Project Report: IMU Filtering

Veera Ragav, Anthony Ruiz, Kevin Sinaga, Jing Wang

April 28, 2021

EECE 5666 Digital Signal Processing

Introduction

This paper will explore inertial mass units, or IMUs, and several techniques that have been developed to filter their output so that it can become useful data to make calculations. The paper will first discuss what an IMU is. This discussion will include its uses, the different types, and the associated components. Then the datasets that will be studied, and the mathematical models associated with the type of IMU used to gather this data will be presented. Following the presentation of the mathematical models, filtering techniques will be presented. Filtering, methods that utilized techniques taught in Northeastern University's ECE 5666 class will be implemented to clean up the data set output so that a discussion of the effectiveness of such filters can be given. As such, a discussion of the Madgwick and Kalman filter will be given, and a moving average, exponential rated moving average, and several custom lowpass filters will be implemented and evaluated in bulk of this report.

Inertial Mass Units (IMUs)

An IMU is an electronic sensor used in many devices that require or benefit from having information about orientation available. As an essential component to device and craft motion detection, IMUs are used in navigation systems to track chances in position that allows vehicles and crafts to move autonomously, or to provide information to a user that is controlling it.

Inertial Navigation Systems (INS) use IMUs to aircrafts and seacrafts such as UAVS, submarines, and large ships. They also have practical use in everyday consumer products that people use every day. They are used to change the screen orientation and for GPS in smartphones and tablets, and to track motion for fitness tracking devices and step counters. They are also used to help balance motorized personal vehicles like the Segway (What is IMU Sensor and How to use with Arduino? 2020).

Two types of IMU's exist that are widely used in the applications mentioned above today. The two types of IMU's in use are the 6 DOF (degree-of-freedom) which consists of three orthogonal accelerometers and three orthogonal gyroscopes, and the 9 DOF version which consists of three orthogonal accelerometers, three orthogonal gyroscopes, and three orthogonal magnetometers. Each of these three different types of sensors are position to measure specific variables in the 3 cartesian directions.

1. Accelerometers

Accelerometers are used to measure changes in velocity. In the context of an IMU, there are three that measure acceleration in each of the cartesian x-direction, y-direction, and z-direction and detect the position of the gravity vector (Reading a IMU WITHOUT Kalman: The Complementary Filter 2013). These three accelerometers constitute three degrees-of-freedom in an IMU. In systems reliant on IMU data, the accelerometer is used to calculate the attitude, or its orientation with respect to the world frame, of a craft.

The attitude estimation based off an accelerometer consists of using the acceleration of the device to calculate the systems rotation with respect to the world frame x-axis, y-axis, and z-axis, or the roll, pitch, and yaw, respectively if we are considering the Euler angles (ENAE788M: Class 2 Part 2 - IMU Basics, Attitude estimation using CF AND MADGWICK 2019). This angle can be seen in the picture below.

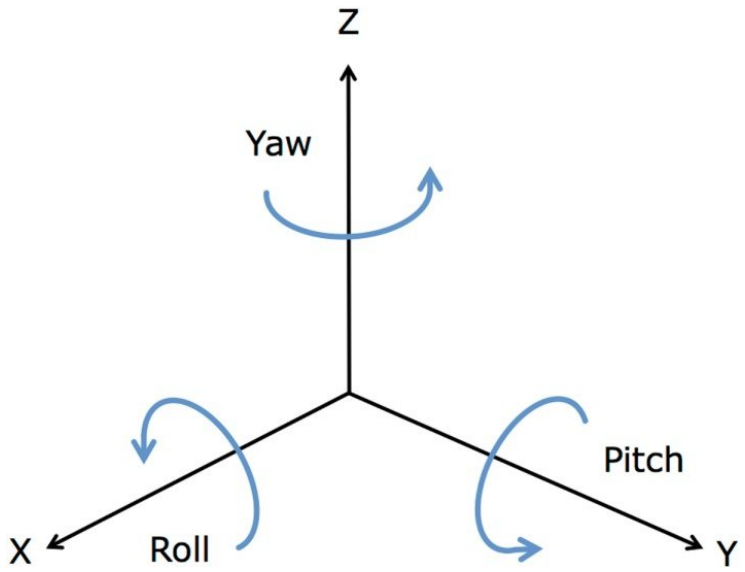

Figure 1: Euler angles - roll, pitch, and yaw (Identifying Active Travel Behaviors in Challenging Environments Using GPS, Accelerometers, and Machine Learning Algorithms)

The angles orientation of the craft can be determined using the following equations below.

$$\varphi\ (roll\ or\ angle\ from\ x-axis) = \tan^{-1}\left(\frac{a_y}{\sqrt{{a_x}^2 + {a_z}^2}}\right)$$

$$\theta\ (pitch\ or\ angle\ from\ y-axis) = \tan^{-1}\left(\frac{a_x}{\sqrt{{a_y}^2 + {a_z}^2}}\right)$$

$$\psi\ (yaw\ or\ angle\ from\ z-axis) = \tan^{-1}\left(\frac{\sqrt{{a_x}^2 + {a_y}^2}}{a_z}\right)$$

$a_x - Acceleration\ in\ the\ x - direction$

$a_y - Acceleration\ in\ the\ y - direction$

$a_z - Acceleration\ in\ the\ z - direction$

The values used in these equations are taken at any given instant (ENAE788M: Class 2 Part 2 - IMU Basics, Attitude estimation using CF AND MADGWICK 2019).

Accelerometers do suffer from some shortcomings. For example, they will always be sensitive to the acceleration due to gravity, and it impacts (attitude) calculations. This has a particularly adverse effect on the yaw calculation (What is IMU Sensor and How to use with Arduino? 2020). Some literature recommends not relying on accelerometer data to calculate yaw at all due to this fact. They are also devices that measure linear acceleration, which cannot be easily decoupled from rotational motion. Linear motion will through off readings that will be used to calculate attitude and effect the results. Lastly, accelerometers are also very sensitive devices. As such there will be affected by all forces the body they are attached is subjected to. This is of particular concern for drones that are propelled by rotors (Reading a IMU WITHOUT Kalman: The Complementary Filter 2013). This makes them less reliable for short term measurements.

2. Gyroscopes

Gyroscopes measure the rotational rate of object they are attached to. Functionally, the three gyroscopes that are used in IMU's to detect angle changes or to help an aircraft maintain a specific orientation (What is IMU Sensor and How to use with Arduino? 2020). Three gyroscopes make up and three additions degrees of freedom in an IMU and when paired with the accelerometers, make a 6 degree-of-freedom IMU.

The attitude estimation as determined by the gyro scopes is completed by integrating angular velocity overtime. To get a proper attitude estimation, there needs to be an initial angle. This is in line with the known facts of integration. The integration will give you the change in angle, but without an initial reference there is no way to properly orient the craft (ENAE788M: Class 2 Part 2 - IMU Basics, Attitude estimation using CF AND MADGWICK 2019).

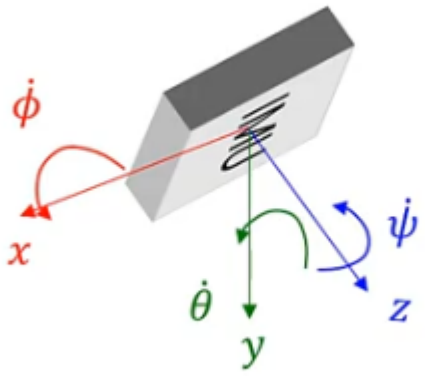

Figure 2: Angular velocities measured by gyroscopes (ENAE788M: Class 2 Part 2 - IMU Basics, Attitude estimation using CF AND MADGWICK 2019)

The equations used to determine the roll, pitch, and yaw using gyroscope data is as follows.

$$\varphi_{t+1}\ (roll) = \varphi_t + \dot{\varphi}\delta t$$

$$\theta_{t+1}\ (pitch) = \theta_t + \dot{\theta}\delta t$$

$$\psi_{t+1}\ (yaw) = \psi_t + \dot{\psi}\delta t$$

$subscript\ t+1 - current\ angle\ calculation$

$subscript\ t - initial\ angle\ calculation$

$\dot{\varphi} - angular\ velocity\ about\ the\ x - axis$

$\dot{\theta} - angular\ velocity\ about\ the\ y - axis$

$\dot{\psi} - angular\ velocity\ about\ the\ z - axis$

Unlike the accelerometers, gyroscopes are not susceptible to external forces, and thus do not need to have values compensated to account for the acceleration due to gravity. This makes them much more reliable for short term measurements (Reading a IMU WITHOUT Kalman: The Complementary Filter 2013).

Although these measurement devices do not experience the same pitfalls as accelerometers, they do suffer from some measurement errors. The main problem with determine attitude using gyroscope data is that integrating introduces drift (Reading a IMU WITHOUT Kalman: The

Complementary Filter 2013). Drift is the growing difference between the calculated attitude and the actual attitude over time. Drift compounds overtime and makes gyroscope-based attitude calculations much less reliable over time. The accelerometer data is more reliable in the long term and is preferred over gyroscope data in that aspect.

3. Magnetometer

The final three degrees-of-freedom that are used in addition to the accelerometers and the gyroscopes to create the nine degree-of-freedom IMU are the magnetometers. These devices can measure the magnetism and use it to find the orientation of a craft in the earth magnetic field (Kumar, Beginner's guide to IMU 2014). Specifically, the three magnetometers provide information the strength of the magnetic field in each of the three cartesian direction. They are used to calculate the yaw of the craft to supplement the weakness of the accelerometers' ability to do so. The following equation is used to calculate the yaw based off data provided by the magnetometers and accelerometers at any given instant where measurement occurs.

$$\psi\,(yaw) = \tan^{-1}\left(\frac{-m_y\cos\varphi + m_z\sin\varphi}{m_x\cos\theta + m_y\sin\theta\sin\varphi + m_z\sin\theta\cos\varphi}\right)$$

$$m_x = \frac{m'_x}{m}$$

$$m_y = \frac{m'_y}{m}$$

$$m_z = \frac{m'_z}{m}$$

$$m = \sqrt{{m'_x}^2 + {m'_y}^2 + {m'_z}^2}$$

$m - Absolute\ magnetic\ field\ strength$

$m'_x - Raw\ magnetic\ field\ strength\ reading\ in\ the\ x - direction$

$m'_y - Raw\ magnetic\ field\ strength\ reading\ in\ the\ y - direction$

$m'_z - Raw\ magnetic\ field\ strength\ reading\ in\ the\ z - direction$

$m_x - Normalized\ magnetic\ field\ strength\ reading\ in\ the\ x - direction$

$m_y - Normalized\ magnetic\ field\ strength\ reading\ in\ the\ y - direction$

$m_z - Normalized\ magnetic\ field\ strength\ reading\ in\ the\ z - direction$

The magnetometers are also not without their own flaws. There measurements suffer from drift as well. As such, the reading need to be mapped in such a way that when the IMU is orthogonal to a cartesian direction it only reads a magnetic field value in that corresponding direction, while every other reading is zero (Vathsangam, Magnetometer - My imu Estimation experience). The IMUs used in this project were all six degree-of-freedom devices. No further discussion of magnetometers will be featured in this report.

Datasets

Two datasets will be analyzed during this project. The first comes from a six degree-of-freedom IMU borrowed from a lab run by Northeastern University professor Hanumant Singh. The data will noise filtered out using both a moving average filter and an exponentially rated moving average filter that will be discussed later in the paper. See the unfiltered data below.

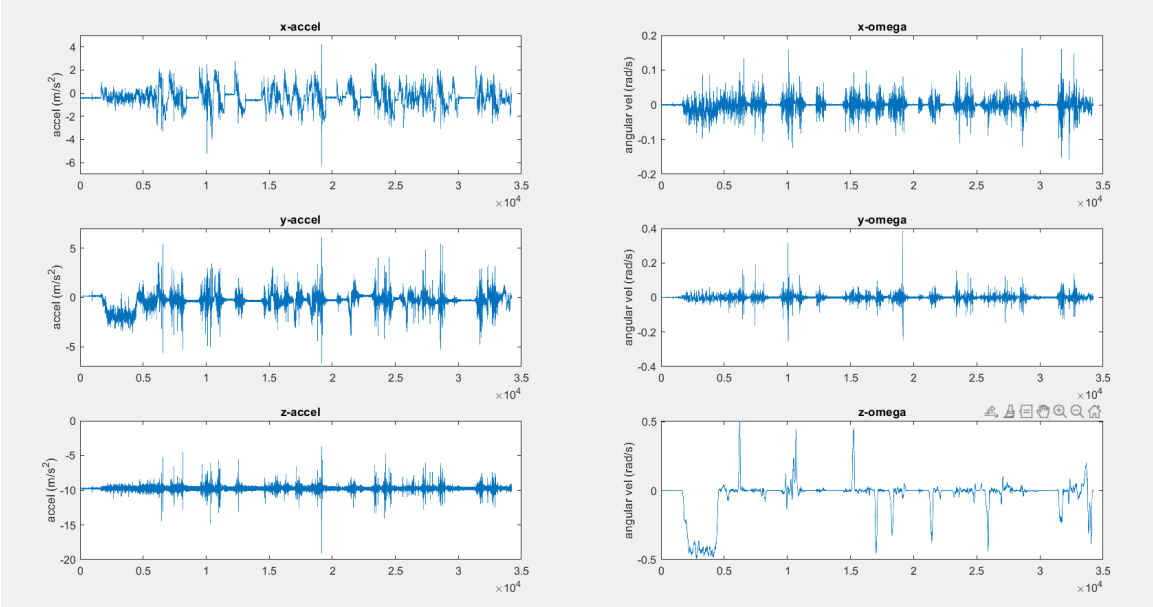

Figure 3: Raw 6-DOF IMU data from Northeastern University lab sensor

The second data set comes from a six degree-of-freedom IMU purchased by the group. Several data sets were recorded so that they can be analyzed using different low pass filters. The low pass filters applied to this data set were created using different windowing methods of various parameters to be discussed in later sections. See the unfiltered data below.

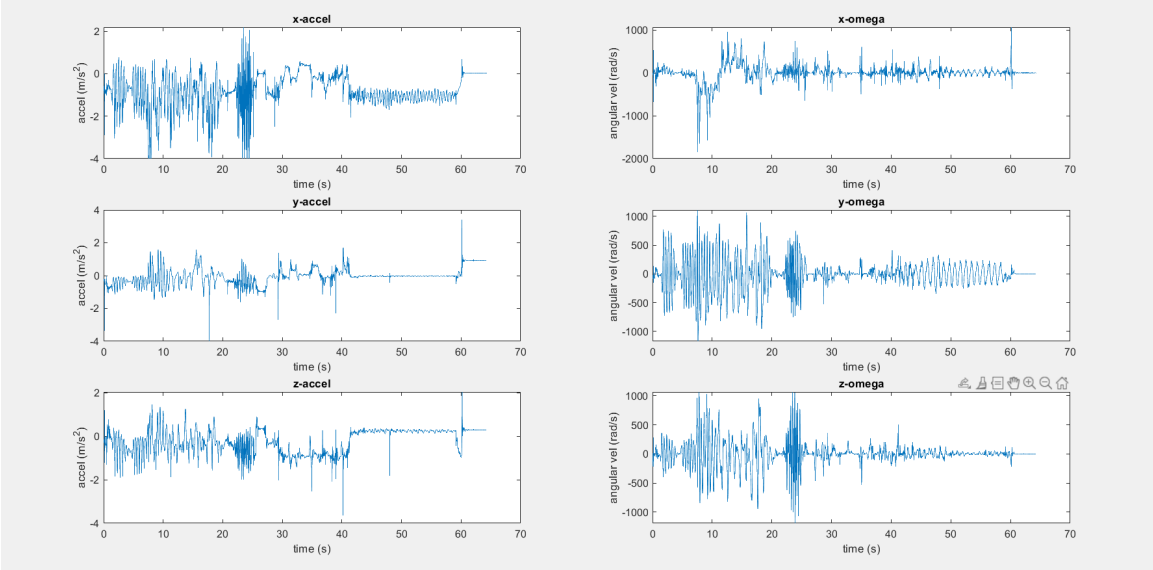

Figure 4: Raw 6-DOF IMU data from sensor purchased by project group

Mathematical Models for 6 DOF IMU Sensors

As can be seen from the data sets, the readings taken from the accelerometers and the gyroscopes do not provide clear or useful data that can be used for attitude estimation, or any other practical application. These two types of sensors suffer from bias and noise. The output of an accelerometer and gyroscope can be modeled using the two equations shown below (ENAE788M: Class 2 Part 2 - IMU Basics, Attitude estimation using CF AND MADGWICK 2019).

$$a = {}^{W}\boldsymbol{R}_{\boldsymbol{B}}^{T}(\hat{a} - g^{W}) + b_a + n_a$$

$a - Measured\ acceleration\ value\ from\ accelerometer$

$\boldsymbol{R}_{\boldsymbol{B}}^{T} - Rotation\ matrix\ from\ the\ body\ IMU\ is\ attached\ to\ the\ world\ frame$

$\hat{a} - Ideal\ acceleration\ value$

$g^{W} - acceleration\ due\ to\ gravity\ in\ world\ frame$

$b_a - bias\ associated\ with\ accelerometer$

$n_a - White\ Gaussian\ noise\ associated\ with\ accelerometer$

$$\omega = \hat{\omega} + b_g + n_g$$

$\omega - Measured\ angular\ velocity\ value\ from\ gyroscope$

$\widehat{\omega} - Ideal\ angular\ velocity\ value$

$b_a - bias\ associated\ with\ gyroscope$

$n_a - White\ Gaussian\ noise\ associated\ with\ gyroscope$

The bias and the noise are responsible for the issues identified with using the data collected by the IMU sensors mentioned in their respective sections. The noise associated with the datasets shown will be addressed using the filters implemented throughout this paper. It is influenced by external factors like temperature, pressure, and vibrations. Bias and noise can also be compensation for by leaving the device at rest and applying a Gaussian distribution to it, where the bias is addressed using the average value of the sensors at rest and noise is addressed using the variance of the measurements taken at rest (ENAE788M: Class 2 Part 2 - IMU Basics, Attitude estimation using CF AND MADGWICK 2019). This is not done to the data presented in this paper.

Common Filters in IMU

1. Madgwick Filter

One filter which is often used to filter IMU data to make attitude estimates is the Madgwick filter. It is an optimization-based filter that makes attitude change estimations based of gyroscope data. The filter also uses the accelerometer data to make attitude change estimations that optimize that of the gyroscope by correcting the drift that occurs due to integration (Colorlib, NJS).

All attitude outputs from the Madgwick filter are in quaternion space. One quaternion representation of attitude is seen below.

$$q = [q_w \ \ q_x \ \ q_y \ \ q_z]$$

$q_w -$ scalar value of quaternion

$q_x - x - direction\ value\ of\ quaternion$

$q_y - y - direction\ value\ of\ quaternion$

$q_z - z - direction\ value\ of\ quaternion$

An initial attitude must be specified for this filter to work, as it calculated incremental changes and adds it to a previous state. Below an overview of the process can be seen (Colorlib, NJS).

