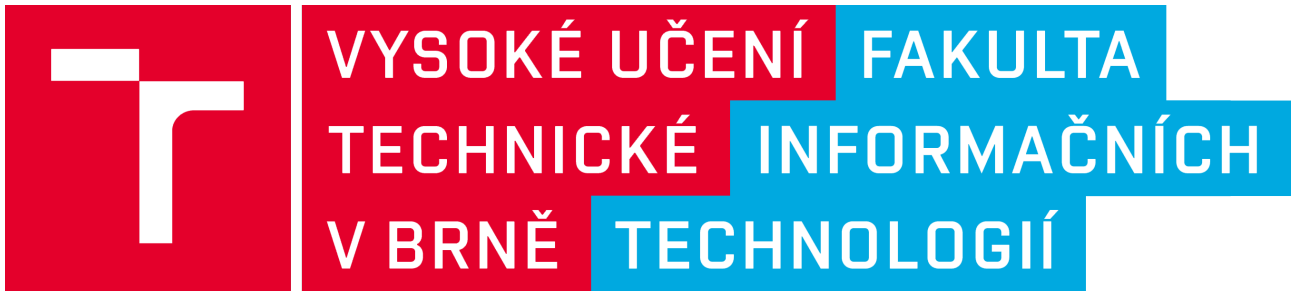
Vysoké učení technické v Brně
Fakulta informačních technologií



Signály a systémy
Projekt

Martin Bublavý (xbubla02)

21. decembra 2021

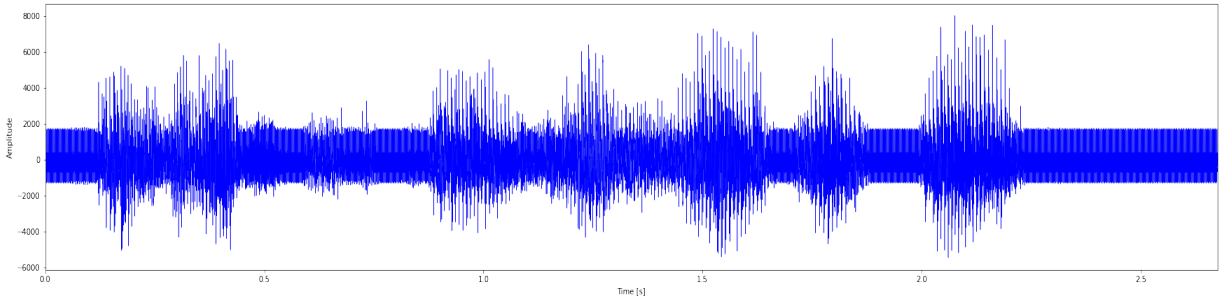
Úloha 1: Základy

- Zvukový súbor bol načítaný pomocou funkcie `wavfile.read`, ktorá je dostupná z knižnice scipy.io.
- Vzorkovacia frekvencia signálu je: 16000 [Hz].
  Bola zistená pomocou parametru, ktorý predáva funkcia `wavfile.read`.
- Dĺžka signálu je: 2.67525 [s].
  Bola zistená pomocou vzorca:

$$\frac{n}{rate}$$

  kde `n` je počet vzorkov signálu a `rate` je vzorkovacia frekvencia.

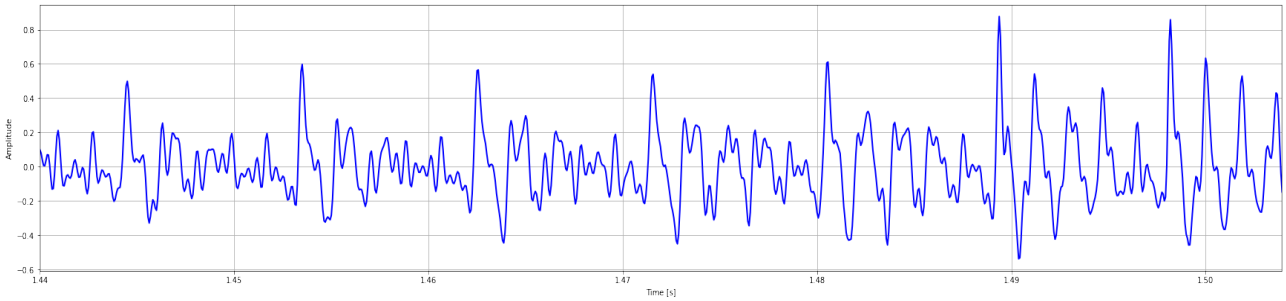
- Dĺžka signálu vo vzorkoch je: 42804 [Vzorkov].
  Bola zistená pomocou parametru, ktorý predáva funkcia `wavfile.read`.
- Minimálna hodnota signálu je: -5470.
  Funkcia `min`.
- Maximálna hodnota signálu je: 8003.
  Funkcia `max`.
- Signál:



