Nonlinear Control of a DC-Motor Based on Radial Basis Function Neural Networks

Konstantinos Ninos, Charalampos Giannakakis, Ioannis Kompogiannis, Ilias Stavrakas, Alex Alexandridis
Department of Electronics
Technological Educational Institute of Athens
Athens, Greece
alexx@teiath.gr

Abstract— **This paper presents a nonlinear controller based on an inverse neural network model of the system under control. The neural controller is implemented as a Radial Basis Function (RBF) network trained with the powerful fuzzy means algorithm. The resulting controller is tested on a nonlinear DC motor control problem and the results illustrate the advantages of the proposed approach.**

Keywords- Neural Networks; Radial Basis Function; Fuzzy Means; Neural Controller; Intelligent Control

I. INTRODUCTION

Neural Networks (NNs) are a set of powerful mathematical tools that simulate the way that the human brain deals with information and the procedure of learning. NNs have the ability to identify and learn highly complex and nonlinear relationships from input-output data only, without the use of first principle equations describing the system. Neural networks are categorized to a variety of architectures, depending on the way the nodes of the neural network are interconnected and the calculations that each node performs.

Radial basis function (RBF) networks [1] constitute a special network architecture that presents some remarkable advantages over other neural network types including better approximation capabilities, simpler network structures and faster learning algorithms. Not surprisingly, RBF networks have found many applications in different scientific areas like pattern recognition [2], optimization [3] and control [4].

The use of neural networks in control applications - including process control, robotics, industrial manufacturing and aerospace applications, among others - has experienced rapid growth in the last decades [5]. The ability of neural networks to model an unknown system without any prior knowledge for the system, based solely on input-output data from it, can be exploited in the control of complex and nonlinear plants. Indeed neural networks can offer an accurate model of the system under control, which can then be inversed through an optimization procedure in order to produce a control law that drives the predictions of the model (and thus the plant itself – assuming that the model is accurate enough) to the set point [6]. This procedure is a special case of a more generic control scheme known as Model Predictive Control (MPC), that has found many successful applications [7].

Though the use of MPC methodologies offers several advantages, it involves solving online an optimization problem which in the case of nonlinear models can be rather computational demanding, while no optimality is guaranteed. This work presents a simpler implementation of a controller based on neural networks, where an RBF network learns directly the inverse law that governs the plant's behavior, thus predicting the current value of the manipulated variable that will drive the system to the set point value within the next discrete time steps.

The rest of this paper is organized as follows: In the next paragraph, we present the RBF neural network architecture and the fuzzy means training algorithm. In paragraph 3, we discuss the implementation of the RBF network as a neural controller and a case study where the proposed neural controller is applied to the control of a nonlinear DC motor. The paper concludes by outlining the advantages of the proposed approach.

II. THE RBF NEURAL NETWORK ARCHITECTURE

A. RBF Networks

An RBF network can be considered as a special three layer neural network, which is linear with respect to the output parameters after fixing all the radial basis function centers and nonlinearities in the hidden layer. The typical structure of an RBF network is shown in Fig. 1. The input layer distributes the N input variables to the L nodes of the hidden layer. Each node in the hidden layer is associated with a center, equal in dimension with the number of input variables. Thus, the hidden layer performs a nonlinear transformation and maps the input space onto a new higher dimensional space. The activity $v_l(\mathbf{x}_k)$ of the l^{th} node is the Euclidean norm of the difference between the k^{th} input vector and the node center and is given by:

$$v_l(\mathbf{x}_k) = \left\|\mathbf{x}_k - \hat{\mathbf{x}}_l\right\| = \sqrt{\sum_{n=1}^{N}(x_{k,n} - \hat{x}_{l,n})^2}\ , \quad k = 1,\ 2,\ \ldots, K \tag{1}$$

978-1-61284-922-5/11/$26.00 ©2011 IEEE

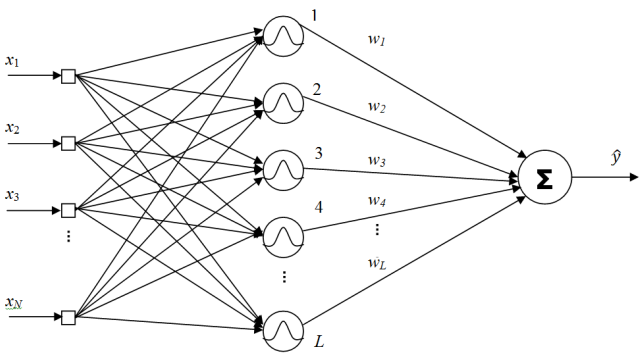

Figure 1. Radial Basis Function Network Architecture

where K is the total number of data, $\mathbf{x}_k^T = [x_{k,1},\ x_{k,2},\ \ldots,\ x_{k,N}]$ is the input vector and $\hat{\mathbf{x}}_l^T = [\hat{x}_{l,1},\ \hat{x}_{l,2},\ \ldots,\ \hat{x}_{l,N}]$ is the center of the l^{th} node.

