A Multilayered Neural Network Adaptive Controller for Robot Manipulators

Badia Amrouche[1], Boualem Kazed[2]

[1] Renewable Energy Development Centre, Algiers, Algeria, amrouche_badia@hotmail.com
[2] Saad Dahlab University, Blida, Algeria, boualem_kazed@yahoo.com

Abstract

The purpose of this work is to show how a fixed gain feedback controller and a set of neural networks can be combined to construct a robust controller that will be insensitive to the payload variations and the model dynamics uncertainties. This controller will be applied to the three main articulations of a PUMA 560 manipulator arm. The role of the neural networks is to compensate the nonlinearities of the manipulator model. The compensation signal, which is a linear combination of the four neural network outputs, is added to a PD controller so that the desired response is obtained.

1. INTRODUCTION

Neural networks present many characteristics and properties [2-6], such as their learning capability, nonlinear mapping and parallel processing. These features make them very useful for modelling, identification, and process control. Neural networks have been studied for use in the field of control, especially for robotic manipulators as a new approach to adaptive control [3] which have proven ineffective [7]. Several architectures were studied and proposed [5] [8] [9][14].

In this article we present the control structure proposed by Carelli et al. [1] applied to the three main degrees of freedom of the PUMA 560 manipulator arm. This structure exploits the neural networks training and generalization properties to compensate for the inverse dynamics of the robotic manipulator. This approach will allow us to obtain an adaptive controller that will be unaffected by the manipulator dynamics' uncertainties and the payload variations for repetitive tasks.

The PUMA 560 robot manipulator has been the object of several modelling studies. To show the feasibility of this controller structure we chose the model proposed by Troch and Desoyer [10].

2. NEURAL NETWORKS ADAPTIVE ROBOT CONTROLLER

The NN controller is expected to compensate the robot nonlinearities by exploiting the neural networks properties. The role of the neural networks is to learn the arm inverse dynamics in the first step. The mathematical notations and theoretical part below are taken from reference [1].

2.1 THE ROBOT NEURAL INVERSE DYNAMICS

The motion equation of a robotic manipulator consisting of a set of m rigid links is given by:

$$\gamma(t) = H(\varphi)\ddot{\varphi} + C(\varphi, \dot{\varphi})\dot{\varphi} + g(\varphi) \qquad (1)$$

Where γ represents the $m \times 1$ vector of joint torques of the manipulator supplied by the actuators. $H(\varphi)$ and $C(\varphi, \dot{\varphi})$ are the $m \times m$ manipulator inertia matrix and the $m \times 1$ vector of the centrifugal and Coriolis terms, respectively. $\varphi, \dot{\varphi}$ and $\ddot{\varphi}$ are, the $m \times 1$ vectors of joint angles, angular velocities, and angular accelerations respectively. $g(\varphi)$ is a $m \times 1$ vector of the gravitational torques.

The dynamic structure of the arm manipulator can be described by the general equation [1][12] :

$$\gamma(t) = W(\varphi, \dot{\varphi}, \ddot{\varphi})\theta \qquad (2)$$

where $W(\varphi, \dot{\varphi}, \ddot{\varphi})$ is an *(m x n)* matrix describing the inverse robot dynamics and actuator drive characteristics. $\theta = (\theta^1, \theta^2, ..., \theta^n)^T$ is an $n \times 1$ vector function of robot parameters. If the function *W* is known, (2) can be used to calculate the joint drives required to follow a desired trajectory, and these estimated drives can be used as feed forward terms in parallel with a feedback controller. The function W is difficult to determine accurately and involves complex computations. It is proved that a four layer (with two hidden layers) perceptron can be used to approximate any continuous function with the desired accuracy [15]. So a NN can be used to approximate the function *W*.

The basic idea is to consider N neural networks, each representing the inverse robot dynamics for a given condition (payload) characterized by a value θ_i.

$$\begin{aligned} \phi_1(x) &= W(x)\theta_1 \\ \phi_2(x) &= W(x)\theta_2 \\ &\vdots \\ \phi_N(x) &= W(x)\theta_N \end{aligned} \qquad (3)$$

with $\phi_i(x)$ an $m \times 1$ vector, and $\theta_i = [\theta_i^1, \theta_i^2, ..., \theta_i^n]^T$ for i=1,2,…,N and $x = [\varphi, \dot{\varphi}, \ddot{\varphi}]^T$.

