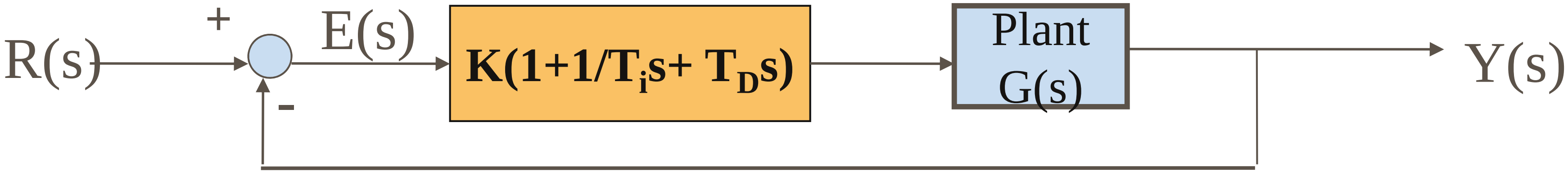
MM7 Practical Issues Using PID Controllers

Readings:

- FC textbook: Section 4.2.7 Integrator Antiwindup (p.196-200)
- Extra reading: Hou Ming's lecture notes (p.60-69)
- Extra reading: M.J. Willis notes on PID controler



What have we talked in **MM6**?

- PID controllers
- Ziegler-Nichols tuning methods

MM6:Characteristics of PID Controllers

- Proportional gain, K_p larger values typically mean faster response. An excessively large proportional gain will lead to process instability and oscillation.

- Integral gain, K_i larger values imply steady state errors are eliminated more quickly. The trade-off is larger overshoot

- Derivative gain, K_d larger values decrease overshoot, but slows down transient response and may lead to instability due to signal noise amplification in the differentiation of the error.

MM6: PID Tuning Methods- Trial-Error

Rules of thumb:

$$K_p > K_i > K_d,$$
$$K_p \approx (5 \sim 10) K_i,$$
$$K_i \approx (5 \sim 10) K_d$$

Procedure:

Step 1: Set $K_i = 0$ & $K_d = 0$. Increase K_p from zero;

Step 2: Fix K_p. Increase K_i from zero;

Step 3: Fix K_p & K_i. Increase K_d from zero.

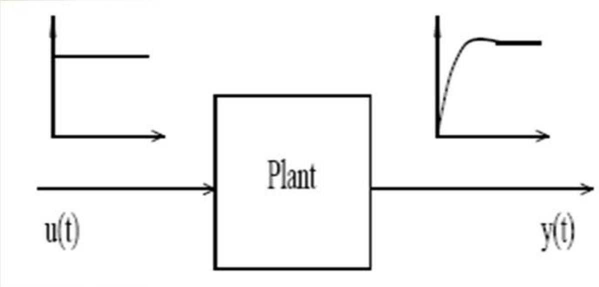
Note: Several iterations of the procedure may be necessary

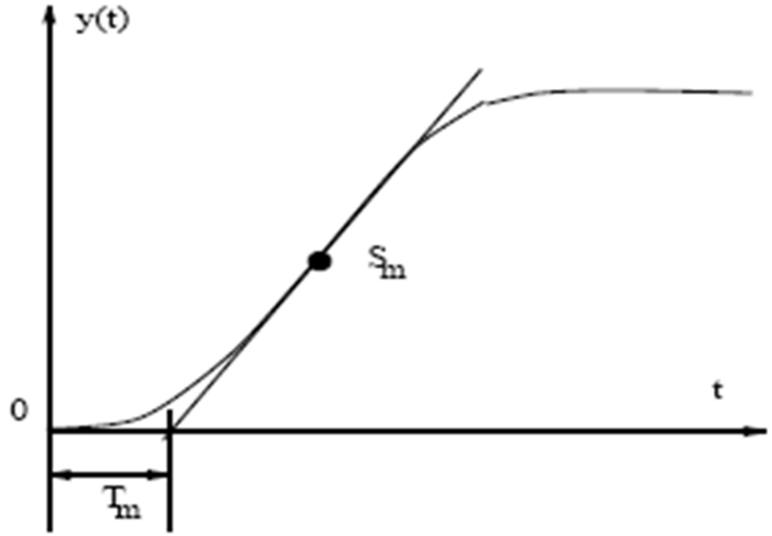
- Adavantages: Simple
- Disadvantages:
  - unsatisfactory performance
  - expensive on-site experiment
  - issues of equipment safety

See Hou Ming's lexture notes

MM6: PID Tuning – Zieglor Niechols (I)

- **Pre-condition**: system has no overshoot of step response



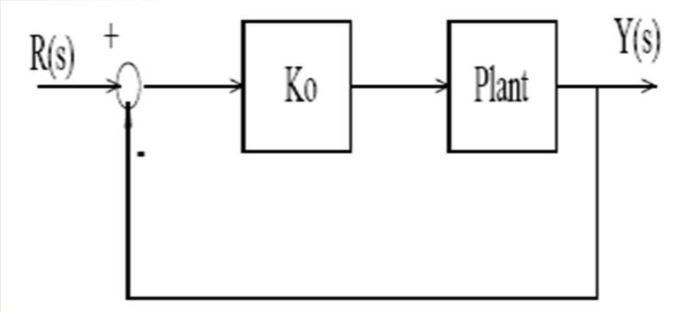


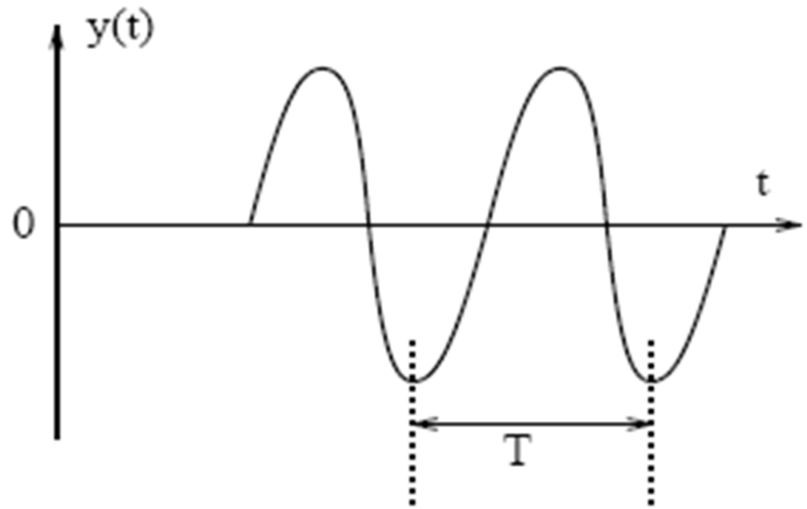
Control	K_p	K_i	K_d
P	$\frac{1}{S_m T_m}$	0	0
PI	$\frac{0.9}{S_m T_m}$	$\frac{3}{S_m}$	0
PID	$\frac{1.2}{S_m T_m}$	$\frac{2.4}{S_m}$	$\frac{0.6}{S_m}$

