Three Link Rigid Manipulator Control Using Improved Neural Network Based PID Controller

Sherif G. Ahmad
Faculty of Engineering, Mansoura University, Egypt
Email:sanafer84@yahoo.com

Mohamed A.El-Gohary
Assistant Professor, Department of Mechanical Engineering, Faculty of Engineering, Alexandria University, Egypt
Email:melgohary@gmail.com

Mohamed S. Elksas
Assistant Professor Department of Computers & Systems Engineering, Mansoura University, Egypt
Email:msmksasy@gmail.com

Fayez G. Areed
Professor at Computers & Systems Engineering, Mansoura University, Egypt

***Abstract*—**This paper presents an artificial neural network based pid controller of a three link rigid manipulator. We develop neural network control algorithms to solve the nonlinear problems for compensating robot manipulator control with uncertainties so that accurate position could be achieved. The back propagation algorithm has been used for training a two layered feedforward artificial neural network. Our proposed controller is simply combining the ANN with other conventional control method and provide the network with more data about the structure and the behavior of the system, the neural network is trained with the data generated by pid controller. The simulation result shows that the controller works well and performs better than the conventional PID.

***Keywords*—**Dynamic modelling, Three link rigid manipulator, Lagrange-Euler, PID controller, ANNC

I. INTRODUCTION

Robot manipulators, referred to as robotic arms, perform operations that require among others high precision, high speed, continuous work, manipulating heavy payloads or dealing with bio-hazardous materials or working in hazardous environments. Robotic arms have a wide applicability, since they resemble human arms in many functional aspects. They are vastly used in industrial manufacturing, doing tasks like pick and place, surface finishing, assembly operations, drilling, palletizing or welding [1]. The physical assembly is made of rigid links connected by mobile joints. The type of joint used to connect links might be: prismatic joint, revolute joint, cylindrical joint, spherical joint [2]. With the knowledge of kinematics and dynamics of a serial link manipulator, we would like to servo the manipulator's joint actuators to achieve a desired task by controlling the manipulator to move to a pre-specified coordinates or to follow a desired path. One of the important parts that defines the accuracy and repeatability of a robot is the manipulator controller. To design the controller, the parameters of the systems is needed as well as solving complex equations which demands time and engineering knowledge. The neural approach was made up when the conventional and ordinary ways to control systems failed to tackle the problems such as vision, speech and pattern because real world cannot be represented in mathematical expressions. Artificial neural networks (ANN), computational models of the brain, are vastly utilized in engineering applications because of their capability to assess the relation between inputs and outputs from a learning process. Motivated by the seminal paper [3]. There exists a constantly raising interest in applying neural networks (NN) to identification and control of nonlinear systems. Most of these applications use feedforward structures [4]. This is due to NN universal function approximation property.

NNC can be classified as non-parametric controllers in the sense that they are not parameterized in terms of system parameters [5].

Also it does not need persistence of excitation and certainty of equivalence in case it is designed correctly.

After introduction of ANN and its growing applications in the control area, many articles were published about the use of the ANN as the controller of a robot. A controller with a very simple structure was proposed in [6]; the whole controller was an ANN and was getting feedback from joint angles of the robot. the inverse dynamics of the robot was solved with two ANN, and the structure of the controller was just like the feedback linearization method in control theory. the only difference was that the equations that solve the robot inverse dynamics were substituted with two ANNs [7]. An adaptive controller with an ANN is introduced [8] which works in the Cartesian space. A controller was proposed to compensate for the structured and unstructured uncertainties in the robot model with combining the computed torque method and ANN [9]. An efficient method for the controlling the robot is proposed and that is production of a new path as a desired path [10]. The performance seems to be better than what is mentioned in [7] and the controller converges quite fast to the desired input. Akio Ishiguro trained ANN such that the network output compensates for the error between the real system and the model. an efficient method for controlling the manipulator is introduced and the ANN is used to regenerate a desired path and feed this path to a computed torque control system such that the output of the system traces the desired input [11]. In [12] mentioning the capability of using ANN as a controller for robotic manipulators and compared the ANN with adaptive control method. Despite both adaptive and ANN controllers demonstrate good performance, they proved that a NN with two linear layer is equal to an adaptive controller, while by use of a three-layer neural network with nonlinear function for the second layer output and a linear function for the third layer output the ANN shows better performance in systems with high nonlinearities. Tetsoro showed that the stability using the backpropagation (BP) method depends on both the initial value of the weight vector and the gain tuning parameters. That is, the (BP) method cannot guarantee the stability by itself and we have to find the quantitative stability condition by trial. In that article there is a comparison between the adaptive and ANN controllers. "An ANN with linear output function has identical structure of the adaptive controller but a three layer ANN with nonlinear output function could demonstrate better performance in event of nonlinearities." He mentions in his article. He suggests that an ANN controller should be utilized if the nonlinearity of the system can't be neglected.

The methods that are mentioned earlier have their own pros and cons. Our proposed controller is simply combining the ANN with other conventional control method and provide the network with more data about the structure and the behavior of the system. We simply train the NN using the data generated from the conventional controller (PID), selecting suitable number of neurons in the hidden layer, we get better response than that of the PID controller.

