Electromyography (EMG) Signal Analysis

Sagynbek Talgatuly
ENGR-AD 1000, Fall 2020

STEP 1: Problem Identification and Statement

Problem statement is to develop a software system, which will do the following:

- Operate with the recorded data, apply a low pass filter, crop the endings, normalize, and plot that data on a graph
- Identify segments of data that correspond to one trial of grasping activity
- Calculate the average of the segments for both EMG and Force signals
- Plot a graph, in which the Force signal (y-axis) is set against the EMG signal (x-axis)
- Obtain the best fit function for the graph of the Force signal against the EMG signal

STEP 2: Gathering of Information and Input and Output Description

Background

One important research about hand-arm system is the information about hand grip force and pressing force on a tool handle. One of the most popular solutions for measuring grip force involves attaching force sensors to the handled tool. However, when used with regular hand tool like a drill, it seems to be uncomfortable because the force sensors interfere with the handle construction. An alternative solution would be to evaluate and record the electrical activity produced by muscles, electromyography (EMG). It has been assumed that EMG signal are proportional to muscle tension responsible for palm grip. This solution has a significant advantage when comparing to force sensors, it can be used with regular tool without interfering in the handle.

Electromyography is a technique connected with receiving, recording, and examining of myoelectric signals. These signals are coming into being thanks to physiological changes which are taking place in muscular fibers. Such signals can be registered through two types of electrodes: needle electrodes (inserted straight into the muscle to show the signal of a motor unit) and surface electrode (put against the surface of the skin over an examined muscle to show a total signal of many motor units). During relaxation, a muscle shows no electric relevant activity so that the electric line is straight. During the contraction, potential of motor units deviates the EMG line. This line is a product of frequency and amplitude produced by registered potentials. In this report, the correlation of muscle activity and hand grasp force will be analyzed.

Data Acquisition System

The muscles of the forearm are responsible for movements of the elbow and the hand as well as for the supination of the forearm. They are divided in three groups: anterior (rear), posterior (dorsal), and lateral. The strongest flexor of the hand during grip force is flexor digitorum superficialis, shown in Figure 1. The experimental setup to measure the EMG and grip force data is shown in Figure 2. The EMG electrode is attached to the flexor digitorum superficialis. The electrode is placed around 18 cm from the wrist. A soft tooth-shaped stress ball equipped with a force sensor to measure the grip force is utilized for the grasping task. The user is asked to grasp and press the soft tooth during which the grip force is recorded through the force sensor while the muscle activity is recorded using the EMG sensor.

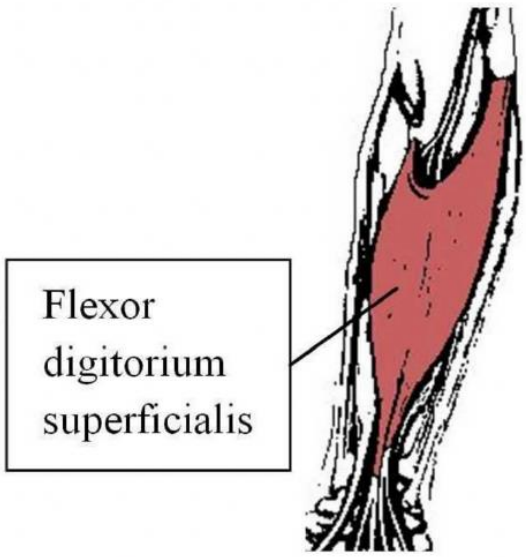


Figure *1* The target muscle

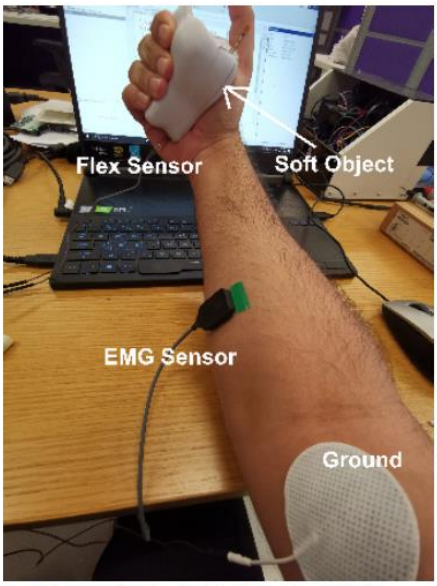


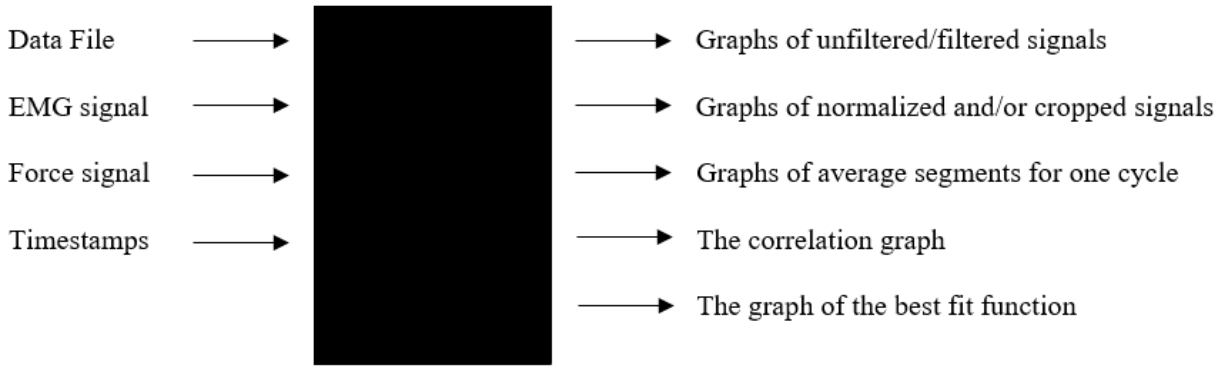
Figure 2 Data Collection Setup

EMG data is collected through the Delsys Bagnoli EMG system along with a flex sensor attached to a soft body. The EMG electrode and force sensor are connected to the data acquisition board which records the EMG and grip force signals (in the form of voltage) and store them into a file. The file stores the time stamp, the grip force, and the EMG muscle activity as three columns of data in the same order. The data is collected at a sampling rate of 1 kHz for a duration of 30 seconds. Here, it would be worth to describe the inputs/outputs of the entire program.

The inputs: Data file, EMG signal, Force signal, Timestamps

The outputs: Graphs of unfiltered/filtered signals, Graphs of normalized and/or cropped signals, Graphs of average segments for one cycle, The correlation graph, The graph of the best fit function

The black-box diagram of input/output of the program is provided below



STEP 3: Test Cases and Algorithm Design

A) Test Cases

1) Low pass filter for EMG and Force signals

The first task of the assignment is to filter the data signals with a lowpass filter. In this part of this step, the lowpass filter will be tested for the EMG signal and the Force signal. The original graphs of EMG and Force signals are expected to be displayed, along with the filtered signal graphs. It is expected that the new filtered graphs will have less noises than the initial ones.

