International Journal of Computing and Digital Systems

http://dx.doi.org/10.12785/ijcds/020302

A Practical Neuro-fuzzy Mapping and Control for a 2 DOF Robotic Arm System

Ebrahim Mattar

Dept. of Electrical and Electronics Engineering,
College of Engineering, University of Bahrain, P.O. Box 32038, Kingdom of Bahrain.

E-mail addresses: ebmattar@theiet.org, ebmattar@ieee.org

Received 23 Sep. 2012, Revised 9 Feb. 2013, Accepted 15 Mar. 2013, Published 1 Sep. 2013

Abstract: Relating an arm Cartesian space to joint space and arm dynamics, is an essential issue in arm control that has been given a substantial attention by number of researches. Arm inverse kinematic, is a nonlinear relation, and a closed form solution is not a straight forward, or does not even always exist. This research is presenting a practical use of Neuro-Fuzzy system to solve inverse kinematics problem that used for a two links robotic arm. The concept here is to learn kinematics relations for a robotic arm system. This is to learn and map its environment and remembers what it learnt. For learning the inverse kinematics, Neuro-fuzzy needs information about coordinates, joint angles and actuator position. Information flow needed for the training for a Neuro-fuzzy network is slow and difficult to get by measuring the real structure. Desired Cartesian coordinates are given as input to a Neuro-fuzzy that returns actuator positions. Hence to express them as linguistics fuzzy rules. Neuro-fuzzy system is to generalize and produce an appropriate output. The assembled system has been equipped with C^{++} interface routines, as being executed from a MATLAB environment, in addition to high-speed low-level communication with the robotic arm sensing devices.

Keywords: UOB Robotic Arm; Inverse Dynamics; Computed Torque Law; Neuro-fuzzy mapping.

I. Introduction

Robotics arms are widely used and employed for industrial and non-industrial applications. However, for more precise and accurate motion control, dynamic model do play important role for such applications. It is always not an easy task to get the forward and the inverse models for robotics structure, specifically, once redundancy exists. Kinematics models are always nonlinear relations, and closed form solutions are not easy tasks to be achieved. There are a number of approaches that have been reported in literature regarding building kinematics models. In its boarder sense, manipulator kinematics is the study of motion without regard to the forces which cause it. Within kinematics, it is possible to study position, velocity and acceleration, and all higher order derivatives of an arm position variables. The kinematics of manipulators involves the study of the geometric and time based properties of the motion, and in particular how the various links move with respect to one another and with time. Typical robots are serial-link manipulators comprising a set of bodies, (*links*), in a chain, connected by joints. Each joint has a single **D**egree **o**f **F**reedom (**DOF**), either translational or rotational. For a manipulator with (n joints numbered from 1 to n, there are ($n+1$) links, numbered from 0 to (n). Link 0 is the base of the manipulator, generally fixed, and link n carries the end-effector. Joint (i) connects links (i) and i first and last links are meaningless, but are arbitrarily chosen to be 0. Joints may be described by two parameters. The link (o) set is the distance from one link to the next along the axis of the joint. The joint angle is the rotation of one link with respect to the next about the joint axis. To facilitate describing the location of each link we affix a coordinate frame to it, frame (i) is attached to link (i).

Denavit and Hartenberg [1] proposed a matrix method of systematically assigning coordinate systems to each link of an articulated chain. Axis of revolute joint (i) is aligned with (Z_i). Parallel link and serial/parallel hybrid structures are possible, though much less common in industrial manipulators.

© 2013 UOB SPC, University of Bahrain

A. Issues Related to Robotics Task-Space Control.

When we restrict ourselves to the control of robotic arms, we generally faced by three dedicating issues: (*i*) If a target (a grasp) position is known, usually in Cartesian and where an arm gripper must move to, the corresponding joint angles must be computed. This problem known as "INVERSE KINEMATICS". (*ii*) Secondly, a path must be generated along which each joints must be moved from the current position. This problem is known as "PATH PLANNING". (*iii*) Third, the right torques must be exerted on the joints (by giving an actuator the accurate current). This problem is also known as "INVERSE DYNAMICS".

B. ANN Arm Control: Advanced Robotic Arm.

Control: In many studies of nonlinear system control, **A**rtificial **N**eural **N**etworks (**ANN**) have been used as effective solutions by exploiting their nonlinear mapping properties. Learning and adaptive capabilities of ANN to control nonlinear systems, have played an important role in performance and have proven its promising future of ANN as an auxiliary nonlinear controller.

Typical popular nonlinear systems, are the multi DOF robotic manipulators. Consequently, they do need nonlinear control methodologies. A one possibility is an ANN based methods. Over the last few decades there has been rapid development in both the theory and application of Artificial Neural Networks. Taking advantage of those characteristics of ANN, many ANN control schemes have been proposed in the literature. The important issue of ANN control application is to determine an appropriate training signal for training purposes. The known "back propagation-learning" algorithm is always used to adjust the internal weights in on-line fashion.

Since the control application of ANN requires an on-line tuning, hence the adaptation capability of ANN plays more important role than that of learning capability. It has been reported that the complete mapping from one domain to another domain, to identify INVERSE DYNAMICS of robot manipulators, is not easy. This often requires an off-line training procedure, which is time consuming.

One of the ANN applications was done by Bogdanov and Timofeev [2]. Here ANN controller was synthesized to compensate dynamics approximation errors in the model of a two link robotic system. Thus providing robust control. Obtained robustness estimates for the developed ANN algorithms establish relation between transients quality and parameter disturbances caused by inaccurate approximation. The controller used to stabilize the system could be any type because it's independent of ANN training parameters. Refer to Fig. (1), Fig. (2), and Fig. (3) for possible robotic arm control topologies. Among the various kinds of ANNs, great attention has been devoted to those (called mapping networks) which, through simple, ordered processing structures, reproduce the main functions of the human brain, in particular those of learning and adaptation. These ANNs can learn a mapping between an input and an output and synthesize an associated memory which gives the appropriate output when a certain input is presented to it. The output is either one corresponding to an input which is known at the start, or is the result of a capacity for generalization when the input is unknown. In addition, due to their inherent nonlinearity, these neural models are capable of executing functional approximations better than the best classical techniques, which are essentially based on linearization hypotheses. Such capabilities, along with others of less importance, are supported by the considerable calculation speed characterizing ANNs, due to their massively parallel architecture. Such characteristics have made the use of ANN a valid alternative to the better-known classical techniques in the solution of various problems ranging from pattern

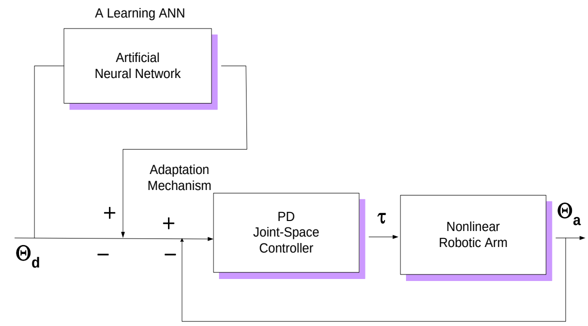


Fig. 1. A static ANN robotic arm control. ANN is trained once. **PD** is standing for **P**roportional **D**erivative Controller synthesis.

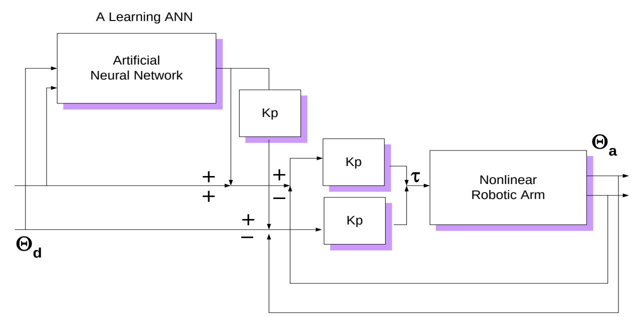


Fig. 2. Another topology of ANN robotic arm control.

