ANSWERS

U6.4 Q and R

State estimation · University · about 30 min

What this lesson is about

The two numbers you actually choose, what they mean, and how to measure them.

- R: how noisy the measurement is
- Q: how wrong the model can be
- The two failure modes
- Checking a filter honestly

Questions

7 marks in all

1. Standing still, a depth sensor's readings have a standard deviation of 3 cm. What R should the filter use? [1 mark]

> **Answer:** 9. R is the variance of the measurement noise, sigma squared: 3 x 3 = 9 cm squared. Using 3 is the classic sigma versus variance mistake.

2. The robot can slip about 0.3 cm in one tick without the model knowing. Using the lesson's rough rule, what is Q? [1 mark]

> **Answer:** 0.09 (accept within 0.001). Q is roughly the square of how far the state can drift unseen in one tick: 0.3 squared = 0.09.

3. Q is doubled and R is doubled. What happens to the Kalman gain? [1 mark]

- ☐ **A** Nothing, because only the ratio Q/R matters
- ☐ **B** It doubles
- ☐ **C** It halves
- ☐ **D** It rises, because Q grew

> **Answer:** A. Scaling both by the same factor scales p by it too, and k = p / (p + R) is unchanged.

4. Q is set far too small. How does the filter fail? [1 mark]

- ☐ **A** It trusts its model, drifts away from the truth, and reports a small variance the whole time
- ☐ **B** Its output is as noisy as the raw sensor
- ☐ **C** It oscillates around the truth
- ☐ **D** It stops predicting and only uses measurements

> **Answer:** A. That is divergence, and it is the dangerous failure because the filter's own uncertainty is the part that is wrong. Q too large gives the noisy output instead.

5. The normalised innovation squared for four fixes, with p + R = 25 each time. What does it print? [1 mark]

```
innovations = [3.0, -6.0, 4.0, -2.0]
S = 25.0
nis = [v ** 2 / S for v in innovations]
print(nis)
print(round(sum(nis) / len(nis), 2))
```

Answer:

```
[0.36, 1.44, 0.64, 0.16]
0.65
```

The squares 9, 36, 16 and 4, over 25, average 0.65. It should be about 1, so this filter is claiming somewhat more uncertainty than it has and could be tuned tighter.

6. A filter's running average of the normalised innovation squared is about 4. What does it mean? [1 mark]

- ☐ **A** The filter claims less uncertainty than it really has, which is the dangerous direction
- ☐ **B** The filter claims more uncertainty than it has and could be tuned tighter
- ☐ **C** The filter is well tuned
- ☐ **D** The measurements are four times more accurate than R says

Answer: A. Innovations much larger than p + R predicts mean the filter is overconfident. Raising Q or R, or finding the model error, is needed.

7. Why is Q = 0 always wrong on a real machine? [1 mark]

- ☐ **A** The variance shrinks towards zero, so the filter eventually stops listening to measurements
- ☐ **B** It makes the gain equal to 1
- ☐ **C** It divides by zero in the gain
- ☐ **D** It makes R meaningless

Answer: A. With no process noise p only ever shrinks, k goes to zero, and the filter ignores every later measurement even though no real model is perfect.

The task: tune Q and R

Standing still, measure `R` and print it as `measured r:`. Then run the filter at two or three values of `Q` and print what each does.

```
from bugbot import *
connect()

readings = []
```

The hint students can ask for: R is measurable: stand still, take a hundred readings, and square the standard deviation. Q is a statement about how much the state can change between ticks that your model does not know about. Print R, then try two or three values of Q and say what each does.

A solution

```
from bugbot import *
connect()

DT = 0.1
readings = []
for i in range(100):
    readings.append(distance())
    wait(DT)
mean = sum(readings) / len(readings)
R = sum((v - mean) ** 2 for v in readings) / len(readings)
print("measured r:", round(R, 1))

for Q in (0.01, 1.0, 100.0):
    est, p = readings[0], 100.0
    for v in readings:
        p += Q
        k = p / (p + R)
        est += k * (v - est)
        p = (1 - k) * p
    print("Q", Q, "settles at gain", round(p / (p + R), 3), "estimate", round(est, 1))
```

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