REPORT DOCUMENTATION PAGE

Form Approved OMB No. 0704-0188

1. AGENCY USE ONLY (Leave blank)	2. REPORT DATE	3. REPORT TYPE AND DATES COVERED
	25 May 1998	Final Report

4. TITLE AND SUBTITLE

Neural Networks for Robot Control

5. FUNDING NUMBERS

F61775-98-WE141

6. AUTHOR(S)

Prof. Chaiban Nasr

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)

Lebanese University-Faculty of Engineering, Section I
El Kobbeh
Tripoli North
Lebanon

8. PERFORMING ORGANIZATION REPORT NUMBER

N/A

9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)

EOARD
PSC 802 BOX 14
FPO 09499-0200

10. SPONSORING/MONITORING AGENCY REPORT NUMBER

SPC 98-4073

11. SUPPLEMENTARY NOTES

12a. DISTRIBUTION/AVAILABILITY STATEMENT

Approved for public release; distribution is unlimited.

12b. DISTRIBUTION CODE

A

13. ABSTRACT (Maximum 200 words)

This report results from a contract tasking Lebanese University-Faculty of Engineering, Section I as follows: The contractor will perform a simulating investigation of the use of artificial neural networks (ANN) in the control of robot manipulators and the application of this knowledge for the search of optimum architectures of the ANN.

14. SUBJECT TERMS

EOARD, Robotics, Neural Networks

15. NUMBER OF PAGES

15

16. PRICE CODE

N/A

17. SECURITY CLASSIFICATION OF REPORT	18. SECURITY CLASSIFICATION OF THIS PAGE	19. SECURITY CLASSIFICATION OF ABSTRACT	20. LIMITATION OF ABSTRACT
UNCLASSIFIED	UNCLASSIFIED	UNCLASSIFIED	UL

EUROPEAN OFFICE OF AEROSPACE RESEARCH AND DEVELOPMENT

Contract F61775-98-WE141

14 September, 1998

FINAL REPORT,

25 May, 1998

NEURAL NETWORKS FOR ROBOT CONTROL

Principal Investigator: Prof. Chaïban NASR

Lebanese University - Faculty of Engineering, Section I
Dept. of Electrical and Electronic Engineering
El Kobbeh, North
LEBANON

Phone: (961) 3 369245
Fax: (961) 6 385089
E-mail: cnasr@syfed.lb.refer.org

1-INTRODUCTION:

Perhaps the most innovative technical development over the last ten years in the field of control has been the introduction of artificial neural networks (ANN) methods for identification, modeling and control [1,2]. The basic concept of artificial neural nets stems from the idea of modeling individual brain cells/neurons in a fairly simple way, and then connecting these models in a highly parallel fashion to offer a complex processing mechanism which exhibits learning in terms of its overall nonlinear characteristics. Actual brain cells are not of one particular form. They are not all identical, whereas at present artificial neural networks tend to consist of one type of neuron. Further, the overall make up of a brain, in terms of connectivity and structure, is highly complex and not well understood, whereas artificial neural networks are generally well structured and simply coupled, thereby enabling the possibility of understanding their mode of operation.

In terms of a control systems environment, the majority of practical controllers actually in use are both simple and linear, and are directed towards the control of a plant which is either reasonably linear or at least linearisable.

However, it is sufficient to state that a neural network is usually a complex nonlinear mapping tool, and the use of such a device on relatively simple linear problems makes very little sense at all, being a case of over-skill [3]. The fact that neural networks have the capability of dealing with complex non-linearity in a fairly general way is an exciting feature. By their nature, nonlinear systems are non-uniform and invariably require custom designed control schemes to deal with individual characteristics. No general theory deals comprehensively with the wide range of nonlinear systems encountered, so an approach, namely, neural networks that can offer a fairly broad coverage are therefore extremely attractive. By viewing neural networks with control applications clearly in mind, a number of observations can be made [4], the first of these being perhaps the most significant:

(i) Neural networks can flexibly and arbitrarily map nonlinear functions. Such networks are best suited for the control of nonlinear systems.
(ii) Neural networks are particularly well suited to multivariable applications due to their ability to map interactions and cross-couplings readily whilst incorporating many inputs and outputs.
(iii) Networks can either be trained off-line and subsequently employed either on or off-line, or they can be trained on-line as part of an adaptive control scheme or simply a real-time system identifier. Understanding of a network's mode of operation within an adaptive controller is, at the present time, extremely limited.
(iv) Neural networks are inherently parallel processing devices which exhibit to an extent, at least, fault tolerant characteristics.

The goal of the present work is a simulating investigation of the use of artificial neural networks in the control of robot manipulators and the application of this knowledge for the search of optimum architectures of the ANN.

2 - 2D PLANAR ARM:

2-1 - Position of the problem:

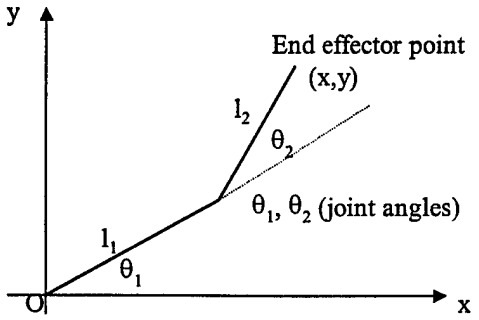

Fig. 1: - Idealized planar robot arm.

