Intelligent Adaptive Systems

Assignment
1. Balancing cart-pole using fuzzy logic.
2. Training a Fuzzy inference system to predict inverse kinematics of a planar RRR serial robot.

Tuesday, 18th April 2017

Edited by

Asheesh Sharma

Candidate: 33074, Student: 1562159

University of Bristol

United Kingdom

/Asheeshkrsharma/RRP-Assignment.git

2017

Contents

Cart pole dynamics

Objective

Design and implement a Mamdani-style Fuzzy Controller to control a simulated cart-pole (inverted pendulum) system.

Introduction

Balancing a broomstick in an upright position requires coordinated horizontal movement of hand based on the visual queues. This nonlinear and inherently unstable control problem is often considered to be a primary benchmark for evaluating the performance and response of new control methods because of its structural simplicities. The same problem is also known as an inverted pendulum [1]. A two-dimensional inverted pendulum consists of a vertical pole freely hinged to a platform which can be driven in the horizontal direction using a belt or cart system.

System description

The cart-pole system shown in Figure 1 consists of a moving cart with a pole hinged to its center of mass. The mass of pole includes any payload which may be attached to the free end. Various properties of the system are specified as follows:

m_c: cart mass
m_p: pendulum mass
l: distance from hinge to pendulum's COG
l_p: pendulum's inertia about its COG
g: gravitational acceleration
x: displacement of cart's center of mass from origin
θ: angle between pendulum and the vertical axis
f: Force applied to the cart

The Lagrangian method is often used to determine the dynamic behavior of a system in terms of work done and the energy stored. This requires describing separable rigid bodies independently [2]. First, the pendulum's COG is described as a Cartesian Lagrangian coordinates. From law of cosines,

$$x_G = x + l\sin(\theta) \cdots 1$$

$$y_G = l\cos(\theta) \cdots 2$$

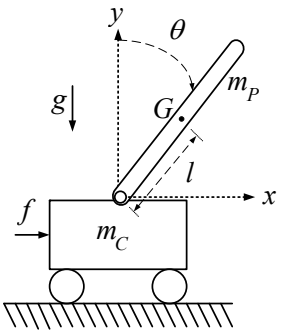

Figure 1 A simplified cartpole schematic

Using equations 1 and 2, the kinetic energy of the pendulum can be defined as

$$K_p = \frac{1}{2}m_p\dot{x}_G^2 + \frac{1}{2}m_p\dot{y}_G^2 + \frac{1}{2}I_p\dot{\theta}^2 \cdots 3$$

It can bee seen that the kinetic energy is dependent on both the cart position and pole angle ($\dot{x}$ and $\dot{\theta}$). However, the potential energy of the pendulum is only dependent on the pole angle.

$$P_p = m_p g l(\cos\theta - 1) \cdots 4$$

Next, the kinetic energy of the cart is simply related to the displacement in the cart position.

$$K_c = \frac{1}{2}m_c\dot{x}^2 \cdots 5$$

Finally, the Lagrangian is a scalar which connects the kinetic energy with potentially as L=K-P, where K and P are total kinetic and potentially energy of the system respectively. Thus,

$$\implies L_{cp} = K_c + K_p - P_p$$

substituting equations 5,4 and 3 in the Lagrangian relation yields,

$$L_{cp} = \frac{1}{2}\left[(m_c + m_p)\dot{x}^2 + (l_p + m_p l^2)\dot{\theta}^2\right]$$

$$+m_p l\left[\dot{x}\dot{\theta}\cos\theta - g(cos\theta - 1)\right]$$

The non-conservative generalized force (ψ) acting on the entire system can then be determined by the Euler-Lagrange equation as

$$\psi = \frac{d}{dt}\left(\frac{\partial L_{cp}}{\partial \dot{q}}\right) - \left(\frac{\partial L_{cp}}{\partial q}\right)$$

therefore,

$$\psi = M(q)\ddot{q} + B(q,\dot{q})\dot{q} + G(q)$$

where,

$$q = \begin{bmatrix} x & \theta \end{bmatrix}^T \quad \psi = \begin{bmatrix} f & 0 \end{bmatrix}^T$$

$$M(q) = \begin{bmatrix} m_c + m_p & m_p l\cos(\theta) \\ m_p l\cos(\theta) & I_p + m_p l^2 \end{bmatrix}$$

,

$$B(q,\dot{q}) = \begin{bmatrix} 0 & -m_p l\dot{\theta}\sin(\theta) \\ 0 & 0 \end{bmatrix}$$

and,

$$G(q) = \begin{bmatrix} 0 \\ -m_p g l\sin(\theta) \end{bmatrix}$$

It can be observed that the dynamics of even a simple inverted pendulum is intuitively hard to understand and more importantly to analyze. There are many methods to derive the dynamics of a system but most of them require elemental derivatives which may not be easy to practically calculate [3], [4]. Furthermore, there are other similar solutions (like partial feedback linearization [5]) heavily rely on linearizing the system dynamics. Thus, it is easier to use a fuzzy control for such systems. Lin and Sheu [6] proposed a control system in which they used linear state feedback control to switch b/w swing up fuzzy control and stabilization. However, it lacked a responsive position control. To achieve position control many techniques have been proposed which involves switching b/w two different fuzzy logic controllers (one for balancing the pole and other to reach a target). Li and Shieh [7] called this the departure and approach mode control. Other than designed controls, many other methods have also been proposed which include training a fuzzy system using neural networks and reinforcement learning. However, such learning methods heavily rely on many tuning parameters and often result in too many inference rules which are hard to intuitively understand (thus defeating the purpose of fuzzy control in the first place). E.g. the reinforcement learning methods proposed by Jun et al. [8] resulted in 63 rules.

Based on these key problems, the following study is aimed towards following goals.

1. To develop a simple method to combine the position and balancing control in a single fuzzy inference system.

2. To minimize the number of rules and membership functions.

A generic fuzzy inference system

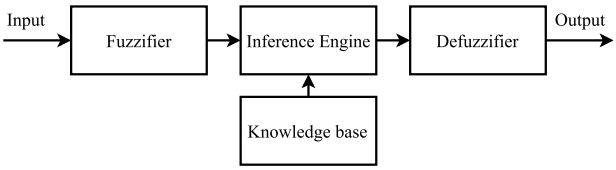

Figure 2 Generic Fuzzy inference system.

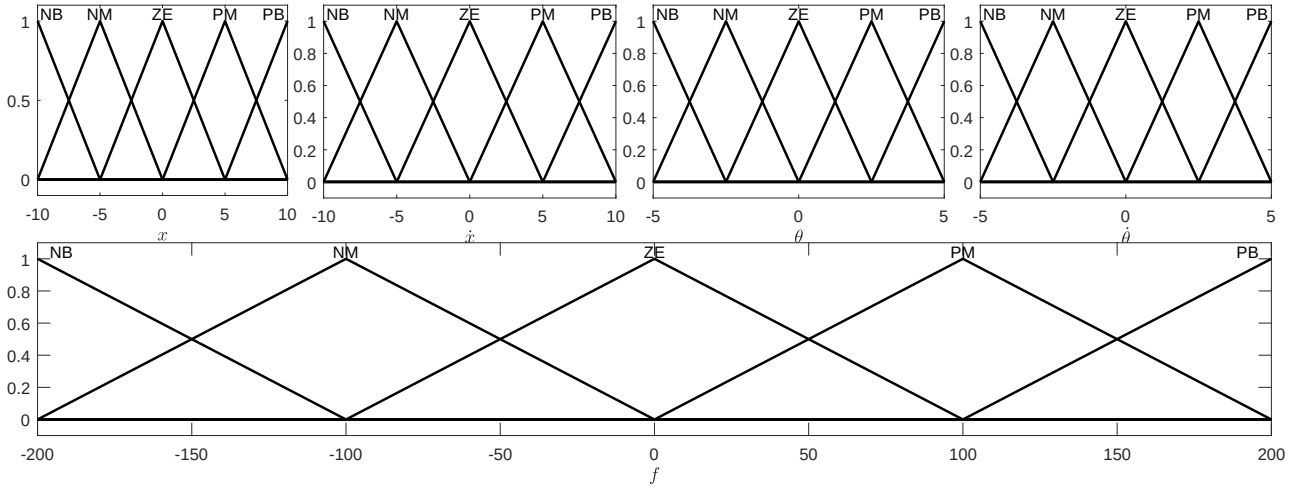

Figure 3 Memberships of separate inputs for Experiment 1.

