# Lab 3

Omar Ramos Escoto

*Due: May 31st, 2023*

ME 220
Introduction to Sensors
Spring 2023

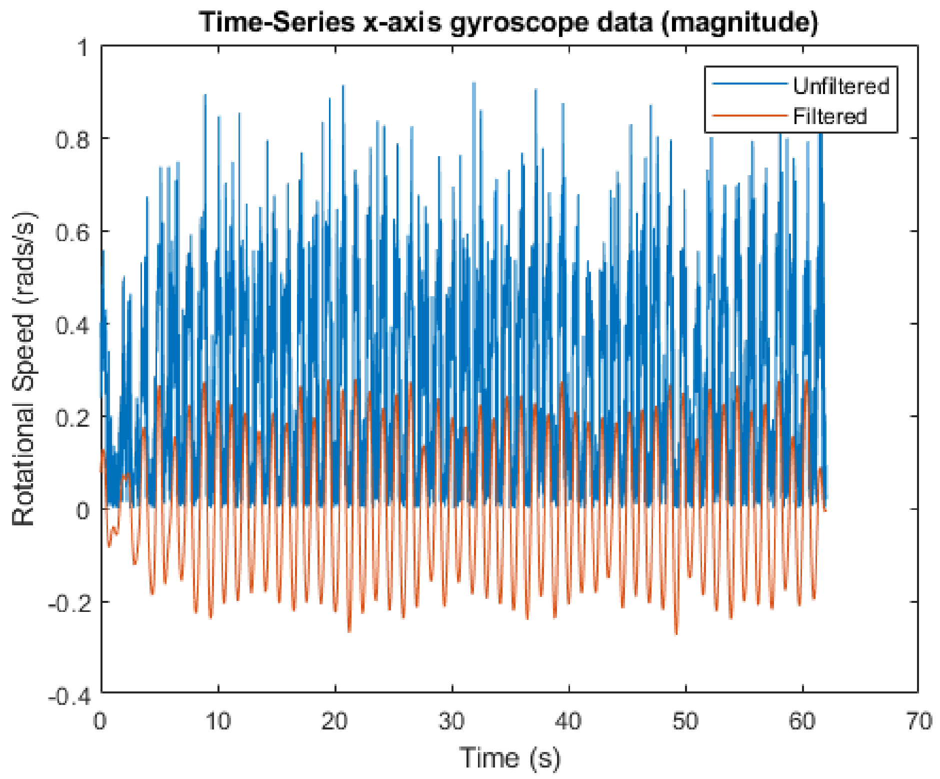
## Part 1: Bike Computer

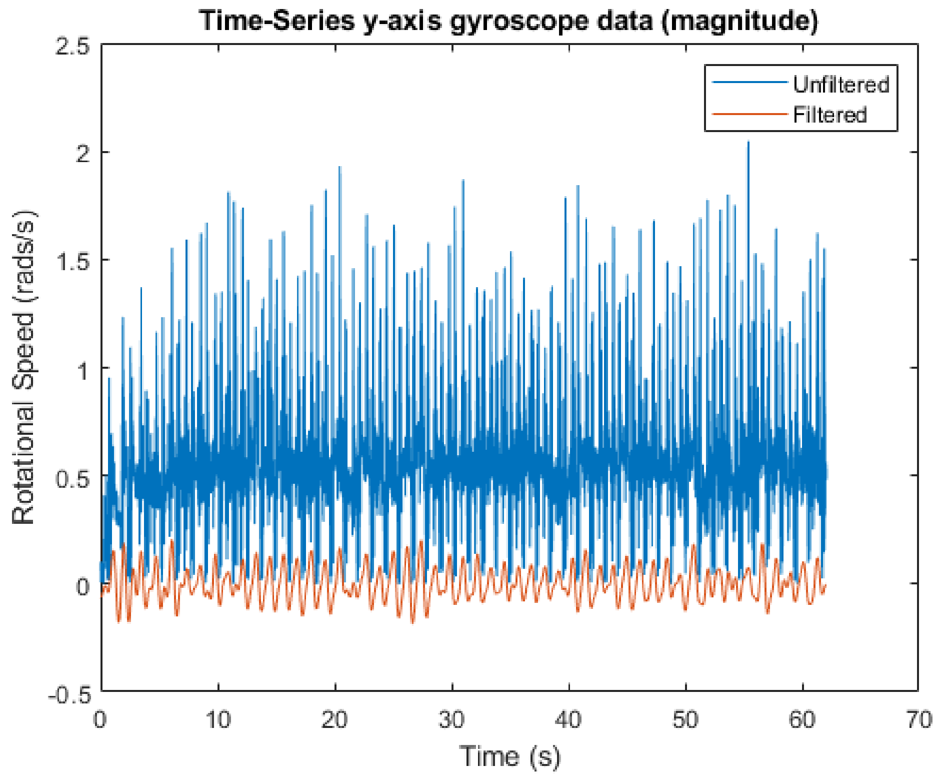
**Q5.** Filtering:

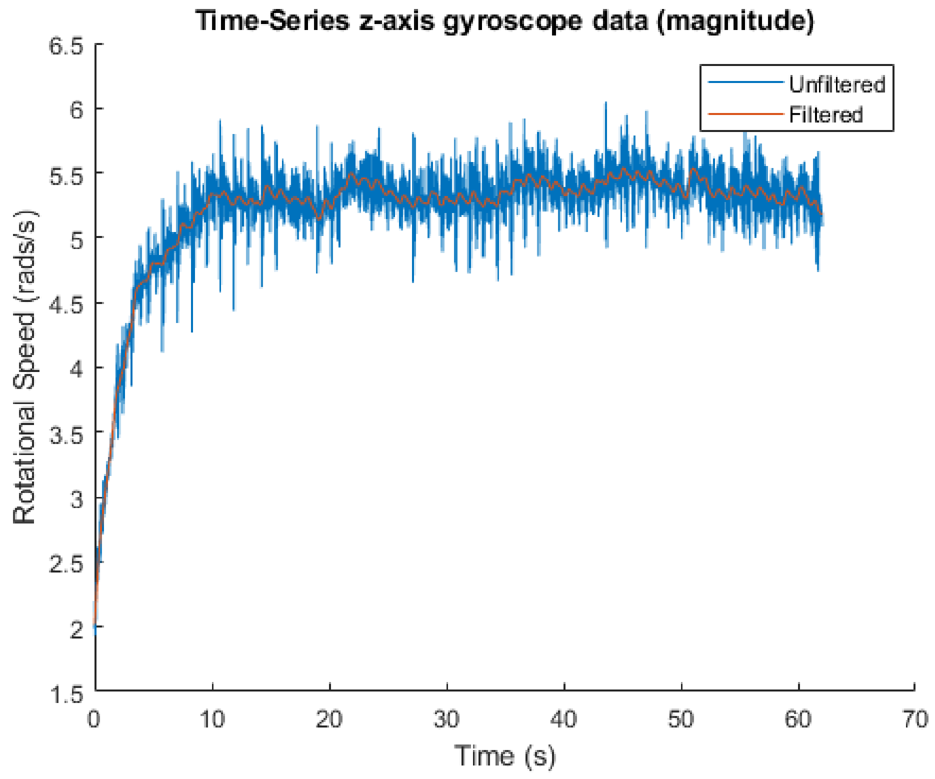
*Gyroscope Data:*

For the x and y axes, I chose a 4th order Butterworth Filter in bandpass mode since this type of filter is often used for filtering motion data and has worked very well in past labs. Bandpass filtering was useful here because I did not care about any constant offsets since there should be minimal rotation data in these axes, so it is mainly useful to measure oscillations and noise in these directions. The order was chosen empirically, with 4th order providing plenty of attenuation graphically for the non-desired frequencies. I chose the cutoff frequencies by plotting the PSD of the signals and evaluating the peak frequencies. The dominating frequency was at around .85 Hz so I set the low frequency cut-off to .4 Hz and high cutoff frequency of 1.3 Hz to get a very smooth signal output.

For the z-axis data, I applied a low pass filter to the data (4th order, Butterworth) in order to better visualize the motion since I want to be able to observe the constant offset to get a sense of the angular velocity and its oscillations at any time. The high frequency cut off was left at 1.3 Hz given the PSD analysis of the signal. This low-pass filtered data is what I will use for the rest of the lab.



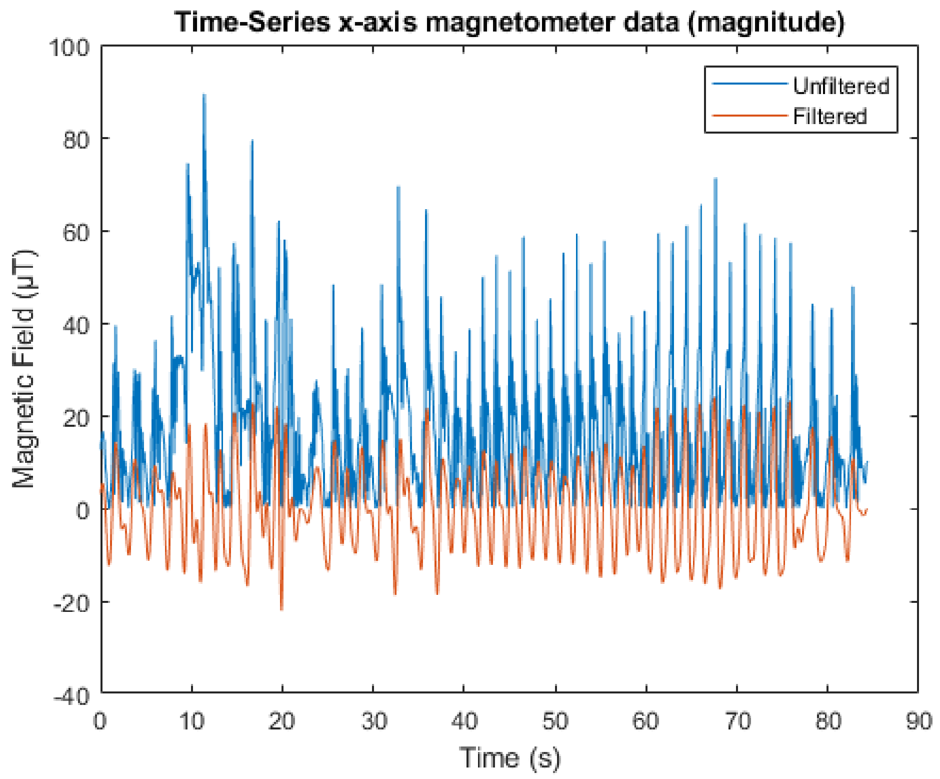

Time-Series y-axis gyroscope data (magnitude)
Unfiltered
Filtered
Rotational Speed (rads/s)
Time (s)
2.5
2
1.5
1
0.5
0
-0.5
0
10
20
30
40
50
60
70

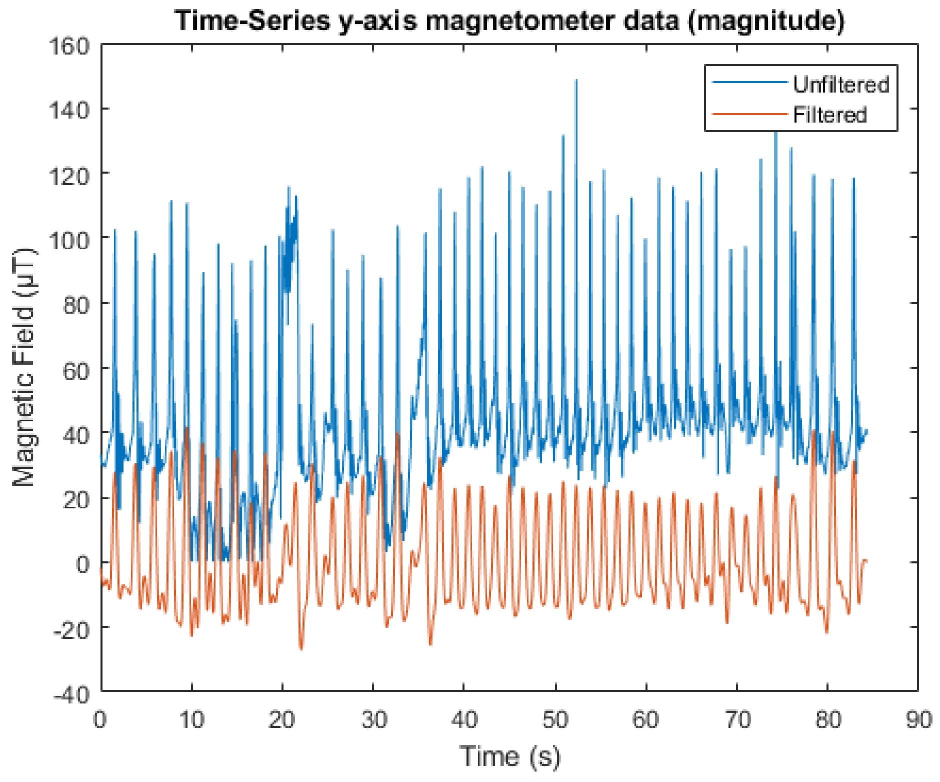

Time-Series z-axis gyroscope data (magnitude)
Unfiltered
Filtered
Rotational Speed (rads/s)
Time (s)
6.5
6
5.5
5
4.5
4
3.5
3
2.5
2
1.5
0
10
20
30
40
50
60
70

