ISS Projekt 2020/21

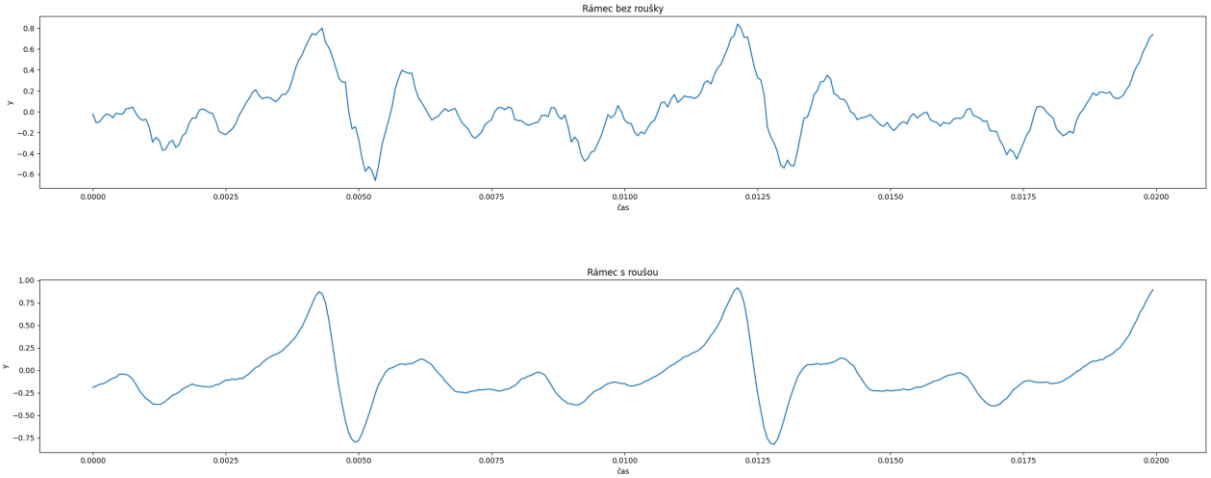
Ladislav Dokoupil
xdokou14

1a 2.

Nahrávka	Délka [vzorky]	Délka [s]
maskoff_tone.waw	99196	06.20
maskon_tone.waw	34737	02.17
maskoff_sentence.waw	51641	03.23
maskon_sentence.waw	47183	02.95

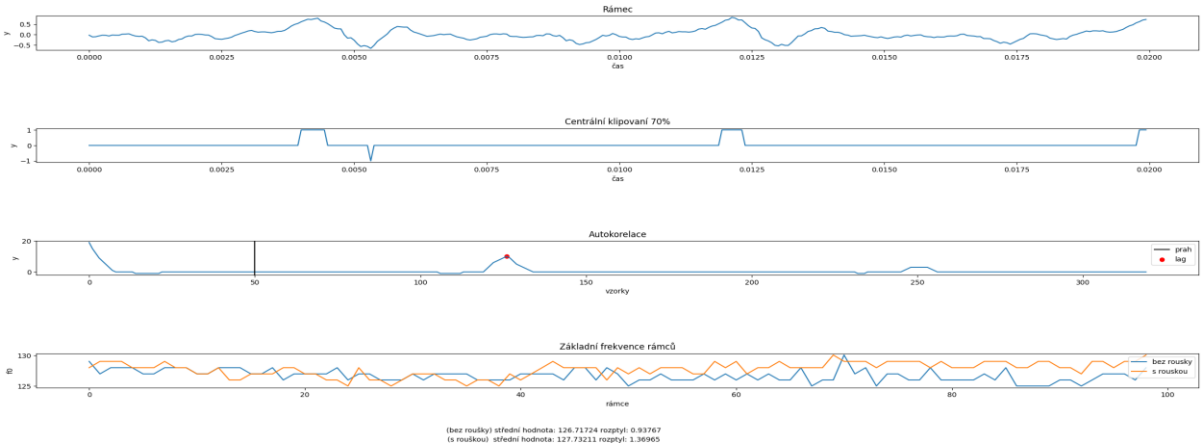
3.

