YILDIZ TECHNICAL UNIVERSITY
Mechatronics Engineering Department

Neural Network Based Inverse Kinematics for a PUMA 560

Lecture Code – Name - Group

MKT4811 Introduction to Artificial Intelligence

Group 1

Lecture Year and Term

2019 – 2020 Fall Term

Project Team

Oğuzhan YARDIMCI / 1506A023

Mustafa UĞUR / 1506A007

Cemil YILMAZ / 15067005

Fatih KARADENİZ / 16067008

Academic Member

Dr. Ahmet KIRLI

09.12.2019

Contents

1. INTRODUCTION

Serial 6 degree of freedom (DOF) robots and robotic cells are very common in industry area and academic studies. They are used for fabrication, assembly, packaging, and so forth. Developing an effective travel path is challenging for many reasons, including the kinematic structure, the robot behavior due to the robot's configuration. To determine the joint angles for a specific end-effector position we want to design a neural network-based inverse kinematics method.

2. INDUSTRIAL ROBOTS

The use of industrial robots became identifiable as unique devices in the 1990s, along with computer-aided design (CAD) systems, and computer-aided manufacturing (CAM) systems. Presently, the robots characterize the latest trends in automation of the manufacturing process.

New disciplines of engineering, such as manufacturing engineering, applications engineering, and knowledge engineering are beginning to deal with the complexity of the field of robotics and the larger area of factory automation. Within a few years, it is possible that robotics engineering will stand on its own as a distinct engineering discipline.

A robot can be considered as a computer controlled industrial manipulator, operating under some degree of autonomy. Such devices are extremely complex electromechanical systems whose analytic description requires advanced methods which present many challenging and interesting research problems. The official definition of such a robot comes from the Robot Institute of America (RIA): A robot is a reprogrammable multifunctional manipulator designed to move material, parts, tools, or specialized devices through variable programmed motions for the performance of a variety of tasks. The key element in this definition is the programmability of robots. It is the computer brain that gives the robot its utility and adaptability. The so-called robotics revolution is, in fact, part of the larger computer revolution [1].

3. PUMA 560 ROBOT

The first surgical robot, PUMA 560, was used in 1985 in a stereotaxic operation, in which computed tomography was used to guide the robot as it inserted a needle into the brain for biopsy, a procedure previously subject to error from hand tremors during needle placement.

This study will focus mainly on the neural network-based inverse kinematics for a PUMA 560 robot. Figure 1 shows the PUMA 560, and Figure 2 shows its degrees of joint rotation [2]. On the basis of its geometry, the PUMA 560 robot is a six revolute jointed articulated manipulator. Table 1 shows the joint angle ranges of PUMA 560 robot.

