Problem set 3 for CS 498 GC

- This homework consists of a coding part only, which must be submitted individually on Gradescope. You only need to upload the python files *problem_set3_p1.py*, *problem_set3_p2.py*, and *problem_set3_p3.py* (see attached templates)

Total points: 100

Question 1 (20 points)
Your company executive team thinks that MEMS Inertial Measurement Sensors that cost $50 should be good enough for the robot your company is building to navigate accurately across the field. They remember from their dynamics class that position is the integral of velocity and velocity is the integral of acceleration. Your job is to convince them that the errors in an un-aided INS system will build up over time. Rummaging through your notes of the field robotics class, you decide to make your case with a very simple model of a one-dimensional point-mass system:

$$\dot{p}(t) = v(t)$$
$$\dot{v}(t) = a(t)$$

where *p(t), v(t), a(t)* are the positions, velocity, and acceleration of the one-dimensional robot.
You then make a model for the accelerometer measurements $\tilde{a}(t)$, you can assume the accelerometer updates at 100Hz (i.e. dt=0.01 sec).

$$\tilde{a}(t) = a(t) + b(t) + \zeta(t)$$

where *b(t)* is the bias of the accelerometer, and ζ(t) is the accelerometer noise.

You model the bias as a zero-mean Gaussian random walk process:

$$\dot{b} = \omega_b$$

Where $\omega_b \sim N(0, \sigma_b^2)$ and ${\sigma_b}^2 = 1.0e-6$ m^2/s^4

You model the accelerometer noise with a zero-mean Gaussian distribution:

$\zeta \sim N(0, \sigma_\zeta^2)$, where $\sigma_\zeta^2 = 2.5e-3$ m^2/s^4

Complete the functions *true_states*, *measurement_model* and *estimated_states* in problem_set3_p1.py.

The function *true_states* computes the position and velocity of the robot by integrating the velocity and acceleration. We define the true acceleration as a known function of time, which is supposed to be the actual robot's acceleration.
Since we do not have available measurements, we need to generate a *measurements model*, by adding noise and bias to the true acceleration.
Finally, we *estimate the state xhat* (position and velocity) of the robot by integrating the estimated velocity and the accelerometer measurements.

Question 2 (30 points)
You now argue to the CEO that you can fix this issue by adding a GPS based position measurement. You make a simple model of the GPS position measurement:

$$\tilde{p}(t) = p(t) + v_p(t)$$

Where $v_p \sim N(0, \sigma_p^2)$ and $\sigma_p^2 = 3$ m^2. In practice, the GPS measurements come at a lower rate than IMU measurements. In this exercise, we will assume the same rate for all the measurements.

You are going to implement a Kalman filter to estimate your states xhat (position, velocity, bias). The acceleration measurements will be used in the prediction state, and the GPS measurements in the correction step. The true states and the measurement model are given.
Complete the functions *estimated_states_prediction*() and *stimated_states_correction()* in *problem_set3_p2.py*.
You may find next slide helpful. These matrices are already defined in the template.

Our filter mechanization equations:

$$\begin{bmatrix} \dot{\hat{p}} \\ \dot{\hat{v}} \\ \dot{\hat{b}} \end{bmatrix} = \begin{bmatrix} 0 & 1 & 0 \\ 0 & 0 & -1 \\ 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} \hat{p} \\ \hat{v} \\ \hat{b} \end{bmatrix} + \begin{bmatrix} 0 & 0 \\ 1 & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} \tilde{a} \\ \omega_b \end{bmatrix} \quad (88)$$

Note that we are using the accelerometer measurements $\tilde{a}$ as a surrogate "control" input
Now we need our measurement equations, let's assume position is being measured, then

$$y = \begin{bmatrix} 1 & 0 & 0 \end{bmatrix} \begin{bmatrix} \hat{p} \\ \hat{v} \\ \hat{b} \end{bmatrix} \quad (89)$$

Now we have enough to do our Kalman Filtering, but will the filter converge? Will we get an optimal gain matrix L?

Question 3 (50 points)

You now have convinced the CEO that you need GPS, you want to have more budget for your project, so you set out to convince them you can integrate a gyro and an encoder which will actually enable the robot to know its position, velocity, and angle really well. You utilize the model below for a differential drive robot.

$$\dot{x} = v\,cos(\theta),$$
$$\dot{y} = v\sin(\theta)$$
$$\dot{\theta} = \omega$$
$$\dot{v} = a$$

Where now you have measurements of the angular rate termed as ω:

$$\tilde{\omega} = \omega + \bar{\zeta}_{\omega}$$

and the accelerometer measurements ã as:

$$\tilde{a}(t) = a(t) + b_a(t) + \zeta_a(t)$$

Assume that the accelerometer noise and bias characteristics are as in question 1, and the gyro noise is:

$$\zeta_{\omega} \sim N(0, \sigma^2), \text{and } \sigma_{\omega}^2 = 2.5 \times 10^{-3} rad^2/s^2$$

In addition you have position measurements x̃ and ỹ with the same noise characteristics as in Question 2.

