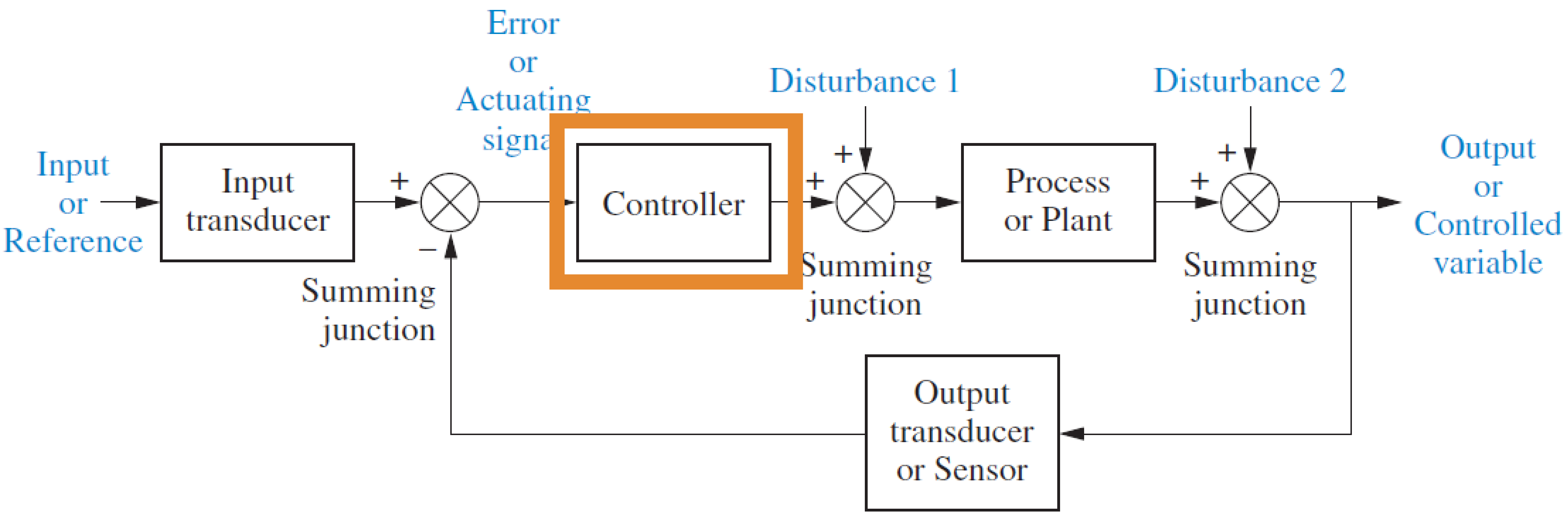
Course Syllabus

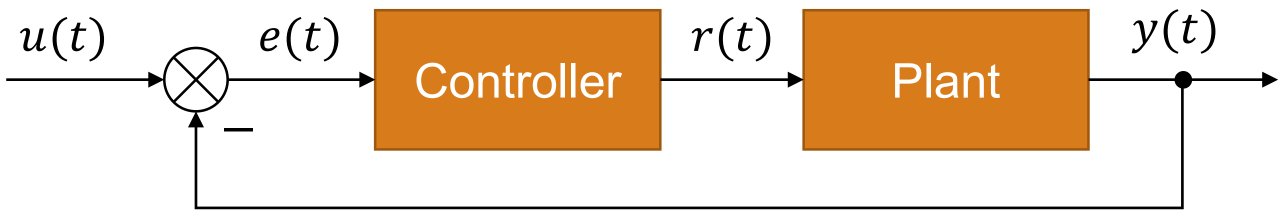
- Basic concepts in control theory
- Basic system classification
- Basic system properties
- System Identification
- **Basic types of controllers**
- Control quality evaluation
- Control systems stability
- Controller design methods
- Digital control
- Sensors

Basic types of controllers



Control Systems Engineering 7th Ed., p. 7

Basic types of controllers



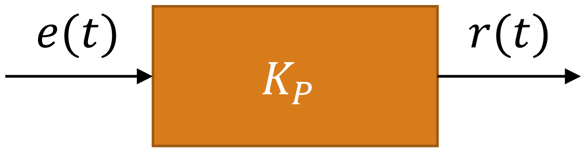
- The controller continuously compares the measured value of the **controlled variable $\boldsymbol{y(t)}$** with the **desired value $\boldsymbol{u(t)}$**
- Based on the resulting **error signal $\boldsymbol{e(t) = u(t) - y(t)}$** it produces an output signal $r(t)$ that influences the plant so as to reduce $e(t) \rightarrow 0$ or to as small a value as possible.
- The controller maintains **accurate tracking** of the desired value $u(t)$ and ensures a **high-quality transient response** (smooth and stable), even when the system is affected by disturbances or parameter changes.

Basic types of controllers

- There are many controller types, but most classical control systems start with PID control, which stands for:
  - **Proportional (P)**
  - **Integral (I)**
  - **Derivative (D)**
- The names of the controllers P, I, and D are derived from the method of generating the output signal based on the error signal $e(t)$
  - The P stands for Proportional, which is based on the current error
  - The I stands for Integral, which is based on the accumulated error over time
  - The D stands for Derivative, which is based on the rate of change of the error.

Basic types of controllers

Proportional Controller



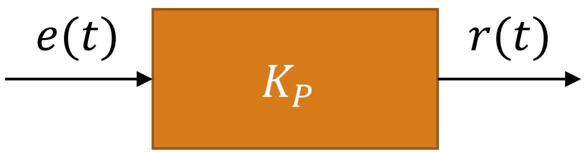
- The proportional controller generates a control signal that is directly proportional to the current error
- The **ideal P controller** is described by the following equation

$$r(t) = K_P e(t)$$

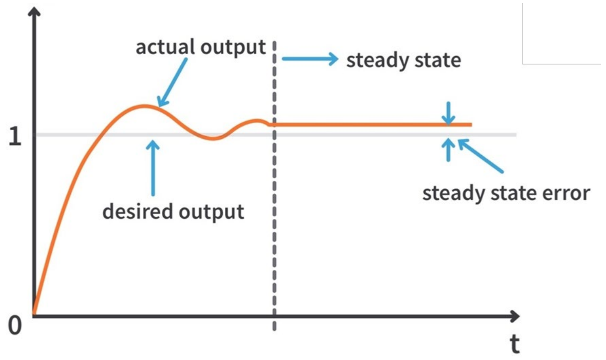
- The constant of proportionality K_P is called **proportional gain**
- **Transfer function**

$$G(s) = \frac{R(s)}{E(s)} = K_P$$

Basic types of controllers
Proportional Controller



- The control action increases when the error is large and decreases when the error is small
- A larger proportional gain K_P leads to a faster response but may cause overshoot or instability
- A larger proportional gain K_P reduces steady-state error but cannot eliminate it completely



Basic types of controllers

Proportional Controller

$$e(t) \longrightarrow \boxed{\frac{K_P}{Ts+1}} \longrightarrow r(t)$$

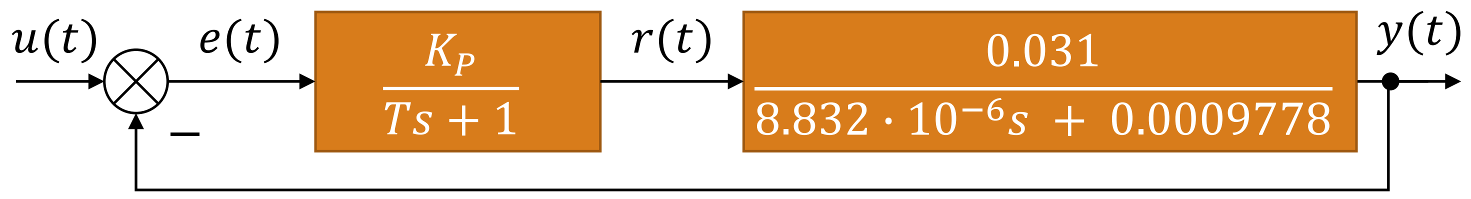
- To model real hardware behavior, a first-order lag (time constant) is introduced:

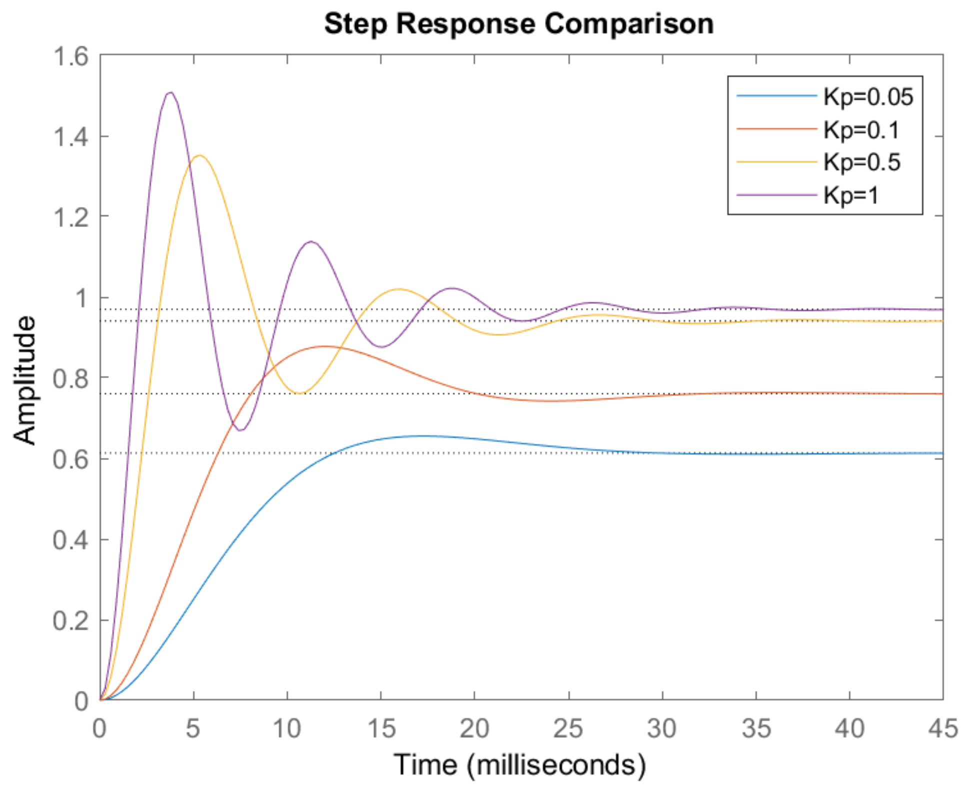
$$G(s) = \frac{R(s)}{E(s)} = \frac{K_P}{Ts+1}$$

- where K_P is **proportional gain** and T is **controller time constant**
- The controller output still increases with the error, but with a finite speed.
- The term $Ts + 1$ introduces a delay (inertia) that reflects the physical dynamics of the controller or actuator.

