NASA Technical Memorandum 107236

A Comparison of Neural Networks and Fuzzy Logic Methods for Process Modeling

Krzysztof J. Cios and Dorel M. Sala
University of Toledo
Toledo, Ohio

Laszlo Berke
Lewis Research Center
Cleveland, Ohio

July 1996

National Aeronautics and
Space Administration

A Comparison of Neural Networks and Fuzzy Logic Methods for Process Modeling

Krzysztof J. Cios [†] , Dorel M. Sala [†] and Laszlo Berke [‡]
[†] University of Toledo and [‡] NASA Lewis Research Center

Abstract

The goal of this work was to analyze the potential of neural networks and fuzzy logic methods to develop approximate response surfaces as process modeling, that is for mapping of input into output. Structural response was chosen as an example. Each of the many methods surveyed are explained and the results are presented. Future research directions are also discussed.

Introduction

Neural networks and fuzzy logic methods have been enjoying vigorous developments. They are well suited for development of computable models for complex processes given sufficient data for the correspondence between input and output variables of the process at hand. The principal goal of this investigation was to compare the accuracy of approximation of these two methods on the same set of data. Fuzzy logic models are limited to only about a half dozen variables because of computational explosion. The value of these approaches is in cases where no computable analytic model exists, only experimental data . For this investigation, in order to generate well controllable data sets, a conveniently computable set was chosen in the form of the response of a small plane truss with well known behavior characteristics. The methods are applicable for physical phenomena for which only experiments can provide reliable data sets. In those cases fast executing computable models are often still desirable, for example for optimization. It should be noted that the two methods, neural networks and fuzzy logic are fundamentally different. The neural networks produce the trained model

as a "black box" of weights associated with the network topology. The fuzzy logic methods, on the other hand produce expert rules captioning the behavior of the system. These rules, in general, can be examined by the experts, modified if needed, and validated on new data. Another case studied was a model from output to input to simulate inverse behavior for reverse engineering for which the inverse of the computable models are not available.

The chosen simple structural example is the well known ten bar truss used intensively in optimization research and literature. Despite the small number of variables the behavior characteristics of statically indeterminate structures, the nonlinear dependence of internal load distribution on relative member size, is sufficiently represented.

The ten-bar truss is depicted in Figure 1. The data for neural networks (NN) and fuzzy logic (FL) methods were supplied in the form of numerical input-output pairs generated by the finite element method.

$E=10^7$ [psi]
$A_1=4.51$
$A_2=2.16$
$A_3=0.1$
$A_4=0.1$
$A_5=4.38$
$A_6=0.1$
$A_7=3.05$
$A_8=3.23$
$A_9=0.1$
$A_{10}=3.05$ [in^2]

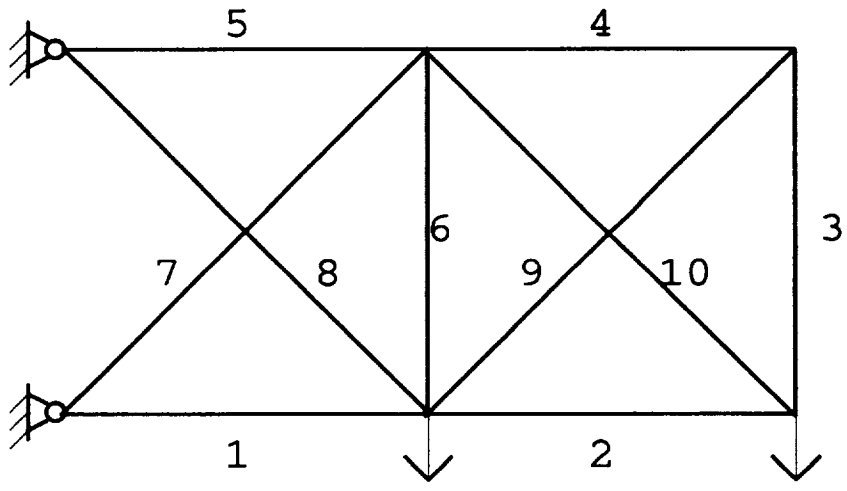

Figure 1. Ten bar truss

The following three types of NN are considered. The linear feedforward

network, the radial basis function (RBF) network, which although needed more time for training than the linear, gave best results in terms of error measures, and the multi-layered feedforward network using backpropagation.

In choosing the fuzzy system we had to deal with the fact that most of the FL applications described in the literature are related to control systems, and often involve just a few input variables. In addition, these methods use heuristic rules which could hardly be generalized. As a general characteristic, FL needs large training sets and training times longer than NN. Also, with the exception of the Sugeno type systems fuzzy logic methods produced, on average, larger errors for the case problem studied. On the other hand, the Sugeno type fuzzy system was not only the best from all FL methods investigated but was also comparable with best results achieved by NN in terms of errors. The drawback of the Sugeno type fuzzy system, however, is the use in the training phase of a very large matrix which needs to be inverted. We were able to circumvent this problem by reducing the number of IF ... THEN rules by clustering the outputs, and partitioning the input space accordingly.

A word on the method of calculating the errors. In order to compare the results the following formula was used for error evaluation. Namely, for each point the error was calculated as an overall error:

$$\varepsilon\% = \sqrt{\frac{\Sigma (y_i - z_i)^2}{\Sigma y_i^2}}\ 100$$

y_i - the target value

z_i - the output value

This method of calculating the error may be misleading in the sense that the overall error can be small even though individual errors may be relatively quite large.

Neural Networks Methods

The approximation problem can be stated as follows : Given a continuous function f(X), defined on a set X, and an approximating function, F(W,X),

continuous in W and X, determine W such that the difference between f(x) and F(W,X) is minimal.

Depending on the choice of the approximating function several cases [1] can be considered :

1.- the classical linear case

$$F(W,X) = W * X$$

where W is an m-by-n matrix and X is an n-dimensional vector. It corresponds to a feedforward network without hidden units.

2.- the nested sigmoids scheme which can be written as

$$F(W,X) = \sigma(\Sigma\, w_n \sigma (\Sigma\, w_i \sigma (\ldots\ \sigma(\Sigma\, w_j \sigma) \ldots)))$$

where σ is a sigmoid function. It corresponds to a multi-layer network of neurons that sum their inputs with weights w_n, w_i, w_j , and then perform a sigmoidal transformation of this sum. This type is used with the backpropagation learning.

Another approach to an approximation problem is to use interpolation [2] : Given N different input points $\{ x_i, i=1, \ldots N\}$ and N output points $\{y_i, i=1, \ldots N\}$ find a function F satisfying the interpolation conditions

$$F(x_i) = y_i \qquad i=1 , \ldots , N$$

The radial basis function method belongs to this category. It requires choosing function F of the following form :

$$F(x) = \Sigma\, a_i f(\|x-x_i\|) + \Sigma\, b_j g_j(x) \qquad m \le n,$$
$$i=1, \ldots N \quad j=1, \ldots m$$

where f is a continuous function, called radial basis function , $\| \bullet \|$ is the Euclidean norm , $\{ g_i \}_{i= 1,\ldots m}$ is a basis of the linear space of algebraic polynomials of degree at most k-1, with k given, and a_i and b_j are scalar coefficients.

1. Linear feedforward network

It is the simplest neural network which can be solved directly by using matrix algebra. Suppose we have a set of input-output pairs, X_i - Y_i , $i = 1, \ldots N$, where $X_i \in \mathbb{R}^n$ and $Y_i \in \mathbb{R}^m$. Then the network is described by the equations :

$$W*X_i = Y_i \qquad i = 1, \ldots N$$

where W is the weight matrix of dimension m*n which can be found using the least square error method :

$$W = Y * X^t * (X X^t)^{-1}$$

where $X = [\, X_1\ X_2 \ldots X_N \,]$ is an n-by-N matrix of input vectors and $Y = [\, Y_1\ Y_2 \ldots Y_N \,]$ is an m-by-N matrix of output vectors.

This method is good for approximating linear or quasilinear functions. It is computationally efficient for moderate values of N. A drawback of the method is the possibility of $X X^t$ being singular.

