International Journal of Enhanced Research in Science, Technology & Engineering
ISSN: 2319-7463, Vol. 5 Issue 2, February-2016

Robust Neural Control for Robotic Manipulators

Ines BELAÏD DLIMI[1], Hichem KALLEL[2]

[1]Electrical Engineering, Department of Physics, National Institute of Applied Sciences and Technology (INSAT), Tunisia

[2]Electrical Engineering, Department of Physics, National Institute of Applied Sciences and Technology (INSAT), Tunisia

ABSTRACT

A robust neural control is designed for nonlinear dynamic systems. The objective of this work is to control the motion of the nonlinear system without any knowledge of its dynamics. This method of control requires only the measurement of the system state and its inputs. A Lyapunov function is proposed to ensure the stability of the nonlinear system equipped with the robust neural control. The online neural network's learning algorithm is concluded. A case study of a three dimensional rigid-link robotic manipulator is developed. Simulation results of the robot manipulator demonstrate the efficiency of the proposed robust neural control.

Keywords: Feedforward neural networks, inverse dynamics estimation, nonlinear dynamic system, robotic manipulator, robust neural control.

I. INTRODUCTION

Control methods based on exact dynamic models and precise physical parameters of dynamic systems are ineffective in practice because of the presence of dynamics uncertainities and other perturbations, such as load changes, frictions, external disturbances [16], [20], [23], [28]. Having exact models may also means that the dynamic system is not able to adapt to changes and uncertainties in its parameters or environement. This issue has been discussed in many works [1], [2] and [10]. Thus, Cheah, Liu and Slotine proposed in 2006 an adaptive controller for robot tracking control where the uncertain kinematics and dynamics' parameters are updated online [2]. According to them, even if the kinematics and dynamics parameters of robot manipulators, which are highly nonlinear, can be obtained with sufficient accuracy by calibration and parametric identification techniques, this can not be achieved for any object the robot manipulates.
Hence, based on their universal approximation properties and learning abilities, neural network controllers seem to be a good alternative for online identification issues, motion planning and control of highly nonlinear dynamic systems. Thus, tracking control using neural networks has inspired a great number of researchers [3], [5], [7], [9], [18], [21], [26-27]. Moreover, this method has been extended to uncertain dynamical systems as in [4], [14], [22], [29-31]. This has been largely debated by Cheng, Hou and Tan who proposed in 2009 a neural network based tracking controller for robot manipulators with uncertain kinematics, dynamics and actuator model. They demonstrated that the neural controller is not only a controller for trajectory tracking but also a kinematics estimator and dynamics compensator.
In this way, a simple and effective neural control for nonlinear dynamic systems with unknown kinematics and dynamics is designed in this paper. In section II, the nonlinear dynamic system is presented and the case of study is introduced. The neural control scheme, developed in section III, is composed by two components: the first corresponds to a neural network for the nonlinear system's inverse dynamics and any other modeled and unmodeled dynamics estimation, and the second is a visco-elastic component for the trajectory tracking. In this work, the dynamics of the nonlinear system is completely unknown. The absence of this knowledge is the reason for the choice of a neural control system, which will have as an objective the estimation of such elements. The online neural network learning algorithm is developed, in section IV, based on Lyapunov stability analysis of the nonlinear dynamic system, which ensures its global stability. This neural control is applied to a three dimensional three degrees of freedom robot manipulator. Perturbations in the end effector load and torques white noise are introduced in the robot dynamics to test and prove the robustness of the neural control through dynamic simulations, in section V, where the simulation results are presented and discussed.

II. NONLINEAR DYNAMIC SYSTEM

The dynamics of nonlinear systems can be considered as follows:

$$\dot{X}(t) = f(X, U, t) \qquad (1)$$

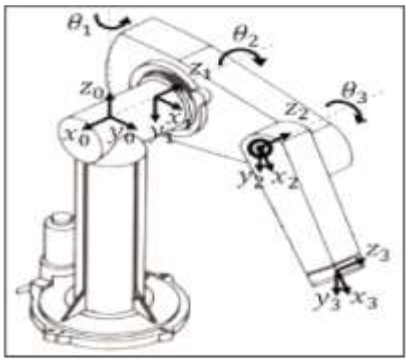

Fig. 1. Three dimensional robot manipulator

Here, X is the state vector of the nonlinear dynamic system, U is its input vector, t represents the variable of time and f is a nonlinear function of the system dynamics. The case study used in this paper corresponds to a three dimensional robotic system. The figure below (Fig. 1) describes the robot manipulator considered.

