EECE 5554 Robotics Sensing and Navigation

Lab-4 Navigation with IMU and Magnetometer v1.1

INTRODUCTION:

In this lab the aim is to integrate data from the GPS and IMU sensors to achieve sensor fusion, where one of the sensors fails to gather data allowing us to depend on the other. Additionally, dead reckoning is relevant to this topic as it is a navigation technique that estimates person current position based on their prior location, direction and speed of travel and can be used when GPS fails. By merging data from both sensors we can enhance navigation accuracy and obtain reliable data, even if one of the sensors is unreliable.

Setup and Analysis of the data collected:

- IMU sensor is placed inside the NUANCE car and is adjusted in a way that the pitch value to be zero or somewhere around zero.
- Our data collection process involves IMU and GPS data that were highly affected by noise and errors. The main goal of this study is to minimize these errors and noise in the data and generate a plot for comparison with alternative calculated data.

YAW Estimation:

Calibration of Magnetometer:

The below part shows the comparison between raw and corrected magnetometer date after removing hard and soft iron errors.

Hard iron correction:

The hard-iron distortion occurs due to materials that add a constant field to the Earth's magnetic field leading to an added constant value to the output of each magnetometer axis. Fig-1 shows the magnetic field plot that has not been corrected shows a center at (0.1,0.25). To correct this translation, we need to calculate the offset of each axis and subtract it from them.

Soft iron correction:

Soft-iron distortion is a type of distortion caused by materials that distort or alter a magnetic field without generating an additional magnetic field, and hence, it is not additive. The extent of soft-iron distortion is influenced by the material's orientation concerning the sensor and the magnetic field. This type of distortion is detectable when an ellipse with angular rotation is observed. As a gist of how to remove the hard and soft iron errors we have to perform Subtracting the following offsets from respective raw magnetic-x and magnetic-y values