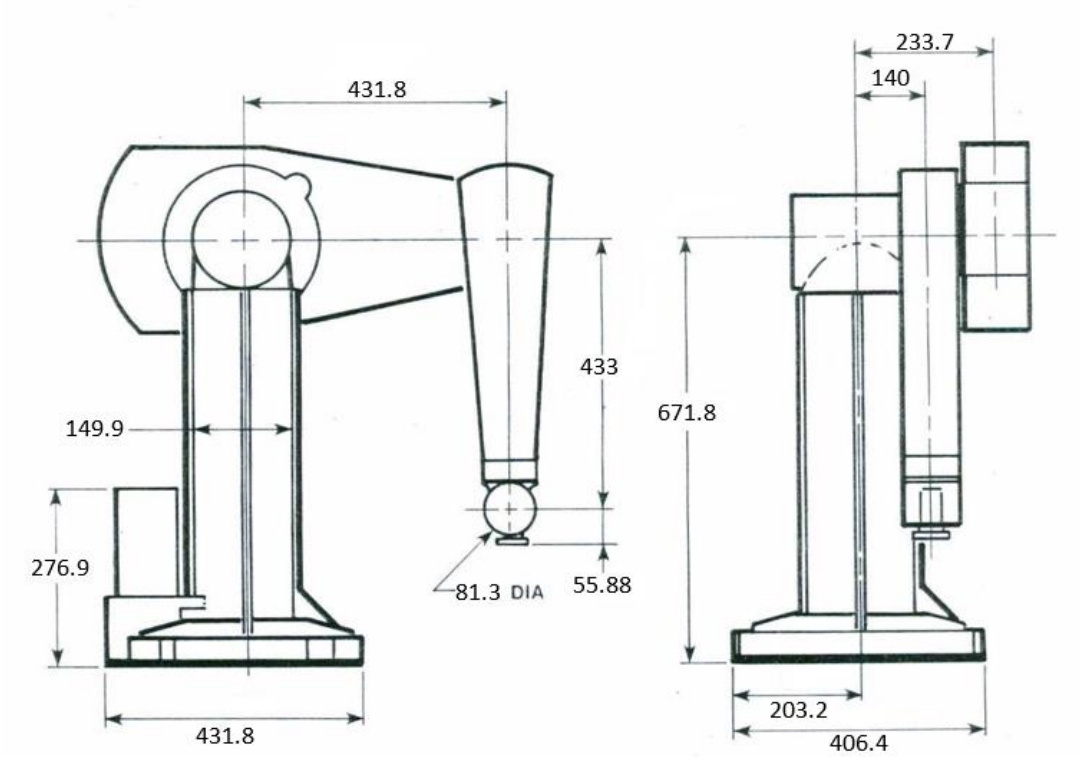

Figure 1: PUMA 560

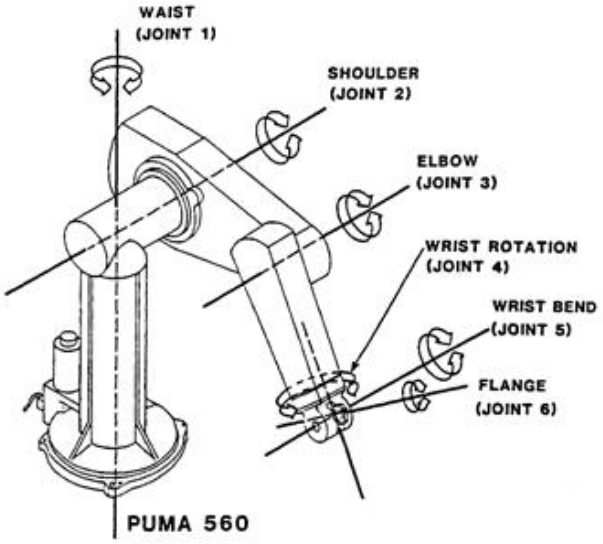

Figure 2: Degrees of Joint Rotation

(degree)	**Joint 1**	**Joint 2**	**Joint 3**
min	*-1.5 rad*	*-1.5 rad*	*-0.8 rad*
max	*1 rad*	*0.7 rad*	*1.5 rad*

Table1: Joint Angle Ranges of PUMA 560 robot.

4. KINEMATIC ANALYSIS

The transformation of coordinates of the end-effector point from the joint space to the world space is known as ***forward kinematic*** transformation. It is also called more clearly, forward kinematic which is to determine the position and orientation of the end effector given the values for the joint variables of the robots.

Axis Number i	Link Length a_i	Twist Angle α_i	Link offset d_i	Joint angle θ_i
1	$a_1 = 0$	$\pi/2$	$d_1 = 0$	θ_1^*
2	$a_2 = 431.8\ mm$	0	$d_2 = 0$	θ_2^*
3	$a_3 = 20.3\ cm$	$-\pi/2$	$d_3 = 150\ cm$	θ_3^*
4	$a_4 = 0$	$\pi/2$	$d_4 = 431.8\ cm$	θ_4^*
5	$a_5 = 0$	$-\pi/2$	$d_5 = 0$	θ_5^*
6	$a_6 = 0$	0	$d_6 = 0$	θ_6^*

Table 2: D-H Parameters of the PUMA 560 Robot Arm

Denavit–Hartenberg (DH) method uses the four parameters including a_i,α_i, d_i and θ_i; which are the link length, twist angle, link offset and joint angle, respectively.

