NEURAL CONTROLLER OF ROBOTIC MANIPULATOR

A Major Project

Submitted In Partial Fulfillment of Requirements

For The Award of the Degree of

MASTER OF ENGINEERING

(CONTROL & INSTRUMENTATION)

Submitted by

ASURJEET SHARMA

University Roll No. 12232

Under the Supervision of

PROF. MADHUSUDAN SINGH

Electrical Engineering Department

Department of Electrical Engineering

Delhi College of Engineering

University of Delhi

Delhi-110042

2009

CERTIFICATE

It is certified that Mr. ASURJEET SHARMA, Roll No. 03/C&I/07, a student of M.E., Control and Instrumentation, Department of Electrical Engineering, Delhi College of Engineering, has submitted the dissertation entitled "NEURAL CONTROLLER OF ROBOTIC MANIPULATOR" under our guidance towards partial fulfillment of the requirements for the award of the degree of Master of Engineering. This dissertation is a bonafide record of project work carried out by him under our guidance and supervision.

(Dr. MADHUSUDAN SINGH)
Professor
Electrical Engineering Department
Delhi College of Engineering
Delhi-110042

(Mr. BHARAT BHUSHAN)
Asst. Professor
Electrical Engineering Department
Delhi College of Engineering
Delhi-110042

ACKNOWLEDGEMENT

I am thankful to the Almighty because without his blessings this work was not possible. It is a great pleasure to have the opportunity to extent my heartfelt gratitude to everybody who helped me throughout the course of this project.

It is distinct pleasure to express my deep sense of gratitude and indebtedness to my project supervisor ***Prof. Madhusudan Singh*** for his invaluable guidance, encouragement and patient reviews. His continuous inspiration has made me complete this dissertation. He kept on boosting me time and again for putting an extra ounce of effort to realize his work.

I would also like to take this opportunity to present my sincere regards to **Prof. Parmod Kumar**, Head Electrical Engineering Department, DCE for his support and encouragement.

I am grateful to my parents and my brother, for their moral support all the time; they have been always around on the phone to cheer me up in the odd times of this work.

ASURJEET SHARMA

College Roll No. 03/C&I/07

University Roll No: 12232

M.E (C&I)

CONTENTS

CERTIFICATE	ii
ACKNOWLEDGEMENT	iii
LIST OF FIGURES	vii
LIST OF SYMBOLS	ix
ABSTRACT	xii
CHAPTER-I INTRODUCTION	1-5
1.0 General	1
1.1 Types of Robotic manipulator	2
1.1.0 JIRA (Japanese Industrial Robot Association) classification	2
1.1.1 Classification based on Drive Technologies	2
1.1.2 Classification based on Work- Envelope Geometry	3
1.1.3 Classification based on Motion Control Methods	3
1.2 Control of robotic manipulator	4
1.3 Conventional versus intelligent control	4
1.4 Objective of present work	5
1.5 Conclusion	5
CHAPTER -II LITERATURE REVIEW	6-10
2.0 General	6
2.1 Literature review on neural controller for robotic manipulator	6
2.2 Literature review on fuzzy logic controller for robotic manipulator	7
2.3 Literature review on intelligence system	8
2.4 Literature review on the importance of neural control	9
2.5 Literature review on the neural network design and environment for simulation	10
2.6 Conclusion	10

CHAPTER-III DYNAMICS AND CONTROLLER OF ROBOTIC MANIPULATOR 11-22
3.0 General 11
3.1 Robot's kinematics, dynamics 11
3.1.1 Kinematics 11
3.1.2 Dynamic 11
3.2 Robot dynamics 12
3.2.1 Euler-lagrange formulation 13
3.3 Equation of motion 13
3.4 Dynamics of a one-link 14
3.5 Dynamic of a 2-link robot arm 15
3.6 Controller of robotic manipulator 16
3.6.1 Conventional controller 17
3.6.1.1 Proportional –integral- derivative controller or PID controller 17
3.6.1.2 Feed forward inverse dynamics control 18
3.6.1.3 Computed torque control 19
3.6.1.4 Critically damped inverse dynamics control 19
3.6.2 Intelligent controller 19
3.6.2.1 Fuzzy logic control 19
3.6.2.2 Neural control 20
3.7 Advantage of neural controller 21
3.8 Conclusion 22

CHAPTER-IV NEURAL NETWORKS 23-28
4.0 General 23
4.1 conventional computers approach versus neural networks approach 23
4.2 Biological and artificial neural systems 24
4.3 Neuron model 24
4.4 Transfer functions 26
4.5 Learning rule 26

4.6 The back-propagation algorithm 27

4.7 Implementations of neural networks 28
4.8 Conclusion 28

CHAPTER-V RESULTS AND OBSERVATON 29-70

5.0 General 29
5.1 Neural controller for one-link manipulator simulation 29
5.2 Effect of the variation of first layer transfer function of neural network 30
5.2.1 Summary 39
5.3 Effect of the variation of second layer transfer function of neural network 38
5.3.1 Summary 48
5.4 Effect of the variation of constant of continuous uncertainties(c) 49
5.4.1 Summary 54
5.5 Effect of the variation of robot arm length 55
5.5.1 Summary 60
5.6 Effect of the variation of robotic manipulator mass (m) 61
5.6.1 Summary 64
5.7 Two-link manipulator simulation 64
5.8 Effect of the variation of learning rate 65
5.8.1 Summary 69
5.9 Conclusion 70

CHAPTER-VI CONCLUSION AND FUTURE SCOPE OF WORK 71
6.1 Conclusion 71
6.2 Future scope of work 71

REFRENCES 72-73

LIST OF FIGURES

Fig. 3.1 One-link robot arm 14
Fig. 3.2 Two-link robot arm 15
Fig. 3.3 General structure of robot control system 16
Fig. 3.4 Feed forward inverse dynamics controller 18
Fig. 3.5 Block diagram of a fuzzy controller 20
Fig. 3.6 Scheme for learning dynamics model 21
Fig. 4.1 Single input neuron 25
Fig. 5.1 Performance with “compet,tansig” transfer function 30
Fig. 5.2 Performance with “hardlim,tansig” transfer function 31
Fig. 5.3 Performance with “hardlims,tansig” transfer function 32
Fig. 5.4 Performance with “purelin,tansig” transfer function 33
Fig. 5.5 Performance with “satlin,tansig” transfer function 34
Fig. 5.6 Performance with “logsig,tansig” transfer function 35
Fig. 5.7 Performance with “tansig,tansig” transfer function 36
Fig. 5.8 Performance with “poslin,tansig” transfer function 37
Fig. 5.9 Performance with “satlins,tansig” transfer function 38
Fig. 5.10 Performance with “purelin,compet” transfer function 40
Fig. 5.11 Performance with “purelin,hardlim” transfer function 41
Fig. 5.12 Performance with “purelin,hardlims” transfer function 42
Fig. 5.13 Performance with “purelin,purelin” transfer function 43
Fig. 5.14 Performance with “purelin,satlin” transfer function 44
Fig. 5.15 Performance with “purelin,logsig” transfer function 45
Fig. 5.16 Performance with “purelin,tansig” transfer function 46
Fig. 5.17 Performance with “purelin,poslin” transfer function 47
Fig. 5.18 Performance with “purelin,satlins” transfer function 48
Fig. 5.19 Performance with c = 0.0001 49
Fig. 5.20 Performance with c = 0.001 50
Fig. 5.21 Performance with c = 0.01 51

Fig. 5.22 Performance with c = 1 52
Fig. 5.23 Performance with c = 2 53
Fig. 5.24 Performance with c = 3 54
Fig. 5.25 Performance with l = 0.1 55
Fig. 5.26 Performance with l = 0.3 56
Fig. 5.27 Performance with l = 2 57
Fig. 5.28 Performance with l = 3 58
Fig. 5.29 Performance with l = 3.1 59
Fig. 5.30 Performance with l = 3.5 60
Fig. 5.31 Performance with m = 0.08 61
Fig. 5.32 Performance with m = 3 62
Fig. 5.33 Performance with m = 4 63
Fig. 5.34 Performance with m = 4.5 64
Fig. 5.35 Performance with learning rate = 0.00000000001 65
Fig. 5.36 Performance with learning rate = 0.001 66
Fig. 5.37 Performance with learning rate = 1 67
Fig. 5.38 Performance with learning rate = 2 68
Fig. 5.39 Performance with learning rate = 2.1 69

LIST OF SYMBOLS

S.No.	Symbols	Descriptions
1	L	Lagrangian multiplier
2	T	Kinetic energy of the robotic manipulator
3	U	Potential energy of the robotic manipulator
4	$\dot{r}_i$	Generalized co-ordinates due to applied forces corresponding to the generalized co-ordinates
5	n	Number of independent generalized co-ordinates
6	ϕ_i	Generalized forces due to applied forces corresponding to the generalized co-ordinates
7	I	n×n generalized inertia matrix.
8	q	n-dimensional vector of joint positions.
9	$\dot{q}$	n-dimensional vector of joint velocities.
10	$\ddot{q}$	n-dimensional vector of joint accelerations.
11	h	n-dimensional vector of centrifugal and coriolis acceleration.
12	Υ	n-dimensional vector of gravitational accelerations.
13	ζ	n-dimensional vector of generalized forces.
14	F_c	Vector of uncertainties effect is decomposed as continuous part
15	F_d	Vector of uncertainties effect is decomposed as discontinuous part
16	m	Mass of arm of the one-link manipulator
17	l	Length of arm of the one-link manipulator
18	g	Acceleration due to gravity

19	l_1	Length of first arm of two-link manipulator
20	l_2	Length of second arm of two-link manipulator

21	m_1	Mass of first arm of two-link manipulator
22	m_2	Mass of second arm of two-link manipulator
23	ζ_{PID}	Vector of PID joint torques
24	K_D	Derivative constant
25	K_P	Proportional constant
26	K_I	Integral constant
27	e	Error between reference trajectory and actual trajectory
28	ζ_{ffid}	Vector of feed forward inverse dynamic torque
29	ζ_{CTC}	Vector of computed torque control torque
30	p	Scalar input apply to neuron model
31	w	Scalar weight of neuron model
32	b	Bias of the neuron model
33	a	Scalar neuron output
34	n	Net input of neuron model
35	f	Transfer function of neuron model
36	c	Co-efficient of the continuous uncertainties
37	q_r	Reference trajectory of first arm of one link manipulator
38	tau_r	Reference torque of first arm of one link manipulator
39	q_{sim}	Simulated output trajectory of first arm of one link manipulator
40	tau_{sim}	Simulated output torque of first arm of one link manipulator
41	l_v	Learning rate of the training method of neural network used in two link manipulator simulation

42	q_{r1}	Reference trajectory of first arm of two link manipulator

43	q_{r2}	Reference trajectory of second arm of one link manipulator
44	q_{sim1}	Simulated trajectory of first arm of one link manipulator
45	q_{sim2}	Simulated trajectory of second arm of one link manipulator
46	$error_1$	Error between first arm reference trajectory and simulated trajectory of two link manipulator
47	$error_2$	Error between second arm reference trajectory and simulated trajectory of two link manipulator

ABSTRACT

In this project, the design and simulation of a neural controller for robotic manipulator is carried out. The simulation study is carried out on one-link and two-link manipulators. This project develops a neural controller designing for the proper choice of the transfer function by the using of the back-propagation function “trainlm” for the one link manipulator. It describes the effect of the variation of parameters related to one-link manipulator with the neural controller. The range of the robotic manipulator parameter where a neural controller performs with tracking capability and identification capability is described in detail. This all performance is done on the basis of the coding which is developed by us.

MATLAB coding for the neural controller for the two link manipulator is developed. The effects of the learning rate of the training method on the neural controller performance and the learning rate variations are also presented.

CHAPTER I
INTRODUCTION

1.0 GENERAL

The name robot came the Czechoslovakian word “Robota” which means a worker or a slave doing heavy work [19]. Robotic manipulators are very complicated nonlinear system. Based on the Robotics Institute of America (RIA) definition: “A robot is a reprogrammable multifunctional manipulator design to move material, parts, tools, or specialized devices through variable programmed motions for the performance of a variety of tasks”[18]. From the engineering point of view, robots are complex, versatile devices that contain a mechanical structure, a sensory system, and an automatic control system. Theoretical fundamentals of robotics rely on the results on the results of research in mechanics, electric, electronics, automatic control, mathematics, and computer sciences.

Robots find a vast variation of industrial applications and are used for various technological operations. Robots enhance productivity in industry and deliver relief from tiresome, monotonous, or hazardous works. Moreover, robots perform many operations better than people do, and they provide higher accuracy and repeatability. In many fields, high technological standard are hardly attainable without robots. Apart from industry, robots are used in extreme environments. They can work at low and high temperatures; they don’t even need lights, rest, fresh air, a salary, or promotions. Today, more complicated applications, such as welding, paining, assembling, require much more motion capability and sensing. Hence, a robot is multi-disciplinary engineering device. Mechanical engineers deal with the design of mechanical components, arms, end-effectors, and also are responsible for kinematics, dynamics and control analyses of robots. Electrical engineers work on robot actuators, sensors, power, and control systems. System design engineering deals with perception, sensing, and control methods of robots. Programming, or software engineering, is responsible for logic, intelligence, communication, and networking.

1.1 TYPES OF ROBOTIC MANIPULATORS

In order to refine the general notion of a robotic manipulator, it is necessary to classify manipulators according to various criteria such as robot association, drive technology, work envelope geometrics, and motion control methods.

1.1.0 JIRA (Japanese Industrial Robot Association) classification

Class 1: manual handling devices: a device with multi degrees of freedom that is actuated by an operator.

Class 2: fixed sequence robot: a device that performs the successive stages of task according to a predetermined and fixed program.

Class 3: variable sequence robot: a device that performs the successive stages of a task according to a predetermined but programmable method.

Class 4: playback robot: a human operator performs the task manually by leading the robot, which records the motions for later playback. The robot repeats the same motions according to the recorded information.

Class 5: numerical control robot: the operator supplies the robot with a motion program rather than the teaching it task manually.

Class 6: intelligent robot: a robot with the ability to understand its environment and the ability to successfully complete a task despite changes in the surrounding conditions under which it is to be performed.

1.1.1 Classification based on Drive Technologies

One of the most fundamental classification schemes is based upon the source of power used to drive the joint of the robot. The two most popular drives technologies are electric and hydraulic. Most robotic manipulators today use electric drives in the form of either DC servomotors or DC stepper motors. However, when high speed manipulation of substantial loads is required, such as molten steel handling or auto body part handling, hydraulic-drive robots are preferred. One serious drawback of hydraulic-drive robots lies in their lack of cleanliness, a characteristic that is important for many assembly applications.

1.1.2 Classification based on Work-Envelope Geometry

A robot is called a serial or open-loop manipulator if its kinematic structure does not make a loop chain. It is called a parallel or closed-loop manipulator if its structure consists of both open and closed-loop chains.

Most industrial manipulators have six degree of freedom (DOF). The open-loop manipulators can be classified based on their joints starting from the grounded joint. Only six different lower-pair joints are possible: revolute (rotary), prismatic (sliding), cylindrical, spherical and planar. Of these, only rotary (R) and prismatic (P) joints are common in manipulator. From the two types of joints there mathematically 72 different manipulator configurations, simply because each joint can be P or R, and the axes of two adjacent joints can be parallel, orthogonal, or perpendicular. Two orthogonal joint axes intersect at a right-angle; however two perpendicular joint axes are in right-angle with respect to their common normal. Two perpendicular joint axes become parallel if one axis turns 90 degree about the common normal. Two perpendiculars joint axes become orthogonal if the length of their common normal tends zero.

Out of the 72 possible manipulators, the important ones are:

1) Selective compliant robot for assembly (SCARA)
2) Elbow, revolute, articulated, or anthropomorphic
3) Spherical
4) Cylindrical
5) Cartesian

1.1.3 Classification based on Motion Control Methods

Another fundamental classification criterion is the method used to control the movement of the end-effecter or tool. The first type is point-to-point motion, where the tool moves to a sequence of discrete points in the workspace. The path between the points is not explicitly controlled by the user. Point-to-point motion is useful for operations which is discrete in nature. For example, spot welding is an application for which point-to-point motion of the tool is all that require.

The other type of motion is continuous-path motion, sometimes called controlled-path motion. Here the end-effecter must follow a prescribed path in three-dimensional space, and the speed of motion along the path may vary. Different applications of robots with continuous-path motion control include paint spraying, arc welding, and the application of glue or sealant.

1.2 CONTROL OF ROBOTIC MANIPULATOR

The robot control problem may be characterized as the described motion of the end-effecter. A described motion is specified as a trajectory in Cartesian coordinates while the control system requires input in joint coordinates.

The robot control comprises three computational problems:

1) Determination of the trajectory in Cartesian space
2) Transformation of the Cartesian trajectory into equivalent joint coordinates spaces, and
3) Generation of the motor torque commands to realize the trajectory.

There are many control strategy that can be applied for control of robotic manipulators. These range from conventional to adaptive and intelligent controllers. The controller or control unit has three roles:

1) Information role, which consists of collecting and processing the information provided by the robot's sensors.
2) Decision role, which consists of planning the geometric motion of the robot structure.
3) Communication role, which consists of organizing the information between robot and its environment. The control unit includes the processor and software.

1.3 CONVENTIONAL VERSUS INTELLIGENT CONTROL

The conventional control is generally based on the assumption of exact knowledge about the system. This assumption is often not valid since the development of any practical system may not include precise information of factors such as friction, backless, un-modelled dynamics and uncertainty arising from any of the source. This control is generally used for linear system and some specified non-linear system.

To the robotics point of view, the use of conventional controllers demands the availability of an accurate dynamic robot model. However, this is rarely achievable because of un-modelled dynamics (neglected time-delays, non-linear friction etc.) and parameter uncertainties (deviation of link length etc. from nominal values).

Intelligent control is a control technology that replaces the human mind in making decisions, planning control strategies, and learning new functions whenever the environment does not allow or does not justify the presence of human operator. Artificial neural network and fuzzy logic are potential tools for intelligent control engineering. Intelligent controllers are Fuzzy logic, neural control, and hybrid control (Neuro-fuzzy). Neural networks are best known for

their learning capabilities. Fuzzy logic is a method of using human skills and thinking processes in a machine. This control is more adaptable for non-linear system.

To the robotics point of view, the emergence of neural networks (type of intelligent control) has provided an alternative means of controlling high-speed robot motion. Neural networks can perform combined system identification and adaptive control function, and the yield a good manipulator trajectory tracking performance without requiring an analytical dynamic model.

1.4 OBJECTIVE OF PRESENT WORK

One of the major difficulties in neural networks application is the selection of the parameters in network configuration and the coefficients in learning rule for fast convergence. This project includes analysis of one link manipulator and two link manipulator by the variation of transfer function and learning methods for minimum path and torque error. Also showed, the effect on path trajectory and torque by the variation of the robot parameter such as length, mass and friction.

1.5 CONCLUSION

In this chapter, definition, application, classification, control of the robot is discussed. The brief discussion about conventional and intelligent control is also presented. At the last, the objective of project is briefly described.

CHAPTER II
LITERATURE SURVEY

2.0 GENERAL

Over the past few years, the use of neural controller in control system has been going widespread popularity. A large number of articles and numerous associations have been devoted to the study on design and implementation of neural controllers. Robotic manipulator is also of critical importance since they are very useful in automation and industries. It is a topic of research due to its wide spectrum of applications. Numerous papers have been written which present different control strategies to control the robotic manipulator. An extensive literature survey on the robotic manipulator and its control was carried out.

2.1 LITERATURE REVIEW ON NEURAL CONTROLLER FOR ROBOTIC MANIPULATORS

Debbache et al. [1] described the motion control of two link robot manipulators with structured and unstructured uncertainties. This paper also described the basic idea of neural network and neural state feedback also demonstrated about asymptotic stability of the control system using Lyapnov's approach.

Toplov et al. [2] described the dynamical on-line learning algorithm for neuro-adaptive control of a class of non-linear system with uncertain dynamics. In this paper, a closed loop control is used, simultaneously with a conventional PD controller and an adaptive variable structure neural controller. Also show the real-time trajectory tracking control of the first three joints of an articulated five degrees-of-freedom (DOF) robot manipulator.

Talebi et al. [3] described the controlling of a non-linear non-minimum phase system using neural controller, it also described the output redefinition strategy and where it used. At last it proved that the redefinition strategy based on neural network was proposed that does not need any a prior knowledge about the non linearity of the system. For prove it, assumed a non-minimum single-link flexible link robotic manipulator.

Efrati and Flashner [4] described the tracking control of mechanical systems (assuming example of robotic link manipulators) based on artificial neural conjunction with a PD controller. Here the neural network is used to approximate the system dynamics in presence of parameter uncertainties and disturbances and the PD controller is designed to ensure

convergence of the tracking error. This paper presents, the neural network configuration and the tuning algorithm are simple and thus can be implemented on-line with low computational and storage requirements. At last shown that the closed loop tracking tends to zero in presence of model uncertainties and disturbances while minimizing the control effort.

Takahashi and Yamada [5] described a neural- network controller for a flexible robot arm and also described the neural controller design using state-space representation. At last, experimental confirmation of the neural controller can control the flexible arm by learning ability of the neural network without exact a prior knowledge of the system even under existing nonlinear disturbances such as solid friction is shown.

Kosmatopoulos et al. [6] solved the identification problem of a robotic manipulator using dynamic distributed multi-layer back propagation network and a novice algorithm is used. By the simulation results, it is shown that the algorithm can handle abrupt changes in input data with the error converges quickly to zero.it is described that the network can effectively perform after training stops even when the input waveforms have been never been presented before.

Fukunda and Shibata [7] described the control of robotic manipulator using neural controller which has been integrated time delays elements. By simulation results proved that the 'active time delay neural network' can obtain desirable gains of the control system by learning, therefore, the active time delay neural network is more applicable and adaptable than the general neural network to the system which has strong non-linearity and whose dynamics are complex, for example, hybrid control of robotic manipulators which can handle unknown objects and its force control suffering collisions.

2.2 LITERATURE REVIEW ON FUZZY LOGIC CONTROLLER FOR ROBOTIC MANIPULATOR

Jnifene and Andrews [8] presented with the design and implementation of active vibration control based on fuzzy logic and neural networks (NNs). The controller is used to dampen the end point vibration in a single-link flexible manipulator mounted on two degrees freedom platform. The inputs of the fuzzy logic controller (FLC) are the angular position of the hub and the end point deflection of the flexible beam. A NN predicting the defection was obtained using a set of three strain gauge pairs mounts on the beam and a linear- variable differential transformer placed on the tip. This paper also discussed how to build the rule base of the flexible beam based on the relation between angular displacement of the hub and the end-

point deflection as well as the approach that presented several experimental results to validate the NN's mode and showed the effectiveness of the FLC in reducing the end point vibration.

Yoo and Ham [9] presented adaptive control schemes for robotic manipulator which has the parametric uncertainties. To compensate these uncertainties, fuzzy logic system (FLS) is used because that has capability to approximate any nonlinear function over the compact input space. Here, the adaptive control is used for decreasing the effect of approximation error. For reducing the rules FLS, assumed some properties of robot dynamics and the decomposition of the uncertainties function. The presented controller, in this paper, is robust not only to the structured uncertainty such as pay load parameter, but also to the unstructured one such a friction model and disturbance. At last, by the simulation of a two-link manipulator proved the validation of the controller.

2.3 LITERATURE REVIEW ON INTELLIGENT SYSTEMS

Wilamowski [10] presented the comparison of various methods of computational intelligence and it is illustrated with examples. The concerned methods in this paper are neural networks, fuzzy system and genetic systems. The main focussed topic in this paper is neural networks. It present learning algorithms and their special architectures. Learning rule such as hebian learning, LMS-least mean square learning, delta learning, WTA-winner take all learning and PCA-principal component analysis are presented. Architecture specific learning algorithms for cascade correlation networks, Sarajenidi and Hecht-Nierlren networks, functional link networks, polynomial networks. Counter propagation networks, RBF-radial basis function networks are described.

Melin and Castillo [11] described the soft computing techniques to controlling non-linear dynamical systems in real world problem. This is described that nonlinear dynamical system are difficult to control due to the instable and even chaotic behaviours that they may occur in these systems. Here, the soft computing consists of fuzzy logic, neural network etc. it is also demonstrated that computational techniques result good performances, the two techniques may also used simultaneously for a system, known as hybrid control. The described applications include robotics aircraft systems, biochemical reactors and manufacturing batteries.

Shoureshi [12] presented an introduction to and appreciation for intelligent control system, their application areas and justifies their need. Specific problem related to automated human control is discussed. Some analytical derivations related to neural networks and fuzzy optimal

control as elements of proposed intelligent control systems, along with experimental results are presented.

2.4 LITERATURE REVIEW ON THE IMPORTANCE OF NEURAL CONTROL

Ng and Cook [13] presented a neural network controller use more robust than classical and adaptive controllers for a plant with unknown time-delay. Recursive least squares (RLS) algorithm is used for the training in given neural networks. By the simulation results, it proved that NN controller with the on line is better than PID controller and self tuning pole assignment controller when the plants time delays are unknown and varying. Hence, the unknown NN controller with the on-line learning algorithm is suitable for real time application to unknown and varying time delay plants. Simulation results have also shown that NN controller is better at its optimum network size than an over parameterized or under parameterized network.

Jin et al. [14] discussed about dynamic recurrent neural networks (DRNNs) which provides the potential for the learning and control of a general class of unknown discrete time nonlinear system which are treated as “black boxes” with multi inputs and multi outputs (MIMO). The DRNNs is described by asset of nonlinear difference equations and suitable analysis for the input output dynamics of the model is performed to obtain the inverse dynamics. Also described about the ability of a DRNN structure.

Chen [15] described about back propagation neural network which is applied to a nonlinear self-tuning tracking problem. Since the traditional self tuning adaptive control techniques can only deal with linear system or some special nonlinear systems. It also limited to unknown nonlinear system. This problem is overcome by introducing back-propagation neural network into the self-tuning control scheme. It is also demonstrated that the new control method has the potential to deal with unknown linearizable nonlinear systems.

Narendra and Parthasarathy [16] demonstrated the application of neural network which are effectively used for the identification and control of nonlinear dynamics systems. Static and dynamic back-propagation methods for the adjustments of parameter are discussed. The model of NN is used, here, is multilayer and recurrent network. At the last, the simulation results reveal that the identification and adaptive control schemes suggested the practical feasible.

2.5 LITERATURE REVIEW ON THE NEURAL NETWORKS DESIGN AND ENVIRONMENT FOR SIMULATION

Yang and Lee [17] describes the selection of the parameters in network configuration and the coefficient in learning rule for fast convergence of the neural network. This paper develops a network design by combining the Taguchi method and the back-propagation network with an adaptive learning rate for minimum training time & effective vibration suppression. The analysis & experiments is shown that the optimal design can be determined in a systematic way thereby avoiding the length trail-and-error.

Tokhi [18] presents the development of an interactive and user friendly environment for simulation and control of flexible manipulator systems. A constrained planer single-link flexible manipulator is considered. Finite-difference algorithm for simulation of a single-link flexible manipulator is used in presented paper. Several open-loop and closed-loop control strategies are developed and incorporated into the environment. Several case studies, demonstrating the utilisation and potential of the environment are presented and discussed. At the last, it is concluded that the environment provides a valuable computer-aided education and research facility for understanding the behaviour of flexible manipulator systems and development of various controller designs.

2.6 CONCLUSION

An extensive literature review of neural controller and fuzzy controller for robotic manipulator is presented. Intelligent control has accelerated the new technological advancement. The importance of the neural controller and its development for an interactive user friendly environment are also described.

CHAPTER III
DYNAMICS AND CONTROLLERS OF ROBOTIC MANIPULATOR

3.0 GENERAL

Kinematics and dynamics is two main branch of science and its related equation is very important for the robot and its controller. Generally, for robot design, the kinematics equation is very important but the controller design of given system dynamics equation play more important role. System modelling is also necessary for the choosing the proper controller for the robotic system. All controllers have its own equation or logic and criteria for fulfilment the robot control.

3.1 ROBOT'S KINEMATICS AND DYNAMICS

3.1.1 Kinematics

Kinematics is a branch of science that analyzes motion with no attention to what causes the motion. By motion mean any type of displacement, which includes changes in position and orientation. Therefore, displacement, and the successive derivatives with respect to time, velocity, acceleration, and jerk, all combine into kinematics.

