Computer Assisted Mechanics and Engineering Sciences, **18**: 275–282, 2011.

An improved Neural Kalman Filtering Algorithm in the analysis of cyclic behaviour of concrete specimens

Agnieszka Krok
Cracow University of Technology
Institute of Computer Methods in Civil Engineering
Warszawska 24, 31-155 Cracov, Poland
e-mail: agakrok@poczta.fm

The article is related to the results of research on Node Decoupled Extended Kalman Filtering (NDEKF) as a learning method for the training of Multilayer Perceptron (MPL). Developments of this method made by the author are presented. The application of NDEKF and MPL and other methods (pruning of MLP, Gauss Process model calibrated by Genetic Algorithm and Bayesian learning methods) are discussed on the problem of hysteresis loop simulations for tests of compressed concrete specimens subjected to cyclic loading.

Keywords: Artificial Neural Networks (ANN), Kalman Filter (KF), Node Decoupled Extended Kalman Filtering (NDEKF), Multilayer Perceptron (MPL), Genetic Algorithm (AG), Bayesian methods, concrete specimens, cyclic loading, hysteresis loops.

1. Introduction

Kalman filtering is a mathematical method suitable for simulation of test measured values by their prediction, estimating their uncertainty and computing a weighted average of the predicted and measured values. The aim of the research presented in the paper is to simulate noisy measurements observed over time and predict values which are supposed to be close to such measurements, see [3]. The Kalman filter approach was adopted in ANN nonlinear models as a new learning technique, see [2]. Selected node learning and pruning of ANN are discussed, see [9]. Statistically grounded ANN learning methods, related to the Gaussian Process model and Bayesian Methods, see [8], are also applied.

2. Kalman filtering as an ANN learning method

The basic KF learning method, supported on Decoupled Extended Kalman Filtering (NDEKF), is based on two equations, called (1) process equation and (2) measurement equation. These equations are modified into a form which can be adopted to learn standard Multilayer Perceptron (MLP), see [2]:

$$\mathbf{w}_{k+1}^{i} = \mathbf{w}_{k}^{i} + \omega_{k}^{i}, \tag{1}$$

$$\mathbf{y}_{k} = \mathbf{h}(\mathbf{w}_{k}, \mathbf{x}_{k}) + \nu_{k}, \tag{2}$$

where: k – discrete pseudo-time parameter; i – neuron index in ANN; w_{k+1}^{i} for $i = 1, 2, ..., W$ – state vector components corresponding to the set of synaptic weights and biases; $\mathbf{h}$ – vector-function

marking a non-linear input-output relation $\mathbf{x}/\mathbf{y}$, ω_k^i, ν_k – Gaussian process and measurement noises with mean and covariance matrices defined by:

$$\mathbf{E}(\nu_k) = \mathbf{E}(\omega_k^i) = 0, \tag{3}$$

$$\mathbf{E}(\omega_k^i \omega_{l^T}^i) = \mathbf{Q}_k^i \delta_{kl}, \tag{4}$$

$$\mathbf{E}(\nu_k \nu_l^T) = \mathbf{R}_k \delta_{kl}. \tag{5}$$

The NDEKF algorithm induces decoupling state vector into groups. The decoupling level related to neurons (nodes $i = 1, 2, ..., N$) is performed. The change of $\mathbf{w}^i$ during the presentation of the k-th learning pattern takes the following form:

$$\mathbf{K}_k^i = \mathbf{P}_k \mathbf{H}_k \left[\sum_{j=1}^{g} (\mathbf{H}^j)_k^T \mathbf{P}_k^j \mathbf{H}_k^j + \mathbf{R}_k \right]^{-1}, \tag{6}$$

$$\widehat{\mathbf{w}}_{k+1}^i = \widehat{\mathbf{w}}_k^i + \mathbf{K}_k^i \xi_k, \tag{7}$$

