Richard Seipel
xseipe00

Protokol k projektu ISS

1. a 2.

	Dĺžka v sekundách	Počet vzoriek
maskoff_tone.wav	9,059	144945
maskon_tone.wav	10,170	162719
maskoff_sentence.wav	2,480	39684
maskon_sentence.wav	4,528	72440

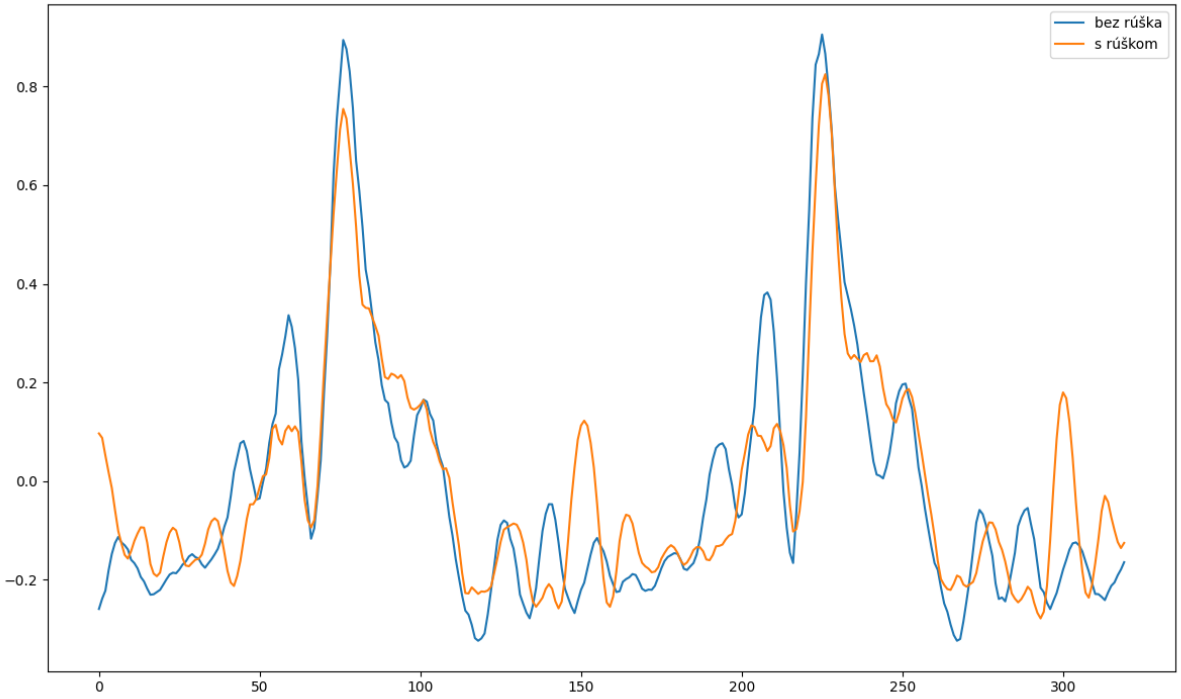
3.

Vzorec na výpočet rámca:

```
offtone_seg_frames = [] # pole rámcov
velkost_ramca = int(0.020 * fs) # 20ms * vzorkovacia frekvencia
prekryv_ramca = int(0.010 * fs) # 10ms * vzorkovacia frakvencia
posun = velkost_ramca - prekryv_ramca # posun v poli segmentu

# iterácia po "dĺžku poľa - posun", pre nezapočítanie posledného
polovičného rámca
for i in range(0, len(offtone_seg) - posun, posun):
    offtone_seg_frames.append(offtone_seg[i:i + velkost_ramca]) #
rámec obsahuje hodnoty v intervale (i, i + veľkosť rámca)
```

Graf rámcov:



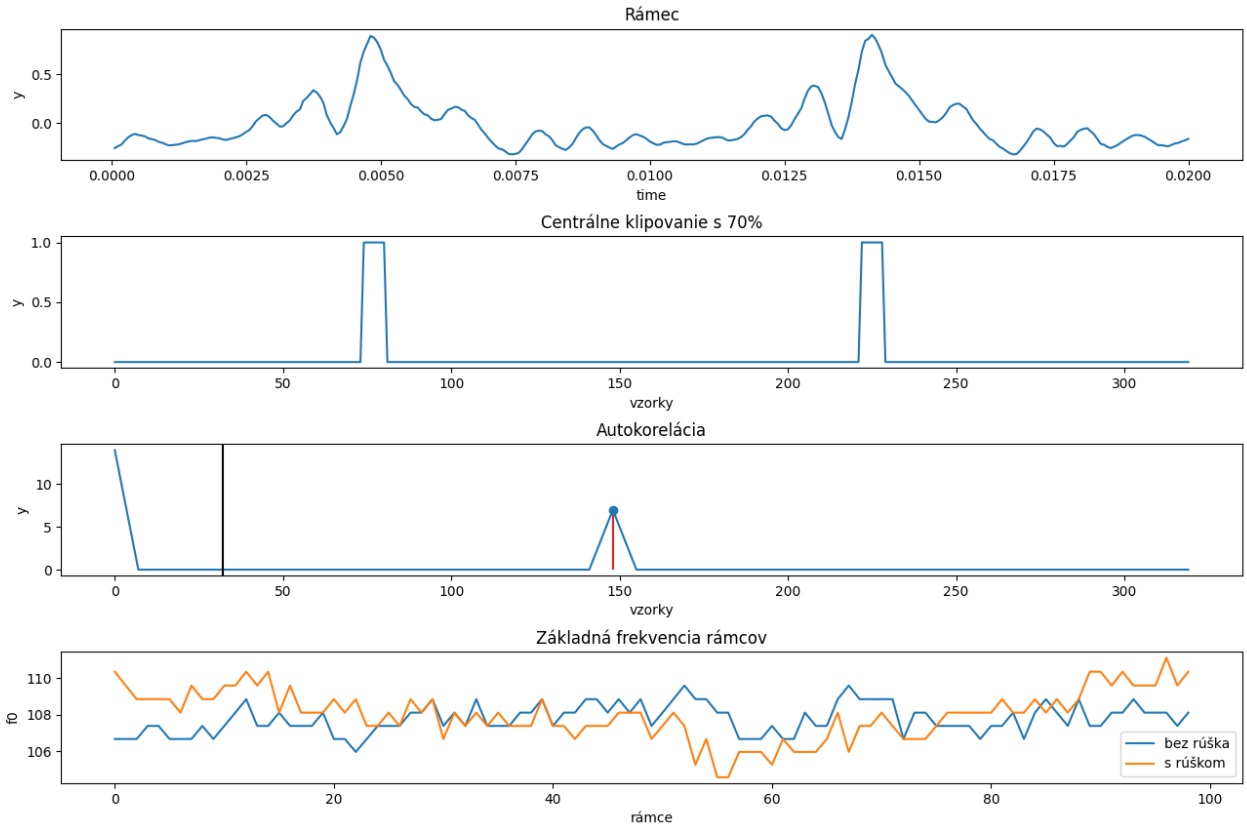
Richard Seipel
xseipe00

Protokol k projektu ISS

4.

a)

Grafy klipovania, autokorelácie a základnej frekvencie rámcov:



b)

Tabuľka základnej frekvencie rámcov:

	Stredná hodnota	Rozptyl
Bez rúška	107,761	0,622
S rúškom	108,022	1,795

c)

Popis problému s veľkou zmenou frekvencie pri zmene lagu o 1:

Pri základnej frekvencii signálu okolo 108 Hz to nie je veľmi poznať, no pri vyšších frekvenciách je veľká zmena vo frekvencii spôsobená malým počtom vzoriek signálu. Napríklad pri základnej frekvencii 800 Hz je perióda signálu zastúpená len 20-timi vzorkami, to spôsobí že pri zmene lagu na 21 vzoriek je základná frekvencia už len 761 Hz. Veľkosť tejto zmeny frekvencie sa dá znížiť, zvýšením vzorkovacej frekvencie.