This robot manipulator is a three-link three dimensional rigid arm articulated by three revolute joints. An exact dynamics model of this robot manipulator may be expressed in matrix form ([11] , [15] and [15]), such as:

$$\mathrm{J}(\theta)\ddot{\theta} + \mathrm{H}(\theta,\dot{\theta}) + \mathrm{G}(\theta) + \Delta(\mathrm{t}) = \mathrm{D}\,\mathrm{U}(\mathrm{t}) \qquad (2)$$

Let d be the number of robotic system degrees of freedom. Here, $\mathrm{U}(\mathrm{t}) \in \mathcal{R}^{d*1}$is the control torques vector. $\theta \in \mathcal{R}^{d*1}, \dot{\theta} \in \mathcal{R}^{d*1}and\ddot{\theta} \in \mathcal{R}^{d*1}$ are respectively the vector of angular positions, velocities and accelerations. $\mathrm{J}(\theta) \in \mathcal{R}^{d*d}$ is the robotic system matrix of inertia. $\mathrm{H}(\theta,\dot{\theta}) \in \mathcal{R}^{d*1}$ is the vector of Coriolis and centripetal forces. $\mathrm{G}(\theta) \in \mathcal{R}^{d*1}$ is the vector of gravitational forces. $\mathrm{D} \in \mathcal{R}^{d*d}$is the actuators action matrix. t symbolizes the time variable. $\Delta(\mathrm{t}) \in \mathcal{R}^{d*1}$ represents the modeled un-modeled dynamics, e.g., joints friction, end effector's load perturbations and torques disturbances.

In this work, the robot matrix of inertiaJ, the Coriolis and centripetal forces vector H and the gravitaty vector G are completley unknown functions. The absence of this knowledge of the system dynamics is the reason for the choice of a neural control system, which will have as an objective the estimation of such elements. Equation (2) may be written in compact form as Equation (1), in order to be used for any nonlinear dynamic system, where the state vector is:

$$\mathrm{X} = [\theta^{\mathrm{T}} \quad \dot{\theta}^{\mathrm{T}}]^{\mathrm{T}} \qquad (3)$$

and the nonlinear function of its dynamics can be considered written as follows:

$$\dot{\mathrm{X}}(\mathrm{t}) = \mathrm{f}_1(\mathrm{X},\mathrm{U},\mathrm{t}) = \begin{bmatrix} \emptyset_{d*d} & \mathrm{I}_{d*d} \\ \emptyset_{d*d} & \emptyset_{d*d} \end{bmatrix} \mathrm{X} + \begin{bmatrix} \emptyset_{d*d} \\ \mathrm{J}(\theta)^{-1}\mathrm{D} \end{bmatrix} \mathrm{U} - \begin{bmatrix} \emptyset_{d} \\ \mathrm{J}(\theta)^{-1}\left(\mathrm{H}(\theta,\dot{\theta}) + \mathrm{G}(\theta) + \Delta(\mathrm{t})\right) \end{bmatrix} \qquad (4)$$

Here, $\mathrm{I}_{d*d} \in \mathcal{R}^{d*d}$ is the identity matrix, $\emptyset_{d*d} \in \mathcal{R}^{d*d}$ is the zeros matrix and $\emptyset_{d} \in \mathcal{R}^{d*1}$ represents the zeros vector.

III. NEURAL CONTROL

The neural control of the nonlinear dynamic system is composed by two elements. The first one is the estimation of the robot inverse dynamics and any other unmodeled dynamics. This component is comptuted through the neural network. The second element corresponds to a visco-elastic component for tracking purposes.

$$U = U_{NC} - [\Lambda_2 \quad \Lambda_1]\left(X - X_{ref}\right) \qquad (5)$$

Here, $\boldsymbol{X_{ref}} \in \mathcal{R}^{2d*1} = \left[\theta_{ref}{}^{T} \quad \dot{\theta}_{ref}{}^{T}\right]^{T}$ reprents the desired state vector, where $\boldsymbol{\theta_{ref}} \in \mathcal{R}^{d*1}$ and $\boldsymbol{\dot{\theta}_{ref}} \in \mathcal{R}^{d*1}$ are respectively the vector of desired angular positions and velocities. $\boldsymbol{\Lambda_1} \in \mathcal{R}^{d*d}$and $\boldsymbol{\Lambda_2} \in \mathcal{R}^{d*d}$ represent the visco-elastic gains and are positive definite diagonal matrices . The block diagram below, in Fig. 2, shows the structure of the neural control law. Here, $\boldsymbol{\phi}$ is the angular positions errors vector. $\boldsymbol{\dot{\phi}}$ is the angular velocities errors vector. $\boldsymbol{f_2}$ is a nonlinear function of the neural network learning algorithm.

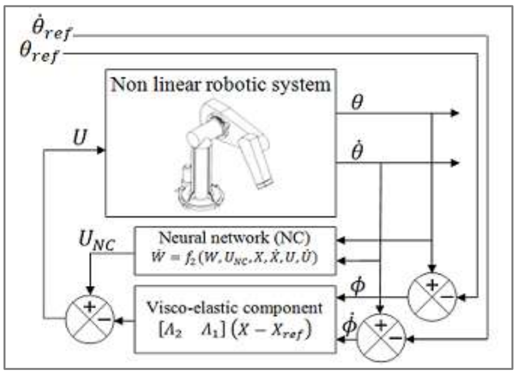

Fig. 2. Neural control structure

The neural control only uses the measurment of the state of the system($\boldsymbol{X}$) and output of the neural network ($\boldsymbol{U_{NC}}$), which is computed based on the robot input and its state.
Since the neurol control ($\boldsymbol{U}$) proposed has no knowledge of robot dynamic parameters, the proposed control is robust if stability conditions are fulfilled. A Lyaunov function is presented in the next section in order to prove the control robustness.

