- Most important python code for this sub-module
- Self-assessment material

notes

- this is the table of contents of this document; each section corresponds to a specific part of the course

Pole Placement with PID Controllers

notes

Contents map

developed content units	**taxonomy levels**
Pole placement with PID	u2, e3

prerequisite content units	**taxonomy levels**
Transfer functions, 1st/2nd order models	u1, e2
PID control basics	u1, e2

notes

Main ILO of sub-module "Pole Placement with PID Controllers"

Design a PID controller to place the closed-loop poles at desired locations

notes

- By the end of this module you shall be able to design a PID controller that achieves desired dynamics via pole placement.

Why Pole Placement with PID?

- Pole placement helps us control system response: speed, overshoot, damping.
- PID controllers can shift poles by adjusting K_P, K_I, K_D.
- This is useful when we want specific time-domain behavior (e.g., settling time, overshoot).

notes

- Pole placement links desired behavior with controller design

Worked Example: First-Order Plant

Given: $G(s) = \frac{1}{s+1}$ (first-order system)
Goal: Place closed-loop pole at $s = -4$
Try: Use a proportional controller: $C(s) = K_P$
Closed-loop: $\frac{K_P G(s)}{1 + K_P G(s)} = \frac{K_P}{s + 1 + K_P}$
Compare: $s + (1 + K_P) = s + 4 \Rightarrow K_P = 3$

notes

- This shows how to match closed-loop poles with desired dynamics

Second-Order System and PID

Given: $G(s) = \frac{1}{s(s+1)}$ (2nd order, non-minimum phase)

Controller: $C(s) = K_P + \frac{K_I}{s} + K_D s$

Closed-loop char. poly: Denominator of $1 + C(s)G(s)$

Choose desired poles (e.g., $s = -2 \pm j2$):

Desired poly: $s^2 + 4s + 8$

Solve for K_P, K_I, K_D so that closed-loop denominator matches this.

notes

- Start from desired poles get desired polynomial match with real system

Summarizing

From desired poles to PID parameters

- Pick desired poles based on time response specs
- Derive desired characteristic polynomial
- Write closed-loop transfer function with PID
- Match polynomials & solve for K_P, K_I, K_D
- Note that you may not be able to place the poles where you want (i.e., the system above to do not have solutions)

notes

- You should now be able to apply this pseudo-algorithm to tune a PID by pole placement

Most important python code for this sub-module

notes

Python Enables Symbolic Matching of PID Coefficients

`sympy`

notes

- Symbolically compute closed-loop polynomial and match to desired one

Self-assessment material

notes

Question 1

What is the first step in designing a PID controller using pole placement?

Potential answers:

I: **(wrong)** Tune K_P using trial-and-error
II: **(wrong)** Write the plant transfer function in state-space
III: **(correct)** Choose desired closed-loop poles based on time-domain specs
IV: **(wrong)** Set the integral gain to zero initially

Solution 1:

The first step is to decide where you want the poles to bethis determines the desired system behavior.

notes

- see the associated solution(s), if compiled with that ones :)

Question 2

What is the main goal of pole placement when designing a controller?

Potential answers:

I: **(wrong)** To cancel all poles and zeros of the system
II: **(correct)** To achieve desired time-domain behavior such as settling time and overshoot
III: **(wrong)** To make the transfer function purely algebraic
IV: **(wrong)** To eliminate the need for feedback
V: **(wrong)** I do not know

Solution 1:

Pole placement is used to ensure the closed-loop poles correspond to desired system dynamics, influencing speed, damping, and stability.

notes

- see the associated solution(s), if compiled with that ones :)

Question 3

How does the derivative term (K_D) in a PID controller primarily affect the pole placement of a system?

Potential answers:

I: **(wrong)** It shifts the system poles toward the imaginary axis
II: **(wrong)** It always eliminates steady-state error
III: **(wrong)** It has no influence on the pole placement
IV: **(correct)** It influences the damping and stability by modifying the characteristic equation
V: **(wrong)** I do not know

Solution 1:

The derivative term modifies the system's dynamics, particularly by increasing damping and thus influencing the position of the closed-loop poles.

notes

- see the associated solution(s), if compiled with that ones :)

Question 4

What is the key mathematical operation used to design PID gains through pole placement?

Potential answers:

I: **(wrong)** Taking the inverse Laplace transform of the plant
II: **(wrong)** Eliminating zeros from the open-loop transfer function
III: **(correct)** Matching the closed-loop characteristic polynomial to a desired one
IV: **(wrong)** Factorizing the numerator of the open-loop transfer function
V: **(wrong)** I do not know

Solution 1:

Pole placement design requires expressing the closed-loop characteristic equation and matching its coefficients with those of a desired polynomial to solve for the PID gains.

notes

- see the associated solution(s), if compiled with that ones :)

Question 5

In a first-order system controlled by a proportional gain K_P, what is the effect of increasing K_P?

Potential answers:

I: **(correct)** The pole moves further left on the real axis, increasing system speed
II: **(wrong)** The pole becomes complex and causes oscillations
III: **(wrong)** The system gain decreases and response slows down
IV: **(wrong)** The zero of the system moves into the right-half plane
V: **(wrong)** I do not know

Solution 1:

For first-order systems, increasing K_P moves the closed-loop pole leftward (more negative real part), which speeds up the response.

notes

- see the associated solution(s), if compiled with that ones :)

Question 6

Which of the following best describes the correct order of steps for PID pole placement design?

Potential answers:

I: **(wrong)** Compute the system output first, then choose PID gains, then set desired poles
II: **(wrong)** Start with experimental PID gains, simulate, and refine based on intuition
III: **(correct)** Choose desired poles, derive the corresponding characteristic polynomial, and match it with the actual closed-loop polynomial to solve for gains
IV: **(wrong)** Eliminate the need for poles by transforming to frequency domain
V: **(wrong)** I do not know

Solution 1:

Correct PID pole placement design involves selecting desired dynamics, deriving the polynomial that produces them, and solving for PID parameters by matching coefficients.

notes

- see the associated solution(s), if compiled with that ones :)

Recap of sub-module "Pole Placement with PID Controllers"

- Pole placement allows us to achieve desired dynamics
- PID gains shift the closed-loop poles
- Match desired characteristic polynomial with actual one
- Use symbolic or numerical tools to solve for K_P, K_I, K_D

notes

- The most important remarks from this sub-module are these ones