MODELING AND CONTROL OF 5DOF ROBOTIC ARM USING NEURO-FUZZY CONTROLLER

by

ABDUL WAHAB NADEEM BUTT
NUST201260407MSMME62112F

A thesis submitted to the
Department of Robotics and Intelligent Machine Engineering
in partial fulfillment of the requirements for the degree of

MASTER OF SCIENCE
IN
ROBOTICS AND INTELLIGENT MACHINE ENGINEERING

Thesis Supervisor
DR. MOHSIN JAMIL

School of Mechanical and Manufacturing Engineering
National University of Science and Technology
Islamabad
2015

MODELING AND CONTROL OF 5DOF ROBOTIC ARM USING NEURO-FUZZY CONTROLLER

by

ABDUL WAHAB NADEEM BUTT
NUST201260407MSMME62112F

A thesis submitted to the
Department of Robotics and Intelligent Machine Engineering
in partial fulfillment of the requirements for the degree of

MASTER OF SCIENCE
IN
ROBOTICS AND INTELLIGENT MACHINE ENGINEERING

Thesis Supervisor
DR. MOHSIN JAMIL

School of Mechanical and Manufacturing Engineering
National University of Science and Technology
Islamabad
2015

FORM TH-4

National University of Sciences & Technology

MASTER THESIS WORK

We hereby recommend that the dissertation prepared under our supervision by: **Abdul Wahab Nadeem Butt**, Reg. No. **NUST201260407MSMME62112F** Titled: **Modeling and Control of 5DOF Robotic Arm using Neuro-Fuzzy Controller** be accepted in partial fulfillment of the requirements for the award of MS in Robotics and Intelligent Machine Engineering degree with **(grade)**

Examination Committee Members

1. Name: ___________________________	Signature: ________________
2. Name: ___________________________	Signature: ________________
3. Name: ___________________________	Signature: ________________
Co-Supervisor's name: __________________	Signature: ________________
Supervisor's name: ____________________	Signature: ________________
	Date: ____________________

________________ Head of Department

_____________ Date

COUNTERSINGED

Date:___________

______________ Dean/Principal

To my parents,

for their boundless care and support

To my wife,

for keeping my spirit alive

To my exuberant and curious son

for being a source of enchantment and aspiration

&

To my friends

for their never ending encouragement

ACKNOWLEDGEMENTS

Al-Hamdulillah.

I would like to express my sincere gratitude to my thesis supervisor Dr. Mohsin Jamil for his support, help and guidance throughout the course of this work and my studies.

I am highly obliged to my Guidance and Examination Committee members; Dr. Shahid Ikramullah Butt, Dr. Syed Omer Gilani, and Dr. Umar Ansari for their sincere guidance and cooperation.

I would like to thank department of Robotics and Intelligent Machine Engineering and the head of the Department Dr. Yasar Ayaz, for facilitating me to complete my MS degree.

I am especially grateful to my dearest friend, Adnan Hanif for all of his assistance, advice and encouragement during this work. I am also grateful for my colleagues at Mohammad Ali Jinnah University for being a delight to be around.

I would like to thank my family for their love and support throughout my life.

Special thanks to those who motivated and encouraged me to pursue higher studies.

DECLARATION

I hereby declare that this thesis is entirely and purely my own work and based on my personal efforts and intellect under the guidance and supervision of my thesis supervisor

Dr. Mohsin Jamil

All the sources used in this thesis are properly cited and no portion of this thesis is an act of plagiarism. This work is purely done for the fulfillment of requirements for aforementioned degree in respective department. No part of this thesis is submitted for any other application for any degree or qualification in this or any other university, degree awarding or non-degree awarding college or institute.

ABSTRACT

Robotic Arms are considered essential components of any automation system. They present considerably complicated electromechanical systems with mutual interactions of robot mechanics and drives, at design of which the mechatronic approach should be taken into consideration.

The modeling problem is necessary before applying control techniques to guarantee the execution of any task according to a desired input with minimum error. The physical modeling in the SimMechanics / SIMULINK environment facilitates simulation efforts of such complex systems by seamless interfacing of ordinary Simulink block diagrams. This is not only more intuitive, it also saves the time and effort in deriving the equations of motion. Problem of Inverse Kinematics (IK) is solved using a machine learning technique i.e. Adaptive Neuro-Fuzzy Inference System (ANFIS) in contrast with the analytical solution. MATLAB, Simulink, SimMechanics and SolidWorks are used as simulation platform.

This research will undertake the following five developmental stages; firstly, the complete computer-aided design (CAD) model of a 5 DOF robotic arm is developed in SolidWorks. In the second stage, the CAD model is to be converted into physical modeling by using SimMechanics Link. Then, the ANFIS networks are trained to compute the inverse kinematics of the robot arm. In the fourth stage, the research intends to perform the simulation in which, the trajectory tracking of robot manipulator's end effector is considered as a test scenario. In last stage, the performance parameters of implemented technique are studied i.e., residual plots, convergence plots, comparison between the predicted results and analytical solution, analysis of trajectory and dynamics of robotic arm, joint torque computation through SimMechanics.

Table of Contents

List of Tables

List of Figures

List of Abbreviations

DOF	Degree of Freedom
ANFIS	Adaptive Neuro-Fuzzy Inference System
ANNs	Artificial Neural Networks
IK	Inverse Kinematics
FL	Fuzzy Logic
FIS	Fuzzy Inference System
MF	Membership Function
FCM	Fuzzy C-Means
CAD	Computer Aided Design
DH	Denavit-Hartenberg
XML	Extensible Markup Language
PID	Proportional, Integral, Derivative
LQR	Linear Quadratic Regulator
SCARA	Selective Compliance Assembly Robot Arm
PUMA	Programmable Universal Machine for Assembly
DC	Direct Current
emf	Electromotive Force
Ref	Reference

Chapter 1

Introduction

In recent years, the industrial automation and computer-aided manufacturing (CAM) has been revolutionized by the deployment of sophisticated robot arms / manipulators. It is now a norm to see robot manipulators being used for welding, painting, fabricating, inspecting and repairing different machines and components. This rapid integration of robotic arms into wide variety of industrial environments and other commercial activities is due to their high accuracy, speed and cost effectiveness. For their swift and seamless operation, modeling and analysis of robot manipulators and applying intelligent control techniques are very important aspects in robotics field of study. Functionality and performance of manipulators can be further enhanced by introducing advanced machine learning methodologies such as neural networks, fuzzy logic and hybrid neuro-fuzzy techniques. This combination is regarded as Intelligent Systems which is great source of inspiration and motivation for engineers in design field.

1.1 System Overview

Mechanical structure is considered as a key distinction between different robot types. Robots with fixed base are classified as robot manipulators. The fundamental structure of a manipulator is the serial or open kinematic chain. A manipulator is a sequence of rigid bodies called links attached to each other by means of articulations (joints). In an open chain mechanism, each joint accounts for a single degree of freedom (DOF) of the structure. The joints can be of two types: revolute (rotary motion) and prismatic (linear motion). Any manipulator can be characterized by its three main components: an arm that defines mobility, a wrist that warrants its dexterity, and an end-effector that performs the desired task required of the robot.

The robotic arm studied in this thesis is a five DOF articulated type [1] manipulator comprising of revolute joints. Dagu 5-DOF robotic arm [2] is selected as a case study, since it presents a simple inexpensive and a good example of robot manipulator which is suitable for educational purposes. Figure 1 shows the schematic diagram of Dagu robotic arm with five DOF.

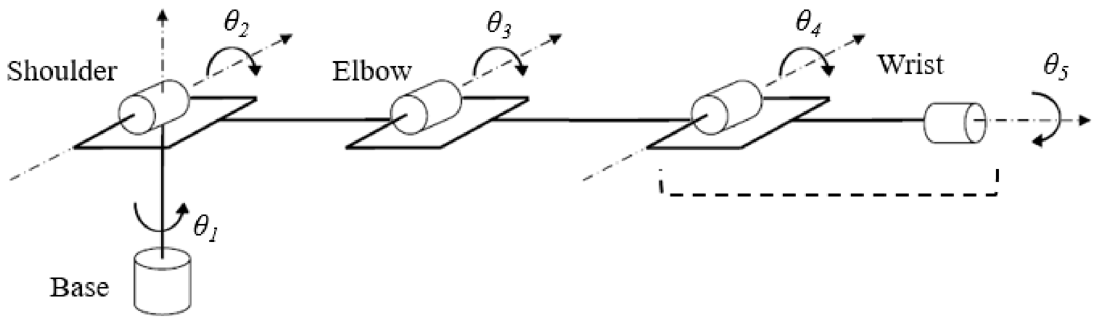


Figure 1: Schematics of 5DOF Robotic Arm with Revolute Joints

1.2 Problem Statement

Study of robotic arms is relatively a young field of multidisciplinary technology. Robot arm is considered as a complex electromechanical systems which involves interactions between robot mechanics and electric drives [3] and requires knowledge from the domains of electrical, mechanical and computer engineering. So a mechatronic approach is required to be taken into consideration for the design and control of such complicated system. The essential problem in controlling robots is to make the manipulator follow a desired trajectory. In general, number of degree of freedom (DOF) of a robot manipulator is equivalent to number of nonlinear, dynamic, coupled differential equations [1]. When designing the control of manipulator, complete mathematical model is required to compute necessary torques and angle of rotation of each motor is to move the end effector of robotic arm on a desired trajectory. Although, accurate mathematical models exist to solve this inverse dynamics task for simpler robotic systems with fewer DOF, but in general, these analytical models gets more elusive to provide a unique solution with increasing number of DOF. Thus, researchers look for intelligent learning techniques as an alternate to solve the problem of robot dynamics. Neural networks and fuzzy systems [4] (neuro-fuzzy systems, in general) are raising more and more interest in the field of real-time control thanks to their superior performance, non-linear characteristics, the capability of learning from examples, the adaptation capability, etc.

1.3 Thesis Objective

The main objective of this thesis is to solve the inverse kinematics problem of a five DOF robotic arm using adaptive neuro fuzzy inference system (ANFIS) [5]. Following points summarize the objective of this thesis work:

- To develop a 3D CAD model of robotic arm in SolidWorks®
- To represent the robotic arm in SimMechanics™ simulation environment as a multibody system
- To derive a complete mathematical model for the forward kinematics and inverse kinematics of robotic arm
- To develop mathematical model of motor actuators for different joints of robotic arm in both time domain and frequency domain.
- To generate a desired end-effector's trajectory by applying path planning algorithm.
- To implement ANFIS for predicting the inverse kinematics of robotic arm moving on a desired trajectory
- To simulate the dynamic behavior of robotic arm along with actuator motors in SimMechanics™ for calculating the velocity kinematics and joint torques.
- To apply PID based classical control technique to track the movement of the end-effector on desired trajectory

1.4 Thesis Outline

This thesis is comprised of seven chapters containing introduction, literature review, computer aided design, forward and inverse kinematics mathematical modeling, adaptive neuro-fuzzy inference system, simulation and results, conclusion and future work followed by the references and MATLAB scripts. A brief description of each chapter is as followed:

Chapter 1 provides the basic introduction, system overview and problem statement of the thesis.

In Chapter 2, literature review relevant to the research conducted in this thesis is discussed. An effort has been made to gather as much relevant information as possible, with the available resources, to comprehensively present the work done in the field of forward and inverse kinematics for serial manipulators and application of adaptive learning algorithms.

Chapter 3 is dedicated to the computer aided modeling of Dagu 5-DOF robotic arm using SolidWorks® CAD software. The importing process of CAD model into SimMechanics explorer and different aspects of SimMechanics environment are also briefly discussed here.

Chapter 4 describes the mathematics of robotic manipulators and formulation of forward and inverse kinematics. It highlights the role of Denavit-Hartenberg parameters in mathematical modeling of robotic arm and explains the mechanics of mapping the joint space into Cartesian space and vice-versa.

Chapter 5 discuss the adaptive neuro-fuzzy inference system methodology and steps involved in developing the ANFIS networks to tackle the problem of inverse kinematics in serial manipulators the multiple DOF. The important aspects of ANFIS implementation like, generating the data samples, training and testing of ANFIS networks and rule base deduction are also the part of this chapter. The residual / error plots and test results comparison are also integrated into this chapter.

Simulations carried out and the subsequent results are presented in Chapter 6. Complete Simulink model along with its subsystems, and their integration with SimMechanics model, required to carry out the necessary simulations are presented here. Trajectory tracking functionality has also been evaluated to get the idea in a broader sense.

The thesis is finally concluded in Chapter 7 with future recommendations and references.

Chapter 2

Literature Review

Industrial robots usually have simplified geometric parameters such as intersecting or parallel joints to reduce kinematic computations.[6], [7] Obtaining the forward and inverse kinematics of the industrial robots like Puma 560, Motoman L-3, Kuka KR robot and SCARA manipulators has been the main concern in field of robotics [8]–[15]. Particularly for the inverse kinematics, the complexity of the solution escalates with increasing robot's degree of freedom.

Traditional methods for computing the inverse kinematics have certain drawbacks. It is very difficult to find a closed form solution for algebraic method [16], in case of robot manipulator with DOF greater than three, geometric method [17] puts the constraint that the manipulator must have a geometric closed form solution for the first three joints, while the iterative method [18] converges to a particular solution depending on the starting conditions and fails to work near singularities. In general these methods turn to be numerically complex and cumbersome.

To overcome the difficulties and complexities of explicit mathematical modeling for robot kinematics, in recent years the focus has been shifted towards more ingenious techniques like adaptive neuro-fuzzy inference system[19]. [20]–[26] addresses the kinematics of robotic manipulators having three or less DOF using learning techniques like neural networks, fuzzy logic and ANFIS. For industrial robots with our degree of freedoms, like SCARA, neural network approach is opted in [27] while the neuro-fuzzy methodology is used in [28], [29]. Kinematics of Puma 560 and its other variants has been solved using neuro-fuzzy inference in [30], [31].

A large amount of research literature available online that discusses the kinematics analysis of industrial and high-end robots. But the same is not the case for the low cost, educational robots, like Dagu 5-DOF, which presents a huge opportunity for the young researchers to explore their kinematics, dynamics and analyze their behavior. Some of these low cost robots have also been subjected to various adaptive learning techniques as presented in [32], [33], but the literature is not rich enough.

SimMechanics and Simulink facilitates the dynamic simulation of robotic manipulators and reduces the mathematical modeling and computation significantly. These software environments are also being deployed for model validation and performance evaluations [34]–[37]. But with each upgraded version of the software every year, the capabilities of performing forward and inverse kinematics and dynamic analysis of robot manipulators are enhancing. So this requires much of the deserved attention and further advancement in research and development. For the sole purpose of highlighting these capabilities, SimMechanics and Simulink is integrated in the core architecture of this thesis.

Chapter 3

Computer Aided Design

3.1 CAD Model

The basic CAD model based on Dagu 5-DOF robotic arm is drawn in SolidWorks®. The SolidWorks® CAD software is a mechanical design automation application which uses component based 3D design approach [38]. Each link is modeled separately as part file and then assembled into a complete model. Figure 2 shows the assembled model of robotic arm in CAD platform. The fully dimensioned, detailed drawing of robotic arm is given in Appendix A.

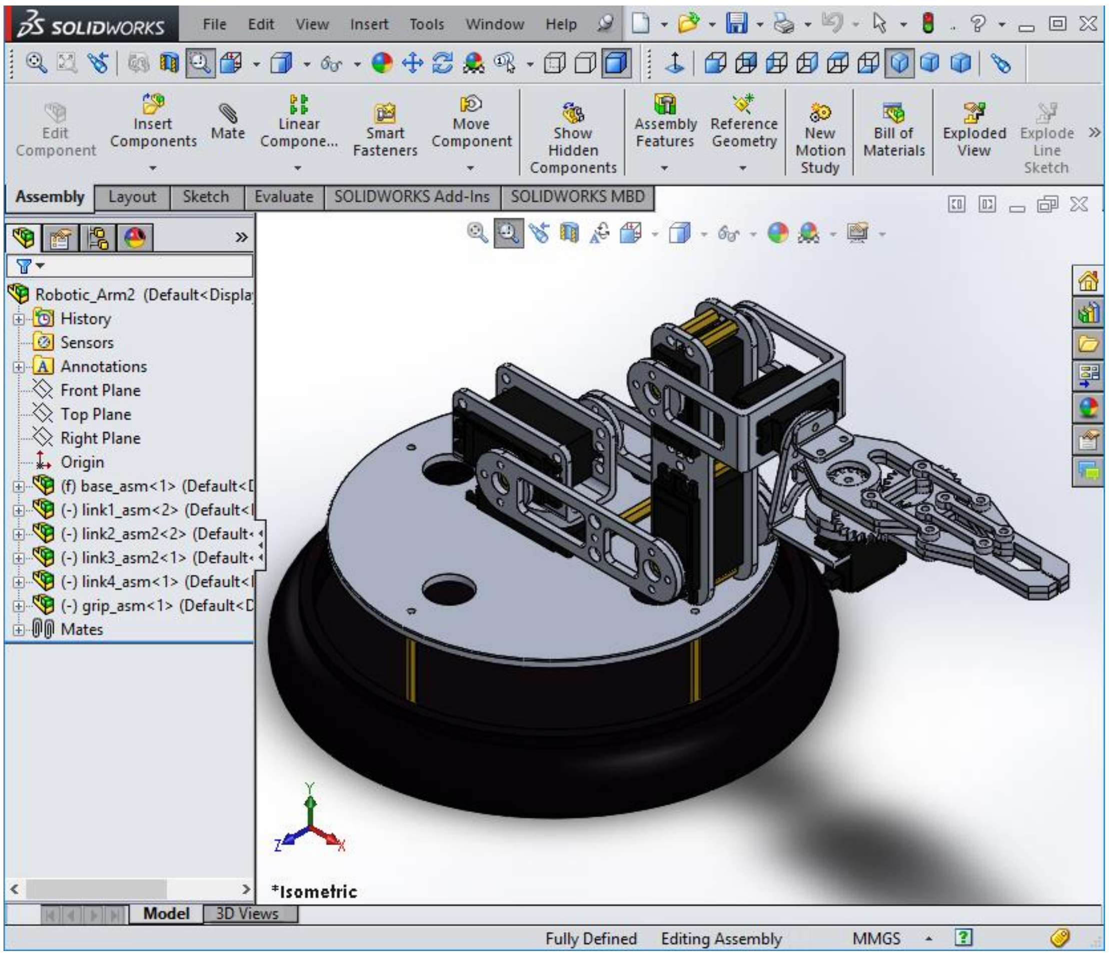


Figure 2: Assembled CAD model of Robotic Arm

3.2 SimMechanics Model

3.2.1 Introduction

Simulink® with *SimMechanics*™ software uses a block-diagram schematic approach for modeling control systems around mechanical devices. SimMechanics is a multibody simulation environment for 3D mechanical systems, such as robots, mechanisms and vehicle components. The multibody system in SimMechanics is modeled using blocks representing bodies, joints, constraints, and force elements. SimMechanics then formulates and solves the equations of motion for the modeled system. Models from CAD systems, including mass, inertia, joint, constraint, and 3D geometry, can be imported into SimMechanics. An automatically generated 3D animation helps to visualize the system dynamics [39].

3.2.2 CAD Import

CAD assembly can be translated into SimMechanics model (as shown in Figure 3) for simulation and analysis using *SimMechanics Link* which automatically bridges the gap between geometric modeling and block diagram modeling. CAD assembly is converted into SimMechanics model in two steps:

i. Exporting CAD assemblies into physical modeling XML
ii. Importing physical modeling XML to generate SimMechanics models

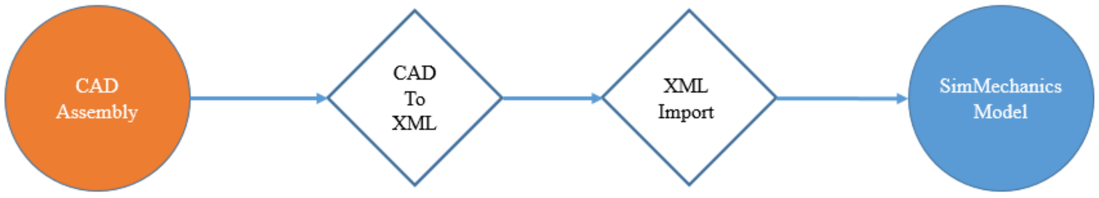


Figure 3: CAD to SimMechanics Translation

The first translation step creates an intermediate physical modeling XML file from CAD assembly. The XML file captures the mass and inertia of each part in the assembly and the constraint definitions between parts. The graphics files capture the body geometries of the assembly parts. The second translation step imports the XML to generate the SimMechanics model. The XML representations of parts and constraints become bodies and joints in a SimMechanics model.

3.2.3 Multibody Model of Robotic Arm

Figure 4 shows the block diagram of 5 DOF robotic arm as multibody system. This model contains *rigid body* subsystem blocks to represent robot base, links and gripper. The model also contains six *revolute joint* blocks for actuation, a *world frame* that provides reference for all other *rigid transform* blocks. A *mechanism configuration* defines the gravity vector in the model. A *transform sensor* is attached with the robotic arm kinematic chain to observe the position and orientation of end-effector in world coordinate system.

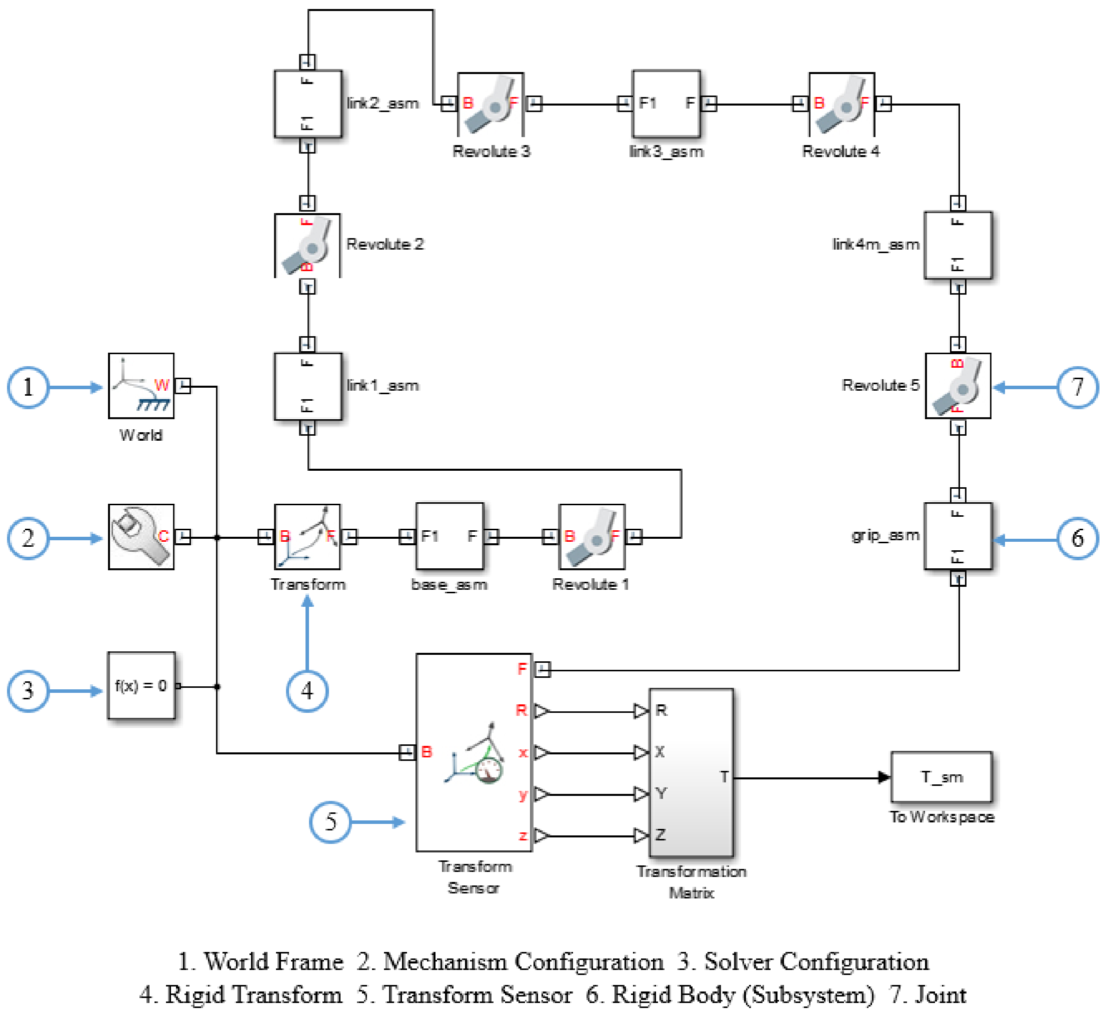
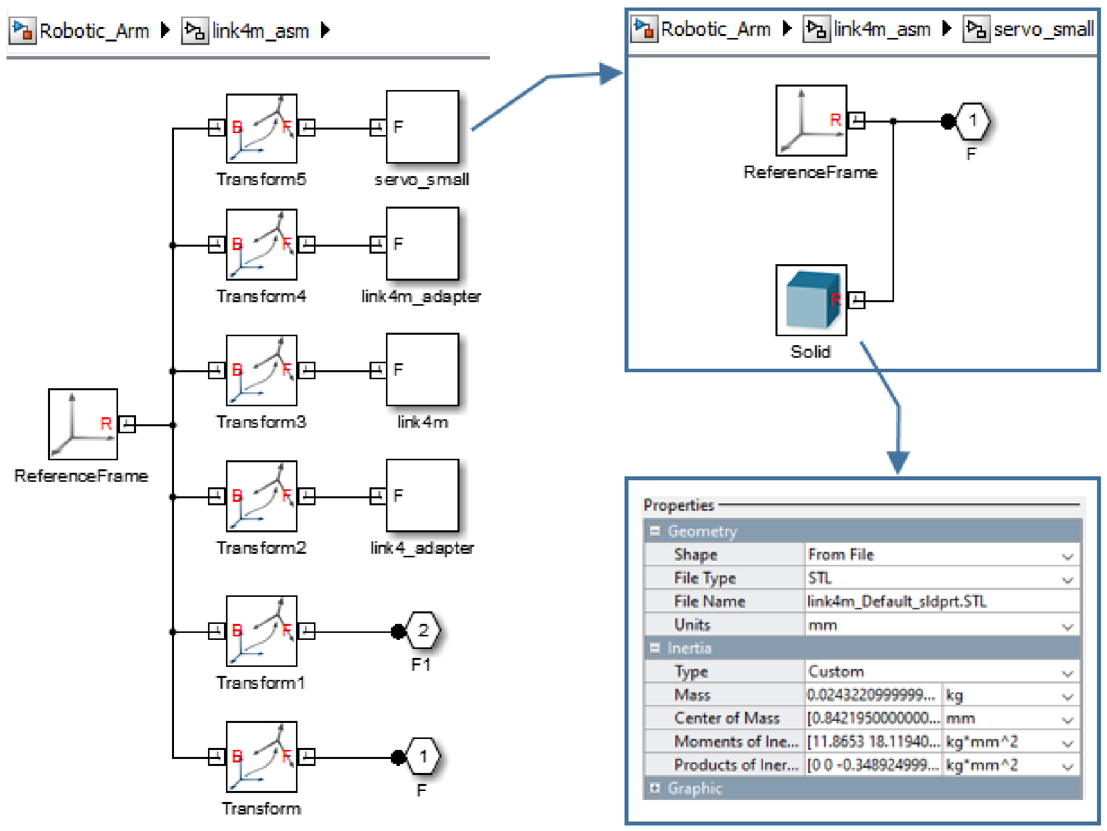


Figure 4: SimMechanics Model of 5 DOF Robotic Arm

Each rigid body subsystem contains solid element blocks which store geometrical, inertial and graphical information of rigid bodies and their spatial relationship as shown in Figure 5 for a particular link of robotic arm.



