Faculty of Engineering

Credit Hours Engineering Programs

Mechatronics Engineering and Automation

Academic Year 2019/2020 – Spring 2019

CSE 488

Computational Intelligence

Project No. (3)

Fuzzy Control Genetic Algorithm, and PSO

Date due 31-Mar-19
Date handed in 31-Mar-19

Submitted by:

---------- ----------
---------- ----------
---------- ----------
---------- --------

Contents

Problem Definition and Importance

The aim of this project is simulating a fuzzy control of a vehicle model and to optimize the response of the system using different methods such as minimum of Integral of Square Error (ISE), genetic optimization, and swarm optimization (Non-derivative optimization methods).

Methods and Algorithms

1. Fuzzy Control

1.1 Introduction

Fuzzy Logic resembles the human decision-making methodology. It deals with vague and imprecise information. This is gross oversimplification of the real-world problems and based on degrees of truth rather than usual true/false or 1/0 like Boolean logic.

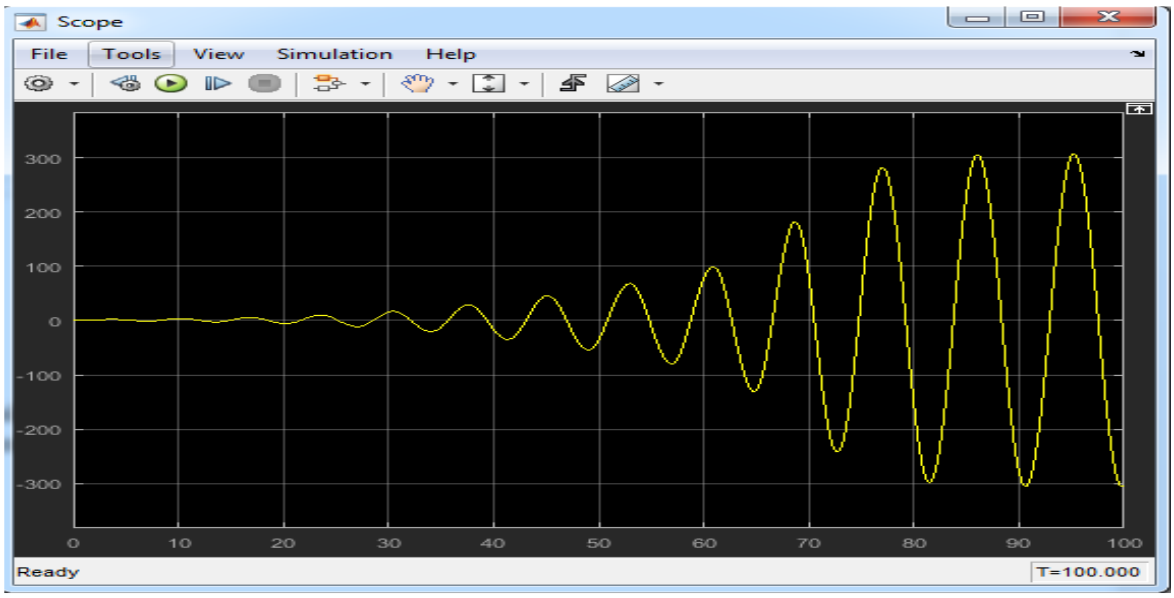
Fuzzy Logic was introduced in 1965 by Lofti A. Zadeh in his research paper “Fuzzy Sets”.

The process is composed of 3 main steps:
Fuzzification: Representing any input value you have with a membership function. This function defines how each point in the input space is mapped to a membership value (or degree of membership) between 0 and 1.
Execute all applicable rules in the rule-base to compute the fuzzy output functions.

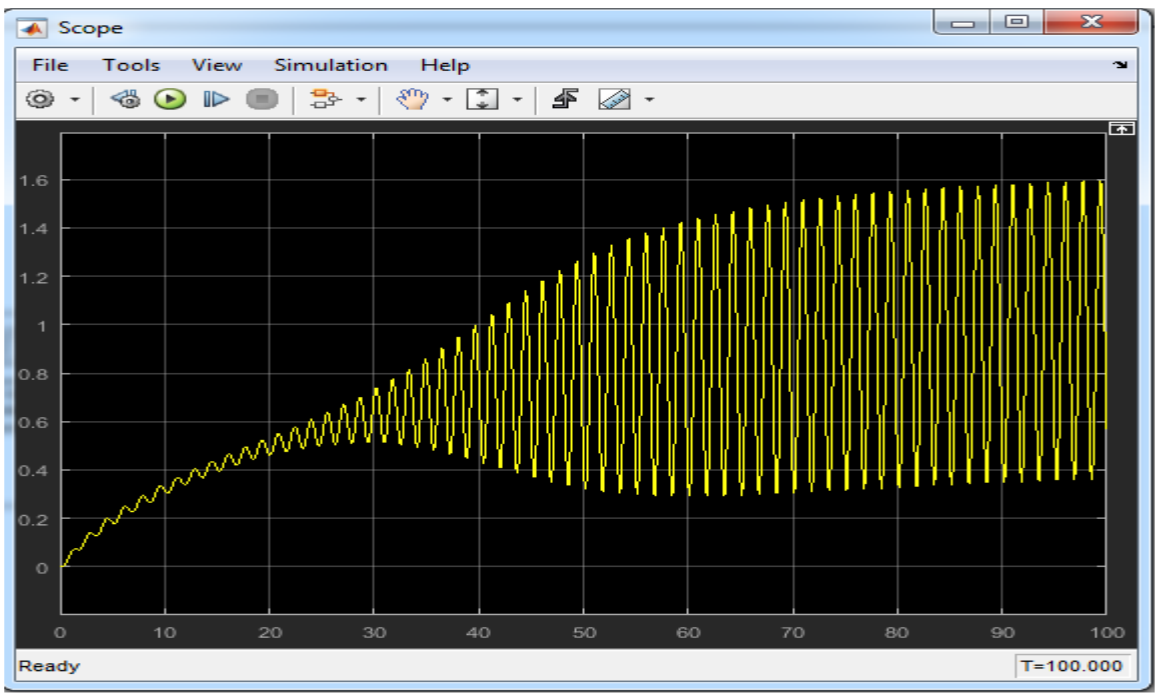
Defuzzification of the output function to get one crisp output.

1.2 Technical discussion

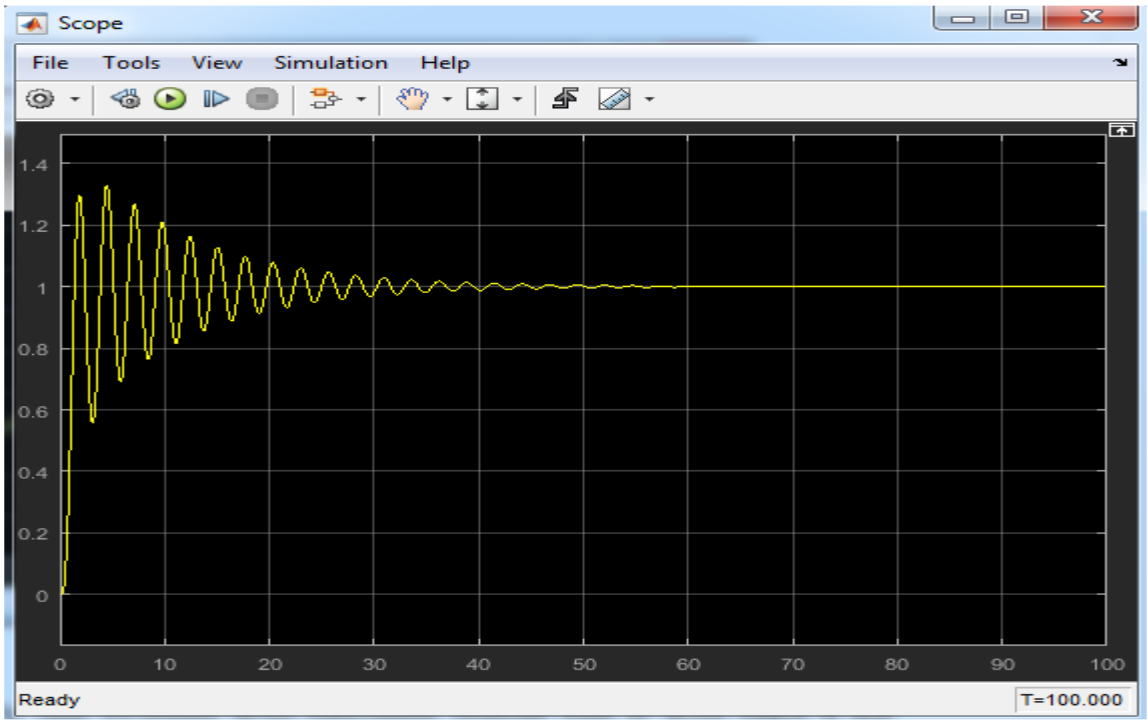
By increasing the value of the G0 only the system becomes unstable and the error keeps increasing.



