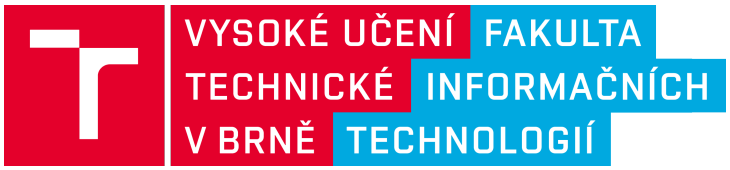
Vysoké učení technické v Brně
Fakulta informačních technologií



Signály a systémy (ISS)

akademický rok 2021/22

Projekt do ISS: Analýza zvukového souboru

Autor:
Petr Junák (xjunak01)

Brno, 7. ledna 2022

Obsah

1 Úvod

Smyslem projektu je analýza krátkého zvokového souboru, detekce zarušených pásem a vytvoření filtru pro umělého odstranění šumu.

Z důvodu osobního procvičení píši komentáře v kódu anglicky a od angličtiny se i odvíjí názvy proměnných.

Byl využit jazyk python a volně dostupné knihovny:

- Matematické operace, operace nad polem, apod.

  math

  numpy

- Práce s grafy

  matplotlib.pyplot

- Práce se zvukovým signálem a tvorba filtrů

  scipy.signal

  soundfile

Projekt byl tvořen a testován ve Visual Studio Code.

2 Projekt

2.1 Základy

Pro načtení signál byl použit příkaz

```
orig_audio, fs = sf.read('audio/xjunak01.wav')
```

Maximální a minimální hodnoty signálu jsou vypsány na obrazovku pomocí jednoduchého printu a funkcí .max() a .min().

```
Maximální hodnota:  0.154754638671875 Minimální hodnota 0.0
```

2.2 Předzpracování rámce

Ustřednění a normalizace jsou provedeny odečtením průměrné hodnoty a následným podělením nejvyšší absolutní hodnotou nad všemi vzorky.
Kód pro rozdělení signálu do matice rámců:

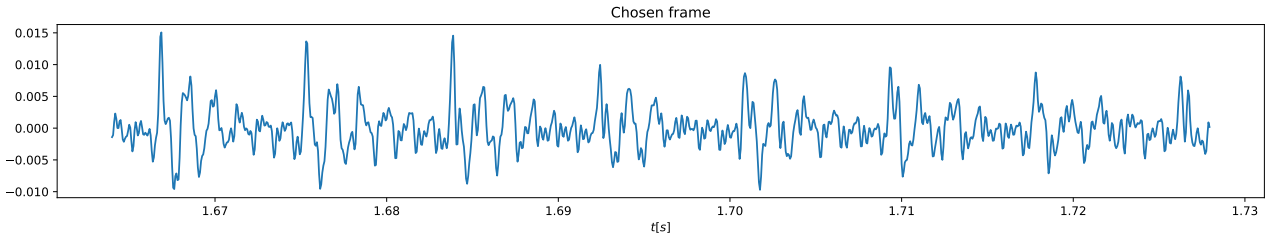
```
# Segment Creation
seg_start = 0   # Beginning of segment in seconds
seg_len = 1024/fs  # Length of the segment in sec
seg_shift = 512/fs # Length of the segment shift in sec

segArray = []
length = int(((orig_audio_t[-1]*fs)/512)-1) # Array length to hold all segments of orig_audio
for i in range(length):     # Fills segArray with segments from orig_audio
    seg_start_sam = int(seg_start * fs)              # start of the segment in samples
    seg_len_sam = int((seg_start + seg_len) * fs)     # end of the segment in samples

    segment = orig_audio[seg_start_sam:seg_len_sam]

    seg_start = seg_start + seg_shift
    segArray.append(segment)
      You, a week ago • Finished project
chosen_frame = 52               #  52nd frame chosen
seg = segArray[chosen_frame]    #  Isolating the chosen frame
```

