DESIGN AND OPTIMIZATION OF A FUZZY-NEURAL HYBRID CONTROLLER STRUCTURE FOR A RUBBERTUATOR ROBOT USING GENETIC ALGORITHMS

by

Erdem Erdemir

B.S.,Mechanical Engineering, Boğaziçi University, 2002

Submitted to the Institute for Graduate Studies in
Science and Engineering in partial fulfillment of
the requirements for the degree of
Master of Science

Graduate Program in Systems and Control Engineering

Boğaziçi University

2006

ACKNOWLEDGEMENTS

I would like to express my appreciation to my thesis supervisor Professor Mehmed Özkan, not only for guiding, support and encouragement, but also the crucial role, he played in building up my future academic career in Vanderbilt University.

Furthermore, I would like to express my sincere gratitude to Professor Yorgo İstefanopulos for his professional support and constructive criticisms.

It is my pleasure to thank Ali Polat, Deniz Diktaş, Özkan Taşçıoğlu, Ümit Uğurlu, Murat Fırat, Erdal Kayacan, Alper Yaman, who have contributed in various ways to the completion of this thesis and Alper Onur, Ender Kasım, from Elektra Company for their technical support and contribution to the modernization of the robot arm by constructing the appropriate transformers. Also, I want to thank to my managers from GSD, Serdar Kolbaşı and Osman Kurdaş, for the extreme patience and flexibility throughout the study.

My sincere thanks go to my family that has always been there for their endless support, love, and encouragement throughout my whole life.

Last but not least, I would like to thank my fiancée, Aysu Kızıltaş, and her family, who have always encouraged and supported me to finish my thesis.

ABSTRACT

DESIGN AND OPTIMIZATION OF A FUZZY-NEURAL HYBRID CONTROLLER STRUCTURE FOR A RUBBERTUATOR ROBOT USING GENETIC ALGORITHMS

This study presents a combination of soft computing techniques, namely back propagation neural network, fuzzy and genetic algorithms that are used to control the Bridgestone Hybrid Robot Arm (BHRA).

The workspace of the BHRA's end effector is divided into small segments and the trajectory independent parameters of all these segments are learned by training small size (only three nodes) neural networks for each segment. The structure of these neural networks is based on the physical model, which is derived from the Language-Euler mechanics of the robot arm. To maintain continuity on the small neural networks, we use a basic fuzzy algorithm whose fuzzy membership function parameters are optimized by genetic algorithm (GA). The proposed technique's performance was compared with only-neural network controller and shown to be more accurate in trajectory control for rubbertuator robots.

The main goal of this study is to maintain a better off-line control on the rubbertuators by using only small size (3 nodes and one hidden layer) neural networks and a simple fuzzy algorithm with minimal linguistic variables and minimal number of rules. On the other hand, finding a better off-line control ensures that small learning rates will be sufficient for future on-line training control, and choosing small learning rates will decrease the prospect of divergence and the risk of instability in control.

ÖZET

KAUÇUK EYLEYİCİLERDEN OLUŞAN BİR ROBOT KOLU İÇİN,MELEZ BULANIK MANTIK-YAPAY SİNİR AĞLARI İLE DENETİM YAPISININ TANIMLANMASI VE GENETIK ALGORİTMALARLA ENİYİLENMESİ

Bu çalışma, Bridgestone firmasının ürettiği, motor ve kauçuk eyleyicilerden oluşan bir robot kolunun denetimini için, geri yayılım yapay sinir ağları,bulanık mantık ve genetik algoritmaları gibi esnek yöntemlerin, kullanımını kapsamaktadır.

Robot kolunun, uç işlevcisinin çalışma alanı küçük parçalara ayrılıp, her bir parçanın, yörüngeden bağımsız parametreleri, küçük yapay sinir ağları kullanılarak öğrenilmiştir. Bu küçük yapay sinir ağlarının yapısı, robot kolunun Langrage - Euler mekaniğine dayanmaktadır. Bu yapay sinir ağları arasında sürekliliği sağlayabilmek için, üyelik fonksiyonu değişkenleri, genetik algoritmalarla eniyilenmiş basit bir bulanık mantık yöntemi kullanılmıştır. Yörünge izlemede, önerilen yöntemin performansı, sadece yapay sinir ağlarından oluşan denetleyicinin performansından daha iyi olduğu gösterilmiştir.

Bu çalışmanın asıl amacı, küçük yapay sinir ağları(3 düğüm ve bir gizli katmandan oluşan) ve minimum sayıda dilbilimsel değişkenler ve kuralları olan bulanık mantık ile, kauçuk eyleyiciler üzerinde, iyi bir çevrimdışı denetim sistemi geliştirmektir.Diğer taraftan,daha iyi bir çevrimdışı denetiminin bulunması, gelecekte kullanılacak çevrimiçi denetiminde, daha küçük öğrenme katsayısı kullanılabilmesini sağlayacaktır ve bu da, ıraksama ve kararsızlık ihtimalini azaltacaktır.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

LIST OF SYMBOLS/ABBREVIATIONS

a_i	D-H parameter for ith link
C_x	Cosine of x
d_i	D-H parameter for ith link
e	Error
$\hat{E}$	Estimated Error
$\tilde{E}$	Calculated Error
F	Force
g	Gravity Force
I_c	SVO control current
K	Total kinetic energy for the selected link
L	Lagrangian difference
L_i, l_i	Length of the ith link
m_i	Mass for ith link
P	Total potential energy for the selected link
P_0	Equilibrium pressure
P_d	Desired pressure
P_f	Final Pressure
P_m	The rubbertuator pressure
P_{Wx}	Position of the W in the x direction
P_{Wy}	Position of the W in the y direction
P_x	Position of the end-effector in the x direction
P_y	Position of the end-effector in the y direction
q	Joint Variable
q_i	Joint Variable for ith link
r	Radius
S_x	Sine of x
t	Time
T, A	Transformation Matrix
w_i	Weights of the neural network

w_{jk}	Weight of the jth hidden neuron for kth output neuron
w_{0k}	Weight of the threshold for kth output neuron
x_i	ith input unit
V	Voltage
v_{ij}	Weight of the ith Input for jth hidden neuron
v_{0j}	Weight of the Threshold for jth hidden neuron
y_in_k	Output of the output neuron k after activation
y_k	Output of the output neuron k after activation
z_in_j	Output of the hidden neuron j before activation
z_j	Output of the hidden neuron j after activation
α	Rubbertuator constant
α_i	D-H parameter for ith link
β	Rubbertuator constant
γ	Rubbertuator constant
δ_k	Error information term of the kth output neuron
ε	Contraction Ratio
θ_g	Generalized angle
$\theta_i, \dot{\theta}_i \ddot{\theta}_i$	Angle,Angular velocity and acceleration of the ith joint
σ	Learning rate
τ	Torque input vector
ϕ	Orientation angle
$ANNET$	Actuator Control Neural Network
$BHRA$	Bridgestone Hybrid Robot Arm
$FPONNET$	Fuzzy Position Control Neural Network
$MONNET$	Motor Control Neural Network
PE	Neural networks processing elements
$PONNET$	Position Control Neural Network
SVO	Servo valve

1. INTRODUCTION

Robot manipulators have become very important in the field of flexible automation. Electric actuators are widely used in robot manipulators and automation systems, due to their advantages of precise control, portability, and cleanliness. However, they also have some disadvantages like large size-to-torque ratio, proneness to environmental hazards such as fire, dirt, and moisture and so on. As a result, the robotics community has been investigating a wide range of actuators for alternatives, especially for the tasks involving human-machine interactions [1].

Rubbertuators (Rubber-Actuator or Pneumatic Muscles), provide an important safe human-robot-cooperation technology that can be used in manipulation, automation and robotic tasks. Lightweight, high power-weight ratio, compliant and spark free nature properties make the rubbertuators advantageous over rigid-classical manipulators especially in explosive and human-robot interaction environments.

A pneumatic drive system using rubbertuators has features of low friction, high compliance and large power height ratio. However, it has been difficult to get the satisfactory tracking performance for the reference trajectory because of the high compressibility of air, poor damping ability, the strong nonlinearity of rubbertuator and the time lag of valve operation [2].

Accordingly, the application of a new control strategy is expected to overcome some of the difficulties mentioned above. In the last decade, techniques based on fuzzy systems, neural networks, genetic algorithms, and various methods of probabilistic reasoning could offer solutions to highly nonlinear systems [3]. So, in the previous study of a pneumatically driven rubbertuator robot arm, a Bridgestone Hybrid Robot Arm (BHRA) shown in Figure 1.1, with one translate-joint and four rotary joints is the plant to be controlled. The physical model of the robot dynamics is combined with a multi-layer backpropagation neural network system. The approach taken proved to be more advantageous in solving a real life problem, and the resulting trajectory control

performance was superior when compared to a well-tuned PID controller output that has been utilized in commercial versions of the same robot system [3]. Nevertheless, the linear PID algorithm might be insufficient to deal with processes with complex dynamics, such as those with large dead time and highly nonlinear characteristics [4].

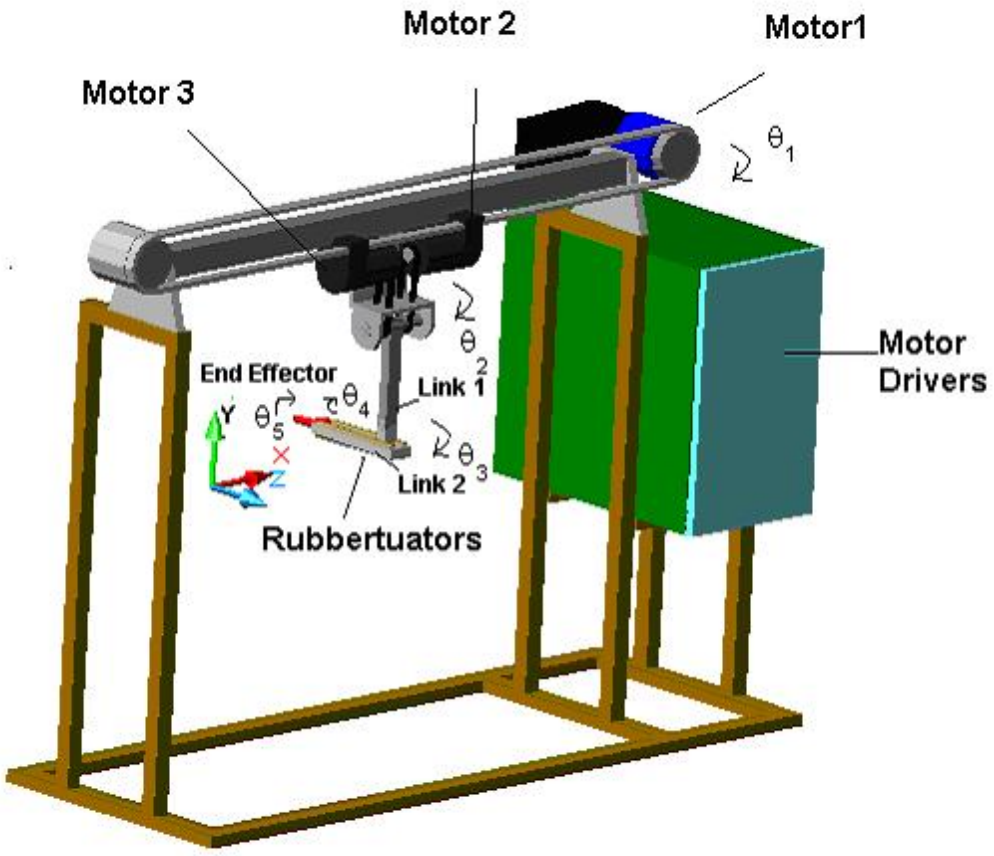

Figure 1.1. Bridgestone Hybrid Robot Arm (BHRA)

1.1. Objective

The main objective of this research is to develop a better off-line fuzzy-neural hybrid controller for BHRA by using only small size (3 nodes and one hidden layer) neural networks and a simple fuzzy algorithm with minimal linguistic variables and minimal number of rules.

1.2. Methodology

Neural networks are essentially low-level computational structures and algorithms that offer good performance in dealing with sensory data, while fuzzy logic techniques often deal with issues such as reasoning on a higher level than neural networks. However, since fuzzy systems do not have much learning capability, it is difficult to tune the fuzzy rules and membership functions from the training data set.On the other hand, as the internal layers of neural networks are always opaque to the user, the mapping rules in the network are not visible and are difficult to understand; furthermore, the convergence of learning is usually very slow and is not guaranteed. Thus, a promising approach for reaping the benefits of both fuzzy systems and neural networks is to merge or fuse them into an integrated system [5]. In this thesis, the learning capability of the neural networks is used to learn the trajectory independent parameters and the fuzzy system is used to manage these neural networks in order to improve the trajectory tracking performance of BHRA.

In a fuzzy-neural hybrid system, both fuzzy logic techniques and neural networks are utilized separately to establish two decoupled subsystems which perform their own tasks in serving different functions in the combined system. The architecture of fuzzy-neural hybrid systems is usually application-oriented. Making use of their individual strengths, fuzzy logic and neural network subsystems complement each other efficiently and effectively to achieve a common goal [5].

In the fuzzy-neural hybrid control structure of BHRA, the physical model of the robot dynamics is combined with a multi-layer backpropagation neural network system. In order to decrease the nonlinear terms, the workspace of the end effector is divided into 4 degree segments, and for each segment, a neural network is trained using the physical model of the system. Another fact, that has to be taken into account, is the hysteretic properties of the rubbertuators. In order to overcome the hysteresis problem of the rubbertutors, two neural networks are used for forward and backward direction of the motion. The discrete consecutive neural networks are combined by using a fuzzy algorithm to get a better and smooth control system for BHRA. The weights of the

rules are determined using genetic algorithm.

1.3. Approach to the Problem and Organization of the Thesis

First, in Chapter 2, the system architecture of the robot arm will be introduced. In Chapter 3, the direct and inverse kinematic analysis will be explained for a three-link planar manipulator. In Chapter 4, the fuzzy-neural hybrid control structure of BHRA will be explained. In Chapter 5, the trajectory tracking performance results of the fuzzy-neural hybrid control system and the only-neural control system will be presented. In Chapter 6, details of OpenGL-C++ based animation software will be explained. In Chapter 7, a summary of the study, its results, conclusions and suggestions for further work on the subject will be given.

2. SYSTEM ARCHITECTURE

Originally, the BHRA was designed for painting applications on a horizontal surface by moving the arm back and forth. Several painting trajectories were downloaded into the memory of the ex-control system, which used to be a transputer. Transputer is the name of a series of processors produced by INMOS. Transputer has a RISC architecture designed to support parallel processing at the lowest processing level [6]. The old system architecture of the BHRA is show in Figure 2.1.

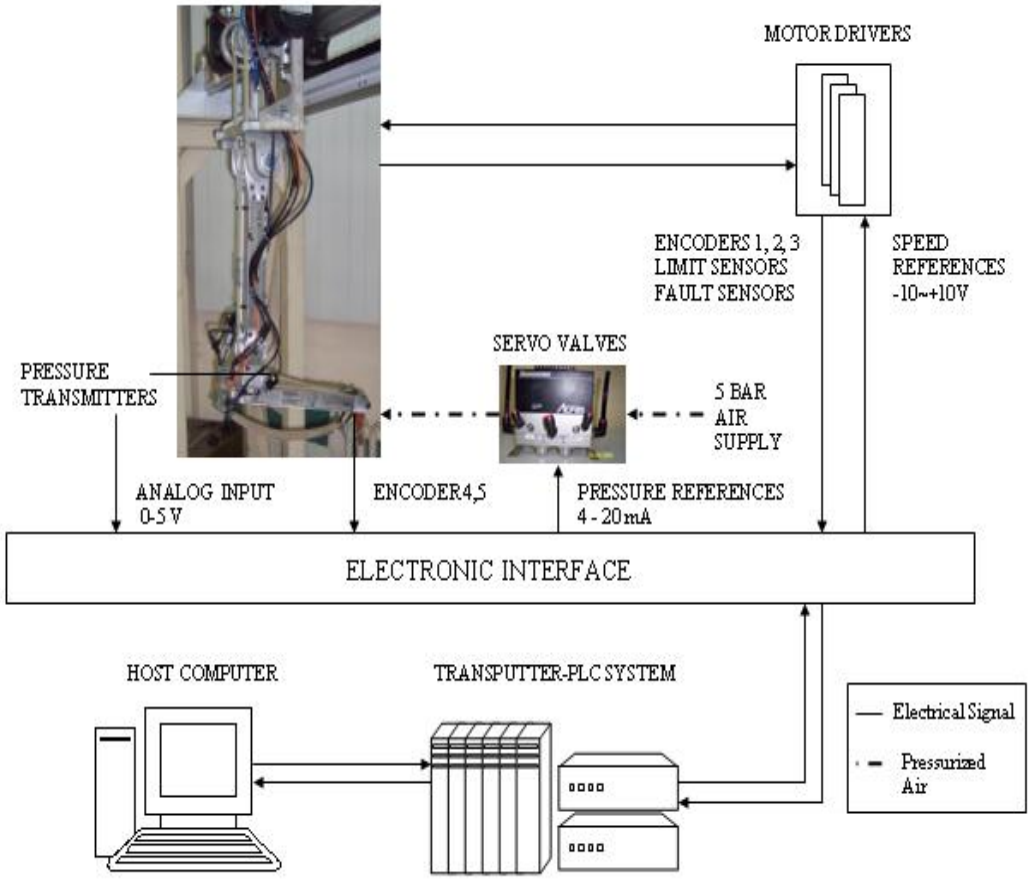

Figure 2.1. The old system architecture of BHRA

In the modernization of BHRA, first, the control system has been changed, because the existing system could not supply a real-time control on BHRA. On the other hand, the SVOs(servo valves) have been changed to overcome the hysteresis problem of the rubbertuators.The new system architecture of the BHRA is show in Figure 2.2.