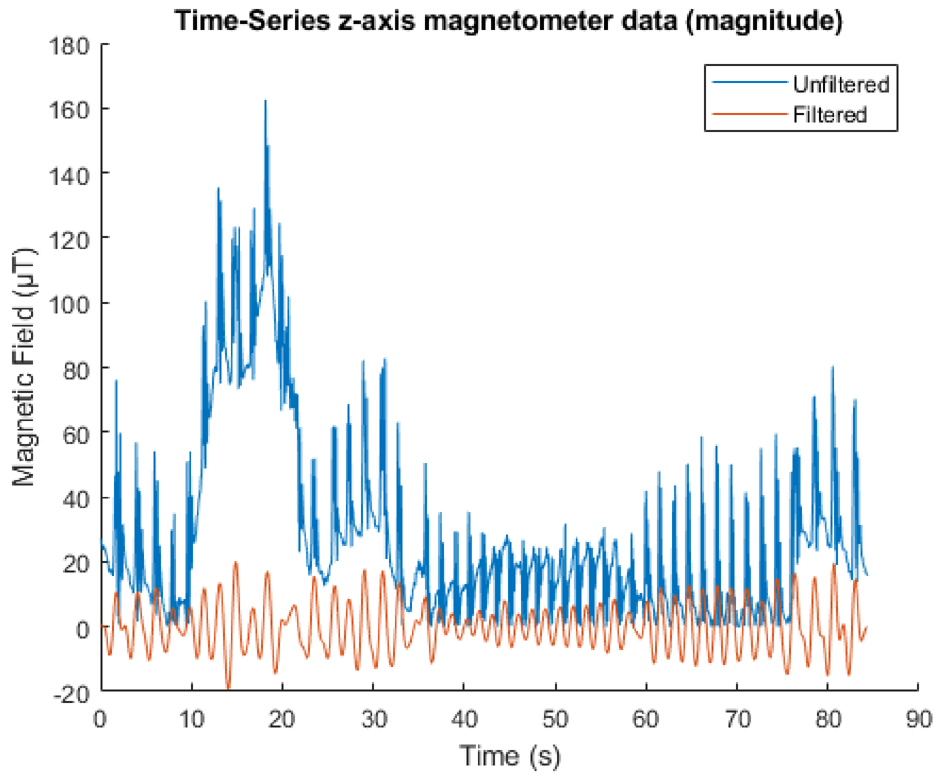
From the resulting plots, we can see that most of the information is coming from the z-axis, as expected since the rotation is along this axis. The data from the other axes seems to follow the same main frequency, but the magnitude of the signals is much smaller. The z-axis data shows the speeding up of the driver and signs of the pedaling being done to keep an angular velocity of around 5 rads/s.

*Magnetometer Data:*

Similarly to the x and y axes for the gyroscope data, I chose a 4th order Butterworth Filter in bandpass mode since this type of filter is often used for filtering motion data and has worked very well in past labs. Bandpass filtering was useful here because I did not care about any constant offsets since this data is useful for measuring oscillations as a way to count the number of rotations. Therefore, it is the relative changes in the magnetic field that matter and therefore only the characteristic frequencies of the motion needs to go through. The order was chosen empirically, with 4th order providing plenty of attenuation graphically for the non-desired frequencies. I chose the cutoff frequencies by plotting the PSD of the signals and evaluating the peak frequencies. The dominating frequency was at around .65 Hz so I set the low frequency cut-off to .3 Hz and high cutoff frequency of 1 Hz to get a very clean signal output of the peaks in the signal.




Time-Series y-axis magnetometer data (magnitude)
Unfiltered
Filtered
Magnetic Field (μT)
Time (s)
160
140
120
100
80
60
40
20
0
-20
-40
0
10
20
30
40
50
60
70
80
90


Time-Series z-axis magnetometer data (magnitude)
Unfiltered
Filtered
Magnetic Field (μT)
Time (s)
180
160
140
120
100
80
60
40
20
0
-20
0
10
20
30
40
50
60
70
80
90

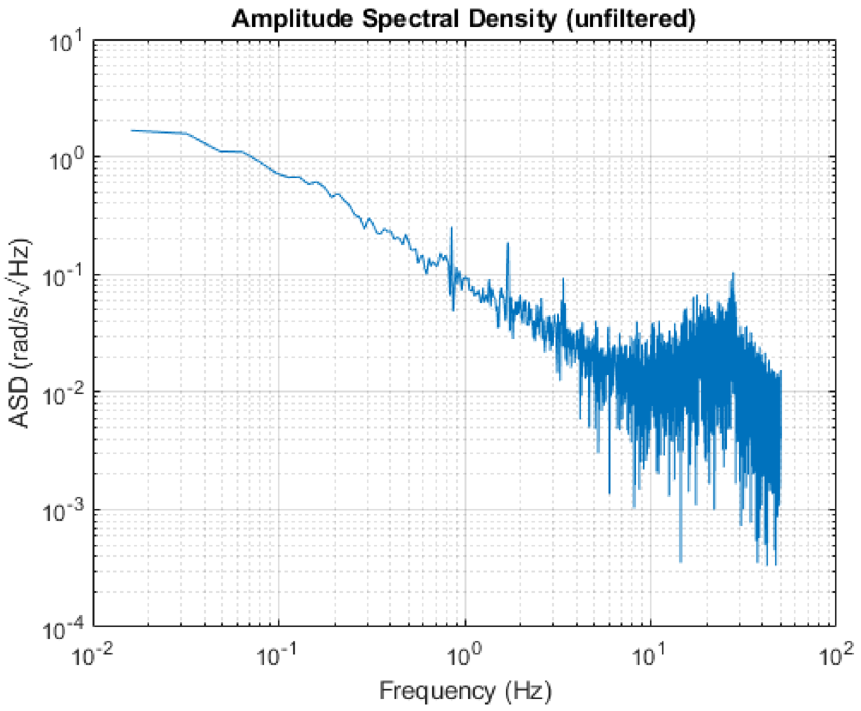
From the resulting plots, we can see that all of the plots exhibit a characteristic frequency of motion. The z-axis data seems to be quite noisy, while the x and y data are a little more comparable. The magnitudes of the magnetic field are comparable across the different axes. Since I will be counting peaks for this data, the x and y axes will most likely be the most useful.

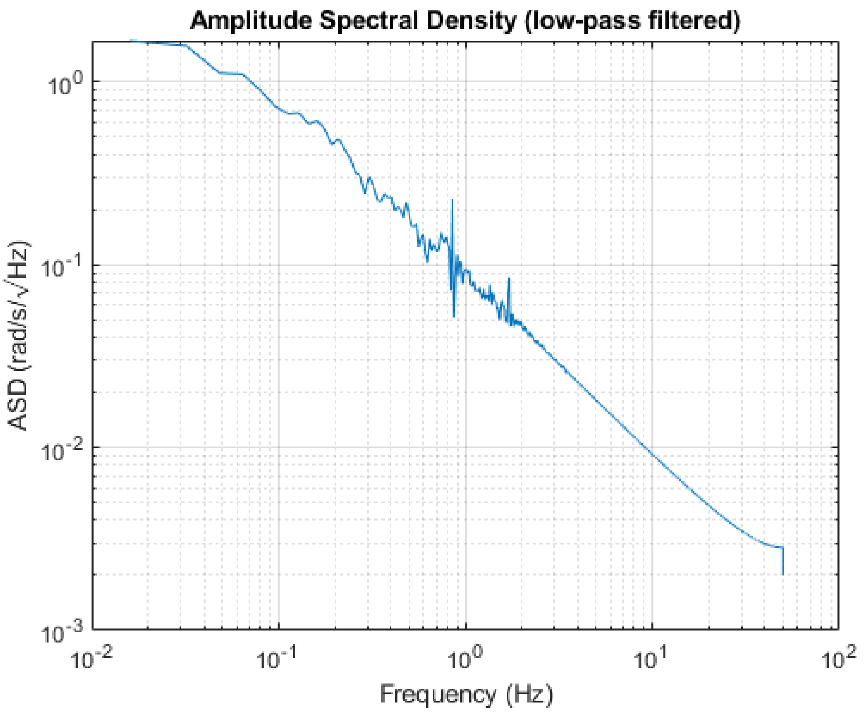
Although I did use the same filter type and order, the z-axis gyroscope data required a low pass filter in order to retain important information. Bandpass filters were useful for the rest of the data since I mainly cared about smoothing out and evaluating the oscillations.

**Q6.** ASD of gyroscope data and magnetometer data:

*Gyroscope Data:*

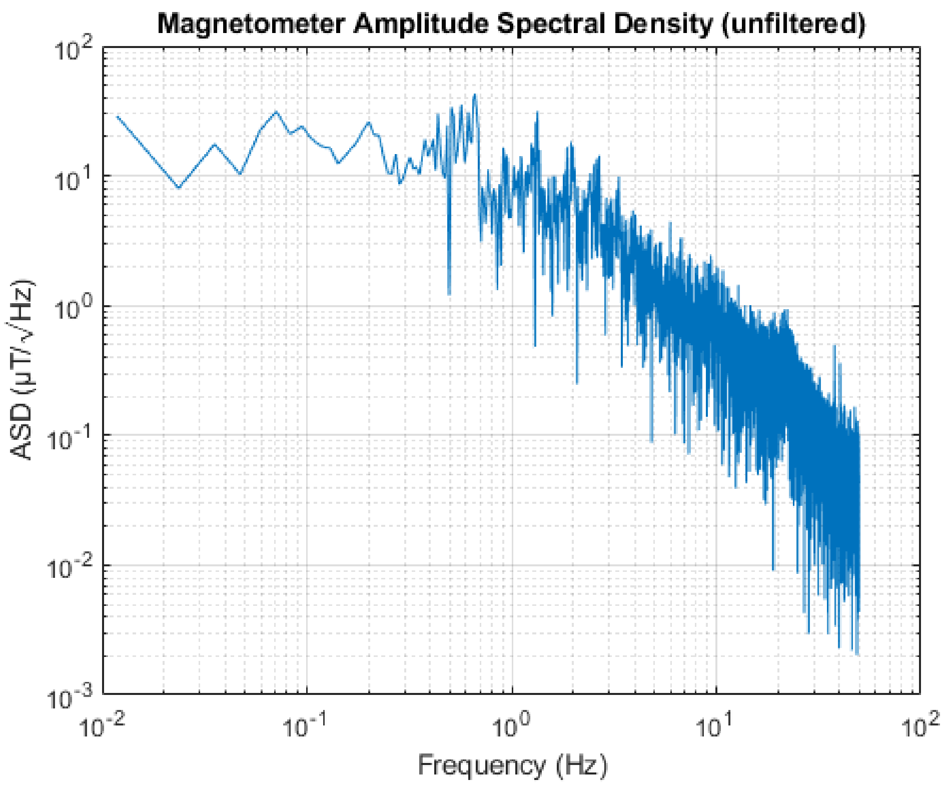
For the gyroscope, I chose the z-axis data since it was the most relevant when it came to bike crank rotation. After plotting the ASD, the highest peak at a reasonable frequency happens at around .85 Hz, so this is the characteristic frequency of the pedaling motion in the gyroscope data. Below is the ASD of the unfiltered z-axis data used to find the characteristic frequency and the ASD after applying a bandpass filter around this frequency.

