Applications of Controllers and Compensators

XMUT315 Control Systems Engineering

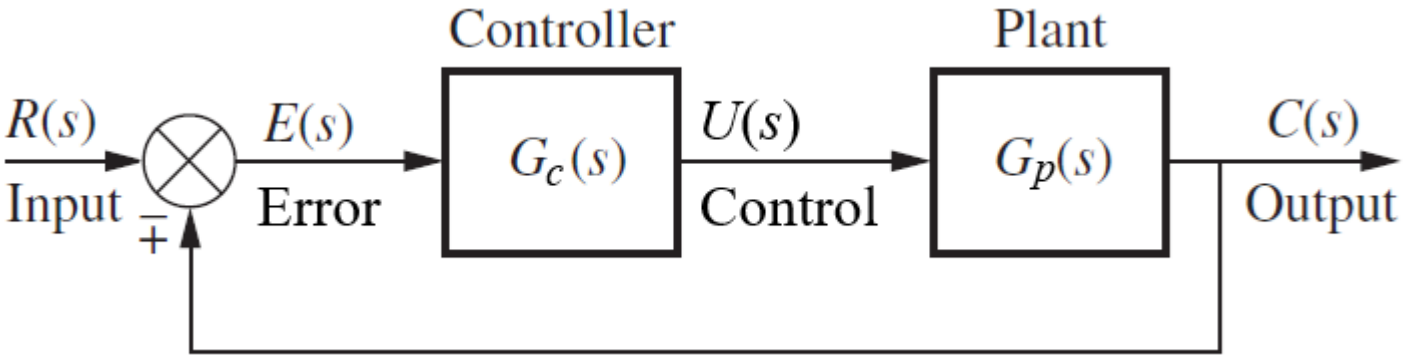
Topics

- Applications of controllers or compensators.
- Examples of applications of Proportional, Derivative, Integral controllers and their combinations.
- Examples of applications of Lag, Lead, and Lag-lead compensators.
- Practical circuit implementations of controllers or compensators.
- Tuning in of the controllers.

Applications of Controllers

- We will consider the following unity-feedback system.
- The output of the controller (u), which is equal to the control input to the plant, is calculated in the time domain from the feedback error (e) as follows:

$$u(t) = G_c(t)e(t)$$



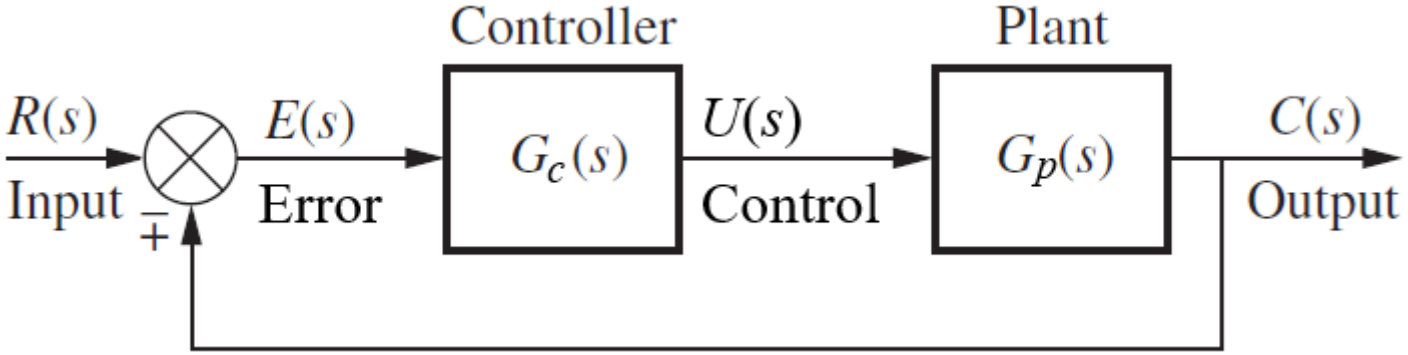
- First, let's take a look at how the controller works in a closed-loop system using the block diagram shown above.

Applications of Controllers

- The variable (e) represents the tracking error, the difference between the desired output (r) and the actual output (c).
- This error signal (e) is fed to the controller, and the controller computes this error-signal parameters of the controller with respect to time.
- Depending on the type of controller, these parameters are:
  - K_p for proportional controller,
  - K_i/s for integral controller,
  - $K_d s$ for derivative controller, or
  - any of their combinations such as: $K_p + K_i/s$ for PI controller, $K_p + K_d s$ for PD controller, and $K_p + K_i/s + K_d s$ for PID controller.

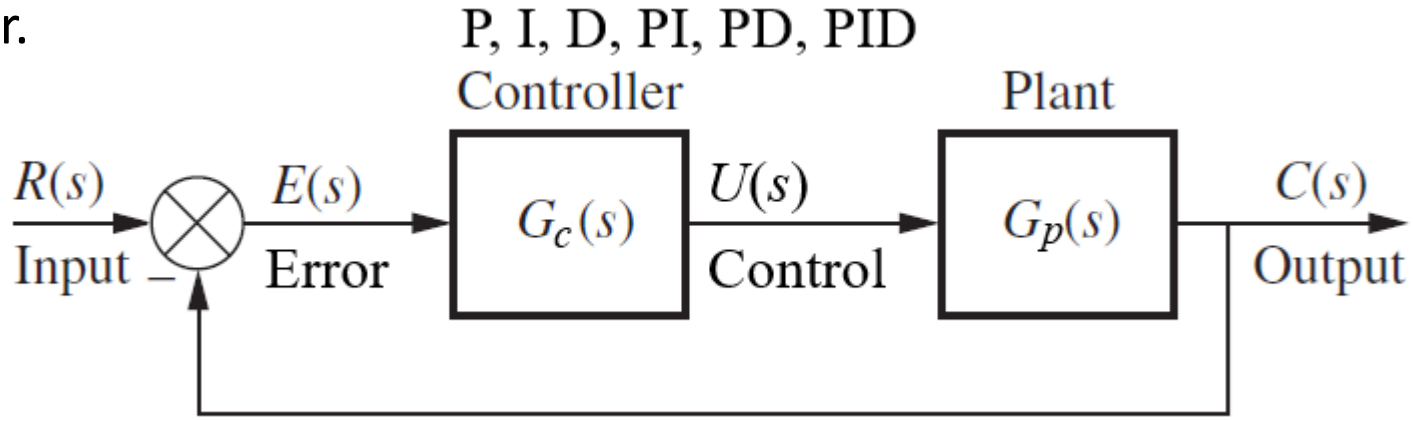
Applications of Controllers

- The control signal (u) to the plant is equal to the error (e) times the magnitude of the parameters of the controller (G_c).
- This control signal (u) is fed to the plant (G_p), and the new output (c) is obtained.
- The new output (c) is then fed back and compared to the reference (r) to find the 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.



Example System for Controller

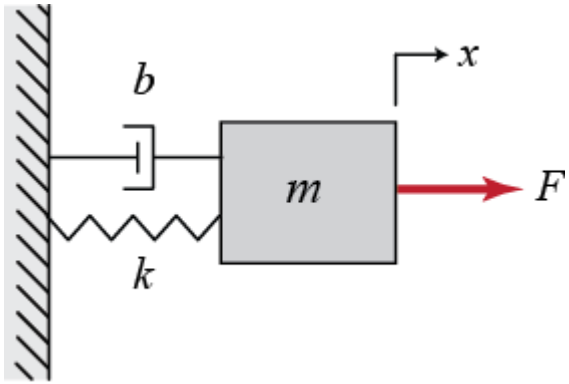
- The goal of examples of controller applications is to show how each of the terms: K_p, K_i, and K_d, contributes to obtaining the common goals of fast rise time, minimal overshoot, and zero steady-state error.



- The goals listed above are commonly found in the design specification of the control systems.
- We might have other (more specific) design specification depending on the needs and requirements.

Example System for Controller

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



The governing equation of this system is:

$$m\left(\frac{d^2x}{dx^2}\right) + b\left(\frac{dx}{dt}\right) + kx = F$$

a. Derive the transfer function of the system. [2 marks]

b. If m = 1 kg, b = 10 N s/m, k = 20 N/m, and F = 1 N, determine the transfer function of the system. [2 marks]

Example System for Controller

c. Simulate the step response of the open-loop system in MATLAB. [4 marks]

d. Describe the result of the simulation in part (c) in terms of DC gain and steady-state error, rise time and settling time. What are the characteristics of the controller or compensator needed to fix the problems? [4 marks]

Example System for Controller

a. Taking the Laplace transform of the transfer function equation of the system, we get:

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

The transfer function equation between the input force and the output displacement then becomes:

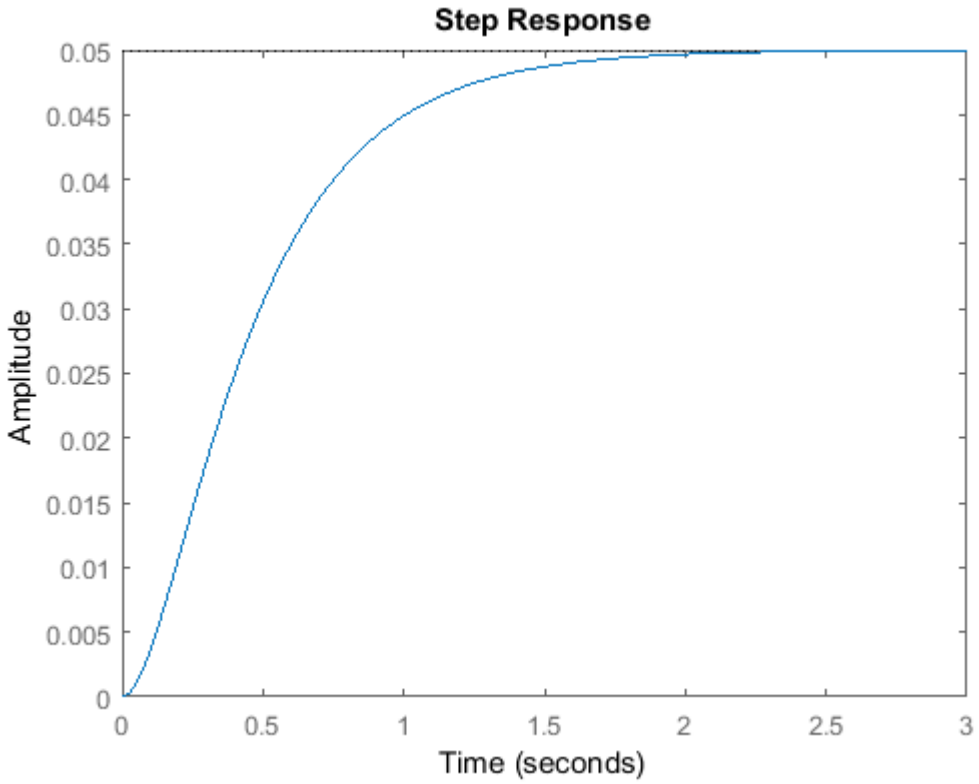
$$\frac{X(s)}{F(s)} = \frac{1}{ms^2 + bs + k}$$

b. Let: m = 1 kg, b = 10 N s/m, k = 20 N/m, and F = 1 N. Substituting these values into the above transfer function equation:

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

Example System for Controller

c. Let's first view the open-loop step response of the system in MATLAB as shown below.



Example System for Controller

d. 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.

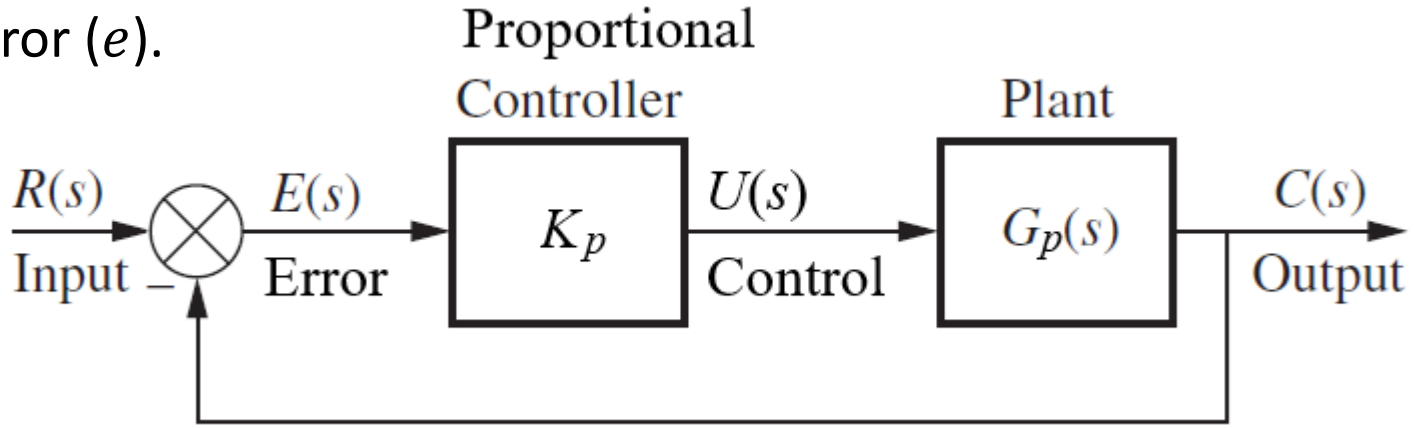
This corresponds to a steady-state error of 0.95, which is quite large.

Furthermore, the rise time is about 1 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 eliminate the steady-state error.

P Controllers

- For a proportional controller, the control signal (u) to the plant is equal to the proportional gain (K_p) times the magnitude of the error (e).



