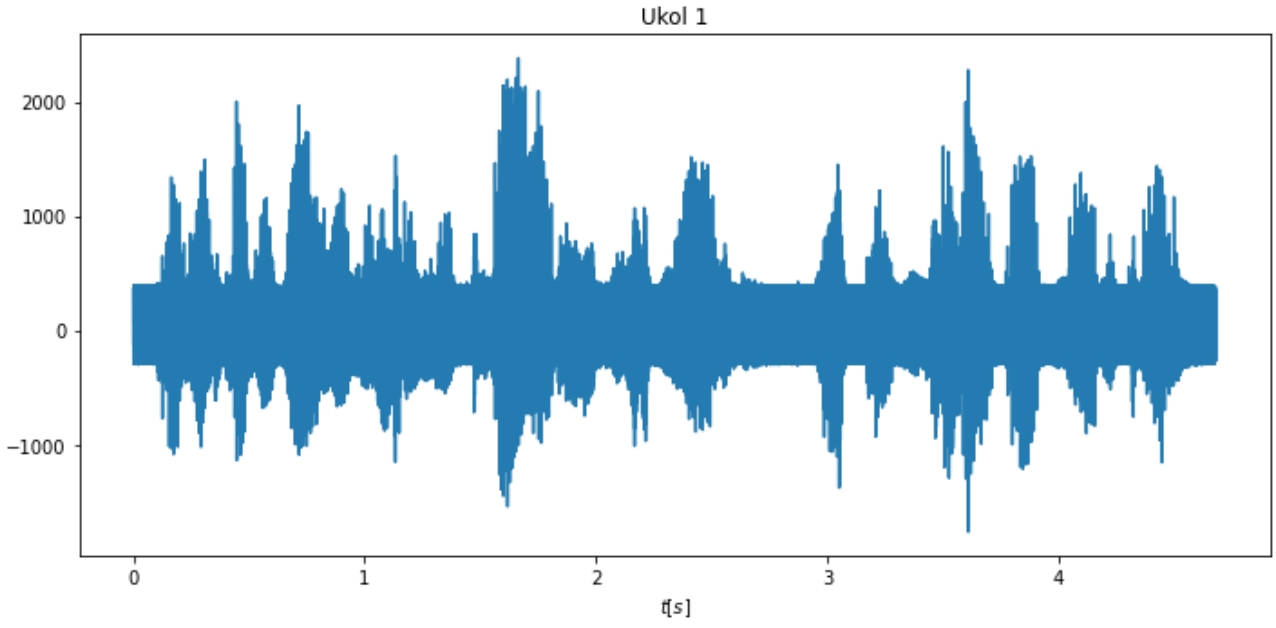
Protokol
Tony Pham (xphamt00)

1. Úloha

Signál jsem načetl pomocí wavfile.read. Počet vzorků jsem vypočítal vydělením počtu vzorků a fs.

```
print("Delka signalu:", data.size / fs, "[s]")
print("Pocet vzorku signalu:", data.size)
print("Minimální hodnota:", np.min(data))
print("Maximální hodnota:", np.max(data))
```

```
Delka signalu: 4.6784375 [s]
Pocet vzorku signalu: 74855
Minimální hodnota: -1765
Maximální hodnota: 2384
```



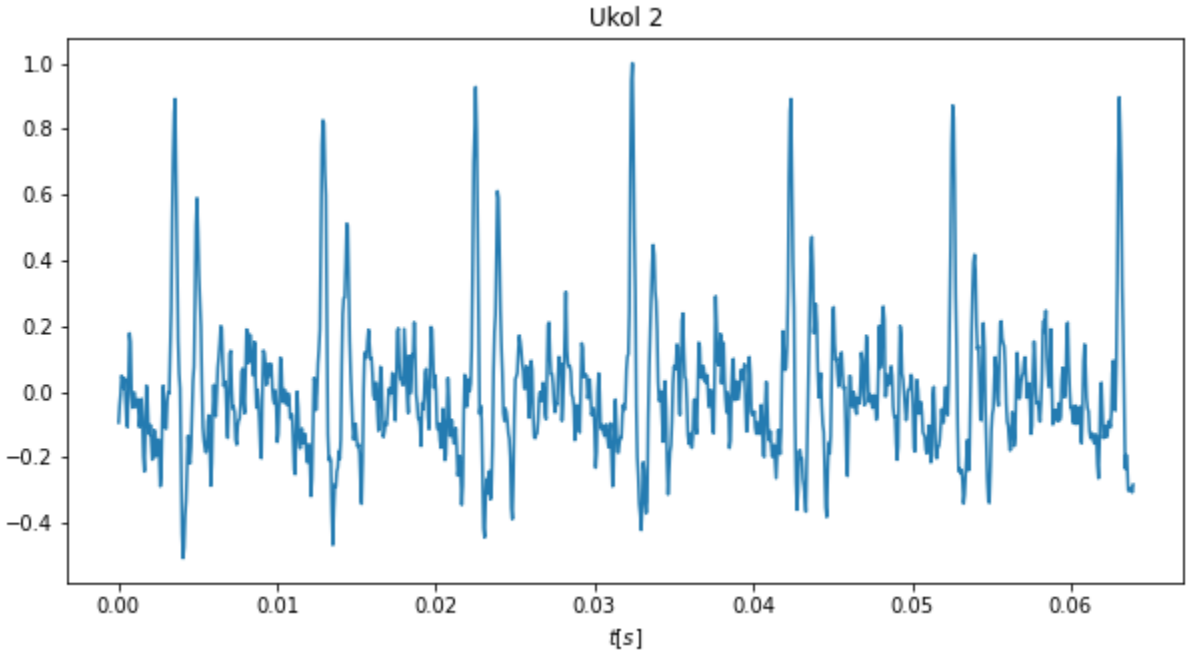
2. Úloha

Nejprve jsem si vypočítal střední hodnotu a odečetl ji. A poté jsem provedl normalizaci.

```
#stredni hodnota
meanValue = np.mean(data)

print("Stredni hodnota:", meanValue)

data = data - meanValue

#normalizace
data = data / max(abs(np.min(data)),np.max(data))
```

Potom jsem si rozdělil signál na rámce.

```
frames = np.array([data[i*overlap:i*overlap + frame_size] for i in range(len(data) //overlap - frame_size//overlap + 1)])
```

