Neural Network Evaluation of Electromagnetic Interferences between HV Power Lines and Underground Metallic Pipelines

Dan D. Micu*, L. Czumbil*, A. Ceclan*, E. Simion*, Denisa Stet*, Liliana Cîmpan*
* Department of Electrical Engineering,
Technical University of Cluj-Napoca,
Baritiu str. 24-26, 400020 Cluj-Napoca, Romania, E-Mail: Dan.Micu@et.utcluj.ro

Abstract **– *This paper presents the use of neural networks in the study of electromagnetic interferences between high voltage power lines and metallic underground pipelines, for various constructive geometries. Results gained with neural networks are compared to the finite element solutions considered as standard ones. Our contribution relates to the implementation of the neural network AI technique, to the study of electromagnetic interference problems and the testing of the neural networks used in the studied case.***

Keywords: ***neural networks (NN), electromagnetic interference, high voltage power lines, pipelines.***

I. INTRODUCTION

Electromagnetic interference study between high voltage (HV) power system lines (PSL) and nearby unburied or buried metallic pipelines (MP) presents a great importance, given by the induced AC potentials. In most of the cases to reduce construction costs of gas, water or oil pipelines, they must share the same distribution corridor with HVPSL. Induced AC voltage in these MP could be very dangerous on both the personnel that may come in contact with them and on their structural integrity, due to electrical corrosive effects. In case of one phase or two phase PSL faults, induced AC voltages in unmitigated MP can reach thousands of volts [1].

Generally, electromagnetic interference problems are studied through the finite element method (FEM). As in all other engineering fields, FEM shows successful solutions in the numerical evaluation of the variation model, describing electromagnetic field problems.

Although the FEM yielded solutions are very accurate, regardless to problem complexity, the computing time of this method increases with the geometry, its mesh, material characteristics and requested evaluation parameters.

Since, on each new problem geometry taken under consideration FEM involves a remesh and a new evaluation of the calculus, the study of electromagnetic interference between HVPSL and MP for different system configurations requires expensive computing time.

Therefore, a scaling method of the results from one configuration case to another may be of interest if it provides less computing time. Currently two artificial intelligence based methods are studied worldwide: Fuzzy Logic Systems (FLS) and Neural Networks (NN). Our concern deals the last.

II. NEURAL NETWORKS

Neural Networks belong to a group of artificial intelligence techniques (AI), for data analysis that do not resemble with other classical analysis techniques. AI are learning about the chosen subject from de data provided to them, rather than being defined by user. NN get their knowledge by detecting relationships between input data [2].

A. Structure of an Artificial Neuron

Neural Networks were designed after the human brain neural network model. Like in the human brain case, the major building block of any neural network is the artificial neuron.

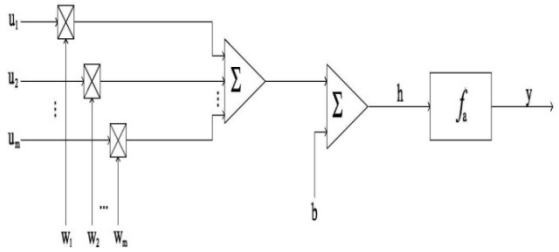


Fig. 1 Basic block diagram of an artificial neuron

As it can be seen from its block diagram (figure 1) an artificial neuron is a system which has a variable number of inputs $u_k, k=\overline{1,m}$ and only one output y. The m inputs of the artificial neurons are multiplied by some w_k parameters, called *weighs* and added to each other. The weighted inputs sum is added to a parameter b called *bias*. Then the last sum, denoted by h is used as an argument of the function which produces the artificial neurons output. This function is called *transfer function* and can take various forms, specific to each neuron in particular. Thus the artificial neurons output is:

$$y = f_a(h) \tag{1}$$

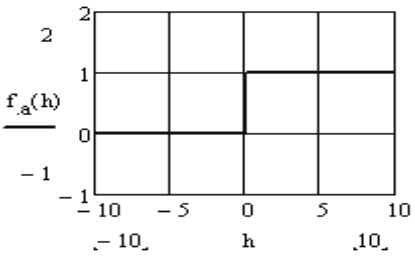
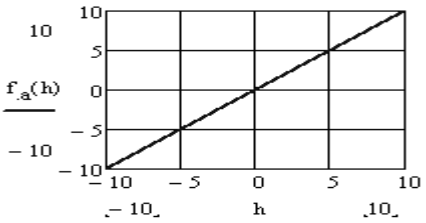
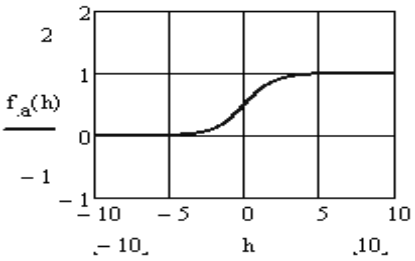
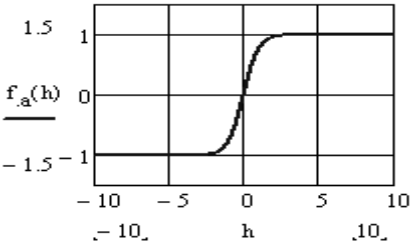
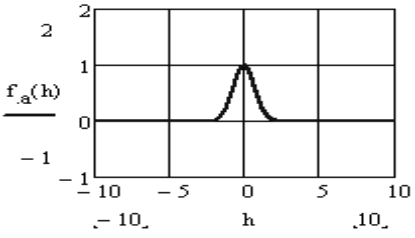
where:

$$h = \sum_{k=1}^{m} (u_k \cdot w_k) + b \qquad (2)$$

Weights and biases of artificial neurons are parameters that can be modified. Usually, their values are set up when the neural network is trained for the desired behavior. So an artificial neuron output depends only on its inputs and on its transfer function.

The most commonly used transfer functions in neural networks, with context relevance, are presented in the following table:

Table 1. Usually used Transfer Functions

Tranfer Function	**Function Relationship** $f_a(h) =$	**Derivative Relationship** $f_a`(h) =$	**Graphical Representation**
Hard limit transfer function	$\begin{cases} 0, h < 0 \\ 1, h \le 0 \end{cases}$	$\delta(h)$	
Linear transfer function	h	1	
Log sigmoid transfer function	$\frac{1}{1+e^{-h}}$	$\frac{e^{-h}}{(1+e^{-h})^2}$	
Hyperbolic tangent sigmoid transfer function	$\frac{1-e^{-2h}}{1+e^{-2h}}$	$\frac{4e^{-2h}}{(1+e^{-2h})^2}$	
Radial basis transfer function	e^{-h^2}	$-2he^{-h^2}$	

B. Neural Networks Structure

The structure of a neural network is specified by the number of layers, the transfer functions used in each layer and the numbers of neurons which compose a certain layer.

A group of artificial neurons from a layer work in parallel, have the same inputs and their outputs have the

same destination. The block diagram of a layer is presented in figure 2:

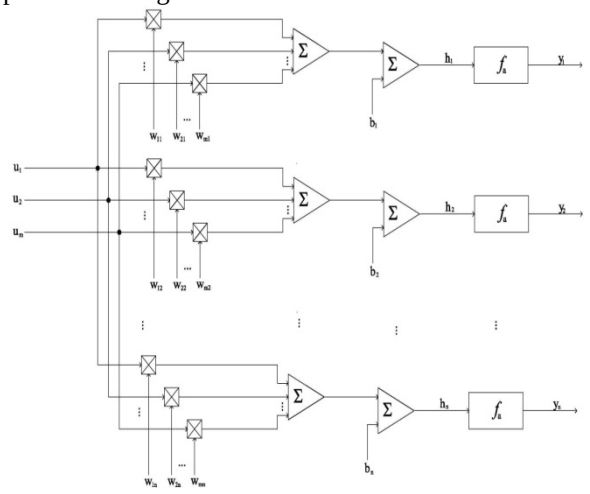


Fig. 2 Basic block diagram of a layer

The structure of any neural network must contain at least one layer of neurons, but can join as many as someone projects. The layer gathering the neurons which give the neural networks output is called *output layer*. This layer cannot miss from a NN structure. Layers which contain the neurons interposed between the global inputs of the neural network and the inputs of the neurons from the output layer are called *hidden layers*. The hidden layer whose inputs are the same to the global inputs of the neural network is also called *input layer*.

Usually, there are used feed-forward NN which contain a hidden layer and the output layer. The transfer function used for feed-forward networks must have a determinable derivative. For example, the hidden layer uses the hyperbolic tangent function, as the linear function is used in the output layer. Figure 3 presents the simplified block diagram of a two layer feed-forward neural network.

