Simulated Comparison, Adaptive and Neural Control Strategies for a Robot Arm

Dorin Popescu, Dan Selisteanu, Cosmin Ionete

Faculty of Automation, Computers and Electronics
University of Craiova
Craiova, 5 Lapus Street, 1100, Romania
e-mail: dorinp@robotics.ucv.ro; dansel@automation.ucv.ro; cosmin@automation.ucv.ro

Abstract

In this paper a comparison of classical, adaptive and neural control strategies for a robot arm with two revolute joints is presented. The classical control scheme is based on a conventional PD controller. Another conventional structure, the computed torque scheme, is also presented. Better results are obtained when a PD - neural control strategy is implemented. If parametric uncertainties occur, an adaptive control scheme is used to preserve the performances. The performances of the implemented control strategies for trajectory tracking are analysed by computer simulation. The advantage of a neural control technique for robot arms is that it avoids this a priory-modelling problem. If a PD controller already controls a robot arm the advantage of proposed structure is that extension to a PD - neural controller for performances improvement is easy.

1. Introduction

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 robotic manipulator control. High speed and high precision trajectory tracking are frequently requirements for applications of robot arms.

Conventional controllers for robot arm structures are based on independent control schemes in which each joint is controlled separately ([3], [7]) by a simple servo loop. This classical control scheme is inadequate for precise trajectory tracking. The imposed performances for industrial applications require the consideration of the complete dynamics of the robot arm. Furthermore, in real-time applications, the ignoring parts of the robot dynamics, or errors in the parameters of the robot arm 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], [7], [8]).

Even in well-structured industrial applications, manipulators 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 [7]. Over the last few years several authors ([5], [8], [9], [12]) have considered the use of neural networks within a control system for robot arms.

2. Classical Control Methods

The robot arm 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 robot arm is given by ([3]; [10])

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

where

- T is an $(n \times 1)$ 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 $(n \times 1)$ vector representing gravity;
- $F(\dot{q})$ is an $(n \times 1)$ vector representing friction forces;
- $q, \dot{q}, \ddot{q}$ are the $(n \times 1)$ vectors of joint positions, speed and accelerations, respectively.

The equation (1) form a set of coupled nonlinear ordinary differential equations, which are quite complex, even for simple robot arms.

The control of the simple planar robot arm with two revolute joints shown in Fig. 1 will be considered.

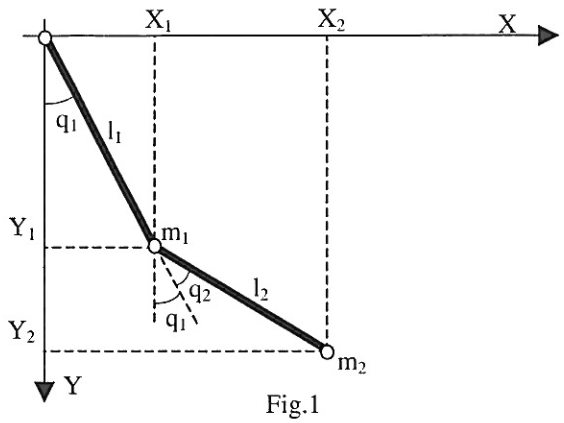

Fig.1

The elements of the dynamic equation (1) for this robot arm with electrical motor dynamics are found to be

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

$$V(q,\dot{q})=m_2l_1l_2s_2\begin{bmatrix} 0 & -(2\dot{q}_1+\dot{q}_2) \\ \dot{q}_1 & 0 \end{bmatrix} \quad (3)$$

$$G(q)=\begin{bmatrix} (m_1+m_2)gl_1s_1+m_2gl_2s_{12} \\ m_2gl_2s_{12} \end{bmatrix} \quad (4)$$

$$F(\dot{q})=\begin{bmatrix} v_1\dot{q}_1+C_1sign(\dot{q}_1) \\ v_2\dot{q}_2+C_2sign(\dot{q}_2) \end{bmatrix} \quad (5)$$

with $c_i = cos\ (q_i)$; $s_i = sin\ (q_i)$;
$c_{12} = cos\ (q_1+ q_2)$; $s_{12} = sin\ (q_1 + q_2)$
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.