Properties

⊟ Geometry		
Shape	From File	
File Type	STL	
File Name	link4m_Default_sldprt.STL	
Units	mm	
⊟ Inertia		
Type	Custom	
Mass	0.0243220999999...	kg
Center of Mass	[0.8421950000000...	mm
Moments of Ine...	[11.8653 18.11940...	kg*mm^2
Products of Iner...	[0 0 -0.348924999...	kg*mm^2
⊞ Graphic		

Figure 5: Rigid Body Subsystem

The *SimMechanics Explorer* automatically creates the 3D visualization model and animates the results from multiple views simultaneously during simulations. The completely rendered model of Dagu 5-DOF robotic arm in SimMechanics Explorer environment is shown in Figure 6.

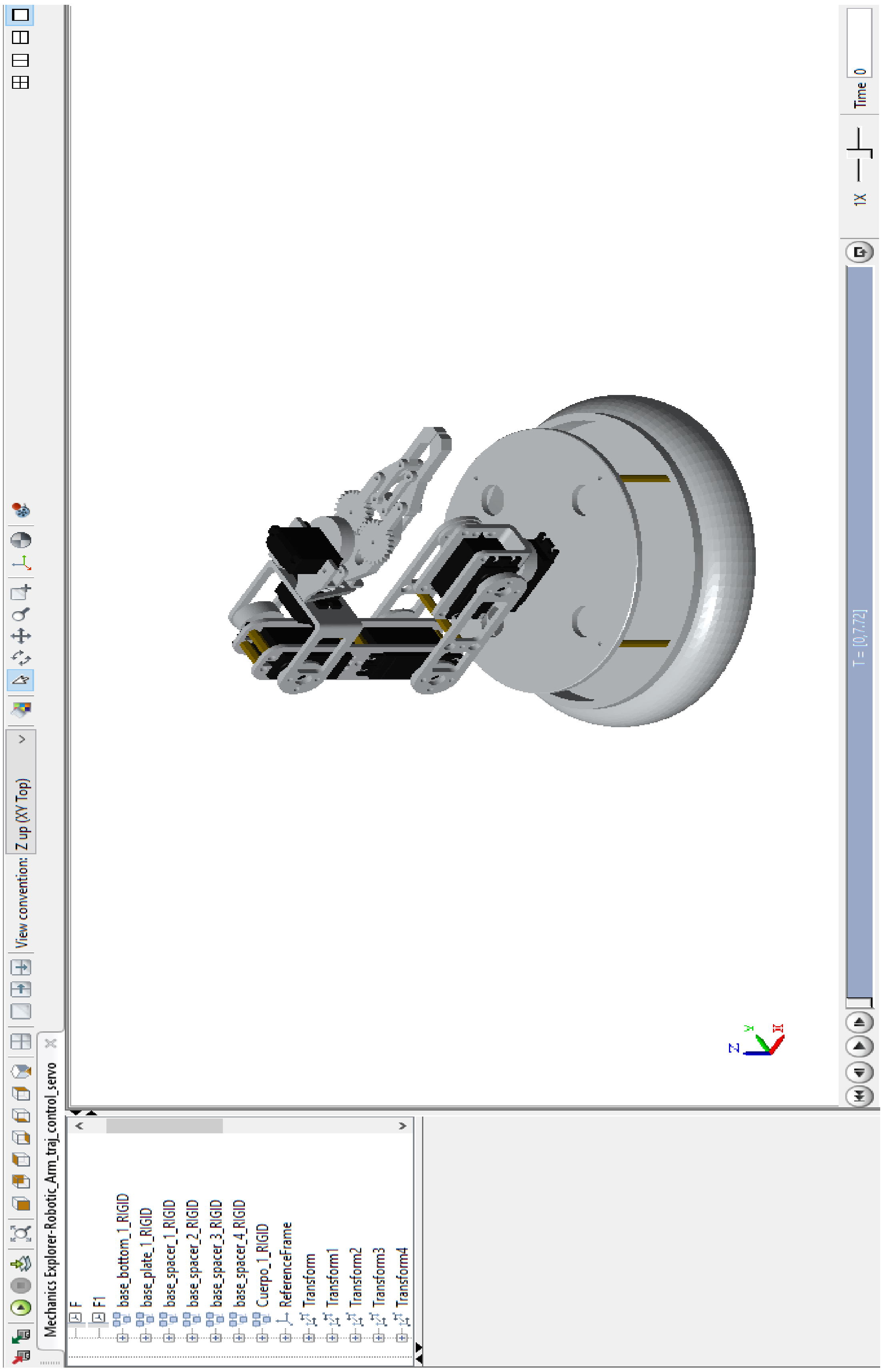


Figure 6: Multibody Model of Robotic Arm Displayed in 3D Animation in SimMechanics Explorer

Chapter 4

Kinematics and Modeling

4.1 Kinematics

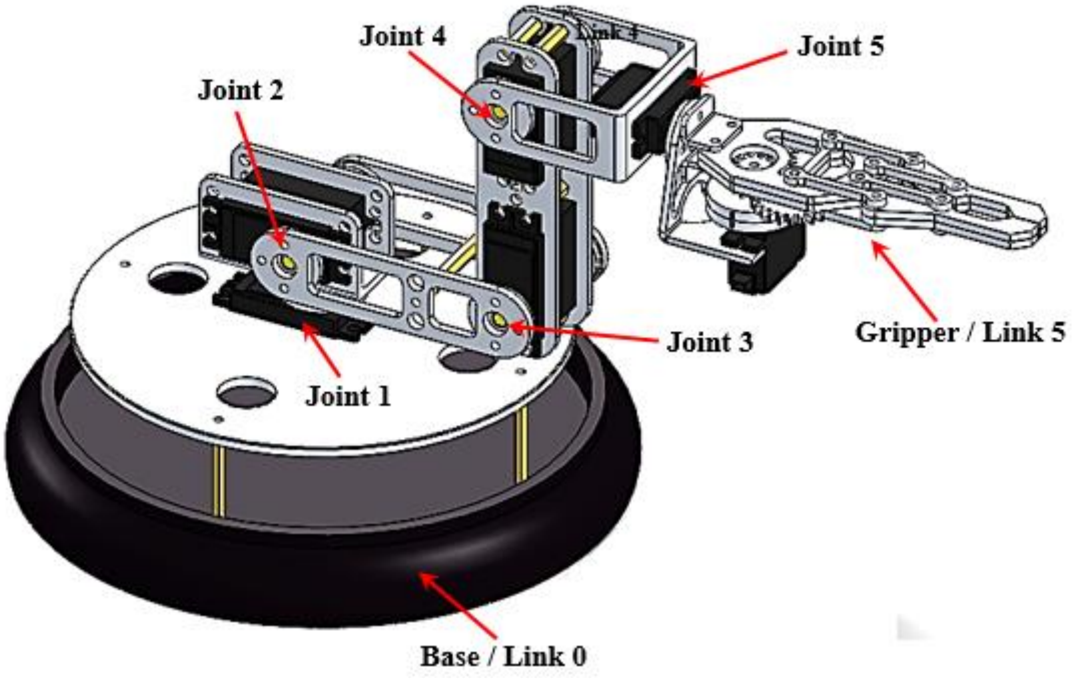
Kinematics is the study of motion of a body without taking into account the dynamics i.e. forces and moments that cause the motion. Basically, it is the study of position, velocity and acceleration and other higher order derivatives of position with respect to time or any other variables [40]. The problem is further divided into two parts: *forward kinematics* and *inverse kinematics*.

4.2 Forward Kinematics

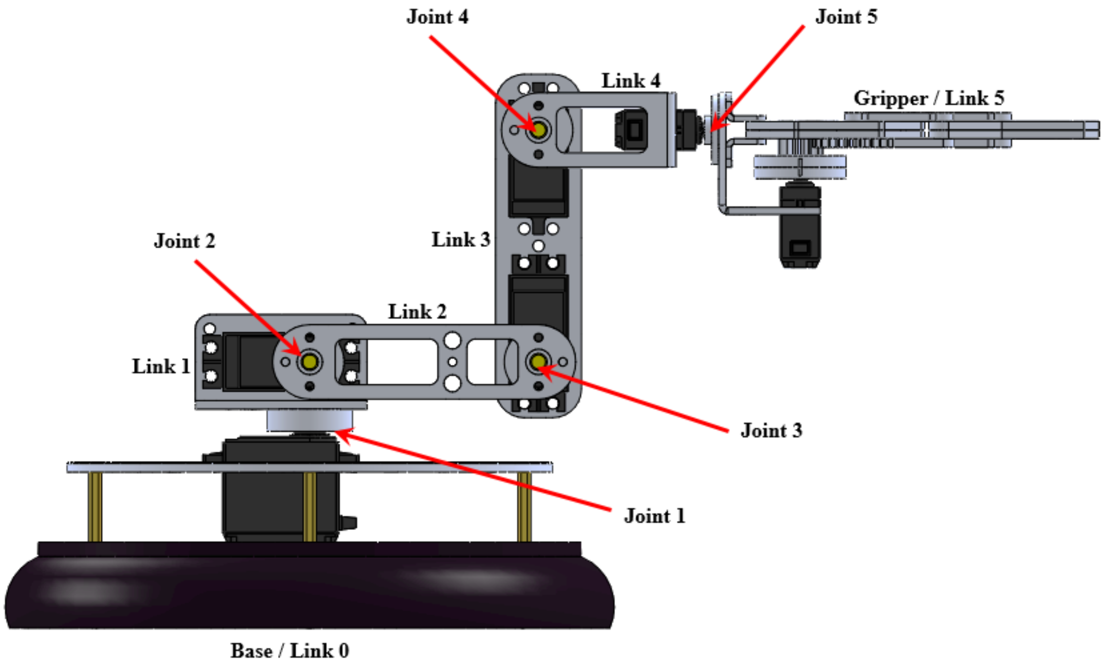
Forward kinematics problem is to determine the position and orientation of the manipulator's end-effector (with reference to a fixed base) in Cartesian space while knowing the values of different joint variables. Robot manipulators are meant to move parts and tools around in space which naturally leads to representing the location of parts, tools and robot mechanism itself. For the purpose of mathematically representing the location of these items, coordinate frames are assigned to these bodies and mapping of these coordinate frames is established with reference to one another. So the forward kinematics problem can be rephrased as to determine a transformation function that maps the frame attached to one link in terms of the frame attached to another link. Then these transformation functions are concatenated to solve for the position and orientation of last frame with respect to fixed world (reference) frame.

4.2.1 Link-Frame Assignment

A manipulator is a set of *links* connected in a serial chain by means of *joints*. The links are numbered as 'link 0' to 'link *n*' starting from the fixed base to the free end of a robotic arm. While the joints are numbered from 'joint 1' to joint *n*' for '*n*' degree of freedom robotic arm. Figure 7 shows different links and joints of Dagu 5-DOF robotic arm. In manipulator's terms, kinematics establish the relationship of these joint variables with the location (position and orientation) of these links.



(a) Isometric View of Dagu 5-DOF



(b) Side View of Dagu 5-DOF

Figure 7: Links and Joints of Robot Manipulator (a) Isometric View (b) Side View

In order to describe the location of each link relative to its neighbours, reference frames are attached with various parts of the mechanism and their change in location is studied as the mechanism actuates. The link frames are named by number according to the link to which they are attached. That is, frame $\{i\}$ is attached rigidly to link i. Figure 8 shows the frame assignment of Dagu 5-DOF robotic arm.

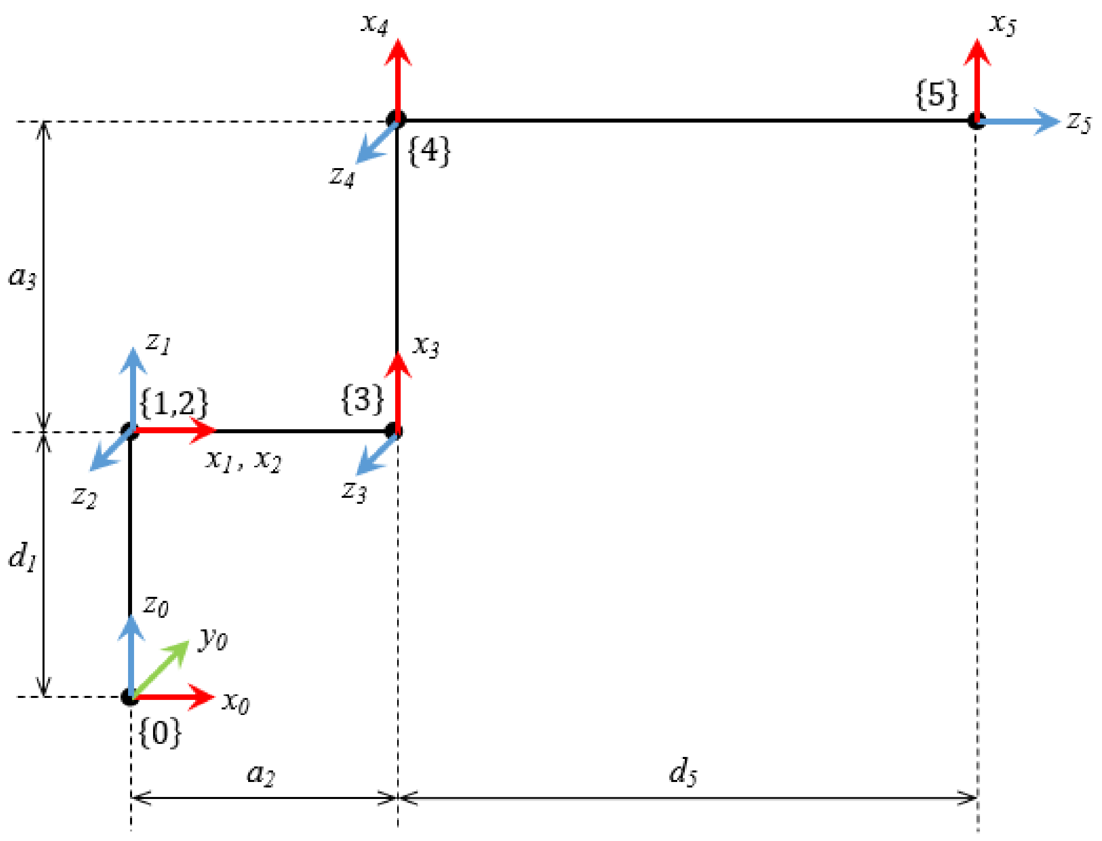


Figure 8: Robotic Arm Frame Assignment

4.2.2 Frame Transformation

A general tool to mathematically represent the description of one frame with respect to another frame is known as *Homogeneous Transform*. It is a 4x4 matrix which contains the position and orientation information of the frames. The description of frame $\{B\}$ with respect to frame $\{A\}$ is expressed as:

$$ {}^{A}_{B}T = \left[\begin{array}{ccc|c} & {}^{A}_{B}R & & {}^{A}P_{BORG} \\ \hline 0 & 0 & 0 & 1 \end{array}\right] \quad (4.1)$$

Where, ${}^{A}_{B}\boldsymbol{T}$ is a 4x4 homogeneous transformation matrix ${}^{A}_{B}\boldsymbol{R}$ is a 3x3 rotation matrix, which describes the orientation of frame $\{B\}$ relative to frame $\{A\}$ and ${}^{A}\boldsymbol{P}_{\boldsymbol{BORG}}$ is a 3x1 position vector, which locates the origin of frame $\{B\}$ in frame $\{A\}$.

4.2.3 Denavit—Hartenberg Parameters

Four parameters known as **Denavit—Hartenberg (DH) parameters** [41] are associated to each link to fully define the kinematics of any robot manipulator. Using conventions defined in [40] for assigning frames to links, the modified DH parameters are defined as:

Link twist: $a_i =$ the distance from $\hat{Z}_i$ to $\hat{Z}_{i+1}$ measured along $\hat{X}_i$

Link length: $\alpha_i =$ the angle from $\hat{Z}_i$ to $\hat{Z}_{i+1}$ measured about $\hat{X}_i$

Link offset: $d_i =$ the distance from $\hat{X}_{i-1}$ to $\hat{X}_i$ measured along $\hat{Z}_i$

Joint angle: $\theta_i =$ the distance from $\hat{X}_{i-1}$ to $\hat{X}_i$ measured along $\hat{Z}_i$

The DH parameters for Dagu 5-DOF robotic arm using the frame assignments as shown in Figure 8 are listed below in Table 1.

Table 1: DH Parameters for Dagu 5-DOF Robotic Arm

i	$\boldsymbol{\alpha_{i-1}}$	$\boldsymbol{a_{i-1}}$	$\boldsymbol{d_i}$	$\boldsymbol{\theta_i}$
1	*0*	*0*	d_1 *= 93mm*	θ_1
2	*90°*	*0*	*0*	θ_2
3	*0*	a_2 *= 80mm*	*0*	θ_3
4	*0*	a_3 *= 81mm*	*0*	θ_4
5	*90°*	*0*	d_5 *= 195mm*	θ_5

The transformation matrix $\boldsymbol{{}^{i-1}_{\ \ i}T}$ that describes the frame $\{i\}$ relative to frame $\{i-1\}$ can be defined in terms of four above mentioned DH parameters as

$$ {}^{i-1}_{\ \ i}T = R_X(\alpha_{i-1})D_X(a_{i-1})R_Z(\theta_i)D_Z(d_i) \tag{4.2}$$

Where,

$$R_X(\alpha_{i-1}) = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & c\alpha_{i-1} & -s\alpha_{i-1} & 0 \\ 0 & s\alpha_{i-1} & c\alpha_{i-1} & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{4.3}$$

$$D_X(a_{i-1}) = \begin{bmatrix} 1 & 0 & 0 & a_{i-1} \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{4.4}$$

$$R_Z(\theta_i) = \begin{bmatrix} c\theta_i & -s\theta_i & 0 & 0 \\ s\theta_i & c\theta_i & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{4.5}$$

$$D_Z(d_i) = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & d_i \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{4.6}$$

Here, $c\theta_i$ is shorthand for $\cos\theta_i$, $s\theta_i$ for $\sin\theta_i$ and so on. This shows that the transformation matrix ${}^{i-1}_{i}T$ is a product of four sub-transforms. Where each sub-transform matrix is function of one link parameter only. Multiplying out each of these transforms in Equation (4.2) to obtain the general form of ${}^{i-1}_{i}T$:

$${}^{i-1}_{i}T = \begin{bmatrix} c\theta_i & -s\theta_i & 0 & a_{i-1} \\ s\theta_i c\alpha_{i-1} & c\theta_i c\alpha_{i-1} & -s\alpha_{i-1} & -s\alpha_{i-1} d_i \\ s\theta_i s\alpha_{i-1} & c\theta_i s\alpha_{i-1} & c\alpha_{i-1} & c\alpha_{i-1} d_i \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{4.7}$$

Once the individual link-transformation matrices have been computed using the DH parameters, the forward kinematic equations of the robot manipulator is straightforward. These link transformations matrices are then multiplied to each other to find the final transformation matrix ${}^{0}_{N}T$, that relates the position and orientation of the end effector's frame $\{N\}$ with respect to the fixed frame or base frame $\{0\}$:

$${}^{0}_{N}T = {}^{0}_{1}T\,{}^{1}_{2}T \ldots {}^{N-1}_{N}T \tag{4.8}$$

The final transformation matrix ${}^{0}_{N}T$ is function of all *n* joint variables. For Dagu 5-DOF robotic arm, the individual link transformation matrices are computed using Equation (4.7) and Table 1, as follows:

$${}^{0}_{1}T = \begin{bmatrix} c_1 & -s_1 & 0 & 0 \\ s_1 & c_1 & 0 & 0 \\ 0 & 0 & 1 & d_1 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{4.9}$$

$$ {}^{1}_{2}T = \begin{bmatrix} c_2 & -s_2 & 0 & 0 \\ 0 & 0 & -1 & 0 \\ s_2 & c_2 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{4.10} $$

$$ {}^{2}_{3}T = \begin{bmatrix} c_3 & -s_3 & 0 & a_2 \\ s_3 & c_3 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{4.11} $$

$$ {}^{3}_{4}T = \begin{bmatrix} c_4 & -s_4 & 0 & a_3 \\ s_4 & c_4 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{4.12} $$

$$ {}^{4}_{5}T = \begin{bmatrix} c_5 & -s_5 & 0 & 0 \\ 0 & 0 & -1 & -d_5 \\ s_5 & c_5 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{4.13} $$

Here, c_1 is shorthand for $\cos(\theta_1)$, s_1 for $\sin(\theta_1)$ and so on.Using Equation (4.8), the link transformations above are multiplied to find the final transformation matrix that relates the frame $\{5\}$ of end-effector to fixed base frame $\{0\}$ of the robotic arm:

$$ {}^{0}_{5}T = {}^{0}_{1}T(\theta_1){}^{1}_{2}T(\theta_2){}^{2}_{3}T(\theta_3){}^{3}_{4}T(\theta_4){}^{4}_{5}T(\theta_5) = \begin{bmatrix} r_{11} & r_{12} & r_{13} & p_x \\ r_{21} & r_{22} & r_{23} & p_y \\ r_{31} & r_{32} & r_{33} & p_z \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{4.14} $$

Equation (4.14) consists mainly of two components: a 3x3 rotation matrix which determines the orientation of the end-effector relative to the base and a 3x1 displacement vector which defines the position of end-effector relative to the base:

$$ {}^{0}_{5}R = \begin{bmatrix} r_{11} & r_{12} & r_{13} \\ r_{21} & r_{22} & r_{23} \\ r_{31} & r_{32} & r_{33} \end{bmatrix} \tag{4.15} $$

and

$$ {}^{0}P_5 = \begin{bmatrix} p_x \\ p_y \\ p_z \end{bmatrix} \tag{4.16} $$

here,

$$ r_{11} = c_1 c_{234} c_5 + s_1 s_5 $$

$$ r_{12} = -c_1 c_{234} s_5 + s_1 c_5 $$

$$r_{13} = c_1 s_{234}$$

$$r_{21} = s_1 c_{234} c_5 - c_1 s_5$$

$$r_{22} = -s_1 c_{234} s_5 - c_1 c_5$$

$$r_{23} = s_1 s_{234}$$

$$r_{31} = s_{234} c_5$$

$$r_{32} = -s_{234} s_5$$

$$r_{33} = -c_{234}$$

$$p_x = (a_2 c_2 + a_3 c_{23} + d_5 s_{234}) c_1$$

$$p_y = (a_2 c_2 + a_3 c_{23} + d_5 s_{234}) s_1$$

$$p_z = d_1 + a_2 s_2 + a_3 s_{23} - d_5 c_{234} \qquad (4.17)$$

Where, c_{23} is shorthand for $\cos(\theta_2 + \theta_3)$, s_{234} for $\sin(\theta_2 + \theta_3 + \theta_4)$ and so on. These are the basic equations for all kinematic analysis of Dagu 5-DOF robotic arm.

4.3 Inverse Kinematics

Inverse kinematics problem is to determine the set of joint angles in order to achieve a desired position and orientation of the manipulator's end-effector (with reference to a fixed base) in Cartesian space. The inverse kinematics problem can be interpreted as computing joint space description of the robotic arm with knowledge of the Cartesian space description of its end-effector as shown in Figure 9. The inverse kinematic solution of a manipulator is a more difficult due to its non-linear and transcendental nature.

