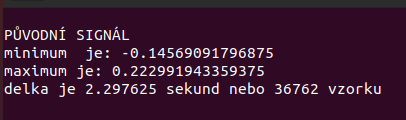
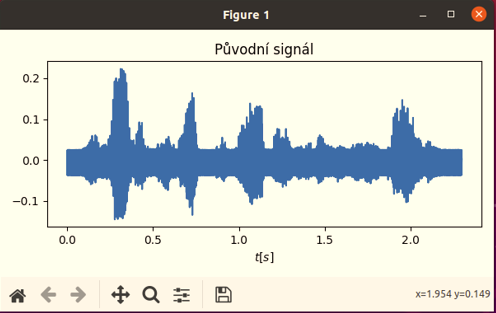
**Protokol k projektu ISS**

1. **Základy**

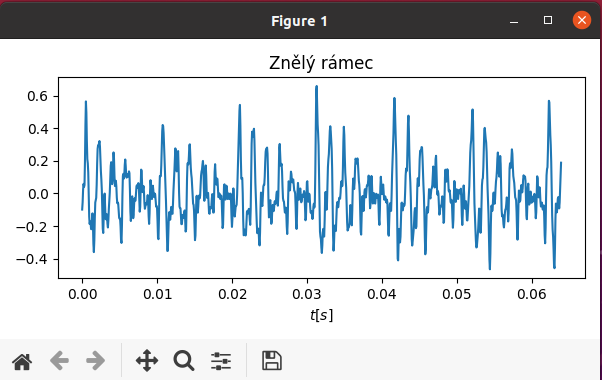
Originální signál jsem načetl pomocí knihovny *soundfile* a její funkce *soundfile.read.* Požadované hodnoty jsem poté zjistil pomocí návratových hodnot této funkce a vypsal je. Nakonec jsem pomocí knihovny *matplotlib* signál zobrazil.

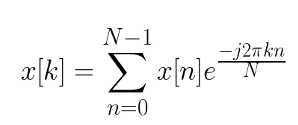




1. **Předzpracování rámce**

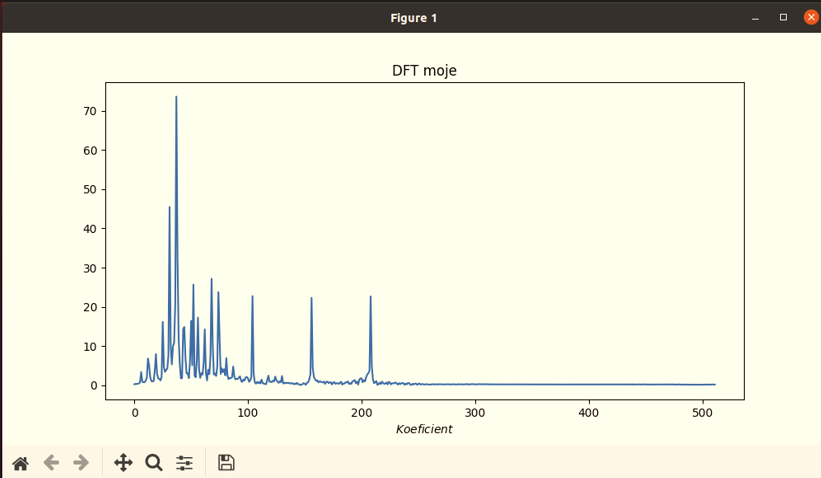
Střední hodnotu jsem zjistil jednoduše pomocí funkce *np.mean,* rozdělení signálu probíhá přes jednoduché vnořené cykly *for*. Pomocí dalšího cyklu *for* jsem si postupně vykresloval všechny rámce, z nichž jsem vybral ten, který se mi zdál jako nejvíce znělý.