The forward kinematics problem is when the kinematical data are known for the joint coordinates and are utilized to find the data in the base coordinates frame. The inverse kinematics problem is when the kinematics data are known for the end- effecter in Cartesian space. Inverse kinematics is highly nonlinear and usually a much more difficult problem than the forward kinematics problem. The inverse velocity and acceleration problems are linear, and much simpler, once the inverse position problem has been solved. An inverse position solution is said to have a closed form if it not iterative.

3.1.2 Dynamic

Dynamics is the study of systems that undergo changes of state as time evolves. In mechanical systems such as robots, the change of states involves motion. Derivation of the equations of motion for the system is the main step in dynamic analysis of the system, since equations of motion are essential in the design, analysis, and control of the system.

The dynamic equations of motion describe dynamic behaviour. They can be used for computer simulation of the robot's motion, design of suitable control equations, and evaluation of the dynamic performance of the design.

The problem of robot dynamics may be considered as direct and inverse dynamics problems. In direct dynamics, we should predict the motion of the robot for a given set of initial conditions and torques at active joints. In the inverse dynamics problem, we should compute the forces and torques necessary to generate the prescribed trajectory for a given set of positions, velocities, and accelerations.

3.2 **ROBOT DYNAMICS**

A set of equations that describe the dynamical behaviour of a robot, also referred to as the dynamical model of the robot, will be developed. This development is important in several ways, namely,

1) A dynamical model can be used to develop the suitable control strategies. A sophisticated controller requires the use of a realistic dynamical model to achieve an optimal performance of the robot under high-speed operations. Some control schemes rely directly on a dynamic model to compute actuator torques and forces required to follow a desired trajectory.
2) The dynamical model can be used for computer simulation of a robotic system. By examining the model under various operating conditions, it is possible to predict how a robotic system will behave when it will be built.
3) The dynamic analysis of a robot gives all the joint reaction forces and moments needed for the design and sizing of links, bearings, and actuators.

There are many methodologies to solve robot dynamics as following:

1) Euler –lagrange method
2) Newton-euler method
3) D'alembart principle
4) Kane's equations of motion
5) Decoupled natural orthogonal complement (DeNOC) method

Here only euler-lagrange used for modelling. The advantage of employing the lagrangian approach is that it eliminates the forces of constraint from the dynamic equations of motion if the generalised coordinates are independently chosen. The elimination makes it suitable for

motion control and simulation. However, these eliminated constraint forces can be recovered using lagrange multiplies, if they are to be used for the purpose of design.

3.2.1 Euler-Lagrange formulation

The dynamic model of a robot can be derived in a systematic way using the concept of generalised coordinates and a scalar function called lagragian. The lagrangian is defined as the difference between the kinetic and potential energy of the mechanical system under study, i.e.

$$L = T-U \tag{3.1}$$

Where L denotes the lagrangian, and T and U are respectively the total kinetic and potential energy of the system at hand. Note that the kinetic energy depends on both configuration, i.e. position and orientation, and the velocity of the links of a robotic system, whereas the potential energy depends only on the configuration of the links. Euler –lagrange equations of motion are then given by,

$$\frac{d}{dt}\left(\frac{\partial L}{\partial \dot{r}i}\right) - \frac{\partial L}{\partial \dot{r}\imath} = \Phi_i, \text{ for i=1, 2...n.} \tag{3.2}$$

Where, n is number of independent generalised coordinates used to define the system's configuration, and $\dot{r}_i$'s and Φ_i s are the generalised coordinates and generalised forces due to applied forces due to applied forces corresponding to the generalised coordinates, respectively.

3.3 EQUATION OF MOTION

The generalised equation for n-link arm, assumed all joints are revolute, is following:

$$I\ddot{q} + h + \gamma = \zeta \tag{3.3}$$

Where,

I = the n×n generalised inertia matrix.

q = the n-dimensional vector of joint positions.

$\dot{q}$ = the n-dimensional vector of joint velocities.

$\ddot{q}$ = the n-dimensional vector of joint accelerations.

h = the n-dimensional vector of centrifugal and coriolis acceleration.

γ= the n-dimensional vector of gravitational accelerations.

ζ= the n-dimensional vector of generalised forces.

If we considered uncertainties effect, the equation would be as followed:

$$I\ddot{q} + h + \gamma + F_c + F_d = \zeta \tag{3.4}$$

Where, F_c and F_d vectors representing the dynamic effects as nonlinear frictions, small joint and link elasticities, backless and bounded torque disturbances. Here, the uncertainties effect is decomposed as continuous part F_c and discontinuous part F_d.

According to gravitational acceleration, we could describe the following the following model of robotic manipulator:

1) If gravitational accelerations, γ is a linear function, $\gamma = Nq$, then we have the "linear oscillator" model:

$$I\ddot{q} + h + Nq + F_c + F_d = \zeta \quad (3.5)$$

2) If gravitational accelerations, γ is a linear function, $\gamma = Nq^2$, then we have the "quadratic oscillator" model:

$$I\ddot{q} + h + Nq^2 + F_c + F_d = \zeta \quad (3.7)$$

3) If gravitational accelerations, γ is a linear function, $\gamma = N\sin(q)$, then we have the "sinusoidally oscillator" model:

$$I\ddot{q} + h + N\sin(q) + F_c + F_d = \zeta \quad (3.8)$$

5.4 DYNAMICS OF A ONE-LINK

The dynamic equation of motion of the one-link one –DOF arm derived using the EULER-Lagrange (EL) formulation. Using the EL formulation, the generalised coordinate is q, whereas l/2 is the distance of the link from its joint origin O. Moreover, let the mass of the link be m, and its inertia tensor about the mass centre is denoted by I.

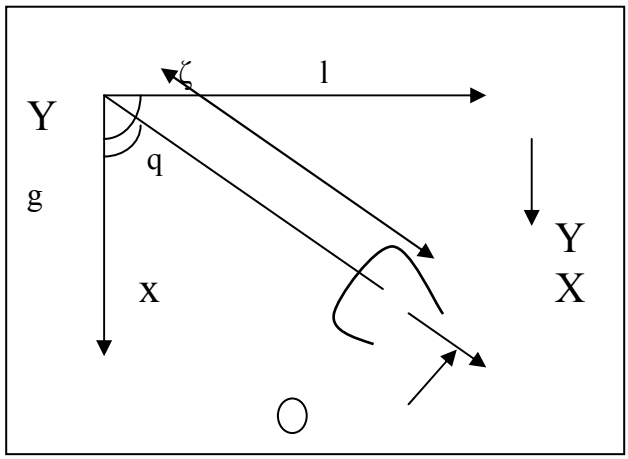

Fig. 3.1 One-link robot arm

The scalar inertia term I is given by,

$I=ma^2/3$;

The elements of the vectors h and γ, i.e. h and γ respectively, are obtained as follows;

The elements of the vectors h and γ respectively,

h=0;

γ = mgasq/2 ;

Where, g = acceleration due to gravity.

And, sq represents sin q.

Now by using general equation, we can find

$$\frac{1}{3}ma^2\ddot{q} + \frac{1}{2}mgasq = \zeta \qquad (3.8)$$

Above equation is one link manipulator equation.

The above equation is general equation, if we take uncertainties in consideration and use linear oscillator in gravitational oscillator, the equation will be followed:

$$\frac{1}{3}ma^2\ddot{q} + \frac{1}{2}mgasq + F_c + F_d = \zeta \qquad (3.9)$$

5.5 DYNAMIC OF A 2-LINK ROBOT ARM

The dynamic equations of the 2-link 2-DOF robot, based on euler-lagrange equations are derived. The vector of generalised coordinates is:

$q= [q_1 \ \ q_2]^T$

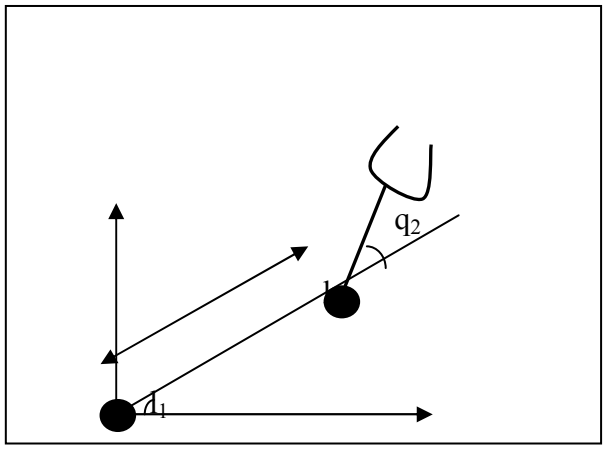

Fig. 3.2 Two-link robot arm

Whereas $l_1/2$ and $l_2/2$, are the distances of the centre of masses of the two links from their respective joint origins, namely at O_1 and O_2. Moreover, let the masses of the two links be m_1 and m_2. With the chosen coordinate frame, i.e. the fixed frame F.

The standard equation of two link robotic manipulator not including uncertainties as follows:

$$I\ddot{q} + h + \gamma = \zeta \qquad (3.10)$$

Where,

$$I = [m_1 l_1^2 + m_2 (l_1^2 + l_2^2 + 2l_1 l_2 \cos(q_2)) \quad m_2 l_2^2 + m_2 l_1 l_2 \cos(q_2)$$
$$m_2 l_2^2 + m_2 l_1 l_2 \cos(q_2) \quad m_2 l_2^2];$$

$$h = [-m_2 l_1 l_2 \sin(q_2)\dot{q}_2^2 - 2m_2 l_1 l_2 \sin(q_2)\dot{q}_1\dot{q}_2$$
$$m_2 l_1 l_2 \sin(q_2)\dot{q}_1^2];$$

$$\gamma = [m_2 l_2 g\cos(q_1+q_2) + (m_1+m_2) l_1 g\cos(q_1)$$
$$m_2 l_2 g\cos(q_1+q_2)];$$

3.6 CONTROLLER FOR ROBOTIC MANIPULATOR

There are many control strategies that can applied for control of robotic manipulator. These strategies are conventional or adaptive and intelligent control strategies. The general structure of a robot manipulator with controller is shown in Fig.3.3 below. The trajectory generator provides the controller with information about the desired position, velocity and acceleration $(q, \dot{q}, \ddot{q})$ for each joint and keeps updating this information at the path update rate. The controller takes this information and compares it with the present (actual) position and velocity (sometimes acceleration also) of joints $(q, \dot{q}, \ddot{q})$, which are provides as feedback through the sensors.

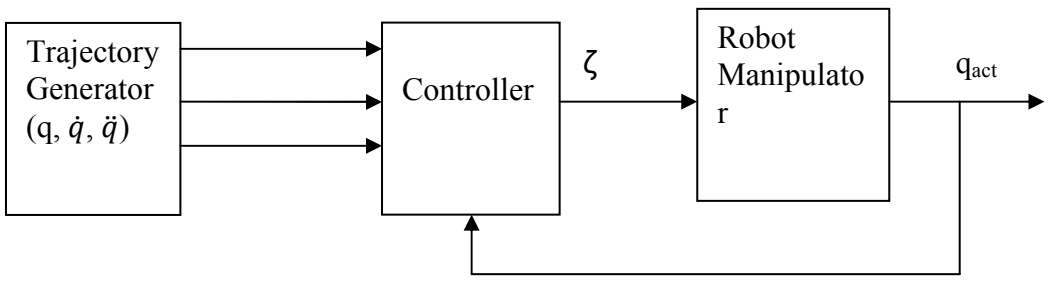

Fig. 3.3 General structure of robot control system

Based upon the error between the desired and actual values. The controller calculates a vector of torques (ζ) , which should be applied at respective joints by the actuators to minimize these errors. The torques is calculated using control law. The goal of the controller is thus, minimization of error, e and its first derivative e. The dynamic model of robotic manipulator is described (previously defined) as below;

$I\ddot{q} + h + \gamma = \zeta$

There are two type controller is used for the control of the robotic manipulator:

3.6.1 Conventional Controller

The use of conventional control (linear control techniques) for any system is valid only when the system to be controlled can be modelled by differential equations. Thus, the conventional control robot manipulator is essentially an approximation, as the manipulator dynamics is described by highly non-linear equations. The linear control strategies for robots give excellent performance for manipulators having highly geared joints.

There are many conventional controller are used in robotic manipulator, some of them are described below:

3.6.1.1 Proportional-Integral-Derivative controller (PID controller)

It is a generic controller widely used in industrial control system. A PID controller attempts to correct the error between a measured process variable and a set-point by calculating and then out putting a corrective action that can adjust the process accordingly and rapidly, to keep the error minimal. It is one common linear control strategy is PID (proportional-derivative and integral) control. The control law used for this strategy is given by:

$\zeta_{PID} = K_D e + K_P e + K_I \int e dt$ (3.11)

K_D, KP and K_I are control gain matrices. ζ_{PID} is the vector of joint torques. It is possible to get the desired performance from the system by choosing the appropriate values of parameters of PID controller. Hand tuning method is used for selection of PID control gains.

A robotic control system cannot be allowed to have an oscillatory response for obvious reasons. For instance, in a pick-n-place operation, an oscillating end-effecter may strike against the object before picking it to manipulate. Hence, highest possible speed of response and yet non-oscillatory response, dictates that the controller design parameters shall be chosen to have the damping ratio equal to unity or least close to it but not less than unity.

3.6.1.2 Feed Forward Inverse Dynamics Control

Feed forward inverse dynamics control is a model based non-linear technique. Scheme for Feed Forward Inverse Dynamics control is shown below Fig. 3.4. This scheme is uses the inverse dynamics equations of robotic manipulator in feed forward mode. As can be seen from this figure, the sum of the outputs of the inverse model and feedback controller (i.e. PID controller) will be the actual input torque to robot.

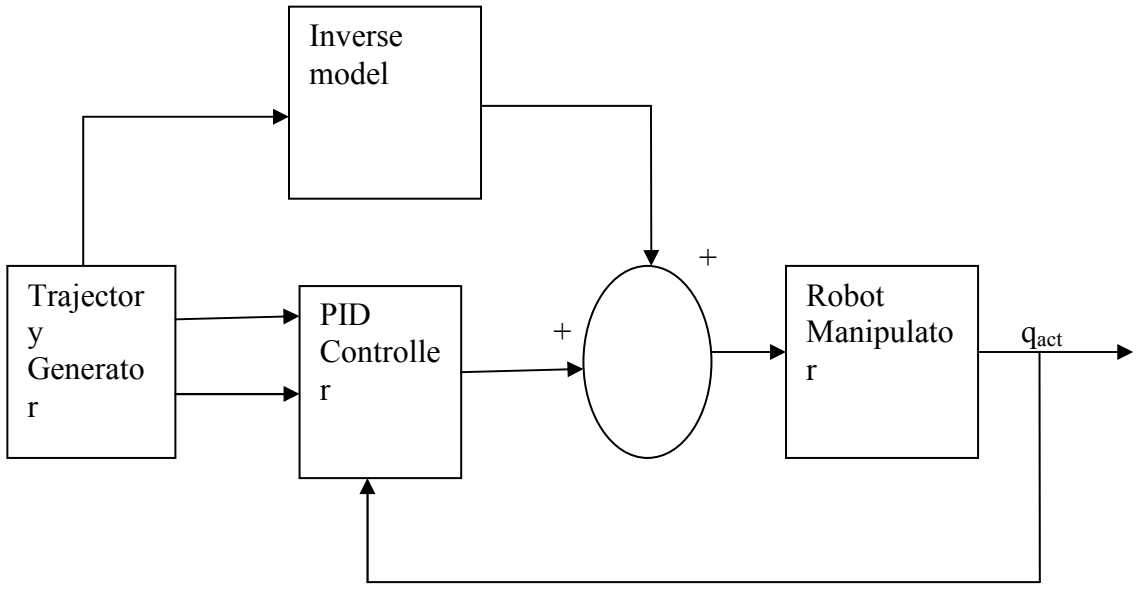

Fig. 3.4 Feed forward inverse dynamics controller

In this strategy the torque is calculated as

$\zeta_{ffid} = I\ddot{q} + h + \gamma$ (3.12)

$\zeta_{PID} = K_D e + K_P e + K_I \int edt$ (3.13)

Total control torque is $\zeta = \zeta_{ffid} + \zeta_{PID}$. The feedback controller plays a role in making the whole system stable.

3.6.1.3 Computed Torque Control

The most common nonlinear control technique for manipulator control is the computed torque control. Scheme is similar to feed forward inverse control. Here the computed torque is given by:

$$\zeta_{CTC} = \zeta_{PID} + I\,[\ddot{q} + K_D e + K_P e] + h + \gamma \quad (3.14)$$

If the manipulator model is known exactly then this scheme results in asymptotically stable and provides asymptotically exact tracking.

3.6.1.4 Critically Damped Inverse Dynamics Control

This control strategy is almost same as inverse dynamics except that the feed forward torque is calculated using reference velocity and reference acceleration instead of the desired values. These reference values are defined as:

$$q_R = q_d + K_P\,(q_d - q) \quad (3.15)$$

$$\ddot{q}_R = \ddot{q}_d + K_D\,(qd-q) \quad (3.16)$$

In this strategy the torque is calculated as:

$$\zeta_{CDID} = \zeta_{PID} + I\ddot{q}_R + h\dot{q}_R + \gamma \quad (3.17)$$

Since the robotic manipulator is highly nonlinear system that why the conventional controller does not give the response with accurately and desirably as well as intelligent controller. In spite of this the conventional controller is used for robotic manipulator because these are cheap and required minimal technical knowledge from the engineer or technician.

3.6.2 Intelligent Controller

Since conventional controller does not respond with accuracy for where the high accuracy is required, there generally intelligent controller. Artificial neural networks and fuzzy logic are potential tools for intelligent control engineering. It is briefly described below:

3.6.2.1 Fuzzy Logic Control

A fuzzy control system is a control system based logic- a mathematical system that analyzes analog input values in terms of logical variables that can take continuous values between 0 and1, in contrast to classical or digital logic, which operates on discrete values of either 0

and1 (true and false). Just as fuzzy logic can be described as simply as "computing with words rather than number". Fuzzy control can described simply as "control with sentences rather than equations". There are specific components characteristics of a fuzzy controller to support a design procedure. In the block diagram in Fig 3.5, the controller is between a pre-processing block and a post-processing block.

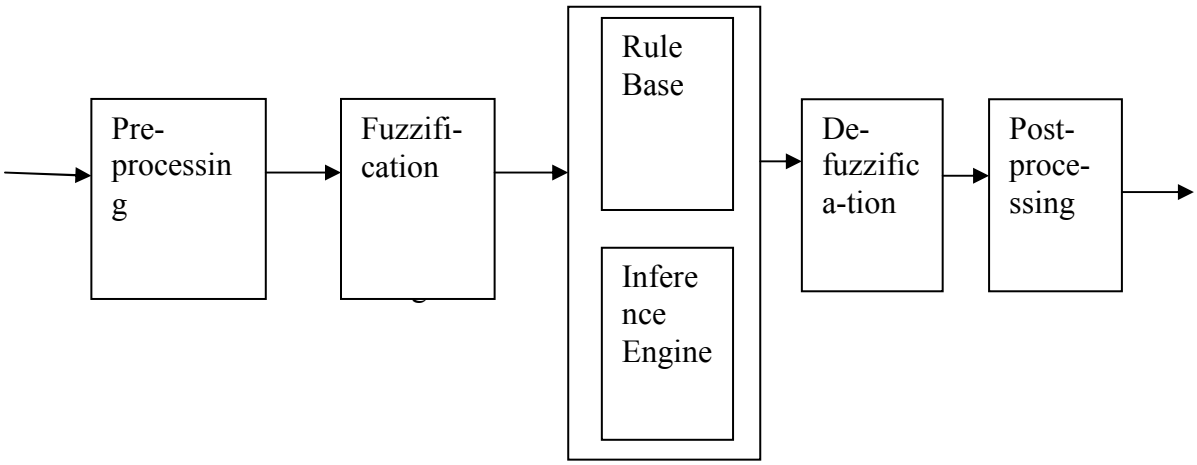

Fig.3.5 Block diagram of a fuzzy controller

3.6.2.2 Neural Control

The key element of this control is the novel structure of the information processing system. It is composed of a large number of highly interconnected processing element (neurons) working in unison to solve specific problems. Artificial neural network's, like people, learning by example. An ANN is configured for a specific application, such as pattern recognition or data classification, through a learning process. Learning is biological systems involves adjustment to synaptic connections that exists between the neurons. This is true of ANN's as well. Neural networks, with their remarkable ability to drive meaning from complicated or imprecise data, can be used to extract patterns and detect trends that are too complex to noticed by either human are computer techniques. A trained neural network can be thought of as an "expert" in the category of information it has been given to analyze.

The neural network has been trained off line to approximate the inverse dynamic model of the robot manipulator. The learning scheme is shown in Fig.3.6.

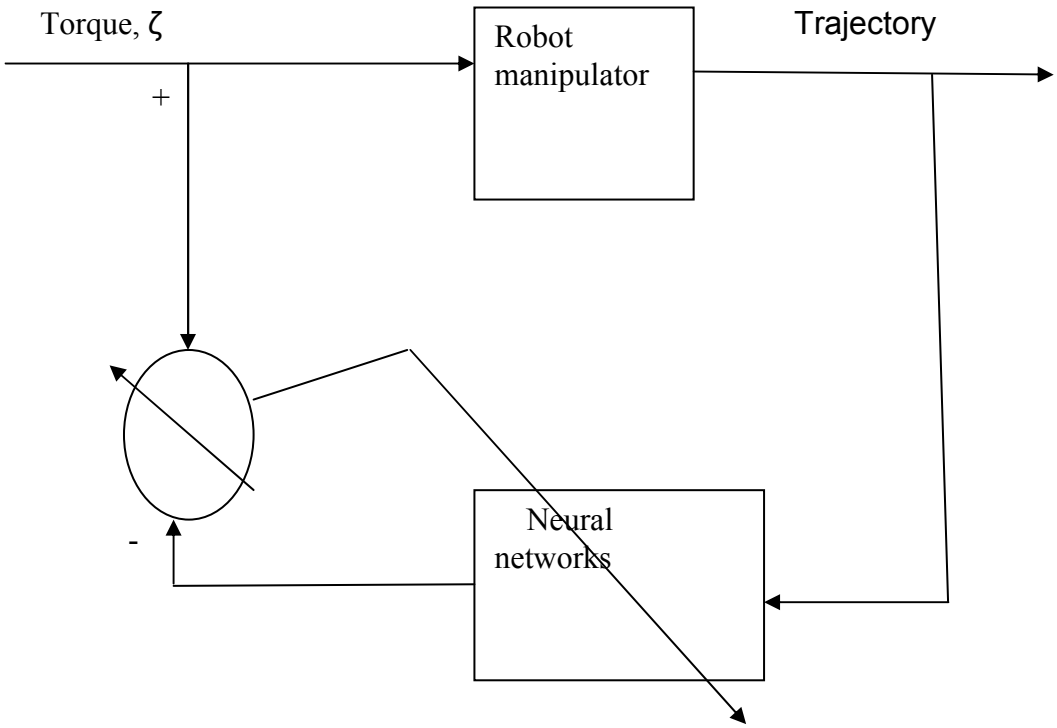

Fig. 3.6 Scheme for learning dynamics model

The manipulator receives the torque ζ and outputs of resulting trajectory q. Inverse dynamic model is set in the opposite input-output direction to that of the manipulator. That is, it receives the trajectory as an input and produces the torque ζ_{NN} as its output. The error signal is the difference between the actual torque and estimated torque. It is expected that this difference tends to zero as learning proceeds. Once the neural network finishes learning, it produces an approximate inverse dynamical model.

The entire controller described above has some limitation, for overcome this limitations; generally, nom-a-days used two or more than two controller simultaneously for a given system. This type of controller is known as hybrid controller. For example, the neural controller with PID controller is used for increase the stability of the whole system.

3.7 ADVANTAGE OF NEURAL CONTROLLER

There are several advantages of neural controller.

1) Ability learning: an ability to learn how to do tasks based on the data given for training or initial experience.

2) Self-organization: an ANN can create its own organization or representation of the information it receives during learning time.
3) Real time operation: ANN computation may be carried out parallel, and special hardware devices are being designed and manufactured which take advantage of this capability.
4) Fault tolerance via redundant information coding: partial destruction of a network leads to the corresponding degradation of performance. However, some network capabilities may be retained even with major network damage.

The advantage of neural controller as mentioned above, to make the interest for researcher and scholars. Now-a-days the research on neural controller is very fluent topic for many fields and many discipline.

3.8 CONCLUSION

In this chapter, a brief discussion presented about kinematics and dynamics. Secondly, the equation of first link and two link manipulator is derived. After that, various type of controller which generally used for the robotic manipulator is described.

CHAPTER IV
NEURAL NETWORKS

4.0 GENERAL

Neural networks provide a unique computing architecture which can be used to realise intelligent controllers. Neural network approach is more capable to conventional computers approach. It may be compared to the human brain. In any neural network design, neuron, transfer function and learning rule play a more important role.

4.1 CONVENTIONAL COMPUTERS APPROACH VERSUS NEURAL NETWORKS APPROACH

Conventional computers use an algorithm approach i.e. the computer follows a set of instructions in order to solve the problem. That restricts the problem solving capability of conventional computers to problems that we already understand and know to solve. But computers would be so more useful if they could do things that we don't exactly know how to do.

Neural networks process information in a similar way the human brain does. The network is compared of a large number of a highly interconnected processing element (neurons) working in parallel to solve a specific problem. Neural networks learn by example. They cannot program to perform a specific task. The example must be selected carefully otherwise useful time is wasted or even worse the network might be functioning incorrectly. The disadvantage is that because the networks finds out how to solve the problem by itself, its operation can be unpredictable. On the other hand, conventional computers use a cognitive approach to problem solving; the way the problem is to solved must be known and stated in small unambiguous instructions. These instructions are then converted to high level language program and into machine code that the computer can understand. These machines are totally predictable; if anything goes wrong is due to the software or hardware fault. Neural networks and conventional algorithmic computer are not in competition but complement each other. There are tasks are more suited to an algorithmic approach like arithmetic operations and tasks that are more suited to neural networks. Even more, a large number of tasks, require system that use a combination of the two approaches (normally a conventional computer is

used to supervise the neural networks) in order to perform at maximum efficiency. Neural networks do not perform miracles. But if used sensibly they can produce some amazing results.

4.2 BIOLOGICAL AND ARTIFICIAL NEURAL SYSTEMS

The brain consists of a large number (approximately10^{11}) of highly connected elements (approximately 10^4 connections per element) called neurons. For our purpose these neurons have three principal components: the dentries, the cell body and the axon. The dentries are tree-like respective networks of nerve fibres that carry electrical signal into the cell body. The cell body effectively sum and thresholds these incoming signals. The axon is single long fibre that carries the signal from the cell body out to other neurons. The point of contact between an axon of a cell and a dentrie of another cell is called synapse. It is the arrangement of neurons and the strengths of the individual synapses, determined by a complex chemical process that establishes the function of the neuron networks.

Artificial neural networks do not approach the complexity of the brain. There are, however, two key similarities between biological and artificial neural networks. First, the building blocks of the both networks are simple computational devices (although artificial neurons are much simpler than biological neurons) that are highly interconnected. Second, the connections between neurons determine the function of the network.

4.3 NEURON MODEL

A single- input neuron model is shown in Fig. 4.1. The scalar input p is multiplied by the scalar weight w to form wp, one of the terms that is sent to the summer. The other input, 1, is multiplied by a bias b and then passed to the summer. The summer output n, often referred to as the net input, goes into a transfer function f, which produces the scalar neuron output a. The actual output depends on the particular transfer function that is chosen. (Some authors use the term “activation function” rather than transfer function and “offset” rather than bias.)

