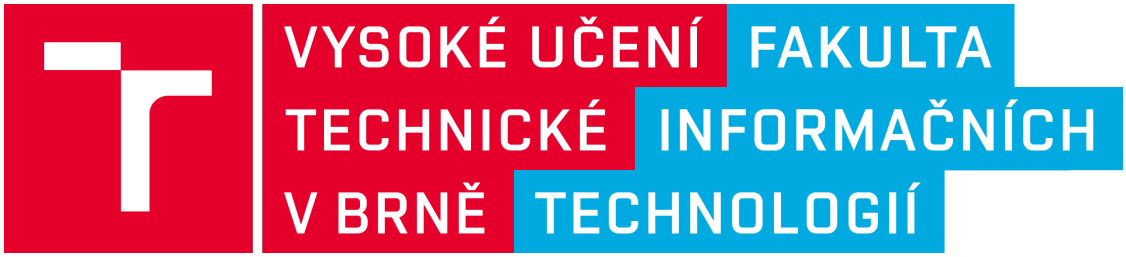


Protokol

ISS/VSG Projekt 2021/22

David Chocholatý (xchoch09)

Brno, 5. ledna 2022

Obsah

1 Řešení

Projekt je řešen v jazyce *Python* za použití programu *Jupyter Notebook*. Veškěré zdrojové kódy se nacházejí v souboru ***xchoch09.ipynb***.

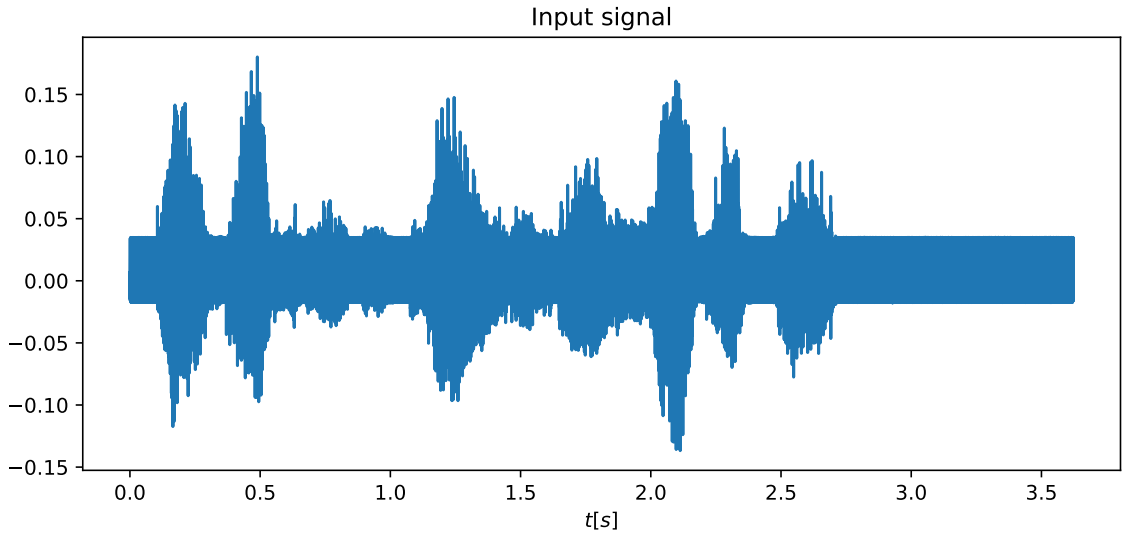
1.1 Úloha 1

Vstupní signál je načten pomocí funkce *read()* implementované v modulu *soundfile*.

Informace o vstupním signálu:

- Délka ve vzorcích: **57959 vzorků**
- Délka v sekundách: **3.6224375 s**
- Minimální hodnota: **-0.136749267578125**
- Maximální hodnota: **0.180206298828125**

Vstupní signál:



1.2 Úloha 2

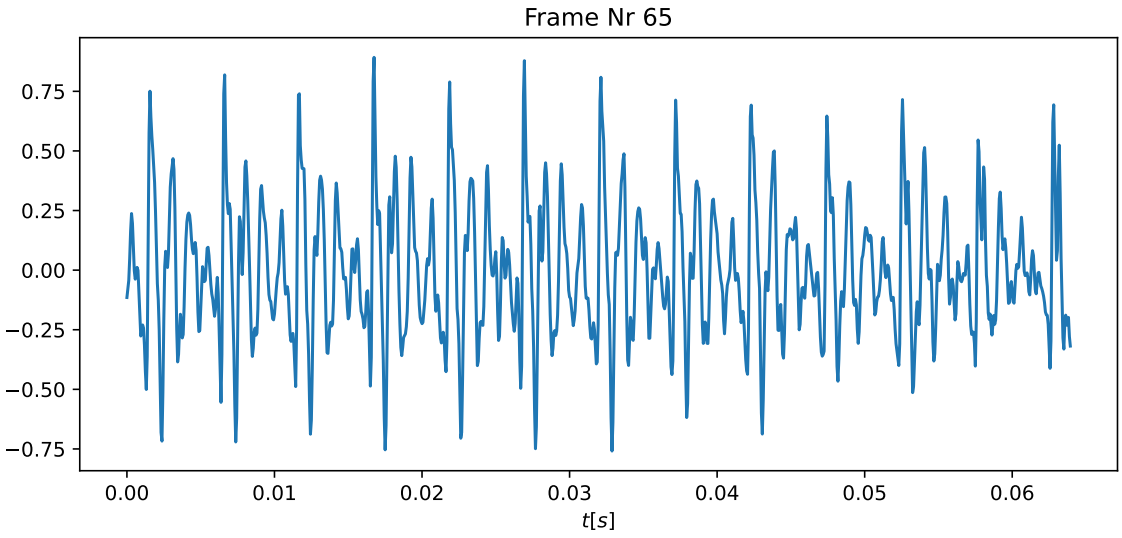
Úsek zdrojového kódu pro ustřednění načteného signálu (odečtení od signálu jeho střední hodnoty) a normalizace do dynamického rozsahu -1 až 1 (dělení signálu maximem absolutní hodnoty):

```
# Center input signal
data = data - np.mean(data)

# Normalize signal to range -1, 1
data = data / max(np.max(data), abs(np.min(data)))
```

