Course Project on

PID controller for automatic voltage regulator using different controller designing techniques

By

Sagar Patel

(Student Id No. 0672037)

Course Name : Control Engineering Concepts – ENGI-5111-FA

Course Instructor: Dr. Xiaoping Liu

PID controller for automatic voltage regulator using different controller designing techniques

Abstract:

The present work presents the different designing methods or techniques for designing a controller with the help of the paper "PID controller for automatic voltage regulator using teaching–learning based optimization technique" by Shamik Chatterjee & V. Mukherjee, Department of Electrical Engineering, Indian School of Mines, Dhanbad, Jharkhand, India. The paper basically deals with teaching–learning based optimization (TLBO) calculation as an improvement procedure in the territory of tuning of the traditional controller introduced in programmed or automatic voltage controller(AVR). Using the transfer functions and other relevant values of the controller in this paper is used to design different controllers such as Pole Placement, Pole Placement with integral control, Observer Based Controller, Observer Based with Integral Control, Linear Quadratic Regulator(LQR) Controller and Linear Quadratic Regulator(LQR) with Integral Control. The Controller are designed using different poles using different overshoot and settling time. Therefore, with the help of the data we have plotted step response for each closed loop controller with the best suitable parameters.

Introduction:

A Programmed or Automatic Voltage Controller (AVR) is a gadget that is intended to consequently control, change or keep up a consistent voltage level of a synchronous generator. The fundamental capacity of the AVR is to keep up the voltage of an alternator at a clear level. Along these lines, the security of the power framework would be genuinely influenced by the strength of the AVR framework.

In power framework, one of the primary control issue is to give consistency and soundness of the nominal voltage level in an electrical power network having every single associated gear, intended for a certain voltage level. There might be reduction in the execution of these types of gear and drop in their anticipation, if the nominal voltage level goes astray from the appraised one. Another purpose behind this control is that the genuine line loss relies on upon the real and reactive power flow. Variety in terminal voltage changes the responsive power flow with a major margin. The AVR, which is utilized to keep up the terminal voltage of a synchronous generator at a predetermined level, is executed in power framework to overcome these control issues. With the variety of the exciter voltage of the alternator, the AVR keeps up the consistency of the terminal voltage. Stable and quick reaction of the controller is hard to accomplish due to the high inductance of the alternator field windings and load variety. Consequently, change of the AVR execution is critical. Insulation breakdown may happen in various parts of the control framework because of high voltage which may harm the equipment. In this way, appropriate controlling system is required for the AVR framework to perform appropriately.

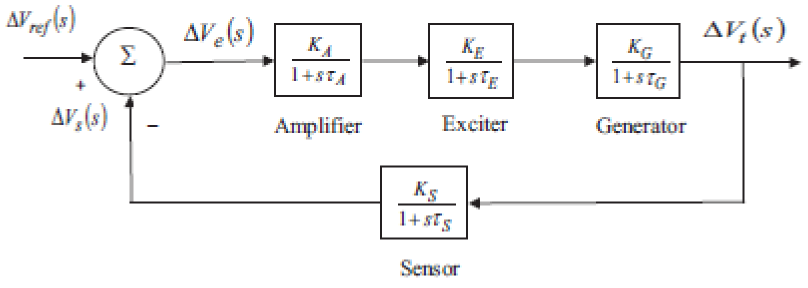

Fig. 1. Transfer Function Model of AVR framework

Modeling of an AVR system

A standout amongst the most critical components to enhance control framework security and nature of electrical power, is the excitation control of the synchronous generator. To hold the size of the terminal voltage of a synchronous generator inside a predetermined point of confinement, the AVR is required. The AVR, involves four principle segments (to be specific, amplifier, exciter, generator and sensor), assumes an essential part in the power framework. Voltage sensor's part is to detect the terminal voltage of the alternator ceaselessly. This flag, in the wake of being amended or smoothened, is contrasted with a reference signal in the comparator. The mistake voltage got from the comparator is utilized in the wake of being opened to control the field twisting of the alternator by method for the exciter. The exchange capacities of the depicted four parts of the AVR framework are thought to be straight for the scientific displaying of the framework in the show work. In Fig. 1, the exchange work piece outline of the AVR framework is appeared. An intensifier's exchange capacity is displayed by a gain and a time constant, as given in

$$G_{Amplifier}(s) = \frac{K_A}{1 + s\tau_A}$$

where K_A and τ_A are the gain and the time constant of the amplifier system. The value of K_A is in the range of 10–40 while the amplifier time constant τ_A ranges from 0.02 s to 0.1 s.

