Lab 5 Part 2

Contents

- 2.1 - FFT of a Sinusoid with Non-Integer Frequency
- 2.2 - Zero Padding
- 2.3 - Windowing
- 2.4 - Determining Average Heart Rate Using FFT


```
clear
```

2.1 - FFT of a Sinusoid with Non-Integer Frequency

2.1.1

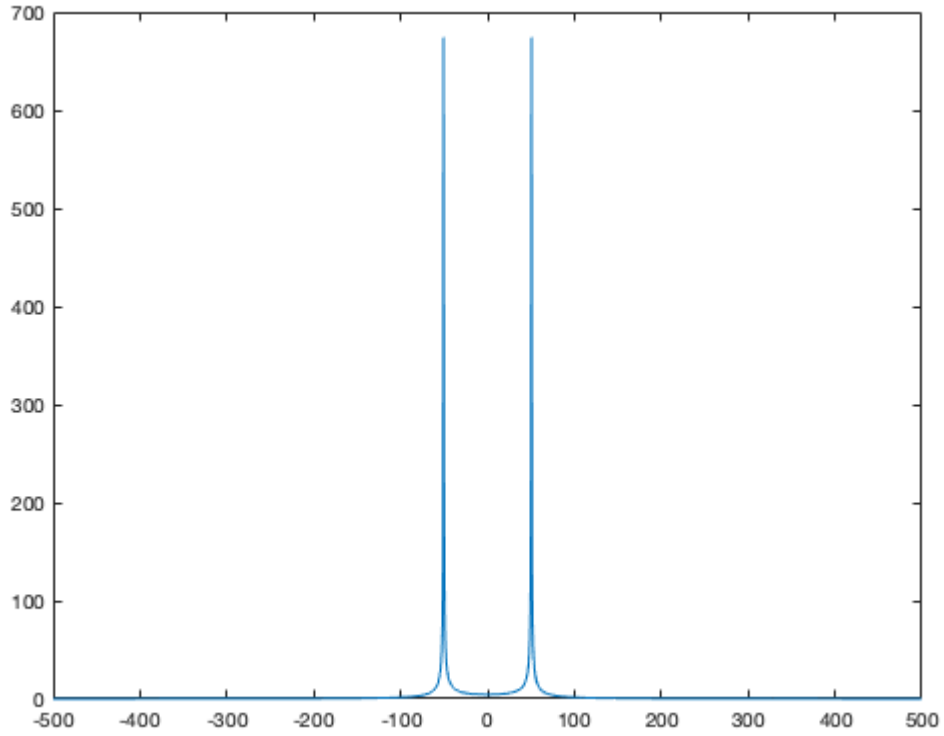
Using 1000 samples/second for fs and a time vector of 1.5 seconds, construct the vector x to represent the given sinusoid.

```
fs = 1000;
tt = 0:1/fs:(1500-1)/fs;
x = cos(2*pi*50.5*tt + 0.25*pi);
```

2.1.2

Use fft() and assign the output to X . Plot the magnitude spectrum. Zoom on the left-most peak. You should notice that the spectrum is nonzero at multiple values surrounding this peak. By introducing a non-integer frequency, we see a spread of energy across the spectrum. Find the frequency bins where this maximum occurs.

```
X = fftshift(fft(x));
ww = -pi:(2*pi/length(X)):(pi-1/length(X));
ff = fs * ww / (2 * pi);
plot(ff, abs(X))

% ff(find(abs(X) > 10))
```



Question 1: Does this exactly match the frequency 50.5 Hz?

