ISS projekt

2021/22

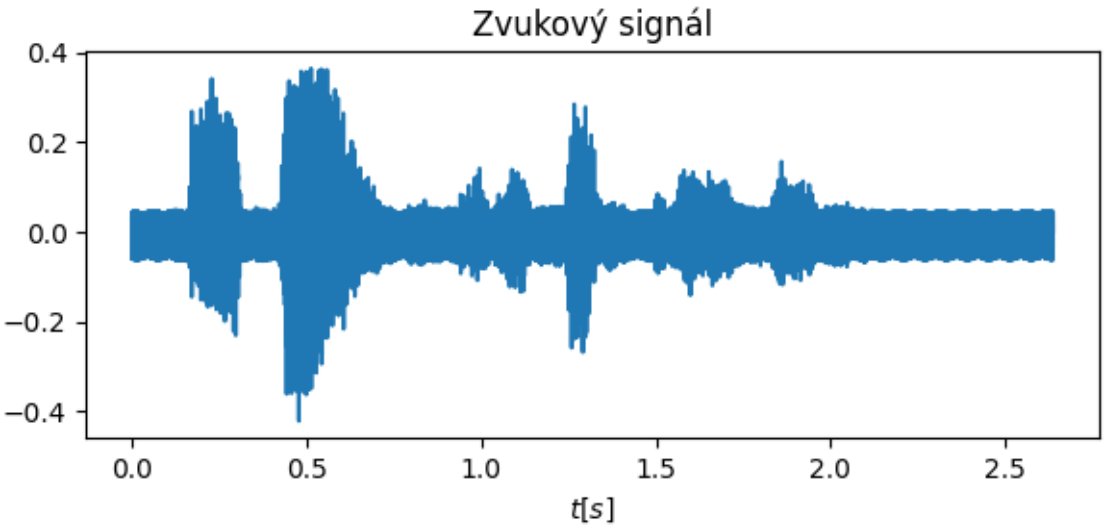
Matej Koreň

5.1.2022

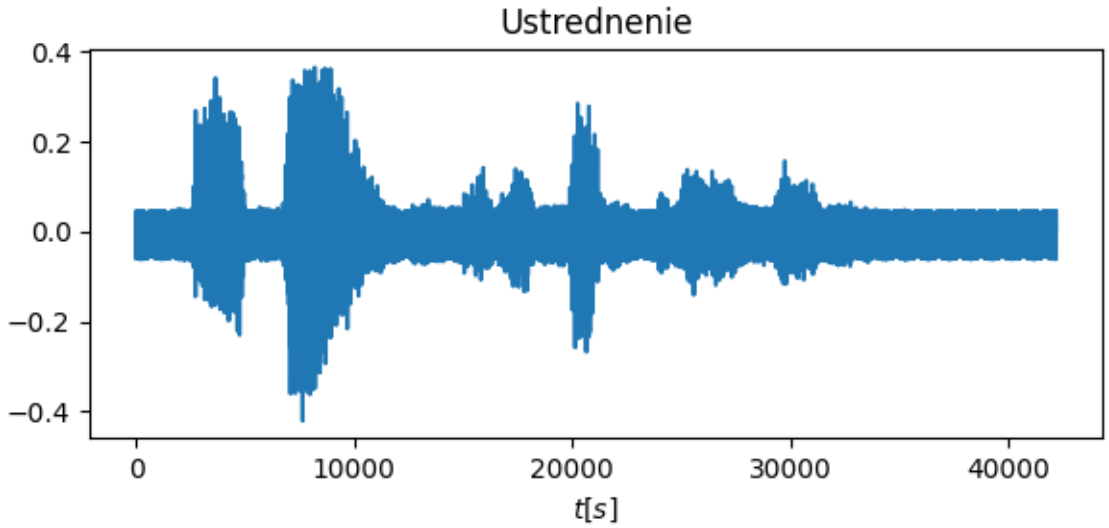
Úloha 1.)

