EMG Signal Processing

Introduction:

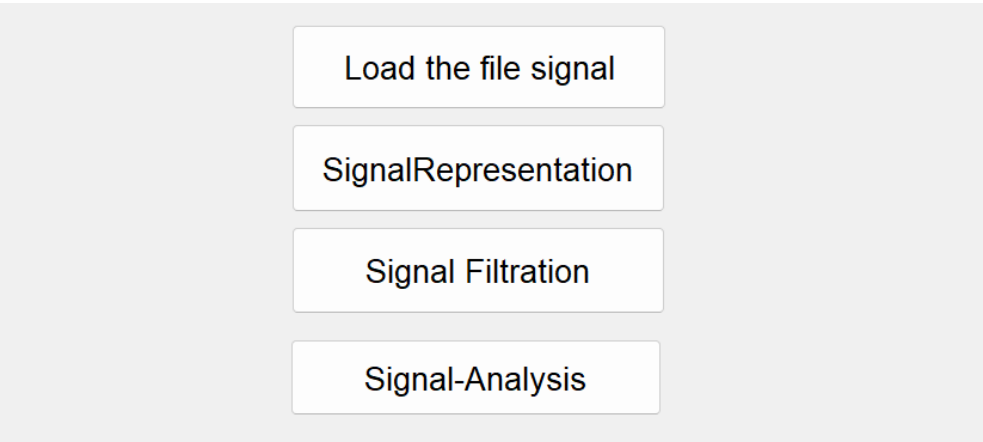
This project presents a graphical user interface (GUI) for EMG signal processing and analysis. The GUI allows users to input EMG signals and visualize them in both time and frequency domains. It includes a spectrogram feature for a detailed frequency analysis. Statistical measures such as mean, median, and variance are computed to provide quantitative insights. Additionally, the GUI incorporates filters to remove unwanted frequencies and enhance the EMG signal. This project offers a user-friendly tool for comprehensive EMG signal analysis, supporting applications in clinical diagnostics and rehabilitation engineering.

Project functions:

1. Representation (time domain, frequency domain, spectrogram)
2. Filtration (butter filter, Chebyshev Filter, fir Filter, notch Filter)
3. Analysis (mean, median, variation, power spectrum)

GUI Explained:

1- loading the signal:
2- Signal representation:
3- Signal filtration:
4- Signal analysis:



- Pseudocode:

```
FUNCTION Project(varargin) RETURNS varargout
    SET gui_Singleton = 1
    SET gui_State = STRUCT WITH FIELDS:
        gui_Name = mfilename
        gui_Singleton = gui_Singleton
        gui_OpeningFcn = Project_OpeningFcn
        gui_OutputFcn = Project_OutputFcn
        gui_LayoutFcn = []
        gui_Callback = []
    IF nargin IS NOT 0 AND FIRST ARGUMENT OF varargin IS A STRING
        SET gui_State.gui_Callback = FUNCTION HANDLE FOR FIRST
ARGUMENT OF varargin
    END IF

    IF nargout IS NOT 0
        SET varargout FROM 1 TO nargout TO OUTPUTS OF gui_mainfcn WITH
INPUTS gui_State AND varargin
    ELSE
        CALL gui_mainfcn WITH INPUTS gui_State AND varargin
    END IF

FUNCTION Project_OpeningFcn(hObject, eventdata, handles, varargin)
    SET handles.output TO hObject
    CALL guidata WITH INPUTS hObject AND handles

FUNCTION Project_OutputFcn(hObject, eventdata, handles) RETURNS
varargout
    SET FIRST ELEMENT OF varargout TO handles.output

FUNCTION pushbutton1_Callback(hObject, eventdata, handles)
    DECLARE firstVariable AS GLOBAL
    SET filename TO SELECTED FILE WITH EXTENSION ".mat"
    IF filename IS NOT 0
        LOAD data FROM filename
        SET fieldNames TO FIELD NAMES OF data
        SET firstVariable TO FIRST FIELD OF data
        SET firstVariable TO FIRST 1100 ELEMENTS OF firstVariable
    END IF

FUNCTION pushbutton2_Callback(hObject, eventdata, handles)
```

```
    DECLARE firstVariable AS GLOBAL
    DECLARE forierTransform AS GLOBAL
    CREATE NEW FIGURE
    SET forierTransform TO signalRepresentation(firstVariable)
    IF RUNNING ON PC AND hObject BACKGROUND COLOR IS DEFAULT
        SET hObject BACKGROUND COLOR TO WHITE
    END IF

FUNCTION pushbutton4_Callback(hObject, eventdata, handles)
    DECLARE firstVariable AS GLOBAL
    DECLARE FILTERED1f AS GLOBAL
    DECLARE FILTERED2f AS GLOBAL
    DECLARE FILTERED3f AS GLOBAL
    DECLARE FILTERED4f AS GLOBAL
    CREATE NEW FIGURE WITH NAME "firstfilteration"
    SET FILTERED1f TO butterfilter(firstVariable)
    SET FILTERED2f TO chebyshevFilter(FILTERED1f)
    SET FILTERED3f TO firFilter(FILTERED2f)
    SET FILTERED4f TO notchFilter(FILTERED3f)

FUNCTION pushbutton5_Callback(hObject, eventdata, handles)
    DECLARE firstVariable AS GLOBAL
    DECLARE FILTERED2f AS GLOBAL
    DECLARE mean1f AS GLOBAL
    DECLARE med1f AS GLOBAL
    DECLARE var1f AS GLOBAL
    DECLARE mean2f AS GLOBAL
    DECLARE med2f AS GLOBAL
    DECLARE var2f AS GLOBAL

    CREATE NEW FIGURE WITH NAME "first"
    SET [mean1f, med1f, var1f, mean2f, med2f, var2f] TO
aianalysisi(firstVariable,FILTERED2f)
END FUNCTION
```

1- loading the signal:

This feature allows users to import pre-recorded EMG data from external sources, such as files or databases.

