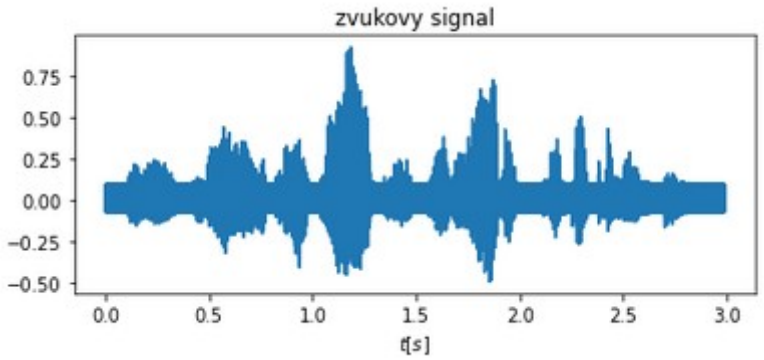
Martin Pentrák
xpentr00
ISS

3.1.2022

Martin Pentrak xpentr00

4.1



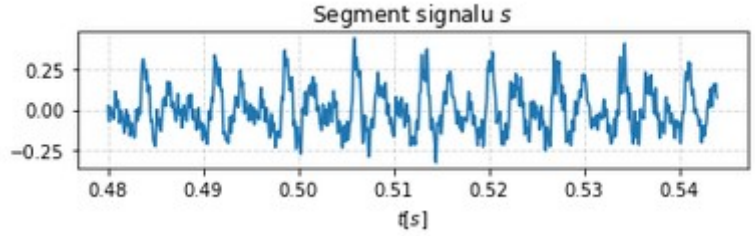
-signál má dĺžku **47821** vzorkov, ak to videlíme frekvenciou dostaneme čas **2.9889 s**
-max. hodnota signálu(po normalizaci) je **0.9256** a min. hodnota je **-0.4962**
-signál je normalizovaný

4.2

-na ustredneni signálu som použil tento vzorec **s=s-np.mean(s)**
-signál som normalizoval do dynamického rozsahi -1 až 1 už v prvej úlohe

```
while(frame+1024<s.size):
    odkud_vzorky = frame
    pokud_vzorky = frame + 1024
    s_seg=s[odkud_vzorky:pokud_vzorky]
    frames.append(s_seg)
    frame=frame+512
    i=i+1
frames=np.asarray(frames)
```

-v tomto cykle som si rozdelil signál na rámce o velkosti 1024 vzoriek s prekritím 512 vzoriek,posledné 2 rámce som zahodil lebo nemali dostačujúcu veľkosť



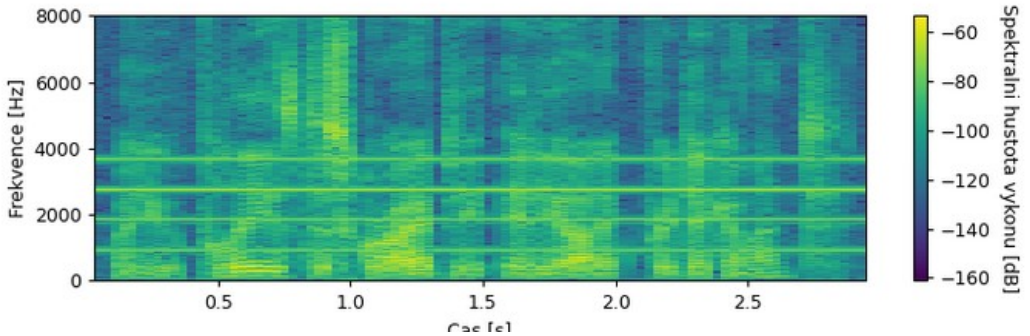
-tento rámec som vybral ako znelý, jedná sa o 17 rámec v poli

4.4

```
f, t, sgr = spectrogram(s, fs,nperseg=1024,noverlap = 512)
sqr_log = 10 * np.log10(sgr+1e-20)
```

```
plt.figure(figsize=(9,3))
```