2. Multi-layered network with sigmoidal units

2.1. *Gradient descent learning.*

The sigmoidal transfer function is given by :

$$\sigma(x) = 1/(1 + e^{-x})$$

and has values between 0 and 1. Therefore if the outputs take on different values we need to scale them to the interval (0,1) and then rescale them back. Instead of doing that, one can scale the sigmoidal function to the outputs range. For example, if the output varies between y_{min} and y_{max} the scaled sigmoidal function becomes [3] :

$$\sigma(x) = y_{min} + (y_{max} - y_{min}) \ / \ (1 + e^{-x})$$

A neural network with one hidden layer using the above sigmoidal function was trained using the gradient descent technique.

2.2. *Learning using Levenberg-Marquardt algorithm.*

This algorithm is a combination of Newton' s method and the steepest descent method [4]. It is not as powerful as Newton's method but it is better than the steepest descent. The update of the weight matrix is given by :

$$\boldsymbol{dw} = (\boldsymbol{J}^t \boldsymbol{J} + \lambda \boldsymbol{I})^{-1} \boldsymbol{J}^t \boldsymbol{e}$$

where J represents the Jacobian of derivatives of error $\boldsymbol{e}$ with respect to weights $\boldsymbol{w}$, $\boldsymbol{I}$ is the identity matrix, and λ is a parameter which controls the search direction. When λ is zero the Levenberg-Marquardt (L-M) algorithm reduces to Newton's method and when λ is very large L-M approximates the steepest descent method. The L-M algorithm requires large memory, however.

3. RBF neural network

Radial basis function network can be designed very quickly, especially when compared with backpropagation networks. There are two different ways of designing an RBF network [5]. One creates as many neurons as there are training vectors, and the other finds the smallest network architecture iteratively by adding one neuron at a time until the error goal is met. In any case, the RBF network has two layers of neurons, the first one uses the radial basis functions and the second one is a linear network. The output is given by:

$$Y = W * F(V*X)$$

where
- W is the augmented weight vector of the output linear layer
- $F(X) = exp(\|X-C_i\|^2 / 2\sigma^2)$ is the Gaussian radial basis function with center C_i, and spread σ.
- V is the weight vector of the first layer

The only parameter to be specified is the spread constant. If it is chosen too small the ability of the network to generalize decreases. On the other hand, if it is chosen too big the information content of the training data is lost as all the neurons will output large values (near 1) for all the inputs.

Fuzzy Logic Methods

As we have seen neural networks map data points in the input space to points in the output space. The mapping is point to point. On the other hand, fuzzy sets perform set to set mapping. This fact is a major difference between the two techniques and cause of FL computational difficulties. The advantage of using FL methods is that the generated IF ... THEN ... rules can be understood and investigated by human experts. The mapping is achieved by first performing point-to-set (fuzzify) conversion of inputs. At the end a set-to-point (defuzzify) conversion of the outputs is usually performed.

There are two main categories of fuzzy systems. One is based on the Sugeno method of reasoning. These models are formed by logical rules of IF (*antecedent*) ... THEN (*consequent*) type that have a fuzzy antecedent part and a functional non-fuzzy consequent. The second category is based on collections of IF...THEN rules and use of fuzzy reasoning. In these models fuzzy quantities are associated with linguistic labels.

In order to develop a fuzzy system one needs to define its three main components :

- the input fuzzy sets or the antecedents of the rules
- the outputs or the consequents
- the inference mechanism

Unfortunately, there is no general method of constructing a fuzzy system, and many of them are based on an a priori knowledge provided by a human expert. In our case, the information is given only in the form of input-output data. Thus, we shall use the following two approaches to define fuzzy sets. One is to define a fuzzy set for each point being centered around that point, and the other is to define equally shaped fuzzy sets covering the entire ranges of the corresponding variables. The antecedent parts of the rules are represented by input fuzzy sets and the consequent parts by the output fuzzy sets. There can be several combinations of inputs and outputs, for instance input fuzzy sets may be defined for each input point and output fuzzy sets may be equally shaped. An inference

mechanism is related to the way in which the fuzzy sets are defined.

One of the problems, especially when the second approach is used to construct fuzzy sets is that of completeness of the fuzzy set system. It becomes critical with the increasing number of input variables.

Let us explain what we mean by completeness. Given a fuzzy set partition $F_{i1,i2,..,in} = F_{i1}xF_{i2}x...xF_{in}$ of a space $I = [a_1,b_1]x[a_2,b_2]x...x[a_n,b_n] \subset \mathbb{R}^n$, we say that the fuzzy sets form a complete partition on I, if for any $\mathbf{x}=(x_1,x_2,...,x_n) \in I$ there exists $F_{i1,i2,..,in}$ such that $\mu_{F_{i1,i2,...,in}}(\mathbf{x})>0$ [6]. This requirement of completeness must be reflected in the set of rules, otherwise the fuzzy sets not covered by any rule are invisible to the system [7]. For example, consider a simple fuzzy system, with the input space [0,1] x [0,1] partitioned into fuzzy sets, as shown in Figure 2, and described by the set of rules:

Rule 1: *IF* x_1 *is* F_{11} *and* x_2 *is* F_{12} *THEN* y *is* G_1
Rule 2: *IF* x_1 *is* F_{11} *and* x_2 *is* F_{22} *THEN* y *is* G_2
Rule 3: *IF* x_1 *is* F_{11} *and* x_2 *is* F_{31} *THEN* y *is* G_3
Rule 4: *IF* x_1 *is* F_{21} *and* x_2 *is* F_{12} *THEN* y *is* G_4
Rule 5: *IF* x_1 *is* F_{21} *and* x_2 *is* F_{22} *THEN* y *is* G_5
Rule 6: *IF* x_1 *is* F_{21} *and* x_2 *is* F_{31} *THEN* y *is* G_6

As we can see set F_{31} is not covered by any rule. Thus, for instance for $x_1 \in [.75, 1]$ and any x_2, after calculating the membership function of the antecedents for all the rules we obtain :

$$\text{Rule1:} \quad \mu_{F_{11,12}}(X) = \mu_{F_{11}}(x_1) \wedge \mu_{F_{12}}(x_2) = 0 \wedge \mu_{F_{12}}(x_2) = 0$$
$$\text{Rule2:} \quad \mu_{F_{11,22}}(X) = \mu_{F_{11}}(x_1) \wedge \mu_{F_{22}}(x_2) = 0 \wedge \mu_{F_{22}}(x_2) = 0$$
$$\text{Rule3:} \quad \mu_{F_{11,32}}(X) = \mu_{F_{11}}(x_1) \wedge \mu_{F_{32}}(x_2) = 0 \wedge \mu_{F_{32}}(x_2) = 0$$
$$\text{Rule4:} \quad \mu_{F_{21,12}}(X) = \mu_{F_{21}}(x_1) \wedge \mu_{F_{12}}(x_2) = 0 \wedge \mu_{F_{12}}(x_2) = 0$$
$$\text{Rule5:} \quad \mu_{F_{21,22}}(X) = \mu_{F_{21}}(x_1) \wedge \mu_{F_{22}}(x_2) = 0 \wedge \mu_{F_{22}}(x_2) = 0$$
$$\text{Rule6:} \quad \mu_{F_{21,32}}(X) = \mu_{F_{21}}(x_1) \wedge \mu_{F_{32}}(x_2) = 0 \wedge \mu_{F_{32}}(x_2) = 0$$

In terms of fuzzy rules, the completeness requirement may be stated as follows : for any value of the input variable the antecedent part for at least one rule must be nonzero.