Richard Seipel
xseipe00

Protokol k projektu ISS

5.

a)

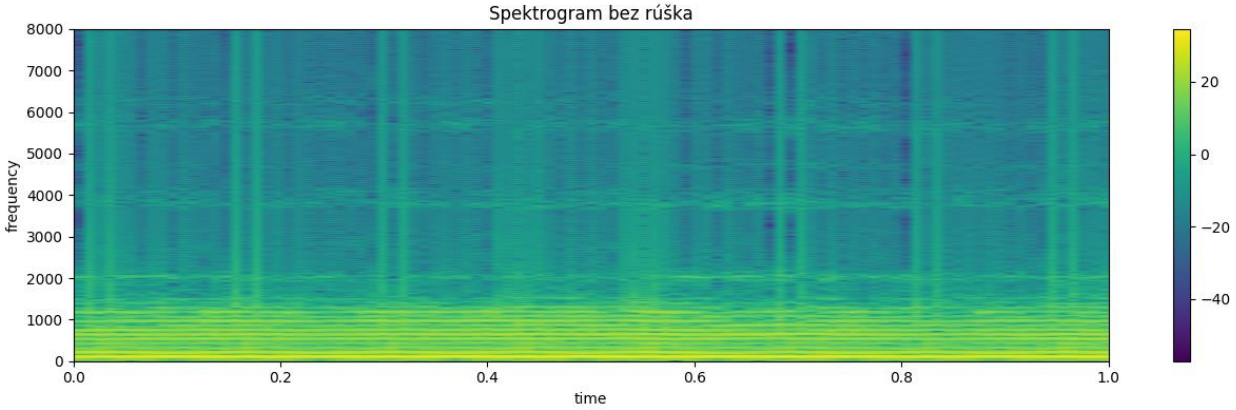
Vlastná implementácia DFT:

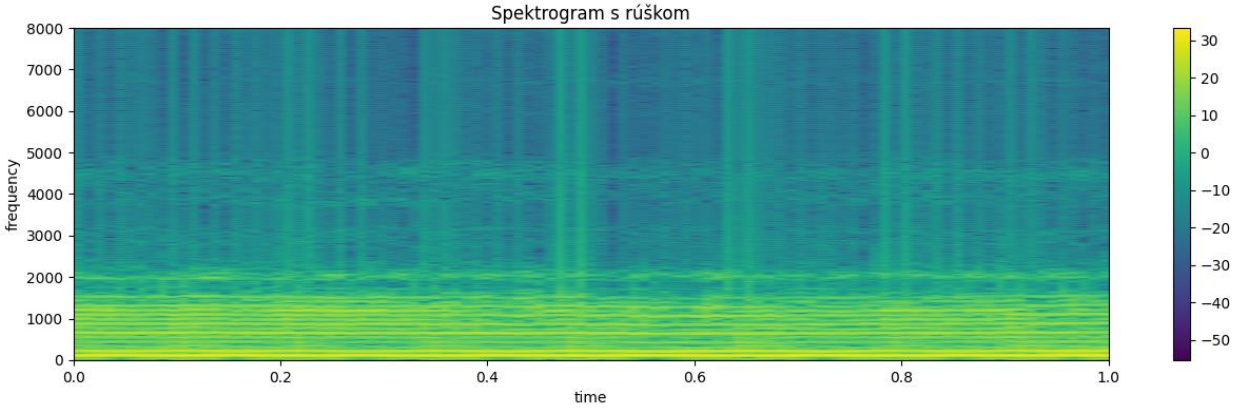
```
#dft
def manualdft(array, N):
    result = []
    padded_array = np.pad(array, N - array.size, 'constant', constant_values=0)
    for n in range(N):
        sum_k = 0.0
        for k in range(N):
            sum_k += padded_array[k] * cmath.exp(complex(0, -n*k*2*math.pi/N))
        result.append(sum_k)
    return result
```

Porovnanie tejto implemetácie s funkciou *np.fft.ftt* je zobrazené v druhom grafe súboru *dft-spectrum.py*. Na pohľad sú identické, no veľký rozdiel je v rýchlosti výpočtu.

b)

Porovnanie spektrogramov s rúškom a bez rúška:





Richard Seipel
xseipe00

Protokol k projektu ISS

6.