- Pseudocode:

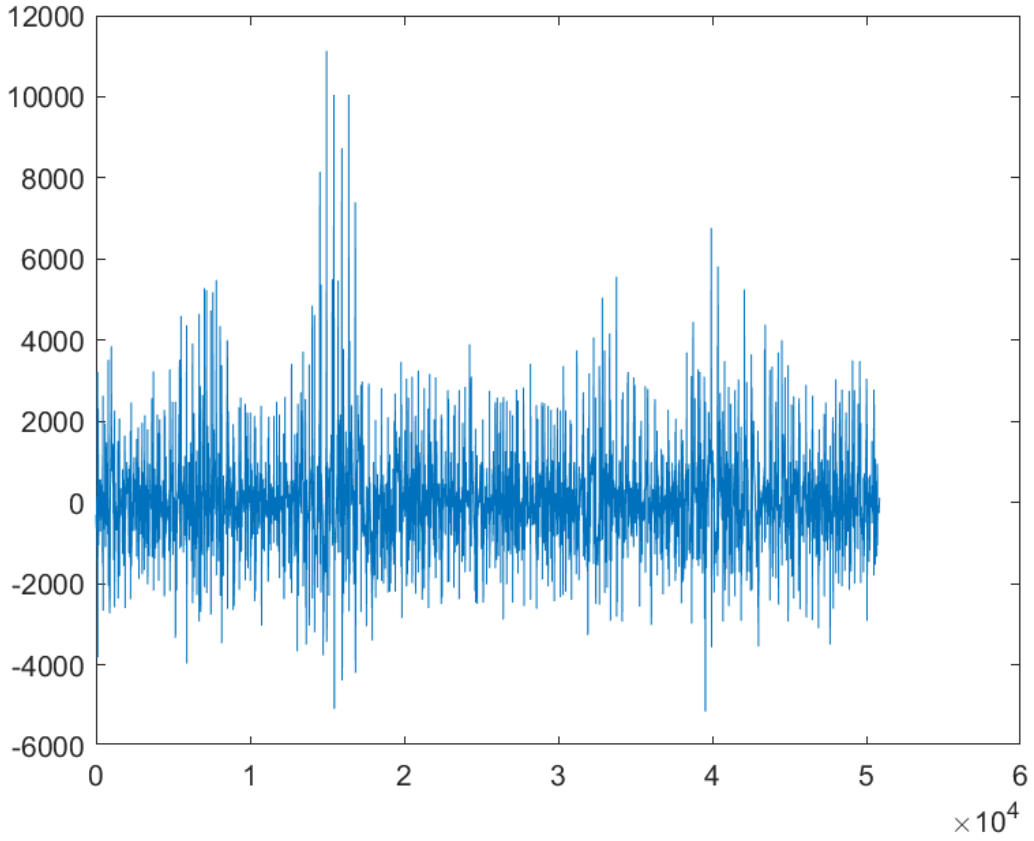
```
FUNCTION pushbutton1_Callback(hObject, eventdata, handles)
    DECLARE firstVariable AS GLOBAL
    SET filename = SELECT FILE WITH EXTENSION ".mat"
    IF filename IS NOT 0
        LOAD data FROM filename
        SET fieldNames = GET FIELD NAMES OF data
        SET firstVariable = GET FIRST FIELD OF data
        SET firstVariable = GET FIRST 1100 ELEMENTS OF firstVariable
    END IF
END FUNCTION
```

- EMG Different signals:

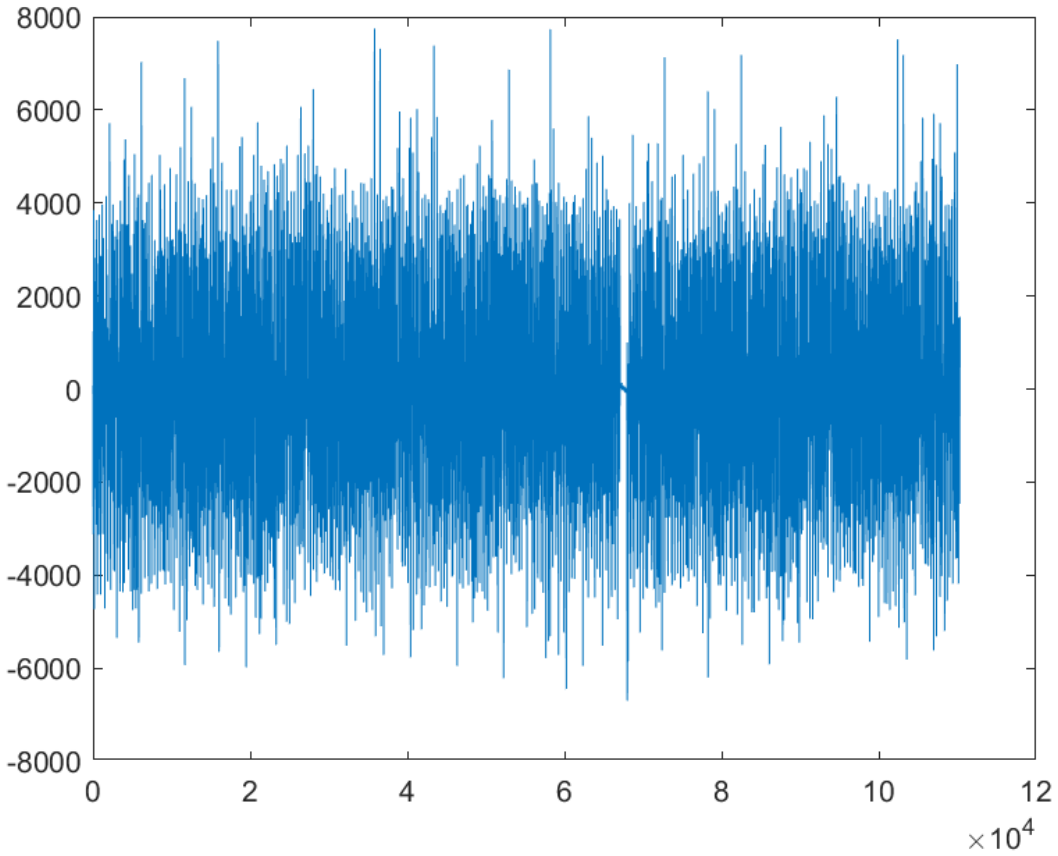
  1- emg_healthy
  2- emg_myopathy
  3- emg_neuropathy