$$\mathbf{P}_{k+1}^i = (\mathbf{I} - \mathbf{K}_k^i (\mathbf{H}_k^i)^T) \mathbf{P}_k^i + \mathbf{Q}_k^i, \tag{8}$$

where: $\mathbf{K}_k^i$ – Kalman gain matrix; $\mathbf{P}_k^i$ – approximate error covariance matrix; g – the number of FLNN nodes (groups of decoupling); $\xi_k = \mathbf{y}_k - \widehat{\mathbf{y}}_k$ – error vector, with the target vector $\mathbf{y}_k$ for the kth presentation of a training pattern; $\widehat{\mathbf{y}}_k$ – output vector computed by MPL, $\mathbf{H}$ – matrix of current linearization of Eq. (2):

$$\mathbf{H}_k^i = \frac{\partial \mathbf{h}}{\partial \mathbf{w}^i}. \tag{9}$$

The analyzed parameters corresponding to the Gaussian noise were adopted in the following form:

$$\mathbf{Q}_k^i = 0.001 \exp((s-1)/50)\mathbf{I}, \tag{10}$$

$$\mathbf{R}_k = 7 * \exp((s-1)/50)\mathbf{I}, \tag{11}$$

where $\mathbf{I}$ – identity matrix whose dimension depends on the state vector dimension in MLP, s – the number of learning epoch.

2.1. Development of KF algorithm

It was numerically proved that the KF algorithm is time consuming in comparison with the traditional Resilient Propagation or Levenberg-Marquardt learning algorithm implemented in the Neural Networks Toolbox for use with MATLAB, see [1, 12]. In order to reduce the computational time some changes were introduced into the original NDEKF. Implementation of separate models for ANN nodes leads to different levels of estimation due to the diversity of initially chosen values and the non symmetric input vector. Higher values of the approximate error covariance matrix $\mathbf{P}$ for a particular node gave better filtering algorithm performance for this node:

$$P_{k+1}^i \approx E(w^i - \widehat{w}_{k+}^i)(w^i - \widehat{w}_{k+1}^i)^T, \tag{12}$$

where $\widehat{w}_{k+1}^i$ – the estimator for the w^i, found after presenting $k+1$-th learning pattern. W_i is the dimension of P^i matrix. The following values are at the main diagonal of this matrix:

$$P_{k+1}^i(m) \approx E(w^i(m) - \widehat{w}_{k+1}^i(m))(w^i(m) - \widehat{w}_{k+1}^i(m))^T \tag{13}$$

for $m = 1, 2, .., W_i$, where W_i is an approximated mean square error of the estimation process for the i-th neuron of the network.

There is a significant difference of performance of the filtering algorithm if the ANN is learned up to an admissible MSE error level. The proposed algorithm is based on a dynamic change of the region of the network that is learned. After the initial learning of the whole network, nodes for which the filtering performance is weaker can be selected. Then they are and learned until the level of filtering for theses nodes reaches the average level of the whole network. During this procedure the rest of neurons are frozen. The procedure is repeated. The level of filtering performance is estimated for each of the network layers separately. The proposed algorithm has the following form:

1. Initial learning during $s = 1, 2, .., S_0$ epochs, then for the each of the network layers separately, computing the average estimation errors in the layer:

   $$M = \text{mean}_{\{m=1,..,n_i, i=1,..,I(1)\}} P^i_{k+1}(m).$$

2. Computation the average estimation errors in the nodes:

   $$M(i) = \text{mean}_{\{m=1,..,W_i\}} P^i_{k+1}(m)$$

   for $i = 1, 2, ..I(1)$.

