**CHITTAGONG UNIVERSITY OF ENGINEERING AND TECHNOLOGY**
**DEPARTMENT OF ELECTRONICS & TELECOMMUNICATION ENGINEERING**
**CHITTAGONG-4349, BANGLADESH.**

**COURSE NO.: ETE 212**

**Experiment No. 2**

# Modeling with PID controller in MATLAB

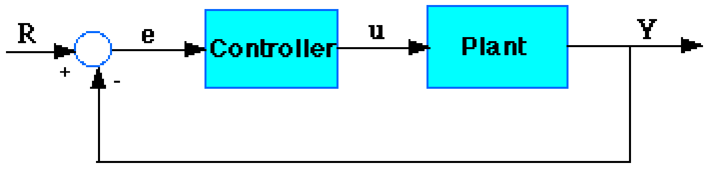
## PRELAB WORK:

- **Read this laboratory manual carefully before coming to the laboratory class, so that you know what is required.**
- Try to follow the lecture notes of ETE 211.
- Familiarize yourself with relevant MATLAB functions and codes necessary for this experiment.
- **Do not bring any prepared MATLAB code in the lab with any portable device.**
- **DONOT** copy others blindly!!!
- **Submit your lab report before the roll call.**

## THEORY WITH EXAMPLES:

### Introduction

This manual will show you the characteristics of the each of proportional (P), the integral (I), and the derivative (D) controls, and how to use them to obtain a desired response. In this manual, we will consider the following unity feedback system:



Plant: A system to be controlled
Controller: Provides the excitation for the plant; Designed to control the overall system behavior

### The three-term controller

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

$$K_P + \frac{K_I}{s} + K_D s = \frac{K_D s^2 + K_P s + K_I}{s}$$

- Kp = Proportional gain
- KI = Integral gain
- Kd = Derivative gain

First, let's take a look at how the PID controller works in a closed-loop system using the schematic shown above. The variable (e) represents the tracking error, the difference between the desired input value (R) and the actual output (Y). This error signal (e) will be sent to the PID controller, and the controller computes both the derivative and the integral of this error signal. The signal (u) just past the controller is now equal to the proportional gain (Kp) times the magnitude of the error plus the integral gain (Ki) times the integral of the error plus the derivative gain (Kd) times the derivative of the error.

$$u = K_P e + K_I \int e\,dt + K_D \frac{de}{dt}$$

This signal (u) will be sent to the plant, and the new output (Y) will be obtained. This new output (Y) will be sent back to the sensor again to find the new error signal (e). The controller takes this new error signal and computes its derivative and its integral again. This process goes on and on.

**The characteristics of P, I, and D controllers**

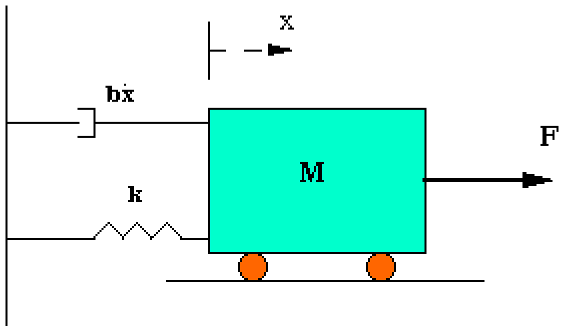
A proportional controller (Kp) will have the effect of reducing the rise time and will reduce but never eliminate the steady-state error. An integral control (Ki) will have the effect of eliminating the steady-state error, but it may make the transient response worse. A derivative control (Kd) will have the effect of increasing the stability of the system, reducing the overshoot, and improving the transient response. Effects of each of controllers Kp, Kd, and Ki on a closed-loop system are summarized in the table shown below.

| **CL RESPONSE** | **RISE TIME** | **OVERSHOOT** | **SETTLING TIME** | **S-S ERROR** |
|---|---|---|---|---|
| **Kp** | Decrease | Increase | Small Change | Decrease |
| **Ki** | Decrease | Increase | Increase | Eliminate |
| **Kd** | Small Change | Decrease | Decrease | Small Change |

Note that these correlations may not be exactly accurate, because Kp, Ki, and Kd are dependent on each other. In fact, changing one of these variab can change the effect of the other two. For this reason, the table should only be used as a reference when you are determining the values for Ki, Kp and Kd.

