ICAS-96-5.6.2

INVESTIGATION INTO THE APPLICABILITY OF NEURAL NETWORK TO LOAD IDENTIFICATION

X.Cao* & Y.Mitsui**
**Research Associate*, ** *Professor, Dept. of Civil Eng., College of Eng., Shinshu Univ., Nagano 380 Japan*
Y.Sugiyama
Professor, College of Eng., Osaka Prefectural Univ., Osaka 593 Japan
J.Tang & B.Song
Senior Research Engineer, China Flight Test Establishment, Xi'an 710089 China

The intended aim of the study is to develop a neural network approach to the identification of loads acting on aircraft wings. The fundamental principle of the idea is focused on attempting to use neural network to represent or replace the load-strain relationship in structural analysis, although exists which in aircraft wing structures, it is difficult to formulate, then identify loads that are difficult to directly measure according to strains that are easily acceptable.

As the first step of the study, the paper describes the application of neural network to the identification of the loads distributed upon cantilevered beam model. The distributed load is approximated by a system of a finite number of concentrated loads. The paper demonstrates the applicability of neural network to the load identification for beam models.

Introduction

The accurate and reliable data on aircraft wing loads is highly necessary not only to both the design and development of an aircraft but also to establishment of Strength and Rigidity Specification for Aircraft Structures. But, aircraft wing loads are difficult to determine accurately and reliably by means of either wind-tunnel tests or theoretical analysis because both wing structure and its loading condition in flight are extremely complicated. Therefore, it is desirable to obtain wing loads through flight tests. However, being different from those physical parameters, such as flight height or velocity, flight loads can not be directly observed and measured in flight. Accordingly, an idea to identify flight loads acting on aircraft wing on the basis of the strain responses caused by flight loads came to us. Furthermore, the load-strain relationship, which stands for dynamical machnics characteristics existing in wing structures, is then represented using neural network.

We attempt to achieve our purpose by the approach described below and shown in Fig.1.

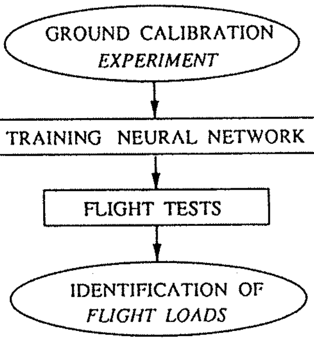

Fig.1 Approach for identifying flight loads utilizing neural network

For this to be realized, the neural network must be trained to provide an appropriate map of the input and output quantities.

1) Ground Calibration Experiment:

To use actual aircraft wing as test object before flight tests and apply calibration sample loads, which are selected based on theoretical analysis and design load envelope over entire region in which aerodynamic center of distributed loads varies. Strain responses are then measured in order to gain data that will next be used as learning data with applied loads for neural network.

2) Training Neural Network :

Using the strains as input signals and loads as teaching signals (desired output values), neural network's learning is performed till that error between actual outputs and desired outputs achieved an acceptable level. Hence, the configuration of neural network for our purpose to identify loads acting on given aircraft wing is decided.

3) Flight Test :

To measure the strain responses in main parts of wing, which are caused by wing external loads through flight tests to obtain strain data.

4) Identification of Flight Loads :

Input the measured strain data of the wing to the trained neural network and identify flight loads on the basis of the outputs of the neural network.

Synthesizing the above discussion, it can be noted that the approach seeks to replace the complex structural analysis module by a neural network model. Besides, for discussed problem, this proposition takes the advantage of it being possible to perform ground test using objective aircraft to obtain calibration data and avoiding complicated or even impossible theoretical analysis. This idea is instructive in case that structures are extremely complicated and machnical properties or machnical relationship between their physical parameters are difficult to formulate.

For this to be realized, the neural network must be trained on the basis of the data from theoretical or experimental analysis to provide an appropriate map of input and output quantities.

In order to accomplish this final aim, as the first step of the research, we investigated suitability of neural network to the load identification for a cantilevered beam model. Distributed loads are approximated by a system of a finite number of concentrated loads. The paper demonstrates the applicability of neural network to the load identification for beam model.

Computational Principle of NN

The most important characteristic of neural network can be described as the ability to efficiently handle the variables that are highly discrete and parallel.

The following sections include a brief overview of neural network configuration and its learning rule.

1) Neural Network Configuration

Multilayer neural network with input and output layer interrupted by hidden layers of neurons shown in Fig.2, is used in the study. The basic computational unit of such network is similar to neuron in brain, which highly mutually connect to compose a network.