This paper is organized as follows: Section II presents system model, conventional PID controller. Section III introduces the neural network controller design, simulation results and Section IIII illustrates conclusion and future work.

II. System model and PID controller

We consider the three-link robotic manipulator. The physical system is shown in fig. (1). The system consists of three masses connected by weightless bars. The bars have length d_1, d_2 & d_3. Let θ_1, θ_2 & θ_3 denote the angles.

The dynamic equations are derived by the Lagrangian Euler (L-E) formulation for a three–links robot manipulator system with revolute joints. All the rotation axes at the joints are along the z-axis perpendicular to the plane of the paper. The mass of the three links m_1, m_2 and m_3 are represented by point masses at the end of the links (weightless bars). The load mass is represented by m3 and is supposed to be at the end of the link 3.

By applying the Lagrange function to the robot arm yields the necessary generalized torque $\boldsymbol{\tau}_i$ for joint i to drive the i-th link of the manipulator [13], for i = 1, 2, 3 which gives:

$$\tau_1 = [(m_1 + m_2 + m_3)\, d_{12} + (m_2 + m_3)\, d_{22} + m_3\, d_{32} + 2\,(m_2 + m_3)\, d_1\, d_2 \cos\theta_2 + 2\, m_3\, d_2\, d_3 \cos\theta_3 + 2\, m_3\, d_1\, d_2 \cos(\theta_2 + \theta_3)]\, \ddot{\theta}_1 + [(m_1 + m_2)\, d_{22} + m_3 d_{32} + (m_2 + m_3)\, d_1\, d_2 \cos\theta_2 + 2\, m_3\, d_2\, d_3 \cos\theta_3 + m_3\, d_1\, d_2 \cos(\theta_2 + \theta_3)]$$

$\ddot{\Theta}_2 + [m_3 d_{32}+m_3 d_2 d_3 \cos\theta_3+m_3 d_1 d_2 d_3 \cos(\theta_2+\theta_3)]\ddot{\Theta}_3 - [(m_2 + m_3) d_1 d_2 \sin\theta_2+ m_3 d_1 d_2 \sin(\theta_2 + \theta_3)]\dot{\theta}_{22} - [m_3 d_2 d_3 \sin\theta_3 + m_3 d_1 d_2 \sin(\theta_2 + \theta_3)]\dot{\theta}_{32} - [2(m_2 + m_3) d_1 d_2 \sin\theta_2 + 2 m_3 d_1 d_2 \sin(\theta_2 + \theta_3)]\dot{\theta}_1\dot{\theta}_2 - [2 m_3 d_2 d_3 \sin\theta_3 + 2 m_3 d_1 d_2 \sin(\theta_2 + \theta_3)]\dot{\theta}_1\dot{\theta}_3 - [2 m_3 d_2 d_3 \sin\theta_3 + 2 m_3 d_1 d_2 \sin(\theta_2 + \theta_3)]\dot{\theta}_2\dot{\theta}_3 + [(m_1 + m_2 + m_3) d_1 g \sin\theta_1+ (m_2 + m_3) d_2 g \sin(\theta_2 + \theta_1)+ m_3 d_3 g \sin(\theta_1 + \theta_2 + \theta_3)$ (1)

$\tau_2 = [(m_1 + m_2) d_{22} + m_3 d_{32} + (m_2 + m_3) d_1 d_2 \cos\theta_2+ 2 m_3 d_2 d_3 \cos\theta_3 + m_3 d_1 d_2 \cos(\theta_1 + \theta_3)]\ddot{\Theta}_1+ [(m_2 + m_3) d_{22} + m_3 d_{32} + 2 m_3 d_1 d_2 \cos\theta_3]\ddot{\Theta}_2 + [m_3 d_{32} + m_3 d_2 d_3 \cos\theta_3]\ddot{\Theta}_3+ [(m_2 + m_3) d_1 d_2 \sin\theta_2 + m_3 d_1 d_2 \sin(\theta_2 + \theta_3)]\dot{\theta}_{12} - [m_2 d_2 d_3 \sin\theta_3]\dot{\theta}_{32} - [2 m_3 d_2 d_3 \sin\theta_3]\dot{\theta}_1\dot{\theta}_3 - [2 m_3 d_1 d_2 \sin\theta_3]\dot{\theta}_2\dot{\theta}_3+ [(m_2 + m_3) d_1 g \sin(\theta_1 + \theta_2) + m_3 d_3 g \sin(\theta_1 + \theta_2 +\theta_3)]$

(2)

$\tau_3 = [m_3 + d_{32} + m_3 d_2 d_3 \cos\theta_3 + m_3 d_1 d_2 \cos(\theta_2 + \theta_3)]\ddot{\Theta}_1 + [m_3 d_{32} + m_3 d_2 d_3 \cos\theta_3]\ddot{\Theta}_2+ m_3 d_{32}\ddot{\Theta}3+ [m_3 d_2 d_3 \sin\theta_3 + m_3 d_1 d_2 \sin(\theta_2 + \theta_3)]\dot{\theta}_{12} + [m_3 d_1 d_2 \sin\theta_3]\dot{\theta}_{22} - [2 m_3 d_2 d_3 \sin\theta_3]\dot{\theta}_1\dot{\theta}_2 + m_3 d_3 g \sin(\theta+\theta_2+\theta_3)$ (3)