Úloha 2: Predzpracovanie a rámce

```python
#Ustrednenie
data = data - np.average(data) #Odčítanie strednej hodnoty

#Normalizácia
data = data / abs(max_d) #Delenie maximálnou hodnotou

#Rozdelenie na rámce
frame = util.view_as_windows(data, window_shape=(1024,), step=512)
frame = frame.T #Transponovanie

#Výber rámca
nice_frame = frame[:, 45]
```

Vybraný rámec:

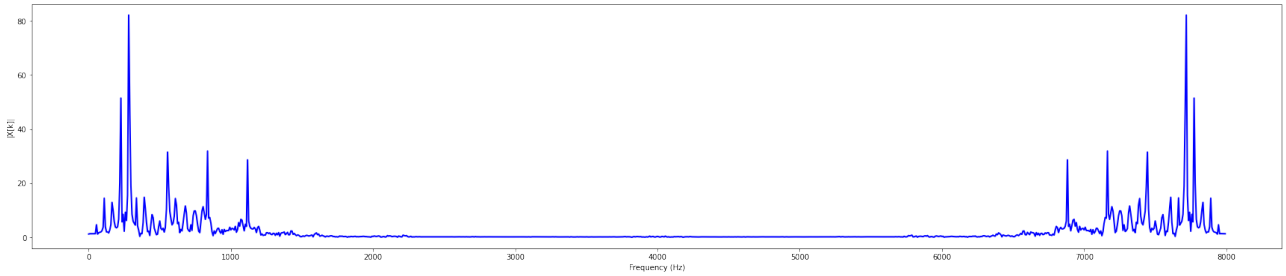


Úloha 3: DFT

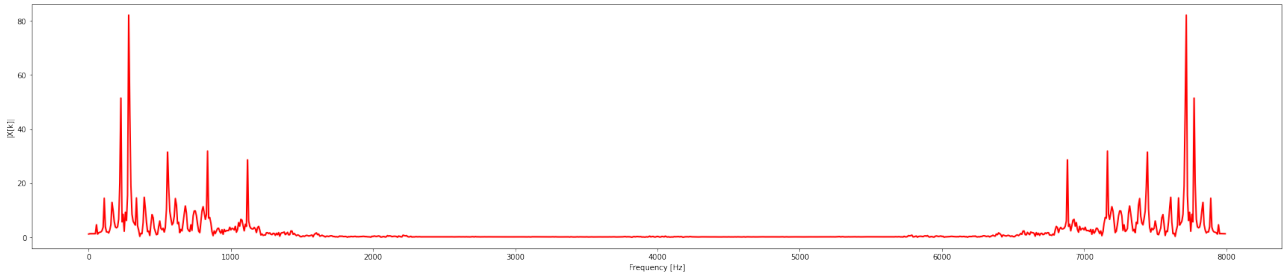
Pre DFT bolo treba vytvoriť vlastnú funckiu:

```
def DFT(array):
    tmp_array = np.empty(1024, dtype = complex) #Vytvorenie prázdneho pola o velkosti 1024
    n = np.arange(1024)


    for k in range(len(array)-1):
        tmp = 0
        #Násobenie bází s vektorom signálu
        tmp = np.dot(array, np.exp((0-1j)*(2*np.pi*n*k/len(array))))
        tmp_n = np.sum(tmp)
        tmp_array[k] = tmp_n
    return tmp_array
```

DFT vybraného rámca pomocou vlastnej funkcie:



DFT vybraného rámca pomocou funkcie z knižnice numpy (`np.fft.fft`):