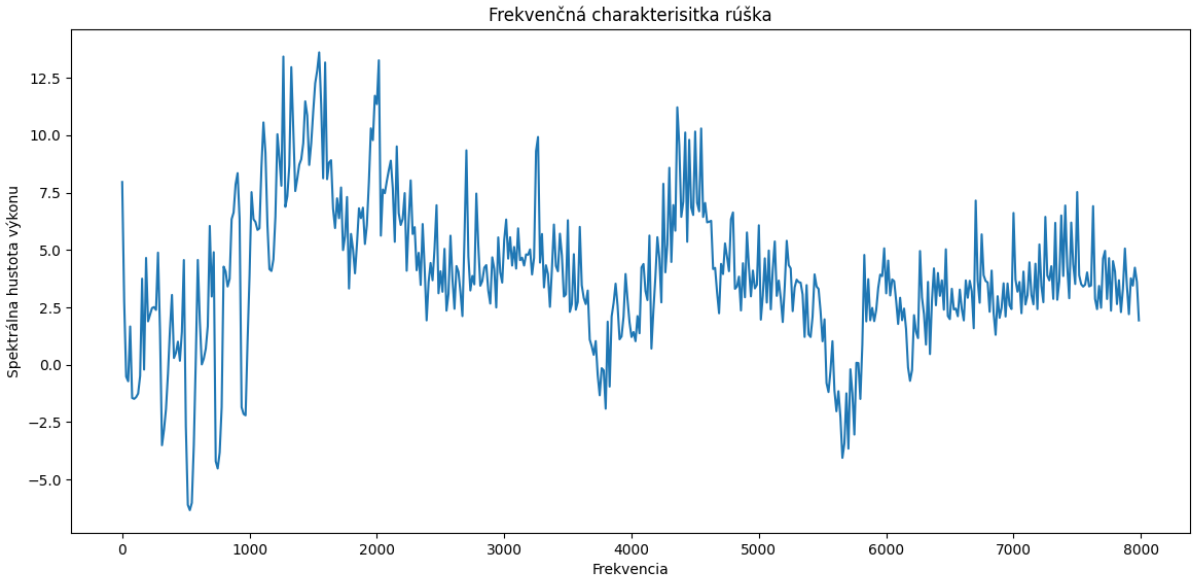
a)

Vzorec na výpočet frekvenčnej charakteristiky:

$$H(e^{j\omega}) = \frac{Y(e^{j\omega})}{X(e^{j\omega})}$$

b)

Frekvenčná charakteristika rúška:



c)

V tomto projekte je filtrom rúško, je typu FIR. Jeho frekvenčnú charakteristiku som vypočítal, ako priemer frekvenčných charakteristík jednotlivých rámcov v absolútnej hodnote. Frekvenčnú charakteristiku pre každý rámec som vypočítal, podelením spektra daného rámca s rúškom, spektrom tohto rámca bez rúška, ktoré boli vypočítané pomocou DFT v predchádzajúcej úlohe.

Richard Seipel
xseipe00

Protokol k projektu ISS

7.

a)

