Assignment 2, Digital Signal processing: FIR filters

Bernd Porr

Semester 1, 2017

The task of this assignment is to filter an ECG with FIR filters. Before you record from yourself please read the info-sheet and sign the consent form.

1 ECG filtering

Form groups of 2 students. Record at least 2 different ECGs so that every member has his or her personal ECG. There are different amplifiers with different gains and converter boards available. When you record the ECG make sure that you take a note of the gain, bits, etc. Make sure the "raw" button is pressed while recording. If you record with a one channel amplifier record Einthoven II (google it what it means). Upload the files onto the ECG database on moodle or e-mail it to yourself. Give the file a name which anonymises the ECG. Use only numbers, characters and the underscore for your filename.

Every student has to work on a different ECG. If your initial recording has been too noisy just record another one. You can minimise artefacts by lying down and/or placing the electrodes on the shoulders / hips instead of wrists/ankles. In Python load the file with the command *numpy.loadtxt('myecg.dat'*. You'll get a numpy matrix with time-stamps in ms in the first column and 3 columns of ecg with raw unsigned integer numbers. Choose the channel with the highest *amplitude*. Also note that the ECG devices will be around all Oct / Nov so there is plenty of time to record your ECG. You can use one of the example ECGs from moodle to kick start your work.

No high level Python functions from scipy except of fft and the window functions are allowed in this section.

1. Display the original ECG in mV (!!!) against msec. Show the whole recording *and one* heartbeat to identify the so called PQRST waves (google it). Remember to take into account the gain of the amplifier. The A/D converter has either 12bit or 24bit resolution and has an input range from $-4.096V \ldots +4.096V$ or $-1.325V \ldots 1.325$ respectively. Do *not* use the pre-defined python functions to convert it. Rather develop your own formula to convert the data. Plot also the frequency spectrum of the ECG ($abs(F(\omega))$) (omitting the mirror). Discuss what is signal and what is noise. In particular, identify the unwanted 50Hz contamination and the DC drift of the ECG. Label the ECG properly with the help of a vector based drawing program such as Inkscape. [10%]

2. Create a Python FIR filter class which implements an FIR filter which has a method of the form `value FIR(value)` where both the value argument and return value are *scalars* and not vectors so that it could be used in a realtime system. The constructor of the class takes the coefficients as its input:

```
class FIR_filter:
def __init__(self,_coefficients):
# your code here
```

```
def filter(self,v):
# your code here
    return result
```

[20%]

3. Calculate the coefficients of an FIR filter analytially (by using the sinc formula from the lecture) for a 50Hz notch filter and filter the ECG with it. [10%].

4. Calculate the coefficients of an FIR filter to remove the baseline shift and 50Hz of the ECG numerically with the help of the inverse Fast Fourier Transform. Filter the ECG and make sure that the ECG is not distorted. [10%].

2 ECG heartrate detection

The task is to detect the *momentary* heart rate $r(t)$ over a longer period of time. For example, after exercise you should see a slow decay of the heart rate to the baseline of perhaps 60 beats per minute. It is not the average heart rate but the frequency derived from the times between adjacent heartbeats.

Record a separate long ECG over a couple of minutes from at least two different people. Ask the person to breathe in quite deeply which will change the heart rate or anything which changes the heart rate for example sending the person down to level 1 and back.

High level filter scipy filter commands are allowed to speed up the filtering operation.

1. Design a matched filter which detects the individual heartbeats in the ECG. Think of the optimum length of the filter and give a reason. Optimise the detection by pre-filtering both the ECG and the template. Remember the detection works best when both the signal and the templates are completely DC free. [20%]

2. Implement a small program which uses the output of the matched filter to calculate the *momentary* heart rate $r(t)$ over time (not the average!) by measuring the intervals between the detected hearbeats over the whole period of the ECG. Detect the heartbeats by employing a threshold (adaptive, if required). Square the output of the detector to improve the signal to noise ratio or try other methods. Generally use any heuristics to weed our wrong detection, for example that a heartbeat is usually below 200bpm and above 30bpm. [30%]

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 Python code as well as plots of the ECGs (timedomain) and their frequency representation (with proper lables). If necessary enhance the plots with InkScape and remember to export plots in vector based image formats, for example EPS, SVG, PDF or EMF and not pixel based formats. Also, show zoomed in ECG traces of one heartbeat so that it is possible to identify the different parts of a single heartbeat (see Fig. 1) and that it's possible to check if it's still intact. Hand in your report at the teaching office (Room 620 James Watt Building South). Deadline is 27th November 3pm.

3 Bonus work (instead of assignment 3 = double the marks)

3.1 Option 1: FIR module in Python/C++/SWIG

Write a class which does both the filtering and the FIR filter design for lowpass, highpass, bandpass and bandstop. Optimise the actual FIR filter routine by implemeting it in C or C++ and create an FIR package/module. Publish this class on github including a demo and howto. If successful this will give you guaranteed 100%. It can be done under any operating system but its easiest under Linux and nearly impossible under Windows (but a nice chanllenge!).

3.2 Option 2: Realtime demo

Write a realtime demo which shows that your FIR filter is able filter the ECG in realime. The USB-DUX boards use a library called comedi which has a Python wrapper. Then on the other hand matplotlib can animate realtime data. This needs to be done under Linux as the USB-DUX boards are made for Linux. Alternatively that can be also done with the Attys (www.attys.tech) which has also python libraries but needs to be adjusted.

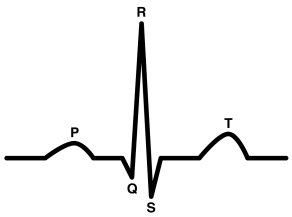

Figure 1: A normal ECG with P,Q,R,S,T waves (taken from Wikipedia)