BBT.HTI.501 Processing of Biosignals

Assignment 1: Using a feature from the power spectrum of the EEG to assess depth of anesthesia

Return your answers to Moodle in PDF format. Check the correct deadline from course instructions! Attach your Matlab codes to the report as an appendix or return them as separate m-files. Remember to check the assessment criteria and general instructions for reporting.

The aim is to implement a measure for depth of anesthesia using a feature extracted from the EEG, the complexity of spectrum. As measure of complexity we use here the so-called Spectral Entropy. The idea is that you make a function that can be used in a patient monitor so that the output is high when the patient is awake, and low for deep levels of anesthesia. For comparison of the performance of the EEG feature, we have the output of a commercial device measuring depth of anesthesia - the BIS (Bispectral index monitor, http://www.covidien.com/rms/brands/bis), which generates an index value between 100 and 0.
In this exercise, your task is to implement the basic version of the algorithm as described in SpectralEntropyDOA.pdf and see how it behaves.
In this exercise, we have patient data recorded during sevoflurane anesthesia (sevoflurane is a short-acting anesthetic gas). The file contains one channel of EEG data and some additional signals.

Load the file `case_1.mat`

You should see the following variables:
`eeg_ch`: array containing eeg channel amplitude at sample instances (in microvolts)
`eeg_time`: time at which corresponding values of eeg_ch were sampled (in seconds)
`before`: has value 1 before start of anesthesia ("awake" situation), otherwise 0
`after`: value 1 after ending of anesthetics delivery ("waking up" situation), otherwise 0
`annot_time`: time (in seconds) at which the before and after values are assessed
`bis_ch`: values from the BIS monitor (dimensionless number in the range 0-100)
`bis_time`: time instances (in seconds) at which BIS values were recorded.

At t=0 the recordings are started, all signals are synchronized to the same time.

The instructions for the implementation and questions to be answered in the report:

1) Make a plot combining the EEG signal and `before` and `after` annotations signals in one graph (use for example `hold on` in Matlab to put the different signals in one graph). It may be a good idea to multiply the `before` and `after` signals with e.g., 100 to make them stand out from the EEG. Can you infer visually from the EEG where the awake part of the EEG is and where the anesthesia period is?

2) What is the sampling frequency of the EEG signal? Make a variable `fs_eeg` and assign this value to it.

Before processing EEG we usually pre-filter the signal to remove very low frequencies that are typically associated with movement artefacts

3) Design a highpass filter with a cut-off frequency of 0.5 Hz. Would you use a FIR or IIR for this type of problem, why? Plot the frequency response of the filter to check whether it really does what you want. Check the filter's phase shift, is it linear? In offline calculations phase shifts can be corrected using the `filtfilt.m` function. You can use it in this situation as well. Name the resulting signal, for example, `eeg_f`

4) Plot both the original and filtered eeg, comment on what you see, in which parts of the recording does this filter especially have effect?

5) We are now going to calculate the spectral entropy of the EEG (similar to as described in SpectralEntropyDOA.pdf). First, we calculate the PSD in a window and then we calculate the Entropy of the PSD. We do that over windows moving through the data with some overlap. In this example, we will use 5-second windows with 50% overlap. The function `entropy_eeg.m` implements the moving of the window and calculation of PSD's. It then calls a function `spectral_entropy.m` that calculates the entropy. The code of the body of the function `spectral_entropy.m` you are supposed to write yourself.

Study the `entropy_eeg.m` function and try to understand how it works. Study the equations 3-5 in SpectralEntropyDOA.pdf and implement them in the function `spectral_entropy.m`

6) Run the function `entropy_eeg.m` to obtain the spectral entropy at different time instances throughout the EEG data (for example, use `[H,H_t] = entropy_eeg(eeg_f,fs_eeg)`,this puts entropy values in variable H at time instances H_t)

7) For monitor use, usability tests have shown it is easier to quickly interpret a number between 0 and 100 instead of a number between 0 and 1 (e.g., "56" instead "0.56"), thus we scale the entropy to that range by just multiplying it with 100. Plot `100*H` together with the `bis_ch` in one picture, and also add the `before` and `after` signals. What parts are similar, what parts are different? Can you guess why these differences occur? How could the spectral entropy signal be improved (think of pre- as well as post-processing possibilities)?