3. Selection the neurons for learning, i.e. finding i such that $M(i) < \alpha M$ where α – the fixed parameter.

4. Learning only the selected up to S_{step} epochs. For the rest of neurons values of weights were unchanged.

5. Computing M, $M(i)$.

6. Repeating the above steps until a stop criterion is reached.

2.1.1. Pruning of MLP learned by KF algorithm

In order to prevent applying the MLP with too many connections, the number of neurons should be reduced, which can shorten computational time. From among different possibilities of automatic improving of ANN architecture the pruning method was applied, cf. [1]. At the beginning, a large ANN is learned and during the learning process some of the connections are cut. Thus the network changes its architecture into a smaller one. The connections are removed after checking their poor influence onto the input-output relations or decreasing of the MSE error. The basis of the proposed algorithm is the 'lprune' algorithm changed into the KF learning, cf. [11]. Let us adopt the vector of weights parameters (synaptic weights and biases of ANN):

$$\mathbf{w} = \{w^j\}_{j=1}^W \in \mathbf{R}^W. \tag{14}$$

For each weight w^j the value of the statistics Λ as calculated:

$$\Lambda(w^j) = \ln \left(\frac{\left| \sum_{p=1}^L w^j - \eta \dfrac{\partial E(p)}{\partial w^j} \right|}{\eta \sqrt{\sum_{p=1}^L \left(\dfrac{\partial E(p)}{\partial w^j} - \text{mean}_{\{p=1,2,..,L\}} \dfrac{\partial E(p)}{\partial w^j} \right)^2}} \right), \tag{15}$$

where L – number of patterns in the learning set of data, η – the percentage parameter, $\text{mean}_{\{p=1,2,..,L\}}$ – arithmetic mean in the learning set, $E(p)$ – error for L learning patters given by the known formula:

$$E(p) = (h(x^p, \mathbf{w}) - y^p)^2, \tag{16}$$

where (x^p, y^p) – target input/output pair for the p-th pattern, h – marking of ANN.

The large values of Λ for the particular weight means that this weight has a significant influence on the learning process. The weight with the smallest values of Λ are removed.

The following algorithm was proposed:

1. Choosing initial architecture.

2. Learning initial architecture for S epochs in order to find the number S_{start} epochs after which the ANN is pruned the first time.

3. In order to make possible changes of weights after cutting of the connection in the KF, a model of noise matrixes is applied for the number of learning epochs $S_{reset} <= S_{start}$.

4. Choosing the time between the consecutive cuts. After the number of the current epoch is divided by selected parameter k, the pruning is made.

5. Choosing the threshold Λ_{prog}. The weights for which values of Λ are smaller then Λ_{prog} are removed. The threshold is changed my means of the following product:

$$\Lambda_{prog} = \alpha mean(\Lambda), \tag{17}$$

where α – percentage element, $mean(\Lambda)$ – average arithmetical value of $\Lambda(w^j)$ for $j = 1, 2, ..., W$.

6. Removed connections equal zero for the learning process.

3. Simulation and prediction of concrete hysteresis loops by means of modified KL learning

The main results obtained by the method presented above and its extension were discussed in [4–7]. The investigated problems were analyzed by means of the neural simulation applied to prediction of data sets corresponding to mining tremors, concrete, steel and superconductor hysteresis loops as well as other data time series analysis, see [7]. Now let us consider the numerical results obtained for the simulation related to computer simulations of concrete hysteresis loops. Some graphics for the accuracy comparison is shown to deduce some general conclusions.

3.1. Experimental data

Tests on 12 concrete cylindrical compressed samples 3x6 [in.] were discussed in [12] with respect to the following cyclic loading programs:

1. Monotonic of the load increase to the maximal value.

2. Decrease of the load to zero value.

3. Subsequent repetition of the load programs (cycles) 1 and 2, see Fig. 1.

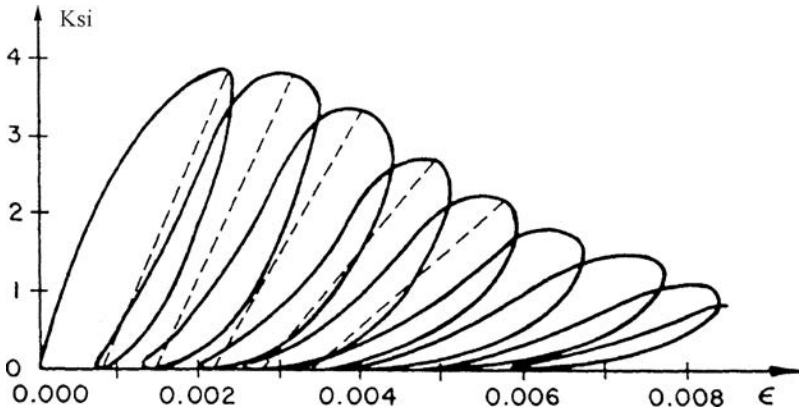

Fig. 1. Experimental data.

