Robotica (2000) volume 18, pp. 505–512. Printed in the United Kingdom © 2000 Cambridge University Press

Design of a neural internal model control system for a robot

D.T. Pham and Şahin Yildirim

Intelligent Systems Research Laboratory, Systems Engineering Division, School of Engineering, University of Wales Cardiff, Newport Road, P.O. Box 688, Cardiff CF24 3TE (UK)

(Received in Final Form: September 7, 1999)

SUMMARY

This paper describes the design of an Internal Model Control (IMC) system for a planar two-degree-of-freedom robot. IMC was investigated as an alternative to the basic inverse control scheme which is difficult to implement. The proposed IMC system consisted of a forward internal neural model of the robot, a neural controller and a conventional feedback controller, all of which were realised easily. Both the neural model and the neural controller were based on recurrent networks which were trained using the back-propagation (BP) algorithm. The paper presents the results obtained with two types of recurrent networks as well as a conventional PID system.

KEYWORDS: Internal model control; Recurrent neural network; Diagonal recurrent network; Backpropagation; Robot control.

1. INTRODUCTION

Robots are systems with highly coupled non-linear dynamics and parametric uncertainties. If a robot's characteristics are known, computed torque and non-linear decoupling controllers can be used to deal with non-linearities and uncertainties and achieve satisfactory trajectory tracking performance.[1,2] However, the chief drawback of those controllers arises from the fact that they require exact cancellation of non-linear terms in order to obtain a linear input-output behaviour.

To improve the performance of robots in non-linear and uncertain environments, considerable research has been focused on developing advanced controllers in recent years. A modem control strategy, which incorporates a direct model of the plant, is provided by the Internal Model Control (IMC) method.[3,4] Recently, it was shown that IMC was a simple and effective technique for designing the underlying control law in a new approach to adaptive robust control of a stable plant.[5] The applicability of IMC to the control of non-linear systems was demonstrated by Economou et al.[6] The inverse model of the plant was shown to play a crucial role in the implementation of the non-linear IMC method. Economou et al. studied analytical and numerical methods for the construction of the required non-linear inverse model.

The idea of employing NNs for non-linear IMC was considered by Bhat and McAvoy.[7] A technique, using NNs directly, was proposed for the adaptive control of non-linear systems by Hunt and Sbarbaro.[8]. The control structure adopted was also IMC. This structure was used to incorporate neural network modelling of the plant and its inverse directly within the control strategy.

The realisation of IMC using NNs is simple: the system forward dynamics model and the controller are implemented by means of NN models.[9] In the work reported in this paper, the NNs were recurrent networks. Such networks possess feedback connections enabling them to have an inherent memory for dynamics.

The paper first describes the two types of recurrent networks experimented with and then discusses their incorporation in an IMC system and application to the modelling and control of a simulated planar two-jointed robot arm. The paper presents simulation results for IMC schemes employing these two types of networks as well as a conventional scheme using only a standard PID controller.

2. FIRST TYPE OF RECURRENT NETWORKS

Figures l(a) and l(b) show the detailed configuration of two type-1 neural networks used as forward internal model and controller. The forward internal model computes the joint rotation φ_m corresponding to a joint torque τ_m. The controller produces torque τ_N aimed at generating the desired angular rotation φ_d. Each network can be represented in a general diagrammatic form as illustrated in Figures 2(a) and 2(b).

These figures depict the hybrid hidden layer of the network as comprising a linear part and a non-linear part and show 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 $\mathbf{u}(t)$ be the input to the network, $\mathbf{y}(t)$, the output of the network, $\mathbf{x_1}(t)$ the output of the linear part of the hidden layer and $\mathbf{x_2}(t)$ the output of the non-linear part of the hidden layer.

The operation of the network is summarised by the following equations (see also Figure 2(b)):

$$\mathbf{x}_1(t+1) = \mathbf{W}^{I1}\mathbf{u}(t+l) + \beta\mathbf{x}_1(t) + \alpha\mathbf{J}_l\mathbf{y}(t) \quad (1)$$

$$\mathbf{x}_2(t+l) = \mathbf{F}\{\mathbf{W}^{I2}\mathbf{u}(t+l) + \beta\mathbf{x}_2(t) + \alpha\mathbf{J}_2\mathbf{y}(t)\} \quad (2)$$

$$\mathbf{y}(t+l) = \mathbf{W}^{H1}\mathbf{x}_l(t+l) + \mathbf{W}^{H2}\mathbf{x}_2(t+l) \quad (3)$$

where $\mathbf{W}^{I1}$ is the vector of weights of the connections between the input layer and the linear hidden layer, $\mathbf{W}^{I2}$ is the vector of weights of the connections between the input layer and the non-linear hidden layer, $\mathbf{W}^{H1}$ is the vector of weights of the connections between the linear hidden layer and the output layer, $\mathbf{W}^{H2}$ is the vector of weights of the

https://doi.org/10.1017/S0263574799002179 Published online by Cambridge University Press

connections between the non-linear hidden layer and the output layer, $\mathbf{F}\{\}$ is the activation function of neurons in the non-linear hidden layer and α and β are the weights of the self-feedback and output feedback connections. $\mathbf{J}_1$ and $\mathbf{J}_2$ 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 the hidden neurons, the above equations simplify to:

$$\mathbf{y}(t+1) = \mathbf{W}^{HI}\mathbf{x}(t+1) \tag{4}$$

$$\mathbf{x}(t+1) = \mathbf{W}^{I1}\mathbf{u}(t+1) + \beta\mathbf{x}(t) + \alpha\mathbf{J}_1\mathbf{y}(t) \tag{5}$$