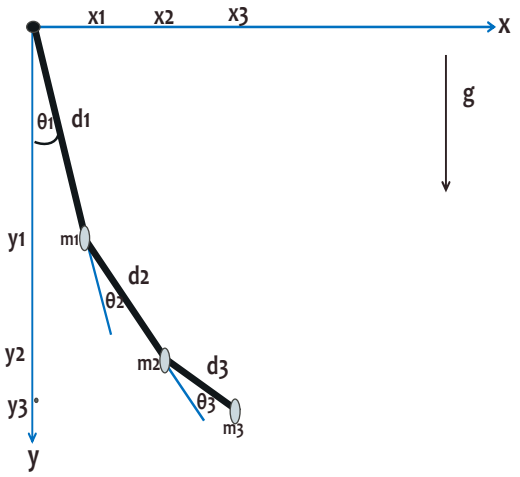


Fig. 1. Three links manipulator

The general form of equations of motion can be written as

Table 1. System description

Description	Notation
Length of link 1	d_1
Length of link 2	d_2
Length of link 3	d_3
Mass of d1	m_1
Mass of d2	m_2
Mass of d3	m_3
Gravitational acceleration	g
Angle of d1	θ_1
Angle of d2	θ_2
Angle of d3	θ_3

$$\boldsymbol{\tau} = M(q)\ddot{q} + V(q,\dot{q}) + G(q) \quad (4)$$

Where, q is the generalized joint coordinates

$M(q)$ is the mass matrix (inertia matrix)

$V(q,\dot{q})$ is the centrifugal & Coriolis forces

$G(q)$ is the gravity forces

τ is the generalized forces. (torques applied to the robot)

$$\text{And} \quad q = \begin{bmatrix}\theta_1\\ \theta_2\\ \theta_3\end{bmatrix} \quad (5)$$

$$M(q) = \begin{bmatrix}D_{11} & 0 & 0\\ 0 & D_{22} & 0\\ 0 & 0 & D_{33}\end{bmatrix} \quad (6)$$

$$V(q\,\dot{q}) = \begin{bmatrix}-D_{122}\dot{\theta}_2^2 - D_{133}\dot{\theta}_3^2 - D_{122}\dot{\theta}_1\dot{\theta}_2 - D_{133}\dot{\theta}_1\dot{\theta}_3 - D_{123}\dot{\theta}_2\dot{\theta}_3\\ D_{233}\dot{\theta}_3 - D_{213}\dot{\theta}_1\dot{\theta}_3 - D_{223}\dot{\theta}_2\dot{\theta}_3 + D_{211}\dot{\theta}_1^2\\ D_{311}\dot{\theta}_1^2 + D_{322}\dot{\theta}_2^2 + D_{312}\dot{\theta}_1\dot{\theta}_2\end{bmatrix} \quad (7)$$

$$G(q) = [D_{311}(\dot{\theta}_1)^2 + D_{322}(\dot{\theta}_2)^2 + D_{312}\dot{\theta}_1\dot{\theta}_2] \quad (8)$$

By solving the function $\ddot{q}$, we get

$$\ddot{q} = M(q)^{-1}[-V(q\,\dot{q}) - G(q)] + F \quad (9)$$

$$\text{Where,} \quad F = M(q)^{-1}\boldsymbol{\tau} \quad (10)$$

The physical torque inputs to the system are

$$\begin{bmatrix} \boldsymbol{\tau}\,\theta_1 \\ \boldsymbol{\tau}\,\theta_2 \\ \boldsymbol{\tau}\,\theta_3 \end{bmatrix} = M\,(q) \begin{bmatrix} \boldsymbol{\tau}_1 \\ \boldsymbol{\tau}_2 \\ \boldsymbol{\tau}_3 \end{bmatrix} \quad (11)$$

The controller for any joint would be

$$Fi = k_{pi}\, e\,(t) + k_{vi}\, \dot{e} + k_{ii} \int e\, dt \;, i = 1, 2, 3 \quad (12)$$

The error (e) signals general form is

$$e\,(\theta_i) = \theta_{if} - \theta_i \;, \quad (13)$$

Where θ_{if} is the target position.

Then the complete system equations would be [13]

$$\begin{bmatrix} \ddot{\theta}_1 \\ \ddot{\theta}_2 \\ \ddot{\theta}_3 \end{bmatrix} = M\,(q)^{-1} * [-V\,(\dot{q}\,q\,) - G\,(q)\,] + \begin{bmatrix} k_{p1}\,(\theta_{1f} - \theta_1) - k_{d1}\,\dot{\theta}_1 + k_{i1}\,x_1 \\ k_{p2}\,(\theta_{2f} - \theta_2) - k_{d2}\,\dot{\theta}_2 + k_{i2}\,x_2 \\ k_{p3}\,(\theta_{3f} - \theta_3) - k_{d3}\,\dot{\theta}_3 + k_{i3}\,x_3 \end{bmatrix} \quad (14)$$

$$\begin{bmatrix} \tau\theta_1 \\ \tau\theta_2 \\ \tau\theta_3 \end{bmatrix} = M\,(q) \begin{bmatrix} k_{p1}\,(\theta_{1f} - \theta_1) - k_{d1}\,\dot{\theta}_1 + k_{i1}\,x_1 \\ k_{p2}\,(\theta_{2f} - \theta_2) - k_{d2}\,\dot{\theta}_2 + k_{i2}\,x_2 \\ k_{p3}\,(\theta_{3f} - \theta_3) - k_{d3}\,\dot{\theta}_3 + k_{i3}\,x_3 \end{bmatrix} \quad (15)$$

This is a second order non-linear system; we have to convert it to a first order system by using state space. By choosing a proper set of state variables, complex systems may be brought to a more convenient form (state-space form), which only requires solving first order ODE's in matrix form [14].