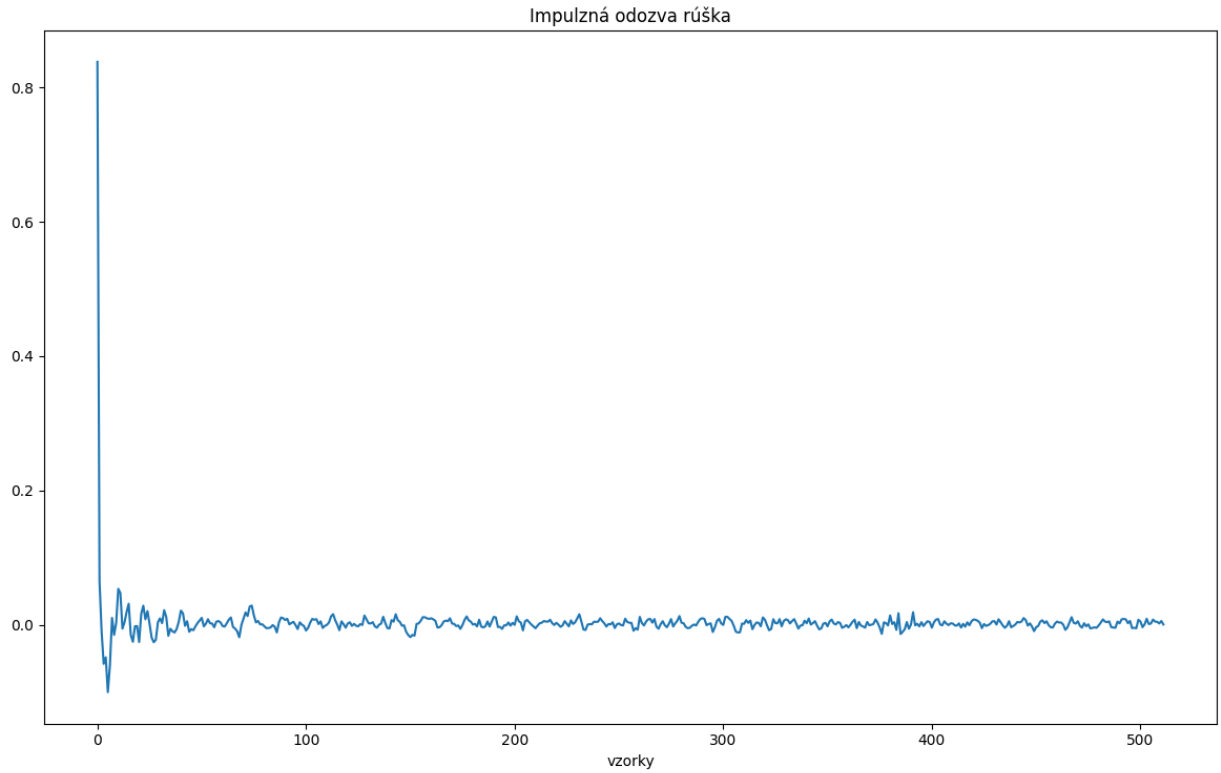
Vlastná implementácia IDFT:

```
#dft
def manualidft(array, N):
    result = []
    padded_array = np.pad(array, N - array.size, 'constant', constant_values=0)
    for k in range(N):
        sum_n = 0.0
        for n in range(N):
            sum_n += padded_array[n] * cmath.exp(complex(0, n*k*2*math.pi/N))
        result.append(sum_n / N)
    return result
```

Porovnanie implementácii je zobrazené v druhom grafe súboru *idft-response.py*. Na pohľad vidno, že moja implementácia má „hustejšie“ zobrazenie odozvy, pravdepodobne spôsobené zero-paddingom, no nepodarilo sa mi to odstrániť. Ako pri DFT v predchádzajúcej úlohe, je aj tu značný rozdiel v rýchlosti výpočtu.

b)

Impulzná odozva rúška:



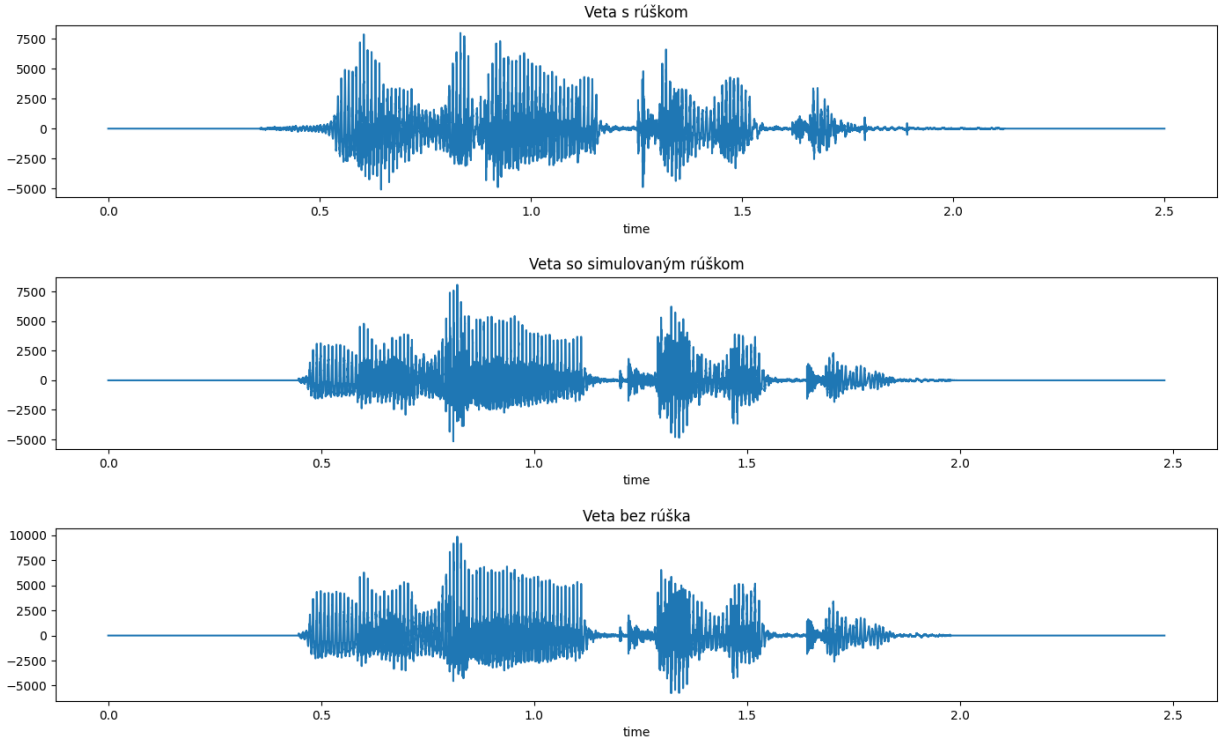
Richard Seipel
xseipe00

Protokol k projektu ISS

8.

a)

Grafy výsledných viet:



b)

Výsledný filter v simulácii rúška nie je veľmi počuť, aj keď v grafe je vidieť celkové stlmenie nahrávky a priblíženie k amplitúde nahrávky s rúškom. Taktiež je vidieť výraznejšie stlmenie v čase po prvej sekunde a pred polovicou druhej sekundy a po priblížení nahrávky je vidieť aj malé zmeny na tvare vlny. Keďže podobne znejú už pôvodné nahrávky, podobajú sa aj nahrávky s rúškom a simulovaným rúškom, no väčšia podobnosť je medzi nahrávkou so simulovaným rúškom a nahrávkou bez rúška.

9.

Záver

Na záver by som dodal, že funkcia filtra nie je najlepšia a určite by sa na nej ešte dalo pracovať. Môže to byť spôsobené nepresným výberom najpodobnejších úsekov nahrávok, nedokonalosťou nahrávok samotných, či mojej implementácie. Na iné nedokonalosti riešenia som nenarazil.

