Comp 141 Probabilistic Robotics Homework 1: Kalman Filter

Daniel Choate

To generate the results displayed below, run <main.py> in python 3. This requires <HW1_utils.py> along with numpy and matplotlib libraries.

1 Kalman Filter: Prediction

A balloon drone has encountered a glitch in its program and needs to reboot its on-board computer. While rebooting, the drone is helpless and cannot issue motor commands. To help the drone, you'll need some understanding of the Kalman filter algorithm.

The drone operates in a 1-D world where x_t is the position at time t, while $\dot{x}_t$ and $\ddot{x}_t$ are the velocity and acceleration. For simplicity, assume that $\triangle t = 1$.

Due to random wind fluctuations, at each new time step, your acceleration is set randomly accordingly to the distribution $\mathcal{N}(\mu_{wind}, \sigma^2_{wind})$, where $\mu_{wind} = 0.0$ and $\sigma^2_{wind} = 1.0$.

Question 1.1: what is the minimal state vector for the Kalman filter so that the resulting system is Markovian?

The Markov assumption states that the past and future data are independent if one knows the current state. The minimal state vector for the Kalman filter consists of only the position x_t and the velocity $\dot{x}_t$

Thus, the state vector at time t can be represented by $X_t = \begin{bmatrix} x_t \\ \dot{x}_t \end{bmatrix}$

Question 1.2: Design the state transition probability function $p(x_t|u_t, x_{t-1})$. The transition function should contain linear matrices A and B and a noise covariance R.

The state transition probability can be written as $p(x_t|u_t, x_{t-1}) = A_t x_{t-1} + B_t u_t + R$.

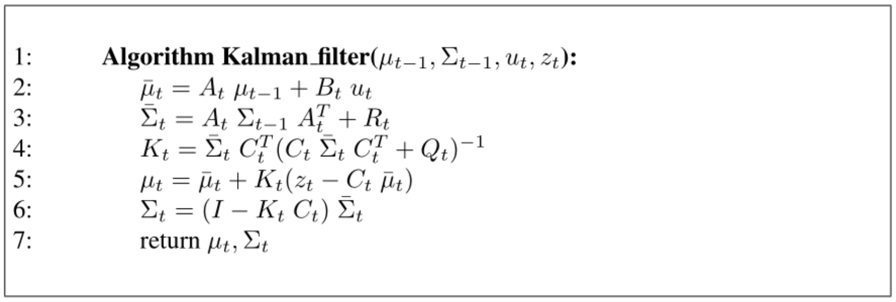
Using physics kinematics equations, and assuming $\triangle t = 1$, we can calculate the transition matrices A, B and the covariance R to be

$$A = \begin{bmatrix} 1 & 1 \\ 0 & 1 \end{bmatrix},\ B = \begin{bmatrix} 1/2 \\ 1 \end{bmatrix},\ R = \sigma^2_{wind} \begin{bmatrix} 1/4 & 1/2 \\ 1/2 & 1 \end{bmatrix}.$$

Thus, the state transition probability is

$$p(x_t|u_t, x_{t-1}) = \begin{bmatrix} 1 & 1 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} x_{t-1} \\ \dot{x}_{t-1} \end{bmatrix} + \begin{bmatrix} 1/2 \\ 1 \end{bmatrix} \begin{bmatrix} u_t \end{bmatrix} + \begin{bmatrix} 1/4 & 1/2 \\ 1/2 & 1 \end{bmatrix}$$

Question 1.3: Implement the state prediction step of the Kalman filter, assuming that at time $t = 0$, we start at rest, i.e., $x_t = \dot{x}_t = \ddot{x}_t = 0.0$. Use your code to calculate the state distribution for times $t = 1, 2, \ldots, 5$.



```
1:  Algorithm Kalman_filter($\mu_{t-1}, \Sigma_{t-1}, u_t, z_t$):
2:      $\bar{\mu}_t = A_t\ \mu_{t-1} + B_t\ u_t$
3:      $\bar{\Sigma}_t = A_t\ \Sigma_{t-1}\ A_t^T + R_t$
4:      $K_t = \bar{\Sigma}_t\ C_t^T (C_t\ \bar{\Sigma}_t\ C_t^T + Q_t)^{-1}$
5:      $\mu_t = \bar{\mu}_t + K_t(z_t - C_t\ \bar{\mu}_t)$
6:      $\Sigma_t = (I - K_t\ C_t)\ \bar{\Sigma}_t$
7:      return $\mu_t, \Sigma_t$
```

The following results generated from the main.py file. Specifically, problem 1 results are within lines 1-65. NOTE: since the acceleration is randomized, results will differ, but the covariance shapes and trends will remain the same (becoming more horizontal).

