Homework

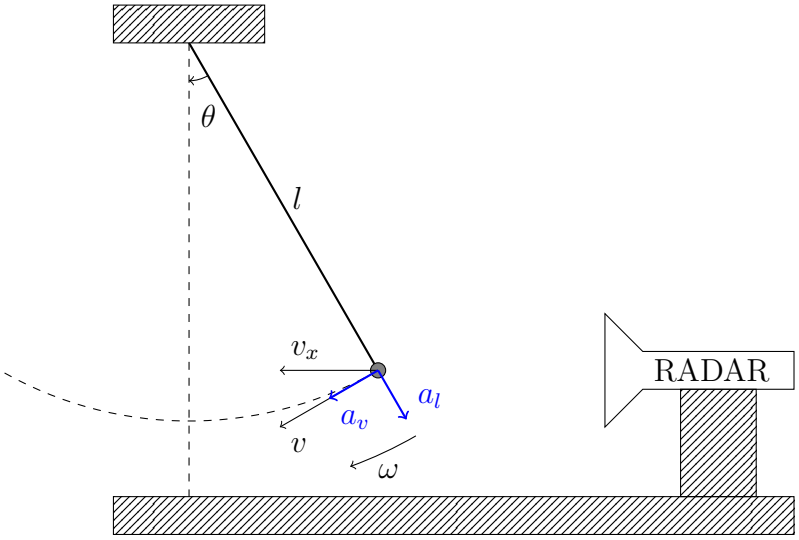
**** Deadline May 5, 10PM ****

version 1

1 Introduction

Consider the free-moving, frictionless, pendulum bellow. The goal of the device is to estimate the pendulum's motion as precisely as possible, and for that, two sensors have been installed:

- a radar capable of measuring the horizontal speed v_x (using Doppler shift) of the the weight of the pendulum.
- an accelerometer installed at the weight of the pendulum, measuring the acceleration along the wire a_l and the acceleration perpendicular to the wire a_v



For those of you that need to freshen up on the physics of the pendulum, I suggest an excellent (and fun) video:

Figure 1: prof. Walter Lewin presenting, in style, the physics of the pendulum

You will not need to solve the differential equations in a closed form, as you will calculate the motion numerically. In consequence you should not need to make the small angle approximation. A few useful relations:

$$v = \omega l \tag{1}$$
$$v_x = v \cos\theta \tag{2}$$
$$a_l = g\cos\theta - v^2/l \quad (g = 9.81 m/s^2 \text{ - the gravitational acceleration}) \tag{3}$$
$$a_v = g\sin\theta \quad \text{(careful with the sign here)} \tag{4}$$
$$\theta_k = \theta_{k-1} + \omega\Delta t \quad \text{(you can imagine a better aproximation by using the angular acceleration as well)} \tag{5}$$

2 Assignment

You will receive a time series with the data from the sensors. Your task is to write an algorithm and the corresponding computer code (Python/Octave) to calculate the position θ of the pendulum at all times, as precisely as possible. The algorithm needs to work online: to estimate θ_k, only use measurement up to and including the moment k. The measurements may be affected by systematic errors, that is, a constant added to all the values returned by the sensor.

The time series will contain:

Time (s)	$v_x(m/s)$, $\sigma^2_{v_x}$	$a_l(m^2/s)$, $\sigma^2_{a_l}$	$a_v(m^2/s)$, $\sigma^2_{a_v}$
0	1, 0.1	0, 0.1	0, 0.1
...			

You are encouraged to discuss the problem/solution extensively. The goal here is that everybody to understands the problem and the possible solutions. However everything you write (code or write-up) must be written by you, independently. Once you start writing the solution stop discussing it with your peers.

Please hand in:

1. A description of your algorithm (one page maximum). If you used any scientific articles for deciding on your algorithm please add them as references. If you discussed the problem/solution with some of your classmates, mention them.

2. A function (suggested in Python/Jupyter) which takes as arguments: the input sensor data time series, the approximate initial position θ and the approximate length of the pendulum l. The function should output: time series with the pendulum position estimate and the corresponding variance, the estimate length l of the pendulum, the estimated systematic error of the accelerometers and the radar (speed).

   You have a skeleton Python/Jupyter code to start with. If you use any other language you are on your own.

3 Grading

You will receive:

- 0.5 points for the description of the algorithm

- 1.0 points for any algorithm that does a reasonable job at using all the data available to estimate the motion of the pendulum
- 0.3 points for any algorithm that also estimates the length of the pendulum
- 0.2 points for any algorithm that does a reasonable job at calculating the covariance of the position/speed estimate
- 0.5 bonus points for any algorithm that estimates and uses the systematic error of the radar/accelerometer

You will receive points for any solution that improves the skeleton function provided, as long as you describe clearly how it works and for what types of sensors it works well, and under what conditions it fails. You have an input data generator. You can test your code under various conditions (variance, bias, etc).

4 Hints

- Chose your state variables and coordinate system carefully. Weigh the trade-offs between cartesian and polar coordinates.

5 Practical details

You have a Jupyter file which contains a configurable input data generator and an skeleton solution which uses only the radar data. You can run it in various places on the web (ex. Azure, Colab), or you can very easily install Jupyter on your computer (recommended). In the file you will find several sections of code which must be executed in order. Ctrl+ENTER is the shortcut which execute the current section of code (where the cursor is).

Modify the function and insert the description of the algorithm in the same file in the reserved cell

You may find the following tutorials useful:

- Jupyter - programming in a notebook
- Python - the basic language
- NumPy - matrix operations, particularly look at the linear algebra section