A proposed neural internal model control for robot manipulators

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

Received 07 July 2005; accepted 18 April 2006

This paper describes the design of a Neural Internal Model Control (NIMC) system for robots, based on Recurrent Hybrid Networks (RHNs). The NIMC, an alternative to the basic inverse control scheme, consists of a forward internal neural model of robot, a neural controller and a conventional feedback controller. An Alopex Learning Algorithm (ALA) was used to adjust weights of the proposed neural network. Backpropagation (BP) algorithm is also employed for comparison. Diagonal Recurrent Networks (DRNs) and Feedforward Neural Networks (FNNs) controllers were used for comparison. The robot in this study was adept one SCARA type robot manipulator.

Keywords: Alopex learning algorithm, Back propagation, Diagonal recurrent network, Feed forward neural network, Internal model control, Recurrent hybrid network

IPC Code: B25J9/00; G05B13/04

Introduction

Robots are systems with highly coupled non-linear dynamics and parametric uncertainties. The ability of Neural Networks (NNs) to represent non-linear relations leads to the idea of using a NN directly in a model-based control strategy. This is due to the possibility of training NNs to learn both a system's input-output relationship and its corresponding inverse relationship. The use of NNs for the identification and control, through Internal Model Control (IMC) strategy, of two nonlinear systems has been presented. The systems considered were a chemical stirred tank reactor and a single link manipulator. Network was fully connected to Hopfield networks. A suitable control strategy, which directly incorporates the plant model, was provided by IMC method. An adaptation law has been proposed for IMC control of nonlinear processes. The idea of employing NNs for non-linear IMC method has been considered. A technique, using NNs directly, was proposed for the adaptive control of non-linear systems. The realization of inverse model control using NNs is straightforward; the system forward dynamics model and the controller were simply implemented by means of NN models.

This paper presents a recurrent hybrid neural NN that is a network with feedback connections and thus an inherent memory for dynamics, and describes its use in the control of a simulated two-joint robot arm. A feature of the network is its hybrid-hidden layer, which includes both linear and non-linear neurons. This facilitates learning by the network of the internal model of the robot, which can be thought of as comprising a linear and a non-linear part. Main objective is to find robust controller for robot manipulator after dynamic changes.

Proposed Recurrent Hybrid Networks

Recurrent Hybrid Neural Networks (RHNNs) for forward internal model (Fig. 1), computes the joint torque $\tau_m$ required to generate model joint rotation $\phi_m$. The controller produces torque $\tau_N$ in response to the desired angular rotation $\phi_d$. The proposed recurrent network can be represented in a general diagrammatic form (Fig. 2), which depicts the hybrid hidden layer as comprising a linear part and a non-linear part and shows that, in addition to the usual feedforward connections, the networks also have feedback connections from the output layer to the hidden layer and self-feedback connections in the hidden layer.

At a given discrete time $t$, let $u(t)$ be input to a recurrent hybrid network, $y(t)$ the output of network, $x_1(t)$ the output of the linear part of the hidden layer, and $x_2(t)$ the output of non-linear part of hidden layer. Network operation is summarized in following equations (Fig. 2):

---

*E-mail: sahiny@erciyes.edu.tr
\[ x(t+1) = W_I u(t+1) + \beta x(t) + \alpha J_1 y(t), \quad \ldots \quad (1) \]

\[ x_2(t+1) = F(W^{H2} u(t+1) + \beta x_2(t) + \alpha J_2 y(t)), \quad \ldots \quad (2) \]

\[ y(t) = W^{H1} x_1(t) + W^{H2} x_2(t+1), \quad \ldots \quad (3) \]

where \( W^{H1} \) is the vector of weights of connections between input layer and linear hidden layer, \( W^{H2} \) is the vector of weights of connections between input layer and non-linear hidden layer, \( W^{H1} \) is the vector of weights of connections between linear hidden layer and output layer, \( W^{H2} \) is the vector of weights of the

connections between non-linear hidden layer and output layer, \( F() \) is the activation function of neurons in non-linear hidden layer and \( \alpha \) and \( \beta \) are the weights of self-feedback and output feedback connections. The feedback connection weights for the links from hidden layer to state layer have the same value \( \gamma \), and connections from the state layer to itself have also the same value \( \beta \). \( J_1 \) and \( J_2 \) are respectively \( n_{H1} \times n_{O} \) and \( n_{H2} \times n_{O} \) matrices with all elements equal to 1, where \( n_{H1} \) and \( n_{H2} \) are the numbers of linear and non-linear hidden neurons, and \( n_{O} \), the number of output neurons.

