membership function (c). For our system, a set of 72 rules for “ElongationFis”...
and another 72 for “cElongationFis”, 6 rules for “dElongationFis” and 168 rules for “d0ElongationFis” are generated.

Table 1 Parameters of the FIS input’s membership functions before training

<table>
<thead>
<tr>
<th>ElongationFis</th>
<th>cElongationFis</th>
<th>dElongationFis</th>
<th>d0ElongationFis</th>
</tr>
</thead>
<tbody>
<tr>
<td>a  b  c</td>
<td>a  b  c</td>
<td>a  b  c</td>
<td>a  b  c</td>
</tr>
<tr>
<td>mf1 7.72 2 120.19 0.22 2 0.5 9.11 2 119.39 1.22 2 0 16.4 142.2 0.97 5 6.78 2 83.31 5.4 2 0</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>mf2 7.72 2 135.65 0.22 2 0.95 9.11 2 137.61 1.22 2 2.44 16.4 141.9 0.97 0 6.78 2 96.88 5.4 2 10.81</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>mf3 7.72 2 151.11 0.22 2 1.4 9.11 2 155.83 1.22 2 4.89 16.4 203.8 0.97 5.5 6.78 2 110.45 5.4 2 21.63</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>mf4 7.72 2 166.56 0.22 2 1.86 9.11 2 174.05 1.22 2 7.33 16.4 214.4 0.97 0 6.78 2 124.02 5.4 2 32.45</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>mf5 7.72 2 182.02 0.22 2 2.31 9.11 2 192.77 1.22 2 9.78 16.4 177.9 0.97 -0.01 6.78 2 137.58 5.4 2 43.26</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>mf6 7.72 2 197.48 0.22 2 2.77 9.11 2 210.49 1.22 2 12.23 16.4 186.6 0.97 5 6.78 2 151.15 5.4 2 54.08</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>mf7 - - - 0.22 2 3.22 - - - 1.22 2 14.67 - - - - - 6.78 2 164.72 5.4 2 64.90</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>mf8 - - - 0.22 2 3.68 - - - 1.22 2 17.12 - - - - - 6.78 2 178.29 5.4 2 75.72</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>mf9 - - - 0.22 2 4.13 - - - 1.22 2 19.57 - - - - - 6.78 2 191.86 5.4 2 86.53</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>mf10 - - - 0.22 2 4.59 - - - 1.22 2 22.01 - - - - - 6.78 2 205.42 5.4 2 97.35</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>mf11 - - - 0.22 2 5.04 - - - 1.22 2 24.46 - - - - - 6.78 2 218.99 5.4 2 108.17</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>mf12 - - - 0.22 2 5.50 - - - 1.22 2 26.90 - - - - - 6.78 2 232.56 5.4 2 118.99</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>mf13 - - - - - - - - - - - - - - - - - - - - - - - - - - 5.4 2 129.81</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>mf14 - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - 5.4 2 140.62</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Fig. 10 FIS evaluations as functions of current or time

<table>
<thead>
<tr>
<th>ElongationFis</th>
<th>cElongationFis</th>
<th>dElongationFis</th>
<th>d0ElongationFis</th>
</tr>
</thead>
<tbody>
<tr>
<td>a  b  c</td>
<td>a  b  c</td>
<td>a  b  c</td>
<td>a  b  c</td>
</tr>
<tr>
<td>a  b  c</td>
<td>a  b  c</td>
<td>a  b  c</td>
<td>a  b  c</td>
</tr>
<tr>
<td>mf1 7.72 2 120.19 0.22 2 0.5 9.11 2 119.39 1.22 2 0 16.4 142.2 0.97 5 6.78 2 83.31 5.4 2 0</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>mf2 7.72 2 135.65 0.22 2 0.95 9.11 2 137.61 1.22 2 2.44 16.4 141.9 0.97 0 6.78 2 96.88 5.4 2 10.81</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>mf3 7.72 2 151.11 0.22 2 1.4 9.11 2 155.83 1.22 2 4.89 16.4 203.8 0.97 5.5 6.78 2 110.45 5.4 2 21.63</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>mf4 7.72 2 166.56 0.22 2 1.86 9.11 2 174.05 1.22 2 7.33 16.4 214.4 0.97 0 6.78 2 124.02 5.4 2 32.45</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>mf5 7.72 2 182.02 0.22 2 2.31 9.11 2 192.77 1.22 2 9.78 16.4 177.9 0.97 -0.01 6.78 2 137.58 5.4 2 43.26</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>mf6 7.72 2 197.48 0.22 2 2.77 9.11 2 210.49 1.22 2 12.23 16.4 186.6 0.97 5 6.78 2 151.15 5.4 2 54.08</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>mf7 - - - 0.22 2 3.22 - - - 1.22 2 14.67 - - - - - 6.78 2 164.72 5.4 2 64.90</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>mf8 - - - 0.22 2 3.68 - - - 1.22 2 17.12 - - - - - 6.78 2 178.29 5.4 2 75.72</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>mf9 - - - 0.22 2 4.13 - - - 1.22 2 19.57 - - - - - 6.78 2 191.86 5.4 2 86.53</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>mf10 - - - 0.22 2 4.59 - - - 1.22 2 22.01 - - - - - 6.78 2 205.42 5.4 2 97.35</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>mf11 - - - 0.22 2 5.04 - - - 1.22 2 24.46 - - - - - 6.78 2 218.99 5.4 2 108.17</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>mf12 - - - 0.22 2 5.50 - - - 1.22 2 26.90 - - - - - 6.78 2 232.56 5.4 2 118.99</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>mf13 - - - - - - - - - - - - - - - - - - - - - - - - - - 5.4 2 129.81</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>mf14 - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - 5.4 2 140.62</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
### Table 2 Parameters of the FIS input’s membership function after training

