

Introduction to Control (00340040)

lecture no. 12

Leonid Mirkin

Faculty of Mechanical Engineering
Technion—IIT



Outline

Industrial (PID) controllers

Tuning PID controllers

PID controller architectures and implementation

2-degrees-of-freedom controller configuration

Concluding remarks

Outline

Industrial (PID) controllers

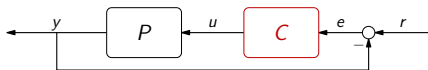
Tuning PID controllers

PID controller architectures and implementation

2-degrees-of-freedom controller configuration

Concluding remarks

PID regulators



– P: $C(s) = k_p$;

– PI:

$$C(s) = k_p \left(1 + \frac{k_i}{s} \right) = k_p \left(1 + \frac{1}{\tau_i s} \right),$$

with $\tau_i \equiv 1/k_i$ called its reset time;

– PD:

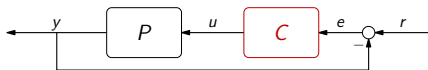
$$C(s) = k_p(1 + \tau_d s),$$

with τ_d called its derivative time;

– PID:

$$C(s) = k_p \left(1 + \frac{1}{\tau_i s} + \tau_d s \right) \quad \text{or} \quad C(s) = k_p \left(1 + \frac{\tau_d}{\tau_i s} \right) (1 + \tau_d s),$$

PID regulators



- P: $C(s) = k_p$;
- PI:

$$C(s) = k_p \left(1 + \frac{k_i}{s} \right) = k_p \left(1 + \frac{1}{\tau_i s} \right),$$

with $\tau_i = 1/k_i$ called its **reset time**;

- PD:

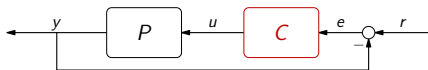
$$C(s) = k_p (1 + \tau_d s),$$

with τ_d called its derivative time;

- PID:

$$C(s) = k_p \left(1 + \frac{1}{\tau_i s} + \tau_d s \right) \quad \text{or} \quad C(s) = k_p \left(1 + \frac{1}{\tau_i s} \right) (1 + \tau_d s),$$

PID regulators



— P: $C(s) = k_p$;

— PI:

$$C(s) = k_p \left(1 + \frac{k_i}{s} \right) = k_p \left(1 + \frac{1}{\tau_i s} \right),$$

with $\tau_i = 1/k_i$ called its reset time;

— PD:

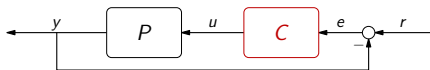
$$C(s) = k_p(1 + \tau_d s),$$

with τ_d called its **derivative time**;

— PID:

$$C(s) = k_p \left(1 + \frac{1}{\tau_i s} + \tau_d s \right) = k_p \left(\frac{\tau_i s^2 + 1 + \tau_i \tau_d s^3}{\tau_i s} \right) = k_p \left(\tau_i s^2 + \frac{1}{\tau_i s} + \tau_d s \right)$$

PID regulators



— P: $C(s) = k_p$;

— PI:

$$C(s) = k_p \left(1 + \frac{k_i}{s} \right) = k_p \left(1 + \frac{1}{\tau_i s} \right),$$

with $\tau_i = 1/k_i$ called its reset time;

— PD:

$$C(s) = k_p (1 + \tau_d s),$$

with τ_d called its derivative time;

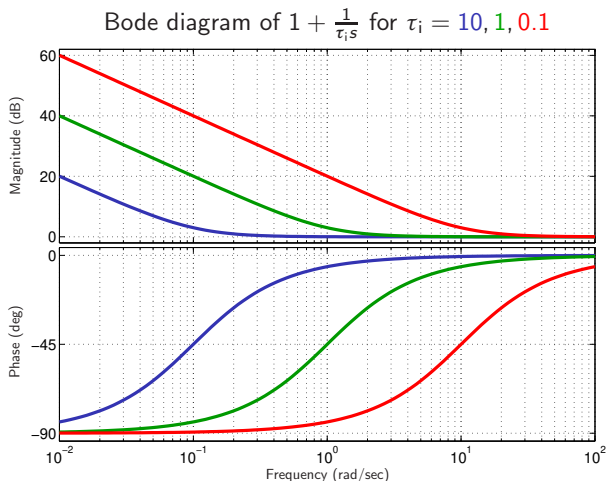
— PID:

$$C(s) = k_p \left(1 + \frac{1}{\tau_i s} + \tau_d s \right) \quad \text{or} \quad C(s) = k_p \left(1 + \frac{1}{\tau_i s} \right) (1 + \tau_d s).$$

PID regulators: why

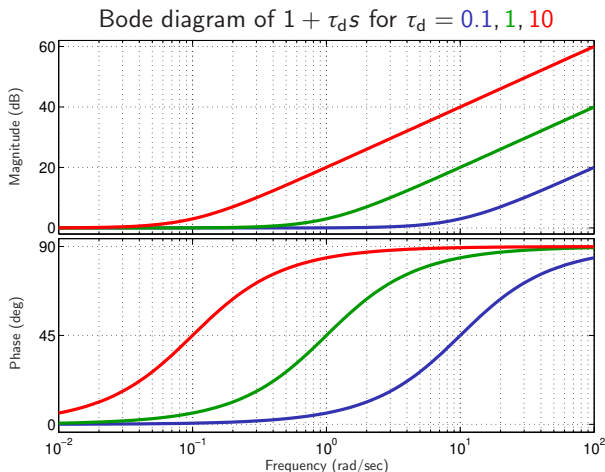
- Relatively simple structure
 - (relatively) easy to implement
 - (relatively) easy to tune
- Intuitively clear interpretation
 - the “P”-part exploits current knowledge of system behavior
 - the “I”-part exploits the past
 - the “D”-part attempts to exploit prediction of the future
- Vast practical experience

Frequency domain interpretation of PI controller



- PI controller adds additional phase lag thus reducing stability margins
- to prevent this, τ_i should be such that $\omega_c \tau_i \gg 1$

Frequency domain interpretation of PD controller



- PD controller adds a phase lead thus increasing stability margins
- to exploit this, τ_d should be such that $\tau_d \omega_c \not\ll 1$.

