Proceedings of the American Control Conference
Albuquerque, New Mexico June 1997
0-7803-3832-4/97/$10.00 © 1997 AACC

# IMPLEMENTATION OF A NEURAL NET TRACKING CONTROLLER FOR A SINGLE FLEXIBLE LINK: COMPARISON WITH PD AND PID CONTROLLERS

Luis B. Gutiérrez and Frank L. Lewis
Automation and Robotics Research Institute
The University of Texas at Arlington
7300 Jack Newell Blvd. South
Fort Worth, TX 76118-7115
lbgutie@janua.upb.edu.co
flewis@controls.uta.edu

## Abstract

The objective of this paper is to show the results of the practical implementation of a neural network tracking controller on a single flexible link and compare its performance to that of PD and PID standard controllers. The NN controller is composed of an outer PD tracking loop, a singular perturbation inner loop for stabilization of the fast flexible mode dynamics, and a neural network inner loop used to feedback linearize the slow pointing dynamics. No off-line training or learning is needed for the NN. It is shown that the tracking performance of the NN controller is far better than that of the PD or PID standard controllers. An extra friction term was added in the tests to demonstrate the ability of the NN to learn unmodeled nonlinear dynamics.

## 1 Introduction

In recent literature there have been many neural network controllers proposed for robot arms or other nonlinear systems [2],[13],[14],[16],[17],[18],[19]. The performance of these neural net controllers on actual systems has been open to question, despite the fact that several of these references provide stability proofs. In this paper we implement the neural net controller derived in [23] on an actual single-flexible-link robot arm which could emulate, for instance, a tank gun barrel in DoD applications. It is found that the NN controller far outperforms standard PD and PID controllers, even for the single-link arm which is basically linear except for nonlinear friction effects.

In [23], a tracking controller for a flexible-arm is designed using singular perturbation plus a NN feedback linearization inner loop. There, a modified output for tracking is defined that does correspond to practical tracking requirements. The structure of that controller includes an outer PD tracking loop, a singular perturbation inner loop for stabilization of the fast dynamics, and a neural network inner loop used to feedback linearize the rigid dynamics. Applying singular perturbation theory it is shown that after stabilizing the fast dynamics, the slow dynamics can be controlled using the same approach used in [9] and [10]. This approach avoids the requirement of the knowledge of friction, gravity and coriolis/centripetal terms, or any regression matrix. In contrast to other NN controllers in the literature, there is no off-line learning phase, the NN weights are easy to initialize without known 'stabilizing initial weights' (the weights are initialized at zero), and the controller guarantees boundedness of the tracking error and control signal.

In this paper, we present some practical implementation results for a single flexible link for the controller designed in [23]. Despite the fact that the dynamics of a single flexible link is linear, an extra friction term was added in the implementation to show the capability of the NN controller to compensate for nonlinearities in the model by learning. A comparison with the performance of standard PD and PID controllers is performed to show the superior tracking performance of the NN controller.

## 2 Dynamics of a Flexible Link Robot Arm

In [1],[3],[4],[5],[11] it is shown that the dynamics of any multi-link Flexible Link Robot can be represented by

$$M(q)\ddot{q} + D(q,\dot{q})\dot{q} + Kq + F(q,\dot{q}) + G(q) = B(q)u\,, \tag{1}$$

with

$$q = \begin{bmatrix} q_r \\ q_f \end{bmatrix}$$

where $q_r$ is the vector of rigid modes (generalized joint coordinates) and $q_f$ is the vector of flexible modes (the amplitudes of the flexible modes). $M(q)$ represents the inertia matrix, $D(q,\dot{q})$ is the Coriolis and centrifugal matrix, $K$ is the stiffness matrix, $F(q,\dot{q})$ is the friction matrix, $G(q)$ is the gravity matrix, $B(q)$ is an input matrix dependent on the boundary conditions selected in the assumed mode shapes method, and $u$ includes the control torques applied to each joint.