2) Cropping & normalization

In this part of this step, the cropping and normalizing functionalities of the software will be tested. According to the prompt, the initial 500 and last 250 points of the data should be eliminated from the data. Along with this, the Force signal should be normalized corresponding to the Maximum Voluntary Contraction. In the testing, the original graphs of EMG and the Force signals will be compared with the cropped/normalized signal graphs. It is expected that the new graphs will have endings cropped, and the Force signal will be normalized (all the points between 0 and 100). It is expected that this test will be conducted on the filtered signal.

3) Envelope function for the EMG signal

Before the segmentation part starts, the EMG signal is smoothed using the built-in rms envelope function. During the actual run, the envelope function will be applied, but the graph of it will not be plotted. In this step, the code will be adjusted so that we can see the graph of the envelope, which cannot be seen during the actual run. In the submitted program, it will not display the envelope of EMG signal but will use it as it should. It is expected that the envelope of the EMG signal will be smoother, positive, and with less noises.

4) Finding the peaks

Before the segmentation and calculation of the average, the program needs to find the peaks of the EMG and Force signals. It is expected that there will be 31 peaks identified with almost equal intervals between them. It is also expected that the peaks of one of the signals will be close to the corresponding peaks of the other signal.

5) Segmentation & finding the average

Another task of the assignment is to segment the data into 31 equal parts that represent 31 cycles of press-release activity. After that, the average segment should be calculated. In this part of this step, this functionality of the software will be tested. It is expected that the software will display the average signal graphs for both EMG and Force signals. It is also expected that these two graphs will have the same x-axis.

6) Plotting the correlation graph & poly fit function

The last functionality of the software to be tested will be the identification of the best fit function to the correlation graph of average EMG and Force segments. According to the prompt, the software should display the Force against EMG signal graph for one cycle of the press-release activity. Also, the software should use only the contraction period of the cycle for finding the best fit function. It is expected that contraction and release parts of the cycle will be plotted, as well as the best fit function on the same graph.

B) Algorithm (pseudocode)

Clear the workspace
Clear the command window
Read the data file called “Data.mat”
Call function figure, passing 1 as argument to create a figure window
Call function subplot, passing 2,1,1 as arguments
Call function plot, passing timestamps, EMG_signal as arguments
Set the title “EMG signal (original)”
Set the label for x-axis “Time (seconds)”
Set the label for y-axis “EMG (in volts)”
Call function subplot, passing 2,1,2 as arguments
Call function plot, passing timestamps, Force_signal as arguments
Set the title “Force signal (original)”
Set the label for x-axis “Time (seconds)”
Set the label for y-axis “Force”
Assign 1000 to sampling_frequency
Assign 200 to cutoff_frequency
Call function lowpass, passing EMG_signal, cutoff_frequency, sampling frequency as arguments and assign its return to EMG_signal
Call function lowpass, passing Force_signal, cutoff_frequency, sampling frequency as arguments and assign its return to Force_signal
Call function figure, passing 2 as argument to create a figure window
Call function subplot, passing 2,1,1 as arguments
Call function plot, passing timestamps, EMG_signal as arguments
Set the title “EMG signal (filtered)”
Set the label for x-axis “Time (seconds)”
Set the label for y-axis “EMG (in volts)”
Call function subplot, passing 2,1,2 as arguments
Call function plot, passing timestamps, Force_signal as arguments
Set the title “Force signal (filtered)”
Set the label for x-axis “Time (seconds)”
Set the label for y-axis “Force”
Call function length, passing timestamps as argument and assign its return to length_data
Assign 501st to (length_data – 250)th elements of EMG_signal to EMG_signal
Assign 501st to (length_data – 250)th elements of Force_signal to Force_signal
Assign 501st to (length_data – 250)th elements of timestamps to timestamps
Call function length, passing timestamps as argument and assign its return to length_data
Assign first element of Force_signal to global_max_of_f
Assign first element of Force_signal to global_min_of_f
Assign 1 to i
Repeat the following while i is less than or equal to length_data
    If Force_signal(i) is greater than global_max_of_f then
        Assign Force_signal(i) to global_max_of_f
    If Force_signal(i) is less than global_min_of_f then
        Assign Force_signal(i) to global_min_of_f
    Increment i by 1
Assign 1 to j
Repeat the following while j is less than or equal to length_data
    Assign Force_signal(j) – global_min_of_f to top
    Assign global_max_of_f – global_min_of_f to bot
    Assign 100*top/bot to Force_signal(j);
    Increment j by 1
Call function figure, passing 3 as argument to create a figure window
Call function subplot, passing 2,1,1 as arguments
Call function plot, passing timestamps, EMG_signal as arguments
Set the title “EMG signal (cropped)”

Set the label for x-axis “Time (seconds)”
Set the label for y-axis “EMG (in volts)”
Call function subplot, passing 2,1,2 as arguments
Call function plot, passing timestamps, Force_signal as arguments
Set the title “Force signal (cropped and normalized)”
Set the label for x-axis “Time (seconds)”
Set the label for y-axis “Force (in percentage of MVC)”
Assign 150 to window_length
Call function envelope, passing EMG_signal, window_length, ‘rms’ as arguments and assign its return to upper_envelope and bottom_envelope
Assign 500 to segment_length
Declare a matrix called average_segment with 2 rows and segment_length columns
Declare a vector called maxPoints_Force with size of 31
Declare a vector called maxPoints_EMG with size of 31
Assign 1 to i
Assign 1 to k
Repeat the following while k is less than or equal to 31
    Repeat the following until it breaks the loop
        If Force_signal(i) is greater than 25 and Force_signal(i) is less than Force_signal(i + 1)
            Exit the loop
        Increment i by 1
    Assign i + 1 to MAX
    Assign 1 to j
    Repeat the following while Force_signal(i + j) is greater than 25
        If Force_signal (i + j) is greater than Force_signal (MAX) then
            Assign i + j to MAX
        Increment j by 1
    Assign i + j to i
    Assign MAX to maxPoints_Force(k)
    Increment k by 1
Assign 1 to k
Assign 250 to i
Repeat the following while i is less than or equal to length_data – 350
    Assign 1 to flag
    If upper_envelope(i) is less than 0.19 then
        Assign 0 to flag
    If flag is 0 then
        Assign 1 to j
        Repeat the following while j is less than or equal to 249
            If upper_envelope(i) is less than upper_envelope(i+j) or is less than upper_envelop(i-j) then
                Assign 0 to flag
            Increment j by 1
    If flag is 1 then
        Assign i to maxPoints_EMG(k)
        Increment k by 1
    Increment i by 1
Assign 1 to k
Repeat the following while k is less than or equal to 31
    Assign maxPoints_Force(k) – ((segment_length)/2 – 1) to a
    Assign maxPoints_Force(k) + segment_length)/2 to b
    Assign first raw of average_segment element-by-element incremented by ath to bth elements of transposed Force_signal to first raw of average_segment
    Assign maxPoints_EMG(k) – ((segment_length)/2 – 1) to a
    Assign maxPoints_EMG(k) + segment_length)/2 to b
    Assign second raw of average_segment element-by-element incremented by ath to bth elements of transposed upper_envelope to second raw of average_segment

