Demonstrative Tools used in the Visualization of DSP

Student Name: Zhang Ting
Student ID: U60112569
Date: 12/08/2019

I. Introduction

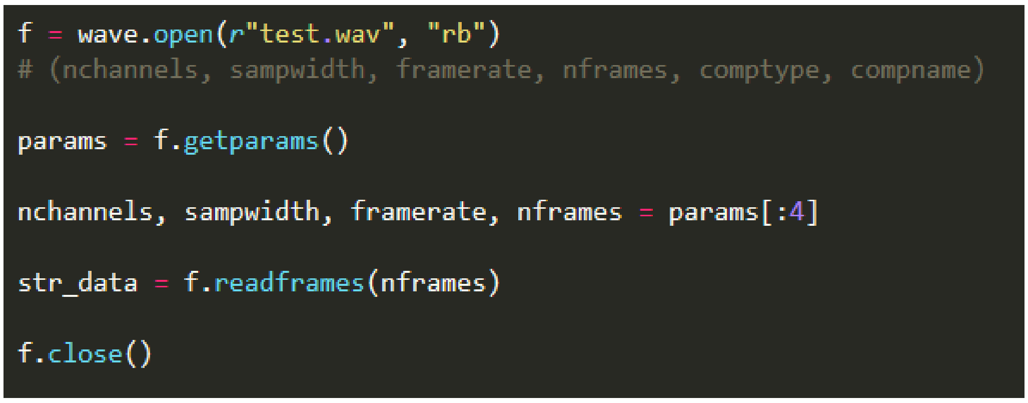
Digital signal processing (DSP) is the use of digital processing, such as by computers or more specialized digital signal processors, to perform a wide variety of signal processing operation. We can implement DSP on the different digital signal files like .wav, .dat and. json. However, it is not so easy to directly fetch the visualized information, so we need to make it more visualized to us like using painting and dot constructing. Here, we are going to use the python and the related module inside its library to make it.

II. Tools

Module --- **Wave**:

The wave module provides a convenient interface to the WAV sound format. It does not support compression/ decompression, but it does support mono/ stereo. It could use the read function to transfer the wav file beginning with RIFF standard to some kind of array with several parameters: (nchannels, sampwidth, framerate, nframes, comptype, compname), which is channel numbers, sample width, frame rate, frame numbers, component type and component names respectively.

In our experiment, we only care about the first four parameters: nchannels, sampwidth, framerate, nframes. Then in order to keep the read write safe, we need to use the close function to close the .wav file.



```python
f = wave.open(r"test.wav", "rb")
# (nchannels, sampwidth, framerate, nframes, comptype, compname)

params = f.getparams()

nchannels, sampwidth, framerate, nframes = params[:4]

str_data = f.readframes(nframes)

f.close()
```

Module --- **Numpy**

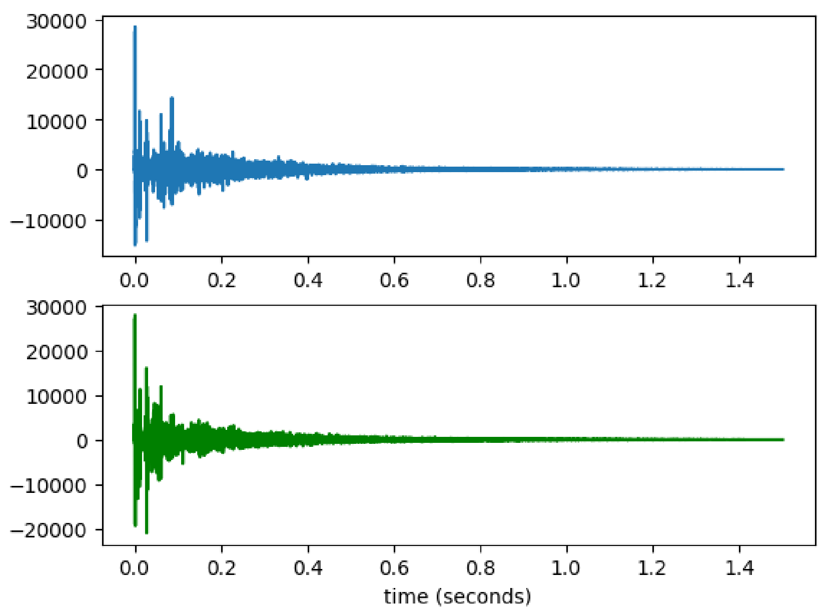
Numpy module is more like a math tool used in python environment to help the users implement their calculation between matrixes and vectors including transpose and dot product. We are going to operate on the parameters from the deconstruction of wav file.

```
wave_data.shape = -1, 2
wave_data = wave_data.T
time = np.arange(0, nframes) * (1.0 / framerate)
```

Module --- **pylab**
In this module provided by python, we are able to draw several subplots in the same view and compare them. Also, it is able to connect lines between the dots in the arrays and make them form a picture in total. It is good to combine the pylab and numpy together since it is pretty easy to organize the dot created from the numpy operations into a plot. Here, we make the timeline as the x axis with the unit of second, and also make the calculation between frame number and frame rate as the y axis.

```
   pl.subplot(211)
   pl.plot(time, wave_data[0])
   pl.subplot(212)
   pl.plot(time, wave_data[1], c="g")
   pl.xlabel("time (seconds)")
   pl.show()
```

III. Demonstration



I just test a gun_voice.wav downloaded from the website and operate the python file on it. The pictures are created by the pylab in the python file. We can hear that in the wav file, there is a clear gun shoot voice at the very beginning. And it is vividly shown in the pictures. Also, I analyze the channel numbers by checking whether it is 1 or 2 which means I will separate the voice apart when there are several channels.

IV. Conclusion

Through this experiment, we can see that we could implement the DSP simply by using python and its related modules. We could open and processing the wav file by using Wave module. We could operate whatever on the data set created by Wave module to some kind of forms we like by using numpy module. Finally, we could use pylab or matplot to draw out the form of the wave data by draw the lines between the matrixes elements and plot them out in several views.