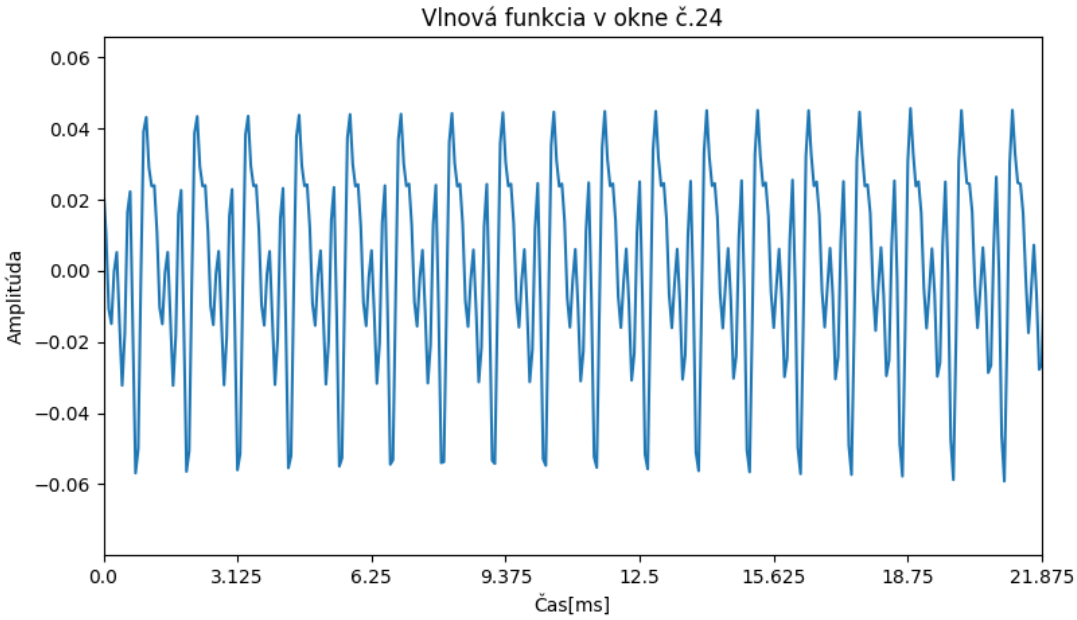
```
Vzorkovacia frekvencia: 16000
Dĺžka vo vzorkoch: 42189
Dĺžka [s]: 2.6368125
Maximum: 0.364990234375
Minimum: -0.420867919921875
```

(Signál som načítaval pomocou *sf.read()*, ktoré ho automaticky znormalizovalo.)



Úloha 2.)



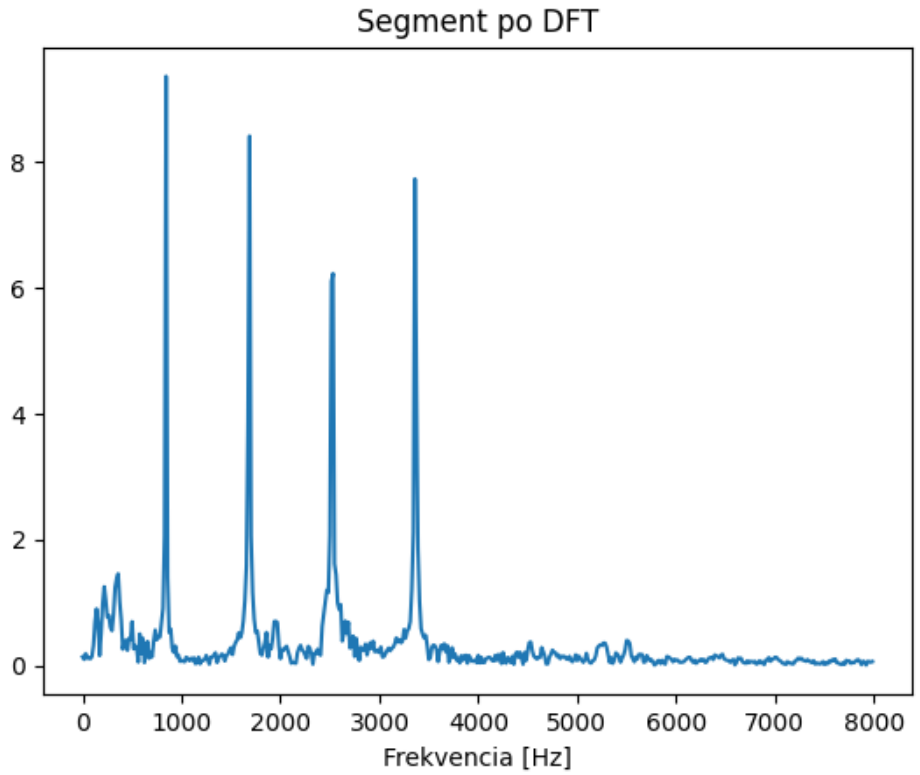


Znelý rámec

Úloha 3.)

