Optimal Neural Network Output Feedback Control for Robot Manipulators

Mario Jungbeck, Marconi K. Madrid

Abstract— **A robust neural network output feedback scheme is developed for the joint motion control of robot manipulators without direct measuring joint velocities. A neural network observer is used to estimate the joint velocities. The neural network weights in both the observer and the controller are tuned on-line, with no off-line learning phase required. No exact knowledge of the robot dynamics is required so that the neural network controller is model-free and so applicable to a class of nonlinear systems which have a similar structure of robot manipulators. Implementation results on a single-link robot are reported to show the performance of the proposed technique.**

Keywords—**Neural networks, Optimal control, Robot control.**

I. Introduction

THERE are some work related to applying optimal control techniques to the nonlinear robotic manipulator, various control schemes are developed in literature for motion control based on full state measurement [1][2]. These approaches often combine feedback linearization and optimal control techniques. However, in actual situations, the robot dynamics is rarely completely known, and it is difficult to express real robot dynamics in exact mathematical equations or to linearize the dynamics with respect to the operating point.

Kim *et al.*[2] designed a nonlinear optimal neural net control structure that integrates linear optimal control techniques and neural network learning methods. The linear optimal control has an inherent robustness over a certain range of model uncertainties and the feed-forward neural networks is used to adaptively estimate nonlinear uncertainties, yielding a controller that can tolerate a wider range of uncertainties.

However, for full state variable feedback control, measurements of joint position and joint velocity are required. The joint position measurements can be obtained through encoders, which can give very accurate measurements of joint displacements. Conversely, joint velocity measurements can be obtained by mean of tachometers, which are often contaminated by noise. This circumstance may reduce the dynamic performance of the manipulator, since, in practice, the values of the controller gain matrices are limited by the presence of noise in the velocity measurements. Therefore it is important to achieve satisfactory control performance considering only joint position measurements.

To estimate joint velocities, a neural network observer is presented in [3]. Only the inertia matrix is assumed known, the Coriolis terms, gravity terms, and friction terms are assumed unknown.

In this paper, neural networks are used for closed-loop output feedback control without joint velocity measurements. First, a neural network observer is designed to estimate the unknown velocity. A feedforward neural network is inserted in the feedback path to capture the nonlinear characteristics of the observed system [3]. The estimated velocity is used as an input to a second neural network that operates as an optimal feedback controller [2]. The neural network weights here are tuned on-line, with no off-line learn-ing phase required. When compared with adaptive techniques, it is not require the knowledge of the exact robot dynamics, or linearity in the unknown system parameters.

II. Background

Let $\Re$ denote the real numbers, $\Re^n$ the real n-vectors, $\Re^{m\times n}$ the real matrices. The norm of a vector $x \in \Re^n$ is defined as $\|x\| = \sqrt{x_1^2 + \cdots + x_n^2}$ and the norm of a matrix $A \in \Re^{m\times n}$ as $\|A\| = \sqrt{\lambda_{max}[A^TA]}$, where $\lambda_{max}[\cdot]$ and $\lambda_{min}[\cdot]$ are the largest and smallest eigenvalues of a matrix. The absolute value is denoted as $|\cdot|$.

Given $A = [a_{ij}]$ and $B \in \Re^{m\times n}$, the Frobenius norm is defined by $\|A\|_F^2 = tr(A^TA) = \sum a_{ij}^2$ with $tr(\cdot)$ the trace operator. The associated inner product is $\langle A, B\rangle_F = tr(A^TB)$. The Frobenius norm is compatible with the two-norm so that $\|Ax\|_2 \leq \|A\|_F\|x\|_2$, with $A \in \Re^{m\times n}$ and $x \in \Re^n$.

A. Neural Networks for Approximation

Fig. 1 shows a functional link neural network (FLNN) which can be considered as one-layer feedforward neural network with input preprocessing element. The FLNN architecture appears to be identical to that of the conventional two-layer neural network, except for the critical differences that only weights in the output layer will be adjusted, [1][2][3].