Our focus in this section is to provide the design fundamentals for artificial neural networks used in robotics systems. Trajectory control of robotic manipulators traditionally consists of following a preprogrammed sequence of end effector movements. Robot control usually requires control signals applied at the joints of the robot while the desired trajectory, or sequence of arm end positions, is specified for the end effector.

Figure 1 shows the geometry of an idealized planar robot manipulator with two degrees of freedom. The robot arm operates in a plane (2D). To make the arm move, the desired coordinates of the end effector point (x; y) are fed to the artificial neural network so that it generates the joint angles (θ_1, θ_2) for the motors that move the arms. To perform end effector position-control of a robotic manipulator, two problems need to be solved:

(i) Inverse kinematics problem: given the Cartesian coordinates of the end effector, specified either as a single point or as a set of points on a trajectory, joint angles or a set of joint angles need to be found.
(ii) Target position control: Given the final end effector position, a joint-angles sequence suitable for achieving the final position needs to be found.

The forward and inverse kinematics problems can be formulated respectively as:

$$\begin{aligned} x &= h(\theta) \\ \theta &= h^{-1}(x) \end{aligned} \qquad (1)$$

In those expressions θ and x are joint angles and end effector Cartesian coordinate vectors, respectively, which are defined as follows:

$$\theta = [\theta_1\ \theta_2]^T$$

$$x = [x_1\ x_2]^T \qquad x_1 = x \text{ and } x_2 = y$$

Even though the numerical solution of (1) can be found, this requires large, real-time computational resources. The neural network approach may be used here to solve both the problems of kinematics task formulation and the setting up of the equations. It also allows for circumvention of the computational complexity involved in their numerical solution.

2-2 - Multi-layer perceptrons method (MLP):

Multi-layer perceptrons [5, 6] provide one arrangement for neural network implementation, by means of nonlinear relationships between, firstly, the network inputs to outputs and, secondly, the network parameters to outputs. Such a network consists of a number of neuron layers, n, linking its input vector, u, to its output vector, y, by means of the equation:

$$y = \varphi_n(W_n\varphi_{n-1}(W_{n-1}\ldots\varphi_1(W_1u + b_1) + \ldots + b_{n-1}) + b_n)$$

In which W_i is the weight matrix associated with the layer, φ_i is a nonlinear operator associated with the ith layer and b_i indicate threshold or bias values associated with each node in the ith layer. The function φ_i is a sigmoid function for all n.

It is known in reality that real neurons, located in different areas of the nervous system, have different modes of behavior [7] ranging from Gaussian-like for visual needs to sigmoid for ocular motor needs. It is generally the case for artificial neural networks, however, that only one type of non-linearity is employed for a particular network, this linking in closely with the fact that each network is only employed for one particular task.

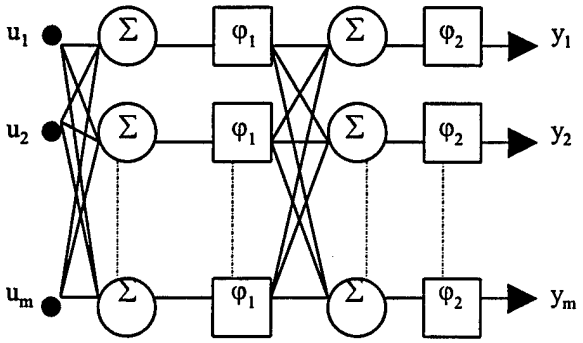

Fig.2: - A two-layer multi-layer perceptron.

Figure 2 shows a fully connected network in that all of the neuron outputs in one layer of the network are connected as inputs to the next layer. This is normal practice: however, it is quite possible for part-connectivity to be realized by connecting a group of outputs to only specific input. By this means sub-models can be formed within the overall MLP, and these can be particularly useful where a specific system characteristic is to be dealt with or inputs/outputs from the system can be categorized into certain types.

Whatever the network connectivity, key questions in the use of MLP are how many layers there should be and how many neurons there should be in each layer. Once these structural features of the network have been selected, it remains for the adjustable weights to be settled on such that the network is completely specified in terms of its functionality.

2-3 - Kinematics back propagation learning:

By far the most popular method employed for weight training in MLP neural networks is called back propagation [8]. In the standard feed-forward MLP network, back propagation solves the problem of missing information to the hidden layers, neither the input to nor the reference signals for the hidden layers are known. This solving problem is obtained by taking the inputs to the hidden layers as being the inputs to the first layers propagated through the network. The reference signals for the hidden layers are then obtained by error back propagation through the network. This is realized by obtaining the partial derivative of the squared error with respect to the parameters.

It is worth pointing out that the back-propagation algorithm has also been used for weight learning in feedback neural networks [9, 10], these being networks in which the network structure incorporates feedback, whereby the output of every neuron is fed back, in weighted form, to the input of every neuron. The architecture of such a network is inherently dynamic and realizes powerful capabilities due to its complexity.

