Faculty of Engineering
Credit Hours Engineering Programs

Mechatronics Engineering and Automation

Academic Year 2019/2020 – Spring 2019

CSE 488

Computational Intelligence

Project No. (1)

Optimization

Date due 30-Mar-19
Date handed in 30-Mar-19

Submitted by:

Ahmed Mohammed Abdullah **14P8024**

Contents

Problem Definition and Importance

Optimization is finding the parameters which optimize the solution and decrease the error as much as possible.
The desired systems can often – but not always- be expressed in the term of a mathematical function. Here, in classical optimization we are concerned with optimization of differentiable and continuous mathematical function. To reach the local minimum or if possible the global minimum that decrease the error to its lowest value.

Methods and Algorithms

PID Z-N Method

Ki = kd =0 and start increasing the value of Kp until the signal critical oscillate

check the periodic time and then substitute in Z-N table

Conventional Gradient Descent:

It is a first-order iterative optimization algorithm for finding the minimum of a function. Is simple and does not depend on dimensionality. However, selection of eta(learning rate) will dramatically affect the algorithm. If eta is too small, convergence will take a long time to reach for the solution. On the other hand, if eta is too big, overshoot the minimum and it will diverge.

1) Initialize n=0 and Select xn,k,gradient,eta. Ɛ = (10^-k)
2) Xn+1 = Xn- eta * gradient(Xn)
3) If Abs (Xn+1-Xn) less than Ɛ or if n > nmax , STOP
4) n = n+1
5) GoTo 2).

Line Search Gradient Descent (Steepest):

It is similar to the Conventional Gradient Descent, However eta value is optimized before selection to reach the solution in less steps.

1) Initialize n=0. And select Xn , K, Ɛ=(10^-k)
2) α(eta)=f(Xn – eta*gradient(Xn)
3) Xn+1=Xn-eta*gradient (Xn)
4) If Abs(Xn+1 – Xn) < Ɛ or if n > nmax , STOP
5) n=n+1
6) GoTo 3)

Newton Raphson Gradient Descent:

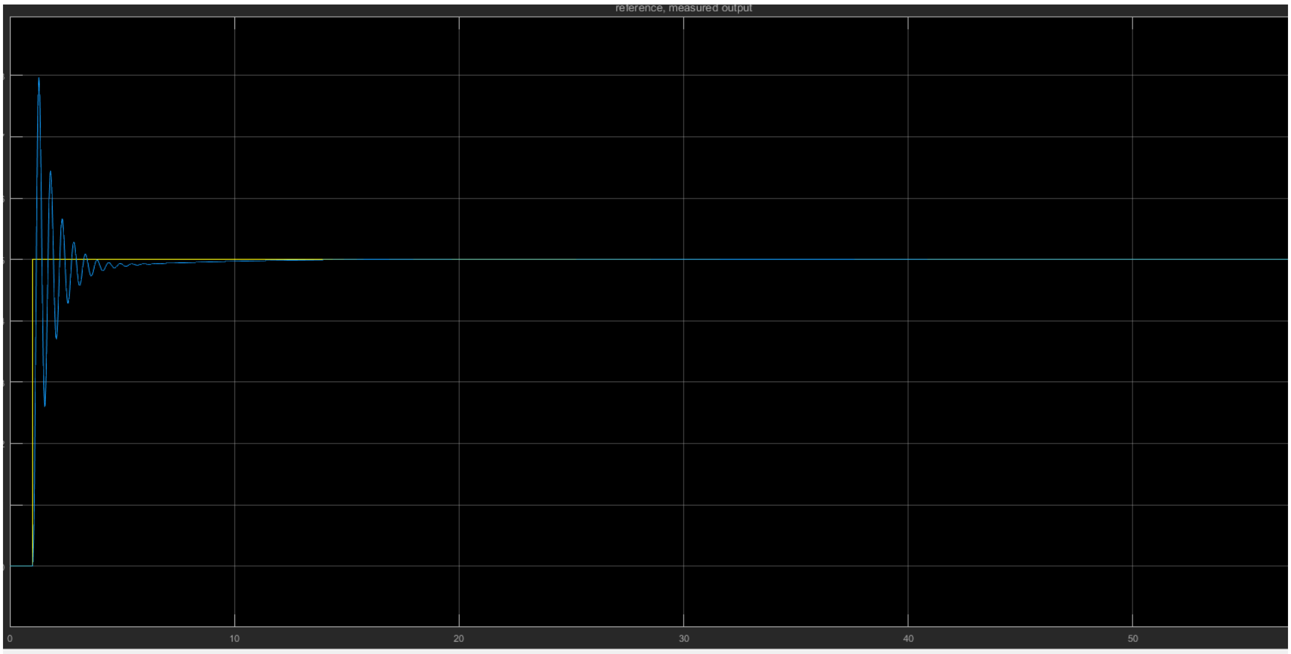
This method is faster than the steepest gradient Decent. However the computation of the inverse of the “hessian” is costly

