ACS6124 Multisensor and Decision Systems
Part I – Multisensor Systems
Lab B: State Estimation with Kalman Filter Designs

Lecturer: Yuanbo Nie
email: y.nie@sheffield.ac.uk

2022/2023

Aims and Objectives

At the end of the lab, you will be able to

- Implement Kalman filter-based state estimation algorithms.
- Understand the need for extended and iterative extended Kalman filters for some nonlinear systems.

How to use MATLAB files

Download the files from ACS6124 Course Content at Blackboard at Labs/ACS6124 Part I - Multisensor Systems - Laboratory/LabBFiles.zip. Extract and put those files in your working directory and add the corresponding working path to MATLAB.

State Estimation with Kalman Filter

Consider the following system

$$\dot{x}(t) = -3\cos^3 x(t) + w(t), \tag{1}$$

with state $x \in \mathbb{R}$ and process noise $w \in \mathbb{R}$. It is known a prior that $\mathbb{E}\{w(t_k)\} = 0$, $\mathbb{E}\{w^2(t_k)\} = \sigma_w^2$ with $\sigma_w = 10$.

Task 1: Kalman Filter and Extended Kalman Filter

To start with, use the following observation and measurement model

$$y(t) = x(t), \tag{2a}$$
$$y_m(t_k) = y(t_k) + v(t_k), \quad k = 0, 1, 2, \ldots. \tag{2b}$$

with output $y \in \mathbb{R}$ and measurement noise $v \in \mathbb{R}$. We also know that $\mathbb{E}\{v(t_k)\} = 0$, $\mathbb{E}\{v^2(t_k)\} = \sigma_v^2$, with $\sigma_v = 1$.

Task 1.1: Use `template.m` to develop a Kalman filter (KF) script to estimate the state trajectory. All the data you will need are in `data_task1_1.mat`:

- `t` contains the time series and `z_k` stores all the measurements,
- The expected value for initial state $x(t_0)$ and noises $w(t_k)$ and $v(t_k)$ are given by `Ex_0`, `Ew` and `Ev`,
- The standard deviation for initial state $x(t_0)$ and noises $w(t_k)$ and $v(t_k)$ are given by `stdx_0`, `stdw` and `stdv`,
- The actual state trajectory is stored in `x_k`. *It is provided for comparison with your Kalman filter-generated estimations, hence should not be used as part of the algorithm.*

As a starting point, you can choose $x = 0$ as the linearization point.

Once satisfied with the result, consider the following nonlinear measurement model instead

$$y(t) = x^3(t), \tag{3a}$$
$$y_m(t_k) = y(t_k) + v(t_k), \quad k = 0, 1, 2, \ldots. \tag{3b}$$

where the corresponding data is stored in `data_task1_2.mat`.

Task 1.2: Update your script to obtain an estimation of the state trajectory with the new model and data. Try different linearization points to improve the estimation result.
Task 1.3: Make a copy of your KF codes and modify it to be an extended Kalman filter (EKF) script. Test this script with the same problem and observe the differences. Also try different ways of generating the simulation result for the prediction step, including the use of Euler as well as higher order methods.

Task 2: Iterated Extended Kalman Filter

Test the performance of your KF and EKF on `data_task2.mat`, which is obtained using the following measurement model

$$y(t) = \arctan(x(t)), \tag{4a}$$
$$y_m(t_k) = y(t_k) + v(t_k), \quad k = 0, 1, 2, \ldots. \tag{4b}$$

Note that this data uses a different setting for `Ex_0` and `stdv`.

Task 2: Make a copy of your EKF codes and modify it to be an iterated extended Kalman filter (IEKF) script. Does it perform differently from the EKF and why?