III. Neural Network Controller Design

A. Limitations of PID

Although a PID controller provides an optimum solution to various processes, it is not an antidote to every control problem which could be encountered. This is specifically valid for processes with ramp-style changes in set-point values or slow disturbances [15].

So, PID controller is easy to implement but its performance is not optimized because the robot arm is not a linear system & considering all the torques that are produced by dynamics of the system such as noise will result in shaky performance of the system & sometimes even instability. The problem for the ANN is to solve the system's inverse dynamics. For training the ANN to learn this function it is possible to train it with random data or use the controller to provide good examples for training the ANN.

B. Controller Structure (ANN)

In other words, the ANN can work alone as the controller or it could get used in parallel with other controllers. The idea is to construct a controller with the simplest structure and observe the performance of the system. If the results are not good, we can increase the intricacy of the controller and simplify the problem to be solved by the ANN.

The problem being that the research space for finding the proper weights for the neurons is too wide and the ANN inputs make a very wide domination. To narrow this search space, we used the previous shown controller (pid) to trace the desired path to act as a supervisor for the ANN, as shown in fig. (2).

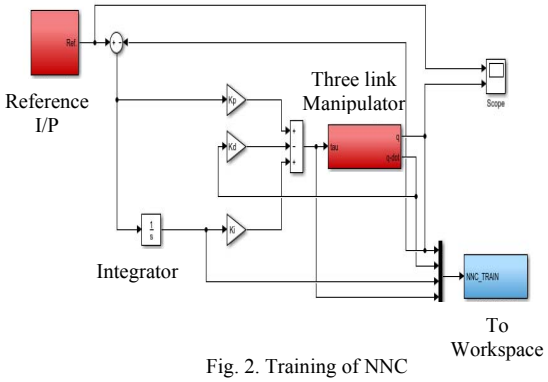


Fig. 2. Training of NNC

The PID controller accepts the angle and angular velocities of joints and generates the required torque for the motors to move the joints. Although this controller does not perform very well, the data which is collected from it are very good samples for training the ANN. After training ANN with samples from PID controller, the ANN takes over PID controller giving magnificent results. The sampled data from the PID controller are used to train the ANN as shown in fig (3).

With this configuration the ANN tries to solve inverse dynamics of the system within a close range to the desired path and generates the required torques to be applied to move the motors for the manipulator follows the desired path. Mostly the manipulator is having a repetitive job and the path that the joints are

moving is periodic. If we define one period of the manipulator path as one cycle the ANN is trained in every cycle and the network identifies the manipulator behavior in more detail. This means that the manipulator's performance is improved and the error between the manipulator trajectory and the desired path limits to zero. Simulation results shows that the error converges to zero and it can be reduced to arbitrary value with enough no. of trainings.

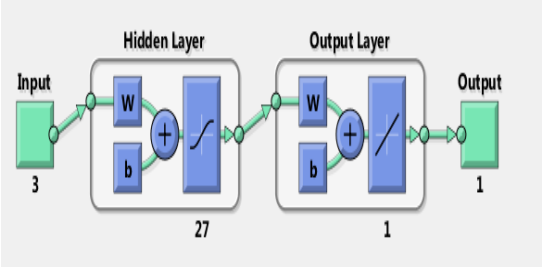


Fig. 3. training window of neurons

C. ANN STRUCTURE

For constructing the ANNC for the manipulator with 3 DOF we used simple yet powerful structure. We used one ANN for every joint of the manipulator, that is for simplifying the inverse dynamic problem. In other words, every ANN outputs one torque for only one joint motor and for n DOF manipulator the ANNC consists of n separate networks. The i/p for every network is angular velocity joint angle and integration of the angle error. Another candidate for the structure of the manipulator is only having one i/p per joint which could be the joint angle and construct the network as a dynamic network with delays in the first layer in order to reconstruct the angular velocity and acceleration of every joint. The drawback of such construction is that the sample time of the controller loop must be fixed and if the controller frequency is changed we would have to retrain the network from the beginning.

Also we did not use neural networks with recursive layers because for training such networks there is still no sufficient way and it may take longer times to be trained however these networks when they are trained they gives better results also they are more powerful in solving differential equations with nonlinearity.

The two-layer standard feedforward neural network (FFNN) is shown in fig (4) is being used as the controller. It has been composed of an input buffer, non-linear hidden layer and a linear output. So, every ANN consists of one hidden layer which is the first layer output function is the sigmoid function and the second layer output function is a linear function. It has been proven that a network with such structure and enough amounts of neurons in its hidden layer can produce any function with limited no. of discontinuity.

