**EMG Filter -12/2/24**

**Algorithm**

Algorithm for the Band-Pass Butterworth IIR digital filter implemented in the provided code:

1. **Initialization**:
   * Initialize the state variables **z1** and **z2** for each filter section to zero.
2. **Filtering Process**:
   * For each input sample:
     + For each filter section:
       - Calculate the intermediate variable **x** using the difference equation of a second-order IIR filter.
       - Update the output using the calculated **x** value and the previous state variables.
       - Update the state variables **z1** and **z2** for the next iteration.
3. **Output**:
   * Return the filtered output.

Here's a breakdown of the steps within the filtering process:

* For each filter section:
  1. Calculate the intermediate variable **x** using the difference equation:

x = input - a1 \* z1 - a2 \* z2

where **input** is the current input sample, **z1** and **z2** are the previous state variables, and **a1** and **a2** are the filter coefficients.

* 1. Update the output using the calculated **x** value and the previous state variables:

output = b0 \* x + b1 \* z1 + b2 \* z2

where **b0**, **b1**, and **b2** are the filter coefficients for the output.

* 1. Update the state variables **z1** and **z2** for the next iteration:

z2 = z1 z1 = x

Repeating these steps for each input sample, obtained the filtered output of the Band-Pass Butterworth IIR digital filter.

**General Difference Equation**

The given code implements a Band-Pass Butterworth IIR digital filter using second-order sections (biquads). Each biquad represents a second-order IIR filter section. Let's break down the mathematical expression for each biquad.

The general difference equation for a second-order IIR filter is:

*y*[*n*]=*b*0​*x*[*n*]+*b*1​*x*[*n*−1]+*b*2​*x*[*n*−2]−*a*1​*y*[*n*−1]−*a*2​*y*[*n*−2]

Where:

* *y*[*n*] is the output at time *n*
* *x*[*n*] is the input at time *n*
* *x*[*n*−1] and *x*[*n*−2] are the previous input samples
* *y*[*n*−1] and *y*[*n*−2] are the previous output samples
* *b*0​,*b*1​,*b*2​ are the feedforward (numerator) coefficients
* *a*1​,*a*2​ are the feedback (denominator) coefficients

Let's express each biquad in the code as difference equations:

**First Biquad:**

z1\_1[n] = x[n] - 0.05159732 \* z1\_1[n-1] - 0.36347401 \* z2\_1[n-1] z2\_1[n] = z1\_1[n-1] y\_1[n] = 0.01856301 \* z1\_1[n] + 0.03712602 \* z2\_1[n] + 0.01856301 \* z2\_1[n-1]

**Second Biquad:**

z1\_2[n] = y\_1[n] - (-0.53945795 \* z1\_2[n-1] - 0.39764934 \* z2\_2[n-1]) z2\_2[n] = z1\_2[n-1] y\_2[n] = 1.00000000 \* z1\_2[n] - 2.00000000 \* z1\_2[n-1] + 1.00000000 \* z2\_2[n-1]

**Third Biquad:**

z1\_3[n] = y\_2[n] - (0.47319594 \* z1\_3[n-1] - 0.70744137 \* z2\_3[n-1]) z2\_3[n] = z1\_3[n-1] y\_3[n] = 1.00000000 \* z1\_3[n] + 2.00000000 \* z1\_3[n-1] + 1.00000000 \* z2\_3[n-1]

**Fourth Biquad:**

z1\_4[n] = y\_3[n] - (-1.00211112 \* z1\_4[n-1] - 0.74520226 \* z2\_4[n-1]) z2\_4[n] = z1\_4[n-1] y\_4[n] = 1.00000000 \* z1\_4[n] - 2.00000000 \* z1\_4[n-1] + 1.00000000 \* z2\_4[n-1]

Where:

* *z*1*i*​[*n*] and *z*2*i*​[*n*] are the state variables for the *i*-th biquad at time *n*
* *x*[*n*] is the input at time *n*
* *yi*​[*n*] is the output of the *i*-th biquad at time *n*

This set of equations describes the mathematical expression for the given Band-Pass Butterworth IIR digital filter implemented in the provided code. Each biquad contributes to the overall filter response, and the output of one biquad serves as the input to the next biquad in the chain.