The model (1) follows the same properties of any standard rigid link robot [11]. That is $M(q)$ is positive definite and upper and lower bounded, $D(q,\dot{q})$ is bounded by $d_b(q)\|\dot{q}\|$, and $D(q,\dot{q})$ can be chosen such that $\dot{M}(q) - 2D(q,\dot{q})$ is skew-symmetric [11].

## 3 Neural Net Control of Flexible Link Robots

### 3.1 Singular Perturbation Approach

The singular perturbation approach basically consists in breaking the dynamics of the system in two parts, each of them in a separate time scale [6],[7],[8]. In this case the slow dynamics corresponds to the rigid modes $q_r$ and the fast dynamics corresponds to the flexible modes $q_f$.

The control objective is that $q_r(t)$ should track $q_d(t)$, a prescribed trajectory. For that purpose define the control

$$u = \bar{u} + u_F \tag{2}$$

where $\overline{u}$ is the slow component and $u_F$ is the fast component.

Applying singular perturbation [21], (1) is split as in [20],[21] to obtain the slow subsystem equation

$$\ddot{\overline{q}}_r = \overline{M}_{rr}^{-1}(-\overline{D}_{rr}\dot{\overline{q}}_r - \overline{F}_r - \overline{G}_r + \overline{u}) . \tag{3}$$

and the fast subsystem equation

$$\frac{d}{d\tau}\begin{bmatrix}\varsigma_1\\ \varsigma_2\end{bmatrix} = \begin{bmatrix}0 & I\\ -\overline{H}_{ff}\widetilde{K}_{ff} & 0\end{bmatrix}\begin{bmatrix}\varsigma_1\\ \varsigma_2\end{bmatrix} + \begin{bmatrix}0\\ \overline{B}_f^1\end{bmatrix}u_F , \tag{4}$$

or

$$\frac{d\varsigma}{d\tau} = A_F\varsigma + B_F u_F , \tag{5}$$

with $\varsigma = \begin{bmatrix}\varsigma_1^T & \varsigma_2^T\end{bmatrix}^T$, using a time scale $\tau = t/\varepsilon$ , and state variables defined by

$$\begin{aligned}\varsigma_1 &\equiv \xi - \overline{\xi}\\ \varsigma_2 &\equiv \varepsilon\dot{\xi}\end{aligned} \tag{6}$$

where

$$\overline{\xi} = \widetilde{K}_{ff}^{-1}\overline{H}_{ff}^{-1}(-\overline{D}_{fr}^1\dot{\overline{q}}_r - \overline{F}_f^1 - \overline{G}_f^1 + \overline{B}_f^1\overline{u}) . \tag{7}$$

According to Tikhonov's theorem [6],[7] the original system (1) can be described to order $\varepsilon$ using (3) and (5) with

$$\begin{aligned}q_r &= \overline{q}_r + O(\varepsilon)\\ q_f &= \varepsilon^2(\overline{\xi} + \varsigma_1) + O(\varepsilon)\end{aligned} \tag{8}$$

with $O(\varepsilon)$ denoting terms of order $\varepsilon$ .

Now define the tracking output

$$y = \begin{bmatrix}\overline{q}_r\\ \dot{\overline{q}}_r\end{bmatrix}, \tag{9}$$

which corresponds to the slow part of the rigid-mode variables (e.g. of the link-tip motion). Assume that $(A_F, B_F)$ is stabilizable, the fast system parameters have bounded uncertainties and perturbations (slow subsystem variables), and the slow system variables vary smoothly with time. The stabilizing assumption on $(A_F, B_F)$ is satisfied in practical systems and is far milder that the requirement for stable zero dynamics. Moreover, the definition (9) corresponds to practical tracking objectives in contrast to the "reflected" outputs defined in [12],[22]. Under these assumptions a stabilizing control $u_F(t)$ can easily be designed using linear techniques (e.g. $H_\infty$ design) so that

$$u_F = -\begin{bmatrix}K_{pF} & K_{dF}\end{bmatrix}\begin{bmatrix}\varsigma_1\\ \varsigma_2\end{bmatrix} = -\frac{K_{pF}}{\varepsilon^2}q_f - \frac{K_{dF}}{\varepsilon}\dot{q}_f + K_{pF}\overline{\xi} \tag{10}$$

stabilizes (5), with $\overline{\xi}$ given by (7).