Replacing $\mathbf{y}(t+1)$ by $\mathbf{W}^{H1}\mathbf{x}(t+1)$ in equation (5) gives

$$\mathbf{x}(t+1) = (\beta\mathbf{I} + \alpha\mathbf{J}_1\mathbf{W}^{H1})\mathbf{x}(t) + \mathbf{W}^{I1}\mathbf{u}(t+1) \tag{6}$$

where $\mathbf{I}$ is a $n_{H1} \times n_{H1}$ identity matrix

Equation (6) is of the form

$$\mathbf{x}(t+1) = \mathbf{A}\mathbf{x}(t) + \mathbf{B}\mathbf{u}(t+1) \tag{7}$$

where $\mathbf{A} = \beta\mathbf{I} + \alpha\mathbf{J}\mathbf{W}^{H1}$ and $\mathbf{B} = \mathbf{W}^{I1}$. Equation (7) represents the state equation of a linear system of which $\mathbf{x}$ is the state vector. The elements of $\mathbf{A}$ and $\mathbf{B}$ can be adjusted through training so that any arbitrary linear system of order n_{H1} can be modelled by the given network. When non-linear neurons are adopted, this gives the network the ability to perform non-linear dynamics mapping and thus model non-linear dynamic systems. The existence in the network of a hidden layer with both linear and non-linear neurons facilitates the modelling of practical non-linear systems comprising linear and non-linear parts.

In this work, the values of the weights of the recurrent connections, α and β, were fixed. This means only the weights of the feedforward connections, $\mathbf{W}^I$ and $\mathbf{W}^H$, needed to be adjusted and this allowed the use of the standard backpropagation algorithm to train the neural internal model and controller.

3. SECOND TYPE OF RECURRENT NETWORKS

Figure 3(a) shows a schematic block diagram representation of a type-2 recurrent network. This type of network was used for control purposes in the work of Ku et al.[10] It differs from the type-1 network presented above in three respects: there are no linear neurons in the hidden layer and 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 network can be described by the following equations (see also Figure 3(a)):

$$\mathbf{x}(t+1) = \mathbf{F}(\mathbf{S}(t+1)) \tag{8}$$

$$\mathbf{S}(t+1) = \mathbf{W}^H\mathbf{x}(t) + \mathbf{W}^I\mathbf{u}(t+1) \tag{9}$$

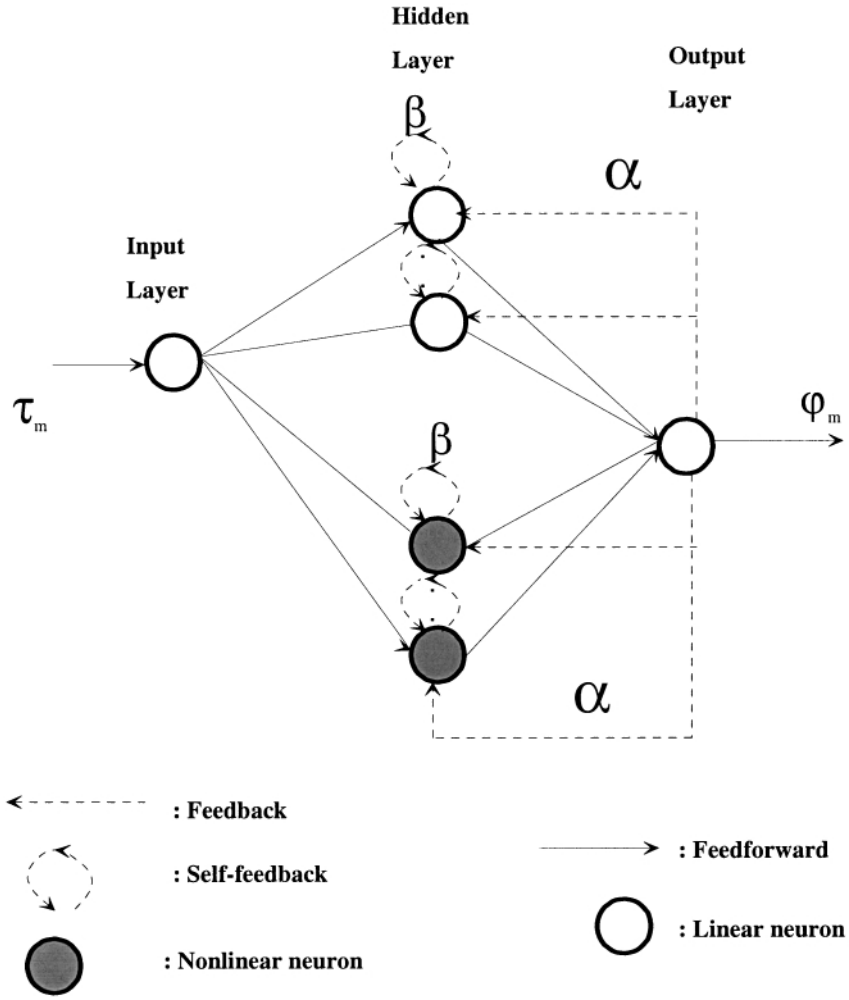

Fig. 1(a). Configuration of forward internal neural model (Type-1 network)

https://doi.org/10.1017/S0263574799002179 Published online by Cambridge University Press

$$\mathbf{y}(t+1) = \mathbf{W}^{o}\mathbf{x}(t+1) \tag{10}$$

where the hidden layer output $\mathbf{x}(t+1) \in \Re^{n_H}$, hidden layer feedback weights $\mathbf{W}^{H} \in \Re^{n_H \times n_H}$, input layer weights $\mathbf{W}^{I} \in \Re^{n_H \times n_I}$, and output layer weights $\mathbf{W}^{o} \in \Re^{n_H \times n_O}$. 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 schematic configuration of a type-2 network employed as a robot controller is shown in Figure 3(b).