<table>
<thead>
<tr>
<th></th>
<th>ElongationFis</th>
<th>cElongationFis</th>
<th>dElongationFis</th>
<th>d0ElongationFis</th>
</tr>
</thead>
<tbody>
<tr>
<td>a</td>
<td>b</td>
<td>c</td>
<td>a</td>
<td>b</td>
</tr>
<tr>
<td>mf1</td>
<td>9.27</td>
<td>2.1</td>
<td>112.4</td>
<td>6.51</td>
</tr>
<tr>
<td>mf2</td>
<td>9.29</td>
<td>2.4</td>
<td>129.7</td>
<td>5.859</td>
</tr>
<tr>
<td>mf3</td>
<td>9.45</td>
<td>2.8</td>
<td>148.7</td>
<td>7.036</td>
</tr>
<tr>
<td>mf4</td>
<td>9.46</td>
<td>2.0</td>
<td>166.6</td>
<td>6.391</td>
</tr>
<tr>
<td>mf5</td>
<td>6.62</td>
<td>2.9</td>
<td>182</td>
<td>6.39</td>
</tr>
<tr>
<td>mf6</td>
<td>9.51</td>
<td>4.2</td>
<td>200</td>
<td>6.3914</td>
</tr>
<tr>
<td>mf7</td>
<td>-</td>
<td>-</td>
<td>6.3917</td>
<td>2.002</td>
</tr>
<tr>
<td>mf8</td>
<td>-</td>
<td>-</td>
<td>6.3918</td>
<td>2.0012</td>
</tr>
<tr>
<td>mf9</td>
<td>-</td>
<td>-</td>
<td>6.3918</td>
<td>2.0007</td>
</tr>
<tr>
<td>mf10</td>
<td>-</td>
<td>-</td>
<td>6.3919</td>
<td>2.0005</td>
</tr>
<tr>
<td>mf11</td>
<td>-</td>
<td>-</td>
<td>6.3919</td>
<td>2.0003</td>
</tr>
<tr>
<td>mf12</td>
<td>-</td>
<td>-</td>
<td>6.392</td>
<td>2.002</td>
</tr>
<tr>
<td>mf13</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>mf14</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
</tbody>
</table>

**Fig. 11** The surfaces produced for all four of the final trained FISs
Comparison of the FISs’ characteristics and membership functions’ parameters before and after training, from Tables 1 and 2, indicates a redistribution of the membership functions in the working domain and a change in their shapes, by modification of the $a$, $b$, and $\sigma$ parameters. According to the parameter values from Table 1, generating FISs with the “genfis1” and “genfis2” functions primarily results in the same values for the $a$, $b$, and $\sigma/2$ parameters for all of the membership functions that characterize an input. A secondary result is the separation of the working space for the respective input using a grid partition on the data (no clustering) if the “genfis1” function is used, or using the fuzzy subtractive clustering method if generating with the “genfis2” function.

For the “dElongationFis” fuzzy inference system (initially generated by using the “genfis2” function) the rules are of the type: if (in1 is in1cluster"k") and (in2 is in2cluster"k") then (out1 is out1cluster"k"). For both of the inputs of this FIS, six Gaussian-type membership functions (mf) were generated; within the set of rules they are noted by: in"j"cluster"k"; $j$ is the input number (1÷2), and $k$ is the number of the membership function (1÷6). The “dElongationFis” fuzzy inference system has the structure shown in Fig. 12, while the corresponding controller (Controller 3) has the structure presented in Fig. 13.
For the other three FISs (initially generated by using the “genfis1” function) the rules are of the type: if (in1 is in1mf“k”) and (in2 is in2mf“p”) then (out1 is out1mf“r”). The number of the output membership functions (mf) is \(k \times p (r=1+(kp))\) and is equal to the number of rules. For these three FISs, generalized bell-type membership functions were generated; within the sets of rules they are noted by: in1”j”mf“n”; \(j\) is the input number (1÷2), and \(n\) is the number of the membership functions. For “ElongationFis” and “cElongationFis”, six membership functions for the first input \((k=6)\) and 12 membership functions for the second input \((p=12 \rightarrow r=72)\) are produced. The “d0ElongationFis” results in 12 membership functions for the first input \((k=12)\), and 14 for the second input \((p=14 \rightarrow r=168)\). For example, the “ElongationFis” fuzzy inference system has the structure shown in Fig. 14, while the corresponding controller (Controller 1) has the same structure as Controller 3 (see Fig. 13).