## 3.2 Neural Net Control of the Rigid Dynamics

The slow dynamics given by (3) can be rewritten as

$$\overline{M}_{rr}\ddot{\overline{q}}_r + \overline{D}_{rr}\dot{\overline{q}}_r + \overline{F}_r + \overline{G}_r = \overline{u} \tag{11}$$

which is exactly the Lagrange form of an n-link rigid robot arm, satisfying the standard robot properties. For this part a neural network controller can be designed [9],[10]. Note that $\dot{\overline{M}}_{rr} - 2\overline{D}_{rr}$ is skew-symmetric.

Given a desired trajectory $q_d(t)$ for $\overline{q}_r$ the tracking error is

$$e = q_d - \overline{q}_r . \tag{12}$$

Define the filtered tracking error as

$$r = \dot{e} + \Lambda e , \tag{13}$$

where $\Lambda = \Lambda^T > 0$ . Using (13), the arm dynamics can be rewritten in terms of the filtered tracking error as

$$\overline{M}_{rr}\dot{r} = -\overline{D}_{rr}r - \overline{u} + h(x) \tag{14}$$

where the nonlinear robot function is

$$h(x) = \overline{M}_{rr}(\overline{q})(\ddot{\overline{q}}_d + \Lambda\dot{e}) + \overline{D}_{rr}(\overline{q},\dot{\overline{q}})(\dot{\overline{q}}_d + \Lambda e) + \overline{F}_r(\dot{\overline{q}}) + \overline{G}_r(\overline{q}) \tag{15}$$

with $x = \begin{bmatrix}e^T & \dot{e}^T & \overline{q}_d^T & \dot{\overline{q}}_d^T & \ddot{\overline{q}}_d^T\end{bmatrix}^T$. It is assumed that $h(x)$ is unknown.

A neural network can be used to estimate $h(x)$ based on the universal approximation property of neural networks [9].

The estimate of $h(x)$ is given by

$$\hat{h}(x) = \hat{W}^T\sigma(\hat{V}^T x) , \tag{16}$$

Let

$$Z \equiv \begin{bmatrix}V & 0\\ 0 & W\end{bmatrix} \tag{17}$$

be the ideal weight matrix, which is unknown.

The functional approximation error of the neural network is

$$\widetilde{h}(x) = h(x) - \hat{h}(x) \tag{18}$$

which can be written using a Taylor expansion, assuming smooth activation functions, as

$$\widetilde{h}(x) = \widetilde{W}^T(\hat{\sigma} - \hat{\sigma}'\hat{V}_h^T x) + \hat{W}^T\hat{\sigma}'\widetilde{V}^T x + w \tag{19}$$

where

$$\hat{\sigma} \equiv \sigma(\hat{V}^T x), \qquad \hat{\sigma}' \equiv \left.\frac{\partial\sigma(z)}{\partial z}\right|_{z=\hat{z}} , \tag{20}$$

and the additional error term

$$w(t) = \widetilde{W}^T\hat{\sigma}'V^T x + W^T O(\widetilde{V}^T x)^2 + \varepsilon_{l_h}(x) \tag{21}$$

is bounded according to

$$w(t) \le C_0 + C_1\|\widetilde{Z}\| + C_2\|x\|\|Z\| . \tag{22}$$

The jacobian $\hat{\sigma}'$ is an easily computed function of $\hat{V}^T x$ .

