Asian Journal of Information Technology 4 (10): 927-934, 2005

Tracking Control for Robot Manipulator Based on Neural Networks with Adaptive Learning Rate

[1]Noureddine Guersi,[1]Messaoud Djeghaba,[2]Dimitri Lefebvre,[2]Fabrice Druaux,[2]Edouard Leclercq
[1]Laboratoire d'Automatique et des Signaux de Annaba
University Badji-Mokhtar B.P 12 El-Hadjar Annaba 23000, Algeria,
[2]Groupe de Recherche en Electrotechnique et Automatique du Havre,
25, rue Philippe Lebon BP 540, 76058 Le Havre, France

Abstract: The selection of learning rates to obtain satisfactory performances for neural network controllers is a challenging problem. In order to skip any time consuming experimentation for the choice of an appropriate value of the learning rate, this paper is concerned with an online adaptive learning rate algorithm derived from the convergence analysis of the usual gradient descent method. Based on the feedback linearization method, a multilayer neural network controller approximates online the unknown dynamics of the system including the non–linear behaviours. The proposed controller does not require any preliminary off-line training. A stability proof of this control scheme is given. Simulations and a comparison with a PD controller and several fixed learning rate neural controllers illustrate the effectiveness of the proposed algorithm in case of adaptive control for robot trajectory tracking.

Key words: Multilayer neural networks, gradient descent method, adaptive learning rate, stable on-line MIMO control, robot trajectory tracking

INTRODUCTION

In the past decade,he application of intelligent control techniques (fuzzy control or neural- network control) to the motion control for robot manipulators have received considerable attention [1,9]. A control system, which comprises PID control and neural network control, was presented by Chen *et al.*, [2] for improving the control performance of the system in real time. Clifton *et al.*, [3] and Misir *et al.*, [7] designed fuzzy-PID controllers which where applied to the position control of robot manipulators. Huang and Lee [5] suggested a stable self-organizing fuzzy controller for robot motion control. This approach has a learning ability for responding to the time-varying characteristic of a robot manipulator. However the fuzzy rule learning scheme has a latent stability problem. Yoo and Ham [9] presented two kinds of adaptive control schemes for robot manipulator via fuzzy compensator in order to confront the unpredictable uncertainties. Though the stability of the whole control system guaranteed, some strict constrained conditions and prior system knowledge are required in the control process. On the other hand, Kim and Lewis [6] dealt with the application of quadratic optimisation for motion control of robotic systems using cerebellar model arithmetic computer neural networks. Lewis *et al.*, [1] developed a multilayer neural-net controller for a general serial-link rigid robot to guarantee the tracking performance. Both system-tracking stability and error convergence can be guaranteed in this neural-based control system [1,6].

Several works related to the use of artificial neural networks (NN) in identification and control applications are reported in the literature [10,11]. One challenging problem of the usual back-propagation algorithm for multilayer NN [12] is the determination of the learning rate (LR), which has to be made with care. A lot of methods are based on fixed LR. If the LR is large, learning may occur quickly, but it may also become unstable. To ensure stable learning, the LR must be sufficiently small. However, with a small learning rate, the NN may adapt reliably, but the learning may take quite a long time. It is thus difficult to select a suitable fixed LR for different initial values of the NN parameters and for different NN structures. This difficulty is a basic characteristic of the NN learning rule that results from the gradient descent (GD) method [13,14]. Such method is known for its slowness and its tendency to become trapped in local minima. To reduce these shortcomings, a number of faster NN training algorithms have been developed, such as adaptive learning algorithms [5,9] and other improved algorithms [16,17]. One may also use second-

Corresponding Author: Noureddine Guersi, Laboratoire d, Automatique et des Signaux de Annaba (LASA),University Badji-Mokhtar B.P 12 El-Hadjar Annaba 23000, Algeria

order non-linear optimising methods to accelerate the learning, such as the conjugate gradient algorithm [18] or the Levenberg-Marquardt based method [19]. In spite of their better convergence, these methods are not based on the optimal instantaneous learning rates of the GD approach. Moreover, some critical drawbacks of such methods have to be noticed: the ill conditioning of the Hessian matrix in many applications and the computational complexity related to the Hessian calculation. In addition, most of these algorithms are developed only for off-line NN training.

Our approach concerns the investigation of adaptive learning rate algorithms. The main contribution is to extend the results obtained by D. Sha and V. B. Bajic for the modelling of SISO non-linear systems [20,21] to the modelling and control of MIMO ones [22]. For this purpose, multilayer NN will be used to model the unknown non-linear behaviours of the system to be controlled. The adaptive control design results from the neural model in order to track a reference trajectory. Simulations and comparisons with a PD controller and several fixed LR neural controllers illustrate the effectiveness of the proposed algorithm in an adaptive control for robot trajectory tracking.

Preliminaries: Consider a mn th order multi-input multi-output continuous time system given by cmpas *et al.*, [23] by:

$$\dot{x}_1 = x_2, \cdots, \dot{x}_{n-1} = x_n, \dot{x}_n = f(x) + u(t) + d(t), y = x_1 \qquad (1)$$

with ,$x = [x_1^T \; x_2^T \ldots x_n^T]^T \in \Re^{mn}$ $x_i(t) \in \Re^m$ $i = 1,2,\ldots n$, $u(t) \in \Re^m$ $d(t) \in \Re^m$ $f(x): \Re^{mn} \to \Re^m$ and $y(t) \in \Re^m$ x0 is the state vector, *u(t)* is the input vector, *d(t)* denotes the unknown disturbance, *f(x)* is an unknown smooth function and *y(t)* is the output vector. Many physical systems, such as robotic ones can be represented in this form. It is assumed that the non-linear function *f(x)* and the external disturbances *d(t)* are unknown to the controller. Given a desired trajectory and its derivatives values $x_d(t) = [y_d^T \; \dot{y}_d^T . y_d^{(n-1)T}]^T$, let us define the tracking error as $e(t) = y(t) - y_d(t) \in \Re^m$ which captures the performance of the closed-loop system output *y(t)* in tracking the desired trajectory $y_d(t)$. It is typical in robotics to define a so-called filtered tracking error as $r(t) \in \Re^m$:

$$r(t) = e^{(n-1)}(t) + \lambda_{n-1} e^{(n-2)}(t) + \ldots + \lambda_1 e(t) \; ; \qquad (2)$$