- θ_i , joint angle is angle from x_{i-1} to x_i measured around the z_{i-1}.
- d_i, link offset is distance from O_{i-1} to O_i measured along z_{i-1}.
- a_i, link length is distance from z_{i-1} to z_i measured along x_i.
- α_i, twist angle is angle from z_{i-1} to z_i measured along x_i

The matrix T_i^{i-1} is known as a D-H convention matrix given in this equation. In the matrix T_i^{i-1}, it means transformation matrix, the quantities of a_{i-1},α_{i-1}, d_i are constant for a given link while the parameter θ_i for a revolute joint is variable [3].

$$A_i = T_i^{i-1} = \begin{bmatrix} cos\theta_i & -cos\alpha_i sin\theta_i & sin\alpha_i sin\theta_i & a_i cos\theta_i \\ sin\theta_i & cos\alpha_i cos\theta_i & -sin\alpha_i cos\theta_i & a_i sin\theta_i \\ 0 & sin\alpha_i & cos\alpha_i & d_i \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

We obtain the matrices that is shown below by using this formulation;

$$A_1 = T_1^0 = \begin{bmatrix} cos\theta_1 & 0 & sin\theta_1 & 0 \\ sin\theta_1 & 0 & -cos\theta_1 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$A_2 = T_2^1 = \begin{bmatrix} cos\theta_2 & -sin\theta_2 & 0 & a_2\, cos\theta_2 \\ sin\theta_2 & cos\theta_2 & 0 & a_2 sin\theta_2 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$A_3 = T_3^2 = \begin{bmatrix} cos\theta_3 & 0 & -sin\theta_3 & a_3\, cos\theta_3 \\ sin\theta_3 & 0 & cos\theta_3 & a_3 sin\theta_3 \\ 0 & -1 & 0 & d_3 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$A_4 = T_4^3 = \begin{bmatrix} cos\theta_4 & 0 & sin\theta_4 & 0 \\ sin\theta_4 & 0 & -cos\theta_4 & 0 \\ 0 & -1 & 0 & d_4 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$A_5 = T_5^4 = \begin{bmatrix} cos\theta_5 & 0 & -sin\theta_5 & 0 \\ sin\theta_5 & 0 & cos\theta_5 & 0 \\ 0 & -1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$A_6 = T_6^5 = \begin{bmatrix} cos\theta_6 & -sin\theta_6 & 0 & 0 \\ sin\theta_6 & cos\theta_6 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$T_6^0 = \prod_{i=1}^{6} T_i^{i-1} = T_1^0\, T_2^1\, T_3^2\, T_4^3\, T_5^4 T_6^5 = \begin{bmatrix} r_{11} & r_{12} & r_{13} & r_{14} \\ r_{21} & r_{22} & r_{23} & r_{24} \\ r_{31} & r_{32} & r_{33} & r_{34} \\ r_{41} & r_{42} & r_{43} & r_{44} \end{bmatrix} = A_1 A_2 A_3 A_4 A_5 A_6$$