**Simulation of BPF in MATLAB**

% Generate the signal composed of sinusoids at specified frequencies

fs = 500.0; % Sampling rate in Hz

t = linspace(0, 2, fs); % 2 seconds

frequencies = [45, 60, 100, 160]; % Frequencies in Hz

sig = sum(sin(2\*pi\*frequencies.'\*t), 1);

% Plot the original signal

figure;

plot(t, sig);

title('Raw EMG Signal with Multiple Frequencies');

xlabel('Time [seconds]');

ylabel('Amplitude');

grid on;

% Design the bandpass filter

[b, a] = butter(4, [74.5, 149.5] / (fs / 2), 'bandpass');

[sos, g] = tf2sos(b, a);

% Apply the bandpass filter

filtered\_sig = sosfilt(sos, sig);

% Plot the frequency response of the filter

figure;

freqz(sos, 1024, fs);

title('Bandpass Filter Frequency Response');

xlabel('Frequency (Hz)');

ylabel('Gain (dB)');

grid on;

% Add -3 dB cutoff point

cutoff\_freq = [74.5, 149.5];

cutoff\_response = -3 \* ones(size(cutoff\_freq));

hold on;

plot(cutoff\_freq, cutoff\_response, 'ro');

text(cutoff\_freq(1), -3, '(-3 dB)', 'VerticalAlignment', 'bottom');

% Add vertical lines penetrating the x-axis

plot([cutoff\_freq(1), cutoff\_freq(1)], ylim, 'k--');

plot([cutoff\_freq(2), cutoff\_freq(2)], ylim, 'k--');

hold off;

% Plot the signal after applying the bandpass filter

figure;

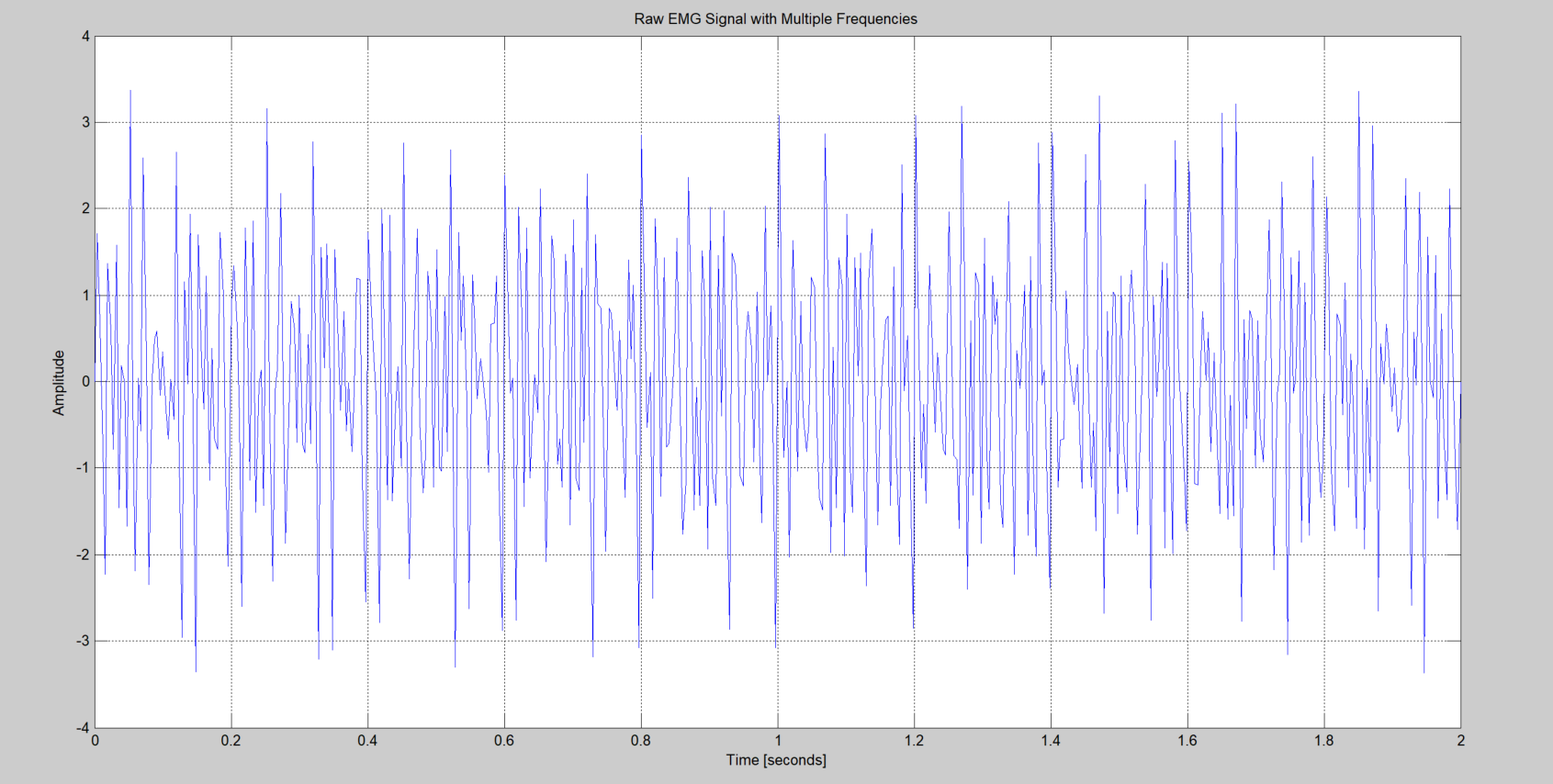
plot(t, filtered\_sig);