Pro analýzu byl zvolen rámec 52 kvůli jeho vlastnostem

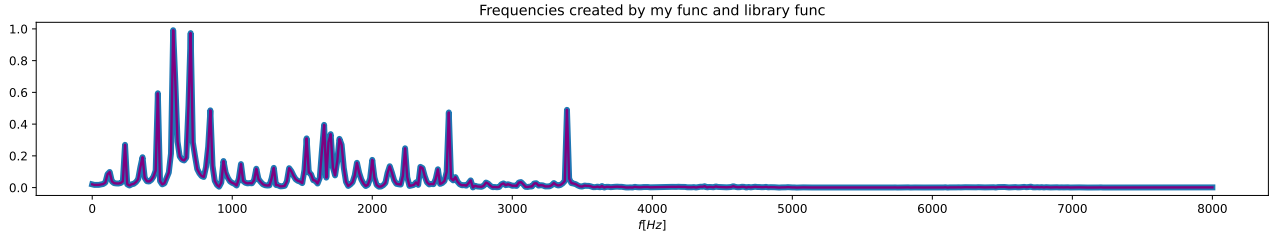


2.3 DFT

Kód pro DFT:

```
# Transformation using library function
Frequencies_lib = np.fft.fft(seg)
# Generates DFT matrix for N = 1024
dftmtx = np.fft.fft(np.eye(1024))
frequencies = dftmtx.dot(seg)
# Multiplying the signal with DFT matrix
```

Následující graf znázorňuje jak se vzájemně překrývají frekvence získané knihovní funkcí a frekvence získané násobením maticí.

