COMPARING ARTIFICIAL NEURAL NETWORK IMPLEMENTATIONS FOR PREDICTION AND MODELING OF DYNAMICAL SYSTEMS

André L. S. Braga, alsbraga@unb.br
Carlos H. Llanos, llanos@unb.br
University of Brasilia - UnB, Mechatronic Systems. Brasilia, Brazil

Leandro dos Santos Coelho, leandro.coelho@pucpr.br
Pontifical Catholic University of Parana - PUCPR, Production and Systems Engineering Graduate Program. Curitiba, Brazil

Abstract. *Identification and prediction of nonlinear dynamic systems is a difficult task. There are several reports in the literature about the utilization of different models of Artificial Neural Networks to approximate this class of systems. This work compares the application of two categories of ANNs in the work of modeling nonlinear dynamical systems and in the prediction of systems outputs: MLP - Multilayer Perceptron and GMDH - Group Method of Data Handling. GMDH networks neurons are composed of low order polynomials and are exempt from the use of complex activation functions which require big efforts from microprocessors to perform their calculations, hence tending to become an interesting option for engineering applications.*

Keywords: *neural network, dynamic system, prediction, gmdh, embedded system*

1 INTRODUCTION

The stability of nonlinear systems can be established for the most part only on a system-by-system basis and hence it is not surprising that design procedures that simultaneously meet the requirements of stability, robustness and good dynamical response are not currently available for large classes of such systems (Narendra and Parthasarathy, 1990). Neural network techniques have been found to be useful in highly uncertain, nonlinear, and complex systems. It is not uncomon to apply Artificial Neural Networks (ANNs) as approximation models of unknown nonlinearities. (Polycarpou, 1997). For example, optimization of classical Proportional-Integral (PI) controllers is difficult to perform when nonlinear and non-stationary processes are considered. ANNs have the ability to adapt their gains in order to minimize a cost function, what can take place in an *on-line* or *off-line* basis, i.e. before starting the system or during its operation (Grzesiak *et al.*, 2006).

This work compares the modeling and the prediction of dynamical systems using two different models of ANNs, namely: MLP - Multi-Layer Perceptron and GMDH - Group Method of Data Handling. MLP is a very known and one of the mostly used model of ANNs. The GMDH algorithm was developed and introduced by Alexey G. Ivakhnenko (Ivakhnenko, 1970, 1971) with the purpose of performing the approximation of the relationship among the inputs and the outputs of complex systems. The GMDH ANNs used in this work have been created and trained with software developed by the authors. The implemented algorithms are written in MATLAB's M-code (MathWorks, 2011) and are available on-line as an open source project (Braga, 2011).

The remaining of this article is structured as follows: Section 2 presents a brief explanation about the two network models; Section 3 details the dynamical systems models used in the experiments performed; Section 4 shows the results of the experiments and; Section 5 concludes the work.

2 NETWORK MODELS

2.1 MLP - Multilayer Perceptron

The MLP is a feed-forward model organized in one input layer, one or more hidden layers and an output layer, where the nodes in one layer are fully interconnected with the nodes in the following layer. Figure 1.a shows a conceptual example of an MLP network. Except on the input layer, nodes on all other layers are processing units in the form of Eq. 1, where: $\mathbf{X}_l(n)$ is the row vector with the inputs to the n^{th} neuron of layer l; $\mathbf{W}_l^T(n)$ is the column vector which contains the weights of that neuron, and $y_l(n)$ is the neuron output. Each neuron has one activation function, $\varphi(.)$, used to constrain the output into one interval. Normally, the activation function is in the form of a sigmoid as, for example, the logistic function (Eq. 2) or the hyperbolic tangent (Eq. 3). Different activation functions are allowed in neurons from different layers.