The conventional PD controller consists of compensation of the gravity effects and of the classical state feedback. The equation of the PD control law is [3]

$$T = G(q)+K_pe+K_v\dot{q} \quad (6)$$

where T is the total torque driving the robot arm, K_v and K_p are $(n \times n)$ constant diagonal matrices and the error e is defined as

$$e = q_d - q \quad (7)$$

A used computed-torque control scheme is based on the exactly linearization of the nonlinear dynamics of the robot arm. This control strategy is illustrated in Fig. 2. If the dynamic model is exact, the dynamic perturbations are exactly cancelled. The total torque driving the robot arm is given by [1]

$$T = \hat{J}(q)T'+\hat{V}(q,\dot{q})\dot{q}+\hat{G}(q)+\hat{F}(\dot{q}) \quad (8)$$

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

$$T' = \ddot{q}_d + K_v\dot{e} + K_pe \quad (9)$$

The closed loop equation is found to be

$$\ddot{e}+K_v\dot{e}+K_pe = \hat{J}^{-1}(q)\left[\tilde{J}(q)\ddot{q}+\tilde{V}(q,\dot{q})\dot{q}+\tilde{G}(q)+\tilde{F}(\dot{q})\right] \quad (10)$$

where $\tilde{J} = J - \hat{J}$; $\tilde{V} = V - \hat{V}$; $\tilde{G} = G - \hat{G}$; $\tilde{F} = F - \hat{F}$ are the modelling errors.

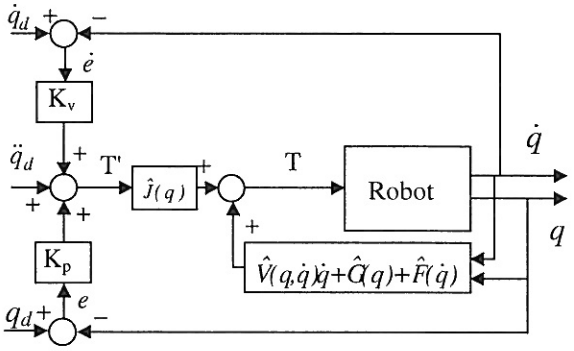

Fig. 2

If the manipulator's parameters are perfectly known, the closed loop equation (10) takes a linear, decoupled form:

$$\ddot{e}+K_v\dot{e}+K_pe = 0 \quad (11)$$

The computed-torque control method has performance problems because of its reliance on a fixed dynamic model. The robot 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 an adaptive control strategy for the robot arm.

3. Adaptive Control Strategy

The adaptive controllers are based on an adaptation mechanism, which keeps extracting parameter information on-line, so these controllers can provide performances in the face of the parametric uncertainties. 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 [12]: direct, indirect and composite.

For the robot arm structure (1) an indirect adaptive controller is proposed in this paper. The method 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. The controller is in fact composed of a modified computed-torque control and a modified least-squares estimator.

Let's consider the uncertain parameters are the viscous friction coefficients, the Coulomb friction coefficients and the load mass. Therefore we have

$$\theta = [m_p \quad v_1 \quad C_1 \quad v_2 \quad C_2]^T \tag{12}$$

where θ is the vector of the uncertain (or unknown) parameters.