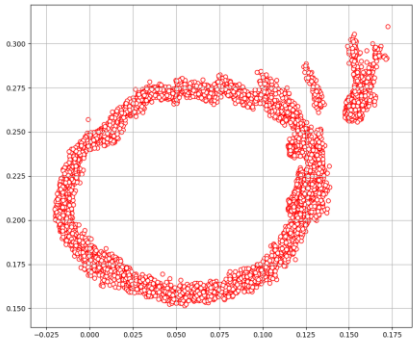
X offset = [maximum(x) + minimum(x)]/2

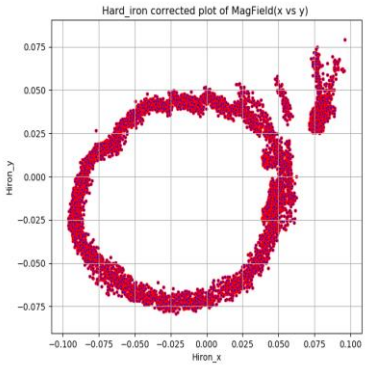
Y offset = [maximum(y) + minimum(y)]/2

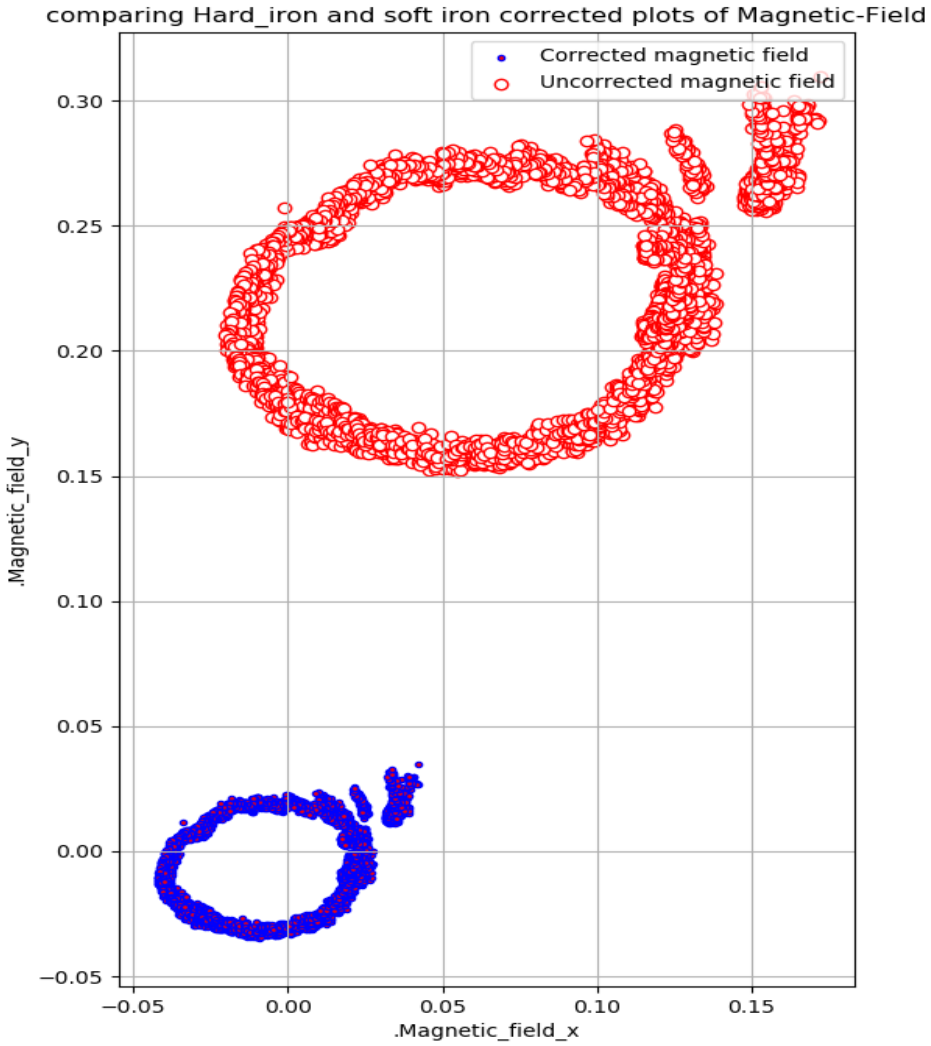
Followed by rotation and scaling with the scale factor which is the ratio of length of major axis to minor axis.

Plot-1: The magnetometer X-Y plot before and after hard and soft iron calibration:

Below plots show that the raw data is presented as a scattered red circle that needs to be adjusted to a corrected circle through a process of translation, rotation, and scaling with the shifted origin as the desired outcome. This correction process aims to eliminate both the hard and soft iron errors from the raw data.

