16-642: Manipulation Estimation and Control
Fall 2022
Problem Set 3

due 11:59pm 16 November 2022

Please show all work and write clearly—all intermediate steps must be shown and legible. **Submit solutions as a zip file that contains written explanations of problem solutions, relevant MATLAB code, and requested plots and movies.**

You are encouraged to work with other students to find the answers to these problems, however each student must submit his/her own *unique* set of solutions. If you do work with other students, please give them credit by listing their names at the end of your solutions.

1. In this problem, you will implement an extended Kalman filter pose estimate for the differential drive robot. The configuration of the robot can can be represented by the vector

$$q = \begin{bmatrix} x_r \\ y_r \\ \theta \end{bmatrix}$$

where (x_r, y_r) is the 2D position in meters of the point midway between the two wheels, and θ is the angle in radians of the body relative to the x-axis. The input vector is

$$u = \begin{bmatrix} v_f \\ \omega \end{bmatrix},$$

where v_f is the forward speed of the body in meters per second, and ω angular velocity of the body in radians per second. The kinematic discrete time equation of motion is

$$q[k+1] = \begin{bmatrix} q_1[k] + T(u_1[k] + v_1[k]) \cos q_3[k] \\ q_2[k] + T(u_1[k] + v_1[k]) \sin q_3[k] \\ q_3[k] + T(u_2[k] + v_2[k]) \end{bmatrix}, \tag{1}$$

where $T = 0.01$ seconds is the rate at which the input is applied, and $v[k]$ is assumed to zero-mean white Gaussian noise with constant covariance matrix $V \in \mathbb{R}^{2\times 2}$.

The robot receives GPS measurements once every 10 timesteps, *i.e.*, the first y corresponds to $x[10]$, the second y corresponds to $x[20]$, and so on. The output equation is

$$y[k] = \begin{bmatrix} q_1[k] \\ q_2[k] \end{bmatrix} + w[k],$$

where $w[k]$ is assumed to zero-mean white Gaussian noise with constant covariance matrix $W \in \mathbb{R}^{2\times 2}$.

(a) (10 points) Find the linearized approximation of the system as a function of the current state. In particular, your approximation should look like

$$q[k+1] \approx F(q[k], u[k])q[k] + G(q[k])u[k] + \Gamma(q[k])v[k],$$

where $F(q[k], u[k])$ is a 3×3 matrix, and $G(q[k])$ and $\Gamma(q[k])$ are both 3×2.

(b) (20 points) The file `calibrate.mat` contains data generated during a run where an expensive ground truth system was available that could measure the full state of the robot. This data can be used to estimate the noise parameters, namely the noise on the input and on the GPS. The variables in calibrate.mat are:

- `t_groundtruth`: a vector of times associated with the input signals.
- `q_groundtruth`: a matrix whose columns are the robot states at the times associated the times in `t_groundtruth`. These states are assumed to be perfect.
- `u`: a matrix whose columns are the robot inputs at the times associated with `t_groundtruth`. This signal is the signal applied to the robot, it does *not* include the noise terms in $v[k]$.
- `t_y`: a vector of times associated with the GPS measurement.
- `y`: a matrix whose columns are the noisy GPS measurements taken at the times in `t_y`.

Using this information, estimate the process covariance V and measurement covariance W. You may want to start with W since computing it is the simpler of the two. Your solution should include an explanation of how you did the computations, your matlab code, and the computed values of V and W.

(c) The file `kfData.mat` contains another data set that you can use to demonstrate the effectiveness of a Kalman filter. The variables it contains are

- `t`: a vector of times associated with the input signals.
- `u`: a matrix whose columns are the robot inputs at the times associated with `t`. This signal is the signal applied to the robot, it does *not* include the noise terms in $v[k]$.
- `t_y`: a vector of times associated with the GPS measurement.
- `y`: a matrix whose columns are the noisy GPS measurements taken at the times in `t_y`.
- `q_groundtruth`: a matrix whose columns are the robot states at the times associated the times in `t`. These states are assumed to be perfect. Note that this should not be used to help you solve the problem, it is meant for comparison only.

