Neural Technologies in Electrical Drive Control

V.B.Klepikov, K.V.Mahotilo, S.A.Sergeev, G.K.Voronovsky

Kharkov State Polytechnic University
21 Frunze St., Kharkov, 310002, Ukraine
E-mail: klepikov@polyt.kharkov.ua

Introduction. Main properties of neural networks

When surveying the genesis of artificial neural networks (ANN) as a scientific branch, one is bound to start with paper [1]. Then he will certainly mention Rosenblatt's perceptron [2], which used to be a great hope of cyberneticists and is unfairly forgotten now. The revival of interest in ANN in the eighties is connected with such names as J.Hopfield, who suggested a network of original cyclic (full-connected) structure [3], and D.Rumelhart, who rediscovered a network training algorithm known as the Backpropagation Algorithm [4].

What is ANN? In the simplest case, it constitutes an assembly of threshold elements arranged in layers (Fig. 1), similar to a biological neuron net. Elements of different layers are linked in such a way that every neuron of the next layer receives signals from all the neurons of the previous layer. Synapse function (synapse is the place where the axon of a previous-layer neuron joins a dendrite of a next-layer neuron) is simulated by changing synaptic weights, which enhance or attenuate the transmitted signal. Input signals x_i received by neuron dendrites (Fig. 2.) are superposed $y = \sum_{i=1}^{n} w_i x_i + w_0$, where w_0 is bias and w_i is the ith synaptic weight. The resulting excitation is converted into output signal in the axon corresponding to a chosen activation function $z = F(y)$ (threshold or sigmoidal functions, or radial basis function (RBF) [5], see Fig. 3). Thus, ANN input layer, or receptor, receives activation x and transmits the excitation z to hidden layers. Network reaction u is released from the output layer, or effector.

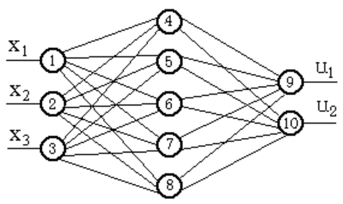


Figure 1 Scheme of a three-layer ANN

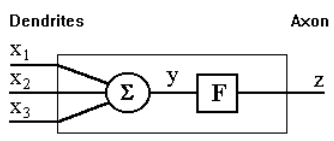


Figure 2 Model of a neuron

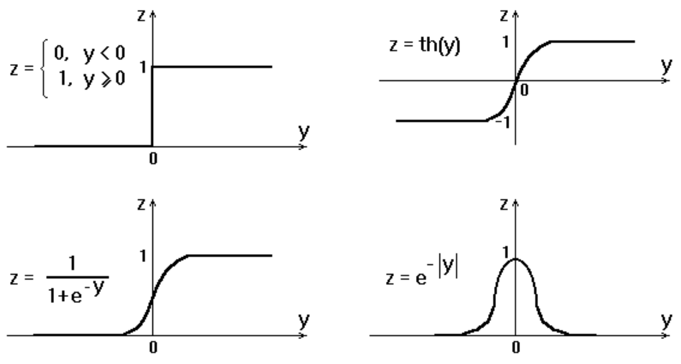


Figure 3 Different kinds of a neuron activation function

In neural network operation there are two different modes: training and examining. Training is considered as a process of changing neuron operation laws (what is obtained by varying synoptic weights and thresholds of the neurons themselves), repeated successively until the network is trained to be capable of mapping a certain set of inputs (images) into a set of desired outputs. To estimate the quality of a network operation one usually uses a sum of squared errors over all the outputs:

$$e = \tfrac{1}{2}\sum\left(d_i - u_i\right)^2, \qquad (1)$$

where d_i and u_i are, respectively, a desired and real values of the ith output. If one succeeds in training the network so that the value e does not exceed a certain threshold simultaneously for all the patterns, the training process is supposed to be over. After that, the network parameters get fixed and the network itself is ready to work.

Due to training, the network gets feasible to distinguish not only the images learned during the training, but also any other input data from the allowed space, classifying them according to the feature set. In this meaning ANN is said to be capable of generalizing the typical features of images produced.