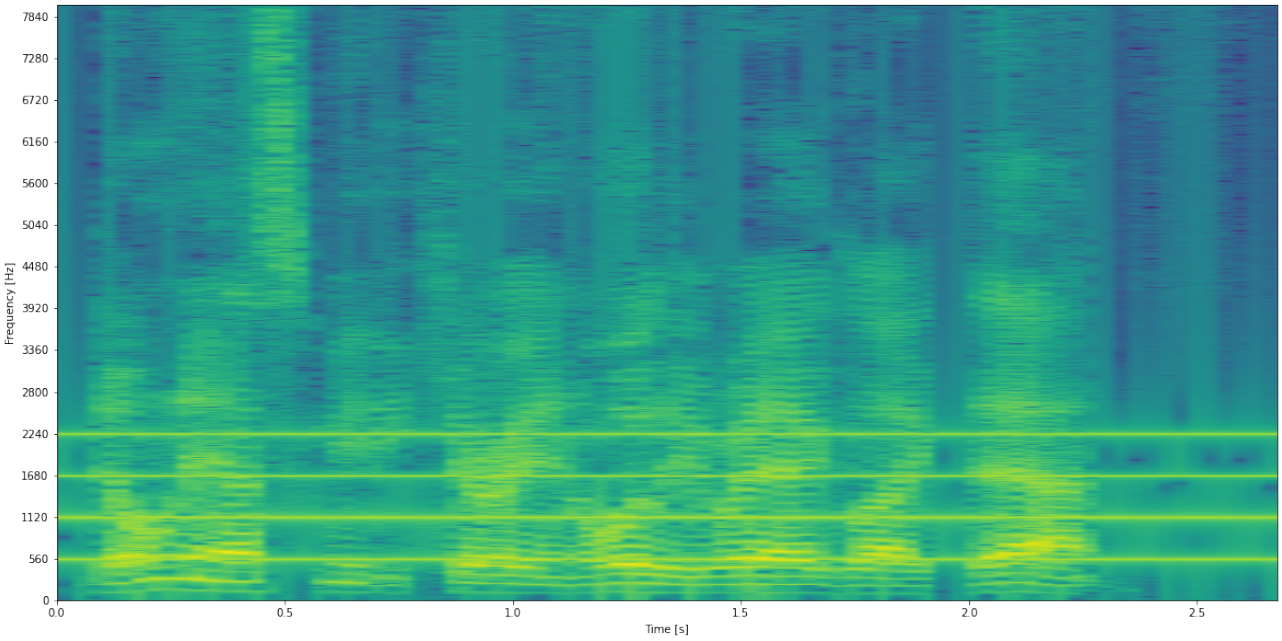
Úloha 4: Spektogram

```
spectrum = np.fft.fft(frame, axis=0)[:1024 // 2 + 1:-1] #DFT nad celým signálom
spectrum = np.abs(spectrum)
spectrum = 10*(np.log10(spectrum**2)) #Úprava koeficientov
```

Najprv sme spravili DFT nad celým signálom. Následovne sme spravili absolútnu hodnotu koeficientov a nakoniec sme upravili podľa vzorca:

$$10 * \log_{10} |X[k]|^2$$

Výsledný spektogram:



Úloha 5: Určenie rušivých frekvencií

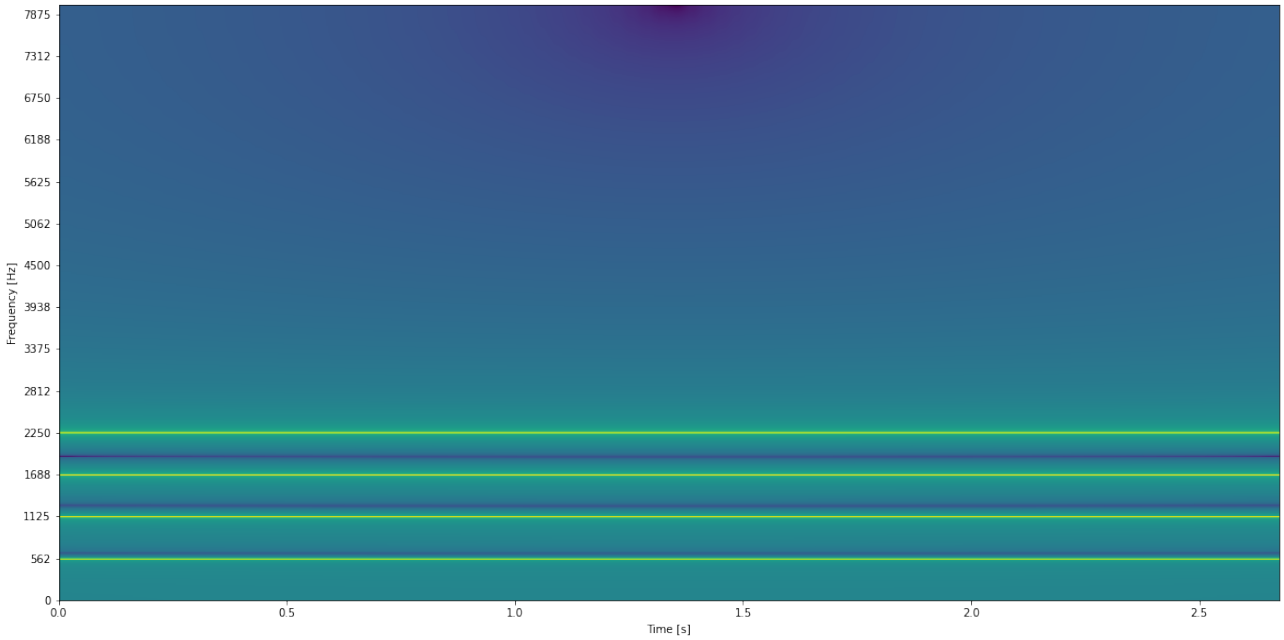
Približné frekvencie boli určené priamo zo spektogramu. Po nastavení ”dobrej”dĺžky krokov na osi ’y’ boli frekvencie ľahko odčítatelné. Presné určenie prebehlo pomocou funkcie `spectrogram` ktorá vracia frekvencie signálu.