The output function of the node is a radially symmetric function. A typical choice, which was also used in this work, is the Gaussian function:

$$f(v) = \exp\left(-\frac{v^2}{\sigma^2}\right) \tag{2}$$

where σ is the width of the node.

The final output $\hat{y}_k$ of the RBF network for the k^{th} data point is produced by a linear combination of the hidden node responses, after adjusting the weights of the network appropriately:

$$\hat{y}_k = \sum_{l=1}^{L} w_l f\left(v_l\left(\mathbf{x}_k\right)\right),\ k = 1, 2, \ldots, K \tag{3}$$

Standard approaches in RBF network training, decompose the problem in two steps: In the first step, the parameters of the basis functions of the nodes (hidden layer) are obtained using the *k*-means algorithm [8], which is an unsupervised clustering method. The second step involves the determination of the output-layer weights by linear least squares regression. However, the *k*-means algorithm has three major drawbacks:

- Several passes of all training examples are required. This iterated procedure increases the computational effort, especially when a large database is available.
- The number of hidden nodes must be predefined by the user. This means that a trial and error procedure must be followed in order to obtain the optimum number of hidden nodes.
- It depends on an initial random selection of centers, so that different sets of centers are obtained for different runs of the same network structure.

Recently, an innovative approach called the fuzzy means algorithm has been introduced in order to replace the *k*-means algorithm in the selection of the hidden layer nodes [9]. The fuzzy means algorithm has several advantages over the typical approach, including faster computational times and automatic determination of the size of the network, and has been used successfully in a number of applications including estimation of critical properties of materials [10], online system identification [11], automatic control of industrial processes [6], variable selection problems [12] etc. A brief discussion about the fuzzy means algorithm is given below, while the interested reader is referred to the original publications.

B. The fuzzy means algorithm

Assuming a system with N input variables, the universe of discourse (domain) of each variable x_i, $i \in 1,\ldots,N$ is evenly partitioned into c_i triangular fuzzy sets: $\{\mathrm{A}_{i,1}, \mathrm{A}_{i,2}, \ldots, \mathrm{A}_{i,c_i}\}$, $1 \le i \le N$. Each fuzzy set can then be fully described by its center element $a_{i,j}$ and half of the respective width $\delta a_{i,j}$.

By expanding the idea of fuzzy partitioning in the entire input space, we can dismember it into C fuzzy subspaces, where:

$$C = \prod_{i=1}^{N} c_i \tag{4}$$

Each fuzzy subset $\mathbf{A}^l$ is a combination of N particular fuzzy sets and is represented as:

$$\mathbf{A}^l = [A^l_{1,j_1},\ A^l_{2,j_2},\ \ldots,\ A^l_{N,j_N}] = \left\{ \begin{matrix} [a^l_{1,j_1},\ a^l_{2,j_2},\ \ldots,\ a^l_{N,j_N}], \\ [\delta\alpha_1,\ \delta\alpha_2,\ \ldots,\ \delta\alpha_N] \end{matrix} \right\} \tag{5}$$

After creating the fuzzy partition, the determination of the appropriate node centers is reduced to the problem of generating a set of fuzzy subspaces which uniformly cover the input data distribution. In order to solve this problem, we use the notion of the multidimensional membership function $\mu_{\mathbf{A}^l}(\mathbf{x}(k))$ of an input vector $\mathbf{x}(k)$ into $\mathbf{A}^l$ [13]:

$$\mu_{\mathbf{A}^l}(\mathbf{x}(k)) = \begin{cases} 1 - rd^l(\mathbf{x}(k)), & if \quad rd^l(\mathbf{x}(k)) \le 1 \\ 0, & otherwise \end{cases} \tag{6}$$

where $rd^l(\mathbf{x}(k))$ is the Euclidean relative distance between $\mathbf{A}^l$ and the input data vector $\mathbf{x}(k)$:

$$rd^l(\mathbf{x}(k)) = \frac{\left[\sum_{i=1}^{N}\left(a^l_{i,j_i} - x_i(k)\right)^2\right]^{1/2}}{\left[\sum_{i=1}^{N}\left(\delta a_i\right)^2\right]^{1/2}} \tag{7}$$

