Protokol k projektu z predmetu ISS

Martin Kubička (xkubic45)

18.12.2022

Obsah

1. Úvod

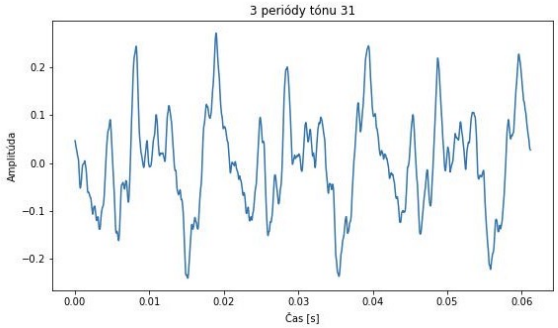
Výpočty jednotlivých úloh projektu boli vypracované v Python notebooku, ktorý je možné nahliadnuť v súbore `xkubic45.tar.gz/src/reseni.ipynb`.

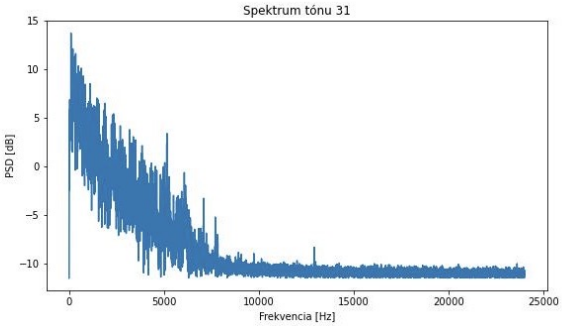
Moje priradené MIDI tóny boli:

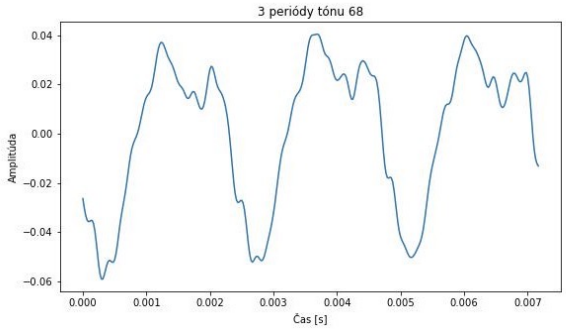
- 31 s frekvenciou 49.00
- 68 s frekvenciou 415.30
- 102 s frekvenciou 2959.96

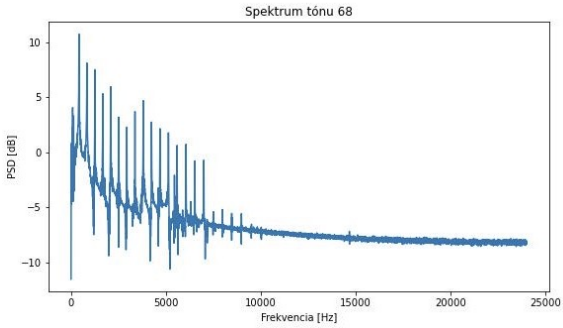
2. 1. úloha

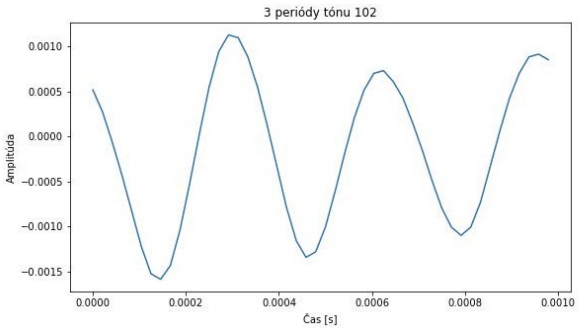
Po načítaní všetkých signálov do matice `xall` (viz. reseni.ipynb), spôsobom podobným ako bolo v zadaní, som zobrazil 3 periódy mojich troch signálov a spektrum 0.5 sekundového úseku takto:

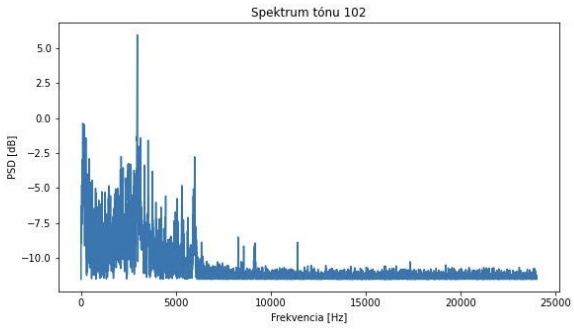












Následne som uložil moje tóny do zložky `xkubic45.tar.gz/audio`, tak ako bolo napísané v zadaní.

