VYSOKÉ UČENÍ TECHNICKÉ
FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
VUT V BRNĚ

ISS - Signály a systémy

Slavomír Svorada (xsvora02)

4.1.2021

Riešenie:

Pr. 1

V audacity som si vytvoril nahrávky a pomocou programu ffmpeg upravil na požadovaný formát.

Názov súboru	Dĺžka nahrávok vo vzorkách	Dĺžka nahrávok v sekundách
maskoff_tone.wav	40000	02.50
maskon_tone.wav	40000	02.50

Pr. 2.

Postup bol rovnaký ako v úlohe 1.

Názov súboru	Dĺžka nahrávok vo vzorkách	Dĺžka nahrávok v sekundách
maskoff_sentence.wav	40124	02.51
maskon_sentence.wav	40124	02.51

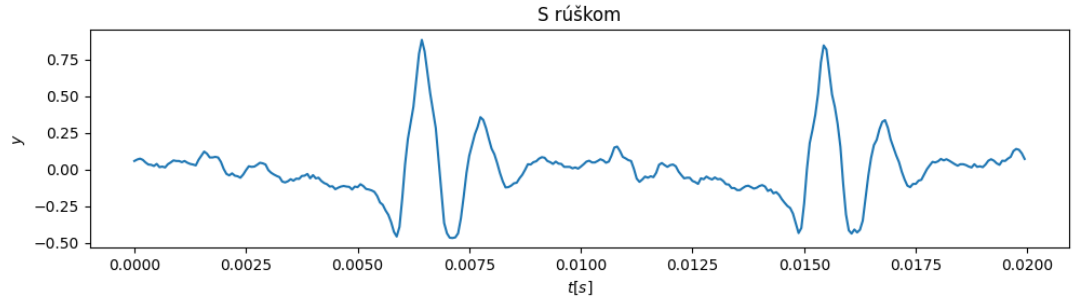
Pr. 3

Vzorec pre výpočet velkosti rámca : počet vzorkov(s = 16000) * 0.02 sekundy

```
fr_size = int(s * 0.02)
```

Zo začiatku som extrahoval časť o dĺžky 1 sekundy a následne urobil ustrednenie a normalizáciu do dynamického rozsahu [-1; 1] pomocou np.mean() a np.abs().max() ktoré boli v zadaní projektu. Ďalej sú signály rozdelené na 100 rámcov o dĺžke 20ms.

Grafy rámcov som vykreslil s indexom 0:



Pr. 4

Najskôr som si zistil 70% maxima absolútnej hodnoty a pomocou for cyklov určujem aké hodnoty budú priradené do daných rámcov.

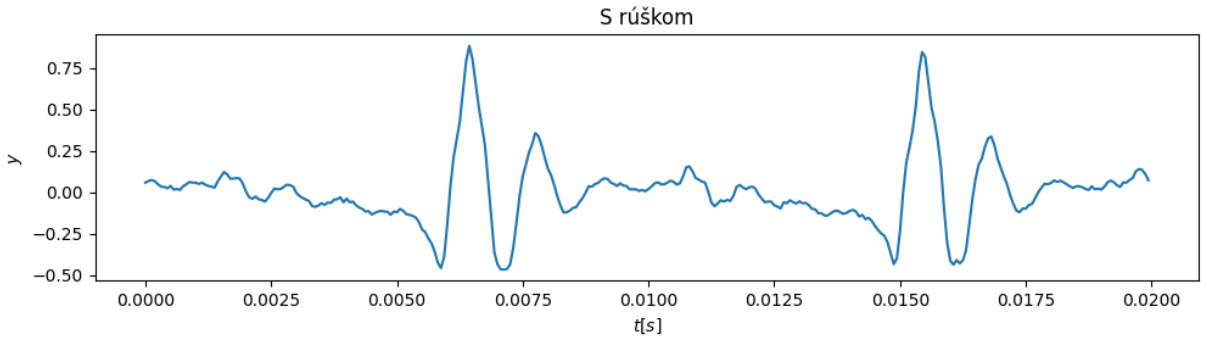
Pri autokorelácií som použil taktiež for cykly:

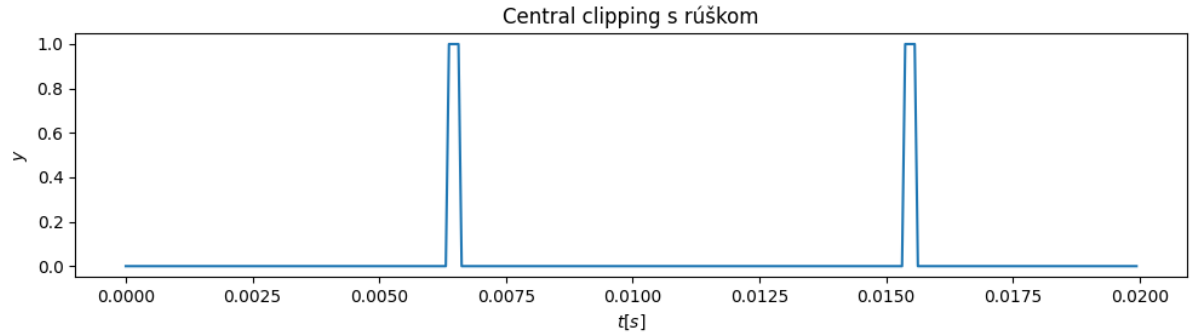
```
for i in range(0, 99):
    for j in range(0, 320):
        autocor_off[i][j] = 0
        autocor_on[i][j] = 0
        for k in range(0, 319 - j):
            result_off = clip_off[i][k] * clip_off[i][k + j]
            autocor_off[i][j] = autocor_off[i][j] + result_off
            result_on = clip_on[i][k] * clip_on[i][k + j]
            autocor_on[i][j] = autocor_on[i][j] + result_on
```

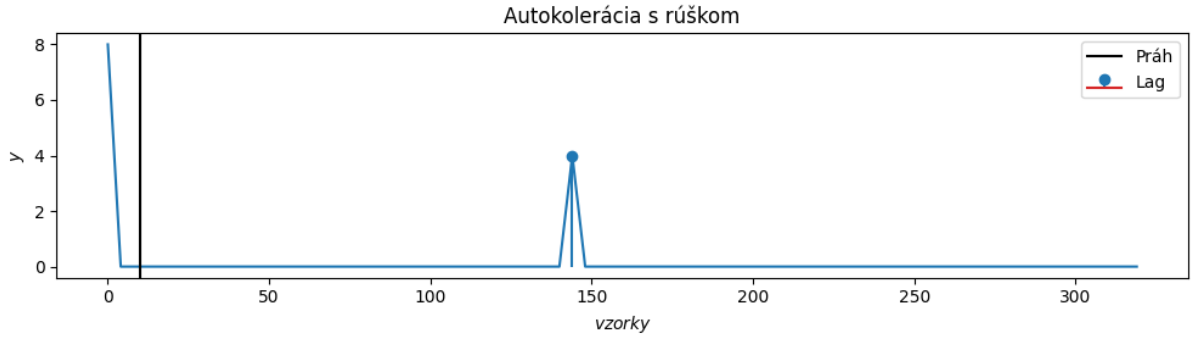
Nakoniec som index maximálneho koeficientu previedol na frekvenciu:

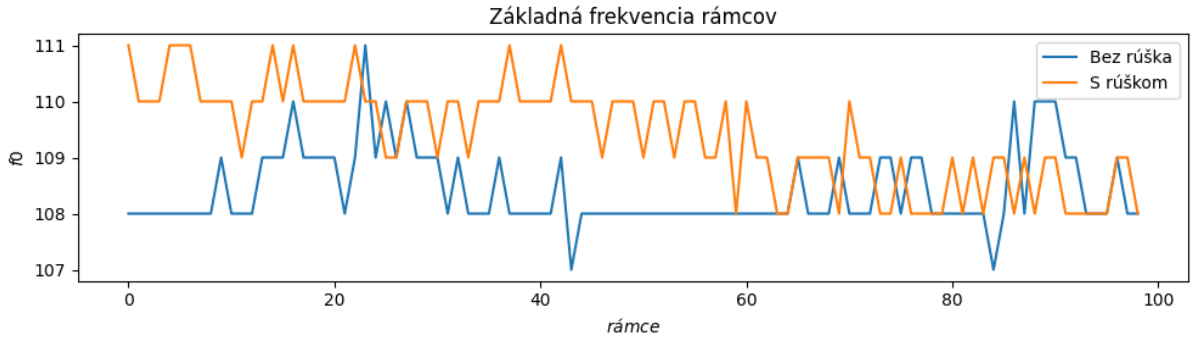
```
for i in range(0, 99):
    f0_off[i] = (1 / lag_off[i]) * s
    f0_on[i] = (1 / lag_on[i]) * s1
```

