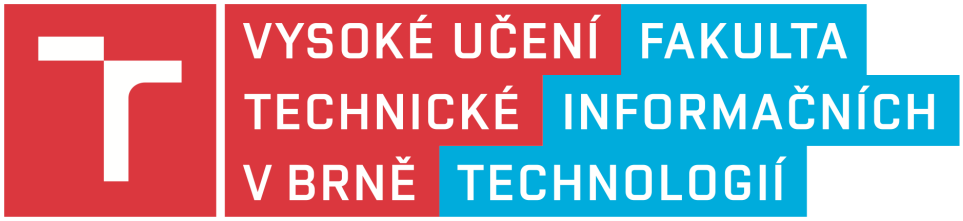


VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ

FAKULTA INFORMAČNÍCH TECHNOLOGIÍ

ISS - Signály a systémy

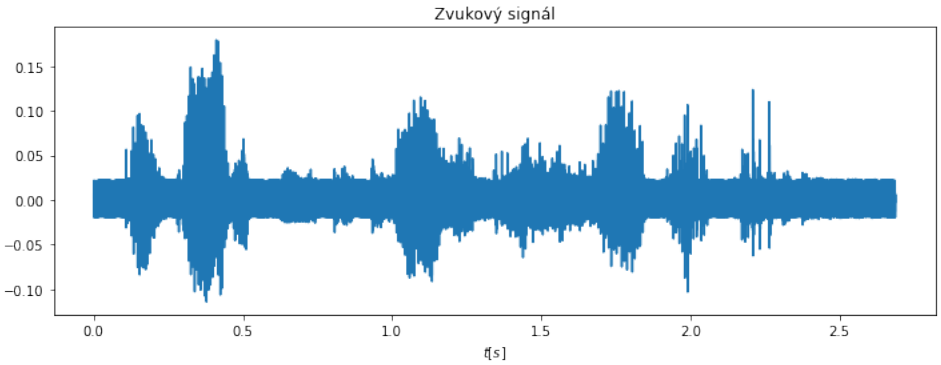
Semestrální projekt

Tadeáš Kachyňa (xkachy00)

7. ledna 2022

Projekt byl vypracován s použitím Python a Jupyter Notebook. V adresáři `src` se nachází soubour `xkachy00.ipynb` obsahující zdrojový kód s výpočty, který generuje grafy použité v této dokumentaci. Projekt vyžaduje také knihovny `numpy, scipy, matplotlib, soundfile a IPython`.

1 Základy



soubor	délka ve vzorcích	délka v sekundách
xkachy00.wav	43008	2.688

min. hodnota	-0.114044189453125
max. hodnota	0.1792602539062
vzorkovací frekvence	16 000 Hz

2 Předzpracování a rámce

Signál jsem ustřednil a normalizoval pomocí funkcí `numpy.mean(x)` a `numpy.abs(x).max(x)`. Následně jsem podle vzorečku spočítal kolik rámců budu potřebovat. Vytvořil jsem si nové pole a pomocí cyklu jsem jednotlivé rámce vytvořil.

$$N_{\text{ram}} = \left[\frac{N - I_{\text{ram}}}{S_{\text{ram}}}\right] \tag{1}$$

kde:

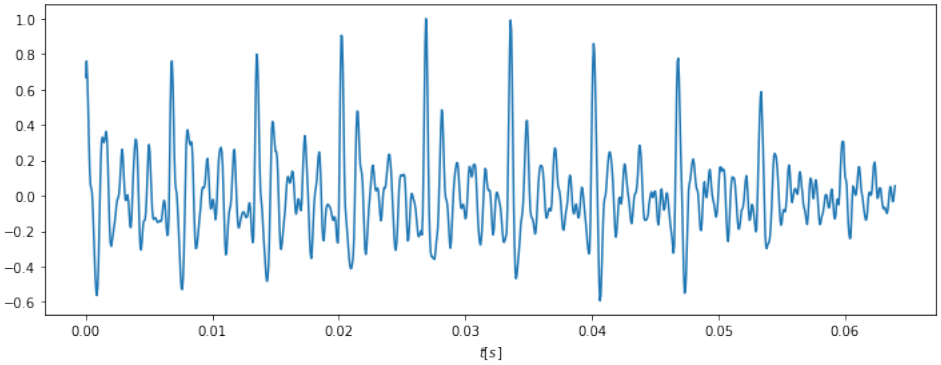
N ... počet vzorků,

I_{ram} ... délka jedno rámce,

N_{ram} ... výsledný počet rámců,

S_{ram} ... polovina jednoho rámce

Níže příkladam jeden ze znělých rámců.