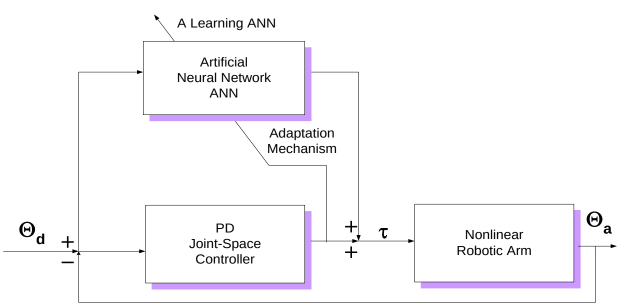


Fig. 3. Adaptive type ANN based arm control detailed structure.

recognition to process control, to telecommunications, to robotics.

C. Feedback Error Learning Technique.

One popular non-model based robot arm control approach is to apply a simple **P**roportional **D**erivative Controller (**PD**) synthesis to a robot manipulator. The PD controllers are continuously stable, but its tracking performance is generally poor due to uncertainties caused by the robot dynamics. To compensate for such uncertainties, feedback error learning scheme has proposed to augment the PD controller as shown in Fig. (2). This approach requires the ANN to identify the complete robotic arm inverse dynamics for possible compensation, De Azevedo and Barreto [3].

D. Other ANN Control Technique.

Hisa, as in [4], has applied a PD controller system for a robot arm. He stated that, "The PD controller system is always stable, but its tracking performance is generally poor due to uncertainties caused by the robot dynamics. To compensate for such uncertainties, feedback error learning (FEL) scheme has proposed to augment the PD controller as shown in Fig. (3). This approach requires the NN to identify the complete robot inverse dynamics for compensation", Hsia [4].

One of the ANN applications most explored in the last few years is the adaptive control of robotic arms with unknown dynamics. This kind of control is generally split into two phases: identification of the most representative parameters of the system dynamics to be controlled, and design of the most suitable regulator for this control. The presence of an identification module and the possibility of on-line tuning of the regulator parameters allow any external disturbance or variations in the system dynamics itself to be compensated for, thus guaranteeing successful control of a process in all its dynamic conditions. In the past three decades, major progress has been made in adaptive identification and control for linear time-invariant plants with unknown parameters. The mathematical theories developed for adaptive control (both identification and design of the regulator) are now well consolidated and are all based on linear algebra and the theory of ordinary linear differential equations. In other words, the choice of the identifier and controller structures is based on well-established results in linear systems theory.

In their research paper, Lakshmi and Mashuq [5], have both introduced an adaptive Neuro-Fuzzy control method. This is for a Cartesian motion control of a 4 DOF robot arm. In their paper, the foucs was the control of Selective Compliant Assembly Robot Arm (SCARA) type robot arm. The main controller concept was based on the use of inverse learning Adaptive Neuro-Fuzzy Inference System (ANFIS) model only to train itself from certain given robot trajectories. In reality, these trajectories should be obtained by directly measuring the robot arm responses for given inputs to capture the actual dynamics in the presence of all uncertainties. The employed approach was used for the design and implementation of an ANFIS controller which has shown to work with satisfactorily performance.

A Neuro–Fuzzy Controller synthesis for position control of robotic arm was also presented by Tavoosi et. al. [6]. In their approach, they stated that, "robot manipulators have become increasingly important in the field of flexible automation. So modeling and control of robots in automation will be very important. But Robots, as complex systems, must detect and isolate faults with high probabilities while doing their tasks with humans or other robots with high precision and they should tolerate the fault with the controller." In this respect and background, they introduced a Neuro-Fuzzy Controller (NFC) for position control of robot arm. Hence, they proposed a five layer ANN to adjust input and output parameters of membership function in a fuzzy logic controller. For training purposes, a hybrid learning algorithm was also used for training of such 5-layres ANN network. While using such a learning algorithm, the least square estimation method is applied for the tuning of linear output membership function parameters and the error backpropagation method is used to tune the nonlinear input membership function parameters. To validate the proposed NFC algorithm, the obtained simulation results show that Neuro fuzzy controller is better and more robust than the PID controller for robot trajectory control.

In [7], both Pham and Fahmy have introduced a Neuro-Fuzzy Modelling and Control technique for Robotics Manipulators Trajectory Tracking system. In their research efforts, they presented a novel Neuro-fuzzy controller synthesis for robotic manipulators control. First, an inductive learning technique is applied to generate the required modelling rules from input/output measurements recorded in the off-line structure learning phase. Second, a fully differentiable fuzzy neural network is developed to construct the inverse dynamics part of the controller for the on-line parameter learning phase. Finally, a fuzzy-PID-like incremental controller was used and employed as feedback servo-controller. For validation purposes, the suggested control system was also tested using dynamic model of a six-axis industrial robot (6-DOF) arm. The control system showed good results compared to the conventional-PID individual joint controller.

Furthermore, Lazarevska [8] have introduced a Neuro-fuzzy modeling network for the issue of inverse kinematics problem of a 4 DOF robotic arm. In this context, the manuscript presented a detailed structure of Neuro-fuzzy model of the inverse kinematics of 4 DOF robotic arm employing the relevance vector learning algorithm. Lazarevska [8] has stated that, "although the direct kinematics of the robotic arm can be modeled with ease by the same approach, the paper focuses on the much more interesting kinematic task, since its solution presents a basis for robot control design". Hence, the presented model was based on the use of a Takagi-Sugeno type, but its parameters and number of fuzzy rules are automatically generated and optimized through the adopted learning algorithm based on M. E. Tipping's relevance vector machine. The presented model illustrates the effectiveness of the adopted neuro-fuzzy modeling approach.

In terms of MIMO NARX models, pham et. al. [9] have presented Dynamic model identification of the 2-Axes PAM robot arm using neural MIMO NARX model. In their resarch, a novel inverse dynamic MIMO NARX model is employed for modeling and identifying simultaneously both of joints of the prototype 2-axes PAM robot armpsilas inverse dynamic model. In reality, the contact force variations and highly nonlinear coupling features of both links of the 2-axes PAM robot arm are modeled thoroughly through an inverse neural MIMO NARX model-based identification process using experiment input-output training data. For the first time, the nonlinear inverse dynamic MIMO NARX model scheme of the prototype 2-axes PAM robot arm has been investigated. For validation, the obtained results have shown that proposed dynamic intelligent model trained by back propagation learning algorithm yields outstanding performance and perfect accuracy. Kelly et. al., [10], and in their proposed control method, they presented and discussed a reasoning and a technique for combining artificial neural networks (ANN) and fuzzy logic structure. Hence, they also presented a discussion of the problem of moving a robotic arm in the presence of an obstacle. The approach was based on the use of several Neuro-fuzzy controllers as are trained, using sample data obtained from a human's control of a robotic arm. Their performance is quantified and compared. Kelly et. al., [10] have shown that the definition of the fuzzy membership functions plays a significant role in the ability of the Neuro-fuzzy controller to learn and generalize. Possible directions for future work are suggested.

E. Manuscript Contribution.

Having presented few literature works within this focus, the main objective of this research is to employ a Neuro-fuzzy architecture for a control of a home built robotic arm system. This would include a secondly defined objective, which is to build a robotic arm system with an adequate sensing abilities being interfaced to a high level computing environments (Matlab and C^{++}). The development of such an arm is useful for gentle exploration of an unknown objects in un-structured environments. Similar to a human, an object position is sensed by the eyes and then the arm move toward it in precise motion to get the object; here we need also feedback sensors to be included in our system, and even cheap feedback sensor can do the job. To accomplish this, our strategy is to develop a good model and controller of the system. Using an arm manipulator, by careful modeling and parameter estimation, does the development of the system model. Usually, a position of an object is given in Cartesian. This dictates, to do the inverse kinematics. After modeling the system and found a suitable controller, a program is needed to predict the manipulator torque required to follow a certain desired trajectory to reach an object.

F. Manuscript Organization.

The manuscript has been divided for six Sections. In SECTION (I), we gave an introduction to concept of model based robotic arm control. Few literature studies are given within this section. The UOB 2-DOF robot arm system and associated models are also described in SECTION (II). In specific, we present the UOB arm system. Here both KINEMATIC and DYNAMIC models do play important role over an arm control. Hence, this section is focused to present the overall hardware implementation and all the work related to communication and interfacing with the arm. In Section (III), we present the concept of the learning system, that was built behind the Neuro-fuzzy system. The Neuro-fuzzy UOB-Arm control I s also presented in SECTION (IV). In SECTION (V), we present few discussion remarks, and discus the achieved results. Finally, in SECTION (VI), we draw few conclusions remarks.