Implementácia:

```
################
# Vektorová dft
################
def dft(array):
    x = np.asarray(array, dtype=float)
    N = len(array)
    n = np.arange(2)
    k = n[:, np.newaxis]
    dft_mat = np.exp(-2j * np.pi * n * k / 2)
    ft_x = np.matmul(dft_mat, x.reshape((2, -1)))

    while ft_x.shape[0] < N:
        ft_x_even = ft_x[:, :ft_x.shape[1] // 2]
        ft_x_odd = ft_x[:, ft_x.shape[1] // 2:]
        factor = np.exp(-1j * np.pi * np.arange(ft_x.shape[0]) / ft_x.shape[0])[:, np.newaxis]
        ft_x = np.vstack([ft_x_even + factor * ft_x_odd, ft_x_even - factor * ft_x_odd])

    return ft_x.ravel()
```

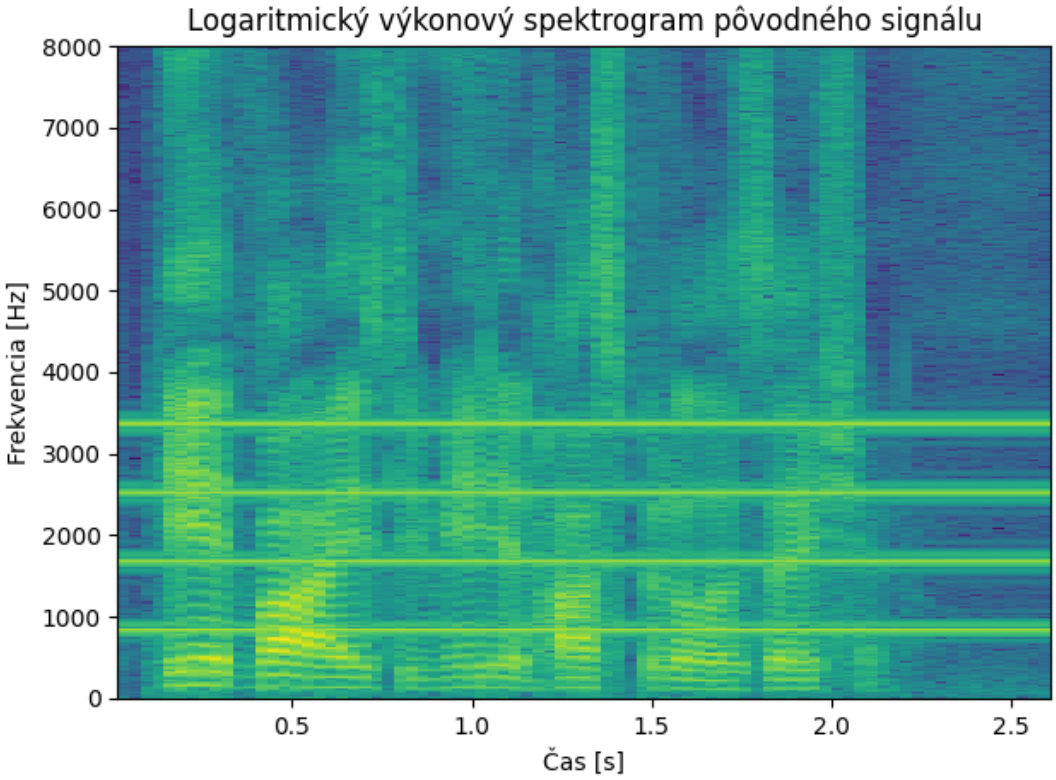


Podobnosť s *numpy.fft.fft()* :

```
start = timer()
dft_frame = dft(nice_frame)
end = timer()


proximity = np.allclose(dft_frame, np.fft.fft(nice_frame))
print("Podobnosť s fft : " + str(proximity) + ", Čas : " + str(timedelta(seconds=end-start)))
```

