Novosibirsk State University

*THE REAL SCIENCE

Lab 3 - Spectrum

Department of Computer Science

Student: Patrushev Boris 19213

Advisor: Ruslan V. Akhpashev

Date: 03/03/2021

1. From Lab2(Signals), take the signal generated by you.

Take the code from the last lab:

In [154…

```
import numpy as np
import matplotlib.pyplot as plt
import sounddevice as sd
from scipy.io.wavfile import write

amplitude = 0.3
def timesamples(fs, duration):
    return (np.arange(np.ceil(duration * fs)) / fs)

def note(name):
    octave = int(name[-1])
    PITCHES = "c,c#,d,d#,e,f,f#,g,g#,a,a#,b".split(",")
    pitch = PITCHES.index(name[:-1].lower())
    return 440 * 2 ** ((octave - 4) + (pitch - 9) / 12)

def melody(fs):
    sig1 = amplitude * np.sin(2 * np.pi * note("G4") * timesamples(fs, 1))
    sig2 = amplitude * np.sin(2 * np.pi * note("C5") * timesamples(fs, 1))
    sig3 = amplitude * np.sin(2 * np.pi * note("G4") * timesamples(fs, 1))
    sig4 = amplitude * np.sin(2 * np.pi * note("A4") * timesamples(fs, 0.25))
    sig5 = amplitude * np.sin(2 * np.pi * note("B4") * timesamples(fs, 0.75))
    return np.append(np.append(np.append(np.append(sig1, sig2), sig3), sig4), sig5)
fs = 80000
hymn_ussr =melody(fs)
sd.play(hymn_ussr, fs, blocking=True)
```

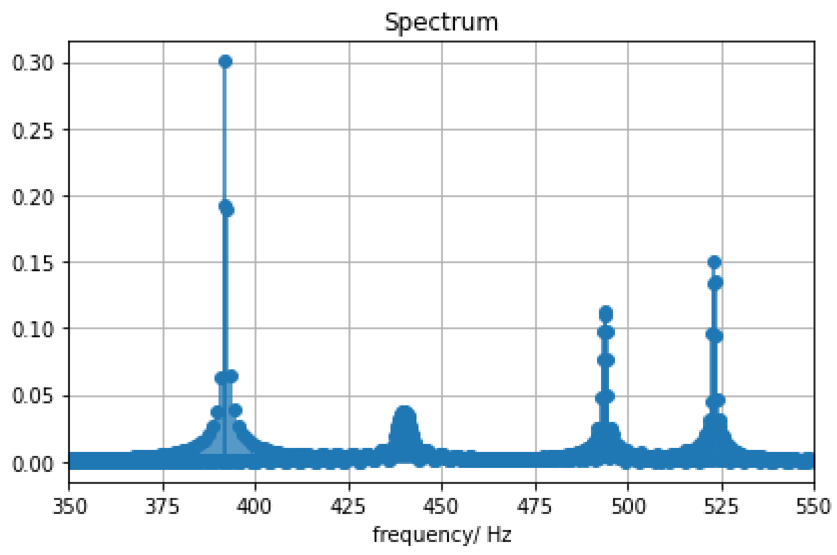
2. Using the Fourier transform, obtain the signal spectrum from item (1). Plot the spectral content.

The *Fourier transform* is a tool that allows you to see the contribution of each of these harmonic components, characterized by a certain frequency, in the signal under investigation. In this sense, the Fourier transform is said to expand the function in terms of frequencies.