Obviously the fuzzy subspace that describes best an input vector is the one that corresponds to the smallest Euclidean relative distance. Thus the algorithm starts with the first data point $\mathbf{x}(1)$ and generates the first hidden node, whose center is the center location of the closest subspace. For all the remaining input examples $\mathbf{x}(k)$, k=2, 3, …, K, the method computes the Euclidean relative distances $rd^l(\mathbf{x}(k))$ between $\mathbf{x}(k)$ and the fuzzy subspaces, which have been generated so far. If the minimum of these distances is greater than unity then $\mathbf{x}(k)$ is not sufficiently described by any of the already created hidden nodes, and the algorithm generates a new node, by finding the closest subspace to $\mathbf{x}(k)$. If not, the algorithm proceeds to the next training example.

It must be emphasized that opposed to the k-means algorithm, the fuzzy means technique does not need the number of clusters to be fixed before the execution of the method. Moreover, due to the fact that it is a one-pass algorithm, it is extremely fast even if a large database of input-output examples is available.

III. Neural Control of a nonlinear DC Motor

A. The nonlinear DC motor

Consider the DC motor depicted in Fig. 2, which is described by the following nonlinear state equations, derived using fundamental electrical and mechanical laws [14]:

$$\begin{aligned} \frac{di_f}{dt} &= \frac{V_f - r_f i_f}{L_{FF}} \\ \frac{di_a}{dt} &= \frac{V_a - r_a i_a - L_{AF} i_f \omega_r}{L_{AA}} \\ \frac{d\omega_r}{dt} &= \frac{L_{AF} i_f i_a - B_L \omega_r}{J} \end{aligned} \quad (8)$$

where $\mathbf{x} = \left[\omega_r \;\; i_f \;\; i_a\right]^T$ is the state vector, $\mathbf{u} = \left[V_a \;\; V_f\right]^T$ is the input vector and $\mathbf{y} = \left[\omega_r \;\; i_f \;\; i_a\right]^T$ is the output vector. The notation is given in Table 1, together with values for each of the DC motor parameters. The system of nonlinear state equations is solved with numerical methods, thus providing a detailed dynamic simulation of the DC motor.

B. Neural controller configuration

The objective when controlling the DC motor depicted in Fig. 2, is to maintain the rotor angular speed ω_r at the desired set-point value, using as manipulated variable the armature voltage V_a. The field voltage V_f can be considered as a disturbance variable. Under this control scheme, a discrete neural controller should receive feedback for the current values of the state variables and the disturbance and produce the current value for the manipulated variable. The control configuration can be seen in Fig. 3.

TABLE I. Notation and Parameter values

Parameters	Symbol	Value
Rotor speed	ω_r	State variable
Armature current	i_a	State variable
Field current	i_f	State variable
Armature voltage	V_a	Manipulated variable
Field voltage	V_f	Disturbance
Armature resistance	R_a	10.5479Ω
Field resistance	R_f	320.6955Ω
Armature inductance	L_{AA}	$3.0948\text{x}10^{-6}H$
Field inductance	L_{FF}	$20.5245H$
Mutual inductance	L_{AF}	$2.6116H$
Inertia	J	$0.0015kg{\cdot}m^2$
Coefficient of load torque	B_L	$0.0042N{\cdot}m{\cdot}sec$

A simple way to design such a neural controller is to train a neural network to approximate the inverse law governing the system:

$$V_a(k) = NN\left(\omega_r(k), i_f(k), i_a(k), V_f(k), \omega_r(k+1)\right) \quad (9)$$

i.e. the neural network should predict the current value of the manipulated variable V_a using as inputs the current values of the state vector $\mathbf{x}$, the current value of the disturbance V_f and the next value of the controlled variable ω_r. Once the training has been completed, then the neural network can be used as a controller that will predict the current value of the manipulated variable V_a that will drive the DC motor to the set point value ω_{rSP} at the next discrete time step k+1. In order to make this prediction the neural controller will use as inputs the current values of the state vector, the disturbance and the set point:

$$V_a(k) = NN\left(\omega_r(k), i_f(k), i_a(k), V_f(k), \omega_{rSP}(k)\right) \quad (10)$$

C. Neural network model training

In order to generate data for training the RBF network, the DC motor was excited by changing randomly the input variables V_a and V_f every 1 second. The random changes in the two input variables followed a normal distribution with mean and variance values shown in Table 2. Using the described configuration, 20000 data points were collected from the DC motor. The data points were split in half into training and validation datasets. In order to find the optimum number of fuzzy sets, the fuzzy means algorithm was applied to partitions from 4 to 26 fuzzy sets.

