Adaptive Neuro Fuzzy Inference System for Kinematics of 6 DOF Robotic Arm

Karna Patel

Abstract— The control of a Robotic Arm includes the calculations of kinematic models. The calculations for inverse kinematics are comparatively complex and require more computational power as the degrees of freedom increase. An Adaptive Neuro Fuzzy Inference System can be employed to simplify this process and reduce the computational time. Here ANFIS uses the forward kinematics to learn the membership function which can be used to replace the solution from inverse kinematics. Further analysis is carried out to reduce total error or the accumulated error in the model output for a region of interest in the work envelope of the robotic arm. The analysis is carried out for 6 degrees of freedom robotic arm.

Index Terms— ANFIS, Accumulation Error, Control, GUI, Inverse Kinematics, Matlab, Robotic Arm

1 Introduction

With advance in field of robotics, more applications are being explored which can be undertaken by robotic arms. These robotic arms are extensively used in manufacturing industries to improve the productivity. The discussion in this paper is regarding the employment of Adaptive Neuro Fuzzy Inference System for reducing the computational time for solving inverse kinematics of the Robotic Arm. ANFIS is helpful when there is undefined external noise in the system. This method is used for enhancement of the productivity of manufacturing processes employing Robot Arms. There are two basic types of Kinematics; Inverse Kinematics and Forward Kinematics. Solution for inverse kinematics has to be carried out each time the end-effector has to be moved at a certain position in the global co-ordinate system as we need the values of joint configurations which is usually angles for servo motors. ANFIS creates a membership function which can be used in place of solving inverse kinematics every time. This membership function has to be trained using data. The data used here is created using the forward kinematics which can be done very conveniently. After training there may be some inherent error in the region of interest which can be reduced by analyzing and tuning the trained features. Here a 6 DOF Robotic Arm is taken into account for analyzing the kinematics and the error associated with the system.

2 Robotic Arm

Robotic Arm is a mechanical arm that exhibits similar functions of a human arm. It is an aggregate of various mechanisms. Such a manipulator has links which are coupled by joints. These joints can be revolute or prismatic by nature. Robotic Arm is an open Kinematic chain. One joint is fastened to the ground which is known as fixed joint or base joint. The last linkage at the terminal of the kinematic chain is called end-effector. Usually grippers are employed as the end-effectors of robotic arms. These grippers are used for gripping objects and are very useful especially in pick and place robots. The mechanical configuration of the gripper may or may not depend on the mechanical configuration of the arm. Here the discussion is in reference to a Robotic Arm with six degrees of freedom. Thus there are six joints in the kinematic chain. There are five revolute joints and one prismatic joint. Revolute joints allow rotational motion about an axis while prismatic joint allows translation motion along one direction. Basic design of the arm is done in 3d CAD modelling software and further kinematic analysis is done with the help of a numerical computing environment.

3 CAD Model

The 3d CAD Model of Robot Arm can be generated in any CAD modelling software. Solidworks is utilized to generate 3d CAD Model of this 6 DOF Robotic Arm reffered in Fig. 1.

Fig. 1. 3D Cad Model. It visualizes the basic structure of the Robotic Arm.The displayed components include a base, three primary linkages and a gripper.

4 Joint Motion

A simple 6 DOF Robot Arm is considered for the analysis. Five of the joints are revolute and one joint is prismatic. Thus it is manipulator having six degrees of freedom. The different degrees of freedom are reffered in Fig. 2 and Fig. 3. R_1, R_2, R_3, R_4 and R_5 are the five revolute joints. T_1 denotes the prismatic joint. The axis of R_1 is along Y-axis and is at the centre of the workspace. The axes of R_2, R_3 and R_4 are parallel to each other. While the axis of rotation of R_5 and the axis of motion of T_1 are co-incident.

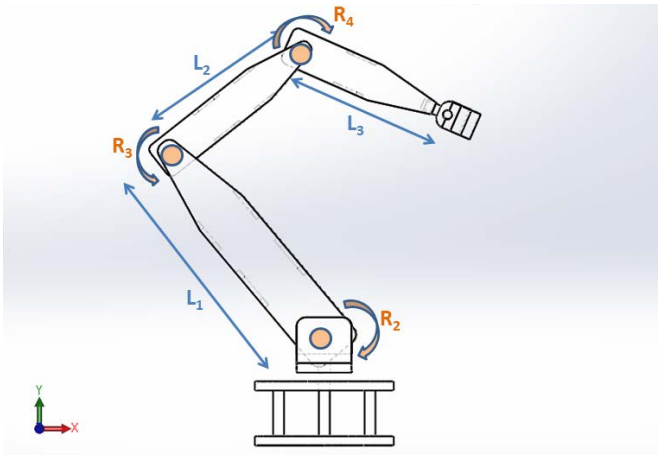

Fig. 2. Joint Motion Orthogonal View. It displays the joint motion of revolute joints R_2, R_3 and R_4 in the XY plane. The axes of these joints are perpendicular to the XY plane.