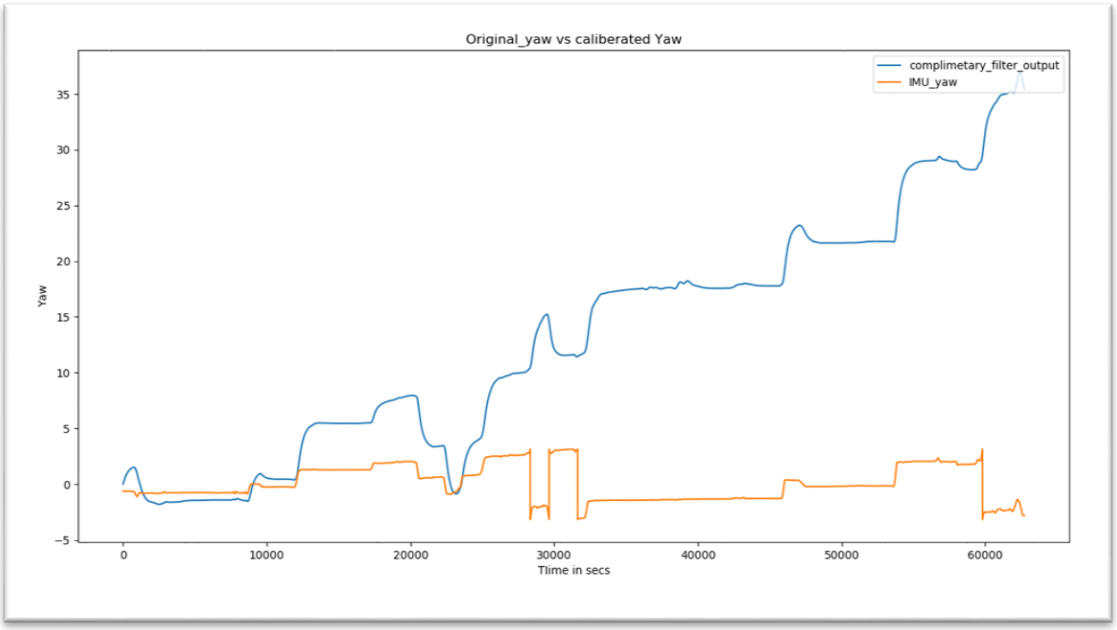




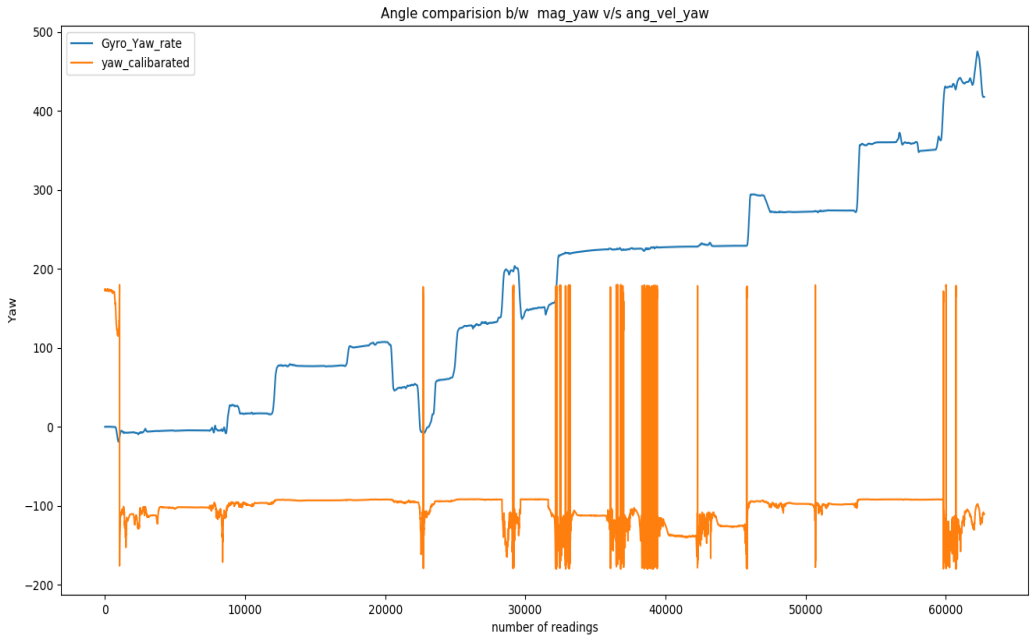


Comparing the Magnetic field in X and Y directions before and after correcting the data.

