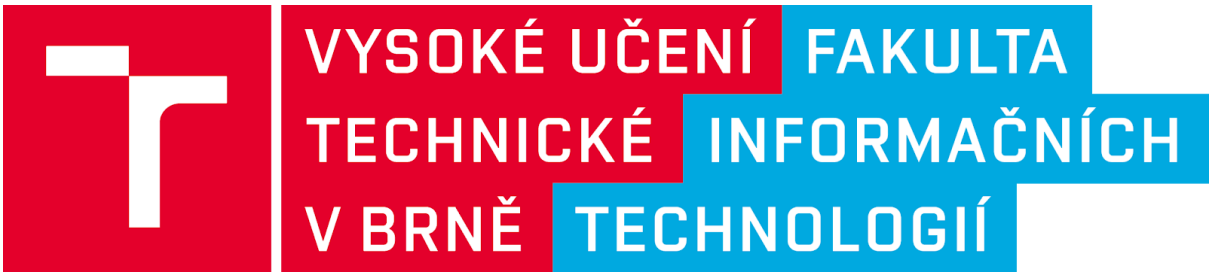


ISS Projekt 2020 / 21

Autor: Yehor Pohrebniak(xpohre00)

Datum: 23.12.2020

Řešení

Řešení jsem implementoval v pythonu. Použil jsem knihovny **numpy, scipy, matplotlib, random.** Všechna audia jsem nahrál při pomoci smartphonu.

Zadání

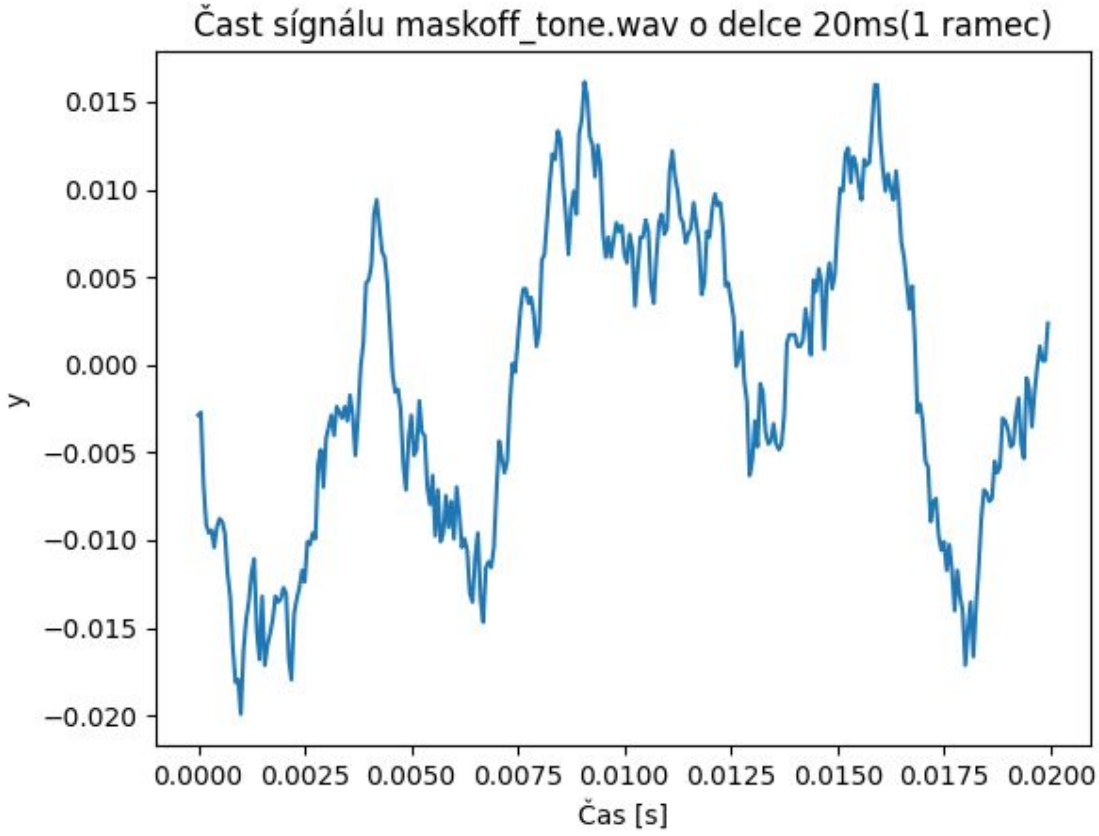
1.