Basic types of controllers
Proportional Controller

- **DC motor control with real P controller**



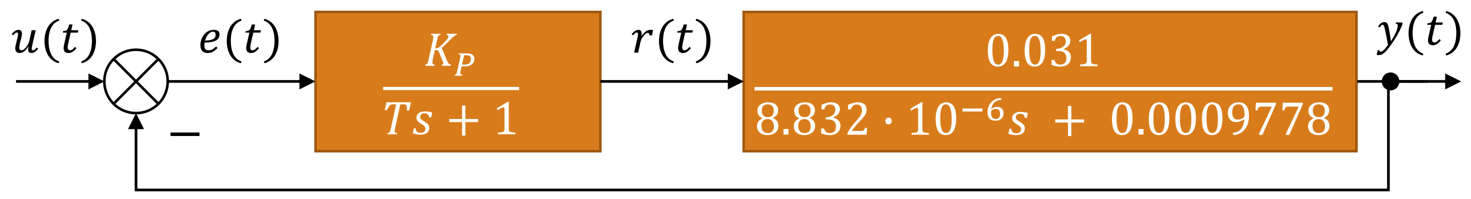


- $K_P = 0.05, 0.1, 0.5, 1$ and $T = 0.005\ s$
- Input $u(t)$: Unit step function

- A larger K_P leads to a faster response
- A larger K_P may cause overshoot or instability
- A larger K_P reduces steady-state error

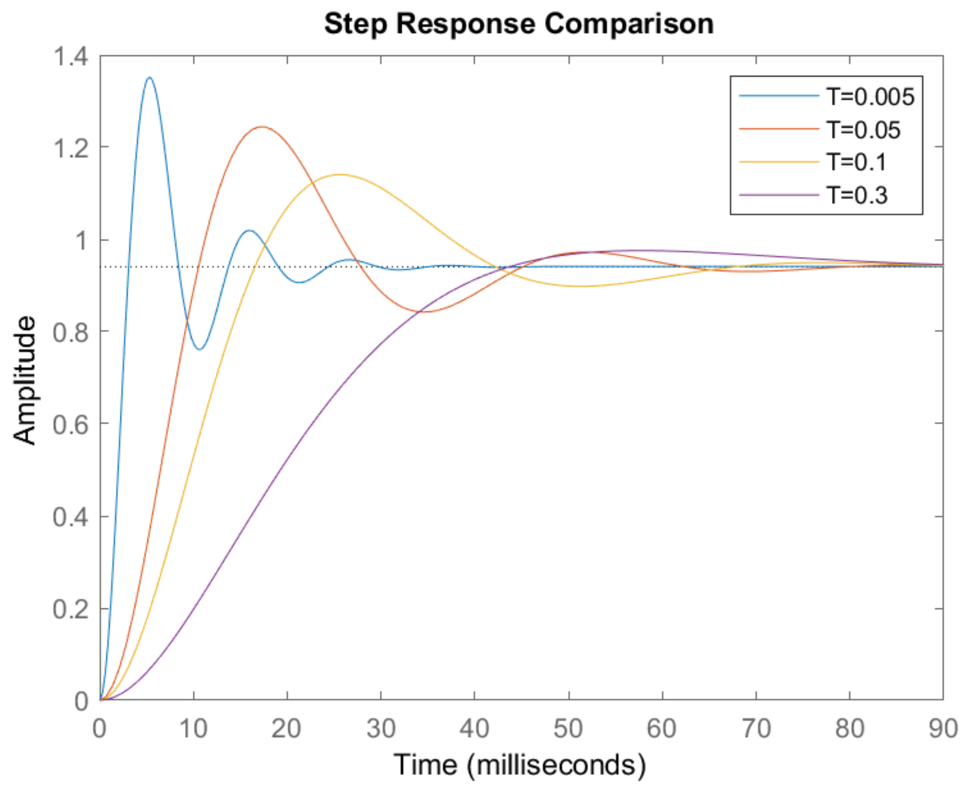
Basic types of controllers
Proportional Controller

- **DC motor control with real P controller**



  - $K_P = 0.5$ and $T = 0.005, 0.05, 0.1, 0.3$
  - Input $u(t)$: Unit step function

  - A larger T leads to a slower response
  - A larger T leads to lower overshoot
  - A larger T does not change steady-state error



Basic types of controllers
Proportional Controller

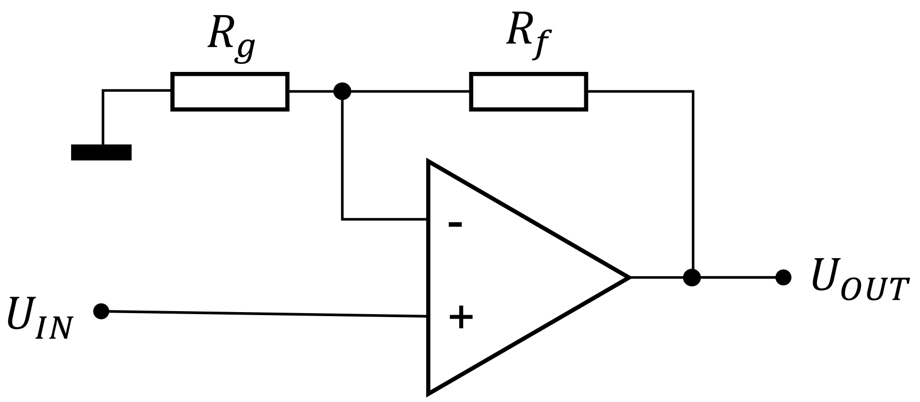
- MATLAB Code that was used to generate step responses

```
clear; clc; close all;
s = tf('s'); % definition of symbolic variable
G1=0.031/(8.832e-6*s+9.9184e-6+0.0009679); % DC Motor first-order approximation
K_list = [0.05 0.1 0.5 1];
T_list = [0.005 0.05 0.1 0.3];
figure;
hold on;
for K=K_list
    C = K/(0.005*s+1); % controller transfer function
    T=feedback(G1*C,1,-1); % feedback loop transfer function
    step(T);
end
legend('Kp=0.05','Kp=0.1','Kp=0.5','Kp=1');

figure;
hold on;
for T=T_list
    C = 0.5/(T*s+1); % controller transfer function
    Tr=feedback(G1*C,1,-1); % feedback loop transfer function
    step(Tr);
end
legend('T=0.005','T=0.05','T=0.1','T=0.3');
```

Basic types of controllers
Proportional Controller

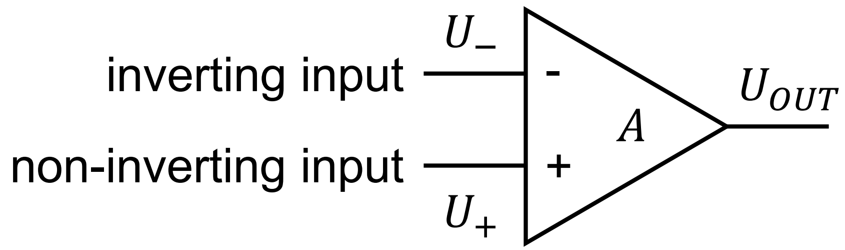
- **Analog implementation of a Proportional (P) controller**
  - A proportional controller produces an output proportional to the instantaneous error
  - In analog hardware this is implemented as a simple amplifier with gain K_P
  - Most common implementation is **non-inverting operational amplifier (op-amp)**



Transfer function with an ideal op-amp

$$\frac{U_{OUT}}{U_{IN}} = 1 + \frac{R_f}{R_g}$$

Basic types of controllers
Proportional Controller

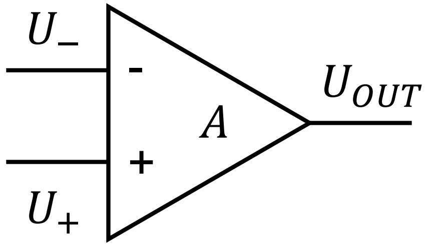


- **Ideal operational amplifier** (op-amp)
  - The op-amp has two inputs:
    - "+" (non-inverting input)
    - "–" (inverting input)
  - It is an electronic device that compares two voltages and produces a large voltage output that is proportional to the difference between them

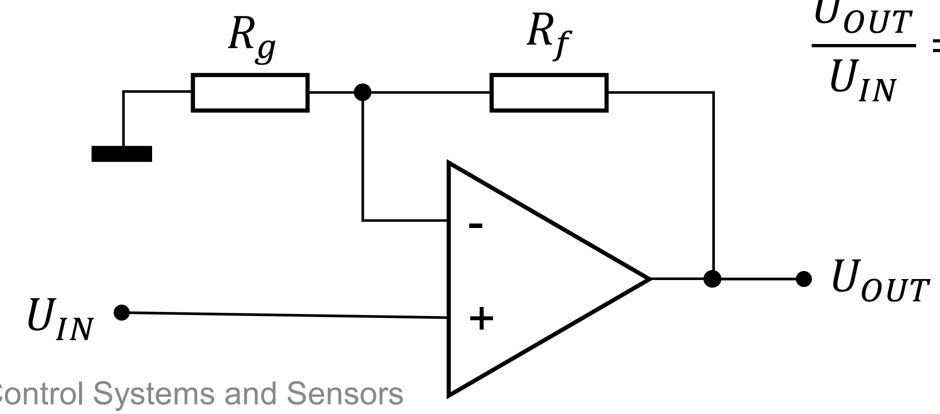
$$U_{OUT} = A(U_+ - U_-)$$

  - where A is extremely high from 10^5 to 10^8 and is called amplifier's (open-loop) gain
  - Because the op-amp can precisely measure and amplify small differences, we can use it to **create analog control laws** – proportional, integral, or derivative actions – simply by connecting resistors and capacitors around it.