(a) Grafy:









(b)

Strednú hodnotu som si vypočítal pomocou funkcie np.median() a rozptyl pomocou funkcie np.var().

Typ	Stredná hodnota	Rozptyl
Bez rúška	108.0	0.48505254565860634
S rúškom	110.0	0.8266503418018569

(c)

Pri zmene hodnoty o 1 by skok mohla zmenšiť vzorkovacia frekvencia. Keby bola nižšia, tak skok nie je taký veľký.

Pr. 5

Funkcia implementujúca DFT:

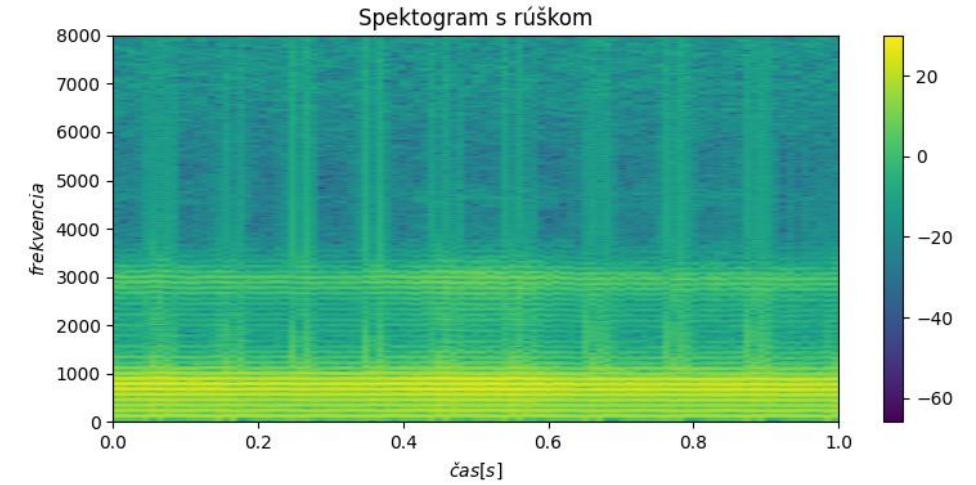
Po vytvorení vlastnej implementácie dft som zistil, že oproti fukncií np.fft.fft je oveľa pomalšia a preto ju ďalej nepoužívam.

