Computer Assisted Mechanics and Engineering Sciences, **13**: 125–159, 2006.

Neurocomputing in the analysis of selected inverse problems of mechanics of structures and materials

Zenon Waszczyszyn[1,2], Leonard Ziemiański[2]

[1] *Institute of Computer Methods in Civil Engineering, Cracow University of Technology Warszawska 24, 31-155 Kraków, Poland*

[2] *Chair of Structural Mechanics, Rzeszów University of Technology W. Pola 2, 35-959 Rzeszów, Poland*

(Received in the final form February 15, 2006)

The main goal of the paper is to show great potential of Artificial Neural Networks (ANNs) as a new tool in the identification analysis of various problems in mechanics of structures and materials. The basics of ANNs are briefly written focusing on the Back Propagation Neural Networks (BPNNs) and their features and possibilities in the analysis of inverse problems. Two groups of problems are analyzed: I) BPNNs are used in five problems as independent tools for the parametric identification and implicit modelling of physical relations, II) BPNNs are parts or procedures in three hybrid FEM/ANN systems or programs. Using measured eigenfrequencies the following problems are discussed: 1) identification of damage parameters of a steel beam, 2) attachment of an additional mass to increase the accuracy of prediction of damage parameters in a beam, 3) identification of location an additional mass attached to a steel plate. Implicit simulation of physical relations is discussed on two problems: 1) concrete fatigue durability of concrete as a function of concrete strength and characteristics of fatigue cycles (besides BPNN also the Fuzzy Weight NN was applied), 2) soil-structure interaction of displacement response spectra of a real building subjected to paraseismic excitations (besides BPNN the application of Kalman filtering is discussed for the NN learning). The following problems of Group II are investigated: 1) using BPNN in the hybrid Monte Carlo method for the reliability analysis of a steel girder, 2) application of BPNN to the calibration of control parameters in the updating of a FE program for dynamic analysis of a plane frame, 3) on-line methods for the NN constitutive model formulation basing on measurements of structural displacements.

1. Introduction

Artificial neural networks, called for short Neural Networks (NNs), belong to "biologically" inspired methods and together with fuzzy systems and genetic algorithms (or, more generally, evolutionary algorithms, systems and strategies) they are new tools for information processing. Their computer simulations create the so-called soft computing (also called intelligent computing, cf. e.g. [17]). Neurocomputing has been applied to the analysis of a great amount of problems in science and technology, see [13]. This concerns also mechanics of structures and materials, which was reflected in the CISM lectures, see [47]. The presented paper in supported on lectures also delivered at an Advanced CISM Course, see [52] and on the latest results obtained by the authors' research groups.

NNs have features associated with their biological origin. They are massively parallel and they can process not only crisp, but also noisy and incomplete data. Neurocomputing can be used for the approximation and classification purposes. NNs have generalization features, i.e. they can be used for the analysis of problems, which obey certain rules which are learned during the NN training process. Such features enable NNs to be applied in both direct and inverse analysis. NNs advantages offer, in fact, complementary possibilities to standard computational methods, especially to the Finite Element Method (FEM). This opens the door to the formulation of hybrid computational strategies and computer programs, see [35, 48].

Different types of NNs can be used in the inverse analysis of structural mechanics problems, see [47, 51, 52]. The majority of engineering applications is related to the multilayer perceptron, called further Back-Propagation Neural Network (BPNN).

After a short presentation of the basics of BPNN and its selected modifications (FWNN – Fuzzy Weight NN and Kalman filtering used for the BPNN training) we focus our attention on two groups of problems. The first one corresponds to applications of BPNN as an independent new tool specified to the identification analysis. Three problems of such applications are related to using vibration eigenfrequencies as inputs of BPNNs to the parametric identification of damage in steel beams [2, 27] and the placement of an attached mass on a steel plate [43]. The fourth problem concerns the prediction of concrete fatigue durability by means of the BPNN and fuzzy network FWNN. The implicit modelling of the relation of the concrete fatigue durability was formulated as a function of concrete strength and parameters of loading cycles [16, 19]. The fifth problem deals with the soil-structure interaction corresponding to the mapping of the ground displacement response spectra to the basement spectra. In this problem besides standard BPNN, cf. [26], also the application of Kalman filtering to the BPNN learning is considered [23].

The next three problems are related to the hybrid approaches. A hybrid Monte Carlo (MC) method was formulated using FEM to the computation of patterns which are used for the training of a BPNN [21]. The trained network is explored to generate MC trials. A great numerical efficiency of this approach is illustrated on the reliability analysis of a steel girder. The second hybrid system corresponds to the discussion of a hybrid approach in the FE model updating [30]. BPNN is explored in the inverse analysis for calibrating the control parameters. The approach is discussed on an example of the updating of a FE model for dynamic analysis of a simple plane frame. The third discussed hybrid program was formulated for the on-line training of the NN constitutive model of material in real solids or structures [10]. Basing on measured structural responses the neural model of an equivalent material enables us to evaluate the degradation of material properties in real structures. This problem is illustrated on an example of a simple truss made of Osgood-Ramberg material [37] and for the elastoplasic plane stress analysis of a beam bending boundary value problem [38].

2. Some basics of used neural networks

Below only the network BPNN is described in more detail since it is was applied in all problems discussed in the paper. The other network FWNN (Fuzzy Weight NN) was used only in the analysis of concrete fatigue durability. Kalman filtering was applied for the BPNN training in the analysis of a soil-structure interaction problem. In order not to extend the basics of neural networks the FWNN and Kalman filtering algorithm are described in short in the corresponding Sections 3.4.2 and 3.5.3.

2.1. Back-Propagation Neural Network (BPNN)

2.1.1. Structure of BPNN

BPNN is a feed-forward, multilayer network of standard structure, i.e. neurons are not connected in the layer but they join the layer neuron with all the neurons of previous and subsequent layers, respectively. Fig. 1a shows an example of a BPNN composed of an input layer, two hidden layers and an output layer. The input layer serves only to introduce signals (values of input variables), and the hidden and output layers are composed of neurons (processing units) of the NN structure shown in Fig. 1b.

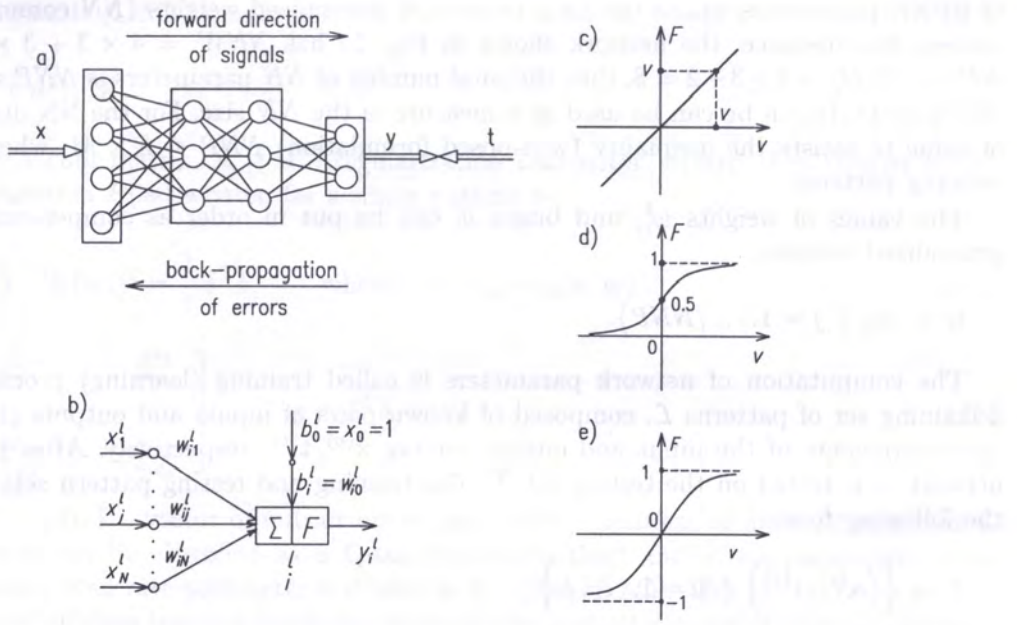


Fig. 1. a) Three-layer BPNN, b) Single neuron i in layer l, c, d, e) Identity, binary sigmoid and bipolar activation functions, respectively

The activation potential v_i^l of a single neuron i in the layer l is cumulated in the summing block $\sum$ and transformed by the activation function F to have only one scalar output y_i^l, see Fig. 1b:

$$y_i^l = F^l(v_i^l), \qquad v_i^l = \sum_{l=1}^{H_l} w_{ij}^l x_j + b_i^l = \sum_{l=0}^{H_l} w_{ij}^l x_j \,, \tag{1}$$

where: l – layer superscript $l = 1, \ldots, NL$ and NL – number of layers (hidden and output layers); w_{ij}^l – weights of connections, b_i^l – threshold parameter (bias) that can be treated as weight w_{ij}^0 corresponding to the unit signal $x_0 = 1$, cf. Fig. 1b, H_l – number of neurons in layer l.

From among many of the used activation functions only three functions are shown in Fig. 1c:

a) identity:

$$F(v) = v\,; \tag{2a}$$

b) sigmoid (logistic or binary sigmoid) function:

$$F(v) = \frac{1}{1+\exp(-\sigma v)} \in (0,1)\,, \qquad \frac{\mathrm{d}F}{\mathrm{d}v} \equiv F'(v) = \sigma\, F\,(1-F) \qquad \text{for } \sigma > 0\,; \tag{2b}$$

c) bipolar sigmoid:

$$F(v) = \frac{1-\exp(-\sigma v)}{1+\exp(-\sigma v)} \in (-1,1)\,, \qquad \frac{\mathrm{d}F}{\mathrm{d}v} \equiv F'(v) = \frac{\sigma}{2}(1+F)(1-F) \qquad \text{for } \sigma > 0\,. \tag{2c}$$

BPNN has a standard structure that can be written in short:

$$\text{BPNN}: \quad N - H_1 - H_2 - \cdots - H_{NL-1} - M, \tag{3}$$

where: N – number of inputs, H_l – number of neurons in the l-th hidden layer for $l = 1, \ldots, NL-1$; M – number of outputs. The weights w_{ij}^l and biases b_i^l are called network parameters. The number

of BPNN parameters equals the total number of generalized weights (*NN* connections and neuron biases). For instance, the network shown in Fig. 1a has $NNW = 4 \times 3 + 3 \times 3 + 3 \times 2 = 27$, $NNB = \Sigma_h H_h = 3+3+2 = 8$, thus the total number of *NN* parameters is $NNP = NNW + NNB = 27 + 8 = 35$. It can be can be used as a measure of the *NN* size. For the NN design purposes it is of value to satisfy the inequality (well-posed formulation) $NNP \leq L \times M$, where: L – number of training patterns.

The values of weights w_{ij}^l and biases b_i^l can be put in order as components of the vector of generalized weights,

$$\mathbf{w} = \{w_j \mid j = 1, \ldots, NNP\}. \tag{4}$$

The computation of network parameters is called training (learning) process. It is based on a training set of patterns $\mathcal{L}$, composed of known pairs of inputs and outputs (targets), which are the components of the input and output vectors $\mathbf{x}^{(p)}, \mathbf{t}^{(p)}$, respectively. After the training of the network it is tested on the testing set $\mathcal{T}$. The training and testing pattern sets can be written in the following form,

$$\mathcal{L} = \left\{ \left(\mathbf{x}^{(p)}, \mathbf{t}^{(p)} \right) \mid p = 1, \ldots, L \right\}, \tag{5a}$$

$$\mathcal{T} = \left\{ \left(\mathbf{x}^{(p)}, \mathbf{t}^{(p)} \right) \mid p = 1, \ldots, T \right\}, \tag{5b}$$

where: $\mathbf{t}^{(p)}$ – vector of known (target) outputs for the p-th patterns; L, T – number of patterns in training and testing sets, respectively.

2.1.2. Learning rules

The components of weight vector w_j (network parameters) are iteratively computed by means of the following formula,

$$w_j(s+1) = w_j(s) + \Delta w_j(s), \tag{6}$$

where: s – the number of iteration step. Let us write the learning formula for the weight increment in the following form,

$$\Delta \mathbf{w} = \eta \mathbf{d}, \tag{7}$$

where: η – learning rate, $\mathbf{d}$ – vector of search direction.

There are many learning rules, see e.g. [13, 17, 45, 47]. The simplest rule corresponds to the gradient method of steepest descent with the search vector $\mathbf{d} = -\mathbf{g}$,

$$\mathbf{g} = \{g_1, \ldots, g_{NNP}\} \equiv \mathbf{grad} E = \{\partial E / \partial w_i \mid i = 1, \ldots, NNP\}. \tag{8}$$

In Eq. (8) a scalar network error is used, e.g. the Least-Mean-Square-Error,

$$E = \frac{1}{2} \sum_{p=1}^{P} \sum_{i=1}^{M} \left(t_i^{(p)} - y_i^{(p)} \right)^2 = \frac{1}{2} \sum_{p} \sum_{i} \left(\delta_i^{(p)} \right)^2. \tag{9}$$

The main disadvantage of the gradient formula is its sensitivity to the selection of the learning rate value η. In case of flat error surface small values of η can significantly prolong the iteration process, for large η the process can be divergent. That is why many other learning rules were formulated. From among them only two rules which were explored in the analysis of problems discussed in the present paper are given below:

a) Levenberg–Marquardt (L-M) formula, see [17, 33],

$$\mathbf{d} = -(\mathbf{J}^T\mathbf{J} + \mu\mathbf{I})^{-1}\mathbf{g}, \tag{10}$$

where: J – the Jacobi matrix, $\mu \geq 0$ – regularization parameter. Matrix $\mathbf{J}$ is related to the network error function $E(\mathbf{w})$ written for a single pattern p,

$$E(\mathbf{w}) = \frac{1}{2}\sum_{i=1}^{M}(\delta_i(w))^2 = \frac{1}{2}\boldsymbol{\delta}^T\boldsymbol{\delta}, \qquad \text{where} \quad \delta_i = t_i - y_i(\mathbf{x}, \mathbf{w}), \tag{11a}$$

$$\mathbf{J}(\mathbf{w}) = \begin{bmatrix} \frac{\partial\delta_1}{\partial w_1} & \cdots & \frac{\partial\delta_1}{\partial w_n} \\ \vdots & \ddots & \vdots \\ \frac{\partial\delta_M}{\partial w_1} & \cdots & \frac{\partial\delta_M}{\partial w_n} \end{bmatrix}, \qquad \mathbf{g} = \mathbf{J}^T\boldsymbol{\delta}, \tag{11b}$$

where $\boldsymbol{\delta} = \{\delta_1, \ldots, \delta_M\}$ – vector of output errors, $n = NNP$ – number of network parameters. The L-M method can be classified as a Quasi-Newton method and a one-parameter global method (only one global rate parameter η is used in Eq. (7)), cf. [45]. This method is evaluated as one of the most efficient learning methods, especially for a small number of training patterns, cf. [33].

b) Resilient-propagation (Rprop) rule,

$$\Delta w_i(s) = -\eta_i(s)\,\mathrm{sgn}(g_i(s)), \tag{12}$$

where

$$\eta_i(s) = \begin{cases} \min(\eta^+\eta_i(s-1),\, \eta_{\max}) & \text{for} \quad g_i(s)\,g_i(s-1) > 0, \\ \max(\eta^-\eta_i(s-1),\, \eta_{\min}) & \text{for} \quad g_i(s)\,g_i(s-1) < 0, \\ \eta_i(s-1) & \text{otherwise}, \end{cases} \tag{13}$$

where the fixed parameters used in Eq. (13) are: $\eta^+ = 1.2$, $\eta^- = 0.5$, $\eta_{\max} = 50$, $\eta_{\min} = 10^{-6}$, cf. [33]. Rprop is a local rule since the learning rates η_i are related to each network parameter. This method is of heuristic type and it is frequently used for the NN training in case of a large number of training patterns, see [33].

During the iteration process all the patterns $p = 1, \ldots, L$ are presented. One, forward transmission of signals for all the patterns and back propagation of errors is called an epoch.

There are many questions concerning the design, training and examination of NNs. These problems are extensively discussed in literature, cf. e.g. [13, 17, 33, 45, 47, 52].

2.2. Measures of errors

Besides the Least-Square-Error E defined in Eq. (9) there are applied other error measures for evaluation of the accuracy of neural approximation, see e.g. [52]. The most popular are the Mean-Square-Error (MSE) and Root-Mean-Square-Error ($RMSE$),

$$MSE = \frac{1}{P}\sum_{p=1}^{P}\sum_{i=1}^{M}\left(t_i^{(p)} - y_i^{(p)}\right)^2, \qquad RMSE = \sqrt{MSE}, \tag{14}$$

where $t_i^{(p)}$, $y_i^{(p)}$ – target and neurally computed i-th outputs for p-th pattern.