f_1 = 562.5 Hz
f_2 = 2*f_1 = 1125 Hz
f_3 = 3*f_1 = 1687.5 Hz
f_4 = 4*f_1 = 2250 Hz

Úloha 6: Generovanie signálu

```python
freq = 562.5 #Frekvencia f1
t = data.shape[0]/rate #Dĺžka signálu v sekundách
FS = int(rate) #Vzorkovacia frekvencia
time = np.linspace(0., t, int(t*FS)) #Vytvorenie časovej osi

cos1 = np.cos(2*np.pi*freq*time)      #Generácia cosinusovky s frekvenciou f1
cos2 = np.cos(2*np.pi*2*freq*time)    #Generácia cosinusovky s frekvenciou 2*f1
cos3 = np.cos(2*np.pi*3*freq*time)    #Generácia cosinusovky s frekvenciou 3*f1
cos4 = np.cos(2*np.pi*4*freq*time)    #Generácia cosinusovky s frekvenciou 4*f1

cos = cos1 + cos2 + cos3 + cos4
sf.write('audio/4cos.wav', cos, FS)   #Zápis výsledného signálu do súboru

window = util.view_as_windows(cos, window_shape=(1024,), step=512) #Rozdelenie signálu na rámce
window = window.T

spectrum = np.fft.fft(window, axis=0)[:1024 // 2 + 1:-1] #DFT nad celým signálom
spectrum = np.abs(spectrum)
spectrum = 10*(np.log10(spectrum**2))  #Úprava koeficientov
```

Výsledný spektogram cosinusoviek:



Úloha 7: Čistiaci filter

Vybral som metódu štyroch pásmových zádrží pomocou funkcie `signal.ellip`.

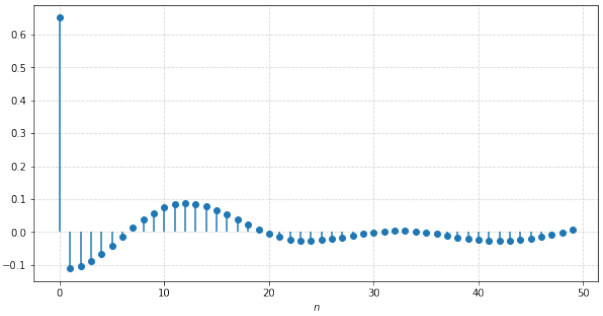
```
sos1 = signal.ellip(4, 3, 80, (freq - 90, freq + 90), 'bandstop', fs = FS, output='sos')       #Filter pre frekvenciu 560Hz
sos2 = signal.ellip(4, 3, 80, (2*freq - 90, 2*freq + 90), 'bandstop', fs = FS, output='sos')   #Filter pre frekvenciu 1120Hz
sos3 = signal.ellip(4, 3, 80, (3*freq - 90, 3*freq + 90), 'bandstop', fs = FS, output='sos')   #Filter pre frekvenciu 1680Hz
sos4 = signal.ellip(4, 3, 80, (4*freq - 90, 4*freq + 90), 'bandstop', fs = FS, output='sos')   #Filter pre frekvenciu 2240Hz
```

