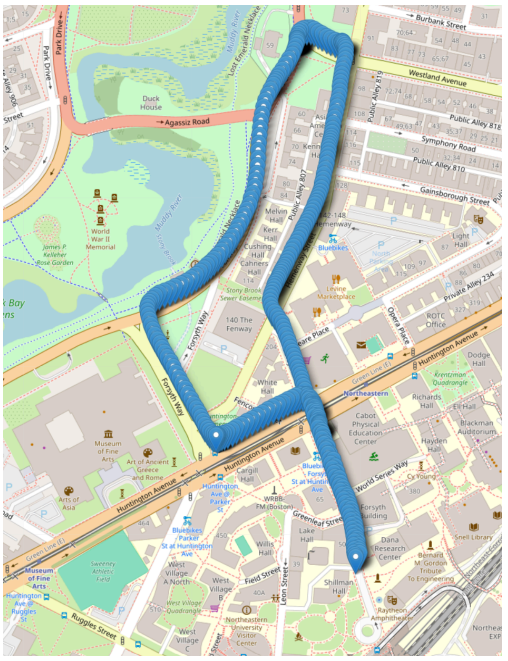
LAB_4 - EECE 5554 Robotics Sensing and Navigation

Report Submitted By - Tapan Shashikant Patil (patil.tap@northeastern.edu)

Summary:

In this lab a navigation stack was built by integrating GPS and IMU sensors. Then two data sets were collected, one was used for magnetometer hard iron and soft iron error calibration and another to calculate the rest of the parameters as instructed.



1. Estimating Heading (Yaw):

1.1. Magnetometer Calibration:

The **hard-iron** distortion is typically caused due to the presence of a constant magnetic field near the sensor which shifts the magnetic field sensed by the sensor; this was verified by observing the magnetometer data which was offset from origin. To correct this I calculated offset for both x,y magnetometer data by averaging the max and min values. Which removed and constant bias in the data.

$$\text{offset}_X = \frac{\max(\text{magX\_data}) + \min(\text{magX\_data})}{2}$$
$$\text{offset}_Y = \frac{\max(\text{magY\_data}) + \min(\text{magY\_data})}{2}$$

Then these offsets were subtracted from the original magnetometer to get corrected magnetometer data, this centered the data being centered around $(0, 0)$

$$\text{magX\_corrected} = \text{magX\_data} - \text{offset}_X$$
$$\text{magY\_corrected} = \text{magY\_data} - \text{offset}_Y$$

Then to correct **soft-iron** error caused by nearby ferromagnetic materials, which alters the shape of data making it elliptical rather than circle. This was corrected by first calculating the scaling factor for soft-iron first rang of X and range of Y was calculated using following formula

$$\text{range}_X = \max(\text{magX\_corrected}) - \min(\text{magX\_corrected})$$
$$\text{range}_Y = \max(\text{magY\_corrected}) - \min(\text{magY\_corrected})$$

The scaling factor was derived to equalize these ranges as,

$$\text{scalingFactor} = \sqrt{\frac{\text{range}_X}{\text{range}_Y}}$$

This scaling factor was then applied to X-axis value to bring both axes into alignment

$$\text{magX\_final} = \frac{\text{magX\_corrected}}{\text{scalingFactor}}$$

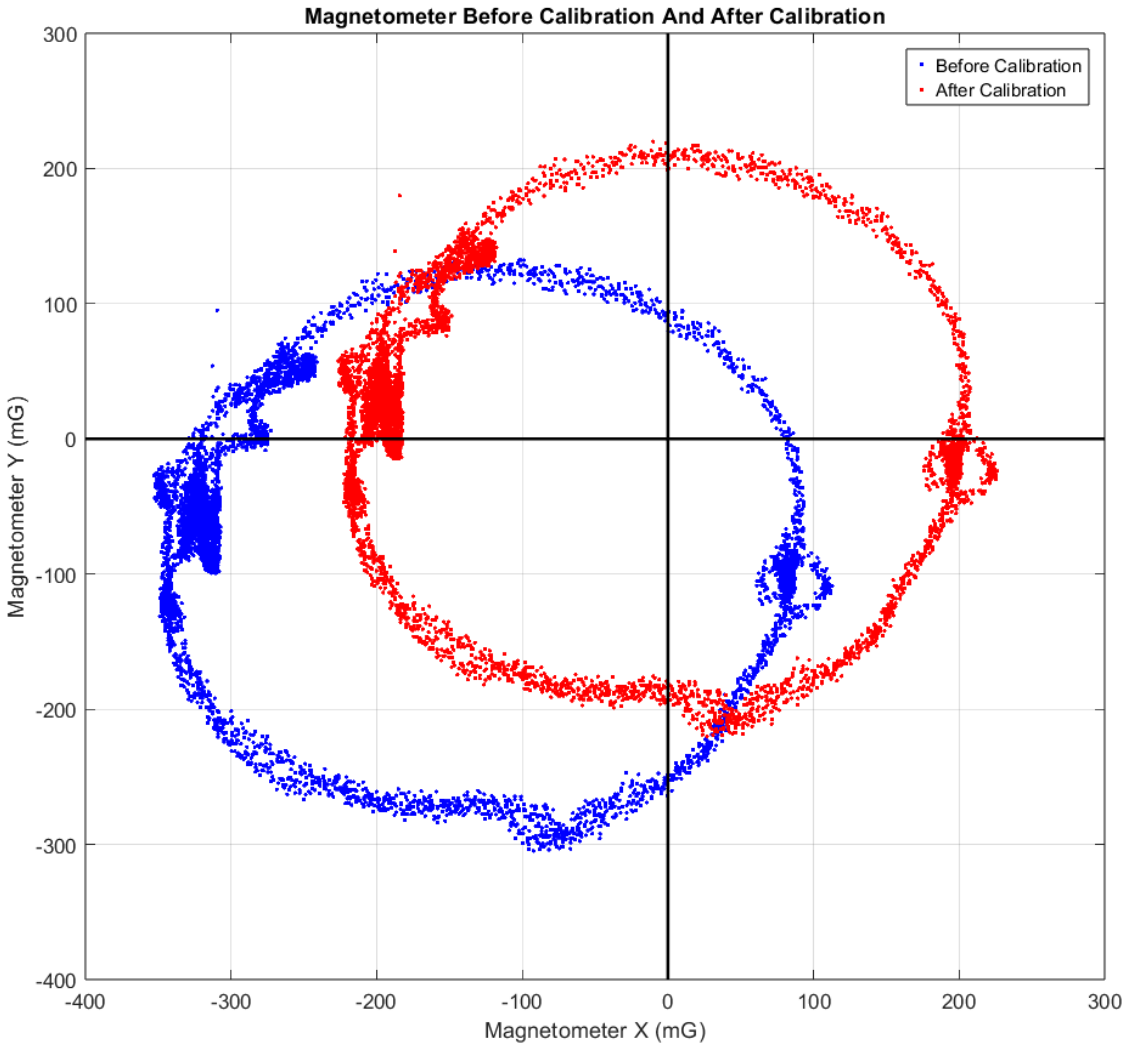


Fig1.1.1. Magnetometer X-Y plot Before and After Hard and Soft Iron Calibration

1.2. Sensor Fusion:

Driving Data Calibration:

The hard-iron offsets and soft iron scaling parameters were then used to calibrate the magnetometer data from driving.