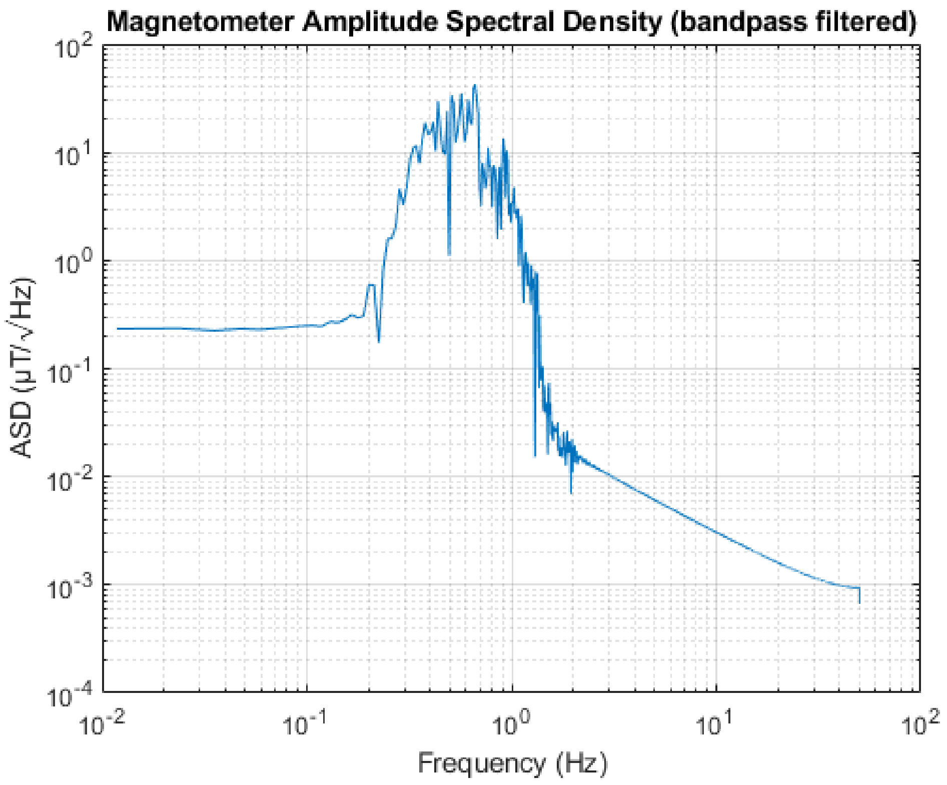




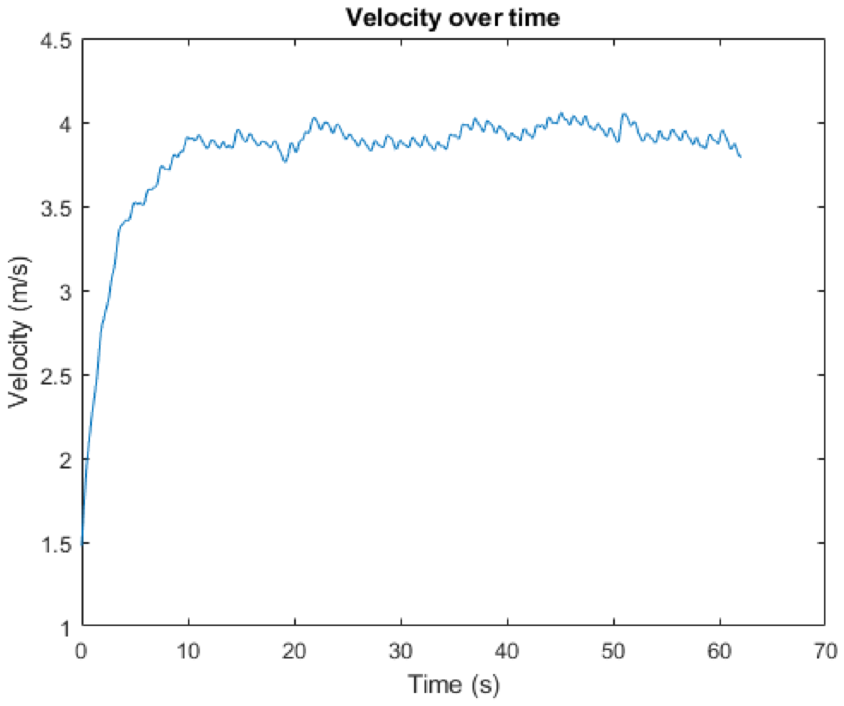
*__Magnetometer Data:__*

For the magnetometer, I chose the magnitude of the x and y axis data since the magnetic field in these directions would see the most change related to the pedaling motion since they are perpendicular to the axis of rotation. However, I mostly chose them because their filtered signals were much cleaner to analyze. After plotting the ASD, the highest peak at a reasonable frequency happens at around .65 Hz, so this is the characteristic frequency of the pedaling motion in the magnetometer data. It differs from the characteristic frequency of the gyroscope because both data sources are different, and the characteristic frequency is related to the frequency of pedaling.

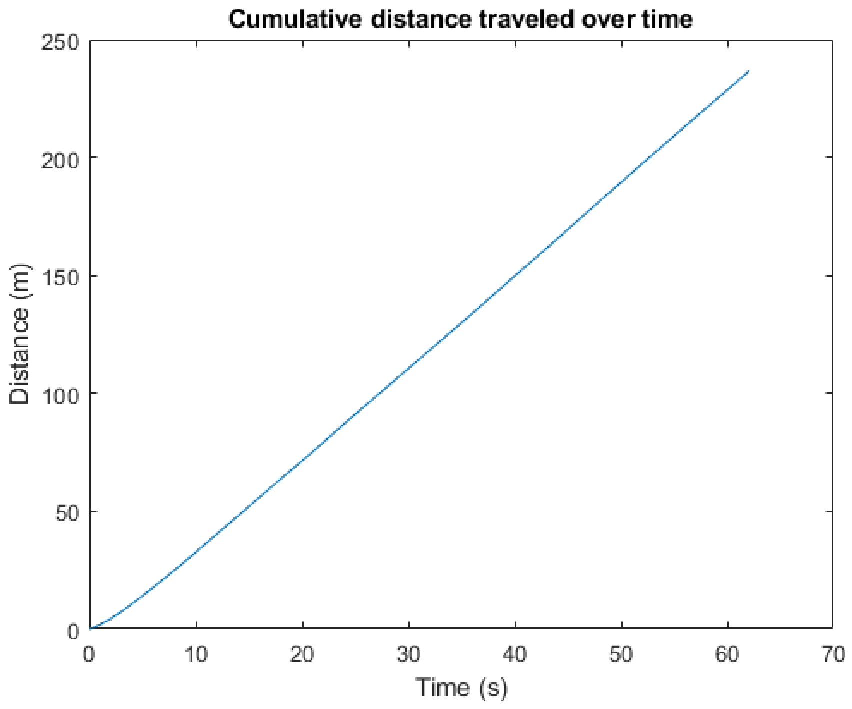

Magnetometer Amplitude Spectral Density (unfiltered)
ASD (µT/√Hz)
Frequency (Hz)
$10^{2}$
$10^{1}$
$10^{0}$
$10^{-1}$
$10^{-2}$
$10^{-3}$
$10^{-2}$
$10^{-1}$
$10^{0}$
$10^{1}$
$10^{2}$


Magnetometer Amplitude Spectral Density (bandpass filtered)
ASD (µT/√Hz)
Frequency (Hz)
$10^{2}$
$10^{1}$
$10^{0}$
$10^{-1}$
$10^{-2}$
$10^{-3}$
$10^{-4}$
$10^{-2}$
$10^{-1}$
$10^{0}$
$10^{1}$
$10^{2}$

**Q7:** I related the gyroscope data to your linear velocity by using the distance traveled per pedal rotation, which is around 4.6 meters. This is equivalent to 4.6 meters per 2π radians, and I can multiply this by the filtered z-axis gyroscope angular velocity data to obtain a graph of linear velocity.

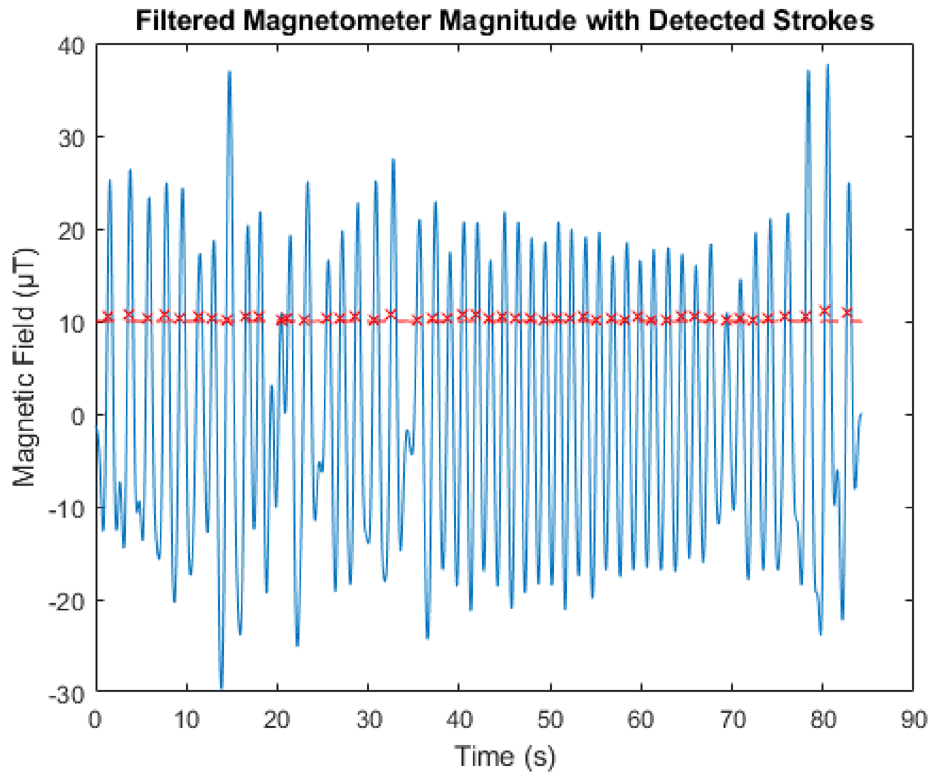


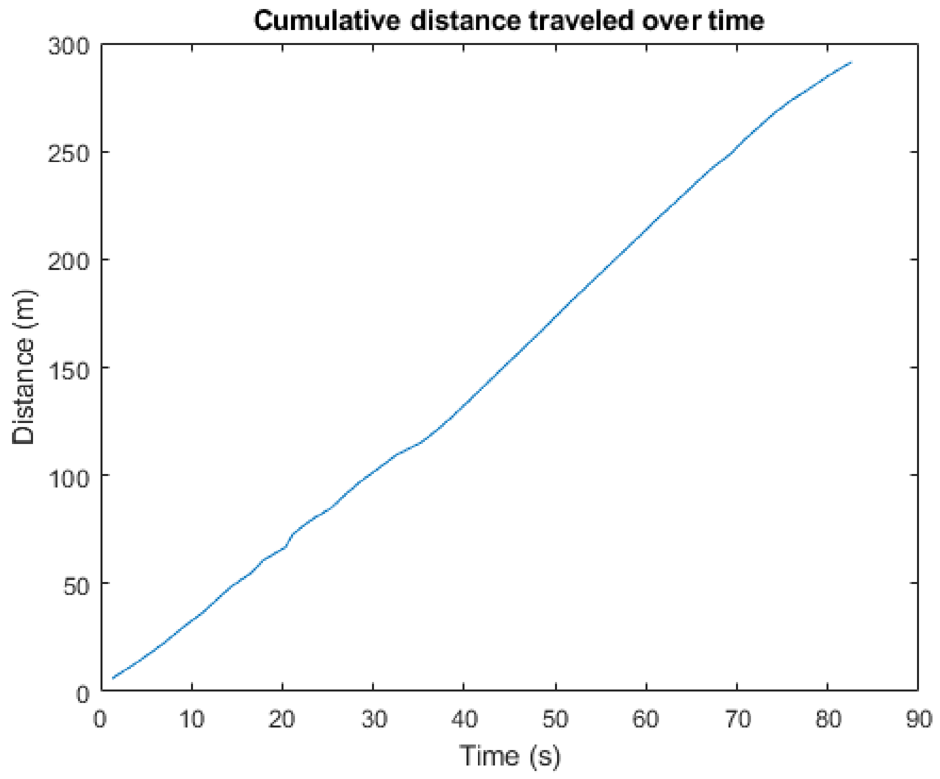
**Q8:** I used the cumulative sum function in matlab and multiplied it by the average time difference between samples to plot the cumulative distance traveled over time, resulting in approximately 236 meters. There was no need to correct for noise after the filtering done in the earlier steps and manually trimming the data at the beginning and end of the pedaling.



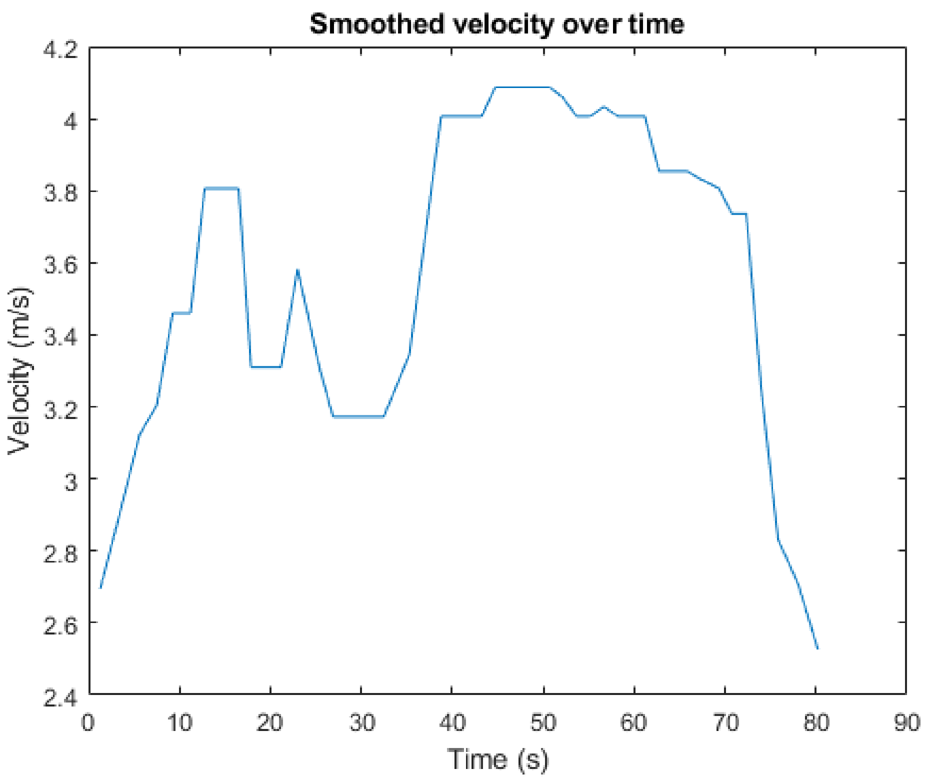
This result is very reasonable and close to the actual value. The cumulative distance from the data is 236 meters, which is very close to the actual distance of 237 meters. Given that I most likely trimmed a little extra data from the beginning and end, the value being so close indicates that the filtering was very effective.

**Q9:** My algorithm uses a rising edge strategy to count the likely number of strokes detected through the filtered data. I first used the gear ratio and diameter of the wheel to calculate the distance covered by a single pedal stroke. Then I set a threshold of a 10 μT relative change in magnetometer magnitude and counted how many times the data crossed this threshold and counted these as strokes. I could then get the times at which the strokes happened and add a wheel rotation every time a stroke happened. The graph below shows the detected strokes and distance plotted. The distance measured was around 292 meters. I don't actually know the distance though since this is not my own data but it seems reasonable when compared to the magnetometer data which has a similar frequency but slightly lower distance travel per stroke.


Filtered Magnetometer Magnitude with Detected Strokes
Magnetic Field (μT)
Time (s)
40
30
20
10
0
-10
-20
-30
0
10
20
30
40
50
60
70
80
90


Cumulative distance traveled over time
Distance (m)
Time (s)
300
250
200
150
100
50
0
0
10
20
30
40
50
60
70
80
90

**Q10.** I calculate the time difference between consecutive pedal strokes using the diff function. The frequency of the pedal strokes is then calculated as the reciprocal of this time difference. The velocity at each time step is calculated by multiplying the frequency by the meters per wheel rotation. The resulting velocity data is then smoothed with a median filtered with a window of 5 data points to help with the noise and this is then plotted over time.