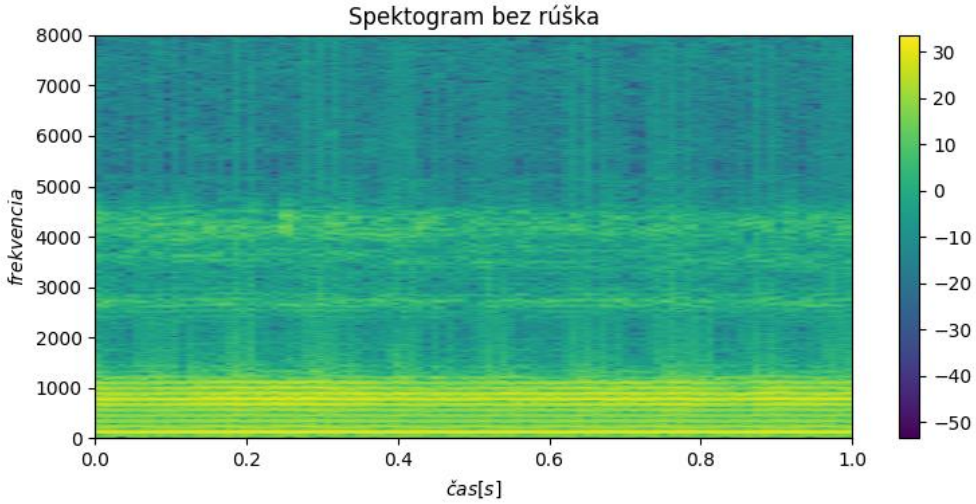
```
dft_off = np.zeros((99, 1024), dtype=complex)
dft_on = np.zeros((99, 1024), dtype=complex)
dft_f_off = np.ndarray((99, 1024), dtype=complex)
dft_f_on = np.ndarray((99, 1024), dtype=complex)
ddft_f_off = np.ndarray((99, 1024), dtype=complex)
ddft_f_on = np.ndarray((99, 1024), dtype=complex)

for i in range(0, 99):
    for j in range(0, 320):
        dft_off[i][j] = frames_off[i][j]
        dft_on[i][j] = frames_on[i][j]

for i in range(0, 99):
    for j in range(0, 1024):
        for k in range(0, 1024):
            dft_f_off[i][j] = dft_f_off[i][j] + dft_off[i][k] * (np.exp((-1j * 2 * np.pi * k * j) / 1024))
            dft_f_on[i][j] = dft_f_on[i][j] + dft_on[i][k] * (np.exp((-1j * 2 * np.pi * k * j) / 1024))
```

Hodnoty jednotlivých koeficientov som upravil podra vzorca: P[k] = 10 log10 |X[k]|^2 ktorý bol v zadaní projektu a zobral polovicu vzorkov z každého rámca:

```
# P[k] = 10 log10 |X[k]|^2
log_off = np.ndarray((99, 1024), float)
log_on = np.ndarray((99, 1024), float)
for i in range(0, 99):
    log_off[i] = 10 * np.log10(np.abs(dft_f_off[i])**2)
    log_on[i] = 10 * np.log10(np.abs(dft_f_on[i])**2)

# k = [0..512]
log_off_k = np.ndarray((99, 512), float)
log_on_k = np.ndarray((99, 512), float)
for i in range(0, 99):
    for j in range(0, 512):
        log_off_k[i][j] = log_off[i][j]
        log_on_k[i][j] = log_on[i][j]
```

Sprektogramy:

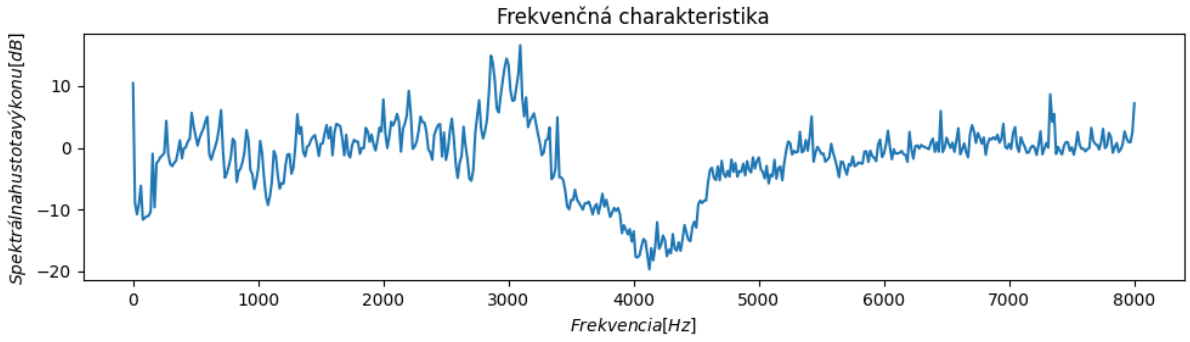




Pr. 6

Pre výpočet frekvenčnej charakteristiky som použil vzorec: $H(e^{j\omega}) = \frac{Y(e^{j\omega})}{X(e^{j\omega})}$. Čítateľ reprezentuje frekvenčnú charakteristiku filtrovaného signálu a menovateľ reprezentuje frekvenčnú charakteristiku pôvodného signálu. Čiže urobím: **DFT_ON / DFT_OFF**. Spriemerujem ju cez všetky rámce a následne zlogaritmujem.