Below is the Time series magnetometer data before and after the correction



Below is the Magnetometer Yaw & Yaw Integrated from Gyro together



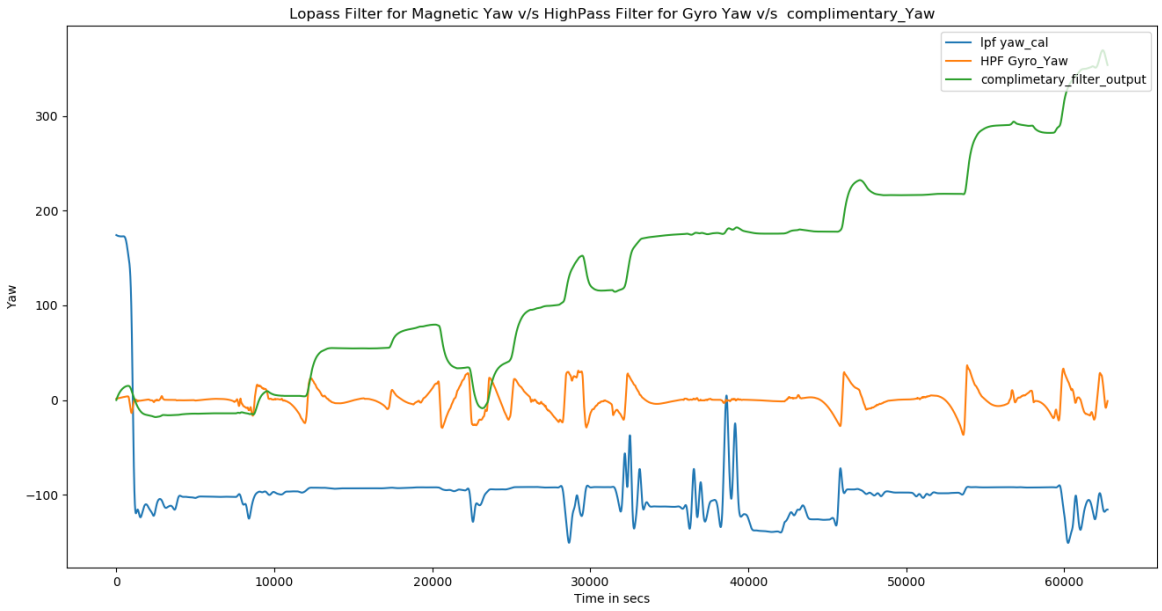
Two factors supported the use of the yaw from complementary output for navigation over the yaw calculated by the IMU. Firstly, the complementary filter output is derived by combining the low-pass and high-pass filters, resulting in a signal that is both smoothened by the low-pass filter and sharpened by the high-pass filter.

Secondly, when comparing trajectory paths estimated using both the gyro integrated yaw and the complementary filter output yaw for velocity and path estimation separately, and comparing them with the GPS trajectory, promising outcomes were observed with the complementary filter output.

How did you use a complementary filter to develop a combined estimate of yaw? What components of the filter were present, and what cutoff frequency(ies) did you use?

Below are the LPF, HPF, and CF plots together.

The complementary filter is utilized to estimate the yaw angle by combining the reliable low-frequency estimate from the magnetometer with the precise high-frequency measurements from the gyro, resulting in a more accurate and stable estimation of the yaw angle.