A FLNN can be used to approximate a nonlinear mapping $z(x) : \chi^n \to \psi^m$, where $\chi^n \subset \Re^n$ is the application specific n-dimensional input space and $\psi^m \subset \Re^m$ is the application specific output space. If $\sigma(\cdot)$ is a continuous discriminant function, then finite sums of the form

$$y_j(x) = \sum_{i=1}^{N}\{w_{ji}\sigma_i(p_i + \theta_i) + \theta_j\} + \epsilon_j(x) \qquad j = 1, \cdots, m \tag{1}$$

M. Jungbeck and M. K. Madrid are with the Laboratório Sistemas Modulares Robóticos, Universidade Estadual de Campinas (UNICAMP), Campinas - SP, Brasil. E-mail: {mjbeck, madrid}@dsce.fee.unicamp.br .

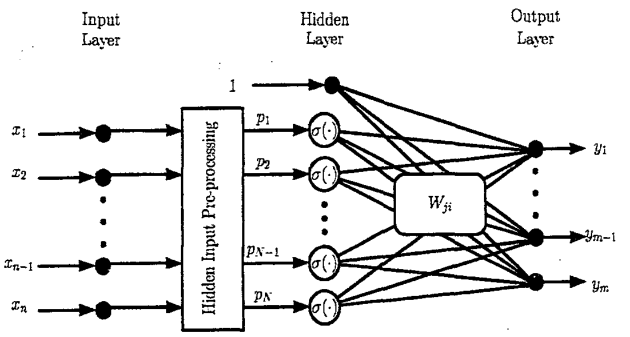

Fig. 1. Functional Link Neural Network Structure

are dense in the space of continuous functions defined on hypercubes or a compact set, $w_{ji} \in \Re$ output layer weight values, θ_i, $\theta_j \in \Re$ threshold values and $p_i \in \Re$ ith input to neuron. $\sigma(\cdot) : \Re \rightarrow \Re$ is called the sigmoidal activation function and N is the number of units (also called nodes or neurons) in the hidden layer. $\sigma_i(\cdot)$ is the output of the ith sigmoidal function. We use the notation $\sigma_i(\cdot)$ to denote the application of the nonlinear function $\sigma(\cdot)$ to each element of the vector $p \in \Re^N$. Note that the activation functions $\sigma(\cdot)$ for each neuron are not necessarily the same. Finally, $\varepsilon_j(x)$ is named *neural network functional reconstruction error.*

In general, even setting best-possible weight values, the given nonlinear function is not exactly approximated and, consequently, functional reconstruction errors $\varepsilon(x) \in \Re^m$ are remaining. We assume that an upper limit ε_M of the functional reconstruction error is known.

$$\|\varepsilon(x)\| \leq \varepsilon_M \qquad \forall x \in \chi^n \tag{2}$$

For control purposes, it is assumed that the ideal approximating neural network weights exist for a specified value of ε_M.

Then the neural network equation can be expressed in a compact matrix form

$$y(x) = W^T \sigma(p) + \varepsilon(x) \tag{3}$$

with $W \in \Re^{(N+1)\times m}$, $\sigma(p) \in \Re^{N+1}$, $y(x) \in \Re^m$, and $\varepsilon(x) \in \Re^m$. The inputs $p \in \Re^N$ to neural network is augmented by input preprocessing method, [1]. Notice that the threshold values θ_i are incorporated into both weight matrices W and sigmoid vector $\sigma(p)$. Then, an estimate $\hat{y}(x)$ of $y(x)$ can be obtained

$$\hat{y}(x) = \hat{W}^T \sigma(p) \qquad \forall x \in \chi^n, \tag{4}$$

where $\hat{W}$ are estimatives of the ideal neural network weights, that can be provided by some on-line weight tuning algorithms. The FLNN in (1) will be considered in the remainder of this paper.

B. Robot Arm Dynamics

The dynamics of an n-link robot manipulator may be expressed in the Lagrange form, [4],

$$M(q)\ddot{q} + V_m(q,\dot{q})\dot{q} + F_v\dot{q} + f_c(\dot{q}) + g(q) + \tau_d(t) = \tau(t) \tag{5}$$

with $q(t) \in \Re^n$ joint variable, $M(q) \in \Re^{n\times n}$ inertia, $V_m(q,\dot{q}) \in \Re^{n\times n}$ Coriolis/centripetal forces, $g(q) \in \Re^n$ gravitational forces, $F_v \in \Re^{n\times n}$ diagonal matrix of viscous friction coefficients, $f_C(\dot{q}) \in \Re^n$ Coulomb friction coefficients, and $\tau_d(t) \in \Re^n$ external disturbances. The bounded values of the external disturbances are given by $\|\tau_d(t)\| \leq b_d$. The external control torques applied to each joints are $\tau(t) \in \Re^n$.

