ESE 650, SPRING 2023

HOMEWORK 2

ANIRUDH KAILAJE [KAILAJE@SEAS.UPENN.EDU],
COLLABORATORS: DHRUV PARIKH [DHRUVKP@SEAS.UPENN.EDU], CHANDRAVARAN KUNJETI [KUNJETI@SEAS.UPENN.EDU]

Solution 1 (Time spent: 4 hours)**. EKF**
The dynamics of the state is as given below:

$$\begin{aligned} x_{k+1} &= ax_k + \epsilon_k \\ y_k &= \sqrt{x_k^2 + 1} + \gamma_k \end{aligned} \tag{1}$$

Where $\epsilon_k \sim N(0, 1)$, and $\gamma_k \sim N(0, 1/2)$ and inital state $x_0 \sim N(1, 2)$

1. SIMULATING SYSTEM

The system was simulated using the equations above. For noise sampling for γ_k and ϵ_k, I used the numpy function np.random.normal() inbuilt function where the parameters loc defines the mean, and the scale defines the variance. The simulation was run for a hundred time-steps for $a = -1$. x_k and the observation y_k is plotted below:

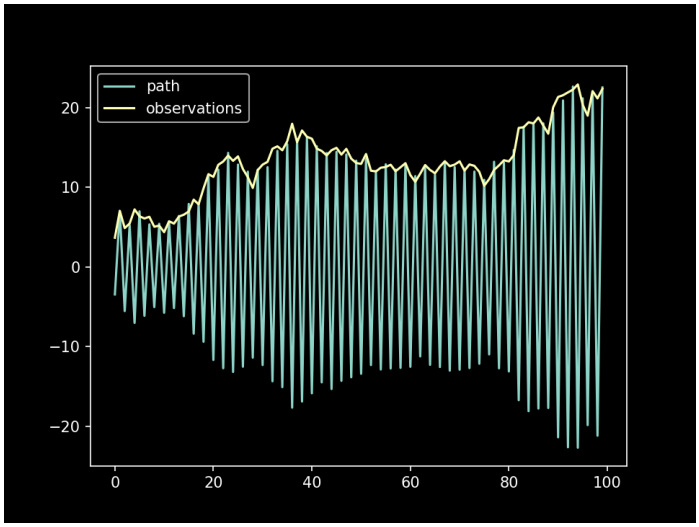

FIGURE 1. System Simulation

2. EKF TO ESTIMATE A

We need to estimate two quantities as defined,

$$\begin{aligned} \mu_k &= E[a_k|y_1, ...y_k] \\ \sigma_k^2 &= Var(a_k|y1, ...y_k) \end{aligned} \tag{2}$$

To do this, I defined the state such that it includes x_k and a. Therefore

$$X_k = \begin{bmatrix} x_k \\ a_k \end{bmatrix} \tag{3}$$

Given this definition of state, the state propagation can be written as:

$$X_{k+1} = \begin{bmatrix} a_k * x_k \\ a_k \end{bmatrix} \tag{4}$$

Because the state is a 2x1 matrix, the covariance Σ_k will be a 2x2 matrix.

The first step for the EKF implementation is the computation of $\mu_{k+1|k}$ and $\Sigma_{k+1|k}$ Thsi can be done by,

$$\begin{aligned} \mu_{k+1|k} &= f(\mu_{k|k}) = \begin{bmatrix} a_k * x_k \\ a_k \end{bmatrix} \\ \Sigma_{k+1|k} &= A\Sigma_{k|k}A^\top + R \end{aligned} \tag{5}$$

Here, A is the Jacobian of the system dynamics, and R is the system noise. Both of them in our system will be defined as:

$$\begin{aligned} A &= \frac{\partial f}{\partial X_{X=X_k}} = \begin{bmatrix} a_k & x_0 \\ 0 & 1 \end{bmatrix} \\ R &= [0.5] \end{aligned} \tag{6}$$

When incorporating the observation, we first calculate our expected sensor readings from $X_{k+1|k}$, then compute the innovation by taking the difference with the actual sensor reading.