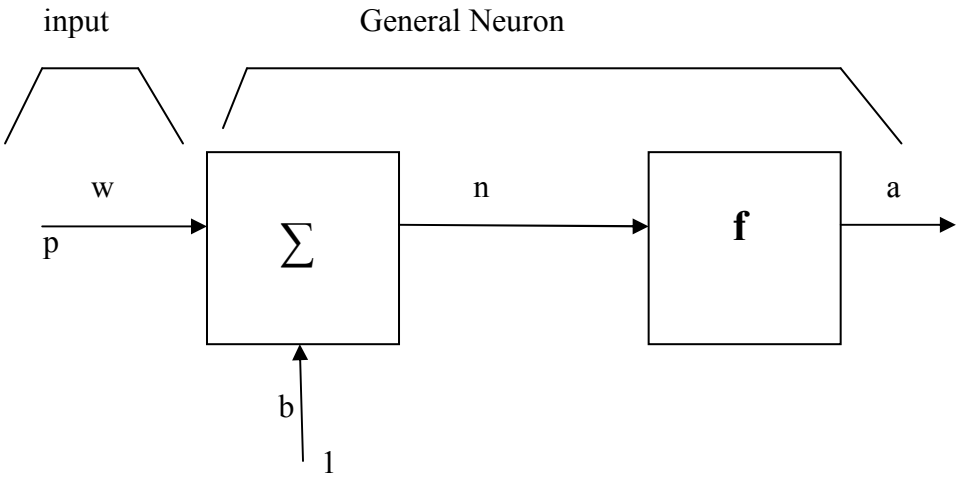

Fig. 4.1 Single input neuron

The neuron output is calculated as,
a=f (wp+b)

If relate this simple model to the biological neuron. The weight w corresponds to the strength of the synapse, the cell body is represented by the simulation and the transfer function, and the neuron output a represents the signal on the axon.

This is the simplest neuron model; we can make the complicated neuron by many ways as following methods-

1) By the increasing the number of the inputs
2) By the increasing numbers neurons in each layer
3) By the increasing of layers (cascading of two or more neuron model).

In spite of these simple methods, to make a very complicated neural network by various others complicated methods.

Here the most important thing is that w and b are both adjustable scalar parameters of the neuron. Typically the transfer function is chosen by the designer and then the parameters w and b will be adjusted by the some learning rule so that the neuron input/output relationship meet some specific goal.

4.4 TRANSFER FUNCTION

The transfer function may be linear or a nonlinear function of net input, n. A particular transfer function is chosen to satisfy some specification of the problem that the neuron is attempting to solve. A variety of standard transfer functions is followed:

Name	Input/output relation	MATLAB
Hard limit	a=0; n<0 a=1; n≥0	hardlim
Symmetrical Hard Limit	a=-1; n<0	hardlims
Linear	a=+1; n≥0	purelin
Saturating linear	a=n	satlin
Symmetric Saturating Linear	a=0; n<0 a=n; 0≤n≤1 a=1; n>1	satlins
Log-Sigmoid	$a=1/(1+e^{-n})$	logsig
Hyperbolic Tangent Sigmoid	$a=(e^{n}-e^{-n})/(e^{n}+e^{-n})$	tansig
Positive Linear	a=0; n<0 a=n: 0≤n	poslin
Competitive	a=1; neuron with max n a=0; all other neuron	compet

Table 4.1 Standard transfer function

4.5 LEARNINING RULE

By learning rule means a procedure for modifying the weights and biases of a network. (This procedure may also be referred to as a training algorithm). The purpose of the learning rule is to the train the network to perform some specific task. There are many types of neural network learning rules, but the back-propagation algorithm is the most popular learning rule.

4.6 THE BACK-PROPAGATION ALGORITHM

In order to train a neural network to perform some task, we must adjust the weights of each unit in such a way that the error between the desired output and the actual output is reduced. This process requires that the neural network compute the error derivative of the weights (dW). In other words, it must calculate how the error changes as each weight is increased or decreased slightly. The back propagation algorithm is the most widely used method for determining the dW.

The back-propagation algorithm is easiest to understand if all the units in the network are linear. The algorithm computes each dW by the first computing the error derivative (EA), the rate at which the error changes as the activity level of a unit is changed. For output units, the EA is simply the difference between the actual and the desired output. To compute the EA for a hidden unit in the layer just before the output layer, we first identify all the between that hidden unit and the output units to which it is connected. We then multiply those weights by the EAs of those output units and add the products. This sum equals the EA for the chosen hidden unit. After calculating all EAs in hidden layer just before the output layer, we can compute in like fashion the EAs for other layers, moving from layer to layer in a direction opposite to the way activities propagate through the network. This is what gives back propagation its name. Once the EAs has been computed for a unit. It is straight forward to compute the dW for each incoming connection of the unit. The dW is the product of the EA and the activity through the incoming connection. Note that for non-linear units, the back-propagation algorithm includes an extra step. Before back-propagation, the EA must be converted into EI, the rate at which the error changes as the total input received by a unit is changed. There are various training function which is used in back-propagation is following:

1) Traingd: gradient descent back propagation
2) Traingda: gradient descent with adaptive learning rate back propagation
3) Traingdm: gradient descent with momentum back propagation
4) Traingdx: gradient descent with momentum and adaptive learning rate back propagation
5) Trainlm: levenberg-marquardt back propagation

4.7 IMPLEMENTATIONS OF NEURAL NETWORKS

Implementations of neural networks come in many forms. The most widely used implementations of neural networks today are software simulators, computer programs that stimulate the operation of neural networks. The speed of the simulation depends on the speed of the hardware upon which the simulation is executed. A variety of accelerator boards are available for individual computers to speed the computations; math processors, and the parallel processors may also be used.

Simulation is key to the development of neural network technology. With a simulator, one can establish most of the design choices in a neural network system. The choice of inputs and out puts can be tested as well as the capabilities of the particular paradigm used. Realistic training sets can be tested in simulated mode.

An implementation could be an individual calculating the changing parameters of the network using pencil and paper. Another implementation would be a collection of people, each one acting as a processing unit, using a hand-held calculator. Although these implementations are not fast enough to be effective for applications, they are nevertheless methods for emulating a parallel computing structure based on neural architectures. The response of an artificial neural networks simulation may be accelerated through the use of specialized hardware. Such hardware may be designed using analog computing technology or a combination of analog and digital.

4.8 CONCLUSION

In this chapter, the neural network is related with neural network. A simple neuron model and its related term such as transfer function and learning rule is also described. At the last, the back-propagation and importance of simulation for neural network is presented.

CHAPTER V
RESULTS AND DISCUSSION

5.0 GENERAL

The performance of the inverse dynamic model of one-link and two-link robotic manipulator is studied with neural controller through simulation in MATLAB. Various performances of robotic manipulator are analyzed in detail by the variation of neural controller parameter and robotic manipulator.

A high performance system consists of a robot and a controller integrated to perform a precise mechanical manoeuvre. This requires the end-effecter speed and/or position of the robot to clearly follow a specified trajectory (may be say reference trajectory) regardless of unknown load variations and other parameter uncertainties.

A back-propagation neural network can be trained to emulate the unknown nonlinear plant dynamics by presenting a suitable set of input/output patterns generated by the plant. Once system dynamics have been identified using a neural network, many conventional control techniques can be applied to achieve the desired objective of trajectory tracking.

5.1 NEURAL CONTROLLER FOR ONE-LINK MANIPULATOR SIMULATION

For simulation of neural controller of one-link manipulator, a robotic manipulator with linear oscillator and uncertainties is considered. MATLAB coding is done on the basis of the inverse dynamics of the robotic manipulator. Effect of parameter variation of neural controller and robotic manipulator is presented.

The following parameters values are used for simulation.

Mass of the robotic manipulator =1kg

Length of the robotic manipulator = 1 metre

Coefficient of the continuous uncertainties, c = 0.2

Coefficient of the discontinuous uncertainties, Fd = 0.03

Transfer function of the first layer of neural network = purelin

Transfer function of the second layer of neural network = satlins

Number of the neuron in first layer of neural network = 5

Number of the neuron in second layer of neural network = 1

Training method used in neural network = trainlm

5.2 EFFECT OF THE VARIATION OF FIRST LAYER TRANSFER FUNCTION OF NEURAL NETWORK

For designing point of view, the neural network for robotic manipulator, the proper transfer function for each layer is very important. Since MATLAB coding for neural controller of one-link robotic manipulator with two layer neural networks is used. The first layer transfer function is changed as standard transfer function as mentioned in table. 1 and second layer assumed as hyperbolic tangent sigmoid transfer function (in MATLAB coding it is represented by 'tansig'), the following performance characteristics are obtained, by the choosing transfer function as mentioned below.

a) The first layer transfer function of the neural network is competitive transfer function "compat" and the second layer is hyperbolic tangent sigmoid transfer function "tansig". The following performance is obtained.

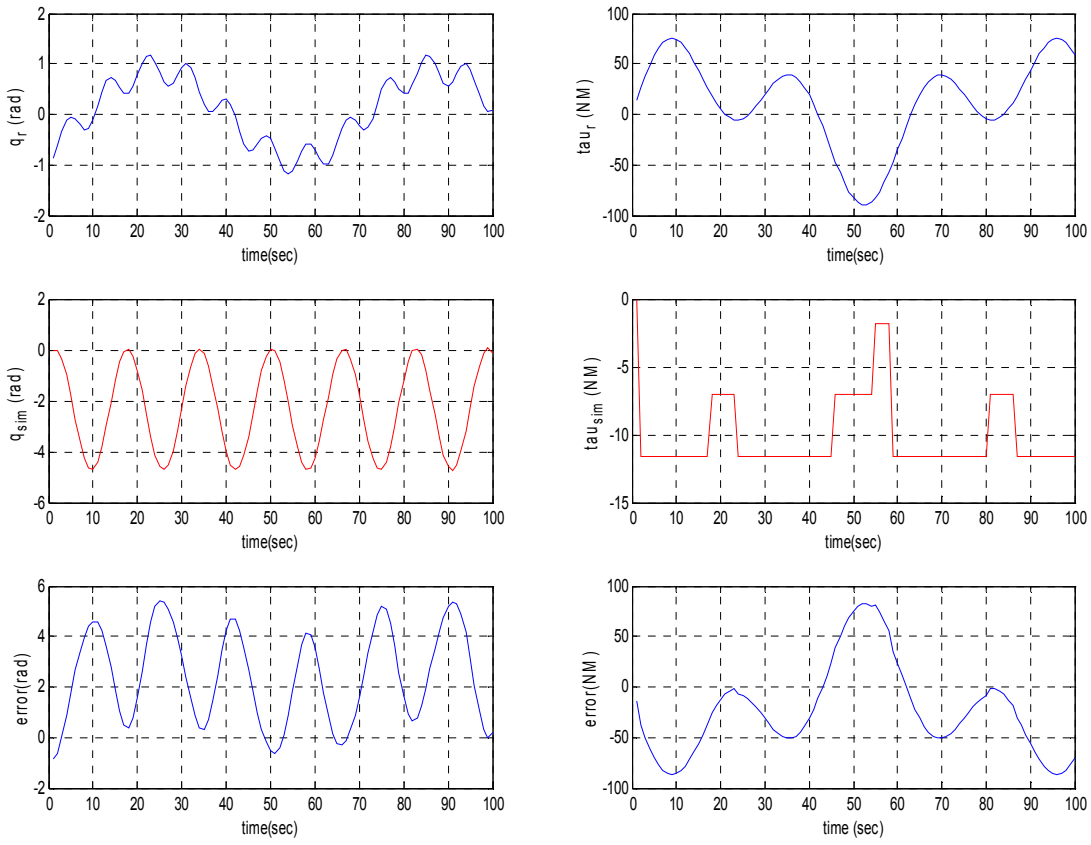

Fig. 5.1 Performance with "compet,tansig" transfer function

In the above performance, the reference trajectory (q_r) and the reference torque (tau_r) is not followed by the simulated trajectory (q_{sim}) and simulated torque (tau_{sim}). Its means this mechanism is not effective.

b) The first layer transfer function of the neural network as the hard limit transfer function "hardlim" and the second layer is hyperbolic tangent sigmoid transfer function "tansig". The following performance is obtained.

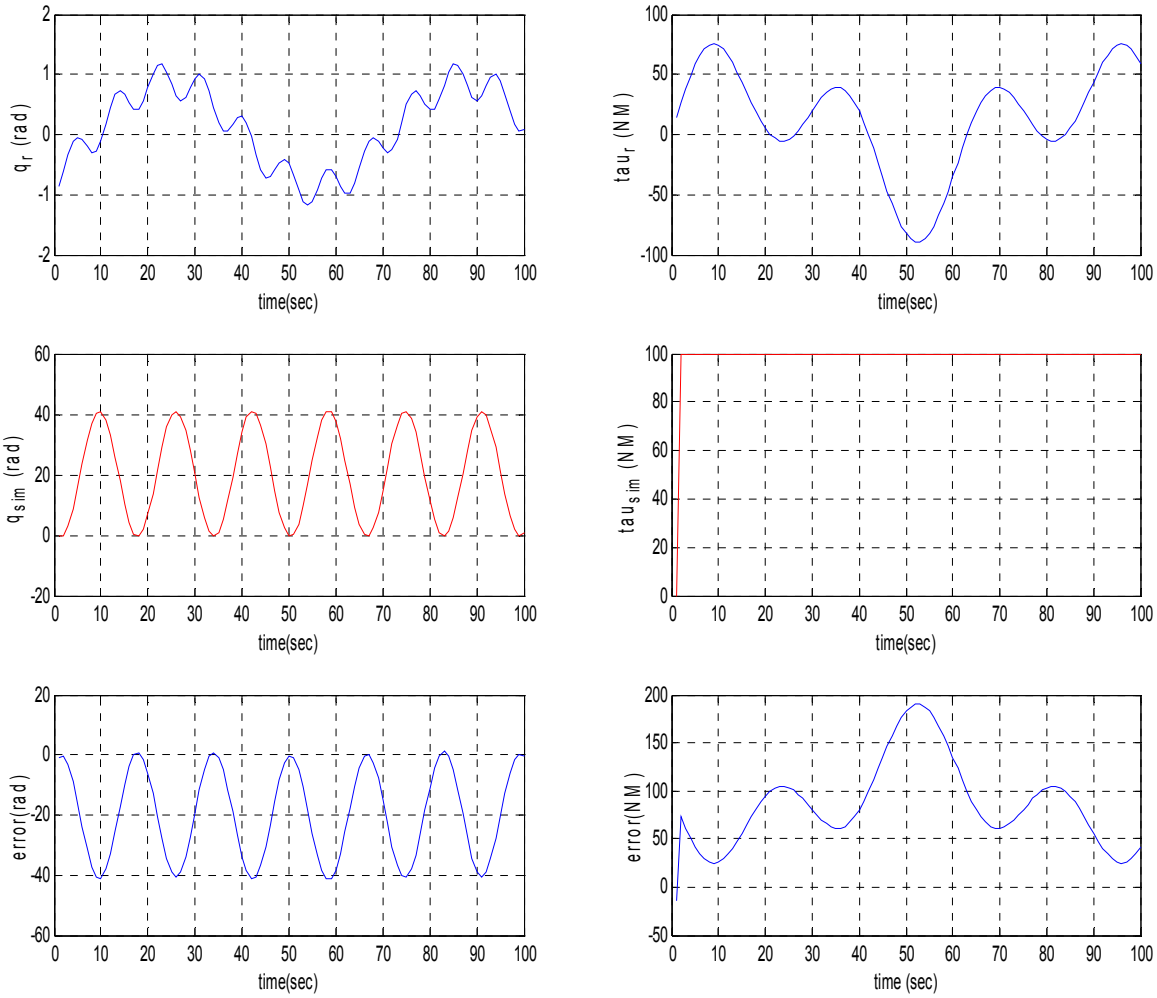

Fig. 5.2 Performance with "hardlim,tansig" transfer function

In the above performance, the reference trajectory (q_r) and the reference torque (tau_r) is not followed by the simulated trajectory (q_{sim}) and simulated torue (tau_{sim}). Its shows that tracking capability and identification capability are not satisfactory.

c) The first layer transfer function of the neural network as the symmetrical hard limit transfer function “hardlims” and the second layer is hyperbolic tangent sigmoid transfer function “tansig”. The following performance is obtained.

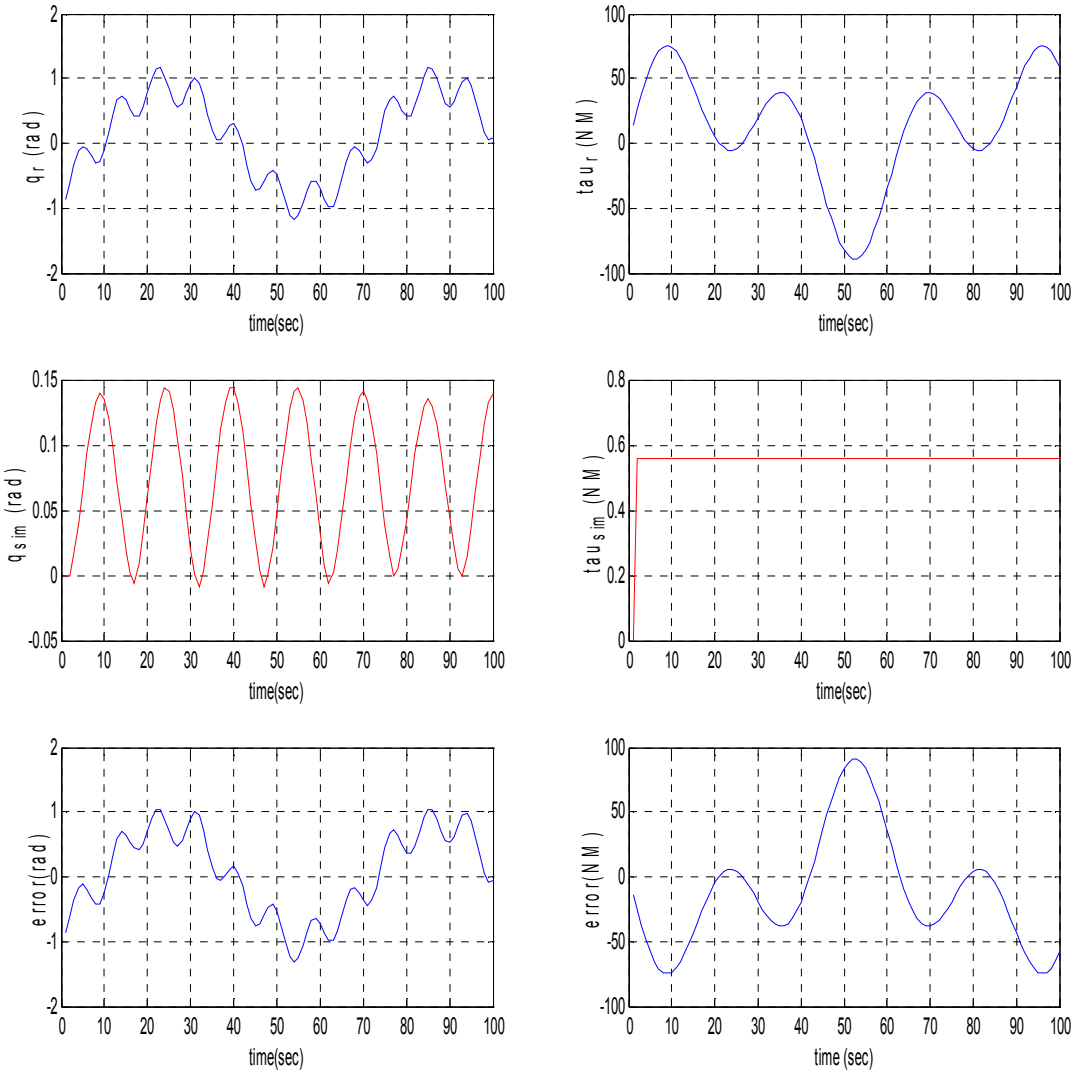

Fig. 5.3 Performance with “hardlims,tansig” transfer function

In the above performance, the reference trajectory (qr) and the reference torque (taur) is not followed by the simulated trajectory (qsim) and simulated torue (tausim). This is also not appropriate tracking and identification capability.

d) The first layer transfer function of the neural network as the linear transfer function "purelin" and the second layer is hyperbolic tangent sigmoid transfer function "tansig". The following performance is obtained.

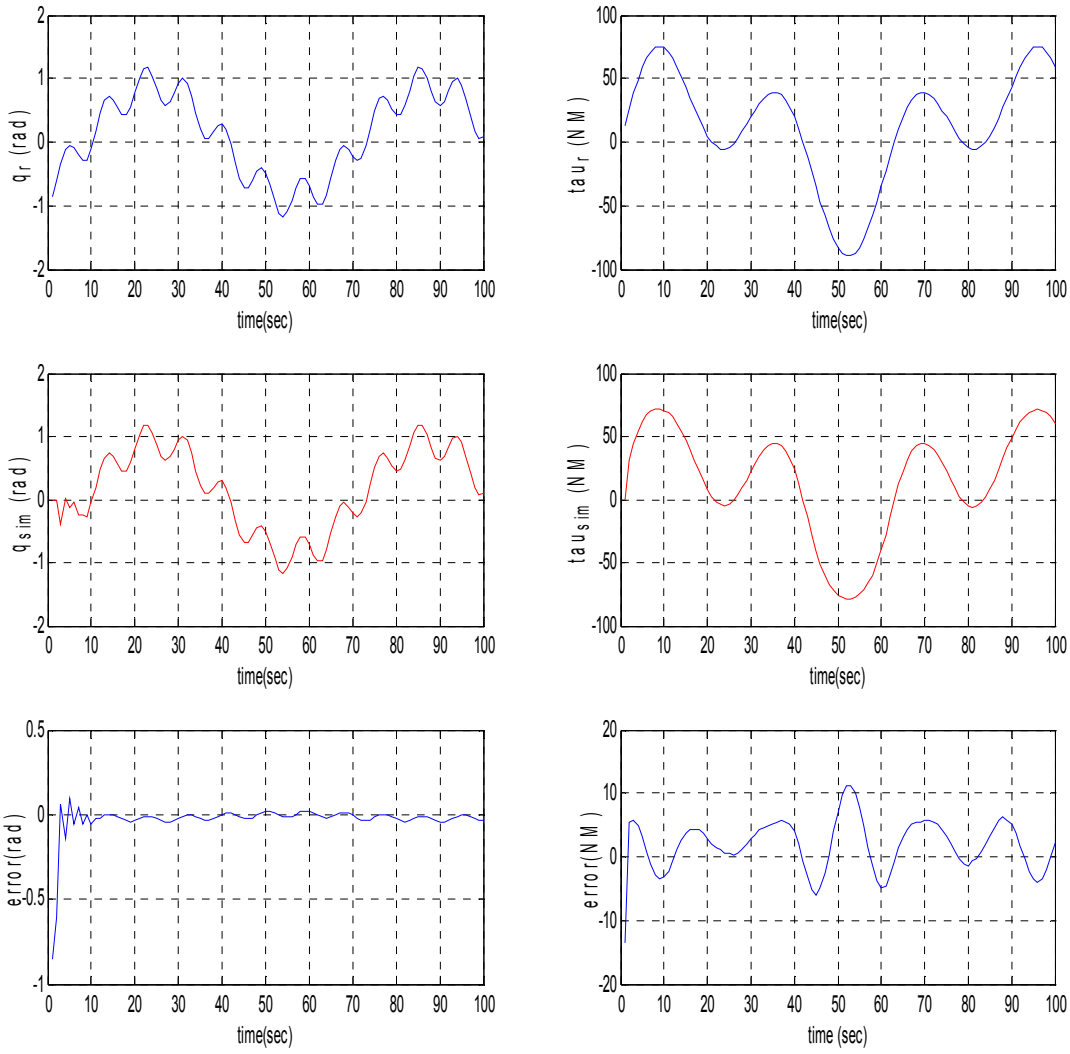

Fig. 5.4 Performance with "purelin,tansig" transfer function

In the above performances, the simulated trajectory is almost followed the reference trajectory, its means it has very good tracking capability but the simulated torque is not followed the reference torque sharply, its means it does not have very good identification capability.

e) The first layer transfer function of the neural network as the saturating linear transfer function "satlin" and the second layer is hyperbolic tangent sigmoid transfer function "tansig". The following performance is obtained.

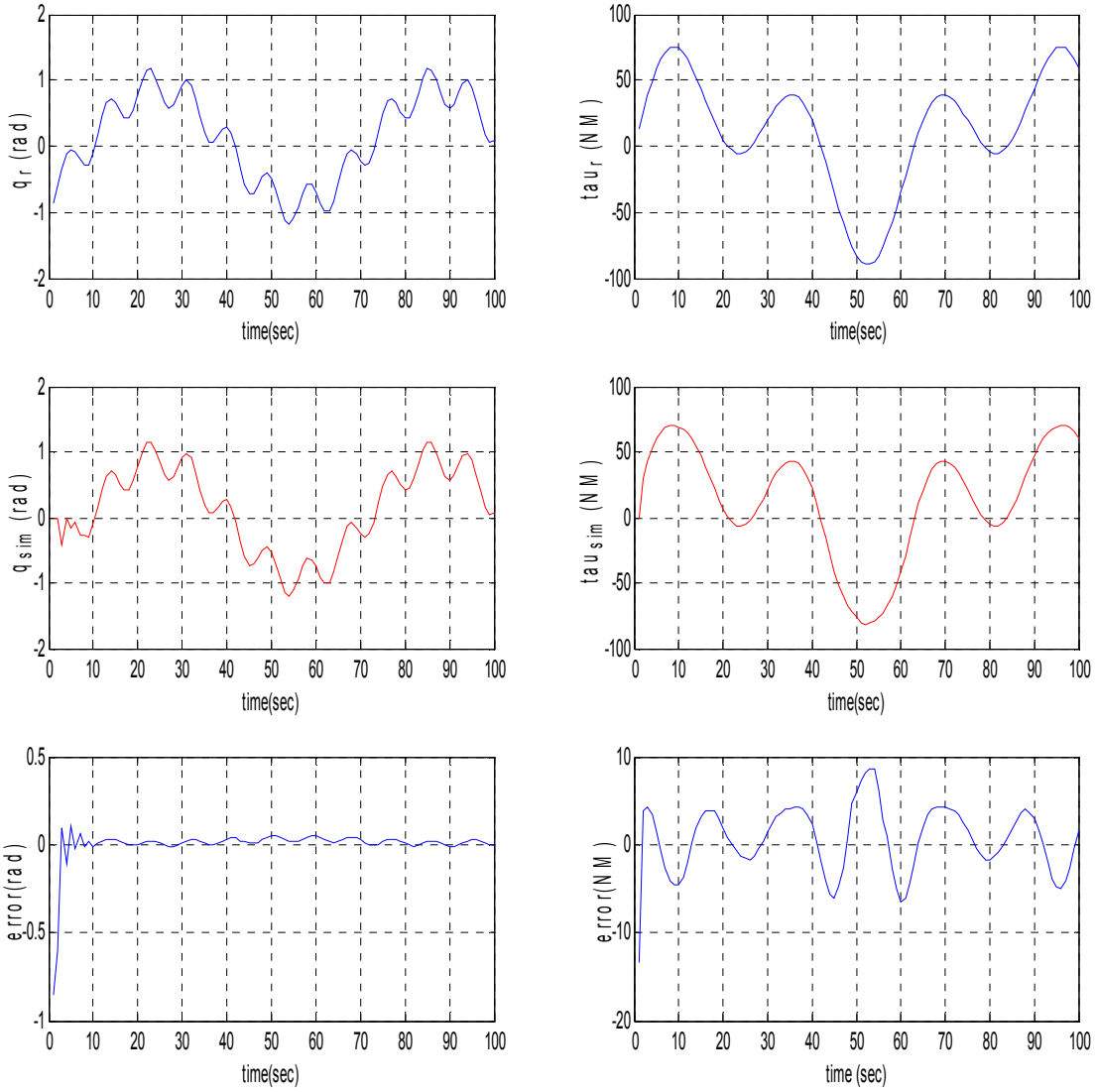

Fig. 5.5 Performance with "satlin,tansig" transfer function

In the above performance, the simulated trajectory is almost followed the reference trajectory, its means neural controller show good tracking capability but the simulated torque is not followed the reference torque sharply, its means it does not have very good identification capability.

f) The first layer transfer function of the neural network as the log-sigmoid transfer function "logsig" and the second layer is hyperbolic tangent sigmoid transfer function "tansig". The following performance is obtained.