Figure 2 shows the basic structure of a fuzzy logic controller. The main building units of an FLC are a fuzzification unit, a fuzzy logic inference unit, a rule-base, and a defuzzification unit.

The purpose of fuzzification is to map the inputs from a set of sensors (or features of those sensors such as amplitude or spectrum) to values from 0 to 1 using a set of input membership functions. These input membership functions, can represent fuzzy concepts such as "left" or "right", "fast" or "slow", "hot" or "cold", etc. For the cart pole balancing, the pole angle and cart position can be represented as "Negative big (NB)", "Negative medium (NM)", "Zero/Stable (ZE)", "Positive medium (PM)" and "Positive Big". Therefore, the process of fuzzification provides convenience by allowing the designer to use the same definition of what is meant by "Negative big" or "Positive big" with different input membership functions which may have different operational ranges.

Next, the fuzzified inputs must be combined according to the fuzzy rules to establish a rule strength. This process is handled by the inference engine and knowledge base units together. Although there are many ways to combine memberships and rules, the Zadeh-and' method (equation 6) is used in this case study.

$$\mu_{A\cap B} = min(\mu_A(x), \mu_B(x)) \cdots (6)$$

Where μ_A is read as "the membership in A" and μ_B is read as "the membership in B".

Finally, defuzzification is the process of converting inferred fuzzy control actions into a crisp control action. Fuzzy reasoning is used to infer the output contributed from each rule. The fuzzy outputs from each rule are aggregated and fuzzified to generate a crisp output. The essence of a fuzzy logic controller is thus based on a linguistic model (rule base and the defined membership functions) as opposed to a mathematical model.

Experiment 1

The objective of this experiment was to develop inference systems for balancing and positioning separately. Secondly, to develop a switching system which decides which control to switch-off based on the pole angle. Both the fuzzy controllers, have two set input input variables ($[\theta, \dot{\theta}]$ and $[x, \dot{x}]$). Both sets of fuzzy inferences share the same output functions illustrated in figure 2.

To design the fuzzy controllers, 5 membership functions are assigned to each input and output state variables. Triangular membership functions are used for all membership functions. "Center of Gravity" (COG) defuzzification method is used to combine the rules represented in table 1.

Table 1 Inference rules

θ, x \ $\dot{\theta}, \dot{x}$	NB	NM	ZE	PM	PB
PB	ZE	PM	PM	PB	PB
PM	NM	Z	PM	PM	PB
ZE	NM	NM	ZE	PM	PM
NM	NB	NM	NM	ZE	PM
NB	NB	NB	NM	NM	ZE

In table 1, NB, NM, ZE, PM, PB refer to "Negative big","Negative medium", "Zero","Positive medium","Positive Big" respectively. The inference rules (specially the nomenclature) are shared in many similar systems ([3], [4] and [9]). The rules can be intuitively interpreted as follows.

For [θ, $\dot{\theta}$] controller, Suppose the cart was stationary (not moving horizontally) and inclined towards the left. Intuitively, the cart should be moved left. Furthermore, the speed by which the cart should be moved is based on its membership of $\dot{\theta}$. For instance, if θ was NB and $\dot{\theta}$ was NM, then a large amount of force is needed in the negative/left direction. Therefore, the output is NB.

For [x, $\dot{x}$] controller, Suppose x is right and $\dot{x}$ is positive (i.e the cart is moving towards right), then the applied force should be right so that the pendulum moves to the left thus stabilizing the system. A similar rule would work when the x is left and $\dot{x}$ is negative (i.e. the cart is moving towards the left).

Using these insights, all others rules can also be derived. In the initial experiments, it was found that the cart kept oscillating even when the desired position had reached. This was because of two reasons. Firstly, due to no overlapping regions in the memberships of NM and PM Figure 3. Secondly, due to lack of a membership function for a stable position. To resolve this issue, either an overlap was needed between NM and PM or a stable state membership function. A ZE membership function was introduced (a design choice).

At this point, it was obvious that the individual controllers although have responsive output to stimulus, they exhibit no cooperative control.

Therefore, a criterion is needed to bias the control according to the situation which needs immediate attention. For instance, if the pole angle is "dangerously" negative, the control should be biased towards the $[\theta, \dot{\theta}]$ controller. Similarly, if the target position is very far away from the current position, the control should be biased towards the $[x, \dot{x}]$ controller. One way to implement this is to have a third inference system which decides the switching based on x and the θ as follows:

1. if the angle is big and position difference is small, control is $[\theta, \dot{\theta}]$. 2. if the angle is small and position difference is big, control is [x, $\dot{x}$]. 3. if the angle is small and position difference is small, control is [x, $\dot{x}$]. 4. if the angle is big and position difference is big, control is $[\theta, \dot{\theta}]$.

Another key observation is that the angular control of the pendulum must be done with priority over the position control of the cart in order to achieve complete stabilization of the whole inverted pendulum system. When the pendulum is not located in the upright position yet, the angular control of the pendulum is first executed so that the pendulum is balanced almost upright. While the balanced state of the pendulum is kept, starting the position control of the cart is allowable. Due to this fact, it can be seen is that priority is always is to balancing the cart pole over reaching the target position. Therefore a simple switch solely based on the angle can be employed to achieve the control.

Conclusions

It was generally observed that large oscillation occurred when the position and angle control alternated to totally opposite outputs. Furthermore, it was harder to determine what was affecting the system due to too many governing rules and membership functions. Thus a simpler and more effective control was developed which inferences the output based on the angle and position in a single fuzzy inference system.

