

1

Process Dynamics and Control

3. Tuning of Single-Loop Control Systems

Kee-Youn Yoo

Department of Chemical & Biomolecular Engineering, SeoulTech

 서울과학기술대학교

2

What are we up to?

A PID controller needs only **three numbers**, K_c , τ_I and τ_D , yet choosing them is one of the most confusing parts of learning control.

Different tuning techniques give **similar but not identical** results. There is no “best” or “absolutely correct” answer, only trade-offs between speed, overshoot and robustness.

Approach	Experiment	Model	Methods
Empirical (3.1)	open-loop step test	FOPDT fit $\frac{Ke^{-\theta s}}{\tau s + 1}$	Ziegler–Nichols, Cohen–Coon, ITAE
	closed-loop test	K_{cu} , T_u only	ZN ultimate cycle
Model-based (3.2)	—	transfer function	direct synthesis, IMC

 서울과학기술대학교

3

3.1 Tuning with Empirical Relations

 서울과학기술대학교

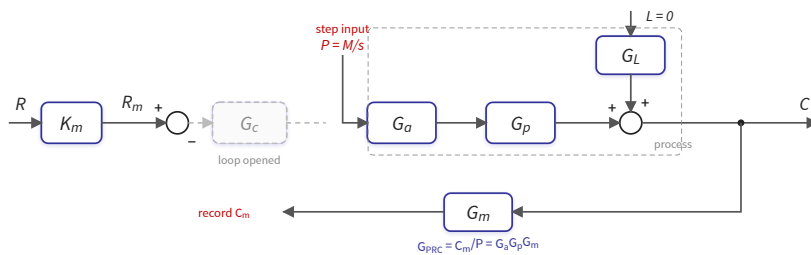
Empirical tuning in practice

1. Run an **experiment**: an open-loop step test, or a closed-loop test.
2. **Fit** the response to a simple model.
3. Compute the settings from **empirical relations**.

The settings are a **quick-and-dirty starting point**. They are seldom optimal and usually need **field tuning** after installation to meet the actual response specifications.

서울과학기술대학교

Process-reaction curve: the open-loop step test

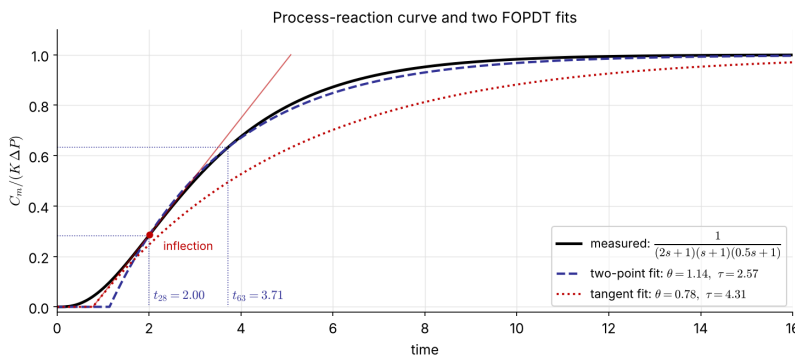


With the controller disconnected, step the controller output by $P = M/s$ and record the **measured** output. What we see is the lumped response of valve, process and sensor:

$$G_{PRC} = \frac{C_m}{P} = G_a G_p G_m \approx \frac{K e^{-\theta s}}{\tau s + 1}$$

서울과학기술대학교

Fitting the FOPDT model



Two-point method (Smith): read t_{28} and t_{63} at 28.3 % and 63.2 % of the final change

$$\tau = 1.5(t_{63} - t_{28}), \quad \theta = t_{63} - \tau, \quad K = \frac{\Delta C_m}{\Delta P}$$

Tangent method: draw the tangent at the inflection point. Its intercepts give θ and τ .

Simple, but it **overestimates** τ and **underestimates** θ , so it gives aggressive settings.

