ANSWERS

U4.3 The low pass filter

Noise and filtering · University · about 30 min

What this lesson is about

A moving average, then the exponential filter that does the same job in one line and no memory.

- The moving average
- The exponential filter
- It is the same idea as the Kalman filter
- Starting it

Questions

7 marks in all

1. An exponential filter starts at 50 and sees three readings of 60. What does it print? [1 mark]

```
ALPHA = 0.2
filtered = 50.0
for raw in (60, 60, 60):
    filtered = ALPHA * raw + (1 - ALPHA) * filtered
    print(round(filtered, 2))
```

Answer:

```
52.0
53.6
54.88
```

Each step closes a fifth of the remaining gap: 50 + 0.2 x 10 = 52, then 52 + 0.2 x 8 = 53.6, then 53.6 + 0.2 x 6.4 = 54.88.

2. Using the lesson's rule of thumb, an exponential filter with alpha = 0.1 behaves roughly like a moving average of how many readings? [1 mark]

Answer: 19. The equivalent window is (2 - alpha) / alpha = 1.9 / 0.1 = 19 readings.

3. Why should an exponential filter be started at the first reading rather than at 0? [1 mark]

- ☐ **A** Starting at 0 makes the output climb from zero for the first second, which looks like a real signal
- ☐ **B** Starting at 0 makes alpha wrong for the whole run
- ☐ **C** Starting at 0 makes the filter unstable
- ☐ **D** The filter needs a nonzero value to avoid dividing by zero

Answer: A. From 0 the filter spends its first steps converging on reality. That startup transient is easily blamed on the hardware; starting at the first reading removes it.

4. Compared with a moving average giving similar smoothing, what is the main practical advantage of the exponential filter? [1 mark]

- ☐ **A** It stores only its previous output, instead of the last n readings
- ☐ **B** It has no delay
- ☐ **C** It rejects outliers
- ☐ **D** Its output reaches a step's new value in exactly one step

> **Answer:** A. The moving average costs n stored numbers and n additions per tick. The exponential filter does the same job in one line with one stored value. Both still delay.

5. Rearranged, the filter is filtered = filtered + alpha * (raw - filtered). In the Kalman filter of U6.3 the update has the same shape. What is different there? [1 mark]

- ☐ **A** The weighting is computed each tick from how uncertain the belief is compared with the measurement
- ☐ **B** The weighting is always 0.5
- ☐ **C** The measurement is filtered twice
- ☐ **D** The difference raw - filtered is squared

> **Answer:** A. The Kalman gain plays the role of alpha, but it is recalculated every tick from the variances instead of being a constant you chose.

6. In signal processing terms, the exponential filter is a first order what kind of filter? [1 mark]

> **Answer:** low pass. It passes slow changes and attenuates fast ones, so it is a first order low pass filter, also called an exponentially weighted moving average.

7. Put these lines in order to run an exponential filter while driving. [1 mark]

Number the lines 1 to 5 to put them in the right order.

___ `filtered = ALPHA * raw + (1 - ALPHA) * filtered`
___ `for i in range(60):`
___ `raw = distance()`
___ `plot("filtered", filtered)`
___ `filtered = distance()`

> **Answer:**
>
> ```
> filtered = distance()
> for i in range(60):
>     raw = distance()
>     filtered = ALPHA * raw + (1 - ALPHA) * filtered
>     plot("filtered", filtered)
> ```
>
> Start the filter at a real reading, then each tick read, update, and plot the filtered value.

The task: a low pass filter

Drive towards the wall with an exponential filter running, plotting `raw` and `filtered` as you go, and print the `alpha:` you chose.

```
from bugbot import *
connect()

ALPHA = 0.25
filtered = distance()
```

The hint students can ask for: filtered = alpha * new + (1 - alpha) * filtered, once per tick. Plot both lines while the robot drives towards the wall, and watch the filtered line follow the raw one at a distance.

A solution

```
from bugbot import *
connect()

ALPHA = 0.25
filtered = distance()
forward(45)
for i in range(70):
    raw = distance()
    filtered = ALPHA * raw + (1 - ALPHA) * filtered
    plot("raw", raw)
    plot("filtered", filtered)
    if filtered < 30:
        stop()
    wait(0.1)
stop()
print("alpha:", ALPHA)
```

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