II. UOB Robot Arm system

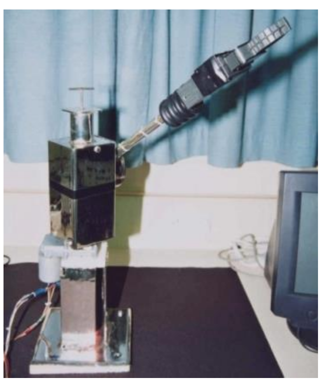
In order to program and control a robotic arm system, a first step towards this task, is to learn the arm dynamics and related kinematics. Robot dynamics algorithm is a process for calculating equations of motion of an arm robot mechanism, and to learn the acting forces that cause these motions. Arm dynamics do appear in two different representations: (*i*) FORWARD DYNAMICS, which calculates how the robot will move in response to a given force, (*ii*) and inverse dynamics. INVERSE DYNAMICS calculates what are the forces that are required to make a defined motion. The former are used in simulators and the latter in control systems.

A. UOB Robot Arm Dynamic Modelling.

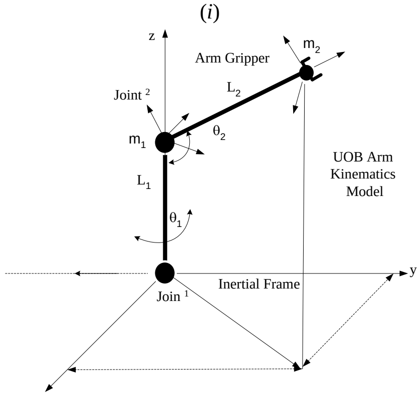
Typical, robotic arms consist of a serial-link. Manipulators are comprising a set serial-links in a chain, as connected by joints. Each joint has a single DOF either rotational or translational. Each joint is supported by a torque. Joints torques are produced by DC motor. It is important to have accurate sensing of the arm motion. Therefore, driving motors are fitted with sensors for control purposes. The two links robotic system consists of two links with representations to their lengths (ℓ_1) and (ℓ_2) for (link_1) and (link_2) respectively. In addition, links weights of (m_1) and (m_2) for link_1 and link_2. A general inverse dynamic equation of a two link robotic system is known in form of Lagrange's equation. It sums total forces affects the two link robotic systems. In terms of (I) is the moment of inertia, $(\alpha = \ddot{\theta})$ is the acceleration of the rigid-body and (τ_{in}) is torque applied. The specific Lagrange's equation for a typical (n) links robotic arm is expressed by:

$$M(\theta)\ddot{\Theta} + N(\theta,\dot{\theta})\dot{\theta} + C(\theta) = \tau_{in} \tag{1}$$

in which the used symbols designate the followings: θ is arm joint trajectory and $\dot{\theta}$: velocity trajectory. (*rad/sec*), $\ddot{\theta}$: acceleration of the trajectory. (*rad/sec*^{2}). $\tau_{in} \in \Re^{(2\times 1)}$ input torque vector. (*N.m*), $M \in \Re^{(2\times 2)}$: inertia forces matrix. (*N.m*). $N \in \Re^{(2\times 1)}$ gravity vector. (*N.m*), $C \in \Re^{(2\times 1)}$ centrifugal force vector. (*N.m*). The two-links UOB manipulator with rotational joints (θ_1, θ_2) is shown in Fig. (4).



(*i*)



(*ii*)

Fig. 4. (*i*) UOB Two-links two actuation robotic arm system. (*ii*) Arm kinematics and related frames assignments model, [11].

Each UOB link has a point masses (m_1) and (m_2) at distal end of links. The dynamic equation for a two DOF manipulator in joint space coordinates are given by the form:

$$M(\theta)\ddot{\Theta} + N(\theta,\dot{\theta})\dot{\theta} + C(\theta) + \tau_F = \tau_{in} \tag{2}$$

The input torque vector $(\tau_{in1} \quad \tau_{in2})^T$ do represent input torques to both arm links (ℓ_1, ℓ_2). Arm inertia force matrix:

$$M = \begin{pmatrix} m_{11} & m_{12} \\ m_{21} & m_{22} \end{pmatrix} \tag{3}$$

$$M = \begin{cases} m_{11} = (m_1 + m_2)\ell_1^2 + m_2\ell_2^2 + 2m_2\ell_1\ell_2\cos(\theta(1)) \\ m_{12} = (m_2)\ell_2^2 + m_2\ell_1\ell_2\cos(\theta(2)) \\ m_{21} = (m_2)\ell_2^2 + m_2\ell_1\ell_2\cos(\theta(2)) \\ m_{22} = m_2\ell_2^2 \end{cases} \tag{4}$$

The arm centrifugal forces, are also given by $N(\theta,\dot{\theta})$:

$$N(\theta,\dot{\theta}) = \begin{pmatrix} n_1 \\ n_2 \end{pmatrix} = \begin{pmatrix} -m_2\ell_1\ell_2\left(2\dot{\theta}(1)\dot{\theta}(2) + \dot{\theta}(2)^2\right)\sin(\theta(2)) \\ +m_2\ell_1\ell_2\dot{\theta}(1)^2\sin(\theta(2)) \end{pmatrix} \tag{5}$$

The arm gravity force vector, is therefore given as $C(\theta)$:

$$C(\theta) = \begin{pmatrix} c_1 \\ c_2 \end{pmatrix} = \begin{pmatrix} (m_1 + m_2)g\ell_1\cos(\theta(1) + m_2 g\ell_2\cos(\theta(1) + \theta(2))) \\ m_2 g\ell_2\cos(\theta(1) + \theta(2)) \end{pmatrix} \tag{6}$$

Further analysis of the UOB robot arm system, shows that it can be represented as a one dynamically defined computational block. This involves the inverse dynamics for the two links arm system and solving for (θ_1, θ_2). Ignoring effect of (τ_F), and rewriting the relationship between joint velocity and Cartesian space velocity, this involves the followings:

$$M(\theta)\ddot{\Theta} + N(\theta,\dot{\theta})\dot{\theta} + C(\theta) = \tau_{in} \tag{7}$$

$$\dot{x} = J\dot{\theta} \quad \text{and} \quad \ddot{x} = J\ddot{\theta} + \dot{J}\dot{\theta} \tag{8}$$

$$\ddot{\theta} = J^{-1}\left(\ddot{x} - \dot{J}\dot{\theta}\right) \tag{9}$$

$$M\left(J^{-1}\left(x - \dot{J}\dot{\theta}\right) + N(\dot{\theta}) + C(\theta)\right) = \tau_{in} \tag{10}$$

$$\tau_{in} = J^T F \tag{11}$$

$$M\left(J^{-1}\left(x - \dot{J}\dot{\theta}\right) + N(\dot{\theta}) + C(\theta)\right) = J^T F \tag{12}$$

$$\left(J^T\right)^{-1} M\left(J^{-1}\left(x - \dot{J}\dot{\theta}\right) + N(\dot{\theta}) + C(\theta)\right) = F \tag{13}$$

The above derived equation, do represent the actuator forces in Cartesian space.

B. Jacobian Based Inverse Kinematics.

In the field of robotics, we generally refer to the Jacobian matrix. Jacobian relates joint position and velocity to Cartesian position and velocity arm posture. An approximation of the kinematics equation for two links arm is obtained as shown below:

$$P_{(x,y,z)} = \begin{cases} P_x = cos(\theta_1)(\ell_2 cos(\theta_2+\theta_3)+\ell_1 cos(\theta_2)) \\ P_y = sin(\theta_1)(\ell_2 cos(\theta_2+\theta_3)+\ell_1 cos(\theta_2)) \\ P_z = sin(\theta_1)(\ell_2 cos(\theta_2+\theta_3)+\ell_1 cos(\theta_2)) \end{cases} \quad (13)$$

$$P_{(x,y,z)} = \begin{pmatrix} P_x \\ P_y \\ P_z \end{pmatrix} \Rightarrow \Theta(\theta_1,\theta_2) = \begin{pmatrix} P_x \\ P_y \\ P_z \end{pmatrix} \quad \text{then}$$

$$J = \frac{\partial P_{(x,y,z)}}{\partial \Theta} = \begin{pmatrix} \frac{\partial P_x}{\partial \theta_1} & \frac{\partial P_y}{\partial \theta_1} & \frac{\partial P_z}{\partial \theta_1} \\ \frac{\partial P_x}{\partial \theta_2} & \frac{\partial P_y}{\partial \theta_2} & \frac{\partial P_z}{\partial \theta_2} \end{pmatrix} \quad (14)$$