For estimation of neural prediction the following relative errors are used,

$$\text{avr } ep = \frac{1}{P}\sum_{p=1}^{P} ep, \qquad \max ep = \max_{p} ep, \qquad \text{where} \quad ep = \left|1 - \frac{y^{(p)}}{t^{(p)}}\right| \cdot 100\%. \tag{15}$$

From among statistical parameters the standard error $\mathrm{St}\,\varepsilon_i$ and linear regression coefficient r_i are frequently used with respect to the set of pairs $\left\{(t_i, y_i)^{(p)} \mid p = 1, \ldots, P\right\}$,

$$\mathrm{St}\,\varepsilon_i = \sqrt{\frac{1}{P}\sum_{p=1}^{P}\left(t_i^{(p)} - y_i^{(p)}\right)^2}, \qquad r_i = \frac{\sum_{p=1}^{P}\left(t_i^{(p)} - \bar{t}_i\right)\left(y_i^{(p)} - \bar{y}_i\right)}{\sqrt{\sum_{p=1}^{P}\left(t_i^{(p)} - \bar{t}_i\right)^2 \sum_{p=1}^{P}\left(y_i^{(p)} - \bar{y}_i\right)^2}}, \tag{16}$$

where: $\bar{t}_i$, $\bar{y}_i$ – mean values of sets $\left\{t_i^{(p)}\right\}$, $\left\{y_i^{(p)}\right\}$ for the fixed subscript i.

2.3. Neural networks and inverse analysis

2.3.1. Mechanical system problems and selection of NN inputs/outputs

Feed-forward NNs can be applied to the analysis of different problems of mechanical systems (MS) depending on the input/output variables used. Following [40] a classification of MS problems corresponds to the selection of inputs and outputs as shown in Table 1.

Table 1. Mechanical system problems and relevant data

Problems	Inputs	Outputs
1. Mechanical system (MS) response simulation	Excitation variables and system parameters	Response variables
2. MS excitation simulation (identification)	Response variables and system parameters	Excitation variables
3. MS parameter identification	MS excitation and response variables	Parameters of MS
4. MS excitation and/or response assessment	All relevant measures of system conditions to be assessed	Assessment of system conditions

In case of one-to-one correspondence the computed response or a part of response variables can be used for the simulation of excitation. Obviously, it is possible to use mixed types of data, e.g. the data taken from theoretical models can be used for the BPNN training and experimental data can serve for testing.

Referring to mathematical models the simulation of MS corresponds to the direct analysis and the identification is related to the inverse analysis.

In the present paper the following inputs are used in the identification analysis:

- structural and/or material parameters,
- response of vibrating structures in spectral domains (response spectra, natural eigenfrequencies),

Selected outputs for the identification analysis are:

- structural parameters, e.g. strength of material, structural element stiffness, etc.,
- FE model control parameters,
- damage parameters.

2.3.2. Data and their preprocessing

Sets of patterns can be taken from:

- computer simulation of direct problems (pseudo-experimental data),
- tests on laboratory models or measurements on full scale structures (e.g. real buildings),
- generating noisy data i.e. superposing artificial perturbation (e.g. adding random noise) on pseudo-experimental data.

Original data are usually preprocessed in order to generate sets which can be efficiently explored during the NN training or testing process. Data preprocessing can be related to:

- standard analytical transformation (scaling, normalization, transition from the time to spectral domains),
- applications of special NNs, e.g. cascade BPNN, for computing the neural outputs and using them as inputs in the cascade package,
- introduction of artificial noise into input variables x_j enables us to increase the number of patterns corresponding to vector components $\tilde{x}_j$ related to the same output y. In what follows the noise set of standard error σ was used and the values σ_k are randomly selected K times from the range $\sigma_k \in [-3\sigma, 3\sigma]$ assuming probability density function (Npdf) with mean value $\mu = 0$ and variance σ^2,

$$\tilde{x}_{j(k)} = (1 + \sigma_k)\, x_j \,. \tag{17}$$

Depending on the nature of phenomena or performed tests the identification parameters can be estimated as crisp or fuzzy sets. That is why besides standard NNs (e.g. BPNNs) also fuzzy NNs (e.g. FWNNs) should be applied.

The main attention is paid to the parameter identification which is related to the so-called explicit modelling, see [9]. That means that certain structural or material characteristics are assumed as functions with unknown parameters which are to be calibrated during the identification process.

NNs open the door also to the implicit modeling, see [9]. This is related to one of NN features that nonlinear relations of input and output variables can be computed without assuming 'a priori' a form of functions with unknown parameters.

The problems considered above and corresponding tools are discussed on examples of engineering problems with special attention to structural experimental mechanics.

3. Neural network as an independent computational tool

3.1. Using structure eigenfrequencies as input variables

Dynamic characteristics of the considered structures can be used as input variables for the efficient parametric identification. In the following three examples the eigenfrequencies of natural vibrations are presented for the identification of damaged steel beams and placement of an additional mass attached to a steel plate. Two examples are related to laboratory tests performed at the Chair of Structural Mechanics of the Rzeszów University of Technology [27, 43].

3.1.1. Identification of damage parameters of a steel beam

A set of cantilever steel beams with an artificial crack (notch) was tested. The laboratory set-up shown in Fig. 2a was described in [27]. It is worth emphasizing that for the computer simulation purposes both the excitation caused by the modal hammer and responses were simultaneously measured in the time domain and then the accelerograms were transformed to the spectral domain. The Inertance Frequency Spectrum was used for computing inputs related to eigenfrequencies of the tested laboratory specimens using the following formula, see [6]:

$$H(\omega) = \frac{A(\omega)}{F(\omega)}\,, \tag{18}$$

where: $A(\omega)$, $F(\omega)$ – response and excitation transforms corresponding to acceleration and impact force, measured by the sensor and in modal hammer; $\omega = 2\pi f$ – natural frequency and its relation to frequency f [Hz]. The transforms are defined as

$$A(\omega) = A_r(\omega) + iA_i(\omega) = \int_0^{\infty} \exp(-i\omega t)\,\ddot{x}_g(t)\,\mathrm{d}t = \sum_{t=0}^{N-1} \exp(-i2\pi\omega t/N)\,\ddot{x}_g(t)\,\Delta t\,. \tag{19}$$

Of course, formula (19) is valid also for the force transform $F(\omega)$.

A set of $P = 40$ specimens with eight notch location $a = 8, 16, \ldots, 64$ cm and five depths $g = 2, 4, \ldots, 10$ mm was prepared, assuming a fixed notch width equal 1 mm. The vibrations of

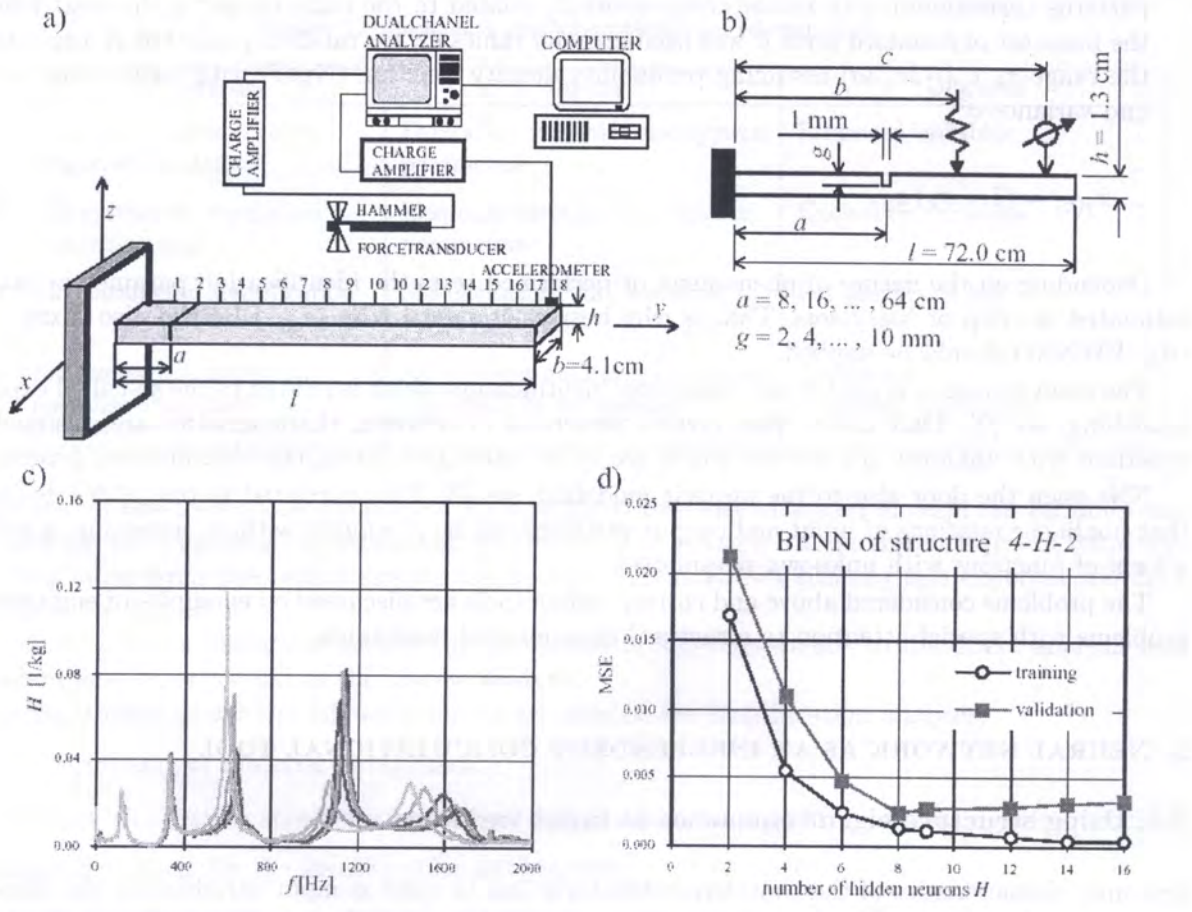


Fig. 2. a) Tested beam and laboratory set-up, b) Cantilever beam specimen, c) Inertance Frequency Spectrum functions $H(f; a, g)$, d) Cross-validation for finding the number of neurons H_{opt} in the hidden layer of BPNN: 4-H-1

the beam were excited by an impact applied at the distance b and accelerations were measured at the distance c, see Fig. 2b.

On the basis of inertance frequency function graphics of sequence frequencies f_j were obtained and then their increments,

$$\Delta f_j = 1 - \frac{f_j}{f_j^o}, \tag{20}$$

where: f_j^o, f_j – frequencies for lack of notch ($g = 0$) and at a fixed notch depth g and its location a, respectively. After scaling frequencies $\Delta \bar{f}_j \in [0.1, 0.9]$ they were used as input values and the outputs correspond to dimensionless variables $\bar{a} = a/l$, $\bar{g} = g/h$,

$$\mathbf{x}_{(4\times1)} = \{\Delta f_j^o \mid j = 1, \ldots, 4\}, \qquad \mathbf{y}_{(2\times1)} = \{\bar{a}, \bar{g}\}. \tag{21}$$

The set of $P = 40$ patterns was randomly split into $L = 35$ training patterns and $T = 5$ testing patterns. The same 10 randomly selected sets were used in the multifold cross-validation procedure (see [13]) for designing a BPNN with a hidden layer and structure 4-H-2, composed of binary sigmoid neurons in the hidden layer and linear outputs. SNNS computer simulator [53] and Rprop learning method were applied assuming the stopping number of epochs $S = 1500$. The results of cross-validation are shown in Fig. 2d, where the training and validation (testing) errors, $MSEL$ and $MSET$, respectively, were computed for each H (number of neurons in the hidden layer) as mean values for ten selected sets of training and testing patters. On the base of the above described

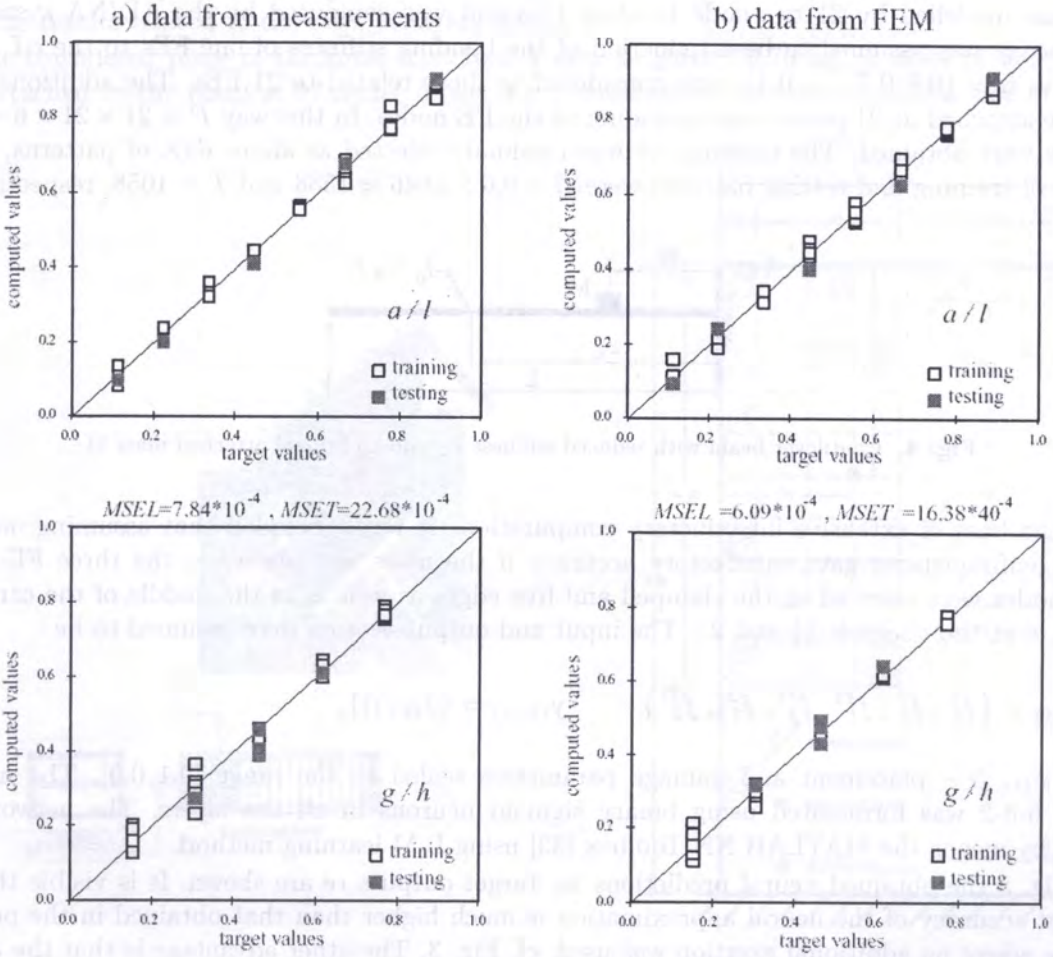


Fig. 3. Neural identification of notch location a/l and depth g/h for patterns taken from: a) Tests, b) FEM

validation process the optimal number of neurons $H_{opt} = 8$ was accepted with mean error values $MSEL = 13.25 \cdot 10^{-4}$ and $MSET = 24.87 \cdot 10^{-4}$. The network BPNN: 4-8-2 has a great number of parameters $NNP = 58$ so an over-fitting could happen. That is why the training errors $MSEL(H)$ are twice lower than $MSET(H)$ for $H \in [4, 8]$ and for higher $H > 8$ the validation errors increase.

In Fig. 3a there are shown the results of identification of the notch location a/l and depth g/h vs. target values. The results by BPNN: 4-8-2 correspond to a set taken from among ten selected sets, whose testing error $MSET = 22.68 \times 10^{-4}$ was the biggest one. The main conclusion from the above discussed results is that despite of the small number of training patterns, the results can be accepted if the testing errors are small.

In Fig. 3b the results of neural identification are shown for patterns generated by the FEM code ADINA [1] using the mesh 145×7 of plane FEs. The computations were performed for the same sets of patterns as those leading to results presented in Fig. 3a. Comparing the results obtained on the basis of FEM with those based on tests it is visible that they are close to each other not only because of similar network errors but also the distribution of computed values in Fig. 3.

3.1.2. Identification of damage using an additional mass

The accuracy of identification can be increased if additional exertions are performed to change the dynamic parameters of the structure considered, see [31]. According to this suggestion an additional mass M was attached to the damaged cantilever beam shown in Fig. 4. In [2] a steel beam of IPE200 cross-section was considered and the mass $M = 8.5$ kg was about 12% of the beam weight. The beam was modelled by 21 two-node bending FEs and was computed by the ADINA system [1]. The damage was assumed to be a reduction of the bending stiffness of one FEs to the αI, where six values $\alpha = \{0.8, 0.7, \ldots, 0.3\}$ were considered as those related to 21 FEs. The additional mass could be attached at 21 points corresponding to the FE nodes. In this way $P = 21 \times 21 \times 6 = 2646$ patterns were obtained. The training set was randomly selected as about 60% of patterns, so the number of training and testing patterns were $L = 0.6 \times 2646 \approx 1588$ and $T = 1058$, respectively.

