Информационные процессы, Том 6, № 2, 2006, стр. 114–123.

A study of a Code Division Multiple Access detector based on a recurrent neural network

D.S. Osipov

Institute for Information Transmission Problems, Russian Academy of Sciences
B. Karetnyi per. 19, GSP-4, 101447 Moscow, Russia
email: d_osipov@iitp.ru

Received April 11,2006

Abstract—In this paper a Code Division Multiple Access detector based on a neural network, which is proposed in [4], is considered. It is shown how the choice of an updating scheme and/or the network parameters influences the detector performance. A weight updating scheme that might be used instead of the classical ones is proposed.

1. INTRODUCTION.

Code Division Multiple Access (CDMA) is one of the most powerful random multiple access techniques. A number of CDMA methods that differ both in the performance manner and tasks set were proposed recently. Direct Sequence CDMA (DS-CDMA) is a simple method of the kind, which enables one to overcome certain imperfections of other multiple access methods as well as those of other CDMA methods. In a DS-CDMA system each information sequence is multiplied by a code sequence, all resulting encoded sequences being transmitted simultaneously. That is why it is necessary to obtain an information sequence from the received one, the process itself being called *detection* and the device, which is to implement it a *detector*.

An optimal detector for asynchronous additive multiple access channel white Gaussian noise (AWGN), based on multiuser maximum likelihood sequence estimation (MU-MLSE), has been proposed by Verdu in [5]. The complexity of MU-MLSE, which is based on the Viterbi algorithm, grows exponentially as the number of users increases. Thus, a low complexity suboptimal detector is needed. One can design such a detector using either the maximum likelihood (ML) principle or the maximum a posteriori (MAP) principle. Our goal is to maximize the balance between computational complexity and detection quality. A possible approach to solving the problem is to use a neural network.

Neural networks (NN) are one of the most powerful Artificial Intelligence techniques. The technique's learning and generalization ability is closely related to its training algorithms' basic principle, i.e. to changing network parameters in order to approximate in the best way possible the network's input-output dependency either in continuous (as in e.g. multilayer perceptrons (MLPs) or radial basis function networks (RBFNs)), or in discrete form (as in e.g. self-organizing feature maps (SOM) or linear vector quantization (LVQ) networks). That is why most NN types might be thought of as a multidimensional optimization tools. Thus, many problems that are describable in terms of multidimensional optimization could be successfully solved by means of neural networks. The same goes for the detection problem.

Several possible approaches to the problem were proposed recently. For example, in [1] a multilayer perceptron - based detector was proposed. In [3] a radial basis function network was used for the same purpose. However, it is stated in [4] that the computational complexity of neural

networks of the type grows exponentially. Moreover, if a feedforward network is used, a parameters choice problem is to be solved. For example, if a multiplayer perceptron is used, the number of hidden layers, the number of nodes in each hidden layer and other parameters are to be chosen.

That was why it was suggested in recent papers that another class of neural networks should be used. These are recurrent neural networks, which are able to learn to take into consideration the previously learnt information due to feedback presence. It is mentioned in [4], that the complexity of a MU-RNN detector grows linear with the growth of the number of users. Thus, in real systems where the number of users is great the complexity of a MU-RNN detector will be much lower than that of a MU-MLSE or a feed-forward neural network-based detector.

In [4] a CDMA detector based on a recurrent neural network, known as a Hopfield neural network, was introduced. A Hopflied neural network, is a McCulloch - Pitts neuron-based neural network, which was initially introduced in [2].

On the basis of analogy with physical systems, J. Hopfield proposed to build a neural network structure, consisting of neurons, formed in accordance with the classic model of McCulloch - Pitts. Each of the latter is connected with the others via feedback connections, i.e., appears afferent towards them. In that case, learning amounts to a search of weight values, minimizing the function of the network potential energy (the Lyapunov function). J. Hopfield had also given the formula for calculation of weight values and shown, that a sufficient condition for the network stabilization is the fact, that all the elements of the main diagonal of the weight coefficient matrix are equal to zero (that follows from lack of the neuron's feedback with itself) and the weight matrix is symmetrical.

