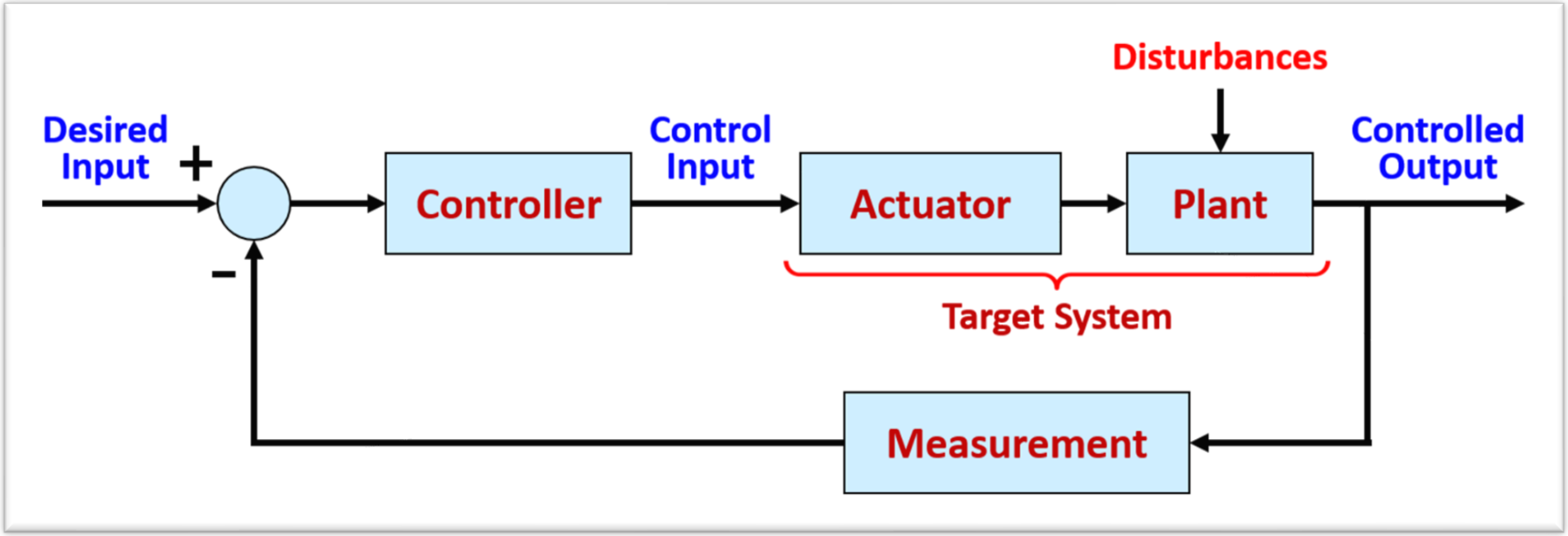
Analysis of Feedback 1

Lecture 6:

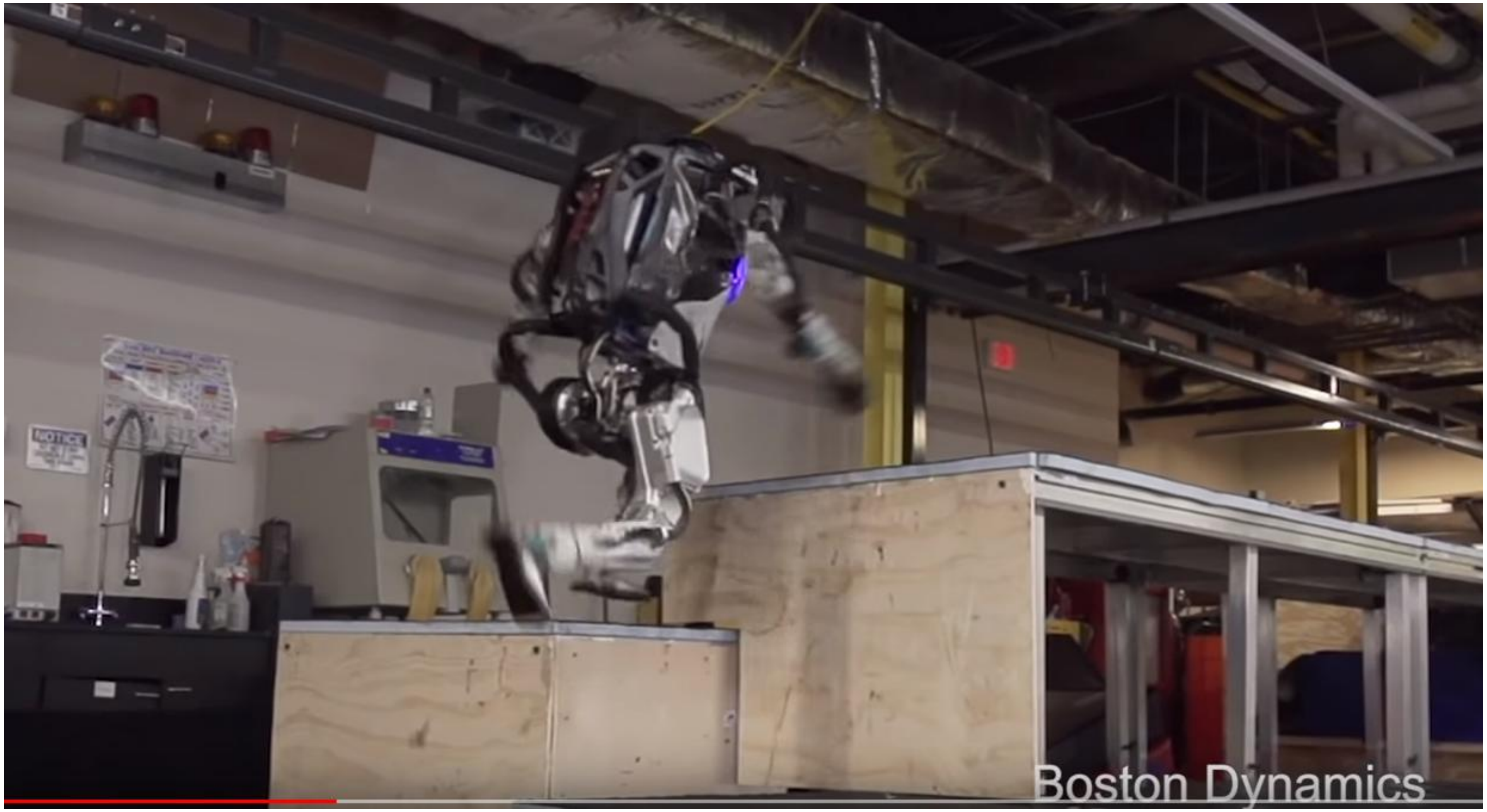
- **Overview of Feedback & PID Control**
- Example of PID Control System



Prof. Seunghoon Woo
Department of Automotive Engineering | College of Automotive Engineering
KOOKMIN UNIVERSITY

New Arrived Robots from Boston Dynamics!!

❖ Parkour Atlas



https://www.youtube.com/watch?v=LikxFZZO2sk

Newly Arrived Robots from Boston Dynamics!!

❖ SpotMini

https://www.youtube.com/watch?v=fUyU3lKzoio

Newly Arrived Robots from Boston Dynamics!!

❖ **Robustness Test for Disturbances!!**

https://www.youtube.com/watch?v=aFuA50H9uek

Basic Equations of Control

❖ Open-Loop System

- The controlled output is given by

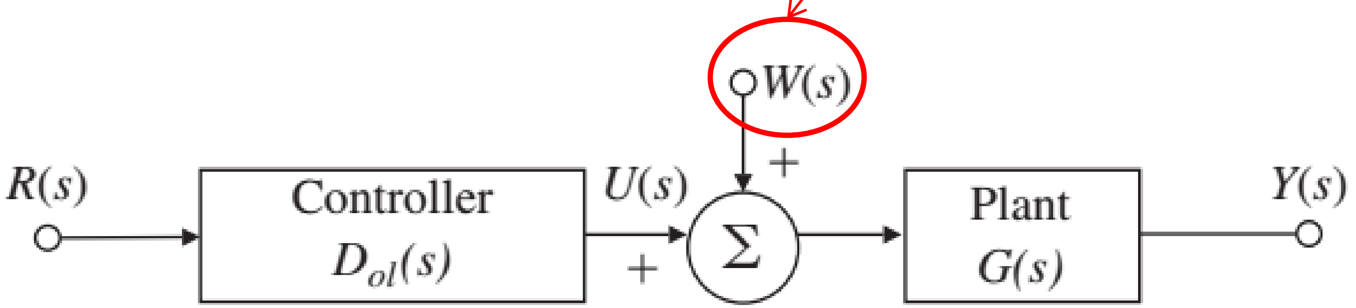
$$Y_{ol} = GD_{ol}R + GW,$$

- Error, difference btw **reference input (R)** and **output (Y)** is given by

$$\begin{aligned} E_{ol} &= R - Y_{ol}, \\ &= R - [GD_{ol}R + GW], \\ &= [1 - GD_{ol}]R - GW. \end{aligned}$$

- **Important Questions:**
  - How to minimize the error?
  (error → zero or R = Y)

Disturbance



Basic Equations of Control (cont'd)

- **Important Questions:**
  - How to minimize the error? (Objectives: error → zero)

❖ Closed-Loop System

- The controlled output is given by

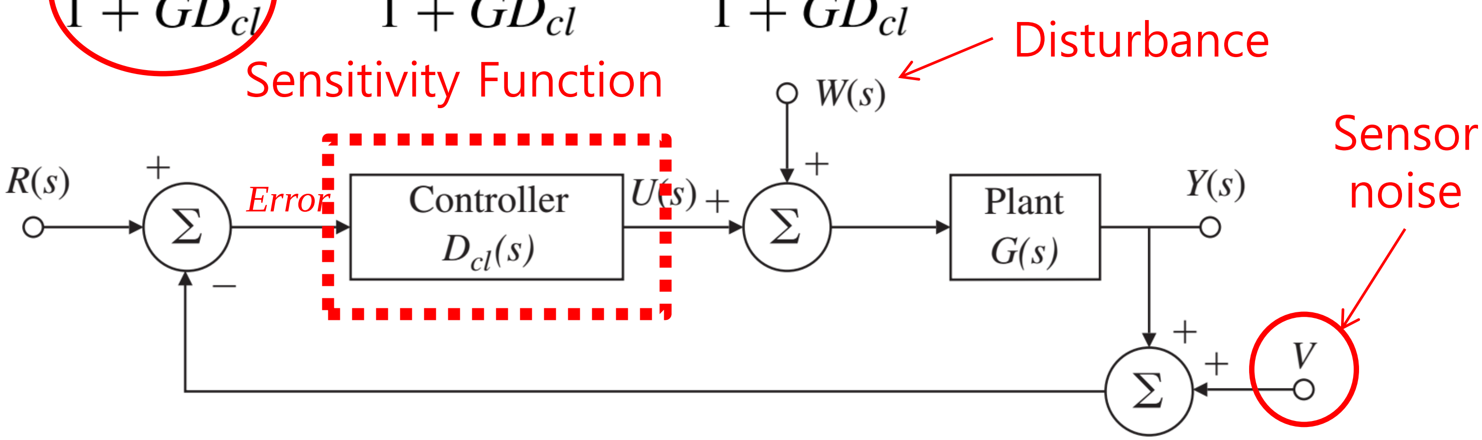
$$Y_{cl} = \frac{GD_{cl}}{1+GD_{cl}}R + \frac{G}{1+GD_{cl}}W - \frac{GD_{cl}}{1+GD_{cl}}V,$$