## Example Problem

Suppose we have a simple mass, spring, and damper problem.



The modeling equation of this system is

$$M\ddot{x} + b\dot{x} + kx = F \qquad (1)$$

Taking the Laplace transform of the modeling equation (1), we get

$$Ms^2X(s) + bsX(s) + kX(s) = F(s)$$

The transfer function between the displacement X(s) and the input F(s) then becomes

$$\frac{X(s)}{F(s)} = \frac{1}{Ms^2 + bs + k}$$

Let

- M = 1kg
- b = 10 N.s/m
- k = 20 N/m
- F(s) = 1

Plug these values into the above transfer function

$$\frac{X(s)}{F(s)} = \frac{1}{s^2 + 10s + 20}$$

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

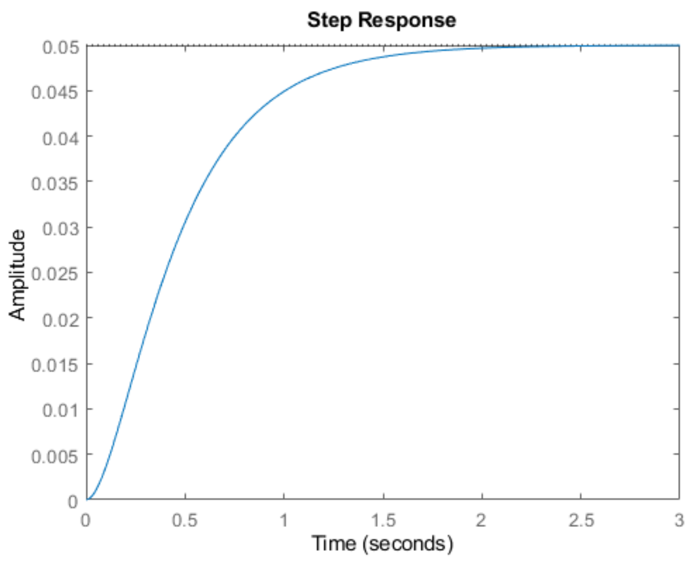
- Fast rise time
- Minimum overshoot
- No steady-state error

**Open-loop step response**

Let's first view the open-loop step response. Create a new m-file and add in the following code:

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

Running this m-file in the MATLAB command window should give you the 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 an 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**

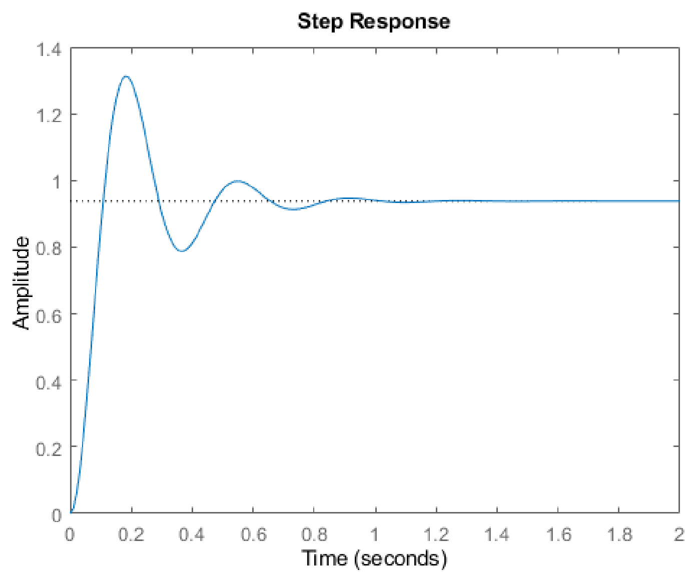
From the table shown above, we see that the proportional controller (Kp) reduces the rise time, increases the overshoot, and reduces the steady-state error. The closed-loop transfer function of the above system with a proportional controller is:

$$\frac{X(s)}{F(s)} = \frac{K_P}{s^2 + 10s + (20 + K_P)}$$

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

```
Kp = 300;
contr = Kp;
sys_cl = feedback(contr*plant,1);
t=0:0.01:2;
step(sys_cl,t)
```

Running this m-file in the MATLAB command window should give you the following plot.