서울과학기술대학교

Ziegler–Nichols and Cohen–Coon (open-loop test)

7

Controller		Cohen–Coon	Ziegler–Nichols
P	$K_c K$	$\frac{\tau}{\theta} + \frac{1}{3}$	$\frac{\tau}{\theta}$
PI	$K_c K$	$0.9 \frac{\tau}{\theta} + \frac{1}{12}$	$0.9 \frac{\tau}{\theta}$
	τ_I	$\theta \frac{30 + 3\theta/\tau}{9 + 20\theta/\tau}$	3.3θ
PID	$K_c K$	$\frac{4}{3} \frac{\tau}{\theta} + \frac{1}{4}$	$1.2 \frac{\tau}{\theta}$
	τ_I	$\theta \frac{32 + 6\theta/\tau}{13 + 8\theta/\tau}$	2θ
	τ_D	$\theta \frac{4}{11 + 2\theta/\tau}$	0.5θ

Both aim at a **quarter-decay ratio**, which is fast but oscillatory. Note $K_c \propto 1/K$ and $K_c \propto \tau/\theta$: the larger the dead time relative to the time constant, the gentler the controller.

Minimum error-integral criteria

8

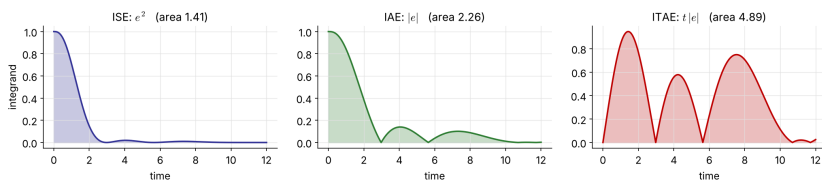
$$\text{ISE} = \int_0^\infty e^2(t) dt$$

$$\text{IAE} = \int_0^\infty |e(t)| dt$$

$$\text{ITAE} = \int_0^\infty t |e(t)| dt$$

- ISE weights large errors heavily: fast initial response, but oscillatory.
- IAE weights all errors equally.
- ITAE weights errors that **persist**: it penalizes slow oscillation, so the settings are well damped and **conservative**. Recommended.

Same response, three weightings: ITAE punishes errors that persist



Minimum-ITAE tuning relations

9

Load change

Set-point change

$$K_c = \frac{a_1}{K} \left(\frac{\tau}{\theta} \right)^{b_1}, \quad \tau_I = \frac{\tau}{a_2} \left(\frac{\theta}{\tau} \right)^{b_2}$$

$$K_c = \frac{a_1}{K} \left(\frac{\tau}{\theta} \right)^{b_1}, \quad \tau_I = \frac{\tau}{a_2 - b_2 \theta/\tau}$$

	a_1	b_1	a_2	b_2	a_3	b_3
PI	0.859	0.977	0.674	0.680	—	—
PID	1.357	0.947	0.842	0.738	0.381	0.995

	a_1	b_1	a_2	b_2	a_3	b_3
PI	0.586	0.916	1.03	0.165	—	—
PID	0.965	0.855	0.796	0.147	0.308	0.929

$$\text{both: } \tau_D = a_3 \tau \left(\frac{\theta}{\tau} \right)^{b_3}$$

Load settings are more aggressive (higher K_c , shorter τ_I) than set-point settings. Choose the table for the disturbance you expect most.

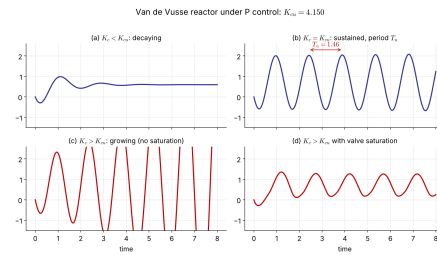
Ziegler–Nichols ultimate-cycle method (closed-loop test)

10

For processes that are **open-loop stable** but can be destabilized by feedback.

