Journal of Engineering and Applied Sciences 1(3) : 262-267, 2006

Direct Adaptive Control of Nonlinear Systems Using Radial Basis Function Network

M. Bahita and K. Belarbi
Laboratoire d'automatique et de robotique, University of Constantine,
Route de Ain el Bey, Constantine, Algeria

Abstract: In this study we propose a direct adaptive control scheme for a class of nonlinear systems. The architecture is based on radial basis functions, RBF, network to construct the adaptive controller. The parameters of the adaptive controller are changed according to a law derived using Lyapunov stability theory and the center of the of the RBF are adapted using the k-means algorithm. Asymptotic stability is established with the tracking errors converging to a neighborhood of the origin. Computer simulations are presented to show the validity and the performance of the proposed method.

Key words: Radial Basis Function network, adaptive control, nonlinear systems

INTRODUCTION

A neural network, NN, is usually taken as a function approximator that approximates a given nonlinear function up to a small error tolerance. It has been proven that artificial neural networks can approximate any nonlinear functions to any desired degree[1-4].

In control engineering Multilayered perceptrons, MLP and radial basis functions, RBF, are the most widely used neural network. Generally this is done in an adaptive control framework. The first applications of NN in control did not include rigorous analysis of the stability[5,6]. However, in control systems, it is important to have design methodologies that provide proofs of stability for the system. Several neural network adaptive control algorithms have been proposed based on Lyapunov's stability theory[7-13].The advantage that the adaptive laws were guarantee the stability of systems.

Problem formulation: Consider a non linear system that can be transformed into the following form[14]:

$$\begin{aligned} \dot{x}_i &= x_{i+1}, \quad i = 1,...,n-1, \\ \dot{x}_n &= f(x) + b \cdot u, \\ y &= x_1, \end{aligned} \tag{1}$$

Where $x = [x_1, x_2,..., x_n]^T \in \Re^n$, u and $y \in R$ are the state variables, system input and output, respectively, f(x) is an unknown non linear smooth functions and b is a positive unknown constant. We assume that the state vector $x \in R^n$ is available for measurement.

Define: $e = [e_1, e_2,....,e_n]^T$ (2)

with $e_1 = x_1 - x_d^{(i-1)}$ i = 1....n

where, $x_d^{(i-1)}$ i=1,…n, represent the reference signal $y_d = x_d$ and its derivatives,

The control objective is to determine a state feedback control u =(u (x,θ) based on RBF network, and an adaptive law for adjusting the parameter vector θ of the network andsuch that

- The closed-loop system must be uniformly bounded.
- The tracking error should be as small as possible.

In order to achieve these objectives, we develop the method for a direct adaptive controller based on RBF network. As in the main trend of neural network adaptive control, we first propose an ideal control law u(t) based on feedback linearization and function of f(x) and b. An RBF controller is then used to approximate this control law. Assuming b to be non zero, the ideal control law u(t) is thus given by

From (1), we have $u^* = 1/b. [v-f(x)]$ (3)

Substituting (3) into (1), we can cancel the nonlinearities and obtain the simple input-state relation (multiple-integrator-form):

$$\begin{aligned} \dot{x}_n &= f(\mathbf{x}) + b.(\frac{1}{b}.[v - f(\mathbf{x})]) \\ \dot{x}_n &= f(\mathbf{x}) + v - f(\mathbf{x}) \\ \dot{x}_n &= v \end{aligned} \tag{4}$$

We then choose the artificial input v as a simple linear pole-placement controller that guarantees the stability of the overall system:

$$v = \dot{x}_d^{(n-1)} - k_0.e_1 - \sum_{i=1}^{n-1} k_i \dot{e}_i \quad (5)$$

with the k_i chosen so that the polynomial: $S_n + K_{n-1}S^{n-1} + ... + K_0 = 0$ has all its roots strictly in the left-half complex plane. Then the ideal control law is:

$$u^* = \frac{1}{b}(\dot{x}_d^{(n-1)} - k_0.e_1 - \sum_{i=1}^{n-1} k_i \dot{e}_i - f(\mathbf{x})) \quad (6)$$

based on (1) and (2) we have

$$\dot{e}_n = x_n - x_d^{(n-1)} = x_n - x_{nd} \quad (7)$$

then (6) becomes:

$$u^* = \frac{1}{b}(\dot{x}_{nd} - k_0.e_1 - \sum_{i=1}^{n-1} k_i \dot{e}_i - f(\mathbf{x})) \quad (8)$$