-získané hodnoty som upravil podla vzorca na druhom riadku



-výsledný spektogram

4.5

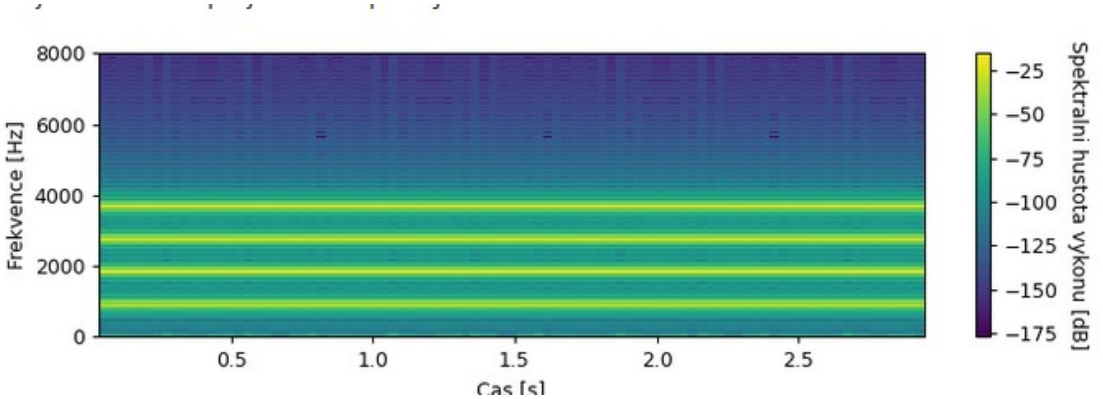
Rušivé frekvencie sú **920 HZ**, **1840 HZ**, **2760 HZ**, **3680 HZ** , jednotlivé frekvenice sú 2,3,4 násobky prvedj frekvencie, tieto údaje som odčítal zo spekogramu po priblížení.

4.6

```
cos1 = np.cos(2 * np.pi * 920 * pocet_vzorkov)
cos2 = np.cos(2 * np.pi * 1840 * pocet_vzorkov
cos3 = np.cos(2 * np.pi * 2760 * pocet_vzorkov
cos4 = np.cos(2 * np.pi * 3680 * pocet_vzorkov

cos_total = cos1 + cos2 + cos3 + cos4
print(cos_total)
```

-vygeneroval som si signál pre jenotlivé cosinusovky a potom som tieto cosinusovky spojil do jedného signálu s rovankou dlžkou akú mal pôvodný signál



-spektogram pre cosinusovky

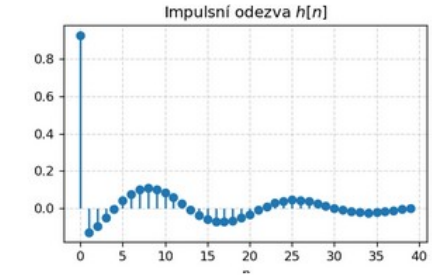
4.7

-v nasledijúcich riadkoch som is vygeneroval koeficienty pre jednotlivé filtre
-koeficienty filtrov sú a1 ,b1 až a4 , b4

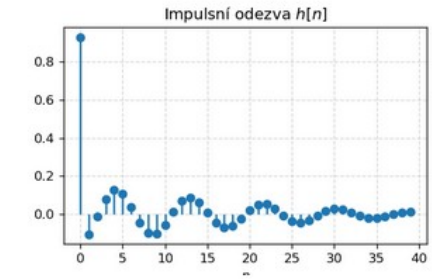
```
#wp=[(920-50)/(0.5*fs),(920+50)/(0.5*fs)]
#ws=[(920-20)/(0.5*fs),(920+20)/(0.5*fs)]
N ,wn =sc.signal.buttord([(920-50)/(0.5*fs),(920+50)/(0.5*fs)],[(920-20)/(0.5*fs),(920+20)/(0.
N2 ,wn2 =sc.signal.buttord([(1840-50)/(0.5*fs),(1840+50)/(0.5*fs)],[(1840-20)/(0.5*fs),(1840+2
N3 ,wn3 =sc.signal.buttord([(2760-50)/(0.5*fs),(2760+50)/(0.5*fs)],[(2760-20)/(0.5*fs),(2760+2
N4 ,wn4 =sc.signal.buttord([(3680-50)/(0.5*fs),(3680+50)/(0.5*fs)],[(3680-20)/(0.5*fs),(3680+2
```

```
b1,a1 = sc.signal.butter(N,wn,'bandstop')
b2,a2 = sc.signal.butter(N2,wn2,'bandstop')
b3,a3 = sc.signal.butter(N3,wn3,'bandstop')
b4,a4 = sc.signal.butter(N4,wn4,'bandstop')
```

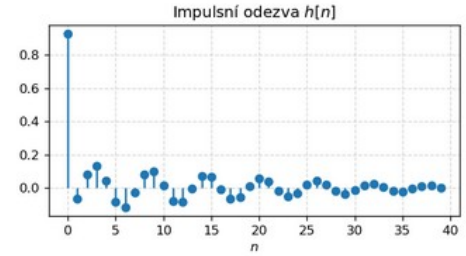
-na tomto obrázku môžeme vydieť impulzné odozvy



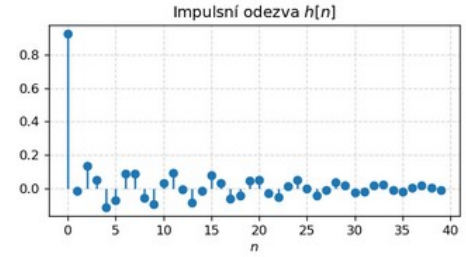

```
<IPython.core.display.Javascript object>
```




```
<IPython.core.display.Javascript object>
```