3. 2. úloha

Túto úlohu som riešil tak, že som si spravil autokoreláciu a DFT pre každý tón. Po vypísani hodnôt vedľa seba, som zistil, že autokorelácia funguje lepšie pre tóny s MIDI menším ako 57 (MIDI s menšími frekvenciami) a DFT pre zvyšok.

3.1 Autokorelácia

Pre tón 31 som teda použil autokoreláciu ktorej výpočet som zabezpečil pomocou funkcie `np.correlate` ako je vidno v následujúcom kóde:

```
acf = np.correlate(xall[tone,:], xall[tone,:], 'full')[-len(xall[tone,:]):]
# druhá polovica
inflection = np.diff(np.sign(np.diff(acf))) # nájdem rozdiely
peaks = (inflection < 0).nonzero()[0] + 1 # nájdem kde sú záporné
delay = peaks[acf[peaks].argmax()]
acf_fmax = Fs/delay # výsledok výpočtu
```

Základná frekvencia tónu 31 po autokorelácií mi vyšla 49.18 Hz, čo nie je úplne presné vzhľadom na očakávanú frekvenciu 49.00 Hz. Spôsobené to môže byť napríklad rozladením piána. Rozdiel ale nie je taký dramatický, takže môžeme povedať, že presnosť tejto metódy na nízkych frekvenciách je dobrá oproti DFT. (viz. nižšie)

3.2 DFT

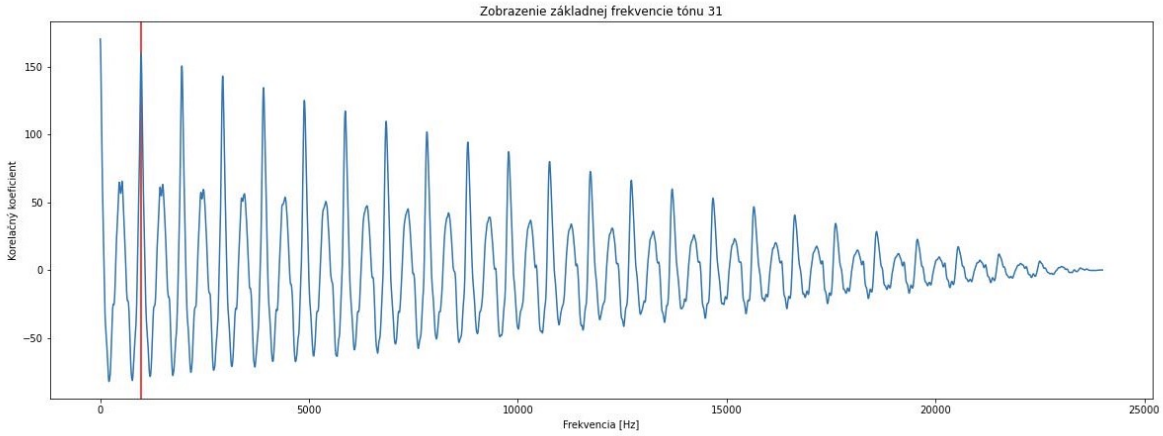
Pre tóny 68 a 102 som použil DFT implementové následovne:

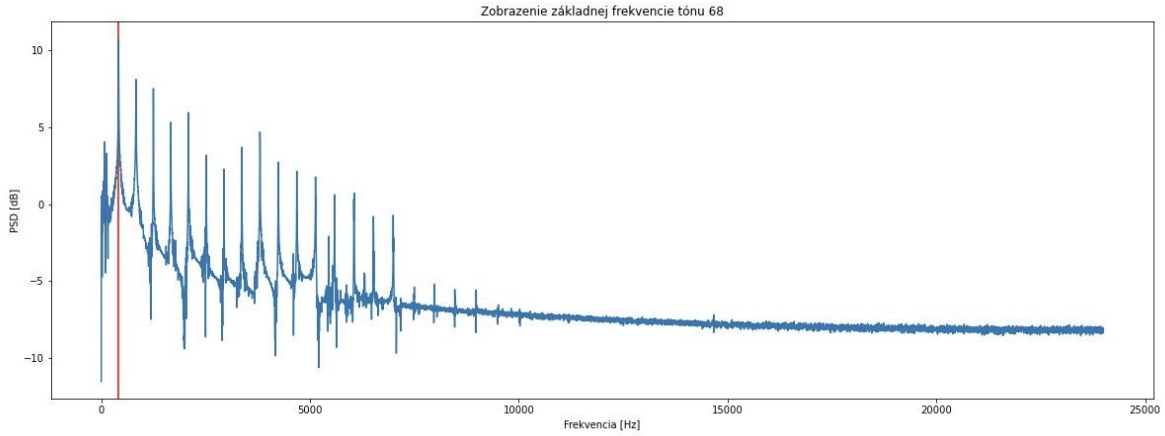
```
toneDFT = np.fft.fft(xall[tone,:])
module = np.abs(toneDFT)
moduleHalf = module[:module.size//2]
kall = np.arange(0,int(N/2) + 1)
f = kall / N * Fs
fmax = f[np.argmax(moduleHalf)] # výsledok výpočtu
```

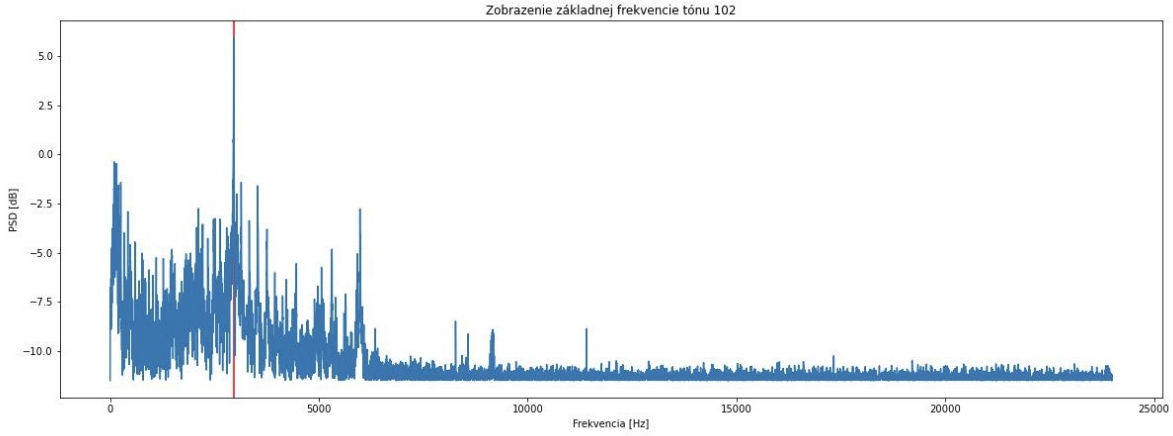
Základná frekvencia tónu 68 po prevedení DFT mi vyšla 416.0 Hz a očakávaná frekvencia bola 415.30 Hz. Pri tóne 102 mi vyšla frekvencia 2962.0, kde očakávaná frekvencia bola 2959.96. Ako som spomínal vyššie, tiež môže byť nepresnosť zapríčinená rozladením piána a zároveň nepresnosťou metódy DFT, pri ktorej si môžeme všimnúť, že čím vyššiu frekvenciu očakávame, tým je nepresnosť vyššia.

3.3 Grafy

Ďalej môžeme vidieť graf s použitými metódami pre jednotlivé tóny, kde červená čiara vyznačuje základnú frekvenciu.







4. 3. úloha

V tejto úlohe som spresnil odhad základnej frekvencie pomocou DTFT. Moje riešenie bolo inšpirované z Python notebookov z prednášok. Zvolil som metódu takú, že som použil frekvenciu +- 2 koeficientov DFT okolo odhadnutého základného tónu. Pri tónoch s MIDI nižšou ako 57, som výsledok podelil dvojkou, tak ako bolo v zadaní.