See Hou Ming's lexture notes

MM6: PID Tuning – Zieglor Niechols (II)

- **Pre-condition**: system order > 2





Control	K_p	K_i	K_d
P	$\frac{K_o}{2}$	0	0
PI	$\frac{9K_o}{20}$	$\frac{3K_oT_o}{8}$	0
PID	$\frac{3K_o}{5}$	$\frac{3K_oT_o}{10}$	$\frac{3K_oT_o}{40}$

See Hou Ming's lexture notes

Goals for this lecture (MM7)

Some **practical issues** when developing a PID controler:

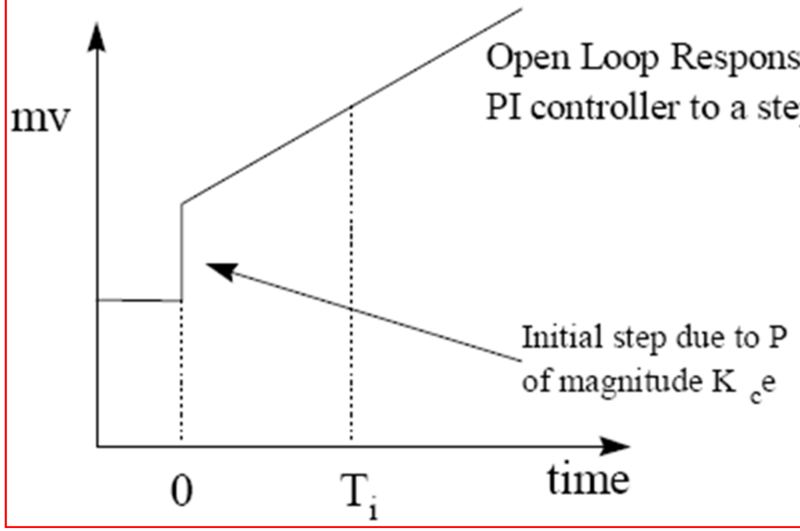
- **Integral windup & Anti-windup methods**
- Derivertive kick
- When to use which controller?
- Operational Amplifier Implementation
- Other tuning methods

PI control: Reset time

- **Control Structure** **T_I – integral/reset time**

$$\text{Time Domain}: \quad u(t) = K(e(t) + \frac{1}{T_I}\int_{t_0}^{t} e(\tau)d\tau)$$

$$\text{Frequency Domain}: D(s) = \frac{U(s)}{E(s)} = K(1 + \frac{1}{T_I s})$$



$$k_c \frac{1}{T_i}\int edt \quad = k_c e$$

Integral Windup

- **Integral windup**

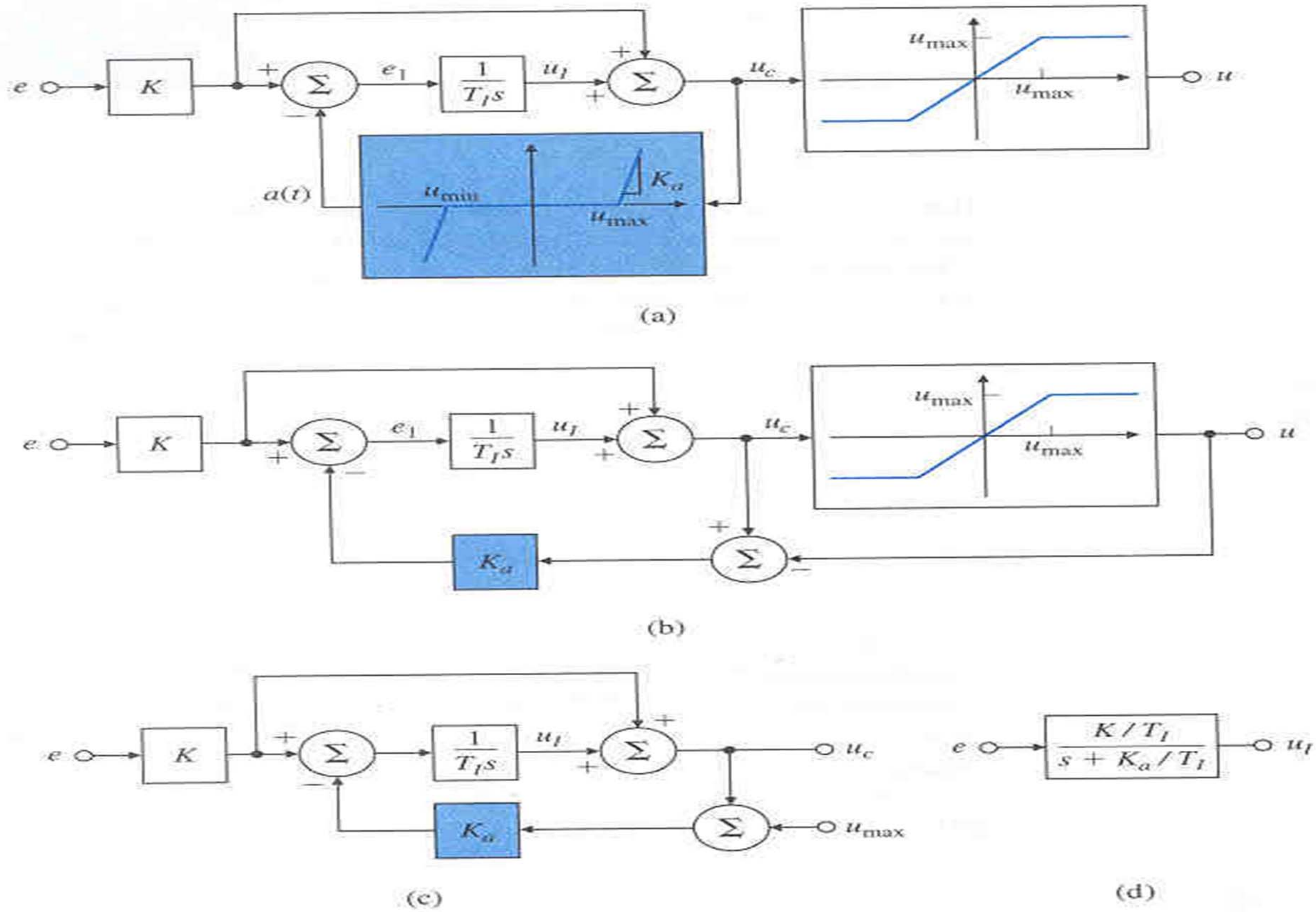
  Integration (I) → actuator saturation phenomena

