Appendices C: Matlab Code:

Part 1: Main code for data visualisation

```
file ='emg_6.lvm';
emg =importdata(file,'\t',23);
a = emg.data;
% column 1 in excel sheet, time
emgX=emg.data(:,1);
% column 2 in excel sheet, the amplitude
emgY=emg.data(:,2);

%% Plot real-time EMG signal
figure(1)
plot(emgX,emgY)
xlabel('Seconds'); ylabel('Amplitude (V)'); legend('Raw Time EMG signal')
grid on
title('Raw EMG signal Case 6')

%% Rectifying the signal
figure(2)
rec_emgY=abs(emgY);      %taking absolute values for all amplitude values
plot(emgX, rec_emgY)
xlabel('Seconds')
ylabel('Amplitude (V)')
grid on
title('Rectified EMG signal Case 6'), legend('Rectified EMG signal')

%% PSD
figure(3)
fs = 1000;
n =length(emgY);
Y = fft(emgY);
Pyy = Y.*conj(Y);
f =fs*(0:n/2-1);
nyquist = 1/(2*0.001);
freq =(1:n/2)/(n/2)*nyquist;
plot(freq,Pyy(1:n/2))
grid on;
title('Power Spectral Density 6');
xlabel('Frequency (Hz)');
ylabel('Power');

%% Plot figure 1,2,3 in one figure
figure(4)
subplot(3,1,3)
fs = 1000;
n =length(emgY);
Y = fft(emgY);
Pyy = Y.*conj(Y);
f =fs*(0:n/2-1);
nyquist = 1/(2*0.001);
freq =(1:n/2)/(n/2)*nyquist;
P = Pyy(1:n)
plot(freq,Pyy(1:n/2))
grid on;
title('Power spectral density');
xlabel('Frequency (Hz)');
ylabel('Power');


subplot(3,1,2);
plot(emgX, rec_emgY);
xlabel('Seconds (s)') ;
ylabel('Amplitude (V)');
grid on;
title('Rectified EMG Signal'), legend('Rectified EMG signal')

subplot(3,1,1)
plot(emgX, emgY);
xlabel('Seconds (s)') ;
ylabel('Amplitude (V)');
grid on;
title('Raw EMG Signal'), legend('Raw EMG signal')
```

```matlab
%% Numerical Values
Ma = max(emgY)
Mi = min(emgY)
r =rms(rec_emgY)
s = std(emgY)
freq1 = medfreq(emgY,fs)
freq2 = meanfreq(emgY,fs)

%% DWT / CWT
[lHaar, hHaar] = dwt(emgY, 'haar');
figure(5)
subplot(1,2,1);
plot([lHaar, hHaar])
subplot(1,2,2);
[lHaar2, hHaar2] = cwt(emgY, 'morse');
plot([lHaar2, hHaar2])

%% RMS envelope with Raw EMG Signal
E = envelope(emgY,150,'rms')
figure(6)
plot (emgX,emgY)
hold on
plot(emgX,E,'r')
xlabel('Seconds (s)') ;
ylabel('Amplitude (V)');
grid on;
title('EMG Signal with RMS envelope')

%% Peak Envelope
figure(7)
[yupper,ylower] = envelope(emgY,500,'peak')
plot ([yupper,ylower] )
hold on
plot (emgY,'k')
xlabel('Samples') ;
ylabel('Amplitude (V)');
title('EMG Signal with Peak envelope')
%% Wavelet Transform Fail
wt = cwt(emgY,1000)
plot (wt)

%% Wavelet Fail
freq = 20: .1 : 500;
fs = scal2frq (freq, 'cmor4-2.5', 0.004);
C = cwt(emgY (1:length(emgY)) ,fs, 'cmor4-2.5')
coefs = wcodemat(abs(C),80,'mat',1);
image ( (1:length(emgY))./250, wrev(freq), coefs);
xlabel('Time (s)') ;
ylabel('Frequency (Hz)');
title('Time Frequency Analysis using CWT')

%% WT Denoising
[SIGDEN,~,thrParams,~,BestNbOfInt] = cmddenoise(emgY,'db3',6);

plot(emgY,'k')
hold on;
set(gca, 'Xlim', [500,1000]) ;

plot(SIGDEN)
title('Denoised Signal vs. Original Signal')
set(gca, 'Xlim', [500,1000]) ;
grid on
xlabel('No. of Samples'); ylabel('Amplitude (V)');
legend('Original Signal','Denoised Signal','Location','NorthEastOutside')

%% DWT Haar
x = emgY
sampling_rate = 1000

subplot(4,2,1)
plot(x);
xlabel('Time (seconds)');
title('Oringinal Signal');
axis([0 256 -1 1]);

subplot(4,2,2)
scale_level = 1;
y = DWT2(x, sampling_rate, scale_level);
xlabel('Time (seconds)');
ylabel('Frequency (Hz)');
```

```
title('Scale Level = 1');

subplot(4,2,3)
scale_level = 2;
y = DWT2(x, sampling_rate, scale_level);
xlabel('Time (seconds)');
ylabel('Frequency (Hz)');
title('Scale Level = 2');

subplot(4,2,4)
scale_level = 3;
y = DWT2(x, sampling_rate, scale_level);
xlabel('Time (seconds)');
ylabel('Frequency (Hz)');
title('Scale Level = 3');

subplot(4,2,5)
scale_level = 4;
y = DWT2(x, sampling_rate, scale_level);
xlabel('Time (seconds)');
ylabel('Frequency (Hz)');
title('Scale Level = 4');

subplot(4,2,6)
scale_level = 5;
y = DWT2(x, sampling_rate, scale_level);
xlabel('Time (seconds)');
ylabel('Frequency (Hz)');
title('Scale Level = 5');

subplot(4,2,7)
scale_level = 6;
y = DWT2(x, sampling_rate, scale_level);
xlabel('Time (seconds)');
ylabel('Frequency (Hz)');
title('Scale Level = 6');

subplot(4,2,8)
scale_level = 7;
y = DWT2(x, sampling_rate, scale_level);
xlabel('Time (seconds)');
ylabel('Frequency (Hz)');
title('Scale Level = 7');
```