The number of output values produced by the ANN is determined by the number or neurons in the output layer.

$$y_l(n) = \varphi[\mathbf{X}_l(n)\mathbf{W}_l^T(n)] \tag{1}$$

$$\varphi(v) = \frac{1}{1+e^{-2v}} \tag{2}$$

$$\varphi(v) = \tanh(v) = \frac{e^{2v} - 1}{e^{2v} + 1} \tag{3}$$

The training process of the MLP is known as the error-backpropagation algorithm, consisting of two phases: feed-forward and error-backpropagation. In the feed-forward phase, one sample with one or more inputs, depending on the topology of the network, is introduced in the input layer and the output(s) of the network is(are) calculated by forwarding the outputs of the neurons on each layer as inputs to the neurons on the following layer, down to the output layer. The error-backpropagation phase is performed by first calculating the error between the desired output $y_o(n)$ of each n^{th} neuron on the output layer and the actual output $\hat{y}_o(n)$ of that neuron, as given by Eq. 4. Starting from the output layer and going backward in the direction of the first hidden layer, the weights are adapted by the delta-rule presented in Eq. 5 where η is a constant that determines the learning rate and δ is the learning gradient. The gradient is calculated using Eq. 6 in the output neurons or Eq. 7 in the hidden neurons (φ' is the first derivative of the activation function). In equation Eq. 7, $\delta_{l+1}(m)$ denotes the previously calculated gradient for the neuron m in layer $l+1$ and $w_{l+1,n}(m)$ is the weight applied in neuron m belonging to layer $l+1$ to the input coming from neuron n in layer l (Haykin, 1999).

$$e(n) = y_o(n) - \hat{y}_o(n) \tag{4}$$

$$\Delta w_l(n) = \eta \delta(n) x_i(n) \tag{5}$$

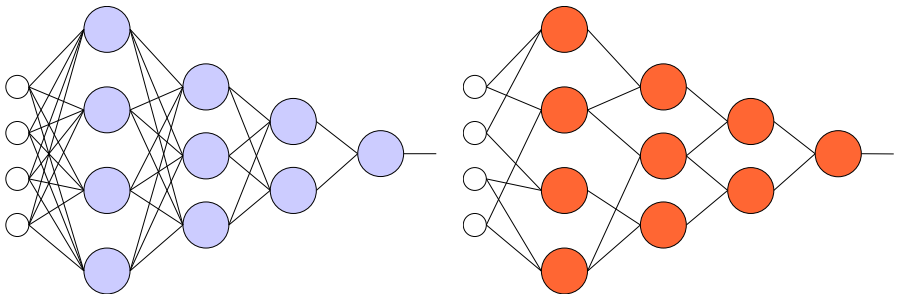
$$\delta_o(n) = e(n)\varphi'[v_o(n)] \tag{6}$$

$$\delta_l(n) = \varphi'_l[v_l(n)] \sum_m \delta_{l+1}(m) w_{l+1,n}(m) \tag{7}$$

2.2 GMDH - Group Method of Data Handling

The GMDH algorithm is based on an inductive self-organizing approach to the estimation of black box models with unknown relationships between variables (Vissikirsky and Stepashko, 2005).

Despite the resemblance to MLP, the structure in the GMDH network is not fully connected. Its nodes can be connected using different combinations, depending on choices regarding the architecture of the network. The mostly used style applies two-combinations of connections, as can be seen in Fig. 1.b.



(a) (b)

Figure 1. Comparison betwen (a) Multilayer Perceptron and (b) GMDH neural networks

Another difference between the ANN models is that the topology of a GMDH network is not defined by the designer, as it is in an MLP. Based on the number of inputs in each sample, the training algorithm instantiates a number of neurons on the first layer. The structure of the neurons used in this work is based on the work of Pham and Liu (Pham and Liu, 1994), where each GMDH neuron has two inputs. When training starts, the number of neurons created on the first layer is defined by the combination of the number of inputs to the network, n, taken two by two (see Eq. 8).

$$C_2^n = \frac{n!}{2(n-2)!} \tag{8}$$

The internal expression of one GMDH neuron is given by Eq. 9 (Pham and Liu, 1994) where x_1 and x_2 are the two inputs to the neuron and $[a_0, ..., a_5]$ are the coefficients or weights of the neuron. Equation 9 can be rewritten in the matrix form given by Eq. 10, where $\mathbf{x}$ is called the preprocessed inputs, expressed by $\mathbf{x} = [1, x_1, x_2, x_1^2, x_2^2, (x_1 x_2)]$, and $\mathbf{A} = [a_0, ..., a_5]^T$ (Pham and Liu, 1994).