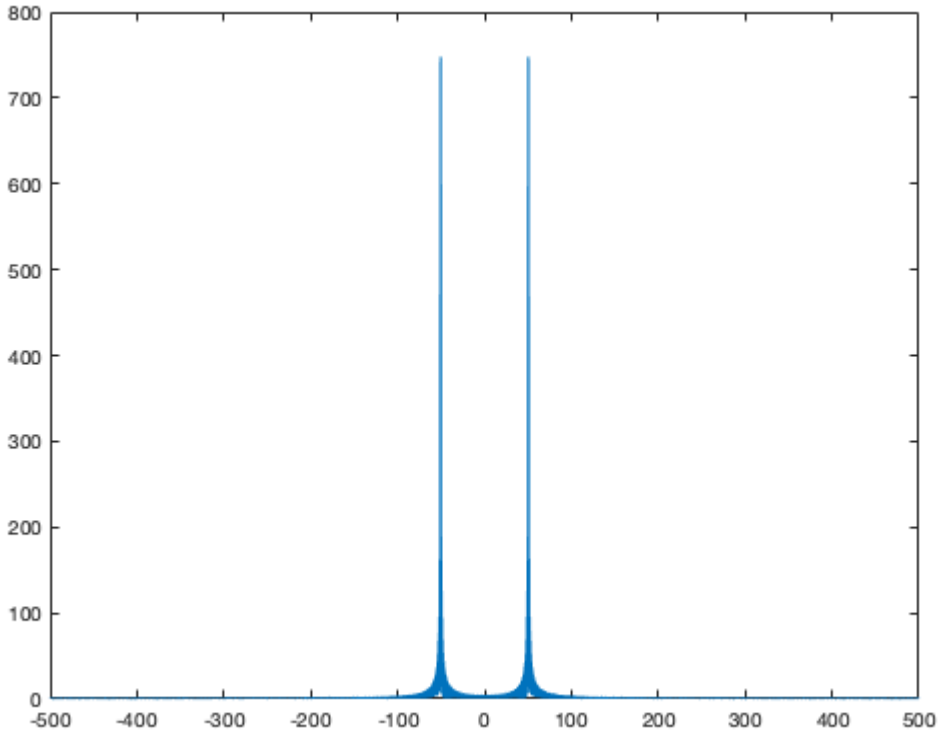
From using the cursor on the plot, it does not **exactly** match the frequency of 50.5 Hz. It is, instead, valued at 50.6667 Hz.

2.2 - Zero Padding

2.2.1

Append 15,000 zeros to our signal from section 1.7 (the one with frequency 50.5 Hz), and run the FFT again.

```
x = [x zeros(1,15000)];
X = fftshift(fft(x));
ww = -pi:(2*pi/length(X)):(pi-1/length(X));
ff = fs * ww / (2 * pi);
plot(ff, abs(X))
```



Question 2: What is the effect of the zero padding on the magnitude spectrum of the signal?

The effect of zero padding was a more precise FFT.

Question 3: What is the frequency of the bin for which the magnitude is maximized?

50.455 Hz was the frequency of the bin that the peak occured at.

2.2.2

The appearance of the magnitude spectrum may be somewhat odd compared to what we have seen before. There should appear to be a main "lobe" and several smaller "side lobes" on the sides.

Question 4: How does the width of these side lobes compare to the main lobe?

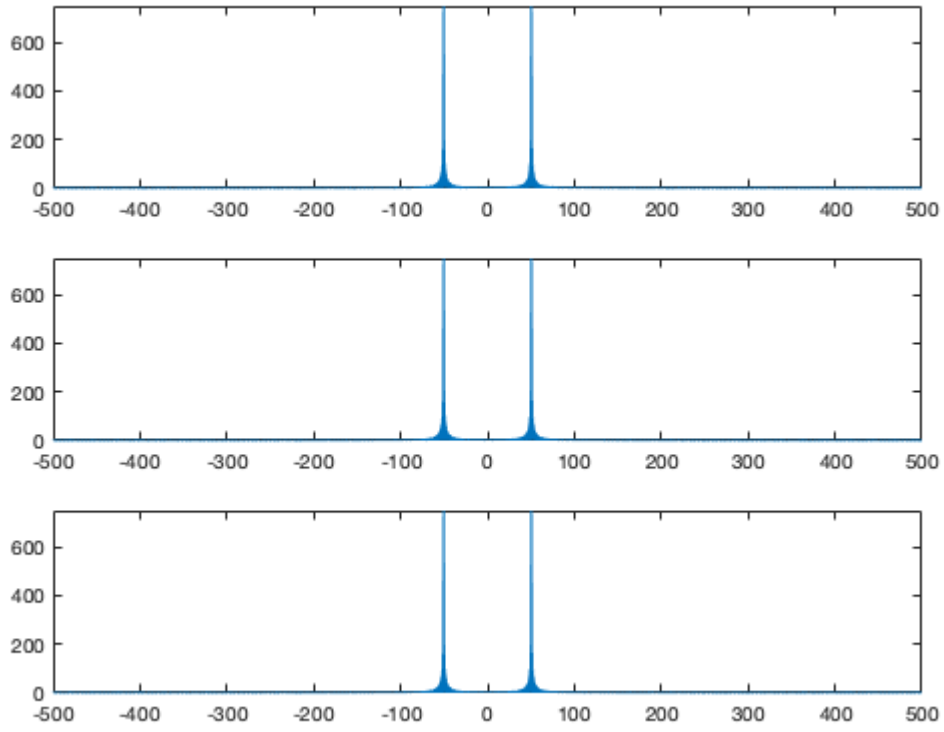
The width of the side lobes is significantly less than the width of the main lobe, as the main lobe is approximately 3 times larger than the side lobes.

