SOLUTION 4.1

$$\dot{\mathbf{z}} = \mathbf{P}^{-1}\mathbf{A}\mathbf{P}\mathbf{z} + \mathbf{P}^{-1}\mathbf{B}u$$

$$y = \mathbf{C}\mathbf{P}\mathbf{z}$$

$$\mathbf{P}^{-1} = \begin{bmatrix} -4 & 9 & -3 \\ 0 & -4 & 7 \\ -1 & -4 & -9 \end{bmatrix}; \quad \therefore \quad \mathbf{P} = \begin{bmatrix} -0.2085 & -0.3029 & -0.1661 \\ 0.0228 & -0.1075 & -0.0912 \\ 0.0130 & 0.0814 & -0.0521 \end{bmatrix}$$

$$\mathbf{P}^{-1}\mathbf{A}\mathbf{P} = \begin{bmatrix} 18.5961 & 25.4756 & 5.6156 \\ -12.9023 & -28.8893 & -8.3909 \\ -0.5733 & 11.4169 & 5.2932 \end{bmatrix}; \quad \mathbf{P}^{-1}\mathbf{B} = \begin{bmatrix} -58 \\ 63 \\ -12 \end{bmatrix}; \quad \mathbf{C}\mathbf{P} = \begin{bmatrix} 1.5668 & 3.0423 & 2.7329 \end{bmatrix}$$

SOLUTION 4.1

```
A = [-1 -7 6; -8 4 8; 4 7 -8];
B = [-5; -7; 5];
C = [-9 -9 -8];
invP = [-4 9 -3; 0 -4 7; -1 -4 -9];
P = inv(invP);
P1AP = invP*A*P
P1B = invP*B
CP = C*P
eig1 = eig(A)
eig2 = eig(P1AP)
```

```
eig1 =

  -15.8352
    0.7515
   10.0837
```

```
eig2 =

  -15.8352
   10.0837
    0.7515
```

SOLUTION 4.2

```
syms s
A=[0 2 ; - 3 -5]; B=[0;1];
C=[1 3]; X0=[2;1];
U = 1 / ( s + 1 ) ;
I=[1 0;0 1];
X=( (s*I-A)^-1) * (X0+B*U)
Y=C*X;
Y=simplify (Y)
y=ilaplace (Y)
pretty(y)
eig(A)
```

$$\mathbf{X}(s) = (s\mathbf{I} - \mathbf{A})^{-1}\mathbf{x}(0) + (s\mathbf{I} - \mathbf{A})^{-1}\mathbf{B}\mathbf{U}(s)$$

$$= \frac{\text{adj}(s\mathbf{I} - \mathbf{A})}{\det(s\mathbf{I} - \mathbf{A})}[\mathbf{x}(0) + \mathbf{B}\mathbf{U}(s)]$$

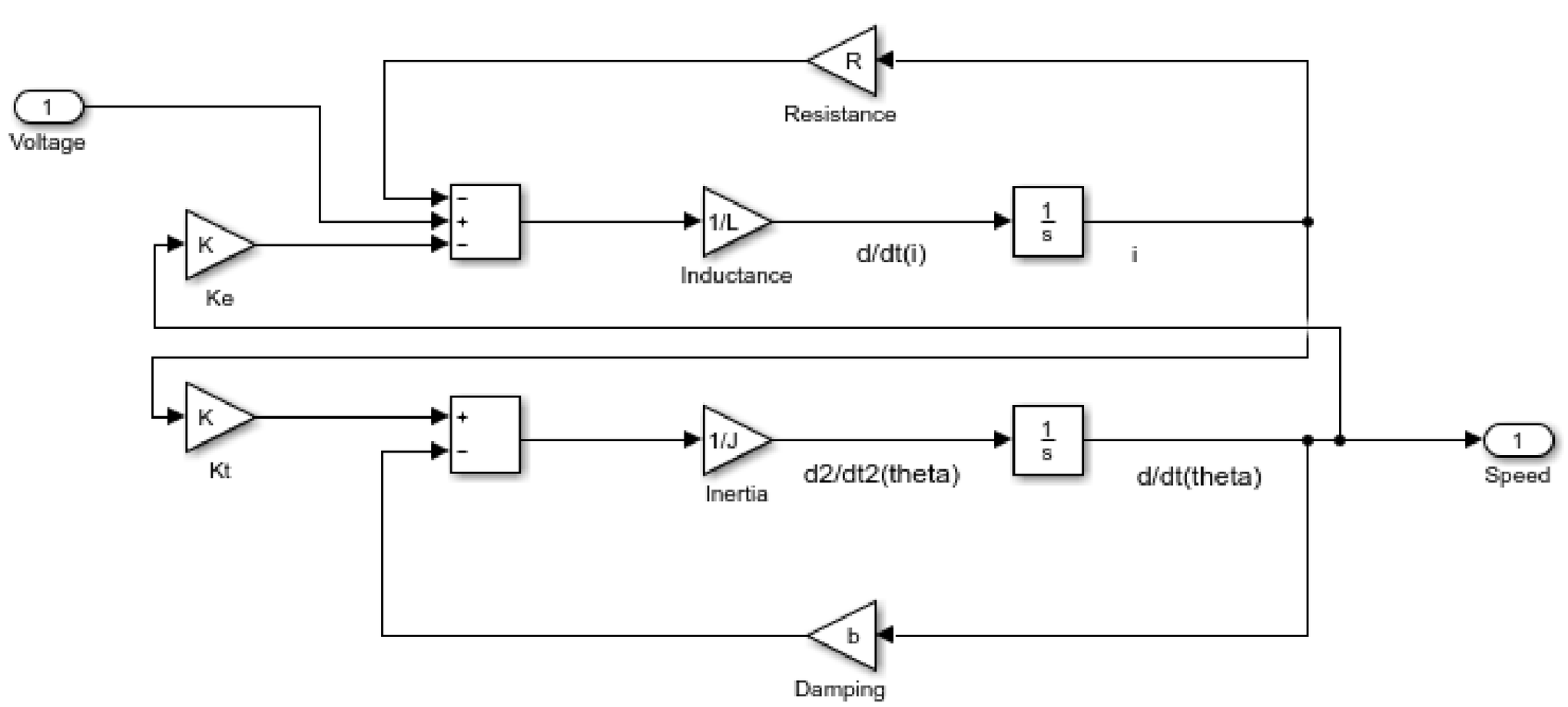
$$\mathbf{Y}(s) = \mathbf{C}\mathbf{X}(s) + \mathbf{D}\mathbf{U}(s)$$

```
exp(-3 t) 35                    exp(-t)
----------- - exp(-2 t) 12 - -------
     2                           2


ans =

   -2.0000
   -3.0000
```

SOLUTION 4.3

$$J\frac{d^2\theta}{dt^2} = T - b\frac{d\theta}{dt} \Longrightarrow \frac{d^2\theta}{dt^2} = \frac{1}{J}(K_t i - b\frac{d\theta}{dt})$$

$$L\frac{di}{dt} = -Ri + V - e \Longrightarrow \frac{di}{dt} = \frac{1}{L}(-Ri + V - K_e\frac{d\theta}{dt})$$

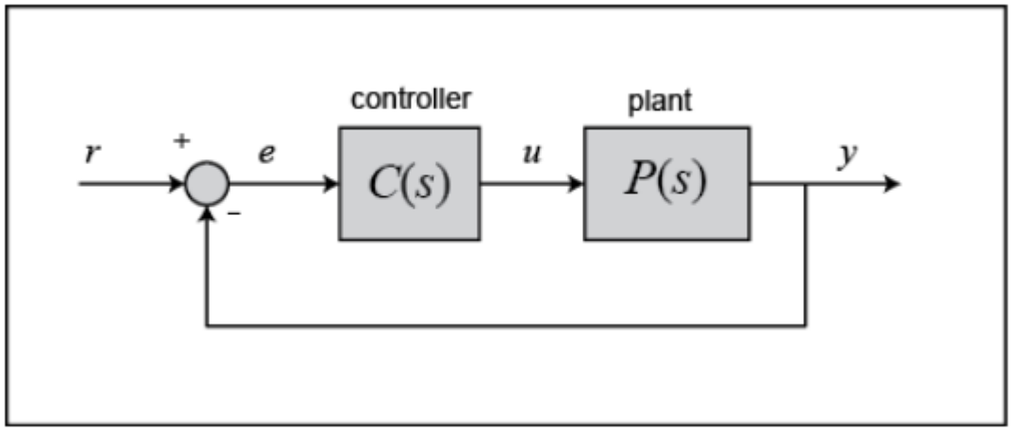


PID CONTROLLER DESIGN

- We will study on a simple, yet versatile, feedback compensator structure: the Proportional-Integral-Derivative (PID) controller. The PID controller is widely employed because it is very understandable and because it is quite effective
- We will discuss the effect of each of the PID parameters on the dynamics of a closed-loop system and will demonstrate how to use a PID controller to improve a system's performance.

PID CONTROLLER DESIGN

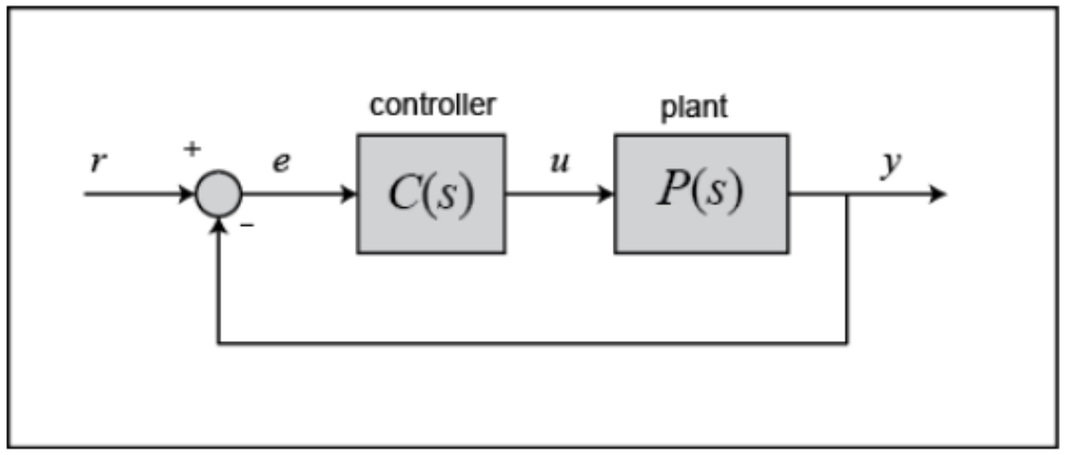
- We will consider the following unity-feedback system:



- The output of a PID controller, which is equal to the control input to the plant, is calculated in the time domain from the feedback error as follows:

$$u(t) = K_p e(t) + K_i \int e(t)dt + K_p \frac{de}{dt}$$

PID CONTROLLER DESIGN



- 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 output (r) and the actual output (y).

- This error signal (e) is fed to the PID controller, and the controller computes both the derivative and the integral of this error signal with respect to time.