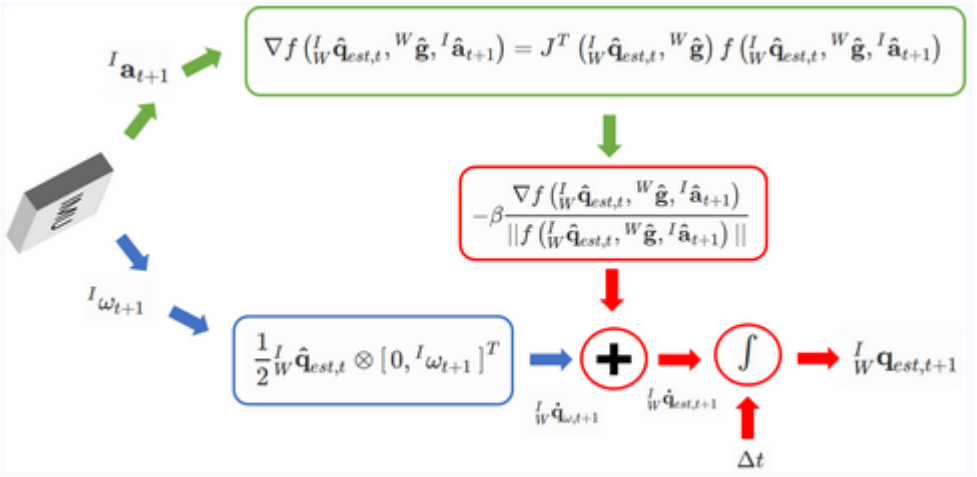

Figure 5: Madgwick filter process overview (Colorlib, NJS)

To implement this filter, the first step is to make incremental orientation change estimations using the gyroscope and accelerometer data. As previously mentioned, the estimation from accelerometer data is set up as an optimization problem. As such, it achieved by minimizing a vector that is a function of the attitude estimation quaternion, the accelerometer due to gravity, and the accelerometer data using gradient decent (ENAE788M: Class 2 Part 2 - IMU Basics, Attitude estimation using CF AND MADGWICK 2019). The mathematical model can be seen in the images below.

$$\min_{{}^{I}_{W}\hat{\mathbf{q}}\in\mathbb{R}^{4\times1}} f\left({}^{I}_{W}\hat{\mathbf{q}}, {}^{W}\hat{\mathbf{g}}, {}^{I}\hat{\mathbf{a}}\right)$$

$$f\left({}^{I}_{W}\hat{\mathbf{q}}, {}^{W}\hat{\mathbf{g}}, {}^{I}\hat{\mathbf{a}}\right) = {}^{I}_{W}\hat{\mathbf{q}}^{*} \otimes {}^{W}\hat{\mathbf{g}} \otimes {}^{I}_{W}\hat{\mathbf{q}} - {}^{I}\hat{\mathbf{a}}$$

Figure 6: Mathematical model used to make accelerometer orientation increment calculations for Madgwick Filter (Colorlib, NJS)

$$\nabla f\left({}^{I}_{W}\hat{\mathbf{q}}_{est,t}, {}^{W}\hat{\mathbf{g}}, {}^{I}\hat{\mathbf{a}}_{t+1}\right) = J^{T}\left({}^{I}_{W}\hat{\mathbf{q}}_{est,t}, {}^{W}\hat{\mathbf{g}}\right) f\left({}^{I}_{W}\hat{\mathbf{q}}_{est,t}, {}^{W}\hat{\mathbf{g}}, {}^{I}\hat{\mathbf{a}}_{t+1}\right)$$

$$f\left({}^{I}_{W}\hat{\mathbf{q}}_{est,t}, {}^{W}\hat{\mathbf{g}}, {}^{I}\hat{\mathbf{a}}_{t+1}\right) = \begin{bmatrix} 2\left(q_2 q_4 - q_1 q_3\right) - a_x \\ 2\left(q_1 q_2 + q_3 q_4\right) - a_y \\ 2\left(\frac{1}{2} - q_2^2 - q_3^2\right) - a_z \end{bmatrix}$$

$$J\left({}^{I}_{W}\hat{\mathbf{q}}_{est,t}, {}^{W}\hat{\mathbf{g}}\right) = \begin{bmatrix} -2q_3 & 2q_4 & -2q_1 & 2q_2 \\ 2q_2 & 2q_1 & 2q_4 & 2q_3 \\ 0 & -4q_2 & -4q_3 & 0 \end{bmatrix}$$

Figure 7: Matrices needed to calculate gradient of accelerometer data for Madgwick filter (Colorlib, NJS)

$${}^{I}_{W}\mathbf{q}_{\nabla,t+1} = -\beta \frac{\nabla f\left({}^{I}_{W}\hat{\mathbf{q}}_{est,t}, {}^{W}\hat{\mathbf{g}}, {}^{I}\hat{\mathbf{a}}_{t+1}\right)}{\left\| f\left({}^{I}_{W}\hat{\mathbf{q}}_{est,t}, {}^{W}\hat{\mathbf{g}}, {}^{I}\hat{\mathbf{a}}_{t+1}\right) \right\|}$$

Figure 8: Accelerometer orientation increment for Madgwick filter (Colorlib, NJS)

Here beta is a tunable parameter that increased with the error associated with the gyroscope error, since the accelerometer estimate is used as a correction for the gyroscope (ENAE788M: Class 2 Part 2 - IMU Basics, Attitude estimation using CF AND MADGWICK 2019).

The gyroscope orientation incremental change estimation is determined by adding the angular velocity to the current attitude quaternion, where the formula is given below.

$${}^{I}_{W}\dot{\mathbf{q}}_{\omega,t+1} = \frac{1}{2}\,{}^{I}_{W}\hat{\mathbf{q}}_{est,t} \otimes \left[0, {}^{I}\omega_{t+1}\right]^{T}$$

Figure 9: Gyroscope orientation increment for Madgwick filter (Colorlib, NJS)

The final step to get the attitude estimation using the Madgwick filter is to combine the two estimations based off the accelerometer and gyroscope, multiply it by the time increment, and then add it to the old attitude estimation. This process can be seen in the figure below.

$$
{}^{I}_{W}\dot{\mathbf{q}}_{est,t+1} = {}^{I}_{W}\dot{\mathbf{q}}_{\omega,t+1} + {}^{I}_{W}\mathbf{q}_{\nabla,t+1}
$$

$$
{}^{I}_{W}\mathbf{q}_{est,t+1} = {}^{I}_{W}\hat{\mathbf{q}}_{est,t} + {}^{I}_{W}\dot{\mathbf{q}}_{est,t+1}\Delta t
$$

Figure 10: Attitude estimation calculation for Madgwick filter (Colorlib, NJS)

It is also important to note that the Madgwick filter does not directly have a way to compensate for bias or noise. Thus, the sensor data must have a Gaussian distribution applied to it in the fashion mentioned in the previous section. Further reading can be found in corresponding section at the end of this paper.

2. Kalman Filter

The Kalman filter is a probabilistic filter that is also used to make attitude estimation based off IMU data. All attitude estimation made this filter is output in the state space representation. It is one of the more difficult filters to understand due to its use of probabilistic modeling. It is also difficult implement on certain devices (Reading a IMU WITHOUT Kalman: The Complementary Filter 2013). Despite that fact it is a powerful technique that requires little computation power and can compensate for noise within its design, making it an attractive filter to use.

Attitude estimation made using this filter is determined in two steps. First, by making a prediction about state of the craft given an initial state. This is the “Prediction” step. Then, the of attitude of the craft is estimated based off the measurements taken. Finally, an estimation of the craft attitude is given by comparing the two values and consider the system noise as a possible source of error (Scherr, Lommatsch, Hooson, & Kaiser, Underwater Navigation System). This is the “Correction/Update” step. As consecutive estimations are made, the previous estimate becomes the “initial state”.

The figure below highlights the process described in more detail.

Prediction:

Predicted state estimate	$\hat{\boldsymbol{x}}_k^- = \boldsymbol{F}\hat{\boldsymbol{x}}_{k-1}^+ + \boldsymbol{B}\boldsymbol{u}_{k-1}$
Predicted error covariance	$\boldsymbol{P}_k^- = \boldsymbol{F}\boldsymbol{P}_{k-1}^+\boldsymbol{F}^T + \boldsymbol{Q}$

Update:

Measurement residual	$\tilde{\boldsymbol{y}}_k = z_k - \boldsymbol{H}\hat{\boldsymbol{x}}_k^-$
Kalman gain	$\boldsymbol{K}_k = \boldsymbol{P}_k^-\boldsymbol{H}^T\left(\boldsymbol{R} + \boldsymbol{H}\boldsymbol{P}_k^-\boldsymbol{H}^T\right)^{-1}$
Updated state estimate	$\hat{\boldsymbol{x}}_k^+ = \hat{\boldsymbol{x}}_k^- + \boldsymbol{K}_k\tilde{\boldsymbol{y}}$
Updated error covariance	$\boldsymbol{P}_k^+ = (\boldsymbol{I} - \boldsymbol{K}_k\boldsymbol{H})\boldsymbol{P}_k^-$

Figure 11: Process for Kalman filter to make attitude estimation (Kim & Bang, Introduction to Kalman filter and its applications 2018)

It is important to note that the Caret symbol (^) as used here indicates an estimate. Subscript k indicates the current state at hand, and k-1 is the previous state. The superscripts + and – indicate the estimate made in the "Update" or "Prediction" states, respectively.

In terms of matrices and vectors, B is the control-input matrix, and u is the control vector. The former is used to define linear equations for needed control factors and the latter is the magnitude of the control system's, or user's control of the movement (Scherr, Lommatsch, Hooson, & Kaiser, Underwater Navigation System).

The matrix F is the state transition matrix, and H is the measure/observation matrix. The former used to make obtain prediction values, the latter is used to obtain a measurement value (Scherr, Lommatsch, Hooson, & Kaiser, Underwater Navigation System).

The two matrices, Q and R, are covariance matrices that are associated with the process of controlling the craft and with sensor measurement respectively (Kim & Bang, Introduction to Kalman filter and its applications 2018). These two values impact the predicted and updated P (state covariance) values.

The vector z, known as the measurement vector, indicates the relationship between the estimate and the measurement and is defined using the equation.

$$z_k = Hx_k + \nu_k$$

Figure 12: Measurement vector used in Kalman filter (Kim & Bang, Introduction to Kalman filter and its applications 2018)

It is used to adjust the attitude estimation based off prediction and measurement to give the estimation that is output during any specific filter iteration. In this equation, v is the measurement noise vector.

Despite being a powerful filter, the Kalman filter is not without limitations. It is based off the assumption that measurements are linear, or able to be expressed in terms of the matrices F, B, and H, and that the noise measurements can be models as Gaussian noise (Kim & Bang, Introduction to Kalman filter and its applications 2018). As much the model is no longer useful if these conditions are not met.

Complementary Filters System Implement

In the real filtering process, three filters are used. Two lowpass filters remove the noise and one complementary filter combines the accelerometer and gyro meter data.

Data Source

Three types of the dataset are from 1. Parrot Mambo quadcopter, 2. offered by group members, 3. LSM9DS1 IMU embedded ARDUINO NANO 33 BLE SENSE. PuTTY is used to log the signal.

In the complementary filters system, LSM9DS1 IMU is the main dataset source. It has 3 acceleration channels, 3 angular rate channels, 3 magnetic field channels and has a linear acceleration full scale of ±2g/±4g/±8/±16 g, a magnetic field full scale of ±4/±8/±12/±16 gauss and an angular rate of ±245/±500/±2000 dps. The acceleration and angular rate sampling frequency is 119 Hz, when magnetic sampling frequency is 20 Hz. (NV, 2015)

Only accelerometer and gyro meter data are used in complementary filters system.

Low pass Filter Design

The lowpass filter is the filter which passes the low frequency signal but attenuates the higher frequency signal. In the IMU filtering process, the lowpass filter is designer to reject the noise signal that has high frequency.

There are 4 FIR lowpass filters designed to meet the requirements.

1. FIR Filters overview

The FIR filters are always stable, have linear phase delay, easy to design, but need more calculation time.

1.1 Stability Performance

The FIR filters are always stable as a non-feedback system or open loop system, which only have poles in the origin and never in Right Half Plane (RHP).

The continuous transfer function in rational function form is like:

$$H(s) = \frac{N(s)}{D(s)} = K\frac{(s-z_1)(s-z_2)(s-z_3)\dots(s-z_n)}{1} \quad (1)$$

where z_n are the zeros in system, K is a constant. The p_n, the poles in transform system, are not showing in the FIR rational function form because there is only one pole in origin in FIR system.

However, the filters are not the continuous system but the discrete system. The discretization needs to apply in the filter. Part of the system delay is from sampling, which is

$$z^{-1} = e^{-sT_s} \tag{2}$$

where T_s is the sampling interval and z^{-1} is the a single-sample delay. Inappropriate sampling interval chosen changes the properties of filters.

1.2 Delay of FIR Filter

One important property of FIR filter is its linear phase delay or constant group delay. The single-sample delay exists in all discrete system, may harming the stability but not influence the phase. The phase response returns the phase shift added to the signal, making the signal distortion if it does not meet requirements (Manolakis & Vinay, 2011). When we simple s as jw, $z^{-1} = e^{-j\omega}$, w as the frequency, the system transform function can replace as $H\left(e^{j\omega}\right)$ and the phase response is $\angle H\left(e^{j\omega}\right)$. Then the phase delay is defined by

$$\tau_{pd}(\omega) = -\frac{\angle H\left(e^{j\omega}\right)}{\omega} \tag{3}$$

The group delay is defined by

$$\tau_{gd}(\omega) = -\frac{d(\psi(\omega))}{d\omega} \tag{4}$$

where $\psi(\omega)$ is the unwrapped phase response. In FIR filter, the phase response is always linear,

$$\angle H\left(e^{j\omega}\right) = \theta_0 - \omega n_d \tag{5}$$

Then the group delay can be easily removed by constant parameter n_d.

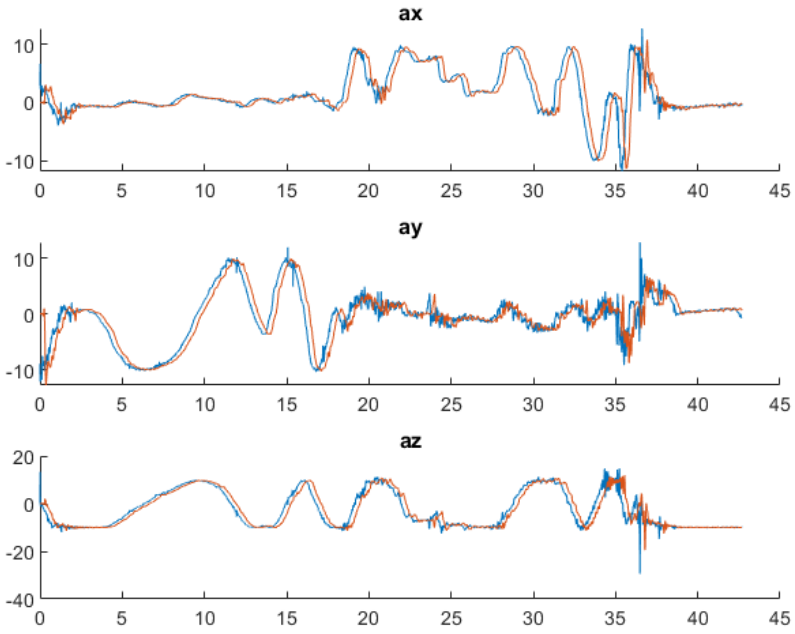

Figure 133: IMU data with group delay

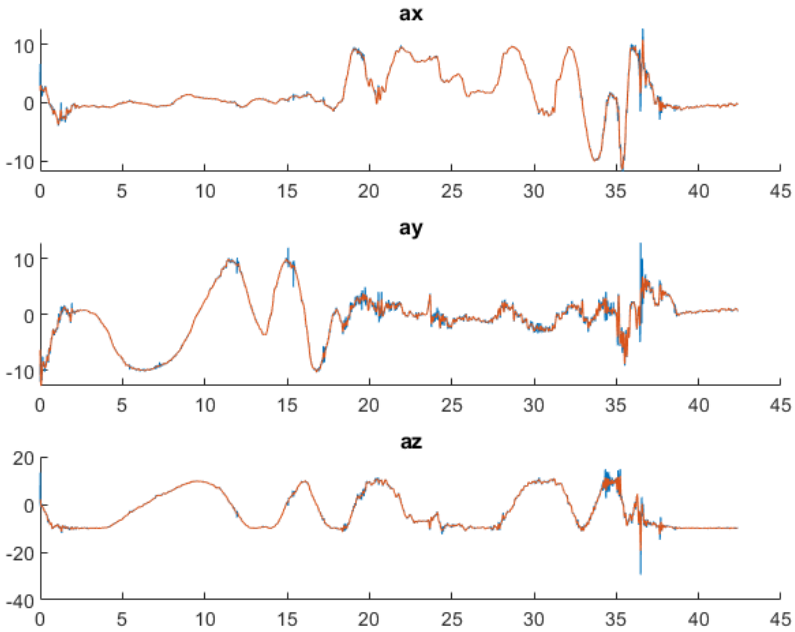

Figure 14: IMU data with group delay removed

MATLAB DSP toolbox provides a convenient function to calculate the filter group delay, which is $grpdelay()$. For Kaiser Window, the group delay n_d will be the half of the filter order. In .mat file, the first n_d filtered data sequence will be removed to cut the delay.

```
%% Cut Delay
fdelay  = round(mean(grpdelay(fir,fs,length(at))));
att = at(1:end-fdelay);

axx = ax(1:end-fdelay);
fx1_1 = fx1;
fx1_1(1:fdelay) = [];

ayy = ay(1:end-fdelay);
fy1_1 = fy1;
fy1_1(1:fdelay) = [];

azz = az(1:end-fdelay);
fz1_1 = fz1;
fz1_1(1:fdelay) = [];
```

Figure 15: Remove the group delay, MATLAB code sample

1.3 Filter Design Method

The main design method in complementary filter system is Window Design Method, with sub design method is Frequency Sampling Method. Comparing with other window function, the Kaiser window (Manolakis & Vinay, 2011) provides a great performance in main lobe, which maximizes the ratio of energy concentration in $\omega = 0$, defined by

$$w[n] = \begin{cases} \dfrac{I_0\left[\beta\sqrt{1-\left[\dfrac{n-\alpha}{\alpha}\right]^2}\right]}{I_0(\beta)}, & 0 \le n \le M \\ 0 & otherwise \end{cases} \tag{6}$$

$$I_0(x) = 1 + \sum_{m=1}^{\infty} \frac{\left(\frac{x}{2}\right)^m}{m!} \tag{7}$$

$$\beta = \begin{cases} 0, & A < 21 \\ 0.5842(A-21)^{0.4} + 0.07886(A-21), & 21 \le A \le 50 \\ 0.1102(A-8.7), & A > 50 \end{cases} \tag{8}$$

All the FIR filters of this section are Kaiser window.

Frequency Sampling Method is also used to design process. It is almost impossible to design a good filter in all environments. The Frequency Sampling Method clearly provides the

main motion frequency range and main noise frequency range, helping designer to determine the cutoff frequency.

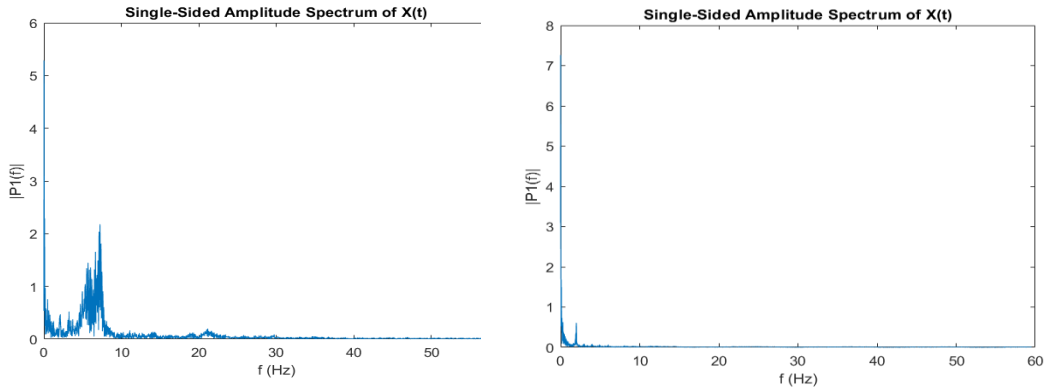

Figure 16: Frequency Spectrum of different motion. Extreme shaking motion(left), slowly rotation(right)

The appropriate ideal cutoff frequency should be 10 Hz in the left figure. For right figure, the cutoff frequency needs discussing. Four different bandwidth FIR lowpass filters are designed in complementary filters system,

Name	Order	Wp	Ws	Ap	As
trfir1	40	3	12	1	50
tfir2	40	6	15	1	50
tfir1	74	7.2	12	1	50
trfir2	118	3	6	1	50

Table 1: FIR lowpass filters parameters

The ripples and transition bandwidth performance will be discussed in the next section.

2. IIR Filters Design

The IIR filters are easy to calculate, have lower order than FIR filters with similar performance, but may be unstable and have non-linear phase delay.