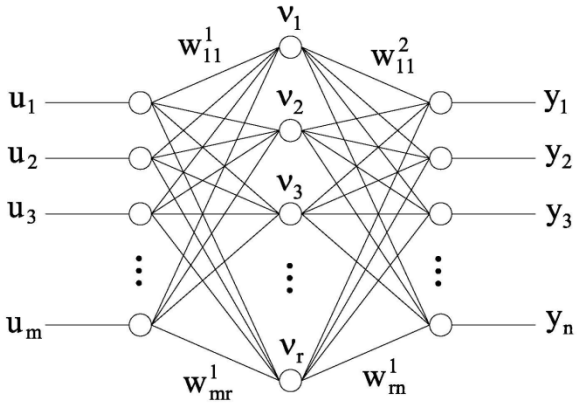


Fig. 3 Feed-forward Neural Networks

From this block diagram, we can deduce the relationships which define a feed-forward NN outputs, if we know its inputs $u_k, k=\overline{1,m}$. The argument of the hidden layer neurons transfer function is:

$$h_j^1 = \sum_{k=1}^{m}\left(u_k \cdot w_{kj}\right) + b_j^1,\ j=\overline{1,r} \tag{3}$$

So, the hidden layer neurons output is described by the following relationship:

$$\nu_j = f_{a1}\left(h_j^1\right) = f_{a1}\left(\sum_k \left(u_k \cdot w_{kj}\right) + b_j^1\right) \tag{4}$$

The argument for the transfer function of hidden layer neurons is:

$$h_i^2 = \sum_{j=1}^{r}\left(\nu_j \cdot w_{kj}\right) + b_i^2 \tag{5}$$

Finally, the feed-forward neural networks outputs are given by relationship (6):

$$y_i = f_{a2}\left(h_i^2\right) = f_{a2}\left(\sum_j \left(w_{kj} \cdot f_{a1}\left(h_j^1\right)\right) + b_j^2\right) \tag{6}$$

II. PROBLEM DESCRIPTION

In this paper we evaluate the magnetic vector potential (MVP), on the MP for the electromagnetic interference problem presented in [3]. The problem refers to a buried metallic gas pipeline which shares the same distribution corridor with a 145 kV HVPSL and 50 Hz frequency. Figure 4 shows a top view of the common distribution corridor:

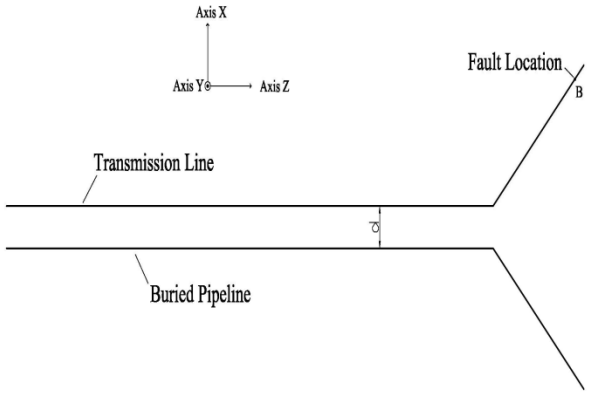


Fig. 4 Top view of the parallel exposure

It is assumed that a phase to ground fault at point B, far away outside the common HVPSL–MP distribution corridor, occures. The earth current associated with this fault has a negligible action upon the buried pipeline. This fact allows us to assume only an inductive interference caused by the flowing fault current in the section where the HVPSL runs parallel to the buried gas pipeline.

The HVPSL consists of two aluminum steel reinforced conductors per phase. Sky wire conductors have a 4 mm radius, the gas pipeline has a 0.195 m inner radius, a 0.2 m outer radius and a 0.1 m coating radius. The characteristics of the materials in this configuration have the following properties: the soil is

assumed to be homogeneous, MP and sky wires have a $\sigma = 7.0E + 05$ S/m conductivity and a $\mu_r = 250$ relative permeability [3].

