1. Úloha

Nahrávky som nahral pomocou aplikácie Audacity, kde som takisto už nastavil vzorkovaciu frekvenciu, mono kanál a 16 bitov na 1 vzorku.

Názov	Dĺžka nahrávky v sekundách	Dĺžka nahrávky vo vzorkách
maskoff_tone	6,67	106640
maskon_tone	8,73	139760

2. Úloha

Nahrávky som takisto nahral pomocou Audacity a nastavil tam potrebné parametre.

Názov	Dĺžka nahrávky v sekundách	Dĺžka nahrávky vo vzorkách
maskoff_sentence	3,40	54320
maskon_sentence	3,21	51440

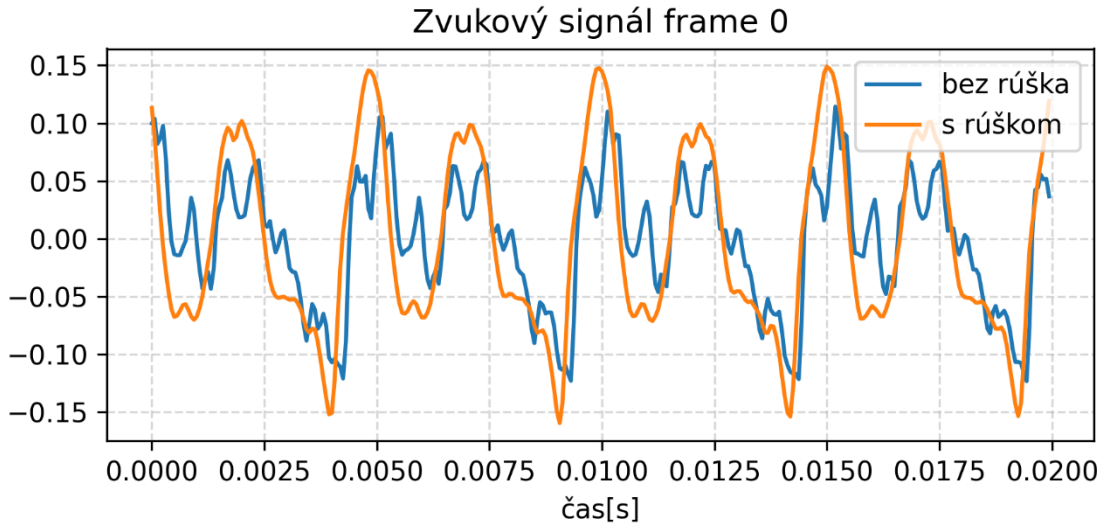
3. Úloha

Na prečítanie nahrávok som použil knižnicu soundfile, keďže má výhodu oproti ostatným a to je automatická normalizácia. Potom som prečítané časti nahrávok ustrednil.

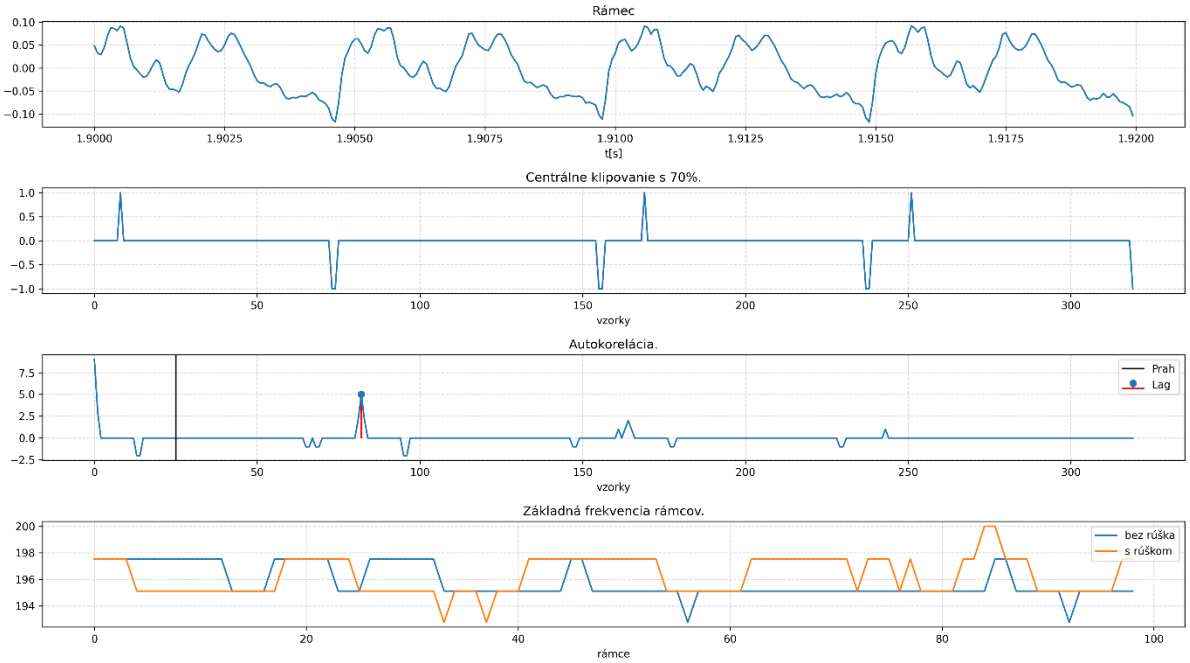
Vzorec:

$$\boldsymbol{ramec = vzorkovacia\ frekvencia * velkost\ ramca}$$
$$\boldsymbol{ramec = 16000 * 0,02 = 320}$$

Graf:



4. Úloha



stredná hodnota mask_off = 195.8044325889691
rozptyl mask_off = 1.4043101123169637
stredná hodnota mask_on = 196.2679658438303
rozptyl mask_on = 1.9166772294343444

Veľkosť zmeny f0 pri chybe ± 1 by sa dala zmenšiť pomocou zväčšenia vzorkovacej frekvencie.

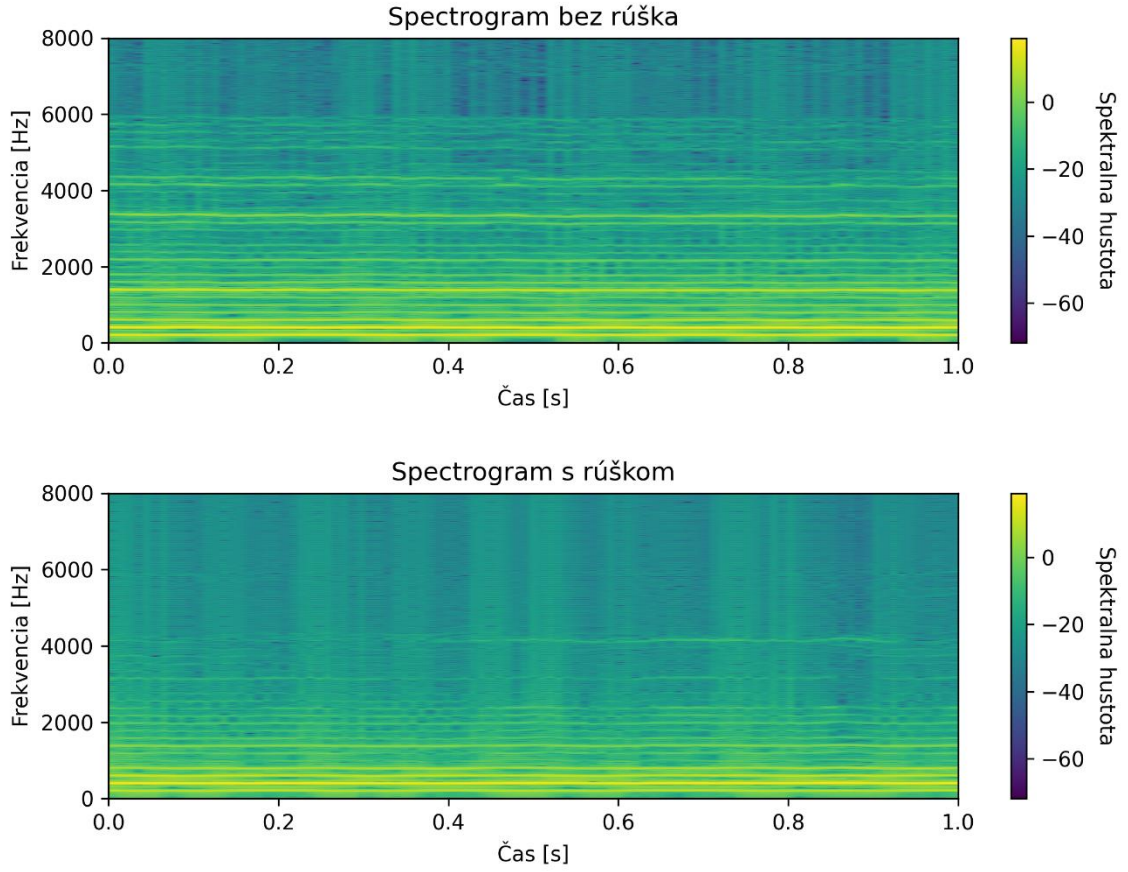
5. Úloha

Implementácia DFT:
Čerpal som z dokumentácie numpy.

```
def dft(frame, lenght):
    t = np.zeros(lenght, dtype = 'complex_')
    N = len(frame)