- Output:

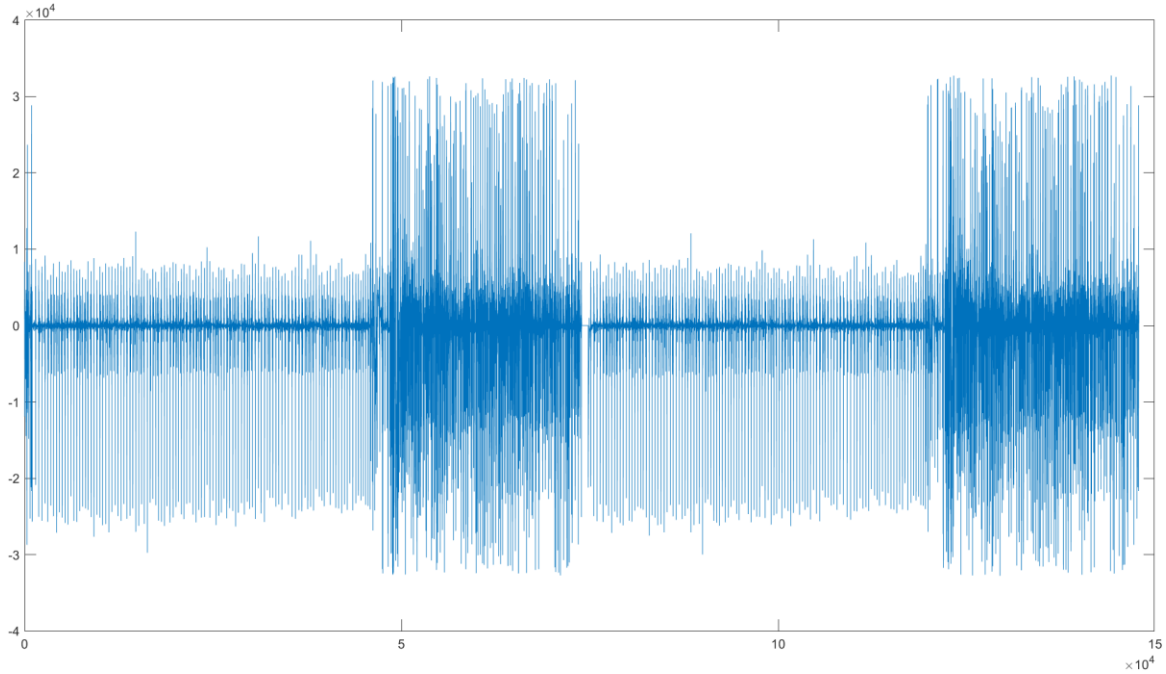
1-healthy:



5- myopathy:



6- neuropathy:



2- Signal representation:

The GUI visually presents the EMG signal in the time and frequency domains. It displays the waveform to observe muscle activity over time and generates a spectrogram for detailed frequency analysis.

Steps:

- Step 1: Display time domain and plot
- Step 2: Fourier transform
- Step 3: Display frequency domain and plot
- Step 4: Spectrogram

- Pseudocode:

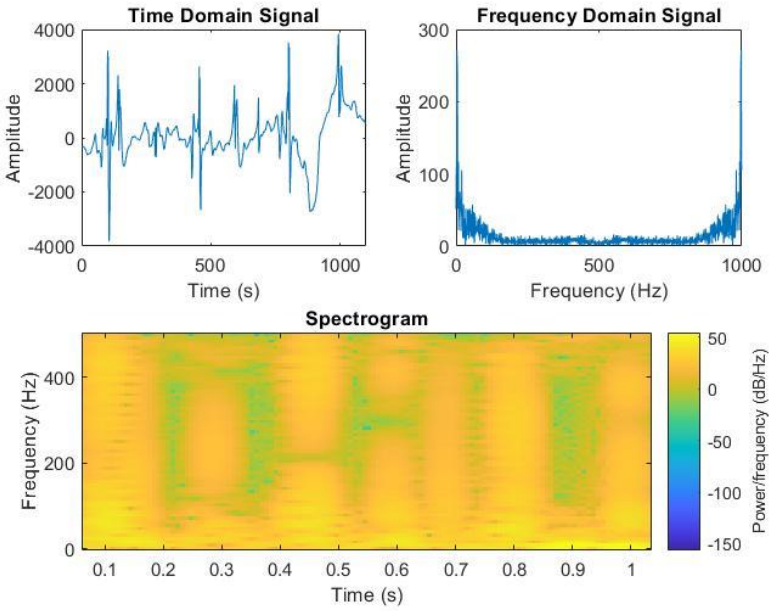
```
FUNCTION pushbutton2_Callback(hObject, eventdata, handles)
    DECLARE firstVariable AS GLOBAL
    DECLARE forierTransform AS GLOBAL
    CREATE NEW FIGURE
    SET forierTransform = signalRepresentation(firstVariable)
    IF RUNNING ON PC AND hObject BACKGROUND COLOR IS DEFAULT
        SET hObject BACKGROUND COLOR TO WHITE
    END IF
END FUNCTION
```

```
FUNCTION signalRepresentation(OriginalSignal) RETURNS fourierTransform
    % Function to perform signal representation
    %% Step 1: Display time domain and plot
    CREATE SUBPLOT (2,2,1)
    PLOT OriginalSignal
    SET X LABEL TO "Time (s)"
    SET Y LABEL TO "Amplitude"
    SET TITLE TO "Time Domain Signal"

    %% Step 2: Fourier transform
    SET fourierTransform = FFT OF OriginalSignal

    %% Step 3: Display frequency domain and plot
    SET samplingFrequency = 1000
    SET frequencies = LINEAR SPACE FROM 0 TO samplingFrequency WITH
LENGTH OF OriginalSignal
```