D. Training method

The problem that the network must solve the inverse dynamics of the system which could be presented as a simple function, Y = F(x). For training the network we must provide the ANN with a pair of [X Y].

As mentioned earlier the ANN is trained with the data which is generated by PID controller such that when the PID generated the Torque [Ti] this Torque is applied to the manipulator with the joint angles of vector [Ji] and angular velocities of [Wi] and then the error integration as [Ai], Y = [Ti]. In other words, we get the data from the forward dynamics of the system which is solved by the simulator or the real manipulator and use this data to solve the inverse dynamics of the system.

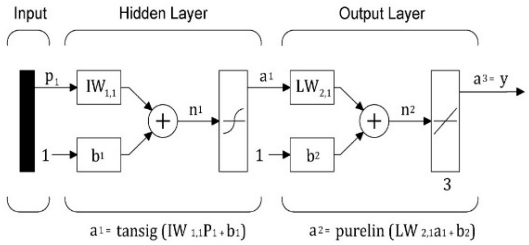


Fig. 4. Structure of ANN

E. Learning Function

The inputs X = [q_d $\dot{q}_d$ $\int e$]^T are multiplied by weights ($\omega^1{}_{ij}$) and summed at each hidden node, then the nodes are activated through a nonlinear function, F(.), called sigmoid function which is bounded between 0 and 1:

$$F(.) = \frac{1}{[1+\exp(-net)]} \quad (16)$$

Where net is the weighted sum of the products, that is;
net = $x_1\,\omega_1 + x_2\,\omega_2 + x_3\,\omega_3 + \ldots\ldots + x_m\,\omega_m$
The activated signals are weighted ($\omega^2{}_{jk}$) and summed at each output node. Thus, the output at a linear output node $\emptyset_n$ can be calculated from inputs as follows:

$$\emptyset_n = [\sum_{j=1}^{n_h} \omega^2{}_{jk} \left(\frac{1}{1+\exp(-\sum_{i=1}^{n_i} x_i \omega^1{}_{ij} + b^1{}_j)}\right)] + b^2{}_K \quad (17)$$

Where;
$\boldsymbol{n_i}$ …….. no. of I/Ps
$\boldsymbol{n_h}$ ……. no. of O/Ps
$\boldsymbol{x_i}$ ……. i^{th} I/P of vector X
$\boldsymbol{\omega^1_{ij}}$…first layer weight between i^{th} i/p and j^{th} hidden layer
$\boldsymbol{\omega^2_{jk}}$ .2nd weight layer between j^{th} hidden and k^{th} o/p layer
$\boldsymbol{b^1_j}$ ……. biased weight for jth hidden layer
$\boldsymbol{b^2_k}$ …… biased weight for kth output layer

The back-propagation algorithm uses supervised learning, and the goal is to reduce the error. The training begins with random weights, until we adjust them to get minimum error. As shown in fig. (5).

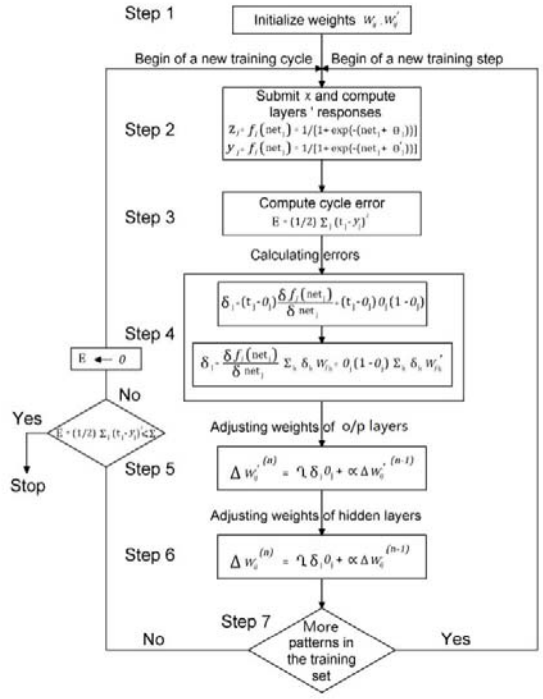


Fig. 5. Flow chart of the error back-propagation algorithm

The ANN o/p serves as the i/p to the closed loop manipulator system as shown in figure (6). The training function is the "trainlm" function from Matlab toolbox. Trainlm is a network training function that updates weight and bias values according to Levenberg Marquardt optimization [16].

This algorithm typically requires more memory but less time. Training automatically stops when generalization stops improving.