2.1 Stability Performance

Different with the FIR filters, the IIR Filters may be unstable because of the poles can locate in RHP. The continues transform function is

$$H(s) = \frac{N(s)}{D(s)} = K\frac{(s-z_1)(s-z_2)(s-z_3)\dots(s-z_n)}{(s-p_1)(s-p_2)(s-p_3)\dots(s-p_m)} \quad (9)$$

A more frequent function form is cascade form, which is

$$H(z) = KH_1(z)H_2(z)H_1(z)\dots H_n(z) \quad (10)$$

when

$$H_n(z) = \frac{b_0 + b_1z^{-1} + b_2z^{-2}}{1 + a_1z^{-1} + a_2z^{-2}} \quad (11)$$

2.2 Delay of IIR Filter

A disadvantage of IIR Filter is that its phase delay is not linear, which bring challenges to eliminate the delay distortion. A zero-phase delay on real time IIR filter is almost impossible. However, for static dataset, MATLAB provides a function $filtfilt()$ to remove the phase delay. The function filters data twice: reversing the filtered sequence and filtering it again. The output will be zero phase distortion and the IIR order will also be double, needing additional calculation. (L., 2018)

The first filtering process can be written by

$$X(e^{-j\omega})H(e^{-j\omega}) \quad (12)$$

when $H(e^{j\omega})$ is the filter Fourier Transform in frequency domain, $X(e^{j\omega})$ is the input data's Fourier Transform. The second filtering process can be written by

$$X(e^{-j\omega})H(e^{-j\omega})H(e^{j\omega}) \quad (13)$$

The final output spectrum is

$$Y(e^{j\omega}) = X(e^{j\omega})H(e^{j\omega})H(e^{-j\omega}) = X(e^{j\omega})|H(e^{j\omega})|^2 \quad (14)$$

Then the phase response $\angle H(e^{j\omega})$ is removed.

```
%% filtfilt
iir = tiir1;

fx1 = filtfilt(iir.sosMatrix,iir.ScaleValues,ax);
fy1 = filtfilt(iir.sosMatrix,iir.ScaleValues,ay);
fz1 = filtfilt(iir.sosMatrix,iir.ScaleValues,az);
```

Figure 14:fltfilt() function, MATLAB code sample

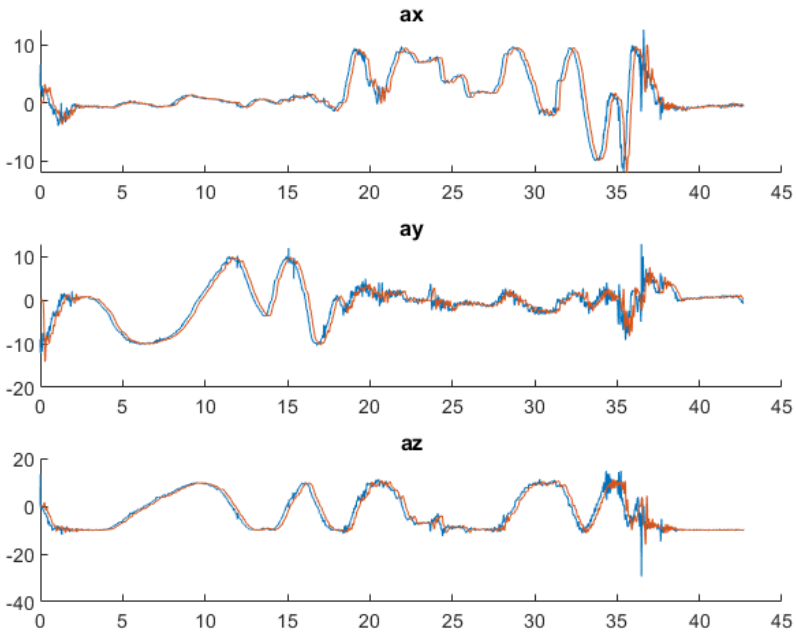

Figure 17: IMU data with group delay

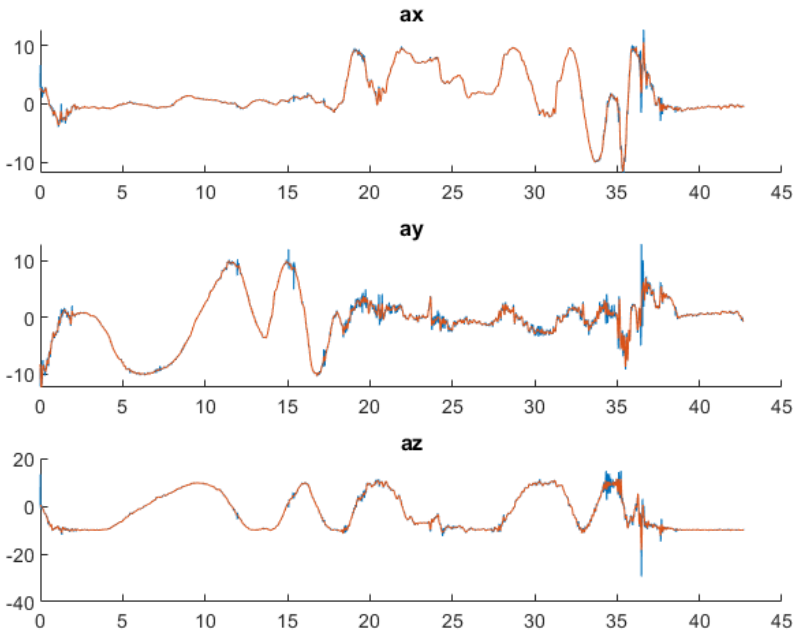

Figure 18: IMU data filtered by filtfilt()

2.3 IIR Filter Design Method

The IIR filter is a Butterworth filter. The Butterworth filter has best flat performance but need high orders to have the similar performance than other IIR filters. The main mission of IIR Filter is comparing the performance with other FIR filters.

1	2	1	1	-1.7207214924291	0.931210878614598
1	2	1	1	-1.6112496268410	0.808347731582239
1	2	1	1	-1.5182418440638	0.703962656667262
1	2	1	1	-1.4403487968081	0.616541246002903
1	2	1	1	-1.3762384416891	0.544588512348784
1	2	1	1	-1.3247029562275	0.486748884846014
1	2	1	1	-1.2847128606923	0.441866951381356
1	2	1	1	-1.2554404734849	0.409013783180312
1	2	1	1	-1.2362670370289	0.387494932380040
1	2	1	1	-1.2267823841447	0.376850057593284

Table 2: IIR SOS matrix

0.0526	0.0493	0.0464	0.0440	0.0421	0.0405	0.0393	0.0384	0.0378	0.0375	1

Table 3: IIR scale value matrix

Complementary Filter Design

The complementary filter is used to combine data from gyro meter and accelerometer. The angle data calculated from gyro meter will be integral twice. The bias instability and angular random walk (RAW) will results drift over time. The angle velocity data from accelerometer will be disturb by noise because of the sensitivity. Thus, the gyroscopes, measuring the angular velocity, will have accumulated error in long term, but are accurate in short term, and the accelerometer data is only reliable on the long term. The combination can eliminate both errors. The lowpass complementary filter equation is

$$y[n] = (1 - \alpha)x[n] + \alpha y[n - 1] \tag{15}$$

$$\alpha = \frac{\tau}{\tau + T_s} \tag{16}$$

when τ is the desired time constant, T_s is the sampling interval.

In IMU, $x[n]$ is the angle data calculated by gyro meter reading, $y[n - 1]$ is the angle data calculated by accelerometer reading.

```
tau = 0.1;  % desired time constant
alpha = tau/(tau+T);
c_ang = (1-alpha)*gyro_ang+alpha*accel_ang;
```

Figure 19: Complementary filter, MATLAB code sample

Filter Implementation Result

1. Lowpass Filter Result

The lowpass filters eliminate the influence of noise or disturbance in the complementary filter system. Both the accelerometer and gyro meter signal need to be filtered. Five lowpass filters are designed to filtered sequence: four FIR filters and one IIR filters. The filter choice should depend on the working environment.

1.1 Pendulum Motion Test

In pendulum motion testing, the target IMU will collect the pendulum motion data, which is periodic.

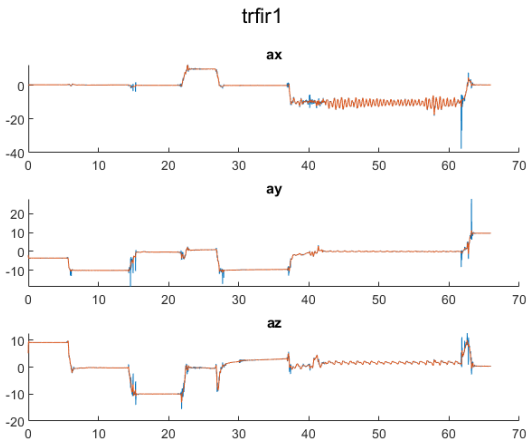

Figure 20: acceleration filter by trfir1

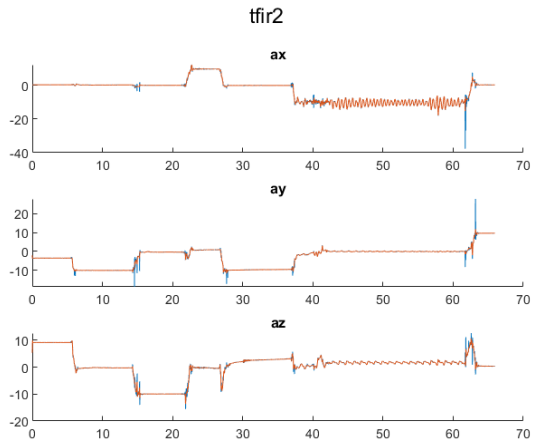

Figure 21: acceleration filter by tfir2

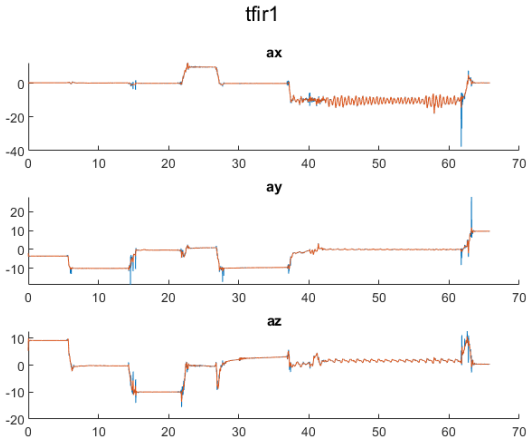

Figure 22: acceleration filter by tfir1

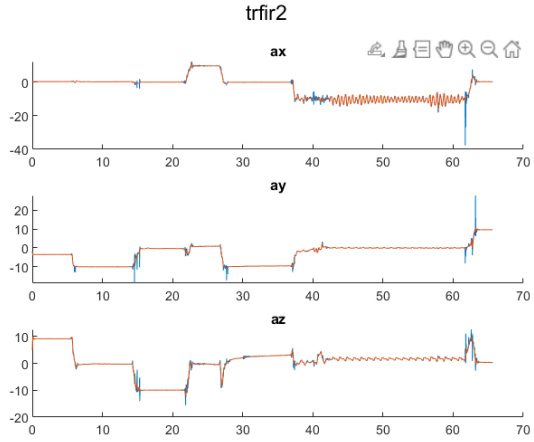

Figure 23: acceleration filter by trfir2

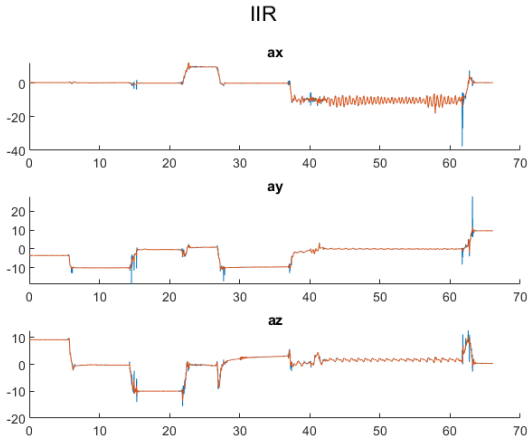

Figure 24: acceleration filter by IIR

The result shows all the filters have the good performance during pendulum motion. They successfully tracked the high frequency motion, eliminated the noise during motionless time and rejected the intense signal.

	Trfir1	Tfir1	Tfir2	Trfir2	IIR
Avg. Time(s)	0.484	0.490	0.490	0.489	0.537
Order	40	40	74	118	40

Table 4: Running time and order

The running time contains the following process: reading the log file, creating filter function, filtering the data. The reading file took the 80% time. Except the IIR filter, the other filters have similar performance in filtering speed.

1.2 Slowly Rotation Test

The second test is slowly rotation, which aim to test the performance in rest.

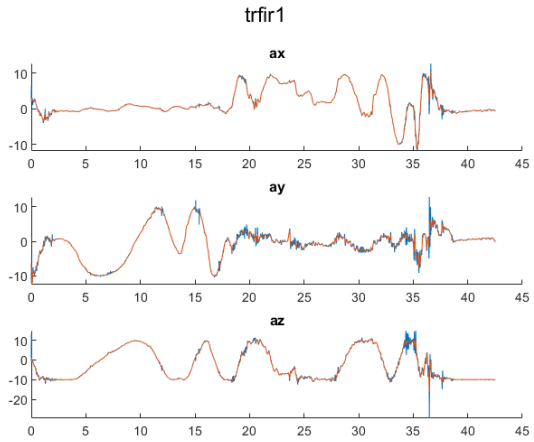

Figure 25: acceleration filter by trfir1

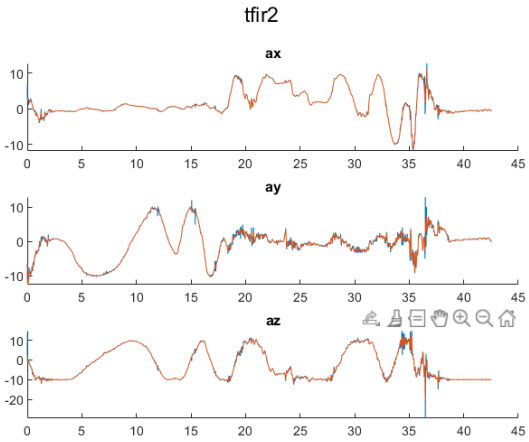

Figure 26: acceleration filter by tfir2

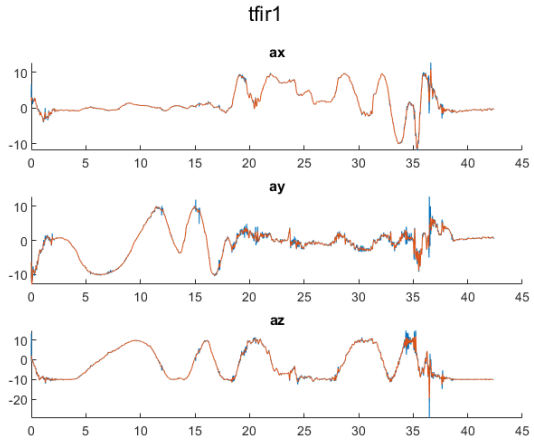

Figure 27: acceleration filter by tfir1

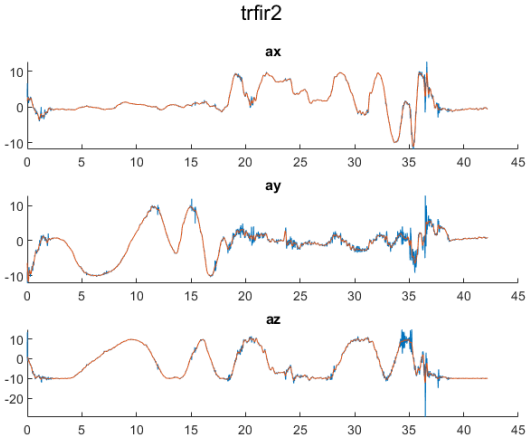

Figure 28: acceleration filter by trfir2

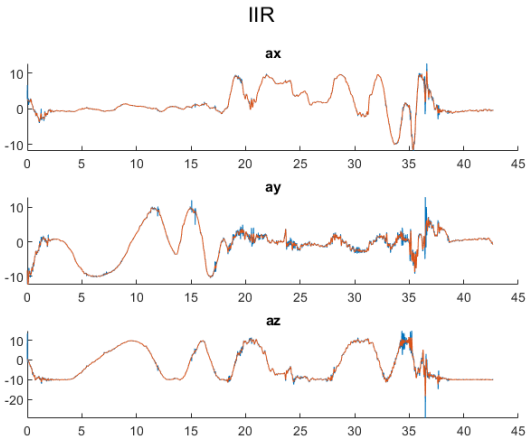

Figure 29: acceleration filter by IIR

According to the test result, the different filters have the different filtering performance. The trfir2 FIR filter, with $\omega_p = 3\ Hz, \omega_s = 6\ Hz$ and 118 order, get the smoothest plot. One of the reasons is that the cutoff frequency of trfir2 is the lowest, losing track in 6-8 Hz.

Another useful point is that trfir1 has better flat performance than tfir2, even they have the same bandwidth and similar attenuating performance. The mainlobe width difference is the reason. The trfir1's mainlobe width is 0.21094 in normalized frequency, when tfir2's mainlobe width is 0.30859 in normalized frequency. When the unit changes to Hz, it means trfir1 has lower cutoff frequency and removes more high frequency signal.

1.3 Extreme Motion Test

In extreme motion test, tester quickly shake the IMU to collect extreme environment data. As the Single-Sided Amplitude Spectrum (Fig.4), the sub-main frequency is 6-7 Hz.

The result shows the $\omega_p \omega_s$ has the most importance influence when the main motion frequency is near to the cutoff frequency. The trfir2 filter, whose $\omega_p = 3\ Hz, \omega_s = 6\ Hz$ has the worst tracking performance in shake motion, almost losing all high frequency signal data. The secondly worse filter is tfir2, whose $\omega_p = 3\ Hz, \omega_s = 12\ Hz$. It lost half of high frequency signal data.

The tfir1 has the best tracing, whose $\omega_p = 7.2\ Hz, \omega_s = 12\ Hz$, because the sub-main frequency is 6-7 Hz.

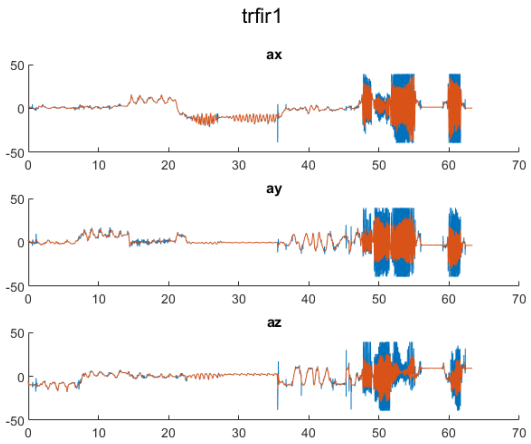

Figure 30: acceleration filter by trfir1

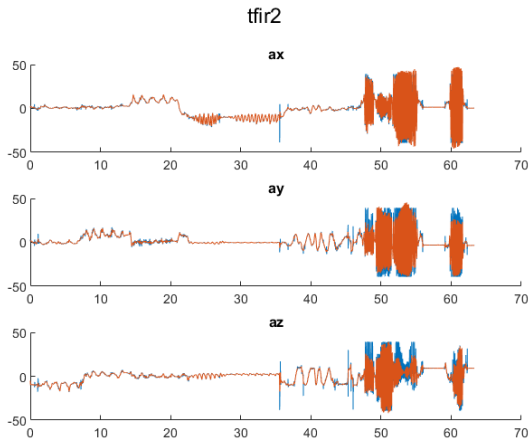

Figure 31: acceleration filter by tfir2

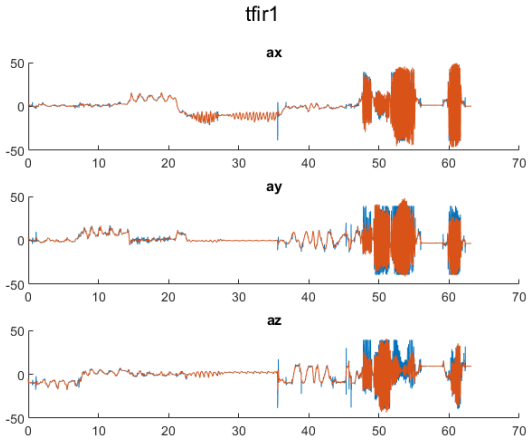

Figure 32: acceleration filter by tfir1

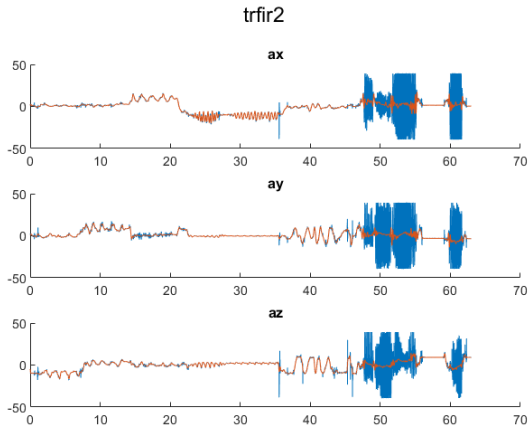

Figure 33: acceleration filter by trfir2

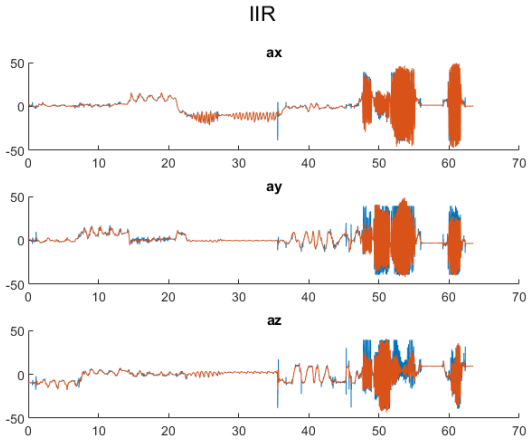

Figure 34: acceleration filter by IIR

1.4 Conclusion

In the low-cost sampling environment, the ripples and transition bandwidth are the most important designing factor. The best ripples and transition bandwidth needs to determine by the

actual motion type. The attenuating frequency should not lower than the motion frequency. Otherwise, the filters will lose the signal tracking. For example, in mini quadcopter, the users want to collect the high frequency data because the wind will move the quadcopter easily. Also, the pass frequency shall not far away the main motion frequency. Otherwise, the filters will not remove the disturbance, noise, and impossible motion data from the original data, like heavy vehicles.

