Recursive expression for the average

k data points $(x_1, x_2, \dots, x_k)$

$$\bar{x}_k = \frac{x_1 + x_2 + \cdots + x_k}{k} \quad \text{(batch expression)}$$

if k is large $k > 10^6$, computationally expensive

$$\bar{x}_{k-1} = \frac{x_1 + x_2 + \cdots + x_{k-1}}{k-1}$$

$$k\,\bar{x}_k = \underbrace{x_1 + x_2 + \cdots + x_{k-1}}_{(k-1)\bar{x}_{k-1}} + x_k$$

divide both sides by k-1

$$\frac{k}{k-1}\bar{x}_k = \bar{x}_{k-1} + \frac{x_k}{k-1}$$

multiply both sides by $\frac{k-1}{k}$

$$\boxed{\bar{x}_k = \left(\frac{k-1}{k}\right)\bar{x}_{k-1} + \frac{1}{k}x_k}$$

Recursive expression for average

(faster, more efficient)

gives $\bar{x}_k$ from previous ave, $\bar{x}_{k-1}$, and new data point x_k, and # data points, k

Example, $x = 10, 20, 30$ | $\bar{x}_3 = \frac{10+20+30}{3} = 20$

k=1 | $\bar{x}_1 = \cancel{\frac{1-1}{1}}\bar{x}_0 + \frac{1}{1}x_1 = 10$

k=2 | $\bar{x}_2 = \frac{1}{2}\bar{x}_1 + \frac{1}{2}x_2 = \frac{1}{2}10 + \frac{1}{2}20 = 15$

k=3 | $\bar{x}_3 = \frac{2}{3}\bar{x}_2 + \frac{1}{3}x_3 = \frac{2}{3}15 + \frac{1}{3}30 = 20$

Let's define $\alpha \equiv \frac{k-1}{k} = 1 - \frac{1}{k}$

$$\Rightarrow \frac{1}{k} = 1 - \alpha$$

Recursive expression for average is

$$\boxed{\bar{x}_k = \alpha\bar{x}_{k-1} + (1-\alpha)x_k}$$

where $\alpha \equiv \frac{k-1}{k}$

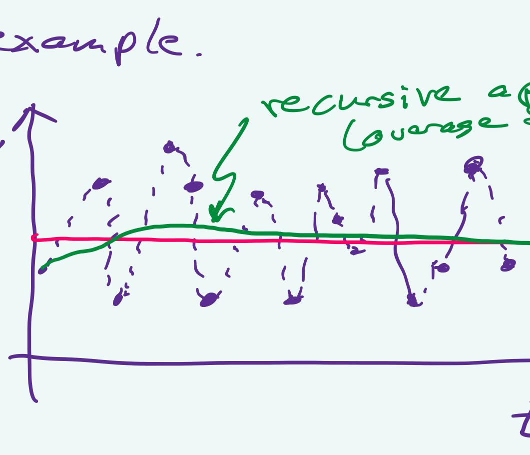
'average filter'

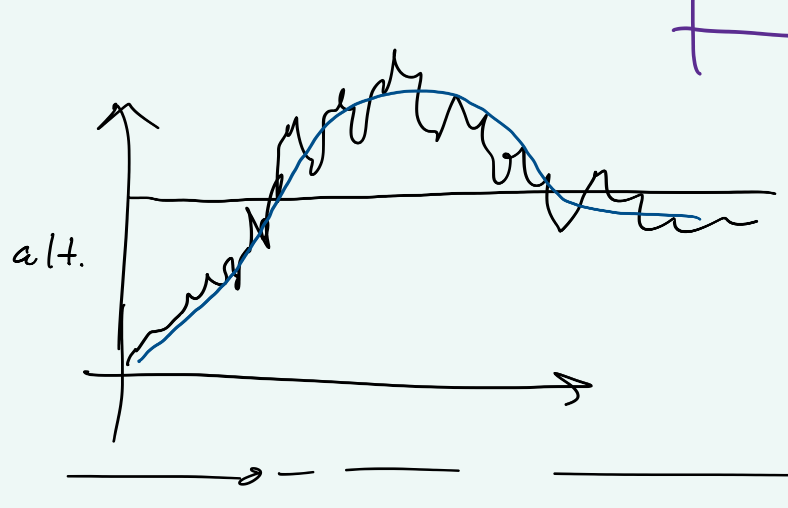
Apply to MATLAB example.

measuring voltage

14.4V + noise

[-4, 4] V





signal changing

→ moving average

Moving average filter

$$\bar{x}_k = \frac{\overbrace{x_{k-n+1} + x_{k-n+2} + \cdots + x_k}^{n \text{ data points}}}{n}$$

different than before.

at the k^{th} data point, we've computed the average of the previous n data points

Written recursively

$$\boxed{\bar{x}_k = \bar{x}_{k-1} + \frac{x_k - x_{k-n}}{n}}$$

Moving average filter

Apply to example sonar altitude data MATLAB (which is noisy)

Low-pass filter

Allows low frequencies (signal) and filters out high frequencies (noise)