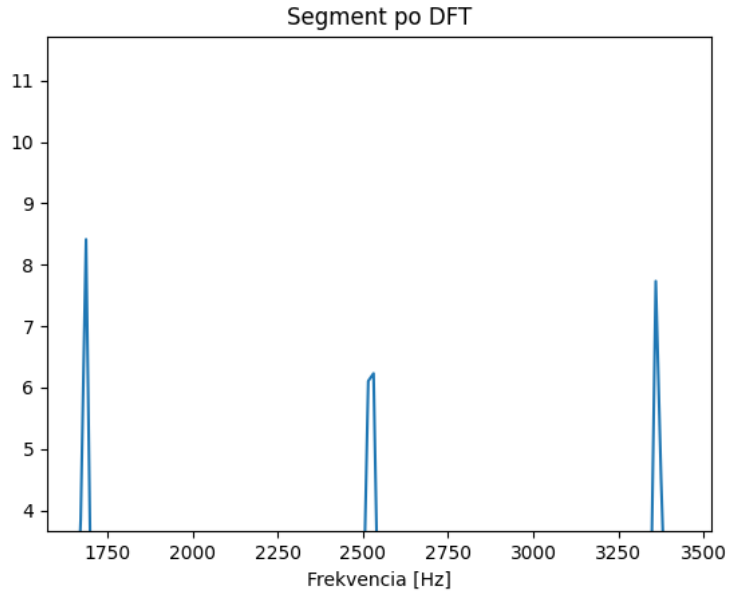
```
Podobnosť s fft : True, Čas : 0:00:00.000504
```

Úloha 4.)



Úloha 5.)

Prvá rušivá frekvencia mala zároveň aj najvyšší koeficient, teda sa dala nájsť jednoducho funkciou. Druhá bola jej dvojnásobok, avšak zvyšné už nie – (viď. odskok na grafe dft). Zvyšok som teda odčítal a skontroloval manuálne.



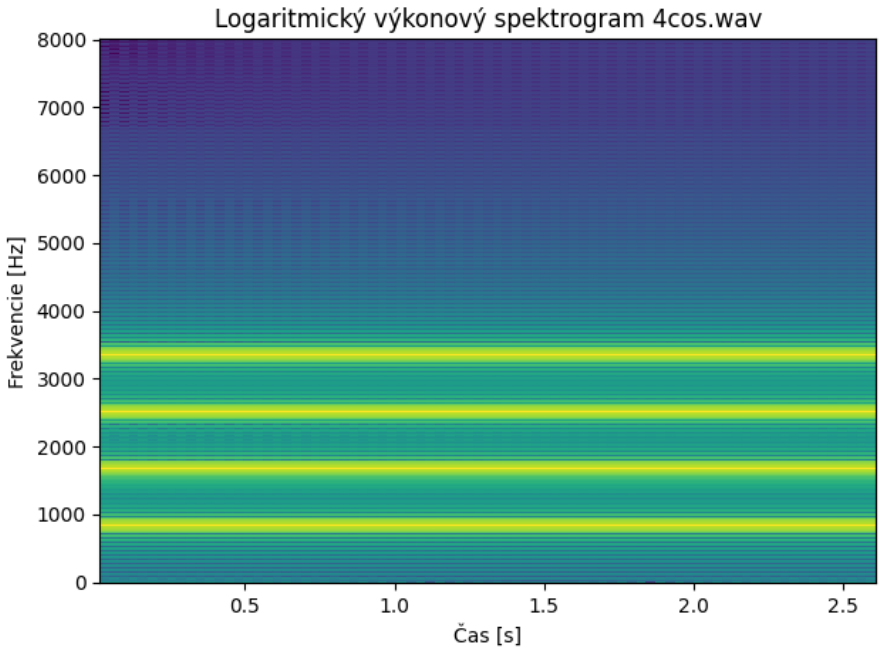
```
####################
# Rušivé frekvencie
####################

dist_freqs = [0, 0, 0, 0]
dist_freqs[0] = (spec_axis[np.argmax(np.abs(dft_frame[:100]))])
dist_freqs[1] = dist_freqs[0] * 2
dist_freqs[2] = 2515.5
dist_freqs[3] = 3359.3

print("Rušivé frekvencie : " + str(dist_freqs))
```

```
Rušivé frekvencie : [843.75, 1687.5, 2515.5, 3359.3]
```

Úloha 6.)