- **Anti-windup**

  Turn off the integral action as soon as the actuator saturates

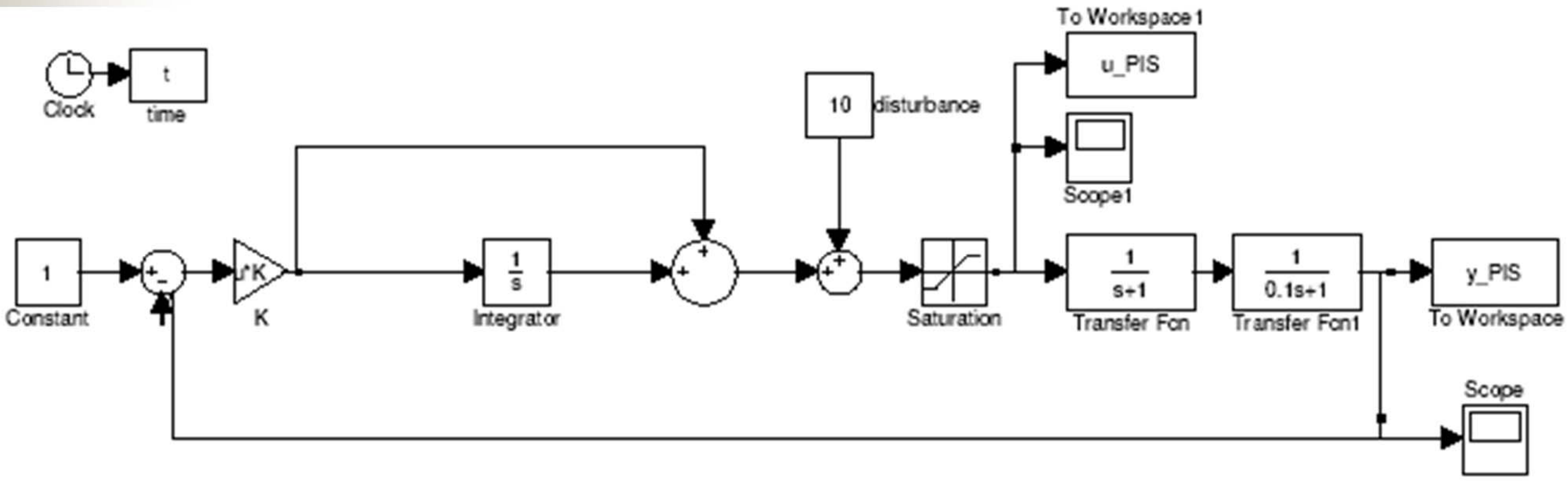
- **Anti-windup methods**
  - Implement with a dead zone
  - Implement with a nonlinearity
  - Others...

