# NEURAL NETWORK SIMULATIONS IN MATLAB

Mansour ABAID, Doctoral Degree Programme (3)
Dept. of Telecommunications, FEEC, BUT
E-mail: Mansour7072003@yahoo.com

Supervised by: Dr. Karol Molnár

## ABSTRACT

This paper introduces the overview of the simulation of the neural network in Matlab environment based on solving the priority-switching problem. This optimization problem is solved by Hopfield neural network with improved performance.

## 1 INTRODUCTION

An artificial neural network with their parallel architecture represents a very perspective way for future data processing. In comparison with classical, sequential data processing neural networks offer a more efficient parallel processing. They are based on the neural system of living organisms what, in highly simplified point of view, is made of a large number of computation elements.

In year 1985 Hopfield presented a new architecture of neural networks with feed-back connections. This network, using iteration processes, is able to solve combinatorial optimization problems well known from mathematics. A lot of work was spent on modifying the original architecture and making it more it more efficient. Very significant results were achieved by S. V. B. Aiyer and A. Gee during their works, focused on algebraic transformation of the original mathematic model of Hopfield network. They presented several new algorithms that give better solutions for standard optimization problems.

However, quite a lot of theoretical work deals with Hopfield neural networks no really significant practical implementations of this network were presented. One possible solution could be to implement Hopfield network to optimize switching process in priority based switch fabric. Such a switch fabric could be used as a part of a switch or router in a computer network. Suitable mathematical expression of the priority-switching problem together with the development of a neural network based optimization process able to solve this problem is the main part of my Ph.D. work. During my Ph.D. studies a simulation model of such a switch fabric was built in Matlab language. In the next stage a graphical user interface was created allowing to simulate the behaviour of the switch fabric and to optimize the parameters of the software realization. The optimization is done based on the results obtained from the simulations in Matlab and presented in this paper.

## 2 PRIORITY SWITCHING PROBLEM

In classical Ethernet networks (using classical switches) there is no native support for frame prioritization. This situation is not very suitable for modern multimedia applications, where the transmission delay is a very important parameter. If the frames are of the same priority (what is the case, when priority support is not implemented), a file transmission can block the switch and can increase the transmission delay for other more important communications e.g. voice or video communication. The only solution for this problem is to add a priority field to the transmitted data expressing the importance of the frame. In the input buffer there can be several frames waiting for the transmission. From the point of view of the switch instead of the destination address contained in the header of the frame the corresponding output port of the switch is more important. So the destination address in the frame is used only to identify the output port. The connection process is then based only on the port numbers. The frames waiting in the input buffer of one input port can be expressed mathematically as a vector, where the position of the element in the vector specifies the required output and the value of the element expresses the priority. A vector (4, 60, 1, 255) means that there are frames with:

priority 4 and destination port 1,

priority 60 and destination port 2,

priority 1 and destination port 3,

priority 255 and destination port 4.

If value 255 is assigned to the least important frame, it could be used in the situation when there is no request for the given port. For each input port such a vector can be constructed. Connecting these vectors a matrix can be constructed. This matrix can be called as priority matrix.

The problems related to finding the optimal switch configuration can be regarded as a problem of priority switching. The solution to this problem is a binary configuration matrix. If the element in the *i*-th row and the *j*-th column of the configuration matrix has value one, then the *i*-th input will be connected to the *j*-th output. As a consequence, only one element with value one can exist in each row and in each column. This restriction is the confinement term of the optimization problem. The object function, which should be minimized, is based on the priority values assigned to the data units. The configuration matrix says which inputs and which outputs will be connected with each other. Thus it is also defined which data units will be forwarded from the input to the output. If the highest priority value is expressed by the smallest numerical value, then the configuration of the switch fabric will be optimal if the sum of the priority values is the smallest. This means that the sum of priority values of the selected data units defines the object function.

## 3 THE HOPFIELD NEURAL NETWORK ALGORITHM

The biggest problem of the classical Hopfield network is that practical implementations of it are very ineffective. It quite often generate not optimal solutions for the optimization problems. The problems with efficiency are caused by the very complicated energy function containing several terms to express both the confinement criteria and the object function of the optimization problem. Due to the complicated energy function, the classical Hopfield network can stuck in the local minimum and will not reach the global minimum

corresponding to the optimal result. A solution to this problem was offered by Aiyer in [2]. In this work the global energy function of the optimization problem is divided into two parts: the confinement function and the object function. The confinement function is responsible for ensuring the limitation criteria of the optimization problem while the object function expresses the pure optimization problem. Ayer also showed in his work how to separate the confinement function from the energy function to prevent them working against each other. The iteration process of the Hopfield model with improved energy function presented in [2] can be seen in Fig. 1. It shows that the iteration runs in two loops. The upper loop represents the projection of the state vector **v** into the subspace of valid solutions. The lower loop represents the optimization of the object function.