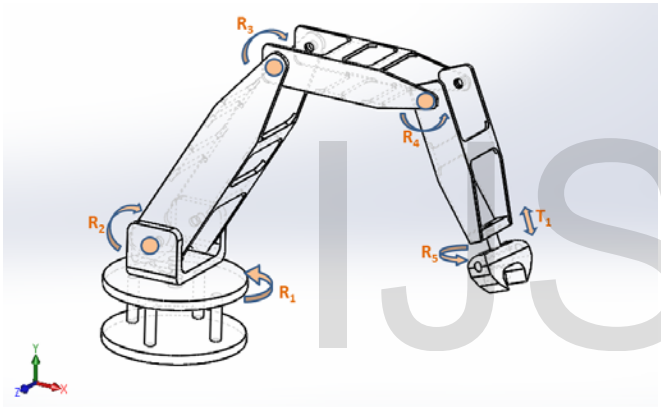

Fig. 3. Joint Motion 3D View. It displays the joint motion of all revolute joints R_1, R_2, R_3 and R_4 along with a prismatic joint T_1.

5 Work Space Envelope

The workspace of a robot manipulator can be described as the set of points that can be reached by its end affector considering a physically possible mechanical configuration. The workspace of Robot Arm is reffered in Fig. 4. It shows a plot in the vertical plane XY. This Fig. indicates the reach of the Robotic Arm given the constraints of each of its joints. The constraints for each joint are given below in the Table 1.

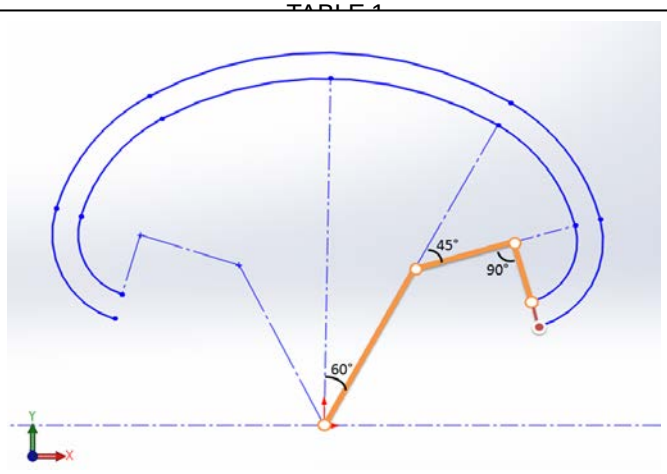

Fig. 4. Workspace in XY plane. It displays the workspace envelope of the Robotic Arm in XY plane and represents the maximum joint positions according to the joint constraints.

6 Kinematics

Kinematics in regard with Robotic Arms can be described as the study of the motion of an open kinematic chain with more than one degrees of freedom. It describes the mechanical configuration of the kinematic chain by representing it geometrically. It gives the relations between the positions and velocities of the links in the kinematic chain. Thus with kinematics, one can determine the positions given the joint parameters and even joint configurations can be determined if positions are given. Kinematics problems can be solved by either Forward Kinematics ar Inverse Kinematics. Forward Kinematics is used for obtaining end-effector position from the joint configurations. While the Inverse Kinematics is used for obtaining joint configurations from the position of end-effector.

6.1 Forward Kinematics

Forward Kinematics is a type of Kinematics. It deals with the problems for finding position and orientation of the end-effector of the Robotic Arm if the joint configurations are given. Usually servo motors are used to actuate the joints of robotic arms. Thus the joint configurations are known at all the times. So forward kinematics is used to find the position and orientation of the end-effector from the position of the servo motors used at joints

of the kinematic chain. A block diagram for forward kinematics is shown in Fig. 5. There are two basic methods for solving forward kinematics problem. One of them uses graphical approach while other uses D-H parameters. Here discussion regarding graphical approach is carried out which is sufficient for training the membership function.

Fig. 5. Forward Kinematics Block Diagram. It describes the working of Forward Kinematics in a pictographic manner.

The graphical approach is the simplest approach for calculating the position of end-effector. Here the calculation is done only for the position of the end-effector and not for its orientation. In XY plane consider a generalized state of the kinematic chain. Fig. 6 shows the angles made by the links. Using these angles, with simple vector algebra, we can present the following equations;

$$x = l_1 cos\theta_1 + l_2 cos(\theta_1 + \theta_2) + (l_3 + d_4) cos(\theta_1 + \theta_2 + \theta_3) \quad (1)$$

$$y = l_1 sin\theta_1 + l_2 sin(\theta_1 + \theta_2) + (l_3 + d_4) sin(\theta_1 + \theta_2 + \theta_3) \quad (2)$$

Here l_1, l_2 and l_3 are the link lengths and θ_1, θ_2 *and* θ_3 are the angles made by these links as shown in Fig. 6. d_4 is the displacement in reference with the prismatic joint. These equations can be used to calculate the end-effector positionby using a numerical computing environment like Matlab.

