Adaptive robust neural controller for robots

Şahin Yıldırım
Robotics Research Laboratory, Mechanical Engineering Department, Engineering Faculty, Erciyes University, Kayseri, Turkey

Received 23 January 2003; received in revised form 2 September 2003; accepted 17 November 2003

Abstract

This paper presents an investigation on the trajectory control of a robot using a new type of recurrent neural network. A three-layered recurrent neural network is employed to estimate the forward dynamics model of the robot. Standard backpropagation (BP) algorithm is used as a learning algorithm for this network to minimise the difference between the robot actual response and that predicted by the neural network. This algorithm is employed to update the connection weights of the neural network controller with three layers using a gradient function.

© 2004 Elsevier B.V. All rights reserved.

Keywords: Neural network; Robot; BP algorithm; Trajectory

1. Introduction

In recent years, multi-layered neural networks (MLNNs) have been used in different aspects of robotics [2,3]. Solutions have been presented for kinematics, dynamics, trajectory planning, sensing and control [1,5,12].

Two classes of neural network which are useful in control applications of robot manipulators are: (i) multi-layered neural networks (MLNNs) and (ii) recurrent neural networks (RNNs). A comprehensive treatment of the topic in control applications is given by Narendra and Mukhopadhyay [6] where several models for identifying an unknown non-linear single-input single-output plant by using MLNNs are proposed. The problem of decoupling an unknown, non-linear, multi-variable dynamical system using neural networks is considered by Narendra and Mukhopadhyay [6]. An on-line adaptive neural-network-based controller is developed based on the multi-layer feedforward neural network, and the control of a specific bioreactor is studied under the control scheme developed in [8].

Some work on employing recurrent networks for system identification has been reported in [7,10,11]. These networks have a dynamic memory and are thus able to model dynamic systems without requiring external feedback lines.

The control of a robot involves two fundamental problems: trajectory planning, and motion control. In the task-planning phase, high-level planners manage and coordinate information concerning the job to be performed. Trajectory planning involves finding the sequence of points through which the manipulator end point must pass, given initial and goal co-ordinates, intermediate points and appropriate constraints. Such constraints may include limits on velocity and acceleration or need to avoid obstacles. Given such a trajectory, the motion control problems consist of finding the joint torques that will cause the arm to follow it while satisfying the constraints. Standard
adaptive controllers for robots have been employed by Kung and Hwang [4]. Neural networks have been applied to both of the problem areas indicated above [4,9]. However, most of the useful work is being done in trajectory planning and motion control.

Also, standard adaptive controllers for robots have been employed by other researchers. Adaptive controllers for robot positioning and tracking using direct visual feedback with camera-in-hand configuration have been proposed by Nasisi and Carelli [14]. In their paper, the controllers were designed to compensate for full robot dynamics. An adaptation algorithm was introduced to reduce the design sensitivity due to robot and payload dynamics uncertainties. But there are still some errors for load uncertainties of the robot. An adaptive control law has been designed for robot manipulators based on the use of reference velocities instead of the actual ones and feedback signals generated from the position errors [15]. In this paper, the robot is not following desired trajectory for load uncertainties. An on-line adaptive neural-network-based robust-PID controller is developed in this paper. The detailed design procedures of the proposed controller are given, along with an illustrative example of controlling a robot manipulator. Simulation results show that the proposed neural controller is successful in controlling non-linear time-varying systems.

This paper is organised as follows: Section 2 gives some details about the proposed recurrent neural network. The details of the proposed control system are presented in Section 3. The effectiveness of the proposed control system is demonstrated by some simulations in Section 5. Finally, this paper is concluded with summary in Section 6.

2. Recurrent neural networks

Recurrent neural networks have been gaining increasing attention in recent years. In a number of applications involving temporal signals, they are able to produce promising results [13]. Though the learning ability of the recurrent network architecture is impressive, no simple algorithm is presently available for training these networks. In their paper, one approach is to modify the feedforward backpropagation learning algorithm for the recurrent architecture.