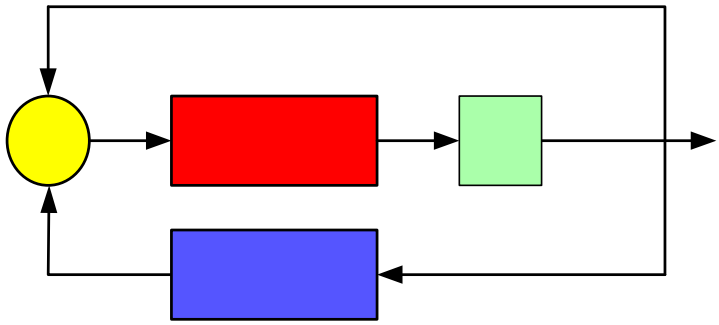

**Fig. 1:** *The iteration process of the Hopfield network with modified energy function*

Matrix **A** and vector **b** define the confinement term and if vector **v** satisfies (1) then the coefficients of the projection function $\mathbf{v} \leftarrow \mathbf{T}_{zs}\mathbf{v} + \mathbf{s}$ can be defined by (2) and (3).

$$\mathbf{A}.\mathbf{v} = \mathbf{b} \quad (1)$$

$$\mathbf{T}^{zs} = \mathbf{I} - \mathbf{A}^T (\mathbf{A}\mathbf{A}^T)^{-1}\mathbf{A} \quad (2)$$

$$\mathbf{s} = \mathbf{A}^T (\mathbf{A}\mathbf{A}^T)\,\mathbf{b} \quad (3)$$

The object function can be expressed by equation (4) where vector **c** contains the weights assigned to the corresponding outputs. If the output is active, this weight is added to the total sum. Matrix **Q** contains the weights assigned to each pair of outputs. That means $Q_{i,j}$ is the weight of the *i*-th and *j*-th neuron being active at the same time.

$$\mathbf{f}(\mathbf{v}) = \mathbf{c}^T\mathbf{v} + \mathbf{v}^T\mathbf{Q}_v \quad (4)$$

By using expressions (2) and (3) and the coefficients of object function (4), the parameters of the lower loop can also derived. The matrix $\mathbf{T}^{op}$ is defined by (5) and the vector of bias vales **i** is defined by (6).

$$\mathbf{T}^{op} = -2\mathbf{Q} + \gamma(\mathbf{T}^{zs} - \mathbf{I}) \quad (5)$$

$$\mathbf{i} = \gamma\mathbf{s} - \mathbf{c} \quad (6)$$

Matrix **I** is an identity matrix and the parameter $\gamma$ defines the weight between the confinement function and the object function.

$$\mathbf{T}^{op} = \mathbf{T}_{zs} - \mathbf{I} \quad (7)$$

$$\mathbf{i}^{op} = \mathbf{s} - \mathbf{c} \quad (8)$$

Vector **c** is the vector expression of the neural network's input matrix defined by the priority values of the frames in the input buffers. The input matrix is transformed to vector **c** using the operation *vec ()*.

## 4 OVERVIEW OF THE IMPLEMENTATION IN MATLAB

To approve the theoretical results a simulation model was developed. The simulation model was created in the Matlab environment. The Matlab program can be divided into three main parts. The first part defines the constants and other variables with a value unchanged during the whole iteration process e.g. matrixes **A** and $\mathbf{T_{zs}}$ or vectors **b** and s etc. (In the case that matrix **A** contains linearly dependent rows, the number of the rows must be reduced. Also vector **b** must be modified according to the reduction of matrix **A**.) The second part of the program realizes the projection of the vector of state values into the subspace of valid solutions and using the activation ensures that this vector will also remain inside the unit hypercube. The third part of the program solves the pure optimization problem. It calculates a vector of differences, which is added to the vector of state values. The second and third parts together realize the iteration process solving our optimization problem. At the beginning of the iteration process a random vector of state values is generated. Than the second part of the program is executed while the difference of the old and new vectors of state values $|\mathbf{v}_{old} - \mathbf{v}_{new}|$ is lower than $0.1*n$, where n is the number of the ports of the switch. When this condition is fulfilled the third part of the program is executed. By this part the new vector of state values is calculated and corrected again by the second part. The iteration process is executed while the matrix form of the state vector will not satisfy the confinement criteria (one single element with value one in each row and each column, all the other elements are equal to zero) or the maximum number of iteration loops is not exceeded. Using the simulation model several important parameters of the neural network were measured. First the dependence of the energy function on the number of iteration loops was analyzed. This relation is shown in Fig. 2. From the chart it can be seen that the network is stable and approximately after 6000 iterations the energy function reaches the global minimum and the iteration can be stopped.