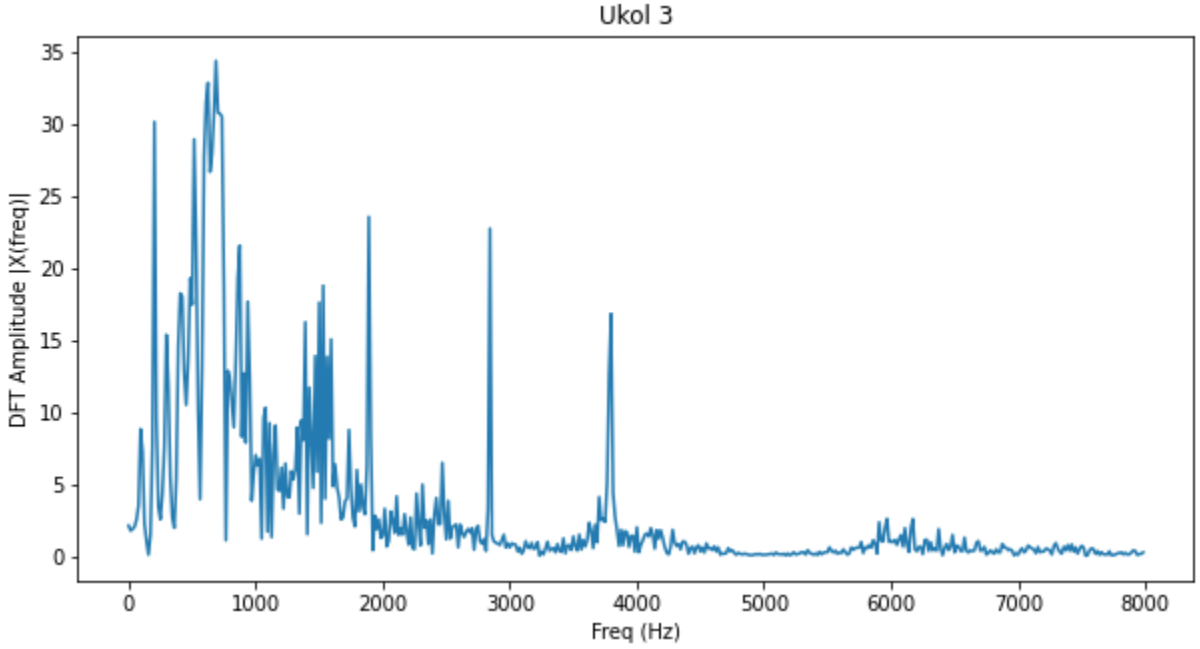


3. Úloha

Vytvořil jsem si DFT funkci.

```
def DFT(x):

    N = len(x)
    n = np.arange(N)
    k = n.reshape((N, 1))
    e = np.exp(-2j * np.pi * k * n / N)

    X = np.dot(e, x)

    return X
```

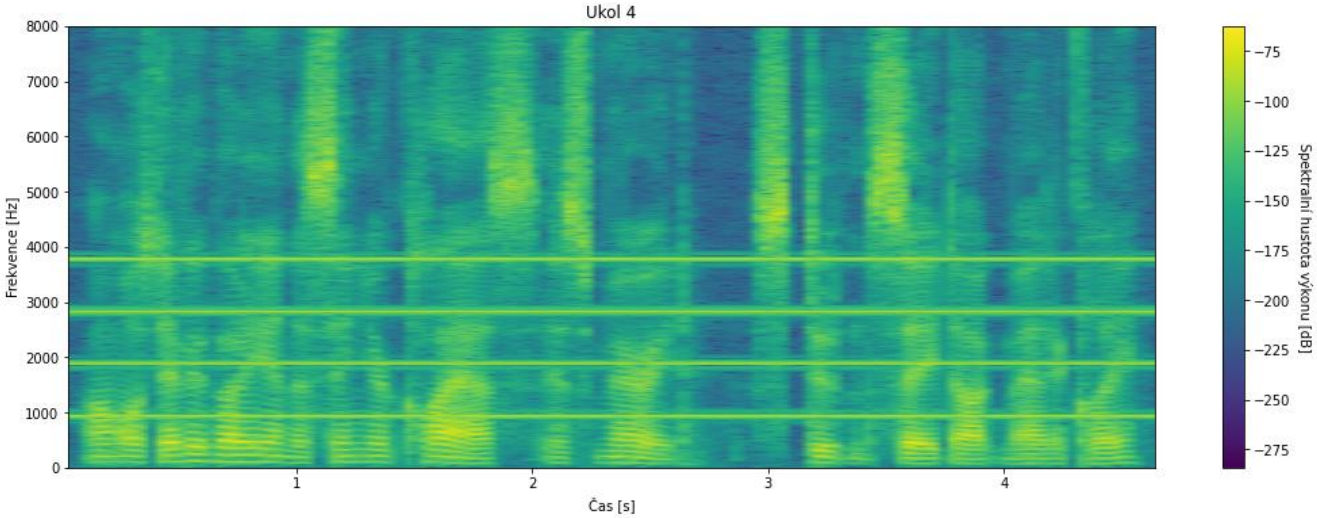
Po zavolání funkce na rámec 51 graf vypadá takto.



4. Úloha

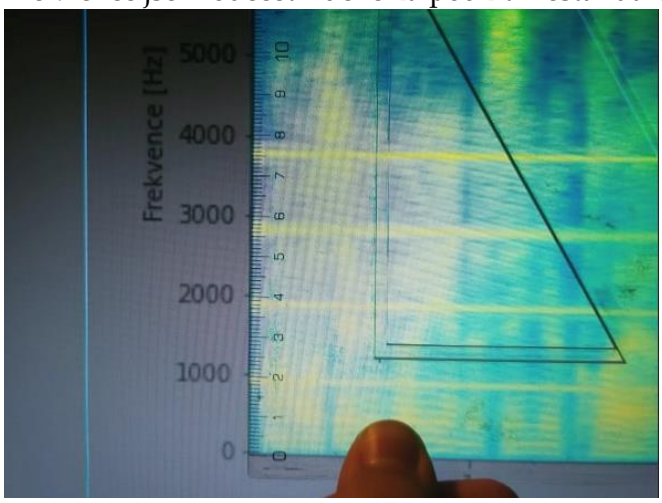
K vytvoření spektrogramu jsem využil funkci spektrogram () a inspiroval jsem se kódem z: https://nbviewer.org/github/zmolikova/ISS_project_study_phase/blob/master/Zvuk_spektra_filtrace.ipynb

```
f, t, sgr = spectrogram(data, fs, nperseg=frame_size, noverlap=overlap)
```