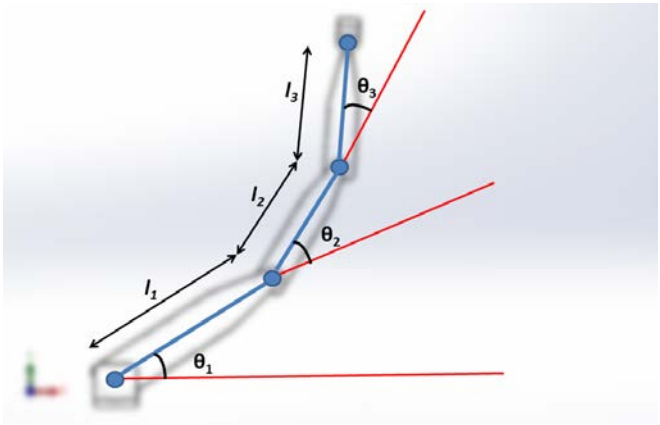

Fig. 6. Link Angles for Forward Kinematics. It displays generic orientation of the Robotic Arm and joint configurations in reference to the solution of Forward Kinematics.

```
function q = fk (theta1,theta2,theta3)
% Calculation for Forward Kinematics
%Inputs should be in radians

l1=0.35;
l2=0.2;
l3=0.07;
d4=0.008;

X = l1*cos(theta1)+l2*cos(theta1+theta2)+(l3+d4)*cos(theta1+theta2+theta3);
Y = l1*sin(theta1)+l2*sin(theta1+theta2)+(l3+d4)*sin(theta1+theta2+theta3);
q=[X Y];      % Answers are displayed in meters
```

Fig. 7. Matlab Script for Forward Kinematics. It displays a function which takes the joint configurations as the input and gives the co-ordinates of end-effector as output.

6.2 Inverse Kinematics

Inverse Kinematics is a type of Kinematics. It deals with the problems for finding joint configurations of the kinematic chain from the position and orientation of the end-effector of Robotic Arm. Usually programmer gives instructions to the robot arm to move the end-actuator to a specified point. To do this magnitude and direction of rotation has to given to the servo motors at each joint by the micro-controller. Thus the joint configurations are required to calculate in such situation. So inverse kinematics is used to find the joint configuration from the end-effector position and orientation given by the programmer. A block diagram for inverse kinematics is shown in Fig. 8.

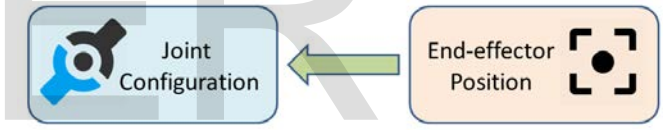

Fig. 8. Inverse Kinematics Block Diagram. It describes the working of Inverse Kinematics in a pictographic manner.

Inverse Kinematics is used for finding the joint angles for a specific position or orientation of the end-effector. In Fig.__ θ1 and θ2 are the joint angles for all mechanical configurations possible. Inverse kinematics is required for computation of agility at various points in the workspace. There may exist more than one possible solutions for a given problem when solved using inverse kinematics. In our problem we are only interested in the position of end-effector and not its orientation.

Let x and y be the co-ordinates of the end-effector in XY plane,

$$x = l_1 cos\theta_1 + l_2 cos(\theta_1 + \theta_2) + (l_3 + d_4) cos(\theta_1 + \theta_2 + \theta_3) \quad (3)$$

$$y = l_1 sin\theta_1 + l_2 sin(\theta_1 + \theta_2) + (l_3 + d_4) sin(\theta_1 + \theta_2 + \theta_3) \quad (4)$$

Here θ_1, θ_2 and θ_3 are the joint angles as referred in Fig. 9.

Rearranging the equations,

$$x - (l_3+d_4)\cos(\theta_1+\theta_2+\theta_3) = l_1\cos\theta_1 + l_2\cos(\theta_1+\theta_2) \quad (5)$$

$$y - (l_3+d_4)\sin(\theta_1+\theta_2+\theta_3) = l_1\sin\theta_1 + l_2\sin(\theta_1+\theta_2) \quad (6)$$

Let the left hand sides of equation__ and __ be x′ and y′ respectively for convenience.
On squaring and simplifying these equations, we get;

$$\Theta_2 = atan2\left(\frac{y'-(l^1s^1)}{l_2}, \frac{x'-(l_1c_1)}{l_2}\right) - \theta_1 \quad (7)$$

$$\Theta_1 = \gamma + \sigma\, arccos\left(\frac{x'^2+y'^2+l_1^2-l_2^2}{2l\sqrt{x'^2+y'^2}}\right) \quad (8)$$

$$Where,\ \gamma = atan2\left(\frac{-y'}{\sqrt{x'^2+y'^2}}, \frac{-x'}{\sqrt{x'^2+y'^2}}\right) \quad (9)$$

and, $\sigma = \pm 1$ (10)

Thus from equation 7 and 8 we can calculate the the joint angles for a particular position of the end-effector.