Vzorec pro velikost ramce: fs[hz]*delka_ramce[s], např.: 16000*0.02



Pro vyhledání úseků s největší korelací jsem postupně procházel celou první nahrávku a každý sekundový úsek první nahrávky jsem koreloval s druhou nahrávkou. Přitom jsem si ukládal maximální dosaženou hodnotu korelace a s ní související indexy do nahrávek, které jsem poté použil jako začátky sekundových úseků.

4.



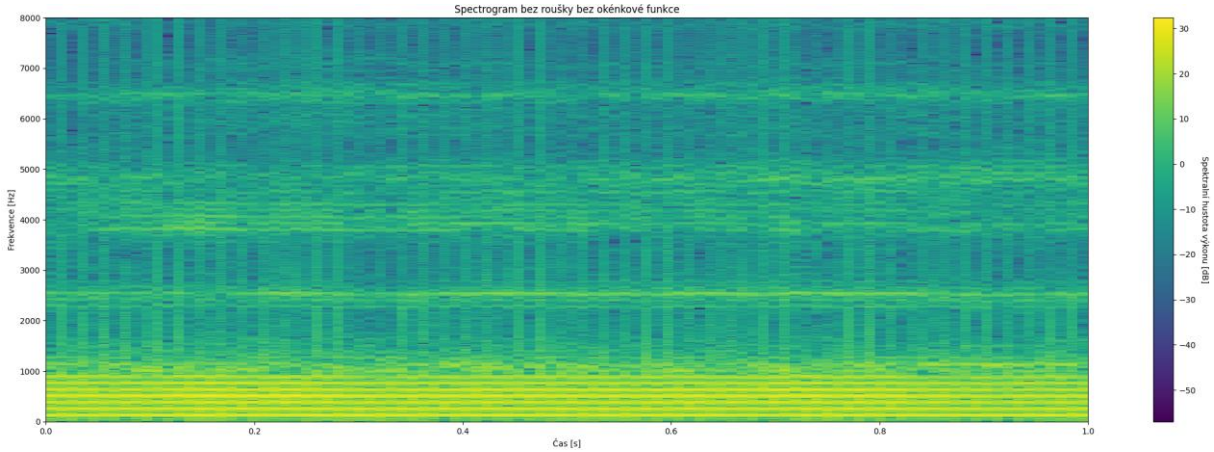
C) pro přesnější výpočet f0 bych si mohl vstupní rámce nadvzorkovat pomocí interpolačního filtru, tím by sice došlo k zvýšení vzorkovací frekvence, ale byl bych schopen přesněji určit index/frekvenci lagu.

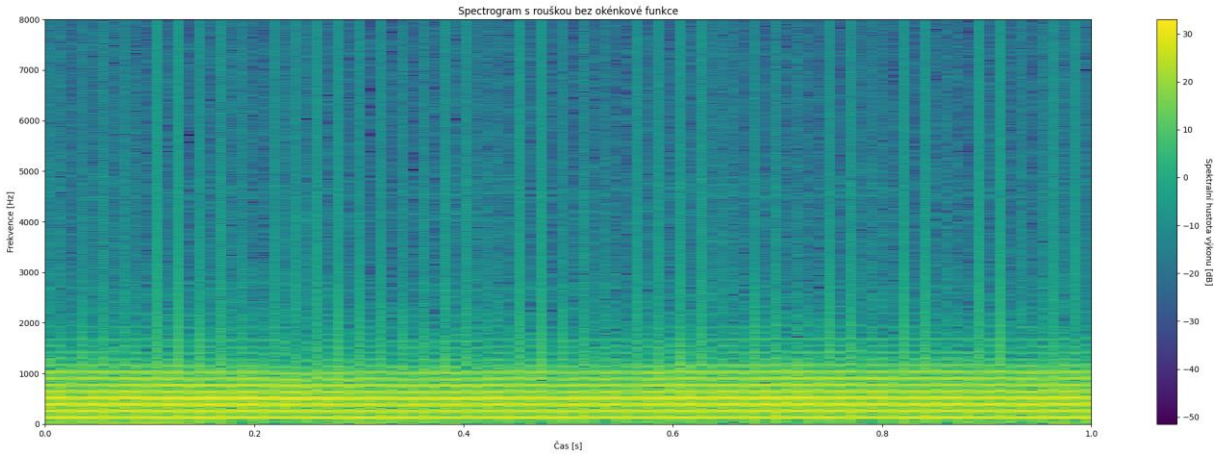
5.