- The control signal (u) to the plant is equal to the proportional gain (Kp) times the magnitude of the error plus the integral gain (Ki) times the ingegral of the error plus the derivative gain (Kd) times the derivative of the error.

$$u(t) = K_p e(t) + K_i \int e(t)dt + K_p \frac{de}{dt}$$

PID CONTROLLER DESIGN

- This control signal (u) is fed to plant and the new output (y) is obtained. The new output (y) is then fed back and compared to the reference to find new error signal (e). The controller takes this new error signal and computes an update of the control input. This process continues while the controller is in effect.

- The transfer functions of a PID controller is found by taking the Laplace transform of the equation :

$$u(t) = K_p e(t) + K_i \int e(t)dt + K_p \frac{de}{dt}$$

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

- Kp = proportional gain
- Ki = integral gain
- Kd = derivative gain

PID CONTROLLER DESIGN

- We can define a PID controller in Matlab using a transfer functions model directly, for example :

```
Kp = 1;
Ki = 1;
Kd = 1;

s = tf('s');
C = Kp + Ki/s + Kd*s
```

```
C =

  s^2 + s + 1
  -----------
       s

Continuous-time transfer function.
```

PID CONTROLLER DESIGN

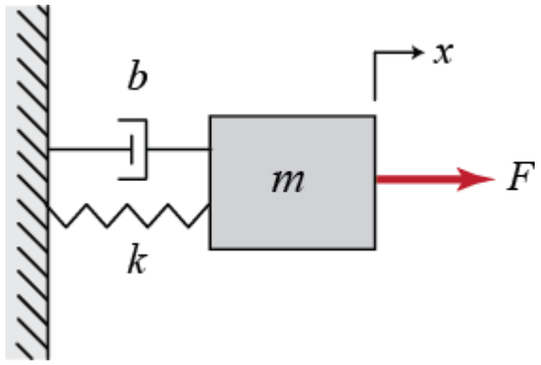
- Alternatively, we may use MATLAB's pid object to generate an equivalent continuous-time controller as follows:

```
C = pid(Kp,Ki,Kd)
```

```
C =

                1
  Kp + Ki * --- + Kd * s
                s

  with Kp = 1, Ki = 1, Kd = 1

Continuous-time PID controller in parallel form.
```

EXAMPLE

- Suppose we have a simple mass-spring-damper system.



- The governing equation of this system is :

$$m\ddot{x} + b\dot{x} + kx = F$$

- Taking the Laplace transform of the governing equation, we get

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

- The transfer functions between the input force F(s) and the output displacement X(s) then becomes

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

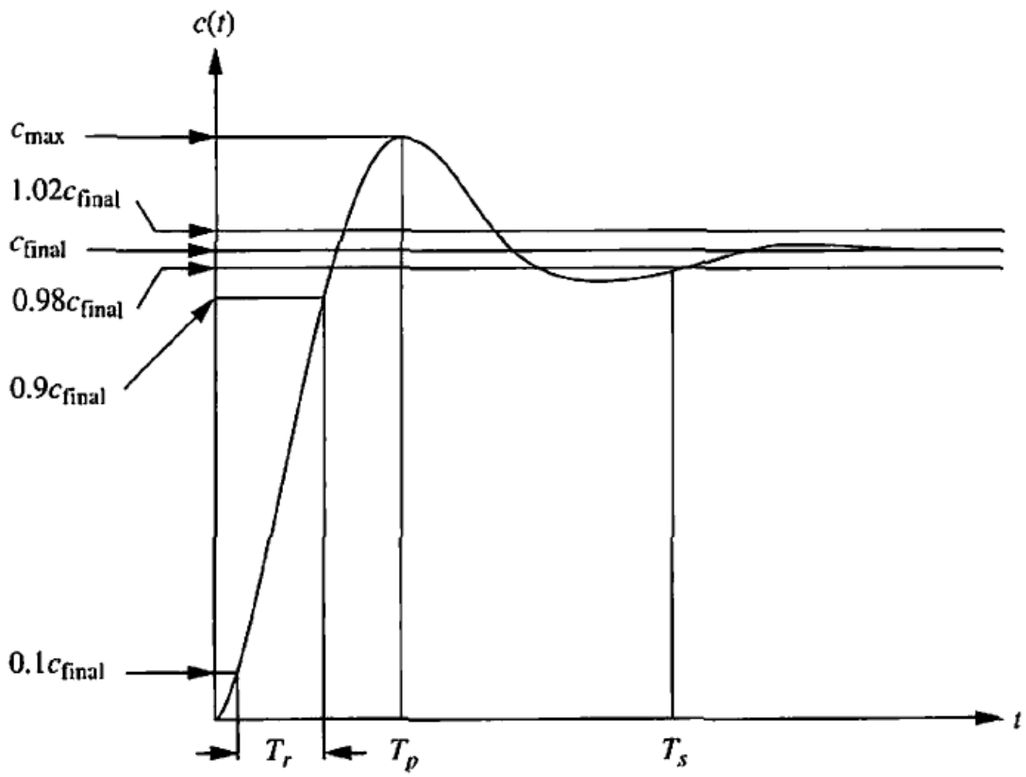
EXAMPLE

- The goal of this problem is to show how each of the terms, Kp, Ki and Kd contributes to obtaining the common goals of :

✓ Fast rise time

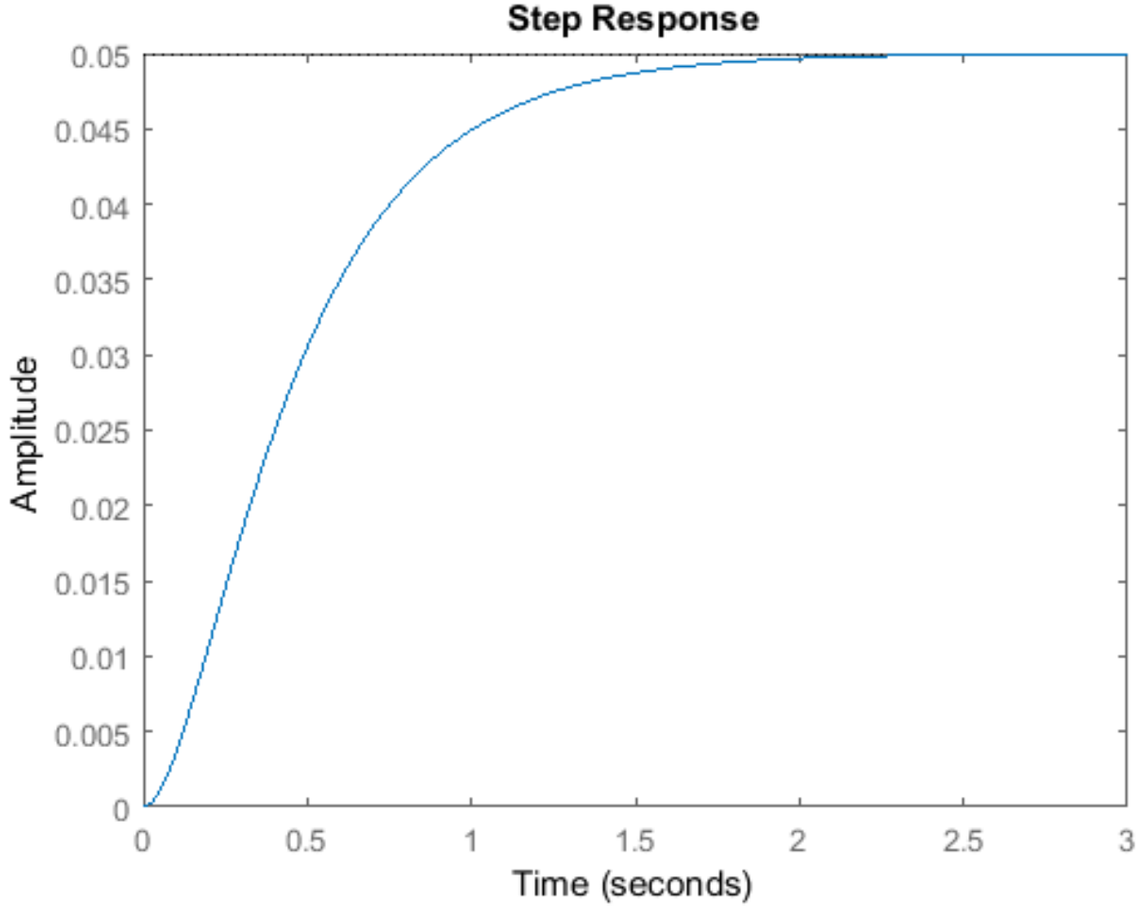
✓ Minimal overshoot

✓ Zero steady-state error



EXAMPLE

```
m = 1; % kg
b = 10; % N s/m
k = 20; % N/m
F = 1; % N
s = tf('s');
P = 1/(m*s^2 + b*s + k)
step(P)
```

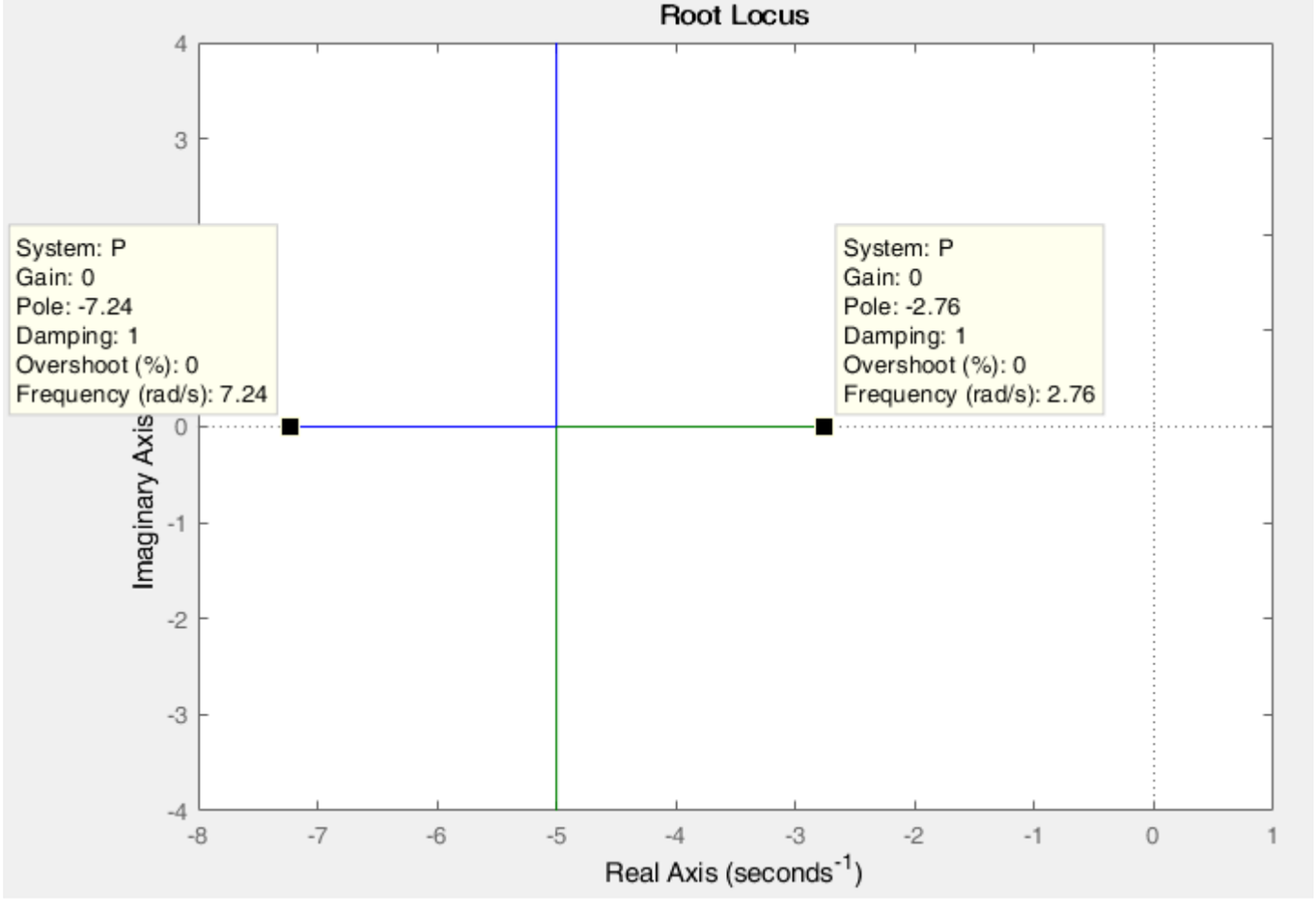


- The final value of the output to a unit step input is 0.05. This corresponds to a steady-state error of 0.95, whic is quite large.
- The rise time is about one second, and the settling time is about 1.5 seconds.
- We will design a controller that will reduce the rise time, reduce the settling time, and eliminate the steady-state error.