Acting on the similarity of the Hopfield network Lyapunov function and the logarithmic maximum likelihood function of an optimal detector, the authors of [4], showed, that such a network, striving to minimize its potential energy, maximizes in that way the maximum likelihood function, i. e. operates as a suboptimal neural network detector. Here a short sketch of the simplest form of the algorithm is presented. Let u_i be the ith component of the received vector U, which corresponds with the information vector D. Then:

Step 1. Initialize the weight matrix

$$w_{ij}^0 = \begin{cases} u_i \cdot u_j & i \neq j \\ 0 & i = j, \end{cases} \tag{1}$$

where w_{ij}^t is the value of the weight between ith neuron and jth neuron at tth iteration.

Step 2. Calculate the network output y:

$$y_n^{s+1} = f\left(\sum_{n=1}^{K} w_{ij}^s \cdot y_n^s + u_n\right), \tag{2}$$

where y_n^{s+1} is the value of the nth neuron output at s+1 iteration, and f is an activation function.

Step 3. Test the network stabilization:
If none of the weights has changed by a value more than δ_0 then y_n^{s+1} is the detected value,
else go to step 4

Step 4. Re-initialize the weight matrix using the appropriate scheme (i.e. a strategy of choosing weights that are to be changed) and the following equation:

$$w_{ij}^{s+1} = \begin{cases} y_i^{s+1} \cdot y_j^{s+1} & i \neq j \\ 0 & i = j. \end{cases} \tag{3}$$

Return to step 2.

In terms of DS-CDMA MU-MLSE detector W is a crosscorrelation matrix. Thus, the training procedure is aimed at finding a crosscorrelation matrix (that is the matrix, which minimizes the potential energy (maximizes the likelihood function)).

Notice that in this short sketch we are concentrating on one information vector only. However, if the ISI is small enough (the latter condition might be satisfied if, for example, a guard interval is used to separate information blocks) the same algorithm might be used. In this case we have a two dimensional neural network, which subdivides at m one dimensional neural networks, each of which is to be trained according to the algorithm mentioned above.

Still it is evident that the performance of the described MU-RNN can be improved still further. One of the simplest ways to improve the detector performance is to choose the weight updating scheme properly. The problem will be investigated further on. We are aiming at investigating the influence of the choice of the weight re-initialization on the communications quality. Therefore we shall consider dependencies of error probability for three different weight re-initialization schemes on the value of signal/noise ratio for four active user in order to demonstrate how an appropriate selection of a weight re-initialization scheme can improve the detector performance. The influence of the network stabilization parameter is also under consideration here. In order to demonstrate the network stabilization parameter influence we shall consider dependencies of error probability for three different network stabilization parameter values on the value of signal/noise ratio for four active users.

2. SIMULATION.

The parameter choice in the course of the experiment planning was enacted taking into account the simulation time limit and computational capacity of the equipment. The following inequations shall be true for the parameters of the model of a multiple access system, for it to comply with the requirements applied to real systems:

$$\begin{array}{l} \frac{Km}{N} < 1, \\ \frac{K_0 m}{N} > 1. \end{array} \tag{4}$$

It follows from the inequations (4), in particular, that the data block length increase requires the proportionate increase of the code matrix dimension; which in its turn causes increase of time necessary for simulation. Since in DS-CDMA systems code sequences are used to separate messages sent by different users the problem of code sequence type selection is of great importance. In real CDMA systems it is common practice to use pseudonoise sequences for the purpose, since they meet several important requirements. In our work we shall use Gold sequences, which are used, for instance, in WCDMA. Such sequences are of a relatively big period, therefore, their length is great. In order to decrease the time spent on simulation we shall use relatively short Gold sequences of the order n=7, with the length equal to the maximum period, i.e.. N=127 symbols. For the same reason we shall use small data blocks of length m=10 symbols. However, we still need long sequences for realistic simulation. Taking into account the simulation period limit, let us accept the minimal block length as equal to 100 data blocks (in that way the length shall amount to L=1000 symbols). Since

we have chosen sequences with a relatively small period the number of users is upper bounded by the total number of Gold sequences of the period. Thus, the total amount of users was accepted as equal to K=129. Consequently, we are to use a small number of active users (we will accept K_0=4), whereas our parameters do meet inequations (4). Thus, the model is realistic and the simulation results can be used if the system parameters meet (4).