C-L Transfer Function

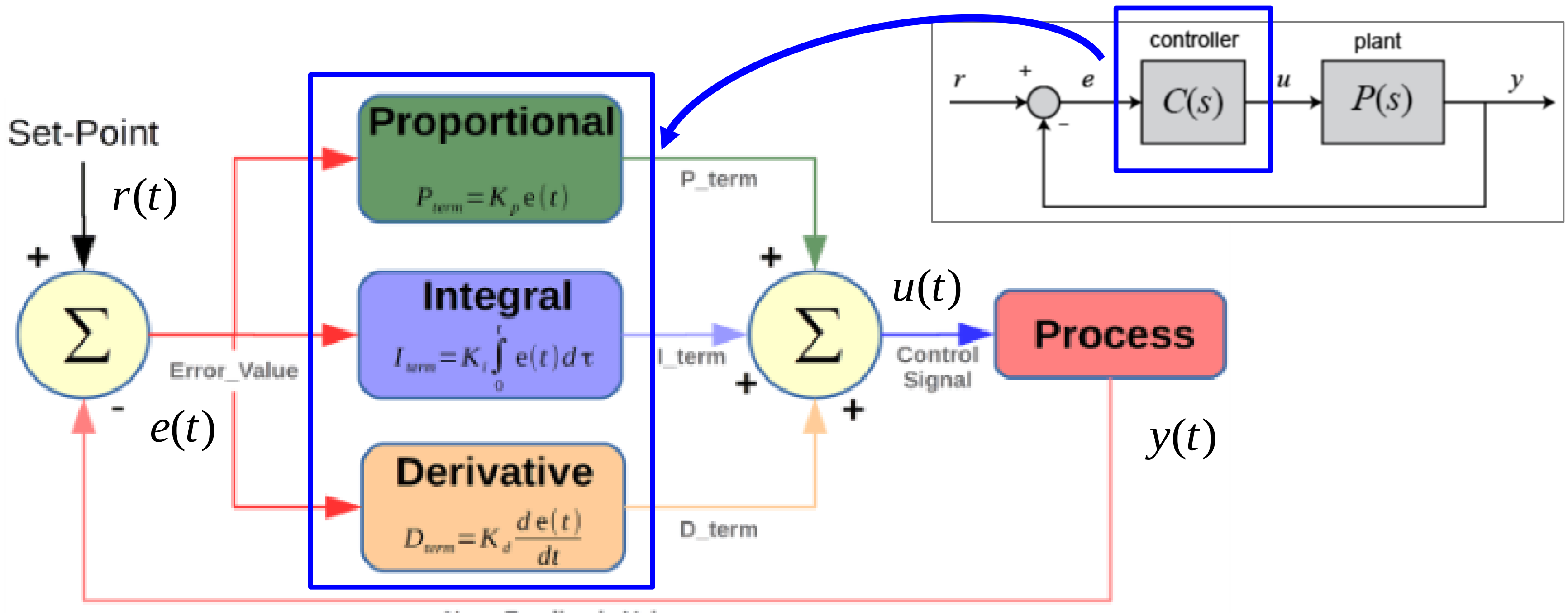
- Error, difference btw **reference input (R)** and **output (Y)** is given by

$$E_{cl} = R - \left[\frac{GD_{cl}}{1+GD_{cl}}R + \frac{G}{1+GD_{cl}}W - \frac{GD_{cl}}{1+GD_{cl}}V\right],$$
$$= \frac{1}{1+GD_{cl}}R - \frac{G}{1+GD_{cl}}W + \frac{GD_{cl}}{1+GD_{cl}}V.$$

Sensitivity Function



Three-Term Controller: PID Control



❖ The control signal of a **PID controller** in the time-domain

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

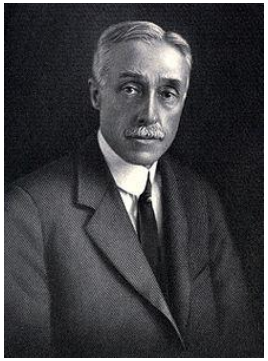
where,

error value, $e(t) = r(t) - y(t)$

Brief History of PID control

❖ **The First real PID-type Controlled by Elmer Sperry in 1911.**

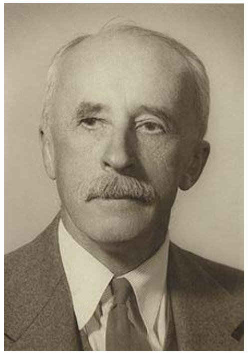
- Gyroscopic compass & stabilizer application



Elmer Ambrose Sperry (1860~1930)

❖ **The first theoretical analysis of a PID controller was published by Nicolas Minorsky in 1922.**

- Automatic steering systems for US Navy ships.
- Developed by observation of a helmsman, steering ships based on not only **current error (P-control)** but also, **past error (I)** as well as **the rate of change (D)**



Nicolas Minorsky (1885~1970)

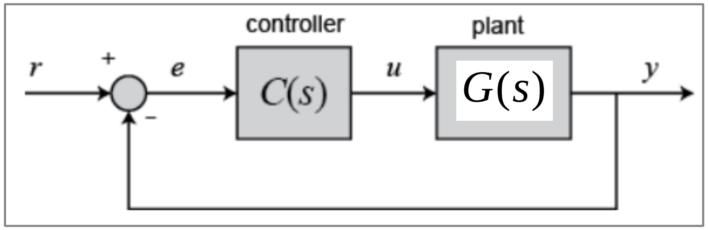
Source: https://en.wikipedia.org/wiki/PID_controller

Three-Term Controller: PID Control

❖ Proportional Control (P)

- Linearly proportional to system error

$$u(t) = k_P e(t),$$



$$C(s) = k_p$$

- Assuming plant transfer function can be given by

$$G(s) = \frac{A}{s^2 + a_1 s + a_2}.$$

$$\frac{Y(s)}{R(s)} = T(s) = \frac{C(s)G(s)}{1 + C(s)G(s)} = \frac{k_p A}{s^2 + a_1 s + a_2 + k_p A}$$

Applying model matching condition (i.e., comparing denominators)

$$s^2 + \underline{a_1} s + \underline{a_2 + k_P A}$$

Damping term including natural frequency

Natural frequency term

$$\boxed{T(s) = \frac{\omega_n^2}{s^2 + 2\zeta\omega_n s + \omega_n^2}.}$$

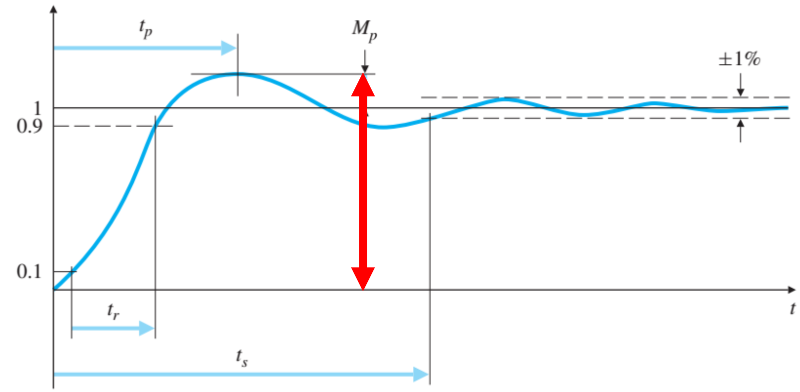
- **Important questions:**
  - ω_n vs. rising time ??

Dynamic Response (Revisited)

(5) Control Performance Index: Step Response (cont'd)

(3) Rise Time

$$y(t) = 1 - \frac{1}{\sqrt{1-\zeta^2}} e^{-\zeta\omega_n t} \sin(\omega_d t + \phi) \quad \text{for } \zeta < 1$$



$$y(t) = 1 @ \text{rising time } (T_r)$$

$$\sin(\omega_d T_r + \varphi) = 0$$

or

$$\therefore T_r = \frac{\pi - \varphi}{\omega_d}$$

where,

$$\omega_d = \omega_n \sqrt{1 - \varsigma^2}$$

$$\cos\varphi = \varsigma$$

$i)\ \varsigma = 0.5$

$\Rightarrow \varphi = \cos^{-1}(0.5) = 1.05$

$\therefore T_r = \dfrac{3.14 - 1.05}{\omega_n \sqrt{1 - 0.5^2}} \approx \dfrac{2.4}{\omega_n}$

$i)\ \varsigma = 0.7$

$\Rightarrow \varphi = \cos^{-1}(0.7) \approx 0.8$

$\therefore T_r = \dfrac{3.14 - 0.8}{\omega_n \sqrt{1 - 0.7^2}} \approx \dfrac{3.3}{\omega_n}$

❖ **Discussion & Question:**

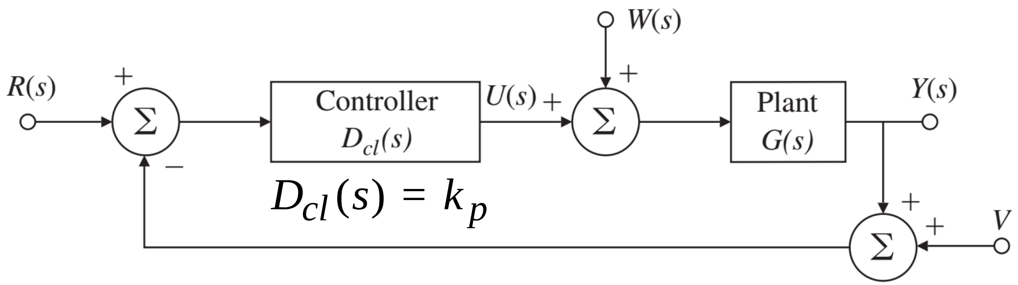
- How does damping coefficient & natural frequency work for the rising time ??

