Design of Adaptive Robot Control System Using Recurrent Neural Network

Article *in* Journal of Intelligent & Robotic Systems · November 2005

DOI: 10.1007/s10846-005-9012-6 · Source: DBLP

1 author:

Sahin YILDIRIM
Erciyes Üniversitesi
180 PUBLICATIONS **1,147** CITATIONS

SEE PROFILE

Journal of Intelligent and Robotic Systems (2005) 44: 247–261
DOI: 10.1007/s10846-005-9012-6

Design of Adaptive Robot Control System Using Recurrent Neural Network

SAHIN YILDIRIM
Robotics Research Laboratory, Mechanical Engineering Department, Faculty of Engineering, University of Erciyes, Kayseri, Turkey; e-mail: sahiny@erciyes.edu.tr

(Received: 5 March 2004; in final form: 22 July 2005)

Abstract. The use of a new Recurrent Neural Network (RNN) for controlling a robot manipulator is presented in this paper. The RNN is a modification of Elman network. In order to solve load uncertainties, a fast-load adaptive identification is also employed in a control system. The weight parameters of the network are updated using the standard Back-Propagation (BP) learning algorithm. The proposed control system is consisted of a NN controller, fast-load adaptation and PID-Robust controller. A general feedforward neural network (FNN) and a Diagonal Recurrent Network (DRN) are utilised for comparison with the proposed RNN. A two-link planar robot manipulator is used to evaluate and compare performance of the proposed NN and the control scheme. The convergence and accuracy of the proposed control scheme is proved.

Key words: back-propagation, diagonal recurrent network, PID-robust controller, recurrent neural network, robot manipulator.

1. Introduction

Several NN models and neural learning algorithms have been applied to system controller design during the last decade and many promising results have been reported. NN has been employed as effective solution by exploiting its non-linear mapping properties [1, 2]. Most researches have used Feed-forward Neural Network (FNN), combined with Tapped Delay Lines (TDL) and the BP training algorithm to solve the dynamic problems. However the FNN is a static mapping and without the aid of TDL, it does not represent a dynamic system mapping. On the other hand, RNNs have important capabilities not found in FNNs, such as attractor dynamics and the ability to store information for later use [3, 4].

The NNs have also been used by several researchers for robot trajectory control [5]. They implemented a separate NN for each manipulator joint. An on-line Feedback Error Learning Method (FELM) has been used by Miyamoto et al. [6]. In their scheme, the NNs learn the required actuator torque along the desired

trajectories. Alternatively, CMAC networks have also been proposed for control of robot manipulator [7]. However, these networks require one to divide up the state space into an arbitrary number of smaller regions. Some methods use the design model [8]. The fundamental approach from this group, for example, that of Kawato who employs a neural structure based on non-recurrent single-layer FNN. This approach has a deterministic nature, but there are several drawbacks related primarily to the inherent complexity of the implementation of a complete model of robot dynamics and poor generalisation. Katic and Vukobratovic proposed a trainable robot controller architecture that is used a NN model as a form of intelligent feed forward control in the form of a decentralised control algorithm with training by a FELM [9]. Extensive research has also been carried out to design NN controllers for robot manipulator [10]. Two NN control schemes for non-model based robot manipulators have been presented along with the FELM by Jung an Hsia [11]. A NN-based adaptive computed-torque control approach is proposed by Li and Ang [12]. In that proposed method, a compensation torque is produced by a NN, which is added on to the nominal torque generated by the conventional computed-torque algorithm to compensate for any modelling errors.

In this paper, a new type of recurrent BP-NN, different from other recurrent networks, is proposed for robot control task. The proposed network is an extended version of the network described by Elman [13]. One NN is employed in total, first to learn the inverse dynamics of the robot. Second, after learning stage, the proposed NN is used as a controller of the robot without inverse equation of the robot.

The contents of the paper are as follows. Section 2 covers the basic theory of the proposed RNN and diagonal recurrent neural network. Section 3 introduces modelling implementation of the robot. The FELM is described in Section 4. In order to overcome load uncertainties of the robot, the load identification algorithm is explained in Section 5. Section 6 describes proposed control system. Simulation results for all cases is presented in Section 7. Some conclusions and discussions are outlined in last Section 8.

2. Neural Networks

2.1. PROPOSED RECURRENT NEURAL NETWORK (RNN)

In this section, a NN for modelling the input-output behaviour of the dynamic systems, which was modified from the Elman network, is developed. Figure 1(a) depicts the structure of the proposed network. In addition to the hidden feedback of the Elman network, there is also feedback connection from the output layer to the context layer. When non-linear activation functions are used for the hidden