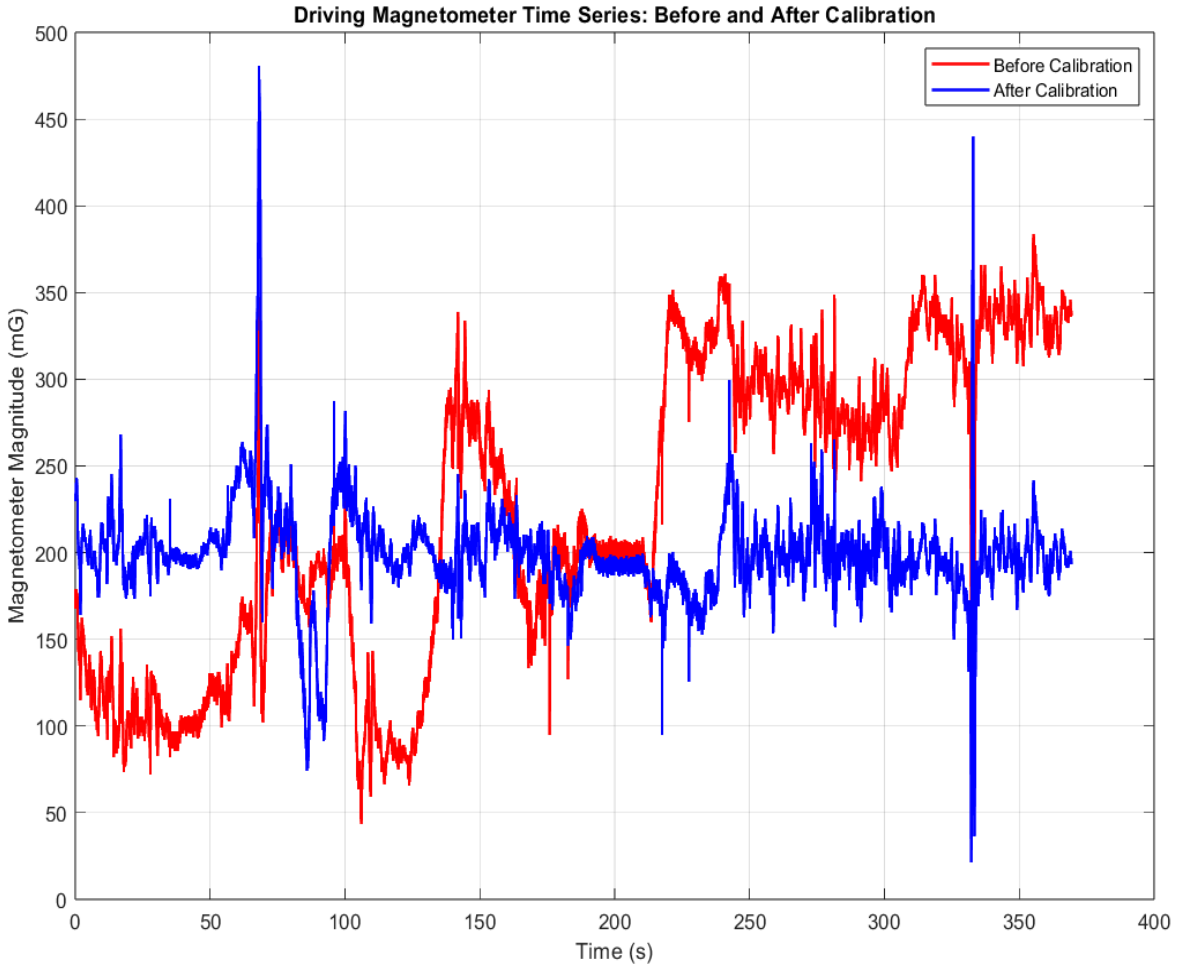


Fig1.2.1. Driving Magnetometer Time Series Plot

The raw and calibrated magnetometer values were then used to calculate raw yaw and calibrated yaw which were then plotted

$$\text{yaw}_{\text{raw}} = \arctan\left(\frac{\text{magY}_{\text{raw}}}{\text{magX}_{\text{raw}}}\right)$$

$$\text{yaw}_{\text{calibrated}} = \arctan\left(\frac{\text{magY}_{\text{corrected}}}{\text{magX}_{\text{corrected}}}\right)$$

The yaw angle was obtained by integrating yaw rate, as **cumtrapz** function provides a cumulative integration result at each time step where as **trapz** computes the total integral over the entire time and gives a final value

$$\text{yaw\_angle} = \int \text{angular\_velocity}_z \, d(\text{time\_vector}) \approx \text{cumtrapz}(\text{time\_vector}, \text{angular\_velocity}_z)$$