4. IMC OF A ROBOT USING NEURAL NETWORKS

The first step in using a neural network within an IMC system for controlling a robot involves training a network to represent its forward dynamics (Figure 4). This network is used as a model of the robot in the IMC structure (Figure 5). A minor-loop controller (not shown in the structure of Figure 5) is also employed to stabilise the robot and simplify the design of the overall controller.[11] Specialised learning[12] for the neural controller of the robot is shown in Figure 6. In that figure, the error signal **e** is used to adjust the weights of the neural controller with sensitivity information obtained for this purpose from the neural model of the robot. Let $\varphi_{di}(t)$ and $\varphi_i(t)$ be the desired and actual responses of joint *i* of the robot. The weights of the proposed neural controller are adjusted using the BP algorithm as follows:

$$\Delta W_{kh}(t) = -\eta \frac{\partial E(t)}{\partial W_{kh}(t)} = -\eta \sum_i \frac{\partial E(t)}{\partial \varphi_i(t)} \frac{\partial \varphi_i(t)}{\partial \tau_k(t)} \frac{\partial \tau_k(t)}{\partial W_{kh}(t)} \tag{11}$$

where $W_{kh}(t)$ is the weight of the connection between neurons *h* and *k* in the controller and $E(t) = \sum_i \frac{1}{2}(\varphi_{di}(t) - \varphi_i(t))^2$. Normally, to obtain $\frac{\partial \varphi_i(t)}{\partial \tau_k(t)}$ the robot would have to be perturbed. However, the approximation $\frac{\partial \varphi_i(t)}{\partial \tau_k(t)} \approx \frac{\partial \varphi_{mi}(t)}{\partial \tau_k(t)}$ where φ_{mi} is the output of the neural model, can be employed once the neural model of the robot is well trained. Therefore, the error signal can be back-propagated to the controller via the neural model.

Consider the connection *kh* between neuron *h* in the hidden layer and neuron *k* in the output layer of the neural controller. Let $\delta_k(t)$ be defined for neuron *k* as:

$$\delta_k(t) = -\sum_i \frac{\partial E(t)}{\partial \varphi_i(t)} \frac{\partial \varphi_i(t)}{\partial \tau_k(t)} \approx -\sum_i \frac{\partial E(t)}{\partial \varphi_i(t)} \frac{\partial \varphi_{mi}(t)}{\partial \tau_k(t)} \tag{12}$$

$$\delta_k(t) = -\sum_i \frac{\partial E(t)}{\partial \varphi_i(t)} \sum_j \frac{\partial \varphi_{mi}(t)}{\partial x_j(t)} \frac{\partial x_j(t)}{\partial z_j(t)} \frac{\partial z_j(t)}{\partial \tau_k(t)} \tag{13}$$

In Equation 13, $x_j(t)$ is the output of neuron *j* in the hidden layer of the neural model, $z_j(t)$ the net input to neuron *j* and

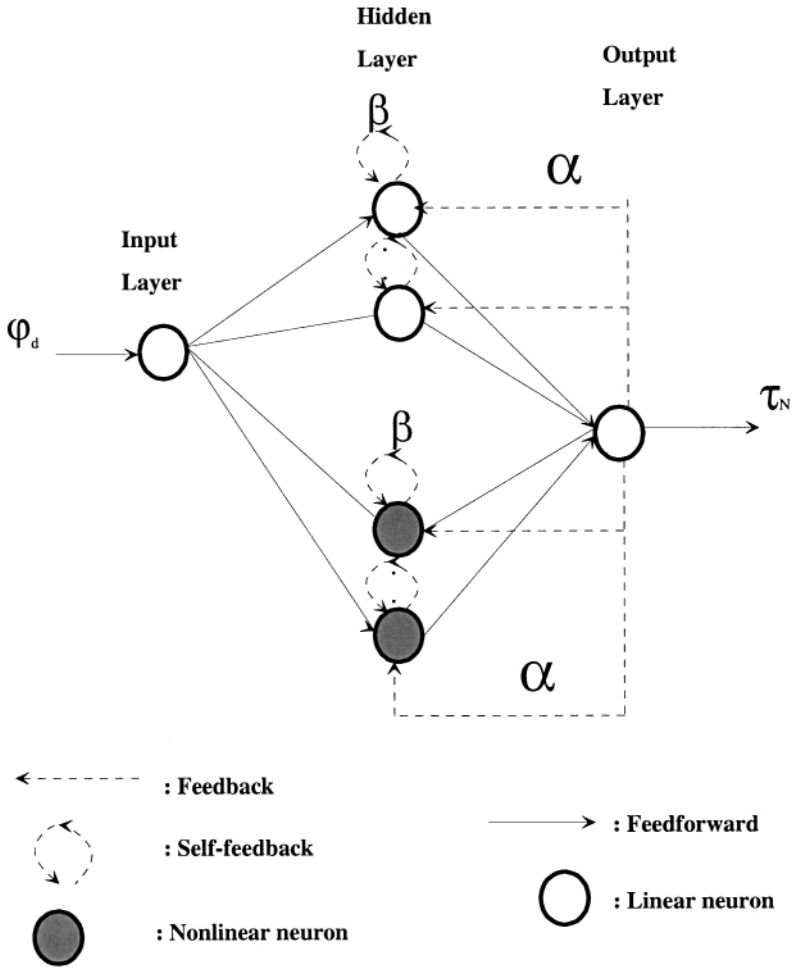

Fig. 1(b). Configuration of neural controller (Type-1 network)

https://doi.org/10.1017/S0263574799002179 Published online by Cambridge University Press