![Fig. 14 Structure of the “ElongationFis” fuzzy inference system](image)

Each of the four FISs is imported at the fuzzy controller level, resulting in four controllers: Controller 1 (“ElongationFis”), Controller 2 (“cElongationFis”), Controller 3 (“dElongationFis”), and Controller 4 (“d0ElongationFis”). These four controllers are integrated using the logical scheme given in Fig. 3; the Matlab/Simulink model in Fig. 15 is the result.

In the Matlab/Simulink model shown in Fig. 15, the second input of Controller 2 and that of Controller 4 (Time) are generated by using integrators, starting from the moment that these inputs are used in Controller 2 or Controller 4 (the input of the Gain block is 0 if the schema decides not to work with one of the Controllers 2 or 4). It is possible that the simulation sample time may be different than the sample time used in the experimental data acquisition process, and therefore we use the “Gain” block that gives their ratio; “Te” is the sample time in the experimental
data and “T” is the simulation sample time. In the schema, the constant “C” represents the maximum time considered for the actuator to recover its initial temperature (approximately 24°C) when the current becomes 0A.

![Fig. 15 The integration model schema in Matlab/Simulink](image)

Evaluating the integrated controller model (see Fig. 15) for all five experimental data cases produces the results shown in Figs. 16 and 17. These graphics show the elongations versus the number of experimental data points and versus the applied electrical current, respectively, using the experimental data and the integrated neuro-fuzzy controller model for the SMA. A good overlapping of the outputs of the integrated neuro-fuzzy controller with the experimental data can be easily observed.

The same observation can be made from the 3D characteristics of the experimental data and the neuro-fuzzy modeled data in terms of temperature, elongation and force, depicted in Fig. 18 a, and in terms of current, elongation and force, depicted in Fig. 18 b.

The mean values of the relative absolute errors of the integrated controller for the five load cases of the SMA actuator, based on adaptive neuro-fuzzy inference systems, are: 0.45997% for 120 N, 0.50295% for 140 N,
0.51319% for 150 N, 0.71609% for 180 N and 0.50775% for 190 N. The mean value of the relative absolute error between the experimental data and the outputs of the integrated controller is 0.54%.

Fig. 16 Elongations versus the number of experimental data points

Fig. 17 Elongations versus the applied electrical current
V. Conclusions

In this paper, an integrated controller based on adaptive neuro-fuzzy inference systems for modeling smart material actuators was obtained. The direct application of this controller is in a morphing wing system. The general aim of the smart material actuators’ desired model is to calculate the elongation of the actuator under the application of a thermo-electro-mechanical load for a certain time. Therefore, the smart material actuators were experimentally tested in conditions close to those in which they will be used. Testing was performed for five load cases, with forces of 120 N, 140 N, 150 N, 180 N and 190 N. Using the experimental data, four Fuzzy Inference Systems were generated and trained to obtain four neuro-fuzzy controllers: one controller for the current increase (“ElongationFis”), one for a constant current (“cElongationFis”), one for the current decrease (“dElongationFis”), and one controller for the null current, after its decrease (“d0ElongationFis”). The “genfis1” and “genfis2” Matlab functions were used to generate the initial FISs, and the adaptive neuro-fuzzy inference system technique was then used to train them. The final values of the relative absolute errors for the four generated and trained FISs were: 0.26909% for “ElongationFis”, 0.26595% for “cElongationFis”, 0.00347% for “dElongationFis”, and 0.60746 % for “d0ElongationFis”.

Each of the four obtained and trained FISs were imported at the fuzzy controller level, resulting in four controllers. Finally, these four controllers were integrated by using the logical scheme given in Fig. 3; resulting in the Matlab/Simulink model for the integrated controller shown in Fig. 15. The integrated controller performances were evaluated for all five load cases; the values obtained for the mean relative absolute errors were: 0.45997% for 120 N, 0.50295% for 140 N, 0.51319% for 150 N, 0.71609% for 180 N and 0.50775% for 190 N. Thus, the mean
value of the relative absolute error between the experimental data and the outputs of the integrated controller was 0.54%.

A particular advantage of this new model is its rapid generation, thanks to the “genfis1”, “genfis2” and “ANFIS” functions already implemented in Matlab. The user need only assume the four FIS’s training performances using the “anfisedit” interface generated with Matlab.

Acknowledgments

We would like to thank for the received funds which make possible this research to CRIAQ (Consortium of Research in the Aerospatial Industry in Quebec), Thales Avionics, Bombardier Aerospace and the NSERC (National Sciences and Engineering Research Council). Many thanks are due to Professors Vladimir Brailovski and Patrick Terriault who gave us the opportunity to use their SMA bench test equipment for the realisation of this new controller method. We mainly thank to George Henri Simon for initiating the CRIAQ 7.1 project as well as to Philippe Molaret from Thales Avionics and to Eric Laurendeau from Bombardier Aeronautics for their collaboration on this work.

References