Increment k by 1
Call function figure, passing 4 as argument to create a figure window
Call function subplot, passing 3,1,1 as arguments
Call function plot, passing vector that contains all the integers from 1 to segment_length, second raw of average_segment as arguments
Set the title “Average segment for EMG”
Set the label for x-axis “Time (milliseconds)”
Set the label for y-axis “EMG (in volts)”
Call function subplot, passing 3,1,2 as arguments
Call function plot, passing vector that contains all the integers from 1 to segment_length, first raw of average_segment as arguments
Set the title “Average segment for Force”
Set the label for x-axis “Time (milliseconds)”
Set the label for y-axis “Force (MVC)”
Call function subplot, passing 3,1,3 as arguments
Assign 1st to (segment_length/2)th elements of second raw of average_segment to vec1
Assign 1st to (segment_length/2)th elements of first raw of average_segment to vec2
Assign (segment_length/2)th to (segment_length)th elements of second raw of average_segment to vec3
Assign (segment_length/2)th to (segment_length)th elements of first raw of average_segment to vec4
Call function plot, passing vec1, vec2 as arguments
Hold the current plot
Call function plot, passing vec3, vec4 as arguments
Call function polyfit, passing vec1, vec2, 1 as arguments and assign its return to polynom_f_coefficients
Call function polyval, passing polynom_f_coefficients, vec1 as arguments and assign its return to best_fit
Call function plot, passing vec1, best_fit as arguments
Set the title “Correlation graph”
Set the label for x-axis “EMG (in volts)”
Set the label for y-axis “Force (MVC)”
Set the legend “Contraction”, “Release”, “Best fit line”, “location”, “northwest”
Stop holding the current plot

C) Algorithm Explanation:

The first task is, in general, preparing the data for the further analysis. The data is filtered, cropped, and normalized. For the filtering part, MATLAB built-in function lowpass was used with parameters 200Hz and 1000Hz. These are the cutoff and sampling frequency values respectively which are provided in the prompt. To crop the data, exactly 29250 data points of signals that store the information about the grasping activity were assigned to the exact same signal data. This is how the initial 500 and last 250 points were eliminated. For the normalization part, the normalization formula was used, which is the following:

$$X(normalized) = 100 * \frac{X - globalMin}{globalMax - globalMin}$$

, where globalMin and globalMax refers to the greatest and the lowest data points.

The second task is to automatically identify the segments that represent cycles of grasping activity. In the solution, the segments for EMG and Force signals were identified separately. However, in both cases, the peaks of the segments were found by the program first. In the segmentation of Force signals the following was done. The loop starts from the first data point and goes until the moment when the current point is greater than 25 and is less than the next data point. Number 25 was chosen because it can be apparently seen from the normalized graph that all the peaks are greater than that number. When this moment is captured, the program seeks the peak in the range of data points from the found data point to the first encountered data point which is less than 25. This process repeats 31 times, because there are 31 grasping cycles according to the prompt. When it comes to the EMG signal segmentation, since the EMG signal has significant noises even after applying the filter and it has negative data points, the process for EMG signal was different from that for the Force signal. The RMS (root-mean-square) envelope of the EMG signal was used first to both smooth the graph and get rid of the negative data

points. For this, the MATLAB built-in function envelope was utilized. Similar to the Force signal segmentation, the process goes through data points in the range from 250th all the way to 28900th. These numbers were chosen, again, because from the EMG graph (rms) it can be seen that the first peak occurred after 750ms passed (taking into account that 500ms were cropped), while the last peak occurred before 28900ms passed. To be identified as a peak, the data points have to pass two conditions: they have to be greater than 0.19 (because all the peak are visually greater than 0.19), and they have to be greater than all the data points in the radius of 250ms. So, if the current data point that is being checked is greater than 0.19 and greater than all the 249 preceding and all the 250 following data points, then this point is the peak. After the peaks of the EMG and Force signals were found by the program, the length of the segment was set to 500 points (the entire grasping activity cycle takes less than 500ms). 500 data points centered at the peaks were added and divided into 31 in order to obtain the average segment. This was done for both signals.

For the last tasks, the average segments found earlier were plotted against time (from 1ms to 500ms). Then, the average segment for the Force signal (y-axis) was plotted against the average segment for the EMG signal (x-axis). Lastly, the built-in MATLAB function Polyfit was used to obtain the best fit function for the correlation graph of two average segments. The Polyfit was applied to only one half of the average segments (only contraction part) according to the prompt provided.

STEP 4: Implementation

Assignment5.m

```
% This script analyzes the correlation between the Force and EMG signals
%{
Name: Sagynbek Talgatuly, Student Number: st4121
Date: December 16. 2020
Program: Assignment5.m
Description:
    This program works with two data signals: EMG signal and
    Force signal. It filters them, crops them, normalizes them.
    The program, then, identifies equal segments of the data
    and calculates the average segment. Lastly, it plots the
    relation graph of this two averages and finds the best fit
    function.
%}
%% Task 0
% clearing the workspace and command window
clc;
clear;
%% Task 1

% loading the data file
load('Data.mat');

% plotting the graphs of the original data
figure(1);
subplot (2,1,1);
plot (timestamps, EMG_signal);
title ('EMG signal (original)');
xlabel ('Time (seconds)');
ylabel ('EMG (in volts)');
subplot (2,1,2);
plot (timestamps, Force_signal);
title ('Force signal (original)');
xlabel ('Time (seconds)');
ylabel ('Force');

% applying the low-pass filter with a cutoff frequency 200Hz
% sampling frequency is given in the problem statement
% cutoff frequency is given in the problem statement
sampling_frequency = 1000;
cutoff_frequency = 200;
EMG_signal = lowpass(EMG_signal,cutoff_frequency,sampling_frequency);
Force_signal = lowpass(Force_signal,cutoff_frequency,sampling_frequency);

% plotting the graphs of the filtered data
figure(2);
subplot (2,1,1);
plot (timestamps, EMG_signal);
```

```
title ('EMG signal (filtered)');
xlabel ('Time (seconds)');
ylabel ('EMG (in volts)');
subplot (2,1,2);
plot (timestamps, Force_signal);
title ('Force signal (filtered)');
xlabel ('Time (seconds)');
ylabel ('Force');
clear sampling_frequency;
clear cutoff_frequency;

% cropping the endings of the data
length_data = length(timestamps);

EMG_signal = EMG_signal(501:length_data-250);
Force_signal = Force_signal(501:length_data-250);
timestamps = timestamps(501:length_data-250);

length_data = length(timestamps);

% normalizing the force signal
global_max_of_f = Force_signal(1);
global_min_of_f = Force_signal(1);

% finding the global max and min
for i = 1:length_data
    if Force_signal(i) > global_max_of_f
        global_max_of_f = Force_signal(i);
    end
    if Force_signal(i) < global_min_of_f
        global_min_of_f = Force_signal(i);
    end
end

% applying the normalization formula
for j = 1:length_data
    top = (Force_signal(j) - global_min_of_f);
    bot = (global_max_of_f - global_min_of_f);
    Force_signal(j) = 100 * top / bot;
end

clear top;
clear bot;
clear i;
clear j;
clear global_max_of_f;
clear global_min_of_f;

% plotting the graphs of cropped and normalized data
figure(3);
subplot (2,1,1);
plot (timestamps, EMG_signal);
title ('EMG signal (cropped)');
xlabel ('Time (seconds)');
ylabel ('EMG (in volts)');
subplot (2,1,2);
plot (timestamps, Force_signal);
title ('Force signal (cropped and normalized)');
xlabel ('Time (seconds)');
ylabel ('Force (in percentage of MVC)');

%% Task 2 & 3

% applying the rms envelope function for EMG signal
window_length = 150;
[upper_envelope, bottom_envelope] = envelope((EMG_signal),window_length,'rms');
clear window_length;

% setting the length of one cycle
% creating the matrix for the average segment
segment_length = 500;
average_segment = zeros(2,segment_length);

maxPoints_Force = zeros(31);
maxPoints_EMG = zeros(31);

% finding the peaks of the Force signal
i = 1;
for k = 1:31
    while 1==1
```