Three-Term Controller: PID Control

❖ Proportional Control (P) – cont'd

- Closed-Loop transfer function of P-control

$$\frac{Y(s)}{R(s)} = T(s) = \frac{k_p G(s)}{1 + k_p G(s)}$$

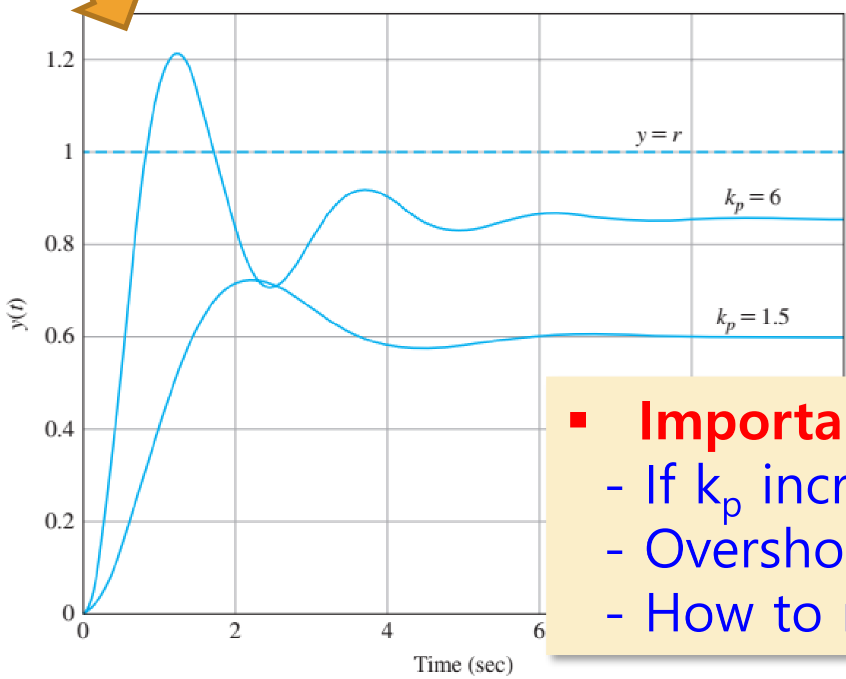


MATLAB SIMULINK **Simulation**

$$G(s) = \frac{A}{s^2 + a_1 s + a_2}.$$

where,

$a_1 = 1.4, a_2 = 1, A = 1$



- **Important questions:**
  - If k_p increases?
  - Overshoot value?
  - How to reject the error?

Three-Term Controller: PID Control

❖ Proportional Control (P) – cont'd

- **Closed-loop TF**

$$\frac{Y(s)}{R(s)} = T(s) = \frac{k_p G(s)}{1 + k_p G(s)}$$

- **Sensitivity TF**

$$\frac{E(s)}{R(s)} = S(s) = \frac{1}{1 + k_p G(s)}$$

- Using **final value theorem** at **G(0) = 1** & **Step input R(s) = 1/s**

$$\boxed{\lim_{t\to\infty} f(t) = \lim_{s\to 0} sF(s) \quad \text{if all the poles of sF(s) are in the left half plane (LHP)}}$$

$$y(\infty) = Y(0) = \cancel{s}\frac{k_p G(0)}{1 + k_p G(0)} \frac{1}{\cancel{s}} \qquad e(\infty) = E(0) = \cancel{s}\frac{1}{1 + k_p G(0)} \frac{1}{\cancel{s}}$$

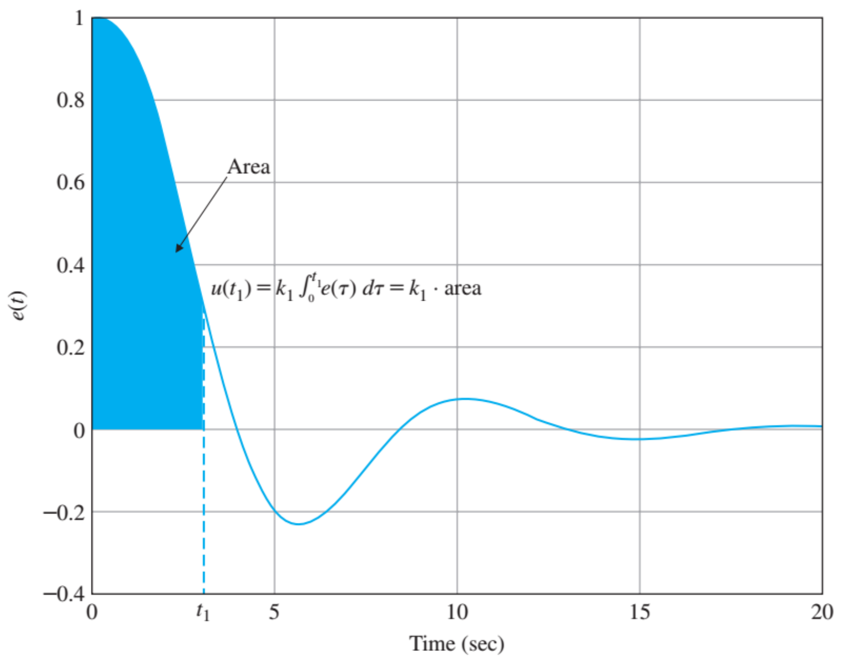
- **Important Findings:**
  - Proportional control (P) vs. regulation error ??
  - Can we have Zero SSE with P-control ?? (If not, How??)

Three-Term Controller: PID Control

❖ Integral Control (I)

- Control law of integral feedback is given by,

$$u(t) = k_I \int_{t_0}^{t} e(\tau)\, d\tau, \quad \xrightarrow{\mathcal{L}} \quad \frac{U(s)}{E(s)} = D_{cl}(s) = \frac{k_I}{s},$$



Three-Term Controller: PID Control

❖ Integral Control (I) – cont'd

- Closed-loop TF is given by,

$$\frac{Y(s)}{R(s)} = \mathcal{T}(s) = \frac{\frac{k_I}{s}G(s)}{1+\frac{k_I}{s}G(s)} = \frac{k_I G(s)}{s + k_I G(s)}.$$

- Sensitivity TF is given by,

$$\frac{E(s)}{R(s)} = \frac{1}{1+\frac{k_I}{s}G(s)} = \frac{s}{s+k_I G(s)},$$

From previous slide,

$$y(\infty) = Y(0) = \cancel{s}\frac{k_pG(0)}{1+k_pG(0)}\frac{1}{\cancel{s}}$$

$$e(\infty) = E(0) = \cancel{s}\frac{1}{1+k_pG(0)}\frac{1}{\cancel{s}}$$

- Using final value theorem with **G(0) = 1 & Step input R(s) = 1/s**

$$y(\infty) = \frac{k_I G(0)}{0 + k_I G(0)} = 1,$$

$$e(\infty) = \frac{0}{0 + k_I G(0)} = 0,$$

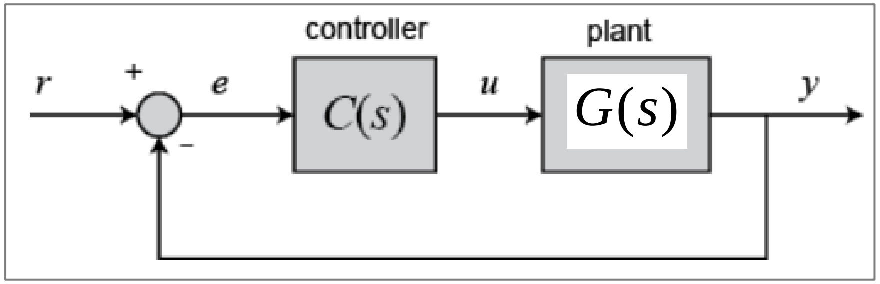
- **Important Findings:**
  - Integral control vs. SSE ??

Three-Term Controller: PID Control

❖ Integral Control (I) – cont'd

- Example:

$$C(s) = \frac{k_i}{s} \qquad G(s) = \frac{b}{s+a}$$



$$T(s)_{CL} = \frac{C(s)G(s)}{1 + C(s)G(s)}$$

$$= \frac{\dfrac{k_i}{s}\dfrac{b}{s+a}}{1 + \dfrac{k_i}{s}\dfrac{b}{s+a}}$$

$$= \frac{k_i b}{s^2 + \underline{as} + \underline{k_i b}}$$

Damping term including natural frequency

Natural frequency term

$$T(s) = \frac{\omega_n^2}{s^2 + 2\zeta\omega_n s + \omega_n^2}.$$

- **Important questions:**
  - ω_n vs. rising time ??
  - I-control vs. damping term ??
  - I-control vs. rising time ??

Three-Term Controller: PID Control

❖ Derivative Control (D)

- Control law of derivative feedback is given by,

$$u(t) = k_D \dot{e}(t), \quad \xrightarrow{\mathcal{L}} \quad \frac{U(s)}{E(s)} = D_{cl}(s) = k_D s.$$

- **Important Findings:**
  - Derivative control is almost never used by itself !!. Why??
    (1) The derivative does not supply information on the desired end state.
    (2) if e(t) were to remain constant, the output of a derivative controller would be zero.

  ☞ Thus, a proportional or integral control would be needed to provide a control signal at this time.

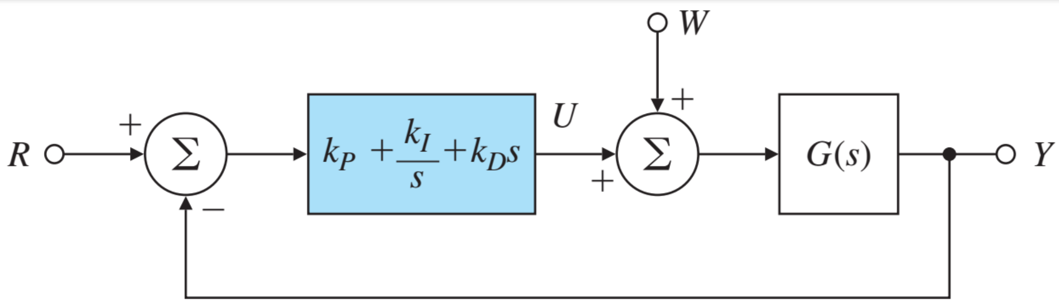
  ※ An important effect of the derivative term is that it gives a sharp response to suddenly changing signals.

