15th Triennial World Congress, Barcelona, Spain

COMPARISON OF FOUR METHODS OF ROBOT TRAJECTORY CONTROL

D. T. PHAM[*] and Şahin YILDIRIM[+]

**Manufacturing Engineering Centre, Cardiff University*
Cardiff CF24 3TE, UK
+Engineering Faculty, Erciyes University
Kayseri 38039, TURKEY

Abstract: This paper describes four methods for robot trajectory control. These methods are a standard PID controller, a Computed Torque Method (CTM), a neural network based inverse controller and a neural network based Internal Model Controller (IMC). The IMC is investigated as an alternative to the basic inverse control scheme that is difficult to implement. The results presented show the superior ability of the proposed neural network based IMC scheme at adapting to changes in the dynamic parameters of the robot. *Copyright © 2002 IFAC*

Keywords: control, neural networks.

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 achieve satisfactory trajectory tracking performance (Arimoto and Miyazaki, 1984; Kreutz, 1989). The chief drawback of these methods, however, arises from the fact that they rely on exact cancellation of non-linear terms in order to obtain linear input-output behaviour.

To improve the performance of robots in uncertain environments, considerable research has been focused on developing advanced controllers in recent years and two main design techniques have emerged, namely robust control and adaptive control.

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. A suitable control strategy, which directly incorporates the plant model, is provided by the IMC method (Garcia and Morari, 1982). Recently, it has been shown that the IMC method is a simple and effective technique for designing the underlying control law in a new approach to adaptive robust control when the plant is stable (Lee *et al*., 1993). The applicability of IMC to the control of non-linear systems was demonstrated by Economou *et al.* (1986). The inverse of the non-linear operator 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 operator inverses. In the present work, NNs are utilised for the construction of plant models and their inverses and it is intended for them to be used directly within the IMC method.

The idea of employing NNs in the non-linear IMC method has been considered by Bhat and McAvoy (1990). A technique, using NNs directly, was proposed for the adaptive control of non-linear systems by Hunt and Sbarbaro (1991). The control structure adopted was the IMC method. This structure was used to incorporate network modelling of the plant and its inverse directly within the control strategy.

The realisation of an IMC using NNs is straightforward: the system's dynamics and the controller are simply implemented by means of NN

models (Pham and Yildirim, 1999). One of the main problems in robot control is to determine the sequence of joint inputs required to cause the end-effector to execute a given motion. The joint inputs are normally joint forces and torques. The required motion is typically specified either as a series of end-effector positions and orientations or as a continuous path.

Usually, to design a controller for a complex plant such as a robot, it is necessary to have detailed knowledge about its structure. This applies to conventional general-purpose Proportional-Integral-Derivative (PID) controllers and specialist model-based controllers such as the computed-torque controller proposed for some robot systems (Armito and Miyazaki, 1984). However, recently, neural-network-based control approaches have been developed that do not require much prior knowledge about the plant to be controlled. This is because neural networks, which are computational models of the brain, are able to learn their control task automatically. For example, Kawato *et al*. (1988) and Psaltis *et al*. (1987) have presented control architectures using neural networks that are efficient at learning non-linear motion control tasks from input and output data obtained from the plant. Pham and Yildirim (1999) have described the training of neural networks to model the forward and inverse dynamics of plants and have applied the trained neural networks to the control of a planar robot. The proposed control scheme is able to adapt to changes in the operating conditions and dynamics parameters of the robot. Ozaki *et al*. (1991) have presented a non-linear compensator for robot control using neural networks that incorporate the concepts of the computed-torque method. The authors have also explained the notion of model learning and demonstrated its usefulness in simulation. Another example of work in the neural control of robots is that of Khemaissia and Morris (1993) who have described a new system identification and adaptive control scheme for robotic devices based on the non-linear functional properties of neural networks.

In most neural control schemes developed so far, the neural network is employed to obtain a model of the dynamics or inverse dynamics of the plant. The neural model is then utilised to replace a mathematical model, supplement a conventional controller or act as the controller itself. A popular neural network structure is the feedforward multi-layered type. However, this type of network is more naturally suited to static mapping than dynamic modelling and thus is not ideal for control applications. To enable a feedforward network to have a dynamic memory as required in such applications, it is usually fitted with tapped delay lines, which increases the number of inputs and makes training more difficult (Pham and Liu, 1997).