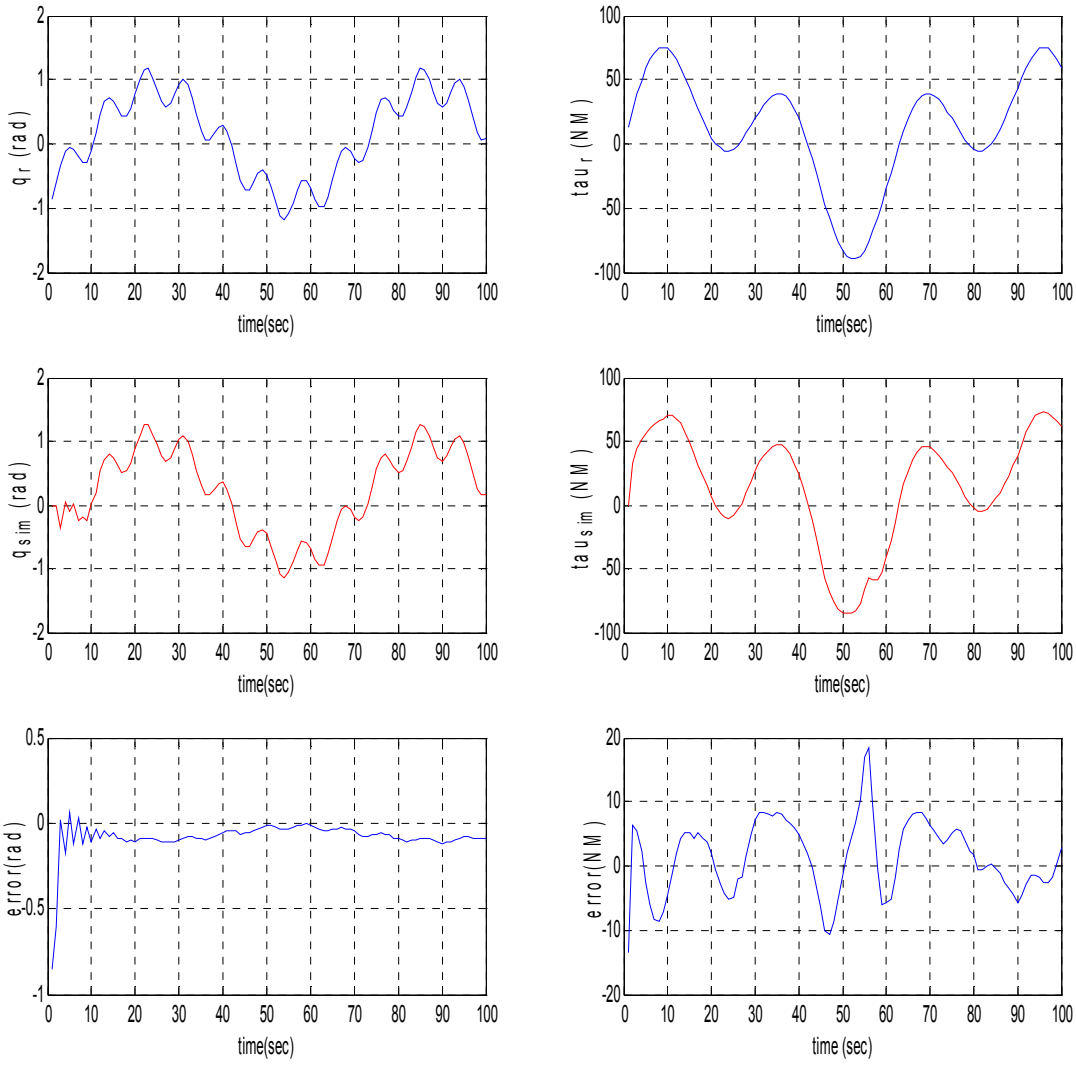

Fig. 5.6 Performance with "logsig,tansig" transfer function

In the above performance, the simulated trajectory is almost followed the reference trajectory, its means neural controller show good tracking capability but the simulated torque is not followed the reference torque sharply, its means it does not have very good identification capability.

g) The first layer transfer function of the neural network as the hyperbolic tangent sigmoid transfer function “tansig” and the second layer is hyperbolic tangent sigmoid transfer function “tansig”. The following performance is obtained.

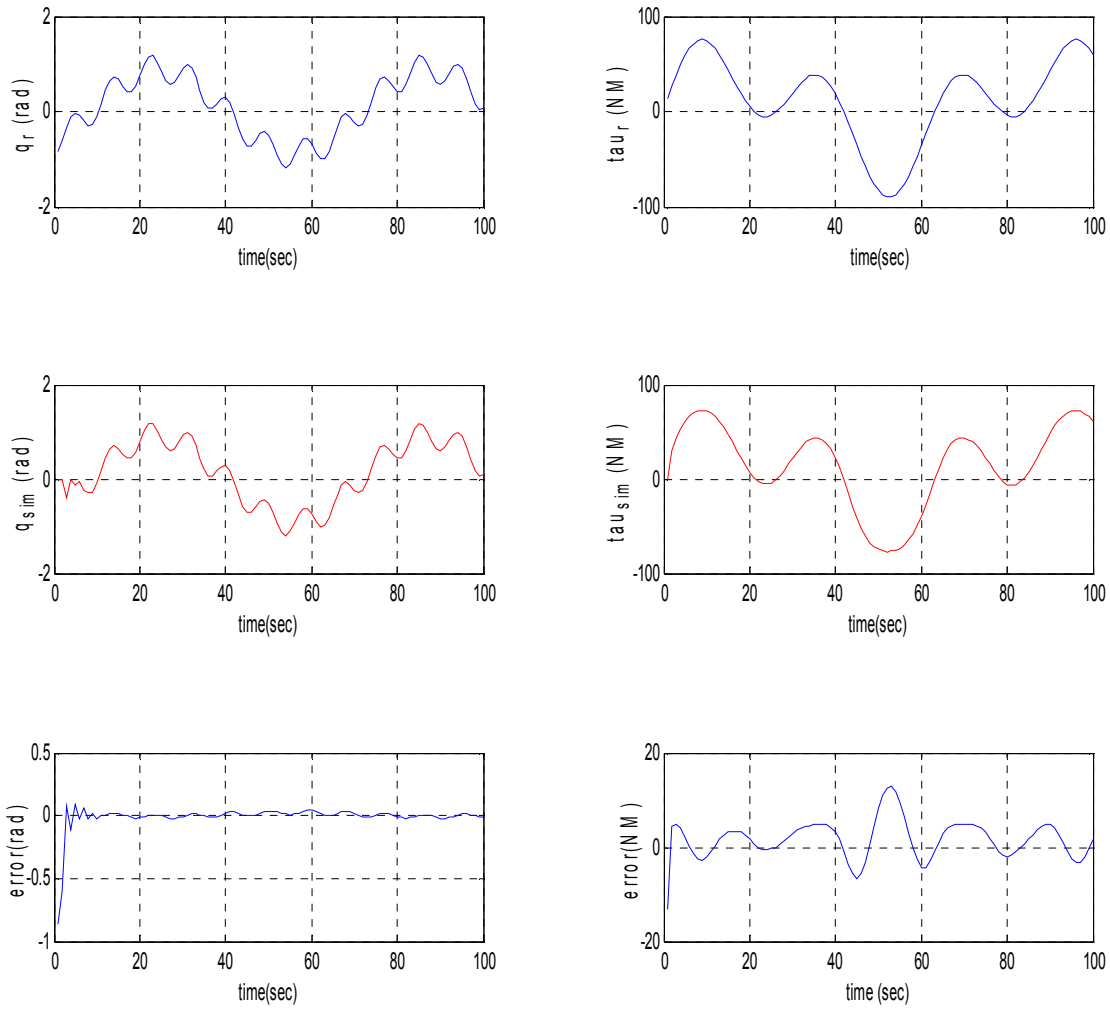

Fig. 5.7 Performance with “tansig,tansig” transfer function

In the above performance, the simulated trajectory is almost followed the reference trajectory, its means neural controller show good tracking capability but the simulated torque is not followed the reference torque sharply, its means it does not have very good identification capability.

h) The first layer transfer function of the neural network as the positive linear transfer function “poslin” and the second layer is hyperbolic tangent sigmoid transfer function “tansig”. The following performance is obtained.

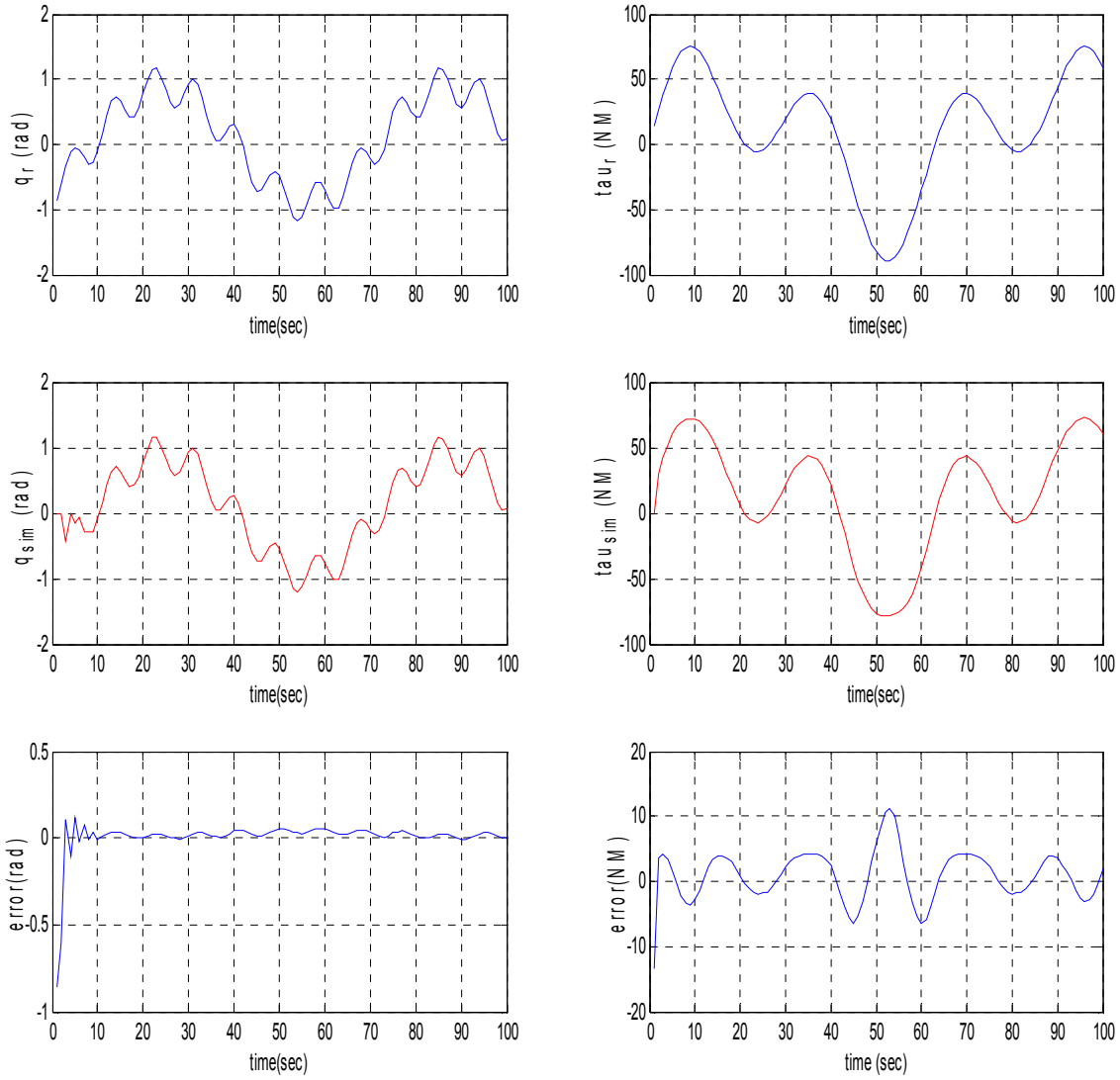

Fig. 5.8 Performance with “poslin,tansig” transfer function

In the above performance, the simulated trajectory is almost followed the reference trajectory, its means neural controller show good tracking capability but the simulated torque is not followed the reference torque sharply, its means it does not have very good identification capability.

i) The first layer transfer function of the neural network as the symmetrical hard limit transfer function "satlins" and the second layer is hyperbolic tangent sigmoid transfer function "tansig". The following performance is obtained.

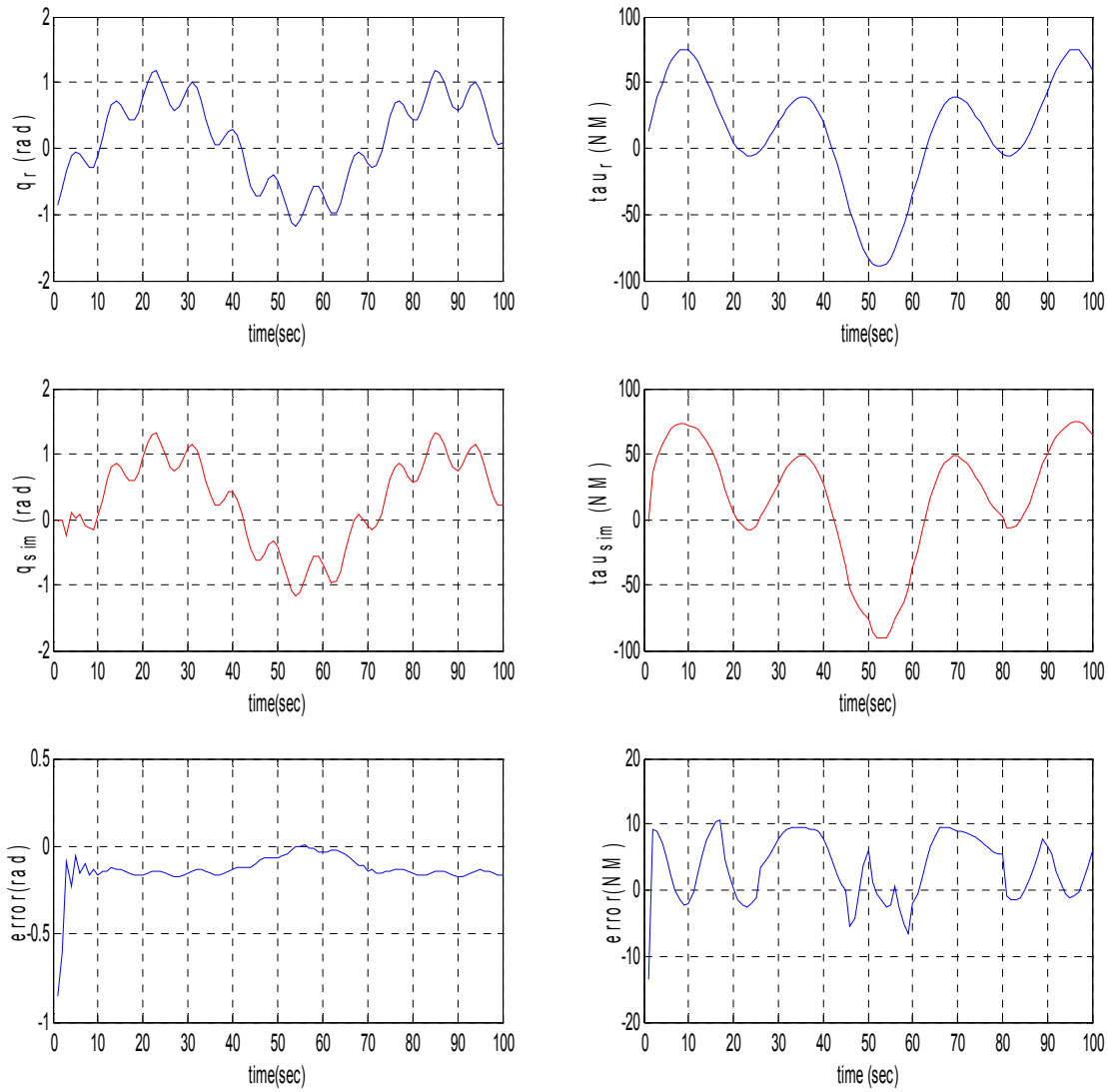

Fig. 5.9 Performance with "satlins,tansig" transfer function

In the above performance, the simulated trajectory is almost followed the reference trajectory, its means neural controller show good tracking capability but the simulated torque is not followed the reference torque sharply, its means it does not have very good identification capability.

5.2.1 **SUMMARY**

For the second layer with “tansig” transfer function and change the first layer, it is observed that:

a) “compet”, “hardlim”, “hardlims” has very bad tracking capability as well as very bad Neural network identification capability.
b) “purelin”, “satlin”, “logsig”, “tansig”, “satlins”, “poslin” has good tracking capability and approximate good neural network identification capability.

It may conclude that the first layer of neural network for given system should not be “compat”, “hardlim”, “hardlims” transfer function. We may be used “purelin”, “satlin”, “logsig”, “tansig”, “satlins”, “poslin” as first layer of transfer function.

5.3 EFFECT OF THE VARIATION OF SECOND LAYER TRANSFER FUNTION OF NEURAL NETWORK

In neural controller design the transfer function and training function method play an important role. Self defined function or the standard function for transfer function and training method of neural controller are used for simulation study. A neural controller design the training method “trainlm” is the first choice of the any designer due to its fast response. But the choice of the transfer function of the neural controller is a complicated work for any designer.

As previously mentioned, for designing point of view for neural network, the transfer function of each layer must be properly chosen. Since a MATLAB coding for neural controller of one-link manipulator, there are two layer neural networks is used. The variation and effect of the first layer transfer function is shown in the previous section, in this we will see the effect of the variation of the second layer transfer function. If we change the second layer transfer function as standard transfer function as mentioned in table. 1 and the first layer assumed as linear transfer function (in MATLAB coding it is represented by ‘purelin’), the following graph is displayed, by the choosing transfer function as mentioned below.

a) The second layer transfer function of the neural network as the competitive transfer function “compet” and the first layer is linear transfer function “purelin”. The following performance is obtained.

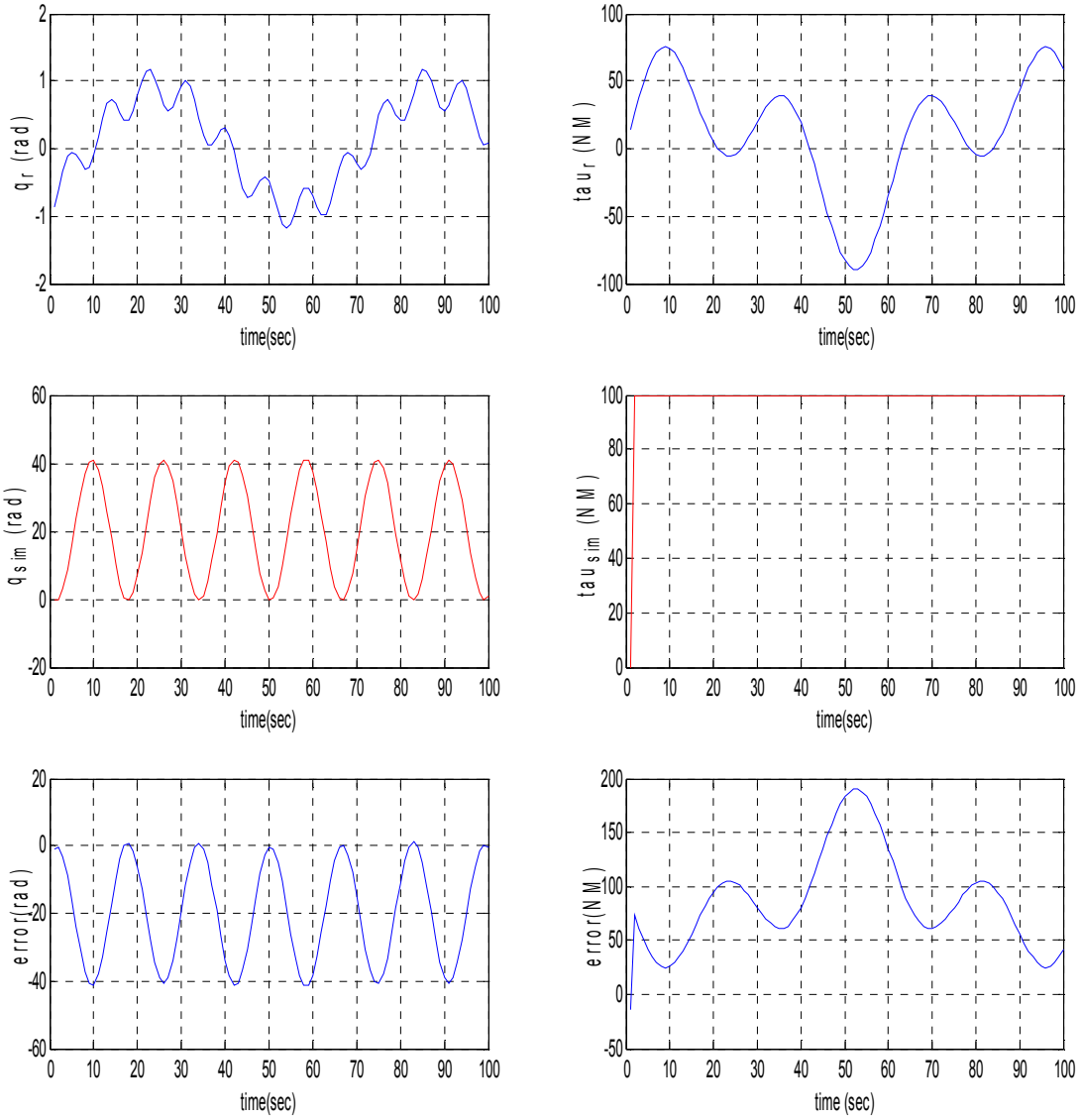

Fig. 5.10 Performance with “purelin,compet” transfer function

In the above performance, the reference trajectory (q_r) and the reference torque (tau_r) is not followed by the simulated trajectory (q_{sim}) and simulated torque (tau_{sim}). Its means has not satisfactory tracking capability as well as identification capability.

b) The second layer transfer function of the neural network as the hard limit transfer function "hardlim" and the first layer is linear transfer function "purelin". The following performance is obtained.

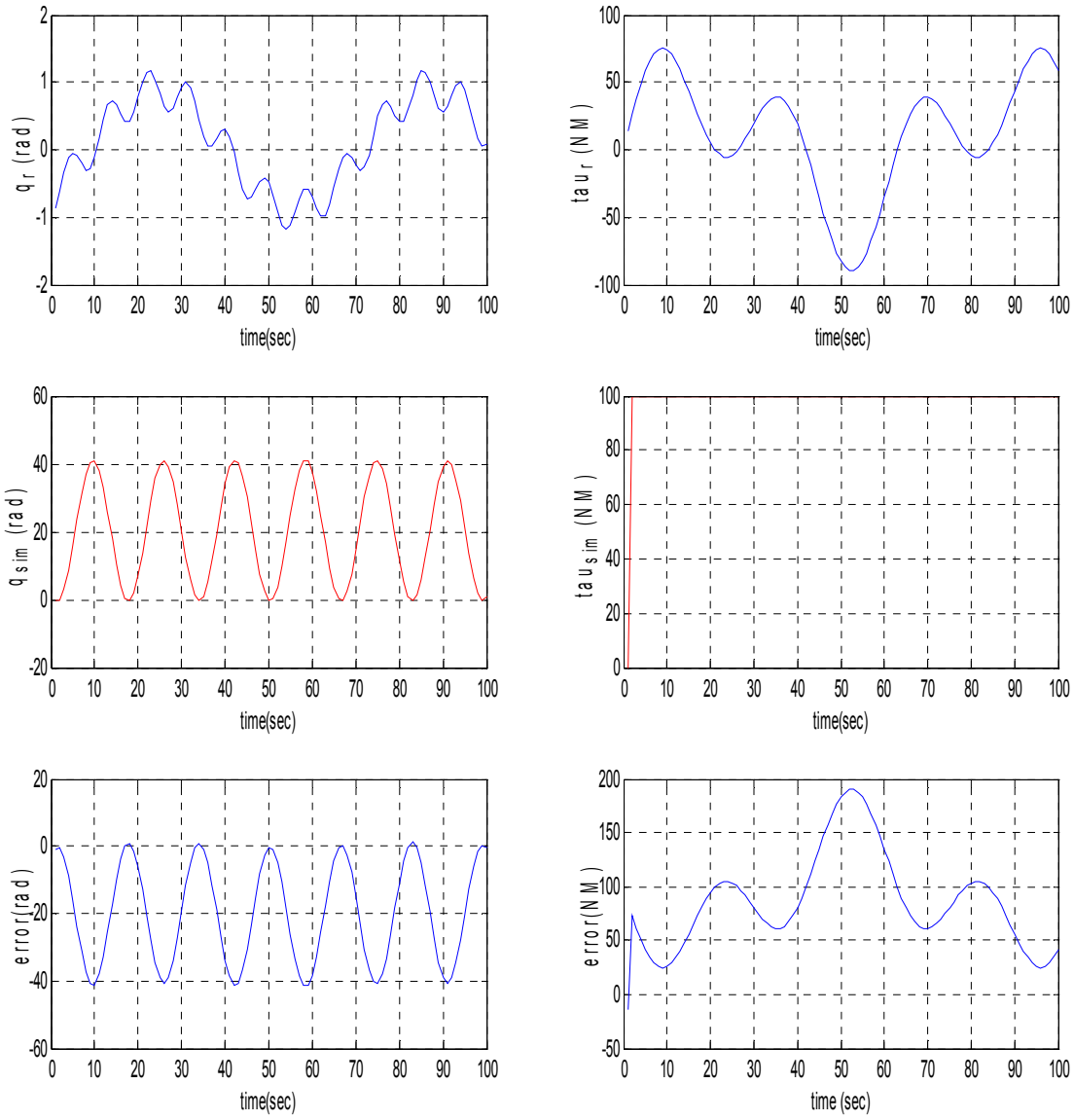

Fig. 5.11 Performance with "purelin,hardlim" transfer function

In the above performance, the reference trajectory (qr) and the reference torque (taur) is not followed by the simulated trajectory (qsim) and simulated torque (tausim). Its means has not satisfactory tracking capability as well as identification capability.

c) The second layer transfer function of the neural network as the symmetrical hard limit transfer function “hardlims” and the first layer is linear transfer function “purelin”. The following performance is obtained.