S využitím nájdených frekvencii som vytvoril 4 kosínusovky a sčítal ich dokopy. Pri zápise som ich vydelil (znížil amplitúdu) kvôli ochrane sluchu.

```
u = np.arange(0, len(s), 1)
y1 = np.cos(dist_freqs[0] / fs * 2 * np.pi * u)
y2 = np.cos(dist_freqs[1] / fs * 2 * np.pi * u)
y3 = np.cos(dist_freqs[2] / fs * 2 * np.pi * u)
y4 = np.cos(dist_freqs[3] / fs * 2 * np.pi * u)
y = y1+y2+y3+y4

sf.write('audio/4cos.wav', y/69, fs)
```



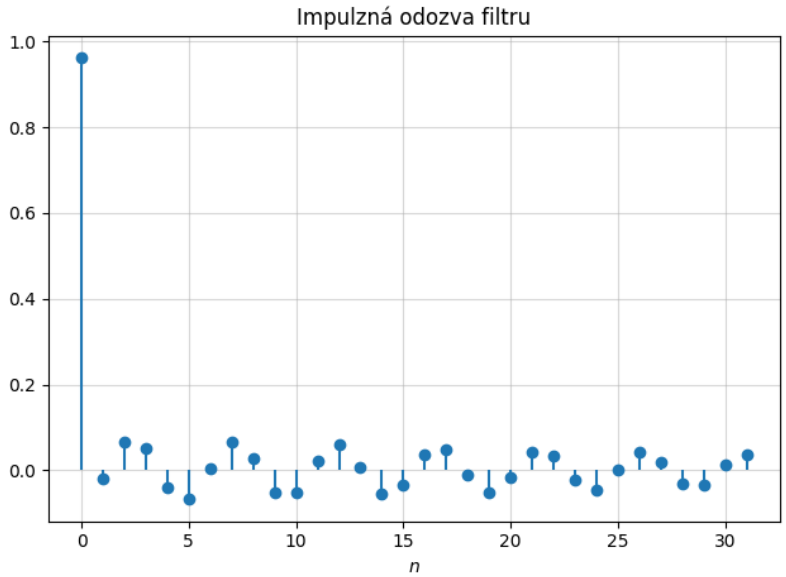
Úloha 7.)

Vybral som si návrh 4 pásmových zádrží pomocou *scipy.signal.butter* a *scipy.signal.buttord*.

Šírka priepustného pásma – 50Hz, šírka záverného pásma – 10Hz

```
wp = 50
ws = 5
filter_a = filter_b = 1

for f in dist_freqs:
    N, wn = signal.buttord([f-wp, f+wp], [f-ws, f+ws], 3, 40, False, fs)
    b, a = signal.butter(N, wn, 'bandstop', False, 'ba', fs)

# Spojenie zádrží do filtrov
    filter_a = np.convolve(filter_a, a)
    filter_b = np.convolve(filter_b, b)
```

```
Koeficient A :[ 1.00000000e+00 -1.50048461e+01  1.13268136e+02 -5.70541983e+02
  2.14767172e+03 -6.41778777e+03  1.57962016e+04 -3.28098448e+04
  5.84740447e+04 -9.04613543e+04  1.22454712e+05 -1.45806141e+05
  1.53163661e+05 -1.42080493e+05  1.16276870e+05 -8.37030419e+04
  5.27233936e+04 -2.88276267e+04  1.35246203e+04 -5.35463149e+03
  1.74618067e+03 -4.52056270e+02  8.74590121e+01 -1.12909613e+01
  7.33357868e-01]
Koeficient B :[ 8.56363164e-01 -1.30171896e+01  9.95471380e+01 -5.07982473e+02
  1.93718817e+03 -5.86452863e+03  1.46232529e+04 -3.07707183e+04
  5.55566862e+04 -8.70708551e+04  1.19403719e+05 -1.44027806e+05
  1.53267317e+05 -1.44027806e+05  1.19403719e+05 -8.70708551e+04
  5.55566862e+04 -3.07707183e+04  1.46232529e+04 -5.86452863e+03
  1.93718817e+03 -5.07982473e+02  9.95471380e+01 -1.30171896e+01
  8.56363164e-01]
```