Limitations of moving average

$$\bar{x}_k = \frac{x_{k-n+1} + x_{k-n+2} + \cdots + x_k}{n}$$

$$= \frac{1}{n}x_{k-n+1} + \frac{1}{n}x_{k-n+2} + \cdots + \frac{1}{n}x_k$$

each term has same weighting 1/n

we want to weight more recent data higher than older data

1^{st} order low pass filter:

$$\boxed{\bar{x}_k = \alpha\bar{x}_{k-1} + (1-\alpha)x_k, \quad 0 < \alpha < 1}$$

looks like expression for average, but here α is a free parameter

$$\left[\alpha = \frac{k-1}{k} \to 1\right]$$

From now on, $\bar{x}_k$ won't be called "average"
we'll call it the "estimate"

Does this overcome problems w/ moving average?

$$\bar{x}_{k-1} = \alpha\bar{x}_{k-2} + (1-\alpha)x_{k-1}$$

sub into above expression

$$\bar{x}_k = \alpha\left[\alpha\bar{x}_{k-2} + (1-\alpha)x_{k-1}\right] + (1-\alpha)x_k$$

$$= \alpha^2\bar{x}_{k-2} + \alpha(1-\alpha)x_{k-1} + (1-\alpha)x_k$$

$\alpha(1-\alpha) < (1-\alpha)$

so previous data gets weighted exponentially less

MATLAB demo w/ same sonar altitude data

Introduction to the Kalman Filter

- It's like a low pass filter with a dynamically changing α
- You can apply KF to real-world data without the heavy theoretical background

Kalman filter algorithm

measurements & states can be different

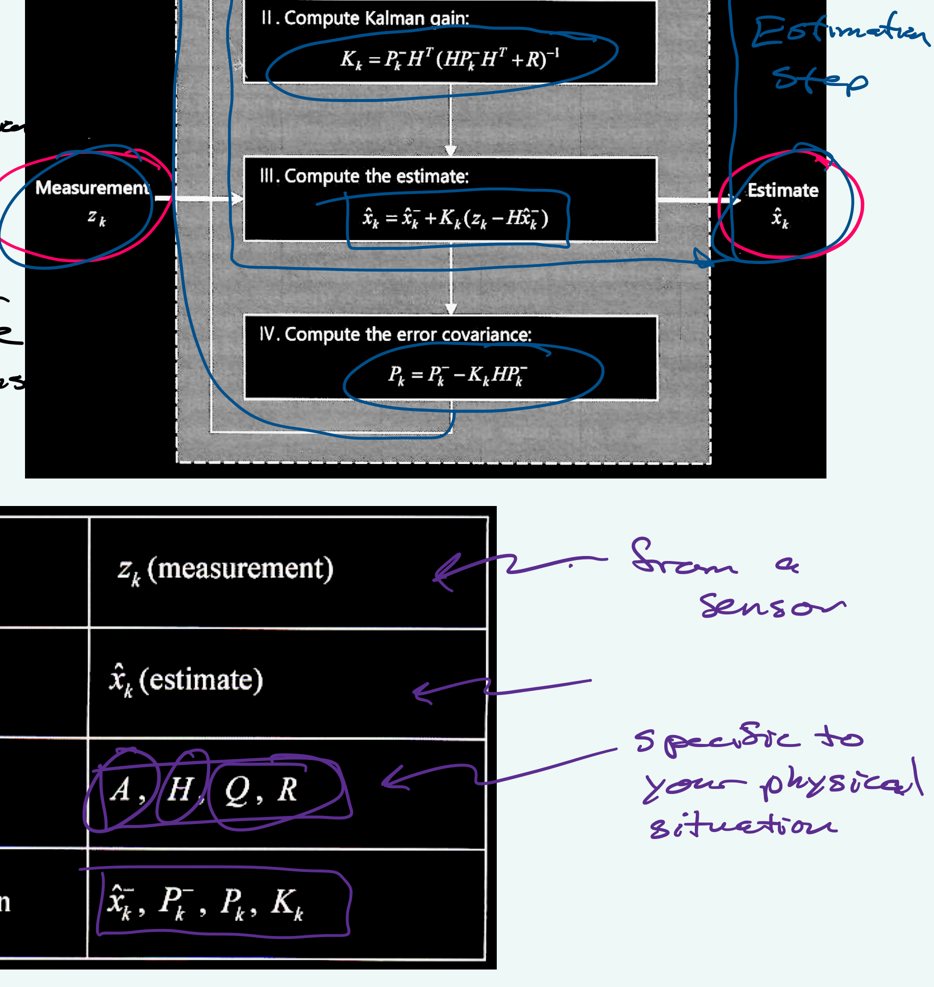
e.g., z_k = sonar data (altitude)

$$\hat{x}_k = \begin{bmatrix} \text{altitude} \\ \text{velocity} \end{bmatrix}$$

"–" means Prediction

$\hat{x}_k^-$ = predicted state

P_k^- = predicted error covariance (measure of noise)



External input	z_k (measurement)
Final output	$\hat{x}_k$ (estimate)
System model	A, H, Q, R
For internal computation	$\hat{x}_k^-, P_k^-, P_k, K_k$

from a sensor

specific to your physical situation

Go into more detail next time