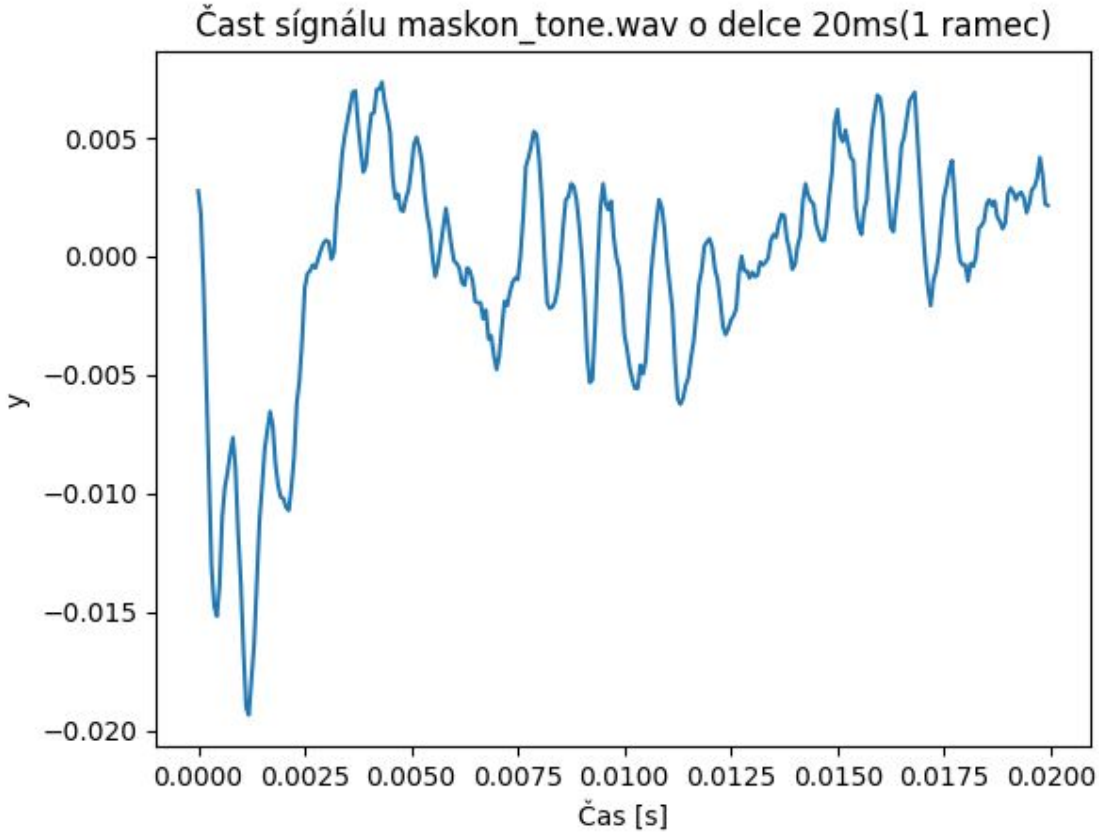
Název	Délka v [s]	Délka ve vzorcích
maskon_tone.wav	1.36	21760
maskoff_tone.wav	1.44	23040

2.

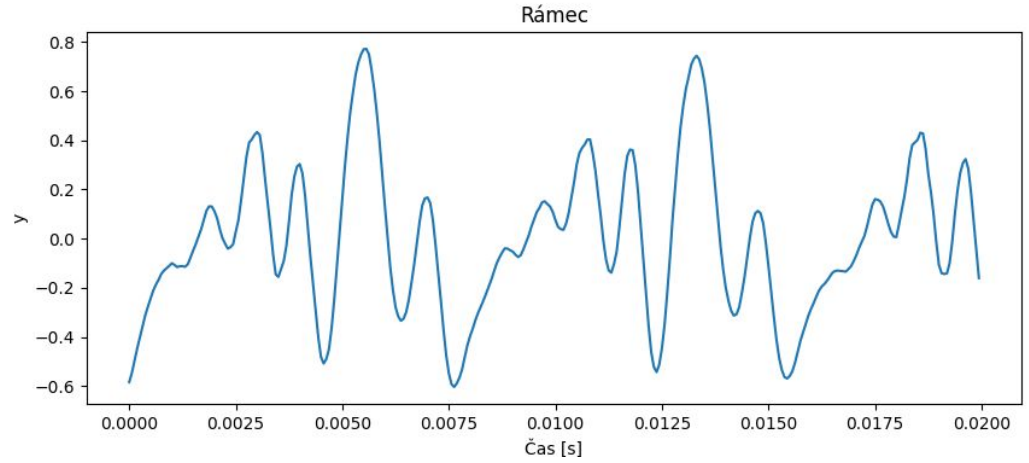
Název	Délka v [s]	Délka ve vzorcích
maskon_sentence.wav	4.4	70400
maskoff_sentence.wav	4.42	70720

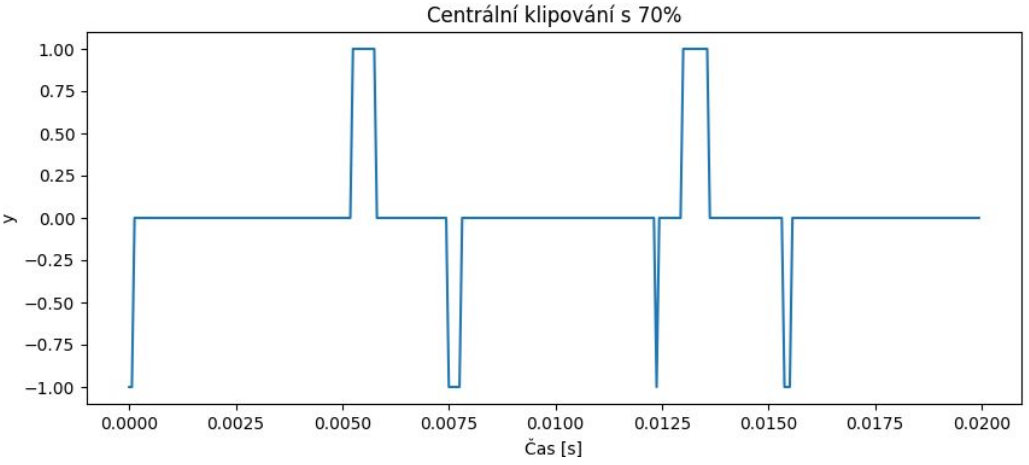
3. Pro vytvoření rámců jsem zahodil poslední 100 ramec pro každý signál. Seznamy `frameMaskOffTone a frameMaskOnTone` obsahují 99 rámců z signalů **maskoff_ton.wav** a **maskon_ton.wav**. Frekvence každého signálu je 16000 Hz, což znamená, že signál delkou v 1s bude mít velikost 16000 vzorců. Pro 1 ramec délkou 20 ms jsem našel velikost ve vzorcích při pomocí vztahu 1s/20ms = 16000/x, x=>320, délka jednoho ramcu = 320 vzorců.

