Neural and Adaptive Control of a Rigid Link Manipulator

DORIN POPESCU, DAN SELISTEANU, LIVIA POPESCU
Faculty of Automation, Computers & Electronics
University of Craiova
107 Decebal Street, 200440
ROMANIA
dorinp@robotics.ucv.ro

Abstract: - In this paper a comparison of classical, adaptive and neural control strategies for a robotic manipulator with two revolute joints is presented. The conventional computed-torque method is presented, as a starting point for the design of the adaptive and neural control techniques. Two adaptive control strategies, a non-model based neural control strategy and a model based neural control strategy are implemented. Computer simulations are performed for the control of a rigid manipulator with two revolute joints, in order to verify the performances of the control strategies and to make some useful comparisons. If a classical controller already controls a robot the advantage of proposed structure is that extension to a model based neural controller for performances improvement is easy.

Key-Words: - robotic manipulator, neural control, computed torque control, adaptive control, trajectory tracking.

1. Introduction

The present paper is addressed to robotic manipulators control. Rigid robot systems are subjects of the research in a both robotic and control fields. The reported research leads to a variety of control methods for rigid robot systems [3]. The present paper is addressed to a robotic manipulator control. High speed and high precision trajectory tracking are frequently requirements for applications of robotic arms.

Conventional controllers for robotic structures are based on independent control schemes in which each joint is controlled separately ([3], [6]) by a simple servo loop. This classical control scheme (for example a PD control) is inadequate for precise trajectory tracking.

The imposed performances for industrial applications require the consideration of the complete dynamics of the robotic manipulator. Furthermore, in real-time applications, the ignoring parts of the robot dynamics or errors in the parameters of the robotic manipulator may cause the inefficiency of this classical control (such as PD controller).

An alternative solution to PD control is the computed torque technique. This classical method is in fact a nonlinear technique that takes account of the dynamic coupling between the robot links. The main disadvantage of this structure is the assumption of an exactly known dynamic model. However, the basic idea of this method remains important and it is the base of the neural and adaptive control structures ([1], [2], [6], [7], [18], [19]).

Even in well-structured industrial applications, robotic arms are subject of the structured uncertainty, i.e. the parameter uncertainty due to unknown load, friction coefficients and so on. When the dynamic model of the system is not known a priori or is not available, a control law is erected based on an estimated model. This is the basic idea behind adaptive control strategies [6].

Over the last few years several authors ([5], [7], [8], [10], [20], [21], [22]) have considered the use of neural networks within a control system for robotic arms.

The organization of this paper is the following. In section 2 the classical control strategies are presented. Subsection 2.1 deals with the computed torque method based on the so-called inverse dynamics of the robotic manipulator. Subsection 2.2 deals with adaptive control method. In section 3 various neural control schemes have been studied, proposed and compared. Model based neural control structures are implemented. The artificial neural network is used to generate auxiliary joint control torque to compensate for the uncertainties in the computed torque based primary robotic manipulator. Subsection 3.1 deals with the feedforward neural control, subsection 3.2 deals with feedback neural control and subsection 3.3 deals with feedback error based neural control. The section 4 is dedicated to the nonlinear model of a

planar robotic manipulator with two revolute joints and to the computer simulation and comparisons. Finally, the section 5 collects the conclusions.

2. Control Strategies

The robotic manipulator is modeled as a set of n rigid bodies connected in series with one end fixed to the ground and the other end free. The bodies are connected via either revolute or prismatic joints and a torque actuator acts at each joint.
The dynamic equation of an *n*-link robotic manipulator is given by ([3], [9]):

$$T = J(q)\ddot{q} + V(q,\dot{q})\dot{q} + G(q) + F(\dot{q}) \qquad (1)$$

where:

- T is an ($n\times1$) vector of joint torques;
- $J(q)$ is the ($n\times n$) manipulator inertia matrix;
- $V(q,\dot{q})$ is an ($n\times n$) matrix representing centrifugal and Coriolis effects;
- $G(q)$ is an ($n\times1$) vector representing gravity;
- $F(\dot{q})$ is an ($n\times1$) vector representing friction forces;
- $q,\dot{q},\ddot{q}$ are the ($n\times1$) vectors of joint positions, speed and accelerations, respectively.

The equations (1) form a set of coupled nonlinear ordinary differential equations which are quite complex, even for simple robotic arms.
For simplicity, we denote

$$V(q,\dot{q})\dot{q} + G(q) + F(\dot{q}) = H(q,\dot{q})$$

so that (1) can be rewritten as:

$$T = J(q)\ddot{q} + H(q,\dot{q}) \qquad (2)$$

2.1. Computed Torque Control

The computed-torque method is a conventional control technique, which takes account of the dynamic coupling between the manipulator links. This method, also called the inverse model control technique [1], [10] leads to a completely decoupled error dynamics equation. The structure of this control strategy is illustrated in Fig. 1.
A used computed-torque control scheme is based on the exactly linearization of the nonlinear dynamics of the robotic manipulator. If the dynamic model is exact, the dynamic perturbations are exactly cancelled. The total torque driving the robotic manipulator is given by ([1]):

$$T = \hat{J}(q)T' + \hat{V}(q,\dot{q})\dot{q} + \hat{G}(q) + \hat{F}(\dot{q}) =$$
$$= \hat{J}(q)T' + \hat{H}(q,\dot{q}) \qquad (3)$$

where: $\hat{J},\hat{V},\hat{G},\hat{F},\hat{H}$ are estimates of *J, V, G, F, H,* respectively, and T' is defined as:

$$T' = \ddot{q}_d + K_V\dot{e} + K_P e \qquad (4)$$

The closed loop equation is found to be:

$$\ddot{e} + K_V\dot{e} + K_P e =$$
$$= \hat{J}^{-1}(q)\left[\tilde{J}(q)\ddot{q} + \tilde{V}(q,\dot{q})\dot{q} + \tilde{G}(q) + \tilde{F}(\dot{q})\right] =$$
$$= \hat{J}^{-1}(q)\left[\tilde{J}(q)\ddot{q} + \tilde{H}(q,\dot{q})\right] \qquad (5)$$

where $\tilde{J} = J - \hat{J};\ \ \tilde{V} = V - \hat{V};\ \ \tilde{G} = G - \hat{G};$ $\tilde{F} = F - \hat{F};\ \ \tilde{H} = H - \hat{H}$ are the modelling errors and the tracking error is $e = q_d - q$.