Where $e^{(n-1)}(t), \ldots, e^{(1)}(t)$ are the derivative values of the error *e(t)*, and $\lambda_1, \ldots, \lambda_{n-1}$ are constant values selected so that $\left|s^{n-1} + \lambda_{n-1}.s^{n-2} + \ldots + \lambda_1\right|$ is stable. The performance measure *r(t)* can be viewed as the real-valued

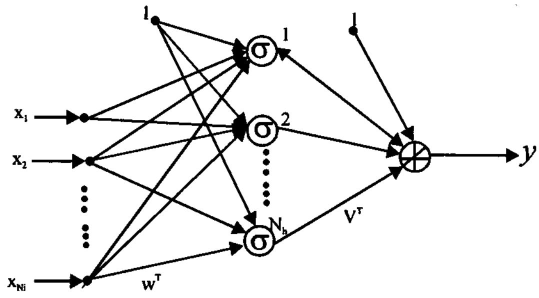

Fig.1: Single input multilayer NN

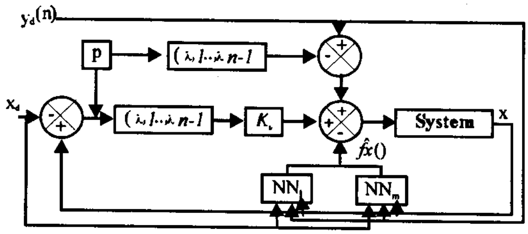

Fig. 2: NN controller

instantaneous utility function of the plant performance: smaller *r(t)*, better the system performan ce.

The single output of a two-layer NN with a linear output activation function is given by (Fig.1) :

$$y_{NN} = \sum_{j=1}^{N_h} v_j \sigma\left(\sum_{k=1}^{N_i} w_{jk}.\varphi_k + \theta_{wj}\right) + \theta_v \qquad (3)$$

where $\varphi_1, \ldots, \varphi_{Ni}$ are the NN inputs, *σ(.)* is a sigmoïdal activation function, w_{jk} are input-to-hidden layer interconnection weights, and θ_v, θ_{wm} m =1,2 ,... are bias. N_i, N_h are the numbers of neurons in the input and hidden layers. By collecting all the NN weights w_{jk}, ,v_j and bias θ_v, θ_{wm} into matrices W^T, V^T (the bias are included as the first column of the weight matrices V^T and W^T) equation 3 may be written in terms of vectors as $y_{NN}(\varphi) = V^T \underset{\sim}{\sigma}(W^T.\varphi)$ with $\varphi = [1, \varphi_1, \ldots, \varphi_{Ni}]^T \in \Re^{(N_i+1)x1}$ For a suitable number of hidden neurons N_h, there exists constant weights and bias such that the estimate of any smooth non-linear function *g(φ)* from $\Re^{N_i+1}$ to $\Re$ is given by $\hat{g}(\varphi) = \hat{V}^T \sigma(\hat{W}^T.\varphi)$ where $\hat{W}$, $\hat{V}$ are estimates of the ideal NN weights w,v [24,26].

NN controller: The control scheme consists of a PD feedback controller and a multilayer neural controller (Fig.2). In the feedback loop, the fixed gain PD controller makes the overall system stable along a desired trajectory. The NN is used to approximate the unmodeled dynamics.

The use of an on-line variable LR algorithm, makes the adaptation process less complicated than other NN schemes, and improves the error convergence speed.

Using equation 1 the dynamics of the performance measure signal Eq 2 can be written as:

$$\dot{r}(t) = f(x) + u(t) + d(t) - y_d^{(n)} + \lambda_{n-1}e^{(n-1)} + \cdots + \lambda_1 e^{(1)} \in \Re^m \quad (4)$$

According to the approximation properties of NN, the continuous non-linear functions $f_i(x)$ components of the vector f(x) can be estimated by $\hat{f}_i(x) = \hat{V}_i^T \sigma(\hat{W}_i^T.\varphi)$ where:

$$\varphi = [1 \quad x_1^T \quad \dot{x}_1^T \cdots x_1^{(n-1)T} \quad y_d^T \quad \dot{y}_d^T \cdots y_d^{(n)T}]^T \in \Re^{1+m(2n+1)}$$

is the NN input vector, and $\hat{V}_i \in \Re^{1+N_h}$ are the estimates of V_i and W_i. A robust compensation scheme is provided by selecting the control input *u(t)* as [1,23]

$$u(t) = K_V r - \hat{f}(x) + y_d^{(n)} - \lambda_{n-1}e^{(n-1)} - \cdots - \lambda_1 e^{(1)} \quad (5)$$

where $K_v \in \Re^{m \times m}$ is the control gain matrix such that $K_v = K_v^t > 0$. The determination of the feedback control gain matrix K_V is well known and not detailed in this work. Let us define the estimation errors as $\tilde{V}_i = V_i - \hat{V}_i$, $\tilde{W}_i = W_i - \hat{W}_i$ and $\hat{\sigma} = \sigma(\hat{W}_i^T.\varphi)$, with. $\tilde{\sigma}(.) = \sigma(.) - \hat{\sigma}(.)$ Using Eq 5, we can rewrite the closed-loop performance measure dynamics Eq 4 as:

$$\dot{r} = K_V r + \varepsilon(x) + d(t) \quad (6)$$

where the functional estimation error is defined as, $\varepsilon(x) = f(x) - \hat{f}(x)$ with $\|\varepsilon(x)\| \le \varepsilon_M(x)$ for some known bounding functional error $\varepsilon_M(x)$.