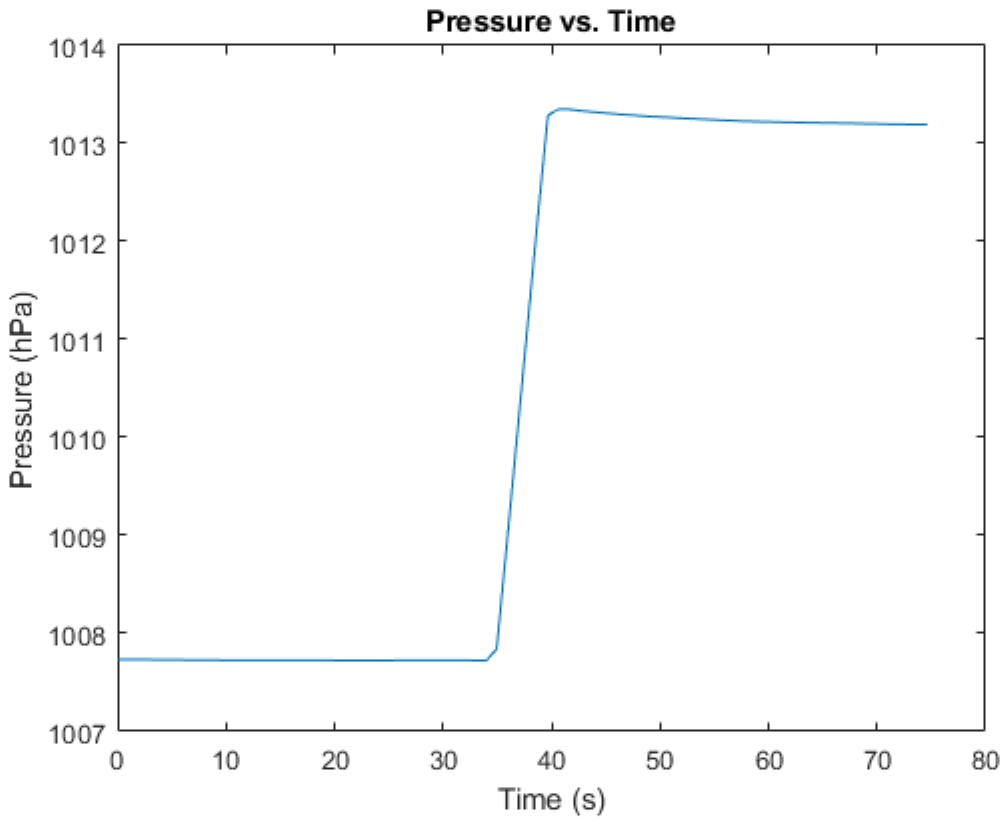


**Q11.** The gyroscope and magnetometer data produced very good plots and approximations for the distance measurements. However, the gyroscope data produced much better results for the velocity measurements. This is most likely due to gyroscope data being easier to filter as well as needing less processing to get to the desired output, whereas the magnetometer data required extracting pedal strokes and then mathematically relate it to the distance and velocity based on the approximations made.

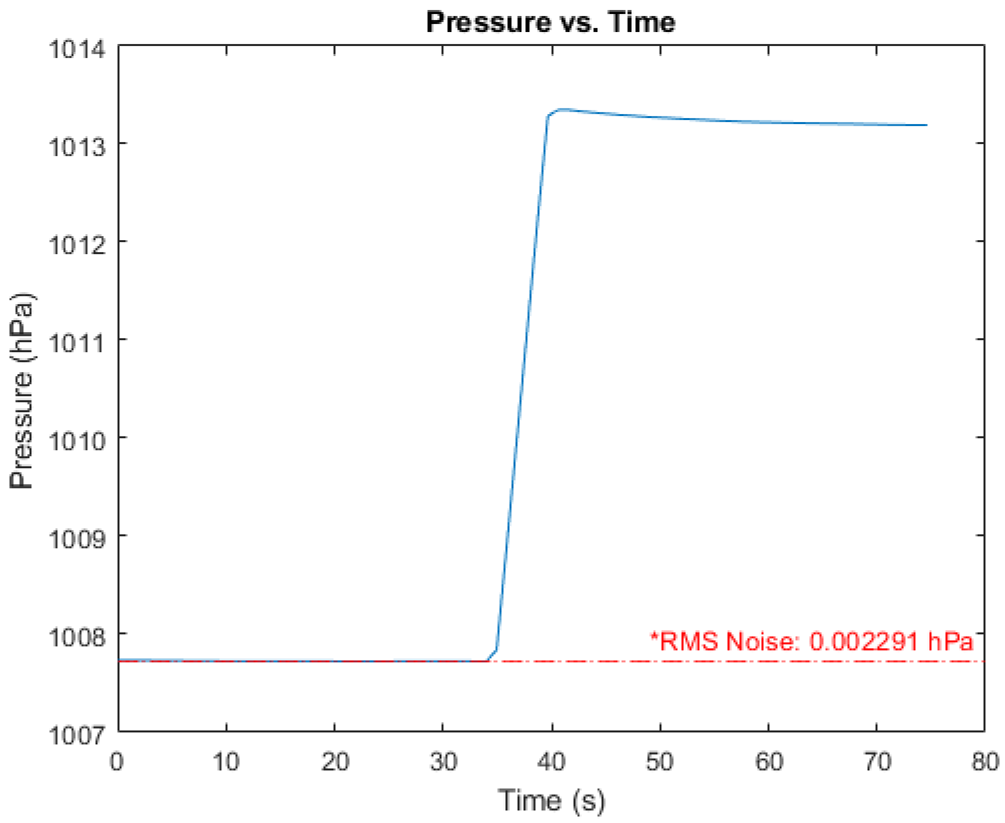
The benefits of the gyroscope include directly measuring the angular velocity of the pedal crank, which made it very easy to relate it to the motion we wanted. It almost provided a continuous direct measure of the desired output apart from a fixed conversion factor related to the wheel radius. However, the gyroscope is very noisy and tends to acquire bias, but heavy filtering mostly fixed this drawback. On the other hand, the magnetometer was able to detect the relative passing of the magnet on the frame and made it easy to calculate the cadence of the pedaling motion. However, it was not very accurate so we needed to combine two axes and needing to process the data heavily and discretize it before being able to get the desired output introduced significant error, especially to the velocity measurement since the data was not very continuous after the need to discretize it into pedal strokes.

## Part 2: Pressure Sensing

**Q4.** Pressure Plot:



**Q5**: Using the data from before the book was placed on the bag, I calculated the **RMS noise of the pressure transducer to be .002291 hPa**. The plot below has this value written, with a dotted line marking the mean RMS value of the pressure before placing the book on the bag (1007.73 hPa).



**Q6.** Using an approximated area of applied pressure of 5 in x 5.5 in, I calculated the **weight of the book to be .9497 kg. The uncertainty in the measurement given the RMS noise previously calculated expressed in mass units is 4.1425e-04 kg.**

**Q7.** Assuming that the initial temperature is somewhere close to 300K, we can expect a **temperature increase of around 1.5K in the bag. Since the temperature coefficient of the sensor is around 1.5 Pa/K, we can therefore expect a change in pressure sensor output of about 2.35 Pa** due to the temperature change, which is insignificant compared to the change in pressure due to the book's weight.

## Table of Contents

# Lab3 Part 1, Gyroscope; Omar Ramos Escoto

```
clear all; close all; clc;

%%%%%%%%%% Part 1  %%%%%%%%%%%%
```

## Problem 1.1-1.4: Get data from Phyphox

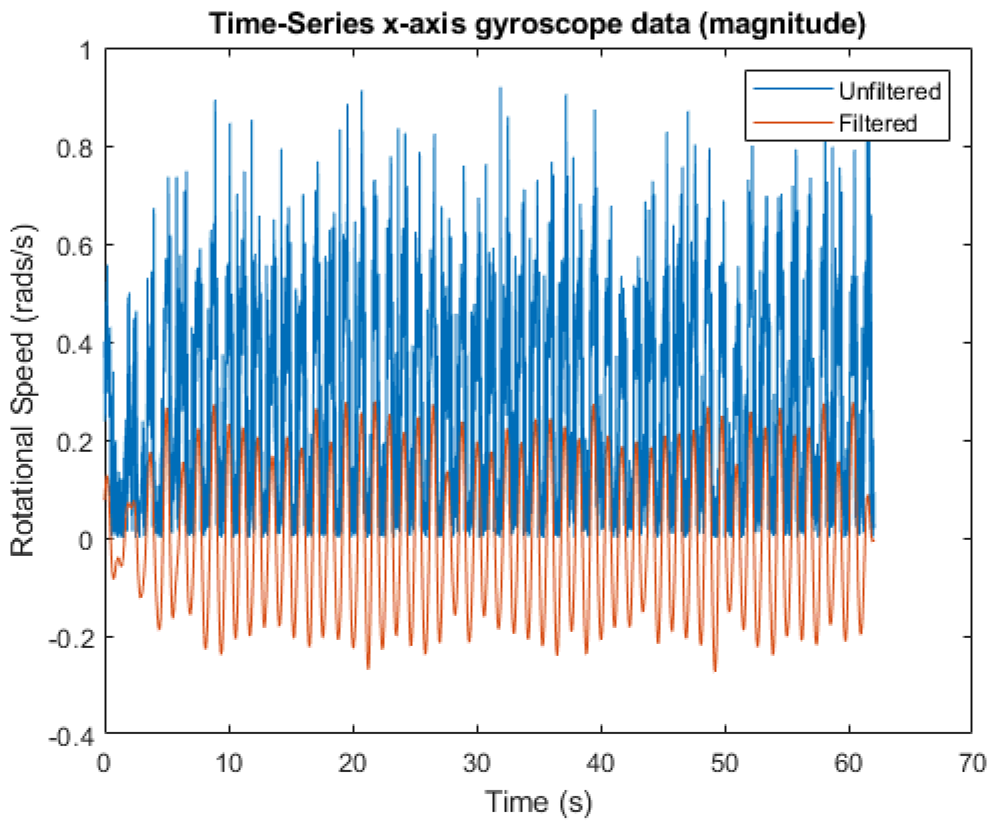
```
T_gyr = readtable("Gyro.xls", 'ReadVariableNames',true);
start_offset = 400;
end_offset = 750;
T_gyr_time = T_gyr.Time_s_(start_offset:end-end_offset) -
 T_gyr.Time_s_(start_offset);
T_gyr_x = T_gyr.X_rad_s_(start_offset:end-end_offset);
T_gyr_y = T_gyr.Y_rad_s_(start_offset:end-end_offset);
T_gyr_z = T_gyr.Z_rad_s_(start_offset:end-end_offset);
```

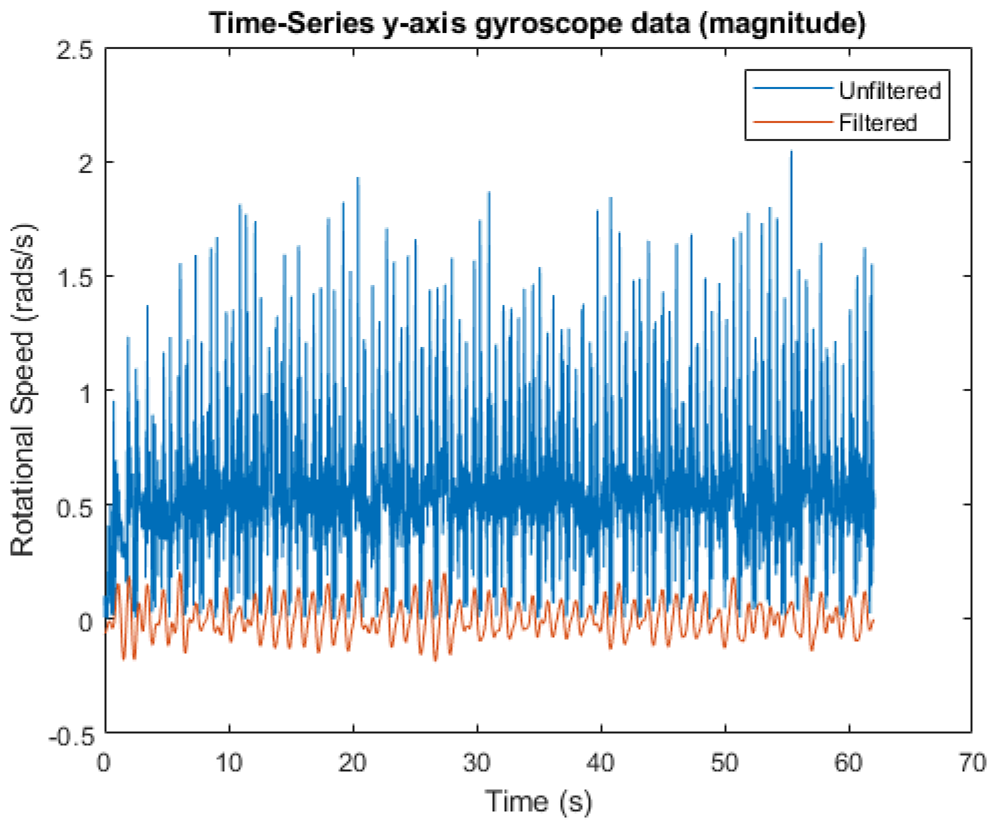
```
Warning: Column headers from the file were modified to make them valid MATLAB
identifiers before creating variable names for the table. The original column
headers are saved in the VariableDescriptions property.
Set 'VariableNamingRule' to 'preserve' to use the original column headers as
table variable names.
```