1. Neural network technologies in Control

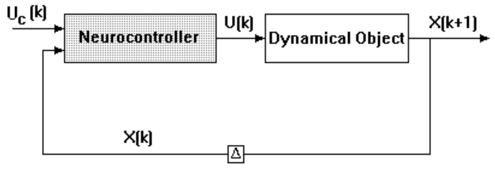
Papers [6] and [7] were basic for implementing ANN in controlling. The first of them stated a neural network to be a universal approximator and the other exhibited the feasibility of designing an ANN-based neurocontroller.

Now we can regard ANN as a recognized paradigm in controlling which has proved its efficiency and gained plenty of fans in scientific societies.

To resume, ANN can be used in control systems as:

- neurocontrollers, which represent nonlinear multiparameter PID controllers;
- neuroemulators, which simulate the total dynamical behaviour of a control object or describe its certain unmodelled characteristics (such as friction effects, etc.);
- adaptive filters.

Figure 4 shows a scheme of a neuroconroller implementation in object controlling under commands $U_c(k)$, and figure 5 gives a scheme for training neuroemulator.



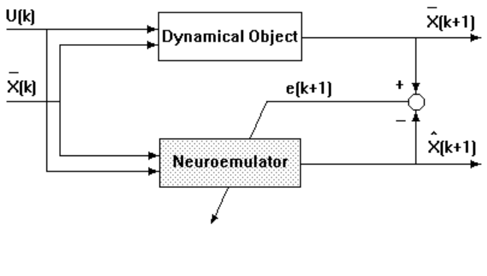


Figure 4 Object controlling by means of a neurocontroller

Figure 5 Neuroemulator training from object input-output data

Training a neurocontroller itself directly from object input-output data may be performed by means of the schemes given in figures 6 and 7. The training objective is to make it capable of producing a control sequence U(k), which transfers the object from an arbitrary initial state X(0) to a given final state X(k) within a certain number of steps Ê.

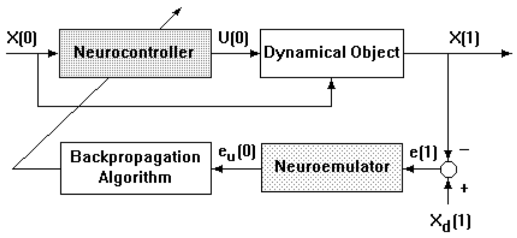


Figure 6 Neurocontroller training by means of the backpropagation technique

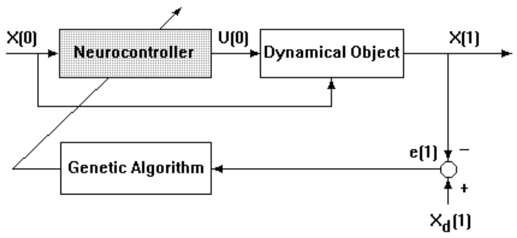


Figure 7 Optimization of neurocontroller parameters by means of genetic algorithm

The Backpropagation Algorithm is quite fit for this purpose, but it requires some information about the error in the ANN output layer. To have this information, the scheme of figure 6 provides for an object inverse neuroemulator to pick out equivalent error $e_u(0)$, which corresponds to the object input error. In other words, this neuroemulator is to convert the object output error to the neurocontroller output. After the neurocontroller being trained once, the entire procedure is repeated again and again for arbitrarily chosen initial states until the network parameters tally the satisfactory values for any X(0).

One of the shortcomings of the Backpropagation Algorithm results from its being a local optimizing procedure in essence. At the same time, there is no doubt that error (1) used to estimate the ANN quality is a multiple extremum function of the network parameters, hence the search of its minimum requires a global method.

From this point of view, it becomes clear why more and more attention has lately been paid to applying genetic algorithms (GAs) into ANN training. GA is an efficient method of global optimization which copies natural mechanisms of genetic information recombination what guarantees adaptation changes within a population. The combination of these two computational techniques (ANN and GA) is regarded nowadays as a potential source of further progress in evolutionary modelling. The detailed survey of the ideas in the field of cross fertilization between GAs and ANN can be found in [8].

Besides its inherent globality, GA as a training procedure has the advantage over the backpropagation algorithm that it is capable of training a neurocontroller just from the object output behaviour, as shown in figure 7. Unfortunately, on-line operation of such a scheme is rather problematic because any adaptation process undergoes through number of generations what takes certain time.

2. Adaptive control system

Figure 8 shows the conceptual design of an ANN-based control system with a training unit implementing GA for optimal network parameters search which is being worked out.