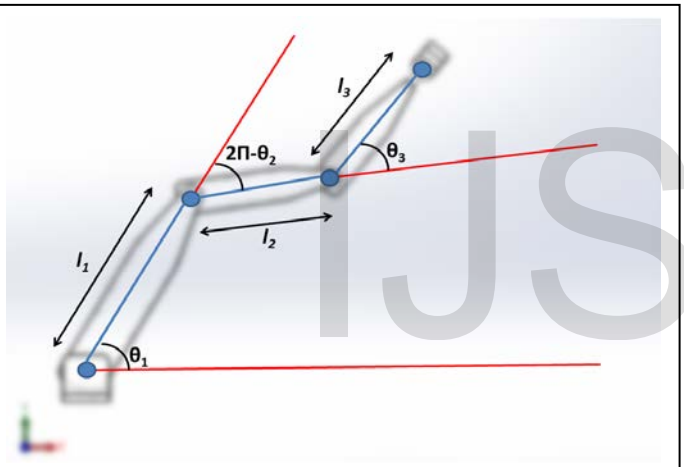

Fig. 9. Link Angles for Inverse Kinematics. It displays generic orientation of the Robotic Arm and joint configurations in reference to the solution of Inverse Kinematics.

```
function q = ik (x,y)
% Calculation for Inverse Kinematics

l1=0.35;
l2=0.2;
l3=0.07;
phi=atan(y/x);
Xi= x - l3*cos(phi);
Yi= y - l3*sin(phi);

gama= atan2((-Yi)/(sqrt(Xi^2+Yi^2)),(-Xi)/(sqrt(Xi^2+Yi^2)));
theta1= gama + acos((Xi^2 + Yi^2 + l1^2 - l2^2)/(2*l1*sqrt(Xi^2+Yi^2)));

sin1=sin(theta1);
cos1=cos(theta1);
theta2=atan2((Yi - l1*sin1)/(l2),(Xi - l1*cos1)/(l2));

theta3= phi - (theta1 + theta2);

q=[theta1 theta2 theta3];     % Answers are displayed in radians
```

Fig. 10. Matlab Script for Inverse Kinematics. It displays a function which takes the co-ordinates of end-effector as the input and gives the joint configurations as output.

7 Need For FIS

FIS stands for Fuzzy Inference system. This type of fuzzy logic is utilized in Robotic Arms with higher degrees of freedom. With more degrees of freedom, the motion of the Robotic Arm takes place in 3-dimensional space. The motion is not a simple planar motion. Thus the calculations for such Robotic Arms become complex with higher degrees of freedom. For instance when the Robotic Arm is instructed to move the end-effector from one position to another in the work envelope, the microcontroller has to calculate the angles of each joint that are required to reach the required point in the workspace. Thus it has to do calculations for inverse kinematics of the entire Robotic Arm. The solutions for such problems of inverse kinematics become complex with higher degrees of freedom. With increase in complexity of the calculations to be carried out by the controller, the time for computing also increases. Subsequently it increases the response time of the Robotic Arm which would eventually lead to decrease in productivity of a manufacturing process or assembly process. This is not desired as the main utilization of Robotic Arms is to improve productivity in industries. The fuzzy inference system is employed to address this problem. Fuzzy logic removes the need for carrying out complex calculations which result in slow response of the Robotic Arm. It attempts to create a simple mathematical function having all the parameters of the arm from a data set so that it becomes easy for the microcontroller of the robot. The microcontroller now just has to extract the values for a given input from the mathematical fuction created by using some algorithms.

8 FIS Method

The fuzzy inference system learns a mathematical function from a set of data. This data set consists of the required angles for achieving different points in the work envelope of the Robotic Arm. Thus from the data we can find angles corresponding to all points distributed in the work envelope. The distribution of the points can be changed as per the requirement of accuracy in the application. This data set which is used to train a mathematical function is created by using the forward kinematics as discussed previously in the paper. As it is observed, the solution for forward kinematics is quite simple as compared to the solution for inverse kinematics. Thus creating this data set is quite simple and not computationally expensive. A numerical compuating environment like Matlab or Octave might be used for this purpose.
Different algorithms can be used for training the required mathematical model. The adaptive neuro fuzzy inference system is used quite widely. It employs neural networks for training the mathematical model. Other algorithms like genetic algorithm can also be employed for this purpose. As it is a fuzzy logic, the solution provided by the mathematical function will not be perfectly accurate. There will be some error in the solutions. The magnitude of errors corresponding to different algorithms will be different. Thus performance of these algorithms is different.