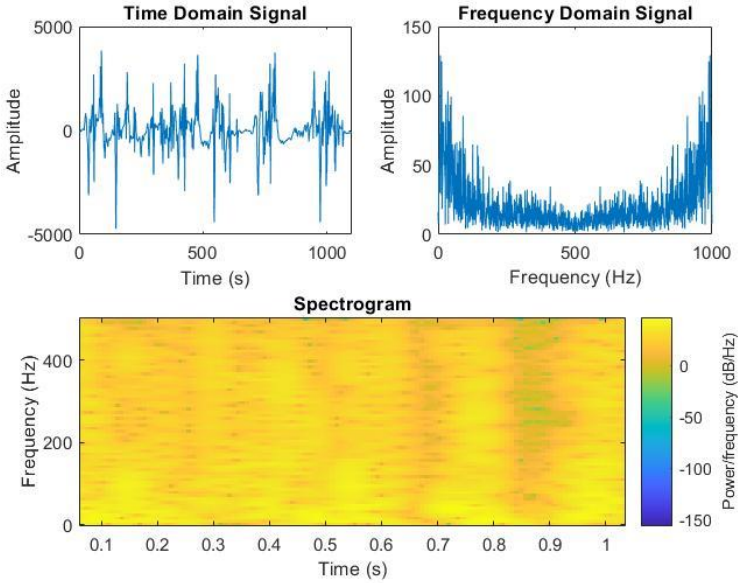
```
    SET amplitudes = ABSOLUTE VALUE OF fourierTransform DIVIDED BY
LENGTH OF OriginalSignal
    CREATE SUBPLOT (2,2,2)
    PLOT frequencies AND amplitudes
    SET X LABEL TO "Frequency (Hz)"
    SET Y LABEL TO "Amplitude"
    SET TITLE TO "Frequency Domain Signal"

    %% Step 4: Spectrogram
    CREATE SUBPLOT (2,2,[3,4])
    CREATE SPECTROGRAM OF OriginalSignal WITH WINDOW SIZE 128, OVERLAP
120, FFT LENGTH 128 AND SAMPLING FREQUENCY samplingFrequency ON Y AXIS
    SET TITLE TO "Spectrogram"
END FUNCTION
```

- Output:

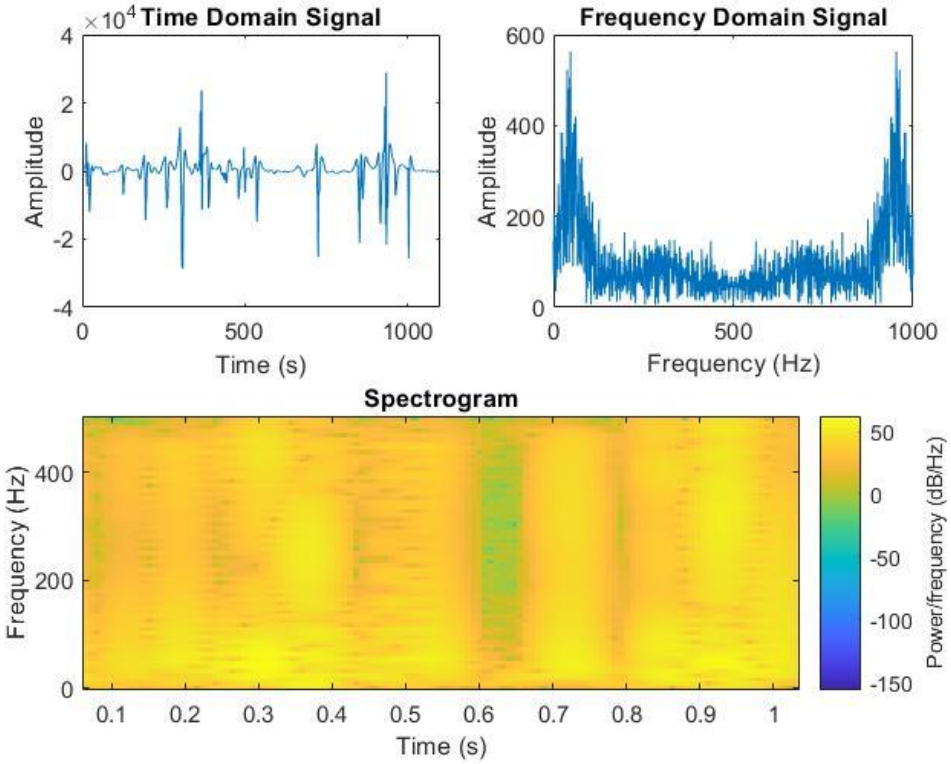
1-healthy:



2- myopathy:



3-neuropathy:



3-Signal filtration:

these operations include applying filters such as Butterworth, Chebyshev, FIR, and a notch filter to the input signal. The resulting filtered signals are stored in separate global variables.