2.1. Proposed recurrent hybrid network

In this section, a neural network for modelling the input–output behaviour of the dynamic systems is developed. Fig. 1 shows the block diagram of the proposed network. In addition to the input and the hidden feedforward connections of the hybrid network, there are also feedback connections from the output layer to the hidden layer and self-feedback connections in the hidden layer [10,11].

The equation of the feedforward hybrid network is as follows:

\[ x_{1}(k+1) = W_{I}u(k) + \beta x_{1}(k) + \alpha y(k), \]

(1)

Fig. 1. Block diagram of proposed hybrid network [10].
\[ x_2(k+1) = F(W_1u(k) + W_2x_1(k) + uJy(k)), \]  
\[ y(k+1) = W_{H1}x_1(k+1) + W_{H2}x_2(k+1). \]  
(2)

If the linear hybrid activation functions are also adopted for the non-linear hidden units, the above equations become

\[ x(k+1) = x_1(k+1) = x_2(k+1), \]  
(4)

\[ y(k+1) = W_H1x_1(k+1), \]  
(5)

\[ x(k+1) = W_Hu(k) + S_1x(k) + S_2y(k), \]  
(6)

\[ x(k+1) = (S_1 + S_2W_H)x(k) + W_Iu(k). \]  
(7)

The weights in \( S_1 \) and \( S_2 \) are related to the feedback connections and are fixed. The weights of the connections from the output layer to the hidden layer have the same value \( \alpha \) and those of the connections from the hidden layer to itself have the same value \( \beta \). Let \( r \) be the number of output units and \( q \) be the number of hidden units. \( S_1 \) and \( S_2 \) are then given by

\[ S_1 = \beta I, \]  
(8)

\[ S_2 = \alpha J, \]  
(9)

where \( J \) is a \( q \times r \) matrix with all elements equal to 1 and \( I \) is the \( q \times q \) identity matrix.

The weights of the proposed neural controller were adjusted using standard backpropagation algorithm. Therefore, the error signal can be backpropagated to the controller via the neural model. The feedback and the weights of the connections between the output and hidden layers can be upgraded by applying the BP algorithm.

This makes the modified network suitable for modelling multi-degrees of freedom robot manipulators.

3. Proposed control system

In most industrial systems, their characteristics change with time owing to components wearing out, the environment change, etc. For such kind of systems, an adaptation mechanism is helpful to achieve a satisfactory control performance in all circumstances. In this section, an on-line adaptive robust–PID control scheme that is suitable for non-linear systems is proposed. To control a time-varying non-linear system, it seems essential that the controller involved should possess the ability to change its own behaviour according to the parameter and/or structural variations occurring in the system. An on-line adaptive neural network robust–PID control system has been devised for such purposes (Fig. 2).

The proposed control system is composed of four main components: a neural network controller, a robust–PID controller to produce the suitable control action, a neural model to emulate the system behaviour and associated...
on-line learning algorithms to adjust the weights matrices of the neural networks according to the available system information. A filter (F), not shown in Fig. 2, is also employed to control the tracking properties and robustness of the control system. Due to initial large input error of the system, the filter was used to overcome this error problem (Fig. 3).

3.1 Neural network controller

The neural network control scheme is selected as shown in Fig. 2. This control scheme receives the desired position $\phi_d$ as its input, and its output $\tau$ is then sent to the system and the neural network model simultaneously. Considering the neural network controller as an element of this control scheme, its desired output, which is the controller action $\tau_{nn}$, cannot be defined explicitly. Hence, it is very difficult to form the training data set for the neural network controller alone. To avoid this difficulty, an auxiliary network controller is constructed, which is a combination of the neural network controller and the neural network model, as shown in Fig. 3. In an efficient training process, the control error will eventually approach zero as the training process continues, i.e.

$$\lim_{t \to \infty} e_c(t) = \lim_{t \to \infty} (\phi_{cd}(t) - \phi_c(t)) = \lim_{t \to \infty} (\phi_{rd}(t) - \phi_m(t)), \quad (12)$$

where $\phi_{rd}(t)$, not shown in the figure, is the desired output and $\phi_m(t)$, not shown in the figure, is the output of the model.

