EE 370 L CONTROL SYSTEM LABORATORY

LABORATORY 5: PID CONTROL

DEPARTMENT OF ELECTRICAL AND COMPUTER ENGINEERING
UNIVERSITY OF NEVADA, LAS VEGAS

1. OBJECTIVE

To demonstrate the concept of proportional-integral-differential (PID) control.

2. COMPONENTS & EQUIPMENT

PC with MATLAB and Simulink toolbox installed.

3. BACKGROUND

In applications where a simple gain compensator is inadequate to achieve stability or other performance specifications, a frequency-dependent (dynamic) compensator is needed. One of the earliest such compensators was the PID controller. The PID control does not require a detailed model of the plant, but is designed by continuously calculating an error value $e(t)$ as the difference between a desired *setpoint* (SP) and a measured *process variable* (PV) and applying a correction based on proportional, integral, and derivative terms (denoted P, I, and D respectively), in other words, "tuning", its parameters, while the system is operational.

In practical terms it automatically applies accurate and responsive correction to a control function. An everyday example is the cruise control on a car, where ascending a hill would lower speed if only constant engine power is applied. The controller's PID algorithm restores the measured speed to the desired speed with minimal delay and overshoot, by increasing the power output of the engine.

A PID compensator has the transfer function of the following form:

$$G_c(s) = K_1 + \frac{K_2}{s} + K_3 s = \frac{K_2 + K_1 s + K_3 s^2}{s} \tag{1}$$

As an example, consider the following closed-loop configuration, which has two zeros and a pole at the origin. One zero and one pole can be designed as ideal integrator; the other zero can be designed as ideal derivative compensator

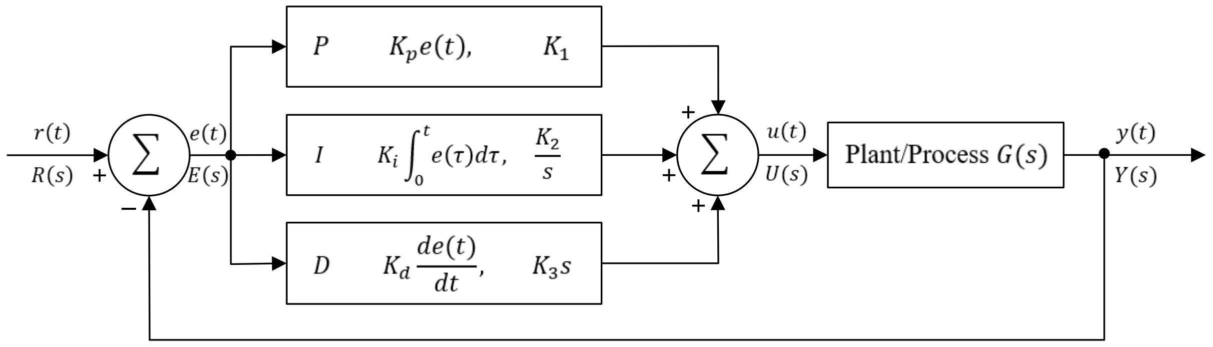


Figure 1. A block diagram of a PID controller in a feedback loop. r(t) is the desired process value or setpoint (SP), and y(t) is the measured process value (PV).

Design Steps:

1. System "tuning" starts with varying K_1 and set $K_2 = K_3 = 0$, while the system is operating. Derive the new system transfer function with K_1 introduced. Tuning K_1 until system has an acceptable step response i.e. drive $e(t)$ toward zero.
2. If a steady-state error exists ($e_{ss0} \neq 0$), K_2 is to be adjusted. Derive the new system transfer function with K_2 introduced. The term K_2/s integrates error $e(t)$ and applies the result to the plant, forcing $e(t) \rightarrow 0$. K_2 is varied until sufficiently small e_{ss0} is achieved.
3. Finally, derive the new system transfer function with K_3 introduced, which is varied to reduce rise time of the output signal. This reduction of rise time occurs since K_3 provides extra "kick" at the start, when $e(t)$ changes the most.

Detail process and example is attached in Appendix.

<u>Note</u>: In some applications, the derivative term in the PID structure causes undesirable phenomena, such as high-frequency noise or oscillations. In such cases, the compensator frequency response can be attenuated at high frequency by introducing additional pole to reduce these stray effects.