Increasing the G1 only will increase the overshooting problem , thus it won't reach the desired set point



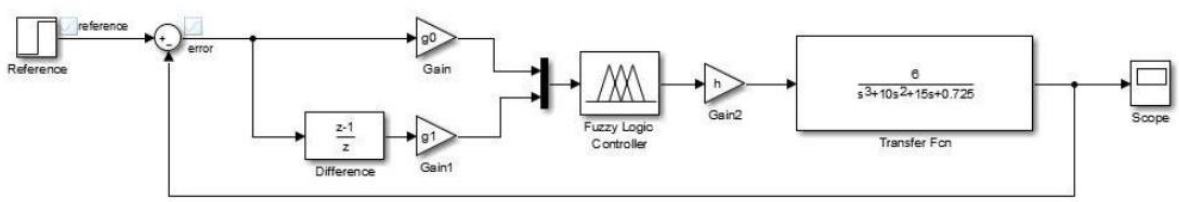
-By increasing H only the system will become stable but it will take a lot of time to reach the settling point

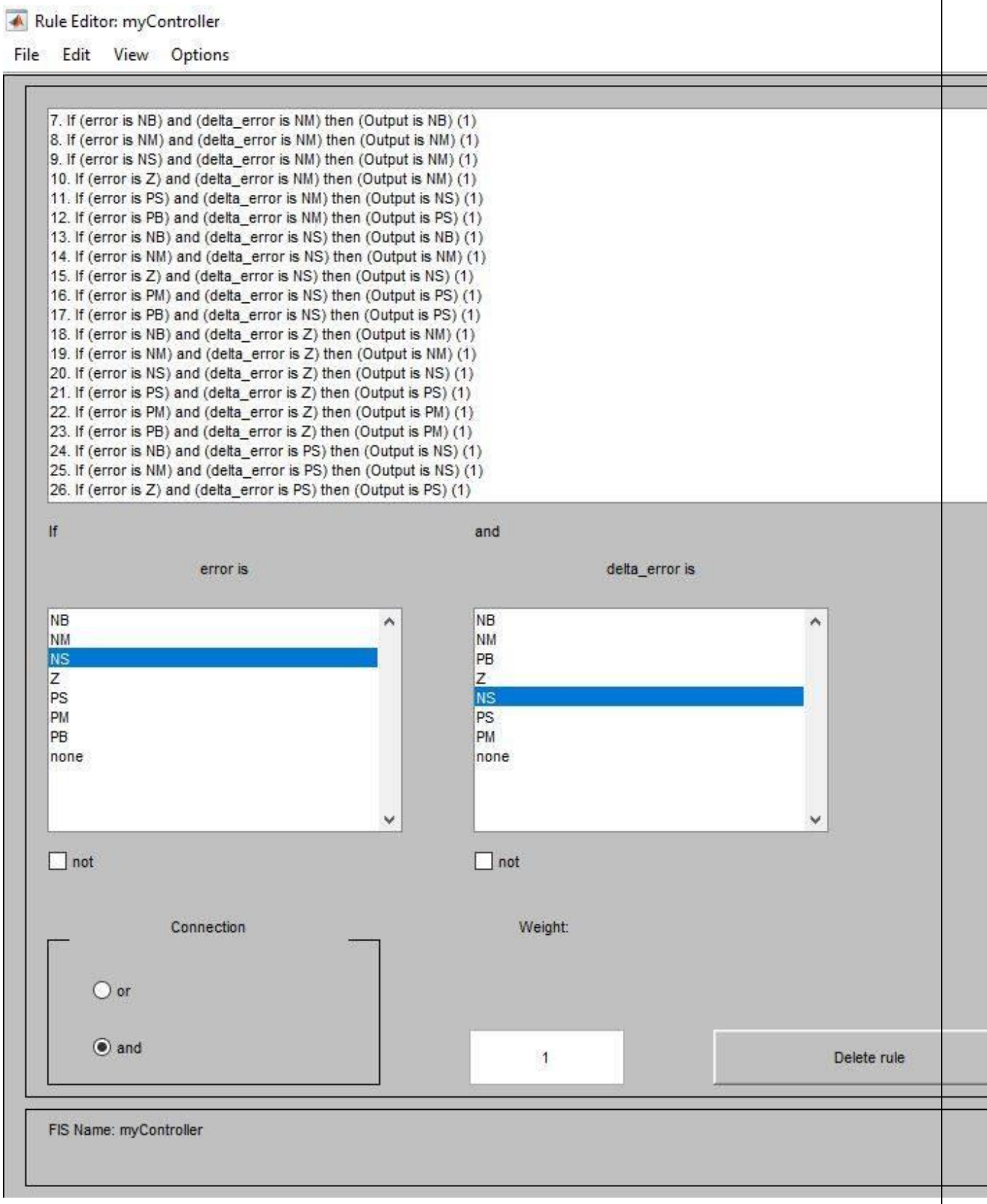


Fuzzy Rules

e \ Δe	NB	NM	NS	Z	PS	PM	PB
NB	NB	NB	NB	NM	NS	NS	ZE
NM	NB	NM	NM	NM	NS	ZE	PS
NS	NB	NM	NS	NS	ZE	PS	PM
Z	NB	NM	NS	ZE	PS	PM	PB
PS	NM	NS	ZE	PS	PS	PM	PB
PM	NS	ZE	PS	PM	PM	PM	PB
PB	ZE	PS	PS	PM	PB	PB	PB

System simulation




Rule Editor: myController
File Edit View Options
7. If (error is NB) and (delta_error is NM) then (Output is NB) (1)
8. If (error is NM) and (delta_error is NM) then (Output is NM) (1)
9. If (error is NS) and (delta_error is NM) then (Output is NM) (1)
10. If (error is Z) and (delta_error is NM) then (Output is NM) (1)
11. If (error is PS) and (delta_error is NM) then (Output is NS) (1)
12. If (error is PB) and (delta_error is NM) then (Output is PS) (1)
13. If (error is NB) and (delta_error is NS) then (Output is NB) (1)
14. If (error is NM) and (delta_error is NS) then (Output is NM) (1)
15. If (error is Z) and (delta_error is NS) then (Output is NS) (1)
16. If (error is PM) and (delta_error is NS) then (Output is PS) (1)
17. If (error is PB) and (delta_error is NS) then (Output is PS) (1)
18. If (error is NB) and (delta_error is Z) then (Output is NM) (1)
19. If (error is NM) and (delta_error is Z) then (Output is NM) (1)
20. If (error is NS) and (delta_error is Z) then (Output is NS) (1)
21. If (error is PS) and (delta_error is Z) then (Output is PS) (1)
22. If (error is PM) and (delta_error is Z) then (Output is PM) (1)
23. If (error is PB) and (delta_error is Z) then (Output is PM) (1)
24. If (error is NB) and (delta_error is PS) then (Output is NS) (1)
25. If (error is NM) and (delta_error is PS) then (Output is NS) (1)
26. If (error is Z) and (delta_error is PS) then (Output is PS) (1)
If
and
error is
delta_error is
NB
NM
NS
Z
PS
PM
PB
none
NB
NM
PB
Z
NS
PS
PM
none
not
not
Connection
or
and
Weight:
1
Delete rule
FIS Name: myController

Experimental Results

error

NB	Trapmf	[-1.72 -1.08 -0.92 -0.6]
NM	Trimf	[-0.9 -0.6 -0.3]
NS	Trimf	[-0.6 -0.3 0]
Z	Trimf	[-0.3 0 0.3]
PS	Trimf	[0 0.3 0.6]
PM	Trimf	[0.3 0.6 0.9]
PB	trampf	[0.6 0.92 1.08 1.72]

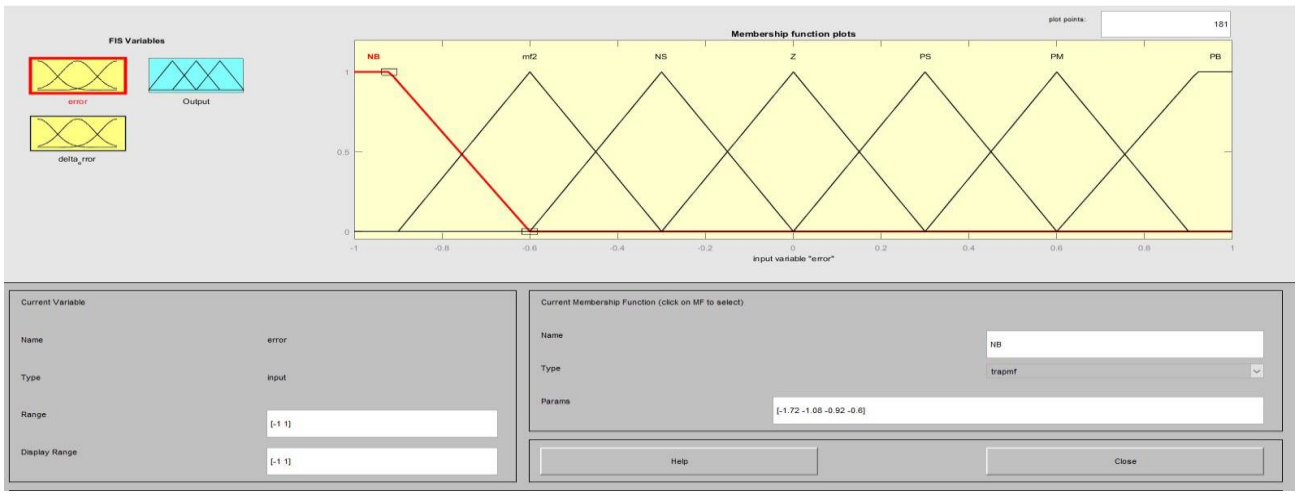
Delta_error

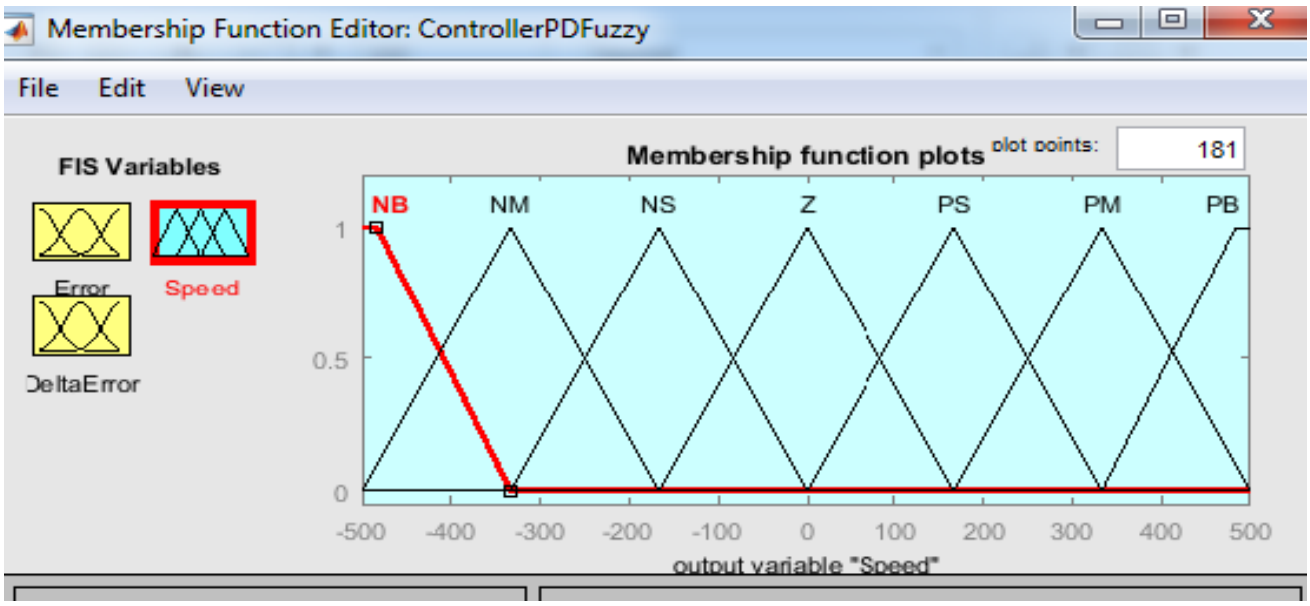
NB	Trapmf	[-1.72 -1.08 -0.92 -0.6]
NM	Trimf	[-0.9 -0.6 -0.3]
NS	Trimf	[-0.6 -0.3 0]
Z	Trimf	[-0.3 0 0.3]
PS	Trimf	[0 0.3 0.6]
PM	Trimf	[0.3 0.6 0.9]
PB	trampf	[0.6 0.92 1.08 1.72]