Substituting (8) into (1), we have:

$$\dot{x}_n = \dot{x}_{nd} - k_0.e_1 - \sum_{i=1}^{n-1} k_i \dot{e}_i \quad (9)$$

using (7) we can obtain:

$$\dot{e}_n + k_{n-1}\dot{e}_{n-1} + ... + k_1\dot{e}_1 + k_0.e_1 = 0 \quad (10)$$

this implies that $\lim_{t\to\infty} e(t) \to 0$ (exponentially stable dynamics). Since f(x) and b are unknown, the ideal control u* of (3) can not be implemented. Our purpose is to design an RBF with output u(x, θ) to approximate this ideal control.

The RBF adaptive controller

The RBF network: The RBF network can be considered as a two-layer network in which the hidden layer performs a fixed nonlinear transformation to map the input space into an intermediate space, then the output layer combines the outputs of the intermediate layer linearly as the outputs of the whole network. An RBF network with n input and a scalar output is represented in Fig. 1. Since the output depends linearly on the weights, then the training is simply a linear optimization problem[15].

More explicitly, the RBF network performs the transformation:

$f_r : R^n \to R$, with:

$$u(\mathbf{x},\theta) = \sum_{i=1}^{nr} \xi_i \theta_i \quad \theta^T \xi(\mathbf{x}) \quad \xi_i = \varphi(\|\mathbf{x} - c_i\|_2) \quad (11)$$

x is the input vector,

φ (.) is a non linear function, called radial basis function, θ_i are connections weights (parameters) between the hidden layer and the output layer, c_i are centres of basis functions, nr is the number of basis functions. We have considered here an RBF network with only one output.

The most used basis function is the Gaussian function.

$$(r) = \exp(-r^2/2.\sigma^2) \quad (12)$$

With $r = \|x-c_i\|_2$, c_i is the centre of $\varphi(r)$, σ is an associated constant to the function φ (r) and represents the width of the Gaussian function.

Training and centre placement in an RBF network: Usually the training procedure for RBF networks is divided in two stages: An unsupervised training for the centres adjustment of basis functions in the hidden layer, followed by a supervised training for the connections weights adjustment between the output layer and the hidden layer.

However, in control applications, online training is only concerned with the connections weights between the hidden and output layer and the centres are fixed off line. In this study, we propose to online adjust both the centres of the basis functions and the connections weights. The k-means algorithm and the recursive least square method are often used respectively for the centres adjustment and the adaptation of connections weights. Here, we will use the k-means algorithm for the centres adjustment.

Centres adjustment: The k_means algorithm is an unsupervised training method for data clustering[16]. It consists in dividing the input space into k classes as follows:

- Choose a number of classes (k basis functions in our case).
- Initialise the centres of the basis functions.
- Compute the Euclidean distances between the centres of each basis function and the input vector x.

$$dist(i) = \|x-c_i\|_2, \quad , \quad i=1 \text{ to } nr \quad (13)$$

then adjust the vector of centres c_i which corresponds to the minimum distance dist(j) = min $\|x-c_i\|_2$ using the following adaptation law:

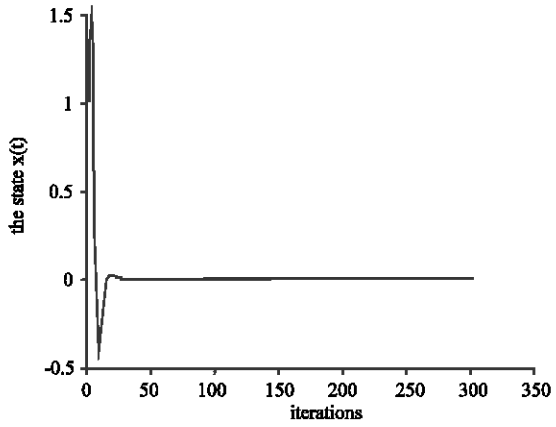

Fig. 1: The system state x (t) and the desired position

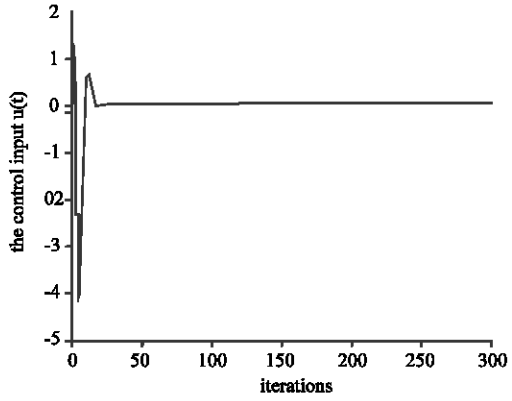

Fig. 2: The control input u (t)

$$c_j(t) = c_j(t-1) + \alpha(t).\left(\mathbf{x}(t) - c_j(t-1)\right) \quad (14)$$

where j is the index of the basis function which corresponds to the minimum Euclidean distance dist(j) and $\alpha(t)$ is a gain belonging to the interval [0 1] and which tends to zero as. One adaptation law for this parameter is:

$$\alpha(k) = \frac{\alpha(t-1)}{\sqrt{1+\mathrm{int}(t/nr)}} \quad (15)$$

where t and is the time, nr is the number of basis functions and int is the integer part of (t/nr) .

Weights adaptation: In the following we derive the adaptation law for the connections weights using Lyapunov synthesis approach.
From (1) we have:

$$\dot{x}_n = f(\mathbf{x}) + b.u(\mathbf{x},\theta) \quad (16)$$

now adding and subtracting b.u* to (16) we will have:

$$\dot{x}_n = f(\mathbf{x}) + b.u(\mathbf{x},\theta) + b.u^* - b.u^* \quad (17)$$

Substituting (8) into (17), we obtain:

$$\dot{x}_n = f(\mathbf{x}) + b.u(\mathbf{x},\theta) - b.u^* + (\dot{x}_{nd} - k_0.e_1 - \sum_{i=1}^{n-1} k_i \dot{e}_i - f(\mathbf{x})) \quad (18)$$

thus:

$$\dot{x}_n - \dot{x}_{nd} + k_0.e_1 + \sum_{i=1}^{n-1} k_i \dot{e}_i = b(u(\mathbf{x},\theta) - u^*) \quad (19)$$

Leading to the error system:

$$\dot{\mathbf{e}} = \mathbf{A}_c\mathbf{e} + \mathbf{B}_c(u(\mathbf{x},\theta) - u^*)) \quad (20)$$

with

$$\mathbf{A}_c = \begin{bmatrix} 0 & 1 & 0 & \dots & 0 & 0 \\ 0 & 0 & 1 & \dots & 0 & 0 \\ & & & & & \\ 0 & 0 & 0 & \dots & 0 & 1 \\ -k_0 & -k_1 & -k_2 & \dots & -k_{n-2} & -k_{n-1} \end{bmatrix} \quad (21)$$

$$\mathbf{B}_c = \begin{bmatrix} 0 \\ 0 \\ . \\ 0 \\ b \end{bmatrix}$$

Let's now study the stability of the system in order to develop an adaptive law to adjust the parameter vector θ of the RBF controller.
From (11) we have :

$$u(\mathbf{x},\theta) = \text{?}^T\xi(\mathbf{x}) \quad (22)$$

From (20) ,the error equation can be rewritten as

$$\dot{\mathbf{e}} = \mathbf{A}_c\mathbf{e} + \mathbf{B}_c(u(\mathbf{x},\theta) - u^*(\mathbf{x},\theta^*)) \quad (23)$$