***Note*: The MATLAB function called feedback was used to obtain a closed-loop transfer function directly from the open-loop transfer function (instead of computing closed-loop transfer function by hand).**

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 small amount.

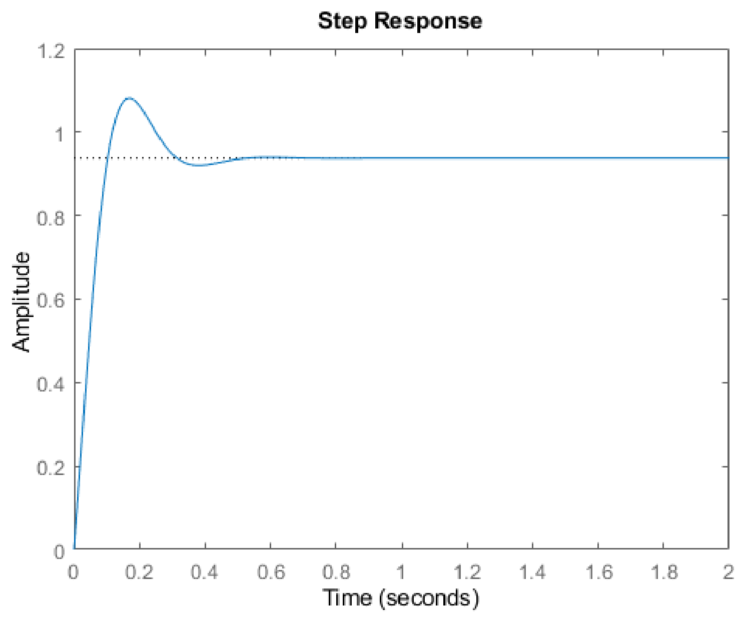
**Proportional-Derivative control**

Now, let's take a look at a 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:

$$\frac{X(s)}{F(s)} = \frac{K_D s + K_P}{s^2 + (10 + K_D)s + (20 + K_P)}$$

Let Kp equal 300 as before and let Kd equal 10. Enter the following commands into a m-file and run it in the MATLAB command window.

```
Kp = 300;
Kd = 10;
contr = tf([Kd Kp],1);
sys_cl = feedback(contr*plant,1);
t=0:0.01:2;
step(sys_cl,t)
```



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

**Proportional-Integral control**

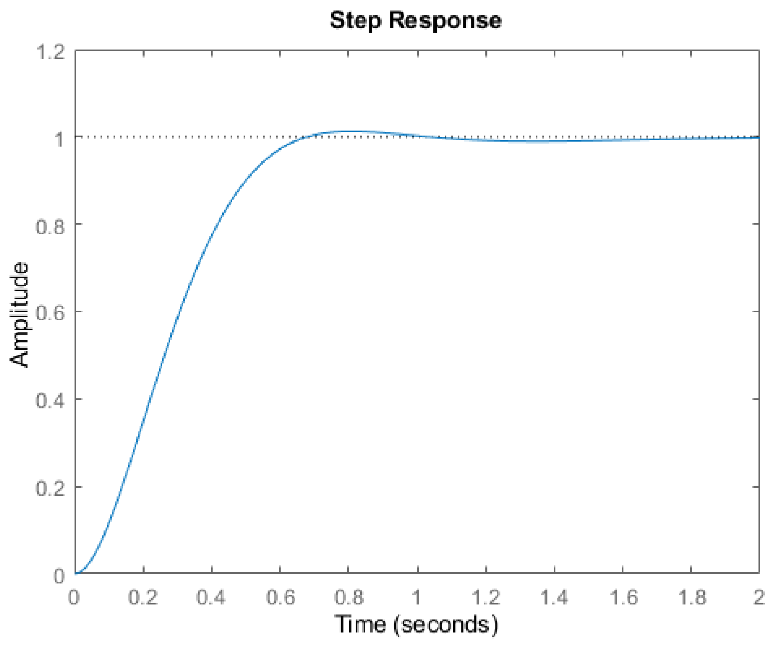
Before going into a PID control, let's take a look at a PI control. From the table, we see that an integral controller (Ki) decreases the rise time, increases both the overshoot and the settling time, and eliminates the steady-state error. For the given system, the closed-loop transfer function with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

