Robot Learning Homework 1

Matteo Zulian s310384
Automation and Intelligent Cyber-Physical Systems
Politecnico di Torino

1 Introduction

In order to determine whether or not a system is performing properly, and ultimately to control the system performance, the behaviour of the system at any given instant of time must be known, or in other words, the states of the system must be known. To accomplish that, a measuring device or devices called sensors, which collect measurements (observations) of the system are required, but these sensory devices are normally corrupted by noise by the electronic and mechanical components associated with the measuring device, also every sensor increases the cost of creation and maintenance of the system. The problem of determining the states of a system from noisy observations is called state estimation and it is of great importance in engineering since the states are not always directly accessible. So with the available states, it may be possible to reconstruct the missing states using filters.

In this exercise we were asked to create a Kalman Filter to estimate the state of a simple unactuated pendulum and to study how its parameters can change the correctness of the estimate. We were also asked to integrate our EKF in ROS and to extend the solution to a more complex system.

2 Design of EKF

The pendulum is a nonlinear system so the Extended Kalman Filter can be used, even if it doesn't guarantee an optimal solution as the KF, which is used for linear system. The EKF is a Bayes filter that uses a recursive approach to estimate the state, where every iteration is divided in two steps, which exploit the properties of Gaussian distributions to compute the result.

- prediction step

$$\textbf{State Prediction:} \quad \hat{\mathbf{x}}_{pk} = \mathbf{A}_k \hat{\mathbf{x}}_{k-1} + \mathbf{B}_k \mathbf{u}_k$$
$$\textbf{Covariance Prediction:} \quad \mathbf{\Sigma}_{pk} = \mathbf{A}_k \mathbf{\Sigma}_{k-1} \mathbf{A}_k^T + \mathbf{Q}_k$$

- update step

$$\textbf{Kalman gain:} \quad K = \Sigma_{pk} C^T (C \Sigma_{pk} C^T + R)^{-1}$$
$$\textbf{Updated state estimate:} \quad \hat{x}_k = \hat{x}_{pk} + K(z_k - C\hat{x}_{pk})$$
$$\textbf{Updated error covariance:} \quad \Sigma_k = (I - KC)\Sigma_{pk}$$

In this exercise it's analysed the free response of the pendulum, so $u_k = 0$ for every k.
The intuitive idea behind the filter is that at each iteration it predicts the state $\hat{x}_p$, measures the observable states with sensors z and then computes the difference between the two to see how far is the current estimation $\hat{x}$ from the real state x, to be able to correct it. This correction allows to well estimate both the observable and the hidden states, in this case, respectably θ and $\dot{\theta}$ and it is one of

7th Conference on Robot Learning (CoRL 2023), Atlanta, USA.

the strength of the Kalman Filter, together with a quite good resistance to noisy measurements. The parameters chosen for the Kalman filter are very important as seen in section 3, and they are:

$$Q = \begin{bmatrix} 0.00001 & 0 \\ 0 & 0.00001 \end{bmatrix} \quad R = 0.05 \quad \Sigma_0 = \begin{bmatrix} 10 & 0 \\ 0 & 10 \end{bmatrix}$$

2.1 discrete domain

The equations of the pendulum in continuous time are :

$$x(t) = \begin{bmatrix} x_1(t) \\ x_2(t) \end{bmatrix} = \begin{bmatrix} \theta(t) \\ \dot{\theta}(t) \end{bmatrix} \tag{1}$$

$$\dot{x}(t) = \begin{bmatrix} f_1(t) \\ f_2(t) \end{bmatrix} = \begin{bmatrix} x_2(t) \\ -\frac{g}{L}\sin(x_1(t)) \end{bmatrix} = \begin{bmatrix} \dot{\theta}(t) \\ -\frac{g}{L}\sin(\theta(t)) \end{bmatrix} \tag{2}$$

In this solution was used the discrete time version of the EKF, so the Euler forward method can be applied to approximate the continuous dynamics in a discrete domain with time-step $\Delta t = 0.01$:

$$x_k = f(x_{k-1}) = x_{k-1} + \Delta t \frac{dx}{dt} \tag{3}$$

obtaining :

$$x_k = f(x_{k-1}) = \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} + \Delta t \begin{bmatrix} x_2 \\ -\frac{g}{L}\sin(x_1) \end{bmatrix} = \begin{bmatrix} x_1 + \Delta t x_2 \\ x_2 - \Delta t \frac{g}{L}\sin(x_1) \end{bmatrix} \tag{4}$$