Where θ is the optimal parameter vector corresponding to the optimal control signal $u^*(x,\theta^*)$ and

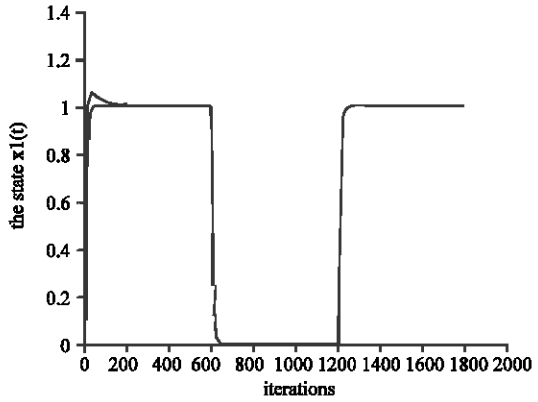

Fig. 3: The system state $x_1(t)$ and the desired position

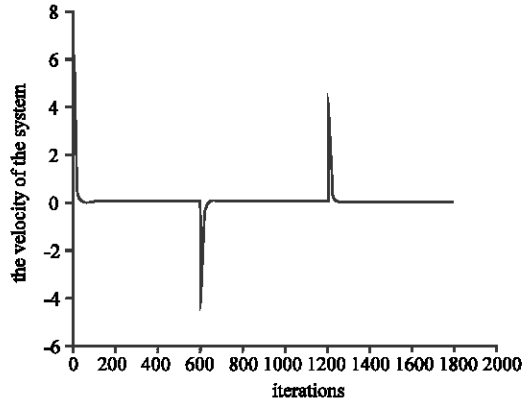

Fig. 4: The control input u(t)

θ^* is the ideal parameter vector corresponding to the ideal control signal $u^*(x,\theta^*)$

We can write: $u(x,\theta) = \theta^T .\xi(x)$ $u^*(x,\theta^*) = \theta^{*T}\xi(x)$ let $\varphi = \theta-\theta^*$ we obtain:

$$\dot{\mathbf{e}} = \mathbf{A}_c\mathbf{e} + \mathbf{B}_c\varphi^T\xi(\mathbf{x}) \tag{24}$$

Define the Lyapunov function candidate:

$$\mathbf{V} = \frac{1}{2}\mathbf{e}^T\mathbf{P}\mathbf{e} + \frac{b}{2.\gamma}\varphi^T\varphi \tag{25}$$

γ is a positive constant and P is a solution of the Lyapunov Eq:

$$\mathbf{A}_c^T\mathbf{P} + \mathbf{P}\mathbf{A}_c = -\mathbf{Q} \text{ with } Q > 0. \tag{26}$$

Differentiate V with respect to time:

$$\dot{\mathbf{V}} = \frac{1}{2}\dot{\mathbf{e}}^T\mathbf{P}\mathbf{e} + \frac{1}{2}\mathbf{e}^T\mathbf{P}\dot{\mathbf{e}} + \frac{b}{2\gamma}\dot{\varphi}^T\varphi + \frac{b}{2\gamma}\mathbf{f}^T\dot{\varphi} \tag{27}$$

using (24) and (26), we have :

$$\dot{\mathbf{V}} = -\frac{1}{2}\mathbf{e}^T\mathbf{Q}\mathbf{e} + \mathbf{e}^T\mathbf{P}\mathbf{B}_c\varphi^T\xi(\mathbf{x}) + \frac{b}{\gamma}\varphi^T\dot{\varphi} \tag{28}$$