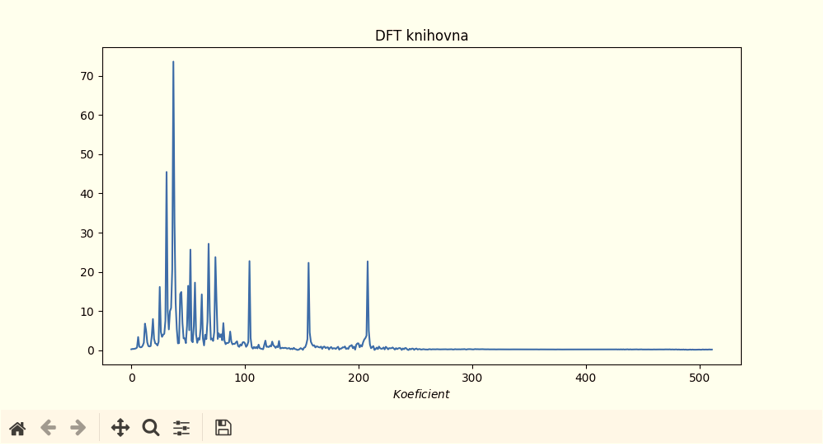


1. **DFT**

DFT jsem vytvářel podle vzorce

Při implementaci jsem využíval maticových operací z knihovny numpy. Z výsledného DFT jsem smazal druhou polovinu a graf následně zobrazil a porovnal s DFT vytvořenou pomocí *np.fft.fft*





1. **Spektrogram**

Na spektrogram jsem využil funkci *spectogram*, kam jsem přímo zadal požadovanou délku okna a překrytí. Následně jsem jej zobrazil opět pomocí knihovny *matplotlib*.

Obsah obrázku text, snímek obrazovky, elektronika, displej

Popis byl vytvořen automaticky

1. **Určení rušivých frekvencí**

Rušivé frekvence jsem určil jednoduše pomocí přiblížení spektogramu.