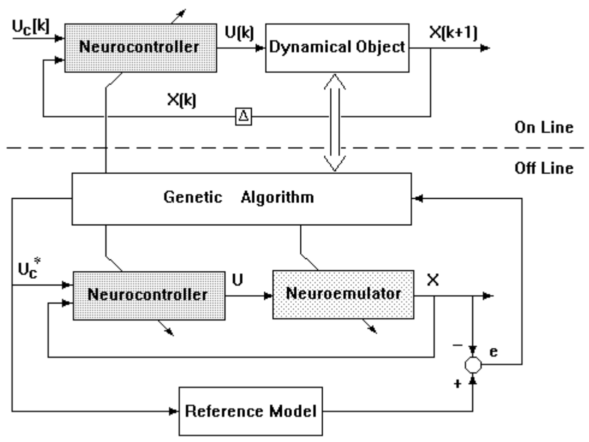


Figure 8 Conceptual design of a control system with a GA-based training unit

Unlike the scheme implementing the backpropagation technique, in our scheme the neurocontroller adaptation is performed through predicting control loop processes by means of numerical experiments with the control object neuroemulator rather than through analysing the transient process in a discrete step of the control algorithm. Synthesis procedure for such a system comprises two stages. First, a neuroemulator is synthesized on the basis of known object characteristics, and then this neuroemulator is applied to train the neurocontroller. While operating the system, the training unit not only makes the neurocontroller adapt, but also traces the co-ordination between the neuroemulator and the control object, and if necessary, accomplishes the correction.

Let's illustrate the prediction unit synthesis procedure for a control system with an example from electromechanics.

3. Electrical drive control

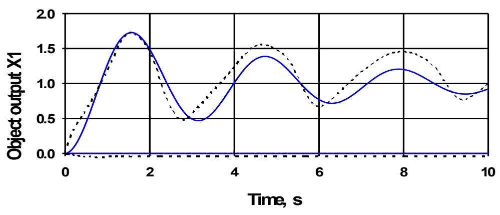
A DC motor as a control object is known to be modelled as an inertial link of the second order with the transmission function in form of

$$\Phi = \frac{k}{T^2 s^2 + 2T\varsigma s + 1} \tag{2}$$

According to the above-mentioned procedure, first we synthesized a DC motor neuroemulator, using for its training the solutions of a linear differential equation system relevant to transmission function (2)

$$\begin{cases} \dot{x}_1 = x_2 \\ \dot{x}_2 = \left(-2T\varsigma x_2 - x_1 + kU\right) / \sqrt{T} \end{cases} \tag{3}$$

where x_1 and x_2 are, respectively, rotor angular speed and armature current. Parameter values $k = 1$, $T = 0{,}5$ and $\varsigma = 0{,}1$ were chosen so that control object oscillation properties were seen very clearly. The control object transmission function for the mentioned parameter values is given in fig. 9.



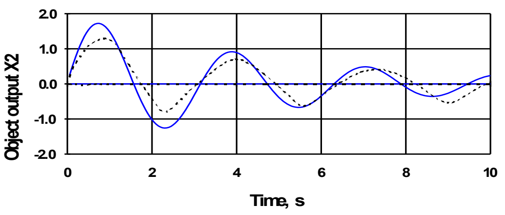


Figure 9 Control object transmission function for outputs x_1 and x_2: solid lines are calculated with (3); dashed lines are emulated by the ANN

While synthesizing the neuroemulator which approximates the right parts of system (3), we made the following assumptions. To be certain, we chose a three-layer network with three nodes in the input layer (input signal U, emulator output signal x_1 in the previous step and its differential x_2), eight nodes in the hidden layer and two nodes in the output layer (first and second differentials $\dot{x}_1$ and $\dot{x}_2$ of the output signal). The network parameter vector constitutes 50 components.

RBFs were chosen as activation functions.

$$z = \exp(-|y|) \tag{4}$$

GA-based search procedure comprised the following steps. First, by means of a random number generator, we created a diploid population of binary vectors [9], each vector coding the neuroemulator parameters. Supposing five-bit coding of every network parameter to assure the proper accuracy, we got genotypes of all the network versions in terms of triplets of 250-bit chromosomes.