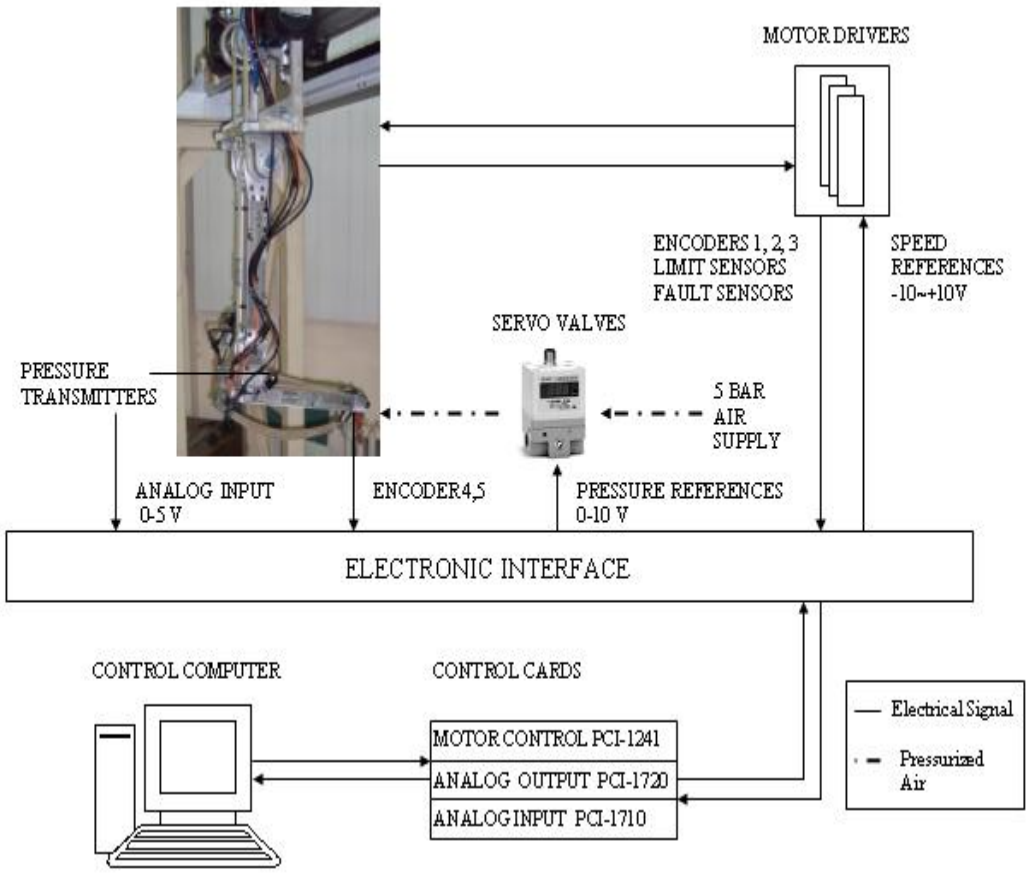

Figure 2.2. The new system architecture of BHRA

2.1. The System Components

The hardware structure of the system is composed of power supply, motors, servo valves, pressure transmitters, limit switches and encoders.

2.1.1. The Power Supply

The old system was designed for 3 phase- 200V AC power supply, which is 3 phase standard in Japan. In order to use it in Turkey, Delta-Delta type 380V AC-3 phase to 200V AC-3 phase transformer is used. The wiring digram of the power supply of the system is shown in Figure 2.3.

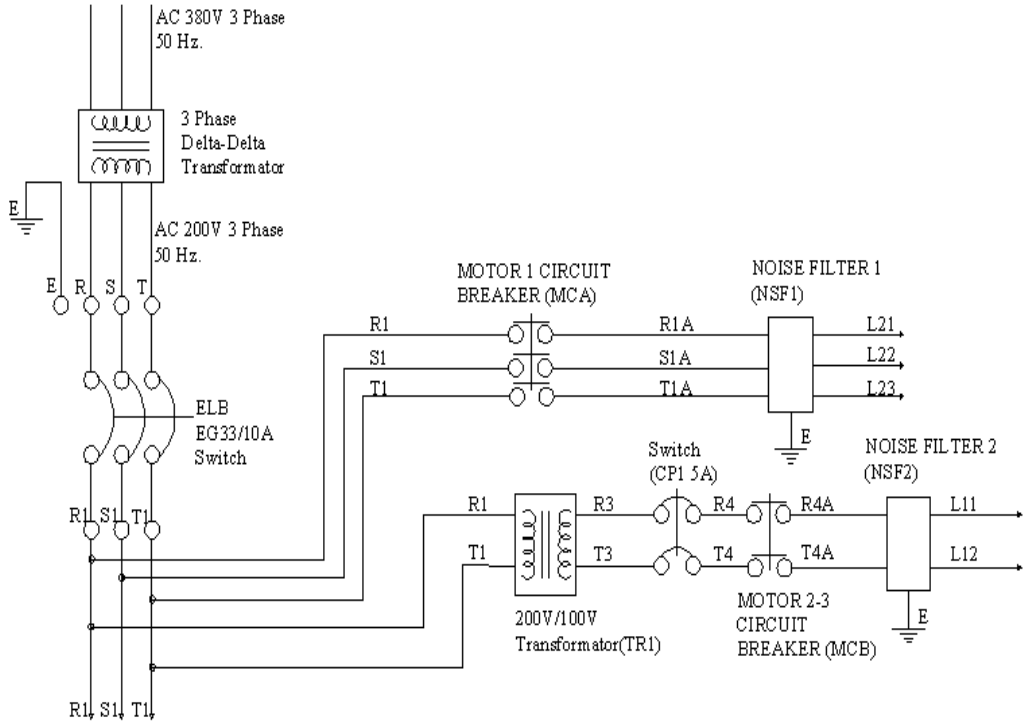

Figure 2.3. The wiring diagram of the power supply

2.1.2. The Motors and The Switches

There are three motors and six limit switches. Motor 1, shown in Figure 1.1, is a three phase-100V AC motor. The speed references of the motors are given by the motor control card as voltage references, shown in Figure 2.4, 2.5, 2.6. The end switches of the Motor1 give simple contact outputs and these outputs activate 24V relays, in order to send "limit switch high" or "limit switch low" signal to the motor control card.

The limit switches of the Motor2 and Motor3 give current outputs and these outputs are converted to contact output by a curent-contact converter, shown in Figure 2.5, 2.6, in order to send limit switch high or limit switch low signal to the motor control card.

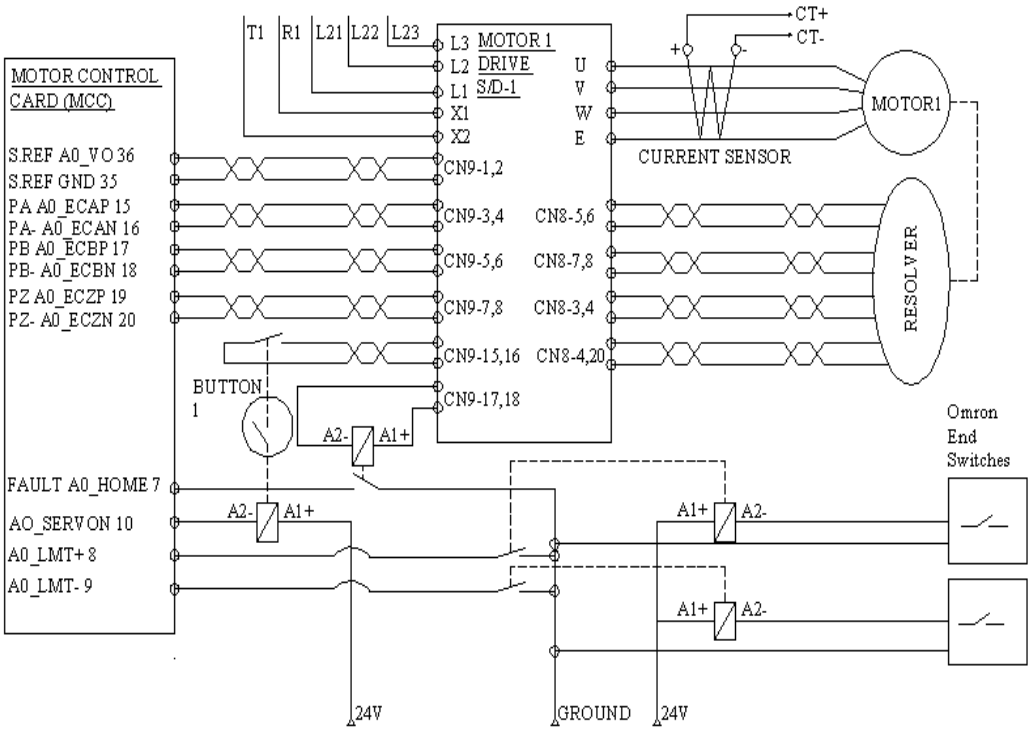

Figure 2.4. The wiring diagram of Motor1

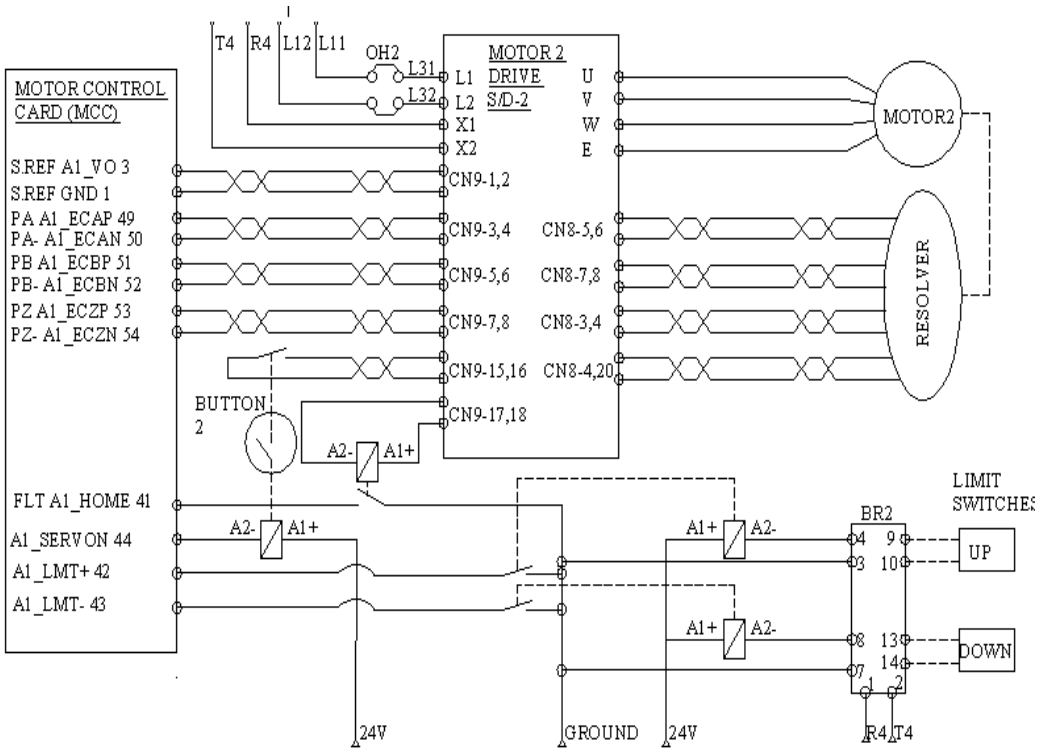

Figure 2.5. The wiring diagram of Motor2

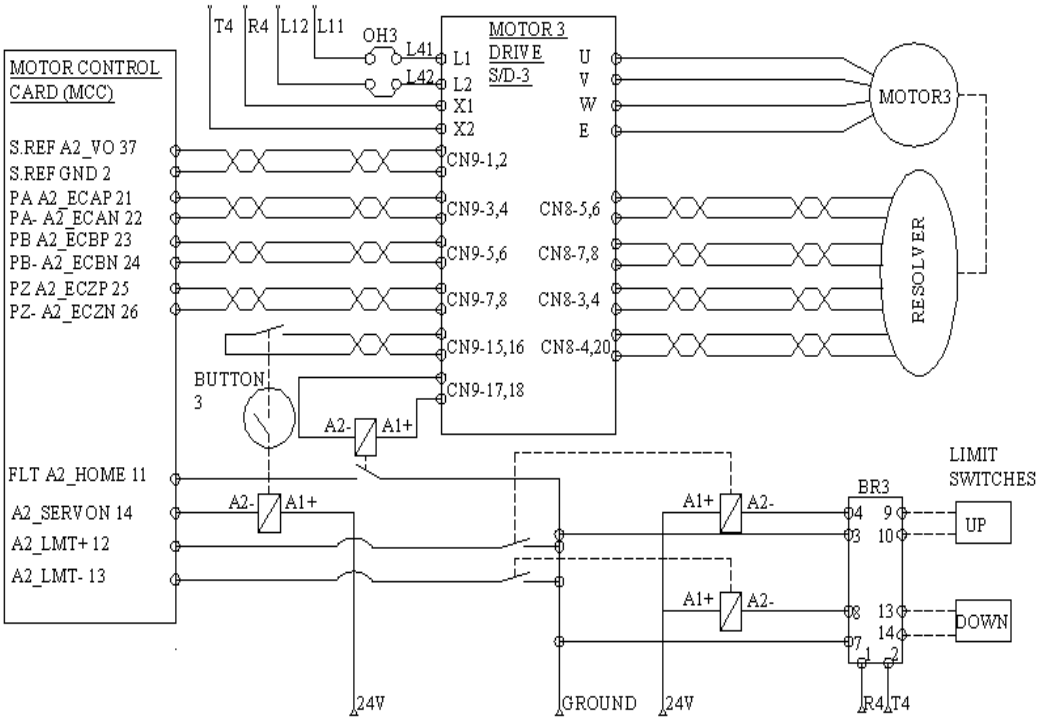

Figure 2.6. The wiring diagram of Motor3

2.1.3. The Servo Valves and The Pressure Transmitters

Servo valve is the unit to control the pressure of the air in the rubbertuators by electrical voltage input.The voltage input should be in the range of 0V and 10V. The output, the controlled pressure, is linearly proportional to the electrical voltage input. The wiring diagram of the servo valves is shown in Figure 2.7.

The pressure transmitter is the unit to measure the air pressure in the rubbertuators. The voltage output of the transmitters is between 0V and 5V.The wiring diagram of the pressure transmitters is shown in Figure 2.8. To filter the voltage output of the transmitters, a simple RC filter is used for each channel.

2.1.4. The Emergency System

In order to maintain the safety, the system cuts off the power of the motors in such cases:

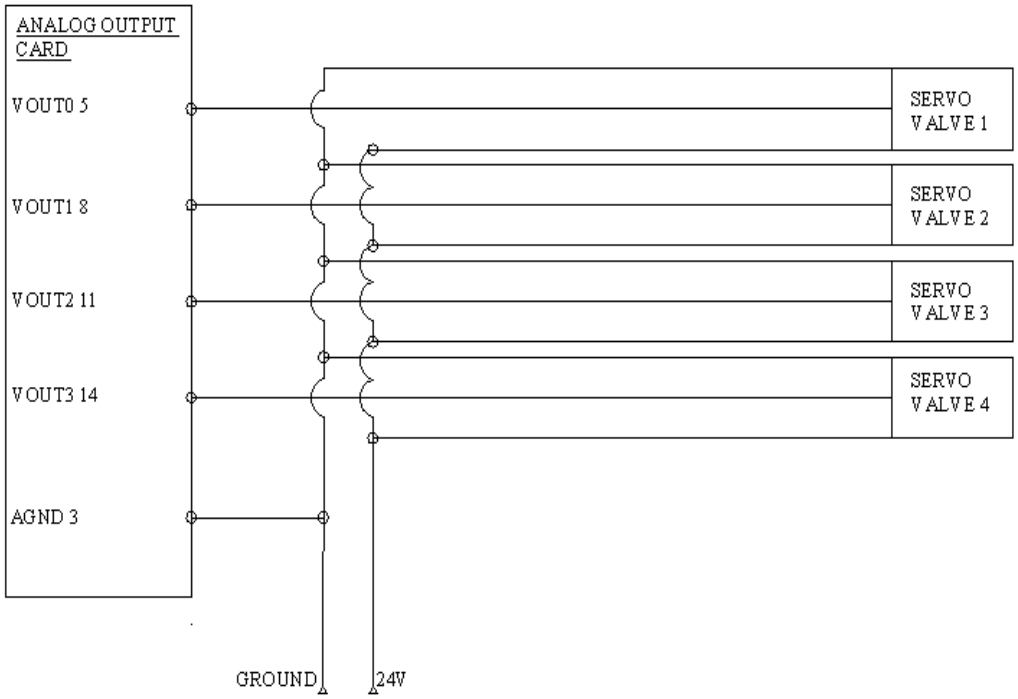

Figure 2.7. The wiring diagram of the servo valves

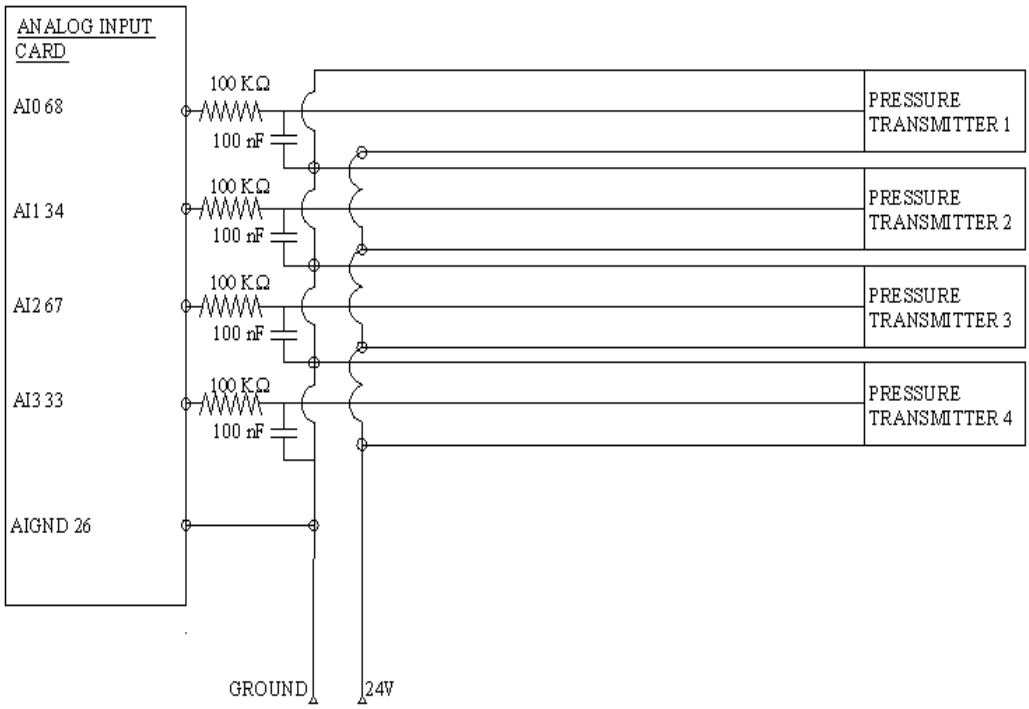

Figure 2.8. The wiring diagram of the pressure transmitters

- If the red-emergency alarm button is pressed,
- If the current consumption of the Motor1 is very high,
- If the over heat sensor of the Motor2 is on,
- If the over heat sensor of the Motor3 is on,

The wiring diagram of the emergency system is shown in Figure 2.9.

