Projekt

Signály a systémy

Protokol

Jakub Bartko (xbartk07)
xbartk07@stud.fit.vutbr.cz

Fakulta informačních technologií
Vysoké učení technické v Brne
20. 12. 2020

1. Tabuľka nahrávok tónov

Nahrávka	**Dĺžka** [*vzorky*]	**Dĺžka** [*s*]
maskoff_tone.wav	71 680	4,48
maskon_tone.wav	64 000	4,00

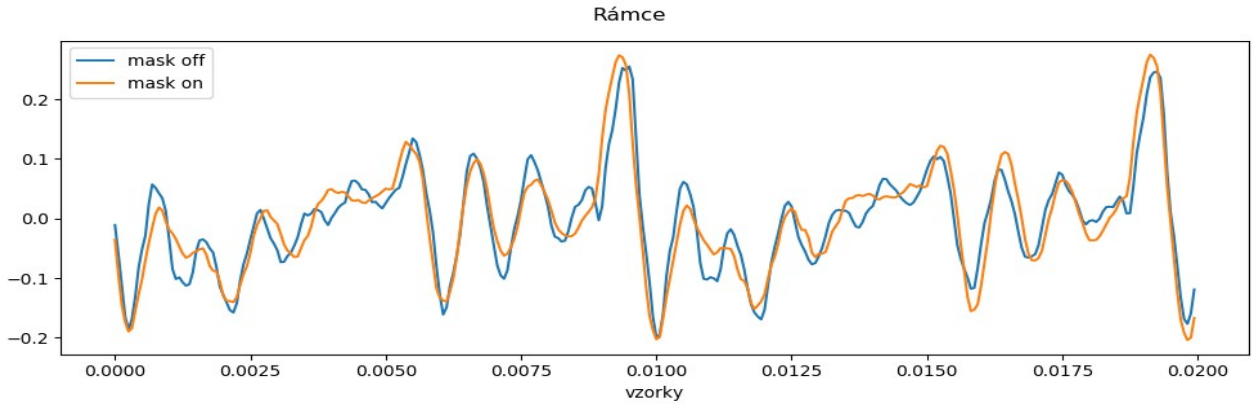
2. Tabuľka nahrávok viet

Nahrávka	**Dĺžka** [*vzorky*]	**Dĺžka** [*s*]
maskoff_sentence.wav	61 440	3,84
maskon_sentence.wav	69 120	4.32

3. a) Vzorec na výpočet veľkosti rámca vo vzorkách

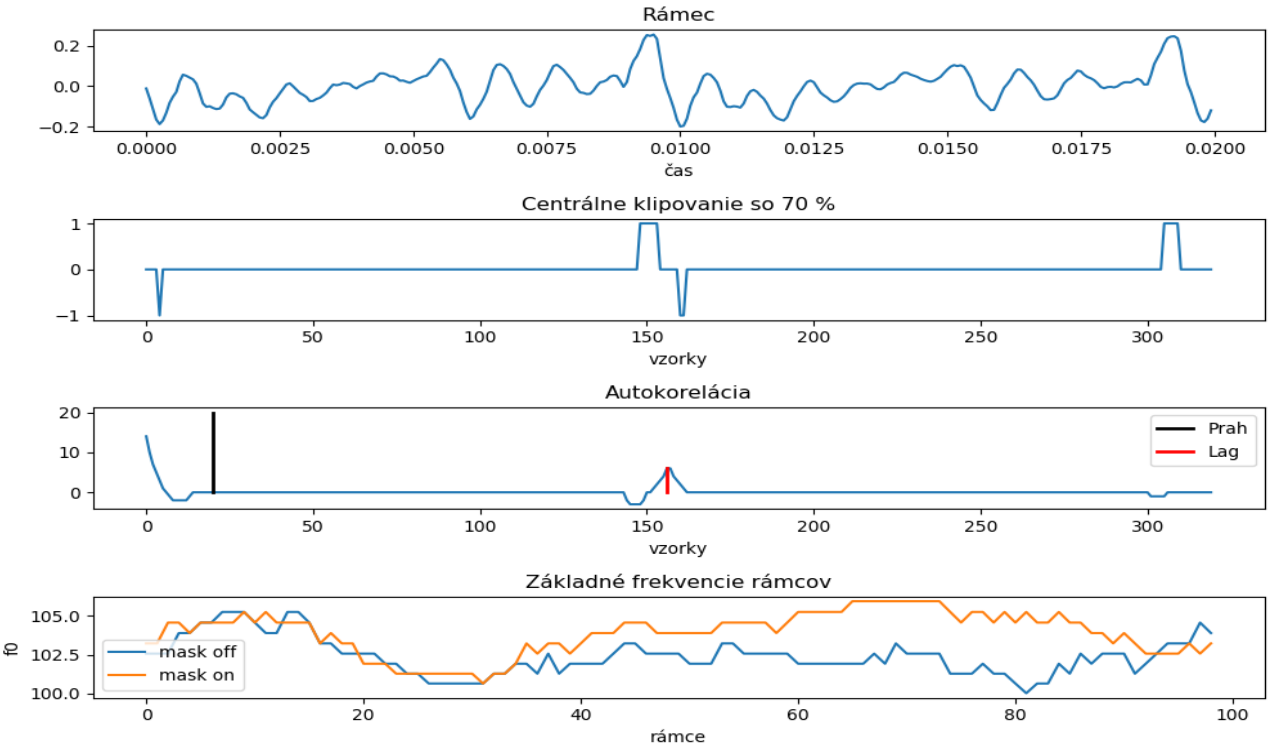
$$veľkosť\,[vzorky] = veľkosť\,[s] * frekvencia$$