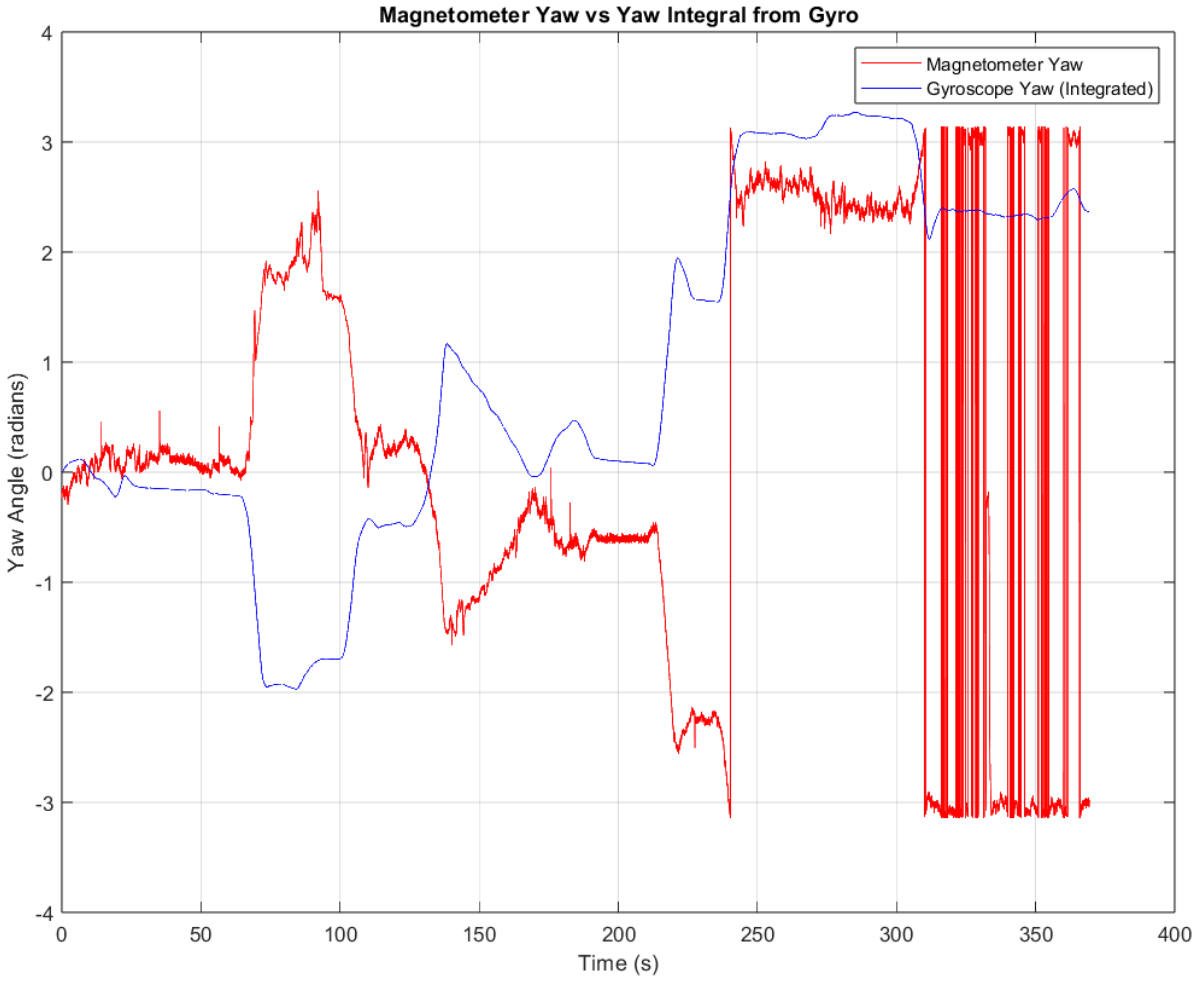


Fig1.2.2. Magnetometer Yaw Vs Yaw Integrated from Gyro

Complementary Filter for Yaw Angle:

To get a combined estimate of yaw angle a complementary filter was implemented to combine data from magnetometer and gyroscope. The complementary filter effectively combines the low frequency data from the magnetometer, which gives absolute orientation with the high frequency data from the gyroscope, which captures rapid changes in yaw.

High - Pass Filter for Gyroscope:

The integrated gyroscope data was passed through a high-pass filter to capture high frequency changes. This helps mitigate drift in data which accumulates over time.

The high pass filter implemented with filter constant α_{gyro} , and a calculated cut-off frequency **120.96 Hz** successfully isolated the high frequency data.

$$\text{yaw}_{\text{gyro\_highpass}}(i) = \alpha_{\text{gyro}} \cdot (\text{yaw}_{\text{combined}}(i-1) + \text{gyroscope\_yaw}(i) \cdot \Delta t)$$

Where,

- α_{gyro} is the high-pass filter constant for gyroscope
- $\text{yaw}_{\text{combined}}(i-1)$ is the combined yaw angle
- $\text{gyroscope\_yaw}(i)$ is the integrated yaw rate from gyroscope
- Δt is the time step calculated as
  $\Delta t = \text{timestamp\_vector}(i) - \text{timestamp\_vector}(i-1)$

Low - Pass Filter for Magnetometer:

The calibrated magnetometer data was passed through a low-pass filter to focus on low-frequency components. This ensures the stability in long term orientation data necessary for stable yaw estimation.

The low pass filter implemented with filter constant α_{mag}, and a calculated cut-off frequency **0.27 Hz** gave reliable data.

$$\text{yaw}_{\text{mag\_lowpass}}(i) = \alpha_{\text{mag}} \cdot \text{magnetometer\_yaw}(i)$$

Where,

- α_{mag} is the low-pass filter constant for magnetometer
- $\text{magnetometer\_yaw}(i)$ is the current yaw angle from gyroscope

Then combined complementary filter equation was calculated by,

$$\text{yaw}_{\text{combined}}(i) = \text{yaw}_{\text{gyro\_highpass}}(i) + \text{yaw}_{\text{mag\_lowpass}}(i)$$

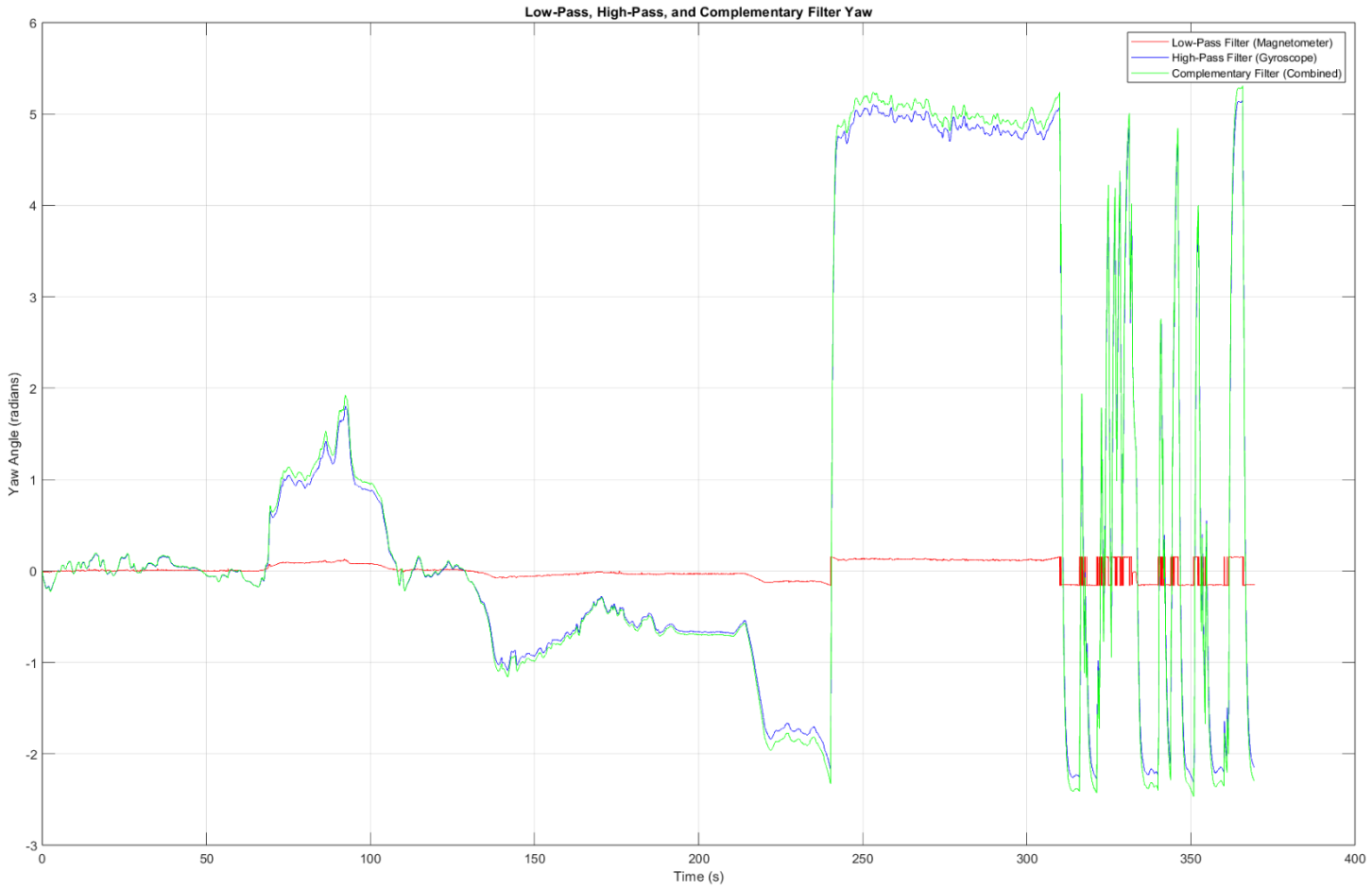


Fig1.2.3. LPF, HPF, and CF plots together

Next to compare sensor fusion yaw result with yaw angle computed by the IMU first yaw angle ψ was calculated from quaternion

$$\psi = \text{atan2}\left(2 \cdot (q_w \cdot q_z + q_x \cdot q_y),\ 1 - 2 \cdot (q_y^2 + q_z^2)\right)$$