It is assumed that the ideal weights of the neural network are bounded so that

$$\|Z\| \le Z_m \tag{23}$$

with $Z_m$ a known bound, and the desired trajectory is bounded according to

$$\left\|\begin{matrix}q_d\\ \dot{q}_d\\ \ddot{q}_d\end{matrix}\right\| \le Q \tag{24}$$

with $Q$ a known bound.

Under all the assumptions stated above, a neural network controller is defined by the following theorem [23].

**Theorem**

Let the desired trajectory and the ideal unknown weights be bounded according to the assumptions. Let the control input for (11) be defined by

$$\overline{u} = \hat{h} + K_v r - v \quad , \qquad \text{for } K_v = K_v^T > 0 \tag{25}$$

with robustifying term

$$v(t) = -K_z(\|\hat{Z}\| + Z_m)r \tag{26}$$

and gain $K_z > C_2$.

Let the neural network weights be tuned by

$$\begin{aligned}\dot{\hat{W}} &= M(\hat{\sigma} - \hat{\sigma}'\hat{V}^T x)r^T - \kappa\|r\|M\hat{W}\\ \dot{\hat{V}} &= Nxr^T\hat{W}^T\hat{\sigma}' - \kappa\|r\|N\hat{V}\end{aligned} \tag{27}$$

with any constant matrices $M = M^T > 0, N = N^T > 0$, and a scalar design parameter $\kappa > 0$ .

Then the filtered tracking error $r(t)$ and the neural network weight errors $\widetilde{V}, \widetilde{W}$ are globally uniformly ultimately bounded. Moreover, the tracking error may be kept as small as desired by increasing the gains $K_v$. □

The proof of this theorem uses Lyapunov theory and is given in [23], where explicit bounds on $\|r\|$ and $\|\hat{Z}\|$ are given. Notice that the training rules in (27) include the standard backpropagation terms plus an e-modification [15] and a second-order correction term. Furthermore, the NN weights can be easily initialized at zero since the PD control stabilizes the system while the NN is learning. The NN controller is designed to control the robot arm while it is learning to improve the performance, hence no off line training is required.

The overall structure of the controller defined in sections 3.1 and 3.2 is shown in Fig. 1.

## 4 Experimental Results

A list of the main characteristics of the practical implementation is given bellow:

- Only the first two flexible modes were considered.
- The robust term $v$ was not included. $K_v$ was selected big enough to avoid the necessity of $v$. The manifold term was not included since the actual model of the flexible link is unknown. As shown by (7), the implementation of $\bar{\xi}$ would require the exact knowledge of the matrices of the model.
- Even though the dynamics for the flexible link with one degree of freedom is linear, an extra nonlinear friction term was added to check the capability of the controller to compensate for the nonlinearities in the model and changes in these nonlinearities. With this extra friction we were able emphasize the advantages of the PD+NN controller over the standard approaches.
- The neural network is composed of ten neurons in the hidden layer, with five inputs ($x = \begin{bmatrix} e & \dot{e} & \bar{q}_d & \dot{\bar{q}}_d & \ddot{\bar{q}}_d \end{bmatrix}^T$) and one output $\hat{h}(x)$.
- The controller defined by (2), (10), (25), and (27) was discretized with sampling period of 5ms. In the discretization process the differential equations in (27) were solved on line using trapezoidal integration.
- The software was implemented in LabView and C.

Standard controllers PD and PID were implemented and tested in the flexible link test-bed to compare their performance with the PD+NN (PD and Neural Net) controller. This comparison allows us to show the advantages of the proposed controller over the standard controllers.

### 4.1 PD control

A PD controller was implemented using the control law

$$u = \bar{u} + u_F$$

with

$$\bar{u} = K_v r = K_v(\dot{e} + \Lambda e)$$

$$u_F = -\begin{bmatrix} K_{pF} & K_{dF} \end{bmatrix} \begin{bmatrix} q_f \\ \dot{q}_f \end{bmatrix}$$

using the parameters $K_v = 36$, $\Lambda = \frac{200}{36}$, $K_{pf} = \begin{bmatrix} -8 & 10 \end{bmatrix}$, $K_{df} = \begin{bmatrix} 0 & 0 \end{bmatrix}$ and the reference signal $q_d = 0.05 Sin(2\pi f t)$ with frequency $f$=0.5Hz.