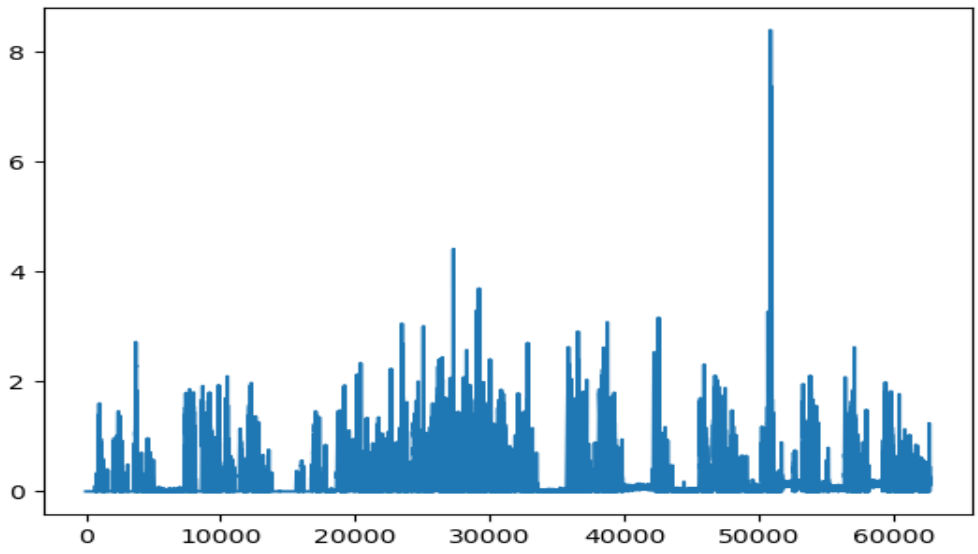


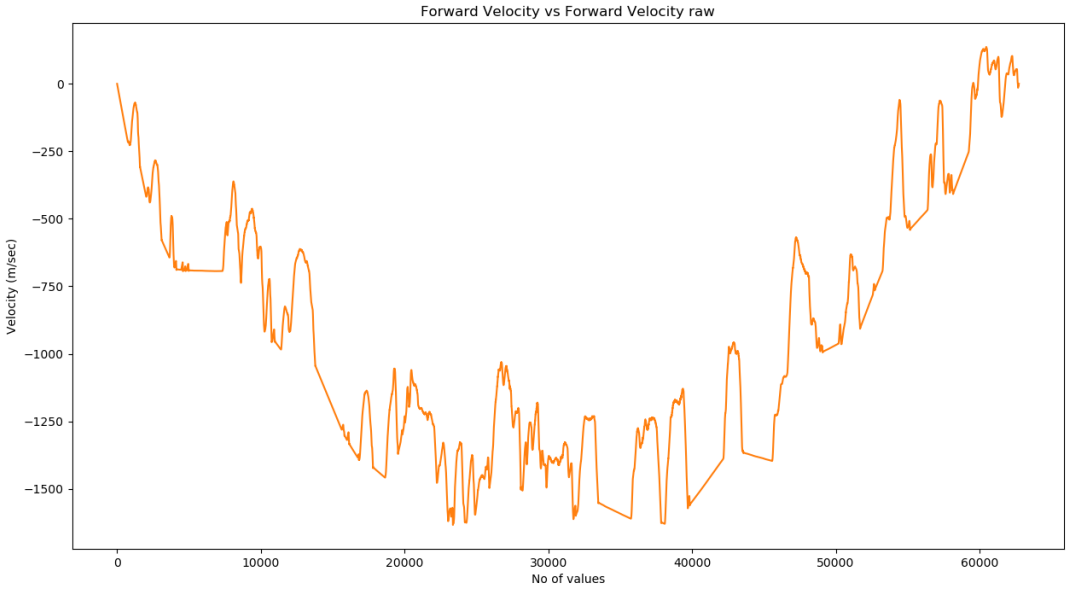
-

Estimate the Forward Velocity

The GPS Velocity over Time for the NUANCE car displays a consistent value over time, indicating that the car was traveling at a constant speed during that period. However, there are specific time intervals where the GPS Velocity drops to zero, indicating that the car was stationary. These intervals could be attributed to instances where the car was stuck in traffic.

Plot of GPS Velocity over Time:





Which estimate or estimates for yaw would you trust for navigation? Why?

- By utilizing the complementary filter output, which is a combination of low pass and high pass filtered signals, the obtained yaw angle is significantly more accurate and reliable compared to other estimation methods.
- Additionally, the presence of the low pass filter results in a smoother output signal while the high pass filter provides sharper results.
- Moreover, the complementary filter output is relatively free from errors such as noise and drift over time, making it a superior choice for navigation applications.