Anti-windup Techniques

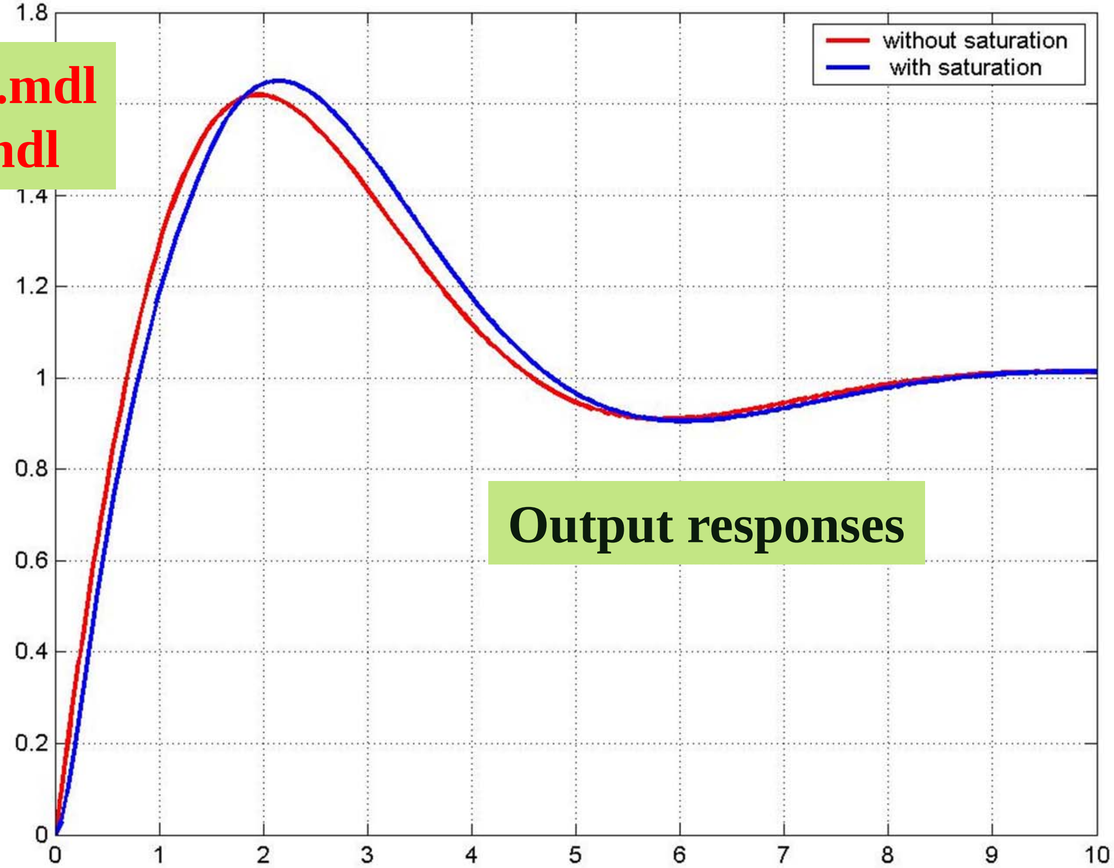


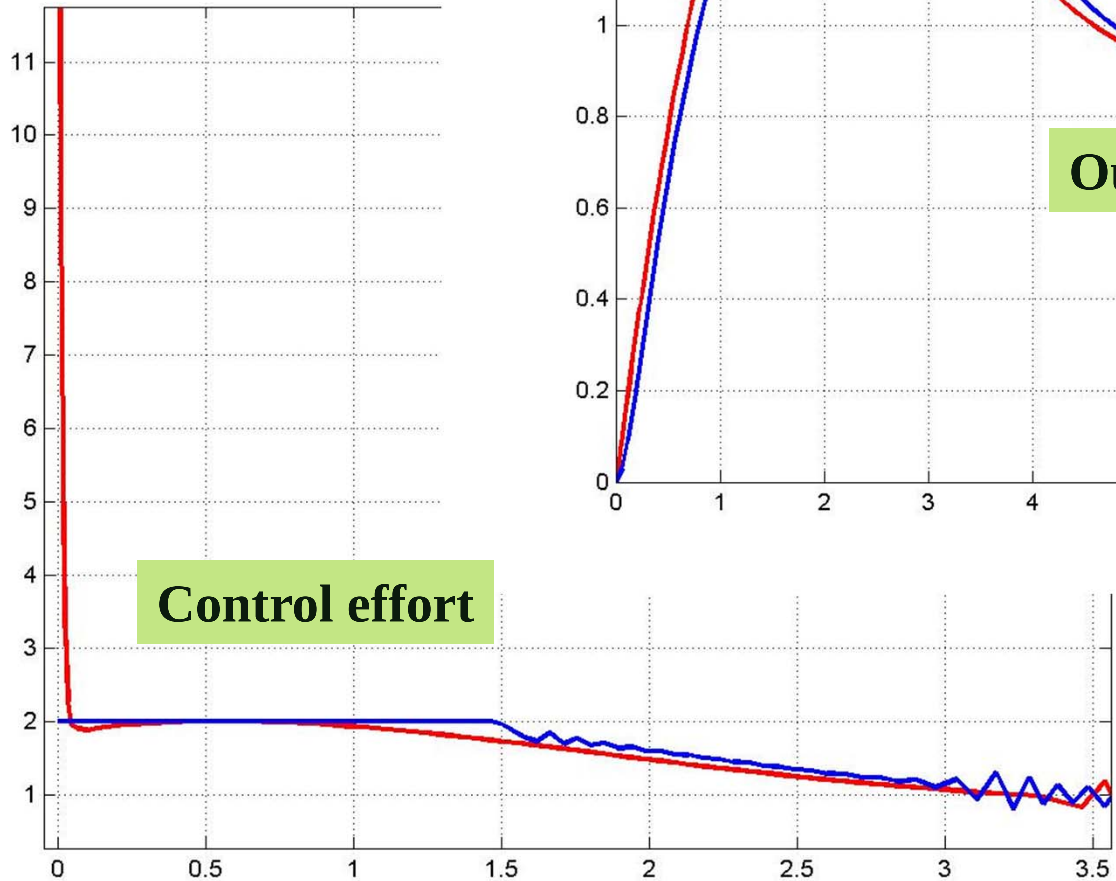
Example: DC Motor Control with Saturation

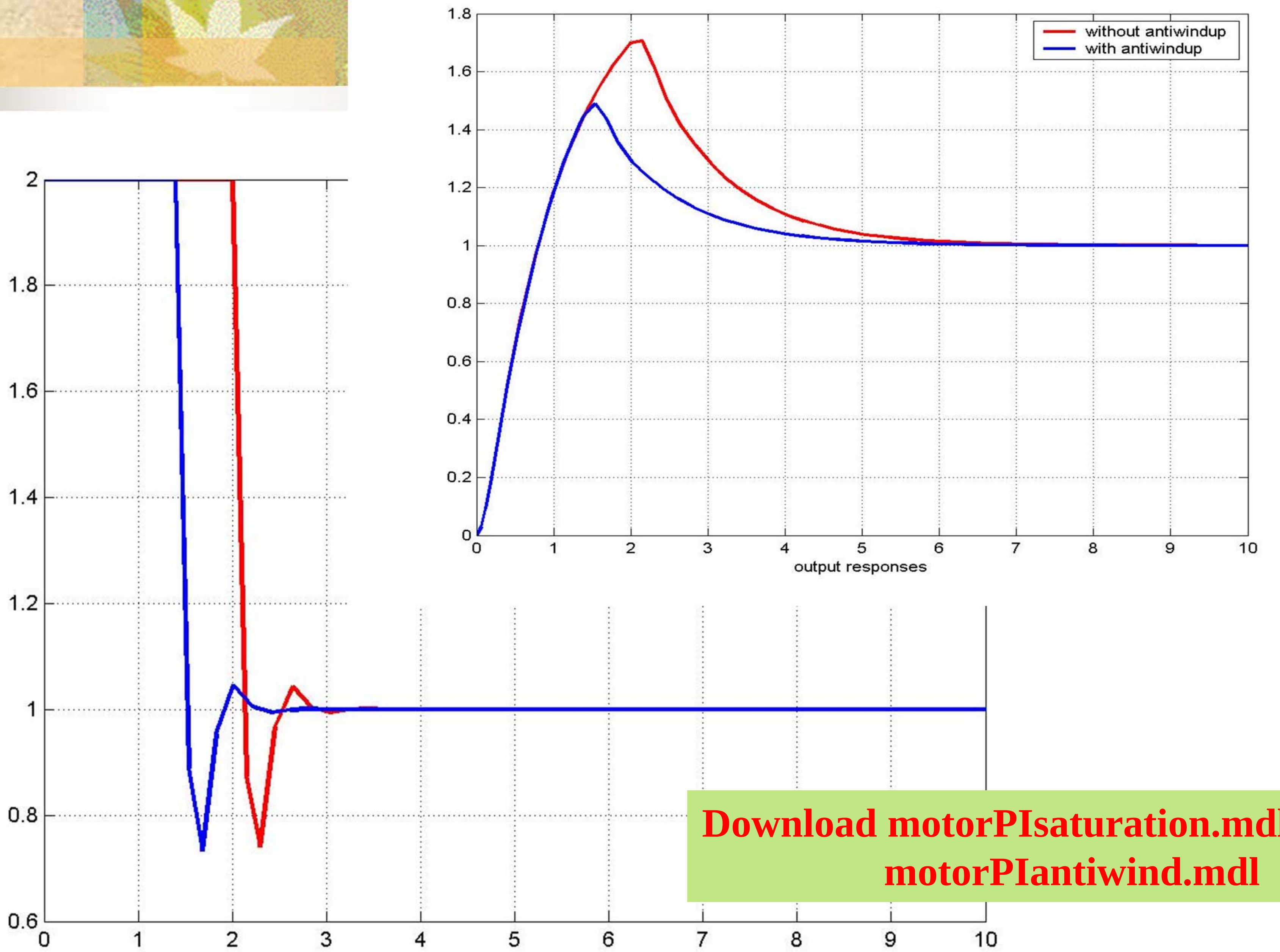
Download motorPIsaturation.mdl
motorPIantiwind.mdl



Download motorPIsaturation.mdl
motorPIantiwind.mdl







Download motorPIsaturation.mdl
motorPIantiwind.mdl

Goals for this lecture (MM7)

Some **practical issues** when developing a PID controler:

- Integral windup & Anti-windup methods
- **Derivertive kick**
- When to use which controller?
- Operational Amplifier Implementation
- Other tuning methods

Derivative Kick (I)

$$u(t) = T_D \dot{e}(t)$$

$$D(s) = \frac{U(s)}{E(s)} = T_D s$$

- **Reducing oscillations in feedback systems is the key advantage of derivative control**

However,

- Does not eliminate offset
- Slows the response

- **Derivative kick**: if we have a setpoint change, a spike will be caused by D controller, which is called derivative kick.