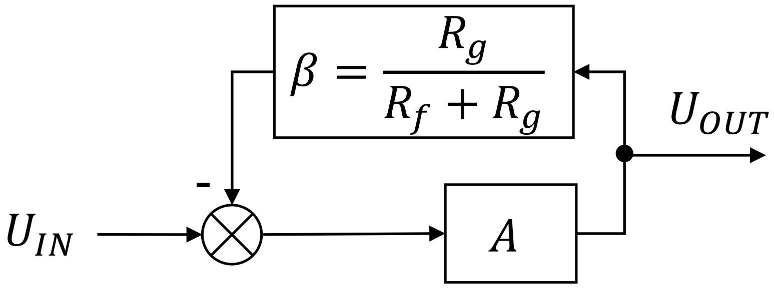
Basic types of controllers
Proportional Controller



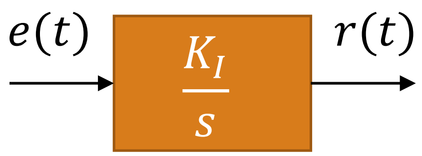
- **Ideal operational amplifier** (op-amp)
  - The magic of the op-amp comes when we use **feedback** – connecting part of the output back to the inverting input.
  - Feedback allows the op-amp to automatically adjust its output so that $U_+ \approx U_-$
  - This makes the circuit's behavior predictable and linear, even though the op-amp itself has a huge gain.
  - With two resistors we can construct the fundamental feedback network called **non-inverting amplifier**



$$\frac{U_{OUT}}{U_{IN}} = 1 + \frac{R_f}{R_g}$$



Basic types of controllers
Integral Controller

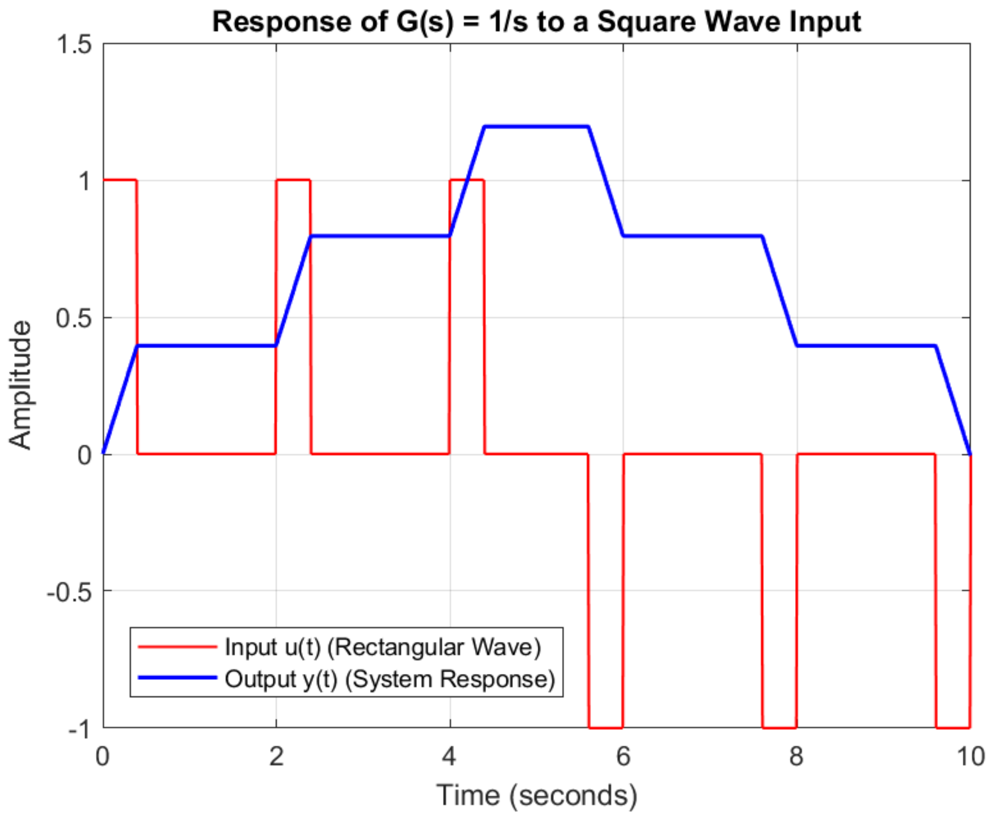


- The integral controller generates a control signal that is proportional to the accumulated (integrated) error over time
- The **ideal I controller** is described by the following equation

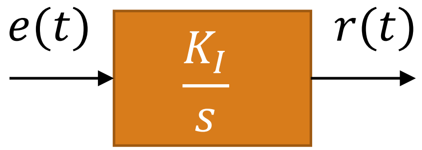
$$r(t) = K_I \int_0^t e(\tau)d\tau$$

- The constant of proportionality K_I is called **integral gain**
- **Transfer function**

$$G(s) = \frac{R(s)}{E(s)} = \frac{K_I}{s}$$



Basic types of controllers
Integral Controller



- The integral term **eliminates steady-state error** by continuously adjusting the control output until $e(t)$ averages to zero
- Provides zero steady-state error but may **slow the response** or **cause oscillations**.
- Often used together with proportional action (**PI control**).

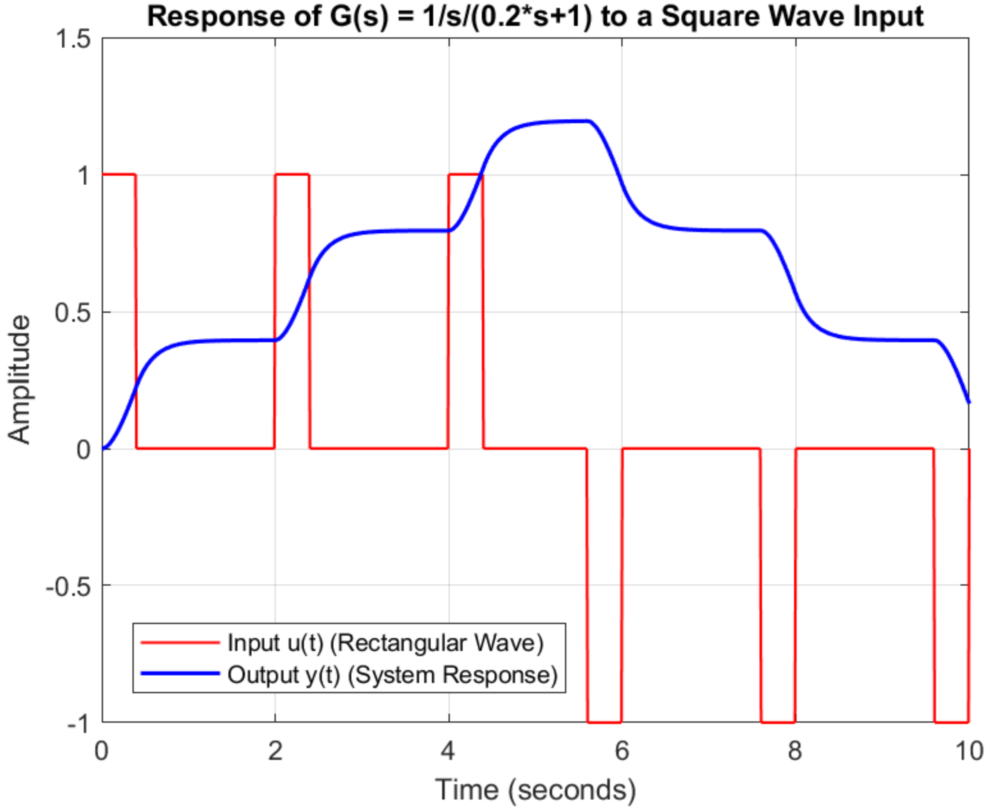
Basic types of controllers
Integral Controller

$$e(t) \longrightarrow \boxed{\frac{K_I}{s(Ts+1)}} \longrightarrow r(t)$$

- The ideal integrator is a useful mathematical model – but not physically realizable
- To model real integral controller, a first-order lag (time constant) is introduced:

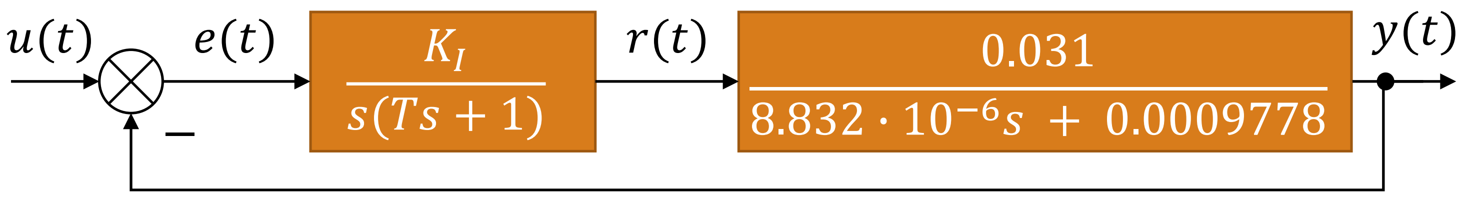
$$G(s) = \frac{R(s)}{E(s)} = \frac{K_I}{s(Ts+1)}$$