5. Úloha

Frekvence jsem odečetl ručně za použití nestandartní ale efektivní metody



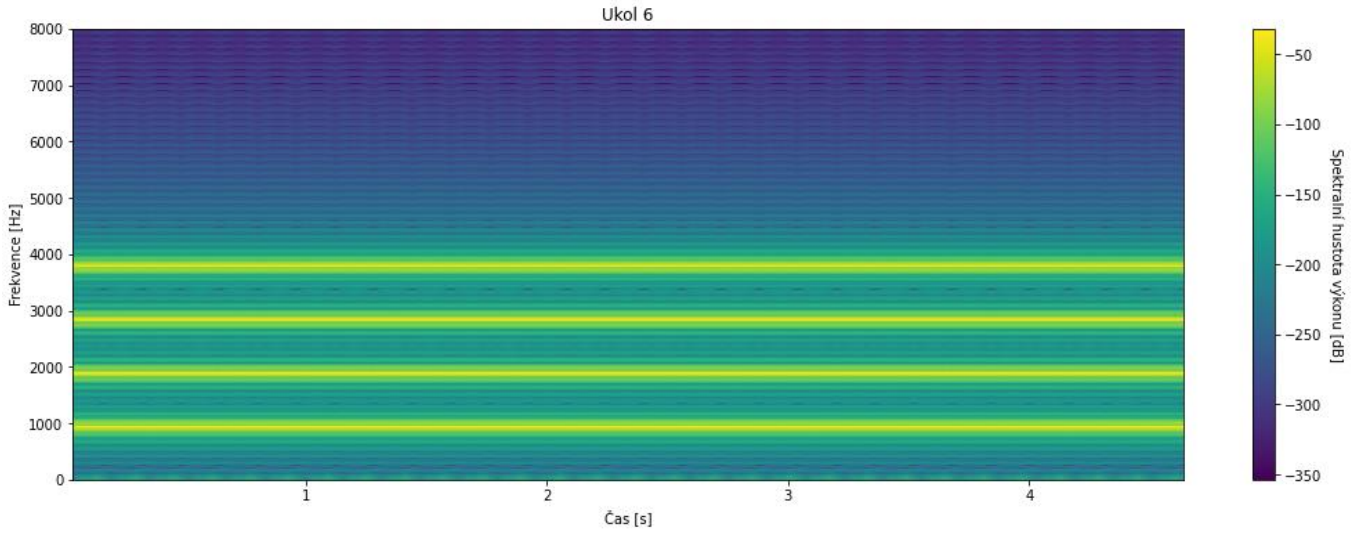

```
Ukol 5: Dělal jsem to ručně. Přiloži jsem pravítko k monitoru -> 1000hz = 2cm:
f1 = 1,9cm = 950 hz
f2 = 3,8cm = 1900 hz
f3 = 5,7cm = 2850 hz
f4 = 7,6cm = 3800 hz
```

6. Úloha

Vygeneroval jsem si cos. pro f1 až f4 a pak jsem je sečetl a vygeneroval .wav výstup

```
cos_1 = np.cos(2*np.pi * 950 * np.arange(data.size) / fs)
cos_2 = np.cos(2*np.pi * 1900 * np.arange(data.size) / fs)
cos_3 = np.cos(2*np.pi * 2850 * np.arange(data.size) / fs)
cos_4 = np.cos(2*np.pi * 3800 * np.arange(data.size) / fs)

final_cos = cos_1 + cos_2 + cos_3 + cos_4
wavfile.write("audio/4cos.wav", fs, final_cos)
```

A výsledný spektrogram final_cos vypadal následovně:



7. Úloha

Tento úkol jsem se rozhodl zhotovit pomocí 4 pásmových zádrží.
Pro vytvoření bandstop funkce jsem se inspirovat zde: https://bechelli.org/signal-and-filter-bandstop.html

A vypadá následovně:

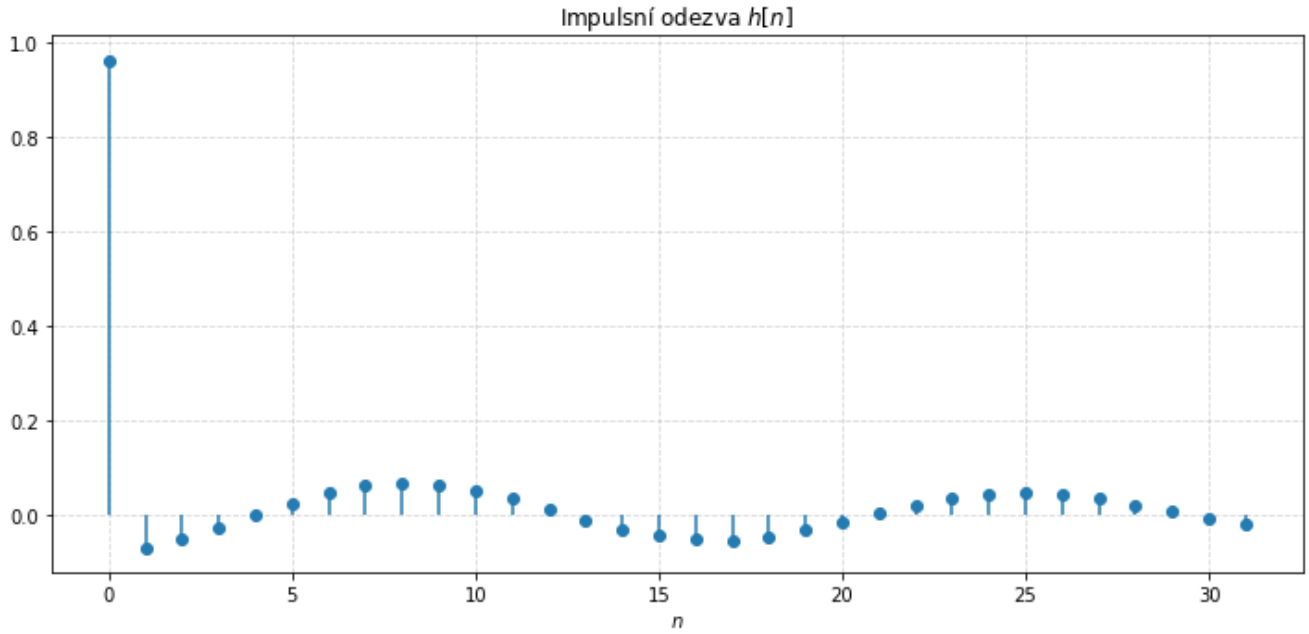
```
def butter_bandstop(lowcut, highcut, sampling_rate, order=3):
    nyq = 0.5 * sampling_rate
    low = lowcut / nyq
    high = highcut / nyq
    b, a = signal.butter(order, [low, high], btype='bandstop')
    return a, b
```

Nakonec jsem funkci volal na každou cos. a výsledek vypadal následovně:

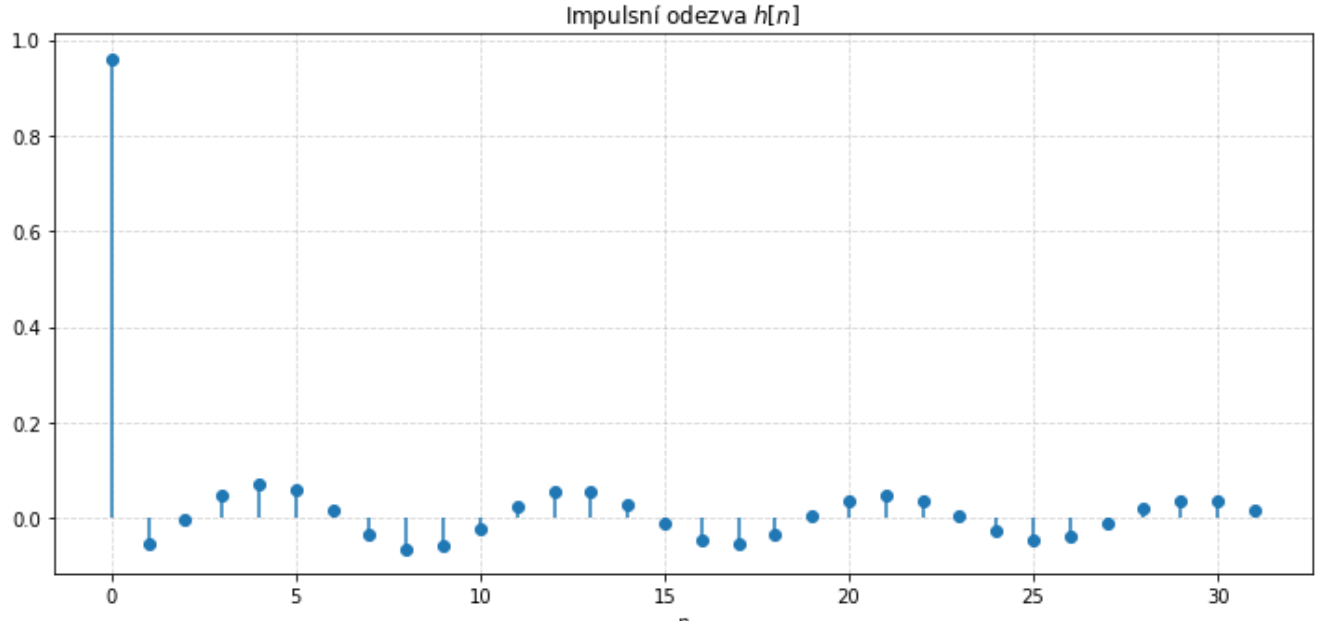
```
# impulsni odezva
N_imp = 32
imp = [1, *np.zeros(N_imp-1)]
```

```
a, b = butter_bandstop(900, 1000, fs)
```

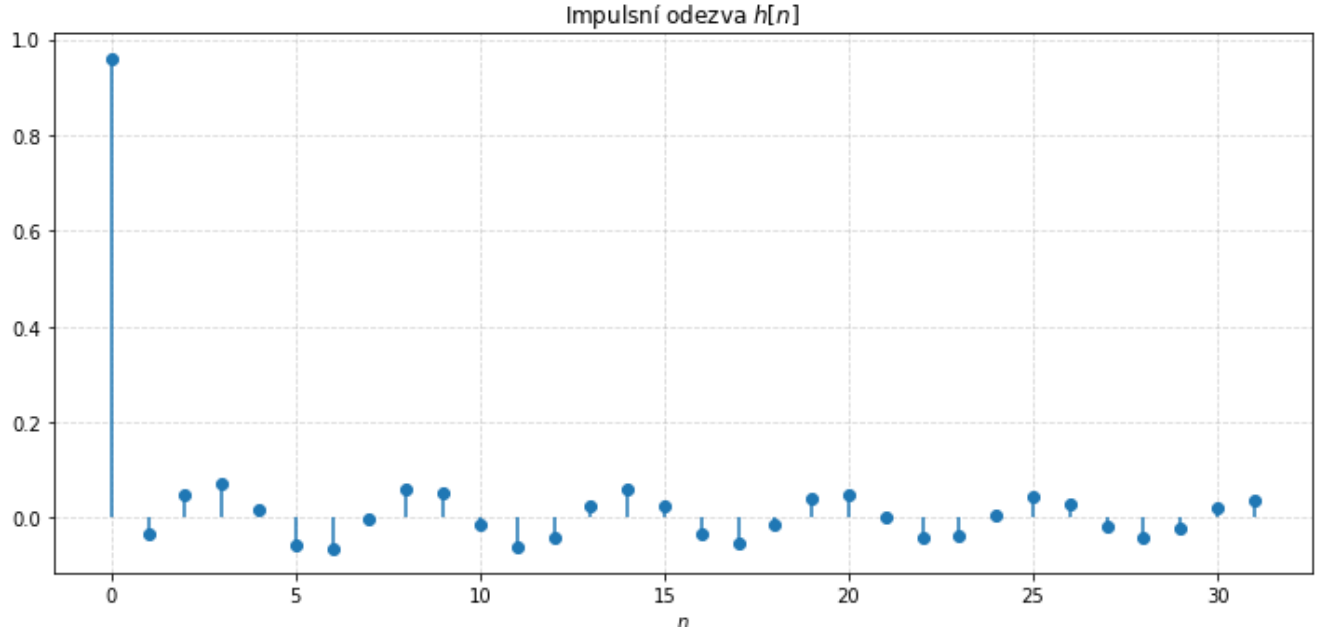
```
Koeficien a: [  1.          -5.51521996  13.06154781 -16.95845513  12.72399671
  -5.23384624   0.92446058]
Koeficien b: [  0.96148873  -5.37315173  12.89351382 -16.96121786  12.89351382
  -5.37315173   0.96148873]
```



