Protokol ISS projektu

Adam Zvara
xzvara01

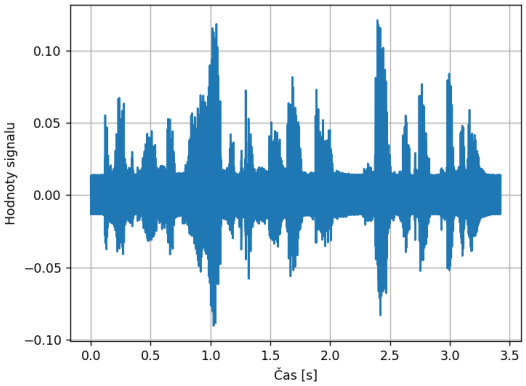
Fakulta Informačních Technologií
Vysoké Učení Technické v Brne
December 21, 2021

4.1 Základy

Súbor načítavam pomocou modulu `scipy.io.wavfile` , po čom ho normalizujem a vypíšem príslušné hodnoty.

```
rate, data = wavfile.read(filename)
data = data / 2**15

max = data.max()
min = data.min()
data_size = len(data)
length = data_size/rate
```

Minimálna hodnota: `-0.090484619140625`
Maximálna hodnota: `0.12091064453125`
Počet vzoriek: `54784`
Dĺžka signálu (v sekundách): `3.424`

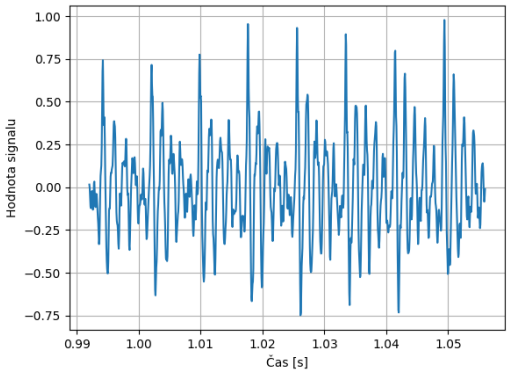


Graf 1: Vstupný zvukový súbor

4.2 Rámce

Signál som ustrednil a normalizoval podľa zadania v jednoduchom cykle. Na vytvorenie jednotlivých rámcov som využil `sliding_window_view` a následne som si nechal zobraziť všetky vytvorené rámce.

```
x_window = sliding_window_view(x_axis, 1024)[::512]
d_window = sliding_window_view(data, 1024)[::512]

for i in range(len(d_window)):
    plt.plot(x_window[i], d_window[i])
    plt.show()
```



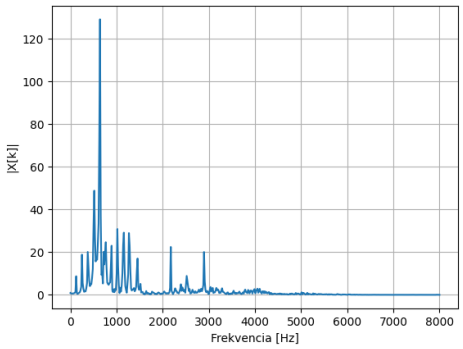
Graf 2: Vybraný periodický (zvučný) rámec

4.3 DFT

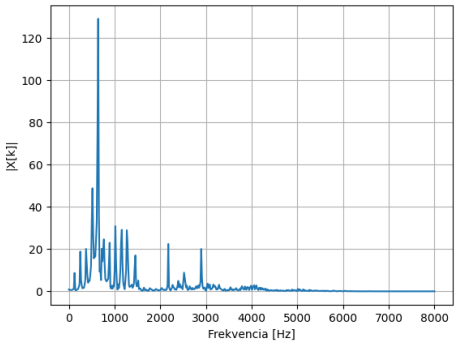
Vytvorím pole `n` hodnôt od 0 do 1023 a stĺpcový vektor `k` (s rovnakými hodnotami). Následne pomocou funkcie `np.exp` vytvorím maticu bází a vynásobím ju so vstupným signálom.

```
def dft(data : list, N = 1024):
    n = np.arange(N)
    k = n.reshape((N,1))
    return np.dot(data, np.exp(-(2j*np.pi*k*n)/N))
```

Použijem DFT funkciu na vybraný rámec pre zistenie frekvencií, z ktorých pozostáva rušivý signál. Výsledok porovnám s funkciou numpy.fft.fft