Like amplifier, the transfer function of a modern exciter may be represented by a gain and a time constant, as presented in

$$G_{Exciter}(s) = \frac{K_E}{1 + s\tau_E}$$

where K_E and τ_E are the gain and the time constant of the exciter system. The value of K_E is in the range of 1–10 and τ_E ranges from 0.4 s to 1.0 s. Similarly, the below equation represents the transfer function relating the generator terminal voltage to its field voltage.

$$G_{Generator}(s) = \frac{K_G}{1 + s\tau_G}$$

where K_G and τ_G are the gain and the time constant of the generator model. The value of K_G is in the range of 0.7–1.0 and τ_G ranges from 1.0 s to 2.0 s. The transfer function of the sensor is modeled by a gain and a time constant, as presented in

$$G_{Sensor}(s) = \frac{K_S}{1 + s\tau_S}$$

where K_S and τ_S are the gain and the time constant of the sensor model. The value of K_S is in the range of 0.9–1.1 and τ_S ranges from0.01s to 0.06s. Using the parameters, the closed loop transfer function of the AVR system (G_{AVR}) becomes as follows:

$$G_{AVR}(s) = \frac{\Delta V_t(s)}{\Delta V_{ref}(s)} = \frac{0.1s + 10}{0.0004s^4 + 0.045s^3 + 0.555s^2 + 1.51s + 11}$$
$$= \frac{250(s + 100)}{(s + 98.82)(s + 12.63)(s^2 + 1.057s + 22.04)}$$

From the above equation, it may be noted that there is one zero and two real poles at Z = -100 and at S1 = -98.82 and S2 = -12.63, respectively, and the two complex poles are at S3,4 = -0.53 ± j4.66. Approximation of GAVR may be done by cancelling the zero at -100 with the pole at -98.82 to obtain G_{AVR}, as:

$$\tilde{G}_{AVR}(s) = \frac{250}{(s + 12.63)(s^2 + 1.057s + 22.04)}$$

With the help of the above said transfer function we have calculated the values of the different poles with the help of the general form of the transfer function which is where we calculate the different ω_o and ζ

$$G_{yd} = \frac{b_1 s^2 + b_2 s}{(s + \alpha\omega_0)(s^2 + 2\zeta\omega_0 s + \omega_0^2)}$$

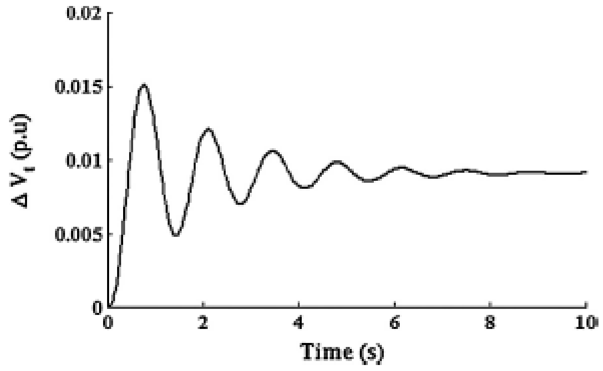

Fig. Voltage response of AVR system without controller

The voltage response of the AVR system without any controller is shown in above figure.

In power system (where the operating voltage is in the order of hundreds of kV), this type of response is completely unwanted and is not permissible. For this purpose, a controller is needed to be installed in the AVR system. It is known that the PID controller is playing a vital role in the industries due its simplicity and robustness. Thus, in the studied AVR system, the PID controller may be implemented for controlling the deviation in terminal voltage. In the next section, the structure of the studied PID controller is discussed.