Dále je uveden zdrojový kód pro rozdělení signálů na rámce a jejich uložení do sloupců matice:

```
# Split signal into frames of size 1024 samples
# with 512 samples overlap
split_size = 1024
split_step = 512

data_seg=[data[i:i+split_size] for i in range(0,len(data),split_step)]

# Save frames as matrix columns
matrix = list(map(list, zip_longest(*data_seg, fillvalue=None)))
```

Vybraný rámec (pořadí 65):

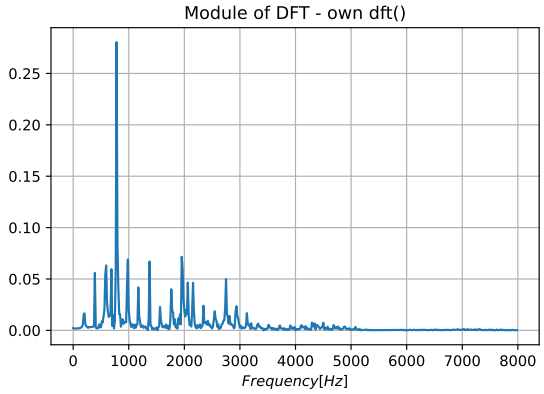


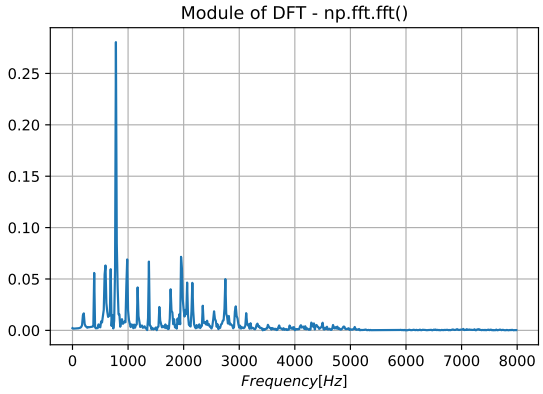
1.3 Úloha 3

Vlastní implementace diskrétní Fourierovy transformace se nachází ve funkci *dft()*. Transformace se vytváří násesobením DFT matice (W) a vektoru (x). $N = 1024$ *vzorků* je získáno přímo ze vstupního vektoru pomocí příkazu $N = x.shape[0]$.

```
def dft(x):
    # Calculation of N
    x = np.asarray(x, dtype=float)
    N = x.shape[0]

    # Create DFT matrix W
    n = np.arange(N)
    k = n.reshape((N, 1))

    W = np.exp(-2j * np.pi * k * n / N)

    # Dot product of matrix W and vector x
    return np.dot(W, x)
```

Grafické porovnání výsledků implementace funkce *dft()* a funkce z modulu numpy *np.fft.fft()*:



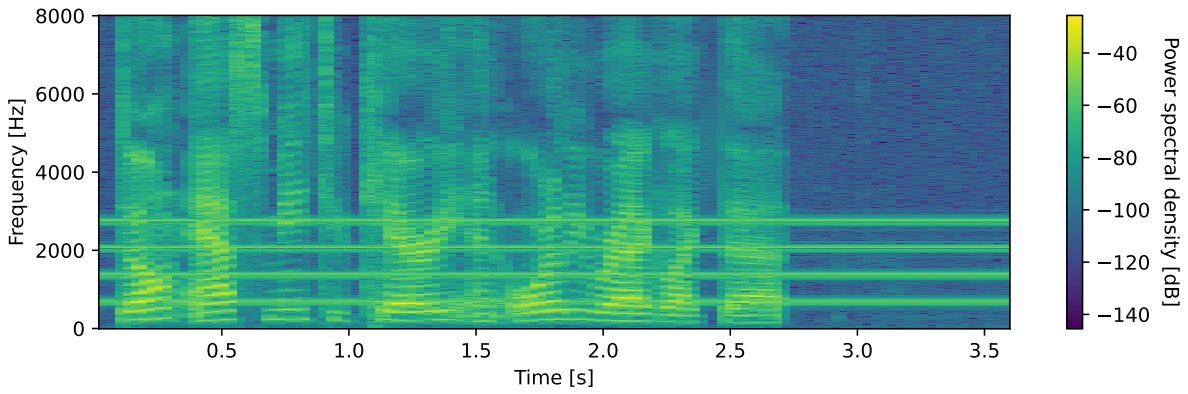


Výsledky byly také porovnány funkcí np.allclose() s úspěšným výsledkem *True*.

1.4 Úloha 4

Spektogram je vytvořen pomocí funkce *spectogram()* z modulu *scipy.signal*. Dále je uveden úsek zdrojového kódu pro vytvoření spektogramu a úpravy pomocí $log_{10}|X[k]|^2$. Z důvodu občasného výskytu nul ve spektogramu a použití logaritmu je přičtena hodnota 1e-20 viz Jupyter Notebook Katky Žmolíkové[1].

```
f, t, sgr = spectrogram(data, fs, nperseg=1024, noverlap=512)

sgr_log = 10 * np.log10(sgr+1e-20)
```

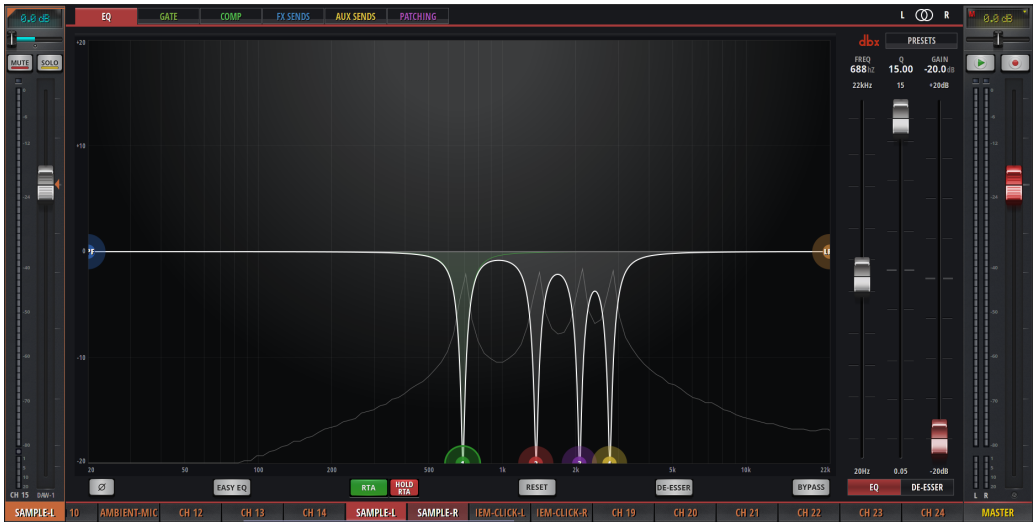


1.5 Úloha 5

Rušivé frekvence byly určeny pomocí dvou metod:

1. Ručně ze spektogramu
2. Použití ekvalizéru s funkcí ***RTA*** (*Real Time Analyzer*) a ***hold RTA***

   Ekvalizér (mixážní pult Soundcraft)



[1] https://nbviewer.org/github/zmolikova/ISS_project_study_phase/blob/master/Zvuk_spektra_filtrace.ipynb

Frekvence rušivých komponent:

$f_1 = 688$ Hz
$f_2 = 1376$ Hz
$f_3 = 2064$ Hz
$f_4 = 2752$ Hz

Rušivé cosinusovky jsou na určených frekvecích a **jsou harmonicky vztažené**:

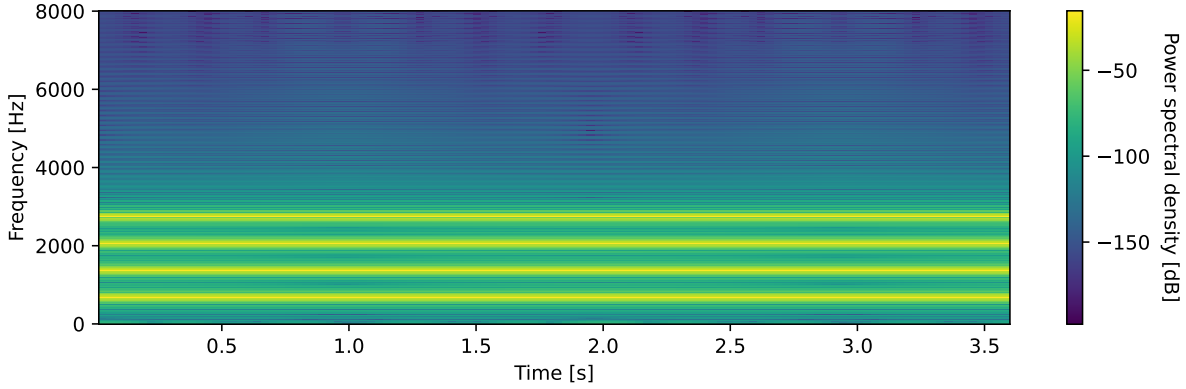
$f_1 = 688$ Hz
$f_2 =$ **2*688** $= 1376$ Hz
$f_3 =$ **3*688** $= 2064$ Hz
$f_4 =$ **4*688** $= 2752$ Hz

1.6 Úloha 6

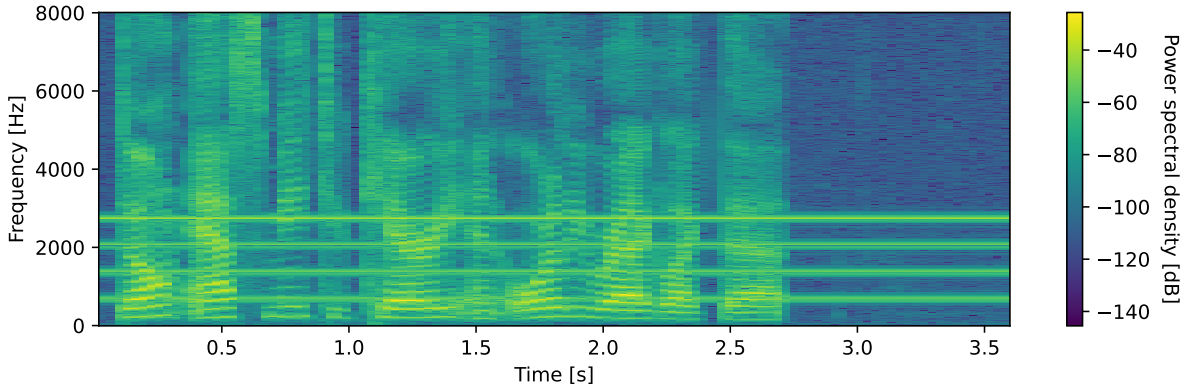
Vygenerovaný signál je vytvořen součtem 4 kosinusovek s využitím funkce *cos()* z modulu *numpy*. Ze zmíněného modulu je využita ještě funkce *linspace()*. Docílení stejné délky jako vstupní signál je docíleno pomocí délky (*length = data.size / fs*) a počtu vzorků *num=data.size*. Úsek zdrojového kódu pro generování signálu se směsí 4 kosinusovek:

```
# Generate signal of 4 cosines
f1 = 688
f2 = 1376 # 2*f1
f3 = 2064 # 3*f1
f4 = 2752 # 4*f1

omega_1 = 2 * np.pi * f1
omega_2 = 2 * np.pi * f2
omega_3 = 2 * np.pi * f3
omega_4 = 2 * np.pi * f4

length = data.size / fs

t = np.linspace(0, length, num=data.size)

signal  = np.cos(omega_1 * t)
signal += np.cos(omega_2 * t)
signal += np.cos(omega_3 * t)
signal += np.cos(omega_4 * t)
```

Spektogram vygenerovaného signálu:



Porovnání se spektogramem vstupního signálu:



Závěr:
Frekvence kosinusovek jsou správné.

1.7 Úloha 7

Číslicový filtr je vytvořen podle varianty návrhu 4 pásmových zádrží. Parametry pro všechny 4 filtry jsou následující:

- Šíře závěrného pásma: **30 Hz**
- Šíře přechodu do propustného pásma na každé straně: **50 Hz**
- Zvlnění v propustném pásmu: **3 dB**
- Potlačení v závěrném pásmu: **-40 dB**

Pro implementaci byly použity funkce *buttord()* a *butter()* implementované v modulu *scipy.signal*.