The performance of the tracking PD control without the neural network is illustrated in Fig. 2(a) without the extra friction term, and Fig. 2(b) with the extra friction term. Notice that the tracking error is very big, its magnitude is comparable to that of the reference signal. Even though the magnitude of the error decreases incrementing the controller gains, the tracking error is not eliminated. These characteristics are preserved in the presence of the extra friction term.

### 4.2 PID Control

A PID controller was implemented using the control law

$$u = \bar{u} + u_F$$

with

$$\bar{u} = K_v r + K_i \int e\, dt = K_v(\dot{e} + \Lambda e) + K_i \int e\, dt$$

$$u_F = -\begin{bmatrix} K_{pF} & K_{dF} \end{bmatrix} \begin{bmatrix} q_f \\ \dot{q}_f \end{bmatrix}$$

using the parameters $K_v = 36$, $\Lambda = \frac{200}{36}$, $K_i = 100$, $K_{pf} = \begin{bmatrix} -8 & 10 \end{bmatrix}$, $K_{df} = \begin{bmatrix} 0 & 0 \end{bmatrix}$ and the reference signal $q_d = 0.05 Sin(2\pi f t)$ with frequency $f$=0.5Hz.

The performance of the tracking PID control is illustrated in Fig. 3(a) without the extra friction term, and Fig. 3(b) with the extra friction term. The integral part of the PID controller is supposed to eliminate the steady state error, but that only works for constant desired trajectories. In this case, with a varying desired trajectory, the tracking is even worse when the integral part is introduced (notice that the tracking error is bigger than with the PD control). As in the case of the PD controller, the PID controller is not able to compensate for the extra friction term.

### 4.3 NN+PD Control

The neural net tracking controller was implemented as described before using the parameters $K_v = 36$, $\Lambda = \frac{200}{36}$, $K_z = 0.2$, $Z_m = 50$, $\frac{K_{pf}}{\varepsilon^2} = \begin{bmatrix} -8 & 10 \end{bmatrix}$, $\frac{K_{df}}{\varepsilon} = \begin{bmatrix} 0 & 0 \end{bmatrix}$, $F = 2$, $G = 20$, $\kappa = 0.000001$. This value of $F$ in the practical implementation was enough. A bigger value produced a response very oscillatory.

The reference signal was $q_d = 0.05 Sin(2\pi f t)$ with frequency $f$=0.5Hz.

The performance of the neural network tracking control is illustrated in Fig. 4 before the learning is complete and in Fig. 5 after the learning is complete. The training of the neural network take less than one minute, after which the tracking error is reduced to almost zero. The learning is really active all the time (on-line training), but we refer to learning complete to the instant when the neural network have learned the model of the link under the actual conditions reducing the tracking error to almost zero.

Notice in Fig. 5(a), without the extra friction term, and Fig. 5(b), with the extra friction term, that the same controller learns the model of the link readapting to changes in it (changes in the model like changes in friction characteristics). Without changing the parameters of the controller, the neural network controller is able to take the tracking error to almost zero in both cases.

## 4.4 Comparison between different approaches

Comparing the tracking performance of the different controllers shown in Fig. 2 for the PD controller, Fig. 3 for the PID controller, and Fig. 4 and Fig. 5 for the neural net controller, it is clear the superiority of the last one. Even the PID cannot be a better tracking controller than the neural network controller under a varying desired trajectory. In this situation the PD controller is better than the PID, but not as good as the neural network controller. Here the error is seen as a delay due to the fact that the desired trajectory is a sinusoid and the dynamics of the controller with the link produces at its output another sinusoid with different amplitude and phase.