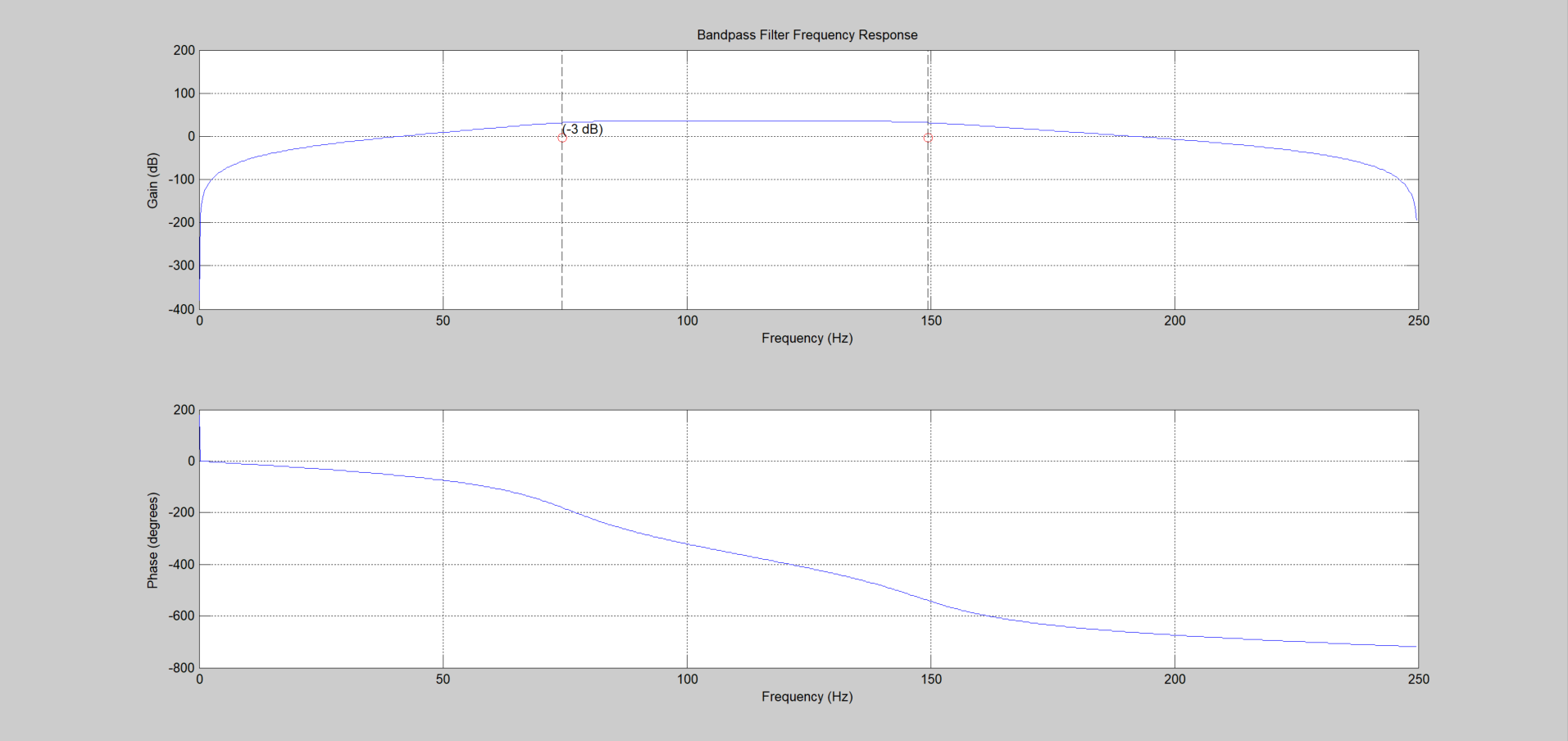
title('EMG Signal After Bandpass Filtering');