The dynamics of the robot arm can be written as

$$T = J(q,\theta)\ddot{q} + V(q,\dot{q},\theta)\dot{q} + G(q,\theta) + F(\dot{q},\theta) \tag{13}$$

A linear parameterization of (13) is

$$\begin{aligned} &J(q,\theta)\ddot{q} + V(q,\dot{q},\theta)\dot{q} + G(q,\theta) + F(\dot{q},\theta) = \\ &= J_C(q)\ddot{q} + V_C(q,\dot{q})\dot{q} + G_C(q) + F_C(\dot{q}) + R(q,\dot{q},\ddot{q})\theta \end{aligned} \tag{14}$$

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 based on the estimate of the torque:

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

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

Now we can calculate the prediction error for the torque from (14), (15):

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

with $\tilde{\theta} \overset{\Delta}{=} \hat{\theta} - \theta$ 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

$$H(s) = \frac{1}{\tau s + 1} = \frac{\omega_f}{s + \omega_f} \tag{17}$$

In (17), $\omega_f = 1/\tau$ is the crossover frequency of the filter.

The filtered torque is the convolution

$$T_f = h(t) * T(t) \tag{18}$$

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

The estimated torque is also filtered. We define

$$T_C = J_C(q)\ddot{q} + V_C(q,\dot{q})\dot{q} + G_C(q) + F_C(\dot{q}) \tag{19}$$

and from (15), (19) the estimated torque can be written as

$$\hat{T} = T_C + R(q,\dot{q},\ddot{q})\hat{\theta} \tag{20}$$

We have:

$$T_{Cf}(t) = h(t) * T_C(t) \tag{21}$$

$$\Phi(t) = h(t) * R(t) \tag{22}$$

In the relations (21), (22), 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 [4].

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

Then we obtain the filtered estimated torque from (20), (21), (22):

$$\hat{T}_f = T_{Cf} + \Phi\cdot\hat{\theta} \tag{23}$$

Now we can obtain the filtered prediction error, which will be used in the adaptation law. From (16), (18), (23) the filtered prediction error is

$$\varepsilon_f = \hat{T}_f - T_f = T_{Cf}(t) + \Phi(t)\cdot\hat{\theta} - h(t) * T(t) \tag{24}$$

The torque T can be written as

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

therefore the filtered prediction error becomes

$$\begin{aligned} \varepsilon_f &= h(t) * T_C(t) + \Phi(t)\cdot\hat{\theta} - h(t) * T_C(t) - \Phi(t)\cdot\theta \\ &= \Phi(q(t),\dot{q}(t))\cdot\tilde{\theta} \end{aligned} \tag{25}$$

The adaptation parameter law is based on a least-squares estimator [3] that it has as input the filtered prediction error (25). The equations of the adaptation law are:

$$\frac{d\tilde{\theta}(t)}{dt} = \frac{d\hat{\theta}(t)}{dt} = -\Gamma(t)\Phi^T(q,\dot{q})\varepsilon_f(t) \tag{26}$$

$$\frac{d\Gamma(t)}{dt} = -\Gamma(t)\Phi^T(q,\dot{q})\Phi(q,\dot{q})\Gamma^T(t) \tag{27}$$

with $\Gamma(0) = \Gamma^T(0) > 0$. The matrix $\Gamma(t) = \Gamma^T(t) > 0$ is the amplification matrix.

The final adaptive control law consists of the computed-torque equation (8) and the estimates provided by the adaptation law (26), (27):

$$\begin{aligned} T &= \hat{J}(q)T' + \hat{V}(q,\dot{q})\dot{q} + \hat{G}(q) + \hat{F}(\dot{q}) \\ &= J(q,\hat{\theta})T' + V(q,\dot{q},\hat{\theta})\dot{q} + G(q,\hat{\theta}) + F(\dot{q},\hat{\theta}) \end{aligned} \tag{28}$$

with T' given by (9).

The adaptive control structure is presented in Fig. 3.