In our work we will not use the Jacobians due to number of reasons. Specifically, some of them are: (*i*) Do not give precise results. (*ii*) The inverse could not be defined due to singularities. (*iii*) Specific closed form solutions, may be out of the environment range. (*iiv*) *Computed Torque Controller (iv)* Defining an Error Based Control Law. As we saw, from the above, that an arm dynamic equation is a highly nonlinear and complex to model. In an absence of an accurate arm model, however, control of a robotic arm motion is frequently carried out with a standard PD controller. Such an approach is referred to as non-model based control. The PD based control for robot manipulators has been of great interest because of its simplicity and stability. The control law has the form:

$$\tau = (K_d \dot{e} + K_p e) \quad (15)$$

In Equ. (15), the term (e) do represent the error in Cartesian arm posture. This is due to joint space motion demand. Substitute (2) in torque equation (1). Evidently, the error dynamics can also be represented by the left hand side:

$$(M\ddot{e} + K_d \dot{e} + K_p e) = (M\ddot{\theta} + N\dot{\theta} + C + \tau_f) \quad (16)$$

Multiplying by inertia matrix inverse, i.e. (M^{-1}):

$$(\ddot{e} + M^{-1}K_d \dot{e} + M^{-1}K_p e) = M^{-1}(M\ddot{\theta} + N\dot{\theta} + C + \tau_f) \quad (17)$$

In above last equation of Equ. (17), this makes and behave unpredictable as the robot arm configuration and dynamics do change once the arm is in motion. However, when controller gains are high such that $(K_d \rangle\rangle 1)$ and $(K_p \rangle\rangle 1)$, the above equation of Equ. (17) can be approximated by a first order differential equation of the following:

$$((\dot{e} + K_d^{-1}K_p e) \cong -K_d^{-1}M\ddot{e} + K_d^{-1}(M\ddot{\theta} + N\dot{\theta} + C + \tau_f)) \cong 0 \quad (18)$$

From Equ. (18), the error dynamics are now nearly independent of arm model. Thus the choice of PD gains has crucial effects on the performance of robot manipulators.

C. Cartesian Control : Non-model Based Control.

Computed torque control in Cartesian space is shown in Fig. (5). The actual arm model system performance in Cartesian space will be elaborated on more in SECTION (IV.) Here the arm performance is degraded and unpredictable. Thus the computed torque based position control in Cartesian space (using system Jacobian) is not robust in practice. In our situation we shall extend the same above design approach to non-model based robot control as will be shown in later in Fig. (7). From Fig. (5) a defined control law is stated as:

$$S = (\ddot{\theta}_d + K_d \dot{e} + K_p e) \quad (19)$$

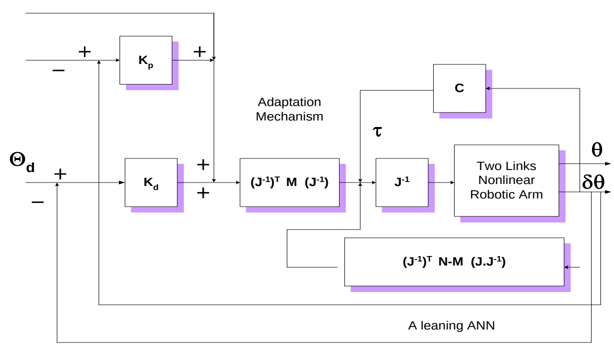


Fig. 5. Computed torque control in Cartesian space.

$$(\ddot{\theta}_d + K_d \dot{e} + K_p e) = (M^*\ddot{x} + N^*\dot{\theta} + F_f^*) \quad (20)$$

A learning mechanism is used to learn parts of the Computed torque control parameters. A computer code was hence written for simulation (using numerical integration to solve the dynamics), where the controller parameter (K_d) and (K_p) were estimated by standard tuning approaches. The resulting input torque to the system will be in the form:

$$\tau_{in} = \hat{M}(\theta)(\ddot{\theta}_d + K_d \dot{e} + K_p e) + \hat{N}(\theta,\dot{\theta})\dot{\theta} + \hat{C}(\theta)\theta \quad (21)$$

Finally, simulating the computer torque method, will generate a set of input-output patterns, that can be used for learning purposes.

The learning mechanisms is based on using the architecture of Neuro-fuzzy system. This is additional enlightened within SECTION (III).

III. A LEARNING NEURO-FUZZY ARCHITCTURE

Neuro-fuzzy systems combine the positive attributes of a ANN and a Fuzzy system. ANN became largely popular, as due to their ability to universally approximate continuous nonlinear function using only the information contained in a set of input/output training pairs. Learning rules are used to adapt the interconnection weights. In other hand, a major criticism of most ANN is their opaque structure where the information stored cannot be easily interpreted by the designer. Fuzzy systems consist of a rule base composed of vague production rules such as: *if* (*input is small*) *then* (*output is small*). The rules are generally linguistic representations and because the information can be easily interpreted by the designer, it is said to be transparent. The power of Fuzzy system lies in the way these production rules are given a precise mathematical meaning so that the resulting system can generalize to produce an appropriate output for previously un seen inputs. Fuzzy systems have a serious drawback when applied to many applications, their rules are often very difficult or even impossible to determine. This has motivated the development of an adaptive FUZZY SYSTEMS, that adjusts their rule base parameters via heuristic training rules about which little can be proved. This is known as the Neuro-Fuzzy shown in Fig. (6).

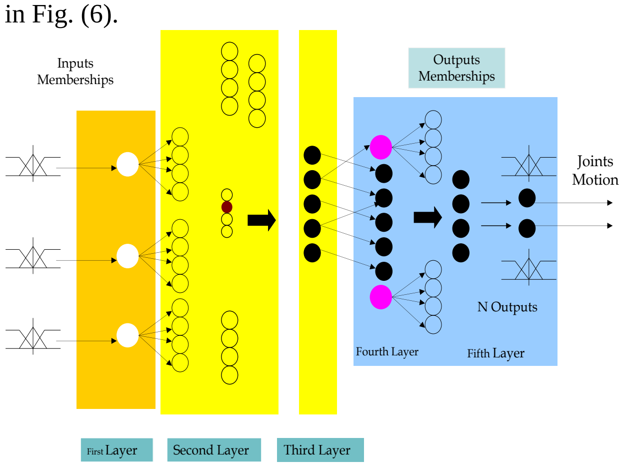


Fig. 6. Fuzzy implementation via cascaded multi-layers ANN (Neuro-fuzzy System).

Recently, the similarities between Neural Networks and Fuzzy systems have been observed, allowing the positive attributes of both approaches to be combine. The result is termed a Neuro-fuzzy system since it embodies the well-established modeling and learning capability of neural network with the transparent knowledge representation of fuzzy systems. To summarize the two approaches for the arm control, this is listed below as:

A. Benefits and Weakness of FUZZY Arm Control.

i. Fuzzy system lets us to compute precise values for arm data points not contained in the training data set i.e. an appropriate output for previously unseen inputs.

ii. Fuzzy rules are generally linguistic representations and because information can be easily interpreted by the designer, it is said to be transparent.

iii. Weakness of fuzzy robot arm Control: Their rules are often very difficult or even impossible to determine.

iv. Benefits of Neural Network Robot Arm Control:

v. The ability to universally approximate any continuous nonlinear function using only the information contained in a set of input/output training pairs.

vi. Weakness of Neural Network Robot Arm Control: The designer cannot easily interpret their opaque structure, where the information stored.

B. The Learning Process.

As mentioned above, the learning process is completed by finding the appropriates weight for each neuron. The initial step toward such learning activity, is to collect training patterns and data from the arm sensors. These data must be collected by moving the arm in the space, and by changing the space variables and reading the joints angle. Since, our work didn't include the space sensor, we collect the data using an approximation equation (using inverse of Jacobian) by referring to arm simulation using the appropriate arm models. However the actual data must come from the sensor.