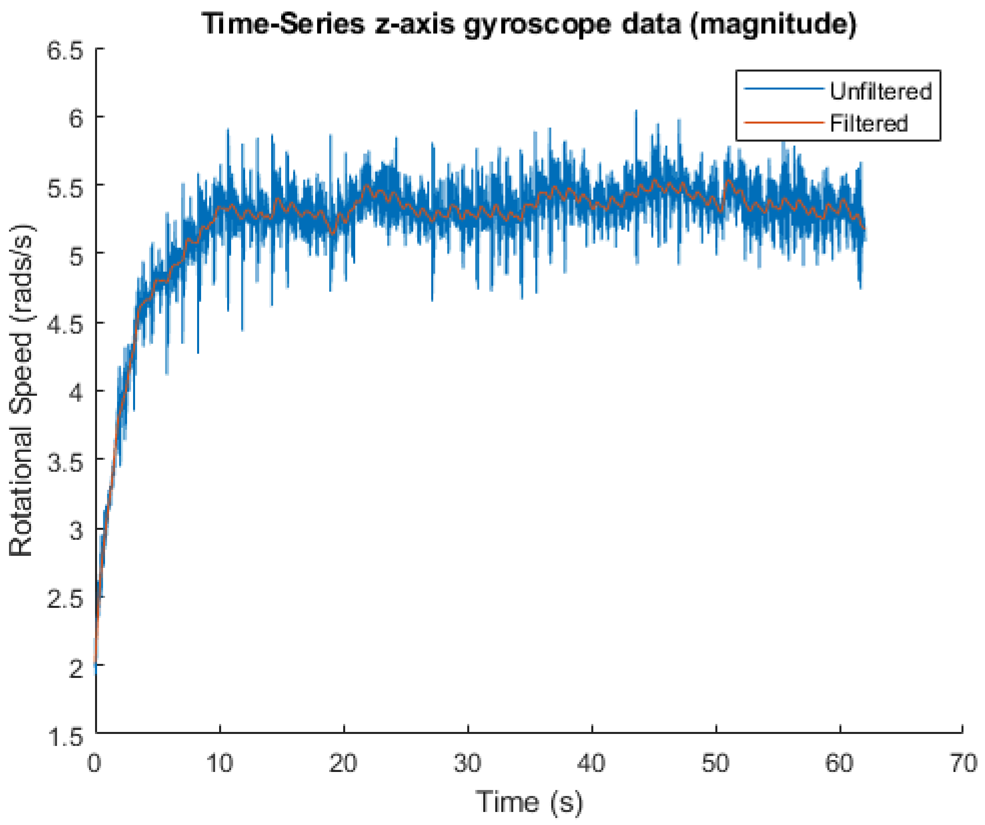
## Problem 1.5 Gyroscope: Filter and plot magnitude of the gyroscope for each axis

```
fs = 100;

% X-axis gyroscope, filtered plot
fcut1 = .4; % Lower cutoff frequency (Hz)
fcut2 = 1.5; % Upper cutoff frequency (Hz)
[b,a] = butter(4, [fcut1 fcut2]/(fs/2), 'bandpass');
magnitude_x_filt = filtfilt(b, a, abs(T_gyr_x));
figure(1);
plot(T_gyr_time, abs(T_gyr_x));
hold on;
plot(T_gyr_time, magnitude_x_filt);
title('Time-Series x-axis gyroscope data (magnitude)');
legend('Unfiltered', 'Filtered')
xlabel('Time (s)');
ylabel('Rotational Speed (rads/s)');
```

```
hold off;

% Y-axis gyroscope, filtered plot
fcut1 = .4; % Lower cutoff frequency (Hz)
fcut2 = 1.5; % Upper cutoff frequency (Hz)
[b,a] = butter(4, [fcut1 fcut2]/(fs/2), 'bandpass');
magnitude_y_filt = filtfilt(b, a, abs(T_gyr_y));
figure(2);
plot(T_gyr_time, abs(T_gyr_y));
hold on;
plot(T_gyr_time, magnitude_y_filt);
title('Time-Series y-axis gyroscope data (magnitude)');
legend('Unfiltered', 'Filtered')
xlabel('Time (s)');
ylabel('Rotational Speed (rads/s)');
hold off;

% Z-axis gyroscope, filtered plot
fcut2 = 1.3; % Upper cutoff frequency (Hz)
[b,a] = butter(2, fcut2/(fs/2), 'low');
magnitude_z_filt = filtfilt(b, a, abs(T_gyr_z));
figure(3);
hold on;
plot(T_gyr_time, abs(T_gyr_z));
plot(T_gyr_time, magnitude_z_filt);
legend('Unfiltered', 'Filtered')
title('Time-Series z-axis gyroscope data (magnitude)');
xlabel('Time (s)');
ylabel('Rotational Speed (rads/s)');
```


Time-Series x-axis gyroscope data (magnitude)
Unfiltered
Filtered
Rotational Speed (rads/s)
Time (s)
1
0.8
0.6
0.4
0.2
0
-0.2
-0.4
0
10
20
30
40
50
60
70


Time-Series y-axis gyroscope data (magnitude)
Unfiltered
Filtered
Rotational Speed (rads/s)
Time (s)
2.5
2
1.5
1
0.5
0
-0.5
0
10
20
30
40
50
60
70

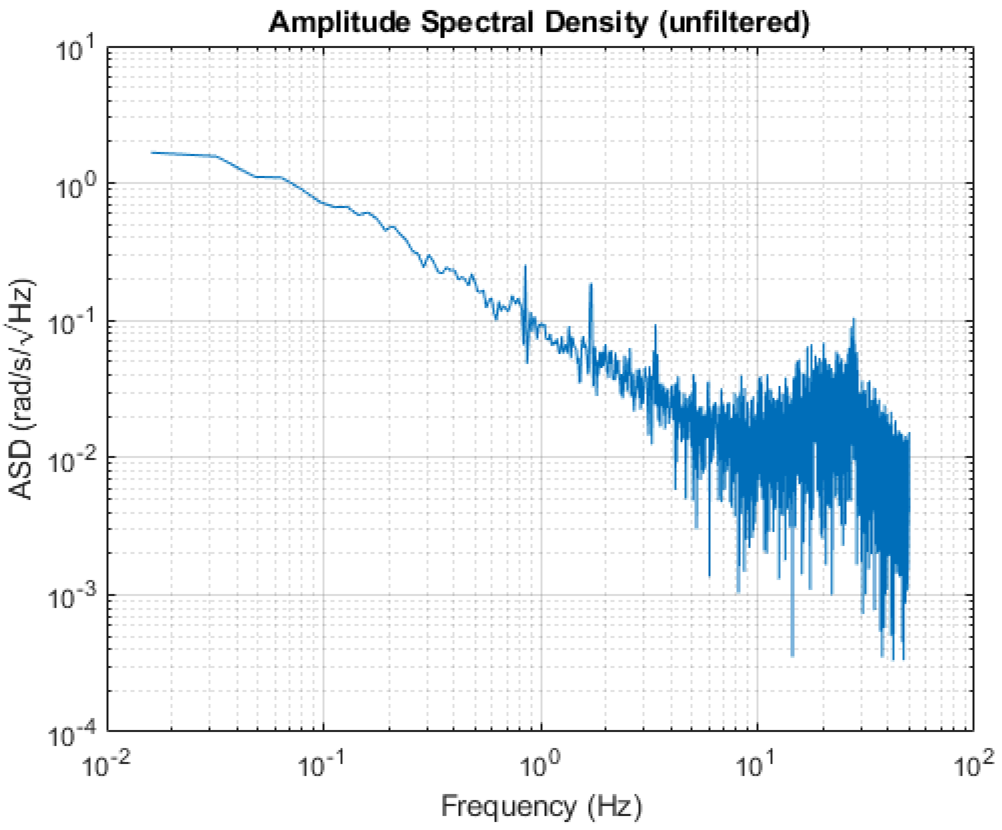


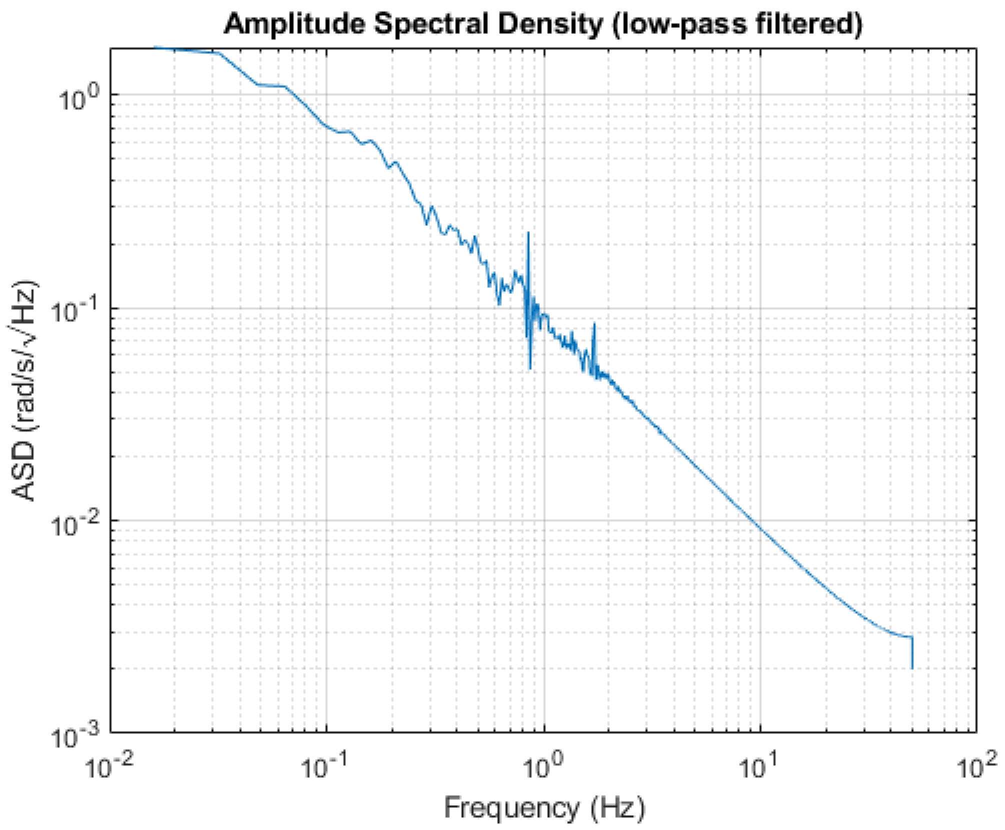
## Problem 1.6 Gyroscope: ASD

```
N = length(T_gyr_z);

% ASD Gyroscope (unfiltered)
mdft = fft(T_gyr_z);
mdft = mdft(1:N/2+1);
psdm = (1/(fs*N)) * abs(mdft).^2;
psdm(2:end-1) = 2*psdm(2:end-1);
freq = 0:fs/length(T_gyr_z):fs/2;
figure(4);
loglog(freq, sqrt(psdm));
grid on
title('Amplitude Spectral Density (unfiltered)');
xlabel('Frequency (Hz)');
ylabel('ASD (rad/s/\surd{Hz})');

% ASD Gyroscope (filtered)
mdft = fft(magnitude_z_filt);
mdft = mdft(1:N/2+1);
psdm = (1/(fs*N)) * abs(mdft).^2;
psdm(2:end-1) = 2*psdm(2:end-1);
freq = 0:fs/length(magnitude_z_filt):fs/2;
figure(5);
loglog(freq, sqrt(psdm));
```

```
grid on
title('Amplitude Spectral Density (low-pass filtered)');
xlabel('Frequency (Hz)');
ylabel('ASD (rad/s/\surd{Hz})');
```

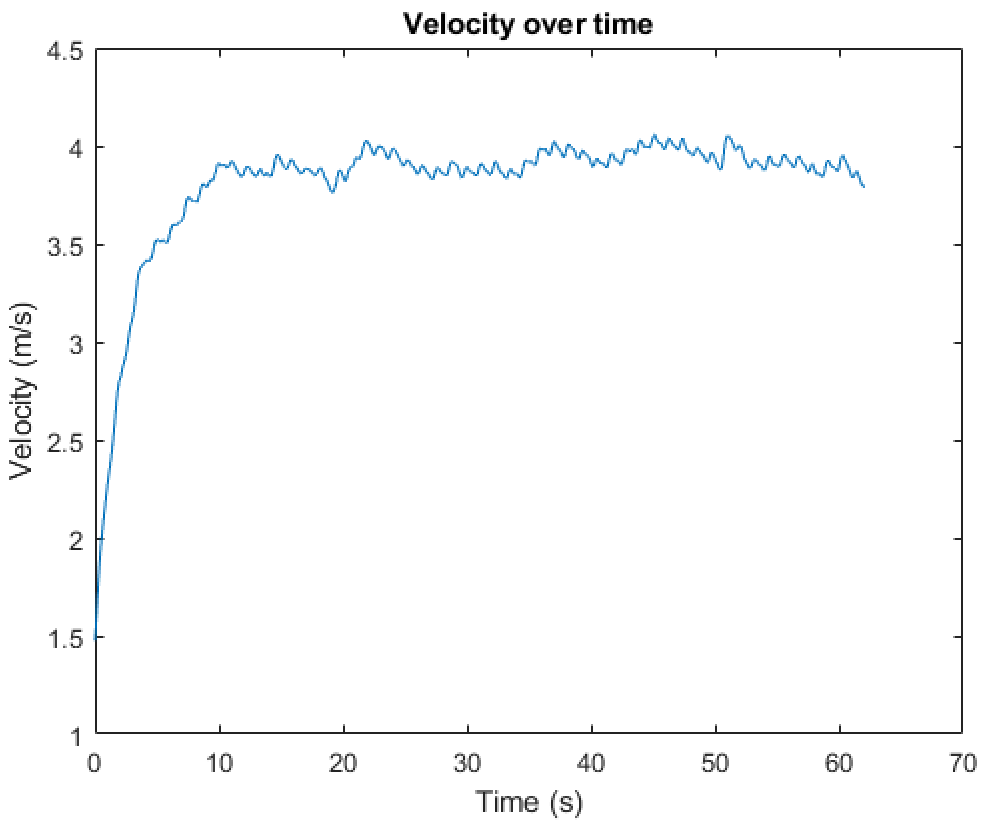
```
Warning: Integer operands are required for colon operator when used as index.
Warning: Integer operands are required for colon operator when used as index.
```