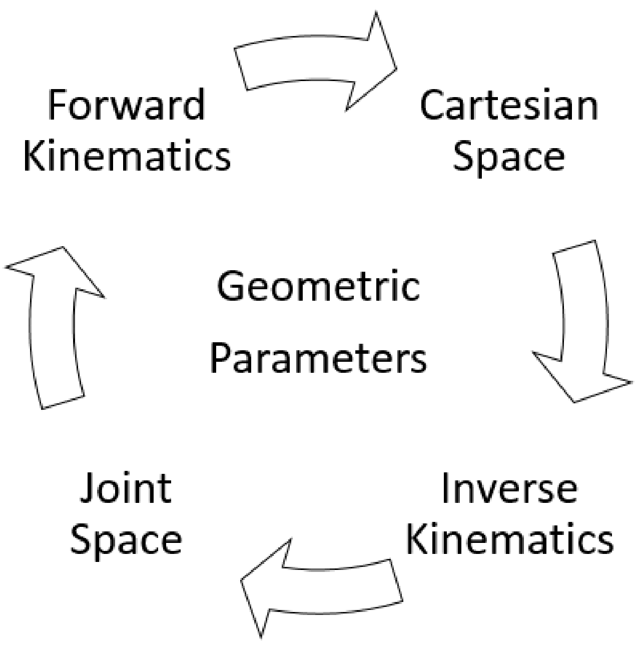


Figure 9: Kinematic Loop

For Dagu 5-DOF robotic arm this IK problem is to solve for the five joint angles θ_1 to θ_5 for given set of numerical values of all sixteen elements of the final transformation matrix ${}^{0}_{5}T$ in Equation (4.14). Normally, there is no unique analytical solution to a particular goal location for a robotic arm with five DOF. The number of solutions depends on the number of joint variables, link parameters (a_i, α_{i-1}, d_i and θ_i) and the allowable range ranges of motion of the joints. The allowable ranges of various joints of Dagu 5-DOF are discussed in Table 2 below:

Table 2: Robotic Arm Joint Limits

Joint	***Joint Limits*** *(radians)*
1	$-\frac{3\pi}{4} \leq \theta_1 \leq \frac{3\pi}{4}$
2	$0 \leq \theta_2 \leq \pi$
3	$-\frac{3\pi}{4} \leq \theta_3 \leq \frac{3\pi}{4}$
4	$-\frac{\pi}{2} \leq \theta_4 \leq \frac{\pi}{2}$
5	$-\pi \leq \theta_5 \leq \pi$

There are two basic approaches namely, geometric and algebraic to find the analytical solution of the inverse kinematics problem. We have used combination of both the approaches to find the analytical solution of Dagu 5-DOF robotic arm as follow:

To solve for first joint variable θ_1, Equation (4.14) is rearranged as

$$[{}^0_1T(\theta_1)]^{-1}\,{}^0_5T = {}^1_2T(\theta_2)\,{}^2_3T(\theta_3)\,{}^3_4T(\theta_4)\,{}^4_5T(\theta_5) \tag{4.18}$$

By using transform arithmetic,

$$[{}^0_1T(\theta_1)]^{-1}\,{}^0_5T = {}^1_0T\,{}^0_5T = {}^1_5T \tag{4.19}$$

Where,

$$ {}^1_5T = {}^1_2T(\theta_2)\,{}^2_3T(\theta_3)\,{}^3_4T(\theta_4)\,{}^4_5T(\theta_5) \tag{4.20}$$

Inverting ${}^0_1T$, we can write (4.19) as

$$\begin{bmatrix} c_1 & s_1 & 0 & 0 \\ -s_1 & c_1 & 0 & 0 \\ 0 & 0 & 1 & -d_1 \\ 0 & 0 & 0 & 1 \end{bmatrix}\begin{bmatrix} r_{11} & r_{12} & r_{13} & p_x \\ r_{21} & r_{22} & r_{23} & p_y \\ r_{31} & r_{32} & r_{33} & p_z \\ 0 & 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} {}^1r_{11} & {}^1r_{12} & {}^1r_{13} & {}^1p_x \\ {}^1r_{21} & {}^1r_{22} & {}^1r_{23} & {}^1p_y \\ {}^1r_{31} & {}^1r_{32} & {}^1r_{33} & {}^1p_z \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{4.21}$$

Equating right hand side of Equation (4.21) using (4.20),

$$\begin{bmatrix} c_1r_{11}+s_1r_{21} & c_1r_{12}+s_1r_{22} & c_1r_{13}+s_1r_{23} & c_1p_x+s_1p_y \\ -s_1r_{11}+c_1r_{21} & -s_1r_{12}+c_1r_{22} & -s_1r_{13}+c_1r_{23} & -s_1p_x+c_1p_y \\ r_{31} & r_{32} & r_{33} & p_z-d_1 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$= \begin{bmatrix} {}^1r_{11} & {}^1r_{12} & s_{234} & {}^1p_x \\ -s_5 & -c_5 & 0 & 0 \\ {}^1r_{31} & {}^1r_{32} & -c_{234} & {}^1p_z \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{4.22}$$

Equating elements (2, 4) on both sides of Equation (4.22),

$$-s_1p_x + c_1p_y = 0$$

$$\boldsymbol{\theta_1 = Atan2(p_y\,, p_x)} \tag{4.23}$$

Now that θ_1 is known, left hand side of Equation (4.22) is known. So, by equating elements (2, 1) and (2, 2) on both sides of this equation, joint variable θ_5 can be computed as

$$s_1 r_{11} - c_1 r_{21} = s_5$$
$$s_1 r_{12} - c_1 r_{22} = c_5$$

$$\boldsymbol{\theta_5 = Atan2(s_1 r_{11} - c_1 r_{21},\ s_1 r_{12} - c_1 r_{22})} \qquad (4.24)$$

Also, by equating elements (1, 3) and (2, 3) of Equation (4.22) we get another useful result:

$$c_1 r_{13} + s_1 r_{23} = s_{234}$$
$$r_{33} = -c_{234}$$

$$\theta_{234} = Atan2(c_1 r_{13} + s_1 r_{23},\ -r_{33}) \qquad (4.25)$$

Here,

$$\theta_{234} = \gamma = \theta_2 + \theta_3 + \theta_4 \qquad (4.26)$$

To solve for individual θ_2, θ_3 and θ_4, we used geometric approach [17]. Figure 10 shows plane of Dagu 5-DOF robotic arm with point A at joint axis 2, point B at joint axis 3 and point C at joint axis 4.

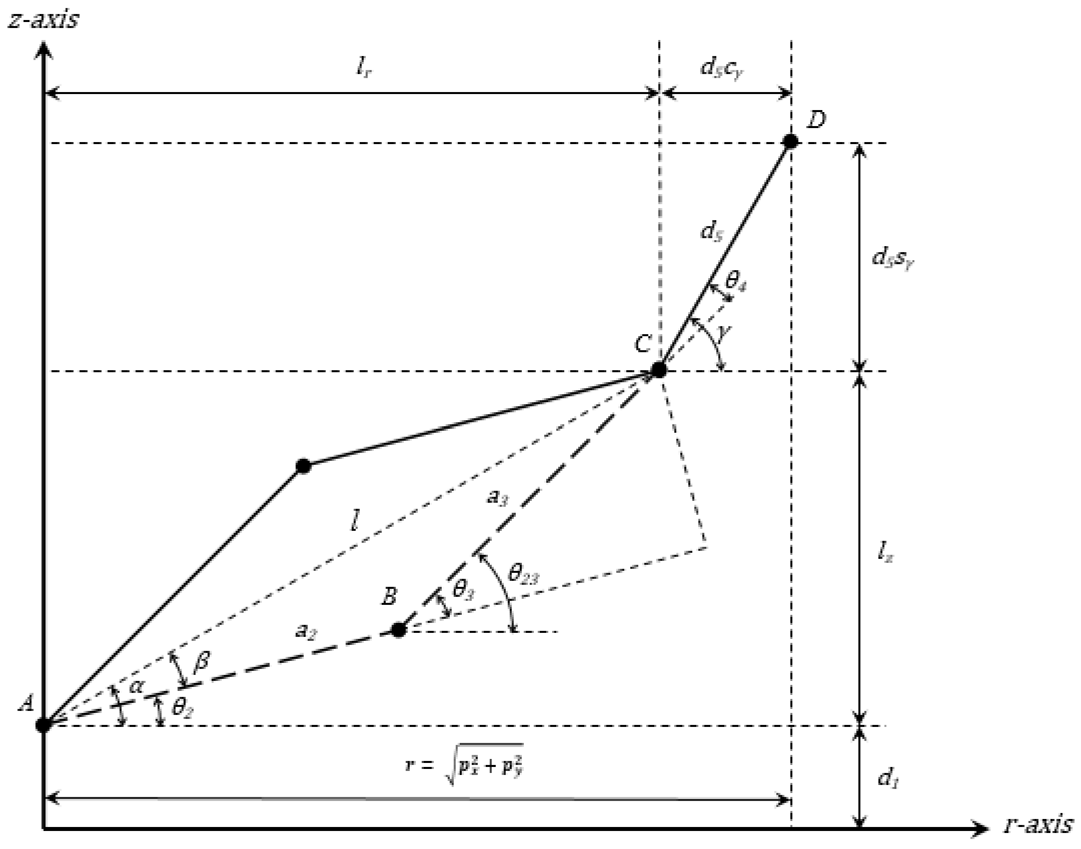


Figure 10: Plane of Robotic Arm

where,

$$l = \sqrt{l_r^2 + l_z^2}$$

$$l_r = r - d_5 c_\gamma$$

$$r = \sqrt{p_x^2 + p_y^2}$$

$$l_z = p_z - d_1 - d_5 s_\gamma$$

$$\alpha = Atan2(l_z\,, l_r) \tag{4.27}$$

Using law of cosines, $\angle ABC$ is solved for joint variable θ_3:

$$l^2 = a_2^2 + a_3^2 - 2a_2a_3\cos(180 - \theta_3)$$

$$l^2 = a_2^2 + a_3^2 + 2a_2a_3c_3$$

$$c_3 = \frac{l^2 - a_2^2 - a_3^2}{2a_2a_3}$$

$$s_3 = \pm\sqrt{1 - \mathrm{c}_3^2}$$

$$\boldsymbol{\theta_3 = Atan2(s_3\,, c_3)} \tag{4.28}$$

Equation (4.28) leads to two different possible solution of joint variable θ_3, corresponding to plus or minus sign for s_3.

To solve for joint variable θ_2,

$$l\cos\beta = a_2 + a_3c_3$$

$$l\sin\beta = a_3s_3$$

$$\beta = Atan2(a_3s_3\,, a_2 + a_3c_3) \tag{4.29}$$

The elbow up or elbow down configuration of robotic arm leads to two possible solutions of θ_2 based on the following geometric information:

$$\theta_2 = \alpha \pm \beta$$

Combining Equation (4.27) and (4.29) we can write above equation as

$$\boldsymbol{\theta_2 = Atan2(l_z\,, l_r) \pm Atan2(a_3s_3\,, a_2 + a_3c_3)} \tag{4.30}$$

Now that θ_2 and θ_3 are known, θ_4 is computed using Equation (4.26) as:

$$\boldsymbol{\theta_4 = \gamma - \theta_2 - \theta_3} \tag{4.31}$$

Equation (4.23), (4.24), (4.28), (4.30) and (4.31) are the desired results of inverse kinematics of Dagu 5-DOF robotic arm.

4.4 Differential Kinematics

In order to analyze and control the motion of manipulator and for smooth trajectory planning of the end-effector, linear and angular velocities of robotic arm are necessary along with the position analysis done in the previous sections. Since a robotic manipulator is a series of links, each one capable of motion relative to its neighbours, therefore velocity of each link is needed to be computed in a particular order, starting from the base frame $\{0\}$. For revolute joints, Equation (4.32) and (4.33) illustrates the computation of each link's linear ($\dot{v}$) and angular ($\dot{\omega}$) velocities in terms of joint velocities $\dot{\theta}$ with respect to origin of each links' attached frame.

$${}^{i+1}\omega_{i+1} = {}^{i+1}_{i}R\ {}^{i}\omega_i + \dot{\theta}_{i+1}\,{}^{i+1}\hat{Z}_{i+1} \tag{4.32}$$

$${}^{i+1}v_{i+1} = {}^{i+1}_{i}R\left({}^{i}v_i + {}^{i}\omega_i \times {}^{i}P_{i+1}\right) \tag{4.33}$$

By applying these equations from link to link, ${}^{n}v_n$ and ${}^{n}\omega_n$for the last link $\{n\}$ are computed. The further mapping between the joint velocities $\dot{\theta}$, and linear ($\dot{v}$) and angular ($\dot{\omega}$) velocities of end-effector is defined by a *Jacobian matrix* illustrated in Equation (4.34).

$$V = J(\Theta)\dot{\Theta} \tag{4.34}$$

where, V is 6x1 vector of linear and angular velocities of a link represented as

$$V = \begin{bmatrix} v \\ \omega \end{bmatrix}$$

Jacobian is a multidimensional form of derivatives. Jacobian of any dimension (even non-square) can be defined. The number of rows of a Jacobean defines the number of degree of freedom in Cartesian space, normally six (*x, y, z, roll (*α*), pitch (*β*)* and *yaw (*γ*)*) and the number of columns defines the number of joints (DOF) of robotic manipulator. Equation (4.35) shows the Jacobian matrix of dimension 6x5 for the Dagu 5-DOF robotic arm.

$$J(\Theta) = [J_1 \quad J_2 \quad J_3 \quad J_4 \quad J_5] = \begin{bmatrix} \frac{\partial x}{\partial \theta_1} & \frac{\partial x}{\partial \theta_2} & \frac{\partial x}{\partial \theta_3} & \frac{\partial x}{\partial \theta_4} & \frac{\partial x}{\partial \theta_5} \\ \frac{\partial y}{\partial \theta_1} & \frac{\partial y}{\partial \theta_2} & \frac{\partial y}{\partial \theta_3} & \frac{\partial y}{\partial \theta_4} & \frac{\partial y}{\partial \theta_5} \\ \frac{\partial z}{\partial \theta_1} & \frac{\partial z}{\partial \theta_2} & \frac{\partial z}{\partial \theta_3} & \frac{\partial z}{\partial \theta_4} & \frac{\partial z}{\partial \theta_5} \\ \frac{\partial \alpha}{\partial \theta_1} & \frac{\partial \alpha}{\partial \theta_2} & \frac{\partial \alpha}{\partial \theta_3} & \frac{\partial \alpha}{\partial \theta_4} & \frac{\partial \alpha}{\partial \theta_5} \\ \frac{\partial \beta}{\partial \theta_1} & \frac{\partial \beta}{\partial \theta_2} & \frac{\partial \beta}{\partial \theta_3} & \frac{\partial \beta}{\partial \theta_4} & \frac{\partial \beta}{\partial \theta_5} \\ \frac{\partial \gamma}{\partial \theta_1} & \frac{\partial \gamma}{\partial \theta_2} & \frac{\partial \gamma}{\partial \theta_3} & \frac{\partial \gamma}{\partial \theta_4} & \frac{\partial \gamma}{\partial \theta_5} \end{bmatrix} \tag{4.35}$$

Here,

$$J_1 = \begin{bmatrix} -s_1c_{234}d_5 - s_1c_{23}a_3 - s_1c_2a_2 \\ c_1c_{234}d_5 + c_1c_{23}a_3 + c_1c_2a_2 \\ 0 \\ 0 \\ 0 \\ 1 \end{bmatrix}$$

$$J_2 = \begin{bmatrix} -s_1c_{234}d_5 - s_1c_{23}a_3 - s_1c_2a_2 \\ c_1c_{234}d_5 + c_1c_{23}a_3 + c_1c_2a_2 \\ 0 \\ 0 \\ 0 \\ 1 \end{bmatrix}$$

$$J_3 = \begin{bmatrix} -c_1(s_{234}d_5 + s_{23}a_3 + s_2a_2) \\ -s_1(s_{234}d_5 + s_{23}a_3 + s_2a_2) \\ s_1(s_1c_{234}d_5 + s_1c_{23}a_3 + s_1c_2a_2) + c_1(c_1c_{234}d_5 + c_1c_{23}a_3 + c_1c_2a_2) \\ s_1 \\ c_1 \\ 0 \end{bmatrix}$$

$$J_4 = \begin{bmatrix} -c_1(s_{234}d_5 + s_{23}a_3) \\ -s_1(s_{234}d_5 + s_{23}a_3) \\ s_1(s_1c_{234}d_5 + s_1c_{23}a_3) + c_1(c_1c_{234}d_5 + c_1c_{23}a_3) \\ s_1 \\ c_1 \\ 0 \end{bmatrix}$$

$$J_5 = \begin{bmatrix} c_1 s_{234} d_5 \\ s_1 s_{234} d_5 \\ c_{234} d_5 \\ s_1 \\ c_1 \\ 0 \end{bmatrix}$$

The inverse of the problem stated in Equation (4.34) is to determine the joint variable velocities $\dot{\theta}$ for a known set of velocities of end-effector. This is more interesting from robotic manipulator's perspective, since in practicality, the desired trajectory on which the end-effector is tasked to move is known in advance and the goal is to compute the respective joint trajectories which will result in this controlled motion. Perhaps, it is also surprising that inverse velocity relationship is simpler than the inverse position kinematics problem. When the Jacobian is square and non-singular matrix then the inverse problem can be simply solved by inverting the Jacobean matrix as:

$$\dot{\Theta} = J(\Theta)^{-1} V \tag{4.36}$$

For Dagu 5-DOF or other manipulators that don't have a square Jacobean, the inversion is either done by pseudo inverse of J or using the integration techniques on Equation (4.36).

4.5 DC Motor Modeling

DC motor is a common actuator device that delivers energy to the load. DC motors have numerous control applications in robotic manipulators due to their well-behaved speed-torque characteristics and controllability. The DC motor works on the principle that a current i carrying conductor in a magnetic field ϕ experiences a Force $f = i \times \phi$. In a *permanent magnet* DC Motor as shown in Figure 11, whose stator consists of a permanent magnet and thus has a constant flux ϕ, the torque τ_m on the rotor is controlled only by controlling the armature current i_a as:

$$\tau_m = K_t i_a \tag{4.37}$$

where K_t is the torque constant. In addition, whenever a conductor while moving cuts through a magnetic field, a voltage v_b, called back emf, is generated across its terminals.

$$v_b = K_e \dot{\theta} = K_e \frac{d\theta}{dt} \tag{4.38}$$

where K_e is the *back emf* constant. Consider the schematic diagram of the Figure 11 where

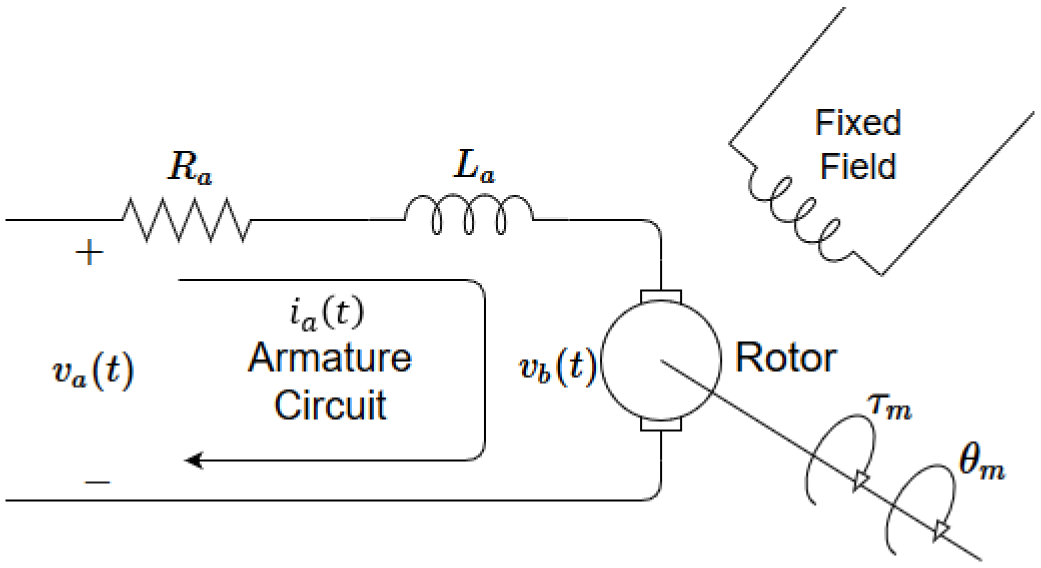


Figure 11: Circuit Diagram of an Armature Controlled DC Motor

$v_a(t)$ = armature voltage

$i_a(t)$ = armature current

R_a = armature resistance

L_a = armature inductance

$v_b(t)$ = back emf

θ_m = rotor position (radians)

τ_m = torque generated

The differential equation for the armature current is then

$$L_a \frac{di_a}{dt} + R_a i_a = v_a(t) - v_b(t) \tag{4.39}$$

The torque available for actuation is equal to the motor torque τ_m minus the torque delivered to the load τ_l with gear ratio r. The equation of motion for this system is then:

$$J_m \frac{d^2\theta_m}{dt^2} + B_m \frac{d\theta_m}{dt} = \tau_m - \tau_l/r \tag{4.40}$$

In Laplace domain, the Equation (4.38), (4.39) and (4.40) may be combined and written as:

$$(L_a s + R_a) I_a(s) = V_a(s) - K_e s \Theta_m(s) \tag{4.41}$$

$$(J_m s^2 + B_m s)\Theta_m(s) = K_t I_a(s) - T_l(s)/r \tag{4.42}$$

The block diagram of the above system is shown in Figure 12. The transfer function between the rotational position $\Theta_m(s)$ and armature voltage $V_a(s)$ is given by (at $\tau_l = 0$):

$$\frac{\Theta_m(s)}{V_a(s)} = \frac{K_m}{s[(L_a s + R_a)(J_m s + B_m) + K_b K_m]} \tag{4.43}$$

In SI units, $K_e = K_t = K$. Thus,

$$\frac{\Theta_m(s)}{V_a(s)} = \frac{K}{s[(L_a s + R_a)(J_m s + B_m) + K^2]} \tag{4.44}$$

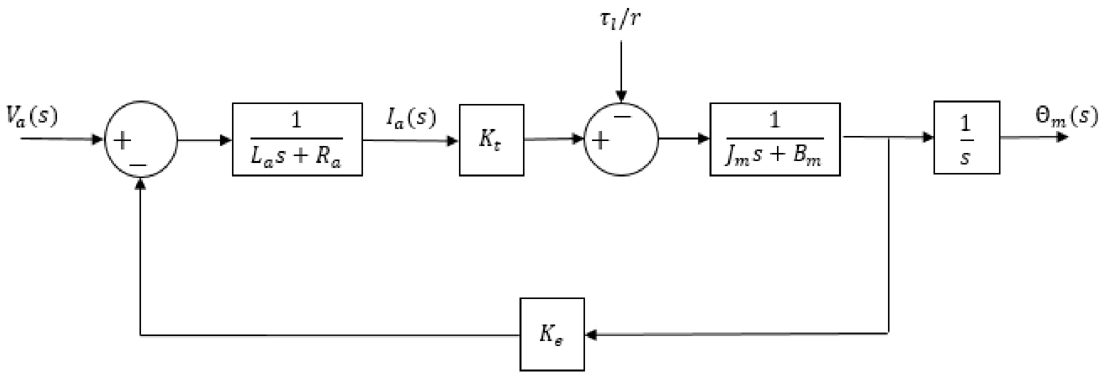


Figure 12: Block Diagram for a DC Motor System

The transfer function between the load torque $T_l(s)$ and rotor position $\Theta_m(s)$ is given by (at $V_a = 0$):

$$\frac{\Theta_m(s)}{T_l(s)} = \frac{-(L_a s + R_a)}{s[(L_a s + R_a)(J_m s + B_m) + K^2]} \tag{4.45}$$

Using Simscape library in MATLAB Simulink, the model of motor is created as shown in the Figure 13 below.

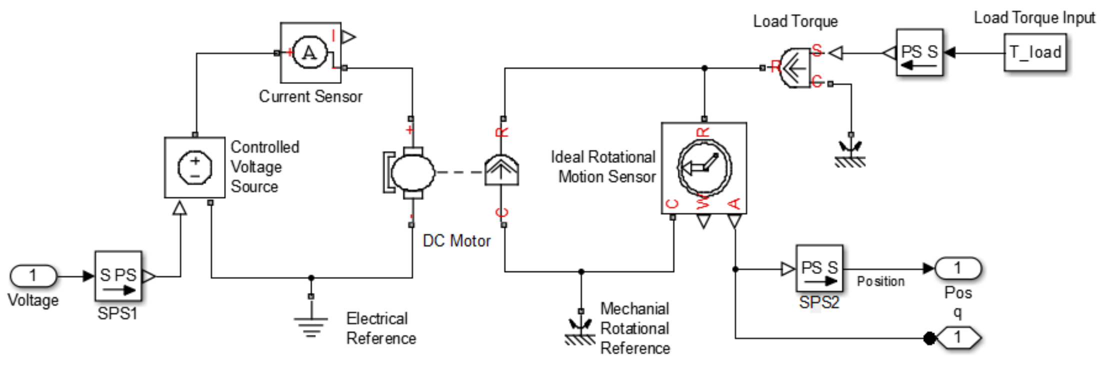


Figure 13: DC Motor Subsystem in Simulink using Simscape Library

Hitec HS-645MG and HS-65MG servo motors are selected for joint and wrist actuation of Dagu 5-DOF robotic arm. The Model parameterization of DC Motor in Simulink is done by providing the *stall torque and no-load speed* values option. General specifications of both Hitec high torque servo motors are listed in Table 3.

Table 3: General Specifications of Hitec Servo Motors

Parameters	***Specifications***	
	HS-645MG	***HS-65MG***
Operating Voltage	4.8V ~ 6.0V	4.8V ~ 6.0V
Stall Torque	7.7 kg.cm ~ 9.6 kg.cm	1.8 kg.cm ~ 2.2 kg.cm
No-load Current	350mA ~ 450mA	180mA ~ 220mA
No-load Speed	0.24sec/60° ~ 0.2sec/60°	0.14sec/60° ~ 0.11sec/60°
Weight	55.2g	11.9g

Chapter 5

Adaptive Neuro-Fuzzy Inference System

5.1 Introduction

Among various combinations of methodologies in soft computing, the one that has highest visibility at this juncture is that of fuzzy logic and neuro-computing, leading to neuro-fuzzy systems. Adaptive neuro-fuzzy inference system (ANFIS) is an intelligent modelling and control approach that combines the fuzzy logic [42], [43] with the field of artificial neural networks [44]. It is one of the most well-known and widely applied neuro-fuzzy architecture developed by Jang and Sun [5], [45]. The fundamentals of neural networks, fuzzy logic and control and neuro-fuzzy system are discussed here.

5.2 Artificial Neural Networks

Artificial neural networks (ANNs) is a machine learning technique inspired by biological neural networks which is used to approximate functions that are dependent on large number of input data. ANNs is an interconnected group of nodes that are composed of solely two elements operating in parallel as shown in Figure 14. The processing elements are called *neurons* and the connections are termed *synapses*. Generally, a processing elements has many inputs and a single output as shown in Figure 15. Each input to neuron has an associated weight. If the sum of all the weighted inputs are above a certain threshold, that neuron is activated. Typically, neural networks are adjusted, or trained, so that a particular input leads to a specific target output. The weights of the neurons are adjusted during the learning process to minimize the error objective function as shown in Figure 16.

