Robot Learning
Exercise 1

November 9th, 2023
TAs: Gabriele Tiboni (*gabriele.tiboni@polito.it*), Andrea Protopapa (*andrea.protopapa@polito.it*), Elena Di Felice (*s303499@studenti.polito.it*)

1 Introduction

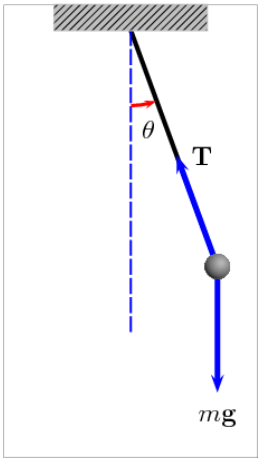
Throughout this exercise session we will familiarize ourselves with the notorious filtering problem in engineering, which revolves around estimating the state of a stochastic system due to noisy observations. In particular, we will deal with a simple unactuated pendulum system, with noisy measurements of its angle. Our primary goal is to implement an Extended Kalman Filter to estimate the true angle and angular velocity of the system as the latter one evolves over time.

As you follow the steps in this document, make sure you're able to answer **all** the *guiding questions* listed at the end of each step. You will be asked to **return a report** of the following exercise session containing the code to complete all the steps, and the requested plots.

To proceed with the assignment, visit the dedicated starting code at the dedicated webpage and follow the steps listed below in this document. Note: if you haven't joined the course's github classroom yet, you will be asked to select your name when you visit it the first time.

2 Unactuated pendulum system

Consider a simple pendulum system, which is free to oscillate given its initial conditions. No input control (i.e. no actuation) is considered in our system, hence the pendulum is solely affected by gravity. The state of the system may be known by 2 state variables, representing the angle and the angular velocity. Given the length of the pendulum and the acceleration of gravity we can fully compute the dynamics of the system. In particular:



$$x_1 = \theta, \quad x_2 = \dot{\theta}$$

$$\dot{\mathbf{x}} = \begin{bmatrix} \dot{x}_1 \\ \dot{x}_2 \end{bmatrix} = \begin{bmatrix} x_2 \\ -\frac{g}{l}\sin(x_1) \end{bmatrix}$$

The latter equation depicts the state-space representation of the system, as a system of two first-order differential equations. For the sake of this assignment, a discrete-time dynamical system must be derived by considering fixed timestep intervals. We can make use of the forward Euler method to approximate continuous dynamics in a discrete domain:

$$\boldsymbol{\beta}(\mathbf{x}) = \begin{bmatrix} x_2 \\ -\frac{g}{l}\sin(x_1) \end{bmatrix}$$

$$\mathbf{x}_k \approx \mathbf{x}_{k-1} + \Delta t \cdot \boldsymbol{\beta}(\mathbf{x}_{k-1})$$

Finally, we consider a sensor that outputs noisy measurements of the angle theta. We fix the noise variance of the measurement model at *0.05*. This should represent the true variance of the sensor which is typically unknown to the user. Note that the system dynamics are not considered to be stochastic, and the only source of stochasticity in the collected data is induced by the measurement model.

Guiding questions:

- Can you derive the Jacobian of the discrete system dynamics w.r.t. the state variables?
- Can you derive the measurement model? How about its Jacobian w.r.t. the state variables?

3 Implement an Extended Kalman Filter in python

Implement an Extended Kalman Filter in python, by simulating the pendulum system previously described. Make use of the provided starting code here. Note that the filter should estimate both the angle and the angular velocity of the system, despite only (noisy) measurements of the angle are given.
Once implemented, plot the true and the estimated state variables side-by-side (2 plots for the angle and angular velocity respectively).

As the true variance of the measurement model is typically unknown, vary the measurement noise covariance “R” used by the EKF, and observe how the estimation results change in the two plots. How can you explain the different behavior?

1. Algorithm **Kalman_filter**($\mu_{t-1}, \Sigma_{t-1}, u_t, z_t$):

2. Prediction:
3. $\mu_t = A_t \mu_{t-1} + B_t u_t$
4. $\bar{\Sigma}_t = A_t \Sigma_{t-1} A_t^T + Q_t$

5. Correction:
6. $K_t = \bar{\Sigma}_t C_t^T (C_t \bar{\Sigma}_t C_t^T + R_t)^{-1}$
7. $\mu_t = \bar{\mu}_t + K_t (z_t - C_t \bar{\mu}_t)$
8. $\Sigma_t = (I - K_t C_t) \bar{\Sigma}_t$

9. Return μ_t, Σ_t

4 ROS integration

Recall the ROS essentials that you studied in the previous assignment, and refactor the EKF implemented in the previous step into a ROS-based application. You're free to choose how to structure your application (e.g. how many nodes to create, name of the topics to communicate on). The goal of this step is to simulate a real-world example, where the different components of the system are communicating over ROS topics.

Hint: for example, you may create a 3-node application: a "pendulum" node which simulates the true physics of the real-world system and outputs the current true state variables, a "sensor" node which returns a noisy measurement of the true state variables, and an "EKF" node which estimates the state of the system by listening to the measurements provided by the "sensor" node.

Guiding questions:

- Can you visualize the true, noisy, and estimated time series of the state variables with rqt_multiplot?

5 Extras

Consider now a Double pendulum system instead. Try modifying your code implementation of the EKF in step 3 to reflect the new dynamics of the double pendulum, and the 4 state variables instead of 2.