This paper compares four methods of trajectory control that were applied to a simulated SCARA arm robot (Pham and Yildirim, 1999).

In practice, some parameters 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 a robot dynamic equation is non-linear, it also incorporates linear terms. This motivates the idea of using a hybrid neural network, as discussed in the Appendix, with both linear and non-linear neurons to represent the robot dynamics.

2. CONTROL METHODS

For the SCARA arm robot (Pham and Yildirim, 1999), 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 φ_1 and φ_2, respectively. The dynamics equations of the robot arm can be written as follows (Pham and Yildirim, 1999):

$$\boldsymbol{\tau} = \mathbf{M}(\boldsymbol{\varphi})\ddot{\boldsymbol{\varphi}} + \mathbf{C}(\boldsymbol{\varphi}, \dot{\boldsymbol{\varphi}}) + \mathbf{F}(\boldsymbol{\varphi}, \dot{\boldsymbol{\varphi}}) \qquad (1)$$

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

2.1 PID control.

PID control is perhaps the most widely used control method. It can provide a fast response, good system stability and small steady-state errors in a linear system with known parameters.

As its name implies, a PID controller consists of three parts: Proportional, Integral and Derivative Control. Assuming that each joint is completely decoupled and controlled independently from other joints, the control input $\boldsymbol{\tau}(t)$ is given by:

$$\boldsymbol{\tau}(t) = \mathbf{K}_P\,\mathbf{e}(t) + \mathbf{K}_I \int \mathbf{e}(t)dt + \mathbf{K}_D \frac{d\mathbf{e}(t)}{dt} \qquad (2)$$

In equation (2), $\mathbf{e}(t)$ is the control error

$$\mathbf{e}(t) = \boldsymbol{\varphi}_d(t) - \boldsymbol{\varphi}(t) \qquad (3)$$

where $\boldsymbol{\varphi}_d(t)$ is the desired robot joint position vector $[\varphi_{d1}\ \varphi_{d2}]^T$ and $\boldsymbol{\varphi}(t)$ is the current measured joint position vector $[\varphi_1\ \varphi_2]^T$. $\mathbf{K}_P$ is called the proportional gain, $\mathbf{K}_I$ the integral gain and $\mathbf{K}_D$ the derivative gain, all of which are $n\times n$ diagonal matrices where n=2 is the number of joints. PID controllers for articulated robots face two main difficulties, non-linearity and cross-coupling [9].

2.2 Computed Torque control.

Due to the aforementioned problems of non-linearity and cross-coupling in robot control, much of linear control theory is not directly applicable. Computed-torque (CT) control schemes based on either the Lagrange-Euler formulation or the Newton-Euler formulation have been proposed (Figure 1).

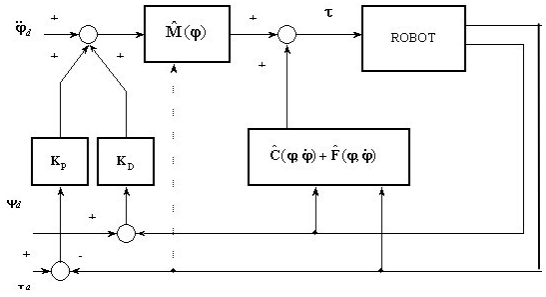

Fig. 1. Configuration of computed torque (CT) control scheme

For a general CT scheme using the Lagrange-Euler formulation, the control input is

$$\boldsymbol{\tau} = \hat{\mathbf{M}}(\boldsymbol{\varphi})\left[\ddot{\boldsymbol{\varphi}}_d + \mathbf{K}_D\dot{\mathbf{e}} + \mathbf{K}_P\mathbf{e}\right] + \hat{\mathbf{C}}(\boldsymbol{\varphi},\dot{\boldsymbol{\varphi}}) + \hat{\mathbf{F}}(\boldsymbol{\varphi},\dot{\boldsymbol{\varphi}}) \quad (4)$$

where $\hat{\mathbf{M}}(\boldsymbol{\varphi})$ is an $n\times n$ inertia matrix, $\hat{\mathbf{C}}(\boldsymbol{\varphi},\dot{\boldsymbol{\varphi}})$ is an $n\times 1$ vector of centrifugal and Coriolis terms and $\hat{\mathbf{F}}(\boldsymbol{\varphi},\dot{\boldsymbol{\varphi}})$ is an $n\times 1$ vector of friction terms. $\hat{\mathbf{M}}$, $\hat{\mathbf{C}}$ and $\hat{\mathbf{F}}$ are models of **M**, **C** and **F** as given in (Pham and Yildirim, 1999).