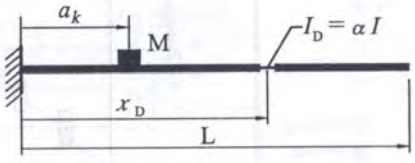


Fig. 4. Cantilever beam with reduced stiffness I_D and additional attached mass M

On the base of extensive introductory computations it was concluded that assuming only two basic eigenfrequencies gave satisfactory accuracy if the mass was placed at the three FE nodes. These nodes were selected at the clamped and free edges as well as in the middle of the cantilever beam, i.e. at the nodes 1, 11 and 21. The input and output vectors were assumed to be

$$\mathbf{x}_{(6\times1)} = \{f_1^1, f_2^1, f_1^{11}, f_2^{11}, f_1^{21}, f_2^{21}\}, \qquad \mathbf{y}_{(2\times1)} = \{\bar{x}_D, \bar{\alpha}\}, \tag{22}$$

where: $\bar{x}_D$, $\bar{\alpha}$ – placement and damage parameters scaled to the range $[0.1, 0.9]$. The network BPNN: 6-8-2 was formulated using binary sigmoid neurons in all the layers. The network was trained by means the MATLAB NN Toolbox [33] using L-M learning method.

In Fig. 5 the obtained neural predictions vs. target outputs re are shown. It is visible that the obtained accuracy of the neural approximation is much higher than that obtained in the previous example where no additional exertion was used, cf. Fig. 3. The other advantage is that the attachment of mass does not need the initial state of structure to be considered, e.g. to an undamaged structure as it was introduced in the increments of eigenfrequencies in Eq. (20).

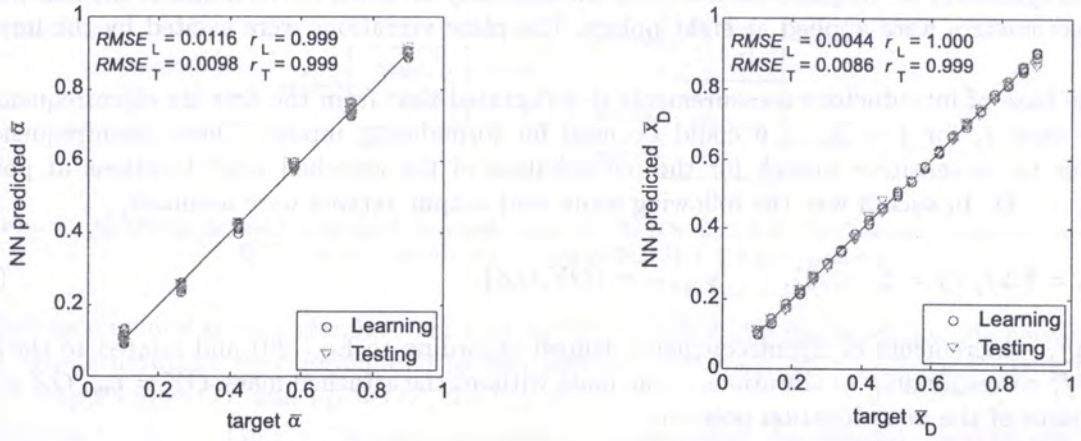


Fig. 5. Target vs. neurally predicted damage parameters: a) reduction of stiffness, b) placement of damage

3.1.3. Identification of location of mass attached to a steel plate

An interesting problem of parameter identification in a structural element was analyzed in [43] where the identification concerns the location of a mass attached to a steel plate, cf. Fig. 6b. All the patterns were formulated on the base of tests performed on a laboratory model, using an experimental set-up shown in Fig. 6a. In [42] both steel and aluminium plates were tested but below only the results related to the steel plate are discussed.

The considered plate of thickness $h = 10 \pm 1$ mm weighted 10.76 kg. A mass of weight 109 g was attached to the plate at 27 nodes of the 3×9 mesh shown on the right-hand side of Fig. 6b.

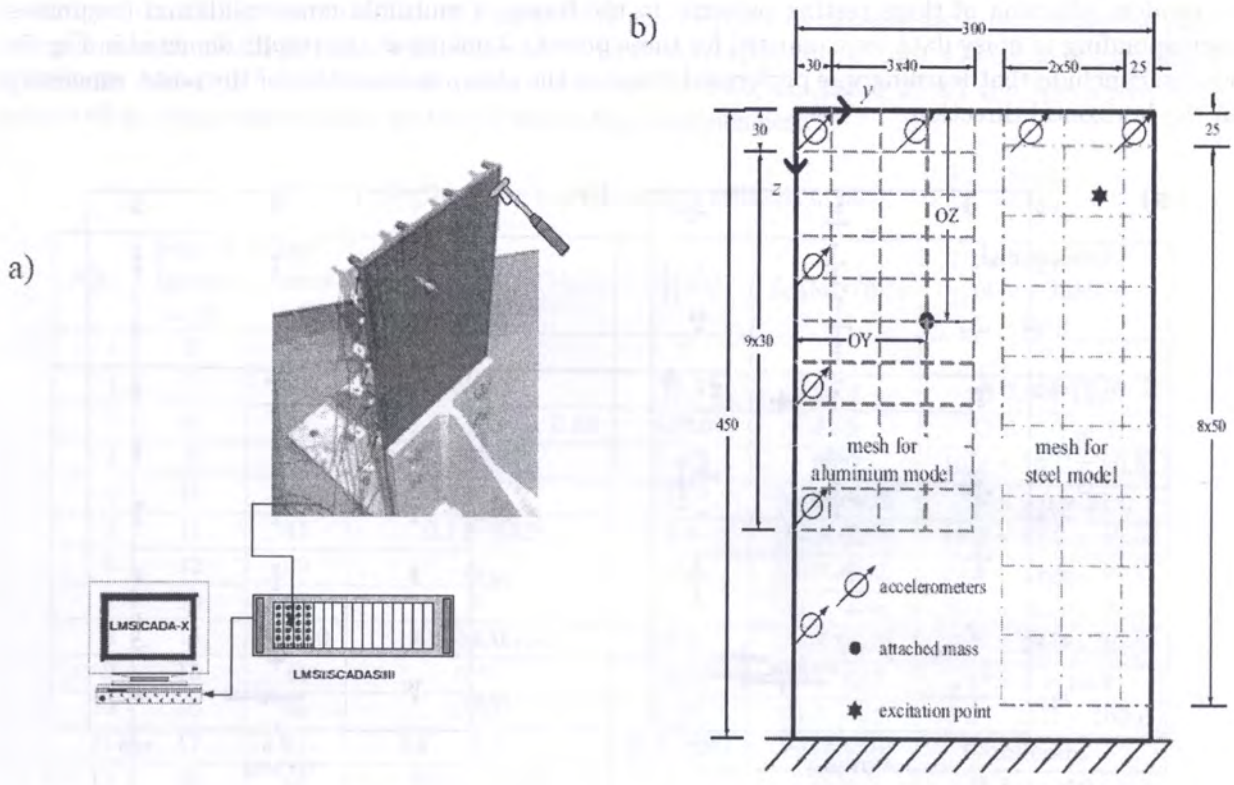


Fig. 6. a) Main parts of experimental set-up, b) Scheme of plate specimen with mesh for mass attachment, points of accelerometer application and a region of hammer impact

Because of symmetry of the plate the mass was subsequently attached to the nodes at the half-plate. The accelerometers were applied at eight points. The plate vibrations were excited by the impact hammer.

On the base of introductory measurements it was stated that from the first six eigenfrequencies only five ones f_j for $j = 2, \ldots, 6$ could be used for formulating inputs. These eigenfrequencies turned out to be sensitive enough for the perturbation of the attached mass locations at points $m = 1, 2, \ldots, 27$. In such a way the following input and output vectors were assumed,

$$\mathbf{x}_{(5\times1)} = \{\Delta f_j \mid j = 2, \ldots, 6\}, \qquad \mathbf{y}_{(2\times1)} = \{OY, OZ\}. \tag{23}$$

where: Δf_j – increments of eigenfrequencies defined according to Eq. (20) and related to the frequencies f_j^0 corresponding to vibrations of the plate without the attached mass, $OY = y_m$, $OZ = z_m$ – coordinates of the mass location point m.

The number of measurements corresponds to the number of the attached mass locations, i.e. to 27 points. In order to increase the number of patterns the artificial noise was added to the measured values according to formula (17). The measured set was randomly split into the sets composed of $L = 24$ and $T = 3$ patterns and then each pattern was perturbed 12 times by randomly selected artificial (Gaussian) noise with standard deviation $\sigma = 0.001$. In this way the training and testing sets were formulated as composed of $L = 24 + 12 \times 24 = 312$ and $T = 3 + 12 \times 24 = 39$ patterns, respectively.

The multifold cross-validation was used to design the network, cf. [13]. The training and testing sets were 100 times randomly selected and after an extensive training the number of binary sigmoid neurons in the hidden layer was evaluated as $H_{\text{opt}} = 5$. The network BPNN: 5-5-2 was designed using the simulator MATLAB NN Toolbox [33] and the L-M learning method. In Fig. 7a there are shown results of neural identification. Instead of the points corresponding to the perfect patterns the intervals are shown in Fig. 7a as associated with the noisy patterns. The points 1S, 2S, 3S are related to random selection of three testing patterns in the frame of multifold cross-validation (responses corresponding to noisy data were omitted for these points). Looking at the results depicted in Fig. 7a we can conclude that learning was performed worse at the clamped boundary of the plate, especially in the horizontal direction.

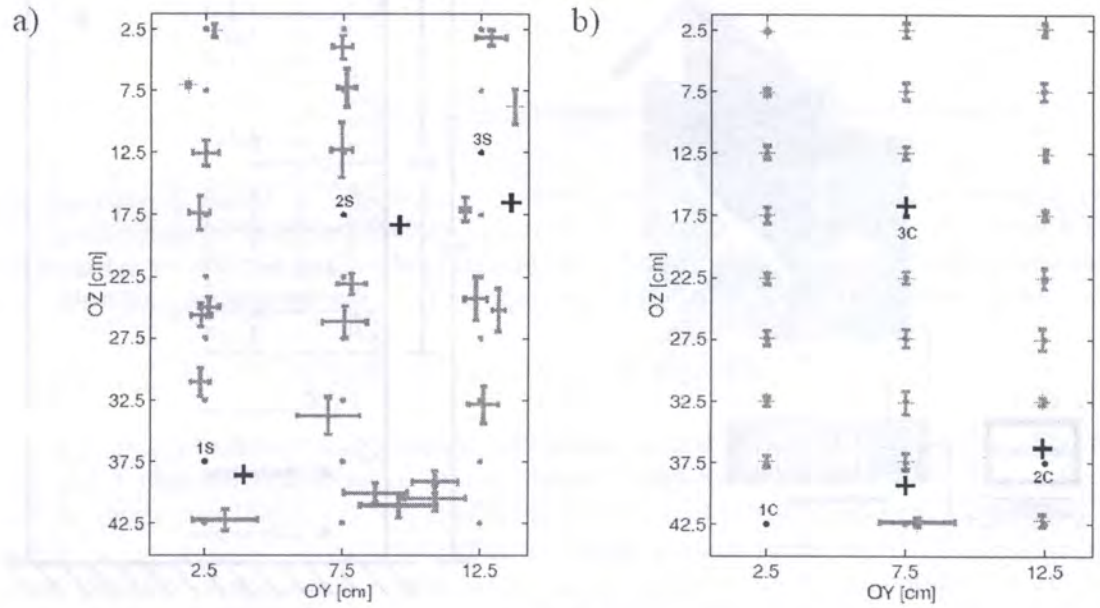


Fig. 7. Results of identification of mass location: by a) Standard network BPNN-M: 5-5-2, b) Cascade BPNN-I: 5-5-1 for $y_I = OY$, BPNN-II: 6-5-1 for $y_{II} = OZ$

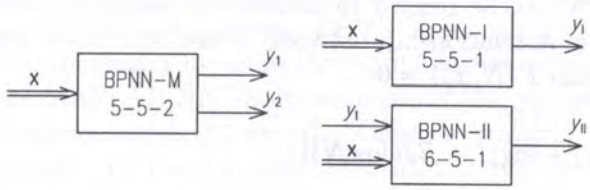


Fig. 8. a) Master network (standard, one-level network) BPNN-M: 5-5-2 , b) Cascade, two-level networks BPNN-I: 5-5-1 for $y_I = OY$ and BPNN-II: 6-5-1 for $y_{II} = OZ$

Besides one level master standard network BPNN-M: 5-5-2 also two level cascade networks were used. These networks were formulated as two networks BPNN-I: 5-5-1 and BPNN-II: 6-5-1 with single outputs $y_I = OY$ and $y_{II} = OZ$, see Fig. 8.

The results depicted in Fig. 7b show a significant improvement of the neural identification (testing points 1C, 2C, 3C were randomly selected and they are different from points 1S, 2S, 3S in Fig. 7a).

A very poor identification of the location of mass at point 1C is probably affected by the clamped edge conditions of the plate and satisfactory identification was obtained with respect to points 2C and 3C. Quite similar conclusions can be deduced for other randomly selected testing measurements.

3.2. Concrete fatigue durability

The main goal of this example is to show an application of BPNNs to the implicit simulation of physical relationships. This means that the network is used for the mapping of input variables to output variables without an *a priori* assumed form of physical relations. In such a way we can omit parametric identification related to the calibration of the relationship parameters, see [9].

Fatigue durability is defined as the number of load cycles N causing fatigue damage of plain concrete specimens. In [8] there was collected evidence corresponding to over 400 cubic or cylindrical specimens, tested in many laboratories in the years 1934-80, cf. Table 2. All the specimens were subjected to compressive loads within cycles at fixed frequencies.

Table 2. Experimental evidence collected in [8]

Nos.	Nos. of figures in [8]	References in [8]	R	f [Hz]	f_c [MN/m^2]	Dimensions of specimens [cm]
1	*2*	*3*	*4*	*5*	*6*	*7*
1	7	37	0.025	16.7	28.0	Ø 7.6 × 15.2
2	8	38	0.15; 0.38; 0.60; 0.88	150.0	41.0	Ø 5.1 × 10.2
3	9	39	0.44	0.025	28.0	10.2 × 10.2 × 30.5
4	10	40	0	5.0	[20.0, 30.0]	7.0 × 7.0 × 21.0
5	11	41	0.14; 0.75	7.5	[14.8, 32.7]	13.0 × 13.0 × 40.0
6	12	42	0	20.0	[20.0, 45.0]	10.2 × 10.2 × 50.8
7	13	43	0	20.0	[33.1, 44.8]	10.2 × 10.2 × 50.8
8	14	19, 20	0.044, 0.75	7.5	[14.8, 32.7]	13.0 × 13.0 × 40.0
9	15	44	0.05	16.7	25.5, 42.7	Ø 7.6 × 15.2
10	16	45	0.05	1.167	24.8, 33.1	15.2 × 15.2 × 162.6
11	17	21	0	[5.0, 16.7]	[20.0, 30.0]	different
12	18	46	0.074, 0.253	10.0	45.2	Ø 5.0 × 10.0
13	19	47	0	0.25	20.7	10.2 × 13.0 × 82.7
14	20	–	0	6.67, 15.0	26.2	15.0 × 15.0 × 15.0

The fatigue durability N can be related to mechanical properties of concrete and to the characteristics of the load cycle. A relationship between N and four input parameters x_j was deduced in [8] as an empirical formula $F(N, x_j) = 0$,

$$\log N = \frac{1}{A}\left[\log\left(C\, C_f/\chi\right) + \log(1 + BR\log N)\right], \tag{24}$$

where five basic variables were used: N – number of cycles associated with fatigue damage, $R = \sigma_{\min}/\sigma_{\max}$ – ratio of minimal and maximal stresses in a loading cycle, f [Hz] – cycle frequency, f_{cN}, f_c – ratio of fatigue strength of concrete f_{cN} and strentgh of concrete in compression f_c and their ratio $\chi = f_{cN}/f_c$. The main variables, used in Eq. (24) are: $A = 0.008 - 0.118\log(\sigma_I/f_c)$, $B = 0.118(\sigma_{II}/\sigma_I - 1)$, $C_f = 1 + 0.07(1 - R)\log f$, C – ratio of dynamic to static strengths for cyclic load, where besides basic variables there are also used critical stresses corresponding to initial and long term strength of concrete σ_I, σ_{II}.

The tests corresponding to data bases listed in Table 2 were split into three groups. Group I corresponds to 8 data bases with crisp values of concrete strength f_c (Nos. 1, 2, 3, 9, 10, 12, 13, 14 in Table 2). Group II is associated with fuzzy values of the strength related to interval $f_c \in [f_{c\min}, f_{c\max}]$ (Nos. 4, 5, 6, 7, 8, 11 in Table 2) and the data base No. 11 is related to interval inputs for both strength f_c and the frequency f.

3.2.1. Implicit modelling of concrete durability using standard network BPNN

The standard neural network was used for modelling the relationship $F(N, x_j) = 0$, where N – number of fatigue cycles, x_j – selected parameters of concrete and loading process.