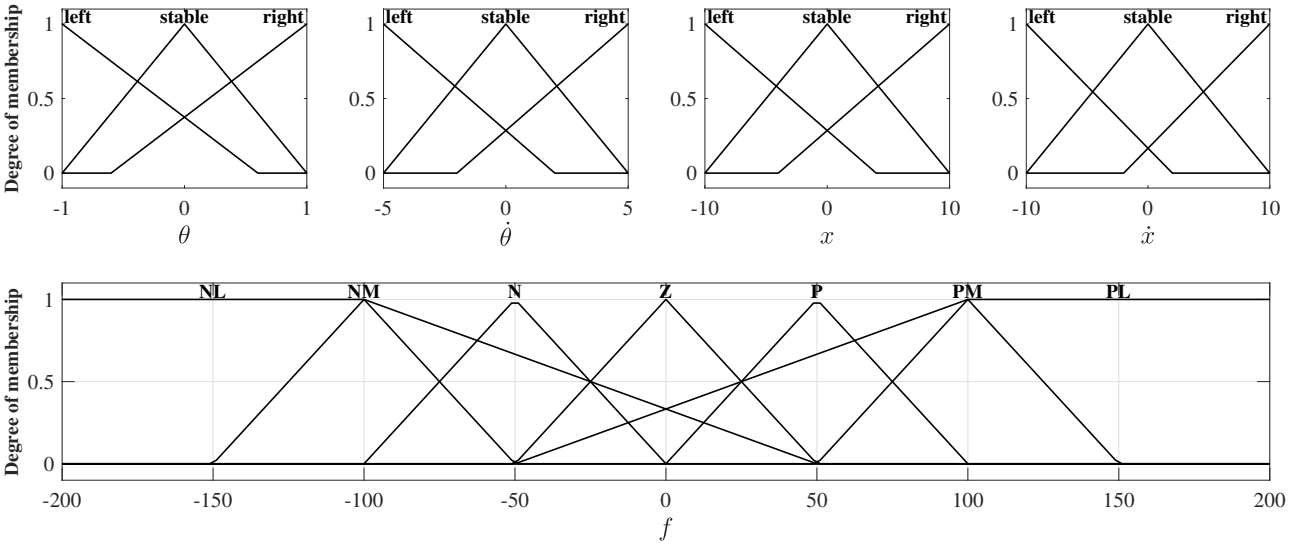

Figure 4 A simplified cartpole schematic

Experiment 2

To combine the individual knowledge base of the separate control systems, new membership functions were constructed (as shown in th figure) based on the following key observations.

1. From the aforementioned initial experiments, it was understood that the out of the two inference systems, the degree of importance of balancing has to be larger than positioning in order to stabilize the whole pendulum system. At the same time, the importance degree of the pendulum angle when the pendulum angle is big should almost be the same as the importance degree of the angular velocity when the angular velocity is big. The same fact also applies to cart position and speed. That is, the importance degree of the cart position when the positive is big should almost be the same as the importance degree of the cart velocity when the velocity is big.

To achieve this bias, the position control share only the most contrasting membership functions (NL and PL) in the output (observe the flat surfaces in figure 5). It was found that having no overlap between these NL and PL in the output introduced a constant time lag in the response to target position.

It was also found that the time lag was directly proportional to an overlap between NL and PL in the output. Therefore a small overlapping region has been introduced. This also serves the purpose of having a decaying control over the position as the angle control needs to be more dominating. Table 2 shows the three rules employed in achieving the position control.

Table 2 x control rules

x \ $\dot{x}$	Left	Stable	Right
Left	NL	-	-
Stable	-	Z	-
Right	-	-	PL

2. It was empirically found that angle control required a much finer control over the output than the position control (thus, the surface plot in figure 5 is smooth). Therefore, two modifications were made. Firstly, exclusive output membership functions were constructed to achieve finer control over the balancing. Table 3 shows the nine rules used for governing the output. It can also be observed that "left" and right membership functions in all inputs share a small overlapping region centered at the "stable" membership function to achieve a smoother output.

Table 3 θ control rules

θ \ $\dot{\theta}$	Left	Stable	Right
Left	NL	NM	N
Stable	N	Z	P
Right	P	PM	PL

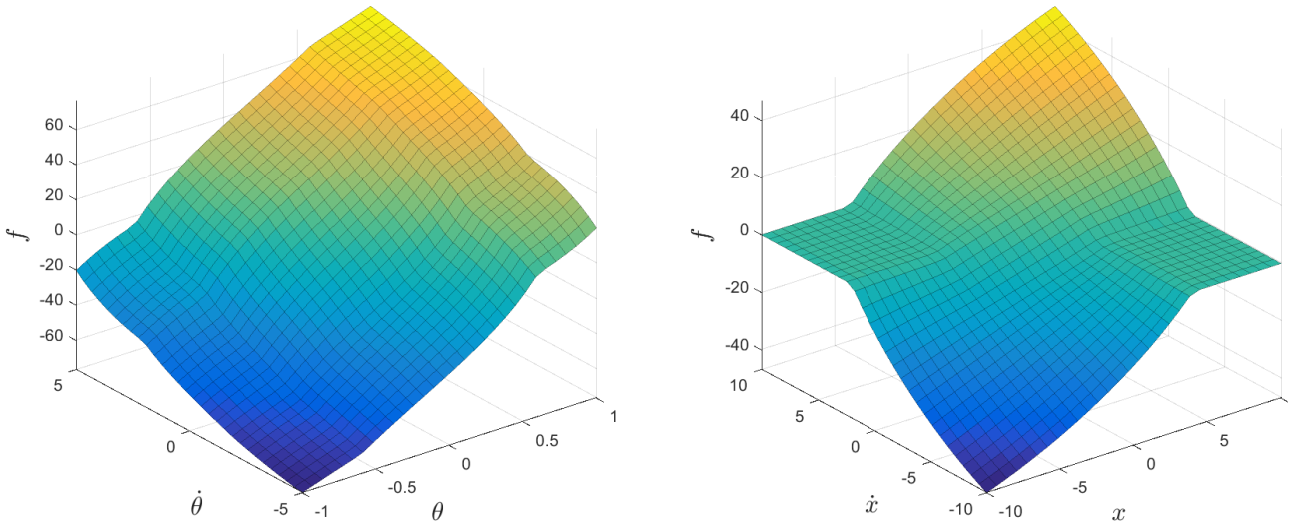

Figure 5 Surface of rule base

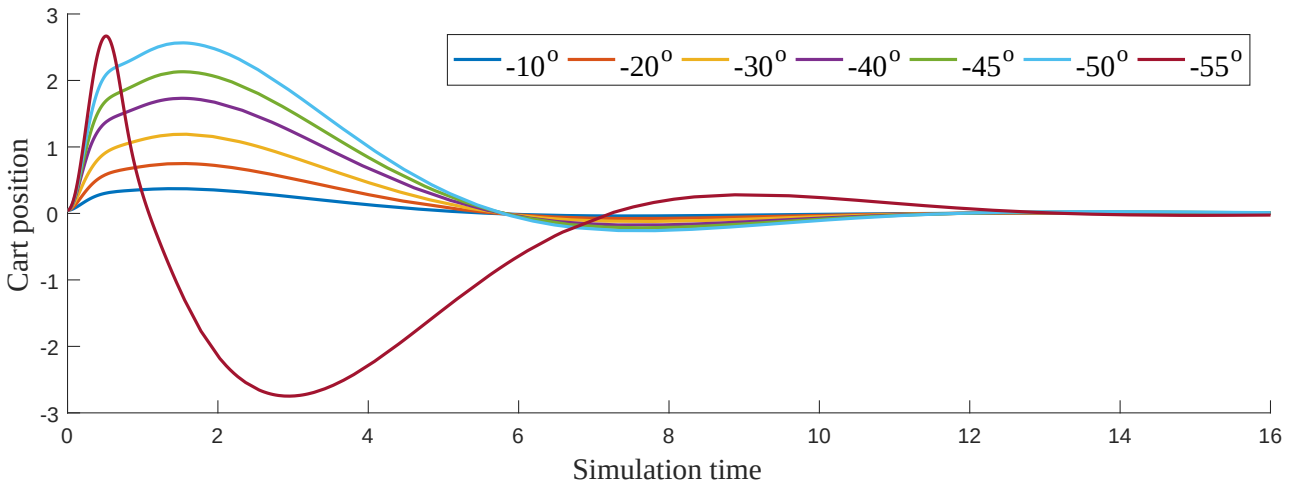

Figure 6 Impulse response with different initial pole angles

Results

To verify the effectiveness of the proposed fuzzy controller, the control system was stimulated with different situations. Firstly, the response time was observed. Figure 5 shows the reaction time observed with increasing initial pole angles. It can be seen that the control system is able to stabilize quickly[1]. Figure 5 also shows the introduced bias in action when the angle was very big (-55^o). To observe the bias in much more clarity, the system was initialized with a pole angle of 55^o and a target position of zero (Figure 6). It can be observed that the pole angle decreases faster (becomes upright faster) than reaching the target position.