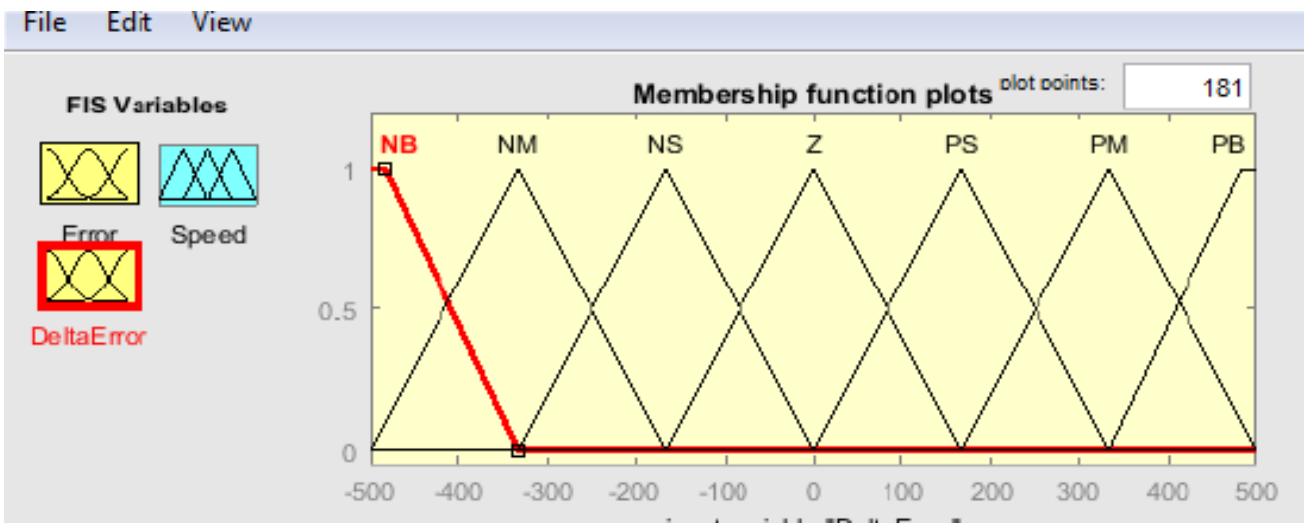
Output

NB	Trapmf	[-1.72 -1.08 -0.92 -0.6]
NM	Trimf	[-0.9 -0.6 -0.3]
NS	Trimf	[-0.6 -0.3 0]
Z	Trimf	[-0.3 0 0.3]
PS	Trimf	[0 0.3 0.6]
PM	Trimf	[0.3 0.6 0.9]
PB	trampf	[0.6 0.92 1.08 1.72]

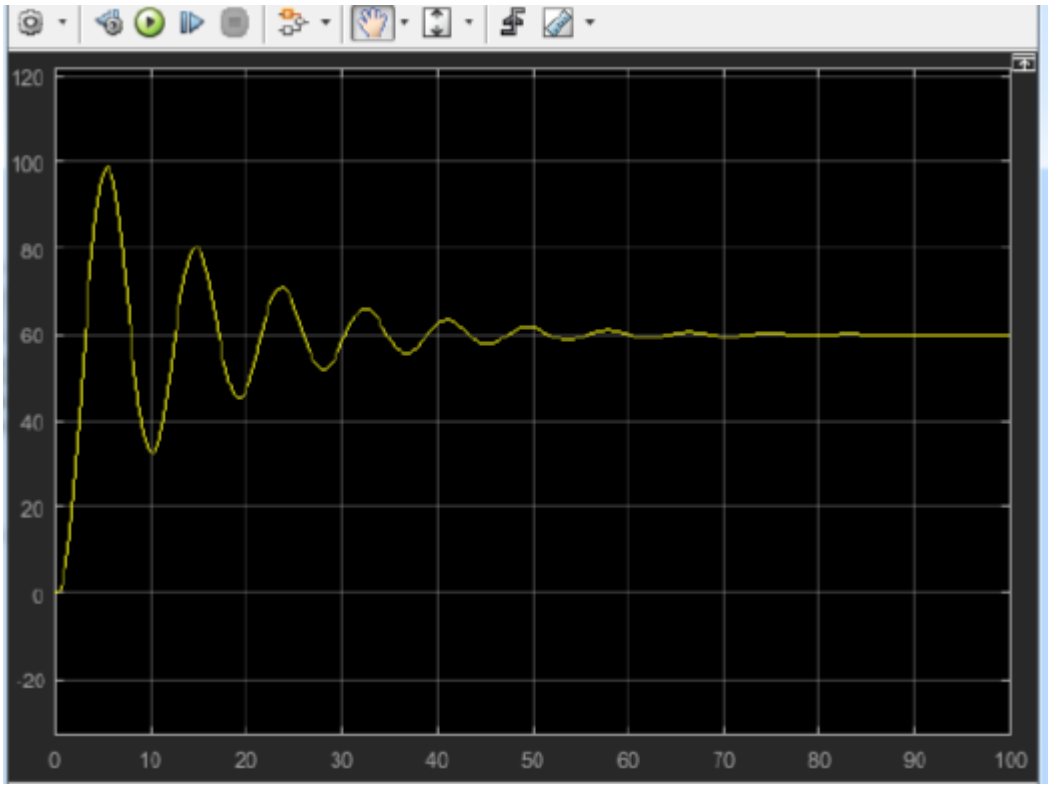
Fuzzy Membership functions:



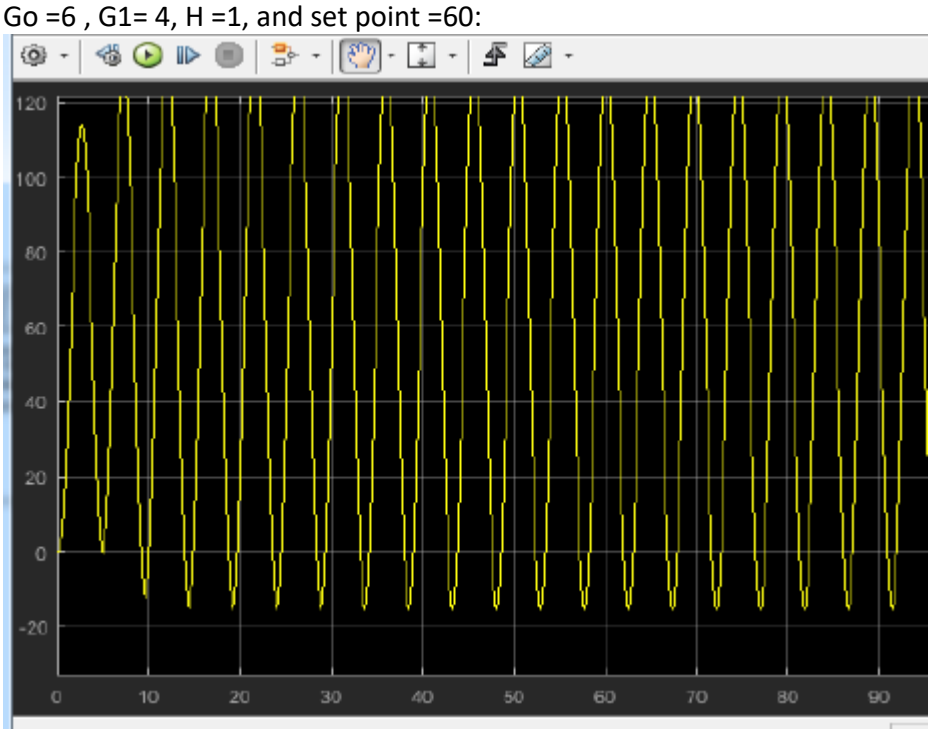




Changing scaling gains on system performance
Go =1 , G1= 1, H =1, and set point =60:

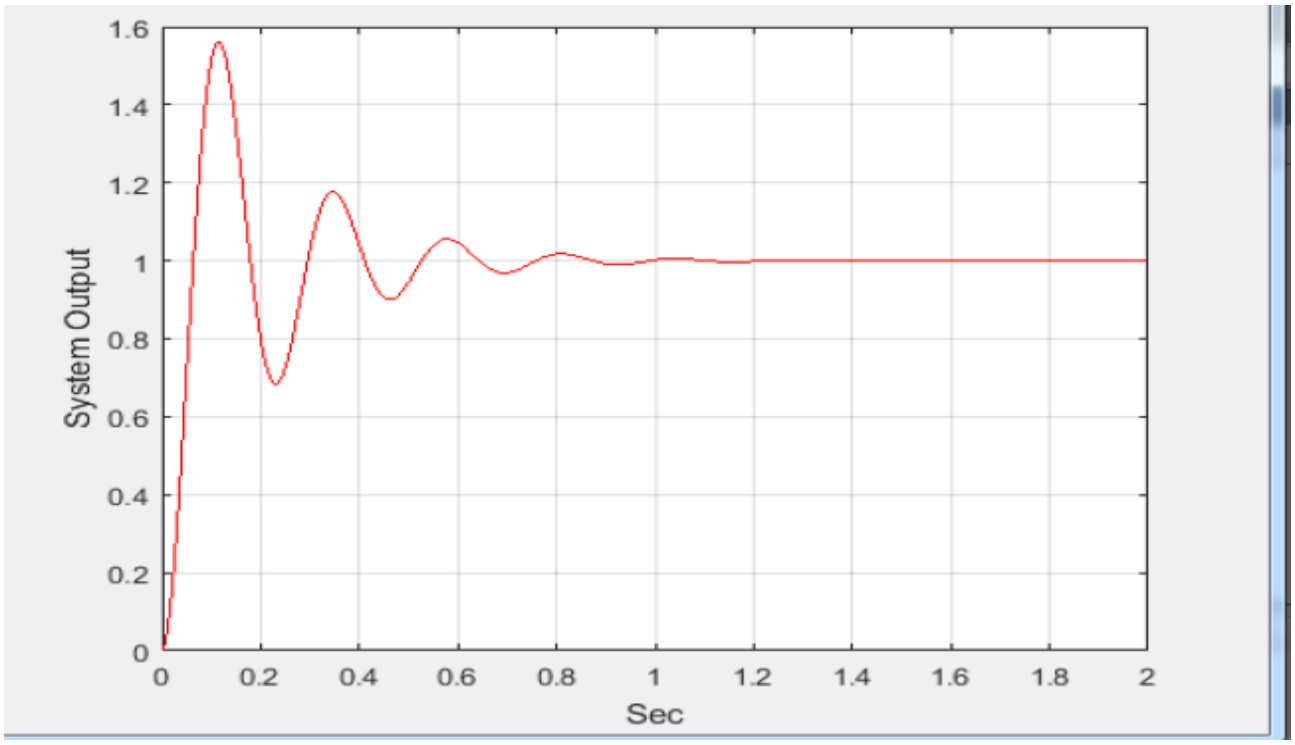


Go =6 , G1= 4, H =1, and set point =60:



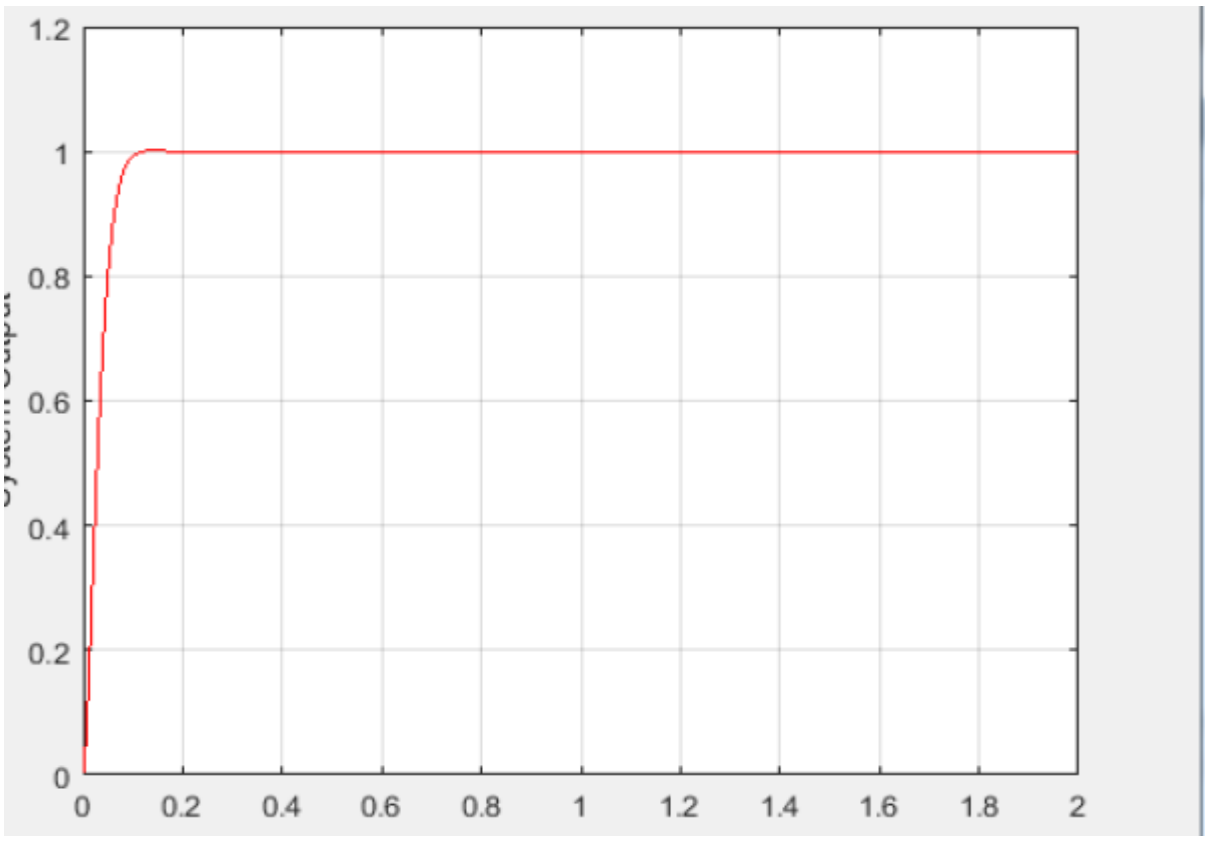
Output of ISE minimization:

Before minimization:



Integral of square error = 0.0558

After minimization:



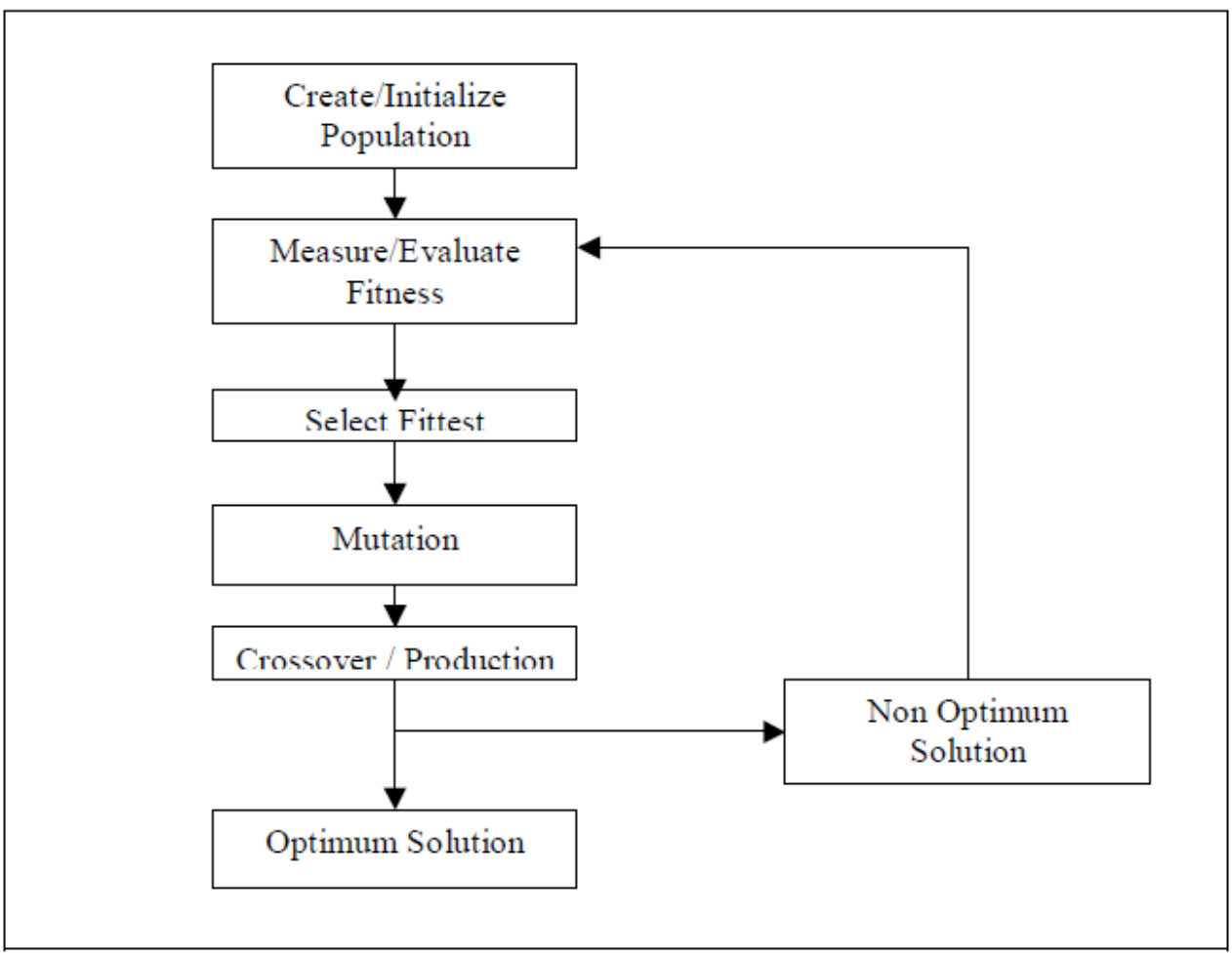
Integral of square error = 0.0210

2. Genetic Algorithm

Introduction

Genetic Algorithms (GA) are a stochastic global search method that mimics the process of natural evolution. It is one of the methods used for optimization.

The genetic algorithm starts with no knowledge of the correct solution and depends entirely on responses from its environment and evolution operators such as reproduction, crossover and mutation to arrive at the best solution. By starting at several independent points and searching in parallel, the algorithm avoids local minima and converging to sub optimal solutions.