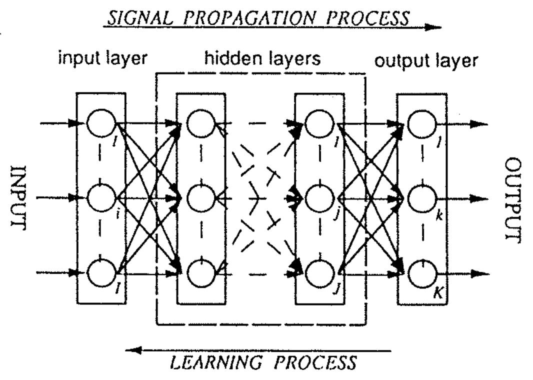

Fig.2 Miltilayer neural network

For multilayer neural network, it can afford a broad foundation, on which the number of layer and neuron in each layers can be optionally altered according to given problem. The neuron numbers in input layer and output layer are decided according to investigated aim. The numbers of hidden layers and neuron in each hidden layers are selected after synthetically considering computational speed and accuracy.

For this type of neural network, there is fully-connected network with each input influencing all output elements.

Each neuron is a fundamental computational element. Fig.3 shows two typical neurons respectively representing the neuron in input layer and output layer. For input layer, its neurons output the incomed signal x_i directly.

$$y_i = x_i$$

where, subscribe i denotes ith neuron in input layer.

For hidden layers and output layer, each neuron forms a weighted sum U_j of n inputs from previous layer, bias being added. Then the sum becomes input signal of the

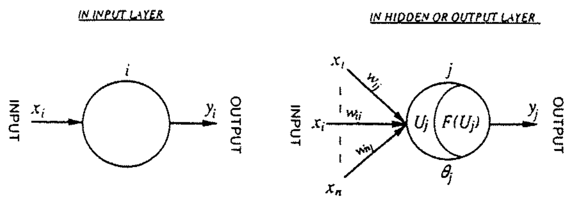

Fig.3 Two typical neurons

processing unit.

$$U_j = \sum_{i=1}^{n} w_{ij} x_i + \theta_j \qquad i = 1, 2, ..., n \qquad (1)$$

and is processed by an activation function F, also called as response function, to obtain its output y_j as follows :

$$y_j = F(U_j) \qquad (2)$$

The w_{ij} coefficients are termed as the weighted coefficients and the θ_j cofficient is termed as bias. The subscript i denotes that the w_{ij} is a weighted coefficient connecting ith neuron in previous layer and jth neuron in Jth layer (disscussed layer). Hidden layers don't interact with the external environment and simply output or input information to neuron within the system. The output of an individual neuron may then become input to other neurons or may simply be one of the network outputs. The former belongs to hidden layer neurons and the latter belongs to output layer neuron.

In this research, a sigmoid function given by expression

$$F(U_j) = \frac{1}{1 + EXP(-U_j/T)} \qquad (3)$$

is adopted as activation function for the neurons.

In theory of probability, the sigmoid function presents

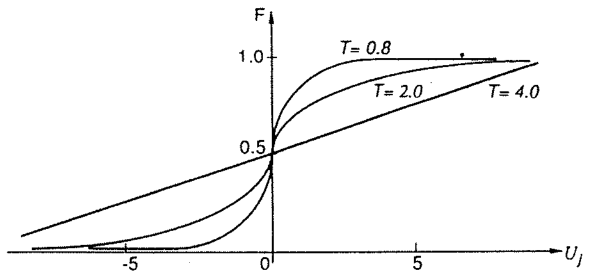

Fig.4 Sigmoid function $F(Uj)=1/[1+EXP(-Uj/T]$

fire probability of the neuron. T corresponds to the temperature in statistics, but, here T is a bias parameter used to modulate the unit output. The output feature of sigmoid function will vary with trend as shown in Fig.4 according T value.

The principle advantage of this function is its ability to handle both large and small input signals. The slope of this function is representative of the available gain. For both large positive and negative values of the input signal, the gain is vanishingly small; at intermediate value of the input signal, the gain is finite. Hence, an appropriate level of gain is obtained for a wide range of input signals. The output obtained from the activation function may be treated as an input to other neurons in the network.

2) Neural Network Learning Rule

The neural network must be "trained" to recognize known patterns and extrapolate from these known patterns when new information is presented.

After having trained neural network only on the basis of discrete and parallel data, the structural mechanics characteristics of researched object can be automatically generated, then recognized and embedded in neural network by the ability that neural network itself possesses. Hence, trained neural network can be instantly used as a simple function which yields the desired output vectors for given input vectors.

The determination of the proper weight coefficients and bias parameters are embodied in the network learning process, which itself is an error minimization problem.