1.7.1 Koeficienty jednotlivých filtrů

1. Koeficienty filtru pro frekvenci f_1 (688 Hz)

   - b_0 = 0.95149545, b_1 = -7.33596752, b_2 = 25.01591737, b_3 = -49.26243775, b_4 = 61.2620112, b_5 = -49.26243775, b_6 = 25.01591737, b_7 = -7.33596752, b_8 = 0.95149545
   - a_0 = 1, a_1 = -7.61410876, a_2 = 25.64218391, a_3 = -49.87017166, a_4 = 61.25091973, a_5 = -48.64563526, a_6 = 24.39838961, a_7 = -7.06689486, a_8 = 0.9053436

2. Koeficienty filtru pro frekvenci f_2 (1376 Hz)

   - b_0 = 0.95073118, b_1 = -6.52235025, b_2 = 20.58253088, b_3 = -38.75271725, b_4 = 47.48987559, b_5 = -38.75271725, b_6 = 20.58253088, b_7 = -6.52235025, b_8 = 0.95073118
   - a_0 = 1, a_1 = -6.77370764, a_2 = 21.10604616, a_3 = -39.23797219, a_4 = 47.48030971, a_5 = -38.25913676, a_6 = 20.06615406, a_7 = -6.27931841, a_8 = 0.90388978

3. Koeficienty filtru pro frekvenci f_3 (2064 Hz)

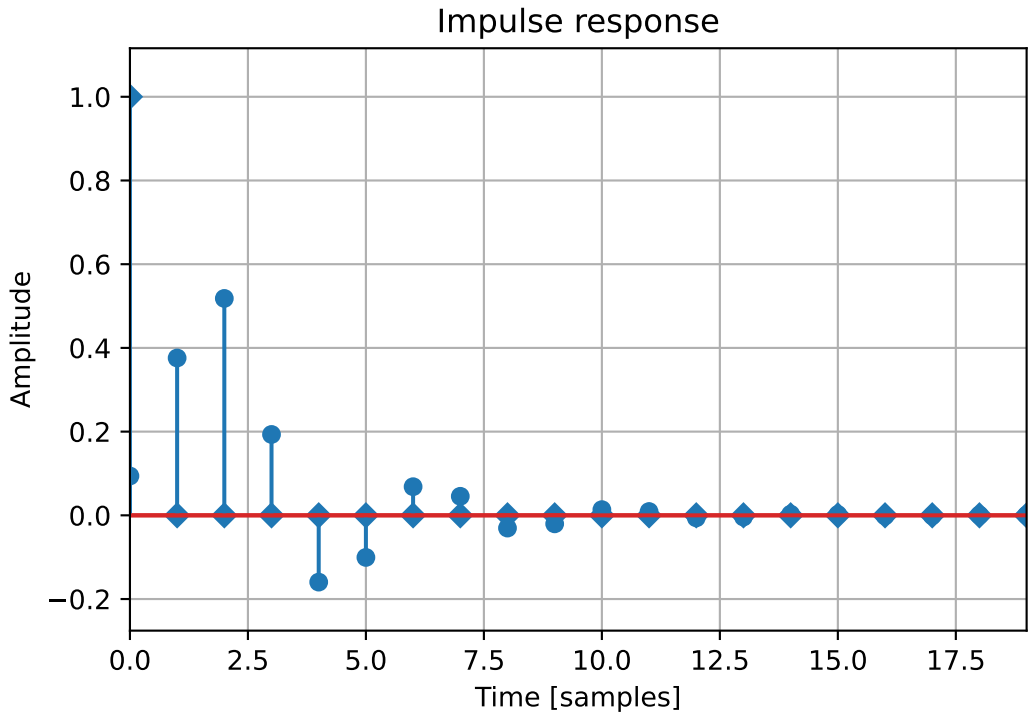
   - b_0 = 0.95042391, b_1 = -5.23971291, b_2 = 14.63420061, b_3 = -25.6724545, b_4 = 30.79710969, b_5 = -25.6724545, b_6 = 14.63420061, b_7 = -5.23971291, b_8 = 0.95042391
   - a_0 = 1, a_1 = -5.44295409, a_2 = 15.00857088, a_3 = -25.9953517, a_4 = 30.78998471, a_5 = -25.34278311, a_6 = 14.26449752, a_7 = -5.0432459, a_8 = 0.90330562

4. Koeficienty filtru pro frekvenci f_4 (2752 Hz)

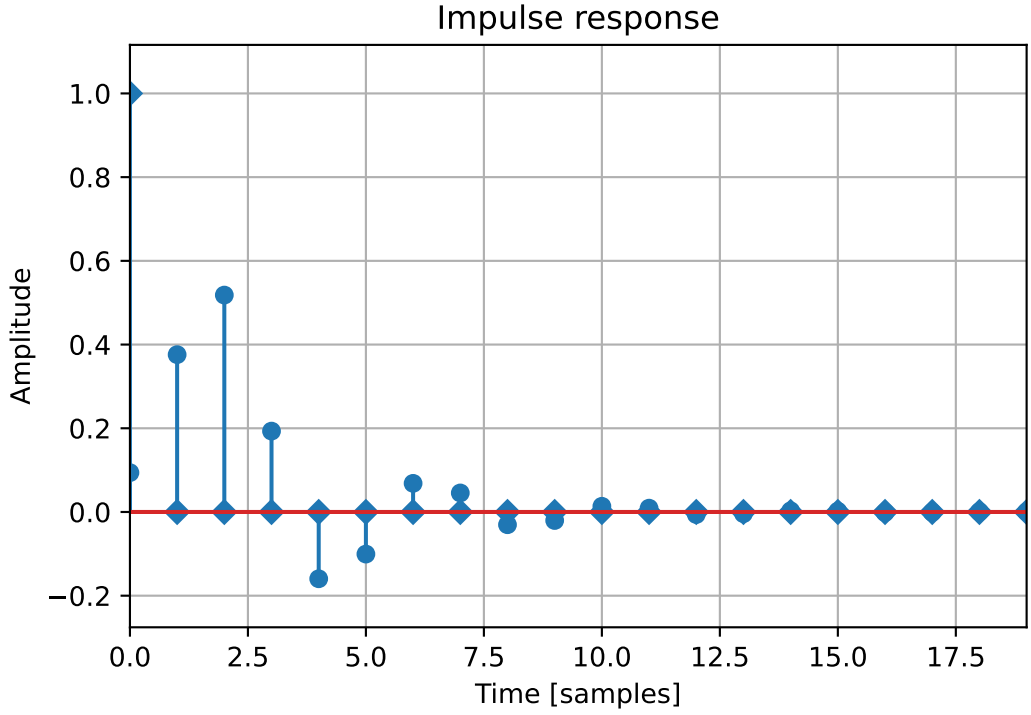
   - b_0 = 0.9502495, b_1 = -3.57837489, b_2 = 8.8541841, b_3 = -13.9066065, b_4 = 16.55430151, b_5 = -13.9066065, b_6 = 8.8541841, b_7 = -3.57837489, b_8 = 0.9502495
   - a_0 = 1, a_1 = -3.71768461, a_2 = 9.08117274, a_3 = -14.08164966, a_4 = 16.54963408, a_5 = -13.72690357, a_6 = 8.62938779, a_7 = -3.44372494, a_8 = 0.90297412