1) Initialize n=0 , Select K, nmax, Ɛ=(10^-k) and X(0) such that Det(Hessian(X(0))) not equal zero
2) Compute Hessian
3) Xn+1=Xn- inverse(Hessian) gradient(Xn)
4) If Abs(Xn+1 – Xn) < Ɛ or if n > nmax , STOP
5) n=n+1
6) GoTo 3)

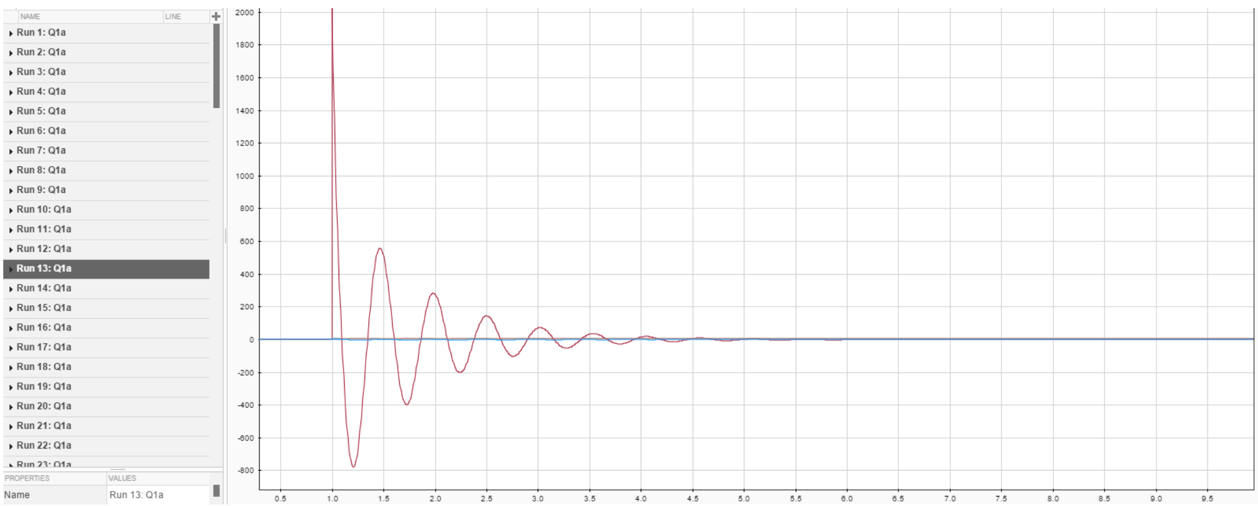
Experimental Results and Discussions

Q)1
a) kp optimal value is 5.000000
ki optimal value is 0.500000
kd optimal value is 20.000000
Total number of iteration 109.000000

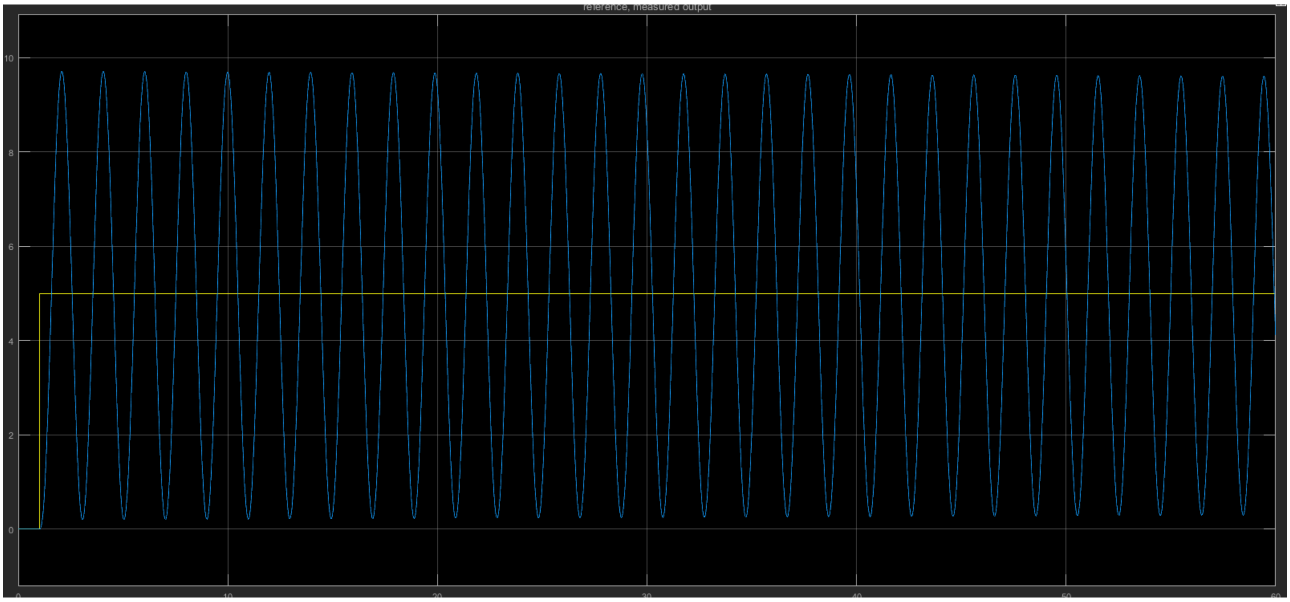
Number of iteration is small to speed up the process
b) at best obtained parameters
the input signal is a step signal starts after 1 second with magnitude of 5
.