Outline

Industrial (PID) controllers

Tuning PID controllers

PID controller architectures and implementation

2-degrees-of-freedom controller configuration

Concluding remarks

Designing PID's

Historically, the design of PID controllers is dubbed their **tuning**. There are roughly two philosophies here:

- **explicit model-based**: starts with a process model, often of the form

$$P(s) = \frac{k}{\tau s + 1} e^{-sh} \quad \text{or} \quad P(s) = \frac{k}{(\tau_1 s + 1)(\tau_2 s + 1)} e^{-sh},$$

and then chooses k_p , τ_i , and τ_d according to whatever algorithm;

- **implicit model-based**: chooses k_p , τ_i , and τ_d directly from outcomes of a (simple) experiment with the plant according to whatever algorithm
 - although frequently regarded as model-free, all such experiments implicitly assume that plant dynamics are “simple,” say that the gain and phase are monotonically decaying functions of ω

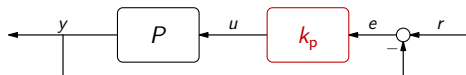
Ziegler-Nichols tuning

In 1942 John G. Ziegler & Nathaniel B. Nichols from Taylor Instrument Co. have published a paper with two methods of tuning PID controllers, based on

1. a closed-loop experiment with a proportional controller,
2. an open-loop step response experiment.

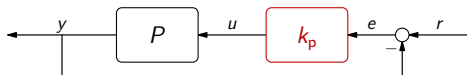
These methods were simple and efficient, soundly outperforming available solutions. They are still relevant today (although may no longer be widely used). We study the first of them below.

Ziegler-Nichols tuning: closed-loop experiment

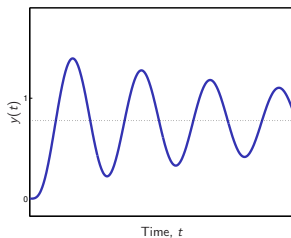


Increase k_p slowly until undamped oscillations arise in y :

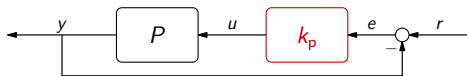
Ziegler-Nichols tuning: closed-loop experiment



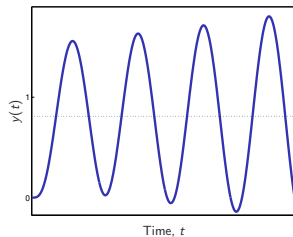
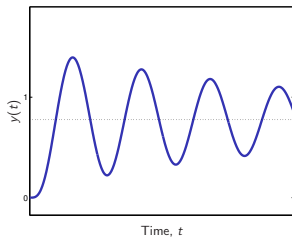
Increase k_p slowly until undamped oscillations arise in y :



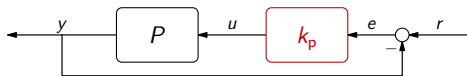
Ziegler-Nichols tuning: closed-loop experiment



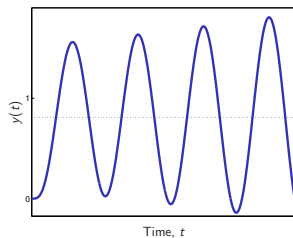
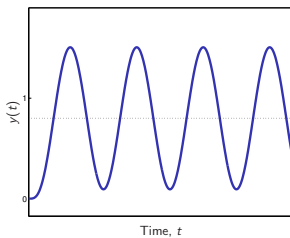
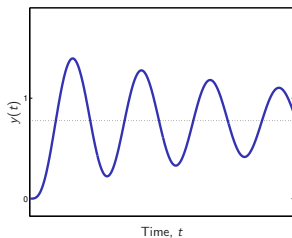
Increase k_p slowly until undamped oscillations arise in y :



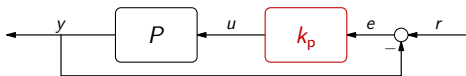
Ziegler-Nichols tuning: closed-loop experiment



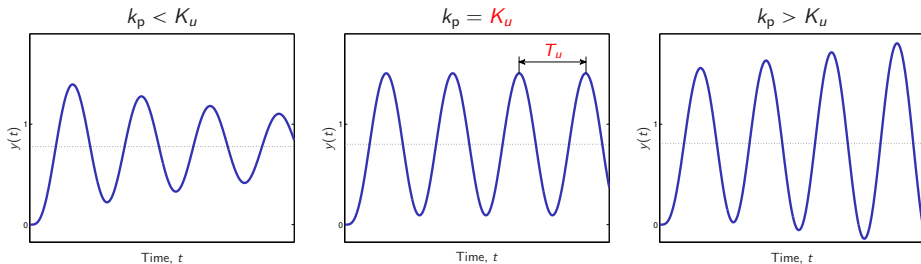
Increase k_p slowly until undamped oscillations arise in y :



Ziegler-Nichols tuning: closed-loop experiment



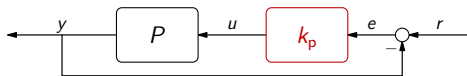
Increase k_p slowly until undamped oscillations arise in y :



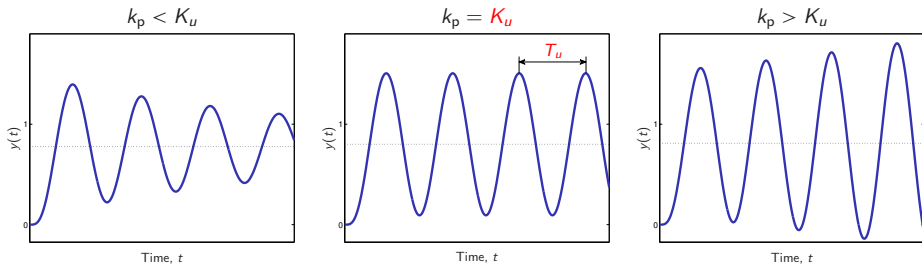
Denote the gain for which this happens as K_u and the oscillation period in steady state as T_u .