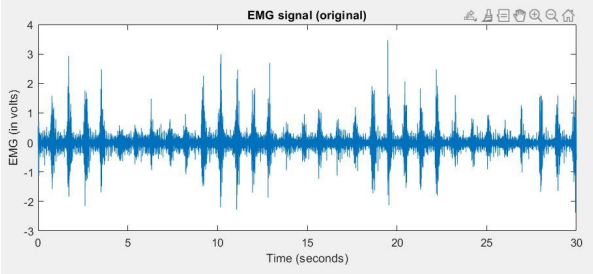
```
        if (Force_signal(i) > 25 && Force_signal(i) < Force_signal(i+1))
            % if some point is greater than 25 and at this point the graph
            % is increasing, then we break the loop at this point
            break;
        end
        i = i + 1;
    end

    MAX = i + 1;

    j = 1;
    % from that point it starts looking for the peak
    % it seeks for the peak until the point is less than 25
    while Force_signal(i+j) > 25
        if Force_signal(i+j) > Force_signal(MAX)
            MAX = i+j;
        end
        j = j + 1;
    end

    i = i + j;
    % the peak is found and assigned to the vector
    % this process goes 31 times
    maxPoints_Force(k) = MAX;
end
clear i;
clear j;
clear MAX;

% finding the peaks of the EMG signal
k = 1;
for i = 250:length_data-350
    % in this loop, all the points from 250 to 28900 are analyzed
    % for this algorith, the program uses a flag variable
    flag = 1;
    % if the current element is less than 0.19 then flag is set to 0
    % because we know that the lowest peak is greater than 0.2 (manually checked)
    if upper_envelope(i)<0.19
        flag = 0;
    end
    % if flag is still 1, the next condition is checked
    % the condition that this point is greater than all the points in a
    % certain scope
    if flag == 1
        for j = 1:249
            if upper_envelope(i)<upper_envelope(i+j) || upper_envelope(i)<upper_envelope(i-j)
                flag = 0;
            end
        end
    end
    % if the point passed the two conditions above
    % then it concludes that this point is indeed the peak
    if flag == 1
        maxPoints_EMG(k) = i;
        k = k + 1;
    end
end

clear flag;
clear i;
clear j;

% calculating the average segment
for k = 1:31
    average_segment(1,:) = average_segment(1,:) + Force_signal(maxPoints_Force(k)-
        (segment_length/2-1):maxPoints_Force(k)+segment_length/2)';
    average_segment(2,:) = average_segment(2,:) + upper_envelope(maxPoints_EMG(k)-
        (segment_length/2-1):maxPoints_EMG(k)+(segment_length/2))';
end
clear k;
clear maxPoints;
average_segment = average_segment / 31;

% plotting the average segment graphs of EMG and Force signals
figure(4);
subplot(3,1,1);
plot(1:segment_length,average_segment(2,:));
title ('Average segment for EMG');
xlabel ('Time (milliseconds)');
ylabel ('EMG (in volts)');
```

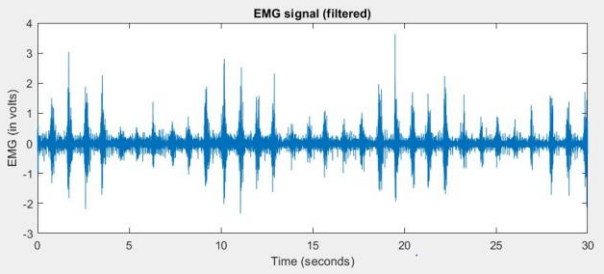
```
subplot(3,1,2);
plot(1:segment_length,average_segment(1,:));
title ('Average segment for Force');
xlabel ('Time (milliseconds)');
ylabel ('Force (MVC)');
subplot(3,1,3);
plot(average_segment(2,(1:segment_length/2)),average_segment(1,(1:segment_length/2)));
hold on;
plot(average_segment(2,(segment_length/2:segment_length)),
      average_segment(1,(segment_length/2:segment_length)));
%% Task 4

% finding the best fit function
polynom_f_coefficients = polyfit(average_segment(2,(1:segment_length/2)),
      average_segment(1,(1:segment_length/2)),1);
best_fit_function = polyval(polynom_f_coefficients,average_segment(2,(1:segment_length/2)));
% plotting the best fit function on the same figure
plot(average_segment(2,(1:segment_length/2)), best_fit_function);
title ('Correlation graph');
xlabel ('EMG (in volts)');
ylabel ('Force (MVC)');
legend({'Contraction', 'Release', 'Best fit line'}, 'location', 'northwest');
hold off;
```