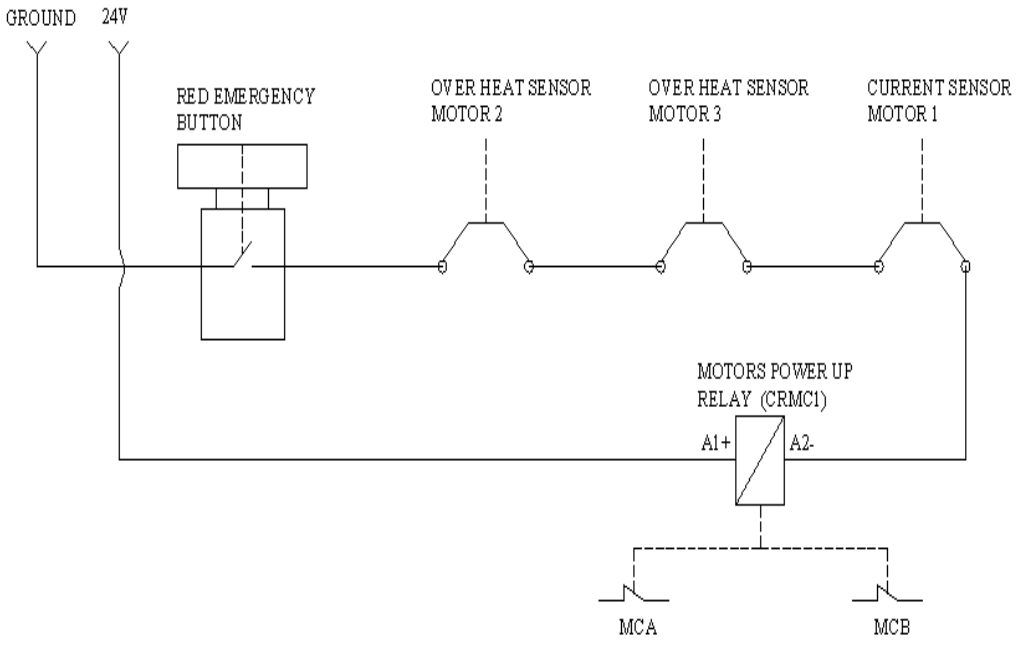

Figure 2.9. The wiring diagram of the emergency system

3. ROBOT KINEMATICS

A manipulator can be schematically represented from a mechanical viewpoint as a kinematic chain of rigid bodies (links) connected by means of revolute or prismatic joints. One end of the chain is constrained to a base, while an end effector is mounted to the other end. The resulting motion of the structure is obtained by composition of the elementary motions of each link with respect to the previous one. Therefore, in order to manipulate an object in space, it is necessary to describe the end-effector position and orientation [7].

Three electric motors and four rubbertuators drive the five-degree of freedom robot BHRA shown in Figure 1.1. The first joint, θ_1, is a translate joint moving the robot arm along the x-axis. The next two joints, θ_2 and θ_3 are the rotary joints. The two rubbertuator joints with angles of rotations θ_4 and θ_5 are differential pairs responsible for the approach and the orientation angles.

This chapter is dedicated to the derivation of the direct kinematics. The Denavit-Hartenberg convention [8], which will allow the end-effector position and orientation to be expressed as a function of the joint variables of the mechanical structure with respect to a reference frame, shown in Figure 3.1, and to construct the direct kinematics function by composition of the individual coordinate transforms expressed by Equation 3.1 into one homogenous transformation matrix as in Equation 3.2.

$$A_i^{i-1}(q_i) = \begin{bmatrix} Cos\theta_i & -Sin\theta_i Cos\alpha_i & Sin\theta_i Sin\alpha_i & a_i Cos\theta_i \\ Sin\theta_i & Cos\theta_i Cos\alpha_i & -Cos\theta_i Sin\alpha_i & a_i Sin\theta_i \\ 0 & Sin\alpha_i & Cos\alpha_i & d_i \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{3.1}$$

where a_i is the normal distance from z_i to z_{i-1} measured along x_i, α_i is the angle between z_i and z_{i+1} measured about x_i, d_i is the distance from x_{i-1} to x_i measured along z_{i-1}, θ_i is the angle between x_{i-1} to x_i measured about z_{i-1}, q_i is the joint

variable.

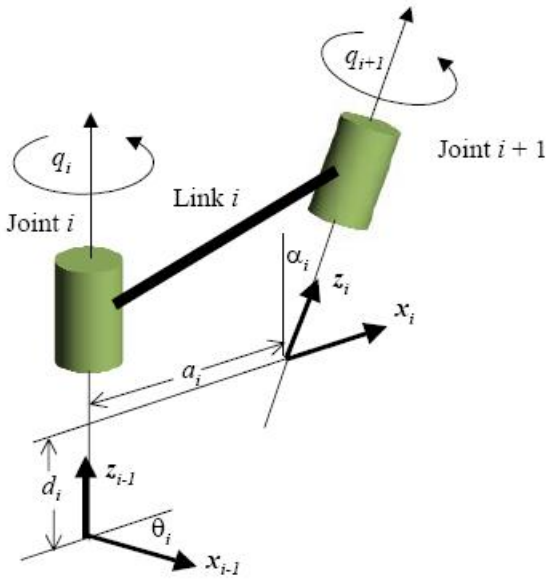

Figure 3.1. The Denavit-Hartenberg link relationship [8]

$$T_n^0(q) = A_1^0(q_1)A_2^1(q_2)\dots A_n^{n-1}(q_n) \tag{3.2}$$

The chapter ends with the derivation of solutions to the inverse kinematics problem, which consists of the determination of the joint variables corresponding to a given end-effector configuration.

3.1. Direct Kinematics

The BHRA corresponds to a three-link planar arm with an additional translate joint, θ_1 along the x-axis and the orientation joint, θ_5 about the end-effector, shown in Figure 1.1. In this respect,finding the direct and inverse kinematics of the translate joint and the orientation joint are very simple. On the other hand, finding the direct and inverse kinematics of revolute joints, θ_2, θ_3 and θ_4 are not as simple as the others.

In order to find direct kinematics of a three-link planar arm, the coordinate frames and Denavit-Hartenberg parameters [8] are attached to the arm, shown in Figure 3.2 and in Table 3.1. The lenghts of the links are l_1, l_2, l_3. The corresponding homogeneous

link transformations are obtained by inserting the Denavit-Hartenberg parameters from Table 3.1 into Equation 3.1 and shown in Equations 3.3, 3.4, 3.5.

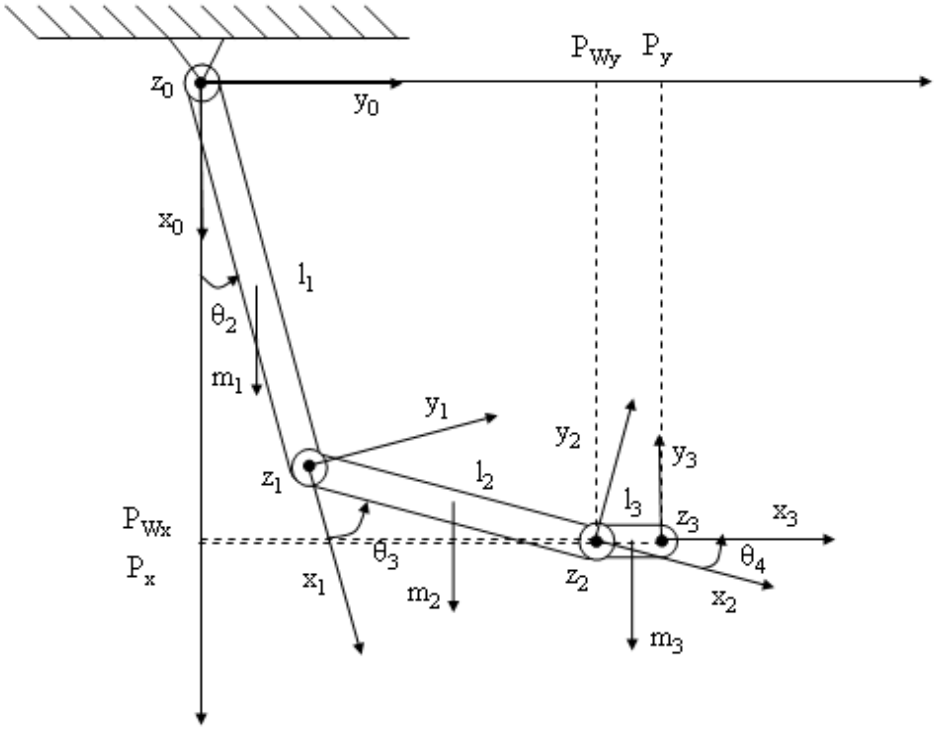

Figure 3.2. The coordinate frame attachment to three-link planar manipulator

Table 3.1. The Denavit-Hartenberg Parameters

Link i	α_i	d_i	ν_i
1	0	l_1	θ_2
2	0	l_2	θ_3
3	0	l_3	θ_4

$$A_1^0 = \begin{bmatrix} C_2 & -S_2 & 0 & l_1C_2 \\ S_2 & C_2 & 0 & l_1S_2 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{3.3}$$

$$A_2^1 = \begin{bmatrix} C_3 & -S_3 & 0 & l_2C_3 \\ S_3 & C_3 & 0 & l_2S_3 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{3.4}$$

$$A_3^2 = \begin{bmatrix} C_4 & -S_4 & 0 & l_3C_4 \\ S_4 & C_4 & 0 & l_3S_4 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{3.5}$$

where C_i=$Cos\theta_i$, S_i=$Sin\theta_i$. Using the homogenous transformation matrix equation as in 3.2:

$$T_3^0 = A_1^0 A_2^1 A_3^1 \tag{3.6}$$

$$T_3^0 = \begin{bmatrix} C_{234} & -S_{234} & 0 & l_1C_2 + l_2C_{23} + l_3C_{234} \\ S_{234} & C_{234} & 0 & l_1S_2 + l_2S_{23} + l_3S_{234} \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{3.7}$$

where C_{ijk}=$Cos(\theta_i + \theta_j + \theta_k)$, S_{ijk}=$Sin(\theta_i + \theta_j + \theta_k)$. So the end-effector position will be:

$$P_x = l_1C_2 + l_2C_{23} + l_3C_{234} \tag{3.8}$$

$$P_y = l_1S_2 + l_2S_{23} + l_3S_{234} \tag{3.9}$$

$$P_z = 0 \tag{3.10}$$

3.2. Inverse Kinematics

The direct kinematics equation, in the form of Equation 3.2 establishes the functional relationship between the joint variables and the end-effector position. The inverse kinematics problem consists of the determination of the joint variables corresponding to a given end-effector position and orientation. The solution to this problem is of fundamental importance in order to transform the motion specifications, assigned to the end-effector in the operational space, into the corresponding joint space motions that allow execution of the desired motion [7].

It is desired to find the joint variables θ_2, θ_3, θ_4 corresponding to a given end-effector position and orientation. Using algebraic solution technique and having specified the orientation, the relation will be:

$$\phi = \theta_2 + \theta_3 + \theta_4 \tag{3.11}$$

which is one of the equations of the system to solve. From Equation 3.7 the following equations can be obtained:

$$P_{Wx} = P_x - l_3 C_\phi = l_1 C_1 + l_2 C_{12} \tag{3.12}$$

$$P_{Wy} = P_y - l_3 S_\phi = l_1 S_1 + l_2 S_{12} \tag{3.13}$$

which describe the position of point W. Squaring and summing the two Equations 3.12 and 3.13:

$$P_{Wx}^2 + P_{Wy}^2 = l_1^2 + l_2^2 + 2 l_1 l_2 C_2 \tag{3.14}$$

from Equation 3.14,

$$C_3 = \frac{P_{Wx}^2 + P_{Wy}^2 - l_1^2 - l_2^2}{2l_1 l_2} \tag{3.15}$$

The solution of Equation 3.15, C_3 can be between -1 and 1 but from the mechanical constraints of the robot arm, it can

$$S_3 = \sqrt{1 - C_3^2} \tag{3.16}$$

$$\theta_3 = Atan2(S_3, C_3) \tag{3.17}$$

Having determined θ_3, the angle θ_2 can be found by substituting θ_3 into Equation 3.14.

$$S_2 = \frac{(l_1 + l_2 C_3)P_{Wy} - l_2 S_3 P_{Wx}}{P_{Wx}^2 + P_{Wy}^2} \tag{3.18}$$

$$C_2 = \frac{(l_1 + l_2 C_3)P_{Wx} + l_2 S_3 P_{Wy}}{P_{Wx}^2 + P_{Wy}^2} \tag{3.19}$$

So θ_2 is

$$\theta_2 = Atan2(S_2, C_2) \tag{3.20}$$

Finally, the angle θ_4 will be:

$$\theta_4 = \phi - \theta_2 - \theta_3 \tag{3.21}$$

4. FUZZY-NEURAL HYBRID CONTROL STRUCTURE

The BHRA control loop comprises two trained neural network, PONNET (Position Control Neural Network) and ANNET (Actuator Control Neural Network), and one FPONNET (Fuzzy Position Control Neural Network) for each rubbertuators and MONNET (Motor Control Neural Network) for each motor, shown in Figure 4.1.

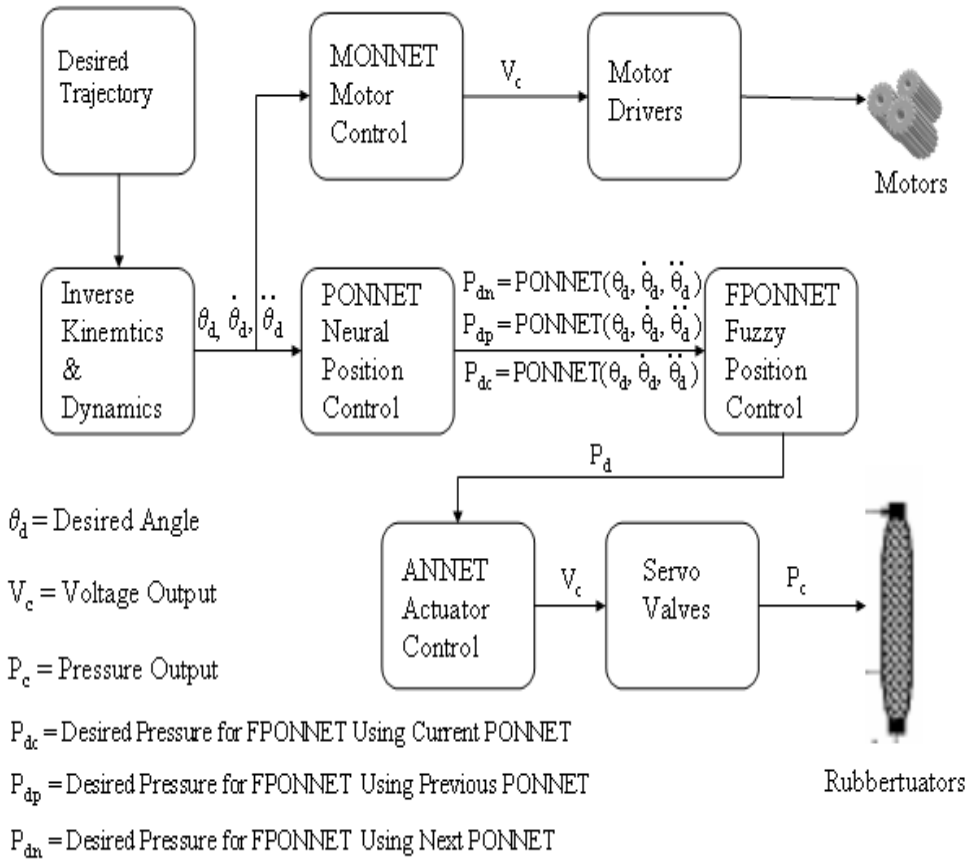

Figure 4.1. BHRA Open Loop Control Structure

The desired trajectory is given to the system and using inverse kinematics and dynamics, the desired angular position, velocity and acceleration are found. The PONNET takes the results of the inverse kinematics block and produces two pressure values such as the desired degree's PONNET result, the previous degree segment's PONNET result using the desired degree or the next degree segment's PONNET result using the desired degree. The FPONNET uses these two pressure values to compute the desired

pressure by combining the two pressure values. The ANNET is used to compensate the delay by using a second order delay function. The output of the ANNET is the desired control voltage reference for the rubbertuators' servo valves. The MONNET is a very simple neural network system to control the motors, which takes the desired trajectory and produces the velocity references as voltage outputs.

4.1. The Artificial Neural Networks

Artificial Neural Networks (ANNs) mimic biological information processing mechanisms. They are typically designed to perform a nonlinear mapping from a set of inputs to a set of outputs. ANNs are developed to try to achieve biological system type performance using a dense interconnection of simple processing elements analogous to biological neurons. ANNs are information driven rather than data driven. They are non-programmed adaptive information processing systems that can autonomously develop operational capabilities in response to an information environment. ANNs learn from experience and generalize from previous examples. They modify their behavior in response to the environment, and are ideal in cases where the required mapping algorithm is not known and tolerance to faulty input information is required [9].

ANNs contain electronic processing elements (PEs) connected in a particular fashion. The behavior of the trained ANN depends on the weights, which are also referred to as strengths of the connections between the PEs. ANNs offer certain advantages over conventional electronic processing techniques. These advantages are the generalization capability, parallelism, distributed memory, redundancy, and learning [9].