To estimate the fitness of every version to imitate the object response to an excitation, after decoding the parameters to base 10 numbers, we transmitted two signals $U = \sigma(t)$ and $U = 0$ to the neuroemulator input.Then the output signals were compared with the relevant solutions of differential equations (3). Mismatch between desired transient processes in the ANN output and the facts was estimated by integrating errors for each of the two signals within time period from t = 0 to t = 10 s and their subsequent summing. Depending on how successfully the ANN emulated the transient process, each version took a certain position in the population during its ranking. Then, guided by individual crossing laws which are common in GA framework [10], we got offspring, estimated their fitness, rearranged the population, and so on until the approximation accuracy met our requirements. Figure 9 also exhibits the results of the network response emulation for the 5,000th generation

The second step was synthesis of the **adaptive PID-neurocontroller capable of compensating control object inertiality and suppressing its oscillation**. The neurocontroller structure constituted three nodes in the input layer, four nodes in the hidden layer and one output node. The input neurons received the difference E between input signal U_c and object output x_1, its integral and differential, and the network output produced control signal U. In this step, each of the 26 network parameters was coded to a 8-bit base 2 number, thus, the chromosomes which coded the neurocontroller represented 208-bit binary sequences.

In general, synthesis procedure for the controller repeated that for the emulator. The initial population was created in the same way, every version fitness was estimated, ranking the population, and crossing the fittest individuals, etc. were performed. What was different is that we used the neuroemulator rather than the mathematical model in form of equations (3) to predict the transient processes in the control loop (Fig. 10). Also, it is important that each testing was conducted three times: for a unit positive, a unit negative and zero signals. The total error over all the signals was chosen as the measure of the version imperfection and used in the population ranking.

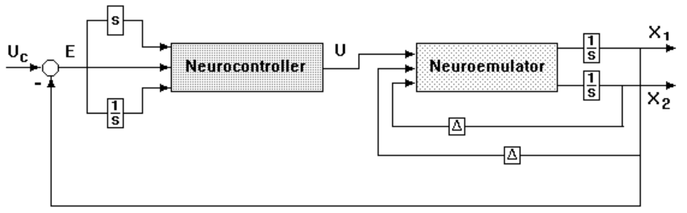
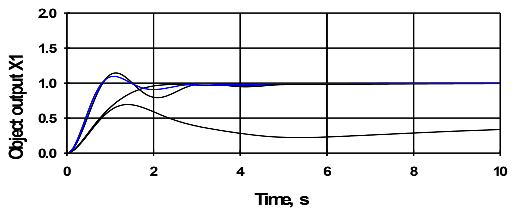


Figure 10 Prediction unit of the DC motor control system

Figure 11 exhibits transient process curves for the best version of the population in different search stages.



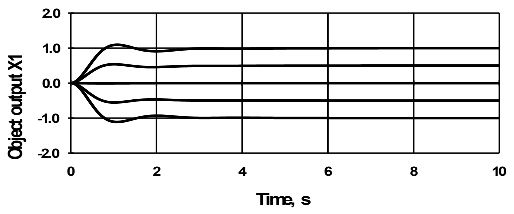


Figure 11 Best neuroemulator output transient processes for Uc=σ(t) in the 1st, 2,000th, 4,000th and 5,000th training steps

Figure 12 Transient processes in the DC motor control system with the trained neurocontroller for Uc=kσ(t), with k =-1, -0.5, 0, 0.5, 1

As one can see in this figure, the longer the algorithm works, the better the solutions are. The neurocontroller parameter vector found by the 5,000th generation provides for quite a satisfactory solution to the above-mentioned problem of the adaptive PID-neurocontroller synthesis.

Figure 12 illustrates the predicted object behaviour for not only the training data set but also for intermediate input signal values, what confirms the universality of ANN approximating properties.

Figure 13 shows the final version of the trained neurocontroller.