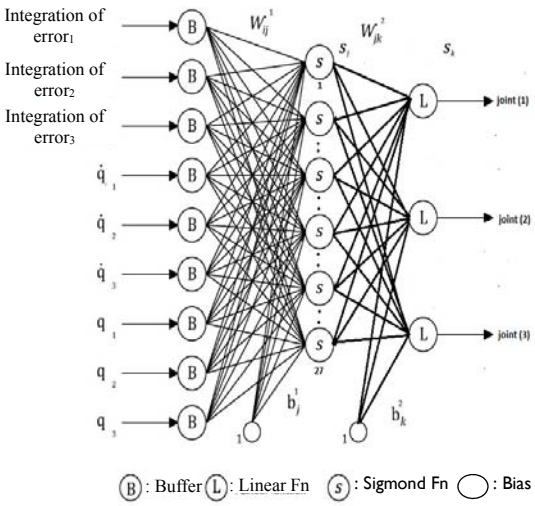


Fig. 6. Multilayered feedforward neural network structure

We started training the ANN starting with random number of neurons in the hidden layer (three neurons) seeing the results (Performance, Error Histogram and Regression), increasing number of neurons the results get better, as shown in table (2), till we have the best Performance, Error Histogram and Regression at 27 neurons in the hidden layer, as shown in figures (7-9). Simulating NN or deploying with Simulink coder tools, generating a Simulink diagram.

Table 2. Performance of NN

NO. OF NEURONS	MSE	ALL REGRESSION
3	1141	0.69
10	54	0.81
27	$2*10^{-3}$	0.93

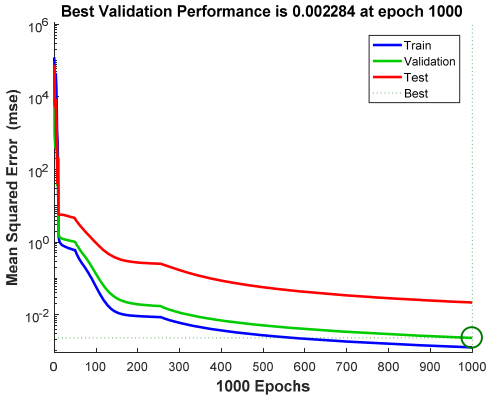


Fig. 7. validation performance

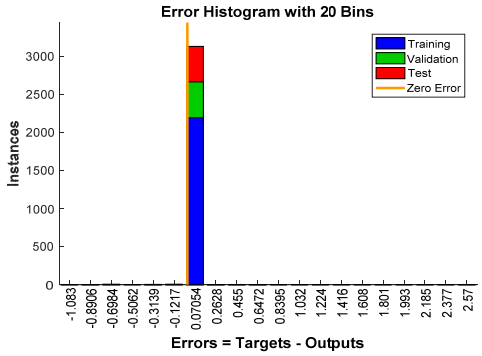


Fig. 8. Error histogram

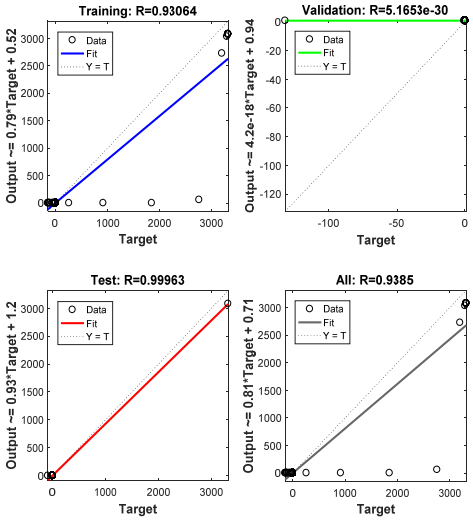
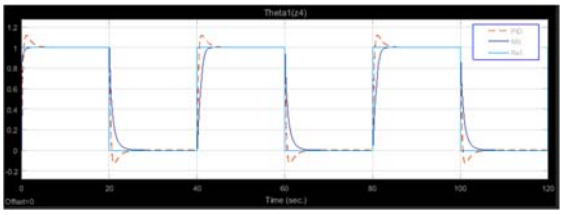


Fig. (9) Over all error

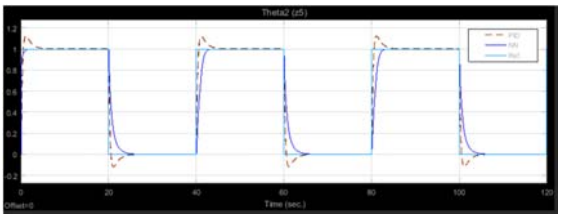
F. *Simulation Results*

The simulation is done in Matlab and Matlab toolbox Peter I. Cork [17]. The simulations were performed depending on the dynamic model previously derived. As we have said the ANN was designed with three layers (including the input layer), three nodes for the input layer, twenty-seven nodes for the hidden layer, and one node for the output layer.

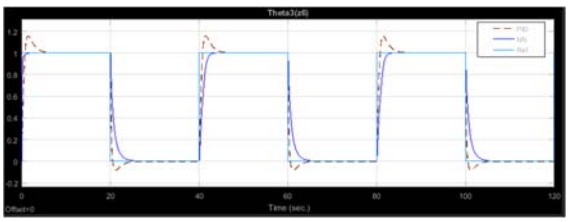
The first input signal used is generated using square generator block in Matlab library sources. Figs. (10 a, b and c) show system response to the input signal of the three joints. Fig. (11 a, b and c) gives the response for another form of input signal. The blue line indicates the desired path and the red line indicates the tracking control result which is not easy to be seen since it is almost completely covered by the blue one.