Three-Term Controller: PID Control

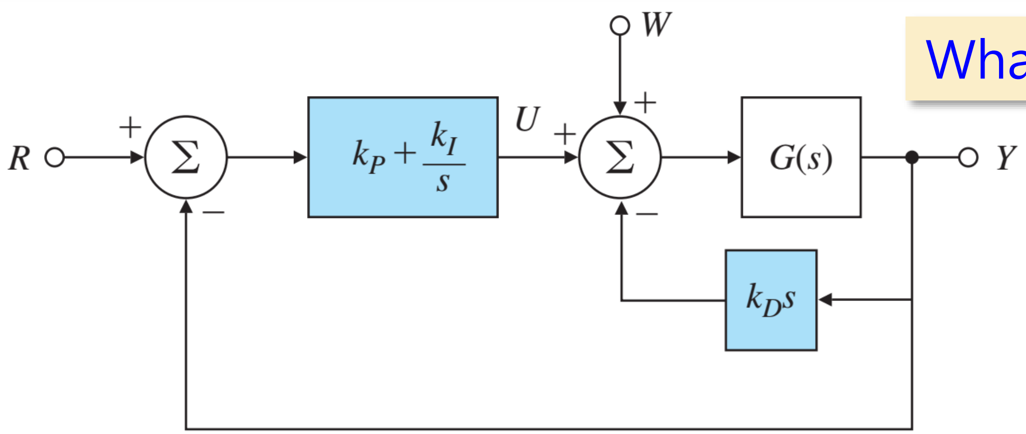
❖ Derivative Control (D) – cont'd

- Two types of derivative control applications – PID vs. PIV

(1) D-term in the forward path – General PID



(2) D-term in the feedback path (or velocity feedback) - PIV

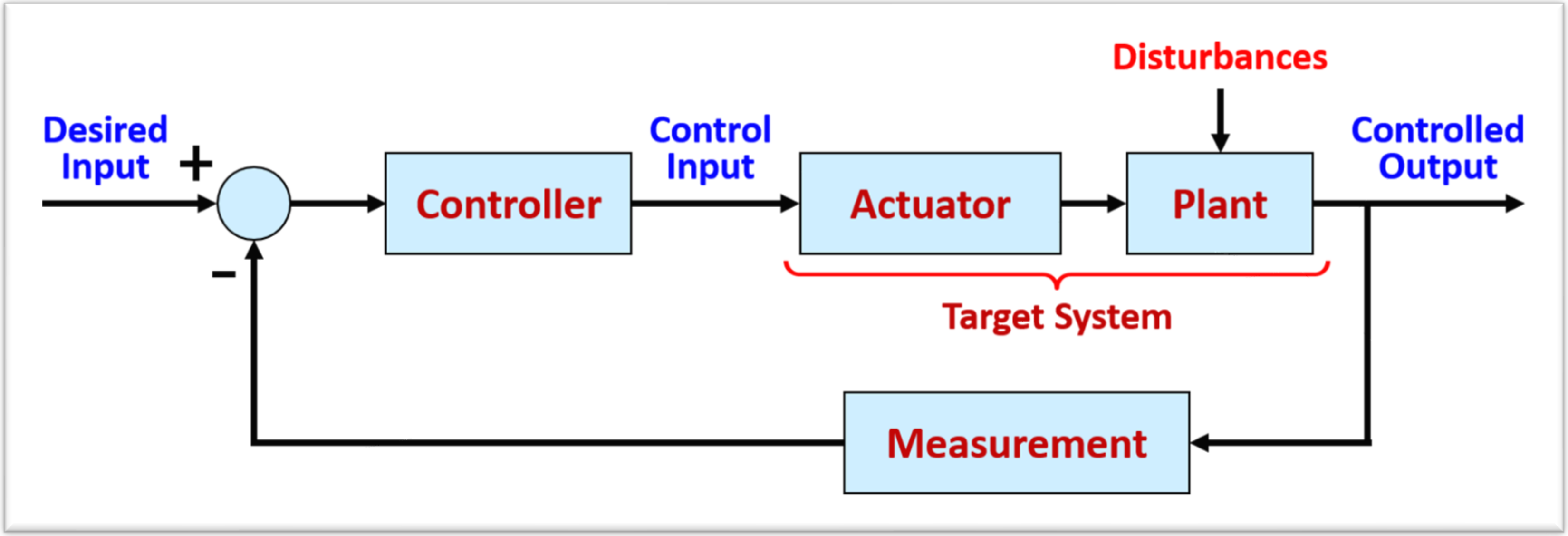


What advantage??

Analysis of Feedback 1

Lecture 6:

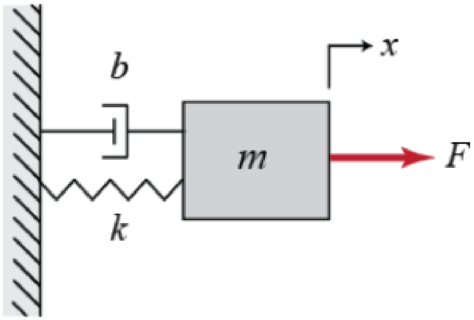
- Overview of Feedback & PID Control
- **Example of PID Control System**



Prof. Seunghoon Woo
Department of Automotive Engineering | College of Automotive Engineering
KOOKMIN UNIVERSITY

Three-Term Controller: PID Control

❖ Example 1: Spring-Damper-Mass system



- Step 1 : Equation of Motion (EOM) of this system $m\ddot{x}+b\dot{x}+kx=F$

- Step 2 : Taking Laplace transform of EOM $\left(ms^2+bs+k\right)X(s)=F(s)$

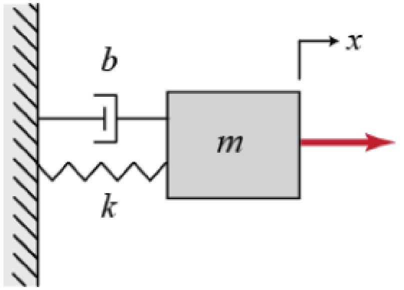
- Step 3 : TF btw the displacement {X(s)} and force {F(s)}

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

Three-Term Controller: PID Control

❖ Example 1: Spring-Damper-Mass system (cont'd)

- Let suppose each parameter, and get TF



$m = 1[\text{kg}]$

$b = 10\,[\text{Ns/m}]$

$k = 20\,\,[\text{N/m}]$

$F = 1\,\,[\text{N}]$

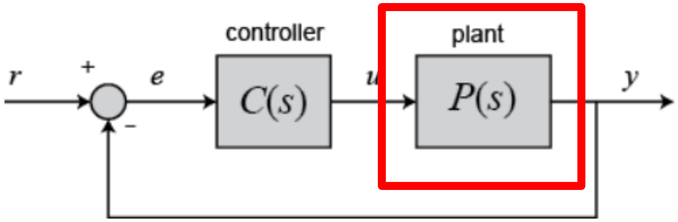
$$\frac{X(s)}{F(s)} = \frac{1}{s^2 + 10s + 20}$$

$$G(s) = \frac{1}{s^2 + 2\varsigma\omega_n s + \omega_n^2}$$

- **Objectives: Design PID controller to obtain**

1. Rise time < 0.2 sec
2. Settling time < 0.5 sec
3. Maximum overshoot < 5%
4. No steady-state error < 0.01

Three-Term Controller: PID

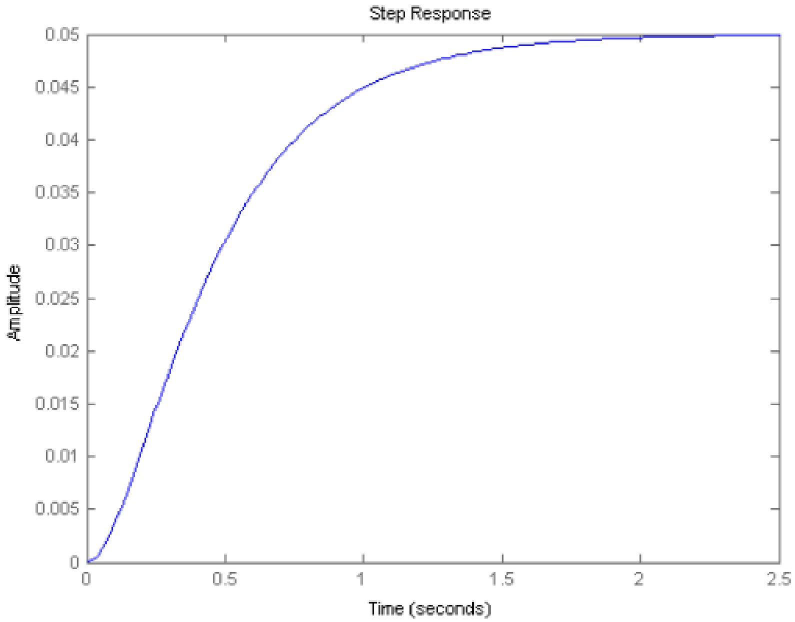


❖ Example 1: Spring-Damper-Mass system (cont'd)

- Check : Open-Loop Step Response

```
s = tf('s');
P = 1/(s^2 + 10*s + 20);
step(P)
```

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



❖ Findings:

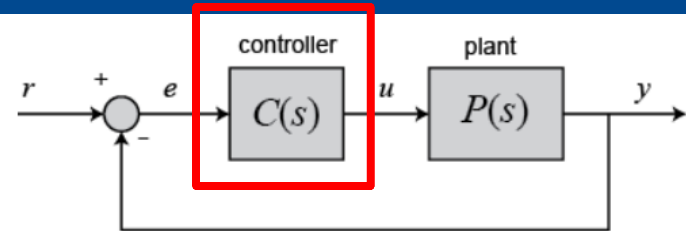
- Rise time is ~1.5sec
- Settling time is ~2sec
- Steady-state error is about 0.95

❖ Design PID controller:

- reduce rise time (<0.2)
- reduce settling time (<0.5)
- eliminate SSE (<0.01)

Three-Term Controller: PID Control

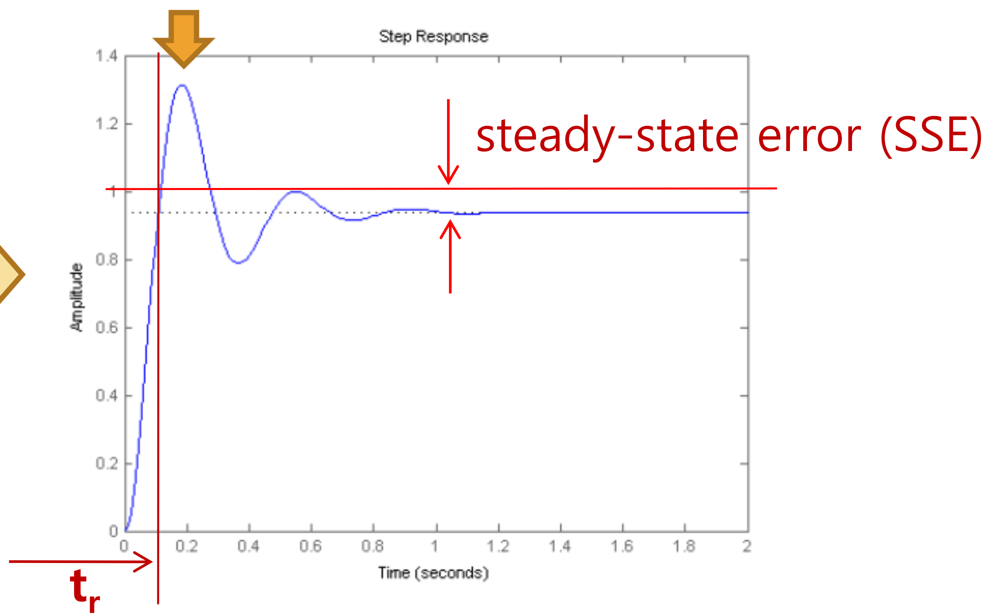
❖ Example 1: Spring-Damper-Mass



▪ Case I : Proportional (P)

$$C(s) = K_p \quad P(s) = \frac{1}{s^2 + 10s + 20} \quad \Rightarrow \quad \frac{X(s)}{F(s)} = \frac{C(s)P(s)}{1 + C(s)P(s)} = \frac{K_p}{s^2 + 10s + (20 + K_p)}$$

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

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



- Kp reduces rise time (1.5 → 0.1sec) and SSE (0.95 → 0.05), but overshoot (30%) !! Then, How to reduce overshoot ??