3. WEIGHT UPDATING SCHEME INFLUENCE.

The easiest way to calculate new weights' values is to use the previously applied formula utilizing the neural network outputs for the current iteration as a standard values. The major question arising in the course of the analysis amounts to determining which of the weights and in what order are to be subjected to re-initialization. We shall consider some possible approaches to solving the problem in the present section.

One of the simplest schemes is the parallel one. The weights of all the neurons are reinitialized under it after each iteration. Another re-initialization method amounts to re-initialization of a random neuron. In both cases the re-initialization process is terminated in case of the weight coefficients' changes not exceeding a certain set in advance small value. Both methods are noted for lack of rationality in the adopted approach to re-initialization. The utilization of the following scheme may be recommended as an alternative approach: let us re-initialize the neurons whose output values differ from the previous iteration values most of all. Let us call it a "goal" scheme.

Diagrams of dependencies of error probability for the schemes under consideration (parallel-"par", random — "rand" and the proposed above scheme (marked "goal") respectively) on the value of the signal/noise ratio in dB for each of the four active randomly chosen users are given below.

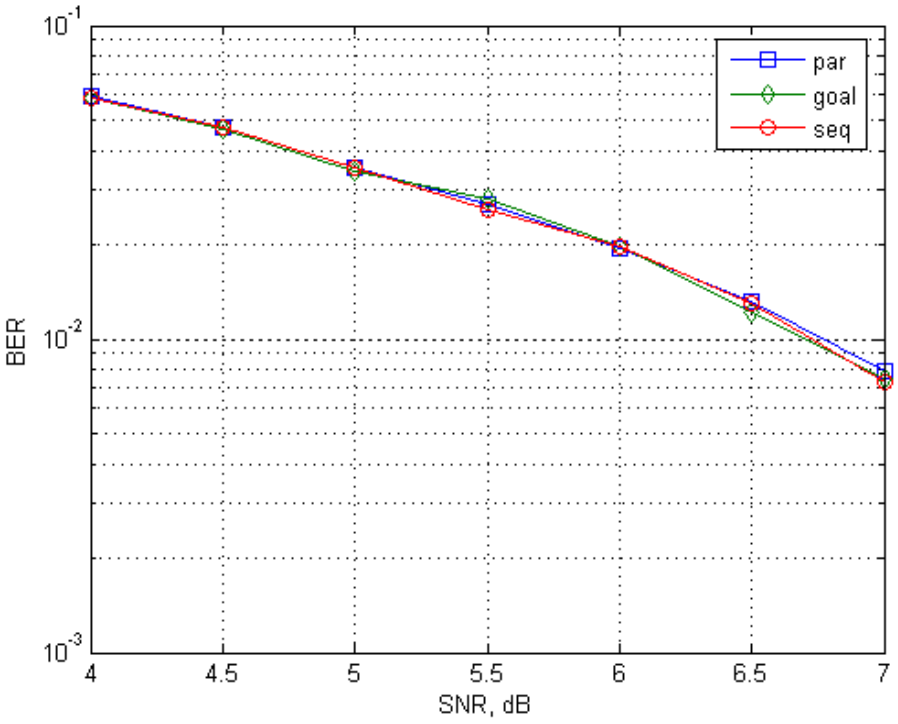

Fig 1a. Dependence of error probability for different weight re-initialization schemes on the value of signal/noise ratio in dB. (For the first user)