Consider a particular robot payload condition characterized by a value of the parameter vector θ, and assume that it can be expressed as a linear combination of the values θ_i as follows:

$$\theta = \alpha_1\theta_1 + \alpha_2\theta_2 + ... + \alpha_N\theta_N \qquad (4)$$

then, the inverse robot dynamics for a payload condition characterized by θ can be expressed as :

$$\begin{aligned} \gamma(t) &= w(x)\theta \\ &= w(x)(\alpha_1\theta_1 + \alpha_2\theta_2 + ... + \alpha_N\theta_N) \\ &= \alpha_1 w(x)\theta_1 + \alpha_2 w(x)\theta_2 + ... + \alpha_N w(x)\theta_N \end{aligned} \qquad (5)$$

referring to (3) and (4), the inverse robot dynamics can be approximated to any payload condition by :

$$\gamma(t) = W(x)\theta = \alpha_1\phi_1(x) + \alpha_2\phi_2(x) + ... + \alpha_N\phi_N(x) \qquad (6)$$

Equation (6) can be written as :

$$\theta = \begin{bmatrix} \theta_1^1 & \theta_2^1 & . & . & \theta_N^1 \\ \theta_1^2 & \theta_2^2 & . & . & \theta_N^2 \\ . & . & . & . & . \\ . & . & . & . & . \\ \theta_1^n & \theta_2^n & . & . & \theta_N^n \end{bmatrix} \cdot \begin{bmatrix} \alpha_1 \\ \alpha_2 \\ . \\ . \\ \alpha_N \end{bmatrix} \qquad (7)$$

$$\theta = \Phi\alpha \qquad (8)$$

if *N = n*, and Φ is not singular, the unique solution α for the equation (8) will be given by

$$\alpha = \Phi^{-1}\theta \qquad (9)$$

If the columns of Φ form a basis, a set of parameters α can be found for all θ. That is how the inverse robot dynamics can be approximated by the set of NNs for any value of the vector parameters uncertainties θ, such as the payload [1].

In this case, for the robot parameterisation, we choose the θ components as follows:

$$\theta = (m_L, m_L .x_L, m_L .x_L{}^2, 1) \tag{10}$$

where m_L is the payload mass and x_L its centre position in *x* coordinate of the last linkfixed system.

Figure 1 illustrate the (θ^1, θ^2, θ^3) space assuming admissible values of m_L between 0 Kg and 30 Kg, and values of x_L between 0,05 *m* and 0,2 *m*.

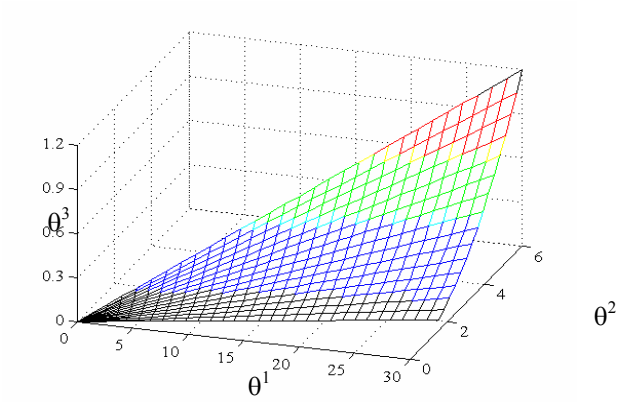

Figure 1. Geometrical surface generated by θ

2.2 THE ADAPTIVE CONTROLLER DESIGN

The adaptive neurocontroller is the sum of two controllers, a PD feedback plus a feed forward compensation given by equations (13) and (14) respectively.

$$\gamma(t) = \gamma_c(t) + \gamma_p(x_d) \tag{12}$$

where $x_d = [\varphi_d, \dot{\varphi}_d, \ddot{\varphi}_d]^T$ is the desired state vector.

$$\gamma_c(t) = -K_p .\delta\varphi - K_v .\delta\dot{\varphi} \tag{13}$$

where $\delta\varphi = \varphi - \varphi_d ; \delta\dot{\varphi} = \dot{\varphi} - \dot{\varphi}_d , K_V , K_p$ positive definite gain matrices.