The below were based on a predefined trajectory of the manipulator arm. The patterns chosen for the training of the neural networks in this work were taken from points in the workspace of the arm, i.e. the area that can be reached by the end effecter of the arm. The arm working space is obtained by considering the robot arm geometry of Fig. 4. For this particular research, the arm working space is a 3-D data, where further tabulation of training patterns will be additionally analyzed at a later stage within this manuscript.

IV. NEURO-FUZZY ARM CONTROL

A. Arm And Motoring System.

In reference to Fig (4), and it was mentioned already in SECTION (II), the UOB robotic arm system has 2-DOF motion in 3-D space. The system is actuated via two high torques and high resolution d.c. motors. Each individual motor is being sensed via a high resolution position potentiometer. This is to measure the joint angular position and angular velocity. In the same sense, each motor has a driving circuitry that work both with analog and digital domains, as will be further explained in details in the following sub-sections.

B. Arm Actuation Closed Loop System.

In an attempt to realize such class of controller and other controllers themes practically, an attempt has been achieved to build a robotic arm system which has been equipped with the right set of motion sensors. This is to be used for implementing advanced robotics control algorithms. In this sense, the UOB arm was implemented with the subsequent closed loop control circuit. The entire system hardware is shown in Fig. (7). It consists of a number of (D/A– A/D) convertors, signals conditioning circuitries, summing points, lead controller and a simple push pull driver for driving the two DC motors.

In addition it consist of a number of digital interfacing circuitries. Both joints has such inartificial circuit structure. The UOB-arm system has two degrees of freedom actuated via two high resolution D.C. motors. Each motor is being sensed via a high resolution position potentiometer. Within same sense, every arm actuator is connected through interfacing buses to PC system as this is illustrated by the hardware shown in Fig (7).

C. Neuro-Fuzzy Training Pattern Generation.

The most familiar or known category of the Neuro-fuzzy is the multilayer ANN that performs fuzzy functions. It consists of five layers, an input layer, three hidden layers, and an output layer. The input layer is made up of sources nodes (sensory units), the second layer is the hidden layer (three layers) of high enough dimensions. Finally, is the output layer that supplies the response of the network to the activation patterns applied to the input layer. The transformation from the input space to the hidden unit space is non-linear. If the input vector is presented each neuron in the hidden layer will output a value (weight) corresponding to how close the input vector is to each neuron's center. Each hidden neuron has an activation function.

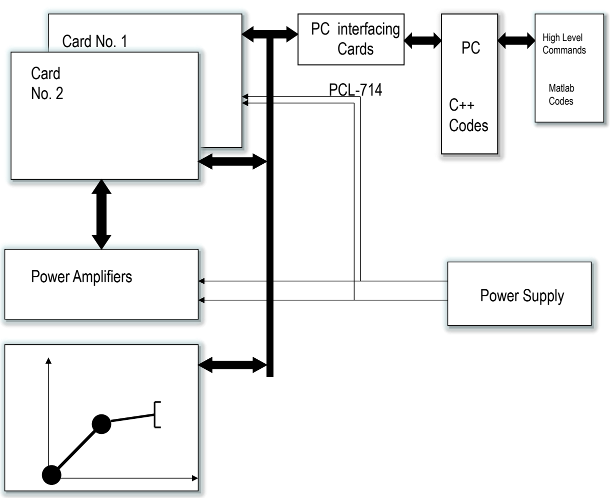


Fig. 7. UOB robotic arm interfacing hardware and associated system layout.

Fig. (8) shows only a segment of a C^{++} coding that was written for interfacing the machine with arm hardware. There are two main cards. Individually, they are paralleled interfaced through specific hardware circuitry. C^{++} coding has been used, as the ability of such coding to communicate with low level ports and the high level Matlab toolboxes and commands. In addition, Fig. (9) shows few segments of the Matlab coding that was employed for the robotic arm control. At such high level, there are a number of defined functions and routines as related to arm kinematics, dynamics, training patterns generation, and the Neuro-fuzzy training itself. Matlab has been used extensively for such robot arm control, as due to the ability of such high level coding to perform the Neuro-fuzzy training and the learning process itself.

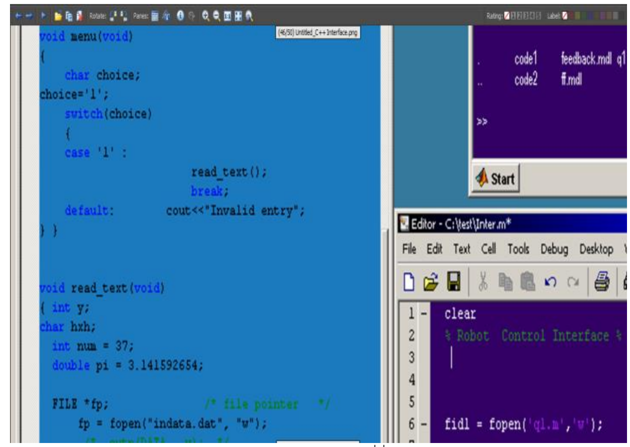


Fig. 8. Arm Hardware and C^{++} low level interfacing windows.

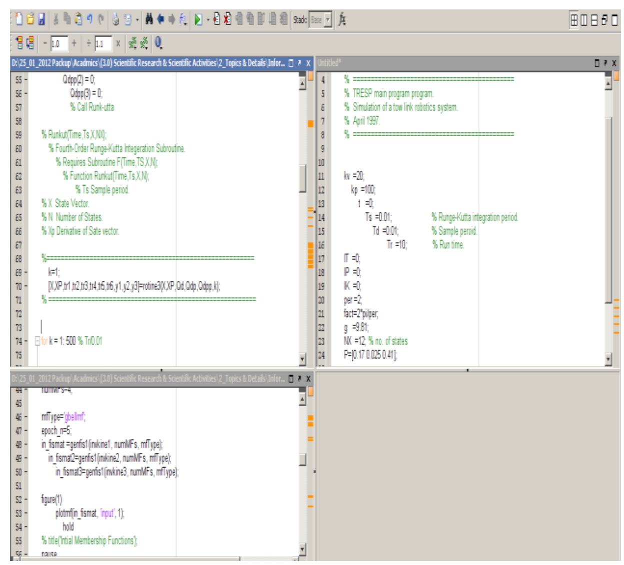


Fig. 9. Generating training patterns. Arm dynamic simulations. Solution of Fourth order Runge-Kutta differnational equation.

This is the overall system structure that have been used for implement the defined Neuro-fuzzy arm control. Few typical Matlab statements and functions used in this research are:

Arm Motion and Dynamics Simulation: The following Matlab coding and segments were written (using approximation) to collect the numerical data required for learning process:

```
[X,XP,tr1,tr2,tr3,tr4,tr5,tr6,y1,y2,y3]=rotine3(X,XP,Qd,Qdp,Qdpp,k);
 for I =1:NX;
        X1(I)=X(I)+0.5*Ts*XP(I);
          end;
 t=t+0.5*Ts;

[X1,XP1,tr1,tr2,tr3,tr4,tr5,tr6,y1,y2,y3]=rotine3(X1,XP1,Qd,Qdp,Qdpp,
k);
          for I =1:NX;
                   XP(I)=XP(I)+2*XP1(I);
                            X1(I)=X(I)+0.5*Ts*XP1(I);
          end;
```

```
[X1,XP1,tr1,tr2,tr3,tr4,tr5,tr6,y1,y2,y3]=rotine3(X1,XP1,Qd,Qdp,Qdpp,
k);
 for I =1:NX;
        XP(I)=XP(I)+2*XP1(I);
           X1(I)=X(I) +Ts*XP1(I);
              end;
 t=t+0.5*Ts;

[X1,XP1,tr1,tr2,tr3,tr4,tr5,tr6,y1,y2,y3]=rotine3(X1,XP1,Qd,Qdp,Qdpp,
k);
 for I =1:NX;
        X(I)=X(I)+Ts*(XP(I)+XP1(I))/6;
          end;
```