b) Graf príkladu rámcov bez rúšky a s ňou



Pozn. Úseky boli zvolené na základe jednoduchej cross-korelácie s využitím iterácie viacerými úsekmi z nahrávky prvej, ktoré boli korelované s úsekmi z nahrávky druhej. Zvolená bola dvojica s najväčším koeficientom.

4. a) Graf získavania základných frekvencií rámcov



b) Stredné hodnoty a rozptyly základných frekvencií

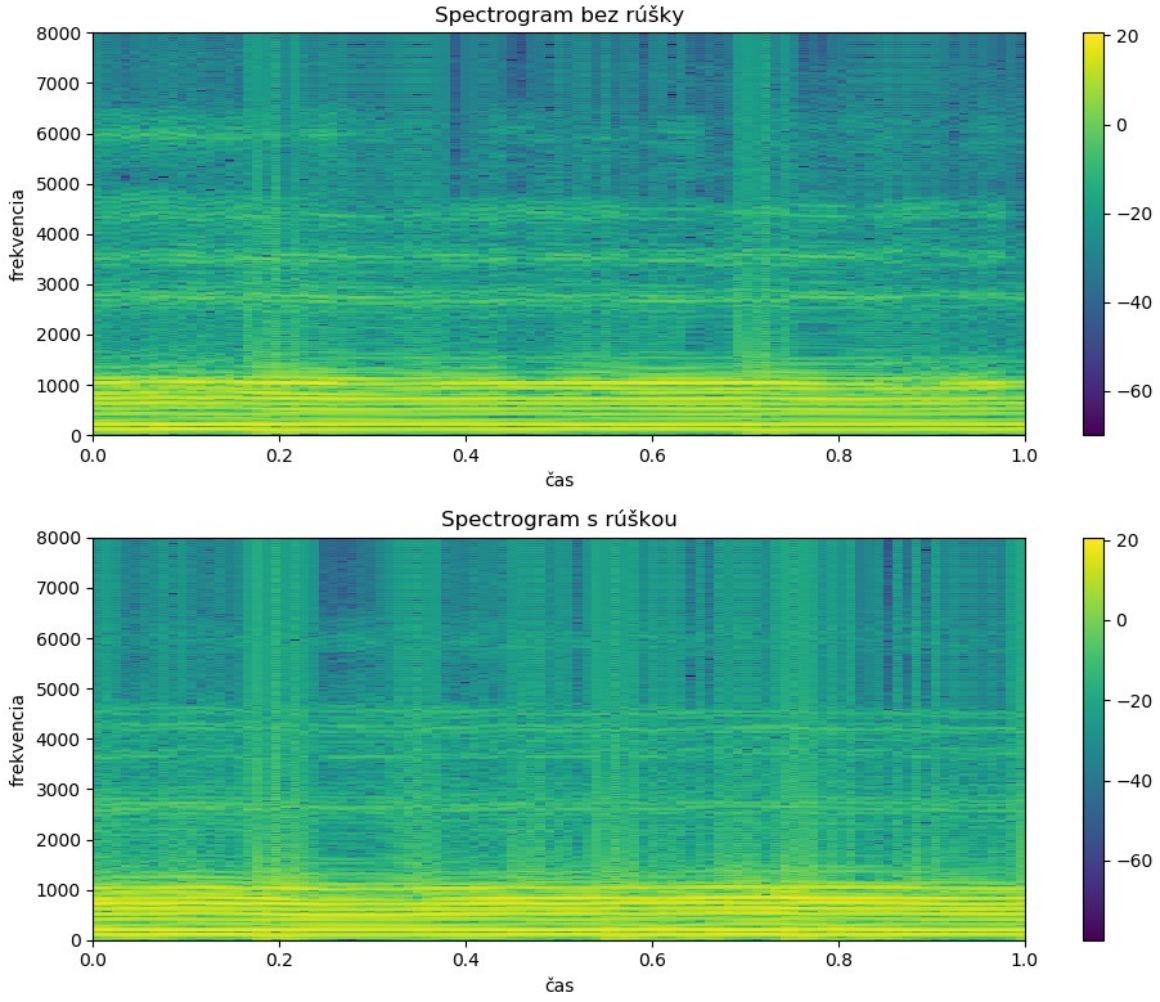
Nahrávka	Stredná hodnota	Rozptyl
maskoff_tone.wav	102,4120	1,4102
maskon_tone.wav	103,8874	1,9088