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

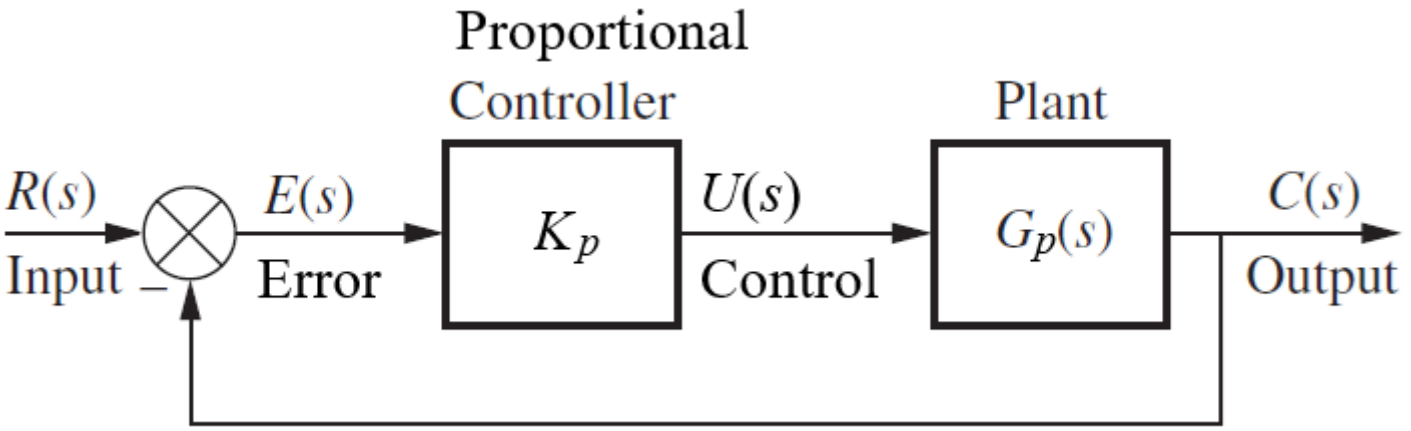
$$u(t) = K_p e(t)$$

P Controllers

- Thus, the transfer function of a proportional controller is found by taking the Laplace transform of the system equation:

$$G_c(s) = K_p \qquad \text{Where: } K_p \text{ = proportional gain.}$$

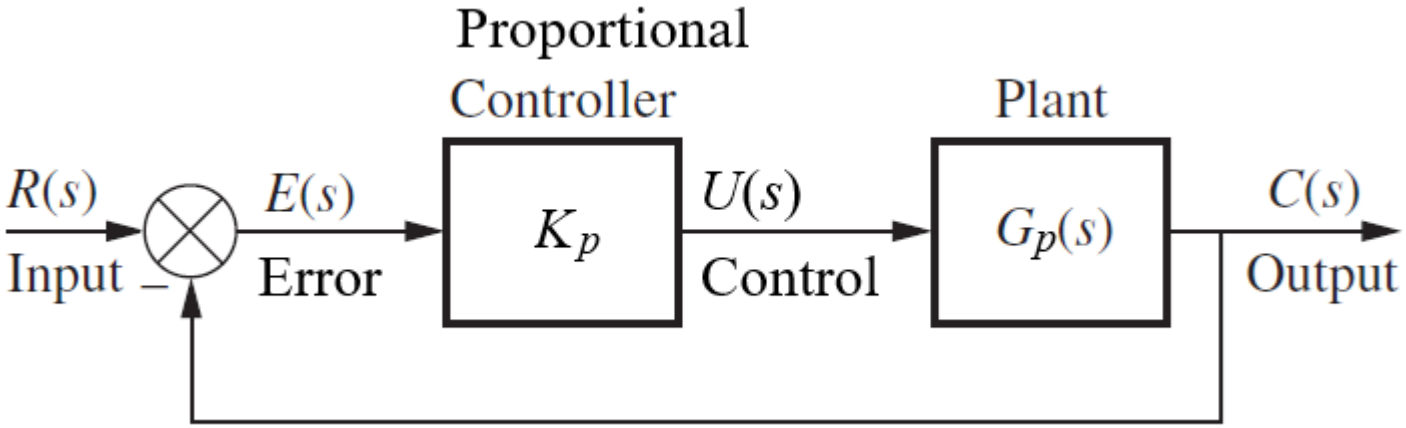
Increasing the proportional gain (K_p) has the effect of proportionally increasing the control signal for the same level of error.



- The fact that the controller will "push" harder for a given level of error tends to cause the closed-loop system to react more quickly, but also to overshoot more.

Proportional Controller

- Another effect of increasing K_p is that it tends to reduce, but not eliminate, the steady-state error.

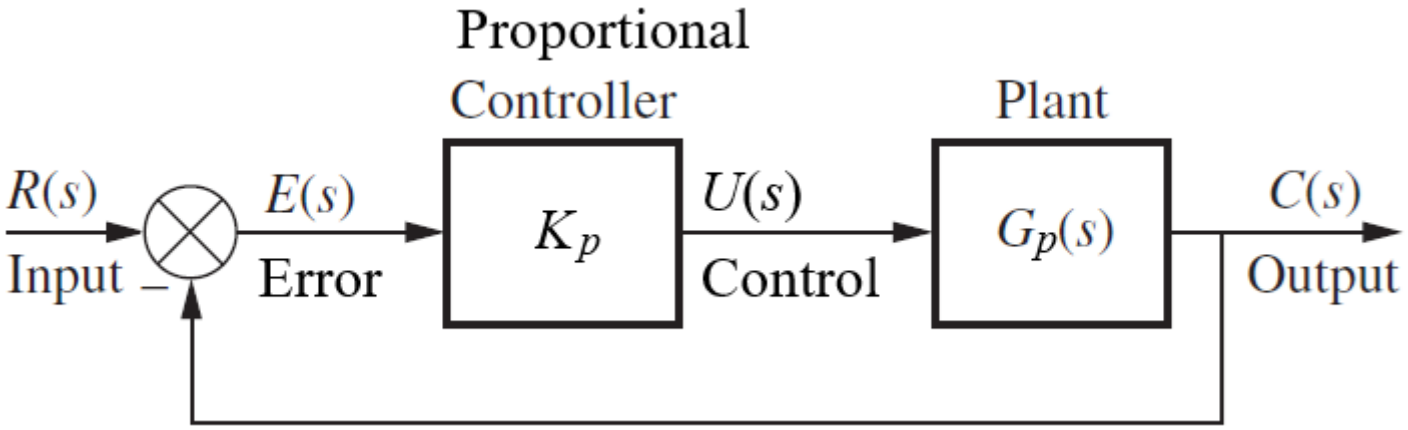


- Let's talk about the characteristics of the proportional controller.
- The proportional controller (K_p) reduces the rise time, increases the overshoot, and reduces the steady-state error.
- Transfer function equation of the proportional controller is:

$$G_c = K_p$$

Example for Proportional Controller

For the given simple mass-spring-damper system in the previous example, add a proportional controller in series with the system.



a. Derive the transfer-function equation of the system. [4 marks]

b. Using the trial-and-error method in MATLAB, determine the appropriate value of the parameter of the controller. Then, simulate the transient response of the system. [6 marks]

c. Describe the response of the system in terms of rise time, overshoot, settling time, and steady-state error. [2 marks]

Example for Proportional Controller

a. The transfer function equation of a proportional controller is:

$$G_c(s) = K_p$$

The closed-loop transfer function of unity-feedback system with a proportional controller is as follow:

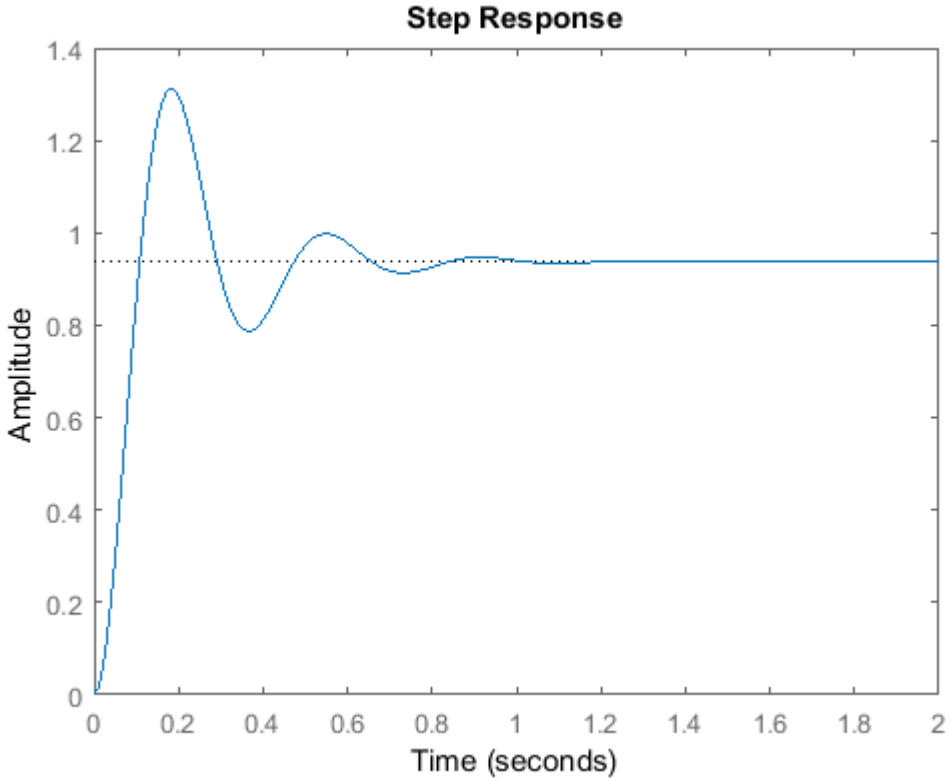
$$T(s) = \frac{C(s)}{R(s)} = \frac{G_c G_p}{1 + G_C G_p}$$

$$= \frac{K_p\left(\frac{1}{s^2 + 10s + 20}\right)}{1 + K_p\left(\frac{1}{s^2 + 10s + 20}\right)} = \frac{K_p}{s^2 + 10s + \left(20 + K_p\right)}$$

Where: C(s) is our output and our reference $R(s)$ is the input.

Example for Proportional Controller

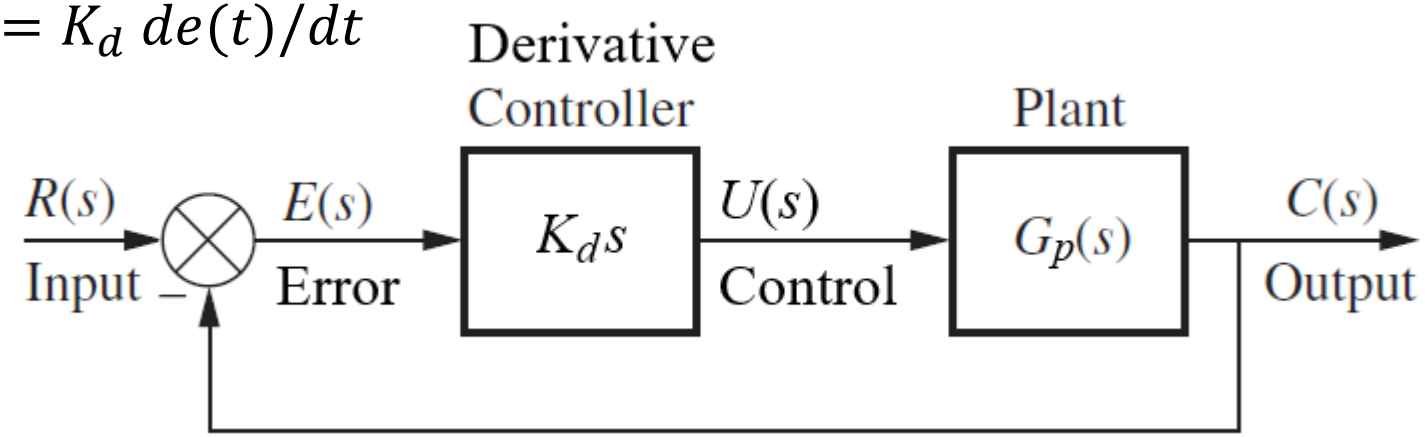
b. Let the proportional gain (K_p) equal 300. The following figure shows the step response of the example system with a proportional controller.



c. The 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.

D Controller

- For a derivative controller, the control signal (u) to the plant is equal to the derivative gain (K_d) times the derivative of the error.
- The output of a derivative controller, which is equal to the control input to the plant, is calculated in the time domain from the feedback error as $u(t) = K_d\, de(t)/dt$



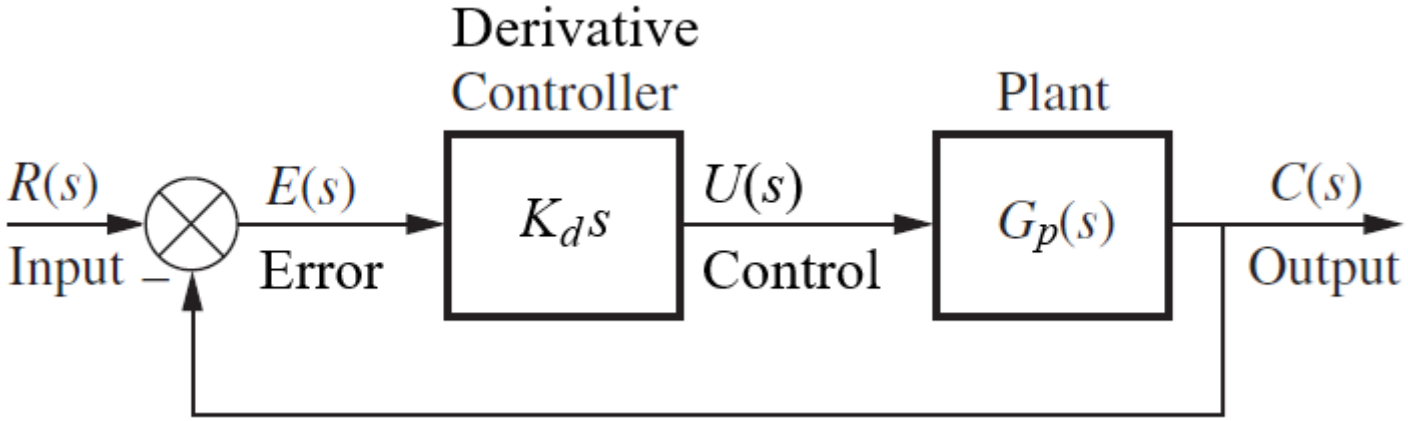
- Thus, the transfer function of a derivative controller is found by taking the Laplace transform of the system equation:

$$G_c(s) = K_d s \qquad \text{Where: } K_d \text{ = derivative gain}$$

- The addition of a derivative term to the controller (K_d) adds the ability of the controller to "anticipate" error.

D Controller

- With simple proportional control, if K_p is fixed, the only way that the control will increase is if the error increases.

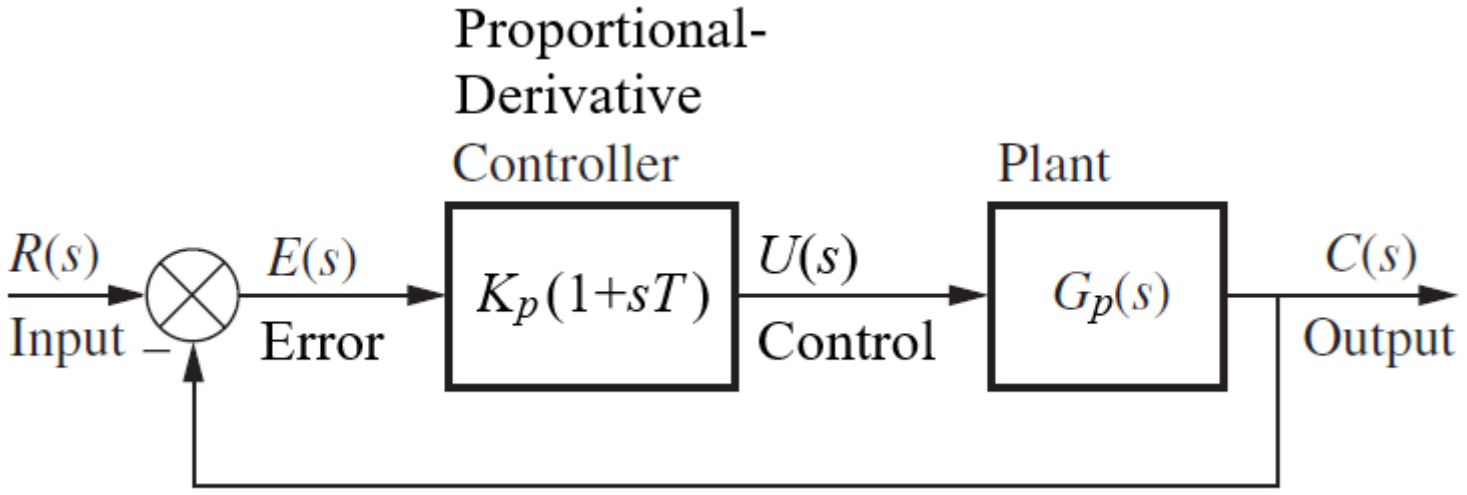


- With derivative control, the control signal can become large if the error begins sloping upward, even while the magnitude of the error is still relatively small.
- This anticipation tends to add damping to the system, thereby decreasing overshoot.
- The addition of a derivative term, however, does not affect the steady-state error.