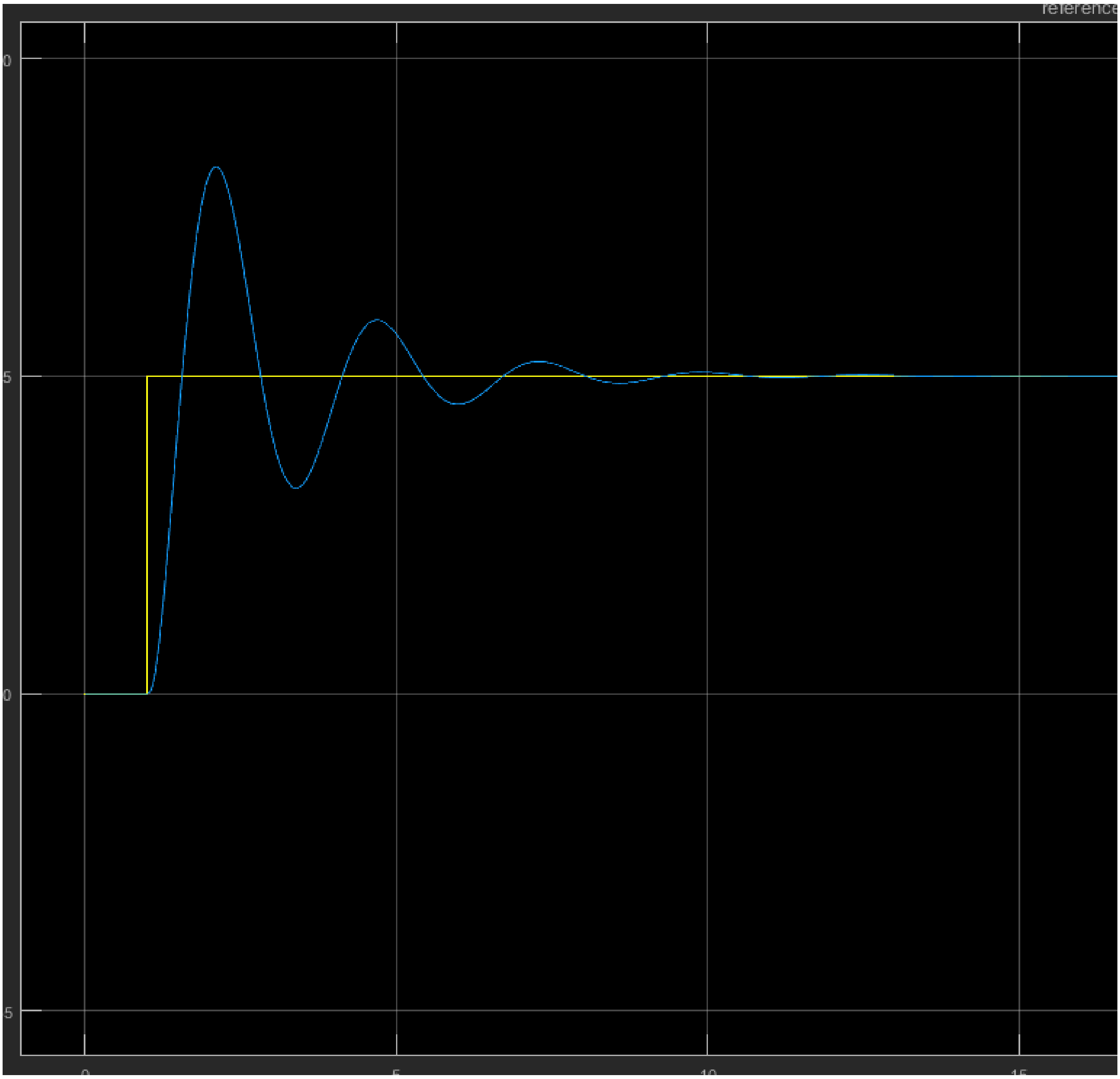
c) from zero to 109 in each case
different outputs