ROOT LOCUS

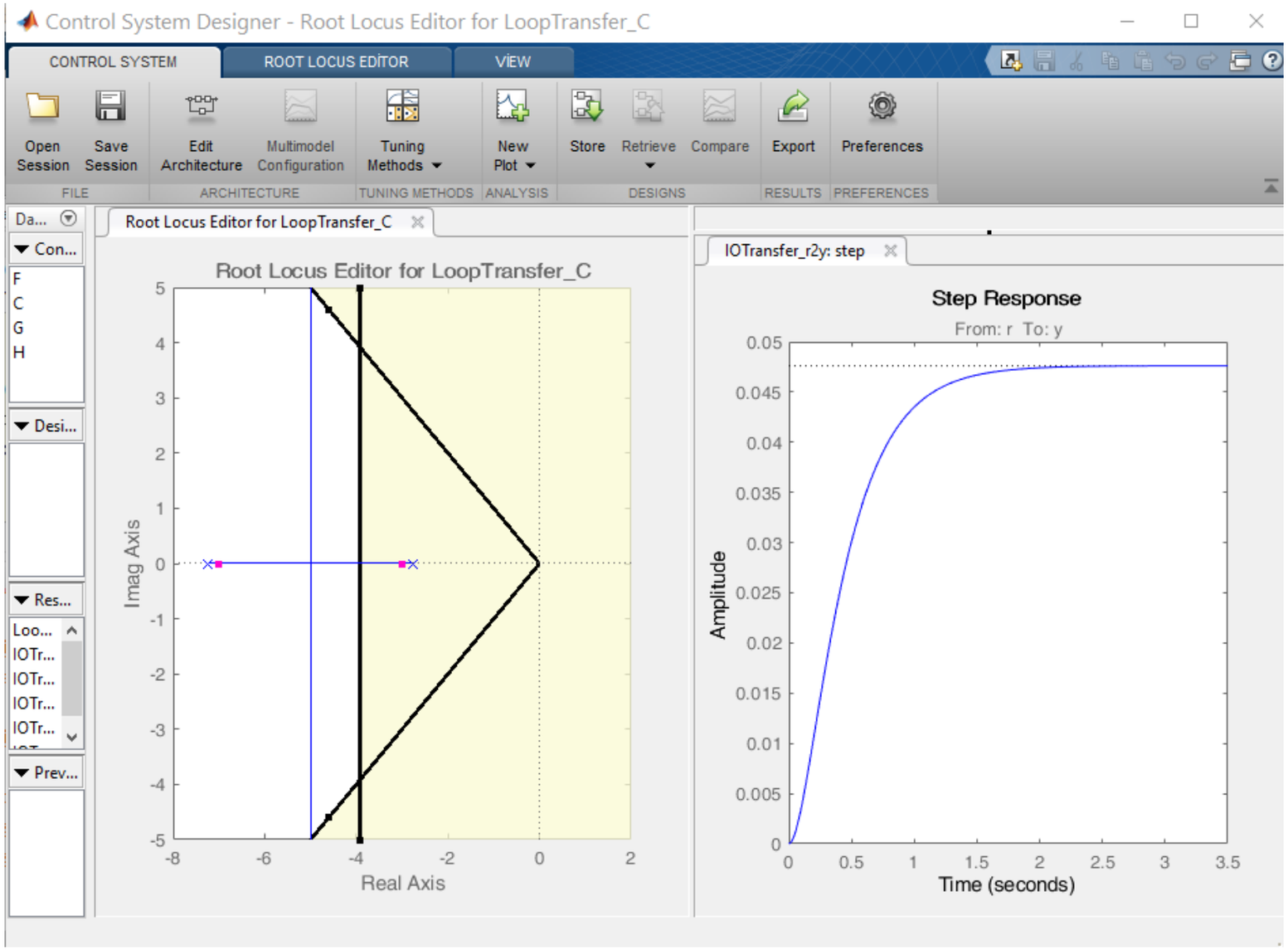
```
>> [p,z] = pzmap(P)

p =

   -7.2361
   -2.7639
```

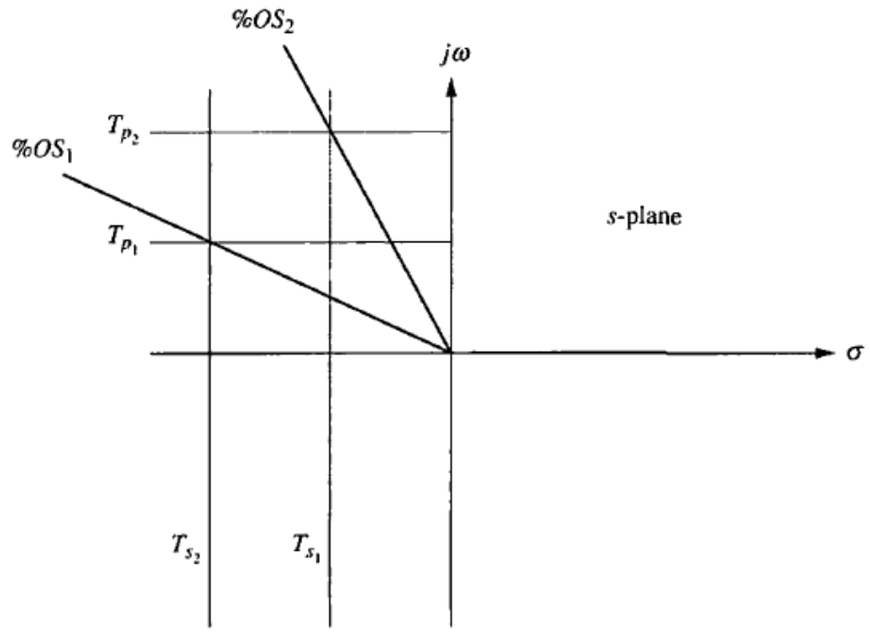
```
rlocus(P)
```



ROOT LOCUS

```
sisotool(P)
```





Lines of constant peak time, T_p, settling time, T_s, and percent overshoot, $\%OS$. Note: $T_{s_2} < T_{s_1}$; $T_{p2} < T_{p1}$; $\%OS_1 < \%OS_2$.

PROPORTIONAL CONTROL

- The closed-loop transfer functions of out unity-feedback system with a proportional controller is the following, where X(s) is our output (equals Y(s)) and our reference R(s) is the input

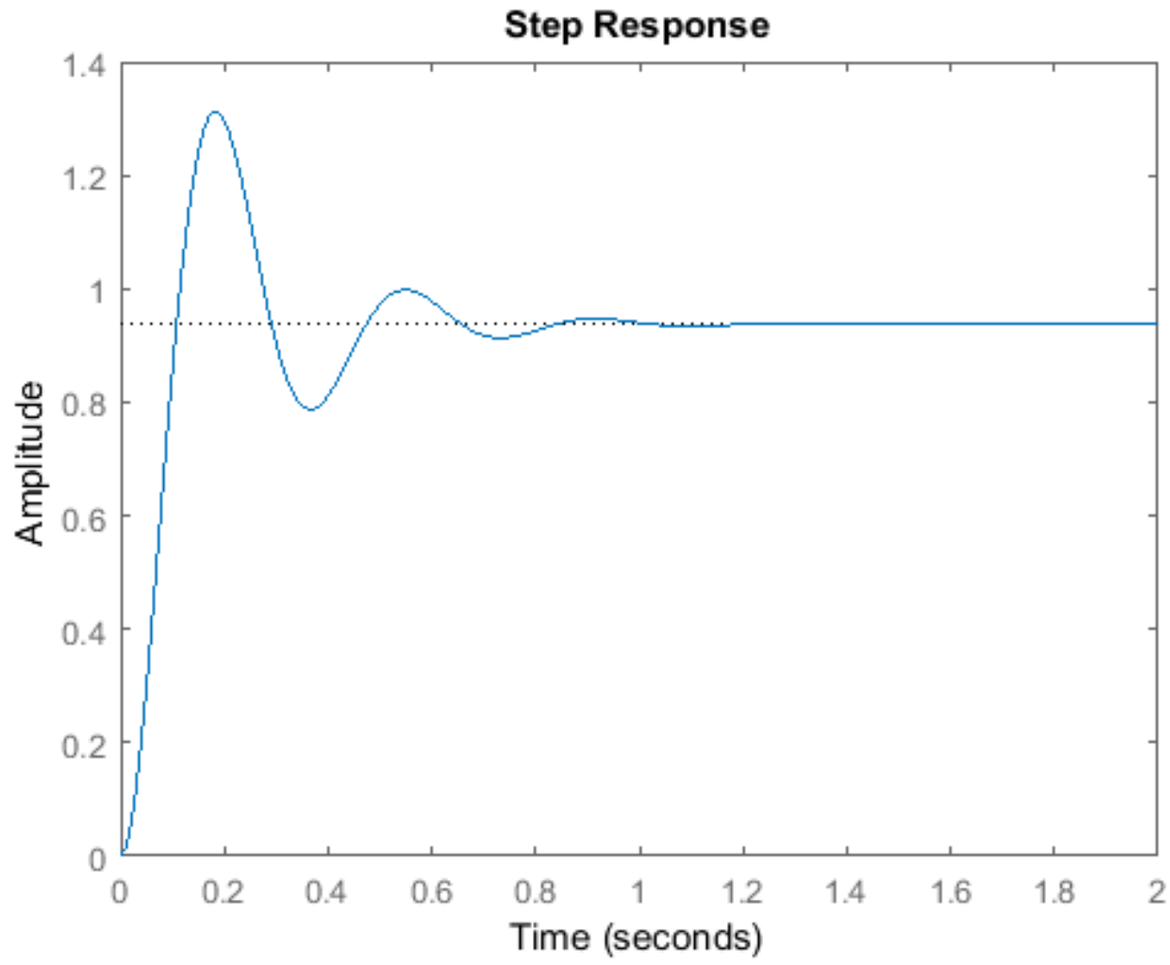
$$T(s) = \frac{X(s)}{R(s)} = \frac{K_p}{s^2 + 10s + (20 + K_p)}$$