Methods and Algorithm

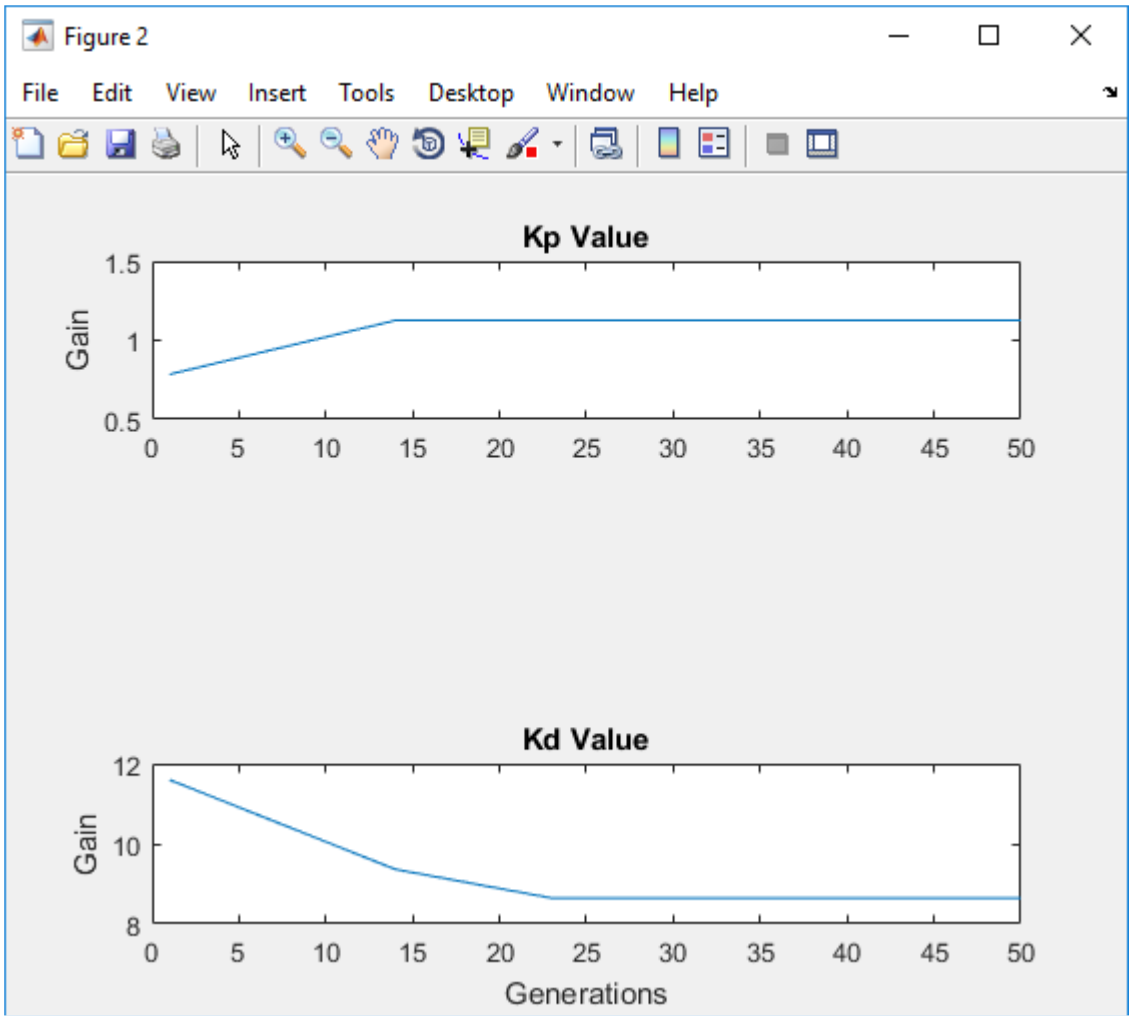
The steps involved in creating and implementing a genetic algorithm

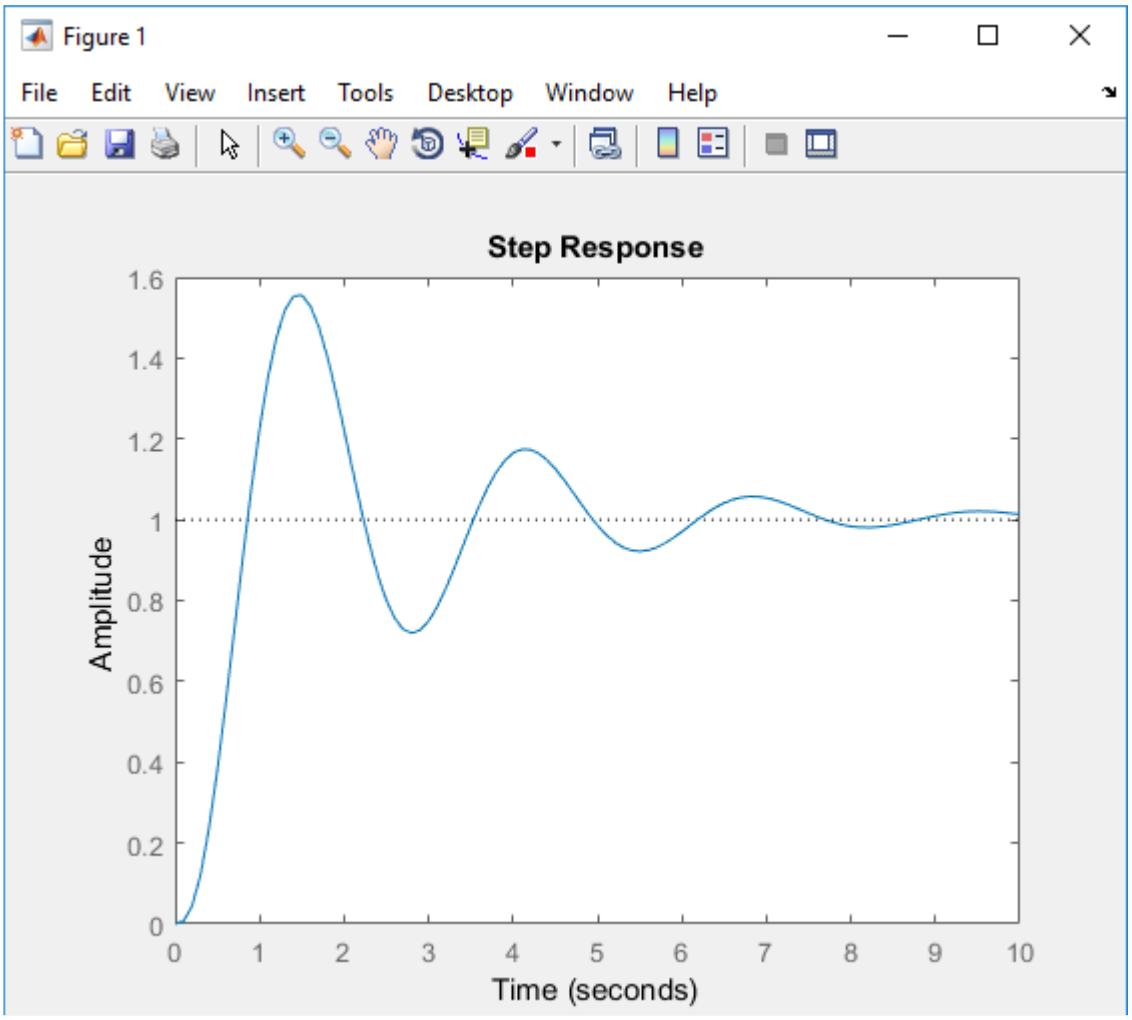
1. Generate an initial, random population of individuals for a fixed size.
2. Evaluate their fitness.
3. Select the fittest members of the population.
4. Reproduce using a probabilistic method (e.g., roulette wheel).
5. Implement crossover operation on the reproduced chromosomes (choosing probabilistically both the crossover site and the 'mates').
6. Execute mutation operation with low probability.
7. Repeat step 2 until a predefined convergence criterion is met

Experimental Results and discussions

We design 2 controllers in the genetic algorithm: PD and PID Controllers.
First, we design a PD controller but to eliminate oscillations and steady state error we use a PID controller; results and comparison are shown below.
2.3.1 GA With PD Controller

First Run






```
x =

    8.6438    1.1258   12.9104
```

X(1) = Kd
X(2) = Kp
X(3) = Ki (not used in PD controller)

Second Run

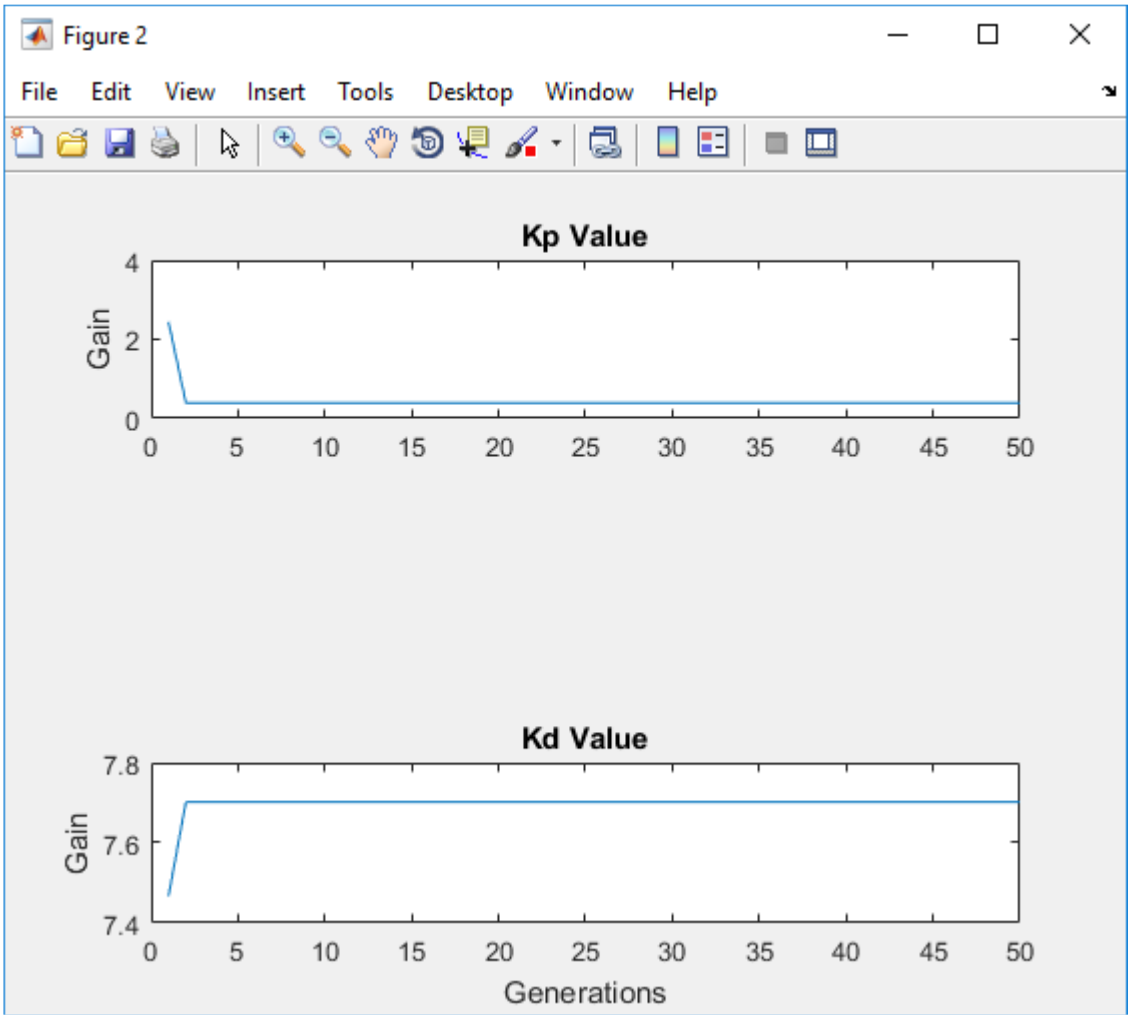
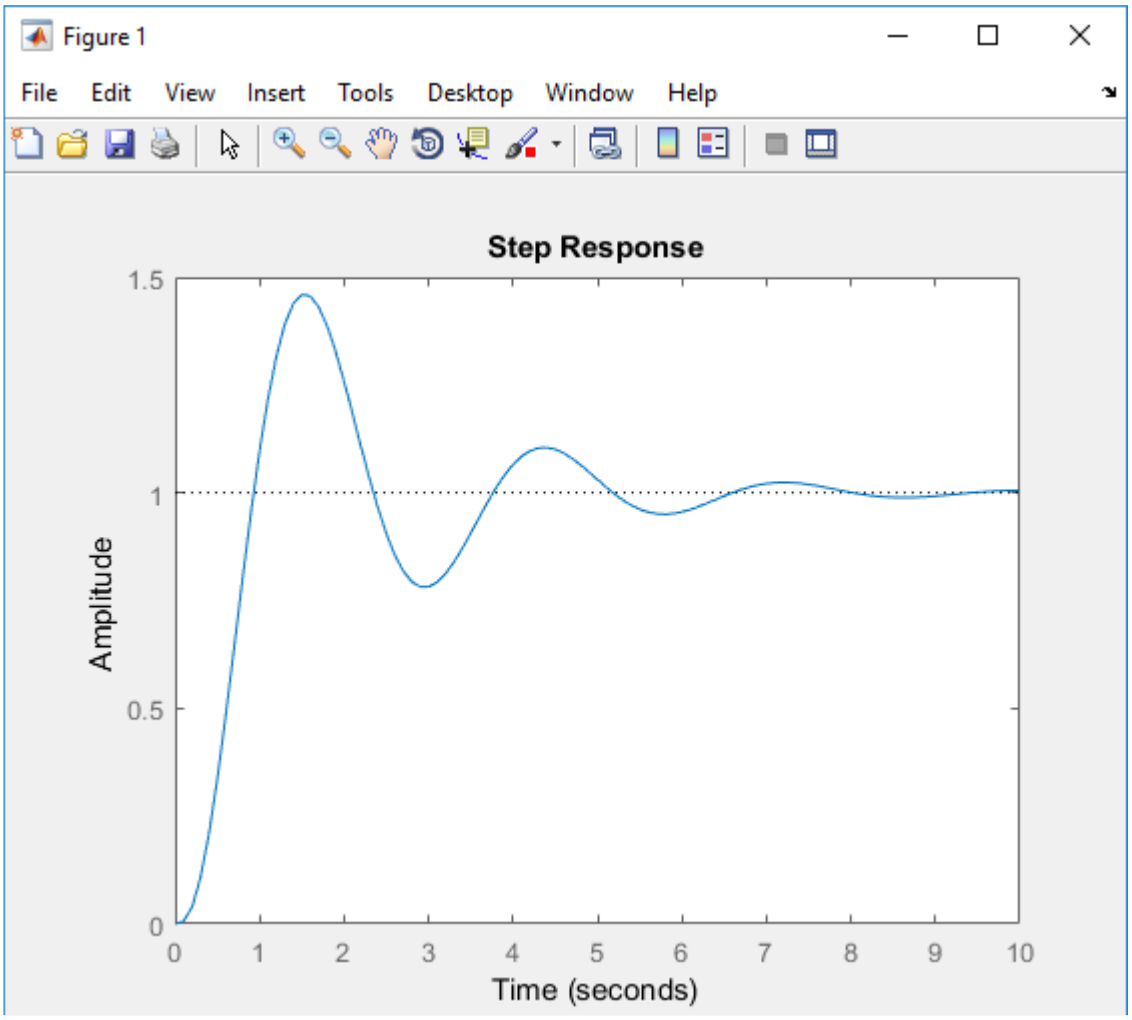