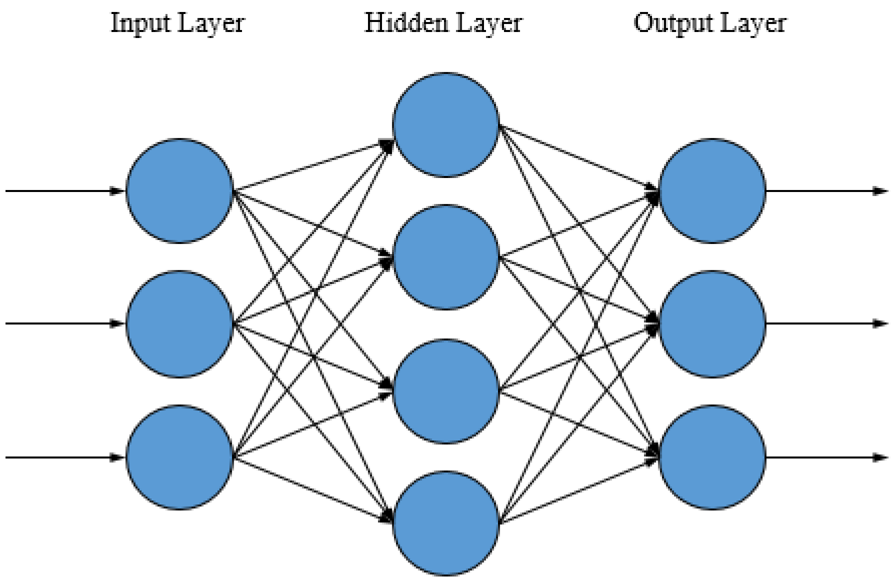


Figure 14: Artificial Neural Network (ANN)

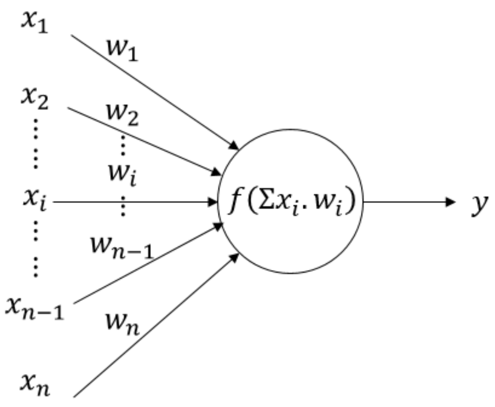


Figure 15: Artificial Neuron and Synapses

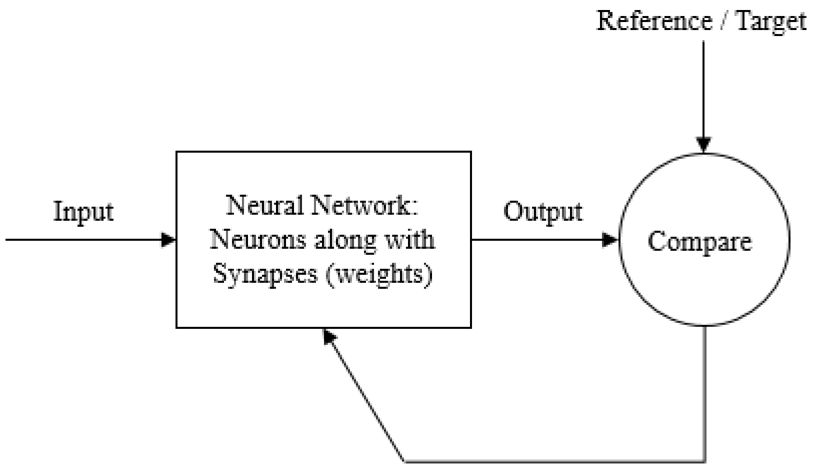


Figure 16: ANN Training Cycle

5.3 Fuzzy Logic

Fuzzy logic is a convenient way to map an input space to an output space. Fuzzy logic works on the principle of working with variables in linguistic terms instead of having a definite single value. It is a method of characterizing knowledge in terms of fuzzy sets and a rule base. Figure 17 shows the basic block diagram of a fuzzy logic system. Fuzzy logic is synonymous with the theory of fuzzy sets [46]. A fuzzy set is a set without a clearly defined boundary. It can contain elements with only a partial degree of membership. The primary mechanism of fuzzy sets is to make a list of *if-then* statements called *rules* which are evaluated in parallel. *Fuzzy Inference System* (FIS) interprets the values in the input vector and, based on these set of rules, assigns values to the output vector. The major advantage of fuzzy control is its flexibility of defining relationship between different physical quantities in different number of inputs and outputs which is done by assigning membership values (or degree of membership) between 0 and 1 using a defined curve known as *Membership Function*.

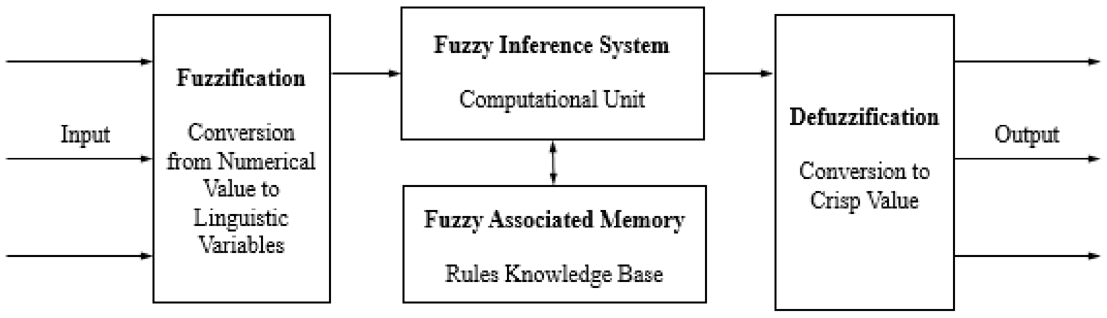


Figure 17: Components of a Fuzzy Logic System

There are two popular types of FIS: *Mamdani* and *Sugeno*. *Mamdani's* method was among the first control systems built using fuzzy set theory. It was proposed in 1975 by Ebrahim Mamdani [47]. Mamdani-type inference outputs membership functions as fuzzy sets of each variable which further need diffuzification to get a crisp value. *Sugeno* or Takagi-Sugeno-Kang, method of fuzzy inference [48] was introduced in 1985. It is similar to the Mamdani method in many respects. The first two parts of the fuzzy inference process, fuzzifying the inputs and applying the fuzzy operator, are exactly the same. The main difference between Mamdani and Sugeno is that the Sugeno output membership functions are either linear or constant. Some advantages of using Sugeno method are that it is computationally efficient, works well with linear (e.g., PID control), optimization and adaptive techniques and it has guaranteed continuity of the output surface.

A typical rule in a Sugeno fuzzy model has the form

If *Input* 1 $= x$ and *Input* 2 $= y$, then *Output* is $z = ax + by + c$.

For a zero-order Sugeno model, the *output level* z is a constant ($a = b = 0$).

The *output level* z_i of each rule is weighted by the *firing strength* w_i of the rule. For example, for an AND rule with *Input* 1 $= x$ and *Input* 2 $= y$, the firing strength is

$$w_i = AndMethod\big(F_1(x), F_2(y)\big)$$

where F_1 *and* F_2 are the *membership functions* for Inputs 1 and 2. The final output of the system is the weighted average of all rule outputs, computed as

$$Final\ Output = \frac{\sum_{i=1}^{N} w_i z_i}{\sum_{i=1}^{N} w_i}$$

Where N is the number of rules. A Sugeno rule operates as shown in the following Figure 18.

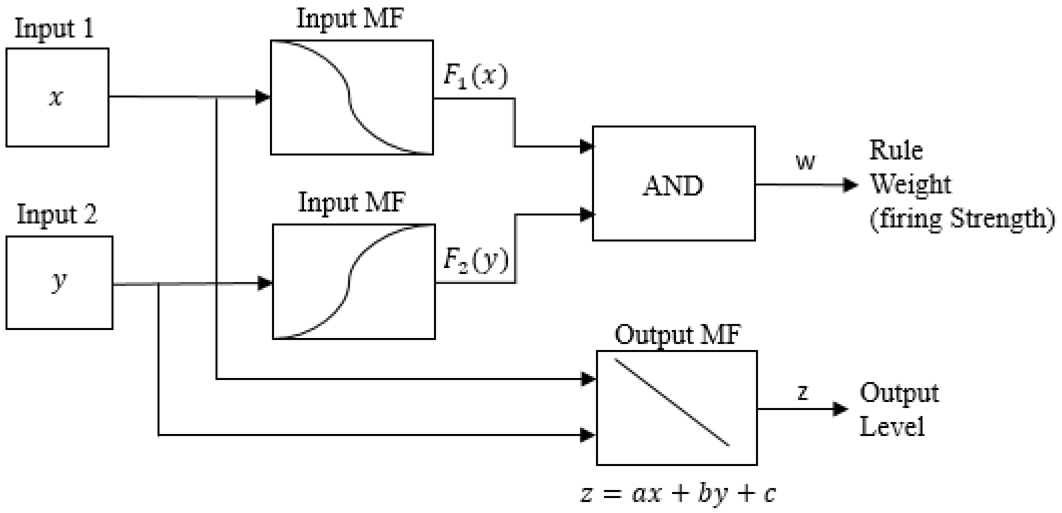


Figure 18: Sugeno Type Inference

Because of the linear dependence of each rule on the input variables, the Sugeno method is ideal for acting as an interpolating supervisor of multiple linear controllers that are to be applied, respectively, to different operating conditions of a dynamic nonlinear system. Typical output surface plot for the sugeno type inference is shown below in Figure 19.

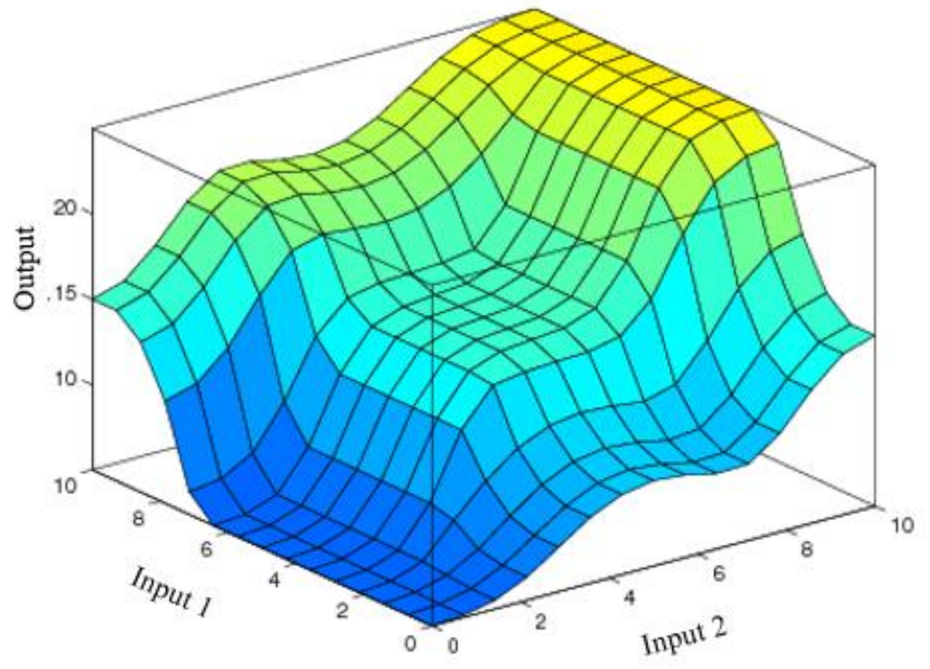


Figure 19: Output Surface Plot of Sugeno Type Inference

5.4 Adaptive Neuro-fuzzy Learning

Neuro-fuzzy learning system uses fuzzy logic along with neuro-adaptive learning scheme. Neuro-adaptive learning works similar to neural networks and provides a method for fuzzy modeling procedure to learn information about the given input data set. Fuzzy logic part computes the membership function parameters that allow the FIS to properly map the input/output data. These membership functions and associated parameters (weights) are tuned / adjusted using the neural network techniques such as backpropagation algorithm and least squares type methods. This adjustment allows fuzzy system to learn from the data they are modeling. The basic ANFIS model structure in fuzzy logic toolbox of MATLAB is shown in Figure 20.

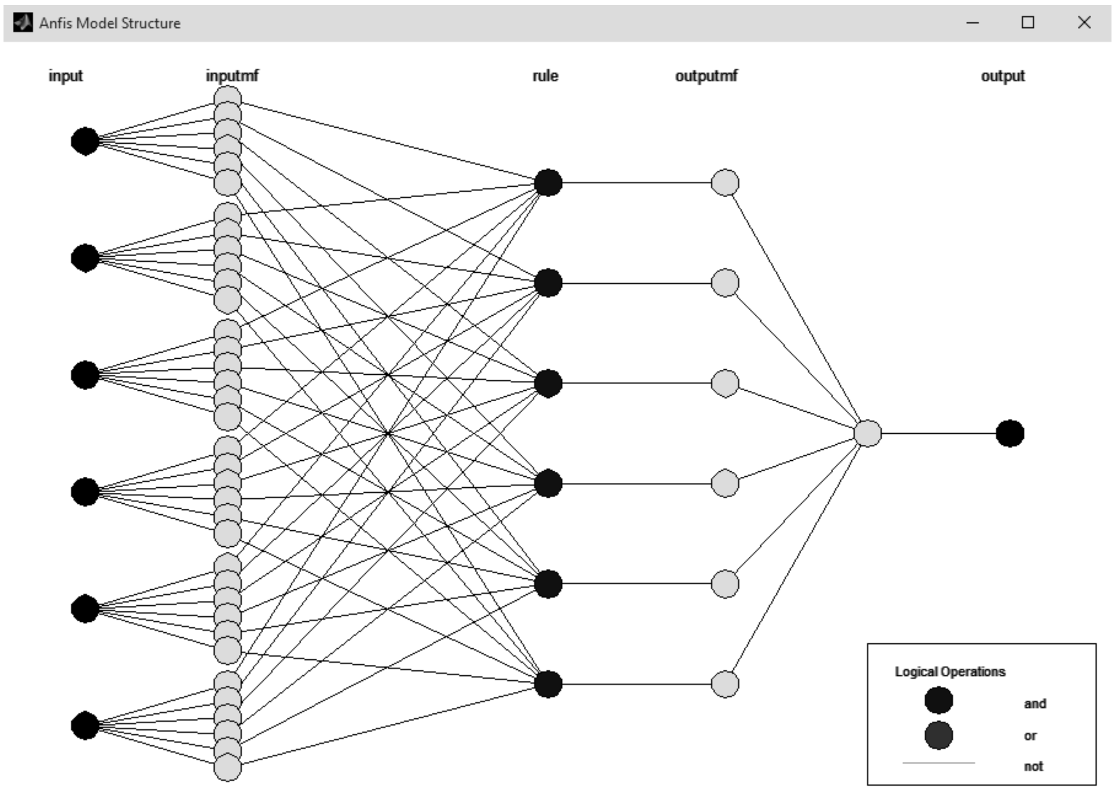
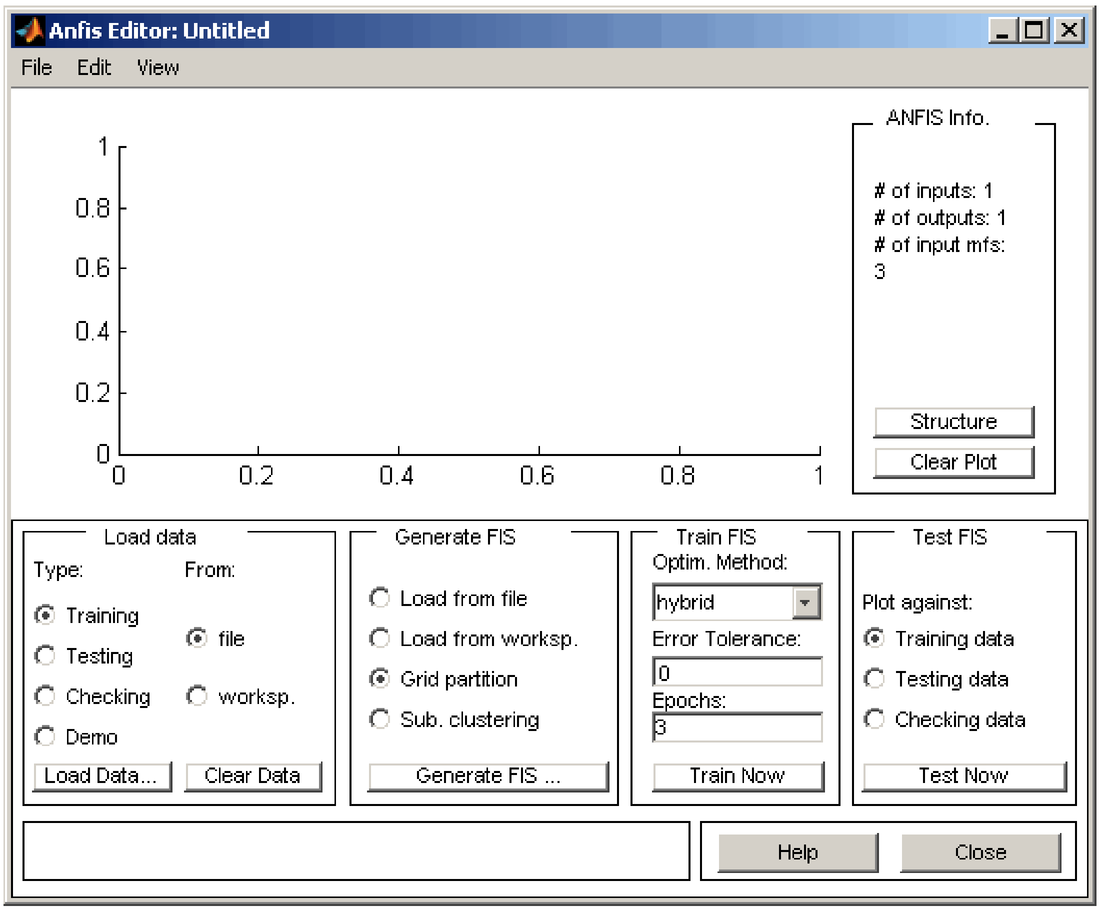


Figure 20: ANFIS Model Structure

The modeling approach used by ANFIS is similar to many system identification techniques. First the parameterized model structure is theorized (relating inputs to membership functions to rules to outputs to membership functions, and so on). Then the input/output data is loaded to ANFIS for *training*. Then the FIS model is trained to emulate the training data that was provided by modifying the membership function parameters according to a choose *error criterion*. Then *model validation* is done in which the input/output data set on which the FIS was not trained, is presented to FIS model to judge how well the FIS model predicts the corresponding dataset output values. All these stages are accomplished in ANFIS editor in MATLAB [49], [50] as shown in Figure 21.



ANFIS Info.

# of inputs: 1
# of outputs: 1
# of input mfs:
3

Load data

Type: [✓ Training] Testing Checking Demo

From: [✓ file] worksp.

Generate FIS

Load from file Load from worksp. [✓ Grid partition] Sub. clustering

Train FIS

Optim. Method: hybrid

Error Tolerance: 0

Epochs: 3

Test FIS

Plot against: [✓ Training data] Testing data Checking data

Figure 21: ANFIS Editor

5.5 Data Clustering

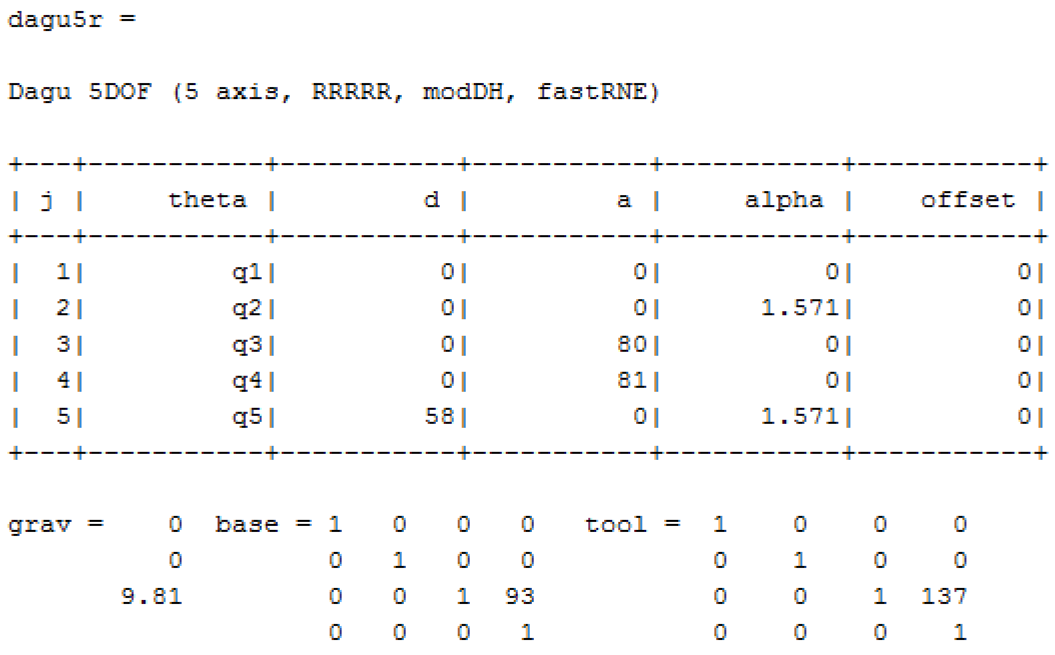
Clustering of numerical data forms the basis of many classification and system modeling algorithms. The purpose of clustering is to identify natural groupings of data from a large data set to produce a concise representation of a system's behavior. Data clustering is performed on the training data to find clusters in input-output data set. This cluster information is then used to generate a Sugeno-type fuzzy inference system that best models the data behavior using a minimum number of rules. The rules partition themselves according to the fuzzy qualities associated with each of the data clusters. There are two techniques of data clustering available in fuzzy logic toolbox of MATLAB, *Fuzzy C-Mean Clustering* and *Subtractive Clustering*. Fuzzy c-means (FCM) is a data clustering technique wherein each data point belongs to a cluster to some degree that is specified by a membership grade [51]. It provides a method that shows how to group data points into a specific number of different clusters. If it is not clear that how many clusters should be created for a given data set then Subtractive clustering [52] is recommended for estimating the number of clusters in a data set. In MATLAB, *genfis2* command generates a Sugeno-type FIS structure using subtractive clustering by extracting a set of rules that models the data behavior. While, genfis3 command generates a FIS using FCM clustering technique.

Chapter 6

Simulation and Results

6.1 Robotics Toolbox

Robotics Toolbox [53]–[55] for MATLAB is used to solve the forward kinematics of Dagu 5-DOF robotic arm. The toolbox provides various functions that are useful for the study and simulation of classical arm-type robotics, for example such things as kinematics, dynamics and trajectory generation. The serial-link arm-type robot is represented by Toolbox in terms of Denavit-Hartenberg parameters as shown in the Figure 22 for Dagu 5-DOF. Various pose configurations are graphically illustrated in the Figure 23.