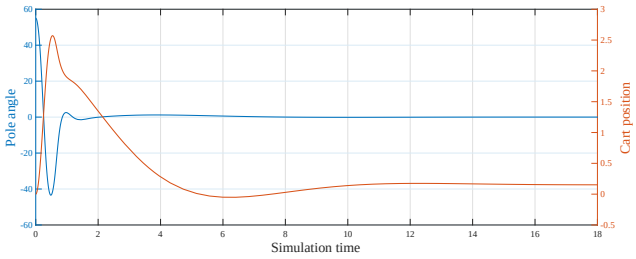

Figure 7 Control result of the pendulum system for initial angle 55^o

At the same time, the control is also able to enforce position control over pole control when the distance from target becomes very large because of the decaying memberships of NL and PL in the fuzzy outputs. This is illustrated in Figure 8.

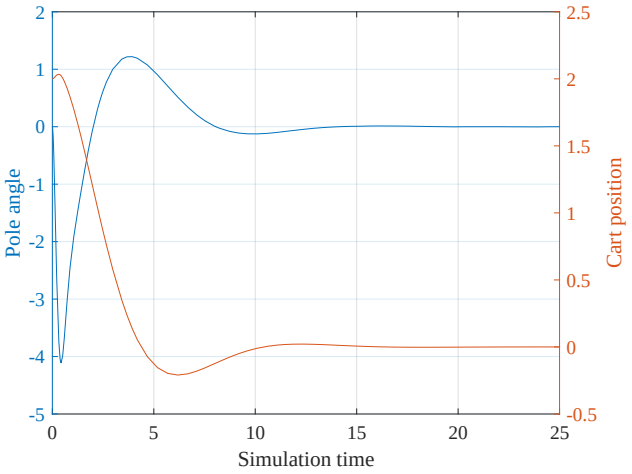

Figure 8 Control result of the pendulum system for initial position 2 m

Finally, the control system is able to desirably respond to extreme contrasting conditions (initial conditions of 55^o and 2 m) as shown in Figure 9.

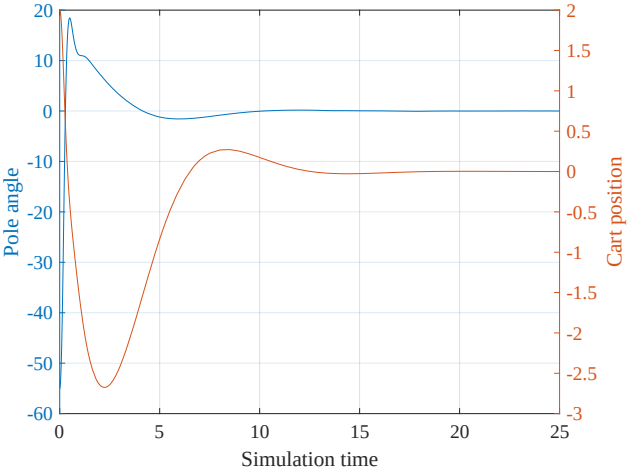

Figure 9 Control result of the pendulum system for initial angle -50^o and intial position of 2 m.

[1] Note that figure 5 represents simulation result, therefore the time is not in seconds

RRR serial link manipulator

Introduction

In rigid body robotics, inverse and forward kinematics are the most common subjects which have been well discussed in standard textbooks. The forward kinematics of a robotic manipulator deals with determining the position of end-effector given the joint variables. On the other hand, the inverse kinematics of robotics manipulator deals with the inverse problem. Solving inverse kinematics for a robotic arm or serially linked manipulator (i.e. only one base) are difficult to solve as the degrees of freedom increase. Furthermore, due to non-linearity between joint-space and Cartesian space, multiple solutions can exist. Therefore, the required calculations are often computationally intensive and not guaranteed to converge. For instance, one phenomenon called the singularity involves positions which cannot be attained because the links collide or attain a position where at least one joint has to move with infinitesimal angular velocity to maintain the course of trajectory.

The literature involving inverse kinematics is extensive and thus a variety of practical approaches to problems like singularity have been well studied [10],[10], . These include numeric, iterative, geometric and algebraic approaches. Out of them, iterative methods are most suitable for real-time control. These methods involve predicting the joint variables rather than actually solving for them. For instance, neural networks can be trained with large data set of joint variables and end effector coordinates (using forward kinematics) to predicted the inverse kinematics.

This study presents a similar approach using fuzzy logic and neural networks for a three degree of freedom robot. ANFIS (Adaptive Neuro-Fuzzy Inference Systems) has been used to train and improve fuzzy inference systems for the purpose of inverse forward mapping.

Forward kinematics of a 3 DOF robot

In this section, the forward kinematic equations for 3(RRR) Degree of freedom serial linked arm is developed. The forward kinematics problem is concerned with the relationship between the individual joints of the robot manipulator and the position and orientation of the tool or end effector. The joint variables are the angles between the links in the case of revolute or rotational joints and the link extension in the case of prismatic or sliding joints. In trivial robots like the two-link planar manipulator, it is easy to solve forward kinematics by simply shifting (translating and rotating) the origin frame to the joint frames towards the end-effector. However, with complex manipulators, this becomes a tedious task. Thus some conventions are necessary while defining frames for individual joints to make the process of shifting frames easy. A commonly used convention for selecting frames in robotic applications is the Denavit-Hartenberg, or DH convention [11]. In this convention, each homogeneous transformation is represented as a product of four basic transformations. According to the convention, the frames should be defined such that all the translations and rotations exist only in z_i and x_i axis. Through this careful selection, it can be realized that the DH representation allows us to calculate the homogeneous transformation matrix using just four parameters. This happens automatically because we have a considerable freedom to choose the placement of origin and the coordinate axis of the frames. In fact, it is not even necessary that a frame resides inside the physical link. To keep the derivation of brief, further discussion on convention will be skipped.

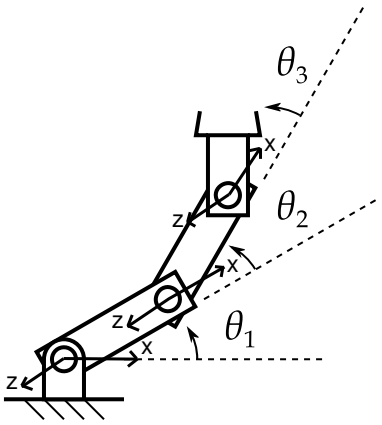

Figure 10 3RRR planar arm

In figure 10, the z-axis is pointing out of the papers and the x-axis pointing to the next link. The Dh parameter can be derived as:

Table 4 Dh standard paramenters for the three joints

Joint	θ	d	a	α
1	θ_1	0	l_1	0
2	θ_2	0	l_2	0
3	θ_3	0	l_3	0

Using these, we can write the homgenous transformations as follows:

$$T_1 = \begin{bmatrix} Cos\theta_1 & -Sin\theta_1 & 0 \\ Sin\theta_1 & Cos\theta_1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \cdots 7$$

$$T_2 = \begin{bmatrix} Cos\theta_2 & -Sin\theta_2 & l_1 \\ Sin\theta_2 & Cos\theta_2 & 0 \\ 0 & 0 & 1 \end{bmatrix} \cdots 8$$

$$T_3 = \begin{bmatrix} Cos\theta_3 & -Sin\theta_3 & l_2 \\ Sin\theta_3 & Cos\theta_3 & 0 \\ 0 & 0 & 1 \end{bmatrix} \cdots 9$$

$$T_4 = \begin{bmatrix} 1 & 0 & l_3 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \cdots 10$$

equations 7-10 can jointly be written as

$$T_4^1 = \begin{bmatrix} C_{123} & -S_{123} & l_1C_1 + l_2C_{12} + l_3C_{123} \\ S_{123} & C_{123} & l_1S_1 + l_2S_{12} + l_3S_{123} \\ 0 & 0 & 1 \end{bmatrix}$$

Which implies

$$x = l_1Cos(\theta_1) + l_2Cos(\theta_1+\theta_2) + l_3Cos(\theta_1+\theta_2+\theta_3)$$

$$y = l_1Sin(\theta_1) + l_2Sin(\theta_1+\theta_2) + l_3Sin(\theta_1+\theta_2+\theta_3)$$

Adaptive Neuro-Fuzzy Inference System (ANFIS)

The ANFIS originated in the 1990s as a framework to tune Sugeno-type fuzzy logic systems based on a learned behavior from neural networks [12]. To train a robust FIS, understanding the effects of different training parameters (like the size and spread of dataset, number of membership functions, number of epochs, stepsize genfis1, genfis2, genfis3 etc.) is necessary.

Configuration space

A 3-DOF arm can reach a same point in the workspace (Figure 11) with a different configuration. This issue is also called the elbow up elbow down dilemma where two inverse kinematic solutions (joint space) are possible for a Cartesian target. This problem is hard to solve because the position and the orientation of the end effector are needed to be known for a significant past duration. To simplify the situation, only the positive angle ranges were considered., where θ_1, θ_2, θ_3 lie in the range of $[0,\pi/3][0,\pi/2][0,\pi]$ respectively. Note that, although the elbow down/up configuration are still possible at the same Cartesian target, the data set is computed with only thelbow up configuration.

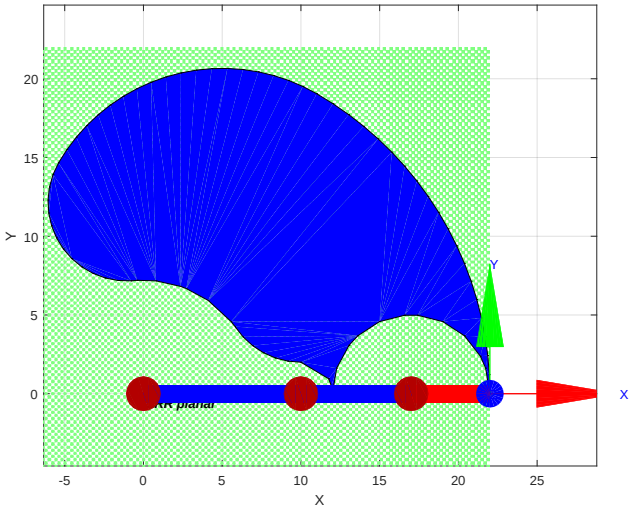

Figure 11 3RRR planar arm

Number of epochs

The number of epochs is really crucial in the process of training the anfis. Epoch is the time elapsed since a process is started, therefore the epochs which we set when initializing a training process with anfis, can directly impact the performance of the derived fuzzy inference system. Table 5 shows the joint space and Cartesian space Root Mean Square error observed for θ_1 and θ_2 as the number of epochs is increased for a dataset of 100 points and two membership functions. From Table 5, it can be seen that both RMSEs in Cartesian and joint space stops decreasing significantly after 500 epochs. Therefore, a maximum of 500 epochs was used.

Table 5 Dh standard paramenters for the three joints

Epochs	$\theta_1, \theta_3, \theta_3$ rmseTr.	xy rmseTr.	Time
50	0.0848	0.1474	8.9448
100	0.0358	0.0795	5.8968
500	0.0312	0.0682	5.5590
1000	0.0311	0.0681	5.5723

Number of Membership Functions

The number of membership functions in genfis1 is really important for accuracy. However, like a polynomial over-fitting on data, having too many membership functions can lead to over-fitting. In order to check for over-fitting, validation data is used. Then over-fitting can be detected as the simultaneous drop and rise in training error and validation error respectively Figure 12. Over-fitting can be avoided by increasing the size of the dataset.

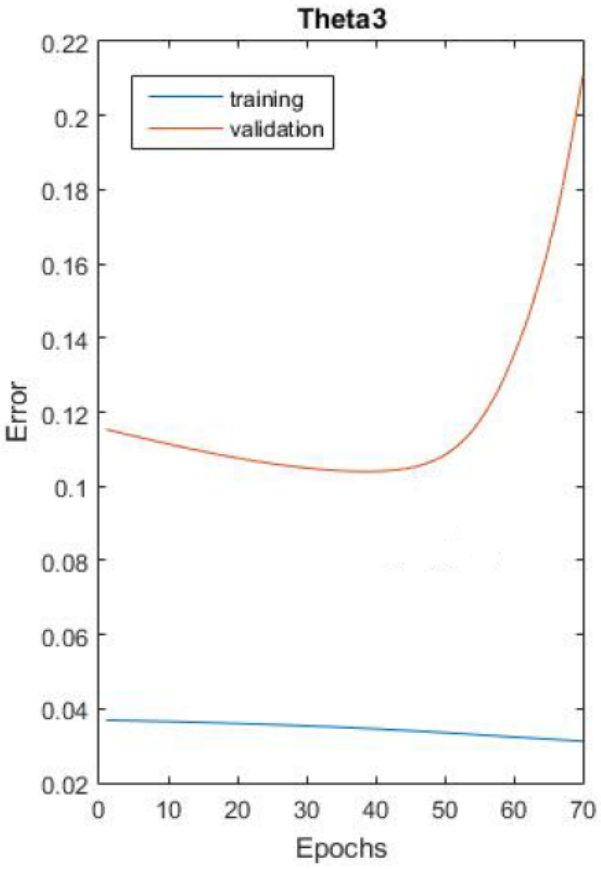

Figure 12 3RRR planar arm

Results

A Fuzzy inference system was tuned to produced accepted levels of error. Figure 11 shows the difference b/w the analytically deduced values and the predict on made by the fis. It can be seen that $\theta_3 fisisthelargestsourceoferror.$

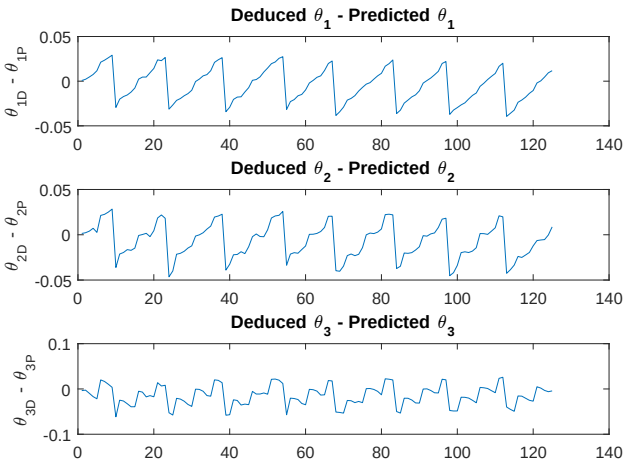

Figure 13 3RRR planar arm

Another way to visualize the performance is through a simulation. Thus, a simulation environment was developed. The simulation has the trained 3 of planar arm with the workspace shown in blue. The figure also shows an elliptical trajectory (red) which the robot has to follow and the actual trajectory based on the joint angles predicted by the FIS (yellow).