Consider, as a starting point, a single neuron with output y_i; then

$$y_i = \varphi(x_i) = \frac{1}{1+\exp(-x_i)}$$

In which:

$$x_i = \sum_{i=1}^{m} w_{ij} u_j + w_0$$

In this expression w_0 is a bias term. If it is assumed that at an instant in time, for an input u_i the output y_i should be equal to the desired output y_d, then the squared error of the output signal is given by:

$$E_i = \frac{1}{2}(y_d - y_i)^2 = \frac{1}{2}e_i^2$$

And it is desired to minimize E_i by means of a suitable/best choice of the weighting coefficients w_{ij} .

Consider the problem of minimizing the scalar error function E (W), where W is a vector of weights to be adjusted by means of an interactive procedure generating a number of searching points, W(k), such that:

$$W(k+1) = \alpha(k).W(k) + \eta(k).d(k)$$

In this relation an initial set of weightings W(0) is made through prior knowledge, by a reasoned guess or even relatively randomly. The term $\alpha(k).W(k)$ represent a momentum term and $\alpha(k)$ is usually a positive number called the momentum constant. The term d(k) indicates the search direction, whereas $\eta(k)$ indicates the length of search step or the amount of learning to be carried out.

In this way, the weights associated with one neuron can be adjusted in order to minimize the squared error. The approach can then be extended in order to adjust all of the weights in the MLP network. So, overall, a set of inputs and desired output data values is used to train the entire network. The input set also realizing a corresponding set of network weights such that the error between the desired output signals and the actual network output signals is minimized in terms of the average overall learning points. The back propagation algorithm employs the steepest-descent method to arrive at a minimum of the mean squared error function. For one specific data pair, the error squared can be written as:

$$E_k = \frac{1}{2}\sum_{i=1}^{m}(y_{dk} - y_{ik})^2 = \frac{1}{2}\sum_{i=1}^{m} e_{ik}^2$$

Where m neurons are assumed to be present in the architecture of the neural network, and y_{ik} is the ith neuron's kth output value.

The global error is then found by minimizing E_k over all the data set. If the number of data values that are present is N, then we have:

$$E = \sum_{k=1}^{N} E_k$$

This error function can then be minimized in batch mode or recursively in an on-line manner.

The network is now fully trained on the data presented and can be employed with any further data, although it may be desirable to present the data again cyclically until the overall error falls below a previously defined minimum value, i.e. until the weights converge. An important feature then emerges in that the MLP network has the ability to generalize when it is presented with new data not previously dealt with.

2-4 - Numerical simulations:

The first numerical simulation aims to study the application of neural networks to an arc of circle, then to the entire circle and finally to the entire disk. The robot is required to learn Cartesian Coordinates (x;y) for the circular trajectory of radius r. A network with two inputs (θ_1 and θ_2), a variable number of hidden nodes, and two output neurons yielding outputs (x; y) has been selected to simulate the arc of a circle, all the circle and the entire disk.

The constraint on the circular trajectory, which needs to be learned, is:

$$\begin{aligned} x &= r\cos(\beta) \\ y &= r\sin(\beta) \end{aligned} \qquad (2)$$

In this expression the angle β is covering the first quadrant. The training has been performed for $l_1 = 3$, $l_2 = 2$ and $r = 5$ at 10 points of selected trajectory. Different networks has been trained using the error back-propagation technique with learning constant $\eta = 0.1$ and $\alpha = 0.5$ for the momentum constant. The training data of the trajectory with $r = 5$ contains angle β between 5° and 85°, spaced by 10°, and $\beta = 90°$.

The results of training are shown in Figure 5. The 2-8-2 multi-layer perceptron represent the optimal network. In this case, a minimum error of 10^{-4} is obtained after 250 cycles.

In the case of a circular trajectory, the constraint that must be learned, is given by equations (2) where angle β is covering the four quadrants. The training has been performed for $l_1 = 3$, $l_2 = 2$ and $r = 5$ at 40 points of the entire trajectory. Different networks has been trained using the error back-propagation technique with learning constant $\eta = 0.1$ and $\alpha = 0.1$ for the momentum

constant. The results of training are shown in Figure 6. The 2-8-8-2 multi-layer perceptron represent the optimal network. In this case a minimum error of 0.02 is obtained after 100 cycles.

Finally, in the case of a disk, the training has been performed for $l_1 = 3$, $l_2=2$ and $r = 5$ at 180 points of entire disk. Different networks has been trained using the error back-propagation technique with learning constant $\eta = 0.1$ and $\alpha=0.1$ for the momentum constant. The training data of the trajectory with a radius r =1:2:3:4 and 5 contains angle β between 5° and 355°, spaced by 10°, and $\beta=360°$.

The results of training are shown in Figure 7. The 2-8-8-2 multi-layer perceptron represent the optimal network. In this case a minimum error of 0.3 is obtained after 1000 cycles.

3 – 3D ARTICULATED ROBOT ARM:

3-1 - Position of the problem:

In this section, our focus is the study of an idealized 3D articulated robot arm by using the kinematics back-propagation method for the adjustment of the parameters. It consists also in searching for optimal architectures in this case.