End effects are neglected, leading to a two dimensional (2D) problem which depends on the separation distance *d* between HVPSL and MP, on the soil resistivity ρ, on the *x* and *y* coordinates of the point where the magnetic vector potential is desired to be determined. Figure 5 shows the studied configuration cross section:

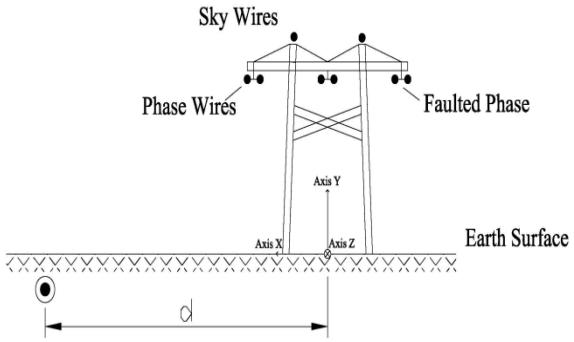


Fig 5. Cross section of the system under investigation

III. NEURAL NETWORK IMPLEMENTATION

In [3] and [4] there are disscused two fuzzy logic systems to evaluate the amplitude and the phase of magnetic vector potential. We intend to implement two neural networks for calculation of magnetic vector potentials amplitude and phase in different HVPSL–MP configuration, starting from some training data, which have been obtained using FEM calculus in [3] and [5]. The data sets, used to train our two neural networks are presented in Table 2.

To implement the amplitude and the phase neural networks, we use the *Neural Network* toolbox from the Matlab software, especially the *newff* function. This Matlab function creates a feed-forward neural network and is called as follows:

net=newff(P,T,S,TF,BTF,BLF,PF,IPF,OPF,DDF)
where:

- P - a $r \times q$ input data matrix (*r* – neural networks inputs number, *q* – training data set length);
- *T* - a $o \times q$ target data matrix (*o* – neural networks outputs number);
- *S* - a list of hidden layers sizes;
- *TF* - the list of transfer functions used for each layer (*tansig* – the hyperbolic tangent function for hidden layers and *purelin* –linear function for the output layer);
- *BTF* - the backpropagation network training function (default is *trainlm* - the Levenberg-Marquardt training function);
- *BLF* - denotes the backpropagation weight/bias learning function (default is *learngdm* – the gradient descent w/momentum weight/bias learning function);
- *PF* - the performance function (default is *mse* – the mean squared error function);
- *IPF* - a list of input processing functions;
- *OPF* - a list of output processing functions;
- *DDF* - the data division function.

It is no need to give all the parameters of *newff* function to call it; there are must be given only the ones which have a different value from Matlabs default values [6].

Table 2. Data Base used to train the Neural Networks

Nr. Crt.	*d* [m]	*x* [m]	*y* [m]	ρ [Ωm]	MPV	
					Amplitude 10^{-5} [Wb/m]	Phase [°]
1	70	70	-15	30	36,1	-22,8
2	100	100	-30	30	29,9	-31,23
3	800	770	-30	30	4,23	-82,64
4	800	785	0	30	4,27	-78,83
5	1000	1030	-15	30	2,48	-90,27
6	2000	1970	-22,5	30	0,476	-108,1
7	2000	2020,69	-8,61	30	0,436	-108,54
8	400	384,81	-7,82	70	17,2	-44,46
9	400	424,77	-6,93	70	15,8	-46,72
10	1000	970	-15	70	5,95	-73,04
11	1000	1007,5	0	70	5,68	-72,98
12	1000	1015	-30	70	5,47	-76,05
13	70	40	-15	100	53,8	-19,34
14	70	40	0	100	55,9	-18,53
15	100	92,25	-25,56	100	41,5	-23,98
16	800	770	0	100	10,4	-59,87
17	1000	1015	-30	100	7,16	-69,22
18	1000	1022,5	0	100	7,23	-67,27
19	300	312,38	-8,1	300	31,7	-29,23
20	300	324,05	-23,53	300	31	-30
21	2000	2007,5	0	300	5,86	-72,55
22	300	281,66	-27,03	500	37,5	-25,93
23	300	290,36	-15,8	500	37,1	-26,01
24	300	322,5	0	500	35,5	-26,74
25	1000	1030	-15	500	17	-44,6
26	150	120	-15	700	54,6	-19,26
27	400	384,81	-7,82	700	35,2	-26,89
28	700	690,36	-15,8	700	25,6	-34,07
29	700	712,38	-8,1	700	25,1	-34,41
30	150	150,55	-16,99	900	53	-19,7
31	200	194,77	-6,93	900	48,8	-20,9
32	800	830	-30	900	24,6	-35,01
33	1500	1499,09	-17,48	900	15,6	-46,35
34	70	54,81	-7,82	1000	70,3	-15,94
35	150	131,66	-27,03	1000	55,8	-18,98
36	500	524,05	-23,53	1000	32,9	-28,27
37	2000	2030	-15	1000	12,2	-52,73

To evaluate the amplitude of PMV we used a feed-forward neural network with 10 neurons in hidden layer and one neuron on the output layer. The hidden layers neurons use the *tansig*, hyperbolic tangent transfer function, and the output layers neuron uses the

purelin, linear transfer function. As performance function we used the *msereg*, mean squared error with regularization function. For the remaining neural network defining functions were kept at Matlabs default functions.