Graf:



Na frekvencií jemne po 4000Hz dochádza ku zoslabeniu. Malý rozdiel vo frekvencií signálu vysvetluje celkom veľký skok v chovaní sa filtra.

Pr. 7

Implementácia IDFT:

Počítam pomocou vzorca: $$x[n] = \frac{1}{N} \sum_{k=0}^{N-1} X[k]\, e^{j2\pi kn/N}$$

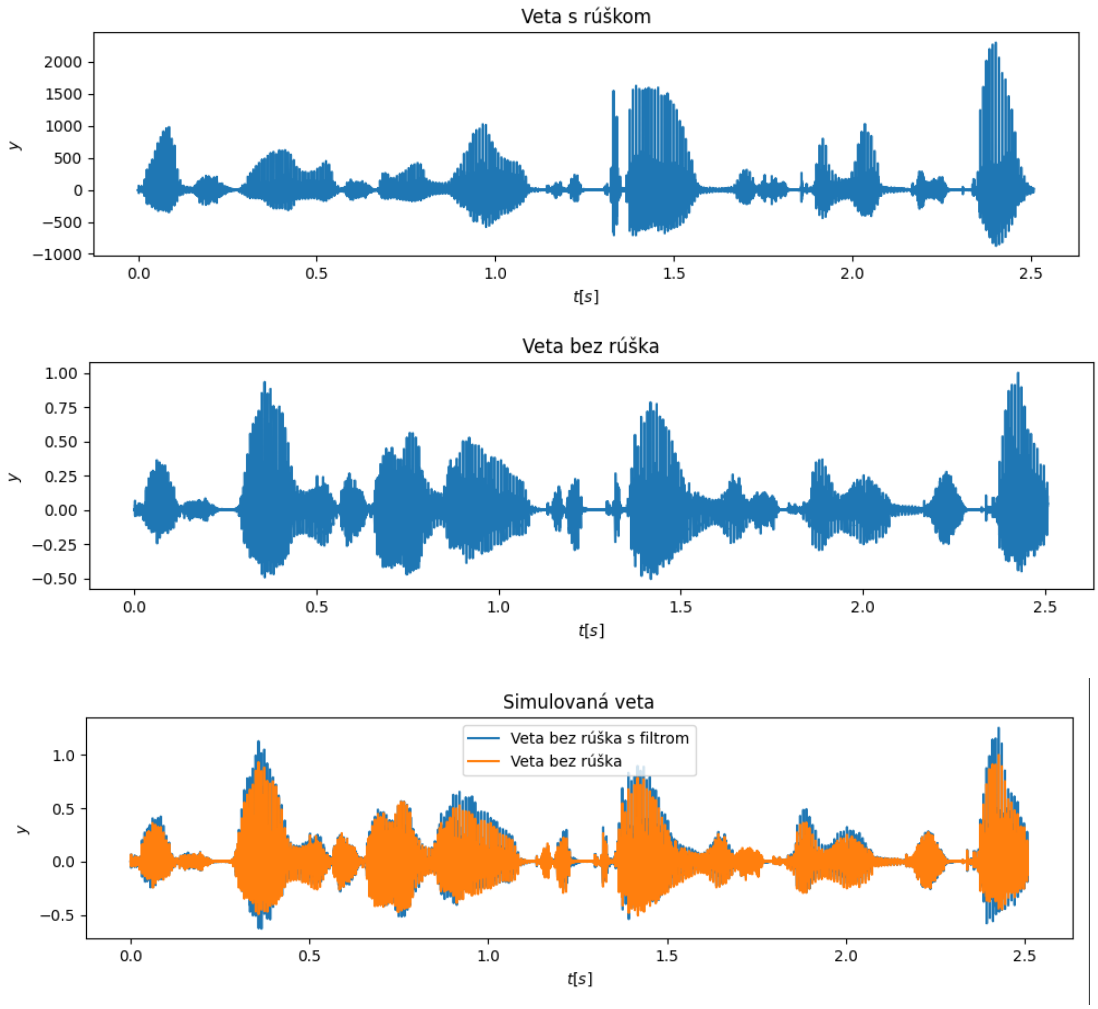
```
for i in range(0, 1024):
    i_dft_my[i] = 0
    for j in range(0, 1024):
        i_dft_my[i] = i_dft_my[i] + f_ch_average[j] * (np.exp((1j * 2 * np.pi * j * i) / 1024))

for i in range(0, 1024):
    i_dft_my[i] = i_dft_my[i] / 1024
```