c) Veľkosť zmeny f_0 pri chybe lagu ±1 by bolo možné zmenšiť znížením (ideálnejšie obmedzením) jeho váhy na výslednú frekvenciu, napr. vyvážením hodnoty lagu s hodnotami pre rámce v jeho okolí.

5. a) Implementácia DFT

```
def dft(x_in, n):
    x = np.zeros(n, dtype=complex)
    x[:x_in.size] = x_in
    X = np.zeros(n, dtype=complex)

    X = [sum((x[k] * np.exp(np.complex(0, -2 * math.pi * i * k / n))
                for k in range(n)))
                    for i in range(n)]
    return X
```

b) Spektrogramy pre tón bez rúšky a s ňou

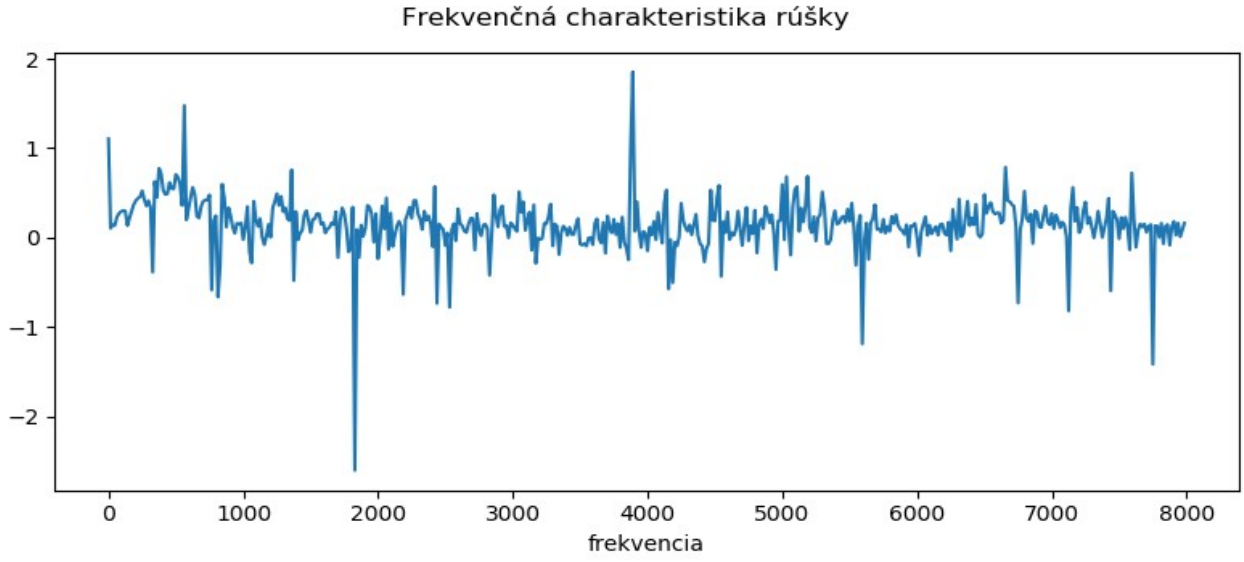


6. a) Vzorec na výpočet frekvenčnej charakteristiky rúšky