Figure 2
File Edit View Insert Tools Desktop Window Help
Kp Value
Gain
4
2
0
0 5 10 15 20 25 30 35 40 45 50
Kd Value
Gain
7.8
7.6
7.4
0 5 10 15 20 25 30 35 40 45 50
Generations

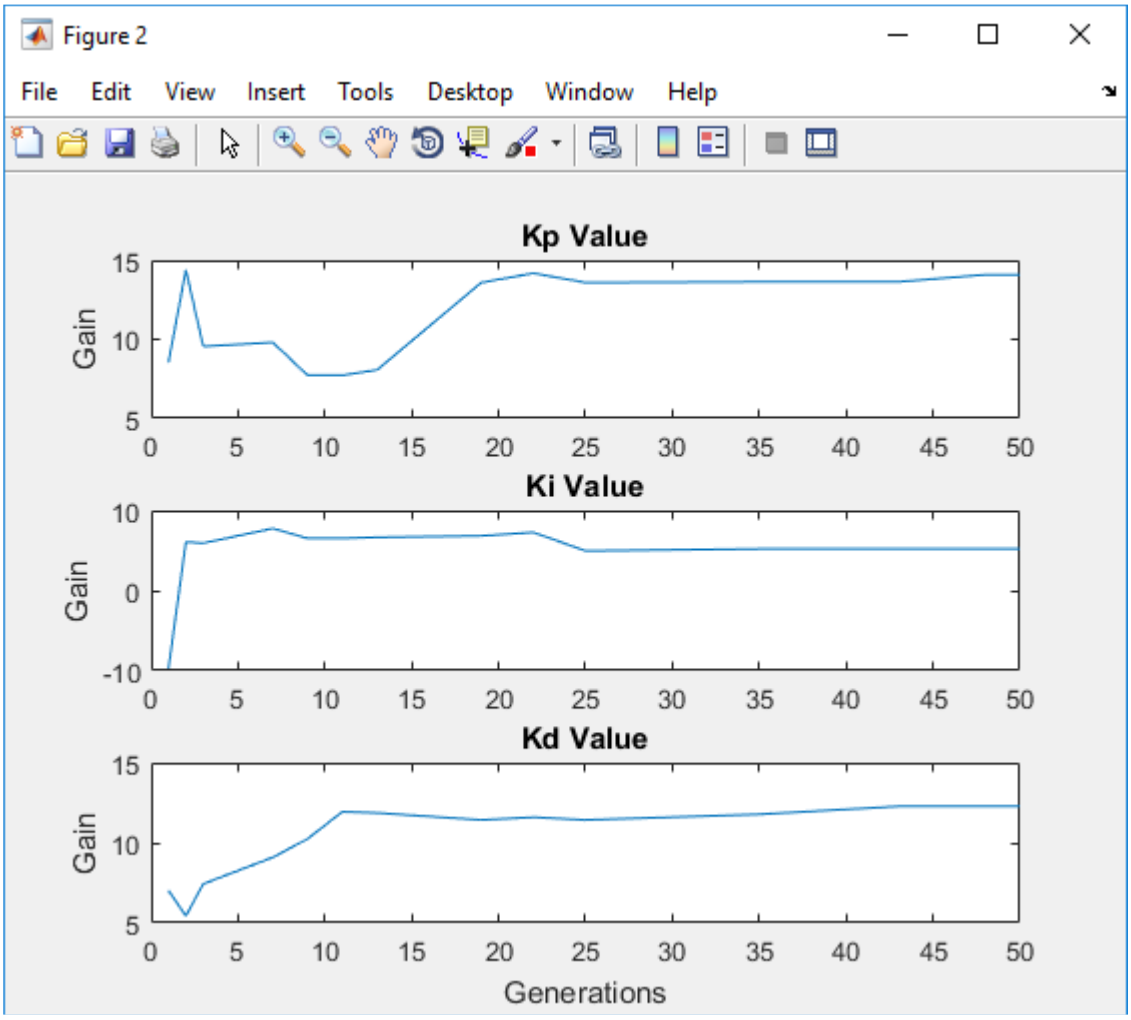



```
x =

    7.7017    0.3948   14.1803
```

GA With PID Controller

First Run



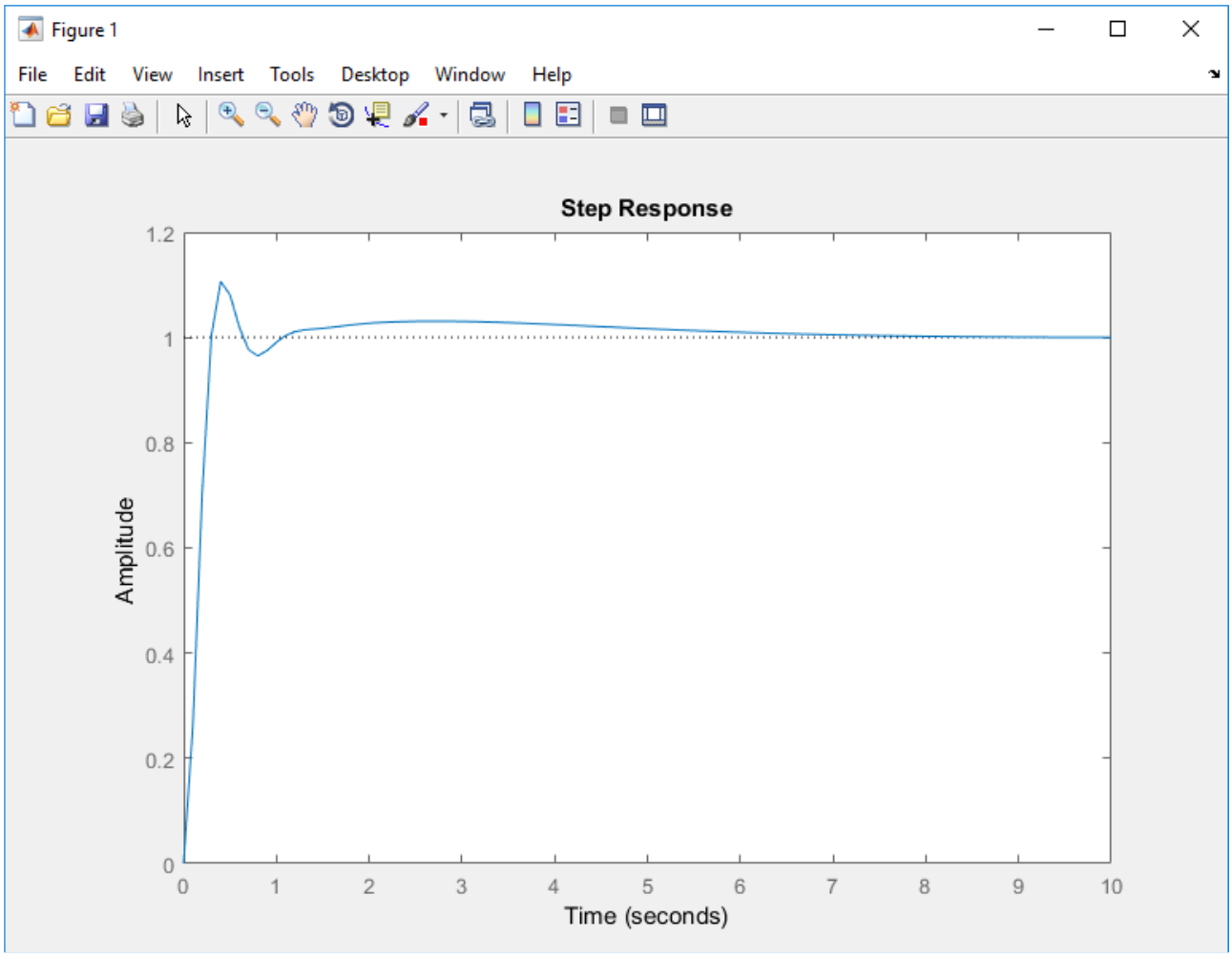
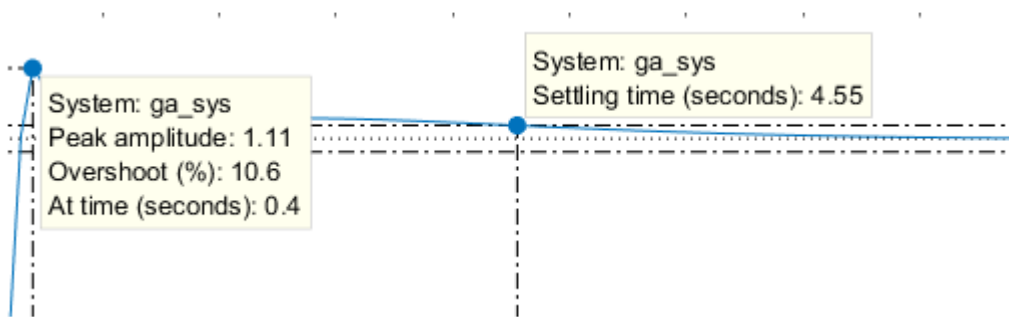