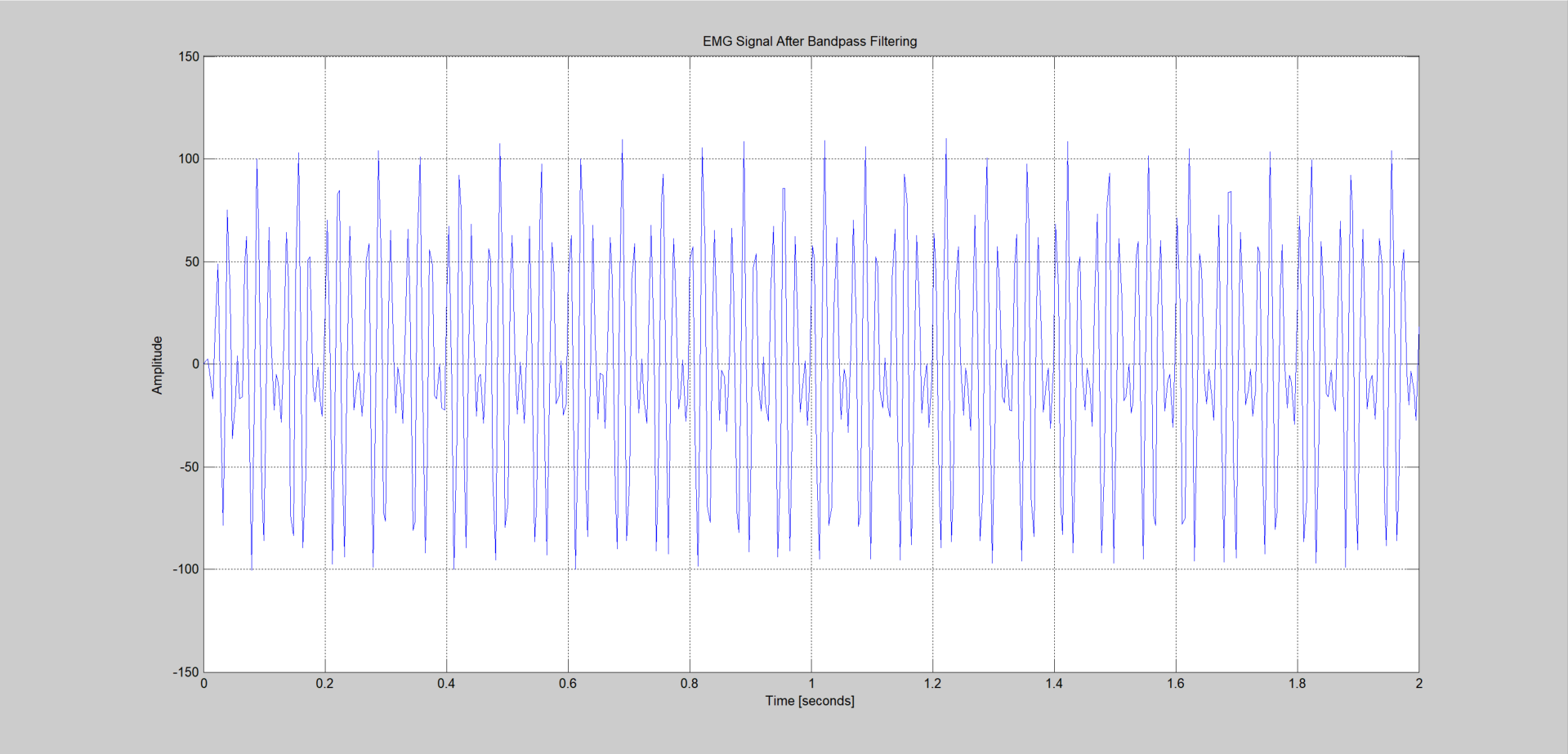
xlabel('Time [seconds]');

ylabel('Amplitude');

grid on;







**Reference:**

<https://docs.scipy.org/doc/scipy/reference/generated/scipy.signal.butter.html>

<https://courses.ideate.cmu.edu/16-223/f2020/Arduino/FilterDemos/filter_gen.py>