4. LAB DELIVERIES

PRELAB:

1. **Study the knowledge of PID controller, briefly introduced in the previous section.**

LAB EXPERIMENTS:

1. **Use MATLAB built-in tool "*pidtool*", and to find Kp, Ki and Kd (if necessary) of the PID controller that meets the following specifications.**

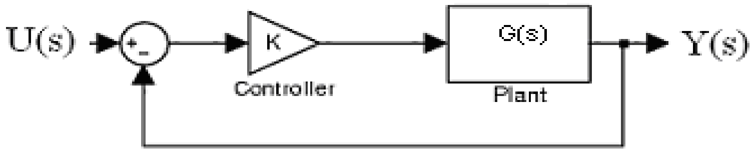


Figure 2. A simple system model with PID compensate

$$G(s) = \frac{1}{s(s+20)}$$

- Percent overshoot: < 5%
- Settling time: < 250 ms
- (Optional) Maximum response to a unit disturbance: < 5×10^{-3}
- You can configure "Plant", "Type" and pull out "Parameters" to meet the system requirement.

2. **Repeat Experiment 1, but use Simulink to build the block diagram and run the simulation.**

3. **Use either "pidtool" or Simulink to design and plot a PID controller response for the following uncompensated transfer function.**

$$H(s) = \frac{1}{s^2 + 12s + 20}$$

- Use both MATLAB programming and Simulink model, respectively.
- Percent overshoot: < 5%
- Settling time: < 250 ms
- (Optional) Maximum response to a unit disturbance: < 5×10^{-3}

4. **Repeat Experiment 3, but write your own MATLAB code. You can use the PID values from Experiment 3.**

POSTLAB REPORT:

Include the following elements in the report document:

Section	Element	
1	**Theory of operation** *Include a brief description of every element and phenomenon that appear during the experiments.*	
2	**Prelab report** 1. Go through the background section of this instruction manual 2. Go through the Appendix and take the screenshots	
3	**Results of the experiments**	
	Experiments	**Experiment Results**
	1	Simulink model and simulation results for Experiment 1.
	2	Simulink model and simulation results for Experiment 2.
	3	"pidtool" screenshot and simulation results for Experiment 3.
	4	MATLAB codes and simulation results for Experiment 4.
4	**Answer the questions**	
	Questions	**Questions**
	1	
5	**Conclusions** *Write down your conclusions, things learned, problems encountered during the lab and how they were solved, etc.*	
6	**Images** *Paste images (e.g. scratches, drafts, screenshots, photos, etc.) in Postlab report document (only .docx, .doc or .pdf format is accepted). If the sizes of images are too large, convert them to jpg/jpeg format first, and then paste them in the document.* **Attachments (If needed)** *Zip your projects. Send through WebCampus as attachments, or provide link to the zip file on Google Drive / Dropbox, etc.*	

5. REFERENCES & ACKNOWLEDGEMENT

1. Norman S. Nise, "Control Systems Engineering", 7th Ed.
2. https://en.wikipedia.org/wiki/PID_controller

I appreciate the help from faculty members and TAs during the composing of this instruction manual. I would also thank students who provide valuable feedback so that we can offer better higher education to the students.

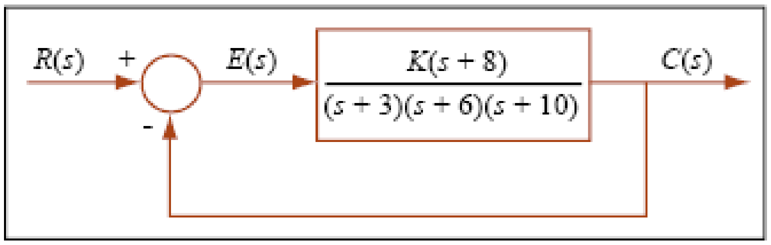
Appendix

Design Steps:

1) Evaluate performance of the uncompensated systems and determine how much improvement in transient parameter is required.
2) Design PD controller to meet the transient response specifications.
3) Simulate the system to see if transient specification requirements are met.
4) Redesign if specifications are not met.
5) Design PI controller to yield the required steady state error specifications.
6) Determine the gains, K1, K2 and K3.
7) Simulate the system to see if all specification requirements are met.
8) Redesign if all the specification parameters are not met.