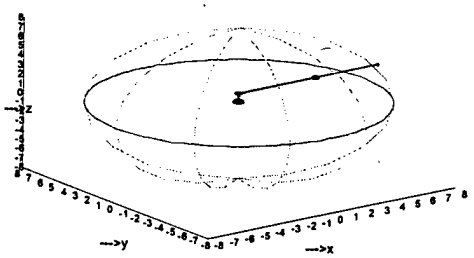

Fig. 3: Idealized articulated robot arm.

Figure 3 shows the geometry of the idealized 3D-articulated robot arm (RRR) with three degrees of freedom. The robot arm operates in the space. To make the arm move, the desired coordinates of the end-effector point (x, y, z) are fed to the artificial neural network so that it generates the joint angles (θ_1, θ_2, θ_3) necessary to control the motors that must move the arms.

3-2 – Numerical simulation:

The numerical simulation of the 3D articulated robot arm aims to study the application of neural networks to the entire sphere of radius 8. The training has been performed for $l_1 = 1$ (first link), $l_2 = 3$ (second link) and $l_3 = 4$ (third link). In this case, an important number of data for learning process is needed because we have to consider the entire sphere corresponding to the workspace of the robot arm. So, we have divided the workspace into five spheres of respective radius r = 8, 7, 5, 4 and 2. We have also considered different points verifying the equations:

$$x = r\cos(\theta)\cos(\varphi)$$
$$y = r\sin(\theta)\cos(\varphi)$$
And $$z = r\sin(\varphi)$$

Where: - φ varies from -60^0 to $+60^0$ with a step of 20^0.
- θ varies from 0^0 to 350^0 with a step of 10^0 for the spheres of radius 8 and 7
- θ varies from 0^0 to 340^0 with a step of 20^0 for the spheres of radius 5 and 4
- θ varies from 0^0 to 324^0 with a step of 36^0 for the sphere of radius 2

Also, we have introduced twelve singular points: (0,0,-8); (0,0,-7); (0,0,-6); (0,0,-5); (0,0,-4); (0,0,-3); (0,0,3); (0,0,4); (0,0,5); (0,0,6); (0,0,7); (0,0,8).

Different networks have been trained for the research of optimal architecture [11, 12] using the kinematics back-propagation error with different learning constants and different momentum constants. The results obtained in the numerical simulation are shown in figure 8. The learning constant $\eta = 0.1$ and the momentum constant $\alpha = 0.1$ gives the best convergence. The optimal network obtained is the (3-23-23-3) multi-layer perceptron with a minimal error of 0.9 obtained after 750 cycles.

4 – RBF NETWORKS APPLIED TO 2D PLANAR ARM:

Radial Basis Function (RBF) networks have been shown to function very efficiently, and to have the ability to model, in a fairly straight forward way, any arbitrary nonlinear system [13]. The output layer of such a network is merely a linear combination of the hidden layer signals, there being only one hidden layer. RBF networks thus offer considerable promise in terms of proving networks stability and robustness, mainly because of the way in which the network can be readily defined by a set of equations.

Let us consider an input vector u. An element of the output vector y is determined by the relation:

$$s(u) = \sum_{i=1}^{k} w_i R_i(u) + w_0$$

In which w_i (i = 1, 2, …, k) are the network weights, w_0 is a bias term and R_i are activation functions of the form:

$$R_i(u) = \Phi(\|u - c_i\|)$$

Where c_i (i = 1, 2, …, k) are a set of basis function centers and $\Phi(.)$ is the radial basis function which can be chosen in one of a number of ways [14]. In our research, we have chosen a Gaussian form, where:

$$\Phi(r_i) = \exp(-\frac{r_i^2}{2\sigma_i^2})$$

In which σ_i is a scaling parameter and $r_i = \|u - c_i\|$ represents the Euclidean distance from the input vector u to the center location c_i.
In their most usual form, as shown in figure 4, the centers, c_i, are variable values and function learning, as for the MLP case, is carried out by adjustment of the weights w_i .

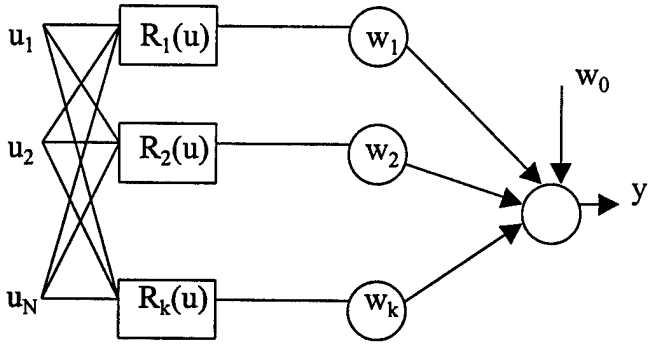

Fig. 4: - Radial Basis Function network.

4-1 – Determination of the centers and of the scaling parameters:

As was the case with MLP networks, a structural selection problem occurs with RBF networks, this time being dependent on how the centers are selected, both in terms of quantity and position. The choice of centers will, though, directly affect the quality of the function approximation obtained from the network, with a bottom line requirement that the number of centers employed must be such that they are sufficiently large in number to span the entire input domain [15]. However, the number of centers required tends to increase exponentially with regard to the input space dimension such that the RBF approach becomes very complicated with a high dimensional space [16].