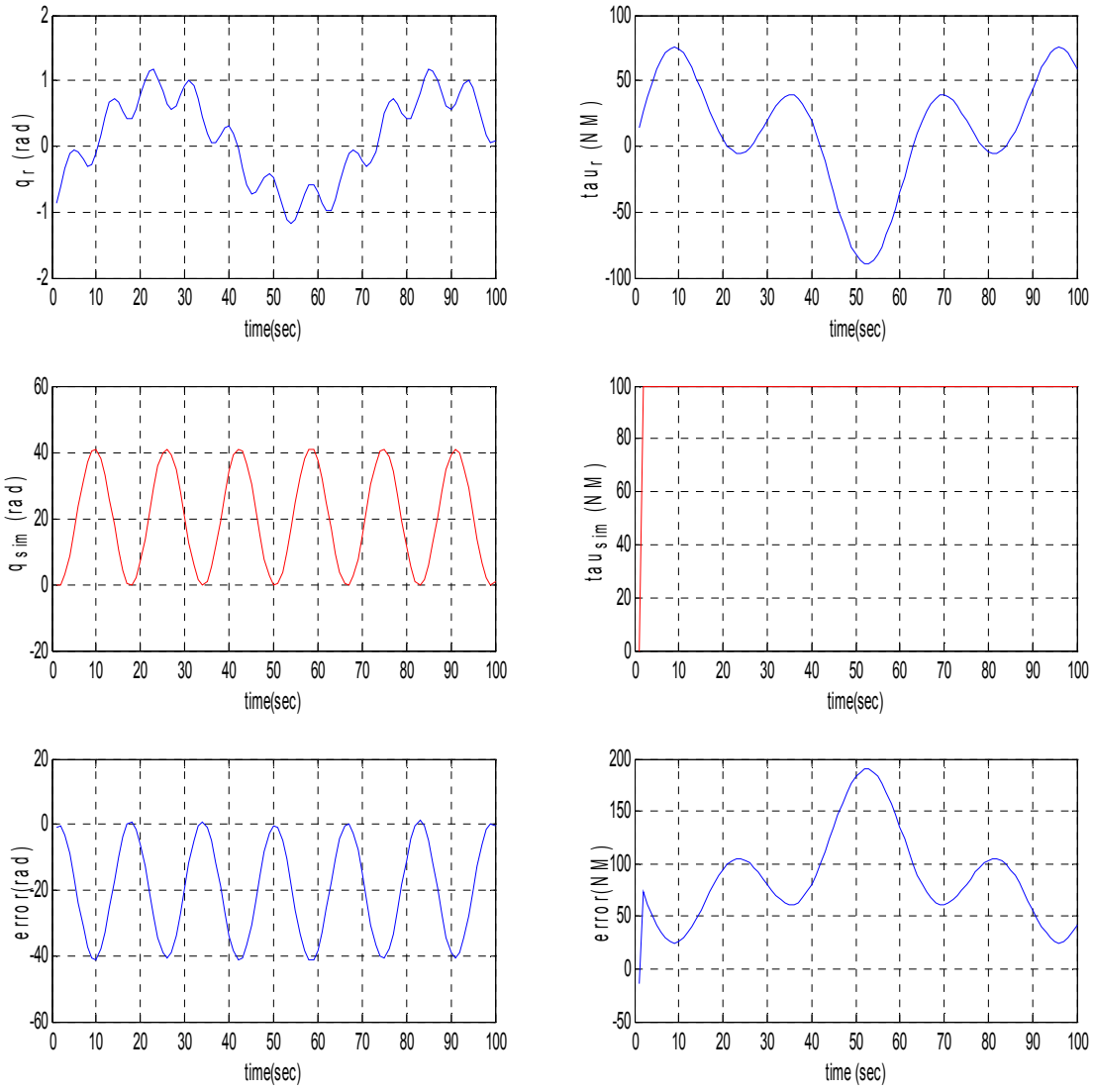

Fig. 5.12 Performance with “purelin,hardlims” transfer function

In the above performance, the reference trajectory (qr) and the reference torque (taur) is not followed by the simulated trajectory (qsim) and simulated torque (tausim). Its means has not satisfactory tracking capability as well as identification capability.

d) The second layer transfer function of the neural network as the linear transfer function "purelin" and the first layer is linear transfer function "purelin". The following performance is obtained.

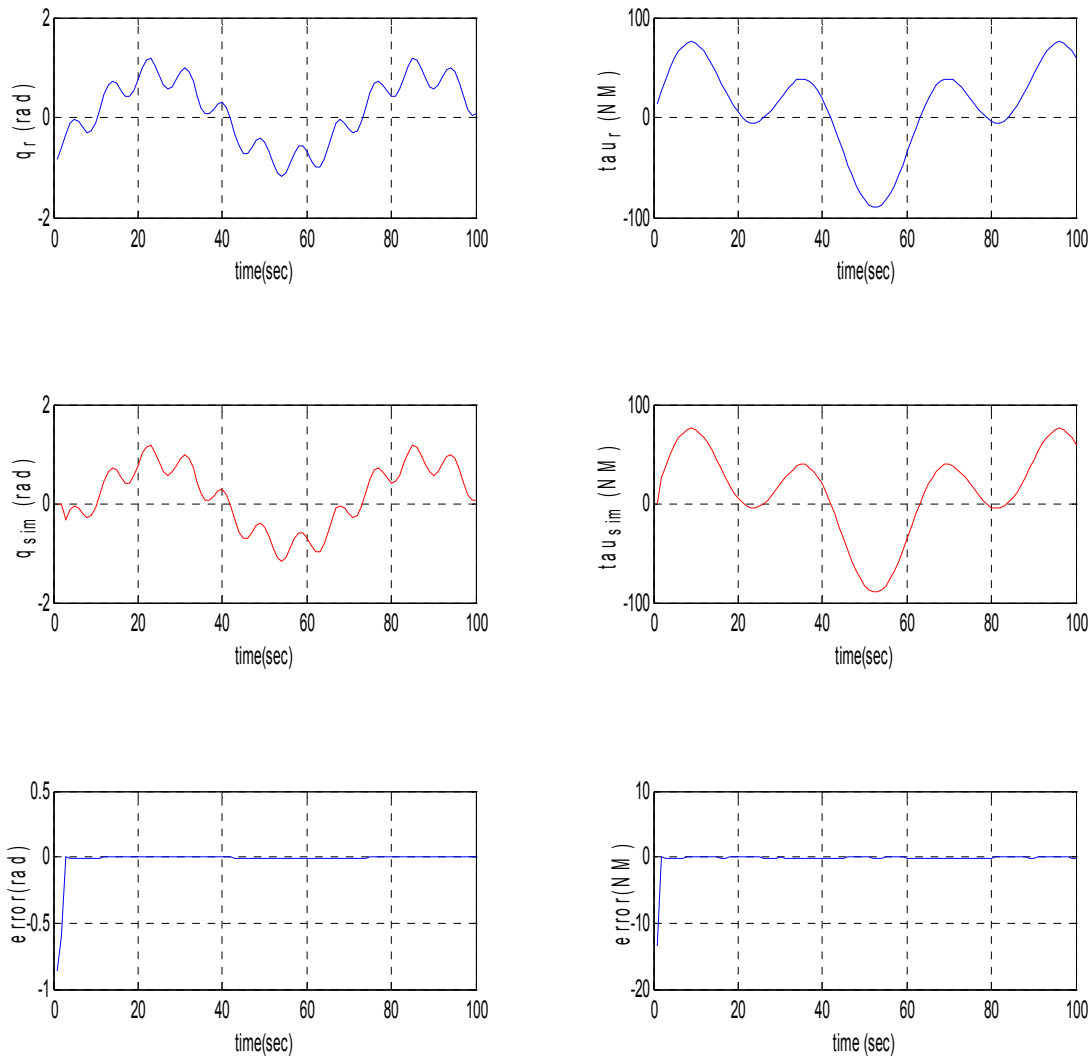

Fig. 5.13 Performance with "purelin,purelin" transfer function

In above performance, both error graphs show zero value. Its means it has very good tracking capability as well as identification capability. This pair of transfer function for neural controller gives the very good response for robotic manipulator.

e) The second layer transfer function of the neural network as the saturating linear transfer function “satlin” and the first layer is linear transfer function “purelin”. The following performance is obtained.

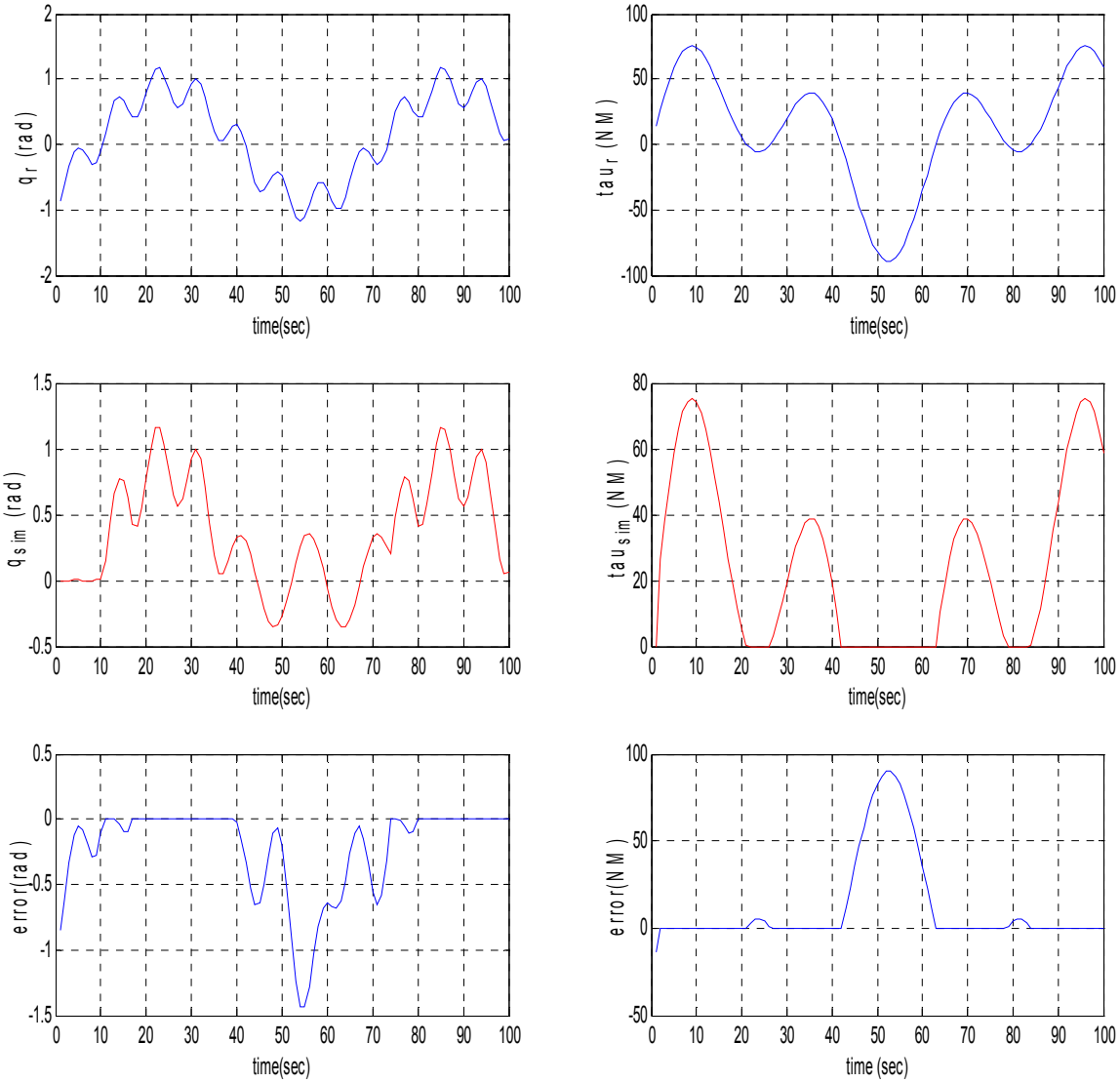

Fig. 5.14 Performance with “purelin,satlin” transfer function

In the above performance is shown that the tracking capability as well as identification capability of neural network is bad. This pair should not be used for the given robotic manipulator.

f) The second layer transfer function of the neural network as the log-sigmoid transfer function “logsig” and the first layer is linear transfer function “purelin”. The following performance is obtained.

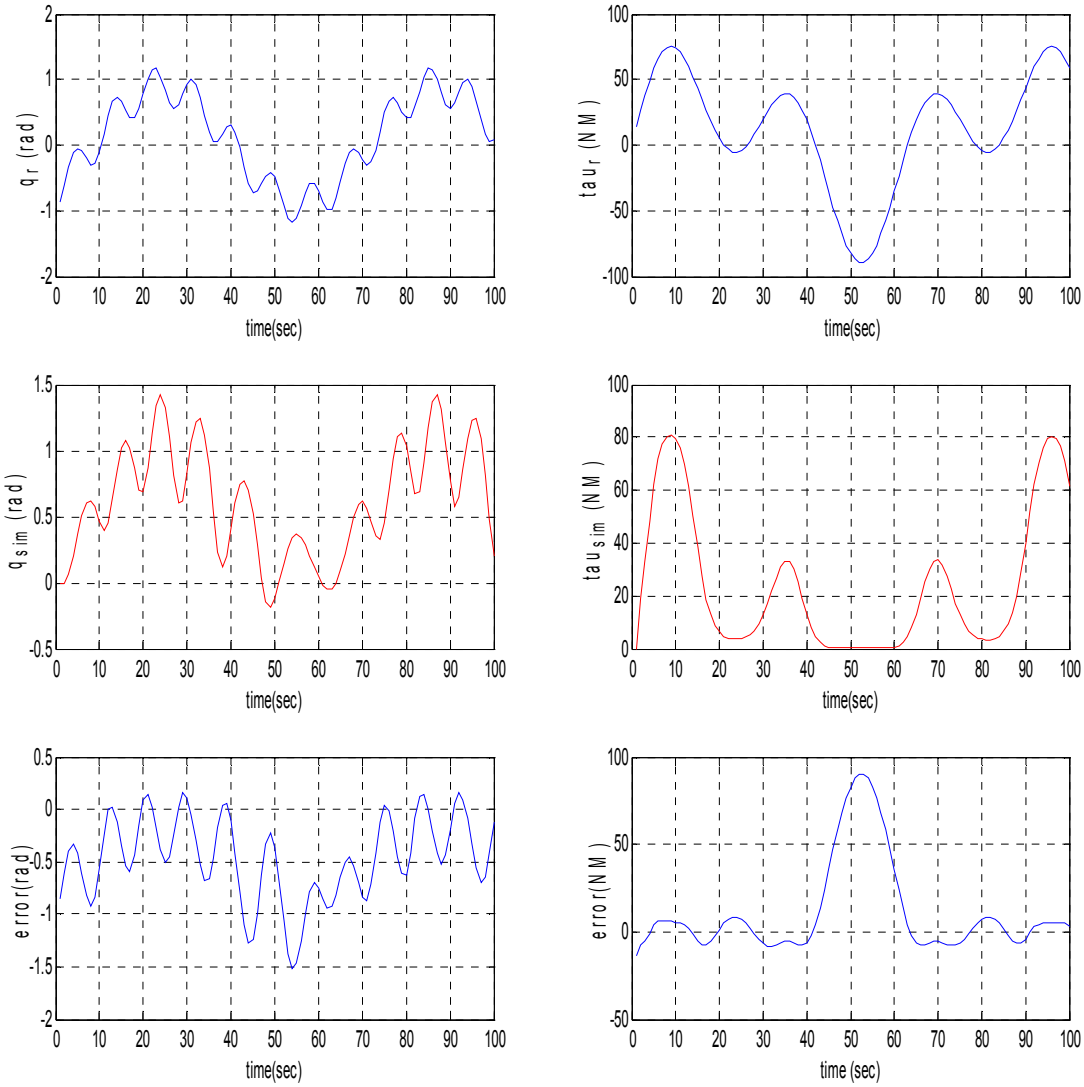

Fig. 5.15 Performance with “purelin,logsig” transfer function

This performance is also show that the unsatisfactory tracking and identification capability as previously mentioned pair of transfer function that why it should not be used for neural network.

g) The second layer transfer function of the neural network as the hyperbolic tangent sigmoid transfer function "tansig" and the first layer is linear transfer function "purelin". The following performance is obtained.

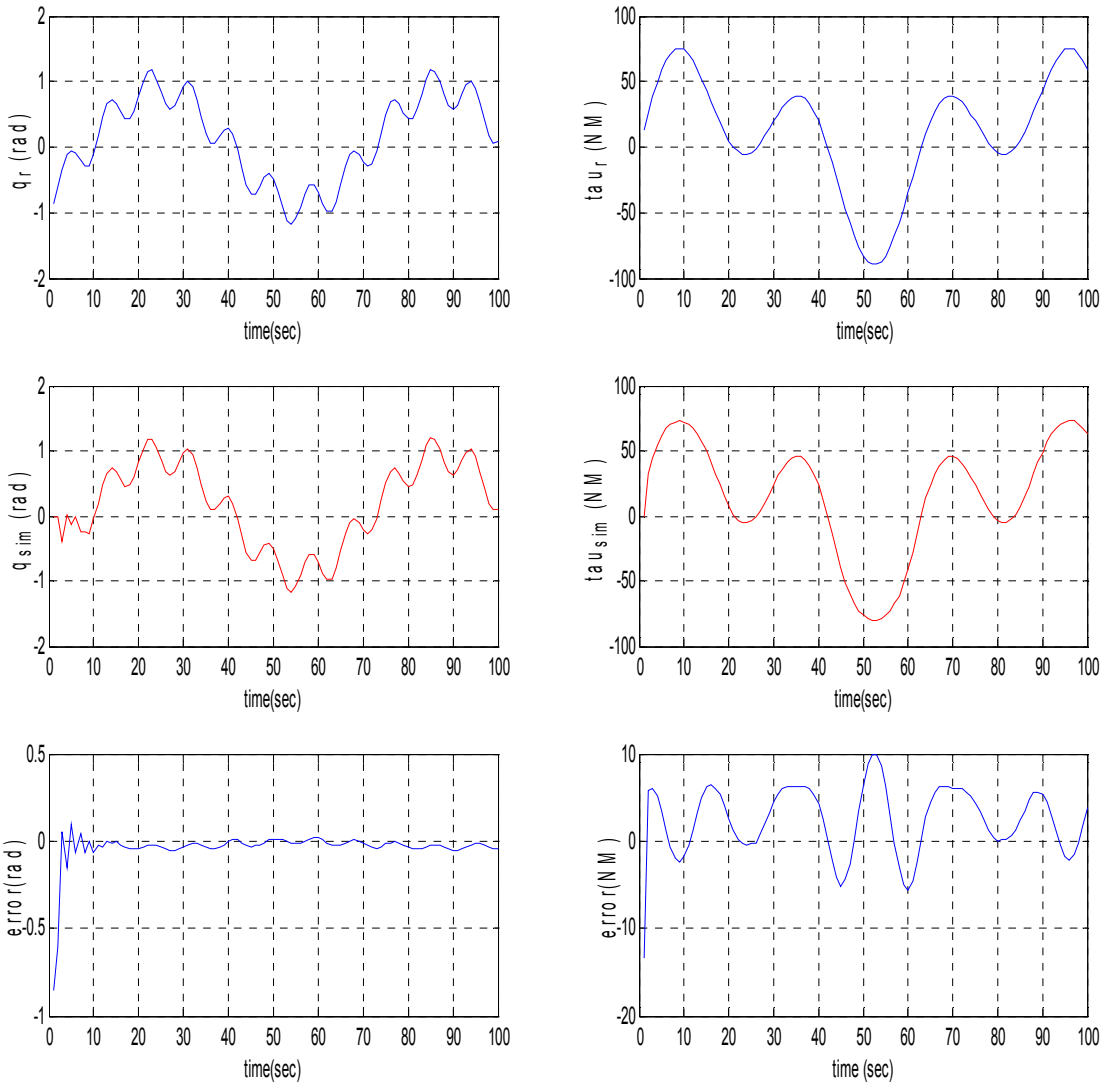

Fig. 5.16 Performance with "purelin,tansig" transfer function

The above performance, the error between reference trajectory between and simulated trajectory is almost zero, hence the tracking capability of neural network is good but the identification capability is not very satisfactory.

h) The second layer transfer function of the neural network as the positive linear transfer function “poslin” and the first layer is linear transfer function “purelin”. The following performance is obtained.

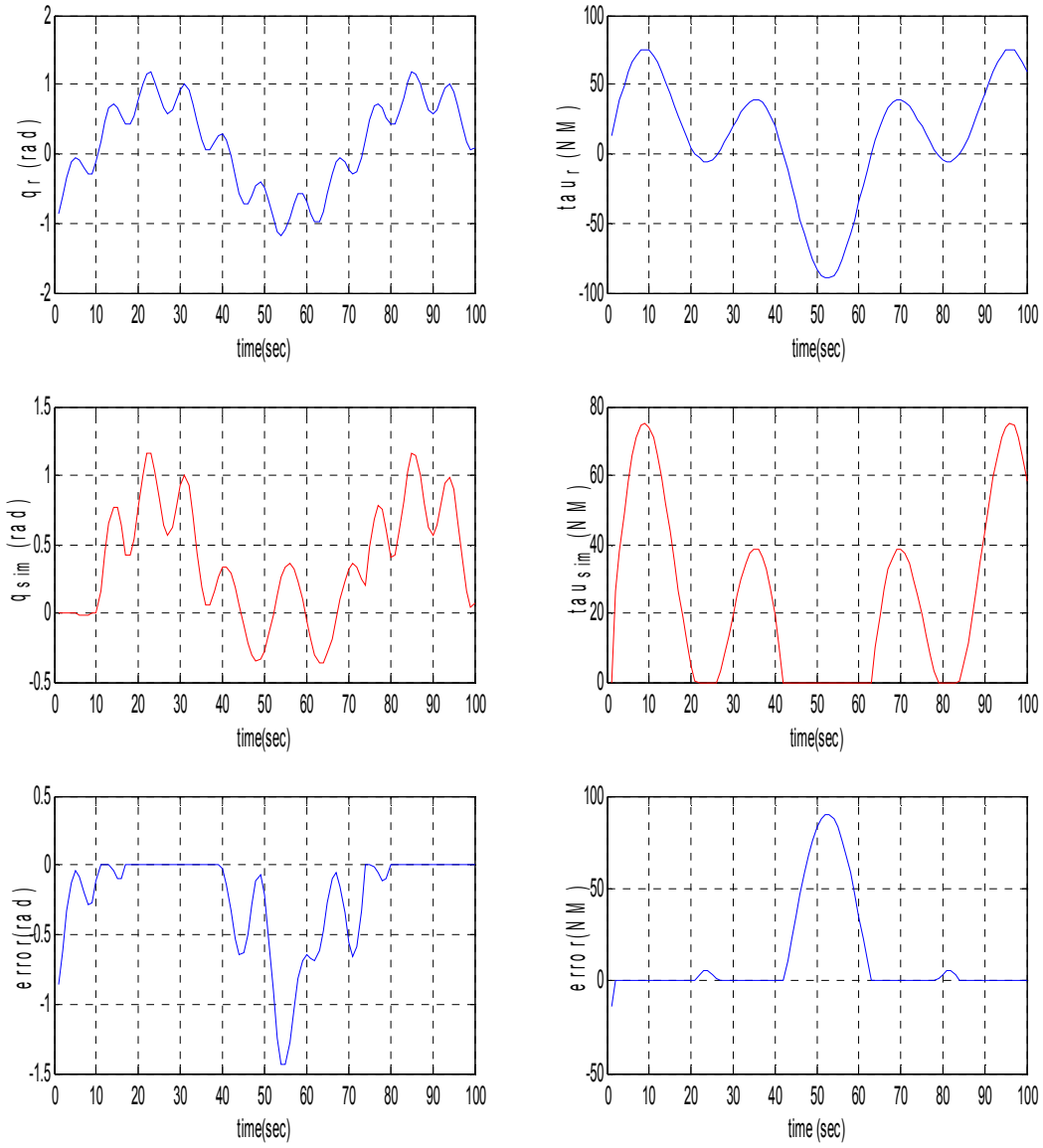

Fig. 5.17 Performance with “purelin,poslin” transfer function

For this combination of transfer function for neural controller show unsatisfactory response for the tracking as well as identification. Therefore it is not used for the combination for the one link manipulator.

i) The second layer transfer function of the neural network as the symmetrical saturating linear transfer function “satlins” and the first layer is linear transfer function “purelin”. The following performance is obtained.

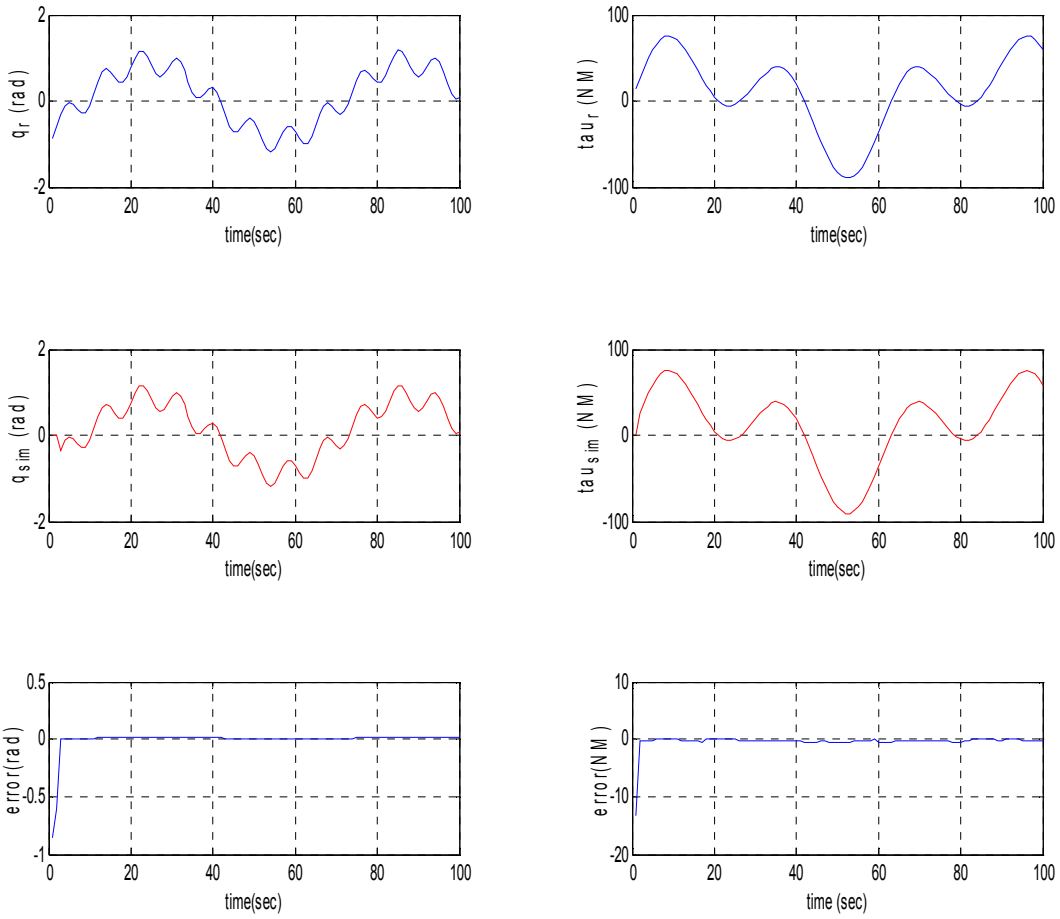

Fig. 5.18 Performance with “purelin,satlins” transfer function

These combinations of transfer function, neural controller show the unsatisfactory response for both tracking and identification for the one link manipulator.

5.3.1 SUMMARY

For the first layer with “purelin” transfer function and the change the second layer, it is observed that:

a) The transfer function “compet”, “hardlim”, “hardlims”, “satlin”, “logsig”, “poslin” has very poor tracking capability as well as bad neural network identification capability.

b) The transfer function “purelin” and “satlins” has almost good tracking capability as well as neural network identification. But for “tansig”, it has good tracking capability but bad neural network identification capability.

It may conclude that for second layer, we may be used “purelin”, “satlins”and “tansig”.

5.4 EFFECT OF THE VARIATION OF CONSTANT OF CONTINUOUS UNCERTAINTIES (c)

There are many parameters in the continuous uncertainties. Here, the continuous uncertainties are assumed as c.signq. Therefore, the value of c is also a factor that may change the response of the system. To see the effect of the variation of it, we vary the value of c step by step.

a) For the value of c = 0.0001. The following performance is obtained.

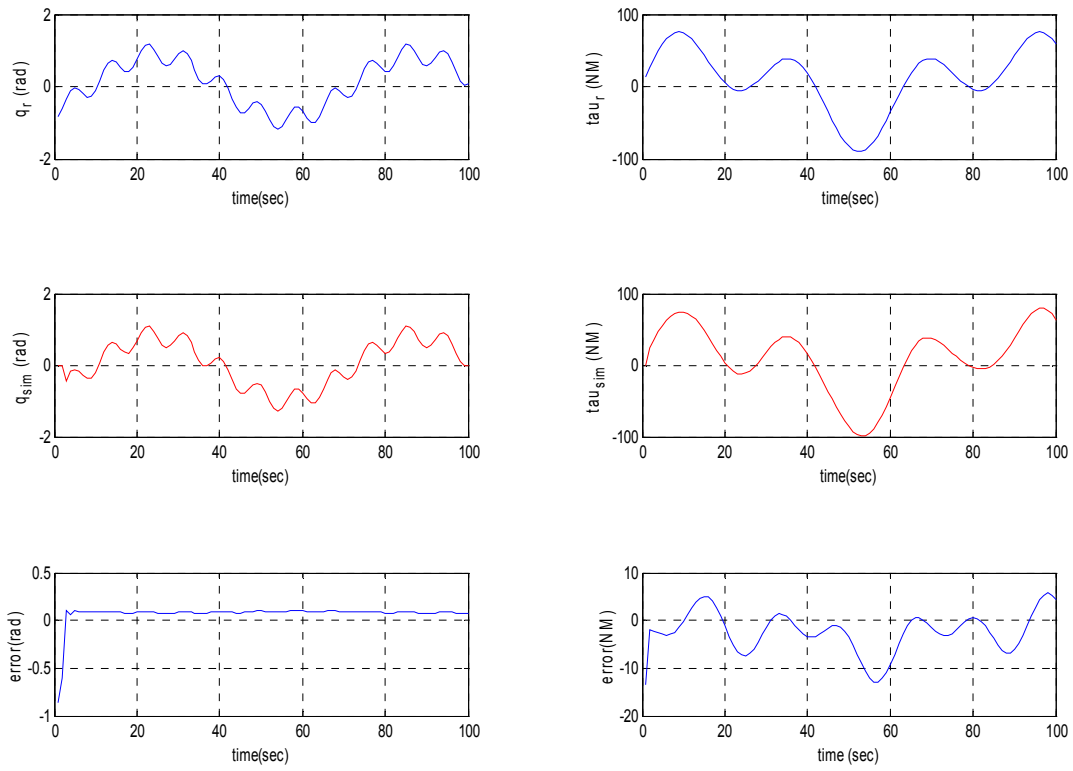

Fig. 5.19 Performance with c = 0.0001

At this value of c, the neural controller show the good tracking capability for the robotic manipulator but it gives unsatisfactory response for the identification of the robotic manipulator.

b) For the value of c = 0.001. The following performance is obtained.