1. P-only control: set τ_I to its maximum and τ_D to its minimum if they cannot be switched off.
2. Raise K_c slowly, with small set-point or load bumps, until the oscillation is **sustained**.
3. Record the **ultimate gain** K_{cu} and the **ultimate period** T_u .

These are exactly the K_{cu} and $\omega_u = 2\pi/T_u$ of Ch 4 (Routh, direct substitution) and Ch 5 (gain margin).



With valve saturation (d) the oscillation stays bounded and looks sustained even above K_{cu} , so check that the valve is not hitting its limits.

서울과학기술대학교

Ziegler–Nichols ultimate-cycle settings

11

Controller	K_c	τ_I	τ_D
P	$0.5 K_{cu}$	—	—
PI	$0.455 K_{cu}$	$0.833 T_u$	—
PID	quarter decay	$0.6 T_u$	$0.125 T_u$
	some overshoot	$0.33 T_u$	$0.333 T_u$
	no overshoot	$0.2 T_u$	$0.333 T_u$

Advantages: no model is needed, and the test is done on the real loop. Drawbacks: the plant is pushed to the edge of instability, and it takes time.

서울과학기술대학교

Example: the Van de Vusse reactor

12

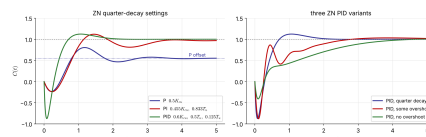
$$G_p(s) = \frac{0.5858(-0.3549s + 1)}{0.1858s^2 + 0.8627s + 1} \quad (\text{RHP zero: inverse response})$$

Raising K_c under P control (ultimate-cycle figure, two slides back):

$$K_{cu} = 4.150, \quad T_u = 1.46$$

(the original read $T_u = 1.45$ off the plot; exact: $\omega_u = 4.297$, $T_u = 2\pi/\omega_u = 1.462$)

K_c	τ_I	τ_D
P 2.07	—	—
PI 1.89	1.22	—
PID 2.49	0.73	0.18



- P: offset, steady state 0.55
- PI: no offset, slow and oscillatory
- PID: fastest, but the deep **inverse response** (-0.88) is the price of high gain on an RHP-zero process

서울과학기술대학교

Van de Vusse in code

13

```

Python
1 import numpy as np, control as ct
2 from tuning import pid_tf # ideal PID with filtered derivative (alpha = 0.1)
3
4 Gp = ct.tf(0.5858*np.array([-0.3549, 1]), [0.1858, 0.8627, 1])
5 Kcu, -, wu, - = ct.margin(Gp) # 4.158 - no trial and error needed
6 Tu = 2*np.pi/wu # 1.462
7
8 settings = {'P': (0.5*Kcu),
9             'PI': (0.455*Kcu, 0.833*Tu),
10             'PID': (0.6*Kcu, 0.5*Tu, 0.125*Tu)}
11 t = np.linspace(0, 5, 2001)
12 for name, pars in settings.items():
13     T = ct.feedback(pid_tf(pars)*Gp, 1)
14     plt.plot(t, ct.step_response(T, t).outputs, Label=name)
15
MATLAB
1 Gp = tf(0.5858*[-0.3549 1], [0.1858 0.8627 1]);
2 [Kcu, -, wu] = margin(Gp); Tu = 2*pi/wu; % 4.158, 1.462
3
4 C_P = 0.5*Kcu;
5 C_PI = pidstd(0.455*Kcu, 0.833*Tu);
6 C_PID = pidstd(0.6*Kcu, 0.5*Tu, 0.125*Tu, 10); % N = 10 <-> alpha = 0.1
7 step(feedback(C_P*Gp,1), feedback(C_PI*Gp,1), feedback(C_PID*Gp,1), 5)
8 legend('P','PI','PID')

```

 서울과학기술대학교

Example 3.6 Same plant, five tuning rules

14

“True” process $\frac{1}{(2s+1)(s+1)(0.5s+1)}$, two-point FOPDT fit
 $K = 1, \tau = 2.57, \theta = 1.14$ ($\theta/\tau = 0.44$)