$$y = a_0 + a_1 x_1 + a_2 x_2 + a_3 x_1^2 + a_4 x_2^2 + a_5 x_1 x_2 \tag{9}$$

$$y = \mathbf{x}\mathbf{A} \tag{10}$$

Letting $\mathbf{X}$ be a matrix containing all the preprocessed two-inputs samples to the neuron and $\mathbf{Y}$ a column vector with all the desired outputs corresponding to the samples, the relation among the weights $\mathbf{A}$, the inputs and the outputs of the neuron can be expressed by Eq. 11, where matrices $\mathbf{Y}$, $\mathbf{X}$, and $\mathbf{A}$ are of order N x 1, N x 6, and 6 x 1, respectively (the first element in each row of the X matrix is unity).

$$\mathbf{Y} = \mathbf{X}\mathbf{A} \tag{11}$$

The normal equations are formed by premultiplying both sides by the transpose of $\mathbf{X}$ (Eq. 12). Matrix $\mathbf{X}^T\mathbf{X}$ is a 6 x 6 matrix and the solution, shown in Eq. 13, is found by inverting this matrix (Ivakhnenko, 1971).

$$\mathbf{X}^T\mathbf{Y} = (\mathbf{X}^T\mathbf{X})\mathbf{A} \tag{12}$$

$$\mathbf{A} = (\mathbf{X}^T\mathbf{X})^{-1}\mathbf{X}^T\mathbf{Y} \tag{13}$$

After calculating the coefficients for all the neurons, those which produce the poorest results according to a previously chosen selection criterion must be removed from the layer. The next step is to create a new layer with the number of neurons defined by Eq. 8 replacing n by the number of the remained neurons in the current layer.

The overall training process consists of adding layers, calculating the weights and removing the poor performable neurons. The process stops when one layer remains with only one neuron or when a new layer added does not improve the previously selected performance criterion. In the case where the last layer does not improve performance, this layer is removed. In any case, if the last layer contains more than one neuron, that one which produces the best result, i.e., the one which mostly fits the selection criterion, is preserved. All other neurons on the last layer are removed, as so the neurons on the previous layers which do not contribute to the value of the output neuron.

3 DYNAMICAL SYSTEMS MODELS

The studies presented in this work have been performed using four examples of non-linear systems introduced in by Narendra and Parthasarathy (Narendra and Parthasarathy, 1990) which resemble the behavior of non-linear plants. In the following subsections, each plant is described in terms of its modeling expression and in terms of the inputs used for training and validating the ANNs.

Three different types of input data signals have been used for training the neural networks. The modeled plants receive sinusoid inputs, therefore, in each example, the ANNs were trained with data sets corresponding to this expected inputs to the plants. These data sets are referred as "original" inputs.

The second group of data sets is composed of random data in the range (maximum and minimum values) of the original data sets. They are referred by the term "random" inputs. The third group is formed by a sinusoid signal, as is the original data set, with the difference that it is created with random inputs given to the sinusoid function. It is referred as "random + sinusoid". Details on the formation of the training data sets are better explained in the following subsections.

One fourth input data set is created for each example in order to evaluate the performance of the ANNs. They are called "validation" inputs.

3.1 System 1

The plant represented in System 1 can show two different behaviors determined by Eq. 14 (Model 1) and by Eq. 15 (Model 2). The input data sets for training and selection based on Model 1 are determined by Eq. 16 and for Model 2, by Eq. 17, where $rnd(n, b_1, b_2)$ is a function that produces n random numbers in the range $[b_1, b_2]$. Equation 18 defines the values entered to $u_m(\cdot)$ to produce the sinusoid training data set. Equation 19 produces 10001 sorted random numbers in the range $[0, 10000]$ used into $u_m(\cdot)$ to generate the "sinusoid + random" training data. The data set i_v defined in Eq. 20 are used to validate the output of the ANNs.

$$y_1(k+1) = 0.3y_1(k) + 0.6y_1(k-1) + 0.6\ \sin[\pi u_1(k)] + 0.3\ \sin[3\pi u_1(k)] + 0.1\ \sin[5\pi u_1(k)] \tag{14}$$

$$y_2(k+1) = 0.3y_2(k) + 0.6y_2(k-1) + u_2^3(k) + 0.3u_2^2(k) - 0.4u_2(k) \tag{15}$$

$$u_1(k) = \begin{cases} \sin\left[2\pi i_t(k)/250\right] & \text{if sinusoid training;} \\ rnd(10001, -1, 1) & \text{if random input training;} \\ \sin\left[2\pi i_t^r(k)/250\right] & \text{if sinusoid + random input training;} \\ \sin\left[2\pi i_v(k)/250\right] & \text{if validating} \end{cases} \tag{16}$$