- where K_I is **integral gain** and T is **controller time constant**



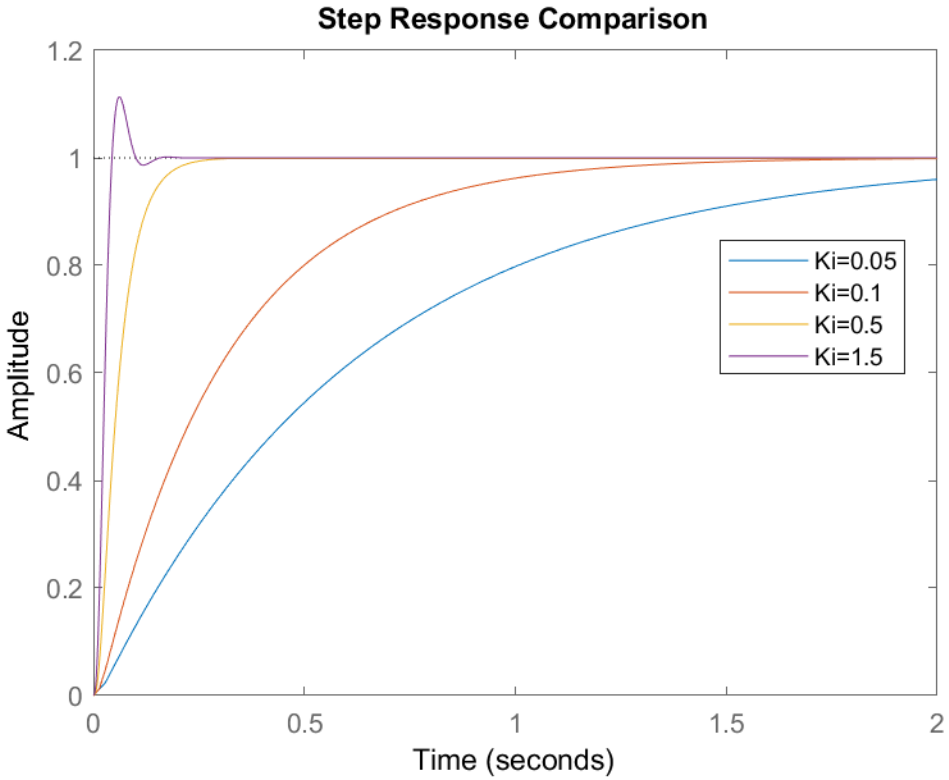
Basic types of controllers
Integral Controller

- **DC motor control with real I controller**



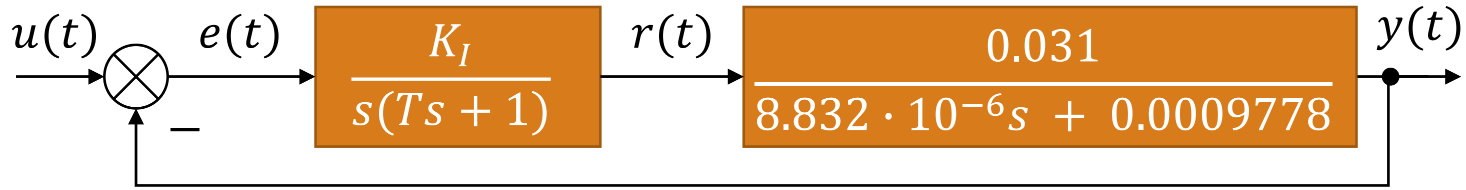
- $K_I = 0.05, 0.1, 0.5, 1.5$ and $T = 0.005\ s$
- Input $u(t)$: Unit step function

- A larger K_I leads to a faster response
- A larger K_I may cause overshoot or instability
- Steady-state error is zero for all K_I



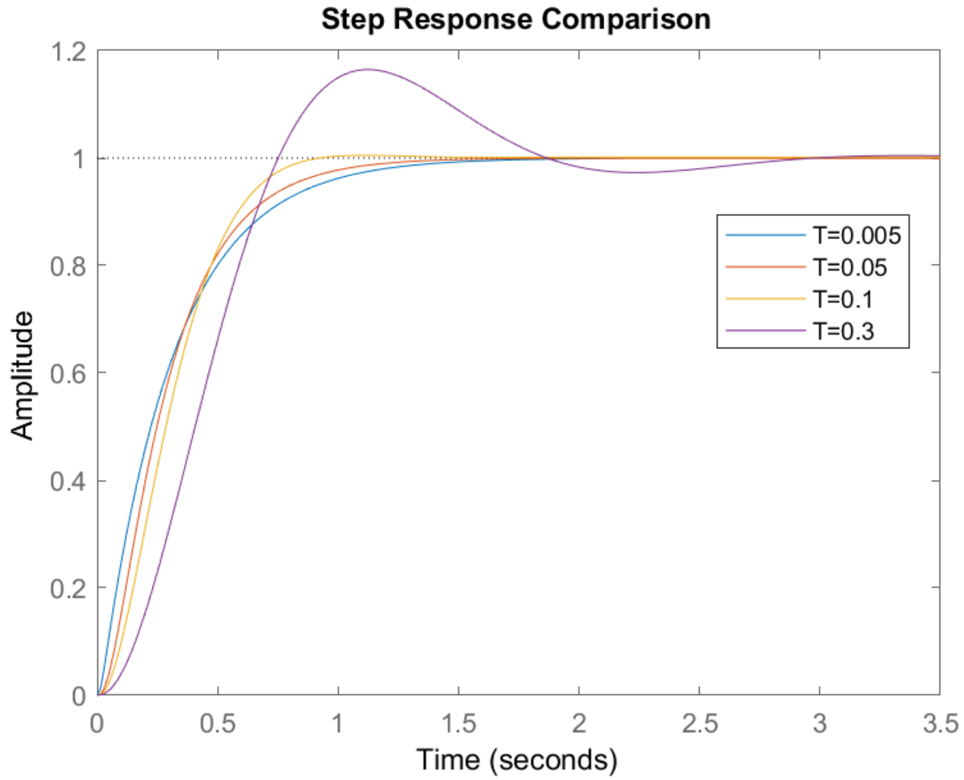
Basic types of controllers
Integral Controller

- **DC motor control with real I controller**



- $K_I = 0.1$ and $T = 0.005, 0.05, 0.1, 0.3$
- Input $u(t)$: Unit step function

- A larger T leads to a slower response
- A larger T may lead to instability



Basic types of controllers
Integral Controller

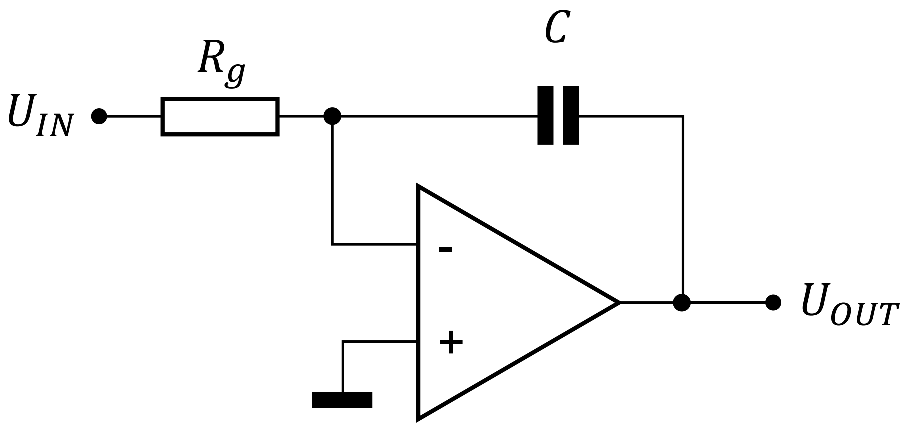
- MATLAB Code that was used to generate step responses

```
clear; clc; close all;
s = tf('s'); % definition of symbolic variable
G1=0.031/(8.832e-6*s+9.9184e-6+0.0009679); % DC Motor first-order approximation
K_list = [0.05 0.1 0.5 1.5];
T_list = [0.005 0.05 0.1 0.3];
figure;
hold on;
for K=K_list
    C = K/s/(0.005*s+1); % controller transfer function
    T=feedback(G1*C,1,-1); % feedback loop transfer function
    step(T);
end
legend('Ki=0.05','Ki=0.1','Ki=0.5','Ki=1.5');

figure;
hold on;
for T=T_list
    C = 0.1/s/(T*s+1); % controller transfer function
    Tr=feedback(G1*C,1,-1); % feedback loop transfer function
    step(Tr);
end
legend('T=0.005','T=0.05','T=0.1','T=0.3');
```

Basic types of controllers
Integral Controller

- **Analog implementation of an Integral (I) controller**
  - An integral controller produces an output proportional to the integrated (accumulated) error over time
  - In analog hardware this is implemented with op-amp, a resistor and a capacitor in **inverting op-amp** configuration



Transfer function with an ideal op-amp

$$\frac{U_{OUT}}{U_{IN}} = -\frac{1}{R_g C s} = \frac{-1/R_g C}{s} \qquad K_I = -\frac{1}{R_g C}$$

Basic types of controllers
Derivative Controller

- The derivative controller generates a control signal that is proportional to the rate of change of the error
- The **ideal D controller** is described by the following equation