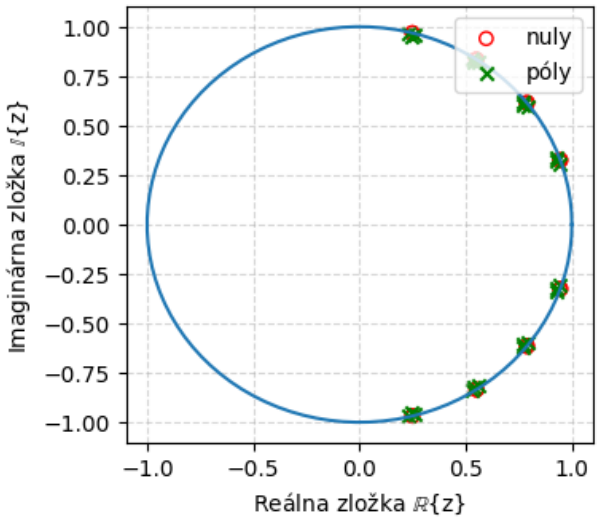


Z impulznej odozvy môžeme vidieť, že sa jedná o IIR filter. (Vybral som preto len 32 vzoriek.)

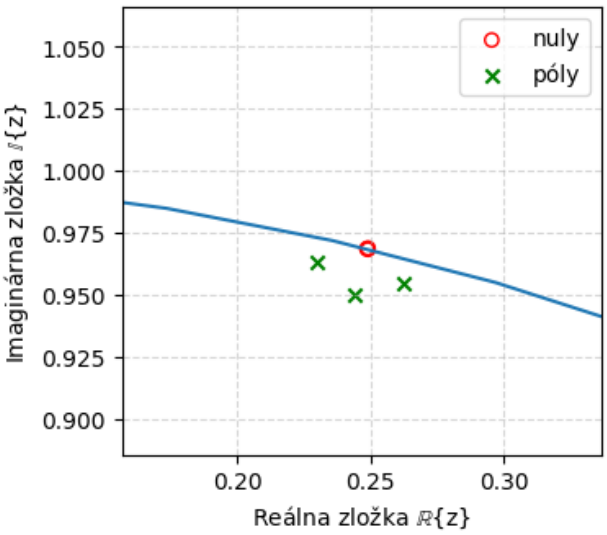
Úloha 8.)

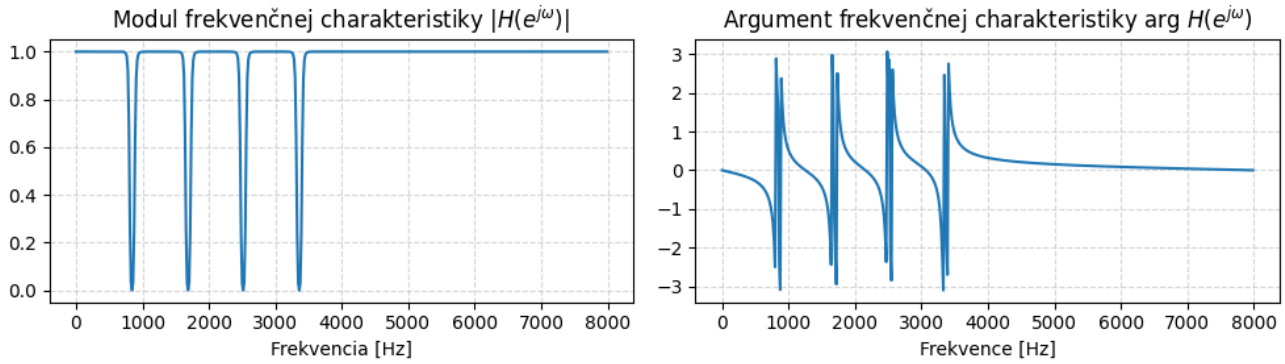
```
##############
# Nuly a póly
##############

z, p, k = tf2zpk(filter_b, filter_a)
w, H = freqz(filter_b, filter_a)

############
# Stabilita
############

is_stable = (p.size == 0) or np.all(np.abs(p) < 1)
print('Filter{} je stabilný.'.format('' if is_stable else ' nie'))
```

```
Filter je stabilný.
```