Structure of PID controller

The PID controller is the most adequate input controller utilized as a part of the businesses. For more than 50 years, it is by and large, effectively, utilized by the business houses. It is effectively reasonable and in addition powerful controller that can offer brilliant control execution regardless of the fluctuated dynamic qualities of the procedure plant. PID controller has time controller picks up i.e. proportional, integral and derivative gain. A relative controller can decrease the estimation of T_R yet can't wipe out E_{SS}. The framework can get to be distinctly unsteady on making the relative pick up too high, though a little estimation of this pick up may bring about a little yield reaction to a substantial info blunder and a less responsive or less touchy controller. The impact of an indispensable controller is to dispose of E_{SS} however the transient reaction deteriorates by it. The estimation of Mp will be high, if necessary pick up is made higher and on putting it low, it might bring about a languid framework reaction. A subsidiary controller may expand the steadiness of the framework, lessen the estimation of Mp and, henceforth, there will be a change in transient reaction. The procedure may get to be distinctly unsteady on making the subordinate pick up, adequately, expansive. For mechanical control handle, the traditional settled pick up PID controller is broadly utilized. The three parameters of this controller viz. proportional gain (Kp), integral gain (Ki) and derivative gain (K_D) are required to be outlined properly. Considering the experience of the planner and plant conduct, the tuning of these parameters is finished by experimentation strategy. In Fig. 3, the PID controller for a closed loop framework is appeared. In Laplace mode, the transfer function of PID controller can be expressed by.

$$T_F = K_P + \frac{K_I}{s} + sK_D$$

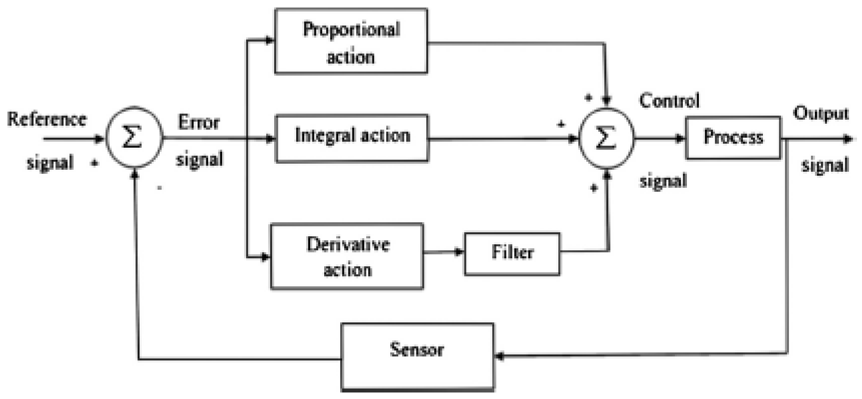

Fig. Block Diagram of the studied PID Controller

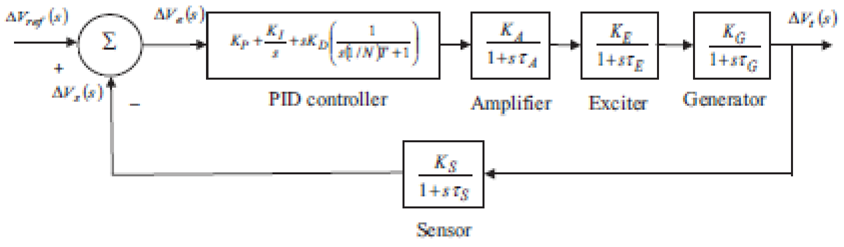

Fig. Transfer function block diagram with PID Controller

Controller Design:

Using the transfer function from the above said paper, we have designed controller using 6 different types of methods, which is as follows:

1. Pole Placement
2. Pole Placement with Integral Control
3. Observer-based Control
4. Observer-based Controller with Integral Control
5. Linear Quadratic Regulator (LQR) Controller
6. Linear Quadratic Regulator (LQR) with Integral Control

