scc0251/5830— Prof. Moacir Ponti

Assignment 3 (Fourier Analysis)

Try to develop and implement yourself, without any help from your colleagues.

Fourier Analysis

In this assignment you have to implement functions in order to produce Fourier Analysis of signals.

Your program must have the following steps:

1. **Parameter input** (read from input):
   a) file name for the signal binary file,
   b) s parameter (to be used in windowing),
   c) c parameter (to be used in low pass filtering),
   d) show flag (0/1) which is 1 for showing the plots and 0 otherwise.
2. **Read** the signal file using `numpy.fromfile` using a `int32` type obtaining signal f;
3. **Apply Gaussian windowing** in the original signal obtaining signal g;
4. **Compute the FFT** of $f \rightarrow F$;
5. **Compute the FFT** of $g \rightarrow G$;
6. (show flag == 1) **Show plots**: signals f, g and respective normalized Fourier spectra;
7. **Apply low pass filter** in F by assigning zero to all frequencies equal or greater than a threshold, obtaining a filtered vector $\hat{F}$;
8. **Compute the IFFT** of $\hat{F} \rightarrow \hat{f}$;
9. (show flag == 1) **Show plots**: signals f and $\hat{f}$;
10. **Output**: max frequency of $|G|$ and its respective amplitude;
11. **Output**: max value of f and $\hat{f}$;

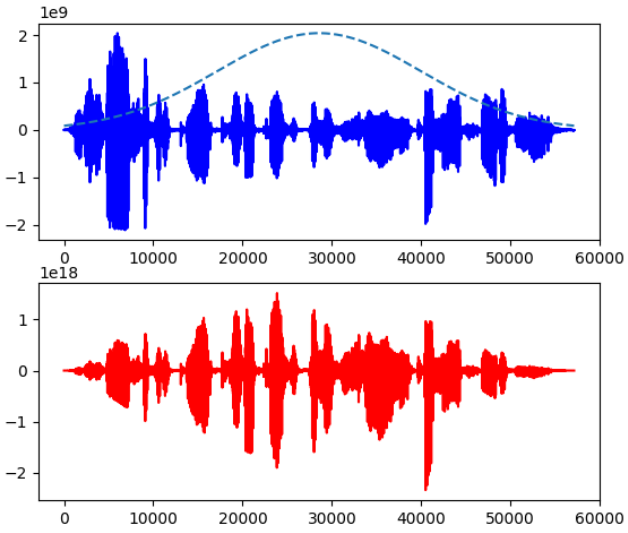
1 Details for implementation

1.1 Reading the signal

Using the `numpy.fromfile` with a `int32` type, we assume that the signal is a raw (without header) binary file containing a series of 32-bit integers.

1.2 Gaussian Windowing

Often applied in signals in order to obtain a "fade in" and "fade out" effect that allows for a better Frequency analysis. In the next Figure an example is shown: an input signal (blue) is multiplied by a Gaussian Function (represented with a dashed line), generating the red signal below.



Let n be the size of the signal (number of discrete observations). The window to be generated as an array must have the same size of the signal (n), and a standard deviation of:

$$\sigma = \frac{n}{s}$$

OBS: Note that the larger value of a Gaussian window is equal to 1. In the last Figure, the Gaussian window was normalized in order to be shown overlayed with the signal.

In order to perform the step (3), you have to multiply each point of the input signal with the Gaussian window, obtaining a new signal g.

Tip: you can use the `gaussian` *function available in* `scipy`*'s Signal Processing package.*

1.3 Fourier Spectrum and Normalized Fourier Spectrum (NFS)

Let F be the Fourier Transform of a signal f. Then its spectum is the absolute value of each value, i.e. $|F|$, and its normalized version (NFS) is

$$\frac{|F|}{2n}.$$

Plotting the NFS : in order to plot a NFS, we just show half the frequencies, i.e. the values up to $n/2$, since the Fourier Transform is symmetric.