You are going to implement an Extended Kalman filter to estimate the robot's state *xhat* (x, y, theta, v, bias). The accelerometer and gyroscope measurements will be used in the prediction step, and the GPS and encoder measurements in the correction step. The true states and the measurement model are given. In the prediction step, you can use the following equations ($\hat{o}$: state estimate, $\tilde{o}$: measurement)

$$\dot{\hat{x}} = \hat{v}cos(\hat{\theta})$$
$$\dot{\hat{y}} = \hat{v}sin(\hat{\theta})$$
$$\dot{\hat{\theta}} = \tilde{\omega}$$
$$\hat{\dot{v}} = \tilde{a} - \hat{b}$$
$$\dot{\hat{b}} = \omega_b$$

ω_b is the same as in question 1.

In the correction step, we need to linearize the model to obtain the transition matrix A. You may find next slide useful.

The states are now $[x, y, \theta, v, b]$ The linearized matrix is (before discretization)

$$A = \begin{bmatrix} 0 & 0 & -v\sin(\theta) & \cos(\theta) & 0 \\ 0 & 0 & v\cos(\theta) & \sin(\theta) & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{bmatrix} \quad (133)$$

For the prediction step, integrate the continuous nonlinear dynamics first
For the correction step use the measurement matrix:

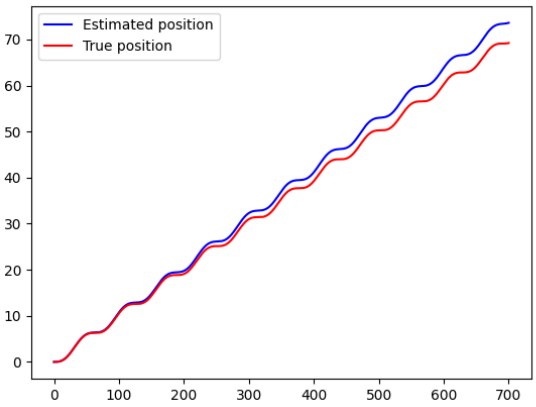
$$y = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} x \\ y \\ \theta \\ v \\ b \end{bmatrix} \quad (134)$$

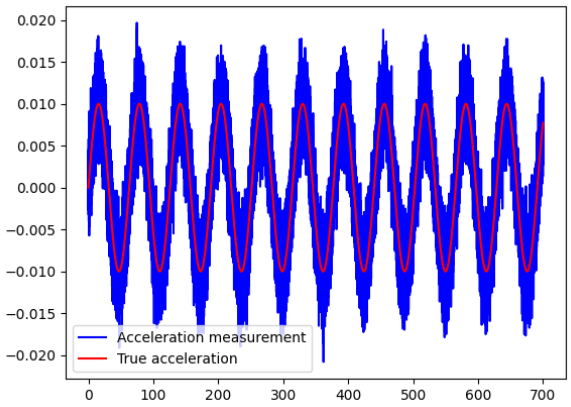
The third measurement is the encoder

Complete the functions *estimated_states_prediction()* and *stimated_states_correction()* in *problem_set3_p3.py*.

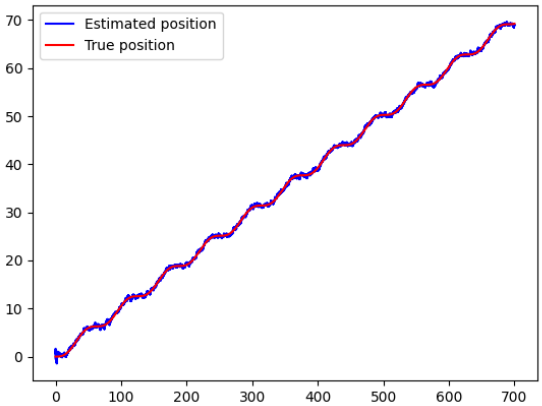
Sample outputs

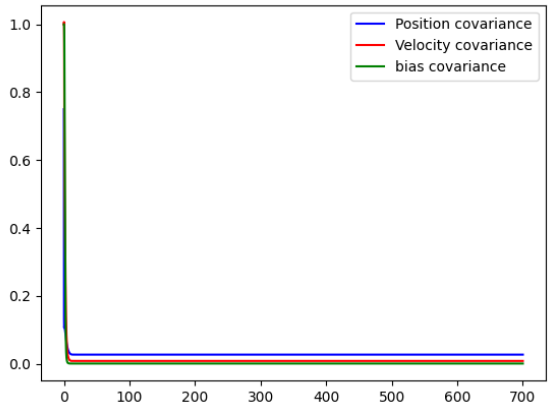
Part 1.





Part 2.





Part 3.