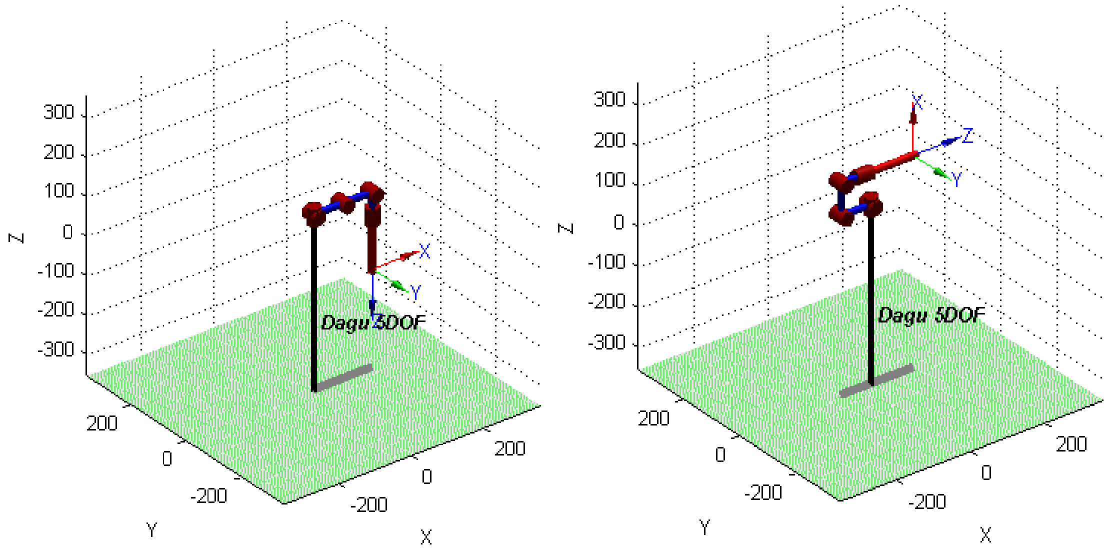
```
dagu5r =

Dagu 5DOF (5 axis, RRRRR, modDH, fastRNE)

+---+-----------+-----------+-----------+-----------+-----------+
| j |     theta |         d |         a |     alpha |    offset |
+---+-----------+-----------+-----------+-----------+-----------+
|  1|         q1|          0|          0|          0|          0|
|  2|         q2|          0|          0|      1.571|          0|
|  3|         q3|          0|         80|          0|          0|
|  4|         q4|          0|         81|          0|          0|
|  5|         q5|         58|          0|      1.571|          0|
+---+-----------+-----------+-----------+-----------+-----------+

grav =     0  base = 1   0   0   0   tool =  1    0    0    0
           0         0   1   0   0           0    1    0    0
        9.81         0   0   1  93           0    0    1  137
                     0   0   0   1           0    0    0    1
```

Figure 22: SerialLink Description of Dagu 5-DOF

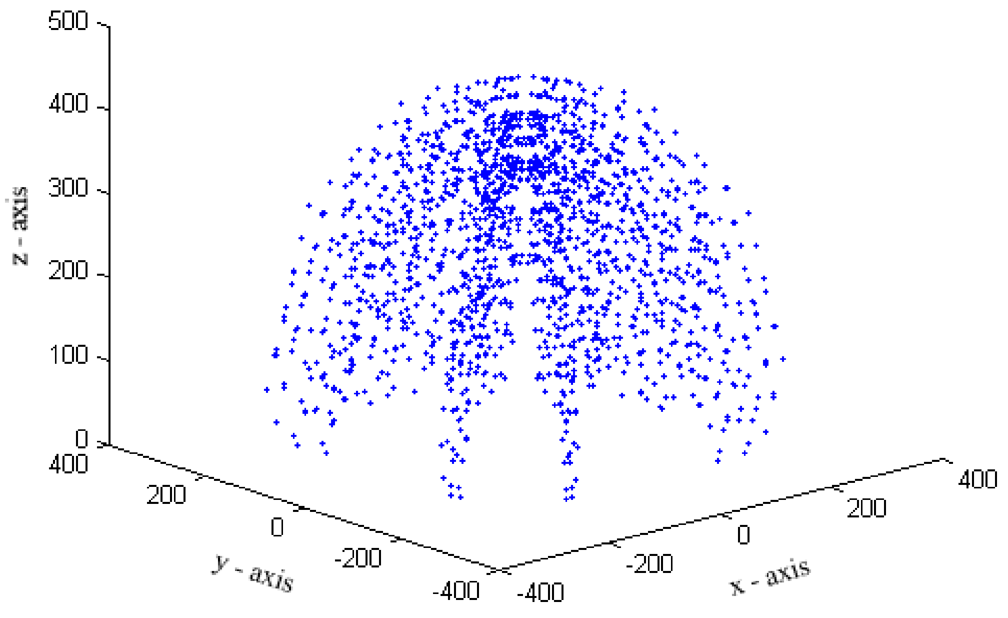


(a) (b)

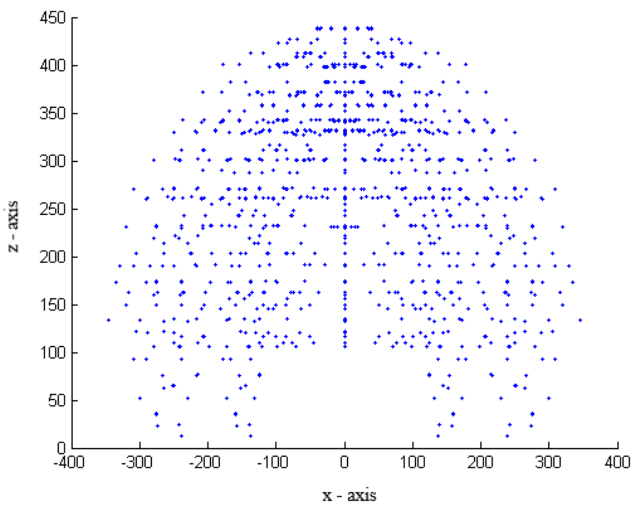
Figure 23: Dagu 5-DOF in different poses (a) zero angle (b) ready pose

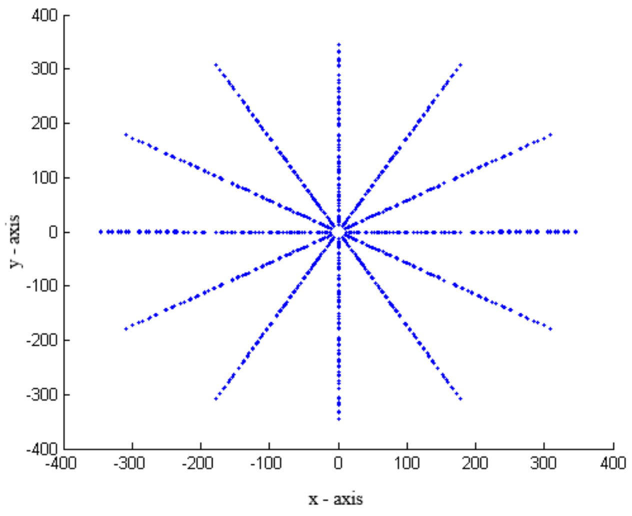
6.2 Workspace Sampling

In order to obtain the training data, the workspace mapping that relates the input joint variables to end-effectors position and orientation in Cartesian space is done by computing the forward kinematics that was formulated in Chapter 4 for the joint intervals stated in Table 2. The sampled Cartesian space for the permissible joint space shown in Figure 24 which accounts for various key problems involved in solving the inverse kinematics through neuro-fuzzy learning, namely the problems of generating and preprocessing training data, handling multiple solutions, reducing the approximation error, and lowering the training time as discussed in [56].



(a)





(b) (c)

Figure 24: Workspace Sampling of Forward Kinematics (a) Isometric View (b) X-Z View (c) X-Y View

6.3 ANFIS Implementation

The coordinates (x, y, z) and the orientation angles (α, β, γ) obtained from forward kinematics solutions are used as the input for training data to ANFIS network with the triangular membership function with a hybrid learning algorithm (back propagation + least square method). For the neuro-fuzzy model used in this work, 13433 data points were analytically obtained using forward kinematics which are shown in the Figure 24, as discussed in the previous section, of which 10075 data points are used for training, 1679 data points are used as model validation and 1679 data points are used as test case for predicting the untrained data output values .For predicting the inverse kinematics solution of Dagu 5-DOF robotic arm five independent ANFIS structures with first order Sugeno model are selected in MATLAB with specific parametric configurations as listed in the Table 4.

Table 4: ANFIS Configuration Parameters for Solving Dagu 5DOF

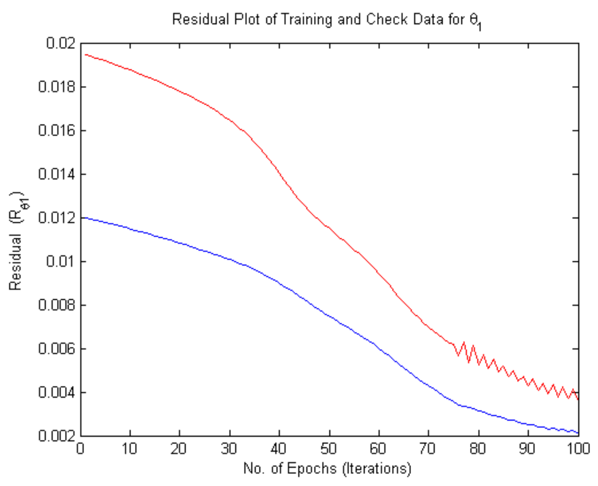
Parameter Name	**ANFIS Description** (for all Joints)
type	sugeno’
andMethod	‘prod’
orMethod	‘max’
defuzzMethod	‘wtaver’
impMethod	‘prod’
aggMethod	‘max’
input	1x6 $(x, y, z, \alpha, \beta, \gamma)$
output	1x1 $(Joint\ Angle\ \theta_i)$
rule	1x216
No. of Training Data Pairs	10075
No. of Checking Data Pairs	1679

6.3.1 Residual Plot of Training Data and Check Data for all Joint angles of Dagu 5-DOF

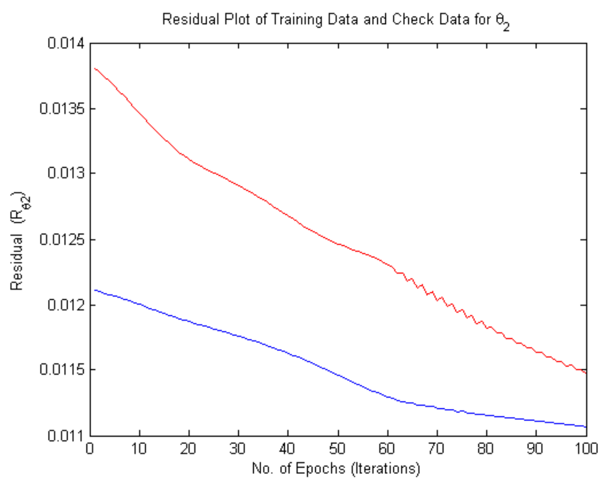
The residual plots of training data for $\theta_1, \theta_2, \theta_3, \theta_4$ and θ_5 of Dagu 5-DOF robotic arm are depicted in Figure 25. The residual plot in all five case is low which suggests that ANFIS is successfully converging to fit the data and since both the residual of training data (blue line) and check data (red line) are close enough this suggest that the FIS has avoided over fitting the data and it serves as an indication that the ANFIS will predict the new unknown data with good accuracy and minimal error.

6.3.2 Prediction of Inverse Kinematics for all Joints Angles of Dagu 5-DOF and Comparison with Analytical Solution

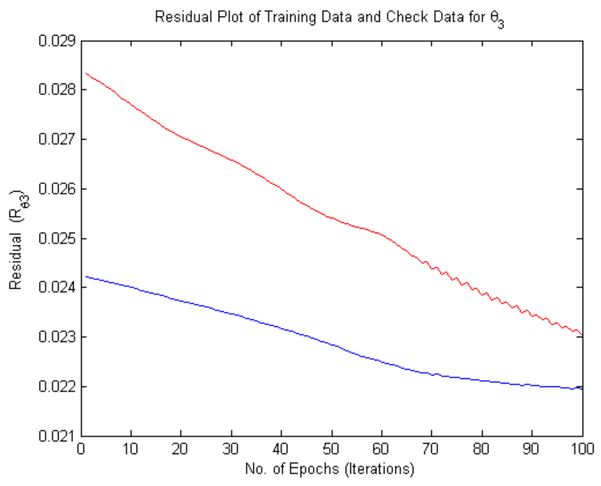
Having trained the networks, an important follow up step is to validate the networks to determine how well the ANFIS networks would perform inside the larger control system. The trained ANFIS for all 5 joint angles is then tested against the newly presented data set using the *evalfis* command in MATLAB. The output predicted by *evalfis* is then compared with the analytical inverse kinematic solution presented in Section 4.3 of Chapter 4. Figure 26 shows how close the FIS predicted outputs are with respect to the deducted values.



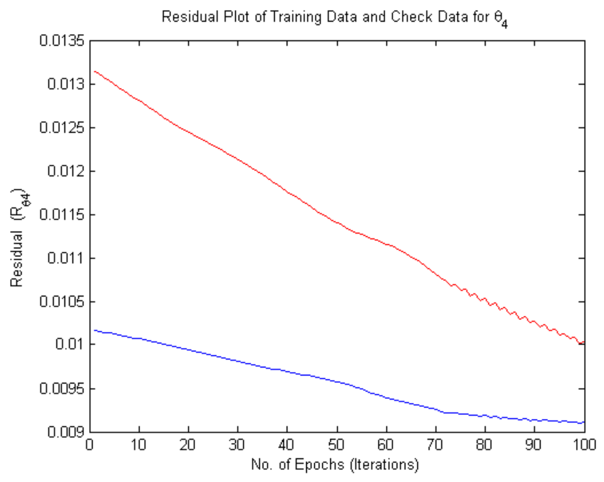
(a)



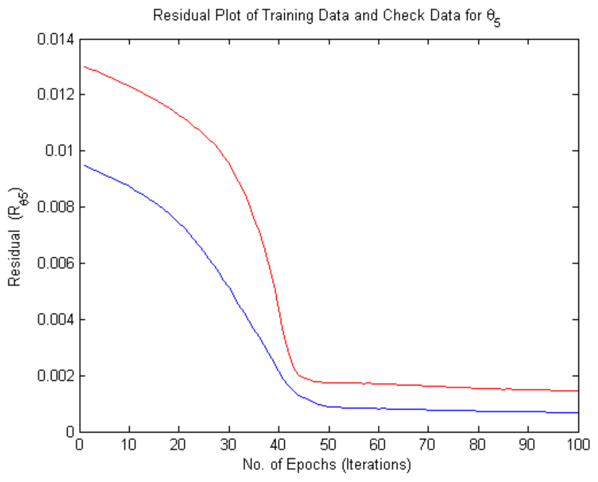
(b)



(c)

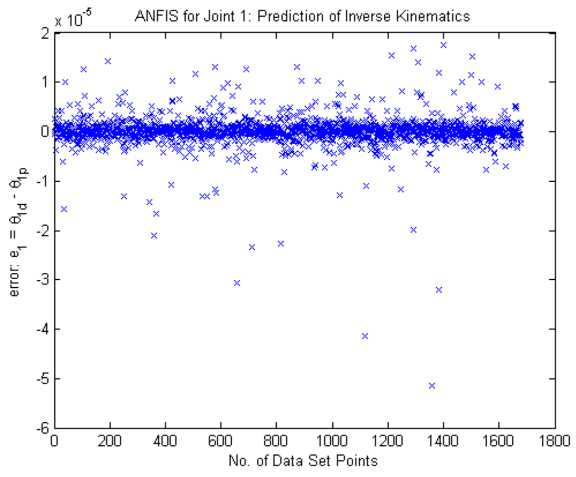


(d)

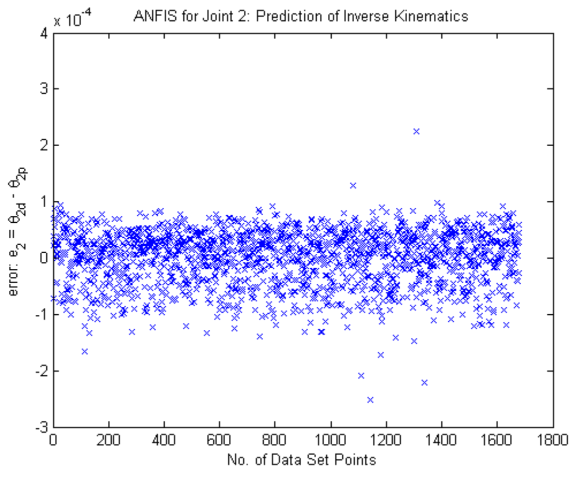


(e)

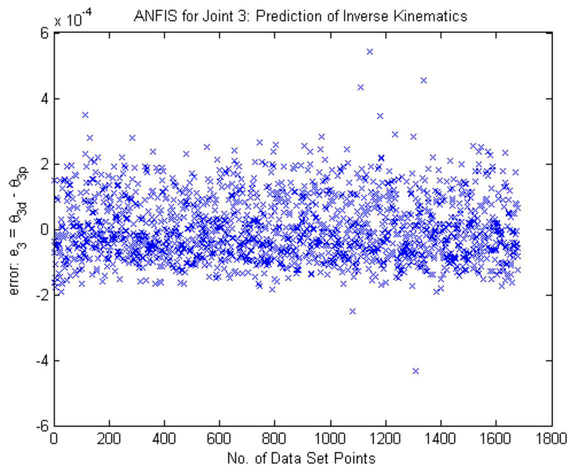
Figure 25: Residual Plots of Training Data and Check Data for all Joint Variables



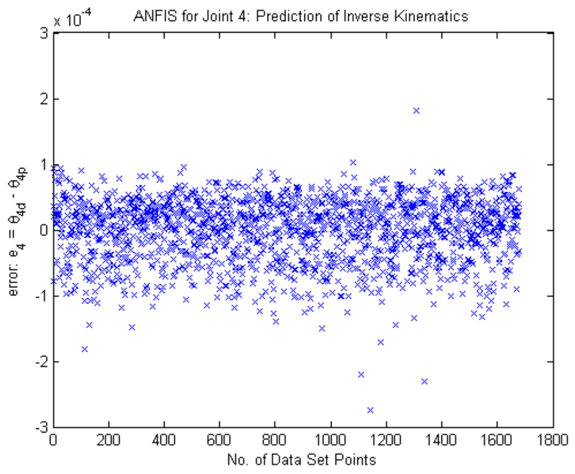
(a)



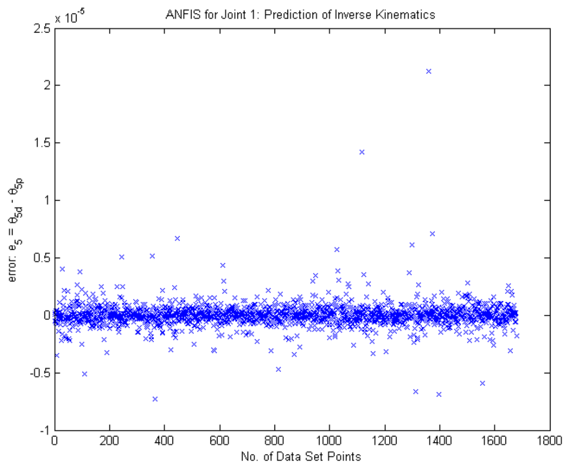
(b)



(c)



(d)



(e)

Figure 26: Model Validation of ANFIS networks for Dagu 5-DOF

6.4 Robot Dynamics in Simulink / SimMechanics

SimMechanics software is a block diagram modeling environment for the engineering design and simulation of rigid multibody machines and their motions, using the standard Newtonian dynamics of forces and torques. 3D CAD model import of robotic arm has already been discussed in chapter 3. One of the most common requirement in robotics is to move the end-effector smoothly from pose A to pose B in a straight line. There exists two main approaches to generate such trajectories: *joint space* motion and *Cartesian space* motion.

Joint space motion planning is easier in the sense as it involves interpolation between two configurations of joint variables θ_i and then by applying forward kinematics the end-effectors position and orientation can easily be determined for all interpolated configurations between the initial joint configuration θ_{i_A} and the final joint configuration θ_{i_B}. But the problem with this method is that it doesn't ensures the motion of end-effector to follow the desired path. This is to be expected since only the Cartesian coordinates of the end-points were specified. As the robot rotates about its waist joint during the motion the end-effector will naturally follow a circular arc. In practice this could lead to collisions between the robot and nearby objects even if they do not lie on the path between poses A and B.

For many applications straight-line motion in Cartesian space is required for which the second strategy is required. That means, for the robot's end-effector to follow a desired path, necessary joints angles are required to be calculated by computing the inverse kinematics at every specified interval.

In this study, the trajectory tracking of the Dagu 5-DOF robotic arm's end effector is considered in the SimMechanics environment using the Cartesian motion scheme. As a test case scenario, the reference trajectory used to evaluate the performance of the overall methodology is shown in Figure 27.

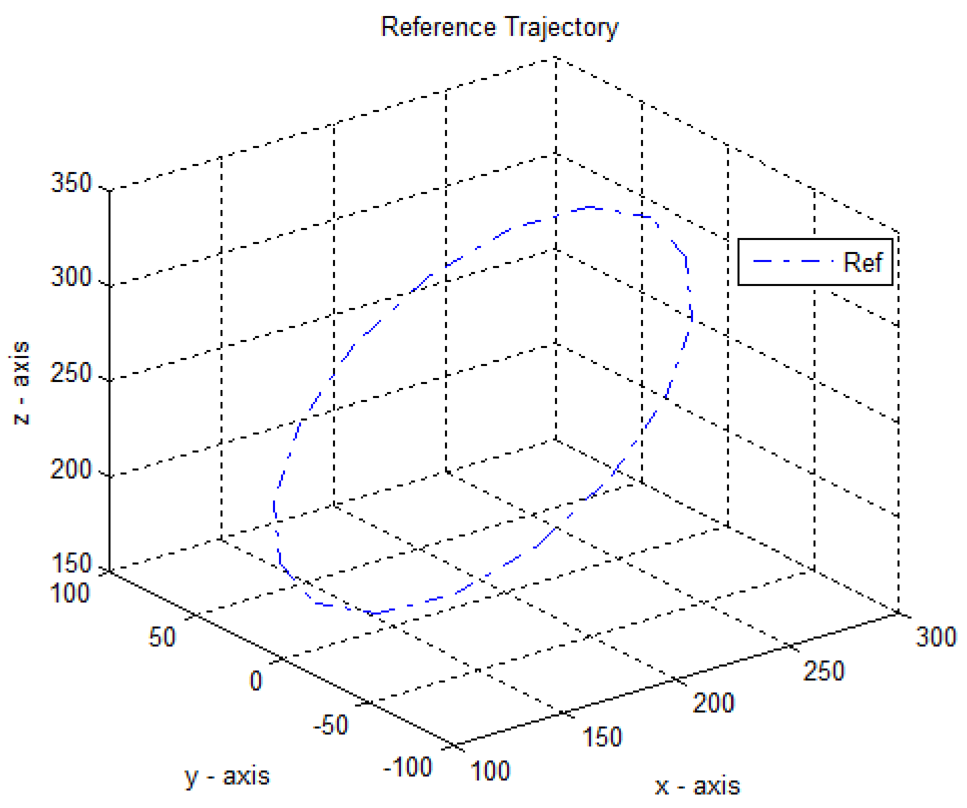


Figure 27: Reference Trajectory for Dagu 5-DOF Robotic Arm's End-Effector

6.4.1 Simulink / SimMechanics Model

The complete implementation in Simulink / SimMechanics environment is shown in Figure 28. The reference trajectory generated with the aid of robotics toolbox for MATLAB is fed into ANFIS network that computes the desired joint angles for all five joint variables of the robotic arm as shown in Figure 29. The outputs of ANFIS networks are then handed over to the servo-controllers which in turn actuate each joint of the SimMechanics multibody model of Dagu 5-DOF robotic arm. Decentralized PID control scheme is used to control all the joint angles independently as shown in Figure 30. Figure 31 shows the Simulink implementation of the PID controller.