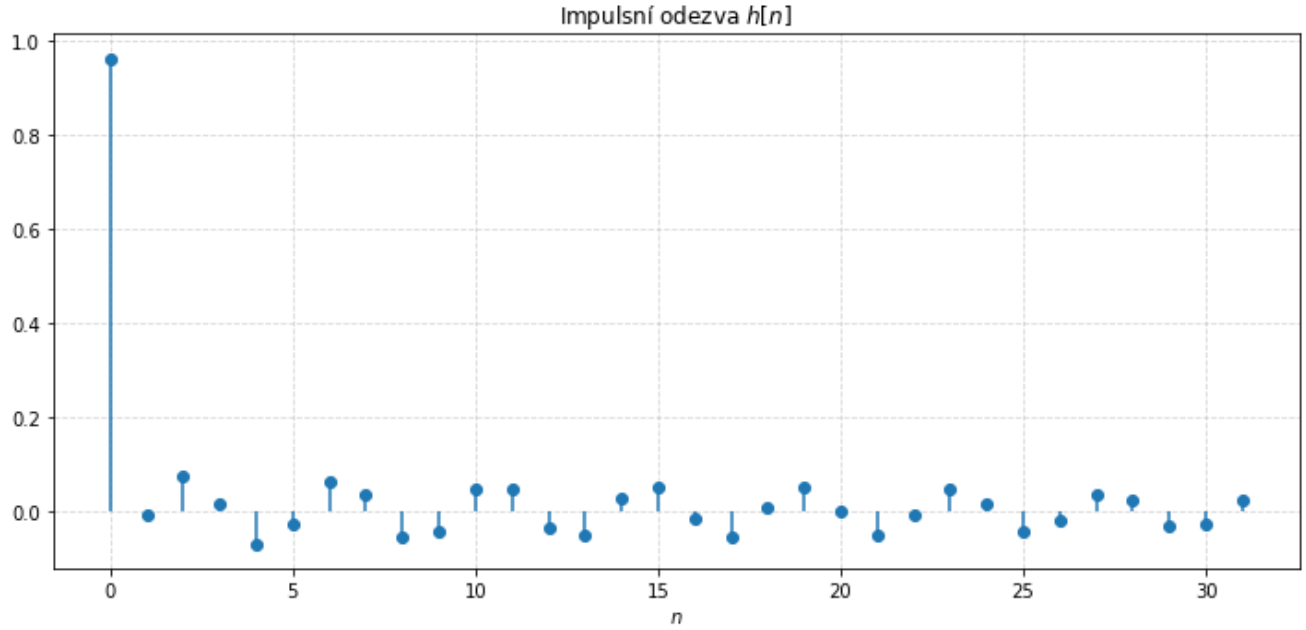
```
Koeficien a: [  1.          -4.34910353   9.22690394 -11.51948388   8.98845902
  -4.12722236   0.92446058]
Koeficien b: [  0.96148873  -4.23707365   9.10842304 -11.52166247   9.10842304
  -4.23707365   0.96148873]
```




```
Koeficien a: [ 1.         -2.58468036  5.14851284 -5.67459332  5.01547374 -2.45281597
  0.92446058]
Koeficien b: [ 0.96148873 -2.51810079  5.08273485 -5.67588806  5.08273485 -2.51810079
  0.96148873]
```



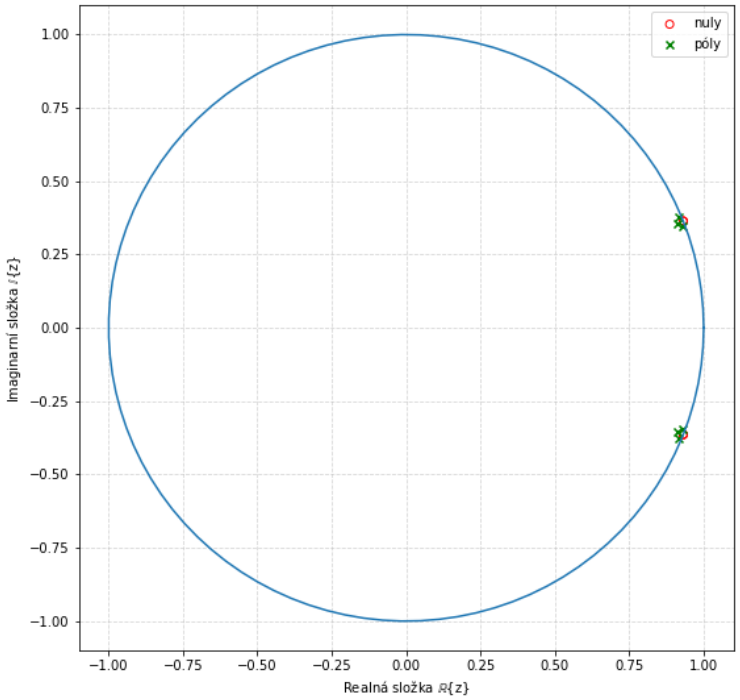
```
Koeficien a: [ 1.         -0.46468238  2.99344794 -0.90890939  2.91610638 -0.44097536
  0.92446058]
Koeficien b: [ 0.96148873 -0.45271248  2.95551872 -0.90914216  2.95551872 -0.45271248
  0.96148873]
```