The problem of fatigue durability prediction was formulated in [16, 19] as the following mapping,

$$\mathbf{x}_{4\times 1} = \{f_c, \chi, R, f\} \to y = \log N. \tag{25}$$

Table 3. Errors and statistical parameters of neural approximation for concrete durability

Simulator	avr $ep(V)$ [%]		r_V		St ε_V	
	L	T	L	T	L	T
BPNN: 4-5-1	13.6	20.5	0.871	0.855	0.701	0.777
FWNN: 4-5-5 for $\alpha = 1.0$	14.0	19.7	0.879	0.861	0.700	0.772
Formula (24)	17.7	26.3	0.843	0.843	0.873	0.991

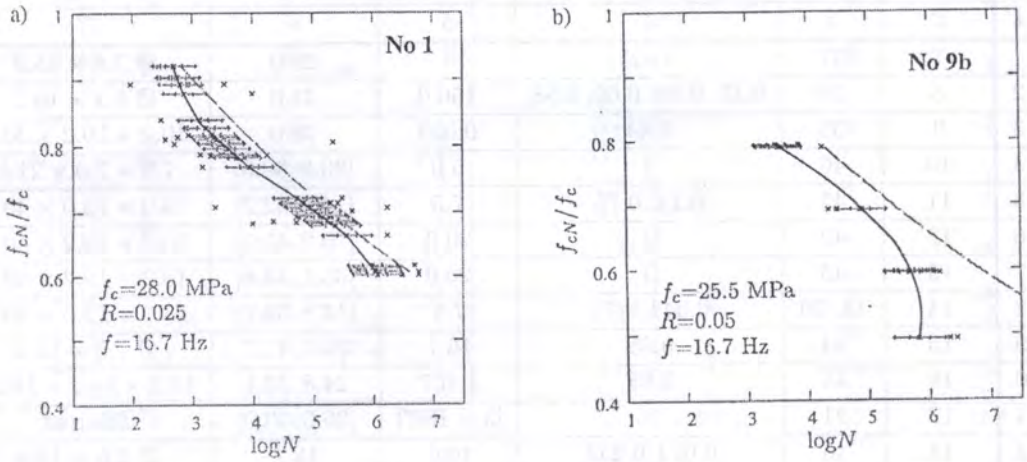


Fig. 9. Neural crisp and interval predictions (for $\alpha = 1.0, 0.9, 0.75$) of relation $\chi = f_{cN}/f_c - \log N$ for data taken from data banks: a) No. 3, b) No. 9b

218 patterns corresponding to Group I of tested specimens were randomly split into $L = 118$ learning patterns and $T = 100$ training patterns. BPNN: 4-5-4-1 was designed and trained as the network composed of binary sigmoid neurons in the hidden layers and linear output. The network was trained by means of the MATLAB NN Toolbox simulator [33] applying in [19] the gradient BP learning method with the momentum term, cf. [47]. Then the training was repeated in [16] using much more efficient L-M method. In Table 3 there are written errors corresponding to this network. It was stated that 85% of correctly predicted patterns was located within 20% of relative errors *ep*.

In two following Figs. 9a,b the results of neural predictions and by empirical formula (24) are shown for data banks listed in Table 2 as No. 1 and No. 9b (these numbers were introduced in Table 2 and No. 9b corresponds to the concrete strength $f_c = 42.7$ MPa). It is visible that the neural predictions fit better a nonlinear neural relations $\chi(\log N)$ than formula (24), especially for the data bank No. 9b.

3.2.2. Neuro-fuzzy network FWNN (Fuzzy Weight Neural Network)

In case of Group II of data banks some of input data were given in the interval form. In such cases we can apply a neuro-fuzzy network. In [39] the network FWNN was successfully used for predicting the concrete durability so below this network is briefly described.

FWNN is bases on formulation of a network using fuzzy sets and interval arithmetic. Following Zadeh's approach, cf. e.g. [17], the fuzzy set is defined with respect to pairs $(x, \mu_A(x))$,

$$A = \{(x, \mu_A(x)) \mid x \in X\}, \tag{26}$$

where: $\mu_A(x) \in [0.0, 1.0]$ – membership function (MF) of the set A which maps each element x into a membership grade (membership value) between 0 and 1. From among many activation functions, introduced in literature, only the triangular MF is shown in Fig. 10a.

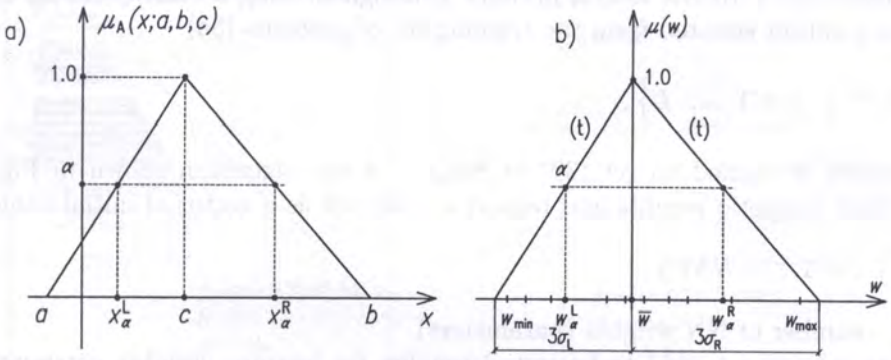


Fig. 10. a) Triangular membership function and α-cut, b) Triangular MF for weight w

One of the most important concepts of fuzzy sets is the concept of α-cut defined as follows,

$$A_\alpha = \{x \mid \mu_A(x) \geq \alpha,\ x \in X\} \qquad \text{for } \alpha \in [0, 1]. \tag{27}$$

A representation of α-cut is the *interval* which can be written as (cf. Fig. 7c):

$$A_\alpha = \left[x^L, x^U\right]_\alpha = \left[x_\alpha^L, x_\alpha^U\right]. \tag{28}$$

For α-cuts the interval arithmetic (addition, multiplication and subtraction, respectively) can be applied. In what follows we restrict our attention only to addition and multiplication of intervals $A_\alpha = [a, b]$ and $B_\alpha = [c, d]$,

$$\begin{aligned} A_\alpha + B_\alpha &= [a, b] + [c, d] = [a + c, b + d] \\ A_\alpha \times B_\alpha &= [a, b] \times [c, d] = [\min(ac, ad, bc, bd), \max(ac, ad, bc, bd)]. \end{aligned} \tag{29}$$

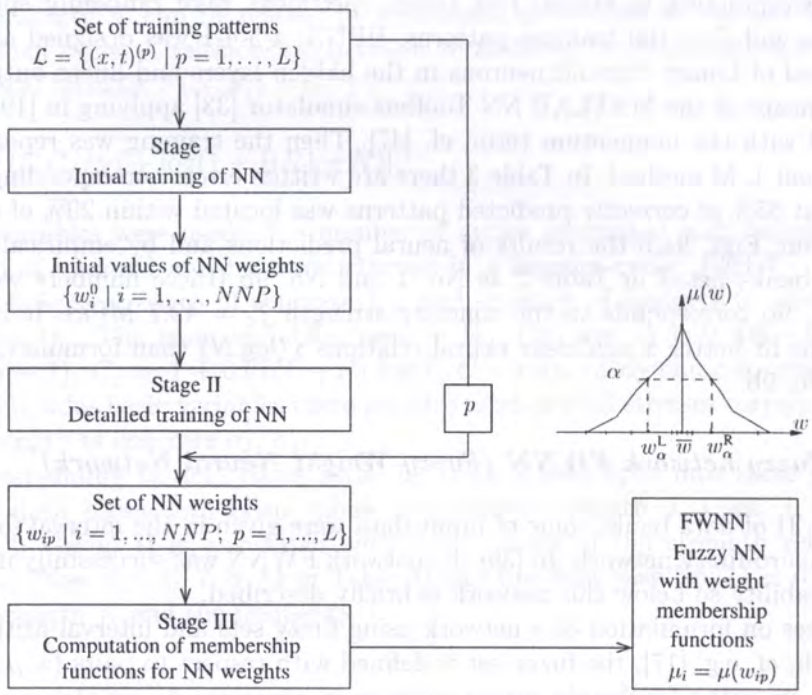


Fig. 11. Schematic algorithm of FWNN formulation

The idea of the FWNN was suggested in [34] and developed in [39]. A schematic algorithm of the FWNN formulation is presented in Fig. 11.

Let as assume that a BPNN neural network is designed using a corresponding cross-validation procedure and a subset selected from the training set of patterns (5a)

$$\mathcal{L} = \left\{ (\mathbf{x},\mathbf{t})^{(p)} \;\middle|\; p = 1,\ldots,L \right\}. \tag{30}$$

Then the network is trained on set (30) at Stage I of the algorithm shown in Fig. 11. A set of NN weights (both synaptic weights and biases) is collected as a vector of initial value weights

$$\mathbf{w}^o = \{w_i^o \mid i = 1,\ldots,NNP\}, \tag{31}$$

where: NNP – number of NN weights (parameters).

The weights w_i^o are assumed to be initial weights for learning weights corresponding to each pattern of the training set (28). At Stage II the network is trained L times for a sequence of single patterns $p = 1,\ldots,L$. After the training a matrix of weights is completed, i.e.

$$\mathbf{W} = \{\mathbf{w}_i\} = \left[w_i^{(p)} \;\middle|\; i = 1,\ldots,NNP\,;\ p = 1,\ldots,L \right]. \tag{32}$$

The membership functions for the NN weights $\mu_i = \mu(w_i)$ are computed at Stage III. Following [34] the triangular MF was assumed. Corresponding to Fig. 10b this MF is shown for the weight w (index i is omitted). The distances $3\sigma_L$ and $3\sigma_R$ are measured from the mean value $\bar{w}$, where: σ_L, σ_R – standard deviations of patterns p that are smaller or greater than $\bar{w}$, respectively. The interval values of α-cut $\left[w^L\,,\,w^R\right]_\alpha$ are depicted in Fig. 10b as w_α^L, w_α^R. In [39] a more refined MF was formulated basing on the histogram of weights $\{w_i\}$, cf. MF shown in Fig. 11.

After the membership functions are formulated for each NN weight the fuzzy network is ready for operation. The network can be used for interval values of inputs $\left[x_j^L, x_j^R\right]_\alpha$ for both fuzzy type variables, i.e. $\left(x_j^L \neq x_j^R\right)_\alpha$, and for crisp inputs, i.e. $x_j^L = x_j^R$ for each α-cut. In general the outputs

are computed as intervals $\left[y_m^L, y_m^R\right]_\alpha$ for both fuzzy and crisp inputs. They are computed by means of interval arithmetic operations for fixed α-cuts.

The predicting phase of FWNN is supported on interval arithmetic. In case of multilayer feed-forward NNs we need only the addition and multiplication operations (29).

3.2.3. Implicit modelling of concrete durability using FWNN

In Figs. 9a,b there are shown results of application of FWNN for implicit modelling of concrete durability at intervals $\left[\log N_{\text{com}}^{(p)L}, \log N_{\text{com}}^{(p)R}\right]_\alpha$ of concrete strength corresponding to the cuts $\alpha = 1.0$, 0.9 and 0.75. The results are related to selected data banks Nos. 3 and 9b corresponding the Group I of laboratory tests, cf. Table 2. The interval prediction by FWNN enables us to "cover" (predict accurately) a percent of laboratory results depending on the used α-cut. The results obtained by the network FWNN: 4-5-4-1 for the cut $\alpha = 1.0$ are slightly different from those computed by means of the standard (crisp) network BPNN: 4-5-4-1, cf. Table 3.

In case of $\alpha = 0.75$ the corresponding percentages of correctly predicted patterns are 66 and 60% for the data banks No. 3 and 9b, respectively. More extended results for other α-cut are completed in [16]. For instance if values of $\alpha = 0.5$ than the prediction was 88 and 100% with respect to the above discussed data banks.

FWNN can also be applied to the interval inputs. i.e. to the tests of the Group II. For instance in Figs. 12 there are presented results for the data banks Nos. 4 and 8a for which the concrete strength f_c was given as intervals [20.0, 30.0] MPa and [14.8, 32.7] MPa. Using $\alpha = 1.0$ the interval prediction enables us to cover 77 and 90 % of durability for laboratory test. In case of $\alpha = 0.75$ to cover 90 and 95% of laboratory results obtained for the data banks Nos. 4 and 8a, respectively, cf. [16].

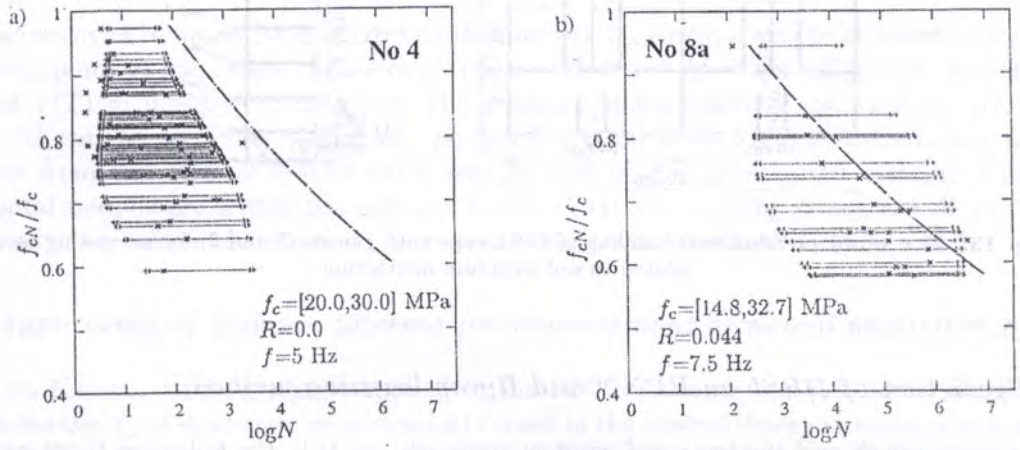


Fig. 12. Neural interval predictions (for $\alpha = 1.0$, 0.9) of relation $\chi = f_{cN}/f_c - \log N$ for data taken from data banks: a) No. 4, b) No. 8a

3.3. Simulation of response spectra for soil-structure interaction problem

3.3.1. Displacement Response Spectra and measurement data

Response spectra are often applied in structural design as well as for determining dynamic resistance of existing buildings, see e.g. [5]. The response spectrum is defined by means of the motion of the 1-DOF oscillator starting from the equation

$$\ddot{x} + 2\xi\omega_i \dot{x} + \omega_i^2 x = -a_g(t), \tag{33}$$

where: $\omega_i = 2\pi f_i = 2\pi/T_i$ – angular frequency, $T_i = 1/f_i$ – period of vibrations, ξ – damping coefficient, $a_g(t)$ – excitation corresponding to ground acceleration. Knowing a measured record of accelarations we can digitize it and use to compute displacements $x_j = x(t_j)$ for fixed values of frequencies f_i or periods of vibration T_i at a fixed damping coefficient ξ. Then the Displacement Response Spectrum (DRS) can be computed as a function which maps the natural periods of oscillators into the maximal values of their displacement response,

$$Sd(T_i) = \max_j |x(t_j\,;\, T_i\,,\, \xi)|. \tag{34}$$

In case of soil-interaction problem the DRSg is measured at the ground level and DRSb is then computed inside the structure at the basement level. It is rather a difficult task since the motion of the considered structure should be analyzed using for instance FE models. In case of real structures there are many serious questions related to modelling of boundary conditions, connections between structural elements, material relationships etc. A number of these issues can be overcome due to NNs applications. The networks trained and tested on measured DRSg and DRSb at monitored structures can be then used for predicting DRSb of other similar structures (of course, close to the monitored ones to have similar ground parameters and paraseismic excitations).

The problem considered is related to medium height (5-storey), prefabricated buildings in Legnica–Głogów Copperfield, Poland, see [26]. The buildings were subjected to paraseismic excitations caused by firings of explosives in nearby strip mines, measured at monitored buildings, see Fig. 13 taken from [25].Ten accelerograms were randomly selected from among those measured at monitored buildings. The corresponding discrete values $Sdg(T_i)$ and corresponding $Sdb(T_i)$ were computed for $i = 1, \ldots, 198$ periods of natural vibrations corresponding to $T_i \in [0.02, 1.3]$ sec.

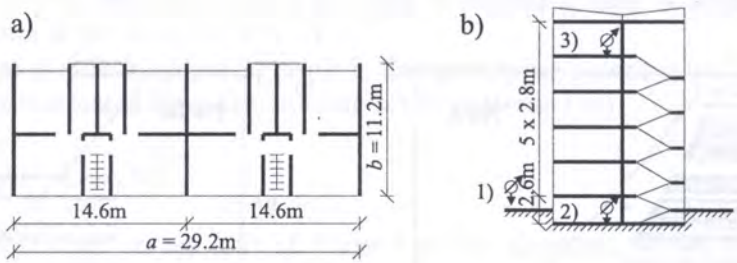


Fig. 13. Five storey prefabricated building of WBL type with gauges 1) and 2) for measuring records related to soil-structure interaction

3.3.2. Prediction of DRSb by BPNN and Rprop learning method

Using static approach and the temporal window approach, see [13], the following input and output vectors were adopted, see [26],

$$\mathbf{x}_{(6\times1)} = \{\, Sdg(i-2),\, Sdg(i-1),\, Sdg(i),\, Sdg(i+1),\, Sdg(i+2),\, T_i\},\qquad y = Sdb(i), \tag{35}$$

where: $T_{i-2}\,,\, T_{i-1}\,,\, T_i\,,\, T_{i+1}\,,\, T_{i+2}$ – successive vibration periods, $i = 3, \ldots, 196$.

The set of 10 pairs of DRS values $\{\{Sdg(i) \mid\ i = 1, \ldots, 198\}, \{Sdb(i) \mid\ i = 3, \ldots, 196\}\}$ was randomly split into an equal number of 5 training and 5 testing sets, respectively. The corresponding numbers of training and testing patterns were $L = T = P/2 = 5 \times 198 = 990$. These patterns were used for the training of BPNNs composed of sigmoid values. The Rprop learning method (12) was used in the applied SNNS simulator [53]. After an extensive cross-validation procedure the network BPNN: 6–5–1 was designed. The errors of this network training and testing are shown in Table 4, cf. [26]. All errors and statistical parameters were computed for the outputs scaled to the range $[0.1, 0.9]$.