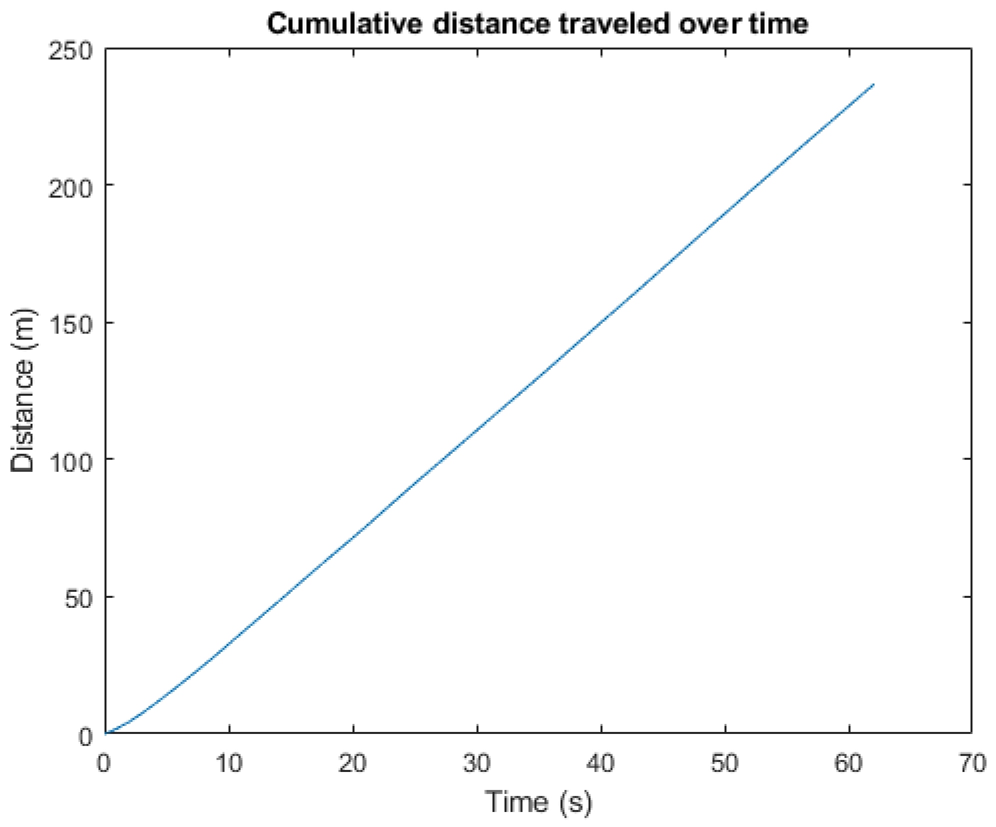
## Problem 1.7: Gyroscope -> Linear Velocity

```
meters_per_rotation = 4.6;

% Convert the filtered gyroscope data from rotations per second to linear
 velocity
velocity = magnitude_z_filt * (meters_per_rotation/(2*pi));

% Plot the results
figure(6);
plot(T_gyr_time, velocity);
title('Velocity over time');
xlabel('Time (s)');
ylabel('Velocity (m/s)');
```



## Problem 1.8: Distance Traveled

```
% Calculate the cumulative distance traveled
dt = mean(diff(T_gyr_time));
distance = cumsum(velocity) * dt;
figure(7);
plot(T_gyr_time, distance);
title('Cumulative distance traveled over time');
xlabel('Time (s)');
ylabel('Distance (m)');
approx_distance_traveled = distance(end);
```



*Published with MATLAB® R2022b*

## Table of Contents

# Lab3 Part 1, Magnetometer; Omar Ramos Escoto

```
clear all; close all; clc;

%%%%%%%%%% Part 1  %%%%%%%%%%%
```

## Problem 1.1-1.4: Get data from Phyphox

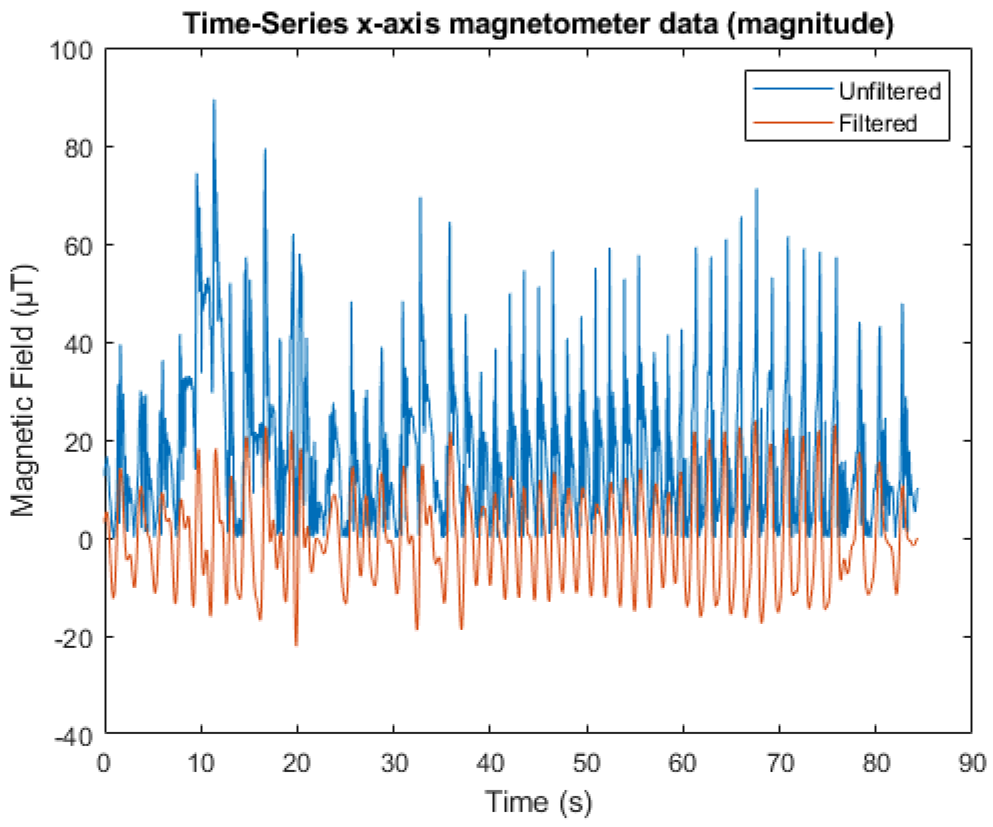
```
T_mag = readtable("Magnetometer.csv", 'ReadVariableNames',true);
start_offset = 400;
end_offset = 500;
T_mag_time = T_mag.Time_s_(start_offset:end-end_offset) -
 T_mag.Time_s_(start_offset);
T_mag_x = T_mag.X__T_(start_offset:end-end_offset);
T_mag_y = T_mag.Y__T_(start_offset:end-end_offset);
T_mag_z = T_mag.Z__T_(start_offset:end-end_offset);
```

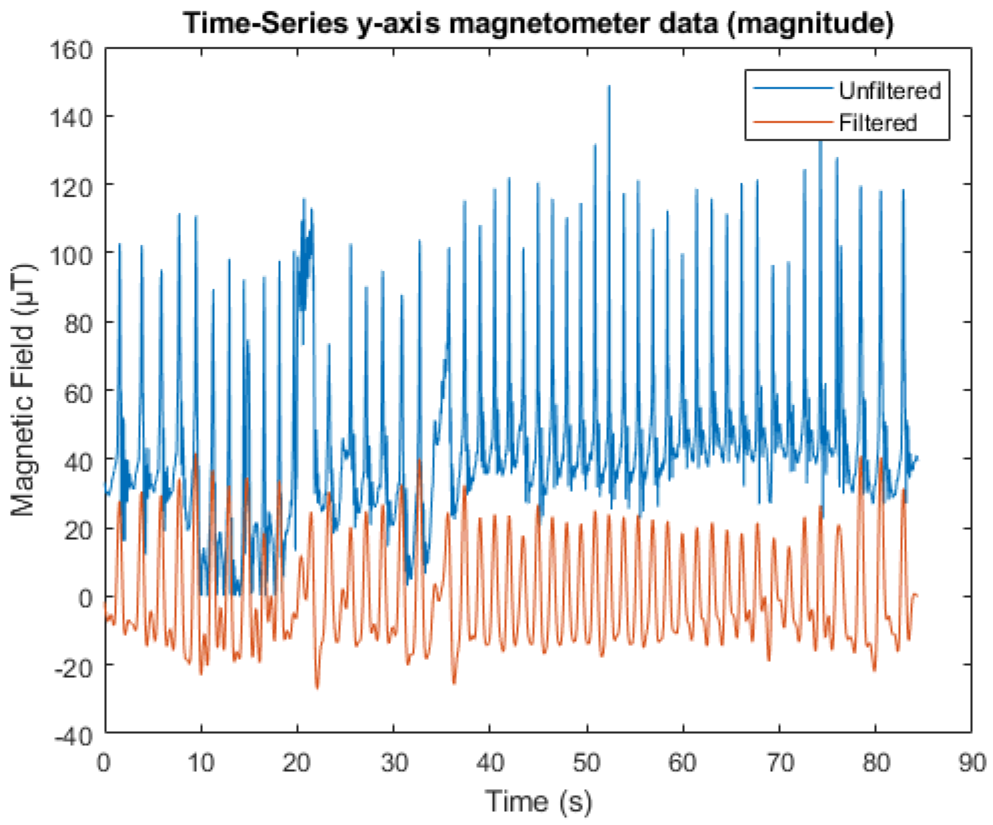
```
Warning: Column headers from the file were modified to make them valid MATLAB
identifiers before creating variable names for the table. The original column
headers are saved in the VariableDescriptions property.
Set 'VariableNamingRule' to 'preserve' to use the original column headers as
table variable names.
```

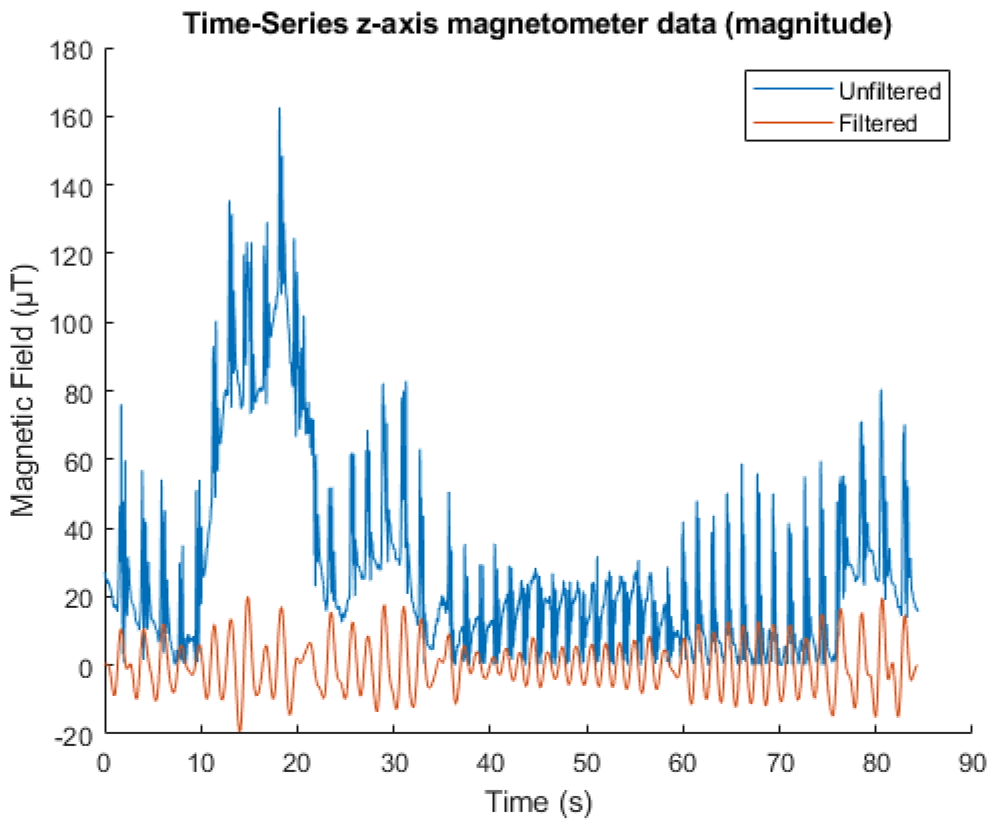
## Problem 1.5 Magnetometer: Filter and plot magnitude of the gyroscope for each axis

```
fs = 100;

% X-axis magnetometer, filtered plot
fcut1 = .3; % Lower cutoff frequency (Hz)
fcut2 = 1.3; % Upper cutoff frequency (Hz)
[b,a] = butter(4, [fcut1 fcut2]/(fs/2), 'bandpass');
magnitude_x_filt = filtfilt(b, a, abs(T_mag_x));
figure(1);
plot(T_mag_time, abs(T_mag_x));
hold on;
plot(T_mag_time, magnitude_x_filt);
title('Time-Series x-axis magnetometer data (magnitude)');
legend('Unfiltered', 'Filtered')
```

```
xlabel('Time (s)');
ylabel('Magnetic Field (μT)');
hold off;

% Y-axis magnetometer, filtered plot
fcut1 = .3; % Lower cutoff frequency (Hz)
fcut2 = 1.3; % Upper cutoff frequnecy (Hz)
[b,a] = butter(4, [fcut1 fcut2]/(fs/2), 'bandpass');
magnitude_y_filt = filtfilt(b, a, abs(T_mag_y));
figure(2);
plot(T_mag_time, abs(T_mag_y));
hold on;
plot(T_mag_time, magnitude_y_filt);
title('Time-Series y-axis magnetometer data (magnitude)');
legend('Unfiltered', 'Filtered')
xlabel('Time (s)');
ylabel('Magnetic Field (μT)');
hold off;

% Z-axis magnetometer, filtered plot
fcut1 = .3; % Lower cutoff frequency (Hz)
fcut2 = 1; % Upper cutoff frequnecy (Hz)
[b,a] = butter(4, [fcut1 fcut2]/(fs/2), 'bandpass');
magnitude_z_filt = filtfilt(b, a, abs(T_mag_z));
figure(3);
hold on;
plot(T_mag_time, abs(T_mag_z));
plot(T_mag_time, magnitude_z_filt);
legend('Unfiltered', 'Filtered')
title('Time-Series z-axis magnetometer data (magnitude)');
xlabel('Time (s)');
ylabel('Magnetic Field (μT)');
hold off;
```