4.2. The Backpropagation Neural Network

The backpropagation neural network is a collection of nodes organized into interconnected layers. The layered structure of the backpropagation network allows it to escape the linear separability limitation making it a much more powerful tool. The backpropagation neural network is not limited to a single binary output; it can have any number of outputs whose values fall within a continuous range. The backpropagation

neural network is ideal for problems involving classification, projection, interpretation, and generalization [10].

The backpropagation neural network contains at least three layers, input, output and middle layers. The example in 4.2 shows sample backpropagation neural network with p middle layer, n input layer nodes and m output layer nodes.

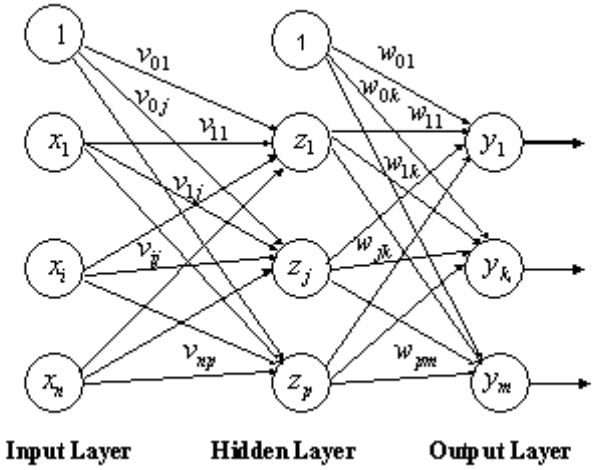

Figure 4.2. Neural network structure with one hidden layer[11]

Backpropagation neural network training involves three stages: the feedforward of the input training pattern, the calculation and backpropagation of the associated error, and the adjustment of the weights. Application of the network involves only the computations of the feedforward phase after training with new weights. The training process is slow, however the trained network can produce its output very rapidly [12]. A sample training algorithm of the backpropagation is:

Step 0. Initialize weights.

Step1. While stopping condition is false, do Steps 2-9.

Step 2. For each training pair, do Steps 3-8.

Feedforward:

Step 3. Each input unit (X_i, i=1,. . . ,n) receives input signal x_i and broadcasts this signal to all units in the layer above (the hidden units).

Step 4. Each hidden unit (Z_j, j=1,. . . ,p) sums its weighted input signals,

$$z\_in_j = v_{0j} + \sum_{i=1}^{n} x_i v_{ij} \tag{4.1}$$

applies its activation function to compute its output signal,

$$z_j = f(z\_in_j) \tag{4.2}$$

and sends this signal to all units in the layer above (output units).

Step 5. Each output unit (Y_k, k=1,. . . ,m) sums its weighted input signals,

$$y\_in_k = w_{0k} + \sum_{k=1}^{p} z_j w_{jk} \tag{4.3}$$

And applies its activation function to compute its output signal,

$$y_k = f(y\_in_k) \tag{4.4}$$

Backpropagation of error:

Step 6. Each output unit (Y_k, k=1,. . . ,m) receives a target pattern corresponding to the input training pattern, computes its error information term,

$$\delta_k = (t_k - y_k) f'(y\_in_k) \tag{4.5}$$

Calculates its weight correction term (used to update w_{jk} later),

$$\Delta w_{jk} = \sigma \delta_k z_j \tag{4.6}$$

where σ is the learning rate. Calculates its bias correction term (used to update w_{0k} later),

$$\Delta w_{0k} = \sigma \delta_k z_j \tag{4.7}$$

and sends δ_k to units in the layer below.

Step 7. Each hidden unit (Z_j, j=1,. . . ,p) sums its delta inputs (from units in the layer above),

$$\sigma\_in_j = \sum_{k=1}^{m} \sigma_k w_{jk} \tag{4.8}$$

Multiplies by the derivative of its activation function to calculate its error information term,

$$\delta_j = \delta\_in_j f'(z\_in_j) \tag{4.9}$$

Calculates weight correction term (used to update v_{ij} later),

$$\Delta v_{ij} = \sigma \delta_j x_i \tag{4.10}$$

and calculates its bias correction term (used to update v_{0j} later),

$$\Delta v_{0j} = \sigma \delta_j \tag{4.11}$$

Update Weight and Biases:

Step 8. Each output unit (Y_k, k=1,. . . ,m) update its bias and weights (j=1,. . . ,p):

$$w_{jk}(new) = w_{jk}(old) + \Delta w_{jk} \tag{4.12}$$

Each hidden unit (Z_j, j=1,. . . ,p) updates its bias and weights (i=1,. . . ,n):

$$v_{ij}(new) = v_{ij}(old) + v_{ij} \tag{4.13}$$

Step 9. Test stopping condition [11], [12].

4.3. The ANNET

The trajectory error, however, is not simply due to the unadjusted dynamics parameters, but also a result of delayed actuator response, mostly because of the air compressibility and rubber elasticity. A second neural network called ANNET is integrated with PONNET for the compensation of the delay characteristics [3].

As described in [3], we describe the rubbertuator-SVO characteristic as a simple exponential delay function as:

$$P_m = AI_c(1 - e^{(-t/T)}) \tag{4.14}$$

where is a scaling constant, P_m the rubbertuator pressure, I_c the SVO control current and T the time constant of the rubbertuator-SVO system. It can be shown that P_m becomes equal to the target pressure, P_d with no delay, if the control input to the SVO is:

$$I_c(t) = \frac{T}{A}\frac{dP_d(t)}{dt} + \frac{P_d(t)}{A} \tag{4.15}$$

Based on Equation 4.15, the input nodes of ANNET neural network are simply two:

one for the desired pressure signal and one for the rate of change of the desired pressure signal. A bias term is also included for the completeness of the neural network. Therefore, the simple form of the neural network will be [3]:

$$net_o = w_1 \frac{dP_d(t)}{dt} + w_2 P_d(t) + w_3 \tag{4.16}$$

where w_1,w_2 and w_3 are the weights of ANNET and the error is defined as E where P_m is the measured pressure of the rubbertuator.

$$E = \frac{1}{2} \sum (P_d - P_m)^2 \tag{4.17}$$

Thus, ANNET is aimed to learn the transfer function of the rubbertuator-SVO pair that is modeled with a first order delay [3]. The mean square error of the system with a very good trained (epoch size of 40000 and 3 neurons) neural network is 0.0506.

In order to improve the ANNET performance described in [3], a simple second order delay function is used such as:

$$P_m = AI_c(1 - e^{(-t/T)} - te^{(-t/T)}) \tag{4.18}$$

The delay function will be:

$$I_c(t) = \frac{T^2}{A} \frac{d^2 P_d(t)}{dt^2} + \frac{T}{A} \frac{dP_d(t)}{dt} + \frac{P_d(t)}{A} \tag{4.19}$$

The simple form of the neural network will be:

$$net_o = w_1 \frac{d^2 P_d(t)}{dt^2} + w_2 \frac{dP_d(t)}{dt} + w_3 P_d(t) + w_4 \tag{4.20}$$

Thus, ANNET is aimed to learn the transfer function of the rubbertuator-SVO pair that is modeled with a second order delay. The mean square error of the system with a very good trained (epoch size of 40000 and 3 neurons) neural network is 0.0136 which provides a better result than the first order delay system.

4.4. The PONNET

To construct the position control neural network (PONNET) based on the physical model as proposed, we need to analyze the Language-Euler formulation of the BHRA. However, none of the dynamics parameters involved in a Lagrange-Euler expression will be computed or solved for. These expressions will be used to choose the correct input vectors and to define the correct neural network architecture of the PONNET layer that will supply a detailed approach instead of a "black box" approach.

The Lagrange dynamic formulation provides a means of deriving the equations of the motion from a scalar function called the "Lagrangian", which is defined as the difference between the kinetic and potential energy of a mechanical system.Lagrangian difference is formulated as:

$$Ł = K - P \tag{4.21}$$

where K and P are respectively the total kinetic energy and the total potential energy of the system [13].

The Lagrange's equations of motion are obtained from Equation 4.22

$$\frac{d}{dt}\frac{\partial L}{\partial \dot{q}} - \frac{\partial L}{\partial q} = \tau_i \tag{4.22}$$

where q is an n-vector of generalized coordinates q_i, τ is an n-vector of generalized forces τ_i.

$$q = [q_1 \dots q_i], \tau = [\tau_1 \dots \tau_i]^T \tag{4.23}$$

The torques acting on the end-effector are calculated by using Equation 4.23 and the equations found in [13]. Only the gravity-related terms are different from the equations in [13], because of the different structures of the two robots; the base of the BHRA is at the top, shown in Figure 3.2 while the robot's base is at the bottom in [13]. So the

result is:

$$\begin{aligned}\tau_3 = \quad & m_3\left(\frac{1}{3}L_3^2+\frac{1}{2}L_3L_2C_4+\frac{1}{2}L_1L_3C_{34}\right)\ddot{\theta_2}+m_3\left(\frac{1}{3}L_3^2+\frac{1}{2}L_2L_3C_4\right)\ddot{\theta_3} \\ & +\frac{1}{3}m_3L_3^2\ddot{\theta_4}+\left(\frac{1}{2}m_3L_2L_3S_4+\frac{1}{2}m_3L_1L_3S_34\right)\dot{\theta_2^2}+(m_3L_2L_3S_4)\dot{\theta_2}\dot{\theta_3} \\ & +\left(\frac{1}{2}m_3L_2L_3S_4\right)\dot{\theta_3^2}+\frac{1}{2}m_3gL_3S_{123} \end{aligned} \tag{4.24}$$

$$\tau_4 = I\ddot{\theta_5} \tag{4.25}$$

where, $C_x = Cos\theta_x$, $S_x = Sin\theta_x$, $C_{xyz} = Cos(\theta_x+\theta_y+\theta_z)$ and $S_{xyz} = Sin(\theta_x+\theta_y+\theta_z)$, m_3 is the point mass of the end effector, and L_1, L_2, L_3 are the length of the links, g for the gravity, θ_4 and θ_5 the approach and orientation angles of the end-effector with respect to the fixed robot coordinate frame. The above equations are different from [3] because the links' movement is horizontal in [3] where, in our system the trajectories are at the vertical direction against gravity.

The frustrating task of finding the dynamics parameters, such as the effective point masses and effective link lengths involved in equation above is left to PONNET neural network through training on sample trajectories. Luckily, these parameters are trajectory independent and once related with the neural network weights properly PONNET is likely to be able to generalize from a representative set of training trajectories [3].

In order to relate these torques with the rubbertuator torques acted on the end-effector, the rubbertuator pressure for the two joints can be summarized the same as in [3].

$$P_i = P_{oi} \pm \left[\frac{\tau_i}{2r(\beta_i-\alpha_i\varepsilon_i)}-\frac{\gamma_i}{(\beta_i-\alpha_i\varepsilon_i)}\right] \tag{4.26}$$

where β,γ and α are rubbertuator specific constants that we also do not wish to compute. Here, i is the rubbertuator index, P_o the equilibrium pressure, ε is the elongation

and " $\pm$ " is positive for an agonist and negative for an antagonist rubbertuator. For the time being if we assume the joint is fixed in position making ε a constant, we can separate the trajectory variables. It is a paradox to assume ε a constant for a robot that is designed for motion, but the paradox is resolved by dividing the motion space of each rubbertuator into several small segments as depicted in Figure 4.4 and assuming ε constant only for that small segment, only. With this assumption, combining Equations 4.24, 4.25, 4.26 a weighted sum of the non-linear functions of trajectory variables can express any of the four rubbertuator pressure as:

$$\begin{aligned} Net_o = \quad & w_1\ddot{\theta}_2 + w_2C_4\ddot{\theta}_2 + w_3C_{34}\ddot{\theta}_2 + w_4\ddot{\theta}_3 + w_5C_4\ddot{\theta}_3 + w_6\ddot{\theta}_4 + w_7S_4\dot{\theta}_2^{\,2} \\ & w_8S_{34}\dot{\theta}_2^{\,2} + w_9S_4\dot{\theta}_2\dot{\theta}_3 + w_{10}S_4\dot{\theta}_3^{\,2} + w_{11}S_{234} + w_{12}\ddot{\theta}_5 \end{aligned} \tag{4.27}$$

Here, the parameters, w are functions of pulley radius, r,initial pressure P_o joint position ε, the rubbertuator parameters (β, γ and α), the robot parameters (m_3, L_1, L_2, L_3 and gravity, g). Furthermore, the rubbertuator pressure is always positive and bounded thus enabling us to use the sigmoid function for error propagation. Therefore we can assign a single layer backpropagation neural network for each rubbertuator using Equation 4.27, which we call PONNET [3].

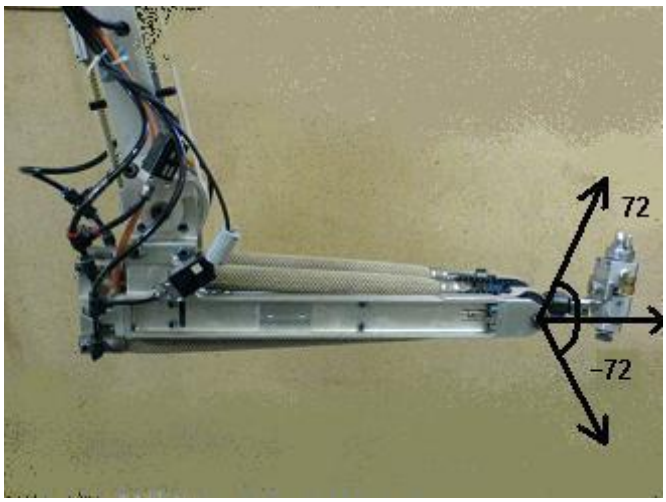

Figure 4.3. The reach angle in the workspace of the end-effector

As mentioned above and in [3], taking small segments decreases the nonlinearity of the ε effect on the system. So for every 4°, a neural network is used for each

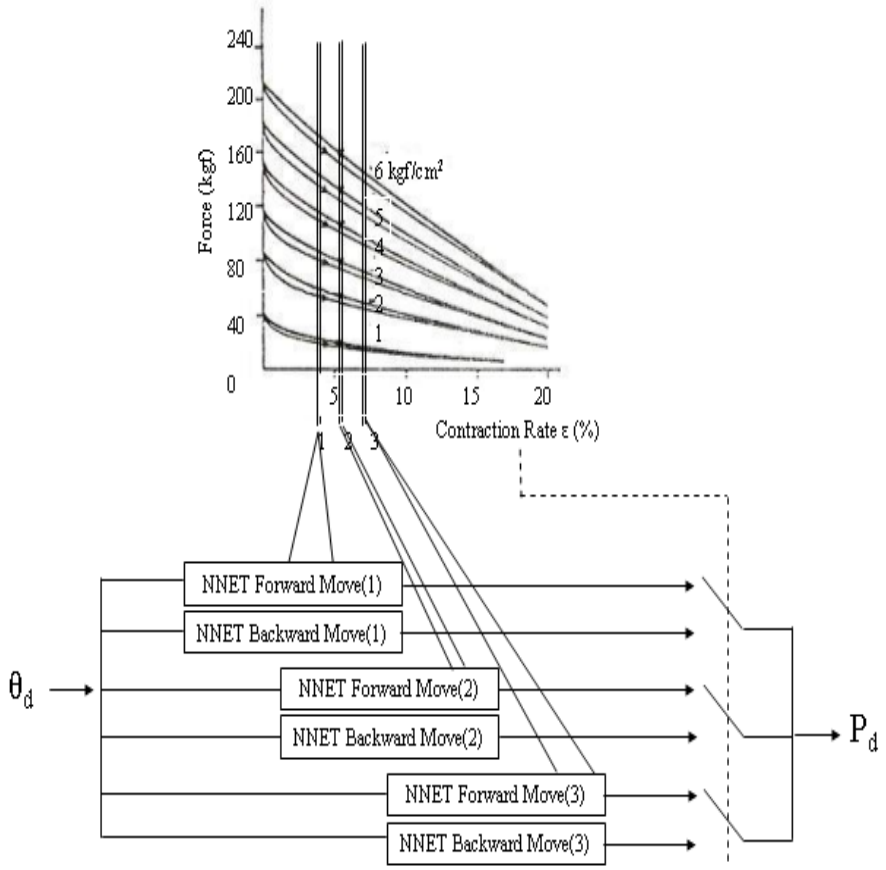

Figure 4.4. Division of the workspace of the end-effector into small segments [7]

rubbertuator. Considering the maximum and minimum reach limits of the workspace as $-72°$ and $72°$, for each rubbertuator 36 small neural networks are trained (Totally 4x36=144 PONNET). As mentioned [1], [2], [3] and [4], the hysteretic characteristics of rubbertuators causes different torques when moving forward or backwards. In order to struggle this problem, we use two different neural network for the same angle segment, so this means that, we used 144x2 = 288 PONNETs for the control of this system. However, only one is active at a given time depending on the desired joint position (θ_d) and moving direction shown in Figure 4.4.

4.5. The Estimated Error Function of PONNET

The actual pressure values of the train data for the PONNET are now taken as the desired pressure, P_d. The error will be:

$$\tilde{E} = P_d - P_{sPONNET} \tag{4.28}$$

where $P_{sPONNET}$ is the simulated pressure output of the PONNET. Instead of finding the estimated error function for each 4° segments, all the segments are reduced to one $0^\circ - 4^\circ$ segments, which will show the general error characteristics of the error distribution, shown in Figure 4.5.

$$\theta_g = \theta_d - integer\ part\ of\ (\theta_d/4) \tag{4.29}$$

where θ_d is the desired angle and θ_g is the generalized angle.

$$\hat{E} = f(\tilde{E}(\theta_g)) \tag{4.30}$$

The estimated error function is constructed by fitting a second order polynomial curve to the error distribution, show in Figure 4.5.

There are two main results:

- In the construction of small neural networks, we assume the joint is fixed in position making ε a constant, in order to decrease the nonlinear terms, but this assumption will give good results in the middle of the segments but at the end segments, the nonlinear terms affect the results more than in the middle, so this causes bigger errors at the ends and smaller errors at the middle.
- The discontinuities between the consecutive neural networks increase the errors at the end and at the begin of the small segments.