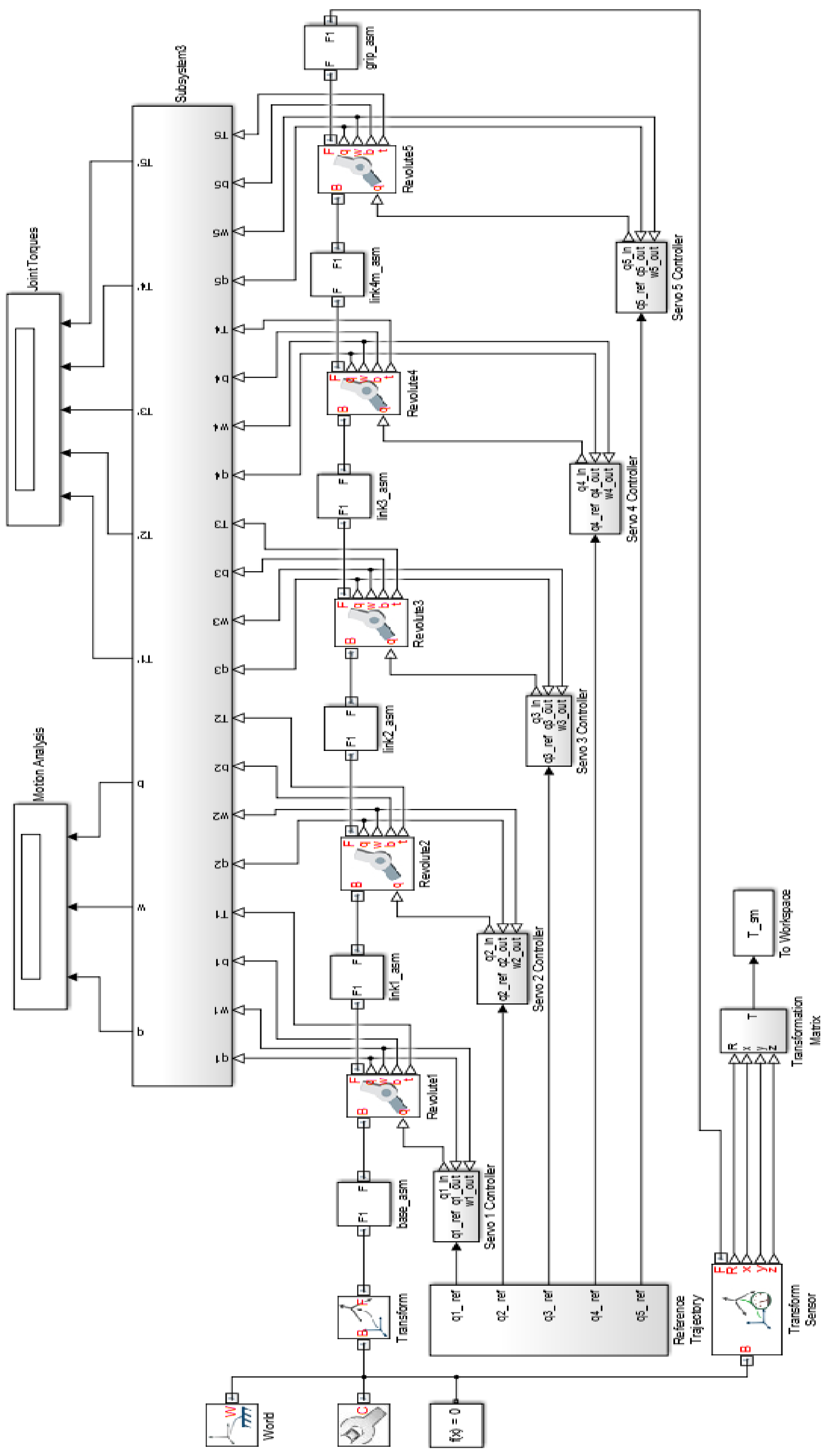


Figure 28: Simulink Model for Trajectory Tracking of Robotic Arm

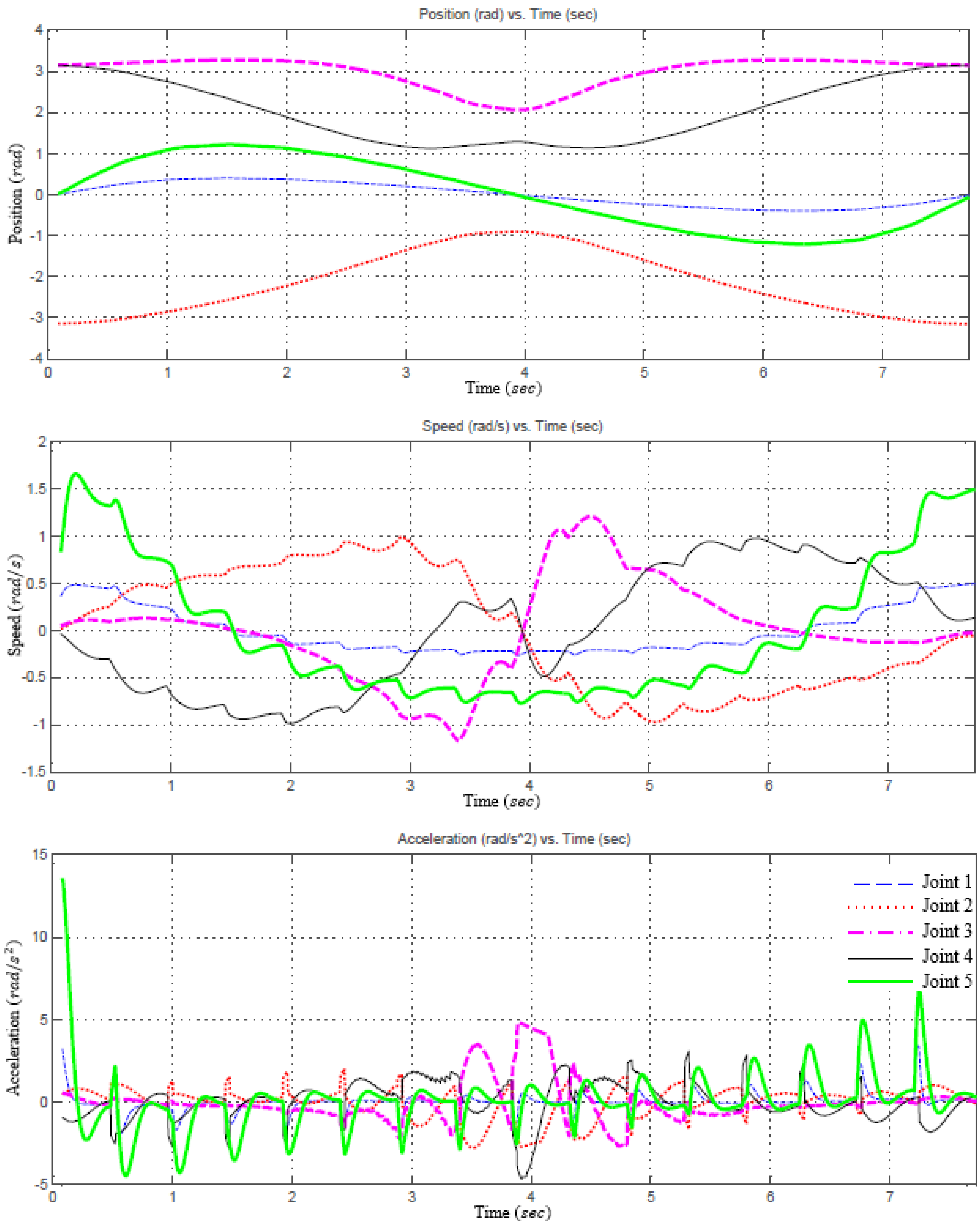


Figure 29: Reference Trajectory Generated from Robotics Toolbox

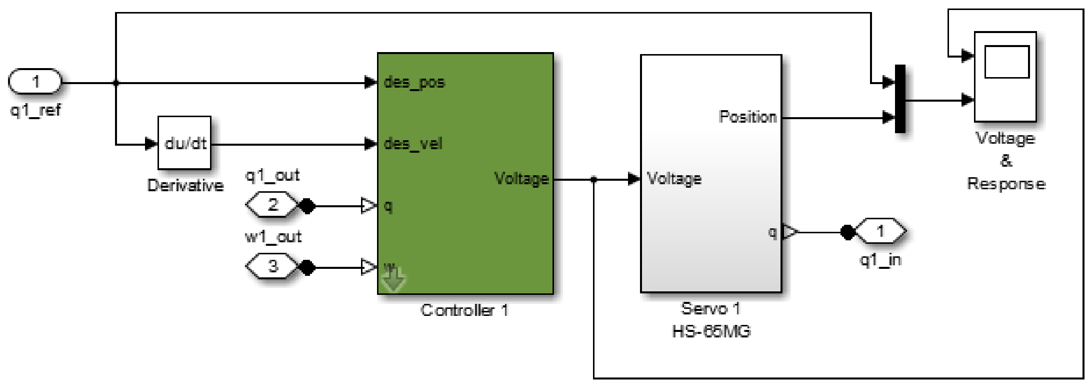


Figure 30: Simulink Subsystem for Individual Joint Control

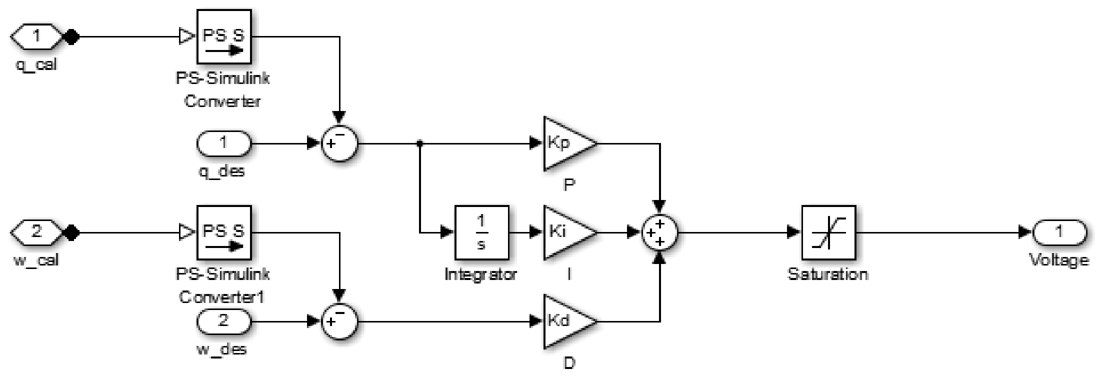


Figure 31: PID Controller Implementation in Simulink

6.4.2 Simulation Results of Robot Dynamics

The inverse kinematics is computed for the end-effector’s desired trajectory using the same ANFIS networks that are trained for entire workspace of Dagu 5-DOF discussed in the previous sections. Figure 32 shows the trajectory followed by the robotic arm in SimMechanics environment by running the simulation based on the results obtained from the ANFIS networks.

The control effort and the position comparisons of all five joints controlled by PID controllers can be seen in Figure 33 to Figure 37. SimMechanics provides various types of force and torque sensing for driving individual joint primitives. The torques computed by SimMechanics multibody dynamics for individual joints are shown in Figure 38.

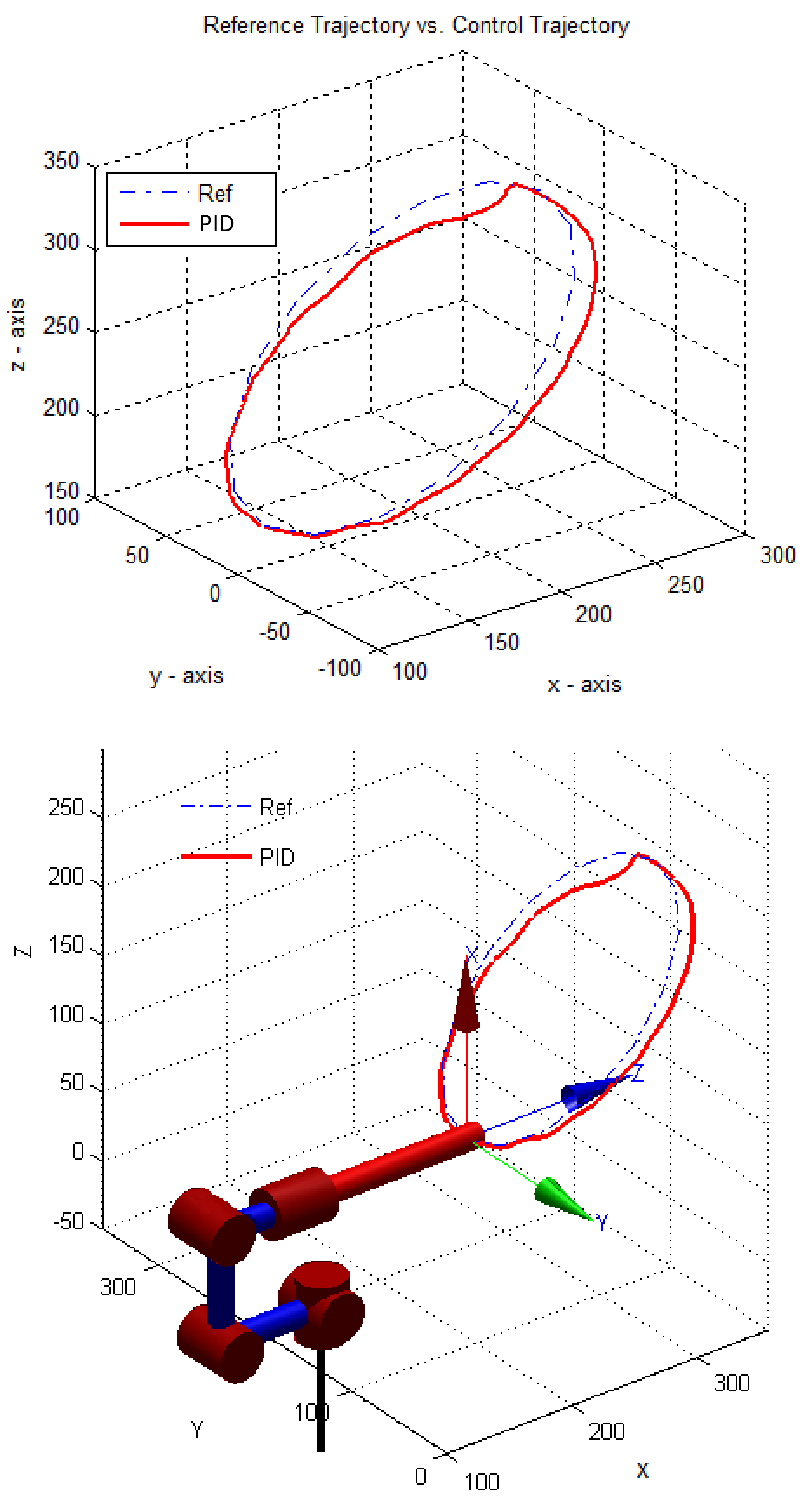
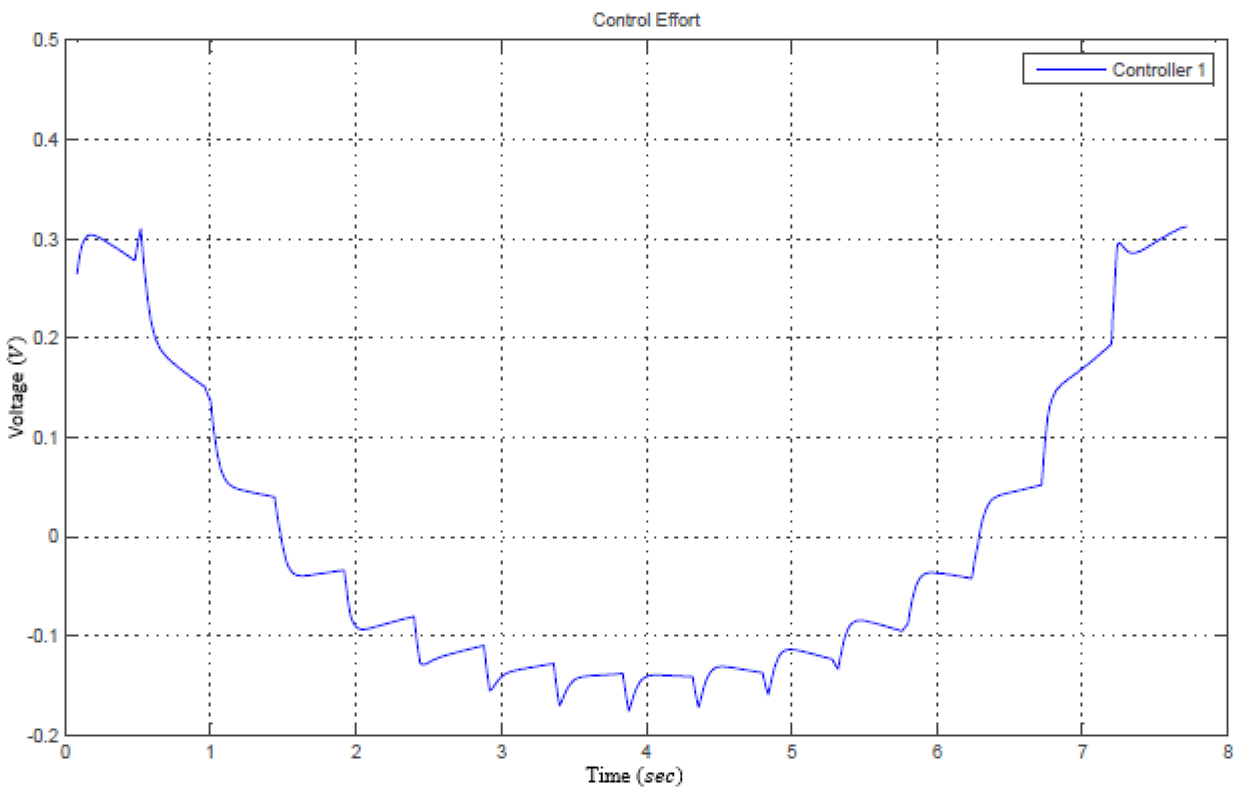


Figure 32: Trajectory Tracking by Dagu 5-DOF Robotic Arm



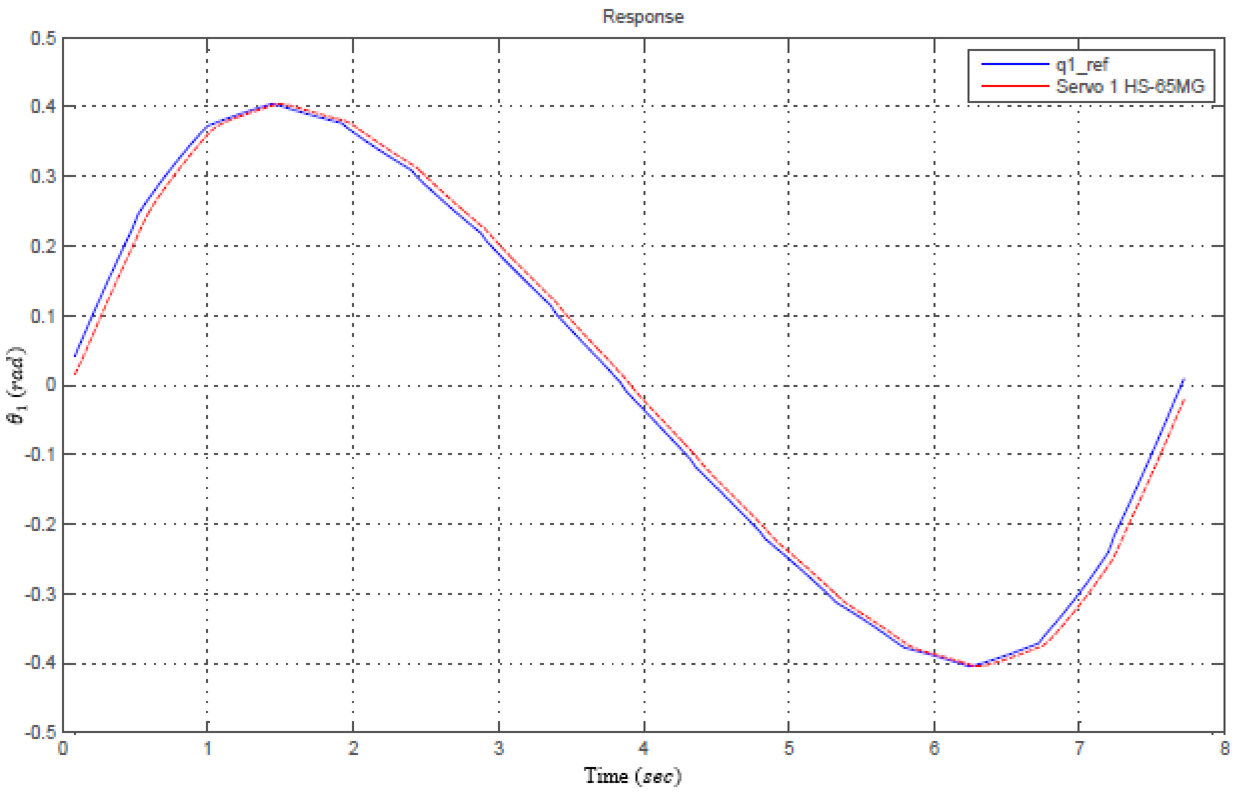
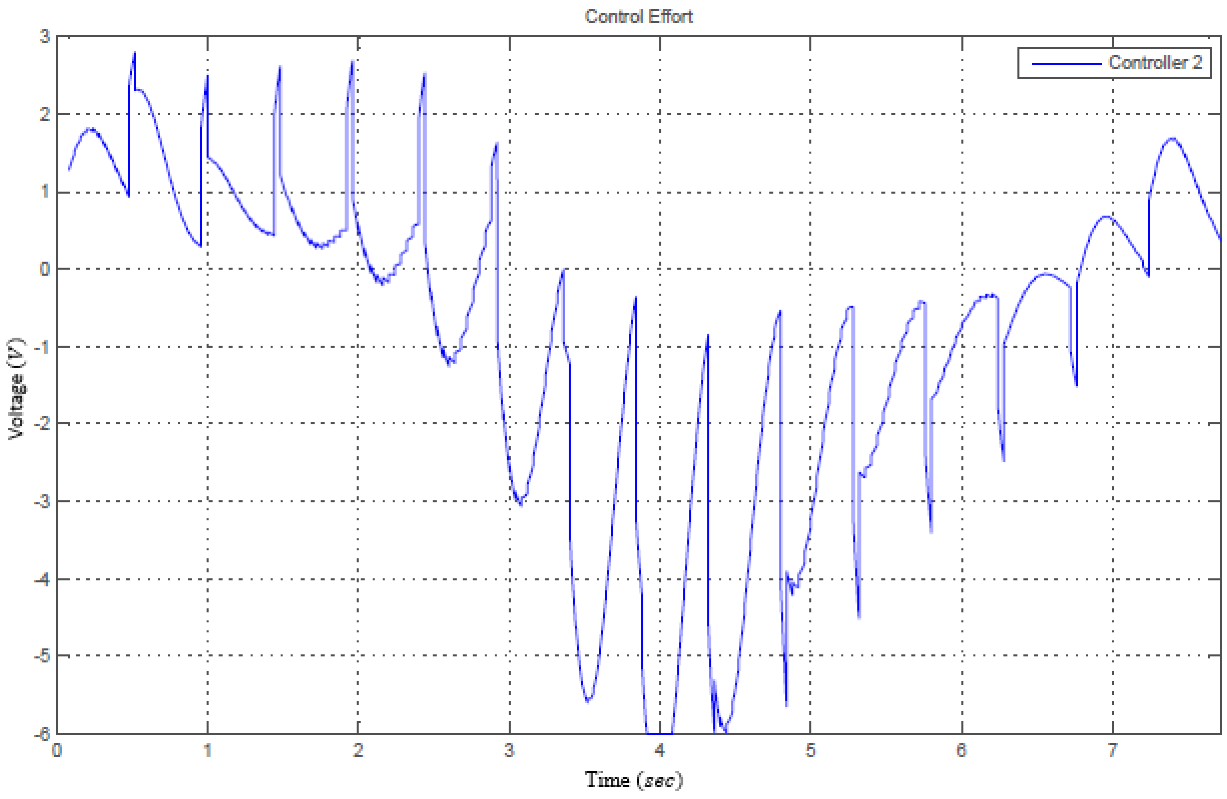


Figure 33: Control Effort and Position Comparison for Joint Variable θ_1



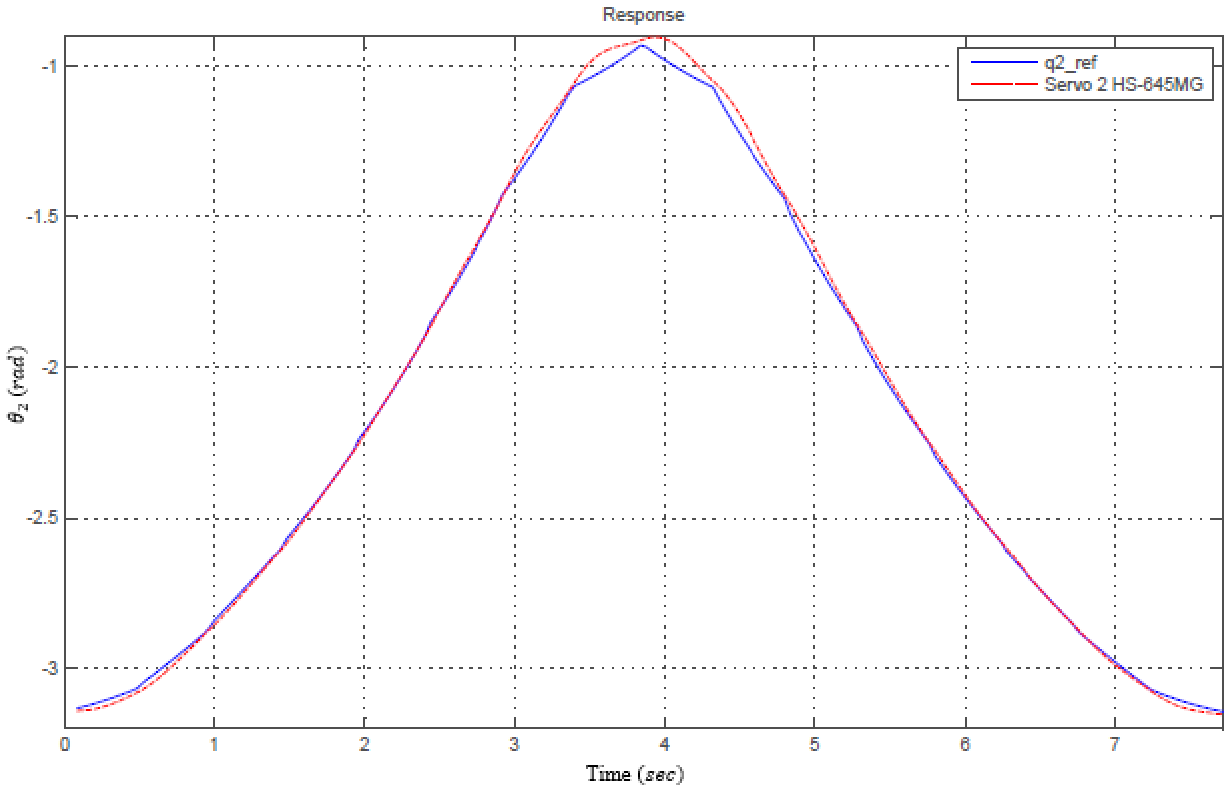
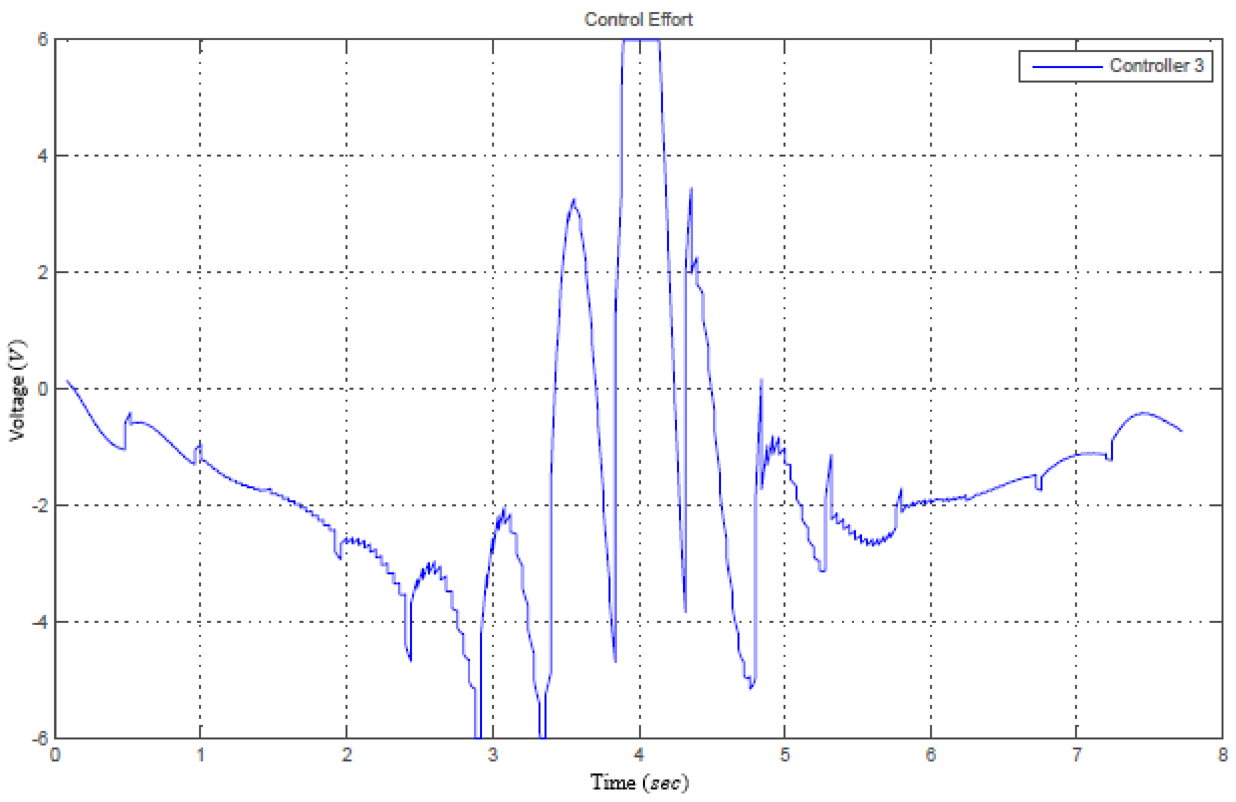