2. Complementary Filter Result

The complementary filter is used to combine data from gyro meter and accelerometer. The gyro meter data will have shift over the time, and the accelerometer data contains the disturbance because of the sensitivity.

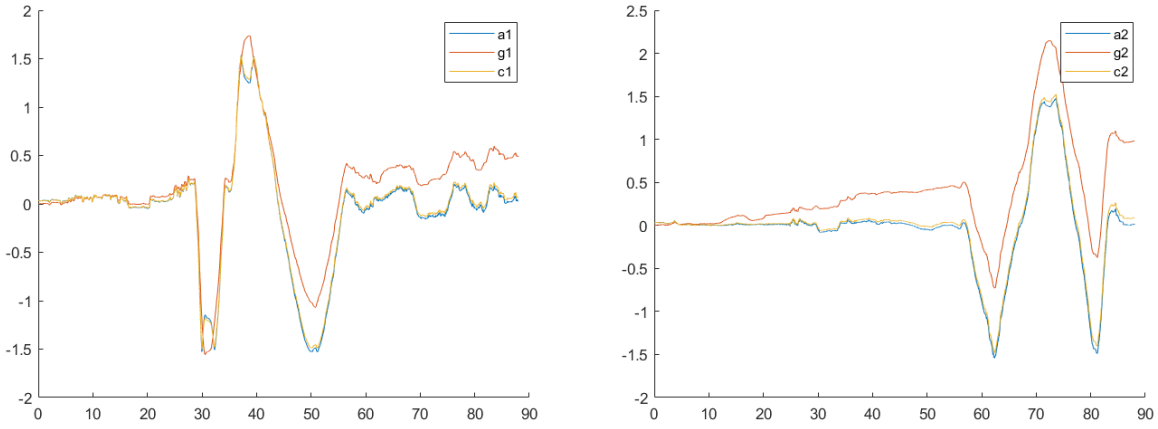

Figure 35: Angle calculated by accelerometer, gyro meter and complementary filter

The angle calculated by gyro meter has the extreme shift over the time. When combining gyro meter and accelerometer data, the shift is successfully eliminated. Also, the noise form accelerometer data will be eliminated because the gyro data is smooth.

The shift dramatically increases Because the test IMU is cheap. Additional step is to set gyro meter returning to 0 when the IMU is not moving.

Simple Moving Average Filter Implementation

This section deals about designing a moving an average filter to filter data from gyroscope. It is assumed that we are interested only in the low frequency data from the gyroscope and hence all the high frequency data are considered as noise. We examine the usage of moving average filter to filter out this high frequency information.

In statistics, a moving average is a calculation to analyze data points by creating a series of averages of different subsets of the full data set. It is also called a moving mean (MM) or rolling mean and is a type of finite impulse response filter. There are different variations of this filter. They are simple, and cumulative, or weighted forms. In this section we will see about simple moving average filter.

Development of the "moving average" dates to 1901, although the name was applied to it later. This technique for smoothing data points was used for decades before this, or any general term, came into use. In 1909 G. U. Yule (Journal of the Royal Statistical Society, 72, 721-730) described the "instantaneous averages" R. H. Hooker calculated in 1901 as "moving-averages." Yule did not adopt the term in his textbook, but it entered circulation through W. I. King's Elements of Statistical Method (1912)

Definition

The moving average filter is a simple Low Pass FIR (Finite Impulse Response) filter commonly used for smoothing an array of sampled data/signal. It takes L samples of input at a time and takes the average of those L-samples and produces a single output point. It is a very simple LPF (Low Pass Filter) structure that comes handy for scientists and engineers to filter unwanted noisy component from the intended data.

As the filter length increases (the parameter L) the smoothness of the output increases, whereas the sharp transitions in the data are made increasingly blunt. This implies that this filter has excellent time domain response but a poor frequency response.

The MA filter performs three important functions:

1) It takes L input points, computes the average of those L-points and produces a single output point

2) Due to the computation/calculations involved, the filter introduces a definite amount of delay

3) The filter acts as a Low Pass Filter (with poor frequency domain response and a good time domain response).

The difference equation for a L-point discrete-time moving average filter with input represented by the vector $\mathbf{x}$ and the averaged output vector $\mathbf{y}$, is

$$y[n] = \left(\frac{1}{L}\right) \sum_{k=0}^{L-1} x[n-k]$$

For example, a 5-point Moving Average FIR filter takes the current and previous four samples of input and calculates the average.

$$y[n] = \frac{1}{5}(x[n] + x[n-1] + x[n-2] + x[n-3] + x[n-4])$$
$$y[n] = 0.2(x[n] + x[n-1] + x[n-2] + x[n-3] + x[n-4])$$

Transfer function

A LTI system is completely characterized by its impulse response h[n] or equivalently the Z-transform of the impulse response H(z)) which is called the *transfer function*. The transfer function describes the input-output relationship of the system and for the L-point Moving Average filter. The transfer function can be obtained by taking Z-transform of the filter described by the difference equation.

$$Z^{-1}\{\, y[n] \,\} = Z^{-1}\left\{\frac{1}{L}\sum_{k=0}^{L-1} x[n-k]\right\}$$

$$Y(z) = \left(\frac{1}{L}\sum_{k=0}^{L-1} z^{-k}\right) X(z)$$

$$\frac{Y(z)}{X(z)} = \left(\frac{1}{L}\sum_{k=0}^{L-1} z^{-k}\right)$$

$$H(z) = \left(\frac{1}{L}\sum_{k=0}^{L-1} z^{-k}\right)$$

The transfer function can be represented in rational form, i.e.,

$$H(z) = \frac{b_0 + b_1 z^{-1} + b_2 z^{-2} + \ldots + b_n z^{-n}}{a_0 + a_1 z^{-1} + a_2 z^{-2} + \ldots + b_n z^{-m}}$$

Where,

$$b = [b_0, b_1, \ldots, b_n] = [1/L, 1/L, \ldots, 1/L]$$

$$a = [a_0, a_1, \ldots, a_m] = [1]$$

$$H(z) = \frac{(1/L) + (1/L)z^{-1} + (1/L)z^{-2} + \ldots + (1/L)z^{-n}}{1}$$

The transfer function can be visualized using magnitude plot and phase plot. For L=10,

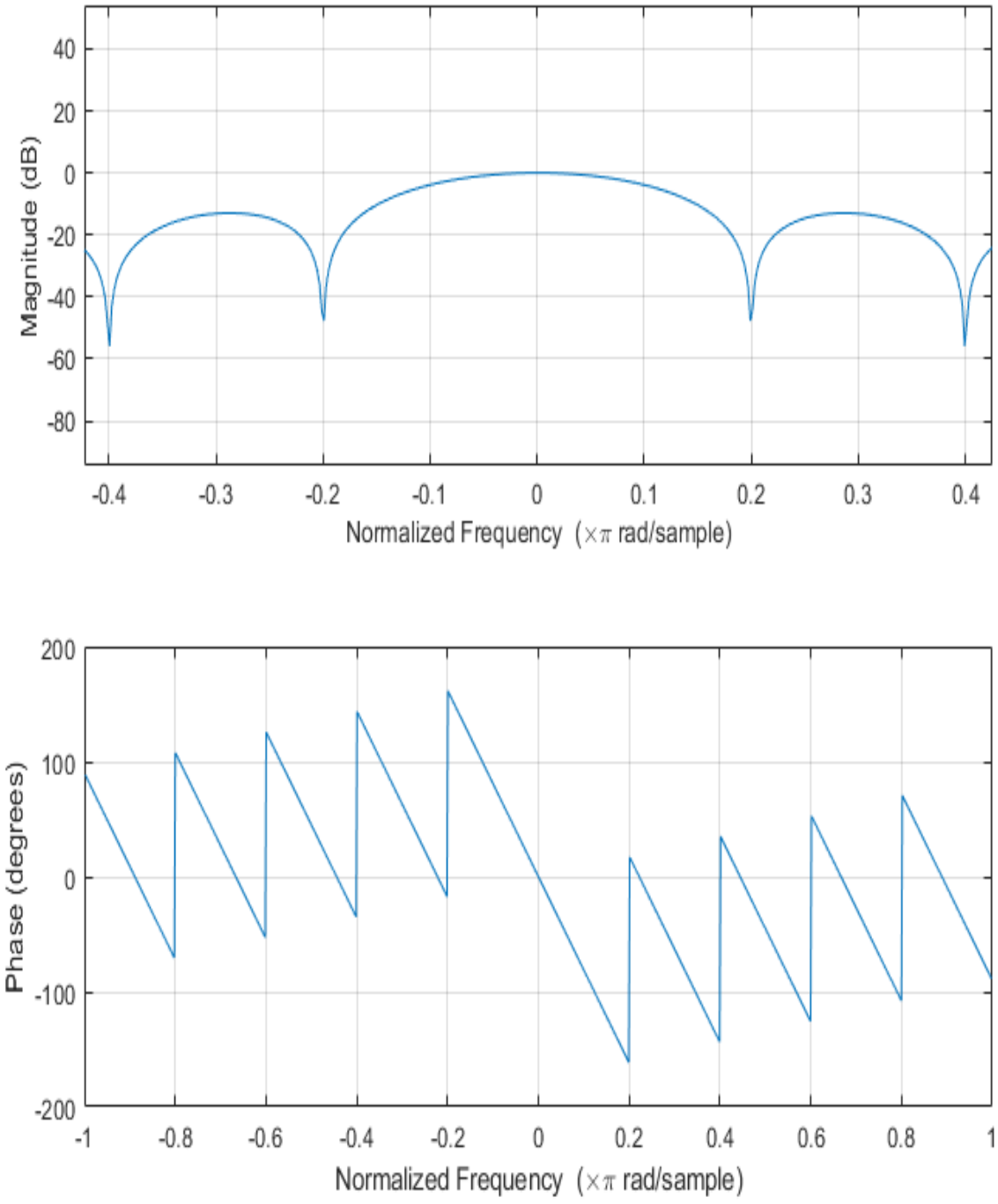

Fig.36: Magnitude(decibels) and phase response (code: appendix B1)

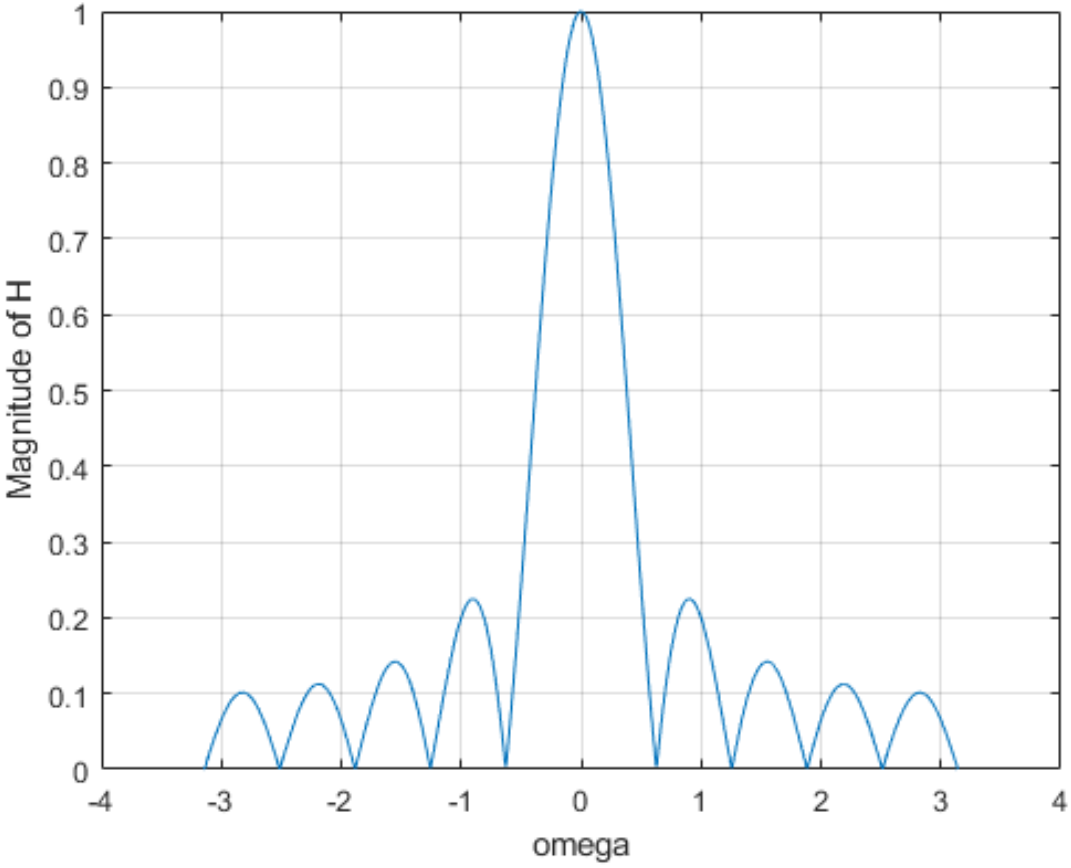

Fig.37 : Magnitude plot of H(z) (code: appendix B2)

The following observations can from the magnitude plot.

1. The main lobe extends from $\omega = -0.626$ to $\omega = 0.626$ rad/sample. The stopband edge is $\omega_s = 0.626$ rad/sample
2. The stop band attenuation is $A_s = 13\ dB$ (It shows that this filter is a poor low pass filter)
3. It has linear phase in the pass band.

The Fig.38 shows the magnitude plot for different values of L.

Poles and Zeros

In this section, we will find the poles and zeros of the simple moving average filter.

$$H(z) = \left(\frac{1}{L}\sum_{k=0}^{L-1} z^{-k}\right)$$

$$H(z) = \left(\frac{1}{L}\sum_{k=0}^{L-1}\frac{1}{z^k}\right)$$

$$H(z) = \frac{1}{L}\,\frac{1 - z^{-L}}{1 - z^{-1}}$$

$$H(z) = \frac{1}{L}\,\frac{z^L - 1}{z^{L-1}(z - 1)}$$

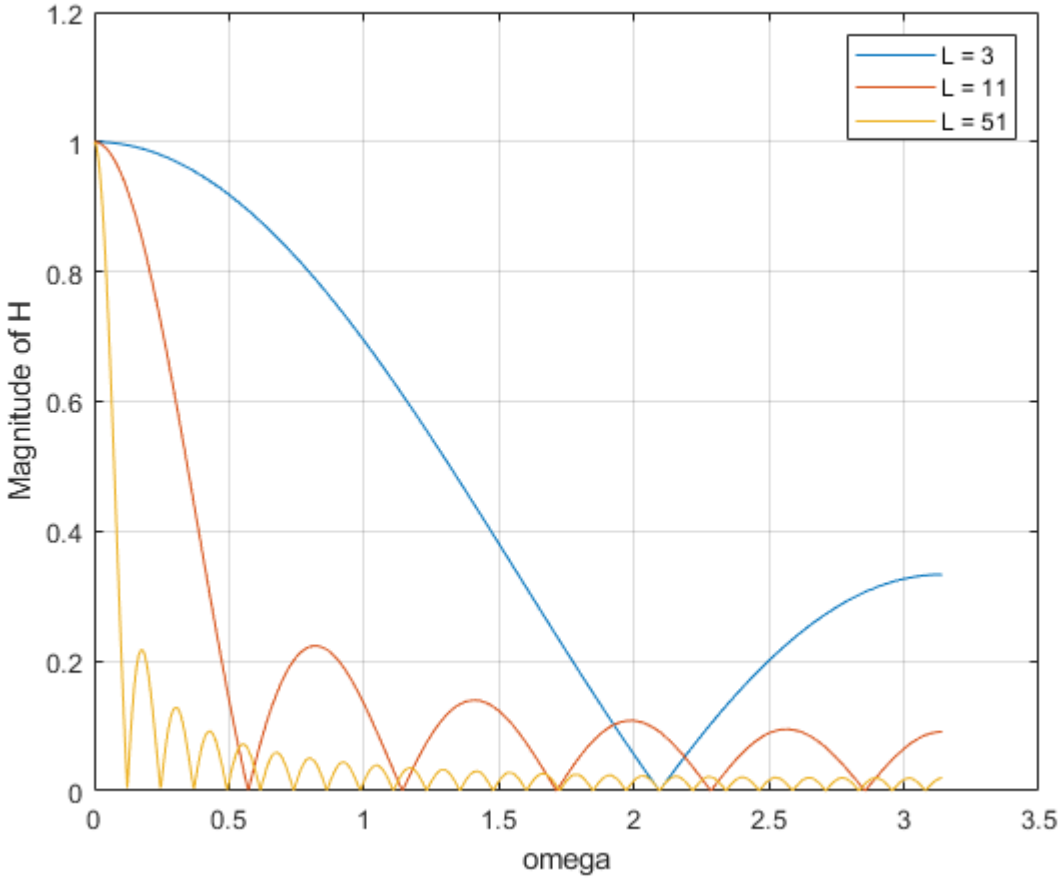

Fig.38: Magnitude response for different L (code: appendix B3)

The roots of the numerator are the solutions of $z^L - 1 = 0$ or $z^L = 1$. Note that this equation has L solutions, namely all L-th roots of unity. One of the solutions is $z = 1$, but there are L−1 complex solutions as well, evenly spaced along the unit circle $e^{-2\pi jk/L}$ for k ϵ [1, L-1].

The roots of the denominator are the solutions of $z^{L-1}(z-1)=0$. This has just two distinct solutions: $z=0$ with a multiplicity of L-1 and $z=1$ with a multiplicity of one. Since both the numerator and denominator have the root $z=1$ in common, we must factor out the common factor $(z-1)$. This leaves us with the following L-1 poles and zeros:

- Zeros = $\left\{e^{-2\pi jk/L}\middle|\mathrm{k}\in[1,\mathrm{L}-1]\right\}$
- Poles = $\{0\}$

This result can be verified graphically from the Fig.39.

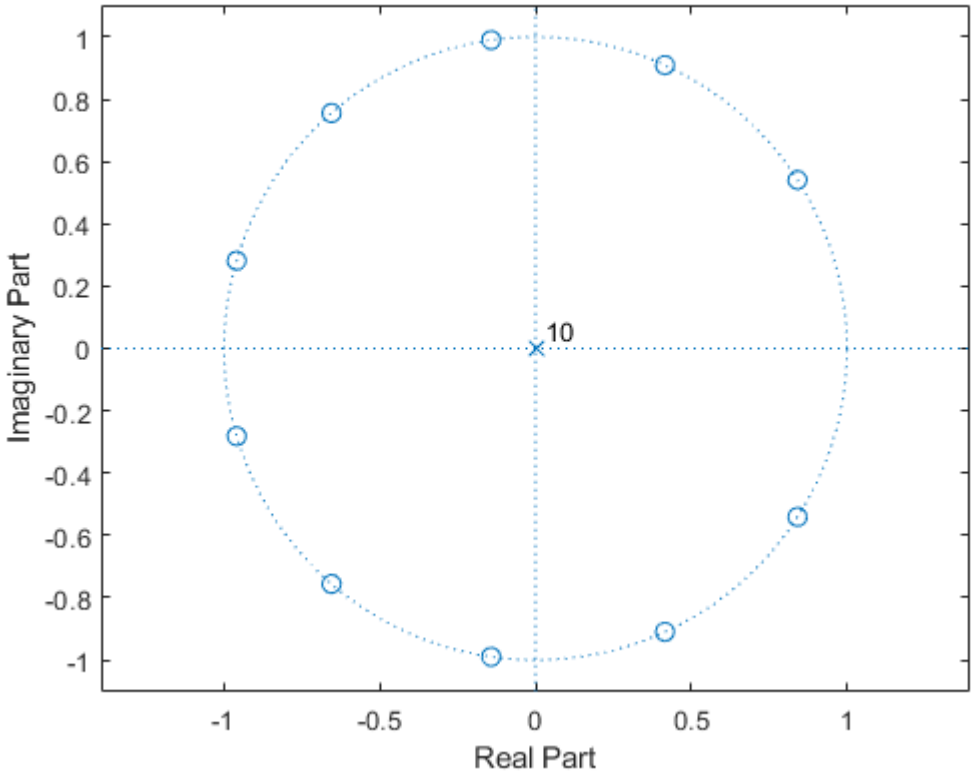

Fig.39: Pole-Zero Plot for SMA with L = 11 (code: appendix B4)

Impulse Response

The impulse response is the output of the filter when a Kronecker delta function is applied to the input. The Kronecker delta function is defined as, $\delta[n]=\begin{cases}1, & n=0\\0, & n\neq 0\end{cases}$

The impulse response of the SMA filter is derived as follows,

$$y[n] = \left(\frac{1}{L}\right)\sum_{k=0}^{L-1} x[n-k]$$

$$y_{impulse}[n] = h[n]$$

$$y_{impulse}[n] = \left(\frac{1}{L}\right)\sum_{k=0}^{L-1} \delta[n-k]$$

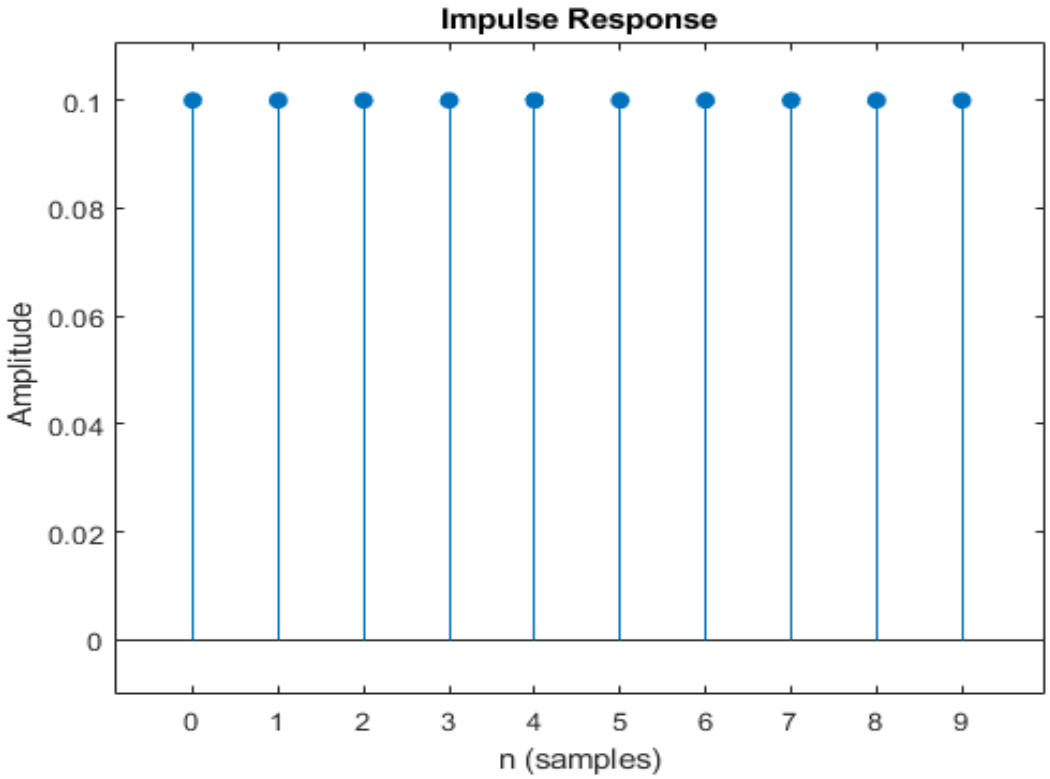

Fig.40 Impulse response of SMA with L = 10 (code: appendix 5)

$$y_{impulse}[n] = \begin{cases} 1/L, & 0 \leq n < 0 \\ 0, & otherwise \end{cases}$$

For example, Fig. 40 shows the impulse response of SMA with L = 10.

