A
Project
on

PSO optimized Neuro-Fuzzy controller

Submitted in Partial Fulfillment of the course
BITS F312, Neural Networks and Fuzzy Logic

By

2014A7PS099P	Patwa Smit Manish
2014A7PS096P	Pranav Misra
2014A7PS033P	Chirag Agarwal
2013B2A3838P	Harsh Sinha
2013B1A8817P	Tarun Kumar Vangani

Birla Institute of Technology and Science, Pilani - Pilani Campus

Contents

1. Introduction
2. Literature Review
3. Implementation
    1. DC Motor Modelling
    2. ANFIS - Adaptive Neuro-Fuzzy Inference System
    3. PSO - Particle Swarm Optimization
4. Results

Introduction

DC Motors are being used everywhere - from large industries to sophisticated robot manipulators. It is very important to have correct parameters governing the DC Motor such as speed, torque etc. The right speed of the motor is fundamental to achieve good performance of a dc motor.

In this project, a neuro-fuzzy inference system is used to control the speed of the DC Motor. The various parameters governing the neural architecture are optimized using particle swarm optimization.

In order to achieve higher precision, for accurate application such as robot manipulators or translating robots, we needed numerical knowledge and linguistic inference. As we know, neuro fuzzy systems combine these main features of artificial neural networks and fuzzy logic systems. To imitate the nonlinear behavior of DC motors, we used ANFIS (Adaptive Neuro Fuzzy Inference System) which has proved itself in wide range of applications over the past years.

Research Papers

1. DC Motor neuro-fuzzy controller using PSO identification

DC motors acts nonlinearly especially when they are subjected to load variance. Conventional controllers, such as PID, have reasonable performance in control of DC motors, but sometimes much more precision is needed in some accurate applications. Neuro-fuzzy systems represent a newly developed class of hybrid intelligent systems. This paper uses PSO to train the neuro-fuzzy model. The neuro-fuzzy model controls the speed of the motor according to the set speed. The embedded controllers have a limited capacity of computations and more computations are equal to more expensive controllers. Hence by optimizing the parameter of ANFIS offline using PSO such embedded controllers can be used. This paper is further discussed in this report in detail.

2. Trajectory tracking for a 3-DOF robot manipulator based on PSO and adaptive neuro-fuzzy inference system

Robot Manipulators are used for varied amount of different tasks, ranging from surgeries to industrial manufacturing and deep sea explorations. But for each task the thing that is inevitable is tracking the trajectory of the manipulator with precision.

Generally, trajectory of robot manipulators are designed using sliding mode controllers and inverse kinematics. Inverse Kinematics uses basic kinematic equations to evaluate discrete joint angles given motion of end effector. The paper under study proposed to evaluate the discrete joint angles of a 3-DOF (degrees of freedom) sliding mode controller using ANFIS optimized by PSO.The inputs to ANFIS are position of end effector (X, Y). The relationship between joint angles [q1, q2, q3]Tand (X, Y) is :

$$X = -L_1 sin(q1) - L_2 sin(q1 + q2) - L_3 sin(q1 + q2 + q3)$$

$$Y = -L_1 cos(q1) - L_2 cos(q1 + q2) - L_3 cos(q1 + q2 + q3)$$

The values of (X, Y) are discretized. so that they can be optimized using PSO. The discretized values are then put into ANFIS structure consisting of layers, namely - input, fuzzification, rule layer, normalization, defuzzification and output layer.Parameters for q1, q2, q3 were calculated separately by different ANFIS units.The membership functions were parameterized using a bell curve.