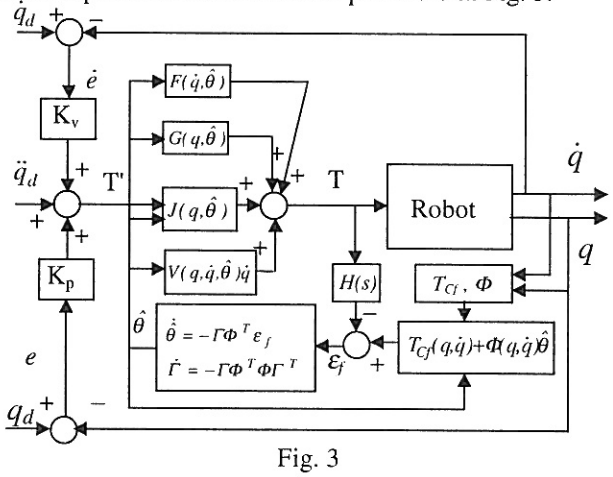

Fig. 3

The least-squares estimator (26), (27) 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 [3]. This disadvantage can be canceled by addition of a stabilizing signal to the control law [1].

4. Neural Controllers

The advantage of a neural control technique for robotic arms is that it avoids this a priory-modelling problem. Although the equations of motion (1) are complex and nonlinear for all robot arms, they have several fundamental properties, which can be exploited to facilitate control system design.

The proposed neural network controller is shown in Fig. 4. The ANN model is used to model the inverse dynamics of each joint for nonlinear compensation of the robot arm ([2], [6], [11]). Control inputs to the joints are composed of both feedback PD control and ANN control. The outputs of the feedback and neural controllers are summed to obtain the total control command applied at the joints. The feedback controller is based on errors between the desired joint states and actual joint states. The ANN output is trained to minimize a quadratic cost function when PD controller is used [12].

The dynamic equation for an n-link rigid robot arm in vectorial form is

$$T = J(q)\ddot{q} + V(q,\dot{q})\dot{q} + G(q) + F(\dot{q}) = J(q)\ddot{q} + Q(q,\dot{q}) \quad (29)$$

Given a desired trajectory defined in terms of joint variables, namely $(q_d, \dot{q}_d, \ddot{q}_d)$ the next step in our control problem is to compute the necessary torques for the joints so that the robot arm follows the desired trajectory. Equation (29) represents the inverse dynamics of a robot arm; that is, given a set of joint variables $(q, \dot{q}, \ddot{q})$, we can obtain the corresponding torque values to be use to drive the actuator.

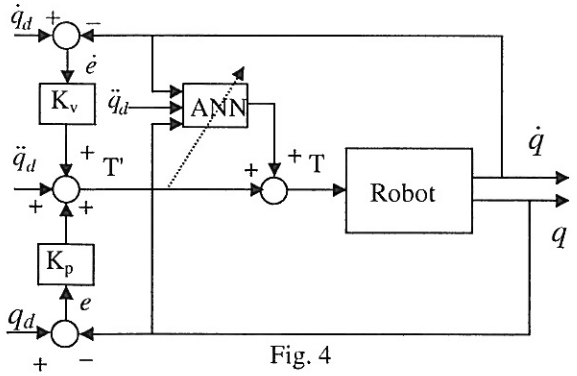

Fig. 4

From equation (29) the robot arm's direct dynamics can be obtained as follows:

$$J(q)\ddot{q} = T - Q(q,\dot{q}) \quad (30)$$

and then

$$\ddot{q} = J^{-1}(q)T - J^{-1}(q)Q(q,\dot{q}) = R(q,\dot{q},T) \quad (31)$$

This equation refers to a nonlinear mapping from the robot input (joint torque T) to the robot output (joint motion). The robot inverse dynamics can be written as

$$T = R^{-1}(q,\dot{q},\ddot{q}) \quad (32)$$

where the transformation R^{-1} is a nonlinear mapping from the joint coordinate space to the joint torque space.

The nonlinear inverse transformation can be decoupled into n less complex transformations, namely:

$$T = R^{-1}(q,\dot{q},\ddot{q}) = \begin{bmatrix} r_1^{-1}(q,\dot{q},\ddot{q}) \\ \cdots \\ r_n^{-1}(q,\dot{q},\ddot{q}) \end{bmatrix} \quad (33)$$

where $r_i^{-1}(q,\dot{q},\ddot{q})$ defines the inverse dynamics transformation of the corresponding joint.