9 ANFIS

ANFIS stands for Adaptive Neuro Fuzzy Inferenece System. It is a hybrid method for creating a fuzzy logic by using machine learning technique of neural networks. The Neural Networks are employed for lerning features or weights of a mathematical function from a given data set. The data set is used for learning and extracting the features. Thus a hypothesis is first created for the parametrized mathematical model and the data is used to train the mathematical model. Here the data set is created by using Forward Kinematics as the complexity of calculations is less and it is computationally not expensive. The Neural Network tunes the membership fuction of Sugeno type fuzzy inference system.

10 Sugeno Type FIS

There are two common types of the fuzzy inference system. One is Mamdani type fuzzy inference system. It is the most commonly used system. Other is Sugeno type fuzzy inference system which has been employed here. Sugeno type FIS is computationally efficient and compact in comparison to Mamdani type FIS. It gives better outputs with linear characteristics and has a guaranteed continuity of the output surface.
For a First Order Sugeno fuzzy model,
If

$$Input\ 1 = p \tag{11}$$

and

$$Input\ 2 = q \tag{12}$$

then Output is

$$z = ap + bq + c \tag{13}$$

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

$$w_i = f(F_1(x), F_2(y)) \tag{14}$$

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

$$\text{Final Output} = \frac{\sum_{i=1}^{N} w_i z_i}{\sum_{i=1}^{N} w_i} \tag{15}$$

where N is the number of rules.
Here for the analysis sugeno type of fuzzy inference system has been employed.

11 Creating Data Set

Creating data set is essential for training the membership fuction. As discussed, the data set will be created using Forward Kinematics. The data set is created by considering all the angles at different joints. Here the range for all rotating joints has been split into parts of 0.1 degrees and data is created for each of them. The equation 1 and equation 2 are used to calculate the position of end-effector. Thus a data set is created for end-effector positions corresponding to all angles. This data set is further given to ANFIS for tuning or training the membership function. The matlab script for creating a data sert is referred in Fig. 11.

```
l1 = 0.35;
l2 = 0.2;
l3 = 0.07;
THETA1 = pi/3:0.1:pi/2;
THETA2 = -pi/4:0.1:pi/4;
THETA3 = -pi/2:0.1:pi/2;

data1 = [];
data2 = [];
data3 = [];
for i=1:1:length(THETA1)
 for j=1:1:length(THETA2)
  for k=1:1:length(THETA3)
   X =l1*cos(THETA1(i))+l2*cos(THETA1(i)+THETA2(j))+l3*cos(THETA1(i)+THETA2(j)+THETA3(k));
   Y =l1*sin(THETA1(i))+l2*sin(THETA1(i)+THETA2(j))+l3*sin(THETA1(i)+THETA2(j)+THETA3(k));
   scatter(X,Y)
   hold on
   data1 = [data1; X Y THETA1(i)];
   data2 = [data2; X Y THETA2(j)];
   data3 = [data3; X Y THETA3(k)];
  end
 end
end
```

Fig. 11. Matlab Script for Creating Data Set. It displays the script for creating data set by using three iterative for loops.

12 Training ANFIS

The membership function has to be tuned or trained using the data set created previously. The number of ANFIS networks to be built is equal to number of joints we want to use as features for corresponding end-effector position. Here three ANFIS network are build for the joints R_2, R_3 and R_4. The prismatic joint T_1 and the revolute joint R_5 are not considered here as the primary interest is finding the end-effector position and not its orientation. Moreover its angle can be determined very simply so it is not included as a feature for tuning the membership function. Similarly calculating the configuration of the revolute joint R_1 is simple as it is the only joint with axis along Z axis. Thus to avoid complexity we can calculate it separately. To train the ANFIS networks x co-ordinates and y co-ordinates will be used as inputs and the joint configuration or the angle of corresponding joint will be used as the output. Here the values for number of membership functions used to characterize inputs and outputs and the number of training epochs are also to be decided. These values can be decided by trial and error method manually. One epoch of training represents one complete presentation of all the samples or datapoints of the training data set to the ANFIS. The number of membership functions is taken as 7 or 6 and the number of training epochs has been taken as 150. The matlab scripts for training all the three ANFIS networks are provided here. Fig. 12 referes to the matlab script of training ANFIS network for revolute joint R_2, R_3 and R_4.

```
21      %%
22
23 -    fprintf('-->%s\n','ANFIS network for R2 is training')
24 -    anfis1 = anfis(data1, 7, 150, [0,0,0,0]);
25 -    fprintf('-->%s\n','ANFIS network for R3 is training')
26 -    anfis2 = anfis(data2, 6, 150, [0,0,0,0]);
27 -    fprintf('-->%s\n','ANFIS network for R4 is training')
28 -    anfis3 = anfis(data3, 6, 150, [0,0,0,0]);
29 -    fprintf('-->%s\n','Training Over')
```

Fig. 12. Matlab Script for training ANFIS. It displays the script for training three ANFIS networks for three different joints.