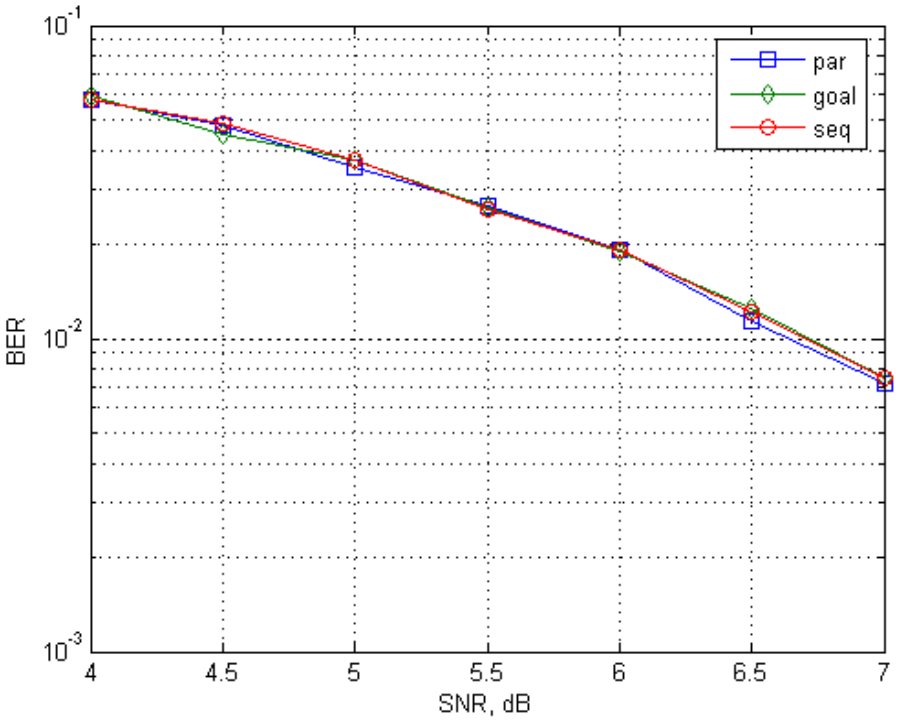

Fig 1b. Dependence of error probability for different weight re-initialization schemes on the value of signal/noise ratio in dB. (For the second user)

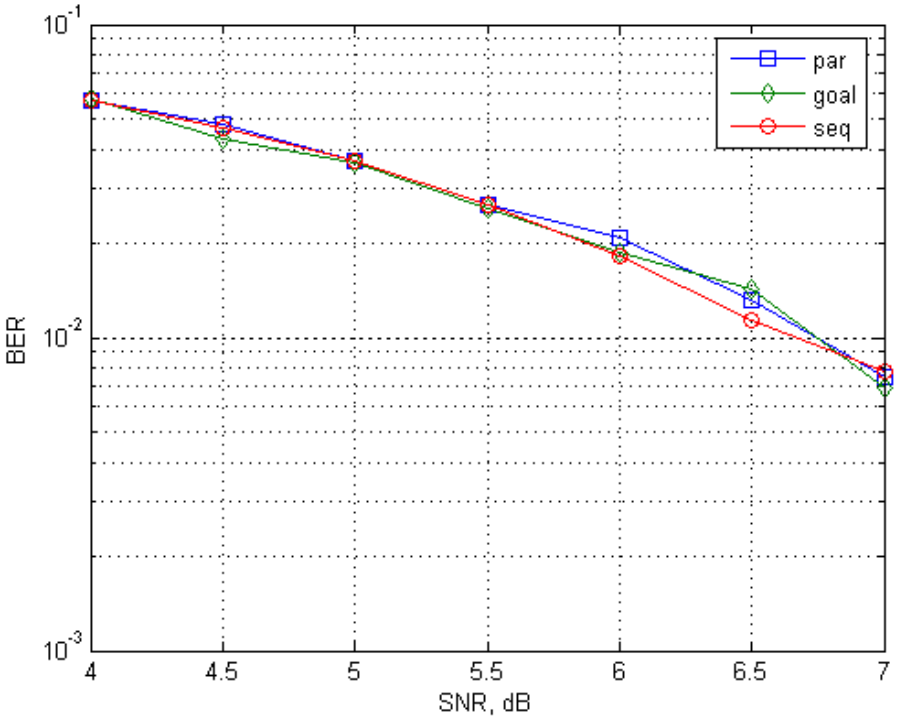

Fig 1c. Dependence of error probability for different weight re-initialization schemes on the value of signal/noise ratio in dB. (For the third user)