If only linear activation is adopted for hidden neurons, above equations simplify to:

\[ y(t+1) = W^{H1} x(t+1), \quad \ldots \quad (4) \]

\[ x(t+1) = W_I u(t+1) + \beta x(t) + \alpha J_1 y(t), \quad \ldots \quad (5) \]

Replacing \( y(t+1) \) by \( W^{H1} x(t+1) \) in Eq. (5) gives

\[ x(t+1) = (\beta I + \alpha J_1 W^{H1}) x(t) + W_I u(t+1), \quad \ldots \quad (6) \]

where \( I \) is a \( n_{H1} \times n_{H1} \) identity matrix.

Eq. (6) is of the form

\[ x(t+1) = Ax(t) + Bu(t+1), \quad \ldots \quad (7) \]

where \( A = \beta I + \alpha J W^{H1} \) and \( B = W_I \). Eq. (7) represents the state equation of a linear system of which \( x \) is the state vector. The elements of \( A \) and \( B \) can be adjusted through training so that any arbitrary linear system of order \( n_{H1} \) can be modeled by the given network. When non-linear neurons are adopted, this gives network the ability to perform non-linear dynamics mapping and thus model non-linear dynamic systems. The existence in the recurrent hybrid network of a hidden layer with both linear and non-linear neurons facilitates the modeling of practical non-linear systems comprising linear and non-linear parts. The inverse dynamics equation of the robot includes both linear terms and non-linear terms.

In this work, the values of the weights of the recurrent connections, \( \alpha \) and \( \beta \), are fixed. This means only the weights of the feedforward connections, \( W^I \) and \( W^{H1} \), need to be adjusted and thus the standard backpropagation algorithm can be employed to train the neural internal model.

**Diagonal Recurrent Networks**

Diagonal Recurrent Network (DRN) \(^8\) is different from the RHN in two respects: i)There are no feedback connections from the output layer to the
hidden layer; and ii) The self-feedback connections in the hidden layer are all trainable (Fig. 3). The operation of the DRN can be described as follows:

\[ x_j(t+1) = F(S_j(t+1)), \ldots \quad (8) \]

\[ S_j(t+1) = I_{n_H} H I_{n_H}^{j j k} k 1 x(t)n(t 1) + \sum W W u(t), \ldots \quad (9) \]

\[ y_j(t+1) = \sum W^O_{ij} x_j(t+1). \ldots \quad (10) \]

where, hidden layer output \( x(t+1) \in \mathbb{R}^{n_H} \), lateral hidden layer weights \( W^H \in \mathbb{R}^{n_H \times n_H} \), input layer weights \( W^I \in \mathbb{R}^{n_I \times n_H} \), and output layer weights \( W^O \in \mathbb{R}^{n_O \times n_H} \); \( n_I \), \( n_H \) and \( n_O \) are numbers of neurons in the input, hidden and output layers respectively; \( u(t+1) \) represents the input vector and \( F(\cdot) \) 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.

Computed Torque Method (CTM)

The difficulties in controlling a robot manipulator are inherent nonlinearity and high coupling. Because of these problems, much of linear control theory is not directly applicable, and as a result, computed torque methods based on either the Lagrange-Euler formulation or the Newton-Euler formulation have been proposed for robot control. For, CTM based on the Lagrange-Euler formulation, the control input is

\[ u = [\ddot{q} + K_p \dot{e} + K_p \dot{e} + C(q, \dot{q}) + G(q) + F_c(q, \dot{q})] \quad \ldots \quad (11) \]

where, \( K_D \) is an \( n \times n \) derivative gain matrix, \( K_P \) is an \( n \times n \) proportional gain matrix, \( C(q, \dot{q}) \) is an \( n \times 1 \) vector of centrifugal and Coriolis terms, \( G(q) \) is an \( n \times 1 \) vector of gravity terms and \( F_c(q, \dot{q}) \) is an \( n \times 1 \) vector of Coulomb friction terms. If the dynamic model is exactly matched to the actual system, the error equation is governed by

\[ \ddot{e} + K_D \dot{e} + K_P e = 0. \quad \ldots \quad (12) \]

Then, by choosing PD gains such that Eq. (12) has negative real roots, \( e \) approaches zero asymptotically. When the gain matrices are chosen to be diagonal such that \( K_{D,k} = K_{P,k} \) for all \( k \), Eq. (12) is critically damped. Due to modelling errors, mismatched terms appear on the right-hand side of Eq. (11).