Steps:

- Step 1: Define the parameters for the four filters, namely Butterworth, Chebyshev, FIR, and notch filters, specifying their characteristics and configurations.
- Step 2: Design the filters based on the defined parameters, ensuring they meet the desired frequency response and filtering requirements.
- Step 3: Apply the designed filters to the input signal, processing the signal through each filter consecutively to obtain the filtered output.
- Step 4: Plot the original signal, visually representing its waveform to observe its characteristics and behavior.
- Step 5: Plot the filtered signal, visualizing the impact of the applied filters on the waveform, enabling a comparison between the original and processed signals.

- Pseudocode:

```
FUNCTION pushbutton4_Callback(hObject, eventdata, handles)
    DECLARE firstVariable AS GLOBAL
    DECLARE FILTERED1f AS GLOBAL
    DECLARE FILTERED2f AS GLOBAL
    DECLARE FILTERED3f AS GLOBAL
    DECLARE FILTERED4f AS GLOBAL
    CREATE NEW FIGURE WITH NAME "firstfilteration"
    SET FILTERED1f = butterfilter(firstVariable)
    SET FILTERED2f = chebyshevFilter(FILTERED1f)
    SET FILTERED3f = firFilter(FILTERED2f)
    SET FILTERED4f = notchFilter(FILTERED3f)
END FUNCTION
```

```
FUNCTION butterfilter(emg_signal) RETURNS y
    % Define the filter parameters
    SET fs = 1000 % Sampling frequency
    SET fc = 50 % Cutoff frequency
    SET n = 4 % Filter order

    % Design the filter
    SET [b,a] = butter(n,fc/(fs/2))
```

```
    % Filter the signal
    SET y = filtfilt(b,a,emg_signal)

    % Plot the original and filtered signals
    CREATE SUBPLOT (5,1,1)
    PLOT emg_signal
    SET TITLE TO "Original EMG Signal"
    SET X LABEL TO "Time (s)"
    SET Y LABEL TO "Amplitude"

    CREATE SUBPLOT (5,1,2)
    PLOT y
    SET TITLE TO "Filtered butter EMG Signal"
    SET X LABEL TO "Time (s)"
    SET Y LABEL TO "Amplitude"
END FUNCTION
```

```
FUNCTION chebyshevFilter(emg_signal) RETURNS y
    % Define the filter parameters
    SET fs = 1000 % Sampling frequency
    SET fc_cheby = 100 % Cutoff frequency for Chebyshev filter
    SET n_cheby = 5 % Filter order for Chebyshev filter
    SET ripple = 1 % Passband ripple (in dB) for Chebyshev filter

    % Design the Chebyshev filter
    SET [b_cheby, a_cheby] = cheby1(n_cheby, ripple, fc_cheby/(fs/2))

    % Apply the Chebyshev filter
    SET y = filtfilt(b_cheby, a_cheby, emg_signal)

    % Plotting the original and filtered signals
    CREATE SUBPLOT (5,1,3)
    PLOT y
    SET TITLE TO "Filtered Chebyshev EMG Signal"
    SET X LABEL TO "Time (s)"
    SET Y LABEL TO "Amplitude"
END FUNCTION
Chebyshev EMG Signal"
    SET X LABEL TO "Time (s)"
    SET Y LABEL TO "Amplitude"
```

```
END FUNCTION
```

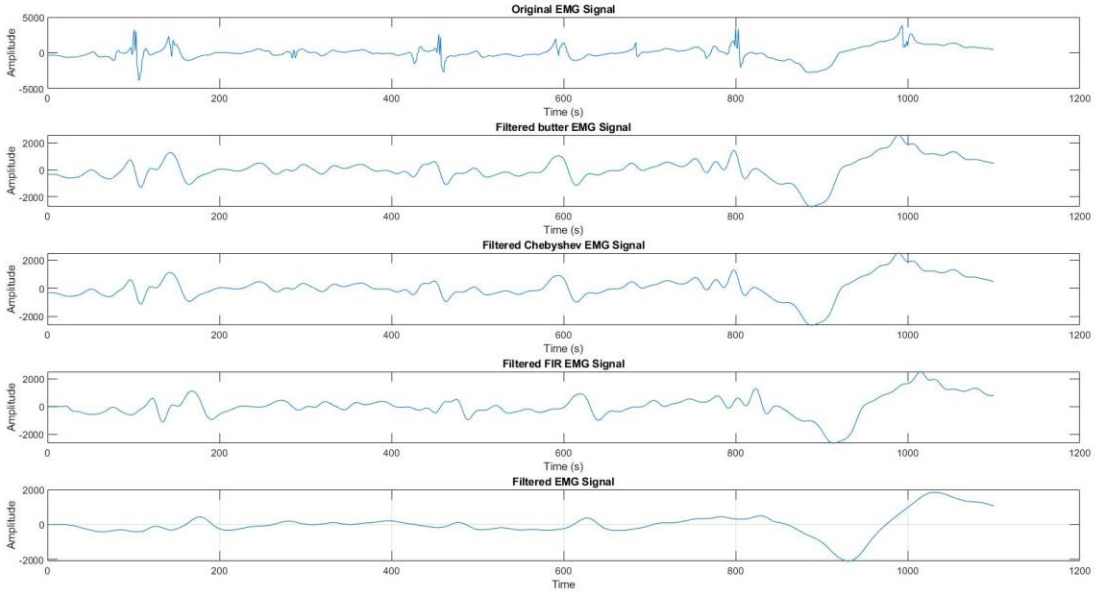
```
FUNCTION firFilter(emg_signal) RETURNS y
    % Define the filter parameters
    SET fs = 1000 % Sampling frequency
    SET fc_fir = 100 % Cutoff frequency for FIR filter
    SET filter_length = 51 % Filter length for the FIR filter

    % Design the FIR filter
    SET b_fir = fir1(filter_length, fc_fir/(fs/2))

    % Apply the FIR filter
    SET y = filter(b_fir, 1, emg_signal)

    % Plotting the original and filtered signals
    CREATE SUBPLOT (5,1,4)
    PLOT y
    SET TITLE TO "Filtered FIR EMG Signal"
    SET X LABEL TO "Time (s)"
    SET Y LABEL TO "Amplitude"
END FUNCTION
```

```
FUNCTION notchFilter(emgSignal) RETURNS filtered
    % Define filter parameters
    SET Fs = 1000 % Sampling frequency
    SET f0 = 50 % Notch frequency
    SET bw = 200 / Fs % Bandwidth of the notch, normalized

    % Design the notch filter
    SET wo = f0 / (Fs/2) % Normalize the frequency
    SET Q = wo / bw % Quality factor
    SET [b, a] = iirnotch(wo, Q) % Design an IIR notch filter

    % Apply the notch filter to the EMG signal
    SET filtered = FILTER emgSignal WITH b AND a

    % Plot the original and filtered signals
    CREATE SUBPLOT (5,1,5)
    PLOT filtered
```