$$u_2(k) = \begin{cases} \sin\left[2\pi i_t(k)/250\right] + \sin\left[2\pi i_t(k)/25\right] & \text{if sinusoid training;} \\ rnd(10001, -1, 1) & \text{if random input training;} \\ \sin\left[2\pi i_t^r(k)/250\right] + \sin\left[2\pi i_t^r(k)/25\right] & \text{if sinusoid + random input training;} \\ \sin\left[2\pi i_v(k)/250\right] + \sin\left[2\pi i_v(k)/25\right] & \text{if validating} \end{cases} \tag{17}$$

$$i_t = [0, 0.1, 0.2, \ldots, 1000] \tag{18}$$

$$i_t^r = sort\,[rnd(10001, 0, 1000)] \tag{19}$$

$$i_v = [0, 0.05, 0.10, 0.15, \ldots, 1000] \tag{20}$$

3.2 System 2

In System 2, the plant to be modeled is represented by Eq. 21. The training and validating inputs are defined by Eq. 22. Equations 23, 24 and 25 define the values used in the "original", "sinusoid + random" and in the validating data sets produced by $u(\cdot)$.

$$y(k+1) = \frac{y(k)y(k-1)[y(k)+2.5]}{1+y^2(k)+y^2(k-1)} + u(k) \tag{21}$$

$$u(k) = \begin{cases} \sin[2\pi i_t(k)/25] & \text{if sinusoid training;} \\ rnd(1001, -2, 2) & \text{if random input training;} \\ \sin[2\pi i_t^r(k)/25] & \text{if sinusoid + random input training;} \\ \sin[2\pi i_v(k)/25] & \text{if validating} \end{cases} \tag{22}$$

$$i_t = [0, 0.1, 0.2, \ldots, 100] \tag{23}$$

$$i_t^r = sort\,[rnd(1001, 0, 100)] \tag{24}$$

$$i_v = [0, 0.01, 0.02, 0.03, \ldots, 100] \tag{25}$$

3.3 System 3

In System 3, the behavior of the plant is defined by Eq. 26 and the input data sets by Eq. 27. Equations 28, 29 and 30 describe the values used in the sinusoid functions to produce the training and validating data sets.

$$y(k+1) = \frac{y(k)}{1+y^2(k)} + u^3(k) \tag{26}$$

$$u(k) = \begin{cases} \sin[2\pi i_t(k)/25] + \sin[2\pi i_t(k)/10] & \text{if sinusoid training;} \\ rnd(1001, -2, 2) & \text{if random input training;} \\ \sin[2\pi i_t^r(k)/25] + \sin[2\pi i_t^r(k)/10] & \text{if sinusoid + random input training;} \\ \sin[2\pi i_v(k)/25] + \sin[2\pi i_v(k)/10] & \text{if validating} \end{cases} \tag{27}$$

$$i_t = [0, 0.1, 0.2, \ldots, 100] \tag{28}$$

$$i_t^r = sort\,[rnd(1001, 0, 100)] \tag{29}$$

$$i_v = [0, 0.01, 0.02, 0.03, \ldots, 100] \tag{30}$$

3.4 System 4

Despite the fact that the plant described by Eq. 31 behaves similarly as the previously described systems, this example shows a slightly different input scheme. Equation 32 defines the input values for training and validating, however the behavior of the sinusoid functions changes according to the index of the original sample, as it is explained by Eq. 33. Equations 34, 35 and 34 determine the values used in 32 for laying down the sinusoid input values.

$$y(k+1) = \frac{y(k)y(k-1)y(k-2)u(k-1)[y(k-2)-1]+u(k)}{1+y^2(k-2)+y^2(k-1)} \tag{31}$$

$$u(k) = \begin{cases} f\,[k, i_t(k)] & \text{if sinusoid training;} \\ rnd(801, -2, 2) & \text{if random input training;} \\ f\,[k, i_t^r(k)] & \text{if sinusoid + random input training;} \\ f\,[k, i_v(k)] & \text{if validating} \end{cases} \tag{32}$$

$$f(k, g(k)) = \begin{cases} \sin[2\pi g(k)/250] & \text{if } k \leq 500; \\ 0.8\,\sin[2\pi g(k)/250] + 0.2\,\sin[2\pi g(k)/25] & \text{if } k > 500 \end{cases} \tag{33}$$

