Vilém Gottwald [xgottw07] 27.12.2021

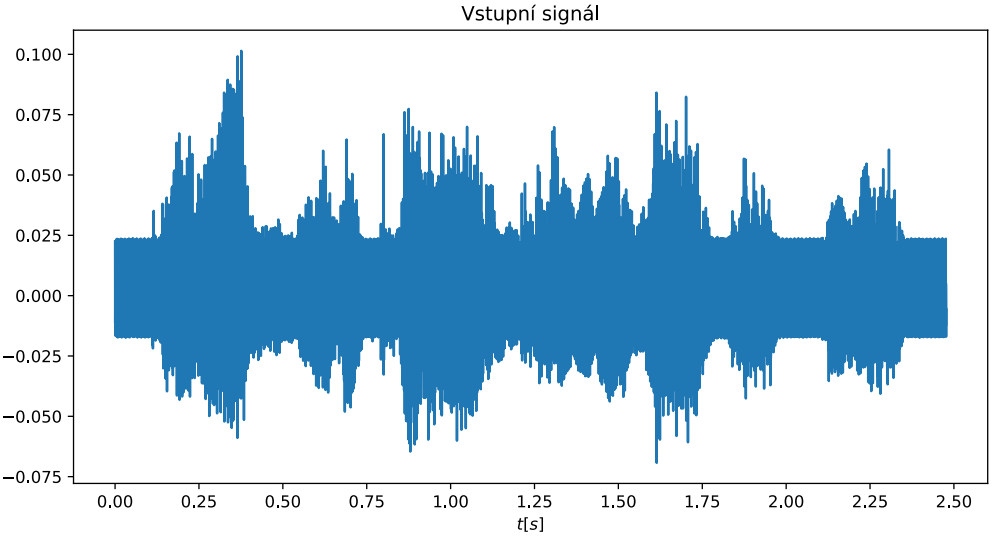
ISS Projekt 2021/2022

1 ZÁKLADY

Signál byl načten pomocí `wavfile.read` a následně vynormován maximální hodnotou datového typu `int` do rozsahu <-1, 1>. Délka signálu v sekundách byla vypočítána jako podíl počtu vzorků signálu a vzorkovací frekvence. Minimum a maximum signálu bylo zjištěno funkcemi `min` a `max`.

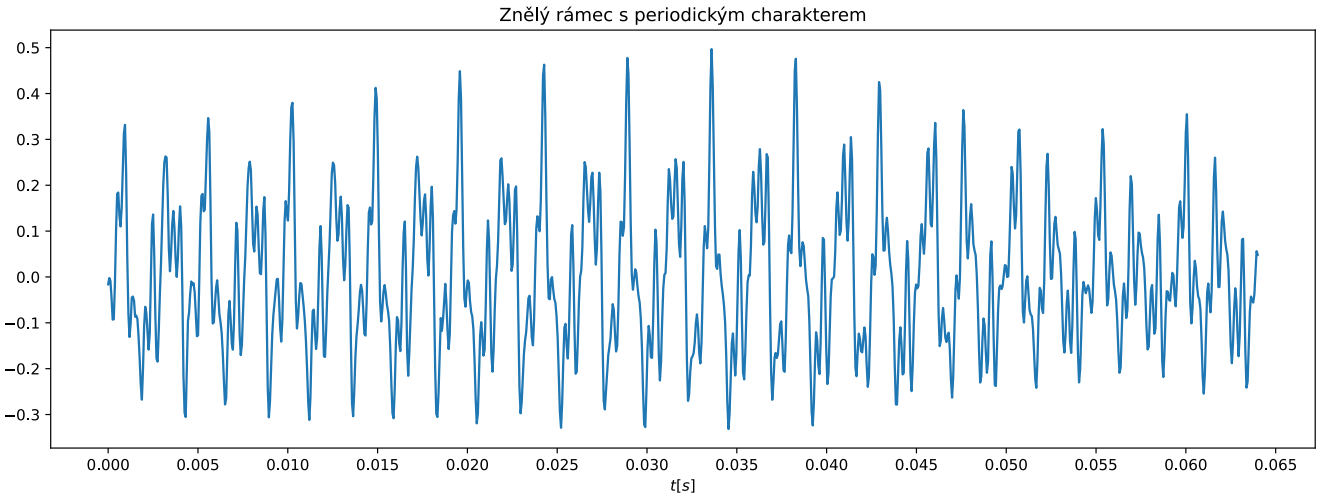
Výsledky:

Délka signálu ve vzorcích: 39629 vzorků
Délka signálu v sekundách: 2.4768125 s
Maximální hodnota: 0.101470947265625
Minimální hodnota: -0.06927490234375



2 PŘEDZPRACOVÁNÍ A RÁMCE

Jako pěkný rámec s periodickým charakterem jsem zvolil rámec s indexem 43. Znělost rámce jsem pro jistotu dodatečně ověřil vytvořením audio souboru a následným poslechem nahrávky.



3 DFT

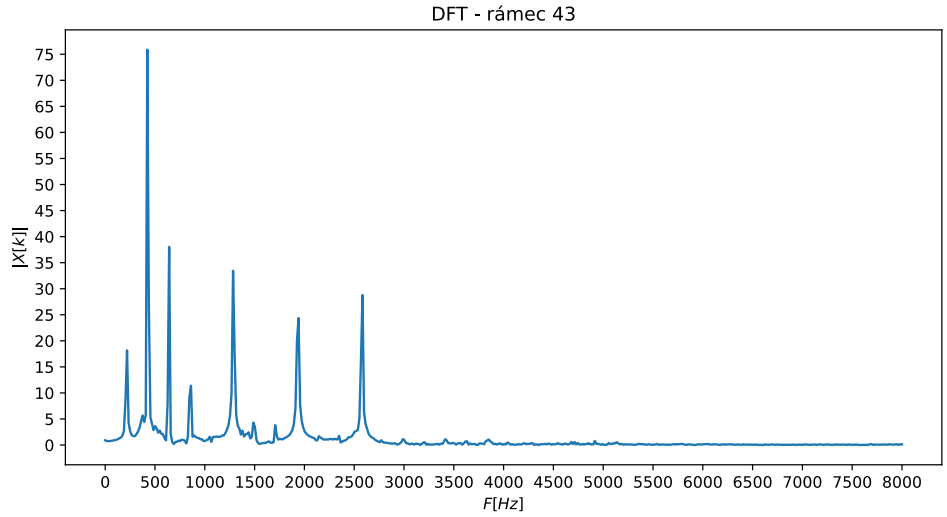
Funkce byla implementována na základě algoritmu popsaného v [4].