IV. NEURAL NETWORK CHARACTERISTICS

The neural network (NC) architecture is identical to that of the standard multi layer perceptron. Fig. 3 shows the structure of the neural network which is a two-layer feedforward neural network with a single hidden layer and an output layer.
The neural network outputs vector is called $\boldsymbol{U_{NC}} \in \mathcal{R}^{d*1}$. The neural network inputs vector $\boldsymbol{X} \in \mathcal{R}^{n*1}$ is composed by the outputs of the nonlinear dynamic system:

$$X = [x_1 x_2 \ \dots \ x_n]^T = \left[\theta^T \dot{\theta}^T\right]^T \quad (6)$$

The neural network weights in both hidden and output layers will be adjusted with the same learning algorithm. The activation function $\boldsymbol{h_1}$ of the neural network hidden layer is the hyperbolic tangent function. The activation function$\boldsymbol{h_2}$ of the neural network output layer is the linear function.

A. Activation functions

Let $\boldsymbol{w_{1,i,j}}$ be the weight of the$\boldsymbol{i^{th}}$ input and the$\boldsymbol{j^{th}}$ neuron of the hidden layer and $\boldsymbol{w_{2,i,j}}$ the weight of the$\boldsymbol{i^{th}}$ input and the$\boldsymbol{j^{th}}$ neuron of the output layer. The neural network outputs are computed by the neurons activation functions of the finite sums of the form:

$$U_{NC_r} = h_2\left(\sum_{j=1}^{m}\left(w_{2,j,r} * h_1\left(\sum_{i=1}^{n} w_{1,i,j} * x_i\right)\right)\right) \quad (7)$$

The activation function of the neural network output layer is the linear function, then the neural network equation can be expressed in the following compact matrix form:

$$U_{NC} = W_2{}^T * h_1\left(W_1{}^T * X\right) \quad (8)$$

Here, $\boldsymbol{W_1} = \left[W_{11}{}^T W_{12}{}^T \ \dots \ W_{1m}{}^T\right]^T \in \mathcal{R}^{(m*n)*1}$ is the vector of the$\boldsymbol{m}$ hidden layer neurons weights fed by the$\boldsymbol{n}$ NC inputs, where the $\boldsymbol{j^{th}}$ hidden neurons weights vector is $\boldsymbol{W_{1j}} = \left[w_{1,1,j} w_{1,2,j} \ \dots \ w_{1,n,j}\right]^T \in \mathcal{R}^{n*1}$.
$\boldsymbol{W_2} = \left[W_{21}{}^T W_{22}{}^T \ \dots \ W_{2p}{}^T\right]^T \in \mathcal{R}^{(m*p)*1}$is the vector of the $\boldsymbol{p}$ output layer neurons weights vector fed by the$\boldsymbol{m}$ hidden layer outputs, where the $\boldsymbol{j^{th}}$ output neurons weights vector is $\boldsymbol{W_{2j}} = \left[w_{2,1,j} w_{2,2,j} \ \dots \ w_{2,m,j}\right]^T \in \mathcal{R}^{m*1}$.

B. Learning algorithm and stability analysis

All the weights in both hidden and output layers of the neural network are adjusted with the same online learning algorithm. Hence, let $\boldsymbol{W} = \left[W_1{}^T W_2{}^T\right]^T$ be the whole neural network weights vector.
The neural network learning algorithm is elaborated based on the robotic system stability analysis, using the Lyapunov stability theorem.
Consider the following Lyapunov candidate function$V$:

$$V = \frac{1}{2}(U - U_{NC})^T(U - U_{NC}) + \frac{1}{2}\phi^T \Lambda_1 \phi \quad (9)$$

Its time derivative can be calculated as:

$$\dot{V} = \left(\dot{U} - \dot{U}_{NC}\right)^T (U - U_{NC}) + \phi^T \Lambda_1 \dot{\phi} \quad (10)$$

Using the established equation of the neural control (5), we obtain:

$$\dot{V} = \left(\dot{U} - \dot{U}_{NC}\right)^T (U - U_{NC}) - \phi^T (U - U_{NC} + \Lambda_2 \phi) \quad (11)$$

After the Lyapunov function time derivative equation factorization:

$$\dot{V} = \left(\dot{U} - \dot{U}_{NC} - \phi\right)^T (U - U_{NC}) - \phi^T \Lambda_2 \phi \quad (12)$$

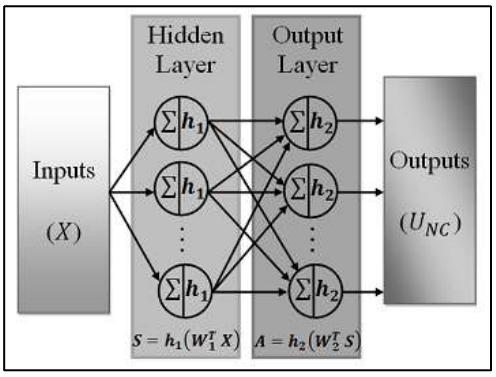

Fig. 3. Neural Network for Control (NC) structure

To ensure that this time derivative$\dot{\boldsymbol{V}}$ is negative, the following equality is imposed:

$$\left(\dot{U} - \dot{U}_{NC} - \phi\right) = -\psi(U - U_{NC}) \qquad (13)$$

With $\boldsymbol{\psi}$ is a known positive gain constant. The global stability of the robotic system is realized if (13) is verified. So, the Lyapunov function time derivative$\dot{\boldsymbol{V}}$ is computed by:

$$\dot{V} = -\{(U - U_{NC})^T \psi (U - U_{NC}) + \phi^T \Lambda_2 \phi\} \leq 0 \qquad (14)$$