2.2.3

Try zero padding the signal with 25,000 zeros, 50,000 zeros, and 100,000 zeros. Using the subplot() function, compare the appearances of the magnitude spectra.

```
x = cos(2*pi*50.5*tt + 0.25*pi);
x = [x zeros(1,25000)];
X = fftshift(fft(x));
```

```
ww = -pi:(2*pi/length(X)):(pi-1/length(X));
ff = fs * ww / (2 * pi);
subplot(3,1,1), plot(ff, abs(X));

x = cos(2*pi*50.5*tt + 0.25*pi);
x = [x zeros(1,50000)];
X = fftshift(fft(x));
ww = -pi:(2*pi/length(X)):(pi-1/length(X));
ff = fs * ww / (2 * pi);
subplot(3,1,2), plot(ff, abs(X));

x = cos(2*pi*50.5*tt + 0.25*pi);
x = [x zeros(1,1000000)];
X = fftshift(fft(x));
ww = -pi:(2*pi/length(X)):(pi-1/length(X));
ff = fs * ww / (2 * pi);
subplot(3,1,3), plot(ff, abs(X));
```



These magnitude spectra appear nearly identical, save for small differences that are very slightly noticeable when the plots are zoomed in.

Question 5: What is the effect of increasing the number of zeros in the accuracy of the frequency calculation?

As more zeroes are padded onto the end of the signal, the resolution of the fft becomes larger and the bins become more accurate. I noticed that after a certain point, it seems as though the resolution didn't increase. Between 50,000 zeroes and 100,000 zeroes, there was no difference in the bin that the peak's frequencyw as placed into.

Question 6: Does the error increase or decrease?

Error decreases as the number of zeros that are padded increases.

Question 7: In the case of our signal, how many zeros would need to be added in order to obtain an exact value for the frequency of our input signal?

As I noted in a previous response, I noticed no increase in the accuracy after 50,000 zeroes. At 50,000 zeroes, we achieved a result of 50.5 Hz, rounded to one decimal place.