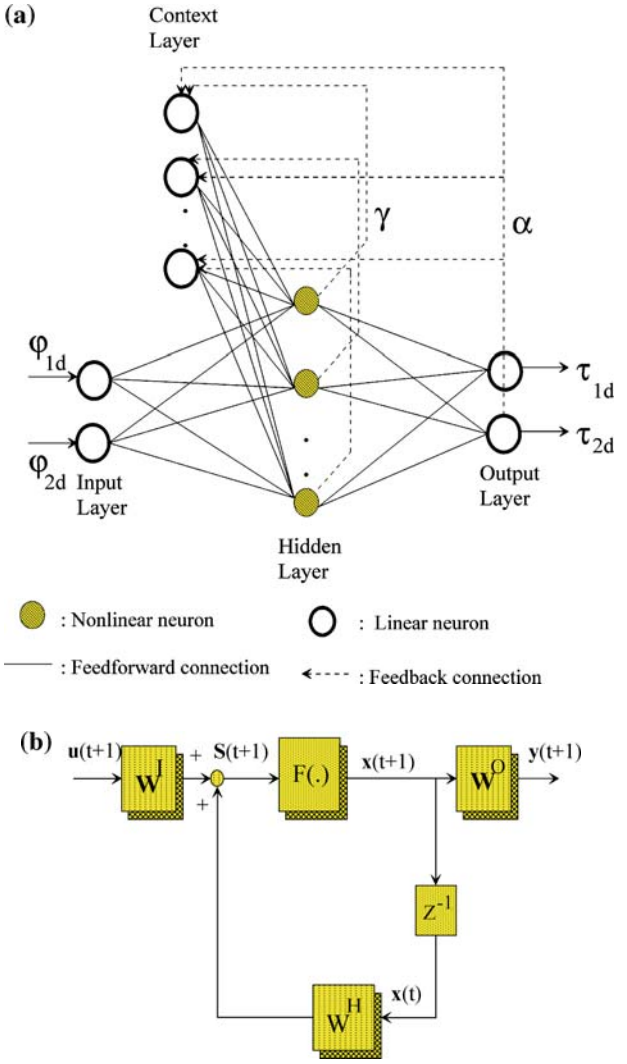

Figure 1. (a) Proposed neural controller. (b) Block diagram of Diagonal Neural Network. (c) Schematic representation of the DNN.

units and linear function for the input units, the proposed RNN can be described by the following state-space equations:

$$\mathbf{x}(k) = f(\mathbf{W}_{\mathrm{H}}(\mathbf{x}(k-1) + \tau(k-1)) + \mathbf{W}_{\mathrm{I}}\varphi(k-1) \tag{1}$$

$$\tau(k) = \mathbf{W}_{\mathrm{O}}\mathbf{x}(k) \tag{2}$$

$$\mathbf{c}(k) = \mathbf{S}_1\tau(k) + \mathbf{S}_2\mathbf{x}(k) \tag{3}$$

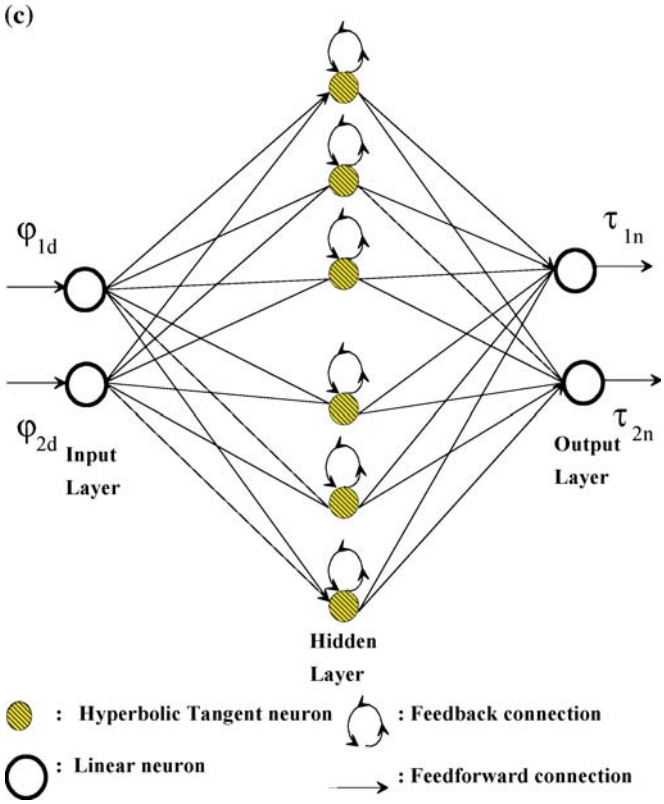

Figure 1. (Continued)

where $\mathbf{W}_I$, $\mathbf{W}_H$, $\mathbf{W}_O$, $\mathbf{S}_1$ and $\mathbf{S}_2$ are weight matrixes and $\varphi(k)$, $\mathbf{x}(k)$ $\mathbf{c}(k)$ and $\tau(k)$ represent the inputs vectors to the network, the outputs of the hidden units, the outputs of the context layer and the outputs of the network respectively. If the linear activation functions are also adopted for hidden units, the above equations become:

$$\tau(k) = \mathbf{W}_O\mathbf{x}(k) \tag{4}$$

$$\mathbf{x}(k) = \mathbf{W}_H\mathbf{c}(k-1) + \mathbf{W}_I\varphi(k-1) \tag{5}$$

$$\mathbf{c}(k) = \mathbf{S}_1\tau(k) + S_2\mathbf{x}(k) \tag{6}$$

Let p be the number of the input layer units, q the numbers of the hidden and context layer units, r the number of the output layer units. $\mathbf{S}_1$ and $\mathbf{S}_2$ are then given by:

$$\mathbf{S}_1 = \alpha\mathbf{J} \tag{7}$$

$$\mathbf{S}_2 = \gamma\mathbf{I} \tag{8}$$

where $\mathbf{J}$ is an $r \times q$ matrix with all elements equal to 1 and $\mathbf{I}$, the $q \times q$ identity matrix. The weights of the connections from the output layer to the context layer have the same value α and those of the connections from the hidden layer to context layer the same value γ.