Time-Series x-axis magnetometer data (magnitude)
Magnetic Field (μT)
Time (s)
Unfiltered
Filtered

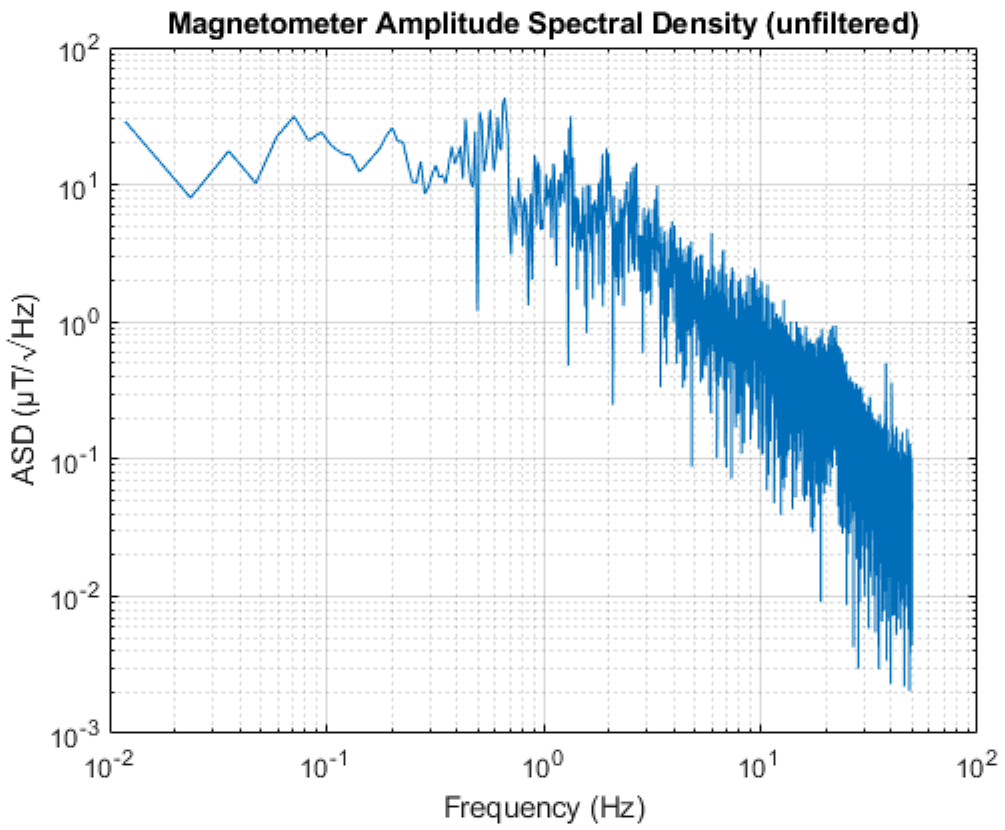

Time-Series y-axis magnetometer data (magnitude)
Magnetic Field (μT)
Time (s)
Unfiltered
Filtered

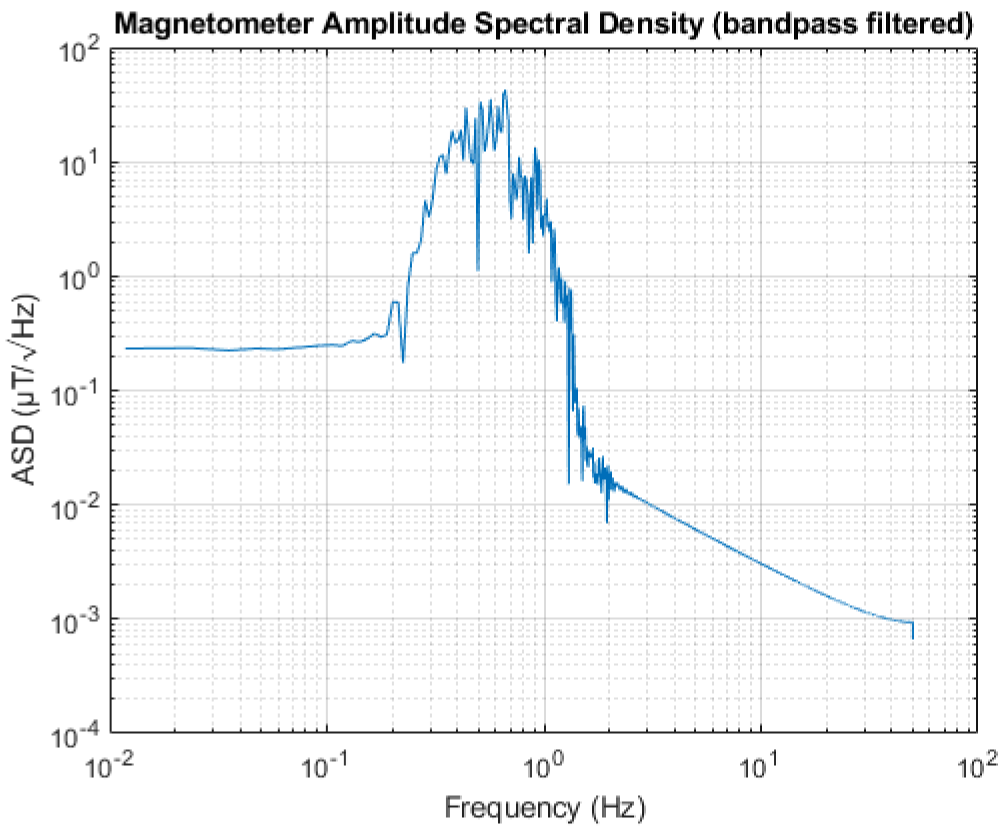


## Problem 1.6 Magnetometer: ASD

```
% Using x and y axes
magnitude_mag = sqrt(T_mag_x.^2 + T_mag_y.^2);
N = length(magnitude_mag);

% ASD Magnetometer (unfiltered)
mdft = fft(magnitude_mag);
mdft = mdft(1:N/2+1);
psdm = (1/(fs*N)) * abs(mdft).^2;
psdm(2:end-1) = 2*psdm(2:end-1);
freq = 0:fs/length(magnitude_mag):fs/2;
figure(4);
loglog(freq, sqrt(psdm));
grid on
title('Magnetometer Amplitude Spectral Density (unfiltered)');
xlabel('Frequency (Hz)');
ylabel('ASD (μT/\surd{Hz})');

% ASD Magnetometer (filtered)
fcut1 = .3; % Lower cutoff frequency (Hz)
fcut2 = 1; % Upper cutoff frequency (Hz)
[b,a] = butter(4, [fcut1 fcut2]/(fs/2), 'bandpass');
magnitude_mag_xy = filtfilt(b, a, magnitude_mag);
mdft = fft(magnitude_mag_xy);
mdft = mdft(1:N/2+1);
psdm = (1/(fs*N)) * abs(mdft).^2;
psdm(2:end-1) = 2*psdm(2:end-1);
freq = 0:fs/length(magnitude_mag_xy):fs/2;
figure(5);
loglog(freq, sqrt(psdm));
grid on
title('Magnetometer Amplitude Spectral Density (bandpass filtered)');
xlabel('Frequency (Hz)');
ylabel('ASD (µT/\surd{Hz})');
```

```
Warning: Integer operands are required for colon operator when used as index.
Warning: Integer operands are required for colon operator when used as index.
```



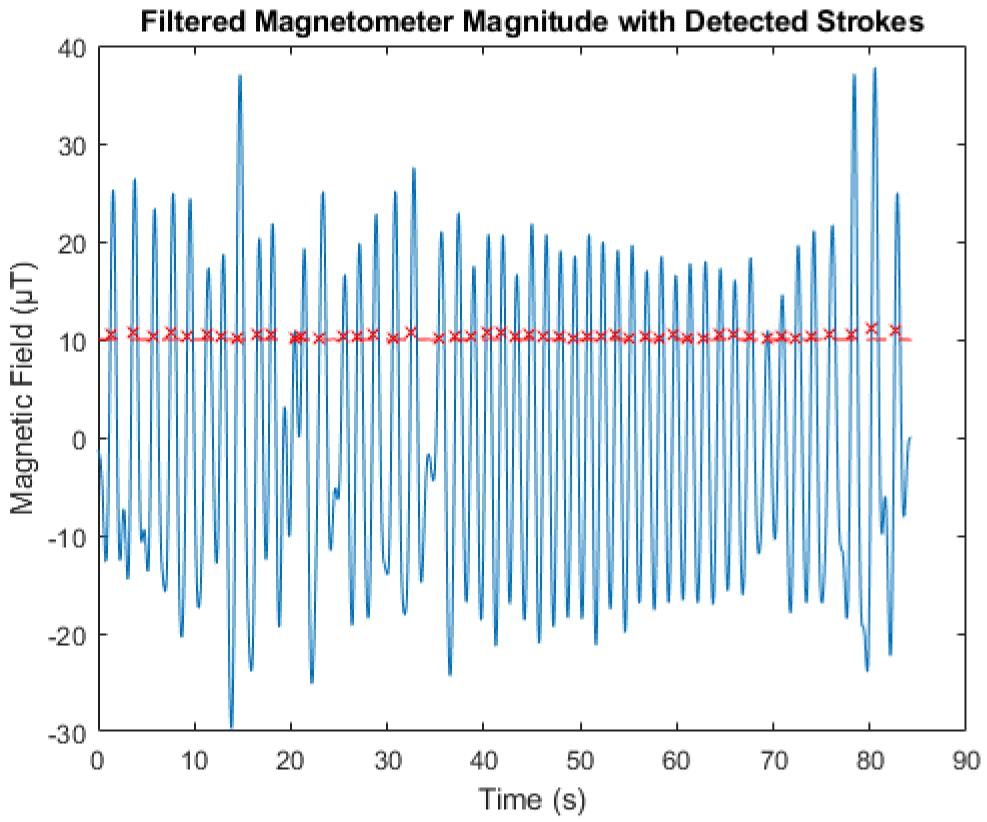


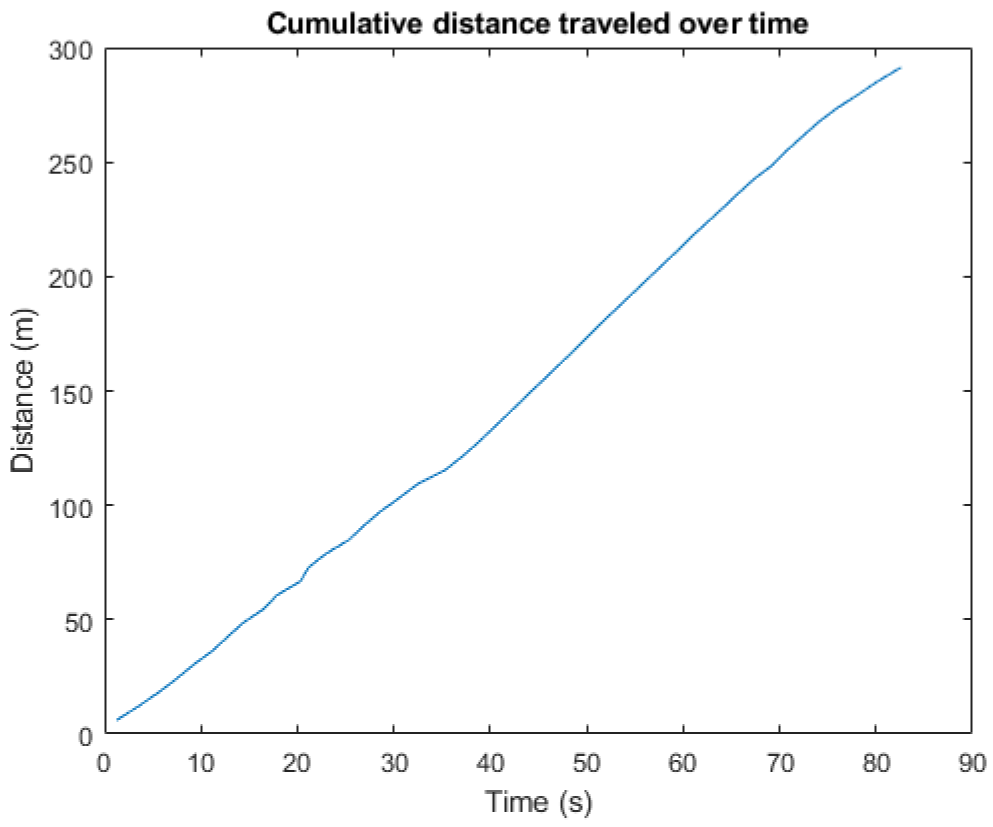
## Problem 2.9: Magnetometer -> Distance

```
wheel_diam = 0.69; % meters
gear_ratio = 2.8;
meters_per_rotation = gear_ratio * wheel_diam * pi;

% Count peaks (rising edges above a emperically set threshold)
threshold = 10; % μT
crossings = diff(magnitude_mag_xy > threshold);
strokes = sum(crossings == 1); % Count positive crossings
fprintf('Number of pedal strokes: %d\n', strokes);

% Find stroke locations
strokes_locs = find(crossings == 1) + 1;

figure(6);
plot(T_mag_time, magnitude_mag_xy);
hold on;
plot(T_mag_time, threshold*ones(size(magnitude_mag_xy)), '--r');
plot(T_mag_time(strokes_locs), magnitude_mag_xy(strokes_locs), 'rx');
hold off;
title('Filtered Magnetometer Magnitude with Detected Strokes');
xlabel('Time (s)');
ylabel('Magnetic Field (μT)');
```

```
% Calculate the time of each pedal stroke
stroke_times = T_mag_time(strokes_locs);

% Calculate the distance traveled at each time step
distance = (1:length(stroke_times))' * meters_per_rotation;

% Plot the results
figure(7);
plot(stroke_times, distance);
title('Cumulative distance traveled over time');
xlabel('Time (s)');
ylabel('Distance (m)');
```