(a)

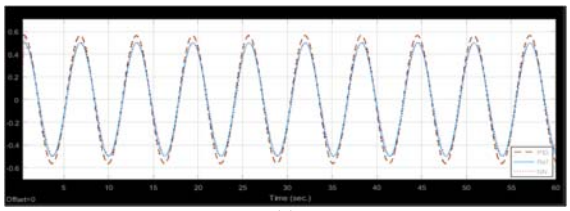


(b)

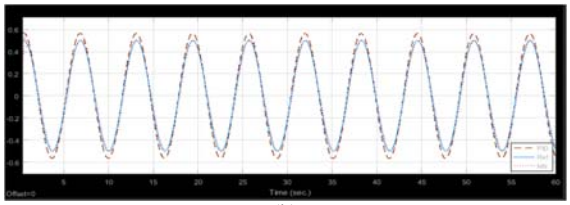


(c)

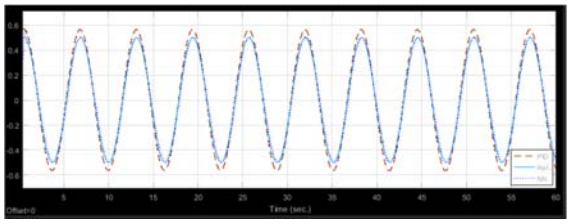
Fig. 10. Simulation results for (a) θ_1, (b) θ_2 and (c) θ_3



(a)



(b)



(c)

Fig. 11. (a, b, c) Simulation results of the three joints for sinusoidal input

As we can see from the above figures different signal forms have been used as input (reference) signal to stand on the efficiency and robustness of the controller. The rise time for θ1, θ2 and θ3 approximately equals (0.8, 0.9 and 0.7) seconds respectively with very small overshoots and with almost no steady state error. The transient response of our controller is more convenient than that of the conventional PID controller. The system has a rapid and efficient response to different input signals with very small overshoot and approximately no error. Also, the controller has a good performance in disturbance rejection as shown in figure (12)

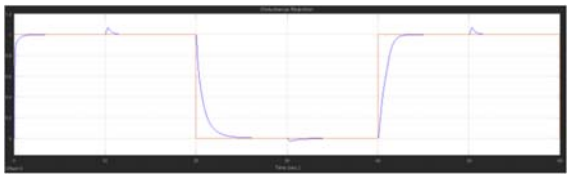


Fig. 12. Disturbance rejection response

As it is obvious in the system the network solves the inverse dynamics of the system very fast and a significant result is observed. The no. of neurons in the network may affect the performance of the system and as the no. of neurons increases the error of the system decreases, however the network needs more time to be trained. One idea to be applied is that at the first cycles of training we could construct the network with a few no. of neurons and as the training continues new networks with much neurons would be substituted with previous networks to accomplish best performance for the system.

IIII. Conclusion and Future work

In this thesis paper the mathematical model of a three link rigid manipulator is presented. This mathematical model is multi-input multi-output MIMO, coupled, and nonlinear equations. An efficient method for control of the manipulator is introduced. The ANN is used to identify both the dynamics and the kinematics of a manipulator. Although during training of the NN the MSE at first is large, however, after changing number of neurons in the hidden layer MSE converges to zero. The controller design is independent from parameters of the system and controller learns the system parameters during its operation. At the end the controller performance is tested by the simulation using MATLAB & SIMULINK.

The topic of the thesis is related to other fields in robotics and there are lots of jobs that can be done in future. One good idea is to use Dynamic networks with concurrent layers and dynamic structures instead of a feed forwarded network because a network with this structure can solve more complex systems and the controller may show better performance in practice. The other good idea is to somehow increase the robustness of controller by changing the structure of the network such that the ANN controller receives feedback from the manipulator as a separate input. It is also possible to feed time to the network and analyze the performance of the controller to time varying systems.

The other task that can increase the performance of the robot is to use a real-time operation system to control the robot or implement the ANN on an FPGA board for online processing. At last since the network can learn the structure of the manipulator using this controller on a robot with flexible links may lead to interesting results.

APPENDIX. A

$$D_{11} = (m_1 + m_2 + m_3)\, d_{12} + (m_2 + m_3)\, d_{22} + m_3\, d_{32} + 2\, m_3\, d_2\, d_3 \cos x_5 + 2\, (m_2 + m_3)\, d_1\, d_2 \cos x_3 + 2\, m_3\, d_1\, d_2 \cos (x_3 + x_5)$$

$$D_{122} = (m_2 + m_3)\, d_1\, d_2 \sin x_3 + m_3 d_1 d_2 \sin (x_3 + x_5)$$

$$D_{133} = m_3\, d_2\, d_3 \sin x_5 + m_3\, d_1\, d_2 \sin (x_3 + x_5)$$

$$D_{112} = 2\, (m_2 + m_3)\, d_1\, d_2 \sin x_3 + 2\, m_3\, d_1\, d_2 \sin (x_3 + x_5)$$

$$D_{113} = 2\, m_3\, d_2\, d_3 \sin x_5 + 2\, m_3\, d_1\, d_2 \sin (x_3 + x_5)$$

$$D_{123} = 2\, m_3\, d_2\, d_3 \sin x_5 + 2\, m_3\, d_1\, d_2 \sin (x_3 + x_5)$$