Combining Equations (1)–(8) yields:

$$\mathbf{c}(k) = [\alpha \mathbf{J}\mathbf{W}_{\mathrm{H}}\mathbf{W}_{\mathrm{O}} + \gamma \mathbf{I}\mathbf{W}_{\mathrm{H}}]\mathbf{c}(k-1) + [\alpha \mathbf{J}\mathbf{W}_{\mathrm{O}}\mathbf{W}_{\mathrm{I}} + \gamma \mathbf{I}\mathbf{W}_{\mathrm{I}}]\varphi(k-1). \quad (9)$$

This is of the form:

$$\mathbf{c}(k) = \mathbf{K}_1\mathbf{c}(k-1) + \mathbf{K}_2\varphi(k-1) \quad (10)$$

where $\mathbf{K}_1 = [\alpha \mathbf{J}\mathbf{W}_{\mathrm{H}}\mathbf{W}_{\mathrm{O}} + \gamma \mathbf{I}\mathbf{W}_{\mathrm{H}}]$ is an $q \times q$ matrix and $\mathbf{K}_2 = [\alpha \mathbf{J}\mathbf{W}_{\mathrm{O}}\mathbf{W}_{\mathrm{I}} + \gamma \mathbf{I}\mathbf{W}_{\mathrm{I}}]$, an $q \times p$ matrix.

Thus, it is clear that Equation (10) represents the state equation of a general nth-order system of which $\mathbf{c}$ is the state vector.

The elements of $\mathbf{K}_1$ and $\mathbf{K}_2$ can be adjusted through training to suit any arbitrary nth-order system.

The proposed RNN has been shown theoretically to be able to represent an arbitrary linear dynamic system with the output of its context layer being equal to the states of the system.

2.2. DIAGONAL RECURRENT NETWORKS (DNN)

Figure 1(b) shows a schematic block diagram representation of a Diagonal Recurrent Neural Network (DNN) used in the work of Ku et al. [16]. This network is different from the RHN presented in the Section 2.1 in two respects: there are no feedback connections from the output layer to the hidden layer and the self-feedback connections in the hidden layer are all trainable. The operation of the DNN can be described as follows:

$$\mathbf{x}_j(t+1) = F\big(S_j(t+1)\big) \quad (11)$$

$$S_j(t+1) = \mathbf{W}_j^{\mathrm{H}}\mathbf{x}_j(t)\sum_{k=1}^{n_{\mathrm{I}}}\mathbf{W}_{jk}^{\mathrm{I}}u_k(t+1) \quad (12)$$

$$y_j(t+1) = \sum_{j=1}^{n_{\mathrm{H}}}\mathbf{W}_{ij}^{\mathrm{O}}\mathbf{x}_j(t+1) \quad (13)$$

where the hidden layer output $\mathbf{x}(t+1) \in \Re^{n_{\mathrm{H}}}$, lateral hidden layer weights $\mathbf{W}^{\mathrm{H}} \in \Re^{n_{\mathrm{H}} \times n_{\mathrm{H}}}$, the input layer weights $\mathbf{W}^{\mathrm{I}} \in \Re^{n_{\mathrm{H}} \times n_{\mathrm{I}}}$, and output layer weights $\mathbf{W}^{\mathrm{O}} \in \Re^{n_{\mathrm{H}} \times n_{\mathrm{O}}}$. n_{I}, n_{H} and n_{O} are the numbers of neurons in the input, hidden and output layers respectively. $\mathbf{u}(t+1)$ represents the input vector and $\mathbf{F}(.)$ is a hyperbolic tangent activation function with limits equal to -1.0 and 1.0. The

network has a feedforward input layer, a totally recurrent hidden layer and a feedforward output layer.

3. Model Implementation

Both the NN control algorithms, FELM and proposed neural control algorithm, were evaluated using the simulation of a two-degree-of-freedom serial link manipulator which is representative of the two servoed links of a SCARA-type robot (Figure 2). Since, only the first two joints of the robot are actuated by D.C. servomotors, it is appropriate to model the SCARA robot as a two-link manipulator. The values of the link parameters were suitably assumed to simplify the simulation. The manipulator was modelled with two links of lengths l_1 and l_2, and mass m_2 located at the distal end of link 2.

Actuator 1 applies torque τ_1 to drive joint 1, which is directly connected to link 1 (with inertia I_1) to the base of the arm. Actuator 2 applies torque τ_2, via a band and pulley system (with inertia I_2), to drive joint 2 connecting link 2 (with inertia I_3 and mass m_2) to link 1.

4. The Control System

The control system for controlling the robot manipulator is shown in Figure 6. The control signal in this diagram is the torque τ_T, which is the sum of the two components:

$$\tau_T = \tau_{NN} + \tau_{PID} \tag{14}$$

where τ_{NN} is a feed-forward torque and τ_{PID} is the feedback torque produced by the conventional PID controller. The feedback errors used to train neural networks NNC is defined as

$$\mathbf{e} = \tau_{PID}(k) \tag{15}$$

where $\tau_{PID}(k)$ is output of the feedback (PID) controller at time k.

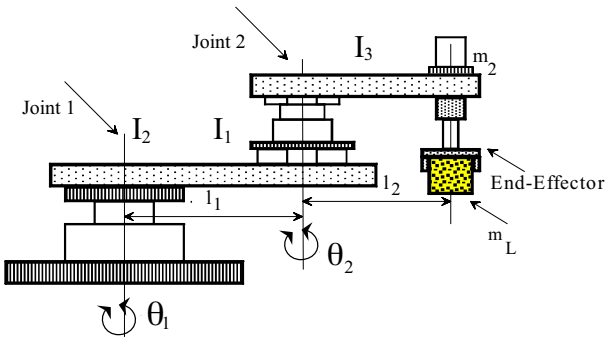

Figure 2. Configuration of the SCARA robot (after dynamics change).