In practice, robot dynamics cannot be modelled exactly. An estimated model $\hat{R}^{-1}$ is used to predict the feedforward torques and servo-feedback is usually included to bring robustness to the overall control scheme. The system dynamics are not time invariant and undergo changes such as variations in payloads, changes in the friction coefficients of the joints etc. Hence, the model estimate $\hat{R}^{-1}$ has to be modified accordingly in order to accommodate these changes. To achieve this, an adaptive or a learning control element is usually associated with the control structure. In the developed control architecture, $\hat{R}^{-1}$ is modelled by artificial neural networks. Each r_i^{-1} can be modelled by a neural network such that

$$T = R^{-1}(q,\dot{q},\ddot{q}) = \begin{bmatrix} r_1^{-1}(q,\dot{q},\ddot{q}) \\ \cdots \\ r_n^{-1}(q,\dot{q},\ddot{q}) \end{bmatrix} = \begin{bmatrix} N_1(q,\dot{q},\ddot{q},W_1) \\ \cdots \\ N_n(q,\dot{q},\ddot{q},W_n) \end{bmatrix} \quad (34)$$

where N_i represents the output of each ANN model used to realize the nonlinear mapping r_i^{-1} and W terms denote the set of adjustable weights of the corresponding ANN.

The inverse dynamics is learned by measuring the input and output signals in the robot and then adjusting the connection weights vector by using a learning algorithm.

5. Computer Simulation and Comparisons

For simulation and comparisons, the planar robot arm with two revolute joints (1)-(5) is used. The simulation model parameters are (SI units):

- $m_1 = 10$, $m_2 = 5$, link masses (at the distal ends of the links)

- $l_1 = 1$, $l_2 = 0.5$, link lengths
- $v_1 = v_2 = 1$, $C_1 = C_2 = 2$.

Mass m_2 consists of mass of link 2, $m_{20} = 2.5$ and mass of payload, $m_p = 2.5$.

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

$$q_{1d} = 0.4 \cdot \sin(0{,}4\pi t); \qquad q_{2d} = -0.5 \cdot \sin(0{,}5\pi t)$$

First, the PD controller (6) is implemented. The PD tuning matrices K_p and K_v are of diagonal form, with K_{p1}=1000, K_{p2}=500, K_{v1}=100, K_{v2}=30. The evolution of tracking errors

$$e = [e_1 \;\; e_2]^T = [q_{1d} - q_1 \;\; q_{2d} - q_2]^T$$

is presented in Fig. 5. We can see that the PD controller works, but the tracking performances are not so good. The PD - neural controller is used when the control structure from Fig. 4 is accomplished, but we used two two-layered neural networks with five inputs $(q_i, \dot{q}_i, \ddot{q}_{di}, e_i, \dot{e}_i)_{i=\overline{1,2}}$, 50 neurons in the hidden layer and one output to learn the robot arm inverse dynamics. After the neural network learned the robot arm's inverse dynamics, it is used in neurocontrol scheme for on-line control of robot arm. The tracking errors in the PD - neural case are shown in Fig. 6. It can be seen that the performances are improved - the errors are smaller compared to the PD control case.

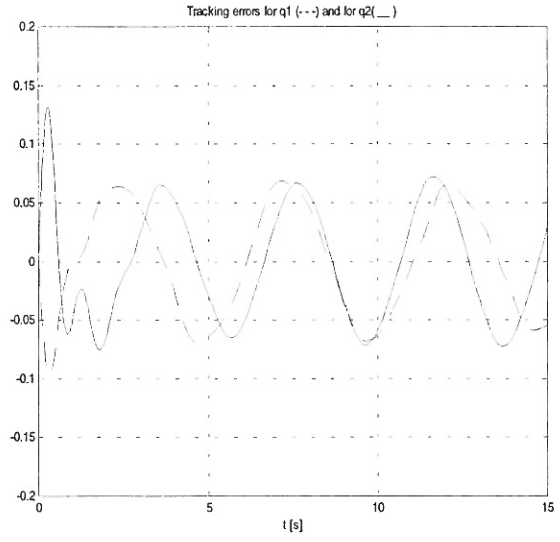

Fig. 5

When the model of the robot arm is well 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 is presented in Fig. 7.