Po vytvorení vlastnej implementácie idft som zistil, že oproti fukncií np.fft.ifft je oveľa pomalšia a preto ju ďalej nepoužívam. Pomocou idft prevediem frekvenčnú charakteristiku na impúlznu odozvu a vykreslím graf.

Graf impúlznej odozvy:

Pr. 8

Na začiatku som si nahral súbory maskoff_sentence.wav a maskon_sentence.wav. Tie som ustrednil pomocou funkcie np.mean() a normalizoval pomocou np.abs().max(). Filtráciu na tóne a vete som urobil pomocou funkcie scipy.signal.lfilter. Následne som si danú vetu a tón uložil a vypočul aby som mohol porovnať zmeny. Simulovaná veta a tón boli o čosi hlasnejšie a viac sa podobali ku signálu bez rúška.

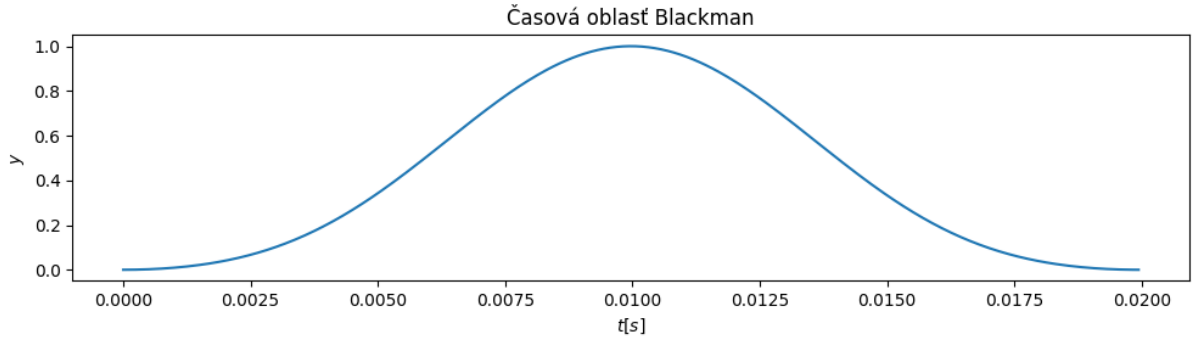


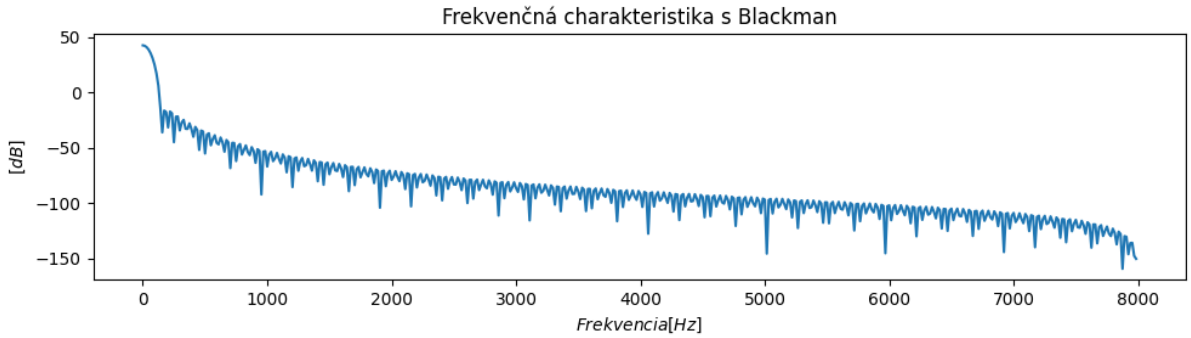
Pr. 9

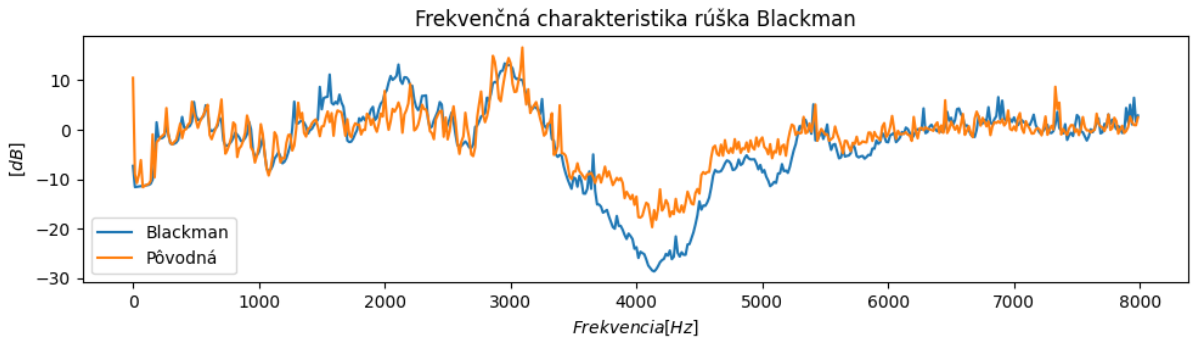
