Exercise Round 4

The deadline of this exercise round is **Thursday February 6, 2025**. The solutions will be discussed during the exercise session in the T2 lecture hall of Computer Science building starting at 14:15.

The problems should be *solved before the exercise session*. During the session those who have completed the exercises will be asked to present their solutions on the board/screen.

Exercise 1. (Extended Kalman Filter)

Consider the following non-linear state space model:

$$\begin{aligned} x_k &= x_{k-1} - 0.01 \, \sin(x_{k-1}) + q_{k-1}, \\ y_k &= 0.5 \, \sin(2 \, x_k) + r_k, \end{aligned} \tag{1}$$

where q_{k-1} has a variance of 0.01^2 and r_k has a variance of 0.02. Derive the required derivatives for an EKF and implement the EKF for the model. Simulate trajectories from the model, compute the RMSE values, and plot the result.

(Note that Exercise 1 does not come with a template, so feel free to use whatever framework you want.)

Exercise 2. (Single Iteration of IEKF)

Show that the iterated extended Kalman filter with a single iteration is equivalent to the first order (non-iterated) extended Kalman filter.

Exercise 3. (Bearings Only Target Tracking with EKF)

In this exercise we consider a classical bearings-only target tracking problem that frequently arises in the context of passive sensor tracking. In this problem there is single target in the scene, and two angular sensors are used for tracking it. The scenario is illustrated in Figure 1.

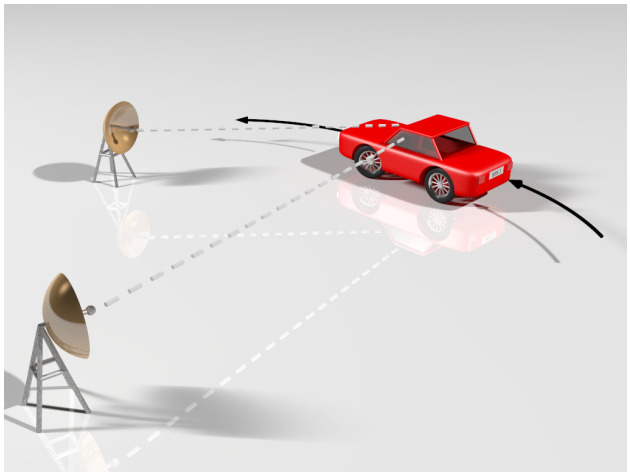

Figure 1: In the bearings-only target tracking problem, the sensors generate angle measurements of the target, and the purpose is to determine the target trajectory.

The state of the target at time step k consists of the position (x_k, y_k) and the velocity $(\dot{x}_k, \dot{y}_k)$. The dynamics of the state vector $\mathbf{x}_k = (x_k \ y_k \ \dot{x}_k \ \dot{y}_k)^\mathsf{T}$ are modeled with the discretized Wiener velocity model:

$$\begin{pmatrix} x_k \\ y_k \\ \dot{x}_k \\ \dot{y}_k \end{pmatrix} = \begin{pmatrix} 1 & 0 & \Delta t & 0 \\ 0 & 1 & 0 & \Delta t \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} x_{k-1} \\ y_{k-1} \\ \dot{x}_{k-1} \\ \dot{y}_{k-1} \end{pmatrix} + \mathbf{q}_{k-1},$$

where $\mathbf{q}_k$ is a zero mean Gaussian process noise with covariance

$$\mathbf{Q} = \begin{pmatrix} q_1^{\mathrm{c}} \, \Delta t^3/3 & 0 & q_1^{\mathrm{c}} \, \Delta t^2/2 & 0 \\ 0 & q_2^{\mathrm{c}} \, \Delta t^3/3 & 0 & q_2^{\mathrm{c}} \, \Delta t^2/2 \\ q_1^{\mathrm{c}} \, \Delta t^2/2 & 0 & q_1^{\mathrm{c}} \, \Delta t & 0 \\ 0 & q_2^{\mathrm{c}} \, \Delta t^2/2 & 0 & q_2^{\mathrm{c}} \, \Delta t \end{pmatrix}.$$

In this scenario the diffusion coefficients are $q_1^{\mathrm{c}} = q_2^{\mathrm{c}} = 0.1$, and the sampling period is $\Delta t = 0.1$. The measurement model for sensor $i \in \{1, 2\}$ is:

$$\theta_k^i = \tan^{-1}\left(\frac{y_k - s_y^i}{x_k - s_x^i}\right) + r_k^i, \tag{2}$$

where (s_x^i, s_y^i) is the position of the sensor i, and $r_k^i \sim \mathrm{N}(0, \sigma^2)$ is a Gaussian measurement noise with a standard deviation of $\sigma = 0.05$ radians. At each sampling time, which occurs 10 times per second (i.e., $\Delta t = 0.1$), both of the two sensors produce a measurement.

In the file `exercise7_2.m` (MATLAB) or `Exercise7_2.ipynb` (Python) of the book's companion code repository[1] there is a baseline solution, which computes estimates of the position from the crossing of the measurements and estimates the velocity to be always zero. Your task is to implement an EKF for the problem and compare the results graphically and in the RMSE sense.

(a) Implement an EKF for the bearings-only target tracking problem, which uses the non-linear measurement model (2) as its measurement model function (not the crossings). *Hints:*

- Use the function `atan2` in the measurement model instead of `atan` to directly get an answer in the range $[-\pi, \pi]$.
- The two measurements at each measurement time can be processed one at a time, that is, you can simply perform two scalar updates instead of a single two-dimensional measurement update.
- Start by computing the Jacobian matrix of the measurement model function with respect to the state components. Before implementing the filter, check by finite differences or automatic differentiation that the Jacobian matrix is correct.

(b) Compute the RMSE values, and plot figures of the estimates.

[1] https://github.com/EEA-sensors/Bayesian-Filtering-and-Smoothing