Assuming that at $t = 0$, we start at rest, implementing the state prediction step (lines 2-3) of the Kalman filter, the state predictions from $t = 0$ to $t = 5$:

$$\begin{bmatrix} x_t \\ \dot{x}_t \end{bmatrix} = \begin{bmatrix} 0 & -0.3514 & -0.6792 & -0.2442 & 0.6405 & 2.148 \\ 0 & -0.7028 & 0.0471 & 0.8229 & 0.9465 & 2.069 \end{bmatrix}$$

For the posterior over x and $\dot{x}$:

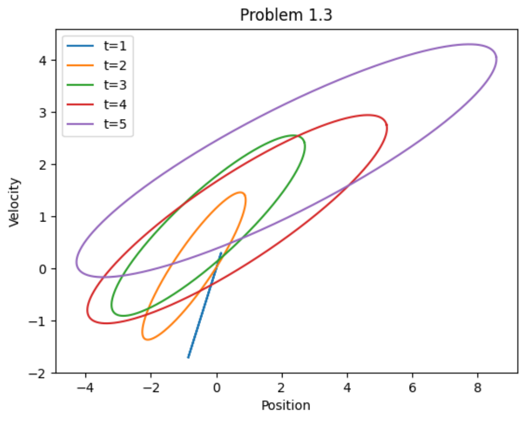
$$\begin{bmatrix} 0 & 0 & 0.25 & 0.5 & 2.5 & 2 & 8.75 & 4.5 & 21 & 8 & 41.25 & 12.5 \\ 0 & 0 & 0.5 & 1 & 2 & 2 & 4.5 & 3 & 8 & 4 & 12.5 & 5 \end{bmatrix}$$

For the random accelerations from $t = 0$ to $t = 5$: $u_t =$

$$\begin{bmatrix} 0 & -0.70282 & 0.74994 & 0.77582 & 0.12362 & 1.12287 \end{bmatrix}$$

Question 1.4: For each value of t in the previous question, plot the joint posterior over x and $\dot{x}$ in a diagram where x is the horizontal and $\dot{x}$ is the vertical axis. For each posterior, you are asked to plot the uncertainty ellipse which is the ellipse of points that are one standard deviation away from the mean.

Below shows a plot of the uncertainty ellipses for times $t = 1, 2, ..., 5$ using the random wind acceleration $\mathcal{N}(\mu_{wind}, \sigma^2_{wind})$ as the control input u_t.



2 Kalman Filter: Measurement

Prediction alone will result in greater and greater uncertainty as time goes on. Fortunately, your drone has a GPS sensor, which in expectation, measures the true position. However, the measurement is corrupted by Gaussian noise with covariance $\sigma^2_{gps} = 8.0$.

Question 2.1: Define the measurement model. You will need to define matrices C and Q.

For the measurement model, the measurement z_t can be written as
$z_t = C_t x_t + \delta_t$

Since the GPS only measures position,
$C = \begin{bmatrix} 1 & 0 \end{bmatrix}$ and $\delta_t = Q = \sigma^2_{GPS} = \begin{bmatrix} 8.0 \end{bmatrix}$

Question 2.2: Implement the measurement update. Suppose at time $t = 5$, the drone's computer has rebooted and we query our sensor for the first time to obtain the measurement $z = 10$. State the parameters of the Gaussian estimate before and after incorporating the measurement. Afterwards, implement the sensor modal to randomly sample the true position, corrupted with noise σ^2_{gps}.

The results calculated below are calculated within lines 70-137 in main.py. NOTE: since the acceleration is random, the results may vary.