Table 4. Errors of the training and testing processes for neural prediction of DRSb

BPNN	Learning algorithm	Number of epochs	$MSE(V) \times 10^4$		avr eV [%]		$r(T)$
			L	T	L	T	
6-5-1	Rprop	7000	13.7	12.5	13.7	13.3	0.879
2-5-1	DEKF	500	3.9	6.5	10.8	10.4	0.958

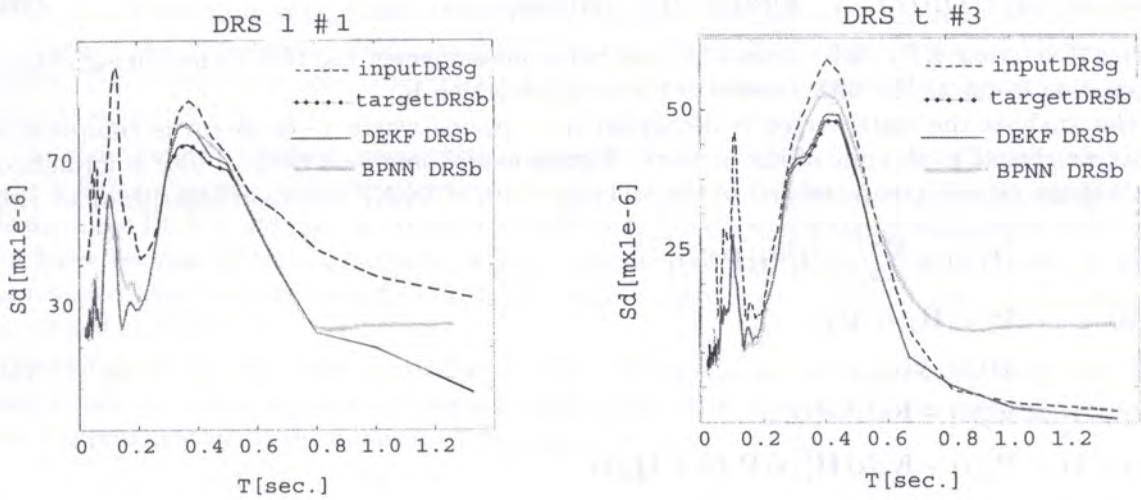


Fig. 14. Displacement Response Spectra for selected learning and testing spectra DRS $l\#1$ and DRS $t\#3$, corresponding to the measured spectra on ground level (input DRSg) and basement level (target DRSb), and computed spectra by means of Kalman filtering (DEFF DRSb) and Rprop learning method (BPNN DRSb)

The accuracy of NN prediction of DRS on the building basement seems to be satisfactory from the engineering point of view since the average relative errors avr $eL =$ avr $eT \approx 13\%$ and correlation coefficient $r(T) = 0.879$ were obtained. The graphics of the selected testing DRS are shown in Fig. 14. As can be seen the satisfactory accuracy was achieved for the fundamental periods of vibrations from the interval $[\sim 0.03, \sim 0.3]$ sec. In case of the investigated 5-storey building the fundamental periods are within the interval $[0.155, 0.294]$ sec., cf. [25], which is well placed in the range of accuracy which was obtained by means of the trained network BPNN: 6–5–1.

3.3.3. Application of Kalman filtering for improvement of neural prediction of DRSb

The neural Kalman filtering can be applied to the BPNN learning, see [13, 14]. This approach is based on the theory of stochastic processes and is used in the control theory as an approach especially suitable for the analysis of discrete dynamic systems. The approach associated with temporal neural networks was used for the approximation of records in time domains [13] but it seems to be also very prospective for NN approximations in spectral domains (as it was stated in [14, 23]). Below a very consistent description of the application of Kalman filtering in neural networks is given and next this approach is shown in the analysis of the problem of soil-structure interaction.

3.3.4. Kalman filter algorithm for NN learning

In what follows the Decoupled Extended Kalman Filter algorithm (DEKF) is used, cf. [14]. The algorithm is based on two equations: 1) process equation and 2) measurement equation:

$$\mathbf{w}_j(i+1) = \mathbf{w}_j(i) + \boldsymbol{\omega}(i)\,, \tag{36a}$$

$$\mathbf{y}(i) \quad = \mathbf{h}(\mathbf{w}(i), \mathbf{x}(i)) + \boldsymbol{\nu}(i)\,, \tag{36b}$$

where: $i = 1, 2, \ldots, I$ – discrete pseudo-time parameter, $j = 1, \ldots, n$ – the number of neuron in NN; $\mathbf{w}_j(i)$ – state vector (one-column matrix) corresponding to the set of vectors $\mathbf{w}$ of synaptic weights and biases for n neurons of NN; $\mathbf{h}$ – nonlinear input-output relation vector; $\mathbf{x}$, $\mathbf{y}$ – input/output vectors; $\boldsymbol{\omega}(i)$, $\boldsymbol{\nu}(i)$ – Gaussian process and measurement noise with mean and covariance matrices defined by

$$E[\boldsymbol{\omega}(i)] = E[\boldsymbol{\nu}(i)] = 0, \tag{37a}$$

$$E[\boldsymbol{\omega}(i)\,\boldsymbol{\omega}^T(l)] = \mathbf{Q}(i)\,\delta_{il}\,, \qquad E[\boldsymbol{\nu}(i)\,\boldsymbol{\nu}^T(l)] = \mathbf{R}(i)\,\delta_{il}\,. \tag{37b}$$

The term 'Extended' KF (EKF) is used because in the measurement Eq. (36b) a non-linear output-input relation is due to the introduction of the vector-function $\mathbf{h}$.

In the analysis the state vector is decoupled to j groups where $j-$th group is composed of synaptic weights of $j-$th node of the network. Then a model described by Eqs. (36) is formulated for each of the neuron groups related to the following form of DEKF recurrent algorithm, see [14],

$$\begin{aligned}
\mathbf{A}(i) &= \left[\mathbf{R}(i) + \sum\nolimits_{j=1}^{g} H_j^T(i)\,\mathbf{P}_j(i)\right]^{-1},\\
\mathbf{K}_j(i) &= \mathbf{P}_j(i)\,\mathbf{H}_j(i)\,\mathbf{A}(i),\\
\boldsymbol{\varepsilon}(i) &= \mathbf{y}(i) - \hat{\mathbf{y}}(i),\\
\hat{\mathbf{w}}_j(i+1) &= \hat{\mathbf{w}}_j(i) + \mathbf{K}_j(i)\,\boldsymbol{\varepsilon}(i),\\
\mathbf{P}_j(i+1) &= \mathbf{P}_j(i) - \mathbf{K}_j(i)\,\mathbf{H}_j^T(i)\,\mathbf{P}_j(i) + \mathbf{Q}_j(i),
\end{aligned} \tag{38}$$

where: $\mathbf{K}_j(i)$ – Kalman gain matrix; $\mathbf{P}_j(i)$ – approximate error covariance matrix; $\boldsymbol{\varepsilon}(i) = \mathbf{y}(i) - \hat{\mathbf{y}}(i)$ – error vector, $\mathbf{y}(i)$ - target vector for the $i-$th presentation of a training pattern; $\hat{\mathbf{w}}_j(i)$, $\hat{\mathbf{y}}_j(i)$ – i-th estimate of weight vector and output vector, $\mathbf{H}_j$ – matrix of current linearization of Eq. (36b) which takes the form

$$\mathbf{H}_j(i) = -\left.\frac{\partial \mathbf{h}_j(i, \mathbf{w})}{\partial \mathbf{w}_j}\right|_{w_j = \hat{\mathbf{w}}_j}. \tag{39}$$

3.3.5. Neural prediction of DRSb by means of DEKF Kalman algorithm

DRS discrete values were adopted as input and output variables,

$$\mathbf{x}_{(2\times 1)} = \{Sdg(i-1),\ Sdb(i-1)\}, \qquad y = Sdb(i), \tag{40}$$

where: $Sdg(i-1)$ – value of DRSg at the ground level for discrete time $i-1$; $Sdb(i-1)$, $Sdb(i)$ – values of DRSb at the basement for $i-1$ and i discrete times for $i = 2, 3, \ldots, 198$.

The autoregressive time-delay input $Sdb(i-1)$ was assumed as a variable well fitting to the character of the Kalman filtering method. Preliminary computations were performed using two types of neural networks suggested in [14], i.e. BPNN and Recurrent Layer Neural Network (RLNN). Contrary to results obtained in [22] it was stated that in the case of the problem of soil-structure interaction, analyzed in this Section, the superiority of a more refined RLNN was not proved. That is why BPNN: 2-5-1 was designed using sigmoidal neurons with bipolar sigmoid activation function in the hidden layer and linear activation function in the output neuron. The training was performed by the authors' procedures written in the MATLAB language related to the simulator MATLAB NN Toolbox [33].

The same as in Section 3.5.2 randomly selected five pairs (DRSg, DRSb) were used for the network training. Initial values of NN parameters were randomly selected from the range $[-0.5, 0.5]$ and on the basis of numerical experiments the following functions of the Gaussian noises were then found,

$$\mathbf{Q}(i) = 0.01\,\exp\left(-\frac{s-1}{50}\right)\mathbf{I}, \qquad R(i) = 7\,\exp\left(-\frac{s-1}{50}\right), \tag{41}$$

where: $\mathbf{I}$ – unit matrices of dimension (3×3) for the $j = 1, 2, \ldots, 5$ neurons of the hidden layer and (6×6) matrix for the output; s – number of the epoch in training process. The stopping criterion was established with respect to the fixed number of epochs S corresponding to the testing error $MSE(T) < \varepsilon_{\text{adm}}$ for output values scaled to the $[0.1, 0.9]$. After introductory computation the stopping criterion was related to $S = 300$ assuming $\varepsilon_{\text{adm}} = 1 \cdot 10^{-4}$.

The training of the network BPNN: 2-5-1 was performed using the algorithm DEKF as a learning method. The same training sets were used as those discussed in the previous point where the network BPNN: 6–5–1 was trained by means of the Rprop learning method. After the network BPNN: 2-5-1 was trained the mean value $Sdb(1) = 0.2577$ was established in order to start with the testing process. The training and testing errors are listed in Table 4 and corresponding graphics of selected DRS are shown in Fig. 14.

The most striking results concern the increased accuracy of neural simulation obtained by means of the Kalman algorithm DEKF vs. results of computation in which standard Rprop learning method was used, see Table 3 and Fig. 14. Another advantage of the Kalman filtering application corresponds to a lower number of training epochs. A great efficiency of the Kalman filtering learning method was resulted also from the introduction of the autoregressive input variable $Sdb(i-1)$. This effect was proved in [22].

Summing up, we can draw a conclusion that the application of Kalman filtering, well based theoretically, as a learning method seems to be a prospective approach to increase the accuracy of neural approximation in the analysis of response spectra.

4. Application of neural networks in hybrid systems FEM/NN

Development of various hybrid systems seems to be a very prospective approach in the computational structures technology. Among different parts of hybrid systems neural networks are introduced, cf. [35]. FE codes and NNs can be used as either independent or interacting parts of a hybrid system. The other approach is associated with the use of NNs as procedures in FE programs, see [48]. Both types of the hybrid systems were used in the mechanics of structures and materials. The analysis of corresponding problems is discussed in the following sections.

4.1. Hybrid Monte Carlo method in the reliability analysis of a steel girder

The assessment of structural reliability needs probabilistic analysis, see e.g. [32, 36]. The complexity of the analysis can be overcome by computer simulations and especially by those associated with Monte Carlo methods, see e.g. [28]. The simplest, classical (crude) Monte Carlo method (CMC) corresponds to generating samples which are then randomly selected and examined whether they fulfill the reliability criteria. Because of the CMC simplicity and numerical efficiency only this method is applied in this Section.

In structural engineering the samples are usually computed by the FE programs, see e.g. [44]. A great number of samples (even thousands) is needed in the MC simulation so the application of FEM programs is numerically inefficient for large scale problems because of high computational time. That is why neural networks were suggested for generating samples in the MC simulation. BPNN were used in [41] as an approximator to compute ultimate load of the structure and in [15] BPNN was applied as a classificator for evaluating the MC estimator. The idea from [41] was developed in [20]. BPNNs were incorporated in FE programs to implement MC hybrid methods. The above mentioned, hybrid MC method, was examined on examples of plane elastoplastic frames, cf. [15, 20, 41] and other, more complicated steel structures, see [18]. A corresponding example of on the reliability analysis of a steel girder, see [21], is discussed below.

4.1.1. Structural failure and MC simulations

Stationary type structural problems are analyzed by the computation of failure probability

$$p_f = \text{Prob}\{G(\mathbf{X}) \equiv R - S \leq 0\} = \int_{G(\mathbf{X})\leq 0} f(\mathbf{X})\,\mathrm{d}\mathbf{X}, \tag{42}$$

where: R – resistance of structure, S – actions (loads) applied to structure, $\mathbf{X} = [\mathbf{X}^R, \mathbf{X}^S]$ – vector of random variables. The complement p_f is structure reliability probability

$$Q = 1 - p_f. \tag{43}$$

The Monte Carlo simulation corresponds to computation of integer in Eq. (42). Following the law of large numbers the Classical Monte Carlo (CMC) estimator of the probability of failure is

$$\bar{p}_f = \frac{1}{NMC}\sum_{i=1}^{NMC} I(X_i), \qquad I(X_i) = \begin{cases} 1 & \text{for } G(X_i) \leq 0, \\ 0 & \text{for } G(X_i) > 0, \end{cases} \tag{44}$$

where NMC – the number of CMC samples.

4.1.2. Data for a steel girder and computing patterns by FEM

The analysed girder and its I-cross-section dimensions are shown in Fig. 15. The girder has only support stiffeners and it was made of steel of the yield point $R_e = 235$ MPa and stiffness modulus $E = 200$ GPa. The girder is subjected to the action of uniform load $S = P$ and the resistance of the structure corresponds to the ultimate load $R = \lambda_{\text{ult}} P^*$, where $P^* = 200$ kN/m is the reference load.

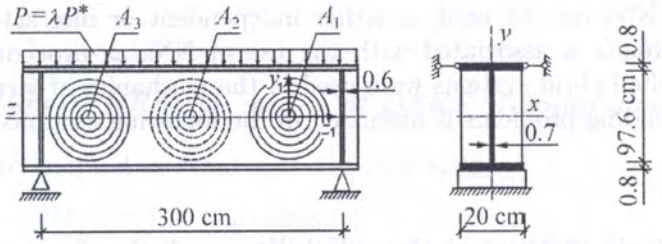


Fig. 15. Steel girder of I cross-section

Initial imperfections of the web plate are modelled as three smooth surfaces of the form taken from [24],

$$w_k(y_1, z_1) = A_k \cos\left(\frac{\pi y_1}{B_y}\right) \cos\left(\frac{\pi z_1}{L_z}\right), \tag{45}$$

where: A_k – amplitudes of imperfections; $B_y = L_z = 97.6$ cm – ranges of imperfections. It was assumed that the imperfections can randomly appear in three equidistant areas $B_y \times L_z$. The amplitudes A_k are random variables of Npdf with parameters $\mu_{Ak} = 0$ mm and $\sigma_{Ak} = A_{\text{ult}}/2 = 3.5$ mm, where $A_{\text{ult}} = 7$ mm is the admissible value.

In the considered problem the input and output vectors are assumed to be

$$\mathbf{x}_{(3\times 1)} = \{A_1, A_2, A_3\}, \qquad y = \lambda_{\text{ult}}, \tag{46}$$

where: x_i – random variables corresponding to structural or material parameters, λ_{ult} – ultimate load parameter.

Training and testing patterns were computed by the FEM system COSMOS/M [3]. A nonlinear module of the system was explored assuming an elastic-plastic material with Huber- von Mises yield surface and isotropic linear strain hardening with $E_p = 0.0001E$. All parts of the girder (web, flanges and stiffeners) were covered by regular rectangular meshes for the plate FEs SHELL4T of 24 DOF. The total number of FEs was 1616. After preliminary computations the displacement control was used assuming 60 steps $\Delta v_0 = 0.01$ mm to compute the displacement $v_0 \in [0, 60]$ mm of the web centre, measured along the y axis, see Fig. 15.

The training patterns were computed for the input data placed regularly in the 3D-cube of coordinates $A_k \in [-3\sigma_{Ak}, 3\sigma_{Ak}]$. Assuming 5 points at the A_k axes the number of training patterns equals $L = 5^3 = 125$. The set of $T = 100$ testing patterns was randomly selected as 100 points in the 3D-cube of variables A_k, assuming Npdf with the same parameters as written below Eq. (45).