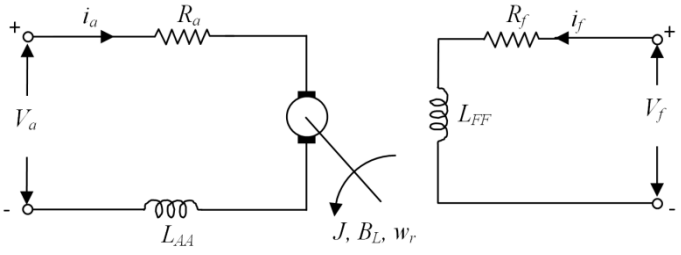

Figure 2. Nonlinear DC Motor

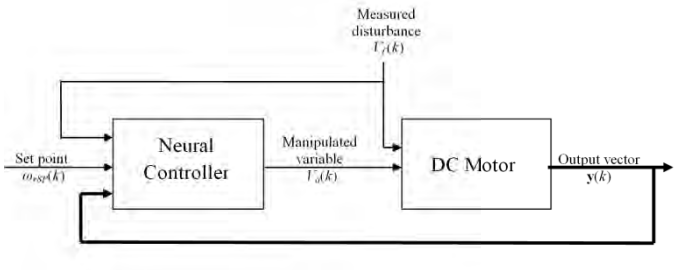

Figure 3. Neural control configuration

TABLE II. MEAN AND VARIANCE VALUES FOR THE INPUT VARIABLES

Input variable	Mean	Variance
V_a	$16V$	$12V$
V_f	$12V$	$4V$

The smallest Mean Absolute Relative Error % (MARE%) corresponded to a fuzzy partition of 16 fuzzy sets which resulted in an RBF network with 205 hidden nodes. Table 3 presents two statistical pointers for the best model that were calculated on the validation dataset.

D. Results and discussion

In order to evaluate the resulting neural control scheme, we have created two separate case studies. The objective was to test the controller performance in the tasks of set point tracking and disturbance rejection. For comparison purposes, we have also employed a standard PID controller tuned with the Simulink PID Tuner tool.

For the first case study of set point tracking, the disturbance V_f was assumed to be constant and equal to $12V$, while the set point changed randomly every 1.3 seconds. The random changes in the set point were chosen from a normal distribution with mean $16rad/s$ and variance $8rad/s$.

The results of the set point tracking case are depicted in Fig. 4, where the responses of the two control schemes are shown, together with the set point changes. It can be seen that both controllers manage to track accurately the set point changes. However, the neural controller response is much faster compared to the PID response. Moreover the neural controller avoids any oscillation, in contrast to the PID controller.

The second case study involves keeping the set point at a constant value while changing the value of the disturbance. To be more specific, for this case study the set point for the angular speed ω_r was set to a value equal to $18rad/s$, while the disturbance changed randomly every 1.5 seconds. The random changes in the disturbance were chosen from a normal distribution with mean $12V$ and variance $4V$.