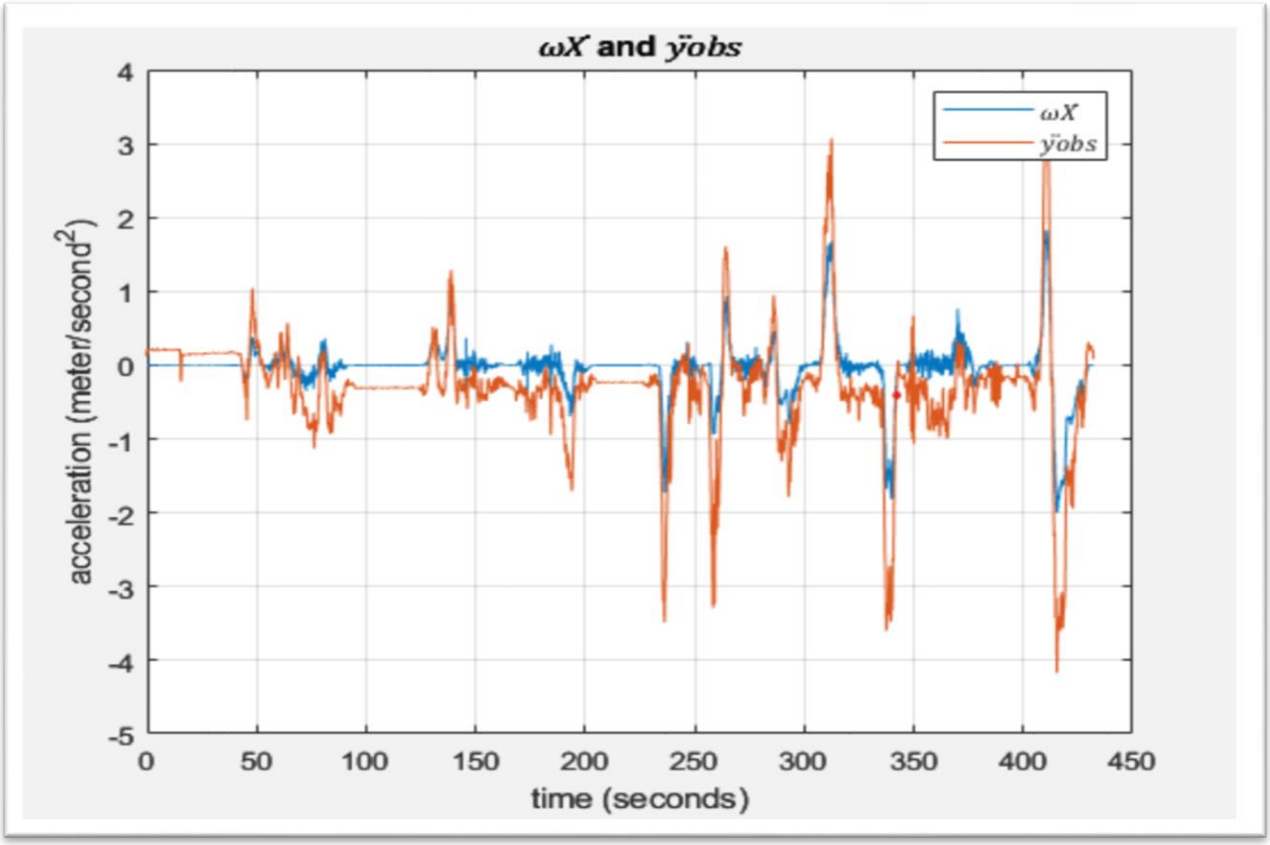
What adjustments did you make to the forward velocity estimate and why?

- The IMU velocity estimate prior to correction displayed negative values due to underestimation, as the true velocity was substantially greater than the estimated velocity.
- To address this error, a bias was calculated and subtracted from the IMU velocity estimate at stationary positions to enhance the accuracy of the forward velocity estimation.
- By observing the plot, it is evident that the removal of bias at stationary positions corrects the IMU velocity estimate and minimizes errors such as noise, drift, and offset.