We have designed the different control with the help of MATLAB. The main code through which all the controller function are simulated is as follows:

```
clear all
close all
clc

%Pole Placement
b = [250];
a = [1 13.687 35.38 278.36];
[A,B,C,D] = tf2ss(b,a)
sys1=PolePlacement(A,B,C,D)
figure
step(sys1)

%Pole Placement with Integral Control
b = [250];
a = [1 13.687 35.38 278.36];
[A,B,C,D] = tf2ss(b,a)
sys2=PolePlacementIntegral(A,B,C,D)
figure
step(sys2)

%Observer-based Controller
b = [250];
a = [1 13.687 35.38 278.36];
[A,B,C,D] = tf2ss(b,a)
sys3=observer(A,B,C,D)
figure
step(sys3)


%Observer-based using Integral Control
b = [250];
a = [1 13.687 35.38 278.36];
[A,B,C,D] = tf2ss(b,a)
sys4=ObserverIntegral(A,B,C,D)
figure
step(sys4)

%Linear Quadratic Regulator(LQR) Control
b = [250];
a = [1 13.687 35.38 278.36];
[A,B,C,D] = tf2ss(b,a)
sys5=LQR(A,B,C,D)
figure
step(sys5)
```

```
%Linear Quadratic Regulator(LQR) with Integral Control
b = [250];
a = [1 13.687 35.38 278.36];
[A,B,C,D] = tf2ss(b,a)
sys6=LQRIntegral(A,B,C,D)
figure
step(sys6)

%PID Controller Tuning
b = [250];
a = [1 13.687 35.38 278.36];
[A,B,C,D] = tf2ss(b,a)
sys7=PIDcontroller(A,B,C,D)
figure
step(sys7)
```

Based on the main code, each controller is designed when the function is called. Therefore, All the Control Designs have their designated function.

Each system is designed with different overshoot and settling time. Therefore, the step response of each controller is plotted using the MATLAB.

1. **Pole Placement Method:**

Pole placement method is a controller design method in which you determine the places of the closed loop system poles on the complex plane by setting a controller gain KK. Poles describe the behaviour of linear dynamical systems. Through use of feedback you are attempting to change that behaviour in a way that is more favorable.

Therefore, in our system using the transfer function we have simulated the controller using the pole placement method.

We have defined the function for pole placement in MATLAB. The following is the code for pole placement method:

```
function sys1=PolePlacement(A,B,C,D)
system = ss(A,B,C,D);
rank(ctrb(system))

ts=0.08
os=0.06
zeta=sqrt((log(os))^2/(pi^2+(log(os))^2));
omega=4/(zeta*ts);
alpha=5

p1=-zeta*omega+1i*omega*sqrt(1-zeta^2);
p2=-zeta*omega-1i*omega*sqrt(1-zeta^2);
p3=-alpha*zeta*omega;
p=[p1 p2 p3]

K = place(A,B,p)
system1 = ss((A-B*K,B,C,D);
d = dcgain(system1);
sys1= system1*(1/d);
```

As soon as the function is simulated the step response of the closed loop control for pole placement is plotted. The plot of the step response for the controller using the pole placement method is as follows:

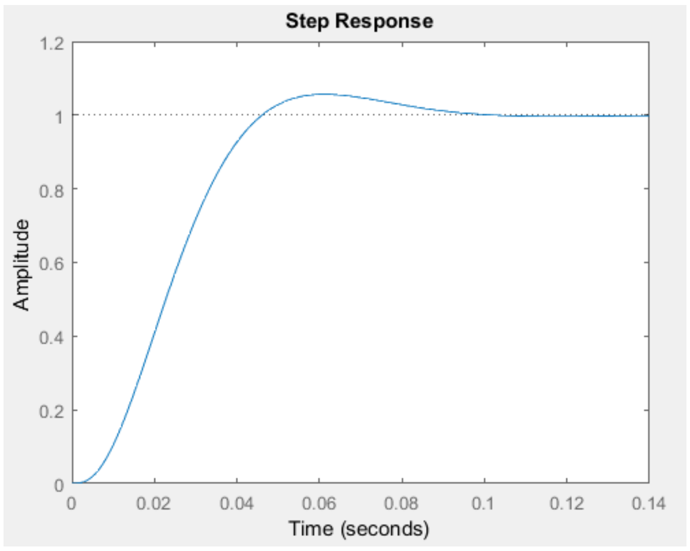

Fig. Step Response for pole placement method

The settling time is 0.08 secs and the overshoot of this system is 6%. Here, we can see the controller has a fast response w.r.t. time and can be tuned with the help of overshoot and settling time.