Remark: As a matter of fact, $T_u = 2\pi/\omega_u$ and $K_u = \mu_u = 1/|P(j\omega_u)|$.

Ziegler-Nichols tuning: closed-loop experiment



Increase k_p slowly until undamped oscillations arise in y :



Denote the gain for which this happens as K_u and the oscillation period in steady state as T_u .

Remark: As a matter of fact, $T_u = 2\pi/\omega_\phi$ and $K_u = \mu_g = 1/|P(j\omega_\phi)|$.

Ziegler-Nichols tuning: rules

	k_p	τ_i	τ_d	
P	$0.5 K_u$			renders $L(j\frac{2\pi}{T_u}) = 0.5 e^{-j\pi}$ (thus, $\mu_g = 2$)
PI	$0.45 K_u$	$\frac{T_u}{1.2}$		renders $L(j\frac{2\pi}{T_u}) = 0.45 e^{-j1.06\pi}$ (11° phase lag)
PID	$0.6 K_u$	$\frac{T_u}{2}$	$\frac{T_u}{8}$	renders $L(j\frac{2\pi}{T_u}) = 0.66 e^{-j0.86\pi}$ (25° phase lead) double zero at $s = -4/T_u$
PID ¹	$0.7 K_u$	$\frac{T_u}{2.5}$	$\frac{T_u}{6.7}$	renders $L(j\frac{2\pi}{T_u}) = 0.8 e^{-j0.84\pi}$ (28° phase lead) two zeros at $s = (-3.35 \pm j2.351)/T_u$ ($\zeta \approx 0.82$ and $\omega_n \approx 4.1/T_u$)

These values are to be used

– only as a starting point.

Typically, some (nontrivial) manual tuning of controller parameters required.

¹D. Pessen rule; apparently, ZN rule affected by no longer existent hardware limitations.

Ziegler-Nichols tuning: rules

	k_p	τ_i	τ_d	
P	$0.5 K_u$			renders $L(j\frac{2\pi}{T_u}) = 0.5 e^{-j\pi}$ (thus, $\mu_g = 2$)
PI	$0.45 K_u$	$\frac{T_u}{1.2}$		renders $L(j\frac{2\pi}{T_u}) = 0.45 e^{-j1.06\pi}$ (11° phase lag)
PID	$0.6 K_u$	$\frac{T_u}{2}$	$\frac{T_u}{8}$	renders $L(j\frac{2\pi}{T_u}) = 0.66 e^{-j0.86\pi}$ (25° phase lead) double zero at $s = -4/T_u$
PID ¹	$0.7 K_u$	$\frac{T_u}{2.5}$	$\frac{T_u}{6.7}$	renders $L(j\frac{2\pi}{T_u}) = 0.8 e^{-j0.84\pi}$ (28° phase lead) two zeros at $s = (-3.35 \pm j2.351)/T_u$ ($\zeta \approx 0.82$ and $\omega_n \approx 4.1/T_u$)

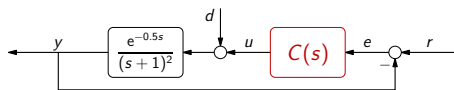
These values are to be used

- only as a starting point.

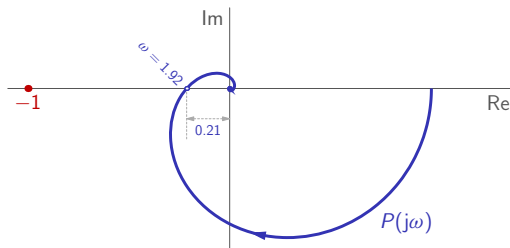
Typically, some (nontrivial) manual tuning of controller parameters required.

¹D. Pessen rule; apparently, ZN rule affected by no longer existent hardware limitations.

Ziegler-Nichols tuning: example



Polar plot of the plant is:



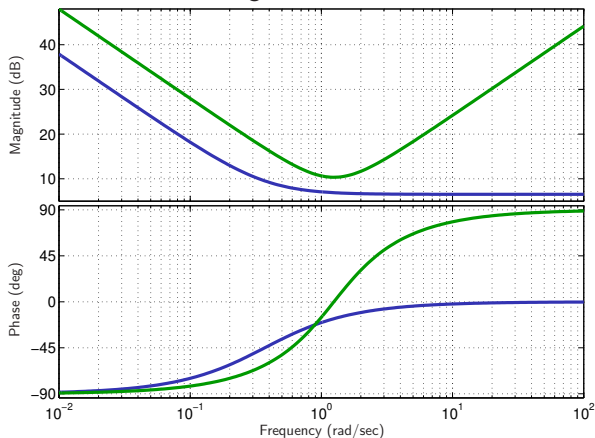
so we get: $K_u \approx 4.688$ and $T_u \approx 3.27$.

Ziegler-Nichols tuning: example (contd)

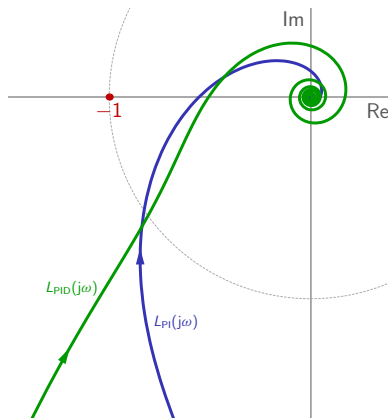
PI controller: $C_{PI}(s) = 2.11\left(1 + \frac{1}{2.73s}\right)$