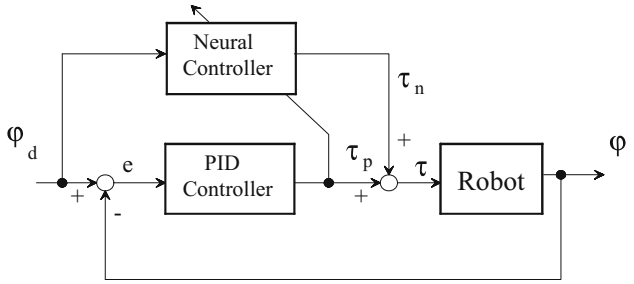

Figure 3. Control scheme of the robot manipulator.

Hence, if the neural networks NNC is trained such that $\lim_{k\to\infty} e_1(k) = 0$ and $\lim_{k\to\infty} e_2(k) = 0$ then asymptotically the output tracking condition is satisfied:

$$\lim_{k\to\infty} \tau_1(k) = 0, \ \lim_{k\to\infty} \tau_2(k) = 0 \tag{16}$$

In this control scheme, recurrent neural networks (RNNs) are employed as a controller of joints of the robot. Neural controller uses the same RNN architecture shown in Figure 1(a), which has an input layer, an output layer, a context layer and one hidden layer with hyperbolic tangent type activation recurrent neurons. The block diagram of the RNN based control system is shown in Figure 6. The inputs to the RNNs controllers are the desired angular position of robot and the output of the RNNs controller is the control signal to the robot manipulator. The output of the neural controller for the robot can be written in the following form as:

$$\tau_{ni}(k) = f(\varphi_i) \quad i = 1, 2 \tag{17}$$

where $f(.)$ is the function approximation of the network for the inverse dynamics, τ_{ni} is the network output as the estimate of the robot input τ.

The weights of the neural controllers are adjusted such that the error between the output of the neural controllers and the desired output from sum of the output of the PID controller.

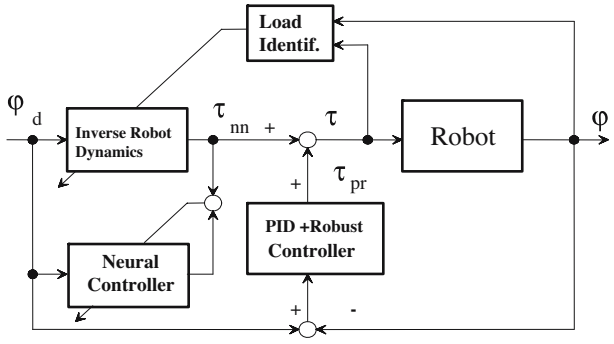

Figure 4. Proposed control scheme.

Training and learning by the proposed connectionist structure is accomplished exclusively using an on-line regime by feedback error or driving torque error learning method. This method is exclusively an on-line method for robot manipulator control. However, this control structure provides an internal teacher so that the control scheme works in an unsupervised manner because there is no external teacher in this case. The adjustment of the network during the control by feedback-error learning is more convenient than the other learning structures such as generalised or specialised learning Figures 3 and 4.

5. Payload Identification

It has been shown by Craig et al. [15] that the linear parameterisation of manipulator dynamics can be made such that the right-hand side of robot dynamics is linear in terms of a suitable selected set of equivalent manipulator and load parameters. Assume that link and friction parameters have been identified off-line; then, with a minor modification, manipulator dynamics can be re-parameterized with respect to load parameters such that

$$Y(\theta(k),\ \theta'(k)\theta''(k))]\Theta(k) = \left[\tau(k) - Y_0\Big(\theta(k),\ \theta'(k),\ \theta''(k)\Big)\right] + \mathbf{W} \tag{18}$$

where $\Theta(k)$ is the equivalent load parameter vector, Y and Y_{O} are known functions of $\theta(k)$, $\theta'(k)$ and $\theta''(k)$ that are not depend on load parameters.

In order to estimate the time-varying load, the most commonly used approach to such an adaptive identification is the forgetting factor method. Assume that $\theta(k)$, $\theta'(k)$, $\theta''(k)$ and $\tau(k)$ are available and define

$$\mathbf{A}^{\mathrm{T}}(k) = \left[Y\Big(\theta(k),\ \theta'(k),\ \theta''(k)\Big)\right]_{k=\mathrm{T}(k)} \tag{19}$$

$$\mathbf{B}(k) = \left[\tau(k) - Y_0\Big(\theta(k),\ \theta'(k),\ \theta''(k)\Big)\right]_{k=\mathrm{T}(k)} \tag{20}$$

where is $\mathrm{T}(k)$ is the sampling time at time k. Then, within the frame work of equation errors, the Recursive Least Square Estimation (RLSE) of $\Theta(k)$ with the forgetting factor $\Phi(k)$ is given by

$$\begin{aligned}\Phi(k) = \Phi(k-1)\big[\big(\mathbf{P}(k-1)\mathbf{A}(k)\big/\big(\lambda + \mathbf{A}^{T}(k)\mathbf{P}(k-1)\mathbf{A}(k)\big)\big)\big] \\ \times\big(\mathbf{B}(k) - \mathrm{A}^{\mathrm{T}}(k)\Phi(k-1)\big).\end{aligned} \tag{21}$$

In above expression, $\mathbf{P}(k)$ is given by

$$\begin{aligned}\mathbf{P}(k) = (1/\lambda)\big[\mathbf{P}(k-1) - \big(\mathbf{P}(k-1)\mathbf{A}(k)\mathbf{A}^{\mathrm{T}}(k)\mathbf{P}(k-1)\big/ \\ \big(\lambda + \mathbf{A}^{\mathrm{T}}(k)\mathbf{P}(k-1)\mathbf{A}(k)\big)\big)\big]\end{aligned} \tag{22}$$