. A direct trade-off exists between the number of RBF centers selected and the quality of function approximation obtained. For a low dimensional/ complexity problem, only a small number of centers are required. However, where the problem is of high dimension, a reasonable solution can be provided by the selection of a fixed, fairly low number of centers, it being realized that the approximation provided by the RBF network is going to be, at least, reasonable. Once the number of centers has been selected and fixed, it then remains for the actual center positions to be chosen, and this is perhaps better done as part of the network's training algorithm with the resultant values themselves being fixed. Allowing the center values to be adjusted on-line, although opening up the possibility of network adaptability, realizes extremely difficult problems, firstly with regard to proof of network stability, and secondly in terms of algorithmically controlling the nature and extent of center adaptation.

The main choices for center selection were taken as follows:

1- A random distribution of points chosen from within the convex hull of the input data set to be the RBF centers.
2- A random input vector u (t) chosen from within the convex hull of the input data set is presented to the network.

3- A research of the winning neuron which represents the minimum Euclidean distance between his center and the input vector.
4- A new center is obtained by the algorithm:

$$c_g(t+1) = c_g(t) + \eta(u(t) - c_g(t))$$

Where $c_g(t+1)$ is the new winning center, $c_g(t)$ is the previous winning center and u(t) represents the input data vector presented to the RBF network.

5- The algorithm is stopped when the Gaussian centers c_{gn} are well distributed in space of the input vectors, which means by minimizing the error E, given by:

$$E = \frac{1}{2}\sum_{n=1}^{K}\left\|u_n - c_{gn}\right\|^2$$

c_{gn} being the winning center related to the input vector u_n.

The scaling parameters are determined by means of the relation [17]:

$$\sigma_g^{\,2} = \frac{1}{K}\sum_{j=1}^{K}\left\|c_g - c_j\right\|^2$$

For a given K, and where c_j are the K-nearest centers to the winning center c_g.

The number of neurons K is chosen by simulation: first, we choose a small value of K and then we increase it until a minimum error is obtained.

4-2 – Results of simulations:

We have used the algorithm described in paragraph 4-1 with step learning η_1 equal to 0.01. For the research of the values of K (number of nearest centers to the winning center), we have considered a single center and we have increased the number by 2 until minimum error is obtained (figure 9). The values of the weights corresponding to the second layer of the RBF network were obtained by choosing the retro-propagation method with momentum η equal to 0.1 and α equal to 0.1. Different networks have been trained for the research of optimal architecture by changing the number of the hidden neurons. The optimal network obtained is a (2-52-2) that converges with a minimum error of 0.39 obtained after 300 cycles (figure 10).

5 - COMMENTS AND CONCLUSION:

Applications in new technologies such as robotics, manufacturing, space technology, and medical instrumentation, as well as those in older technologies such as process control and aircraft control, are creating a wide spectrum of control problems in which non-linearity, uncertainties, and complexity play a major role. For the solution of many of these problems techniques based on ANN are beginning to complement conventional control techniques [18], and in some cases they are emerging as the only viable alternatives. From a control theoretic point of view, ANN may be considered as tractable parameterized

families of nonlinear maps. As such they have found wide application in pattern recognition problems, which require nonlinear decision surfaces. With the introduction of dynamics and feedback, the scope of such networks as identifiers and controllers in nonlinear dynamical systems has increased significantly.

Our works aims to study the application of artificial neural networks (multi-layer perceptrons, MLP) for 2D planar robot arm and 3D articulated robot arm by using the kinematics back-propagation methods for the adjustment of parameters. It consists also in searching for optimal architectures in every case. We have implemented the different algorithms and we have obtained the optimal architecture in four different cases. The last case, concerning the simulation of the 3D articulated robot-arm, needs an important number of data for learning process because we have to consider the entire sphere corresponding to the workspace of the robot arm. We have used a technical partition of the workspace of the robot arm to build the necessary data for learning process in supervised neural network algorithms. Good results where obtained through the convergence of the algorithms used in the research of optimum architecture used in neural networks.

Another target was the study of the application of artificial neural networks (Radial Basis Functions, RBF) for the 2D planar robot arm by using the K-nearest centers algorithm combined with the back-propagation methods for the adjustment of parameters. It consists also in searching for optimal architectures. The results of the simulation show a very important influence of the choice of the number K, which represent the K-nearest centers.

Our objectives from this work are the real-time process (implementation on DSP of different architectures in neural networks and studying the behavior of all the process). So, now we are working on the research of optimal architectures in the case of dynamical robot arms.

6 - REFERENCES:

[1]: W.T.Miller, R.S.Sutton, and P.J.Werbos (eds.), Neural networks for control, MIT Press, Cambridge, MA (1990).

[2]: K.Warwick, G.W.Irwin, and K.J.Hunt (eds.), Neural networks for control and systems, Peter Peregrinus Ltd (1992).

[3]: K.Warwick, Neural networks for control: counter arguments, Proc. IEE Int. Conference Control 94, Warwick University, pp.95-99 (1994).