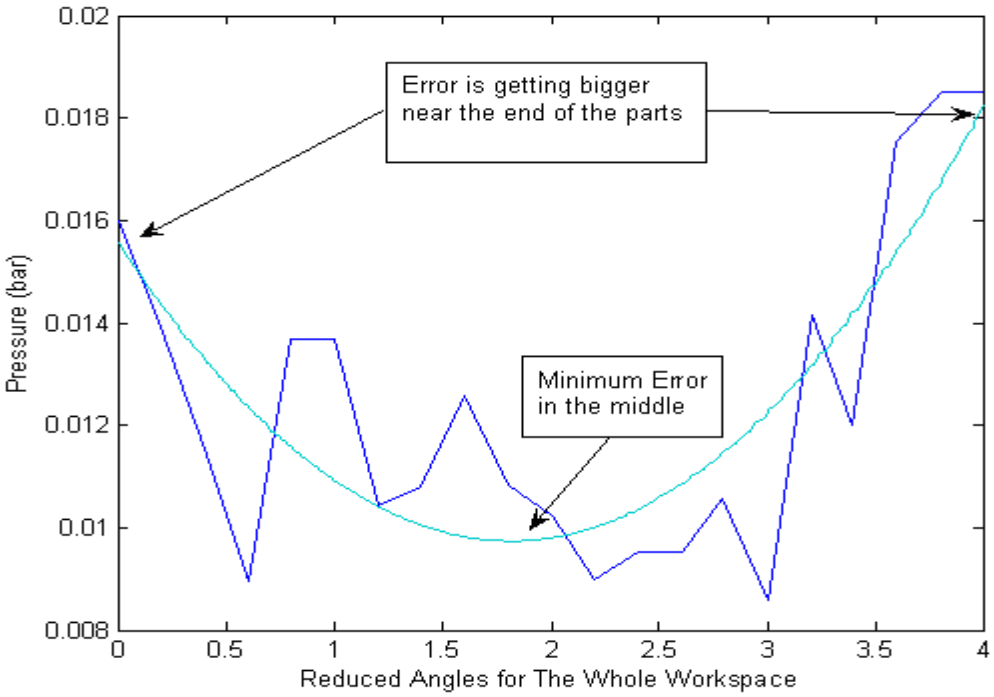

Figure 4.5. The Generalized Estimated Error Function

4.6. The Fuzzy-Neural Hybrid System Approach for Robot Arm Controlling

Fuzzy logic approach, that is firstly developed in 1965 by Zadeh [14], is an efficient way to map input spaces to output spaces, especially when the physical relationship between these spaces is too complex to be described by mathematical models [15]. The main idea of the proposed fuzzy algorithm in this study is to combine consecutive neural network pressure outputs and model the elongation nonlinearity to give a final pressure value for controlling the robot arm. There are basically two linguistic variables: estimated error, $\hat{E}$, (unit in pressure) and convex combination parameter, α. The antecedent variable, estimated error, is obtained from the error distribution of artificial neural network system outputs. Convex combination parameter, α, is the consequent linguistic variable of the fuzzy algorithm. Both the error and combination parameter have five normal type triangular membership functions that are namely very low, low, medium and high, very high. The membership functions are basically specified with two parameters, medium of symmetric triangular, x_i, and the width of its sides, μ_i,

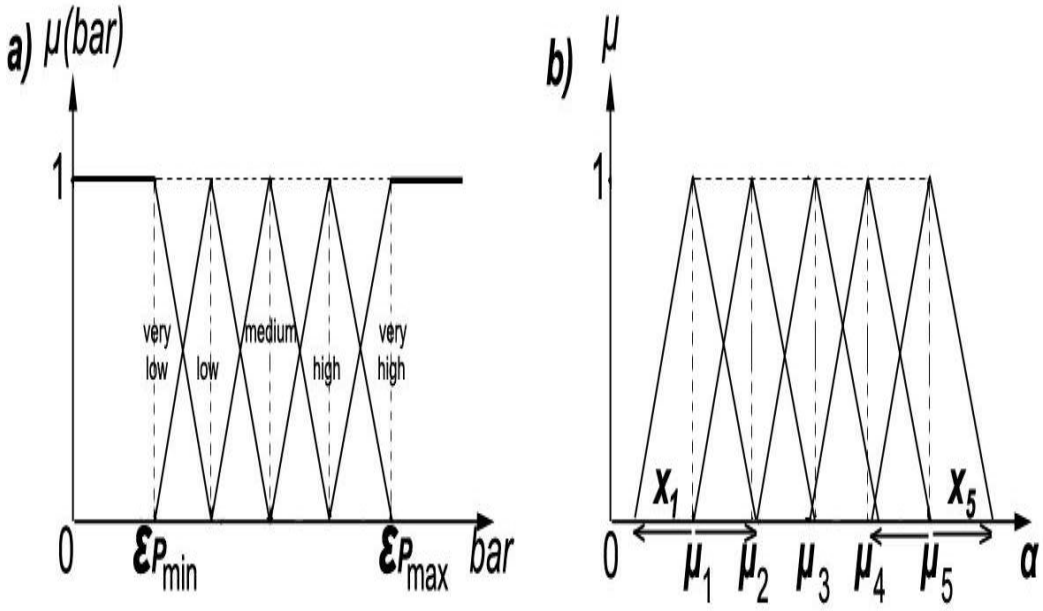

Figure 4.6. The fuzzy method linguistic variables a)error b)convex combination parameter

where i=1...5, shown in Figure 4.6.

The final pressure is calculated as a convex combination of two consecutive neural network pressure outputs as follows:

$$P_f = \alpha P_{N1} + (1 - \alpha) P_{N2} \tag{4.31}$$

where P_f is the final pressure, P_{N1} and P_{N2} are consecutive neural network pressure outputs.

All the possible five rules are used in the rule base. The rules are adjusted such that minimum error is manipulated in a way that the pressure output of the PONNET is used with maximum contribution to the final pressure output, i.e. alpha becomes 1 and when the error is maximum, the contribution of the consecutive neural network outputs are forced to be equal, this means alpha becomes 0.5.

Since there is only one variable in the system there is no need to use a t-norm in the inference. The fuzzy membership function value of the antecedent is directly becomes the fuzzy number of the consequent variable. Center of gravity is used as defuzzification method in the fuzzy algorithm.

4.7. The GA Optimization

On the previous section, the fuzzy membership function variables are chosen by experience. In order to find a more appropriate membership function variables, a basic GA optimization is used because it can be applied to solve a variety of optimization problems that are not well suited for standard optimization algorithms, including problems in which the objective function is discontinuous, nondifferentiable, stochastic, or highly nonlinear [16].

The objective or the performance criteria is to minimize the root mean squared error of the desired pressures and the pressure output of the fuzzy-neural hybrid system. The train data for PONNET is also used for GA optimization. As mentioned before the main goal is to find an off-line control structure, so a set of train data is used during the construction of the control system.

The inputs to the GA are the medium and the width of the membership functions shown in figure 4.6 and given in 4.32, thus for each of five membership functions, there are two variables (the width of the membership functions) and consequently the number of variables for GA is 10.

$$\begin{aligned} &minimize\ Root\ Mean\ Square\ Error \\ &Optimization\ parameters = \{x_1, x_2, x_3, x_4, x_5, \mu_1, \mu_2, \mu_3, \mu_4, \mu_5\} \end{aligned} \tag{4.32}$$

The genetic algorithm optimization toolbox uses the root mean square error as the fitness function, which is written by us that simulates the output of the fuzzy-neural hybrid system and then, calculates the root mean square error between the output and the actual pressure values taken from the train data by using uses optimization

parameters $x_1, x_2, x_3, x_4, x_5, \mu_1, \mu_2, \mu_3, \mu_4, \mu_5$. The population size is 20, the crossover fraction is 0.8, the migration fraction is 0.2. These values are the default values used in GA toolbox in matlab and changing these values will not affect the result very much.

5. EXPERIMENTAL RESULTS

The goal of trajectory planning is to generate the reference inputs to the motion control system, which ensures that the manipulator executes the planned trajectory. The user typically specifies a number of parameters to describe the desired trajectory. Planning consists of generating a time sequence of values obtained by polynomial function interpolating the desired trajectory [7]. In the study, the trajectories, which are desired from robot to follow, are the sine wave, ramp and step paths. The user specified parameters for the sine wave path are the frequencies and the amplitudes of the sine wave path. The desired trajectories of the end-effector are accomplished by controlling each of the involved joints trajectories simultaneously.

The control objective is to keep the end-effector always perpendicular to the X-Y plane of the operational space, shown in Figure 1.1. It is assumed that the end-effector paints the desired trajectories on the X-Y plane, and it holds a pen, which has to be always perpendicular to the surface. There is a very important remark at that point; as the BHRA follows a trajectory in the operational space, the joint 4, in the joint space, follows the same trajectory in the opposite direction in order to keep the third link (end-effector) always perpendicular to the surface, shown in the figures of this chapter. For example, as the first and the second link moves up, the third link has to move down in order to guarantee the Equation 3.21

The trajectory tracking performance results of the two methods, the only-neural network control method cited in [3] and the fuzzy-neural hybrid control method, in the operational space and in the joint space, are shown in this chapter.

The green lines corresponds to the to the desired trajectory, the red lines stand for the only-neural network controller output and the blue lines stand for the fuzzy-neural hybrid controller output.The results of the outputs are shown in Table 7.1 and discussed in Chapter 7.

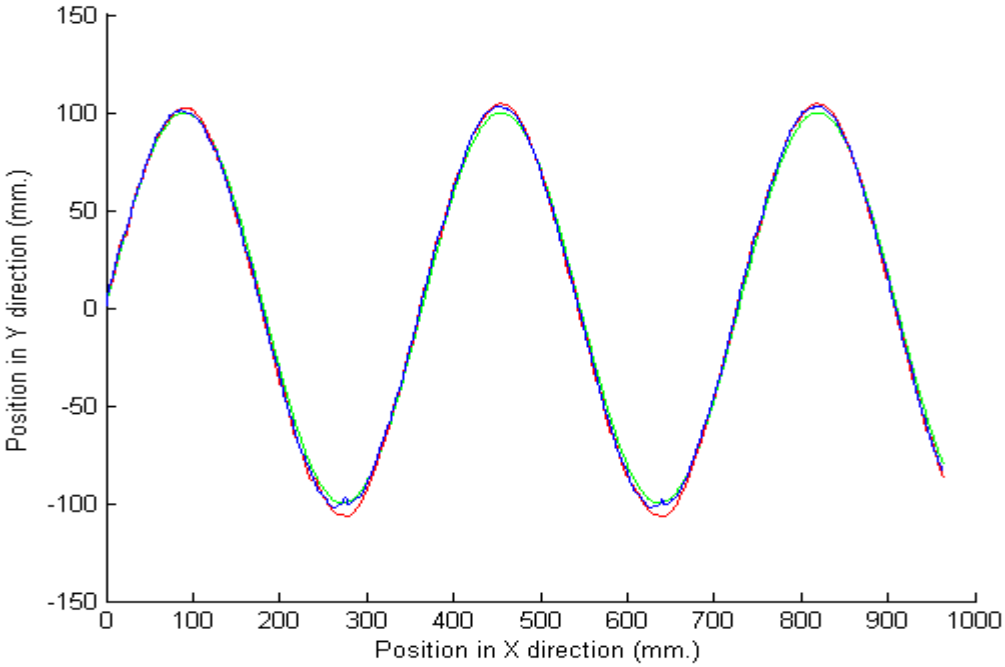

Figure 5.1. The sine wave trajectory with frequency = 0.1 s^{-1} and amplitude = 100mm. in the operational space

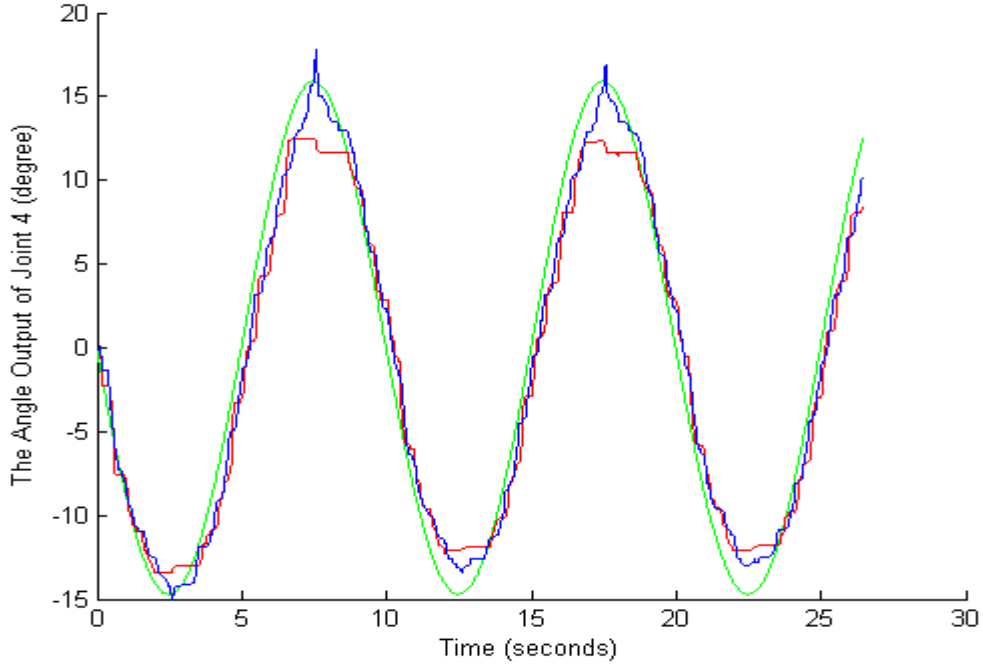

Figure 5.2. The sine wave trajectory with frequency = 0.1 s^{-1} of joint 4, in the joint space

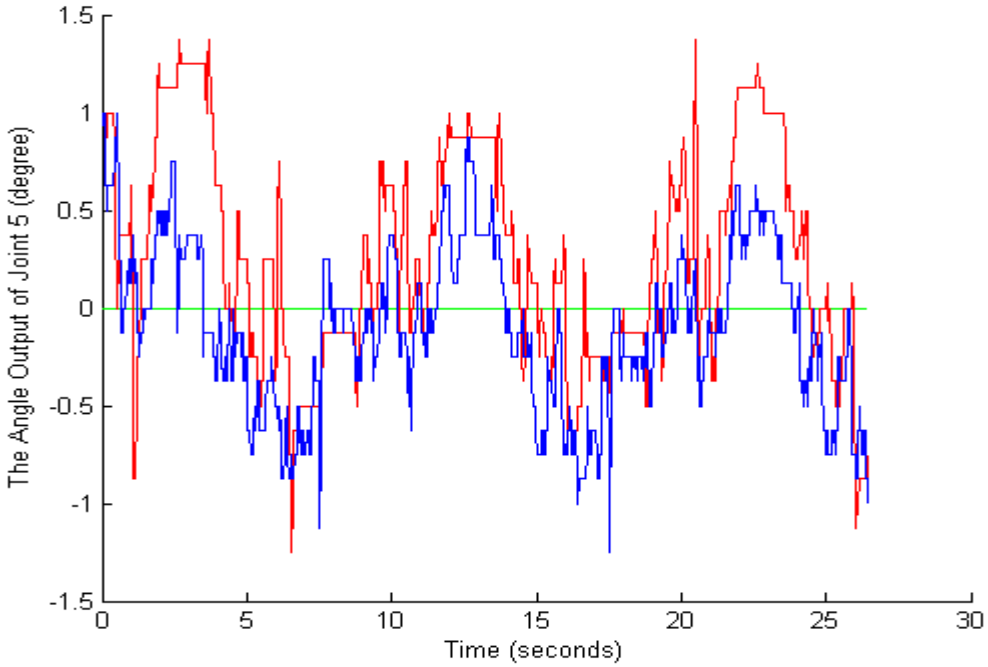

Figure 5.3. The sine wave trajectory with frequency = 0.1 s^{-1} of joint 5, in the joint space

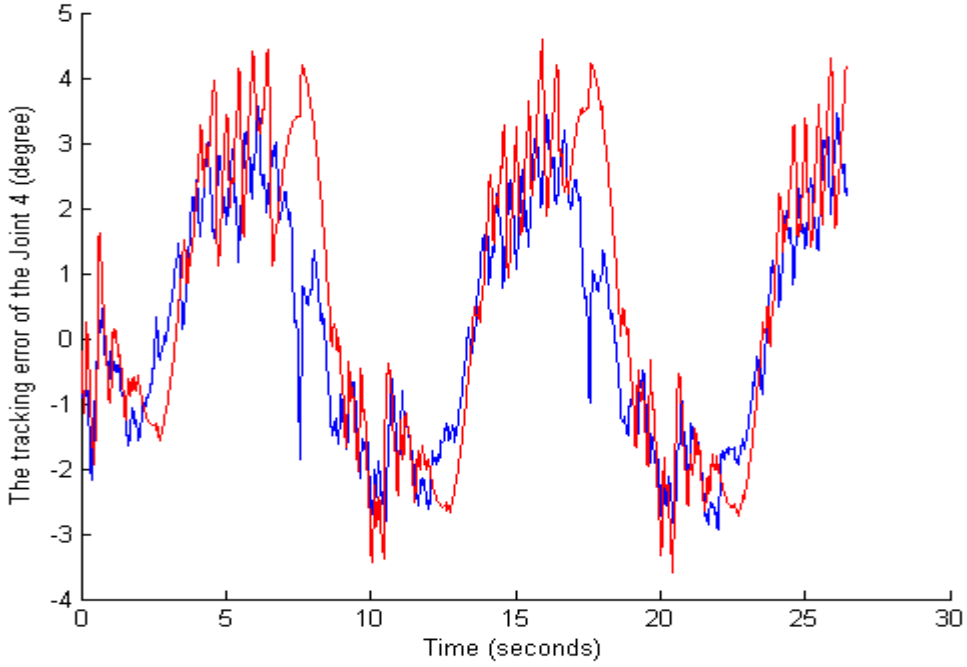

Figure 5.4. The tracking error of the sine wave trajectory with frequency = 0.1 s^{-1} of joint 4, in the joint space