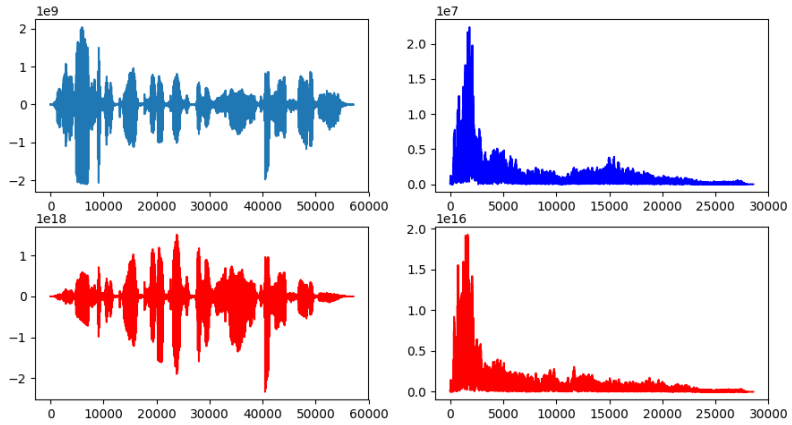
For the step (6), using `matplotlib` you can display the functions f, g and its NFS by using subplots as follows:

```
fig = plt.figure()
a1 = fig.add_subplot(rcp),
```

where `l` is the number of rows in the figure, `c` is the number of columns and `p` is the position of the plot `a1`. For example, in the following code we generated 2 subplots by using 2 rows and 1 column. In `a1` will be plotted the `x` vector, and in `a2` the `y` vector:

```
fig = plt.figure()
a1 = fig.add_subplot(211)
a2 = fig.add_subplot(212)
a1.plot(x)
a2.plot(y)
```

In your program you have to show the four plots, in the first row f and $\frac{|F|}{2n}$, and in the second row g and $\frac{|G|}{2n}$. Remember to plot only half the frequencies for the NFS. An example is given in the next Figure.



1.4 Low pass filtering

In order to apply a low pass filter on the signal, we just have to find a threshold. All frequencies equal or greater than this threshold will be cancelled (i.e. set to zero).

Our threshold will be:

$$T = c \cdot \arg\max(|G|)$$

where c is given as input, and $\arg\max(|G|)$ is the maximum frequency observed in the Fourier Spectrum of g.

After finding the threshold we will set to zero all frequencies of F that are greater or equal to T creating $\hat{F}$, i.e.

$$\hat{F}(\omega) = \begin{cases} F(\omega) & \text{if } \omega < T \\ 0 & \text{if } \omega \geq T \end{cases} \tag{1}$$

Note that we used G to find the threshold, but the filtering is performed using F.

Finally, we obtain the processed version of the input signal by obtaining the Inverse Fourier Transform of $\hat{F}$, generating $\hat{f}$.

2 Patterns for input/output

Example of input (filename, s, c, flag):

```
sound1.bin
5
2
0
```

Example of output (max frequency of input$f$, max frequency of windowed signal g, and max values for f and $\hat{f}$):

```
164\n
200\n
1222574080\n
394798650\n
```

Where `\n` is a line break. Note that the run.codes system will compare the output trying to find an exact match, so please follow the exact output pattern.

Attached to this pdf you can also find the `sound1.bin` file to run your own tests.

2.1 Writing the signal (optional)

You can use `numpy.tofile` in order to write the resulting (processed) signal. This way you can, for example, give as input some audio file and play its filtered version. Note that because of the filtering, it is possible that the amplitude values of the processed signal are lower. You can then re-normalize it to the maximum amplitude of the input signal in order to obtain the same amplitude range of the input file.

3 Instructions

- Individual homework assignment
- Deadline as found in the Run.Codes system

Submission Use the Run.Codes system for submission, including ONLY the `.py` file

It is **obligatory** to comment your code. As a header, use **name and USP number** (your grading will be discounted if this is missing).

Questions should be asked in the hours of the teaching assistant, or via email with the subject `[IP HA3] <question>`, especifying the question.

The source-codes will be tested for plagiarism. If any copy is detected, even in few parts of the code (such as a single function), the homework will be assigned with grade 0.