$$\frac{X(s)}{F(s)} = \frac{K_P s + K_I}{s^3 + 10s^2 + (20 + K_P)s + K_I}$$

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

```
Kp = 30;
Ki = 70;
contr = tf([Kp Ki],[1 0]);
sys_cl = feedback(contr*plant,1);
t = 0:0.01:2;
step(sys_cl,t)
```

Run this m-file in the MATLAB command window, and you should get the following plot.



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.

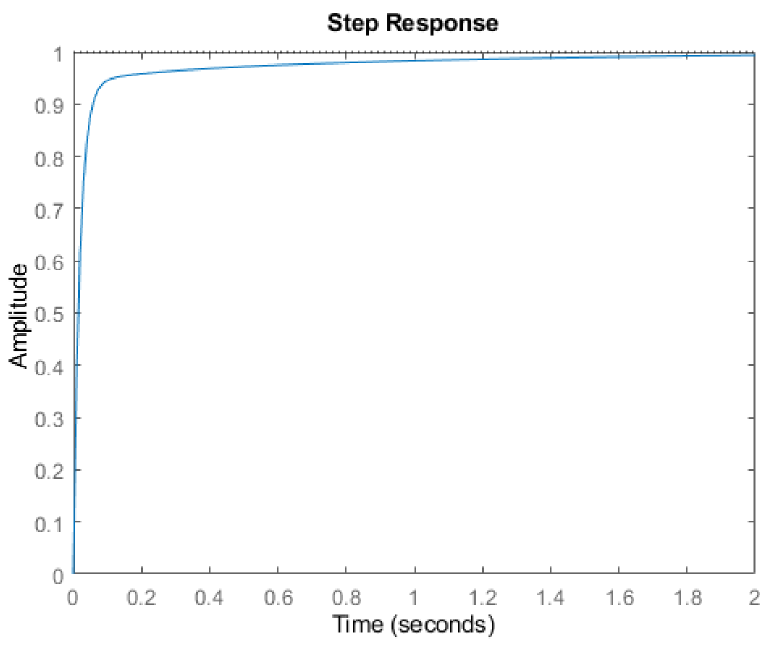
**Proportional-Integral-Derivative control**

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

$$\frac{X(s)}{F(s)} = \frac{K_D s^2 + K_P s + K_I}{s^3 + (10 + K_D)s^2 + (20 + K_P)s + K_I}$$

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 a m-file and run it in the command window. You should get the following step response.

```
Kp = 350;
Ki = 300;
Kd = 50;
contr = tf([Kd Kp Ki],[1 0]);
sys_cl = feedback(contr*plant,1);
t = 0:0.01:2;
step(sys_cl,t)
```



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

**General tips for designing a PID controller**

When you are designing a PID controller for a given system, follow the steps shown below to obtain a desired response.

1. Obtain an open-loop response and determine what needs to be improved
2. Add a proportional control to improve the rise time
3. Add a derivative control to improve the overshoot
4. Add an integral control to eliminate the steady-state error
5. Adjust each of Kp, Ki, and Kd until you obtain a desired overall response.

**Lastly**, please keep in mind that you do not need to implement all three controllers (proportional, derivative, and integral) into a single system, if not necessary. For example, if a PI controller gives a good enough response (like the above example), then you don't need to implement a derivative controller on the system. Keep the controller as simple as possible.