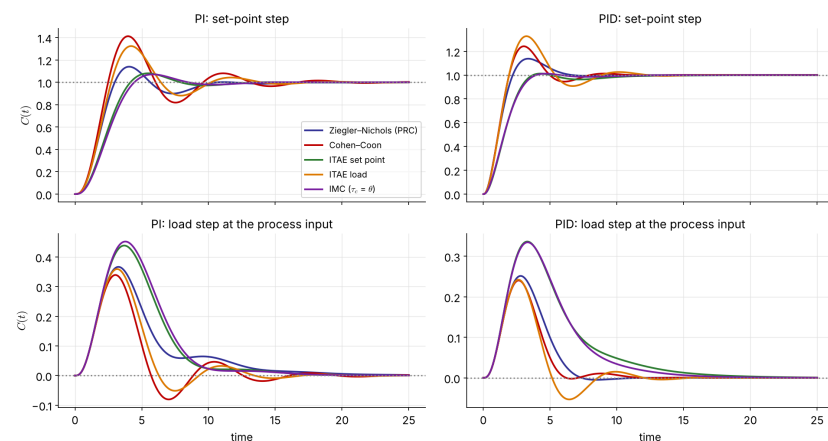
PID	K_c	τ_I	τ_D	OS	IAE sp	IAE load	GM	PM
Ziegler-Nichols	2.71	2.28	0.57	14%	1.48	0.87	21	55°
Cohen-Coon	3.26	2.39	0.38	24%	1.67	0.74	21	44°
ITAE set point	1.93	3.52	0.37	1%	1.84	1.82	37	67°
ITAE load	2.93	1.68	0.44	33%	2.07	0.81	23	38°
IMC, $\tau_c = \theta$	1.84	3.14	0.47	1%	1.74	1.71	37	68°

The aggressive rules (CC, ITAE load) reject loads best and overshoot most. The gentle ones (ITAE set point, IMC) are almost overshoot-free and have twice the phase margin. **No rule wins on every criterion.**

 서울과학기술대학교

Example 3.6: responses

15


 서울과학기술대학교

Tuning rules in code

16

Python

```

1 import numpy as np, control as ct
2 from tuning import RULES, pid_tf, fit_two_point # tuning.py in ch3/
3
4 G = ct.tf(1, np.polymul(np.polymul([2, 1], [1, 1]), [0.5, 1]))
5 t = np.linspace(0, 30, 30001)
6 K, tau, theta = fit_two_point(t, ct.step_response(G, t).outputs) # 1, 2.57, 1.14
7
8 for name, rule in RULES.items():
9     Kc, tI, tD = rule('PID', K, tau, theta)
10    T = ct.feedback(pid_tf(Kc, tI, tD)*G, 1)
11    gm, pm, _ = ct.margin(pid_tf(Kc, tI, tD)*G)
12    print(f'{name:22s} Kc={Kc:.2f} tI={tI:.2f} tD={tD:.2f} GM={gm:.1f} PM={pm:.0f}')

```

MATLAB

```

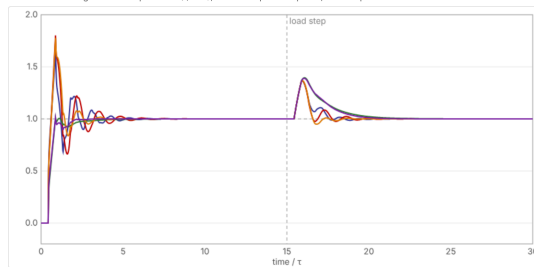
1 G = tf(1, conv(conv([2 1],[1 1]),[0.5 1]));
2 [y, t] = step(G, 30);
3 t28 = t(find(y >= 0.283, 1)); t63 = t(find(y >= 0.632, 1));
4 tau = 1.5*(t63 - t28); theta = t63 - tau; K = y(end); % 2.57, 1.14, 1
5
6 r = theta/tau; % ITAE set point, PID
7 Kc = 0.965/K*(1/r)^0.855; tI = tau/(0.796 - 0.147*r); tD = 0.388*tau*r^0.929;
8 C = pidstd(Kc, tI, tD, 10);
9 figure, step(feedback(C*G, 1)), figure, margin(C*G)