Adaptive learning rate algorithm: Let us consider the error equation in discrete time, with a sampling period Δt:

$$\Delta\varepsilon_i(t) = \varepsilon_i(t+\Delta t) - \varepsilon_i(t) = \Delta f_i(x(t)) - \Delta\hat{f}_i(x(t)),$$

where: $\varepsilon_i(t) = f_i(x(t)) - \hat{f}_i(x(t))$, and

$$\Delta f_i(x(t)) = f_i(x(t+\Delta t) - f_i(x(t))$$

Let us assume that $|\Delta f_i(x(t))| << |\Delta\hat{f}_i(x(t))|$, [20,21] (i.e. the variations of the function to be estimated are slower compared to the variations of the NN output). This assumption is realistic for many processes. Then, during the parameters adaptation of the NN, the error equation is given by:

$$\Delta\varepsilon_i(t) = -\left(\left(\sigma(\hat{W}_i^T\varphi)\right)^T.\left(\Delta\hat{V}_i\right) + \hat{V}_i^T.\dot{\sigma}(\hat{W}_i^T\varphi).\left(\Delta\hat{W}_i\right)^T.\varphi\right) \quad (7)$$

with:

$$\sigma'(\hat{W}_i^T\varphi) = \begin{pmatrix} 0 & \cdots & 0 \\ \sigma'(\hat{w}_{i,1}^T\varphi) & 0 & 0 \\ 0 & \ddots & 0 \\ 0 & 0 & \sigma'(\hat{w}_{i,N_h}^T\varphi) \end{pmatrix} \in \Re^{(N_h+1)\times N_h}$$

and:

$$\sigma'(\hat{w}_{i,k}^T\varphi) = \frac{\beta.e^{-\beta.\hat{w}_{i,k}^T\varphi}}{\left(1 + e^{-\beta.\hat{w}_{i,k}^T\varphi}\right)^2}$$

Considering a criterion $J_i = 1/2.(\epsilon_i(t))^2$, the standard GD method [20,21], leads to:

$$\Delta\hat{V}_i = \eta_i.\sigma(\hat{W}_i^T.\varphi).\varepsilon_i.\Delta t \qquad I = 1,\ldots,m \quad (8a)$$

$$\Delta\hat{W}_i = \eta_i.\varphi.\left(\hat{V}_i\right)^T.\dot{\sigma}(\hat{W}_i^T.\varphi).\varepsilon_i.\Delta t \qquad I = 1,\ldots,m \quad (8b)$$

Replacing $\Delta\hat{V}_i(t)$ and $\Delta\hat{W}_i(t)$ by their expressions given by Eq 8, we obtain $\Delta\varepsilon_i(t) \approx -\eta_i(t).\zeta_i(t).\varepsilon_i(t)$ with:

$$\xi_i \approx \Delta t.\left(\left(\sigma(\hat{W}_i^T.\varphi)\right)^T.\sigma(\hat{W}_i^T.\varphi) + \left(\hat{V}_i\right)^T.\dot{\sigma}(\hat{W}_i^T.\varphi).\left(\dot{\sigma}(\hat{W}_i^T.\varphi)\right)^T.\hat{V}_i.\varphi^T.\varphi\right) \quad (9)$$

Thus $\varepsilon_i(t+\Delta t) \approx [1 - \eta_i(t).\zeta_i(t)].\varepsilon_i(t)$ As a consequence, the error $\varepsilon_i(t)$ tends to 0 when *t* tends to infinity if the condition $0 < \eta_i(t) < 2.\zeta_i^{-1}(t)$ or $|\eta_i(t)| \le 2\zeta_i^{-1}(t) = \eta_M$ is satisfied. Let us notice that the upper bound $2.\zeta_i^{-1}(t)$ of the learning rate $\eta_i(t)$ is variable because the value of $\zeta_i^{-1}(t)$ depends on the input φ and the current values of the NN parameters $\hat{V}$ and $\hat{W}$. In order to obtain the fastest learning, the LR and the weights are adapted according to $\eta_i(t) = \zeta_i^{-1}(t)$, and:

$$\Delta\hat{V}_i(t) = \frac{\Delta t}{\zeta_i(t)}.\sigma(\hat{W}_i^T(t).\varphi(t)).\varepsilon_i(t), \quad i=1,\ldots,m \quad (10a)$$

$$\Delta\hat{W}_i(t) = \frac{\Delta t}{\zeta_i(t)}.\varphi(t).\left(\hat{V}_i(t)\right)^T.\dot{\sigma}(\hat{W}_i^T(t).\varphi(t)).\varepsilon_i(t), \quad i=1,\ldots,m \quad (10b)$$

These on line updating rules are used in the simulations of section 6.