PD Controller

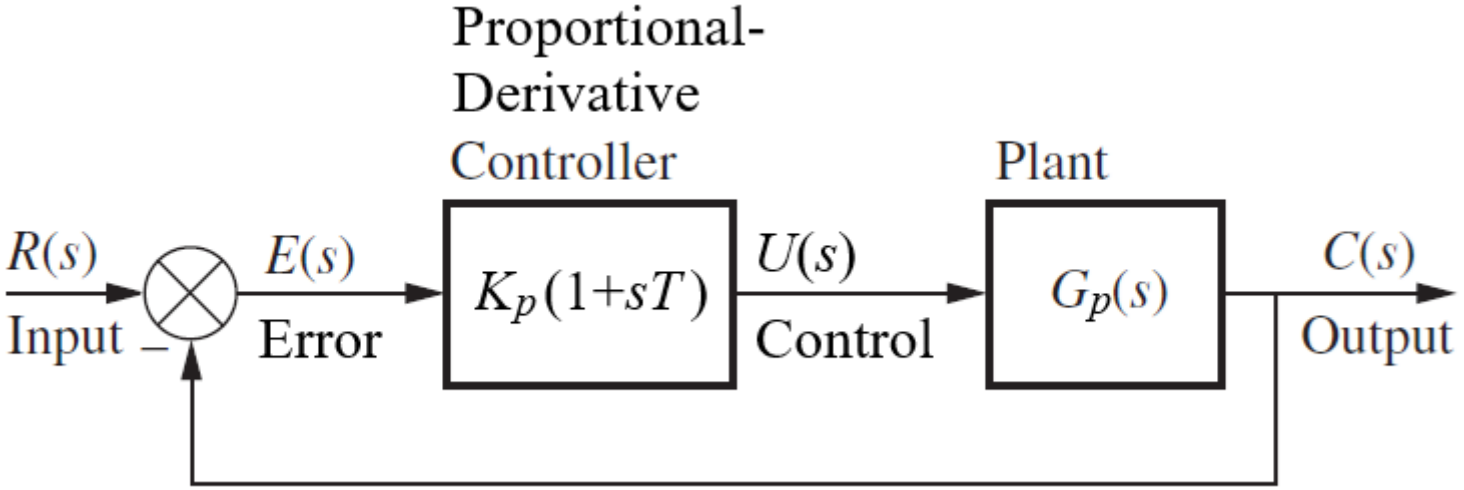
- Now, let's look at the characteristics of the PD controller.
- Addition of the proportional-derivative control ($K_d(1 + sT_d)$) tends to reduce both the overshoot and the settling time.
- Transfer function equation of the PD controller is:

$$G_c = K_p(1 + sT_d)$$



Example for PD Controller

For the simple mass-spring-damper system in the previous example, add a proportional-derivative controller in series with the system.



a. Derive the transfer function equation of the system. [4 marks]

b. Using the trial-and-error method in MATLAB, determine the appropriate values of the parameters of the controller. Then, simulate the transient response of the system. [6 marks]

c. Describe the response of the system in terms of rise time, overshoot, settling time and steady-state error. [2 marks]

Example for PD Controller

a. The transfer function of the PD controller is (note: $K_d = K_p T_d$):

$$G_c(s) = K_d s + K_p$$

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

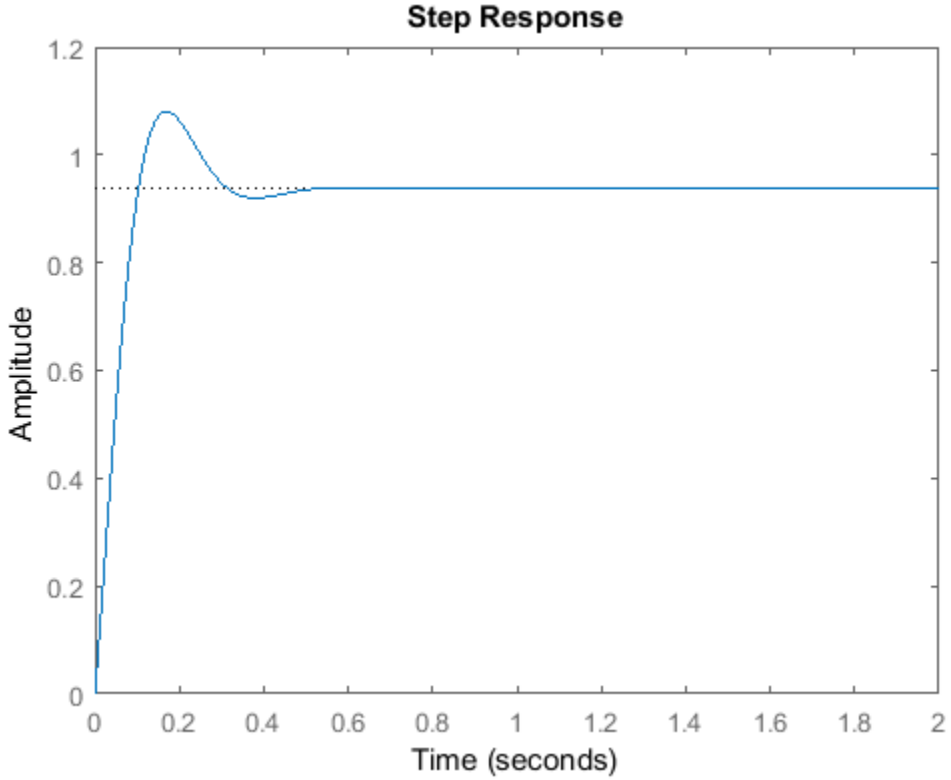
$$T(s) = \frac{C(s)}{R(s)} = \frac{G_c(s)G_p(s)}{1 + G_c(s)G_p(s)}$$

$$= \frac{(K_d s + K_p)\left(\frac{1}{s^2 + 10s + 20}\right)}{1 + (K_d s + K_p)\left(\frac{1}{s^2 + 10s + 20}\right)}$$

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

Example for PD Controller

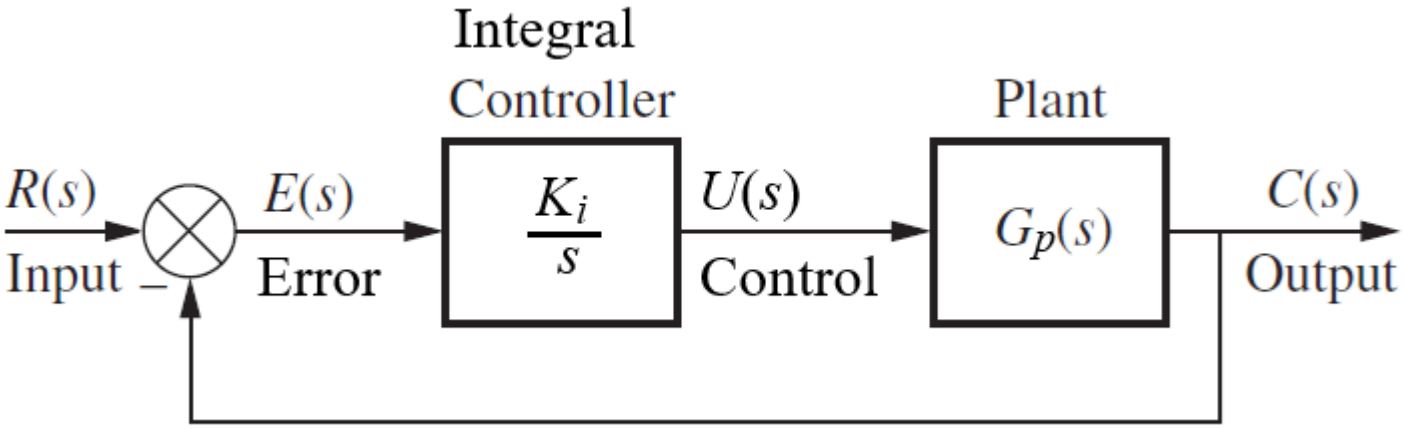
b. Let K_p equal 300 as before and let K_d equal 10. The simulation in MATLAB is shown below.



c. This plot shows 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.

I Controllers

- For an integral controller, the control signal (u) to the plant is equal to the integral gain (K_i) times the integral of the error.
- The output of an integral controller, which is equal to the control input to the plant, is calculated in the time domain from the feedback error as $u(t) = K_i \int e(t)dt$.

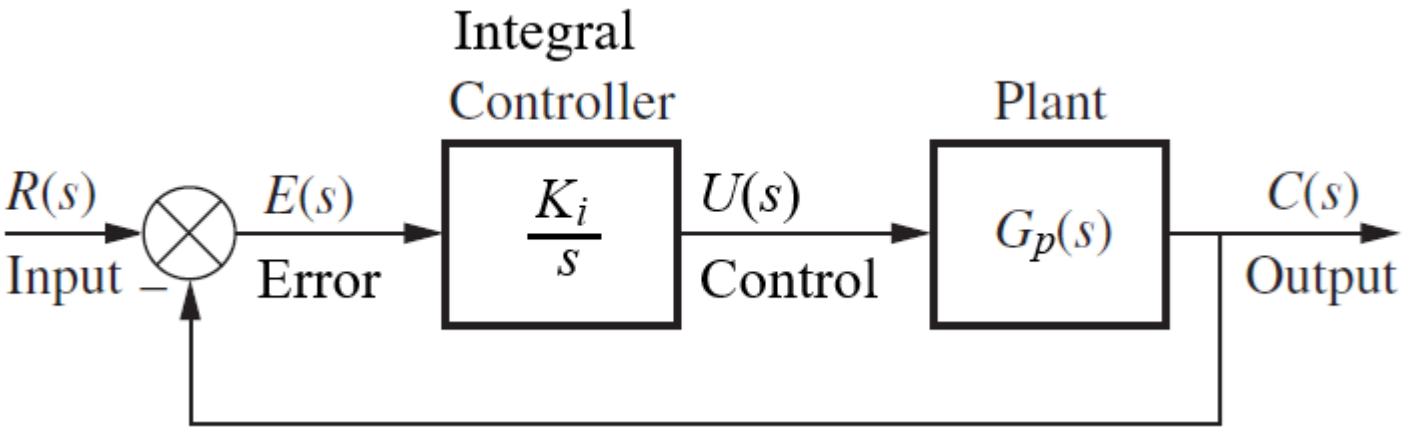


- Thus, the transfer function of an integral controller is found by taking the Laplace transform of the system equation:

$$G_c(s) = K_i/s \quad \text{Where: } K_i \text{ = integral gain.}$$

I Controllers

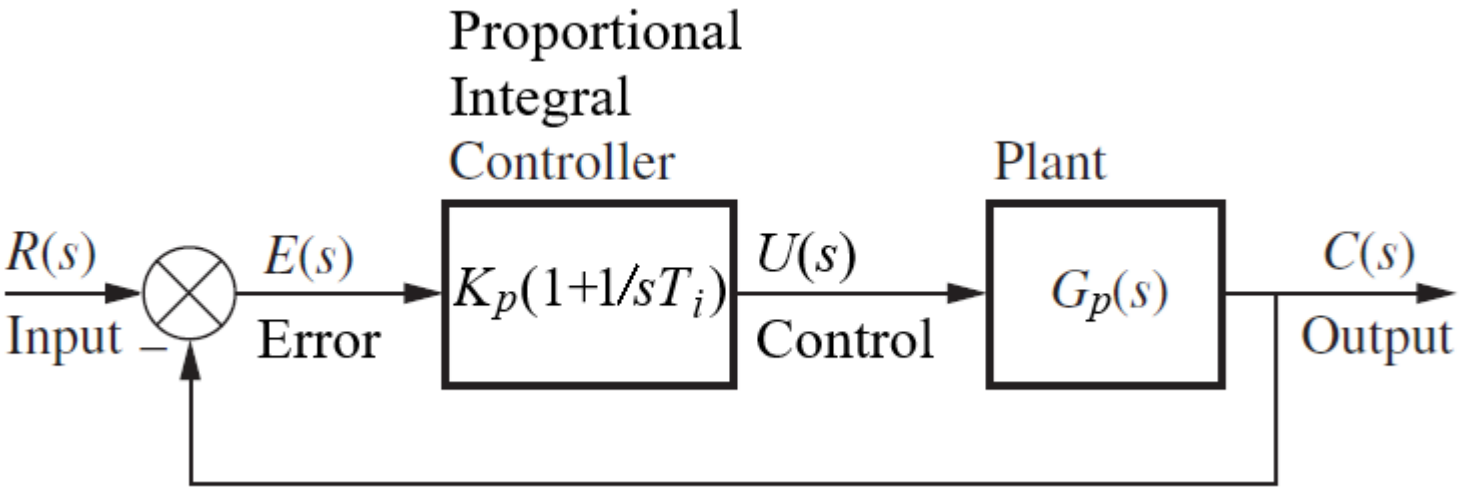
- The addition of an integral term to the controller (K_i) tends to help reduce the steady-state error.
- If there is a persistent, steady-state error, the integrator builds and builds, thereby increasing the control signal and driving the error down.
- A drawback of the integral term, however, is that it can make the system more sluggish (and oscillatory) since when the error signal changes sign, it may take a while for the integrator to "unwind."



PI Controller

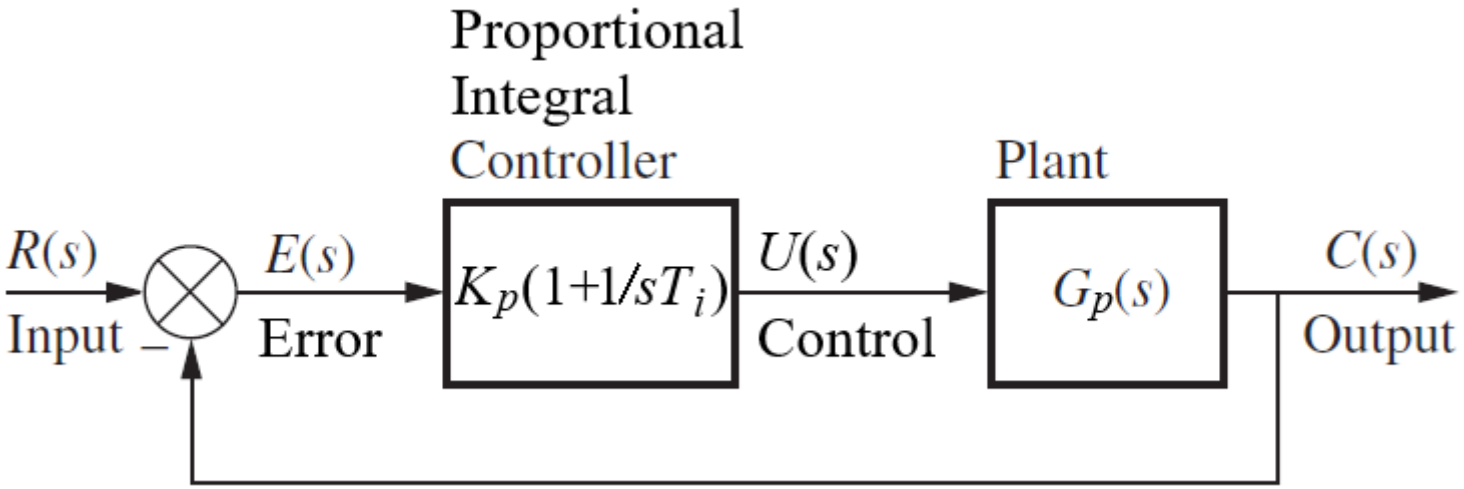
- Let's investigate the characteristics of the PI controller.
- Addition of proportional-integral control ($K_p(1 + 1/sT_i)$) tends to decrease the rise time, increase both the overshoot and the settling time, and reduce the steady-state error.
- The transfer-function equation of PI controller is:

$$G_c(s) = K_p\left(1 + \frac{1}{sT_i}\right)$$



Example for PI Controller

For the simple mass-spring-damper system in the previous example, add a proportional-integral controller in series with the system.