Even though the neural network tracking controller was not designed to track a step function (see assumption given by (24)), it was tested with a step desired trajectory for purposes of comparison with the PD and PID controllers described above. These controllers were tested with a step desired trajectory with and without the extra friction term. The results are plotted in Fig. 6.

Notice that the PD controller has good transient response (Fig. 6(a)) but is not able to get rid of the steady state error, and it gets worse in the case of the extra friction (Fig. 6(b)). Increasing the gains in this case improves the steady state error but makes the transient response more oscillatory.

The PID controller presents a worse transient response with a higher overshoot (Fig. 6(c)) but tries to eliminate the steady state error, even though is very slow (Fig. 6(d)). It is possible to increase the speed of the PID controller increasing $K_i$ but that produces a bad transient response with a big overshoot and very oscillatory, besides the tracking performance gets worse.

The neural network controller presents a response a little oscillatory but the overshot is not too high (Fig. 6(e)), being comparable to that of the PD controller, and it always takes the steady state error to zero, even it is able to compensate for the extra friction term (Fig. 6(f)). There is less oscillation with the extra friction term because of the extra damping that friction implies. The worse transient response of the PD+NN controller is explained by the fact that the desired trajectory (a step function in this case) have no bounded derivative, as it is required by the design. However, the main advantage of the NN+PD controller is that it does not require a priori knowledge of the model of the flexible robot arm and is able to compensate nonlinear effects that standard controllers don't.

## 5 Conclusions

The practical implementation of a multiloop nonlinear neural network tracking controller for a single flexible link has been tested and its performance compared to the one of the standard PD and PID controllers. An extra friction term was added in the implementation to show the ability of the neural network controller to learn and compensate for the nonlinearities.

The controller includes an outer PD tracking loop, a singular perturbation inner loop for stabilization of the fast dynamics, and a neural network inner loop used to feedback linearize the slow dynamics. This NN controller requires no off-line learning phase, the NN weights are easily initialized, and guarantees boundedness of the tracking error and control signal.

The practical results corroborate the simulations showing that standard PD or PID controllers are not able to track a varying desired trajectory, while the neural network controller takes the tracking error to almost zero readapting to any changes in the model of the link (extra friction terms).

## References

[1] S. Centikunt, B. Siciliano, and W. J. Book, "Symbolic modeling and dynamic analysis of flexible manipulators," in *Proc. IEEE Int. Conf. Syst., Man, Cybern.*, Oct. 1986, pp.798-803.

[2] F.-C. Chen and C.-C. Liu, "Adaptively controlling nonlinear continuous-time systems using multilayer neural networks," *IEEE Trans. Automat.Control*, vol. 39, no. 6, pp. 1306-1310, 1994.

[3] A. De Luca, and B. Siciliano, "Closed-form dynamic model of planar multilink lightweight robots," *IEEE Trans. on Systems Man and Cybernetics*, vol. 21, no. 4, pp. 826-839, Jul./Aug. 1991.

[4] G. G. Hastings and W. J. Book, "A linear dynamic model for flexible robotic manipulators," *IEEE. Control Systems Magazine*, vol. 7, no. 1, pp. 61-64, 1987.

[5] G. G. Hastings and W. J. Book, "Verification of a linear dynamic model for flexible robotic manipulators," in *Proc. IEEE Int. Conf. Robotics and Automation*, Apr. 1986, pp. 1024-1029.

[6] P. V. Kokotovic, H. K. Kalil, and J. O'Reilly, *Singular perturbation methods in control: analysis and design*. London: Academic Press, 1986.

[7] P. V. Kokotovic, "Applications of singular perturbation techniques to control problems," *SIAM Review*, vol. 26, no. 4, pp.501-550, Oct. 1984.

[8] P. V. Kokotovic, R. E. O'Malley Jr., and P. Sannuti, "Singular perturbations and order reduction in control theory-An overview," *Automatica*, vol. 12, pp.123-132, 1976.