$$\mu = \frac{1}{1 + [(\frac{x-c}{b})^2]d}$$

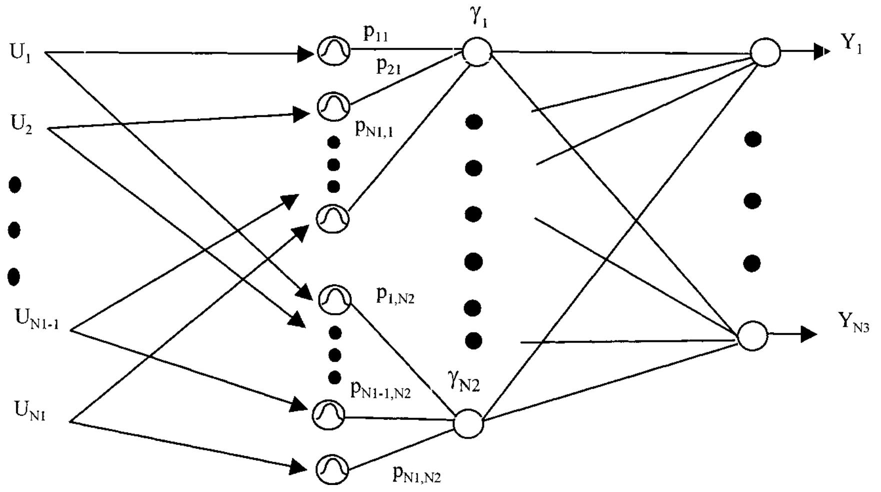
3. Sugeno type neuro-fuzzy system for modelling robot manipulators

This paper deals with the use of PSO to train an adaptive neuro-fuzzy inference system that efficiently changes the torque applied at N joints of a robotic element. By doing this, the paper attempts to make the N torques of the robotic arm equal to the unknown final torques corresponding to a target angular velocities or arm configuration or any other function F(τ). This paper was tested on a robotic element having three two or three links. The particle swarm optimisation trained neuro-fuzzy system performed well for these tests and converged to the required values within 20 iterations of the sampling and changing torque.

4. Obstacle Avoidance of mobile robot using PSO based Neuro Fuzzy Technique

This paper deals with the topic of Obstacle Avoidance of Robots. Path Planning is one of the key components in Autonomous Navigation of Robots. Existing approach to path planning including obstacle avoidance include Graph Algorithms like Visibility Graphs and and analytical methods. These methods suffer from two major problems - (1) The algorithms make it difficult to account for dynamic obstacles. (2) These cannot be used for online planning due to very high computational complexity of the algorithms.

The author proposes using ANFIS for overcoming these issues. He uses a modified version of the traditional Radial basis function network by adding an additional hidden layer which applies the min fuzzy operator to calculate degree of membership for different rules. The input to the ANFIS is sensor input which provides distance of obstacles. The output acts as signals for the actuators which move the robots. This calculation is made several times on a trained network during actual motion of the robot. The training is done from existing trajectories made by humans.



The image above represents the structure of neural network. The first layer consists of radial basis neurons and models the membership function. The output layer consists of linear neurons. The number of output neurons is decided by actuator signals. Number of neurons in hidden layer is determined by the rules.

The cost function E is optimised using the PSO algorithm exactly like the one described in the later section of this report. The parameters optimised include the weights between the hidden layer and the means and variance of the gaussian membership function.

$$E = \frac{1}{2}\Sigma(Y_k - y_k)$$

Y represents each output node.

The parameters used for training are C1 and C2 as 2 and Maximum Particle Velocity as 5.

Learning outcomes from this paper included understanding of PSO and how the basic setup for training is for a controller given input signals and actuators. This problem being very similar to the implementation we have done for DC Motor served as a helpful exercise for the project.

5. Particle swarm optimization based fuzzy-neural like PID controller for TCP/AQM router

This paper deals with congestion control in a network. Network scheduler uses AQM(Active Queue Management) to handle such congestions. A Fuzzy Proportional Integral genetic controller has been adopted as AQM for routers. This paper attempts to better the performance of AQM by using a PID neuro fuzzy controller. The parameters we are monitoring here are TCP window size, queue length, round trip time and these are dependent on link capacity(packets/second), load factor(number of TCP sessions) and probability of packet drop. This paper used PSO as a suitable optimization method for tuning the parameters of the Gaussian function and got much better performance than the FPI controller.

In this report we have implemented and analysed the first paper “**DC Motor neuro-fuzzy controller using PSO identification”.**