```

서울과학기술대학교

Interactive: compare the rules yourself

17

controller: $\theta/\tau = 0.44$ IMC $\tau_c/\theta = 1.00$ ▲ = still oscillating at $t = 30\tau$: plant $e^{-\theta s}/(\tau s + 1)$, $\tau = 1$: set-point step at 0, load step at $t = 15\tau$ 

	$K_c K$	τ_I/τ	τ_D/τ	IAE
☒ Ziegler-Nichols	2.73	0.88	0.22	1.26
☒ Cohen-Coon	3.28	0.92	0.15	1.44
☒ ITAE set point	1.95	1.37	0.14	1.40
☒ ITAE load	2.95	0.65	0.17	1.20
☒ IMC ($\tau_c=1.00\theta$)	1.85	1.22	0.18	1.32

Interactive version: seoultechpse.github.io/ProCon/ch3/ch3_tuning.html — vary θ/τ , PI/PID and the IMC τ_c .

서울과학기술대학교

3.2 Direct Synthesis and Internal Model Control

서울과학기술대학교

Design the closed-loop response first

19

A different philosophy: **specify the desired closed-loop response**, then solve for the controller. The result depends on the process model, so this is a **model-based design**.

Servo problem ($L = 0, G_m = G_a = 1$):

$$\frac{C}{R} = \frac{G_c G_p}{1 + G_c G_p} \Rightarrow G_c = \frac{1}{G_p} \left(\frac{C/R}{1 - C/R} \right) \quad (3.2-1)$$

$C/R = 1$ would need infinite gain. Instead ask for a first-order response with **unit gain** (no offset):

$$\frac{C}{R} = \frac{1}{\tau_c s + 1} \Rightarrow G_c = \frac{1}{G_p} \frac{1}{\tau_c s}$$

(3.2-2), (3.2-3)

τ_c is the **design parameter**: the closed-loop time constant. The closed-loop pole is $s = -1/\tau_c$ whatever G_p is. It looks like magic. In reality, saturation limits how small τ_c can be, and there is a more serious catch (next slide).

서울과학기술대학교

The catch, and two more target responses

20

$G_c = 1/(G_p \tau_c s)$ cancels the process: the poles of G_c are the zeros of G_p . An RHP zero (Van de Vusse) gives an **unstable controller**, and the cancellation is internally unstable even though C/R looks fine. Dead time would need $e^{+\theta s}$, which requires knowing the future. IMC fixes both.

Dead time in the target (cannot be removed):

$$\frac{C}{R} = \frac{e^{-\theta s}}{\tau_c s + 1} \quad (3.2-4)$$

$$G_c = \frac{1}{G_p} \frac{e^{-\theta s}}{(\tau_c s + 1) - e^{-\theta s}} \approx \frac{1}{G_p} \frac{e^{-\theta s}}{(\tau_c + \theta)s}$$

using $e^{-\theta s} \approx 1 - \theta s$ in the denominator.

Underdamped target:

$$\frac{C}{R} = \frac{1}{\tau^2 s^2 + 2\zeta \tau s + 1} \quad (3.2-6)$$

$$G_c = \frac{1}{G_p} \frac{1}{\tau^2 s^2 + 2\zeta \tau s} \quad (3.2-7)$$

서울과학기술대학교

Example 3.1 First-order process $\rightarrow$ PI

21

$$G_p = \frac{K_p}{\tau_p s + 1}, \quad \frac{C}{R} = \frac{1}{\tau_c s + 1}$$

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

PI controller: $K_c = \frac{\tau_p}{K_p \tau_c}, \quad \tau_I = \tau_p$

K_c is inversely proportional to the process gain, and a faster target (small τ_c) means a larger gain. The integral time simply equals the process time constant.