(a) Vlastná funkcia DFT nad vybraným rámcom

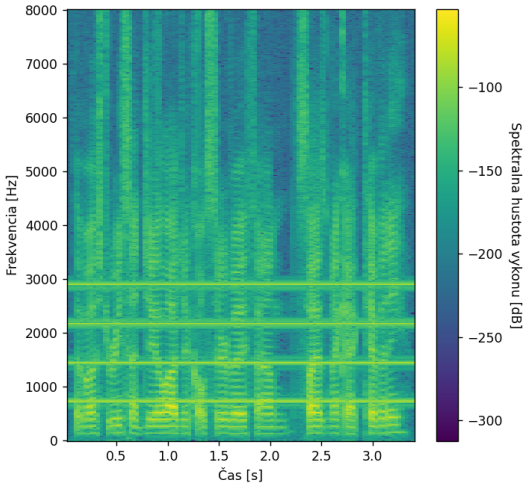


(b) numpy.fft.fft nad vybraným rámcom

4.4 Spektogram

Na vytvorenie spektogramu som použil funkciu scipy.signal.spectrogram , ktorej som nastavil požadované parametre šírky okna a prekrývania. Výsledok som upravil podľa zadania a nechal zobraziť.

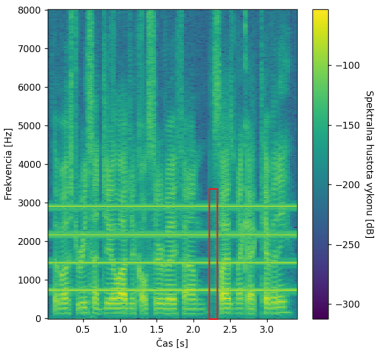
```
# spectogram over data with Fs, window size 1024, overlapping 512
f, t, Sxx = spectrogram(data, rate, nperseg=1024, noverlap=512)
Sxx = 10*np.log10(np.power(np.absolute(Sxx), 2))
```



Graf 3: Spektogram signálu

4.5 Určenie rušivých frekvencií

Na spektograme sú jasne viditeľné 4 vodorovné čiary, ktoré predstavujú konštantný rušivý signál. Frekvencie sa budú pohybovať približne v rozmedzí 500Hz až 3000Hz. Na určenie frekvencií použijem časť spektra, ktorá neobsahuje rozprávanie.



Graf 4: Časť spektogramu na vyčítanie frekvencií

Je vidieť, že hľadané spektrum bez rozprávania je v okolí 2.25 sekundy, čo odpovedá približne 70 indexu v spektrálnej hustote výkonu. Na danom spektre má najvyššie hodnoty (približne nad -90) hustoty práve rušivý signál. Použijem funkciu `get_freq` na vyhľadanie týchto hustôt a z indexu zistím približnú frekvenciu.

```
def get_freq(data : list, treshold : int):
    freq = []
    for index,val in enumerate(data):
        if (val > treshold):
            freq.append((rate//2)*index//len(data))
    return freq

frequencies = get_freq(np.transpose(Sxx)[70], -90)
```

Približné hodnoty frekvencí f_n: 717Hz, 1450Hz, 2167Hz, 2884Hz, pričom platí:

$$2 * f_1 \approx 2 * 717Hz \approx 1434Hz \approx f_2$$

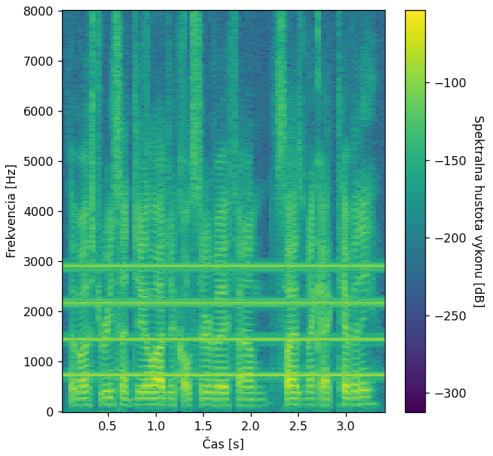
$$3 * f_1 \approx 3 * 717Hz \approx 2151Hz \approx f_3$$