CTM using dynamic information on the robot manipulator performs better than a simple PID method. However, its implementation requires accurate knowledge of terms such as \( F_c(q, \dot{q}) \) which are difficult to obtain in practice.

IMC using Neural Networks

A two-step procedure for using a NN directly within the IMC structure is proposed. The first step involves training a network to represent the robot forward dynamics (Fig. 4). This network is used as a model of the robot in the IMC structure (Fig. 5). In addition to the control loop, the system also included an inner-loop PID controller that acted directly on the joint motors to provide stable speed control. The error signal \( e \) is used to adjust the weights of the neural controller with sensitivity information (Fig. 6) obtained for this purpose from the neural model of the robot. Let \( \phi_d(t) \) and \( \phi_i(t) \) be the desired and actual responses of joint \( i \) of the robot. The weights of the proposed neural controller are adjusted using ALA as follows:

\[ \phi_d(t) + K_p \dot{e} + K_p \dot{e} + C(q, \dot{q}) + G(q) + F_c(q, \dot{q}) \]

Fig. 3—Schematic representation of the DRNN
Consider a neuron $i$ with an interconnection strength $w_{ij}$ from neuron $j$ in the lower layer. The output of the neuron $I_i$, during the $n^\text{th}$ iteration, is given by:

$$net_i(n)= \sum_j w_{ij}(n)out_j(n)+\theta_i(n), \quad ... \ (13)$$

where, $\theta_i$ = threshold of neuron $i$.

Applying a sigmoidal transformation to $net_i$, output of the hidden layer in the $i$th neuron can be obtained as follows:

$$out_i(n)=\frac{1-\exp(-net_i(n)/T_0)}{1+\exp(-net_i(n)/T_0)}, \quad ... \ (14)$$

where $T_0$ is the sigmoidal gain.

During the $n^\text{th}$ iteration, the weight $w_{ij}$ is updated as,

$$w_{ij}(n)=w_{ij}(n-1)+\delta_{ij}(n)$$
$$\theta_i(n)=\theta_i(n-1)+\delta_i(n), \quad ... \ (15)$$

where $\delta_i(n)$ would have a small positive or negative step of size $\delta$ with following probabilities:

$$\delta_{ij}(n)=-\delta \text{ with probability } P_{ij}(n)$$
$$\delta_{ij}(n)=+\delta \text{ with probability } (1-P_{ij}(n)), \quad ... \ (16)$$

The probability $P_{ij}(n)$ is given by following expression:

$$P_{ij}(n)=\frac{1}{1+\exp(-\Delta_{ij}(n)/T)}, \quad ... \ (17)$$

where $\Delta_{ij}(n)$ is given by the correlation:

$$\Delta_{ij}(n)=\Delta w_{ij}(n)\Delta E(n), \quad ... \ (18)$$

$\Delta w_{ij}(n)$ and $\Delta E(n)$ are the changes in the weight $w_{ij}$ and the error measure $E$ over the previous two iterations, $E(n)=\frac{1}{2}(\sum_i (\varphi_{di}(n)\varphi_i(n))$ . For the first two iterations, $P_{ij}(n)$ was taken as 0.5. In the
expression for \( P_{\beta}(n) \), \( T \) is a positive temperature that determines the effective randomness in the system. The stochastic approach of the ALA differs from the instantaneous learning schemes such as backpropagation.

**Applications to the Control of a Robot**

This section describes the results of a simulated implementation of NIMC to the control of the adept one SCARA robot. The plant to be controlled is the simplest example of an articulated robot arm (Fig. 7). This is a planar device comprising two main links with two actuated joints. Actuator 1 applies torque \( \tau_1 \) to drive joint 1 that directly connects link 1 (with inertia \( I_1 \)) to the base of the arm. Actuator 2 applies torque \( \tau_2 \) to drive joint 2 (with inertia \( I_2 \)) connecting link 2 (with inertia \( I_3 \) and carrying payload \( m_2 \)) to link 1.

The distance between the axes of joints 1 and 2 is \( l_1 \) and the distance between the concentrated payload \( m_2 \) and the axis of joint 2 is \( l_2 \). The angles of rotation of the actuators are \( \phi_1, \phi_2 \), respectively. The dynamics equations of the robot arm can be written as follows \(^{10,11} \):

\[
\begin{align*}
\tau &= M(\phi)\ddot{\phi} + C(\phi, \dot{\phi}) + F(\phi, \dot{\phi}) \quad \cdots \quad (19)
\end{align*}
\]

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.