III. Neural Network Observer Design

In the development of the observer and the controller the subscripts "o" and "c" refer to the observer and the controller quantities, respectively. The link position and velocity estimation error are given by

$$\tilde{q} = q - \hat{q}, \qquad \dot{\tilde{q}} = \dot{q} - \dot{\hat{q}} \tag{6}$$

Choosing state variables $x_1 = q$, $x_2 = \dot{q}$ the robot dynamics (5) can be written as, [3]

$$\begin{aligned} \dot{x}_1 &= x_2 \\ \dot{x}_2 &= h_o(x) + M^{-1}(x_1)\tau(t) \end{aligned} \tag{7}$$

where $h_o(x)$ is a nonlinear function given by

$$\begin{aligned} h_o(x) = -M^{-1}(q)[V_m(q,\dot{q}) + \\ + g(q) + F_v\dot{q} + f_c(\dot{q}) + \tau_d(t)]. \end{aligned} \tag{8}$$

It will be assumed that the inertia matrix is known. It is usual to have uncertainties in the Coriolis terms, which are difficult to compute, and the friction terms, which may have a complicated form. According to the well-established approximation property of neural network, it can be represented by a neural network for some constant "ideal" weights and some sufficient number of N_o input basis functions,

$$h_o(x) = W_o^T \sigma_o(x) + \varepsilon_o(x) \tag{9}$$

with $\|\varepsilon_o(x)\| \leq \varepsilon_{o,M}$ and $\sigma_o(x) \in \Re^{N_o}$. The functional estimate of (9) in terms of $\hat{x}_1, \hat{x}_2$ will be given by

$$\hat{h}_o(\hat{x}) = \hat{W}_o^T \sigma_o(\hat{x}) \tag{10}$$

with the current values of the neural network weights as provided by the tuning algorithm. The proposed observer has the form

$$\begin{aligned} \dot{\hat{z}}_1 &= \hat{x}_2 + k_D\tilde{x}_1 \\ \dot{\hat{z}}_2 &= \hat{W}_o^T\sigma_o(\hat{x}) + M^{-1}(x_1)\tau(t) + K\tilde{x}_1 \end{aligned} \tag{11}$$

with $k_D > 0$ and $K = K^T > 0$. The estimates of the system state (7) are defined as, [5],

$$\begin{aligned} \hat{x}_1 &= \hat{z}_1 \\ \hat{x}_2 &= \hat{z}_2 + k_P\tilde{x}_1 \end{aligned} \tag{12}$$

with $k_P > 0$. Therefore, the dynamic equations of the observer can be rewritten in terms of $\hat{x}_1$, $\hat{x}_2$ as

$$\begin{aligned} \dot{\hat{x}}_1 &= \hat{x}_2 - k_D\tilde{x}_1 \\ \dot{\hat{x}}_2 &= \dot{\hat{z}}_2 + k_D\dot{\tilde{x}}_1 = \hat{W}_o^T\sigma_o(\hat{x}) \\ &\quad + M^{-1}(x_1)\tau(t) + K\tilde{x}_1 + k_P\dot{\tilde{x}}_1 \end{aligned} \tag{13}$$

with $\tilde{x}_1 = x_1 - \hat{x}_1$ and $\tilde{x}_2 = x_2 - \hat{x}_2$.

Note that the observer (13) is not feasible, since it requires the knowledge of the error $\dot{\tilde{x}}_1$ which is not available. Therefore, (13) can be used only for analysis purposes [5]. The proposed observer formulates a dynamic neural network system with a feed-forward neural network inserted in the feedback path.