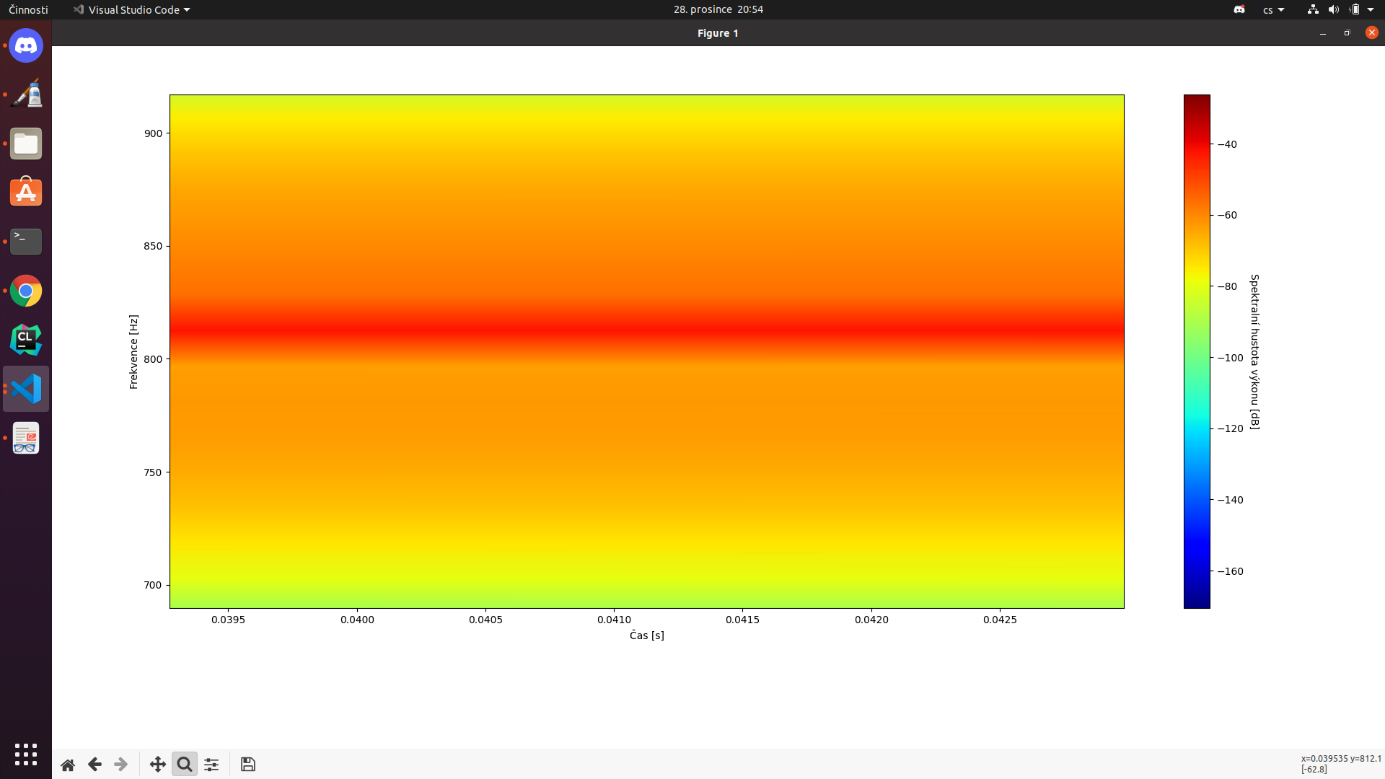
f1 = 812 Hz

f2 = 1624 Hz

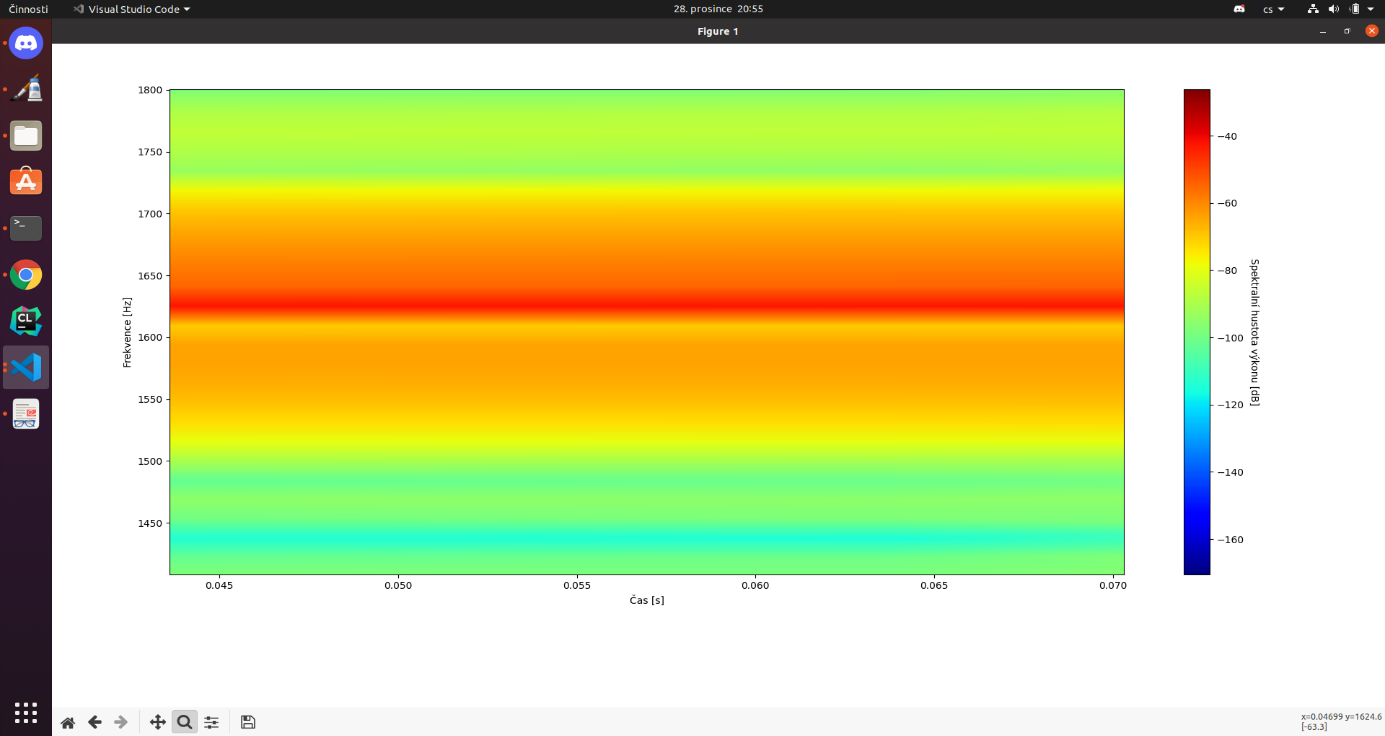
f3 = 2436 Hz

f4 = 3248 Hz

Z vyčtených hodnot vyplývá, že jsou cosinusovky harmonicky vztažené.