Kód implementující DFT:

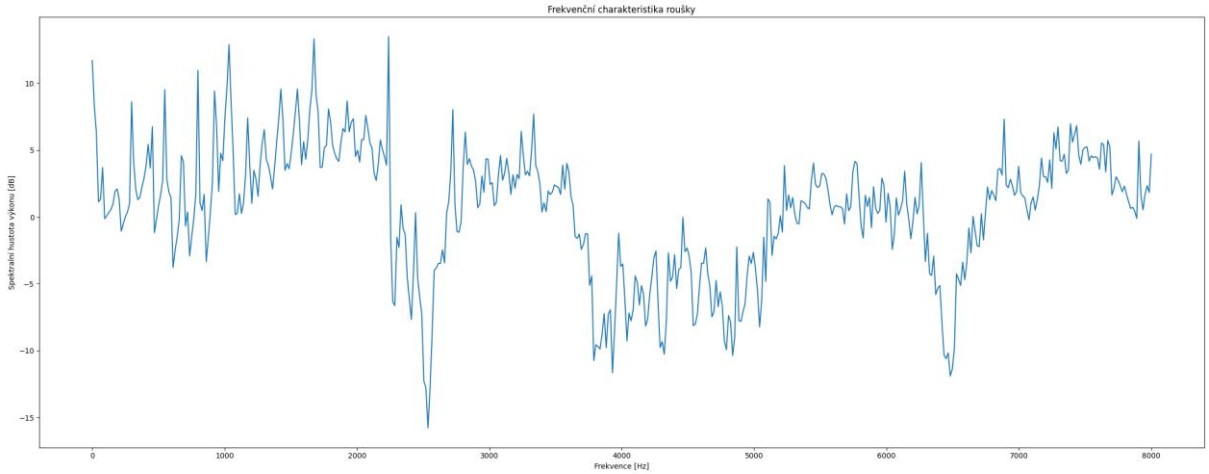
```
def dft(input, padding=0):
    mmax = len(input[0])
    if (mmax < padding):
        mmax = padding
    f = np.empty((0, mmax), np.complex)
    for sample in input:
        f0 = []
        if (len(sample) < padding):
            sample = np.append(sample, np.zeros(padding - len(sample)))
        N = len(sample)
        for p in range(N):
            a = 0j
            for k in range(N):
                a += sample[k] * cmath.exp(-2j *cmath.pi* p * k * (1 / N))
            f0 = np.append(f0, a)
        f = np.append(f, f0.reshape((1, mmax)), axis=0)
    return f
```

Dft funkci jsem navrhl tak, abych jí mohl na vstup dát všechny rámce zároveň a volal ji pouze jednou. Knihovní implementace fft je řádově rychlejší, zpracování fft trvalo několik sekund zatímco zpracovaní dft trvalo pár minut.





6.



$$H(e^{j\omega}) = \frac{B(e^{j\omega})}{A(e^{j\omega})}$$

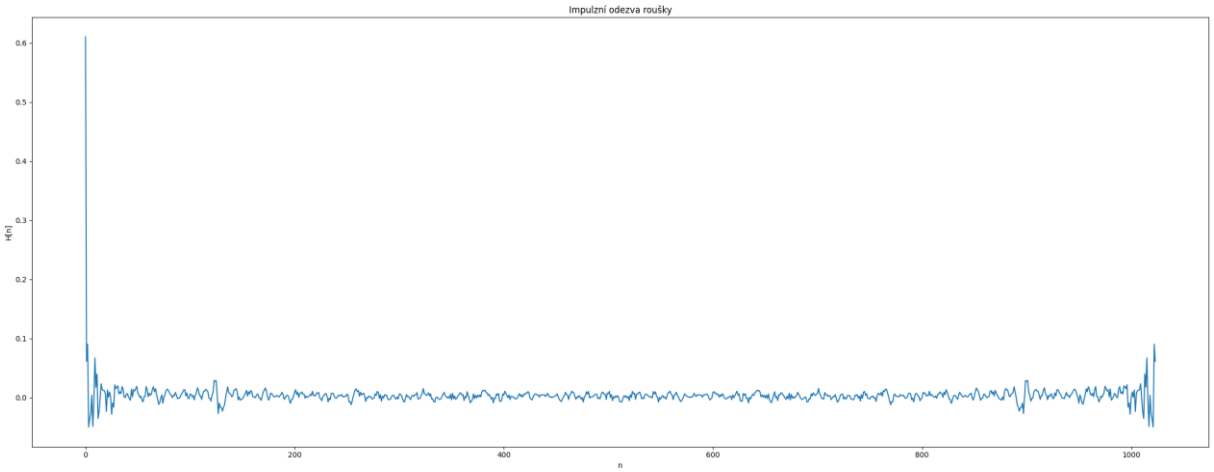
Frekvenční charakteristika filtru sice není nejpřesnější, ale vypadá, že vlivem roušky nastává největší útlum na frekvenci 2.5kHz a harmonických frekvencích. Energie z 2.5 kHz se zřejmě přesunula na nižší frekvence okolo 1.5kHz, kde je vidět značný nárůst výkonu. Filtr se chová jako pásmová zádrž pro frekvence 2.5kHz, 4-5kHz a 6.5 kHz.