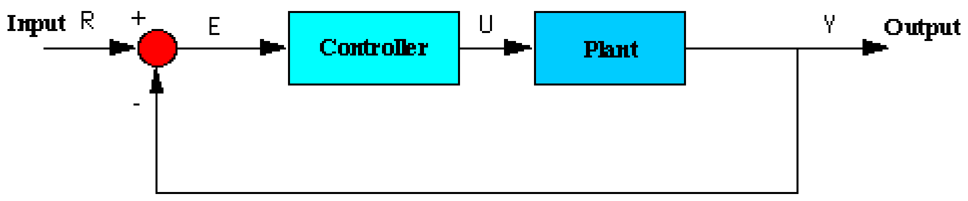
**A Real-time Example: Modeling a Cruise Control System Using PID control**

The transfer function for this cruise control problem is the following,

$$\frac{Y(s)}{U(s)} = \frac{1}{ms + b}$$

- m = 1000
- b = 50
- U(s) = 10
- Y(s) = velocity output

and the block diagram of an typical unity feedback system is shown below.



The design criteria for this problem are:

Rise time < 5 sec
Overshoot < 10%
Steady state error < 2%

The transfer function of a PID controller is

$$K_P + \frac{K_I}{s} + K_D s = \frac{K_D s^2 + K_P s + K_I}{s}$$

Let's first take a look at the proportional control.

## Proportional control

The first thing to do in this problem is to find a closed-loop transfer function with a proportional control (Kp) added. By reducing the block diagram, the closed-loop transfer function with a proportional controller becomes:

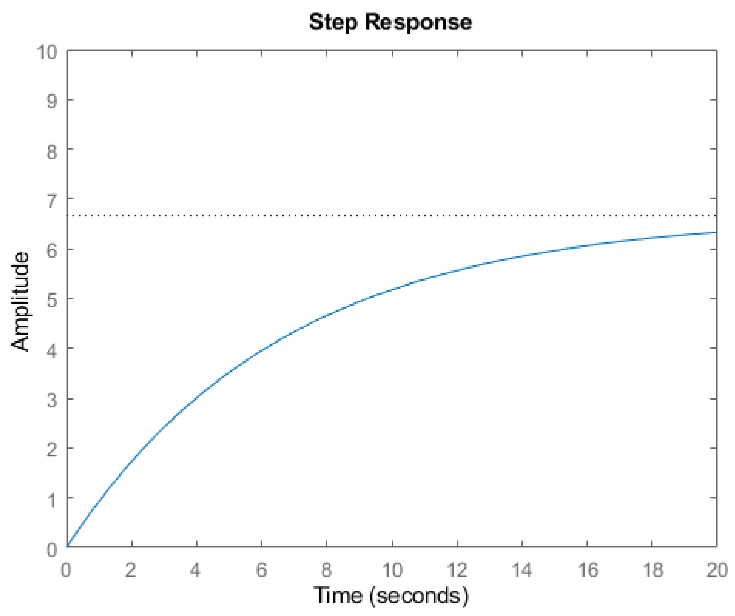
$$\frac{Y(s)}{R(s)} = \frac{k_P}{ms + (b + k_P)}$$

Recall from the PID manual page, a proportional controller (Kp) decreases the rise time. This is what we need, if you refer to the Cruise Control Modeling page.

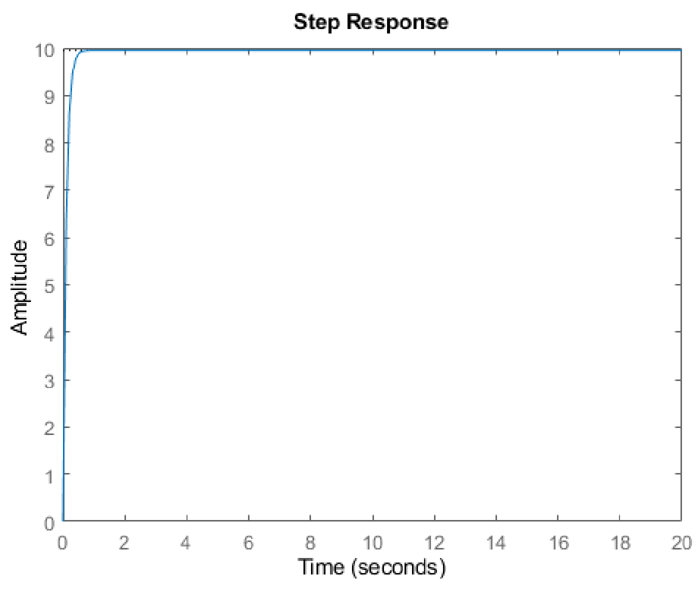
For now, let Kp equal 100 and see what happens to the response. Create a new m-file and enter the following commands.

```
Kp = 100;
m = 1000;
b = 50;
u = 10;
num = [1];
den = [m b];
cruise = tf(num,den);
sys_cl = feedback(Kp*cruise,1);
t = 0:0.1:20;
step(u*sys_cl,t)
axis([0 20 0 10])
```

Running this m-file in the MATLAB command window should give you the following step response.