$$\gamma_P(t) \ = \tilde{\alpha}_1\phi_1(x_d) + \tilde{\alpha}_2\phi_2(x_d) + ... + \tilde{\alpha}_N\phi_N(x_d) \tag{14}$$

The vector $\tilde{\alpha}(t) = [\tilde{\alpha}_1(t), \tilde{\alpha}_2(t), ..., \tilde{\alpha}_N(t)]^T$ is adapted with the following adaptation law [1]:

$$\frac{d}{dt}\tilde{\alpha}(t) = -\Gamma\phi^T(x_d).\gamma_C(t) - K\tilde{\alpha}(t) \tag{15}$$

Γ, K : gain matices, and $\phi(x_d) = [\phi_1(x_d), ..., \phi(x_d)_N]^T$ an (N × m) matrix.

3. SIMULATION RESULTS

The training mode characteristics and the need to follow its evolution led us to program the differential equations which govern the arm dynamical equations and those dealing with the adaptation law using the fifth-order Runge-Kutta method [13].

The next step was to select the four training conditions, values of the θ parameters. This has been done according to the procedure described in [1]. Table 1 summarizes those condition values.

Neural network	ANN 1	ANN 2	ANN 3	ANN 4
Payload (Kg)	0	30	30	30
x_L (m)	0,05	0,2	0,05	0,125

Table 1. NN training conditions

3.1 THE NN TRAINING

To approximate the arm inverse dynamics, we used sigmoidal multi layer perceptrons with two hidden layers of 15 and 9 neurons. Each perceptron has 9 inputs corresponding to the state vector, and 3 neurons in the output layer corresponding to γ_c vector dimension. The NN was trained by using the back propagation algorithm in 2 phases, generalized learning then specialized learning. The structures of training are illustrated by figure (2) [1][7]. The feedback PD gains have the following values: K_{p1}=135, K_{p2}=130, K_{p3}=120, K_{v1}=55, K_{v2}=65, K_{v3}=55.

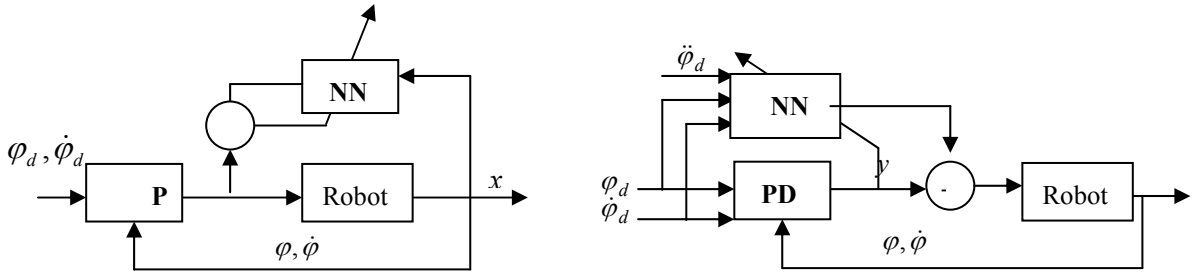

Figure 2. Training structures: generalized learning and specialized learning

At the end of the training phase, the four neural networks are ready. Each one represents the arm inverse dynamics for a particular condition. These models can be used to approximate the inverse arm dynamics for all other condition belonging to the surface represented in figure 1. Estimated inverse dynamics is the weighted sum of the networks outputs. The weighting coefficients of the neuronal outputs are updated following equation 15.

To show the controller effectiveness, we planned to vary the load and the x_L coordinate as illustrated by figure 3. These changes subdivide the simulation period in six different intervals each one corresponds to a particular working condition.