$$r_{11} = -s_6\big(c_4 s_1 - s_4 c_1(s_2 s_3 - c_2 c_3)\big) - c_6\Big(c_5\big(s_1 s_4 + c_4 c_1(s_2 s_3 - c_2 c_3)\big) - s_5 c_1(c_2 s_3 + c_3 s_2)\Big)$$
$$r_{12} = s_6\Big(c_5\big(s_1 s_4 + c_4 c_1(s_2 s_3 - c_2 c_3)\big) - s_5 c_1(c_2 s_3 + c_3 s_2)\Big) - c_6\big(c_4 s_1 - s_4 c_1(s_2 s_3 - c_2 c_3)\big)$$
$$r_{13} = s_5\big(s_1 s_4 + c_4 c_1(s_2 s_3 - c_2 c_3)\big) + c_5 c_1(c_2 s_3 + c_3 s_2)$$
$$r_{14} = d_3 s_1 - d_4 c_1(c_2 s_3 + c_3 s_2) + a_2 c_1 c_2 + a_3 c_1(c_2 c_3 - s_2 s_3)$$
$$r_{21} = s_6\big(c_1 c_4 + s_4 s_1(s_2 s_3 - c_2 c_3)\big) + c_6\Big(c_5\big(c_1 s_4 - c_4 s_1(s_2 s_3 - c_2 c_3)\big) + s_5 s_1(c_2 s_3 + c_3 s_2)\Big)$$
$$r_{22} = c_6\big(c_1 c_4 + s_4 s_1(s_2 s_3 - c_2 c_3)\big) - s_6\Big(c_5\big(c_1 s_4 - c_4 s_1(s_2 s_3 - c_2 c_3)\big) + s_5 s_1(c_2 s_3 + c_3 s_2)\Big)$$
$$r_{23} = c_5 s_1(c_2 s_3 + c_3 s_2) - s_5\big(c_1 s_4 - c_4 s_1(s_2 s_3 - c_2 c_3)\big)$$
$$r_{24} = a_2 c_2 s_1 - d_3 c_1 - d_4 s_1(c_2 s_3 + c_3 s_2) + a_3 s_1(c_2 c_3 - s_2 s_3)$$
$$r_{31} = -c_6(c_{23} s_5 - s_{23} c_4 c_5) - s_{23} s_4 s_6$$
$$r_{32} = s_6(c_{23} s_5 - s_{23} c_4 c_5) - s_{23} c_6 s_4$$
$$r_{33} = -c_{23} c_5 - s_{23} c_4 s_5$$
$$r_{34} = d_4 c_{23} + a_3 s_{23} + a_2 s_2$$
$$r_{41} = 0$$
$$r_{42} = 0$$
$$r_{43} = 0$$
$$r_{44} = 1$$

In the expressions of these elements of transformation matrix, the variables are defined as follow:

$$c_i = cos\theta_i \quad , s_i = sin\theta_i \quad , c_{ij} = cos(\theta_i + \theta_j) \quad , s_{ij} = sin(\theta_i + \theta_j)$$

The components of the transformation matrix of T_6^0 can be regulated clearly.

$$r_{11} = -s_6(c_4 s_1 + s_4 c_1 c_{23}) - c_6(c_5(s_1 s_4 - c_4 c_1 c_{23}) - s_5 c_1 s_{23})$$
$$r_{12} = s_6(c_5(s_1 s_4 - c_4 c_1 c_{23}) - s_5 c_1 s_{23}) - c_6(c_4 s_1 + s_4 c_1 c_{23})$$
$$r_{13} = s_5(s_1 s_4 - c_4 c_1 c_{23}) + c_5 c_1 s_{23}$$
$$r_{14} = d_3 s_1 - d_4 c_1 s_{23} + a_2 c_1 c_2 + a_3 c_1 c_{23}$$
$$r_{21} = s_6(c_1 c_4 - s_4 s_1 c_{23}) + c_6(c_5(c_1 s_4 + c_4 s_1 c_{23}) + s_5 s_1 s_{23})$$
$$r_{22} = c_6(c_1 c_4 - s_4 s_1 c_{23}) - s_6(c_5(c_1 s_4 + c_4 s_1 c_{23}) + s_5 s_1 s_{23})$$
$$r_{23} = c_5 s_1 s_{23} - s_5(c_1 s_4 + c_4 s_1 c_{23})$$
$$r_{24} = a_2 c_2 s_1 - d_3 c_1 - d_4 s_1 s_{23} + a_3 s_1 c_{23}$$
$$r_{31} = -c_6(c_{23} s_5 - s_{23} c_4 c_5) - s_{23} s_4 s_6$$
$$r_{32} = s_6(c_{23} s_5 - s_{23} c_4 c_5) - s_{23} c_6 s_4$$
$$r_{33} = -c_{23} c_5 - s_{23} c_4 s_5$$
$$r_{34} = d_4 c_{23} + a_3 s_{23} + a_2 s_2$$
$$r_{41} = 0$$
$$r_{42} = 0$$
$$r_{43} = 0$$
$$r_{44} = 1$$

5. ARTIFICIAL NEURAL NETWORKS

5.1. Background

Artificial neural networks are generally used in the modelling of nonlinear processes. An artificial neural network is a parallel-distributed information processing system. It stores the samples with distributed coding thus forming a trainable nonlinear system. Training of a neural network can be expressed as a mapping between any given input and output data set. Neural networks have some advantages, such as adoption, learning and generalization. Implementation of a neural-network model requires us to decide the structure of the model, the type of activation function and the learning algorithm.