$$i_t = [0, 1, 2, 3, \ldots, 800] \tag{34}$$

$$i_t^r = sort\,[rnd(801, 0, 800)] \tag{35}$$

$$i_v = [0, 0.1, 0.2, 0.3, \ldots, 800] \tag{36}$$

4 EXPERIMENTAL RESULTS

The values yielded by the systems represented in Eqs. 14, 15, 21, 26 and 31 are functions of the present input and of the present and past outputs. To perform the prediction tasks, the networks have been trained with delayed inputs but with non-modified targets. In the experiments carried out on this work, the order of prediction is of twenty samples of advance. Hence, the previously mentioned equations can be rewritten, respectively, as Eqs. 37, 38, 39, 40 and 41. The inputs given to the networks were the ones produced by the original systems and the expected outputs from the networks are the prediction of the systems' outputs twenty samples in advance.

To perform the validation of the networks designed for modeling tasks, the original expression used to produce the systems' outputs were substituted by the trained ANNs. The expected result is that the neural system be able to replace the original system and lay down similar results.

$$\begin{aligned} y_1(k+1) \quad = \quad & 0.3y_1(k-20) + 0.6y_1(k-21) + 0.6\ \sin[\pi u_1(k-20)] + 0.3\ \sin[3\pi u_1(k-20)] + \\ & 0.1\ \sin[5\pi u_1(k-20)] \end{aligned} \tag{37}$$

$$y_2(k+1) = 0.3y_2(k-20) + 0.6y_2(k-21) + u_2^3(k-20) + 0.3\ u_2^2(k-20) - 0.4\ u_2(k-20) \tag{38}$$

$$y(k+1) = \frac{y(k-20)y(k-21)[y(k-20)+2.5]}{1+y^2(k-20)+y^2(k-21)} + u(k-20) \tag{39}$$

$$y(k+1) = \frac{y(k-20)}{1+y^2(k-20)} + u^3(k-20) \tag{40}$$

$$y(k+1) = \frac{y(k-20)y(k-21)y(k-22)u(k-21)[y(k-22)-1] + u(k-20)}{1+y^2(k-22)+y^2(k-21)} \tag{41}$$

Taking into account the input schemes and the tasks for which the network has been trained, a number of experiments were performed. The parameters for each experiment are (a) ANN Type, which can be GMDH or MLP; (b) Simulation Task: prediction, denoted by the letter **P**, and modeling, denoted by **M**; (c) Input type: **S** - sigmoid, **R** - random and **SR** - sigmoid + random and; (d) Samples delay: 1 and 20 samples.

To evaluate the performance of the ANNs after training, they have been simulated with the validation data sets of each system. The performance measurement is based in the errors between the validation targets (expected value) and the network output. A success match, or a hit, was computed for every error smaller than 15%. The number of hits divided by the total number of targets denotes the performance value, in percentage, for each particular experiment. A secondary parameter of comparison is the training time.

4.1 Results for System 1

Table 1 shows the results for the 36 experiments performed with System 1. Since this system has two different models of behavior, Tab. 1 contains one column informing which model is analyzed (1 or 2). Figure 2 compares the performance of the MLP and of the GMDH implementations. Each mark on the x-axis represent a comparison between one GMDH experiment and one MLP experiment with similar parameters. On the first mark the values yielded by experiment 1 (GMDH) and experiment 19 (MLP) are shown, both aiming a one-sample prediction. Results for experiments 2 and 20, which target 20-samples prediction, are presented on the second mark. This pattern of analysis keeps on the rest of the marks, i.e., over one mark there are plots comparing two one-sample prediction and over the immediately following mark, plots comparing two 20-samples prediction. The last six nodes in each line are related to the experiments 13 through 18 and 31 through 36, which are "modeling" experiments.

Looking at Fig. 2 it is noticeable that, in general, the GMDH networks present better results than MLP networks for prediction tasks regarding System 1, except when comparing experiments 8 and 26, where the MLP version has a better performance. On the other hand, the GMDH networks could not implement the modeling tasks. Nonetheless, only two MLP networks, from experiments 31 and 34, presented serviceable modeling results. Is is also worth noticing that the the GMDH network in the modeling experiment 15 could not give forth convergent outputs during the validation process. Moreover, its MLP counterpart, in experiment 33, yielded the worse result among the MLP modeling experiments.