Design Example:

Given the system shown in figure below:

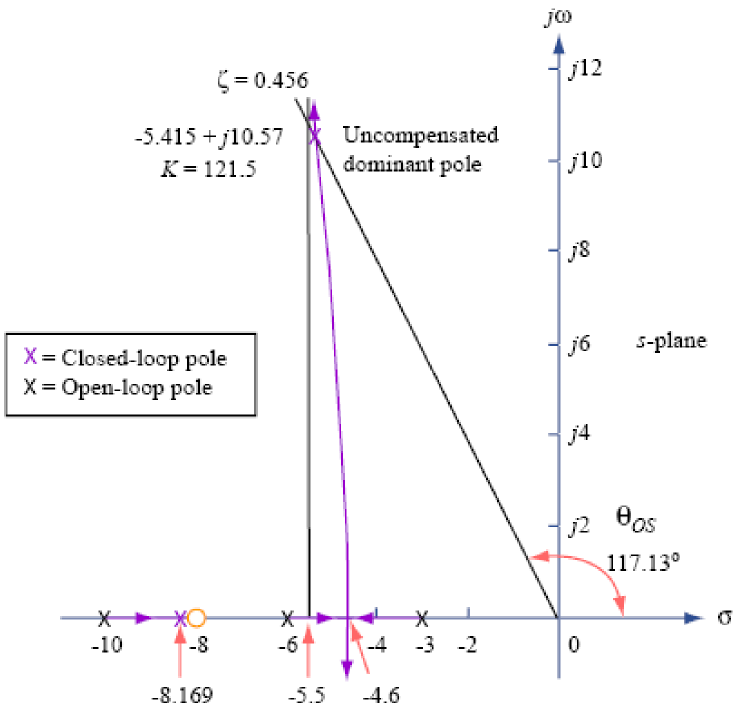


Design the PID controller that meets the following specifications:

1) maintain 20% overshoot
2) reduce peak time to 2/3 of the uncompensated system's peak time
3) eliminate steady-state error

Step1: Uncompensated system performance:

4) 20% overshoot ↔ ζ = 0.456 line crosses the root locus at proportional gain K = 121 achieves overshoot target;
5) Tp = 0.297 sec, must be reduced to ~0.2sec
6) $e(\infty) = 0.156$, must be reduced to 0



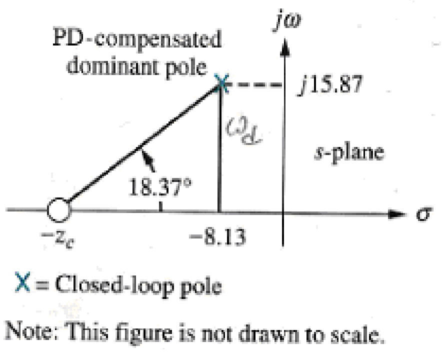
<u>Step 2:</u> Since we want to make the system response faster, we proceed with design of PD controller. Peak required time is 2/3 of the uncompensated time. Therefore, imaginary part of the dominant pole is

$$\omega_d = \frac{\pi}{T_d} \quad \Rightarrow \quad \omega_d = \frac{\pi}{0.2} \approx 15.87 \tag{2}$$

and the real part of the dominant pole is

$$\sigma_d = \frac{\omega_d}{\tan\theta_0 s} = \frac{\omega_d}{\tan 117^{\circ}} \approx -8.13 \tag{3}$$

We find the necessary location of the compensator zero by requiring angular contribution to the dominant pole at $-8.13 \pm j15.87$ to be 180 degree. From root locus, sum of the uncompensated system's poles and zeros angular contribution at the desired dominant pole is -198.37 degree. Therefore, contribution from the compensatory zero is 18.37 degree.