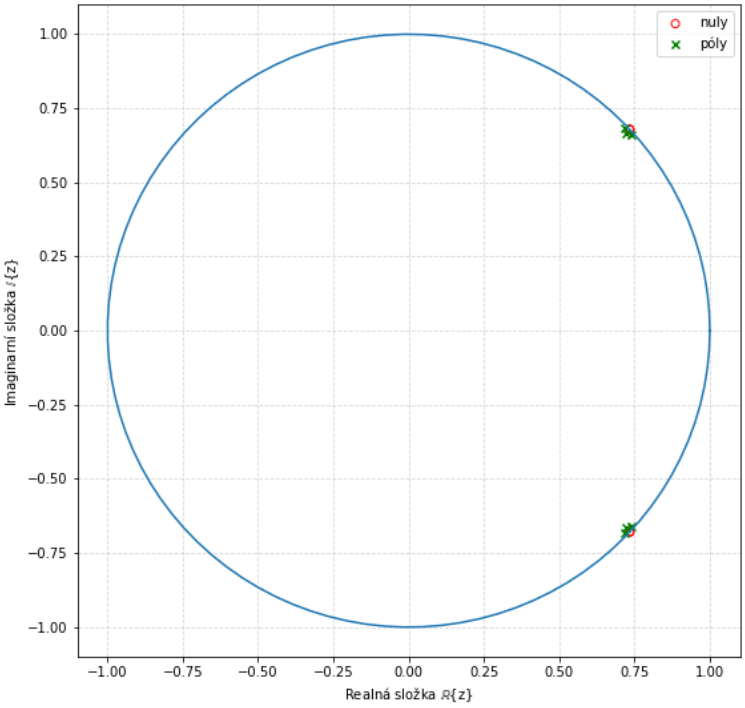


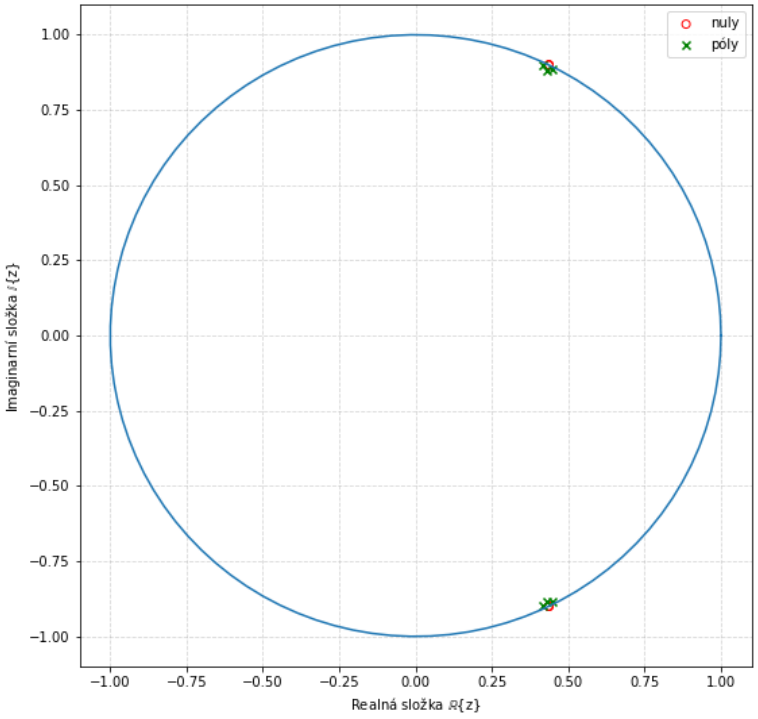
8. Úloha

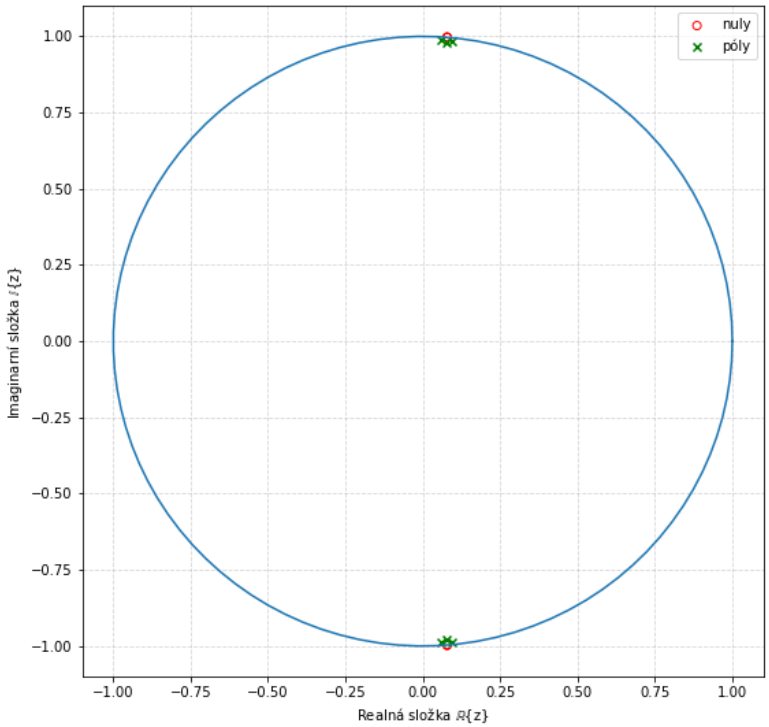
Při vypracování tohoto úkolu jsem se inspiroval zde:
https://nbviewer.org/github/zmolikova/ISS_project_study_phase/blob/master/Zvuk_spektra_filtrace.ipynb

```
a, b = butter_bandstop(900, 1000, fs)
z, p, k = tf2zpk(b, a)
plt.figure(figsize=(8,7.6))

# jednotkova kruznice
ang = np.linspace(0, 2*np.pi,100)
plt.plot(np.cos(ang), np.sin(ang))

# nuly, poly
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')
```


nuly
póly
Imaginární složka $\mathcal{I}\{z\}$
Realná složka $\mathcal{R}\{z\}$


nuly
póly
Imaginární složka $\mathcal{I}\{z\}$
Realná složka $\mathcal{R}\{z\}$

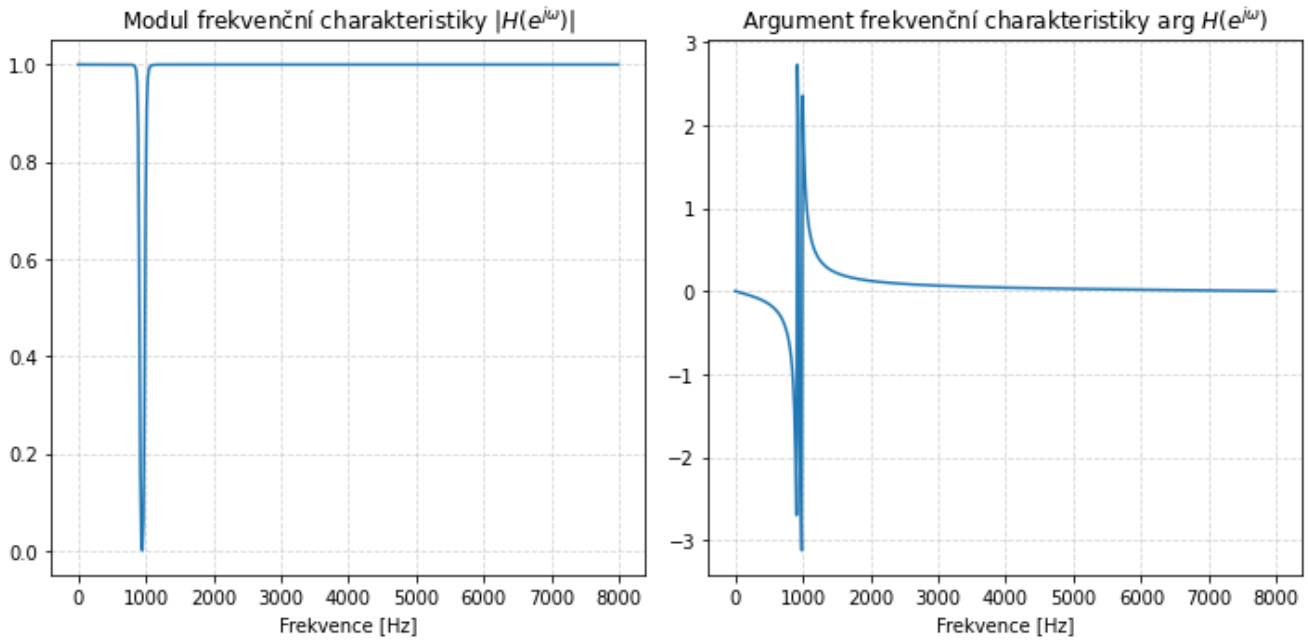

nuly
póly
1.00
0.75
0.50
0.25
0.00
−0.25
−0.50
−0.75
−1.00
−1.00 −0.75 −0.50 −0.25 0.00 0.25 0.50 0.75 1.00
Imaginarní složka $\Im\{z\}$
Realná složka $\Re\{z\}$