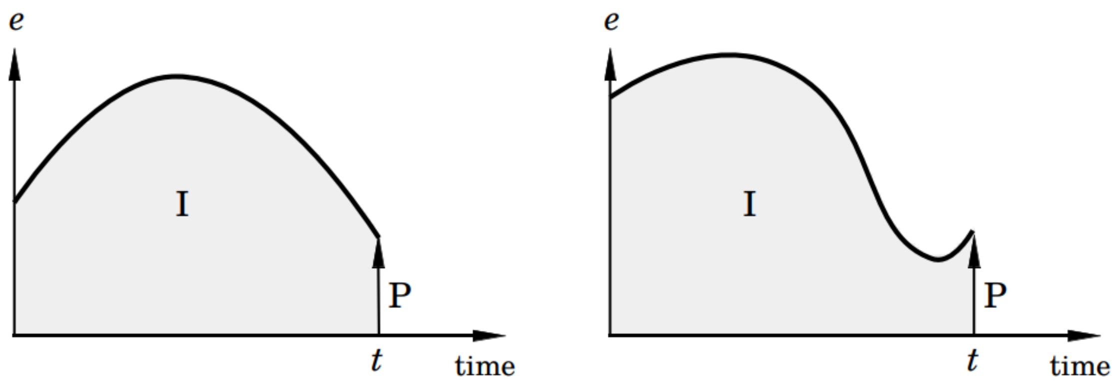
$$r(t) = K_D \frac{de(t)}{dt}$$

- The constant of proportionality K_D is called **derivative gain**
- **Transfer function**

$$G(s) = \frac{R(s)}{E(s)} = K_D \mathrm{s}$$

Basic types of controllers
Derivative Controller

- Reacts to **how quickly the error is changing**, predicting future behavior of the system



- In the left curve, the control error decreases rapidly and the control action should be cautious, in order not to cause an overshoot
- In the right curve, a decrease in the control error is followed by a sudden increase; here, the controller should apply a large control signal in order to lower the control error.

Basic types of controllers
Derivative Controller

- **Improves transient response** and damping, reducing overshoot and oscillations
- Highly **sensitive to measurement noise**, so derivative action must be used carefully and often with filtering
- Derivative controllers can only be used in a feedback path, not alone in the main control path
  - The output of a derivative controller depends on the rate of change of the error $e(t)$
  - This means that the output is nonzero only when the input is changing.
  - If the control error $e(t)$ is constant, then its derivative is zero – so the controller output is zero as well

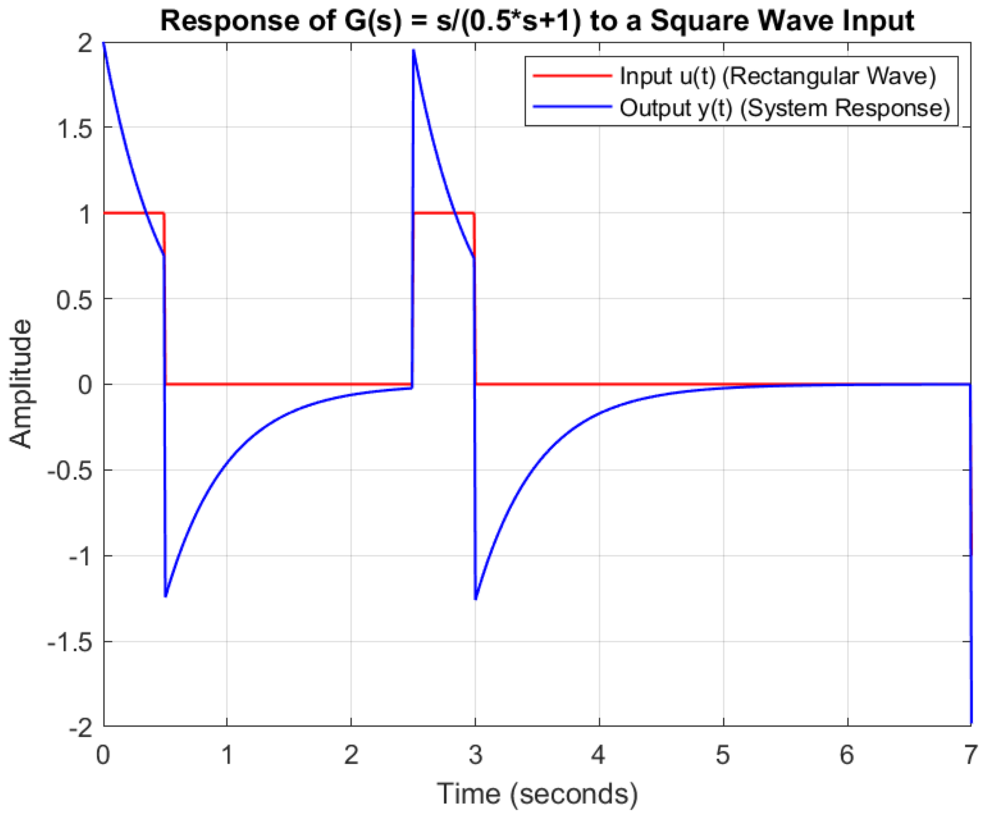
Basic types of controllers

Derivative Controller

- The ideal derivative is not physically implementable, because it would require infinite bandwidth and infinite gain
- To make it physically realizable and noise-resistant, we add a first-order low-pass filter that limits the high-frequency gain

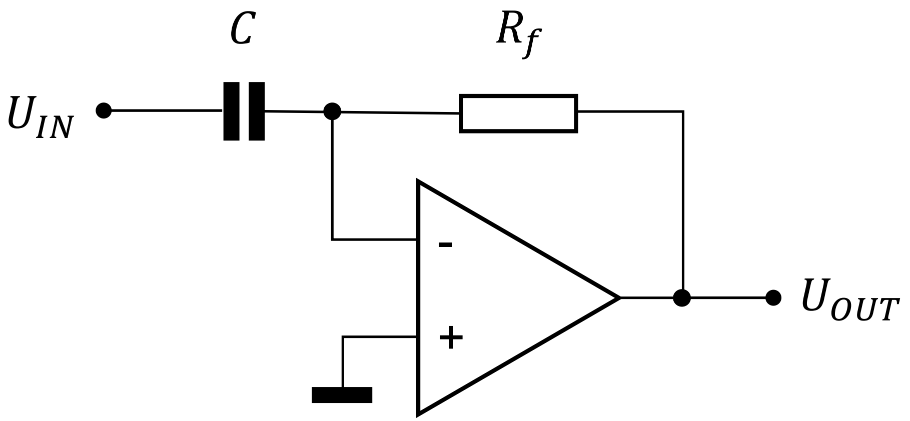
$$G(s) = \frac{R(s)}{E(s)} = \frac{K_D s}{Ts + 1}$$

- where K_D is **derivative gain** and T is **controller time constant**



Basic types of controllers
Derivative Controller

- **Analog implementation of a Derivative (D) controller**
  - An derivative controller produces an output proportional to the to the rate of change of the error
  - In analog hardware this is implemented with op-amp, a resistor and a capacitor in **inverting op-amp** configuration



Transfer function with an ideal op-amp

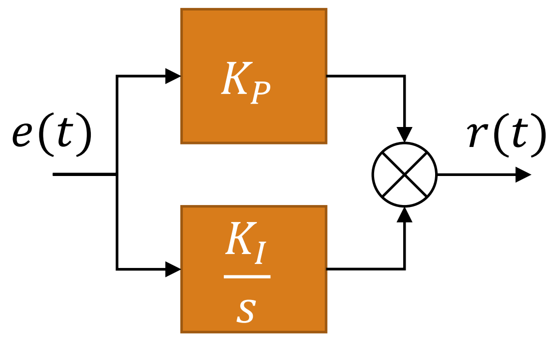
$$\frac{U_{OUT}}{U_{IN}} = -R_f C s \qquad K_D = -R_f C$$

Basic types of controllers
Basic Controllers Summary

Controller	Control Law	Main effect	Limitations
P	$K_P e(t)$	Faster response, reduces error	Steady-state error remains
I	$K_I \int_0^t e(\tau)d\tau$	Eliminates steady-state error	Slower, may oscillate
D	$K_D \frac{de(t)}{dt}$	Improves stability and damping	Sensitive to noise

Basic types of controllers

PI Controller



- The PI controller combines the fast response of the P action with the zero steady-state error of the I action
- The **ideal PI controller** is described by the following equation

$$r(t) = K_P e(t) + K_I \int_0^t e(\tau) d\tau$$

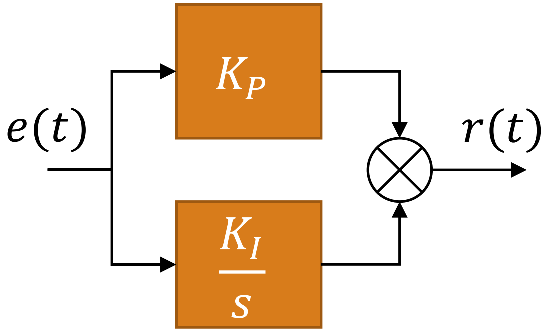
- where K_P is **proportional gain**, K_I is **integral gain**
- **Transfer function**

$$G(s) = \frac{R(s)}{E(s)} = K_P + \frac{K_I}{s} = K_P \left(1 + \frac{K_I}{K_P} \frac{1}{s}\right) = K_P \left(1 + \frac{1}{T_I s}\right)$$