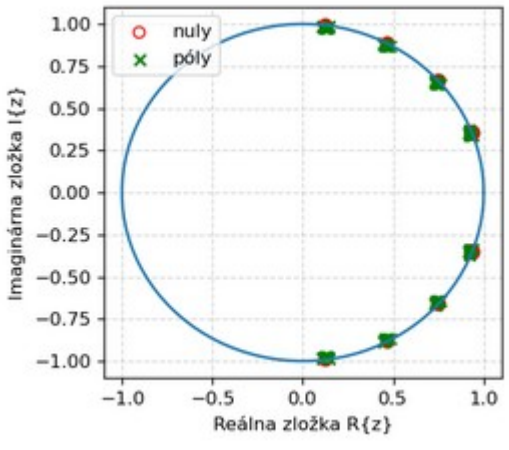

```
<IPython.core.display.Javascript object>
```



4.8

```
z, p, k = tf2zpk(b1, a1)
z2, p2, k2 = tf2zpk(b2, a2)
z3, p3, k3 = tf2zpk(b3, a3)
z4, p4, k4 = tf2zpk(b4, a4)
```

-pomocou tejto funkcie som si zistil nuly a póly pre jednotlivé filtre



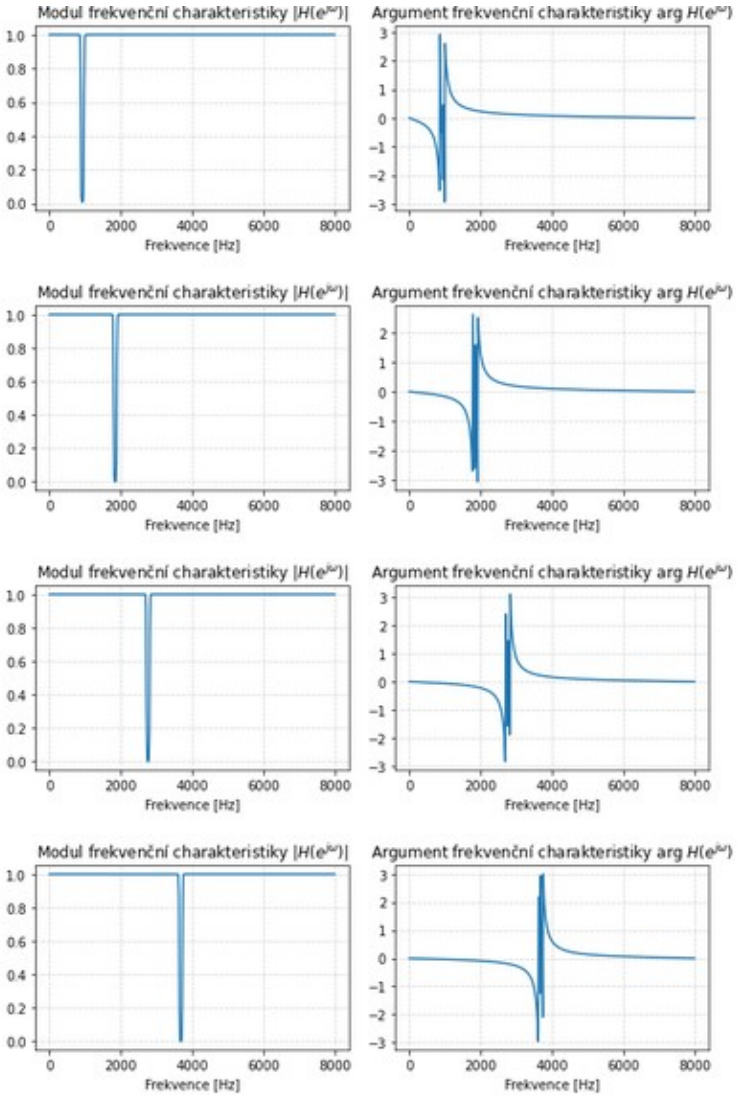
-nuly a póly v komplexnej rovine

4.9

```
w1, H1 = freqz(b1,a1)
w2,H2=freqz(b2,a2)
w3,H3=freqz(b3,a3)
w4,H4=freqz(b4,a4)
```

-pomocou tejto funkcie som si vygeneroval parametre pre jednotlivé frekvenčné charakteristiky

-frekvenčné charakteristky



4.10

```
sf = lfilter(b1,a1,s)

sf = lfilter(b2,a2,sf)

sf = lfilter(b3,a3,sf)

sf = lfilter(b4,a4,sf)
```

-filtrovanie signálu postupne jednotlivými filtrami (výstup z jedného filtra ide do ďalšieho)
-posluchom som prišiel nato že sa signál vyčistil aj keď na začiatku je počuť ešte zvyšky šumu

Martin Pentrak xpentr00