If these models exactly match the actual system, the error is governed by

$$\ddot{\mathbf{e}} + \mathbf{K}_D\dot{\mathbf{e}} + \mathbf{K}_P\mathbf{e} = 0 \quad (5)$$

Then, by choosing $\mathbf{K}_P$ and $\mathbf{K}_D$ such that Equation (5) has negative real roots, **e** can be made to approach zero asymptotically. When the gain matrices are chosen to be diagonal such that $K_{Dkk}^2 = 4K_{Pkk}$ for all k, critical damping is achieved for the error dynamics. However, due to modelling inaccuracies, mismatched terms appear on the right-hand side of (5).

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 situations, 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 method significantly degrades due to the inaccuracy of the dynamic model for computing the control torques.

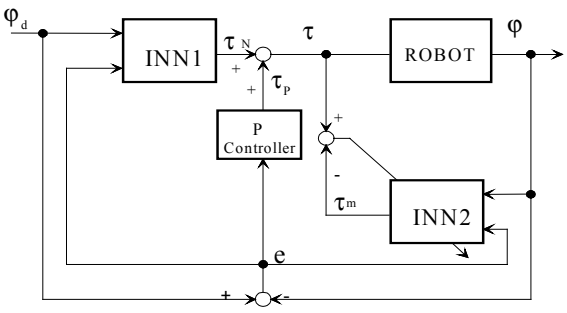

Fig. 2. Proposed inverse control system

2.3 Inverse Model Control

As shown in Figure 2, the proposed control system includes a feedforward controller (INN1) which is a copy of the neural network (INN2) used to learn the inverse dynamics of the plant (that is INN2 has to learn to produce the torque required to generate given joint rotations. INN1 and INN2 are recurrent hybrid neural networks as detailed below. INN2 is trained on-line during control to give the system the ability to adapt to change. In addition, there is a simple conventional feedback controller to enable the system to be controlled before INN1 and INN2 are ready.

From Figure 2, the control input to the robot is given by

$$\boldsymbol{\tau} = \boldsymbol{\tau}_N + \boldsymbol{\tau}_P \quad (6)$$

where $\boldsymbol{\tau}_N$ is the output of INN1 and $\boldsymbol{\tau}_P$ the output of the feedback controller. The latter is taken to be a simple proportional controller with gain $\mathbf{K}_f$ and thus

$$\boldsymbol{\tau}_P = \mathbf{K}_f\,\mathbf{e} \quad (7)$$

In addition to the control loop formed by the neural and P controllers, the system also includes an inner-loop PID controller that acts directly on the joint motors to provide stable speed control.

2.4 Internal Model Control (IMC)

A two-step procedure for using a neural network directly within the IMC structure is proposed. The first step involves training a network to represent the robot's forward dynamics. This network is used as a model of the robot in the IMC structure (Figure 3). The minor-loop controller, which is not shown, is also employed to simplify the design of the overall controller and obtain more stable control (Pham and Yildirim, 2000). 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:

$$\Delta W_{kh}(t) = -\eta \frac{\partial E(t)}{\partial W_{kh}(t)}$$

$$= -\eta \sum_i \frac{\partial E(t)}{\partial \varphi_i(t)} \frac{\partial \varphi_i(t)}{\partial \tau_k(t)} \frac{\partial \tau_k(t)}{\partial W_{kh}(t)} \quad (8)$$

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 i^{th} output of neural model, can be employed once the neural model of the robot is well trained. 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 [11].

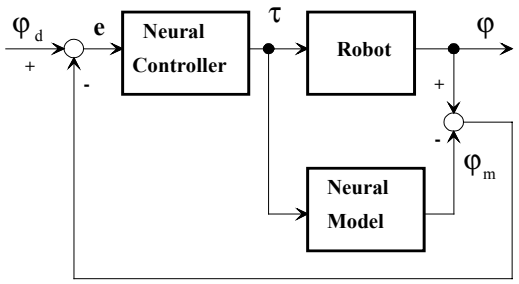

Fig.3 IMC system

3. SIMULATION RESULTS

The robot chosen to be controlled in this study was a planar 2-degree-of freedom SCARA arm robot (Pham and Yildirim, 1999). The results of the simulated implementation of the four methods are given below.