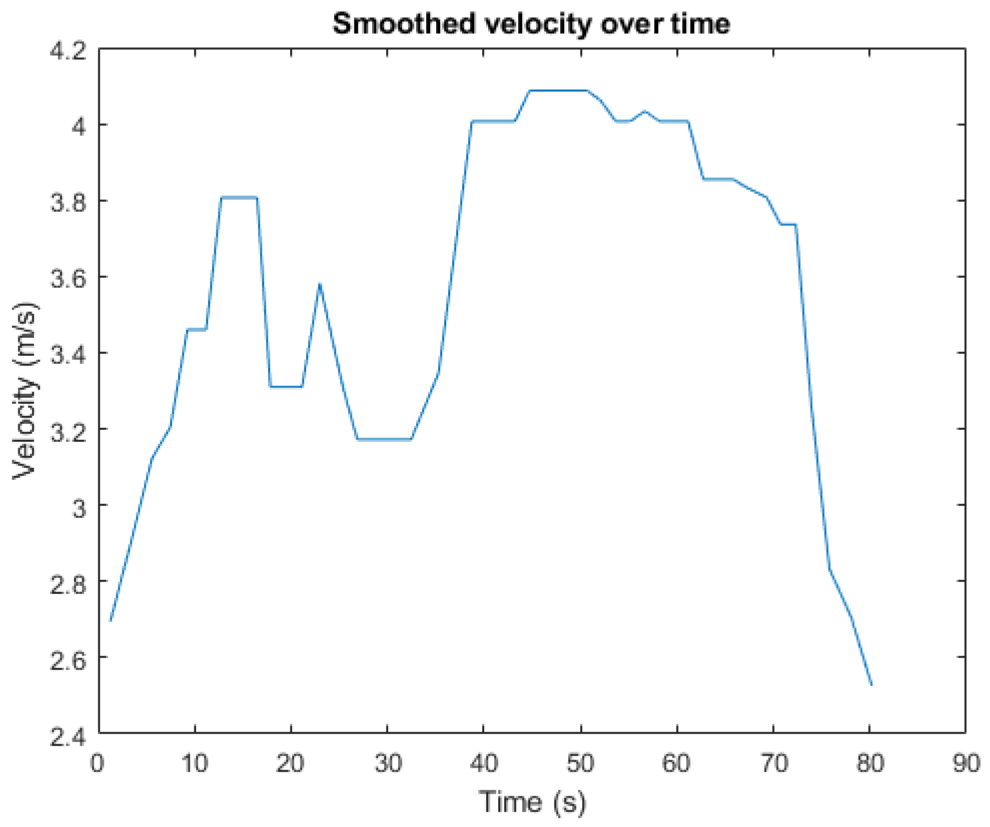
```
Number of pedal strokes: 48
```





## Part 1.10: Magnometer -> Velocity

```
% Calculate the time difference between consecutive pedal strokes
dt = diff(stroke_times);

% Calculate the frequency of the pedal strokes
frequency = 1 ./ dt;

% Calculate the velocity at each time stroke
velocity = frequency * meters_per_rotation;
window_size = 5;
velocity_smooth = medfilt1(velocity, window_size);

% Plot the results
figure(8);
plot(stroke_times(1:end-1), velocity_smooth);
title('Smoothed velocity over time');
xlabel('Time (s)');
ylabel('Velocity (m/s)');
```



*Published with MATLAB® R2022b*

## Table of Contents

# Lab3, Part 2; Omar Ramos Escoto

```
clear all; close all; clc;

%%%%%%%%%% Part 2  %%%%%%%%%%%%
```

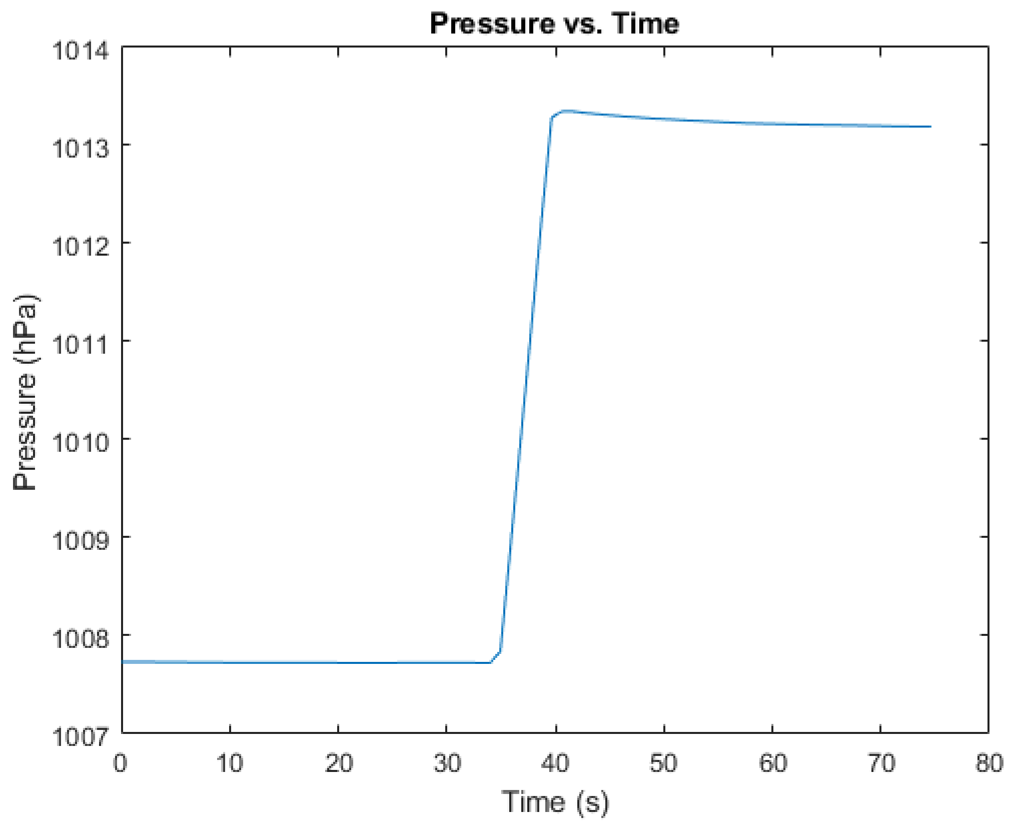
## Problem 2.1-2.3: Get data from Phyphox

```
T_pres = readtable("pressure.xls", 'ReadVariableNames',true);
start_offset = 35;    % Skip the first 35 seconds (inflating, moving)
end_offset = 10;       % Ignore the last 10 seconds (getting phone out of bag)
pres_time = T_pres.Time_s_(start_offset:end-end_offset) -
 T_pres.Time_s_(start_offset);
pres = T_pres.X_hPa_(start_offset:end-end_offset);
pres_noBook = pres(5:25);
pres_Book = pres(40:60);
```

```
Warning: Column headers from the file were modified to make them valid MATLAB
identifiers before creating variable names for the table. The original column
headers are saved in the VariableDescriptions property.
Set 'VariableNamingRule' to 'preserve' to use the original column headers as
table variable names.
```

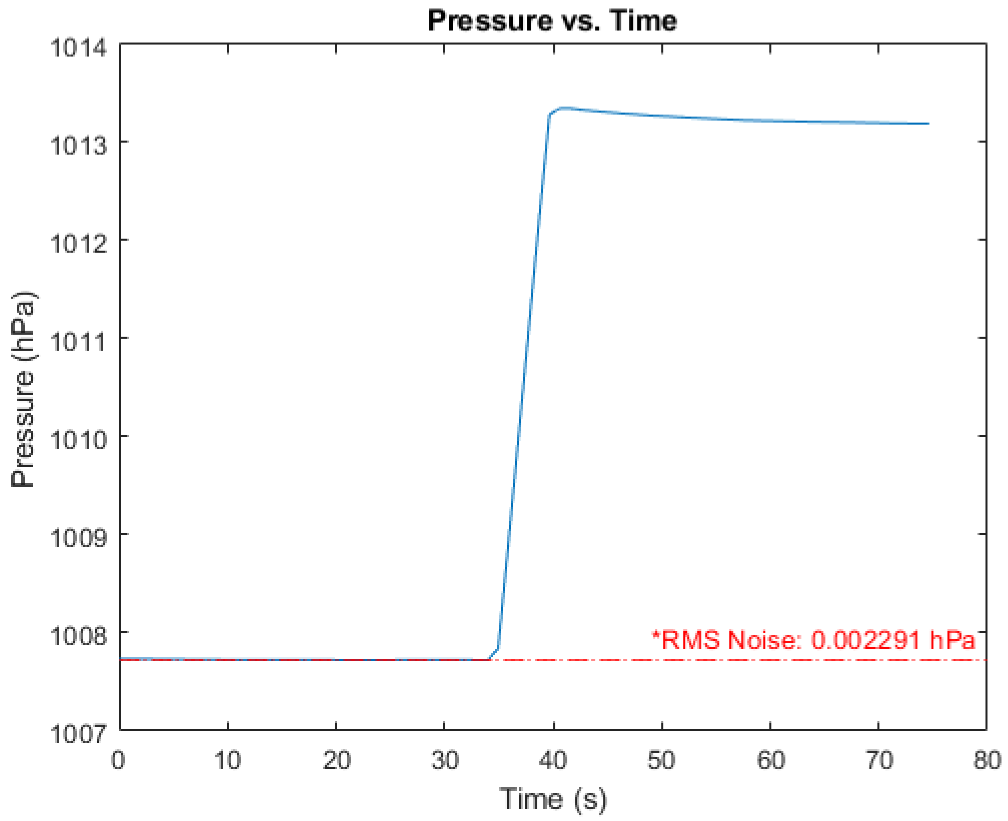
## Problem 2.4: Plot Pressure vs Time

```
figure(1);
plot(pres_time, pres);
title('Pressure vs. Time');
xlabel('Time (s)');
ylabel('Pressure (hPa)');
```



## Problem 2.5: RMS noise

```
pres_noBook_mean = mean(pres_noBook);
pres_noise = rms(pres_noBook - pres_noBook_mean);

% Annotate the RMS noise value on the plot
hold on;
yline(pres_noBook_mean, '--r', ['*RMS Noise: ' num2str(pres_noise) ' hPa']);
% Add bounding lines above and below to indicate the RMS noise
yline(pres_noBook_mean + pres_noise, ':r');
yline(pres_noBook_mean - pres_noise, ':r');
hold off;
```



## Part 2.6: Calculate weight and uncertainty

```
% Calculate the pressure change
delta_p = mean(pres_Book) - pres_noBook_mean; % in hPa

% Convert pressure change from hPa to Pa
delta_p = delta_p * 100; % in Pa

% Calculate the weight of the book
A = 0.127 * 0.1397; % Area over which the pressure is applied (in m^2)
 (approx. 5in x 5.5in)
g = 9.81; % Acceleration due to gravity (in m/s^2)
m_book = (delta_p * A) / g; % Weight of the book (in kg)

% Calculate the uncertainty in the measurement
pres_noise_Pa = pres_noise * 100; % Convert from hPa to Pa
m_noise = (pres_noise_Pa * A) / g; % RMS pressure noise expressed in mass
 units (in kg)
```

## Part 2.7: Temperature Change

```
% Assuming that the initial temperature is a litle above room temp (breath)
 and no volume change
T1 = 300; % K
T2 = mean(pres_Book) * (T1 / pres_noBook_mean);
```

```
delta_T = T2 - T1;

% From datasheet, temperature coefficient offset is 1.5 Pa/K approximately
TCO = 1.5; % in Pa/K
delta_p_Temp = TCO * delta_T; % in Pa
```

*Published with MATLAB® R2022b*