Now, if we consider (8), the neural network outputs depend only on the vector $\boldsymbol{W}$ of the whole neural network weights and the vector$\boldsymbol{X}$ of the neural network inputs, such as:

$$U_{NC} = U_{NC}(W, X) \qquad (15)$$

The time derivative of $\boldsymbol{U_{NC}}$ is computed by:

$$\dot{U}_{NC} = \frac{\partial U_{NC}}{\partial W}\dot{W} + \frac{\partial U_{NC}}{\partial X}\dot{X} \qquad (16)$$

The neural learning algorithm is developed as to verify (13). Substituting $\dot{\boldsymbol{U}}_{\boldsymbol{NC}}$ in (13) by its value computed in (16), we can write the following equality:

$$\left(\dot{U} - \frac{\partial U_{NC}}{\partial W}\dot{W} - \frac{\partial U_{NC}}{\partial W}\dot{X} - \phi\right) = -\psi(U - U_{NC}) \qquad (17)$$

The learning algorithm of the neural network is thus obtained:

$$\dot{W} = \left(\frac{\partial U_{NC}}{\partial W}\right)^{pinv}\left(\dot{U} + \psi(U - U_{NC}) - \frac{\partial U_{NC}}{\partial X}\dot{X} - \phi\right) \qquad (18)$$

Note that since the Jacobian matrix $\left(\frac{\partial \boldsymbol{U_{NC}}}{\partial \boldsymbol{W}}\right)$ has more columns than rows, to compute its inversion, the pseudoinverse is required. The result is found using the Moore-Penrose pseudo- inverse [24] :

$$\left(\frac{\partial U_{NC}}{\partial W}\right)^{pinv} = \left(\frac{\partial U_{NC}}{\partial W}\right)^T \left(\left(\frac{\partial U_{NC}}{\partial W}\right)\left(\frac{\partial U_{NC}}{\partial W}\right)^T\right)^{-1} \qquad (19)$$

This solution satisfies some properties that are defined in Appendix B. The particularity of this method that it construct the solution with its minimum Frobenius norm. That may make the neural network's learning algorithm running faster.
Since (13) is verified, applying this learning algorithm (18) to compute the neural control ensures the global stability of the robotic system.

V. SIMULATION RESULTS AND DISCUSSION

In this section, the simulation results of the developed neural control method is applied to the three degrees of freedom three dimensional robot manipulator, described in section II.
According to the establishing link coordinate system of the robot manipulator presented in Fig. 1, the geometric parameters are defined using the Denavit-Hartenberg (D-H) representation [11-12].
The table below (Table 1) summarizes these geometric parameters of each joint $\boldsymbol{i}$and the robot arm inertial parameters of each link :
The poles of the system are chosen such as the visco-elastic component gains are:

$$\begin{cases}\Lambda_1 = diag([15 \quad 15 \quad 15]) \\ \Lambda_2 = diag([50 \quad 50 \quad 50])\end{cases}$$

The motion task is to move the robot end effector from the initial angular positions: $\boldsymbol{\theta_i} = [\pi/2 \quad 0 \quad 0]^T$to the desired angular positions: $\boldsymbol{\theta_{ref}} = [0 \quad -\pi/2 \quad \pi/4]^T$. The learning law's gain is$\boldsymbol{\psi} = 2$.
Two types of disturbances have been introduced into the robotic system:
The first one is a continuous white noise $\boldsymbol{U_{WN}}(t)$ applied to the actuators, with random normally distributed amplitude $\boldsymbol{A_{WN}(t)}$.

TABLE 1: LINKS AND JOINTS PARAMETERS OF THE ROBOTIC SYSTEM

Joint $\boldsymbol{i}$	$\boldsymbol{\theta_i}$ $(\boldsymbol{rad})$	$\boldsymbol{\alpha_i}$ $(\boldsymbol{rad})$	$\boldsymbol{a_i}$ $(\boldsymbol{mm})$	$\boldsymbol{d_i}$ $(\boldsymbol{mm})$	Link $\boldsymbol{i}$	Length $\boldsymbol{l_i}(m)$	Mass $\boldsymbol{m_i}(Kg)$	Center of mass $\boldsymbol{k_i}(m)$	Inertia $\boldsymbol{I_i}(Kg.m^2) * 10^{-3}$
1	θ_1	$-\pi/2$	0.3	0.8	1	0.3	3	$\begin{bmatrix}0.15\\0.04\\0.04\end{bmatrix}$	$\begin{bmatrix}33.8&0&0\\0&2.4&0\\0&0&2.4\end{bmatrix}$
2	θ_2	0	0.45	0	2	0.45	2	$\begin{bmatrix}0.25\\0.05\\0.05\end{bmatrix}$	$\begin{bmatrix}62.5&0&0\\0&2.5&0\\0&0&2.5\end{bmatrix}$
3	θ_3	0	0.33	0	3	0.33	1	$\begin{bmatrix}0.16\\0.03\\0.03\end{bmatrix}$	$\begin{bmatrix}24.2&0&0\\0&0.4&0\\0&0&0.4\end{bmatrix}$