PID controller: $C_{PID}(s) = 3.28\left(1 + \frac{1}{1.31s} + 0.49s\right)$

Bode diagrams of C_{PI} and C_{PID}



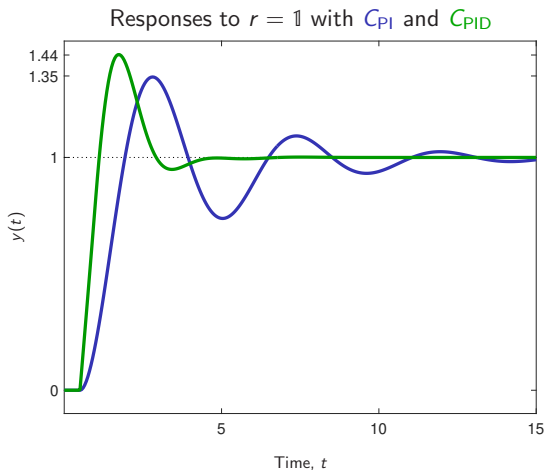
Ziegler-Nichols tuning: example (contd)



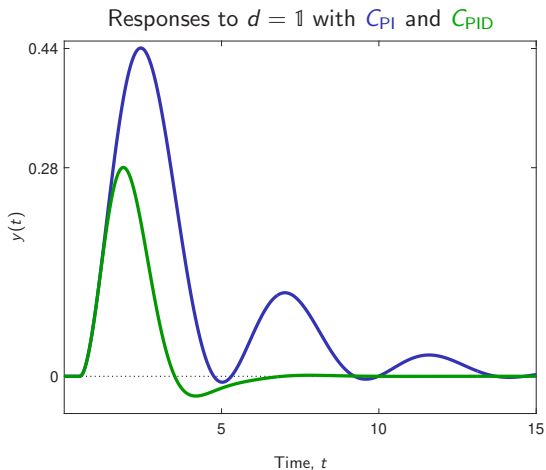
PI controller: $\omega_c = 1.11$, $\mu_g = 1.79$, $\mu_{ph} = 33.7^\circ$

PID controller: $\omega_c = 1.55$, $\mu_g = 1.97$, $\mu_{ph} = 36.3^\circ$

Ziegler-Nichols tuning: example (contd)



Ziegler-Nichols tuning: example (contd)

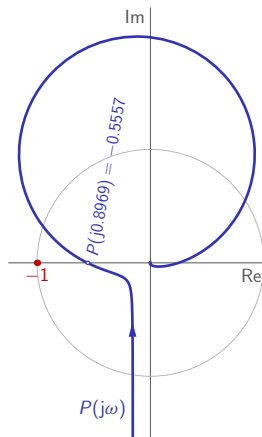


Ziegler-Nichols tuning *cum grano salis*

Let

$$P(s) = \frac{0.1e^{-1.5s}}{s(s^2 + 0.05s + 1)}$$

with



It has

$$\omega_\phi \approx 0.897 \quad \text{and} \quad |P(j\omega_\phi)| \approx 0.556,$$

so

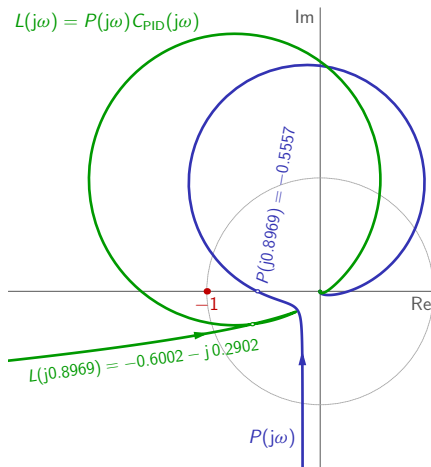
$$K_u \approx 1.8 \quad \text{and} \quad T_u \approx 7.$$

Hence,

$$\begin{aligned} C_{\text{PID}}(s) &= 1.08 \left(1 + \frac{1}{3.5s} + 0.875s \right) \\ &= \frac{0.31(1.75s + 1)^2}{s}. \end{aligned}$$

Ziegler-Nichols tuning *cum grano salis* (contd)

The loop then results in unstable closed-loop system:



Moral:

- it's still model based, just model is hidden and assumptions not clear

Where is it now?

Explicit model-based approaches seem to be dominant nowadays. Arguably, the most widely used plant model is the so-called “first-order+delay,” like

$$P(s) = \frac{k}{\tau s + 1} e^{-sh} \quad \text{or} \quad P(s) = \frac{k}{s} e^{-sh}$$

for some $\tau > 0$. In some cases, when more inertial systems are considered, “second-order+delay” models of the form

$$P(s) = \frac{k}{(\tau_1 s + 1)(\tau_2 s + 1)} e^{-sh} \quad \text{or} \quad P(s) = \frac{k}{s(\tau s + 1)} e^{-sh}$$

may be picked.

Where is it now?

Explicit model-based approaches seem to be dominant nowadays. Arguably, the most widely used plant model is the so-called “first-order+delay,” like

$$P(s) = \frac{k}{\tau s + 1} e^{-sh} \quad \text{or} \quad P(s) = \frac{k}{s} e^{-sh}$$

for some $\tau > 0$. In some cases, when more inertial systems are considered, “second-order+delay” models of the form

$$P(s) = \frac{k}{(\tau_1 s + 1)(\tau_2 s + 1)} e^{-sh} \quad \text{or} \quad P(s) = \frac{k}{s(\tau s + 1)} e^{-sh}$$

may be picked.

Where is it now? (contd)

Typical course of action:

1. identify parameters by fitting the model to system response
 - experiments may be carried out in open-loop or in closed-loop settings
 - may be based on step or frequency response characteristics
2. tune k_p , τ_i , τ_d to result in a “best” response for a given model
 - some methods are heuristic lookup tables
e.g. the so-called SIMC rule sets

$$k_p = \frac{1}{k} \frac{\tau}{h + \tau_c} \quad \text{and} \quad \tau_i = \min\{\tau, 4(h + \tau_c)\},$$

where τ_c is a tuning parameter (the closed-loop dominant time constant)

- others use advanced optimization techniques to tune PID parameters (we may also employ brute force parametric search to minimize whatever cost function . . .)