For miltilayer neural network, its training is performed in the process continually renewing synapse weight, also called as weighted coefficient or connection coefficient, and bias to direct toward minimizing mean square error between desired output, also called as teaching signal, and actual output of neuron in output layer. The training of neural network is ended up and configuration of neural network is decided as soon as the error reaches acceptable level. Although many kinds of algorithm on training neural network have been proposed so for, the Error Back Propagation algorithm, which is proposed by Rumehard in 1986 and is abbreviated as EBP algorithm in this paper, is most broadly adopted. The EBP algorithm is an iterative gradient algorithm that centers around fastest descending error, from which many kinds of other learning algorithms are derived.

In our study, The EBP algorithm is used as supervised approach, after having added an inertia term, also called as momentum term. The momentum term smooths the varieties of weighted coefficients and bias coefficient, and sometimes speed up the convergence and increase stability of the learning algorithm. In addition, it can prevent convergence from local optimum and direct toward to global one.

In order to get a better insight into learning algorithm, it is best to facus attention on one processing element. We are going to describe the principle of the EBP algorithm using kth neuron in output layer as example (Fig.3). Its error can be expressed as follows:

$$e_k = [D_k - y_k] \tag{4}$$

where, D_k and y_k is desired and actual output of neuron k, respectively. Mean square error is defined as error function and is expressed by E_k

$$E_k = \frac{1}{2}e_k^{\,2}, \quad E_k = \frac{1}{2}[D_k - y_k]^2 \tag{5}$$

It should be noted that the error function can be given only for neurons in output layer, however for neurons in other layers, since such comparison is not possible, the error function can not be determinated directly. Therefore the learning of NN (neural network) must begin from output layer first. That is the cause why weighted coefficients and biases must be adjusted by starting at the output nodes and working back to input layer.

Let's observe the error function E_k,

$$\begin{aligned}
&E_k = \frac{1}{2}[D_k - y_k]^2 \\
&y_k = \frac{1}{1 + EXP(-U_k/T)} \qquad j = 1, 2, \ldots\ldots, J \\
&U_k = \sum_{j=1}^{J} w_{jk}x_j + \theta_k \\
&E_k = \frac{1}{2}[D_k - \frac{1}{1 + EXP[-(\sum_{j=1}^{J} w_{jk}x_j + \theta_k)/T]}]^2
\end{aligned} \tag{6}$$

Where, J is over all notes in the layer lower than the one where node k is.

We can consider E_k as a multi-dimension function of connection coefficients w_{jk} and bias θ_k. The minimum value of E_k, which, sometimes, may not be sure to be a desired global minimum instead of a local minimum, appears in the point where partially differentiating with respect to connection coefficients w_{jk} and bias θ_k, $\frac{\partial E_k}{\partial w_{jk}}$ and $\frac{\partial E_k}{\partial \theta_k}$ are all equal to zero.

$$\frac{\partial E_k}{\partial w_{jk}} = 0 \tag{7}$$

$$\frac{\partial E_k}{\partial \theta_k} = 0 \tag{8}$$

It is obvious that the learning algorithm of neural network is substantially a process to gradually adapt w_{jk} and θ_k according to the expression (7) and (8).

From eq.(7)

$$\frac{\partial E_k}{\partial w_{jk}} = \frac{\partial E_k}{\partial y_k}\cdot\frac{\partial y_k}{\partial U_k}\cdot\frac{\partial U_k}{\partial w_{jk}} = -[D_k - y_k]F'(U_k)x_j \tag{9}$$

From eq.(8)

$$\frac{\partial E_k}{\partial \theta_k} = \frac{\partial E_k}{\partial y_k}\cdot\frac{\partial y_k}{\partial U_k}\cdot\frac{\partial U_k}{\partial \theta_k} = -[D_k - y_k]F'(U_k) \tag{10}$$

$$F'(U_k) = \frac{\partial F}{\partial U_k} = y_k[1 - y_k]\frac{1}{T} \tag{11}$$

Substitute eq.(11) into eq.(9) and (10)

$$\frac{\partial E_k}{\partial w_{jk}} = -[D_k - y_k]y_k[1 - y_k]x_j\frac{1}{T} \tag{12}$$

$$\frac{\partial E_k}{\partial \theta_k} = -[D_k - y_k]y_k[1 - y_k]\frac{1}{T} \tag{13}$$