Derivative Kick (II)

$$u(t) = K(e(t) + \frac{1}{T_I}\int_{t_0}^{t} e(\tau)d\tau + T_D\dot{e}(t)$$

$$D(s) = \frac{U(s)}{E(s)} = K(1 + \frac{1}{T_I s} + T_D s)$$

- Derivative kick can be removed by replacing the derivative term with just output (y), instead of (r_{set}-y).

$$u(t) = K(e(t) + \frac{1}{T_I}\int_{t_0}^{t} e(\tau)d\tau + T_D\dot{y}(t))$$

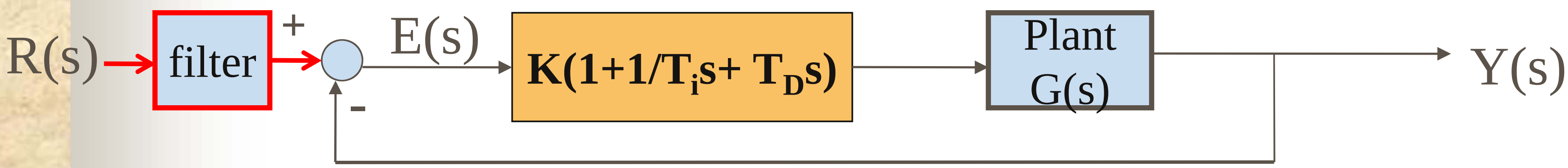
$$U(s) = K(1 + \frac{1}{T_I s})E(s) + T_D s Y(s)$$

Derivative Kick (III)

$$u(t) = K(e(t) + \frac{1}{T_I}\int_{t_0}^{t} e(\tau)d\tau + T_D\dot{e}(t)$$

$$D(s) = \frac{U(s)}{E(s)} = K(1 + \frac{1}{T_I s} + T_D s)$$

- Derivative kick can be reduced by introducing a lowpass filter before the set-point enters the system
- The bandwidth of the filter should be much larger than the closed-loop system's bandwidth



Goals for this lecture (MM7)

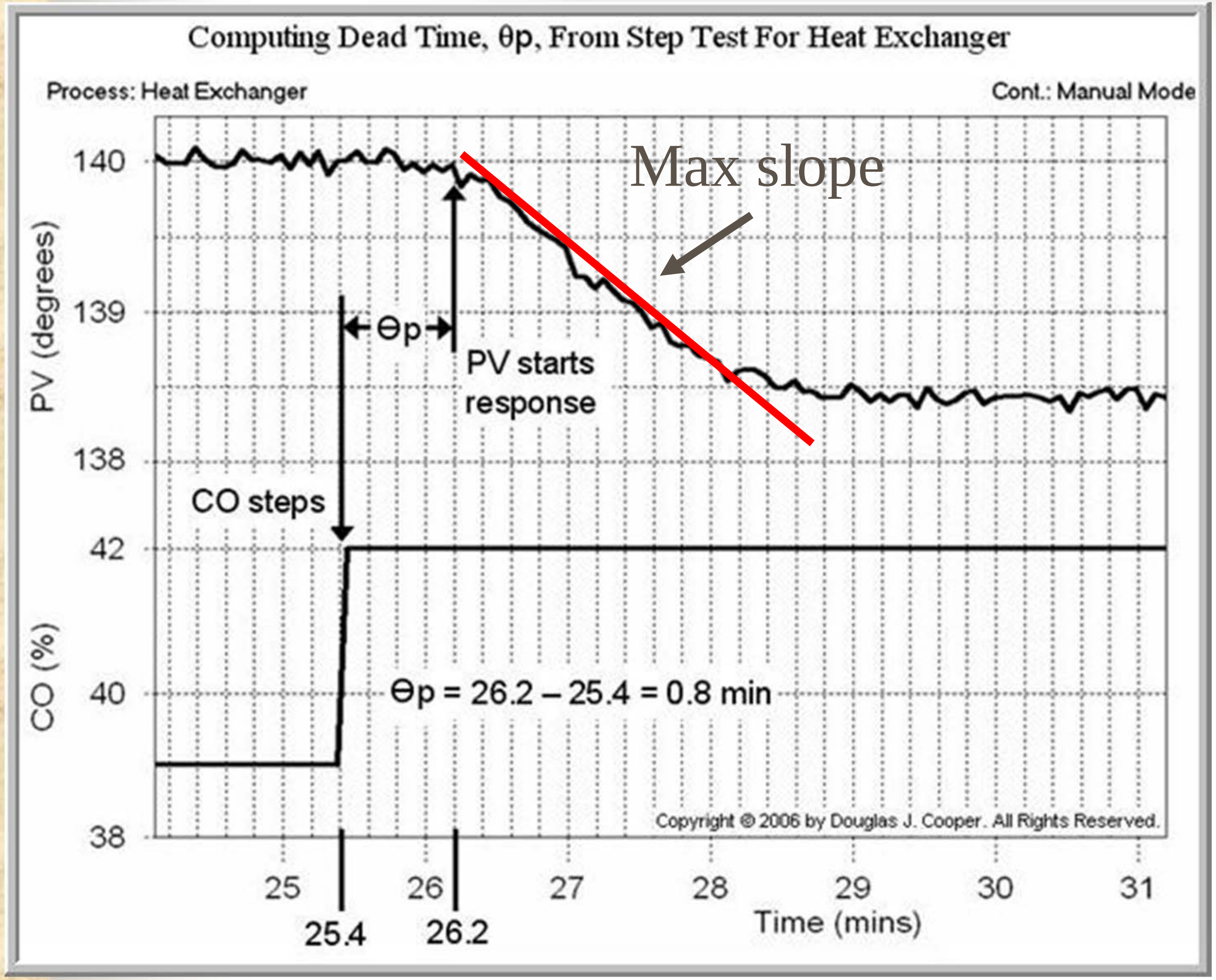
Some **practical issues** when developing a PID controler:

- Integral windup & Anti-windup methods
- Derivertive kick
- **When to use which controller?**
- Operational Amplifier Implementation
- Other tuning methods

When to use which controller?