a. Derive the transfer function equation of the system. [4 marks]

b. Using the trial-and-error methods in MATLAB, determine the appropriate values of the parameters of the controller. Then, simulate the transient response of the system. [6 marks]

c. Describe the response of the system in terms of rise time, overshoot, settling time and steady-state error. [2 marks]

Example for PI Controller

a. The transfer function of the PI controller is (note: $K_i = K_p/T_i$):

$$G_c(s) = K_i/s + K_p$$

For the given system, the closed-loop transfer function with a PI controller is:

$$T(s) = \frac{C(s)}{R(s)} = \frac{G_c(s)G_p(s)}{1 + G_c(s)G_p(s)}$$

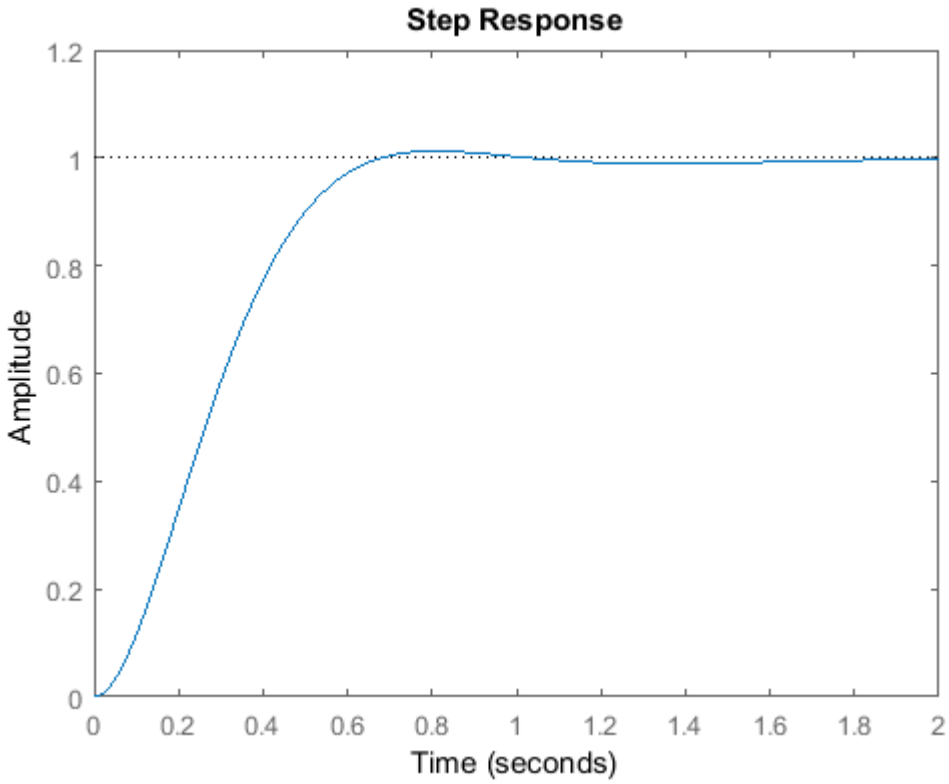
$$= \frac{\left(K_i/s + K_p\right)\left(\frac{1}{s^2 + 10s + 20}\right)}{1 + \left(K_i/s + K_p\right)\left(\frac{1}{s^2 + 10s + 20}\right)}$$

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

Example for PI Controller

b. Let's reduce K_p to 30 and let K_i equal 70. It's shown below.

c. We have reduced the proportional gain (K_p) 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 eliminates the steady-state error in this case.

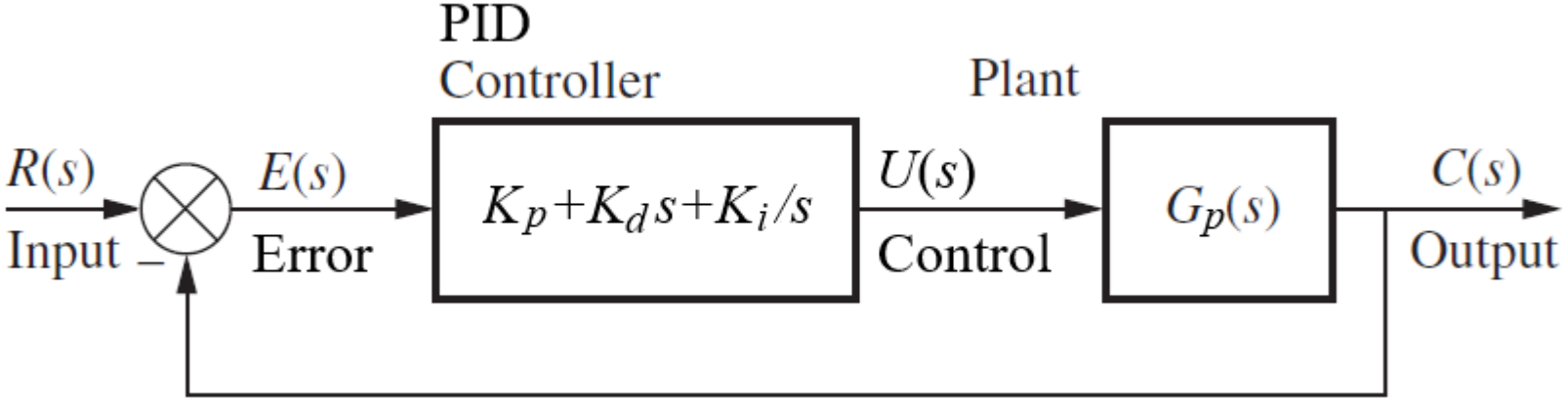
PID Controllers

- For a PID controller, 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:

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

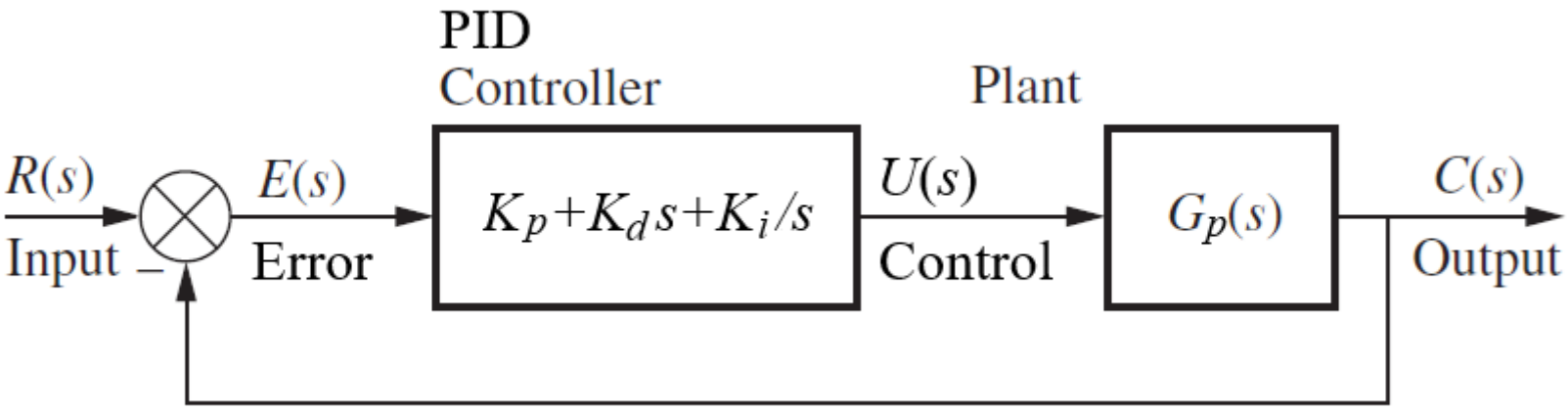
- The transfer function of a PID controller is found by taking the Laplace transform of system equation: Where: K_p = proportional gain, K_i = integral gain, and K_d = derivative gain.

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



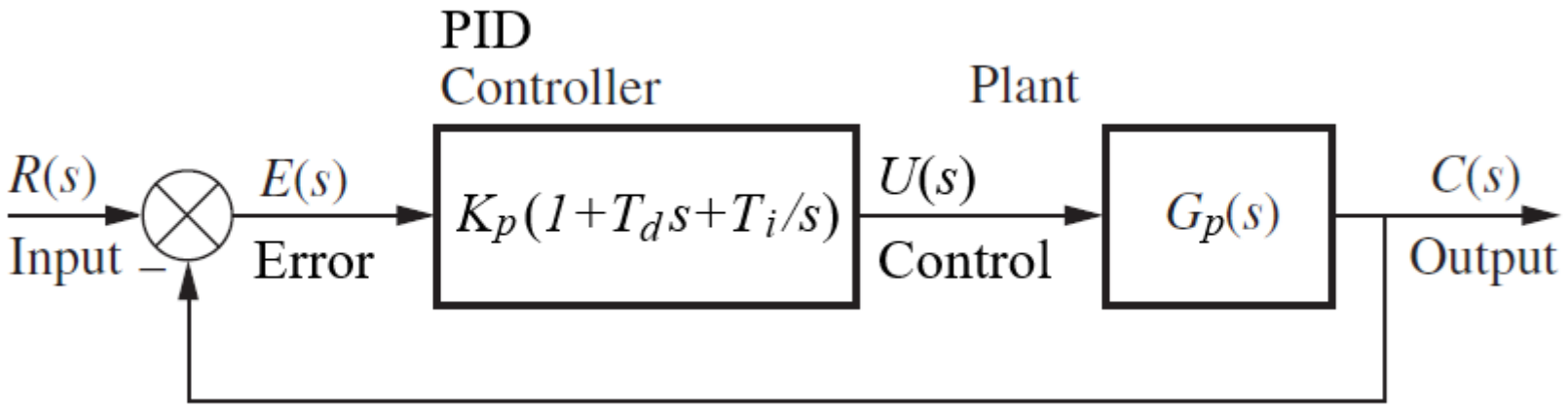
PID Controller

- Now, let's examine the characteristics of the PID control.
- PID controller tends to combine the characteristics of PI and PD controller.
- So, it is capable for improving both the transient response and steady-state characteristics of the system.



Example for PID Controller

For the simple mass-spring-damper system in the previous example, add a proportional-integral-derivative controller in series with the system.



a. Derive the transfer function equation of the system. [4 marks]

b. Using the trial-and-error methods in MATLAB, determine the appropriate values of the parameters of the controller. Then, simulate the transient response of the system. [6 marks]

c. Describe the response of the system in terms of rise time, overshoot, settling time and steady-state error. [2 marks]

Example for PID Controller

a. Now, let's examine PID control. The transfer function of the PID controller is (note: $K_d = K_p T_d$ and $K_i = K_p/T_i$):

$$G_c(s) = K_d s + K_i/s + K_p$$

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

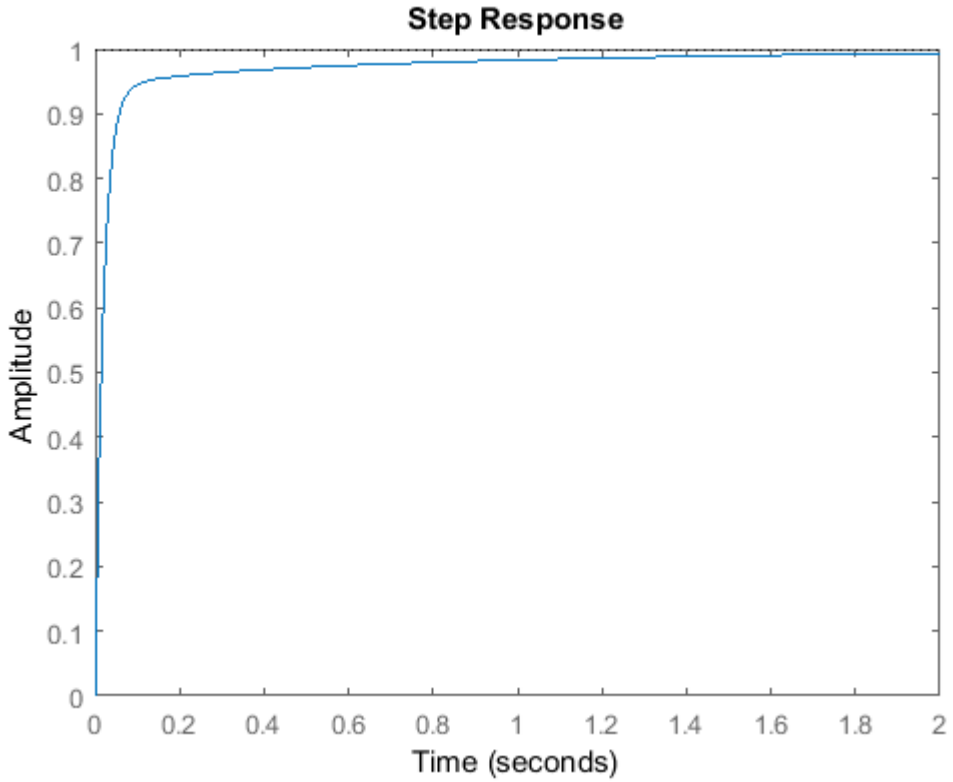
$$T(s) = \frac{X(s)}{R(s)} = \frac{G_c(s)G_p(s)}{1 + G_c(s)G_p(s)}$$

$$= \frac{\left(K_d s + K_i/s + K_p\right)\left(\frac{1}{s^2 + 10s + 20}\right)}{1 + \left(K_d s + K_i/s + K_p\right)\left(\frac{1}{s^2 + 10s + 20}\right)}$$

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

Example for PID Controller

b. After several iterations of tuning, the gains K_p = 350, K_i = 300, and K_d = 50 provided the desired response. The simulation in MATLAB is shown below.



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

Summary of Controllers

- Summary of controllers:

Controller	Transfer Function	Characteristics
P	K_p	Reduces the rise time, increases the overshoot, and reduces the steady-state error.
I	$\frac{K_i}{s}$	Reduces steady-state error.
D	$K_d s$	Increases the transient response responsiveness and characteristics.
PI	$K_p + K_i/s$	Decrease the rise time, increase both the overshoot and the settling time, and reduces the steady-state error.
PD	$K_p + K_d s$	Reduce both the overshoot and the settling time.
PID	$\frac{K_d s^2 + K_p s + K_i}{s}$	Improve both of the transient response and steady-state characteristics.

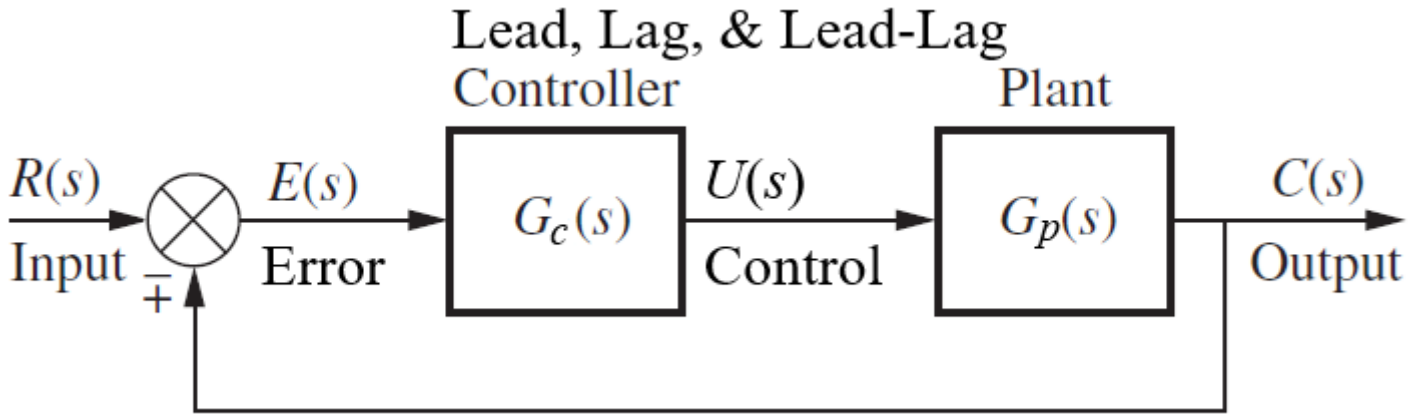
Summary of Controllers

- The general effects of each controller parameter : K_p, K_i, and K_d on a closed-loop system are summarised in the table below.
- Note, these guidelines hold in many cases, but not all.
- If you truly want to know the effect of tuning the individual gains, you will have to do more analysis or will have to perform testing on the actual system.

