1 Kalman Filter

1.1 Write a Kalman Filter.

In the file `linkalman.py`, you find a basic interface for a linear Kalman filter class. Its methods produce semantically correct but zero output. Implement the class to realize the Kalman filter that was presented in the lecture. The file contains matrix definitions for a simple one-dimensional problem ($z \sim N(20, 0.01)$ and a method to generate simple linear plot from the results. Test it and play around with the parameters!

1.2 Kalman Filter - Constant Velocity Model.

Use your implementation from section 1.1 to implement an 1D constant velocity model, i.e. a model that follows

$$\mathbf{x} = \begin{bmatrix} d \\ v \end{bmatrix}, \mathbf{x}_{t+1} = A\mathbf{x}_t = \begin{bmatrix} d_{t-1} + \delta_t v_{t-1} \\ v_{t-1} \end{bmatrix}$$

where d is the distance to our sensor, v is the velocity the object moves at, and $\delta_t = 1$ is the time step.

Make the following assumptions:

- Implement a measurement model that can only measure a (noisy) distance $\hat{d}$.
- Initialize the model with an initial guess of $\hat{d} = 90, \hat{v} = 0$ for $\mathbf{x}$.
- Assume $R = 5$ for measurement noise and $Q = \begin{bmatrix} 1 & 0 \\ 0 & 10 \end{bmatrix}$ for model uncertainty.

Run an experiment where you start with a ground truth distance of $d = 100$ and a ground truth velocity of $v = 1$, without adding any measurement noise, for 50 steps, and plot the results.

1.3 Kalman Filter - Constant Acceleration Model.

Copy and extend your implementation from section 1.2 to implement an 1D constant acceleration model, i.e. a model that follows

$$\mathbf{x} = \begin{bmatrix} d \\ v \\ a \end{bmatrix}, \mathbf{x}_{t+1} = A\mathbf{x}_t = \begin{bmatrix} d_{t-1} + \delta_t v_{t-1} + \frac{1}{2}\delta_t^2 a_{t-1} \\ v_{t-1} + \delta_t a_{t-1} \\ a_{t-1} \end{bmatrix}$$

Make the following assumptions:

- Implement once again a measurement model that can only measure a noisy distance $\hat{d}$.
- Initialize the model with zero distance, zero velocity, and zero acceleration.
- Assume $R = 10$ for measurement noise.
- Assume $Q = \begin{bmatrix} 0.01 & 0 & 0 \\ 0 & 10 & 0 \\ 0 & 0 & 1 \end{bmatrix}$ for model uncertainty.

Run an experiment where you start with a ground truth distance of $d = 100$, a ground truth velocity of $v = 1$, and a ground truth acceleration of $a = 0.1$. Run an experiment for 1000 steps, supplying $\hat{d}$ by adding noise that follows $w \sim N(0, R)$, and plot the results.

2 Kalman Faces

2.1 Kalman Filter - Face Position Filtering.

Use your implementation from section 1.1 to improve the face detection algorithm. Implement a very simple filter with a state model of $x_t = x_{t-1}, y_t = yt - 1, z_t = z_{t-1}$. You know you can measure x, y, z anyway (use the solution from programming exercise 2). Only fill the methods `createKalman()` and `stepKalman()` to create and step the Kalman filter with your input. Observe the results and play with your assumptions for measurement and model noise.

Can you extend your filter to use x, y, z velocities? How does that improve the result?