STEP 5: Tests and Verification

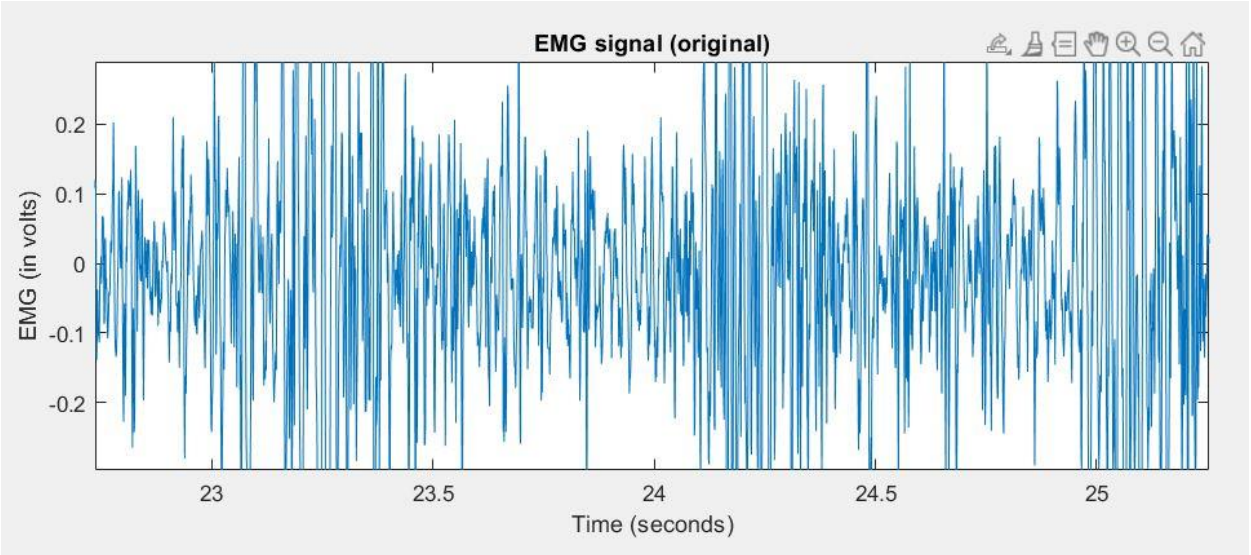
1. For the Lowpass Filter Testing

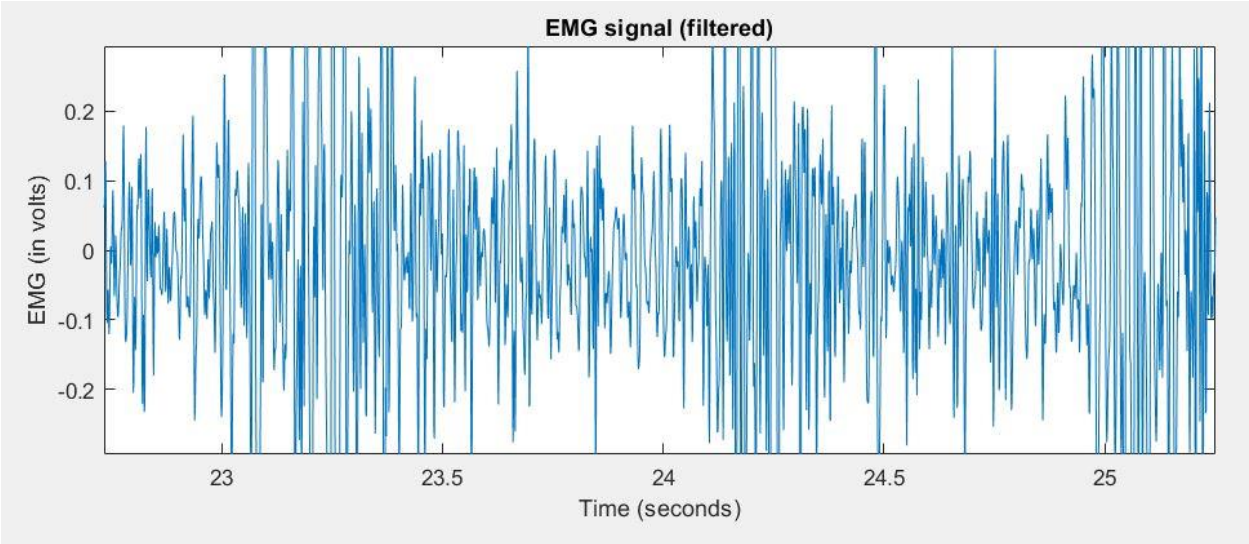
Test case 1: EMG signal





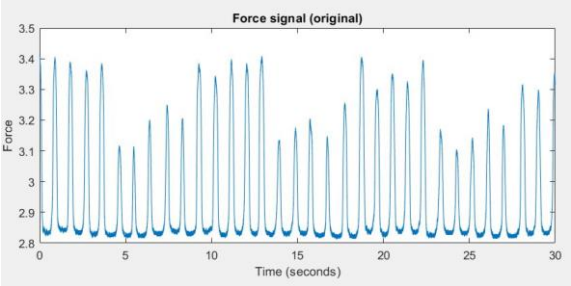
If we look at the graphs in their entirety, the differences are difficult to be noticed. However, if we take a closer look, the differences are more visible.

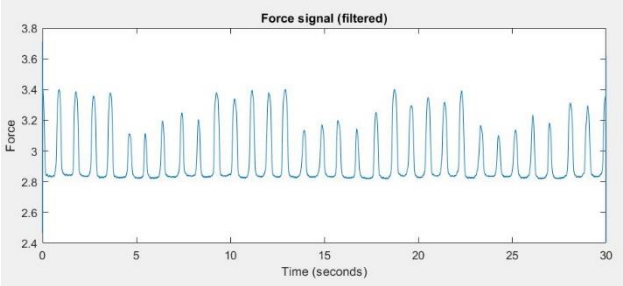


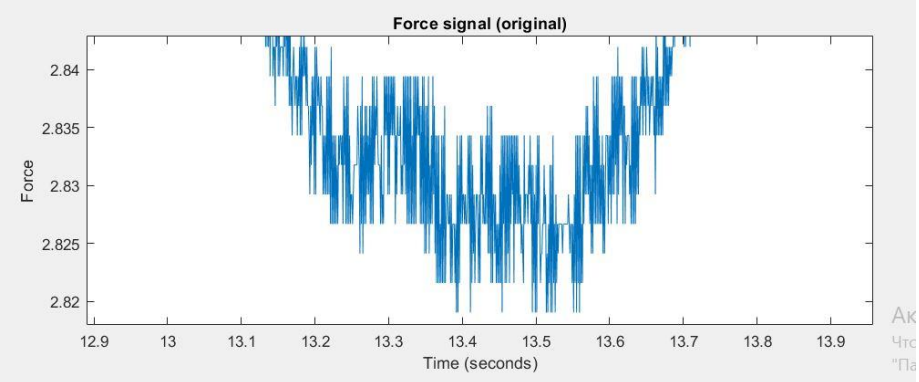


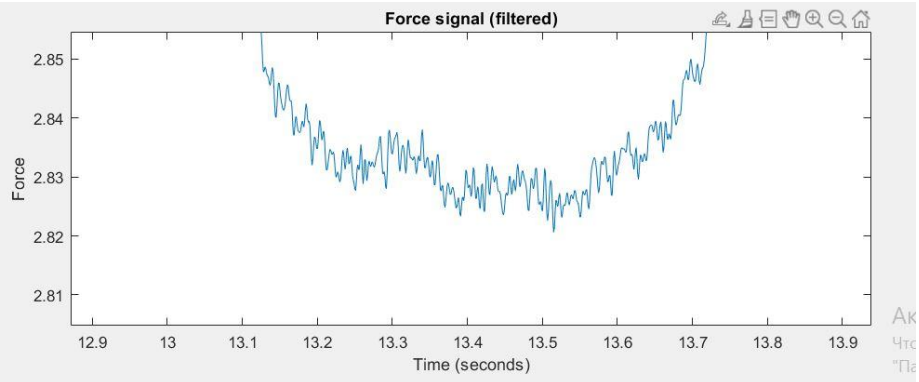
Test case 2: Force signal

For the Force signal, in contrast, the difference are more noticeable.