The observer error dynamics is obtained by subtracting (13) from (7), [3],

$$\begin{aligned} \dot{\tilde{x}}_1 &= \tilde{x}_2 - k_D\tilde{x}_1 \\ \dot{\tilde{x}}_2 &= W_o^T\sigma_o(x) - \hat{W}_o^T\sigma_o(\hat{x}) \\ &\quad -K\tilde{x}_1 - k_P\dot{\tilde{x}}_1 + \varepsilon_o(x) \end{aligned} \tag{14}$$

Adding and subtracting $W_o^T\sigma_o(\hat{x})$ yields

$$\begin{aligned} \dot{\tilde{x}}_1 &= \tilde{x}_2 - k_D\tilde{x}_1 \\ \dot{\tilde{x}}_2 &= W_o^T\sigma_o(\hat{x}) - K\tilde{x}_1 - k_P\dot{\tilde{x}}_1 + \varepsilon_o(\tilde{x}) \end{aligned} \tag{15}$$

with the weight estimation error vector $\tilde{W} = W - \hat{W}$. Considering a disturbance $w_o(\tilde{x})$

$$w_o(\tilde{x}) = W_o^T[\sigma_o(x) - \sigma_o(\hat{x})]. \tag{16}$$

bounded by

$$\|w_o(\tilde{x})\| \leq \varsigma_{o,M}, \tag{17}$$

with $\varsigma_{o,M} > 0$.

Then the observer error dynamics comes

$$\begin{aligned} \dot{\tilde{x}}_1 &= \tilde{x}_2 - k_D\tilde{x}_1 \\ \dot{\tilde{x}}_2 &= -k_p\tilde{x}_2 - (K - k_Pk_dI)\tilde{x}_1 + \tilde{W}_o^T\sigma_o(\hat{x}) + \\ &\quad +\varepsilon_o(x) + w_o(\tilde{x}) \end{aligned} \tag{18}$$

where $I \in \Re^{n\times n}$ is an identity matrix. This error dynamics is useful for deriving a weight update rule for the neural network observer based on the known position estimation error.

To guarantee the stability of the neural network observer, a suitable tuning algorithm must be provided for the neural network weights [3]. Suppose the observer gains are chosen to satisfy the following sufficient conditions:

$$k_P > k_D^2/2 - N_o/2 \tag{19}$$

$$\lambda_{min}(K) > (k_P^2 + N_ok_D^2)/2k_D \tag{20}$$

where N_o is the number of neurons in the neural network. Let weight tuning be provided by

$$\dot{\hat{W}}_o = -k_DF_o\sigma_o(\hat{x})\tilde{x}_1^T - \kappa_oF_o\|\tilde{x}_1\|\hat{W}_o - \kappa_oF_o\hat{W}_o \tag{21}$$

with any constant matrix $F_o = F_o^T > 0$, $\kappa_o > 0$ a sufficiently large design parameter. Then the state estimation error $\tilde{x}_1$, $\tilde{x}_2$, and weight estimation error $\tilde{W}_o$ are Uniformly Ultimately Bounded, [3].

IV. Optimal Control Design

Given a desired trajectory $q_d(t) \in \Re^n$, the robot link tracking error vectors, $e(t)$ and $\dot{e}(t)$ are defined as

$$\begin{aligned} e(t) &= q_d(t) - q(t) \\ \dot{e}(t) &= \dot{q}_d(t) - \dot{q}(t) \end{aligned} \tag{22}$$

and the instantaneous performance measure is defined as

$$\hat{r}(t) = \dot{\hat{e}}(t) + \Lambda e(t) \tag{23}$$

where Λ is a constant gain matrix or critic gain (not necessarily symmetric).

The robot dynamics (5) can be written in terms of the modified filtering tracking error as

$$M(q)\dot{\hat{r}}(t) = -V_m(q,\dot{q})\hat{r}(t) - \tau(t) + h(x) \tag{24}$$

where the robot nonlinear function is

$$\begin{aligned} h(x) &= M(q)(\ddot{q}_d + \ddot{\tilde{q}} + \Lambda\dot{e}) + \\ &+ V_m(q,\dot{q})(\dot{q}_d + \dot{\tilde{q}} + \Lambda e) + \\ &+ g(q) + F_v\dot{q} + f_c(\dot{q}) + \tau_d(t). \end{aligned} \tag{25}$$

This key function $h(x)$ considers all the unknown dynamics of the robot arm.