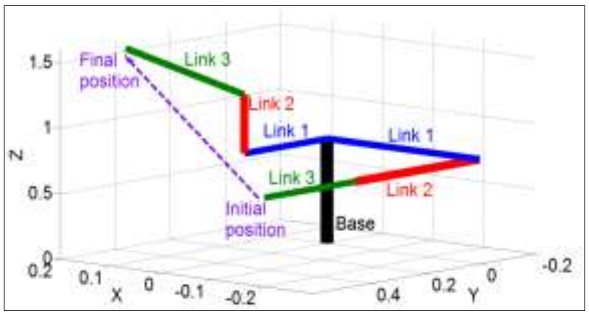

Fig. 4. Movement of the three links robot manipulator in space

The second one is an external load imposition applied suddenly in the robot end effector.
To calculate the neural control performance ($\boldsymbol{E_U}$), the torques vector of our neural control($\boldsymbol{U}$) is compared to the torques vector resulting from the dynamics of the robot manipulator ($\boldsymbol{U_{Rob}}$), expressed in (2). The performance measure is obtained at the end of simulation ($\boldsymbol{t = t_f}$), such as:

$$E_U = \sum_{t=0}^{t_f}\left((U - U_{Rob})^T(U - U_{Rob})\right) \qquad (20)$$

The matrix of inertia$\boldsymbol{J}$, the Coriolis and centripetal forces vector $\boldsymbol{H}$, gravitational forces vector $\boldsymbol{G}$ and the actuators action matrix $\boldsymbol{D}$ are defined and detailed in Appendix A. These matrices are used to simulate the robot dynamics but never used in computation of the neural control.
The designed neural network (NC) has a single neuron in the hidden and three neurons in the output layers. The evolution of the robotic system with $\boldsymbol{NC}$ is simulated in all cases considering the disturbance $\boldsymbol{\Delta(t)}$.
The

Fig. 6 below illustrates the mouvement of the robot manipulator from its initial position to its Final position.
First, the neural control has been computed considering a small continuous torques white noise as a disturbance $\boldsymbol{\Delta(t)}$ with a maximum amplitude$\boldsymbol{A_{WN} = 1Nm}$. The Fig. 7 below shows the angular positions and the input torques evolution with a small torques disturbance.
At the end of simulation time, we find:

- The performance measure: $E_U = 1.4694 * 10^{-26}$
- The neural control torques vector: $U = [-8.6098 \;\; 0.0338 \;\; 0.0274]^T(Nm)$

Secondly, the neural control has been computed considering the same small random torques white noise, in addition to a sudden end effector load change. Here, the robot third link weight changes suddenly from $\boldsymbol{1Kg}$ to $\boldsymbol{3Kg}$ at time $\boldsymbol{t = t_f/10}$. The Fig. 7 below gives the angular positions and the input torques evolution with a small torques disturbance and a sudden load change.
At the end of simulation time, we find:

- The performance measure: $E_U = 2.2917 * 10^{-26}$
- The neural control torques vector: $U = [-7.5853 \;\; 0.3150 \;\; 0.0571]^T(Nm)$

Finally, the neural control has been computed considering a hard continuous torques white noise as a disturbance $\boldsymbol{\Delta(t)}$. The maximum of this white noise amplitude $\boldsymbol{A_{WN} = 5Nm}$. The Fig. 7 below shows the angular positions and the input torques evolution with a hard torques disturbance.
At the end of simulation time, we find:

- The performance measure: $E_U = 5.9112 * 10^{-26}$
- The neural control torques vector: $U = [-7.1535 \quad 0.3909 \quad 0.2896]^T(Nm)$

These simulation results demonstrate that the neural control stabilizes the nonlinear robotic system and brings it to its desired state even in presence of several perturbations.
Indeed, we clearly see that even if we apply a sudden end effector mass change (Fig. 6) or a hard continuous disturbance on the robot torques (Fig. 7), the neural control is capable to quickly stabilize the system and drives it to its desired state. It gives excellent simulation results with negligible static positions errors ($\boldsymbol{\phi}$) and infinitely small performance measures ($\boldsymbol{E_U}$).
Therefore, we selected the simplest neural structure, with just one hidden neuron and only two layers, in order to minimize the time estimation due to the online application of the proposed neural control.
It is important to note that the neural network learning algorithm depends on the input torques vector time derivative that has been estimated using the mean value theorem.

VI. CONCLUSION

In this article, we developed a simple and effective robust neural control for nonlinear dynamic systems.
The robust neural control proposed is composed by two elements: first the output of a neural network which estimates the unknown nonlinear system inverse dynamics, and second, a visco-elastic component used to have a stable tracking behaviour. This neural control requires only the Only the state of this system and its inputs are available for its control.

The system parameters and dynamics are totally unknown. Furthermore, the neural network is designed with no more than one single neuron in the hidden layer and only two layers. The neural network learning algorithm is computed through a Lyapunov function which ensures the global stability of the nonlinear system.
The neural control is applied to a three degrees of freedom three dimensional robot manipulator. The simulation results demonstrate the efficiency and prove the robustness of this neural control.
This results lead us to conclude that more work needs to be realized in the selection of the neural network functions in order to minimize the inverse dynamics estimation time.