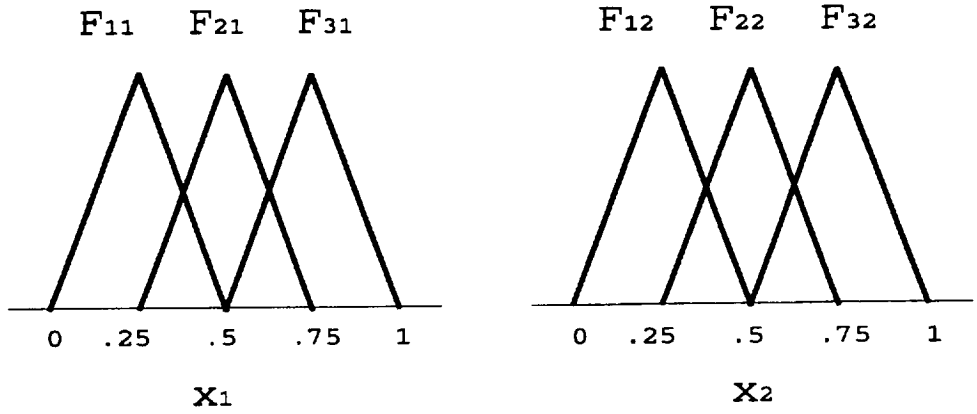

Figure 2. Fuzzy sets partition of x_1 and x_2

1. Sugeno fuzzy system

In this form of a fuzzy system the output is determined by a parameterized function of inputs, not by fuzzy values. If we consider a multi-input single-output mapping the Sugeno system [8] can be described as follows :

R^1 : IF x_1 is F_1^1 and x_2 is F_2^1 and ... and x_n is F_n^1
THEN $y^1 = p_0^1 + p_1^1 x_1 + ... + p_n^1 x_n$

.
.
.

R^M : IF x_1 is F_1^M and x_2 is F_2^M and ... and x_n is F_n^M
THEN $y^M = p_0^M + p_1^M x_1 + ... + p_n^M x_n$

where F_i^l are fuzzy sets, p_i are real-valued parameters, y^l is the system output

from rule R^l . The output of this fuzzy system is a weighted average :

$$y = \frac{\sum w^l y^l}{\sum w^l}$$

where the weight w^l represents the overall truth value of the premise of rule R^l and is calculated as :

$$w^l = \prod \mu_{Fi}^l(x_i)$$

$\mu_{Fi}^l(x_i)$ is the degree of membership of the crisp input x_i to the fuzzy set F_i^l . Using the following notation :

$$\alpha^i = w^i \; / \; \sum w^i$$

$$X = [\; \alpha^1 \ldots \alpha^M \;\; \alpha^1 x_1 \ldots \alpha^M x_1 \;\; \ldots \;\; \alpha^1 x_n \ldots \alpha^M x_n \;]$$

$$P = [\; p_0^1 \ldots p_0^M \;\; p_1^1 \ldots p_1^M \;\; \ldots \;\; p_n^1 \ldots p_n^M \;]^t$$

the output can be rewritten in a vector form :

$$y = X P$$

The parameter vector P is unknown, and we need (n+1)M input-output pairs in order to completely determine it. Often there are more than (n+1)M pairs in the training set and the linear system of equations is overdetermined; thus, P can be calculated using the pseudo-inverse of X :

$$P = (X^t X)^{-1} X^t Y$$

which minimizes the mean square error.

The important factors here are the number of rules and the choice of the membership functions. Usually one starts with a small number of fuzzy partitions and increases it until the error goal is reached.

2. Mamdani model

The rule base of the Mamdani system [9, 10] has the following form:

r^1: *IF* x_1 *is* F_{11} *AND ... AND* x_n *is* F_{1n} *THEN* y *is* G_1

ALSO

r^2: *IF* x_1 *is* F_{21} *AND ... AND* x_n *is* F_{2n} *THEN* y *is* G_2

ALSO

.

r^n: *IF* x_1 *is* F_{r1} *AND ... AND* x_n *is* F_{rn} *THEN* y *is* G_r

where $x_1,...,x_n$ are the input variables and y is the output variable. F_{ij} and G_i, ($i=1,r$; $j=1,n$) are fuzzy subsets of the universes of discourse of $x_1,...,x_n$ and y.

The output is constructed by superimposing the outputs of the individual rules. The AND operator in the antecedents of the rules is interpreted as a fuzzy intersection. Each individual implication, or rule, is expressed as a fuzzy relation R between the antecedent part and the consequent part. In Mamdani model the relation is interpreted as a fuzzy intersection of the antecedent and consequent fuzzy sets and for rule r^i we have :

$$R_i = (F_{i1} \cap F_{i2} \cap ... \cap F_{in}) \cap G_i$$

The ALSO connective corresponding to aggregation of the rules r^i, or fuzzy relations R_i , is accomplished through the union of the individual fuzzy relations :

$$R = \cup R_i$$

For a given input fuzzy sets $x_1=A_1$, ... , $x_n=A_n$, the fuzzy output $y=G$ is obtained by using the max-min inference rule :

$$G= (A_1 ... A_n) \circ R$$

The membership function of the inferred fuzzy set G is :

$$G(y) = (A_1(x_1) \wedge \ldots \wedge A_n(x_n)) \wedge R(x_1 \ldots x_n, y)$$
$$= \bigvee_{j=1}^{r} (A_1(x_1) \wedge \ldots \wedge A_n(x_n)) \wedge R_j(x_1 \ldots x_n, y)$$
$$= \bigvee_{j=1}^{r} (A_1(x_1) \wedge \ldots \wedge A_n(x_n)) \wedge (F_{j1}(x_1) \wedge \ldots \wedge F_{jn}(x_n)) \wedge G_j(y)$$
$$= \bigvee_{j=1}^{r} [(A_1(x_1) \wedge F_{j1}(x_1)) \wedge \ldots \wedge (A_n(x_n) \wedge F_{jn}(x_n)) \wedge G_j(y)]$$

The term

$$\mu_j = (A_1(x_1) \wedge F_{j1}(x_1)) \wedge \ldots \wedge (A_n(x_n) \wedge F_{jn}(x_n))$$

denotes the degree into which the j-th rule fires, or in other words the degree of relevance of the j-th rule . With this notation we have:

$$G(y) = \bigvee_{j=1}^{r} \mu_j \wedge G_j(y)$$

In the case the inputs are crisp values x_1^*, ... , x_n^*, the input fuzzy sets A_1, ... , A_n are considered as fuzzy singletons and μ_j is given by :

$$\mu_j = F_{j1}(x_1^*) \wedge \ldots \wedge F_{jn}(x_n^*)$$

For crisp inputs and outputs the simplified method of reasoning gives :

$$y = \frac{\sum_{j=1}^{m} \mu_j \, y_j^*}{\sum_{j=1}^{m} \mu_j}$$

where μ_j is given by the former expression.

2.1. *Fuzzy model using least squares method for learning consequents.*

The basic structure of this fuzzy model is like that of the Mamdani model.

The input space is partitioned into equally shaped fuzzy sets and the antecedents of the rules are formed by taking all the combination of input fuzzy sets. Thus, we can easily determine the degree of firing of each rule, but the consequents of all rules are not known. The consequents, or rather the consequent centroids, can be determined by first observing that the output is calculated as a linear combination of the consequents [9]:

$$Y = \mu_1 * y_1 + \mu_2 * y_2 + \ldots + \mu_3 * y_m$$

where μ_i is the normalized degree of firing of rule i and y_i is the consequent of rule i. For a set of n input-output pairs we have n such equations which can be rewritten in a matrix form :

$$\boldsymbol{Y} = \boldsymbol{\mu} * \boldsymbol{y}$$

where $\boldsymbol{Y}$ is the vector of outputs , $\boldsymbol{\mu}$ is the matrix of the degrees of firing and $\boldsymbol{y}$ is the vector of consequents. Using the least square method $\boldsymbol{y}$ is calculated from :

$$\boldsymbol{y} = (\boldsymbol{\mu}^t * \boldsymbol{\mu})^{-1} * \boldsymbol{\mu}^t * Y$$

Results

The numerical experiments were performed on the ten-bar-truss shown in Figure 1. The programs were implemented using MATLAB software on a SUN 10 Sparc station. For each example the training set consisted of 250 input-output vector pairs (IOP). The tests were performed on 200 IOP. The inputs represent cross sectional areas of the elements and the outputs represent the stresses. The training and test sets were generated with uniform distribution within +/- 10% of the nominal values of the cross-sectional areas. The errors obtained by applying each of the methods described in the paper are shown in Figures 3 through 10 .