Stability Analysis: For the neural network training algorithm to improve the tracking performance of the closed-loop system it is required to demonstrate that the tracking error, r_i is suitably small. Theorem 1 provide sufficient conditions for stability

Theorem 1: The system (1) with control input defined as in (5) and (10) is stable if the following conditions are satisfied:

$$\varepsilon_i \ge \delta_i + \alpha_i \ge 0$$

$$\|r\|.\cos(k_{vi}, r) \le \frac{-(\varepsilon_i + d_i)}{\|k_{vi}\|} \text{ if } r_i \ge 0 \text{ and}$$

$$\|r\|.\cos(k_{vi}, r) \ge \frac{-(\varepsilon_i + d_i)}{\|k_{vi}\|} \text{ if } r_i \le 0. \quad (11)$$

Proof: Let us define the Lyapunov function for the i^{th} output :

$$L_i = \frac{1}{2}r_i^2 + \frac{1}{2}tr(\tilde{W}_i^T.\tilde{W}_i) + \frac{1}{2}.(\tilde{V}_i^T.\tilde{V}_i) \quad (12)$$

where $tr(.)$ stands for the trace of (.), hence

$$\dot{L}_i = \dot{r}_i r_i + tr(\tilde{W}_i^T.\dot{\tilde{W}}_i) + (\tilde{V}_i^T.\dot{\tilde{V}}_i)$$

with $\dot{r}_i(t)$ given by equation (6) we obtain

$$\dot{L}_i = (k_{Vi}.r + \varepsilon_i(x) + d_i(t))r_i + tr(\tilde{W}_i^T.\dot{\tilde{W}}) + (\tilde{V}_i^T.\dot{\tilde{V}}_i) \;;$$

k_{Vi} is the i^{th} row of matrix K_V

Since $\dot{\tilde{W}}_i = -\dot{\hat{W}}_i = -\eta_i.\varphi.(\hat{V}_i)^T\dot{\hat{\sigma}}.\varepsilon_i$ with constant (similarly for $\dot{\tilde{V}}_i = -\dot{\hat{V}} = -\eta_i\hat{\sigma}\varepsilon_i$). Substitution of the training rules gives

$$\dot{L}_i = (k_{Vi}.r + \varepsilon_i + d_i)r_i + tr(\tilde{W}_i^T(-\eta_i.\varphi\hat{V}_i^T\dot{\hat{\sigma}}\varepsilon_i)) + .\tilde{V}_i^T(-\eta_i\hat{\sigma}\varepsilon_i)$$

with $tr(A.B)=tr(B.A)$, we have:

$$\dot{L}_i = k_{Vi}.r.r_i + (\varepsilon_i + d_i)r_i - \eta_i\varepsilon_i.(tr(\hat{V}_i^T\dot{\hat{\sigma}}.\tilde{W}_i^T\varphi) + \tilde{V}_i^T.\hat{\sigma}) \quad (13)$$

Where $\hat{V}_i^T\dot{\hat{\sigma}}.\tilde{W}_i^T\varphi$ is a scalar, then

$$\dot{L}_i = k_{Vi}.r.r_i + (\varepsilon_i + d_i)r_i - \eta_i\varepsilon_i.(\hat{V}_i^T\dot{\hat{\sigma}}.\tilde{W}_i^T\varphi + \tilde{V}_i^T.\hat{\sigma}) \quad (14)$$

with $\varepsilon_i = V_i^T\sigma(W_i^T\varphi) - \hat{V}_i^T\sigma(\hat{W}_i^T\varphi) + \alpha_i(\varphi)$
where the functional estimation error $\alpha_i(\varphi)$ is bounded. Adding and subtracting $V_i^T\hat{\sigma}$ and $\hat{V}_i^T\tilde{\sigma}$ to ε_i leads to:

$$\varepsilon_i = V_i^T\hat{\sigma} + \hat{V}_i^T\tilde{\sigma} + \tilde{V}_i^T\tilde{\sigma} + \alpha_i(\varphi) \quad (15)$$

The Taylor series expansion of $\sigma(W_i^T.\varphi)$ for a given φ may be written as:

$$\sigma(W_i^T\varphi) = \sigma(\hat{W}_i^T\varphi) + \dot{\sigma}(\hat{W}_i^T\varphi).\tilde{W}^T\varphi + o(\tilde{W}_i^T\varphi) \quad (16)$$

with: $\dot{\sigma}(\hat{W}_i^T\varphi)$ as defined by Eq 10. Equations 16 can be rewritten as:

$$\tilde{\sigma}(W_i^T\varphi) = \dot{\sigma}(\hat{W}_i^T\varphi)\tilde{W}_i^T\varphi + o(\tilde{W}_i^T\varphi) \quad (17)$$

Substituting Eq 17 in Eq 15 the functional error ε_i becomes

$$\varepsilon_i = \tilde{V}_i^T\hat{\sigma} + \hat{V}_i^T\dot{\sigma}(\hat{W}_i^T.\varphi).\tilde{W}_i^T\varphi + \delta_i + \alpha_i(\varphi) \quad (18)$$

Where $\delta_i = \tilde{V}_i^T.\tilde{\sigma} + \hat{V}_i^T.o(\tilde{W}_i^T\varphi)$ corresponds to high-order terms in the Taylor series and is bounded by positive constant δ_M, (i.e, $|\delta_i| < \delta_M$).