![Fig. 7—Schematic configuration of the Adept one robot](image)

\[
M = \begin{bmatrix}
I_1 + m_2 l_1^2 & -m_2 l_1 l_2 \cos(k_e(\phi_1 + \phi_2)) \\
-m_2 l_1 l_2 \cos(k_e(\phi_1 + \phi_2)) & I_2 + I_3 + m_2 l_2^2
\end{bmatrix}
\]

\[
C = \begin{bmatrix}
m_2 l_1 l_2 k_e \sin(k_e(\phi_1 + \phi_2)) \dot{\phi}_2 \\
m_2 l_1 l_2 k_e \sin(k_e(\phi_1 + \phi_2)) \dot{\phi}_1
\end{bmatrix}
\]

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

Eq. (19) can be rewritten as:

\[
\begin{bmatrix}
\tau_1 \\
\tau_2
\end{bmatrix} = 
\begin{bmatrix}
M_{11} & M_{12} \\
M_{21} & M_{22}
\end{bmatrix}
\begin{bmatrix}
\dot{\phi}_1 \\
\dot{\phi}_2
\end{bmatrix} + 
\begin{bmatrix}
C_{11} & C_{12} \\
C_{21} & C_{22}
\end{bmatrix}
\begin{bmatrix}
\phi_1 \\
\phi_2
\end{bmatrix} + 
\begin{bmatrix}
F_1 \text{sgn}(\dot{\phi}_1) \\
F_2 \text{sgn}(\dot{\phi}_2)
\end{bmatrix}
\]

where, \( M_{ij} \) is component \((i, j)\) of \( M \); \( C_{ij} = C_{ji} = m_2 l_1 l_2 k_e \sin(k_e(\phi_1 + \phi_2)) \); \( F_{ij} = F_{ji} = V_i \); \( C_{11} = V_1; \ C_{22} = V_2; \ V_1, \ V_2 \) are viscous friction coefficients of the arm joints, and \( F_1, F_2 \) are joint Coulomb friction torques and \( k_e \) is the actuator encoder constant.

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. Although Eq. (23) is non-linear, it also incorporates linear terms. This motivates the idea of using a hybrid NN with both linear and non-linear neurons to represent the robot dynamics.

The RHN, DRN and Feedforward Neural Network (FNN) structures were employed in this study, using backpropagation (BP) and Alopex learning algorithms (Table 1).

**Simulation I**

In this simulation, the RHN structure was adopted as that of the neural controllers and neural model. The neural controller was trained when the robot had a payload of 10 kg. To demonstrate the adaptability of the proposed control scheme, payload was suddenly

<table>
<thead>
<tr>
<th>NN</th>
<th>( \eta )</th>
<th>( \mu )</th>
<th>( \alpha )</th>
<th>( \beta )</th>
<th>( N )</th>
<th>( n )</th>
</tr>
</thead>
<tbody>
<tr>
<td>RHN</td>
<td>0.001</td>
<td>0.1</td>
<td>0.6</td>
<td>0.6</td>
<td>60000</td>
<td>8+8</td>
</tr>
<tr>
<td>DRNN</td>
<td>0.001</td>
<td>0.1</td>
<td>—</td>
<td>0.6</td>
<td>60000</td>
<td>16</td>
</tr>
<tr>
<td>FNN</td>
<td>0.001</td>
<td>0.1</td>
<td>—</td>
<td>—</td>
<td>60000</td>
<td>16</td>
</tr>
</tbody>
</table>
changed from 10 kg to 30 kg. The feedback gains of network $\alpha$ and $\beta$ were selected empirically and set to $\alpha = 0.6$ and $\beta = 0.6$. The result (Fig. 8a) shows that a 300 percent load increase can be accommodated by this controller, which is a very good measure of its robustness (Fig. 8b). The results of proposed NN were compared with BP learning algorithm (Fig. 9; Table 2).

Simulation II

In this simulation, use of DRNNs for the modeling and control of the SCARA robot was demonstrated. The result obtained after the dynamics change is plotted (Fig. 10a). The response of the control system is also shown (Fig. 10b). The BP algorithm was also employed as a learning algorithm for this network (Fig. 11a). The root mean squared errors (RMSE) for different controllers were calculated (Table 3).