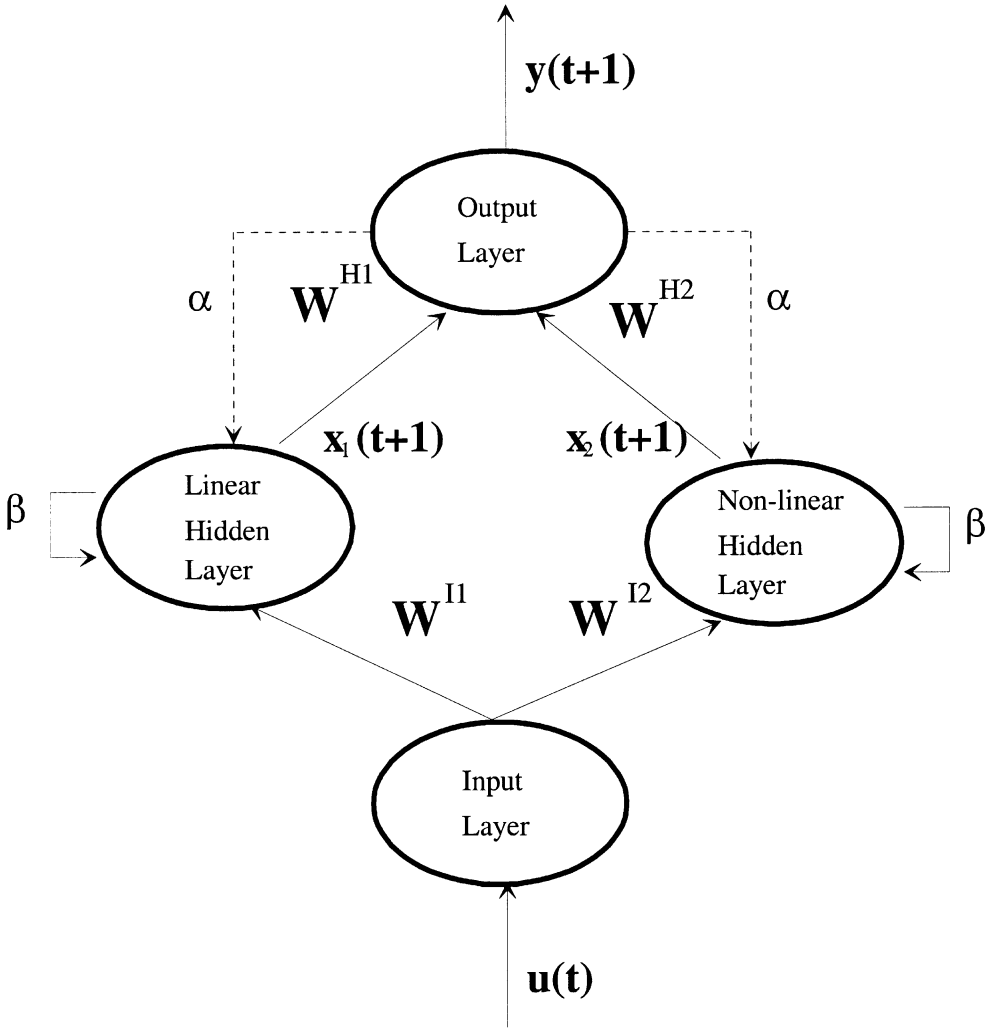

Fig. 2(a). Type-1 recurrent network

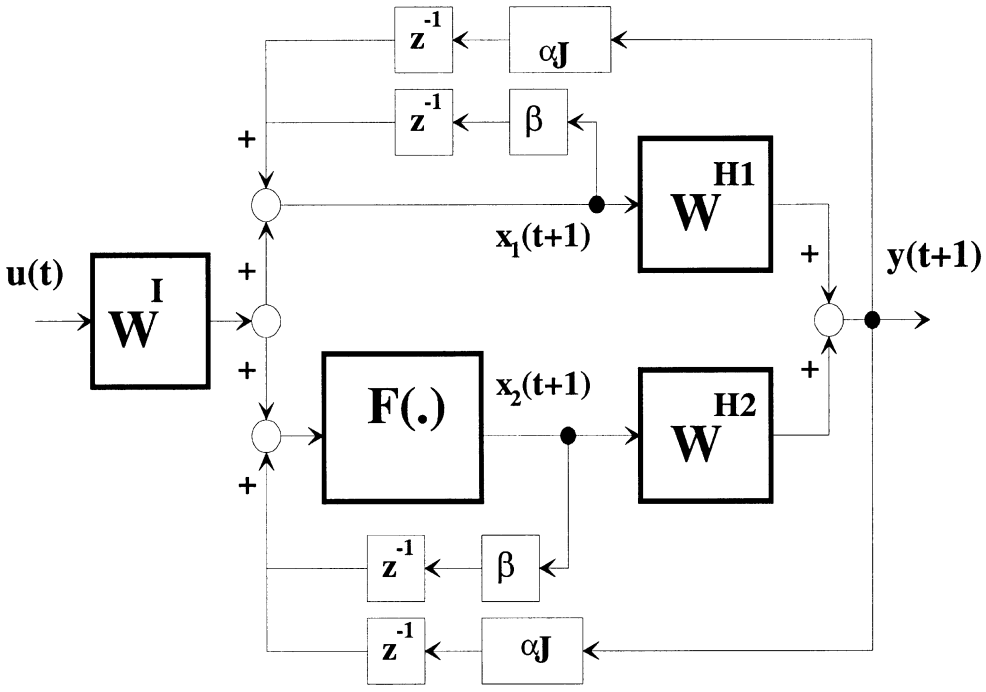

Fig. 2(b). Block diagram of Type-1 recurrent network

https://doi.org/10.1017/S0263574799002179 Published online by Cambridge University Press

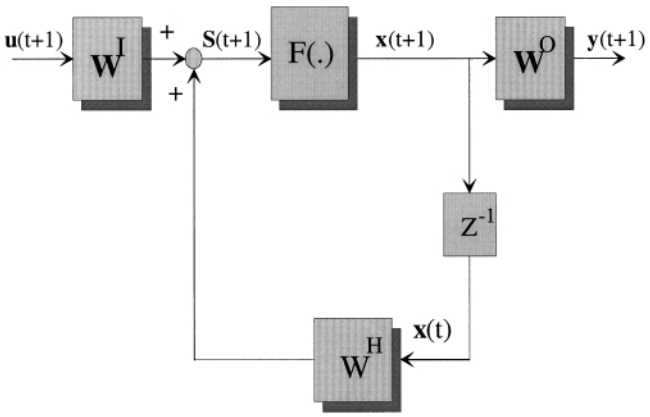

Fig. 3(a). Block diagram of Type-2 recurrent network

$\tau_k(t)$ the input to neuron *k* in the input layer of the neural model. Equation (13) can be rewritten as:

$$\delta_k(t) = \sum_j x_j(t) \sum_i e_i(t) W_{ij}(t) W_{jk}(t) \tag{14}$$

$$\delta_k(t) = \sum_j \delta_j(t) W_{jk}(t) \tag{15}$$

where

$$\delta_j(t) = x_j'(t) \sum_i e_i(t) W_{ij}(t),\ x_j'(t) = \frac{\partial x_i(t)}{\partial z_j(t)}$$

and

$$e_i(t) = \varphi_{di}(t) - \varphi_i(t)$$

Replacing Equation (15) in (11) yields the controller's

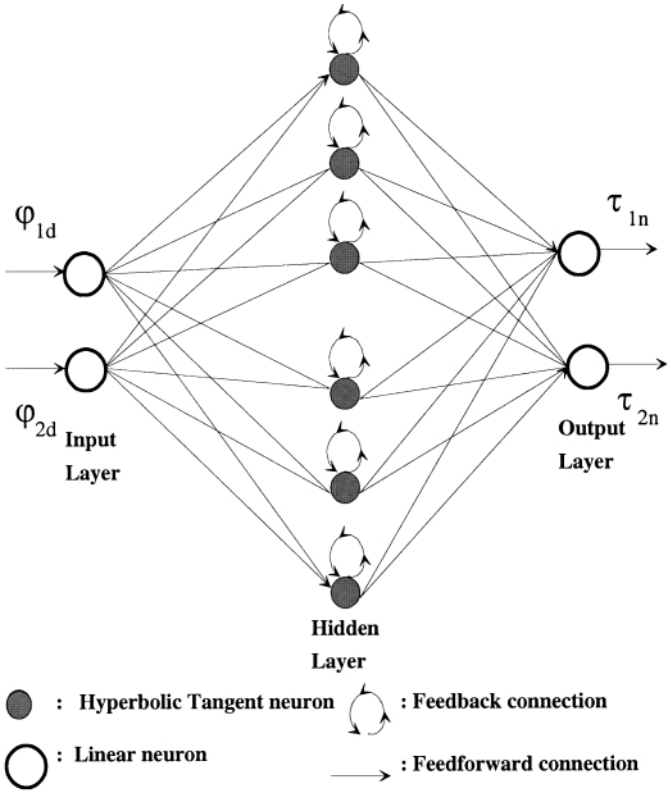

Fig. 3(b). Schematic configuration of a controller based on a Type-2 recurrent network

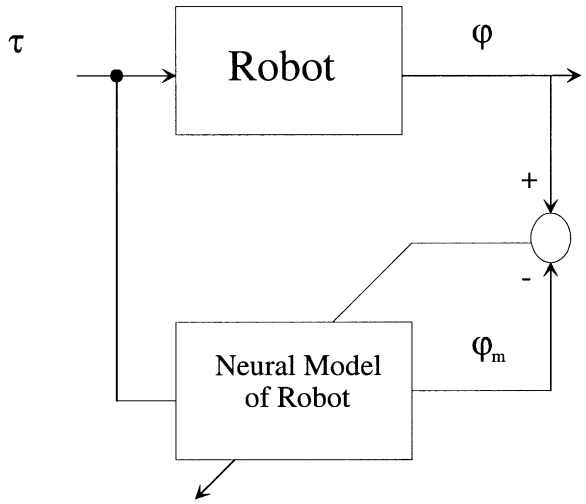

Fig. 4. Modelling of the robot's forward dynamics

output layer weight adjustment

$$\Delta W_{kh}(t) = \eta\, \delta_k(t)\, x_h(t) \tag{16}$$

where $x_h(t)$ is the output of the hidden layer neuron *h*.

The above are the standard BP equations for weight training in a multi-layered neural network as applied to the training of the weights of the connections between the

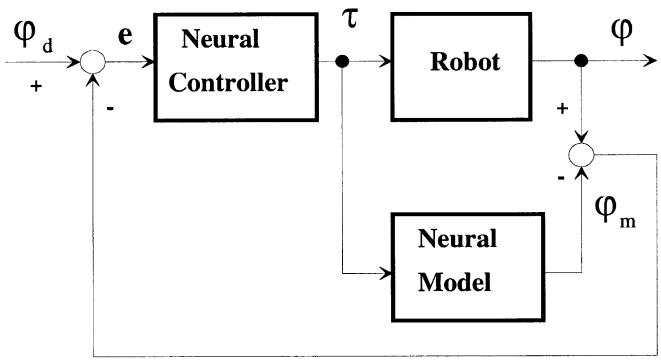

Fig. 5. Neural-network-based IMC system for robot control

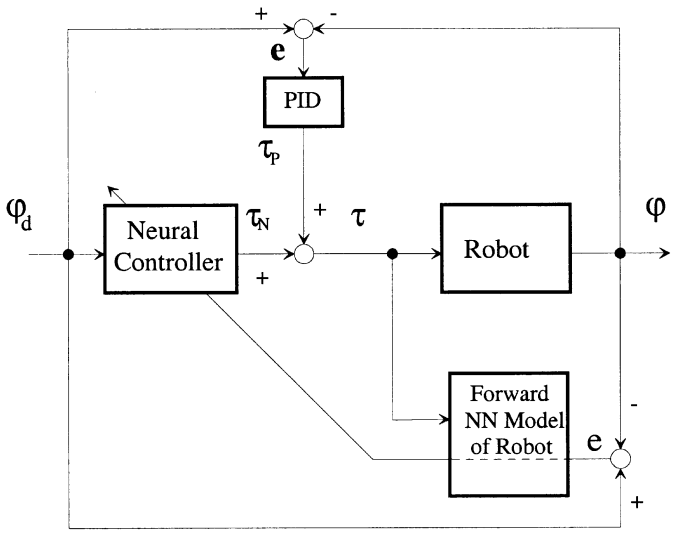

Fig. 6. Specialised learning for neural controller

https://doi.org/10.1017/S0263574799002179 Published online by Cambridge University Press