Controller	Rise Time	Overshoot	Settling Time	Steady-State Error
K_p	Decrease	Increase	Small Change	Decrease
K_i	Decrease	Increase	Increase	Decrease
K_d	Small Change	Decrease	Decrease	No Change

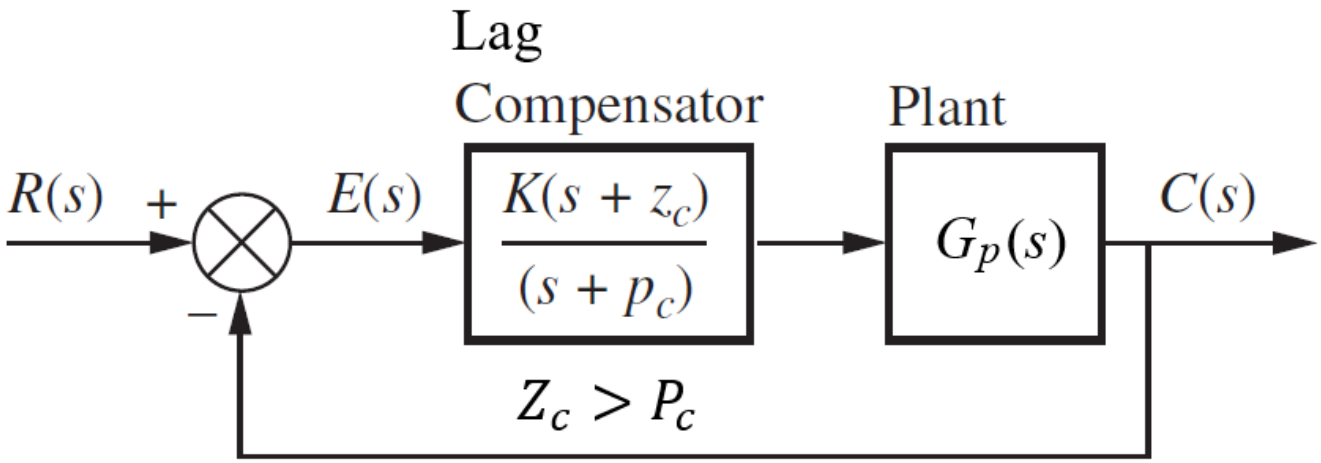
Example Systems for Compensators

- Controllers and compensators are slightly different in terms of their characteristics and purpose, their practical implementations, and their designs.
- We will be looking at some control systems with example compensators, i.e. Lead, Lag and Lead-Lag compensators.



Lag Compensator

- Lag compensator is commonly employed in the control systems to improve steady-state conditions but has little influence on the transient response of the systems.



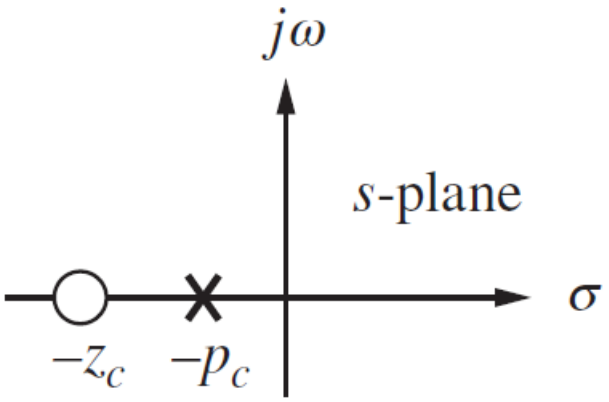
- Lag compensators reduce steady-state error, so sometimes we want smaller steady-state error rather than shorter rise and settling time as in a lead compensator.

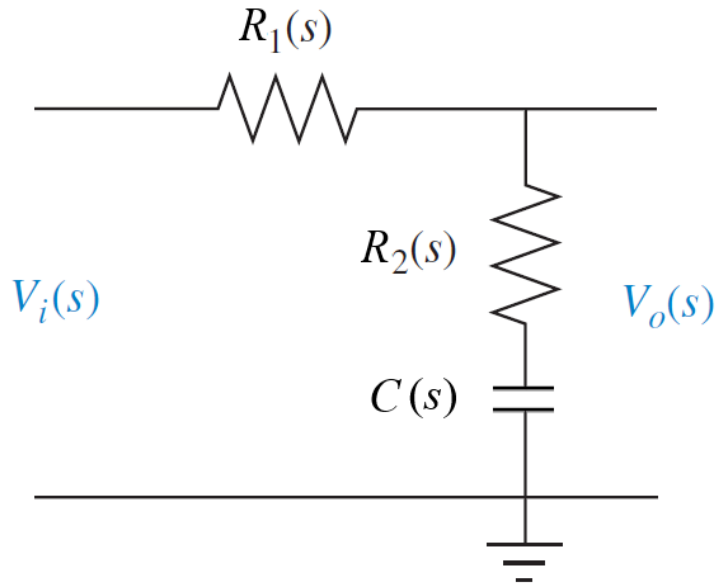
Lag Compensator

Phase-lag compensator:

- The integrator in PI controller can cause some practical problems; e.g., “integrator windup” due to actuator saturation.
- PI controller is often approximated by “lag control.”

$$G_c(s) = \frac{(s - z_0)}{(s - p_0)} \quad \text{with} \quad |p_0| < |z_0|$$



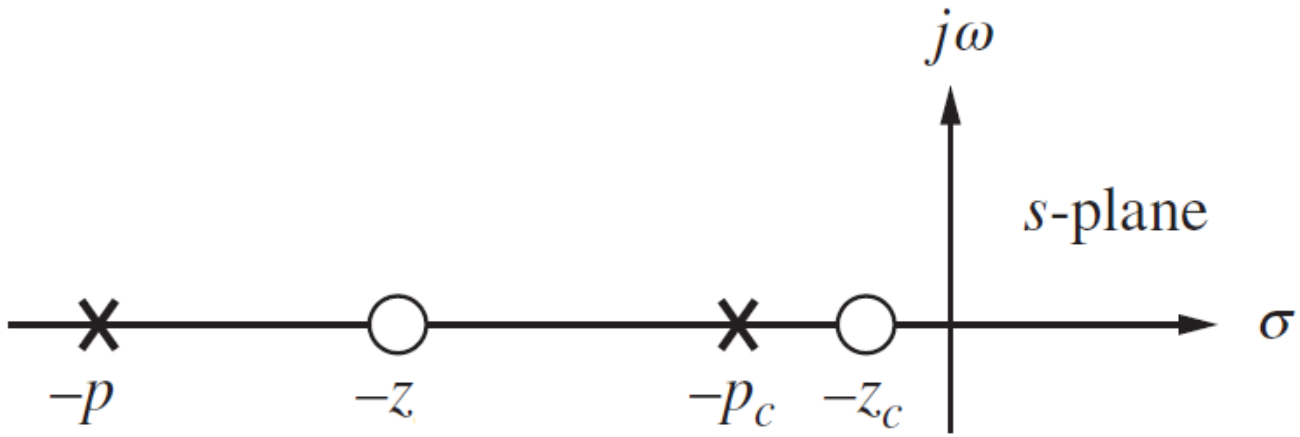


Lag Compensator

- That is, the pole is closer to the origin than the zero.
- Because $|z_0| > |p_0|$, the phase " added to the open-loop transfer function is negative. . . “phase lag”
- The pole is often placed very close to origin e.g., $p_0 \approx 0.01$ to minimise impact towards transient response performance.
- The zero is placed near pole. e.g., $z_0 \approx 0.1$.
- We want $|G_c(s)| \approx 1$ for all s to preserve transient response (and hence, have nearly the same placement of poles as for a proportional controller).
- Idea is to improve steady-state error but to modify the transient response as little as possible.

Lag Compensator

- That is, using proportional control, we have pole locations we like already, but poor steady-state error.
- So, we add a lag compensator to minimally disturb the existing good pole locations but improve steady-state error.



- Good steady-state error without overflow problems.
- Very similar to proportional control.

Lag Compensator

- The uncompensated system had loop gain:

$$K(\text{before}) = \lim_{s \to 0} G(s)$$

- The lag-compensated system has loop gain:

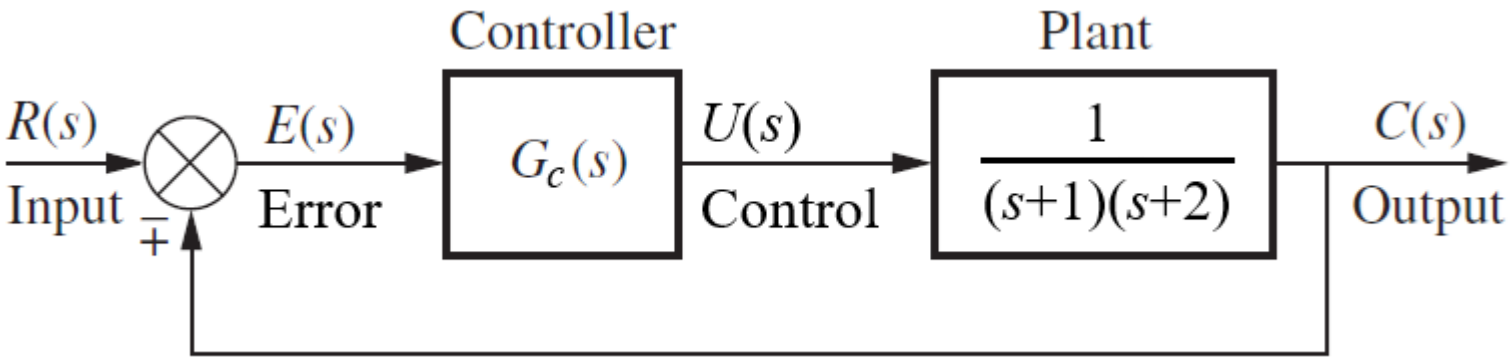
$$K(\text{after}) = \lim_{s \to 0} G_c(s)G(s) = \left(\frac{Z_0}{P_0}\right) \lim_{s \to 0} G(s)$$

- Since $|z_0| > |p_0|$, there is an improvement in the position or velocity or acceleration error constant of the system, and a reduction in steady-state error.
- Transient response is mostly unchanged, but slightly slower settling due to small-magnitude slow “tail” caused by lag compensator.

Example for Lag Compensator

When $G_C(s) = 1$, the control system given below suffers from issues in both steady-state and transient-response conditions:

- Steady-state: non-zero steady-state error.
- Transient response: sluggish system that takes time to settle down.



Example System for Lag Compensator

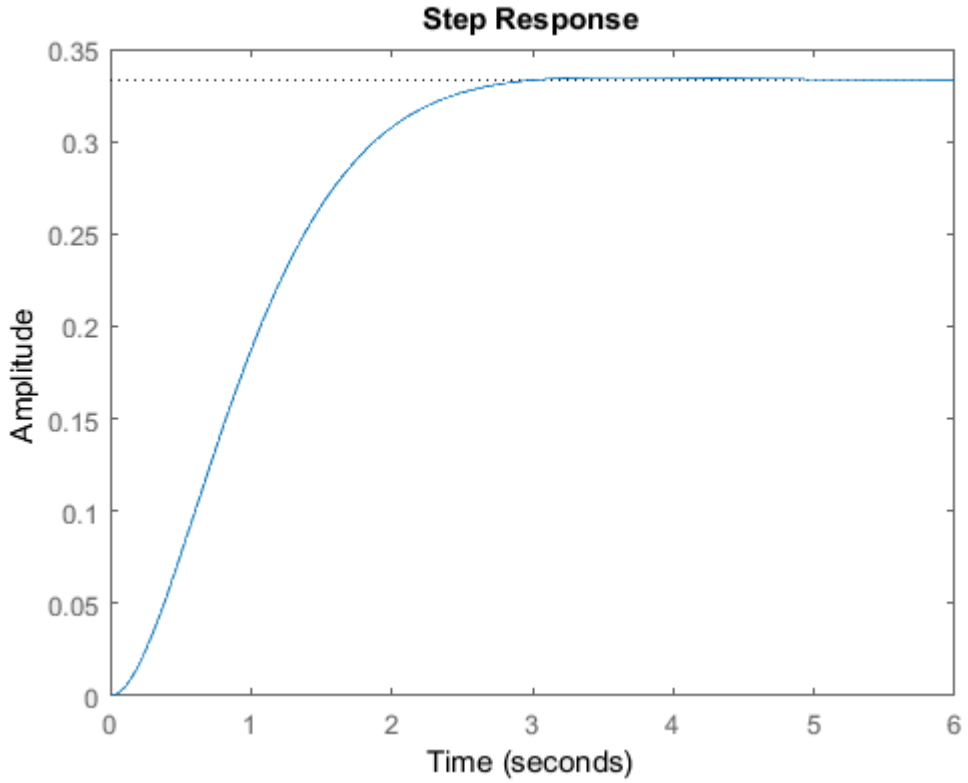
a. Simulate the uncompensated system in MATLAB. Comment on the result of the simulation. [6 marks]

b. Using the trial-and-error methods in MATLAB, design a lead compensator that will be able to fix the problem observed in part (a). [4 marks]

c. Simulate in MATLAB and compare the uncompensated and compensated systems. Observe whether the compensator has achieved its purpose. [6 marks]

Example System for Lag Compensator

a. The result of the simulation in MATLAB is shown below.

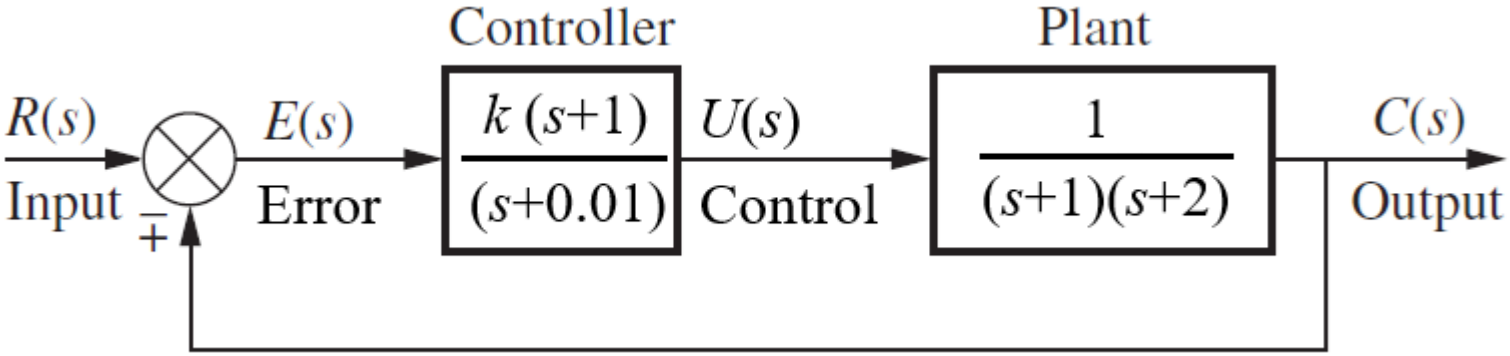


Looking into the unit step response of the given system, there are issues as highlighted before, e.g. non-zero steady-state error and slow (sluggish) response of the system.