$$y' = g(\mu_{k+1|k}) = \sqrt{\mu_{k+1|k}[0]^2 + 1} \tag{7}$$

$$innovation = y_{k+1} - y' \tag{8}$$

Once this is calculated, we can compute the Kalman Gain as

$$\begin{aligned} K &= \Sigma_{k+1|k} C^\top (C\Sigma_{k+1|k} C^\top + Q)^{-1} \\ C &= \frac{\partial g}{\partial X} = \begin{bmatrix} \frac{mu_{k+1|k}[0]}{\sqrt{\mu_{k+1|k}[0]^2+1}} \\ 0 \end{bmatrix} \end{aligned} \tag{9}$$

Once the Kalman gain is computed, we can get the estimates as

$$\begin{aligned} \mu_{k+1|k+1} &= \mu_{k+1|k} + K * innovation \\ \Sigma_{k+1|k+1} &= (I - KC)\Sigma_{k+1|k} \end{aligned} \tag{10}$$

This process is repeated over the dataset recursively.

3. RESULTS

Below is the plot of the a_k over the observations:

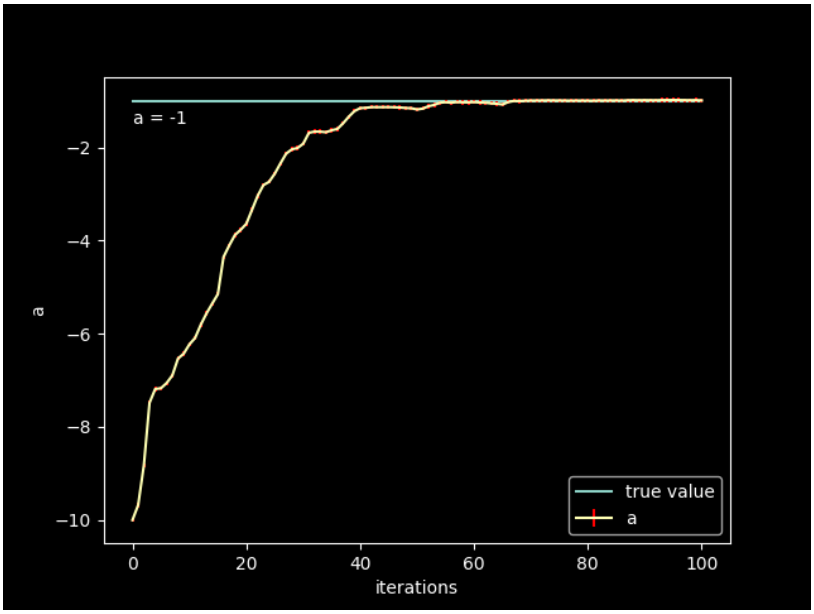

FIGURE 2. Convergence of a over observations

I've zoomed in near the convergence to show the error bars:

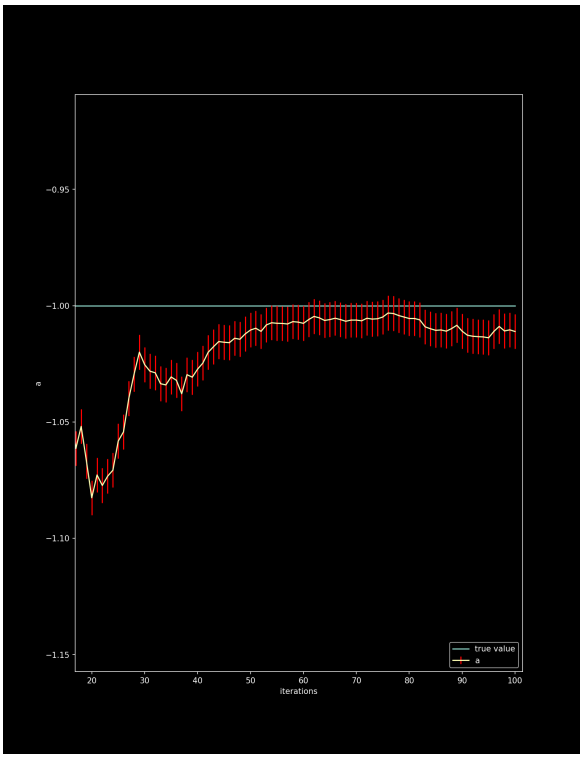

FIGURE 3. Error bars

This is a plot of the standard deviation over the observations:

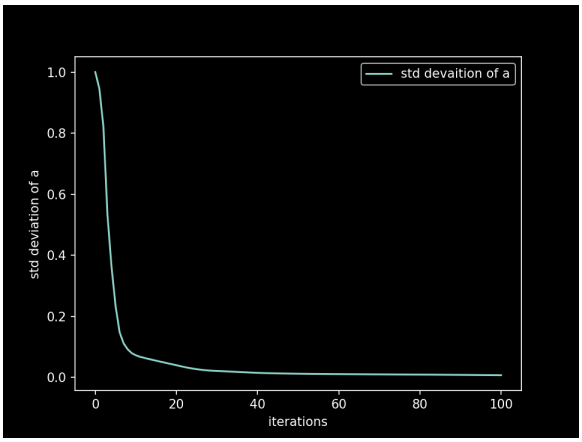

FIGURE 4. Standard deviation of a over observations

This is expected since we have added a_k as a part of the state; the filter will now learn the value of a as observations come to explain the observations themselves better. The innovation contains information about the inaccurate estimate of a.

Solution 2. 4. SENSOR CALIBRATION

The exercise started with the calibration of the sensor. The task was to find sensor sensitivity and bias for each axis of the accelerometer and the gyroscope to match the ground truth captured by the VICON system.

- The rotated matrices obtained were used to find the ground truth accelerations in the x, y, and z directions. These values are calculated by multiplying the rotation matrix with a numpy array containing the gravity vector in the x, y, and z directions.
- The next step of the code calculates the angular accelerations by taking the derivative of the rotation matrix with respect to time using the numpy diff function. The angular velocities are then calculated using quaternion representations of the rotation matrix differences and converting them into their axis-angle representation divided by the time step.
- The angular velocity data is then filtered to remove any outlier values present in the data due to small time steps by checking if its absolute value is greater than 5. If it is, the value is replaced with the previous value in the array. Finally, a smoothing filter is applied to the angular velocity data using the numpy convolution function with a window size of 10.
- The data from the three files were concatenated by adjusting for the unequal lengths of the data and passed through a least squares linear regression solver to get a preliminary calibration. These calibration values were used to plot each sensor data for each file and then manually tweaked to match the signals correctly.