3 DFT

DFT jsem implementoval podle základního vztahu pro její výpočet:

$$X[k] = \sum_{n=0}^{N-1} x[n] \cdot e^{-j2\pi \frac{n}{N} k}$$

Podle zadání jsme měli pracovat co nejvíce vektorově, což si myslím, že se mi povedlo. Implementaci jsem následně porovnal s knihovní funkcí `np.fft.fft`.

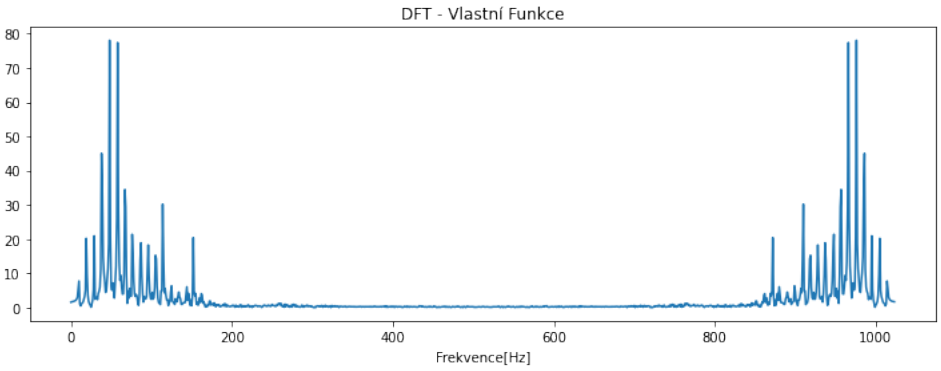
```
DFT_log = []
for frame in range(83): # pro každý rámec
    x = frames[frame]
    N = x.shape[0]
    n = np.arange(N)
    k = n.reshape((N,1))
    M = np.exp(-2j * np.pi * k * n / N)
    DFT_log.append(np.dot(M,x))

    Val = 10 * np.log10(np.abs(DFT_log[frame])**2) # úprava koeficientů pro spektrogram
    DFT_log[frame] = Val
    DFT_log[frame] = DFT_log[frame][:512]
```

Koeficienty DFT jsem si následně ihned upravil pro tvorbu spektrogramu pomocí vzorečku:

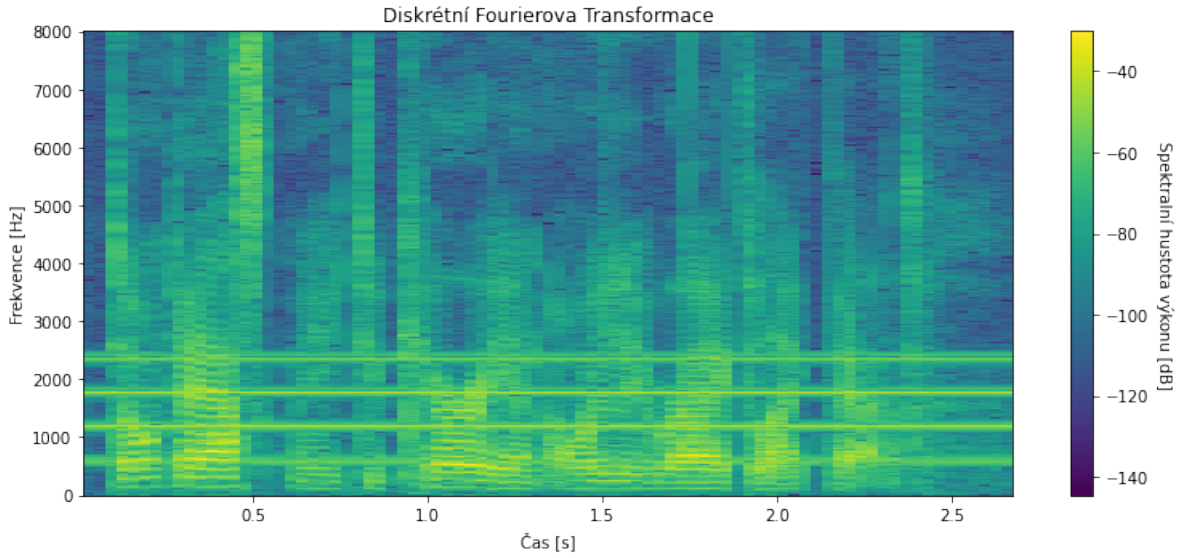
$$P[k] = 10 * \log_{10} |X[k]|^2$$

Jako výsledek přikládám graf DFT dvanáctého rámce.