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

Y – spektrum s súškou
X – spektrum bez rúšky

b) Frekvenčná charakteristika rúšky

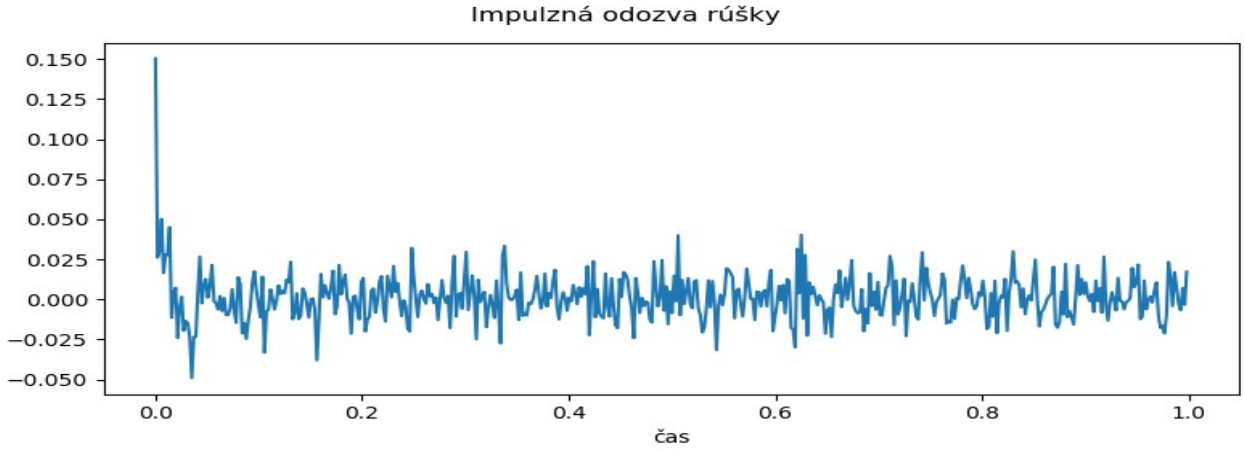


c) Zdá sa, že tento filter potláča okrajové časti spektra ľudskej reči, hlavne okolo hornej hranice (200 – 300 Hz)

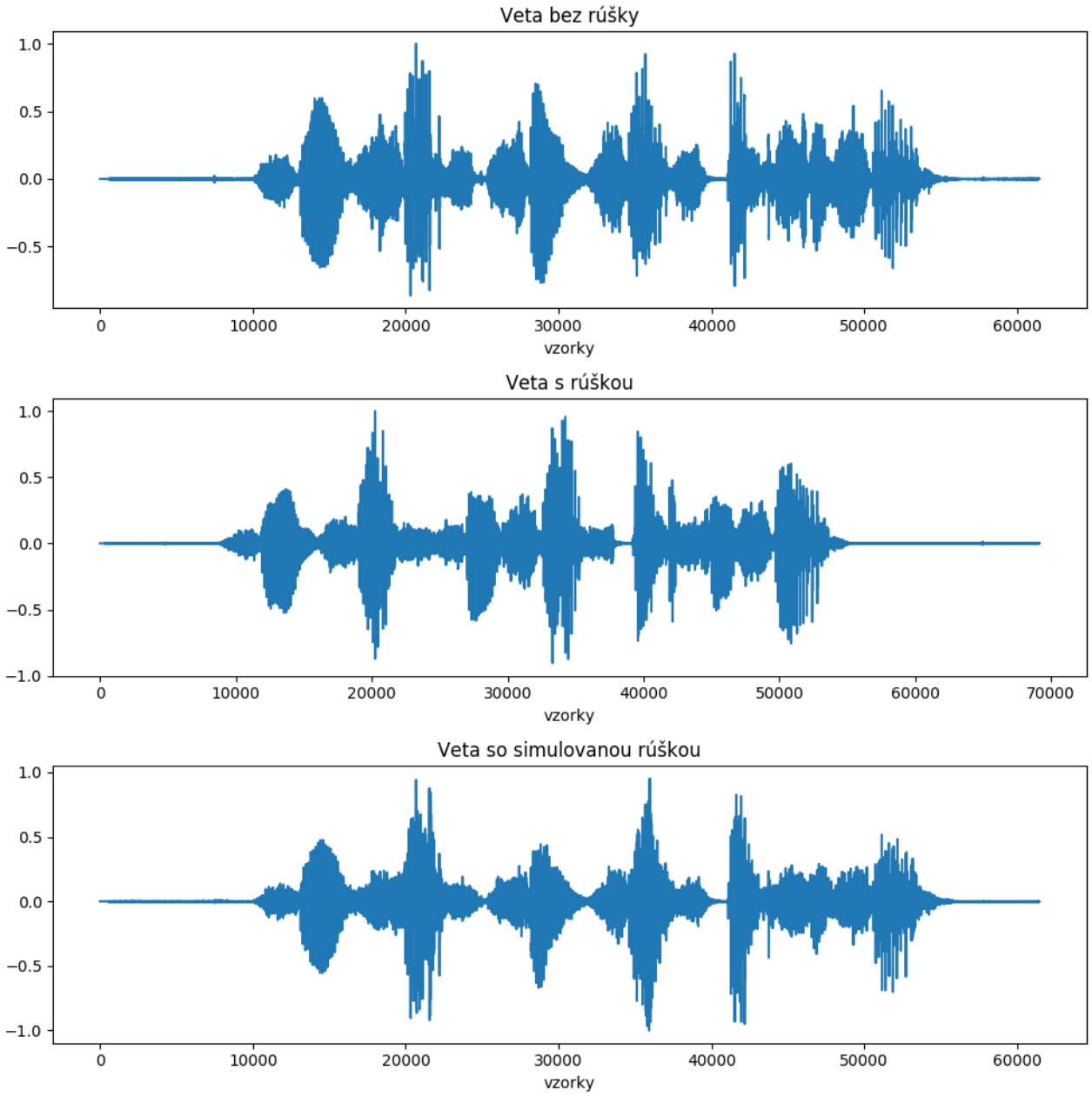
7. a) Implementácia IDFT

```
def idft(x):
    N = x.size
    K = x[0].size
    X = np.zeros(N, dtype=complex)

    X[:-1] = [sum((x[k] * np.exp(np.complex(0, 2 * math.pi * k * N / n))
          for k in range(K)))
             for n in range(1, N)]

    return X / N
```