Porovnanie pôvodnej základnej frekvencie a základnej frekvencie po DTFT:

- MIDI 31 - **Autokorelácia:** 49.18 Hz, **DTFT:** 48.97 Hz (očakávaná frekvencia 49.00 Hz)
- MIDI 68 - **DFT:** 416.0 Hz, **DTFT:** 415.45 Hz (očakávaná frekvencia 415.30 Hz)
- MIDI 102 - **DFT:** 2962.0 Hz, **DTFT:** 2961.25 Hz (očakávaná frekvencia 2959.96 Hz)

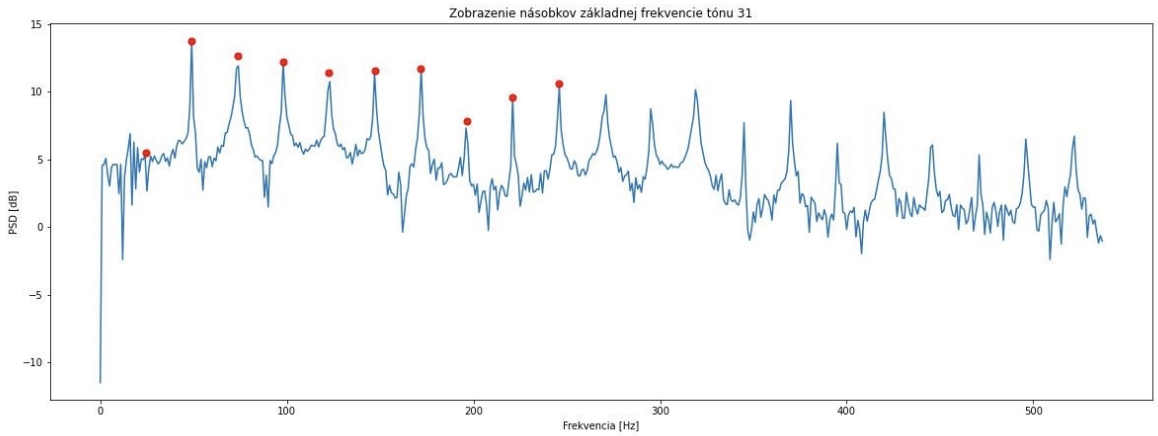
Ako môžeme z výsledkov vyššie vidieť, presnejšie sme sa priblížili k očakávaným frekvenciám.

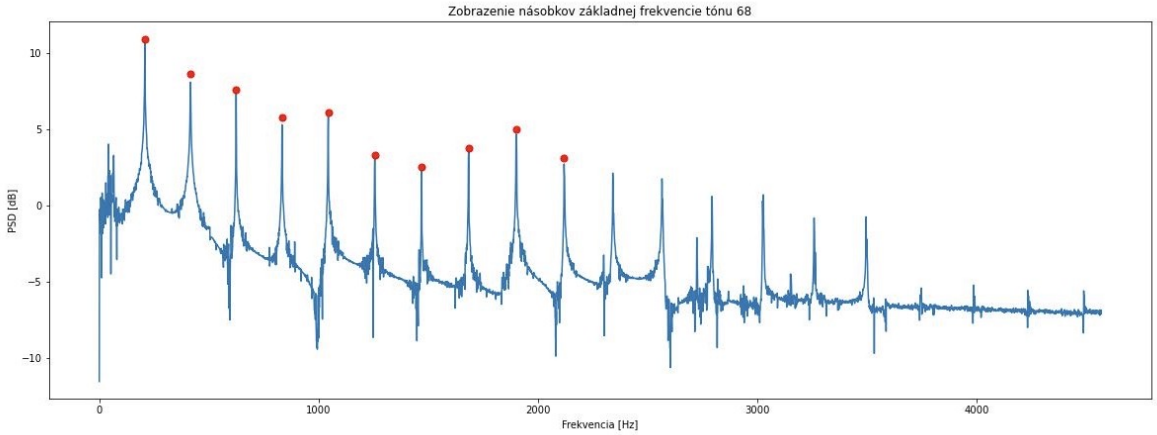
Implementácia DTFT:

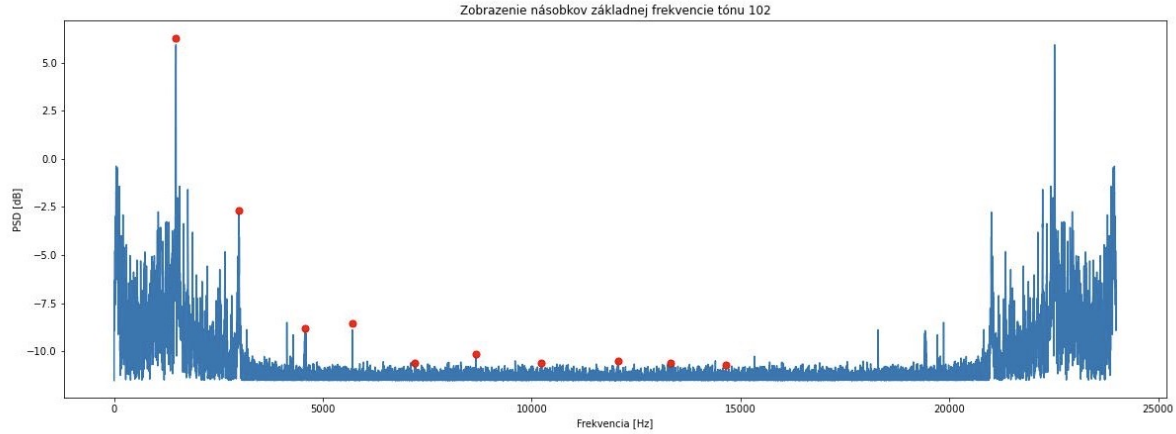
```
FREQRANGE = 2
FREQPOINTS = 200

n = np.arange(0,N)
fsweep = np.linspace(fmax-FREQRANGE, fmax+FREQRANGE,FREQPOINTS)
A = np.zeros([FREQPOINTS, N],dtype=complex)
for k in np.arange(0,FREQPOINTS):
   A[k,:]  = np.exp(-1j * 2 * np.pi * fsweep[k] / Fs * n)
Xdtft = np.matmul(A,xall[tone,:].T)

precisefmax = fsweep[np.argmax(np.abs(Xdtft))]
```

5. 4. úloha

Pri tejto úlohe som zvolil postup taký, že som si vždy spravil násobok základnej frekvencie a následne spravil DTFT. Vždy som si do jedného pola uložil 10 základných frekvencií aktuálneho tónu a do ďalšieho pola 10 modulov. Následne som vykreslil jednotlivé spektrá takto:





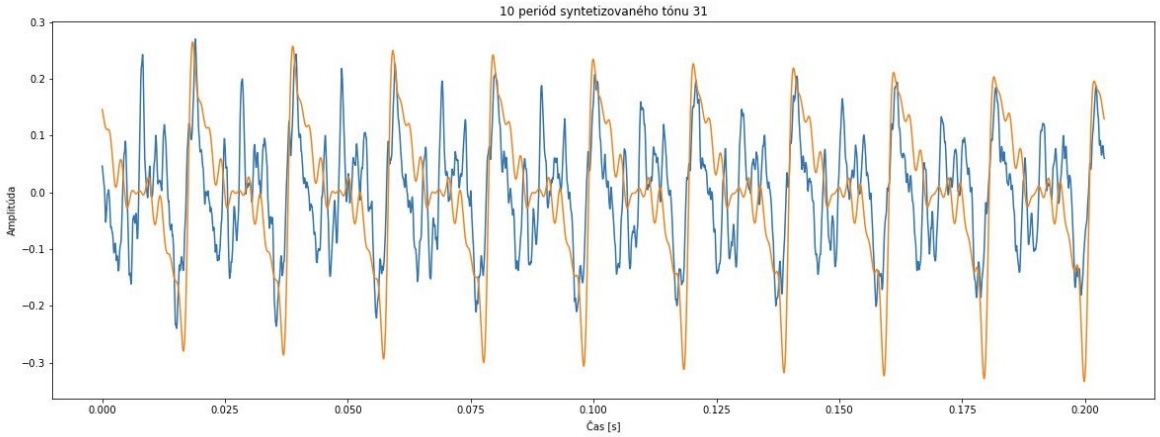


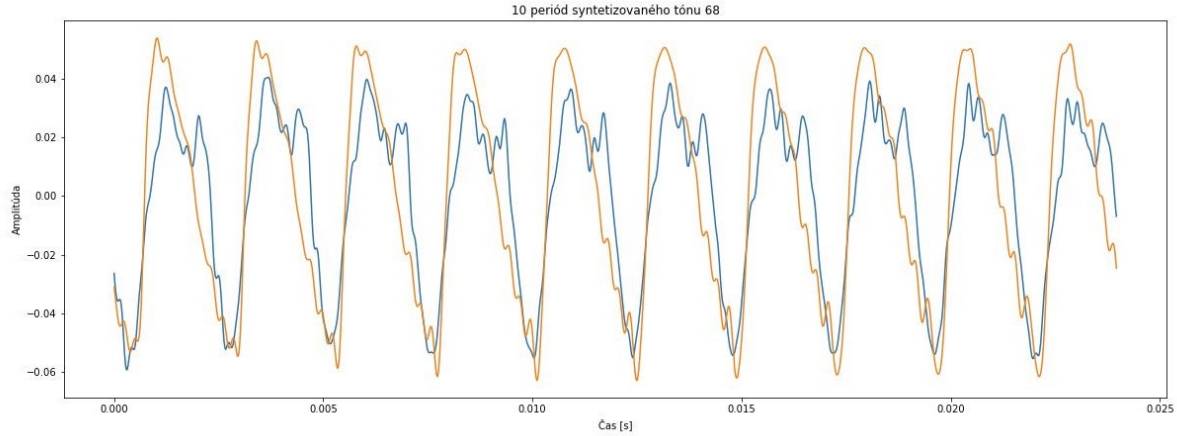
Môžeme si všimnúť, že pri nižších násobkoch základnej frekvencie vyšli presne, ale čím vyšší tón a čím väčší násobok, tým nepresnejšia frekvencia vyšla. Najlepšie je to vidno pri tóne 102. Riešenie by bolo napríklad zväčšiť `FREQRANGE`.

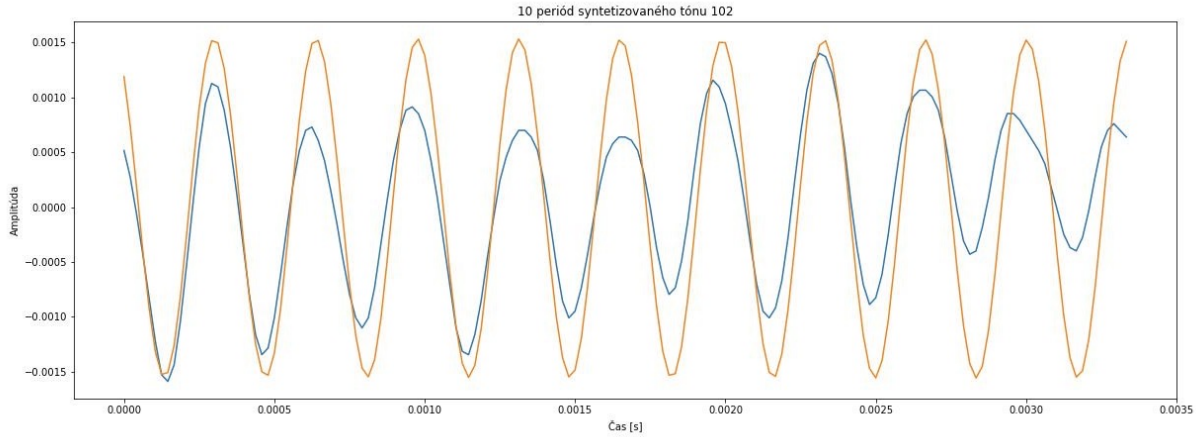
6. 5. úloha

V tejto úlohe som mal spraviť syntézu tónov na základe odhadnutých frekvencií a modulov. Robil som to spôsobom takým, že som vždy výnásobil sinusoidu získanou frekvenciou a amplitúdou. Výsledné jedno sekundové tóny som uložil do `xkubic45.tar.gz/audio`.