Q1)D
Critically stable



Periodic time is 2seconds
According to Wikipedia:
Kp=0,6 kcr =6.6
Ki =1.2 kcr /T =6.6
Kd=(0.6 kcr* T)/40=1.66



We can find that the manual tuning will result to a better response that is because, while using brute force the number of iteration was very small only 120 iterations and it is preferable to run another test with atleast 100k iterations to reach for a better output

Question2)a
```
gradient(f,[x1 x2 x3])
```

The gradient Vector is

<u>With respect to x1</u>

9*x1 - 3*cos(x2*x3) + 2*x1*(sin(x3) - 81*(x2 + 1/10)^2 + x1^2 + 53/50) - x2*exp(-x1*x2)*(20*x3 + exp(-x1*x2) + 5332248173269055/562949953421312) - 3/2

<u>with respect to x2</u>

- (162*x2 + 81/5)*(sin(x3) - 81*(x2 + 1/10)^2 + x1^2 + 53/50) - x1*exp(-x1*x2)*(20*x3 + exp(-x1*x2) + 5332248173269055/562949953421312) - x3*sin(x2*x3)*(cos(x2*x3) - 3*x1 + 1/2)

<u>with respect to x3</u>

400*x3 + 20*exp(-x1*x2) + cos(x3)*(sin(x3) - 81*(x2 + 1/10)^2 + x1^2 + 53/50) - x2*sin(x2*x3)*(cos(x2*x3) - 3*x1 + 1/2) + 26661240866345275/140737488355328

```
Q2a

radientMatrix =

                  9*x1 - 3*cos(x2*x3) + 2*x1*(sin(x3) - 81*(x2 + 1/10)^2 + x1^2 + 53/50) - x2*exp(-x1*x2)*(20*x3 + exp(-x1*x2) + 5332248173269055/562949953421312) - 3/2
(162*x2 + 81/5)*(sin(x3) - 81*(x2 + 1/10)^2 + x1^2 + 53/50) - x1*exp(-x1*x2)*(20*x3 + exp(-x1*x2) + 5332248173269055/562949953421312) - x3*sin(x2*x3)*(cos(x2*x3) - 3*x1 + 1/2)
               400*x3 + 20*exp(-x1*x2) + cos(x3)*(sin(x3) - 81*(x2 + 1/10)^2 + x1^2 + 53/50) - x2*sin(x2*x3)*(cos(x2*x3) - 3*x1 + 1/2) + 26661240866345275/140737488355328
```

Q2)b
```
hessian(f,[x1 x2 x3])
```

X11= 2*sin(x3) - 162*(x2 + 1/10)^2 + 6*x1^2 + x2^2*exp(-2*x1*x2) + x2^2*exp(-x1*x2)*(20*x3 + exp(-x1*x2) + 5332248173269055/562949953421312) + 278/25

X12= 3*x3*sin(x2*x3) - 2*x1*(162*x2 + 81/5) - exp(-x1*x2)*(20*x3 + exp(-x1*x2) + 5332248173269055/562949953421312) + x1*x2*exp(-2*x1*x2) + x1*x2*exp(-x1*x2)*(20*x3 + exp(-x1*x2) + 5332248173269055/562949953421312)

X13= 3*x2*sin(x2*x3) - 20*x2*exp(-x1*x2) + 2*x1*cos(x3)

X21= 3*x3*sin(x2*x3) - 2*x1*(162*x2 + 81/5) - exp(-x1*x2)*(20*x3 + exp(-x1*x2) + 5332248173269055/562949953421312) + x1*x2*exp(-2*x1*x2) + x1*x2*exp(-x1*x2)*(20*x3 + exp(-x1*x2) + 5332248173269055/562949953421312)

X22= 13122*(x2 + 1/10)^2 - 162*sin(x3) + x3^2*sin(x2*x3)^2 + (162*x2 + 81/5)^2 - 162*x1^2 + x1^2*exp(-2*x1*x2) + x1^2*exp(-x1*x2)*(20*x3 + exp(-x1*x2) + 5332248173269055/562949953421312) - x3^2*cos(x2*x3)*(cos(x2*x3) - 3*x1 + 1/2) - 4293/25