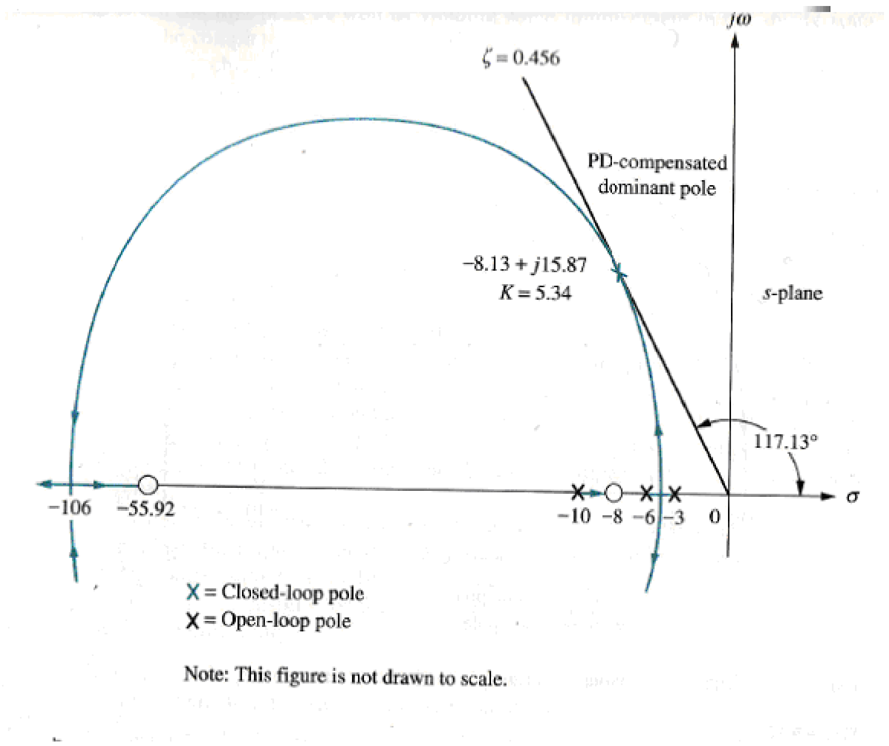
For the figure above,

$$\frac{15.87}{z_c - 8.13} = \tan 18.37^\circ \qquad \Rightarrow \qquad z_c = 55.92$$

Thus, PD controller is

$$G_{pd} = s + 55.92$$

Root locus of the PD compensator is shown below:



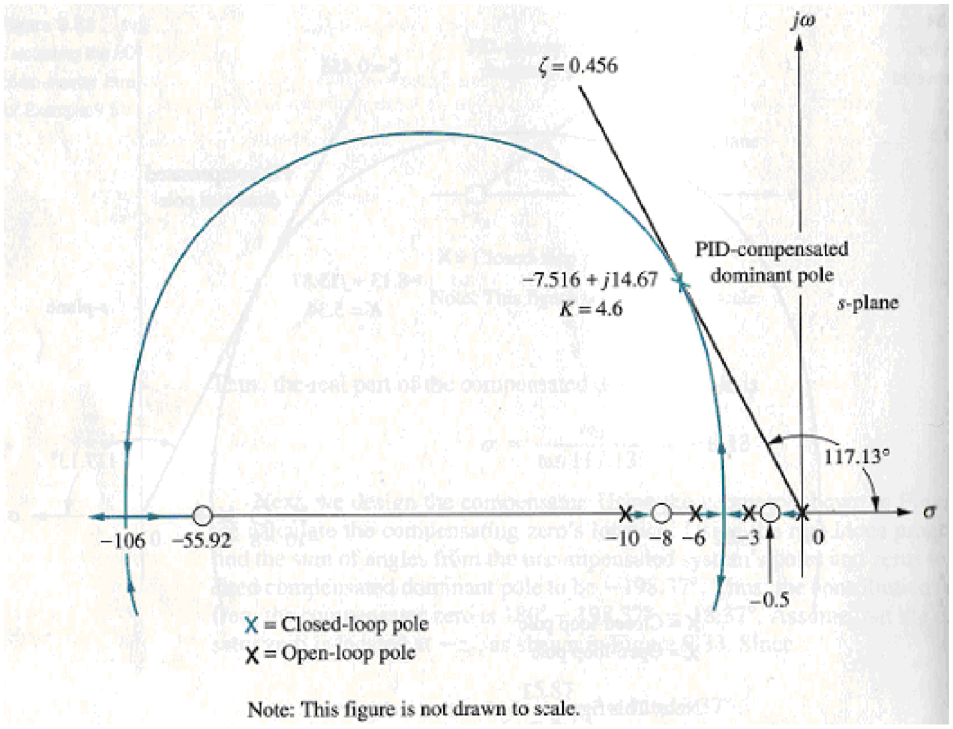
From the root locus, PD gain is 5.34 at the desired point.

Steps 3 & 4: We simulate PD compensated systems to see if the transient specifications are met. From simulation, we find reduction in rise time and improvement in steady state.