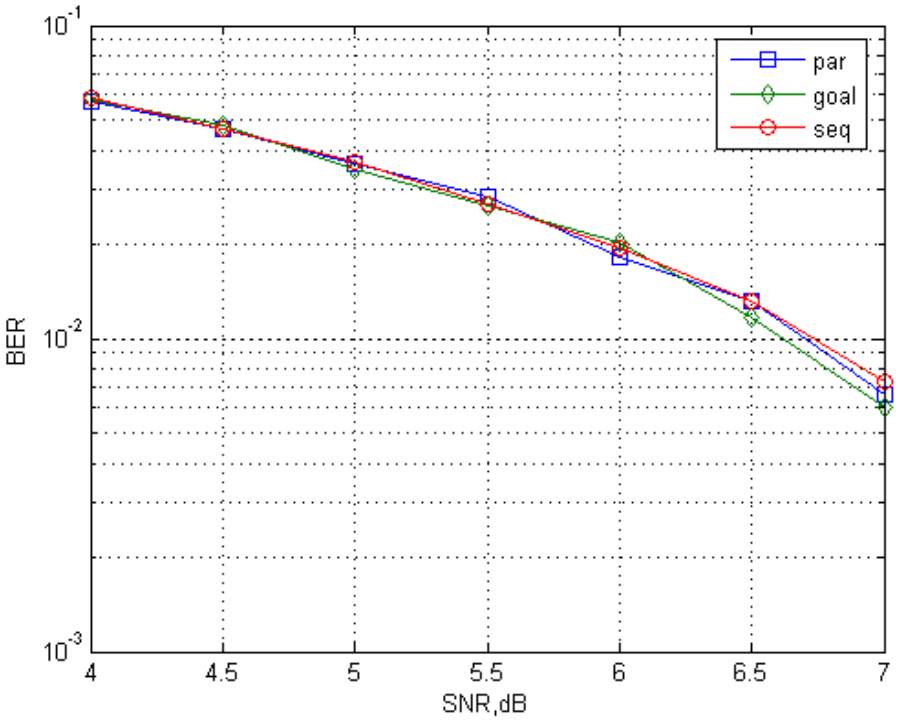

Fig 1d. Dependence of error probability for different weight re-initialization schemes on the value of signal/noise ratio in dB. (For the fourth user)

Fig. 1 demonstrates that all the three schemes under consideration ensure very close values of BER (Bit Error Rate) SNR. It must be mentioned that both hardware implementation requirements and computational complexity of the schemes under consideration differ. In case of the goal and random schemes implementation a block is needed, which chooses the neuron to be re-initialised. On the other hand the average number of training steps for the random scheme is about fifty times more than that for other schemes (see Table 1). Thus it is advisable to use parallel or goal scheme.

Table 1. Dependence of average number of training steps for different schemes on the value of SNR in dB.

scheme	average number of training steps			
parallel	6.1500	6.2270	6.3320	6.4792
goal	7.4870	7.6200	7.7460	7.8918
random	389.1250	411.1460	413.8600	453.3207
SNR, dB	5	6	7	8

All the further experiments are to be carried out with a neural network detector functioning on the basis of the training scheme described above in order to show how an accuracy parameter value change influences the network performance.

4. DEPENDENCE OF BER ON THE REQUIREMENTS TO CLOSENESS OF ESTIMATION OF THE NETWORK STABILIZATION.

In the preceding paragraph we have already mentioned the network outputs stabilization as the parameter characterizing the network subsequent training ability. The network condition, under

which new iterations do not lead to relevant changes of output values, are called output stabilization here. The latter means that further training is inexpedient.

The network is accordingly considered stable in case the maximum element δ of the Δ matrix, determined as the difference between the current and the previous outputs conditions does not exceed a certain set value of δ_0. It is clear from the above that the dynamics of the network training is to depend on what level of the outputs changes shall be considered relevant, i.e. on the value of δ_0.