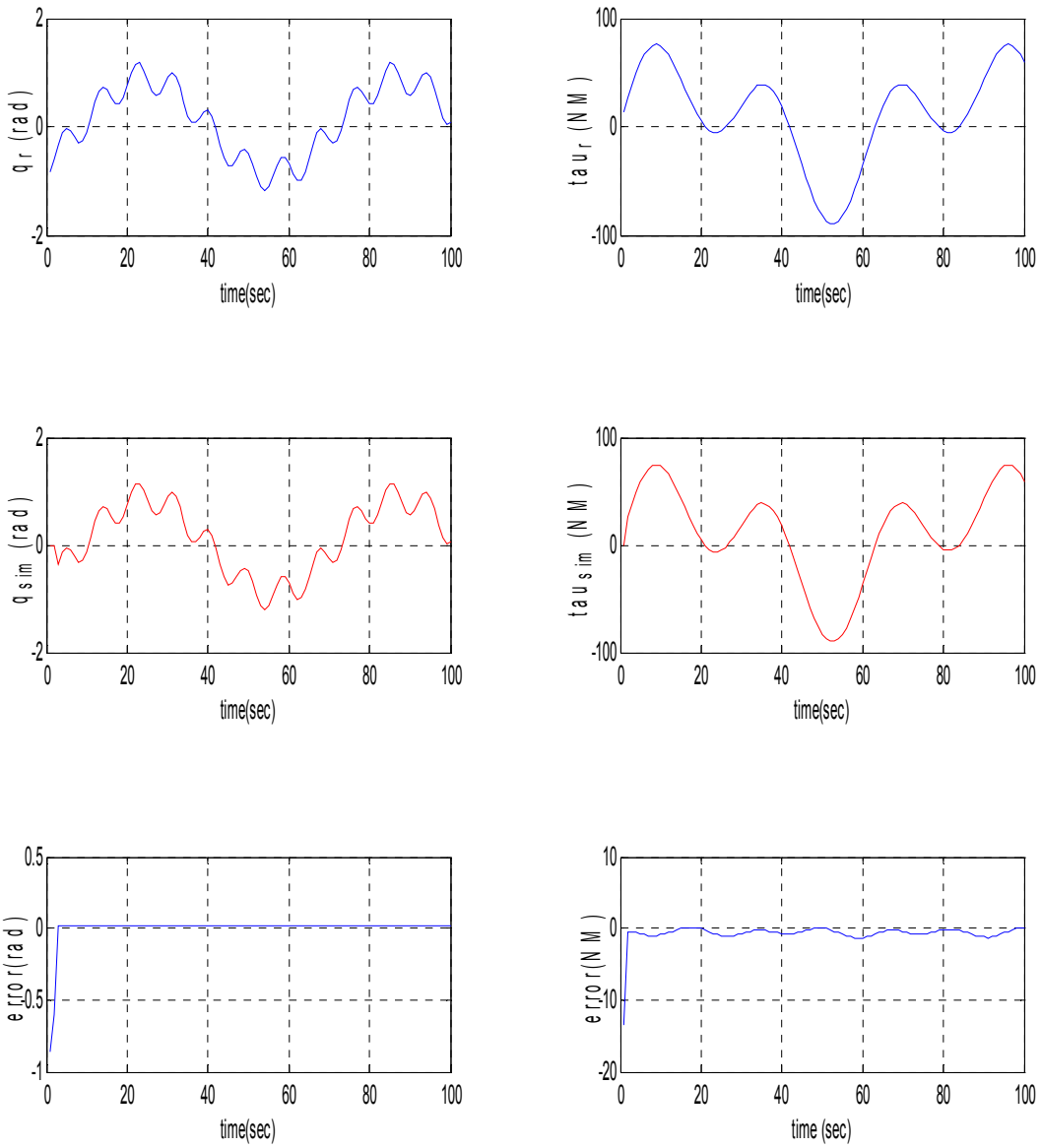

Fig. 5.20 Performance with c = 0.001

For the value of c = 0.001, the neural controller response is satisfactory for the robotic arm tracking, neural controller also show the almost satisfactory identification capability.

c) For the value of c = 0.01. The following performance is obtained.

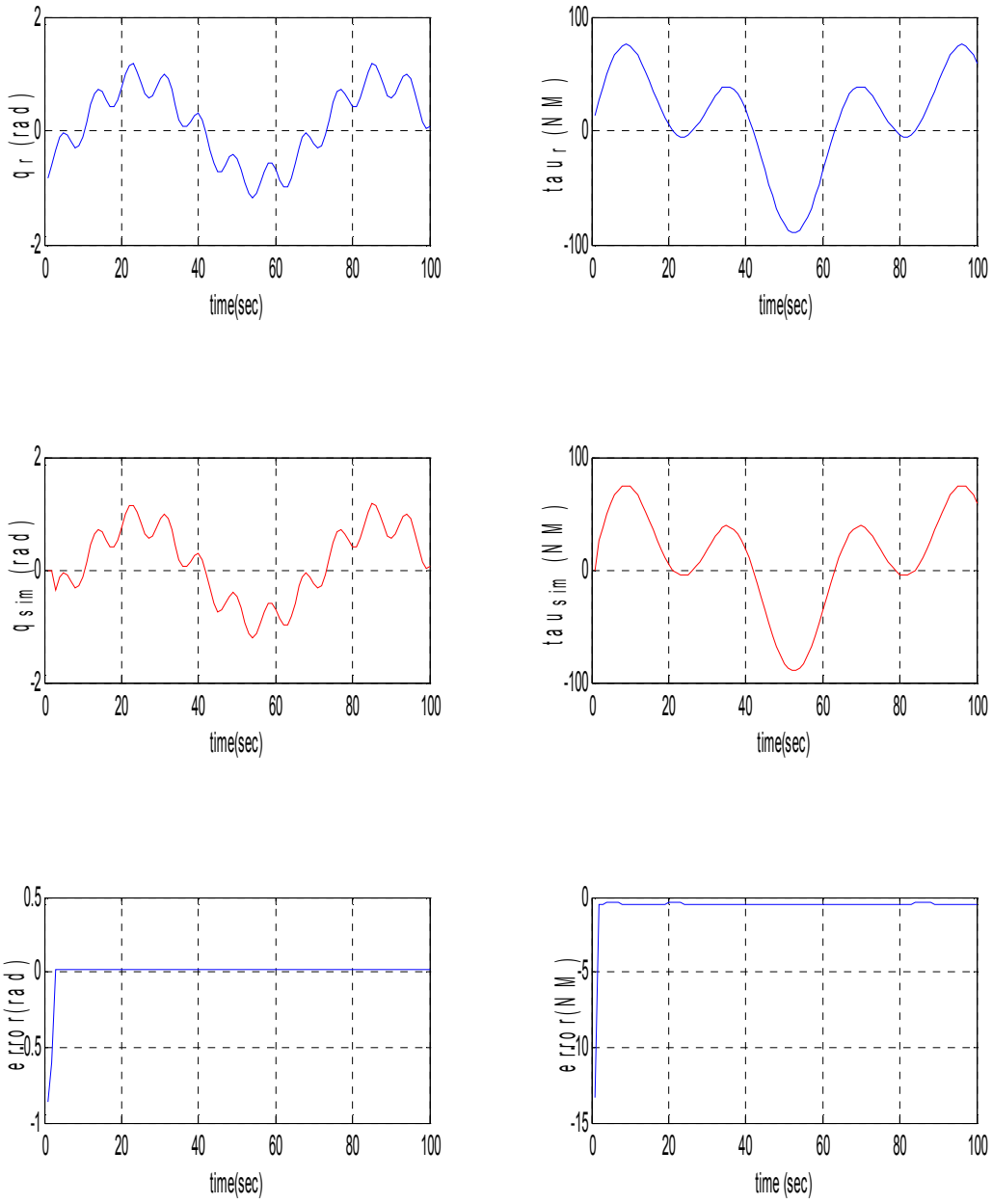

Fig. 5.21 Performance with c = 0.01

Neural controller at this value of c gives satisfactory response for both the tracking and identification of the robot arm.

d) For the value of c = 1. The following performance is obtained.

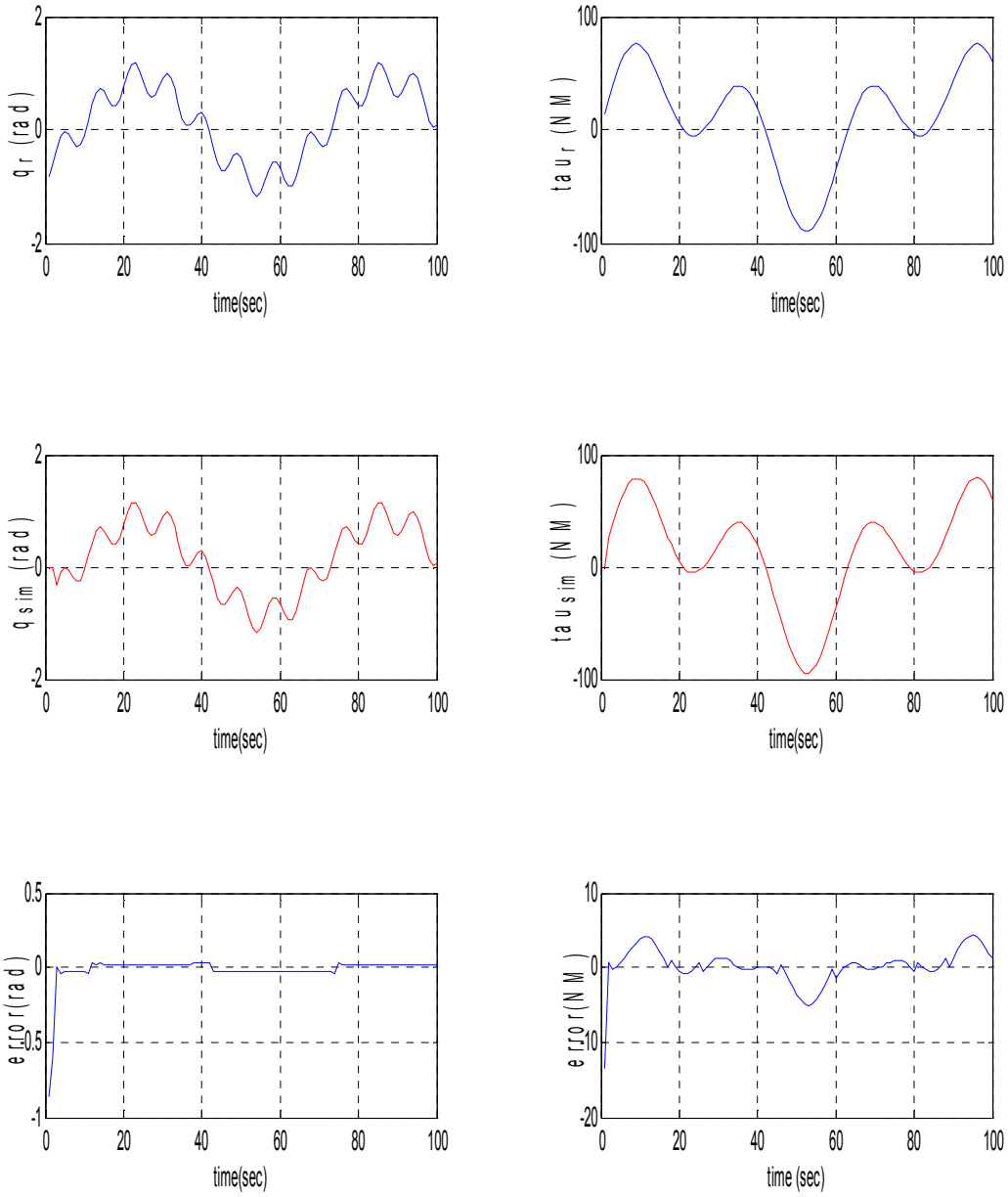

Fig. 5.22 Performance with c = 1

For the value of c = 1, neural controller control the robotic arm trajectory in well manner but identification of it show a little deviation.

e) For the value of c = 2. The following performance is obtained.

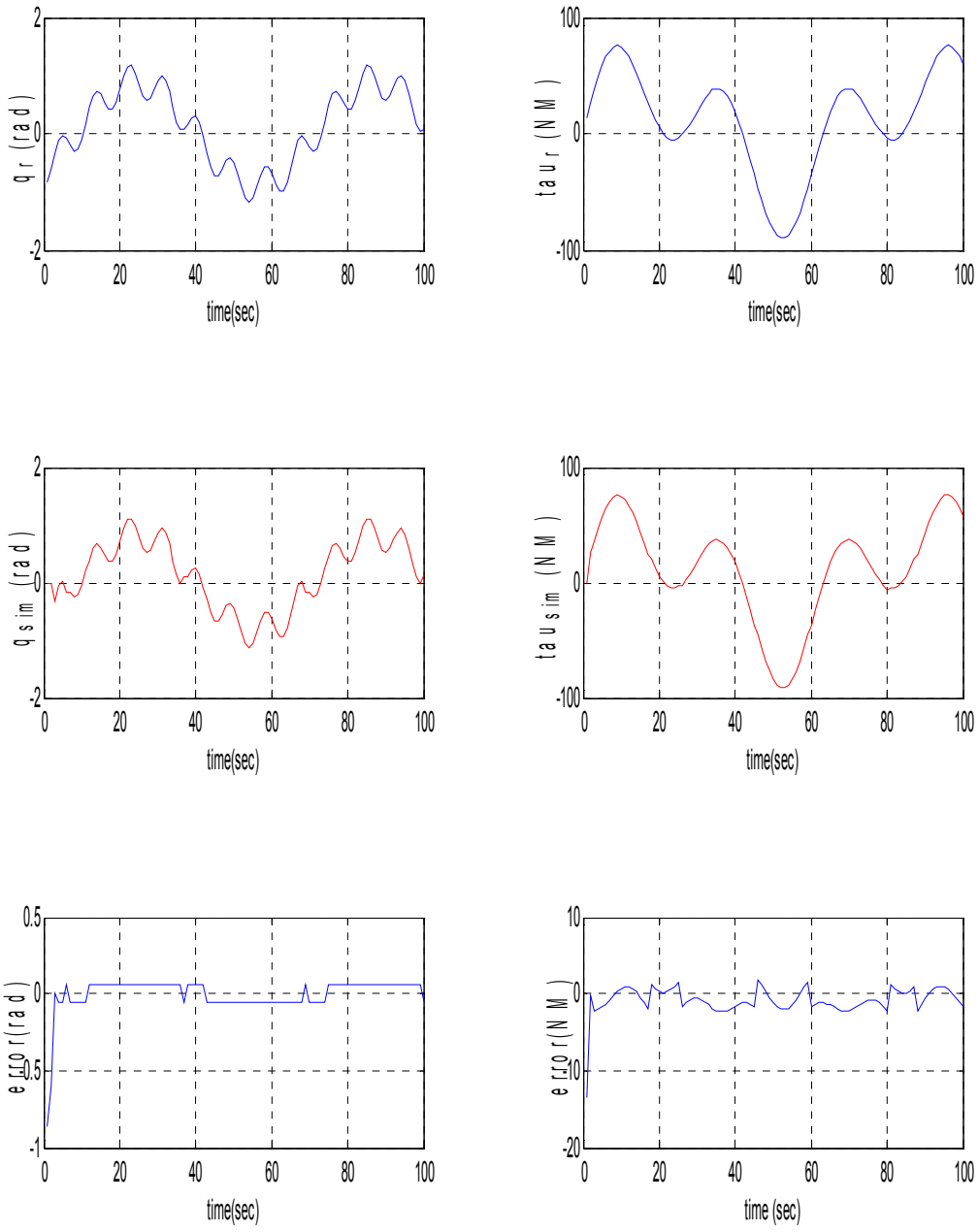

Fig. 5.23 Performance with c = 2

Neural controller at the value of c equal to two show the some variation in tracking of reference trajectory as well as also show the some variation to follow the reference torque.

f) For the value of c = 3. The following performance is obtained.

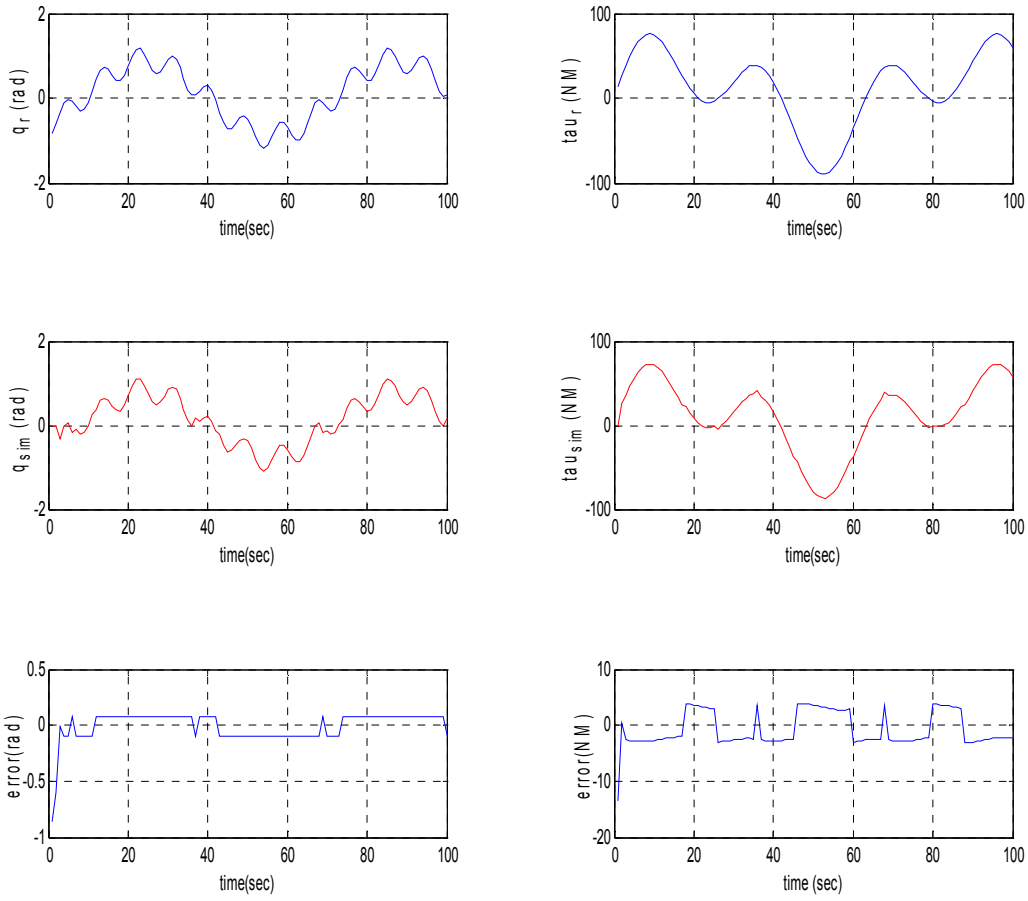

Fig. 5.24 Performance with c = 3

At the value of c = 3, the tracking capability as well as identification capability of neural controller is not satisfactory. It shows a lot variation.

5.4.1 SUMMARY

It is observed from the variation of continuous uncertainties c that from the value between 0.01 to 2 the neural controller show very good tracking capability as well as good capability of the identification of the robotic manipulator, but beyond this region neural controller show bad response for the both tracking as well as identification of the robotic manipulator. Therefore, for the proper operation of the neural controller, the value of c should be selected between 0.01 and 2.

5.5 EFFECT OF THE VARIATION OF ROBOT ARM LENGHT (l)

Length of the robotic manipulator is one of more important factor for designing point of view. For particular range of the length, the neural controller gives satisfactory response: beyond this range it will show variation in tracking as well as identification capability. For finding that range the length of robotic manipulator, length is changed in suitable step.

a) For the length of robotic manipulator, l = 0.1. The following performance is obtained.

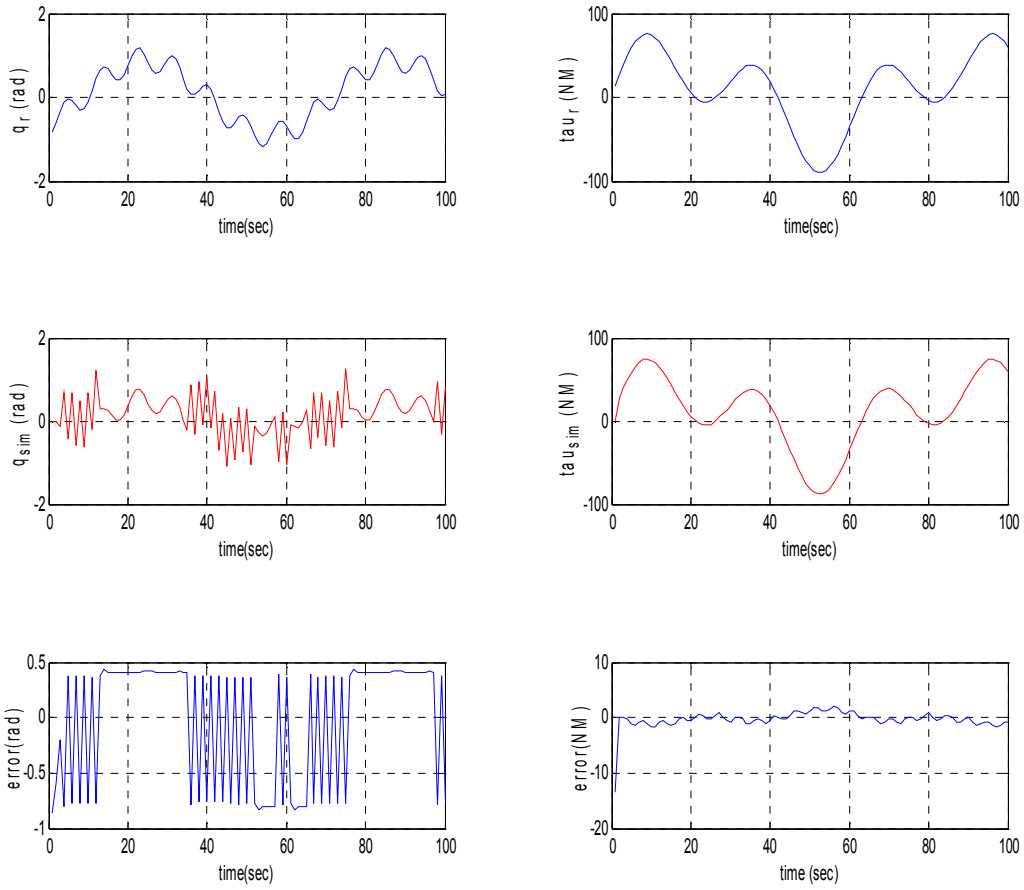

Fig. 5.25 Performance with l = 0.1

For the length of 0.1, simulated trajectory of the arm show a lot of variation from the reference trajectory of the robotic arm. Its means at this length, neural controller fail to control the robotic arm, but neural controller has satisfactory identification capability at this value.

b) For the length of robotic manipulator, l = 0.3. The following performance is obtained.

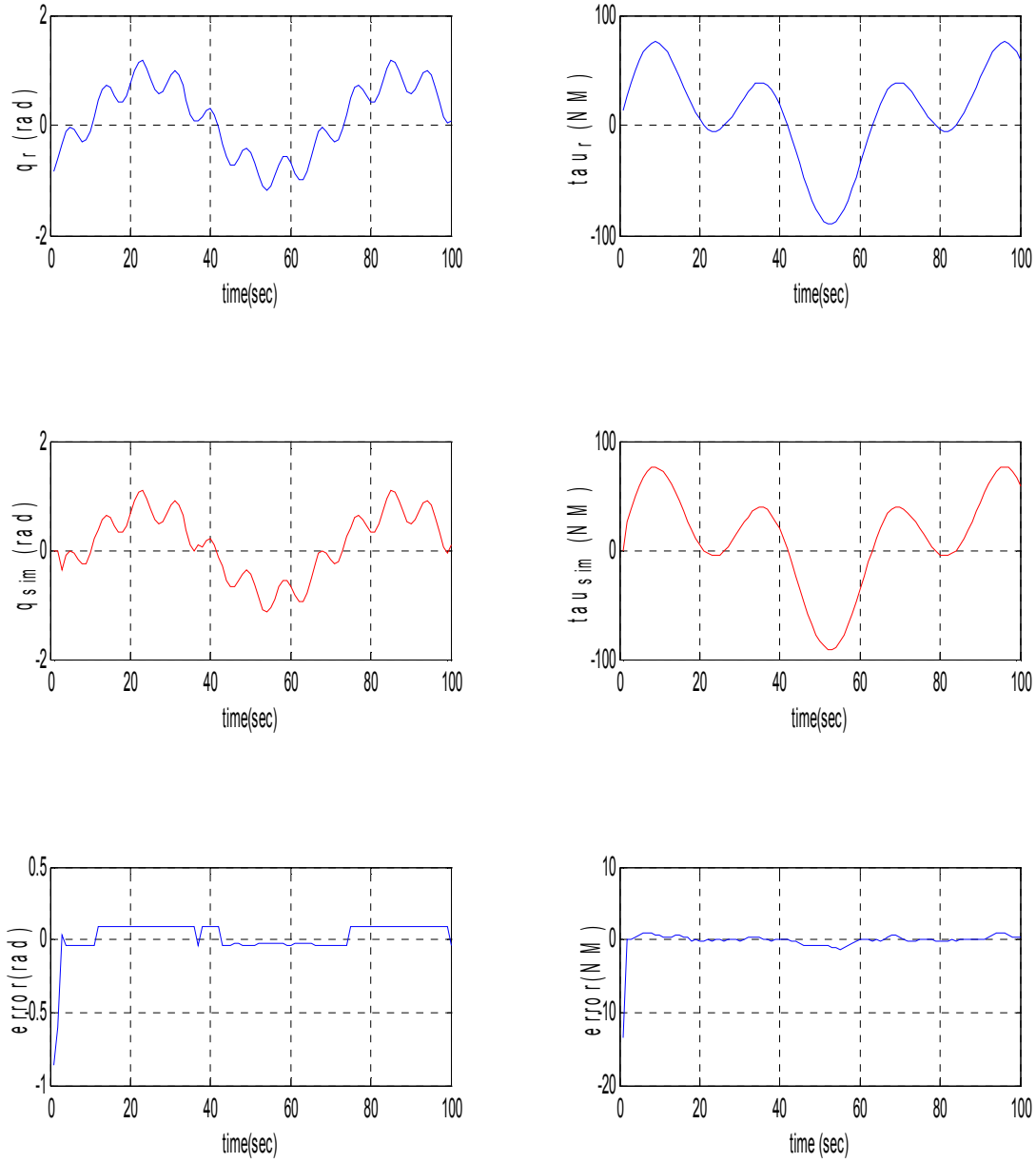

Fig. 5.26 Performance with l = 0.3

For this length of the robotic manipulator, neural controller work properly for both the tracking and identification of the robotic manipulator.

c) For the length of robotic manipulator, l = 2. The following performance is obtained.