[9] F. L. Lewis, K. Liu, and A. Ye| ildirek, "Neural net robot controller with guaranteed tracking performance," *IEEE Trans. Neural Networks,* vol.6, no. 3, pp. 703-715, May 1995.

[10] F. L. Lewis, K. Liu, and A. Ye| ildirek, "Neural net robot controller: structure and stability proofs," in *Proc. 32nd IEEE Conf. Decision and Control*, Dec. 1993, pp. 2785-2791.

[11] J. Lin, and F. L. Lewis, "Dynamic equations of a manipulator with rigid and flexible links: derivation and symbolic computation," in *Proc. American Control Conf.*, Jun. 1993, pp. 2868-2872.

[12] S. K. Madhavan and S. N. Singh, "Inverse trajectory control and zero dynamics sensitivity of an elastic manipulator," in *Proc. 1991 American Control Conference*, Jun.1991, pp. 1879-1884.

[13] W.T. Miller, R.S. Sutton, P.J. Werbos, ed., *Neural Networks for Control*. Cambridge, MA: MIT Press, 1991.

[14] K.S. Narendra, "Adaptive Control Using Neural Networks," *Neural Networks for Control*, pp 115-142. ed. W.T. Miller, R.S. Sutton, P.J. Werbos, Cambridge: MIT Press, 1991.

[15] K.S. Narendra and A.M. Annaswamy, "A new adaptive law for robust adaptation without persistent excitation," *IEEE Trans. Automat. Control*, vol. AC-32, no. 2, pp. 134-145, 1987.

[16] M.M. Polycarpou and P.A. Ioannou, "Identification and control using neural network models: design and stability analysis," *Tech. Report 91-09-01*, Dept. Elect. Eng. Sys., Univ. S. Cal., Sept. 1991.

[17] G.A. Rovithakis and M.A. Christodoulou, "Adaptive control of unknown plants using dynamical neural networks," *IEEE Trans. Systems, Man, and Cybernetics*, vol. 24, no. 3, pp. 400-412, 1994.

[18] N. Sadegh, "A perceptron network for functional identification and control of nonlinear systems," *IEEE Trans. Neural Networks* vol. 4, no. 6, pp. 982-988, 1993.

[19] R.M. Sanner and J.-J.E. Slotine, "Stable adaptive control and recursive identification using radial gaussian networks," in *Proc. IEEE Conf. Decision and Control*, Brighton, 1991.

[20] B. Siciliano and W. J. Book, "A singular perturbation approach to control of lightweight manipulators," *Int. J. Robotics Research* vol.7, no. 4, pp.79-90, Aug. 1988.

[21] M. W. Vandegrift, and F. L. Lewis, and S. Zhu, "Flexible-link robot arm control by a feedback linearization/singular perturbation approach," *J. Robotic Systems*, vol. 11, no. 7, pp.591-603, 1994.

[22] D. Wang and M. Vidyagasar, "Transfer functions for a single flexible link," *Int. J. Robotics Research*, vol. 10, no. 5, pp. 540-549, Oct. 1991.

[23] A. Yeşildirek, M. W. Vandegrift, and F. L. Lewis, "A neural network controller for flexible-link robots," in *IEEE Int. Symp. Intelligent Control* Aug. 1994, pp. 63-68.

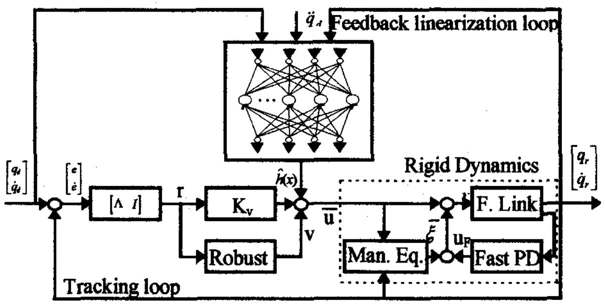
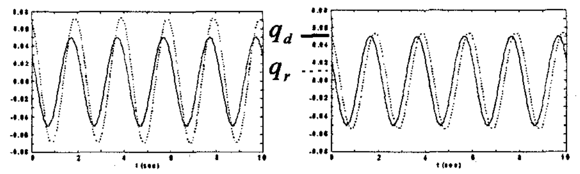
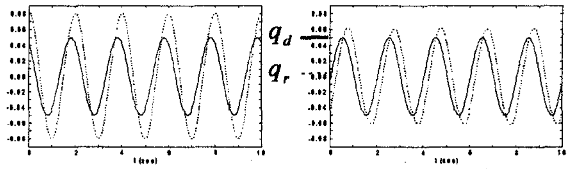