DC Motor Modelling

The electrical and mechanical equations representing a DC motor is given as,

$$v_a = R_a i_a + L_a(di/dt) + e_a \tag{1}$$

$$J_m(dw/dt) = \tau_m - B_m \omega_m - \tau_s \tag{2}$$

$$e_a = K_e w_m \tag{3}$$

$$\tau_m = K_m i_a \tag{4}$$

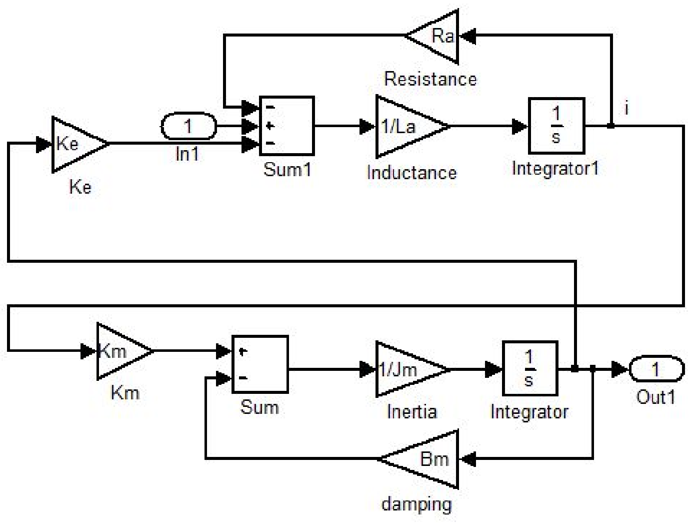
The various variables are

Va,	Motor Armature Voltage
Ra,	Armature Coil Resistance
Ia,	Armature Current
Ea,	Back EMF Voltage
w_m,	Angular Motor Speed
m,	Motor Torque
B_m,	Coefficient of Viscous Motor Friction
s,	Transmitted Shaft Torque

Ke and Km are constants

The specification of the DC motors dynamic model is as follows:

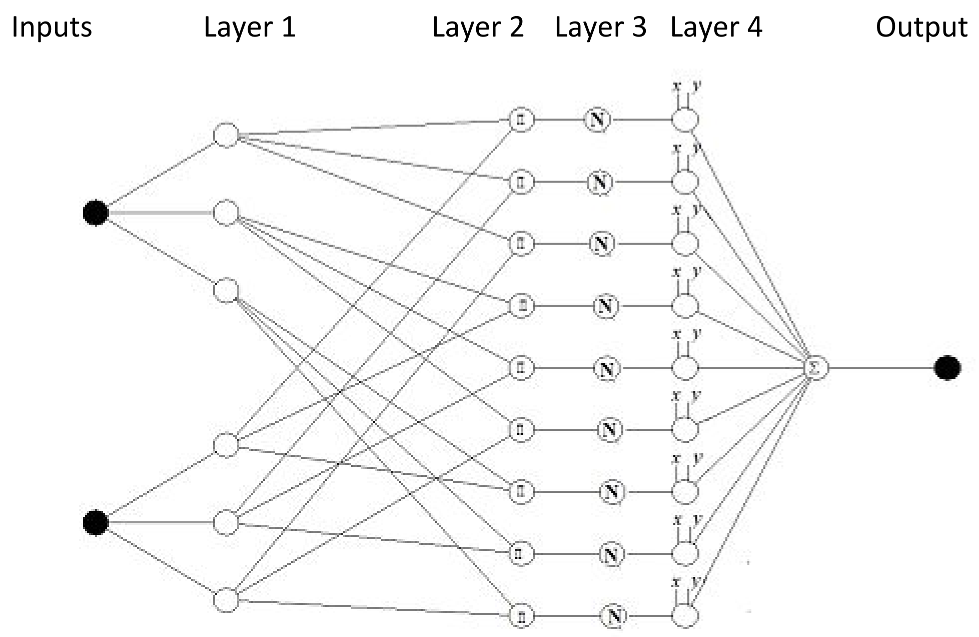
Jm = 0.068kg−m2
Bm = 0.03475N−m/(rad/s) Ra =7.56Ω
La = 0.055H
µ = 0.0039 Nm/(rad/s)2
Ke = 3.474 Nm amp
Km=3.475 V



Simulink model of the motor

ANFIS - Adaptive Neuro Fuzzy Inference System

ANFIS is based on Takagi -Sugeno-Kang fuzzy inference system. In ANFIS, neural network is used to learn and adapt the parameters (the membership function shapes) of a fuzzy logic system.