2. Pole Placement with integral control

Using the pole placement with integral control, following is the code for the controller which uses the pole placement with integral control technique.

```
function sys2=PolePlacementIntegral(A,B,C,D)
Ahat = [A zeros(3,1); -C 0]
Bhat = [B; 0]
C=[0 C];
system = ss(Ahat,Bhat,C,D);
rank(ctrb(system))

ts=0.08
os=0.06
zeta=sqrt((log(os))^2/(pi^2+(log(os))^2));
omega=4/(zeta*ts);
alpha=5

p1=-zeta*omega+1i*omega*sqrt(1-zeta^2);
p2=-zeta*omega-1i*omega*sqrt(1-zeta^2);
p3=-alpha*zeta*omega;
p=[p1 p2 p3 10*real(p1)];

K = place(Ahat,Bhat,p)
system1 = ss((Ahat-Bhat*K),Bhat,C,D);
d = dcgain(system1);
sys2= system1*(1/d);
```

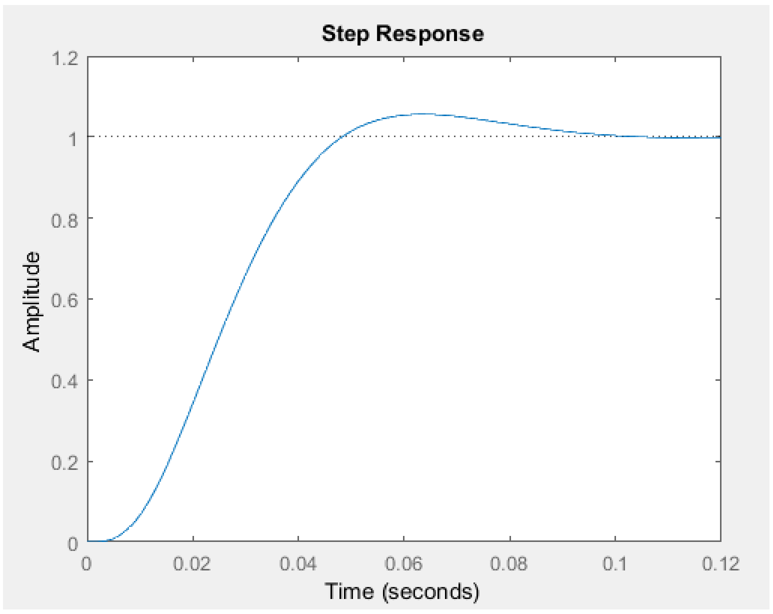

Fig. Step Response of Pole Placement with Integral Control

The overshoot for the above system is 6% and the settling time is 0.08. Here, we can see the controller has a fast response w.r.t. time and can be tuned with the help of overshoot and settling time.

3. Observer Based Controller

In control theory, a state observer is a system that provides an estimate of the internal state of a given real system, from measurements of the input and output of the real system. It is typically computer-implemented, and provides the basis of many practical applications.

Knowing the system state is necessary to solve many control theory problems; for example, stabilizing a system using state feedback. In most practical cases, the physical state of the system cannot be determined by direct observation.

The Observer Based controller is also designed using different settling time and overshoot. Following is the code for the observer based controller:

```
function sys3=observer(A,B,C,D)
system = ss(A,B,C,D);
rank(obsv(system))

ts=0.1
os=0.02
zeta=sqrt((log(os))^2/(pi^2+(log(os))^2));
omega=4/(zeta*ts);
alpha=5;

p1=-zeta*omega+1i*omega*sqrt(1-zeta^2);
p2=-zeta*omega-1i*omega*sqrt(1-zeta^2);
p3=-alpha*zeta*omega;
p=[p1 p2 p3]

L1= -40;
L2= -60;
L3= -80;
P=[L1 L2 L3]
```

```
L = place(A',C',P)'
K = place(A,B,p)
aa= A-L*C-B*K;
bb= L;
cc= K;
dd= D;
[num,den]=ss2tf(aa,bb,cc,dd);
observer_tf=tf(num,den)
sys3=feedback(system*observer_tf,1);
```