It is important to note that the neural network reconstruction error ϵ_i, the plant disturbance d_i, and the high-order terms δ_i in the Taylor series expansion of f all act as disturbances in the error system. From Eq 18 we have:

$$\hat{V}_i^T\dot{\hat{\sigma}}.\tilde{W}_i^T\varphi + \tilde{V}_i^T.\hat{\sigma} = \varepsilon_i - \delta_i - \alpha_i$$

so we can rewrite Eq 14 as :

$$\dot{L}_i = k_{Vi}.r.r_i + (\varepsilon_i + d_i)r_i - \eta_i\varepsilon_i(\varepsilon_i - \delta_i - \alpha_i) \quad (19)$$

Thus $\dot{L}_i$ is negative as long as:

$$-\eta_i\varepsilon_i(\varepsilon_i - \delta_i - \alpha_i) \le 0 \quad (20)$$

and $$k_{Vi}.r.r_i + (\varepsilon_i + d_i)r_i \le 0 \quad (21)$$

Equation 20 is satisfied as long as ϵ_i and $\epsilon_i - \delta_i - \alpha_i$ have the same sign. A sufficient condition is given according to

$$\varepsilon_i \ge \delta_i + \alpha_i \ge 0 \quad (22)$$

Similarly, equation (21) is satisfied as long as r_i and $k_{Vi}.r + (\varepsilon_i + d_i)$ have opposite signs. A sufficient condition is given according to

$$\|r\|.\cos(k_{vi}, r) \le \frac{-(\varepsilon_i + d_i)}{\|k_{vi}\|} \quad \text{if } r_i \ge 0 \quad (23a)$$

$$\|r\|.\cos(k_{vi}, r) \ge \frac{-(\varepsilon_i + d_i)}{\|k_{vi}\|} \quad \text{if } r_i \le 0 \quad (23b)$$

In order to satisfy Eq 23a and Eq 23b the error vector r must remain in a convex domain included in $\Re^m$.

Simulation Experiments: To illustrate the performance of the proposed NN controller, a two-link robot arm (Fig. 3.) is simulated. The dynamics equation for such a manipulator is given by:

$$M(\dot{q})\ddot{q} + Vm(q,\dot{q})\dot{q} + G(q) + F(\dot{q}) + \tau_d = \tau \quad (24)$$

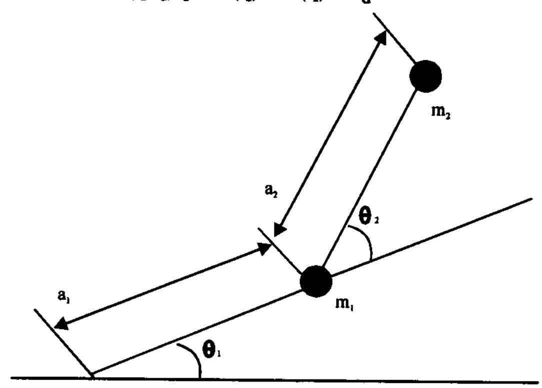

Fig. 3: Two-link robot arm.

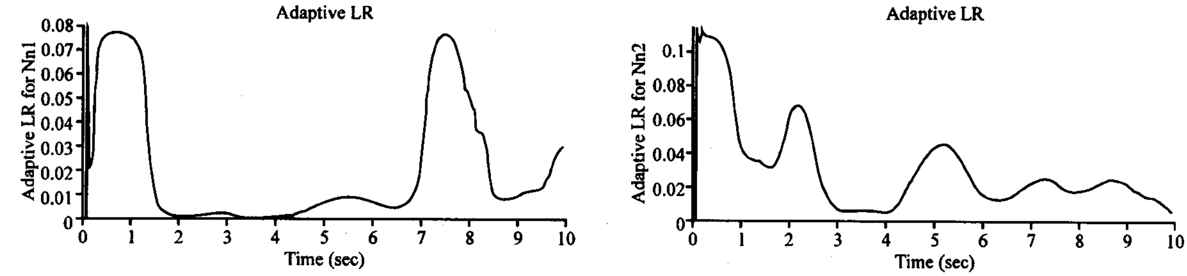

Fig. 4: Adaptive LR for NN1 (left) and NN2 (right) in function of time

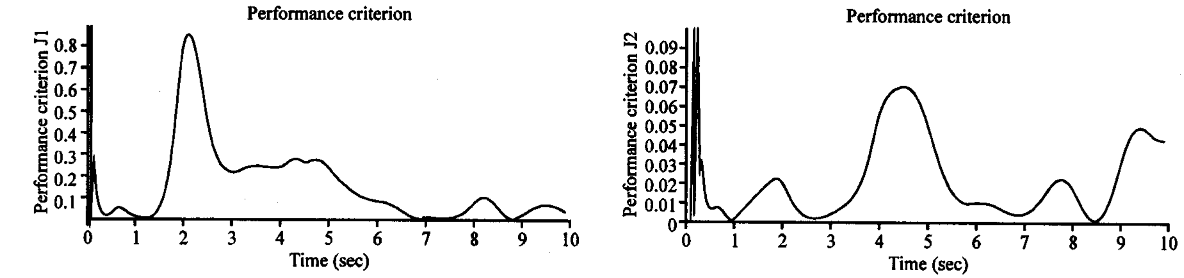

Fig. 5: Performance criterion J for NN1 (left) and NN2 (right) in function of time

