J. King Saud Univ., Vol. 9, *Eng. Sci.* (2), pp 169-189 (A.H. 1417/1997)

Prediction of Stress-strain Relationships for Reinforced Concrete Sections by Implementing Neural Network Techniques

Mansour Nasser Jadid

Department of Building Science and Technology, King Faisal University
P. O. Box 30973, AlKhobar 31952, Saudi Arabia

(Received 27 May 1996; accepted for publication 9 October 1996)

Abstract. The application of neural networks for predicting the stress-strain relationships of reinforced concrete sections is presented. Computation algorithms in the form of numerical analysis were performed on reinforced concrete sections to simulate existing experimental data. A systematic approach is provided by implementing neural networks in the form of prediction by backpropagation algorithms. The efficiency of neural network techniques is demonstrated by means of reconstructing previous experimental work and evaluating several parameters based on neural networks which are in agreement with experimental results. The procedure establishes valid mathematical relationships without relying on a particular algorithm and depends entirely on the manipulation of numerical data.

Introduction

The brain in the human biological neural network is composed of dense nerve cells which are highly interconnected and estimated to total 100 billion neurons of different types which are constantly sending and receiving messages. These nerve cells are fundamental elements to the central nervous system and determine any action which is taken. It is estimated that each nerve cell is connected to 10,000 nerve cells resulting in an immensely complicated network. They are inspired by human biological neural networks whereby they capture the brain function manipulation to approach a specific problem by applying certain rules to achieve reasonable results. The processing element in the artificial network is analogous to the nerve cell in the human brain.

The era of neural computation began in 1943 when McCulloch and Pitts [1] proposed a general mathematical model which represented a neuron-like threshold by deriving a theorem related to a model of the neuron system which was adopted by many Artificial Intelligence (AI) researchers involved in modern neural computing. A learning rule theory based on updating the synaptic strengths between two neurons when both of them are active was postulated by Hebb [2] in 1949. Combining the ideas of Hebb, McCulloch and Pitts, the perceptron model was introduced by Rosenblatt [3] in 1958 to calculate logical functions by modifying the connections between the synapses. In 1969 Minsky and Papert [4] proved the limitations of the perceptron as a linear classifier which was a setback for the parallel distributed community and as a result most researchers turned to the symbolic logic approach. A renewed interest in this area emerged in 1982 when Hopfield [5] introduced his network that persuaded many scientists and mathematicians to re-evaluate neural networks. Rumelhart, Hinton, and Williams [6] laid the mathematical foundation of the backpropagation network by presenting a clear and concise learning algorithm for the multilayer artificial neural network which opened a new era in modern neurocomputing.

Learning Algorithms and Transfer Functions

Neural networks or artificial neural nets are based on modern research in the field of neurophysiology, a study of the human nervous system and its biological brain neurons. Learning is a fundamental property for neural networks to generate their own rules where adaptation is achieved through the learning rule that adjusts weights of the intermediate connections in response to the input data. This adaptability is the ability of the network to adjust its internal algorithm to satisfy the desired response. Neural networks are trained by several training techniques which have developed from a common root and share many characteristics. The training algorithm can be classified into three main groups: supervised, unsupervised and reinforcement learning.

Neural networks commonly apply a transfer function 'threshold' which specifies how the processing elements scale their response to the incoming signals, and then produce the activation. The processing element will output a signal if the activation is strong enough, that is, it passes certain threshold criteria. The output is usually of the form (0,1 or -1,+1). The primary transfer functions used are non-linear functions (Sigmoid, Fig. 1(a) or hyperbolic tangent function, Fig. 1(b)). A wide range of complicated functions are available and can be used for various special purposes.

General Overview of the Backpropagation Algorithm

Neural networks are essentially a computing system consisting of a large number of interconnected processing elements arranged in several layers to provide output signals as result of a series of input signals. Neural networks are a valuable information processing technology that is more efficient and robust than conventional programming. It is a learning mechanism

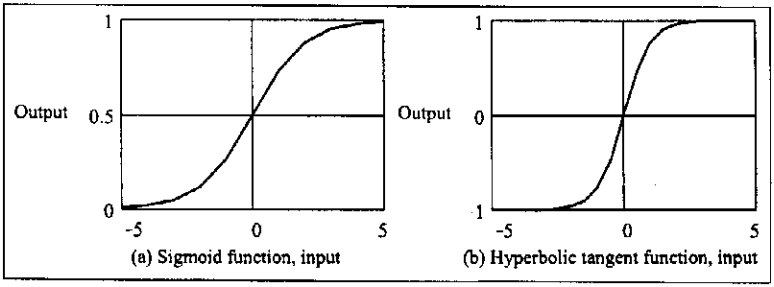

Fig. 1. Non-linear functions.

which involves a process of automatic weight adjustment between the layers of the individual processing elements to ensure that a stability approximation of the outcome results.

Figure 2 shows a simplified processing element of the backpropagation network with its summation and activation functions. These play a major role in backpropagation algorithm training.

A backpropagation network always has an input layer with one or more internal layers and one output layer. The processing elements in the input layer represent a set of input data, while the processing elements in the output layer represent the desired output. The internal processing elements represent an approximation to complex non-linear mapping between the input and output layers. The backpropagation algorithm involves a forward propagation start when a set of input patterns is presented to the network and a backward

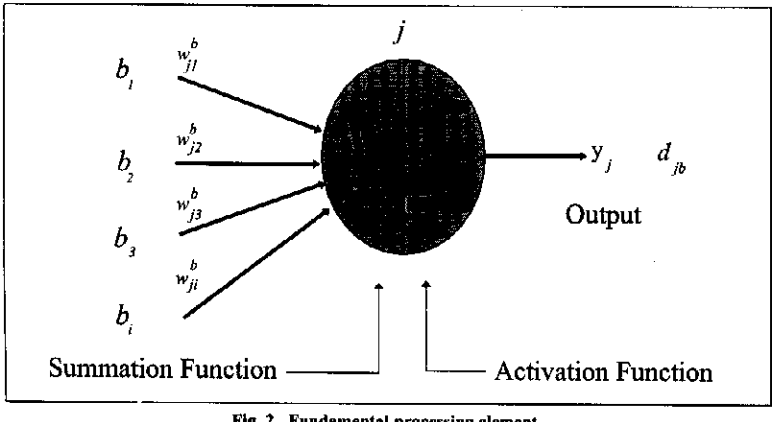

Fig. 2. Fundamental processing element.

