ANSWERS

8.6 Tuning itself

Learning · Robot club · about 20 min

What this lesson is about

Learning a controller gain by trial: measure, change, keep the better.

- This robot leaks
- Steering by the error
- The search

Questions

7 marks in all

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

```
xs = [2, -4, 1, -1]
errors = [abs(x) for x in xs]
print(sum(errors) / len(errors))
```

Answer:

```
2.0
```

Take the size of each error, ignoring its sign: 2, 4, 1 and 1. Their mean is 8 / 4 = 2.0.

2. A trial's score is the mean of `abs(x)`. Gain 6 scores 2.2 and gain 12 scores 1.3. Which is better? [1 mark]

- ☐ **A** Gain 12, because a lower mean error is better
- ☐ **B** Gain 6, because a higher score is better
- ☐ **C** They are the same
- ☐ **D** It cannot be told without a third trial

Answer: A. The score measures how far off the centre line the robot was on average. Lower is better.

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

```
def trial(gain):
    return abs(gain - 10) + 1

gain, direction = 2.0, 1
best_gain, best_error = gain, 1e9
for t in range(5):
    error = trial(gain)
    if error < best_error:
        best_gain, best_error = gain, error
    else:
        direction = -direction
    gain = max(1.0, best_gain + direction * 4.0)
print(best_gain)
```

Answer:

```
10.0
```

It tries 2, 6 and 10, each better. 14 is worse, so it turns round and tries 6 from the best, also worse, then turns again. The best stays 10.0.

4. Why does a gain of 1 never steer this robot until it is 15 cm out? [1 mark]

- ☐ **A** A sideways command under about 15 does nothing, so -x x 1 is too small to move it
- ☐ **B** Gain 1 is below the servo's range
- ☐ **C** The position reading only updates every 15 cm
- ☐ **D** The lane is 15 cm wide

Answer: A. The motors need a minimum push. That is why the useful gains here are bigger than Module 3's.

5. Suppose a robot leaks 2 cm sideways every second, and the lane is 8 cm wide with the robot starting on the centre line. After how many seconds is it out of the lane? [1 mark]

Answer: 2. Half the lane is 4 cm, and at 2 cm per second that takes 2 seconds.

6. In the hill climb, a trial scores worse than the best so far. What happens next? [1 mark]

- ☐ **A** It turns round and steps the other way from the best gain
- ☐ **B** It stops searching
- ☐ **C** It keeps stepping the same way from the worse gain
- ☐ **D** It resets the gain to 2

Answer: A. Better: keep going that way. Worse: turn round, starting again from the best.

7. The same measure, change, keep the best loop can tune which of these? [1 mark]

Tick every answer that is true.

- ☐ **A** A driving speed
- ☐ **B** A threshold
- ☐ **C** A timeout
- ☐ **D** The label on a sample

> **Answer:** A, B, C. It tunes anything with a number in it that you can score. A label is your judgement, not a number to optimise.

The task: tune yourself

Drive up and down the lane, trying gains and keeping the best. After 30 seconds the robot must stay inside the lane. Print `best gain: <value>` at the end.

```
# the two lines every program starts with: the commands, then the robot
from bugbot import *
connect()
gain = 0.3
# do this 10 times (leg counts from 0)
for leg in range(10):
    # do this 50 times (tick counts from 0)
    for tick in range(50):
        # where am I? (cm from where I started)
        x, y = position()
        # forward, sideways, rotation: -100 to 100 each, until the next command
        drive(40 if leg % 2 == 0 else -40, -x * gain, 0)
        # pause 0.1 s (the robot keeps doing what it was told)
        wait(0.1)
# all motors off
stop()
print(f'best gain: {gain:.1f}')
```

The hint students can ask for: This robot leaks badly sideways. Drive up and down the lane, and let the program try different sideways gains, keeping whichever holds the lane best. After 30 seconds it must stay inside the lane. Print `best gain: <value>` at the end.

A solution

```
from bugbot import *
connect()
gain = 2.0
best_gain, best_error = gain, 1e9
direction = 1
for trial in range(5):
    errors = []
    for leg in range(2):
        for tick in range(50):
            x, y = position()
            drive(40 if leg == 0 else -40, -x * gain, 0)       # up 5 s, back 5 s, steering
sideways from x
            errors.append(abs(x))
            wait(0.1)
    error = sum(errors) / len(errors)
    print(f"trial {trial}: gain {gain:.1f}, mean error {error:.1f}")
    if error < best_error:
        best_gain, best_error = gain, error
    else:
        direction = -direction                                 # that was worse: try the
other way
    gain = max(1.0, best_gain + direction * 4.0)
stop()
print(f"best gain: {best_gain:.1f}")
```

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