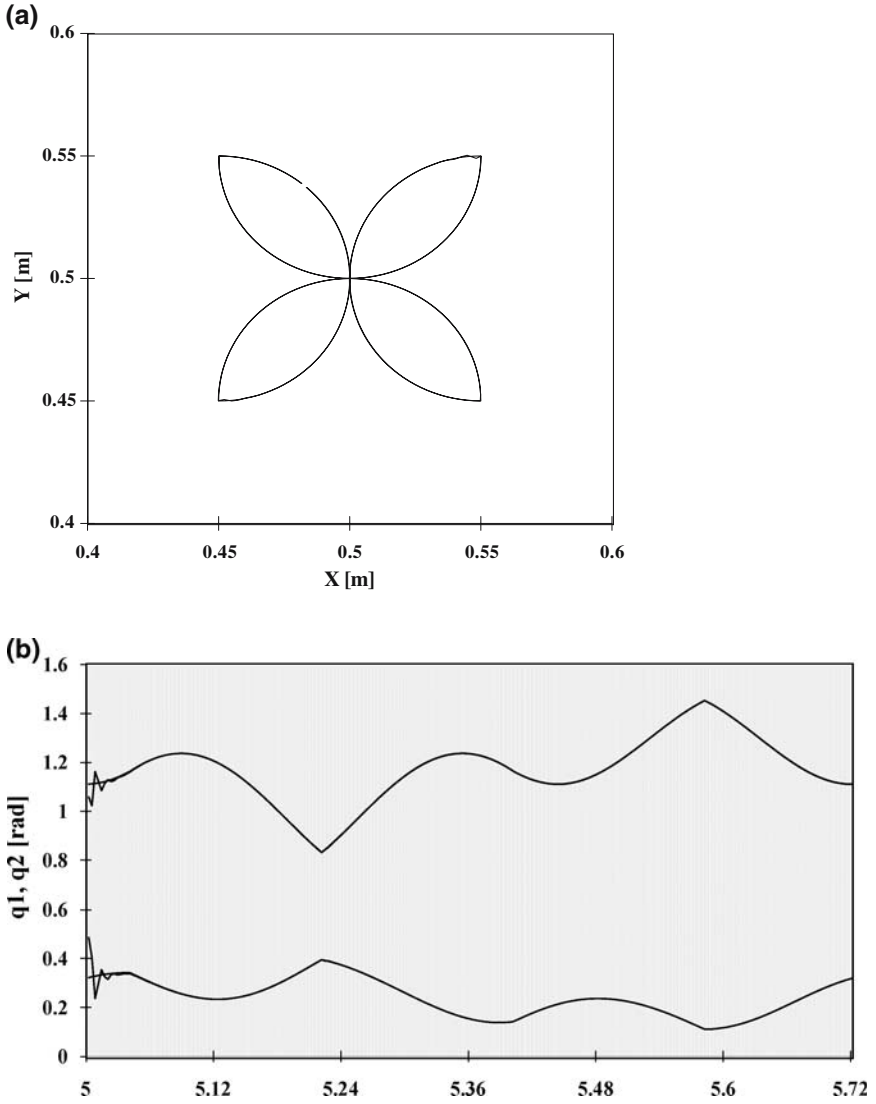

Figure 5. (a) Four-leafed-rose trajectory (using proposed control scheme, m_L = 90 kg). (b) Control response of robot (using proposed control scheme, m_L = 90 kg).

where $\Theta(0)$ and $\mathbf{P}(0) > 0$ are given, where the single joint of robots has been assumed for simplicity of presentation. The constant forgetting factor is usually taken as 0.95–0.99.

6. Proposed Control System

The proposed control system consists of a robust controller Equation (24) with the load parameter updated by load-adaptive identification algorithm Equation (22),

Table I. RMSE performances of RNN, DNN and FNN controllers.

m_L [kg]	NN type	RMSE (first trial)	RMSE (second trial)
10	MEN	0.000762	0.000035
90	MEN	0.003593	0.000089
10	DNN	0.000805	0.000232
90	DNN	0.003908	0.000242
10	FNN	0.000949	0.000654
90	FNN	0.008679	0.000968

a conventional PI controller and a neural controller. The control scheme for controlling a robot is shown Figure 7. The control signal in this diagram is the torque $\tau_T(k)$, which is the sum of the three components:

$$\tau_T(k) = \tau_n(k) + \tau_p(k) + \tau_r(k) \tag{23}$$

where $\tau_n(k)$ is feed-forward toque produced by NN controller, $\tau_p(k)$ is feedback torque produced by conventional PID controller and $\tau_r(k)$ is feedback torque produced by robust controller. The robust controller is used as follows;

$$\tau_r(k) = -\mathbf{K}_r\mathbf{e}(k) - \mathbf{S}\text{sgn}(\dot{\mathbf{e}}(k)) \tag{24}$$

where $\mathbf{e}(k) = \varphi(k) - \varphi_d(k)$ denotes the position tracking error, $\mathbf{K}_r$ is a positive definite matrix and the notation $\mathbf{S}\text{sgn}(\dot{\mathbf{e}}(k)) = [s_1\text{sgn}(\dot{e}_1(k)), s_2\text{sgn}(\dot{e}_2(k)) \ldots . s_n\text{sgn}(\dot{e}_n(k))]$ is used for simplicity.

7. Simulation Results

In order to demonstrate the feasibility of the proposed RNN controller, a two-link planar SCARA manipulator was used for the simulation. The SCARA is robot very stiff vertically but has good lateral compliance, a property, which is very useful in typical industrial applications such as assembly of parts. The SCARA has two joints that are driven by D.C. servomotors. Both the gripper and *Z*-axis up-down motion are actuated pneumatically. The gripper roll is controlled by a stepper motor. Simulation was carried out using a fourth-order Runge–Kutta algorithm with a step size of 0.0001 s.