7.

```
def idft(input, padding=0):
    input = input.reshape((1,len(input)))
    mmax = len(input[0])
    if (mmax < padding):
        mmax = padding
    f = np.empty((0, mmax), np.complex)
    for sample in input:
        f0 = []
        if (len(sample) < padding):
            sample = np.append(sample, np.zeros(padding - len(sample)))
        N = len(sample)
        for p in range(N):
            a = 0j
```

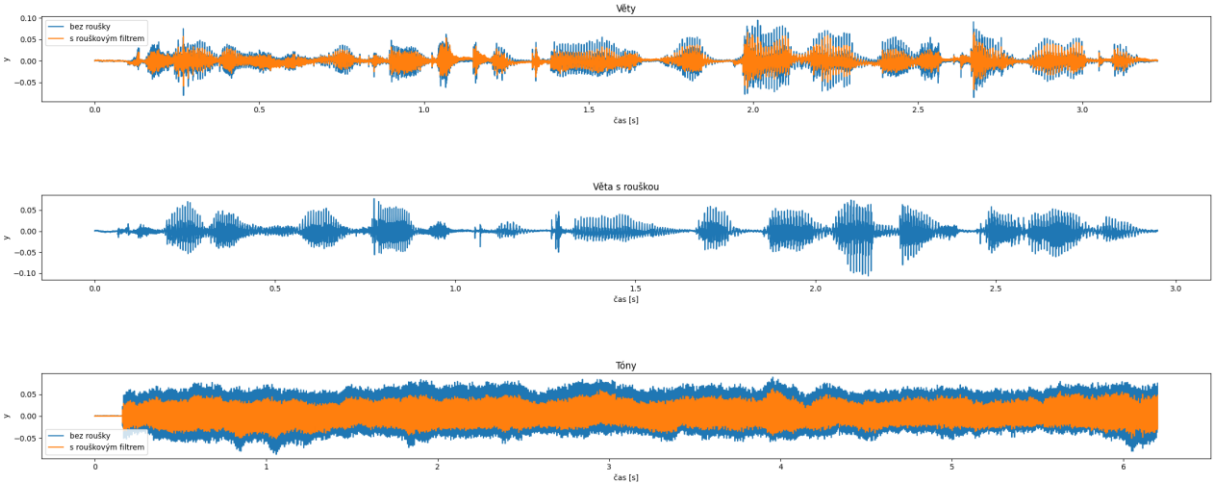
```
            for k in range(N):
                a += sample[k] * cmath.exp(2j * cmath.pi * p * k * (1 / N))
            a /= N
            f0 = np.append(f0, a)
        return np.array(f0)
```

Pro implementaci IDFT jsem musel více pozměnit DFT, jelikož ve vstupu DFT jsem měl 2D pole a zde mám jen 1D pole frekvenční charakteristiky filtru. Zde byla opět knihovní funkce řádově rychlejší.



8.

Věta se simulovanou rouškou vypadá více utlumená pro vyšší amplitudy signálu/nižší frekvence.