Figure 1
File Edit View Insert Tools Desktop Window Help
Step Response
Amplitude
Time (seconds)
0 0.2 0.4 0.6 0.8 1 1.2
0 1 2 3 4 5 6 7 8 9 10


System: ga_sys
Peak amplitude: 1.11
Overshoot (%): 10.6
At time (seconds): 0.4
System: ga_sys
Settling time (seconds): 4.55

Second Run

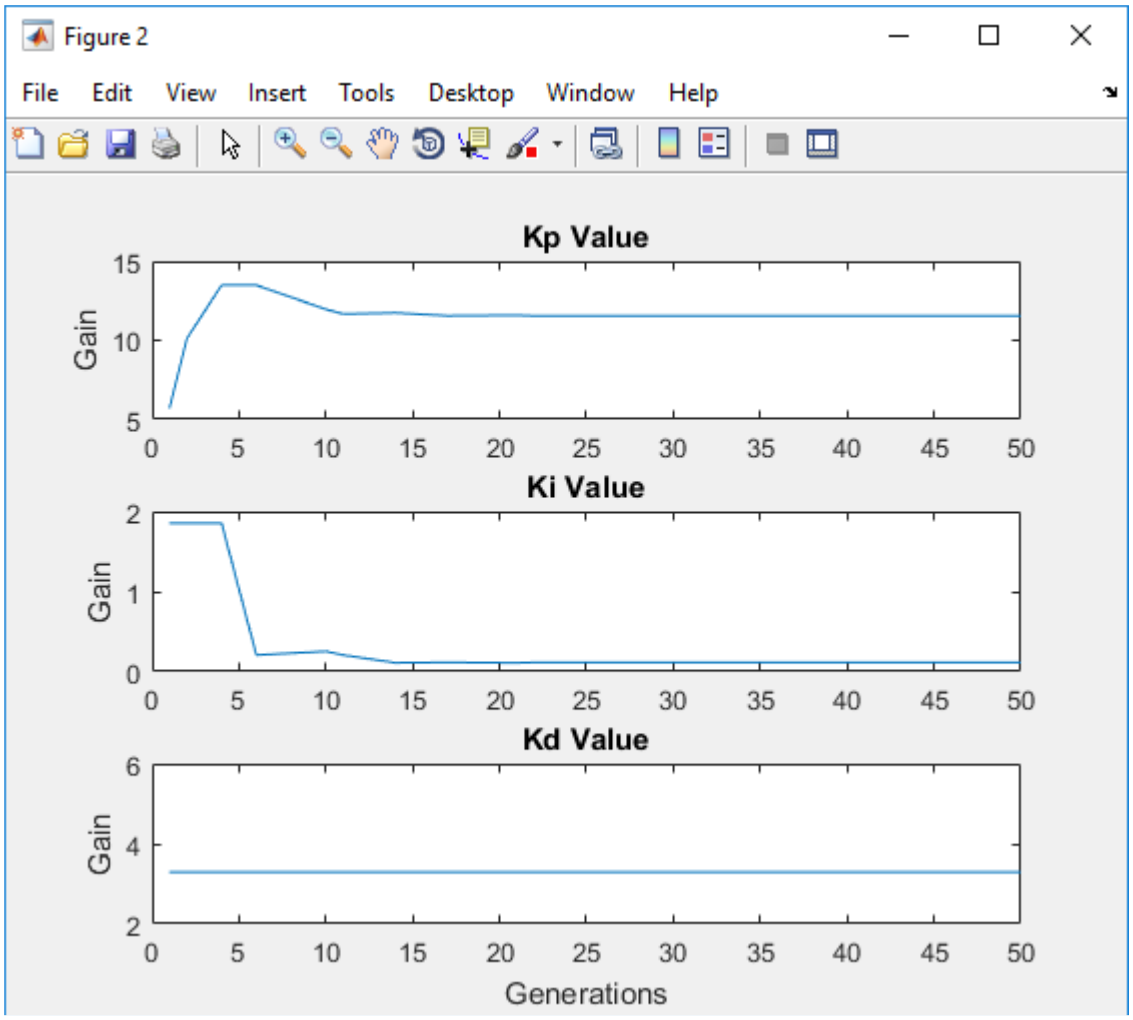


Figure 2
File Edit View Insert Tools Desktop Window Help
Kp Value
Gain
15
10
5
0 5 10 15 20 25 30 35 40 45 50
Ki Value
Gain
2
1
0
0 5 10 15 20 25 30 35 40 45 50
Kd Value
Gain
6
4
2
0 5 10 15 20 25 30 35 40 45 50
Generations

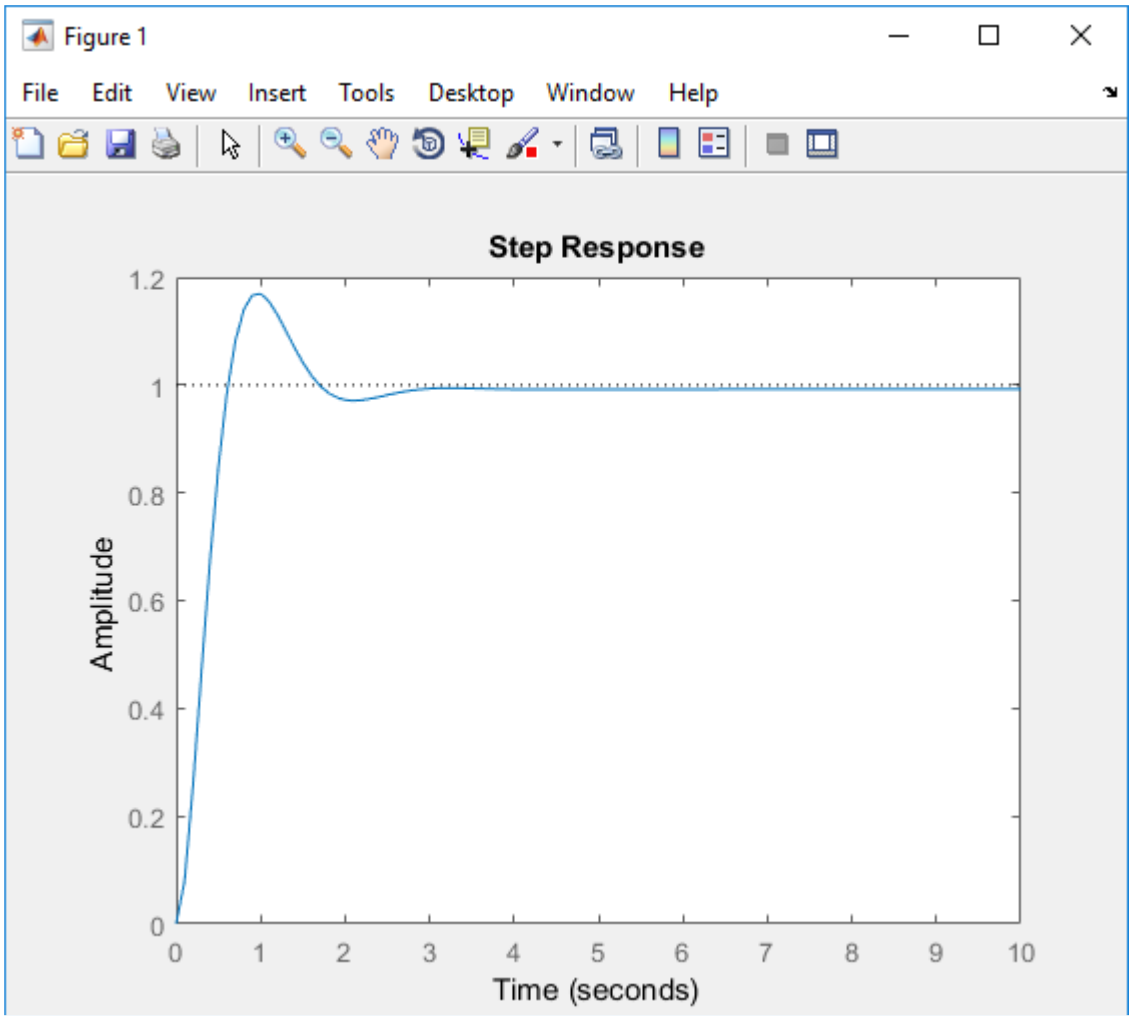
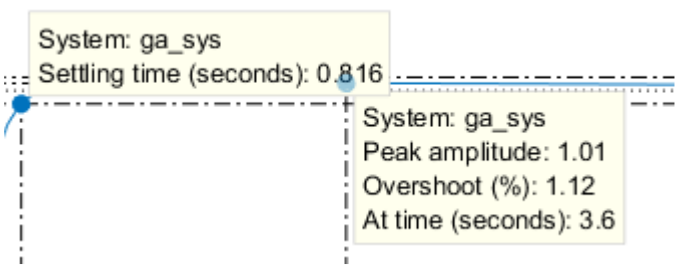