- Let the proportional gain (Kp) equal 300 :

```
Kp = 300;
C = pid(Kp)
T = feedback(C*P,1)

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

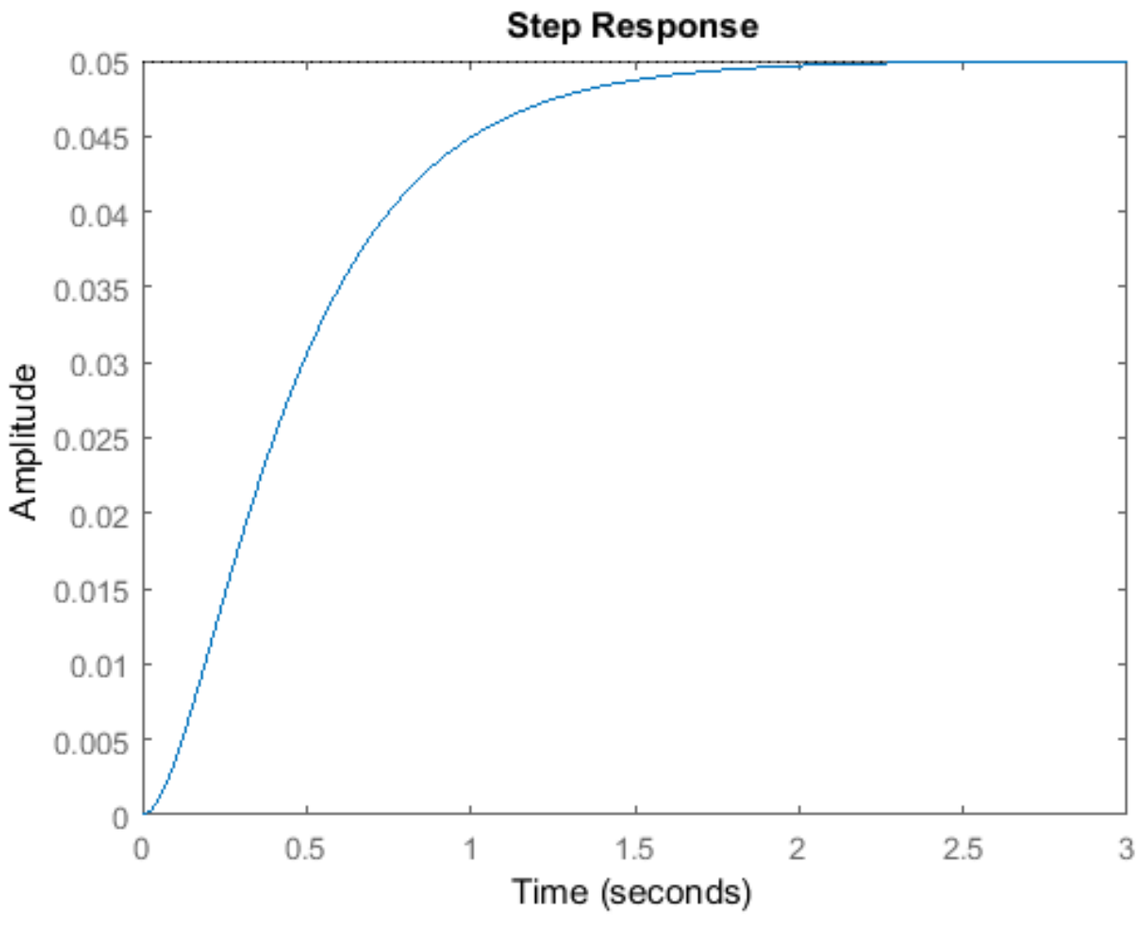
PROPORTIONAL CONTROL

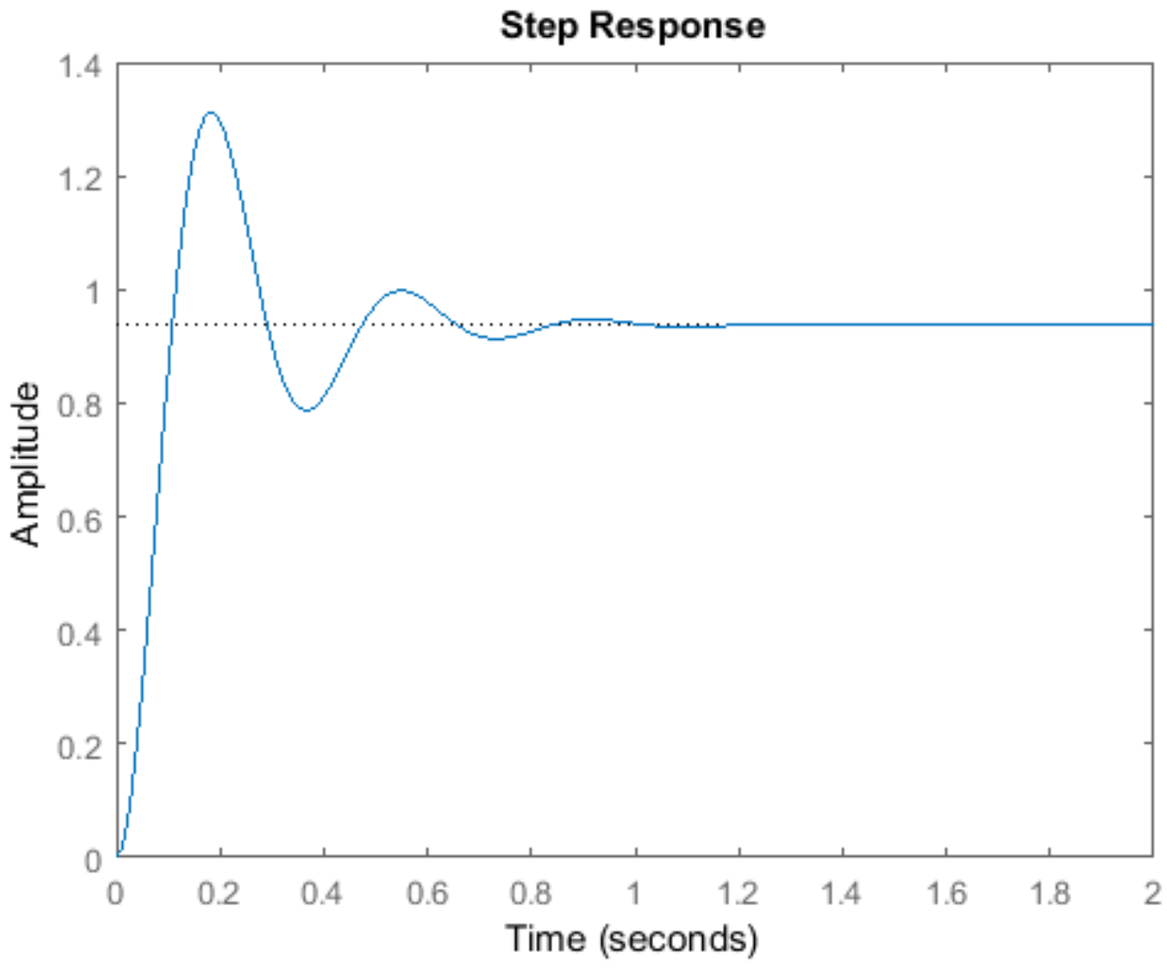
```
C =

  Kp = 300

P-only controller.