1.7.2 Impulzní odezvy jednotlivých filtrů

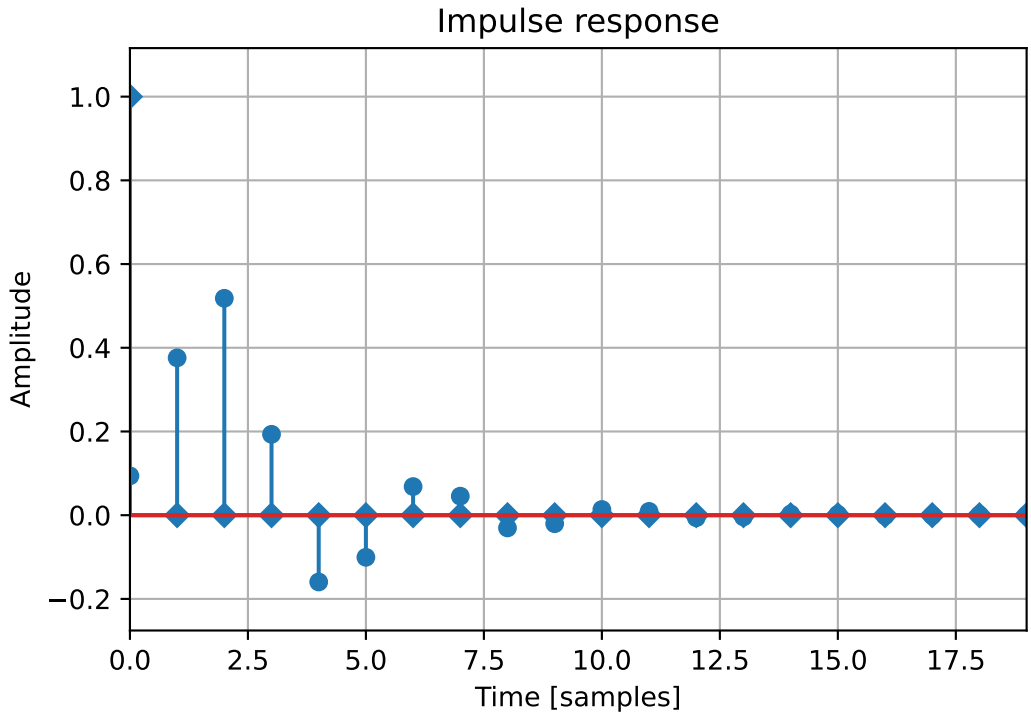
- Impulzní odezva filtru pro frekvenci f_1



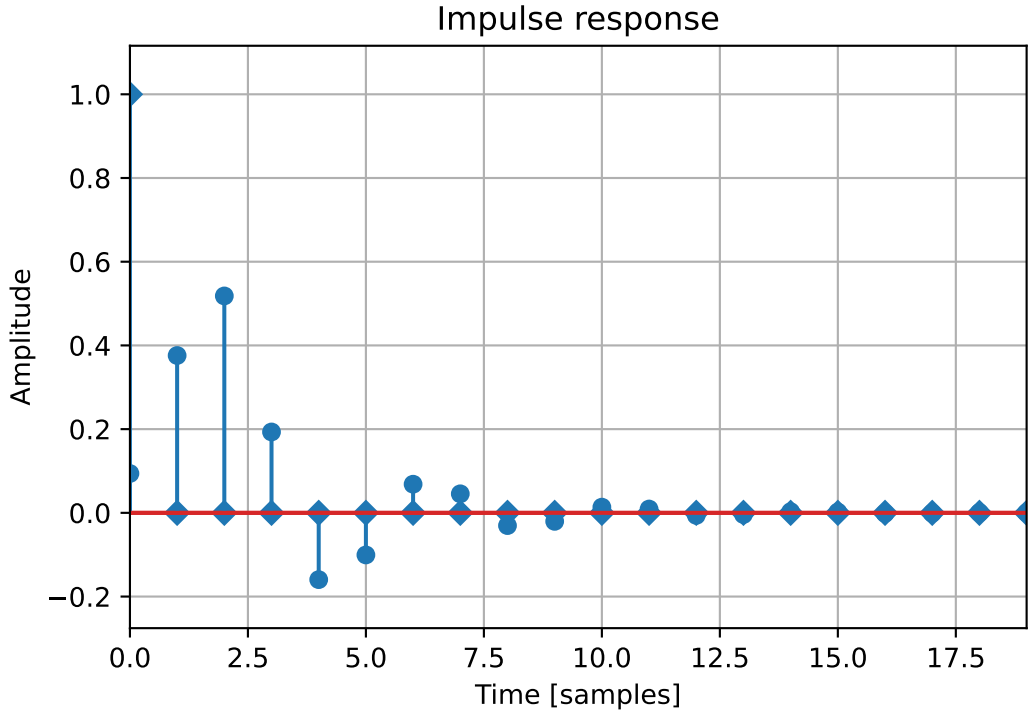
- Impulzní odezva filtru pro frekvenci f_2



- Impulzní odezva filtru pro frekvenci f_3



- Impulzní odezva filtru pro frekvenci f_4



1.8 Úloha 8

Funkce *zplane()* implementovaná v jazyce Python:

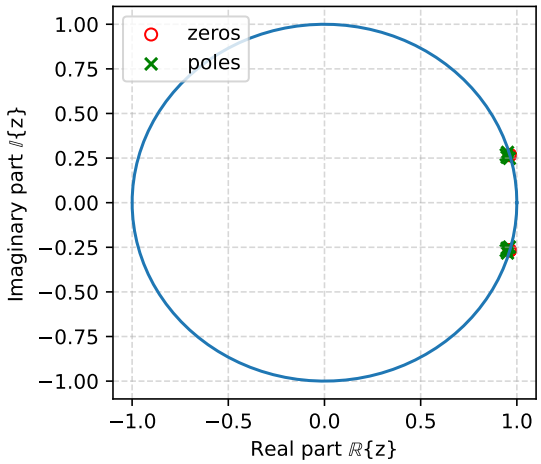
```
def zplane(b,a):
    # The coefficients are less than 1, normalize the coeficients
    if np.max(b) > 1:
        kn = np.max(b)
        b = b/float(kn)
    else:
        kn = 1

    if np.max(a) > 1:
        kd = np.max(a)
        a = a/float(kd)
    else:
        kd = 1

    # Get the poles and zeros
    p = np.roots(a)
    z = np.roots(b)
    k = kn/float(kd)

    # Plot
    plt.figure(figsize=(4,3.5))

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

    # Zeros and poles
    plt.scatter(np.real(z), np.imag(z), marker='o',
                facecolors='none', edgecolors='r',
                label='zeros')
    plt.scatter(np.real(p), np.imag(p), marker='x',
                color='g', label='poles')

    plt.gca().set_xlabel('Real part $\mathbb{R}\{$z$\}$')
    plt.gca().set_ylabel('Imaginary part $\mathbb{I}\{$z$\}$')

    plt.grid(alpha=0.5, linestyle='--')
    plt.legend(loc='upper left')

    plt.tight_layout()

    return z, p, k
```

1.8.1 Nulové body a póly jednotlivých filtrů

- Nulové body a póly filtru pro frekvenci f_1 (688 Hz)

  Nulové body: 0.96450313+0.26703613j, 0.96450313-0.26703613j, 0.96354344+0.26759669j, 0.96354344-0.26759669j, 0.96394017+0.266074j, 0.96394017-0.266074j, 0.96298045+0.2666394j, 0.96298045-0.2666394j

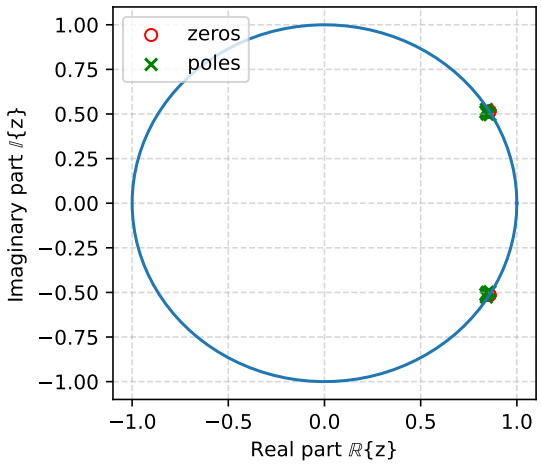
  Póly: 0.95137798+0.28198785j, 0.95137798-0.28198785j, 0.96158897+0.24860161j, 0.96158897-0.24860161j, 0.94469844+0.26851548j, 0.94469844-0.26851548j, 0.94938899+0.25496146j, 0.94938899-0.25496146j



- Nulové body a póly filtru pro frekvenci f_2 (1376 Hz)

  Nulové body: 0.85727494+0.51466142j, 0.85727494-0.51466142j, 0.85779464+0.51467976j, 0.85779464-0.51467976j, 0.85781308+0.51415995j, 0.85781308-0.51415995j, 0.85729318+0.51414163j, 0.85729318-0.51414163j

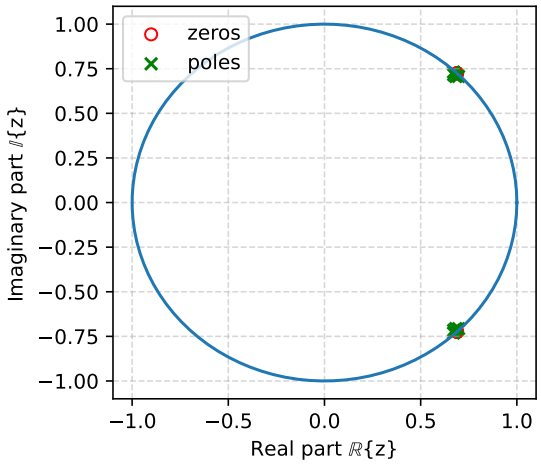
  Póly: 0.84166625+0.52581231j, 0.84166625-0.52581231j, 0.86028902+0.49563219j, 0.86028902-0.49563219j, 0.83852482+0.5112259j, 0.83852482-0.5112259j, 0.84637372+0.49897739j, 0.84637372-0.49897739j



- Nulové body a póly filtru pro frekvenci f_3 (2064 Hz)

  Nulové body: 0.68934918+0.72477544j, 0.68934918-0.72477544j, 0.68899221+0.72486j, 0.68899221-0.72486j, 0.68926456+0.72441848j, 0.68926456-0.72441848j, 0.68890766+0.72450309j, 0.68890766-0.72450309j

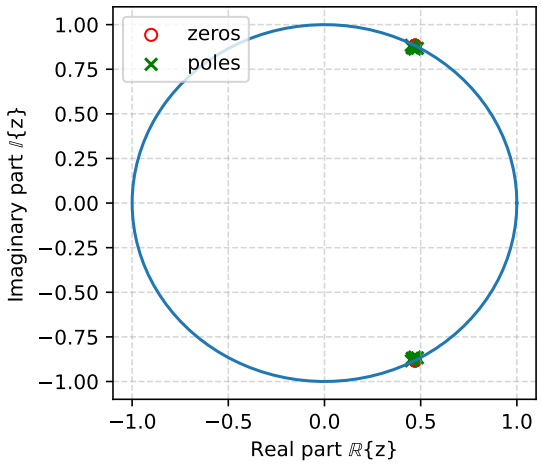
  Póly: 0.67079854+0.73143685j, 0.67079854-0.73143685j, 0.69683502+0.70703111j, 0.69683502-0.70703111j, 0.67153451+0.71657264j, 0.67153451-0.71657264j, 0.68230898+0.70666891j, 0.68230898-0.70666891j



- Nulové body a póly filtru pro frekvenci f_4 (2752 Hz)