서울과학기술대학교

Example 3.2 Second-order overdamped process → PID

22

$$G_p = \frac{K_p}{(\tau_1 s + 1)(\tau_2 s + 1)} \Rightarrow G_c = \frac{(\tau_1 s + 1)(\tau_2 s + 1)}{K_p \tau_c s}$$

Ideal PID: expand the numerator and factor out $(\tau_1 + \tau_2)$:

Series (interacting) PID, $\tau_1 > \tau_2$:

$$G_c = \frac{\tau_1 + \tau_2}{K_p \tau_c} \left[1 + \frac{1}{(\tau_1 + \tau_2)s} + \frac{\tau_1 \tau_2}{\tau_1 + \tau_2} s \right]$$

$$G_c = \frac{\tau_1}{K_p \tau_c} \left(1 + \frac{1}{\tau_1 s} \right) (\tau_2 s + 1)$$

Same controller, two parameterizations: industrial controllers come in both forms, so check which one before entering the numbers.

 서울과학기술대학교
Example 3.3 FOPDT process → PI

23

(left blank in the original)

$$G_p = \frac{K_p e^{-\theta s}}{\tau_p s + 1}, \quad \frac{C}{R} = \frac{e^{-\theta s}}{\tau_c s + 1} \quad (3.2-4)$$

$$G_c \approx \frac{\tau_p s + 1}{K_p e^{-\theta s}} \frac{e^{-\theta s}}{(\tau_c + \theta)s} = \frac{\tau_p s + 1}{K_p (\tau_c + \theta)s}$$

$$\text{PI controller: } K_c = \frac{\tau_p}{K_p (\tau_c + \theta)}, \quad \tau_I = \tau_p$$

The dead time cancels exactly and acts like extra τ_c , so K_c stays finite even for $\tau_c \rightarrow 0$.

 서울과학기술대학교
Example 3.4 Overdamped process, underdamped target

24

(left blank in the original)

$$G_p = \frac{K_p}{(\tau_1 s + 1)(\tau_2 s + 1)}, \quad \frac{C}{R} = \frac{1}{\tau^2 s^2 + 2\zeta \tau s + 1} \quad (3.2-6)$$

$$G_c = \frac{(\tau_1 s + 1)(\tau_2 s + 1)}{K_p s (\tau^2 s^2 + 2\zeta \tau s)} = \frac{(\tau_1 s + 1)(\tau_2 s + 1)}{K_p 2\zeta \tau s \left(\frac{\tau}{2\zeta} s + 1 \right)}$$

PID with a derivative filter of time constant $\tau/2\zeta$:

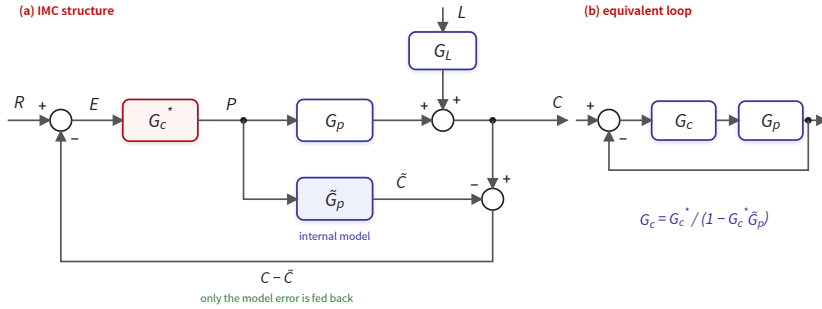
$$K_c = \frac{\tau_1 + \tau_2}{2\zeta \tau K_p}, \quad \tau_I = \tau_1 + \tau_2, \quad \tau_D = \frac{\tau_1 \tau_2}{\tau_1 + \tau_2}$$