b) Impulzná odozva



8. a) Porovnanie vety bez rúšky, s ňou, a s nasimulovanou rúškou



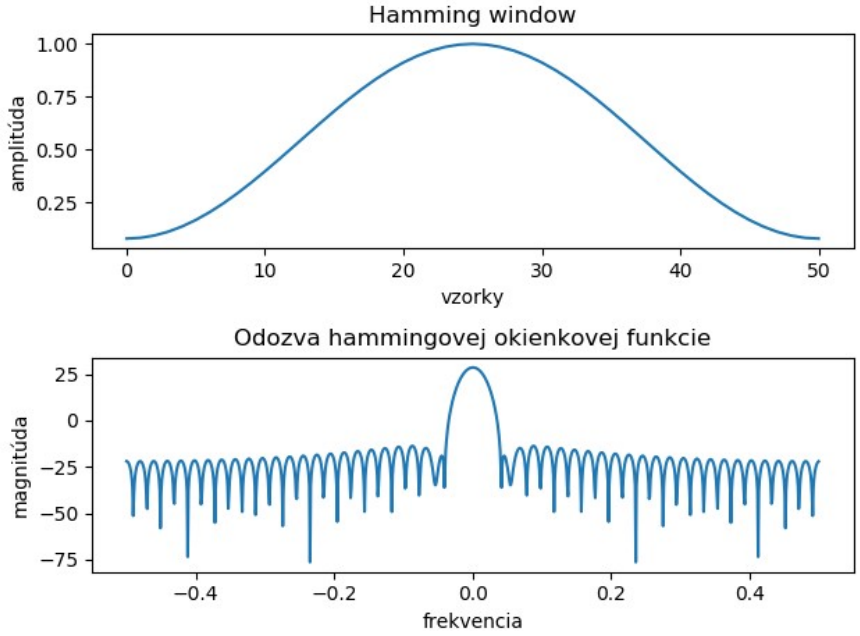
b) Signály s rúškou a nasimulovanou rúškou sú si podobné najmä v miestach s nižšou frekvenciou. Zdá sa, že simulovaná rúška v týchto miestach signál správne utlmuje. Vyššie frekvencie však utlmuje až príliš, alebo ich naopak nechá v skoro pôvodnej podobe.