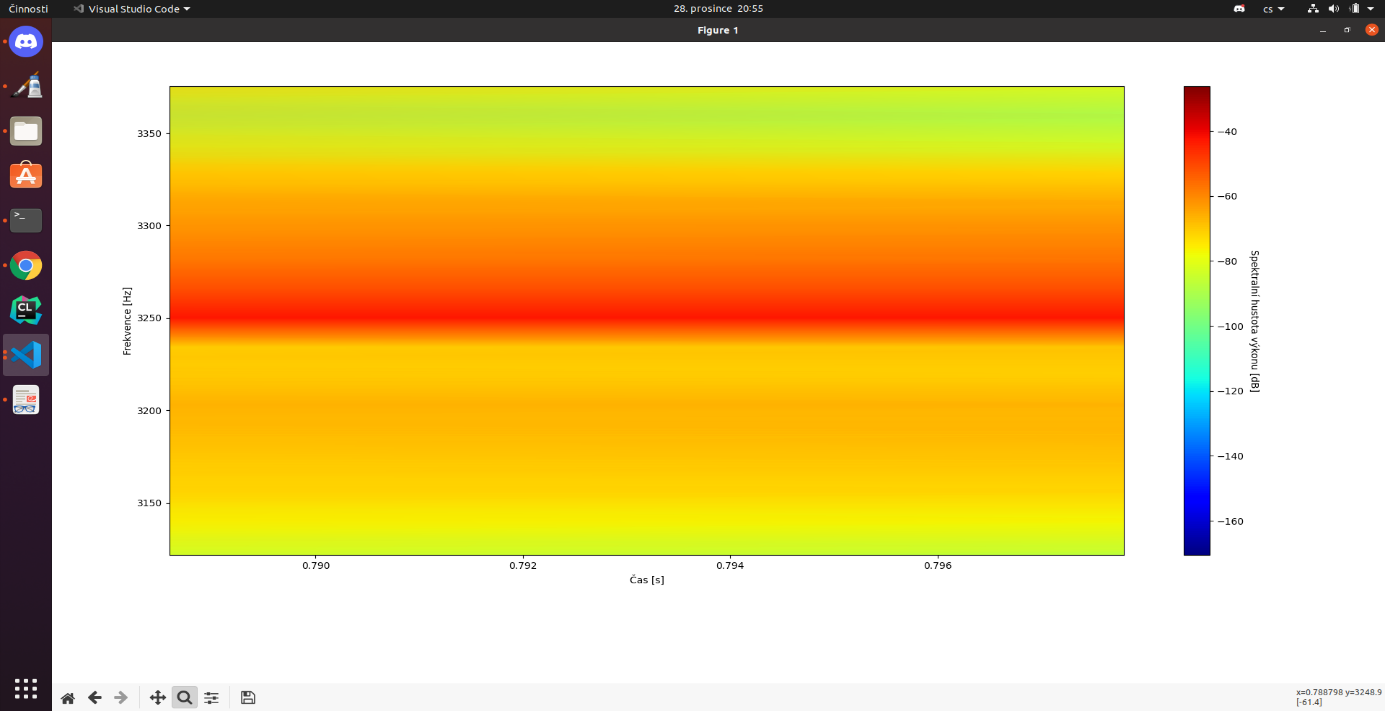
Frekvence – 812 Hz



Frekvence – 1624 Hz



Frekvence – 2436 Hz



Frekvence – 3248 Hz

oj

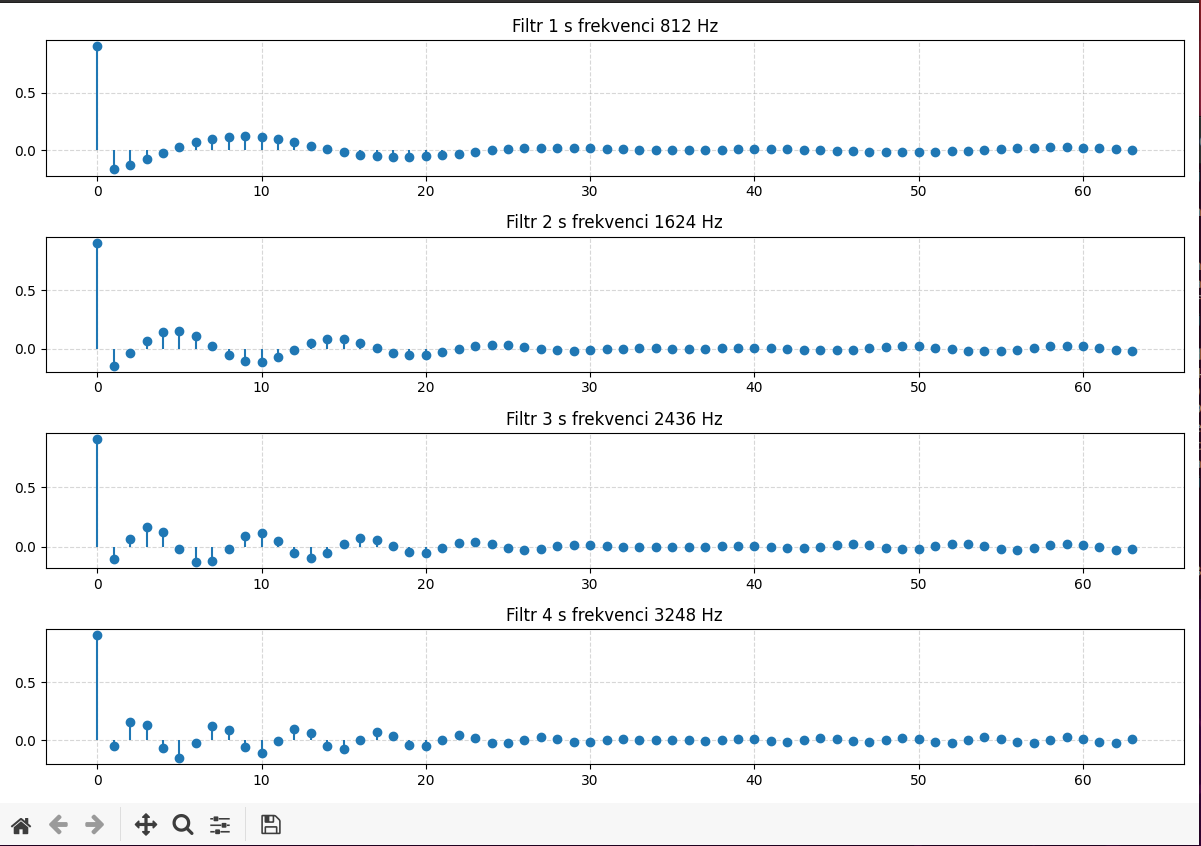
1. **Generování signálu**

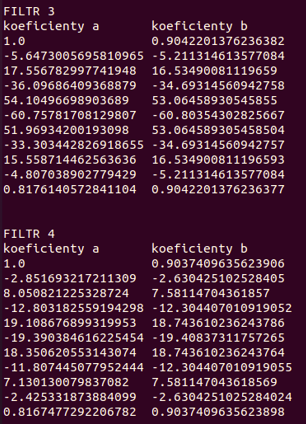
Nejprve jsem si vytvořil pole, které jsem pomocí cyklu zaplnil vzorky. Následně jsem vytvořil 4 cosinusovky pomocí *numpy* funkce *np.cos*, kde jsem jako vstupní parametry, vyjma dvojky a π, využil již zmiňované pole se vzorky a vyčtenou frekvenci. Poté jsem všechny cosinusovky sečetl. Spektrogram jsem zobrazil stejně jako v úkolu 4 a pro zapsání signálu byla využita funkce *soundfile.write.*