4 Spektrogram

Spektrogram jsem vytvořil jak knihovní funkcí `spectrogram`, tak i mojí funkcí. Níže je výsledek knihovní funkce. Už teď můžeme přibližně určit na jakých frekvencích se rušivé signály nachází.



5 Určení rušivých frekvencí

Viditelné rušivé frekvence ve spektrogramu jsem určoval "ručně". Vyšlo mi tedy, že se nacházejí přibližně na frekvencích:

$$f_1 = 594 \text{ Hz}$$
$$f_2 = 1188 \text{ Hz}$$
$$f_3 = 1782 \text{ Hz}$$
$$f_4 = 2376 \text{ Hz}$$

Dokážu o nich říct, že jsou harmonicky vztažené, tzn. že druhá, třetí a čtvrtá frekvence jsou násboky té nejnižší frekvence.

6 Generování signálu

Pomocí zjištěných rušivých frekvencí jsem vygeneroval 4 cosinusovky dle níže uvedeného kódu a sečetl je. Před uložením bylo třeba signál normalizovat a vynásobit maximální hodnotou integeru. Signál jsem poté uložil do `audio/4cos.wav`. Následně jsem si pomocí poslechu a spektrogramu ověřil, že jsem určil frekvence a vygeneroval signál správně.

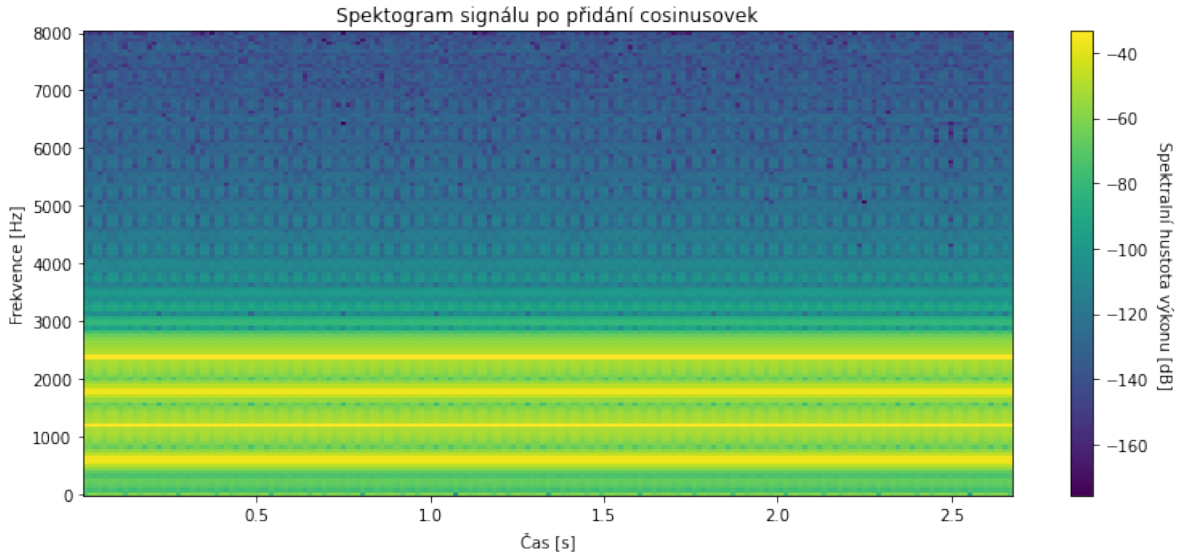
```
ts = np.arange(43008)
ts = ts / 16000

f1 = 594
f2 = 1188
f3 = 1782
f4 = 2376

output_cos1 = np.cos(2 * np.pi * f1 * ts)
output_cos2 = np.cos(2 * np.pi * f2 * ts)
output_cos3 = np.cos(2 * np.pi * f3 * ts)
output_cos4 = np.cos(2 * np.pi * f4 * ts)

# sečtení všech 4 cosinusovek, normalizování a vynásobení maximální hodnotou integeru
output_cos_total = output_cos1 + output_cos2 + output_cos3 + output_cos4
output_cos_total = output_cos_total / np.abs(output_cos_total).max()
output_cos_total = output_cos_total * np.iinfo(np.int16).max

wavfile.write("../audio/4cos.wav", fs , output_cos_total.astype(np.int16))
```