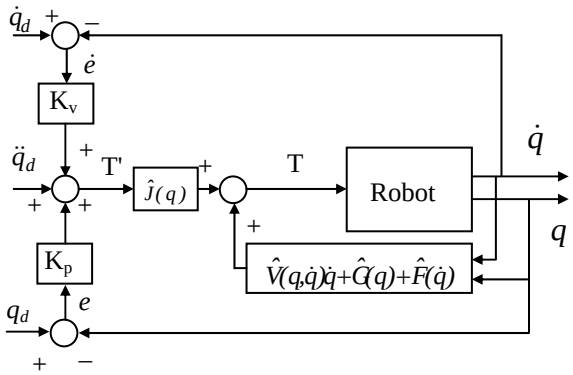

Fig. 1

If the robotic manipulator's parameters are perfectly known, the closed loop equation (5) takes a linear, decoupled form:

$$\ddot{e} + K_v\dot{e} + K_p e = 0 \qquad (6)$$

The computed-torque control method has performance problems because of its reliance on a fixed dynamic model. The robotic arm structures have to face uncertainty in the dynamics parameters. Two classes of approach have been studied to maintain performances in the presence of parametric uncertainties - the robust control and the adaptive control. The next section deals with the adaptive control strategy for the robotic manipulator.

2.2. Adaptive Control

Adaptive controllers can be a good alternative when it is neither possible nor economical to make a thorough investigation of the causes of the process variations. In other situations, some of the dynamics may be well understood, but other parts are unknown. This is the case of robots, for which the geometry, motors and gearboxes do not change, but the load does change. An adaptive controller can be defined as a controller with adjustable parameters and a mechanism for adjusting the parameters [11]. The modern adaptive control approach consists in the explicit introduction of the linear parameterization of the robot dynamics. The adaptive controllers can be classified into three major categories [10]: direct, indirect and composite.

The direct adaptive controllers use tracking errors of the joint motion to drive parameter adaptation. The main goal of the control strategy is to reduce the tracking errors. Such a direct technique is an adaptive control method based on computed torque control. This method has been pioneered by Craig et al. [12], and the properties of stability and convergence are established in [6], [15]. The controller is in fact composed of a modified computed-torque control and an adaptation law.

Next, this direct adaptive strategy is used for the robot arm structure (1). Let's consider θ the vector of the uncertain (unknown) parameters, which are the viscous friction coefficients, the Coulomb friction coefficients and the load mass. Then, the dynamics of the robot arm can be written as:

$$T = J(q,\theta)\ddot{q} + V(q,\dot{q},\theta)\dot{q} + G(q,\theta) + F(\dot{q},\theta) \qquad (7)$$

A linear parameterization of (7) is:

$$\begin{aligned} &J(q,\theta)\ddot{q} + V(q,\dot{q},\theta)\dot{q} + G(q,\theta) + F(\dot{q},\theta) = \\ &= J_C(q)\ddot{q} + V_C(q,\dot{q})\dot{q} + G_C(q) + F_C(\dot{q}) + R(q,\dot{q},\ddot{q})\theta \end{aligned} \qquad (8)$$

where $J_C(.), V_C(.), G_C(.), F_C(.)$ represent the known (certain) part of the dynamics and $R(q,\dot{q},\ddot{q})$ is the regressor matrix.

The design of the control law is reached by using in (7) the vector of the estimated parameters. The linearization (8) allows us to obtain the torque:

$$\begin{aligned} &T = J(q,\hat{\theta})\ddot{q} + V(q,\dot{q},\hat{\theta})\dot{q} + G(q,\hat{\theta}) + F(\dot{q},\hat{\theta}) = \\ &= J_C(q)\ddot{q} + V_C(q,\dot{q})\dot{q} + G_C(q) + F_C(\dot{q}) + R(q,\dot{q},\ddot{q})\hat{\theta} \end{aligned} \qquad (9)$$

where $\hat{\theta}$ is the vector of estimated parameters.

From the equations (4), (9) the closed loop dynamics is obtained:

$$J(q,\hat{\theta})(\ddot{e} + K_v\dot{e} + K_p e) = R(q,\dot{q},\ddot{q})\tilde{\theta} \qquad (10)$$

with $\tilde{\theta} \overset{\Delta}{=} \hat{\theta} - \theta$ the estimation parameter error vector. If the inertia matrix is nonsingular, we can write:

$$\ddot{e} + K_v\dot{e} + K_p e = J^{-1}(q,\hat{\theta})R(q,\dot{q},\ddot{q})\tilde{\theta} \qquad (11)$$

The state representation of (11) can be obtained if the state $x \overset{\Delta}{=} [e \;\; \dot{e}]^T$ is used:

$$\dot{x} = A_m x + B_m J^{-1}(q,\hat{\theta})R(q,\dot{q},\ddot{q})\tilde{\theta} \qquad (12)$$

where $A_m = \begin{bmatrix} 0 & I \\ -K_p & -K_v \end{bmatrix}$, $B_m = \begin{bmatrix} 0 \\ I \end{bmatrix}$.