Step 5: Design ideal integrator to reduce steady state error. Any integral compensator zero will work as long as the zero is placed closed to unity. Choosing ideal integrator:

$$G_{pi}(s) = \frac{s + 0.5}{s}$$

We sketch root locus of the PID controlled systems as shown below:



From the figure above, we the dominant poles at $-7.516 \pm j14.67$ with associated gain of 4.6 for damping ratio of 0.456.

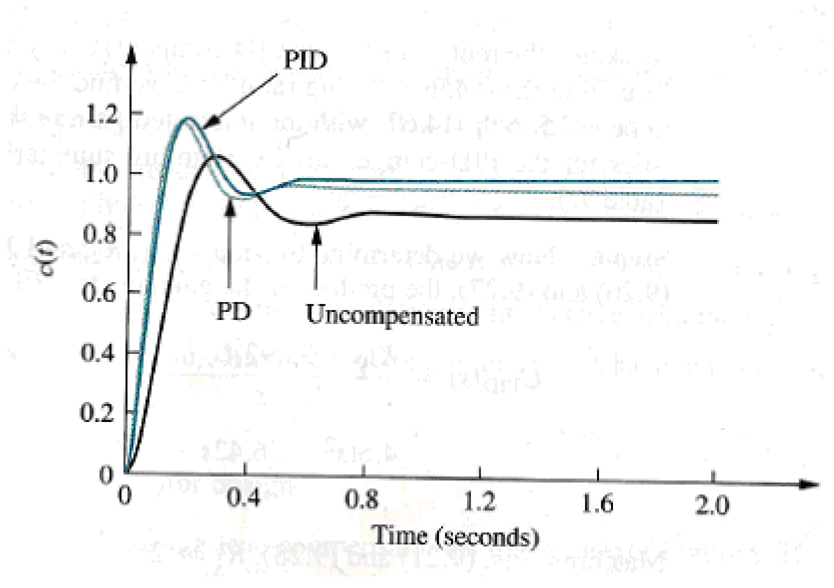
Step 6: Now we determine K_1, K_2 and K_3.

$$K(s + 55.92) \cdot \frac{s + 0.5}{s} = \frac{4.6(s^2 + 56.42s + 27.96)}{s}$$

Matching the form of a PID controller as $(K_1 s + K_2 + K_3 s^2)/s$. Thus,

$$K_1 = 259.5, \quad K_2 = 128.6, \quad K_3 = 4.6$$

<u>Steps 7 & 8:</u> We simulate PID compensated systems to see if the transient specifications are met.



MATLAB Commands

The transfer function of the PID controller looks like the following:

$$K_p + \frac{K_i}{s} + K_d s = \frac{K_d s^2 + K_p s + K_i}{s}$$

- K_p = Proportional gain
- K_i = Integral gain
- K_d = Derivative gain

The characteristics of P, I, and D controllers

CL RESPONSE	RISE TIME	OVERSHOOT	SETTLING TIME	S-S ERROR
K_p	Decrease	Increase	Small Change	Decrease
K_i	Decrease	Increase	Increase	Eliminate
K_d	Small Change	Decrease	Decrease	Small Change

Given the transfer function of a system as given below:

$$H(s) = \frac{1}{S^2 + 10s + 20}$$

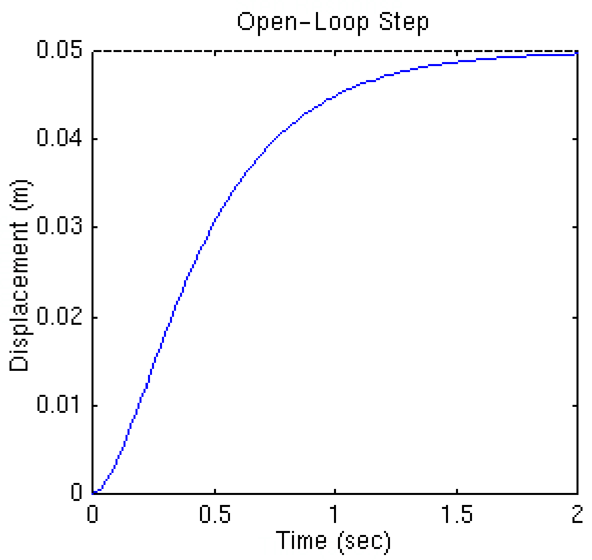
The goal of this problem is to show you how each of Kp, Ki and Kd contribute to obtain

- Fast rise time
- Minimum overshoot
- No steady-state error

Find Open Loop Response:

```
num = 1;
den = [1 10 20];
step(num, den);
```

Running this m-file in the MATLAB command window, you will get plot shown below.