7 Čistící filtr

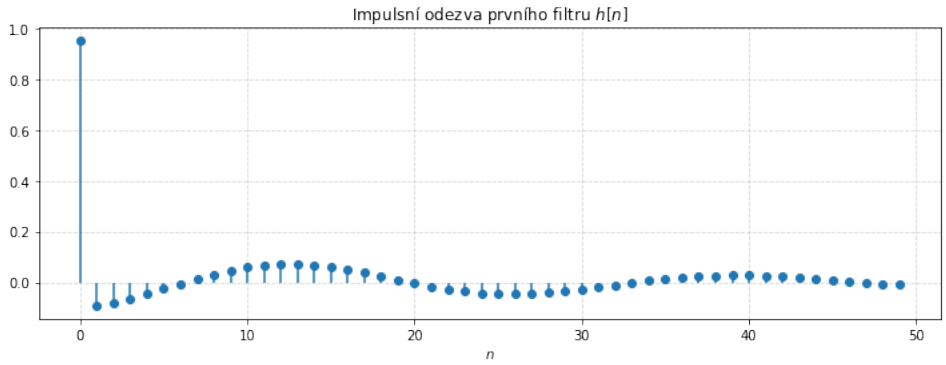
Pro čistící filtr jsem si vybral třetí variantu, tzn. vytvoření čtyř pásmových zádrží. Implementoval jsem je pomocí funkcí `buttord` a `butter` ze kterých jsem následně dostal koeficienty `a`, `b`, ze kterých jsem vygeneroval graf impulsní odezvy. Držel jsem zadání a parametry funkce `buttord` jsem určil, tak jak nám bylo doporučeno. To znamená šiři závěrného pásmena 30 Hz, šíři propustného pásma 50 Hz na každou stranu. Zvlnění jsem nastavil

na 3dB a potlačení v závěrném pásmu na -40dB. Níže přikládám koeficienty a impulsní odezvu pro každý filtr.

První pásmová zádrž:

a: 1, -7.68719722, 26.06207514, -50.88908984 , 62.58745781, -49.64574474, 24.80411925, -7.137409, 0.90579677

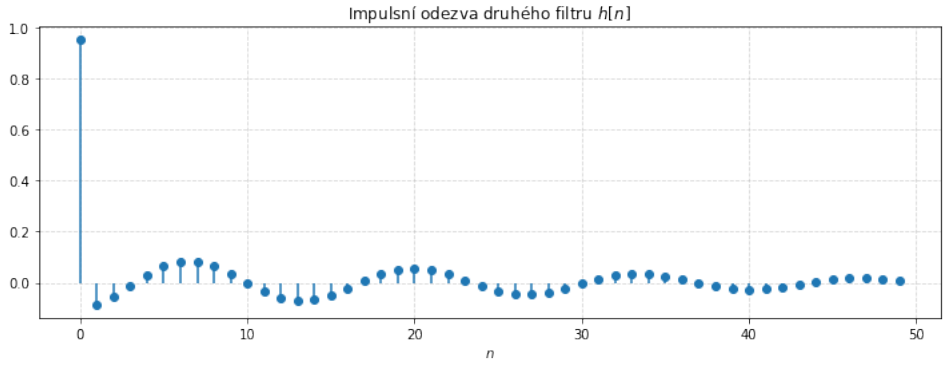
b: 0.95173356, -7.4077704, 25.42868768, -50.27194999, 62.59860649, -50.27194999 25.42868768, -7.4077704, 0.95173356



Druhá pásmová zádrž:

a: 1, -7.05520536, 22.5662882, -42.58731811, 51.78642656, -41.5287562, 21.45840604, -6.54208049, 0.90422329

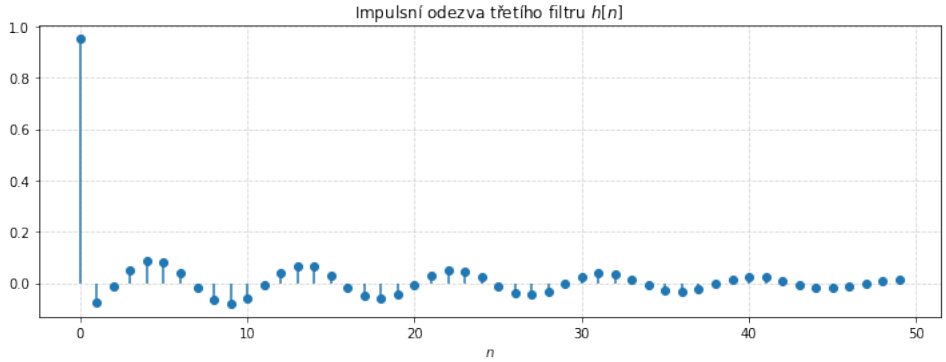
b: 0.95090656, -6.79433816, 22.0085029, -42.06234192, 51.79652518, -42.06234192 22.0085029, -6.79433816, 0.95090656