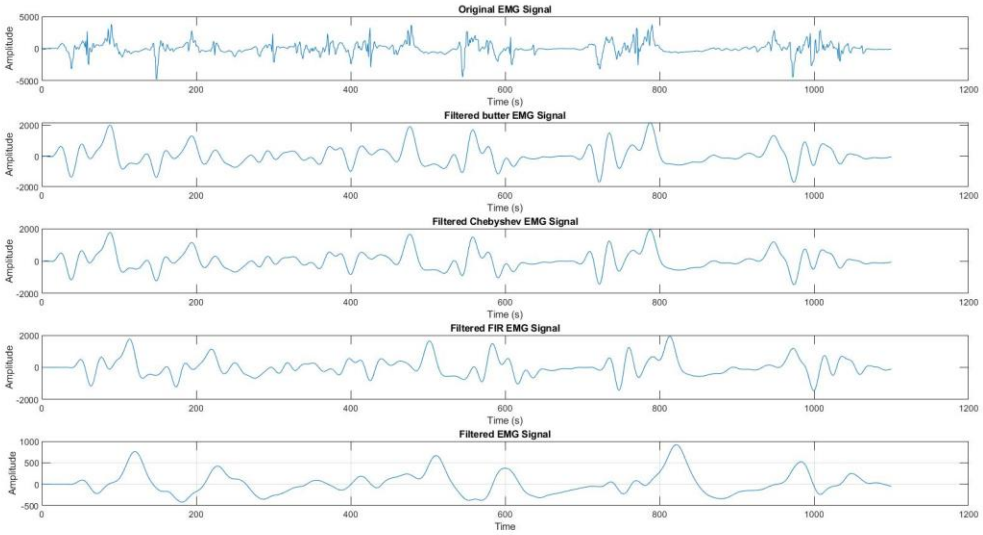
```
    SET X LABEL TO "Time"
    SET Y LABEL TO "Amplitude"
    SET TITLE TO "Filtered EMG Signal"
    TURN ON GRID

    % Display the filtered signal
    DISPLAY "Filtered notch EMG Signal:"
    DISPLAY filtered
END FUNCTION
```

Outputs:

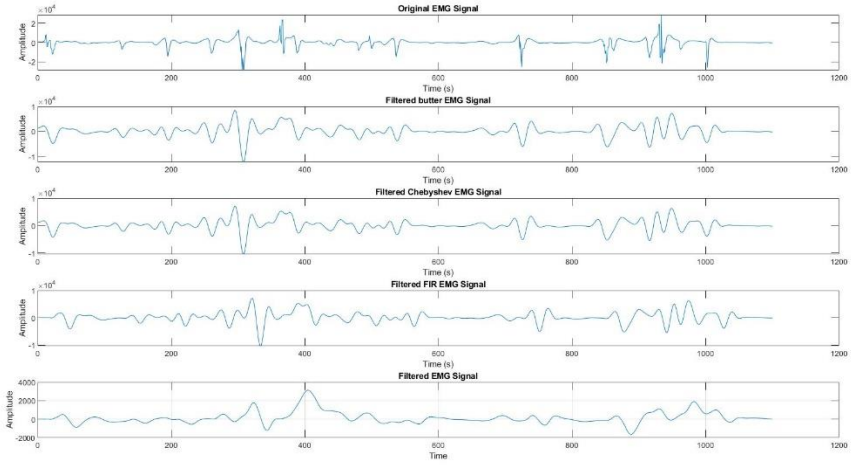
1- Healthy:



2- Myopathy:



3- Neuropathy:



4. <u>Signal analysis:</u>

It computes various characteristics of EMG signals, including mean, median, and variance, and displays the power spectrum. The analysis provides insights into the original and filtered signals, aiding in understanding their characteristics and identifying abnormalities.

Step 1: Compute and plot the mean value for the original signal.

Step 2: Compute and plot the median value for the original signal.

Step 3: Compute and plot the variance for the original signal.

Step 4: Compute and plot the power spectrum of the original signal using the Welch's method.