3.2. Application of MLP with KF learning

Data for the MLP learning and testing, i.e. the neural simulation and prediction, were adopted as discrete points at the stress-strain dimensionless relations $\overline{\sigma} - \overline{\varepsilon} \in [0, 0.9]$. The measurement data set was randomly split into the learning and testing sets. The testing set was composed of points corresponding to three last hysteresis loops. The properly learned ANN was able to simulate the behavior of the material both in the first and the final parts of the hysteresis loops. Numerical experiments showed that sequential components of input vectors could be written with respect to the following process:

- Progress of the experiment (tests on concrete samples) in time k/P for *marker* $k = 1, 2, ..., P$, where P – the number of data points according to the experiment scheme.

- Progress of the experiment in time for each loop is analyzed by means of the marker. It is a parameter of numbering patterns for the network learning and testing inside each loop. The numbering is made for each loop independent of other loops, normalizing the intervals to the range [0,1]:

$$marker = [1/N_1, .., N_1/N_1, ..., 1/N_i, .., N_i/N_i, ..., 1/N_9, .., N_9/N_9], \tag{18}$$

  where N_i – the number of data points in i-th hysteresis loop.

- For the control of load increase and decrease the $marker_1$ was adopted. Inside i-th hysteresis loop the following parameters turned out to be the most numerically efficient:

$$marker_{1,i} = [1/M_i, 2/M_i, .., M_i/M_i, (M_i - 1)/M_i, (M_i - 2)/M_i, ..., (M_i - N_i)/M_i], \tag{19}$$

  where M_i – the number of experimental points for which the material is loaded, N_i – the number of experimental points whose material is unloaded inside the i-th hysteresis loop. $marker_{1,i}$ was scaled to the interval [0,0.9].

- value of σ predicted by ANN in the previous step computation was written as: $\sigma_{\mathrm{ANN}}(k-1)$.

The network MLP: 3-6-3-1 was applied. The output of neural network, predicted by the network in the current step of computation, was written in figures as *Ksi* or σ, corresponding to the pressure-dependent stress σ.

The learning set for this experiment consist of first six hysteresis loops, which gave $L = 273$ data points. The testing set were selected as the following three loops, which gives $T = 132$ data points.

From among many input vectors the most efficient was input vector, composed in the form:

$$\mathbf{x}(k) = [\sigma(ssn)(k-1), k/(273+132), marker_1], \tag{20}$$

where k – the number of current pattern, $k = 1, .., 405$, see, [11].

The improved curves $\sigma - \varepsilon$ for the MLP learning and testing by means of Kalman filtering are shown in Fig. 2.

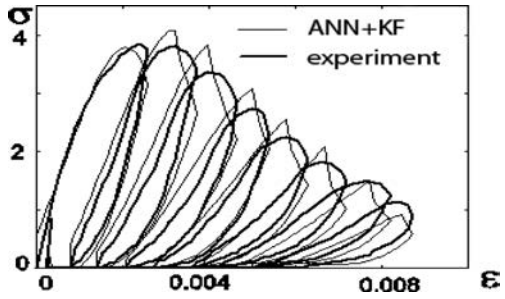

Fig. 2. ANN and KF learning method for 1000 learning epochs.

