ANSWERS

U5.4 The integral term

Feedback control · University · about 30 min

What this lesson is about

Removing the error that P leaves behind, and the windup that comes with it.

- In code
- Windup
- Choosing Ki

Questions

7 marks in all

1. A PI controller has settled exactly on target. Which term is supplying the command? [1 mark]

- ☐ **A** The integral term
- ☐ **B** The proportional term
- ☐ **C** Both equally
- ☐ **D** Neither, the command is zero

> **Answer:** A. At zero error the P term is zero, so whatever push is needed to stay there comes from the integral's accumulated value.

2. A constant 4 cm error is held for five ticks. What does this print? [1 mark]

```
KP, KI, DT = 1.6, 0.5, 0.1
integral = 0.0
for tick in range(5):
    error = 4.0
    integral += error * DT
    print(round(KP * error + KI * integral, 2))
```

> **Answer:**
>
> ```
> 6.6
> 6.8
> 7.0
> 7.2
> 7.4
> ```
>
> The P term stays at 6.4 while the integral grows by 0.4 per tick, adding 0.2 each time: 6.6, 6.8, 7.0, 7.2, 7.4. A stubborn error keeps pushing harder.

3. Using the lesson's rule of thumb, how many seconds does the integral take to contribute as much as the proportional term when Kp = 1.6 and Ki = 0.5? Give one decimal place. [1 mark]

> **Answer:** 3.2 (accept within 0.05). Kp / Ki = 1.6 / 0.5 = 3.2 s. Make it comparable with the settling time you want.

4. Which of these are the standard defences against integral windup given in the lesson? [1 mark]

Tick every answer that is true.

- ☐ **A** Clamp the integral to a sensible range
- ☐ **B** Stop integrating while the output is saturated
- ☐ **C** Subtract the clipped amount, scaled, from the integral
- ☐ **D** Raise Kp so the error shrinks faster
- ☐ **E** Filter the error before integrating it

> **Answer:** A, B, C. Clamping, conditional integration and back-calculation all stop the integral storing effort the actuator could not deliver. Gain or filtering changes do not.

5. An obstacle holds the robot at 40 cm while the target is 25 cm. Ten seconds later it is removed and the robot charges far past the target. Why? [1 mark]

- ☐ **A** The integral grew the whole time and must be unwound by an error in the other direction
- ☐ **B** Kp is too high
- ☐ **C** The D term amplified the removal of the obstacle
- ☐ **D** The sensor was biased by the obstacle

> **Answer:** A. That is windup: a large stored integral pins the command at full speed until an equal and opposite error cancels it.

6. The error is in cm, the command in percent, and time in seconds. What are the units of Ki? [1 mark]

> **Answer:** percent per centimetre per second. Ki multiplies the integral of error, in cm s, to give percent, so its units are percent per centimetre per second.

7. The robot sails slowly past the target, turns round, and sails past again with large, slow swings. Which gain is the likely cause? [1 mark]

- ☐ **A** Ki too large
- ☐ **B** Kd too large
- ☐ **C** Kp too small
- ☐ **D** The dead band too small

> **Answer:** A. Too much integral action is its own kind of oscillation: slow and large. Start with an integral that acts slower than the P term.

The task: no offset left

Settle exactly 25 cm from the wall, within 3 cm, from fourteen seconds onwards. Proportional control alone at a gentle gain will not do it: at `Kp = 1.5` it stops about 8 cm short, because this drive does nothing below about 15 percent. Plot `error` and `i term`.

```
from bugbot import *
connect()

KP, KI = 1.5, 0.8
TARGET = 25.0
DT = 0.1
integral = 0.0
```

The hint students can ask for: 25 cm from the wall is y = 65 from the start, and the tolerance is 3 cm, which proportional control at a gentle gain will not manage: below about 15 percent this drive does not move at all, so a small error produces a command that does nothing. Add up the error over time and let that push as well.

A solution

```
from bugbot import *
connect()

KP, KI = 1.6, 0.9
TARGET = 25.0
DT = 0.1
integral = 0.0

for tick in range(480):
    error = distance() - TARGET
    integral += error * DT
    integral = max(-60.0, min(60.0, integral))
    cmd = KP * error + KI * integral
    plot("error", error)
    plot("i term", KI * integral)
    drive(max(-60.0, min(60.0, cmd)), 0, 0)
    wait(DT)
stop()
```

Any program that meets the task's checks is marked correct in the simulator; this is one way, not the only way.