2.3 - Windowing

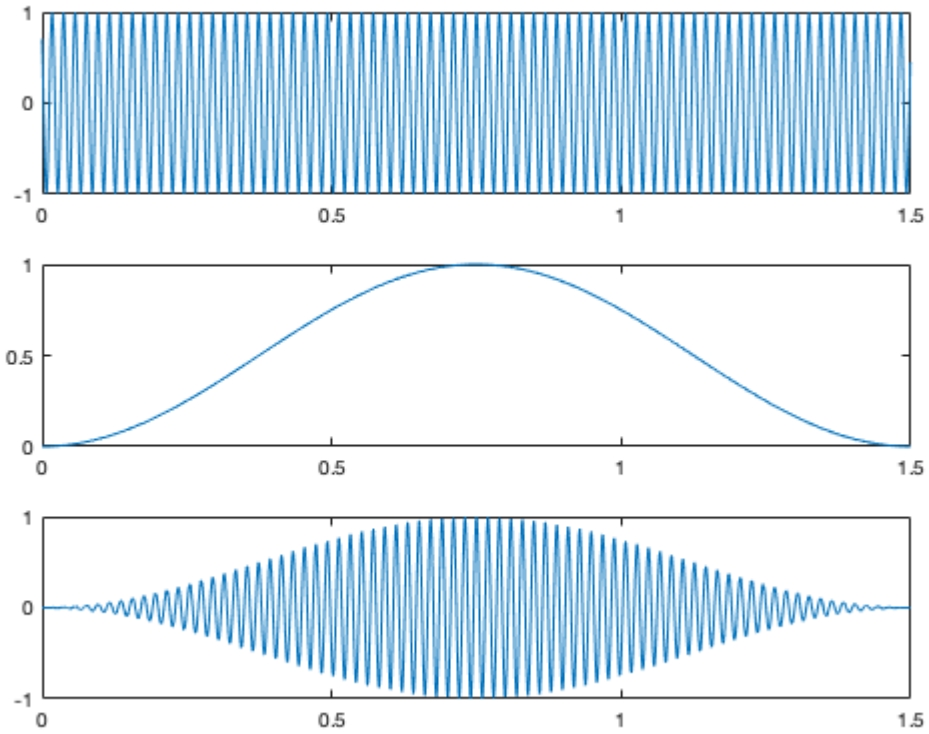
```
cl
```

2.3.1

Using the script, window x(t) with a Hanning window.

```
type hann

fs = 1000;
tt = 0:1/fs:(1500-1)/fs;
xx = cos(2*pi*50.5*tt + 0.25*pi);
hann = hann(length(xx));
xx_hann = xx.*hann;

subplot(3,1,1), plot(tt, xx);
subplot(3,1,2), plot(tt, hann);
subplot(3,1,3), plot(tt, xx_hann);
```

```
function hh = hann(L)
%HANN generates a length L Hann window, hh
    hh = .5*(1 - cos(2*pi*(1:L)/(L+1)));
end
```

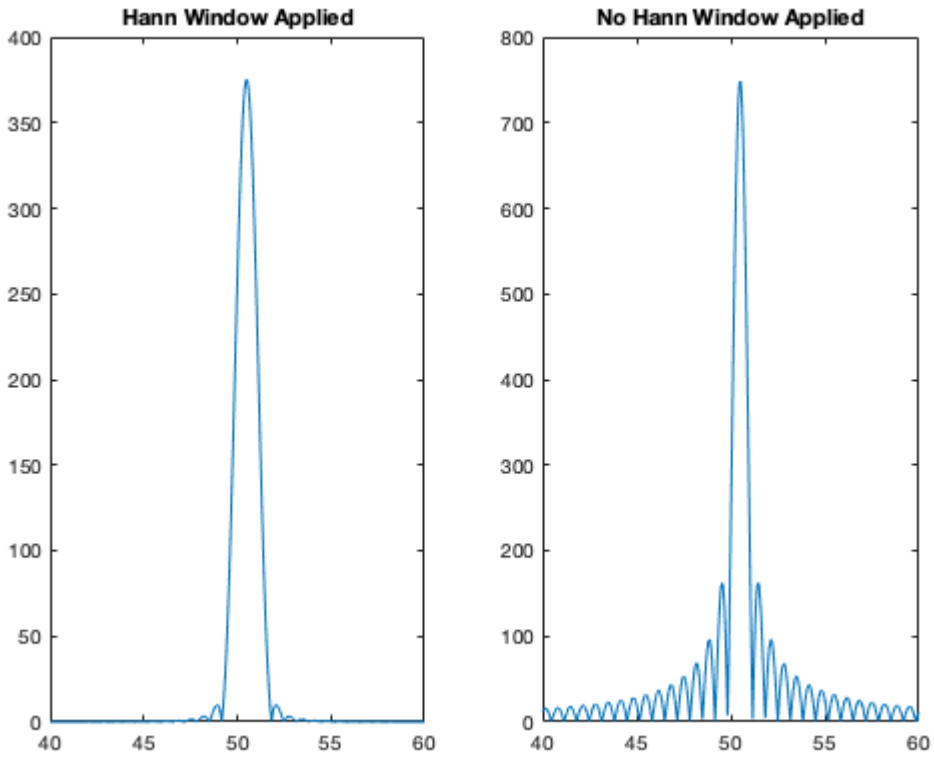


2.3.2

Padding with an appropriately large number of zeros, plot the magnitude spectrum of x(t) after windowing and zero padding. Make a side-by-side plot of the magnitude spectra, zoomed in at the peak value, for X with windowing and X without windowing.

TODO

```
xx_padded = [xx zeros(1,50000)];
xx_hann_padded = [xx_hann zeros(1,50000)];

XX = fftshift(fft(xx_padded));
XX_hann = fftshift(fft(xx_hann_padded));

ww = -pi:(2*pi/length(XX)):(pi-1/length(XX));
ff = fs * ww / (2 * pi);

subplot(1,2,1), plot(ff, abs(XX_hann)), title("Hann Window Applied"), axis([40,60,0,400]);
subplot(1,2,2), plot(ff, abs(XX)), title("No Hann Window Applied"), axis([40,60,0,800]);
```



Question 8: What effect does windowing have on the FFT?