Example for Lag Compensator

b. By trial and error, the gain and the pole and zero of the lead compensator are determined:

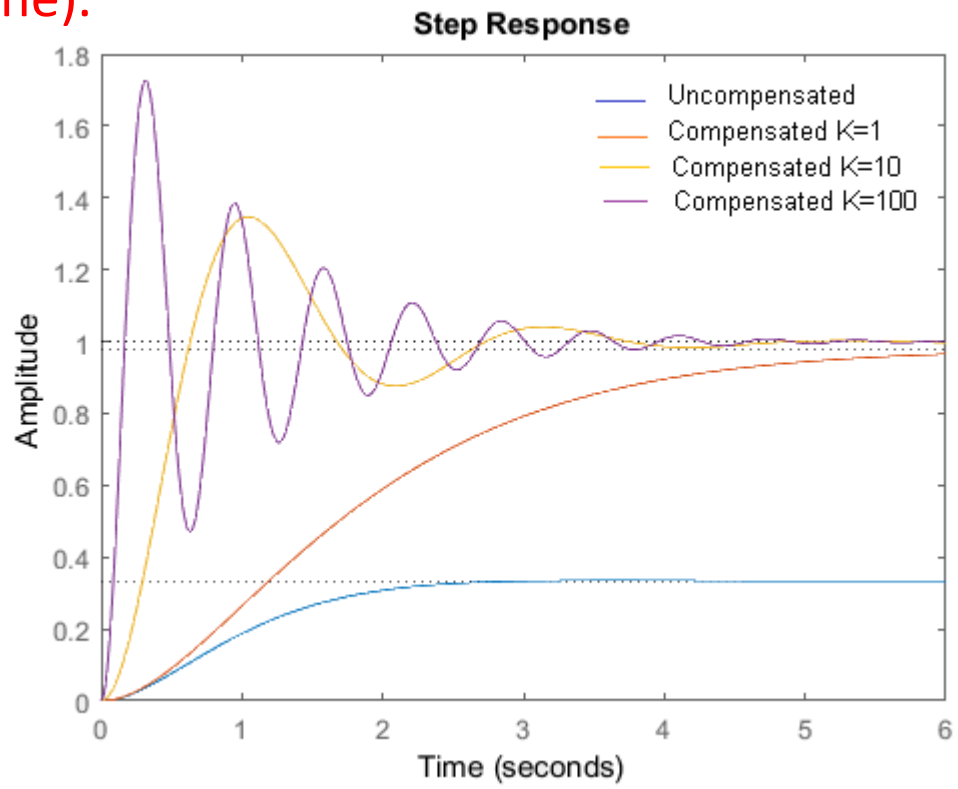
$$C(s) = \frac{K(s+1)}{s+0.01}$$



c. Looking into the response of the compensated system, the plot shows smaller steady-state error than uncompensated system

Example for Lag Compensator

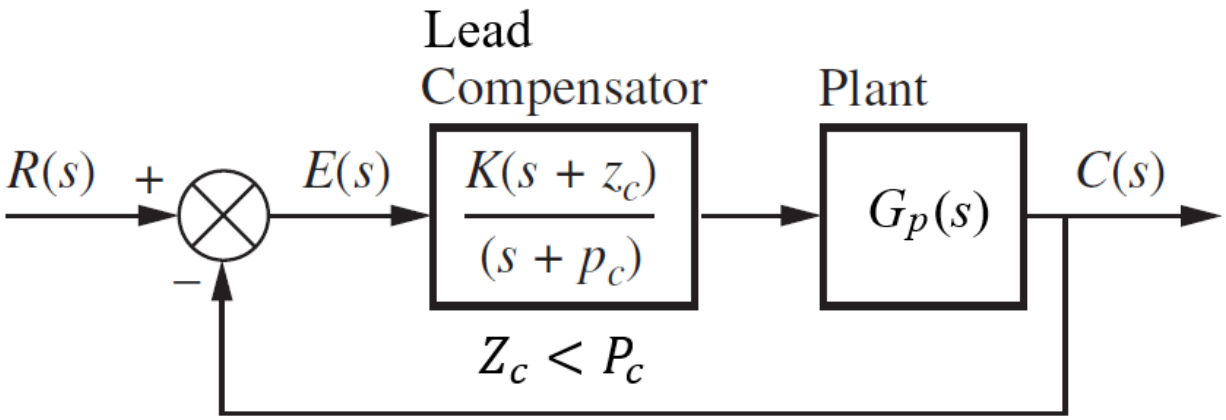
Plots shown are with K = 1 (orange line), K = 10 (yellow line), and K = 100 (purple line).



Notice the growing oscillation as you increase the system gain (K), but settling time increases for all cases.

Lead Compensator

- Lead compensator is typically used in the control systems to improve the transient response and hence the stability of the systems.



- The lead compensators improve transient response and stability, but they do not typically reduce steady-state error.

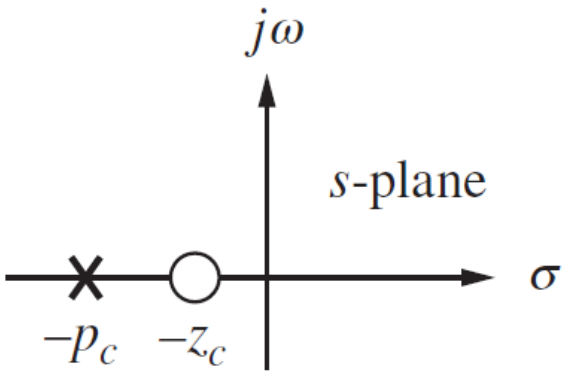
Lead Compensator

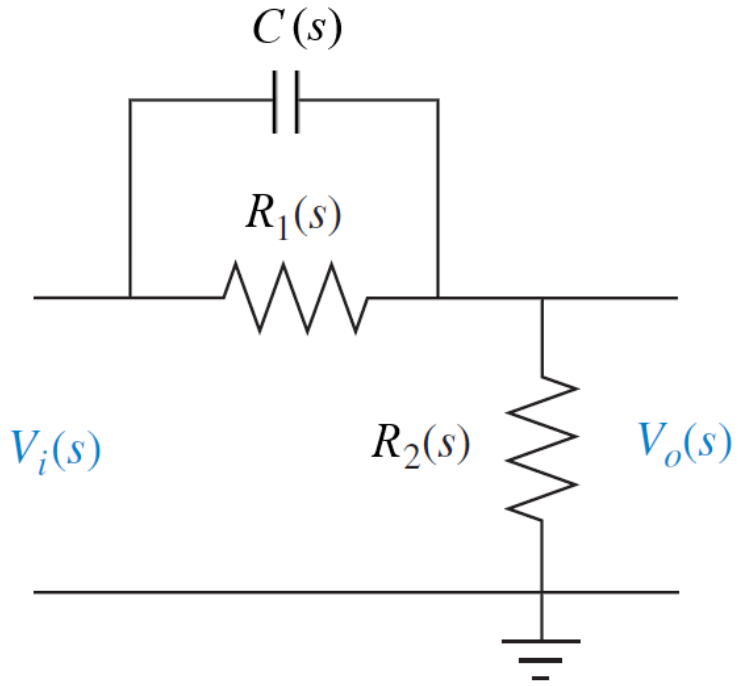
Phase-lead compensator:

- Derivative magnifies noise.
- Instead of D-control or PD-control use “lead control.”

$$G_c(s) = \frac{(s - z_0)}{(s - p_0)} \quad \text{with} \quad |z_0| < |p_0|$$

- That is, the zero is closer to the origin than the pole.





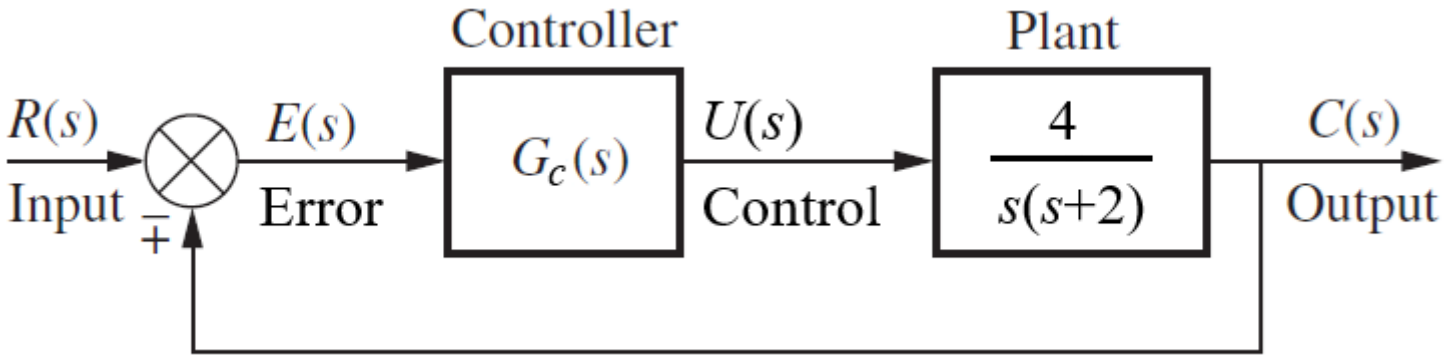
Lead Compensator

- Lead compensator is the same form as the lag compensator, but with a different intention:
  - Lag compensator does not change much since $P_0 \approx Z_0 \approx 0$. Instead, the lag compensator improves the steady-state error.
  - Lead compensator does change locus. Pole and zero locations chosen so that poles will pass through desired point $s = s_1$.

Example for Lead Compensator

When $G_C(s) = 1$, the control system given below suffers from issues in the transient response conditions.

- Rise time: take some time for the system to rise up.
- Settling time: sluggish system that takes time to settle down.



Example for Lead Compensator

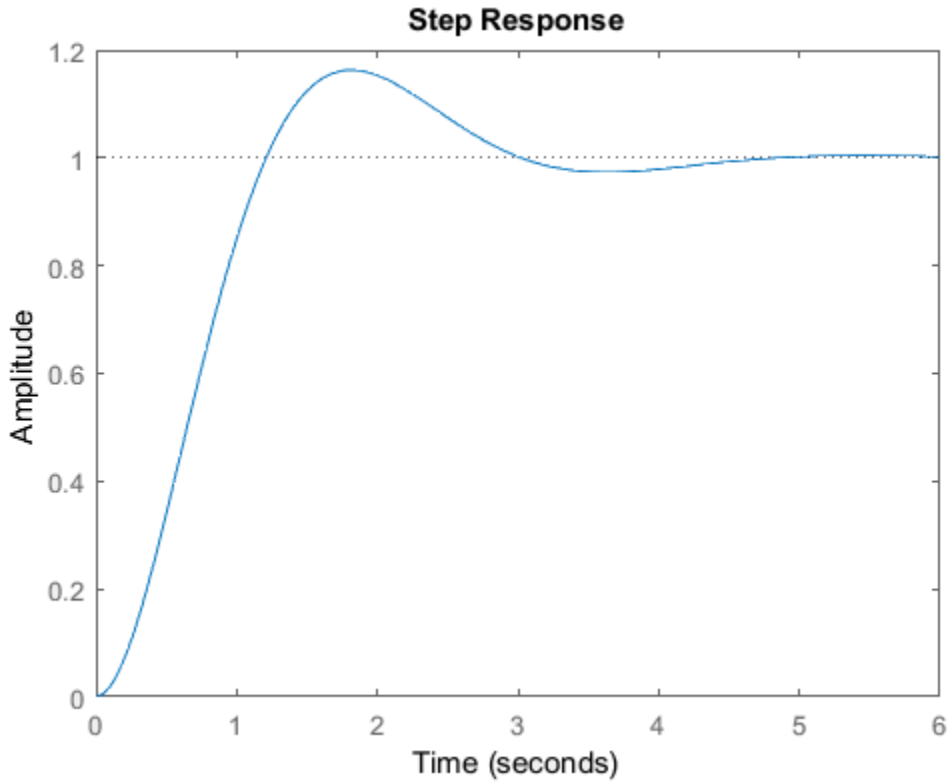
a. Simulate the uncompensated system in MATLAB. Comment on the result of the simulation. [6 marks]

b. Using the trial-and-error methods in MATLAB, design a lead compensator that will be able to fix the problem observed in part (a). [4 marks]

c. Simulate in MATLAB and compare the uncompensated and compensated systems. Observe whether the compensator has achieved its purpose. [6 marks]

Example for Lead Compensator

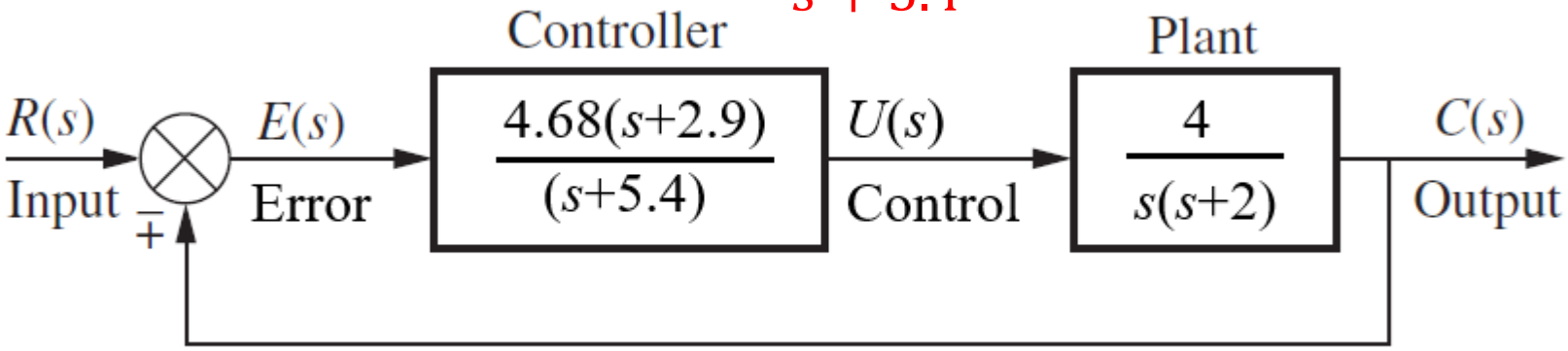
a. Looking into the unit-step response of the given system, there are issues as highlighted before, e.g. slow (sluggish) response of the system, e.g. long rise time and settling time.