Pri spracovaní úloh som natrafil na rôzne problémy, ktoré som nakoniec vyriešil. Základná frekvencia rámcov v úlohe 4 nie je najlepšia ale opakovane som sa snažil nahrať lepšie nahrávky. Takže výsledok je najlepší aký sa mi podarilo spraviť. Z rozšírených úloh som robil úlohy 11, 12 a 13. Výsledok základnej frekvencie rámcov trocha ovplyvňuje úlohu 13 tým, že je len 16 rámcov ktoré maju rovnakú základnú frekvenciu ale inač by malo riešenie normálne fungovať. Ostatné riešenia by mali taktiež fungovať.

Pr. 11

Ako okienkovú funkciu som si vybral Blackman fukciu. Blackman funkciu aplikujem tesne pred spektrálnou analýzou. Zmeny na okrajoch rámcov spôsobuje rozdelenie signálov na rámce. Cielom funkcie je uhladiť tieto okraje rámcov.

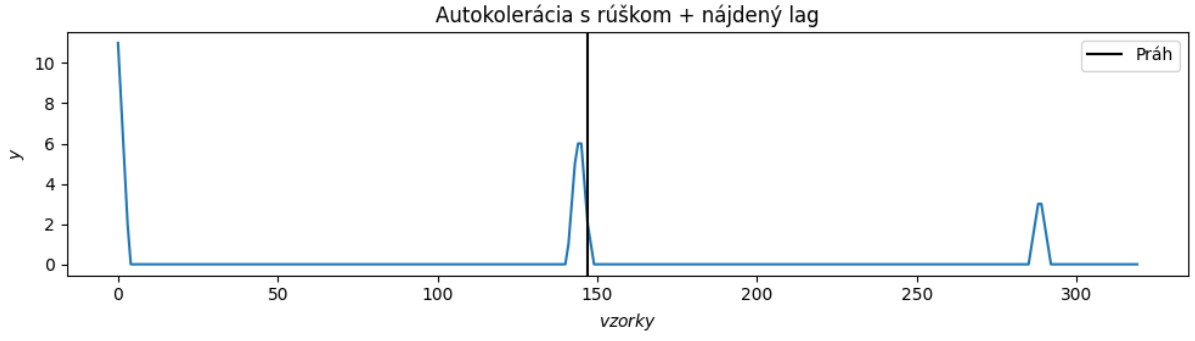






Pr. 12

Na základe toho, že som detekciu n-násobného lagu nezaznamenal, musel som si danú situáciu nasimulovať. Urobil som to tak, že som posúval prah detekcie kým sa detekcia prejavila.