T =

        300
  ----------------
  s^2 + 10 s + 320

Continuous-time transfer function.
```



PROPORTIONAL CONTROL

What is the effect of proportional controller on a closed-loop system ?

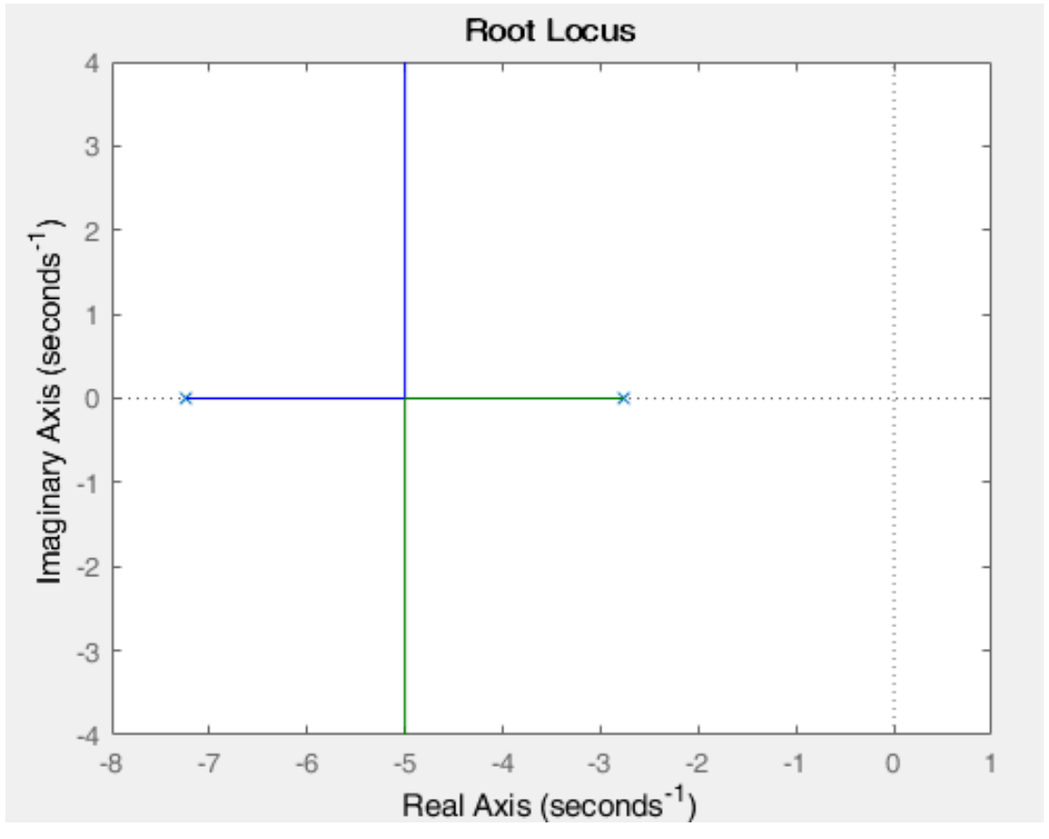


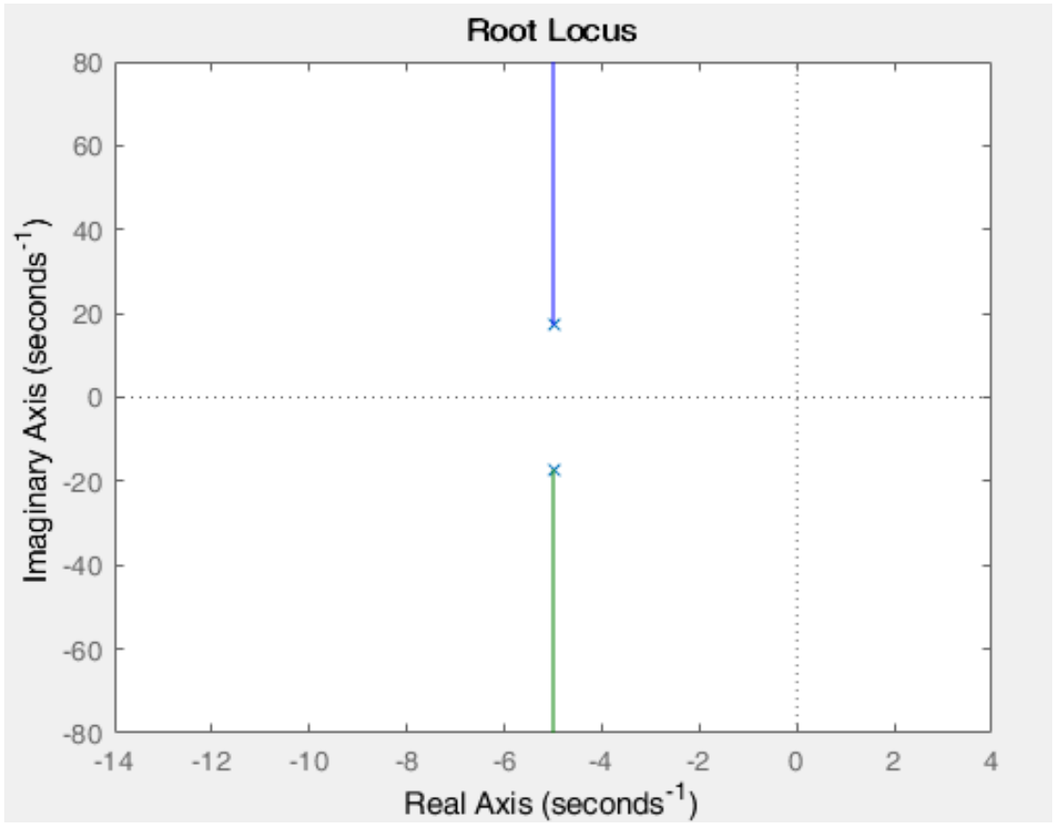


P

PROPORTIONAL CONTROL

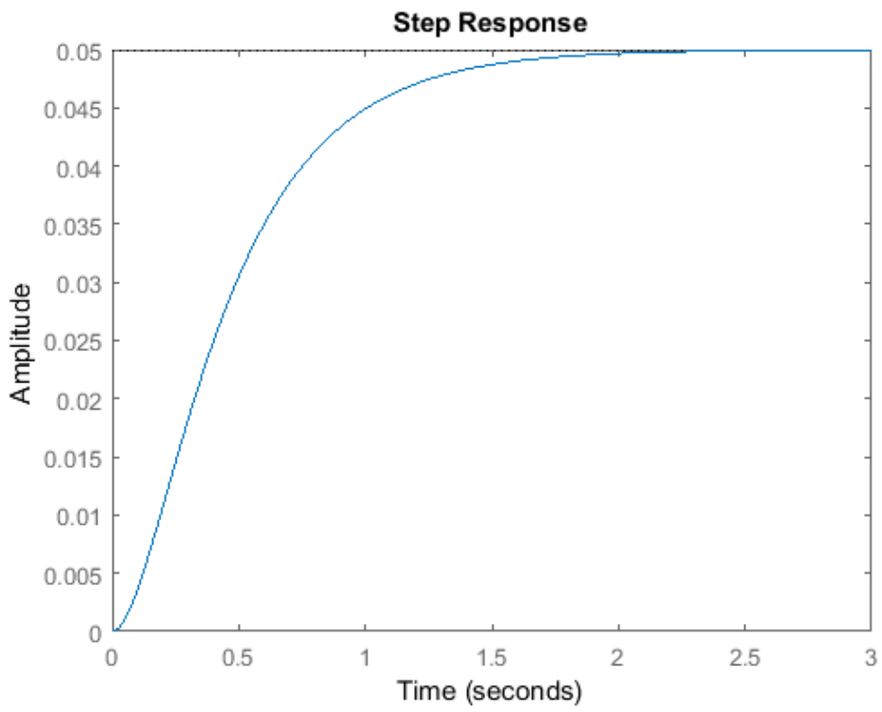
What is the effect of proportional controller on a closed-loop system ?

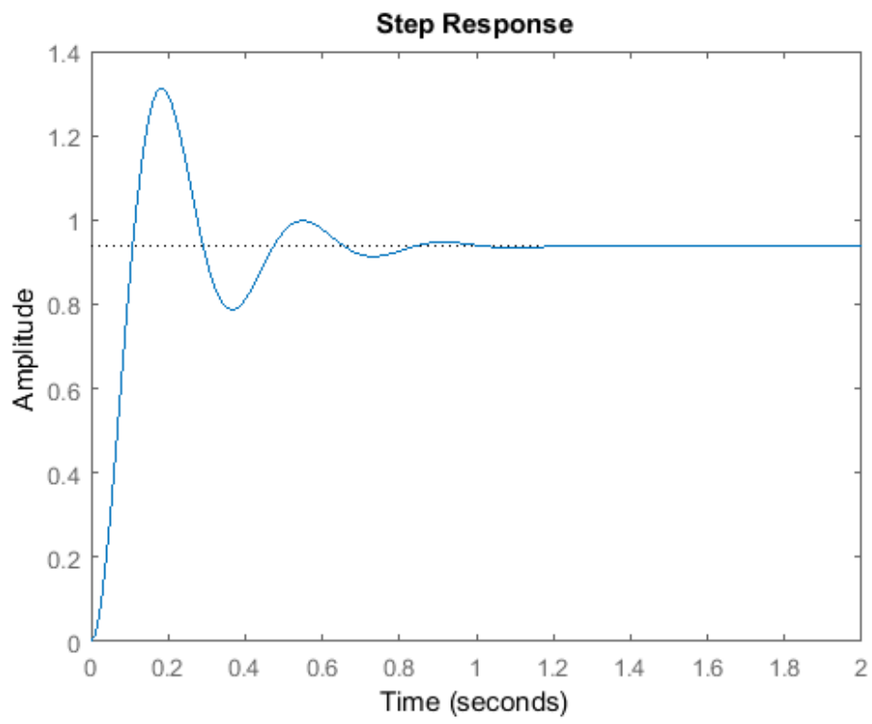




P

PROPORTIONAL CONTROL





P

- The above plots show that the proportional controller reduced both the rise time and the stead-state error, increased the overshoot, and decreased the settling time by a small amount

PROPORTIONAL – DERIVATIVE CONTROL

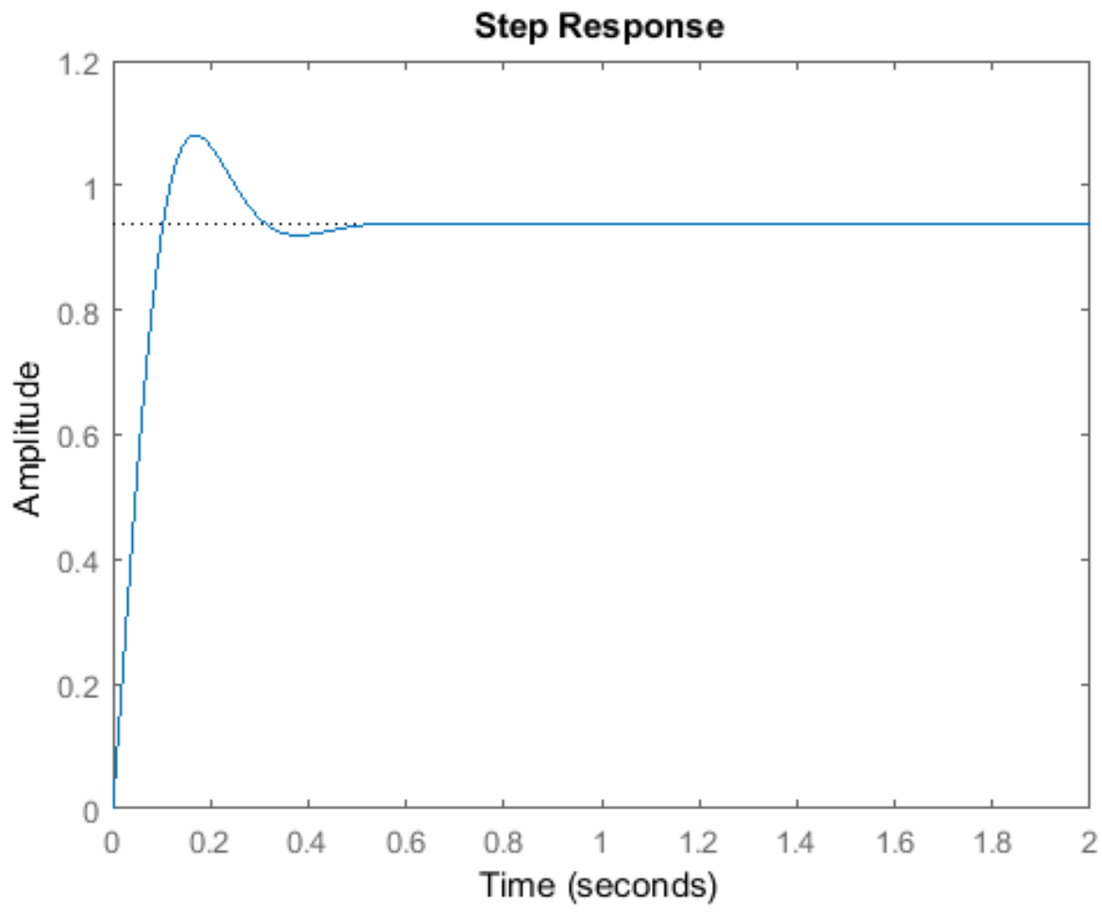
- The closed-loop transfer functions of the given system with a PD controller is :

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

```
Kp = 300;
Kd = 10;
C = pid(Kp,0,Kd)
T = feedback(C*P,1)