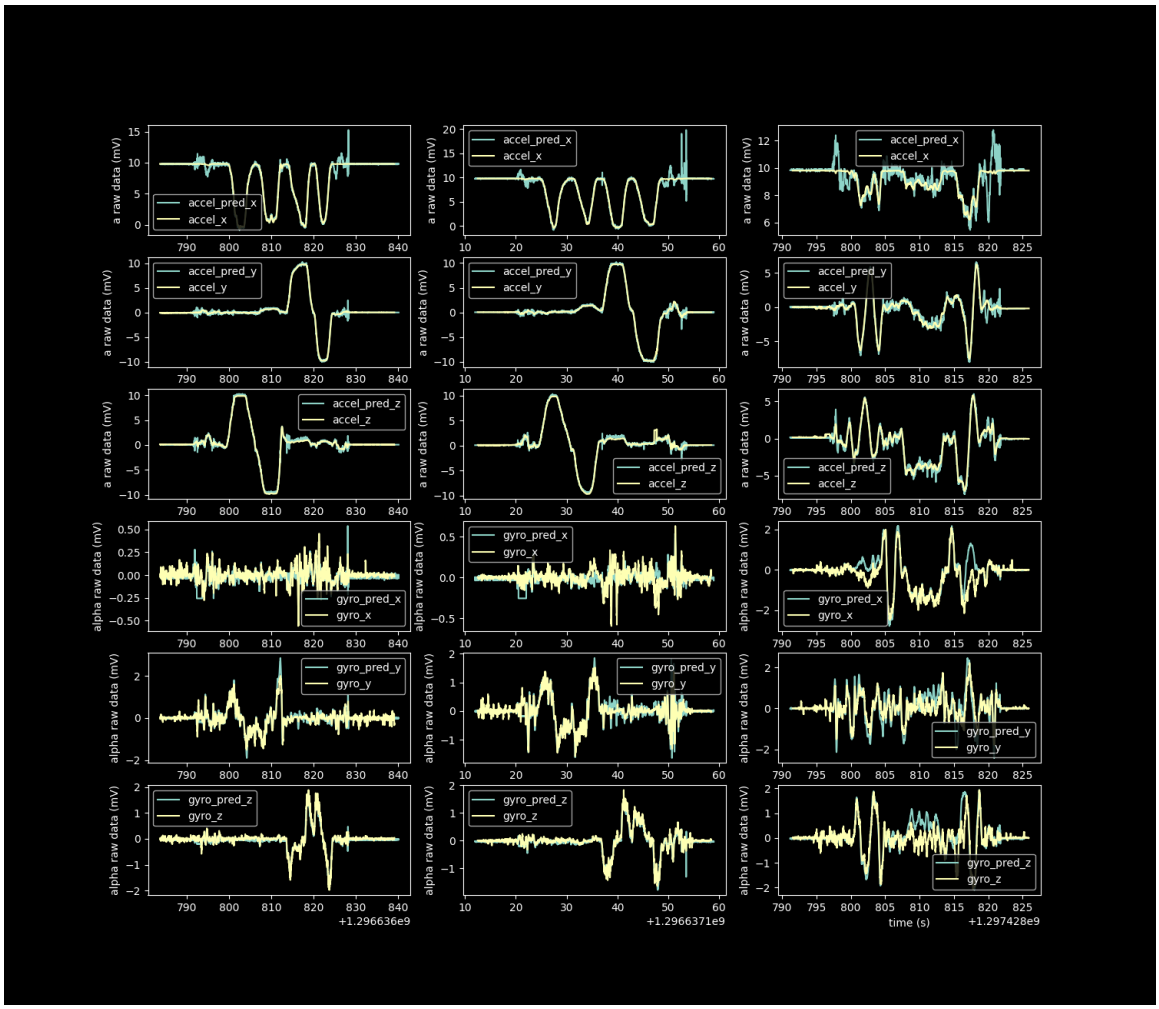

FIGURE 5. Sensor Calibration Results

5. RESULTS

I have plotted the quaternion and the diagonal of the covariance with respect to time below:

File 1

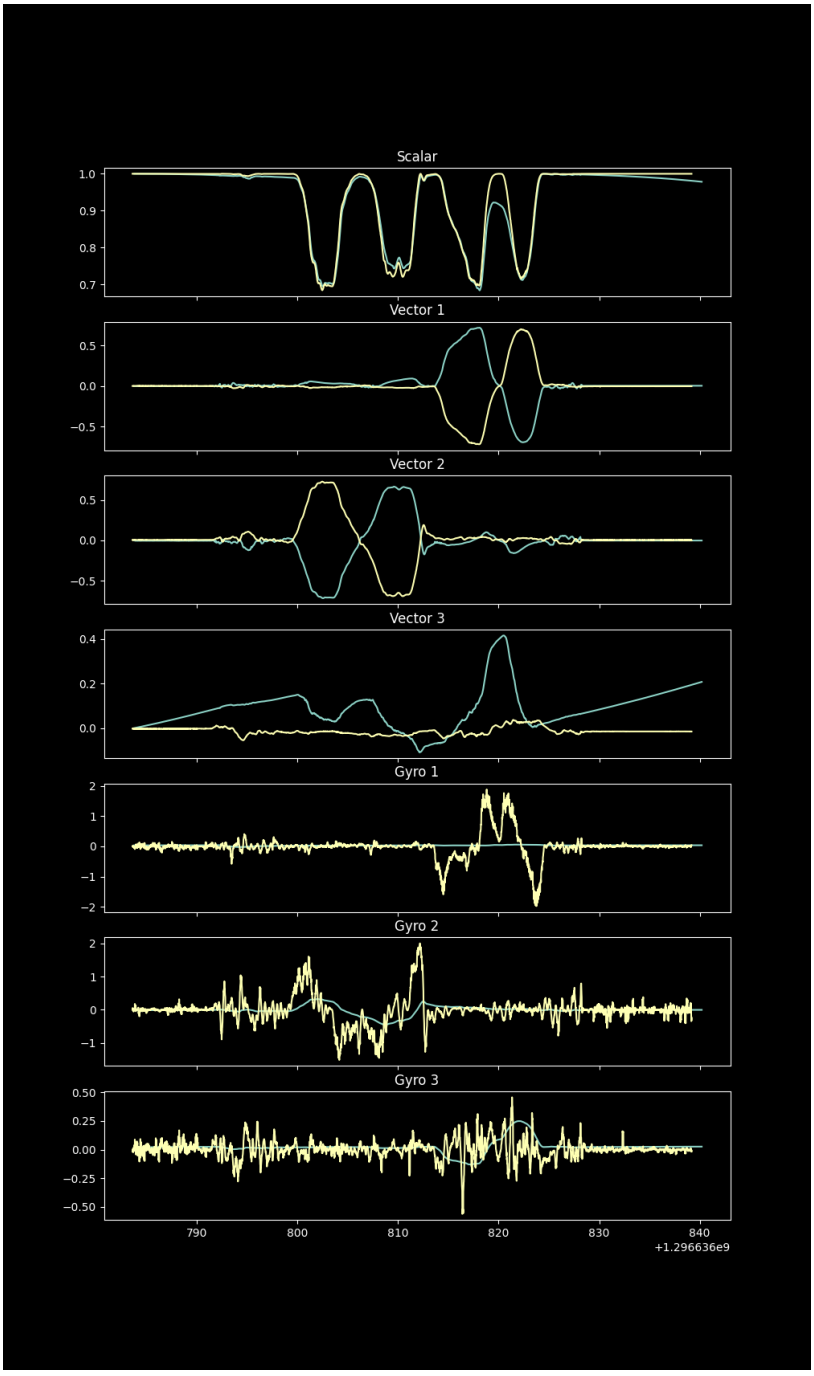

FIGURE 6. State plot for file 1

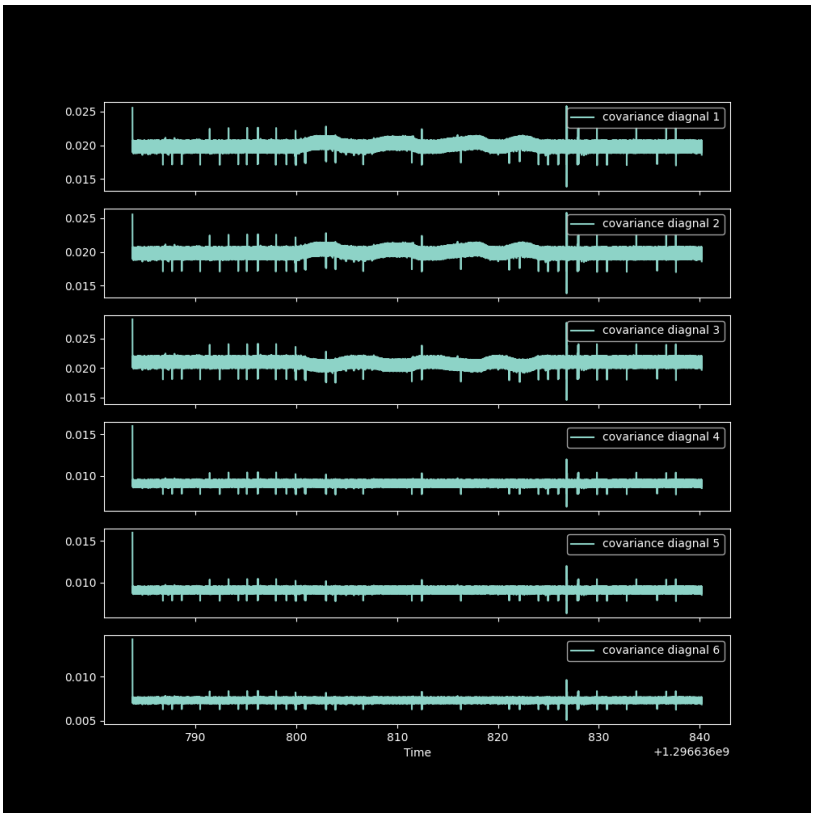

FIGURE 7. Covariance diagonal plot

File2

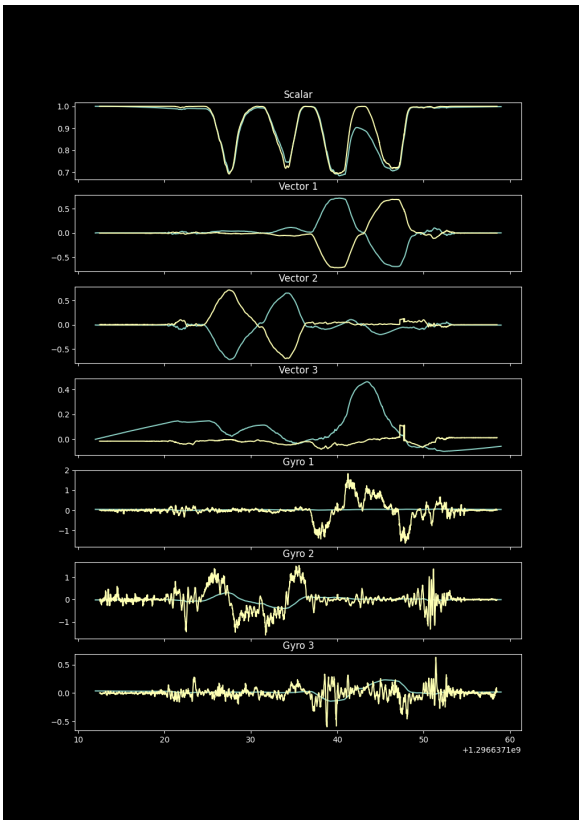

FIGURE 8. State plot for file 2

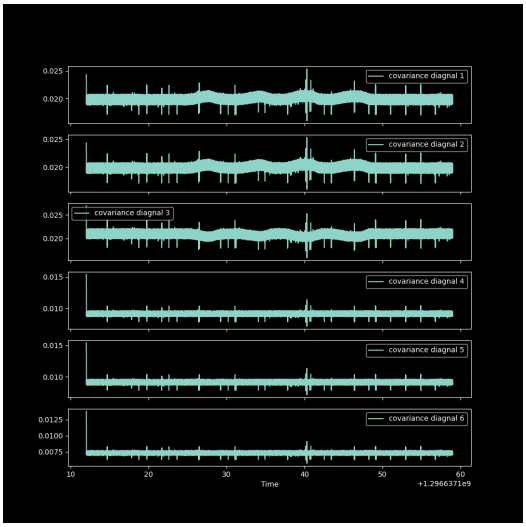

FIGURE 9. Covariance diagonal plot

File3

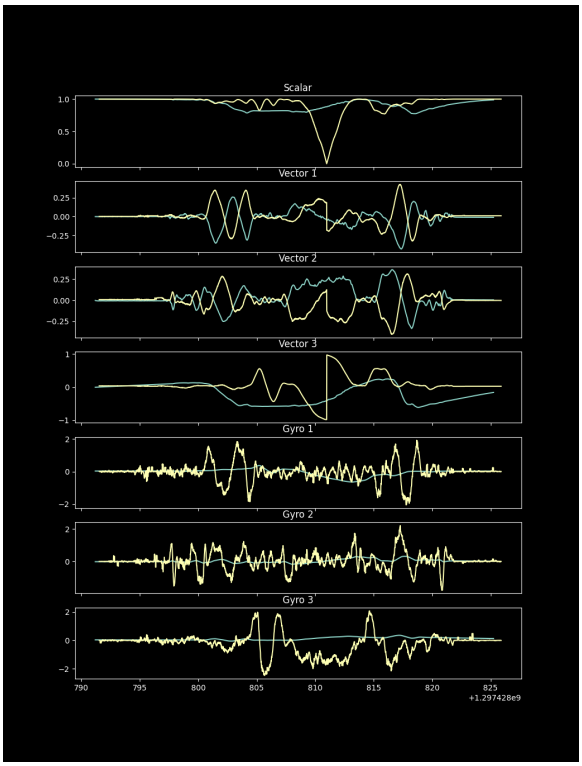

FIGURE 10. State plot for file 3

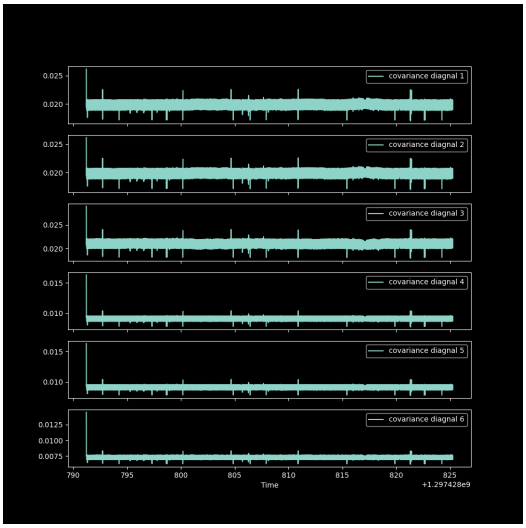

FIGURE 11. Covariance diagonal plot

I plotted Roll Pitch and Yaw Estimates (Orange) against the VICON (Blue) data below for three files.