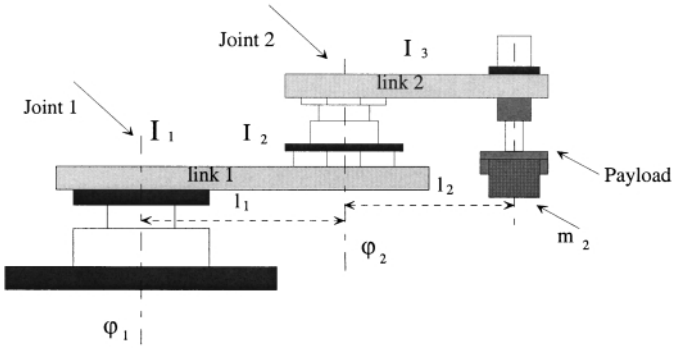

Fig. 7. Configuration of a SCARA robot

hidden layer and the output layer in the two types of recurrent networks.

The weights between the input and hidden layers can also be updated by applying the standard BP algorithm. In the case of the type-2 networks, the feedback weights were traincd using a version of the BP algorithm modified to handle recurrency.[10]

5. APPLICATION TO THE CONTROL OF A SCARA ROBOT

This section describes the results of a simulated implementation of IMC to the control of a two-jointed (SCARA) robot.

The plant to be controlled is the simplest example of an articulated robot arm (Figure 7). This planar device comprises two main links with two actuated joints. Actuator 1 applies torque τ_1 to drive joint 1 that directly connects link 1 (with inertia I_1) to the base of the arm. Actuator 2 applies torque τ_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 I 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 φ_1 and φ_2, respectively.

The dynamics equations of the robot arm can be written as follows:[9]

$$\tau = \mathbf{M}(\varphi)\ddot{\varphi} + \mathbf{C}(\varphi, \dot{\varphi}) + \mathbf{F}(\varphi, \dot{\varphi}) \tag{17}$$

where $\tau = [\tau_1\ \tau_2]^T$, $\varphi = [\varphi_1\ \varphi_2]^T$, $\dot{\varphi}$ and $\ddot{\varphi}$ are the first and second time derivatives of φ, $\mathbf{M}(\varphi)$ is the inertia matrix of the robot, $\mathbf{C}(\varphi, \dot{\varphi})$ is a 2×1 vector of centrifugal and Coriolis terms and $\mathbf{F}(\varphi, \dot{\varphi})$ is a 2×1 vector of viscous and Coulomb friction terms.

$$\mathbf{M} = \begin{bmatrix} I_1 + m_2 l_1^2 & -m_2 l_1 l_2 Cos(k_e(\varphi_1+\varphi_2)) \\ -m_2 l_1 l_2 Cos(k_e(\varphi_1+\varphi_2)) & I_2 + I_3 + m_2 l_2^2 \end{bmatrix} \tag{18}$$

$$\mathbf{C} = \begin{bmatrix} m_2 l_1 l_2 k_e Sin(k_e(\varphi_1+\varphi_2))\dot{\varphi}_2 \\ m_2 l_1 l_2 k_e Sin(k_e(\varphi_1+\varphi_2))\dot{\varphi}_1 \end{bmatrix} \tag{19}$$

$$\mathbf{F} = \begin{bmatrix} V_1\dot{\varphi}_1 + F_1 Sgn(\dot{\varphi}_1) \\ V_2\dot{\varphi}_2 + F_2 Sgn(\dot{\varphi}_2) \end{bmatrix} \tag{20}$$

Equation (17) can be rewritten as:

$$\begin{bmatrix} \tau_1 \\ \tau_2 \end{bmatrix} = \begin{bmatrix} M_{11} M_{12} \\ M_{21} M_{22} \end{bmatrix} \begin{bmatrix} \ddot{\varphi}_1 \\ \ddot{\varphi}_2 \end{bmatrix} + \begin{bmatrix} C_{11} C_{12} \\ C_{21} C_{22} \end{bmatrix} \begin{bmatrix} \dot{\varphi}_1 \\ \dot{\varphi}_2 \end{bmatrix} + \begin{bmatrix} F_1 Sgn(\dot{\varphi}_1) \\ F_2 Sgn(\dot{\varphi}_2) \end{bmatrix} \tag{21}$$

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

$$C_{12} = C_{21} = m_2 l_1 l_2 k_e Sin(k_e(\varphi_1 + \varphi_2))$$

$$C_{11} = V_1$$

$$C_{22} = V_2$$

V_1, V_2 are viscous friction coefficients of the arm joints, 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. Note that although equation (21) is non-linear, it also incorporates linear terms. This was the motivation for using the type-1 neural network with both linear and non-linear neurons to represent the robot dynamics.

The structural and training parameters for the various neural networks are given in Table I.

5.1. Simulation I

In this simulation, type-1 networks were adopted as the neural controller 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, the payload was suddenly changed from 10 kg to 30 kg. The result obtained is shown in Figure 8. It can be seen that a 300 percent load increase can be accommodated by this controller which is a very good measure of its robustness. The resulting outputs of the control system are shown in Figures 9(a) to 9(d)

5.2. Simulation II

In this simulation, the use of type-2 neural networks for the modelling and control of the SCARA robot was demonstrated. The result obtained after the dynamics change is plotted in Figure 10.

5.3. Simulation III

This simulation cmployed a conventional PID controller for comparison with the neural control systems. The parameters of the PID controller were: $\mathbf{K}_p = \text{diag}[150\ 150]$, $\mathbf{K}_1 = \text{diag}[10\ 10]$ and $\mathbf{K}_D = \text{diag}[1\ 1]$. Its performance after the dynamics change is illustrated in Figure 11.