2.2 Jacobian of states

The jacombian matrix J_f (called A previously) can be found by computing the derivative of the discrete time equations (2), obtaining :

$$J_f = \begin{bmatrix} \frac{\partial f_1}{\partial x_1} & \frac{\partial f_1}{\partial x_2} \\ \frac{\partial f_2}{\partial x_1} & \frac{\partial f_2}{\partial x_2} \end{bmatrix} = \begin{bmatrix} 1 & \Delta t \\ -\Delta t \frac{g}{L}\cos(x_1) & 1 \end{bmatrix} \tag{5}$$

2.3 Jacobian of measurements

The sensor used in this system only measure the angle of the pendulum so the measurements vector is defined as $z(t) = \theta(t) + \eta$, where η is a random noise. So the output function is $h = x_1$ from which J_h (called C previously) can be computed as:

$$J_h = \begin{bmatrix} \frac{\partial h_1}{\partial x_1} & \frac{\partial h_1}{\partial x_2} \end{bmatrix} = [1 \quad 0] \tag{6}$$

2.4 Result

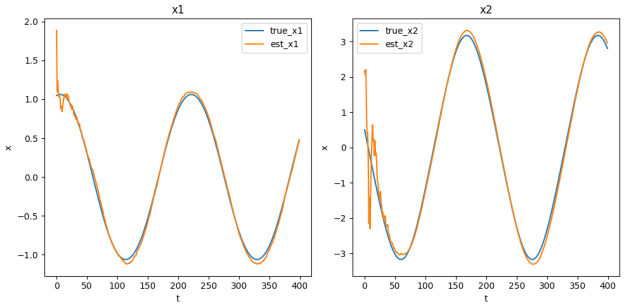

Figure 1: state estimate pendulum

3 The measurement noise covariance “R”

The measurement noise covariance matrix R is an important hyper parameter of the kalman filter because it is used in the computation of the Kalman Gain K which is the weight of the correction made in the update step. This means that an higher R will result in an higher K and this changes the behaviour of the filter. An higher R will lead to a faster estimation but with a loss in precision while a lower R will do the opposite and the limit cases where it’s too small or too big will lead to imprecise or wrong estimations.