The results for the linear feedforward neural network (FFN) are shown in Figure 3. The results for backpropagation neural network using the Levenberg-Marquardt algorithm are shown in Figure 4. Backpropagation using the standard

gradient descent technique resulted in the largest errors of all NN as can be seen in Figure 5.

Figure 6 shows the results for the radial basis functions network (RBF). The inputs were normalized to the interval [0,1]. The hidden layer, the one using radial basis functions, consisted of 30 neurons . These errors are smaller than those of any feedforward neural networks studied.

Fuzzy logic Sugeno type method couldn't be used as the matrix to be inverted became too large for the software used (using only two fuzzy sets partitions for the inputs would result in 1024 rules). To avoid this problem we used clustering of the input space, based on clustering of the output space. Then we applied the Sugeno method. The results are shown in Figure 7.

The Mamdani model was implemented using the center fuzzification method and was tested with various aggregation operators [11]. The best performance, shown in Figure 8, was obtained with the classical max-min operator. An example of the fuzzy rules obtained using Mamdani method with clustering is given below. The rules shown refer to a single output y which represents the stress in element 1. The rules for all other outputs are similar and thus are not shown here. First, the range of the output is divided into nine adjacent intervals ($Y_1,...,Y_9$) . The output fuzzy sets are constructed for these intervals. The inputs are clustered into nine clusters $C_i = \{ \mathbf{x} \ / \ y(\mathbf{x}) \in Y_i \}$, i=1,...,9. If we define by Y_{mi} , i=1,..,9 the nine output fuzzy sets then the rules are given by:

R_1: IF $\mathbf{x} \in C_1$ THEN $y \in Y_{m1}$

.

.

.

R_9: IF $\mathbf{x} \in C_9$ THEN $y \in Y_{m9}$

The degree of firing of rule Ri is given by the degree of membership of $\mathbf{x}$ to the cluster C_i.

The inference mechanism for The Mamdani model is illustrated in Figure 9,

where dotted lines indicate the nine fuzzy sets and solid lines the resultant fuzzy output. The crisp values obtained from deffuzification using the centroid and maxgrade methods are shown on the x-axis. The results in terms of the errors for the Mamdani model usin clustering are shown in Figure 10.

The results for the fuzzy model using the least squares method and clustering (in the same manner as for Mamdani model) are depicted in Figure 11. These errors are comparable with those of Mamdani model.

For each method the histogram of the probability density function and distribution function can be obtained by calling a special function. An example for element 1 of the ten bar truss for the radial basis functions network is shown in Figure 12.

Table 1 summarizes the results of this investigation. It shows the average errors for each method used. In addition, it gives details about the number of neurons used by each neural network and the number of rules used by the fuzzy systems. For the original Sugeno fuzzy system the results could not be calculated although we show the minimum number of rules which would need to be used. The results of the methods shown in Table 1 are grouped into four major categories. Two for each, neural networks and fuzzy systems. The first three results are for the feedforward neural networks. Sugeno method was used in two variations: with and without clustering and Mamdani fuzzy system was used in three variations : with and without clustering and using least square method.

Another experiment consisted of applying the reverse engineering method to determine the cross-sectional areas given the desired or prescribed stresses in the bar elements. A radial basis functions neural network was used and trained using 250 IOP patterns. This time, though, stress values were presented as inputs of and the cross-sectional areas were the outputs. The neural network consisted of two layers of neurons, one hidden layer with 30 neurons and the output layer with 10 neurons. The main purpose of this experiment was to find the ranges of the areas given a set of random stress values within the same range as that of the training set. The results of Table 2 show the original ranges in the test set, and

the ranges found by using the RBF neural network for the same data set, respectively. The corresponding errors are shown in Figure 13.

A menu driven interface was written such that the user can easily access the desired method and run it independently. Examples of the graphical user interface outputs are shown in Figure 14.

Conclusions

Both neural networks and fuzzy logic methods proved to be effective in simulating the behavior of small modeling problems. The studied case was a ten bar plain truss. While neural networks are easy to implement and train and give very good approximations, fuzzy sets need more training time and are not computationally as effective as neural networks with the exception of the Sugeno method. However, the knowledge imbedded in a fuzzy logic system can be expressed in terms of IF ... THEN ... rules which can be checked and modified by the domain expert. In case of fuzzy systems further work may involve applying machine learning methods [12,13] and genetic algorithms for learning the IF ... THEN rules and may also involve the optimization of structures using fuzzy equations.

References

[1] Poggio, T. and Girosi, F., " Networks for Approximation and Learning ", Proceedings of the IEEE, vol. 78, No. 9, Sept. 1990, pp. 1481-1495.

[2] Wu, J.-K., " Neural Networks and Simulation Methods ", Marcel Dekker : NY, 1994.

[3] Fausset, L., " Fundamentals of Neural Networks ", Prentice Hall : Englewood Cliffs, 1994.

[4] Bazaraa, M.S., Sherali, H.D., and Shetty, C.M., " Nonlinear Programming Theory and Algorithms ", 2nd ed., Wiley : NY, 1993.

[5] Haykin, S., " Neural Networks A Comprehensive Foundation ", Macmillan : NY, 1994.

[6] Zeng, X.-J. and Singh, M.G., " Approximation Theory of Fuzzy Systems - MIMO Case ", IEEE Trans. on Fuzzy Systems, vol. 3, No. 2, May 1995, pp. 219-235.

[7] Driankov, D., Hellendoorn, H. and Reinfrank, M., " An Introduction to Fuzzy Control ", Springer-Verlag : Berlin, 1993.

[8] Takagi, T. and Sugeno, M., " Fuzzy Identification of Systems and Its Applications to Modeling and Control ", IEEE Trans. on Systems, Man, and Cybernetics, vol. SMC-15, No. 1, Jan./Feb. 1985, pp. 116-132

[9] Yager, R.R. and Filev, D.P., " Essentials of Fuzzy Modeling and Control ", Wiley : NY, 1994.

[10] Pedrycz, W., " Fuzzy Control and Fuzzy Systems ", 2nd ed., Wiley : NY, 1995.

[11] Klir, G.J. and Yuan, B., " Fuzzy Sets and Fuzzy Logic. Theory and Applications ", Prentice Hall : Englewood cliffs, 1995.

[12] Cios, K.J. and Liu, N., "An Algorithm which Learns Multiple Covers via Integer Linear programming, Part I - The CLILP2 Algorithm", Kybernetes, MCB University Press, U.K., Vol. 24 (2), pp. 29-50, 1995

[13] Cios, K.J. and Liu, N., "An Algorithm which Learns Multiple Covers via Integer Linear programming, Part II - Experimental Results and Conclusions", Kybernetes, MCB University Press, U.K., Vol. 24 (3), pp. 28-40, 1995

