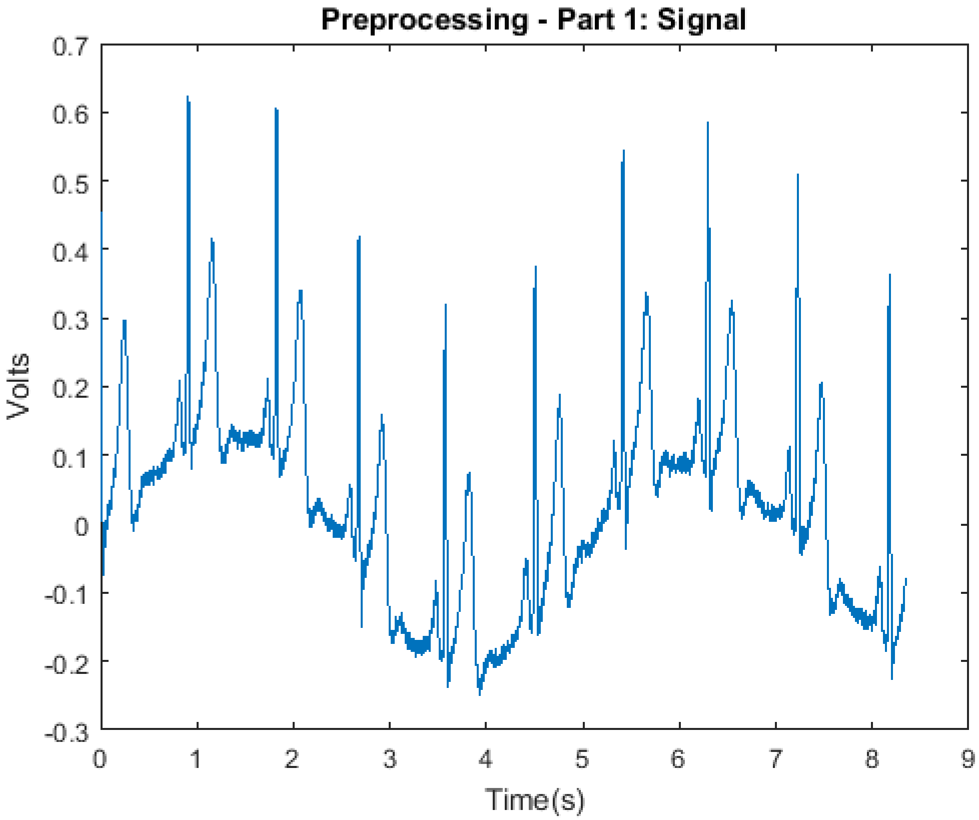
Preprocessing

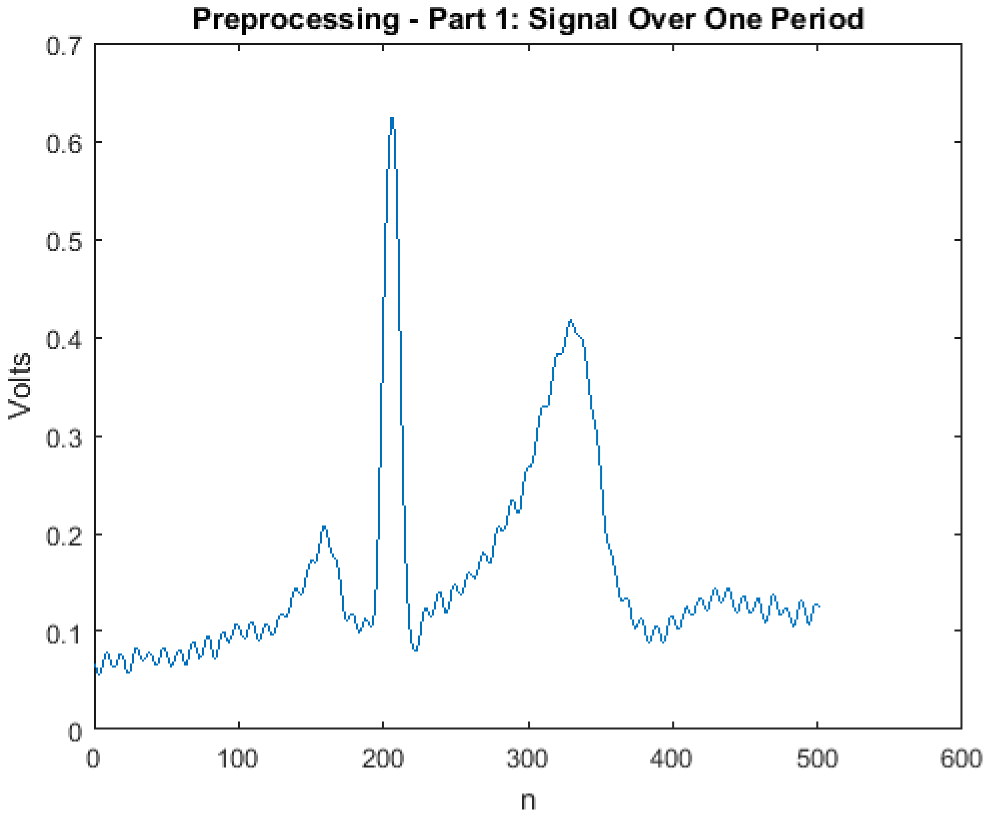
Table of Contents

Part 1 - Visual Signal Validation

After loading the data into the workspace, we plot the signal using sampling frequency in time domain and consequently, over one period.

```
ECG = load('Data.mat');
d = ECG.data.signal;
fs = ECG.data.samplingfreq;
unit = ECG.data.measurement_unit;
x = (250:+1:250+fs);
t = (1:+1:length(d))./fs;
figure(1);
plot(t,d);
xlabel('Time(s)');
ylabel(unit);
title('Preprocessing - Part 1: Signal');
p = d(x);
figure(2);
plot(p);
ylabel(unit);
xlabel('n');
title('Preprocessing - Part 1: Signal Over One Period');
```

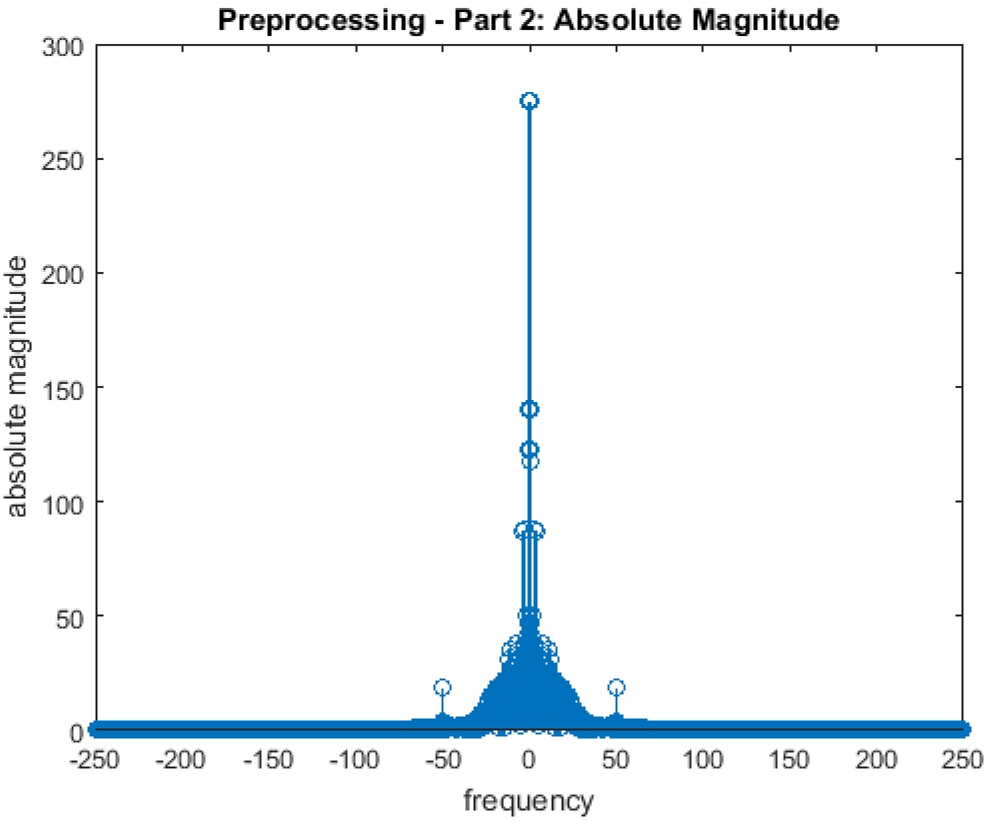

Preprocessing - Part 1: Signal
Volts
0.7
0.6
0.5
0.4
0.3
0.2
0.1
0
-0.1
-0.2
-0.3
0
1
2
3
4
5
6
7
8
9
Time(s)

