Assignment 2
Digital Signal Processing
FIR Filters

Scott Watson, Rami Ghannam

Form groups of 4-6 people.

The task of this assignment is to filter an ECG with FIR filters and to detect the heartbeats. In contrast to the FFT assignment we write filter code which can be used for *real-time processing*. This means that the FIR filter needs to be implemented as a class with the help of delay lines and the impulse response is truncated. The class must accept single samples at a time; passing a whole array to be filtered is not real-time. Make sure you understand why.

ECG filtering

You are provided with two ECGs (please download files from Moodle according to your team number): a noise free one with as few artefacts as possible - recorded lying down and placing the electrodes on hip/shoulder. And a 2nd noisier ECG – recorded while standing with electrodes on the ankles and wrists.

1. Create a function which calculates and returns the FIR filter coefficients *analytically* (= using sinc functions) for a *combined* highpass and bandstop filter.

The function should automatically decide how many coefficients are required. The function arguments should be a) the sampling rate and b) the cutoff frequencies. Decide which cutoff frequencies are needed and provide explanations by referring to the spectra and/or fundamental frequencies of the ECG. **[25%]**

2. Create an *efficient* Python FIR filter class which implements an FIR filter and has a method of the form value dofilter(self,value) where both the value argument and return value are *scalars* and not vectors (!) so that it can be used in a real-time system. The constructor of the class takes the coefficients as its input:

```
class FIRfilter:
def __init__(self,_coefficients):
# your code here
def dofilter(self,v):
# your code here
return result
```

Filter both ECG recordings with the above FIR filter class using the coefficients from 1. Simulate real-time processing by feeding the ECGs sample by sample into your FIR filter class. Make sure that the ECGs look intact and that they are not distorted (PQRST intact). Provide appropriate plots in a vector-graphics format. **[25%]**

3. Use an adaptive LMS filter to filter out DC and 50Hz by providing it with a 50Hz sine wave with DC as reference. Note that both the amplitudes for the 50Hz and DC references scale with the learning rate. Make appropriate choices for the amplitudes and the learning rate so that both DC and 50Hz are removed.

Add an adaptive LMS filter method to your FIR filter class (from 2.) and name it: "doFilterAdaptive(self,signal,noise,learningRate)" which returns the cleaned up ECG. As before also this function must receive only scalars (i.e. sample by sample) and return a scalar. Plot and compare the result from the adaptive filter and that from the FIR filter design. **[25%]**

4. ECG heartbeat detection: The task is to detect R-peaks in the *noisy* ECG recording. Use the FIR filter from 2. as a matched filter and use an R-peak as a template from the *noise-free* ECG. Plot the momentary heart rate (i.e. inverse intervals between R-peaks) against time. **[25%]**

Every report must be based on different ECG recordings. Please keep it short but it should make clear what you have done and why you have done it. Include the complete Python code as well as plots of the ECGs (timedomain) and their frequency representation (with proper labels). If necessary annotate the plots with a vector-graphics editor such as InkScape, Corel Draw or Illustrator by labelling the ECG peaks and remember to use vector based image formats, for example EPS, SVG, PDF or EMF and not pixel based formats for the report. These figures need to be in the report at the correct place and not attached separately. Also, show zoomed in ECG traces of one heartbeat so that it is possible to identify the different parts of a single heartbeat and that it's possible to check if it's still intact.

No high level Python scipy.signal functions except of FFT/IFFT and the window functions are allowed. Any use of "filtfilt", "lfilter", "firwin", "conv", "correl" and any a-causal processing (i.e. data array in and array out) commands will result in zero or very low marks. As before submit a zip file containing all files and test the zip before submission by unzipping it and then running python from the command-line in a terminal (not spyder or similar). Also check that all plots are generated when running the script from the command line which requires plt.show() for the graphs to appear. See moodle for the exact filename conventions.

Deadline is 18th November, 3pm.