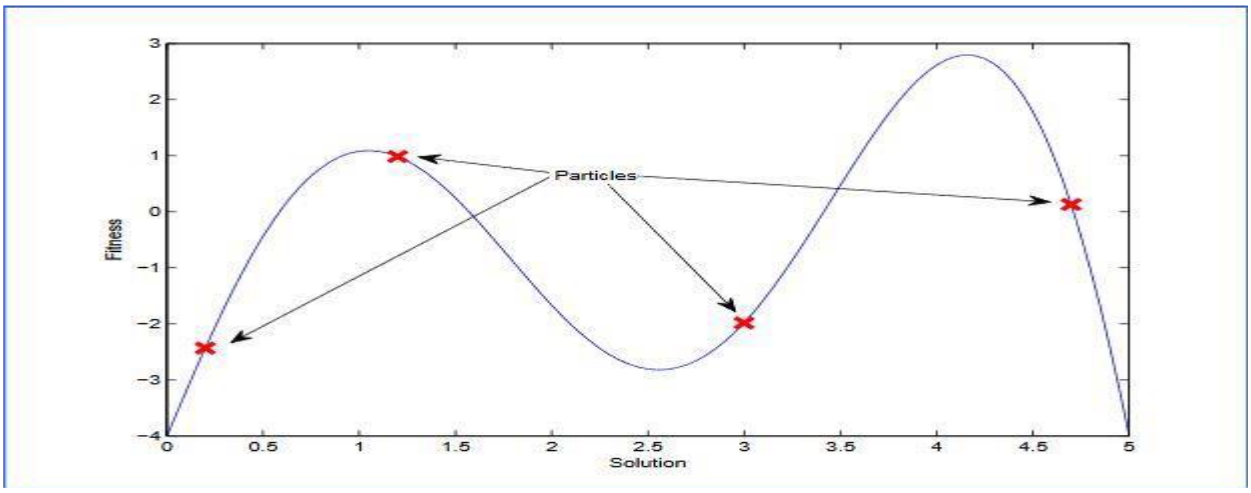
Figure 1
File Edit View Insert Tools Desktop Window Help
Step Response
Amplitude
Time (seconds)
0
0.2
0.4
0.6
0.8
1
1.2
0 1 2 3 4 5 6 7 8 9 10


System: ga_sys
Settling time (seconds): 0.816
System: ga_sys
Peak amplitude: 1.01
Overshoot (%): 1.12
At time (seconds): 3.6

Particle Swarm Optimization

PSO Concepts

The PSO algorithm maintains multiple solutions at one time During each iteration of the algorithm, each solution is evaluated by an objective function to determine its fitness Each solution is represented by a particle in the fitness landscape and each particle want to reach the (max Or min) value " local or Global



Each particle maintains:

a) Position in the search space
b) Velocity
c) Individual best position
In addition, the swarm maintains its global best position

PSO Algorithm

1. Evaluate fitness of each particle:
2. Update individual and global bests:
3. Update velocity and position of each particle

update the velocity

That's happen to each particle by using the equation of velocity depending on the equation

$$v_i(t+1)=wv_i(t)+c_1r_1[\hat{}x_i(t)-x_i(t)]+c_2r_2[g(t)-x_i(t)$$

1. I is particle index
2. W is inertial coefficient
3. C1,c1 are acceleration coefficient
4. R1 , r2 is random values

Position Update

Each particle's position is updated using this equation:

$$x_i(t+1)=x_i(t)+v_i(t+1)$$

Appendix

MATLAB Code

4. Appendix

4.1 Fuzzy Control MATLAB Code

```
%Clearing out any variables
clc;
clear all;
close all;
%set point I want to reach
r=1;
%Indicating time step
dt=0.001;
%The output of the system
c=0;
%The previous output of the system
cp=0;
%Previous error of the system
eprev=0;
%Implementing the fuzzy controller we created before
fis=readfis('ControllerPDFuzzy');
%Output of the fuzzy integrator
yi=0;
%Number of samples
Ns=2000;
%Creating a value MinError this value is big so that the if condition
%becomes true in the 1st itteration
MinError =10^10;
%Total error is the value of the integral of square
%after each itteration for every value of Tau and K
Total_Error = 0;
%This nested for loop assumes different values for Tau and
%K and calculates the ISE and it will store the value of K
%and Tau that will make the ISE minimum
for K = 1:5*r
for Tau = 20:-0.01:0.01
for i=1:Ns
e=r-c;% calculates the error
de=(e-eprev);%Calculates the error Change
u=evalfis([e de],fis);%Compute controller output
yi=yi+dt*u; %Integrator Output
c=Tau*cp/dt+K*yi;
c=c/(1+Tau/dt); %Lag Output
cp=c;%Storing Previous Output
eprev=e;%Storing Previous Errror
Y(i)=c;%Storing System Output
end
%calculation of the ISE
Total_Error = dt*sum((Y-1).^2);
%Checking if the ISE is smaller than The MinError where
%Min error is 10^10 at first then it will be The value of
%The Min ISE if it is smaller than the previous value and it will
%also store The values of K and Tau
if(Total_Error < MinError)
MinError = Total_Error
Tau1 = Tau;
K1=K;
end
```

```
%Resetting the variables before each itteration
Total_Error =0;
c=0; cp=0; e=0; Y=0; eprev=0; yi=0; u=0;
end
end

c=0;%System Output
cp=0;%Previous Output
eprev=0;%Prev Error
Tau=Tau1;%Lag Time Constant
K=K1;%System Gain
yi=0;%Integrator Output
Ns=2000;%Total number of Samples
for i=1:Ns
e=r-c;%Error
de=(e-eprev);%Error Change
u=evalfis([e de],fis);%Compute controller
output
yi=yi+dt*u; %Integrator Output
c=Tau*cp/dt+K*yi;
c=c/(1+Tau/dt); %Lag Output
cp=c;%Storing Previous Output
eprev=e;%Storing Previous Errror
Y(i)=c;%Storing System Output
end
%Computing the minimum Sum of Squared Error
ISE1=dt*sum((Y-1).^2)
plot(dt*[0:(Ns-1)],Y,'r-')
xlabel('Sec')
ylabel('System Output')
grid
hold on
```

Genetic Algorithm MATLAB Code:

All functions used in GA from GAOT (genetic algorithm Optimization toolbox)
https://github.com/estsauver/GAOT

GA main Code :

```
global sys_controlled
global time
global sysrl %Plant.
%_____________________________________________________________________
%Defining sysrl
den1=[1 10 15 0.725];
num1=6;
sysrl=tf(num1,den1);
%_____________________________________________________________________
%Initialising the genetic algorithm
populationSize=14;
variableBounds=[-15 + (15+15)*rand(1,30);-15 + (15+15)*rand(1,30);-15 +
(15+15)*rand(1,30)];
evalFN='PID_objfun_MSE';
%Change this to relevant object function
evalOps=[];
options=[1e-6 1];
initPop=initializega(populationSize,variableBounds,evalFN,evalOps,options);
%_____________________________________________________________________
%Setting the parameters for the genetic algorithm
bounds=[-15 + (15+15)*rand(1,30);-15 + (15+15)*rand(1,30);-15 +
(15+15)*rand(1,30)];
evalFN='PID_objfun_MSE';%change this to relevant object function
evalOps=[];
startPop=initPop;
opts=[1e-6 1 0];
termFN='maxGenTerm';
termOps=50;
selectFN='normGeomSelect';
selectOps=0.08;
xOverFNs='arithXover';
xOverOps=4;
mutFNs='unifMutation';
mutOps=8;
%_____________________________________________________________________
%Iterating the genetic algorithm
[x,endPop,bPop,traceInfo]=ga(bounds,evalFN,evalOps,startPop,opts,...
termFN,termOps,selectFN,selectOps,xOverFNs,xOverOps,mutFNs,mutOps);
%_____________________________________________________________________
%Plotting Genetic algorithm controller
den1=[1 10 15 0.725];
num1=6;
sysrl=tf(num1,den1);
%Creating the optimal PID controller from GA results
ga_pid=tf([x(1) x(2) x(3)],[1 0]);
ga_sys=feedback(series(ga_pid,sysrl),1);
figure(1)
%hold on;
step(ga_sys,time);%Green-genetic algorithm
%_____________________________________________________________________
%Plotting best population progress
figure(2)
subplot(3,1,1),plot(bPop(:,1),bPop(:,3)),...
title('Kp Value'), ylabel('Gain');
subplot(3,1,2),plot(bPop(:,1),bPop(:,4)),...
title('Ki Value'), ylabel('Gain');
subplot(3,1,3),plot(bPop(:,1),bPop(:,2)),...
title('Kd Value'),xlabel('Generations'), ylabel('Gain'); %
```

GA PID Objective Function

```
%_____________________________________________________________________
function [x_pop, fx_val]=PID_objfun_MSE(x_pop,options)
global sys_controlled
global time
global sysrl
%_____________________________________________________________________
Kp=x_pop(2);
Ki=x_pop(3);
Kd=x_pop(1);
%_____________________________________________________________________
%creating the PID controller from current values
pid_den=[1 0];
pid_num=[Kd Kp Ki];
pid_sys=tf(pid_num,pid_den); %overall PID controller
%Placing PID controller in unity feedback system with 'sysrl'
sys_series=series(pid_sys,sysrl);
sys_controlled=feedback(sys_series,1);
%_____________________________________________________________________
time =0:0.1:10;
[y t] = step(sys_controlled,time); % Step response of closed-loop
%_____________________________________________________________
%Calculating the error
for i=1:101
error(i) = 1-y(i);
end
%Calculating the MSE
error_sq = error*error';
MSE=error_sq/max(size(error));
%_____________________________________________________________________
%Ensuring controlled system is stable
poles=pole(sys_controlled);
if poles(1)>0
MSE=100e300;
elseif poles(2)>0
MSE=100e300;
elseif poles(3)>0
MSE=100e300;
elseif poles(4)>0
MSE=100e300;
%elseif poles(5)>0
%MSE=100e300;
end
fx_val=1/MSE;
%_____________________________________________________________________
```