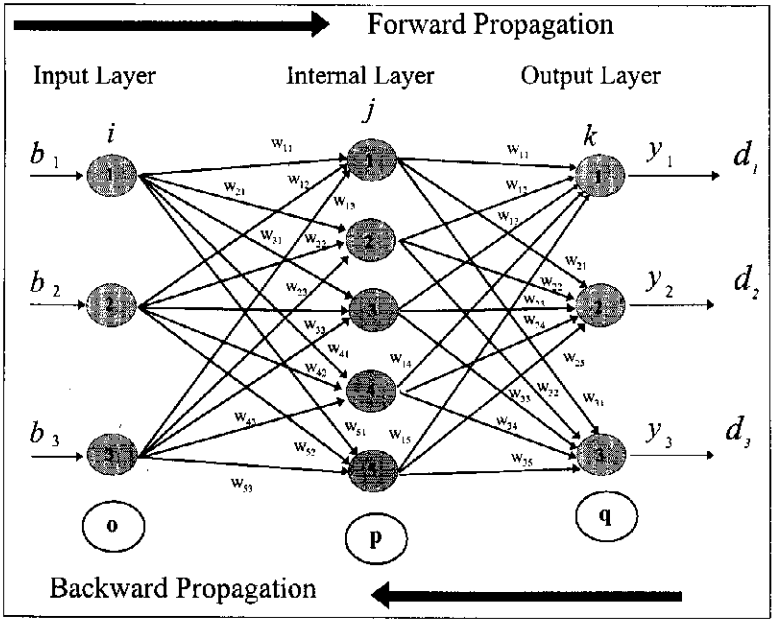

Fig. 3. Backpropagation network.

error propagation begins at the output layer when errors propagate through the internal layers toward the input layer. Figure 3 shows a typical backpropagation network with three layers, each of which is connected to the processing elements in the next layer.

Basic Mathematical Formulation and Weight Adjustments

The backpropagation network basic mathematical formulation requires each processing element to perform four main steps:

1. Input processing elements receive information from other or start with known input data.
2. Summation function which involves the activation of each processing element with its weight.
3. Transform the summation input activation data to an output activation data by threshold function.
4. At output layer, output processing elements are computed as a result of the previous process.

Initially, a set of input data b1, b2, ... bi, is applied through a set of associated weights $w^b_{j1}, w^b_{j2}, w^b_{j3}$ ... w^b_{ji} to produce an output processing element yj which becomes the input to the activation function which is represented mathematically as:

$$S_j = \sum_{i=1}^{n} b_i W^b_{ji} \tag{1}$$

$$y_j = f(S_j) \tag{2}$$

where:

b_i = input data for the backpropagation network.

y_j = output data of the backpropagation network.

d_{jb} = desired signals of the backpropagation network.

w^b_{ji} = backpropagation weight adjustment between the input and output signals.

The general activation functions are non-linear and for sigmoid activation function, equation (2), becomes

$$y_j = f(S_j) = \frac{1}{1+e^{-s_i}} \tag{3}$$

And for hyperbolic activation tangent function, equation (2) becomes

$$y_j = f(S_j) = \frac{e^{sj} - e^{-sj}}{e^{sj} + e^{-sj}} \tag{4}$$

The process of weight adjustment started at the output layer when the network produces a single real number for each processing element, (y_{1k}, y_{2k}, ..., y_{qk}). These real numbers are then compared to the desired output (d_{1k}, d_{2k}, ...,d_{qk}) and specified by the input training set to obtain the error signal. The network performance for one processing element in the output layer is known as the error signal, e_{qk}, and is computed as

$$e_{qk} = (d_{qk} - y_{qk}) \tag{5}$$

where:

q = backpropagation processing element in the output layer.

k = refers to the output layer in backpropagation.

e_{qk} = backpropagation error signal for one processing element in the output layer.

d_{qk} = backpropagation desired output.

y_{qk} = backpropagation actual output for one processing element.

From equation (5), the error signal, e_{qk}, is then multiplied by an activation function derivative to obtain the error value, δ_{qk}, which can be computed from

$$\delta_{qk} = f'_k(S_{qk})e_{qk} \tag{6}$$

From equation (6), the error value, δ_{qk}, is then multiplied by y_{pj}, the output of one processing element in the internal layer which provides the connection weight adjustment known as the generalised delta rule, $\Delta w^b_{qp,k}$, which was developed by Rumelhart, Hinton, and Williams [6]. This weight adjustment is computed as

$$\Delta W^b_{qp,k} = \eta f'_k(S_{qk})e_{ek}y_{pj} \tag{7}$$

where:

$\Delta w^b_{qp,k}$ = backpropagation adjusted weight between the q^{th} processing element in the output layer and the p^{th} processing element in the internal layer.

y_{pj} = backpropagation output for one processing element in the internal layer.