3.1 Results

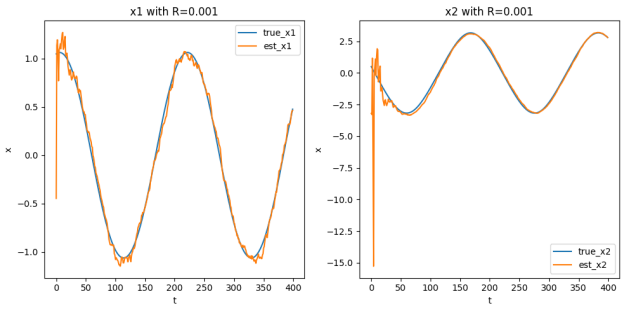

Figure 2: R = 0.001

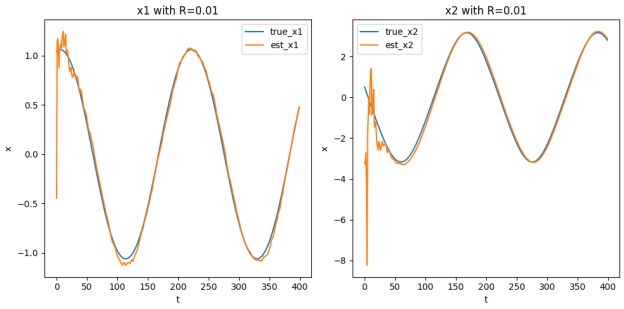

Figure 3: R = 0.01

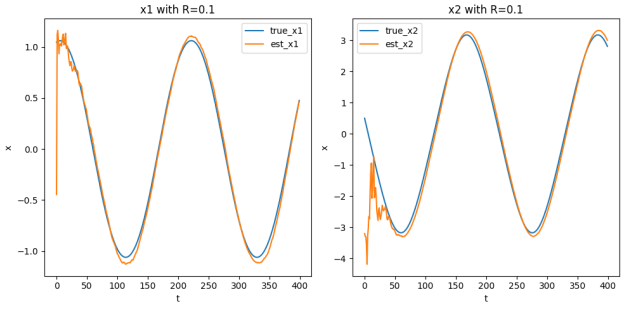

Figure 4: R = 0.1

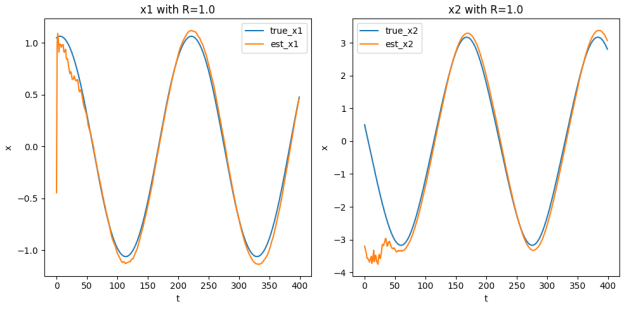

Figure 5: R = 1

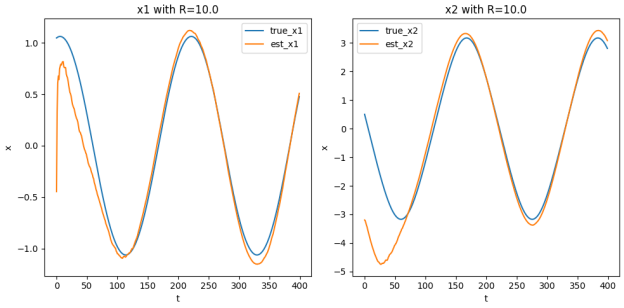

Figure 6: R = 10

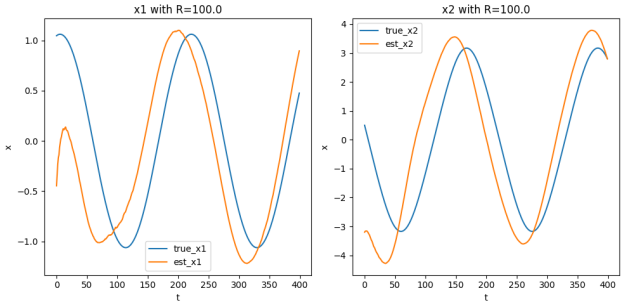

Figure 7: R = 100

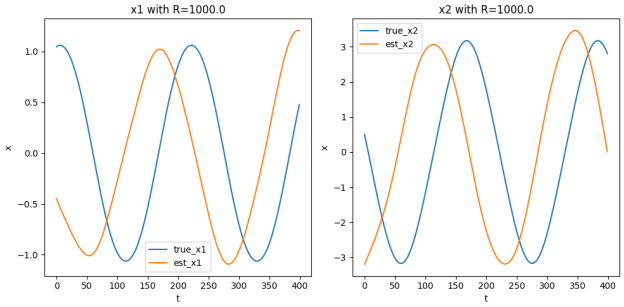

Figure 8: R = 1000

4 ROS integration

ROS is a software that allows to run concurrently different processes, called nodes, which can exchange messages and informations through topics. In order to run this system on ROS, was created a pendulum node which computes the true state over time and sends it, through a topic, to a sensor node, which corrupts it with some noise and sends it to the kalman filter node which estimates the true states.

Figure 9: nodes and topics

The nodes were created inside a package as python scripts and started by a launcher on command line. ROS has its plot application called rqt_multiplot, that was used to plot the state variables to verify the correctness of the solution.