13 Validation Of Model

After training an ANFIS model, the model has to be tested to verify its performance. It gives an idea about the accuracy of the outputs of trained ANFIS model. In this manner, validation of the model is carried out. To check the accuracy, outputs from the ANFIS model are be compared to the actual values. The best way to do this is to compare the values of output from ANFIS model with the values that are obtained from inverse kinematics of the Robotic Arm. The calculations for inverse kinematics which are discussed earlier in this paper are to be used to calculate actual values and these are compared with the values from ANFIS model. The error is calculated as simple difference of the value obtained from ANFIS model and the value obtained from inverse kinematics.

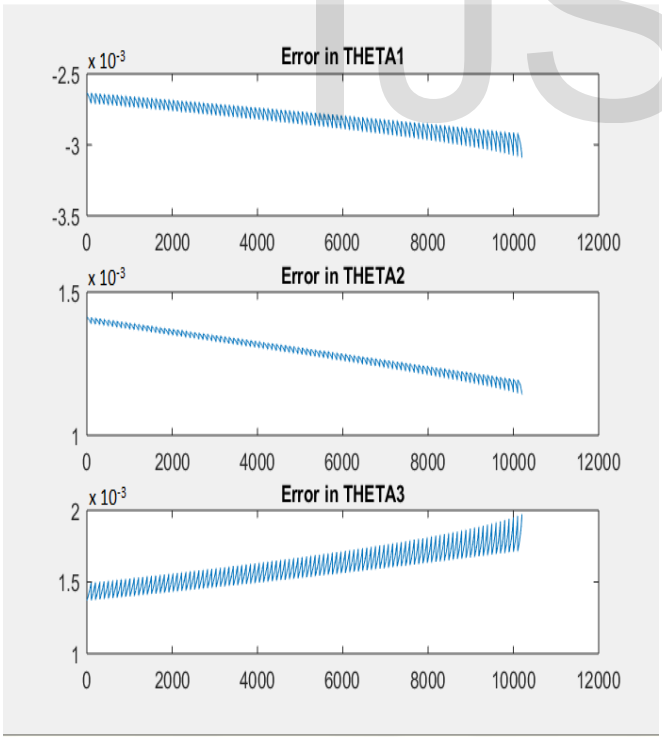

Fig. 13. Validation of Model. It displays the inherent error or deviation in the trained ANFIS network for three joints.

14 Error

Let the value of joint configuration for revolute joint R2 be θ_2. Now the value of θ_2 obtained from the ANFIS model is denoted by $(\theta_2)_{FIS}$ and the value of θ_2 obtained from the inverse kinematics solution is denoted by $(\theta_2)_{IK}$. As error is the simple difference, it can be given as,

$$\text{Error for } R_2 = (\theta_2)_{IK} - (\theta_2)_{FIS} \quad (16)$$

Similarly the errors for other joints can be given as,

$$\text{Error for } R_3 = (\theta_3)_{IK} - (\theta_3)_{FIS} \quad (17)$$

$$\text{Error for } R_4 = (\theta_4)_{IK} - (\theta_4)_{FIS} \quad (18)$$

The errors will be considered only in the region of interest in the work envelope. Thus the errors for each joint will be considered for a specific range of x co-ordinates and y co-ordinates. Thus the error for each joint can be plotted in three dimensional space against x co-ordinate and y co-ordinate. The value of error obtained at different joints for different configurations, may be positive or negative.The plots of error for each of the four joints considered in ANFIS network are provided here. Fig. 13 refers to the error plot for the revolute joint R_2, R_3 and R_4.

```
50      %%
51 -    XY = [X(:) Y(:)];
52 -    THETA1P = evalfis(XY, anfis1);
53 -    THETA2P = evalfis(XY, anfis2);
54 -    THETA3P = evalfis(XY, anfis3);
55
56 -    theta1diff = THETA1D(:) - THETA1P;
57 -    theta2diff = THETA2D(:) - THETA2P;
58 -    theta3diff = THETA3D(:) - THETA3P;
59
60 -    subplot(3,1,1);
61 -    plot(theta1diff);
62 -    title('Error in THETA1','fontsize',10)
63
64 -    subplot(3,1,2);
65 -    plot(theta2diff);
66 -    title('Error in THETA2','fontsize',10)
67
68 -    subplot(3,1,3);
69 -    plot(theta3diff);
70 -    title('Error in THETA3','fontsize',10)
```

Fig. 14. Matlab Script for Plotting Error. It displays the script for calculating error for three joints and creating three plots respectively.