This filter has finite impulse response and hence the system is stable.

Step Response

The step response is the output of the filter when a Heaviside step function is applied to the input. The Heaviside step function is defined as,The step response is the output of the filter when a Heaviside step function is applied to the input. The Heaviside step function is defined as,

$$H[n] = \begin{cases} 0, & n < 0 \\ 1, & n \geq 0 \end{cases}$$

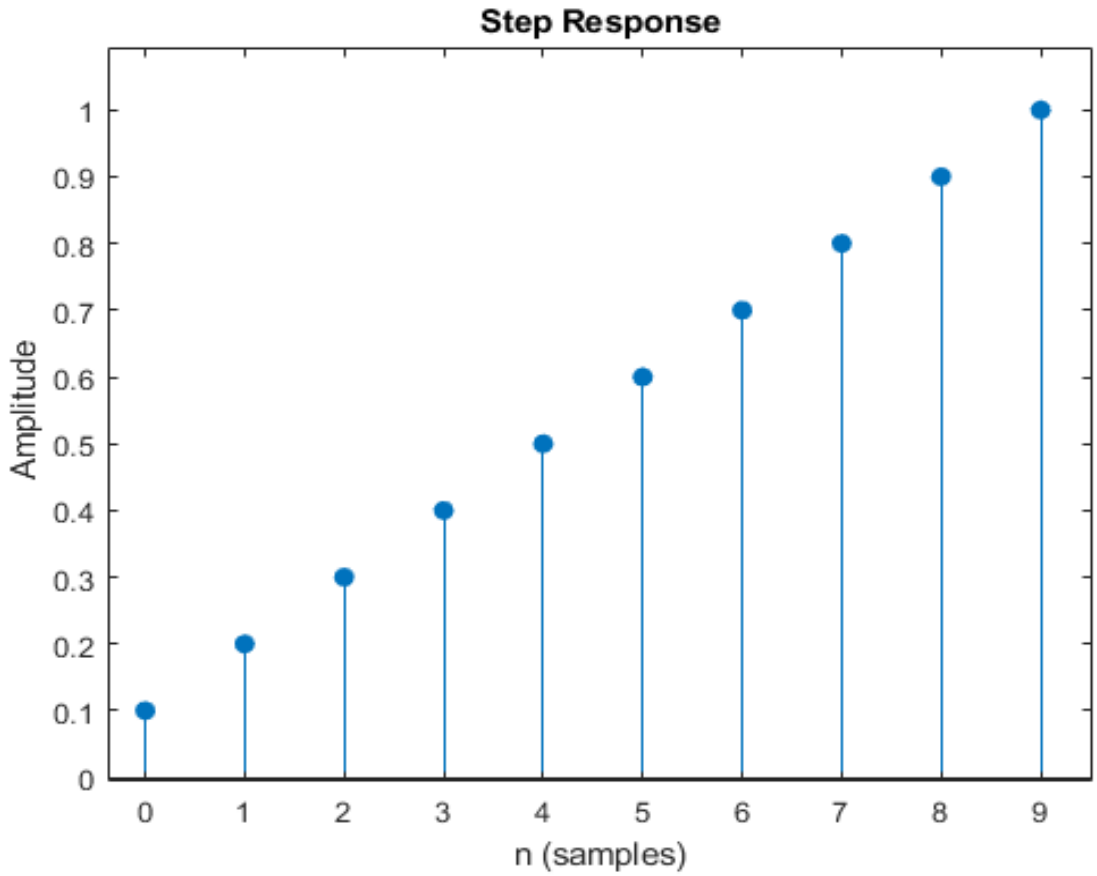

Fig.41 Step response of SMA with L = 10 (code: appendix 6)

The step response of the SMA filter is given by,

$$y_{step}[n] = \begin{cases} 0, & n < 0 \\ (n+1)/L, & 0 \leq n < L \\ 1, & n \geq L \end{cases}$$

For example, Fig.41 shows the step response of SMA with L = 10

Phase Delay

The phase delay shows the time shift (in number of sampling intervals) experienced by each sinusoidal component of the input signal. Constant phase delay indicates linear phase. It is defined as,

$$\tau_{pd} \triangleq \frac{\angle \mathbf{H}(e^{j\omega})}{\omega}$$

For example, Fig. 42 shows the step response of SMA with L = 10. Fig. 43 shows the phase delay for different values of L. It is observed that the phase delay increases proportionally with increase in L.

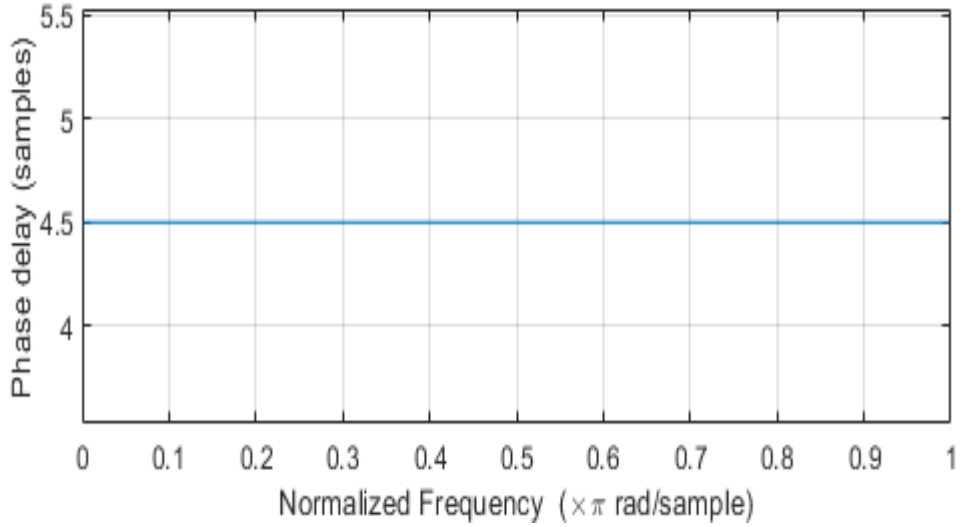

Fig.42: Phase delay for SMA with L =10 (code: appendix B7)

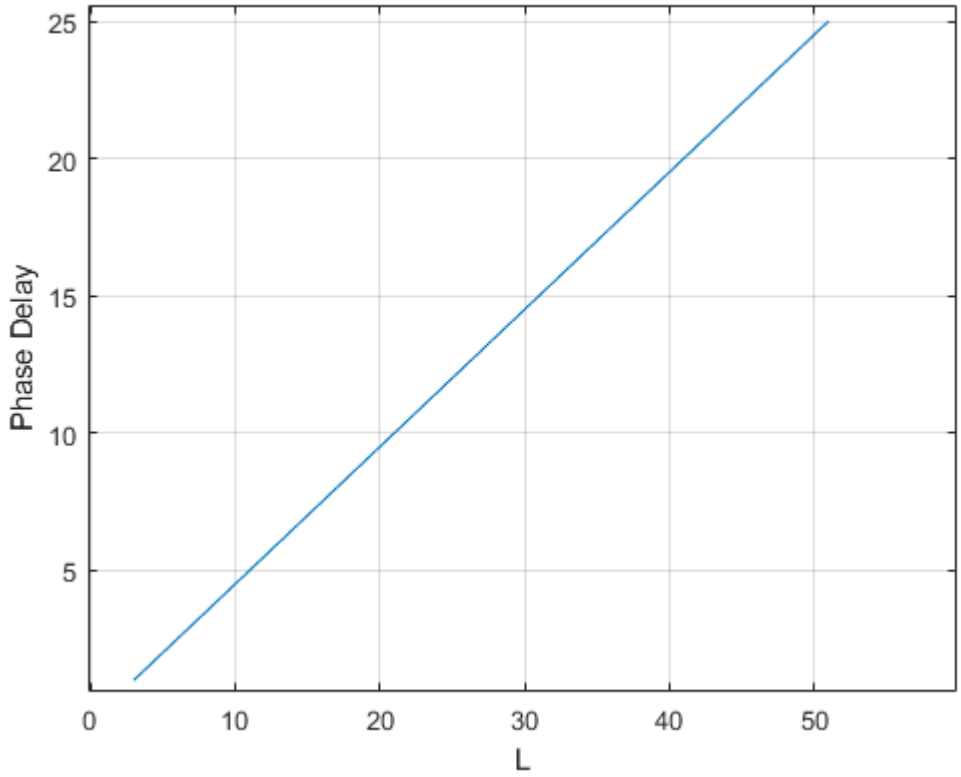

Fig.43: Phase Delay for SMA filter for different values of L (code: appendix B7)

Results

The SMA filter with L =10 is applied to accelereromter and gyroscope data and the output is visualized in the Fig. 44 and Fig. 45.

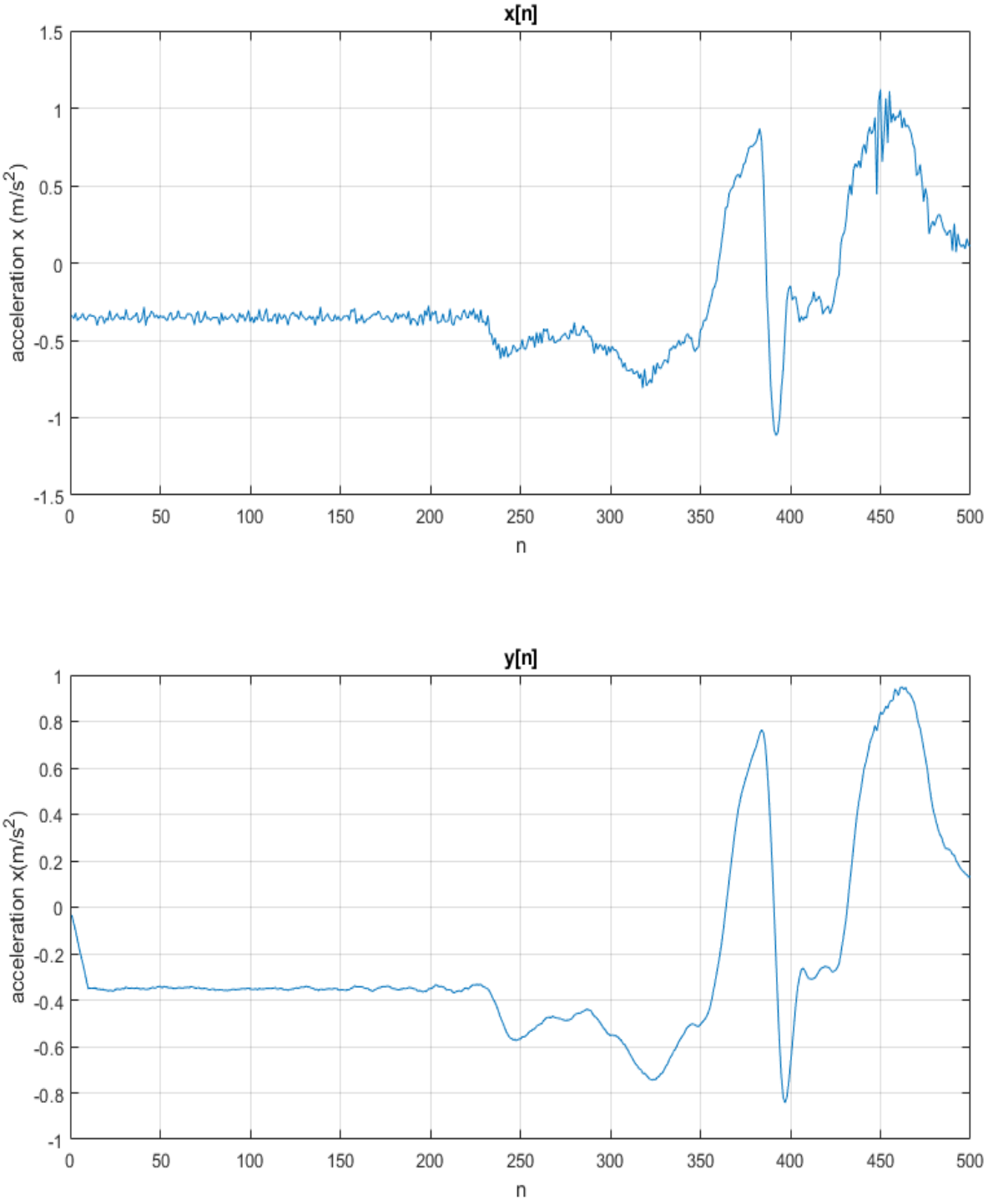

Fig. 44: Filtering of accelerometer data (code: appendix 8)

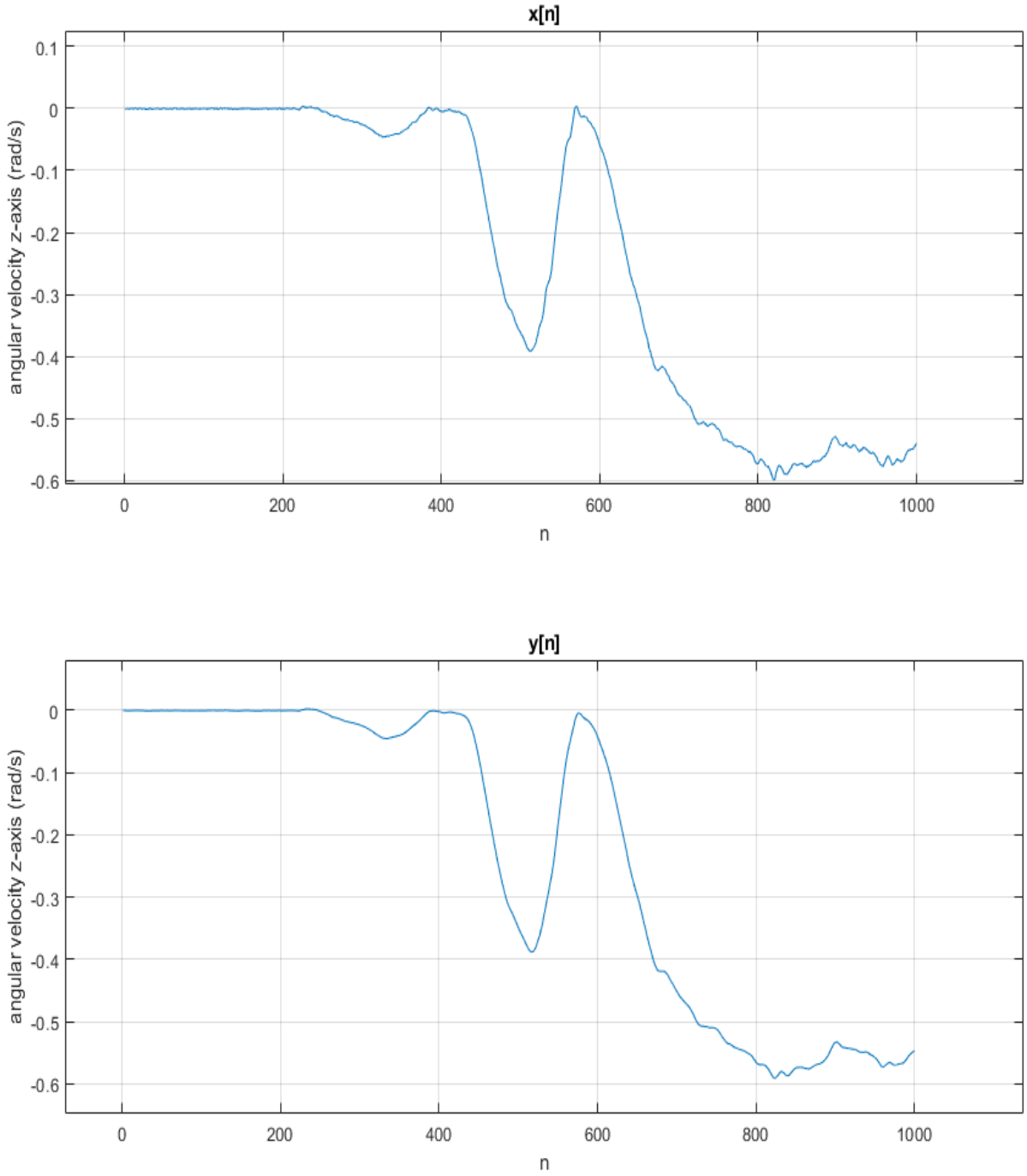

Fig.45: Filtering of gyroscope data (code: appendix 9)

Exponentially Weighted Moving Average Filter Implementation

This section presents a discussion of the use of an exponentially weighted moving average filter to remove noise from the gyroscope. As mentioned in the previous section with the simple moving average filter, our goal is to filter out high frequency noise to extract low frequency gyroscope data. The main assumption behind the exponentially weighted moving average filter is that, in dynamic systems, the more current values reflect better the state of the process, and should be considered to be of higher priority than values lying further away from the current position (Tham, 2010).

A weighted moving average filter introduces weights to each of the data points in the sample window, giving more weight to data points closer to the current position and less weight to the data points further away. In the case of the exponentially weighted moving average filter, this weighting factor decreases exponentially for each successive data point lying further away from the current position, though never reaching zero due to the nature of exponential weighting. Adding more weight to the more recently recorded data points allow the filter to adapt better to trends in the collected data.

1. Definition

The exponentially weighted moving average filter is an IIR low-pass filter; it is the implementation of the simplest case of exponential smoothing (. Here, we make use of a constant smoothing factor, or sometimes referred to as the filter constant or the forgetting factor α with a value between 0 and 1. A value of α closer to 1 increases the rate at which previous data values are discounted. Given the value of α, the exponential moving average, or EMA, of a series is recursively calculated by way of the following difference equation.

$$s[n] = \alpha y[n] + (1 - \alpha)s[n - 1]$$

Eq. 17: Difference Equation of the EMA filter

Where s[n] is the smoothed value at point n, or the value of the EMA at time n, y[n] is the value of the data at point n, and α the predetermined smoothing factor.