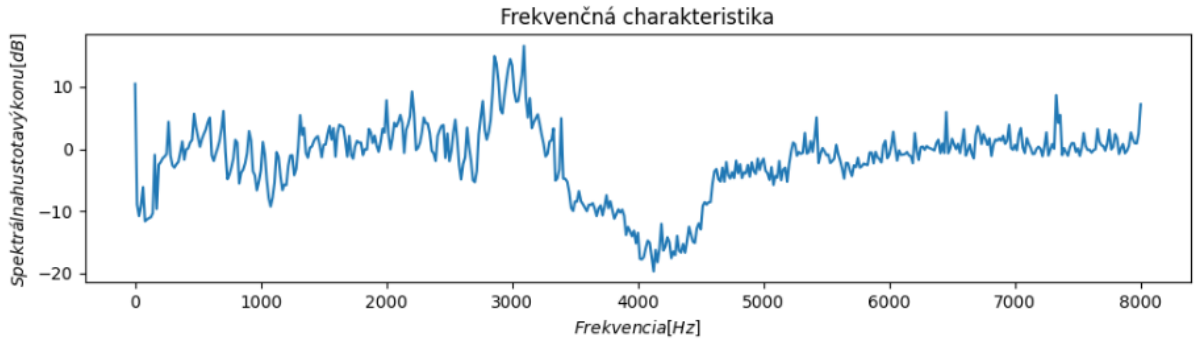
Chyba sa odstráni tak, že sa filtrácia urobí podľa vzťahu:

L(i) = med [L(i − k),L(i − k + 1), . . . ,L(i), . . . ,L(i + k)] čiže medián zoradí hodnoty podľa veľkosti a vyberie hodnotu, ktorá sa nachádza uprostred.

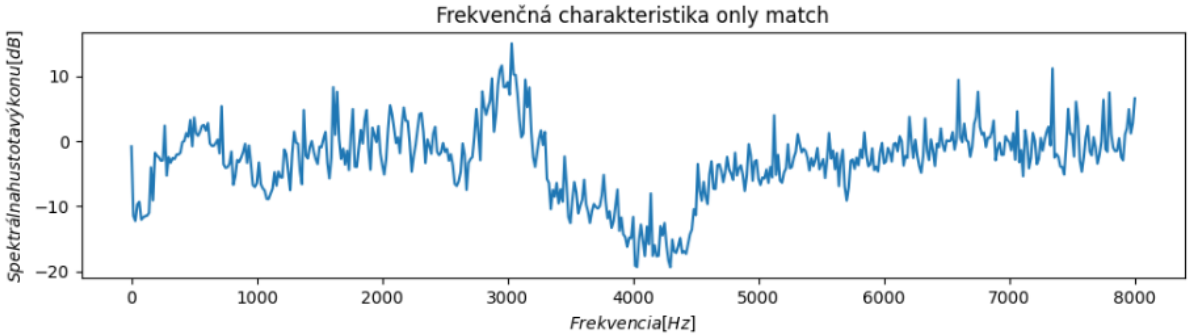
Pr. 13

V tejto úlohe som počítal frekvenčnú charakteristiku iba s rámcami ktoré majú rovnakú základnú frekvenciu. Pri výpočtoch som používal teda 16 rámcov. Postup bol veľmi podobný ako v predchadzajúcich úlohách 5, 6, 7, 8.

Pôvodná frekvenčná charakteristika:



Frekvenčná charakteristika **only match**:



Pri porovnaní grafov pôvodnej frekvenčnej charakteristiky a frekvenčnej charakteristiky s only match som zistil, že zmeny sú pozorovatelné ale veľmi malé. Následne som si uložil a vypočul nahrávku pre tón a pre vetu kde som zistil, že zmeny sú taktiež malé.

Zdroje:

http://www.stud.fit.vutbr.cz/~izmolikova/ISS/project/

https://deerishi.wordpress.com/2013/09/23/signal-processing-using-python-part-1/

https://www.fit.vutbr.cz/study/courses/ISS/public/.cs