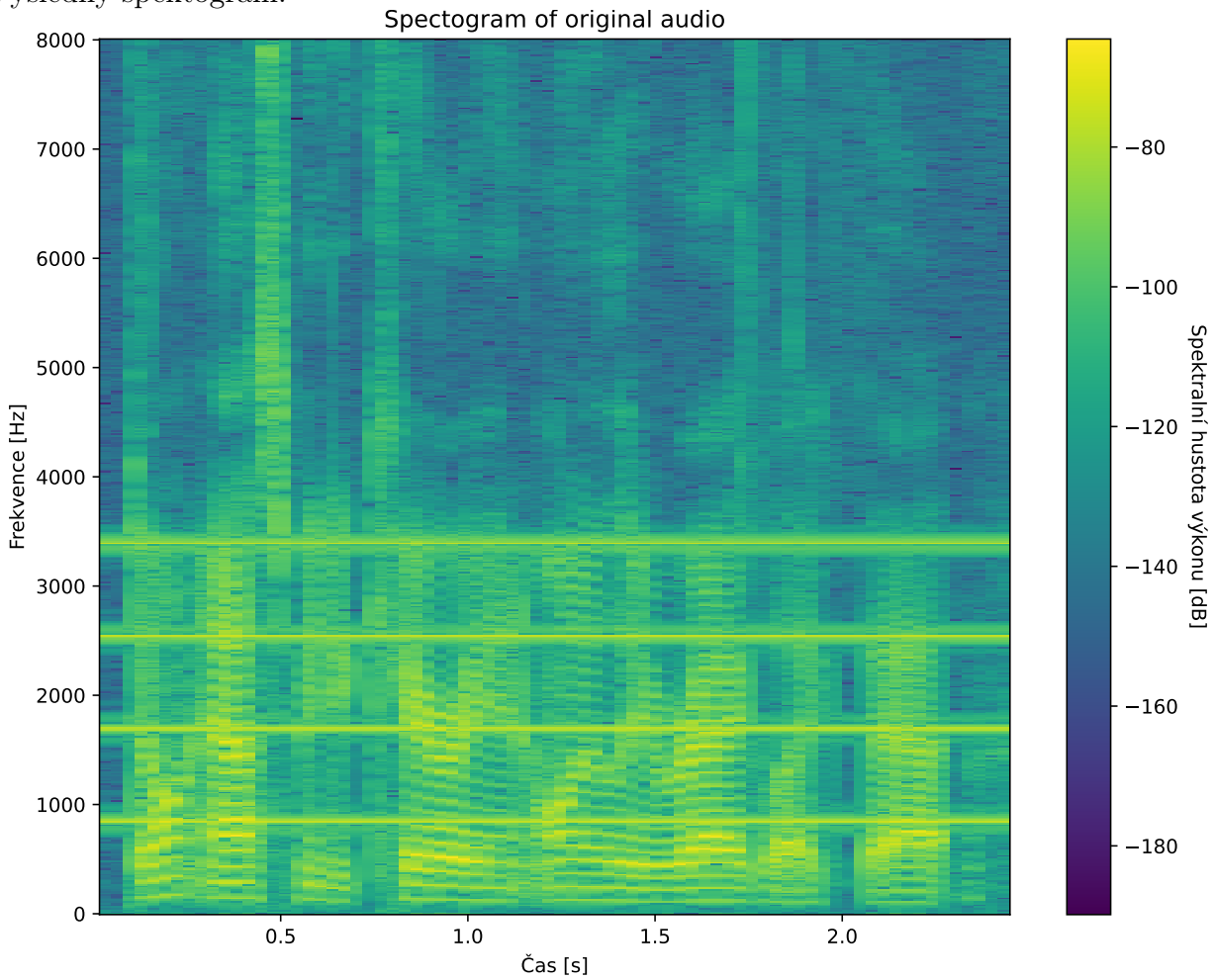


2.4 Spektrogram

Kód pro Spektogram:

```
# Creating spectogram for the whole audio
f, t, sgr = spectrogram(orig_audio, fs, nperseg=1024, noverlap=512)
sgr_log = 10 * np.log10(sgr+1e-20)
```

Výsledný spektogram:



2.5 Určení rušivých frekvencí

Ze spectogramu byly následně určeny frekvence rušivého signálu.

Nejdříve přibližně pomocí odhadu a následně po dokončení filtru byly modifikovány kontrolou míry funkčnosti dokud nebyla trefena frekvence s přesností na 1 hz.