ANFIS structure

A typical Sugeno model has the form : If *Input 1* is x, *Input 2* is y, then *Output* z = px + qy + r.
For a zero order Sugeno model , output z is a constant . Hence, p = q = 0.
Let the *Input A* be Error in Speed and *Input B* be Change in Error. Layer 1 represents the membership grade in fuzzy set Input A (or Input B).

$$O_{1,i} = \mu_{A,i}(x) \quad for\ i = 1,\ 2,\ 3$$

$$O_{1,j} = \mu_{B,i}(y) \quad for\ j = 1,\ 2,\ 3$$

We used a gaussian function to parametrize the membership functions.
In Layer 2, weights define firing strengths of the defined rules. These weights (w_i) then define the output level (z_i).

$$O_{2,i} = w_i = \mu_x \,.\, \mu_y$$

where μ_x, μ_y represent the membership functions for Input 1 and Input 2.

In Layer 3, every node calculates the ratio of i-th weight to sum of all weights. Hence, weights are now normalized.

$$O_{3,i} = \frac{w_i}{\Sigma w_i}$$

In Layer 4, the normalized weights are then multiplied to $z = p_i x + q_i y + r_i$
The final output is decided by weighted average of all rule outputs (N).

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

PSO - Particle Swarm Optimization

Eberhart and Kennedy introduced particle swarm optimization (PSO). It is an efficient fast convergence algorithm, widely used in recent years. PSO imitates the movement of birds flocking or fish schooling looking for food. In this optimization technique, each particle has two parameters: position and velocity. The particle adjusts the velocity and position according to the best experiences that are called the Pbest found by itself and Gbest found by all its neighbors.

The pseudocode for PSO is given below:

```
for each particle $i$ = 1, ..., $S$ do
  Initialize the particle's position with a uniformly distributed random vector: $\mathbf{x}_i \sim U(\mathbf{b}_{lo}, \mathbf{b}_{up})$
  Initialize the particle's best known position to its initial position: $\mathbf{p}_i \leftarrow \mathbf{x}_i$
  if $cost(\mathbf{p}_i) < cost(\mathbf{g})$ then
     update the swarm's best known  position: $\mathbf{g} \leftarrow \mathbf{p}_i$
while iterations are left do:
  for each particle $i$ = 1, ..., $S$ do
      Update the particle's velocity: $\mathbf{v}_{i,d} \leftarrow \omega\ \mathbf{v}_{i,d} + c1 * r_p (\mathbf{p}_{i,d} - \mathbf{x}_{i,d}) + c2 * r_g (\mathbf{g}_d - \mathbf{x}_{i,d})$
      Update the particle's position: $\mathbf{x}_i \leftarrow \mathbf{x}_i + \mathbf{v}_i$
      if $cost(\mathbf{x}_i) < cost(\mathbf{p}_i)$ then
        Update the particle's best known position: $\mathbf{p}_i \leftarrow \mathbf{x}_i$
        if $cost(\mathbf{p}_i) < cost(\mathbf{g})$ then
          Update the swarm's best known position: $\mathbf{g} \leftarrow \mathbf{p}_i$
```

Our particle swarm structure is:

μ1	μ2	μ3	μ4	μ5	μ6	σ1	σ2	σ3	σ4	σ5	σ6	r1	r2	r3	r4	r5	r6	r7	r8	r9

Where $(\mu 1, \sigma 1)$, $(\mu 2, \sigma 2)$, $(\mu 3, \sigma 3)$ are mean and variance of gaussian distribution membership functions for Error and $(\mu 4, \sigma 4)$, $(\mu 5, \sigma 5)$, $(\mu 6, \sigma 6)$ are mean and variance of gaussian distribution membership functions for Change in error.

And r1,r2,r3,r4,r5,r6,r7,r8,r9 are consequent parameters of ANFIS.

The Cost for each particle is calculated as:

```
for each train set do:
    x = find the output of ANFIS using the particle parameters.
    if the sign of x is wrong:
        add 2000 to cost
    if value of x is too large for given error:
        add 1000 to cost
```

(Note: trainset contains the error of speed and change in error values)

The parameter used in our PSO are:

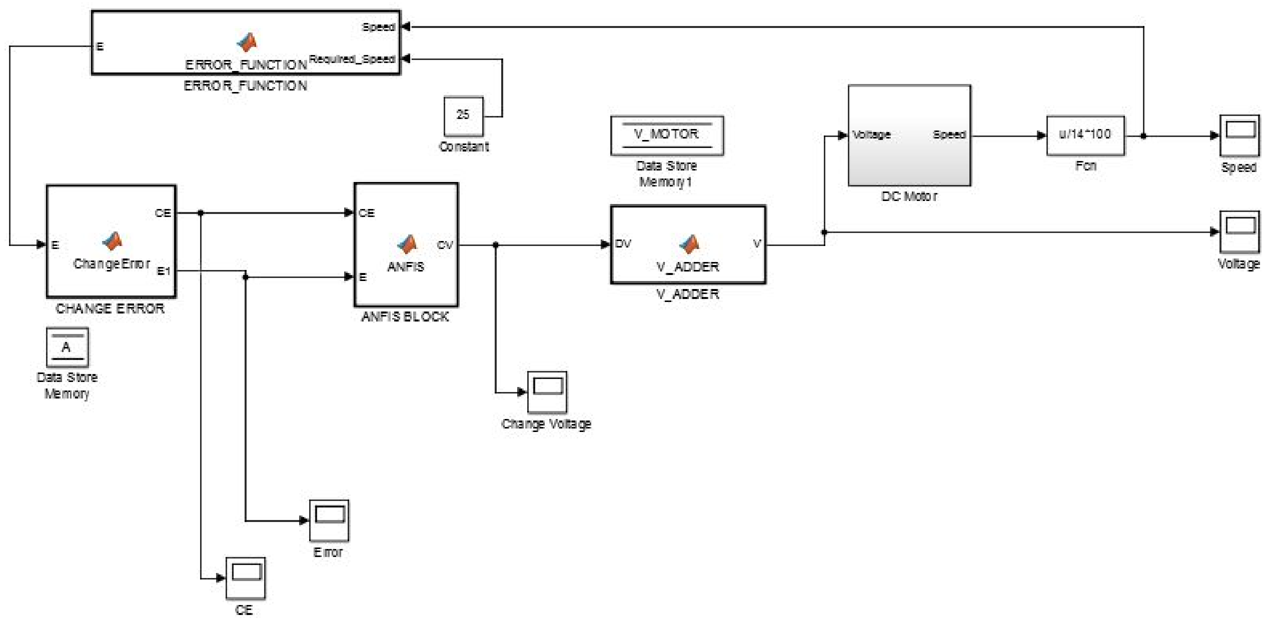
```
iterations=1000;
population_size=20;
w=0.7298;
c1=1.4962;
c2=1.4962;
particle_size=21;
min_limit = [-20 -20 -20 -.5 -.5 -.5   5  5  5 1 1 1 -3 -3 -3 -3 -3 -3 -3 -3 -3];
max_limit = [ 20  20  20  .5  .5  .5  30 30 30 4 4 4  3  3  3  3  3  3  3  3  3];
```

Where,

iterations	-	number of epochs of PSO
population_size	-	number of particles
w, c1, c2	-	learning factors of PSO
particle_size	-	number of parameters to optimize
max_limit, min_limit	-	vector of size 21 signifying domain in which particles should search

The final best particle returned by PSO is then used in ANFIS for simulations.