    while(N < lenght):
        N +=1
        frame = np.append(frame, 0)

    for k in range(N):
        for n in range(N):
            t[k] += frame[n]*cmath.exp(-2j*cmath.pi*k*n*(1/N))
    return t
```

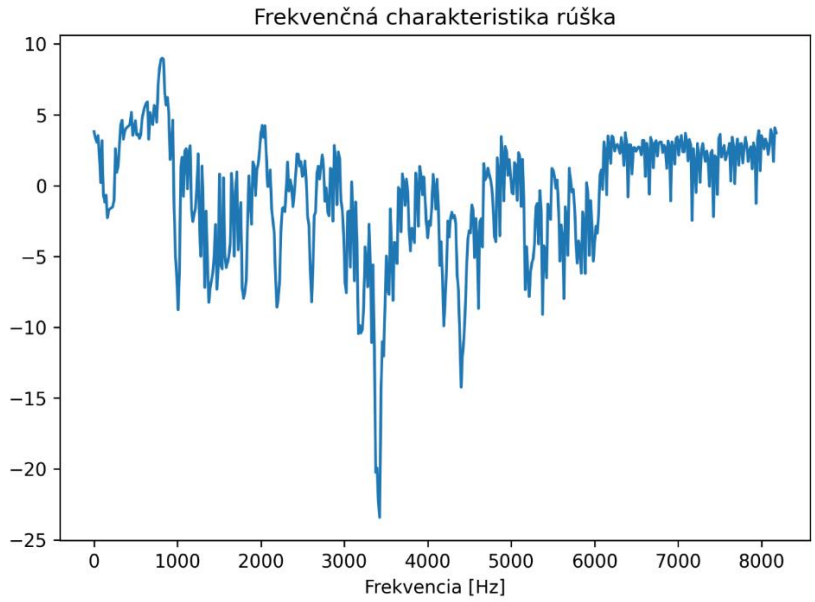
Spektogramy:



6. Úloha

Vzťah:

$$H(e^{jw}) = \frac{B(e^{j\omega})}{A(e^{j\omega})} = \frac{b[0] + b[1]e^{-j\omega*1} + b[2]e^{-j\omega*2} + \cdots + b[M]e^{-j\omega*M}}{a[0] + a[1]e^{-j\omega*1} + a[2]e^{-j\omega*2} + \cdots + a[N]e^{-j\omega*N}}$$



Rúškový filter v mojom prípade filture najmä frekvencie od približne 1000Hz do približne 6200Hz.