D. Neurofuzzy Training Cycle.

```
% To do train

for i=1:791
 ndx1(i) = data1((i),1); ndx2(i) = data1((i),2);
          ndx3(i) = data1((i),3); nx1(i)  = data1((i),4);
                   ndx12(i) = data2((i),1); ndx22(i) = data2((i),2);
                   ndx23(i) = data2((i),3); nx2(i)     = data2((i),4);
end

trndata =[ndx1'    ndx2'    ndx3'];
          trnout  = [nx1 ];
                   trndata2 =[ndx12'    ndx22'   ndx23'];
                            trnout2  = [nx2 ];

Kin_Comp_1=[ndx1'    ndx2'     ndx3'     nx1'];
                   Kin_Comp_2=[ndx12'  ndx22'   ndx23'   nx2'];


numMFs=4;
          mfType='gbellmf';
                   epoch_n=5;
                   in_fismat     =genfis1(Kin_Comp_1,     numMFs,
                   mfType);

in_fismat2=genfis1(Kin_Comp_2, numMFs, mfType)
                                                                  %
                                                          End
```

V. RESULTS AND DISCUSSION

Having presented a detailed description of the overall system structure hardware and software, within this section we shall be presenting few results. In this context, after the extensive mathematical kinematics and dynamic modeling of the UOB robotic arm system, the following stages were followed for validating the proposed control methodology.

i. *Step_1*: This involves the arm kinematics and dynamics mathematical modeling and expression in terms of a defined Cartesian coordinate frame located at the arm base.

ii. *Step_2*: Within this step, the robotic arm system was dynamically and kinematically simulated using Matlab. Over this stage, large number of training data and patterns where therefore gathered and tabulated. The simulation has taken all arm dynamics and other effects in consideration. Such resulting movements are shown in Fig. (10). Gathered training patterns are also listed in Table (1). Smoothness can be seen among the collected data, i.e. indicating how the arm was moving in periodic sinusoidal fashion.

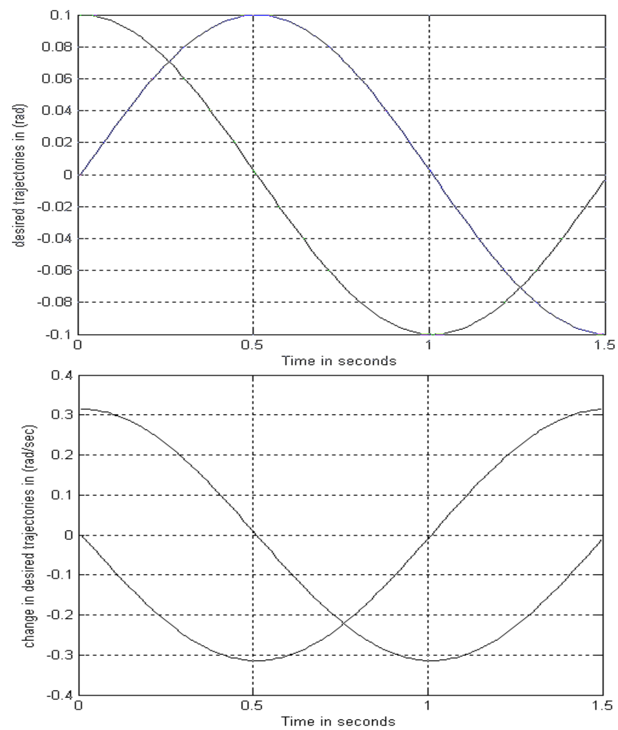


Fig. 10. Training patterns generation. Simulating the robotic arm. (*i*) Joint-space arm motion. (*ii*) Joint-space rate of change.

iii. *Step_3*: Once the training patterns are available, a five layers ANN that can implement the *fuzzy if then* rules is used for learning the relations between arm posture in 3-D space and the correspondence arm joints displacements movements.

iv. *Step_4*: Adjustment of fuzzy memberships. This gives an indication that the fuzzy system has learned relations relating arm motion to the corresponding $P_{(x,y,z)}$ postures.

Table 1. Part of generated training patterns.

X	*Y*	*Z*	*θ1*	*X*	*Y*	*Z*	*θ2*
1.8503	-0.3600	-0.3600	-0.1033-	0.6508	-0.2744	-0.3600	-0.0178
-0.1033	-0.3600	-0.2744	-1.8503	-0.0178	-0.3600	-0.1889	-0.4827
-0.1033	-0.3600	-0.1889	-1.8503	-0.0178	-0.3600	-0.1033	-0.2792
-0.1033	-0.3600	-0.1033	-1.8503	-0.0178	-0.3600	-0.0178	-0.0493
-0.1033	-0.3600	-0.0178	-1.8503	-0.0178	-0.3600	0.0678	0.1859
-0.1033	-0.3600	0.0678	-1.8503	-0.0178	-0.3600	0.1533	0.4022
-0.1033	-0.3600	0.1533	-1.8503	-0.0178	-0.3600	0.2389	0.5853
-0.1033	-0.3600	0.2389	-1.8503	-0.0178	-0.3600	0.3244	0.7329
-0.1033	-0.3600	0.3244	-1.8503	-0.0178	-0.3600	0.4100	0.8496
-0.1033	-0.3600	0.4100	-1.8503	-0.0178	-0.2744	-0.3600	-0.9184
-0.1033	-0.2744	-0.3600	-1.9309	-0.0178	-0.2744	-0.2744	-0.7844
-0.1033	-0.2744	-0.2744	-1.9309	-0.0178	-0.2744	-0.1889	-0.6018

-0.1033	-0.2744	-0.1889	-1.9309	-0.0178	-0.2744	-0.1033	-0.3594
-0.1033	-0.2744	-0.1033	-1.9309	-0.0178	-0.2744	-0.0178	-0.0646
-0.1033	-0.2744	-0.0178	-1.9309	-0.0178	-0.2744	0.0678	0.2416
-0.1033	-0.2744	0.0678	-1.9309	-0.0178	-0.2744	0.1533	0.5086
-0.1033	-0.2744	0.1533	-1.9309	-0.0178	-0.2744	0.2389	0.7152
-0.1033	-0.2744	0.2389	-1.9309	-0.0178	-0.2744	0.3244	0.8677
-0.1033	-0.2744	0.3244	-1.9309	-0.0178	-0.2744	0.4100	0.9800
-0.1033	-0.2744	0.4100	-1.9309	-0.0178	-0.1889	-0.3600	-1.0858
-0.1033	-0.1889	-0.3600	-2.0714	-0.0178	-0.1889	-0.2744	-0.9659
-0.1033	-0.1889	-0.2744	-2.0714	-0.0178	-0.1889	-0.1889	-0.7832
-0.1033	-0.1889	-0.1889	-2.0714	-0.0178	-0.1889	-0.1033	-0.4987
-0.1033	-0.1889	-0.1033	-2.0714	-0.0178	-0.1889	-0.0178	-0.0934
-0.1033	-0.1889	-0.0178	-2.0714	-0.0178	-0.1889	0.0678	0.3431

Fig. (10) shows a typical full arm simulated joint space motion. This is based on the full arm dynamic model already derived in SECTION (III). For such a robotic arm, as also in other similar designs and situations, this involved a solution of a four nonlinear state space model for both (θ_1, θ_2). After simulated arm model was achieved, then training data are gathered and collected. Normalized sample data collected for both joints and the $P_{(x,y,z)}$ arm (x,y,z) posture, were collected as on Table (1). The arm model parameters are as listed here: Weights of link_1 and link_2 are: $(m_1 = 0.4kg)$ and $(m_2 = 0.3kg)$. Lengths of link_1 and link_2 are: $\ell_1 = 0.3m$ and $\ell_2 = 0.25m$. The defined trajectories to the system are, ($A_m = 0.1$):

$$\theta_d(1) = A_m \sin(0.5\pi\times t),\ \theta_d(2) = B_m \cos(0.5\pi\times t) \tag{22}$$

$$\dot{\theta}_d(1) = A_m \frac{\pi}{2}\cos(0.5\pi\times t), \dot{\theta}_d(2) = -A_m \frac{\pi}{2}\sin(0.5\pi\times t) \tag{23}$$

$$\ddot{\theta}_d(1) = -A_m\left(\frac{\pi}{2}\right)^2 \sin(0.5\pi\times t),\ \ddot{\theta}_d(1) = -A_m\left(\frac{\pi}{2}\right)^2 \cos(0.5\pi\times t) \tag{24}$$

Collected training patterns are then used for training the designed Neuro-fuzzy system. Initial fuzzy memberships were used at the starting stage prior to training. Subsequently, after training, the shape of such adopted memberships were updated. This can be observed evidently in Fig. (11). This gives an indication that the used Neuro-fuzzy system has learned the defined relations. In addition, Fig. (12) shows 3-D maps governing the adopted Neuro-fuzzy rules, whereas Fig. (13) shows the 3-D map of the learned Neuro-fuzzy system, i.e. the plot of the fired (*fuzzy if then rule*). Using the Cartesian data as inputs to the Neuro-fuzzy, the joints angle will be outputted from the Neuro-fuzzy and it will be used to solve for the required joints positions. This is also shown in Fig. (14).