We can choose a gradient type adaptation law for the on-line estimation of the parameters:

$$\frac{d\tilde{\theta}(t)}{dt} = \frac{d\hat{\theta}(t)}{dt} = -\Omega \cdot R^T(q,\dot{q},\ddot{q})J^{-1}(q,\hat{\theta}) \cdot B_m^T P x \qquad (13)$$

with $\Omega = \Omega^T > 0$ the amplification matrix and $P = P^T > 0$ is a quadratic $n \times n$ matrix, solution of the Lyapunov equation:

$$A_m^T P + P A_m = -Q \qquad (14)$$

where $Q = Q^T > 0$.

Remark: The Lyapunov function $V = x^T P x + \tilde{\theta}^T \tilde{\theta}$ can be used to show that the tracking errors go to zero.

The final adaptive control law consists of the computed-torque Eq. (3) and the estimates provided by the adaptation law (13).

The global convergence of the direct adaptive controller based on computed-torque method is demonstrated in [15]. The disadvantages of this adaptive method are the use of the acceleration measurements and the necessity of inversion of the estimated inertia matrix. The advantages are the simplicity of the method (comparatively to a least squares indirect method for example) and the rejection of the parametric disturbances, inherent for an adaptive method.

The indirect adaptive control method for manipulators has been pioneered by Middleton and Goodwin [4], who used prediction errors on the

filtered joint torques to generate parameter estimates to be used in the control law.
Such indirect adaptive controller can be composed of a modified computed-torque control and a modified least-squares estimator. The design of this indirect control law for the manipulator (1) is based on the estimate of the torque:

$$\hat{T} = J_C(q)\ddot{q} + V_C(q,\dot{q})\dot{q} + G_C(q) + F_C(\dot{q}) + R(q,\dot{q},\ddot{q})\hat{\theta} \quad (15)$$

where $\hat{\theta}$ is the vector of estimated parameters.

Now we can calculate the prediction error for the torque from (8), (15)

$$\varepsilon \stackrel{\Delta}{=} \hat{T} - T = R(q,\dot{q},\ddot{q})\cdot(\hat{\theta}-\theta) = R(q,\dot{q},\ddot{q})\tilde{\theta} \quad (16)$$

with $\tilde{\theta} \stackrel{\Delta}{=} \hat{\theta} - \theta$ the estimation parameter error vector.

The prediction error is filtered to eliminate the measurements of the accelerations in the control law. First, the torque T is filtered through a first-degree filter with the transfer function $H(s) = \frac{\omega_f}{s+\omega_f}$, where ω_f is the crossover frequency of the filter. The filtered torque is the convolution

$$T_f = h(t) * T(t) \quad (17)$$

where *h(t)* is the impulse response of *H(s)*.

The estimated torque is also filtered. We define

$$T_C = J_C(q)\ddot{q} + V_C(q,\dot{q})\dot{q} + G_C(q) + F_C(\dot{q}) \quad (18)$$

and from (15), (18) the estimated torque can be written as

$$\hat{T} = T_C + R(q,\dot{q},\ddot{q})\hat{\theta} \quad (19)$$

We have

$$T_{Cf}(t) = h(t) * T_C(t) \quad (20)$$

$$\Phi(t) = h(t) * R(t) \quad (21)$$

In the relations (20), (21), T_{Cf} and the filtered regressor matrix Φ depend only of the state *q(t)* and of the time derivative $\dot{q}(t)$, and not of the accelerations [3].

$$T_{Cf}(t) = T_{Cf}(q(t),\dot{q}(t)); \quad \Phi(t) = \Phi(q(t),\dot{q}(t))$$

We obtain the filtered estimated torque from (19), (20), (21)

$$\hat{T}_f = T_{Cf} + \Phi\cdot\hat{\theta} \quad (22)$$

Now we can obtain the filtered prediction error, which will be used in the adaptation law. From (16), (17), (22) the filtered prediction error is

$$\varepsilon_f = \hat{T}_f - T_f = T_{Cf}(t) + \Phi(t)\cdot\hat{\theta} - h(t) * T(t) \quad (23)$$

The torque T can be written as

$$T = T_C + R\cdot\theta$$

therefore the filtered prediction error becomes

$$\begin{aligned}\varepsilon_f &= h(t) * T_C(t) + \Phi(t)\cdot\hat{\theta} - h(t) * T_C(t) - \Phi(t)\cdot\theta \\ &= \Phi(q(t),\dot{q}(t))\cdot\tilde{\theta}\end{aligned} \quad (24)$$

The adaptation parameter law is based on a least-squares estimator [1] that it has as input the filtered prediction error (24). The equations of the adaptation law are

$$\frac{d\tilde{\theta}(t)}{dt} = \frac{d\hat{\theta}(t)}{dt} = -\Gamma(t)\Phi^T(q,\dot{q})\varepsilon_f(t) \quad (25)$$

$$\frac{d\Gamma(t)}{dt} = -\Gamma(t)\Phi^T(q,\dot{q})\Phi(q,\dot{q})\Gamma^T(t) \quad (26)$$

with $\Gamma(0) = \Gamma^T(0) > 0$. The matrix $\Gamma(t) = \Gamma^T(t) > 0$ is the amplification matrix.
The final indirect adaptive control law consists of the computed-torque equation (3) and the estimates provided by the adaptation law (25), (26):

$$\begin{aligned}T &= \hat{J}(q)T' + \hat{V}(q,\dot{q})\dot{q} + \hat{G}(q) + \hat{F}(\dot{q}) \\ &= J(q,\hat{\theta})T' + V(q,\dot{q},\hat{\theta})\dot{q} + G(q,\hat{\theta}) + F(\dot{q},\hat{\theta})\end{aligned} \quad (27)$$

with *T'* given by (4).
The indirect adaptive control structure is presented in Fig. 2.