First-order+delay richness: example

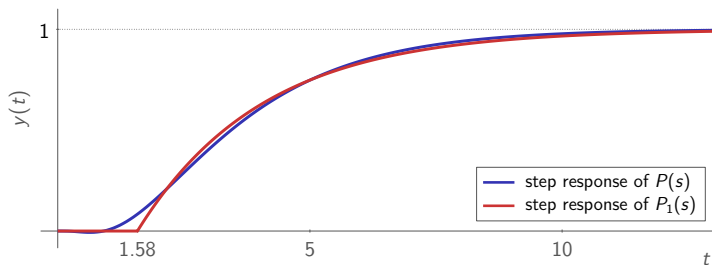
Let the “actual”

$$P(s) = \frac{(-0.3s + 1)(0.08s + 1)}{(2s + 1)(s + 1)(0.4s + 1)(0.2s + 1)(0.05s + 1)^3}.$$

It may be approximated by

$$P_1(s) = \frac{1}{2.48s + 1} e^{-1.58s}$$

reasonably well:



Second-order+delay richness: example

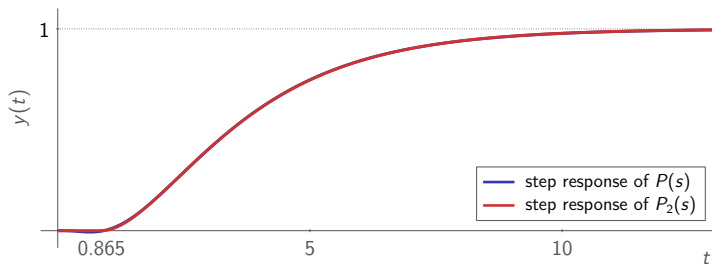
Let the “actual”

$$P(s) = \frac{(-0.3s + 1)(0.08s + 1)}{(2s + 1)(s + 1)(0.4s + 1)(0.2s + 1)(0.05s + 1)^3}.$$

Its approximation by

$$P_2(s) = \frac{1}{(1.19s + 1)(1.91s + 1)} e^{-0.865s}$$

is even better:



Outline

Industrial (PID) controllers

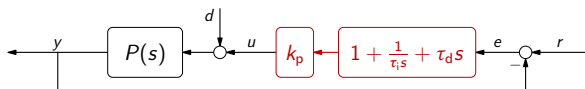
Tuning PID controllers

PID controller architectures and implementation

2-degrees-of-freedom controller configuration

Concluding remarks

“Natural” implementation and command response



with

$$u(t) = k_p \left(e(t) + \frac{1}{\tau_i} \int_0^t e(\theta) d\theta + \tau_d \dot{e}(t) \right).$$

The transfer function of the controller,

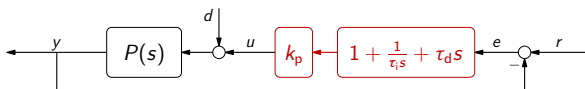
$$C(s) = k_p \left(1 + \frac{1}{\tau_i s} + \tau_d s \right) = \frac{k_p (\tau_d \tau_i s^2 + \tau_i s + 1)}{\tau_i s},$$

has zeros. If $P(s) = N_P(s)/D_P(s)$ and $C(s) = N_C(s)/D_C(s)$, then

$$T(s) = \frac{N_P(s)N_C(s)}{\chi_d(s)} = \frac{k_p (\tau_d \tau_i s^2 + \tau_i s + 1) N_P(s)}{\chi_d(s)}$$

Therefore, controller zeros are zeros of $T(s)$, unless canceled by plant poles. This might be problematic, dominant zeros tend to increase overshoot.

“Natural” implementation and command response



with

$$u(t) = k_p \left(e(t) + \frac{1}{\tau_i} \int_0^t e(\theta) d\theta + \tau_d \dot{e}(t) \right).$$

The transfer function of the controller,

$$C(s) = k_p \left(1 + \frac{1}{\tau_i s} + \tau_d s \right) = \frac{k_p (\tau_d \tau_i s^2 + \tau_i s + 1)}{\tau_i s},$$

has zeros. If $P(s) = N_P(s)/D_P(s)$ and $C(s) = N_C(s)/D_C(s)$, then

$$T(s) = \frac{N_P(s)N_C(s)}{\chi_d(s)} = \frac{k_p (\tau_d \tau_i s^2 + \tau_i s + 1) N_P(s)}{\chi_d(s)}$$

Therefore, controller zeros are zeros of $T(s)$, unless canceled by plant poles. This might be problematic, dominant zeros tend to increase overshoot.

“Natural” implementation and command response



with

$$u(t) = k_p \left(e(t) + \frac{1}{\tau_i} \int_0^t e(\theta) d\theta + \tau_d \dot{e}(t) \right).$$

The transfer function of the controller,

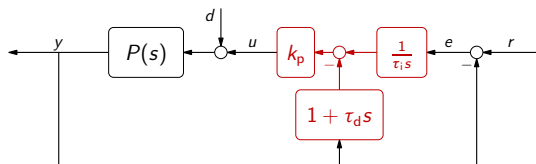
$$C(s) = k_p \left(1 + \frac{1}{\tau_i s} + \tau_d s \right) = \frac{k_p (\tau_d \tau_i s^2 + \tau_i s + 1)}{\tau_i s},$$

has zeros. If $P(s) = N_P(s)/D_P(s)$ and $C(s) = N_C(s)/D_C(s)$, then

$$T(s) = \frac{N_P(s)N_C(s)}{\chi_{cl}(s)} = \frac{k_p (\tau_d \tau_i s^2 + \tau_i s + 1) N_P(s)}{\chi_{cl}(s)}$$

Therefore, controller zeros are zeros of $T(s)$, unless canceled by plant poles. This might be problematic, dominant zeros tend to increase overshoot.

Alternative architecture



with

$$u(t) = k_p \left(-y(t) + \frac{1}{\tau_i} \int_0^t e(\theta) d\theta - \tau_d \dot{y}(t) \right).$$

Now the transfer function of the system $r \mapsto y$,

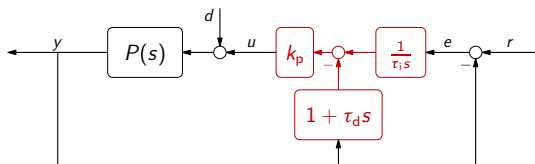
$$T_{rr}(s) = \frac{P(s) k_p / (\tau_i s)}{1 + P(s) C(s)} = \frac{k_p N_P(s)}{\chi_d(s)},$$

has only zeros of the plant as its zeros, which may yield smoother transients.