The weight adjustment for the internal layer requires a different procedure because of the absence of desired outputs in the internal layer. For the internal layer, the error value, δ_{pj}, is generated without the desired outputs. Calculate each processing element error value in the output layer, δ_{qk}, from equation (6). These are used to adjust weights going into the output layer where they propagate to the internal layer to generate δ_{pj}, for the internal layer which is computed as

$$\delta_{pj} = f'_k(S_{pj})\sum_{j=1}^{n} W^b_{kj}\,\delta_{qk} \tag{8}$$

The adjustment of the internal layer is computed as

$$\Delta W^b_{po,j} = \eta f'_k(S_{pj})(\sum_{j=1}^{n} W^b_{kj}\delta_{qk})Y_{oi} \tag{9}$$

where:

$\Delta w^b_{po,j}$ = backpropagation adjusted weight between the p^{th} processing element in internal layer and the o^{th} element in the input layer.

y_{oi} = backpropagation output for one processing element in the input layer.

i = refers to input layer in backpropagation.

j = refers to internal layer in backpropagation.

o = backpropagation processing element in input layer.

Neural Network Implementations in Civil Engineering

A considerable interest in the application of neural networks to civil engineering has emerged in recent years, due to the development of non-linear mapping which shows that the strength of neural network lies in the non-linearity function that provides an internal representation of the input data. The technology of neural networks shows tremendous promise in an area where conventional programming and logic symbolic (Manipulation of symbols) failed to provide adequate solutions. Neural networks can be of benefit in the fields of civil engineering and structural engineering, in particular, due to their valuable information processing technology which can be used to solve problems which are resistant to other computational technology.

An adaptive finite element mesh generation implementation using neural networks in structural analysis was carried by Jadid and Fairbairn [7]. The application of the neural network technique was focused on the adaptive re-meshing of an idealized square shape and individual triangle by using triangular elements. A supervised training technique by backpropagation learning algorithm was implemented which deals with the problem of re-meshing structural elements in a structural analysis. The main objective of the author's study was to show how neural networks can be employed to re-mesh structural elements without using numerically intensive computations. The main criteria was the selection of feasible and appropriate domain for generating training and test data. It also demonstrated that the capability of neural networks lies in its presentation of n-dimensional space which keeps track of each individual characteristic which does not make prior assumptions about mathematical formulation or the algorithm involved, but bases its results on the analytical results that feed into it.

Jadid and Fairbairn [8] carried out evaluations for the shear stress carried by the stirrups of a beam-column joint to adjust error by neural networks. It demonstrated the effectiveness of backpropagation in addressing problems analytically that produce solutions for which problems have not been formulated explicitly.

System Tools Environment

NeuralWorks® Professional II/Plus [9] version 5 is a system tool for neural network implementation on a microcomputer for several architectures. It is a complete, comprehensive and sophisticated graphics software package with multi-paradigm prototyping for system development. It employs one of the most popular and powerful algorithms to perform parallel distributed processing. It can be used to design, build, train, test and deploy a neural network in many forms for approximating complex non-

linear mathematical relationships. The networks are displayed either as networks or Hinton diagrams. NeuralWorks® [9] supports multi-platform hardware that has a common user-friendly graphical environment interface which is organized into three main areas: a mix of floating down menus used to control the network; network working area interface; and "fly-out" tool boxes of several icons.

Primary Approach in Implementing Neural Network

The implementation of a neural network tool requires the setting up of training and test data for each individual task and the selection of correct parameters that the network requires to provide a reasonable and acceptable trained network. Figure 4 illustrates the methodology process of implementing a neural network, involving two main stages; **pre-processing** and **post-processing.**

Methodology Process

The methodology process of implementing neural network in predicting parameters involves:

Gathering of necessary data

The process involves collecting the required data in one place by generating a FORTRAN program or using a mathematical package such as Mathcad® [10] for each specific task. Then separate the data into two sets, one for training and the other for testing. The testing data is normally taken between 9% or 10% of the whole data such that the 9^{th} or 10^{th} element of each training set is reserved for the testing data which will provide the ***best*** picture representations that increase confidence in the performance of the trained network.

Approach for data scaling

Transform the input data to acceptable values to the network. The neural network accepts only values from 0 to 1 for the sigmoid function and -1 to 1 for the hyperbolic tangent function. The input and test sets are normalized by a specific tool within the neural networks under the *MinMax* table. The calculation involves the computation of low and high values of each training and test example data field in the selected data files and stores them in the *MinMax* table.

Selection of network requirements

A backpropagation network is a general purpose network that can be implemented for prediction, classification, system monitoring, filtering and solving other problems. The advantages of the backpropagation network are the use of non-linear regression techniques that attempt to minimize global error, its ability to provide compact distribution representations of complex data and its potential to manipulate multiple-dimensional functions. However,