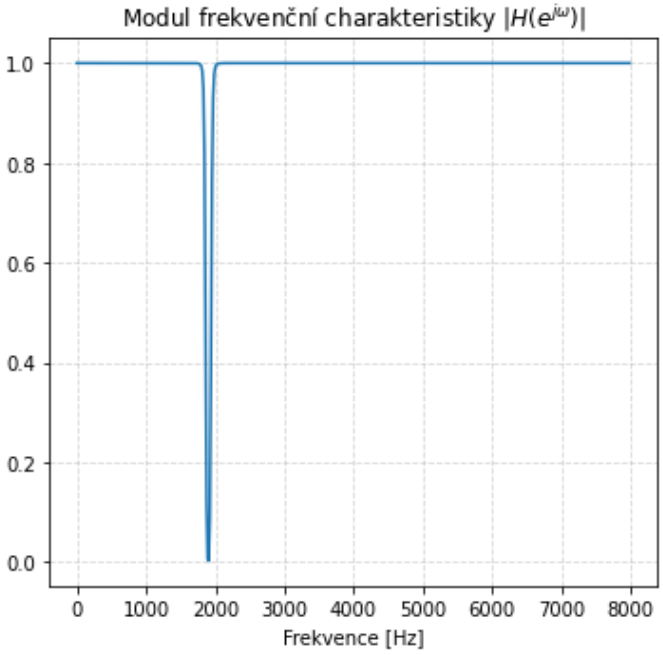

nuly
póly
1.00
0.75
0.50
0.25
0.00
−0.25
−0.50
−0.75
−1.00
−1.00 −0.75 −0.50 −0.25 0.00 0.25 0.50 0.75 1.00
Imaginarní složka $\Im\{z\}$
Realná složka $\Re\{z\}$

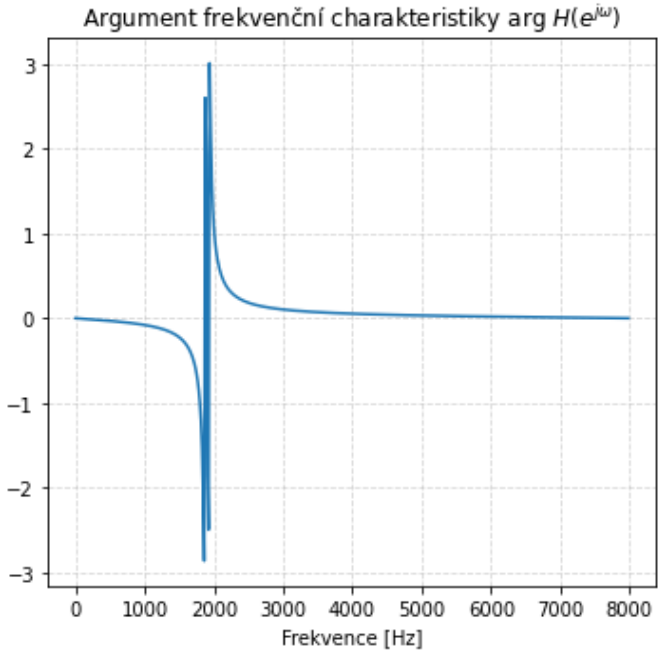
9. Úloha

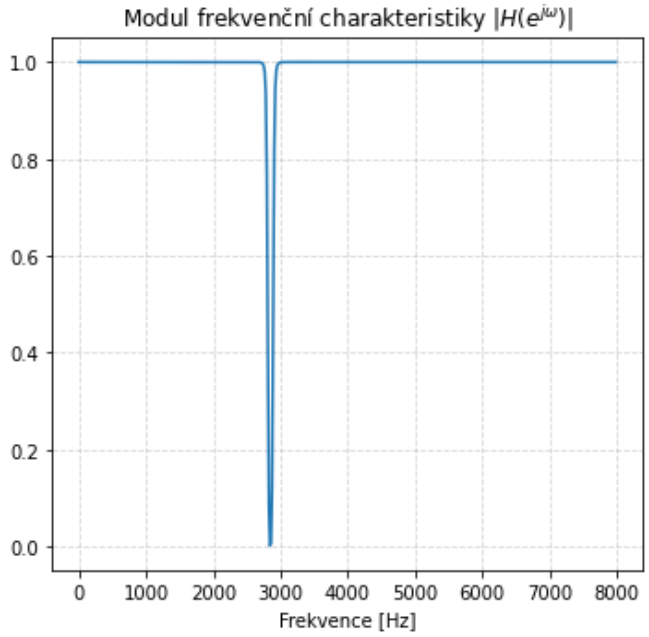
Při vypracování tohoto úkolu jsem se inspiroval zde:
https://nbviewer.org/github/zmolikova/ISS_project_study_phase/blob/master/Zvuk_spektra_filtrace.ipynb

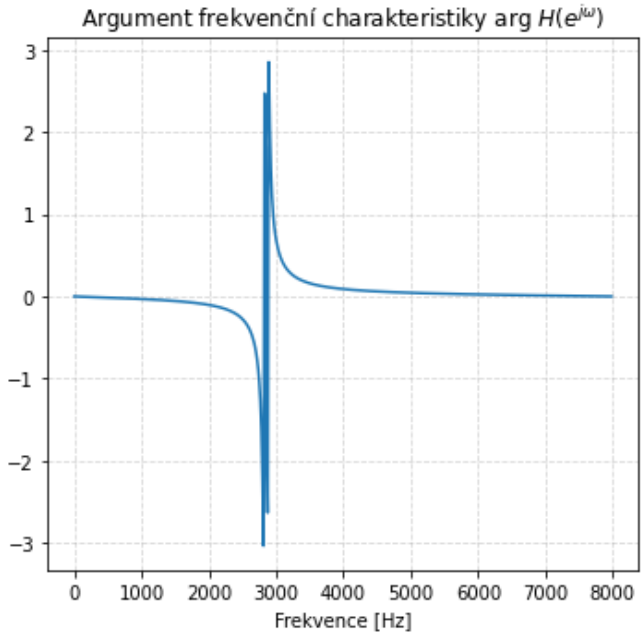
```
w, H = freqz(b, a)
_, ax = plt.subplots(1, 2, figsize=(10,5))

ax[0].plot(w / 2 / np.pi * fs, np.abs(H))
ax[0].set_xlabel('Frekvence [Hz]')
ax[0].set_title('Modul frekvenční charakteristiky $|H(e^{j\omega})|$')

ax[1].plot(w / 2 / np.pi * fs, np.angle(H))
ax[1].set_xlabel('Frekvence [Hz]')
ax[1].set_title('Argument frekvenční charakteristiky $\mathrm{arg}\ H(e^{j\omega})$')

for ax1 in ax:
    ax1.grid(alpha=0.5, linestyle='--')

plt.tight_layout()
```