 서울과학기술대학교

Internal model control (IMC)

25

(a) IMC structure



The controller G_c^* runs alongside a model $\tilde{G}_p$. Only the mismatch $C - \tilde{C}$ (model error plus disturbance) is fed back.

$$E = R - (C - \tilde{C}), \quad P = G_c^* E, \quad \tilde{C} = \tilde{G}_p P$$

$$P = \frac{G_c^*}{1 - G_c^* \tilde{G}_p} (R - C) \Rightarrow G_c = \frac{G_c^*}{1 - G_c^* \tilde{G}_p} \quad (3.2-8)$$

An IMC design is implemented as a conventional controller G_c .

서울과학기술대학교

IMC closed loop and the model controller

26

$$C = \left[\frac{(1 - G_c^* \tilde{G}_p) G_L}{1 + G_c^* (G_p - \tilde{G}_p)} \right] L + \left[\frac{G_p G_c^*}{1 + G_c^* (G_p - \tilde{G}_p)} \right] R$$

With a perfect model ($\tilde{G}_p = G_p$), $C/R = G_p G_c^*$: an open-loop design, which is stable whenever G_p and G_c^* are.

Split the model so the inverse never contains an RHP zero or a dead time:

$$\tilde{G}_p = \tilde{G}_{p+} \tilde{G}_{p-}, \quad G_c^* = \frac{1}{\tilde{G}_{p-}} \underbrace{\left[\frac{1}{\tau_c s + 1} \right]^r}_{\text{IMC filter}}, \quad r = 1, 2, \dots \quad (3.2-9)$$

- $\tilde{G}_{p+}$: all RHP zeros and dead time, scaled to steady-state gain 1
- $\tilde{G}_{p-}$: the invertible rest

Perfect model: $\frac{C}{R} = \frac{\tilde{G}_{p+}}{(\tau_c s + 1)^r}$. The non-invertible part is kept, not cancelled. Choose r so that G_c^* is proper.

서울과학기술대학교

Example 3.5 IMC for a FOPDT process

27

$$\tilde{G}_p = \frac{K_p e^{-\theta s}}{\tau_p s + 1} \approx \underbrace{\frac{K_p}{(\tau_p s + 1) \left(\frac{\theta}{2} s + 1 \right)}}_{\tilde{G}_{p-}} \underbrace{\left(-\frac{\theta}{2} s + 1 \right)}_{\tilde{G}_{p+}} \quad (\text{1st-order Padé})$$

$r = 1$:

$$G_c^* = \frac{(\tau_p s + 1) \left(\frac{\theta}{2} s + 1 \right)}{K_p (\tau_c s + 1)} \quad (3.2-11)$$

Into (3.2-8):

$$G_c = \frac{(\tau_p s + 1)(\theta s + 2)}{K_p (\theta + 2\tau_c) s}$$

Ideal PID

$$K_c = \frac{1}{K_p} \frac{2\tau_p/\theta + 1}{2\tau_c/\theta + 1}$$

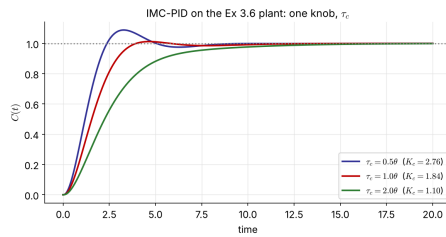
$$\tau_I = \tau_p + \frac{\theta}{2}, \quad \tau_D = \frac{\tau_p}{2\tau_p/\theta + 1}$$

Checked symbolically in the notebook. Dropping the Padé factor in $\tilde{G}_{p-}$ gives the PI of Ex 3.3.