Figures 4 and 5, respectively, show the trajectories produced using a PID controller and a CT controller tuned for a system with a payload of 10 kg but handling an actual load of 30 kg. The P and D gains of the CT controller were $\mathbf{K}_P$=diag[850 850] and $\mathbf{K}_D$=diag[150 150]. The gains of the PID controller were $\mathbf{K}_P$=diag[150 150], $\mathbf{K}_I$=diag[10 10] and $\mathbf{K}_D$=diag[1 1].

To test the ability of the neural network based inverse model control system to adapt to different robot payloads, the value of m_2 was increased to 30 kg following training on the 10 kg payload. The performance of the system was computed on the second attempt at following the specified trajectory. The actual trajectories of the robot end effector superimposed on the specified trajectory are plotted in Figure 6.

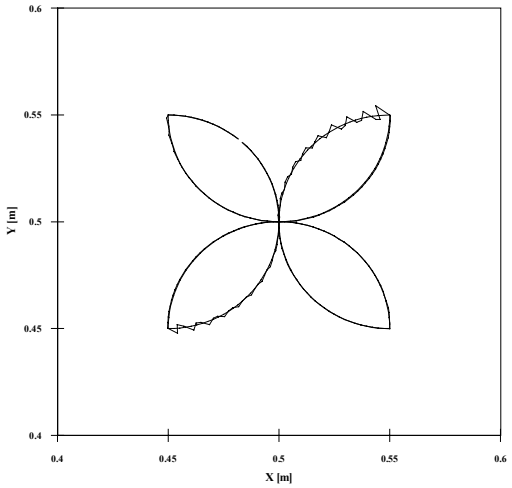

Fig. 4. Desired and actual trajectories of the endeffector after dynamics change(using PID controller)

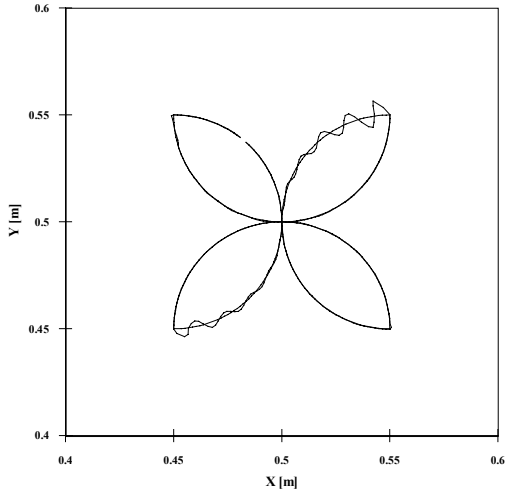

Fig. 5. Desired and actual trajectories of the endeffector after dynamics change(using CT controller)

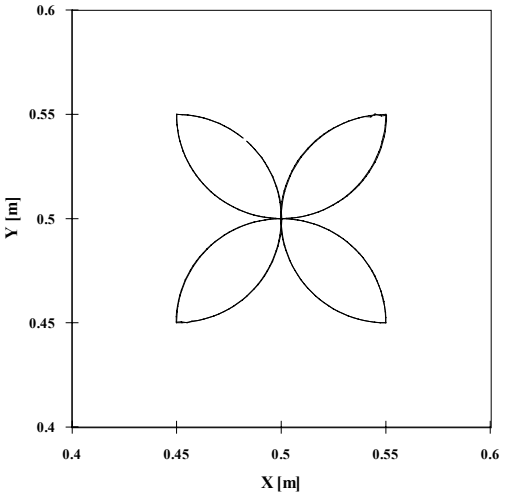

Fig. 6. Desired and actual trajectories of the end effector after dynamics change(using inverse control system)

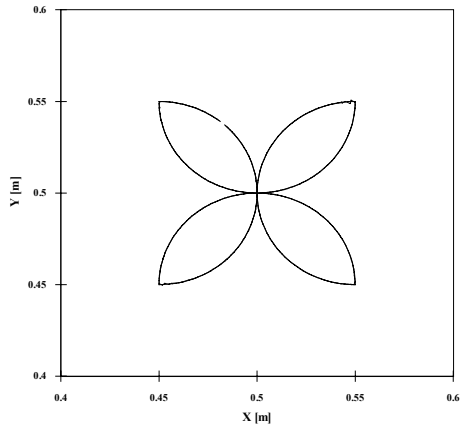

Fig. 7. Desired and actual trajectories of the end effector after dynamics change(using IMC)