X23= x2*x3*sin(x2*x3)^2 - cos(x3)*(162*x2 + 81/5) - 20*x1*exp(-x1*x2) - sin(x2*x3)*(cos(x2*x3) - 3*x1 + 1/2) - x2*x3*cos(x2*x3)*(cos(x2*x3) - 3*x1 + 1/2)

X31= 3*x2*sin(x2*x3) - 20*x2*exp(-x1*x2) + 2*x1*cos(x3),

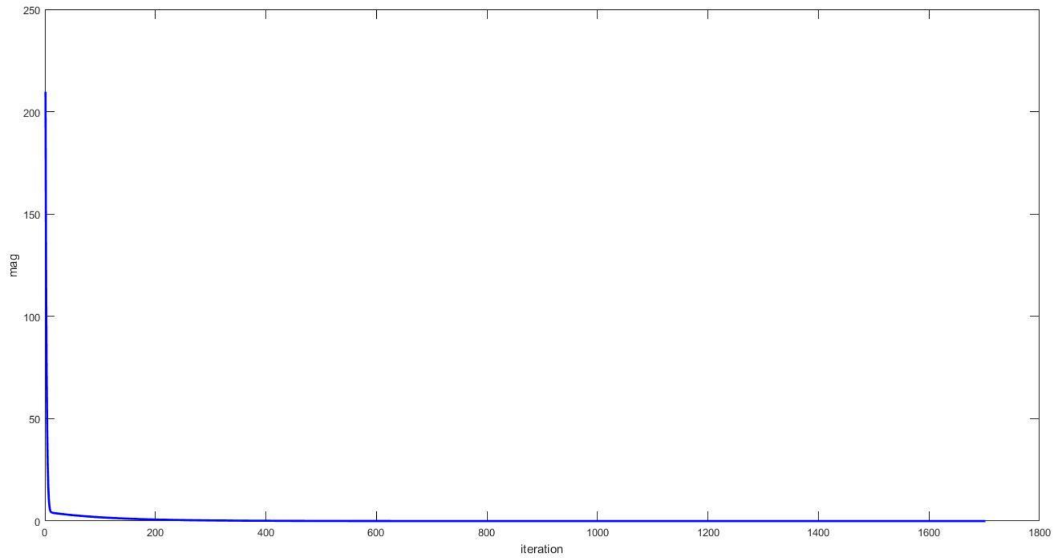
X32= x2*x3*sin(x2*x3)^2 - cos(x3)*(162*x2 + 81/5) - 20*x1*exp(-x1*x2) - sin(x2*x3)*(cos(x2*x3) - 3*x1 + 1/2) - x2*x3*cos(x2*x3)*(cos(x2*x3) - 3*x1 + 1/2)

X33= cos(x3)^2 - sin(x3)*(sin(x3) - 81*(x2 + 1/10)^2 + x1^2 + 53/50) + x2^2*sin(x2*x3)^2 - x2^2*cos(x2*x3)*(cos(x2*x3) - 3*x1 + 1/2) + 400]

Q2)C
Choosing
`epsilon = 10^-6`

Eta	Starting Point	Result	#Iteration
0.0001	(1,1,1)	(0.498145,-0.199606,-0.528826)	17 042
0.0001	(0,0,0)	(0.500000,0.000000,-0.523599)	17 075
0.0000001	(2,-3,6)	(0. 500000,0. 000000, -0.523599	1 745 023

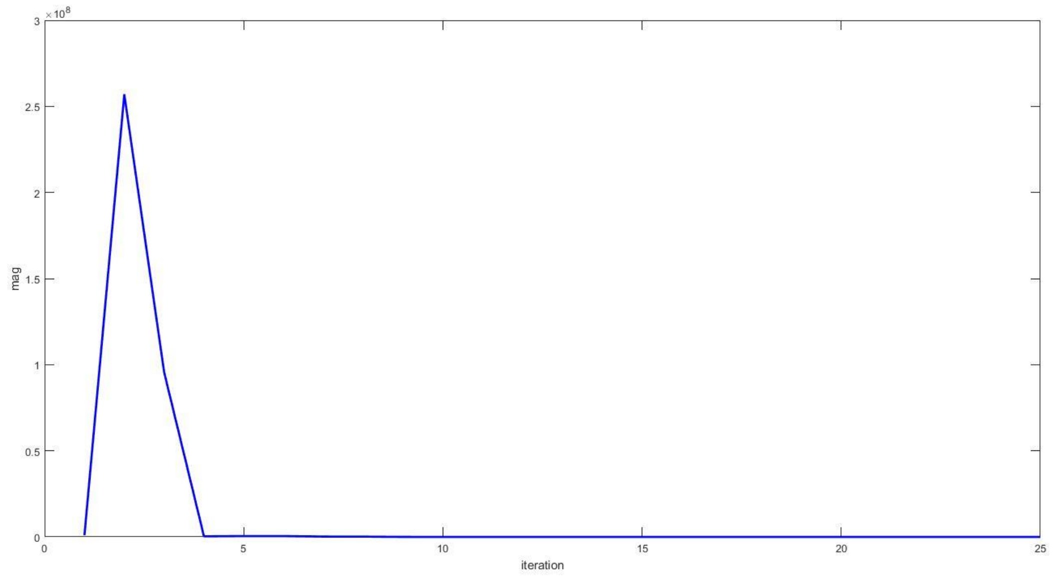
Point(1,1,1)