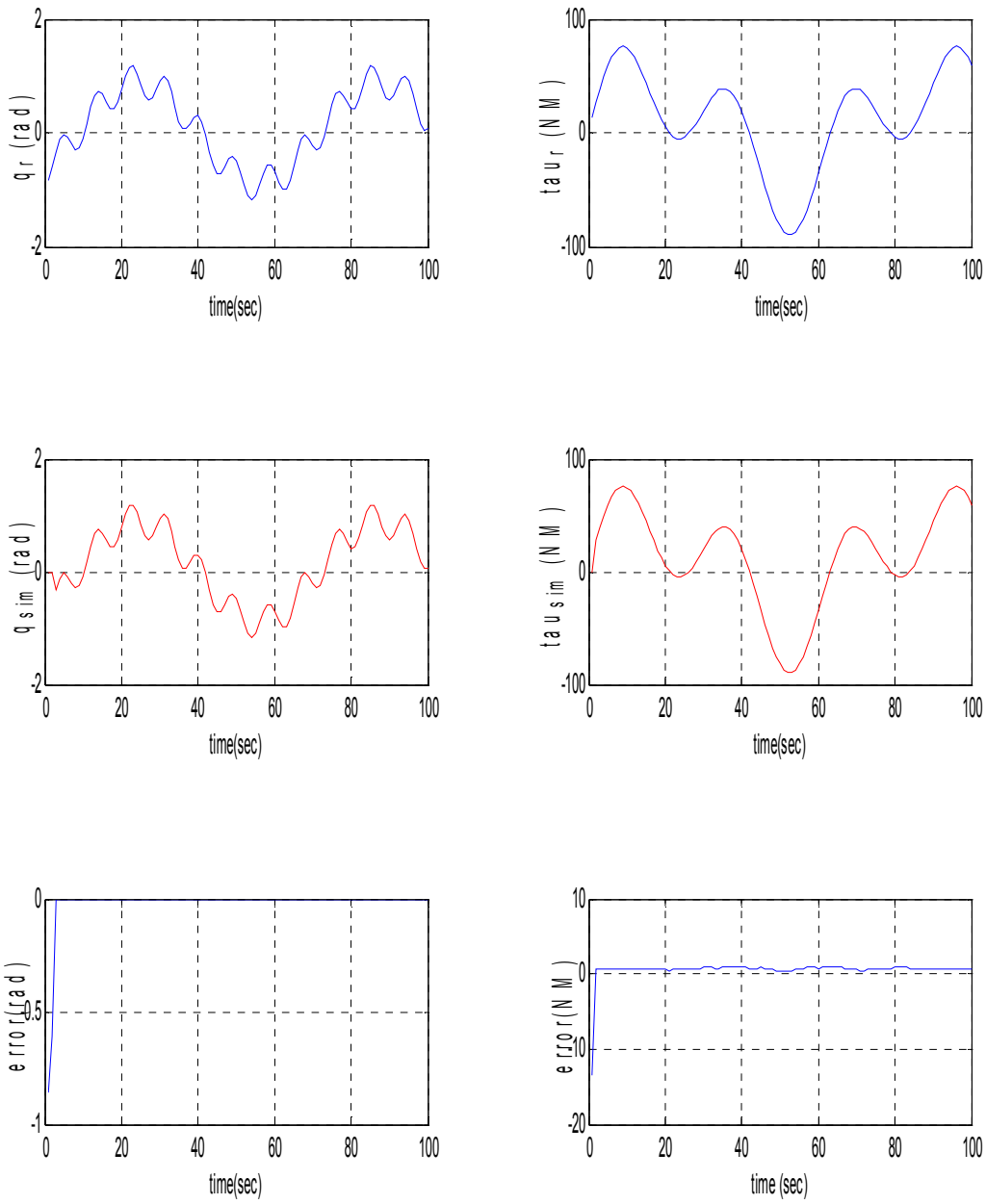

Fig. 5.27 Performance with l = 2

For this arm length of the robotic manipulator, neural controller also works properly for both the tracking and identification of the robotic manipulator.

d) For the length of robotic manipulator, l = 3. The following performance is obtained.

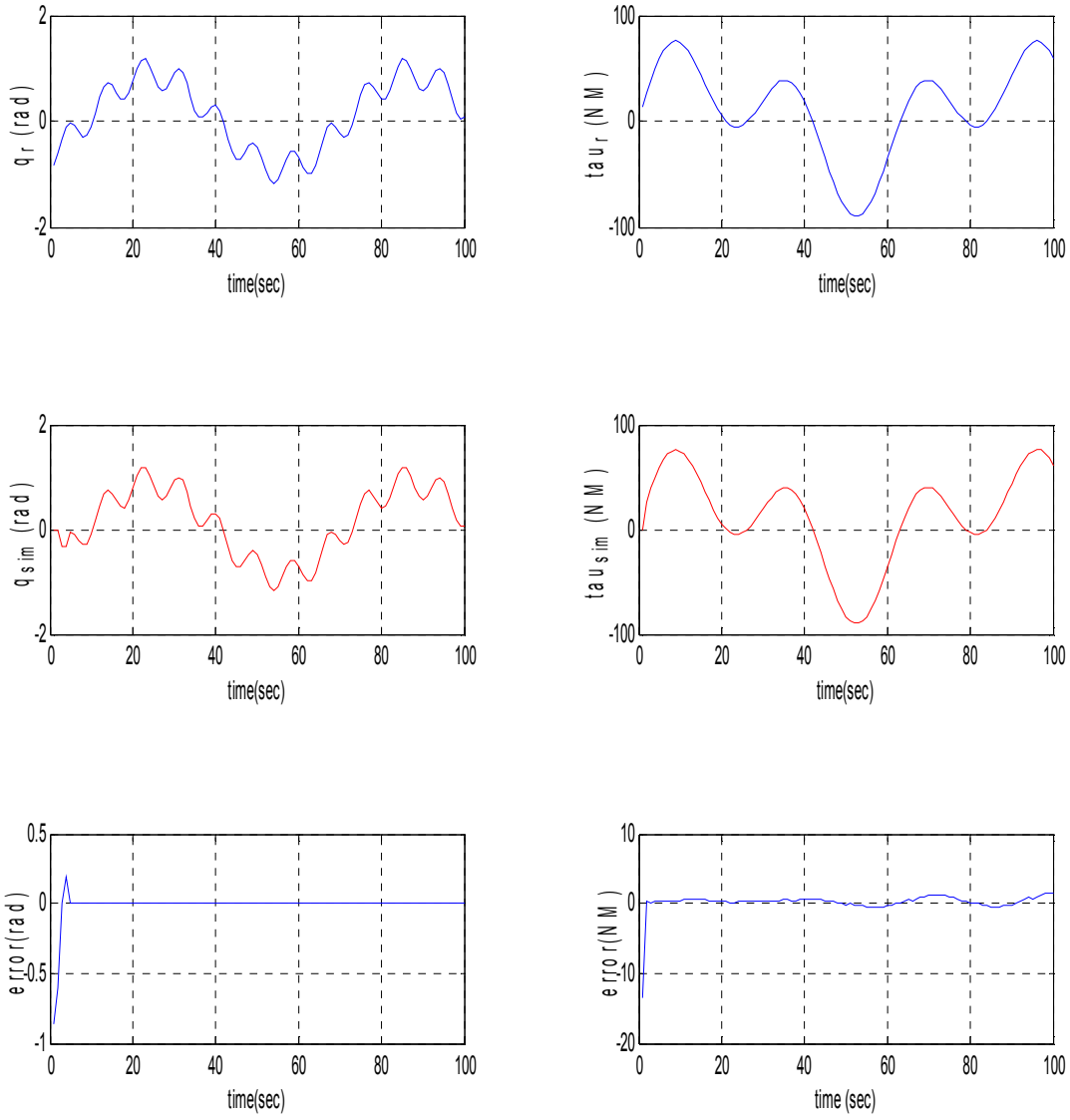

Fig. 5.28 Performance with l = 3

For this length of the robotic manipulator, neural controller is satisfactory for tracking of the robotic manipulator; it also has satisfactory identification capability.

e) For the length of robotic manipulator, l = 3.1. The following performance is obtained.

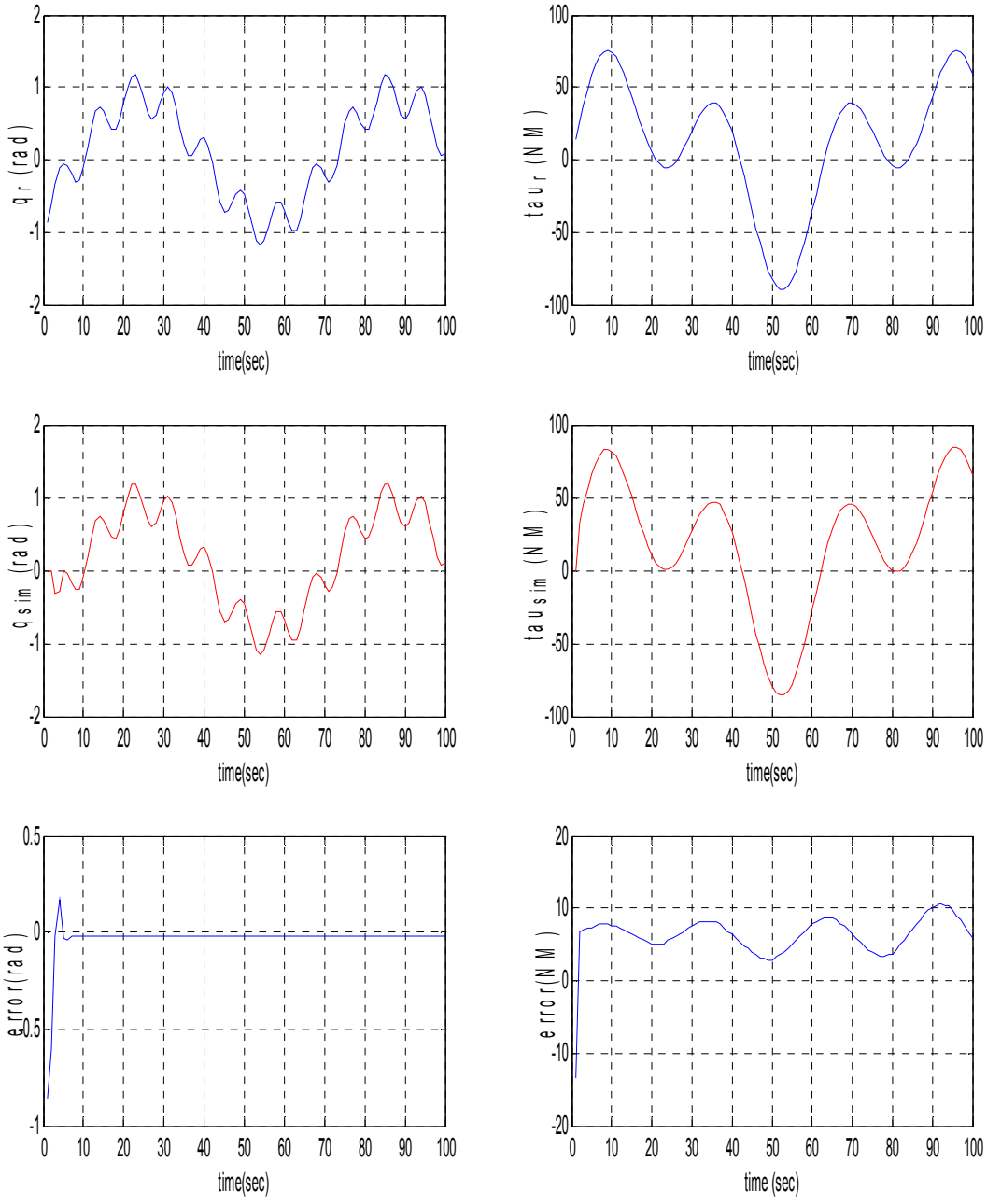

Fig. 5.29 Performance with l = 3.1

For this value the neural controller has satisfactory tracking capability but does not have satisfactory identification capability.

f) For the length of robotic manipulator, l = 3.5. The following performance is obtained.

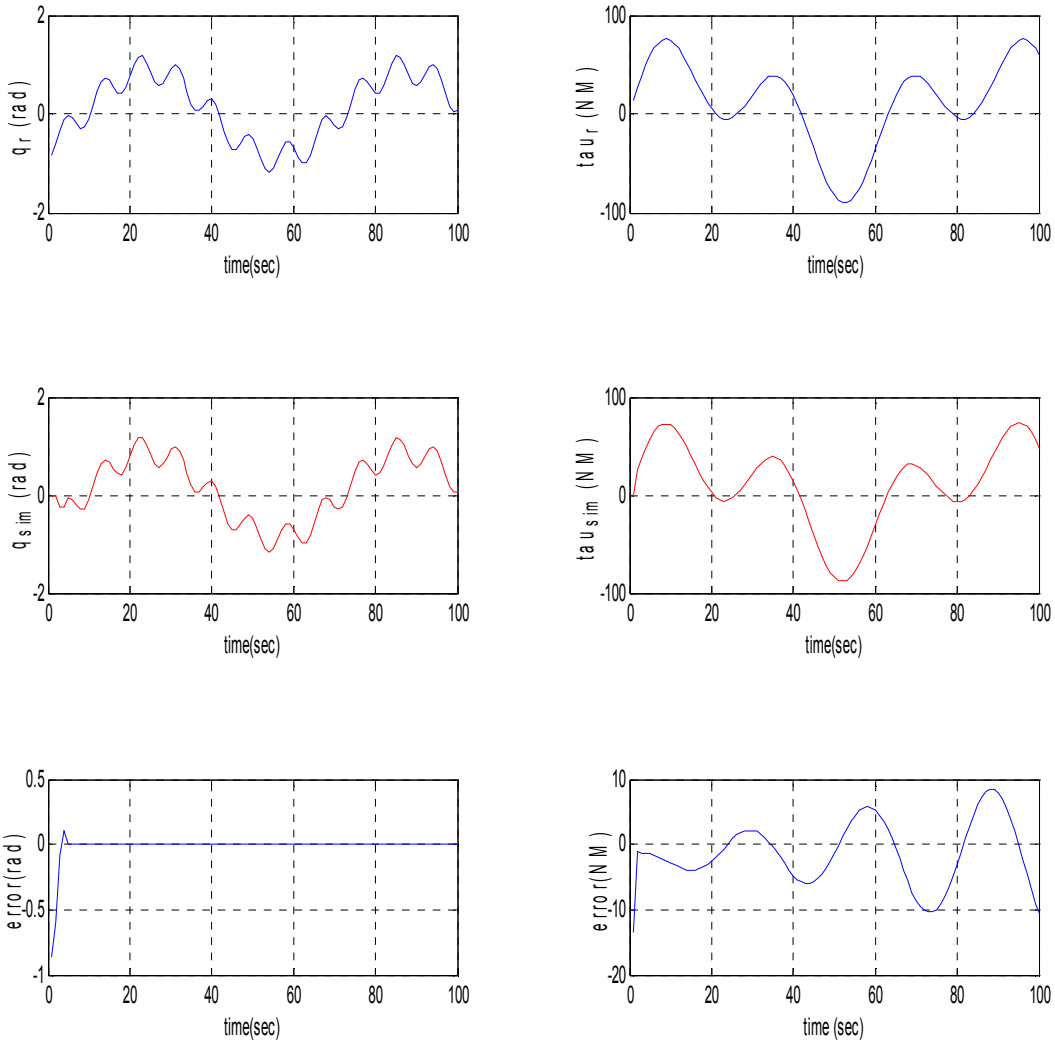

Fig. 5.30 Performance with l = 3.5

At this value the neural controller has good tracking capability but it has unsatisfactory identification capability of the robotic manipulator.

5.5.1 SUMMARY

By the variation of the length of robotic manipulator, it is observed that neural controller respond very well for tracking and identification of the robotic manipulator between 0.3 and 3. Beyond the length between o.3 and 3 the neural controller response in tracking capability is satisfactory but it loses its identification capability.

5.6 EFFECT OF THE VARIATION OF ROBOTIC MANIPULATOR MASS (m)

Mass of the robotic manipulator performance also plays an important role for this system. The mass is also affected the controller performance. For a certain range, the neural controller would do its job properly beyond this range it shows the bad result. For finding that range the mass of robotic manipulator, mass is changed in suitable step.

a) For the mass of robotic manipulator, m = 0.08. The following performance is obtained.

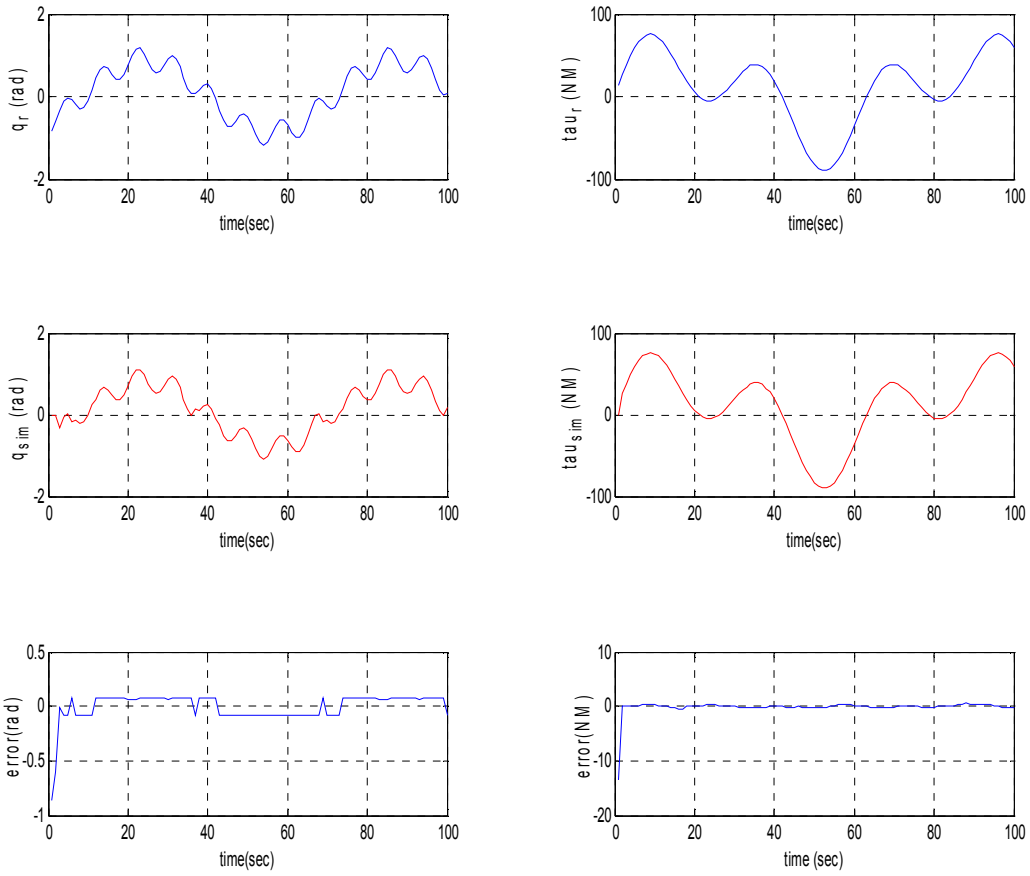

Fig. 5.31 Performance with m = 0.08

For this value of mass of robotic manipulator, the neural controller is working properly. Its means the tracking error as well as torque error is almost zero.

b) For the mass of robotic manipulator, m = 3. The following performance is obtained.

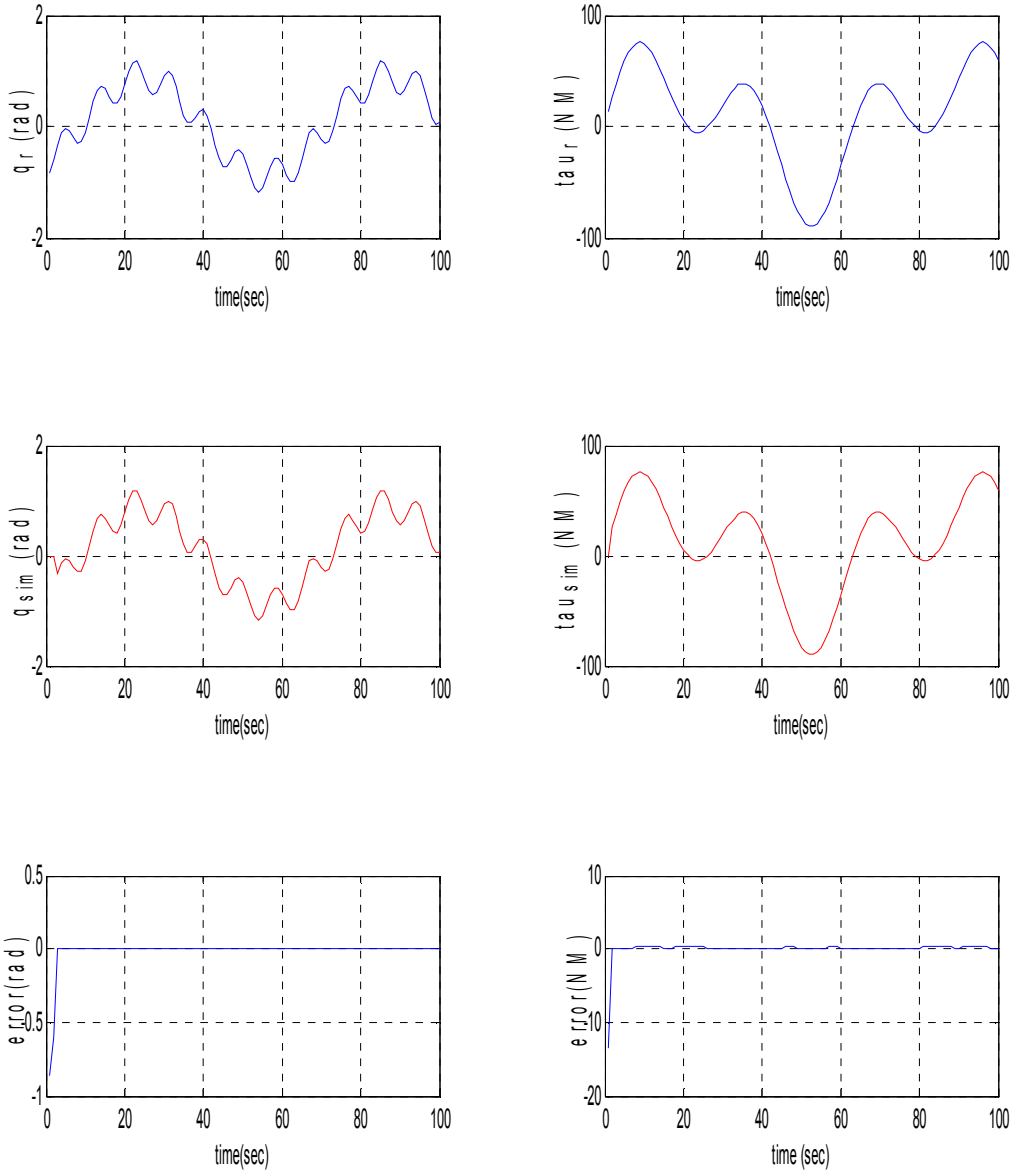

Fig. 5.32 Performance with m = 3

For this value of robotic manipulator, neural controller responds very well for both tracking as well as identification of the robotic manipulator.

c) For the mass of robotic manipulator, m = 4. The following performance is obtained.

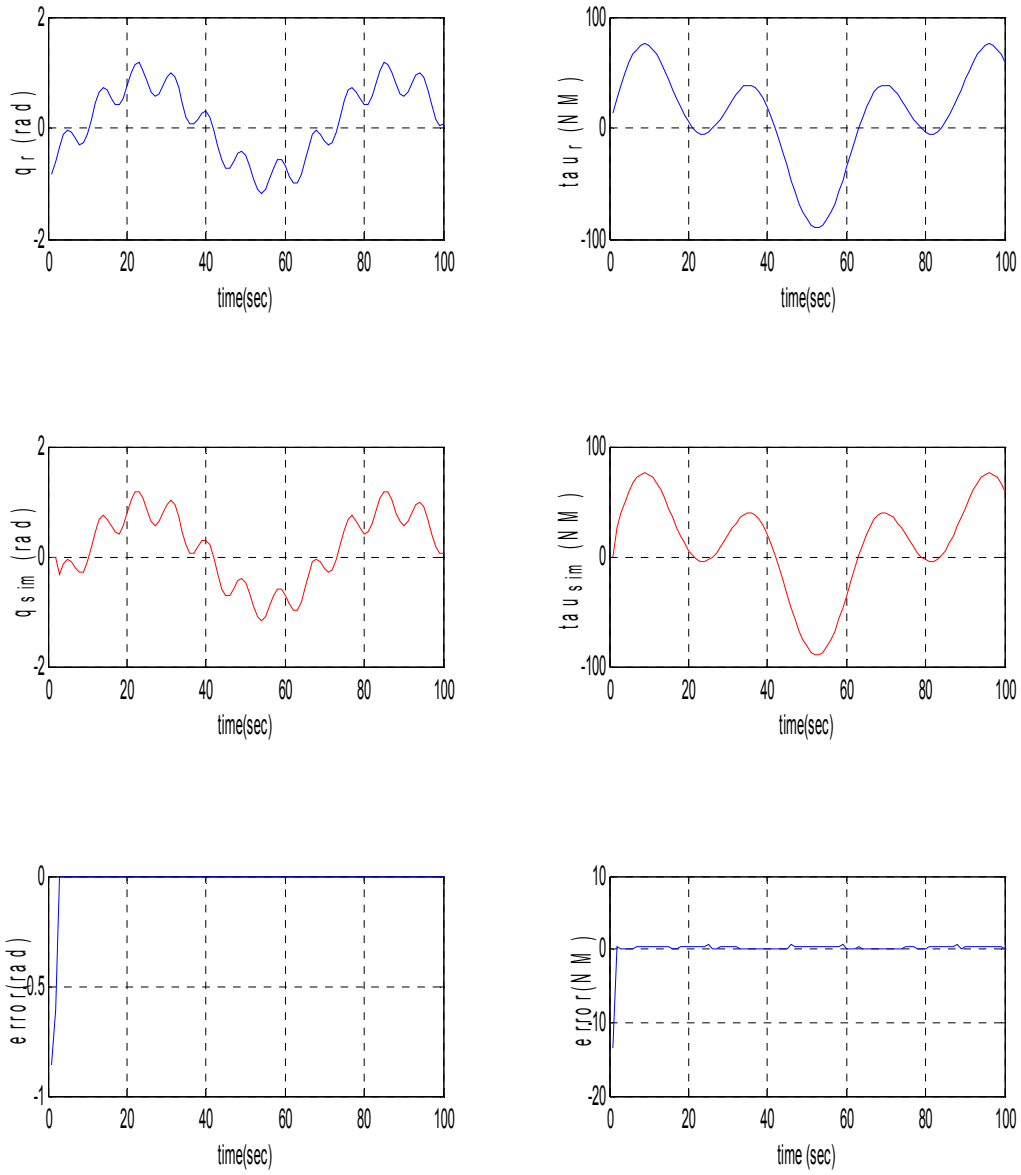

Fig. 5.33 Performance with m = 4

Neural controller is worked properly both tracking as well as identification of the robotic manipulator at mass of robotic manipulator is equal to 4.

d) For the mass of robotic manipulator, m = 4.5. The following performance is obtained.