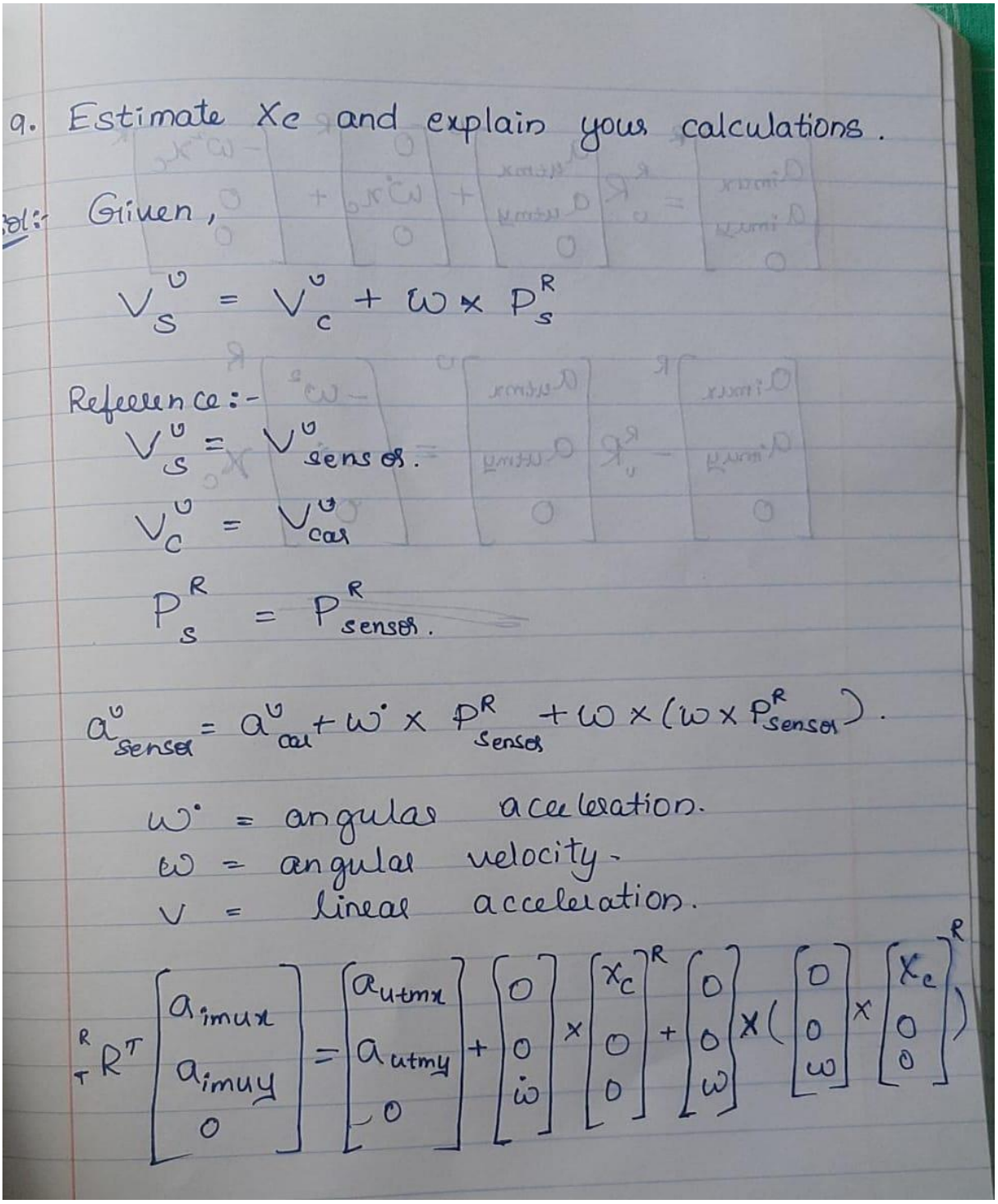
What discrepancies are present in the velocity estimate between acceleration and GPS? Why?

- The dissimilarity noticed between the velocity estimates derived from IMU acceleration and GPS is attributed to the negative velocity values present in the IMU estimate, which are absent in the GPS estimate.
- The GPS velocity estimate remains constant over time, with zero velocity during stationary periods.
- On the other hand, the IMU acceleration values contain bias due to the velocity estimate being considerably smaller than the true velocity.
- Using the notation (X, Y,0) for the position of the center-of-mass (CM) of the vehicle and (0,0, ω) for its rotation rate about the CM
- we denote the position of the inertial sensor in space by (x,y,0) and its position in the vehicle frame by (xc,0,0).
- If Y = 0 (the vehicle is not skidding sideways) and ignoring the offset by setting xc = 0 (the IMU is at the center of mass of the vehicle), the first equation reduces to X= $\ddot{x}obs$. Integrating X yields X .
- which is the velocity of the vehicle w.r.t the center of mass.
- We can use this to compute ωX and compare it to $\ddot{y}\ obs$, as shown in Figure 8.
- ωX is the product of the velocity obtained from linear acceleration in the x direction and the angular velocity in the z direction. The corresponding plots show that the noise in $\ddot{y}\ obs$ is much greater than that in ωX.
- This is because when integrating, errors also get integrated with the original data, causing more noise.
- However, for ωX, multiplying X velocity with ω compensates for angular rotation and reduces the error.