In this equation the rotation about the z axis was isolated after converting quaternion to ZYX Euler angles. Then matlab's unwrap function was used to prevent abrupt transition between $\pm\pi$ then to ensure all angles stay in between $[-pi, pi]$

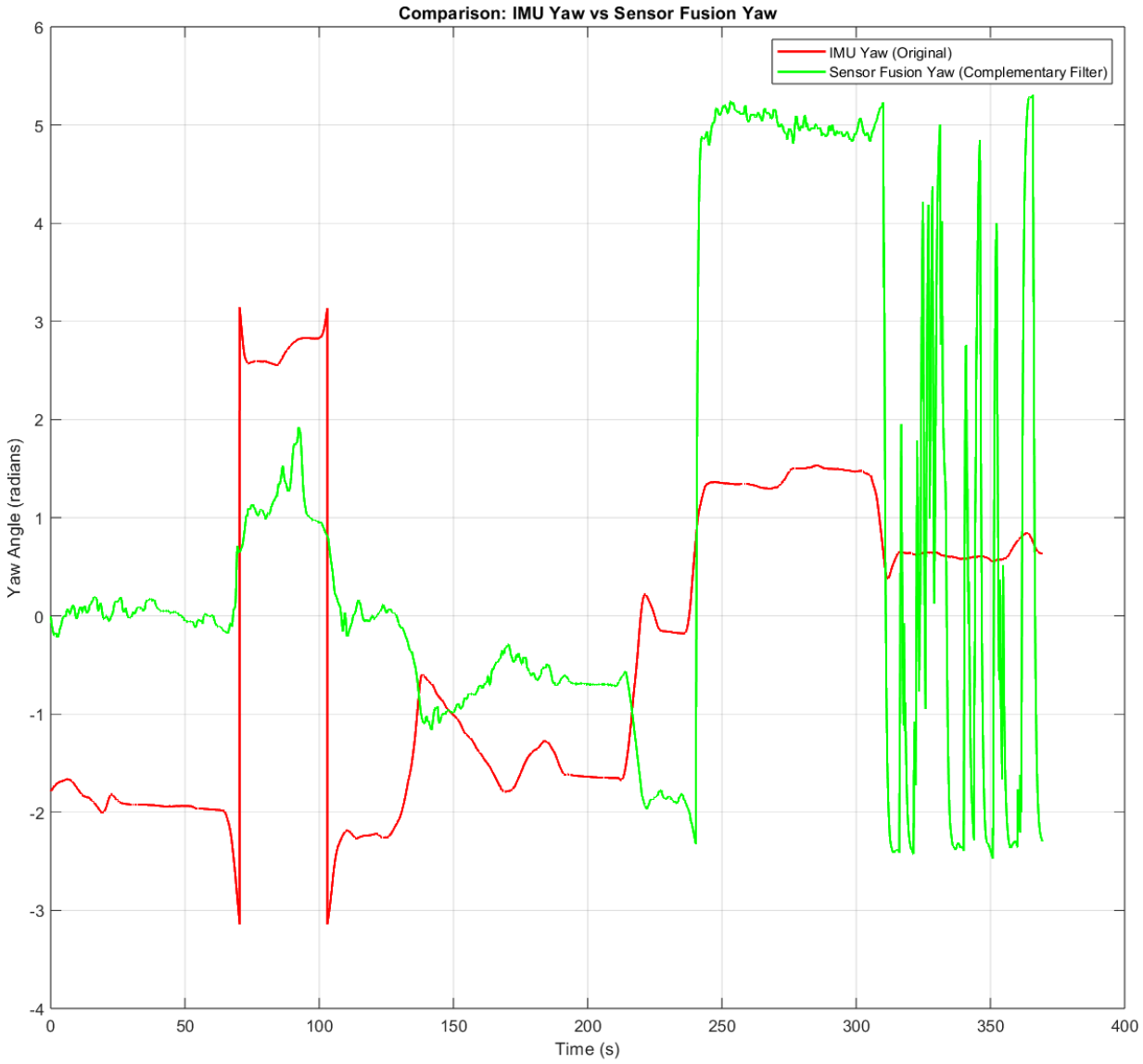


Fig1.2.4. Yaw from the Complementary filter & Yaw angle computed by the IMU together

After analyzing the plot it was evident that the complementary filter is effectively fusing the gyroscope and magnetometer data resulting in a more stable and reliable yaw angle than raw IMU output.
The complementary filter is more resistant to noise drift and sudden variations in sensor reading makes it more favorable for applications where accurate yaw estimate is required over an extended period of time.

2. Estimate The Forward Velocity:

The forward velocity was calculated by using following formula,

$$\text{driving\_forward\_velocity} = \text{cumtrapz}(\text{driving\_timestamp\_vector}, \text{driving\_forward\_acceleration})$$

Then to calculate velocity estimate from GPS data first difference in latitude and longitudes was calculated in radian

$$\Delta\text{lat} = \text{deg2rad}(\text{driving\_gps\_latitude}(i) - \text{driving\_gps\_latitude}(i-1))$$
$$\Delta\text{lon} = \text{deg2rad}(\text{driving\_gps\_longitude}(i) - \text{driving\_gps\_longitude}(i-1))$$

Then longitude and latitude converted into radian for further calculation

$$\text{lat1} = \text{deg2rad}(\text{driving\_gps\_latitude}(i-1))$$
$$\text{lat2} = \text{deg2rad}(\text{driving\_gps\_latitude}(i))$$

Using Haversine Formula the great-circle distance between two points on the Earth's surface was calculated

$$a = \sin^2\left(\frac{\Delta\text{lat}}{2}\right) + \cos(\text{lat1}) \cdot \cos(\text{lat2}) \cdot \sin^2\left(\frac{\Delta\text{lon}}{2}\right)$$

Where,

$$c = 2 \cdot \text{atan2}(\sqrt{a}, \sqrt{1-a})$$

Distance between two points is calculated by

$$\text{distance} = \text{earth\_radius} \cdot c$$

Similarly time difference was calculated by

$$\text{delta\_time} = \text{driving\_gps\_time\_vector}(i) - \text{driving\_gps\_time\_vector}(i-1)$$

Finally, the velocity was calculated as

$$\text{driving\_gps\_velocity}(i-1) = \frac{\text{distance}}{\text{delta\_time}}$$