Q2)D

Starting Point	Result	#Iteration
(1,1,1)	(0.500000,0.000000,-0.523599)	13
(0,0,0)	(0.500000,0.000000,-0.523599)	5
(2,-3,6)	(0.500000,0.000000,-0.523599)	25

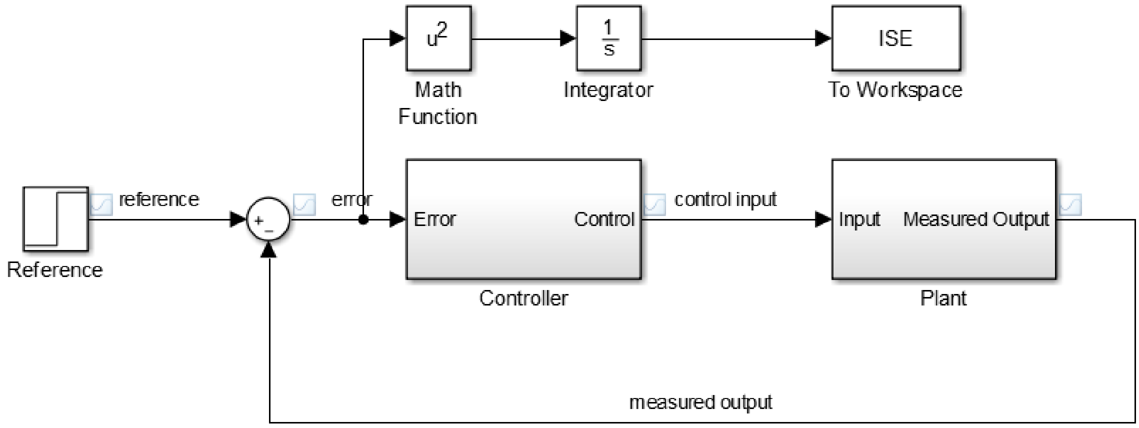
Point (2,-3,6) Plot

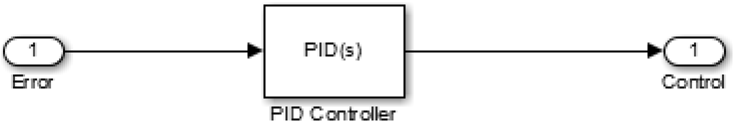


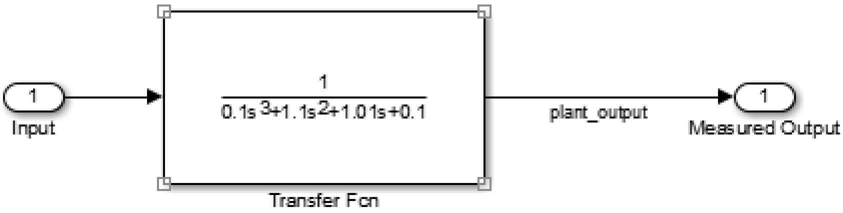
Appendix

MATLAB Code

Q1)








```
min=999999999999999;
n=1;
ma=0;
for Kp = 5:10
    for Ki = 0.1:0.2:0.6
        for Kd =20:25
```

```
            sim('Q1a');
            [w,h]=size(ISE);
            ISE_1=ISE(w,h);
            y(n)=n;
            x(n)=ISE_1;
            if(min==0)
                min=ISE_1;

            elseif(ISE_1 < min)
                min= ISE_1;
                kp=Kp;
                ki=Ki;
                kd=Kd;
              n_best=n;
            end
        n=n+1;
        end

    end
end
fprintf ('kp optimal value is %f/n',kp);
fprintf ('ki optimal value is %f/n',ki);
fprintf ('kd optimal value is %f/n',kd);
fprintf ('Total number of iteration %f',n);
fprintf ('best iteration is  %f/n',n_best);


figure;plot(x(:),y(:));
    xlabel('iteration');
    ylabel('ISE');
```