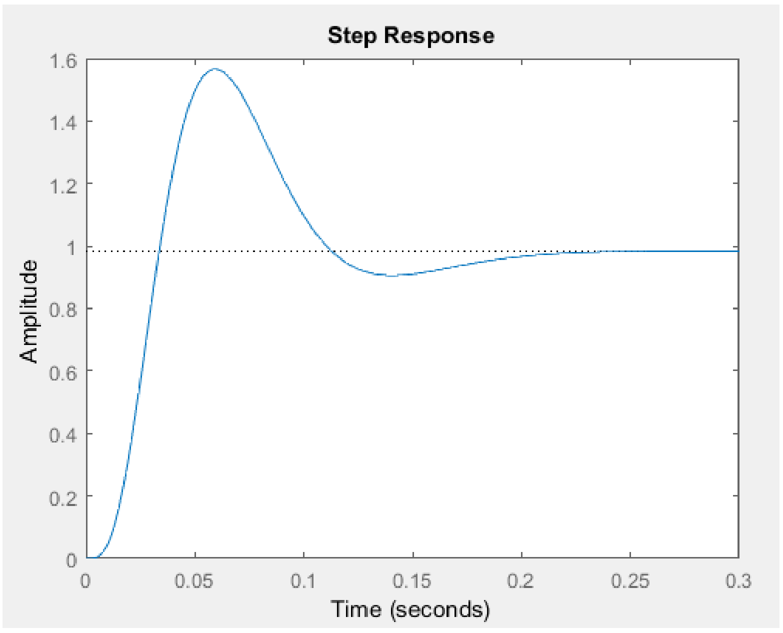

Fig. Step Response for Observer Based Controller

Here, the settling time of the observer based controller is 0.1sec and overshoot is 2%. Here, we can see the controller has a slow response w.r.t. time and can be tuned with the help of overshoot and settling time. As the overshoot of the system is greater the slower will be the response and the response will oscillate.

4. Observer Based with Integral Control

Using the observer based with integral control, following is the code for the controller which uses the observer based controller with integral control technique.

```
function sys4=ObserverIntegral(A,B,C,D)
Ahat = [A zeros(3,1); -C 0]
Bhat = [B; 0]
C=[0 C];

system = ss(Ahat,Bhat,C,D);
rank(obsv(system))

ts=0.1
os=0.02
zeta=sqrt((log(os))^2/(pi^2+(log(os))^2));
omega=4/(zeta*ts);
```

```
alpha=5;


p1=-zeta*omega+1i*omega*sqrt(1-zeta^2);
p2=-zeta*omega-1i*omega*sqrt(1-zeta^2);
p3=-alpha*zeta*omega;
p=[p1 p2 p3 10*real(p1)];


L1= -10;
L2= -20;
L3= -30;
L4= -40;
P = [L1 L2 L3 L4]


L = place(Ahat',C',P)'
K = place(Ahat,Bhat,p)
aa = Ahat-L*C-Bhat*K;
bb=L;
cc=K;
dd=D;
[num,den]=ss2tf(aa,bb,cc,dd);
observer_tf=tf(num,den)
sys4=feedback(system*observer_tf,1);
```

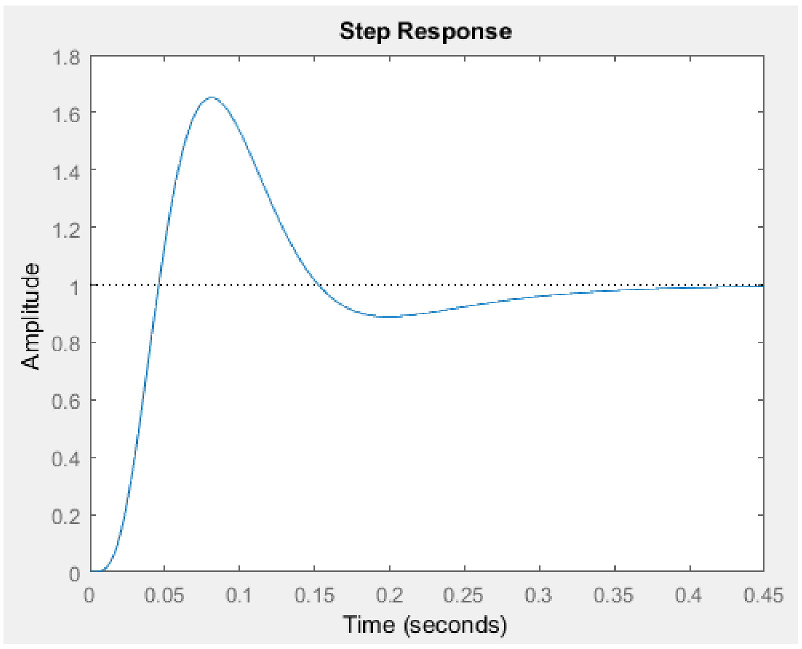

Fig. Step Response of Observer Based with Integral Control

Here, the settling time of the observer based controller is 0.1sec and overshoot is 2%. Here, we can see the controller has a slow response w.r.t. time and can be tuned with the help of overshoot and settling time. As the overshoot of the system is greater the slower will be the response and the response will oscillate.