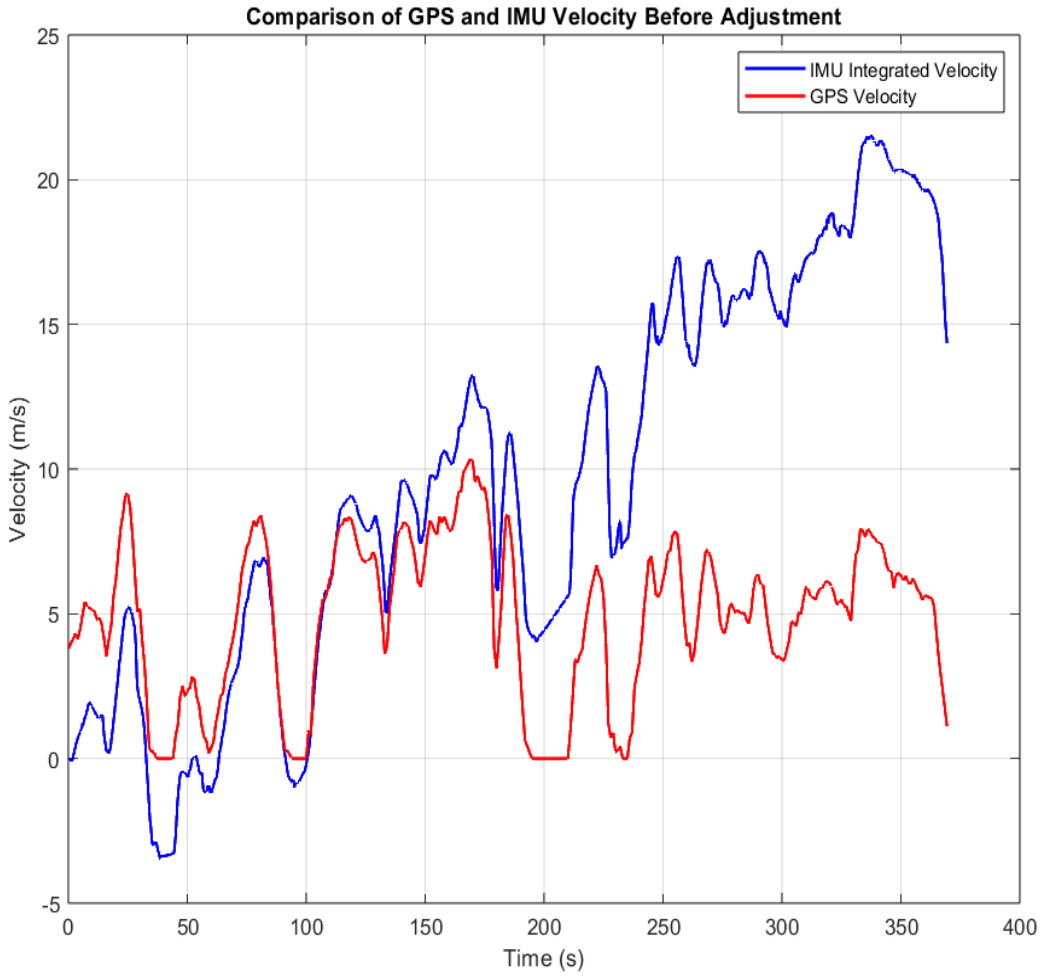


Fig 2.1.Velocity estimate from the GPS with Velocity estimate from accelerometer before adjustment

After analyzing the plots, it was observed that the velocity estimate from GPS is more reliable when compared to IMU. At first both of the velocities follow similar patterns but overtime IMU velocity drifts upwards reaching a maximum of < 20 m/s while GPS velocity stays within reasonable range around 10 m/s considering data was collected while the car was traveling within light traffic. This upward drift in IMU data might be because of errors accumulating from integrating acceleration as on multiple occasions car stopped for signals and slowed down, which is a common problem with IMU sensors.This shows the importance of using GPS to correct the drift in IMU velocity estimates.
The IMU velocity is then adjusted by first applying a bias threshold which was defined by identifying small bias threshold for detecting non-zero velocities
From GPS data,

$$\text{bias\_threshold} = 0.01\,\text{m/s}$$

Then from GPS data specific indices was found where velocity is below threshold

$$\text{near\_zero\_velocity\_indices} = \{i : |\text{gps\_velocity\_interp}[i]| < \text{bias\_threshold}\}$$

The mean acceleration at identified indices represent the bias in forward acceleration

$$\text{bias\_acceleration} = \frac{1}{N} \sum_{i \in \text{near\_zero\_velocity\_indices}} \text{driving\_forward\_acceleration}[i]$$

Where,
N is the number of indices in $\text{near\_zero\_velocity\_indices}$

To correct for acceleration bias bias acceleration was subtracted from IMU forward acceleration

$$\text{driving\_forward\_acceleration\_corrected} = \text{driving\_forward\_acceleration} - \text{bias\_acceleration}$$

A threshold was also applied to driving_forward_acceleration_corrected to make the accelerations zero below threshold.

$$driving\_forward\_acceleration\_corrected(abs(driving\_forward\_acceleration\_corrected) <= 0.33) = 0$$

Corrected forward velocity was then obtained by re-integrating corrected forward acceleration

$$\text{driving\_forward\_velocity\_corrected} = \int_0^t \text{driving\_forward\_acceleration\_corrected}\, dt$$

Then to correct for offset in plots mean of GPS velocity and IMU velocity is calculated

$$\text{velocity\_offset} = \text{mean}(\text{gps\_velocity\_interp}) - \text{mean}(\text{driving\_forward\_velocity\_corrected})$$

After applying this offset final corrected forward velocity was obtained

$$\text{driving\_forward\_velocity\_corrected\_shifted} = \text{driving\_forward\_velocity\_corrected} + \text{velocity\_offset}$$