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

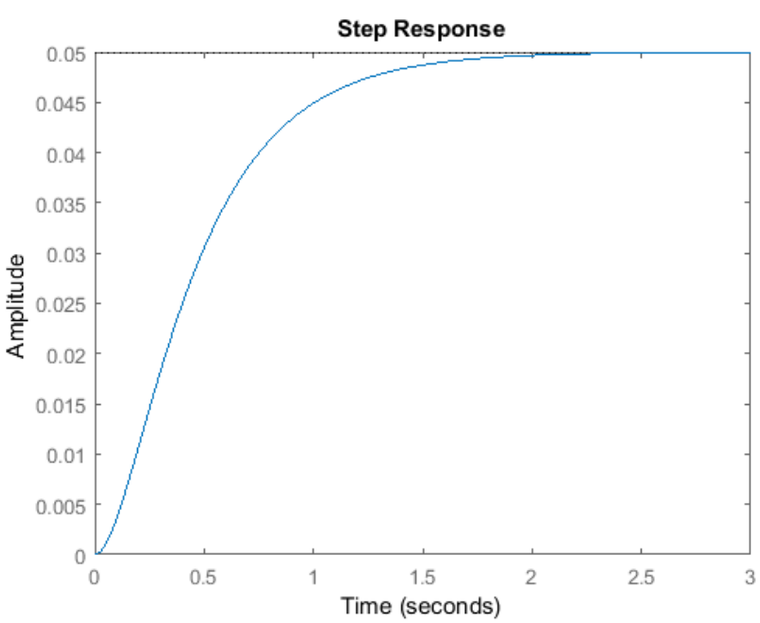
PROPORTIONAL – DERIVATIVE CONTROL

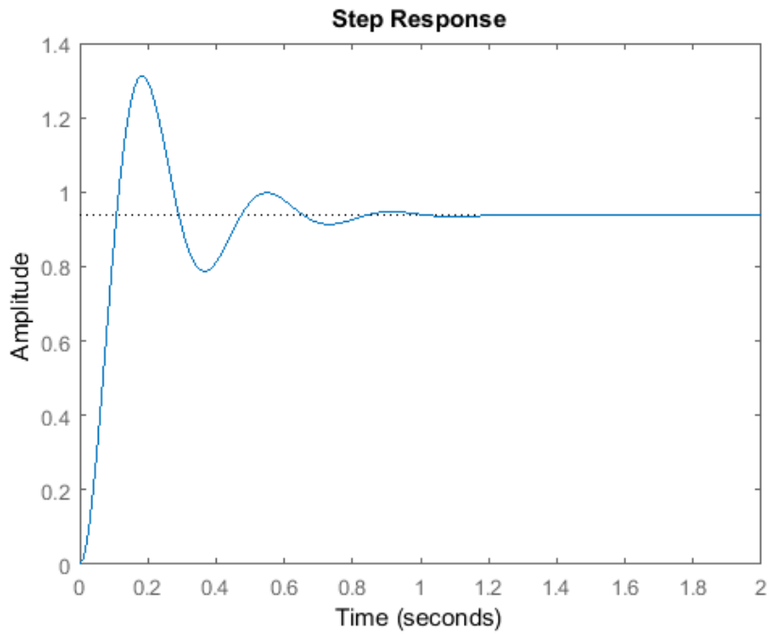
```
C =

  Kp + Kd * s

  with Kp = 300, Kd = 10

Continuous-time PD controller in parallel form.


T =

     10 s + 300
  ----------------
  s^2 + 20 s + 320

Continuous-time transfer function.
```



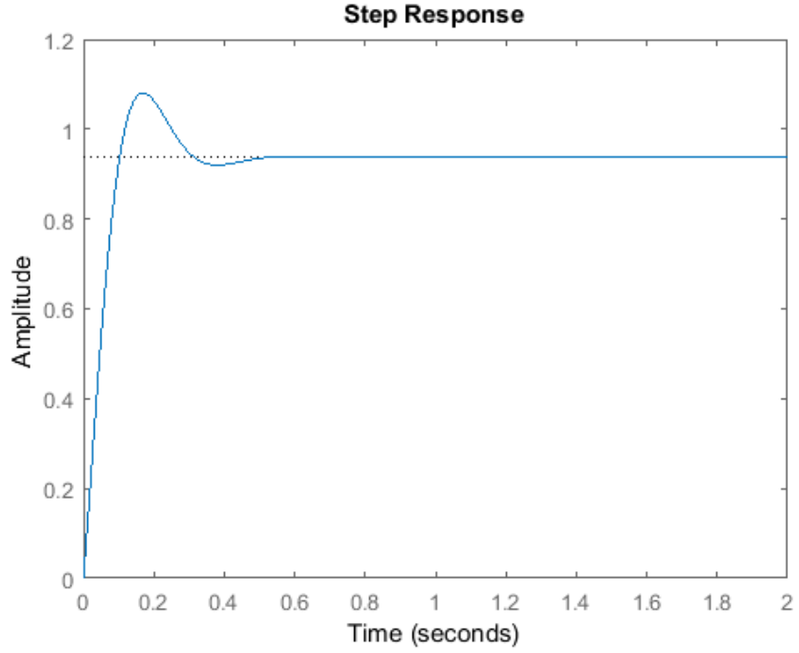
PROPORTIONAL – DERIVATIVE CONTROL

What is the effect of addition of the derivative term on a closed-loop system ?



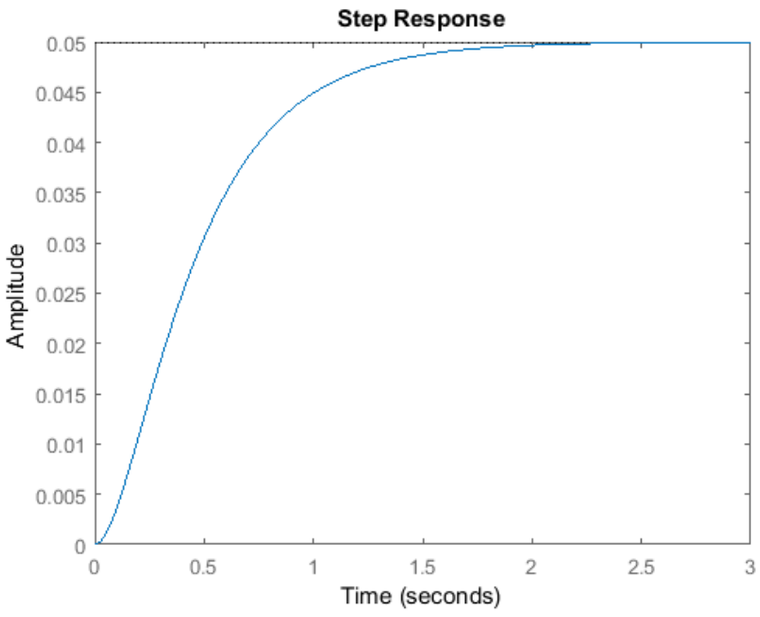


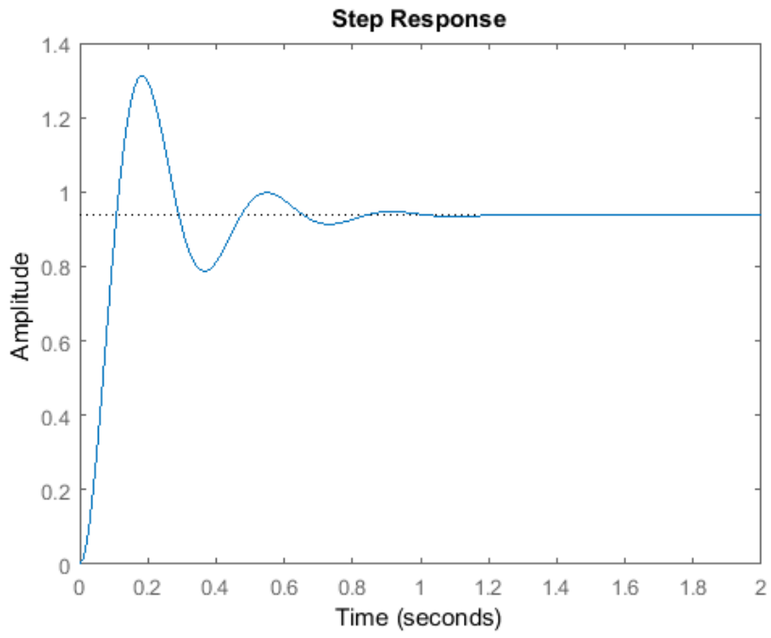
P

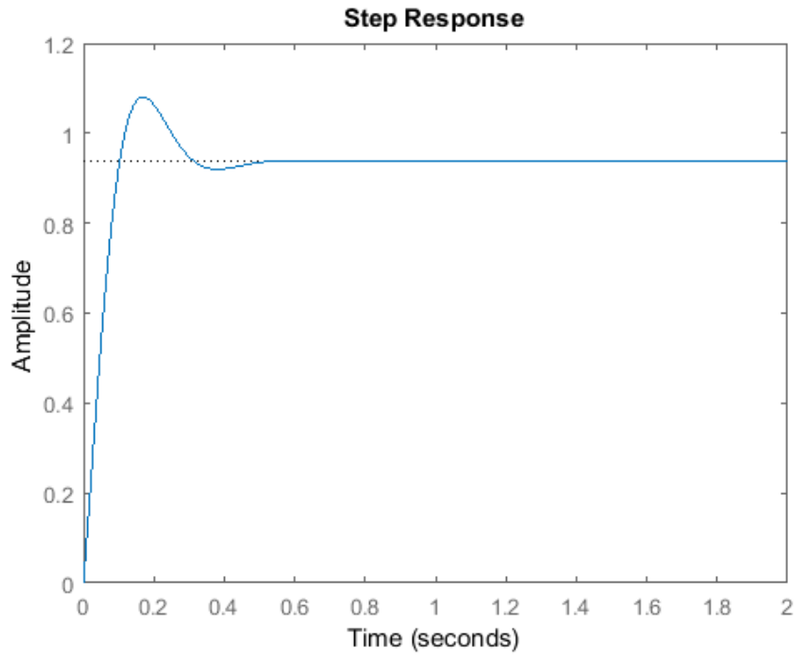


PD

PROPORTIONAL – DERIVATIVE CONTROL







P **PD**

- The above plots show that the addition of the derivative term reduced both the overshoot and the settling time, and had a negligible effect on the rise time and the steady – state error.