We can see that all past data points contribute to the smoothing of the data at a given instance, as the filtered value at the point t-1 accounts for all the previous values going back to the starting point of the dataset in question. Substituting the previous values into the above equation, we obtain the following, as noted by this article from Pieter P. (Pieter P., n.d.).

$$s[n] = \alpha y[n] + (1-\alpha)(\alpha y[n-1] + (1-\alpha)s[n-2])$$
$$= \alpha y[n] + (1-\alpha)(\alpha y[n-1] + (1-\alpha)(\alpha y[n-2] + (1-\alpha)s[n-3])$$
$$= \alpha y[n] + (1-\alpha)(\alpha y[n-1] + (1-\alpha)(\alpha y[n-2] + (1-\alpha)(\alpha y[n-3]$$
$$+ (1-\alpha)s[n-4]))$$

And so on, from which by simplifying we obtain the following expression.

$$s[n] = \alpha \sum_{k=0}^{n} (1-\alpha)^k y[n-k]$$

We observe from the above expression that the

For the time being, we will make use of the generally accepted value of the smoothing factor α as follows (Tham, 2010). We will be looking at the effects of the different values of α later on.

$$\alpha = \frac{n}{n+1}$$

Eq. 18: Definition of smoothing factor

2. Impulse and Step Response

Now, with the difference equation we have derived above, we will examine the impulse response and the step responses of the EMA filter. The article by Pieter P. (Pieter P., n.d.) is used as a reference and point of comparison for the following parts regarding the impulse and step responses of the EMA filter, as well as the transfer function later on. We obtain the impulse response by applying a DT unit impulse, or a delta function, to the input of the filter.

$$s[n] = \alpha \sum_{k=0}^{n} (1-\alpha)^k \delta[n-k] = \alpha(1-\alpha)^n$$

Taking the length of the unit impulse as 1, we obtain a value of 0.5 for the smoothing factor α. The following figure displays the impulse response of the EMA filter with α=0.5. The MATLAB code used to generate the following plot can be found in Appendix C1. We can observe in the following plot that as time goes on the impulse response approaches but never reaches 0, therefore making the exponentially weighted moving average filter an IIR filter.

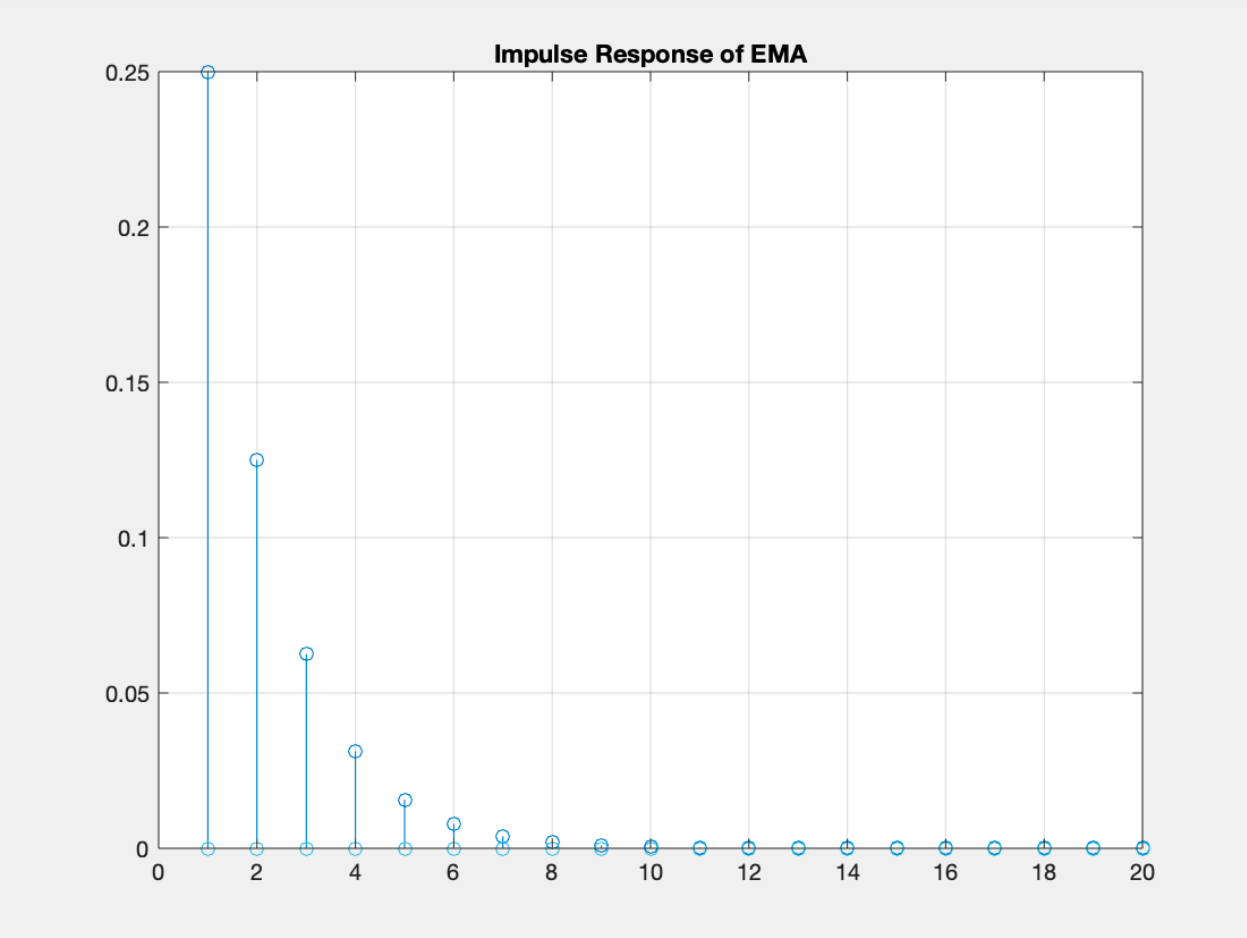

Fig. 46: Impulse Response of EMA Filter

The step response of the EMA filter is obtained by applying a DT unit step function to the input of the filter. We use the notation H[n] to denote the unit step, also known as the Heaviside, function in the following equation.

$$s[n] = \alpha \sum_{k=0}^{n} (1-\alpha)^k H[n-k] = 1 - (1-\alpha)^k$$

The following plot displays the step response of the EMA filter, due to the infinite length of the unit step function, we will be using α=0.5 just as in the case with the step response. The MATLAB code used to generate the following plot can be found in Appendix C2.

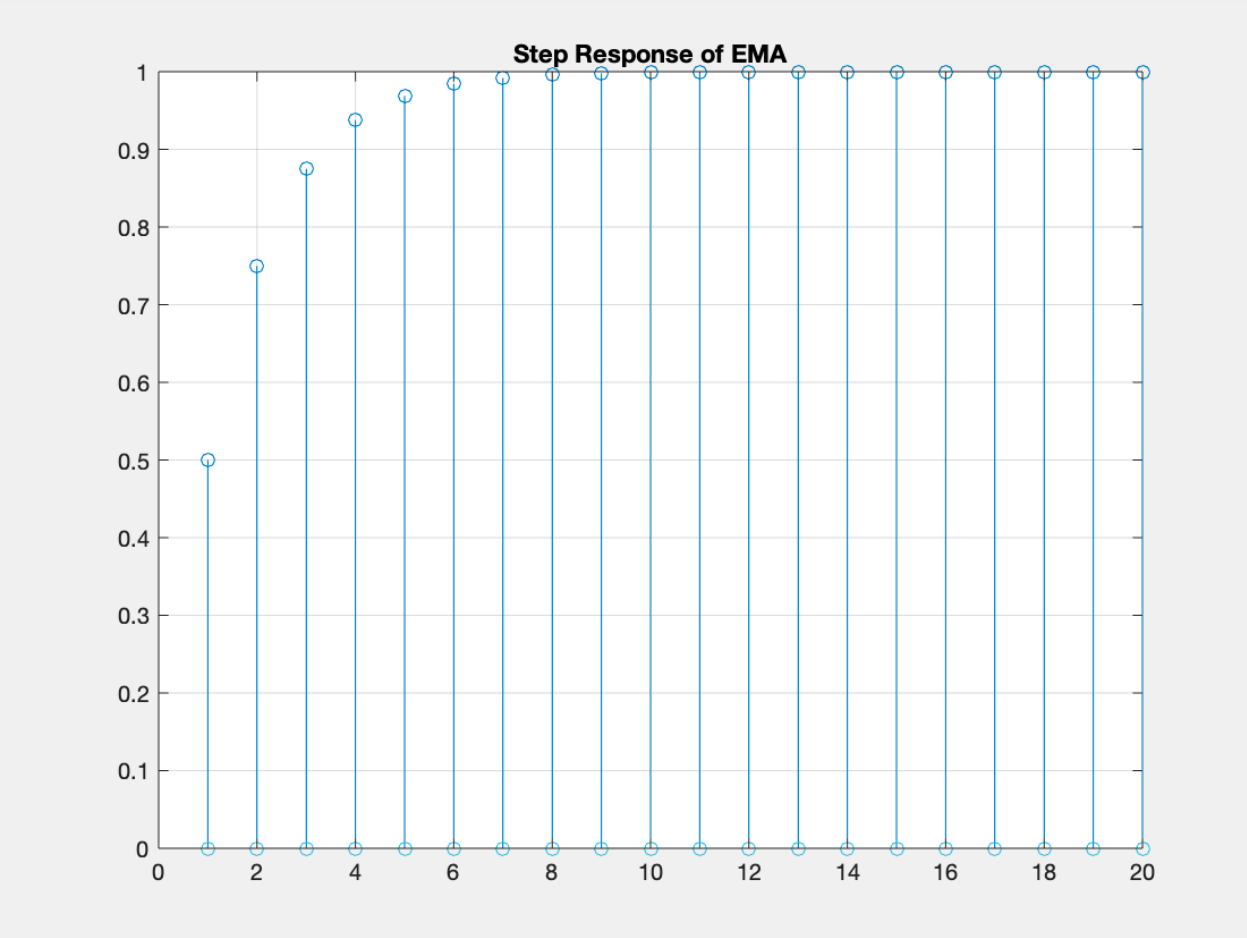

Fig. 47: Step Response of EMA Filter

3. Transfer Function

Having derived the impulse and step responses of the filter, we will now derive the transfer function of the EMA filter. As the EMA filter is an LTI system, the transfer function of the EMA filter can be obtained by taking the z-transform of the impulse response. The difference equation representation of the EMA filter allows us to easily obtain the transfer function.

$$y[n] = \alpha x[n] + (1-\alpha)y[n-1]$$

$$Z\{y[n]\} = Z\{\alpha x[n] + (1-\alpha)y[n-1]\} = \alpha Z\{x[n]\} + (1-\alpha)Z\{y[n-1]\}$$

$$= \alpha Z\{x[n]\} + (1-\alpha)z^{-1}Z\{y[n]\}$$

$$Y(z) = \alpha X(z) + (1-\alpha)z^{-1}Y(z)$$

$$Y(z)[1-(1-\alpha)z^{-1}] = \alpha X(z)$$

$$H(z) = \frac{Y(z)}{X(z)} = \frac{\alpha}{1-(1-\alpha)z^{-1}}$$

By rewriting the expression above to the following form.

$$H(z) = \frac{Y(z)}{X(z)} = \frac{\alpha z}{z - (1 - \alpha)}$$

We find that z = 0 is a zero and z = 1 – α is a pole.

a. Phase and Magnitude Plots

From the above transfer function, we can obtain the magnitude and phase plots in order to evaluate the effects of using different values of α. The following equation can be used to obtain the magnitude of the transfer function for a given value of the smoothing factor α. The following plots display the magnitude and phase of the transfer function above, here we take 0.5 as the value of smoothing coefficient α. The MATLAB code used to generate the following plots can be found in Appendix C3.

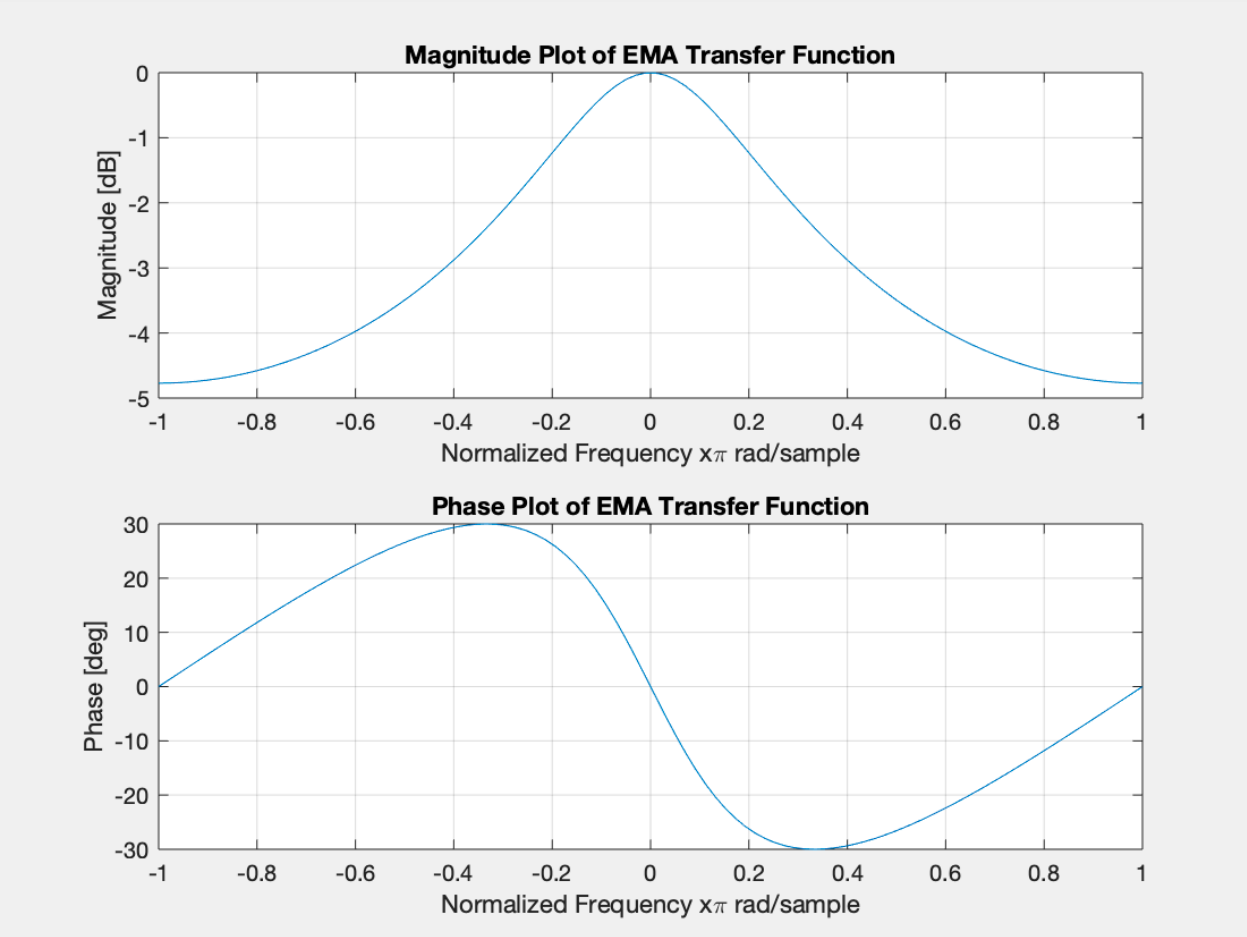

Figure 48: Magnitude and Phase of the EMA Filter with α=0.5

We observe that the EMA filter has a stopband attenuation of 5 dB, with α=0.5 which is very poor performance for a low pass filter in the general sense.

b. Smoothing Factor

Now that we have derived the transfer function of the EMA filter, we will examine the effects of using different values of the smoothing factor α on the filter's behavior. The following plots display the magnitude plot of the EMA filter with α=0.1, α=0.3, α=0.5 and α=0.9. The MATLAB code used to generate the following plot can be found on Appendix C4.

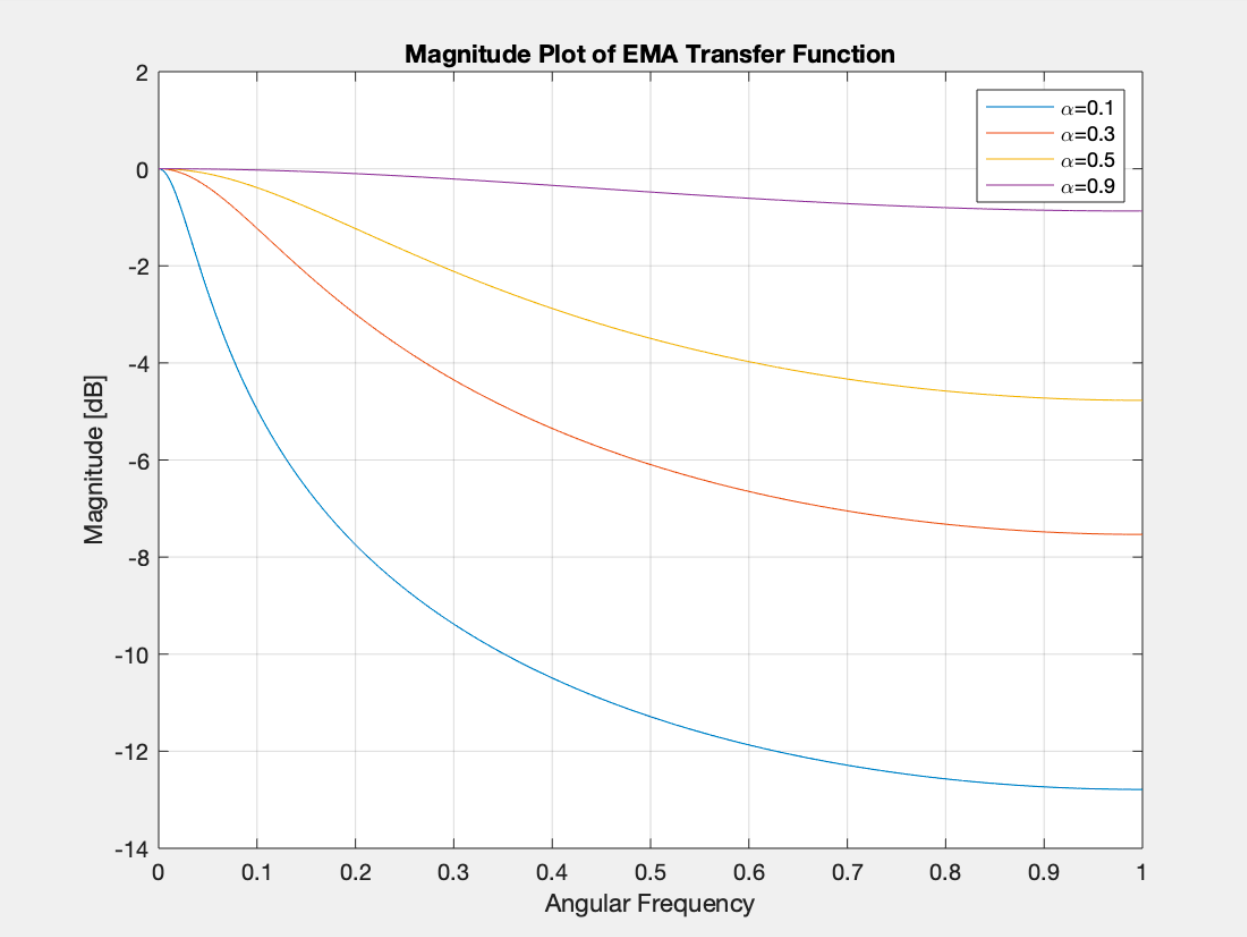

Fig 49: Magnitude Plots with Different Values of α

We can see that the different values of the smoothing coefficient α affects the width of the passband and the attenuation of the stopband. Again, the highest value of stopband attenuation observed with this filter is considerably poor, though the filter only allows frequencies within the single lobe to pass through. Note that the exponentially weighted moving average function of the MATLAB DSP toolbox defaults α to a value of 0.9 if not otherwise specified.

c. Phase Delay

We will now examine the phase delay of the EMA filter, as well as the effects of using different values of the smoothing coefficient on the phase delay. The following plot displays the phase delay with α=0.5. The MATLAB code for this plot can be found in Appendix C5.