TABLE III. STATISTICAL POINTERS ON THE VALIDATION DATASET

Statistical Pointer	Value
MARE %	0.041137
R^2	0.99997

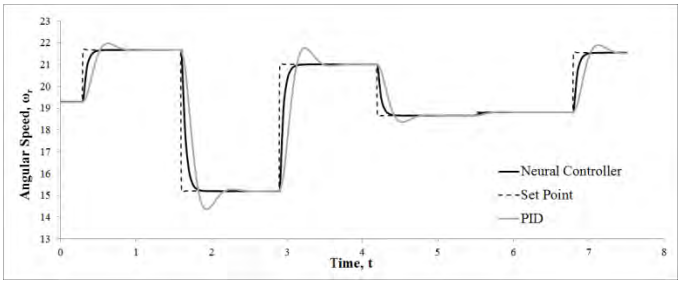

Figure 4. Set point tracking: Responses of the control schemes

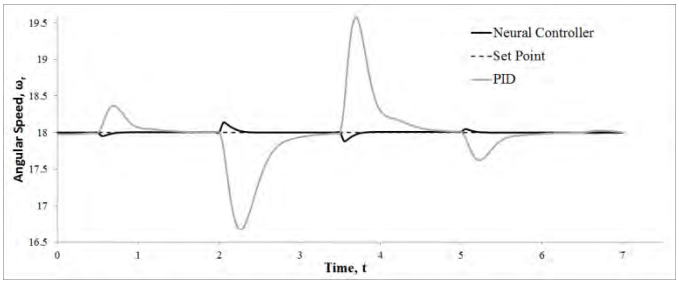

Figure 5. Disturbance rejection: Responses of the control schemes

The results of the disturbance rejection case are depicted in Fig. 5, where the responses of the two control schemes are shown. Evidently, the two controllers manage to reject the disturbance and maintain the DC Motor to the set point value. However, once more the neural controller response is obviously superior compared to the PID response.

IV. CONCLUSIONS

This paper presents a novel methodology for the control of nonlinear systems. The proposed control scheme is based on approximating the inverse plant dynamics with an RBF neural network which receives as inputs the state variables, disturbances and set point and produces as output the appropriate value for the manipulated variable. Training of the RBF network is performed using the fuzzy means algorithm which guarantees increased accuracy and lower computational times. The resulting controller was applied for controlling a nonlinear DC motor in two case studies involving set point tracking and disturbance rejection and the results have shown superior performance compared to a PID controller tuned with the Simulink PID Tuner tool.

REFERENCES

[1] J. Moody, C. Darken, "Fast learning in networks of locally-tuned processing units," Neural Computation, vol. 2, pp. 281-294, 1989.

[2] H. Sarimveis, P. Doganis, and A. Alexandridis, "A classification technique based on radial basis function neural networks," Advances in Engineering Software, vol. 37, pp. 218-221, 2006.

[3] P. Patrinos, A. Alexandridis, K. Ninos, and H. Sarimveis, "Variable selection in nonlinear modeling based on RBF networks and evolutionary computation," International Journal of Neural Systems, vol. 20, pp. 365-379, 2010.

[4] A. Alexandridis and H. Sarimveis, "Nonlinear Adaptive Model Predictive Control Based on Self-Correcting Neural Network Models," AICHE Journal, vol. 51, pp. 2495-2506, 2005.

[5] J. M. Zurada, Introduction to artificial neural systems. St. Paul, MN: West Publishing Company, 1992.

[6] A. Alexandridis, H. Sarimveis, "Nonlinear Adaptive Model Predictive Control Based on Self-Correcting Neural Network Models," AICHE Journal, vol. 51, pp. 2495-2506, 2005.

[7] S. J. Qin and T. A. Badgwell, "A survey of industrial model predictive control technology," Control Engineering Practice, vol. 11, pp. 733-764, 2003.

[8] C. Darken, J. Moody, "Fast Adaptive K-Means Clustering: Some Empirical Results," in IEEE INNS International Joint Conference On Neural Networks, San Diego, CA, 1990, pp. 233-238.

[9] H. Sarimveis, A. Alexandridis, G. Tsekouras, G. Bafas, "A fast and efficient algorithm for training radial basis function neural networks based on a fuzzy partition of the input space," Industrial and Engineering Chemistry Research, vol. 41, pp. 751-759, 2002.

[10] A. Afantitis, G. Melagraki, K. Makridima, A. Alexandridis, H. Sarimveis, O. Iglessi-Markopoulou, "Prediction of high weight polymers glass transition temperature using RBF neural networks," Journal of Molecular Structure: THEOCHEM, vol. 716, pp. 193–198, 2005.

[11] A. Alexandridis, H. Sarimveis, G. Bafas, "A new algorithm for online structure and parameter adaptation of RBF networks," Neural Networks, vol. 16, pp. 1003-1017, 2003.

[12] A. Alexandridis, P. Patrinos, H. Sarimveis, G. Tsekouras, "A two-stage evolutionary algorithm for variable selection in the development of RBF neural network models," Chemometrics and Intelligent Laboratory Systems, vol. 75, pp. 149-162, 2005.

[13] J. Nie, "Fuzzy control of Multivariable Nonlinear Servomechanisms with Explicit Decoupling Scheme," IEEE Transactions on Fuzzy Systems, vol. 5, pp. 304-311, 1997.

[14] P. C. Krause, Analysis of Electric Machinery. Singapore: MacGraw-Hill International Editions, 1987.