Figure 34: Control Effort and Position Comparison for Joint Variable θ_2



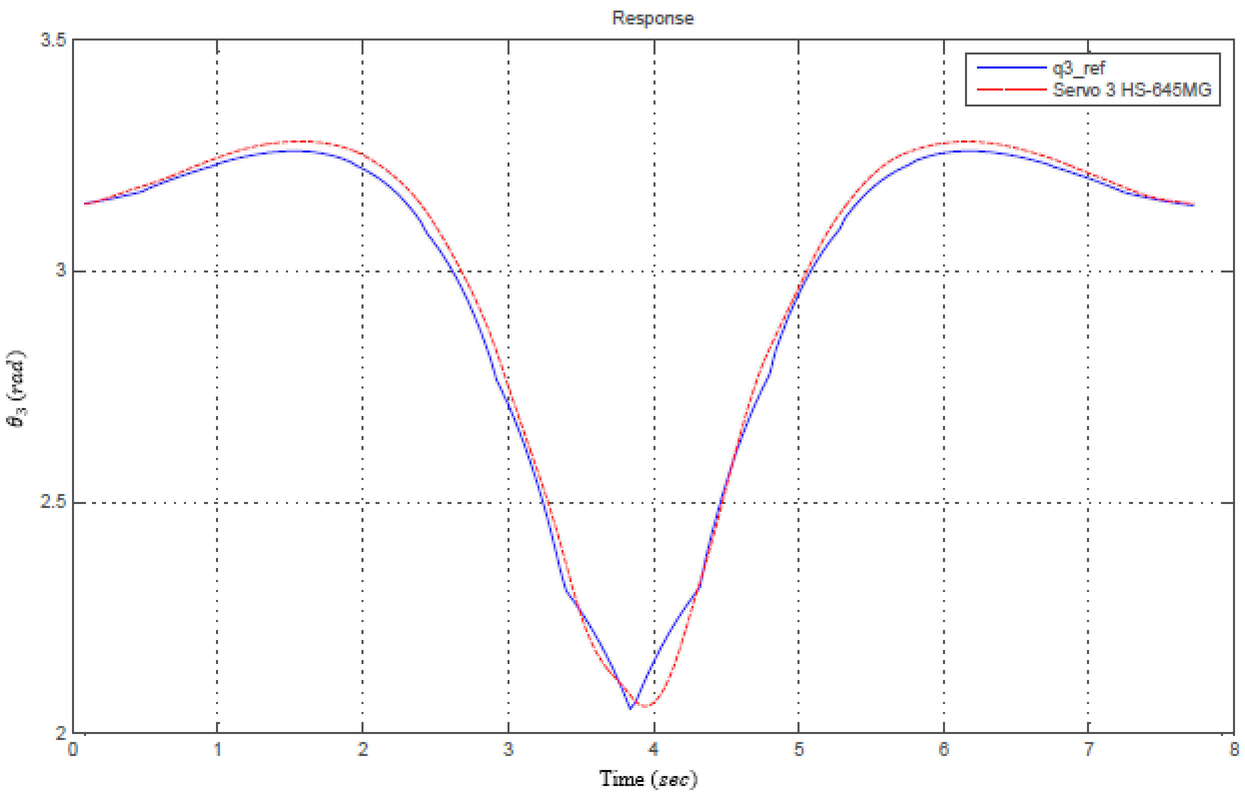


Figure 35: Control Effort and Position Comparison for Joint Variable θ_3

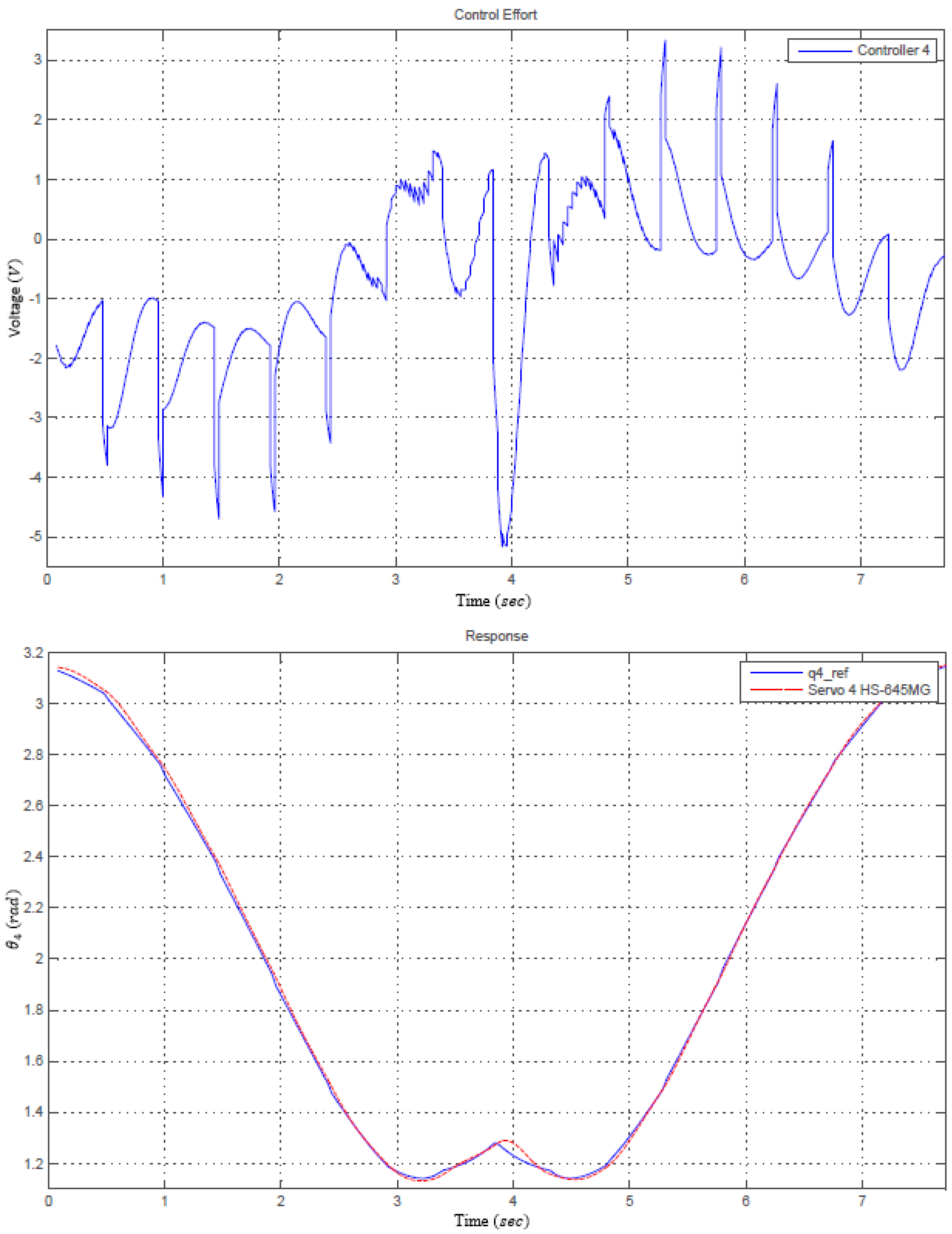
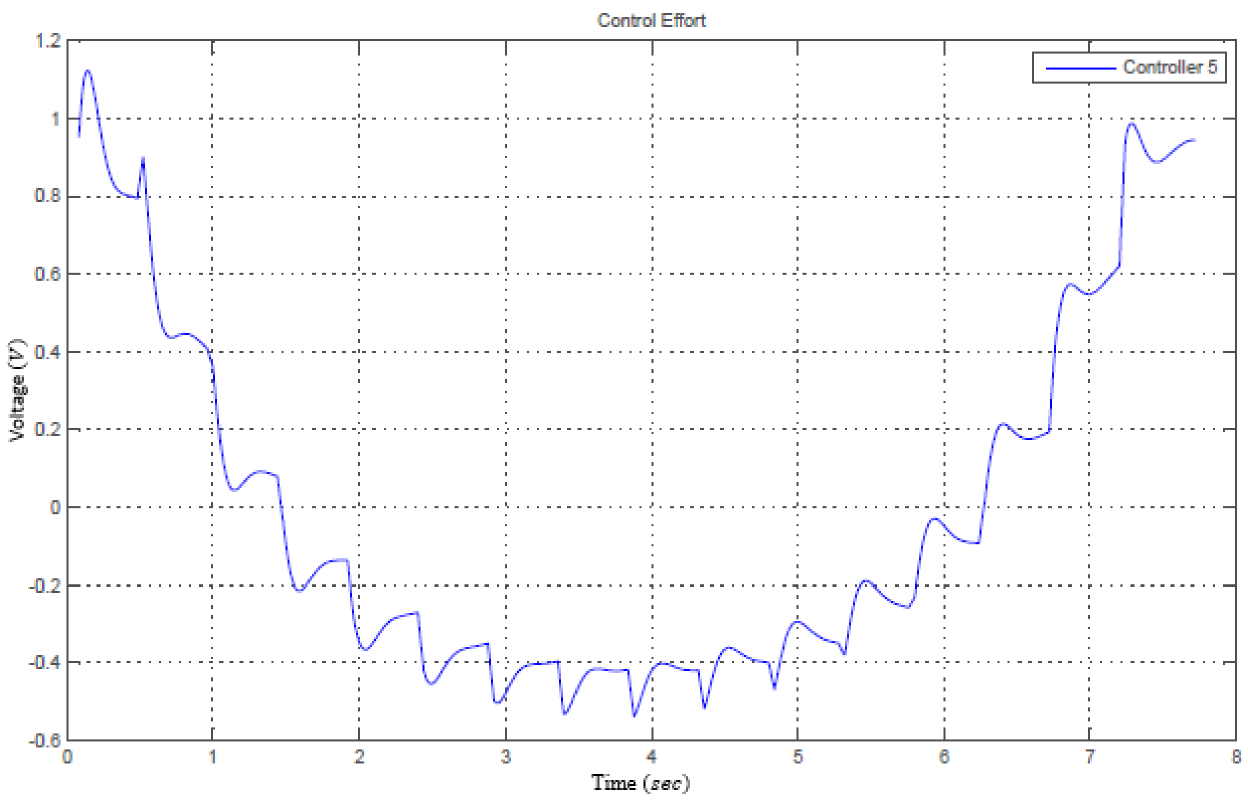


Figure 36: Control Effort and Position Comparison for Joint Variable θ_4



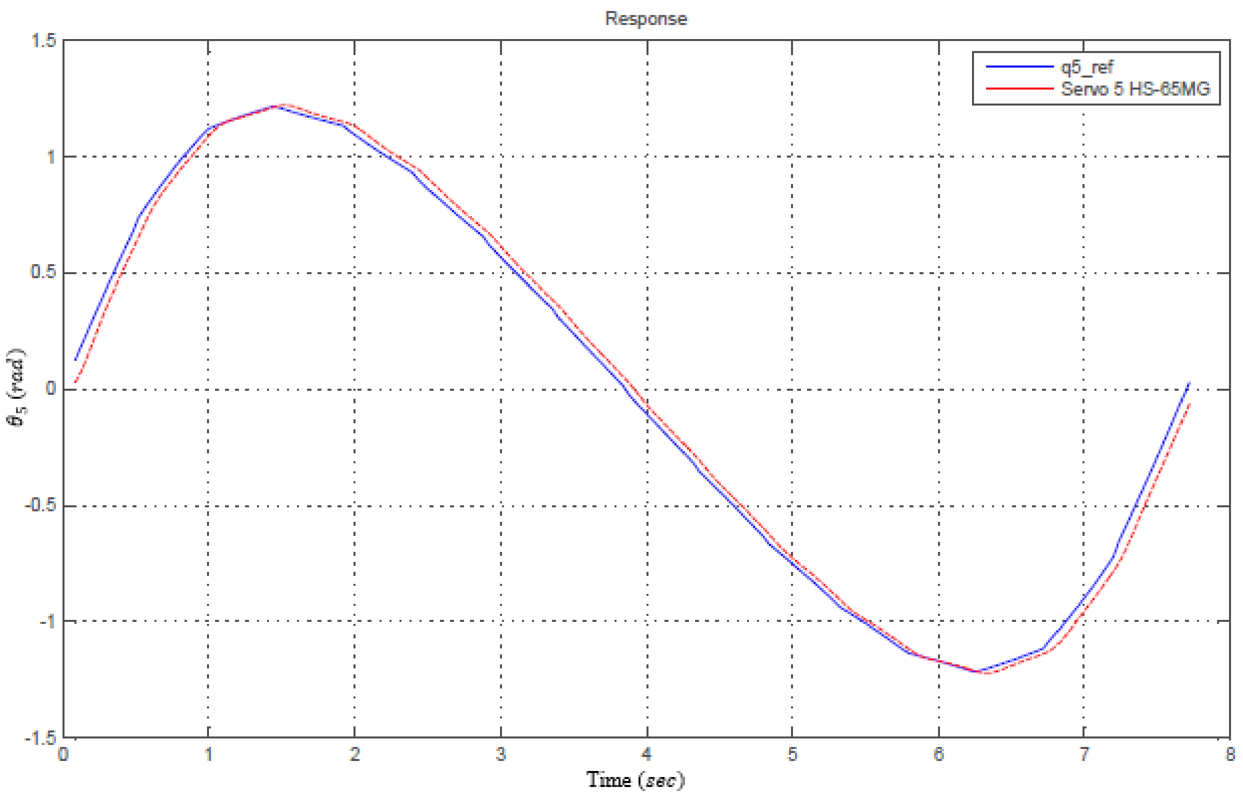


Figure 37: Control Effort and Position Comparison for Joint Variable θ_5

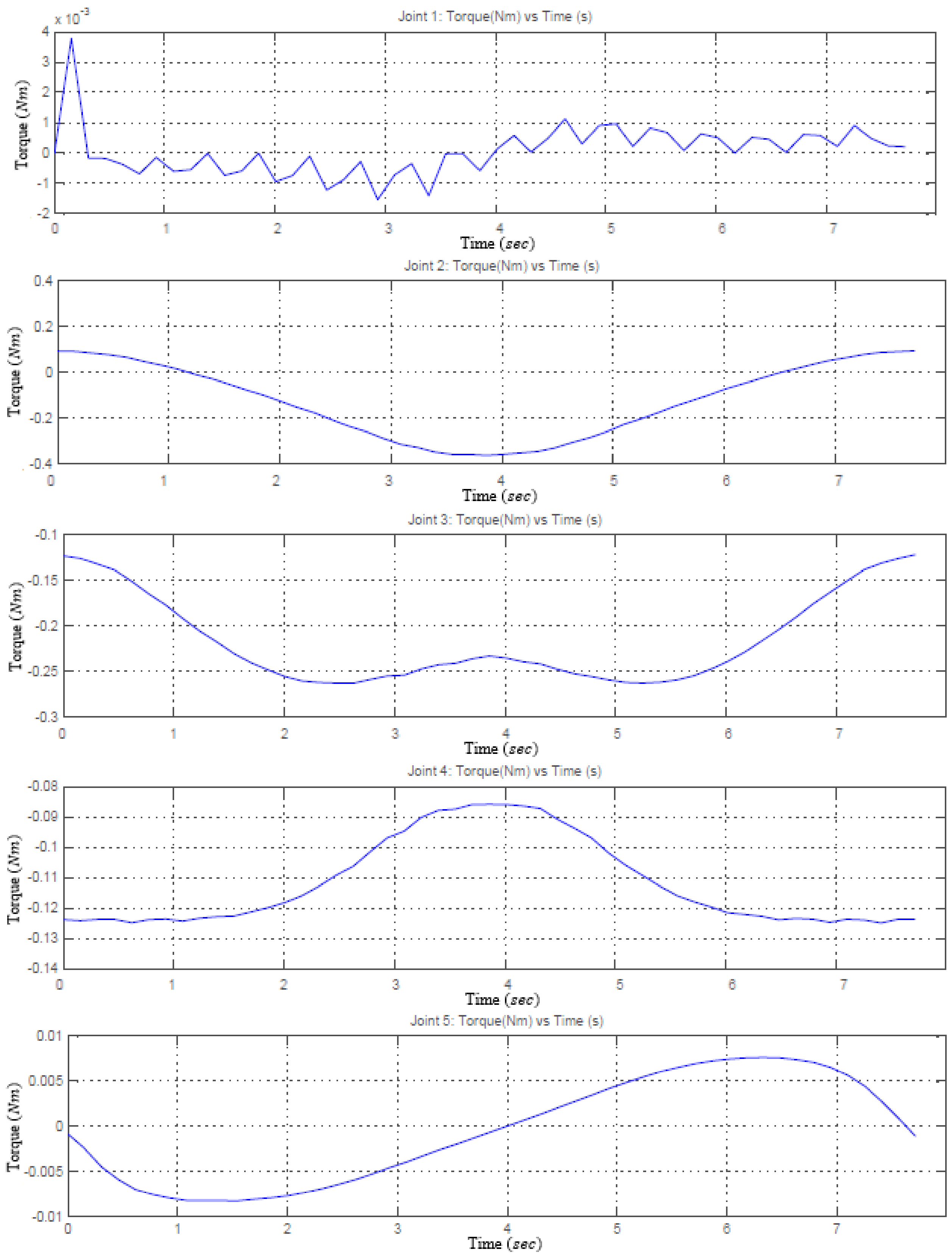


Figure 38: Joint Torques Computation of Dagu 5-DOF

Chapter 7

Conclusion and Future Recommendations

SimMechanics presents a powerful tool for modeling mechanics of rigid bodies. It is suitable for modeling of dynamics and kinematics of considerably complicated systems with many joints without using any mathematical description. For these advantageous properties it is often used in the first phase of designing robotic systems, esp. due to simplicity of changing parameters and dimensions of particular bodies without necessity to repeat design of new model.

In this thesis, five ANFIS networks were trained to emulate control of a robotic arm. The fuzzification of neural networks' inputs / outputs allow the system to learn more complex functions than ever before as encountered in this thesis for solving inverse kinematics. From the results of Chapter 6 it can be seen that for the successful implementation of ANFIS, number of training examples, iterations and membership functions are very crucial factors.

At present time the developed model serves for further research – forward kinematics, analysis of dynamics and design of controllers for joint drives. In the proposed design, only off-line simulation were conducted which can be extended to an on-line controlling of the serial robot manipulator. The work done in this thesis was based on having PID control structure. Since the system is nonlinear, it would be interesting to apply nonlinear controllers such sliding mode control [57], [21], [58] algorithms. Alternate control schemes like LQR control [59]–[61], repetitive control [62]–[65], H-infinity (H-∞) control [66], etc. can be implemented for performance comparisons. Also, programming some heuristic rules into the neuro-fuzzy inference for the robotic arm could improve performance, which is left for future implementation. Future work in this area should also look into determining the adequacy of training samples. Are there enough training samples? Are all areas of the state space represented? A study of the relationship of the training samples and the fuzzy membership functions would be particularly helpful.

References

[1] B. Siciliano, L. Sciavicco, L. Villani, and G. Oriolo, *Robotics - Modelling, Planning and Control*. Springer London, 2009.

[2] Arexx Engineering, “ROBOT ARM PRO V3: Documentation.” [Online]. Available: http://www.arexx.com/robot_arm/html/en/documentation.htm.

[3] V. Fedak, F. Ďurovsky, and R. Üveges, “Analysis of Robotic System Motion in SimMechanics and MATLAB GUI Environment,” in *MATLAB Applications for the Practical Engineer,* InTech, 2014.

[4] F. Berardi, M. Chiaberge, E. Miranda, L. M. Reyneri, D. Elettronica, and P. Torino, “A Neuro-Fuzzy Systems for Control Applications,” in *International Symposium on Neuro-Fuzzy Systems,* 1996.

[5] J.-S. R. Jang, “ANFIS: adaptive-network-based fuzzy inference system,” *IEEE Trans. Syst. Man Cybern.*, vol. 23, no. 3, pp. 665–685, 1993.

[6] T. C. Manjunath, “Kinematic Modelling and Maneuvering of A 5-Axes Articulated Robot Arm,” *Int. J. Mech. Aerospace, Ind. Mechatron. Manuf. Eng.*, vol. 1, no. 4, pp. 216–222, 2007.

[7] S. Elgazzar, “Efficient kinematic transformations for the PUMA 560 robot,” *IEEE J. Robot. Autom.*, vol. 1, no. 3, pp. 142–151, 1985.

[8] Q. Chen, S. Zhu, and X. Zhang, “Improved Inverse Kinematics Algorithm Using Screw Theory for a Six-DOF Robot Manipulator,” *Int. J. Adv. Robot. Syst.*, p. 1, 2015.

[9] H. Liu, W. Zhou, X. Lai, and S. Zhu, “An Efficient Inverse Kinematic Algorithm for a PUMA560-Structured Robot Manipulator,” *Int. J. Adv. Robot. Syst.*, p. 1, 2013.

[10] N. Chen and G. A. Parker, “Inverse kinematic solution to a calibrated Puma 560 industrial robot,” *Control Eng. Pract.*, vol. 2, no. 2, pp. 239–245, 1994.

[11] S. Kucuk and Z. Bingul, “Inverse kinematics solutions for industrial robot manipulators with offset wrists,” *Appl. Math. Model.*, vol. 38, no. 7–8, pp. 1983–

1999, 2014.

[12] S. (Kocaeli U. Küçük and Z. (Department of M. E. U. Bingül, “The Inverse Kinematics Solutions of Industrial Robot Manipulators,” *Mechatronics, 2004. ICM ’04. Proc. IEEE Int. Conf.*, pp. 274–279, 2004.

[13] I.-C. Ha, “Kinematic parameter calibration method for industrial robot manipulator using the relative position,” *J. Mech. Sci. Technol.*, vol. 22, pp. 1084–1090, 2008.

[14] M. Dahari and J. D. Tan, “Forward and inverse kinematics model for robotic welding process using KR-16KS KUKA robot,” in *2011 4th International Conference on Modeling, Simulation and Applied Optimization, ICMSAO 2011*, 2011.

[15] U. Abubakar, W. Zhongmin, and G. Ying, “Kinematic Analysis and Simulation of a 6-DOF Industrial Manipulator,” *Int. J. Sci. Res.*, vol. 3, no. 3, pp. 323–326, 2014.

[16] R. M. Murray, Z. Li, and S. S. Sastry, *A Mathematical Introduction to Robotic Manipulation*, vol. 29. 1994.

[17] C. Lee and M. Ziegler, “Geometric Approach in Solving Inverse Kinematics of PUMA Robots,” *IEEE Trans. Aerosp. Electron. Syst.*, vol. AES-20, no. 6, pp. 695–706, 1984.

[18] J. U. Korein and N. I. Badler, “TECHNIQUES FOR GENERATING THE GOAL-DIRECTED MOTION OF ARTICULATED STRUCTURES.,” *IEEE Computer Graphics and Applications*, vol. 2, no. 9. 1982.

[19] A. Zilouchian and D. Howard, “Application of Adaptive Neuro-Fuzzy Inference System to Robotics,” in *Intelligent Control Systems Using Soft Computing Methodologies*, 2001.

[20] S. Alavandar and M. J. Nigam, “Inverse kinematics solution of 3DOF planar robot using ANFIS,” *Int. J. Comp. Commun. Cont*, vol. III, no. 2008, pp. 150–155, 2008.

[21] A. F. Amer, E. A. Sallam, and W. M. Elawady, “Adaptive fuzzy sliding mode control using supervisory fuzzy control for 3 DOF planar robot manipulators,”

Appl. Soft Comput. J., vol. 11, no. 8, pp. 4943–4953, 2011.

[22] I. Branch, “A new method for position control of a 2-DOF robot arm using neuro – fuzzy controller,” *Indian J. Sci. Technol.*, vol. 5, no. 3, pp. 2253–2257, 2012.

[23] A.-V. Duka, “ANFIS Based Solution to the Inverse Kinematics of a 3DOF Planar Manipulator,” *Procedia Technol.*, vol. 19, pp. 526–533, 2015.

[24] W. E. K. III, “NEURO-FUZZY CONTROL OF A ROBOTIC ARM,” 1994.

[25] S. Kumar and K. Irshad, “Implementation of Artificial Neural Network applied for the solution of inverse kinematics of 2-link serial chain,” *Int. J. Eng. Sci. Technol.*, vol. 4, no. 09, pp. 4012–4024, 2012.

[26] E. Mattar, “A Practical Neuro-fuzzy Mapping and Control for a 2 DOF Robotic Arm System,” *Int. J. Comput. Digit. Syst.*, vol. 2, no. 3, pp. 109–121, Sep. 2013.

[27] P. Jha and B. B. Biswal, “A Neural Network Approach for Inverse Kinematic of a SCARA Manipulator,” *Int. J. Robot. Autom.*, vol. 3, no. 1, pp. 52–61, 2014.

[28] W. Mohammed Jasim, “Solution of Inverse Kinematics for SCARA Manipulator Using Adaptive Neuro-Fuzzy Network,” *Int. J. Soft Comput.*, vol. 2, no. 4, pp. 59–66, Nov. 2011.

[29] M. Yousif Ismail and H. Safwan Mawlood, “Modeling and simulation of Industrial SCARA Robot Arm,” *Int. J. Eng. Adv. Technol.*, vol. 4, no. 4, pp. 220–229, 2015.

[30] M. Aghajarian and K. Kiani, “Inverse Kinematics solution of PUMA 560 robot arm using ANFIS,” in *URAI 2011 - 2011 8th International Conference on Ubiquitous Robots and Ambient Intelligence*, 2011, pp. 574–578.

[31] Y. I. Al Mashhadany, “ANFIS-Inverse-Controlled PUMA 560 Workspace Robot with Spherical Wrist,” *Procedia Eng.*, vol. 41, no. Iris, pp. 700–709, 2012.

[32] D. Xu, C. a. Acosta Calderon, J. Q. Gan, H. Hu, and M. Tan, “An analysis of the inverse kinematics for a 5-DOF manipulator,” *Int. J. Autom. Comput.*, vol. 2, no. 2, pp. 114–124, Dec. 2005.

[33] P. Agnihotri, V. K. B. Er, and G. Singh, “Review of Anfis Tool used in 5 Dof Robotic Arm,” *Int. J. Eng. Res. Technol.*, vol. 4, no. 09, pp. 896–900, 2015.

[34] M. T. Das and L. C. Dülger, “Mathematical modelling, simulation and experimental verification of a scara robot,” *Simul. Model. Pract. Theory*, vol. 13, no. 3, pp. 257–271, 2005.

[35] S. Yuan, Z. Liu, and X. Li, “Modeling and simulation of robot based on matlab/simmechanics,” in *Proceedings of the 27th Chinese Control Conference, CCC*, 2008, pp. 161–165.

[36] Y. Shaoqiang, L. Zhong, and L. Xingshan, “Modeling and Simulation of Robot Based on Matlab / SimMechanics,” in *Proceedings of the 27th Chinese Control Conference*, 2008, pp. 1–5.

[37] T. D. Le, H. J. Kang, and Y. S. Ro, “Robot manipulator modeling in Matlab-Simmechanics with PD control and online gravity compensation,” in *2010 International Forum on Strategic Technology, IFOST 2010*, 2010, pp. 446–449.

[38] Dassault, *Introducing SolidWorks*. SolidWorks Corporation, 2014.