[4]: K.J.Hunt, D.Sbarbaro, R. Zbikowski and P.J.Gawthrop, Neural networks for control systems - a survey, Automatica, **28**, pp.1083-1112 (1992).

[5]: K.S.Narendra and K. Parthasarathy, Identification and control of dynamical systems using neural networks, IEEE Trans. on Neural Networks, **1**, pp.4-27 (1990).

[6]: A.U.Levin and K.S.Narendra, "Control of nonlinear dynamical systems using neural networks partII", IEEE Trans on neural networks, vol.7, pp. 31-42, n°1, January 1996.

[7]: D.H.Ballar, Cortical connections and parallel processing: structure and function. In Vision, brain and cooperative computation, M.Arbib and Hamson (eds.), pp.563-621, MIT Press, Cambridge, MA (1988).

[8]: D.E.Rumelhart and J.L.McClelland (eds.), Parallel distributed processing: explorations in microstructure of cognition, **1**: Foundations, MIT Press, Cambridge, MA (1986).

[9]: F.J.Pineda, Recurrent back propagation and dynamical approach to adaptive neural computation, Neural computation, **1**, pp.162-172 (1989).

[10]: K.S.Narendra and K.Parthasarathy, "Gradient methods for the optimization of dynamical systems containing neural networks", IEEE Trans. on Neural Networks, **2**, pp.252-262 (1991).

[11]: G.Mirchandani and Weicao, "On hidden nodes for neural nets", IEEE Trans on circuits and systems, vol.36 pp.661-664, n°5, May 1989.

[12]: Shih-Chi Huang and Yih-Fang Huang, "Bound on the number of hidden neurons in multi-layer perceptrons", IEEE Trans. On neural networks, vol.2, pp.47-55, n°1, January 1991.

[13]: F.Girosi and T.Paggio, Neural networks and the best approximation property, Biol. Cybernetics, 63, 169 – 176 (1990).

[14]: A.Cichocki and R.Unbehauen, Neural networks for optimization and signal processing, John Wiley and Sons (1993).

[15]: S.Chen, S.A.Billings, C.F.Cowan and P.M.Grant. Practical identification of NARMAX models using radial basis functions. Int. Journal of Control, 52, 1327 - 1350 (1990)

[16]: K.S.Narendra, Adaptive control of dynamical systems using neural networks in handbook of intelligent control, Van Nostrand (1994).

[17]: D.Hamad, C.Firmin, Introduction aux réseaux de neurones. France - 1997.

[18]: K.S.Narendra, "Neural Networks for Control: Theory and Practice", Proceedings of the IEEE, vol.84 n°10, (1996).