9. Závěr

Na poslech zněla simulovaná věta velice podobně s větou bez roušky, jen se mi zdála méně „živější". Filtr zřejmě není perfektní, nejspíše vlivem nedodržení přesné frekvence při nahrávání, ale patrné utlumení signálu lze pozorovat.

10.

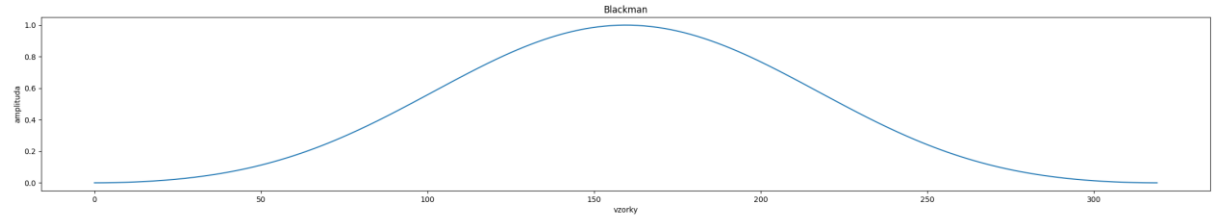
Implementace overlap-add:

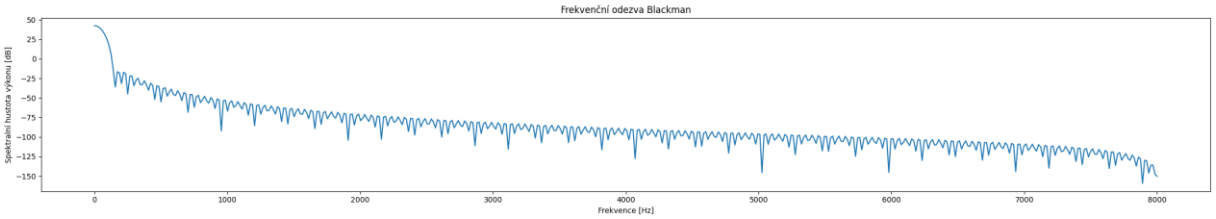
```
M = len(mask_response)
N = 8 * M
step_size = N - M
```

```
H = np.fft.fft(mask_response, N)
position = 0
sentece_filter = np.zeros(len(sentece_of) + M)
while position + step_size <= len(sentece_of):
    sentece_filter[position:position + N] += (
        np.fft.ifft(np.fft.fft(sentece_of[position:step_size + position],
N) * H)).real
    position += step_size
```

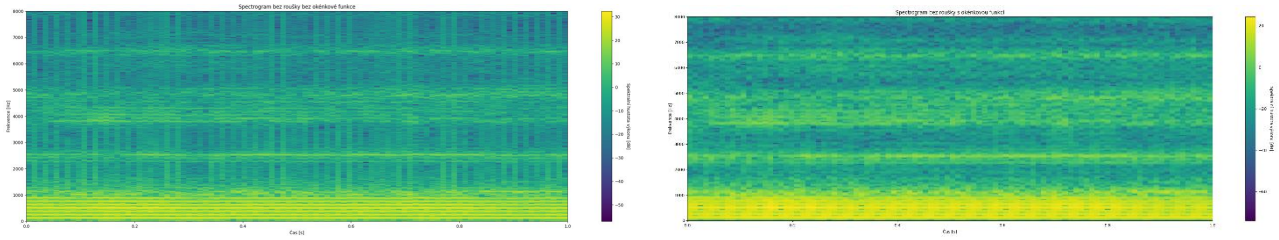
11.

Po vyzkoušní několika okénkových funkcí jsem mezi nimi neslyšel výrazný rozdíl a vybral jsem Blackmanovu funkci. Ze spectrogramu se ukazuje její tlumivý vliv na vyšší (harmonické) frekvence.