Step 5: Detect and display abnormalities in the original signal based on a defined threshold.

Step 6: Compute and plot the power spectrum of the filtered signal using the Welch's method.

Step 7: Detect and display abnormalities in the filtered signal based on a defined threshold.

Step 8: Compute and plot the mean, median, and modified variance for the original signal.

Step 9: Compute and plot the mean, median, and variance for the filtered signal.

- Pseudocode:

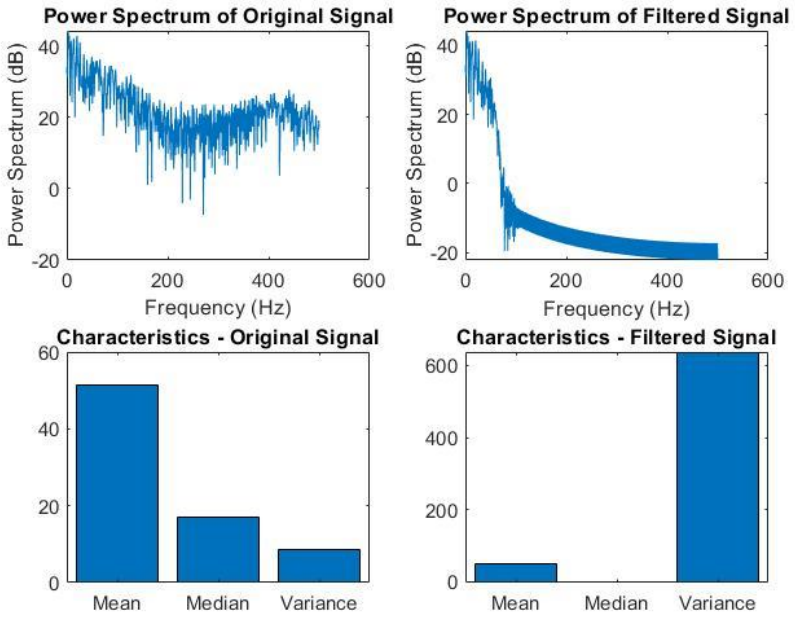
```
FUNCTION aianalysisi(originalSignal, filtred) RETURNS [mean1, med1,
var1, mean2, med2, var2]
    % Step 1: Compute and plot mean for original signal
    SET mean1 = MEAN OF originalSignal

    % Step 2: Compute and plot median for original signal
    SET med1 = MEDIAN OF originalSignal

    % Step 3: Compute and plot variance for original signal
    SET var1 = VARIANCE OF originalSignal

    % Step 4: Compute and plot spectral characteristics for original
signal
    SET fs = 1000 % Sampling frequency (modify according to your
signal)
    SET window_size = 1024 % Window size for FFT
    SET overlap = 0.5 % Overlap between consecutive windows (50%)
    SET noverlap = window_size * overlap % Number of samples
overlapped
    SET nfft = 2 * window_size % Number of FFT points
```

```
    SET [Pxx, f] = pwelch(originalSignal, window_size, noverlap, nfft,
fs)

    % Step 5: Detect abnormalities in original signal
    SET abnormal_threshold = 10 % Define the abnormality threshold in
dB

    % Find indices where power exceeds the threshold
    SET abnormal_indices = FIND WHERE 10 * log10(Pxx) >
abnormal_threshold

    IF abnormal_indices IS NOT EMPTY
        DISPLAY "Abnormalities detected in original signal at
frequencies:"
        DISPLAY f(abnormal_indices)
    ELSE
        DISPLAY "No abnormalities detected in original signal."
    END IF

    % Plot the power spectrum of original signal
    CREATE SUBPLOT (2, 2, 1)
    PLOT f AND 10 * log10(Pxx)
    SET X LABEL TO "Frequency (Hz)"
    SET Y LABEL TO "Power Spectrum (dB)"
    SET TITLE TO "Power Spectrum of Original Signal"

    % Compute and plot spectral characteristics for filtered signal
    SET [Pxx_filt, f_filt] = pwelch(filtred, window_size, noverlap,
nfft, fs)

    % Detect abnormalities in filtered signal
    SET abnormal_indices_filt = FIND WHERE 10 * log10(Pxx_filt) >
abnormal_threshold

    IF abnormal_indices_filt IS NOT EMPTY
        DISPLAY "Abnormalities detected in filtered signal at
frequencies:"
        DISPLAY f_filt(abnormal_indices_filt)
    ELSE
        DISPLAY "No abnormalities detected in filtered signal."
    END IF
```