Define $[D_k - y_k]y_k[1 - y_k]\frac{1}{T}$ as error term of kth neuron in output layer and denote it using δ_k:

$$\delta_k = [D_k - y_k]y_k[1 - y_k]\frac{1}{T} \tag{14}$$

hence,

$$\frac{\partial E_k}{\partial w_{jk}} = -\delta_k x_j \tag{15}$$

$$\frac{\partial E_k}{\partial \theta_k} = -\delta_k \tag{16}$$

The coefficients η and γ are termed as training rates of weighted coefficients w_{jk} and bias θ_k respectively. Using a gradient descent technique to minimize the mean square difference between the desired and actual net work output, the change in value of the w_{jk} and θ_k are respectively given by the expressions (17) and (18),

$$\Delta w_{jk} = -\eta \frac{\partial E_k}{\partial w_{jk}} = \eta \delta_k x_j \quad (17)$$

$$\Delta \theta_k = -\gamma \frac{\partial E_k}{\partial \theta_k} = \gamma \delta_k \quad (18)$$

Throught a recursive algorithm starting at the output nodes and working back to the first hidden layer, the weighted coefficients and bias are adjusted respectively by the below expressions,

$$w_{jk}(t+1) = w_{jk}(t) + \eta \delta_k x_j \quad (19)$$

$$\theta_k(t+1) = \theta_k(t) + \gamma \delta_k \quad (20)$$

where, t denotes the learning cycle.

The learning algorithm expressed by eq.(19) and (20) is called as EBP (Error Back Propagation) algorithm, which is an iterative gradient algorithm designed to minimize the mean square error between actual and target output.

The training algorithm may be improved by adding an inertia term

$$\alpha \Delta w_{jk}(t-1) = \alpha [w_{jk}(t) - w_{jk}(t-1)] \quad (21)$$

$$\beta \Delta \theta_k(t-1) = \beta [\theta_k(t) - \theta_k(t-1)] \quad (22)$$

into expressions (19) and (20).

$$w_{jk}(t+1) = w_{jk}(t) + \eta \delta_k x_j + \alpha [w_{jk}(t) - w_{jk}(t-1)] \quad (23)$$

$$\theta_k(t+1) = \theta_k(t) + \gamma \delta_k + \beta [\theta_k(t) - \theta_k(t-1)] \quad (24)$$

The learning rule shown by eq.(23) and eq.(24) is called as Improved Error Back Pagatation, which is abbreviated as IEBP algorithm in this paper. α and β are termed as inertia coefficients respectively for the weighted and bias coefficient.

The varieties of weighted coefficients and bias coefficient can be smoothed by means of adding the inertia term. The convergence is sometimes speeded up. Besides, the stability of algorithm is increased. For optimum problem it can prevent convergence to the local optimum.This is a gradient research technique in order to improve performance and reduce the occurrence of local minimum, also.

In expression (23) and (24) the term

$$\delta_k = [D_k - y_k] y_k [1 - y_k] \frac{1}{T} \quad (25)$$

is contained and called as error term.

We have described training principle to the kth neuron in output layer. The principle mentioned above can be used to hidden-layers after being somewhat corrected.

The error of the neuron in hidden-layer $J(\delta_j)$ can be determinated making use the error of output layer if J layer is a layer lower than output layer, or next layer if J layer is not a being adjusted layer but connecting J layer and K layer.

$$\delta_j = h_j [1 - h_j] \sum_{k=1}^{K} \delta_k w_{jk} \frac{1}{T} \quad (26)$$

where, h_j denotes the output of J neuron.

Comparing eq.(26) with (25), it is clear that the term $\sum_{k=1}^{K} \delta_k w_{jk}$ is corresponding to the error $[D_k - y_k]$ of the neuron in output-layer and can be termed as the error of the hidden-layers. Since the teaching signals (desired outputs) are not available for hidden layer neurons, the error in output layer must be computed first, after correcting weighted coefficients connecting output layer and lower layer and bias of output layer, the error in hidden layer can be obtained by eq.(26). It is reason why the recursive algorithm of EBP or IEBP must start at output nodes first and work back to first hidden layer.