$$4 * f_1 \approx 4 * 717Hz \approx 2868Hz \approx f_4$$

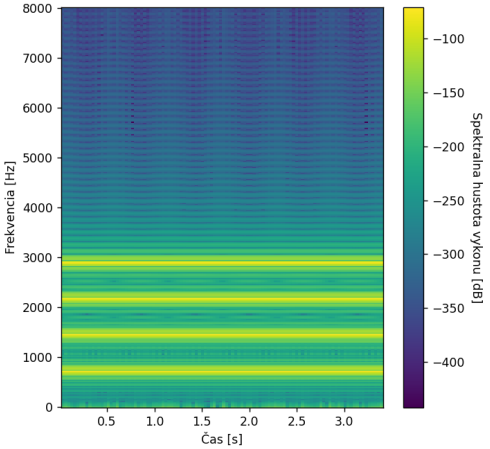
4.6 Generovanie signálu

Na generovanie používam funkciu numpy.cos . Amplitúdu kosínusoviek nastavujem na 0.1 z dôvodu, aby vygenerovaný signál nebol príliš hlučný.

```
t = np.arange(0, length, 1/rate)
cos_data = 0

for i in frequencies:
    f = 0.1*np.cos(2*np.pi *i* t)
    cos_data += f

wavfile.write("audio/4cos.wav", rate, cos_data)
```



(a) Pôvodný spektogram



(b) Spektogram vygenerovaných kosínusoviek

Je vidieť, že sa spektogramy pôvodného signálu a vygenerovaných kosínusoviek zhodujú.

4.7 Filter

Na filtrovanie som použil 4 filtre typu pásmová zádrž, ktoré som vytvoril podľa bodu 3 zo zadania.

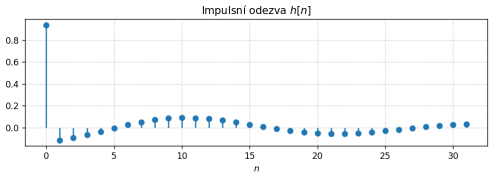
```
stop_width = 15 # polovica sirky nepriepustneho pasma
pass_width = 50 # sirka prechodu do priepustneho pasma

for freq in frequencies:
    stop_freq = [freq-stop_width, freq+stop_width]
    pass_freq = [stop_freq[0]-pass_width, stop_freq[1]+pass_width]

    ord, wn = buttord(pass_freq, stop_freq, 3, 40, fs=rate)
    b, a = butter(ord, wn, 'bandstop', fs=rate)
    # koeficienty filtru b, a
```

Na získanie koeficientov filtrov som použil funkciu scipy.signal.butter , ktorá požaduje výstupy z funkcie scipy.signal.buttord. Do tejto funkcie som predal všetky potrebné parametre ako šírku blokovaného pásma alebo šírku prechodu medzi prechodným a neprechodným pásmom atď.

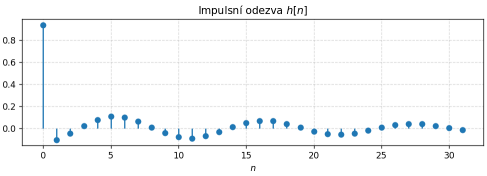
F1	b	0.938	-7.208	24.524	-48.231	59.953	-48.231	24.524	-7.208	0.938
	a	1	-7.562	25.318	-48.996	59.935	-47.451	23.746	-6.869	0.880
F2	b	0.937	-6.311	19.693	-36.840	45.053	-36.840	19.693	-6.311	0.937
	a	1	-6.628	20.343	-37.436	45.038	-36.231	19.054	-6.008	0.877
F3	b	0.936	-4.937	13.510	-23.396	27.977	-23.396	13.510	-4.937	0.936
	a	1	-5.187	13.959	-23.776	27.966	-23.005	13.068	-4.698	0.876
F4	b	0.936	-3.177	7.788	-11.820	14.191	-11.820	7.788	-3.177	0.936
	a	1	-3.337	8.048	-12.013	14.183	-11.621	7.532	-3.023	0.876

Tabuľka 1: Koeficienty jednotlivých filtrov

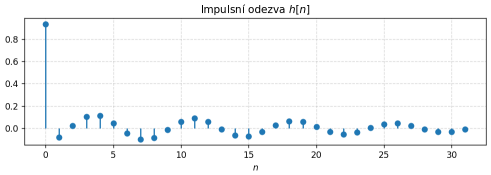
Impulzné odozvy sú zobrazené na 32 vzorkách