Třetí pásmová zádrž:

a: 1, -6.04223991, 17.59020092, -31.4589995 37.60485825, -30.67236847, 16.72151324, -5.60022021, 0.90367081

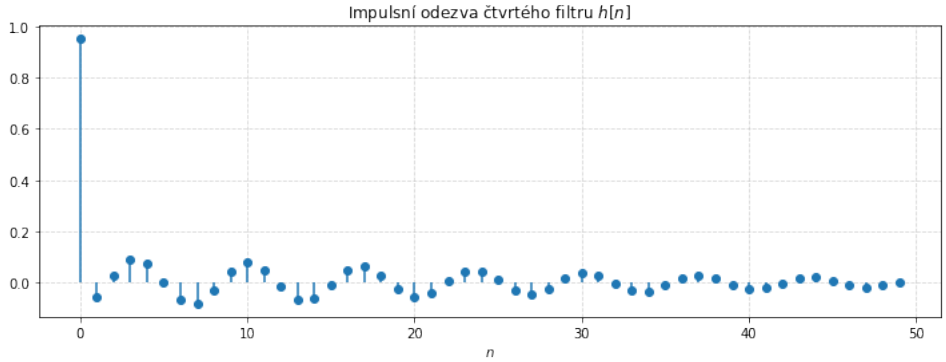
b: 0.95061602, -5.81749931, 17.15300373, -31.06941473, 37.61300373, -31.06941473 17.15300373, -5.81749931, 0.95061602



Čtvrtá pásmová zádrž:

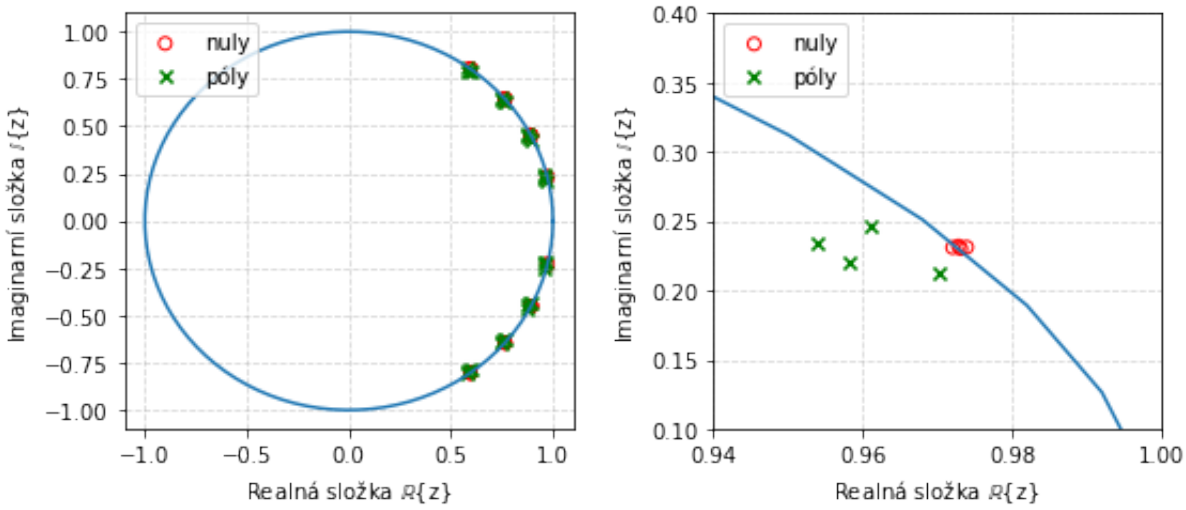
a: 1, -4.70230634, 12.19073637, -20.24967059 23.77777626, -19.74179834, 11.58690192 -4.35728984, 0.90338898

b: 0.95046777, -4.52687711, 11.88708069, -19.99865545, 23.78370663, -19.99865545 11.88708069, -4.52687711, 0.95046777