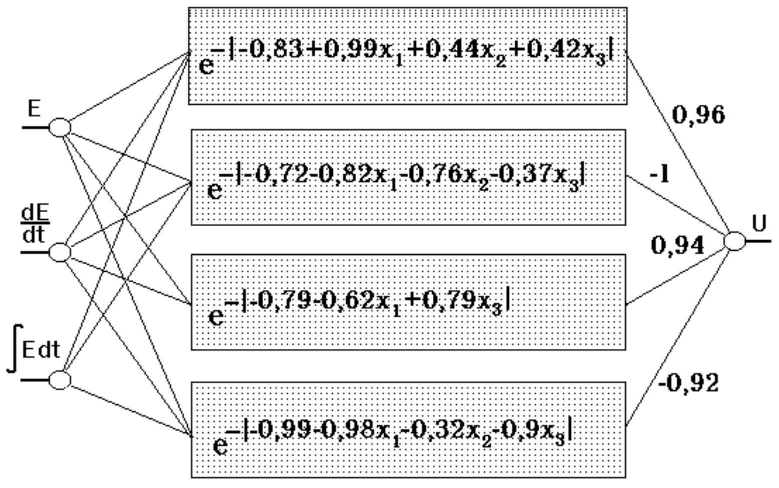

E
dE/dt
∫Edt
e^{-|-0,83+0,99x_1+0,44x_2+0,42x_3|}
e^{-|-0,72-0,82x_1-0,76x_2-0,37x_3|}
e^{-|-0,79-0,62x_1+0,79x_3|}
e^{-|-0,99-0,98x_1-0,32x_2-0,9x_3|}
0,96
-1
0,94
-0,92
U

Figure 13 Trained neurocontroller

Conclusion

Even though the considered example is simple enough, it gives a good illustration of further ANN implementations in electrical drive control systems. In reality, we do not always know a control object transmission function, but it is not, however, a strict requirement. To synthesize a control system, one only needs some experimental data about control object dynamical behaviour, by means of which he would manage to train the neuroemulator and then, bearing in mind the controlling objectives, the neurocontroller as well. As for the implementation of the system components, GA-based training unit with the prediction unit can be represented in form of software and the neurocontroller itself - in form of hardware.

Our nearest problem to solve is to develop software which allows to perform on-line training and adaptation of real control system neural elements.

Acknowledgements

This work was supported by Ukrainian Academy of Technical Sciences and, in part, by the International Soros Science Education Program (ISSEP) through grant N GSU052104.

References

1. McCulloch W.S., Pitts W. A logical calculus of ideas imminent in nervous activity // *Bulletin Mathematical Biophysics.*- 1943.- 5.- pp.115-133.
2. Rosenblatt F. The perceptron: A probabilistic model for information storage and organization in the brain // *Psychological Review*.- 1958.- 65.- pp.386-407.
3. Hopfield J.J. Neural networks and physical systems with emergent collective computational abilities // *Proc. of the National Academy of Sciences*.- 1982.- 79.- pp.2554-2558.
4. Rumelhart D.E., Hinton G.E., Williams R.J. Learning internal representation by error propagation // In: *Parallel Distributed Processing* (Eds. D.E.Rumelhart and J.L.McClelland), Vol. I Foundations. Cambridge, MA: MIT Press.- 1986.- pp.318-362.
5. Hlavackova K., Neruda R. Radial Basis Function Networks//*Neural network World.*- 1993.- vol.3.- N 1.- pp.93-101.
6. Hornik K., Stinchcomb M. and White H. Multilayer Feedforward Networks are Universal Approximators // *Neural Networks.*- 1989.- N 2.- pp.359-366.
7. Narendra K.S., Parthasarathy K. Identification and control of dynamical systems using neural networks // *IEEE Trans. on Neur. Net*..- 1990.- vol.1.- N 1.- pp. 4-27.
8. Schaffer J.D., Whitley D., Eshelman L.J. Combinations of Genetic Algorithms and Neural Networks: A Survey of the State of the Art // *Proc. Int. Workshop on Combinations of Genetic Algorithms and Neural Networks* (Eds. L.D.Whitley, J.D.Schaffer).- 1992.- Baltimore, Maryland.- pp.1-37.
9. Klepikov V.B. et al. Diploidy-based Genetic Algorithm in Nonstationary Environment //In: *Automatic Electrical Drive Problems. Theory, Practice* (Eds. V.B.Klepikov et al). Kharkov, OSNOVA Press.- 1995.- pp.108-110.
10. Klepikov V.B., Mahotilo K.V., Sergeev S.A. (1995) Modification of Holland's reproductive plan for diploid populations // In: *Artificial Neural Nets and Genetic Algorithms* (Eds. D.Pearson et al), Springer Verlag, pp.337-339.