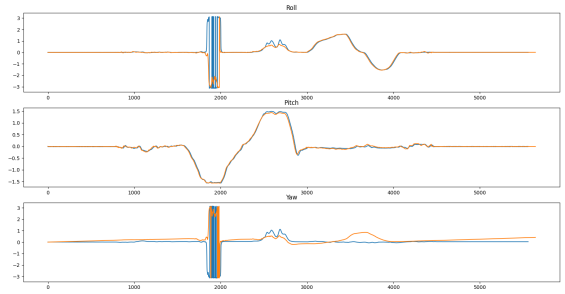

FIGURE 12. Roll, Pitch, Yaw for File 1

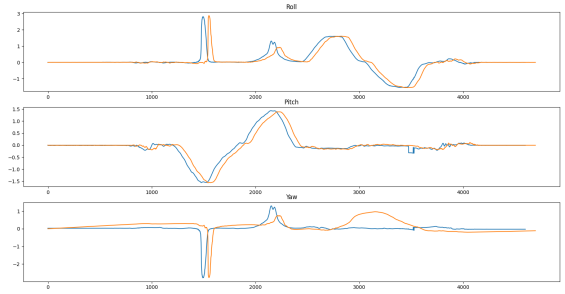

FIGURE 13. Roll, Pitch, Yaw for File 2

FIGURE 14. Roll, Pitch, Yaw for File 3

6. Noise Parameters

I calculated the error from the raw sensor data after my calibration to get preliminary values for the measurement noise. To get a diagonal matrix, I found the variances for each axes and sensor separately.

To estimate the initial process noise, I took the VICON data in orientation when the IMU was stationary and modeled the noise similar to the sensor noise. These preliminary estimates could have yielded more satisfactory filter performance, and I iteratively tuned individual values to improve the performance.

7. Discussion

Although the roll and pitch estimates are accurate, the yaw estimates could be better. Further careful calibration of the noise parameters (both process and measurement noise) will improve this. This stems from the sensitivity of the Yaw estimate accuracy on the R and Q I observed during my tuning.