WORKSHEET

U6.4 Q and R

State estimation · University · about 30 min

Name _____ Class _____ Date _____

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]

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]

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

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

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))
```

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

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

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 = []
```

Plan your program here, then type it in and press Run.

Do it on the robot

www.bugbotlab.com/learn/u6-4-q-and-r/

The simulator checks it and tells you when it passes. Nothing to install, no account.

Challenges

1. Compute the normalised innovation squared for your filter and check it is near 1.
2. Set `Q = 0` and drive the robot. Watch the estimate stop listening.
3. Make `R` depend on range: double it beyond 100 cm. Does the estimate improve?