At the same time,

— disturbance response is not affected by this change of the architecture.

Alternative architecture



with

$$u(t) = k_p \left(-y(t) + \frac{1}{\tau_i} \int_0^t e(\theta) d\theta - \tau_d \dot{y}(t) \right).$$

Now the transfer function of the system $r \mapsto y$,

$$T_{yr}(s) = \frac{P(s) k_p / (\tau_i s)}{1 + P(s) C(s)} = \frac{k_p N_P(s)}{\chi_{cl}(s)},$$

has only zeros of the plant as its zeros, which may yield smoother transients. At the same time,

- disturbance response is not affected by this change of the architecture.

Alternative architecture: example

For example, let

$$P(s) = \frac{k}{s+a} \quad \text{and} \quad C(s) = k_p \left(1 + \frac{1}{\tau_i s}\right)$$

Then

$$T(s) = \frac{kk_p(\tau_i s + 1)}{\tau_i s^2 + \tau_i(a + kk_p)s + kk_p}$$

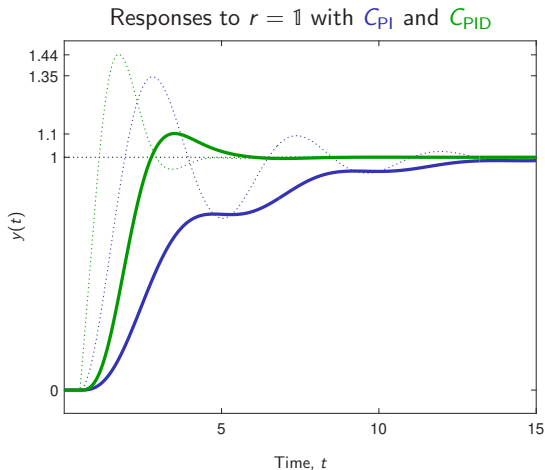
has a zero at $s = -1/\tau_i$ (unless $\tau_i = 1/a$), whereas

$$T_{yr}(s) = \frac{kk_p}{\tau_i s^2 + \tau_i(a + kk_p)s + kk_p}$$

is a 2-order transfer function without zeros (hence, easier to understand).

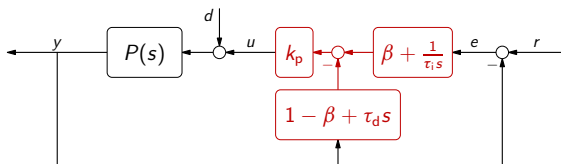
Ziegler-Nichols tuning: example (contd)

The same tuning, but different implementation architecture:



The disturbance response remains the same.

PID with set-point weighting



with

$$u(t) = k_p \left(\beta r(t) - y(t) + \frac{1}{\tau_i} \int_0^t e(\theta) d\theta - \tau_d \dot{y}(t) \right)$$

for some tuning parameter $\beta \in [0, 1]$. In this case

$$T_{yr}(s) = \frac{k_p(\beta \tau_i s + 1)N_P(s)}{\chi_{cl}(s)},$$

so we can affect controller-induced zero, at $s = -1/(\beta \tau_i)$. In this case

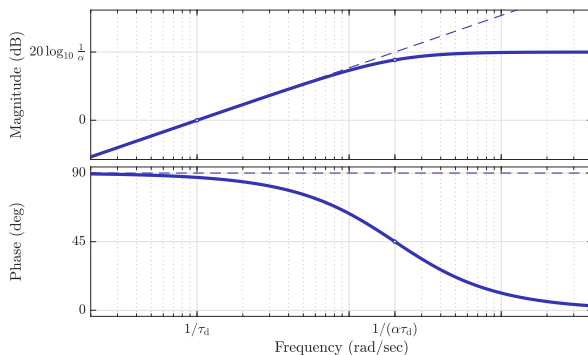
- increasing β leads to faster responses with larger overshoot,
- decreasing β leads to slower responses with smaller overshoot.

Implementing the D part

The non-proper $\tau_d s$ is normally implemented as

$$\frac{\tau_d s}{\alpha \tau_d s + 1}$$

with sufficiently small α (typically, $0.05 \leq \alpha \leq 0.3$):



Outline

Industrial (PID) controllers

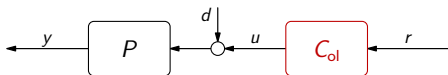
Tuning PID controllers

PID controller architectures and implementation

2-degrees-of-freedom controller configuration

Concluding remarks

Open-loop control: architecture



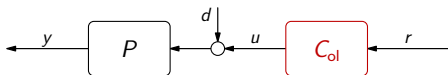
Signals of interest:

$$y = PC_{ol}r + Pd \quad \& \quad u = C_{ol}r,$$

where

- r is a reference signal (requirements)
- d is a load disturbance

Open-loop control: strategy



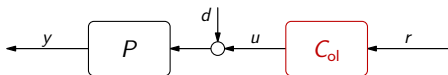
Plant inversion with reference model:

$$C = C_{ol} = P^{-1} T_{ref},$$

where **stable** T_{ref} should endeavor to have ...