Na následujúcich obrázkoch porovnávam 10 períód pôvodných tónov (modrá čiara) s mojími syntetizovanými tónmi (oranžová čiara), kde som ručne signály synchronizoval tak, aby začínali rovnakou fázou:







Na grafoch vidíme určité nepresnosti a rozdiely, ktoré môžu byť zapríčinené nepresným odhadom základnej frekvencie (napr. kvôli rozladeniu piána) a následne aj modulov.

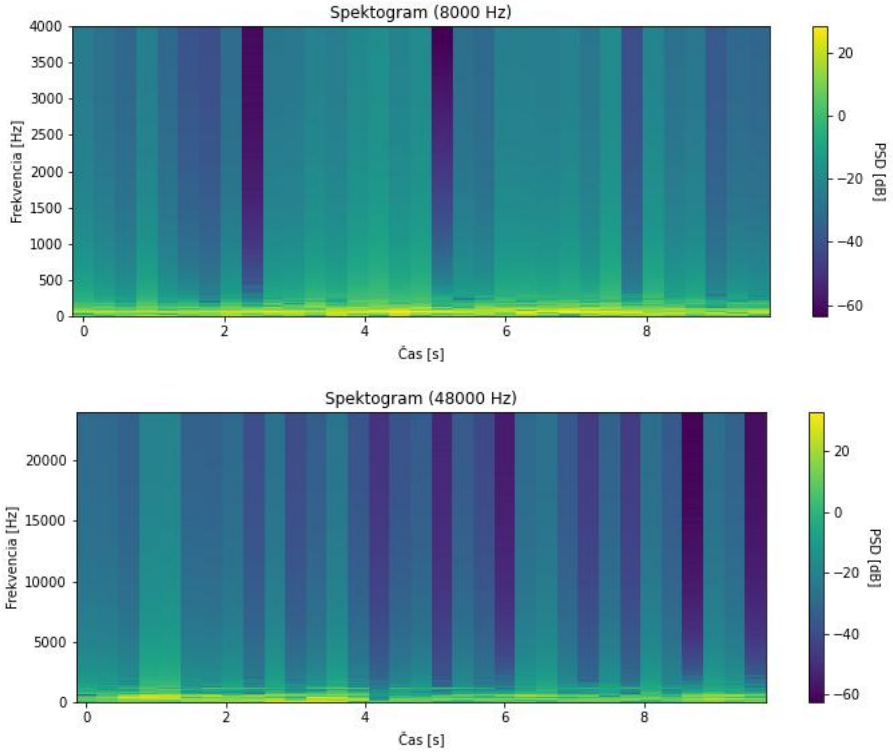
Presnejšie riešenie by mohlo byť, že by sme zobrali kratšie intervaly medzi jednotlivými násobkami.

7. 6. úloha

Túto úlohu som riešil tak, že som prechádzal jednotlivé riadky zadaného súboru a do vynulovaného signalú som pridával jednotlivé tóny z uloženého pola. Prekrytie tónov som vyriešil jednoducho pričítaním a hlasitosť vynásobením (viz. python notebook). Po prejdení celého súboru som uložil 10 sekúnd nahrávky s vzorkovaciou frekvenciou 8000 Hz a 48000 Hz.

8. 7. úloha

Spektogram som vykreslil spôsobom akým bolo ukazované na prednáške (v python notebooku z prednášky), kde som podľa zadania zmenil veľkosť okna, prekryv a počet vzorkov DFT pre jednotlivé frekvencie. Spektogramy mi vyšli následovne:



Na spektograme môžeme vidieť na x-ovej osi čas, na y-ovej osi frekvenciu. Jednotlivé farby nám hovoria, ako silný ten signál je. Keď porovnáme spektogramy vyššie, tak vidíme, že pri 8000 Hz máme nižšie frekvencie, narozdiel od 48000 Hz. To nám spôsobuje "temnejšie" tóny pri vypočutí nahrávky z úlohy vyššie.