ANSWERS

U5.7 Project: park it

Feedback control · University · about 45 min

What this lesson is about

Three loops at once: x, y and heading, all settling together in a tight box.

- The structure
- What makes it interesting
- What comes next

Questions

6 marks in all

1. The angle wrap from the parking controller. What does it print? [1 mark]

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

print(wrapped(90 - 350), wrapped(270), wrapped(-190))
```

Answer:

```
100 -90 170
```

90 - 350 = -260, which is 100 degrees the other way. 270 is -90, and -190 is 170. The controller always turns the short way round.

2. The world-frame velocity is turned into drive commands at heading 90. What does it print? [1 mark]

```
import math
V_MAX, V_LAT = 20.0, 15.0
wx, wy, h = 13.0, 0.0, 90.0
a = math.radians(h)
fwd = 100 * (wx * math.sin(a) + wy * math.cos(a)) / V_MAX
lat = 100 * (wx * math.cos(a) - wy * math.sin(a)) / V_LAT
print(round(fwd, 1), round(lat, 1))
```

Answer:

```
65.0 0.0
```

Facing 90 degrees, a world velocity along x is straight ahead, so all 13 cm/s goes forwards: 100 x 13 / 20 = 65 percent, and nothing laterally.

3. The heading is read once, before the loop, and used for all the maths while the heading controller turns the robot 130 degrees. What does the lesson say happens? [1 mark]

- ☐ **A** The robot drives away from the target and into the edge of the mat
- ☐ **B** It curves in slowly but still parks
- ☐ **C** The robot spins on the spot
- ☐ **D** No visible effect, since the position loop corrects it

> **Answer:** A. The rotation matrix needs the heading the robot has now. After a 130 degree turn a stale heading aims every move the wrong way, so the position errors grow instead of shrinking. Read heading() every tick.

4. Why should the position gain and the heading gain not simply be equal? [1 mark]

- ☐ **A** 90 degrees of heading error and 90 cm of position error are different sizes of problem, and both errors should reach zero together
- ☐ **B** Heading is measured in radians inside drive()
- ☐ **C** The heading loop must always be ten times faster
- ☐ **D** Equal gains make the loops unstable

> **Answer:** A. If the heading settles in one second and the position in ten, the robot twitches its nose for nine seconds. Scale the gains so they finish together.

5. The robot steers by odometry() instead of position() and heading(). Over one short park it ends a couple of centimetres further out, and over a long run it drifts much further. What is needed? [1 mark]

- ☐ **A** A better state estimate, fusing measurements with dead reckoning
- ☐ **B** Higher controller gains
- ☐ **C** A wider dead band
- ☐ **D** A slower loop

> **Answer:** A. The controller is fine; it is steering by a drifting estimate. An estimator that is better than dead reckoning is the subject of U6.

6. Rotating while translating moves the position errors even when the position controller has done nothing new. What does the lesson call this? [1 mark]

- ☐ **A** The loops interacting, or coupling
- ☐ **B** Integral windup
- ☐ **C** Derivative kick
- ☐ **D** Steady state error

> **Answer:** A. The three loops share one robot, so each disturbs the others. Mild here; on an aircraft it is the whole subject of coupled dynamics.

The task: park it

Stop within 6 cm of the target and within 8 degrees of heading 90, and be settled there by twenty seconds. The target is 80 across and 90 up from where the robot starts.

```
from bugbot import *
import math
connect()

V_MAX, V_LAT, W_MAX = 20.0, 15.0, 120.0
TX, TY, TH = 80.0, 90.0, 90.0
```

The hint students can ask for: Three errors at once: x, y and heading. Run a proportional controller on each, put the two translation terms through the inverse kinematics from U2, and add the heading term as rotation. The target is 80 across and 90 up from the start, facing 90.

A solution

```
from bugbot import *
import math
connect()

V_MAX, V_LAT, W_MAX = 20.0, 15.0, 120.0
TX, TY, TH = 80.0, 90.0, 90.0

def wrapped(a):
    return (a + 180) % 360 - 180

for tick in range(500):
    x, y = position()
    ex, ey = TX - x, TY - y
    eh = wrapped(TH - heading())
    wx = max(-13.0, min(13.0, 0.7 * ex))
    wy = max(-13.0, min(13.0, 0.7 * ey))
    a = math.radians(heading())
    fwd = 100 * (wx * math.sin(a) + wy * math.cos(a)) / V_MAX
    lat = 100 * (wx * math.cos(a) - wy * math.sin(a)) / V_LAT
    rot = 100 * max(-70.0, min(70.0, 2.0 * eh)) / W_MAX
    if abs(ex) < 2 and abs(ey) < 2 and abs(eh) < 3:
        stop()
    else:
        drive(fwd, lat, rot)
    wait(0.1)
stop()
print("parked at", position(), round(heading()))
```

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