Let P_n be the last column of P and using (21) we obtain:

$$\mathbf{e}^T\mathbf{P}\mathbf{B}_c = \mathbf{e}^T\mathbf{P}_n b \tag{29}$$

Substituting (29) into (28), we have

$$\dot{\mathbf{V}} = -\frac{1}{2}\mathbf{e}^T\mathbf{Q}\mathbf{e} + \frac{b}{\gamma}\mathbf{f}^T(\gamma.\mathbf{e}^T\mathbf{P}_n\xi(\mathbf{x}) + \dot{\varphi}) \tag{30}$$

in order to make $V < 0$, setting the second term of V equals to zero, i.e:

$$\frac{b}{\gamma}\varphi^T(\gamma.\mathbf{e}^T\mathbf{P}_n\xi(\mathbf{x}) + \dot{\varphi}) = 0 \tag{31}$$

recalling that $\varphi = \theta-\theta^* = \theta$, from (31) we obtain the adaptation law

$$\dot{\theta} = -\gamma.\mathbf{e}^T\mathbf{P}_n\xi(\mathbf{x}) \tag{32}$$

which guarantees $\dot{\mathbf{V}} = -\frac{1}{2}\mathbf{e}^T\mathbf{Q}\mathbf{e} \le 0$.

Design of the direct adaptive RBF controller: The design of the RBF adaptive controller can be summarized in the following steps

Step 1 Off line computations:

- Define the number of basis functions with centres uniformly cover the domain of data variation.
- Specify the parameters k_i such that all roots of $S^n + K_{n-1}S^{n-1} + ... + K_1.S^1 + K_0 = 0$, are in the open left-half plane.
- Specify a positive definite n x n matrix Q.
- Solve the Lyapunov Eq. (26) to obtain a symmetric p>0.

Step 2: On-line adaptation: Apply the feedback control (11) to the plant (1).

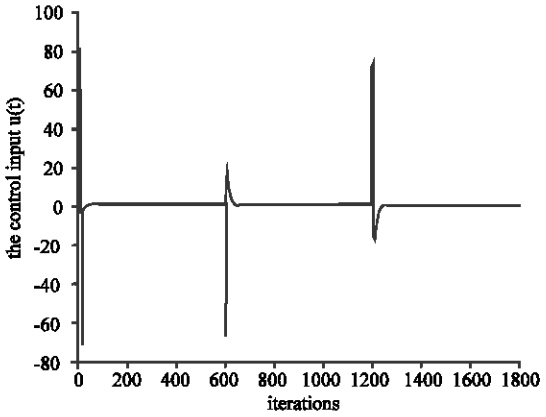

Fig. 5: The velocity of the system $x_2(t)$ and the desired velocity

Use the adaptive law (32) to adjust the connections weights and the k means algorithm to adjust the center of the RBF's.

RESULTS

Example 1: In this example, we apply the anddirect adaptive controller RBF and to regulate to the origin an unstable system[17]:

$$\dot{x}(t) = \frac{1 - e^{-x(t)}}{1 + e^{-x(t)}} + u(t) \qquad (33)$$

From (33) we have: $x(t) = 1\text{-}e^{-x(t)}/1+e^{-x(t)} > 0$ for x(t) and $1\text{-}e^{-x(t)}/1+e^{-x(t)} < 0$ forx(t) < 0.

We set γ=2.5 and $k_0 = 3$ in order to have all roots of $S+k_0 = 0$ are in the open left-half plane and with Q = 12 we can obtain: $A_c = -k_0 = -3$ (see 20) and p=2.

The RBF network comprises five radial basis functions. The parameters θ_i are randomly initialised andin the interval [0 1]. The andcentres and of the basis functions are uniformly distributed in the interval [-2 2]. The initial condition is x(0) =1. Fig. 1 shows the system state x(t). We see from this figure that the proposed direct adaptive control could regulate the plant to the origin and Fig. 2 shows the control input u(t).

Example 2: In this example, we consider a two dimensional non linear system[12]:

$$\dot{x}_1 = x_2$$
$$\dot{x}_2 = 4.\left(\frac{\sin(4.\pi.x_1)}{\pi.x_1}\right).\left(\frac{\sin(\pi.x_2)}{\pi.x_2}\right)^2 + .u. \qquad (34)$$

We apply the anddirect adaptive RBF controller and to control the system state $x_1(t)$ to track a desired trajectory which is specified as the output of a third order, critically damped system with a bandwidth of 10 rad/s driven by a unity amplitude, 0.5 mean, 0.4 Hz. square wave.