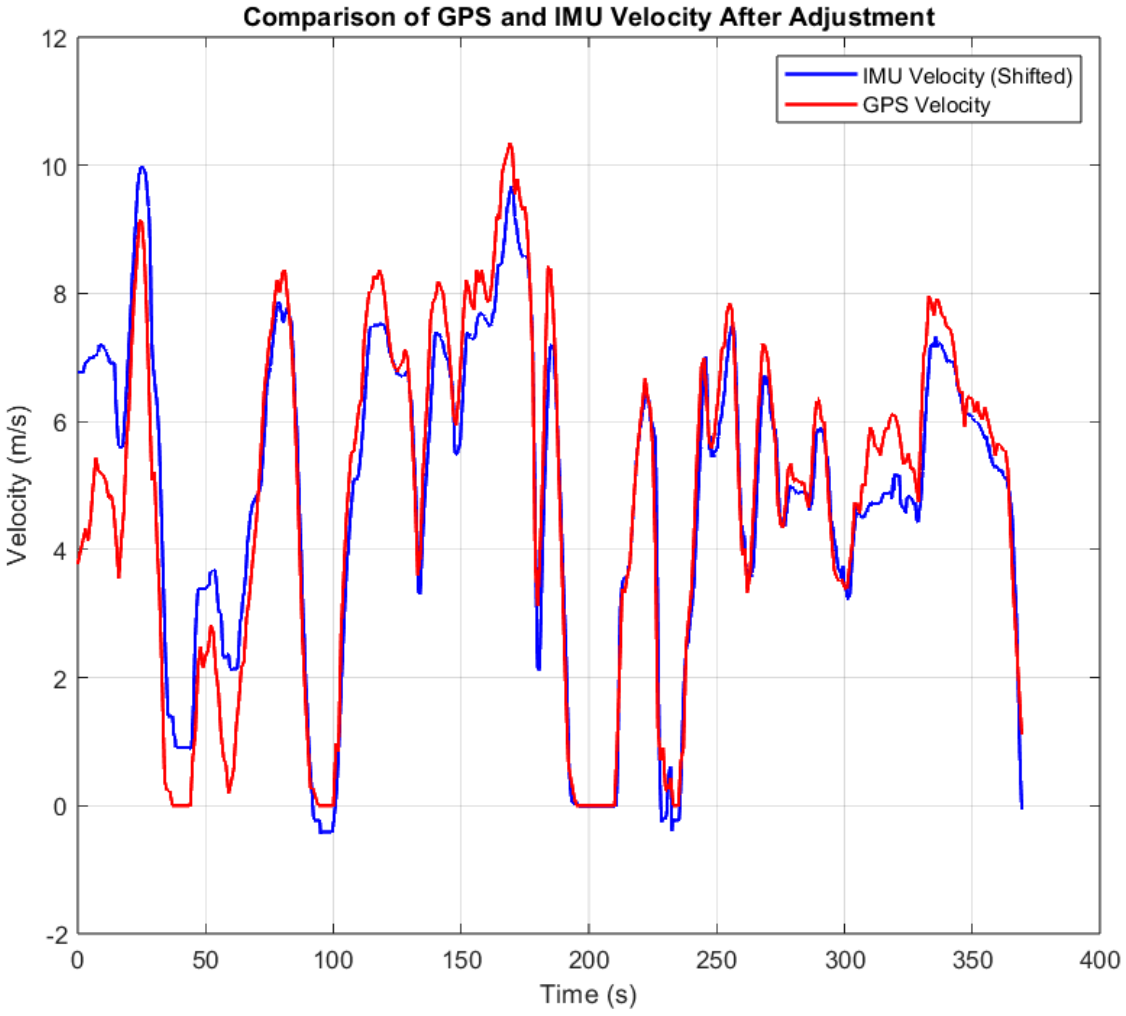


Fig 2.2.Velocity estimate from the GPS with Velocity estimate from accelerometer after adjustment

3. Dead Reckoning with IMU

Using latitude and longitude from GPS displacement between two successive coordinates is calculated using Haversine formula. The Haversine formula calculates a great-circle distance between two points on earth. Given their longitude and latitude for two successive points $(i-1)$ and i with coordinated $(lat1, lon1)$ and $(lat2, lon2)$ respectively, distance between them is calculated as,

$$lat1 = deg2rad(\text{driving\_gps\_latitude}(i-1))$$
$$lon1 = deg2rad(\text{driving\_gps\_longitude}(i-1))$$
$$lat2 = deg2rad(\text{driving\_gps\_latitude}(i))$$
$$lon2 = deg2rad(\text{driving\_gps\_longitude}(i))$$

Then the differences was calculated as,

$$\Delta lat = lat2 - lat1$$
$$\Delta lon = lon2 - lon1$$

The Haversine formula is then used, to calculate displacement between successive GPS samples

$$a = \sin^2\left(\frac{\Delta \text{lat}}{2}\right) + \cos(\text{lat1}) \cdot \cos(\text{lat2}) \cdot \sin^2\left(\frac{\Delta \text{lon}}{2}\right)$$

Where,
$$c = 2 \cdot \text{atan2}\left(\sqrt{a}, \sqrt{1-a}\right)$$

Then distance was calculated, which was then added to $\text{driving\_gps\_displacement}$ iteratively to obtain cumulative GPS-based displacement over time.

$$\text{distance} = earth\_radius \cdot c$$

The IMU displacement was obtained by integrating forward velocity .This displacement represents the distance traveled based on IMU velocity data over time.

$$\text{driving\_imu\_displacement} = cumtrapz(\text{driving\_timestamps}, \text{driving\_forward\_velocity\_corrected\_shifted})$$

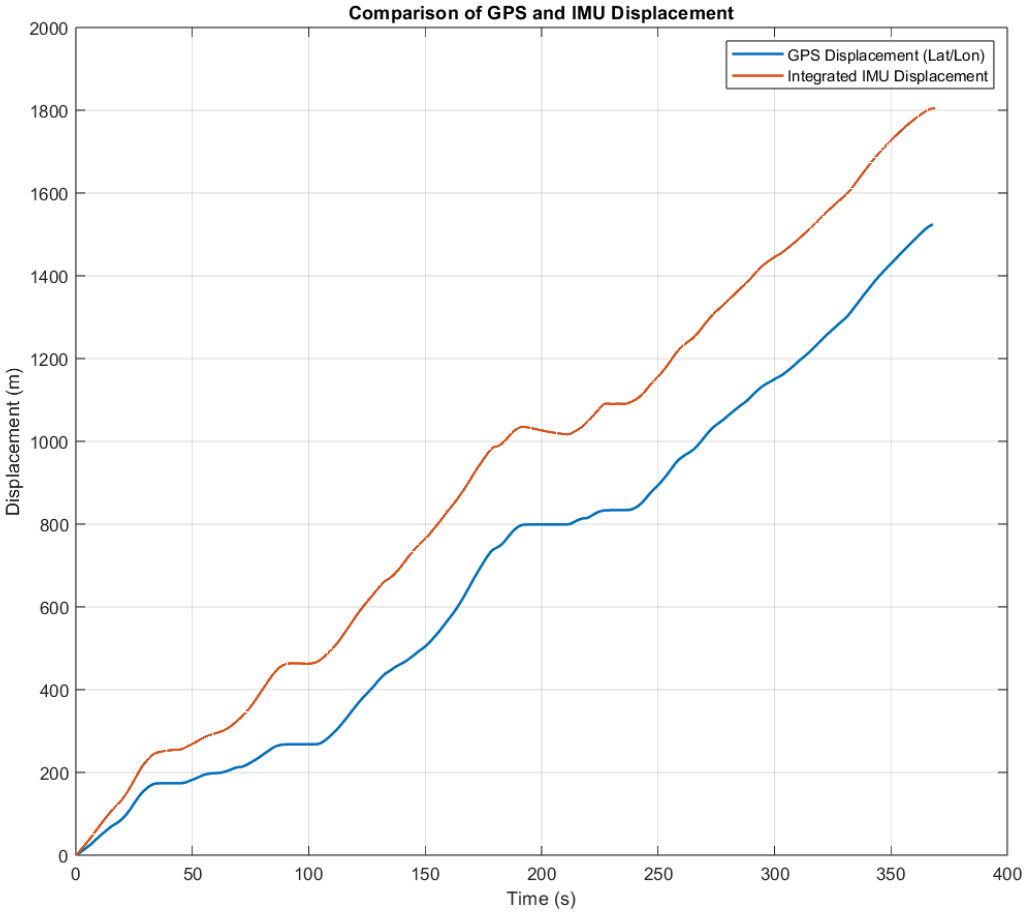


Fig 3.1 GPS vs IMU Displacement

The acceleration measured by initial sensor is give as,

$$\ddot{x}_{\text{obs}} = \ddot{X} - \omega Y - \omega^2 x_c$$
$$\ddot{y}_{\text{obs}} = \ddot{Y} + \omega \dot{X} + \dot{\omega} x_c$$

With assumptions Y = 0 (no lateral skidding) and $\mathrm{x_c}$= 0 (sensor at CM) these equations simplify to,

$$\ddot{y}_{\text{obs}} = \omega \dot{X}$$

Here, $\mathrm{driving\_timestamp\_vector}$ represents time and $\mathrm{driving\_forward\_acceleration\_corrected}$ represents the corrected forward acceleration then X can be calculated as,

$$X = \mathrm{cumtrapz}(\mathrm{driving\_timestamp\_vector}, \mathrm{driving\_forward\_acceleration\_corrected})$$