Let us consider the training process of the neural controller in more detail. In the auxiliary neural controller, where the connected neural controller and the neural network model are viewed as a single neural network, the backpropagation algorithm can be readily extended to carry out the associated training process.
The on-line adaptive neural-network-based robust PID control algorithm can be implemented through the following iterative process. First, the neural model (NM) of robot is obtained after training process, and the neural controller is trained through the specialized inverse learning scheme. The use of a neural network to learn the plant inverse model avoids the need for a priori knowledge or assumptions about the plant. Then the control is calculated by the forward processing of the neural controller (NC) for the given reference points. Later, this control is sent to both the neural model and the system itself, simultaneously. A one-step-ahead prediction of the system output can be obtained from the neural model and is compared with the system output to produce the error signal, which will be used for the training of the neural model. This procedure is repeated as time progresses.

4. Application to the robot

This section describes the results of simulation for control of a robot using neural networks. The implementation was done for the two major links of the Adept One, which has four axes and Scara configuration. The Scara configuration has three parallel revolute joints allowing it to move in a plane, with a fourth prismatic joint for moving the end-effector normal to the plane. The chief advantage is that the first three joints do not have to support any of the weight of the manipulator or the load. In addition, link 0 can easily house the actuators for the first two joints. The actuators can be made very large, so the robot can move very fast. For example, the Adept One Scara manipulator can move at up to 9.11 m per second, about 10 times faster than most articulated industrial robots. This configuration is best suited to planar tasks.

The Adept One robot is unique in that it was the first commercial robot to implement a direct drive system for actuation.

4.1. Dynamics of the robot

Fig. 4 shows a schematic drawing of the links 1 and 2 of the Adept One. Both actuators are located at the base of the robot. Joint 1’s actuator applies torque $\tau_1$ to the link 1 structure, which has rotational inertia, $I_1$. Joint 2’s actuators apply a torque $\tau_2$ to the inner column (with inertia $I_2$), which drives joint 2 through a steel band. The dynamic equations of the robot are given in Appendix A.

5. Simulation

In order to verify and demonstrate the effectiveness of the proposed neural network approach, a simulation example of two-degrees-of-freedom planar Scara type robot is studied. The payload is considered as a point mass
Table 1
Network and training parameters

<table>
<thead>
<tr>
<th>Control scheme</th>
<th>Parameters</th>
<th>Activation function</th>
</tr>
</thead>
<tbody>
<tr>
<td>Neural controller</td>
<td>η: learning rate, µ: momentum term, α: feedback gain, β: feedback gain, n: number of hidden neurons, N: iteration numbers</td>
<td>η: 0.0001, µ: 0.01, α: 0.8, β: 0.8, n: 6, N: 50000, sin</td>
</tr>
</tbody>
</table>

In this work, the following empirically determined values of PID controller were adopted for $K_P$, $K_I$ and $K_D$: $K_P = \text{diag}[1000, 1000]$, $K_I = \text{diag}[100, 100]$ and $K_D = \text{diag}[1, 1]$. The values for $\eta$ (learning rate) and $\mu$ (momentum term) are listed in Table 1.

The robust controller, which is employed to overcome large disturbances such as large payloads, is used as follows:

$$\tau_r = -K_0 e - S \text{sgn}(e), \quad (13)$$

where $e = \phi_e(t) - \phi_d(t)$ and $e$ are the position tracking error and its time derivative, $K_0$ is a proportional gain matrix, empirically set to $K_0 = \text{diag}[10, 10]$, sgn is the signum function, $S = \{s_1, s_2\}^T$, $s_1$ and $s_2$ are constants also empirically set to 0.5 and 0.5.