In Fig. 16 one of the equilibrium paths, computed for the input data $A_1 = -0.525$ cm, $A_2 = 1.05$ cm, $A_3 = 1.05$ cm is shown. The ultimate state of the girder corresponds to the load parameter $\lambda^G = \lambda^G_{\min} = 1.180$. This state is related to the overall instability of the girder caused by the buckling of the upper flange and the web plate. In case of a perfect girder, i.e. for $A_1 = A_2 = A_3 = 0$ the ultimate load parameter is $\lambda^G_{\text{perf}} = 1.248$ and for the initial imperfections $A_1 = -1.05$ cm, $A_2 = 0.525$ cm, $A_3 = -1.05$ cm the ultimate load corresponds to $\lambda^G = \lambda^G_{\max} = 1.393$. The average CPU time to compute one pattern was about 300 sec.

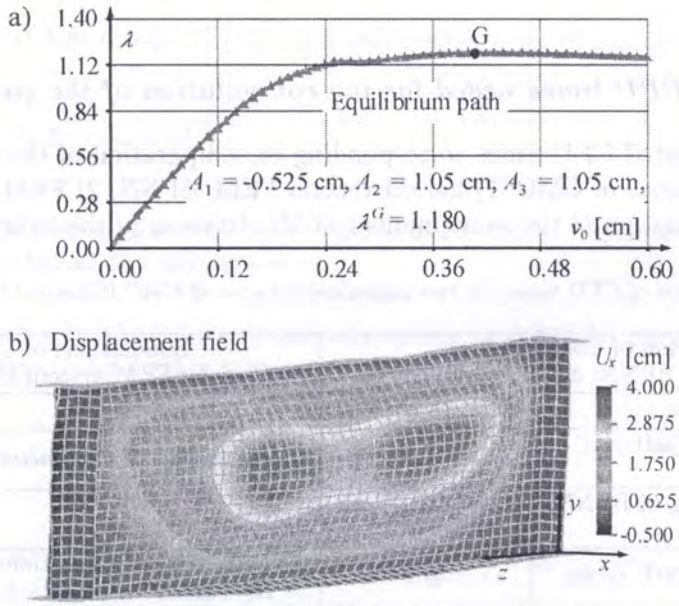


Fig. 16. a) Equilibrium path, b) Displacements of girder at load factor $\lambda^G = 1.180$

4.1.3. Using BPNN for simulation of MC trials and reliability curve for the girder

The BPNN network of structure 3-8-1 with sigmoid hidden neurons and linear output was designed using MATLAB NN Toolbox [33] and L-M learning method . In case of the trained network BPNN: 3-8-1 the errors were: avr $epL \approx$ avr $epT = 0.77\%$, max $epT = 3.90\%$, St $\varepsilon T = 0.0136$, $r_L = 0.959$, $r_T = 0.790$.

The designed network BPNN: 3-8-1 was used for the simulation of MC trials. First of all, it was checked that for computing of 10^8 MC trials the network consumed 416 sec. of CPU time. This time is comparable with 300 sec. needed for the computation of one pattern by the FEM system COSMOS/M.

Then reliability curves were computed for the girder considered. In the definition of the reliability curve there are two cases corresponding to the assumption of the action variable: 1) load P is random value with the Npdf parameters $\bar{P}_j = \mu_{Pj}$, $\bar{\sigma}_{Pj} = 0.1\bar{P}_j$, 2) P_j is a deterministic real value. In Fig. 17 there are two curves corresponding to both cases. It is worth mentioning that in Case 1 the reliability curve $Q(\bar{P}_j)$ is smooth, without discontinuity type parts which occur in Case 2 of the curve $Q(P_j)$.

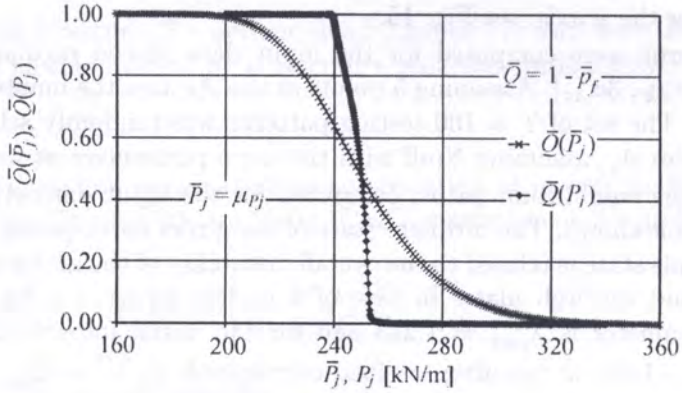


Fig. 17. Reliability curves for random loads $\bar{P}_j$ and deterministic loads P_j

4.1.4. Analysis of CPU times neded for the computation of the girder reliability curve

In Table 5 there are listed CPU times corresponding to computation of the reliability curve $\bar{Q}(\bar{P}_i)$ for two numerical versions of CMC: 1) hybrid version FEM/BPNN, 2) FEM is hypothetically used for the computer simulation of the same number of MC trials as in the hybrid version.

Table 5. Comparison of CPU times for two numerical versions of CMC (Classical Monte Carlo method)

Simulation of CMC trials by BPNN: 3-8-1		Simulations of CMC trials by FEM system COSMOS/M	
Operations	CPU time [sec.]	Operations	CPU time [sec.]
Preparation of 225 patterns by FEM 225 × 300	67500	Computation of one pattern	300
Training and testing of BPNN, about 20 hrs	72000	—	—
Simulation of 10^8 CMC trials	416	Hypothetical computations of 10^8 trials	300×10^8
Total CPU time	**1.4×10^5 sec**	**Total CPU time**	**3.0×10^{10} sec**

The computations were performed by a PC with processor AMD ATHLON XP 2.4 1.3 GHz. Application of the hybrid FEM/BPNN method needs $1.4 \cdot 10^5$ sec $\approx$ 39 hrs = 1.62 days. The hypothetical time of computing 10^8 CMC trials by the FEM system COSMOS/M equals about $3.0 \cdot 10^{10}$ sec $\approx 3.47 \cdot 10^5$ days. If we assume hypothetically that we have at our disposal a very efficient numerical version of the MC method and we need only 10^4 trials computed by COSMOS/M, then the total time would be $300 \cdot 10^4$ sec. This gives the computation 200 times longer than for the CPU time needed for the hybrid method.

4.2. Neural Networks in hybrid updating of FEM Models

Modelling of structures is a difficult task because of many uncertainties associated with material characteristics, load parameters, joints of structural elements, boundary conditions, etc. Let us focus

on FE models which are usually applied to the analysis of structures but do not give results which agree with tests on laboratory specimens or with measurements on natural scale structures (real structures).

The FE model is an example of a numerical model suitable for computer simulations. A laboratory model constructed for testing a natural scale structure can be called empirical models. Let us assume that the same excitations are applied to these models and we can measure differences between responses of numerical and empirical models. Due to selected parameters, which are introduced into the numerical model we can control its responses and try to diminish the differences between them and responses of the empirical model. This approach leads to a modification process which can be based on various direct or iterative techniques. The process of modification of the model control parameters is called structural model updating, see e.g. [7].

Below we discuss applications of NNs to updating of FE models of a simple structure, i.e. a three member plane frame using a hybrid updating approach in which BPNNs are used for the identification of control parameters basing on dynamic responses of the considered models, see [30].

4.2.1. Hybrid updating of FE models

A hybrid approach is based on the following stages, see [7, 30]:

I. Direct analysis of FE model is related to generating a set of patterns

$$\mathcal{P}' = \left\{ (\boldsymbol{\alpha}, \mathbf{r})^{(p)} \;\middle|\; p = 1, \ldots, P \right\}, \tag{47}$$

where: $\boldsymbol{\alpha}$ – vector of control parameters, $\mathbf{r}$ – vector of FE model response (e.g. eigenfrequencies) as mapping $\boldsymbol{\alpha} \to \mathbf{r}$ for all the patterns p.

II. Inverse analysis is related to the training and testing of a BPNN using a set of patterns (47) but with inverse input and output vectors

$$\mathcal{P} = \left\{ (\mathbf{x} = \mathbf{r},\, \mathbf{t} = \boldsymbol{\alpha})^{(p)} \;\middle|\; p = 1, \ldots, P \right\}. \tag{48}$$

The set $\mathcal{P}$ is split into training and testing sets $\mathcal{L}$ and $\mathcal{T}$, where $\mathcal{P} = \mathcal{L} \cup \mathcal{T}$, $\mathcal{L} \cap \mathcal{T} = \emptyset$, which are then used for designing of a BPNN.

III. Calibration of control parameters is performed by the trained BPNN exploring responses $\mathbf{r}_{\text{exp}}$ measured on an empirical model

$$\boldsymbol{\alpha}_{\text{ident}} = \mathbf{y}_{\text{BPNN}}(\mathbf{r}_{\text{exp}}). \tag{49}$$

IV. Verification of calibrated parameters by substitution of $\boldsymbol{\alpha}_{\text{ident}}$ into the FE model for computing its responses $\mathbf{r}_{\text{FEM}}$ and comparing them with measured responses $\mathbf{r}_{\text{exp}}$,

$$\mathbf{r}_{\text{FEM}}(\boldsymbol{\alpha}_{\text{ident}}) - \mathbf{r}_{\text{exp}} = \boldsymbol{\varepsilon}_{\text{upd}}\,. \tag{50}$$

When the identification error vector $\boldsymbol{\varepsilon}_{\text{upd}}$ is not admissible we should consider other control parameters which could be introduced into the considered FE model and return to Stage I.

4.2.2. Updating of a plane portal frame

From among many problems analyzed in [29] the updating of a simple frame is discussed below, cf. [30]. The laboratory model of an aluminium frame is shown in Fig. 18a and two FE models FRAME1 and FRAME2 are depicted in Figs. 18b, c. The supporting ends of the tested frame are modelled either with springs of stiffness k_1, k_2 (model FRAME1) or by means of built-in FEs with control factors α_1, α_2 (model FRAME2).

The FE models were formulated as FE systems composed of 12 plane frame FEs. The FE program was written in the MATLAB language and then first four computed eigenfrequencies were adopted as the response vectors

$$\mathbf{r}_{(4\times1)} = \{f_1, f_2, f_3, f_4\}. \tag{51}$$

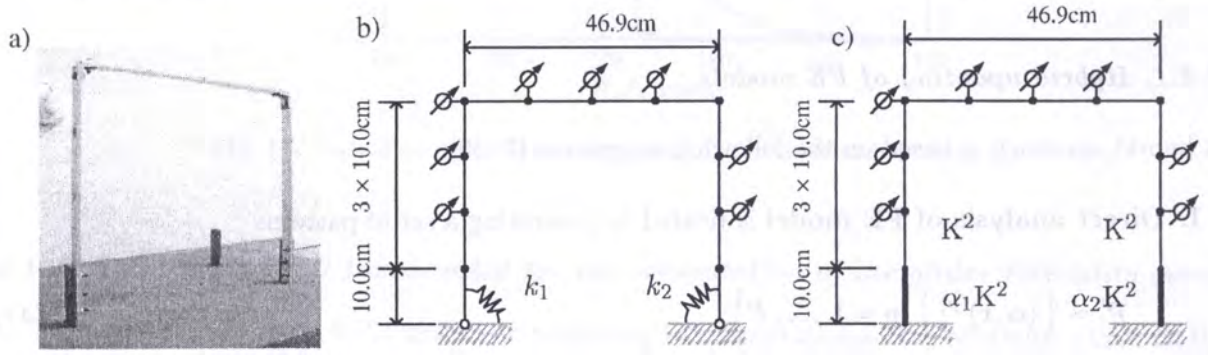


Fig. 18. a) Laboratory model of a portal frame, b) FE model FRAME1, c) FE model FRAME2

The patterns for two models of the frame were computed by the FE program and after introducing the artificial noise with standard deviation $\sigma \in [0.0001, 0.021]$ the total number of $P = 11560$ patterns was generated. The learning set P was randomly split into the training and testing sets composed of $L = T = P/2 = 5780$ patterns. The MATLAB NN Toolbox [33] and L-M learning method were used for design of two BPNNs: 4-10-2 with the input vector (51) and the following output vectors,

$$1)\ \mathbf{y} = \{k_1, k_2\}, \qquad 2)\ \mathbf{y} = \{\alpha_1, \alpha_2\}. \tag{52}$$

On the base of measured frequency response function the following experimental eigenfrequencies were deduced,

$$f_{1\,\mathrm{exp}} = 29\ \mathrm{Hz}, \qquad f_{2\,\mathrm{exp}} = 91\ \mathrm{Hz}, \qquad f_{3\,\mathrm{exp}} = 177\ \mathrm{Hz}, \qquad f_{4\,\mathrm{exp}} = 183\ \mathrm{Hz}. \tag{53}$$

Using Eq. (53) as the values of inputs the trained BPNNs the following values of control parameters were computed,

$$1)\ \text{FRAME1:} \quad k_{1\mathrm{BPNN}} = 329.7\ \mathrm{Nm/rad}, \quad k_{2\mathrm{BPNN}} = 678.4\ \mathrm{Nm/rad}, \tag{54a}$$

$$2)\ \text{FRAME2:} \quad \alpha_{1\mathrm{BPNN}} = 1.117, \quad \alpha_{2\mathrm{BPNN}} = 1.225. \tag{54b}$$

Parameters (54) were substituted to the FE program and the computed eigenfrequencies are listed in Table 6.

Looking at the results listed in Table 6 we can conclude that the accuracy of updating is better if control parameters α_1 and α_2 are used (model FRAME2). The updating errors (50) for this FE model are $|\varepsilon_{\mathrm{upd}i}(\alpha_1, \alpha_2)| \leq 4.2\%$ vs. errors for the FRAME2 model $|\varepsilon_{\mathrm{upd}i}(k_1, k_2)| \leq 8.6\%$.

Table 6. Eigenfrequencies measured on laboratory model and computed by updated FE models

Models	Identified parameters	Eigenfrequencies [Hz]			
		f_1	f_2	f_3	f_4
Laboratory	—	29	91	177	183
FRAME1	$k_1 = 329.7$ Nm/rad	26.6	93	185.6	198.8
	$k_2 = 678.4$ Nm/rad	8.3%	−2.2%	−4.9%	−8.6%
FRAME2	$\alpha_1 = 1.117$	27.9	92.8	181.1	190.6
	$\alpha_2 = 1.225$	2.8%	−2.0%	−2.3%	−4.2%

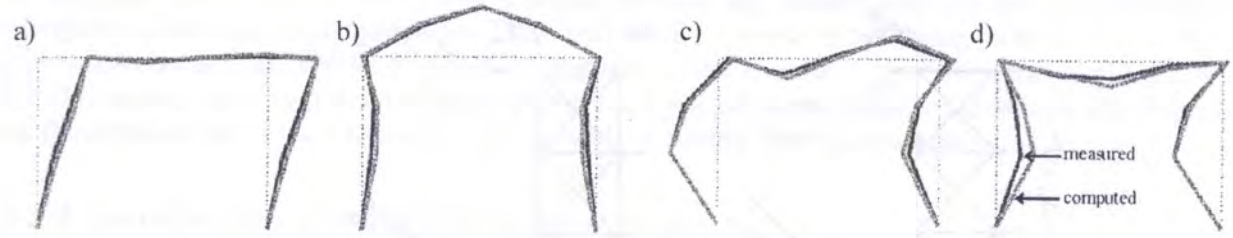


Fig. 19. Comparison of the eigenforms measured and computed by updated FE program ADINA: a) the 1st, b) 2nd, c) 3rd, d) 4th eigenforms

Four eigenforms computed by the updated FE program (values (54a) of parameters k_1, k_2 were used), are shown in Fig. 19. It is visible that the measured and updated forms are very close to each other (especially the 1st and 2nd eigenforms) and with a small loss of antisymmetry (form 3rd) and symmetry (form 4th).

In [29] other problems of updating of different FE models were considered. It is worth emphasizing that Stage II of the hybrid algorithm presented in Section 4.2.1 can be interpreted as identification of damage, corresponding to the introduced control parameters. The accuracy of identification is estimated not only by the network testing but also by responses of the updated model and their comparison with the test on a laboratory model.

4.3. BPNNs as procedures trained on-line in hybrid FEM/NN models

In many papers ANNs were used as procedures in FE programs. For instance in [50] BPNN was designed off-line as a procedure for the implicit modelling of an elastoplastic material for structures in the plane stress state. The main problem of this approach is the formulation of patterns for the network training and testing, which have to be computed as corresponding to the considered problems. The other possible approach was suggested in [10] where the network called Neural Network Constitutive Model (NNCM) was formuled and trained on-line basing on measured responses on the structure considered. The method called the autoprogressive training was then developed in [46]. This method was modified using an algorithm called in [37] as the cumulative training of NNCM.

On line learning of NNCM is based on measured responses of the considered structures. Such an approach corresponds in fact to an inverse analysis problem of the implicit modelling of an equivalent material in real solids or structures. That is why the discussed hybrid approach is worth developing since it can open the door to new nondestructive methods of the material identification.

Following [37] a short description of the autoprogressive and cumulative training of NNCM is given below, discussing these algorithms on a simple plane frame taken from [10]. Then the first results obtained in [38] concerning an improved algorithm of the autoprogressive training for the plane stresses are shown.

4.3.1. Algorithms of on-line training of NNCM

The algorithms are discussed on the example of a plane truss taken from [10] and shown in Fig. 20a. The truss is subjected to action of a single parameter load. Let us assume that the incremental FE equations are used and then the loading process is performed using the increment of the load parameter $\Delta_n^c\lambda$, where $n = 1, \ldots, NP$ – load levels, $c = 1, \ldots, NC$ – cycles of load programs to satisfy the coincidence of the measured and computed displacements, cf. Fig. 20b.