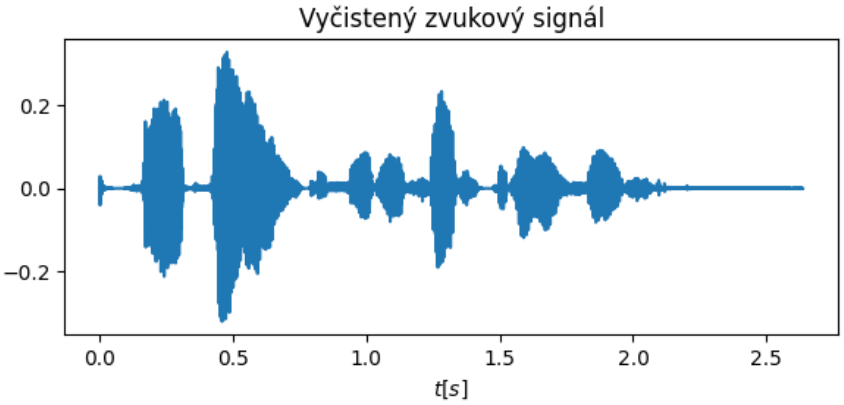
Detail:



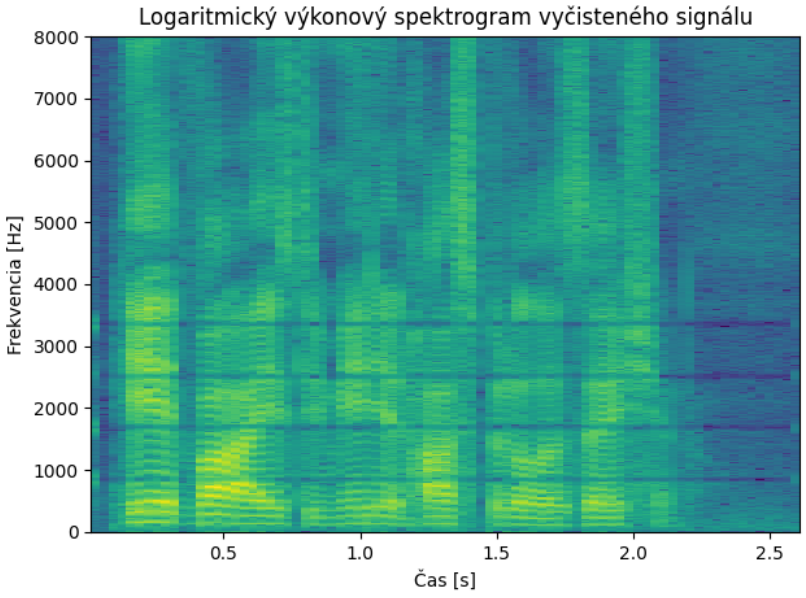


Úloha 9.)

```
filtered_data = lfilter(filter_b, filter_a, s)
sf.write('audio/clean_bandstop.wav',  np.real(filtered_data), fs)
```



Na začiatku vyfiltrovanej nahrávky môžeme vidieť (aj počuť) zvyšok „pípnutia“ – je to kvôli tomu, že filter využíva vždy predošlé vzorky zo signálu (na začiatku 0).



Záver

Na spektrograme vidno, že došlo k eliminácii rušivých elementov. Vyčistená nahrávka je v dobrej kvalite aj napriek zvolenej šírke záverného pásma, ktoré čiastočne odstránilo aj prirodzenú frekvenciu hlasu. Na začiatku vyfiltrovanej nahrávky môžeme vidieť (aj počuť) zvyšné „pípnutia“ – je to kvôli tomu, že filter využíva vždy predošlé vzorky zo signálu (na začiatku 0).

Zdroje

https://www.fit.vutbr.cz/~izmolikova/ISS/project/

https://www.fit.vutbr.cz/study/courses/ISS/public/proj_studijni_etapa/3_sound/3_sound.pdf

https://towardsdatascience.com/fast-fourier-transform-937926e591cb

https://docs.scipy.org/doc/scipy/reference/generated/

https://numpy.org/doc/stable/

https://www.youtube.com/watch?v=s2K1JfNR7Sc&t=349s&ab_channel=SteveBrunton

https://www.youtube.com/watch?v=h7apO7q16V0&ab_channel=Reducible