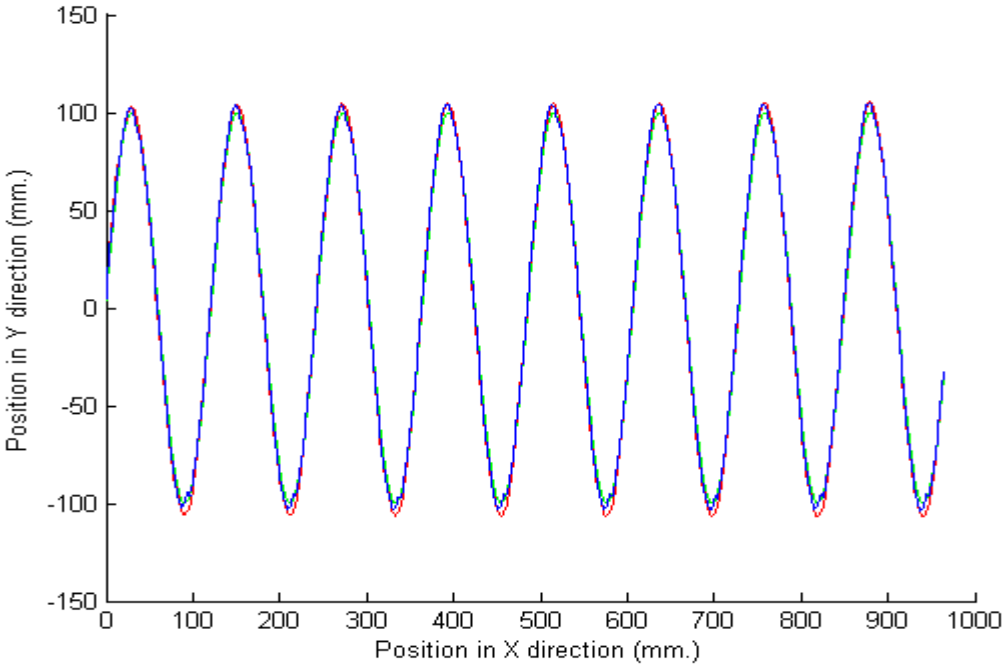

Figure 5.5. The sine wave trajectory with frequency = 0.3 s^{-1} and amplitude = 100mm. in the operational space

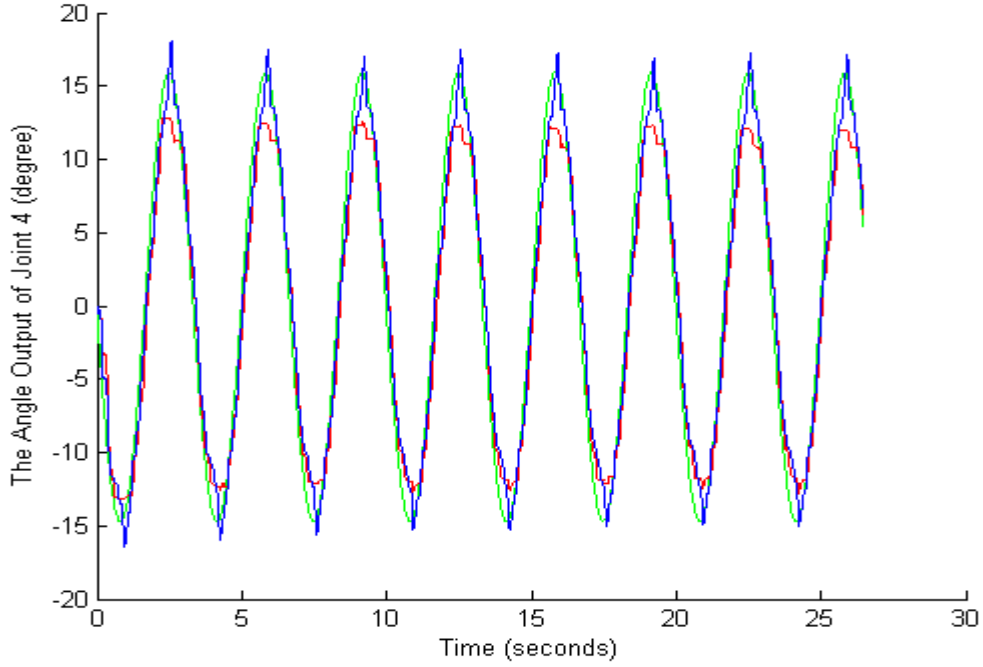

Figure 5.6. The sine wave trajectory with frequency = 0.3 s^{-1} of joint 4, in the joint space

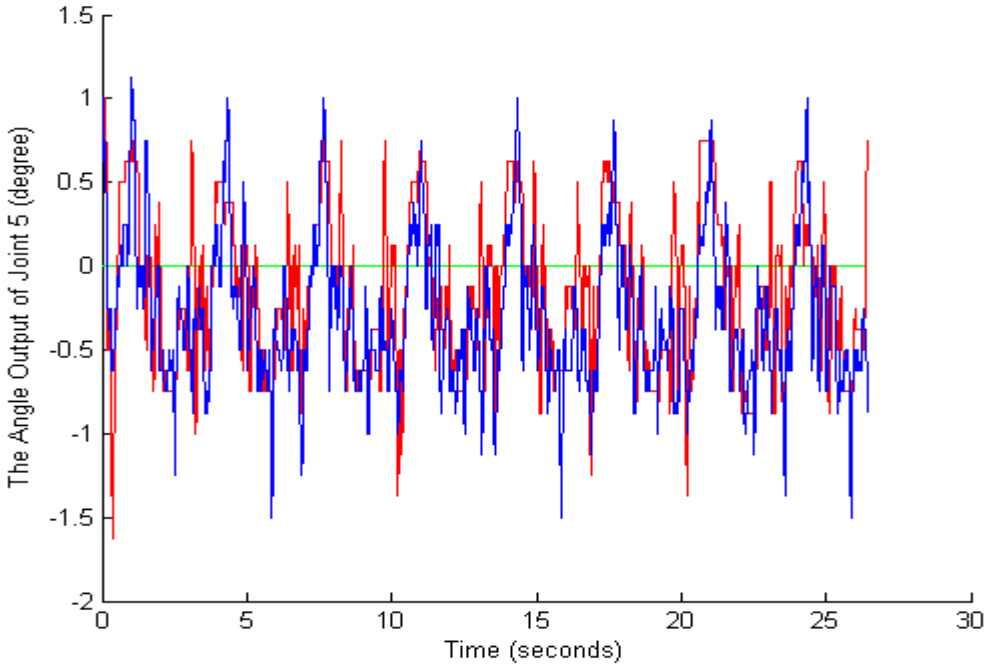

Figure 5.7. The sine wave trajectory with frequency = 0.3 s^{-1} of joint 5, in the joint space

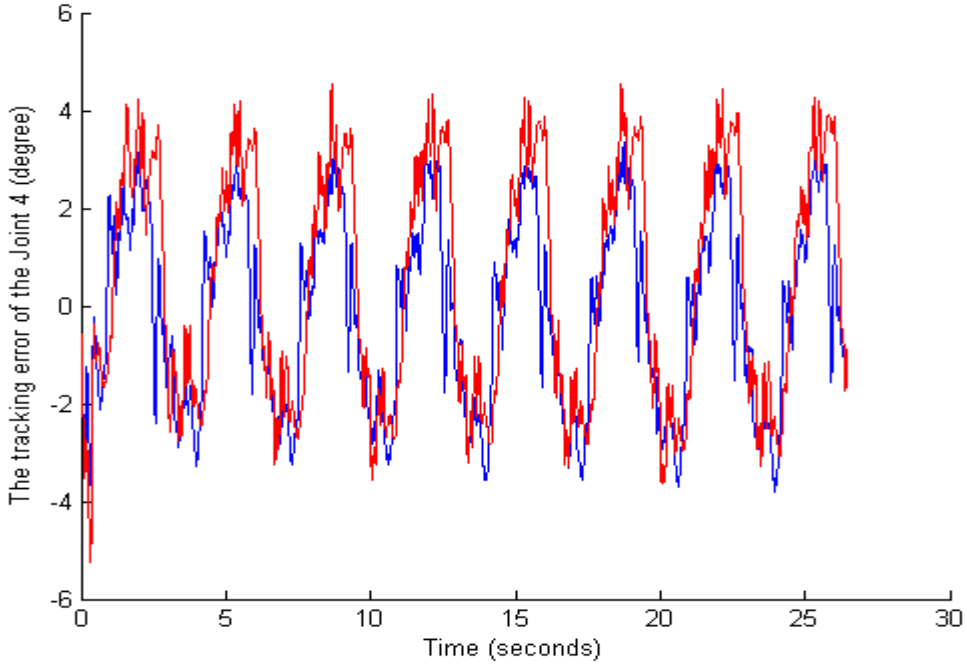

Figure 5.8. The tracking error of the sine wave trajectory with frequency = 0.3 s^{-1} of joint 4, in the joint space

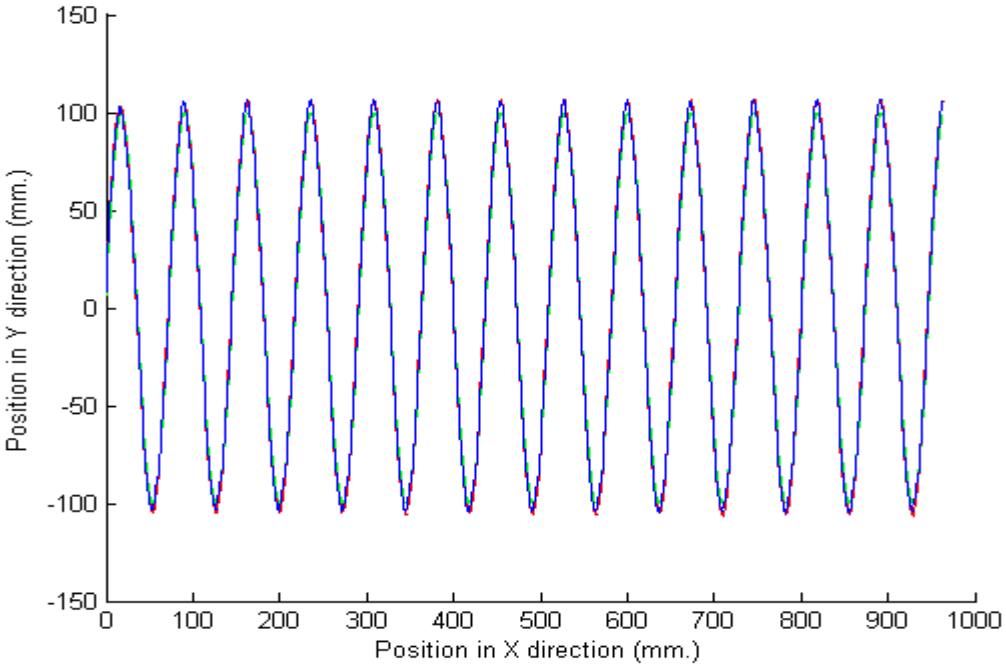

Figure 5.9. The sine wave trajectory with frequency = 5 s^{-1} and amplitude = 100mm. in the operational space

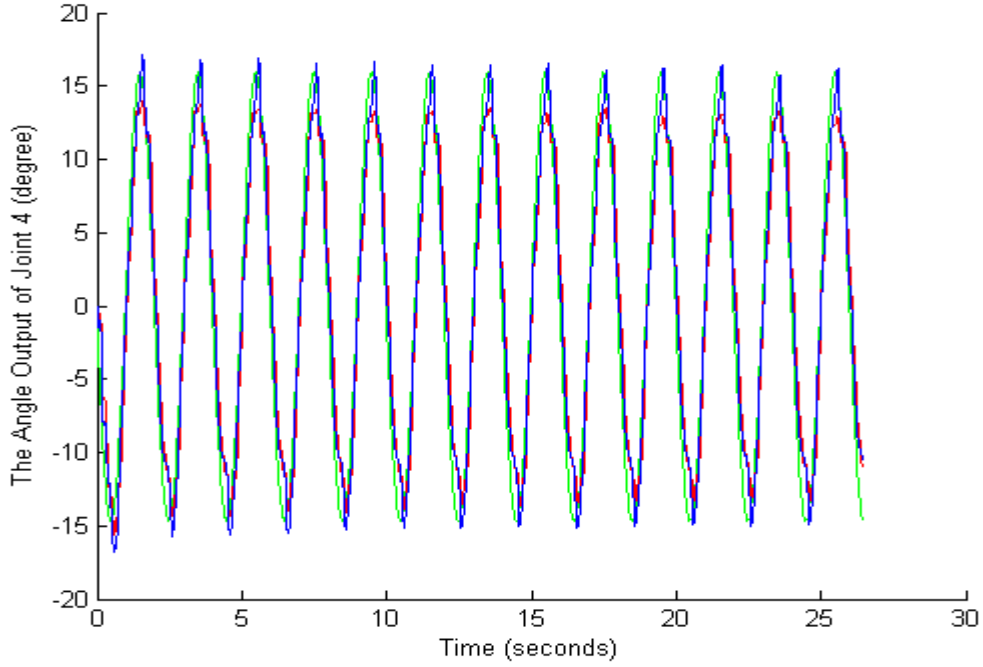

Figure 5.10. The sine wave trajectory with frequency = 0.5 s^{-1} of joint 4, in the joint space

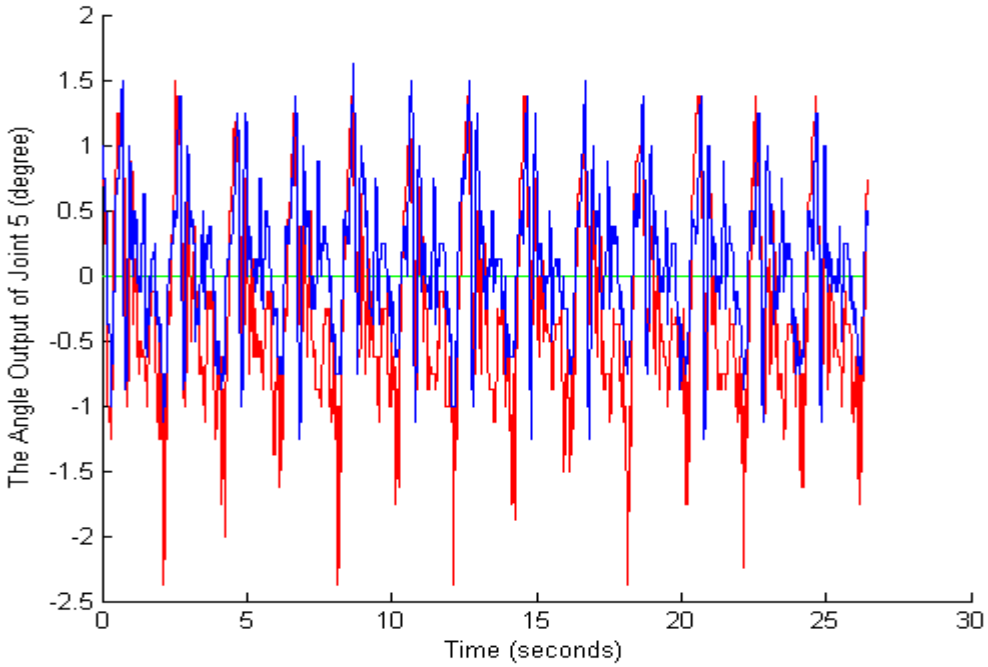

Figure 5.11. The sine wave trajectory with frequency = 0.5 s^{-1} of joint 5, in the joint space

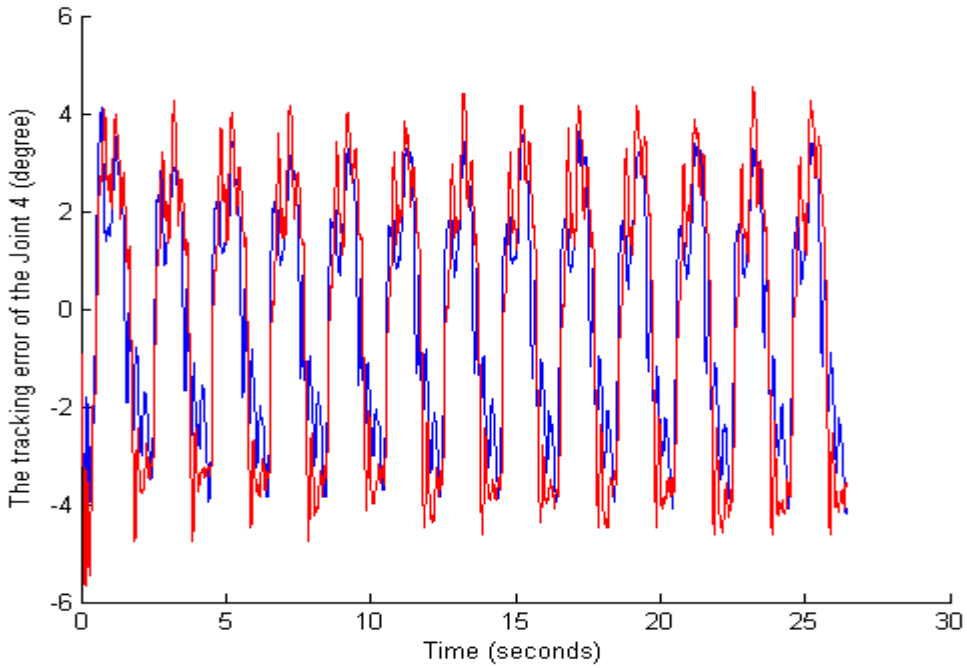

Figure 5.12. The tracking error of the sine wave trajectory with frequency = 0.5 s^{-1} of joint 4, in the joint space

