ANSWERS

U5.3 The derivative term

Feedback control · University · about 30 min

What this lesson is about

Damping the overshoot, and why D on a noisy signal needs a filter or it is useless.

- In code
- Why D is difficult in practice
- The shape it produces

Questions

6 marks in all

1. The distance sensor in the lesson wobbles by about 1 cm, so two readings 0.1 s apart can differ by 4 cm with the robot all but still. What approach speed, in cm/s, does a differenced derivative report? [1 mark]

> **Answer:** 40. 4 cm / 0.1 s = 40 cm/s of motion that is not happening, about three times the robot's speed at 70 percent. Multiplied by Kd, that is what makes the motors kick.

2. One tick of the PD controller from the lesson. What does it print? [1 mark]

```
KP, KD, DT = 3.0, 1.2, 0.1
last, error = 20.0, 14.0
d = (error - last) / DT
cmd = max(-70, min(70, KP * error + KD * d))
print(round(d, 1), round(cmd, 1))
```

> **Answer:**
>
> ```
> -60.0 -30.0
> ```
>
> The error fell by 6 cm in 0.1 s, so d = -60 cm/s. The command is 3 x 14 + 1.2 x (-60) = 42 - 72 = -30: the controller is already braking before it arrives.

3. The robot is approaching the target quickly, so the error is shrinking fast. What does the D term do? [1 mark]

- ☐ **A** It is negative and subtracts from the command, easing off before arrival
- ☐ **B** It is positive and adds to the command
- ☐ **C** It is zero until the error changes sign
- ☐ **D** It accumulates the error over time

> **Answer:** A. A fast-shrinking error has a large negative derivative, so Kd x d reduces the command. That is damping.

4. The setpoint jumps from 30 to 40 cm and the command spikes hugely for one tick. What is the standard fix? [1 mark]

- ☐ **A** Differentiate the measurement instead of the error
- ☐ **B** Lower Kp
- ☐ **C** Filter the error
- ☐ **D** Clamp the integral

> **Answer:** A. The error jumps with the setpoint, and the derivative of a jump is enormous. The measurement does not jump, so differentiating it gives the same damping without the kick.

5. Which is the best treatment for derivative noise on this robot, according to the lesson? [1 mark]

- ☐ **A** Use the measured speed from flow() instead of differencing distance()
- ☐ **B** Filter distance() heavily before differencing
- ☐ **C** Lower the loop rate so the differences are bigger
- ☐ **D** Raise Kd until the noise averages out

> **Answer:** A. A measured rate is far better behaved than a computed one and adds no filter delay, which is why serious systems put a rate sensor on anything they differentiate.

6. On the quarter turn, Kp = 4 alone overshoots by about 20 degrees. With a D term added, the response has no overshoot, a slower approach, and stops about 1.4 degrees short. What does that say about Kd? [1 mark]

- ☐ **A** It is on the high side: a lot of D
- ☐ **B** There is no D at all
- ☐ **C** It is too small to matter
- ☐ **D** It is making the loop wind up

> **Answer:** A. Heavy damping removes the overshoot and slows the approach, and braking that early leaves the last push inside the dead band. A little D, 0.8 in the lesson, keeps the speed and loses the overshoot.

The task: damp the overshoot

This robot is carrying a load, so it takes about three times as long to speed up and to slow down. That extra lag is exactly the job the D term is for. Turn to face 90 degrees and be within 3 degrees of it from three seconds onwards. Plot `error` and `d term`. On this robot P alone cannot do it, whatever the gain and however fast the loop runs: a gain low enough not to swing past arrives too late, and a gain high enough to arrive in time swings past and is still swinging at three seconds. Start from `KP = 4` and find a `KD` that brakes it in time.

```
from bugbot import *
connect()

KP, KD = 4.0, 0.0
DT = 0.25
last = 90.0          # the error at the start
```

The hint students can ask for: This robot is carrying a load, so it takes about three times as long to speed up and slow down. With P alone it either creeps in too slowly or swings past, whatever the gain. Add a term that pushes against how fast the error is shrinking, so the robot starts braking before it arrives.

A solution

```
from bugbot import *
connect()

KP, KD = 4.0, 1.6
DT = 0.25
last = 90.0          # the error at the start

for tick in range(40):
    error = (90 - heading() + 180) % 360 - 180
    d = (error - last) / DT
    last = error
    plot("error", error)
    plot("d term", KD * d)
    drive(0, 0, KP * error + KD * d)
    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.