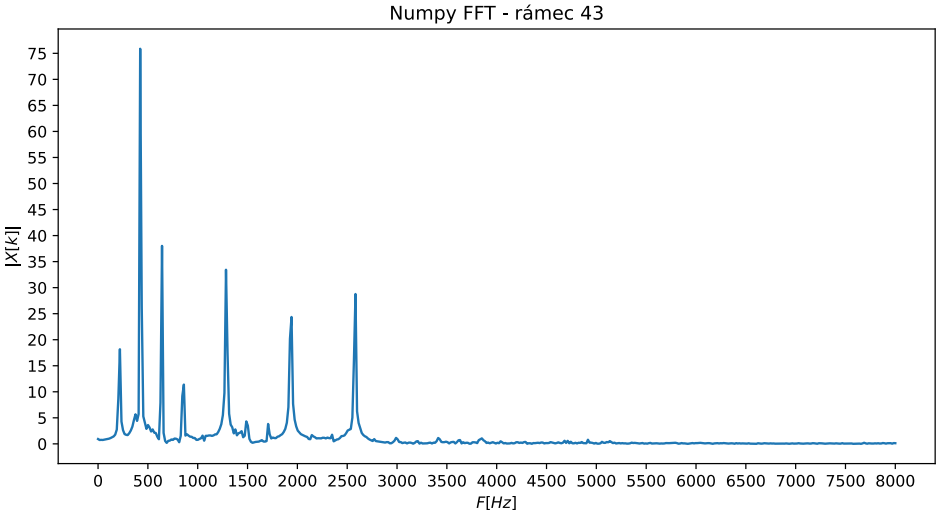
```
# Vytvoreni komlexniho koeficientu (twiddle factor)
N = 1024
W_N = np.e**(-2j*np.pi/N)

# Vytvoreni matice bazi
TF_matrix = []
for n in range(N):
	col = []
	for k in range(N):
		col.append(W_N**(n*k))
	TF_matrix.append(col)

# DFT jako skalarni soucin matice bazi s vektorem ramce 43
DFT = np.array(TF_matrix).dot(np.array(frames[43]))
```

Algoritmus spočívá v tom, že si nejdříve vytvoříme komplexní exponenciálu, jejíž mocninami následně naplníme matici bází. Skalárním součinem matice bází a vektoru námi zvoleného rámce získáme vektor koeficientů DFT.

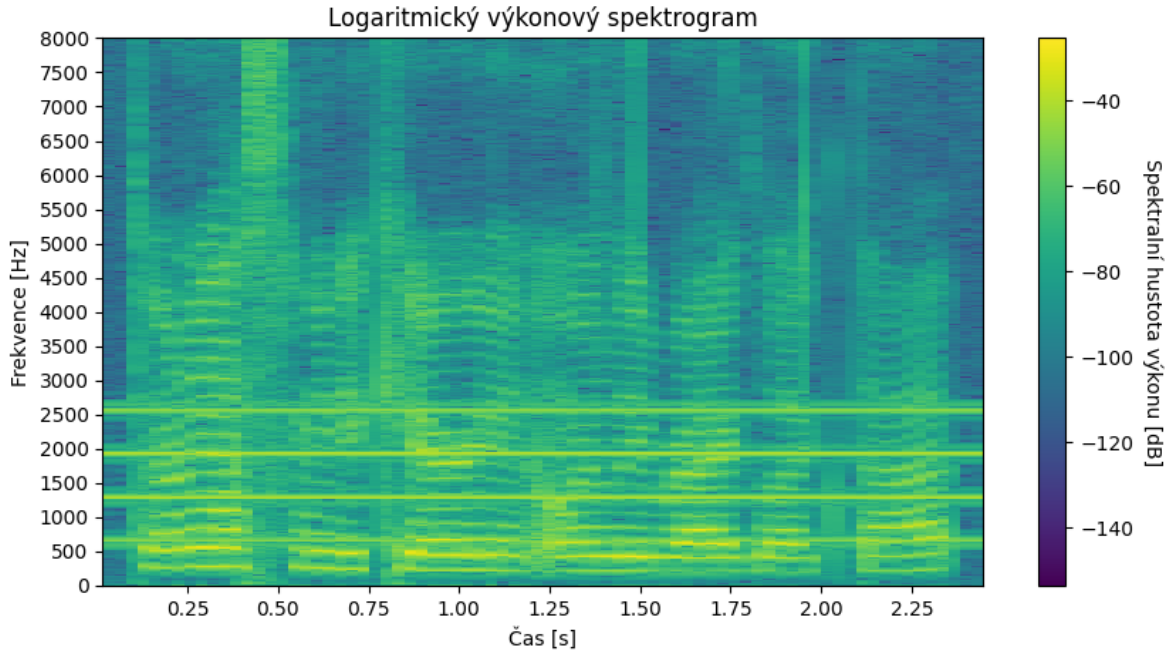




Na pohled jsou výsledky vlastní implementace DFT a knihovní implementace FFT shodné. Tuto skutečnost jsem ověřil i pomocí funkce `np.allclose`, která shodnost obou výsledků potvrdila.

4 SPEKTROGRAM

Spektrogram celého signálu byl vytvořen pomocí funkce `spectrogram` a koeficienty byly následně upraveny podle vzorce uvedeného na ISS stránkách Katky Žmolíkové.



Ve spektrogramu můžeme pozorovat 4 rušivé komponenty – nejlépe jsou vidět na začátku a na konci signálu, protože se zde nevyskytuje hlasová aktivita.

5 URČENÍ RUŠIVÝCH FREKVENCÍ

Ze spektrogramu je patrné, že rušivé komponenty mají spektrální hustotu výkonu větší než -50 dB. Pokud tedy vybereme ze spektra v čase 0 pouze tyto frekvence, dostaneme následující hodnoty:

```
frekvence:  640.625 Hz, hodnota: -39.047548 dB, index:  41
frekvence:  656.250 Hz, hodnota: -47.989973 dB, index:  42
frekvence: 1281.250 Hz, hodnota: -40.173255 dB, index:  82
frekvence: 1296.875 Hz, hodnota: -43.770534 dB, index:  83
frekvence: 1921.875 Hz, hodnota: -42.216177 dB, index: 123
frekvence: 1937.500 Hz, hodnota: -41.135650 dB, index: 124
frekvence: 2562.500 Hz, hodnota: -45.482056 dB, index: 164
frekvence: 2578.125 Hz, hodnota: -39.547722 dB, index: 165
```