Use an initial estimate

$$\hat{q}[1] = \begin{bmatrix} 0.355 \\ -1.590 \\ 0.682 \end{bmatrix}$$

and assume that the initial covariance matrix is

$$P[1 \mid 1] = \begin{bmatrix} 25 & 0 & 0 \\ 0 & 25 & 0 \\ 0 & 0 & 0.154 \end{bmatrix}$$

i. (5 points) Write a dead reckoner, i.e., make an extended Kalman filter that ignores the GPS signal and only does prediction steps. Plot y_r vs. x_r for your dead reckoner on the same plot with y_r vs. x_r for the ground truth. And submit your code.

ii. (25 points) Write a full extended Kalman filter. Submit your code, and make a plot that contains the following:

- a trajectory plot of y_r vs. x_r for the ground truth.
- a scatter plot or y_r vs. x_r from the GPS measurements.

- a trajectory plot of y_r vs. x_r for your EKF.

iii. (5 points) Rerun your EKF, only this time, scale W down by a factor of 100. Generate all of the same plots. Explain why the difference between this plot and the prior one makes sense.

2. (35 points) The objective of this problem is to write a particle filter that estimates the pose of a differential drive robot using measurements from two range beacons. The motion model is the same as for the Kalman filter problem (Equation 1), though the timestep T may be different. $v[k]$ is zero-mean Gaussian with unknown covariance. At every timestep, the robot receives noisy range measurements to two fixed beacons that are located at

$$b_1 = \begin{bmatrix} x_{b1} \\ y_{b1} \end{bmatrix} = \begin{bmatrix} 5 \\ 5 \end{bmatrix} \qquad \text{and} \qquad b_2 = \begin{bmatrix} x_{b2} \\ y_{b2} \end{bmatrix} = \begin{bmatrix} 15 \\ 5 \end{bmatrix}.$$

The measurement model is then given by

$$y[k] = \begin{bmatrix} \sqrt{(q_1[k] - x_{b1})^2 + (q_2[k] - y_{b1})^2} \\ \sqrt{(q_1[k] - x_{b2})^2 + (q_2[k] - y_{b2})^2} \end{bmatrix} + w[k],$$

where $w[k]$ is zero-mean Gaussian with unknown covariance.

The file `pfData.mat` contains another data set that you can use to demonstrate your particle filter. The variables it contains are

- `t`: a vector of times associated with the input signals.
- `u`: a matrix whose columns are the robot inputs at the times associated with `t`. This signal is the signal applied to the robot, it does *not* include the noise terms in $v[k]$.
- `y`: a matrix whose columns are the noisy range measurements taken at every timestep. $y_1[k]$ is the range to b_1 at time k, $y_2[k]$ is the range to b_2.
- `q_groundtruth`: a matrix whose columns are the robot states at the times associated the times in `t`. These states are assumed to be perfect. Note that this should not be used to help you solve the problem, it is meant for comparison only.

A template you can use to structure your particle filter along with some useful functions for visualizing the particle cloud and robot location is in `pfTemplate.m`. Your job is to modify this code to add a particle filter visualize the results at every timestep. Your particle filter should have a constant number of particles, and it should resample the particles at every step. Your particle filter will have several parameters which you will need to tune to get reasonable performance, in particular:

- number of particles in the cloud
- covariance of particle filter process noise.
- covariance of particle filter measurement noise.

At each timestep, you should make a plot that contains the following:

- Beacon locations
- Robot ground truth pose
- Robot ground truth trajectory
- Particle cloud
- Robot pose estimate (derived from particle cloud)

You should submit a write-up describing what choices you made in creating the filter, your code, and an `.mp4` movie showing the plots created at every timestep.