FILTERS
Take an fan which has been just turned off....

Sensor:
Tachometer measuring the angular velocity
What we know about it:

Due to viscous friction the velocity slowly decreases to 0.

There might be some random torques (say due to air currents)

\[ T = I \frac{d\omega}{dt} = -k \omega \quad (+\text{random } T) \]

\[ \frac{d\omega}{dt} = -c \omega \quad (+\text{random accl.}) \]
Here is the data measured by Tachometer....
Pretty noisy!

Solution:
Filter of the noise
Finite Impulse response filters (window filter)

• Idea:
  current data = linear function of a few previous data points (including the new one).

\[ y_n^{filt} = a_n y_n + a_{n-1} y_{n-1} + a_{n-2} y_{n-2} \]

where: \( a_n + a_{n-1} + a_{n-2} = 1 \)

• If all the 'a's are 1/3. We call it a window average.
Let's try window averages

Problems:
- Delay
- Initialization
INFinite Impulse reponse filters (exponential filter)

• Idea:
  current data = linear function of a few previous data points (including new one) + few previous filtered data points

• Example: exponential filter

\[ y_{n}^{filt} = \alpha y_{n-1}^{filt} + (1 - \alpha) y_{n} \]

where: \( \alpha \) is the filter factor
Exponential filter

Issues:
• Delay
• Not much problem in Initialization
• Easier to code higher order filters
But these are not very good!

Even our eye can do much better

We are using our intuition about how the line should be, to filter in our mind

We are using **A MODEL**

of

the *process* and the *noise*
Using more information

- If we had 2 independent sensors (y1 and y2) & we know their errors variance (V1 and V2) then we can do better:

- Variance of combination:
  \[
  y_{total} = \beta y_1 + (1 - \beta) y_2 \\
  V_{total} = \beta^2 V_1 + (1 - \beta)^2 V_2
  \]

- Minimum variance:
  \[
  V_{total}^{\text{min var}} = \frac{V_2 V_1}{V_1 + V_2} \quad \text{at:} \quad \beta = \frac{V_2}{V_1 + V_2}
  \]
Kalman filter

• 1st sensor = tachometer, 2nd sensor = model!

• **Step1:** Predict
  \[ \omega_{n+1} = (1 - c \, \delta \, t) \omega_n \]
  \[ V_{n+1} = (1 - c \, \delta \, t)^2 V_n + Q \]

• **Step2:** Update
  \[ \omega_{n+1} = \beta \omega_{n+1} + (1 - \beta) y_{n+1} \]
  \[ \beta = \frac{R}{(V_{n+1}^\gamma + R)} \quad \text{and} \quad V_{n+1} = \frac{V_{n+1}^\gamma R}{(V_{n+1}^\gamma + R)} \]

• People prefer to call \((1 - \text{beta})\) as Kalman gain \((K)\) and they write it as:
  \[ \omega_{n+1} = \omega_{n+1}^\gamma + K \left( y_{n+1} - \omega_{n+1}^\gamma \right) \]
Kalman filter

fan speed

- real data
- noisy data
- window avg of size 8
- exp filter factor 0.8
- kalman filter R = 0.08 Q = 0.0004
Issues

- Computationally more expensive
- Is linear only,
- works best for uncorrelated noise.
- Sensitive to unmodelled dynamics and process noise.