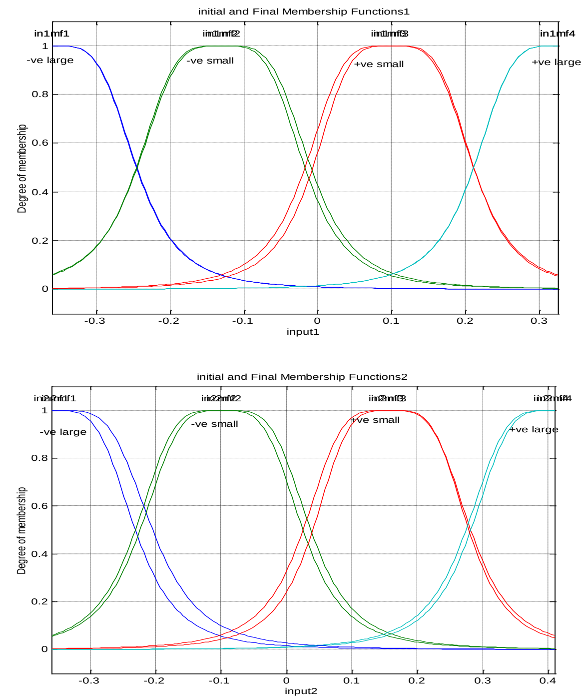
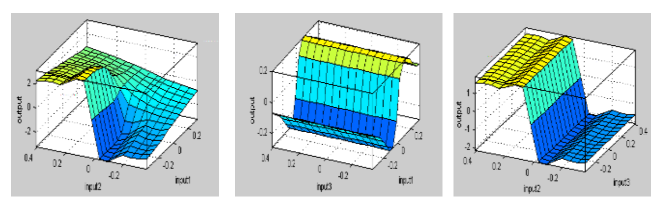


Fig. 11. Initial (actual) and final (learned) fuzzy membership functions for input 1 & 2 of our system.



(*i*) (*ii*) (*iii*)

Fig. 12. 3-D plots of relations between various fuzzy inputs, and fuzzy outputs. i.e. (θ_1, θ_2).

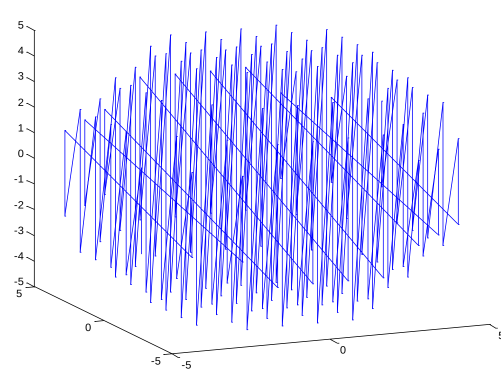


Fig. 13. A 3-D map of the learned Neuro-fuzzy system. The 3-D plot is displaying how actively (*fuzzy if then rule*) are fired for different values of the input training sample.

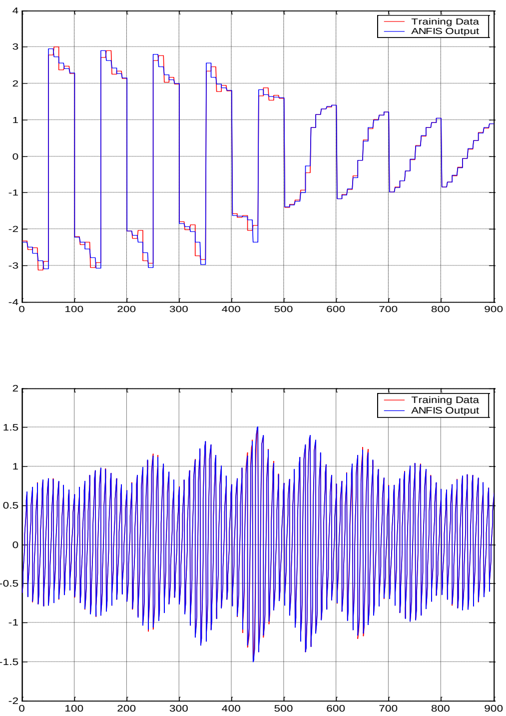


Fig. 14. Verification of the learned Neuro-fuzzy system.
(Top): Output for a randomly excited square wave.
(Bottom): Output for a randomly excited sinusoidal wave.

After the learning phase, the adopted and trained Neuro-fuzzy system is ready to be used within the control loop. In this respect, Fig. (15) shows a schematic diagram for the verification stage. In particular to Fig. (15), the “Jacobian inverse” is playing a key role with the forward loop, whereas in Fig. (16), we show the practically implemented controller design verification in action. This show how the learned Neuro-fuzzy system was used within the forward loop, hence it is replacing the “Jacobian inverse”. Fig. (17) shows the robotic arm motions while using the Neuro-fuzzy maps within the forward controller path. The robotic arm has been under experimental trails a number of times. During such trails, the arm has shown accurate and precise motion in 3-D. For real verifications, the arm physical motions were recorded while the arm was in motion. Such snaps are also filmed as in Fig. (18).

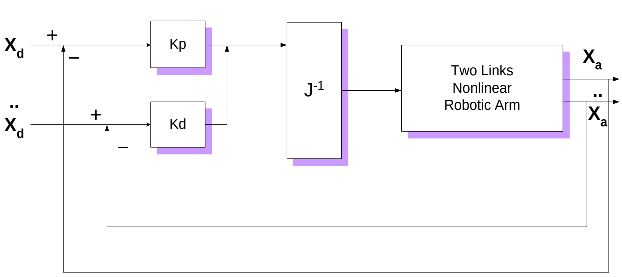


Fig. 15. “Jacobian inverse” controller design verification. A schematic of a non-model based robotic arm control in Cartesian space.

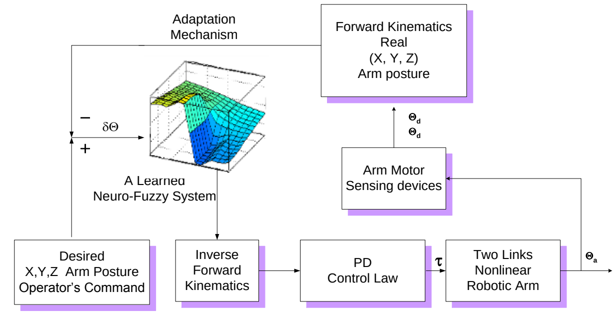


Fig. 16. Learned Nuero-fuzzy mapping verification. Insertion of the trained Neuro-fuzzy system is within the controller forward loop.

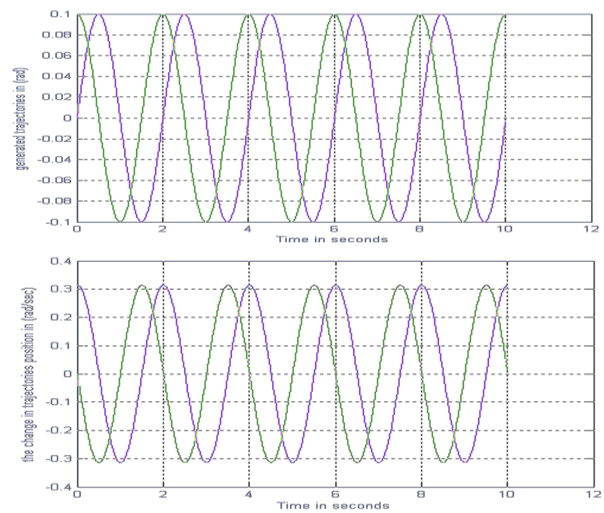


Fig. 17. Arm joint space motion control, as in reference to the controller scheme of Fig. (16).