We may assume, that the network for which the $\delta_0$value amounts to less shall be trained better. We endeavor to show later on in what way the change of the $\delta_0$value influences the neural network operation quality.

The following charts show the dependences of the error probability on the signal/noise ratio for three different values of δ_0.

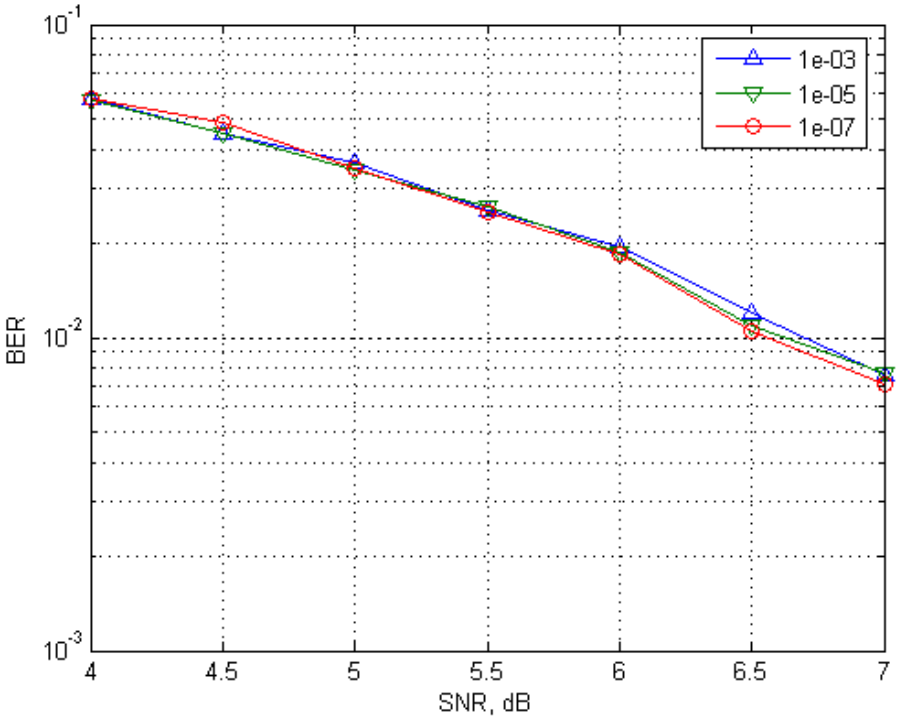

Fig. 2a. (For the first user)