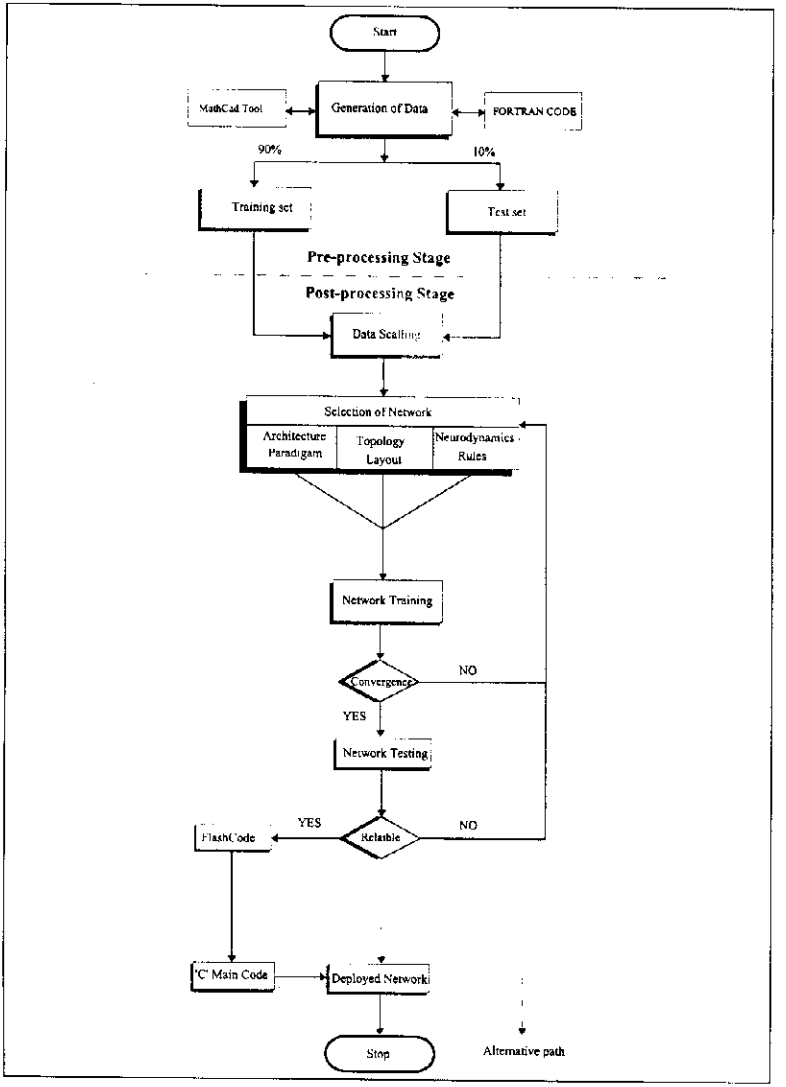

Fig. 4. Neural network strategies implementation.

there are also disadvantages for the backpropagation network due to its slow learning, its weakness in solving fundamentally different problems, it entrapments in local minimum and its difficulty to set good learning parameters. To overcome these shortcomings, variants of backpropagation networks are available that can resolve the inadequacies in backpropagation. These variant networks are Delta-Bar-Delta rule (DBD), Extended-Delta-Bar-Delta (EDBD), QuickProp and MaxProp and the Logicon Projection Network.

The selection of an appropriate network is based on the three following configurations: architecture paradigm, topology layout and neurodynamics rules. These three main aspects are essential in selecting a specific network paradigm that gives and dictates the characteristics of a given network.

Architecture paradigm

The architecture paradigm is essential in applying a neural network effectively since it requires skills and knowledge of the network paradigm to meet the specific problem. Tuning the network's performance is essential in establishing a reliable network which requires skill that can be learned only with practice. The backpropagation and its variant network architecture is selected for this work according to the availability of series pattern pairs, where each pair consists of an input pattern with desired output pattern. The learning technique offered by the backpropagation is supervised learning. This learning rule is classified as; Hebbian learning rule, Perceptron learning rule, Widrow-Hoff learning rule and Generalized delta learning rule which is an extension of the Widrow-Hoff learning rule and is widely implemented, especially in the backpropagation algorithm.

Topology layout

Network topology layout consists of the number of input and output layers, the processing elements (PEs) they contain, the number of internal layers and processing elements, their interconnectivity and the properties of geometrical configurations. For simple problems one internal layer is selected while more than one internal layer is selected for complex problems. Within each internal layer, the fewer the number of processing elements (PEs) the better the network performs. As a general rule, the amount of input data that can be used as an upper bound for the number of PEs in the internal layer is as follows [9]

$$I_{PE} = \frac{Row_{num}}{Const*(out_{PE} + input_{PE})} \qquad (10)$$

where:

I_{PE}	=	upper limit for the number of PEs in the internal layer.
Row_{num}	=	number of rows in training data.
out_{PE}	=	number of PEs in the output layer.
$input_{PE}$	=	number of PEs in the input data.

Const = constant between 5 and 10.

Neurodynamics rules

Neurodynamics represents the learning rules and transfer activation functions that represent a specific network. One of the most popular learning rules used by backpropagation networks is the Generalized-Delta-Rule developed by Rumelhart, Hinton and Williams [6]. Extensions of the generalized delta rule implemented in the NeuralWorks® [9] tool are the Cumulative-Delta-Rule which accumulates weights changes over several examples and the Normalized Cumulative Delta Rule.

The NeuralWorks® [9] tool incorporates five types of activation functions: sigmoid, hyperbolic tangent, sine, linear, and Digital Neural Network Architecture (DNNA) functions. The selection of a transfer function is entirely determined by the type of data and the requirements of the network. The recommended activation functions are the sigmoid activation function which is best for learning about an "average" behavior while the hyperbolic tangent activation function is suited for learning about "deviation" from the average.

Process of training and testing the network

Selecting the correct network configuration has a substantial impact on the network results. The basic training processing involves the presentation of the input data with the desired output. The network then adjusts its internal presentation by carrying out an iteration procedure to correct the error to produce acceptable results. This iteration process continues until it runs for a specific time or the network converges to acceptable levels. Usually, the number of iterations is specified as the number of learnings in the run menu, or as an acceptable error in the *RMS* diagnosis tool. Once the network is trained and converges, the test set is presented to the network sequentially only once to increase the confidence of the network performance and accounts for accuracy. During the process of learning the network is monitored visually by graphical instruments provided by the software to observe the learning process and to adjust any configurations that might arise.

Network monitoring

The network performance is monitored during training by **confusion matrix, weight histogram, Root-Mean-Square** (RMS) and **classification rate** diagnostic instruments provided by the tool to achieve a better understanding of the network performance. The **confusion matrix is** a visual display diagnostic instrument used to monitor the performance of each network output processing element and compares it to the desired output. The **x-axis along the confusion matrix** provides the network output while the y-axis provides the desired output. The interior quadrants are discretized into bins to show the network outputs. A value of one means an excellent correlation between the desired and network output.