Now, multiplying X with $\mathrm{driving\_angular\_velocity\_Z}$ which is ω we get,

$$\omega \dot{X} = \mathrm{driving\_angular\_velocity\_Z} \cdot X$$

Similarly, yobs is obtained from $\mathrm{linear\_acceleration.y}$

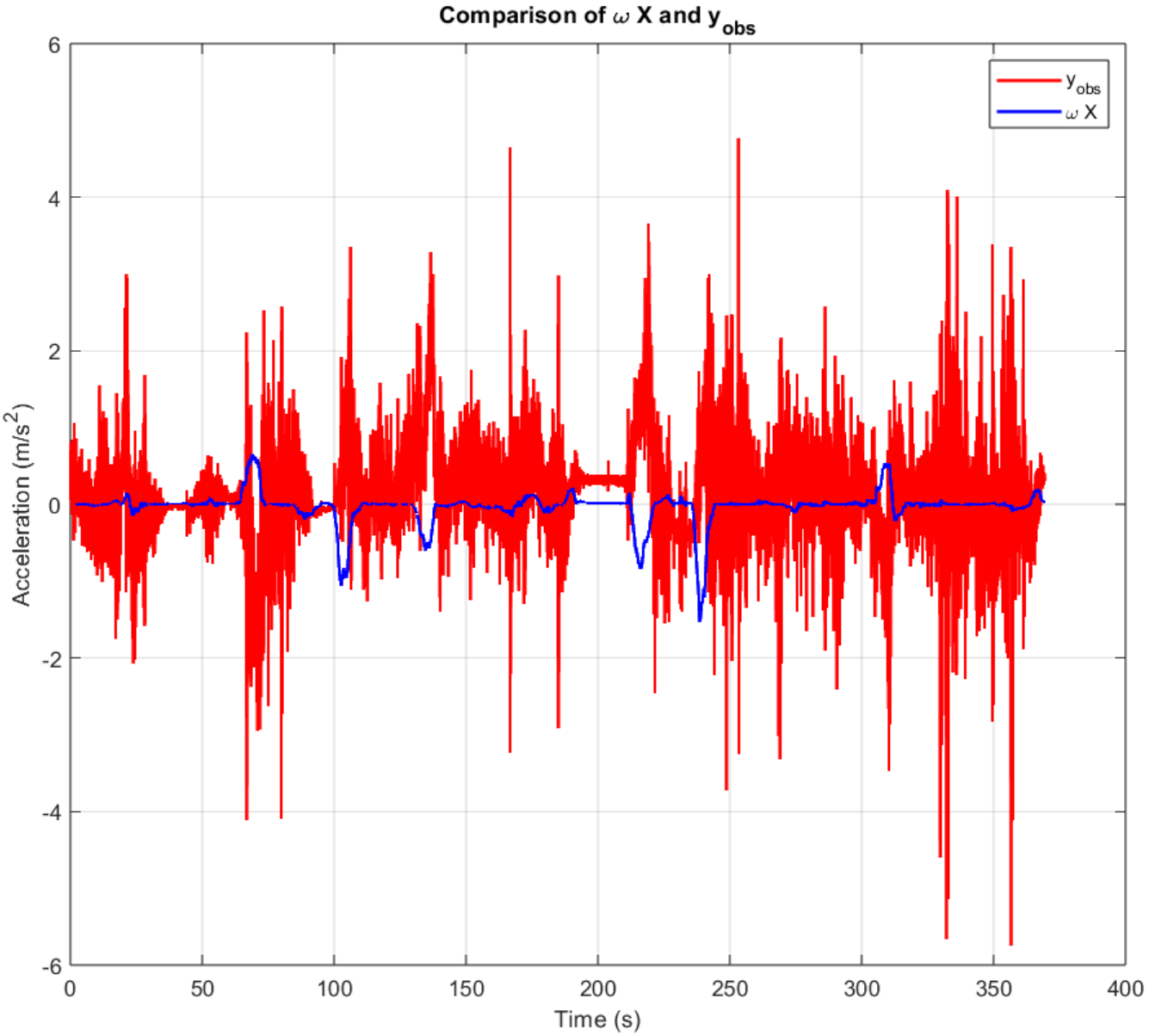


Fig 3.2 $\omega\dot{X}$ Vs yobs original

After comparison of both plots it was evident that omega X dot remains relatively stable and centered while compared to y_{obs} which shows significant noise fluctuation.
Both plots obviously don't agree as they don't align well and there is a lot of high frequency noise in y_{obs} which might be due to factors like vibrations, external environmental factors or even some bias.
To address the high frequency noise in y_{obs} a low-pass filter was applied to it as follows

$$\mathrm{sample\_rate} = \frac{1}{\mathrm{mean}(\mathrm{diff}(\mathrm{driving\_timestamp\_vector}))}$$

$$[b, a] = \mathrm{butter}(2, \mathrm{cutoff\_frequency}/(\mathrm{sample\_rate}/2))$$

$[b, a]$ is a 2nd order Butterworth Filter where cutoff frequency of **0.7** is used

Now filter is applied as,

$$y_{\mathrm{obs\_filtered}} = \mathrm{filtfilt}(b, a, \mathrm{driving\_lateral\_acceleration})$$

After applying the filter $\omega\dot{X}$ was plotted with $y_{\text{obs\_filtered}}$