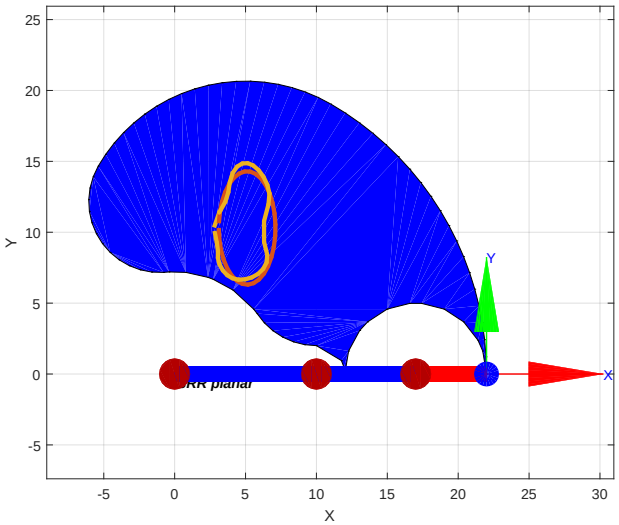

Figure 14 3RRR planar arm

Optinal task optimising Genfis2 and ANFIS with genetic algorithm

An encoding was developed for the following options in the genfis2 and anfis functions. Genfis2 generates Sugeno-type FIS structures using subtractive clustering. It requires radii vector to adjust

the dimensional influence of inputs and outputs. For every θ, the radii was a vector three values. In total nine such values are present in the chromosomes. Anfis has the option of step size and a number of epoch. Thus for every θ there are two such values. In total 6 values.

Thus the chromosome can be represented as

$[r1_{\theta 1}, r2_{\theta 1}, r3_{\theta 1}, r1_{\theta 2}, r2_{\theta 2}, r3_{\theta 2}, r1_{\theta 3}, r2_{\theta 3}, r3_{\theta 3}, Epoch_{\theta 1}, Epoch_{\theta 2}, Epoch_{\theta 3}, ss_{\theta 1}, ss_{\theta 2}, ss_{\theta 3}]$

A single point cross over was used (as shown in figure).

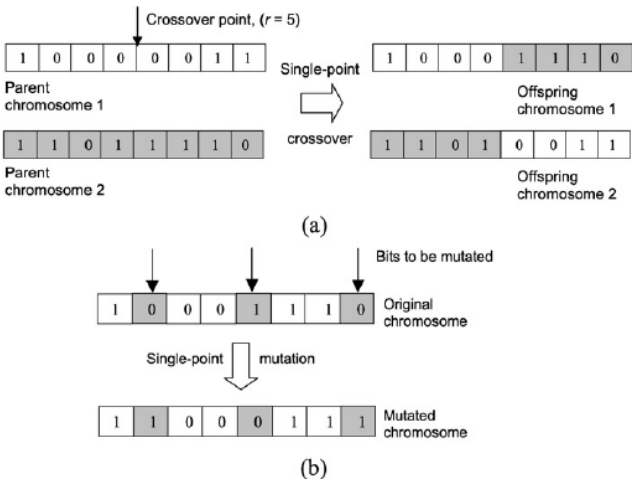

Figure 15 3RRR planar arm

The fitness is calculated as

$max(RMSE_{\theta 1}, RMSE_{\theta 2}, RMSE_{\theta 3}) + \frac{\sum_{i=0}^{n} \sum_{j=1}^{3} \theta_a j(i) - \theta_p j(i)}{n}$

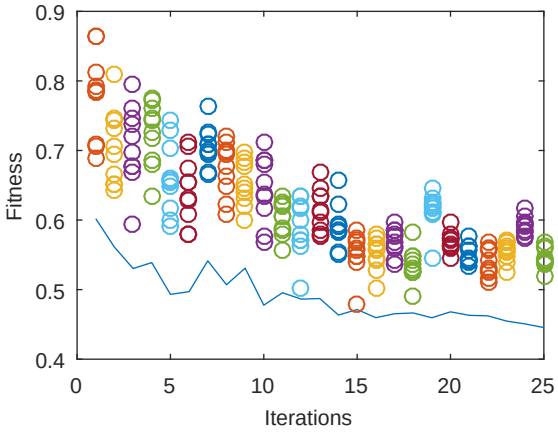

Figure 16 3RRR planar arm

References

[1] Y. Pan and K. Furuta, "Vss controller design for discrete-time systems," in *Industrial Electronics, Control, and Instrumentation, 1993. Proceedings of the IECON'93., International Conference on*, pp. 1950–1955, IEEE, 1993.

[2] M. W. Spong, S. Hutchinson, and M. Vidyasagar, *Robot modeling and control*, vol. 3. wiley New York, 2006.

[3] A. K. Yadav, P. Gaur, A. Mittal, and M. Anzar, "Comparative analysis of various control techniques for inverted pendulum," in *Power Electronics (IICPE), 2010 India International Conference on*, pp. 1–6, IEEE, 2011.

[4] S. K. Tripathi, H. Panday, and P. Gaur, "Robust control of inverted pendulum using fuzzy logic controller," in *Engineering and Systems (SCES), 2013 Students Conference on*, pp. 1–6, IEEE, 2013.

[5] K. Pathak, J. Franch, and S. K. Agrawal, "Velocity and position control of a wheeled inverted pendulum by partial feedback linearization," *IEEE Transactions on robotics*, vol. 21, no. 3, pp. 505–513, 2005.

[6] C. E. Lin and Y.-R. Sheu, "A hybrid-control approach for pendulum-car control," *IEEE Transactions on Industrial Electronics*, vol. 39, no. 3, pp. 208–214, 1992.

[7] T.-H. S. Li and M.-Y. Shieh, "Switching-type fuzzy sliding mode control of a cart–pole system," *Mechatronics*, vol. 10, no. 1, pp. 91–109, 2000.

[8] H.-B. Jun, D.-W. Lee, D.-J. Kim, and K.-B. Sim, "Fuzzy inference-based reinforcement learning of dynamic recurrent neural networks," in *SICE'97. Proceedings of the 36th SICE Annual Conference. International Session Papers*, pp. 1083–1088, IEEE, 1997.

[9] S.-G. Kwon, "Two fuzzy controllers alternating for cartpole system," *International Journal of Fuzzy Logic and Intelligent Systems*, vol. 9, no. 2, pp. 154–160, 2009.

[10] L.-C. Wang and C.-C. Chen, "A combined optimization method for solving the inverse kinematics problems of mechanical manipulators," *IEEE Transactions on Robotics and Automation*, vol. 7, no. 4, pp. 489–499, 1991.

[11] R. S. Hartenberg and J. Denavit, "A kinematic notation for lower pair mechanisms based on matrices," *Journal of applied mechanics*, vol. 77, no. 2, pp. 215–221, 1955.

[12] J.-S. Jang, "Anfis: adaptive-network-based fuzzy inference system," *IEEE transactions on systems, man, and cybernetics*, vol. 23, no. 3, pp. 665–685, 1993.