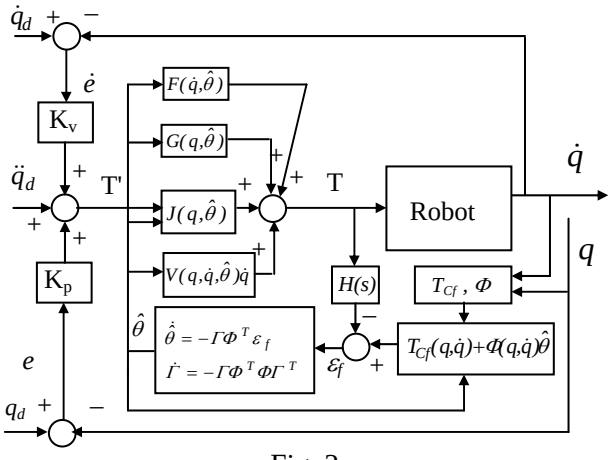

Fig. 2.

The least-squares estimator (25), (26) has good convergence and stability properties [3]. A disadvantage can be the complexity of the algorithm and the correlation between the prediction error and the estimation parameter error [1], [3]. This disadvantage can be canceled by addition of a stabilizing signal to the control law [1].

Remark: Indirect controllers allow the various parameter-estimation algorithms to be used to select time variations of the adaptation gains.

Composite adaptive controllers for manipulators have been developed by Slotine and Li [16]. These adaptive control strategies use both tracking errors in the joint motions and prediction errors on the filtered torque to drive the parameter adaptation.

3. Neural Control

Various neural control schemes have been studied, proposed and compared. The differences in these schemes are in the role that artificial neural network (ANN) is playing in the control system and the way it is trained for achieving desired trajectory tracking performance [2], [10], [14], [20], [21], [22].

Two classes of approach have been studied: non-model based neural control and model based neural control. Non-model based neural control consists of PD feedback controller and an ANN. The inverse dynamics is learned by measuring the input and output signals in the manipulator and then adjusting the connection weights vector by using a learning algorithm. After the learning was finished, the actual trajectory of the manipulator followed well the desired trajectory. But, when the desired trajectory was changed to one not used in the training of ANN, the error between the actual and desired trajectory became large. This means that the ANN had fitted a relationship between the input/output data but had no succeeded in learning the inverse-dynamics model [10].

We want that training doesn't depend on desired trajectory. Hence, we proposed to train the ANN with $(q, \dot{q}, \ddot{q}_d, e, \dot{e})$ (fig. 3).

For training of ANN, there are two possibilities: off-line or on-line. From the viewpoint of real time control it's better to train ANN on-line. But, from the viewpoint of initial weights and biases, rate of convergence and stability of learning it's better to train ANN off-line. The tracking performance was better if ANN was trained off-line and then ANN was used to improve the performance of PD feedback controller [13].

In this section, model based neural control structures for a robotic manipulator is implemented. Various neural control schemes have been studied, proposed and compared. The differences in these schemes are in the role that artificial neural network (ANN) is playing in the control system and the way it is trained for achieving desired trajectory tracking performance.

The most popular control scheme is one which uses ANN to generate auxiliary joint control torque to compensate for the uncertainties in the computed torque based primary robotic manipulator controller that is designed based on a nominal robotic manipulator dynamic model. This is accomplished by implementing the neural controller in either a feedforward or a feedback configuration, and the ANN is trained on-line.

Based on the computed torque method, a training

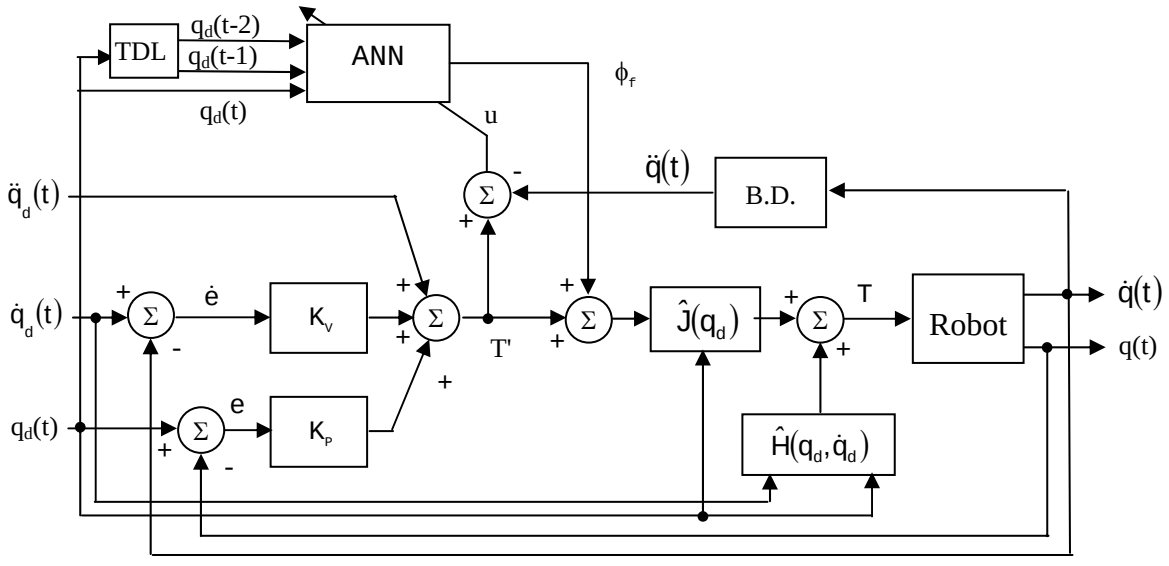

Fig. 3

signal is derived for neural controller. Comparison studies based on a robotic planar manipulator have been made for the neural controller implemented in both feedforward and feedback configurations.
A feedback error based neural controller is proposed. In this approach, a feedback error function is minimized and the advantage over Jacobian based approach is that Jacobian estimation is not required.