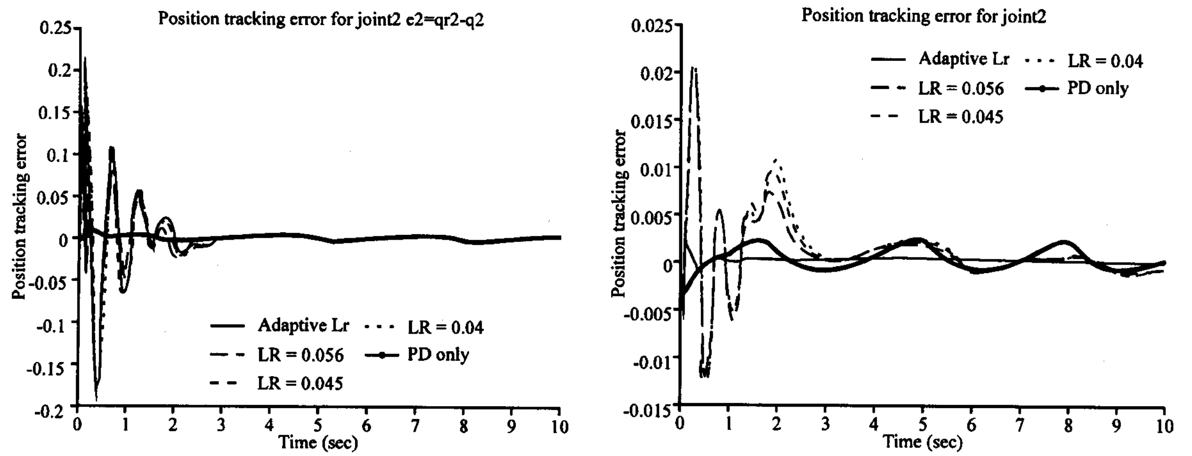

Fig. 6: Position (left) and velocity (right) tracking errors for link 2 in function of time

where $q(t) \in \Re^2$ is the joint variable $M(\dot{q})$ vector, is the inertia matrix, $Vm(q,\dot{q})$) is the Coriolis / centripetal matrix, $G(q)$ is the gravity vector, and $F(\dot{q})$ is the friction. Bounded unknown disturbances (including unstructured, unmodeled dynamics) are denoed by τ_d and the control input torque is τ_d. The Equation 24 can be written in the Brunovsky form 25.

Assuming that is known, $u(t)$ can be computed as in Eq 5. It is important to notice that non-linear terms such as friction, gravity, and Coriolis terms are unknown. The system parameters are $a_1 = 1.0$, $a_2 = 1.0$, $m_1 = 1$ and $m_2 = 2.3$. The controller is composed of two NN that approximate $f(x) = [f_1(x) \;\; f_2(x)]^T$ plus a PD feedback gain.

The NN input vector is given by $\varphi = [1 \;\; q^T \;\; \dot{q}^T \;\; q_d^T \;\; \dot{q}_d^T \;\; \ddot{q}_d^T]^T$ The NN have $N_h = 10$ hidden-layer nodes. The controller parameters are chosen as $\lambda = 5$,K_v= diag{20,20}. The reference signals used for each joint are $q_{d1}(t) = \sin(t)$, $q_{d2}(t) = \cos(t)$ with initial conditions $q(0) = q_d(0)$ and $\dot{q}(0) = \dot{q}_d(0)$.

$$\begin{cases} \dot{x}_1 = x_2 \\ \dot{x}_2 = f(x) + u + d \end{cases}$$

$$x_1 = [q_1 \;\; q_2]^T, \quad x_2 = [\dot{q}_1 \;\; \dot{q}_2]^T$$

$$u = M^{-1}(\dot{q}).\tau, \quad d = M^{-1}(\dot{q}).\tau_d$$

$$f(x) = -M^{-1}(\dot{q})[V_m(q,\dot{q}) + G(q) + F(\dot{q})] \qquad (25)$$

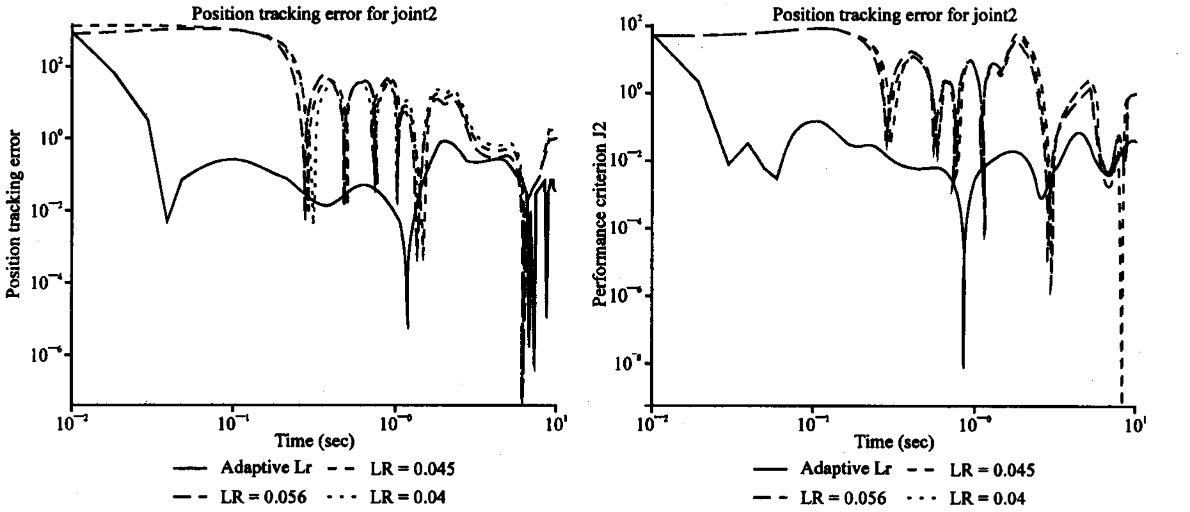

Fig. 7. Comparison of the performance criterions J1 (left) and J2 (right) in log-log scale