5. Linear Quadratic Regulator (LQR)

The settings of a controller governing either a machine or process are found by using a mathematical algorithm that minimizes a cost function with weighting factors supplied by a engineer. The cost function is often defined as a sum of the deviations of key measurements, desired altitude or process temperature, from their desired values. The algorithm thus finds those controller settings that minimize undesired deviations. The magnitude of the control action itself may also be included in the cost function.

The LQR algorithm reduces the amount of work done by the control systems engineer to optimize the controller.

LQR is always calculated using the following equation:

PA + A’P – PBR^-1B’P + Q = 0

Where **k = -R^-1*B’*P**

The LQR is designed using the following code:

```
function sys5=LQR(A,B,C,D)
system=ss(A,B,C,D);

Q = [eye(3)]
R = [0.001]

[K,P,e] = lqr(A,B,Q,R)

sys5 = ss(A-B*K,B,C,D);
n=dcgain(sys5);
sys5=sys5*(1/n);
```

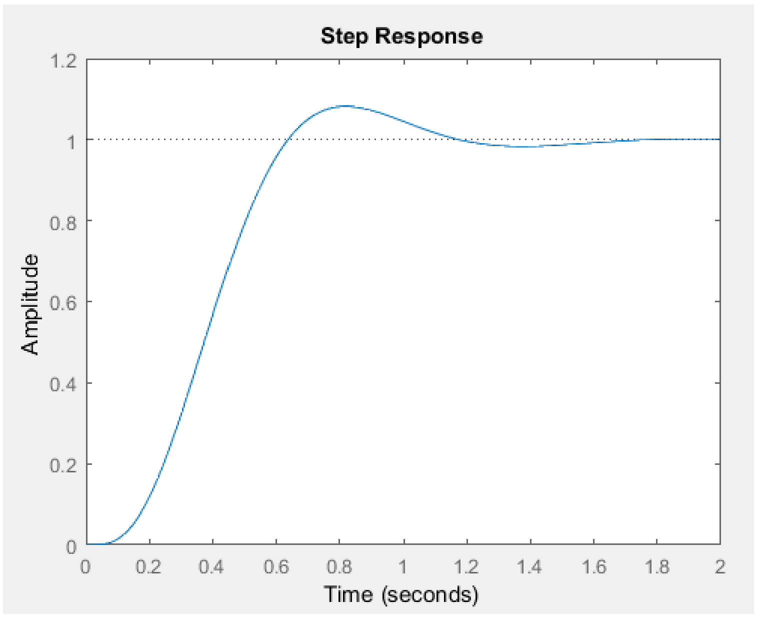

Fig. Step Response of LQR Controller

The step response of the LQR Controller is fast w.r.t time and it can be tuned if we change the value of **R.** The less the value of R, the faster is the step response.

6. LQR with Integral control

LQR with Integral control is always calculated using the following equation:

PA +A'P – PBR^-1B'P + Q = 0

Where **k = -R^-1*B'*P**

```
function sys6=LQRIntegral(A,B,C,D)
Ahat = [A zeros(3,1); -C 0]
Bhat = [B; 0]
C = [0 C]

system=ss(Ahat,Bhat,C,D);

% LQR
Q = [eye(4)]
R = [0.001]

[K,P,e] = lqr(Ahat,Bhat,Q,R)

sys6 = ss(Ahat-Bhat*K,Bhat,C,D);
n=dcgain(sys6);
sys6=sys6*(1/n);
step(sys6);
```