- **Filter 1 (Pásmová zádrž pre 562.5Hz):**

  Koeficienty 'a': [1 -7.6392677 25.72773818 -49.88698848 60.91253912
  -47.95817991 23.77827807 -6.78875166 0.85463703]

  Koeficienty 'b': [0.65133783 -5.08702402 17.50418235 -34.65464097 43.17229288
  -34.65464097 17.50418235 -5.08702402 0.65133783]

  Impulzná odozva:

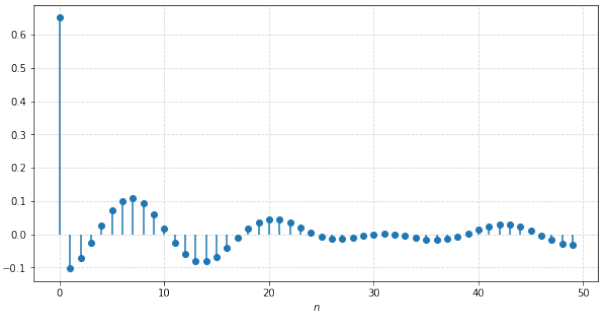


- **Filter 2 (Pásmová zádrž pre 1125Hz):**

  Koeficienty 'a': [1 -7.07779146 22.62635854 -42.55552561 51.44162168
  -40.91020406 20.9119127 -6.28978725 0.85463703]

  Koeficienty 'b': [0.65133783 -4.71313437 15.39454156 -29.56323186 36.46183848
  -29.56323186 15.39454156 -4.71313437 0.65133783]

  Impulzná odozva:

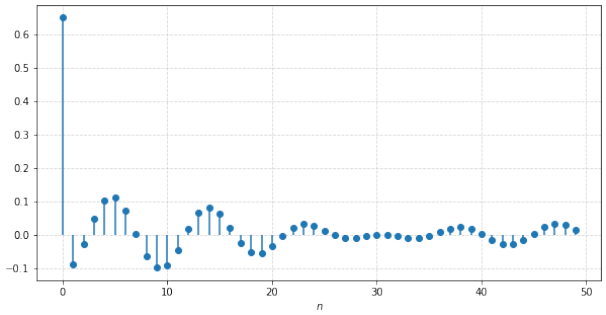


- **Filter 3 (Pásmová zádrž pre 1687.5Hz):**

  Koeficienty 'a': [1 -6.17236472 18.12361891 -32.47147947 38.64571315 -31.21607327 16.75037867 -5.48516597 0.85463703]

  Koeficienty 'b': [0.65133783 -4.11020647 12.33165848 -22.55991062 27.39495912 -22.55991062 12.33165848 -4.11020647 0.65133783]

  Impulzná odozva:

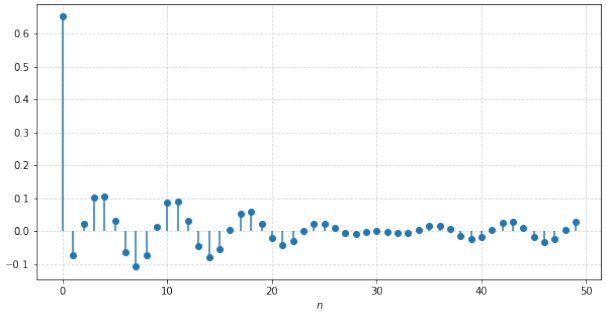


- **Filter 4 (Pásmová zádrž pre 2250Hz):**

  Koeficienty 'a': [1 -4.96698734 13.08414173 -21.95117407 25.66498374 -21.10253953 12.09277962 -4.41398899 0.85463703]

  Koeficienty 'b': [0.65133783 -3.30754005 8.90367227 -15.25291879 18.19627465 -15.25291879 8.90367227 -3.30754005 0.65133783]

  Impulzná odozva:

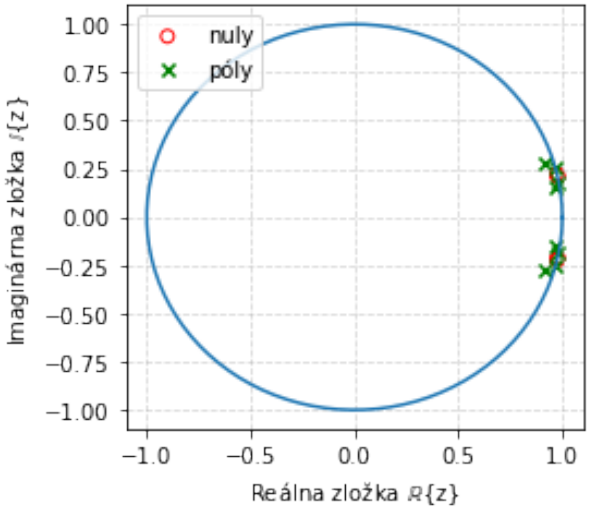


Úloha 8: Nulové body a póly

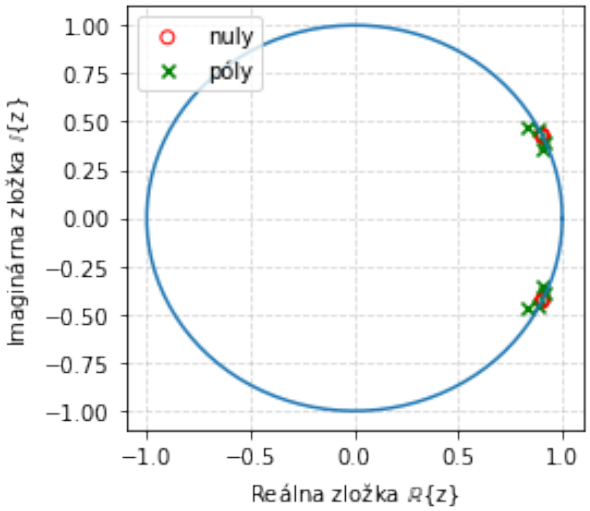
Pre zobrazenie núl a pólov bola použítá funkcia `zplane`, ktorá z koeficientov získa nuly a póly pomocou funkcie `np.roots` a zobrazí ich v komplexnej rovine.

```python
def zplane(a, b):

    z = np.roots(b) #Získanie núl
    p = np.roots(a) #Získanie pólov


    plt.figure(figsize=(4,3.5))
    ang = np.linspace(0, 2*np.pi,100)
    plt.plot(np.cos(ang), np.sin(ang))
    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')

    plt.gca().set_xlabel('Reálna zložka $\mathbb{R}\{$z$\}$')
    plt.gca().set_ylabel('Imaginárna zložka $\mathbb{I}\{$z$\}$')
    plt.grid(alpha=0.5, linestyle='--')
    plt.legend(loc='upper left')
    plt.tight_layout()
    plt.show()
```

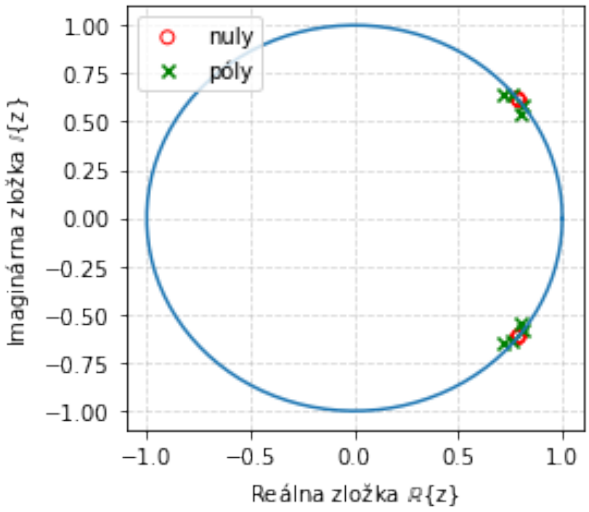
- **Filter 1 (Pásmová zádrž pre 562.5Hz):**



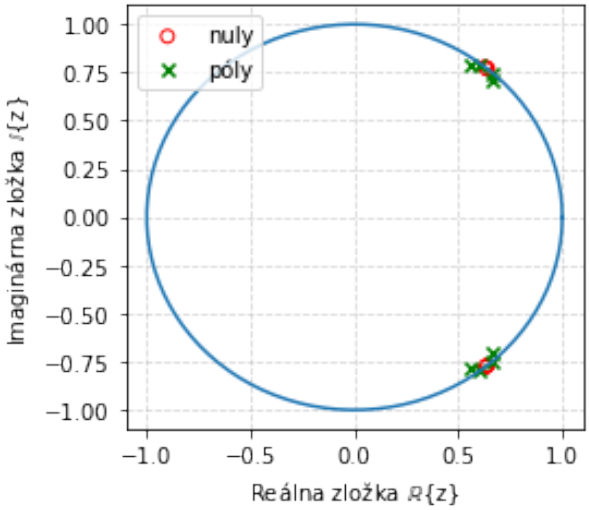
- **Filter 2 (Pásmová zádrž pre 1125Hz):**