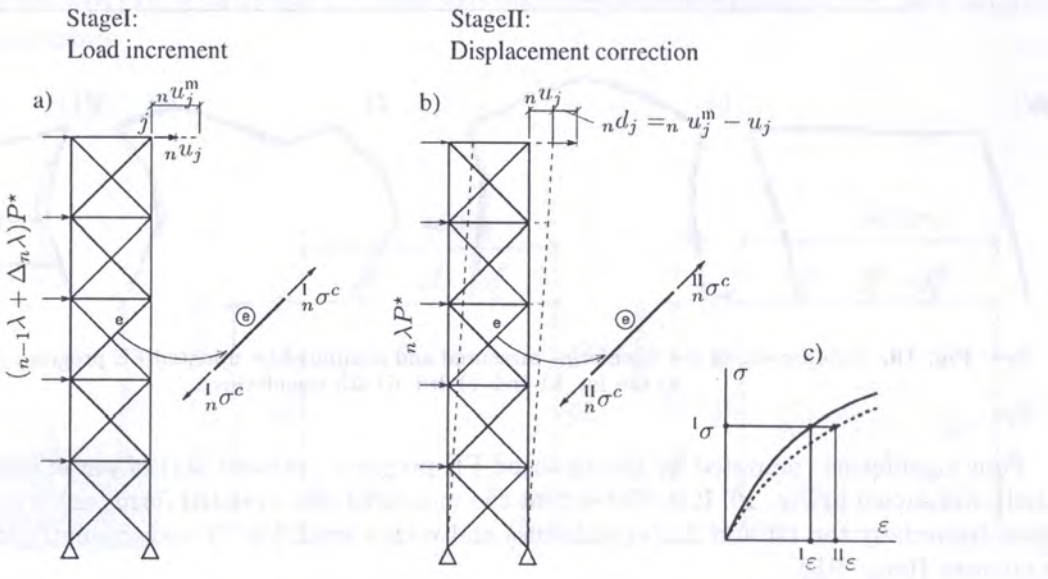


Fig. 20. Two stages of analysis for each increment of load parameter

The Newton–Raphson method is modified assuming two stages of computations at each load parameter increment:

Stage I: This stage fully corresponds to the classical N-R method. The displacements $_nu_j$ are computed for each load parameter level $_n\lambda$. The strains $_n^I\varepsilon^e$ and stresses $_n^I\sigma^e$ are computed in each truss member (e-th FE). The computations are performed for the material corresponding to the previous load level $n-1$ or for the material chacteristic fixed for the cycle c.

Stage II: This stage is performed if the displacements $_nu_j$ computed at points j are comparable with the measured displacemets $_nu_j^m$ assuming an error $\mathrm{er}\,u_{\mathrm{adm}}$, cf. Fig. 20b,

$$_nd_j \equiv {}_n|u_j^m - u_j| > \mathrm{er}\,u_{\mathrm{adm}}\,. \tag{55}$$

The same model of material is used in the Stage II. After computation of strains $_n^{II}\varepsilon^e$ and stresses $_n^{II}\sigma^e$ new "fictitious" patterns are formulated, e.g. the pairs $\{_n^{II}\varepsilon^e, {}_n^I\sigma^e\}$ are completed corresponding to a fictitious σ–ε relationship shown in Fig. 20c. These patterns are added to the training set and enable us to approach the correct σ–ε relationship.

In order to start with iteration an initial material has to be assumed, eg. elastic material with known mechanical characteristics. The next iteration steps can follow one of two algorithms discussed below:

A) Autoprogressive algorithm depends on the computation of new patterns, retraining and reformulation of NNCM at each load step n;

B) Cumulative algorithm uses the same network NNCM during all the loading in the cycle c and new patterns are generating and cumulating up to NP load increments. At the end of the loading cycle the network NNCM is retrained and reformulated.

The trained network NNCM can be used for the computation of the consistent tangential stiffness matrix of material ${}_n\mathbf{k} = \{\partial\Delta\sigma_i/\partial\Delta\varepsilon_j\,;\,\mathbf{w}\}$, where $\mathbf{w}$ is the vector of generalized weights (synaptic weights and biases of used neurons). It was stated in [37, 38] that the BPNN networks used as NNCM should have two hidden layers to assure needed numerical accuracy of derivatives $\partial\Delta\sigma_i/\partial\Delta\varepsilon_j$.

In both algorithms the main problem concerns selecting and storing patterns from the previous load steps. It should be added that in the algorithm A) an extrapolation of the training process can be very sensitive to the formulation of artificial patterns and numerically unstable because of the extrapolation of neural approximation. The algorithm B) is sensitive to storing patterns from all the previous steps and can also be numerically unstable. These questions were discussed in [10, 12, 46] but the consuming of suggestions formulated in these papers is very difficult since they also depend on the architecture of used networks and methods of tuning their parameters.

4.3.2. Identification of material in a plane frame

In order to illustrate the above algorithms the hybrid approach was used in [37] in order to identify an equivalent material of a simple plane truss taken from [10] and shown in Fig. 21a. It was assumed that the pseudo-measured results related to the horizontal displacements of the node 2 of the truss correspond to the Ramberg–Osgood material of parameters given in Fig. 21b.

The truss was analyzed by the incremental FE equations taken from [49] applying the load control using 35 steps of load parameter increments $\Delta\lambda = 0.5$. The applied network, shown in Fig. 22, is the

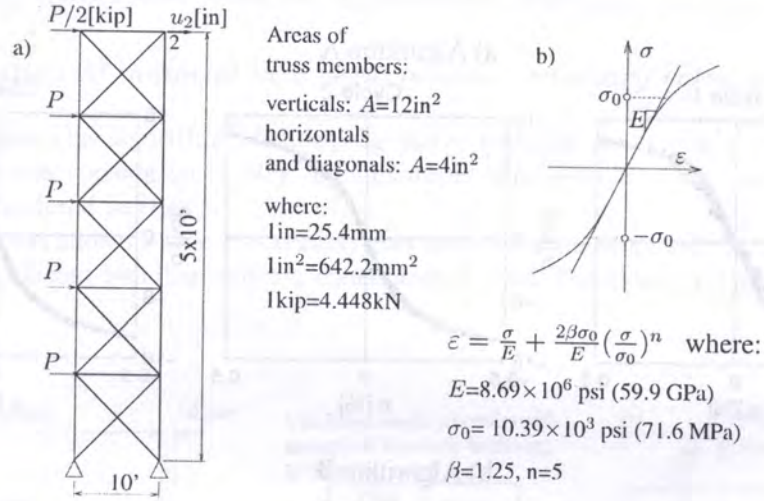


Fig. 21. Data for truss taken from [12]

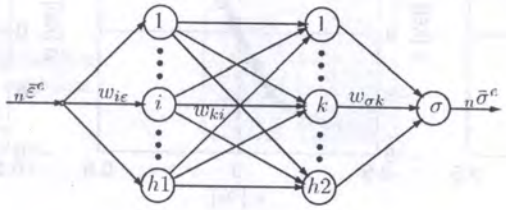


Fig. 22. Architecture of the NNCM network

BPNN network of structure 1-$h1$-$h2$-1 composed of bipolar sigmoid (hyperbolic tangens) neurons. In case of the truss in question the stiffness matrix is reduced to one element corresponding to a member of the truss,

$$
\begin{aligned}
{}_nk^e &\equiv \frac{\partial\, {}_n\Delta\varepsilon^e}{\partial\, {}_n\Delta\sigma^e} \\
&= \frac{S_\sigma}{S_\varepsilon} \sum_{k=1}^{h2} \left(\left\{ \left(1 - ({}_n\bar{\sigma}^e)^2\right) w_{\sigma k} \right\} \times \left[\sum_{i=1}^{h1} \left\{ \left(1 - ({}_nB_k)^2\right) w_{ki} \right\} \left\{ \left(1 - ({}_nA_i)^2\right) w_{i\varepsilon} \right\} \right] \right),
\end{aligned} \tag{56}
$$

where

$$
\begin{aligned}
{}_nA_i &= \mathrm{th}\left(w_{i\varepsilon}\, {}_n\bar{\varepsilon}^e + w_{i0}\right), \\
{}_nB_k &= \mathrm{th}\left(\sum\nolimits_{i=1}^{h1} w_{ki}\, {}_nA_i + w_{k0} \right), \\
{}_n\bar{\sigma}^e &= \mathrm{th}\left(\sum\nolimits_{k=1}^{h2} w_{\sigma k}\, {}_nB_k + w_{\sigma 0} \right), \\
{}_n\bar{\varepsilon}^e &= {}_n\varepsilon^e / S_\varepsilon \in (-1, 1), \\
{}_n\bar{\sigma}^e &= {}_n\sigma^e / S_\sigma \in (-1, 1), \\
S_\varepsilon\,, S_\sigma &\quad \text{– scaling parameters.}
\end{aligned} \tag{57}
$$

After introductory computations the network composed of $h1 = h2 = 4$ hidden neurons was designed using the MATLAB NN Toolbox [33] and L-M learning method.

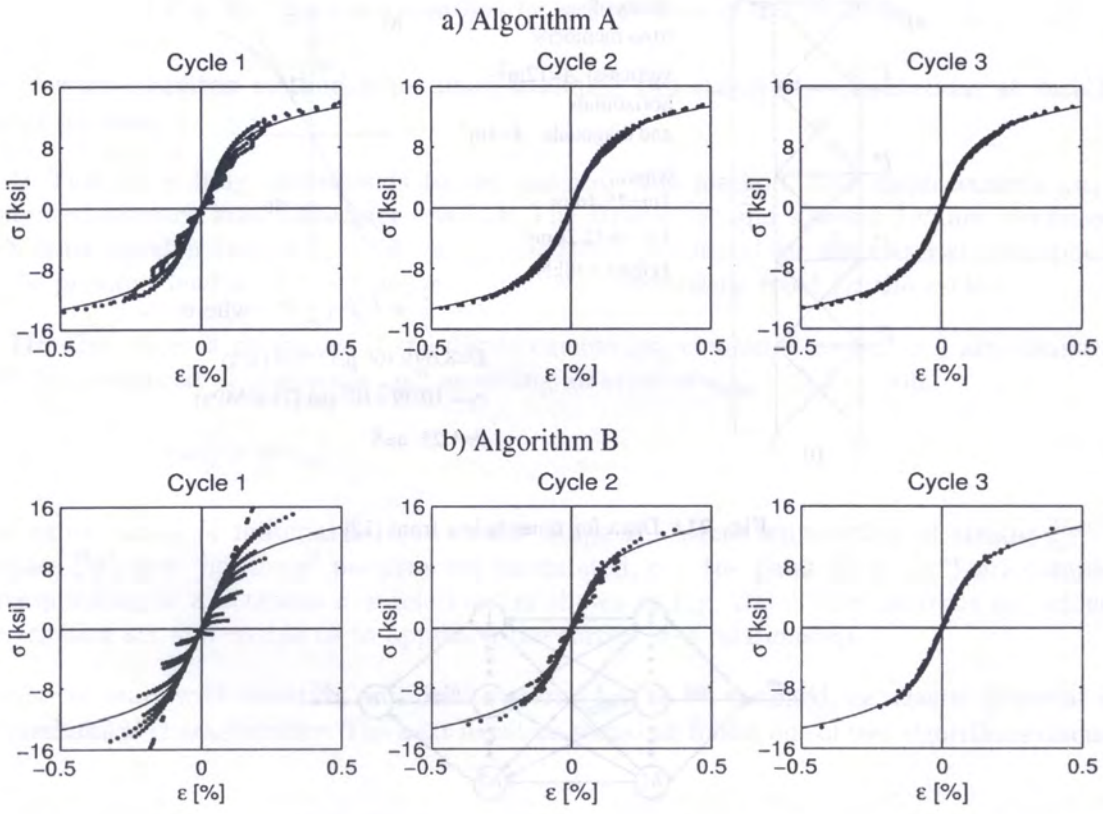


Fig. 23. Patterns generated by algorithms A and B after two first cycles of loading

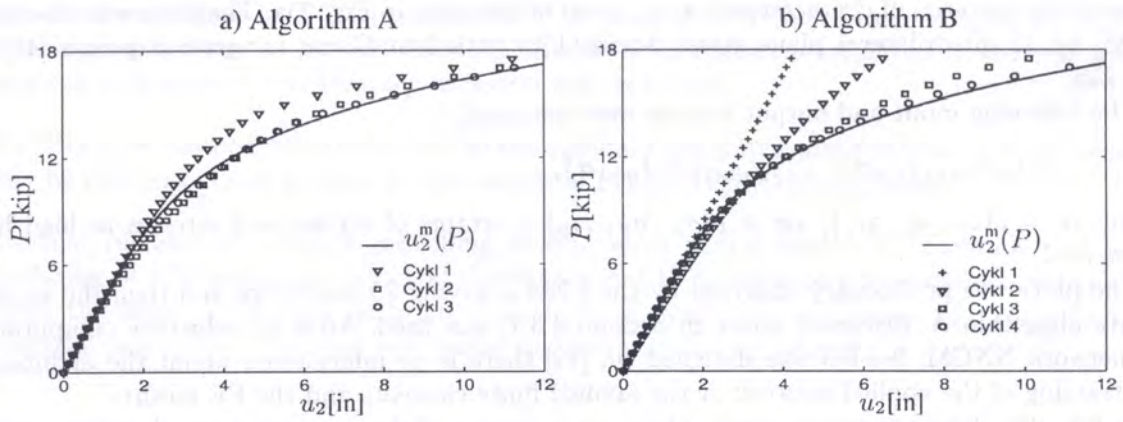


Fig. 24. Curve $u_2^m(P)$ of pseudo-measured horizontal displacements of point 2 of the truss in Fig. 21 and displacements of the same point computed by algorithms A and B for subsequent loading cycles c_i

In Fig. 23 patterns generated by the algorithms A and B are shown. The autoprogressive algorithm A gave patterns placed closer to the predicted σ–ε constitutive relationship (corresponding to the Ramberg–Osgood material) than the cumulative algorithm B in the first cycles.

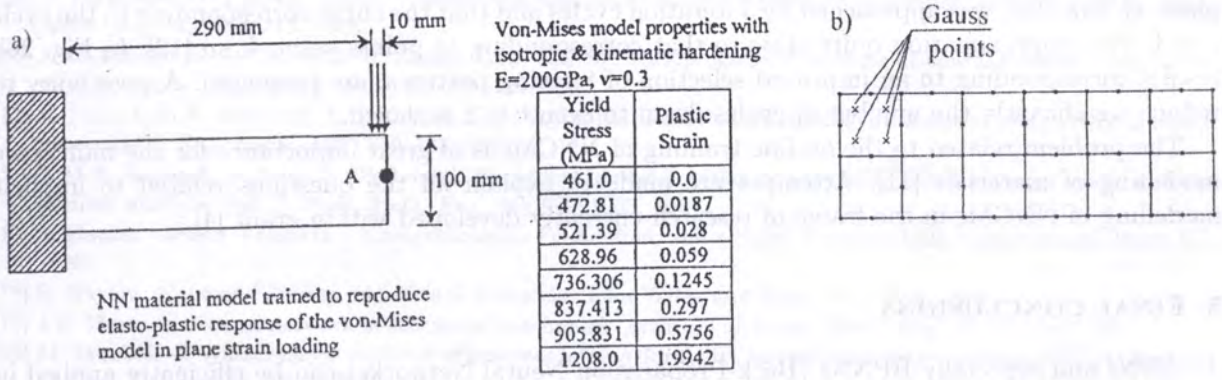
The subsequent approaching the pseudo-measured curve by iterations performed in the algorithms A and B is presented in Fig. 24. The autoprogressive algorithm A used the elastic model of material only at the initial load steps but the cumulative algorithm B explored the elastic model during the whole first cycle of loading.

It is worth adding that the storing of generated patterns and application of L-M learning method used in [37] gave a smaller network NNCM: 1-4-4-1 than the network NNCM: 1-6-6-1 designed in [10] without decreasing of the neural prediction approximation accuracy.

4.3.3. Identification of material in a beam bending boundary value problem

In order to compare the algorithm of autoprogressive training an example corresponding to the plane stress state was considered in [38]. This example corresponds to a beam bending boundary value problem considered in [12].

In Fig. 25a there is shown a thin metal cantilever plate of dimensions 300×100 mm and thickness 1 mm subjected to beam bending under a concentrated load. Following [12] the plate was made of



Yield Stress (MPa)	Plastic Strain
461.0	0.0
472.81	0.0187
521.39	0.028
628.96	0.059
736.306	0.1245
837.413	0.297
905.831	0.5756
1208.0	1.9942

Fig. 25. a) Beam bending of a steel plate and material characteristics data, b) FE mesh and nodes of the plate

elastoplastic material of characteristic σ_e–ε_e given in the table in Fig. 25a. The plate was discretized in [38] by 12 quadrilateral plane stress 8-node FEs with four Gauss integration points [49], see Fig. 25b.