Part 2: DWT2.m for testing using Wavelet Transform

```
function y = DWT2(x, sampling_rate, scale_level)
%
%  y = DWT2(x, sampling_rate, scale_level)
%
%  DWT2 produces the discrete wavelet Haar transform of "x", for
%  the number of "scale-levels" indicated.  The length of "x" is
%  expected to be a power of 2; otherwise, it is zero padded to the
%  next power of 2 greater than the length of "x".  "Sampling_rate"
%  is the time increment between samples in seconds.  The output image
%  is normalized to show the high frequencies.
%

if (nargin ~= 3)
      disp('DWT2 requires 3 input arguments!')
   return;
end
if ((scale_level < 1) | (round(scale_level) ~= scale_level))
    disp('The argument "scale_level" must be an integer greater than 0.');
   return;
end
if (sampling_rate <= 0)
    disp('The argument "sampling_rate" must be greater than 0.');
   return;
end

[m,n] = size(x);
if (m ~= 1)
    X = x';
else
    X = x;
end
[m,n] = size(X);
```

```
if ( m ~= 1)
    disp(' X must be a vector, not a matrix!');
    return;
end

LENX = length(X);
PAD = 2^(ceil(log2(LENX))) - LENX;
XX = [X zeros(1, PAD)];
LENXX = length(XX);

if (scale_level > log2(LENXX))
    disp(' The argument "scale_level" is too large.');
    return;
end

SL = 2^scale_level;
SLT = SL;
y = zeros((LENXX/2), SL);
y_img = zeros((LENXX/2), SL);

h0 = [sqrt(2)/2 sqrt(2)/2];
h1 = [-sqrt(2)/2 sqrt(2)/2];

TY = [];
c = XX;
for jj = 1:1:scale_level
    L = length(c);
    d = conv(c, h1);
    d = d(1:2:L);
    c = conv(c, h0);
    c = c(1:2:L);
    TY = [d TY];
end
TY = [c TY];

TY = fliplr(TY);
ST = 1;     COL = SL;
for kk = 1:1:(scale_level+1)

% CHOP UP DATA VECTOR FROM TY INTO T3, ITERATIVELY.

    SLT = SLT/2;
    SP = 2^(kk-1);
    SA = LENXX/(SP*2);
    T3 = zeros(1,(LENXX/2));
   if (kk ~= (scale_level+1))
       T3(1:SP:(LENXX/2)) = TY(ST:1:(ST+SA-1));
       ST = ST + SA;
    else
       SLT = SLT*2;
       SP = SP/2;
       T3(1:SP:(LENXX/2)) = TY(ST:1:LENXX);
    end

% SHIFT AND COPY T3 TO FILL IN TIME SLOTS AT LOWER FREQUENCIES.

   for nn = 2:1:SP
       T2 = [zeros(1,(nn-1)) T3];   T2 = T2(1:1:LENXX/2);
       T3 = T3 + T2;
    end

% NORMALIZE T3

    T3 = abs(T3)/max(abs(T3));

% REPEAT T3 VECTOR TO FILL IN FREQUENCY SLOTS IN MATRIX "y".

   for mm = SLT:-1:1
       y(:,COL) = T3';
       y_img(:,COL) = T3';
       COL = COL - 1;
    end
end

y = flipud(y);
y_img = flipud(y_img);

freq = (1/sampling_rate)/2;
imagesc([0:sampling_rate*2:(sampling_rate*(LENXX-1))],[0 (freq/(SL-1))*[1:1:(SL-1)]],(y_img').^2);
xlabel(' Time (seconds)');
```

```
ylabel('Frequency (Hz)');
axis([0 (sampling_rate*LENXX) 0 freq]);
axis('xy')
```

Part 3: Smoothing Matlab built-in function

```
%% X axis limit values changes for each case based on the peaks observed
figure()
s1 = smooth(rec_emgY,'lowess')
subplot(6,1,1)
plot(emgX, s1)
set(gca, 'Xlim', [8, 9]) ;
title('Comparison of Different Smoothing Methods for Selected Segment')

s2 = smooth(rec_emgY,'moving')
subplot(6,1,2)
plot(emgX, s2)
set(gca, 'Xlim', [8, 9]) ;

s3 = smooth(rec_emgY,'loess')
subplot(6,1,3)
plot(emgX, s3)
set(gca, 'Xlim', [8, 9]) ;

s4 = smooth(rec_emgY,'sgolay')
subplot(6,1,4)
plot(emgX, s4)
set(gca, 'Xlim', [8, 9]) ;

s5 = smooth(rec_emgY,'rlowess')
subplot(6,1,5)
plot(emgX, s5)
set(gca, 'Xlim', [8, 9]) ;

s6 = smooth(rec_emgY,'rloess')
subplot(6,1,6)
plot(emgX, s6)
set(gca, 'Xlim', [8, 9]) ;
xlabel('Seconds')
ylabel('Amplitude (V)')
```