Establishing a control input torque as

$$\tau(t) = h(x) - u(t), \tag{26}$$

with $u(x) \in \Re^n$ an auxiliary control input to be optimized. The closed-loop system becomes

$$M(q)\dot{\hat{r}}(t) = -V_m(q,\dot{q})\hat{r}(t) + u(t) \tag{27}$$

Defining the position error dynamics as

$$\dot{e}(t) = -\Lambda e(t) + \hat{r}(t) \tag{28}$$

the following augmented system is obtained:

$$\dot{\tilde{z}} = \begin{bmatrix} \dot{e} \\ \dot{\hat{r}} \end{bmatrix} = \begin{bmatrix} -\Lambda & I \\ 0 & -M^{-1}V_m \end{bmatrix}\begin{bmatrix} e \\ \hat{r} \end{bmatrix} + \begin{bmatrix} 0 \\ M^{-1} \end{bmatrix}u(t) \tag{29}$$

or with shorter notation

$$\dot{\tilde{z}} = A(q,\dot{q})\tilde{z} + B(q)u(t) \tag{30}$$

with $A(q,\dot{q}) \in \Re^{2n\times 2n}$, $B(q) \in \Re^{2n\times n}$, $\tilde{z}(t) \in \Re^{2n\times 1}$. A quadratic performance index $J(u)$ is defined as follows:

$$J(u) = \int_{t_o}^{\infty} L(\tilde{z},u)dt \tag{31}$$

with the Lagrangian

$$L(\tilde{z},u) = \frac{1}{2}\tilde{z}^TQ\tilde{z} + \frac{1}{2}u^TRu. \tag{32}$$

Then the control objective is to find the auxiliary control input $u(t)$ that minimizes (31) subjected to the differential constraints imposed by (29). The optimal control that reaches this objective will be denoted by $u^*(t)$. A necessary and sufficient condition for $u^*(t)$ to minimize (31) subject to (29) is that there exists a function $V = V(\tilde{z}, t)$ satisfying the H-J-B equation,[6].

The following function V composed by $\tilde{z}$, $M(q)$ and a positive symmetric matrix $K = K^T \in \Re^{n\times n}$ satisfies the H-J-B equation, [2],

$$V = \frac{1}{2}\tilde{z}^T P(q)\tilde{z} = \frac{1}{2}\tilde{z}^T \begin{bmatrix} K & 0 \\ 0 & M(q) \end{bmatrix} \tilde{z} \tag{33}$$

where Λ and K in (23) and (33) can be found the Riccati's differential equation

$$PA + A^T P^T + PBR^{-1}B^T P + \dot{P} + Q = 0. \tag{34}$$

The optimal control $u^*(t)$ that minimizes (31) subject to (29) is, [2],

$$u^*(t) = -R^{-1}B^T P(q)\tilde{z} = -R^{-1}\hat{r} \tag{35}$$

V. Controller Design

The block diagram in Fig. 2 shows the major components that embody the FLNN controller. The external control torques to the joints are composed by the optimal feedback control law plus the FLNN output components.

The nonlinear robot function can be represented by a FLNN

$$h_c(x) = W_c^T \sigma_c(p) + \varepsilon_c(x), \qquad \|\varepsilon_c(x)\| \le \varepsilon_{c,M} \tag{36}$$

where the input vector p to the hidden layer neurons is provided by the input preprocessing method, [1]. The neural network reconstruction error $\|\varepsilon_c(x)\|$ is bounded by the known constant $\varsigma_{c,M}$.

Then a functional estimate $\hat{h}_c(x)$ of $h_c(x)$ can be written as

$$\hat{y}_c = \hat{W}_c^T \sigma_c(p). \tag{37}$$

The external torques is given by

$$\tau(t) = \hat{W}_c^T \sigma_c(p) - u^*(t) - \eta(t), \tag{38}$$

where $\eta(t)$ is a robustifying vector. Then, (24) becomes

$$\begin{aligned} M(q)\dot{\hat{r}}(t) &= -V_m(q,\dot{q})\hat{r}(t) + \tilde{W}_c^T \sigma_c(p) + \\ &\quad + \varepsilon_c(x) + u^*(t) + \eta(t) + \tau_d(t), \end{aligned} \tag{39}$$