The PID feedback control was chosen with next values for local feedback gains:

$$\mathbf{K}_P = \text{diag}[1000, 1000],\ \mathbf{K}_I = \text{diag}[100, 100] \text{ and } \mathbf{K}_D = \text{diag}[1, 1].$$

The effectiveness of the proposed neural controllers was tested to change dynamic parameters of the robot. Figure 5(a) illustrates four-leafed-rose trajectory after a change in the dynamics of the robot. After one training epochs

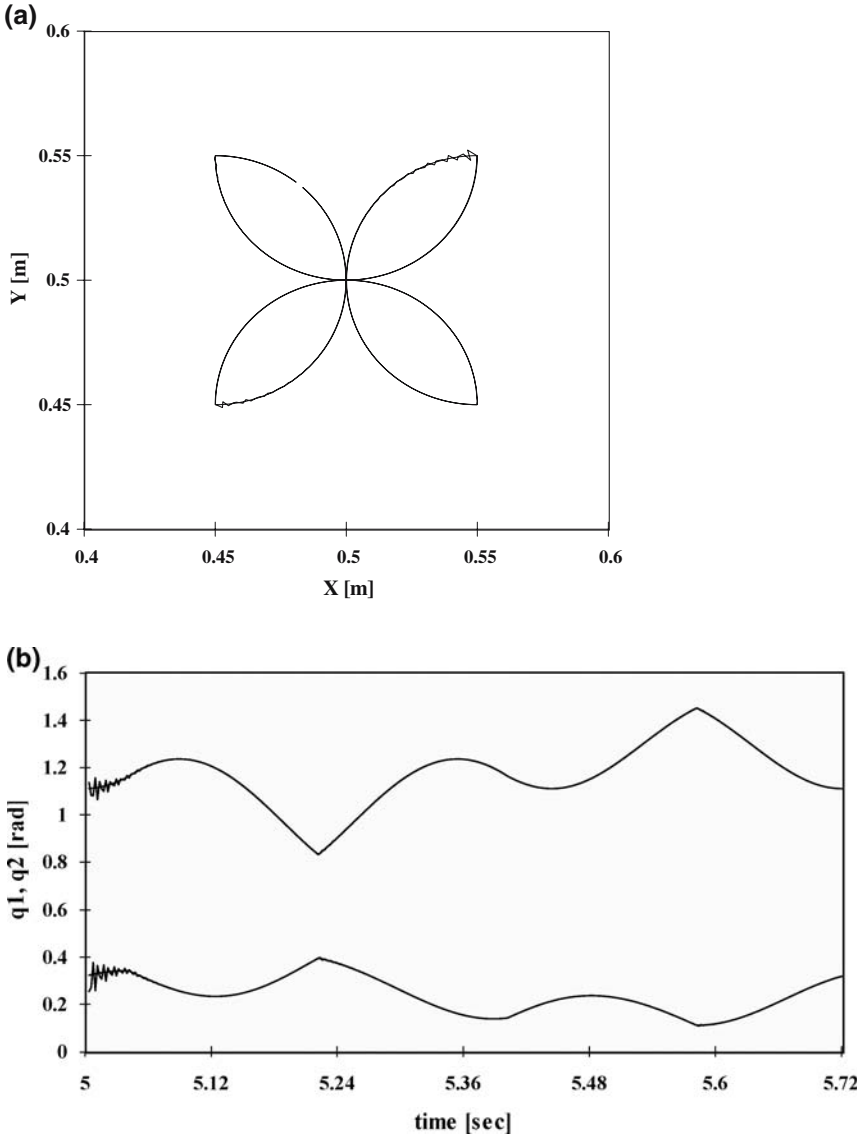

Figure 6. (a) Four-leafed-rose trajectory (using FELM, m_L = 90 kg). (b) Control response (using FELM, m_L = 90 kg).

or cycles, the error convergence to very small value. The RMSE error is shown in Table I, and control responses obtained with MEN controller of the end-effector of the robot are shown Figure 5(b). As can be seen from these figures, the actual trajectory of robot arm is exactly the same as the desired trajectory. It is observed that there is an oscillation for first instant in Figure 5(b). This is because dynamics parameters of the robot are suddenly changed from 10 to 90 kg.