To calculate the phase of PMV we also used a feed-forward neural network with 10 neurons in hidden layer and one neuron on the output layer. The hidden layers neurons appeal the *tansig*, hyperbolic tangent transfer function, and the output layers neuron uses the *purelin*, linear transfer function. But in this case as performance function we needed the *mse*, mean squared error function. The remaining neural network defining functions were kept also at Matlabs default functions.

To train these two neural networks we chose the *train* Matlab function, which is called as follows:

net=train(Net,P,T,Xi,Ai)

where:

- *Net* is the neural network which has to be trained;
- *P* is the networks inputs list;
- *T* is the networks target outputs list;
- *Xi* is the initial input delay condition (0 by default);
- *Ai* is the initial layer delay condition (0 by default).

To obtain a higher accuracy for the results given by the two neural networks the training data base presented in Table 2 was multiplied twenty times.

The training process for both amplitude and phase neural networks took less than 10 seconds.

IV. NUMERICAL RESULTS

After training the two neural networks we verified the solutions provided by them. Therefore, we have compared the results obtained through the neural network method with the ones provided in [3] with FEM. The comparison was done for the training data set and also for a testing data set presented in table 3:

Table 3. Data Base used to test the Neural Network

Nr. Crt.	*d* [m]	*x* [m]	*y* [m]	*ρ* [Ωm]	MPV	
					Amplitude 10^{-5} [Wb/m]	Phase [°]
1	70	81,66	-27,03	30	32,9	-25,57
2	800	818,25	-13,5	30	3,88	-82,61
3	400	392,25	-25,56	70	16,7	-46,05
4	70	40	-30	100	50,9	-20,45
5	1000	980,55	-16,99	100	7,58	-67,1
6	200	215	-30	500	41,8	-23,83
7	700	670	-22,5	700	26	-33,74
8	1500	1524,77	-6,93	900	15,4	-46,56

To test the implemented neural networks for MPVs amplitude and phase, we used the *sim* Matlab function, which may be called as follows:

ResData=sim(Net,TestData)

where:

- *Net* - the neural network which has to be tasted;
- *TestData* - the input data used to test the neural network;
- *ResData* - the matrix where are placed the obtained outputs for the testing data set.

The results are presented in the graphical form of the absolute deviation between the solutions given by the implemented neural networks and those provided by FEM, as we can see in the following figures:

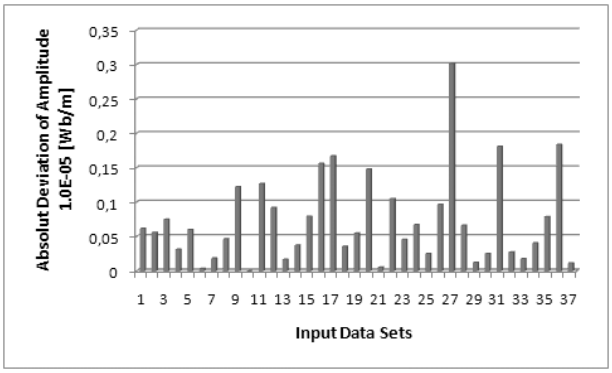


Fig. 6 Absolute deviation of amplitude for the training data

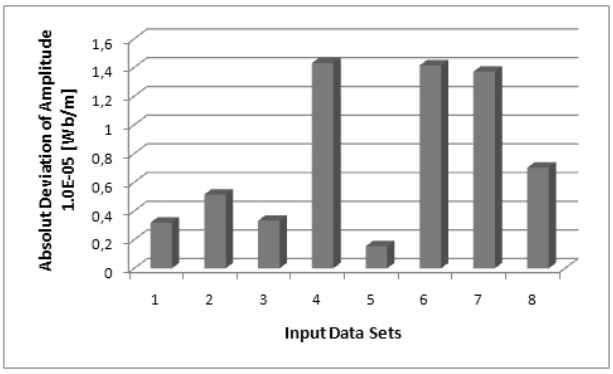


Fig. 7 Absolute deviation of amplitude for the testing data

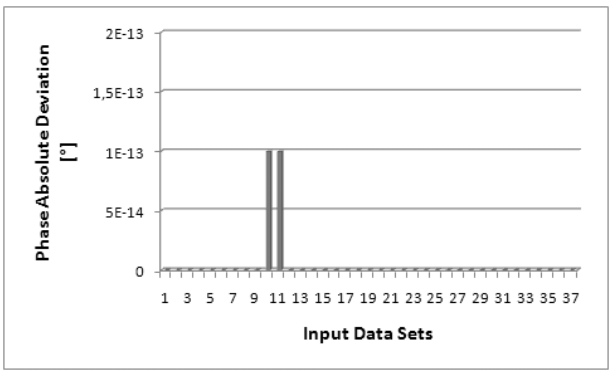


Fig. 8 Phase absolute deviation for the training data

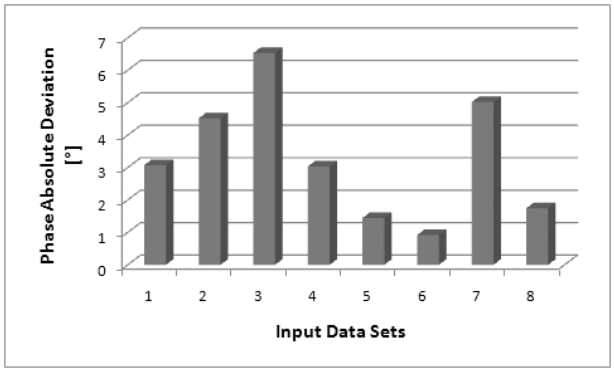


Fig. 9 Phase absolute deviation for the testing data

V. CONCLUSIONS

From the above presented figures it can be seen that absolute deviation of the solutions provided by the implemented neural networks, to those provided by FEM are small, almost insignificant for the training data incomes and somewhat higher but still negligible for the testing data incomes.

The evaluation of the MPV for different geometrical configurations, using neural networks is a very effective one, especially if we take into account the fact that the solutions provided by neural networks are obtained instantaneously and we do not have to pay expansive calculus as with FEM.

Our contribution relates to the implementation of the neural network AI technique, to the study of electromagnetic interference problems and the testing of the neural networks used in the studied case.

REFERENCES

[1] S. Al-Alawi, A. Al-Badi, K. Ellity: *An artificial neural network model for predicting gas pipeline induced voltage caused by power lines under fault conditions*, COMPEL: International Journal for Computation and Mathematics in Electrica land Electronic Engineering,Vol. 24, No. 1, 2005.

[2] A. Al-Badi, K. Ellithy, S. Al-Alawi: *Prediction of Voltages on Mitigated Pipelines Paralleling Electric Transmission Lines Using an Arificial Neural Network*, The Journal of Corrosion Sience and Engineering, Vol. 10, 29 March 2007, ISSN: 1466-8858.

[3] K.J. Satsios, D.P. Labridis, P.S. Dokopoulos: *An Artificial Intelligence System for a Complex Electromagnetic Fiels Problem: Part I – Finite Element Calculations and Fuzzy Logic Dvelopment*, IEEE Transaction on Magnetics, Vol. 35, No. 1, January 1999.

[4] K.J. Satsios, D.P. Labridis, P.S. Dokopoulos: *An Artificial Intelligence System for a Complex Electromagnetic Fiels Problem: Part II – Method Implementation and Performance Analysis*, IEEE Transaction on Magnetics, Vol. 35, No. 1, January 1999.

[5] Dan D. Micu, Emil Simion, Dan Micu, Andrei Ceclan: *Numerical Methods for Induced Voltage Evaluation in Electromagnetic Interference Problems*, 9th International Conference, Electric Power Quality and Utilisation, Barcelona, 9-11 October 2007, IEEExplore, Compendex 10.1109/EPQU.2007.4424091.

[6] H. Demuth, M. Beale: *Neural Networks Toolbox. Users's Guide*, ver. 3.0, The MATHWORKS Inc., 1998.