with the weight estimation error $\tilde{W} = W - \hat{W}$. The state space description of (39) can be given by

$$\dot{\tilde{z}} = A\tilde{z} + B[\tilde{W}_c^T \sigma_c(p) + \varepsilon_c(x) + u^*(t) + \eta(t) + \tau_d(t)] \tag{40}$$

with $\tilde{z}$, A and B given in (30). Inserting the optimal feedback control law (35) into (40), then

$$\begin{aligned} \dot{\tilde{z}} &= (A - BR^{-1}B^T P)\tilde{z} + \\ &\quad + B[\tilde{W}_c^T \sigma_c(p) + \varepsilon_c(x) + \eta(t) + \tau_d(t)] \end{aligned} \tag{41}$$

and imposing the control $u^*(t)$ provided by the optimal controller (35) with the robustifying term given by,

$$\eta(t) = k_z \frac{\hat{r}(t)}{\|\hat{r}(t)\|} \tag{42}$$

with $k_z \le b_d$ and $\hat{r}(t)$ defined as the instantaneous performance measure, [2]. Let the adaptive learning rule for neural network weights be given by,

$$\dot{\hat{W}}_c = F_c \sigma_c(p)(B^T P(q)\tilde{z})^T - \kappa_c \|\tilde{z}\| \hat{W}_c \tag{43}$$

with $F_c = F_c^T > 0$ and $\kappa_c > 0$. Then the errors $e(t)$, $\hat{r}(t)$, and $\tilde{W}_c(t)$ are uniformly ultimately bounded, [2].

VI. Implementation Results

The neural net observer and the neural net controller discussed in Section III and V were implemented on a single-link test robot, and some of the results obtained are presented here.

A. Description of the Implementation

A list of the main characteristics of the practical implementation is given below.

- The dynamics of the single-link as expressed in the Lagrange form (5) are the following

$$\begin{aligned} M(q) &= 0.060, \quad V_m(q,\dot{q}) = 0 \\ F_v &= 0.085, \quad f_c(\dot{q}) = 0 \\ g(q) &= 2.06\sin(q). \end{aligned} \tag{44}$$

- The parameters of mass and length are $m = 0.21Kg$ and $\ell = 0.285m$.
- Both the observer and the controller were discretized with a sampling period of $5ms$. In the discretization process, the differential equations were solved on line using trapezoidal integration.
- The routines that perform the control action in real time are written in C. The execution of these routines is fired periodically fixed by and hardware interruption.
- The weighting matrices are as follows:

$$Q = \begin{bmatrix} 3 & -1 \\ -1 & 4 \end{bmatrix}, \qquad R^{-1} = 4. \tag{45}$$

- The controller FLNN is composed of eight neurons in the hidden layer, with eight inputs

$$x_c = p_c = \begin{bmatrix} 1 & q^T & \dot{q}^T & e^T & \dot{e}^T & \hat{r}^T & q_d^T & \dot{q}_d^T \end{bmatrix} \tag{46}$$

- The observer FLNN is composed of seven neurons in the hidden layer, with seven inputs

$$x_o = p_o = \begin{bmatrix} 1 & \tilde{q}^T & \hat{q}^T & \dot{\hat{q}}^T & \|\tilde{q}\| & \|\hat{q}\| & \|\dot{\hat{q}}\| \end{bmatrix} \tag{47}$$

A block diagram describing the practical implementation of the proposed controller is shown in Fig. 3.