10. Implementácia metódy overlap-add

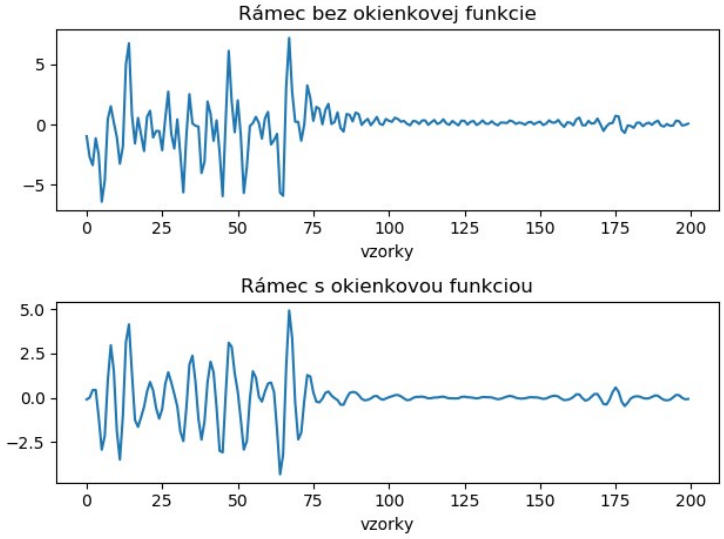
```
def overlapAdd(x, h, N):
    M = 1024
    size = x[0].size
    step = int(size / 2)
    out = np.zeros(step * N + M, dtype=complex)
    H = np.fft.fft(h, M)

    for i in range(0,N):
        pos = step * i
        ramec = np.zeros(M, dtype=complex)
        ramec[:size] = x[i]
        f = np.fft.ifft(np.fft.fft(ramec, M) * H)
        out[pos:pos+M] = out[pos:pos+M] + f

    return out
```

11. a) V implementácií bola využitá Hammingova okienková funkcia

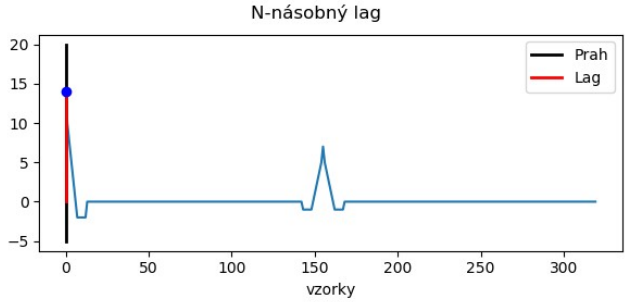
b) Hammingova okienková funkcia



c) Využitie okienkovej funkcie zníži veľkosť zmien - skokov v spektrách rámcov (t. j. ich vyhľadí), čím sa zlepší "čírosť" (z angl. *clarity*) signálu



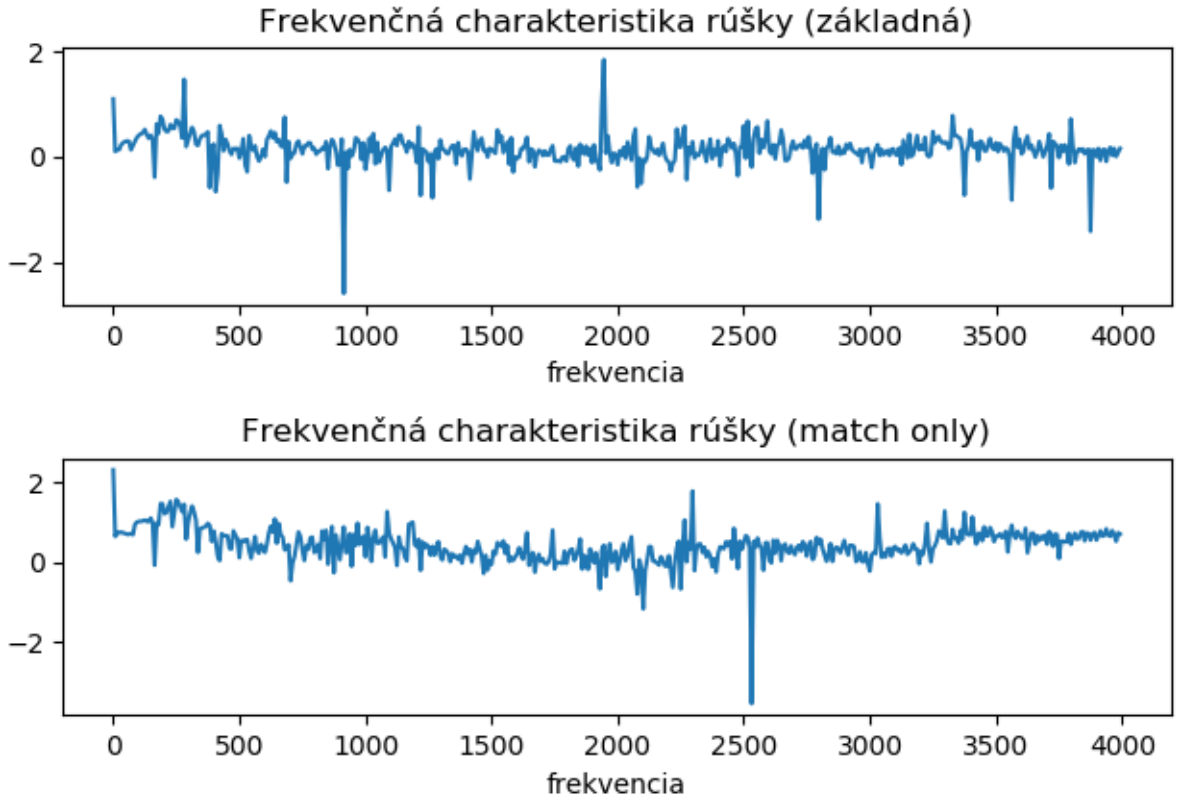
12. a) Graf korelačných koeficientov rámca s využitým n-násobným lagom