  Nulové body: 0.47071782+0.88245182j, 0.47071782-0.88245182j, 0.47088175+0.88228258j, 0.47088175-0.88228258j, 0.47054859+0.88228787j, 0.47054859-0.88228787j, 0.47071253+0.88211865j, 0.47071253-0.88211865j

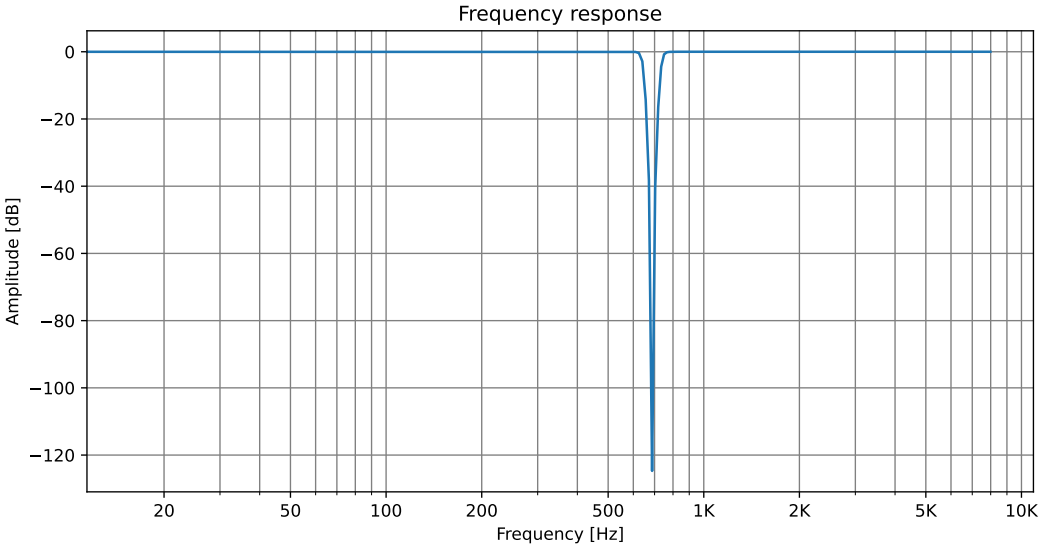
  Póly: 0.45124039+0.88397329j, 0.45124039-0.88397329j, 0.4829066 +0.86724348j, 0.4829066 -0.86724348j, 0.45583671+0.86984453j, 0.45583671-0.86984453j, 0.46885861+0.86305615j, 0.46885861-0.86305615j



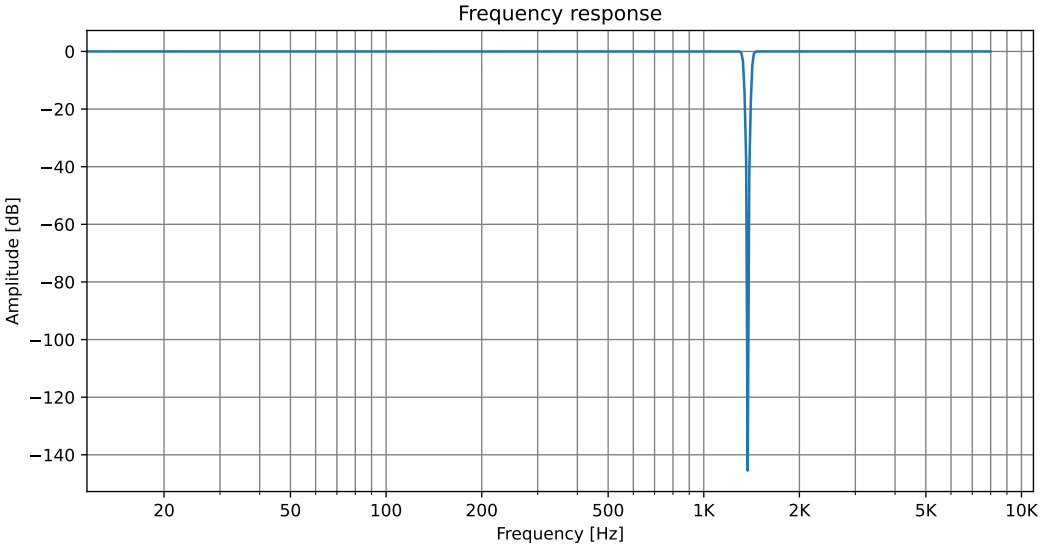
1.9 Úloha 9

1.9.1 Frekvenční charakteristiky jednotlivých filtrů

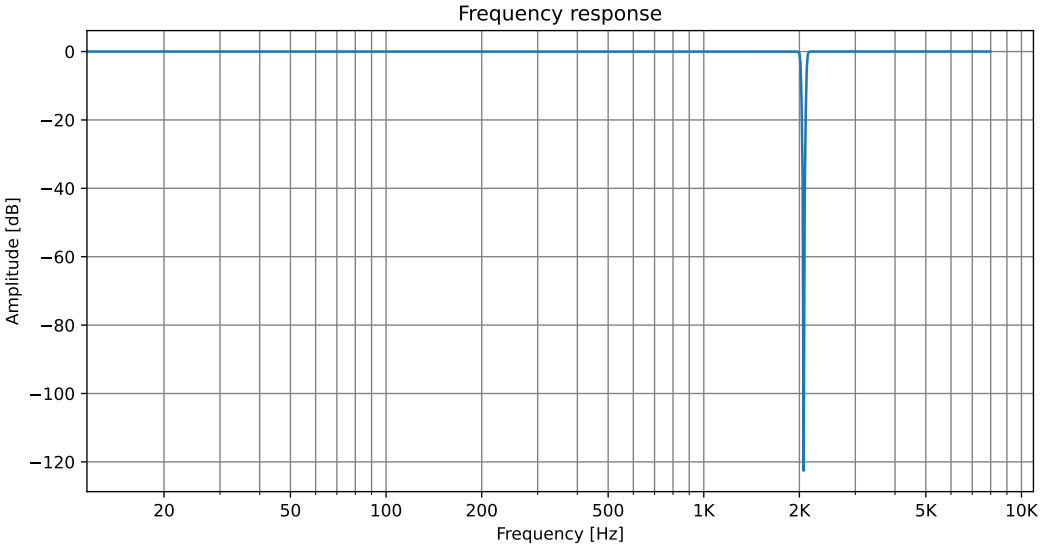
- Frekvenční charakteristika filtru pro frekvenci f_1 (688 Hz)