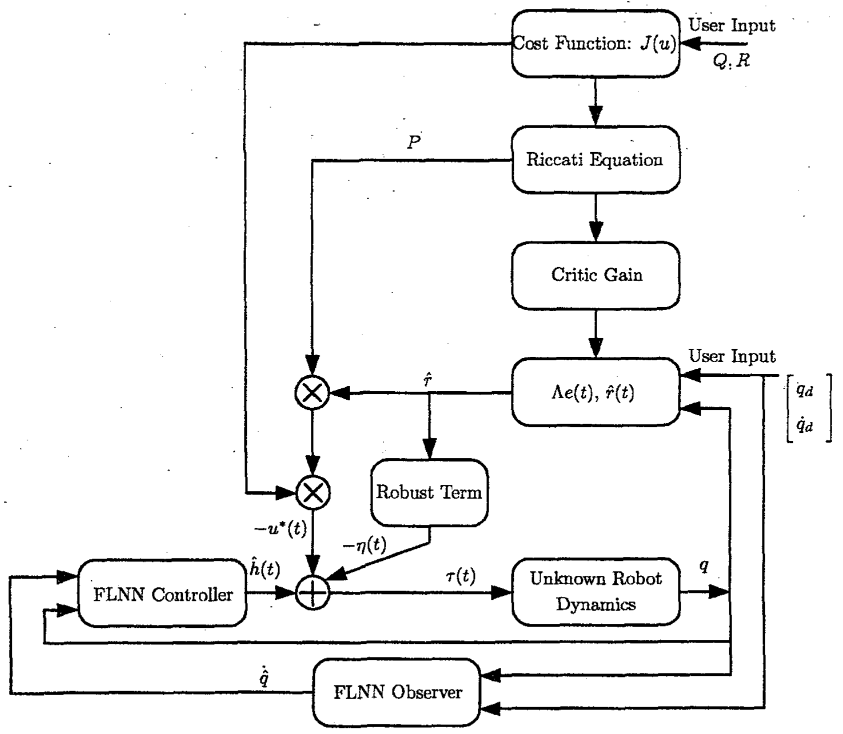

Fig. 2. Optimal Neural Controller

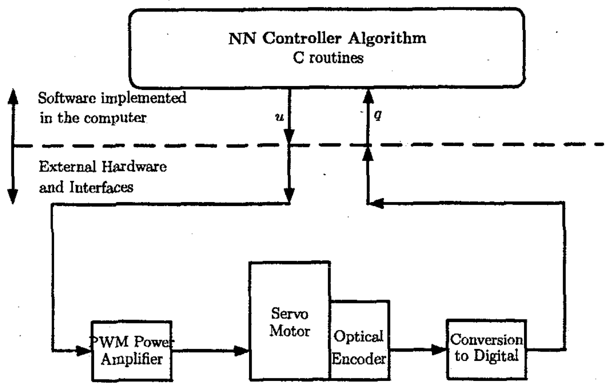

Fig. 3. Block Diagram of FLNN Controller Implementation

B. Experimental Results

The FLNN observer was implemented using $F_o = diag[2,\cdots,2]$, $\kappa_o = 0.0001$, $k_D = 20$, $k_P = 200$, $K = diag[400,\cdots,400]$, and the FLNN controller was implemented using $F_c = diag[2,\cdots,2]$, $\kappa_c = 0.0001$.

It is obvious that the control system performance may be quite different in low speed and high speed, therefore, it was made experiments based on two trajectories.

The reference signal was

$$q_d = \pi \sin(2\pi f t) \tag{48}$$

with frequencies $f = 0.05Hz$ in low speed, and $f = 0.25Hz$ high speed.

The performance of the NN tracking control based in the position error is presented in Fig. 4 and Fig. 5. The learning is really active all the time (on-line training), it was noticed that a change in the reference signal increases the error signal momentarily, requiring a readaptation of the FLNN weights. However, after some time , when the NN learned the new condition, it was able to get rid of the tracking error.

VII. Conclusions

We have presented an optimal neural network output feedback control scheme that includes a neural network observer for the motion control of robot manipulators. First, a neural network observer is designed to estimate the joint velocities. The method does not requires the exact robot dynamics knowledge, hence the same neural network observer-controller can be applied generically to any rigid link robotic system. Only the inversion boundedness of the inertia matrix is required. A key point in developing an intelligent control system is the reusability of the low-level controllers, and this is the case of the controller proposed in this paper. Compared with adaptive controllers, no linearity in the unknown system parameters is needed and no persistent excitation condition is required. And it's important to observe that any off-line "training or learning phase" is required.