[39] MathWorks Inc., “SimMechanics User’s Guide,” in *SpringerReference*, Springer-Verlag, 2014.

[40] J. J. Craig, “Introduction to Robotics: Mechanics and Control 3rd,” *Prentice Hall*, vol. 1, no. 3, p. 408, 2004.

[41] R. S. Hartenberg and J. Denavit, “A kinematic notation for lower-pair mechanisms based on metrics,” *Trans. ASME. J. Appl. Mech.*, vol. 22, pp. 215–221, 1955.

[42] C. C. Lee, “Fuzzy Logic in Control Systems : Fuzzy Logic Controller - Part 1,” *IEEE Trans. Syst. Man Cybern.*, vol. 20, no. 2, pp. 404–418, 1990.

[43] C. C. C. Lee, “Fuzzy logic in control systems: fuzzy logic controller - Part 2,” *IEEE Trans. Syst. Man. Cybern.*, vol. 20, no. 2, pp. 404–418, 1990.

[44] G. G. (University O. W. Towell and J. W. Shavlik, “Knowledge-based artifical neural networks,” *Artif. Intell.*, vol. 70, pp. 119–165, 1994.

[45] J.-S. R. Jang and C.-T. Sun, “Neuro-Fuzzy Modeling and Control,” *Proc. IEEE*, vol. 83, no. 3, pp. 378 – 406, 1995.

[46] L. Zadeh, “Fuzzy Sets,” *Inf. Control*, vol. 8, no. 3, pp. 338–353, 1965.

[47] E. H. Mamdani and S. Assilian, “An experiment in linguistic synthesis with a fuzzy logic controller,” *Int. J. Man. Mach. Stud.*, vol. 7, no. 1, pp. 1–13, 1975.

[48] M. Sugeno, “Industrial applications of fuzzy control,” *Elsevier Sci. Pub. Co.*, 1985.

[49] F. J. Chang and Y. T. Chang, “Adaptive neuro-fuzzy inference system for prediction of water level in reservoir,” *Adv. Water Resour.*, vol. 29, no. 1, pp. 1–10, 2006.

[50] S. Kurnaz, O. Cetin, and O. Kaynak, “Adaptive neuro-fuzzy inference system based autonomous flight control of unmanned air vehicles,” *Expert Syst. Appl.*, vol. 37, no. 2, pp. 1229–1234, 2010.

[51] J. C. Bezdek, “Pattern Recognition with Fuzzy Objective Function Algorithms,” *SIAM Rev.*, vol. 25, no. 3, pp. 442–442, 1983.

[52] S. L. Chiu, “Fuzzy model identification based on cluster estimation.,” *Journal of intelligent and Fuzzy systems*, vol. 2, no. 3. pp. 267–278, 1994.

[53] P. Corke, *Robotics, Vision and Control*, vol. 73. Berlin, Heidelberg: Springer Berlin Heidelberg, 2011.

[54] P. Corke, *Robotics, Vision and Control - Fundamental Algorithms in MATLAB*. 2011.

[55] P. Corke, “MATLAB toolboxes: Robotics and vision for students and teachers,” *IEEE Robot. Autom. Mag.*, vol. 14, no. 4, pp. 16–17, 2007.

[56] S. Raptis and E. Tzafestas, “Robot inverse kinematics via neural and neurofuzzy networks: architectural and computational aspects for improved performance,” *J. Inf. Optim. Sci.*, vol. 28, no. 6, pp. 905–933, 2007.

[57] J. J. Slotine and S. S. Sastry, “Tracking control of non-linear systems using sliding surfaces with application to robot manipulators,” in *American Control Conference*, 1983, pp. 132–135.

[58] V. Parra-Vega, S. Arimoto, Y. H. Liu, G. Hirzinger, and P. Akella, “Dynamic sliding PID control for tracking of robot manipulators: Theory and experiments,” *IEEE Trans. Robot. Autom.*, vol. 19, no. 6, pp. 967–976, 2003.

[59] F. Lin and R. D. Brandt, “An Optimal Control Approach to Robust Control of Robot Manipulators,” *Robot. Autom.*, vol. 14, no. 1, pp. 69–77, 1998.

[60] H. Pan and M. Xin, “Nonlinear robust and optimal control of robot manipulators,” *Nonlinear Dyn.*, vol. 76, no. 1, pp. 237–254, 2014.

[61] L. Bascetta and P. Rocco, “Revising the robust-control design for rigid robot manipulators,” *IEEE Trans. Robot.*, vol. 26, no. 1, pp. 180–187, 2010.

[62] J. J. Fu, “Repetitive Learning Control of Robotic Manipulators,” 1993.

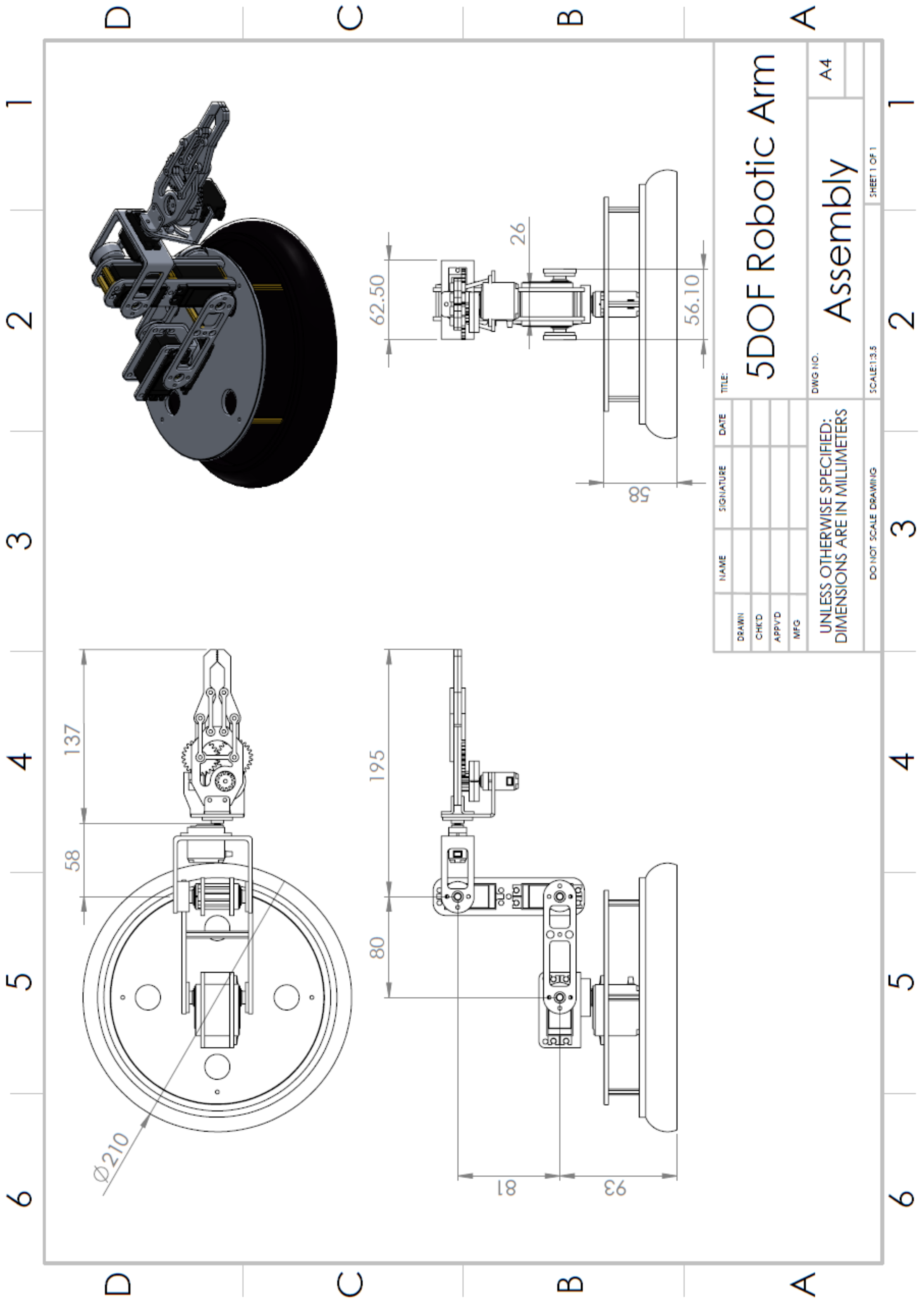
[63] A. Tayebi, “Adaptive iterative learning control for robot manipulators,” *Automatica*, vol. 40, no. 7, pp. 1195–1203, 2004.

[64] K. Kaneko and R. Horowitz, “Repetitive and adaptive control of robot manipulators with velocity estimation,” *IEEE Trans. Robot. Autom.*, vol. 13, no. 2, pp. 204–217, 1997.

[65] J. Kasac, B. Novakovic, D. Majetic, and D. Brezak, “Passive Finite-Dimensional Repetitive Control of Robot Manipulators,” *IEEE Trans. Control Syst. Technol.*, vol. 16, no. 3, pp. 570–576, 2008.

[66] W. L. Stout and M. E. Sawan, “Application of <e1>H</e1>-infinity theory to robot manipulator control,” in *[Proceedings 1992] The First IEEE Conference on Control Applications*, 1992, no. 2, pp. 148–153.

Appendix A



Detailed Drawing of Dagu 5-DOF Robotic Arm

Appendix B

MATLAB SCRIPTS

```
%---------------------------------------------------------------------%
%%                            dagu5dof.m                             %%
%---------------------------------------------------------------------%

clear all
close all
clc

qz = [0 0 0 0 0]; % zero position
qr = [0 pi -pi/2 0 0]; % ready position
qf = [0 0 pi/2 0 0]; % forward ready position
qv = [0 pi/2 0 pi/2 0]; % vertical stretch
qh = [0 0 0 pi/2 0]; % horizontal stretch

L(1) = Link([0, 0, 0, 0],'modified');
L(2) = Link([0, 0, 0, pi/2],'modified');
L(3) = Link([0, 0, 80, 0],'modified');
L(4) = Link([0, 0, 81, 0],'modified');
L(5) = Link([0, 58, 0, pi/2],'modified');

dagu5r = SerialLink(L, 'name', 'Dagu 5DOF');
dagu5r.base = [1 0 0 0; 0 1 0 0; 0 0 1 93; 0 0 0 1];
dagu5r.tool = [1 0 0 0; 0 1 0 0; 0 0 1 137; 0 0 0 1];

dagu5r.qlim(1,:)= [-pi, pi];
dagu5r.qlim(2,1)=0;
dagu5r.qlim(3,:)= [-3*pi/4, 3*pi/4]; % [-3*pi/4, 3*pi/4]
dagu5r.qlim(4,:)= [-pi/4, 5*pi/4]; %[-pi/4, 5*pi/4]

T1 = dagu5r.fkine(qz);
dagu5r.plot(qz)
T2 = dagu5r.fkine(qr);

T= ctraj(T1,T2,51); % cartesian space
ik_sol = dagu5r.ikcon(T,qz);

via =[qh;qf;qv;qr]; % multi-segment joint space
t = (0:4/100:4)';
q = mstraj(via, [], [1 1 1 1], qz, 0.04, 0.16);

ik_sol = [q; q(end,:)];

T = cat(3,(ctraj(dagu5r.fkine(qz),dagu5r.fkine(qh),100/4)),...
     (ctraj(dagu5r.fkine(qh),dagu5r.fkine(qf),100/4)),...
     (ctraj(dagu5r.fkine(qf),dagu5r.fkine(qv),100/4)),...
     (ctraj(dagu5r.fkine(qv),dagu5r.fkine(qr),100/4)));
T = cat(3, T , T(:,:,end));

plot(squeeze(T(1:3,4,:))');
hold on
plot(squeeze(T_sm.data(1:3,4,:))','.');
```

```
figure(2)
plot(abs(tr2rpy(T)));
hold on
plot(abs(tr2rpy(T_sm.data)),'.');
%-------------------------------------------------------------------%

%-------------------------------------------------------------------%
%%                            dagu_fkine.m                        %%
%-------------------------------------------------------------------%

th1 = -pi:pi/6:pi;
th2 = 0:pi/6:pi;
th3 = -2*pi/3:pi/6:3*pi/4; % -3*pi/4::3*pi/4
th4 = -pi/6:pi/6:5*pi/4; % -pi/4::5*pi/4
th5 = -pi:pi/6:pi;

syms a2 a3 d1 d5 c1 s1 c2 s2 c3 s3 c4 s4 c5 s5;

T01 = [c1 -s1 0 0; s1 c1 0 0; 0 0 1 d1; 0 0 0 1];
T12 = [c2 -s2 0 0; 0 0 -1 0; s2 c2 0 0; 0 0 0 1];
T23 = [c3 -s3 0 a2; s3 c3 0 0; 0 0 1 0; 0 0 0 1];
T34 = [c4 -s4 0 a3; s4 c4 0 0; 0 0 1 0; 0 0 0 1];
T45 = [c5 -s5 0 0; 0 0 -1 -d5; s5 c5 0 0; 0 0 0 1];

T35 = T34 * T45;
T35 = simplify(T35, 'Steps', 150);

T25 = T23 * T35;
T25 = simplify(T25, 'Steps', 150);

T15 = T12 * T25;
T15 = simplify(T15, 'Steps', 150);

T05 = T01 * T15;
T05 = simplify(T05, 'Steps', 150);

d1 = 93;
d5 = 195;
a2 = 80;
a3 = 81;

[T_BW, TH] = TT_mapping(T05, th1, th2, th3, th4, th5);
RPY = tr2rpy(T_BW);
XYZ = [squeeze(T_BW(1,4,:)) squeeze(T_BW(2,4,:))
squeeze(T_BW(3,4,:))];
plot3(XYZ(:,1),XYZ(:,2),XYZ(:,3),'o','MarkerSize',2,'MarkerFaceColor'
,'b');

data1 = [XYZ RPY TH(:,1)];
data2 = [XYZ RPY TH(:,2)];
data3 = [XYZ RPY TH(:,3)];
data4 = [XYZ RPY TH(:,4)];
data5 = [XYZ RPY TH(:,5)];
%-------------------------------------------------------------------%
```

```
%--------------------------------------------------------------------%
%%                         TT_mapping.m                            %%
%--------------------------------------------------------------------%

function [T, THETA] = TT_mapping(T_sym, theta1, theta2, theta3,
theta4, theta5 )

[TH1, TH2, TH3, TH4, TH5] = ndgrid(theta1, theta2, theta3, theta4,
theta5);
size(TH1)

if numel(TH1) == 1
    T = sym2num(T_sym, TH1, TH2, TH3, TH4, TH5);
    THETA = [TH1, TH2, TH3, TH4, TH5];
    return
end

    % Randomly select data points
    rand_indices = randperm(numel(TH1));
    TH1 = TH1(rand_indices(1:end));
    TH2 = TH2(rand_indices(1:end));
    TH3 = TH3(rand_indices(1:end));
    TH4 = TH4(rand_indices(1:end));
    TH5 = TH5(rand_indices(1:end));

size(TH1)

c = 1;
for i = 1:numel(TH1)
    if ((TH2(i)== pi && TH3(i)<-pi/4) || (TH2(i)== 0 && TH3(i)> pi/4)
|| (TH2(i)== 0 && TH3(i)== -3*pi/4 && TH4(i)<3*pi/4) || (TH2(i)== pi
&& TH3(i)== -pi/4 && TH4(i)>3*pi/4)) == false

        T_temp(:,:,c) = sym2num(T_sym, TH1(i), TH2(i), TH3(i),
TH4(i), TH5(i));
        TH_temp(c,:) = [TH1(i), TH2(i), TH3(i), TH4(i), TH5(i)];
        c = c + 1;
    end
end

size(TH_temp)

c = 1;

for i = 1:length(TH_temp)
    if ((T_temp(1,4,i) > 0 ) || (T_temp(2,4,i) < 0 ) ||
(T_temp(3,4,i) < 0 ) || ((T_temp(3,4,i) < 100) && (T_temp(1,4,i) <
120) && (T_temp(1,4,i) > -120)) || ((T_temp(3,4,i) < 100) &&
(T_temp(2,4,i) < 120) && (T_temp(2,4,i) > -120)) || (T_temp(3,4,i) >
440) || (T_temp(1,4,i) < 5) && (T_temp(1,4,i) > -5) && (T_temp(2,4,i)
< 5) && (T_temp(2,4,i) > -5)) == false

        T(:,:,c) = T_temp(:,:,i);
        THETA(c,:) = TH_temp(i,:);
        c = c + 1;
    end
end
```

```
size(THETA)

end
%-------------------------------------------------------------------%


%-------------------------------------------------------------------
%%                            sym2num.m                            %%
%-------------------------------------------------------------------%

function T = sym2num(Ts, TH1, TH2, TH3, TH4, TH5)

s1 = sin(TH1);
s2 = sin(TH2);
s3 = sin(TH3);
s4 = sin(TH4);
s5 = sin(TH5);

c1 = cos(TH1);
c2 = cos(TH2);
c3 = cos(TH3);
c4 = cos(TH4);
c5 = cos(TH5);

T = double(subs(Ts));

end
%-------------------------------------------------------------------%

%-------------------------------------------------------------------%
%%                          mapFeature.m                           %%
%-------------------------------------------------------------------%

function out = mapFeature(X1, X2)

%   MAPFEATURE Feature mapping function to polynomial features

%   MAPFEATURE(X1, X2) maps the two input features to quadratic
%   features used in the regularization exercise.
%   Returns a new feature array with more features, comprising of
%   X1, X2, X1.^2, X2.^2, X1*X2, X1*X2.^2, etc..
%   Inputs X1, X2 must be the same size

degree = 2;

out = ones(size(X1(:,1)));
for i = 1:degree
    for j = 0:i
        out(:, end+1) = (X1.^(i-j)).*(X2.^j);
    end
end

out = out(:,2:end);

end
%-------------------------------------------------------------------%
```

```
%-------------------------------------------------------------------%
%%                          traj_cir3d.m                          %%
%-------------------------------------------------------------------%

clear qcir_c J
R = 75;
N = 16 ;
Cxy = circle([115+R,0], R, 'n', N);
Cxy = [Cxy, Cxy(:,1)]';

Cz = (174+R) - R*cos(0:2*pi/(N):2*pi)';
Cz = Cz(1:end-1,:);
Cz = circshift(Cz, N/2);
Cz = [Cz; Cz(1,:)];

Cxyz = [Cxy, Cz];
Cxyz = Cxyz(1:end-1,:);
Cxyz = circshift(Cxyz, N/2);

Cp = [Cxyz; Cxyz(1,:)];

figure(1)
plot3(Cp(:,1),Cp(:,2),Cp(:,3),'b-.')
hold on

rx = zeros(1,length(Cp)-1);
ry = zeros(1,length(Cp)-1);
rz = zeros(1,length(Cp)-1);

for i = 1:length(Cp)-1
    rx(i) = atan2((Cp(i+1,3) - Cp(i,3)),(Cp(i+1,2) - Cp(i,2)));
    ry(i) = atan2((Cp(i+1,3) - Cp(i,3)),(Cp(i+1,1) - Cp(i,1)));
    rz(i) = atan2((Cp(i+1,2) - Cp(i,2)),(Cp(i+1,1) - Cp(i,1)));
    Rcir = rotx(0)*roty(pi/2 + ry(i))*rotz(0);
end

for i = 1:length(Cp)
    Tcir(:,:,i)= rt2tr (Rcir,Cp(i,:)');
    qcir(i,:) = dagu5r.ikine(Tcir(:,:,i), qr, [1 1 1 0 0 0]);
%   ready position modified
end

Tcir_i = dagu5r.fkine(qcir);

Cp_i=transl(Tcir_i);
Crpy = tr2rpy(Tcir_i);
ccir=[Cp_i, Crpy];

mstraj_c = mstraj(ccir(2:end,:),[],0.5*ones(1,1,length(qcir)-
1),ccir(1,:),0.04,0);
tim = 0:0.04:length(mstraj_c)*0.04;
mstraj_c = [ccir(1,:); mstraj_c];

Pcir = transl(mstraj_c(:,1:3));
Rcir = rpy2tr(mstraj_c(:,4:6));

for i = 1:length(Pcir)
```

```
    Tcir_c(:,:,i)=Pcir(:,:,i)*Rcir(:,:,i);
    qcir_c(i,:) = dagu5r.ikine(Tcir_c(:,:,i), qr, [1 1 1 0 0 0]);
%   w.r.t. ready position
end

qcir_c(:,5) = 3*qcir_c(:,1);
length(qcir_c)*0.04;

for i = 1:5
    J(i) =mean(abs(ScopeTor.signals(i).values)) /
mean(abs(ScopeVel.signals(3).values(:,i)));
end

p_sm = squeeze(T_sm.data(1:3,4,:))';
plot3(p_sm(:,1),p_sm(:,2),p_sm(:,3),'r', 'LineWidth',2)
%--------------------------------------------------------------------%

%--------------------------------------------------------------------%
%%                 Inverse_Kinematics_genfis1.m                     %%
%--------------------------------------------------------------------%

inmfType = 'gbellmf';%'psigmf';%'gbellmf';
outmftype = 'linear';
epoch = 100; %500
trnOpt = [epoch NaN NaN NaN];
dispOpt = [1 1 1 1];
optMethod = 1; % 1 = hybrid , 0 = back=propagation

% split training data and check data
cut1 = length(data1)- round(0.25*length(data1));
% split check data and test data
cut2 = cut1 + round((length(data1)-cut1)/2);

% Train first ANFIS network
fprintf('-->%s\n','Start training first ANFIS network. It may take
few minutes depending on your computer system.')
infis1 = genfis1([data1(1:cut1,1:6) data1(1:cut1,7)], [3 3 2 2 2 3],
inmfType, outmftype);
[anfis1,error1,stepsize1,chkFis1,chkErr1] = anfis(data1(1:cut1,:),
infis1, trnOpt, dispOpt,data1(cut1+1:cut2,:),optMethod);

% Train second ANFIS network
fprintf('-->%s\n','Start training second ANFIS network. It may take
few minutes depending on your computer system.')
infis2 = genfis1([data2(1:cut1,1:6) data2(1:cut1,7)], [2 3 3 3 2 2],
inmfType, outmftype);
[anfis2,error2,stepsize2,chkFis2,chkErr2] = anfis(data2(1:cut1,:),
infis2, trnOpt, dispOpt,data2(cut1+1:cut2,:),optMethod);

% Train third ANFIS network
fprintf('-->%s\n','Start training third ANFIS network. It may take
few minutes depending on your computer system.')
infis3 = genfis1([data3(1:cut1,1:6) data3(1:cut1,7)], [2 3 3 3 2 2],
inmfType, outmftype);
[anfis3,error3,stepsize3,chkFis3,chkErr3] = anfis(data3(1:cut1,:),
infis3, trnOpt, dispOpt,data3(cut1+1:cut2,:),optMethod);
```

```
% Train fourth ANFIS network
fprintf('-->%s\n','Start training fourth ANFIS network. It may take
few minutes depending on your computer system.')
infis4 = genfis1([data4(1:cut1,1:6) data4(1:cut1,7)], [2 3 3 3 2 2],
inmfType, outmftype);
[anfis4,error4,stepsize4,chkFis4,chkErr4] = anfis(data4(1:cut1,:),
infis4, trnOpt, dispOpt,data4(cut1+1:cut2,:),optMethod);

% Train fifth ANFIS network
fprintf('-->%s\n','Start training fifth ANFIS network. It may take
few minutes depending on your computer system.')
infis5 = genfis1([data5(1:cut1,1:6) data5(1:cut1,7)], [2 2 2 3 3 3],
inmfType, outmftype)
[anfis5,error5,stepsize5,chkFis5,chkErr5] = anfis(data5(1:cut1,:),
infis5, trnOpt, dispOpt,data5(cut1+1:cut2,:),optMethod);
%-------------------------------------------------------------------%

%-------------------------------------------------------------------%
%%                            Testing.m                            %%
%-------------------------------------------------------------------%

input = [XYZ(cut2+1:end,:) RPY(cut2+1:end,:)];
THETA1D = TH(cut2+1:end,1);
THETA2D = TH(cut2+1:end,2);
THETA3D = TH(cut2+1:end,3);
THETA4D = TH(cut2+1:end,4);
THETA5D = TH(cut2+1:end,5);

THETA1P = evalfis(input, anfis1); % theta1 predicted by anfis1
THETA2P = evalfis(input, anfis2); % theta2 predicted by anfis2
THETA3P = evalfis(input, anfis3); % theta1 predicted by anfis1
THETA4P = evalfis(input, anfis4); % theta2 predicted by anfis2
THETA5P = evalfis(input, anfis5); % theta1 predicted by anfis1

theta1diff = (THETA1D - THETA1P);
theta2diff = (THETA2D - THETA2P);
theta3diff = (THETA3D - THETA3P);
theta4diff = (THETA4D - THETA4P);
theta5diff = (THETA5D - THETA5P);

figure(1);
plot(theta1diff,'x');
title('ANFIS for Joint 1: Prediction of Inverse
Kinematics','fontsize',10)
xlabel('No. of Data Set Points','fontsize',10)
ylabel('error: e_1 = \theta_1_d - \theta_1_p','fontsize',10)

figure(2);
plot(theta2diff,'x');
title('ANFIS for Joint 2: Prediction of Inverse
Kinematics','fontsize',10)
xlabel('No. of Data Set Points','fontsize',10)
ylabel('error: e_2 = \theta_2_d - \theta_2_p','fontsize',10)

figure(3);
plot(theta3diff,'x');
title('ANFIS for Joint 3: Prediction of Inverse
Kinematics','fontsize',10)
xlabel('No. of Data Set Points','fontsize',10)
```

```
ylabel('error: e_3 = \theta_3_d - \theta_3_p','fontsize',10)

figure (4);
plot(theta4diff,'x');
title('ANFIS for Joint 4: Prediction of Inverse
Kinematics','fontsize',10)
xlabel('No. of Data Set Points','fontsize',10)
ylabel('error: e_4 = \theta_4_d - \theta_4_p','fontsize',10)

figure(5);
plot(theta5diff,'x');
title('ANFIS for Joint 1: Prediction of Inverse
Kinematics','fontsize',10)
xlabel('No. of Data Set Points','fontsize',10)
ylabel('error: e_5 = \theta_5_d - \theta_5_p','fontsize',10)
%-----------------------------------------------------------------%
```