15 Accumulation of Error

The error exists at all joints of the Robotic Arm. The errors calculated previously at different joints have different effect of error at the end-effector position. For instance error at the revolute joint R2 will create a greater error at the end-effector position as compared to the error at the revolute joint R4. The local position error for each joint can be given as product of the angle error and link length.

$$\text{Error})_p \text{ for } R_2 = [\,(\theta_2)_{IK} - (\theta_2)_{FIS}\,] * L_1 \quad (19)$$

$$\text{Error})_p \text{ for } R_3 = [\,(\theta_3)_{IK} - (\theta_3)_{FIS}\,] * L_2 \quad (20)$$

$\text{Error})_p \text{ for } R_4 = [\ (\theta_4)_{IK} - (\theta_4)_{FIS}\ ] * L_3$ *(21)*

Here L_1, L_2 and L_3 can be considered as weights given to the error according to their effect on the error of the end-effector position. As Robotic Arm is an open kinematic chain, the error will be accumulative. To keep the analysis simple we will consider the accumulated error as the sum of position errors for all joints.

$$E = \text{Error})_p \text{ for } R_2 + \text{Error})_p \text{ for } R_3 + \text{Error})_p \text{ for } R_4 \quad (22)$$

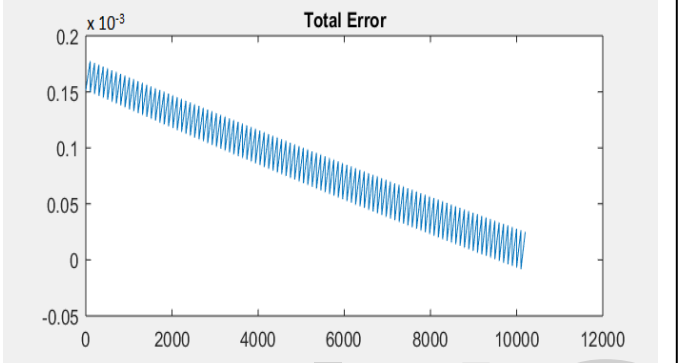

Fig. 15. Total Error. It displays the error resulted by simple addition of the errors at each joints.

Here the error will be sensitive to the direction of rotation of the revolute joint. If the direction of two consecutive revolute joints is in the same direction, then the error will be added. But if the direction of rotation of these consecutive joints is opposite, then the error will be difference. If the direction of rotations are same, then it is advantageous if the values of errors have opposite signs (+ or -) as it will give less total error. Similarly if the directions of rotations are opposite, it is advantageous if the values of errors have same sign. But to understand the direction of rotation, the previous configuration of the joint must be known. Suppose the link roatates about R2 from θ_{21} to θ_{22} and a link rotates about R3 from θ_{31} to θ_{32}. Thus it can be given,

If $[\ (\theta_{21} > \theta_{22})$ and $(\theta_{31} > \theta_{32})\ ]$ or $[\ (\theta_{21} < \theta_{22})$ and $(\theta_{31} < \theta_{32})\ ]$

Then E = $\text{Error})_p \text{ for } R_2 + \text{Error})_p \text{ for } R_3$

ElseIf $[\ (\theta_{21} > \theta_{22})$ and $(\theta_{31} < \theta_{32})\ ]$ or $[\ (\theta_{21} < \theta_{22})$ and $(\theta_{31} > \theta_{32})\ ]$

Then E = $\text{Error})_p \text{ for } R_2 - \text{Error})_p \text{ for } R_3$

Now the generalized form for this logic can be given by,
If $[\ (\theta_{i1} > \theta_{i2})$ and $(\theta_{(i+1)1} > \theta_{(i+1)2})\ ]$ or $[\ (\theta_{i1} < \theta_{i2})$ and $(\theta_{(i+1)1} < \theta_{(i+1)2})\ ]$

Then E = $\text{Error})_p \text{ for } R_i + \text{Error})_p \text{ for } R_{i+1}$

ElseIf $[\ (\theta_{i1} > \theta_{i2})$ and $(\theta_{(i+1)1} < \theta_{(i+1)2})\ ]$ or $[\ (\theta_{i1} < \theta_{i2})$ and $(\theta_{(i+1)1} > \theta_{(i+1)2})\ ]$

Then E = $\text{Error})_p \text{ for } R_i - \text{Error})_p \text{ for } R_{i+1}$
Fig. 16 refers to the plot of accumulated error.