The *weight histogram* instrument is used to monitor the network performance and is created by selection within the network menu. The *weight histogram* provides a normalized histogram of all the variables in the network that changes during the training session.

The RMS error instrument provides a measure of the output network performance over the number of training iterations. This instrument is created by selection within the network menu. The RMS error calculates each error signal in the output layer, adds them up, then divides these by the total number of processing elements to obtain the average value for the output layer. This results in calculating the square root of that average. Mathematically, the RMS error is computed from [11]

$$RMS = \sqrt{\frac{\sum_{pt}\sum_{oPE}(d_{out} - y_{out})^2}{n_{pt}\ n_{oPE}}} \tag{11}$$

where:

RMS	=	*R*oot-*M*ean-*S*qaure.
$\sum_{pt}$	=	summation over all patterns in the training set.
$\sum_{oPE}$	=	summation over all output processing elements.
d_{out}	=	desired output.
y_{out}	=	network output.
n_{pt}	=	number of patterns in training set.
n_{oPE}	=	number of processing elements in the output layer.

The *classification rate* is a visual representation of desired output to the actual network output in a two dimensional form.

One of the main factors that control convergence is the *epoch size*. The *epoch size* is the number of data presentations over which weight changes in the network are accumulated. This value is set up in the dialog box of the I/O menu before the network commences its learning. The *epoch size* can be tuned and adjusted to provide better learning procedures by monitoring as it evolves.

Network optimization requirements

Network performance is crucial which is a part of optimization process after the network is set and trained. This can be done by tuning the *internal layer size*, *epoch size* and *learning rates* to obtain reliable network.

Deployment of workable network

The final network can be deployed as part of the system application after it is completely trained and tested. This can be accomplished by either converting the trained network into a 'C' subroutine provided by the *FlashCode* [9] and linking it with a main 'C' source code or by entering data interactively through the keyboard and getting results instantaneously. However, for a large number of input and test sets, an input ASCII file can be called within the NeuralWorks® [9] since the interactive is not recommended due to its prolonging of entering data and the possibility of entering the wrong data.

Development of trained networks

The trained and deployed networks require continuous monitoring and maintenance during implementation to check reliability. Developing and debugging the trained, testing and creating maintenance are part of the network development which allows computers to perform tasks that would otherwise require continuous human input and attention.

Previous Experimental Model Evaluation

The behavior of a beam-column joint was investigated experimentally and analytically where empirical relationships established by Nirjar [12] for the structural behavior of cast-in-situ beam-column joints under static loading conditions. The study investigates the relationship between the behavior of beam-column joints and geometrical shape, amount and size of steel reinforcement, fixed beam and column cross-sectional dimensions and concrete strength. Tests were carried out on a total of 34 specimens under several conditions. The three-dimensional beam-column corner joint, commonly used in reinforced concrete building frames, provided guidance in the selection of the test specimen dimensions. The corner beam-column joint was subjected to a complex stress distribution due to the effect of biaxial bending forces. Figure 5 shows how a beam-column joint was assembled by Nirjar [12]. The beams selected where based on spans common in multi-storey building frames. These usually vary from 3.5 to 16 meters where the column height varies only between 2.5 and 4.5 meters. A column height of 550 mm with square cross-section dimensions of 100x100 mm was selected. The beam dimensions were 75 mm wide by 125 mm deep with a cantilever span of 350 mm and a loading position of 300 mm from the column face. The members were designed according to the British Code of Practice of the 1972 CP110 [13,154] and checked by the American Concrete Institute, ACI code 318-71 [14] and the first recommendation of the ACI-ASCE Committee Report 352-76 [15].

Application of Neural Networks in Prediction

Theoretical derivations of models of concrete structures are very complex due to the complexity of the structural shapes and the presence of different material properties. A

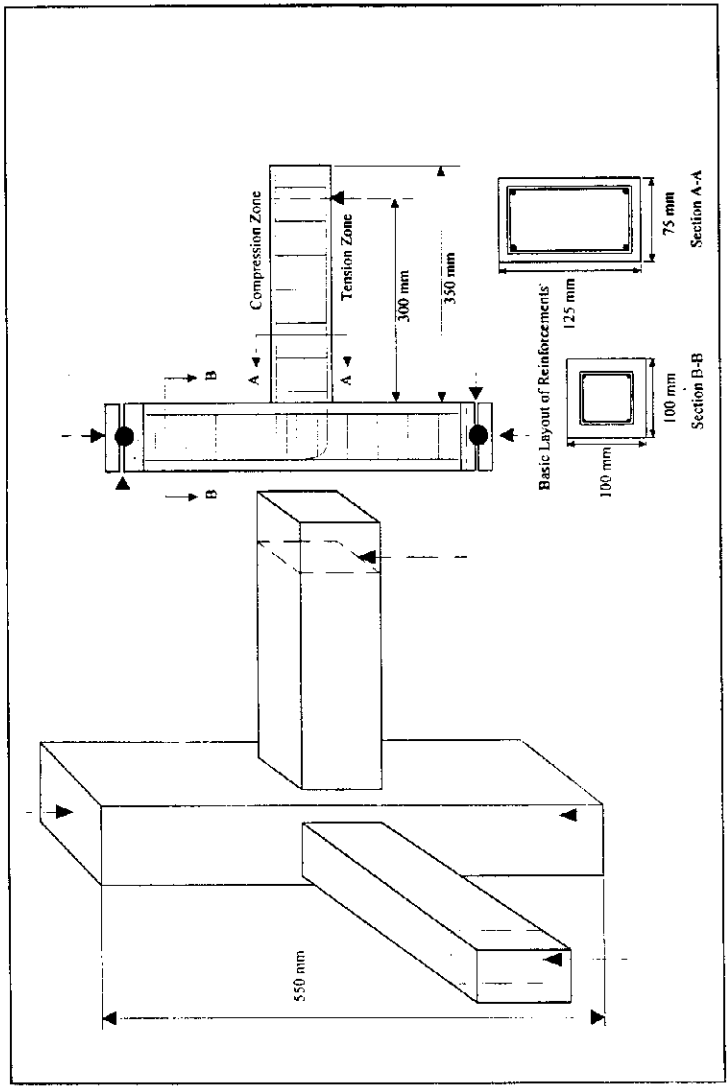

Fig. 5. Corner beam-column joint details.