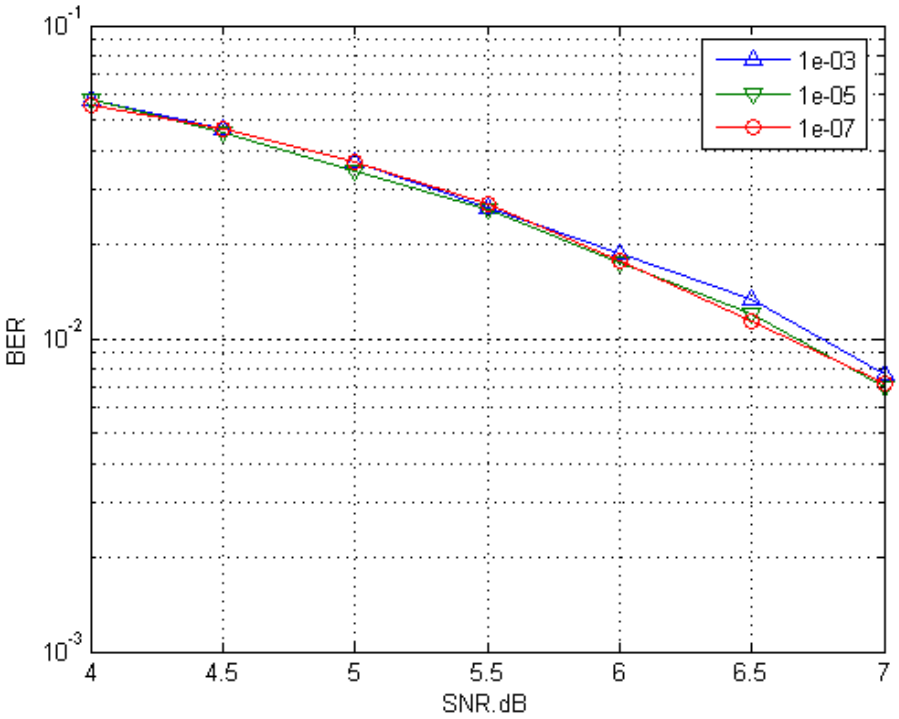

Fig. 2b. (For the second user)

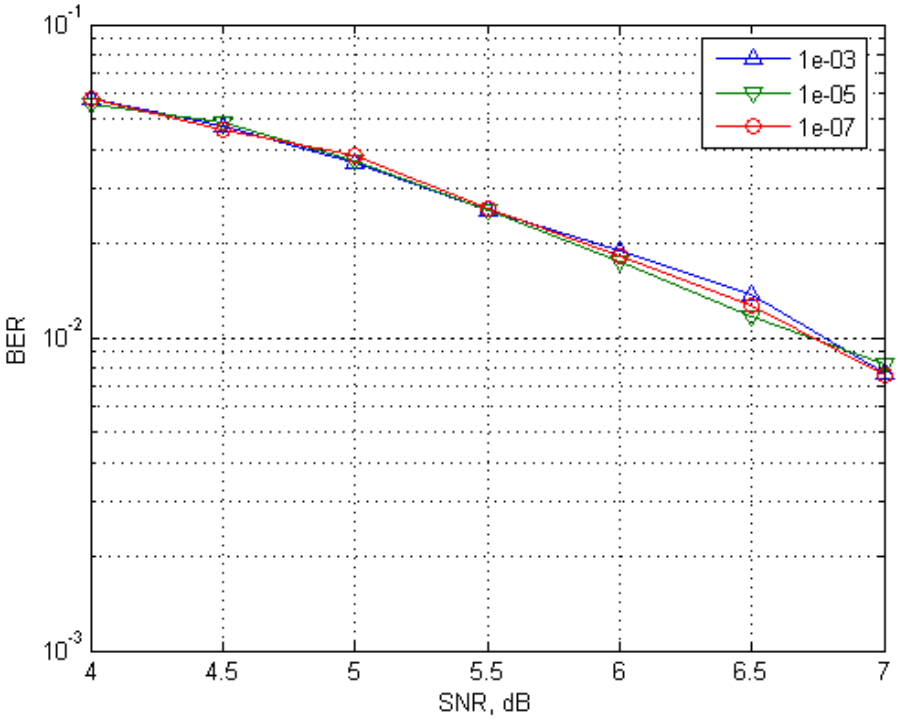

Fig. 2c. (For the third user)