In [158…

```
signalFFT = np.fft.fft(hymn_ussr)
# Graph: signal spectrum
```

```
plt.title('Spectrum')
plt.stem(timesamples(fs, 4) * fs/4, np.abs(signalFFT)/fs, basefmt='C0')
plt.xlim(350,550)
plt.xlabel('frequency/ Hz')
plt.grid()
plt.tight_layout()
```



We can see on the graph that the leftmost frequency is G4(frequency about 391.9954)and then anothers

3. Perform the inverse Fourier transform for the resulting spectral component. Compare with the original in step 1.

Now we can do the *inverse Fourier transform*, listen and compare it with original

In [161…
```
sd.play(np.float64(np.fft.ifft(signalFFT)), 80000, blocking=True)
```

```
<ipython-input-161-f7fb1788812c>:1: ComplexWarning: Casting complex values to real d
iscards the imaginary part
  sd.play(np.float64(np.fft.ifft(signalFFT)), 80000, blocking=True)
```

Original:

In [160…
```
sd.play(hymn_ussr, fs, blocking=True)
```

As we hear-everything is the same

4. Record your voice with the microphone

We will record our voice(3 sec) and immediately listen to what happened

In [211…
```
fs=100000
duration = 3  # seconds
myrecord = sd.rec(duration * fs, samplerate=fs, channels=1,dtype='float64')
myrecord = myrecord.reshape(myrecord.size,)
print ("Recording Audio")
sd.wait()
print ("Audio recording complete , Play Audio")
sd.play(myrecord, fs)
```

```
Recording Audio
Audio recording complete , Play Audio
```

5. Analyze the influence of the sampling frequency on the quality of the reproduced sound. To conclude.

Now let's try to listen my voice at different sampling frequency and understand how the quality changes

In [216…

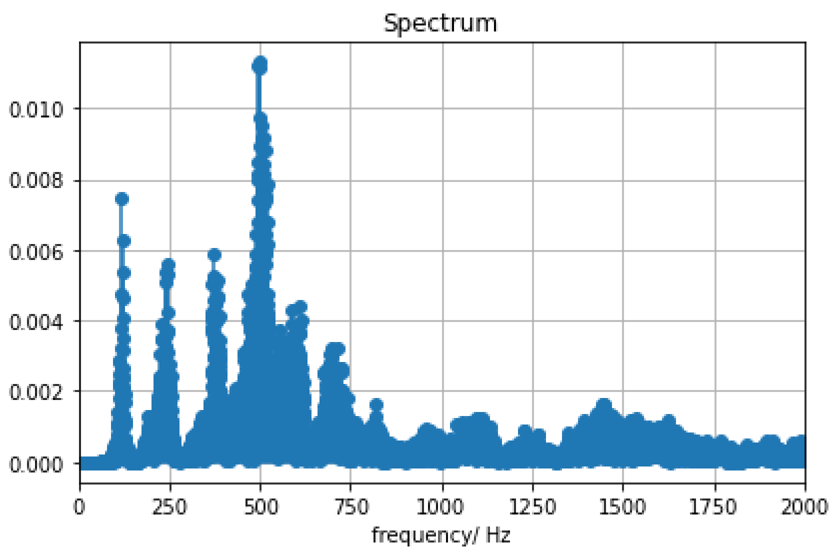
```
def zipmelody(melody, duration, fs):
    f = len(melody) // duration
    ans = []
    for i in range (fs * duration):
        idx = int(i * (f / fs))
        ans.append(melody[idx])
    return ans

fss = [100000, 70000, 44000, 10000, 5000, 1000]
for i in fss:
    sd.play(zipmelody(myrecord, 3, i), i)
    sd.wait()
```

- 100000, 70000, 44000 - no difference
- 10000 - there was an unpleasant hiss and the voice became less clear
- 5000 - the voice became even worse for perception
- it feels like I'm in a tank or a submarine :)

6. Get the frequency spectrum of the recorded sound using the Fourier transform.

In [180…

```
signalFFT = np.fft.fft(myrecord)
# Graph: signal spectrum
plt.title('Spectrum')
plt.stem(timesamples(fs, 3) * fs/3, np.abs(signalFFT)/fs, basefmt='C0')
plt.xlim(0,2000)
plt.xlabel('frequency/ Hz')
plt.grid()
plt.tight_layout()
```


Spectrum
frequency/ Hz