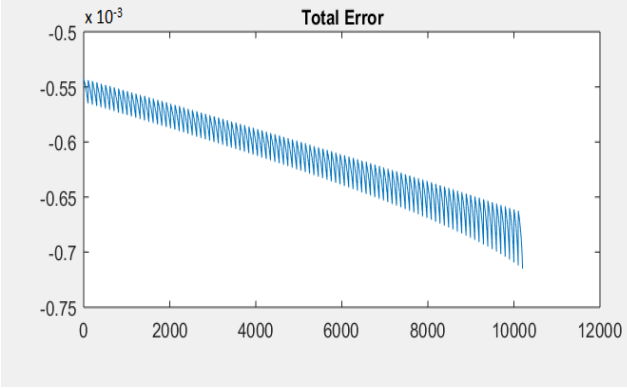

Fig. 16. Total Error. It displays the error resulted by addition of the errors at each joint by using linkage length as weights.

16 Reducing Error

The ANFIS network was trained with condition of reducing the squared error over the entire range of values. Usually there is a region of interest in the work envelope where maximum accuracy is required. But the ANFIS network reduces the values over entire region. Thus the learned features of the membership function have to be tweaked slightly in order to get minimum error in the region of interest.
The values are tweaked for better performance of the Robotic Arm at particular region of interest. Here the ranges of joint configurations for selected region of interest are given in Table 2. The attempt to reduce error is done for these ranges.

TABLE 2
UNITS FOR MAGENTIC PROPERTIES

Joint	Range
R_2	-55 to -60
R_3	-35 to -40
R_4	-45 to -50

The notations R_2, R_3 and R_4 represent the joints for which ANFIS network is trained. The Range is given in degrees.

The details regarding the finalized ANFIS network for anfis1 are reffered in Fig. 17.

```
1.  Name             anfis1
2.  Type             sugeno
3.  Inputs/Outputs   [2 1]
4.  NumInputMFs      [7 7]
5.  NumOutputMFs     49
6.  NumRules         49
7.  AndMethod        prod
8.  OrMethod         max
9.  ImpMethod        prod
10. AggMethod        sum
11. DefuzzMethod     wtaver
```

Fig. 17. Final ANFIS. It displays the details regarding final anfis1 after tweaking required parameters.

17 Softwares Used

Software is required for generating a 3d CAD model of Robot Arm and for computation, calculation and creating a graphical user interface.

17.1 Solidworks

Solidworks is a 3d CAD modelling software used here for generating CAD model of the 6 DOF Robotic Arm. It is used to define the geometric properties, material properties and for assembly of the different components of the robot.

17.1 Matlab

Matlab is a numerical-computing environment. Here it has been employed for calculating and solving the problems of forward kinematics and inverse kinematics. It has also been employed for training ANFIS, calculating errors and plotting respective graphs.

18 Conclusion

The paper puts forward solution of kinematic problem for a 6 DOF Robotic Arm by using Adaptive Neuro Fuzzy Inference System. The ANFIS is trained using the data obtained from Forward Kinematics to create a membership function for each joint. This membership fuction avoids the complex calculations related to the inverse kinematics. The error prevalent in the membership function can be reduced for a particular region of interest. Thus use of ANFIS reduces computational cost and time which in turn is helpful in improving the productivity of manufacturing processes that employ Robotic Arms.

References

[1] C. Loganathan 'Hybrid Learning for Adaptive Neuro Fuzzy Inference System' International Journal of Engineering and Science Vol. 2 Issue 11 (April 2013) Pp 06-13 Issn(e): 2278-4721, Issn(p): 2319-6483.

[2] Jiin-Po Yeh "Comparison between Neural Network and Adaptive Neuro-Fuzzy Inference System for Forecasting Chaotic Traffic Volumes" Journal of Intelligent Learning Systems and Applications, 2012, 4, 247-254 (November 2012).

[3] R.Pushpavalli "Image Enhancement Using Adaptive Neuro-Fuzzy Inference System" International Journal of Scientific & Technology research" Volume. 2, Issue 6, June 2013 ISSN 2277-8616.

[4] Dinesh C. S. Bisht "Discharge Modelling using Adaptive Neuro - Fuzzy Inference System" International Journal of Advanced Science and Technology Volume. 31, June, 2011.

[5] Hazlina Hamdan "An Exploration of the Adaptive Neuro-Fuzzy Inference System (ANFIS) in Modelling Survival" School of Computer Science The University of Nottingham Nottingham, United Kingdom March 2013.

[6] N. Sarikaya "Adaptive neuro-fuzzy inference system for the computation of the characteristic impedance and the effective permittivity of the micro-coplanar strip line" Progress In Electromagnetics Research B, Vol. 6, 225–237, 2008.

[7] Mrinal Buragohain "Adaptive Network based Fuzzy Inference System (ANFIS) as a Tool for System Identification with Special Emphasis on Training Data Minimization" Department of Electronics and Communication Engineering Indian Institute of Technology Guwahati July, 2008.