Three-Term Controller: PID Control

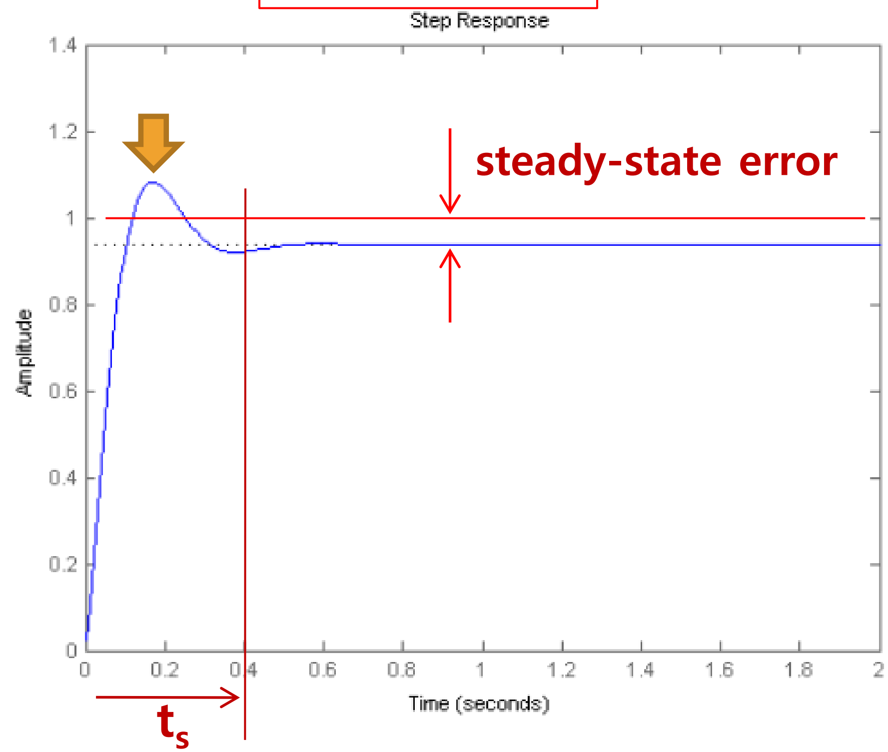
❖ Example 1: Spring-Damper-Mass system (cont'd)

- **Case II : Proportional + Derivative (PD)** $C(s)_{PD} = K_p + K_d s$

$$\left.\frac{X(s)}{F(s)}\right|_{CL} = \frac{K_d s + K_p}{s^2 + (10 + K_d)s + (20 + K_p)}$$

$$\boxed{G(s)_{CL} = \frac{K_d s + K_p}{s^2 + 2\varsigma\omega_n s + \omega_n^2}}$$

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

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



- PD reduces overshoot (30% → 15%) and settling time (1.0 → 0.4sec) !!

  But, there is still the steady-state error (~0.05). How to eliminate SSE ??

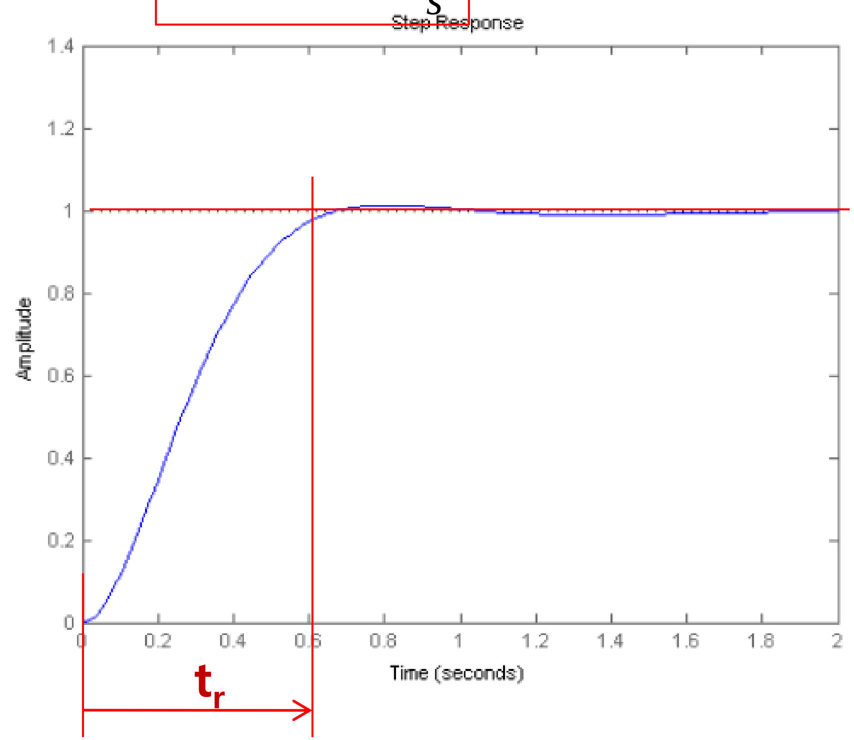
Three-Term Controller: PID Control

❖ Example 1: Spring-Damper-Mass system (cont'd)

- **Case III : Proportional + Integral (PI)** $C(s)_{PI} = K_p + \frac{K_i}{s}$

$$\left.\frac{X(s)}{F(s)}\right|_{CL} = \frac{K_p s + K_i}{s^3 + 10s^2 + (K_p + 20)s + K_i}$$

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


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



- PI control eliminates the steady-state error (0.05 → almost zero) !!
  But, rising time (0.1 → 0.6) & settling time (0.4 → 0.8) becomes longer !!

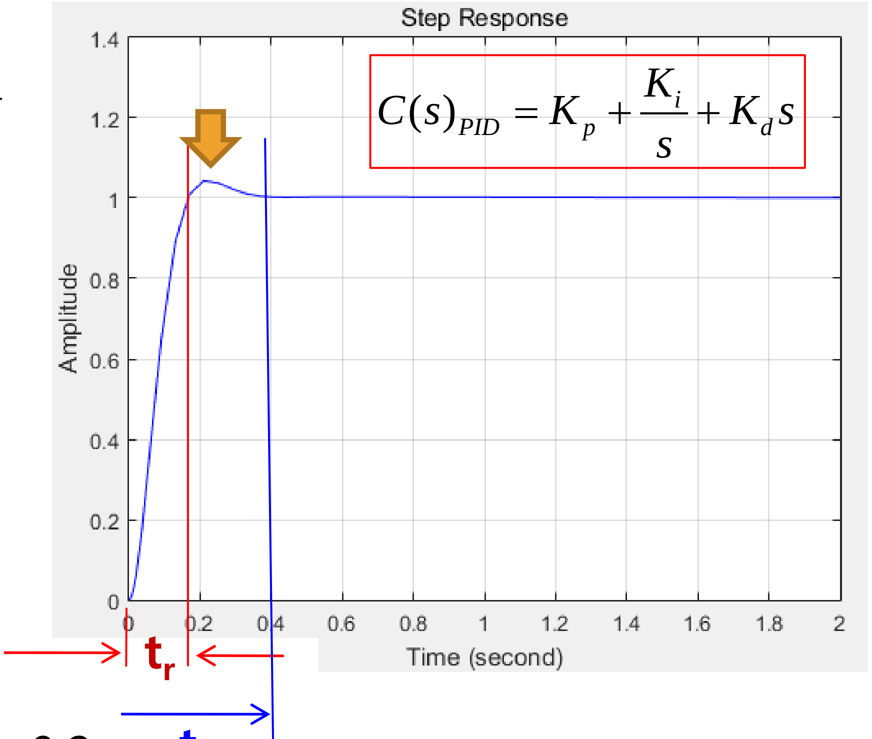
Three-Term Controller: PID Control

❖ Example 1: Spring-Damper-Mass system (cont'd)