4.2 Results for System 2

Results for the 18 experiments performed with the ANNs trained with data from System 2 are presented in Tab. 2. The performance comparisons among MLP and GMDH networks are presented in Fig. 3.

GMDH implementations for System 2 did not converge in experiments 2, 7 and 8. Also, only GMDH experiments 1 and 3 produced usable results. With the exceptions of experiments 11 and 15, MLP implementations performed well and had exceptional outcomes in the modeling experiments 16, 17 and 18.

Table 1. Results for System 1

Exp. No.	Net Type	Model	Simulation	Input	Delay (samples)	Train Time(s)	Hits
1	GMDH	1	P	S	1	0.03	99.99%
2	GMDH	1	P	S	20	0.04	93.94%
3	GMDH	1	P	R	1	0.03	99.96%
4	GMDH	1	P	R	20	0.04	95.56%
5	GMDH	1	P	SR	1	0.04	98.74%
6	GMDH	1	P	SR	20	0.04	55.66%
7	GMDH	2	P	S	1	0.07	99.78%
8	GMDH	2	P	S	20	0.04	61.37%
9	GMDH	2	P	R	1	0.03	87.05%
10	GMDH	2	P	R	20	0.04	25.55%
11	GMDH	2	P	SR	1	0.04	30.24%
12	GMDH	2	P	SR	20	0.04	4.96%
13	GMDH	1	M	S	1	0.04	5.86%
14	GMDH	1	M	R	1	0.04	15.74%
15	GMDH	1	M	SR	1	0.04	0.00%
16	GMDH	2	M	S	1	0.04	7.42%
17	GMDH	2	M	R	1	0.04	2.85%
18	GMDH	2	M	SR	1	0.03	2.71%
19	MLP	1	P	S	1	397.56	100.00%
20	MLP	1	P	S	20	425.85	88.60%
21	MLP	1	P	R	1	400.83	53.80%
22	MLP	1	P	R	20	173.62	41.88%
23	MLP	1	P	SR	1	162.79	1.72%
24	MLP	1	P	SR	20	112.21	2.06%
25	MLP	2	P	S	1	258.63	99.98%
26	MLP	2	P	S	20	272.62	98.18%
27	MLP	2	P	R	1	460.27	29.20%
28	MLP	2	P	R	20	163.64	20.62%
29	MLP	2	P	SR	1	222.43	14.28%
30	MLP	2	P	SR	20	242.97	6.42%
31	MLP	1	M	S	1	414.42	81.72%
32	MLP	1	M	R	1	147.88	12.52%
33	MLP	1	M	SR	1	156.87	2.31%
34	MLP	2	M	S	1	413.19	89.22%
35	MLP	2	M	R	1	232.48	3.22%
36	MLP	2	M	SR	1	214.58	3.49%

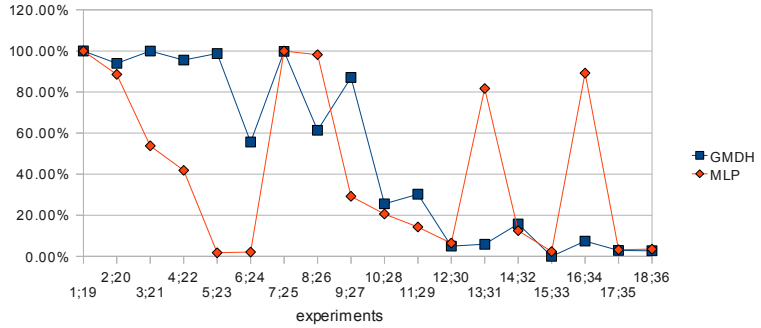


Figure 2. Comparing GMDH and MLP performances for System 1

4.3 Results for System 3

Both GMDH and MLP groups of networks trained for System 3 were successful for 1-sample prediction but neither was able to produce the 20-samples forecasting. Table 3 and Fig. 4 bring forward that both network types behave similarly when they are trained for the prediction task. Nonetheless, regarding the modeling task, MLP ANNs are effective while GMDH were not capable to approximate the system responses.

4.4 Results for System 4

Table 4 and Fig. 5 present the results for the training of the artificial neural networks for predicting and modeling the behavior of System 4. From Fig. 5 it is possible to see that the MLP types lay down somewhat better results. Experiments 6 and 15 are both unsuccessful and GMDH experiment 7 produces an unconvergent behavior during the validation phase.