The windowing makes the FFT have a smaller magnitude. This happens as the Hann window reduces the magnitude of all points on the signal. The main lobe of the FFT is also larger, and the other side lobes are greatly reduced. The hann window version of the FFT seems much more preferable, as we do not have the oscillations seen in the version without the hann window.

Question 9: How does the magnitude of the maximum peak change with windowing?

The magnitude of the Hann window is eqaul to 0.5, it seems that the maximum peak of the FFT was linearly affected by this Hann window magnitude. By that, I mean that the magnitude of the peak was halved.

Question 10: Is it higher or lower than that of the maximum peak without windowing?

As I touched on previously, the magnitude of the maximum peak in the version with the Hann window is lower.

Question 11: Has the width changed at all?

Yes, as I have previosuly discussed, the width of the maximum peak is much larger when compared to the version without the hann window.

2.4 - Determining Average Heart Rate Using FFT

```
cl;
```

2.4.1

Load the ECG data. There will be four arrays - am101, am105, am107, and cm110 - each containing an hour of ECG data.

```
load('heartratelabdata.mat');
```

2.4.2

In this step, you will program a function or write a MATLAB script to read each signal and determine the average heart rate. Since each signal is one hour long, you will need to divide the signal into subsamples (or 'windows') of equal width. It is suggested you start by using a window about 2 minutes long. DO THE MATH in order to condense it to 2 minutes with even windows.

2.4.3

In order to plot the average heart rate over time, you will need to iterate through the windows and take the FFT for each window. It is suggested that there be an overlap between each window as the function iterates. This overlap is suggested to start to be about 10 seconds.

2.4.4

Take the absolute value of the FFT of the sample and identify the peak closest to bin zero, but not at zero. You are looking for the first harmonic. Your algorithm should automatically identify the first harmonic each time the function iterates, because it is this peak that stores the frequency information of the heart rate.

2.4.5

Record the index/bin of the peak you identified. Convert this bin number to frequency in Hz, or beats per second. Note that this may be near one beat per second. Multiply this frequency by 60 to obtain the heart rate, in beats per minute, for each window.

2.4.6

Store these average heart rates into a vector, and then produce a plot showing the change in the average heart rate over time.

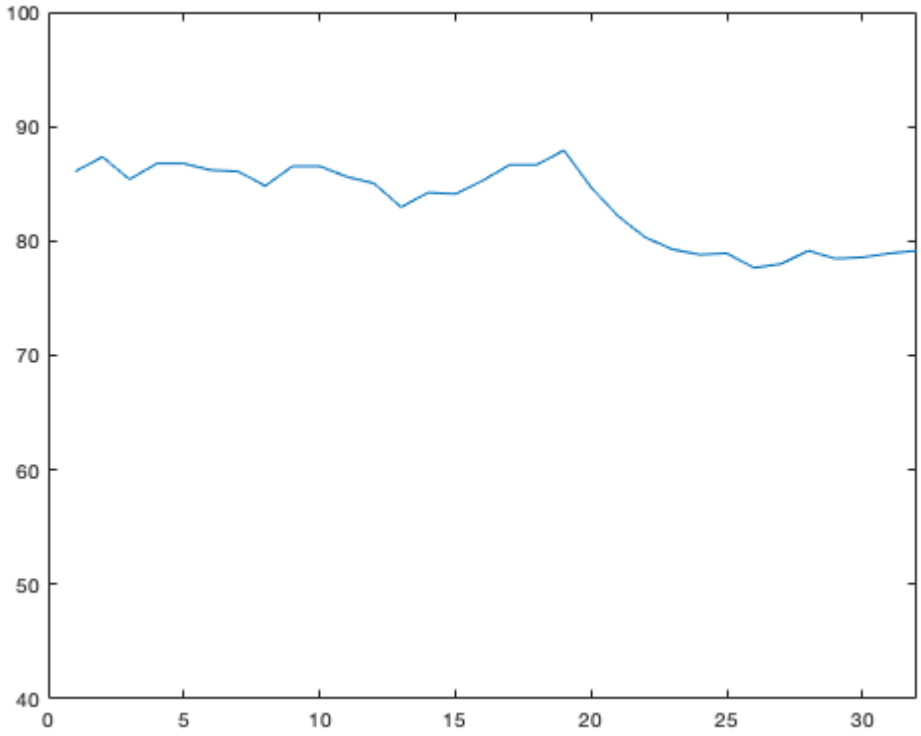
2.4.8

Use that function to produce the four plots, one for each signal.