- steady-state** $|1 - T_{ref}(j\omega)| \ll 1$ at ω 's where the spectrum of r dominates
- transients**
- dominant poles of $T_{ref}(s)$ are in “good” regions
 - sufficiently wide, but not too wide, bandwidth of $T_{ref}(j\omega)$
 - no high resonant peaks of $|T_{ref}(j\omega)|$

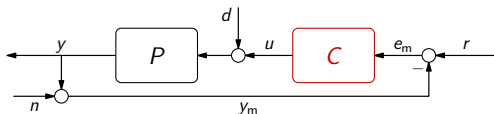
Open-loop control: properties



Open-loop architecture is

- ☺ efficient in handling command following requirements
- ☺ technically simple
 - both T_{ref} and C_{ol} are stable
 - all nonminimum-phase zeros of $P(s)$ as zeros of $T_{\text{ref}}(s)$
 - pole excess of $T_{\text{ref}}(s) \geq$ poles excess of $P(s)$ unless $\dot{r}$, $\ddot{r}$, etc measurable
- ☹ does not help in handling uncertainty
 - modeling inaccuracies in P
 - disturbances
- ☹ inapplicable if the plant P is unstable

Closed-loop control: architecture (unity feedback)



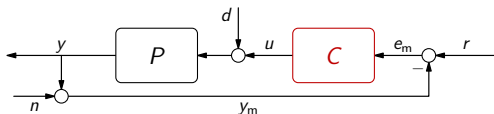
Signals of interest:

$$y = Tr + T_d d - T_n \quad \& \quad u = T_c r - T_d - T_c n,$$

where

- $T(s) = \frac{P(s)C(s)}{1 + P(s)C(s)}$ complementary sensitivity
- $T_d(s) = \frac{P(s)}{1 + P(s)C(s)}$ disturbance sensitivity
- $T_c(s) = \frac{C(s)}{1 + P(s)C(s)}$ control sensitivity
- $S(s) = 1 - T(s)$ sensitivity

Closed-loop control: strategy



Internally **stabilizing** C should endeavor to have ...

Command following:

- $|1 - T(j\omega)| \ll 1$ at ω 's where the spectrum of r dominates
- $T(j\omega)$ has sufficiently wide, but not too wide, bandwidth
- $T(j\omega)$ has no high resonance peaks

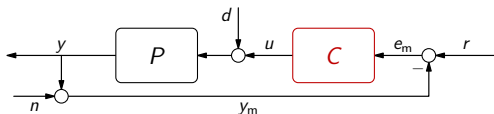
Disturbance attenuation:

- $|T_d(j\omega)| \ll 1$ at ω 's where the spectrum of d dominates

Noise sensitivity:

- $|T(j\omega)| \ll 1$ & $|T_c(j\omega)| \not\ll 1$ at ω 's where the spectrum of n dominates

Closed-loop control: properties



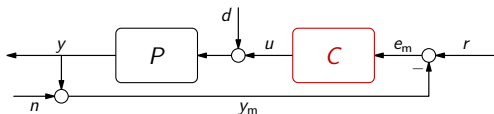
Closed-loop architecture is

- 😊 efficient in stabilizing
- 😊 efficient in handling command following requirements
- 😊 efficient in attenuating disturbances
- 😞 technically nontrivial
 - high-low gain tradeoffs
 - crossover region acrobatics

partially because it has to address

→ too many intrinsically conflicting goals.

Closed-loop control: properties



Closed-loop architecture is

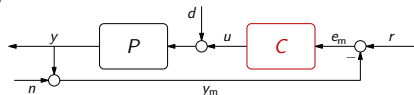
- ☺ efficient in stabilizing
- ☺ efficient in handling command following requirements
- ☺ efficient in attenuating disturbances
- ☹ technically nontrivial
 - high-low gain tradeoffs
 - crossover region acrobatics

partially because it has to address

- ☹ too many intrinsically conflicting goals.

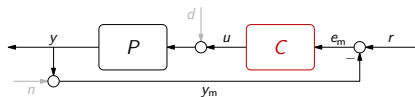
The best of two worlds

Handling uncertainty:

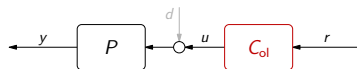


- no alternative to feedback
- no alternative architecture (C acts after n and before d)

Nominal command following:



VS.



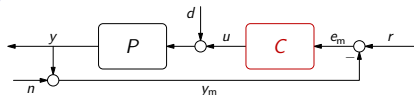
- no advantage of feedback
- open-loop design is simpler

Natural question:

- can we synergize open- and closed-loop architectures?

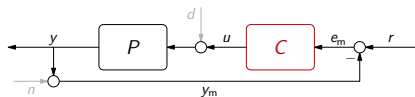
The best of two worlds

Handling uncertainty:

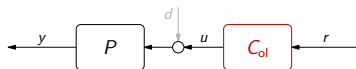


- no alternative to feedback
- no alternative architecture (C acts after n and before d)

Nominal command following:



vs.



- no advantage of feedback
- open-loop design is simpler

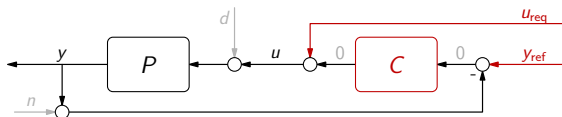
Natural question:

- can we synergize open- and closed-loop architectures?

Circumvent C in the nominal case

The idea is to

- negate the effect of C if everything behaves expectably.



Expectable behavior is expressed as two requirements to added signals:

- y_{ref} and u_{req} are bounded stability
- $y_{\text{ref}} = Pu_{\text{req}}$ consistency

If $d = n = 0$, then $T_C = I$ and $S + T_C = I$

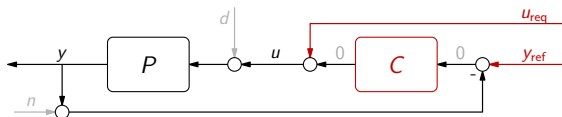
$$u = u_{\text{req}} + C(y_{\text{ref}} - Pu) \iff u = Su_{\text{req}} + T_C y_{\text{ref}} = u_{\text{req}}$$

and $y = Pu = y_{\text{ref}}$, regardless C (provided it is stabilizing, of course).

Circumvent C in the nominal case

The idea is to

- negate the effect of C if everything behaves expectably.