<table>
<thead>
<tr>
<th>$m_0$ kg</th>
<th>Learning Algorithm</th>
<th>RMSE First trial</th>
<th>RMSE Second trial</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>ALA</td>
<td>0.002402</td>
<td>0.000468</td>
</tr>
<tr>
<td>20</td>
<td>ALA</td>
<td>0.002609</td>
<td>0.000469</td>
</tr>
<tr>
<td>90</td>
<td>ALA</td>
<td>0.007836</td>
<td>0.000532</td>
</tr>
<tr>
<td>0</td>
<td>BP</td>
<td>0.002412</td>
<td>0.000470</td>
</tr>
<tr>
<td>20</td>
<td>BP</td>
<td>0.002633</td>
<td>0.0004720</td>
</tr>
<tr>
<td>90</td>
<td>BP</td>
<td>0.008178</td>
<td>0.000538</td>
</tr>
</tbody>
</table>
Table 3—Performance of the proposed control system using different networks

<table>
<thead>
<tr>
<th>m_L</th>
<th>NN</th>
<th>RMSE First trial</th>
<th>RMSE Second trial</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>RHN</td>
<td>0.002521</td>
<td>0.000470</td>
</tr>
<tr>
<td>20</td>
<td>RHN</td>
<td>0.002640</td>
<td>0.000471</td>
</tr>
<tr>
<td>90</td>
<td>RHN</td>
<td>0.007441</td>
<td>0.000529</td>
</tr>
<tr>
<td>0</td>
<td>DRNN</td>
<td>0.002649</td>
<td>0.001472</td>
</tr>
<tr>
<td>20</td>
<td>DRNN</td>
<td>0.003202</td>
<td>0.001479</td>
</tr>
<tr>
<td>90</td>
<td>DRNN</td>
<td>0.008182</td>
<td>0.001480</td>
</tr>
<tr>
<td>0</td>
<td>FNN</td>
<td>0.002962</td>
<td>0.001967</td>
</tr>
<tr>
<td>20</td>
<td>FNN</td>
<td>0.004179</td>
<td>0.002418</td>
</tr>
<tr>
<td>90</td>
<td>FNN</td>
<td>0.012380</td>
<td>0.007935</td>
</tr>
</tbody>
</table>

Fig. 10(b)—Control response of robot (after dynamic change, using the ALA-DRNN)

Fig. 11—Desired and actual trajectories of the end-effector (after dynamic change, using a) BP-DRNN, b) ALA-FNN, c) BP-FNN, d) PID controller)
Simulation III

In the third simulation, FNNs for modelling and control of the adept one robot was utilized (Table 1). The trajectories results of the end-effector for desired and actual are given in Fig. 11b. BP algorithm was utilized to update the weight of the network. Fig. 11c represents desired and actual trajectories of the robot with FNN-BP. The error between desired and actual plot were increased by 15 times in RMSE (Table 3).

Simulation IV

In the last simulation, PID controller for controlling control of the adept one robot was employed. The trajectories results of the end-effector for desired and actual are considerably bigger than the proposed neural control approach (Fig. 11d).

Discussion

The simulation results supported that the proposed hybrid recurrent NN was able to represent different types of dynamic systems. The network was trainable using the simple BP algorithm to model both the forward dynamics and the inverse dynamics of a variety non-linear multi-input multi-output plant. The adoption of both linear and non-linear neurons in the network facilitates learning because non-linear plants often can be decomposed into a linear part than can be modeled by the linear neurons and a non-linear part that can be learnt by non-linear neurons.

In the proposed neural control scheme tested, the performance as measured by the ability to follow prescribed trajectories under conditions of varying or unknown dynamic disturbances was better than that of conventional PID controller. Also, the proposed recurrent neural control scheme was considerably easier to implement than conventional schemes, which were very sensitive to tuning. The IMC system employing a RHN controller, because the inclusion of both linear and non-linear neurons in a RHN, produced the best performance. 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. In contrast, in the DRN as proposed by Ku & Lee, non-linear neurons only had to learn both the linear and non-linear parts with those neurons, which gave less accurate results. Also, the results of the proposed learning algorithm (ALA) are better than the standard BP algorithm.

Conclusions

This paper has presented the use of recurrent NNs with ALA to implement internal model control (IMC) of a SCARA robot arm. Both the controller and the model were realized with the proposed NNs. Other two types, DRN and FNN, neural networks have also presented for comparison with the proposed NN. The RHN structure previously described was shown to produce better training performance and adaptability than the DRN structure proposed by other researchers and standard FNN structure. ALA-RHN model has great potential; it can be used to develop control and prediction on different models.

References