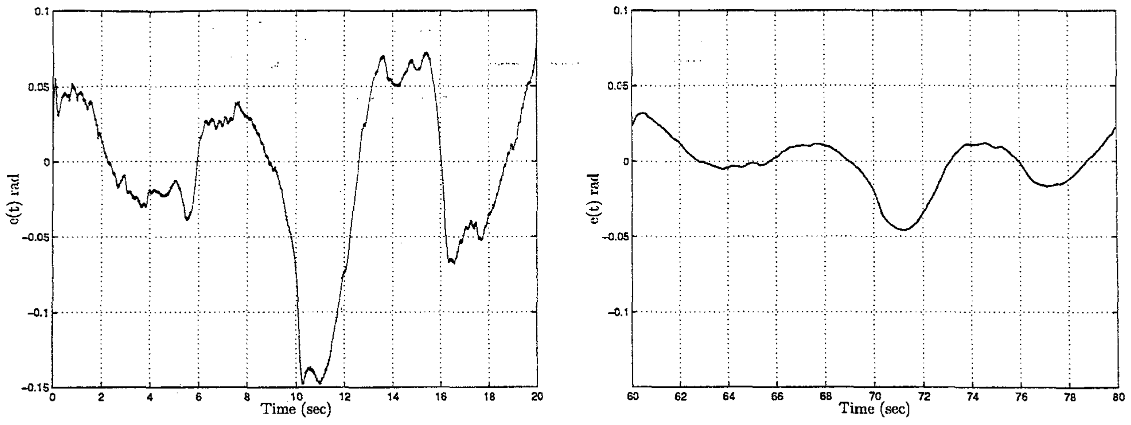

Fig. 4. Performance of the FLNN Controller for Low Speed

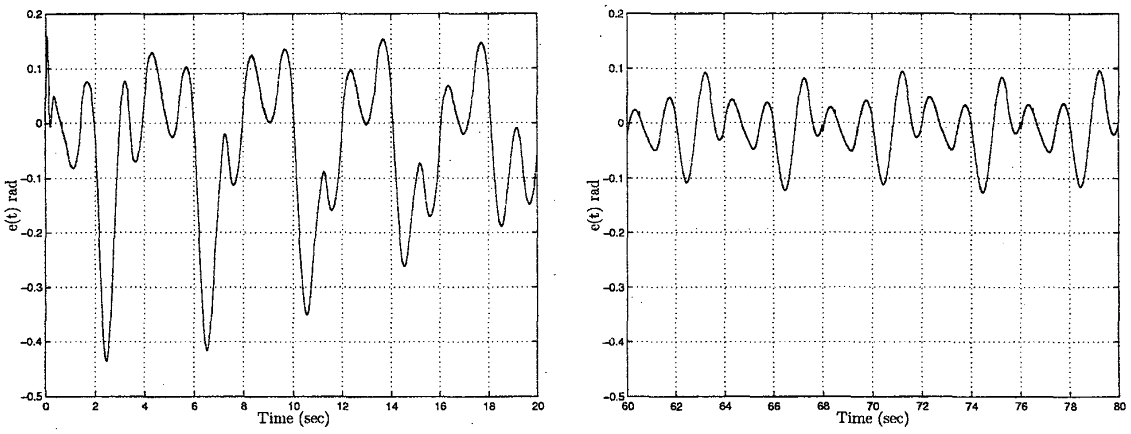

Fig. 5. Performance of the FLNN Controller for High Speed

References

[1] F.L. Lewis, K. Liu, and A. Yesildirek, "Neural net robot controler with guaranteed tracking performance," *IEEE Transactions on Neural Networks*, vol. 6, no. 3, pp. 703–715, 1995.

[2] Y.H. Kim, F.L. Lewis, and D. Dawson, "Intelligent optimal control of robotic manipulators using neural networks," *Automatica*, vol. 36, no. 9, pp. 1355–1364, 2000.

[3] Y.H. Kim and F.L. Lewis, "Neural network output feedback control of robot manipulators," *IEEE Transactions on Robotics and Automation*, vol. 15, no. 2, pp. 301–309, 1999.

[4] F.L. Lewis, S. Jagannathan, and A. Yesildirek, *Neural Network Control of Robot Manupipulators and Nonlinear Systems*, Taylor and Francis, 1999.

[5] S. Nicosia and P. Tomei, "Robot control by using only position measurements," *IEEE Transctions on Automatic Control*, vol. 35, pp. 703–715, 1990.

[6] F.L. Lewis and V.L. Syrmos, *Optimal Control*, New York: Wiley, 2nd edition, 1995.