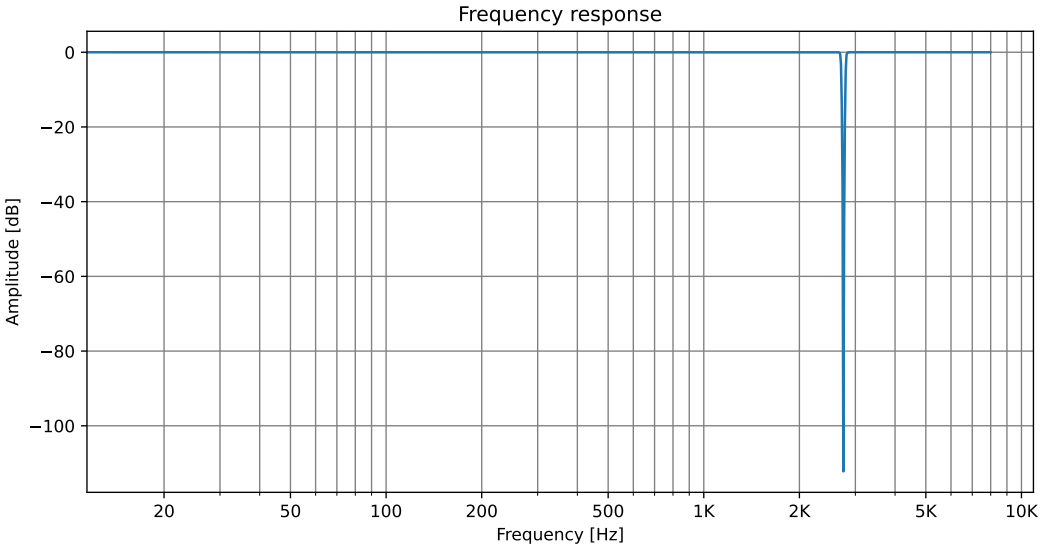
- Frekvenční charakteristika filtru pro frekvenci f_2 (1376 Hz)



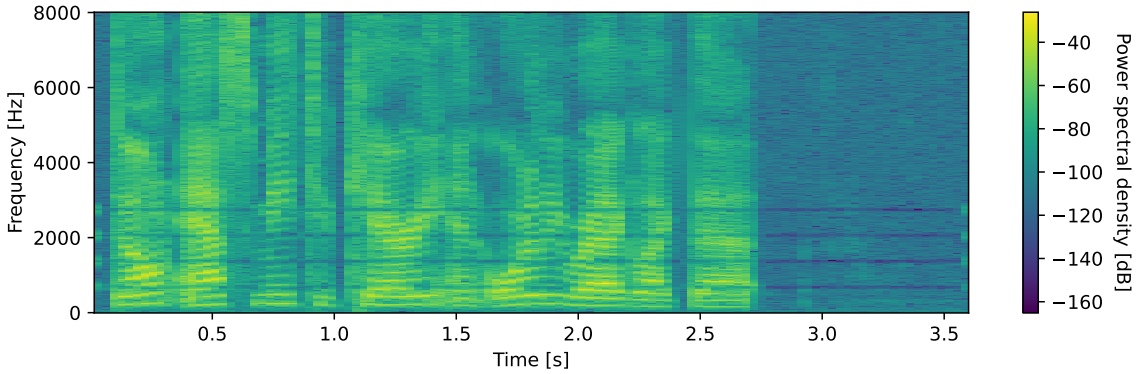
- Frekvenční charakteristika filtru pro frekvenci f_3 (2064 Hz)



- Frekvenční charakteristika filtru pro frekvenci f_4 (2752 Hz)



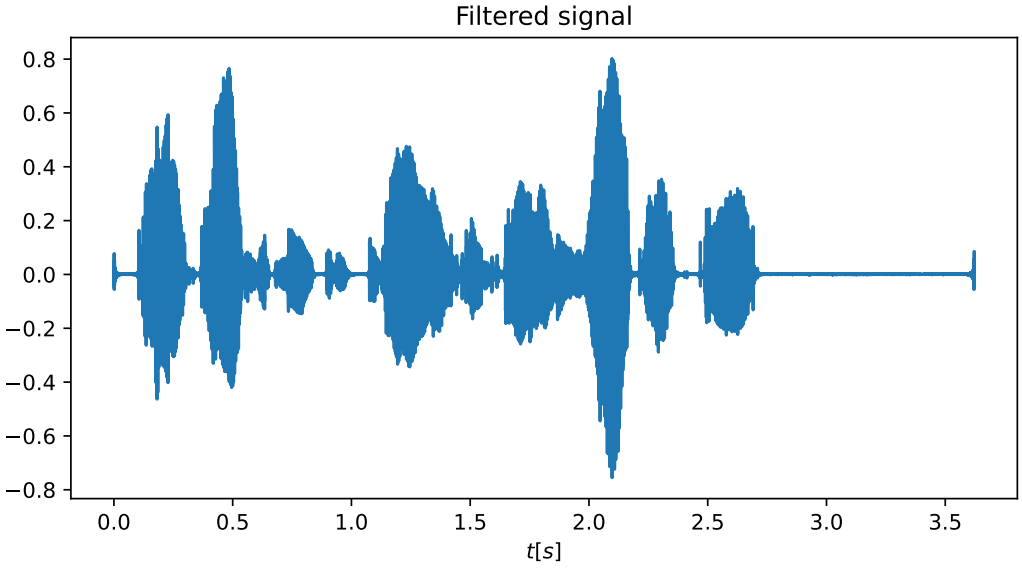
1.9.2 Ověření filtrace na správných frekvencích (spektogram)



Závěr:
Filtry potlačují rušivý signál na správných frekvencích.

1.10 Úloha 10

Po provedení filtrace vypadá výsledný signál následovně:



Před uložením byl signál normalizován do dynamického rozsahu od -1 do +1 pomocí:

```
# Normalize signal to range -1, 1
data = data / max(data.max(), abs(data.min()))
```

Soubor s výsledným signálem je na základě použití 4 pásmových zádrží pojmenován jako ***clean_bandstop.wav***.

Při posouzení výsledného signálu, spektogramu a poslechu výsledné náhravky je možné prohlásit filtraci za úspěšnou.