Commanding the end point to track a four-petal-flower trajectory of the robot as shown in Figs. 5 and 6 to test the performances of the proposed control scheme. It is clear that proposed control scheme is faster with better accuracy than the linear PID control scheme tested in this study. To test the ability of the neural controller to adapt to different robot payloads, the value of $m_2$ was increased to 30kg following training on the 10kg payload. The effect of a change in the payload mass from 10 to 30kg for the proposed neural control system and PID control system may be seen in Figs. 7 and 8. The PID controller with empirically selected gain parameters has the best structure for...
trajectory control of robot with load uncertainties. The performance of the system was computed on the second attempt at the following specified trajectory. The actual trajectory of the robot end effector superimposed on the specified trajectory is plotted in Fig. 7. By comparing Figs. 7 and 8, it can be seen that the best results were obtained with the proposed neural controller and control system.

From the results obtained, it can be seen that the proposed control system employing a proposed neural controller produced the best performance while the PID controller yielded poor control. A reason for the strong performance of the neural-network-based control system was the inclusion of both linear and non-linear neurons in the network. This facilitated the training of the controller because the linear neurons could readily learn the linear part of the robot dynamics and the non-linear neurons, the non-linear part.
The speed of the end-effector of the robot is one of the important criteria of robot performance. If the speed is increased the tracking performance of the robot becomes worse. However a suitable selection of gains of linear feedback may improve tracking performance.

The model-based computed-torque approach works well and can give better control than a simple PID scheme when an accurate dynamics model of the robot is available. However, in practical simulations, it is very difficult, if not impossible, for the parameters associated with the robot model to be determined exactly. Moreover, during operation, the dynamics of a robot may change significantly and rapidly. Under such circumstances, the trajectory tracking performance when using the computed-torque control significantly degrades due to the inaccuracy of the dynamic model for computing the control torques.

6. Conclusion and discussion

In this investigation, the proposed neural control scheme has been used to control trajectory of a robot manipulator for different payload conditions, which has characteristics of time variance and non-linearity. The reason for selecting the predicted performance error is that system output is usually contaminated with dynamic change of robot and noise, resulting in a more difficult training process for the neural network. It is shown that the proposed algorithm and the controller architecture provide an effective on-line control strategy for the Scara robot. Simulation studies for the control of the robot indicate that the proposed strategy gives good tracking performance. It may be noted that the generalisation capabilities of the learning algorithm in the presence of change of parameters is an important factor affecting the tracking performance of an on-line controller.

The proposed recurrent hybrid network has two advantages: the recurrent conditions provide an inherent dynamic memory which facilitates modelling and the hybrid hidden layer enables real systems comprising linear and non-linear parts to be modelled accurately. The results of applying the recurrent hybrid network to the simulated control of the trajectory of a planar robot arm have demonstrated its superiority over feedforward and purely linear or non-linear networks. The proposed neural control scheme has also been shown to perform better than the conventional computed torque and PID schemes [7].
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.

Appendix A. Dynamic of the robot

The dynamics equations of each robot arm can be written as follows [10]:

\[ \tau = M(\phi)\ddot{\phi} + C(\phi, \dot{\phi}) + F(\phi, \dot{\phi}), \]  
(A.1)

where \( \tau = [\tau_1 \tau_2]^T \), \( \phi = [\phi_1 \phi_2]^T \), \( \ddot{\phi} \) and \( \dot{\phi} \) are the first and second time derivatives of \( \phi \), \( M(\phi) \) is the inertia matrix of the robot, \( C(\phi, \dot{\phi}) \) is a \( 2 \times 1 \) vector of centrifugal and Coriolis terms and \( F(\phi, \dot{\phi}) \) is a \( 2 \times 1 \) vector of viscous and Coulomb friction terms:

\[ M = \begin{bmatrix} I_1 + m_2 l_2^2 & -m_2 l_2 c(k_e(\phi_1 + \phi_2)) \\ -m_2 l_2 c(k_e(\phi_1 + \phi_2)) & I_2 + l_1 + m_2 l_2^2 \end{bmatrix}, \]  
(A.2)