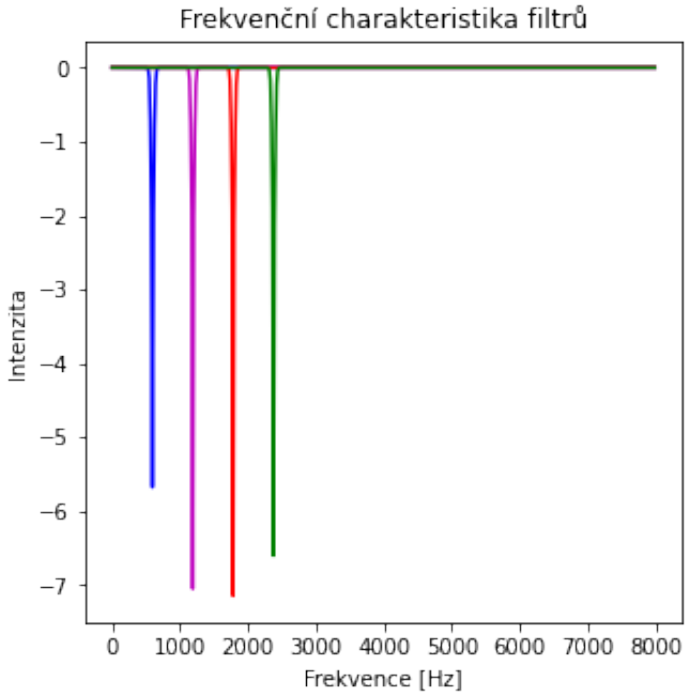
8 Nulové body a póly

Jsem spočítal pomocí funkce `tf2zpk`. Levý screen znázorňuje jednotkovou kružnici vyzobrazenou se všemi nulovými body a póly. Pravý screen přináší detailnější pohled.



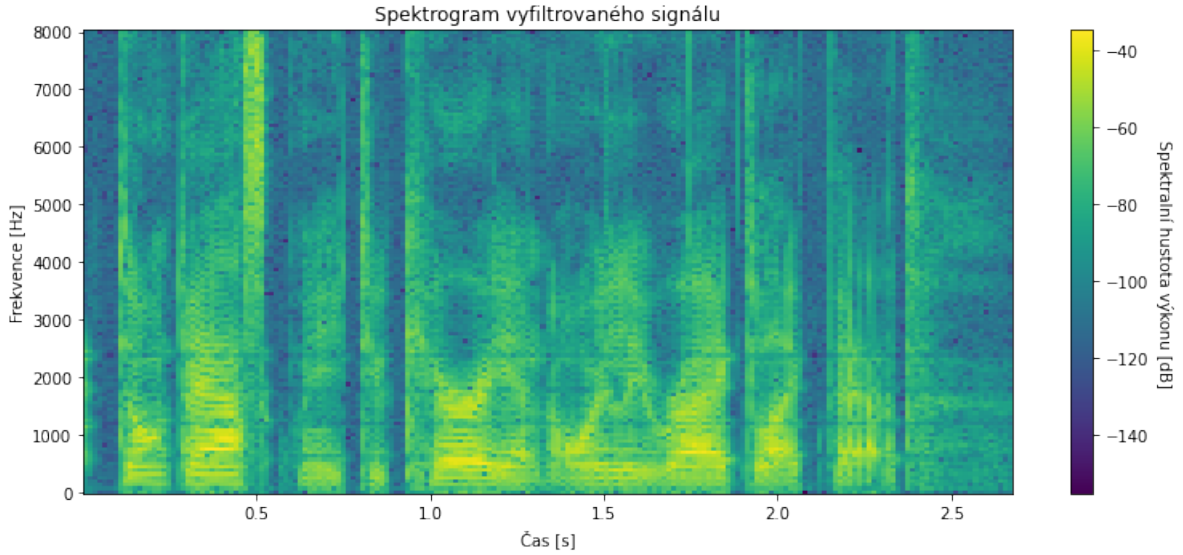
9 Frekvenční charakteristika

Jsem určil funkcí `freqz` do které jsem dosadil koeficienty `a` a `b`. Můžeme vidět, že potlačuje rušivý signál na správných frekvencích.



10 Filtrace

Pomocí navržených 4 pásmových zadrží a funkce `signal.lfilter` jsem signál vyfiltroval a dostal následující výsledek. Výsledek je uložen v souboru `/audio/clean_bandstop.wav`.



Provedl jsem i následnou kontrolu, zda-li se výsledný signál nachází v příslušném dynamickém rozsahu.