reinforced concrete beam-column joint model addresses this complex behavior of yielding of the ductile material, brittle fracture of the concrete, localization of multi-stresses, micro-cracking and different loading conditions, all of which make numerical implementation of such a complex structure very difficult, if not impossible.

The problem and the different activities encountered when investigating experimental problems involve two main classes of activity: ***experimental modelling*** and ***analytical computation***. During the experimental stage, models are built to describe the relevant state of the structure. This is achieved by building a simplified structure of selected material properties and geometrical shapes which is often time-consuming and very costly. On the other hand analytical models involve parameters that can be selected to give the best fit for given real data usually supplied from experimental models. These analytical computations are adopted to establish a general procedure to derive several parameters for specified conditions and in such cases it is obvious that the neural network approach can be adopted to determine the missing parameters.

Civil engineering confronts massive complex mathematical computation in which parallelism can provide a great potential utilization of neural network in civil engineering in general and structural engineering in particular. Recent developments in neural networks and computer technology have merged to incorporate advanced predictive capability solutions, which were not available before and simulate humans recognizing similarities in patterns to reach suitable solutions.

This paper is to demonstrate a ***concept*** and ***methodology***, rather than to build a full-scale knowledge-based system model, by incorporating most of the fundamental aspects of a neural network in solving the complex non-linear mapping for a beam-column joint. Generally, it may be possible to identify certain parameters and allow the neural network to develop the model and account for the observed behavior without relying on a particular algorithm but depending entirely on the manipulation of numerical data to extrapolate the missing parameters.

Therefore, in order to enhance the efficiency and effectiveness of neural networks in the study of beam-column joints and extrapolation of the stress-strain relationships, the study is formulated on the basis of three stages: **preparatory, renew** and **final** stages where only the preparatory stage is discussed in this paper. A general neural network technique to analyze reinforced concrete beam-column joints to axial load and bi-axial bending was investigated by Jadid [16].

Preparatory stage

Experimental results are of great importance in improving codes of practice in general and understanding in this case the behavior of beam-column joints. The preparatory stage basically involves re-evaluating the previous experimental model for all 34 test specimens and also relying entirely on the analytical formation produced by Nirjar [12].

Prior to implementation of the neural network, intensive numerical programming was performed to generate the desired data required by the network. The experimental data was also used as an enhancement to the numerical data. The approach is to generate training and test data is organized according to intended requirements so that certain data are not presented to the network, thus ensuring the capability of the network to recognize and predict the missing desirable data.

The process starts to examine the experiments and analytical results by performing numerical computation for each series in the group. Having identified the specific group for neural network prediction application, two data files are established as training data and test data. The next step is to select and setup a specific network for certain tasks and begins training, adjusting, retraining and manipulating network parameters until a final acceptable trained network is obtained. The trained network is then deployed to predicate the missing series of that group by providing only the required input missing data and expecting that the network will provide results close to the desired output. The aim is to verify and observe the ability of the neural network in working out the missing series. For positive response identifications the results can provide considerable information and knowledge because of the capability of the network to predict a result for the series that is normally obtained through experimental procedure.

Evaluation is achieved by selecting concrete cylinder strengths of 20, 25, 30, 35 and 40 N/mm^2 and training each curve individually for selected points along the curve. With the numerical methods, the curves proposed by Nirjar [12] are used to generate the stress-strain relationship for each concrete strength. Certain points along each curve which are not generated are used to predicate the neural network results. Experimental data by Nirjar [12] is also used as part of the training data to enhance the network performance.

The process of determining the correct neural networks architecture with topology and neurodynamics requires delicate techniques to avoid underfitting, which results in selecting fewer internal layers and processing elements (PEs), and to avoid overfitting, which results in more internal layers and PEs. A balance is required which will produce a neural network that will not memorize the data or lack the ability to generalize. A standard backpropagation network is selected for all the concrete cylinder strengths. These are composed of one PE in the input layer representing the strain, 10 PEs in the internal layer and one PE in the output layer representing the stress. The Delta-Rule learning rule was implemented with a learning rate of 0.3 for the internal layer, 0.15 for the output layer and with a momentum of 0.5. A Hyperbolic Tangent transfer function is used which performs effectively with the Delta-Rule. An *epoch* size of 16 was selected with RMS convergence criteria of 0.3%. Table 1 shows the full parameters which were used at the preparatory stage to process the network predictions. Table 1 also provides the network results performance. The evaluation results are shown in Table 2, 3, 4, 5, and 6 for concrete strengths of 20, 25, 30, 35 and 40 N/mm^2 respectively. The strain values in the shaded area were passed to the trained network to predict the stress-strain relationship.

Table 2 shows that large errors occur at high concrete strengths in the experimental testing. The results obtained are encouraging in most cases and the error computation was calculated for the neural network and compared to the experimental and analytical results. This demonstrates the potentiality of adopting such a procedure to minimize the time required to record the test data by relying on specific points on the curve and allowing the neural network to obtain the results.