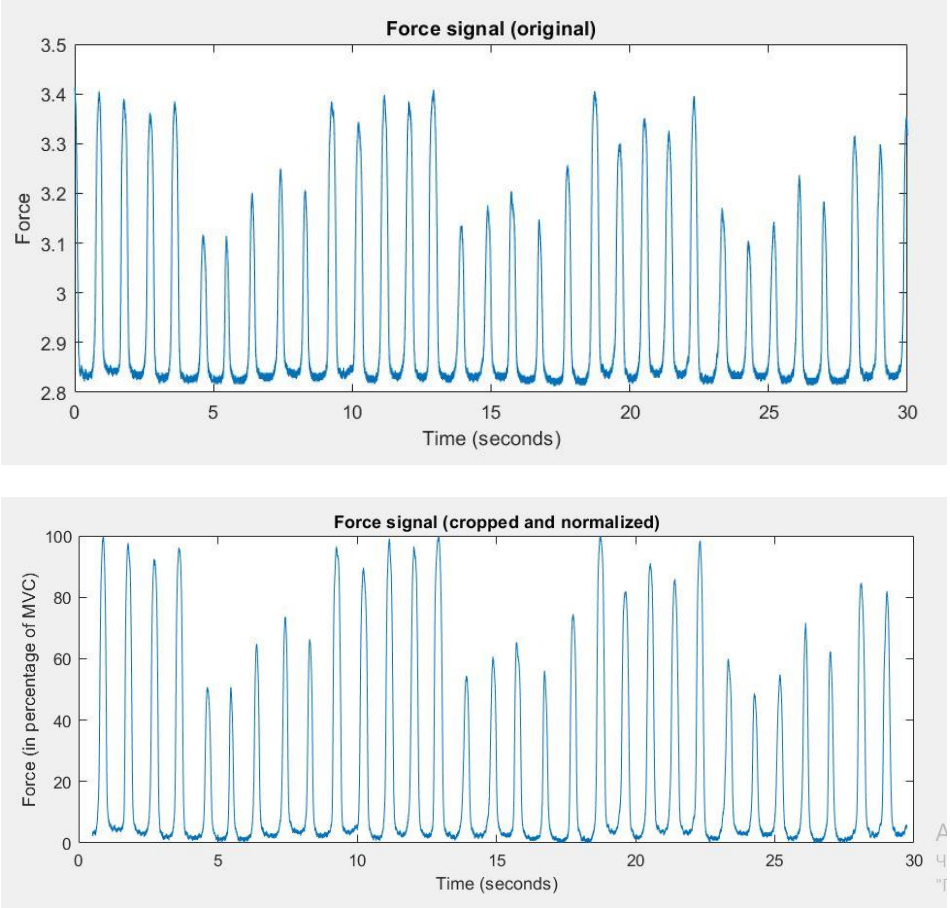






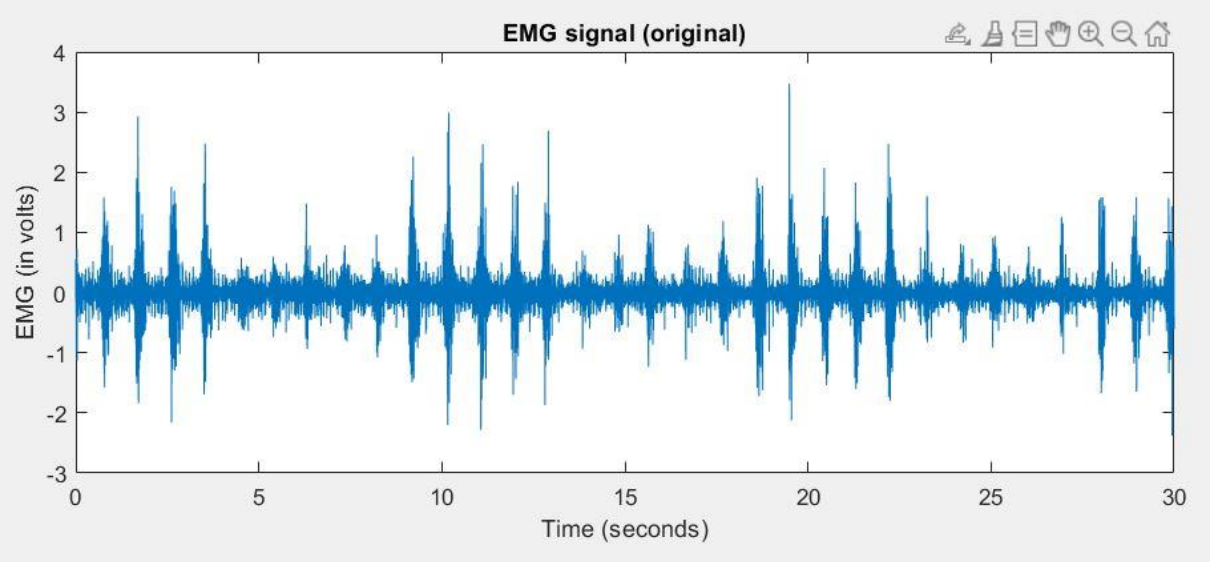
2. For the Cropping/Normalization Testing

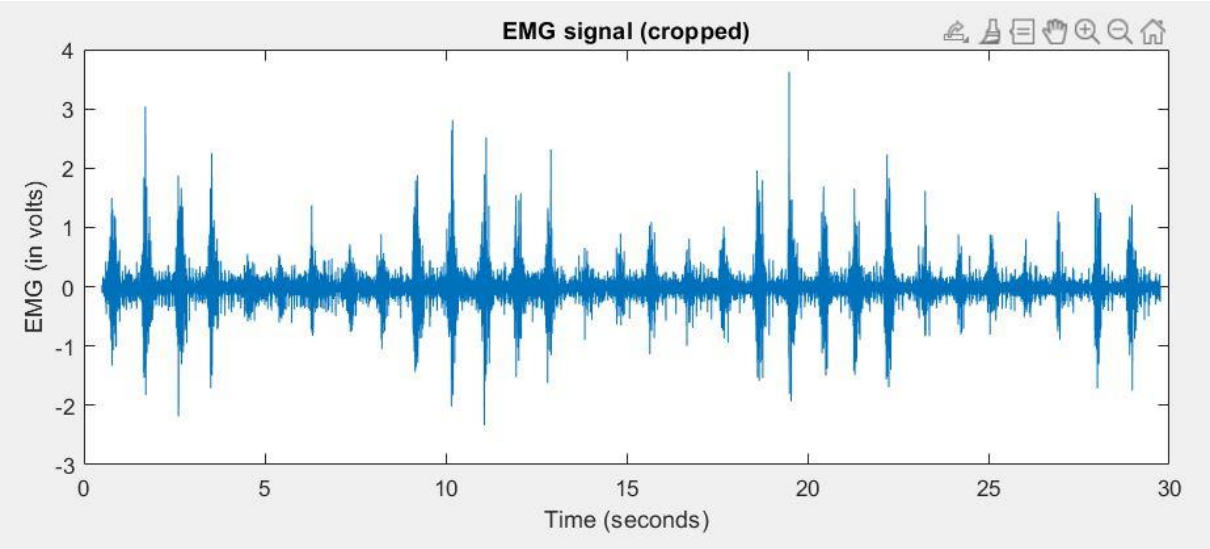
Test case 1: Force signal



As it can be seen, the Force signal was normalized (0 to 100) and endings of it were cropped