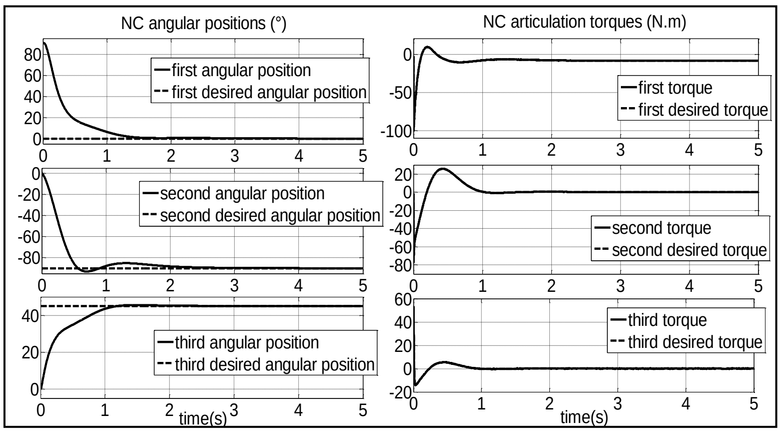

Fig. 5. The evolution of the angular positions and the input torques with a small torques disturbance

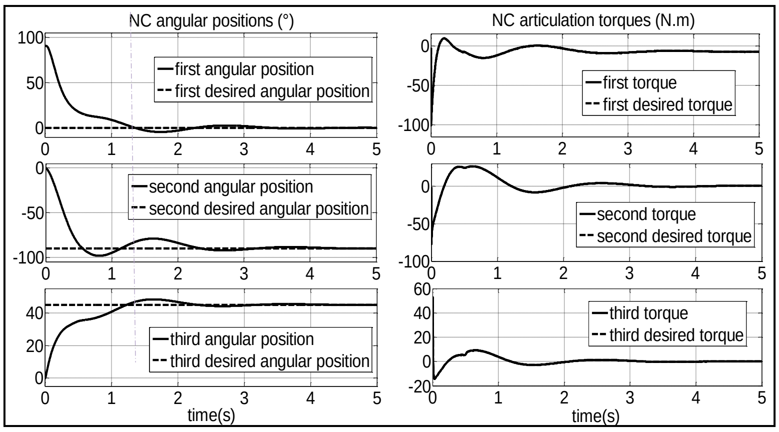

Fig. 6. The evolution of the angular positions and the input torques with a small torques disturbance and a sudden load change

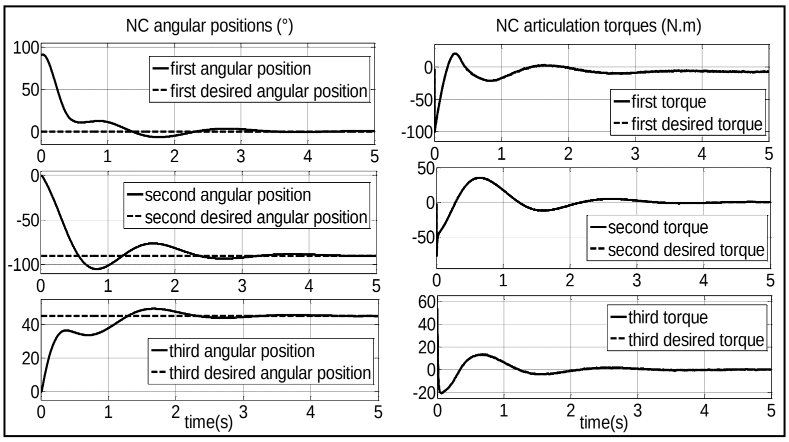

Fig. 7. The evolution of the angular positions and the input torques with a hard torques disturbance

VII. APPENDIX

A. Dynamics' model formulation

The matrices composing the dynamics' model of the three degrees of freedom three dimensional robotic manipulator, expressed in (2) are found by applying the Lagrange-Euler formulation [11], [12], such as:

$$\frac{d}{dt}\left(\frac{\partial L}{\partial\dot{\theta}_i}\right)-\frac{\partial L}{\partial\theta_i}=U_i(i=1,2,3) \qquad (21)$$

Here,$\boldsymbol{L}$ is the Lagrangien function of the robotic system:

$$L=K-P \qquad (22)$$

$\boldsymbol{K}$ and $\boldsymbol{P}$ are respectively the kinetic and the potentiel energies of the robot manipulator, given by:

$$K=\frac{1}{2}\sum_{i=1}^{3}\sum_{p=1}^{i}\sum_{r=1}^{i}\left(trace\left(U_{ip}J_iU_{ir}{}^T\right)\dot{\theta}_p\dot{\theta}_r\right),\quad P=\sum_{i=1}^{3}\left(-m_ig\ {}^0r_i\right) \qquad (23)$$

Let ${}^i\boldsymbol{r}_i=[x_iy_iz_i\ \ 1]^T$ be the homogeneous coordinates, expressed with respect to the i^{th} link coordinate frame. Thus, the matrix $\boldsymbol{U_{ij}}$ is the transformation matrix of the points ${}^ir_i$ on link i as the angular position θ_j changes, relative to the inertial coordinate system. The matrix $\boldsymbol{J_i}=\int\left({}^ir_i{}^ir_i{}^T\right)dm$ is the inertia of all the points of link i [12]. ${}^0\boldsymbol{r}_i$ is the same point ${}^ir_i$ with respect to the inertial coordinate system. The row vector $\boldsymbol{g}=[x_i\ y_i\ z_i\ \ 1]$ is the gravity force vector expressed in the inertial coordinate system.