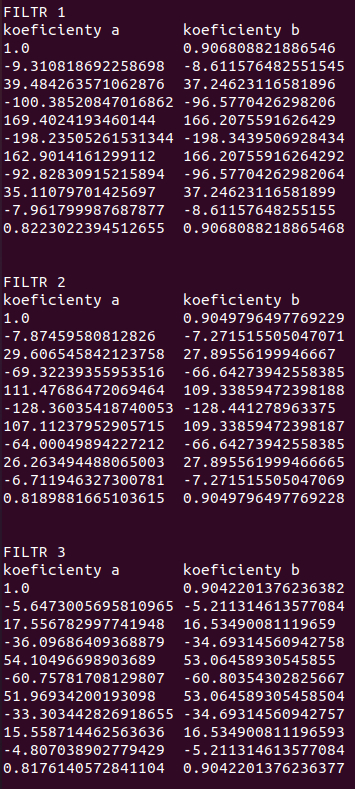
Obsah obrázku text, snímek obrazovky, monitor

Popis byl vytvořen automaticky

1. **Čistící filtr**

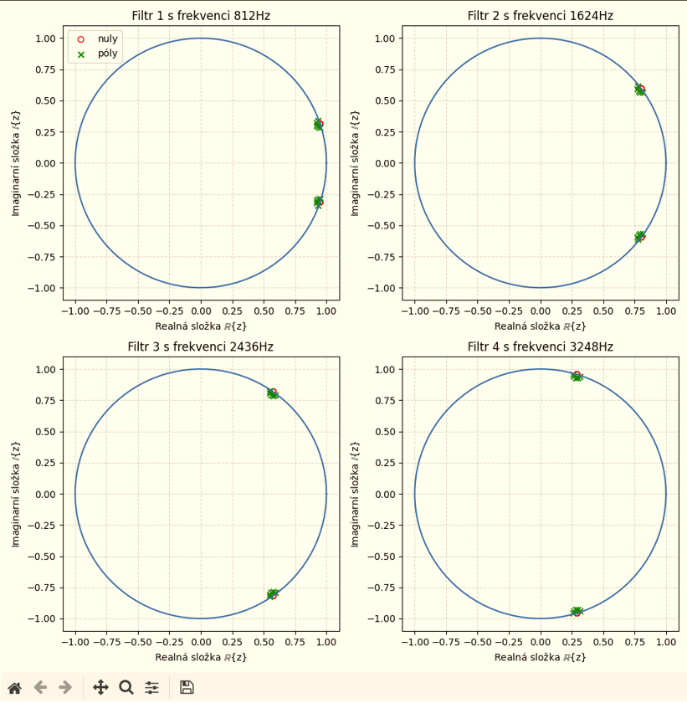
Vybral jsem třetí alternativu ze zadání, tedy návrh 4 pásmových propustí. Na toto jsem využil knihovnu *scipy*, konkrétně funkce *signal.buttord* a *signal.butter.* Při tvorbě jsem se řídil jejich dokumentací. Parametry jako závěrné pásmo, šíři přechodů do propustného pásma, zvlnění a potlačení šumu jsem nastavil dle hodnot v zadání. Impulzní odezvu jsem zjistil pomocí jednotkového impulzu a poté ji zobrazil. Koeficienty se mi vrátily po použití funkce *butter* a vzhledem k tomu,že mám 4 samostatné filtry, ukládal jsem si je postupně do pole, která jsem vypsal.