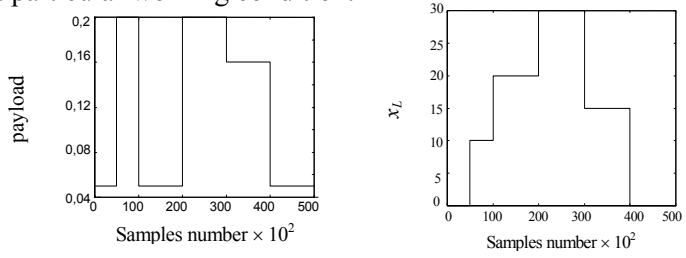

Figure 3. Variable simulation conditions

To fix the suitable gains values achieving the weighting coefficient adaptations, we carried out several control tests to choose values which maintain the tracking errors in the [–0.025, 0.025] radian interval and which reflects the NN contribution in the presence of the arm inverse dynamics variations. The chosen values are summarized in the following table:

Gain	Γ_1	Γ_2	Γ_3	Γ_4	K_1	K_2	K_3	K_4
Value × 10^{-3}	-4	-1	-1	-2	10	8	10	8

Table 2. The weighting coefficient adaptations gains

Figure 4 below shows that the application of the neuronal adaptive controller ensures the maintenance of the tracking errors within acceptable limits during all the simulation period. This is explained by the presence of the four NN which compensate for the change in the arm inverse dynamics for the new working conditions. This compensation is achieved by the weighting coefficient variations represented in figure 5.

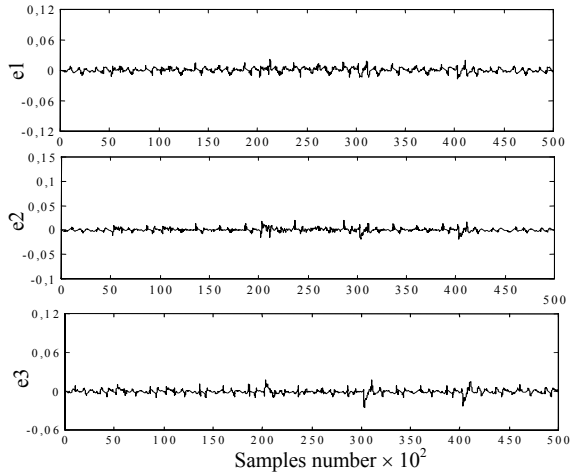

Figure 4. Adaptive controller : tracking errors evolution

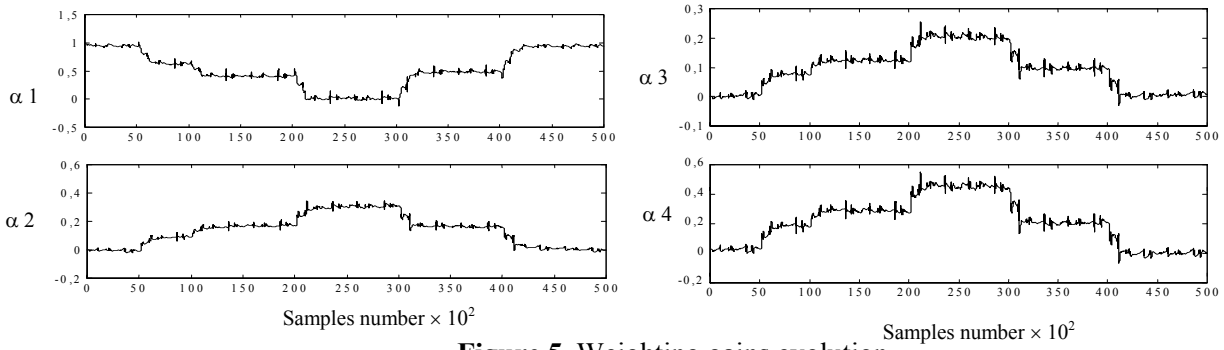

Figure 5. Weighting gains evolution

To show the contribution of this adaptive controller we have repeated the same simulation by fixing the adjustment factors to the values corresponding to the first working condition. We noticed that the tracking errors remain within the limits fixed during the first and the last intervals of simulations only. However the controller performances are clearly degraded during the other intervals, as shown in figure 6.

