CIE482: Path Planning & Navigation for Mobile Robots

Assignment 4

Instructions:

- This is a **graded** assignment of 6%.
- No typesetting is allowed, you must write your solutions.
- You can write on the pdf directly with a tablet & electronic pen **or** print the pdf and answer with an ink-pen.
- Good handwriting and neatness makes your work easier to read and understand and is highly recommended.
- Upload your solution as a **single .zip file** including the **MATLAB code, Screenshots of the GUI, and pdf** on Blackboard by **12:00 pm Tuesday 14 April 2026 (Before the Lecture).**
- Plagiarism will **not tolerated** at all.

Student Name:	
Student ID:	

Question 1:
For the following multiple-choice questions, choose **one** possible **correct answer** from the options provided.

1. What is the primary goal of **Bayesian filtering** in a robot localization context?
   A. To directly compute the next control input for the robot.
   B. To maintain a probabilistic estimate (belief) of the robot's state as it evolves over time.
   C. To eliminate sensor noise by ignoring it.
   D. To provide exact solutions for non-linear dynamic systems.

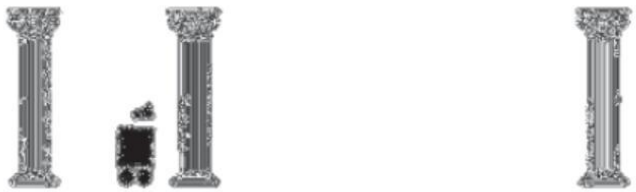
2. In the **prediction update** step of Bayesian filtering, we:
   A. Incorporate new sensor measurements to correct our state estimate.
   B. Integrate prior knowledge of control inputs and motion models to predict the next state.
   C. Scale the covariance matrix by zero to eliminate uncertainty.
   D. Skip any knowledge about the system dynamics.

3. The **correction update** (or measurement update) in Bayesian filtering:
   A. Updates the motion model.
   B. Incorporates sensor data to refine the predicted state.
   C. Computes the expected sensor reading from the state transition matrix.
   D. Eliminates the need for a measurement model.

4. Which of the following assumptions *must* be satisfied for a **Kalman Filter** to provide exact (optimal) solutions?
   A. A linear system with Gaussian noise in both process and measurements.
   B. A nonlinear system with uniform noise.
   C. No noise in either the system or the measurements.
   D. Infinite memory of all past measurements.

5. The **Kalman gain** K_k in the measurement update is computed to:
   A. Maximize the covariance after the measurement update.
   B. Balance between trusting the prediction and the new measurement.
   C. Always favor the measurement over the prediction.
   D. Remain constant at all times.

Question 2:



A robot moves along a line. Its discrete-time model is

$$x_k = x_{k-1} + \Delta t\, u_k + \eta_k$$

and its position sensor is

$$z_k = x_k + w_k.$$

Use $\Delta t = 1$s and the initial condition $x_0 = 2.0$ m.

For each time step:

1. Compute the next true state x_k.
2. Compute the measured position z_k.
3. Compute the innovation $\beta_k = z_k - x_k$.

Data table to complete

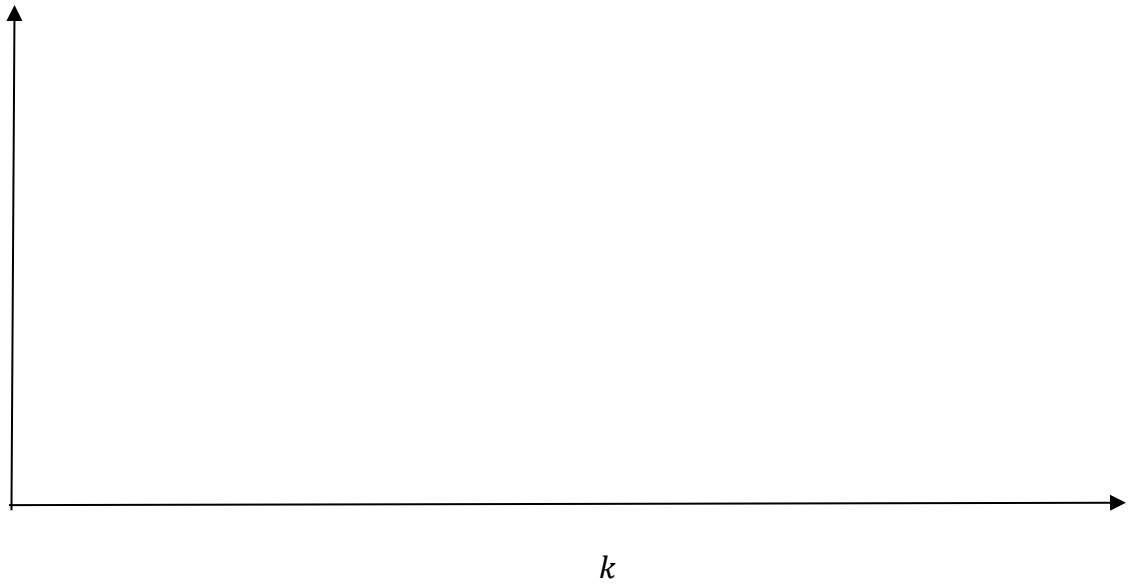
k	u_k (m/s)	η_k (m)	w_k (m)	x_k (m)	z_k (m)	β_k (m)
0	–	–	–	2.0	–	–
1	1.0	0.2	0.3			
2	1.5	-0.1	-0.2			
3	1.0	0.0	0.1			
4	0.5	0.1	0.0			
5	1.0	-0.2	-0.1			