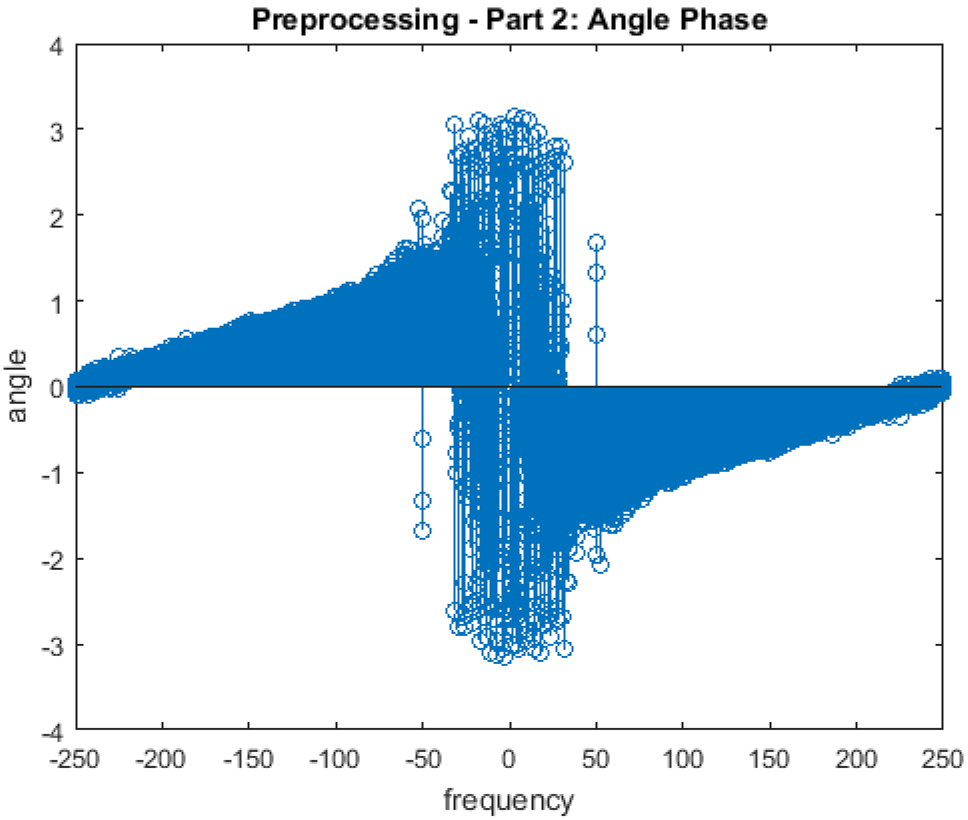

Preprocessing - Part 1: Signal Over One Period
Volts
0.7
0.6
0.5
0.4
0.3
0.2
0.1
0
0
100
200
300
400
500
600
n

Part 2 - FFT of the Signal

We plot the signal in frequency domain using techniques learnt in previous assignments.

```
y = fft(d);
d_f = fftshift(y);
ny = length(y);
fd_f = (-ny/2:ny/2-1)*(fs/ny);
figure(3);
stem(fd_f, abs(d_f));
xlabel('frequency');
ylabel('absolute magnitude');
title('Preprocessing - Part 2: Absolute Magnitude');
figure(4);
stem(fd_f, angle(d_f));
xlabel('frequency');
ylabel('angle');
title('Preprocessing - Part 2: Angle Phase');
```