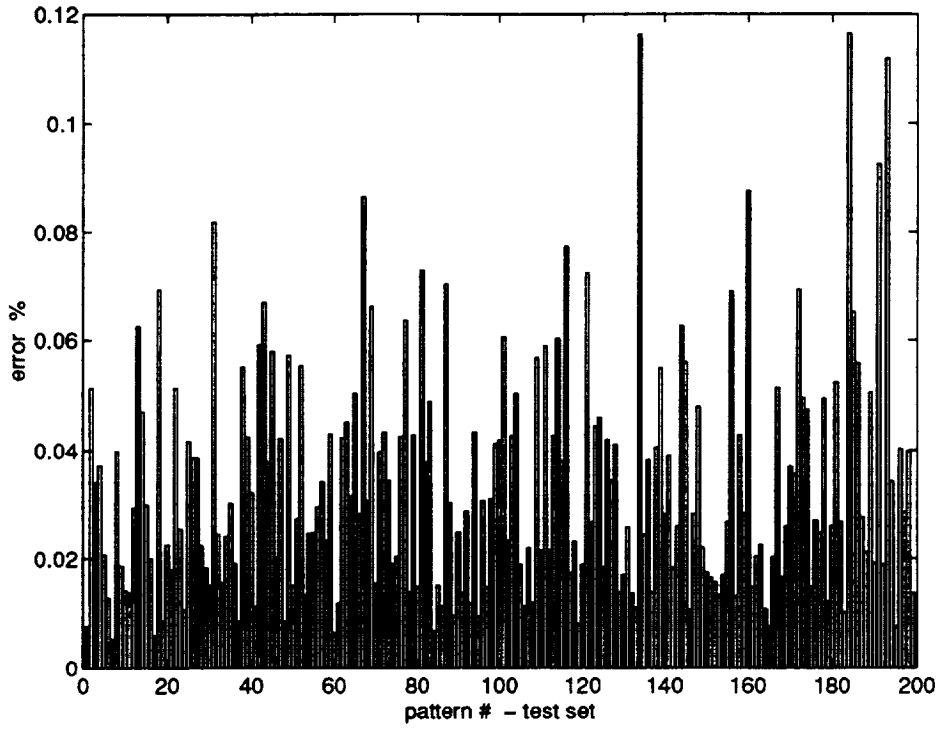

Figure 3. Results of linear feedforward neural network

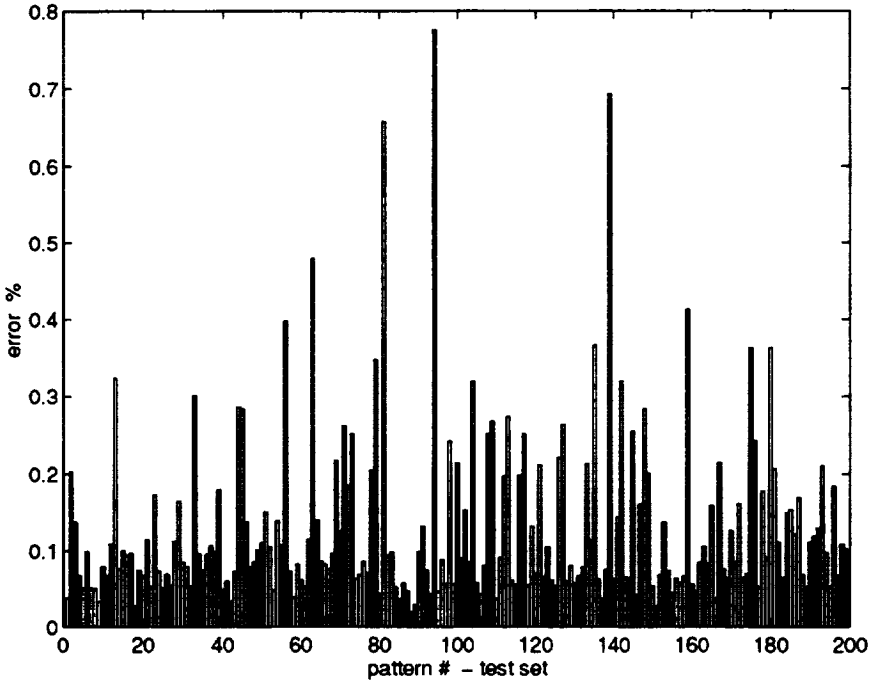

Figure 4. Results of NN using backpropagation with L-M algorithm

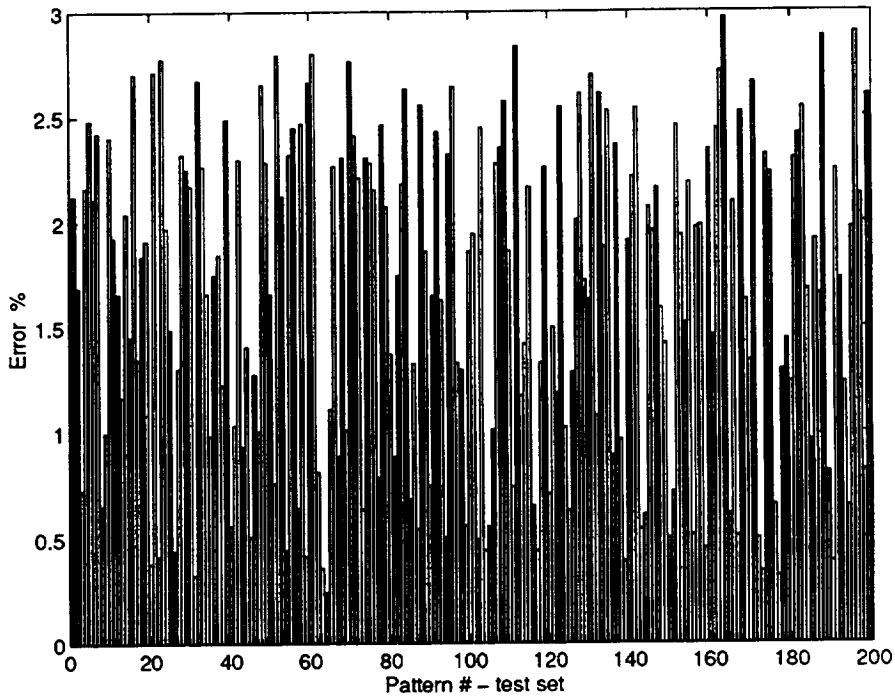

Figure 5. Results of NN using backpropagation with gradient descent

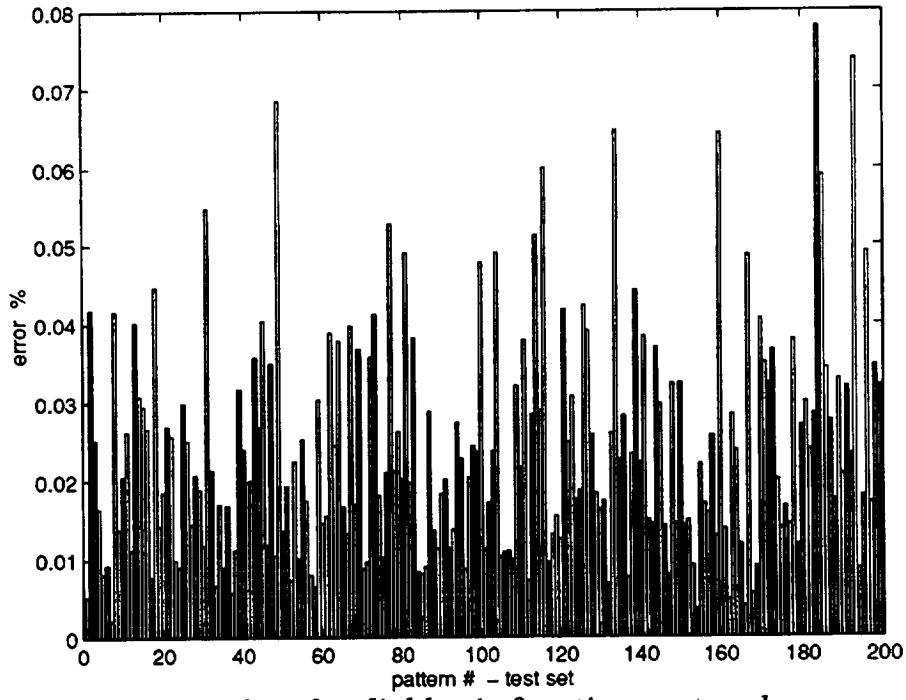

Figure 6. Results of radial basis functions network

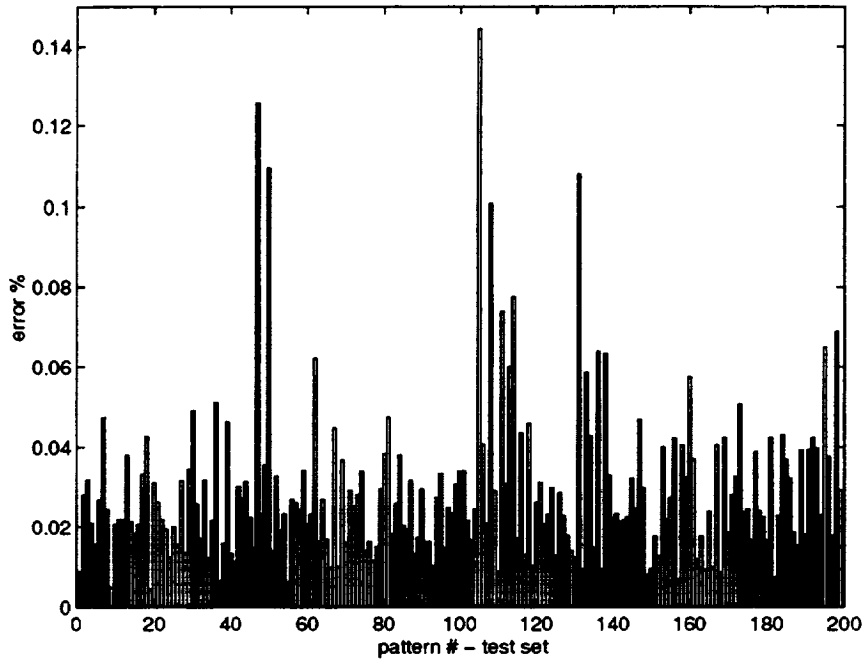

Figure 7. Results of Sugeno fuzzy system using clustering

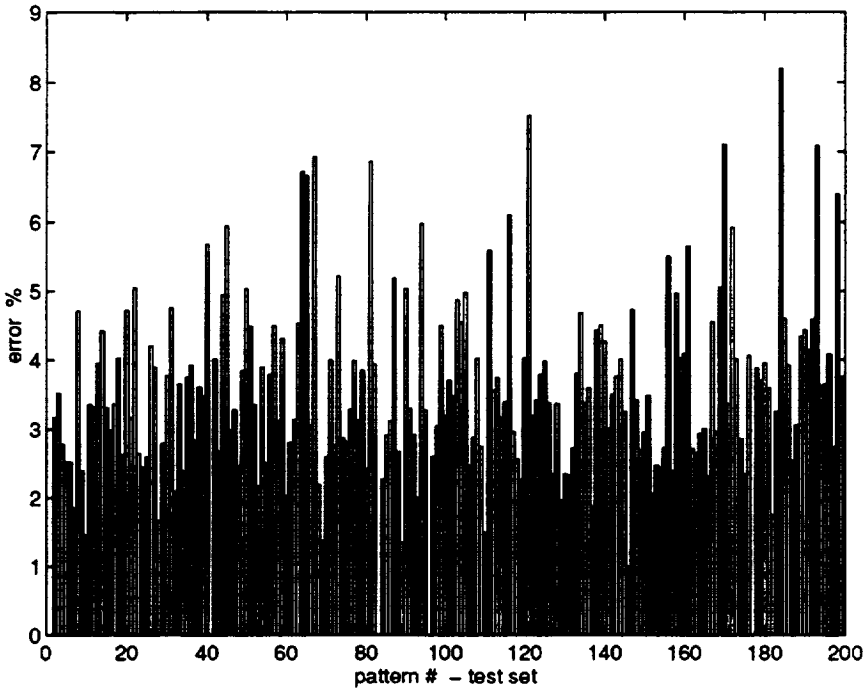

Figure 8. Results of Mamdani model

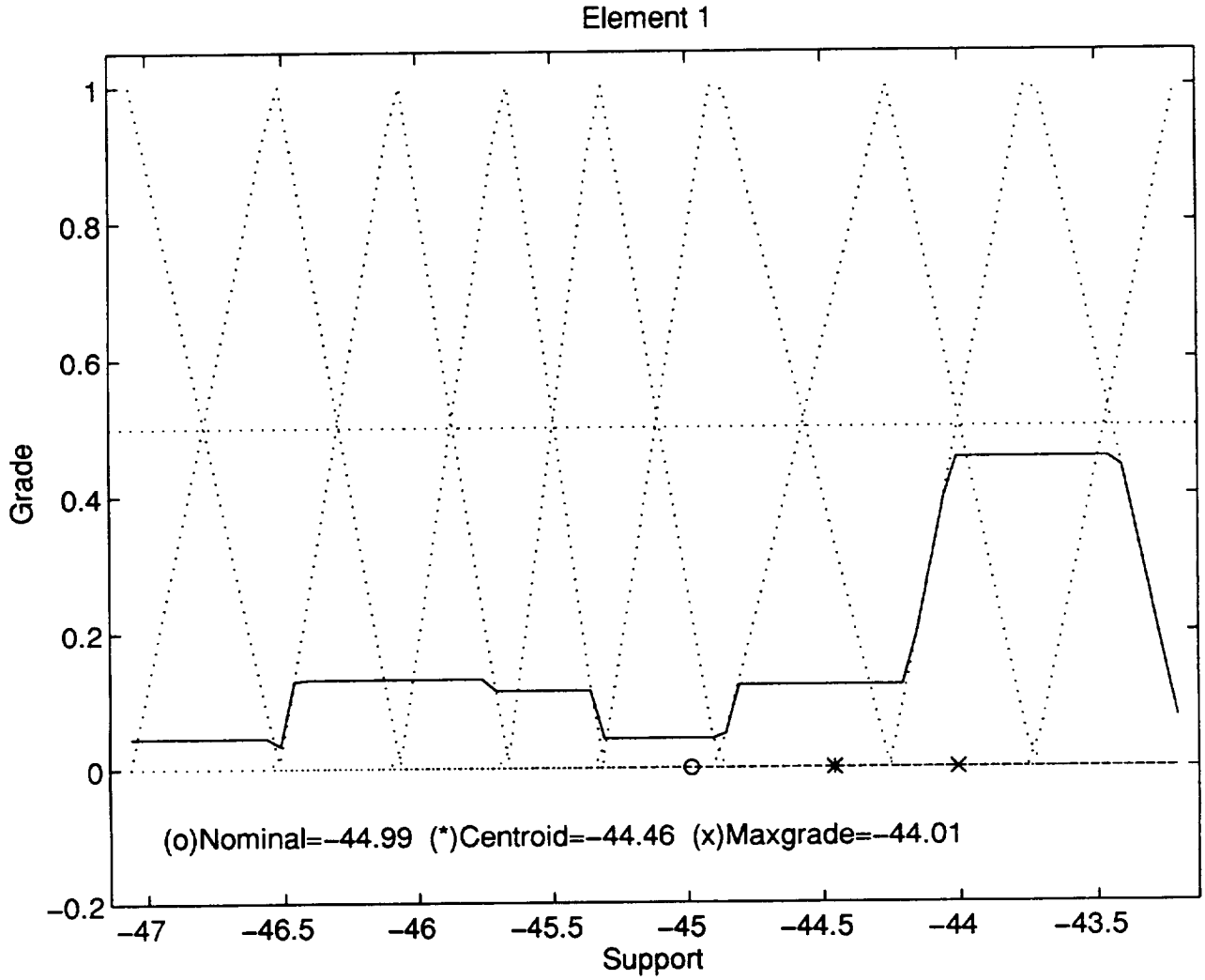