- where $T_I = \frac{K_P}{K_I}$ is integral time constant

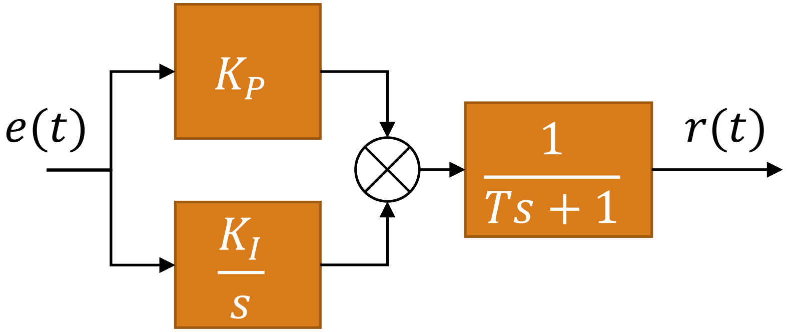
Basic types of controllers

PI Controller



- The PI controller is the **most widely used** type of controller in practice
- Compared to Proportional controller alone → the PI controller removes steady-state offset
- Compared to Integral controller alone → it is faster and causes less overshoot
- It provides good dynamic and steady-state performance, and it is suitable even for **controlling Type 1 systems**, where the plant itself already contains one integrator
- Easy to tune (manual or automatic methods) – has only two parameters K_P and K_I

Basic types of controllers
PI Controller



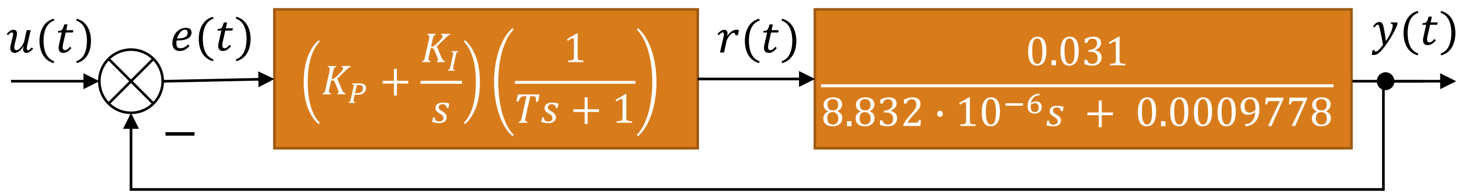
- **Real (practical) PI controller**
  - The ideal form can be made practical by limiting the integrator's bandwidth with a filter, giving:

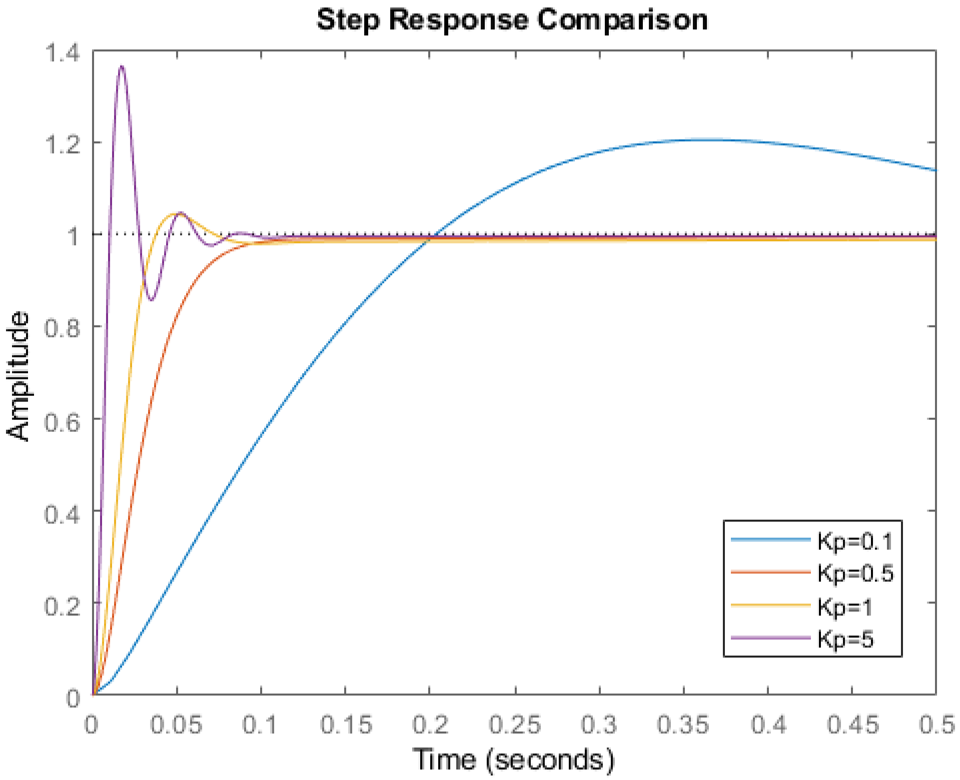
$$G(s) = \frac{R(s)}{E(s)} = \left(K_P + \frac{K_I}{s}\right)\left(\frac{1}{Ts+1}\right)$$

where T is **filter time constant**

Basic types of controllers
PI Controller

- **DC motor control with real PI controller**



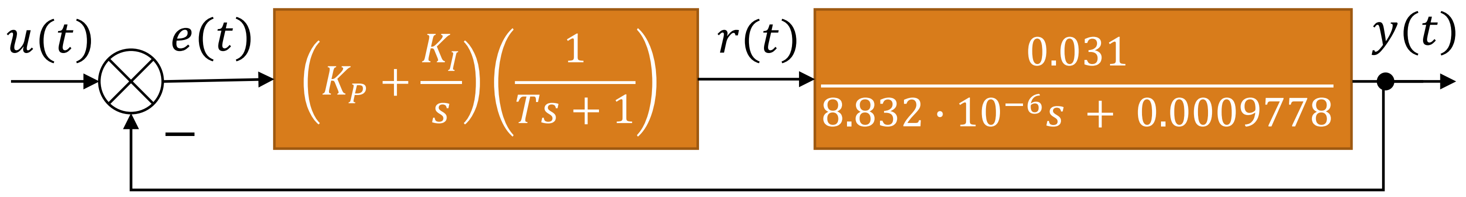


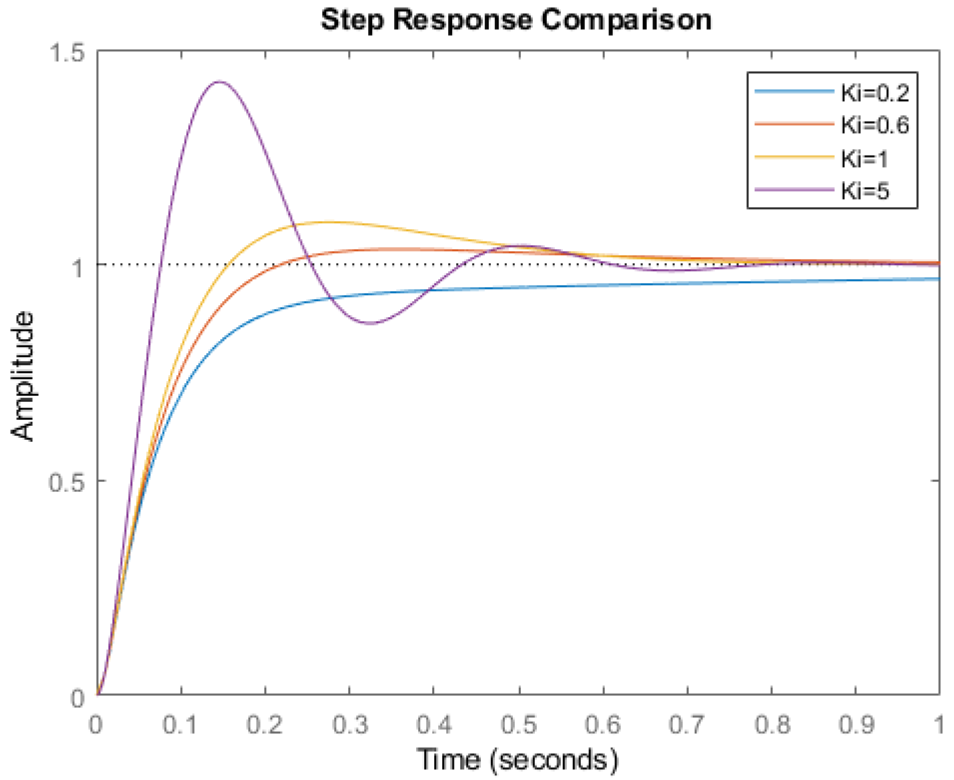
  - $K_P = 0.1, 0.5, 1.0, 5.0$ and $K_I = 0.8$ and $T = 0.5\ s$
  - Input $u(t)$: Unit step function

  - A larger K_P leads to a faster response but may cause overshoot
  - A larger K_P amplifies measurement noise

Basic types of controllers
PI Controller

- **DC motor control with real PI controller**





- $K_P = 0.2$ and $K_I = 0.2, 0.6\ 1.0\ 5.0$ and $T = 0.5\ s$
- Input $u(t)$: Unit step function