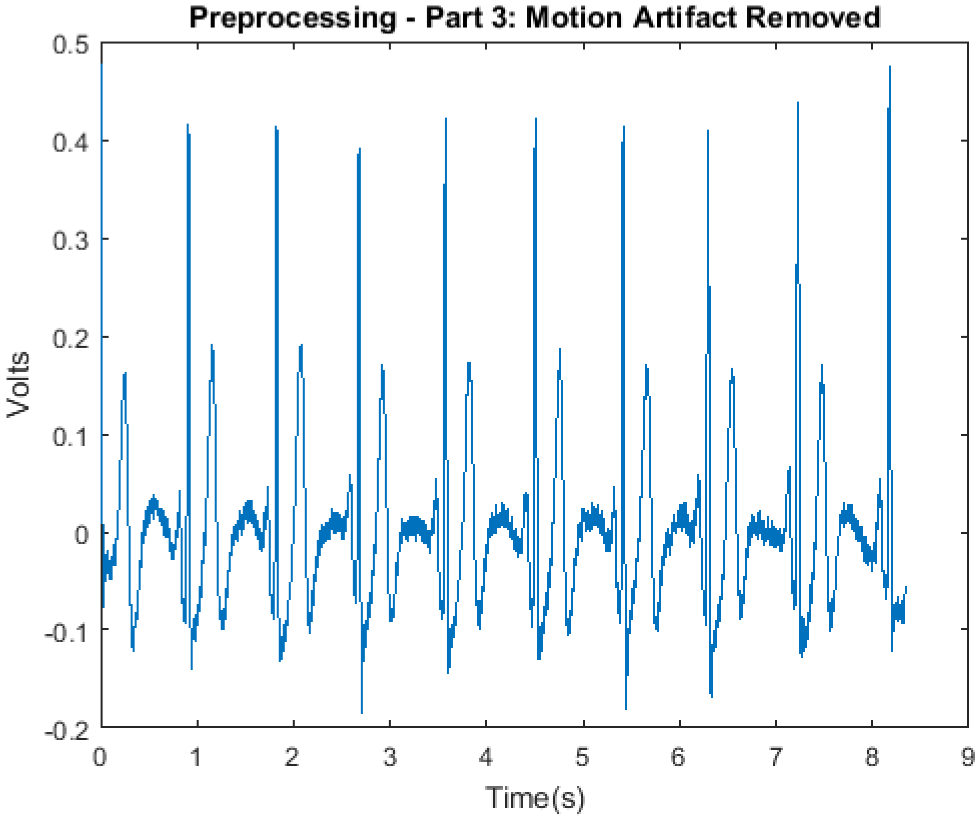


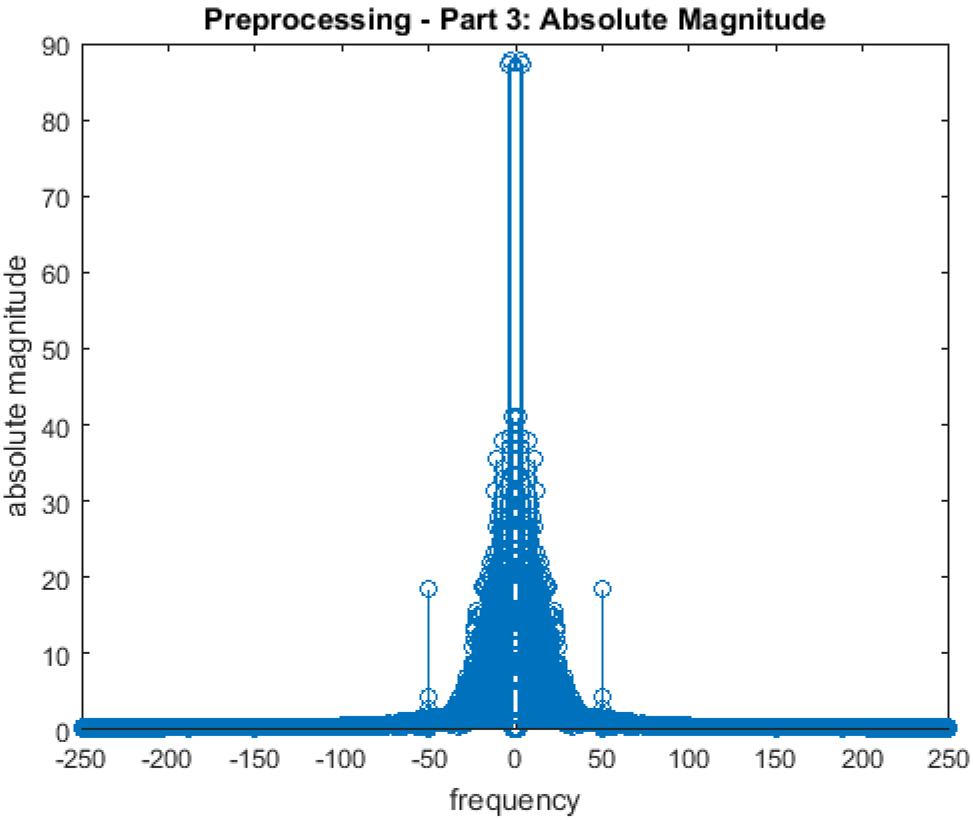
Part 3 - Motion Artifact Removal

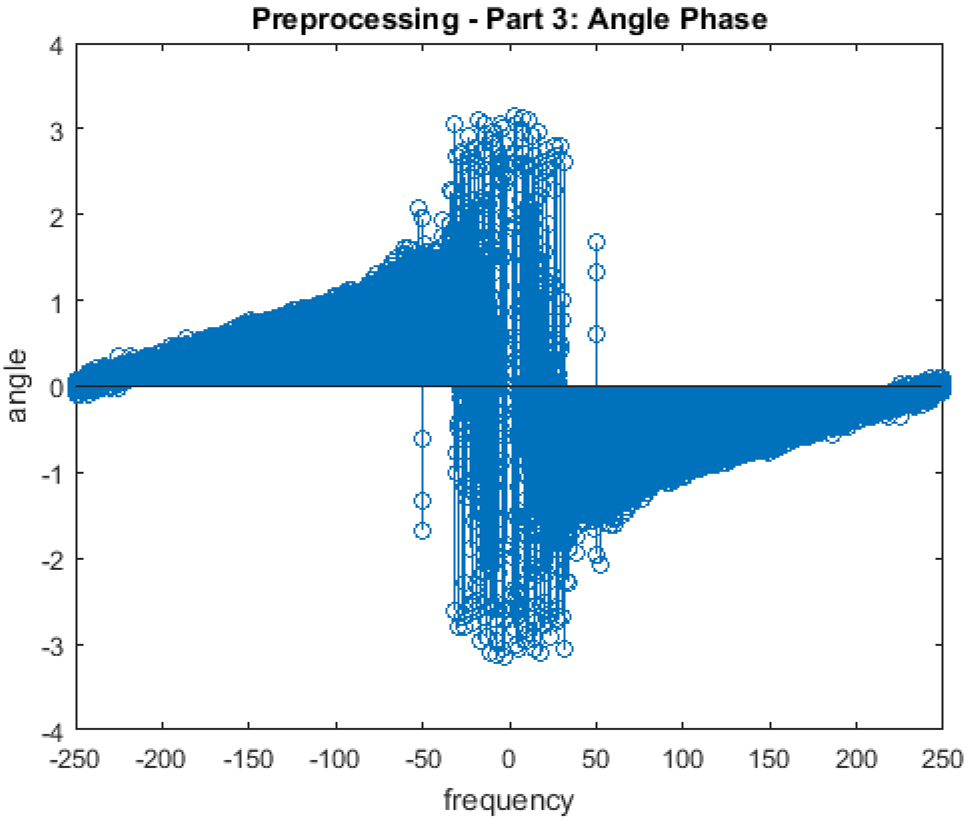
By a little amount of search through the internet, we figure that the appropriate cut off frequency is 2 Hz. We can easily design an ideal high pass filter by just assigning 0 to the values in the corresponding interval.

```
cof = 2; % cut off frequency
b = cof.*ny./fs; % stop band
hb = floor(b); % high band
lb = ceil(-1.*b); % lower band
mar_f = d_f; % motion artifact removed
mar_f([lb+ny/2+1:hb+ny/2+1])=0; % ideal high pass filter
mar = ifft(ifftshift(mar_f));
figure(5);
plot(t,mar);
ylabel(unit);
xlabel('Time(s)');
title('Preprocessing - Part 3: Motion Artifact Removed');
figure(6);
stem(fd_f, abs(mar_f));
xlabel('frequency');
ylabel('absolute magnitude');
title('Preprocessing - Part 3: Absolute Magnitude');
figure(7);
stem(fd_f, angle(mar_f));
xlabel('frequency');
ylabel('angle');
```

```
title('Preprocessing - Part 3: Angle Phase');
```