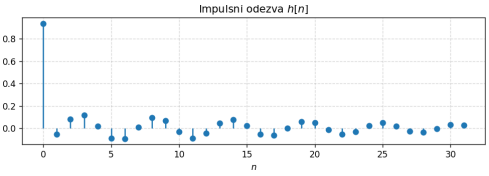
(a) IR F1



(b) IR F2



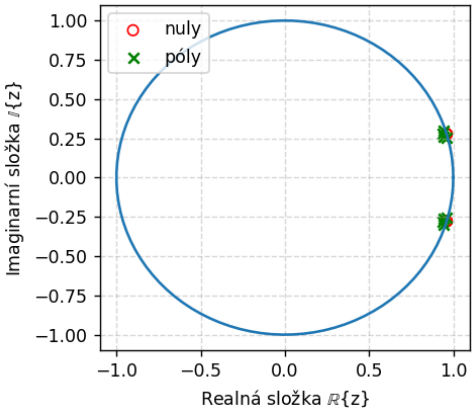
(c) IR F3



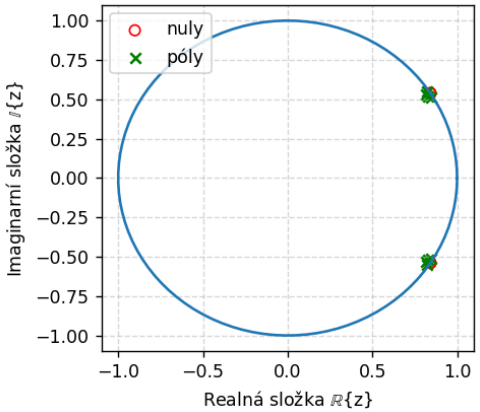
(d) IR F4

4.8 Nulové body a póly

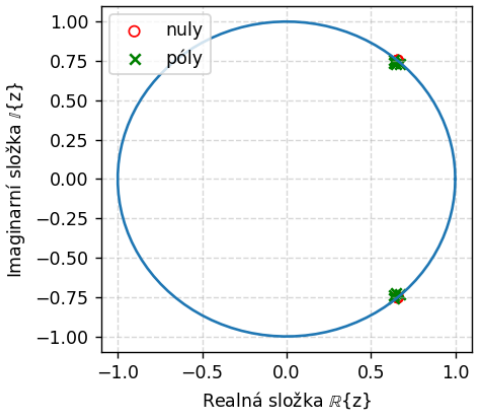
Nulové body a póly počítam pomocou funkcie scipy.signal.tf2zpk.