For the random accelerations from $t = 0$ to $t = 5$: $u_t =$

$$\begin{bmatrix} 0 & 0.4047 & 0.7904 & -0.4986 & -0.8553 & 0.6687 \end{bmatrix}$$

At time $t = 5$, the predicted state $\begin{bmatrix} x_t \\ \dot{x}_t \end{bmatrix} = \begin{bmatrix} 2.3926 \\ 0.5098 \end{bmatrix}$

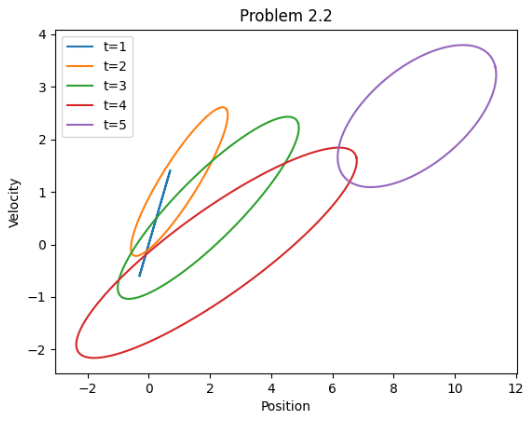
Predicted covariance $\bar{\Sigma} = \begin{bmatrix} 41.25 & 12.5 \\ 12.5 & 5 \end{bmatrix}$

The sensor reading (given) $z_t = \begin{bmatrix} 10 \end{bmatrix}$

The updated state (lines 4-6 of the Kalman Filter) $\begin{bmatrix} x_t \\ \dot{x}_t \end{bmatrix} = \begin{bmatrix} 8.7643 \\ 2.4407 \end{bmatrix}$

Updated covariance $\Sigma = \begin{bmatrix} 6.701 & 2.030 \\ 2.030 & 1.827 \end{bmatrix}$

The plot shown below shows the plotted covariance ellipses at each time t. As evidenced by the values shown above, once the measurement z_t is incorporated at time $t = 5$, the covariance ellipse shrinks, due to our prediction having less uncertainty.

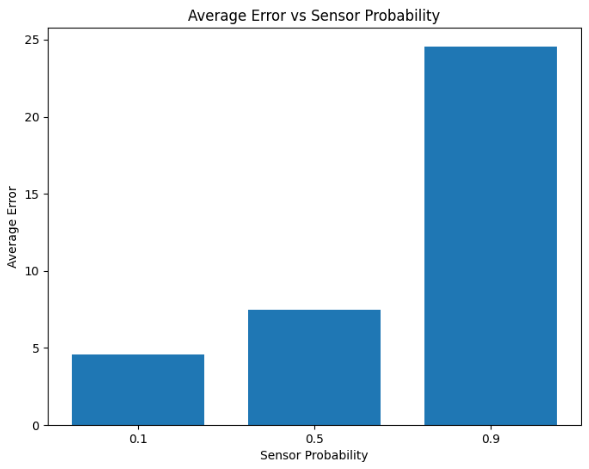


Question 2.3: All of a sudden, the sky gets cloudy which may cause the sensor to fail and not produce a measurement with probability $p_{gps-fail}$. For three different values of this probability (e.g., 0.1, 0.5, and 0.9), compute and plot the expected error from the true position at time $t = 20$. You may do so by running up to N simulations and use the observed errors to obtain the expected error empirically.

The results calculated below are calculated within lines 143-241 in main.py. NOTE: since the acceleration is random, the results may vary.

The plot shows the expected error from the true position at time $t = 20$ for $N = 100$ simulations.

$prob = 0.1,\ E_{expected} = 4.6$
$prob = 0.5,\ E_{expected} = 7.5$
$prob = 0.9,\ E_{expected} = 24.6$



3 Kalman Filter: Movement

Question 3.1 The drone is now fully operational and can not only take measurements, but also issue motor commands in the form of acceleration commands to its propeller. For example, a command of 1.0 will increase the drone's velocity by 1.0. Revisit Question 1.3 to provide the matrix B. If at time $t-1$, the drone's position and velocity are 5.0 and 1.0, compute the mean estimate for the state at time t given a motor command of 1.0. Your answer should be based on the constants provided but also include a random variable due to the wind effects. State the distribution of that random variable.

The matrix B in this case would be $B = \begin{bmatrix} 0 \\ 1 \end{bmatrix}$

The mean estimate for the state at time t given a motor command of 1.0 is calculated as follows (also shown in main.py lines 245-298).

$$\begin{bmatrix} x_t \\ \dot{x}_t \end{bmatrix} = \begin{bmatrix} 1 & 1 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} 5 \\ 1 \end{bmatrix} + \begin{bmatrix} 0 \\ 1 \end{bmatrix} \begin{bmatrix} 1 \end{bmatrix} = \begin{bmatrix} 6 \\ 2 \end{bmatrix};\ \bar{\Sigma} = \begin{bmatrix} .25 & .5 \\ .5 & 1 \end{bmatrix}$$