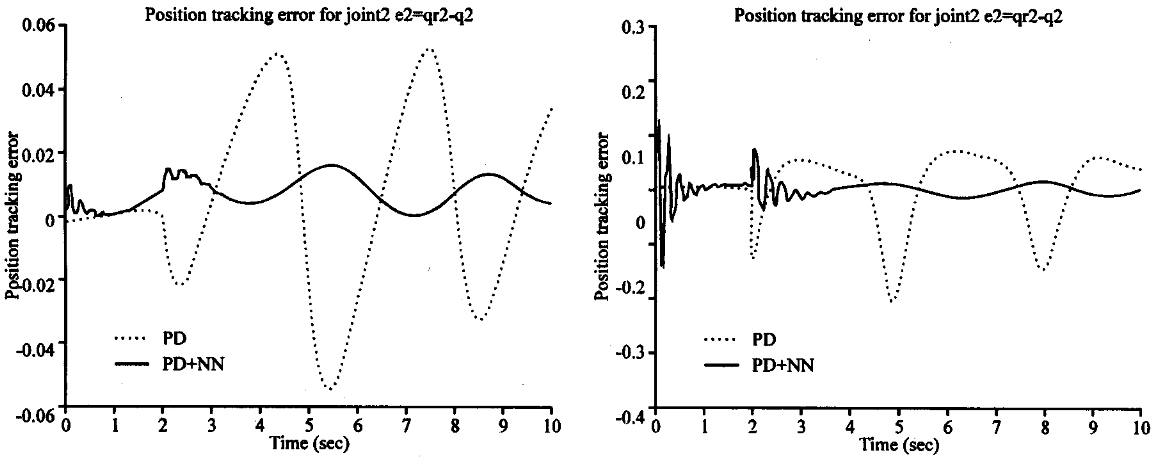

Fig. 8: Position and velocity tracking errors for link 2 :m2 parameter change at t=2s

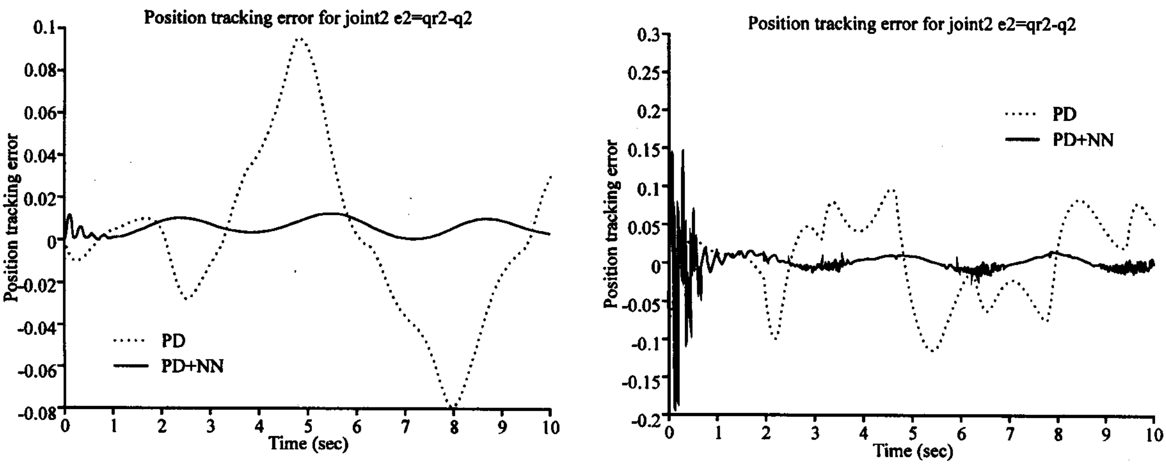

Fig .9: Position and velocity tracking errors for link 2 : Disturbance and friction forces injection at t=2s

The Fig. 4 and 5 show the adaptive LR and the performance criterions J1 for NN1 and J2 for NN 2. The adaptive LR increases as the inverse of the performance criterion, and the non-linear function *f(x)* is quickly well approximated by NN 1 and NN 2.

The tracking performances (position and velocity) of PD control design, PD gain plus a fixed LR NN controller, and PD gain plus the adaptive LR NN controller have been compared for link 2 (Fig. 6). The tracking performance of the PD control design is not satisfactory: a steady-state error results from the non-linear dynamics. The tracking errors obtained with the fixed LR NN controllers, converge to smaller values, but the best value is obtained with the adaptive LR NN controller. Similar results were obtained for link 1. As a conclusion, the proposed algorithm is suitable to cancel the non-linear behaviours.

The Fig.7 compares the performance criterions obtained for fixed LR NN controllers and the adaptive LR NN controller. The variable LR algorithm achieves similar or better results than the others directly, without any requirements for tuning the learning process.

To test the robust characteristic of the proposed control system we first consider a parameter variation condition : at $t=2s$, 1kg is added to the mass of link 2, i.e $m2=$ 3.3kg, and then we consider a disturbance : at $t=2s$, external forces are injected into the robotic system according to [9]:

$$\tau_d(t) = [0.1\sin(t) \quad 0.1\sin(t)]^T$$

Moreover, friction forces are also considered in this simulation and are given as [5]:

$$F(\dot{q}) = [0.3\dot{q}_1 + 0.2\,\mathrm{sgn}(\dot{q}_1) \quad 0.3\dot{q}_2 + 0.2\,\mathrm{sgn}(\dot{q}_2)]^T$$

Tracking errors for link 2 are depicted in fig.8.in the case of parameter *m2* change, and in Fig.9 in the case of injection of external disturbance in the robot system.

Since all the parameters and weights of the NN are randomly initialized, the tracking errors are gradually reduced through on-line training methodology of the PD plus NN with adaptive LR control system.

In case of joint friction, parameter variation and external disturbance, the tracking errors remain relatively small for the system with PD plus NN. On the other hand for the system with PD alone one notes a relative increase in the error.

Moreover, the robust control performance of this control scheme, in case of joint friction, parameter variation and external disturbance are suggested as shown in fig.8 and Fig.9, compared with the simulated results of the PD position control, the proposed control scheme is effective and yields superior tracking performance.