Pozn. Prah pre lag bol odstránený na ukážku možnej chyby pri jeho získavaní (koeficient blízko nuly, namiesto požadovaného koeficientu, ktorý by zodpovedal základnej frekvencií)

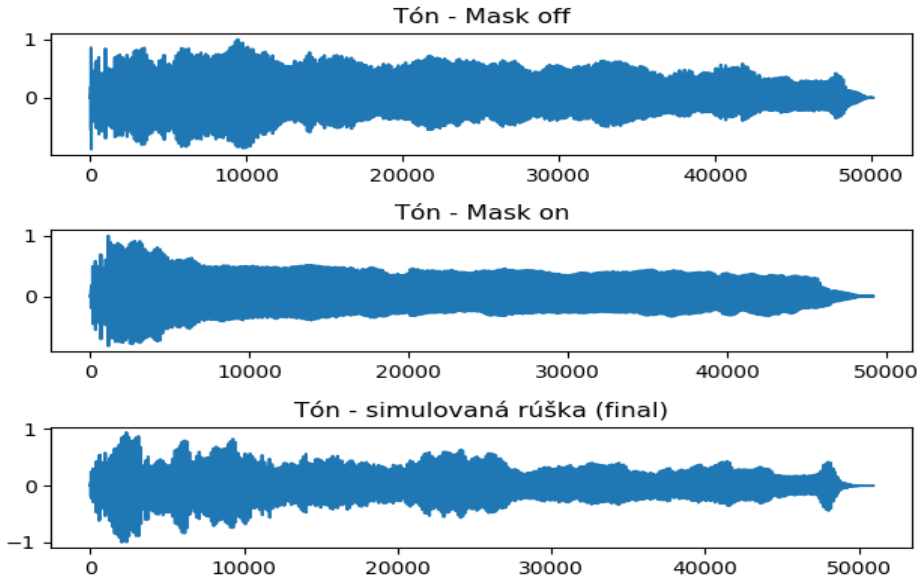
b) Chyba bola odstránená využitím mediánového filtra: nová hodnota lagu je mediánom 5 hodnôt okolitých lagov (2 susedných rámcov a rámca samotného); ktorý je v implementácií obmedzený na výraznejšie rozdiely medzi mediánom a pôvodnou hodnotou lagu. Tým sa vzniklá chyba "vyhladí" vzhľadom na svoje okolie.