Example for Lead Compensator

b. By trial and error, the gain and the pole and zero of the lead compensator are determined:

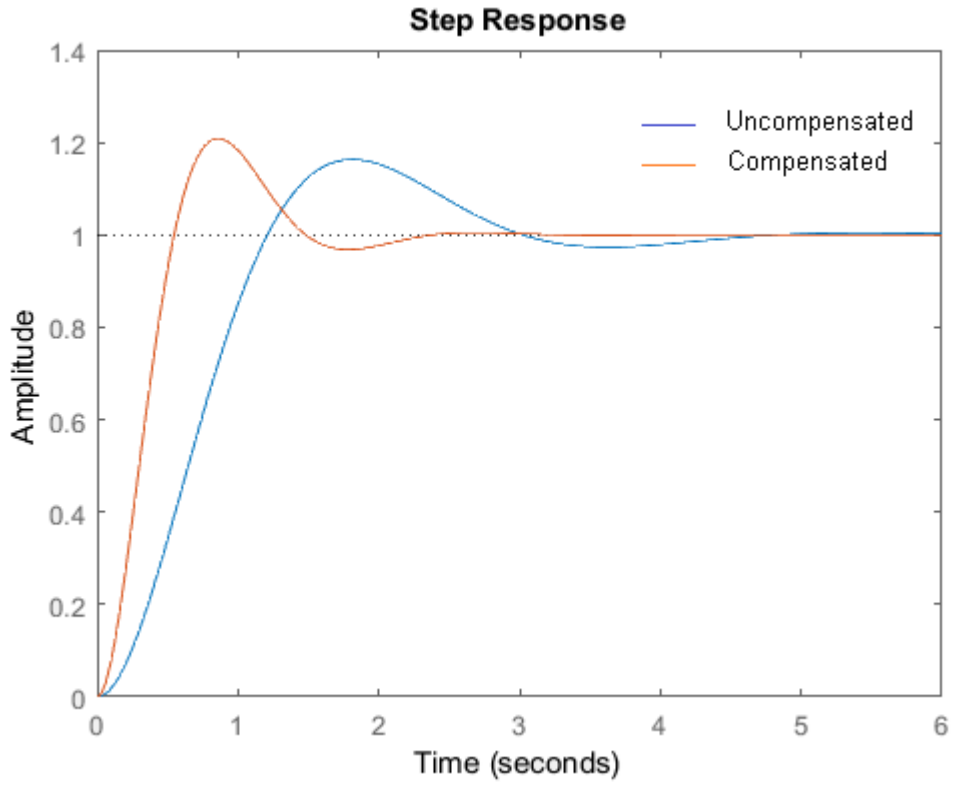
$$C(s) = \frac{4.68(s + 2.9)}{s + 5.4}$$



c. From the plot compensated system (i.e. red line) reaches steady state faster (shorter rise and settling times) than the uncompensated system (i.e. blue line), although it has a higher percentage overshoot, M_p.

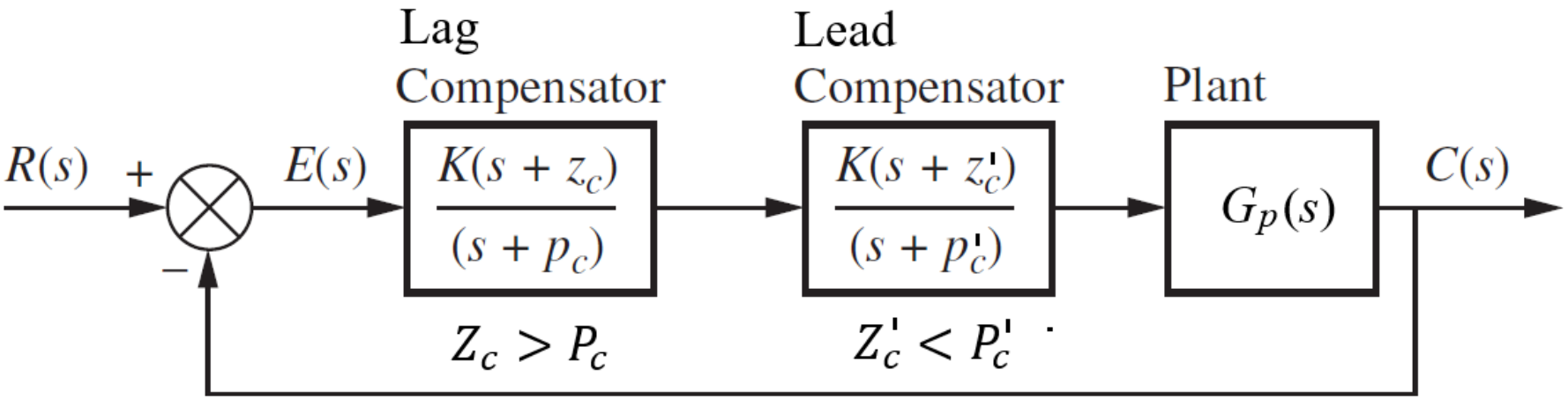
Example for Lead Compensator

The step response plot of the uncompensated and compensated systems with a lead compensator is shown in the figure below.



Lead-Lag Compensator

- For a lead-lag compensator, it combines a lead compensator and a lag compensator.



- Lead-lag compensator provides the benefits of both lead and lag compensators, e.g. improve performance in terms of steady-state conditions and transient responses.

Lead-Lag Compensator

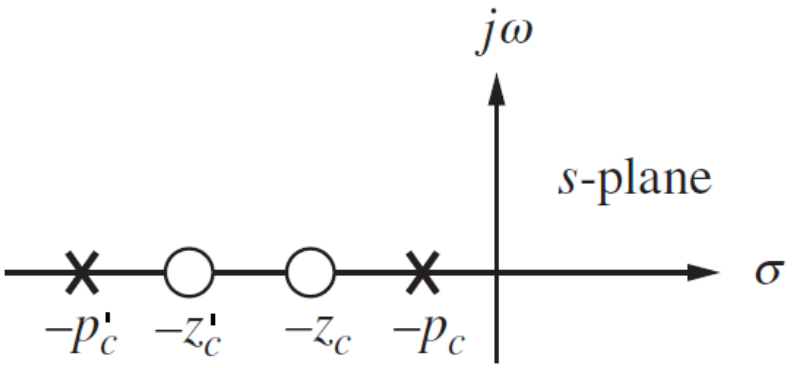
Lead-lag compensator:

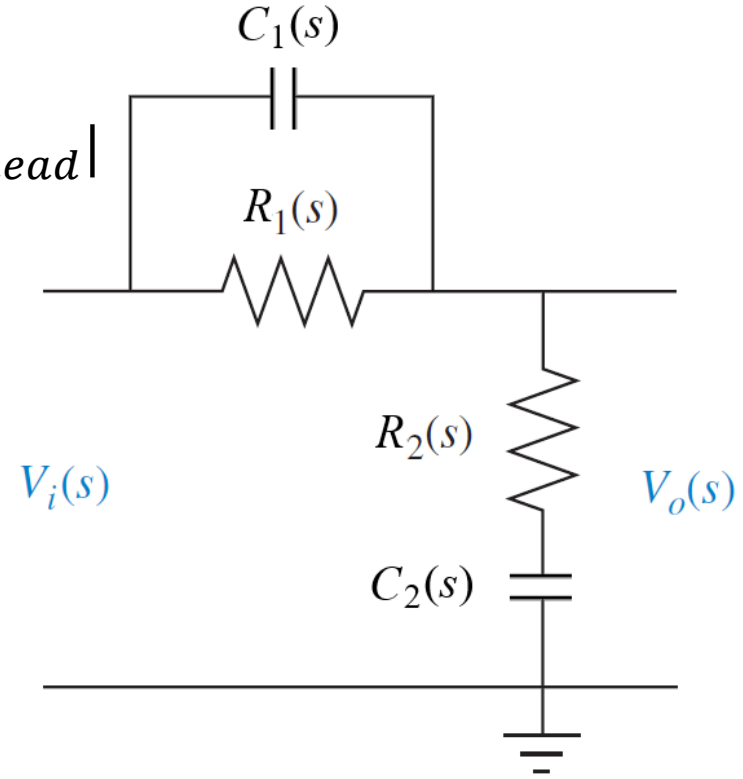
- The transfer function of the lead-lag compensator is:

$$G_c(s) = \frac{(s - z_{lag})}{(s - p_{lag})} \frac{(s - z_{lead})}{(s - p_{lead})}$$

- with

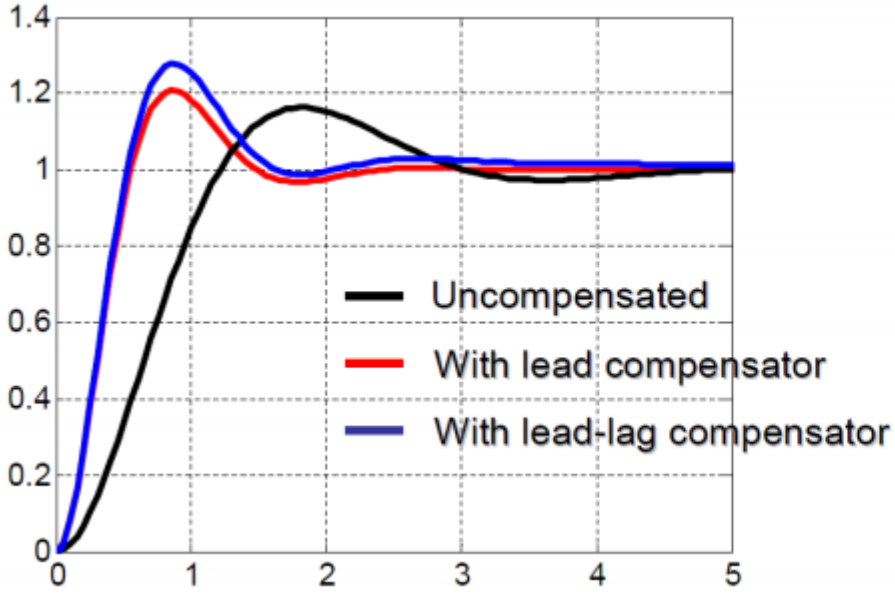
$$|p_{lag}| < |z_{lag}| \text{ and } |z_{lead}| < |p_{lead}|$$

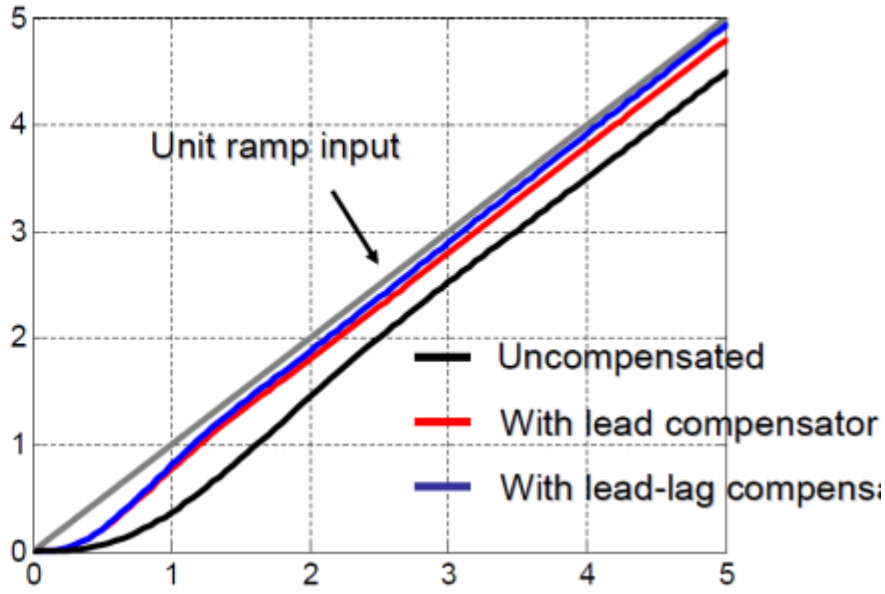




Lead-Lag Compensator

- For lead-lag compensator, left figure (step response), right figure (ramp response)
- Faster transient response when given step input (left).
- Smaller steady-state error when given ramp input (right)





Lead-Lag Compensator

- Lead-lag compensator improves transient response and steady-state condition.
- Design of the lead-lag compensator requires careful design of its individual parts e.g. lag compensator and lead compensator.
- Trial and error is typically employed to get the best set up for the lead-lag compensator.

Lead-Lag Compensator

If we must satisfy both the transient and steady-state specifications:

1. Design a lead compensator to meet the transient specification first.
2. Include a lead compensator with the plant after its design is final.
3. Design a lag compensator (where “plant” = actual plant and lead compensator combined) to meet the steady-state specification.

Summary of Compensator

- Summary of compensators:

Compensator	Transfer Function Equation	Characteristics
Lag	$\frac{(s-z_0)}{(s-p_0)}$ with $\lvert p_0\rvert < \lvert z_0\rvert$	It improves steady-state error.
Lead	$\frac{(s-z_0)}{(s-p_0)}$ with $\lvert z_0\rvert < \lvert p_0\rvert$	It improves transient response performance.
Lead-lag	$\frac{(s-z_{lag})}{(s-p_{lag})}\frac{(s-z_{lead})}{(s-p_{lead})}$ with $\lvert p_{lag}\rvert < \lvert z_{lag}\rvert$ and $\lvert z_{lead}\rvert < \lvert p_{lead}\rvert$	It improves both steady-state error and transient response performance.

Practical Implementations

- Practical implementation of controllers or compensators with op amp-based amplifier circuits.
- Practical implementations of controllers P, I, D and any of their combinations.
- Practical implementations of Lead, Lag, and Lead-lag compensators.
- Due to requirements for realising the gain (P), integral function (I) which is ∞ at low frequency, and derivative function (D) which is ∞ at high frequency, controllers are realised with active devices such as op amps
- On the other hand, compensators can be realised with both active and passive components, such as R, L, and C.

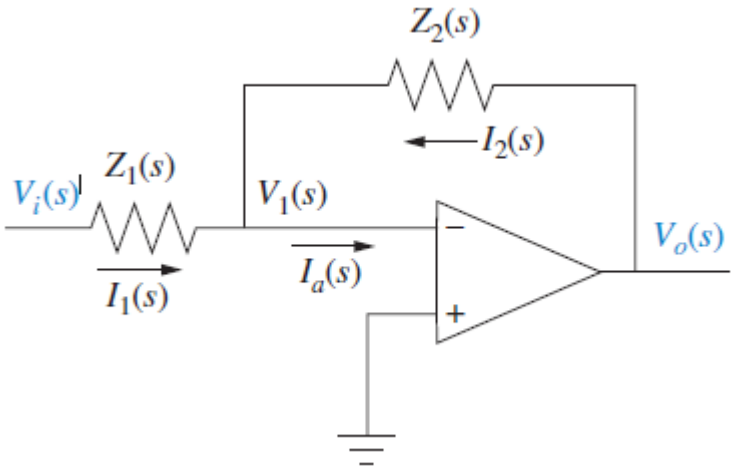
Practical Active Implementations

- We derive the transfer function equation of an inverting operational amplifier whose configuration is shown above:

$$I_1(s) + I_2(s) = I_a(s)$$

- And

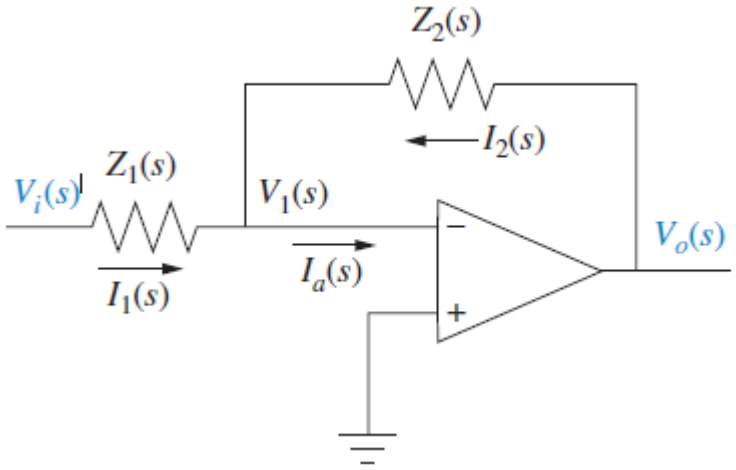
$$\frac{V_1(s) - V_i(s)}{Z_1(s)} + \frac{V_o(s) - V_i(s)}{Z_2(s)} = 0$$



Practical Active Implementations

- With $V_1(s) = V_- = V_+ = 0$ and $I_a(s) = I_- = I_+ = 0$, thus:

$$\frac{V_o(s)}{V_i(s)} = \frac{Z_2(s)}{Z_1(s)}$$



- By judicious choice of $Z_1(s)$ and $Z_2(s)$, this circuit is used as a building block to implement the compensators and controllers, such as PID controllers and lag-lead compensators using op amps.