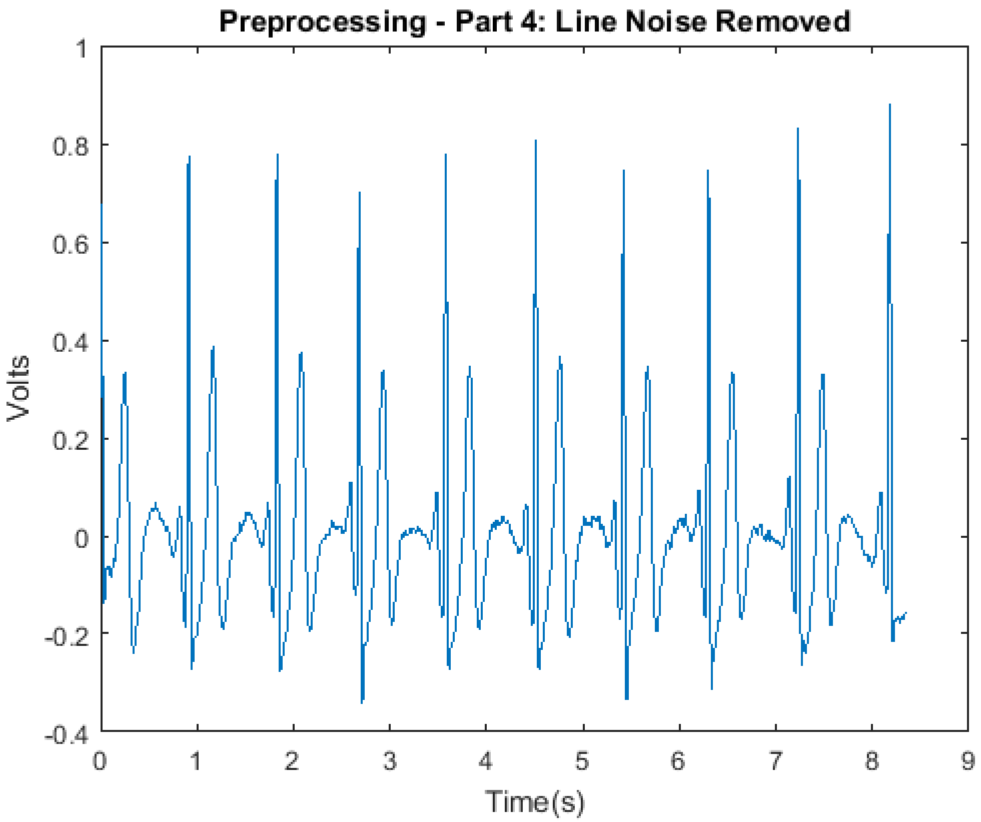

Preprocessing - Part 3: Absolute Magnitude
absolute magnitude
90
80
70
60
50
40
30
20
10
0
-250
-200
-150
-100
-50
0
50
100
150
200
250
frequency