The DC gain of the plant transfer function is 1/20, so 0.05 is the final value of the output to a unit step input. This corresponds to the steady-state error of 0.95, quite large indeed. Furthermore, the rise time is about one second, and the settling time is about 1.5 seconds. Let's design a controller that will reduce the rise time, reduce the settling time, and eliminates the steady-state error.

Proportional control

From the table shown above, we see that the proportional controller (Kp) reduces the rise time, increases the overshoot, and reduces the steady-state error. The closed-loop transfer function of the above system with a proportional controller is:

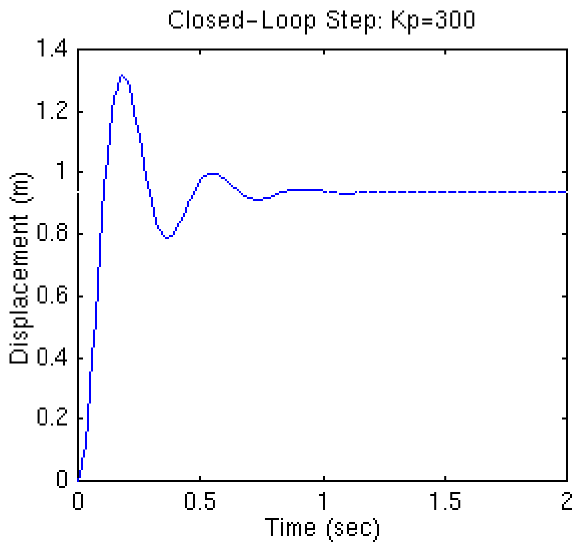
$$H(s) = \frac{K_p}{s^2 + 10s + (20 + K_p)}$$

Let the proportional gain (Kp) equals 300 and change the m-file to the following:

```
Kp=300;
num=[Kp];
den=[1 10 20+Kp];

t=0:0.01:2;
step(num,den,t)
```

Running this m-file in the MATLAB command window, you will get plot shown below.



Note: Alternatively, we could MATLAB function called *cloop* to obtain a closed-loop transfer function directly from the open-loop transfer function (instead of obtaining closed-loop transfer function by hand). The following m-file uses the *cloop* command that should give you the identical plot as the one shown above.

```
num=1;
den=[1 10 20];
Kp=300;

[numCL, denCL]=cloop(Kp*num, den);
t=0:0.01:2;
step(numCL, denCL, t)
```

The above plot shows that the proportional controller reduced both the rise time and the steady-state error, increased the overshoot, and decreased the settling time by a small amount.

Proportional-Integral control

From the table, we see that an integral controller (Ki) decreases the rise time, increases both the overshoot and the settling time, and eliminates the steady-state error. For the given system, the closed-loop transfer function with a PI control is:

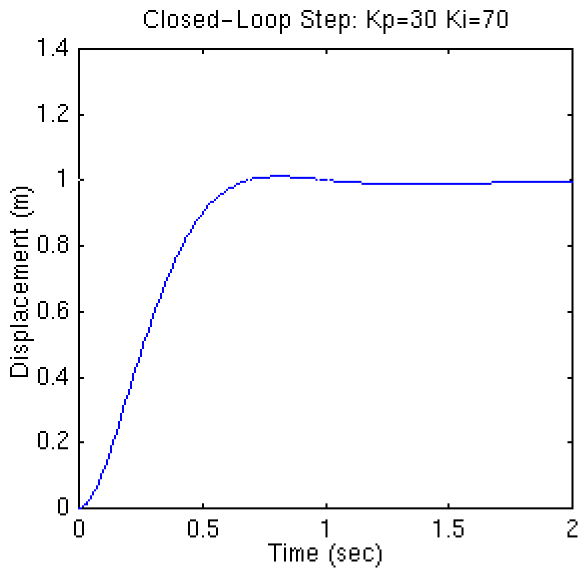
$$H(s) = \frac{K_p S + K_i}{s^3 + 10s^2 + (20 + K_p)s + K_i}$$

Let's reduce the Kp to 30, and let Ki equals to 70. Create a new m-file and enter the following commands.

```
Kp=30;
Ki=70;
num=[Kp Ki];
den=[1 10 20+Kp Ki];

t=0:0.01:2;
step(num,den,t)
```

Running this m-file in the MATLAB command window, you will get plot shown below.