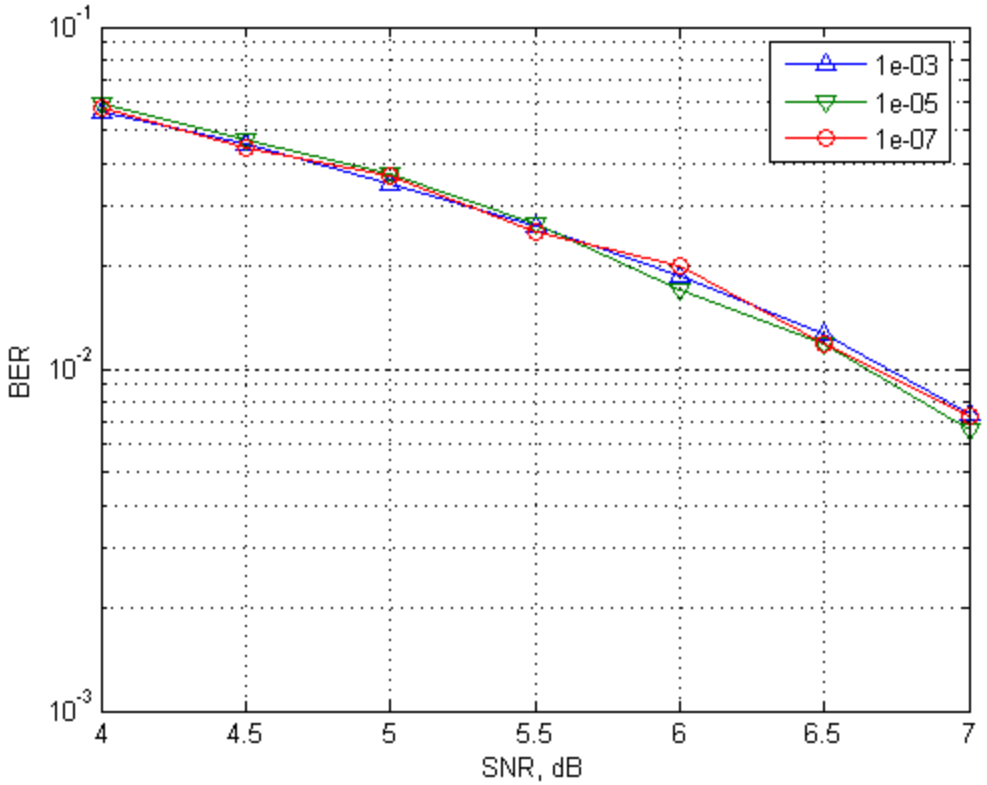

Fig. 2d. (For the fourth user)

As is evident from Fig. 2, the MU-RNN detector BER values for all the three accuracy parameter values under consideration are very close. For SNR values exceeding 6 dB in the first and second cases (see Fig. 2 a. and Fig 2. b), BER values ensured by the neural network detectors operating with the values of the δ_0 parameter of 10^{-9} and 10^{-7} respectively are slightly better than the corresponding BER value of the detector with the accuracy level of network stabilization of degree 10^{-5}, whereas in the cases of the third and the fourth user BER values are almost equal for all the SNR values under consideration.

Data from Table 2 demonstrate that an average number of steps amounts to approximately an absolute value of the degree of the $\delta_0$parameter. Therefore in most cases it is advisable to use relatively small values of the accuracy parameter.

Table 2. The dependency of the average number of iterations for different values of the accuracy parameter on the SNR values in dB.

precision	average number of iterations			
1e-05	5.4380	5.5830	5.7160	5.7883
1e-07	7.5110	7.6260	7.7570	7.8945
1e-09	9.5870	9.6780	9.8500	10.0492
SNR, dB	5	6	7	8

5. CONCLUSION.

Thus, whereas it is in principle possible to increase the algorithm efficiency by the appropriate choice of the network parameters and/or training scheme the performance improvement ensured by such change is too small and computation consuming. Nevertheless the above proposed scheme may be used instead of the random one if sequential processing is used.

REFERENCES

1. Aazhang B., Paris B.- P. Orsak G.C. "Neural Networks for Multiuser Detection in Code-Division Multiple-Access Communications".- IEEE Transactions on Communications 1992 vol. 40 pp. 1212-1222

2. Hopfield J. J., Tank D.W. "Neural Computation of Decision in Optimization Problems".- Biological Cybernetics 1985. vol. 52, pp.141-152

3. Mitra U., Poor H.V. "Neural Network Techniques for Adaptive Multiuser Demodulations".- IEEE Journal on Selected Areas in Communications. 1994 vol.12, pp. 1960-1970

4. Teich W. G., Seidl M. "Code Division Multiple Access Communications: Multiuser Detection based on a Recurrent Neural Network Structure" IEEE ISSSTA'96 pp. 979-984.

5. Verdu S. "Minimum Probability of Error for Asynchronous Gaussian Multiple-Access Channel"- IEEE Transactions on Information Theory. 1986. vol. IT-32, pp. 85-95, January

This paper was recommended for publication by V.V.Zyablov, a member of the Editorial Board