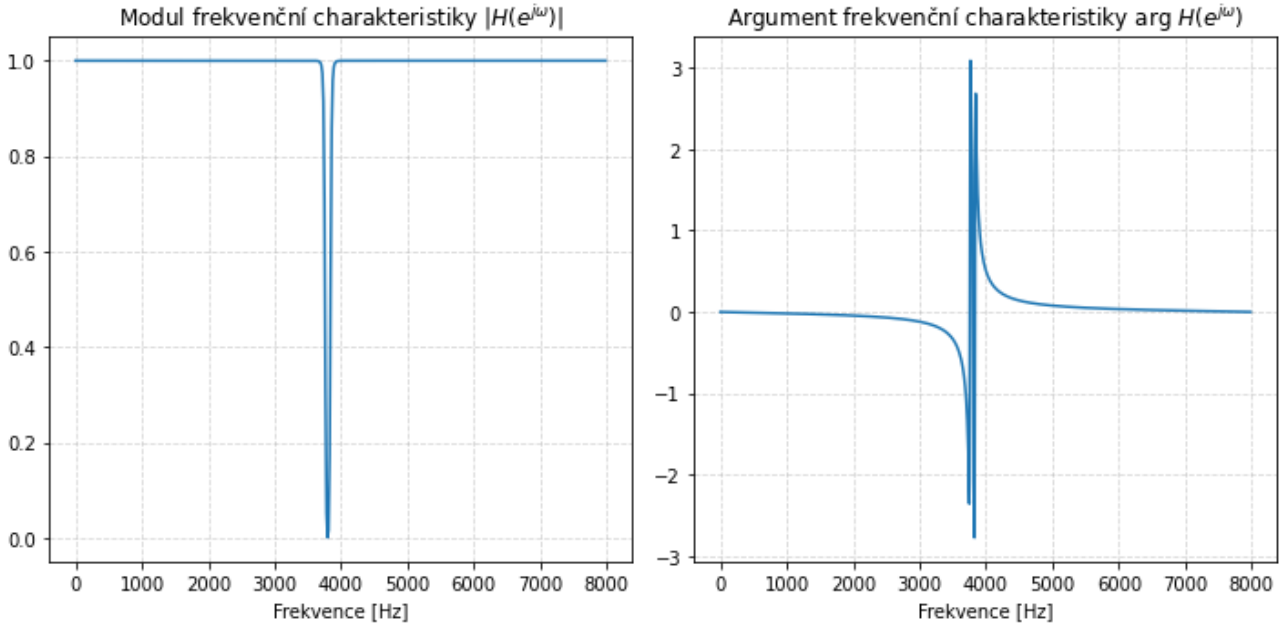



Modul frekvenční charakteristiky $|H(e^{j\omega})|$
1.0
0.8
0.6
0.4
0.2
0.0
0 1000 2000 3000 4000 5000 6000 7000 8000
Frekvence [Hz]


Argument frekvenční charakteristiky arg $H(e^{j\omega})$
3
2
1
0
−1
−2
−3
0 1000 2000 3000 4000 5000 6000 7000 8000
Frekvence [Hz]


Modul frekvenční charakteristiky $|H(e^{j\omega})|$
1.0
0.8
0.6
0.4
0.2
0.0
0 1000 2000 3000 4000 5000 6000 7000 8000
Frekvence [Hz]


Argument frekvenční charakteristiky arg $H(e^{j\omega})$
3
2
1
0
−1
−2
−3
0 1000 2000 3000 4000 5000 6000 7000 8000
Frekvence [Hz]

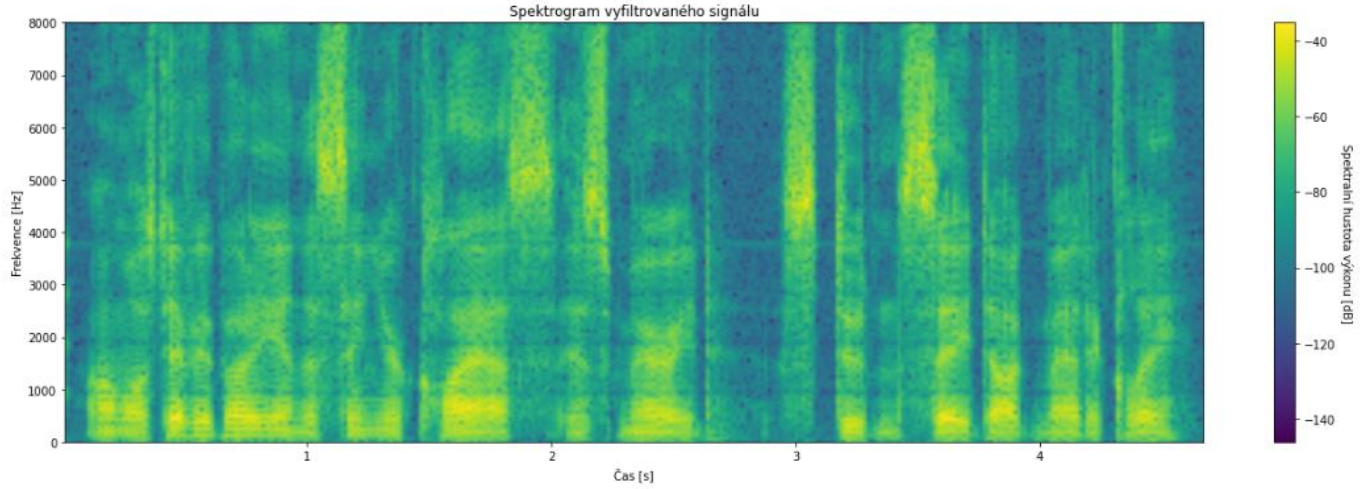


10. Úloha

Filtraci jsem zavolal nad každým koeficientem.

```
# filtrace
sf = lfilter(b, a, sf)
f, t, sfgr = spectrogram(sf, fs)
sfgr_log = 10 * np.log10(sfgr+1e-20)
```

A po vyfiltrování šumu jsem si zobrazil spektrogram, abych si ověřil, že jsou skutečně pryč.



Výsledek mě překvapil, slyšel jsem krásně čistý hlas, který byl zároveň hlasitější než původní nahrávka, bez rušivých šumů.