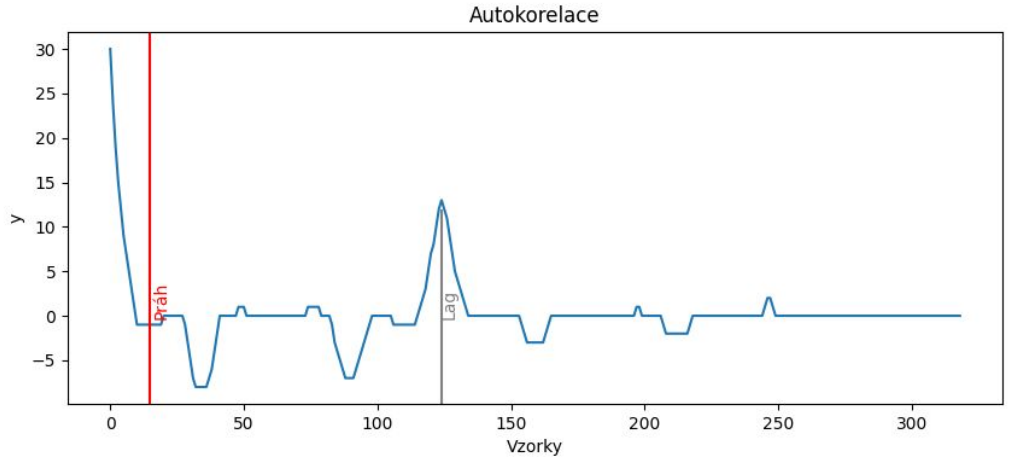

Část sígnálu maskoff_tone.wav o delce 20ms(1 ramec)
0.015
0.010
0.005
0.000
−0.005
−0.010
−0.015
−0.020
y
0.0000 0.0025 0.0050 0.0075 0.0100 0.0125 0.0150 0.0175 0.0200
Čas [s]

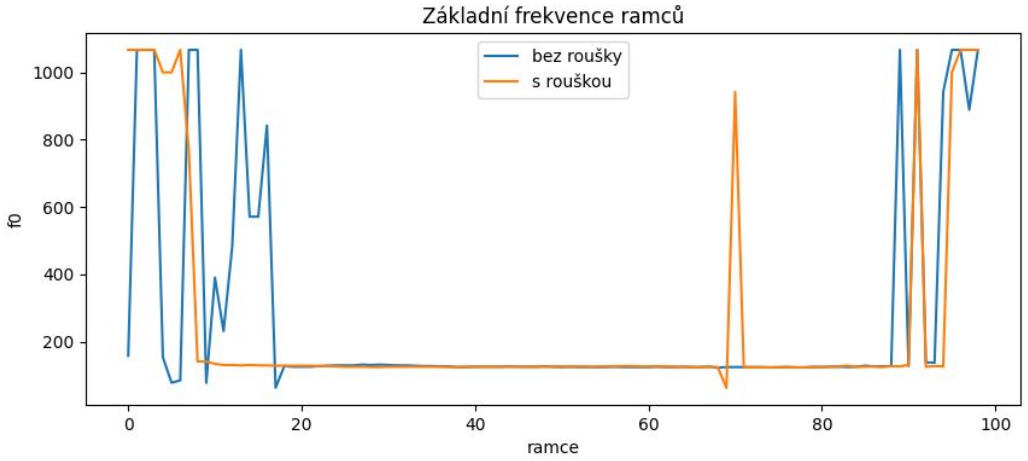

Část sígnálu maskon_tone.wav o delce 20ms(1 ramec)
0.005
0.000
−0.005
−0.010
−0.015
−0.020
y
0.0000 0.0025 0.0050 0.0075 0.0100 0.0125 0.0150 0.0175 0.0200
Čas [s]

4.