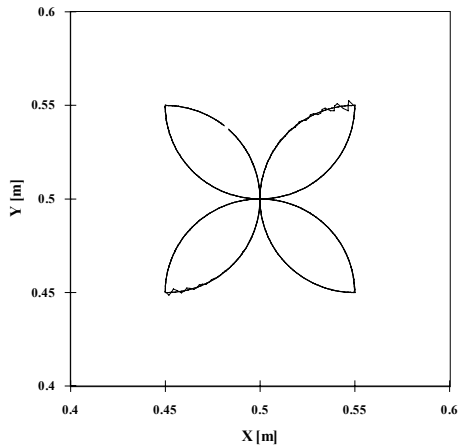

Figure 8. Desired and actual trajectories of end effector (using FNN, IMC)

In the neural network based IMC, the recurrent hybrid network (RHN) structure adopted was 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, the payload was suddenly changed from 10 kg to 30 kg. The result obtained is shown in Figure 7. Also, FNN was employed as a neural controller and neural model. Figure 8 shows the results of the desired and actual trajectories of the end effector using FNN.

This is confirmed by inspecting Table 1 which presents the RMS error values for the different types of controller.

Controller	RMS error (before dynamic change)	RMS error (after dynamic change)
PID	0.00592	0.00488
CTM	0.00493	0.0510
Inverse Model Control	0.00211	0.00617
IMC (RHN)	0.0019361	0.005141
IMC (FNN)	0.002747	0.04882

Table 1. RMS errors of different controllers

4. DISCUSSION

From the results obtained, it can be seen that the IMC system employing a RHN controller produced the best performance while the inverse control system yielded poor control. A reason for the strong performance of the RHN-based IMC system was the inclusion of both linear and non-linear neurons in a RHN. 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.

5. CONCLUSION

Four types of control methods have been presented in this paper. It is well known that inverse model control is difficult to implement due to problems with obtaining an accurate model of the plant to be controlled. The use of a neural network to learn the plant inverse model avoids the need for *a priori* knowledge or assumptions about the plant. The proposed recurrent hybrid neural network has two advantages: the recurrent connections 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.

6. APPENDIX: PROPOSED RECURRENT HYBRID NETWORKS

Fig. A.1 shows the structure of the proposed RHN. In addition to the input and hidden feedforward connections, there are also feedback connections from the output layer to the hidden layer and self feedback connections in the hidden layer (Pham and Yildirim, 2000).

At a given discrete time t, let $\mathbf{u}(t)$ be the input to a RHN, $\mathbf{y}(t)$, 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:

$$\mathbf{x}_1(t+1)=\mathbf{W}_{I1}\,\mathbf{u}(t+1)+\beta\,\mathbf{x}_1(t)+\alpha\mathbf{J}_1\mathbf{y}(t) \qquad (A.1)$$

$$\mathbf{x}_2(t+1)=\mathbf{F}\{\mathbf{W}_{I2}\,\mathbf{u}(t+1)+\beta\,\mathbf{x}_2(t)+\alpha\mathbf{J}_2\mathbf{y}(t)\} \qquad (A.2)$$

$$\mathbf{y}(t+1)=\mathbf{W}_{H1}\,\mathbf{x}_1(t+1)+\mathbf{W}_{H2}\,\mathbf{x}_2(t+1) \qquad (A.3)$$

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 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:

$$\mathbf{y}(t+1)= \mathbf{W}_{H1}\, \mathbf{x}(t+1) \qquad (A.4)$$

$$\mathbf{x}(t+1)= \mathbf{W}_{I1}\, \mathbf{u}(t+1)+\beta\mathbf{x}(t)+\alpha\mathbf{J}_1\, \mathbf{y}(t) \qquad (A.5)$$

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

$$\mathbf{x}(t+1)= (\beta\, \mathbf{I}+\alpha\, \mathbf{J}_1\, \mathbf{W}_{H1})\mathbf{x}(t)+\mathbf{W}_{I1}\, \mathbf{u}(t+1) \qquad (A.6)$$

where $\mathbf{I}$ is a $n_{H1}\times n_{H1}$ identity matrix. Equation(A.6) is of the form

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

where $\mathbf{A}= \beta\mathbf{I}+\alpha\mathbf{J}\mathbf{W}_{H1}$ and $\mathbf{B}= \mathbf{W}_{I1}$. Equation (A.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 dynamic mapping and thus model non-linear dynamic systems. The existence in the RHN 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. The inverse dynamics equation of the robot includes both linear terms and non-linear terms.