We have reduced the proportional gain (Kp) because the integral controller also reduces the rise time and increases the overshoot as the proportional controller does (double effect). The above response shows that the integral controller eliminated the steady-state error.

Proportional-Derivative (PD) control

From the table shown above, we see that the derivative controller (Kd) reduces both the overshoot and the settling time. The closed-loop transfer function of the given system with a PD controller is:

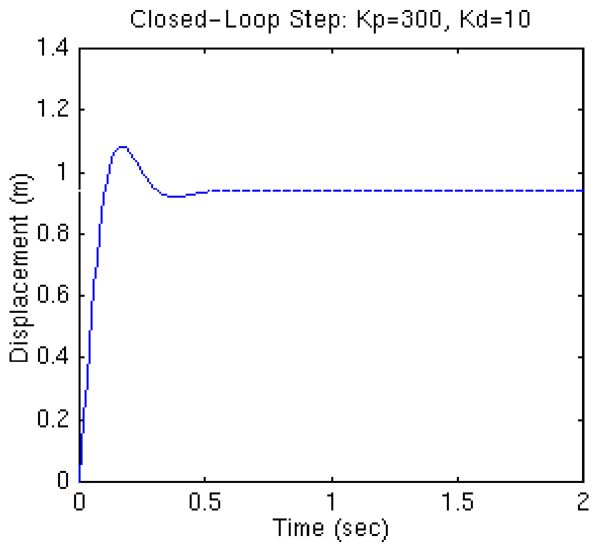
$$H(s) = \frac{K_d s + K_p}{s^2 + (10 + K_d)s + (20 + K_p)}$$

Let Kp equals to 300 as before and let Kd equals 10, change the m-file to the following:

```
Kp=300;
Kd=10;
num=[Kd Kp];
den=[1 10+Kd 20+Kp];

t=0:0.01:2;
step(num,den,t)
```

Running this m-file in the MATLAB command window, you will get plot shown below.



This plot shows that the derivative controller reduced both the overshoot and the settling time, and had small effect on the rise time and the steady-state error.

Proportional-Integral-Derivative control

Now, let's take a look at a PID controller. The closed-loop transfer function of the given system with a PID controller is:

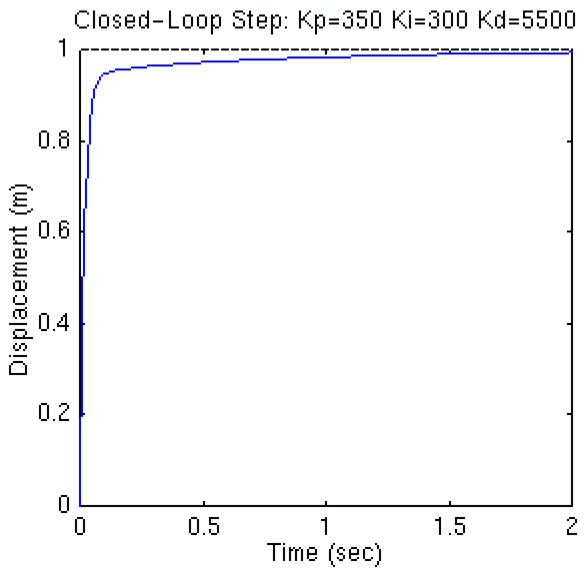
$$H(s) = \frac{\left(K_d s^2 + K_p s + K_i\right)}{s^3 + (10 + K_d)s^2 + \left(20 + K_p\right)s + K_i}$$

After several trial and error runs, the gains Kp=350, Ki=300, and Kd=50 provided the desired response. To confirm, enter the following commands to an m-file.

```
Kp=350;
Ki=300;
Kd=50;
num=[Kd Kp Ki];
den=[1 10+Kd 20+Kp Ki];

t=0:0.01:2;
step(num,den,t)
```

Running this m-file in the Matlab command window, you will get plot shown below.



Now, we have obtained the system with no overshoot, fast rise time, and no steady-state error.