The root-mean-squared errors (RMSE) for the different controllers are given in Table II

Table I. Structural and training parameters

NN	η	μ	α	β	N	n
Type-1	0.0001	0.01	0.8	0.8	50 000	6+6
Type-2	0.0001	0.01	–	–	50 000	12

https://doi.org/10.1017/S0263574799002179 Published online by Cambridge University Press

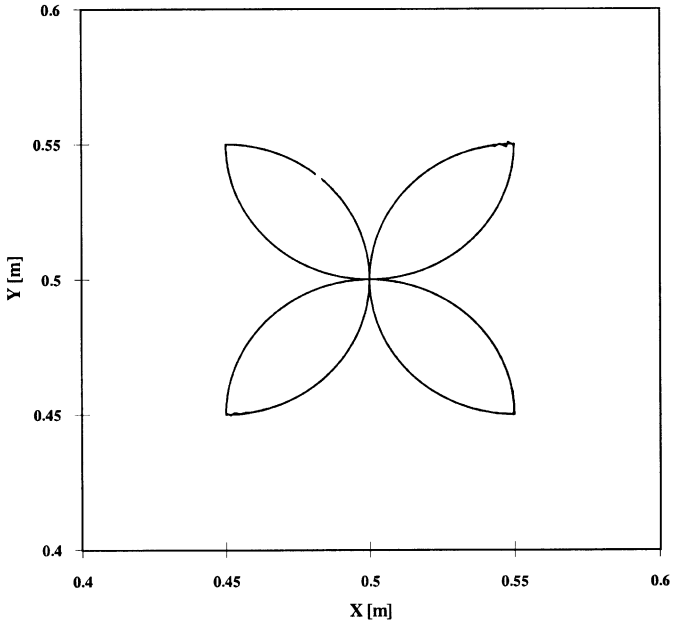

Fig. 8. Desired and actual trajectories of the end-effector (after dynamics change, using Type-1 neural networks)

6. CONCLUSION

This paper presents the use of recurrent neural networks to implement internal model control (IMC) of a SCARA robot arm. Both the controller and the model were realised with neural networks.

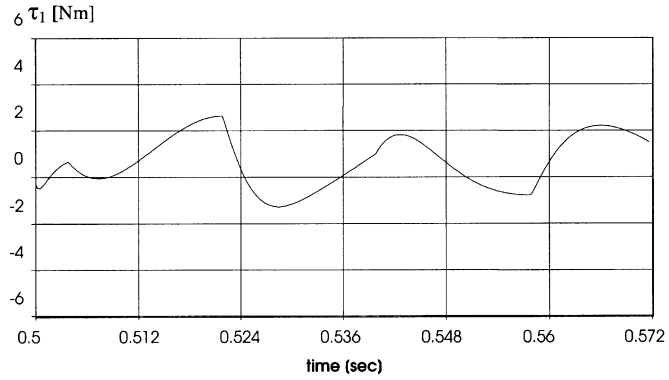

Fig. 9(a). Output τ_1 of the control system (after dynamics change, using Type-1 neural networks, first trial)

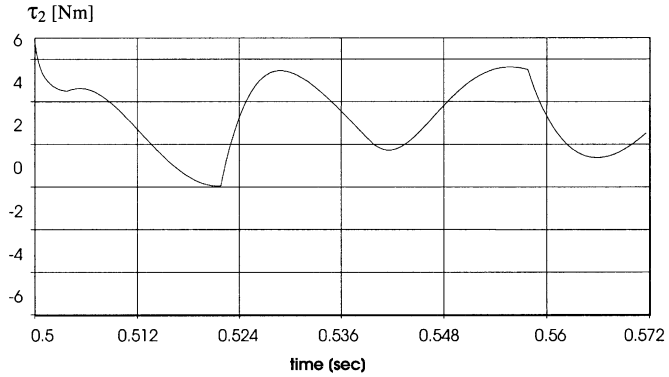

Fig. 9(b). Output τ_2 of the control system (after dynamics change, using Type-1 neural networks, first trial)

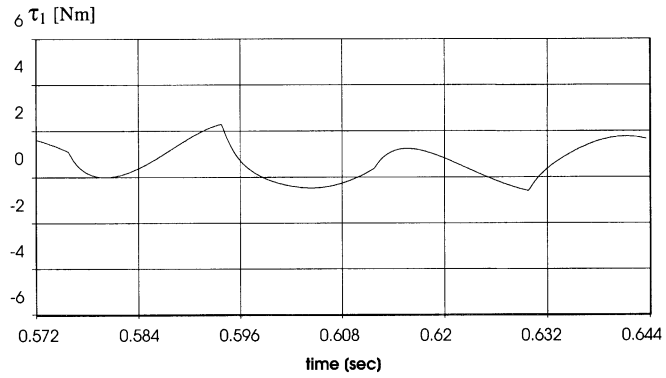

Fig. 9(c). Output τ_1 of the control system (after dynamics change, using Type-1 neural networks, second trial)

From the results obtained, it can be seen that the IMC system employing type-1 neural networks produced the best performance, while the PID system yielded the poorest control. A reason for the strong performance of the type-

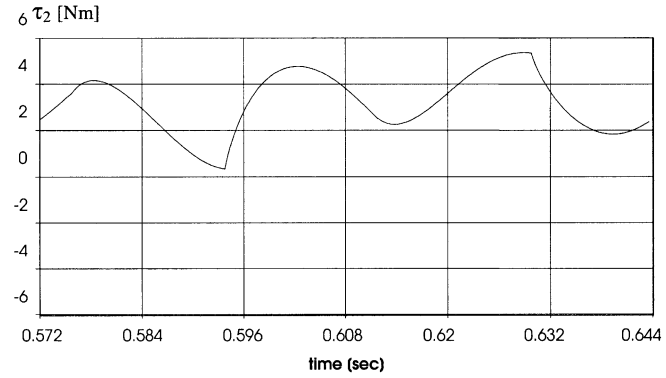

Fig. 9(d). Output τ_2 of the control system (after dynamics change, using Type-1 neural networks, second trial)