Rámec
0.8
0.6
0.4
0.2
0.0
−0.2
−0.4
−0.6
y
0.0000
0.0025
0.0050
0.0075
0.0100
0.0125
0.0150
0.0175
0.0200
Čas [s]


Centrální klipování s 70%
1.00
0.75
0.50
0.25
0.00
−0.25
−0.50
−0.75
−1.00
y
0.0000
0.0025
0.0050
0.0075
0.0100
0.0125
0.0150
0.0175
0.0200
Čas [s]





Střední hodnota:

základní frekvence rámců bez roušky = 269.27

základní frekvence rámců s rouškou = 252.95

Rozptyl:

základní frekvence rámců bez roušky = 102856.40

základní frekvence rámců s rouškou = 98317.72

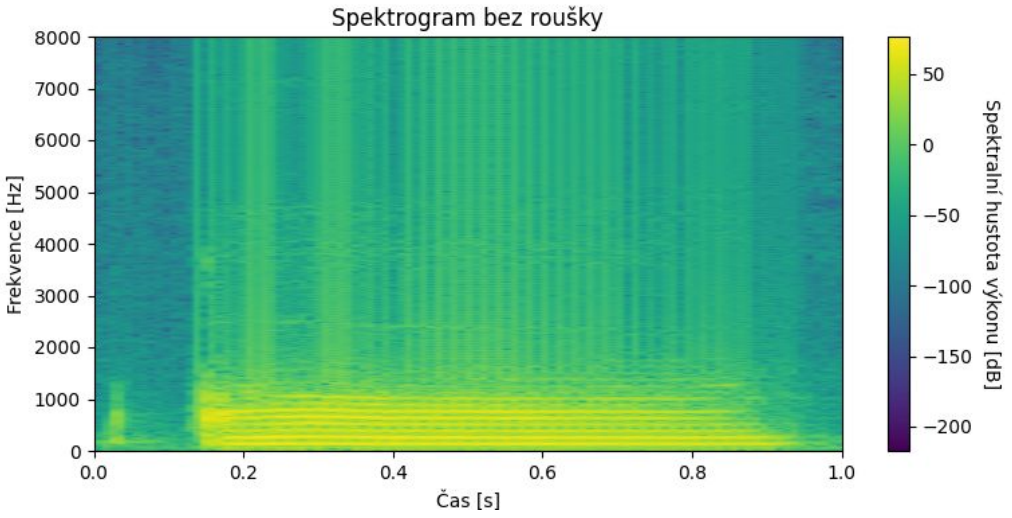
Odpověď na otázku: Zmenšení základní frekvence, by se dalo zmenšení velikosti f0 pry chybě +-1. Protože f0 = Fs/lag, coz kvůli většímu Fs bude tvořit velký rozdíl mezi f0

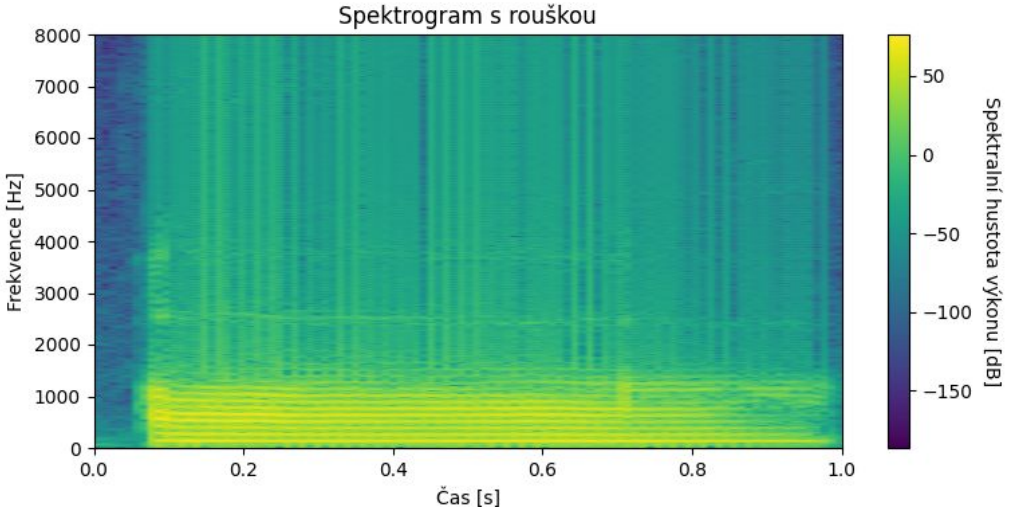
5. Funkci DFT jsem implementoval v Pythonu (funkce se jmenuje dft).

```
def dft(s):
    s = np.asarray(s, dtype=float)
    N = s.shape[0]
    n = np.arange(N)
    k = n.reshape((N, 1))
    M = np.exp(-2j * np.pi * k * n/N)
    return np.dot(M, s)
```