In Figure 3, the schematic representation of a neural-network-based inverse kinematics solution is given. The solution system is based on training a neural network to solve an inverse kinematics problem based on the prepared training data set using direct kinematics equations.

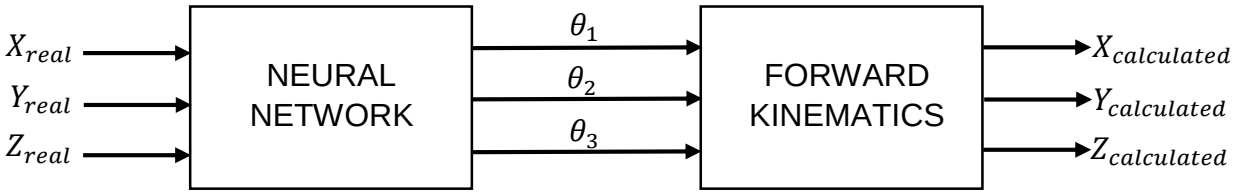

Figure 3: Schematic Representation of a Neural Network Based System

5.2. Artificial Neural Network Training

In order to let the ANN learn the inverse kinematics of PUMA 560, the data set created previously should be used. The saved file has relatively large data, and if this amount of data pairs is used, this will consume relatively long time. So the solution was to reduce the amount of data used without reducing the distribution of the points that cover the working envelope of the robot.

```
net=feedforwarded(30);
```

```
[net,tr] = train(net, position_matrix, theta_values);
```

This part of code trains a shallow neural network. For deep learning with convolutional or LSTM neural networks, see `trainNetwork` instead. When the network weights and biases are initialized, the network is ready for training. The multilayer feedforward network can be trained for function approximation (nonlinear regression) or pattern recognition.