Table 1. Parameters used in stress-strain relationship for preparatory stage (Jadid [16])

f'_c N/mm^2	Input data sets			Network results			
	Training set	Test set	Cycle learn	Confusion matrix		Classification rate	
				Training	Test	Training	Test
20	82	12	27757	0.9999	0.9794	1.0	0.8750
25	87	12	14118	0.9994	0.9963	1.0	1.0
30	91	14	29383	0.9998	0.9994	1.0	1.0
35	94	13	31889	0.9999	0.9999	1.0	1.0
40	85	12	78619	0.9999	0.9999	1.0	10

Table 2. Stress-strain comparison between Nirjar [12] and neural network for f'_c = 20 N/mm^2 (Jadid [16])

Nirjar [12]		Neural networks		Computation of error	
Strain ε *10^{-3}	Experimental = E f (N/mm^2)	Analytical = A f (N/mm^2)	Network = N f (N/mm^2)	% Error = Abs N. (w.r.t) E.	(I-network/desired)*100 N. (w.r.t) A.
0.45	9.50	9.62	9.99	5.16	3.85
0.90	16.00	14.98	16.02	6.38	6.51
1.35	19.00	19.05	19.45	2.37	2.10
1.80	20.00	19.99	19.96	0.20	0.15
2.25	19.25	19.77	19.92	3.48	0.76
2.70	17.50	19.05	19.35	10.57	1.57
3.15	15.50	18.13	18.24	17.68	0.61
3.55	14.50	17.26	17.27	19.10	0.06

Table 3. Stress-strain comparison between Nirjar [12] and neural network for f'_c = 25 N/mm² (Jadid [16])

Nirjar [12]		Neural networks		Computation of error	
Strain	Experimental = E	Analytical = A	Network = N	% Error = Abs (1-network/desired)*100	
ε *10⁻³	f (N/mm²)	f (N/mm²)	f (N/mm²)	N. (w.r.t) E.	N. (w.r.t) A.
0.50	11.75	12.08	12.56	6.89	3.97
1.00	20.00	20.21	21.67	8.35	7.22
1.50	24.00	24.04	24.57	2.38	2.38
2.00	25.00	25.00	24.97	0.12	0.12
2.50	24.00	24.47	24.58	2.42	0.45
3.00	22.50	23.30	23.38	3.91	0.34
3.35	21.00	22.33	22.34	6.38	0.04

Table 4. Stress-strain comparison between Nirjar [12] and n eural network for f'_c = 30 N/mm² (Jadid [16])

Nirjar [12]		Neural networks		Computation of error	
Strain	Experimental = E	Analytical = A	Network = N	% Error = Abs (1-network/desired)*100	
ε *10⁻³ f	(N/mm²)	f (N/mm²)	f (N/mm²)	N. (w.r.t) E.	N. (w.r.t) A.
0.54	14.50	14.43	14.88	2.62	3.84
1.08	24.50	24.36	25.72	4.98	5.58
1.60	28.75	28.90	29.29	1.88	1.35
2.15	30.00	29.99	29.95	0.17	0.13
2.70	29.00	29.10	29.22	0.76	0.41
3.00	28.00	28.22	28.26	0.93	0.14
3.20	27.00	27.55	27.54	2.00	0.04

Table 5. Stress-strain comparison between Nirjar [12] and neural network for f'_c = 35 N/mm² (Jadid [16])

Nirjar [12]		Neural networks		Computation of error	
Strain	Experimental = E	Analytical = A	Network = N	% Error = Abs (1-network/desired)*100	
ε *10⁻³	f (N/mm²)	f (N/mm²)	f (N/mm²)	N. (w.r.t) E.	N. (w.r.t) A.
0.55	16.50	16.08	16.65	0.91	3.54
1.10	28.50	27.73	29.54	3.65	6.53
1.65	33.50	33.55	34.19	2.06	1.91
2.20	35.00	35.00	34.96	0.11	0.11
2.85	34.00	33.71	33.77	0.68	0.18
3.10	33.00	32.81	32.80	0.61	0.03

Table 6. Stress-strain comparison between Nirjar [12] and neural network for f'_c = 40 N/mm^2 (Jadid [16])

Nirjar [12]		Neural networks		Computation of error	
Strain	Experimental = E	Analytical = A	Network = N	% Error = Abs (I-network/desired)*100	
ε *10^{-3}	f (N/mm^2)	f (N/mm^2)	f (N/mm^2)	N. (w.r.t) E.	N. (w.r.t) A.
0.56	18.00	17.67	18.26	1.44	3.34
1.125	33.00	31.09	32.77	0.70	5.40
1.690	38.50	38.12	38.78	0.73	1.73
2.250	40.00	40.00	39.95	0.13	0.13
2.800	39.25	38.96	39.00	0.10	0.10
3.00	38.25	38.20	38.22	0.08	0.05

Conclusion

The most accurate information is produced by experimental testing which reproduces the actual structural behavior, monitored through instrumentation. The changes in structural behavior are observed as the experiment progresses. However, with increases in the cost of experimental testing, neural networks can be implemented to assist with or speed up the testing procedures by providing an alternative method of generating the required data which will consequently reduce the overall cost.

Identifying the potential applications of neural networks in structural engineering is perhaps the most skilful and difficult task facing the engineer and requires correct procedures which is a mixture of art and science. This paper intends to significantly integrate numerical algorithm computations and neural network applicability in the extrapolation of stress-strain relationships for reinforced concrete sections of reinforced beam-column joint behavior. Emphasis is particularly concentrated on shifting away from the complicated numerical routines computation to the manipulation of complex relationships among data by neural network. It is an effective approach to capture the fundamental aspects of the patterns in the data and categorize it as a form of classification which results in feature extraction in a form recognizable by the neural network. The power of the neural network predictivity lies in the flexible and explicit data implementations rather than the representations of algorithmic procedure. The encoded trained network can automatically retrieve valuable information with less time and can be updated for future use without intensive numerical or analytical formulation.