We choose γ =75, k_0 =2 and k_1 =3 in order to have all roots of $S^2+k_1.S+k_0 = 0$ in the open left-half plane, Q =diag(8,8) >0, then by solving (26) we can obtain:

$$P = \begin{bmatrix} 10 & 2 \\ 2 & 2 \end{bmatrix} \qquad (35)$$

The RBF network has five radial basis functions. The parameters θ_i are initialised to zero, andthe andcentres and of the basis functions are uniformly distributed in the interval [-1 2]. All initial conditions are taken as 0. Fig. 3 shows the system state $x_1(t)$ and the desired position. We see from this figure that the system state $x_1(t)$ tracks the desired trajectory. Fig. 4 shows the control input u(t). Fig. 5 shows the velocity of the system $x_2(t)$ and the desired velocity.

CONCLUSION

In this study, the RBF(Radial Basis Function) neural network system is used in the direct adaptive RBF controller. The major advantage is that the accurate mathematical model of the system is not required to be known. The proposed method can guarantee the stability of the resulting closed-loop system in the sense that all signals involved are uniformly bounded. Finally, we use the direct adaptive RBF controller to control two nonlinear systems.

REFERENCES

1. Chen, T.P. and H. Chen, 1995. Approximation capability to functions of several variables, nonlinear functionals and operators by radial basis function neural networks, IEEE Trans. Neural Networks, 6: 904-910.
2. Poggio, T. and F. Girosi, 1990. Networks for approximation and learning, Proc. IEEE, 78: 1481-1497.
3. Funahashi, K.L., 1989. On the approximate realization of continuous mapping by neural netorks. Neural Networks, 2: 183-192.
4. Hornik, K. M. Stinchombe and H. White, 1989. Multilayer feedforward networks are universal approximators. Neural netwoks, 2: 1083-1112.
5. Psaltis, D. A. Sideris and A. Yamamura, 1988. A multilayered neural network controller. IEEE Control Systems Magazine, 8: 17-21.
6. Narendra, K.S. and K. Parthasarathy, 1990. Identification and control of dynamic systems using neural networks, IEEE Trans. Neural Networks, 1: 4-17.

7. Chen, F.C. and C.C. Lin, 1994. Adaptively controlling nonlinear continuous time systems using multilayer neural networks. IEEE Transactions on Automatic Control, 39: 1306-1310.
8. Chen, F.C. and H.K. Khalil, 1995. Adaptive control of a class of non linear discrete-time systems using neural networks. IEEE Transactions on Automatic Control, 40: 791-801.
9. Ge, S.S. Hang and T. Zhang, 1999. Adaptive neural network control of nonlinear systems by state and output feedback. IEEE Trans. on Systems, Man and Cybernetics-part B: Cybernetics, 29: 6.
10. Lewis, F.L., A. Yesildirek and K. Liu, 1996. Multilayer neural net robot controller with guaranteed tracking performance. IEEE Transactions on Neural Network, 7: 388-399.
11. Polycarpou, M.M. and M. Mears, 1998. Stable adaptive tracking of uncertain systems using nonlinearly parameterized online approximators. Intl. J. Control, 70: 363-384.
12. Sanner, R.M. and J.E. Slotine, 1992. Gaussian networks for direct adaptive control. IEEE Transactions on Neural Networks, 3: 837-863.
13. Zhang, T., S.S. Ge and C.C. Hang, 1999. Design and performance analysis of a direct adaptive controller for nonlinear systems. Automatica, 35: 1809-1817.
14. Isidori., A., 1989. Non Linear Control Systems. (2nd Edn.) Berlin: Springer.
15. Haykin, S., 1994. Neurazl networks, A Comprehensive foundation, Prentice Hall.
16. Zheru Chi and Hong Yan, 1995. Image segmentation using fuzzy rules derived from k-means clusters, J. Electronic imaging, 4: 199-206.
17. Wang, L.X., 1993. Stable adaptive fuzzy control of nonlinear systems, submitted to IEEE Trans. Fuzzy Systems. 1: 2.