Richard Seipel
xseipe00

Protokol k projektu ISS - doplňujúce úlohy

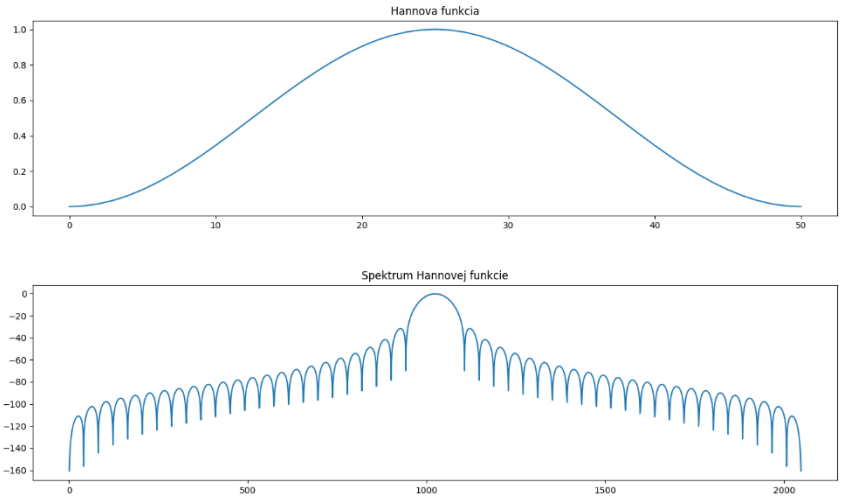
11.

a)

Použil som Hannovu funkciu okna. Využíva hore posunutý kosínus, preto je známa aj ako „raised cosine“. Je jednou z najpoužívanejších.

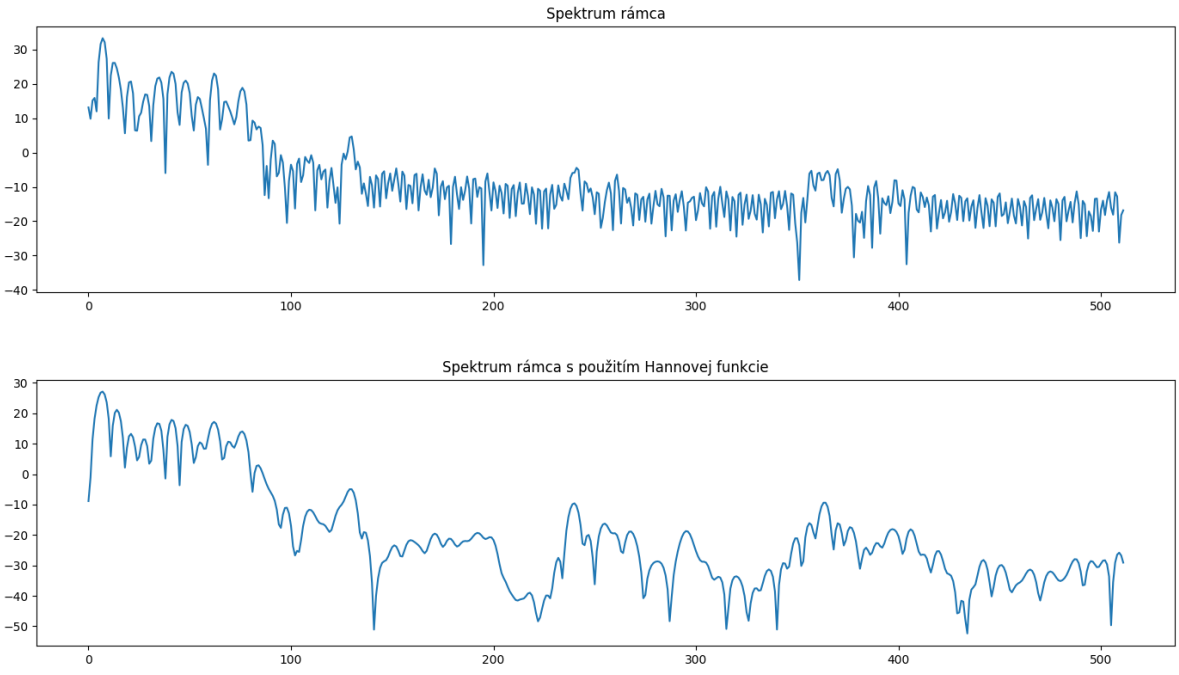
b)

Graf Hannovej funkcie a jej spektrum:



c)

Porovnanie spektier s funkciou okna a bez nej:



Použitie tejto funkcie je užitočné na utlmenie okrajov rámca.