Figure 9. Inference mechanism for Mamdani model

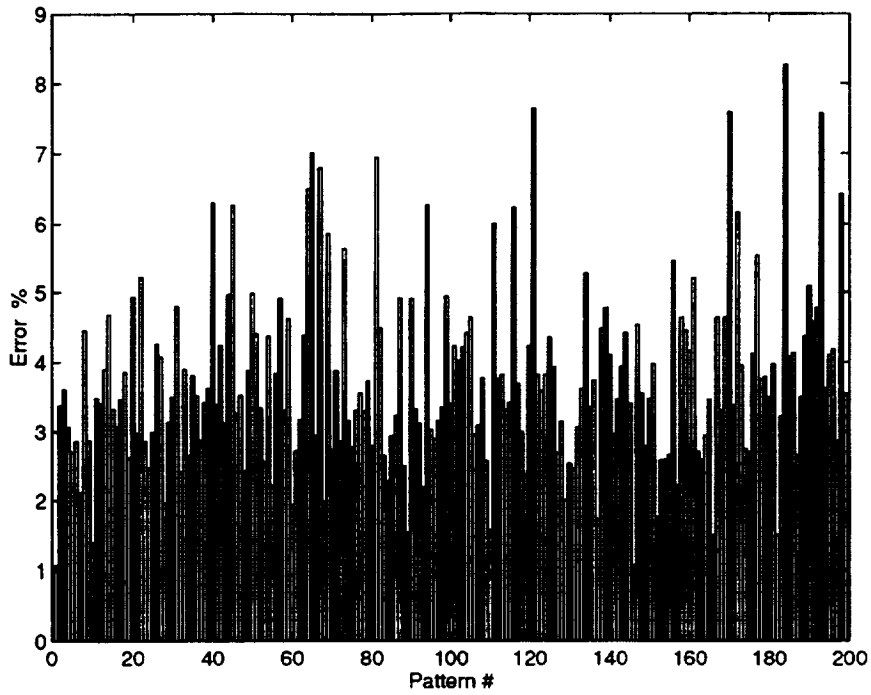

Figure 10. Results of Mamdani model using clustering

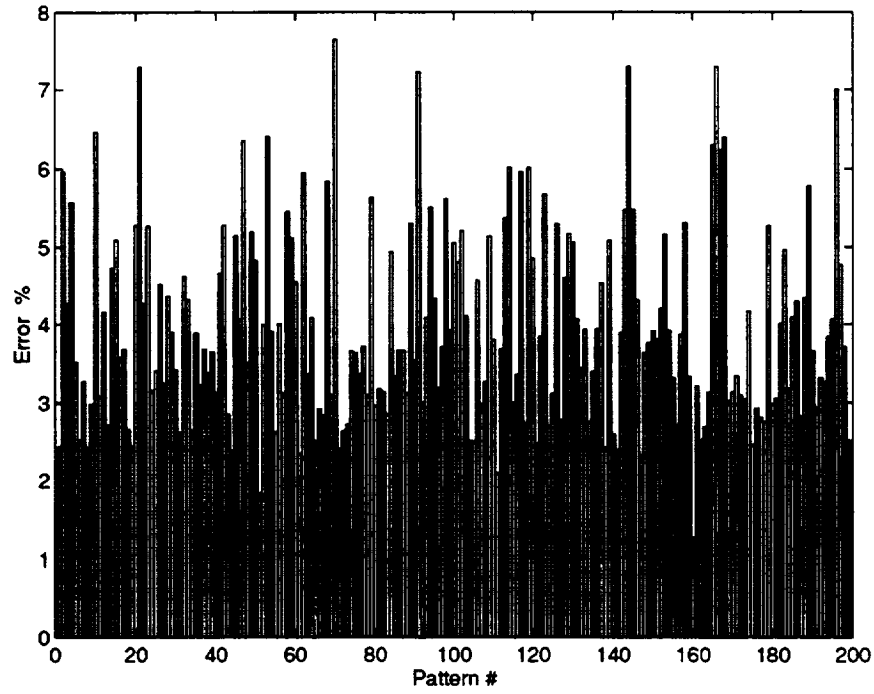

Figure 11. Results of fuzzy system using LS and clustering

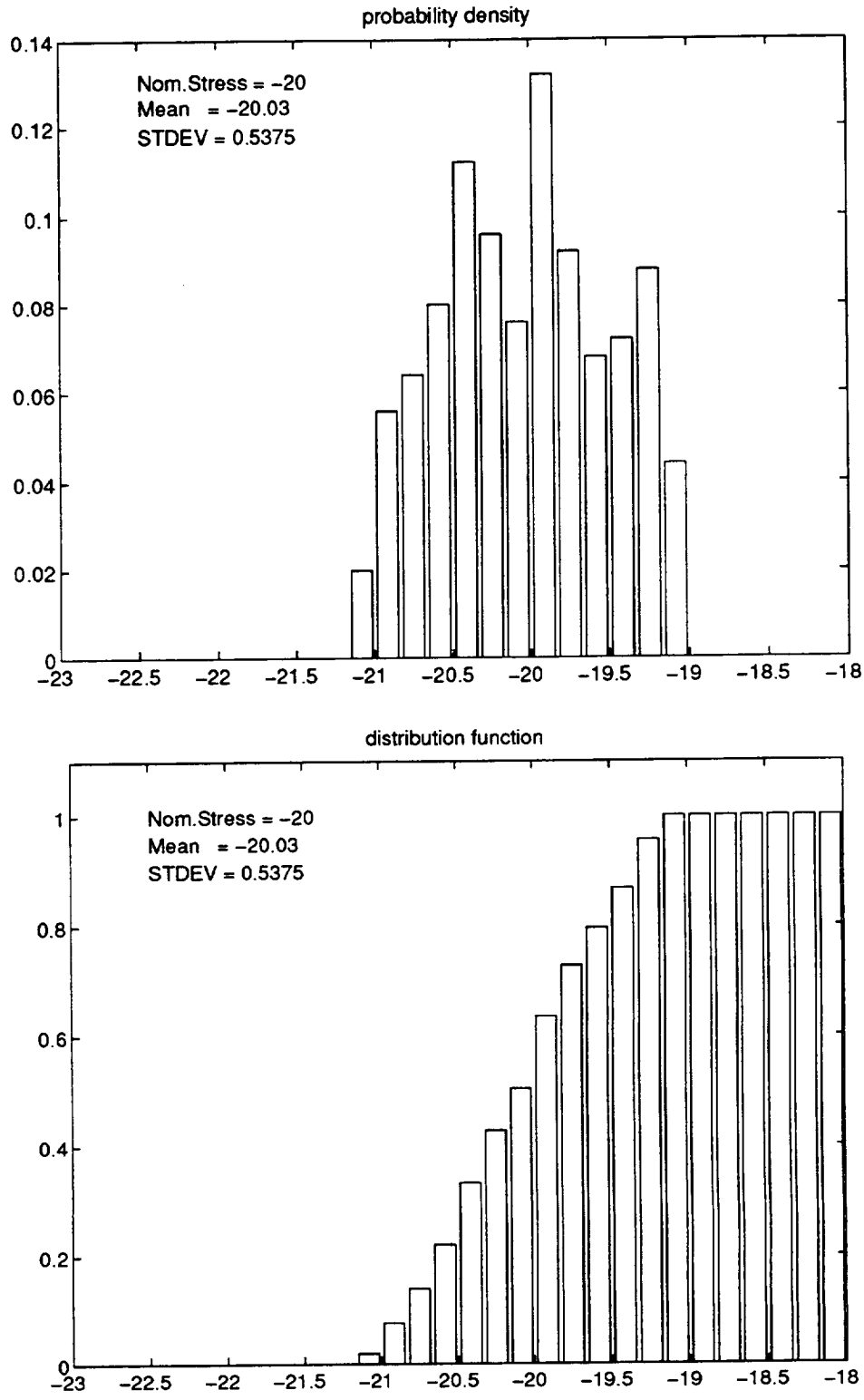

Figure 12. Probability density and distribution functions

Method	Ten-bar-truss
Linear feedforward network	**0.033** (10 neurons)
Gradient descent	**1.83** (10 neurons)
Levenberg-Marquardt	**0.1297** (10 : 10 neurons)
Radial basis function	**0.027** (30 : 10 neurons)
Sugeno with equally shaped fuzzy inputs	would need over 1000 rules
Sugeno using clustering	**0.0286** (9 rules)
Mamdani	**3.5847** (250 rules)
Mamdani using clustering	**3.707** (9 rules)
Fuzzy system using LS and clustering	**2.95** (9 rules)

Table 1. Average percentage error

Element	Original cross-sectional areas		RBF network results	
	minimum	maximum	minimum	maximum
1	4.2847	4.7287	4.2814	4.7262
2	2.0545	2.2675	2.0505	2.2674
3	0.095	0.105	0.0939	0.106
4	0.095	0.105	0.0934	0.1053
5	4.1674	4.5952	4.1662	4.5909
6	0.095	0.105	0.0961	0.1027
7	2.9011	3.2042	2.8978	3.2003
8	3.0737	3.3936	3.0708	3.3898
9	0.0952	0.1049	0.0941	0.1069
10	2.9034	3.2037	2.9002	3.2016

Table 2. Original cross-sectional area ranges and the ranges found by the RBF network