- **Case IV** : **Proportional + Integral + Derivative (PID)**

$$\left.\frac{X(s)}{F(s)}\right|_{CL} = \frac{K_d s^2 + K_p s + K_i}{s^3 + (10 + K_d)s^2 + (20 + K_p)s + K_i}$$

```
Kp = 350;
Ki = 300;
Kd = 15 ;
C = pid(Kp,Ki,Kd)
T = feedback(C*P,1);

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

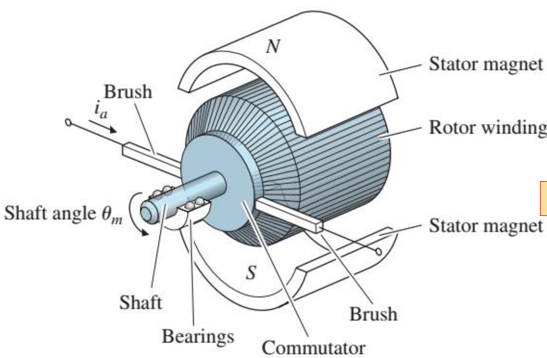


- **All goals are satisfied !!**

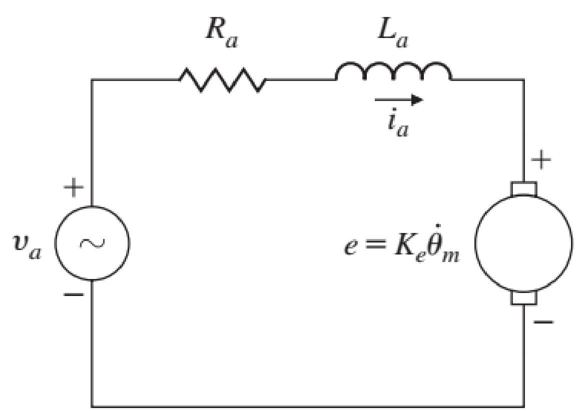
1. Fast rise time (t_r) ☞ less than 0.2 sec t_s
2. Settling time ☞ less then 0.4 sec
3. Maximum overshoot ☞ less than 5%
4. No steady-state error ☞ almost zero (< 0.001)

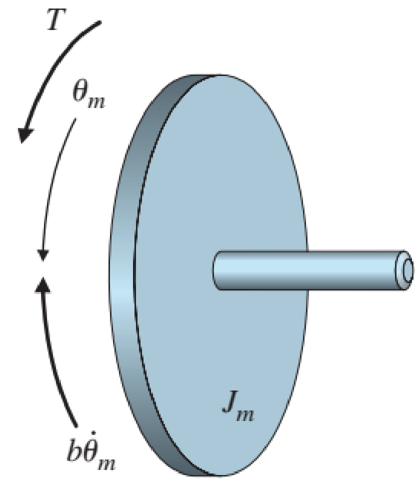
Three-Term Controller: PID Control

❖ Example 2: DC Motor – Speed Control



FBD





- **Part I: Mechanical motion**

$$J_m\ddot{\theta}_m + b\dot{\theta}_m = K_t i_a.$$

$$\Big\Downarrow \mathcal{L}$$

$$\frac{\theta_m(s)}{I_a(s)} = \frac{K_t}{J_m s^2 + bs}$$

- **Part II: Electric circuit equation**

$$L_a\frac{di_a}{dt} + R_a i_a = v_a - K_e\dot{\theta}_m.$$

$$\Big\Downarrow \mathcal{L}$$

$$(L_a s + R_a)I_a(s) = V_a(s) - (K_e s\theta_m)(s)$$

Three-Term Controller: PID Control

❖ Example 2: DC Motor – Speed Control (cont'd)

$$(L_a s + R_a)I_a(s) = V_a(s) - (K_e s\theta_m)(s) \qquad I_a(s) = \frac{J_m s^2 + bs}{K_t}\theta(s)$$

Arranging

$$\frac{\Theta_m(s)}{V_a(s)} = \frac{K_t}{s[(J_m s + b)(L_a s + R_a) + K_t K_e]}.$$

But, practically $L_a << J_m$ or R_a, Thus it is negligible !!

Re-arranging

$$\frac{\Theta_m(s)}{V_a(s)} = \frac{\frac{K_t}{R_a}}{J_m s^2 + \left(b + \frac{K_t K_e}{R_a}\right)s} = \frac{K}{s(\tau s + 1)},$$

Where,

$$\boxed{\frac{\Omega(s)}{V_a(s)} = s\frac{\Theta_m(s)}{V_a(s)} = \frac{K}{\tau s + 1}.}$$

Velocity Model

$$K = \frac{K_t}{bR_a + K_t K_e},$$

$$\tau = \frac{R_a J_m}{bR_a + K_t K_e}.$$

Three-Term Controller: PID Control

❖ Example 2: DC Motor – Speed Control (cont'd)

- **Transfer function of DC motor speed control (output)**

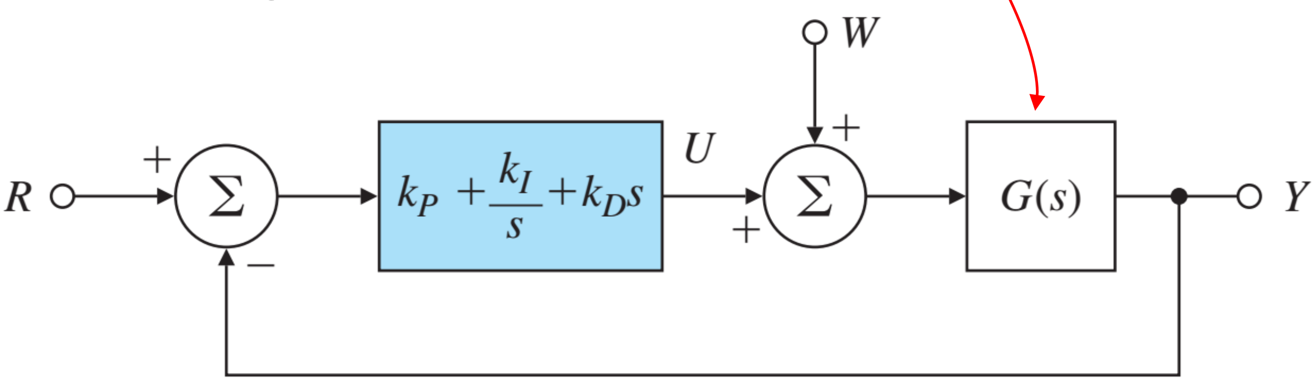
$$\frac{\Omega(s)}{V_a(s)} = s\frac{\Theta_m(s)}{V_a(s)} = \frac{K}{\tau s + 1}.$$

$$K = \frac{K_t}{bR_a + K_t K_e},$$

$$\tau = \frac{R_a J_m}{bR_a + K_t K_e}.$$

$J_m = 1.13 \times 10^{-2}$ N·m· sec^2 /rad,	$b = 0.028$ N·m·sec/rad,	$L_a = 10^{-1}$ H,
$R_a = 0.45\ \Omega$,	$K_t = 0.067$ N·m/amp,	$K_e = 0.067$ V·sec/rad.

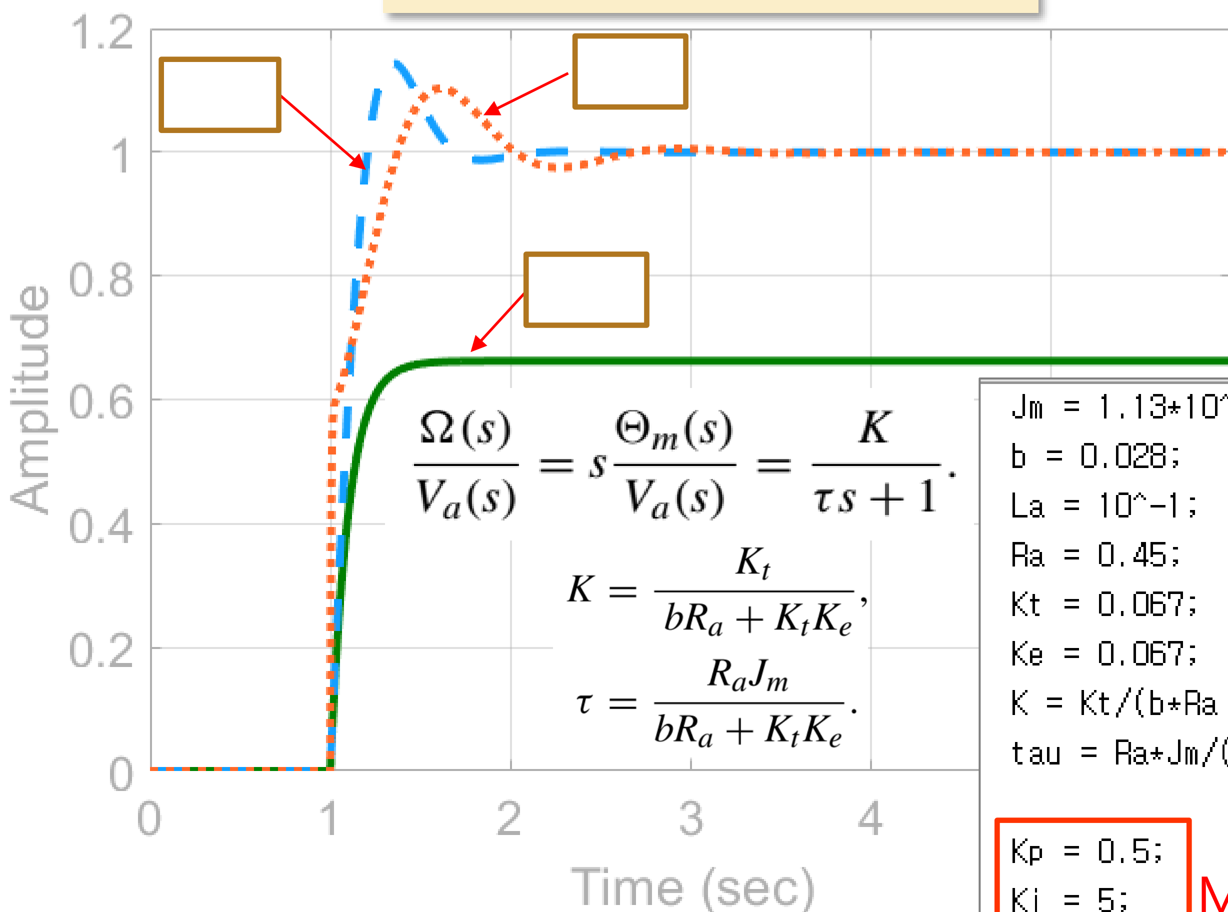
- **Block diagram of closed-loop with PID controller**



Three-Term Controller: PID Control

❖ Example 2: DC Motor – Speed Control (cont'd)

P vs PI vs PID ??



$$\frac{\Omega(s)}{V_a(s)} = s\frac{\Theta_m(s)}{V_a(s)} = \frac{K}{\tau s + 1}.$$

$$K = \frac{K_t}{bR_a + K_t K_e},$$

$$\tau = \frac{R_a J_m}{bR_a + K_t K_e}.$$

```
Jm  = 1.13*10^-2;      % Nm.sec^2/rad
b = 0.028;             % Nm.sec/rad
La = 10^-1;            % H
Ra = 0.45;             % ohm
Kt  = 0.067;           % Nm/A
Ke  = 0.067;           % V.sec/rad
K = Kt/(b*Ra + Kt*Ke);
tau = Ra*Jm/(b*Ra + Kt*Ke);