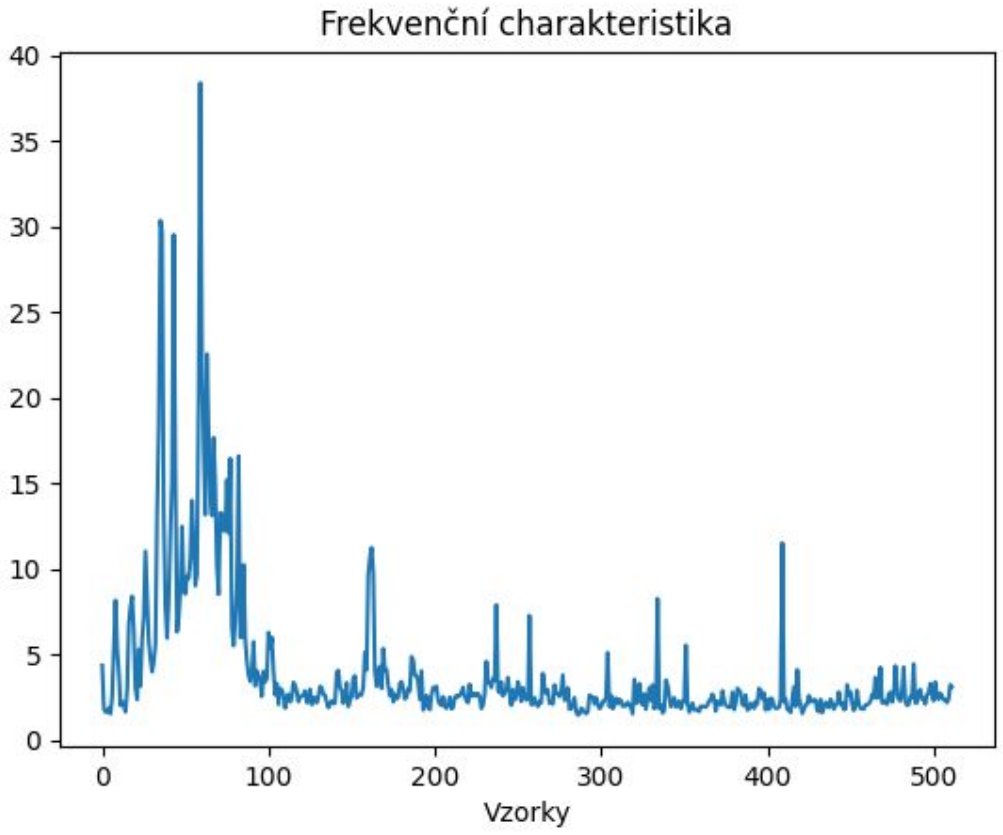
Kontrola probiha s funkci np.fft.fft()(Náhodně se vybere ramec, který bude transformován dvěma funkcemi a výsledky se pak porovnají) . DFT pro signaly provádím při pomoci np.fft.fft(), protože moje funkce je pomalejší a trvá to nějaký čas, než na DFT převede 1024 x 99 vzorků. Pro funkci dft jsem použil následující vzorec