PROPORTIONAL – INTEGRAL CONTROL

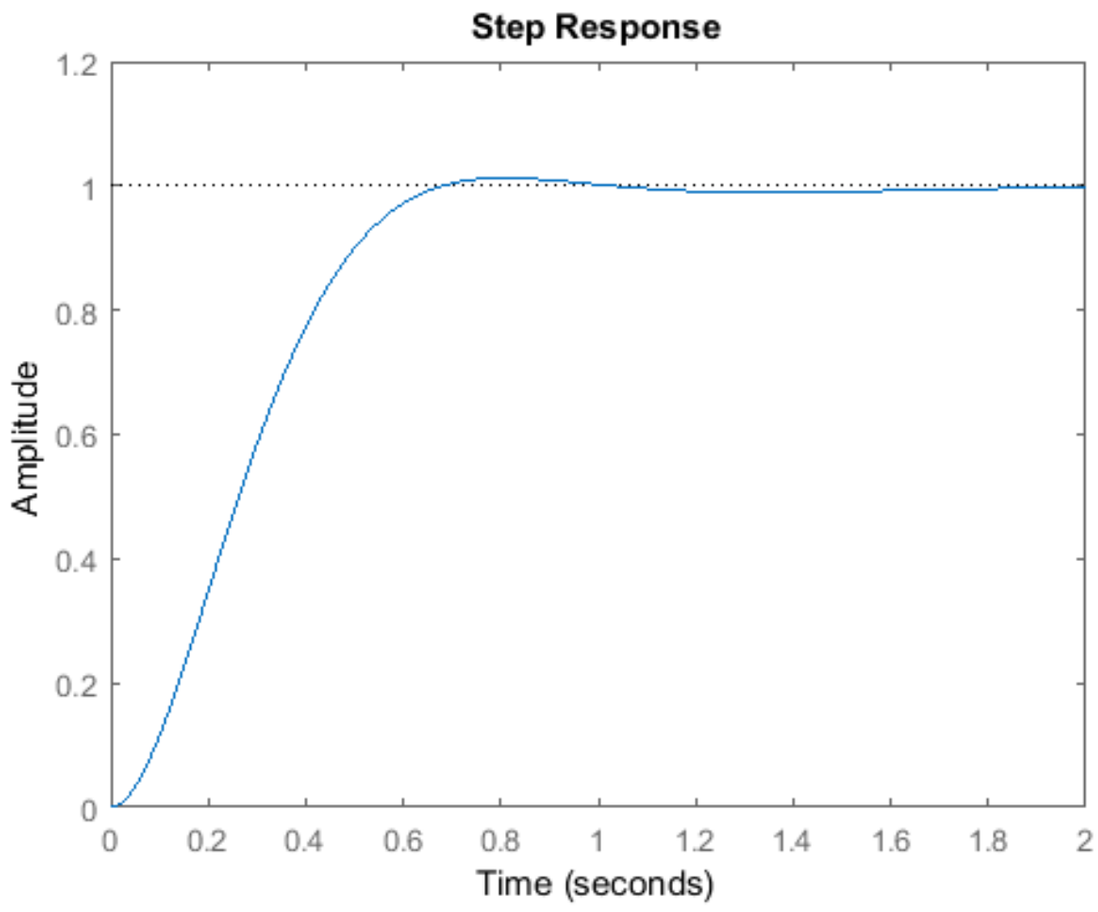
- The closed-loop transfer functions of the given system with a PD controller is :

$$T(s) = \frac{X(s)}{R(s)} = \frac{K_p s + K_i}{s^3 + 10s^2 + (20 + K_p)s + K_i}$$

```
Kp = 30;
Ki = 70;
C = pid(Kp,Ki)
T = feedback(C*P,1)

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

PROPORTIONAL – INTEGRAL CONTROL

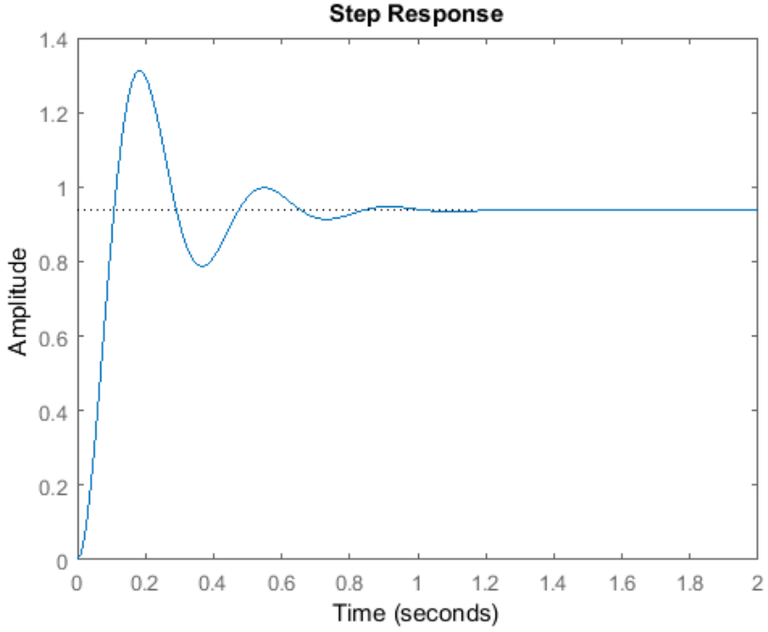
```
C =

             1
  Kp + Ki * ---
             s

  with Kp = 30, Ki = 70

Continuous-time PI controller in parallel form.
```



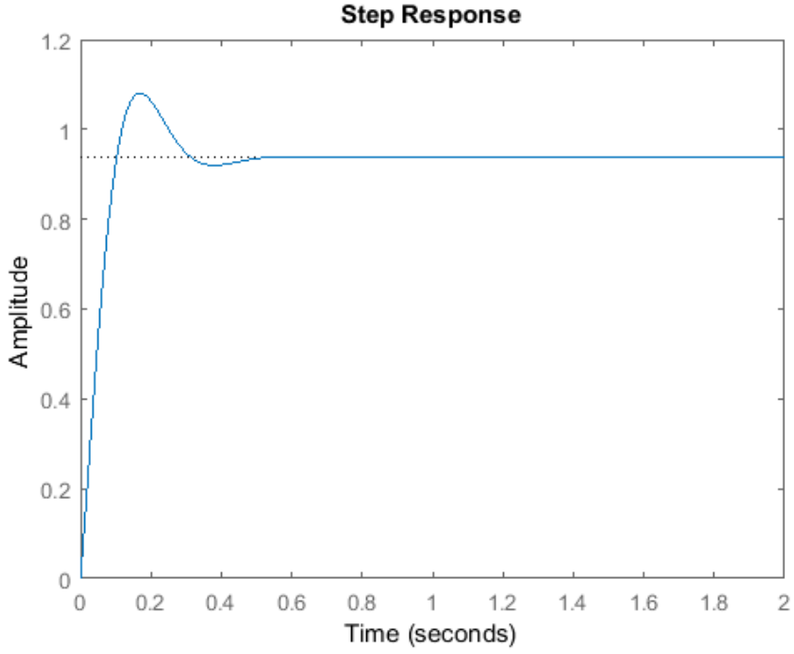