Expectable behavior is expressed as two requirements to added signals:

- y_{ref} and u_{req} are bounded stability
- $y_{\text{ref}} = Pu_{\text{req}}$ consistency

If $d = n = 0$, then

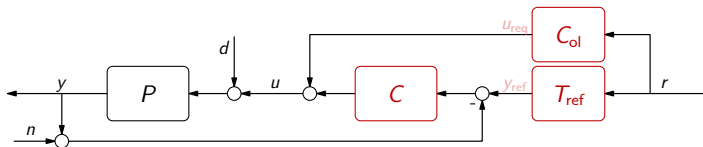
use $T_c P = T$ and $S + T = 1$

$$u = u_{\text{req}} + C(y_{\text{ref}} - Pu) \iff u = Su_{\text{req}} + Tcy_{\text{ref}} = u_{\text{req}}$$

and $y = Pu = y_{\text{ref}}$, regardless C (provided it is stabilizing, of course).

2DOF architecture (when a reference model is used)

If we take $y_{\text{ref}} = T_{\text{ref}}r$ and $u_{\text{req}} = C_{\text{ol}}r$ for $C_{\text{ol}} = P^{-1}T_{\text{ref}}$ (consistent), then



and signals of interest are

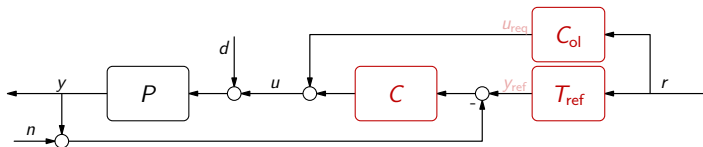
$$\boxed{y = T_{\text{ref}}r + T_d d - T_n n} \quad \& \quad \boxed{u = C_{\text{ol}}r - T_d d - T_c n},$$

This controller blends open- and closed-loop controller architectures, with a complete separation of

- nominal command response (standard open-loop design)
- stabilization and handling uncertainty (standard feedback design, just w/o taking the command response into account)

2DOF architecture (when a reference model is used)

If we take $y_{\text{ref}} = T_{\text{ref}}r$ and $u_{\text{req}} = C_{\text{ol}}r$ for $C_{\text{ol}} = P^{-1}T_{\text{ref}}$ (consistent), then



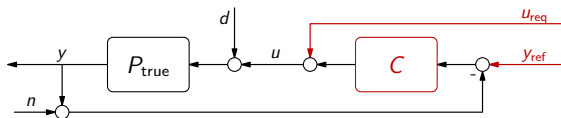
and signals of interest are

$$y = T_{\text{ref}}r + T_d d - T_n n \quad \& \quad u = C_{\text{ol}}r - T_d d - T_c n,$$

This controller blends open- and closed-loop controller architectures, with a complete **separation** of

- nominal command response shaped by C_{ol}
standard open-loop design
- stabilization and handling uncertainty shaped by C
standard feedback design, just w/o taking the command response into account

2DOF architecture in the presence of uncertainty



In this case

$$y = y_{\text{ref}} + T_d d - T n - S(y_{\text{ref}} - P_{\text{true}} u_{\text{req}})$$

and

$$u = u_{\text{req}} - T d - T_c n + T_c(y_{\text{ref}} - P_{\text{true}} u_{\text{req}})$$

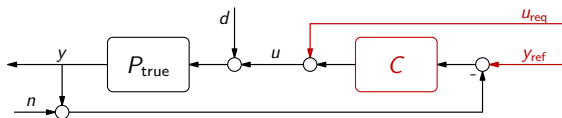
for $S = (1 + P_{\text{true}} C)^{-1}$, et cetera. Mismatched command following can be affected by feedback now. Namely (for simplicity, assume that $d = n = 0$),

$$y_{\text{ref}} = P u_{\text{req}} \implies y_{\text{ref}} - y = S(I - P_{\text{true}} P^{-1}) y_{\text{ref}}.$$

The sensitivity of command following to modeling mismatch $I - P_{\text{true}} P^{-1}$ is reduced at frequencies where we manage to keep $|S(j\omega)| \ll 1$.

For an integrator-based controller, $|S(j\omega)| \ll 1$ is achieved at high frequencies, which is not the case for a PID controller.

2DOF architecture in the presence of uncertainty



In this case

$$y = y_{\text{ref}} + T_d d - T n - S(y_{\text{ref}} - P_{\text{true}} u_{\text{req}})$$

and

$$u = u_{\text{req}} - T d - T_c n + T_c(y_{\text{ref}} - P_{\text{true}} u_{\text{req}})$$

for $S = (1 + P_{\text{true}} C)^{-1}$, et cetera. Mismatched command following can be affected by feedback now. Namely (for simplicity, assume that $d = n = 0$),

$$y_{\text{ref}} = P u_{\text{req}} \implies y_{\text{ref}} - y = S(I - P_{\text{true}} P^{-1}) y_{\text{ref}}.$$

The sensitivity of command following to modeling mismatch $I - P_{\text{true}} P^{-1}$

— is reduced at frequencies where we manage to keep $|S(j\omega)| \ll 1$

(e.g. an integrator in $C(s) \implies S(0) = 0 \implies e_{\text{ss}} = 0$ if $y_{\text{ref}} \xrightarrow{t \rightarrow \infty} \text{const}$).

Outline

Industrial (PID) controllers

Tuning PID controllers

PID controller architectures and implementation

2-degrees-of-freedom controller configuration

Concluding remarks

Summary

Learned in this course,

- flavor of basic control ideas,

so you could communicate with control engineers in their native language.

A long and winding road to become a control engineer / R&D yourselves,

00350188 Control Theory

00340406 Advanced Control Laboratory

00350036 Control Systems Design

00360709 Sampled-Data Control Systems

00360012 Linear Control Systems

00360050 Nonlinear Control Systems

00360013 Process Optimization

Summary

Learned in this course,

- flavor of basic control ideas,
- so you could communicate with control engineers in their native language.

A long and winding road to become a control engineer / R&D yourselves,

00350188 Control Theory

00340406 Advanced Control Laboratory

00350036 Control Systems Design

00360709 Sampled-Data Control Systems

00360012 Linear Control Systems

00360050 Nonlinear Control Systems

00360013 Process Optimization

⋮