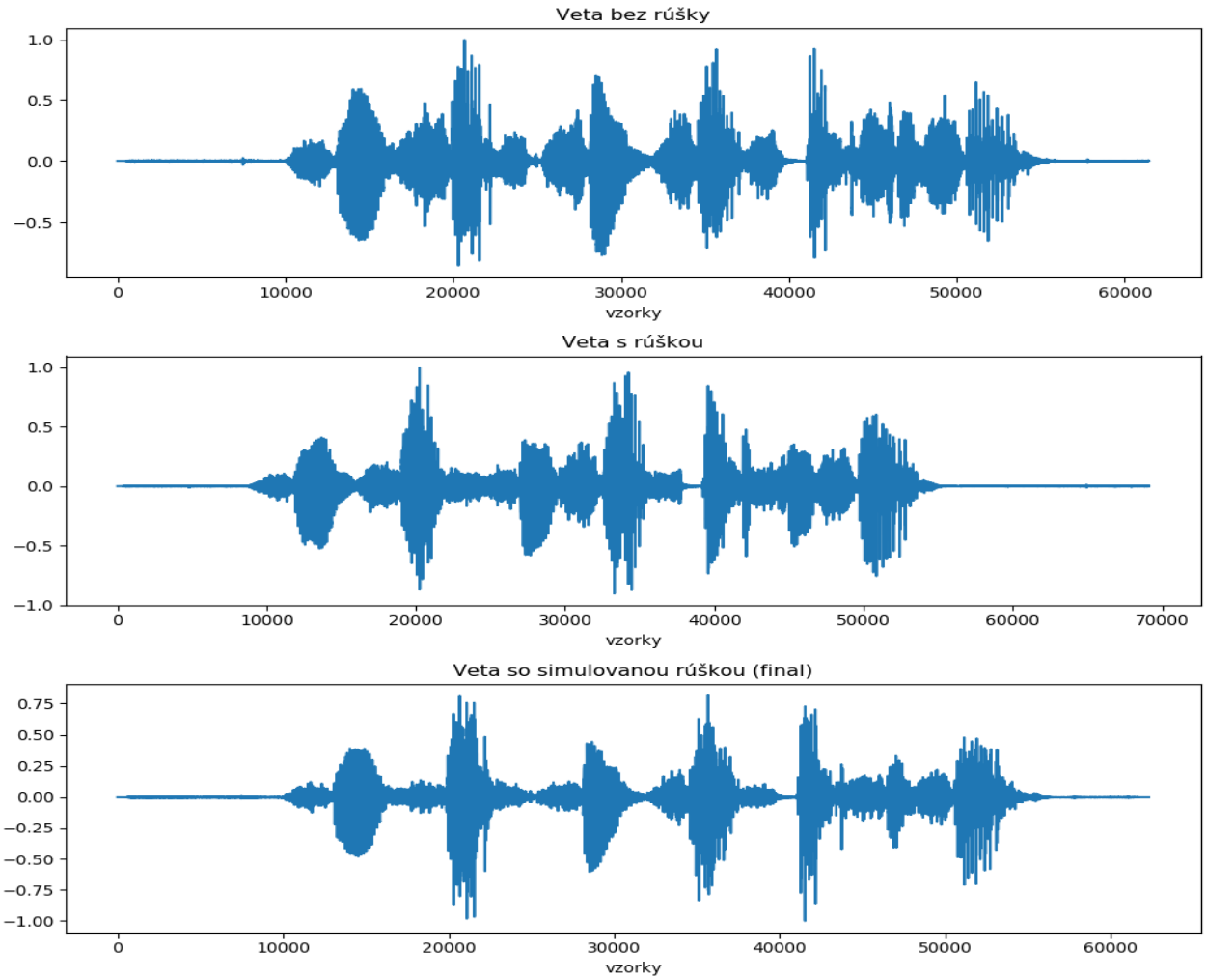
13. Frekvenčná charakteristika využívajúca metódu *match only* by mala vychádzať len z dvojíc rámcov, v ktorých sa základné frekvencie (pre maskoff a maskon) zhodujú. Časť pre frekvencie v intervale (0, 500) sa s touto metódou zdá byť vyrovnanejšia a frekvencie v okrajových častiach tohoto intervalu jasnejšie potlačené.



Záver

Kombináciou implementovaných metód, ktoré sú popísané v tomto protokole, sme získali výstupný signál so simulovanou rúškou, na ktorého grafe je po normalizácií viditeľný prechod od vstupného signálu (maskoff tón alebo veta) smerujúci k podobe signálu so skutočnou rúškou. Táto zmena vyzerá lepšie pri použití na nahrávke celej vety, než samostatného tónu.





Vplyv simulovanej rúšky je možné počuť a aj na vygenerovaných nahrávkach. Prejavuje sa v podobe mierneho zvýšenia tónu – pravdepodobne spôsobené prílišným zovretím nosa rúškou (respirátorom) prí nahrávaní tónov. Ďalej je rozoznateľné utlmenie dôrazov – vo vete na jednotlivé slová; na grafe tónov sa však ich “vyhladenie” javí byť poškodené značnými skokmi vo frekvencií. Okrem mierneho bzučania je ešte počuteľný zvuk pripomínajúci fúkanie do sklenenej fľaše – môže ísť o nedokonalý pokus zvýšenia tóniny (simulovanie zovretého nosu). Výstup implementácie využívajúcej doplňujúce úlohy však znie čistejšie a vplyv filtra je, oproti základnej verzií, výraznejší.

Všetky grafy použité v tomto protokole je možné vygenerovať spustením scriptu *src/solution.py* s parametrom *-g*