```
T =

        30 s + 70
  ----------------------
  s^3 + 10 s^2 + 50 s + 70

Continuous-time transfer function.
```

PROPORTIONAL – INTEGRAL CONTROL

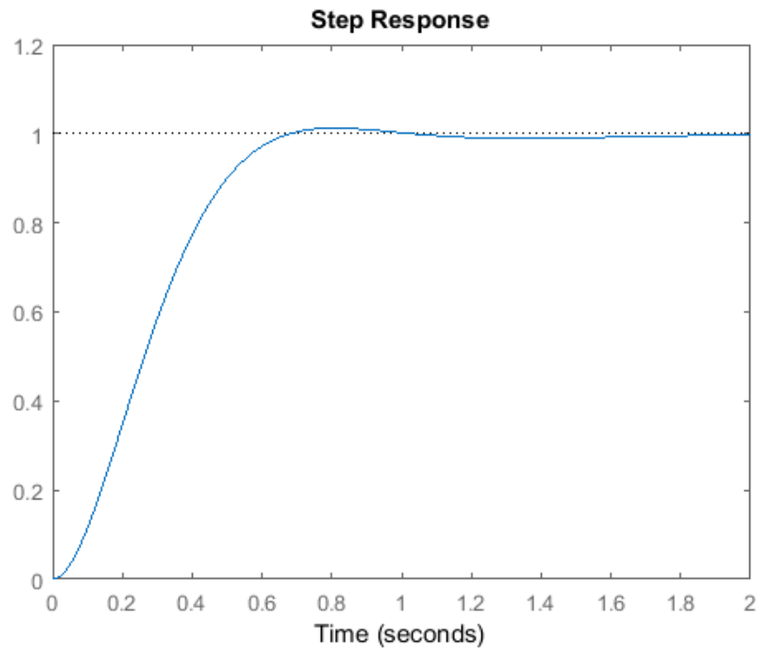
What is the effect of addition of the integration term on a closed-loop system ?



P

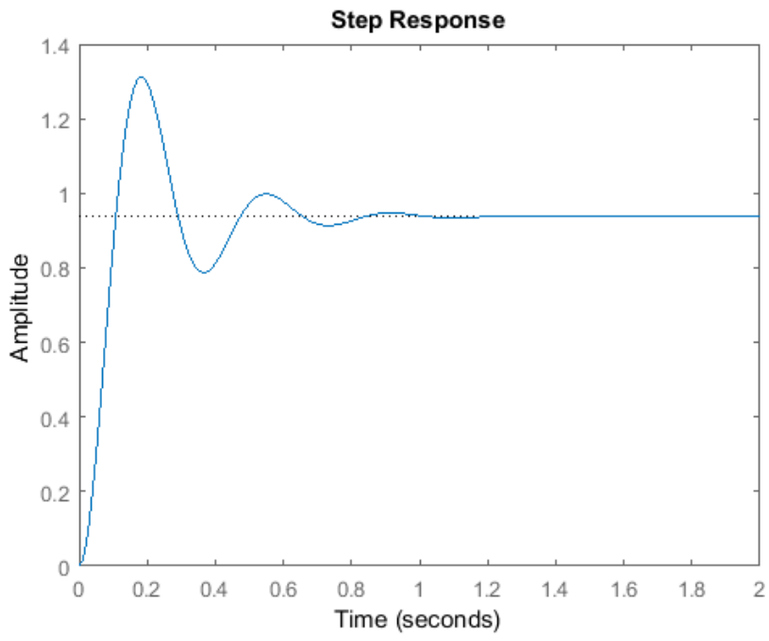


PD

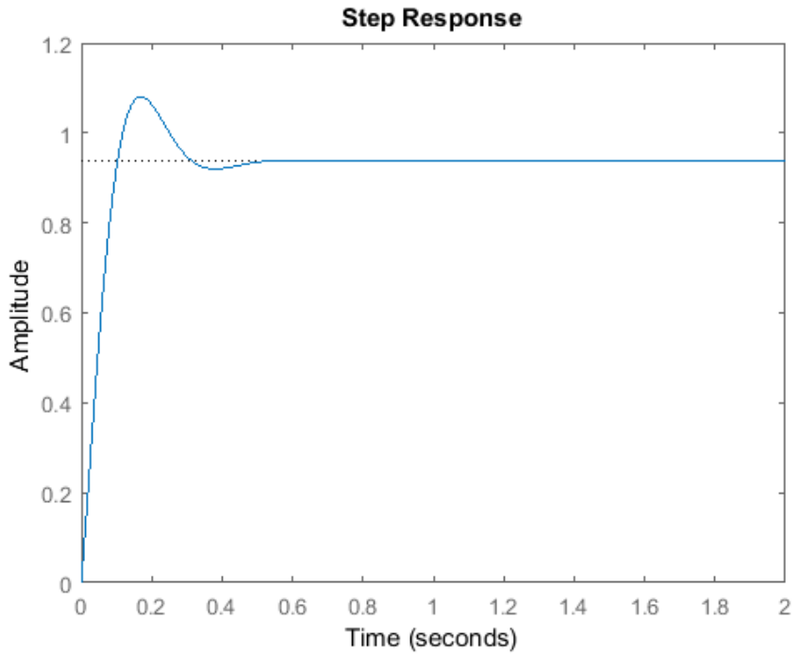


PI

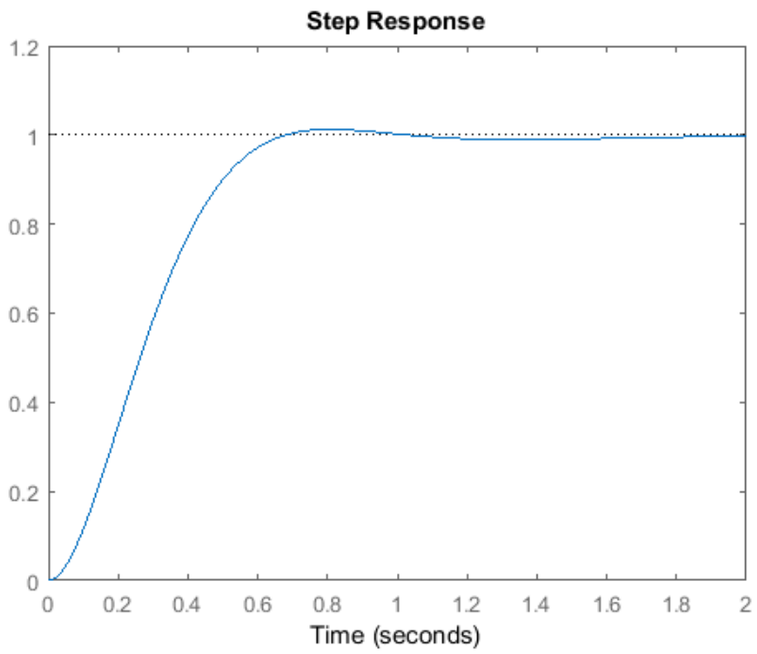
PROPORTIONAL – INTEGRAL CONTROL



P



PD



PI

- The above plots show that the addition of the integration term reduced the rise time and increased the overshoot as the proportional controller does (double effect) and eliminated the steady-state error in this case.

PROPORTIONAL – INTEGRAL – DERIVATIVE CONTROL

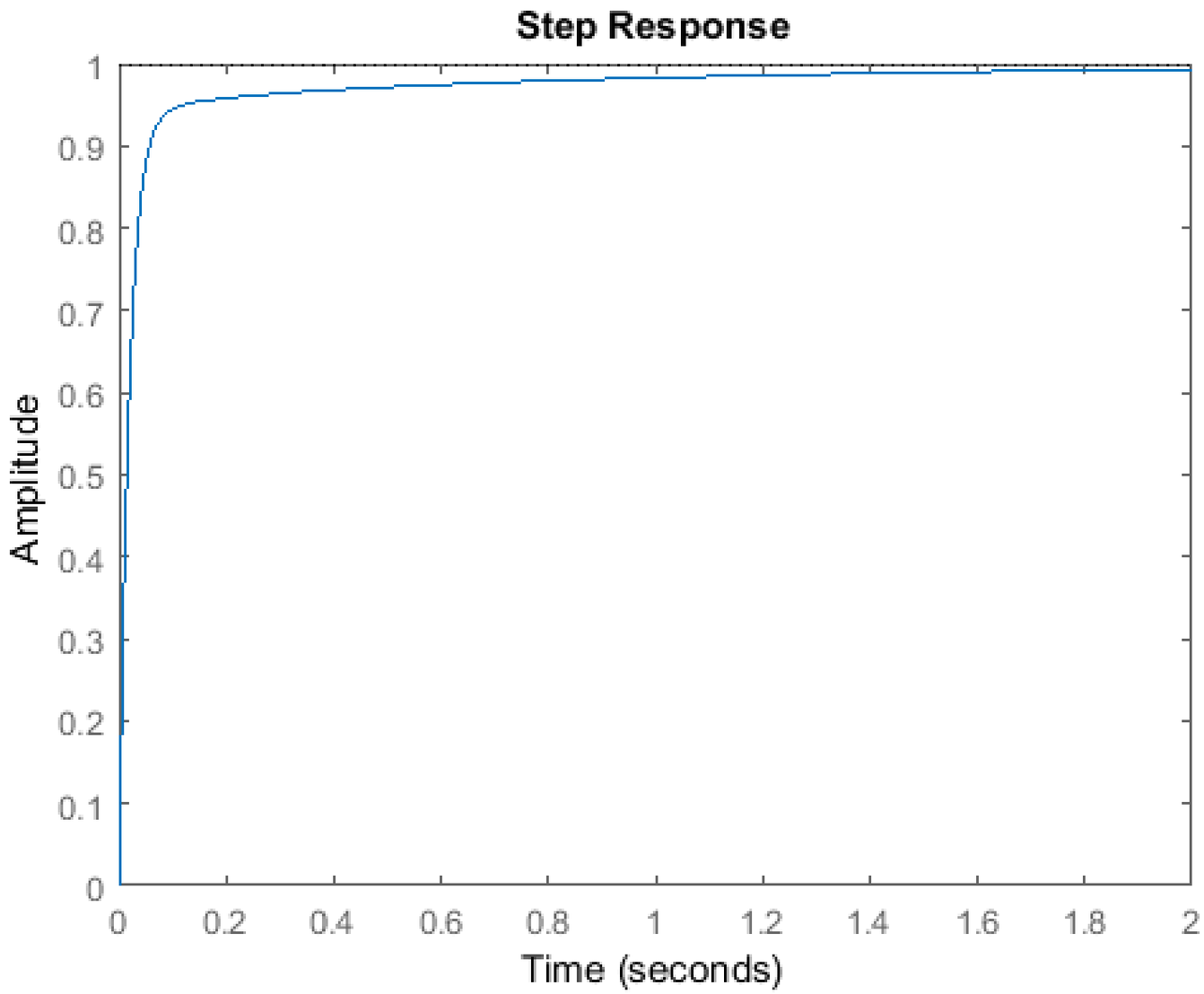
- The closed-loop transfer functions of the given system with a PID controller is :

$$T(s) = \frac{X(s)}{R(s)} = \frac{K_d s^2 + K_p s + K_i}{s^3 + (10 + K_d)s^2 + (20 + K_p)s + K_i}$$

```
Kp = 350;
Ki = 300;
Kd = 50;
C = pid(Kp,Ki,Kd)
T = feedback(C*P,1);

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

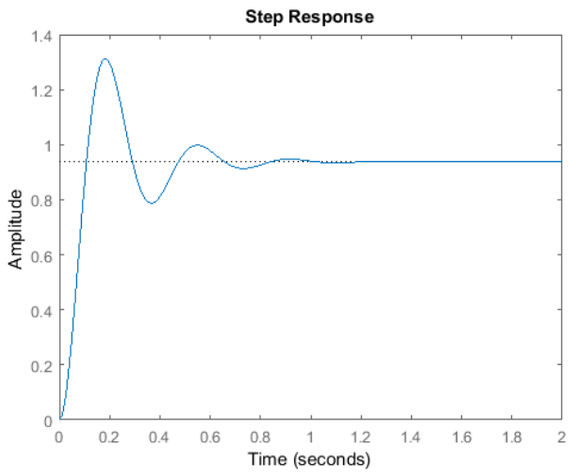
PROPORTIONAL – INTEGRAL – DERIVATIVE CONTROL

```
C =

                1
  Kp + Ki * --- + Kd * s
                s

  with Kp = 350, Ki = 300, Kd = 50

Continuous-time PID controller in parallel form.
```

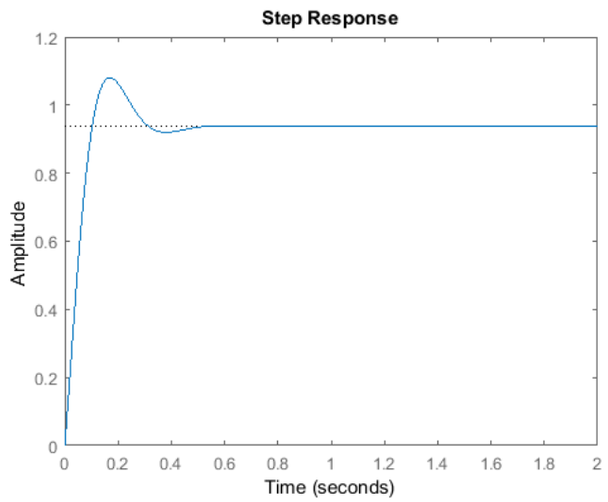


- Now, we have designed a closed-loop system with no overshoot,fast rise time, and no steady-state error.

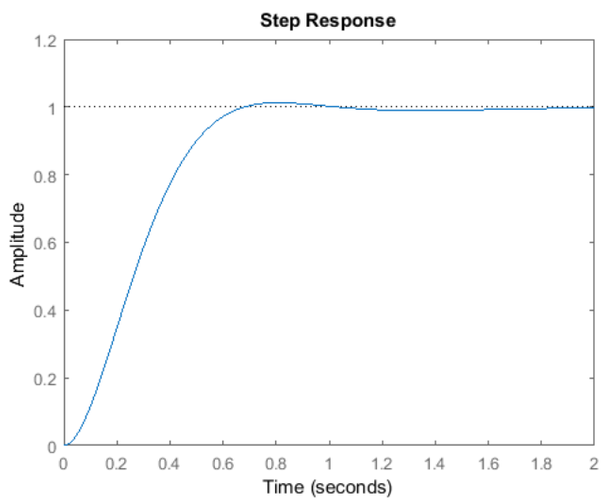
THE CHARACTERISTICS OF THE P, I AND D TERMS



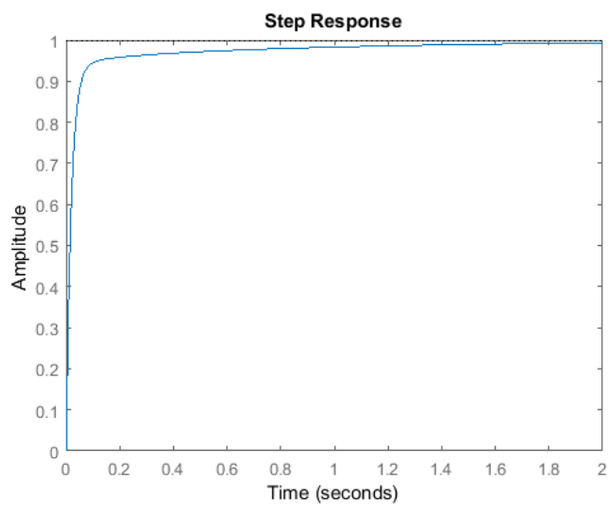
P



PD



PI



PID

CL RESPONSE	RISE TIME	OVERSHOOT	SETTLING TIME	S-S ERROR
Kp	Decrease	Increase	Small Change	Decrease
Ki	Decrease	Increase	Increase	Decrease
Kd	Small Change	Decrease	Decrease	No Change

HOMEWORK 5.1

MATLAB provides tools for automatically choosing optimal PID gains which makes the trial and error process. You can access the tuning algorithm directly using **pidtune** or through a nice graphical user interface (GUI) using **pidTuner**

- Choose a dynamic system and design 3 controllers (PI, PD, PID) using **pidTuner**.
- Explain how the pidTuner GUI is used step by step with figures.
- Explaing how the system was effected by controllers.
- Show the overshoot, settling time and final value of the system without a controller and with a controller on figures