서울과학기술대학교

Choosing τ_c

28



τ_c is the single knob that trades speed for robustness:

- smaller τ_c : faster, more overshoot, less tolerant of model error
- larger τ_c : slower, more robust

Rule of thumb: $\tau_c \approx \theta$. For the IMC-PID on an exact FOPDT the loop becomes $e^{-\theta s}/(2\theta s)$:

$$GM = \pi = 3.1, \quad PM = 61^\circ$$

서울과학기술대학교

Example 3.7 IMC for the Van de Vusse reactor

29

$$\tilde{G}_{p+} = -0.3549s + 1 \text{ (RHP zero),}$$

$$\tilde{G}_{p-} = \frac{0.5858}{0.1858s^2 + 0.8627s + 1}$$

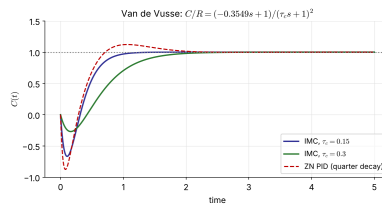
$\tilde{G}_{p-}$ has relative degree 2, so take $r = 2$:

$$\frac{C}{R} = \frac{-0.3549s + 1}{(\tau_c s + 1)^2}$$

(3.2-8) gives a PID with a derivative filter:

$$G_c = \frac{0.1858s^2 + 0.8627s + 1}{0.5858s(\tau_c^2 s^2 + 2\tau_c s + 0.3549)}$$

$$K_c = \frac{0.8627}{0.5858(2\tau_c + 0.3549)}, \quad \tau_I = 0.863, \quad \tau_D = 0.215$$



No overshoot, and τ_c sets the depth of the inverse response: -0.67 for $\tau_c = 0.15$ ($K_c = 2.25$), -0.27 for $\tau_c = 0.3$ ($K_c = 1.54$), against -0.88 for ZN PID.

서울과학기술대학교

Summary: which method when?

30

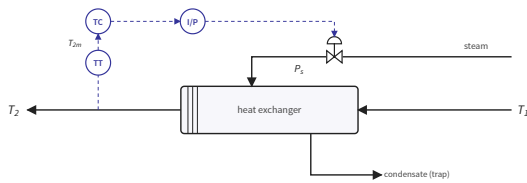
Situation	Method	Remarks
a step test is allowed	ZN / Cohen-Coon / ITAE from a FOPDT fit	fit by the two-point method
loop cannot be opened; plant tolerates cycling	ZN ultimate cycle	K_{cu} , T_u ; Ch 4-5 compute them from a model
a model is available	IMC (or direct synthesis)	one physical knob τ_c ; handles RHP zeros and dead time
robustness matters most	ITAE set point, IMC with $\tau_c \geq \theta$	check GM and PM (Ch 5)
load rejection matters most	ITAE load, Cohen-Coon	expect more overshoot on set-point changes

Whatever the rule: verify by simulation, then field-tune.

서울과학기술대학교

Problem #1 (homework)

31



t (min)	T_{2m} (mA)
0	12.0
1	12.0
2	12.5
3	13.1
4	14.0
5	14.8
6	15.4
7	16.1
8	16.4
9	16.8
10	16.9
11	17.0
12	16.9

A process stream is heated in a shell-and-tube exchanger, and T_2 is controlled with the steam valve. In an open-loop test the steam pressure P_1 was stepped from 18 to 20 psig. Valve $K_v = 0.9$ psi/psi, I/P transducer $K_{IP} = 0.75$ psi/mA. Determine PID settings by (a) IMC (choose a reasonable τ_c), (b) ITAE (set point), (c) ITAE (disturbance).

서울과학기술대학교

Resources

32

- Slides (interactive HTML): seoultechpse.github.io/ProCon/ch3/ch3_tuning.html
- PDF: [slides](#) · [handout with note space](#)
- Python notebook: [Open in Colab](#) — every calculation and figure of this chapter
- Tuning relations: [tuning.py](#) (ZN, Cohen-Coon, ITAE, IMC, FOPDT fitting)

서울과학기술대학교