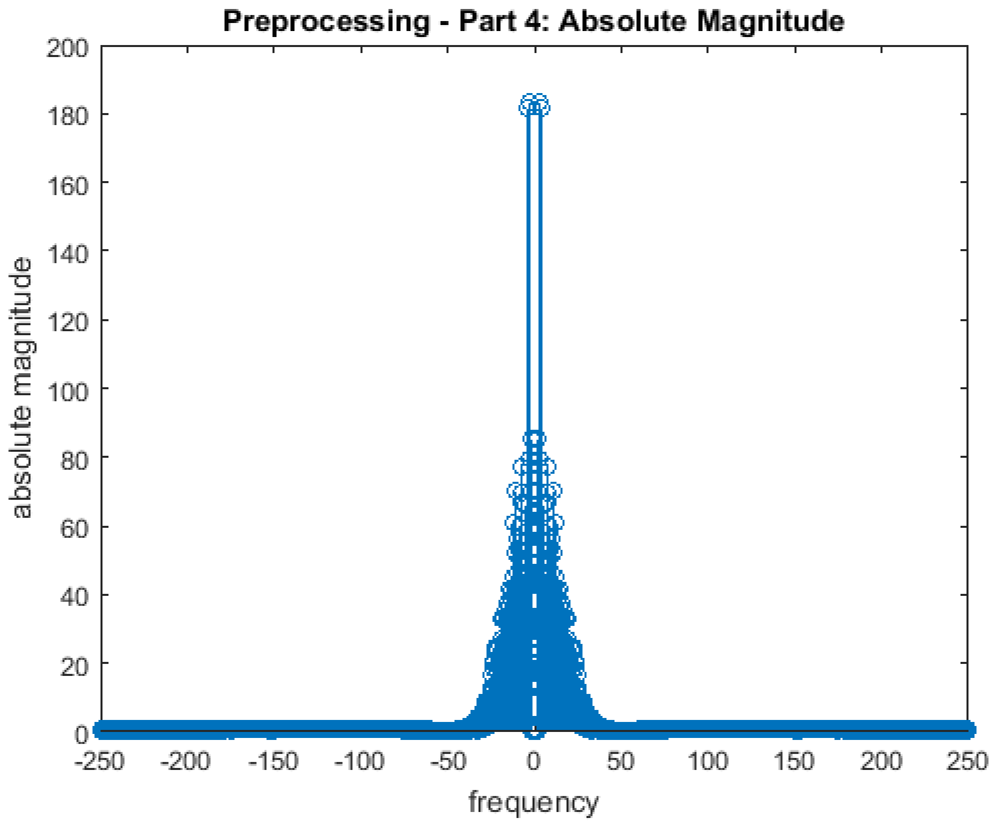

Preprocessing - Part 3: Angle Phase
angle
4
3
2
1
0
-1
-2
-3
-4
-250
-200
-150
-100
-50
0
50
100
150
200
250
frequency

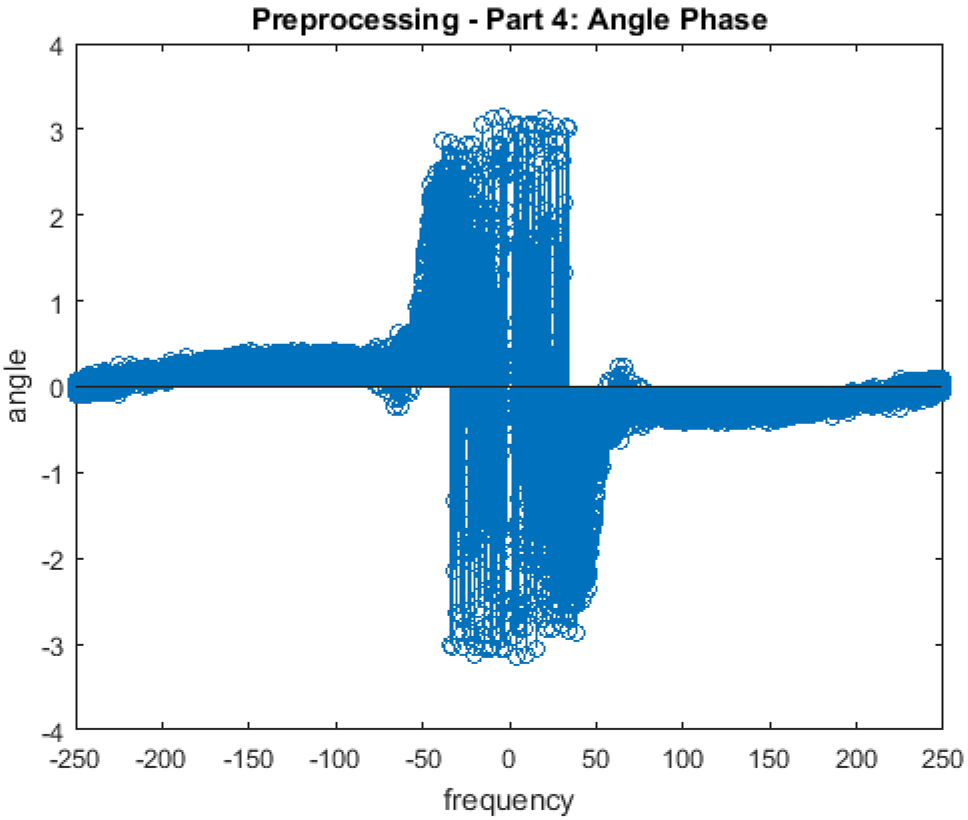
Part 4 - Line Noise Removal

Having a glance on the signal in frequency domain, it is easily seen that the line noise is at 50 Hz. We remove this frequency by applying a notch filter with proper bandwidth and attenuation.

```
fn = 50; % notch frequency
bw = 120; % bandwidth
apass = 1; % bandwidth attenuation
[b,a]=iirnotch(fn/(fs/2),bw/(fs/2),apass);
hd=dfilt.df2(b,a);
r = mar./max(mar);
lnr=filter(hd,r); % line noise removed
figure(8);
plot(t,lnr);
ylabel(unit);
xlabel('Time(s)');
title('Preprocessing - Part 4: Line Noise Removed');
y2 = fft(lnr);
lnr_f = fftshift(y2);
ny2 = length(y2);
flnr_f = (-ny2/2:ny2/2-1)*(fs/ny2);
figure(9);
stem(flnr_f, abs(lnr_f));
xlabel('frequency');
ylabel('absolute magnitude');
title('Preprocessing - Part 4: Absolute Magnitude');
figure(10);
stem(flnr_f, angle(lnr_f));
xlabel('frequency');
ylabel('angle');
title('Preprocessing - Part 4: Angle Phase');
```