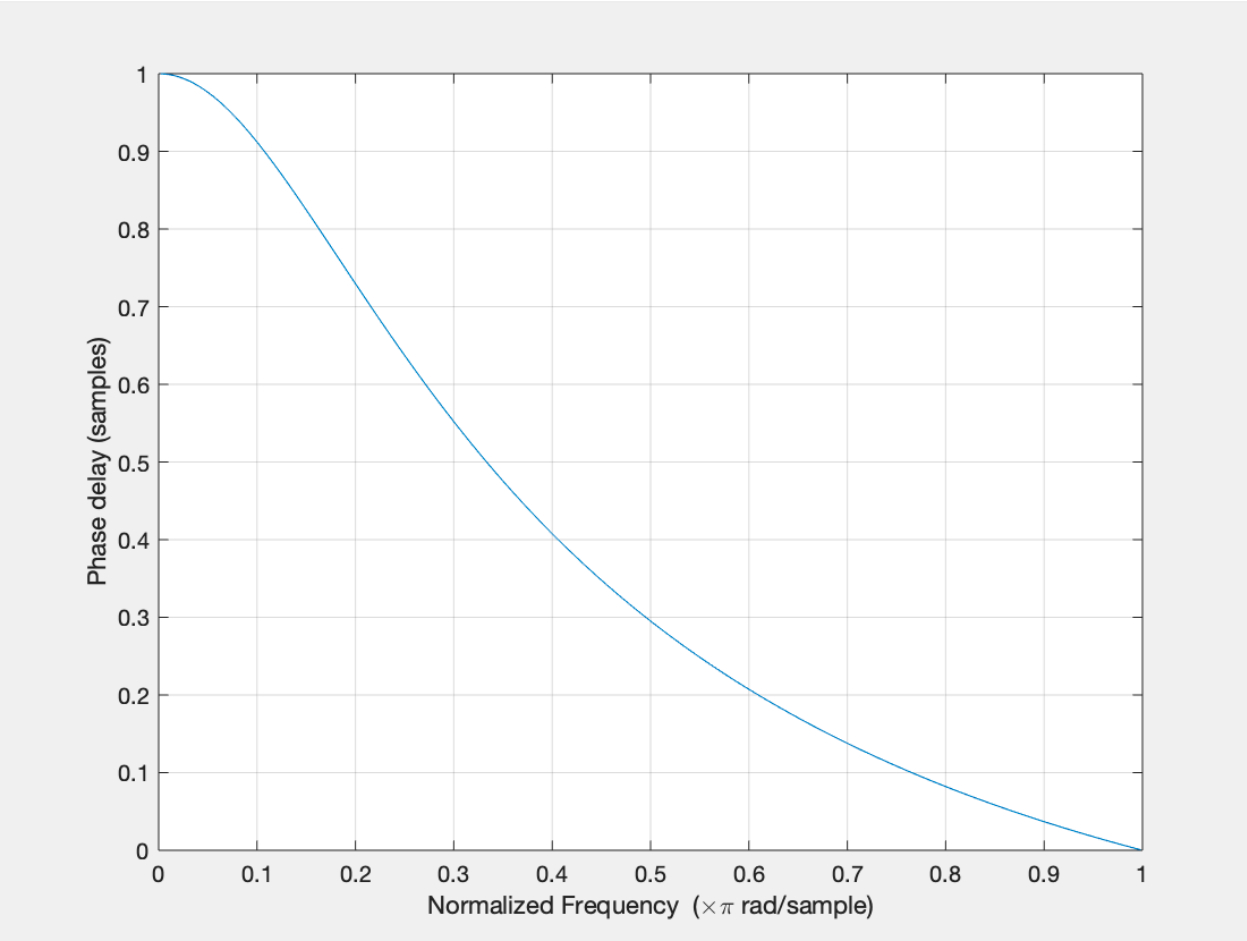

Figure 50: Phase Delay of EMA Filter with α=0.5

We observe that the EMA filter with α=0.5 does not have a constant phase delay, and as we can see in its phase plot, the EMA does not have linear phase and may thus introduce distortion. The following plot displays the effects of varying α, with the values listed previously, on phase delay. The MATLAB code used to generate the following plot can be found on Appendix C6.

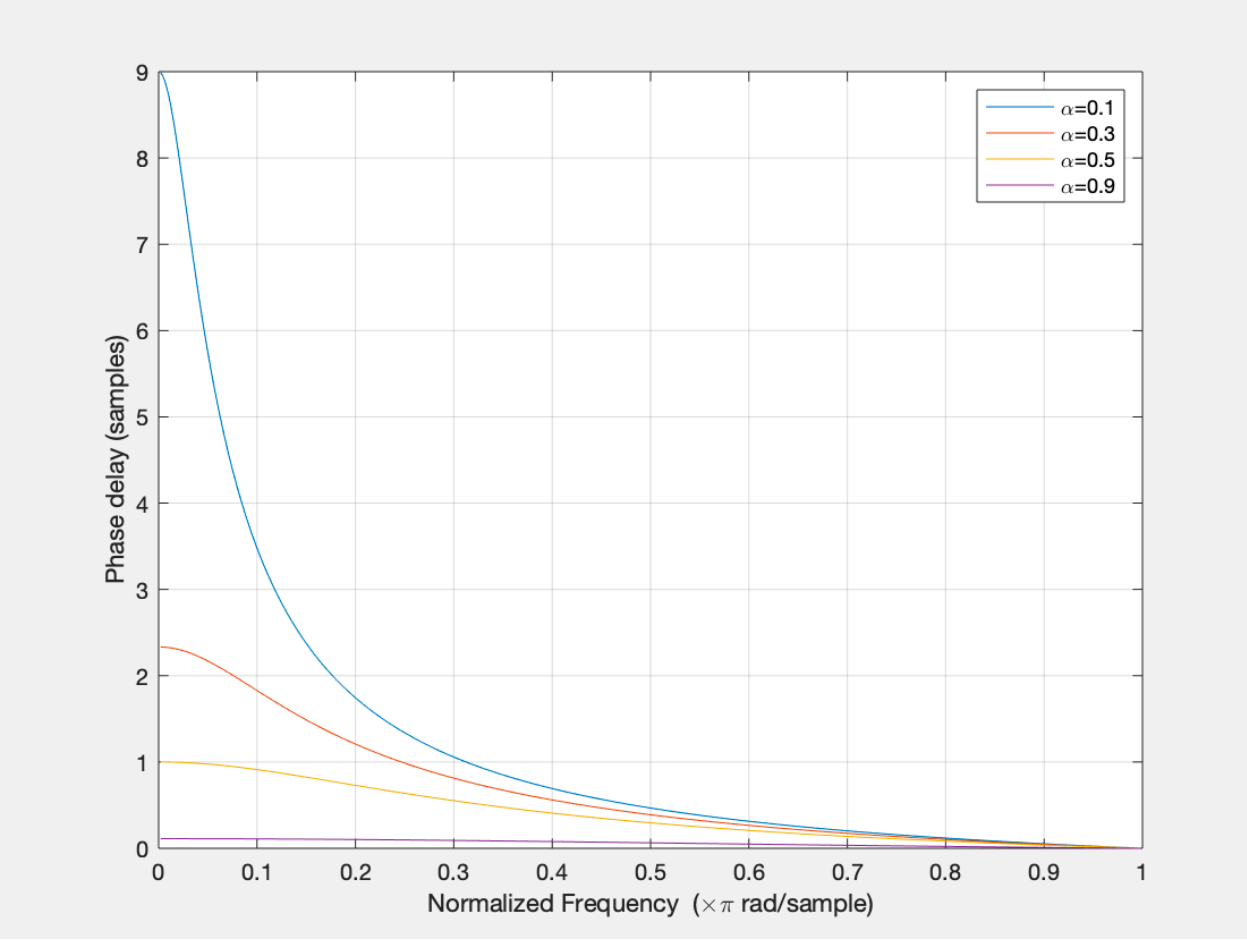

Figure 51: Phase Delay with Different Values of α

Here we can observe that the phase delay introduced by the EMA filter with different values of α do not remain at a constant level. The change in phase delay is observed to be much greater with the smaller values of α and much closer to a flat line with values of α closer to 1. The greater the smoothing factor, the less prone the EMA filter is to creating disturbance, but the greater the effects of data points located further away from the sample currently evaluated. Judging by its performance as observed thus far, the exponentially weighted moving average filter may not be the best filter to be used in smoothing IMU data. We will apply our EMA filter on our IMU dataset next to observe its performance in smoothing our IMU data.

4. Results on IMU Dataset

Now that we have defined and characterized the exponentially weighted moving average filter and observed its behavior, we will apply this exponentially weighted moving average filter on our IMU dataset. The dataset used here is the same dataset used in the section on the Simple Moving Average filter. We examine the effect of the EMA filter on the orientation and acceleration in the z-direction in the given dataset. For this purpose, we have chosen to use the definition of the smoothing factor α as found in Eq. 18. The MATLAB code used to generate the plots below can be found in Appendix C7.

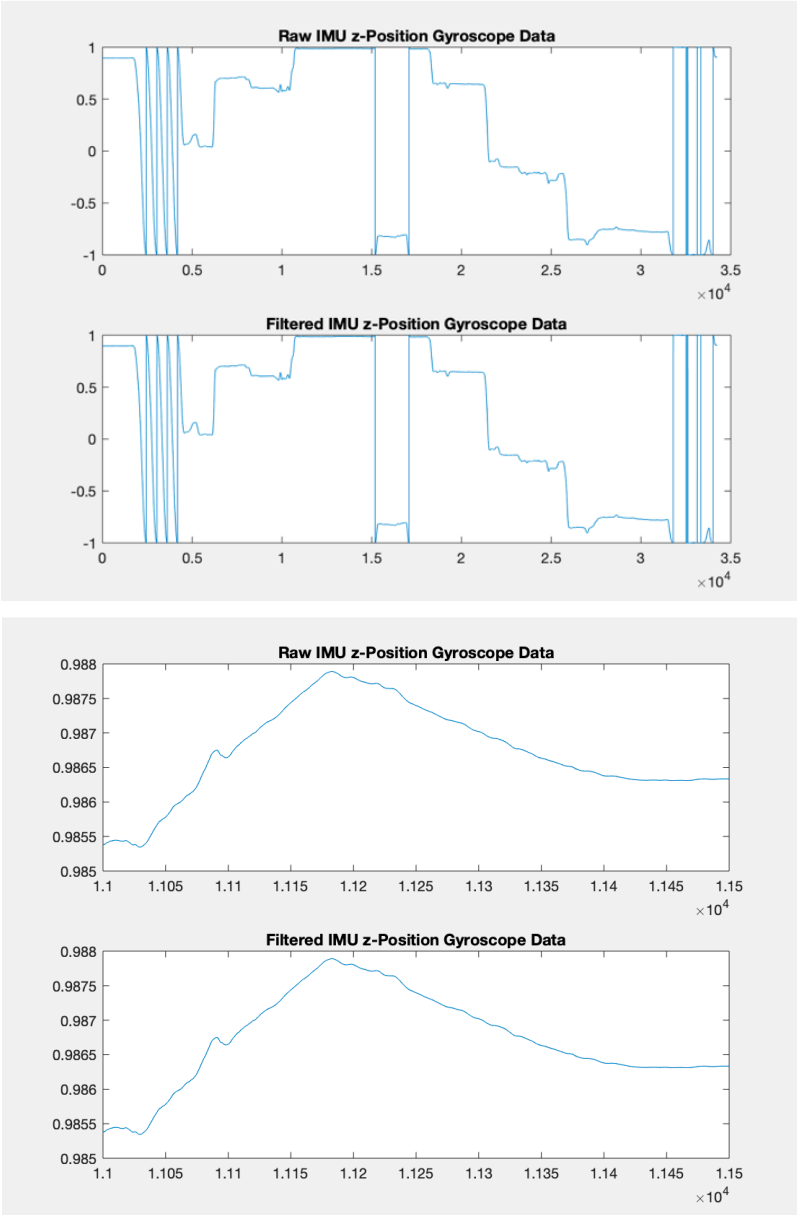

Figure 52: Raw and Filtered IMU Gyroscope Data – all data points and zoomed in.

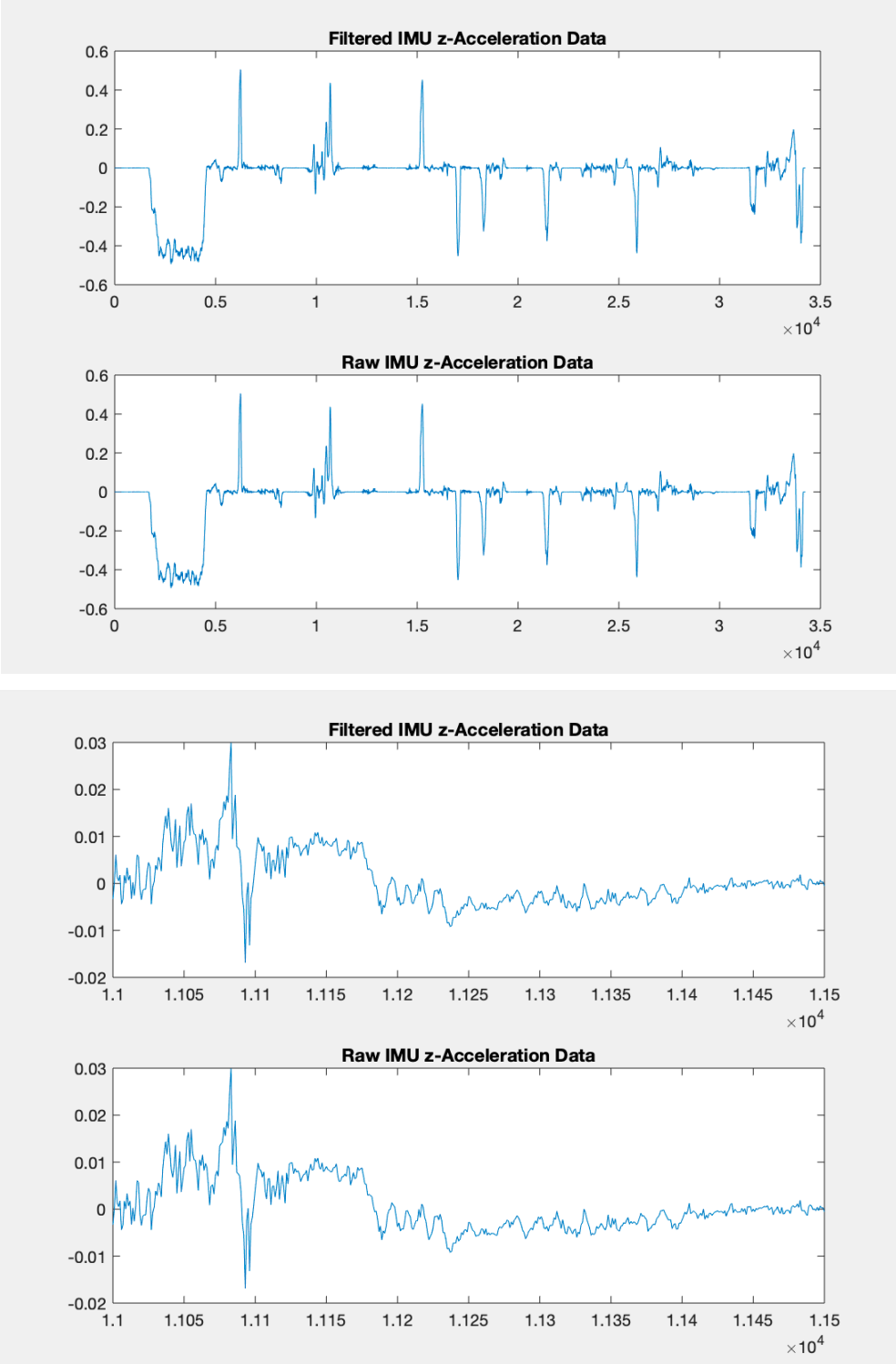

Figure 53: Raw and Filtered IMU Acceleration Data – all data points and zoomed in

As the above figures suggest, the EMA filter we used which has a smoothing factor close to 1 does little to filter out unwanted noise, though does not introduce distortions in the signal. Next, we will examine the effect of using a smaller value of α in filtering performance.

The following plots display the raw and filtered IMU data from the above dataset in the fields as mentioned previously. Here we chose a value of α=0.1 as to ensure better low-pass filter performance. The same code as found in Appendix C7 is used, replacing the value of α with our predetermined value.

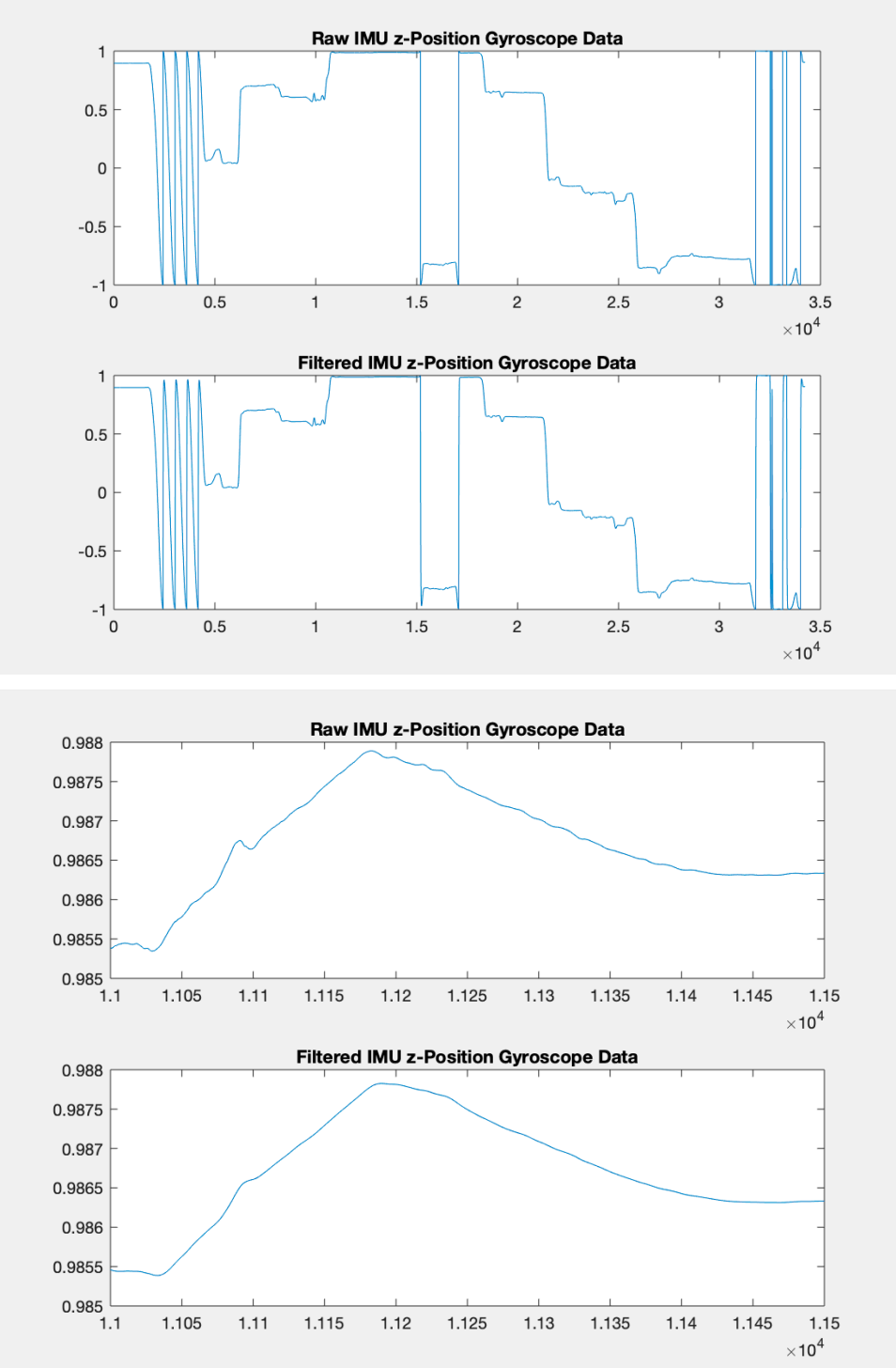

Figure 54: Raw and Filtered IMU z-Position Gyroscope Data – all data points and zoomed in.