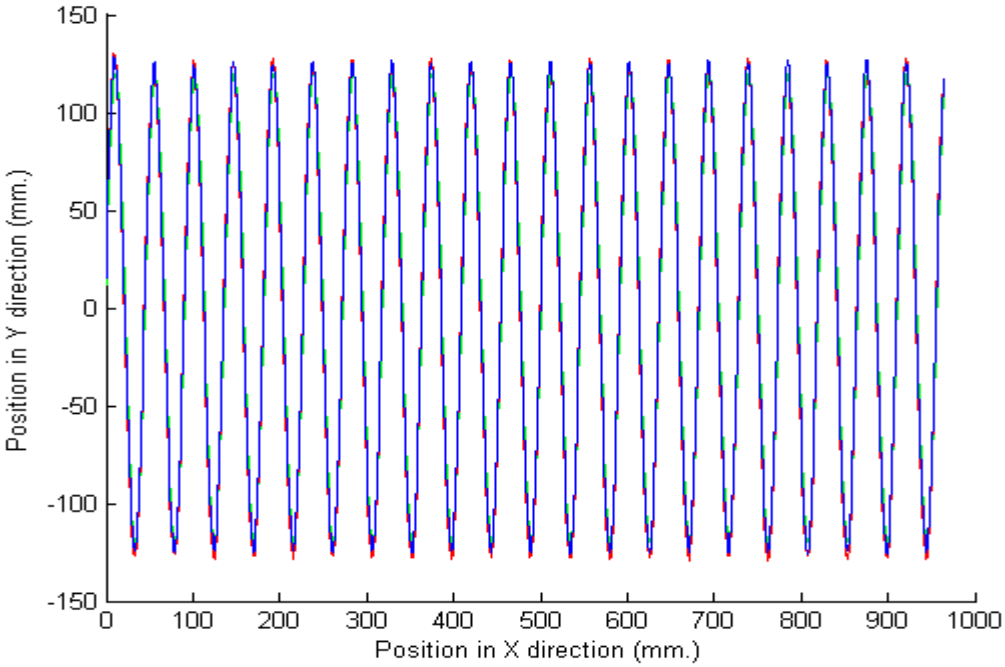

Figure 5.13. The sine wave trajectory with frequency = 0.8 s^{-1} and amplitude = 120mm. in the operational space

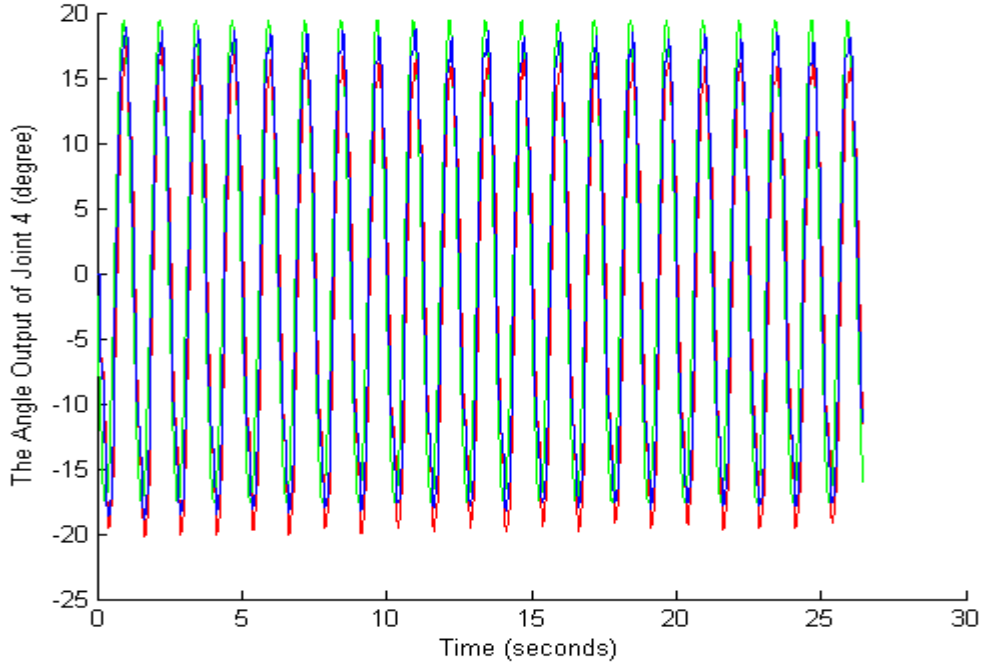

Figure 5.14. The sine wave trajectory with frequency = 0.8 s^{-1} of joint 4, in the joint space

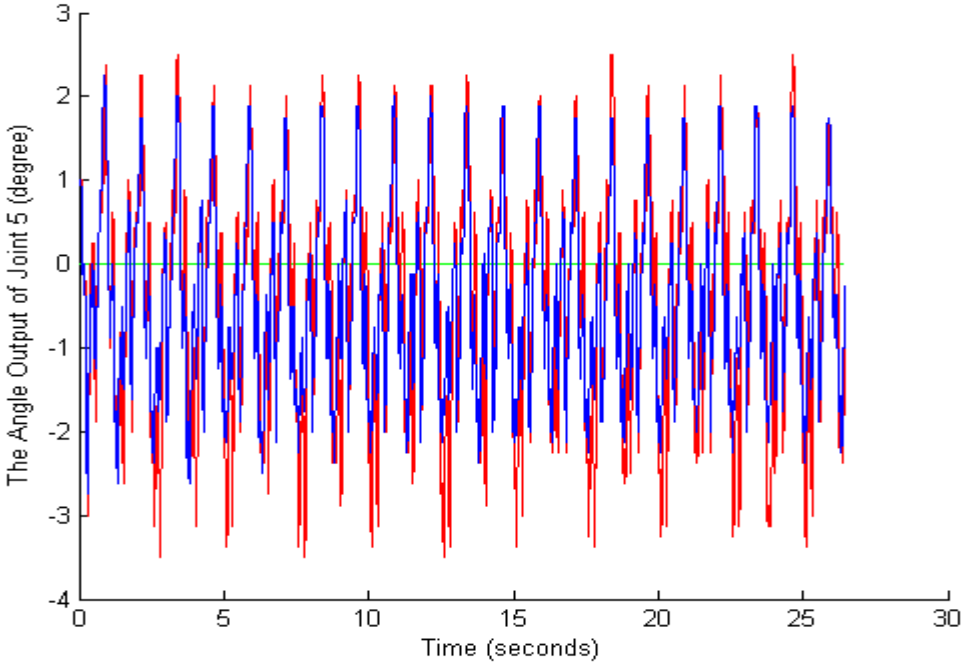

Figure 5.15. The sine wave trajectory with frequency = 0.8 s^{-1} of joint 5, in the joint space

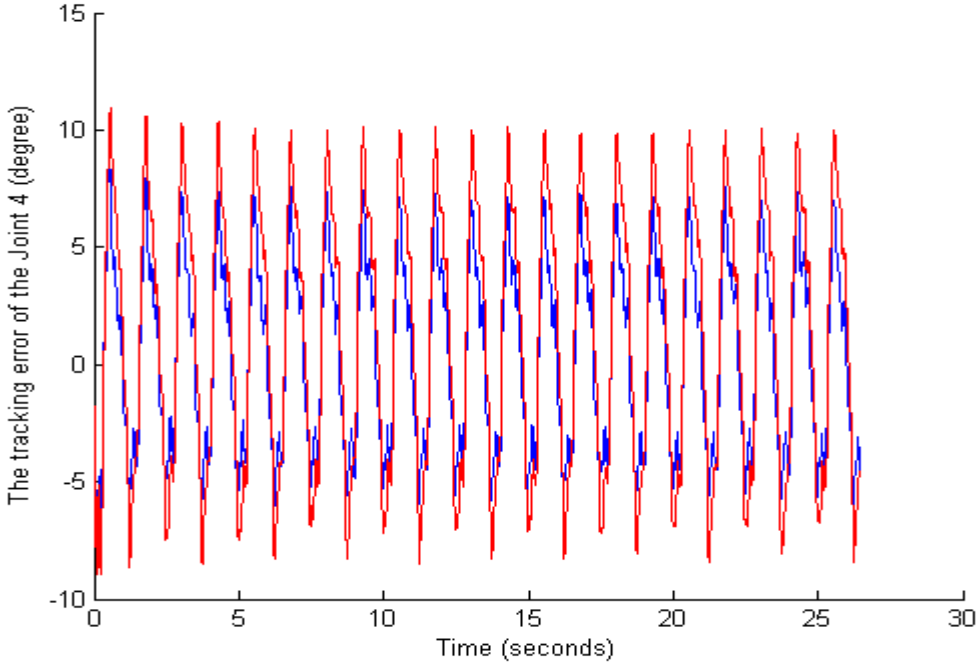

Figure 5.16. The tracking error of the sine wave trajectory with frequency = 0.8 s^{-1} of joint 4, in the joint space

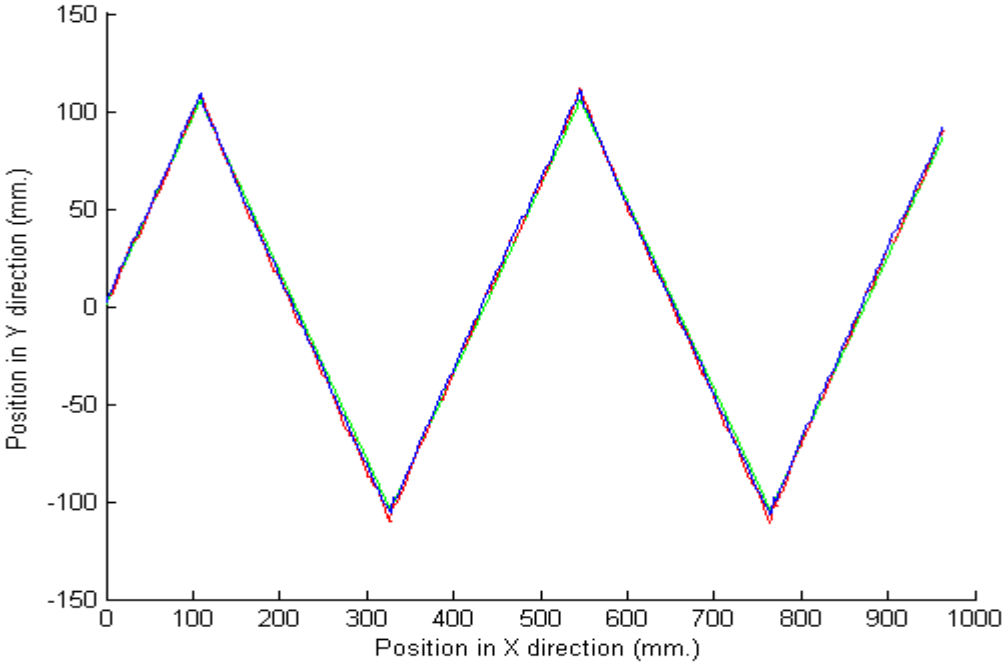

Figure 5.17. The ramp wave trajectory in the operational space

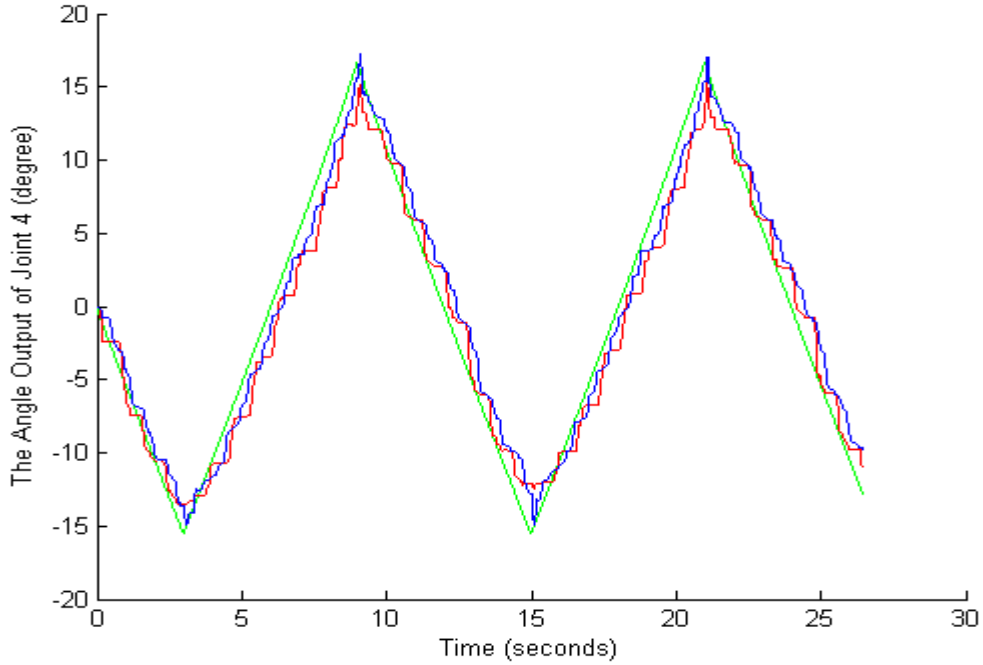

Figure 5.18. The ramp wave trajectory of joint 4, in the joint space

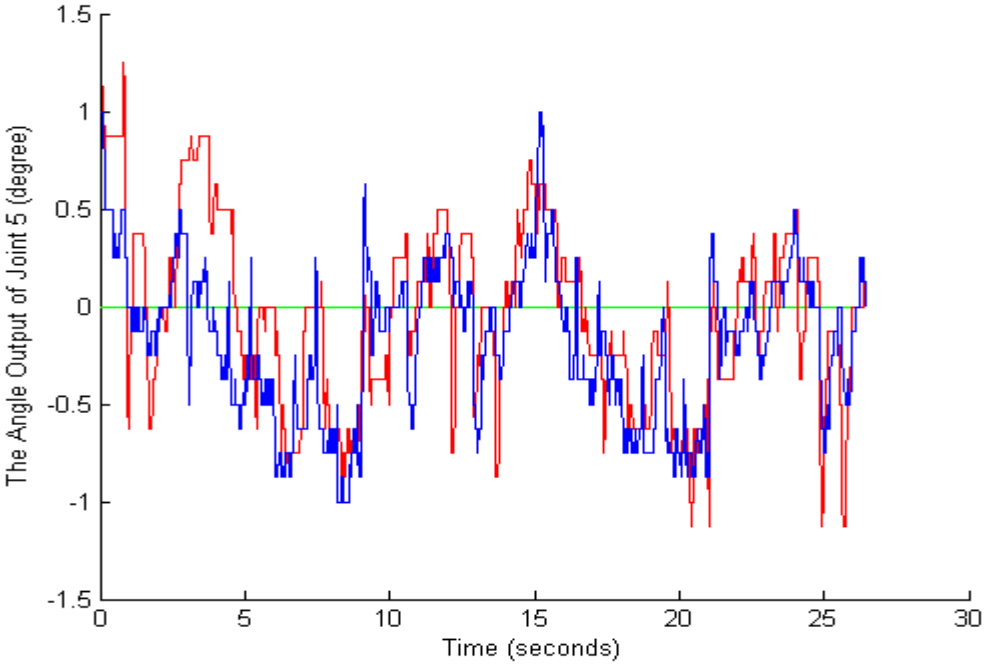

Figure 5.19. The ramp wave trajectory of joint 5, in the joint space

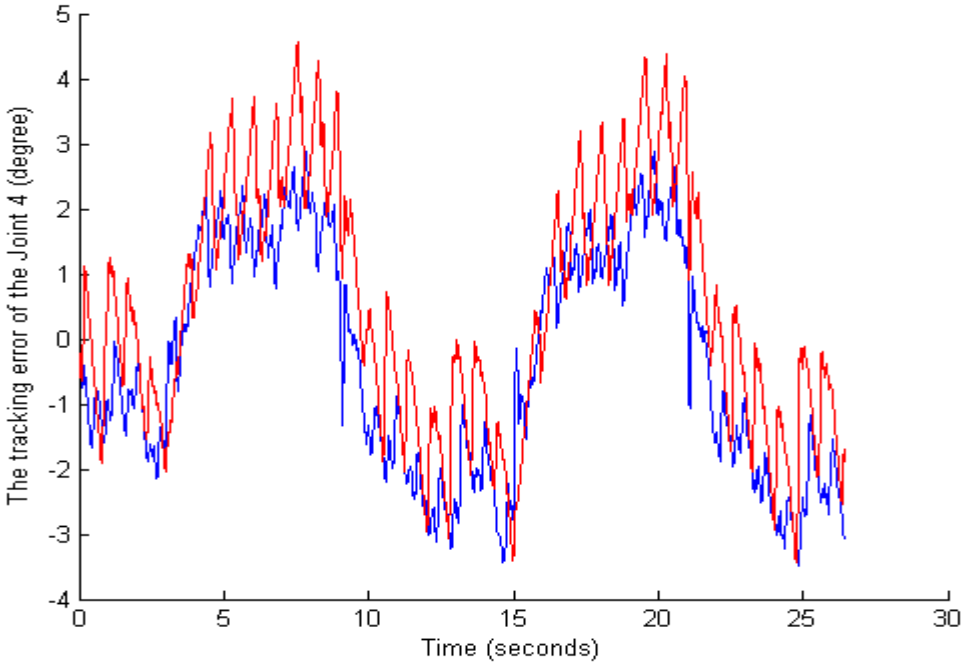

Figure 5.20. The tracking error of the ramp wave trajectory of joint 4, in the joint space