3.1. Feedforward Neural Controller

The feedforward neural controller (Fig. 3) is designed to achieve perturbation rejection for a computed torque control system of a robotic manipulator.
The ANN output cancels out the uncertainties caused by inaccurate robotic manipulator's model in the computed torque controller. The robot joint torques are:

$$T = \hat{J}(q_d)\left(T' + \phi_f\right) + \hat{H}(q_d, \dot{q}_d) \qquad (28)$$

The closed loop error system is

$$\ddot{e} + K_V \dot{e} + K_P e = \hat{J}^{-1}\left(\tilde{J}\ddot{q} + \tilde{H}\right) - \phi_f \qquad (29)$$

Since the control objective is to generate ϕ_f to reduce u to zero, we therefore propose to use u:

$$u = \ddot{e} + K_V \dot{e} + K_P e \qquad (30)$$

as the error signal for training the ANN. The ideal value of ϕ_f at $u = 0$ then is:

$$\phi_f = \hat{J}^{-1}\left(\tilde{J}\ddot{q} + \tilde{H}\right) \qquad (31)$$

Minimizing the error signal u allows achieving ideal computed torque control directly.

3.2. Feedback Neural Controller

The main difference between feedforward and feedback neural controller schemes is that the joint variables used in the ANN inputs and the computed torque controller are either the desired values $q_d(t)$ or the actual values $q(t)$. The ANN inputs can be either $q_d(t)$, $\dot{q}_d(t), \ddot{q}_d(t)$ or $q(t)$, $\dot{q}(t), \ddot{q}(t)$, or the time-delayed values $q_d(t)$, $q_d(t\text{-}1)$, $q_d(t\text{-}2)$ or $q(t)$, $q(t\text{-}1)$, $q(t\text{-}2)$. Delay time is chosen as the sampling period of the controller. In simulations the ANN performs better when time-delayed joint values are used instead of the velocity and acceleration values calculated from finite difference approximations based on samples of $q(t)$.
For feedback neural controller (Fig. 4) the robotic manipulator joint torques are:

$$T = \hat{J}(q)\left(T' + \phi_b\right) + \hat{H}(q, \dot{q}) \qquad (32)$$

The three-layer feedforward neural network is used as the compensator. It is composed of an input layer (6 neurons), a nonlinear hidden layer, and a linear output layer (2 neurons).
The weight updating law minimizes the objective function J which is a quadratic function of the training signal u:

$$\mathrm{J} = \frac{1}{2}\left(u^T u\right) \qquad (33)$$

For simplicity, we use ϕ for ϕ_f or ϕ_b. Differentiating equation (33) and making use of (29) yields the

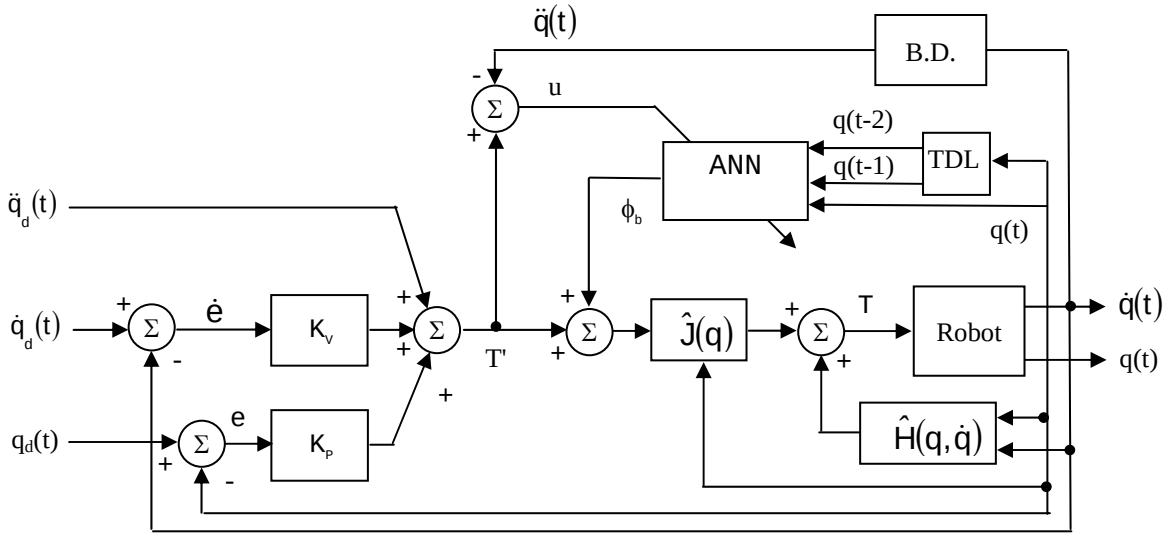

Fig. 4

gradient of J as follows:

$$\frac{\partial \mathrm{J}}{\partial w}=\frac{\partial u^T}{\partial w}u=-\frac{\partial \phi^T}{\partial w}u \tag{34}$$

The backpropagation update rule for the weights with a momentum term is:

$$\Delta w(t)=-\eta\frac{\partial \mathrm{J}}{\partial w}+\alpha\Delta w(t-1)=\eta\frac{\partial \phi^T}{\partial w}u+\alpha\Delta w(t-1) \tag{35}$$

where η is learning rate and α is the momentum coefficient.