- A larger K_I leads to a faster response but may cause overshoot

Basic types of controllers
PI Controller

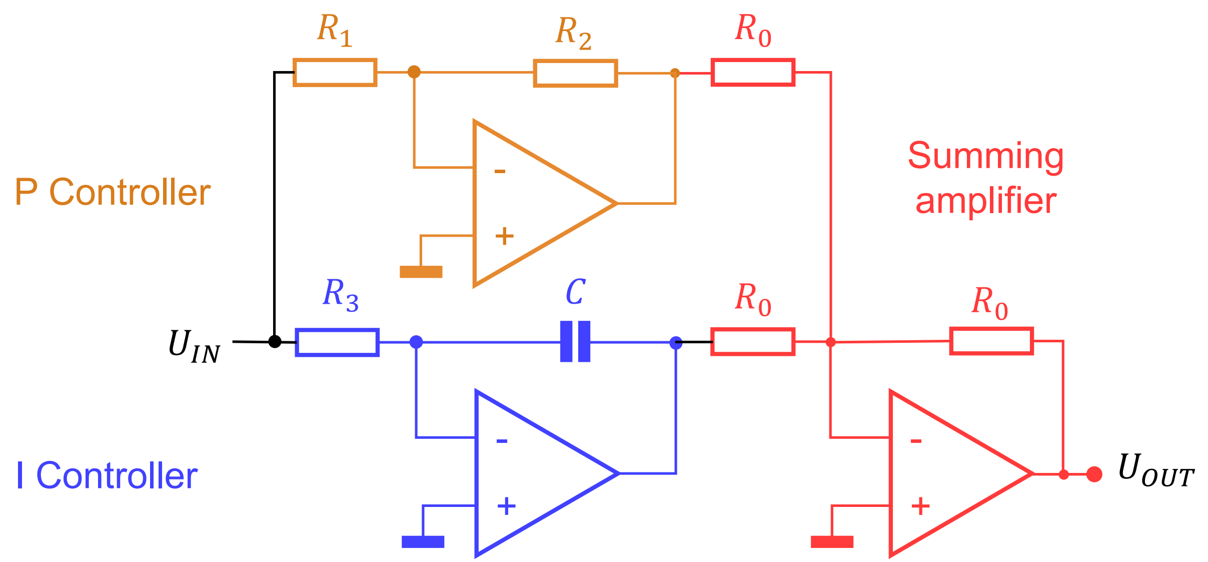
- MATLAB Code that was used to generate step responses

```
clear; clc; close all;
s = tf('s'); % definition of symbolic variable
G1=0.031/(8.832e-6*s+9.9184e-6+0.0009679); % DC Motor first-order approximation
Kp_list = [0.1 0.5 1 5];
Ki_list = [0.2 0.6 1 5];
figure; hold on;
Ki = 0.8; %integral gain
Tf = 0.1; % filter time constant
for Kp=Kp_list
    C = (Kp+Ki/s)/(Tf*s+1); % controller transfer function
    T=feedback(G1*C,1,-1); % feedback loop transfer function
    step(T,1);
end
title('Step Response Comparison');
legend('Kp=0.1','Kp=0.5','Kp=1','Kp=5');

figure; hold on;
Kp = 0.8;
for Ki=Ki_list
    C = (Kp+Ki/s)/(Tf*s+1); % controller transfer function
    Tr=feedback(G1*C,1,-1); % feedback loop transfer function
    step(Tr,1);
end
title('Step Response Comparison');
legend('Ki=0.2','Ki=0.6','Ki=1','Ki=5');
```

Basic types of controllers
PI Controller

- **Analog implementation of PI controller**
  - In analog hardware it is implemented with **three op-amps** in inverting op-amp configuration

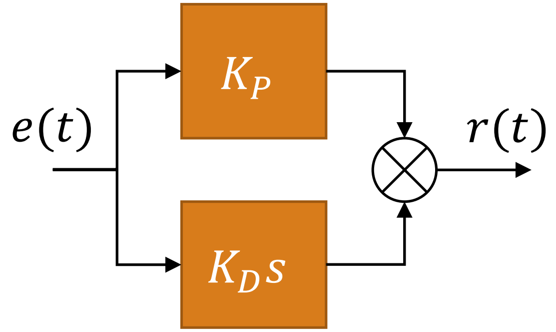


Transfer function with ideal op-amps

$$\frac{U_{OUT}}{U_{IN}} = -\left(-\frac{R_2}{R_1} - \frac{1}{R_3 C s}\right) = \frac{R_2}{R_1} + \frac{1}{R_3 C s}$$

$$K_P = \frac{R_2}{R_1}, K_I = \frac{1}{R_3 C}$$

Basic types of controllers
PD Controller



- A PD controller improves speed and stability rather than steady-state accuracy
- The **ideal PD controller** is described by the following equation

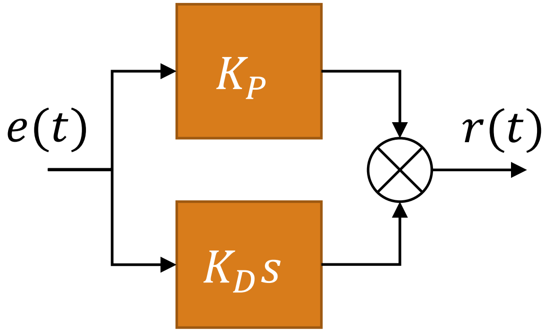
$$r(t) = K_P e(t) + K_D \frac{de(t)}{dt}$$

- where K_P is **proportional gain**, K_D is **derivative gain**
- **Transfer function**

$$G(s) = \frac{R(s)}{E(s)} = K_P + K_D s = K_P\left(1 + \frac{K_D}{K_P} s\right) = K_P(1 + T_D s)$$

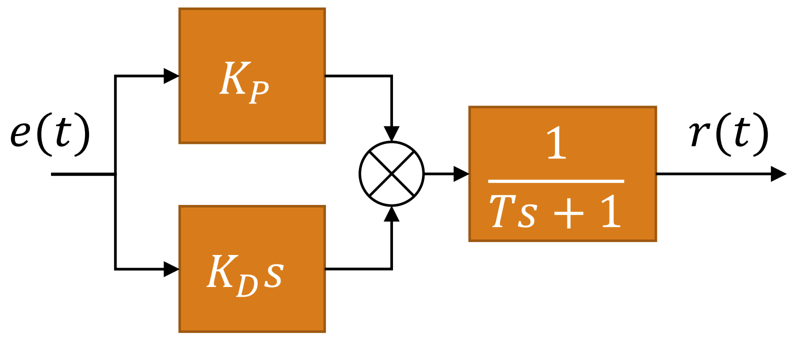
- where $T_D = \frac{K_D}{K_P}$ is derivative time constant

Basic types of controllers
PD Controller



- A PD controller improves speed and stability rather than steady-state accuracy
- The limitation of a P controller: increasing K_P speeds up the response but also increases overshoot and can cause oscillation
- We want the system to respond faster without becoming unstable
- So we add a derivative term to anticipate changes in the error – like predicting where the system is heading

Basic types of controllers
PD Controller



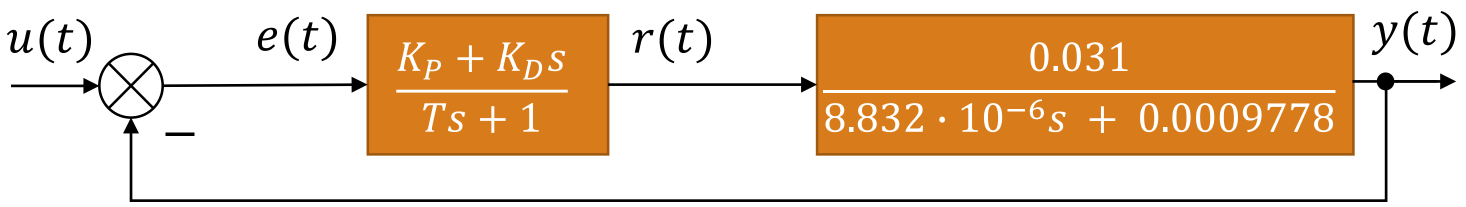
- **Real (practical) PD controller**
- To make ideal PD controller realizable, we need to add a small time constant T (a low-pass filter) in the denominator:

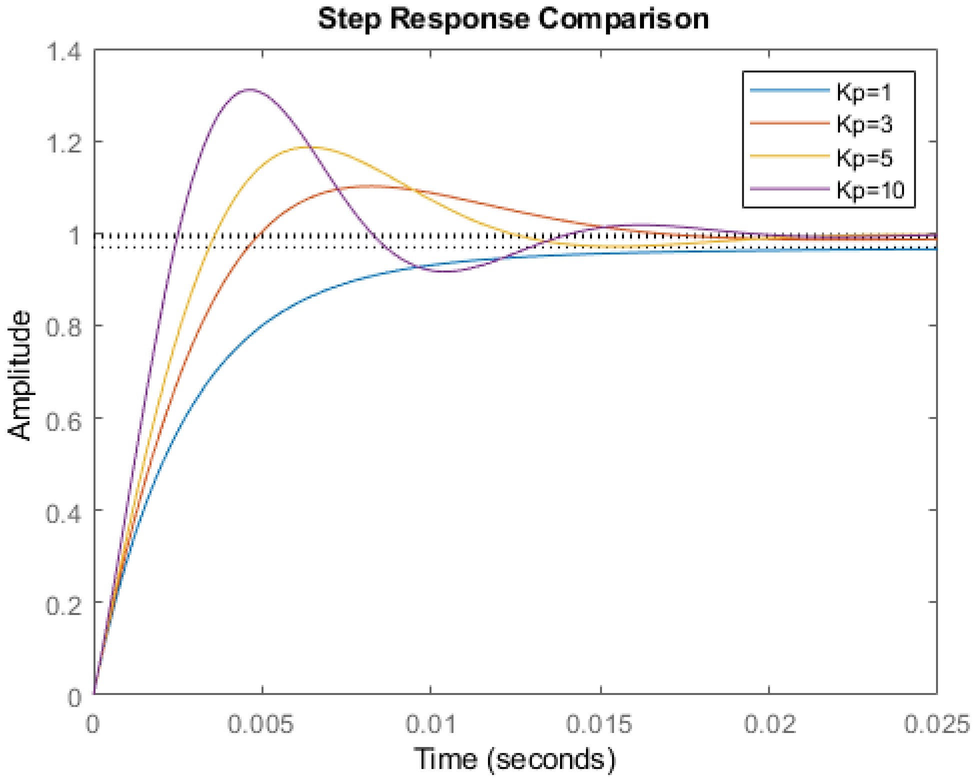
$$G(s) = \frac{R(s)}{E(s)} = \frac{K_P + K_D s}{Ts + 1}$$