Simulink

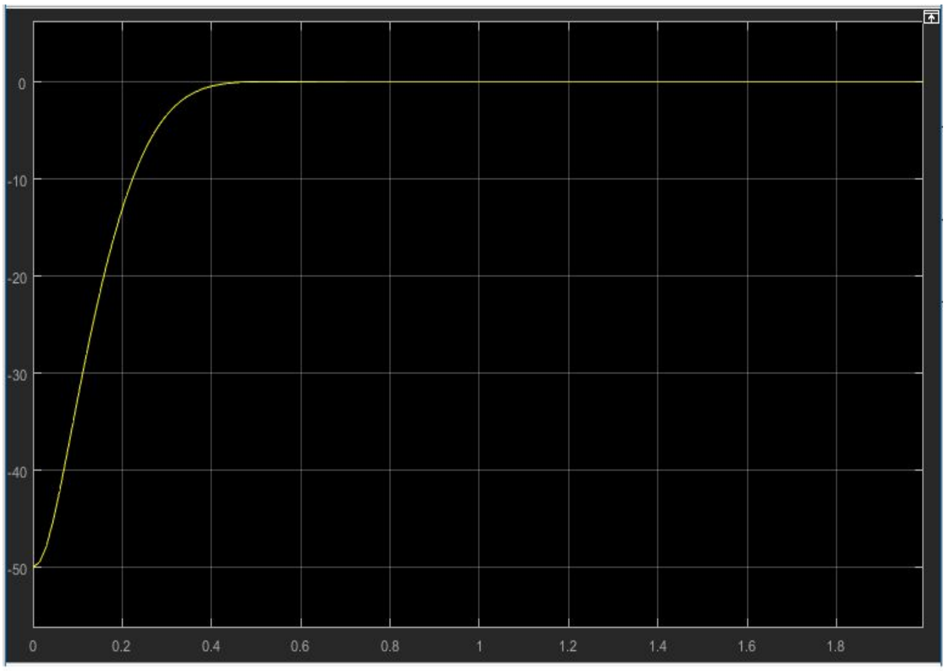


In simulink the operations perform by different units

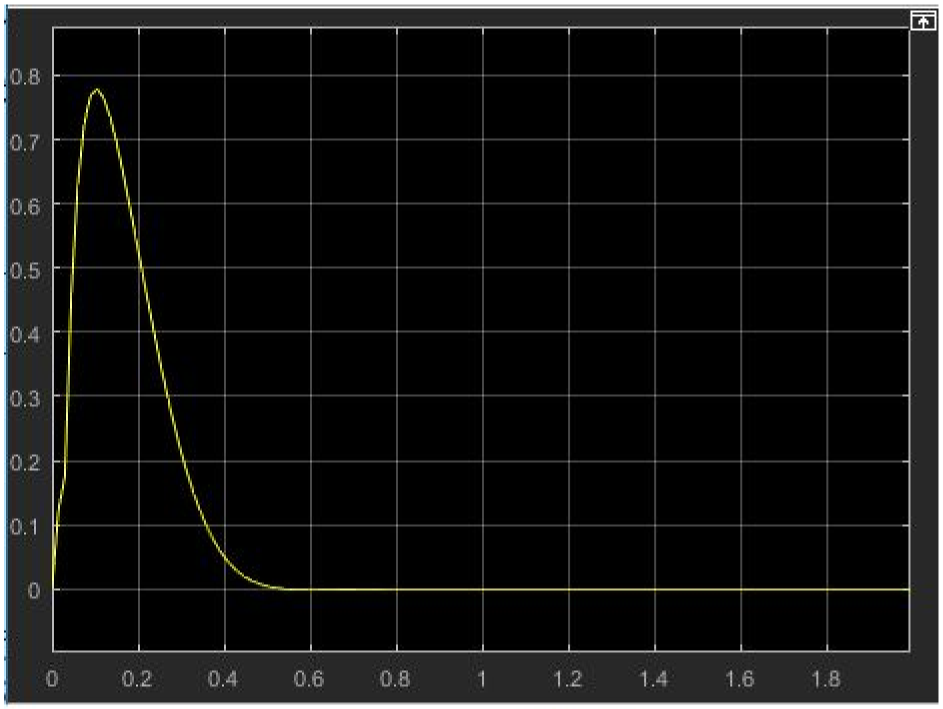
ERROR_FUNCTION	-	Calculates current error in speed from speed and required speed
ChangeError	-	Outputs Error and Change in Error required by ANFIS
ANFIS	-	Contains ANFIS code
V_ADDER	-	Adds the value given by ANFIS to the current voltage

The solver type user in MATLAB Simulink is fixed type. The sampling time is setted to 0.015s (i.e. 15ms).

Graphs



Error VS Time Graph



Change in Error VS Time

References

1. Farid, Ali Moltajaei, and S. Masoud Barakati. "DC Motor neuro-fuzzy controller using PSO identification." *2014 22nd Iranian Conference on Electrical Engineering (ICEE)*. IEEE, 2014.
2. Zeng, Ke, et al. "Trajectory tracking for a 3-DOF robot manipulator based on PSO and adaptive neuro-fuzzy inference system." *Control Conference (CCC), 2016 35th Chinese*. TCCT, 2016.
3. Dutta, Sourav. "Obstacle Avoidance of Mobile Robot using PSO-based Neuro Fuzzy Technique." *International Journal of Computer Science and Engineering* 2.2 (2010): 301-304.
4. Chatterjee, Amitava, and Keigo Watanabe. "An optimized Takagi-Sugeno type neuro-fuzzy system for modeling robot manipulators." *Neural Computing & Applications* 15.1 (2006): 55-61.
5. Mohammed Z, Al-Faiz, and Sadeq Shahad A. "Particle swarm optimization based fuzzy-neural like PID controller for TCP/AQM router." *Intelligent Control and Automation* 2012 (2012).