Q2)a

```
syms x1 x2 x3

g1=3*x1-cos(x2*x3)-0.5;
g2=x1^2-81*((x2+0.1)^2)+sin(x3)+1.06;
g3=exp(-x1*x2)+20*x3+((10*pi-3)/3);
f=(0.5*(g1^2))+(0.5*(g2^2))+(0.5*(g3^2));
f_GradientMatrix=gradient(f,[x1 x2 x3]);
```

Q2)b

```
f_HessianMatrix=hessian(f,[x1 x2 x3])
```

Q2)c

```
function [a] = der_WRTo_x1 (z)
x1=z(1,1);
x2=z(2,1);
x3=z(3,1);
a=9*x1 - 3*cos(x2*x3) + 2*x1*(sin(x3) - 81*(x2 + 1/10)^2 + x1^2 + 53/50) -
x2*exp(-x1*x2)*(20*x3 + exp(-x1*x2) + 5332248173269055/562949953421312) - 3/2;
end

function [b] = der_WRTo_x2 (z)
```

```
x1=z(1,1);
x2=z(2,1);
x3=z(3,1);
b= -(162*x2 + 81/5)*(sin(x3) - 81*(x2 + 1/10)^2 + x1^2 + 53/50) - x1*exp(-
x1*x2)*(20*x3 + exp(-x1*x2) + 5332248173269055/562949953421312) -
x3*sin(x2*x3)*(cos(x2*x3) - 3*x1 + 1/2);
end

function [c] = der_WRTo_x3 (z)
x1=z(1,1);
x2=z(2,1);
x3=z(3,1);
c=400*x3 + 20*exp(-x1*x2) + cos(x3)*(sin(x3) - 81*(x2 + 1/10)^2 + x1^2 + 53/50)
- x2*sin(x2*x3)*(cos(x2*x3) - 3*x1 + 1/2) + 2666124086634527 5/140737488355328;
end

%conventioal gradient descent
Eta = 0.001 ;
epsilon = 10^-6 ;
x0 =[0;0;0];%intial point
% g1=@(x1,x2,x3) 3*x1-cos(x2*x3)-0.5;
% g2=@(x1,x2,x3) x1^2-81*((x2+0.1)^2)+sin(x3)+1.06;
% g3=@(x1,x2,x3) exp(-x1*x2)+20*x3+((10*pi-3)/3);
% f=@(g1,g2,g3)(0.5*(g1^2))+(0.5*(g2^2))+(0.5*(g3^2));
a=der_WRTo_x1(x0);
b=der_WRTo_x2(x0);
c=der_WRTo_x3(x0);
nabla=[a;b;c];
mag=sqrt((nabla(1,1)^2)+(nabla(2,1)^2)+(nabla(3,1)^2) );
n=1;

while (mag>epsilon)
    a=der_WRTo_x1(x0);
    b=der_WRTo_x2(x0);
    c=der_WRTo_x3(x0);
    nabla=[a;b;c];
    mag=sqrt((nabla(1,1)^2)+(nabla(2,1)^2)+(nabla(3,1)^2) );
    y(n)=mag;
    x(n)=n;

    fprintf('itteration %d\t',n);
    fprintf('mag %f\n',mag);
    x0=x0-Eta*nabla ;
    fprintf('x1 is%f\n ',x0(1,1) );
    fprintf('x2 is%f\n ',x0(2,1) );
    fprintf('x3 is%f\n ',x0(3,1) );


    n = n+1 ;


end
plot(x(:),y(:), '-b', 'LineWidth', 2);
xlabel('iteration');
ylabel('mag');
```