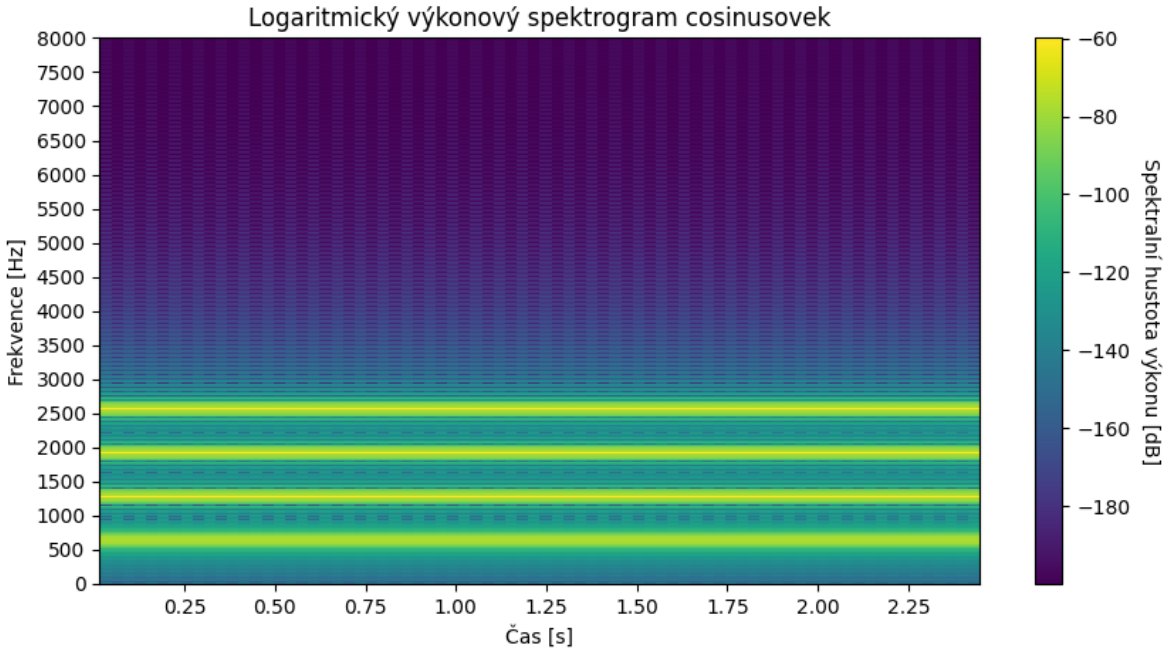
Následně je potřeba vždy vybrat jednu ze dvojice frekvencí s blízkými hodnotami, přičemž jsem zvolil frekvenci s největší spektrální hustotou výkonu (640, 625 Hz) a její násobky.

Výslednými rušivými frekvencemi tedy jsou:

$f_1 = 640.625$ Hz, $f_2 = 1281.250$ Hz, $f_3 = 1921.875$ Hz, $f_4 = 2562.500$ Hz

6 GENEROVÁNÍ SIGNÁLU

Signál byl vygenerován jako součet 4 cosinusovek vynormovaný do rozsahu <-1, 1>. Audio soubor byl vytvořen funkcí `wawfile.write` za použití délky a vzorkovací frekvence původního signálu. Tato nahrávka však byla velmi hlasitá, takže jsem z grafu původního signálu v části bez hlasové aktivity vyčetl amplitudu signálu a pronásobil jí normovaný signál cosinusovek. Po této úpravě již signál zní stejně jako pískání v původní nahrávce.



Správnost určených frekvencí je patrná i ze spektrogramu cosinusovek, jelikož se rušivé komponenty vyskytují na stejných frekvencích jako ve spektrogramu původního signálu.

7 ČISTÍCÍ FILTR – PÁSMOVÉ ZÁDRŽE

Filtr byl navržen jako 4 samostatné pásmové zádrže pro potlačení jednotlivých rušivých frekvencí.

```
# Nyquistova frekvence
nyq = fs/2
sos_l = []

for i in range(len(freqs)):
	# Vytvoreni hranic zaverneho a propustneho pasma
	pass_min = (freqs[i]  - 15 - 50) / nyq
	pass_max = (freqs[i]  + 15 + 50) / nyq
	stop_min = (freqs[i] - 15) / nyq
	stop_max = (freqs[i] + 15) /nyq

	# Vytvoreni filtru
	N, Wn = buttord([pass_min, pass_max], [stop_min, stop_max], 3, 40)
	sos_l.append(butter(N, Wn, 'bandstop', False, 'sos'))
```

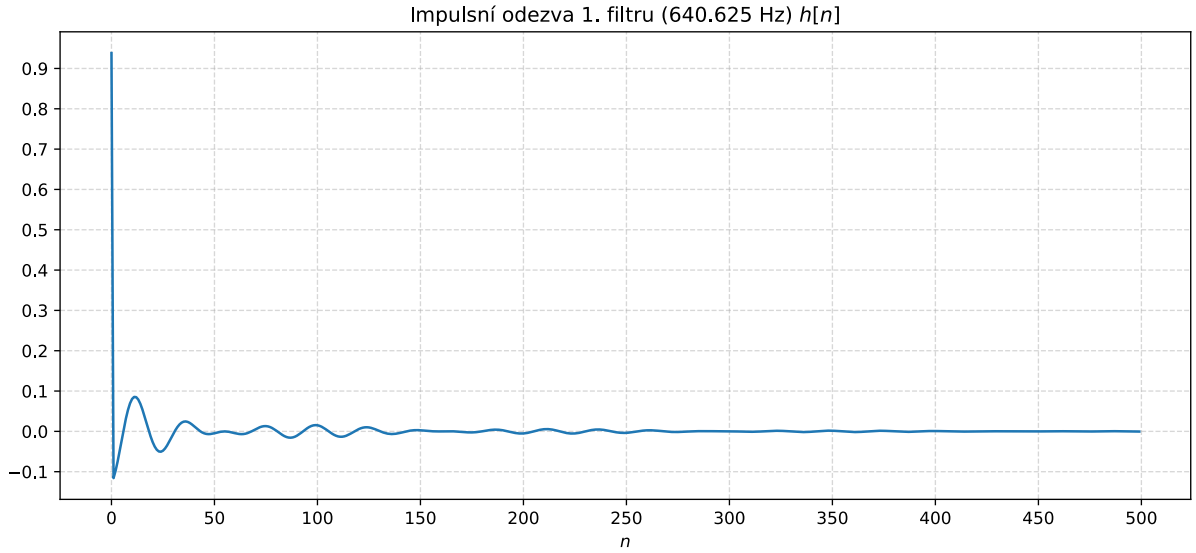
Pro každý filtr jsou nejdříve na základě doporučení v zadání vypočítány hranice zádržného a propustného pásma. Následně pomocí funkce `buttord` zjistíme řád filtru a kritické frekvence, ty jsou poté předány funkci `butter`, která vytvoří `SOS` reprezentaci filtru. `SOS` reprezentace byla zvolena, protože je ve `scipy` dokumentaci[2] doporučována z důvodu numerické přesnosti.

Z SOS reprezentace jsou pomocí funkce `sos2tf` zjištěny *a* a *b* koeficienty filtrů. Impulsní odezvy filtrů jsou získány aplikováním filtru na jednotkový impuls použitím funkce `sosfilt`.