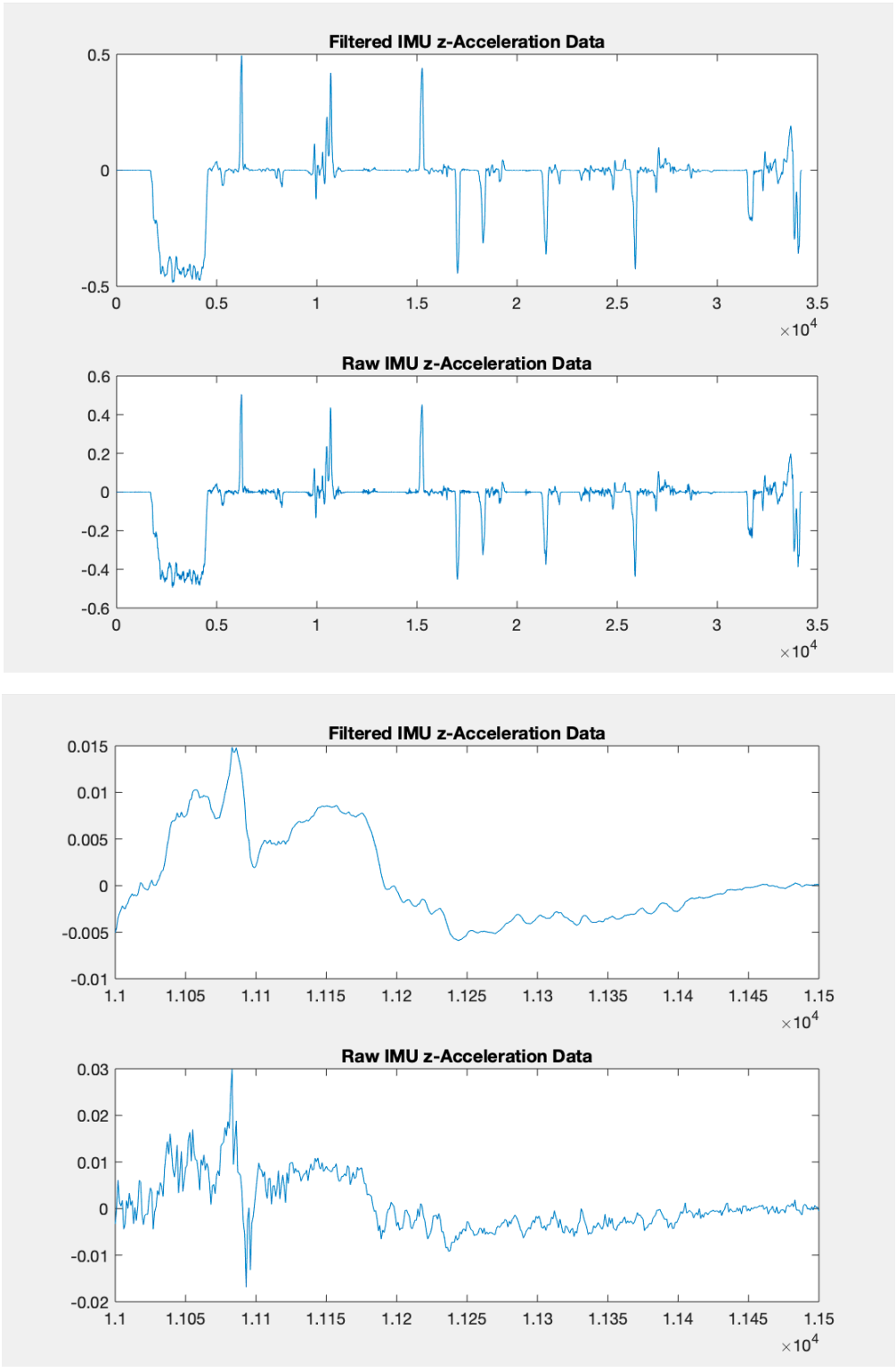

Figure 55: Raw and Filtered IMU z-Acceleration Data - all data points and zoomed in. Here, we can observe better LPF performance than in the previous case.

Conclusion

In this paper, we have presented three types of simple filters for IMU filtering: Complementary Filters, Moving Average Filters, and Exponentially Weighted Moving Average Filters. These filters are suitable for embedded systems which have 8-bit system. There are other common IMU filters like Kalman Filters and Madgwick Filters discussed in the introduction. Comparing with our finished work, Kalman and Madgwick Filters have better performance in noise removing, but more complex and stricter with the hardware environment. IMU magnetometer reading is the useful data for angular rate sensor drift cancellation, which may be complemented in the future.

References

Colorlib. (n.d.). NJS. Retrieved April 24, 2021, from https://nitinjsanket.github.io/tutorials/attitudeest/madgwick

Ellis, K., Godbole, S., Marshall, S., Lanckriet, G., Staudenmayer, J., & Kerr, J. (2014, April 22). Identifying active travel behaviors in challenging environments using GPS, accelerometers, and machine learning algorithms. Retrieved April 23, 2021, from https://www.researchgate.net/publication/262055313_Identifying_Active_Travel_Behaviors_in_Challenging_Environments_Using_GPS_Accelerometers_and_Machine_Learning_Algorithms

ENAE788M: Class 2 Part 2 - IMU Basics, Attitude estimation using CF AND MADGWICK [Video file]. (2019, August 27). Retrieved April 24, 2021, from https://www.youtube.com/watch?v=8hRoASoBEwY

Kim, Y., & Bang, H. (2018, November 05). Introduction to Kalman filter and its applications. Retrieved April 25, 2021, from https://www.intechopen.com/books/introduction-and-implementations-of-the-kalman-filter/introduction-to-kalman-filter-and-its-applications

Kumar, H. (2014, December 21). Beginner's guide to IMU. Retrieved April 24, 2021, from https://students.iitk.ac.in/roboclub/2017/12/21/Beginners-Guide-to-IMU.html

Pieter P. (n.d.). Exponential moving average. Retrieved April 28, 2021, from https://tttapa.github.io/Pages/Mathematics/Systems-and-Control-Theory/Digital-filters/Exponential%20Moving%20Average/Exponential-Moving-Average.html

Reading a IMU WITHOUT Kalman: The Complementary Filter. (2013, April 26). Retrieved April 23, 2021, from https://www.pieter-jan.com/node/11

S. (2020, January 19). What is IMU Sensor and How to use with Arduino? Retrieved April 23, 2021, from https://www.seeedstudio.com/blog/2020/01/17/what-is-imu-sensor-overview-with-arduino-usage-guide/

Scherr, C., Lommatsch, G., Hooson, S., & Kaiser, T. (n.d.). Underwater Navigation System. Retrieved April 25, 2021, from https://ece.montana.edu/seniordesign/archive/SP14/UnderwaterNavigation/kalman_filter.html

Tham, M. T. (2010, March 29). Dealing With Measurement Noise: An Introduction to Noise Filtering. Retrieved April 28, 2021, from https://web.archive.org/web/20100329135531/http:/lorien.ncl.ac.uk/ming/filter/filewma.htm

Vathsangam, H. (n.d.). Magnetometer - My imu Estimation experience. Retrieved April 24, 2021, from https://sites.google.com/site/myimuestimationexperience/sensors/magnetometer

S. NV, "iNEMO inertial module: 3D accelerometer, 3D gyroscope, 3D magnetometer," 2015. [Online]. Available: https://content.arduino.cc/assets/Nano_BLE_Sense_lsm9ds1.pdf.

D. Manolakis and I. Vinay, Applied Digital Signal Processing, Boston: Cambridge University Press, 2011, p. 217.

M. L., "What is the advantage of MATLAB's filtfilt," 12 JUN 2018. [Online]. Available: https://dsp.stackexchange.com/questions/9467/what-is-the-advantage-of-matlabs-filtfilt.

Appendix A:

Complementary Filters System

The code can be reached through GitHub Repository: https://github.com/wwtse/eece5666gp1

1) Lowpass filter for gyrometer

```
clear;
%close all;
%% load
%Hover data
%{
fname = 'RSdataMine';
load("IMU_Hover/"+fname+".mat");

at = rt_sensor.time;
T = at(2)- at(1);
fs = round(1/T);

ax = rt_sensor.signals.values(:,1);
ay = rt_sensor.signals.values(:,2);
az = rt_sensor.signals.values(:,3);
%}

%calib data
%{
load("imu_calib.csv")
at = imu_calib(:,1)*10^-9;
T = at(2)- at(1);
fs = round(1/T);

ax = imu_calib(:,29);
ay = imu_calib(:,30);
az = imu_calib(:,31);
%}

%arduino data

fs = 119;
g = 9.81;

fo = 3; %5 is useless
fname = "putty_test"+fo;
data1 = table2array(readtable("IMU_Hover/"+fname));
ax = data1(:,1)*g;
ay = data1(:,2)*g;
az = data1(:,3)*g;
at = 1/fs*(0:(length(data1)-1));
%% filter
fir = trfir1;

fx1 = filter(fir,1,ax);
fy1 = filter(fir,1,ay);
```

```
fz1 = filter(fir,1,az);

%% cut linear delay
fdelay  = round(mean(grpdelay(fir,fs,length(at))));
att = at(1:end-fdelay);

axx = ax(1:end-fdelay);
fx1_1 = fx1;
fx1_1(1:fdelay) = [];

ayy = ay(1:end-fdelay);
fy1_1 = fy1;
fy1_1(1:fdelay) = [];

azz = az(1:end-fdelay);
fz1_1 = fz1;
fz1_1(1:fdelay) = [];

%% plot

figure;

subplot(3,1,1)
hold on;
plot(att,axx);
plot(att,fx1_1);
title("ax")

subplot(3,1,2)
hold on;
plot(att,ayy);
plot(att,fy1_1);
title("ay")

subplot(3,1,3)
hold on;
plot(att,azz);
plot(att,fz1_1);
title("az")

%sgtitle('trfir1');

%without cut
%{
figure;
title(char(fir))
subplot(3,1,1)
```

```
hold on;
plot(at,ax);
plot(at,fx1);
title("ax")

subplot(3,1,2)
hold on;
plot(at,ay);
plot(at,fy1);
title("ay")

subplot(3,1,3)
hold on;
plot(at,az);
plot(at,fz1);
title("az")
%}

%% output
%save(fname(7:end)+"facc"+".mat",'att','fx1_1','fy1_1','fz1_1');
%save("calib"+"facc"+".mat",'att','fx1_1','fy1_1','fz1_1');

%save("Arduino"+fname+"facc"+".mat",'att','fx1_1','fy1_1','fz1_1');
```

2) Lowpass filter for gyrometer

```
clear;
%close all;
%% load
%Hover data
%{
fname = 'RSdataMine';
load("IMU_Hover/"+fname+".mat");

wt = rt_sensor.time;
T = wt(2)- wt(1);
fs = round(1/T);

p = rt_sensor.signals.values(:,4);
q = rt_sensor.signals.values(:,5);
r = rt_sensor.signals.values(:,6);
%}
%calib data
%{
load("imu_calib.csv")
wt = imu_calib(:,1)*10^-9;
T = wt(2)- wt(1);
fs = round(1/T);

p = imu_calib(:,17);
q = imu_calib(:,18);
r = imu_calib(:,19);
%}
%arduino data

fs = 119;
dps2SI = 0.01745329252;

fo = 10;
fname = "putty_test"+fo;
data1 = table2array(readtable("IMU_Hover/"+fname));
p = data1(:,4)*dps2SI;
q = data1(:,5)*dps2SI;
r = data1(:,6)*dps2SI;
wt = 1/fs*(0:(length(data1)-1));

%%

fir1 = tfir1;
fir2 = tfir1;
```

```
fp1 = filter(fir1,1,p);
fq1 = filter(fir1,1,q);
fr1 = filter(fir2,1,r);


fdelay  = mean(grpdelay(fir1,fs,length(wt)));
wtt = wt(1:end-fdelay);

pp = p(1:end-fdelay);
fp1_1 = fp1;
fp1_1(1:fdelay) = [];

qq = q(1:end-fdelay);
fq1_1 = fq1;
fq1_1(1:fdelay) = [];

fdelayr  = mean(grpdelay(fir2,fs,length(wt)));
wtt2 = wt(1:end-fdelayr);
rr2 = r(1:end-fdelayr);
fr1_1 = fr1;
fr1_1(1:fdelayr) = [];

%% plot

figure;
subplot(3,1,1)
hold on;
plot(wtt,pp);
plot(wtt,fp1_1);
title("p")

subplot(3,1,2)
hold on;
plot(wtt,qq);
plot(wtt,fq1_1);
title("q")

subplot(3,1,3)
hold on;
plot(wtt2,rr2);
plot(wtt2,fr1_1);
title("r")

%save(fname(7:end)+"fgyro"+".mat",'wtt','fp1_1','fq1_1','wtt2','fr1_1');
save("Arduino"+fname+"fgyro"+".mat",'wtt','fp1_1','fq1_1','wtt2','fr1_1');
```

3) IIR Lowpass filter

```
%iir

clear;
close all;
%% load
%arduino data

fs = 119;
g = 9.81;

fo = 6;
fname = "putty_test"+fo;
data1 = table2array(readtable("IMU_Hover/"+fname));
ax = data1(:,1)*g;
ay = data1(:,2)*g;
az = data1(:,3)*g;
at = 1/fs*(0:(length(data1)-1));

ax(end) = [];
ay(end) = [];
az(end) = [];
at(end) = [];
%% filter
iir = tiir1;

%filtfilt

fx1 = filtfilt(iir.sosMatrix,iir.ScaleValues,ax);
fy1 = filtfilt(iir.sosMatrix,iir.ScaleValues,ay);
fz1 = filtfilt(iir.sosMatrix,iir.ScaleValues,az);


%filter
%{
fx1 = filter(iir,ax);
fy1 = filter(iir,ay);
fz1 = filter(iir,az);
%}
%% plot
%{
figure;
title(iir.FilterStructure)
subplot(3,1,1)
hold on;
plot(at,ax);
plot(at,fx1);
```

```
title(″ax″)

subplot(3,1,2)
hold on;
plot(at,ay);
plot(at,fy1);
title("ay")

subplot(3,1,3)
hold on;
plot(at,az);
plot(at,fz1);
title("az")

sgtitle('IIR');
%}
```

4) Complementary Filter

```
clear;
fo = 10;
fname = "Arduino"+"putty_test"+fo;

load(fname+"facc.mat");
load(fname+"fgyro.mat");

L = min([length(att) length(wtt) length(wtt2)]);
%T = att(2,1)-att(1,1);
T = att(2)-att(1);% arduino dataset

f = zeros(L,7);
f(:,1) =fx1_1(1:L);
f(:,2) =fy1_1(1:L);
f(:,3) =fz1_1(1:L);
f(:,4) =fp1_1(1:L);
f(:,5) =fq1_1(1:L);
f(:,6) =fr1_1(1:L);
f(:,7) = att(1:L);

%accelometer angles
accel_ang = zeros(L,3);

for i = 1:L

    x = -f(i,2)/(((f(i,1))^2 + (f(i,3))^2)^0.5);
    y = f(i,1)/(((f(i,2))^2 + (f(i,3))^2)^0.5);
    z = (((f(i,1))^2 + (f(i,2))^2)^0.5)/f(i,3);

    accel_ang(i,1) = atan(x);
    accel_ang(i,2) = atan(y);
    accel_ang(i,3) = atan(z);

%     accel_ang(i,1) = phi;
%     accel_ang(i,2) = theta;
%     accel_ang(i,3) = ksi;

end

%gyroscope angles
gyro_ang = zeros(L,3);

for i = 1:L

    gyro_ang(i+1,1) = gyro_ang(i,1)+f(i,4)*T;
    gyro_ang(i+1,2) = gyro_ang(i,2)+f(i,5)*T;
```

```
    gyro_ang(i+1,3) = gyro_ang(i,3)+f(i,6)*T;
end
gyro_ang(end,:)=[];

%Assuming that the initial orientation is zero.
tau = 0.1;  % desired time constant
alpha = tau/(tau+T);
c_ang = (1-alpha)*gyro_ang+alpha*accel_ang;

for i = 1:2
    figure();
    hold on;
    plot(f(:,7),accel_ang(:,i),'DisplayName',"a"+i);
    plot(f(:,7),gyro_ang(:,i),'DisplayName',"g"+i);
    plot(f(:,7),c_ang(:,i),'DisplayName',"c"+i);
    legend
end
```

Appendix B: Moving Average Filter

1)

```
L = 10;
a = [1];
b = repelem(1/L, L);
omega = linspace(-pi, pi, 1000);
freqz(b,a, omega);
```

2)

```
L = 10;
a = [1];
b = repelem(1/L, L);
H= freqz(b,a, omega);
figure();
plot(omega, abs(H)); grid;
xlabel('omega'); ylabel('Magnitude of H');
```

3)

```
L = 3;
omega = linspace(0, pi, 1000);
a = [1];
b = repelem(1/L, L);
H= freqz(b,a, omega);
figure();
plot(omega, abs(H)); grid;

L = 11;
omega = linspace(0, pi, 1000);
a = [1];
b = repelem(1/L, L);
H= freqz(b,a, omega);
hold on;
plot(omega, abs(H)); grid;

L = 51;
omega = linspace(0, pi, 1000);
a = [1];
b = repelem(1/L, L);
H= freqz(b,a, omega);
hold on;
plot(omega, abs(H)); grid;
xlabel('omega'); ylabel('Magnitude of H');
legend('L = 3','L = 11', 'L = 51')
```

4)

```
L = 11;
omega = linspace(0, pi, 1000);
a = [1];
b = repelem(1/L, L);
zplane(b,a)
```

5)

```
L = 10;
a = [1];
b = repelem(1/L, L);
figure();
impz(b,a);
```

6)

```
L = 10;
a = [1];
b = repelem(1/L, L);
figure();
stepz(b,a);
```

7)

```
L = 10;
a = [1];
b = repelem(1/L, L);
figure();
phasedelay(b,a);
pd = zeros(1,49);
i = 0;
for L = 3:1:51
    i = i + 1;
    a = [1];
    b = repelem(1/L, L);
    [phi, w] = phasedelay(b,a);
    pd(i) = phi(2);
end
figure();
plot([3:1:51], pd); grid;
xlabel('L'); ylabel('Phase Delay');
```

Appendix C: Exponentially Weighted Moving Average

1. Impulse Response

```
filtered = zeros(20);
for i = 1:20
    filtered(i) = alpha*(1-alpha)^i;
end

figure
stem(filtered)
grid on
title("Impulse Response of EMA")
```

2. Step Response

```
filtered = zeros(20);
for i = 1:20
    filtered(i) = 1-(1-alpha)^i;
end

figure
stem(filtered)
grid on
title("Step Response of EMA")
```

3. Magnitude and Phase Plot

```
alpha = 0.5;
num = alpha;
denom = [1 -(1-alpha)];
w = linspace(-pi,pi);
H = freqz(num,denom,w);
mag = abs(H);
phase = rad2deg(angle(H));
figure
subplot(2,1,1)
plot(w/pi,10*log10(mag))
ylabel("Magnitude [dB]")
xlabel("Normalized Frequency x\pi rad/sample")
title("Magnitude Plot of EMA Transfer Function")
grid on
subplot(2,1,2)
plot(w/pi,phase)
ylabel("Phase [deg]")
xlabel("Normalized Frequency x\pi rad/sample")
title("Phase Plot of EMA Transfer Function")
grid on
```

4. Magnitude Plot of Different Smoothing Factors

```
alpha = 0.1;
num = alpha;
denom = [1 -(1-alpha)];
w = linspace(0,pi);
H = freqz(num,denom,w);
mag = abs(H);
figure
plot(w/pi,10*log10(mag))
hold on

alpha = 0.3;
num = alpha;
denom = [1 -(1-alpha)];
H = freqz(num,denom,w);
mag = abs(H);
plot(w/pi,10*log10(mag))
hold on

alpha = 0.5;
num = alpha;
denom = [1 -(1-alpha)];
H = freqz(num,denom,w);
mag = abs(H);
plot(w/pi,10*log10(mag))
hold on

alpha = 0.9;
num = alpha;
denom = [1 -(1-alpha)];
H = freqz(num,denom,w);
mag = abs(H);
plot(w/pi,10*log10(mag))
hold on
ylabel("Magnitude [dB]")
xlabel("Angular Frequency")
title("Magnitude Plot of EMA Transfer Function")
legend("\alpha=0.1","\alpha=0.3","\alpha=0.5","\alpha=0.9");
grid on
```

5. Phase Delay

```
alpha = 0.5;
num = alpha;
denom = [1 -(1-alpha)];
figure
phasedelay(num,denom);
```

6. Phase Delay of Different Smoothing Factors

```
alpha = 0.1;
num = alpha;
denom = [1 -(1-alpha)];
figure
phasedelay(num,denom);
hold on
```

```
alpha = 0.3;
num = alpha;
denom = [1 -(1-alpha)];
phasedelay(num,denom);
hold on

alpha = 0.5;
num = alpha;
denom = [1 -(1-alpha)];
phasedelay(num,denom);
hold on

alpha = 0.9;
num = alpha;
denom = [1 -(1-alpha)];
phasedelay(num,denom);
legend("\alpha=0.1","\alpha=0.3","\alpha=0.5","\alpha=0.9");
```

7. Applying the Exponentially Weighted Moving Average Filter on the IMU Dataset

```
alpha = length(z_pos)/((length(z_pos)+1));
pos_filtered = zeros(1,length(z_pos));
for i = 1:length(z_pos)
    if i == 1
        pos_filtered(i) = z_pos(i);
    else
        pos_filtered(i) = alpha*z_pos(i) + pos_filtered(i-1)*(1-alpha);
    end
end

accel_filtered = zeros(1,length(z_accel));
for i = 1:length(z_accel)
    if i == 1
        accel_filtered(i) = z_accel(i);
    else
        accel_filtered(i) = alpha*z_accel(i) + accel_filtered(i-1)*(1-
alpha);
    end
end

figure
subplot(2,1,1)
plot(z_pos)
title('Raw IMU z-Position Gyroscope Data')
subplot(2,1,2)
plot(pos_filtered)
title('Filtered IMU z-Position Gyroscope Data')

figure
subplot(2,1,2)
plot(z_accel)
title('Raw IMU z-Acceleration Data')
subplot(2,1,1)
plot(accel_filtered)
title('Filtered IMU z-Acceleration Data')
```