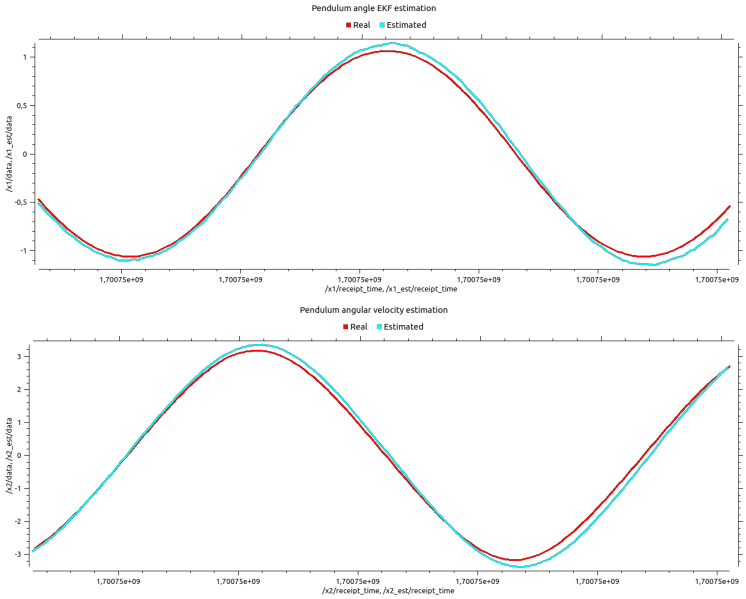

Figure 10: rqt multiplot pendulum

On ROS is also possible to make 3D simulations using rviz. Firstly was created an rviz-object with markers, similar to a pendulum: a line with a sphere. Then the true state of the pendulum and its estimation were translated into coordinates in the space and used to create the simulation.

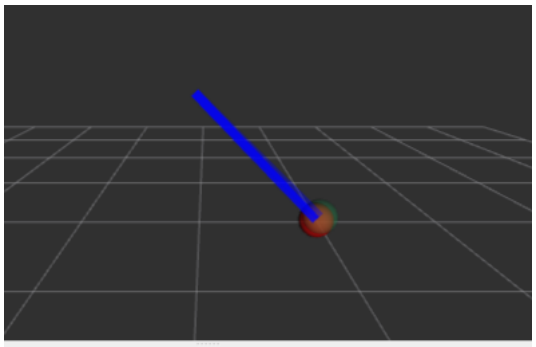

Figure 11: pendulum simulation

5 Extra: double pendulum

5.1 state equations and Jacobians

A more complex system like a double pendulum can be estimated by the EKF with the right modification on the previous solution. Firstly the state variables and the system equation are defined as:

$$x = \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{bmatrix} = \begin{bmatrix} \theta_1 \\ \omega_1 \\ \theta_2 \\ \omega_2 \end{bmatrix} \tag{7}$$

$$\dot{\mathbf{x}} = \begin{bmatrix} \dot{x_1} \\ \dot{x_2} \\ \dot{x_3} \\ \dot{x_4} \end{bmatrix} = \begin{bmatrix} \dot{\theta}_1 \\ \dot{\omega}_1 \\ \dot{\theta}_2 \\ \dot{\omega}_2 \end{bmatrix} = \begin{bmatrix} \omega_1 \\ \frac{-g(2m_1+m_2)\sin\theta_1 - m_2 g\sin(\theta_1-2\theta_2) - 2\sin(\theta_1-\theta_2)m_2(\omega_2^2 L_2+\omega_1^2 L_1\cos(\theta_1-\theta_2))}{L_1(2m_1+m_2-m_2\cos(2(\theta_1-\theta_2)))} \\ \omega_2 \\ \frac{2\sin(\theta_1-\theta_2)(\omega_1^2 L_1(m_1+m_2)+g(m_1+m_2)\cos\theta_1+\omega_2^2 L_2 m_2\cos(\theta_1-\theta_2))}{L_2(2m_1+m_2-m_2\cos(2(\theta_1-\theta_2)))} \end{bmatrix} \tag{8}$$

similarly as before to compute the Jacobians the system has to be discretized with Euler (3), obtaining:

$$J_f = \begin{bmatrix} \frac{\partial f_1}{\partial x_1} & \frac{\partial f_1}{\partial x_2} & \frac{\partial f_1}{\partial x_3} & \frac{\partial f_1}{\partial x_4} \\ \frac{\partial f_2}{\partial x_1} & \frac{\partial f_2}{\partial x_2} & \frac{\partial f_2}{\partial x_3} & \frac{\partial f_2}{\partial x_4} \\ \frac{\partial f_3}{\partial x_1} & \frac{\partial f_3}{\partial x_2} & \frac{\partial f_3}{\partial x_3} & \frac{\partial f_3}{\partial x_4} \\ \frac{\partial f_4}{\partial x_1} & \frac{\partial f_4}{\partial x_2} & \frac{\partial f_4}{\partial x_3} & \frac{\partial f_4}{\partial x_4} \end{bmatrix} = \begin{bmatrix} 1 & \Delta t & 0 & 0 \\ \Delta t\frac{\partial \dot{x_2}}{\partial x_1} & 1+\Delta t\frac{\partial \dot{x_2}}{\partial x_2} & \Delta t\frac{\partial \dot{x_2}}{\partial x_3} & \Delta t\frac{\partial \dot{x_2}}{\partial x_4} \\ 0 & 0 & 1 & \Delta t \\ \Delta t\frac{\partial \dot{x_4}}{\partial x_1} & \Delta t\frac{\partial \dot{x_4}}{\partial x_2} & \Delta t\frac{\partial \dot{x_4}}{\partial x_3} & 1+\Delta t\frac{\partial \dot{x_4}}{\partial x_4} \end{bmatrix} \tag{9}$$

In order to well estimate the angular velocity of the second angle w_2, an extra sensor must be add to the system to measure also the angle θ_2

$$z(t) = \begin{bmatrix} \theta_1(t)+\eta \\ \theta_2(t)+\eta \end{bmatrix} \quad h = \begin{bmatrix} x1 \\ x3 \end{bmatrix} \tag{10}$$

obtaining :

$$J_h = \begin{bmatrix} \frac{\partial h_1}{\partial x_1} & \frac{\partial h_1}{\partial x_2} & \frac{\partial h_1}{\partial x_3} & \frac{\partial h_1}{\partial x_4} \\ \frac{\partial h_2}{\partial x_1} & \frac{\partial h_2}{\partial x_2} & \frac{\partial h_2}{\partial x_3} & \frac{\partial h_2}{\partial x_4} \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \tag{11}$$

5.2 Results

Looking at figure (12) is possible to see the real pendulum (in red) and the estimated one (in green) and how at every time step the green one tracks the red one.

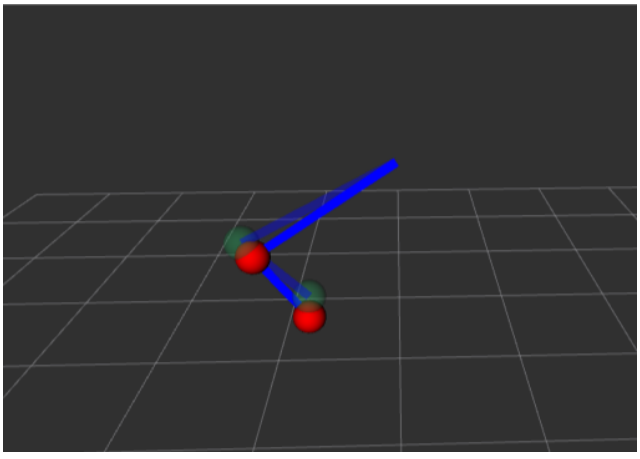

Figure 12: double pendulum simulation

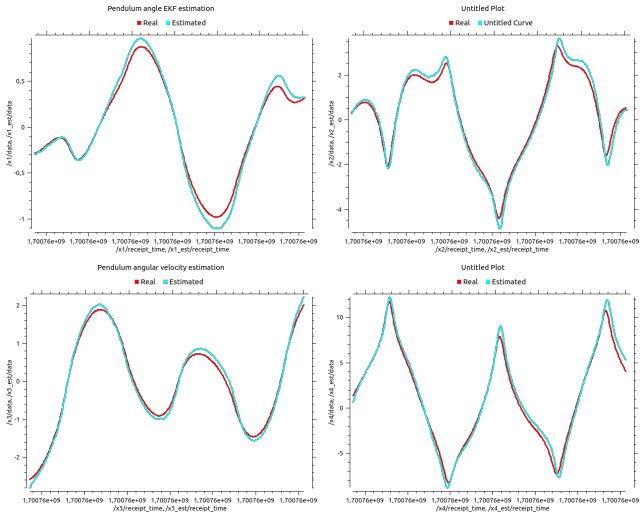

Figure 13: rqt double pendulum