7-RESULTS OF THE SIMULATIONS

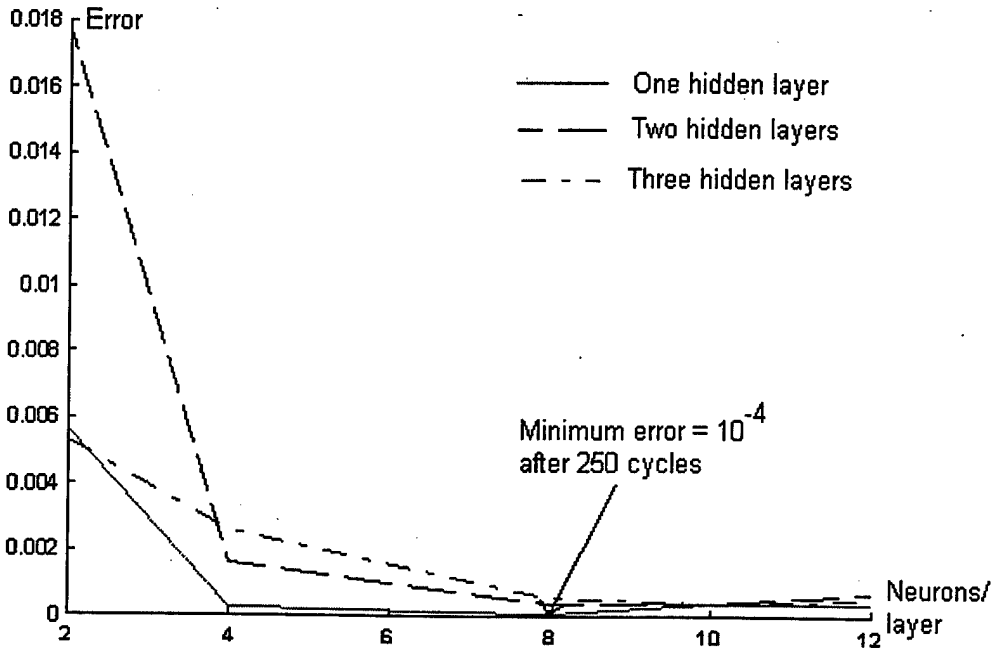

Fig.5:-Optimized hidden layers in the case of an arc of a circle of radius 5

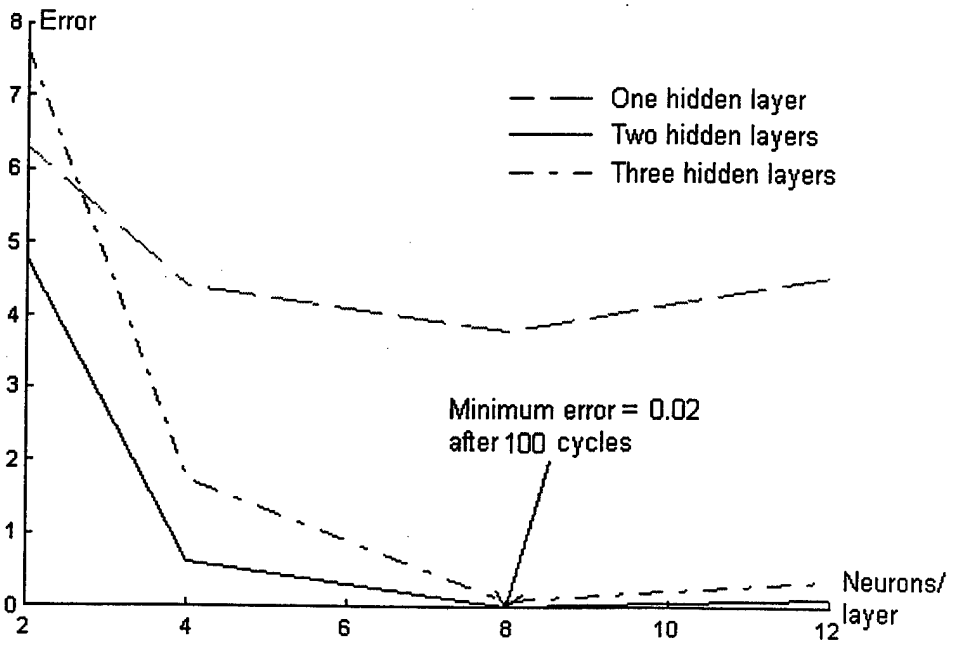

Fig.6:-Optimized hidden layers in the case of a circle of radius 5

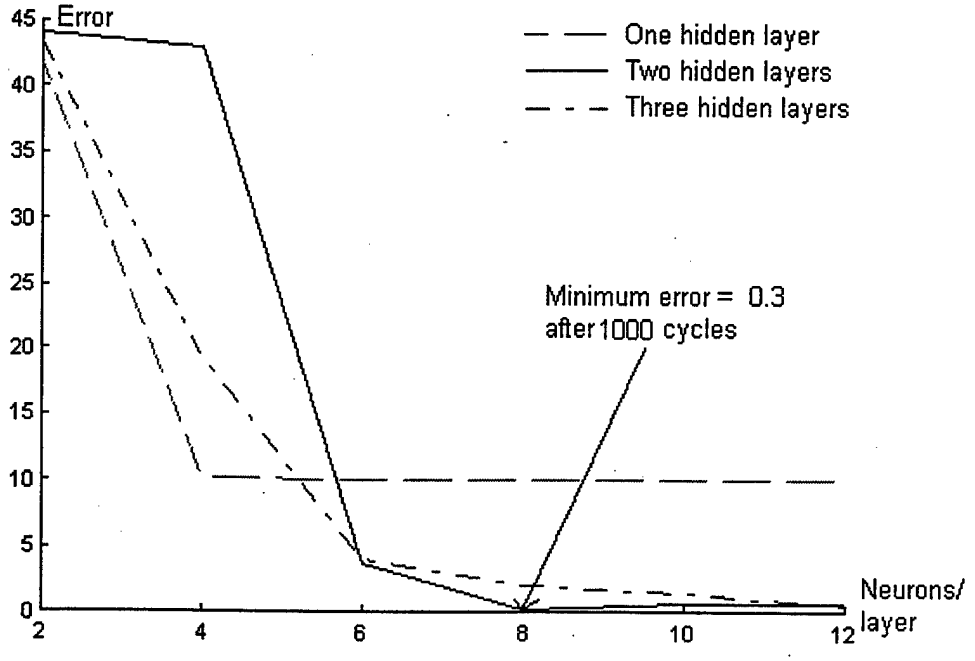

Fig.7:-Optimized hidden layers in the case of a disk of radius 5