3.3. Feedback Error Based Neural Controller

In this approach, a feedback error function is minimized and the advantage over Jacobian based approach is that Jacobian estimation is not required. The inputs to the neural controller (Fig. 5) are the required trajectories $q_d(t)$, $\dot{q}_d(t)$, $\ddot{q}_d(t)$. The compensating signals from ANN, ϕ_p, ϕ_v, ϕ_a, are added to the desired trajectories. The control law is:

$$T=\hat{J}\left(\ddot{q}_d+\phi_a+K_V\left(\dot{e}+\phi_v\right)+K_P\left(e+\phi_p\right)\right)+\hat{H} \tag{36}$$

Combining (36) with dynamic equation of robotic manipulator yields:

$$u=\ddot{e}+K_V\dot{e}+K_Pe=\hat{J}^{-1}\left(\tilde{J}\ddot{q}+\tilde{H}\right)-\Phi \tag{37}$$

where $\Phi=\phi_a+K_V\phi_v+K_P\phi_p$. Ideally, at $u = 0$, the ideal value of Φ is:

$$\Phi=\hat{J}^{-1}\left(\tilde{J}\ddot{q}+\tilde{H}\right) \tag{38}$$

The error function u is minimized and the objective function is the same (33). The gradient of J is:

$$\frac{\partial \mathrm{J}}{\partial w}=\frac{\partial u^T}{\partial w}u=-\frac{\partial \Phi^T}{\partial w}u \tag{39}$$

The backpropagation updating rule for the weights with momentum term is:

$$\Delta w(t)=-\eta\frac{\partial \mathrm{J}}{\partial w}+\alpha\Delta w(t-1)=\eta\frac{\partial \Phi^T}{\partial w}u+\alpha\Delta w(t-1) \tag{40}$$

4. Simulation Studies

The control of the simple planar robotic manipulator with two revolute joints shown in Fig. 6 will be considered.
The elements of the dynamic equation (1) for this robotic manipulator with electrical motor dynamics are:

$$J(q)=\begin{bmatrix} l_1^2(m_1+m_2)+m_2l_2^2+2m_2l_1l_2c_2+J_1n_1^2 & m_2l_2^2+m_2l_1l_2c_2 \\ m_2l_2^2+m_2l_1l_2c_2 & m_2l_2^2+J_2n_2^2 \end{bmatrix} \tag{41}$$

$$V(q,\dot{q})=m_2l_1l_2s_2\begin{bmatrix} 0 & -(2\dot{q}_1+\dot{q}_2) \\ \dot{q}_1 & 0 \end{bmatrix} \tag{42}$$

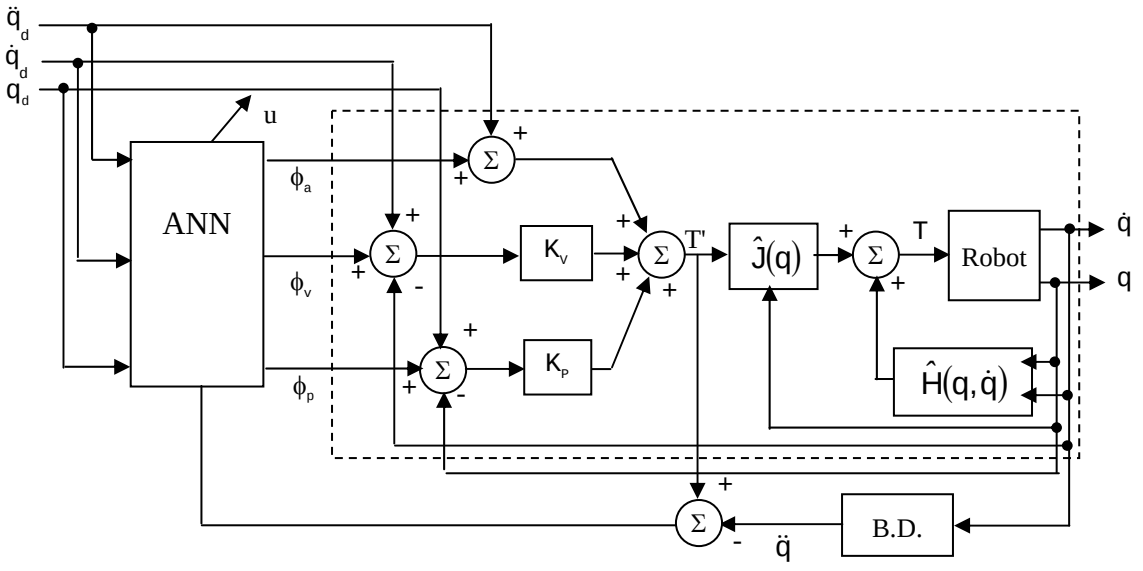

Fig. 5

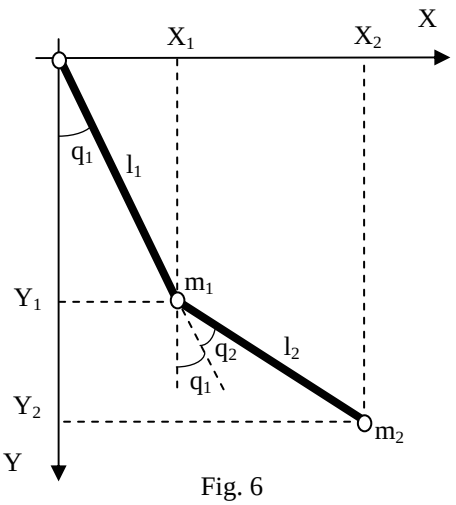

Fig. 6

$$G(q)=\begin{bmatrix}(m_1+m_2)gl_1s_1+m_2gl_2s_{12}\\ m_2gl_2s_{12}\end{bmatrix} \qquad (43)$$

