Homework 4- Lab Instructions

Kalman Filtering

Sara Farahani

In this lab, we implement the Kalman filter for tracking an object moving on a 2D plane. First, we make predictions based on the equations of motion. Then, we implement the Kalman filter. To evaluate it, we begin by considering constant velocity. Finally, we add constant acceleration to the state variables, and compare the results.

Note: You are allowed to use Python and the NumPy library. Using any other packages is prohibited.

Display Trajectories

In this lab, we track an object for 3 different trajectories. Run the below code in the trajectory.py file to generate and plot different trajectories.

```
def get_gt_trajectory(trajectory_type, dt, num_steps, *, radius=20,
angular_velocity=np.pi/4, angular_acceleration=0.05, b=0.2):

      if trajectory_type == "circular":

         t = np.linspace(0, num_steps*dt, num_steps)

         true_x = radius * np.cos(angular_velocity*t)

         true_y = radius * np.sin(angular_velocity*t)

         return np.column_stack((true_x, true_y))

      elif trajectory_type == "parabolic":

         t = np.linspace(-num_steps*dt, num_steps*dt, num_steps)

         true_x = 10 * t + t ** 2

         true_y = 20 * t - 4.9 * t ** 2
```

```
            return np.column_stack((true_x, true_y))
        elif trajectory_type == "spiral":
            t = np.linspace(0, num_steps*dt, num_steps)
            true_x = radius * np.cos(t+angular_acceleration*t**2)*np.exp(b*t)
            true_y = radius * np.sin(t+angular_acceleration*t**2)*np.exp(b*t)
            return np.column_stack((true_x, true_y))
        else:
            raise Exception("invalid trajectory type")
def get_measurements(trajectory_type, true_xy, noise_std):
    noise = np.random.normal(0, noise_std, true_xy.shape)
    measured_xy = true_xy + noise
    return measured_xy
def plot(true_xy, measured_xy):
   plt.figure(figsize=(8, 8))
   plt.plot(true_xy[:, 0], true_xy[:, 1], label="Ground Truth Trajectory")
   plt.plot(measured_xy[:, 0], measured_xy[:, 1], label="Measurements")
   plt.xlabel("X position")
   plt.ylabel("Y position")
   plt.title("Trajectory")
   plt.legend()
   plt.grid(True)
   plt.show()
if __name__ == "__main__":
    trajectory_types = ["circular", "parabolic", "spiral"]
```

```
measurement_noises = [1, 5, 5]

for i, trajectory_type in enumerate(trajectory_types):

  true_xy = get_gt_trajectory(trajectory_type, dt=0.1, num_steps=100)

  measured_xy=get_measurements(trajectory_type,true_xy,
  noise_std=measurement_noises[i])

  plot(true_xy, measured_xy)
```

Task 1: Equations of motion

In the first step, write a program and try to make predictions for the velocity and acceleration of each of the three given trajectories according to:

$$v_t = \frac{\Delta x}{\Delta t} = \frac{x_t - x_{t-1}}{\Delta t}$$

$$a_t = \frac{\Delta v}{\Delta t} = \frac{v_t - v_{t-1}}{\Delta t}$$

Notice that you have to do this using the available observation (noisy location) and not the ground truth. Plot each of the two coordinates of the estimated velocity and acceleration (a total of 4 plots).

Task 2: Implement the Kalman Filter

To implement the Kalman filter, complete the `KalmanFilter` class in the file kalman_filter.py. Complete the `predict` function using the following prediction equations of the Kalman filter:

$$X_t = A X_{t-1} + \epsilon_X$$

$$\Sigma'_t = \Sigma_X + A \Sigma_{t-1} A^T$$

Where:

- X_t is the state at time t
- X_{t-1} is the state at time t-1
- A is the state transition model
- Σ'_t is the covariance matrix of the state estimate at time t
- Σ_X is the covariance matrix of the process noise
- Σ_{t-1} is the covariance matrix of the state estimate at time t-1

- ϵ_X is the process noise, which is drawn from Gaussian distribution $N(0, \Sigma_X)$

Then, complete the `update` function based on the correction equations of the Kalman filter:

$$K = \Sigma'_t B^T (\Sigma_O + B \Sigma'_t B^T)^{-1}$$
$$X_t = X'_t + K(O_t - B X'_t)$$
$$\Sigma_t = \Sigma'_t - K B \Sigma'_t$$

Where:

- B is the measurement model
- K is the Kalman gain
- O_t is the measurement at time t
- Σ_O is the covariance matrix of the measurement noise
- X_t is the updated state at time t
- X'_t is the predicted state at time t
- Σ_t is the covariance matrix of the state estimate at time t
- Σ'_t is the predicted covariance matrix of the state estimate at time t

```
class KalmanFilter():

    def __init__(self, A, Σ_X, B, Σ_O, X_0, Σ'_t):

        # TODO: Define & initialize the matrix A

        # TODO: Define & initialize the matrix Σ_X

        # TODO: Define & initialize the matrix B

        # TODO: Define & initialize the matrix Σ_O

        # TODO: Define the initial state estimate vector X

        # TODO: Define the initial state covariance matrix Σ'_t

    def predict(self):
```

```
    # TODO: Predict X_t based on the prediction formula

    # TODO: Predict Σ'_t based on the prediction formula

  def update(self, measured_xy):

    # TODO: Update K based on the correction formula

    # TODO: Update X_t based on the correction formula

    # TODO: Update Σ_t based on the correction formula
```

Up to this point, you have implemented the equations. But to run the filter you need to define the transition and observation models that are the matrices **A** and **B**. In the next steps, you will define these matrices and create instances of this `KalmanFilter` class for evaluation.

Task 3: Evaluate for Constant Velocity

In this part, we assume that the velocity is constant. Note that since the object is assumed to move with constant velocity in R^2, the state variable is $X = [x, y, \dot{x}, \dot{y}]$, where x, y are the position variables, and $\dot{x}, \dot{y}$ are the velocity variables. Additionally, we know the equations of motion as follows:

$$x_t = x_{t-1} + v_{t-1}\Delta t + \epsilon_x$$
$$v_t = v_{t-1} + \epsilon_v$$

We know that in the Kalman filter, the transition model is:

$$X_t = A X_{t-1} + \epsilon_X$$

Where:

- X_t is the state at time t,
- X_{t-1} is the state at time t-1,
- A is the state transition model,
- ϵ_X is the process noise, which is drawn from $N(0, \Sigma_X)$, Σ_X is the covariance matrix of the process noise

Now, you should find the state transition matrix (A) by rewriting the above formulas in matrix representation. What will this matrix be?

Next, you need to find the observation matrix. The formula for observation is:

$$O_t = B X_t + \epsilon_o$$

Where:

- B is the observation model, and
- ϵ_O is the measurement noise, which is drawn from $N(0, \Sigma_O)$, Σ_O is the covariance matrix of the measurement noise.

Considering that we just measure the position, what will the matrix B be?

Now, you should:

- Write a program and use the `KalmanFilter` class to make predictions and corrections on different trajectories in the trajectory.py file.
- Plot the estimated positions, measured values and the ground truth trajectory.
- Compare the predicted velocity with the predicted values of task 1 by calculating and plotting the error.
- Try to make changes in the covariance matrix of the process noise (Σ_X), the covariance matrix of the measurement noise (Σ_O) and the initial state estimate(X_0) parameters. Explain their effects on the results.

Task 4: Evaluate for Constant Acceleration

Now, we add constant acceleration to the state variables. Repeat what you did in task 3. Consider that with the addition of acceleration, the state variable is now represented as $X = [x, y, \dot{x}, \dot{y}, \ddot{x}, \ddot{y}]$, where x, y are the position variables, $\dot{x}$, $\dot{y}$ are the velocity variables and $\ddot{x}$, $\ddot{y}$ are the acceleration variables. Additionally, we know the equations of motion as below:

$$x_t = x_{t-1} + v_{t-1}\Delta t + \frac{1}{2}a_{t-1}\Delta t^2$$
$$v_t = v_{t-1} + a_{t-1}\Delta t$$
$$a_t = a_{t-1}$$

Recall the prediction and observation formulas from the previous task. What will the state transition matrix (A) and the observation matrix (B) be here?

Now, like what you did in task 3:

- Write a program and use the `KalmanFilter` class to make predictions and corrections on different trajectories in the trajectory.py file.
- Plot the predicted positions, measured values, and the ground truth trajectory.
- Compare the predicted velocity and acceleration with the predicted values of task 1 by calculating and plotting the error.
- Try to make changes in the covariance matrix of the process noise (Σ_X), the covariance matrix of the measurement noise (Σ_O) and the initial state estimate(X_0) parameters. Explain their effects on the results.