Hence, the matrices of the robot dynamics described in (2) are identified through this Lagrange-Euler equation of motion (21) such as:

The symmetric matrix of inertia $J(\theta)\in\mathcal{R}^{3*3}$ is computed by:

$$\begin{cases}J_{ik}=\sum_{j=\max(i,k)}^{3}trace\left(U_{jk}J_jU_{ji}{}^T\right)\\ i,k=1,2,3\end{cases} \qquad (24)$$

The nonlinear Coriolis and centrifugal forces vector $H(\theta,\dot{\theta})\in\mathcal{R}^{3*1}$ is defined by:

$$\begin{cases}h_i=\sum_{k=1}^{3}\sum_{m=1}^{3}\sum_{j=max\,(i,k,m)}^{3}\left(h_{ikm}\dot{\theta}_m\dot{\theta}_k\right)\\ h_{ikm}=trace\left(U_{jkm}J_jU_{ji}{}^T\right),\quad i,k,m=1,2,3\end{cases} \qquad (25)$$

The gravity loading forces vector $G(\theta)\in\mathcal{R}^{3*1}$ is:

$$\begin{cases}G_i=\sum_{j=i}^{3}\left(-m_jgU_{ji}\ {}^jr_j\right),\qquad i=1,2,3\\ \qquad g=[0\ \ 0\ -9{,}8062\ \ 0]\end{cases} \qquad (26)$$

B. Pseudoinverse computing

The Jacobian $\left(\frac{\partial\boldsymbol{U_{NC}}}{\partial\boldsymbol{W}}\right)$ has 3 rows and 9 columns. To compute its inversion, we used the Moore-Penrose pseudoinverse:

$$\left(\frac{\partial U_{NC}}{\partial W}\right)^{pinv}=\left(\frac{\partial U_{NC}}{\partial W}\right)^T\left(\left(\frac{\partial U_{NC}}{\partial W}\right)\left(\frac{\partial U_{NC}}{\partial W}\right)^T\right)^{-1} \qquad (27)$$

Let $\boldsymbol{A}$ denotes the matrix $\left(\frac{\partial \mathrm{U_{NC}}}{\partial \mathrm{W}}\right)$that we want to inverse and $\boldsymbol{B}$ its pseudoinverse, such as:

$$A=\left(\frac{\partial U_{NC}}{\partial W}\right)andB=\left(\frac{\partial U_{NC}}{\partial W}\right)^{pinv}(28)$$

The solution computed of the Moore-Penrose pseudoinverse satisfies the following properties [24]:

$$\begin{cases}BAB=B\\ ABA=A\\ (AB)^T=AB\\ (BA)^T=BA\end{cases}(29)$$

Note that the Moore-Penrose pseudoinverse construct the solution with minimum Frobenius norm:$\|B\|_F^2=tr(B^TB)$, where $tr(.)$ is the trace operator.

VIII. REFERENCES

[1] C. C. Cheah, C. Liu and J.J.E. Slotine, "Adaptive Jacobian Tracking Control of Robots With Uncertainties in Kinematic, Dynamic and Actuator Models", *IEEE transactions on automatic control*, vol.51, no.6, pp.1024-29, June 2006.

[2] C.C. Cheah, C. Liu, J.J.E. Slotine, "Adaptive tracking control for robots with unknown kinematic and dynamic properties", *The Int. Journal of Robotics Research*, vol. 25, no. 3, pp. 283-296, March 2006.

[3] Chia Ju Wu, Ching Huo Huang, "A neural network controller with PID compensation for trajectory tracking of robotic manipulators", *Jour. of the Franklin Institute*, Vol. 333, No. 4, pp. 523-537, July 1996.

[4] Dlimi, I.B., Kallel, H., "Optimal neural control for constrained robotic manipulators", *Intelligent Systems (IS), 2010 5th IEEE Int. Conf.*, pp.302-308, 7-9 July 2010.

[5] H. D. Patiño, R. Carelli, B. R. Kuchen, "Neural networks for advanced control of robot manipulators", *IEEE Trans. Neural Networks*, Vol. 13, No. 2, March 2002.

[6] H. Hemami "A General Framework for Rigid Body Dynamics, Stability, and Control," *in Journal of Dynamic Systems, Measurement and Control*, vol. 124, No. 2, 241-251, 2002.

[7] Huaguang Zhang, Yanhong Luo, Derong Liu, "Neural-Network-Based Near-Optimal Control for a Class of Discrete-Time Affine Nonlinear Systems with Control Constraints," *Neural Networks, IEEE Transactions on*, vol.20, no.9, pp.1490-1503, September 2009.