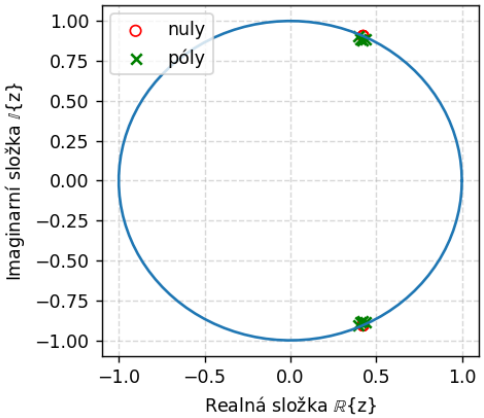
(a) Nuly a póly F1



(b) Nuly a póly F2

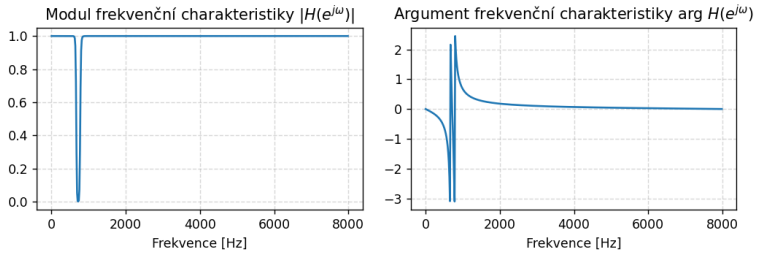


(c) Nuly a póly F3

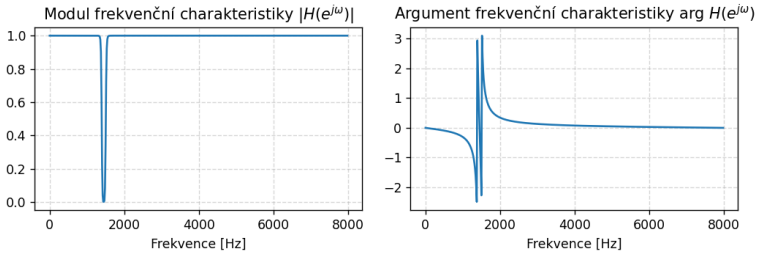


(d) Nuly a póly F4

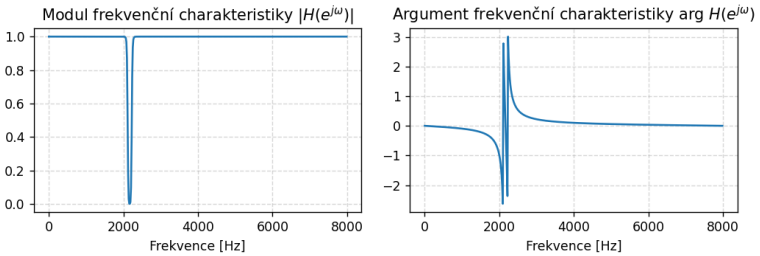
4.9 Frekvenčná charakteristika



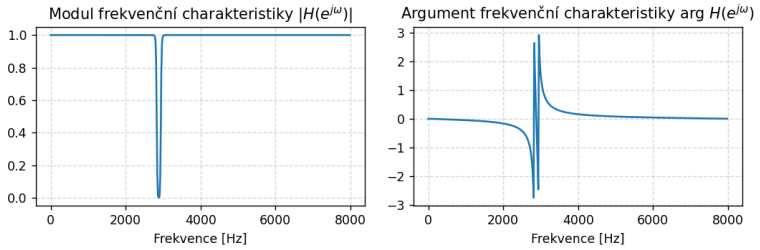
Graf 5: Frekvenčná charakteristika F1



Graf 6: Frekvenčná charakteristika F2



Graf 7: Frekvenčná charakteristika F3



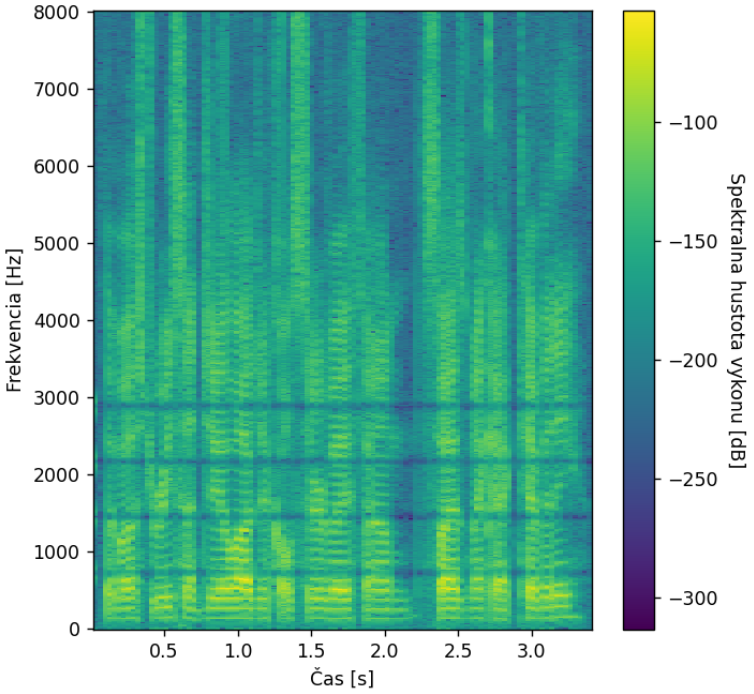
Graf 8: Frekvenčná charakteristika F4

4.10 Filtrácia

Vstupný signál postupne filtrujem podľa vzniknutých filtrov pomocou `scipy.signal.lfilter` a zapíšem do výstupného súboru.

```
# stale v cykle 'for freq in frequencies'
    filtered_data = lfilter(b, a, filtered_data)

wavfile.write("audio/clean_bandstop.wav", rate, filtered_data)
```

Po vypočutí vygenerovaného filtrovaného signálu je hneď počuť, že rušivé frekvencie boli úspešne vyfiltrované. To potvrdzuje aj spektrogram výstupu, na ktorom chýbajú frekvencie rušivého signálu.



Graf 9: Spektrogram vyfiltrovaného vstupného súboru