As you can see from the plot, neither the steady-state error nor the rise time do not satisfy our design criteria. You can increase the proportional gain (Kp) to improve the system output. Change the existing m-file so that Kp equal 10000 and rerun it in the MATLAB command window. You should see the following plot.



The steady-state error has dropped to near zero and the rise time has decreased to less than 0.5 second. However, this response is unrealistic because a real cruise control system generally cannot change the speed of the vehicle from 0 to 10 m/s in less than 0.5 second.

The solution to this problem is to choose a proportional gain (Kp) that will give a reasonable rise time, and add an integral controller to eliminate the steady-state error.

**PI control**

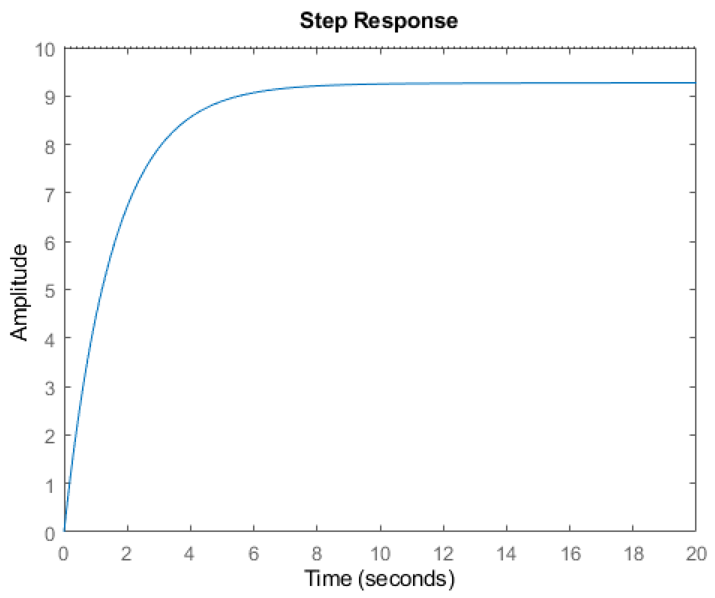
The closed-loop transfer function of this cruise control system with a PI controller is:

$$\frac{Y(s)}{R(s)} = \frac{K_P s + K_I}{ms^2 + (b + K_P)s + K_I}$$

Recall from the PID manual page, an addition of an integral controller to the system eliminates the steady-state error. For now, let Kp equal 600 and Ki equal 1 and see what happens to the response. Change your m-file to the following.

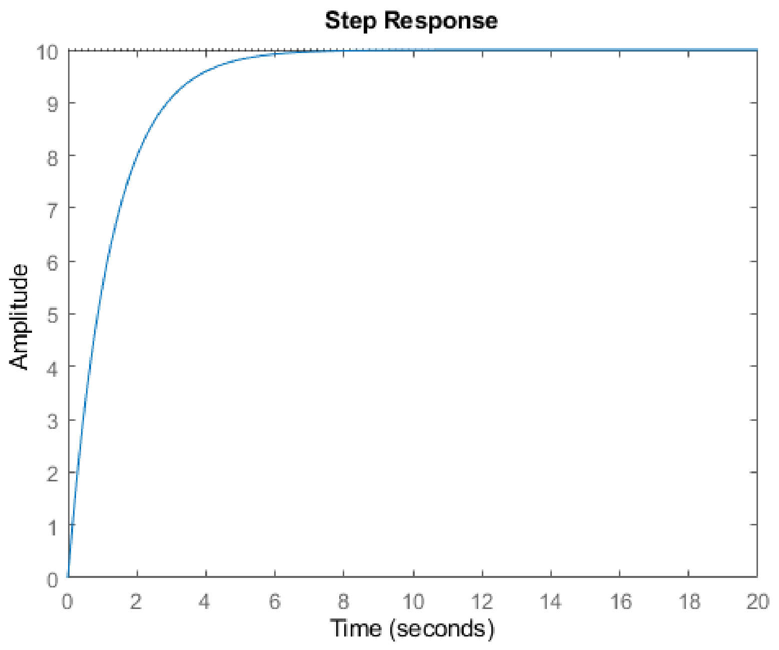
```
m = 1000;
b = 50;
num = [1];
den = [m b];
cruise = tf(num,den);
Kp = 600;
Ki = 1;
contr = tf([Kp Ki],[1 0]);
sys_cl = feedback(contr*cruise,1);
u = 10;
t = 0:0.1:20;
step(u*sys_cl,t)
axis([0 20 0 10])
```