4. Bayesian approach to simulation of hysteresis loops

4.1. Gaussian process modelling calibrated by Genetic Algorithm

Let us consider the stochastic process Y, generated by the set of fixed basis functions with random weights:

$$Y(X) = \sum_{j=1}^{M} W_j \phi_j(X), \tag{21}$$

where X – input random variable, [3], fulfilling the following relation: if $W_j \sim N(0, \Sigma)$ then $E_W[Y(X)] = 0$ and $E_W[Y(X)Y'(X)] = \phi^T(X)\Sigma\phi(X)$.

Two covariance functions were considered, see [11], i.e. squared exponential and rational quadratic functions. The non-linear optimizer is used to find the maximum likelihood values of the parameters. The starting parameters for the algorithm were found by Genetic Algorithm (GA) for the fitness function $MSE(\overline{k}, m, \sigma)$. The results of simulation are shown in Fig. 3 with parameters obtained by the genetic algorithm and maximum number of iterations of 35.

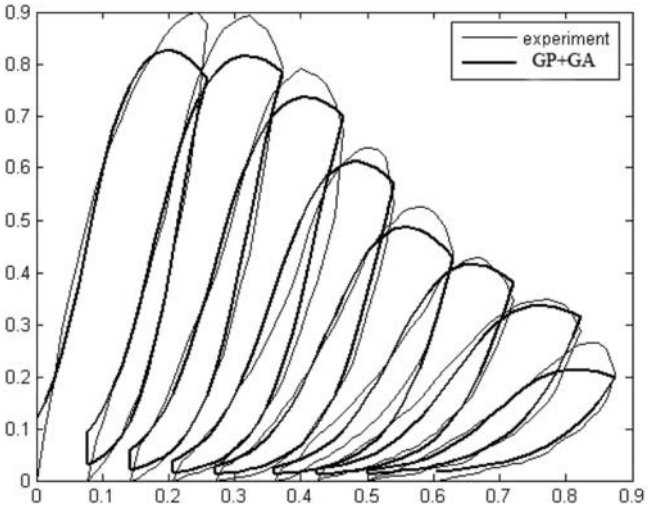

Fig. 3. Gaussian process modelling for squared exponential function.

4.2. Bayesian learning method

Next, the stochastic approach was applied. MLP weights were adopted as random variables with the known 'a priori' Gaussian distribution. They were adopted according to Bayes' updating scheme in the model, in which 'a priori' distribution of the weights is $p(w)\,N(0,\sigma) = p(w|D)\;N(0,\sigma)$.

The posterior distribution of the w represents knowledge about the values given data D was presented to the network. They were computed via the application of the Bayes' theorem: $p(w|d) = p(D|w)p(w)/p(D)$ where $p(D|w)$ – data set likelihood, $p(D) = \int p(D|w)p(w)dw$ – normalization constant. The evidence procedure was taken from [9] as an iterative algorithm for determining optimal weights and hyperparameters during Bayesian learning of MLP. The results of Bayesian learning and testing of MLP are shown in Fig. 4. The number of training cycles in the inner loop was 500, the number of inner loops 2 and the number of outer loops 2.

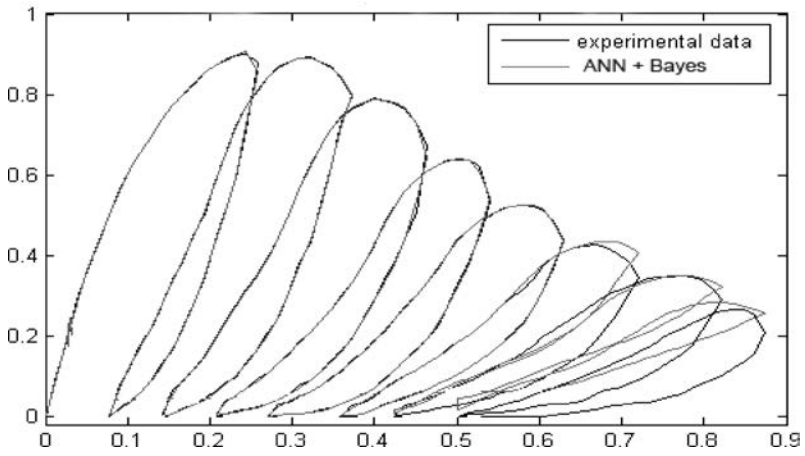

Fig. 4. ANN and Bayesian learning method.