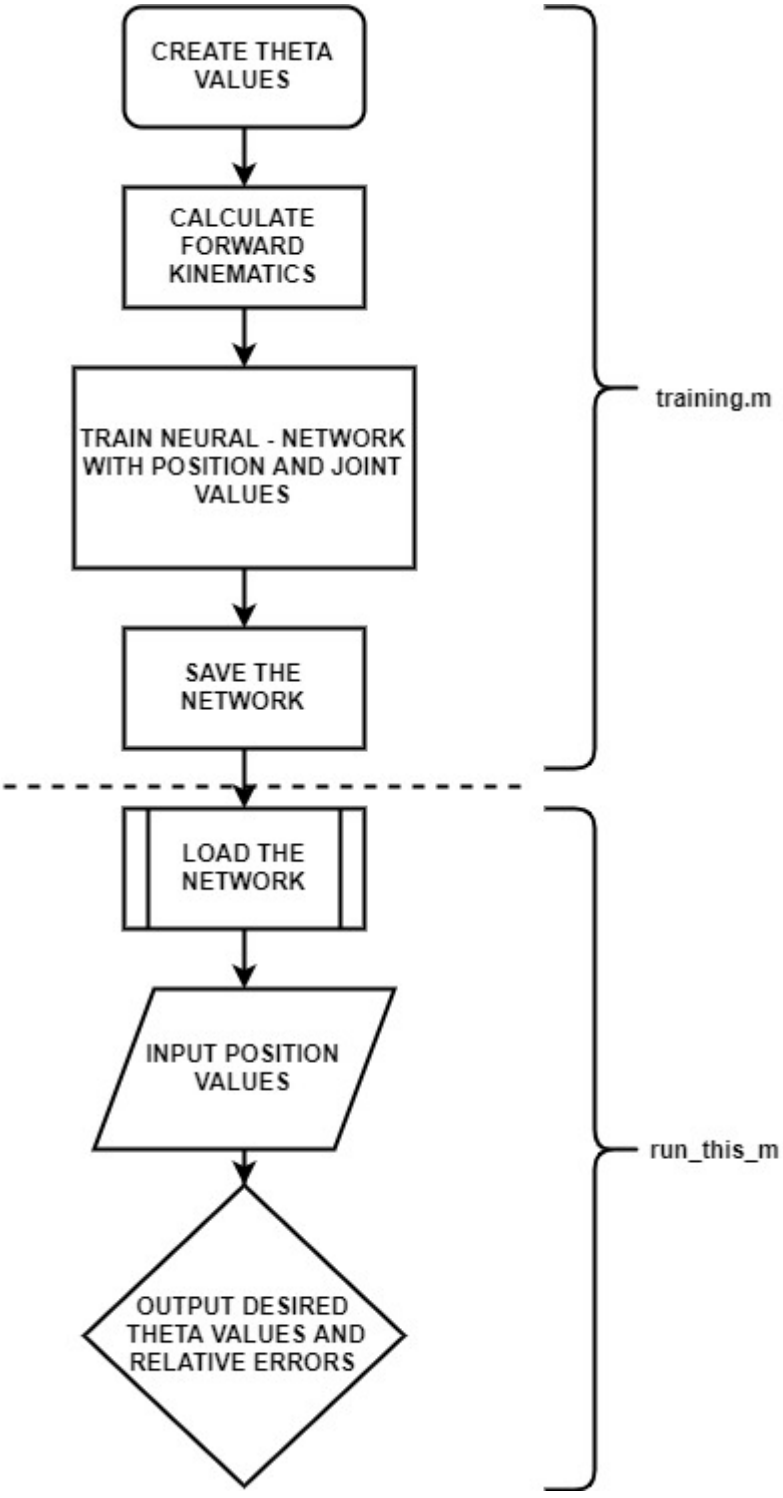

Figure 3: Flowchart of Crating the Learning Data

5.3. Pseudo Code

Pseudo Code for ***create_training_data.m***

```
theta_values_1 ← [-1.5 -1.4 -1.3 … 0.9 1.0]
theta_values_2 ← [-1.5 -1.4 -1.3 … 0.6 0.7]
theta_values_3 ← [-0.8 -0.7 -0.6 … 1.4 1.5]

length1 ← length of array that contains joint1 angles
length2 ← length of array that contains joint2 angles
length3 ← length of array that contains joint3 angles
lenght_of_for_loop ← length1 * length2 * length3

theta_values ← [ ]
for i=1 to length1
    for j=1 to length2
        for k=1 to length3
            add_matrix ← [theta_values_1(i) theta_values(j) theta_values(k)]
            theta_values ← theta_values.addToItsEnd(add_matrix)
        end
    end
end
theta_values ← transpose(theta_values)
x_position ←  ones(lenght_of_for_loop,1)
y_position ←  ones(lenght_of_for_loop,1)
z_position ←  ones(lenght_of_for_loop,1)
position_matrix ←  ones(3, lenght_of_for_loop)

for n=1 to lenght_of_for_loop
    position_matrix(:,n) ← forward_kinematic_calculator(:,n)
end
```

Pseudo Code for ***forward_kinematic_calculator function***

Input: thetaInput

Output: positionOutput

a2 ← 0.4318

a3 ← 0.0203

d3 ← 0.1500

d4 ← 0.4318

alpha1 ← pi/2

alpha3 ← -pi/2

alpha4 ← pi/2

alpha5 ← -pi/2

theta1 ← thetaInput(1)

theta2 ← thetaInput(2)

theta3 ← thetaInput(3)

x_position ← Calculations for x position

y_position ← Calculations for y position

z_position ← Calculations for z position

postionOutput ← [x_position; y_position; z_position]

Pseudo Code for ***run_this.m***

```
position_values ← load(calculated position matrix)
theta_values ← load(calculated joints matrix)
net ← load(trained neural network)

print(iteration numbers)
plot(cost function)
print(total samples used in training)
print(workspace in x direction)
print(workspace in y direction)
print(workspace in z direction)
selection ← input(whether the user wants to work with training data or new data)
if selection == data
        index ← input(get the index number of position in x,y and z)
        plot(workspace in 3D space)
        print(joint values produced byneural network)
        print(joint values in real)
        print(real position vaues in each direction)
        print(calculated position values for nn outputs)
        print(relative errors in each direction)
else
        x_input ← input(get x position)
        y_input ← input(get y position)
        z_input ← input(get z position)
        print(joint values produced by neural network)
        print(calculated position values for nn outputs)
        print(relative errors in each direction)
end
```

6. RESULTS

Figure 5 shows the validation performance of the networks that produce ($\theta_1, \theta_2\ and\ \theta_3$ respectively) and the epoch that contains best performance while training. As depicted in figure, the Mean Square Error is almost under 10^{-2} degree, which indicates good learning performance.