$$F(\dot{q})=\begin{bmatrix}v_1\dot{q}_1+C_1sign(\dot{q}_1)\\ v_2\dot{q}_2+C_2sign(\dot{q}_2)\end{bmatrix} \qquad (44)$$

with m_1 = mass of link 1; $m_2 = m_{20} + m_p$;
m_{20} = mass of link 2; m_p = mass of payload;
l_1 = length of link 1; l_2 = length of link 2;
$c_i = cos\ (q_i)$; $s_i = sin\ (q_i)$;
$c_{12} = cos\ (q_1 + q_2)$; $s_{12} = sin\ (q_1 + q_2)$
J_i = moments of inertia for electrical motor *i*.
n_i = factor of reduction gear *i*.
v_i = viscous friction for joint *i*.
C_i = Coulomb friction for joint *i*.

For simulation and comparisons, the planar robot manipulator with two revolute joints (1), (41)-(44) is used. The simulation model parameters are (SI units):

- $m_1 = 10$, m_2 consists of mass of link 2, $m_{20} = 2.5$ and mass of payload, m_p
- $l_1 = 1$, $l_2 = 0.5$, link lengths

The robot manipulator starts at position ($q_1 = 0$, $q_2 = 0$) and the control objective is to track the desired trajectory given by:

$$q_{1d} = 0.4 \cdot sin(0{,}4\pi t)$$

$$q_{2d} = -0.5 \cdot sin(0{,}5\pi t)$$

When the model of the robot manipulator is known, the use of the computed-torque method is recommended. The equations (8), (9) are used and a simulation has been done for the tuning parameters $K_{p1} = 50$, $K_{p2} = 50$, $K_{v1} = 6$, $K_{v2} = 15$. The time evolution of errors in this simulated case (with unknown parameters m_p, v_1, v_2, C_1, C_2) is presented in Fig. 7.

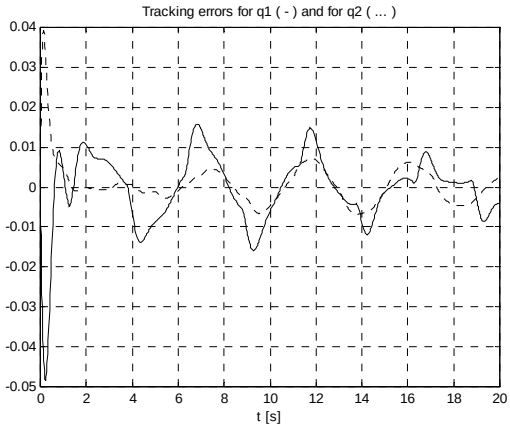

Fig. 7

For feedforward and feedback neural controller (6 x 10 x 2) with update backpropagation rule (35) the tracking errors are presented in Fig. 8 and Fig. 9.

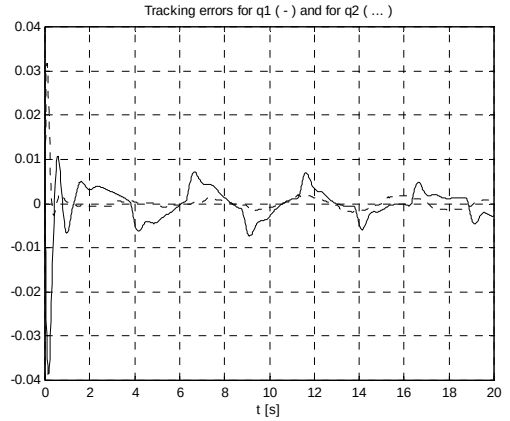

Fig. 8

For feedback error based neural controller (6 x 9 x 2) with update backpropagation rule (40) the tracking errors are presented in Fig. 10.

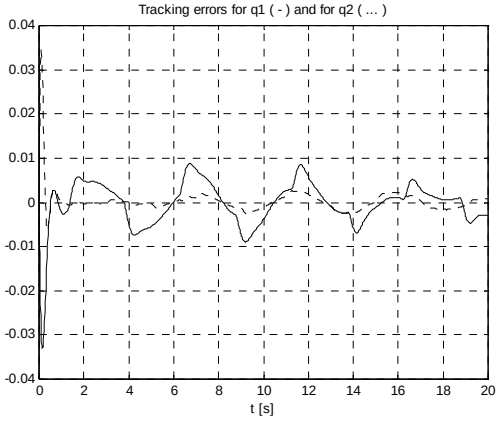

Fig. 9

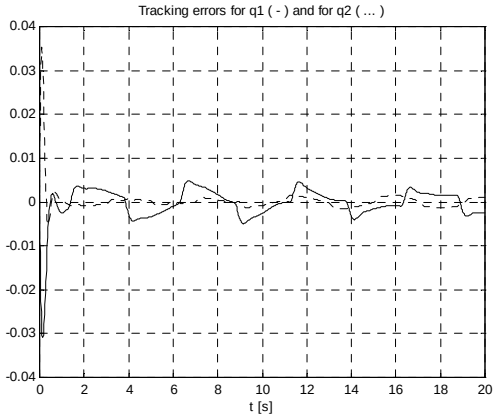

Fig. 10

The comparisons between the control strategies can be done by visualization of tracking errors, but accurate comparisons can be done by considering a criterion based on averaged square tracking errors (see [17]):

$$I_1 = \frac{1}{T}\int_0^T e_1^2(t)dt \qquad (45)$$

$$I_2 = \frac{1}{T}\int_0^T e_2^2(t)dt \qquad (46)$$

where T is the total simulation time.