7. Úloha

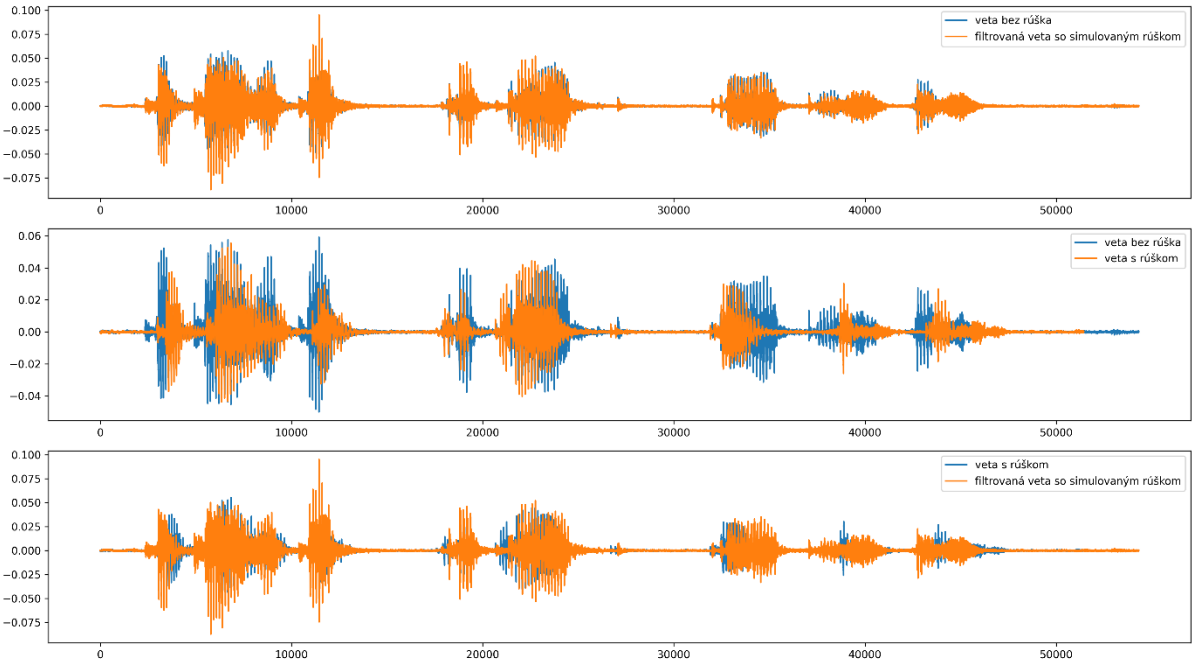
Implementácia IDFT:
Takisto ako pri DFT som čerpal z dokumentácie numpy.

```
def idft(frame, lenght):
    x = np.zeros(lenght, dtype = 'complex_')
    N = len(frame)
    while(N < lenght):
        N +=1
        frame = np.append(frame, 0)

    for n in range(N):
        for k in range(N):
            x[n] += frame[k]*cmath.exp(2j*cmath.pi*k*n*(1/N))
        x[n] /= N
    return x
```

Graf impulznej odozvy:

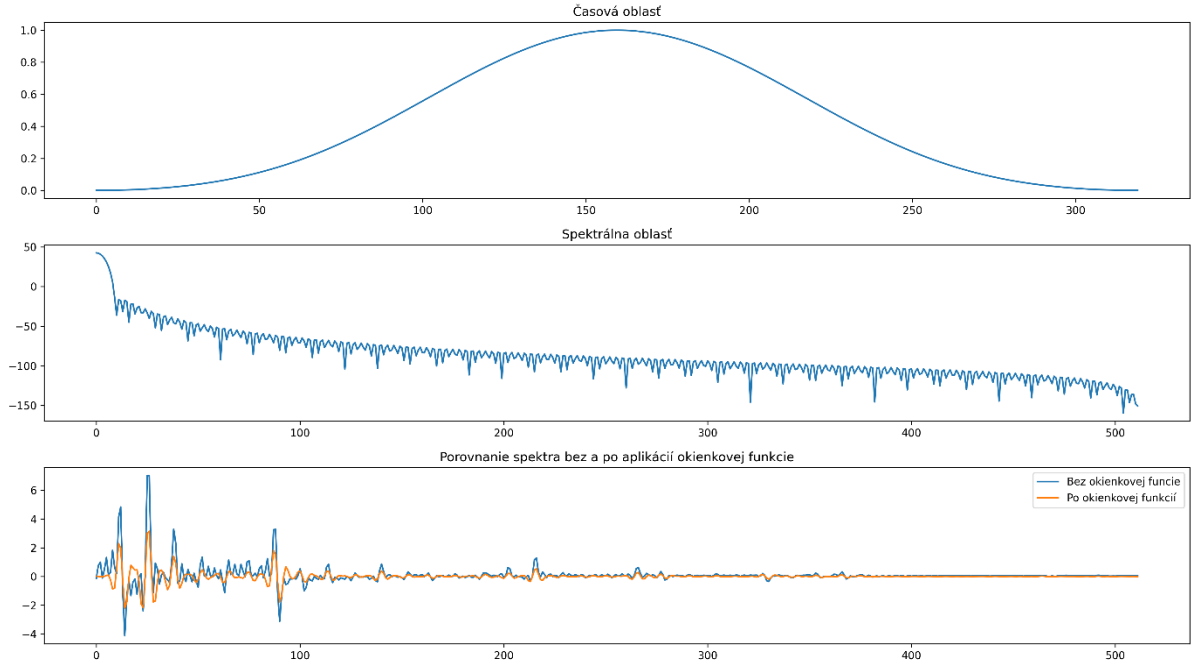
8. Úloha



Moje vety bez rúšky a s rúškom rozhodne nie sú dokonalé, je to kvôli "ľudskej" chybe, keďže 2 úplne identické tóny sa mi nepodarili nahrať. Povedal by som však že spĺňajú to, čo som očakával. Veta so simulovaným rúškom sa podľa mňa dosť podobá na vetu s rúškom a myslím si že simulované rúško funguje celkom správne.

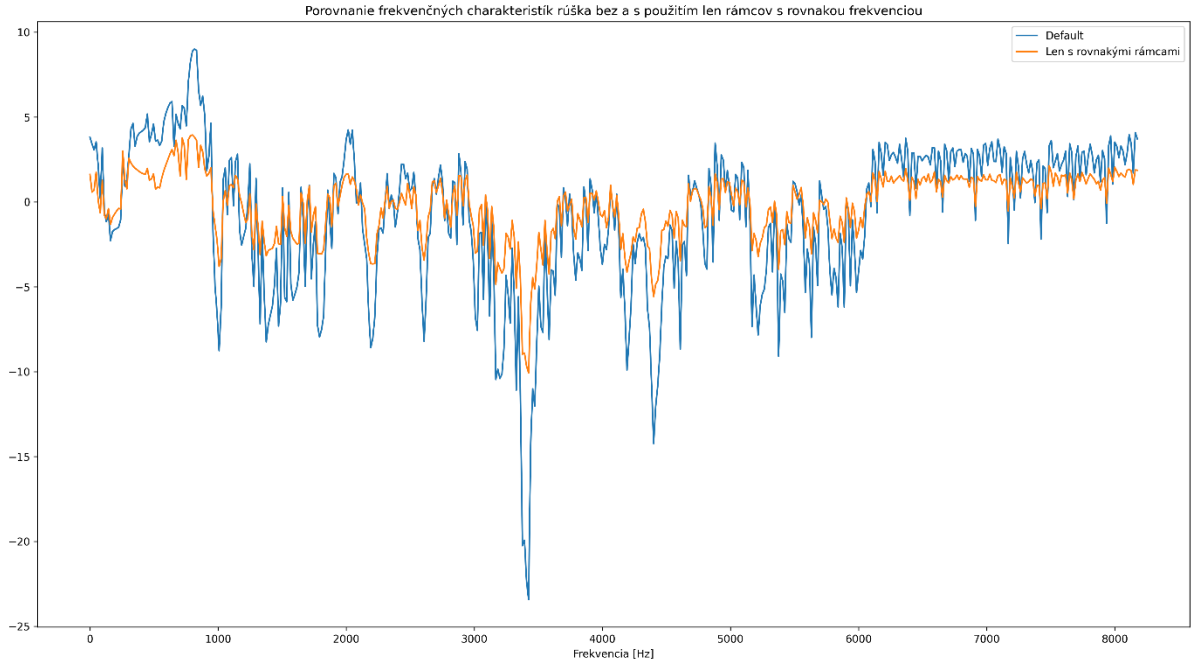
11. Úloha

Ako okienkovú funkciu som použil blackman.



Okienková funckia slúži na vyčistenie signálu od vonkajších "cudzích" vplyvov a teda je po jej použitý signál čistejší.

13. Úloha



Ak použijeme rámce len s rovnakým záladným tónom, tak nám frekvenčná charakteristika vyjde menej rozptýlená a celkovo sa jej amplitúda zmenší.

9. Záver

Výsledky môjho projektu by som ohodnotil ako dostačujúce, priam až dobré. Myslím si že tóny a vety som nahral najlepšie ako som vedel. Takisto si myslím že chyby ktoré z toho vyplynuli by mohli byť ovplivnené už len použitím iného postupu aký sme použili, alebo použitím iných, presnejších, nahrávok.