\[ C = \begin{bmatrix} m_2 l_2 c(k_e(\phi_1 + \phi_2)) \dot{\phi}_2 \\ m_2 l_2 c(k_e(\phi_1 + \phi_2)) \dot{\phi}_1 \end{bmatrix}, \]  
(A.3)

\[ F = \begin{bmatrix} V_1 \dot{\phi}_1 + F_1 sgn(\dot{\phi}_1) \\ V_2 \dot{\phi}_2 + F_2 sgn(\dot{\phi}_2) \end{bmatrix}. \]  
(A.4)

Eq. (A.1) can be rewritten as

\[ \begin{bmatrix} \tau_1 \\ \tau_2 \end{bmatrix} = \begin{bmatrix} M_{11} & M_{12} \\ M_{21} & M_{22} \end{bmatrix} \begin{bmatrix} \ddot{\phi}_1 \\ \ddot{\phi}_2 \end{bmatrix} + \begin{bmatrix} C_{11} & C_{12} \\ C_{21} & C_{22} \end{bmatrix} \begin{bmatrix} \dot{\phi}_1 \\ \dot{\phi}_2 \end{bmatrix} + \begin{bmatrix} F_1 sgn(\dot{\phi}_1) \\ F_2 sgn(\dot{\phi}_2) \end{bmatrix}, \]  
(A.5)

where \( M_{ij} \) is the component \((i, j)\) of \( M \)

\[ C_{12} = C_{21} = m_2 l_2 k_e(\phi_1 + \phi_2), \quad C_{11} = V_1, \quad C_{22} = V_2, \]

where \( V_1, V_2 \) are the viscous friction coefficients of the arm joints, \( F_1, F_2 \) are the joint Coulomb friction torques and \( k_e \) is the actuator encoder constant. Dynamics parameters of the robots are given in Table 2.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Robot 1</th>
</tr>
</thead>
<tbody>
<tr>
<td>( l_1 )</td>
<td>0.4 m</td>
</tr>
<tr>
<td>( l_2 )</td>
<td>0.35 m</td>
</tr>
<tr>
<td>( m_2 )</td>
<td>10 kg</td>
</tr>
<tr>
<td>( I_1 )</td>
<td>1.8 kg m^2</td>
</tr>
<tr>
<td>( I_2 )</td>
<td>0.041 kg m^2</td>
</tr>
<tr>
<td>( C_{11} ) ((= V_1))</td>
<td>5 kg m/s</td>
</tr>
<tr>
<td>( C_{12} ) ((= V_1))</td>
<td>2 kg m/s</td>
</tr>
<tr>
<td>( F_1, F_2 )</td>
<td>0</td>
</tr>
<tr>
<td>( k_e )</td>
<td>0.5</td>
</tr>
</tbody>
</table>
In practice, parameters such as $V_1$, $V_2$, $F_1$ and $F_2$ are not known accurately and the dynamic model of the robot is further complicated by the presence of factors like clearances in bearings and backlash in the transmission system. Note that although Eq. (12) is non-linear, it also incorporates linear terms. This motivates the idea of using a hybrid neural network with both linear and non-linear neurons to represent the robot dynamics.

References


Dr. Şahin Yıldırım was born in Nevşehir, TURKEY, in 1965. He received the Mechanical Engineering degree from the Erciyes University of Kayseri, Turkey, the MS degree in mechanical engineering from the Erciyes University, Kayseri, Turkey, and the Ph.D. degree from the Wales University of Cardiff, UK in 1989, 1991, 1998, respectively. Since 1998, he has been with the Mechanical Engineering Department, Erciyes University, where he currently is a full professor. His research areas of interest are artificial neural networks, robotics, vibrations and adaptive control. He has authored or co-authored over 50 referred journal and conference proceeding papers in the above areas.