Progress Report ECG Feature Extraction

Moritz Konarski

January 29, 2021

Contents

1 Introduction

This is an overview of my progress this far. I attempted to perform ECG feature extraction with the help of the fast Fourier Transform (FFT). I chose to use the Julia programming language, a free, open source, fast, and scientific programming language.

As of now I have not been successful in extracting features using FFT. I can perform the FFT, but how to get from there to feature recognition is not clear to me. I have not found a paper in which the process was explained in a way that I could replicate it. I am still working on this issue. Below you can find my current progress.

The ECG data that I used is an MITBIH ECG file (number 100; downloaded and converted to csv using WFDB).

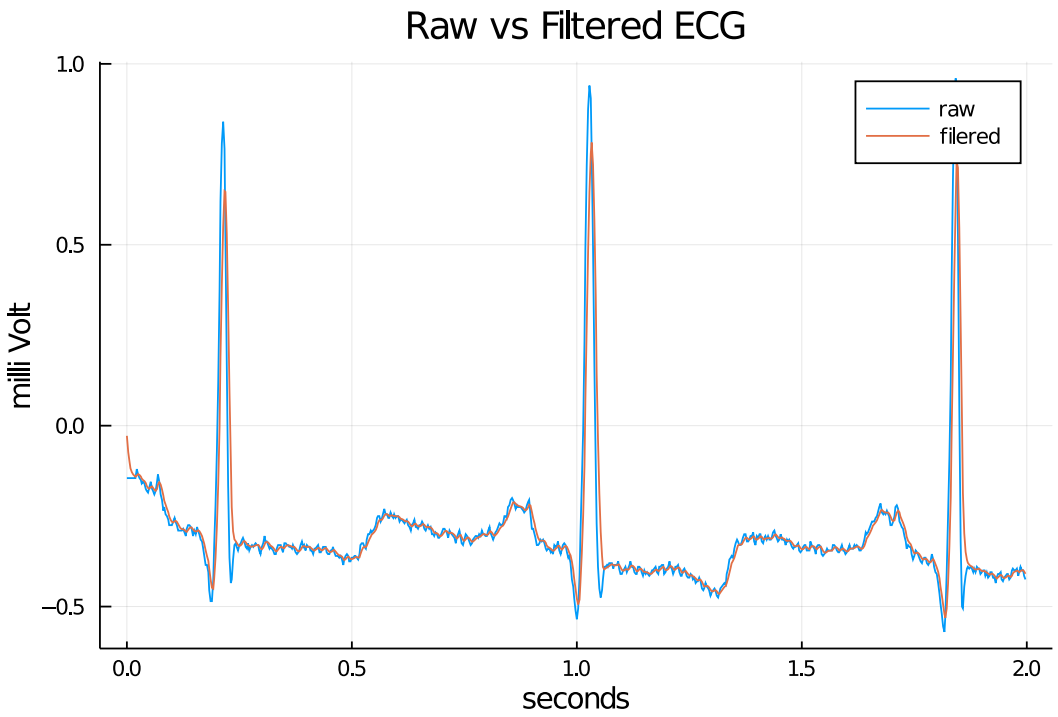
2 Hand-coded FFT Processing

In this section I perform most parts of the analysis by coding it myself instead of using a library to do all the work. This was my first approach. The code below is annotated to improve clarity. In case of questions please ask me.

[1]:
```
# plotting library
using Plots; gr()
# library for FFT, signal processing, data importing, matrices
using FFTW, DSP, CSV, Tables

# read in the ecg file as a matrix (CSV and Tables libraries)
ecg_matrix = CSV.File("../mitbih/100.mitbih") |> Tables.matrix

# set the sampling frequency (based on the ECG data)
fs = 360      # 360 samples per second
```

```
# set the used segment to the first 2 seconds
N  = 2 * fs  # equivalent to 2 seconds

# create a smaller section of the ECG to use
ecg = ecg_matrix[1:N, 1:2]

# create a simple digital filter (DSP library)
digital_filter = digitalfilter(Lowpass(0.15), Butterworth(1))

# create a filtered version of the ECG
ecg_filtered = copy(ecg)
ecg_filtered[:,2] = filt(digital_filter, ecg[:,2])

# plot the raw and filtered ECGs for comparison (Plots library)
plot(ecg[:,1], ecg[:,2], title="Raw vs Filtered ECG", label="raw")
plot!(ecg_filtered[:,1], ecg_filtered[:,2], xlabel="seconds",␣
 ↪ylabel="milli Volt", label="filered")
```