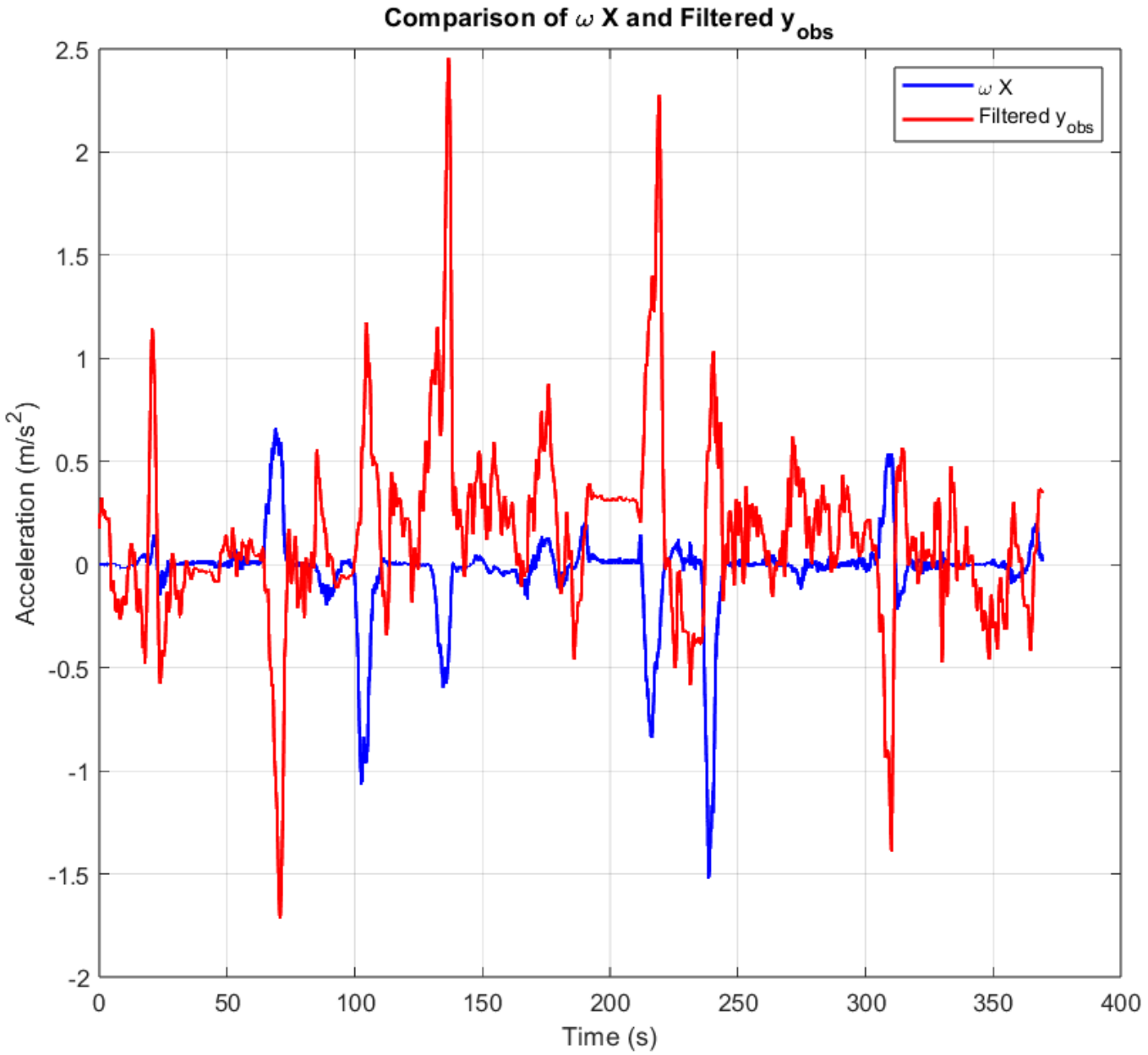


Fig 3.3 $\omega\dot{X}$ Vs yobs corrected

For the next part heading was calculated from GPS, IMU and they were plotted together.
First, initial heading was calculated using first two GPS points, then this initial GPS heading was used to calculate an offset with IMU's initial heading (driving_yaw_calibrated which was obtained after applying hard iron and soft iron correction on magnetometer data) creating an aligned yaw that matched GPS direction at start.
Now after matching the GPS and IMU direction, IMU's forward_velocity was decomposed into Easting and Northing components using aligned yaw.

$$v_e = \text{velocity} \times \cos(\text{aligned\_yaw})$$

$$v_n = \text{velocity} \times \sin(\text{aligned\_yaw})$$

Where,
v_e is the Easting Component
v_n is the Northing Component

After this further alignment of both plots was done by aligning first turn

$$\text{aligned\_yaw}[i] = \text{aligned\_yaw}[i] + \text{additional\_yaw\_offset} \quad \text{for } i \geq \text{first\_turn\_idx}$$

Where,

$$\text{turn\_threshold} = \frac{\pi}{6} \quad (30 \text{ degrees})$$

$$\text{additional\_yaw\_offset} = \text{gps\_turn\_yaw} - \text{imu\_turn\_yaw}$$

Then the estimated IMU trajectory was obtained by integrating velocity components v_e & v_n

$$x_e[i] = x_e[i-1] + v_e[i] \times \Delta t$$
$$x_n[i] = x_n[i-1] + v_n[i] \times \Delta t$$

Finally GPS and IMU paths were converted into relative coordinates to better visualize the trajectorie's shape and orientation.

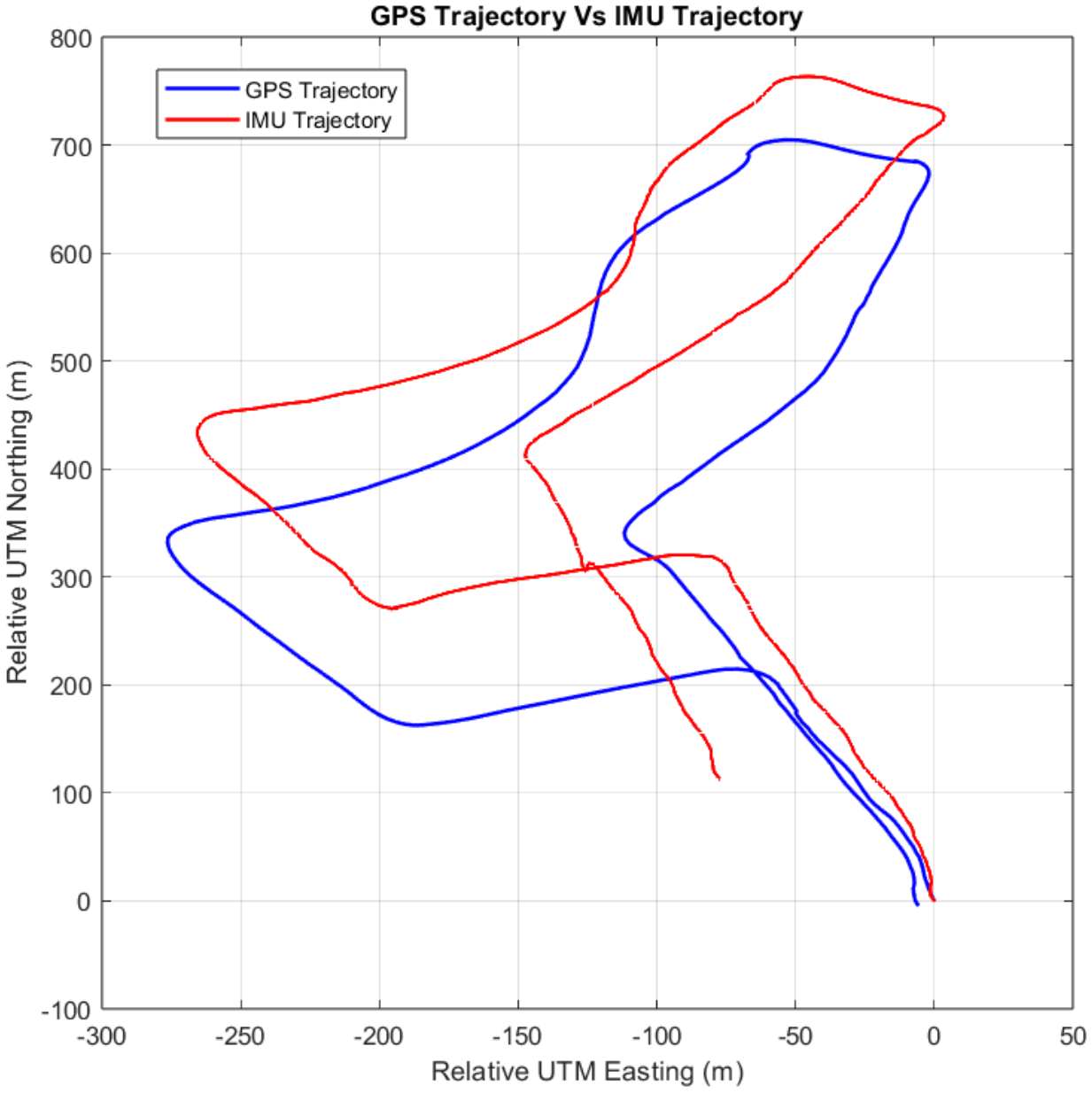


Fig 3.4 GPS Trajectory Vs IMU Trajectory