$$x[k] = \sum_{n=0}^{N-1} x[n] e^{\frac{-j2\pi kn}{N}}$$





6. **$H(e^{jw})$ = |DFT(MaskOn)| / |DFT(MaskOff)|**



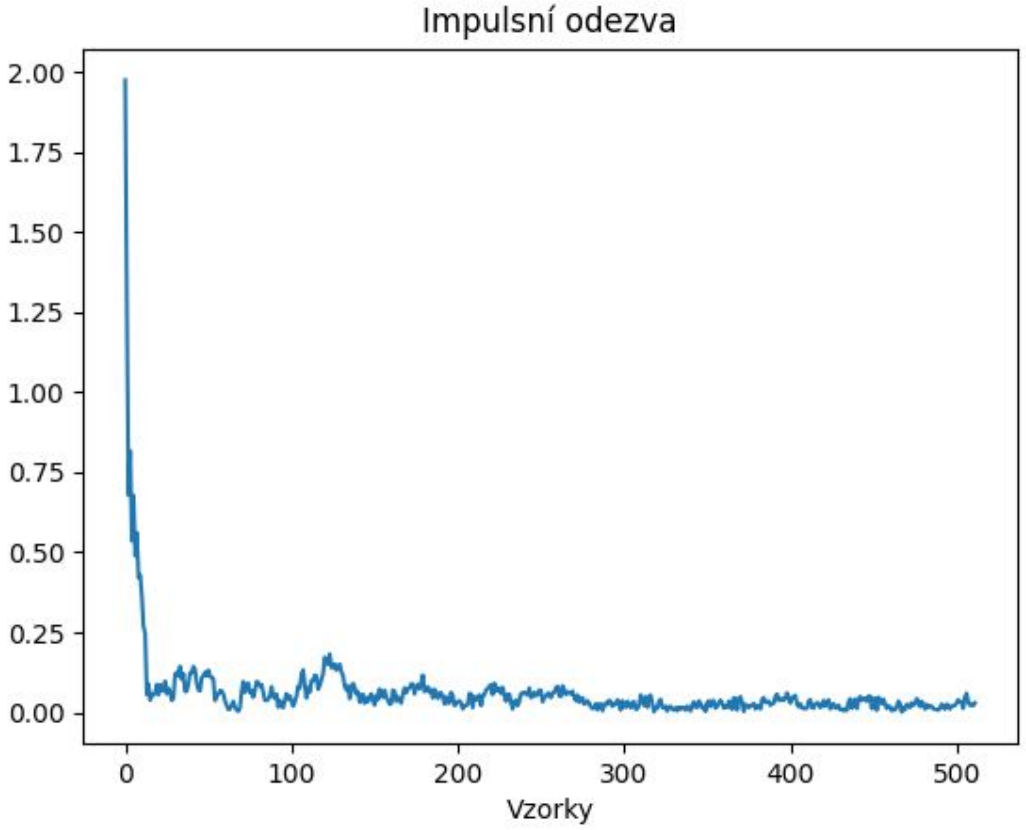
Je to FIR filtr s horní propustí

7. Funkci IDFT jsem implementoval v Pythonu(funkce jmenuje se idft).

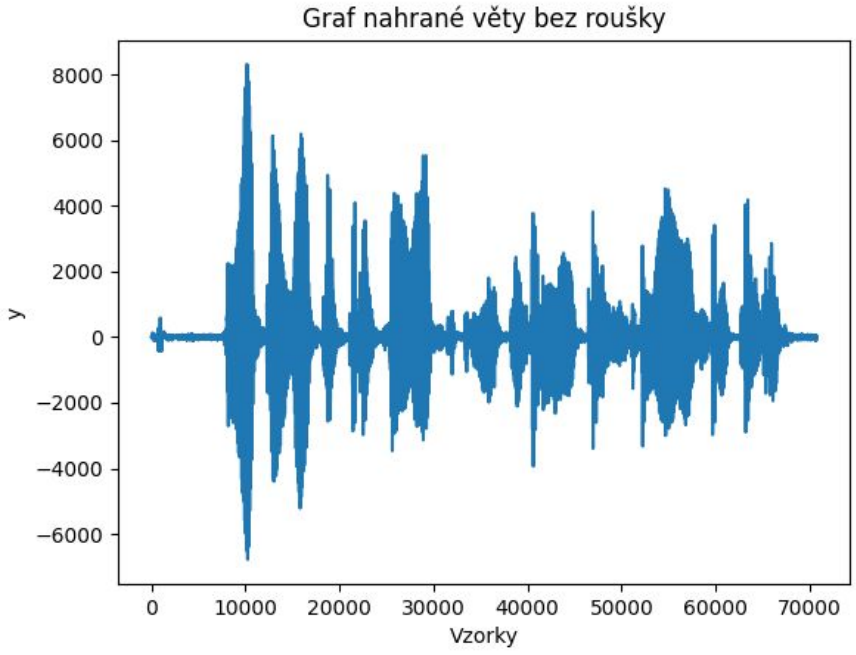
```
def idft(s):
    s = np.asarray(s, dtype=float)
    N = s.shape[0]
    n = np.arange(N)
    k = n.reshape((N, 1))
    M = np.exp(2j * np.pi * k * n/N)
    return np.dot(M, s)/N
```

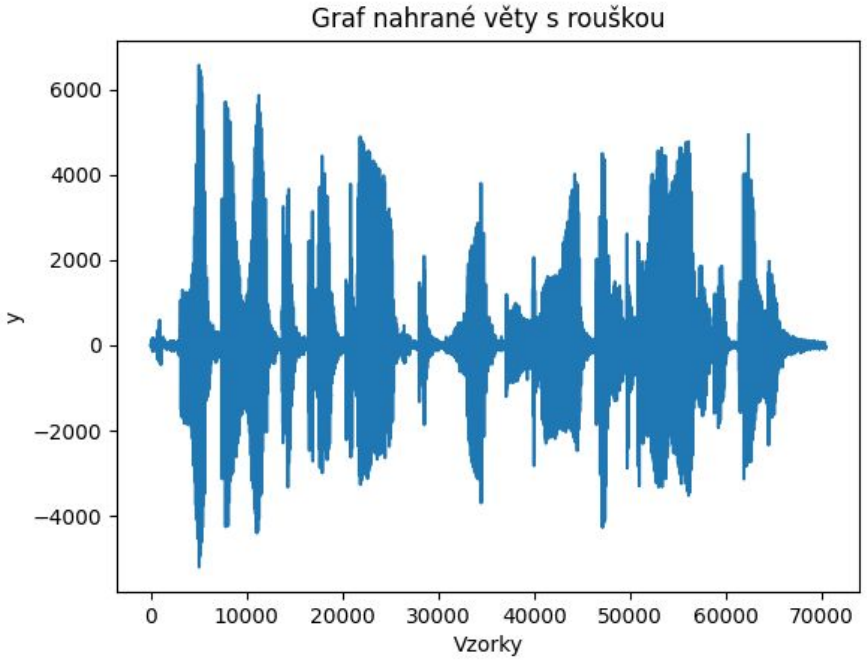
Kontrola probiha s funkci np.fft.ifft(). Použil jsem následující vzorec pro implementaci IDFT:

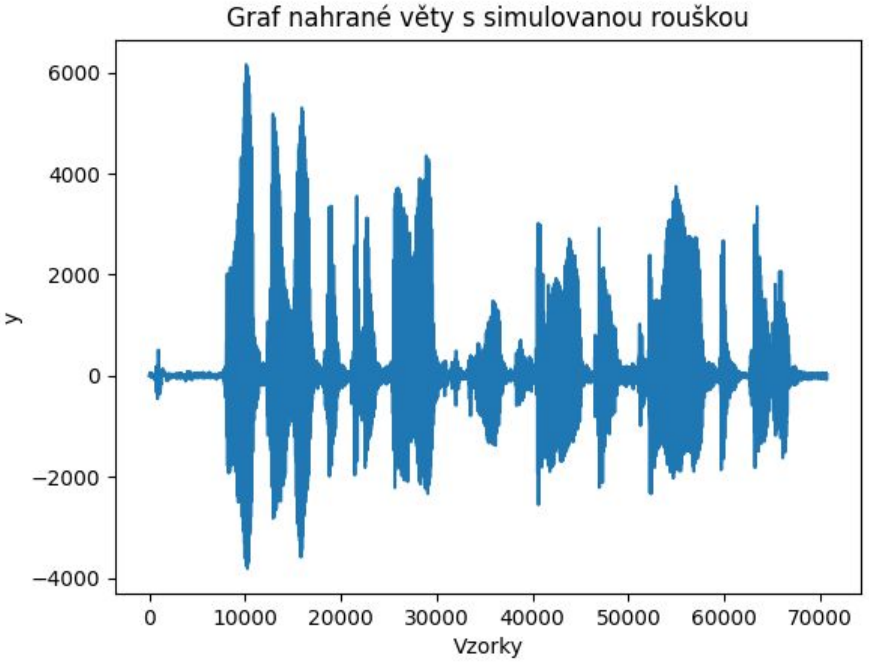
$$x(n) = \frac{1}{N} \sum_{k=0}^{N-1} X(k) e^{j \frac{2\pi k n}{N}}$$