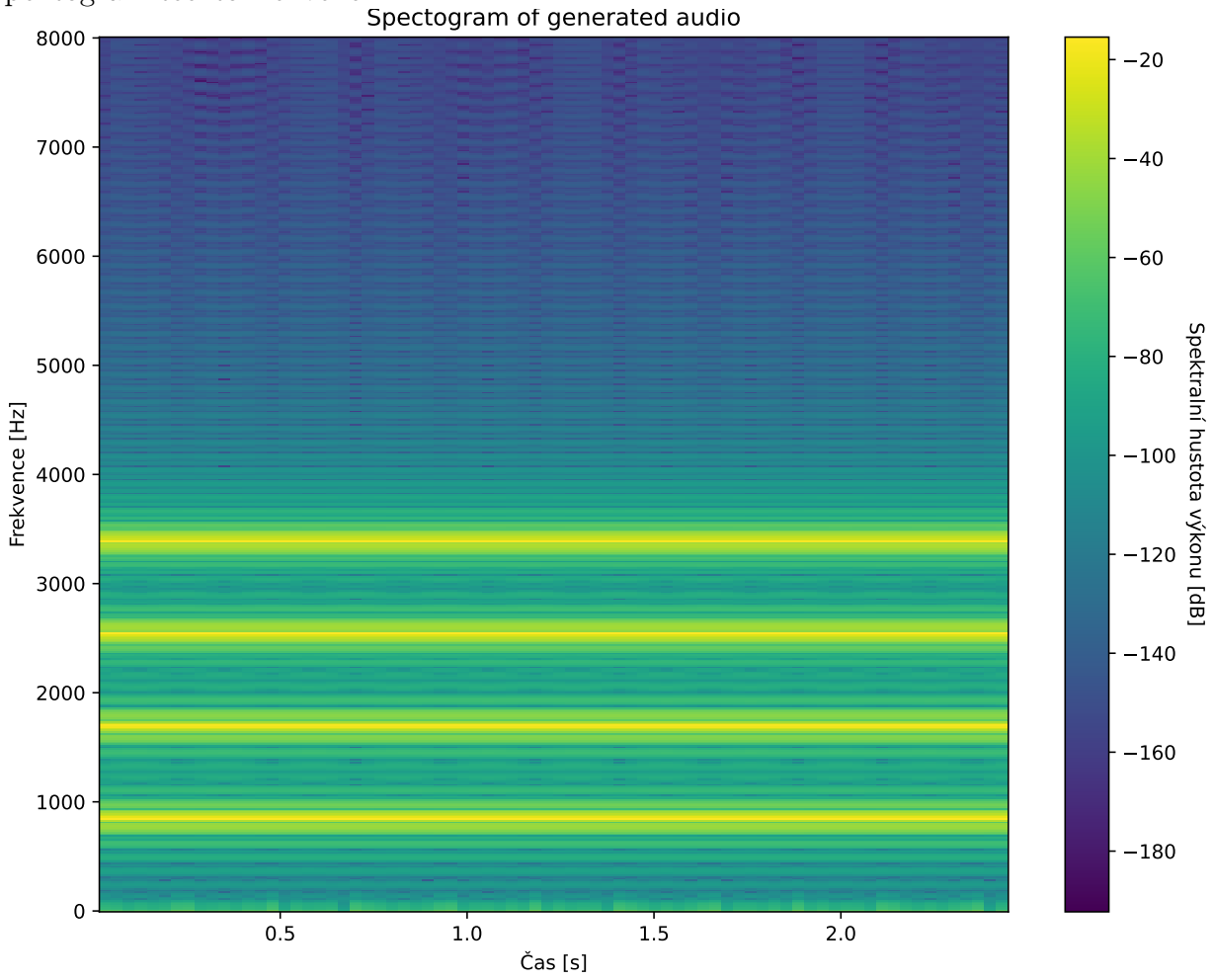
```
# Deteremining the noise frequencies
freq1 = 848       # The lowest noise frequency determined by analyzing the spectogram
freq2 = freq1 * 2
freq3 = freq1 * 3
freq4 = freq1 * 4
```

2.6 Generování signálu

Kód pro generovní signálu:

```
# Substracting the noise frequencies
samples = []
for i in range(orig_audio.size):
    samples.append(i*1/fs)
# generating cosinuses from known noise frequencies
cos1 = np.cos(np.array(samples) * 2 * np.pi * freq1)
cos2 = np.cos(np.array(samples) * 2 * np.pi * freq2)
cos3 = np.cos(np.array(samples) * 2 * np.pi * freq3)
cos4 = np.cos(np.array(samples) * 2 * np.pi * freq4)
# Adding all noise cosinuses into one
cos = cos1 + cos2 + cos3 + cos4
# Creating file from the result cosinus
sf.write("../audio/generated.wav", cos, fs)
```

Spektogram těchto frekvencí:



2.7 Čisticí filtr

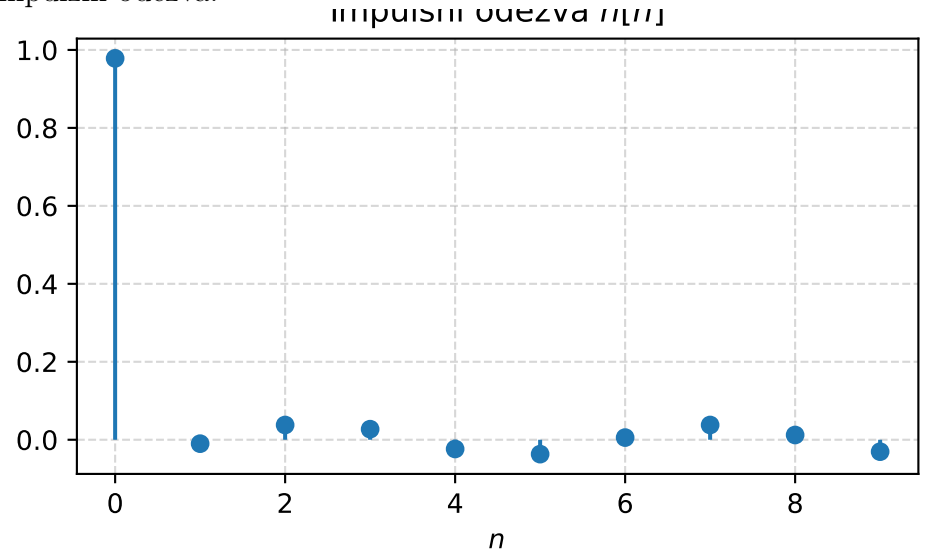
Kód pro filtr a impulzní odezvu (zahrnuje ukládání frekvvenčních charakteristik jednotlivých filtrů pro pozdější použití):

```
# Creating filter
Q = 30.0  # Filter quality factor
# Design notch filter for every noise frequency
b, a = iirnotch(freq1, Q, fs)
freq1, h1 = freqz(b, a, 2048)
filter1 = 10 * np.log10(abs(h1+1e-20))
filtered = lfilter(b, a, orig_audio)
b, a = iirnotch(freq2, Q, fs)
freq2, h2 = freqz(b, a, 2048)
filter2 = 10 * np.log10(abs(h2+1e-20))
filtered = lfilter(b, a, filtered)
b, a = iirnotch(freq3, Q, fs)
freq3, h3 = freqz(b, a, 2048)
filter3 = 10 * np.log10(abs(h3+1e-20))
filtered = lfilter(b, a, filtered)
b, a = iirnotch(freq4, Q, fs)
freq4, h4 = freqz(b, a, 2048)
filter4 = 10 * np.log10(abs(h4+1e-20))
w4, H4 = freqz(b, a)
filtered = lfilter(b, a, filtered)
# Impuls response
N_imp = 10
imp = [1, *np.zeros(N_imp-1)]
h = lfilter(b, a, imp)
```

Pro tvorbu filtru byla opakovaně použita funkce iirnotch a filtry byly průběžně postupně aplikovány.

```
iirnotch(<filtrovaná frekvence>,
<faktor kvality (přesnosti) filtru>,
<vzorkovací frekvence>)
```

Impulzní odezva:

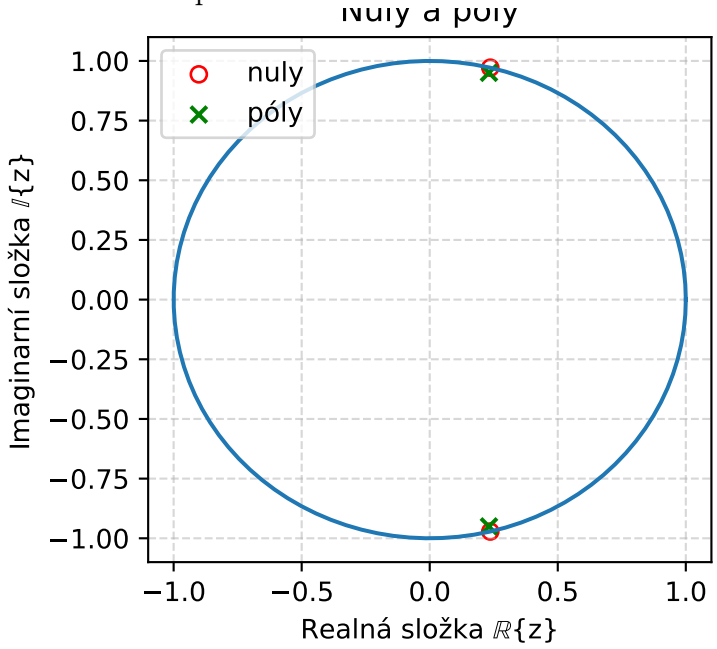


2.8 Nulové body a póly

Kód pro nuly a póly (z důvodu podstaty úkolu zahrnuta i tvorba grafu):