The obtained results from calculus of I_1 and I_2 for the studied control strategies (manipulator with two revolute joints) are presented in Table 1.

Table 1. Performance Criterion

Control Strategies	Performances	
	I_1	I_2
Classical PD controller (exactly known model and manipulator parameters)	$4.2*10^{-4}$	$3.1*10^{-4}$
Computed-torque method (exactly known model and manipulator parameters)	$1.2*10^{-4}$	$0.8*10^{-4}$
Direct adaptive controller (gradient adaptation law)	$1.7*10^{-4}$	$1.0*10^{-4}$
Indirect adaptive controller (least-squares estimator as adaptation law)	$1.1*10^{-4}$	$0.7*10^{-4}$
Non-model based neural controller	$2*10^{-4}$	$1.1*10^{-4}$
Model based neural controller	$0.9*10^{-4}$	$0.5*10^{-4}$

5. Conclusions

Classical and neural strategies have been applied to the control of a simple planar robotic manipulator with two revolute joints.
The computed-torque method solves the precision tracking problem using an exactly linearization of the nonlinearities of the robotic manipulator model. The main disadvantage is the assumption of an exactly known dynamic model. If the model is imprecise known, a model based neural control law based on the computed torque method is implemented. The simulation results prove that the tracking performances are better.
The simulation showed that the proposed neural controllers obtain results comparable to those achieved using adaptive control strategies.
If a classical controller already controls a robotic manipulator the advantage of proposed structures is that extension to a neural controller for performances improvement is easy.

References:

[1] St. Dumbrava, I. Olah, Robustness analysis of computed torque based robot controllers, *5-th Symposium on Automatic Control and Computer Science*, Iasi, 1997, pp. 228-233.
[2] M.M. Gupta, D.H. Rao, *Neuro-Control Systems*, IEEE Press, 1994.
[3] M. Ivanescu, *Industrial robots*, Ed. Universitaria, Craiova, 1994, pp. 149-186.
[4] R. Middleton, G. Goodwin, Adaptive computed torque control for rigid link manipulators, *Systems and Control Letters*, vol. 10, 1998, pp. 9-16.
[5] H. Miyamoto, M. Kawato, T. Setoyama, R. Suzuki, Feedback error learning neural networks for trajectory control of a robotic manipulator, *Neural Networks*, 1, 1998, pp. 251-265.
[6] R. Ortega, M.W. Spong, Adaptive motion control of rigid robots: a tutorial, *Automatica*, 25, 1999, pp. 877-888.
[7] T. Ozaki, T. Suzuki, T. Furuhashi, Trajectory control of robotic manipulators using neural networks, *IEEE Transaction on Industrial Electronics*, 38, 1991, pp. 641-657.
[8] D.T. Pham, S.J. Oh, Adaptive control of a robot using neural networks, *Robotica*, 1994, pp. 553-561.
[9] D. Popescu, Neural control of manipulators using a supervisory algorithm, *A&Q'98 International Conference on Automation and*

Quality Control, Cluj-Napoca, 1998, pp. A576-A581.

[10] A. Zalzala, A. Morris, *Neural networks for robotic control*, Prentice Hall, 1996, pp 26-63.

[11] K.J. Astrom, B. Wittenmark, *Adaptive Control*, Addison-Wesley, 1995.

[12] J.J. Craig, P. Hsu, S. Sastry, Adaptive control of mechanical manipulators, *Int. J. Robotics Res.*, 2, 1987, pp. 10-20.

[13] D. Popescu, D. Selisteanu, C. Ionete, Non-model based neural robot control, *Proc. of the 10th Int. Workshop on Robotics in Alpe-Adria-Danube Region*, Vienna, 2001, RD-038.

[14] D. Psaltis, A. Sideris, A. Yamamura, A multilayered neural network controller, *IEEE Control Systems Magazine*, 8, 1988, pp. 17-21.

[15] J.J.E. Slotine, W. Li, On the adaptive control of robot manipulators, *Int. J. Robotics Res.*, 3, 1987, pp. 49-59.

[16] J.J.E. Slotine, W. Li, Composite adaptive manipulator control, *Automatica*, 25, 4, 1989, pp. 509-519.

[17] L.L. Whitcomb, A.A. Rizzi, D.E. Kodishek, Comparative Experiments with a New Adaptive Controller for Robot Arms, *Proc. IEEE Conf. on Robotics & Automation*, Sacremento, USA, 1991, pp. 2-7.

[18] E. Petre, D. Selisteanu, D Sendrescu, Nonlinear and Neural Networks Based Adaptive Control for a Class of Dynamical Nonlinear Processes, *WSEAS Transactions on Systems*, Issue 7, Volume 5, July 2006, pp. 1517-1524.

[19] E. Bobasu, D. Popescu, On Modelling and Multivariable Adaptive Control of Robotic Manipulators, *WSEAS Transactions on Systems*, Issue 7, Volume 5, July 2006, pp. 1579-1586.

[20] S. Aoughellanet, T. Mohammedi, Y. Bouterfa, Neural network path planning applied to PUMA560 robot arm, *WSEAS Transactions on Systems*, Issue 4, Volume 4, April 2005, pp. 446-450.

[21] F. M. Raimondi, M. Melluso, V. Bonafede, A neuro fuzzy controller for planar robot manipulators, *WSEAS Transactions on Systems*, Issue 10, Volume 3, December 2004, pp. 2991-2996.

[22] B. H. Dinh, M. W. Dunnigan, D. S. Reay, A Practical Approach for Position Control of a Robotic Manipulator Using a Radial Basis Function Network and a Simple Vision System, *WSEAS Transactions on Systems and Control*, Issue 4, Volume 3, April 2008, pp. 289-298.