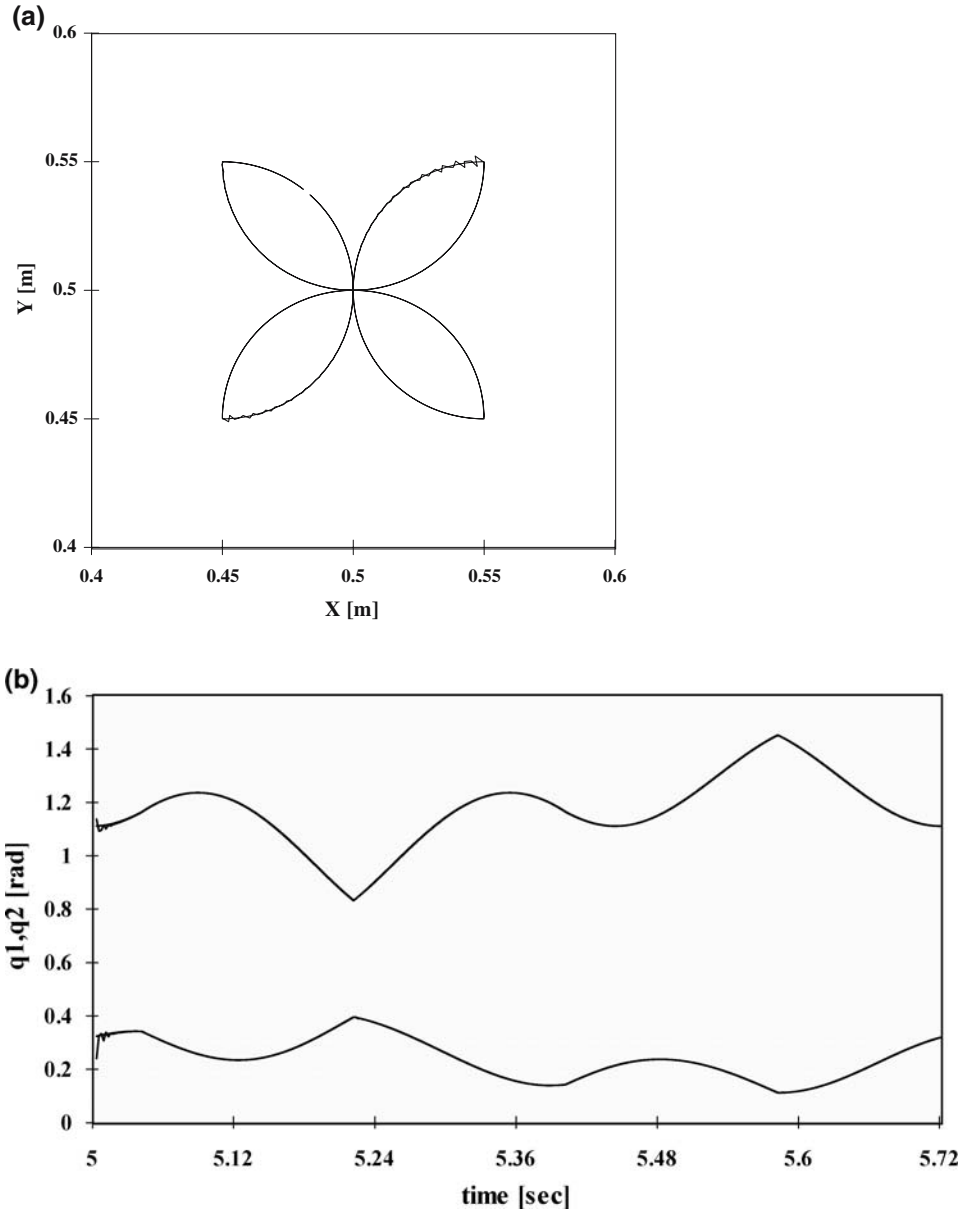

Figure 7. (a) Four-leafed-rose trajectory (using DRN controller, m_L = 90 kg). (b) Control responses obtained with DRN controller (m_L = 90 kg).

When FELM used for controlling the robot manipulator. Some simulation results are shown in Figure 6(a,b). According to these results, that proposed robust adaptive control scheme is considerably better than non-adaptive control scheme FELM.

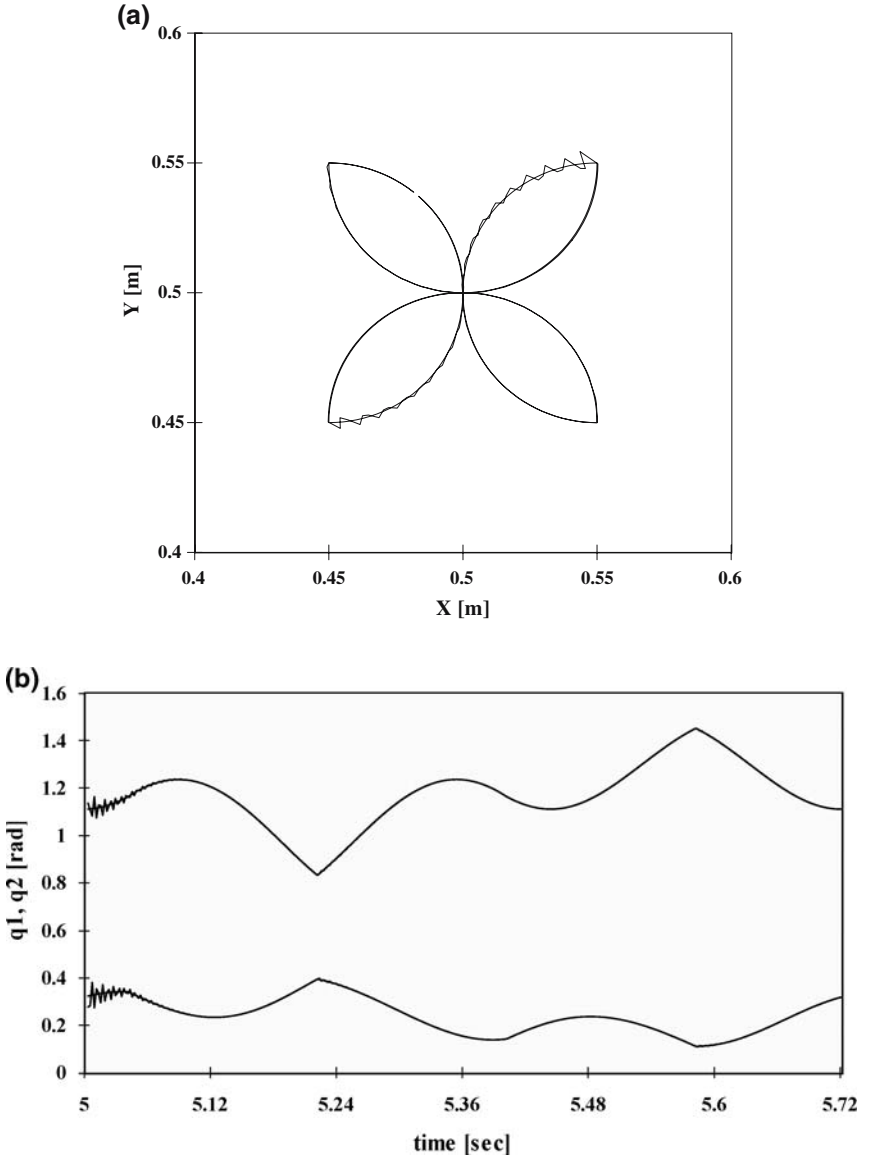

Figure 8. (a) Four-leafed-rose trajectory (using FNN, m_L = 90 kg). (b) Control responses obtained with FNN controller (m_L = 90 kg).