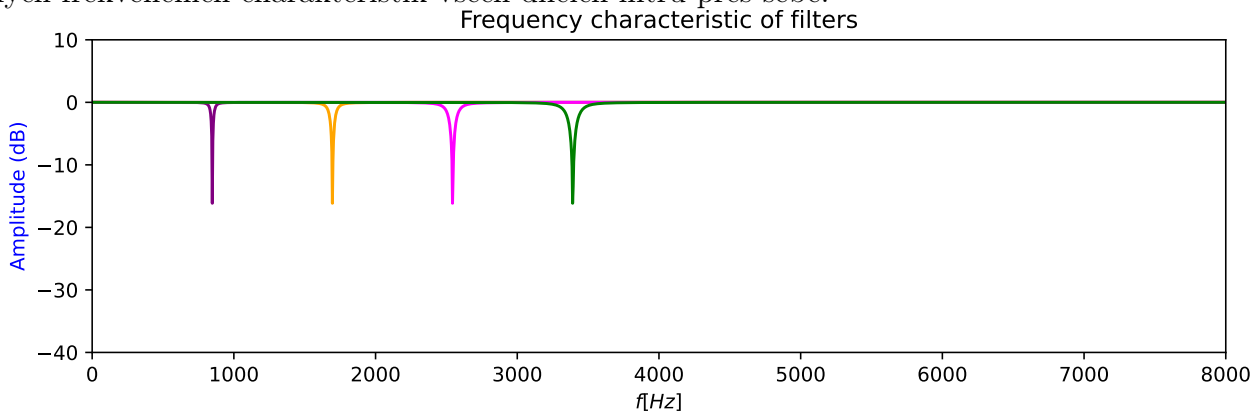
```
# Zero points and poles      You, a week ago • Finished project
z, p, k = tf2zpk(b, a)
plt.figure(figsize=(4,3.5))
# unit circle
ang = np.linspace(0, 2*np.pi,100)
plt.plot(np.cos(ang), np.sin(ang))
plt.scatter(np.real(z), np.imag(z), marker='o', facecolors='none', edgecolors='r', label='nuly')
plt.scatter(np.real(p), np.imag(p), marker='x', color='g', label='póly')
plt.gca().set_xlabel('Realná složka $\mathbb{R}\{$z$\}$')
plt.gca().set_ylabel('Imaginarní složka $\mathbb{I}\{$z$\}$')
plt.grid(alpha=0.5, linestyle='--')
plt.legend(loc='upper left')
plt.tight_layout()
plt.title('Nuly a póly')
plt.savefig('zeroes_and_poles.pdf')
```

Zobrazení nul a pólů:



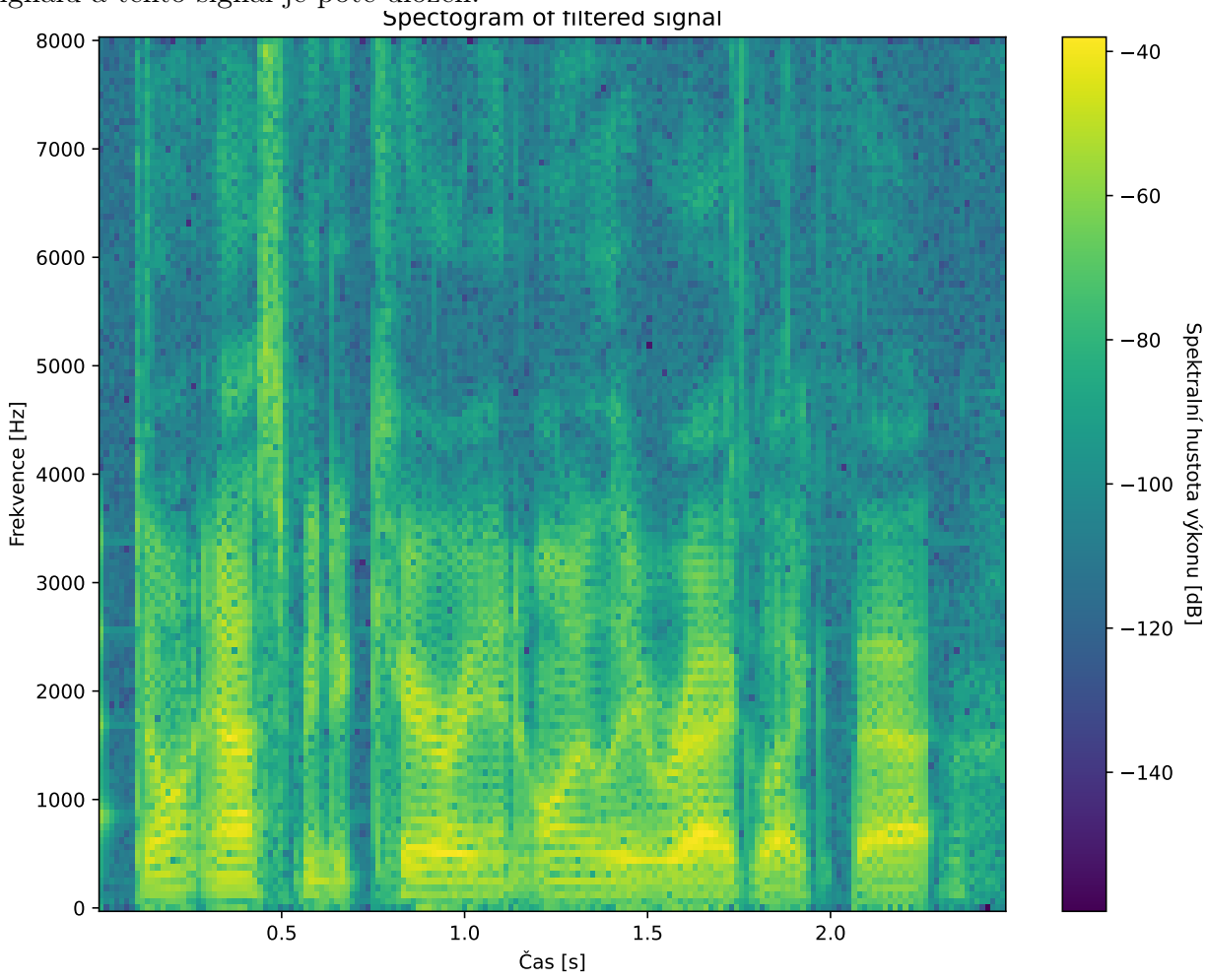
2.9 Frekvenční charakteristika

Frekvenční charakteristika výsledného filtru je znázorněna pomocí zobrazení již připravených frekvenčních charakteristik všech dílčích filtrů přes sebe.



2.10 Filtrace

Filtrace byla provedena už v dřívější části, zde je pouze vytvořen spektogram vyfiltrovaného signálu a tento signál je poté uložen.



3 Závěr

Po nalezení přesné frekvence rušivého signálu vylo možné velmi zvýšit přesnost filtru a zachovat tak mnohem více z původního signálu i při plném odstranění šumu. To bylo možné hlavně díky silné izolovanosti šumu na 4 konkrétní frekvence, které bylo možné prakticky umlčet téměř bez znatelných ztrát.

4 Bibliografie

Reference

[1] Web Kateřiny Zmolíkové [online]. Dostupné z: https://www.fit.vutbr.cz/~izmolikova/ISS/project/

[2] NumPy documentation — NumPy v1.22 Manual. NumPy [online]. Copyright © Copyright 2008 [cit. 07.01.2022]. Dostupné z: https://numpy.org/doc/stable/

[3] Numpy and Scipy Documentation — Numpy and Scipy documentation. 302 Found [online]. Copyright © Copyright 2021 SciPy developers. [cit. 07.01.2022]. Dostupné z: https://docs.scipy.org/doc/

[4] DevDocs — Matplotlib 3.1 documentation. DevDocs API Documentation [online]. Dostupné z: https://devdocs.io/matplotlib~3.1/