	P	I	D	PI	PID
Estimate	present	back	forward	Present & back	All time
When to use	Systems with slow responses, tolerant to offset	Not often used alone, as is too slow	Not used alone because is too sensitive to noise and does not have setpoint	Often used	Often used, most robust, but can be noise sensitive
Examples	Example use: float valves, thermostats, humidistat.	Example use: used for very noisy systems	Example use: none	Example use: thermostats, flow control, pressure control	Examples: Cases where the system has inertia that could get out of hand: temperature and concentration measurements on a reactor for example. Avoid runaway.

¼ decay ratio is not conservative standard (too oscillatory).



- Change set point from 39 to 42% CO
- Observe delay (0.8)
- Observe max slope of response at T=27

Slope=

$$\frac{(140-139)}{(26.2-27.5)} = -0.77$$

Kmax= output change/ Input change=k1/k2

$$\frac{-0.77}{3} = -0.26$$

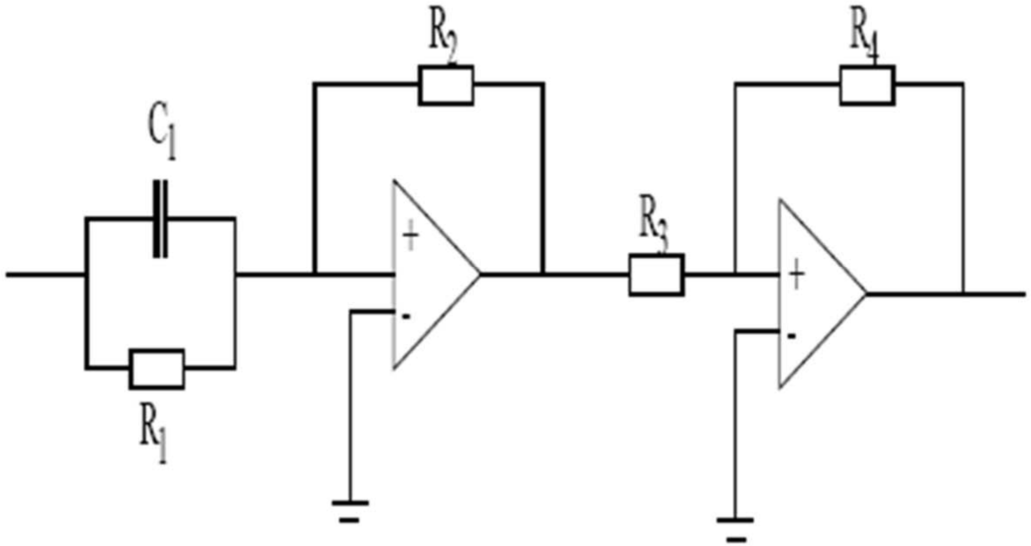
Example from http://www.controlguru.com

Goals for this lecture (MM7)

Some **practical issues** when developing a PID controler:

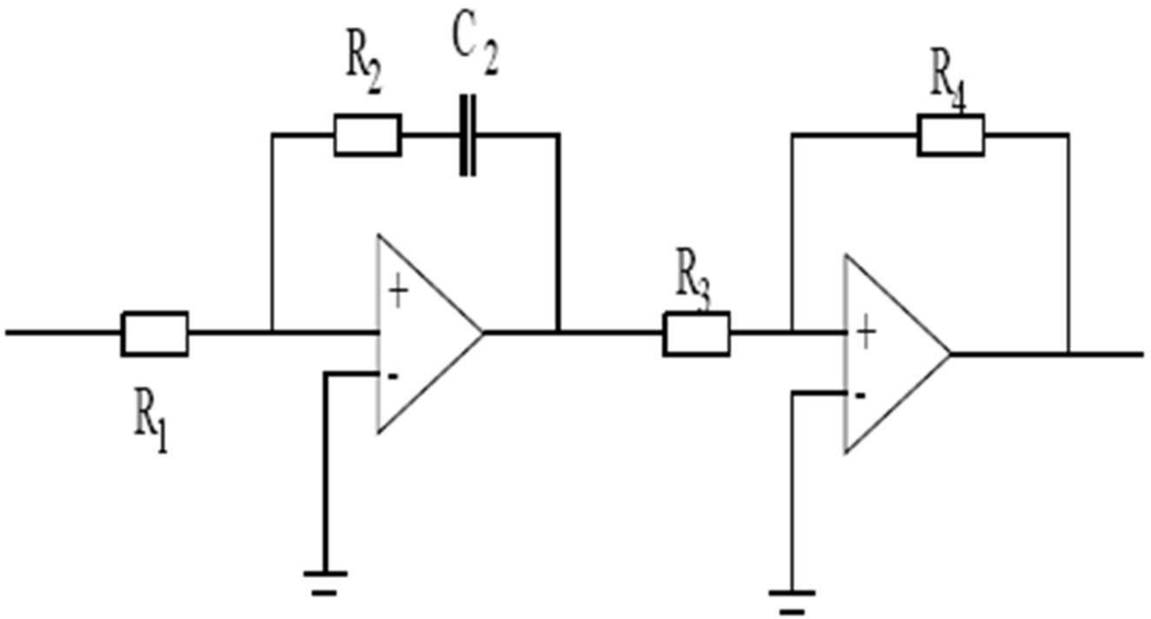
- Integral windup & Anti-windup methods
- Derivertive kick
- When to use which controller?
- **Op-Amp Implementation**
- Other tuning methods

Op-Amp Implementation (I)



