ANSWERS

3.2 Proportional control

BugBotLab

Control · Robot club · about 15 min

What this lesson is about

Correct in proportion to the error. Gain, sign, feedback.

- The robot that curves
- Steer by the error
- The sign
- Why it is called feedback

Questions

6 marks in all

1. What does a proportional controller do? [1 mark]

- ☐ **A** Pushes in proportion to the error: hard when far off, gently when close
- ☐ **B** Goes at one fixed speed until the error is zero
- ☐ **C** Turns a set amount every loop
- ☐ **D** Waits until the error is big before doing anything

> **Answer:** A. The correction is the error times a gain, so it fades away as the error does.

2. The gain is 4 and the heading error is 7.5 degrees. What rotation does `drive(70, 0, error * 4)` ask for? [1 mark]

> **Answer:** 30. 7.5 times 4 is 30. Half the error would give half the rotation.

3. What does this program print? [1 mark]

```
def wrapped(h):
    return (h + 180) % 360 - 180

for heading in [10, 350]:
    error = wrapped(0 - heading)
    print(heading, error * 4)
```

> **Answer:**
>
> ```
> 10 -40
> 350 40
> ```
>
> Facing 10, the robot is right of the target, so the rotation is negative and turns it back left. Facing 350, it is left of the target, so it turns right.

4. You write `drive(70, 0, -error * 4)` by mistake. What happens? [1 mark]

- ☐ **A** The controller runs away and the error grows
- ☐ **B** The robot holds the heading more gently
- ☐ **C** Nothing, the sign does not matter
- ☐ **D** The robot drives backwards

> **Answer:** A. With the wrong sign every correction pushes further the wrong way. When a controller does something wild, check the sign first.

5. Why is it called a feedback loop? [1 mark]

- ☐ **A** The heading comes out of the robot and goes back to the motors as a correction
- ☐ **B** The robot prints its heading every loop
- ☐ **C** It uses a `for` loop
- ☐ **D** The robot drives in a circle

> **Answer:** A. Output is fed back to input. Open loop hopes; closed loop checks.

6. Why does the P controller not need bursts and settling like the crude fix? [1 mark]

- ☐ **A** Small errors give small rotations, so the correction fades away with no overshoot
- ☐ **B** It always drives at a low speed
- ☐ **C** It stops after every loop
- ☐ **D** It only runs once

> **Answer:** A. Four degrees off gives rotation 16, half a degree gives 2. The robot eases in rather than slamming on.

The task: hold the heading

Drive into the green zone and finish still facing 0, within 6 degrees, on the curving robot.

```
# the two lines every program starts with: the commands, then the robot
from bugbot import *
connect()

# drive forward at 70 for 70 cm, then stop
forward(70, distance=70)
```

The hint students can ask for: This robot curves to the right whenever it drives. Steer against the heading error while driving, and finish in the green zone still facing 0.

A solution

```
from bugbot import *
connect()
def wrapped(h):
    return (h + 180) % 360 - 180
while position()[1] < 65:
    error = wrapped(heading())
    drive(70, 0, -error * 4)
    wait(0.1)
stop()
```

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