Kalam Filter Assignment

Introduction:

In this task, you are required to implement a Kalman Filter state estimator to localize a mobile robot moving in a 2D world. The robot is assumed to move directly in x or y directions without any need to change orientation. Thus, robot pose is: (x, y) and robot state is: $X = [x, y, \dot{x}, \dot{y}]^T$ (i.e. we omitted robot orientation so that robot state can be described by a linear model). The robot is controlled through two control variables: the 1st is the acceleration in x-direction and the 2nd is the acceleration in y-direction. A positioning system is used to measure robot x and y position at each time step. Due to model uncertainty and sensor errors, system dynamics suffers from additive noise to motion and measurement models. The whole dynamics of the system will be as follows:

$$X_k = F * X_{k-1} + B * u_{k-1} + w\,, \quad where \quad u{:}\, control\ action \quad w \sim N(0, Q)$$

$$Z_k = H * X_k + v\,, \quad where\ \ v \sim N(0, R)$$

System matrices are known. However we need to estimate the covariance matrices Q, and R of the additive noise. Every time step a control action is applied to the system to change robot state and a measurement of that state is taken to determine robot position. Kalman Filter is used to integrate between the erroneous state predictions and noisy measurements so that we get a good estimate for robot positions over time.

Requirements:

In this assignment, you are required to exactly perform four tasks:

1. Determine model uncertainty (i.e. estimate the values of Q matrix).
2. Determine sensor noise (i.e. estimate the values of R matrix).
3. Initialize the state covariance matrix P (Diagonal matrix, start with small values).
4. Complete the supported code with the kalman filter 3 steps.

To complete the requirements, you will use Kalman_Loop.m script where you will modify the values of matrices and write your code in the determined sections. Also three helper functions are supported to be used for estimating Q and R and for update step in kalman loop.

Note: Assume that Q and R matrices are diagonal for simplicity.

Note: You can use the functions but not to see their implementation as this is the case in a real world scenario where you deal with a system that you can't model with 100% accuracy.

Model Uncertainty

To model uncertainty we assume that it is Gaussian noise, and estimate its covariance matrix Q. The matlab function: motion_model.p is called for this purpose. It receives two parameters: dt and u where dt is the sampling time (assumed to be 0.05s) and u is a sequence of control actions to be applied to the initial state. The function then returns the final state of the robot.

$$X = motion\_model(0.05, u\_motion)$$

By calling this function several times (choose appropriate number) with the same known control sequence you get different final states due to motion uncertainty. Your task is to determine the variance in each state element and fill the covariance matrix Q appropriately.

Note: You aren't required to generate a control sequence. The u_motion.mat file contains a one for you. Just load it before calling the function.

Note: By inspecting u_motion, you are supposed to determine the ***ideal*** final state of the system if there is no noise.

Sensor Noise

To model sensor noise, we assume that it is Gaussian, and estimate its covariance matrix R. The matlab function: measurement_model.p is called for this purpose. It doesn't receive any parameter. It only returns a noisy measurement of the robot zero position.

$$Z = measurement\_model()$$

By calling this function several times (choose appropriate number) you get different measurements for the same state. Your task is to determine the variance in each element and fill the covariance matrix R appropriately.

State Estimation

You are given the control sequence u that will move the robot and your task is to continuously estimate robot state over time.

Note: Each column in u array represents a control action at a certain time step.

In Kalman Filter section of Kalman_Loop script, you will implement the three steps of kalman filter. One of these steps requires a real measurement after each time you apply the control action. So every iteration, you feed the real system with the same control action you use to update the state of the ideal model, and take a real measurement of the robot new position. You simply call the given real_system.p function for this purpose.

Z = real_system(X_init, dt, u(:,i));

Note: ground truth is supported to compare between ideal path of the robot (if the given control sequence is applied to it) and your estimated path.