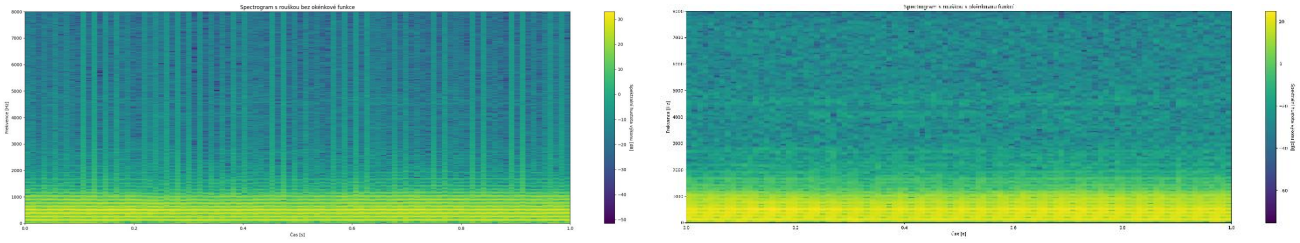




Bez rouška:



S rouškou:

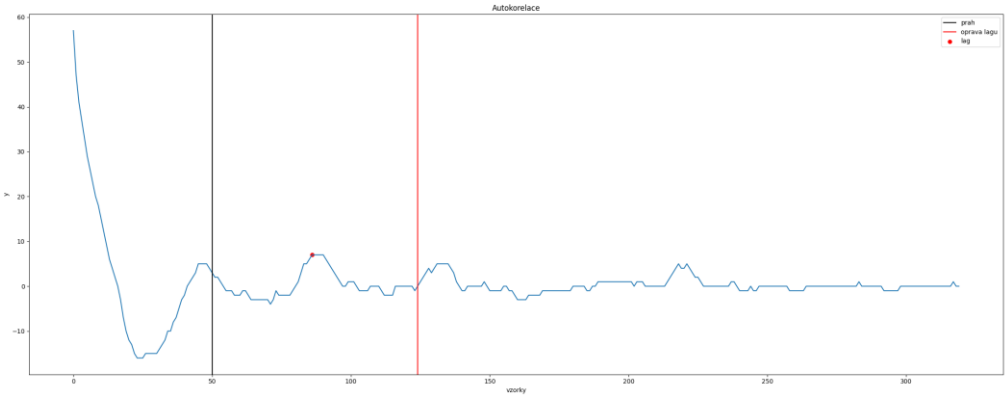


12.

Pro nalezení double lagu jsem musel snížit úroveň klipování na 60% a zároveň vybrat jiný kus nahrávky. N násobný lag jsem odstranil mediánovým filtrem, který byl spočten ze všech lagů. Double lag jsem detekoval naivní metodou, která by ale díky setrvačnosti lidského hlasu měla být pro tento účel dostačující.