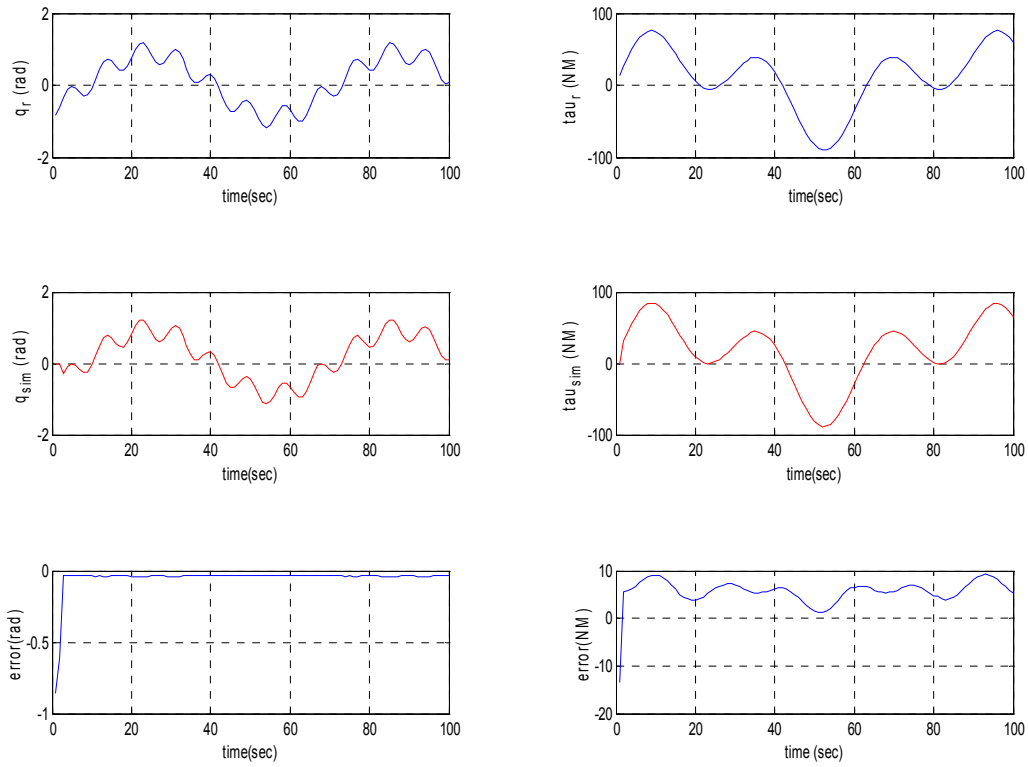

Fig. 5.34 Performance with m = 4.5

For this value, the neural controller has good tracking capability but identification capability is unsatisfactory for the robotic manipulator.

5.6.1 SUMMARY

For variation of the mass of the robotic manipulator, it is observed that the neural controller controls the robotic manipulator to its desired trajectory as well as has good identification capability for robotic manipulator at the value of mass of the robotic manipulator between 0.08 and 4.

5.7 TWO-LINK MANIPULATOR SIMULATION

For simulation of neural controller of two-link manipulator, the continuous and discontinuous uncertainties are assumed to be neglected. MATLAB coding is done on the basis of the inverse dynamics of the robotic manipulator. There is only presented the learning rate variation of training function of neural controller and observed its effect on the tracking capability of the neural controller.

The following parameters are considered for simulation study.

Mass of the first arm of the robotic manipulator = 4 kg

Mass of the second arm of the robotic manipulator = 2 kg

Length of the first arm of the robotic manipulator = 1 metre

Length of the second arm of the robotic manipulator = 0.5 metre

Value of the continuous uncertainties, Fc = 0

Value of the discontinuous uncertainties, Fd = 0

Here the transfer function and training method of neural controller is self defined function. Standard transfer function and training method is not used here.

5.8 EFFECT OF THE VARIATION OF LEAENING RATE

Learning rate is one important factor for neural network due to the reason of the stability of the neural network. Since, one major problem in neural network system is the stability. It may be overcome by the properly choosing of learning rate of given learning algorithm. To see the effect of learning rate in the neural network, we vary the learning rate step by step.

a) For the value of learning rate = 0.00000000001. The following performance is obtained.

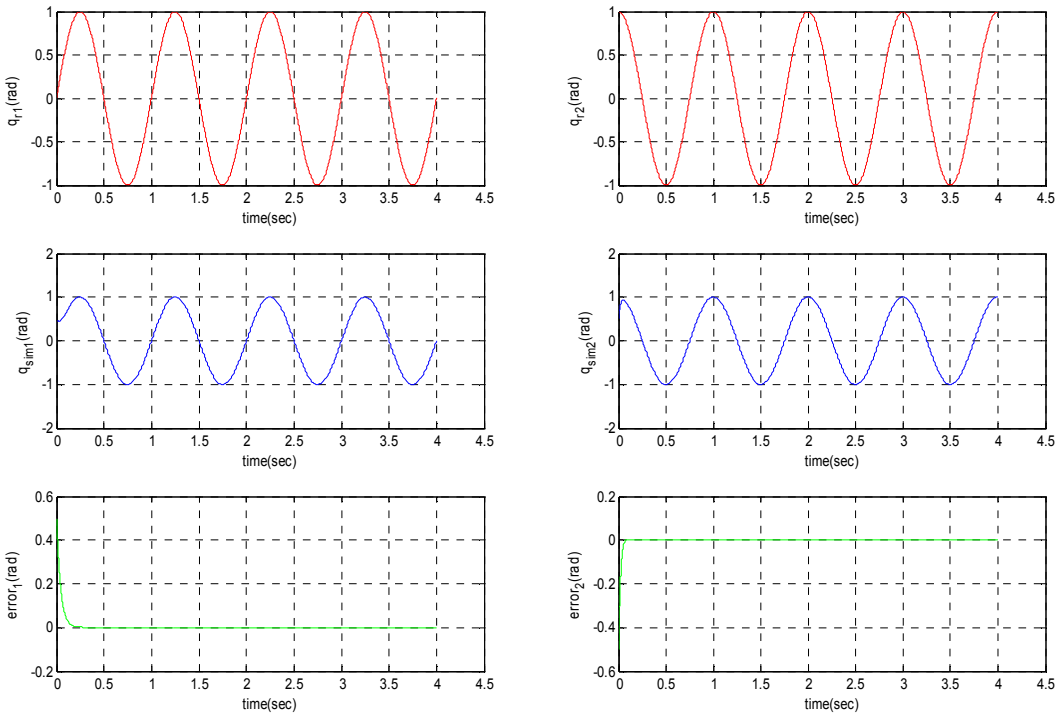

Fig. 5.35 Performance with learning rate = 0.00000000001

In the above performance, the error between desired trajectory and simulated trajectory for both first arm and second arm is almost zero. Therefore, neural controller for this value of learning rate performed satisfactory tracking capability.

b) For the value of learning rate = 0.001. The following performance is obtained.

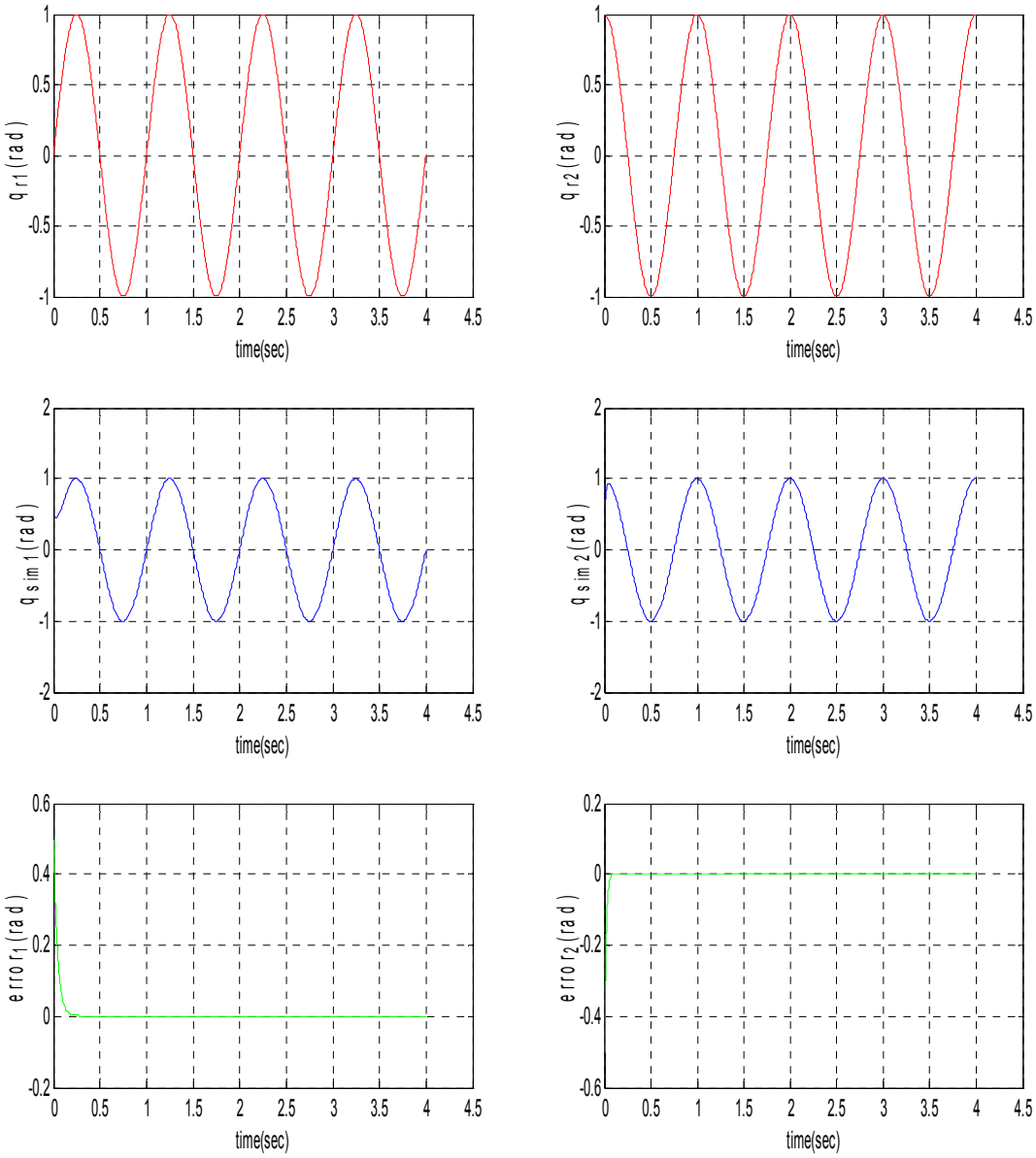

Fig. 5.36 Performance with learning rate = 0.001

For this value the first arm and second arm trajectory follow their desired trajectory, its means the neural controller control the robotic manipulator very well.

c) For the value of learning rate = 1. The following performance is obtained.

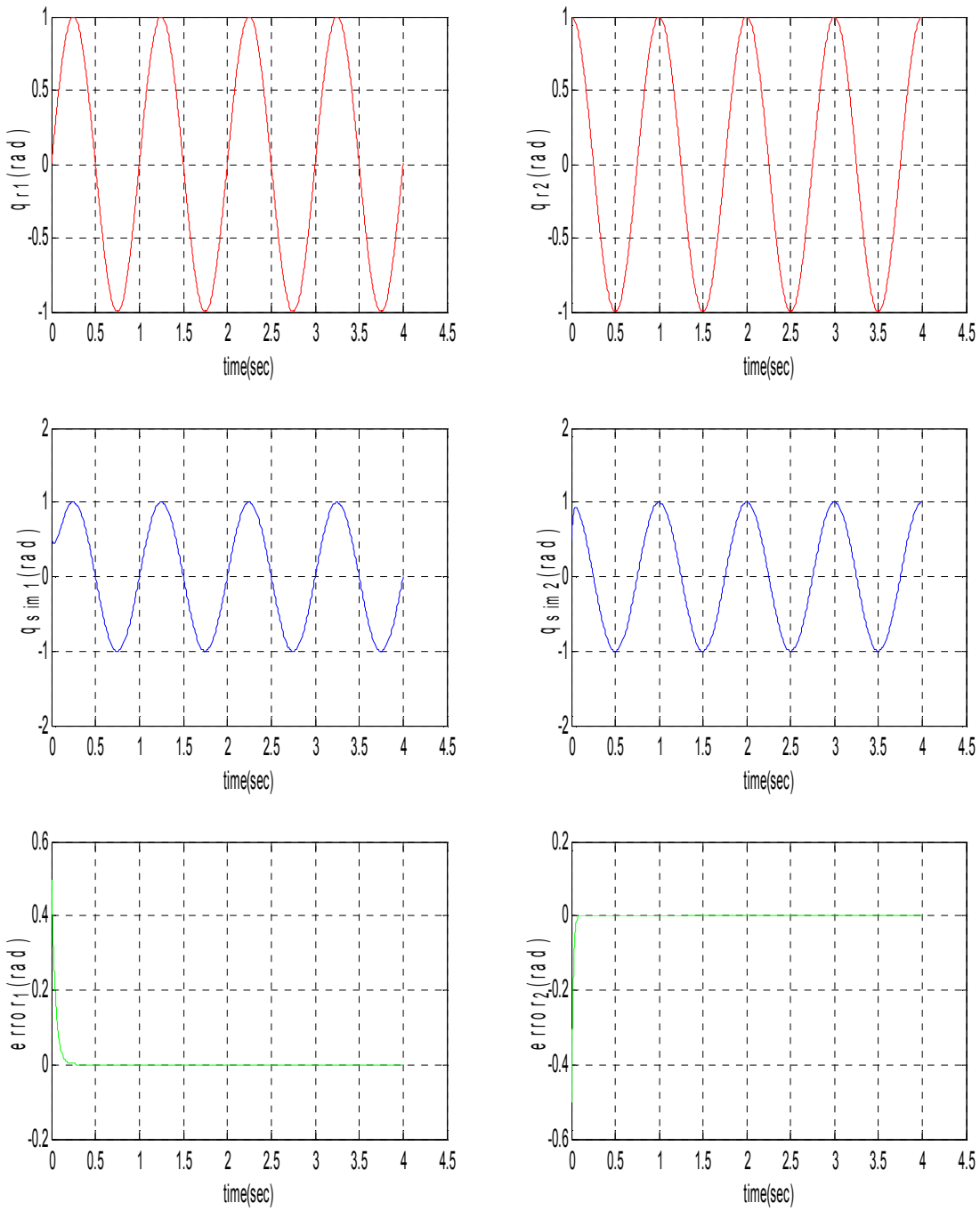

Fig. 5.37 Performance with learning rate = 1

At this value of the learning rate at 1, the neural controller control the robotic first and second arm properly as we seen from the error curve of first arm and second arm.

d) For the value of learning rate = 2. The following performance is obtained.

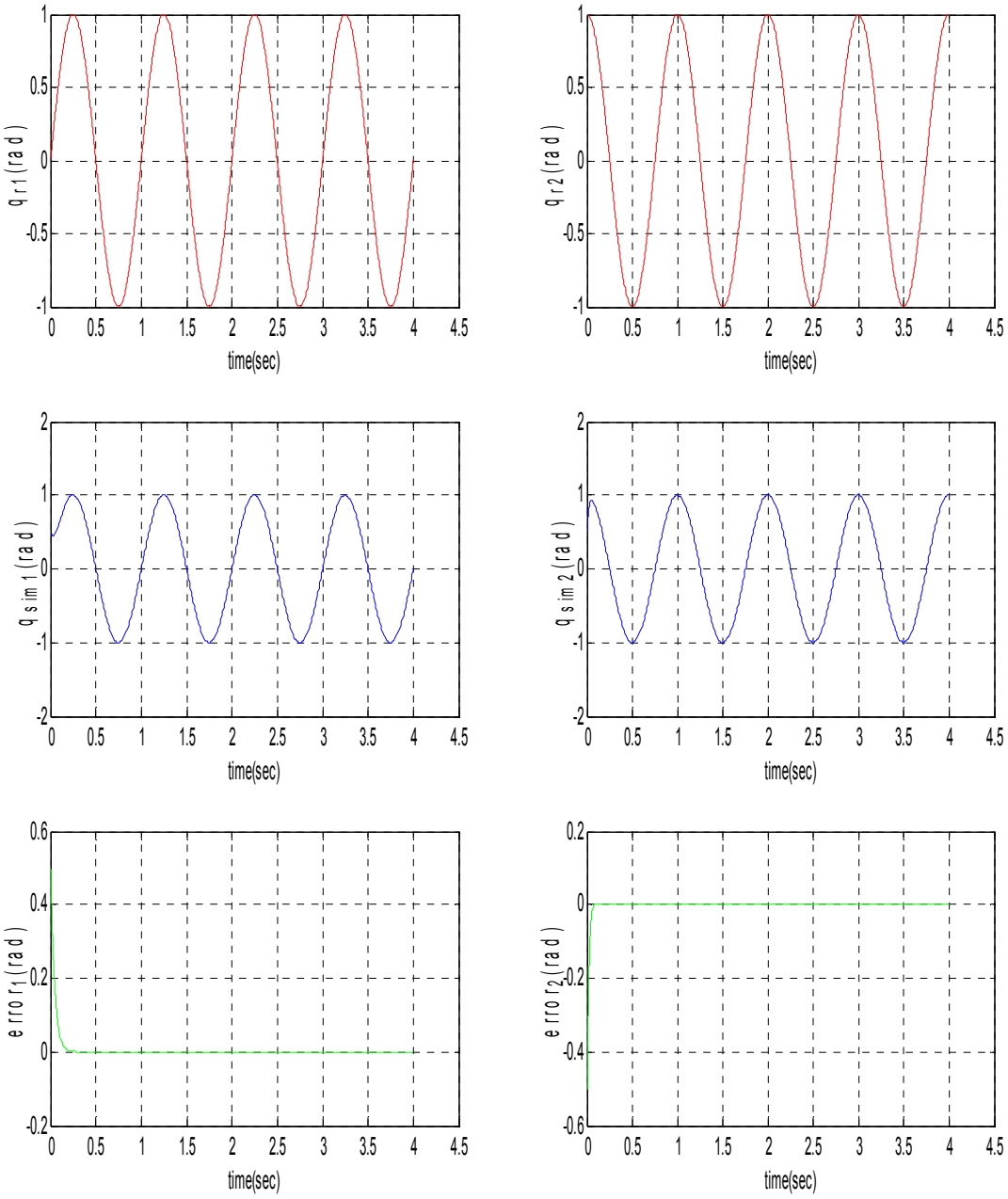

Fig. 5.38 Performance with learning rate = 2

For this value of learning rate, neural controller show satisfactory tracking capability for first arm as well as second arm of the two-link manipulator.

e) For the value of learning rate = 2.1. The following performance is obtained.

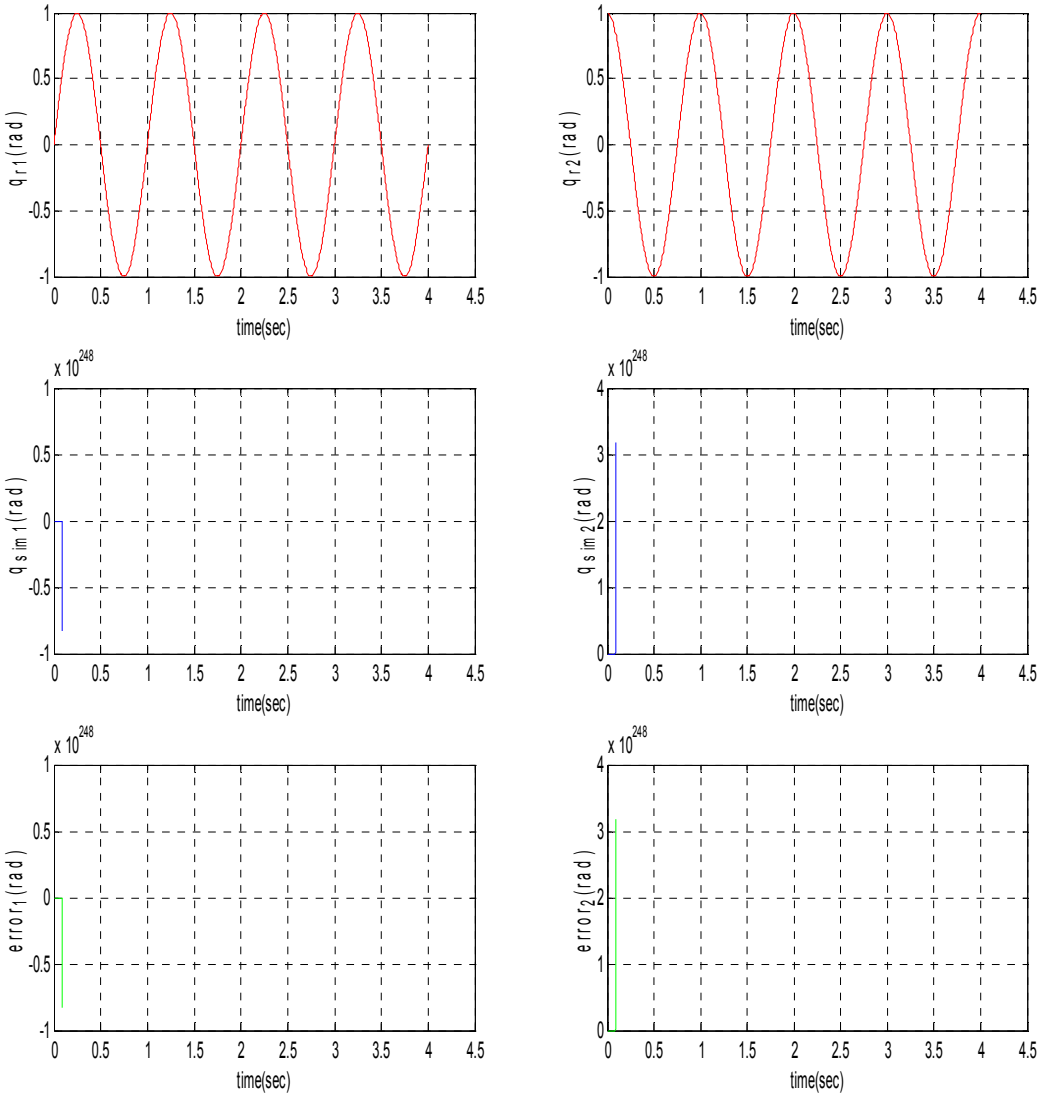

Fig. 5.39 Performance with learning rate = 2.1

For this value of the learning rate, the neural controller show drastic change for the two-link manipulator. From above results, it is observed that neural controller perform unsatisfactory tracking capability because it is not properly trained at this value.

5.8.1 SUMMARY

For value of the learning rate till two neural network give good response because it is properly, but the value more than two, the neural controller is not trained properly. That why neural controller is giving unsatisfactory response.

5.9 CONCLUSION

In this chapter the various simulation results are shown which is carried out in MATLAB simulink. The performance of the one-link manipulator and two-link manipulator with neural controller is presented. From this simulation results, it is concluded that one link manipulator for neural controller give good response for following parameters:

1) Length of robotic manipulator should be between 0.3 metre and 3 metre.
2) Mass of robotic manipulator m, should be between 0.08 kg and 4 kg.
3) Continuous uncertainties constant c, should be between 0.01 and 2.
4) For neural network design, we may use “logsig”, “purelin”, “satlins”, “poslin”, “ tansig” transfer function, but the “poslin” and “purelin” give better result for the first layer of the neural network.
5) For neural network design, we may use “purelin”, “satlins”, “tansig” transfer function, but “purelin” give better result for the second layer of the neural network.

   It can say that from point 4 and point 5, for neural controller design for one link robotic manipulator, “poslin” and “purelin” or “purelin” and “purelin” transfer function combination should be used for find out satisfactory performance.

It is also concluded for the given two-link manipulator with neural controller that for proper training and satisfactory tracking capability the learning rate should be 2 or less than two.

CHAPTER VI
CONCLUSION AND FUTURE SCOPE OF WORK

6.0 CONCLUSION

In this project, dynamics and performance simulation with neural controller for one-link and two-link manipulator through neural controller is described in details. Effect of parameter variation which is related to the robotic manipulator or neural controller has been studied and simulation performance is obtained and analyzed in detail. Neural networks and various controller of robotic manipulator are also discussed in brief.

6.1 FUTURE SCOPE OF WORK

There are many other learning rules for neural controller which could be use for comparison of the response of one-link and two-link manipulator. Effect of parameters of two-link manipulator by which variation and analysis could be further analyse. Variation in the reference trajectory may be split way and necessary analysis of performance with the neural controller could be determined. Neural controller identification quality may be also analyzed for the two link manipulator.

REFERENCES

1. Ghania Debbache, Abdelhak Bennia, Noureddeine Golea, “Neural Networks-based Adaptive State Feedback Control of Robot Manipulators”, International Journal of Computers, Communication & control, Vol. 2, No.4,200,pp.328-339.
2. Andon V. Topalov, Okyay Kaynak, Gokhan Aydin, “Neuro-adaptive sliding-mode tracking control of robot manipulators”, International Journal of Adaptive Control and Signal Processing, Int.J.Adapt. Control Signal Process. 2007,27:674-691.
3. H.A.Talebi,R.V.Patel,K.Khorasani, “A Neural Network Controller for a Class of Nonlinear Non-minimum phase systems with application to a Flexible-Link Manipulator”, Journal of Dynamic Systems, Measurement, and Control, Vol. 127, June 2005, pp. 289-294.
4. T. Efrati, H. Flashner, “Neural network based tracking control of mechanical systems”, Transactions of the ASME, Vol.121, March 1999, pp. 148-154.
5. Kazuhiku Takahashi, Ichiro Yamada, “Neural-Network Based Learning Control of Flexible Mechanism with Application to a Single-Link Flexible Arm”, Transactions of the ASME, Vol. 116, December 1994, pp. 792-795.
6. E.B.Kosmatopoulos, A.K. Chassiakos, M.A.Cchrisodoulou, “Robot Identification using Dynamical Neural Networks”, Proceeding of the 30th Conference on Decision and Control Brighton, England, December 1991, pp.2934-2935.
7. Toshio Fukunda, Takanori Shibata, “Neuromorphic Control for Robotic manipulators”, IEEE, 1990, pp.310-315.
8. Amor Jnifene, William Andrews, “Experimental Study on Active Vibration Control of a Single-link Flexible Manipulator Using Tolls of Fuzzy Logic and Neural Networks”, IEEE Transactions on Instrumentation and Measurement, Vol. 54, No.3, June 2005, pp. 1200-1208.
9. Byung Kook Yoo, Woon Chul Ham, “Adaptive Control of Robotic Manipulator Using Fuzzy Compensator”, IEEE Transactions on Fuzzy Systems, Vol.8, No.2, April2000,pp. 186-199.
10. Bogdan M. Wilamowsaki, “Methods of Computational Intelligence”, IEEE International Conference on Industrial Technology (ICIT), 2004.

11. Patrica Melin, Oscar Castillo, “Soft Computing for Intelligent Control of Nonlinear Dynamical Systems”, International Journal of Computational Cognition, Vol.2, No. 1, March 2004, pp. 45-78.
12. Rahmat Shoureshi, “Intelligent Control Systems: Are They for Real?”, Transactions of the ASME, Vol. 115, June 1993, pp. 392-401.
13. G.W.Ng, P.A.Cook, “Neural Networks in Control of Systems with Unknown and Varying Time-Delays”, UKACC International Conference on Control, Sep. 1996, pp.188-193.
14. Liang Jin, Peter N. Nikiforuk, Madan M. Gupta, “Dynamic Recurrent Neural Networks for Control of Unknown Nonlinear System”, Journal of Dynamic Systems, Measurement, and Control,Vol.116, December 1994, pp.567-576.
15. Fu-Chuang Chen, “Back-Propagation Neural Networks for Nonlinear Networks Self-Tuning Adaptive Control”, IEEE Control Systems Magazine,1990, pp. 44-48.
16. Kumpati S. Narendra, Kannan Parthasarathy, “Identification and Control of Dynamical Systems Using Neural Networks”, IEEE Transactions on Neural Networks, Vol.1, March 1990, pp. 4-26.
17. S.M.Yang, G.S.Lee, “Neural Networks Design by Using Taguchi Method”, Transactions of the ASME, Vol.121, Sep. 1999, pp.560-563.
18. M.O.Tokhi, A.K.M.Azad, H.Poerwanto, “SCEFMAS: An Environment for Dynamic Characterisation and Control of Flexible Robot Manipulators”, Int.J. Engng. Ed., Vol.15, No.3, 1999, pp.213-226.
19. K.S.Fu, R.C.Gonzalez, C.S.G.Lee, “Robotics”, McGraw-Hill Book Company.
20. S.R.Deb, “Robotics Technology and Flexible Automation”, Tata McGraw-Hill Publication Company Limited.
21. Robin R. Murfy, “Introduction to AI Robotics”, Prentice Hall of India Private Limited.
22. Martin T. Hagan, Howard B. Demuth, Mark Beale, “Neural Network Design”, Vikas Publication House.