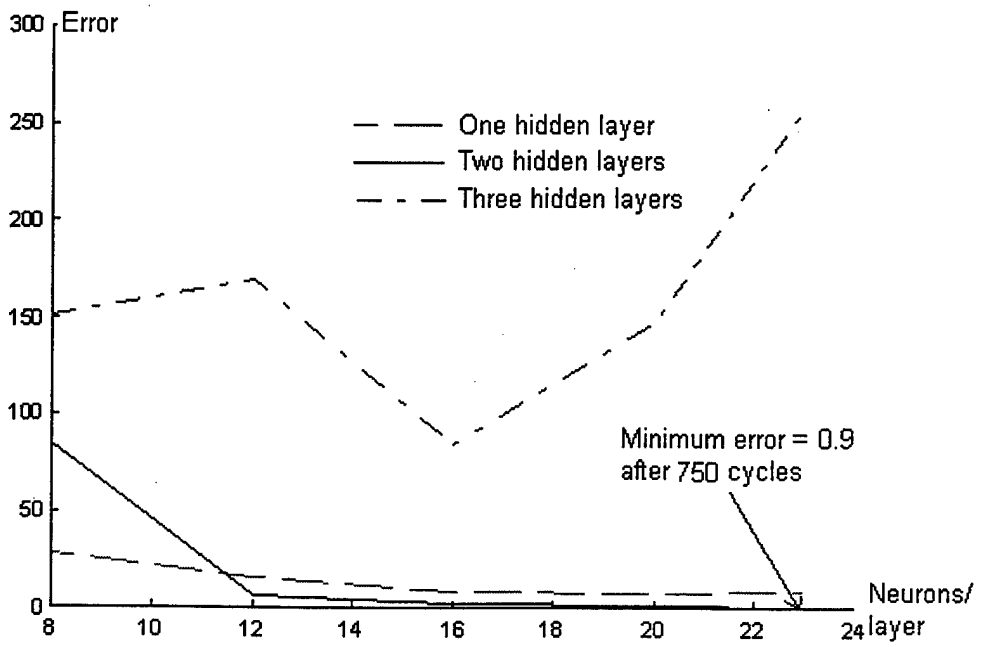

Fig.8:-Optimized hidden layers in the case of a sphere of radius 8

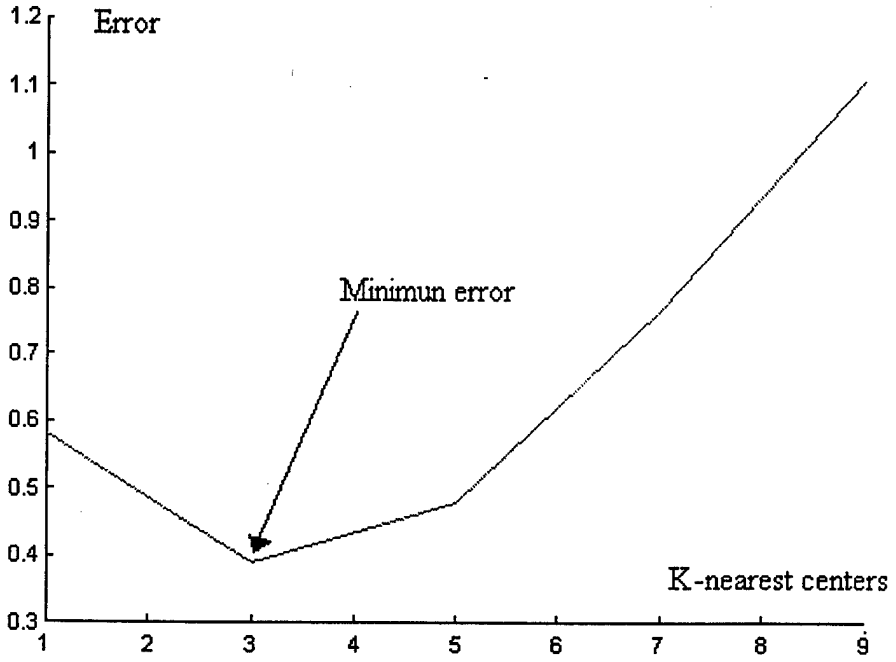

Fig.9:-Optimized number of the K-nearest centers to the winning center.

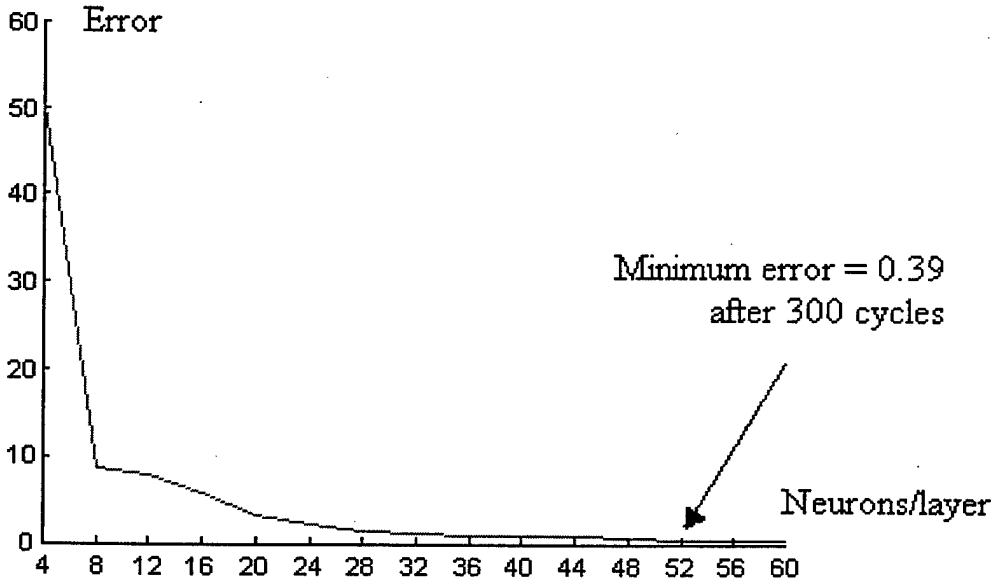

Fig.10:-Optimized hidden layers (RBF network) applied to a 2-D planar robot arm.