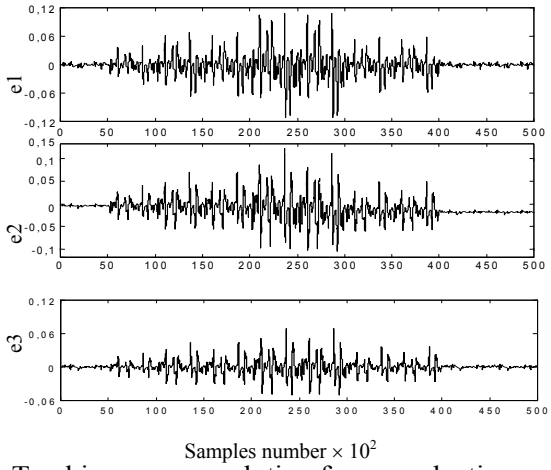

Figure 6. Tracking errors evolution for non adaptive controller

4. CONCLUSION

This work was aimed to extend the results obtained by Carelli et al [1] knowing that we managed to preserve the performances of the controller with a reduced number of neurons and more severe working conditions. The four neural networks used to compensate the changing conditions of the manipulator inverse dynamics were made up with two hidden layers of 15 and 9 neurons respectively. The authors in [1] used layers with 50 and 25 neurons to control two articulations of the manipulator. Moreover, in our simulations the mass of the second segment of the manipulator was reduced from 100 Kg [1] to 50 Kg, this was an attempt to make the controller more sensitive to the payload variations, with all these changes the performances of the controller were preserved even tough the payload variations were in the range of 10 Kg. The neuronal part of the controller described above was used to compensate the arm inverse dynamics whereas the fixed gain feedback controller was to minimize the position and velocity errors.

REFERENCES

[1] Ricardo Carelli, Eduardo F. Camacho, Daniel Patino, "A Neural Network Based Feedforward Adaptive Controller for Robots", IEEE Transactions on Systems, Man and Cybernetics, Vol.25, No.9, September 1995.

[2] Kumpati S. Narendra, Snehasis Mukhopadhyay, "Intelligent Control Using Neural Networks", IEEE Contr. Syst. Mag., Vol. 12, No. 2, pp. 11-18, April 1992.

[3] Kio Ishiguro, Takeshi Furuhashi, Shigero Okuma, "A Neural Network Compensator for Uncertainties of Robotics Manipulators", IEEE Transactions on Industrial Electronics, Vol. 39, No. 6, December 1992.

[4] A. M. S. Zalzala and A. S. Morris, "Neural Networks For Robotic Contrl: Theory and Applications", Ellis Horhood Ltd, 1996.

[5] Stamatios V. Kartalopoulos, "Understanding Neural Networks And Fuzzy Logic; Basic Concepts And Applications", IEEE Press, 1996.

[6] K. Hornik, M. Stinchcombe, and H. White, "Multilayer Feedforward Networks are Universal Approximators", Neural Networks, vol. 2, pp. 359-366, 1989, Pergamon Press Ltd., Oxford, England.

[7] Demetri Psaltis, Athanasios Sideris, and Alan A. Yamamura, "A Multilayered Neural Network Controller", IEEE Control Systems Magazine, April 1988.

[8] Madan. M. Gupta, Dandina H. Rao KHALIL, ‘‘Neuro-Control Systems: Theory and Applications", IEEE PRESS , USA, 1994.

[9] Tetsuro YABUTA, Takayuki YAMADA, "Possibility of Neural Networks Controller for Robot Manipulators", IEEE 1990.

[10] K. Troch and K. Desoyer, "Benchmark problems for control system design", Report of the IFAC Theory Committee, E. J. Davinson, Ed., pp. 45-49, 1990.

[11] Simon HAYKIN, "Neural Networks, A Comprehensive Foundation", ed. by Macmillan College Publishing Company, USA 1994.

[12]Khosla, P. and Kanade, T., "Parameter Identification of Robot Dynamics", Proc. of the 24th IEEE Conference on Decision & Control, pp. 1754-1760, December 1985.
[13] M. Crouzeix, A. L. Mignot, "Analyse numérique des équations différentielles", ed. 2, MASSON 1989.
[14]W. T. Miller, R. P. Hewess, F. H. Glanz and L. G. Kraft, "Real-Time Dynamic Control of an Industrial Manipulator Using a Neural-Network-Based Learning Controller", IEEE Transactions on Robotics and Automation, vol. 6, no. 1, February 1990.
[15]Xianzhong Cui, Kang G. Shin, "direct control and coordination using neural networks", IEEE TRANSATIONS ON SYSTEMS, MA, and cybernetics, vol. 23, No. 3, May/June 1993.