Appendices

Fuzzy inference .fis for the cart pole simulation

```
[System]
Name='inference'
Type='mamdani'
Version=2.0
NumInputs=4
NumOutputs=1
NumRules=12
AndMethod='prod'
OrMethod='max'
ImpMethod='min'
AggMethod='sum'
DefuzzMethod='centroid'

[Input1]
Name='angle'
Range=[-1 1]
NumMFs=3
MF1='left':'trimf',[-2 -1 0.6]
MF2='right':'trimf',[-0.6 1 2]
MF3='stable':'trimf',[-1 0 1]

[Input2]
Name='angle-velocity'
Range=[-5 5]
NumMFs=3
MF1='left':'trimf',[-10 -5 2]
MF2='right':'trimf',[-2 5 10]
MF3='stable':'trimf',[-5 0 5]

[Input3]
Name='cart-position'
Range=[-10 10]
NumMFs=3
MF1='left':'trimf',[-20 -10 4]
MF2='right':'trimf',[-4 10 20]
MF3='stable':'trimf',[-10 -1.11e-16 10]

[Input4]
Name='cart-velocity'
Range=[-10 10]
NumMFs=3
MF1='left':'trimf',[-20 -10 2]
MF2='right':'trimf',[-2 10 20]
MF3='stable':'trimf',[-10 -1.11e-16 10]

[Output1]
Name='F'
Range=[-200 200]
NumMFs=7
MF1='N':'trimf',[-100 -50 0]
MF2='Z':'trimf',[-50 0 50]
MF3='P':'trimf',[0 50 100]
MF4='NL':'trapmf',[-400 -400 -100 50]
MF5='NM':'trimf',[-150 -100 -50]
MF6='PL':'trapmf',[-50 100 400 400]
MF7='PM':'trimf',[50 100 150]

[Rules]
1 1 0 0, 4 (1) : 1
1 3 0 0, 5 (1) : 1
1 2 0 0, 1 (1) : 1
3 1 0 0, 1 (1) : 1
3 3 0 0, 2 (1) : 1
```

```
3 2 0 0, 3 (1) : 1
2 1 0 0, 3 (1) : 1
2 3 0 0, 7 (1) : 1
2 2 0 0, 6 (1) : 1
0 0 1 1, 4 (1) : 1
0 0 2 2, 6 (1) : 1
0 0 3 3, 2 (1) : 1
```

Training 3DOF robots using anfis and genfis2 (Requires robotic toolbox)

```
%Requires Robotic toolbox
clear
L1 = Link('d', 0, 'a', 10, 'alpha', 0);
L2 = Link('d', 0, 'a', 7, 'alpha', 0);
L3 = Link('d', 0, 'a', 5, 'alpha', 0);
bot = SerialLink([L1 L2 L3], 'name', '3RR planar')

%%  Forward kine
qtr=[]; % Matix to hold angle values
n=5; %workspace resolution
range1=linspace(0,pi/3,n);
range2=linspace(0,pi/2,n);
range3=linspace(0,pi,n);

%theta1 goes from 0 to 90
for i=1:n
    %theta2 goes from 0 to 90
    for j=1:n
        %theta3 goes from 0 to 90
        for k=1:n
            qtr=[qtr;range1(i) range2(j) range3(k)];
        end
    end
end

position=bot.fkine(qtr); %caculate forward homogenous matrix

position=squeeze(position([1 2 3],4,:))';

f = figure;
ax = axes;
workspace= dunley_workspace(0, position );
fill(workspace(:,1),workspace(:,2),'b');
hold on;
bot.plot([0,0,0]);
hold off;

%% Stratified selection of training and validation dataset
% %First we assign a random weight to every datum.
% wieghts=randperm(size(position,1),size(position,1)); %generate random numbers
% %Lets resample
% I=resampleStratified(wieghts);
%
% position_valid=position(I(1:size(I,2)/2),:)
%% data
ratio=0.5;
%divide_train=1:round(size(position,1)*ratio);
divide_train=1:size(position,1);
divide_valid=round(size(position,1)*ratio):size(position,1);
%Training
data1 = [position(divide_train,1) position(divide_train,2) qtr(divide_train,1)]; %
    create x-y-theta1 dataset
data2 = [position(divide_train,1) position(divide_train,2) qtr(divide_train,2)]; %
    create x-y-theta2 dataset
data3 = [position(divide_train,1) position(divide_train,2) qtr(divide_train,3)]; %
    create x-y-theta3 dataset

```

```
%Validation
data4 = [position(divide_valid,1) position(divide_valid,2) qtr(divide_valid,1)]; %
    create x-y-theta1 dataset
data5 = [position(divide_valid,1) position(divide_valid,2) qtr(divide_valid,2)]; %
    create x-y-theta2 dataset
data6 = [position(divide_valid,1) position(divide_valid,2) qtr(divide_valid,3)]; %
    create x-y-theta3 dataset
%% Train
genmat1=genfis2(data1(:,[1,2]),data1(:,3),[0.7522    0.5246    0.5807]);
genmat2=genfis2(data2(:,[1,2]),data2(:,3),[ 0.6826    0.5383    0.9066]);
genmat3=genfis2(data3(:,[1,2]),data3(:,3),[0.5169    0.5356    0.5024]);

disp('started');
fismat1=anfis([data1(:,[1,2]),data1(:,3)],genmat1,[500 0 0.6483]);
fismat2=anfis([data2(:,[1,2]),data2(:,3)],genmat2,[500 0 0.6594]);
fismat3=anfis([data3(:,[1,2]),data3(:,3)],genmat3,[500 0 0.7121]);

save('fismat1.mat','fismat1');
save('fismat2.mat','fismat2');
save('fismat3.mat','fismat3');
%% GUI
anfis1=load('fismat1.mat');
anfis1=anfis1.fismat1;
anfis2=load('fismat2.mat');
anfis2=anfis2.fismat2;
anfis3=load('fismat3.mat');
anfis3=anfis3.fismat3;

XY=[5.112010725586194,10.302837652737612]; %target5

[x,y] = elipseo(2,4,XY(1),XY(2)); % Get an eliptical profile


actual_position=[]; %To recod the actual position of the robot
% qtrio=[evalfis(XY(:,[1,2]), anfis1) evalfis(XY(:,[1,2]), anfis2) evalfis(XY(:,[1,2]),
    anfis3)] %anfis prediction
for i=1:size(x,2)
    xbit=x(i);
    ybit=y(i);
%   position=bot.fkine(qtr); %caculate forward homogenous matrix
    a_p=bot.animate([evalfis([xbit,ybit], anfis1) evalfis([xbit,ybit], anfis2) evalfis
    ([xbit,ybit], anfis3)]);
    actual_position=[actual_position; a_p(1) a_p(2)];
    %    qtrio=[qtrio; evalfis([xbit,ybit], anfis1) evalfis([xbit,ybit], anfis2)
    evalfis([xbit,ybit], anfis3)]; %anfis prediction
end

hold on;
theta = 0:4/5*pi:4*pi;
plot(cos(theta),sin(theta),'y-')
plot(x,y,'LineWidth',3)
hold on;
plot(actual_position(:,1),actual_position(:,2),'LineWidth',3);
bot.plot([0,0,0]);

hold off;

%% plots
figure(3)
position2=bot.fkine(qtr);
IK=bot.ikine(position2,[0 0 0],[0 0 0 0 0 1]);
theta1ik=IK(:,1);
theta2ik=IK(:,2);
theta3ik=IK(:,3);
theta1fz=evalfis([position(:,1),position(:,2)], anfis1);
theta2fz=evalfis([position(:,1),position(:,2)], anfis2);
```

```
theta3fz=evalfis([position(:,1),position(:,2)], anfis3);

subplot(3,1,1);
plot((theta1ik-theta1fz)*10^-2);
ylabel('\theta_{1D} - \theta_{1P}');
title('Deduced \theta_1 - Predicted \theta_1');

subplot(3,1,2);
plot((theta2ik-theta2fz)*10^-2);
ylabel('\theta_{2D} - \theta_{2P}');
title('Deduced \theta_2 - Predicted \theta_2');

subplot(3,1,3);
plot((theta3ik-theta3fz)*10^-2);
ylabel('\theta_{3D} - \theta_{3P}');
title('Deduced \theta_{3} - Predicted \theta_{3}');
```