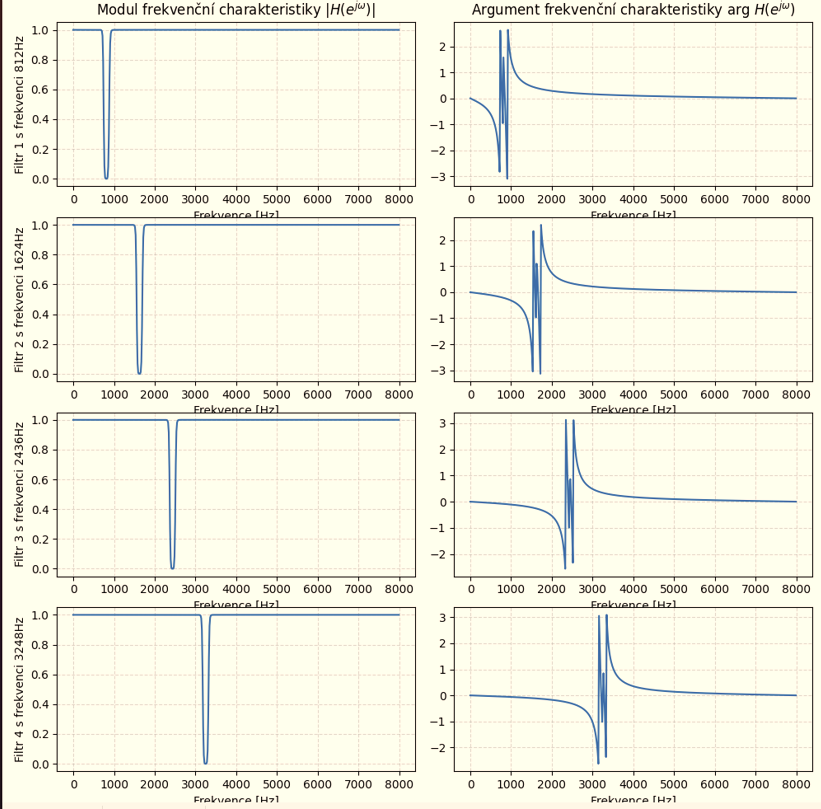
1. **Nulové body a póly**

Nuly a póly jsem získal opět funkcí *butter*, akorát s tím rozdílem, že jsem změnil output z *ba* na *zpk*, což mi namísto koeficientů vrácí nuly a póly. Graf jsem vytvářel tak, že jsem vytvořil kružnici, do které jsem promítl nalezené hodnoty pomocí funkce *scatter*.



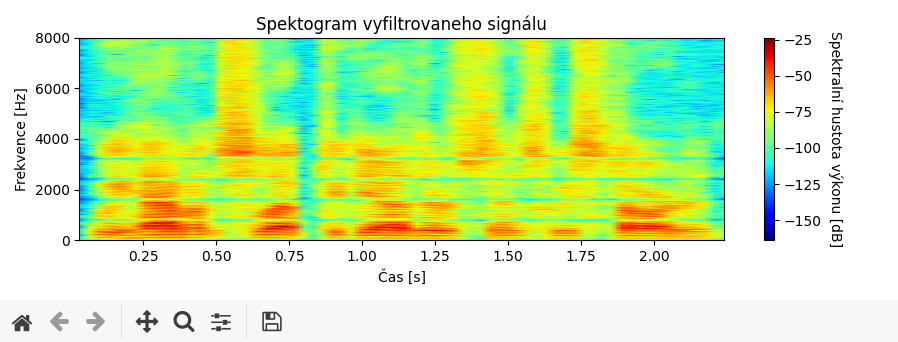
1. **Frekvenční charakteristika**

Frekvenční charakteristiky a jejich argumenty jsem získal pomocí funkce *freqz*, kam jsem nahrál získané koeficienty *b,a.* Poté jsem všechny frekvenční charakteristiky zobrazil. O jaký filtr se jedná je zobrazeno na ose Y před grafem modulů frekvenčních charakteristik.



1. **Filtrace**

Filtraci jsem provedl pomocí postupného aplikování všech filtrů na původní signál. Signál ještě nebyl v rozsahu od -1 do +1, upravil jsem to tedy opět vydělením maximem absolutní hodnoty. Správné vyfiltrování jsem kromě poslechu ještě ověřil spektrogramem. Po spuštění nahrávky se zde stále nacházelo malé lupnutí na začátku, které potvrdil i spektogram, vynuloval jsem tedy prvních 0.05 sekund nahrávky, kde se pípnutí dle spektogramu nacházelo.



1. **Zdroje**

Převážná část informací pochází z příkladů od Katky Žmolíkové a dokumentací knihoven scipy, numpy a matplotlib