[8] J. Slotine, W. Li, "Applied Nonlinear Control", *Prentice-Hall*, NJ, 1991.

[9] Jiang Y., Jiang Z.-P., "Computational Adaptive Dynamic Programming and Feedback Stabilization of Nonlinear Systems", *Neural Networks and Learning Systems, IEEE Transactions on*, vol. PP, no. 99, pp. 1, Jan 2014.

[10] Jiang Y., Jiang Z.-P., "Computational adaptive optimal control for continuous-time linear systems with completely unknown dynamics", *Automatica*, Vol. 48, no. 10, pp. 2699–2704, October 2012.

[11] Khalil W., "Modélisation et commande par calculateur du manipulateur MA-23; extension à la conception par ordinateur des manipulateurs", *EngineerPh.D., USTL*, Montpellier, Sept. 1976.

[12] King S. Fu, Rafael C. Gonzalez, C. S. George Lee, "Control, Sensing, Vision, and Intelligence", McGraw-Hill, inc., New York, 1987.

[13] L. Sciavicco, B. Siciliano, "Modeling and Control of Robot Manipulators", *New York Springer*, 2000.

[14] Long Cheng, Zeng-Guang Hou and Min Tan, "Adaptive neural network tracking control for manipulators with uncertain kinematics, dynamics and actuator model", *Automatica*, Vol. 45, no. 10, pp. 2699–2704, October 2009.

[15] M.W. Spong, M. Vidyasagar, "Robot Dynamics and Control", *New York Wiley*, 1989.

[16] Mark W. Spong, Frank L. Lewis, Chaouki T. Abdallah, "Robot control: dynamics, motion planning, and analysis", *IEEE Control Systems Society*, 1993.

[17] Michel Anthony N., Hou Ling, Liu Derong, "Stability of dynamical systems: continuous, discontinuous, and discrete systems", *Boston Springer*, 2008.

[18] Min Chee Choy, Srinivasan D., Cheu R.L., "Neural Networks for Continuous Online Learning and Control", *Neural Networks, IEEE Transactions on* , vol.17, no.6, pp.1511-1531, Nov. 2006.

[19] Molchanov A. and Derong Liu, "Necessary and sufficient conditions for robust absolute stability of time varying nonlinear continuous time systems", *Decision and Control, Proceedings of the 40th IEEE Conference on*, vol.1, pp. 45-50, 2001.

[20] NidalFarhat, Vicente Mata, Álvaro Page, Francisco Valero, "Identification of dynamic parameters of a 3-DOF RPS parallel manipulator", *Mechanism and Machine Theory*, vol. 43, no. 1, pp. 1–17, Jan 2008.

[21] O. Barambones, V. Etxebarria, "Robust neural control for robotic manipulator", *Automatica*, Vol. 38, No. 2, pp. 235–242, Feb. 2002.

[22] Qinmin Yang; Zaiyue Yang; Youxian Sun, "Universal Neural Network Control of MIMO Uncertain Nonlinear Systems", *Neural Networks and Learning Systems, IEEE Transactions on* , vol.23, no.7, pp. 1163-1169, July 2012.

[23] R. Ortega, M. Spong, "Adaptive motion control of rigid robots: A tutorial", *Automatica*, vol.25, no.6 pp. 877-888, 1989

[24] R. Penrose, J. A. Todd, "A generalized inverse for matrices", *Mathematical Proceedings of the Cambridge Philosophical Society*, vol. 51, no.3, 1955.

[25] Sanner, R.M.; Slotine, J.-J.E., "Gaussian networks for direct adaptive control", *Neural Networks, IEEE Transactions on*, vol.3, no.6, pp.837-863, Nov 1992.

[26] Tao Cheng, Lewis F. L., Abu-Khalaf M., "A Neural Network Solution for Fixed-Final Time Optimal Control of Nonlinear Systems," *Control and Automation, 2006. MED '06. 14th Mediterranean Conference on*, vol., no., p.p. 1-5, June 2006.

[27] Tao Cheng, Lewis F. L., Abu-Khalaf M., "Fixed-Final-Time Constrained Optimal Control of Nonlinear Systems Using Neural Network HJB Approach", in *IEEE Trans. on Neural Networks*, vol.18, no.6, pp.1725-37, Nov. 2007.

[28] Vicente Mata, NidalFarhat, Miguel Díaz-Rodríguez, Ángel Valera and Álvaro Page, "Dynamic Parameters Identification for Parallel Manipulators", *INTECH*, 2008.

[29] Xuemei Ren; Rad, A.B.; Lewis, F.L., "Neural Network-Based Compensation Control of Robot Manipulators with Unknown Dynamics", *American Control Conference ACC'07,* pp.13-18, July 2007.

[30] Yan, Z.; Wang, J., "Robust Model Predictive Control of Nonlinear Systems with Unmodeled Dynamics and Bounded Uncertainties Based on Neural Networks", *Neural Networks and Learning Systems, IEEE Transactions on*, vol. 25, no. 3, pp.457-469, March 2014.

[31] Zhao-Hui Jiang; Ishida, T., "Trajectory tracking control of industrial robot manipulators using a neural network controller", *IEEE Int. Conf. on Systems, Man and Cybernetics, ISIC,* pp.2390-2395, 7-10 Oct. 2007.