In Table 1 the learning and testing errors MSEL and MSET obtained by the numerical models are listed for MLP with applied Bayesian methods (written in Table 2 as MLP+Bayes). As shown in Table 1, the approximation MLP+Bayes gives the best results for the discussed problem of the hysteresis loops simulation.

Table 1. Comparison of errors obtained by different numerical models.

Method	MSEL	MSET
MLP+KF	0.00092	0.0005
GP+GA	0.00004	0.0002
MPL+Bayes	0.00001	0.00009

5. Conclusions

1. The numerical analysis proved that the Node Decoupled Extended Kalman Filtering (NDEKF) can be a learning method for the training of Multilayer Perceptron (MPL), however it is numerically costly. The author improved the learning method of NDEKF applying pruning and dynamic changes in MLP structure in order to decrease the number of operations.

2. The proposed algorithms seem to be numerically efficient. The application of KF leads to the development of other statistically based methods, i.e. Bayesian learning and Gaussian Process model.

3. Comparison of the applied method shows that this methodology can be efficient in simulation of concrete subjected to cyclic loading.

4. From the viewpoint of numerical efficiency, the Gaussian Process modelling calibrated by Genetic Algorithm seems to be one of the favourable methods.

5. MLP gives the best estimations of hysteresis loops simulation, see Table 1. But this method and the Gaussian Process model and the MLP with Bayesian methods need an additional effort with respect to the implementation process. That is why for the simplicity of implementation, the KF learning can also be recommended.

References

[1] S. Haykin. *Neural Networks*, A Comprehensive Foundation, 2nd Ed. MacMillan College Publ., Engle-wood Cliffs, NJ, 1999.
[2] S. Haykin. [Ed.], *Kalman Filtering and Neural Networks*, John Wiley & Sons, New York, 2001.
[3] R.E. Kalman. A new approach to linear filtering and prediction problems. Transactions of ASME, *Journal of Basic Engineering*, **82(D)**: 35–45, 1960.
[4] A. Krok, Z. Waszczyszyn. Neural prediction of response spectra from mining tremors using recurrent layered networks and Kalman filtering. In: J-K.Bathe [Ed.] *Proc. 3rd MIT Conf. Computational Fluid and Solid Mechanics*, pp. 302–305. Elsevier, 2005.
[5] A. Krok, Z. Waszczyszyn. Simulation of building loops for a superconductor using neural networks with Kalman filtering. *Computer Assisted Mechanics and Engineering Sciences*, **13**: 575–582, 2006.
[6] A.Krok, Z. Waszczyszyn. Kalman filtering for neural prediction of response spectra from mining tremors. *Computers and Structures*, **85**(15–16): 1257–1263, 2007.
[7] A. Krok. *Analysis of Mechanics of Structures and Material Problems Applying Artificial Neural Networks Learnt by Means of Kalman Fltering* (in Polish), Ph.D. Thesis, Institute of Computer Methods in Civil Engineering. Cracow Univ. of Technology, 2007.
[8] D. MacKay. *Information Theory, Inference, and Learning Algorithms*. Cambridge Univ. Press, 2003.
[9] I.T. Nabney. Netlab, Algorithms for pattern recognition. Springer, 2006.
[10] L. Prechelt. *Adaptive Parameter Pruning in Neural Networks*. International Computer Science Institute University of California Technical Report TR-95-009, 1995.
[11] B.P. Sinha, K. H. Gerstle, L. G. Tulin, Stress-strain relations for concrete under cyclic loading. *Journal of American Concrete Inst.*, **61**(12): 1964.
[12] *Neural Network Toolbox for Use with MATLAB, User's Guide*, Version 4. The MathWorks, Inc., Natick, MA, 2000.