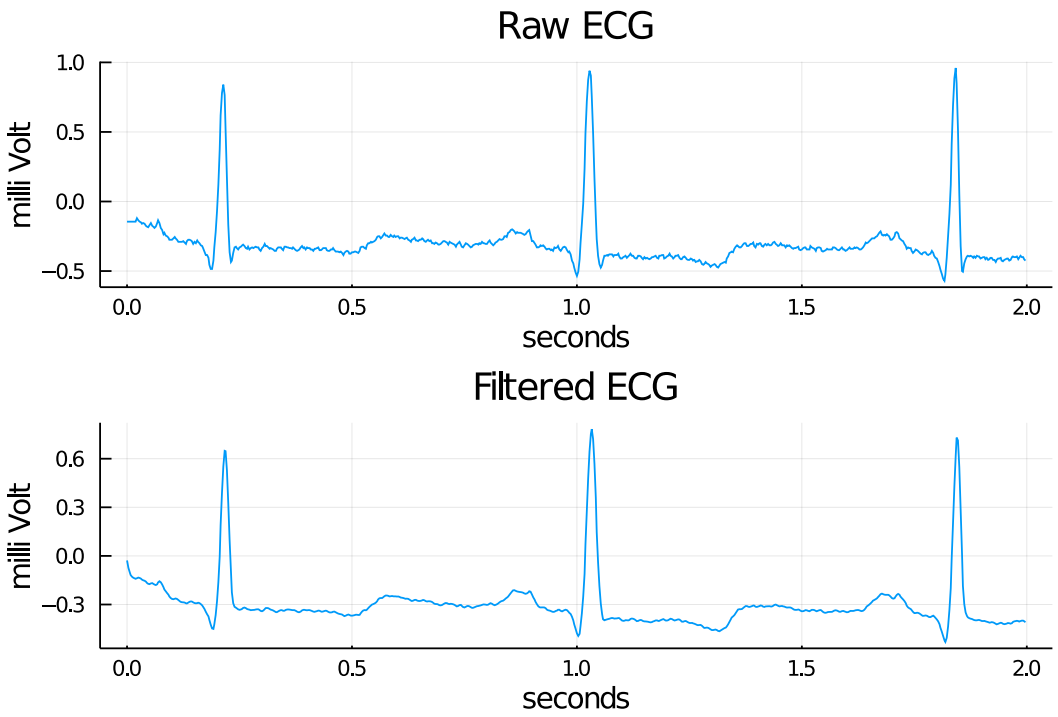
[1]:



[2]:
```
# create a side-by-side plot of the raw and filtered ECG (Plots␣
 ↪library)
p1 = plot(ecg[:,1], ecg[:,2], title="Raw ECG", xlabel="seconds",␣
 ↪ylabel="milli Volt")
p2 = plot(ecg_filtered[:,1], ecg_filtered[:,2], title="Filtered ECG",␣
 ↪xlabel="seconds", ylabel="milli Volt")
```

```
plot(p1, p2, layout=(2,1), label="")
```

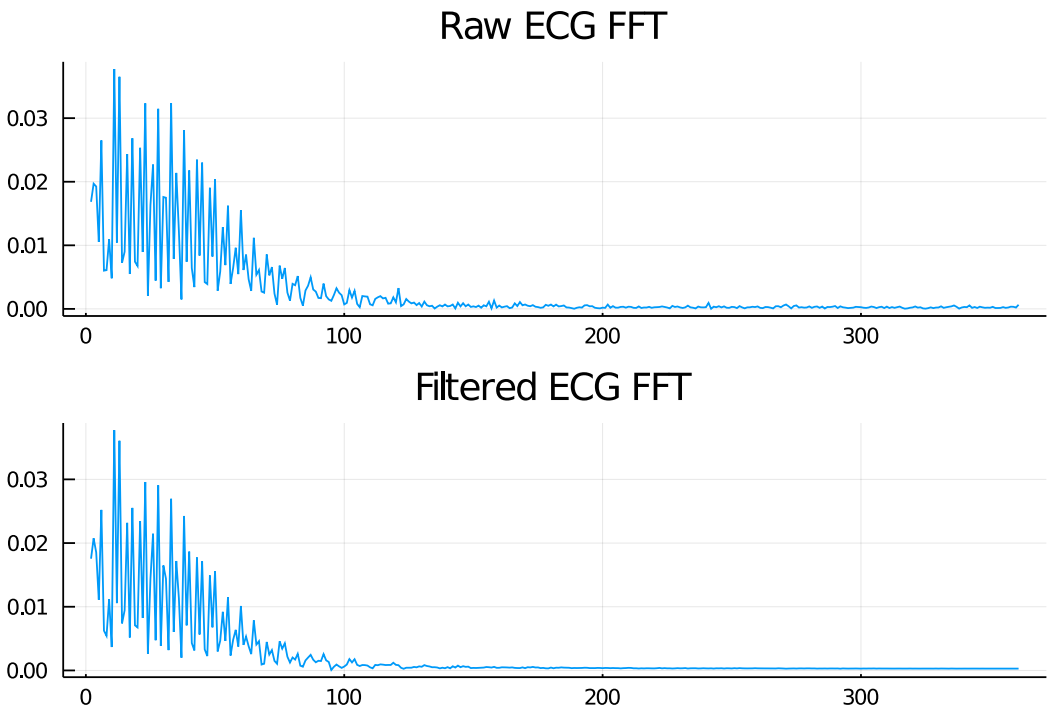
[2]:



[3]:

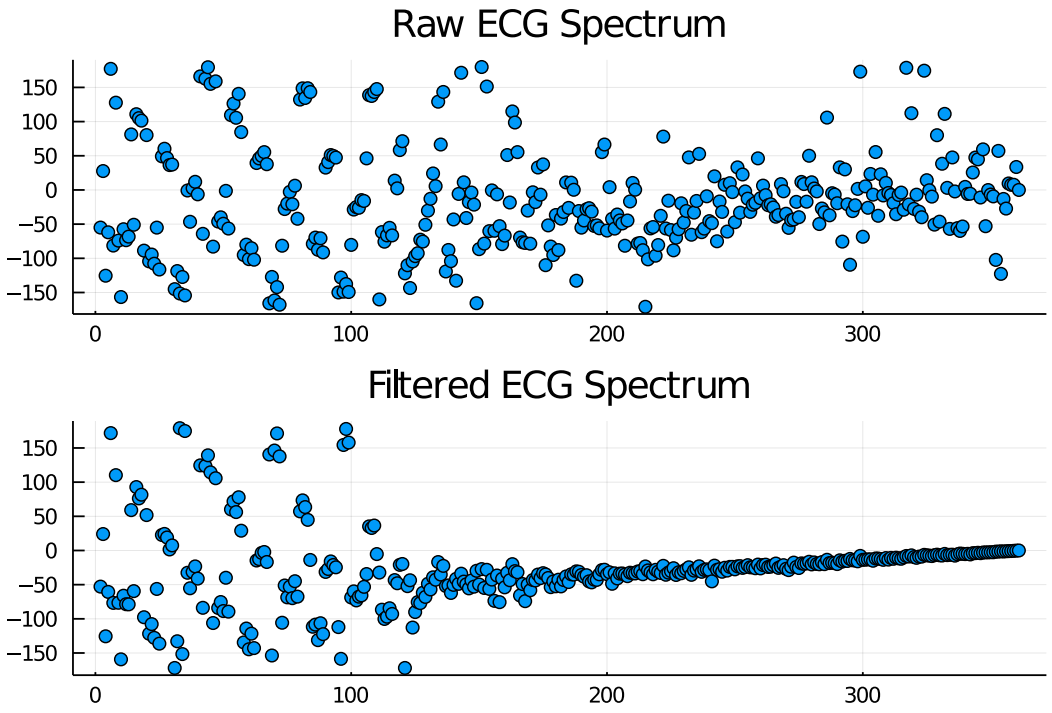
```
frequencies = 2:div(N,2)+1

# create a FFT of the raw ECG signal (FFTW library)
F  = 1 / N * fft(ecg[:,2])[frequencies]

# create a FFT of the filtered ECG signal (FFTW library)
F_filtered = 1 / N * fft(ecg_filtered[:,2])[frequencies]

# plot the FFTs

p3 = plot(frequencies, abs.(F), title="Raw ECG FFT")
p4 = plot(frequencies, abs.(F_filtered), title="Filtered ECG FFT")
plot(p3, p4, layout=(2,1), label="")
```

[3]:



[4]:
```
# plotting phase spectrums
phase_list = [atan(imag(i), real(i)) for i in F] * 180 / pi
p5 = plot(frequencies, phase_list, seriestype = :scatter, label="",␣
 ↪title="Raw ECG Spectrum")
phase_list_filtered = [atan(imag(i), real(i)) for i in F_filtered] *␣
 ↪180 / pi
p6 = plot(frequencies, phase_list_filtered, seriestype = :scatter,␣
 ↪label="", title="Filtered ECG Spectrum")
plot(p5, p6, layout=(2,1), label="")
```

[4]:

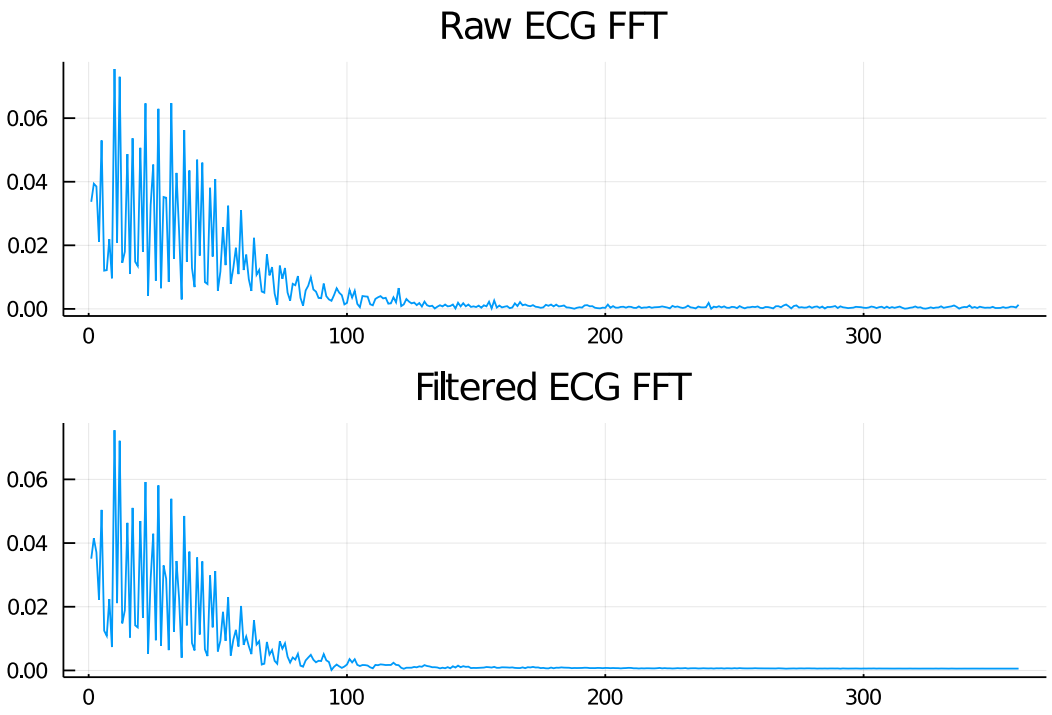


3 FFT Processing using a library

In this section I perform the same operations as before, but I use a library to do most of the work for me.

This code works as an extension of the code above and I am reusing the exact same values/arrays for the ecg.

[5]:
```
# importing the new library
using FourierAnalysis

# computing fouriers spectra with library and plotting them
spec          = spectra(ecg[:,2], fs, N; tapering=rectangular, func=√)
spec_filtered = spectra(ecg_filtered[:,2], fs, N; tapering=rectangular,␣
 ↪func=√)

p7 = plot(abs.(spec.y[brange(N),:]), title="Raw ECG FFT")
p8 = plot(abs.(spec_filtered.y[brange(N),:]), title="Filtered ECG FFT")

plot(p7, p8, layout=(2,1), label="")
```
[5]:



Below I show that both approaches yield the same ECG (given some minor error probably caused by different numeric approaches).

[15]:
```
# illustration that the two approaches are identical
print("Max Difference raw FFTs: ")
println(maximum(abs.(F) - abs.(spec.y[brange(N),:])))

print("Max Difference filtered FFTs: ")
println(maximum(abs.(F_filtered) - abs.(spec_filtered.y[brange(N),:])))
```

```
Max Difference raw FFTs: -1.5556475161214077e-5
Max Difference filtered FFTs: -7.289324365218629e-5
```

This is as far as I progressed this far. I will continue to work on this. Any feedback is greatly appreatiated.