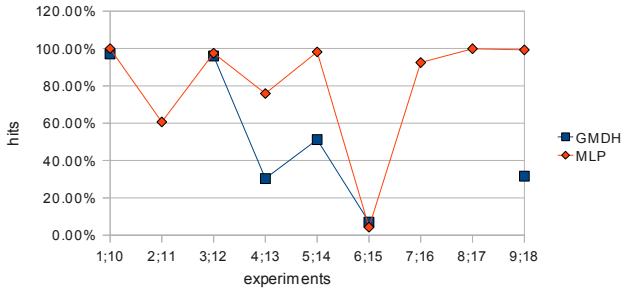


Figure 3. GMDH and MLP performances for System 2

Table 2. Results for System 2

Exp. No.	Net Type	Simulation	Input	Delay (samples)	Train Time (s)	Hits
1	GMDH	P	S	1	0.11	97.04%
2	GMDH	P	S	20	(-)	(-)
3	GMDH	P	SR	1	0.05	95.96%
4	GMDH	P	SR	20	0.06	30.32%
5	GMDH	P	R	1	0.06	51.20%
6	GMDH	P	R	20	0.02	7.01%
7	GMDH	M	S	1	0.11	(-)
8	GMDH	M	SR	1	0.07	(-)
9	GMDH	M	R	1	0.14	31.60%
10	MLP	P	S	1	72.24	99.92%
11	MLP	P	S	20	14.46	60.66%
12	MLP	P	SR	1	2.02	97.48%
13	MLP	P	SR	20	2.48	75.86%
14	MLP	P	R	1	9.13	98.16%
15	MLP	P	R	20	1.13	4.29%
16	MLP	M	S	1	66.6	92.48%
17	MLP	M	SR	1	66.54	99.84%
18	MLP	M	R	1	15.97	99.24%

Table 3. Results for System 3

Exp. No.	Net Type	Simulation	Input	Delay (samples)	Train Time (s)	Hits
1	GMDH	P	S	1	0.09	99.26%
2	GMDH	P	S	20	0.09	22.54%
3	GMDH	P	SR	1	0.06	97.46%
4	GMDH	P	SR	20	0.01	23.44%
5	GMDH	P	R	1	0.04	82.99%
6	GMDH	P	R	20	0.03	18.54%
7	GMDH	M	S	1	0.08	12.82%
8	GMDH	M	SR	1	0.03	4.22%
9	GMDH	M	R	1	0.01	3.68%
10	MLP	P	S	1	150.61	99.77%
11	MLP	P	S	20	129.7	9.68%
12	MLP	P	SR	1	67.65	99.93%
13	MLP	P	SR	20	2.29	27.07%
14	MLP	P	R	1	152.2	99.79%
15	MLP	P	R	20	37.27	42.18%
16	MLP	M	S	1	110.06	91.76%
17	MLP	M	SR	1	131.32	98.59%
18	MLP	M	R	1	3.87	86.71%

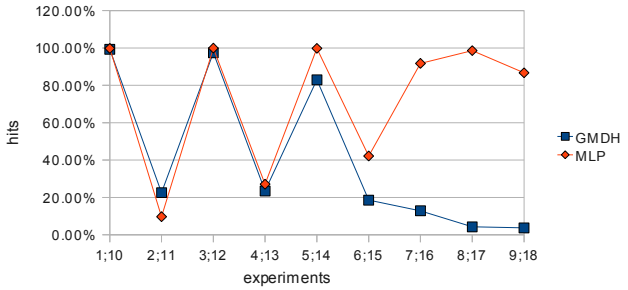


Figure 4. GMDH and MLP performances for System 3

Table 4. Results for System 4

Exp. No.	Net Type	Simulation	Input	Delay (samples)	Train Time (s)	Hits
1	GMDH	P	S	1	0.03	99.74%
2	GMDH	P	S	20	0.01	48.91%
3	GMDH	P	SR	1	0.02	56.78%
4	GMDH	P	SR	20	0.01	44.22%
5	GMDH	P	R	1	0.05	51.69%
6	GMDH	P	R	20	0.02	0.75%
7	GMDH	M	S	1	0.03	(-)
8	GMDH	M	SR	1	0.03	15.86%
9	GMDH	M	R	1	0.08	24.61%
10	MLP	P	S	1	77.39	99.93%
11	MLP	P	S	20	2.6	56.87%
12	MLP	P	SR	1	7.64	67.98%
13	MLP	P	SR	20	2.18	51.36%
14	MLP	P	R	1	1.79	79.43%
15	MLP	P	R	20	0.88	2.37%
16	MLP	M	S	1	67.3	99.93%
17	MLP	M	SR	1	64.09	78.25%
18	MLP	M	R	1	1.2	24.47%