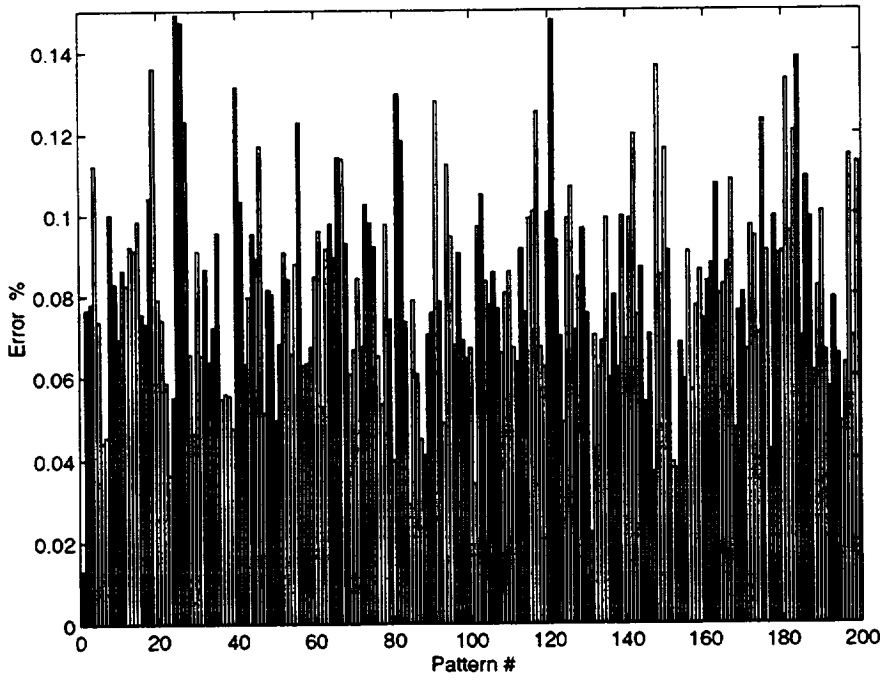

Figure 13. Errors for reverse engineering method

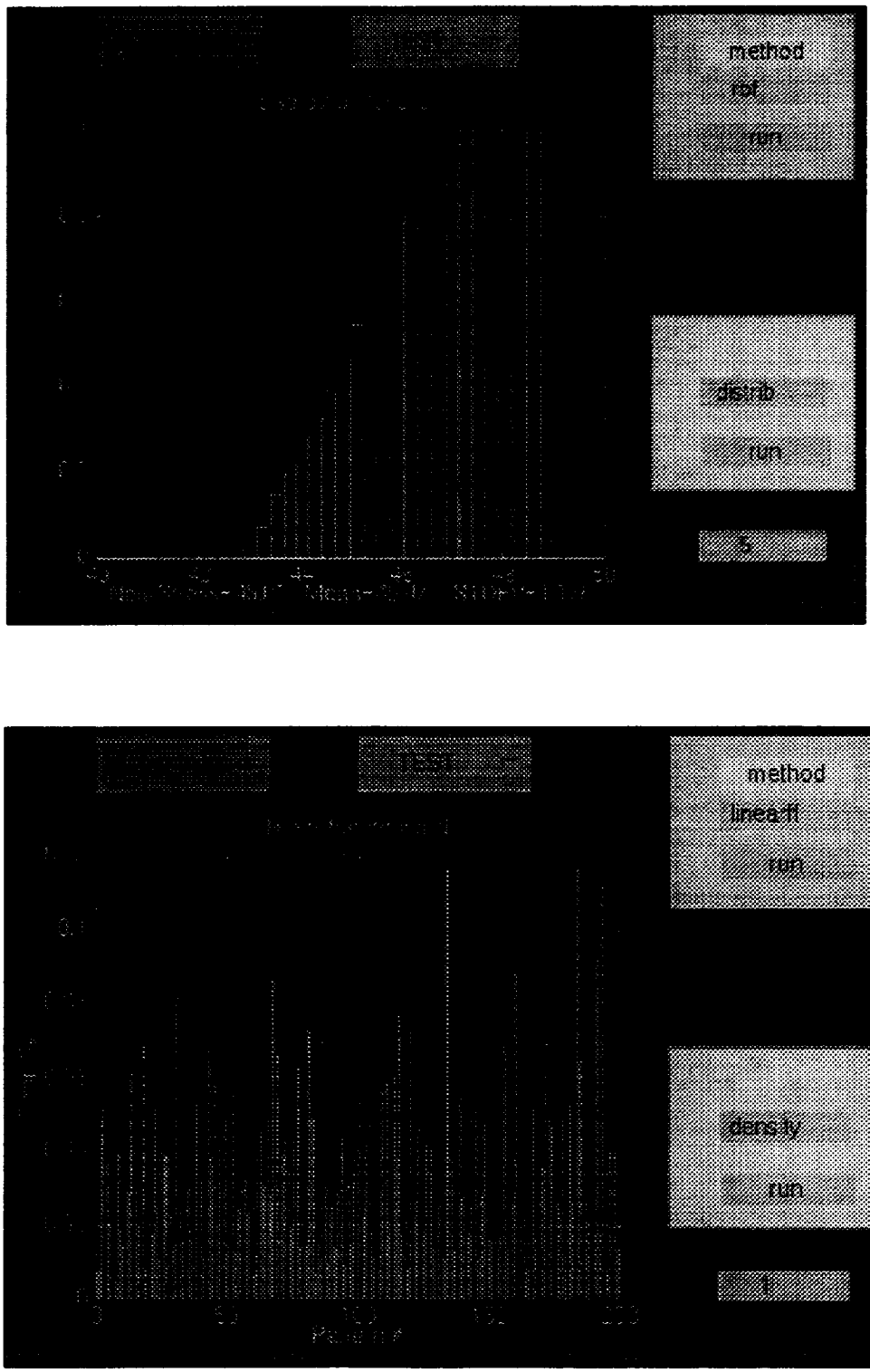

Figure 14. Graphical user interface

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704-0188

1. AGENCY USE ONLY (*Leave blank*)	2. REPORT DATE	3. REPORT TYPE AND DATES COVERED
	July 1996	Technical Memorandum

4. TITLE AND SUBTITLE

A Comparison of Neural Networks and Fuzzy Logic Methods for Process Modeling

5. FUNDING NUMBERS

WU-505-90-00

6. AUTHOR(S)

Krzysztof J. Cios, Dorel M. Sala, and Laszlo Berke

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)

National Aeronautics and Space Administration
Lewis Research Center
Cleveland, Ohio 44135-3191

8. PERFORMING ORGANIZATION REPORT NUMBER

E-10280

9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)

National Aeronautics and Space Administration
Washington, D.C. 20546-0001

10. SPONSORING/MONITORING AGENCY REPORT NUMBER

NASA TM-107236

11. SUPPLEMENTARY NOTES

Krzysztof J. Cios and Dorel M. Sala, University of Toledo, Toledo, Ohio 43606-3328; Laszlo Berke, NASA Lewis Research Center. Responsible person, Laszlo Berke, organization code 5200, (216) 433-5648.

12a. DISTRIBUTION/AVAILABILITY STATEMENT

Unclassified - Unlimited
Subject Category 64

This publication is available from the NASA Center for AeroSpace Information, (301) 621-0390.

12b. DISTRIBUTION CODE

13. ABSTRACT (*Maximum 200 words*)

The goal of this work was to analyze the potential of neural networks and fuzzy logic methods to develop approximate response surfaces as process modeling, that is for mapping of input into output. Structural response was chosen as an example. Each of the many methods surveyed are explained and the results are presented. Future research directions are also discussed.

14. SUBJECT TERMS

Structures; Neural nets; Fuzzy logic

15. NUMBER OF PAGES

28

16. PRICE CODE

A03

17. SECURITY CLASSIFICATION OF REPORT	18. SECURITY CLASSIFICATION OF THIS PAGE	19. SECURITY CLASSIFICATION OF ABSTRACT	20. LIMITATION OF ABSTRACT
Unclassified	Unclassified	Unclassified	

National Aeronautics and
Space Administration

Lewis Research Center
21000 Brookpark Rd.
Cleveland, OH 44135-3191