CONCLUSIONS

In this paper, multilayer NN with adaptive LR are investigated to control non-linear continuous-time systems. The adaptive LR algorithm has an improved convergence that is useful to model the unknown non-linear dynamics of the system to be controlled. Such an algorithm is based on the analysis of the convergence of the GD method. Compared to a fixed LR algorithm, it makes the tuning process less complicated than other NN schemes, and results in similar or better performances in terms of learning speed and training error.

Our perspectives are to investigate further the indirect adaptive control schemes with NN. Stability issues and noise sensitivity will be studied according to the LR updating rule.

REFERENCES

1. Lewis, F.L., A. Yesildirek, and K. Liu, Multilayer 1996. Neural-Net Robot Controller with Guaranteed Tracking Performance, IEEE , Trans. Neural Networks, vol. 7, No. 2, pp. 388-399.
2. Chen, P.C.Y., J.K. Mills, and G. Vukovich, 1996. Neural network learning and generalization for performance improvement of industrial robots, Canadien Conference on Electrical and Computer Engineering .
3. Clifton, C., A. Homaifar, and M. Bikdash, 1996. Design of Generalized Sugeno controller by approximating hybrid fuzzy-PID controllers, IEEE International Conference on Fuzzy Systems, p. 1906.
4. Guitierrez, L.B., F.L. Lewis, and J.A. Lowe, 1998. Implementation of a neural network traking controller for a single flexible link : comparison with PD and PID controller IEEE Trans. Ind. Electron. 45: 307.
5. Huang, S.J., and J.S. Lee, 2000. A stable self-organizing fuzzy controller for robotic motion control, IEEE Trans. Ind. Electron. 47: 421.
6. Kim, Y.H., and F.L. Lewis, 2000. Optimal design of CMAC neural-network controller for robot manipulators, IEEE Trans. Systems Man Cybernet. 30: 22.
7. Misir, D., H.A. Malki, and G. Chen, 1998. Graphical stability analysis for a fuzzy PID controlled robot arm model, IEEE International Conference on Fuzzy Systems, p: 451
8. Vemuri, A.T., and M.M. Polycarpou, 1997. Neural-network-based robust fault diagnosis in robotic system, IEEE Trans. Neural Networks 8: 1410.
9. Yoo, B.K., and W.C. Ham, 2000. Adaptive control of robot manipulator using fuzzy compensator, IEEE Trans. Fuzzy Systems., 8:186.List of Figures

10. IEEE, 1996 . Control system magazine. Special Issues on Neural Network Control Systems.
11. Narendra, K.S. and K. Parthasarathy, 1990. Identification and control of dynamical systems using neural networks. IEEE, Transaction on Neural Networks, 1: pp: 4-27, 1990.
12. Rumelhart, D.E. and J.L. McClelland, 1986. Parallel Distributed Processing. Cambridge, MA: MIT Press, 1986.
13. Widrow, B., and M.A. Lehr, 1990. 30 Years of Adaptive Neural Networks: Perceptron, Madaline, and Backpropagation, Proceedings of The IEEE, 78, Special Issue on Neural Networks, I: Theory & Modelling, pp: 1415-1442,
14. Kuan, C.M. and K. Hornik, 1991. Convergence of Learning Algorithms with Constant Learning Rates, I.E.E.E Transactions on Neural Networks, 2, pp: 484-489, 1991.
15. Jacobs, R.A., 1988. Increased rates of convergence through learning rate adaptation, Neural Networks, Vol.1, pp. 295-307,
16. Oh, S.H., 1997.Improving the error backpropagation algorithm with a modified error function, IEEE, Trans. Neural Networks, 8 pp: 799-803,
17. Cavalieri, S. and O. Mirabella, 1999. A novel learning algorithm which improves the partial fault tolerance of multilayer neural networks, Neural Networks, 12, pp: .91-106, 1999.
18. Brent, R.P., 1991. Fast Training Algorithms for Multilayer Neural Nets, IEEE, Transactions on Neural Networks, 2, pp: 346-354,
19. Hagan, M.T. and M. Menhaj, 1994. Training feedforward networks with Marquardt algorithm, IEEE, Transactions on Neural Networks, 5, pp: 989-993,
20. Sha, D. and V.B. Baji c̄, 1999. On-line Variable Learning Rate BP Algorithm for Multilayer Feedforward Neural Networks, International Conference on Artifitial intelligence (ICAI'99), Durban, South Africa.
21. Sha, D. and V.B. Baji c̄ , 1992. An on-line hybrid learning algorithm for multiplayer perceptron in identification problems. Computers and Electrical Engineering 587: 598.
22. Guersi, N., M. Djeghaba, D. Lefebvre, F. Druaux and E. Leclercq, 2004. Adaptive On-line ANN Learning Algorithm for modelling and control of non-linear systems. MMAR 10 th IEEE, International Conference on Methods and Models in Automation and Robotics 30/08/2004-2/09/2004, Miedzyzdroje, Poland 2004 .
23. Campos, J., F. L. Lewis and R. Selmic, 2000. Backlash Compensation in discrete Time Nonlinear Systems Using Dynamic Inversion by Neural Networks, IEEE, Conference on Robotics and Automation, ICRA,
24. Cybenko,G., 1989. Approximation by superpositions of a sigmoidal function, Math. Control Signals Syst., 2, pp: 303-314,
25. Funahashi, K., 1989. On the approximate realization of continuous mappings by neural networks, Neural Networks, 2, pp: 183-192,
26. Hornik, K., M. Stinchombe, and H. White, 1989. Multilayer feedforward networks are universal approximators, Neural Networks, 2, pp: 359-366,