Kp  = 0.5;
Ki  = 5;
Kd  = 0.1;
```

Manually tuned !!

PID Controller on Discrete Time

❖ Implementation

- The analysis for designing a **digital implementation** of a PID controller in a **microcontroller (MCU)** or **FPGA device** requires the standard form of the PID controller to be **discretized**.

- **The integral term** is discretized, with a sampling time Δt, as follows,

$$\int_0^{t_k} e(t)\,dt = \sum_{k=1}^{n} e(t_k)\,\Delta t$$

- **The derivative term** is approximated as,

$$\frac{de(t)}{dt} = \frac{e(t_k) - e(t_{k-1})}{\Delta t}$$

It is called "backward finite difference"
(It is the nature of the feedback)

A Spartan FPGA from Xilinx

• **FPGA**:
Field
Programmable
Gate
Array

https://en.wikipedia.org/wiki/Field-programmable_gate_array

PID Controller on Discrete Time (cont'd)

❖ Implementation (cont'd)

- Thus, a PID algorithm for implementation of the **discretized PID controller in a MCU** is obtained by differentiating $y(t)$, using the numerical definitions of the first and second derivative and solving for $y(t_k)$ and finally obtaining:

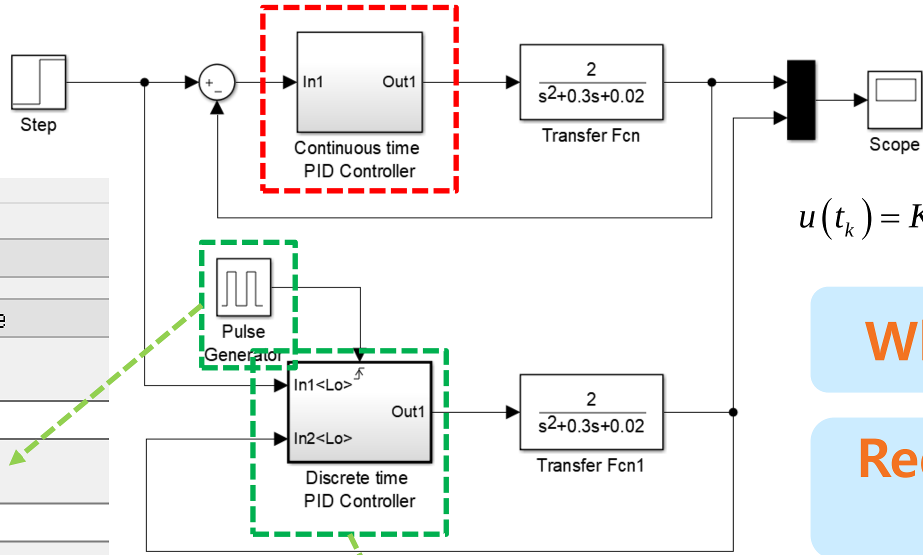
$$u(t) = K_p e(t) + K_i \int e(t)dt + K_d \frac{de(t)}{dt}$$

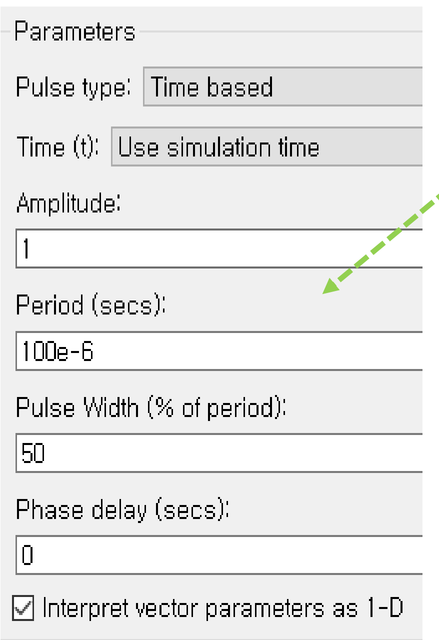
$$\Downarrow$$

$$u(t_k) = K_p e(t_k) + K_i \sum_{k=1}^{n} e(t_k)\Delta t + K_d \frac{e(t_k) - e(t_{k-1})}{\Delta t}$$

PID Controller on Discrete Time (cont'd)

❖ Example: Implementation with C-code

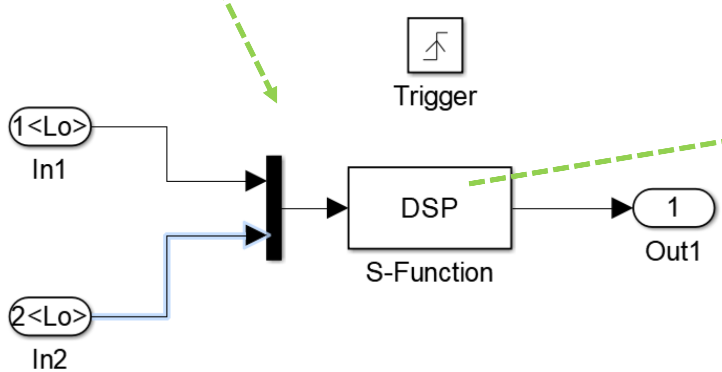


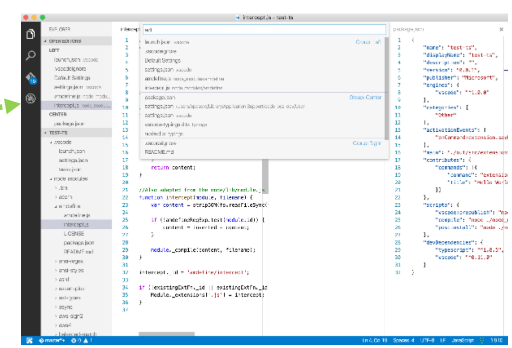


$$u(t_k) = K_p e(t_k) + K_i \sum_{k=1}^{n} e(t_k)\Delta t + K_d \frac{e(t_k) - e(t_{k-1})}{\Delta t}$$

What is a S-Function?

Require an installation of MEX-compiler.





PID Controller on Discrete Time (cont'd)

❖ **PID pseudo code for the discrete-time,**

$$u(t_k) = K_p e(t_k) + K_i \sum_{k=1}^{n} e(t_k)\Delta t + K_d \frac{e(t_k) - e(t_{k-1})}{\Delta t}$$

```
error_prior = 0
integral = 0
K_P = Some value you need to come up (see tuning section below)
K_I = Some value you need to come up (see tuning section below)
K_D = Some value you need to come up (see tuning section below)

while(1) {
  error = desired_value – actual_value
  integral = integral + (error*iteration_time)
  derivative = (error – error_prior)/iteration_time
  output = K_P*error + K_I*integral + K_D*derivative + bias
  error_prior = error
  sleep(iteration_time)
}
```

PID Controller on Discrete Time (cont'd)

❖ PID C++ code for the discrete-time: PID + Saturation

```
double PIDImpl::calculate( double setpoint, double pv )
{

    // Calculate error
    double error = setpoint - pv;

    // Proportional term
    double Pout = _Kp * error;

    // Integral term
    _integral += error * _dt;
    double Iout = _Ki * _integral;

    // Derivative term
    double derivative = (error - _pre_error) / _dt;
    double Dout = _Kd * derivative;

    // Calculate total output
    double output = Pout + Iout + Dout;

    // Restrict to max/min
    if( output > _max )
        output = _max;
    else if( output < _min )
        output = _min;

    // Save error to previous error
    _pre_error = error;

    return output;
}
```

Measured output (or filtered output)

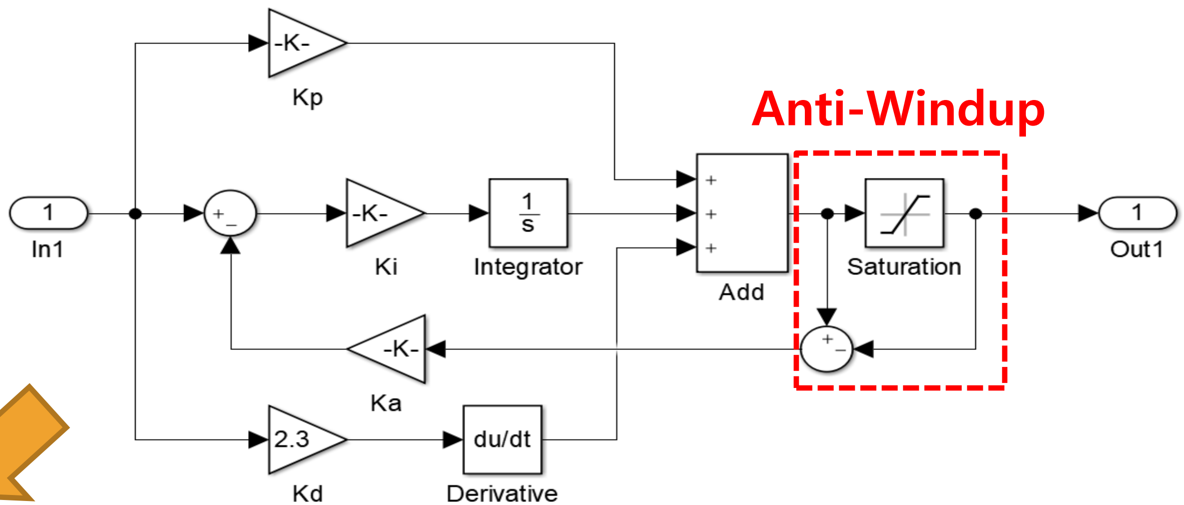
Saturation (Restriction)

https://gist.github.com/bradley219/5373998

PID Controller on Discrete Time (cont'd)

❖ Example: PID with Anti-Windup in C-code (cont'd)

- **With Anti-Windup**



- **C-language code**

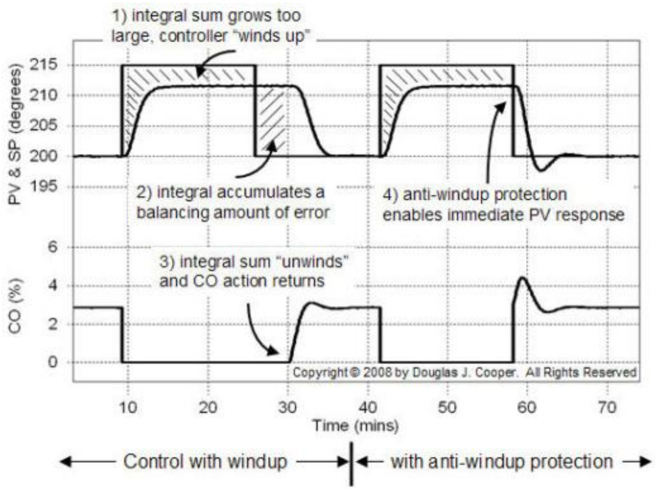
```c
#define TS          100e-6
#define Out_MAX     10.0

double err;
double err_old;
double out;
double kp;
double ki;
double kd;
double integ;
double ref;
double feed;
double ka;
double out_sat;
```

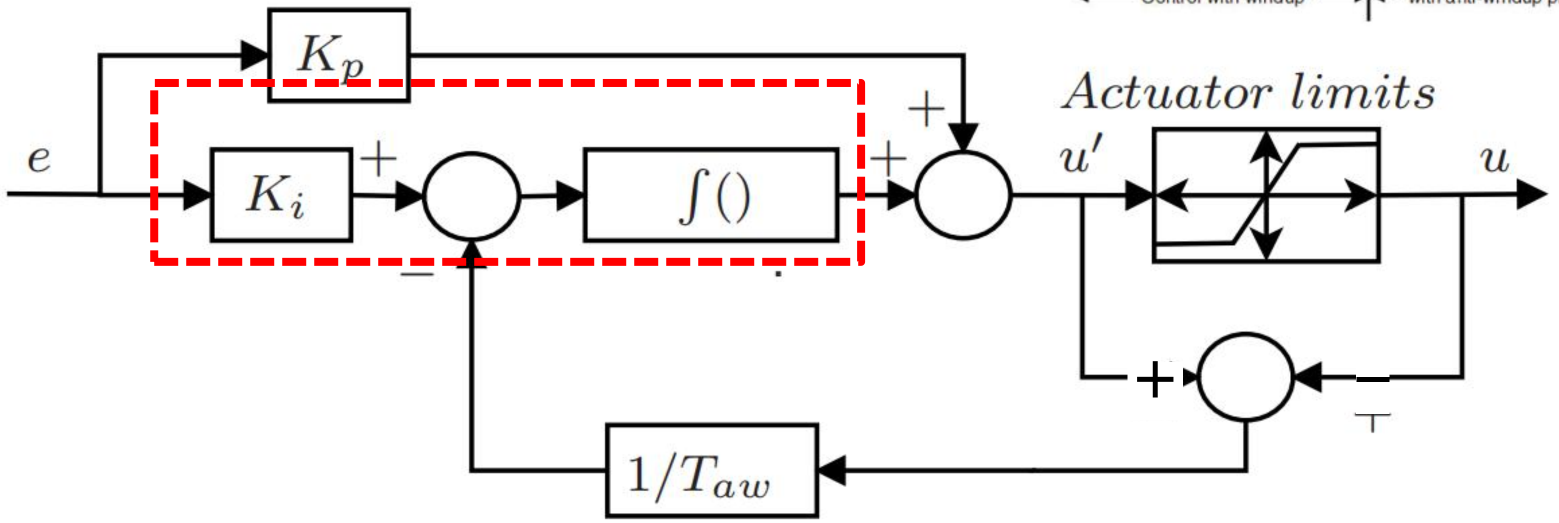
```c
err = 0.0;
kp = 10.01;
ki = 11.12 * TS;
kd = 2.3 / TS;
ka = 1/(6*kp);