- **Filter 3 (Pásmová zádrž pre 1687.5Hz):**

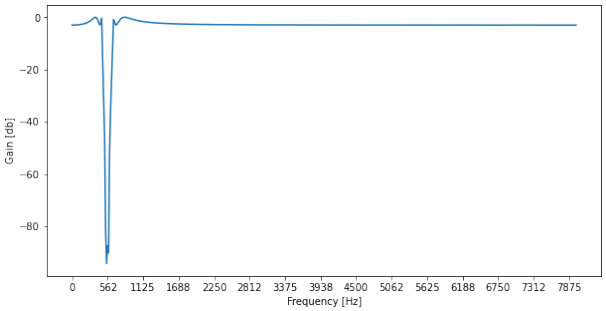


- **Filter 4 (Pásmová zádrž pre 2250Hz):**

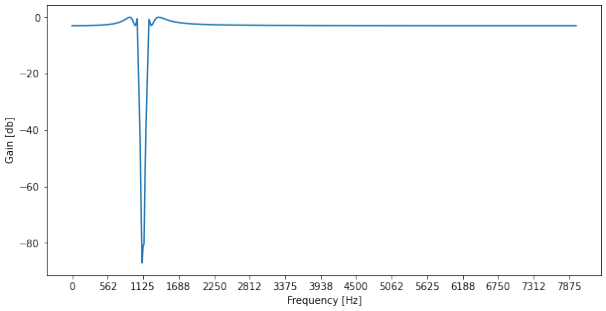


Úloha 9: Frekvenčná charakteristika

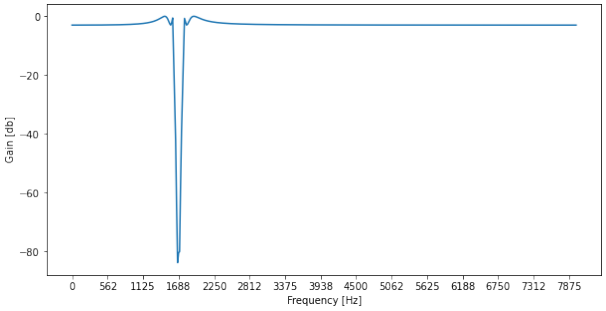
- **Filter 1 (Pásmová zádrž pre 562.5Hz):**



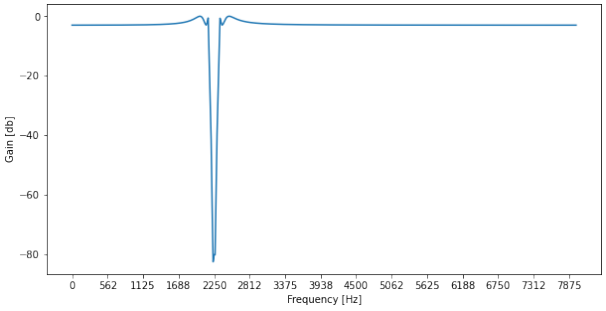
- **Filter 2 (Pásmová zádrž pre 1125Hz):**



- **Filter 3 (Pásmová zádrž pre 1687.5Hz):**



- **Filter 4 (Pásmová zádrž pre 2250Hz):**



Úloha 10: Filtrácia

Vstupný signál bol postupne filtrovaný cez všetky 4 filtre pomocou funkcie `sosfilt` a nakoniec zapísaný do súboru ”clean_bandstop.wav”

```
filtered1 = signal.sosfilt(sos1, data)      #Filtrácia 1. cosinusovky
filtered2 = signal.sosfilt(sos2, filtered1) #Filtrácia 2. cosinusovky
filtered3 = signal.sosfilt(sos3, filtered2) #Filtrácia 3. cosinusovky
filtered4 = signal.sosfilt(sos4, filtered3) #Filtrácia 4. cosinusovky

filename = 'audio/clean_bandstop.wav'
sf.write(filename, filtered4, FS)
```