Q2)D

```
function [H] = Compute_hessian (z)
x1=z(1,1);
x2=z(2,1);
```

```
x3=z(3,1);
X11= 2*sin(x3) - 162*(x2 + 1/10)^2 + 6*x1^2 + x2^2*exp(-2*x1*x2) + x2^2*exp(-
x1*x2)*(20*x3 + exp(-x1*x2) + 5332248173269055/562949953421312) + 278/25;

X12= 3*x3*sin(x2*x3) - 2*x1*(162*x2 + 81/5) - exp(-x1*x2)*(20*x3 + exp(-x1*x2) +
5332248173269055/562949953421312) + x1*x2*exp(-2*x1*x2) + x1*x2*exp(-
x1*x2)*(20*x3 + exp(-x1*x2) + 5332248173269055/562949953421312);

X13= 3*x2*sin(x2*x3) - 20*x2*exp(-x1*x2) + 2*x1*cos(x3);

X21= 3*x3*sin(x2*x3) - 2*x1*(162*x2 + 81/5) - exp(-x1*x2)*(20*x3 + exp(-x1*x2) +
5332248173269055/562949953421312) + x1*x2*exp(-2*x1*x2) + x1*x2*exp(-
x1*x2)*(20*x3 + exp(-x1*x2) + 5332248173269055/562949953421312);

X22= 13122*(x2 + 1/10)^2 - 162*sin(x3) + x3^2*sin(x2*x3)^2 + (162*x2 + 81/5)^2 -
162*x1^2 + x1^2*exp(-2*x1*x2) + x1^2*exp(-x1*x2)*(20*x3 + exp(-x1*x2) +
5332248173269055/562949953421312) - x3^2*cos(x2*x3)*(cos(x2*x3) - 3*x1 + 1/2) -
4293/25;

X23= x2*x3*sin(x2*x3)^2 - cos(x3)*(162*x2 + 81/5) - 20*x1*exp(-x1*x2) -
sin(x2*x3)*(cos(x2*x3) - 3*x1 + 1/2) - x2*x3*cos(x2*x3)*(cos(x2*x3) - 3*x1 +
1/2);

X31= 3*x2*sin(x2*x3) - 20*x2*exp(-x1*x2) + 2*x1*cos(x3);

X32= x2*x3*sin(x2*x3)^2 - cos(x3)*(162*x2 + 81/5) - 20*x1*exp(-x1*x2) -
sin(x2*x3)*(cos(x2*x3) - 3*x1 + 1/2) - x2*x3*cos(x2*x3)*(cos(x2*x3) - 3*x1 +
1/2);

X33= cos(x3)^2 - sin(x3)*(sin(x3) - 81*(x2 + 1/10)^2 + x1^2 + 53/50) +
x2^2*sin(x2*x3)^2 - x2^2*cos(x2*x3)*(cos(x2*x3) - 3*x1 + 1/2) + 400 ;


H=[X11 X12 X13;X21 X22 X23; X31 X32 X33];
end

%Newton Raphson Gradient Descent


x0 =[2;-3;6];%starting Point
a=der_WRTo_x1(x0);
b=der_WRTo_x2(x0);
c=der_WRTo_x3(x0);
nabla=[a;b;c];
n=1;
VEC=[];
epsilon = 10^-6 ;
mag=sqrt((nabla(1,1)^2)+(nabla(2,1)^2)+(nabla(3,1)^2) );
while (mag>epsilon)
    a=der_WRTo_x1(x0);
    b=der_WRTo_x2(x0);
    c=der_WRTo_x3(x0);
    nabla=[a;b;c];
    mag=sqrt((nabla(1,1)^2)+(nabla(2,1)^2)+(nabla(3,1)^2) );
    VEC = [VEC;mag];
    y(n)=mag;
    x(n)=n;
    fprintf('Number Of Itteration is:  %d\t',n);
    fprintf('magnitude is: %f\n',mag);

    x0=x0-inv(Compute_hessian(x0))*nabla ;
```

```
    fprintf('x1 is%f\n ',x0(1,1) );
    fprintf('x2 is%f\n ',x0(2,1) );
    fprintf('x3 is%f\n ',x0(3,1) );


    grid

    n = n+1 ;
end
    plot(x(:),y(:), '-b', 'LineWidth', 2);
    xlabel('iteration');
    ylabel('mag');
```

Q2)E

```
%Steepest Descent
% g1=3*x1-cos(x2*x3)-0.5;
% g2=x1^2-81*((x2+0.1)^2)+sin(x3)+1.06;
% g3=exp(-x1*x2)+20*x3+((10*pi-3)/3);
syms x1 x2 x3 eta
f=(0.5*((3*x1-cos(x2*x3)-0.5)^2))+(0.5*((x1^2-
81*((x2+0.1)^2)+sin(x3)+1.06)^2))+(0.5*((exp(-x1*x2)+20*x3+((10*pi-3)/3))^2));


x0=[1;1;1];

a=der_WRTo_x1(x0);
b=der_WRTo_x2(x0);
c=der_WRTo_x1(x0);

nabla=[a;b;c];
x1=x0-eta.*nabla;
fi=@fun_eta;
options = optimoptions(@fminunc,'Display','iter','Algorithm','quasi-newton');
fminunc(fi,0,options)
function [f2]=fun_eta(x0)


f=(0.5*((3*(eta-x0(1,1))-cos(x2*x3)-0.5)^2))+(0.5*((x1^2-
81*((x2+0.1)^2)+sin(x3)+1.06)^2))+(0.5*((exp(-x1*x2)+20*x3+((10*pi-3)/3))^2));
f2=subs(f,[x1,x2,x3],[sym('eta')-x0(1,1),sym('eta')-x0(3,1),sym('eta')-
x0(3,1)]);
end
```