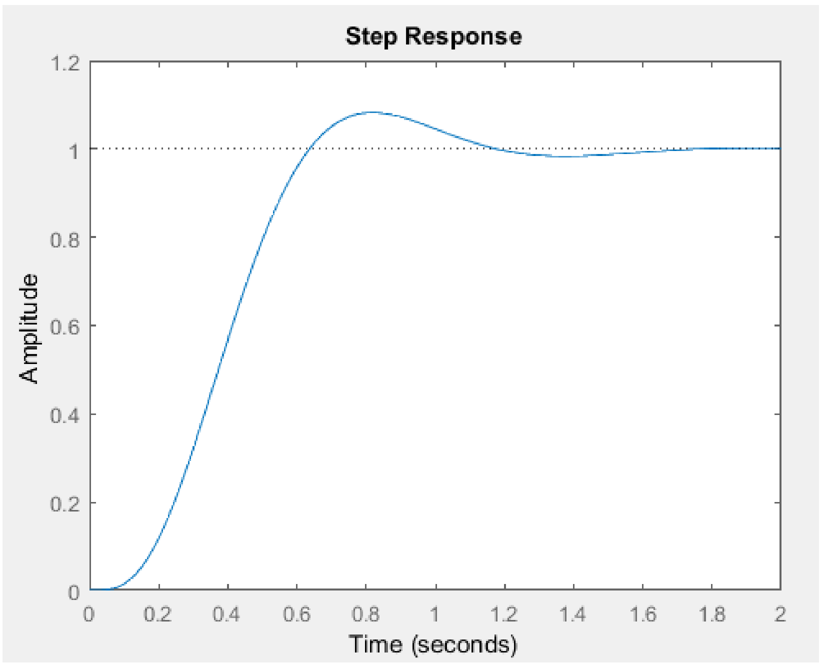

Fig. Step Response of LQR with Integral Control

The step response of the LQR with integral control is almost the same when the value of R is same i.e. it has a faster response w.r.t. time.

PID Tuning:

```
function sys7=PIDcontroller(A,B,C,D)
system=ss(A,B,C,D);
s=tf('s');
Kp = 19.5
Ki=8.81
Kd=10.8
PID = Kp + Ki*(1/s) + Kd*(s)
sys7=feedback(system*PID,1);
```

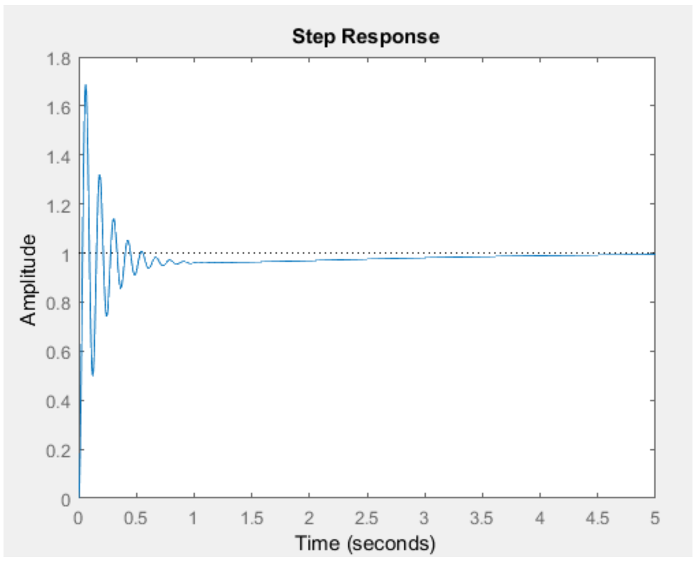

Fig. PID Controller

The PID controller is tuned w.r.t. time and it can be tuned with the help of the overshoot. The higher value of overshoot results in faster Step response but the no. of oscillations will be more and the settling time will be less. If the overshoot is less the controller will take more time to settle and will have less no. of oscillations. Therefore, accordingly it can be tuned.

Conclusion:

In this course project, we have presented the tuning of the PID Controller gains. Also, the different controller technique are used such as Pole Placement, Pole Placement with Integral Control, Observer Based Controller, Observer Based with Integral Control, Linear Quadratic Regulator Controller, Linear Quadratic Regulator with Integral Control are used in order to find the closed loop step response of each controller. All the converters are designed using different overshoot and settling time which will help the controller to work in best condition.