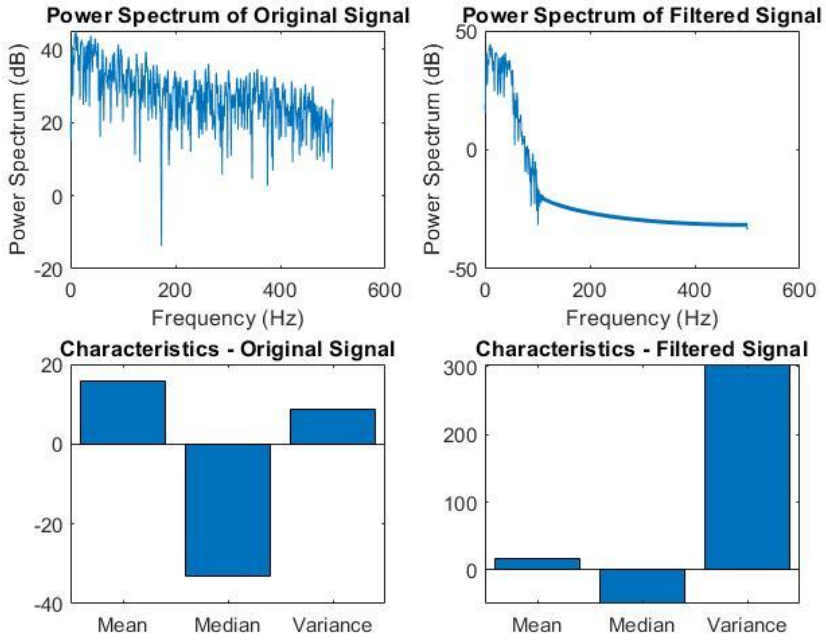
```
    % Plot the power spectrum of filtered signal
    CREATE SUBPLOT (2, 2, 2)
    PLOT f_filt AND 10 * log10(Pxx_filt)
    SET X LABEL TO "Frequency (Hz)"
    SET Y LABEL TO "Power Spectrum (dB)"
    SET TITLE TO "Power Spectrum of Filtered Signal"

    % Compute and plot mean, median, and modified variance for
original signal
    SET var2 = var1 * power(10, -5)
    SET X = CATEGORICAL ARRAY ["Mean", "Median", "Variance"]
    REORDER CATEGORIES OF X TO ["Mean", "Median", "Variance"]
    SET Y = ARRAY [mean1, med1, var2]

    CREATE SUBPLOT (2, 2, 3)
    CREATE BAR CHART WITH X AND Y
    SET TITLE TO "Characteristics - Original Signal"

    % Compute and plot mean, median, and variance for filtered signal
    SET mean2 = MEAN OF filtred
    SET med2 = MEDIAN OF filtred
    SET var2 = VARIANCE OF filtred
    SET var2 = var2 * power(10, -3)
    SET X_filt = CATEGORICAL ARRAY ["Mean", "Median", "Variance"]
    REORDER CATEGORIES OF X_filt TO ["Mean", "Median", "Variance"]
    SET Y_filt = ARRAY [mean2, med2, var2]

    CREATE SUBPLOT (2, 2, 4)
    CREATE BAR CHART WITH X_filt AND Y_filt
    SET TITLE TO "Characteristics - Filtered Signal"
END FUNCTION
```

Outputs:

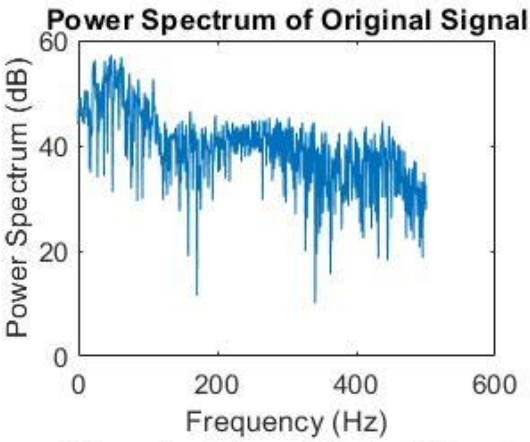
1- Healthy:

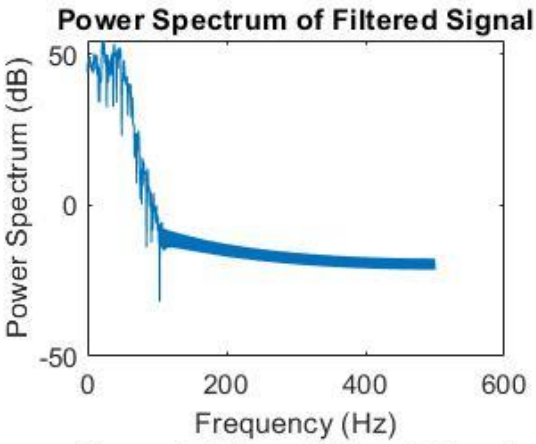


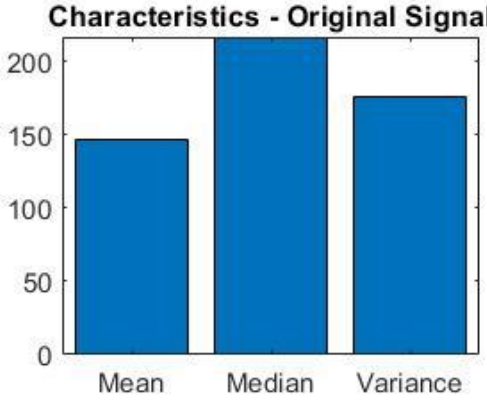
2- Myopathy:

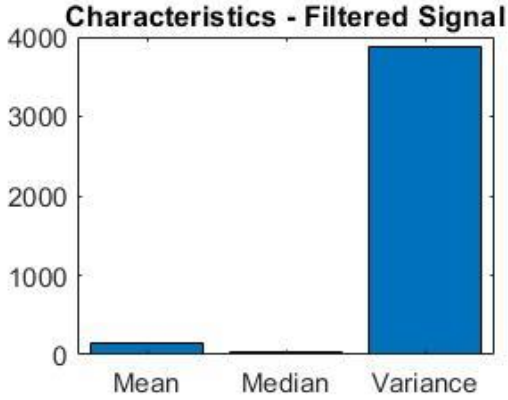


3- Neuropathy:


Power Spectrum of Original Signal
Power Spectrum (dB)
Frequency (Hz)
0
20
40
60
0
200
400
600


Power Spectrum of Filtered Signal
Power Spectrum (dB)
Frequency (Hz)
-50
0
50
0
200
400
600


Characteristics - Original Signal
0
50
100
150
200
Mean
Median
Variance


Characteristics - Filtered Signal
0
1000
2000
3000
4000
Mean
Median
Variance