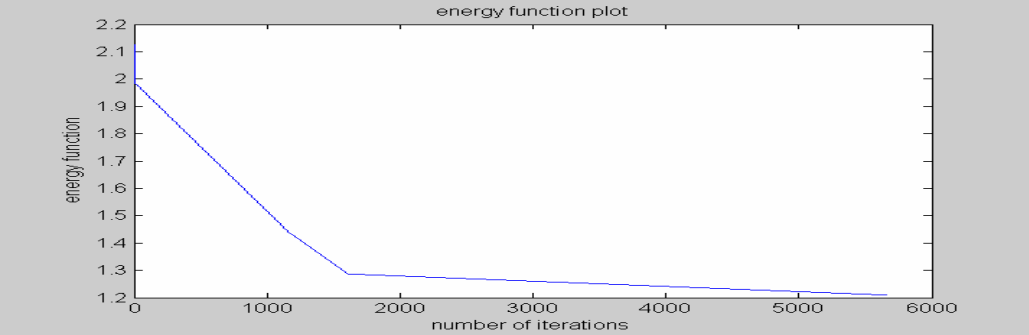


**Fig. 2:** *Dependance of the energy function on the maximum number of iteration loops*

The described algorithm is solved by an iteration process realizing numerical integration. To solve this integration an Euler method was selected because of its simplicity and speed. To improve the efficiency of the iteration process the effect of the Euler discretization time step was also examined. Fig. 3 shows the relation of the quality of the solutions to the time step $\Delta t$.

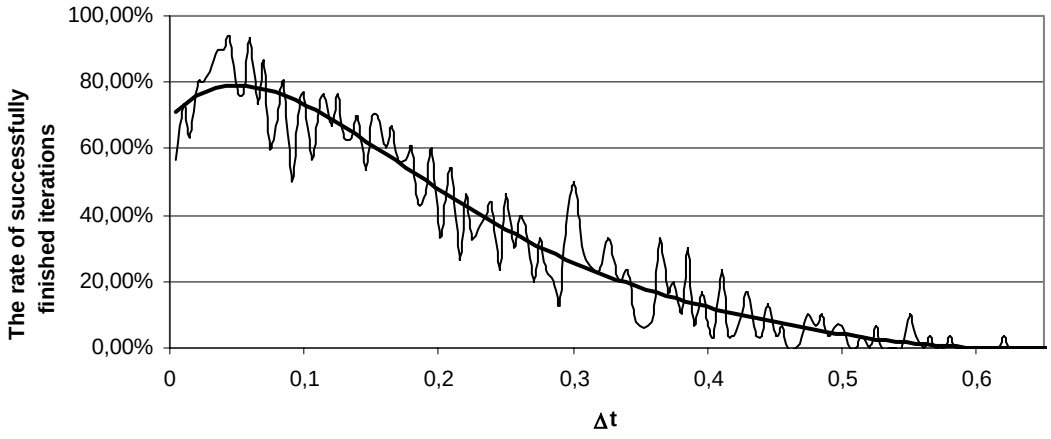


**Fig. 3:** *Dependence of success rate of iterations on the timestep Δt*

The time step *Δt* was examined in range from 0.0001 to 0.8 with step 0.005. Fig. 3 shows that an unambiguous value of *Δt* can be found, for which the efficiency of the iteration process is maximized. For too small values of Δ*t* the maximum number of iteration loops is exceeded before finding the result. On the other hand with large values of *Δt* the algorithm becomes too inexact and often misses the optimal solution. The optimal value of *Δt* is about 0.08.

## 5 CONCLUSION

Very effective system architecture can be built by using the Hopfield network with improved performance. The simulation in Matlab was used to perform a detailed analysis of the proposed neural network. The theoretical results obtained meet my expectations.

## REFERENCES

[1] Molnar, K.: Switching Optimization by Neural Networks

[2] Aiyer, S. V. B.: Solving combinatorial optimization problems using neural networks with application in speech recognition, University of Cambridge, United Kingdom, 1991

[3] Hopfield, J. J., Tank, D. W.: “Neural” Computation of decisions in optimization problems, Biological Cybernetics, Springer-Verlag.1985

[4] Smith, A. K.: Solving combinatorial optimization problems using neural networks, University of Melbourne, Australia, 1996.

[5] Troudet, T. P., Walters, S. M.: Neural Network Architecture for Crossbar Switch Control, IEEE Transaction on Circuits and Systems, vol. 38, No. 1. 1991