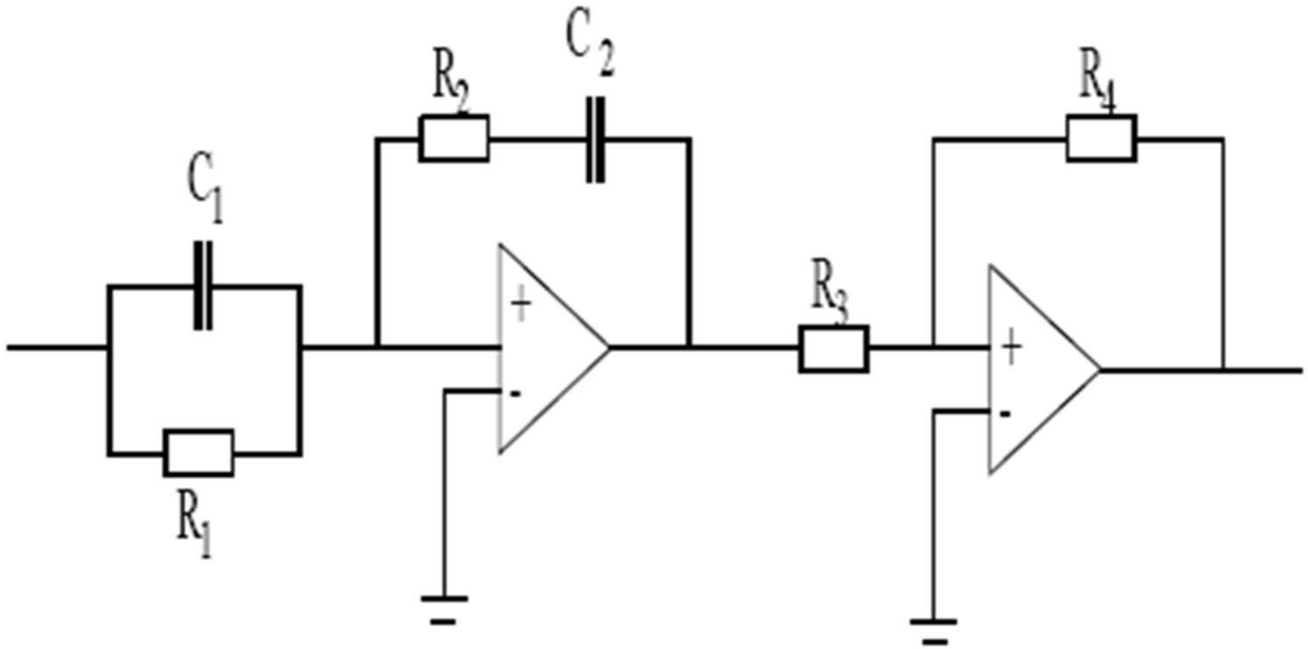
PD controller with $G_c(s) \stackrel{?}{=} \frac{R_4 R_2}{R_3 R_1}(R_1 C_1 s + 1)$

Op-Amp Implementation (II)



PI controller with $G_c(s) = \dfrac{R_4(R_2C_2s+1)}{R_3R_1C_2s}$?

Op-Amp Implementation (III)



PID controller with

$$G_c(s) = \frac{R_4(R_1C_1s+1)(R_2C_2s+1)}{R_3R_1C_2s}$$

?

Goals for this lecture (MM7)

Some **practical issues** when developing a PID controler:

- Integral windup & Anti-windup methods
- Derivertive kick
- When to use which controller?
- Op-Amp Implementation
- **Other PID tuning methods**

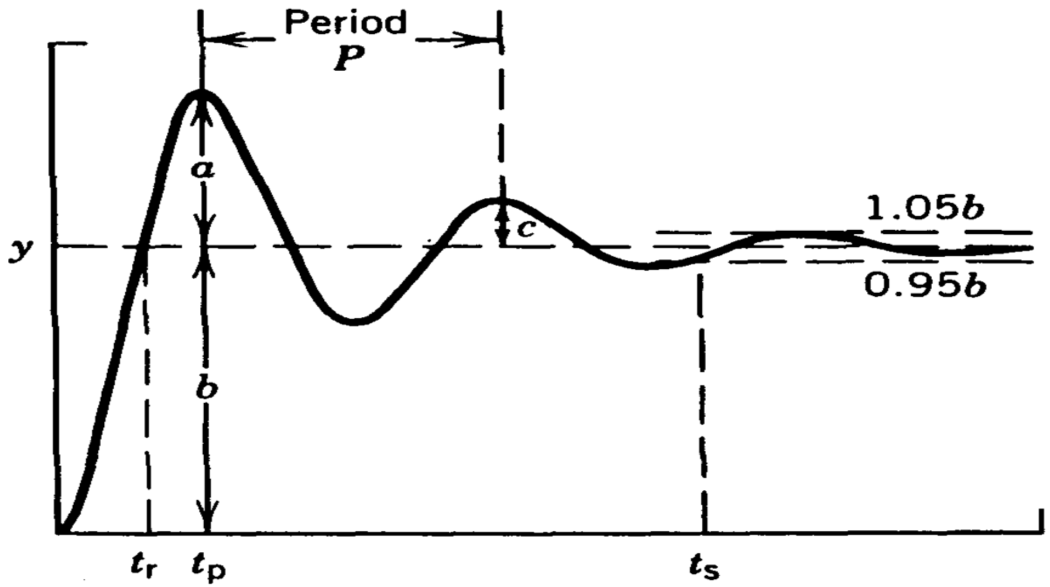
Controller Synthesis - Time Domain

Time-domain techniques can be classified into two groups:

- Criteria based on a few points in the response

  settling time, overshoot, rise time, decay ratio, settling time

- Criteria based on the entire response, or integral criteria



$$G(s) = \frac{Ke^{-\theta s}}{\tau s + 1} \qquad \text{(1st order)}$$

Cohen-Coon Tuning Method

- **Pre-condition:** first-order system with some time delay
- **Objective:** ¼ decay ratio & minimum offset

	k_c	T_i	T_D
P	$\frac{1}{k_p}\frac{\tau}{\theta}(1+\frac{\theta}{3\tau})$		
PI	$\frac{1}{k_p}\frac{\tau}{\theta}(\frac{9}{10}+\frac{\theta}{12\tau})$	$\theta\frac{30+3(\theta/\tau)}{9+20(\theta/\tau)}$	
PID	$\frac{1}{k_p}\frac{\tau}{\theta}(\frac{4}{3}+\frac{\theta}{4\tau})$	$\theta\frac{32+6(\theta/\tau)}{13+8(\theta/\tau)}$	$\theta\frac{4}{11+2(\theta/\tau)}$

In the table k_p is the process gain, τ the process time constant and θ the process time delay.

$$G(s)=\frac{Ke^{-\theta s}}{\tau s+1} \qquad \text{(1st order)}$$

Comparison of Ziegler-Nichols and Cohen-Coon Equations for Controller Tuning (1940's, 50's)

Controller	Ziegler-Nichols	Cohen-Coon
Proportional	$KK_C=\left(\tau/\theta\right)$	$KK_C=\left(\tau/\theta\right)+1/3$
Proportional + Integral	$KK_C=0.9\left(\tau/\theta\right)$ $\frac{\tau_I}{\tau}=3.33\left(\theta/\tau\right)$	$KK_C=0.9\left(\tau/\theta\right)+0.083$ $\frac{\tau_I}{\tau}=\frac{\theta\left[3.33+0.33\left(\theta/\tau\right)\right]}{1.0+2.2\left(\theta/\tau\right)}$
Proportional + Integral + Derivative	$KK_C=1.2\left(\tau/\theta\right)$ $\frac{\tau_I}{\tau}=2.0\left(\theta/\tau\right)$ $\frac{\tau_D}{\tau}=0.5\left(\theta/\tau\right)$	$KK_C=1.35\left(\tau/\theta\right)+0.270$ $\frac{\tau_I}{\tau}=\frac{\theta\left[32+6\left(\theta/\tau\right)\right]}{13+8\left(\theta/\tau\right)}$ $\frac{\tau_D}{\tau}=\frac{0.37\left(\theta/\tau\right)}{1.0+0.2\left(\theta/\tau\right)}$

These methods are not suitable for systems where there is zero(s) or virtually no time delay!