8. Filtraci signálu jsem provedl pří pomocí funkcí **signal.lfilter**. Pro vytváření nahrávek **sim_maskon_tone.wav** a **sim_maskon_sentence.wav** jsem použil funkci wavfile.write a okomentoval jsem 2 řádky v kódu, aby při spuštění se nevytvořili opět soubory. Signály jsem převedl na int16 pro soubory wav.







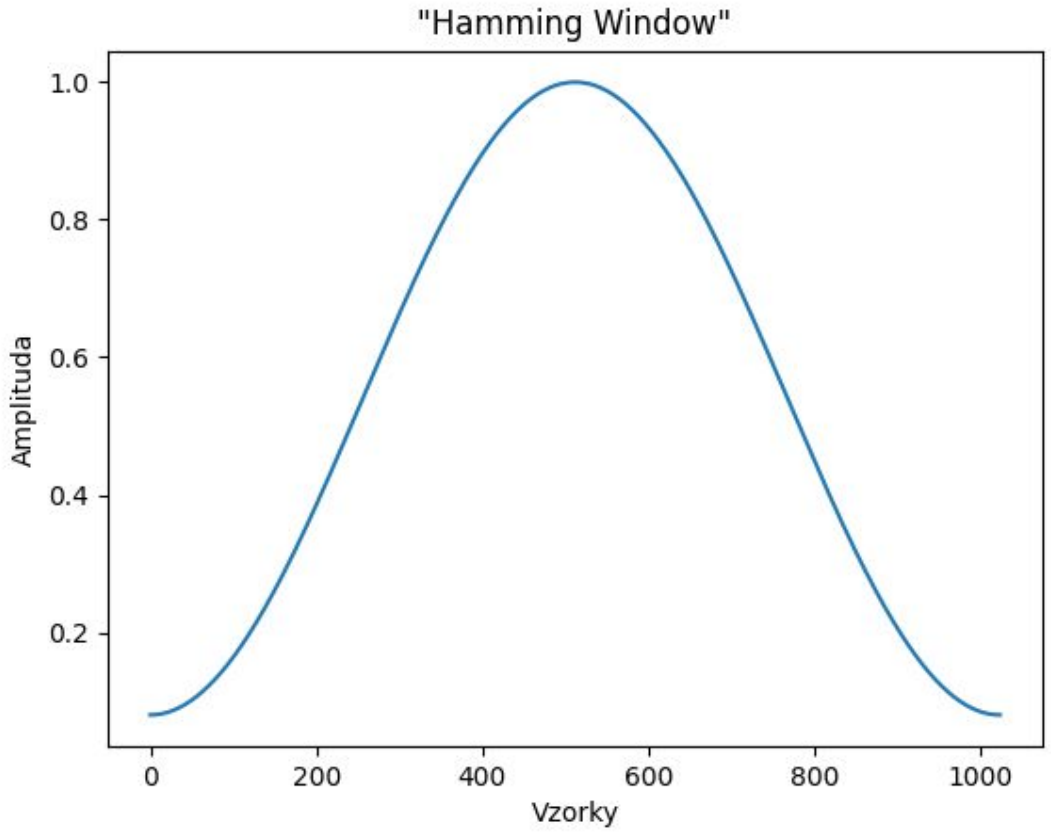
Signály s rouškou a simulovanou rouškou jsou mezi sebou podobný. Nejvíce se věty podobají na začátku a na konci a trochu se liší ve středu v intervalech(30000 - 60000) v signálu s rouškou a (20000 - 60000) v signálu s simulovanou rouškou.