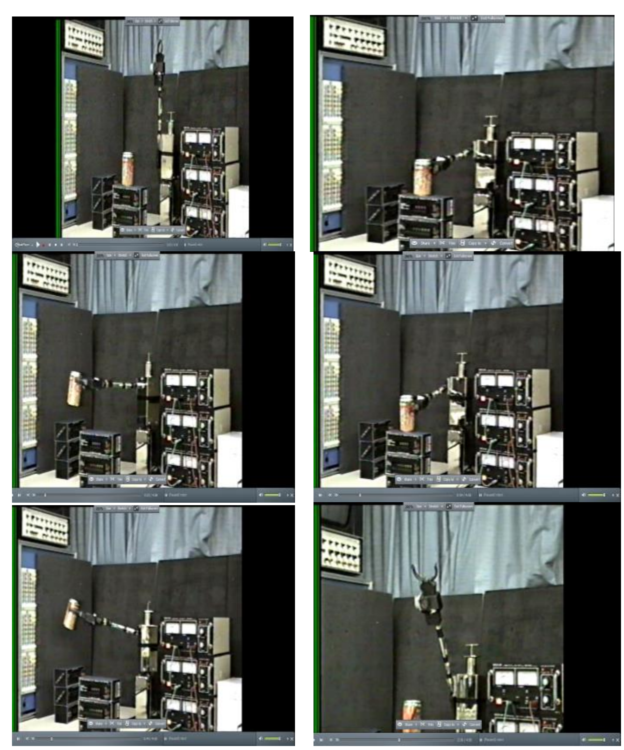


Fig. 18. Snaps of recorded UOB arm video motion.

VI. CONCLUSION

This research was focused towards two parts. The first part was completely dedicated towards designing and physically building a two DOF robotic arm with two, with its associated hardware. The second part was directed towards control synthesis, purposes and an employing a Neuro-fuzzy learning system for controlling such a locally designed robotic arm system. This involves an availability of very precise robot arm kinemtics and dynamics models, hence to verify such models.

The arm kinemtics and dynamics models are to be essentially available for control. This allowed to simulate the arm motion over time, hence to generate the precise size of the training samples. It also involves an arrangement of the control system hardware, i.e. the interfacing electronics and the manipulator low level control electronics and convertors. Results have indicated that, the adopted Neuro-fuzzy learning mechanism was an excellent approach to approximate the nonlinear maps that exist between arm joint-space motion and the arm end point displacements. Training the employed Neuro-fuzzy was a real issue. This is due to the complexity of such learning architectures. As the training samples was increased, the Neuro-fuzzy needs more time to learn the right maps. Arm control was after then achieved using the created maps relating arm joint-space displacements to the arm end point motions. Results have indicated high degree of accuracy of arm point-end motion.

It is worth to mention that, the first part of this research (i.e. building the arm physical system), was indeed an excellent experience for the UOB control laboratory. This is due to the involvement of a number of design stages from the mechanical setup to the actuation and computer interfacing. The second part was also furthermore interesting, (i.e. model based control). Neuro-fuzzy system for inverse kinematics has shown good degree of accuracy, rather than inverting the UOB arm Jacobian. The built UOB arm will furthermore be used for even advanced control methodologies within the Control Laboratory at UOB, especially within the extent of intelligent control.

REFERENCES

[1] R. S. Hartenberg and Denavit, "A Kinematics Notation for Lower Pair Mechanisms Based on Matrices," *International Journal of Applied mechanics*, vol. 77, pp. 215-221, 1955.

[2] A. A. Bogdanov, and A. V. Timofeev "Robust optimal neural control of robots". Proceedings of the IEEE IJCNN'-99, *International Joint Conference on Neural Networks*, vol. 3, 1999.

[3] F. M. De Azevedo and J. M. Barreto, "IMC Scheme Using Neural Networks for Robot Arm," *Proceedings of the CBA 6th Congress Latinoamericano de Controle Automático*, 2010.

[4] S. J. Hsia, "On an Effective Design Approach of Cartesian Space Neural Network Control for Robot Manipulator," *International Journal of Robotica*, vol. 15, pp. 305-312, 1997.

[5] V. Lakshmi, U. Mashuq, "An adaptive Neuro-Fuzzy control approach for motion control of a robot arm," *(ICIEV) International Conference on Informatics, Electronics & Vision*, vol. 1, pp.832-836, 18-19 May, 2012.

[6] J. Tavoosi, M. Alaei, and B. Jahani, " Neuro–Fuzzy Controller for Position Control of Robot Arm," Paper Reference No. :(0113-795), *Proceedings of the 5th Symposium on Advances in Sciences and Technology*, Iran, May 12-17, 2011.

[7] D. T. Pham, and A. A. Fahmy. "Neuro-Fuzzy Modelling and Control of Robot Manipulators for Trajectory Tracking," *The 16th IFAC WORLD CONGRESS*, 2005.

[8] Lazarevska, E., "A Neuro-fuzzy Model of the Inverse Kinematics of a 4 DOF Robotic Arm," *2012 UKSim 14th International Conference on Computer Modeling and Simulation (UKSim)*, vol. 1, pp. 306-311, 28-30 March, 2012.

[9] H. Pham Huy Anh, K. Kwan Ahn, and Y. Jong Il, "Dynamic model identification of the 2-Axes PAM robot arm using Neural MIMO NARX model," *Proceedings of the ICCE 2008. 2nd International Conference on Communications and Electronics*, vol. 1, pp.18-23, 4-6 June, 2008.

[10] W. Kelly, R. Challoo, R. Mclauchlan, S. Iqbal, "Neuro-fuzzy Control of a Robotic Arm", *Proceedings of the Artificial Neural Networks In Engineering Conference, St. Louis, MO*, November 10-13, 1996, pp. 837-842, 1996.

[11] E. Mattar, "UOB-Two DOF Robotic Arm," College of Engineering, University of Bahrain. *UOB-Arm Internal Report*. 2003.

Ebrahim Mattar is an Associate Professor of Intelligent Control and Robotics at University of Bahrain. He has received BSc. in Electrical Engineering (from University of Bahrain in 1986), done MSc. in Electronics in 1989 (from University of Southampton, UK), and in 1994 he received University of Reading Ph.D. in Cybernetics and Robotics Control. Dr. Mattar has interests in Computational Intelligence, Robotics Control, Modeling and Control. This includes clustering with fuzzy, Neural Networks, Evolutionary Computation, and their real applications in Robotics and Control. Mattar is a member of a number of professional societies locally and internationally. Locally, he is a member of Bahrain Society of Engineers (BSE), board member of the

Academic Society in Bahrain, board member of the Technology Transfer Society. Internationally, Mattar is a member of the IEEE, member of IEEE Control Educational Society, IEEE Robotics and Automation Society (RAS), an active IET member, IET – Bahrain Network Honorary Chair, X-IET EMEA Regional Board Member, X-member of IET Knowledge Programme Advisory Member, and member of some control societies world-wide. Dr. Mattar has published a number of journal and conferences articles in the area of dexterous robotic hands, Control, Optimal robotic hand forces, Neural multi-finger robot hand grasping and control, optimal fuzzy control, and Neuro-fuzzy systems, intelligent control. Dr. Mattar has been a responsible body for two major conferences in Bahrain, including the IET ICIS-2008 (International Conference on Intelligent Systems, December 2008), and a responsible body for three IET symposiums in Bahrain, including (New Directions in Automatic Control: Theories and Applications, April 2010. Dr. Mattar is also an ABET accreditation expert, as has been leading a team for a positive full Electrical and Electronics Engineering Programs accreditation over the period from 2005-2010, for College of Engineering at Bahrain University. Elected Honorary Chair for the IEE and IET for two Sessions, Bahrain Local Network Committee. . Organizing Head for the International Conference on (Millennium Dawn in Training and Continuing Education), 16-18th of April 2001. A responsible organizer for a Power Forum : (Trends Toward Power System Networks Enhancement), 26th February. 2008. A responsible organizer for a Communication Forum : Challenges and Trends in Modern Communication), 7th May 2008. A responsible organizer for an IET International Conference on Intelligent Systems (ICIS-2008), 1-3rd December 2008. A responsible organizer for an IET Control Symposium: (New Directions in Automatic Control: Theories and Applications), 26th of April 2010. Chairing Continuing Engineering Education Program, University of Bahrain (1998-2002). Chairing Electrical and Electronics Engineering Department, University of Bahrain (2004-2009), also seconded from UOB as the Director General of the Bahrain Training Institute (2011-2012).