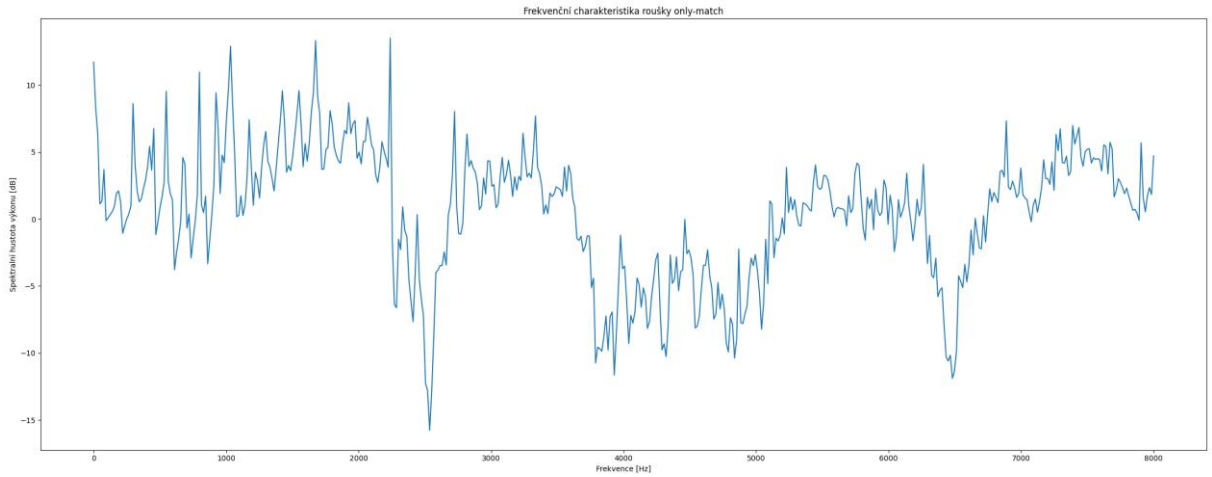
Ladislav Dokoupil, xdokou14

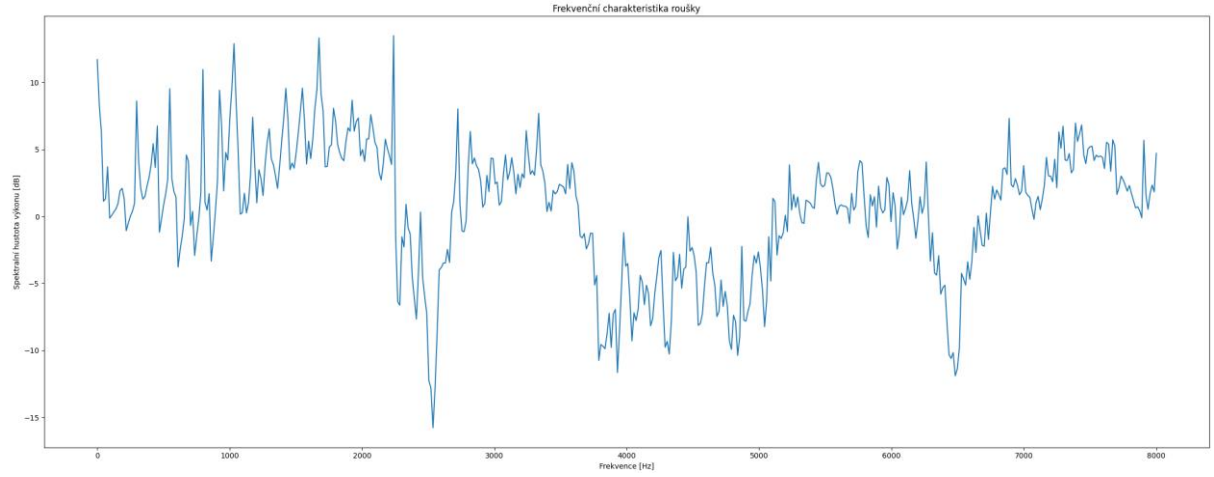
Detekce lagu:
if(np.abs(lags[i] - med) > med / 4) lags[i]=med



13.

Frekvenční charakteristiky se mi zdají být totožné nejspíš díky dobrému prvotnímu výběru úseku nahrávky.

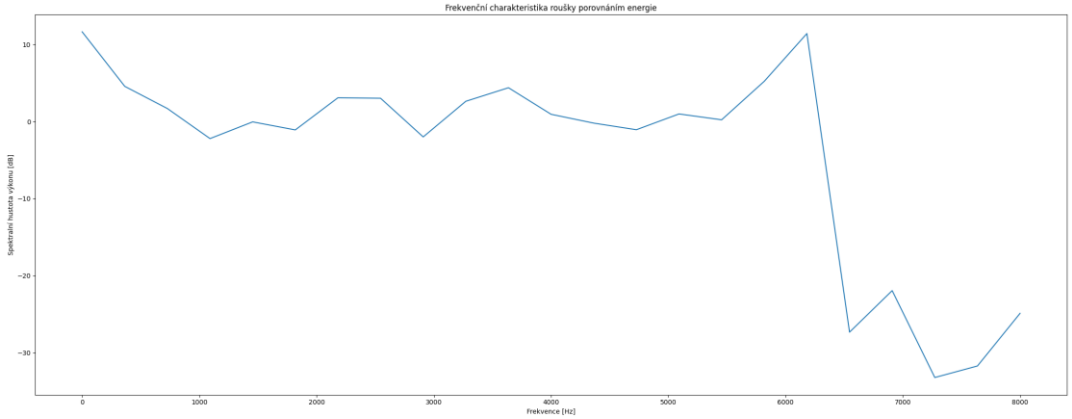




14.

Banku filtru jsem použil ze stránek ISS. Charakteristika roušky porovnáním energie velice hrubě připomíná skutečnou charakteristiku, ale dokáže dát odhad, jak zhruba má skutečný filtr vypadat. Předpokládám, že při použití více dolních propustí by se podobnost se skutečnou charakteristikou zvětšovala.

Ladislav Dokoupil, xdokou14



15.