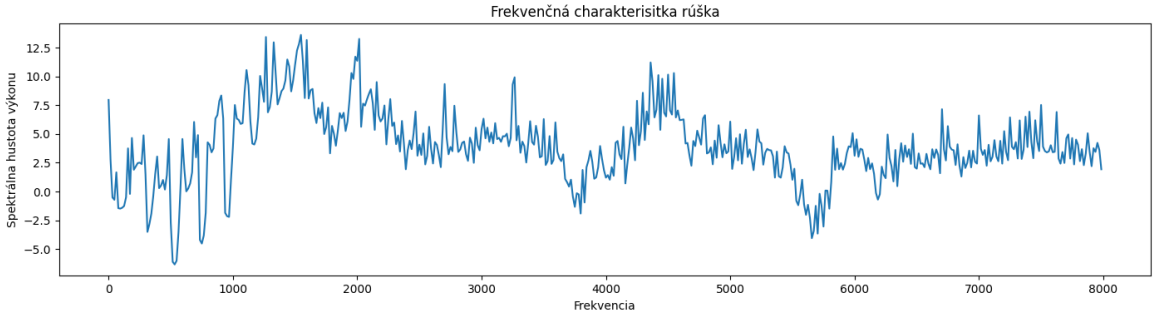
Richard Seipel
xseipe00

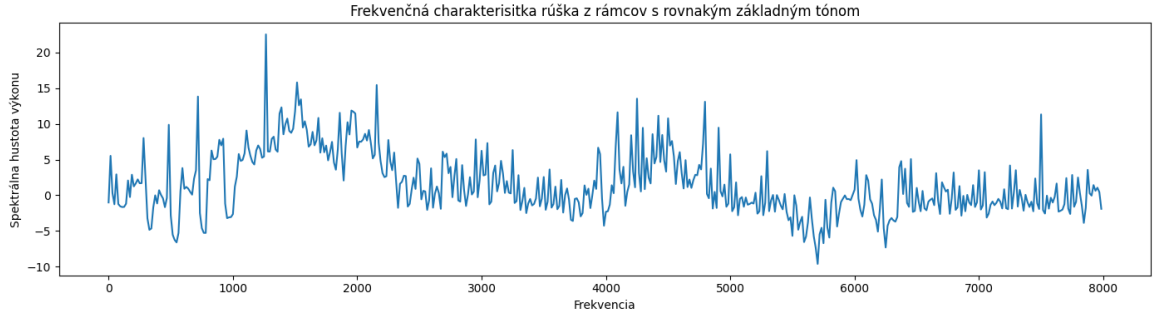
Protokol k projektu ISS - doplňujúce úlohy

V súbore *filtering_window.py* je dostupný graf vety so simulovaným rúškom (podobný ako v úlohe 8.) po aplikovaní funkcie okna. Tento graf sa viac podobá grafu vety s reálnym rúškom, no na nahrávke stále nie je počuť výraznejšie zmeny. Tieto nahrávky sú v priečinku audio s príponou window.

13.

Porovnanie frekvenčnej charakteristiky zo všetkých rámcov a frekvenčnej charakteristiky len z rámcov s rovnakým základným tónom:





V priečinku audio sú pridané nahrávky s upravenou frekvenčnou charakteristikou s príponou *only_match*. Opäť po porovnaní grafu (poslený graf v súbore *freq_characteristic_same.py*) vidno značné zmeny v tvare vlny, no pri prehraní zvuku to nie je výrazné.

Záver

Záver ostáva rovnaký ako pri hlavnej časti. Aj keď tieto doplňujúce úlohy robili výrazné zmeny v grafoch, nepodarilo sa mi dosiahnuť aby nahrávka so simulovaným rúškom znela ako nahrávka rúškom reálnym. Možno by pomohlo použiť rúško s väčším vplyvom na nahrávaný zvuk.