The following input and output vectors were assumed,

$$\mathbf{x}_{(9\times1)} = \{{}_n\boldsymbol{\varepsilon},\ {}_{n+1}\boldsymbol{\varepsilon},\ {}_n\boldsymbol{\sigma}\}, \qquad \mathbf{y}_{(3\times1)} = \{{}_{n+1}\boldsymbol{\sigma}\}, \tag{58}$$

where: ${}_k\boldsymbol{\varepsilon} = {}_k\{\varepsilon_x\ ,\ \varepsilon_y\ ,\ \gamma_{xy}\}$, ${}_k\boldsymbol{\sigma} = {}_k\{\sigma_x\ ,\ \sigma_y\ ,\ \tau_{xy}\}$ – vectors of strains and stresses at load levels $k = n, n+1$.

The plate was preliminary analyzed by the FEM applying 25 load steps and then the autoprogressive algorithm A, discussed above in Section 4.3.1, was used. After introductory computations the network NNCM: 9-8-6-3 was designed (in [12] there is no information about the architecture and training of the applied network or the applied finite elements and the FE mesh).

In Fig. 26a there are shown results obtained by means of the autoregressive algorithm and the training was performed as in the analysis of the plane truss discussed in Section 4.3.2. In order to compute the tangent stiffness matrix at the Gauss points of the plane FEs formula (57) was generalized for the 3D stress and strain vectors following formula (18) in [12].

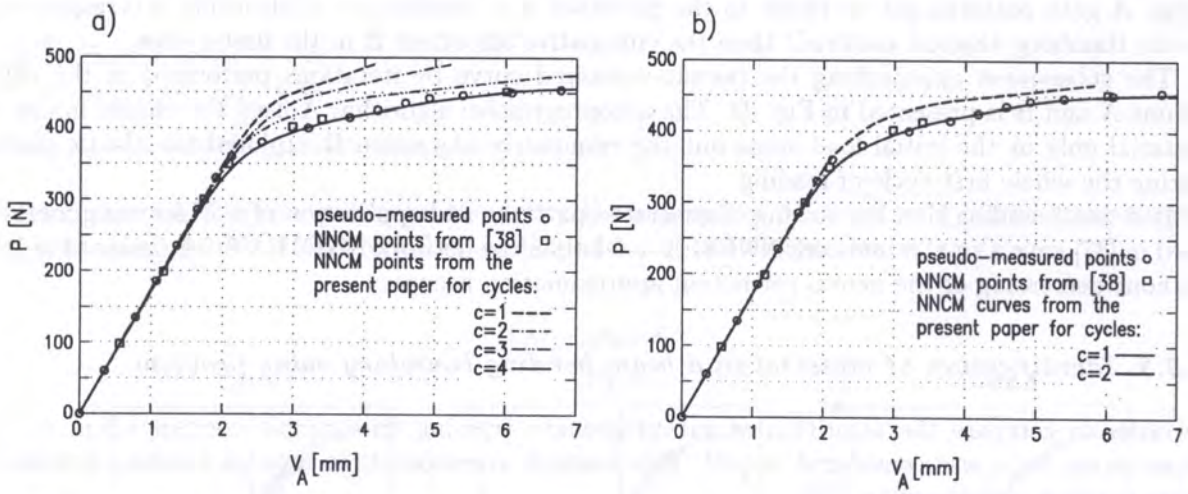


Fig. 26. Pseudo-measured points and neurally predicted equilibrium curves computed by mean of the autoprogressive algorithm A — various on-line selections of patterns and formulations of neural network constitutive models (NNCMs) leading to: a) four iteration cycles $c = 4$, b) two iteration cycles $c = 2$

As can be seen the pseudo experimental curve of measured displacements at the point A of the plate, cf. Fig. 26a, was approached by 4 iteration cycles and that the curve corresponding to the cycle $c = 4$ gives approximation quite close to that corresponding to points taken from [12]. In Fig. 26b results corresponding to an improved selection of training patterns. are presented. A possibility to reduce significantly the number of cycles down to even $c = 2$ is shown.

The problem related to the on-line training of NNCMs is of great importance for the multiscale modelling of materials [11]. Attempts are made to explain all the questions related to implicit modelling of NNCMs in the frame of research currently developed within grant [4].

5. Final conclusions

1. ANNs and especially BPNNs (Back-Propagation Neural Networks) can be efficiently applied in the inverse analysis corresponding to parametric identification and implicit modelling of physical relations associated with many problems of mechanics of structures and materials.

2. BPNNs can be used as a new independent tool for the analysis of problems mentioned above.

3. Modifications based on BPNNs (neuro-fuzzy network FWNN, i.e. Fuzzy Weight NN, Kalman filtering applied for the network learning) enable us to extend possibilities to perform interval analysis and increase the accuracy of neural approximation.

4. BPNNs have complementary features to the standard computational methods so these networks can be efficiently used as parts or procedures of hybrid FEM/ANNs systems.

5. Current research on implicit modelling of equivalent neural models of materials and hybrid updating of FE models of structural elements seems to be especially interesting since they can open the door to new non-destructive methods of evaluation of damage and degradation of material properties in existing real solids and natural scale structures.

AKCNOWLEDGMENTS

Financial supports of the Polish Ministry of Scientific Research and Information Technology, grants: No. 4 T07E 065 26 *"Assessment and Active Monitoring of the Technical State of Building Structures"* and No. 4 T07E 060 29 *"Coupling of FEM and ANNs in the Analysis of Solids with Micro Structure and Structures of Unknown Constitutive Equations"* are gratefully acknowledged.

REFERENCES

[1] *ADINA, Theory and Modelling Guide.* Adina R&D Inc., 1992.

[2] A. Borowiec, L. Ziemiański. Identification of stiffness reduction in beams using parameter dependent frequency changes and neural networks. In: K.J. Bathe, ed., *Computational Fluid and Solid Mechanics 2005, Proc. 3rd MIT Conf., June 14–17, 2005*, 97–100. Elsevier, Amsterdam, 2005.

[3] *COSMOS/M, Finite element analysis system, Version 2.5.* Structural Research and Analysis Corp., Los Angeles, California, 1999.

[4] *Coupling of FEM and ANNs in the Analysis of Solids with Micro Structure and Structures of Unknown Constitutive Equations*, Polish Ministry of Scientific Research and Information Technology, Grant No. 4 T07E 060 29, Institute of Computer Meth. in Civil Eng., Cracow, Poland, 2005–2007.

[5] *Eurocode 8. Design Provisions for Earthquake Resistance of Structures*, European Prestandard. European Committee for Standardization, Brussels, 1994.

[6] D.J. Ewins. *Modal Testing: Theory, Practice and Applications.* 2nd Ed. Research Studies Press, Baldock, Hertfordshire, 2000.

[7] M.I. Friswell, J.E. Motterhead. *Finite Element Model Updating in Structural Mechanics.* Kluwer Academic Publishers, Dordrecht, 1996.

[8] K. Furtak. Strength of concrete subjected to multiple repeat loadings (in Polish). *Archives of Civil Engineering*, **30**: 677–698, 1984.

[9] T. Furukawa, G. Yagawa. Implicit constitutive modelling for viscoplasticity using neural networks. *Int. J. Num. Meth. Eng.*, **43**: 195–219, 1998.

[10] J. Ghaboussi, D.A. Pecknold, M. Zhang, R.M. Haj-Ali. Autoprogressive training of neural network constitutive models. *Int. J. Num. Meth. Eng.*, **42**: 105–126, 1998.

[11] R. Haj-Ali, D.A. Pecknold, J. Ghaboussi, G.Z. Voyiadjis. Simulated micromechanical model using artificial neural networks. *J. Eng. Mech.*, **127**: 730–738, 2001.

[12] Y.M.A. Hashash, S. Jung, J. Ghaboussi. Numerical implementation of a neurally based material model in finite element analysis. *Int. J. Num. Meth. Eng.*, **59**: 989–1005, 2004.

[13] S. Haykin. *Neural Networks – Comprehensible Foundation*, 2nd edition. Prentice-Hall. Upper Saddle River NJ, 1999.

[14] S. Haykin. *Kalman Filtering and Neural Networks.* John Wiley and Sons, New York, 2001.

[15] J.E. Hurtado. Neural network in stochastic mechanics. *Archives of Comp. Meth. Eng.*, **8**: 303–342, 2001.

[16] M. Jakubek, Z. Waszczyszyn. Analysis of concrete fatigue durability by the neuro-fuzzy network FWNN. *Archives of Civil Engineering*, **52**, 2006 (in print).

[17] J.-Sh.R. Jang, Ch-T. Sun., E. Mizutani. *Neuro-Fuzzy and Soft Computing.* Prentice-Hall, Upper Saddle River NJ, 1997.

[18] J. Kaliszuk. *Reliability Analysis of Structures and Structural Elements by Artificial Neural Networks* (in Polish). Ph.D. Thesis. Dept. of Civil and Environmental Eng., Univ. of Zielona Góra, Poland, 2005.

[19] J. Kaliszuk, A. Urbańska, Z. Waszczyszyn, K. Furtak. Neural analysis of concrete fatigue durability on the basis of experimental evidence. *Archives of Civil Engineering*, **47**: 327–339, 2001.

[20] J. Kaliszuk, Z. Waszczyszyn. Reliability analysis of structures by neural network supported Monte Carlo methods. In: L. Rutkowski, J. Kacprzyk, eds., *Neural Networks and Soft Computing. Proc. 6th Int. Conf., Zakopane*, 754–759. Springer, Berlin–Heidelberg, Germany, 2003.

[21] J. Kaliszuk, Z. Waszczyszyn. Reliability analysis of a steel girder by the hybrid Monte Carlo method. *Proc. XI Intern. Conf. on Metal Structures ICMS-2006*. A.A. Balkema, Leiden, The Netherlanads, 2006 (in print).

[22] A. Krok, Z. Waszczyszyn. Neural prediction of response spectra from mining tremors using reccurent layered networks and Kalman filtering. In: K.J. Bathe, ed., *Computational Fluid and Solid Mechanics 2005, Proc. 3rd MIT Conf., June 14–17, 2005*, 302–305. Elsevier, Amsterdam, 2005.

[23] A. Krok, Z. Waszczyszyn. Application of neural networks and Kalman filtering to simulation of displacement response spectra on buildings subjected to paraseismic excitations. In: Z. Waszczyszyn, M. Słoński, eds, *Proc. Int. Symposium on Neural Networks and Soft Computing in Structural Engineering NNSC-2005, ECCOMAS Thematic Conferences, Cracow, Poland, June 30–July 2, 2005*, CD-ROM, 7 pages. Cracow Univ. of Technology, Cracow, 2005.

[24] M. Królak. *Postcritical behaviour and load carrying capacity of thin-walled girders* (in Polish). PWN, Warszawa-Łódź, Poland, 1990.

[25] K. Kuźniar. *Analysis of Vibrations of Medium-Height Buildings with Load Bearing Walls Subjected to Mining Tremors Using Neural Networks* (in Polish). Monograph 310, Series of Civil Eng., Cracow Univ. of Technology, Cracow, Poland, 2004.

[26] K. Kuźniar, E. Maciąg. Neural analysis of soil-structure interaction in case pf mining tremors. In: D. Doolin, A. Kammerer, T. Nogami, R.B. Seed, I. Towhata, eds., *11th Intern. Conf. on Soil Dynamics and Earthquake Engineering* and *3rd Intern. Conf. on Earthquake Geotechnical Engineering*, Vol. 2: 829–83. Berkeley, USA, 2004.

[27] W. Łakota. *Detection and Location of Damage in Beam Structures* (in Polish). Ofic. Wyd. Polit. Reszowskiej, Rzeszów, 1999.

[28] P. Marek, J. Brozetti, M. Guštar. *Probabilistic Assessment of Structures using Monte Carlo Simulation: Background, Exercises and Software*. Institute of Theoretical and Applied Mechanics, Academy of Sci. of the Czech Republic, Praha, 2001.

[29] B. Miller. *Updating of Mathematical Model of Structure to Physical Model* (in Polish). Ph.D. Thesis. Dept. of Building and Environmental Eng., Rzeszów Univ. of Technology, Rzeszów, Poland, 2001.

[30] B. Miller, L. Ziemiański: Neural networks in updating of dynamic models with experimental verification. In: L. Rutkowski, J. Kacprzyk, eds., *Neural Networks and Soft Computing. Proc. 6th Int. Conf., Zakopane*, 766–771. Springer, Berlin–Heidelberg, Germany, 2003.

[31] Z. Mróz, K. Dems. Identification of damage in beam and plate strucutres using parameter depending modal changes and thermographic methods. In: Z. Mróz, G. Stavroulakis, eds., *Parameter Identification of Materials and Structures*, CISM Courses and Lectures No. 469, 95–137. Springer, Wien–New York, 2005.

[32] J. Murzewski. *Reliability of Engineering Structures* (in Polish). Arkady, Warsaw, Poland, 1989.

[33] *Neural Network Toolbox for Use with MATLAB. User's Guide Version 3*. The MathWorks Inc., Natick, MA, 1998.

[34] S.H. Ni, P.C. Lu, C.H. Yang. A fuzzy neural network approach to evaluation of slope failure potential. *Microcomputers in Civil Engineering*, **11**: 59–66, 1996.

[35] A.K. Noor. Computational structures technology: leap frogging into the twenty-first century. *Computers and Structures*, **73**: 1–31, 1999.

[36] A.S. Nowak, K.R. Collins. *Reliability of structures*. McGraw-Hill, 2000.

[37] E. Pabisek. Hybrid FEM/ANN identification of a model of equivalent material basing on measurements of structure displacements (in Polish). *Proc. 50th Polish Civil Eng. Conf., Krynica, Poland, September 12–17, 2005*, Vol. 2: 81–88. Gdańsk, 2005.

[38] E. Pabisek. On-line training of neural network based constitutive models (in Polish). Report on current research in grant [4], paper in preparation for publishing. Cracow Univ. of Technology, 2005.

[39] E. Pabisek, M. Jakubek, Z. Waszczyszyn. A fuzzy network for the analysis of experimental structural engineering problems. In: L. Rutkowski, J. Kacprzyk, eds., *Neural Networks and Soft Computing. Proc. 6th Int. Conf., Zakopane*, 771–777. Springer, Berlin–Heidelberg, Germany, 2003.

[40] Th.L. Paez. Neural networks in mechanical system simulation, identification and assessment. *Shock and Vibration*, **1**: 177–199, 1993.

[41] M. Papadrakakis, V. Papadopoulos, N.D. Lagaros. Structural reliability analysis of elastic-plastic structures using neural networks and Monte Carlo simulation. *Comp. Meth. Appl. Mech. Eng.*, **136**: 145–163, 1996.

[42] G. Piątkowski. *Detection of Damage in Structural Elements Using Artificial Neural Networks* (in Polish). Ph.D. Thesis. Dept. of Building and Environmental Eng., Rzeszów Univ. of Technology, Rzeszów, Poland, 2003.

[43] G. Piątkowski, L. Ziemiański, The solution of an inverse problem in plates by means of artificial neural networks. In: L. Rutkowski *et al.*, eds., *Lecture Notes on Artificial Intelligence*, LNAI 3070, *Artificial Intelligence and Soft Computing – ICAISC 2004, 7th Intern. Conf., Zakopane, Poland, June 2004*, 1081–1086. Springer-Verlag, Berlin–Heidelberg, 2004.

[44] J.E. Pulido, T.L. Jacobs, E.C. Prates de Lima. Structural reliability using Monte Carlo simulation with variance reduction techniques on elastic-plastic structures. *Computers and Structures*, **43**: 419–430, 1992.

[45] P. Rojas. *Neural Networks – A Systematic Introduction*. Springer Verlag, Berlin–Heidelberg, 1996.

[46] H.S. Shin, G.N. Pande. On self-learning finite element codes based on monitored response of structures. *Computers and Geotechniques*, **27**: 161–178, 2000.

[47] Z. Waszczyszyn, ed. *Neural Networks in the Analysis and Design of Structures*. CISM Courses and Lectures No. 404, Springer, Wien–New York, 1999.

[48] Z. Waszczyszyn. Neurocomputing and finite element method. In: T. Burczyński, P. Fedeliński, E. Majchrzak, eds., *Computer Methods in Mechanics, Proc. 1st CEACM Conf. and 15th Intern. Conf. CMM-2003*, Gliwice/Wisła, Poland, June 3–6, 2003. CD-ROM, 10 pages. Silesian Univ. of Technoligy, Gliwice, 2003.

[49] Z. Waszczyszyn, Cz. Cichoń, M. Radwańska. *Stability of Structures by Finite Element Methods*. Elsevier, Amsterdam, 1999.

[50] Z. Waszczyszyn, E. Pabisek. Hybrid NN/FEM analysis of the elastoplastic plane stress problem. *Computer Assisted Mechanics and Engineering Sciences*, **6**: 177–188, 1999.

[51] Z. Waszczyszyn, L. Ziemiański. Neural networks in mechanics of structures and materials – new results and prospects of applications. *Computers and Structures*, **79**: 2261–76, 2003.

[52] Z. Waszczyszyn, L. Ziemiański. Neural networks in the identification analysis of structural mechanics problems. In: Z. Mróz and G. Stavroulakis, eds., *Parameter Identification of Materials and Structures*, CISM Courses and Lectures No. 469, 265–340. Springer, Wien–New York, 2005.

[53] E. Zelle. *SNNS – Stuttgart Neural Network Simulator. User's Manual, Version 4.1*. Univ. Stuttgart, 1995.