where T is **filter time constant**

Basic types of controllers
PD Controller

- **DC motor control with real PD controller**



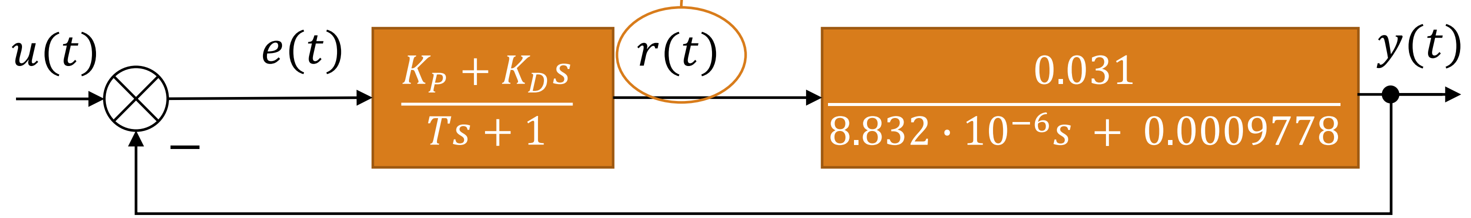


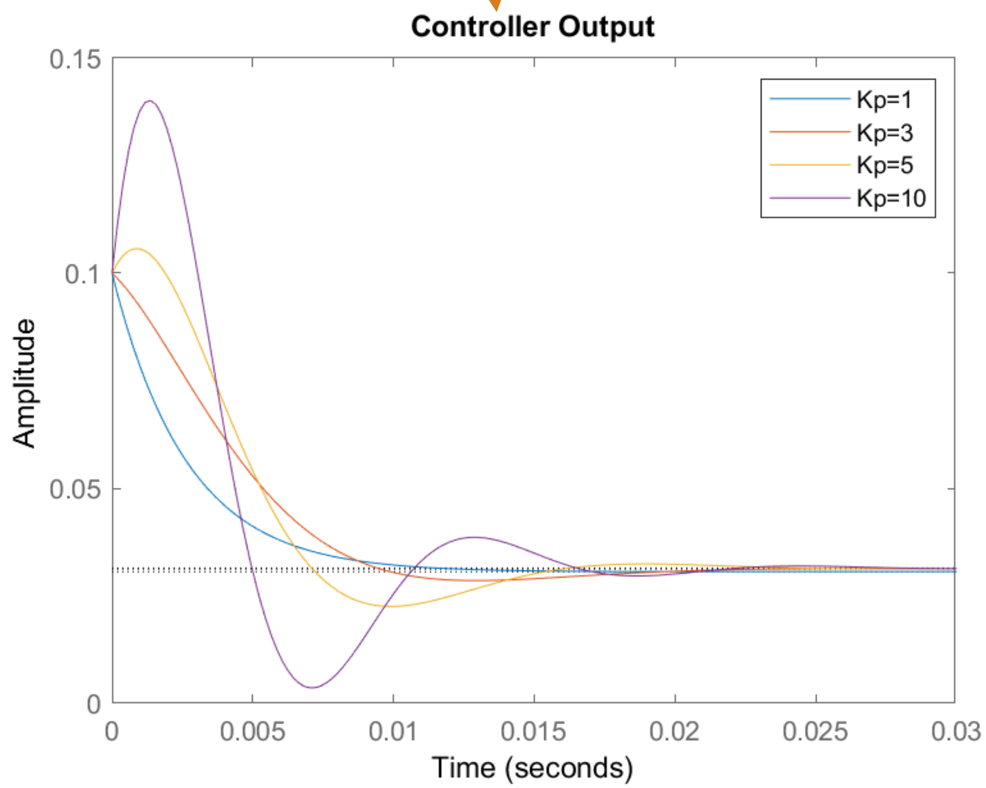
- $K_P = 1, 3, 5, 10$ and $K_D = 0.01$ and $T = 0.1\ s$
- Input $u(t)$: Unit step function

- A larger K_P leads to a faster response but may cause overshoot
- A larger K_P amplifies measurement noise

Basic types of controllers
PD Controller

- **DC motor control with real PD controller**



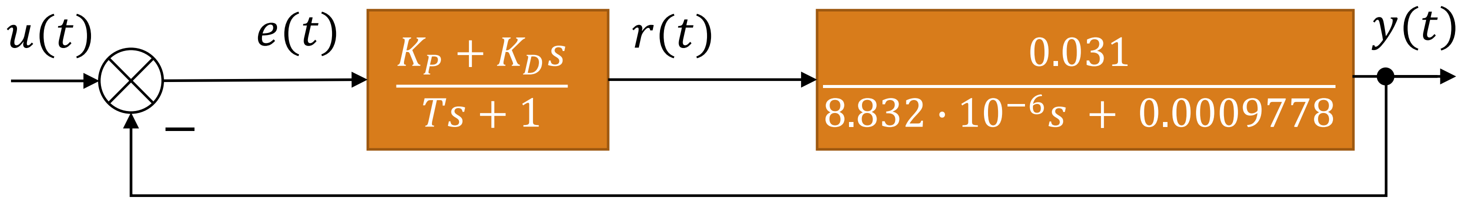


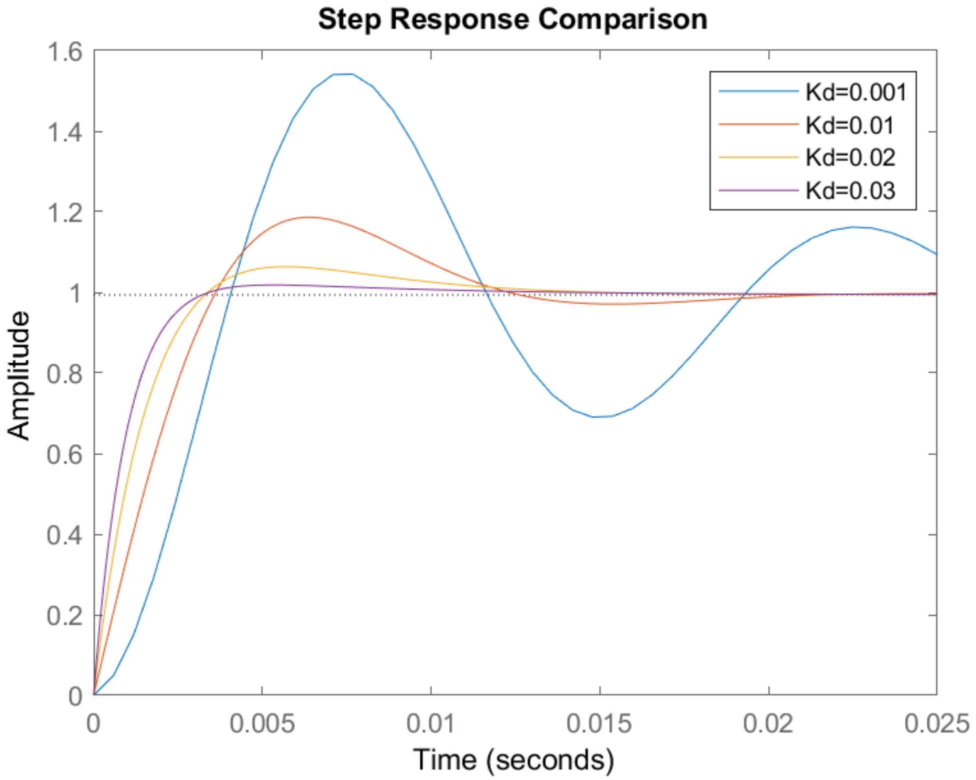
- $K_P = 1, 3, 5, 10$ and $K_D = 0.01$ and $T = 0.1\ s$
- Input $u(t)$: Unit step function

- A larger K_P leads to a faster response but may cause overshoot
- A larger K_P amplifies measurement noise

Basic types of controllers
PD Controller

- **DC motor control with real PD controller**





  - $K_P = 5$ and $K_D = 0.001, 0.01, 0.02, 0.03$ and $T = 0.1\ s$
  - Input $u(t)$: Unit step function

  - A larger K_D leads to a faster response and improved stability (lower overshoot)
  - A larger K_D amplifies measurement noise

Basic types of controllers
PD Controller

- MATLAB Code that was used to generate step responses

```
clear; clc; close all;
s = tf('s'); % definition of symbolic variable
G1=0.031/(8.832e-6*s+9.9184e-6+0.0009679); % DC Motor first-order approximation
Kp_list = [1 3 5 10];
Kd_list = [0.001 0.01 0.02 0.03];
f1 = figure; hold on; % open figure for plotting step responses
f2 = figure; hold on; % open another figure for ploting controller outputs
Kd = 0.01; %derivative gain
Tf = 0.1; % filter time constant
for Kp=Kp_list
    C = (Kp+Kd*s)/(Tf*s+1); % controller transfer function
    T=feedback(G1*C,1,-1); % feedback loop transfer function
    PDout=feedback(C,G1); % controller output vs. input transfer func.
    figure(f1); step(T);
    figure(f2); step(PDout);
end
figure(f1);
title('Step Response Comparison');
legend('Kp=1','Kp=3','Kp=5','Kp=10');
figure(f2);
title('Controller Output');
legend('Kp=1','Kp=3','Kp=5','Kp=10');
```

```
figure; hold on;
Kp = 5;
for Kd=Kd_list
    C = (Kp+Kd*s)/(Tf*s+1); % controller transfer function
    Tr=feedback(G1*C,1,-1); % feedback loop transfer function
    step(Tr);
end
title('Step Response Comparison');
legend('Kd=0.001','Kd=0.01','Kd=0.02','Kd=0.03');
```