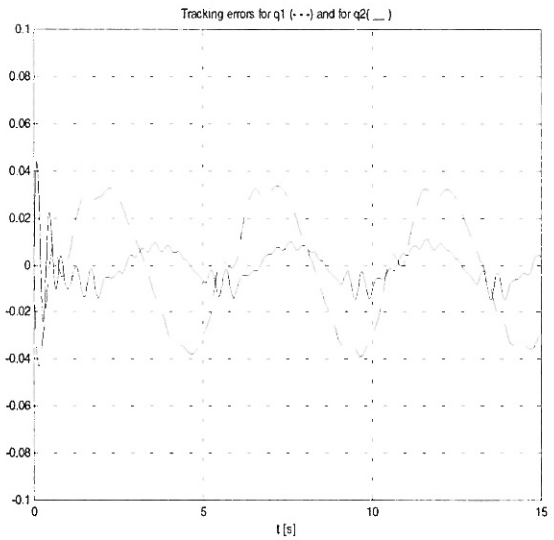

Fig. 6

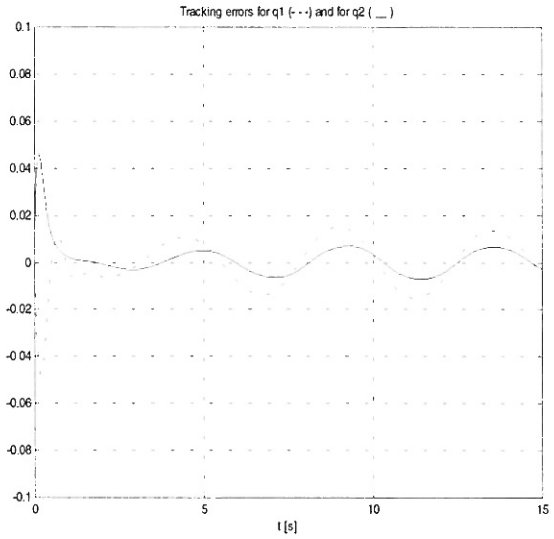

Fig. 7

The computed-torque method can be successfully used when the model is exactly known. If parametric uncertainties occur, an adaptive control method can be used. The adaptive control law (26)-(28) is implemented with the design parameters $K_{p1} = 50$, $K_{p2} = 50$, $K_{v1} = 6$, $K_{v2} = 15$, $\omega_f = 5$ and the diagonal matrix $\Gamma(0) = [\gamma_{ii}]_{i=\overline{1,5}}$, $\gamma_{ii} = 15$. The results are presented in Fig. 8. We can see that even if the estimated parameters $\hat{\theta} = [\hat{m}_p \;\; \hat{v}_1 \;\; \hat{C}_1 \;\; \hat{v}_2 \;\; \hat{C}_2]^T$ are used, the evolution of tracking errors remains good. The profile of the estimated

payload $\hat{m}_p$ is shown in Fig. 9. The estimated parameter has a fast convergence to its actual ("true") value.