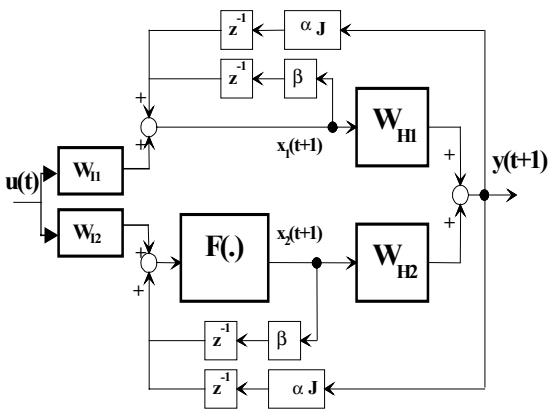

Figure A.1. Block diagram of recurrent hybrid network

In this work, the values of the weights of the recurrent connections, α and β, are fixed. This means only the weights of the feedforward connections, $\mathbf{W}_{Ii}$ and $\mathbf{W}_{Hj}$, need to be adjusted and thus the standard backpropagation algorithm can be employed to train the neural internal model.

ACKNOWLEDGEMENT

The authors wish to thank the European Commission for funding this work under the ERDF Programme, and the Royal Society and TUBITAK for supporting Dr. Ş. Yildirim's visit to Cardiff University.

REFERENCES

Armito, S., and Miyazaki, F. (1984), Stability and robustness of PID feedback control for robot manipulators of sensory capability. *First Int. Symposium on Robotics Research*, Gouvieux, France, pp. 783-799.

Bhat, N., and Mcavoy, T. J. (1990), Use of neural nets for dynamical modelling a control of chemical process systems. *Computers & Chemical Engineering*, **Vol. 14**, No. 4-5, pp. 573-583.

Craig, J.J, *Introduction to Robotics*, Second Edition (Addison-Wesley, Wokingham, UK, 1989)

Economou, C. G., Morari, M., and Palson, B. O. (1986), Internal model control, extension to non-linear systems. *Industrial & Engineering Chemistry, Process Design Development*, **Vol. 25**, No. 2, pp. 403-411.

Garcia, C. E., and Morari, M. (1982), Internal model control-1; a unifying review and some new results. *Ind. Eng. Chem. Process Des. Dev*, **Vol. 21**, No.2, pp. 308-323.

Hunt, K. J. and Sbarbaro, D. (1991), Neural networks for non-linear internal model control. *IEE Proceedings, Part-D: Control Theory and Applications*, **Vol. 138**, pp. 431-438.

Kawato, M., Uno, Y., Isabe, M. and Suzuki, R. (1988), Hierarchical Neural Network Model for Voluntary Movement with Application to Robotics, *IEEE Control Systems Magazine*, pp. 8-16.

Khemaissia, S. and Morris, A.S. (1993), "Neuro-Adaptive Control of Robot Manipulators", *Robotica*,**11**, pp. 456-473.

Kreutz, K. (1989), On manipulator control by exact linearization. *IEEE Trans. Automation Control*, **Vol. 34**, No. 7, pp. 763-767.

Lee, W. S., Anderson, B. D. O., Kosut, R. L., and Mareels, I. M. Y. (1993), A new approach to adaptive robust control, *Int. J. Adaptive Control and Signal Processing*, **7**, No. 3, pp. 183-211.

Ozaki, T., Suzuki, T. and Furuhashi, T. (1991), "Trajectory Control of Robotic Manipulators Using Neural Networks", *IEEE Trans. on Industrial Electronics*, **38**, No.3, pp. 195-203.

Pham, D.T. and Liu, X. (1997), *Neural Networks for Identification, Prediction and Control*, Third Printing, Springer-Verlag, London, U.K.