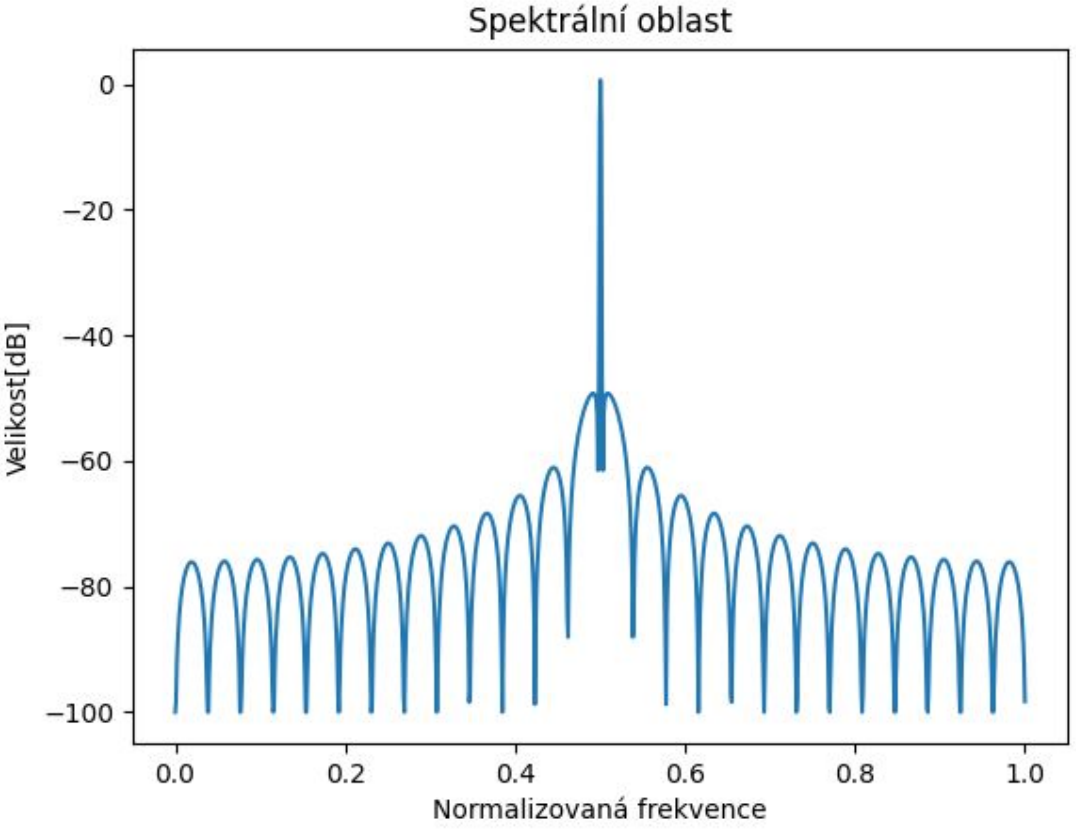
9. **Závěr:** Moje řešení funguje správně, jenom trochu pomalu, kvůli tomu, že jsem nenašel lepší způsob jak implementovat autokorelaci(trva to nějakých 7 - 8 sekund). Taky nejsem si jistý, že moje nahrávky mají stejný tón i když jsem několikrát nahrával.

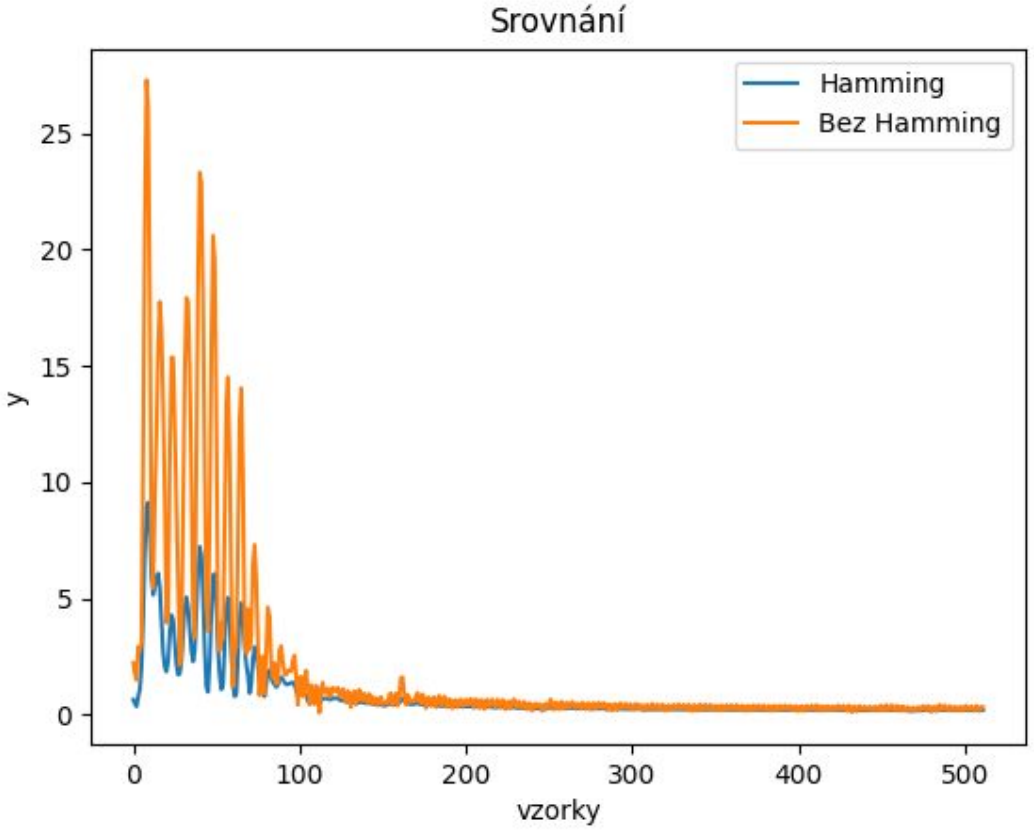
11. Vybral jsem Hammingovou okenkovou funkci(**numpy.hamming()**).

$$w(n) = 0.54 - 0.46cos\left(\frac{2\pi n}{M-1}\right) \qquad 0 \leq n \leq M-1$$

Vytvořil jsem pole obsahující 1024 prvky a provedl jsem nasobení ramce před použitim funkci np.fft.fft().




Spektrální oblast
Velikost[dB]
0
−20
−40
−60
−80
−100
0.0
0.2
0.4
0.6
0.8
1.0
Normalizovaná frekvence


Srovnání
Hamming
Bez Hamming
y
25
20
15
10
5
0
0
100
200
300
400
500
vzorky