1. Filtr - 640.625 Hz

a: 1, -7.62474816, 25.6755607, -49.8603828, 61.06453369, -48.29480685, 24.08850499, -6.92884863, 0.88020176

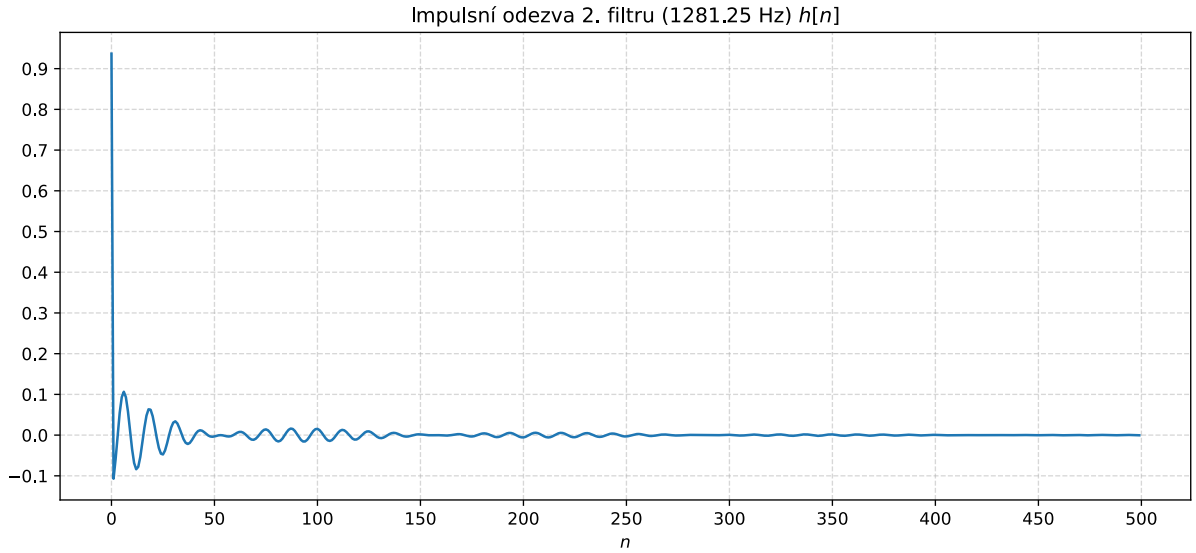
b: 0.93819068, -7.26939929, 24.87486785, -49.08499393, 61.08268408, -49.08499393, 24.87486785, -7.26939929, 0.93819068



2. Filtr - 1281.250 Hz

a: 1, -6.89437821, 21.69576196, -40.49910777, 48.94893811, -39.19897907, 20.32514331, -6.25149093, 0.87764641

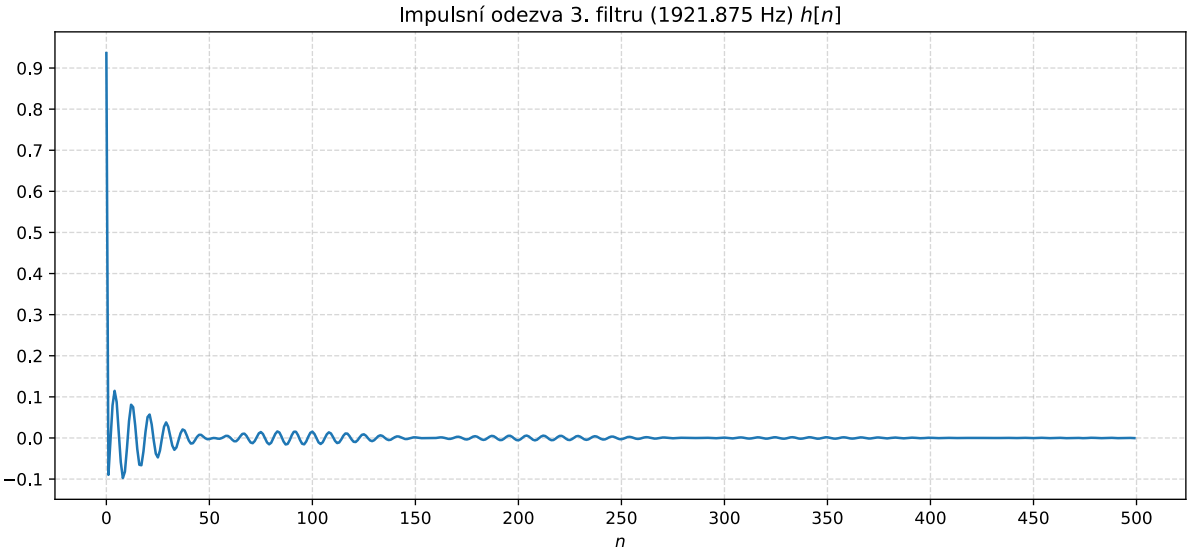
b: 0.93682784, -6.56594341, 21.00432923, -39.85603459, 48.96517566, -39.85603459, 21.00432923, -6.56594341, 0.93682784



3. Filtr - 1921.875 Hz

a: 1, -5.73195783, 16.19024618, -28.40661205, 33.67420574, -27.48693596, 15.15887957, -5.19305834, 0.87665471

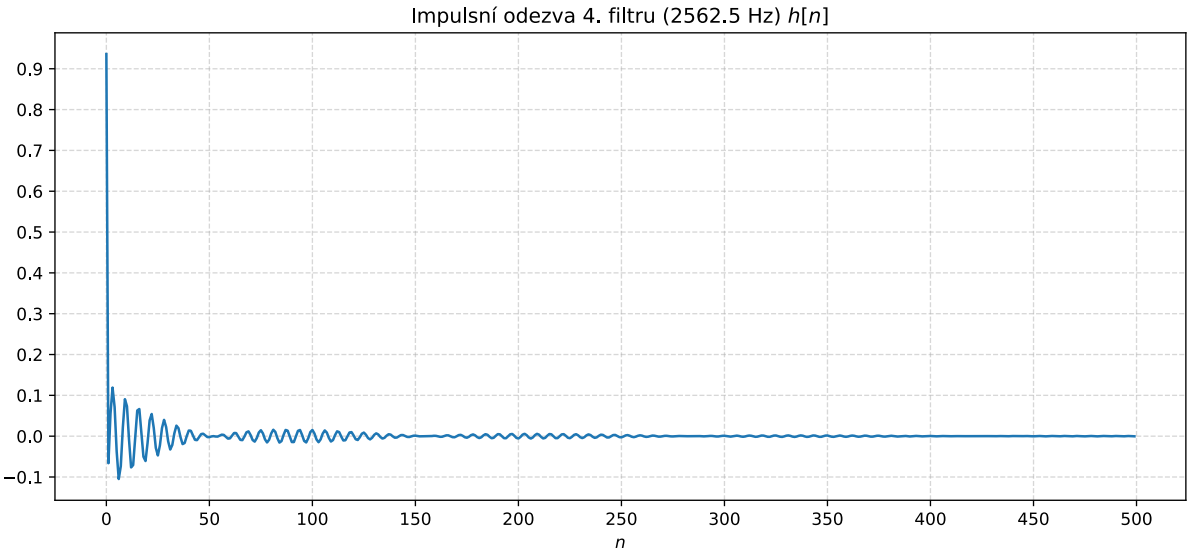
b: 0.93629841, -5.456597, 15.67025827, -27.9526851, 33.68687283, -27.9526851, 15.67025827, -5.456597, 0.93629841