$\ddot{q}_d$
Feedback linearization loop
Rigid Dynamics
$\begin{bmatrix} q_d \\ \dot{q}_d \end{bmatrix}$
$\begin{bmatrix} e \\ \dot{e} \end{bmatrix}$
$[\Lambda \; I]$
r
$K_v$
$\hat{h}(x)$
$\bar{u}$
v
Robust
F. Link
$\begin{bmatrix} q_r \\ \dot{q}_r \end{bmatrix}$
Man. Eq.
$\bar{\xi}$
$u_F$
Fast PD
Tracking loop

**Fig. 1.** Overall control structure of the neural network controller for a flexible link robot.



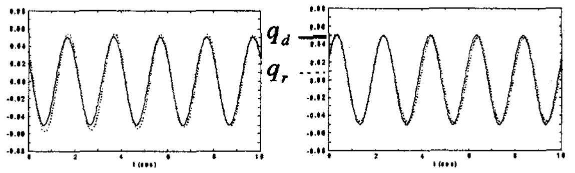
(a) (b)

**Fig. 2.** Performance of the PD control. (a) Without additional friction. (b) With additional friction.



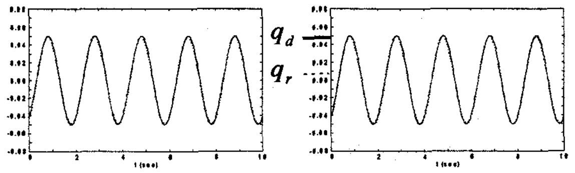
(a) (b)

**Fig. 3.** Performance of the PID control. (a) Without additional friction. (b) With additional friction.



(a) (b)

**Fig. 4.** Performance of the PD+NN control before learning. (a) Without additional friction. (b) With additional friction.



(a) (b)

**Fig. 5.** Performance of the PD+NN control after learning. (a) Without additional friction. (b) With additional friction.

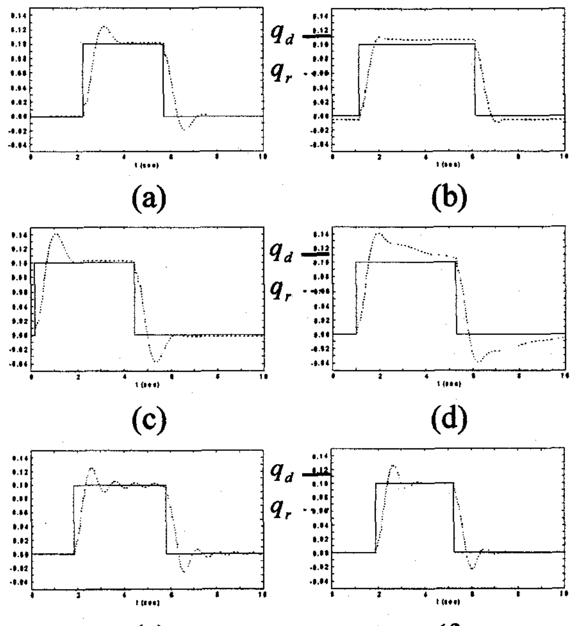


**Fig. 6.** Step response of the controllers. (a) PD control. (b) PD control with extra friction. (c) PID control. (d) PID control with extra friction. (e) PD+NN control. (f) PD+NN control with extra friction.