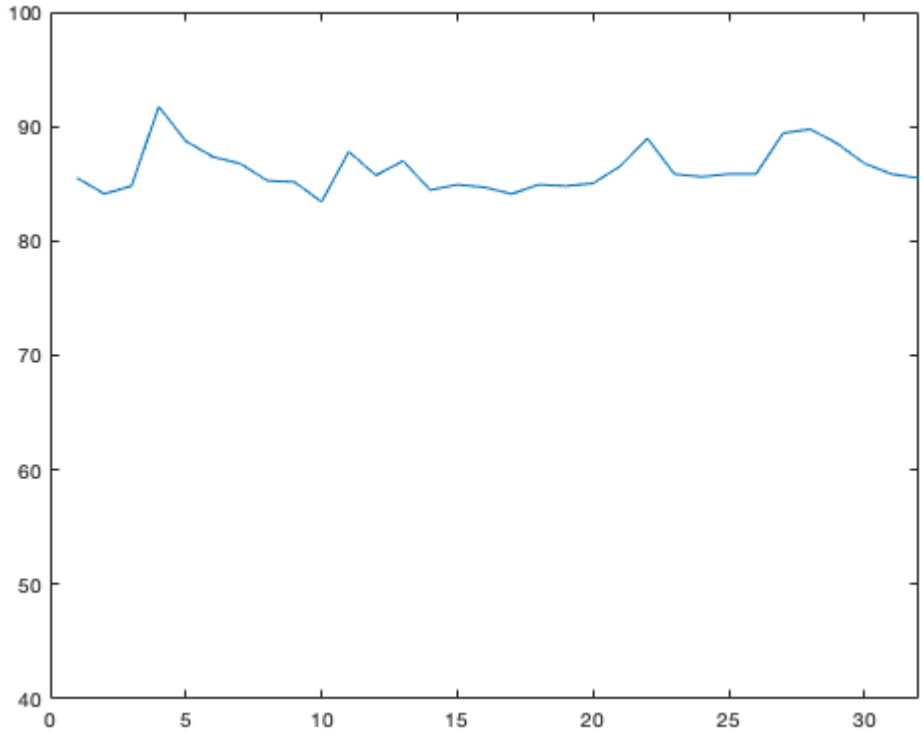
```
type analyzeHeartRate

am101_bpms = analyzeHeartRate(am101);
plot(am101_bpms), axis([0 length(am101_bpms) 40 100]);
```

```
function bpms = analyzeHeartRate(ecg)
%ANALYZEHEARTRATE extracts the BPM from an ECG in windows

    signal_length_sec = 1 * 60 * 60;
    window_length_sec = 60 * 2;
    overlap_length_sec = 10;
```