For comparison with RNN, DRN was also used as a controller of the robot control system. Simulation results are shown in Figure 7(a,b). Due to complexity of the DRN, training progress of the network is taken long time.

On the other hand, another comparison for FNN controller was employed on the robot control system. Simulation results of this case are shown in Figure 8(a,b). As can be depicted from the figures and table, the results of this case is

not fair enough for such applications. Table I illustrated the performances of the RNN, DRN and FNN controllers.

Table I illustrated the performances of the RNN, DRN and FNN controllers.

8. Conclusion and Discussion

A new RNN was employed to adaptive robot control and satisfactory results were obtained. The proposed RNN has a recurrent nature, have potentially more computational power than pure FNN. This type of NN is very promising exploitation in the robotic system. Unlike the basic Elman network, it was able to learn the input-output behaviour of the robot manipulator. Due to the continuous on-line nature of the NN training, the proposed control system was able to adapt to changes in the robot operating conditions and dynamics characteristics. The results obtained with FELM are given for comparison with the proposed RNN controller. The proposed adaptive control scheme is considerably better than the non-model based FELM. Its applicability and effectiveness have been demonstrated by simulation on a planar robot manipulator. Its superior performance becomes very attractive for real-time implementation and application in complex industrial robots.

The results of applying the proposed recurrent network to the simulated control of the trajectory of a planar robot arm have demonstrated its superiority over feedforward and diagonal recurrent networks. The proposed neural control scheme has also been shown to perform better than the conventional control schemes [14]. The robustness of the controller to system parameter changes was tested and observed as well. The results show that the neural network is able to adapt effectively after parameters change of the robot.

It is reasonable to believe that the proposed control system will be able to control other similar non-linear processes as well.

The proposed neuromorphic structure with robust controller structure and fast convergence properties has also great possibility in real-time control and learning of robots.

Acknowledgement

The author wishes to express his thanks to Erciyes University Research Fund for supporting Dr. Şahin YILDIRIM's research at Erciyes University.

References

1. Psaltis, D., Sideris, A., and Yamamura, A.: A multilayered neural network controller, *IEEE Control Syst. Mag.* (1989), 17–21.
2. Narendra, K. S. and Parthasarathy, K.: Identification and control of dynamical systems using neural networks, *IEEE Trans. Neural Netw.* **1** (1990), 4–27

3. Pineda, F. J.: Dynamics and architecture for neural computation, *J. Complex.* **4** (1988), 216–245
4. Pearlmutter, B. A.: Learning state space trajectories in recurrent neural networks, *Neural Comput.* **1** (1989), 263–269.
5. Khemaissia, S. and Morris, A. S.: Neuro-adaptive control of robot manipulators, *Robotica* **11** (1993), 456–473.
6. Miyamoto, H., Kawato, M., Setoyama, T., and Suzuki, R.: Feedback-error-learning neural networks for trajectory control of a robot manipulator, *Neural Netw.* **1** (1988), 251–265.
7. Miller, W. T., Glanz, F. H., and Kraft, L. G.: CMAC: An associative neural network alternative to backpropagtion, *Proc. IEEE* **78** (1990), 1561–1567.
8. Kawato, M., Uno, Y., Isabe, M., and Suzuki, R.: Hierarchical neural network model for voluntary movement with application to robotics, *IEEE Control Syst. Mag.* (1988), 8–16.
9. Katic, D. M. and Vukobratovic, M. K.: Highly efficient robot dynamics learning by decomposed connectionist feedforward control structure, *IEEE Trans. Syst. Man Cybern.* **25**(1) (1995), 145–158.
10. Ishiguro, A., Furuhashii, T., Okuma, S., and Uchikawa, Y.: A neural network compensator for uncertainties of robot manipulator, *IEEE Trans. Ind. Electron.* **39** (1992), 61–66.
11. Jung, S. and Hsia, T. C.: New neural network control technique for non-linear based robot manipulator control, *IEEE Int. Conf. Syst., Man Cybernetics, Canada* **3** (1995), 2928–2933.
12. Li, Q., Poo, A. N., and Ang, M.: An enhanced computed-torque control scheme for robot manipulators with a neuro-compensator, *IEEE Int. Conf. Syst., Man Cybernetics, Canada* **1** (1995), 56–60.
13. Elman, J. L.: Finding structure in time, *Cogn. Sci.* **14** (1990), 179–211.
14. Yildirim, S.: Neural network controller for cooperating robots, *Electron. Lett.* **37**(22) (2001), 1351–1352.
15. Craig, J. J., Hsu P., and Sastry, S. S.: Adaptive control of mechanical manipulators, *Int. J. Rob. Res.* **6**(6) (1987), 16–28.
16. Ku, C. C. and Lee, K. Y.: Diagonal recurrent neural networks for dynamic systems control, *IEEE Trans. Neural Netw.* **6** (1995), 144–156.