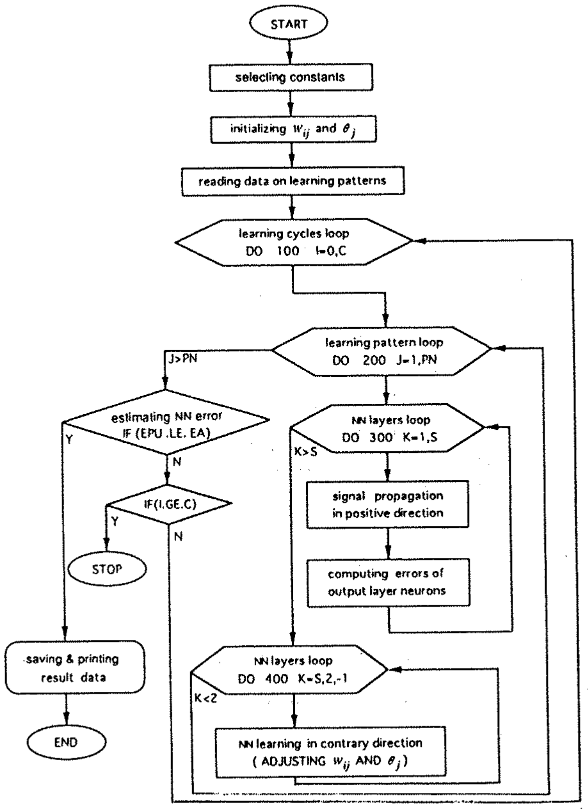
START
selecting constants
initializing w_{ij} and θ_j
reading data on learning patterns
learning cycles loop DO 100 I=0,C
learning pattern loop DO 200 J=1,PN
J>PN
estimating NN error IF (EPU .LE. EA)
Y
N
IF(I.GE.C)
Y
N
STOP
saving & printing result data
END
NN layers loop DO 300 K=1,S
K>S
signal propagation in positive direction
computing errors of output layer neurons
NN layers loop DO 400 K=S,2,-1
K<2
NN learning in contrary direction (ADJUSTING w_{ij} AND θ_j)

Fig.5 Flow-chart of NN learning

For EBP or IEBP algorithm, adjustment of weighted coefficients and bias can be roughtly classified into two ways.

1) Gradual Correction
The weighted coefficients and bias are corrected every time when each new pattern is input.

2) Collective Correction
Suming up the correction quantities of weighted coefficients and bias respectively, which are determined based on the error for each learning pattern, weighted coefficients and bias are collectively corrected.

In this study, the way of gradually correction is employed. The flow-chart of neural network learning is shown in Fig.5.

Identified Results and Discussion

As shown in Fig.6, an aircraft wing is simplified to a cantilevered beam model, nonuniform distributed loads acting on wing are approximated by a sysem of a finite number of concentrated loads that are applied vertically to axis direction.

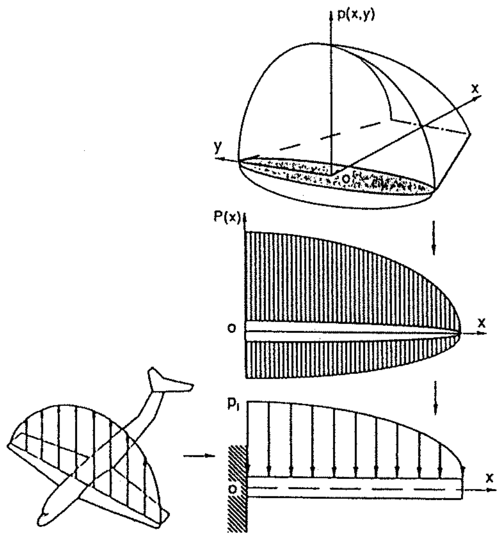

Fig.6 Modelling wing loads

A 3-layer neural network is adopted to this work with 11 reurons in input layer, one hidden layer and output layer with 11 output neurons.

As a results of learning neural network, an architecture of neural network, which has 4 neurons in hidden-layer, sigmoid temperature T of the neurons in the hidden-layer and output-layer being equal to 1.1 deg., momentum coefficients α and β being equal to -0.20, learning rates η and γ being equal to 1.7, is formed and the learning of NN is accomplished. The relative error of each neurons in output layer is reduced to 5% in 4151 cycles training.

Identifying of loads is smoothly and idealy accomplished and 6 sets of outputs are obtained as soon as 6 sets of strain values are input, in order, into the trained neural network.

Summary

Having trained neural network using several sets of training data from theoretical computation, the neural network understood and realized the load-strain relationship of researched object, which represents the static structural machnics characteristics for the considered cantilevered beam model. Then, when new strains from checking samples are inputted, the trained neural network immediately output loads, which are desired value and enough satisfy demands for accuracy by comparing these outputs with theoretical values. Hence, the paper demonstrates the applicability of neural network to load identification for beam model.

References

[1] P.Hajela and L.Berke, Neurobiolo gical computational models in structural analysis and design, *Computers & Structures,* Vol.41, No.4, pp.657-667(1991).

[2] D.E.Rumelhart, G.E.Hinton and R,J.Williams, Learning representations by back-propagating errors, *Nature*, 323-9, 533-536(1986).

[3] H.Asou, Treatment of information with neural network, *Industry Books Press*, Tokyo(1992).