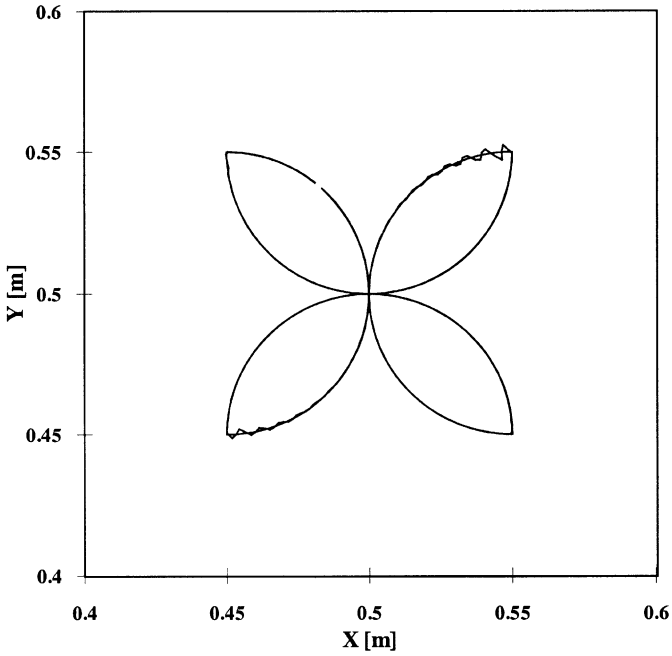

Fig. 10. Desired and actual trajectories of end-effector (after dynamics change, using Type-2 neural networks)

https://doi.org/10.1017/S0263574799002179 Published online by Cambridge University Press

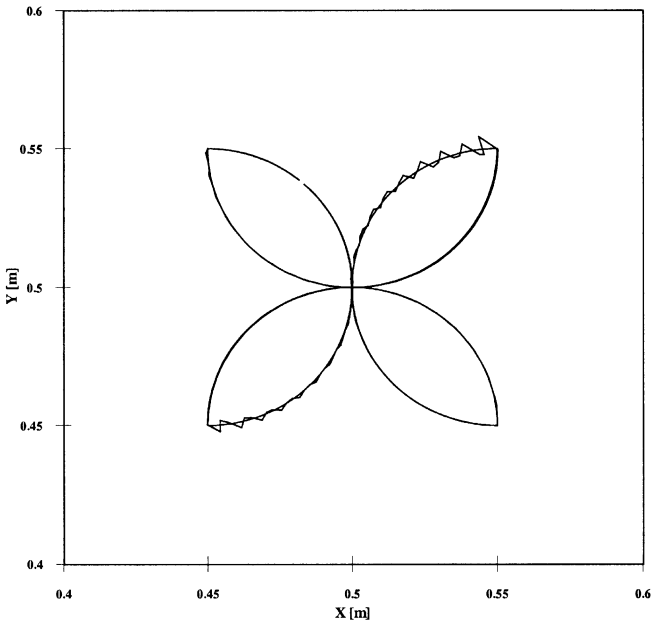

Fig. 11. Desired and actual trajectories of end-effector (after dynamics change, using PID controller)

1-network-based system was the inclusion of both linear and non-linear neurons in the neural 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.

In contrast, in the type-2 neural network as proposed by Ku et al.,[10] non-linear neurons had to learn both the linear and non-linear parts, which gave less accurate results.

Table II. Performance of different controllers

Controller	**RMSE** (Before dynamics change)	**RMSE** (After dynamics change)
Type-1	0.0019361	0.005141
Type-2	0.002747	0.009282
PID	0.005929	0.04882

ACKNOWLEDGEMENTS

The authors wish to thank the European Commission for funding this work under the ERDF Programme, and Erciyes University for supporting Ş. Yildirim's doctoral studies.

References

1. S. Armito and F. Miyazaki, "Stability and robustness of PID feedback control for robot manipulators of sensory capability," *First Int. Symposium on Robotics Research*, Gouvieux, France (1984) pp. 783–799.
2. K. Kreutz, "On manipulator control by exact linearization," *IEEE Trans. Automation Control* **34**, No. 7, 763–767 (1989).
3. C.E. Garcia and M. Morari, "Internal model control-1; a unifying review and some new results," *Ind. Eng. Chem. Process Des. Dev.* **21**, No. 2, 308–323 (1982).
4. K. Warwick, G.W. Irwin and K.J. Hunt, *Neural Networks for Control and Systems* (IEE, London, 1991).
5. W.S. Lee, B.D.O. Anderson, R.L. Kosut and I.M.Y. Mareels, "A new approach to adaptive robust control," *Int. J. Adaptive Control and Signal Processing* **7**, No. 3, 183–211(1993).
6. C.G. Economou, M. Morari and B.O. Palson, "Internal model control, extension to non-linear systems," *Industrial and Engineering Chemistry, Process Design Development* **25**, No. 2, 403–411(1986).
7. N. Bhat and T.J. McAvoy, "Use of neural nets for dynamical modelling and control of chemical process systems," *Computers and Chemical Engineering* **14**, Nos. 4–5, 573–583 (1990).
8. K.J. Hunt and D. Sbarbaro, "Neural networks for non-linear internal model control," *IEE Proceedings, Part-D: Control Theory and Applications* **138**, 431–438 (1991)
9. D.T. Pham and S. Yildirim, "Control of trajectory of a planar robot using recurrent hybrid networks," *Int J. of Machine Tools and Manufacture* **39**, No. 3, pp. 415–429 (1999)
10. C.C. Ku and K.Y. Lee, "Diagonal recurrent neural networks for dynamic systems control," *IEEE Trans. on Neural Networks* **6**, 144–156 (1995)
11. M.W. Spong and M. Vidyasagar, *Robot Dynamics and Control* (Wiley, New York, 1989).
12. D.T. Pham and X. Liu, *Neural Networks for Identification, Prediction and Control*, Fourth Printing (Springer-Verlag, London, 1999).

https://doi.org/10.1017/S0263574799002179 Published online by Cambridge University Press