Dead Reckoning with IMU



Estimate xc and explain your calculations:



9. Estimate X_c and explain your calculations.

Sol:- Given,

$$V_s^v = V_c^v + \omega \times P_s^R$$

Reference:-

$$V_s^v = V_{sensor}^v$$

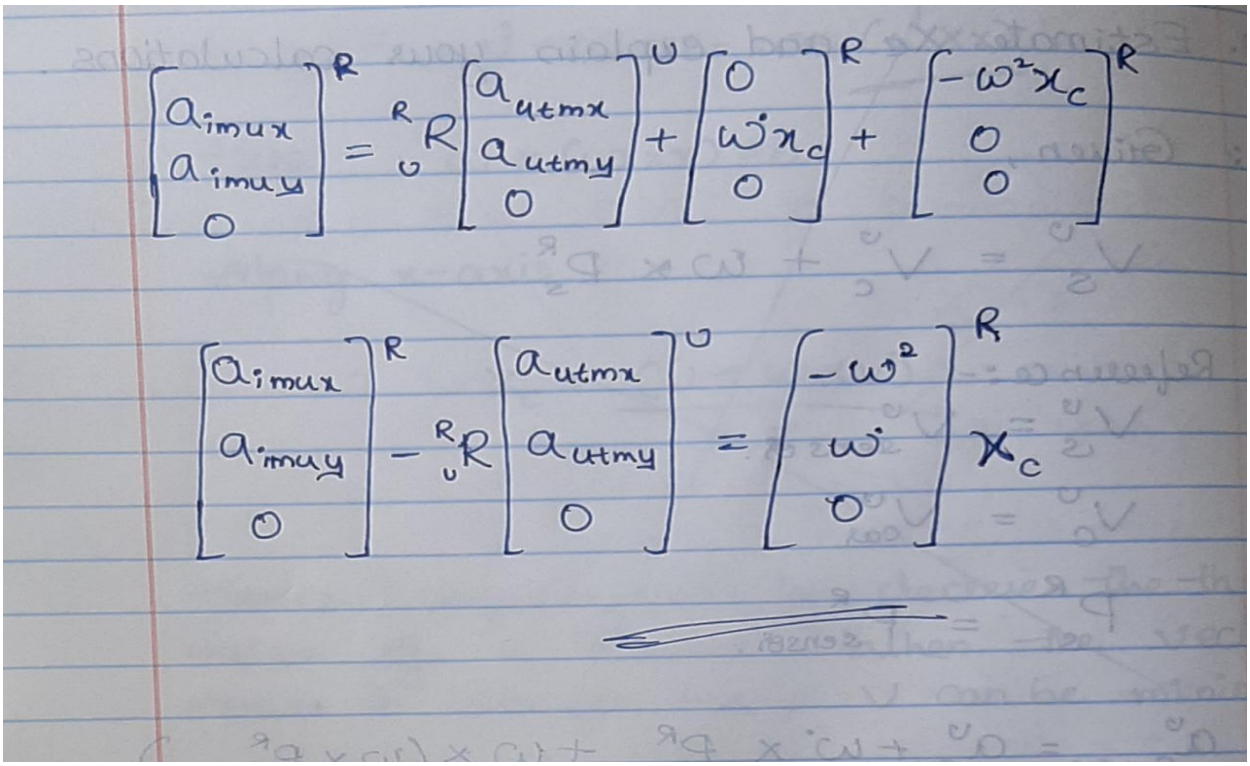
$$V_c^v = V_{cal}^v$$

$$P_s^R = P_{sensor}^R$$

$$a_{sensor}^v = a_{cal}^v + \omega^{\cdot} \times P_{sensor}^R + \omega \times (\omega \times P_{sensor}^R).$$

$\omega^{\cdot}$ = angular acceleration.
ω = angular velocity.
V = linear acceleration.

$$\begin{matrix}R\\T\end{matrix} R^T \begin{bmatrix} a_{imux} \\ a_{imuy} \\ 0 \end{bmatrix} = \begin{bmatrix} a_{utmx} \\ a_{utmy} \\ 0 \end{bmatrix} + \begin{bmatrix} 0 \\ 0 \\ \dot{\omega} \end{bmatrix} \times \begin{bmatrix} X_c \\ 0 \\ 0 \end{bmatrix}^R + \begin{bmatrix} 0 \\ 0 \\ \omega \end{bmatrix} \times \left(\begin{bmatrix} 0 \\ 0 \\ \omega \end{bmatrix} \times \begin{bmatrix} X_c \\ 0 \\ 0 \end{bmatrix}^R \right)$$



Note:

The driver code used for data collection belongs to my team member Santhosh Rao Sripathy (Group 0).