FORTD Model Approximation

- **Motivation:**

  many empirical PID tuning methods are based on first-order system with time delay
- **FORTD model approximation**
  - System identifcation method
  - Matlab: ident

Other Criteria for Performance

1. Integral of square error (ISE) $\quad \text{ISE} = \int_0^\infty [e(t)]^2 \, dt$

2. Integral of absolute value of error (IAE) $\quad \text{IAE} = \int_0^\infty |e(t)| \, dt$

3. Time-weighted IAE $\quad \text{ITAE} = \int_0^\infty t|e(t)| \, dt$

Design: Pick controller parameters to minimize integral.

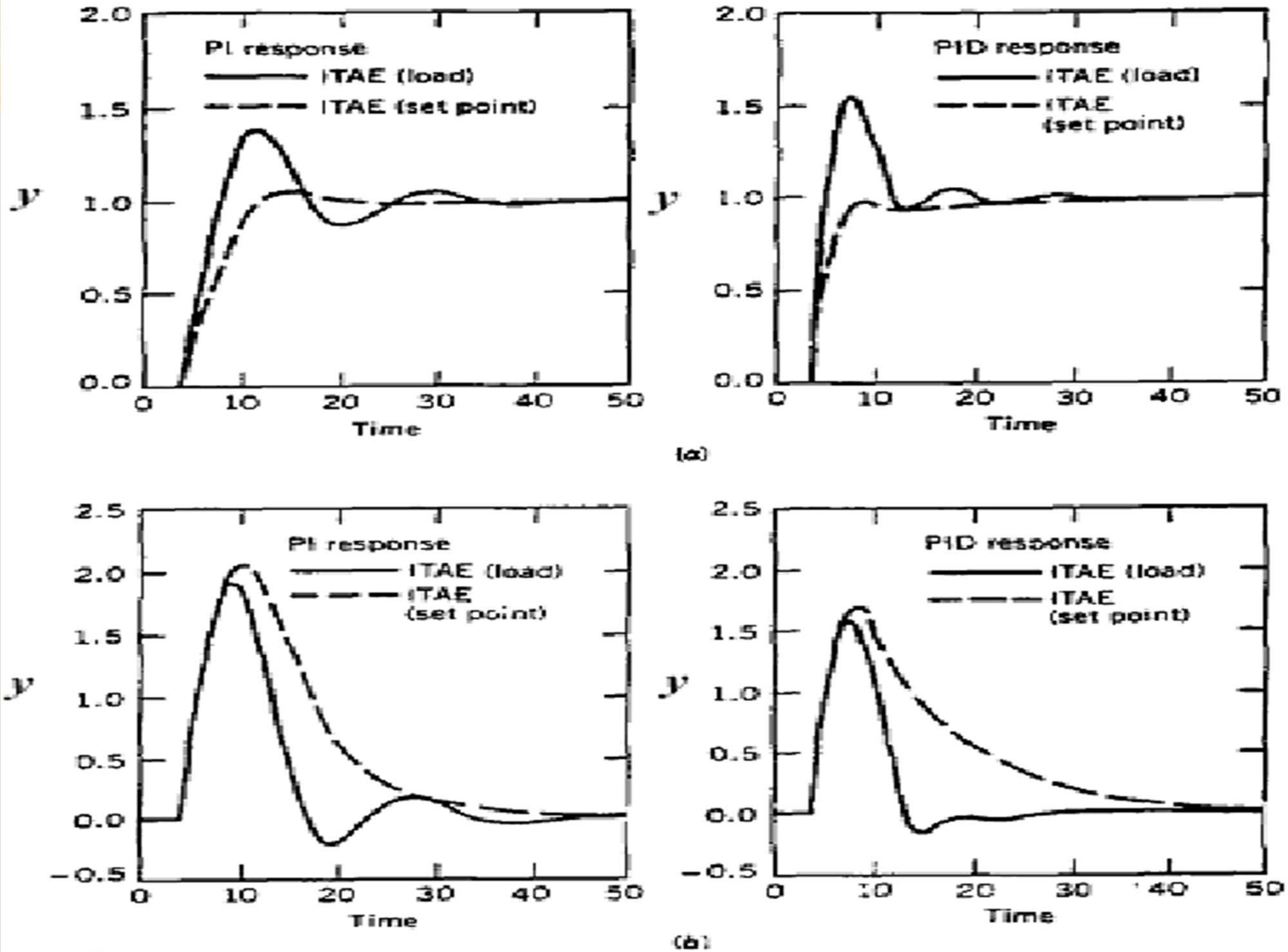
IAE allows larger deviation than ISE (smaller overshoots)
ISE longer settling time
ITAE weights errors occurring later more heavily
Approximate optimum tuning parameters are correlated with K, θ, τ...

$$G(s) = \frac{Ke^{-\theta s}}{\tau s + 1} \qquad \text{(1st order)}$$

Table 12.3 Controller Design Relations Based on the ITAE Performance Index and a First-Order plus Time-Delay Model

Type of Input	*Type of Controller*	*Mode*	*A*	*B*
Load	PI	P	0.859	−0.977
		I	0.674	−0.680
Load	PID	P	1.357	−0.947
		I	0.842	−0.738
		D	0.381	0.995
Set point	PI	P	0.586	−0.916
		I	1.03[b]	−0.165[b]
Set point	PID	P	0.965	−0.85
		I	0.796[b]	−0.1465[b]
		D	0.308	0.929

[a]Design relation: $Y = A(\theta/\tau)^B$ where $Y = KK_c$ for the proportional mode, τ/τ_I for the integral mode, and τ_D/τ for the derivative mode.
[b]For set-point changes, the design relation for the integral mode is $\tau/\tau_I = A + B(\theta/\tau)$. [8]



Comparison of controllers designed using ITAE criteria for (a) set-point and (b) load changes.

MM7 Exercise

continue MM6 execise: Design a P, PI, PID controller for the following DC motor speed control, According to quarter decay method.

Implement the above system with an actuator saturation in simulink model with $u_{max}=2$, $u_{min}=-2$. Design an integrator antiwindup strategy for your designed PI controller.

Download ZN_tuning_motor.mdl