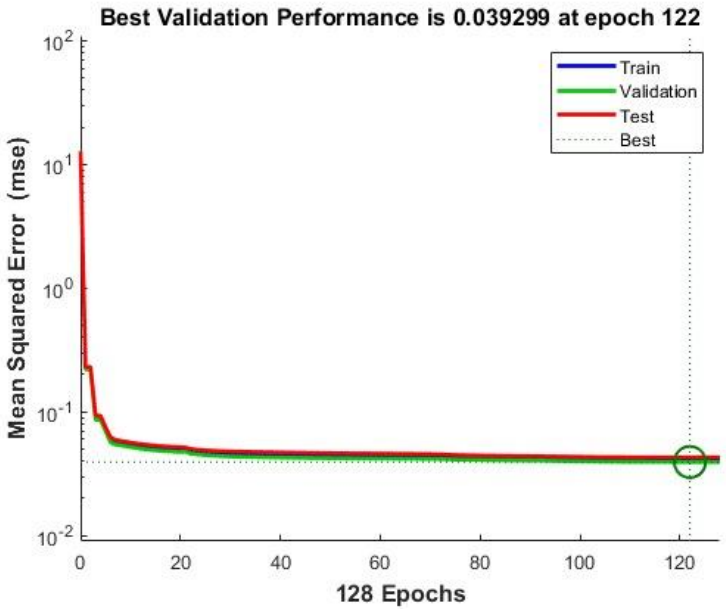

Figure 4: Validation Performance

Figure 6 represent the error histogram for the proposed ($\theta_1, \theta_2\ and\ \theta_3$ respectively) networks. As presented in figure, the error histogram is composed of 20 bars showing the value of most repeated error. It also shows that error margins are almost between (-0.7, 0.7) degrees.

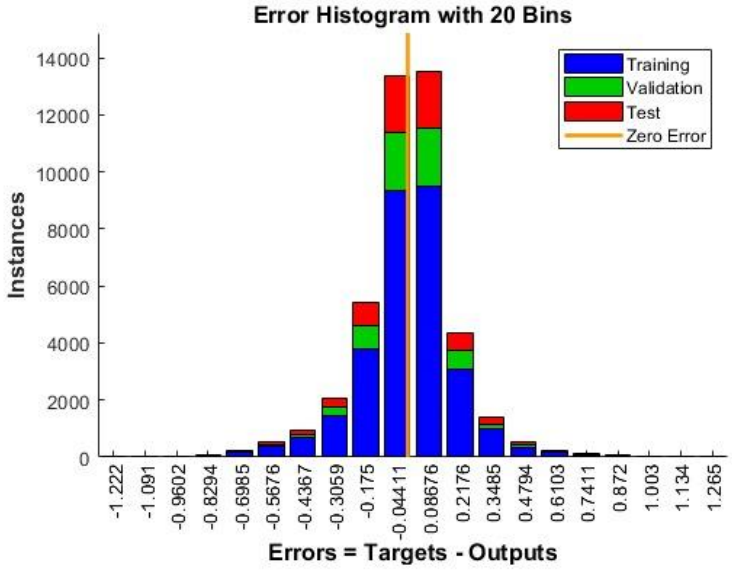

Figure 5: Error Histogram

The Regression plot, which compares the target values and the network output values is shown in Figure 8. For this network, the regression value obtained is 0.96361 indicating an 96.3% fitness.