Practical Active Implementations

- Controller's active circuit realisations:

Controller	$\boldsymbol{Z_1(s)}$	$\boldsymbol{Z_2(s)}$	$\boldsymbol{G_c(s) = \frac{Z_1(s)}{Z_2(s)}}$
Proportional (gain)	R_1	R_2	$-\frac{R_1}{R_2}$
Integral	R	C	$-\frac{\left(\frac{1}{RC}\right)}{s}$
Derivative	C	R	$-RCs$
Proportional-Integral (PI)	R_1	R_2 and C (in series)	$-\left(\frac{R_1}{R_2}\right)\left(\frac{s+\frac{1}{R_2C}}{s}\right)$
Proportional-Derivative (PD)	C and R_1 (in parallel)	R_2	$-R_2C\left(s+\frac{1}{R_1C}\right)$
Proportional-Derivative-Integral (PID)	C_1 and R_1 (in parallel)	R_2 and C_2 (in series)	$-\left(\frac{R_2}{R_1}+\frac{C_1}{C_2}+R_2C_1s+\frac{\frac{1}{R_1C_2}}{s}\right)$

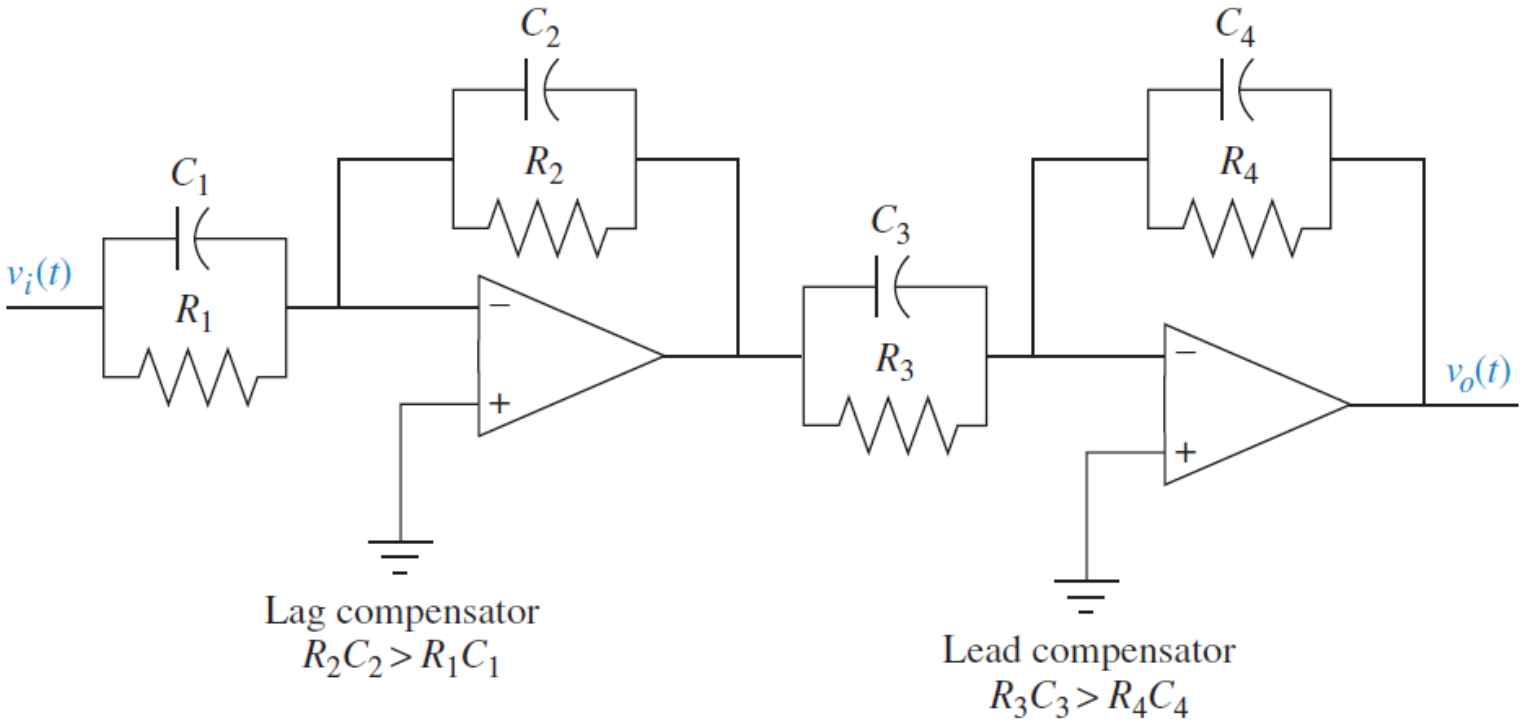
Practical Active Implementations

- Compensator's active circuit realisations:

Compensator	$\boldsymbol{Z_1(s)}$	$\boldsymbol{Z_2(s)}$	$\boldsymbol{G_c(s) = \frac{Z_1(s)}{Z_2(s)}}$
Lag	C_1 and R_1(in parallel)	C_2 and R_2(in parallel)	$-\left(\frac{C_1}{C_2}\right)\left(\frac{s+\frac{1}{R_1C_1}}{s+\frac{1}{R_2C_2}}\right)$ where $R_2C_2 > R_1C_1$
Lead	C_1 and R_1(in parallel)	C_2 and R_2(in parallel)	$-\left(\frac{C_1}{C_2}\right)\left(\frac{s+\frac{1}{R_1C_1}}{s+\frac{1}{R_2C_2}}\right)$ where $R_2C_2 < R_1C_1$
Lead-Lag	Cascading lag compensator with lead compensator (as shown below).		

Practical Active Implementations

- Cascading lag-lead compensator's active circuit realisations:



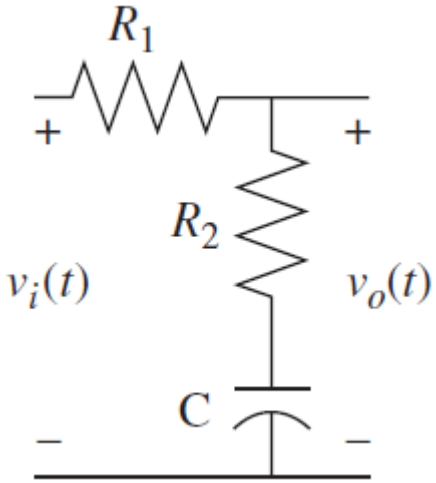
Practical Passive Implementations

- Compensator's passive circuit realisations:

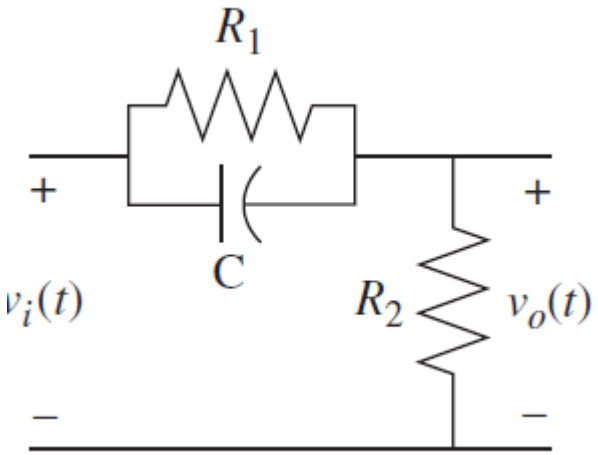
Type	$\boldsymbol{V_i(s)}$	$\boldsymbol{V_o(s)}$	$\boldsymbol{G_c(s) = \frac{V_o(s)}{V_i(s)}}$
Lag	R_1, R_2, and C (all in series)	C and R_2 (both in series)	$\left(\frac{R_1}{R_1+R_2}\right)\left(\frac{s+\frac{1}{R_2C}}{s+\frac{1}{(R_1+R_2)C}}\right)$
Lead	C and R_1 (both in parallel) and in series with R_2	R_2	$\frac{s+\frac{1}{R_1C}}{s+\frac{1}{R_1C}+\frac{1}{R_2C}}$
Lead -Lag	C_1 and R_1(both in parallel) and in series with C_2 and R_2 (both in series)	C_1 and R_2 (both in series)	$\frac{\left(s+\frac{1}{R_1C_1}\right)\left(s+\frac{1}{R_2C_2}\right)}{s+\left(\frac{1}{R_1C_1}+\frac{1}{R_2C_2}+\frac{1}{R_2C_1}\right)s+\frac{1}{R_1R_2C_1C_2}}$

Practical Passive Implementations

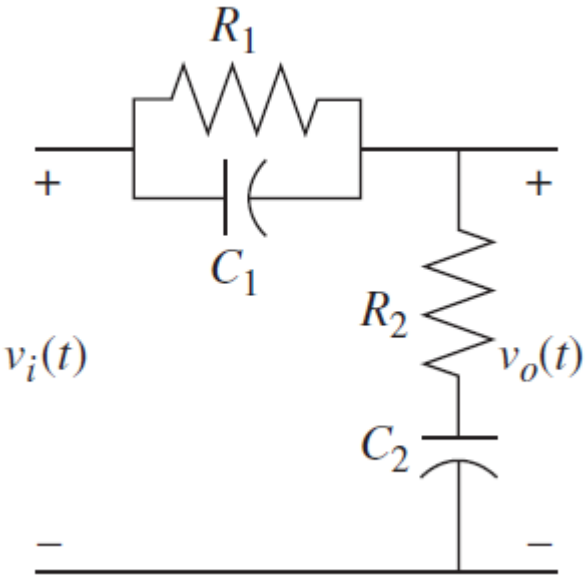
- Compensator's passive circuit realisations:



Lag compensator



Lead compensator



Lag-lead compensator

Example of Practical Implementation

Implement the PID controller when the transfer function of the PID controller is: [10 marks]

$$G_c(s) = \frac{(s + 55.92)(s + 0.5)}{s}$$

Example of Practical Implementation

Answer

- The transfer function of the controller can be put in the form:

$$G_c(s) = s + 56.42 + \frac{27.96}{s} = \left(\frac{R_2}{R_1} + \frac{C_1}{C_2}\right) + R_2 C_1 s + \frac{\frac{1}{R_1 C_2}}{s}$$

- Equating the transfer function of the controller with the PID controller:

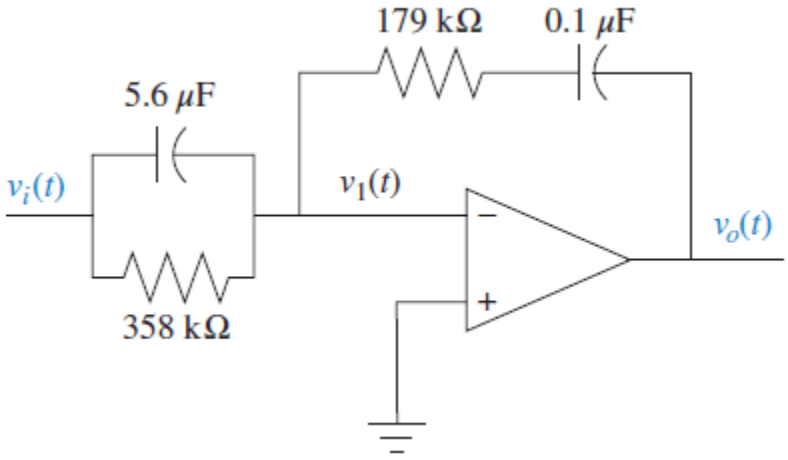
$$s + 56.42 + \frac{27.96}{s} = \left(\frac{R_2}{R_1} + \frac{C_1}{C_2}\right) + R_2 C_1 s + \frac{\frac{1}{R_1 C_2}}{s}$$

Example of Practical Implementation

- Comparing the PID controller in the table with the controller, we obtain the following three relationships:

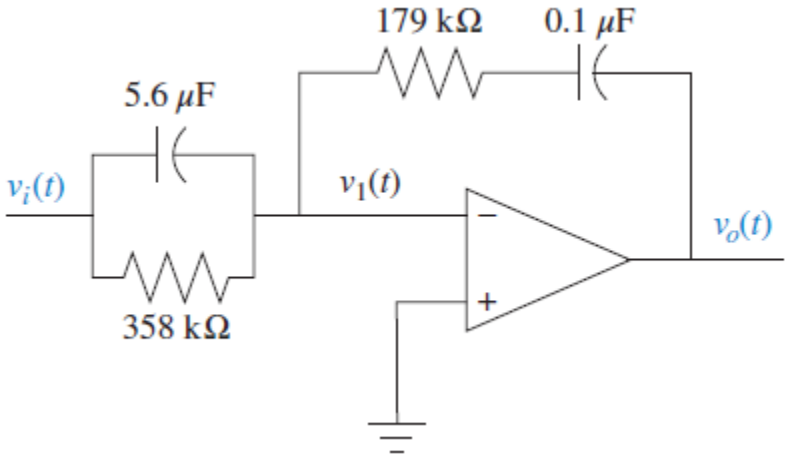
$$\frac{R_2}{R_1} + \frac{C_1}{C_2} = 56.42 \qquad R_2 C_1 = 1 \qquad 1/R_2 C_1 = 27.96$$

- Since there are four unknowns and three equations, we arbitrarily select a practical value for one of the elements.



Example of Practical Implementation

- Selecting C_2 = 0.1 μF, the remaining values are found to be R_1 = 357.65 kΩ, R_2 = 178,891 kΩ, and C_1 = 5.59 μF.
- The complete circuit is shown in the figure below, where the circuit element values have been rounded off.



Configuring the Controller

- In the majority of cases, a compensator is commonly designed for tackling a particular issue or problem in a control system with its specific set-up or arrangement.
- On the other hand, a controller is typically intended and designed to be able to be adjusted or tuned in to manage the operation of the system.
- There are many approaches to configure controllers. But these are typically classified into e.g. ad-hoc (on the spot), experimentation, or prescriptive formulas.

Configuring the Controller

- There are various ways to configure and to tune in the controller in control system.

Tuning	Advantages	Disadvantages
Manual	No math required.	Requires experience.
Software tools	Consistent tuning, can employ computer-automated control system design techniques, may include devices analysis, allows simulation before implementation, and can support non-steady-state (NSS) tuning.	Some cost or training involved.
Ziegler–Nichols	Proven method.	Process upset, some trial-and-error, very aggressive tuning.

Configuring the Controller

Tuning	Advantages	Disadvantages
Tyreus-Luyben	Proven method.	Process upset, some trial-and-error, very aggressive tuning.
Cohen–Coon	Good process models.	Some math required and only good for first-order processes.
Åström-Hägglund	Can be used for auto tuning; amplitude is minimum, so this method has lowest process upset.	The process itself is inherently oscillatory.

Configuring the Controller

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 reduce the overshoot.
4. Add an integral control to reduce the steady-state error.
5. Adjust each of the gains K_p, K_i, and K_d until you obtain a desired overall response. You can always refer to the table shown to find out which controller controls which characteristics.

Configuring the Controller

- 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 meets the given requirements (like the above example), then you don't need to implement a derivative controller on the system.
- Keep the controller as simple as possible.