err = ref - feed;
integ += err - (ka*(out-out_sat));
out = (kp*err) + (ki*integ) + (kd*(err-err_old));
out_sat = (out > Out_MAX) ? Out_MAX : (out < -Out_MAX) ? -Out_MAX : out;
err_old = err;
```

PID Controller on Discrete Time (cont'd)

❖ Example: Implementation with C-code (cont'd)

- **Anti-Windup for the Integral Control**

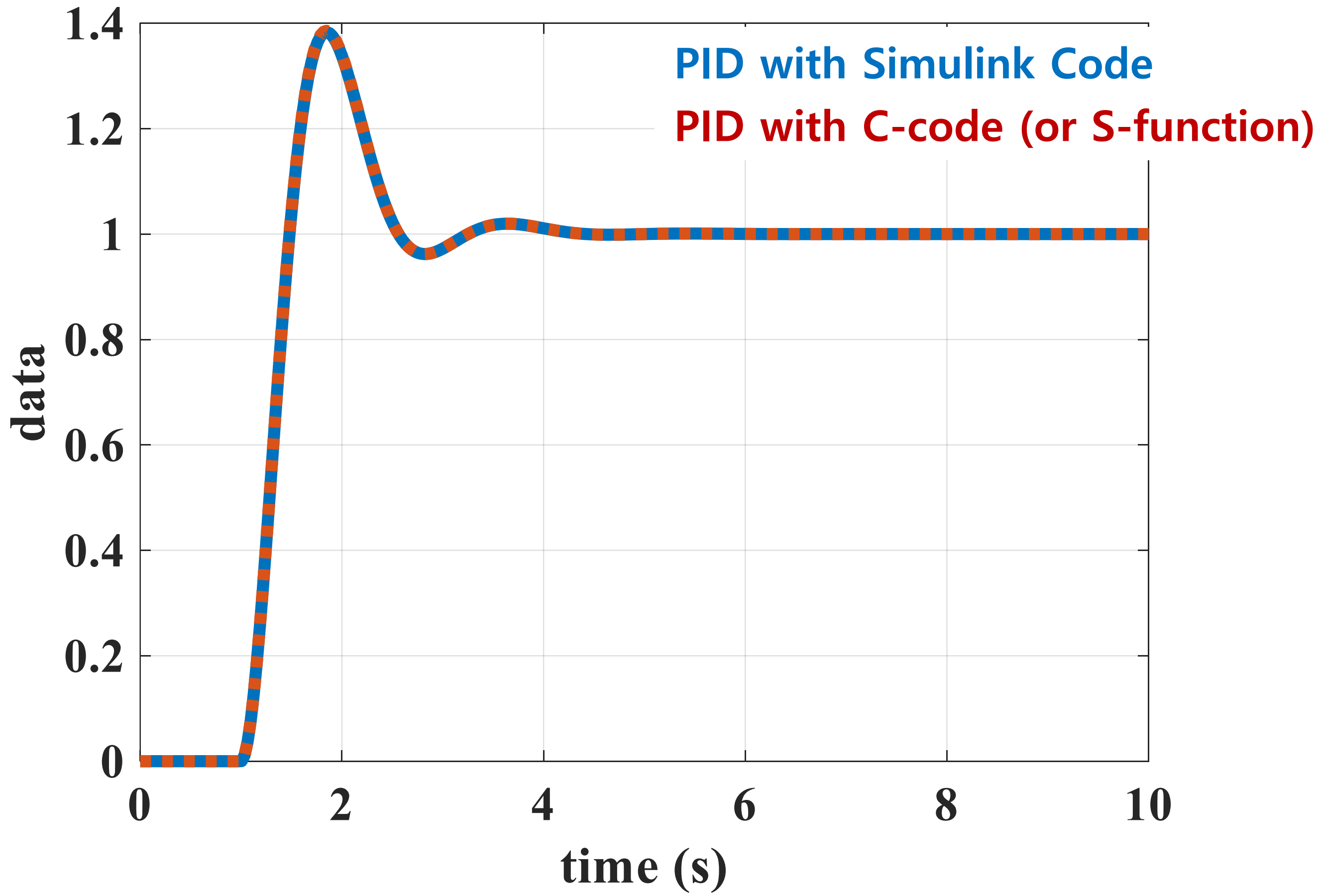




https://www.researchgate.net/figure/PI-controller-with-back-calculation-anti-windup-scheme_fig3_267642783

PID Controller on Discrete Time (cont'd)

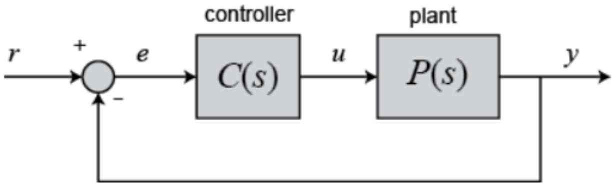
❖ **Example: Implementation with C-code (cont'd)**



Summary: PID Controller

❖ Summary of PID

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



- **Proportional (P)**
  - reduce the rise time and reduce the steady-state error (SSE)
  - but, never eliminate the SSE
    (e.g., certain amount of error will be actuated by input constraints)
- **Integral (I)**
  - eliminate the SSE for a constant or step input,
    (by accumulating error between r(t) and y(t)
  - but, worse transient response(settling time or overshoot)
- **Derivative (D)**
  - Improve the transient response (settling time), reduce the overshoot
  - but, amplify higher frequency measurement (sensor) or process noise

Limitations of PID & Modifications

- **PID is Feedback !! ☞ Thus, response is reactive!!**
  - ✓ Feed-forward control with the knowledge about the process model will be able to increase responding performance !!

- **PID has constant parameters (or gains) !!**

  ☞ Thus, if system variations exist, performance gets worse !!

  - ✓ Changing the control parameters (i.e., gains) based on the process variations (e.g., gain scheduling and adaptive control)

- **PID control is no direct knowledge of the process !!**
  - ✓ Identify the process model and optimize controller gain (e.g., by using Matlab tool(e.g., PID tuner), auto-tuning method, model matching condition and etc)

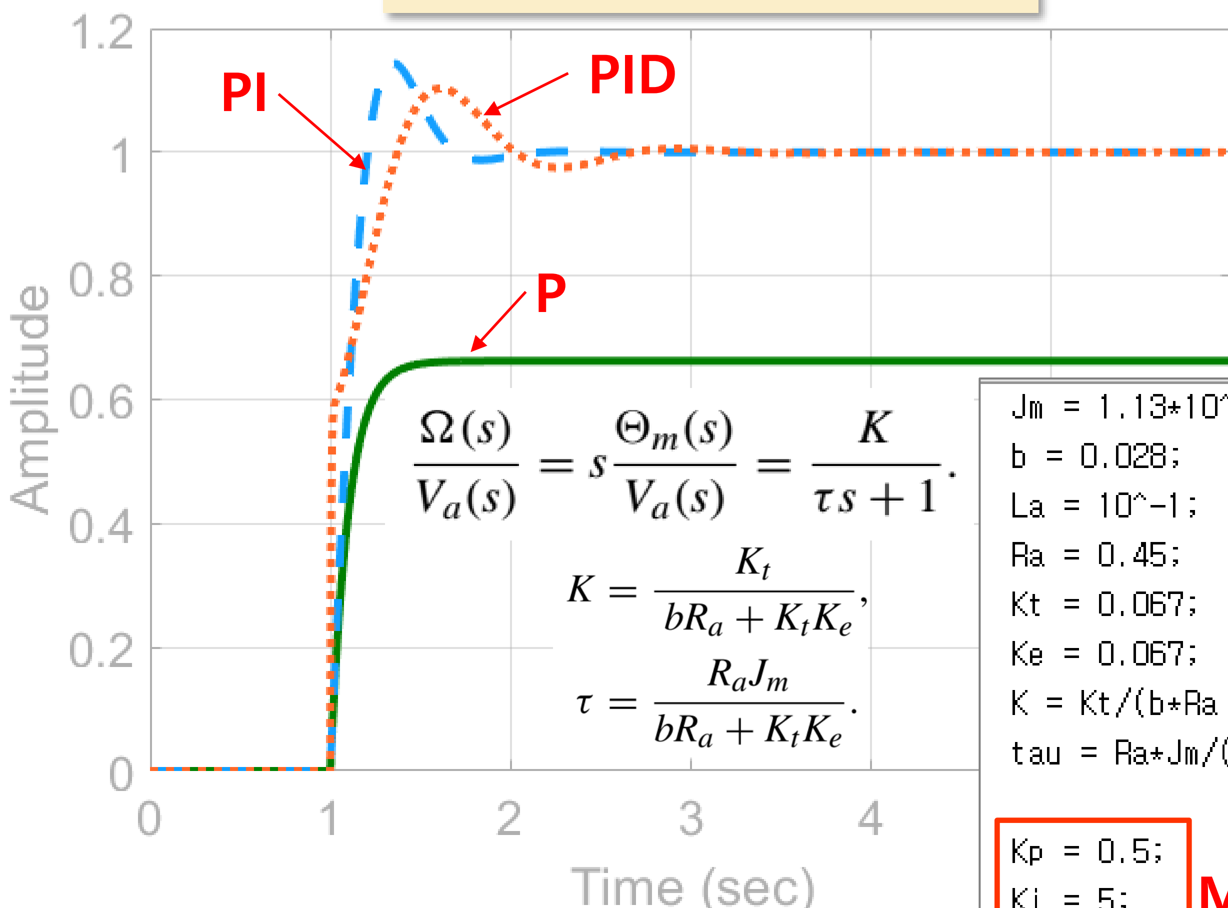
- **Integral wind-up & High-frequency noise amplification on Derivative !!**

  (1) Anti-windup schemes (e.g., temporally stopping integral action)
  (2) Low-pass-filter !! but, slow response is following.

Three-Term Controller: PID Control (Again)

❖ Example 2: DC Motor – Speed Control (cont'd)

P vs PI vs PID ??



$$\frac{\Omega(s)}{V_a(s)} = s\frac{\Theta_m(s)}{V_a(s)} = \frac{K}{\tau s + 1}.$$

$$K = \frac{K_t}{bR_a + K_t K_e},$$

$$\tau = \frac{R_a J_m}{bR_a + K_t K_e}.$$

```
Jm  = 1.13*10^-2;     % Nm.sec^2/rad
b = 0.028;            % Nm.sec/rad
La = 10^-1;           % H
Ra  = 0.45;           % ohm
Kt  = 0.067;          % Nm/A
Ke  = 0.067;          % V.sec/rad
K = Kt/(b*Ra + Kt*Ke);
tau = Ra*Jm/(b*Ra + Kt*Ke);

Kp  = 0.5;
Ki  = 5;
Kd  = 0.1;
```

Manually tuned !!

Summary

❖ **Summary:**

- Transfer function of open- & closed-loop system.
- PID control design and how it works.
- PID Control with C-code.
- Limitation and modification of PID.