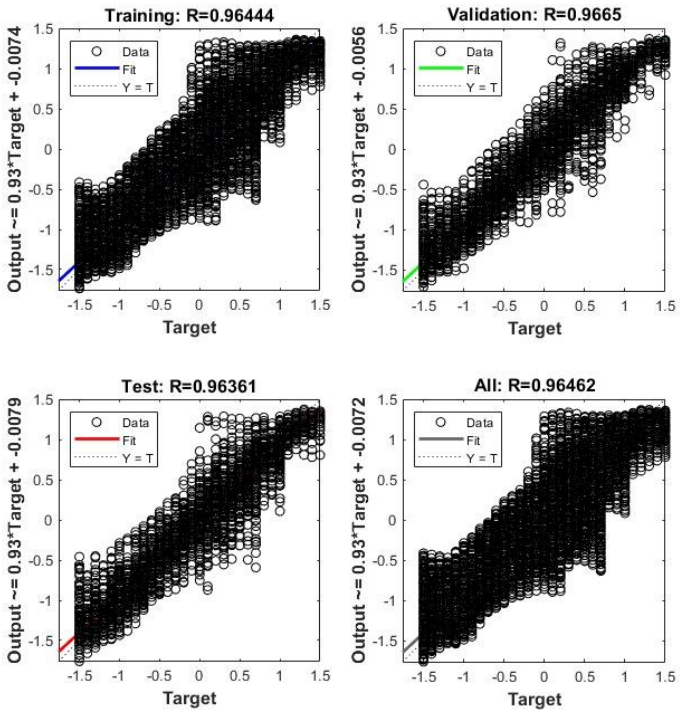

Figure 6: Regression Plot

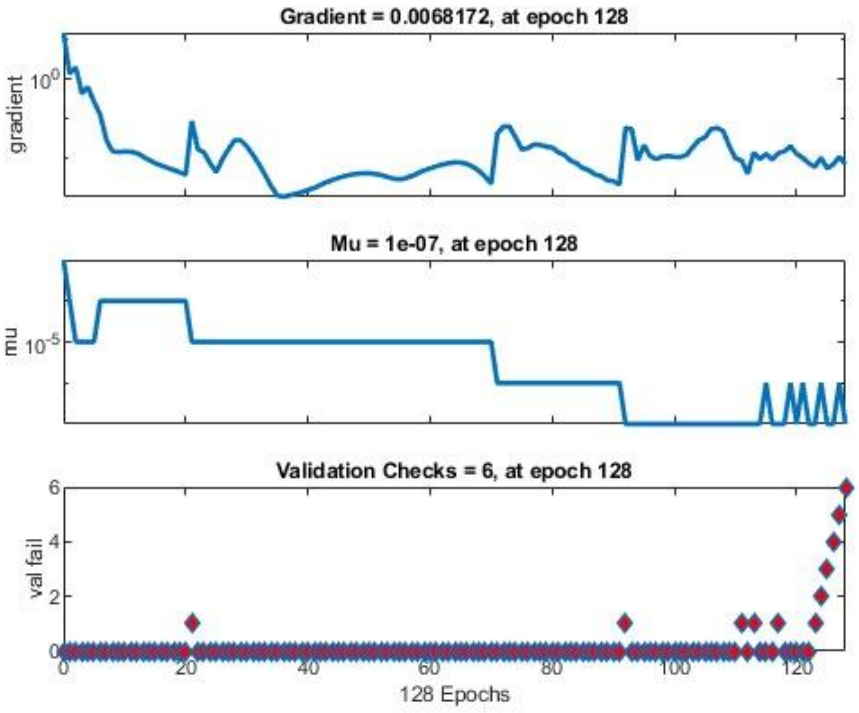

Figure 7: Gradient - Mu – Validation

7. REFERENCES

[1] Aggarwal, Luv & Aggarwal, Kush & Urbanic, Ruth Jill. (2014). Use of Artificial Neural Networks for the Development of an Inverse Kinematic Solution and Visual Identification of Singularity Zone(s). Procedia CIRP. 17. 812-817. 10.1016/j.procir.2014.01.107.

[2] Izadbakhsh, Alireza. (2009). Closed-form dynamic model of PUMA 560 robot arm. ICARA 2009 - Proceedings of the 4th International Conference on Autonomous Robots and Agents. 675-680. 10.1109/ICARA.2000.4803940.

[3] John J. Craig, Introduction to Robotics: Mechanics and Control, Pearson Education Limited, 2014, pp. 165-201