Task 3 – Signal Equalizer

Introduction: Signal equalizer is a basic tool in the music and speech industry. It also serves in several biomedical applications like hearing aid abnormalities detection.

Description: Develop a desktop application that can open a signal and allow the user to change the magnitude of some frequency components via sliders then reconstruct back the signal.

- The application should be able to work in different modes:
  1. **Uniform Range Mode:** where the total frequency range of the input signal is divided uniformly into 10 equal ranges of frequencies, each is controlled by one slider in the UI.
     - To validate your work, each group should prepare a synthetic signal file. The signal is an artificial signal that you should prepare, composed of a summation of several pure single frequencies across all the frequency range. This should help in tracking what happens to each frequency when an equalizer action is taken and whether the program behaves correctly or not.
  2. **Musical Instruments Mode:** where each slider can control the magnitude of a specific musical instrument in the input music signal that is a mixture of at least four different musical instruments.
  3. **Animal Sounds Mode:** where each slider can control the magnitude of a specific animal sound in a mixture of at least four animal sounds.
  4. **ECG Abnormalities Mode:** You need to search for and collect 4 ECG signals, one normal and each of the other three has a specific type of arrythmia. Then, each slider can control the magnitude of the arrythmia component in the input signal.

  Note: For modes 2 to 4, each slider does not necessarily correspond to one continuous range of frequencies. Each slider can easily correspond to multiple frequency ranges/windows. Think carefully how your code will handle that.
- The user can switch between the modes easily (probably through an option menu or combobox). The UI is not expected to change dramatically among the different modes. The main expected change in UI when changing from one mode to the other is the change in the sliders captions and maybe the number of sliders.
- For all modes, the graph that displays the Fourier transform of the signal should have the flexibility to show the frequency range in either a linear scale or the Audiogram scale (Please, make your research for what Audiogram is). The user can switch between the two scales via some UI without interrupting or resetting any functionality.
- Beside the equalizer sliders, the UI should contain:
  - Two linked cine signal viewers, one for the input and one for the output signals, with a full set of functionality panel (play/stop/pause/speed-control/zoom/pan/reset) that respects all the boundary conditions. The two viewers should allow the signals to run in time in a synchronous way (i.e. the two viewers have to be exactly linked. They should always show the same exact time-part of the signal if the user scroll or zoom on any one of them).
    - Cine viewer: A viewer that allows the signals to run in time.
    - Linked viewers: All viewers should show the same exact view (in time and magnitude directions) of the displayed signal.
  - Two spectrograms, one for the input and one for the output signals. Upon any change in any of the equalizer sliders, the output spectrogram should reflect the performed changes.
    - There should be an option to toggle show/hide of the spectrograms.

Code practice:

1. Same practices from Task 1 & 2 (i.e. proper variable and function names) will continue with task 3.
2. **Avoid code repetition**! If you notice similar lines in your code with only a few numbers or variables different, then these lines are very good candidates for a "function"! Code repetition is a very bad practice and will be penalized (each repetition will be penalized with -5%) from now on in the course tasks evaluation.