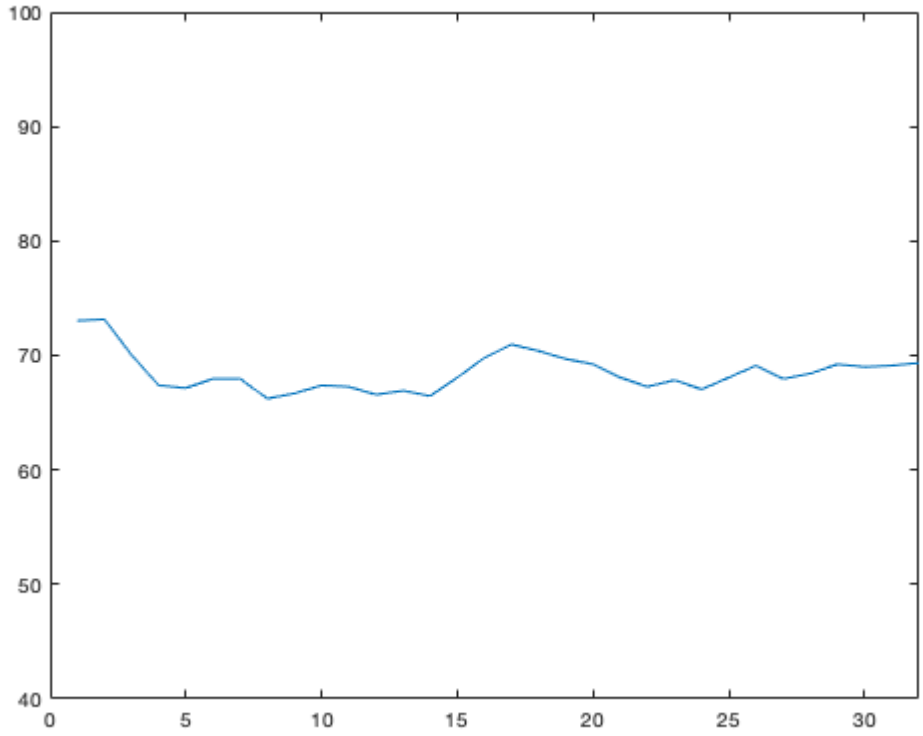
```
    fs = round(length(ecg)/signal_length_sec);

    window_length_samp = window_length_sec*fs;
    overlap_length_samp = overlap_length_sec*fs;

    start_samp = 1;
    end_samp = start_samp + window_length_samp - 1;

    bpms = [];
    while end_samp < length(ecg)
        xx = ecg(start_samp : end_samp);
        h = hann(length(xx));
        xx = xx .* h;
        xx = [xx zeros(1,50000)];
        XX = abs(fftshift(fft(xx)));

        ww = -pi:(2*pi/length(XX)):(pi-1/length(XX));
        ff = fs * ww / (2 * pi);
        hr_range = find(ff > 1 & ff < 2);
        ff = ff(hr_range);
        XX = XX(hr_range);

        bpm = ff(XX == max(XX)) * 60;
        bpms = [bpms bpm];

        start_samp = end_samp - overlap_length_samp;
        end_samp = start_samp + window_length_samp - 1;
    end
end
```



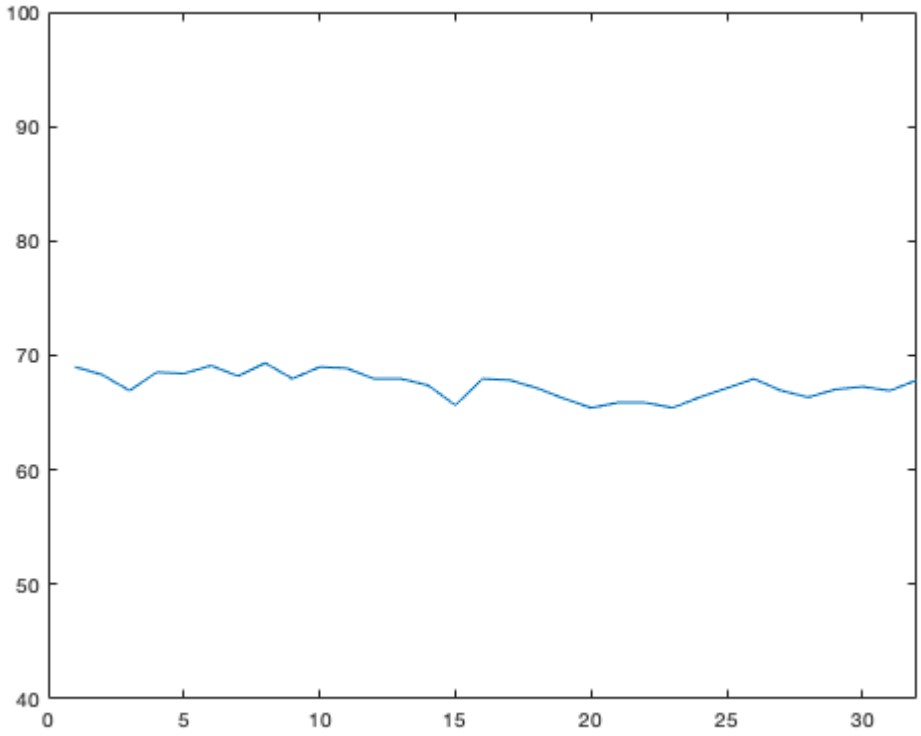

```
am105_bpms = analyzeHeartRate(am105);
plot(am105_bpms), axis([0 length(am105_bpms) 40 100]);
```




```
am107_bpms = analyzeHeartRate(am107);
plot(am107_bpms), axis([0 length(am107_bpms) 40 100]);
```




```
cm110_bpms = analyzeHeartRate(cm110);
plot(cm110_bpms), axis([0 length(cm110_bpms) 40 100]);
```



Published with MATLAB® R2019a