Preprocessing - Part 4: Line Noise Removed
Volts
1
0.8
0.6
0.4
0.2
0
-0.2
-0.4
0
1
2
3
4
5
6
7
8
9
Time(s)


Preprocessing - Part 4: Absolute Magnitude
absolute magnitude
200
180
160
140
120
100
80
60
40
20
0
-250
-200
-150
-100
-50
0
50
100
150
200
250
frequency

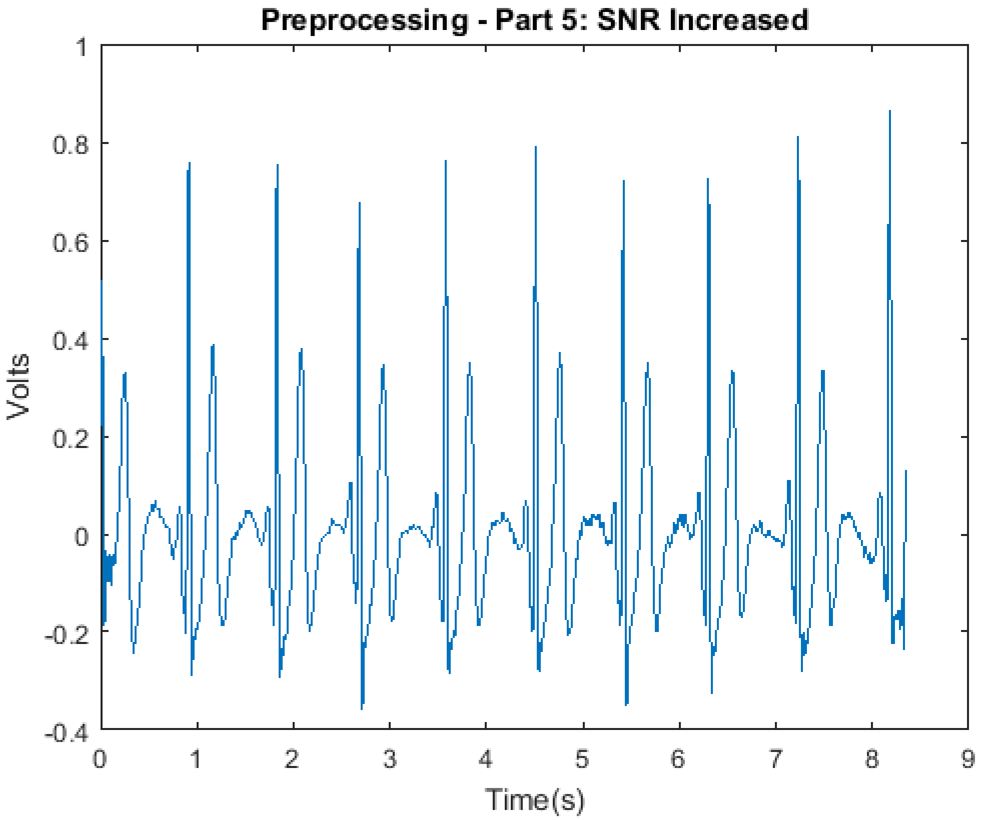


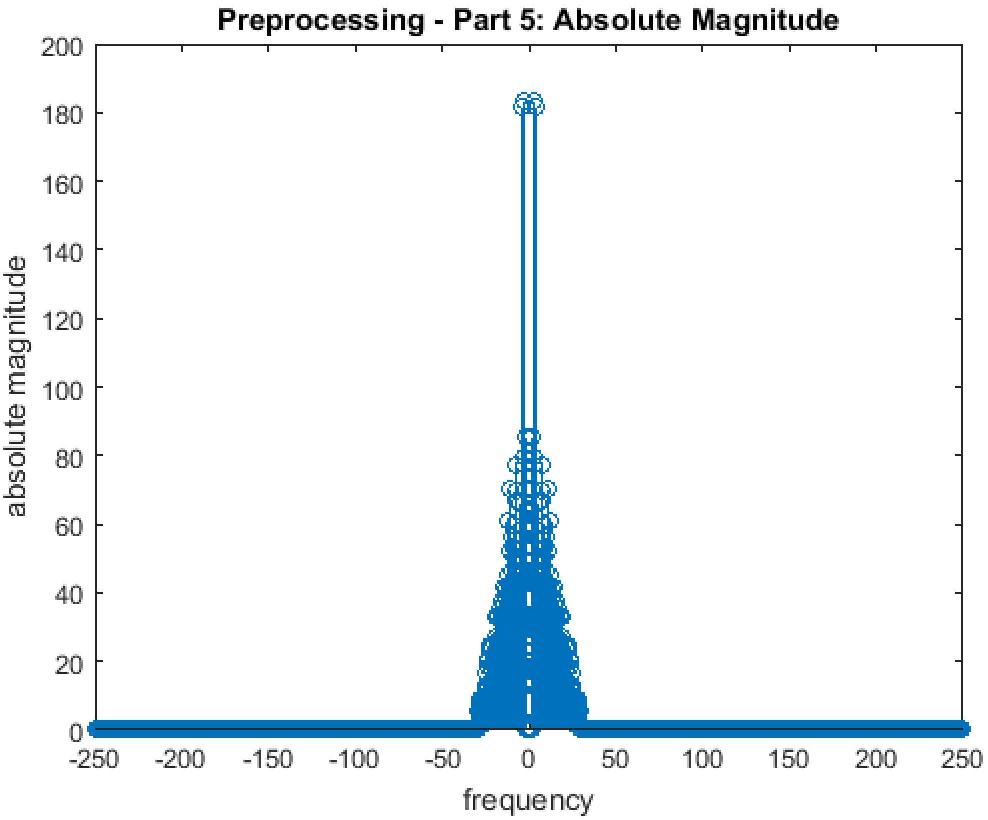
Part 5 - SNR Increasing

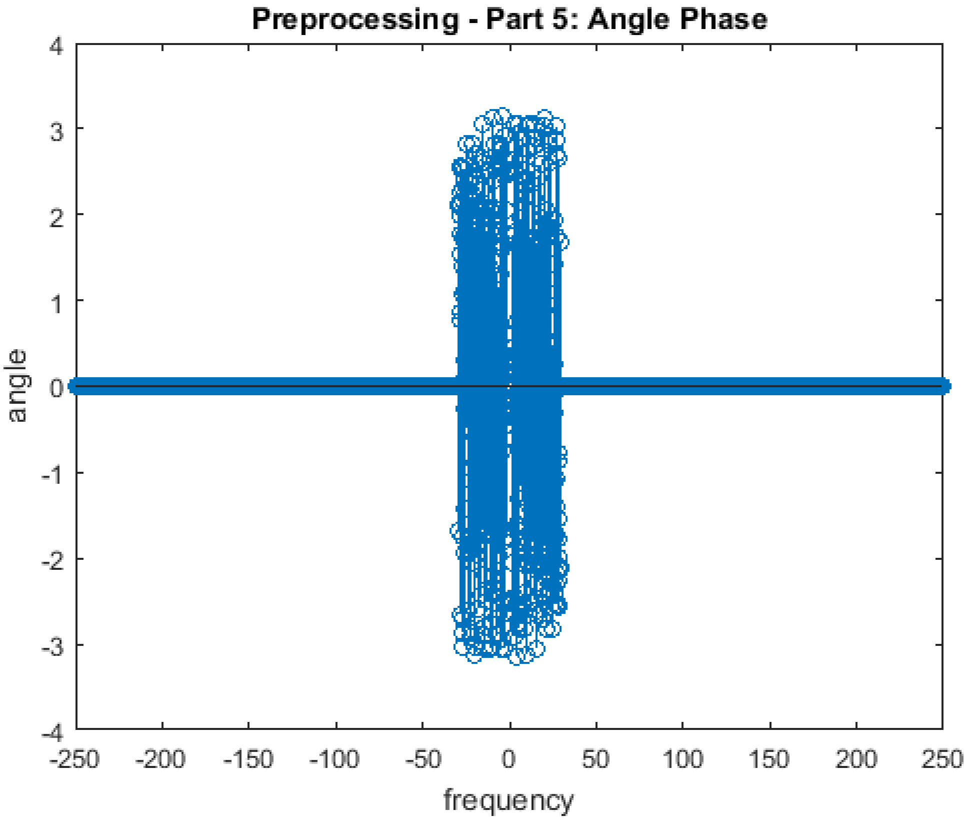
The obw() function gives us the band in which the signal has 99% of its power. So we just basically remove other frequencies out of this range by using an ideal low pass filter.

```
[wd,lo,hi,power]=obw(lnr,fs);
b = hi.*ny./fs; % stop band
hb = floor(b); % high band
lb = ceil(-1.*b); % lower band
snr_inc_f = zeros(ny,1); % SNR increasing
for i=1:ny
    if (i>=lb+ny/2+1 && i<=hb+ny/2+1)
        snr_inc_f(i)=lnr_f(i); % ideal low pass filter
    end
end
snr_inc = ifft(ifftshift(snr_inc_f));
figure(11);
plot(t,snr_inc);
ylabel(unit);
xlabel('Time(s)');
title('Preprocessing - Part 5: SNR Increased');
figure(12);
stem(fd_f, abs(snr_inc_f));
xlabel('frequency');
ylabel('absolute magnitude');
title('Preprocessing - Part 5: Absolute Magnitude');
```

```
figure(13);
stem(fd_f, angle(snr_inc_f));
xlabel('frequency');
ylabel('angle');
title('Preprocessing - Part 5: Angle Phase');
```




Preprocessing - Part 5: Absolute Magnitude
absolute magnitude
200
180
160
140
120
100
80
60
40
20
0
-250
-200
-150
-100
-50
0
50
100
150
200
250
frequency


Preprocessing - Part 5: Angle Phase
angle
4
3
2
1
0
-1
-2
-3
-4
-250
-200
-150
-100
-50
0
50
100
150
200
250
frequency

Published with MATLAB® R2015a