$$C_1 = (m_1 + m_2 + m_3)\, g\, d_1 \sin x_1 + (m_2 + m_3)\, g\, d_2 \sin (x_1 + x_3) + m_3\, g\, d_3 \sin (x_1 + x_3 + x_5)$$

$$D_{22} = (m_2 + m_3)\, d_{22} + m_3\, d_{32} + 2\, m_3\, d_1\, d_2 \cos x_5$$

$$D_{211} = (m_2 + m_3)\, d_1\, d_2 \sin x_3 + m_3\, d_1\, d_2 \sin (x_3 + x_5)$$

$$D_{233} = m_3\, d_2\, d_3 \sin x_5$$

$$D_{213} = 2\, m_3\, d_2\, d_3 \sin x_5$$

$$D_{223} = 2\, m_3\, d_1\, d_2 \sin x_5$$

$$C_2 = (m_2 + m_3)\, d_2\, g \sin (x_1 x_3) + m_3\, d_3\, g \sin (x_1 + x_3 + x_5)$$

$$D_{33} = m_3\, d_{32}$$

$$D_{311} = m_3\, d_2\, d_3 \sin x_5 + m_3\, d_1\, d_2 \sin (x_3 + x_5)$$

$$D_{322} = m_3\, d_1\, d_2 \sin x_5$$

$$D_{312} = 2\, m_3\, d_2\, d_3 \sin x_5$$

$$C_3 = m_3\, d_3\, g \sin (x_1 + x_5 + x_3)$$

REFERENCES

[1] Frank L. Lewis, Darren M. Dawson, and Chaouki T. Abdallah. "*Robot Manipulator Control: Theory and Practice*", chapter 1 - Commercial Robot Manipulators, pages 1–19. CRC Press, 2003.

[2] Maple Soft. Robot Manipulators - Position, Orientation and Coordinate Transformations [User-guide]. Retrieved from: https: //www. maplesoft.com /content/ EngineeringFundamentals/13/MapleDocument_13/Position%20Orientation%20and%20Coordinate%20Transformations.pdf., 2016.

[3] Narendra K. S. and K Parthasarathy, "Identification and control of dynamical systems using neural networks", IEEE Trans. on Neural Networks, vol. 1, no. 1, pp 4-27, 1990.

[4] K. Hunt, G. Irwin and K. Warwick (Eds.), Neural Networks Engineering in Dynamic Control Systems, Springer Verlang, New York, USA, 1995.

[5] George Philipp, Jaime G. Carbonell, " Nonparametric Neural Networks" ICLR, 2017.

[6] Proceedings of I993 International Joint Conference on Neural Networks, ROBOTIC MANIPULAI'OR TRAJECTORY CONTROL USING NEURALNETWORKS Bin Jin Department of Electrical Engineering, Shanghai University of Technology, 149 Yan-chang Road, Shanghai 200072, P.R. Chinam, 1993.

[7] Trajectory Control of Robotic Manipulators Using Neural Networks, Tomochika Ozaki, Tatsuya Suzuki, Member, IEEE, Takeshi Furuhashi, Member, IEEE, Shigeru Okuma, Member, IEEE, and Yoshiki Uchikawa, 1991.

[8] Adaptive Neural Network Control of Robot Manipulators in Task Space, Shuzhi S. Ge, Member, IEEE, C. C. Hang, Senior Member, IEEE, and L. C. Woon. 1998.

[9] A Neural Network for the Trajectory Control of Robotic Manipulators with Uncertainties, Boo Hee Nam, Sang Jae Lee, and Seok Won Lee, ERC-ACI, Dept. of Control and Instrumentation Eng., Kangwon National Univ., Chunchon, Kangwondo 200-701, Korea. 1997.

[10] A new neural network control technique for robot manipulators, seul jung and T.C. Hsia, Robotics Research Laboratory, Department of Electrical and computer Engineering, University of California, 1995.

[11] A Neural Network Compensator for Uncertainties of Robotics Manipulators Akio Ishiguro, Takeshi Furuhashi, Shigeru Okuma, Member, ZEEE, and Yoshiki Uchikawa. 1992.

[12] Possibility of Neural Networks Controller for Robot Manipulators' Tetsuro YABUTA and Takayuki YAMADA NTT Transmission Systems Laboratories, Tokai, Ibaraki, 31 9-1 1, JAPAN. 1990.

[13] Sherif ahmad, A. E International Journal of Computer Applications 179(34), Dynamic Modelling with a Modified PID Controller of a Three Link Rigid Manipulator. 2018.

[14] Derek Rowell, 2002. 'State-Space Representation of LTI Systems'.

[15] Sung, S. W. and In-Beum Lee. “Limitations and Countermeasures of PID Controllers,” Department of Chemical Engineering, Pohang University of Science and Technology, Pohang, Korea, 1996.

[16] A Method for the Solution of Certain Non-Linear Problems in Least Squares, Kenneth. Levenberg (1944). The Quarterly of Applied Mathematics 2: 164–168.

[17] Matlab robotics toolbox by peter I. Corke. http://www.ict.csiro.au/downloads/robotics.2018.