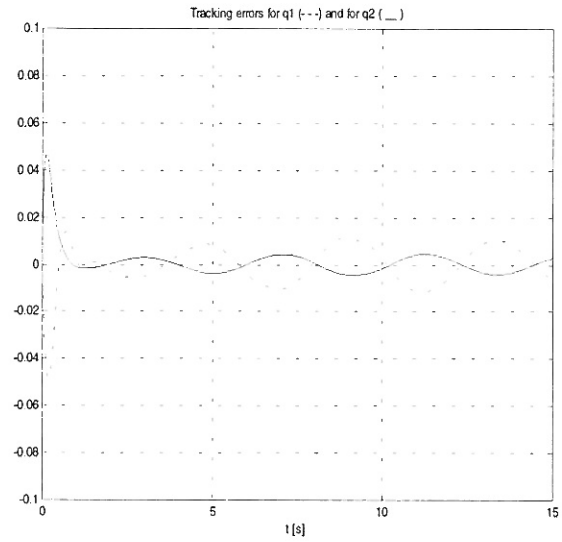

Fig. 8

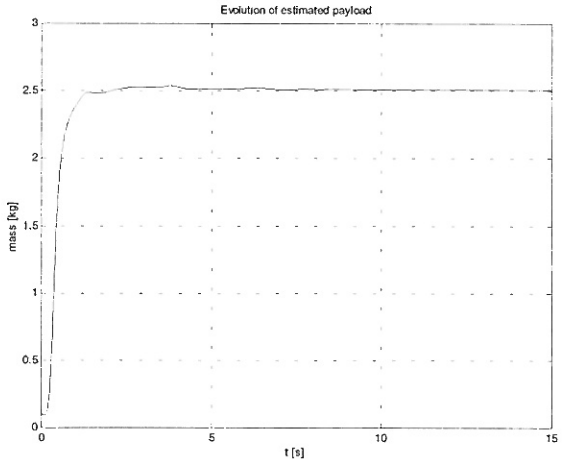

Fig. 9

The imposed trajectories are preserved in the presence of the parametric uncertainties.

6. Conclusions

Classical, adaptive and neural strategies have been applied to the control of a simple planar robot arm with two revolute joints. The results obtained show that the classical PD control can be used for unpretentious applications, but it is inadequate for precise trajectory tracking. The computed-torque method solves the precision tracking problem using an exactly linearization of the nonlinearities of the robot arm model. The main disadvantage is the assumption of an exactly known dynamic model. If the model is imprecise known, an adaptive control law based on the computed torque method, with a least squares estimator as adaptation law, is implemented. The simulation results prove that the tracking performances are preserved. The simulation showed that the proposed neural controller obtains results comparable to those achieved using conventional control strategies. The advantage of a neural control technique for robot arm is that it avoids this a priory-modelling problem. If a PD controller already controls a robot arm the advantage of proposed structure is that extension to a PD-neural controller for performances improvement is easy.

References

[1] Dumbrava St., Olah I., "Robustness analysis of computed torque based robot controllers", *5-th Symposium on Automatic Control and Computer Science*, Iasi, pp. 228-233, 1997.

[2] Gupta M.M., Rao D.H., *Neuro-Control Systems*, IEEE Press, 1994.

[3] Ivanescu M., *Industrial robots*, Ed. Universitaria, Craiova, pp. 149-186, 1994.

[4] Middleton R., Goodwin G., "Adaptive computed torque control for rigid link manipulators", *Systems and Control Letters*, vol. 10, pp. 9-16, 1988.

[5] Miyamoto H., Kawato M., Setoyama T., Suzuki R., "Feedback error learning neural networks for trajectory control of a robotic manipulator", *Neural Networks*, 1, pp. 251-265, 1988.

[6] Narendra K.S., Parthasaranthy K., "Identification and control of dynamical systems using neural networks", *IEEE Trans. on Neural Networks*, 1, pp. 4-27, 1990.

[7] Ortega R., Spong M.W., "Adaptive motion control of rigid robots: a tutorial", *Automatica*, 25,pp.877-888, 1989.

[8] Ozaki T., Suzuki T., Furuhashi T., "Trajectory control of robotic manipulators using neural networks", *IEEE Trans. on Industrial Electronics*, 38, pp. 641-657, 1991.

[9] Pham D.T., Oh S.J., "Adaptive control of a robot using neural networks", *Robotica*, 12, pp. 553-561, 1994.

[10] Popescu D., "Neural control of manipulators using a supervisory algorithm", *A&Q'98 International Conference on Automation and Quality Control*, Cluj-Napoca, pp. A576-A581, 1998.

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

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