Then plot on the following graph:

- the **true position** x_k using ‘o’ marker,
- the **measured position** z_k using ‘x’ marker

for $k = 1, \dots, 5$

Position



Question 3: [Programming]

In this part, you will implement a 1D Kalman Filter to estimate the position of a robot moving along a single axis. We provide you with a MATLAB GUI script, **RobotLocalizationGUI.m**, that simulates the robot's motion, periodically generates noisy measurements, and visualizes your filter's performance in real-time.

Your primary task is to complete two key functions that the GUI calls each time the robot moves or receives a measurement:

kalman_predict.m (the prediction step)

kalman_update.m (the measurement update step)

Once you implement these functions correctly, running the GUI will allow you to move the robot with the arrow keys, see the noisy measurements, and watch in real-time as your Kalman Filter updates its position estimate. You will then run a series of experiments—adjusting parameters (e.g., noise levels, initial conditions, etc.)—to analyze how the filter behaves.

Below is a short list of **2 targeted experiments** you should conduct using the provided **RobotLocalizationGUI.m**. In each experiment, you will adjust the relevant parameters (initial estimate, noise variances, measurement frequency) and observe how the **Kalman Filter** behaves over time. Use the **sliders** in the GUI where applicable, or modify the script constants (var_u, var_z, z_Ts) directly in the code for more precise control. For each experiment, simply make qualitative observations of the filter's performance and write them in the boxes below:

Experiment a)

- **Goal:** Examine how the filter corrects itself from an intentionally incorrect initial belief.

- **Procedure:**
    1. Use the “Initial state” slider to set an initial estimate that is far from the true robot position.
    2. Use the “Initial Variance” slider to set a large initial uncertainty.
    3. Press the left / right arrows to move the robot several times, allowing the filter to predict and update.

- **Observation:**
    - Watch the Gaussian distribution in the middle axis shrink or shift as new measurements arrive.
    - Monitor how quickly (or slowly) the estimated position converges toward the real robot position on the right-axis plot.
    - Repeat with a very small initial variance (e.g., 0.01) and see if the filter becomes overly “confident” in its initial position before measurements arrive.
- **Write your observations below and include 2-3 screenshots of your GUI in your submission.**

Experiment b)

- **Goal:** Investigate whether taking measurements more often (but each being noisier) helps or hinders the filter's ability to track position.

- **Procedure:**
    1. Set the measurement interval z_Ts to a smaller value (e.g., z_Ts = 2), meaning measurements occur more frequently.
    2. Increase measurement noise variance var_z (e.g., to 0.5 or 1.0) to represent that each measurement is less reliable.
    3. Press the left / right arrows to move the robot several times, allowing the filter to predict and update.

- **Observation:**
    - Compare the filter's estimate against the true trajectory in the right-axis plot.
    - Does receiving frequent but noisy measurements lead to a stable estimate, or does it introduce more variability into the filter's belief?
    - Try different noise levels (moderately high vs. extremely high) at the same increased measurement frequency.
- **Write your observations below and include 2-3 screenshots of your GUI in your submission.**

Appendix

How to use the GUI:

1. **Top Axis (Robot Visualization)**
   - Shows the robot as a rectangle with two wheels on a 1D track.
   - Whenever you press the arrow keys, the robot moves left or right (bounded between 0 and 10).
2. **Middle Axis (Gaussian Distribution)**
   - Plots the current estimated probability distribution over position.
   - It should be centered around your current state estimate $x_{k|k}$ with a spread proportional to your covariance $P_{k|k}$.
3. **Right Axis (Position History)**
   - Shows a step-by-step history of:
     - True position (blue)
     - Estimated position (red)
     - Measured position (green)
   - You can press **"Clear History"** to reset this plot.
4. **Sliders**
   - **Robot Position**: Manually set the true robot position (for debugging or initial placement).
   - **Initial State**: Manually set the initial guess for the robot's position.
   - **Initial Variance**: Set the initial covariance for your state estimate.
5. **Keyboard Control**
   - **Left Arrow** moves the robot to the left.
   - **Right Arrow** moves the robot to the right.
   - Each key press triggers a **predict** step in your Kalman Filter, followed by an **update** step at certain intervals (whenever a measurement is generated).