You should get the following output:



Now adjust both the proportional gain (Kp) and the integral gain (Ki) to obtain the desired response. When you adjust the integral gain (Ki), we suggest you to start with a small value since a large (Ki) can destabilize the response.

When Kp equal 800 and Ki equal 40, the step response will look like the following:



As you can see, this step response meets all design criteria.

**PID control**

For this particular example, no implementation of a derivative controller was needed to obtain a required output. However, you might want to see how to work with a PID control for the future reference. The closed-loop transfer function for this cruise control system with a PID controller is.

$$\frac{Y(s)}{R(s)} = \frac{K_D s^2 + K_P s + K_I}{(m + K_D)s^2 + (b + K_P)s + K_I}$$

Let Kp equal 1, Ki equal 1, and Kd equal 1 and enter the following commands into a new m-file.

```
Kp = 1;
Ki = 1;
Kd = 1;
m = 1000;
b = 50;
u = 10;
cruise = tf(num,den);
contr = tf([Kd Kp Ki],[1 0]);
sys_cl = feedback(contr*cruise,1);
t = 0:0.1:20;
step(u*sys_cl,t)
axis ([0 20 0 10])
```

Running this m-file should give you the step response of the system with PID controller. Adjust all of Kp, Kd, and Ki until you obtain satisfactory results. We will leave this as an exercise for you to work on.

**Suggestion:** Usually choosing appropriate gains require trial and error processes. The best way to attack this tedious process is to adjust one variable (Kp, Kd, or Ki) at a time and observe how changing one variable influences the system output.

## Home Task

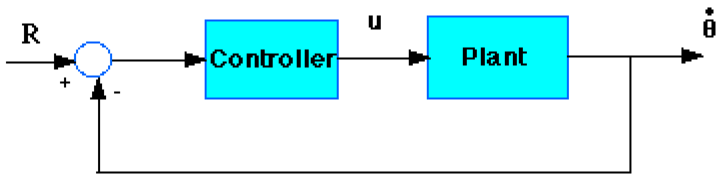
The dynamic equations and the open-loop transfer function of the DC Motor are:

$$s(Js + b)\Theta(s) = KI(s)$$

$$(Ls + R)I(s) = V - Ks\Theta(s)$$

$$\frac{\dot{\theta}}{V} = \frac{K}{(Js + b)(Ls + R) + K^2}$$

And the system schematic looks like:



Assume the following values for the physical parameters.

moment of inertia of the rotor (J) = (0.01+ <your ID>)E-6 kg.m^2/s^2
damping ratio of the mechanical system (b) = (0.1 + <your ID>)E-6 Nms
electromotive force constant (K=Ke=Kt) = (0.01 + 0.<your ID>) Nm/Amp
electric resistance (R) = 1 ohm
electric inductance (L) = (0.5 + <your ID>) H
input (V): Source Voltage
output (theta): position of shaft
The rotor and shaft are assumed to be rigid

With a 1 rad/sec step input, the design criteria are:

Settling time less than 2 seconds
Overshoot less than 5%
Steady-stage error less than 1%

Design a **PID controller** satisfying all design requirements.