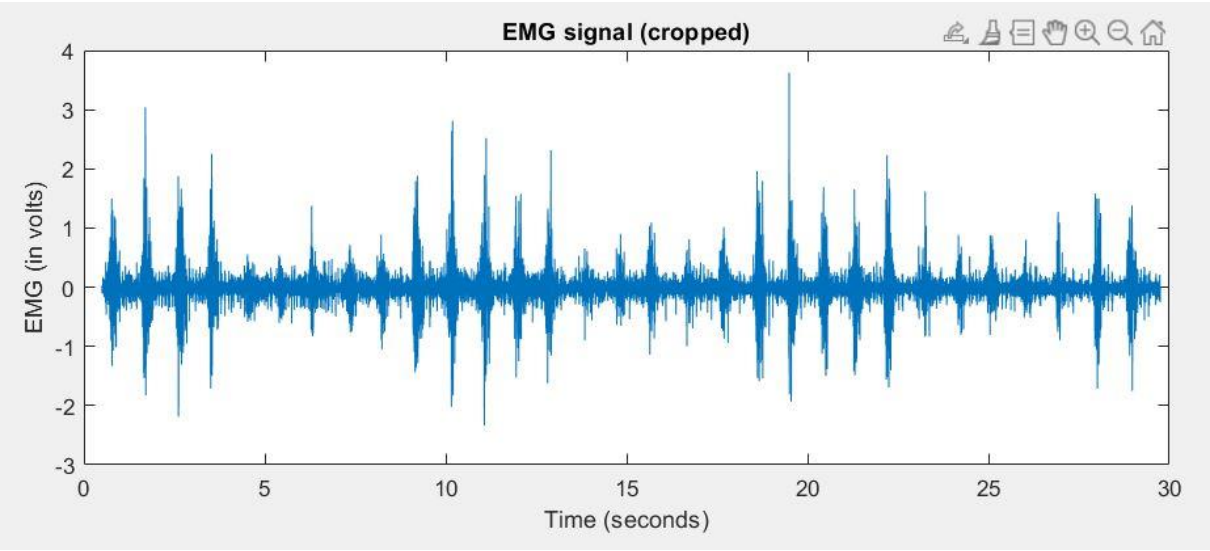
Test case 2: EMG signal

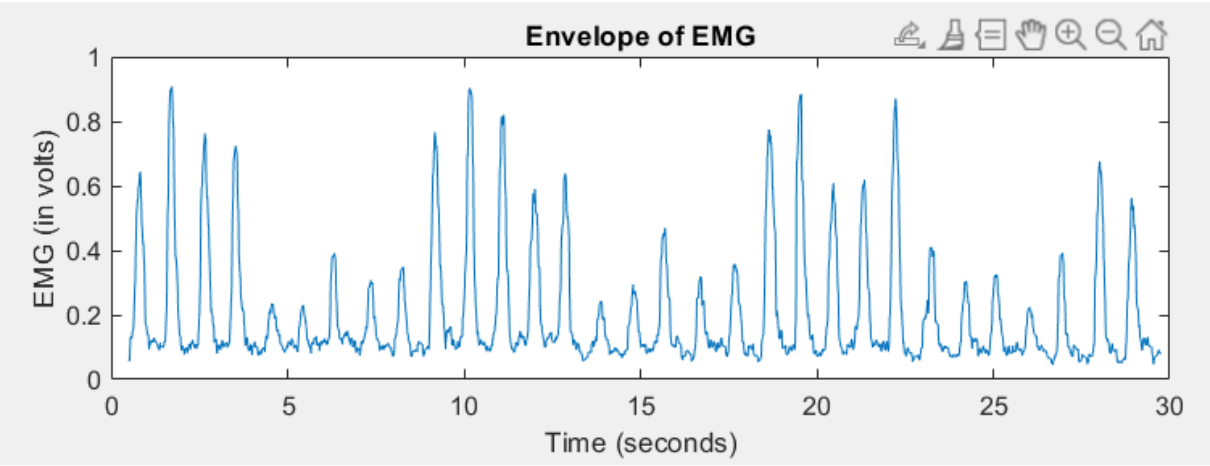




As it can be seen from the snapshots, the endings of the EMG signal were cropped.

3. For the Envelope Function Testing





As it can be seen, the rms envelope of the EMG signal is much smoother and has no negative points.

4. For the Peak Finding Testing

	1
1	322
2	1217
3	2156
4	3026
5	4072
6	4943
7	5822
8	6864
9	7776
10	8675
11	9666
12	10632
13	11513
14	12368
15	13383
16	14290
17	15198

18	16205
19	17183
20	18142
21	19058
22	19970
23	20850
24	21732
25	22733
26	23739
27	24589
28	25520
29	26465
30	27517
31	28425

Peaks for the EMG (rms) signal

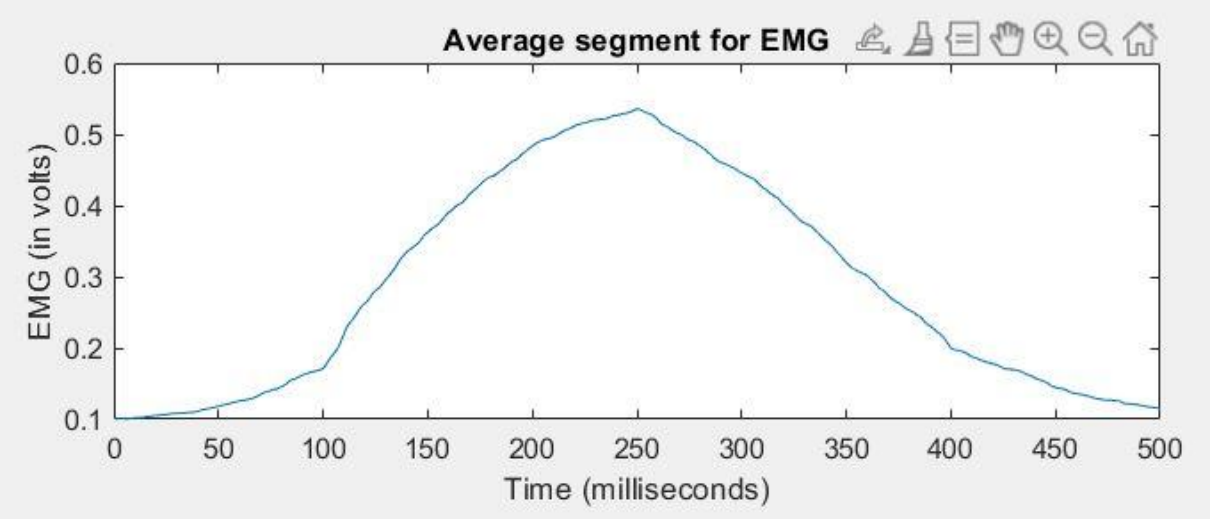
	1
1	385
2	1275
3	2212
4	3105
5	4128
6	4967
7	5897
8	6920
9	7799
10	8759
11	9728
12	10660
13	11541
14	12437
15	13424
16	14380
17	15239

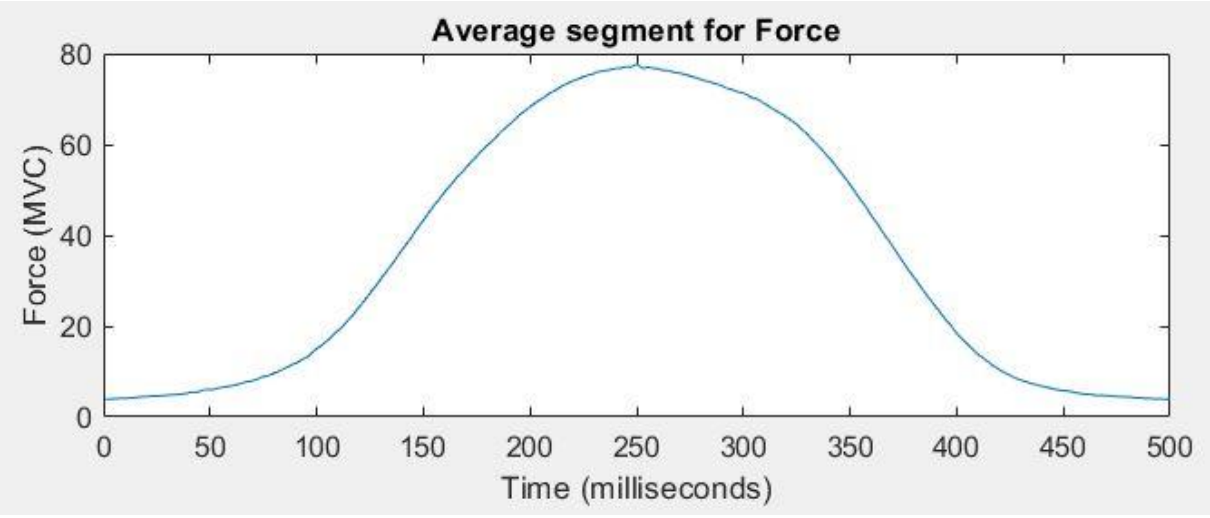
18	16230
19	17262
20	18249
21	19150
22	20030
23	20908
24	21825
25	22831
26	23766
27	24688
28	25607
29	26503
30	27614
31	28532