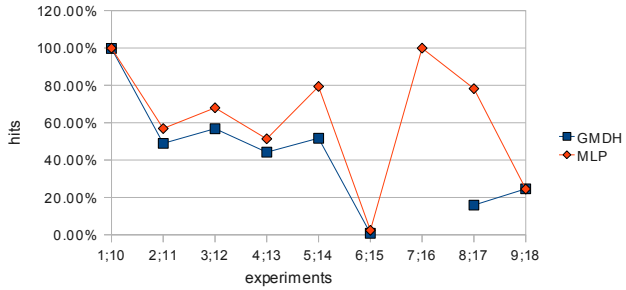


Figure 5. GMDH and MLP performances for System 4

5 CONCLUSION

This work presented the comparison among several implementations of artificial neural networks for the tasks of prediction and modeling of dynamical nonlinear systems. Two types of ANNs have been applied in the experiments: the well known Multilayer Perceptron (MLP) model and the Group Method of Data Handling (GMDH).

The analysis performed showed that the GMDH networks can be trained much faster than the MLPs but their overall performance have shown similar results regarding the prediction tasks. On the other hand, MLPs showed to be superior in terms of the modeling experiments. Despite these results, GMDH networks showed an interesting potential for the proposed application. Considering that neural networks of the GMDH type can be simpler than the MLPs in terms of computer resources for computation, and taking into account that there are reports of other works using GMDH for the approximation of complex nonlinear systems, it is plausible to state that it worths continuing to improve the training algorithms in order to apply this model in real world problems.

6 REFERENCES

Braga, A.L.S., 2011. "Gmdh-t-ann - gmdh type artificial neural network". URL `http://opengmdh.org/wiki/GMDH-T-ANN`.

Grzesiak, L.M., Meganck, V., Sobolewski, J. and Ufnalski, B., 2006. "On-line trained neural speed controller with variable weight update period for direct-torque-controlled ac drive". In *Power Electronics and Motion Control Conference, 2006. EPE-PEMC 2006. 12th International*. pp. 1127–1132. doi:10.1109/EPEPEMC.2006.4778553.

Haykin, S., 1999. *Neural Networks : A Comprehensive Foundation*. Upper Saddle River, EUA : Prentice-Hall.

Ivakhnenko, A.G., 1970. "Heuristic Self-Organization in Problems of Engineering Cybernetics". *Automatica*, Vol. 6, pp. 207–219.

Ivakhnenko, A.G., 1971. "Polynomial theory of complex systems". *Systems, Man and Cybernetics, IEEE Transactions on*, Vol. 1, No. 4, pp. 364–378. ISSN 0018-9472. doi:10.1109/TSMC.1971.4308320.

MathWorks, 2011. "MathWorks' website". URL `http://www.mathworks.com/`.

Narendra, K.S. and Parthasarathy, K., 1990. "Identification and Control of Dynamical Systems Using Neural Networks". *IEEE TRANSACTIONS ON NEURAL NETWORKS*, Vol. 1, No. 1, pp. 4–27.

Pham, D.T. and Liu, X., 1994. "Modelling and prediction using GMDH networks of Adalines with nonlinear preprocessors". *International Journal of Systems Science*, Vol. 25, pp. 1743–1759.

Polycarpou, M.M., 1997. "Stable Adaptive Neural Control Scheme for Nonlinear Systems". *IEEE TRANSACTIONS ON AUTOMATIC CONTROL*, Vol. 41, No. 3, pp. 447–451.

Vissikirsky, V.A. and Stepashko, V.S., 2005. "Growth Dynamics of Trees Irrigated with Wastewater: GMDH Modeling, Assessment, and Control Issues". *Instrumentation Science and Technology*, Vol. 2, pp. 229–249. doi:10.1081/CI-200048085.