The results clearly indicated that the neural network application represents a method of evaluating the stress-strain relationships of a cross-section which is comparable with established methods using experimental and analytical procedures.

The author has demonstrated the concept of backpropagation in artificial neural networks to assist experimental testing. The results demonstrated that neural network methodology offers potential and valuable alternatives to extrapolate from experimental data. Instead of carrying out extensive experimental tests, neural networks offer an alternative procedure which requires less experimental specimen, concrete strengths, specific bar arrangements and maximum and minimum bar sizes. By constructing a model that allows for different experimental testing combinations with numerical analysis potentiality, neural network paradigms provide an advance in the prediction of parameters that have previously required repetition of the experimental work.

The concept can be applied to any engineering field or area which requires extensive experimental work and produces data based on analytical or numerical methods.

References

[1] McCulloch, W. C. and Pitts, W. "A Logical Calculus of the Ideas Immanent in Nervous Activity". *Bulletin of Mathematical Biophysics*, 5 (1943), 115-133.

[2] Hebb, D. O. *The Organization of Behavior*. New Yrok: John Wiley, 1949.

[3] Rosenblatt, F. "The Perceptron: A Probabilistic Model for Information Storage and Organization in the Brain". *Psychological Review*, 65 (1958), 386-408.

[4] Minsky, M. and Papert, S. *Perceptrons*, Cambridge, MA: MIT Press, 1969.

[5] Hopfield, J. J. "Neural Networks and Physical Systems with Emergent Collective Computational Abilities". *Proceedings of the National Academy Sciences*, 79 (1982), 2554-2558.

[6] Rumelhart, D. E., Hinton, G. E. and Williams, R. J. "Learning Internal Representations by Error Propagation." In: *Parallel Distributed Processing: Explorations in the Microstructures of Cognition*, (Eds.). Rumelhart, D. E., Hinton, G. E. and Williams, R. J. Cambridge, MA USA: MIT Press, 1 (1986), 316-362.

[7] Jadid, M. N. and Fairbairn, D. R. "The Application of Neural Network Techniques to Structural Analysis by Implementing an Adaptive Finite Element Mesh Generation". *Artificial Intelligence for Engineering Design, Analysis and Manufacturing (AI EDAM)*, 8 (1994), 177-191.

[8] Jadid, M. N. and Fairbairn, D. R. "Adjustments of Error by Neural Networks for the Shear Stress Carried by the Stirrups of a Beam-Column Joint". *Expert Systems with Applications*, 9 No. 3, (1995), 257-269.

[9] NeuralWare Inc. "Neural computing: A Technology Handbook for Professional II/Plus and NeuralWorks Explorer". *Technical Publications Group*, Pittsburgh, PA., USA: 1993.

[10] MathSoft Inc. "Mathcad 4.0: User's Guide: Windows Version." *MathcadSoft Inc.*, 1993.

[11] Dayhoff, J. *Neural Networks Architecture*, New York: Van Nostrand Reinhold, 1990.

[12] Nirjar, R. S. "Strength and Behavior of Reinforced Concrete Beam Column Joints under Bi-axial Bending". *Ph.D Thesis*, Department of Civil Engineering and Building Science, The University of Edinburgh, 1977.

[13] British Standard Institution. "The Structural Use of Concrete Design Material and Workmanship". *CP110 (Part 1)*, 1972.

[14] American Concrete Institute. "Building Code Requirements for Reinforced Concrete", *ACI Code - 71*, Detroit, 1971.

[15] ACI-ASCE Committee 352. "Recommendations for Design of Beam-Column Joints in Monolithic Reinforced Concrete Structures". *ACI Journal*, Proceedings, 73, No. 7 (July 1976), 375-393.

[16] Jadid, M. N. "The Application of Neural Network Techniques to the Analysis of Reinforced Concrete Beam-Column Joints Subjected to Axial Load and Bi-axial Bending." *Ph.D Thesis*. Department of Civil and Building Science, The University of Edinburgh, 1994.

تطبيق تقنية الشبكة العصبية لاستنتاج علاقة الإجهاد والتمدّد لمقاطع الخرسانة المسلحة

منصور بن ناصر الجديد

قسم علوم وتقنية البناء ، جامعة الملك فيصل ، ص.ب ٣٠٩٧٣ ، الخبر ٣١٩٥٢ ، المملكة العربية السعودية ، البريد الإكتروني: 100407.2437@compuserve . com

(استلم في ١٩٩٦/٥/٢٧م ؛ وقبل للنشر في ١٩٩٦/١٠/٩م)

ملخص البحث . يقدّم هذا البحث دراسة تطبيق تقنية الشبكة العصبية (Neural Networks) من الذكاء الاصطناعي لاستنتاج علاقة الإجهاد والتمدّد لمقاطع الخرسانة المسلّحة. وتطلّبت الدراسة استخدام التحليل العددي وذلك للحصول على بيانات تشابه النتائج المختبرية التي أُجريت على عيّنات الخرسانة المسلّحة . ولقد تمّ استعمال نُظم التقارب وذلك بتطبيق الشبكة العصبية خلال ما يسمى بـ Backpropagation Algorithm . وقد أظهرت الدراسة فعاليّة تقنية الشبكة العصبية ببناء تجارب وهميّة تشابه التجارب المختبرية وإمكانيّة استخلاص عدّة متغيّرات بواسطة الشبكة العصبية وتكون متّفقة مع التجارب المختبرية . أثبتت الخطوات الدراسية إمكانية بناء علاقات رياضيّة معتمدة على ترتيب بيانات عدديّة دون الخوض في التجارب المختبرية .