Peaks for the Force signal

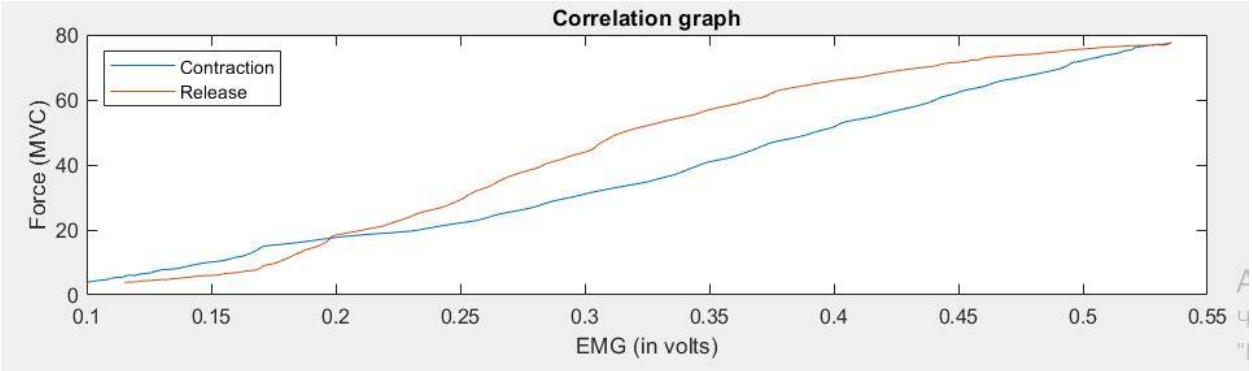
The peaks found are as expected. Exactly 31 of them and the intervals between them are almost equal.

5. For the Segmentation Testing



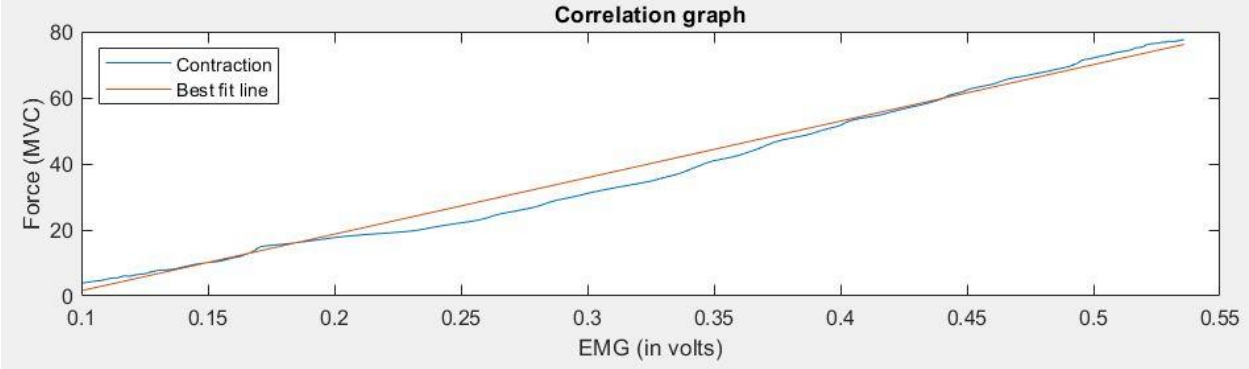


6. For the Correlation/Polyfit function Testing



Comments and Analysis:

The correlation graph along with the two graphs above showing the segment averages indicate that the correlation is linear. Both contraction and release parts of the correlation allow us to conclude that the EMG and Force signals are linearly proportional.

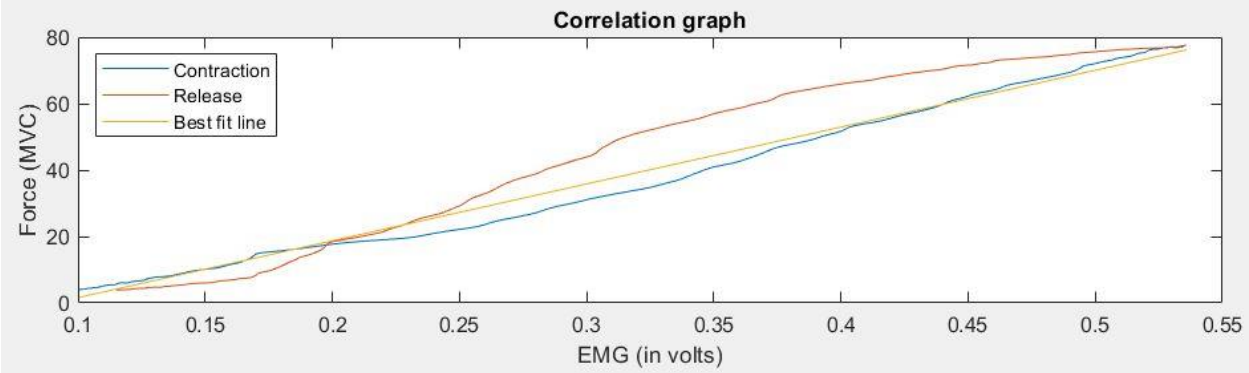


Comments and Analysis:

The best fit function for the contraction part of the cycle was found using the Polyfit function with degree of 1, since the relationship is assumed to be linear. As a result, the line that was calculated very closely matches the actual correlation. This shows that the correlation is indeed linear. However, it also can be seen that there are some parts of the graph doesn't fit the other graph perfectly. The reason for this is assumed to be the errors in rms enveloping, errors of the sensor, errors that arise after filtering, and so on. The equation of the best fit line is the next:

$$y = kx + m$$

, where k = 171.1079 and m = -15.5077



The screenshot of the graph that will be displayed when the program is run.

Overall, the screenshots above are in agreement with the test cases expected output.

Thus, we conclude that the program is functioning correctly since all test cases are verified.

Important Notes:

1. *To execute the program, the user needs to compile and run the code found in the file named Assignment5.m*
2. *There is no menu to interact with a user. When the program is run, it starts the calculations and graphs plotting automatically and no actions are required from the user.*
3. *The data (Force signal, EMG signal, timestamps) should be included in a data file Data.m. The Data.m is assumed to be located in the MATLAB folder together with Assignment5.m*
4. *All the important values such as data length, window length (for rms envelope), segment length are assigned to variables, and thus, can be easily adjusted in case of user's need.*