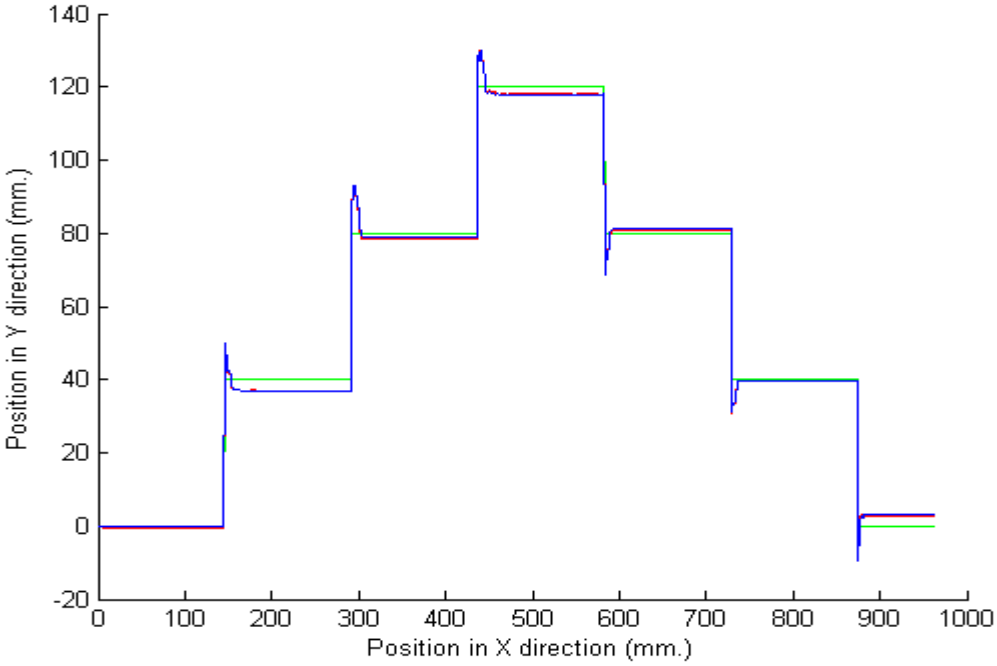

Figure 5.21. The step wave trajectory in the operational space

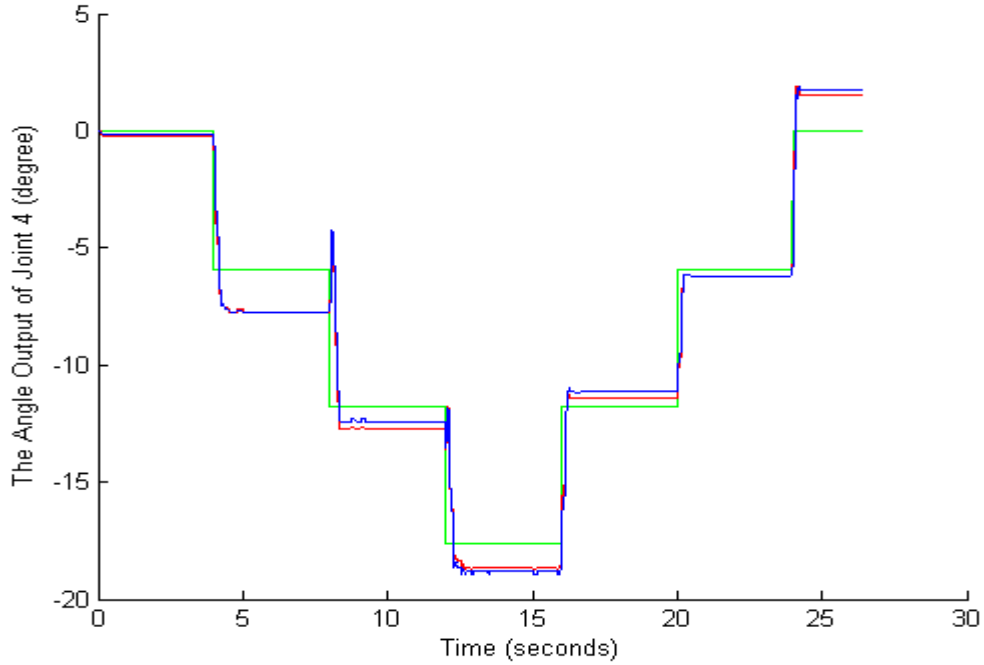

Figure 5.22. The step wave trajectory of joint 4, in the joint space

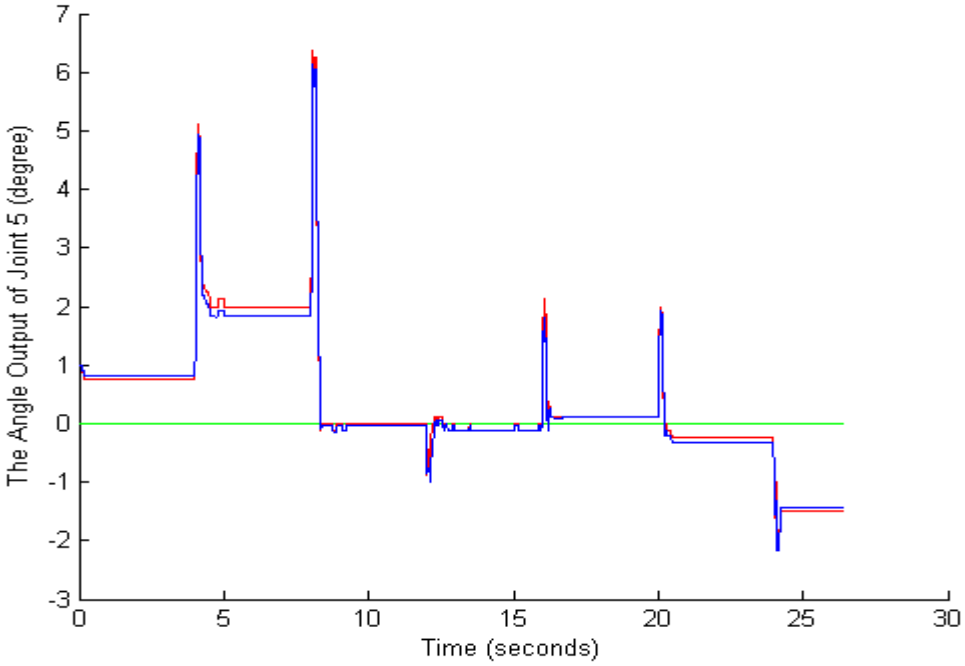

Figure 5.23. The step wave trajectory of joint 5, in the joint space

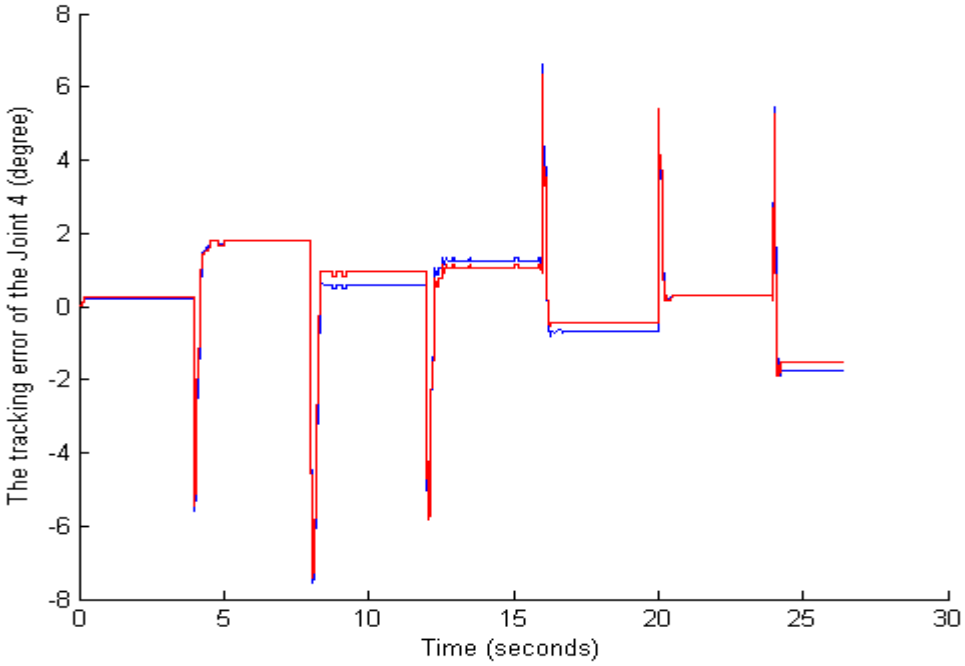

Figure 5.24. The tracking error of the step wave trajectory of joint 4, in the joint space

6. ANIMATION SOFTWARE AND USER'S MANUAL

In the pervious chapters, a detailed structure, kinematics and control methods have been developed for the BHRA. In this section, we will be presenting the visual animation software that has been created using the obtained model and user's manual. Animation has been created using OPENGL-software and the Visul C++ programming software. The graphical user interface of the animation is shown in Figure 6.1

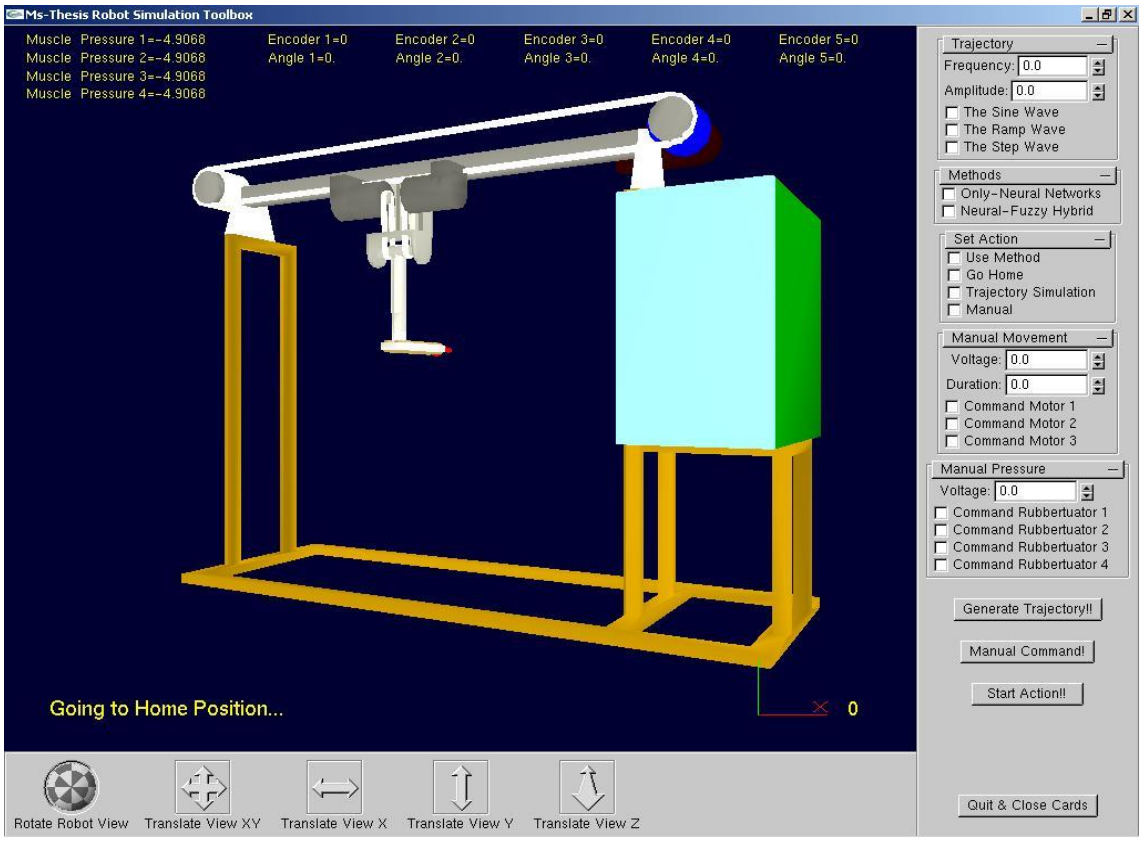

Figure 6.1. The graphical user interface of the animation software

6.1. User's Manual

In order to start the animation software, RobotConsole.exe file has to executed. A Opengl-Glui window will appear in the screen. At the same time the control cards will be initialized.

From the "Manual Movement Panel" and the "Manual Pressure Panel", the volt-

age references can be given directly to the motors and the rubbertuators. To start the command, "Manual Command Button" has to be pressed.

In order to make the BHRA track a trajectory, first a trajectory has to be generated from the "Trajectory Panel" by entering desired frequency and amplitude values. A sine wave, a step wave and a ramp wave can be selected from the panel. Later, the "Generate Trajectory Button" has to pressed. Later from the methods panel, the controller method can be set by just clicking on the checkbox near the methods. At the end, pressing the "Start Action Button" will start the motion of the BHRA.

7. CONCLUSIONS

7.1. Conclusions and Discussions

The end effector angle outputs of the fuzzy-neural hybrid control system, summarized in Figure 7.1 and the only neural control system, defined in [3], are compared in different sinusoidal, step and ramp trajectories. The root mean square of the errors (RMSE), using Equation 7.1, of the end-effector's desired trajectory angle, θ_4 and the two systems output angles, are calculated for both systems and shown on table 7.1.

$$RMSE = \frac{\sqrt{\sum_{i=1}^{n} e_i^2}}{n} \tag{7.1}$$

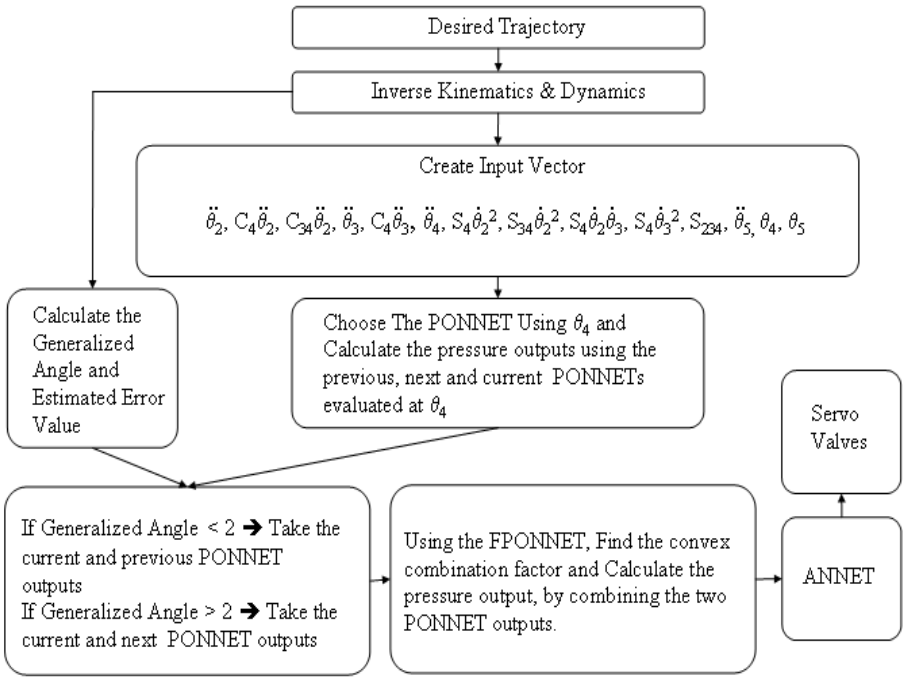

Figure 7.1. The fuzzy-neural hybrid control system

In conclusion, we improve the system defined in [3] by using a second order delay function and constructing a fuzzy-neural hybrid control system. The results on table 7.1, clearly demonstrates that the new controller system supplies a better performance in trajectory tracking of the end effector for continuous trajectories and higher speeds.

Table 7.1. The RMSE Performance Analysis of Both Systems on Various Trajectories

Trajectory	Only-Neural Network	Fuzzy-Neural Hybrid	Change(%)
Sine, f=$0.1s^{-1}$	2.2836	1.7699	22.50%
Sine, f=$0.3s^{-1}$	2.4624	1.9286	21.68%
Sine, f=$0.5s^{-1}$	2.9403	2.2518	23.42%
Sine, f=$0.8s^{-1}$	5.7398	4.0077	30.18%
Ramp	1.9413	1.7791	8.36%
Step	2.3789	2.4221	-1.81%

The RMSEs of both systems are nearly the same for the step outputs, because the fuzzy system is constructed only for the discontinuities between consecutive small neural networks, not for the high discontinuities in the desired trajectories. It is a desirable result which shows that the fuzzy only maintains continuity and does not change the characteristics of the desired trajectory.

7.2. Future Improvements

The PONNETs have been trained by using continuous train data trajectories that span all the workspace and the reaching limits of the end-effector, because of that, both only-neural and fuzzy-neural hybrid control system can not achieve the step satisfactorily. In the future, using an online control system or adopting online training ability to the neural networks, can overcome these problems.As cited in [12] and mentioned before, finding a better off-line control ensures that small learning rates will be sufficient for future on-line training control, so using small learning rates can easily overcome the step response performance problem.

Using two fuzzy control system instead of one will improve the performance. In this study, one fuzzy system is used to overcome the discontinuity and the backward-forward direction change error (at the peaks of the trajectories, system changes its motion direction, from forward to backward or from backward to forward), using two different fuzzy systems for these two separate problems will increase the performance.

REFERENCES

1. Pandian S. R., S. Kawamura, F. Takemura and Y. Hayakawa, "Control Performance of an Air Motor", *IEEE International Conference on Robotics Automation*, Vol.1, pp.518-524, 1999.

2. Hashimoto K., M. Ewaeda and A. Ishii, "Trajectory control of a one-link arm using pneumatic rubbertuator", *SICE '96. Proceedings of the 35th SICE Annual Conference. International Session Papers*, July 24-26, pp.1163-1166, 1996.

3. Ozkan M., K. Inoue, K. Negishia and T. Yamanaka, "Defining a Neural Network Controller Structure for a Rubbertuator Robot", *Neural Networks*, Vol.13, pp.533-544, 2000.

4. Izawa J., T. Kondo and K. Ito, "Biological Arm Motion Through Reinforcement Learning", *Robotics and Automation, 2002. Proceedings. ICRA '02. IEEE International Conference*, Vol.4, pp.3398- 3403, 2002.

5. Chin-Teng Lin and C.S. George Lee, *Neural fuzzy systems : a neuro-fuzzy synergism to intelligent systems*, Prentice Hall, NJ, 1996.

6. Ozkan M., *Neural Network Controller for APRS Rubbertuator Joints Bridgestone*, BridegeStone Corporation, Manual, 1994.

7. Sciavicco L. and B.Siciliano, *Modeling and Control of Robot Manipulators*, Springer, New York, 2000.

8. Denavit J. and R. S. Hartenberg, "A kinematic notation for lower-pair mechanisms based on matrices", *ASME Journal of Applied Mechanics*, Vol.4, pp: 215-221, 1955.

9. Lakhmi C. Jain and N.M. Martin, *Fusion of Neural Networks, Fuzzy Systems and Genetic Algorithms: Industrial Applications*, CRC Press, 1996.

10. Rogers Joey, *Object-Oriented Neural Networks in C++*, Academic Press, 1997.

11. Polat Ali, *Design of a Robot Control Experiment Setup for a 2-Degree of Freedom Rubbertuator Robot*, M.S. Thesis, Boğaziçi University, 2005.

12. Fausett L., *Fundamentals of Neural Networks*, Prentice Hall, 1994

13. Bulut Hasip, *Control and Animation of a Three-Link Robotic Arm Using Direct Kinematics and Inverse Kinematics, Boğaziçi University*, 2002.

14. L.A. Zadeh, "Fuzzy sets", *Information and Control*, Vol.8, pp.338-353, 1965.

15. Tsourveloudis. N. C. and Y. A. Phillis, "Fuzzy Assessment of Machine Flexibility", *IEEE Transactions on Engineering Management*, Vol.45, pp.78-87, 1998.

16. Matlab, *GA toolbox* , Version 7.0, Mathworks, 2005.