4. Filtr - 2562.500 Hz

a: 1, -4.20929553, 10.51261385, -16.87504949, 19.69372153, -16.32592884, 9.83957168, -3.81158701, 0.87605264

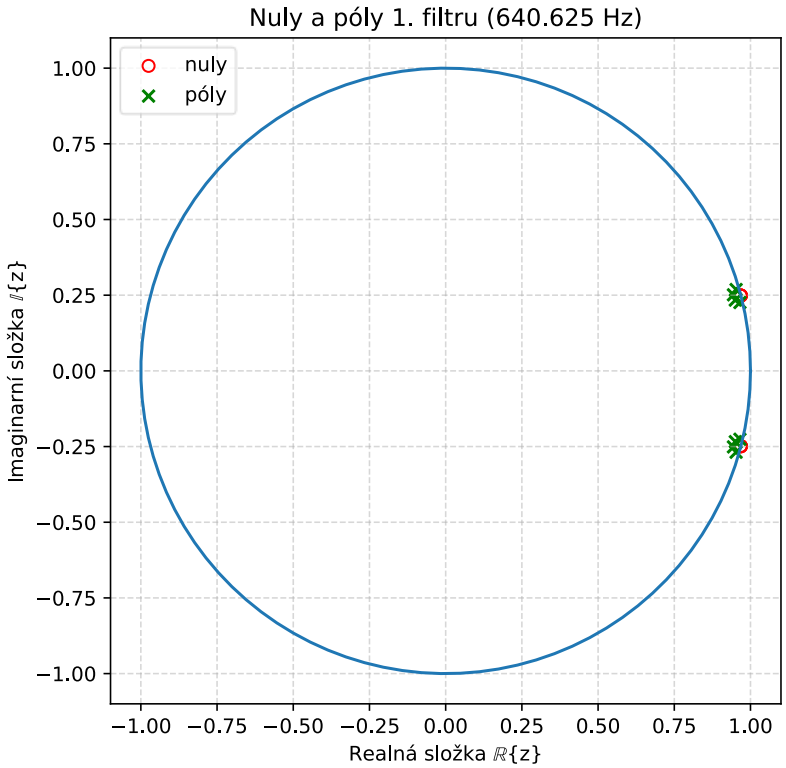
b: 0.93597684, -4.00605612, 10.1737482, -16.60487432, 19.70250962, -16.60487432, 0.1737482, -4.00605612, 0.93597684



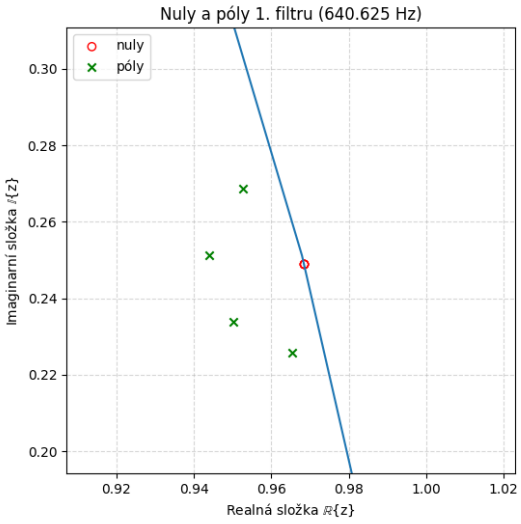
8 NULOVÉ BODY A PÓLY

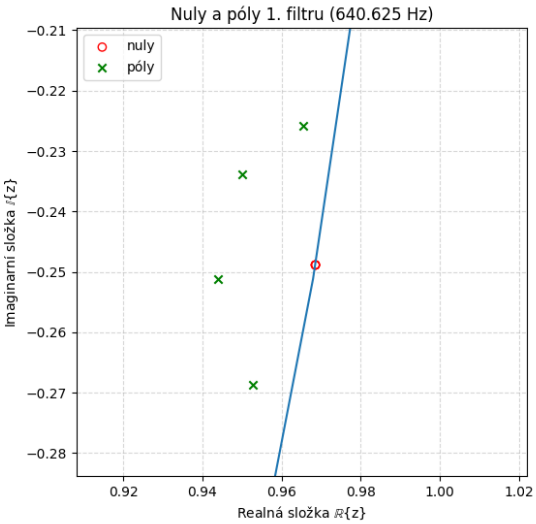
Nuly a póly byly zjištěny z SOS reprezentace filtru pomocí funkce sos2zpk.

1. Filtr - 640.625 Hz

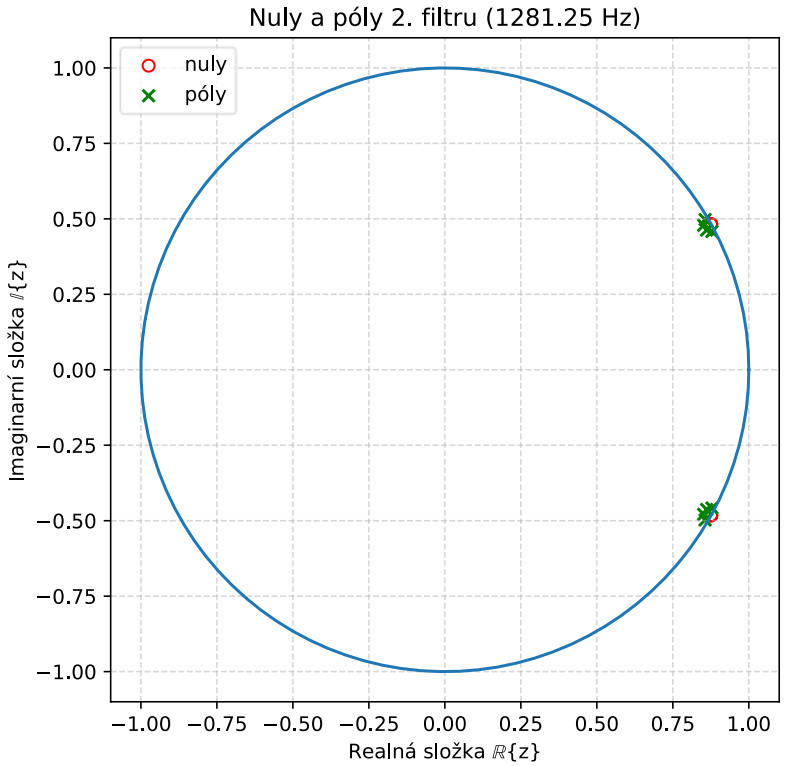


Detailní pohled na nuly a póly:

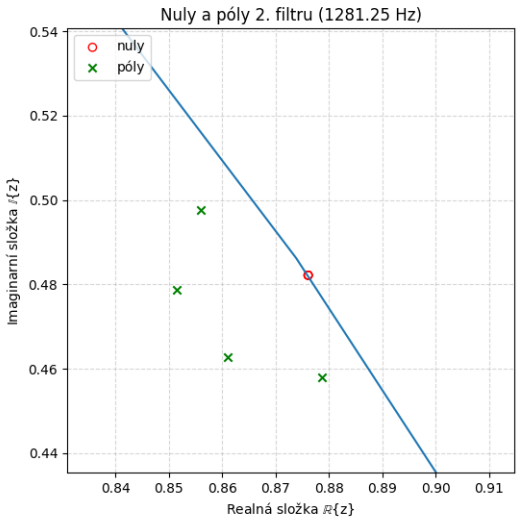


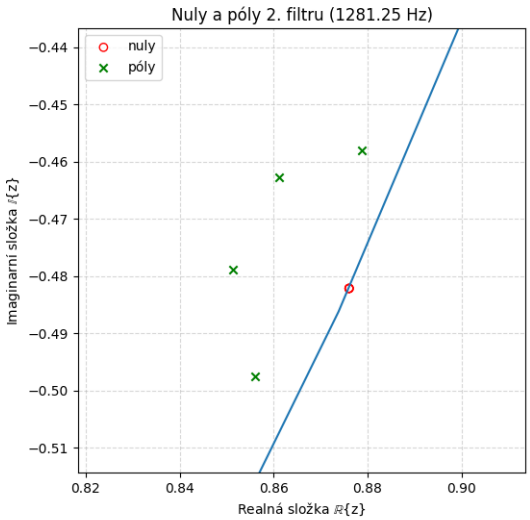


2. Filtr - 1281.250 Hz

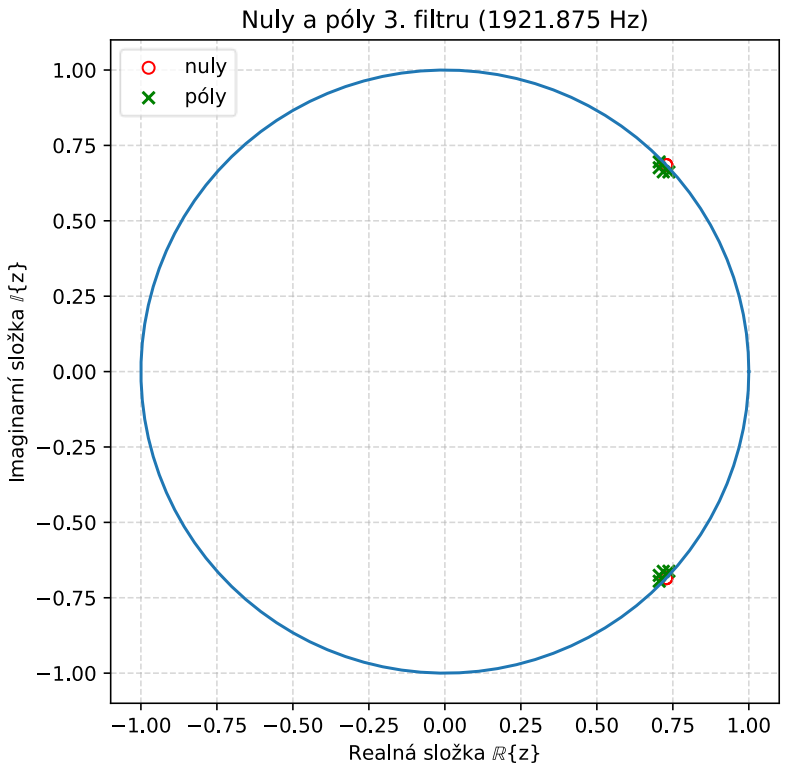


Detailní pohled na nuly a póly:

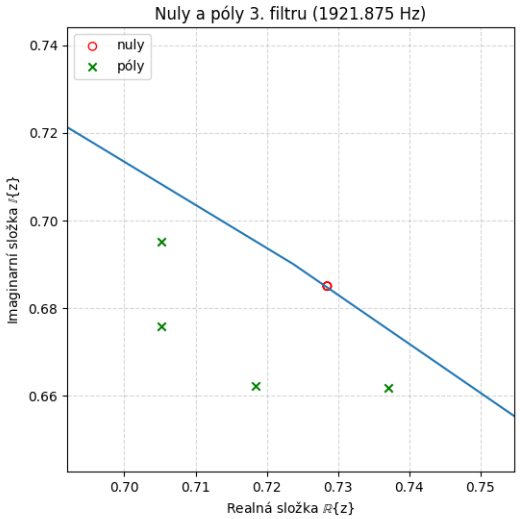


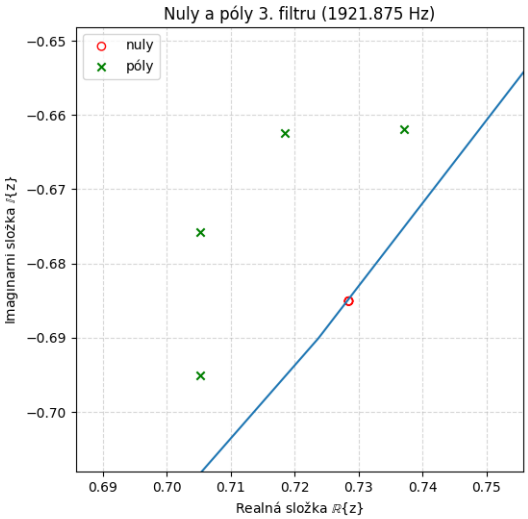


3. Filtr - 1921.875 Hz

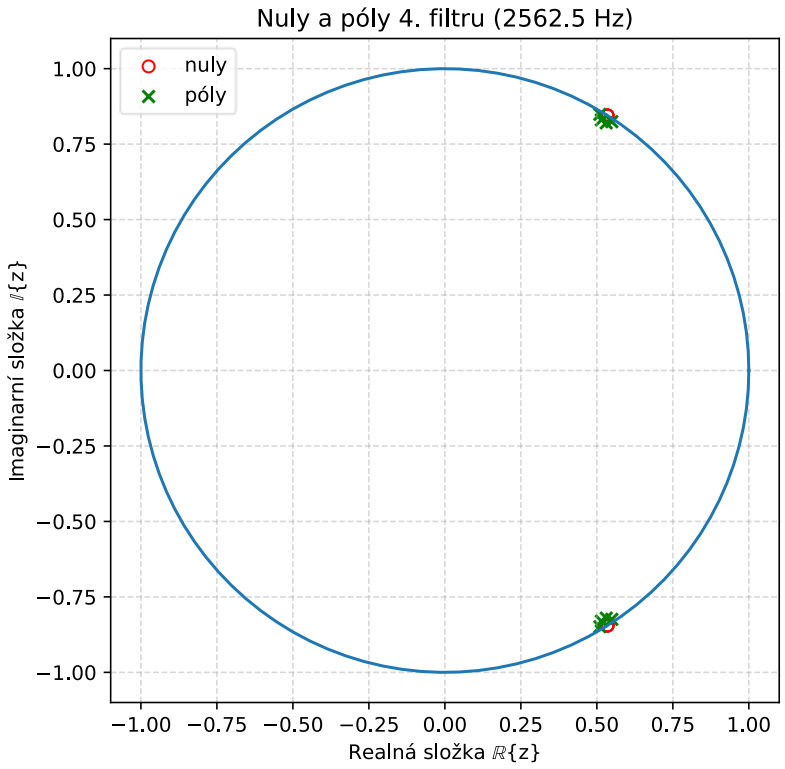


Detailní pohled na nuly a póly:

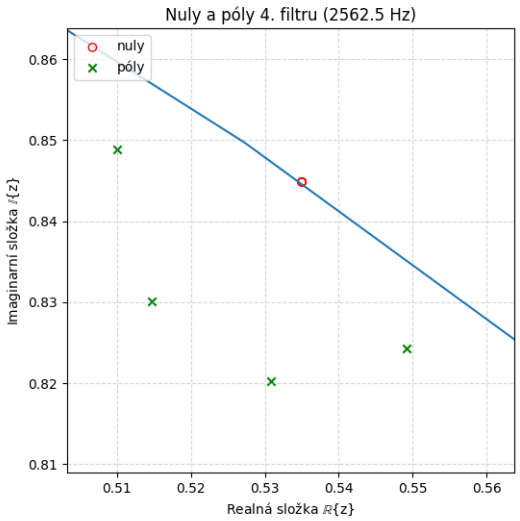


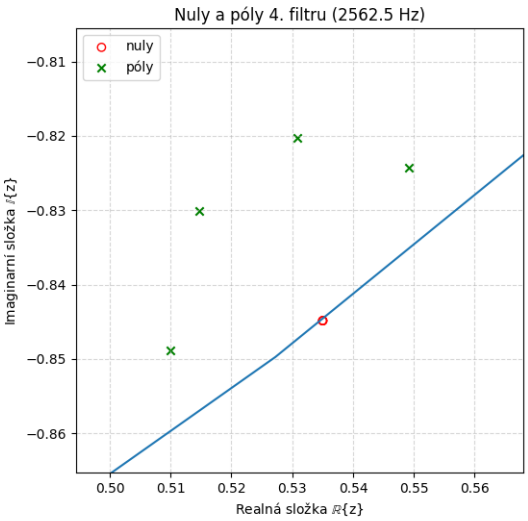


4. Filtr - 2562.500 Hz



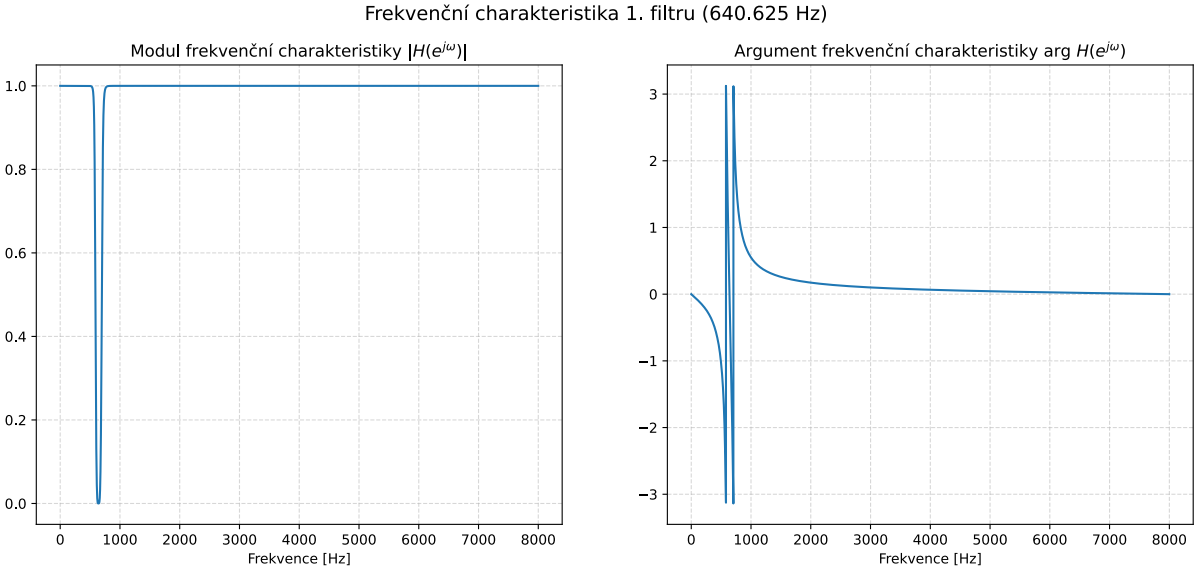
Detailní pohled na nuly a póly:

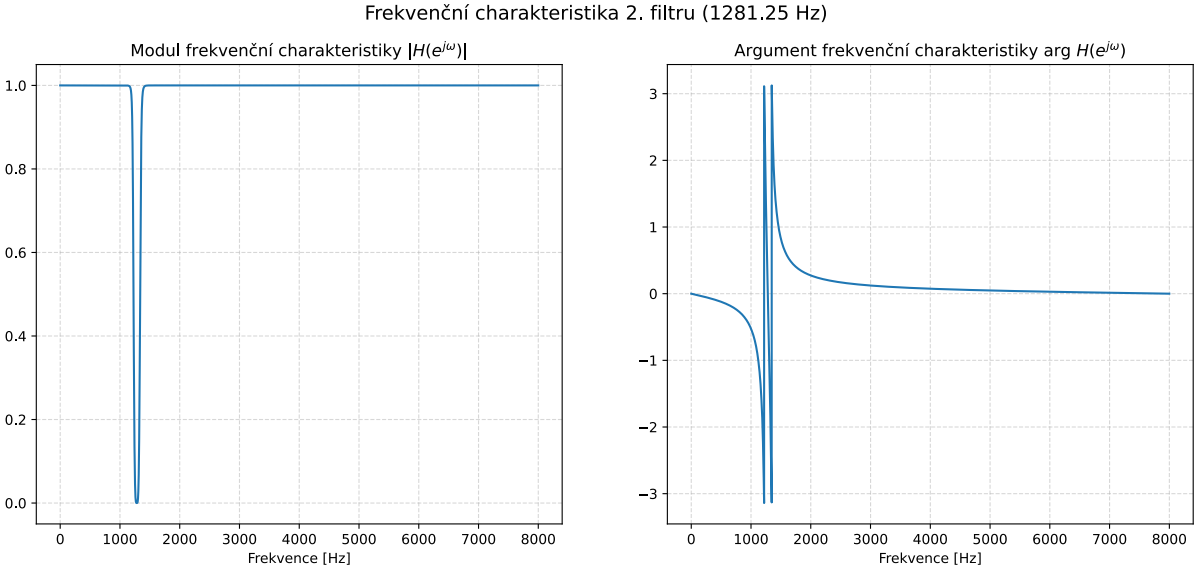


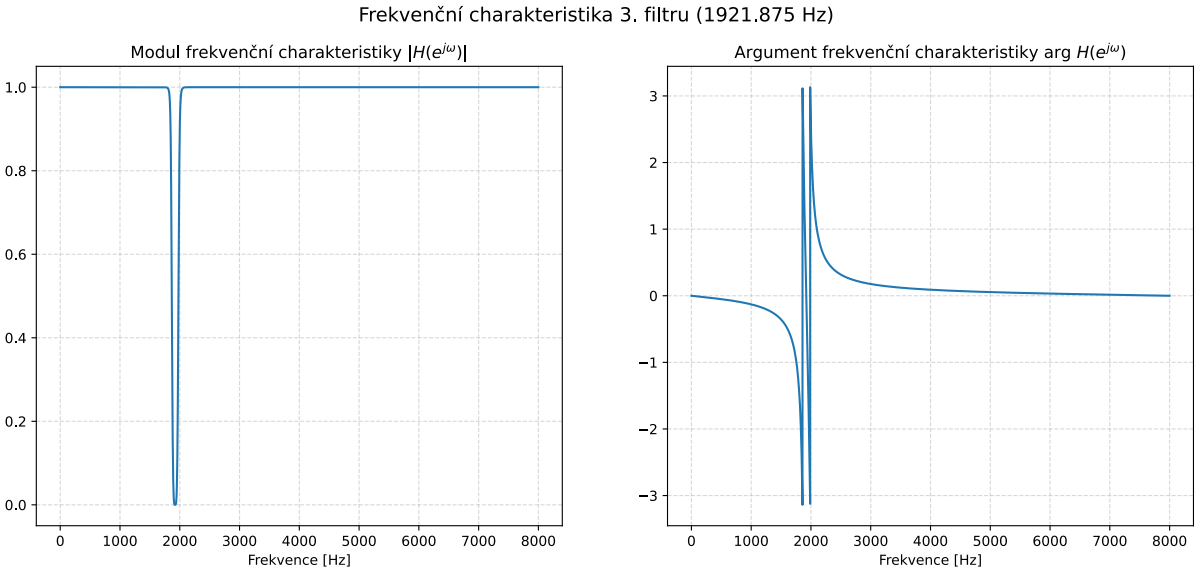


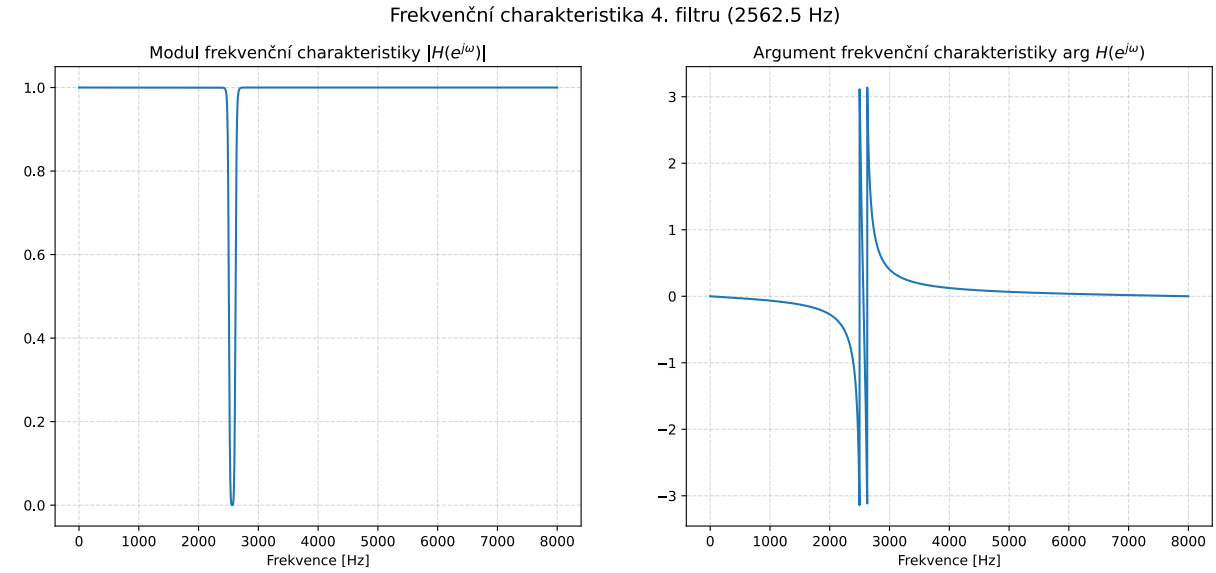
9 FREKVENČNÍ CHARAKTERISTIKA

Frekvenční charakteristiky filtrů byly zjištěny z jejich SOS reprezentace pomocí funkce `sosfreqz`.









Moduly frekvenčních charakteristik filtrů ukazují, že je rušivý signál potlačován na správných frekvencích. Potlačované frekvence jsou ty, kde se modul frekvenční charakteristiky přibližuje 0 – frekvence v těchto místech jsou opravdu shodné s frekvencemi rušivých signálů.

10 Filtrace

Filtrace signálu byla provedena pomocí funkce `sosfilt` a výsledek uložen podle instrukcí v zadání. Ve výsledné nahrávce již není slyšet žádné pískání a zvuk je čistý. Osobně neslyším ani žádné zkreslení hlasové aktivity, které filtrace způsobila – zkreslení je pravděpodobně velmi malé a moje uši nejsou dostatečně citlivé. Vzhledem k těmto skutečnostem považuji výsledek projektu za úspěšný.

ZDROJE

[1] `Numpy` dokumentace - https://numpy.org/doc/stable/

[2] `Scipy` dokumentace - https://docs.scipy.org/doc/scipy/reference/

[3] `Matplotlib` dokumentace - https://matplotlib.org/3.5.1/index.html#

[4] Maticové DFT - https://www.slideshare.net/sarang1689/computing-dft-using-matrix-method