Optimising genfis2 and ANFIS inputs using genetic algorithm

```
clear
L1 = Link('d', 0, 'a', 10, 'alpha', 0);
L2 = Link('d', 0, 'a', 7, 'alpha', 0);
L3 = Link('d', 0, 'a', 5, 'alpha', 0);
bot = SerialLink([L1 L2 L3], 'name', '3RR planar')

%%  Forward kine
qtr=[]; % Matix to hold angle values
n=20; %workspace resolution
range1=linspace(0,pi/3,n);
range2=linspace(0,pi/2,n);
range3=linspace(0,pi,n);

%theta1 goes from 0 to 90
for i=1:n
    %theta2 goes from 0 to 90
    for j=1:n
        %theta3 goes from 0 to 90
        for k=1:n
            qtr=[qtr;range1(i) range2(j) range3(k)];
        end
    end
end

position=bot.fkine(qtr); %caculate forward homogenous matrix

position=squeeze(position([1 2 3],4,:))';

position2=bot.fkine(qtr);
IK=bot.ikine(position2,[0 0 0],[0 0 0 0 0 1]);


%% data
ratio=0.5;
%divide_train=1:round(size(position,1)*ratio);
divide_train=1:size(position,1);
divide_valid=round(size(position,1)*ratio):size(position,1);
%Training
data1 = [position(divide_train,1) position(divide_train,2) qtr(divide_train,1)]; %
    create x-y-theta1 dataset
data2 = [position(divide_train,1) position(divide_train,2) qtr(divide_train,2)]; %
    create x-y-theta2 dataset
data3 = [position(divide_train,1) position(divide_train,2) qtr(divide_train,3)]; %
    create x-y-theta3 dataset

%Validation
data4 = [position(divide_valid,1) position(divide_valid,2) qtr(divide_valid,1)]; %
    create x-y-theta1 dataset
```

```
data5 = [position(divide_valid,1) position(divide_valid,2) qtr(divide_valid,2)]; %
    create x-y-theta2 dataset
data6 = [position(divide_valid,1) position(divide_valid,2) qtr(divide_valid,3)]; %
    create x-y-theta3 dataset
%% Train

%First thing needed is the radii vector. i.e. a vector b/w 0 and 1.
%chromosome first nine values are the radii for input and outputs.
%chromosome=[onel,onem,oner,twol,twom,twor,threel,threem,threer];
%next it also has training parameters for anfis. The first number
%is number of epochs and the last argument is stepsize.
chromosomes=[];
%initial_population

for i=1:5
    %generate three numbers b/w 0 and 1 for radius
    r1 = (1-0.4).*rand(9,1)' + 0.4;
    %generate three numbers b/w 300 and 900 for epochs
    r2 = round((550-300).*rand(3,1)' + 300);
    %%generate three numbers b/w 0 and 1 for stepsize
    r3 = (1-0.5).*rand(3,1)' + 0.5;
    %%Chromosome
    chrome=[r1 r2 r3];
    chromosomes=[chromosomes;chrome];
end
bests=[];

for i=1:10
%evaluate population
ereo=[];
    for j=1:size(chromosomes,1)
        error = evaluate_GA(chromosomes(j,:),data1,data2,data3,data4,data5,data6,IK,
    position);
        ereo=[ereo;error];
    end

    %develop top four parents
    [~,I] = sort(ereo,'ascend'); %Sort in ascending order.
    %The first element is the best one so far
    chromea=chromosomes(I(1:4),:);%retreive the best four cromosomes.
    bests=[bests; chromea(1,:) ereo(I(1))]; %Store the best

    ereo(I(1))

    %Generate five new chromosomes
    [newChromosome1, newChromosome2]=PerformCrossover(chromea(1,:), chromea(2,:));
    [newChromosome3, newChromosome4]=PerformCrossover(chromea(3,:), chromea(4,:));
    newChromosome5=chromosomes(I(5),:);
    chromosomes=[newChromosome1;newChromosome2;newChromosome3;newChromosome4;
    newChromosome5]
end
```

Crossover operator

```
function [newChromosome1, newChromosome2] = PerformCrossover( c1, c2 )
pos1=randi(length(c1)-1);
newChromosome1 = [ c1(1:pos1)  c2(pos1+1:length(c1)) ];
newChromosome2 = [ c2(1:pos1)  c1(pos1+1:length(c1)) ];
end
```

Fitness

```
function [error] = evaluate_GA(chrome,data1,data2,data3,data4,data5,data6,IK,position)
genmat1=genfis2(data1(:,[1,2]),data1(:,3),chrome(:,[1,2,3]));
genmat2=genfis2(data2(:,[1,2]),data2(:,3),chrome(:,[4,5,6]));
genmat3=genfis2(data3(:,[1,2]),data3(:,3),chrome(:,[7,8,9]));

```

```
disp('started');
fismat1=anfis([data1(:,[1,2]),data1(:,3)],genmat1,[chrome(10) 0 chrome(13)]);
fismat2=anfis([data2(:,[1,2]),data2(:,3)],genmat2,[chrome(11) 0 chrome(14)]);
fismat3=anfis([data3(:,[1,2]),data3(:,3)],genmat3,[chrome(12) 0 chrome(15)]);

save('fismat1.mat','fismat1');
save('fismat2.mat','fismat2');
save('fismat3.mat','fismat3');

disp('ended');

%% Error
fuzout=evalfis(data1(:,[1,2]),fismat1);
trnRMSE=norm(fuzout-data1(:,3))/sqrt(length(fuzout));
chkfuzout=evalfis(data4(:,[1,2]),fismat1);
chkRMSE1=norm(chkfuzout-data4(:,3))/sqrt(length(chkfuzout));

fuzout=evalfis(data2(:,[1,2]),fismat2);
trnRMSE=norm(fuzout-data2(:,3))/sqrt(length(fuzout));
chkfuzout=evalfis(data5(:,[1,2]),fismat2);
chkRMSE2=norm(chkfuzout-data5(:,3))/sqrt(length(chkfuzout));

fuzout=evalfis(data3(:,[1,2]),fismat3);
trnRMSE=norm(fuzout-data3(:,3))/sqrt(length(fuzout));
chkfuzout=evalfis(data6(:,[1,2]),fismat3);
chkRMSE3=norm(chkfuzout-data6(:,3))/sqrt(length(chkfuzout));

chkRMSE=max(abs([chkRMSE1,chkRMSE2,chkRMSE3]));
%% Joint error
theta1ik=IK(:,1);
theta2ik=IK(:,2);
theta3ik=IK(:,3);
theta1fz=evalfis([position(:,1),position(:,2)], fismat1);
theta2fz=evalfis([position(:,1),position(:,2)], fismat2);
theta3fz=evalfis([position(:,1),position(:,2)], fismat3);
joint_error=max([abs(mean(theta1ik-theta1fz)),abs(mean(theta2ik-theta2fz)),abs(mean(
    theta3ik-theta3fz))])
error=chkRMSE+joint_error;
end
```