Vysoké učení technické v Brně
Fakulta Informačních technologií

Signály a systémy

Projekt

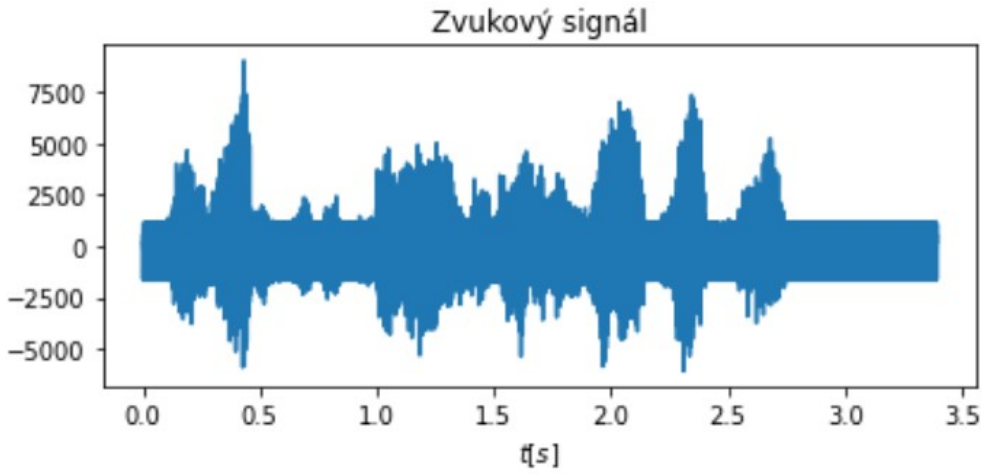
Kristián Kováč (xkovac61)
xkovac61@stud.fit.vutbr.cz

07.01.2022

4.1 Základy

Signál sme zo súboru načítali pomocou knižnice *wawfile.Z* načítaného signálu sme zistili nasledujúce údaje:

- Vzorkovacia frekvencia: 16kHz
- Celkový počet vzorkov: 54272
- DĺŽka nahrávky: 3,392s
- Minimálna hodnota signálu: -6063
- Maximálna hodnota signálu: 9041



4.2 Predspracovanie a rámce

Signál sme ustrednili odčítaním stredej hodnoty signálu do každej vzorky.

```
data = np.mean(data)
```

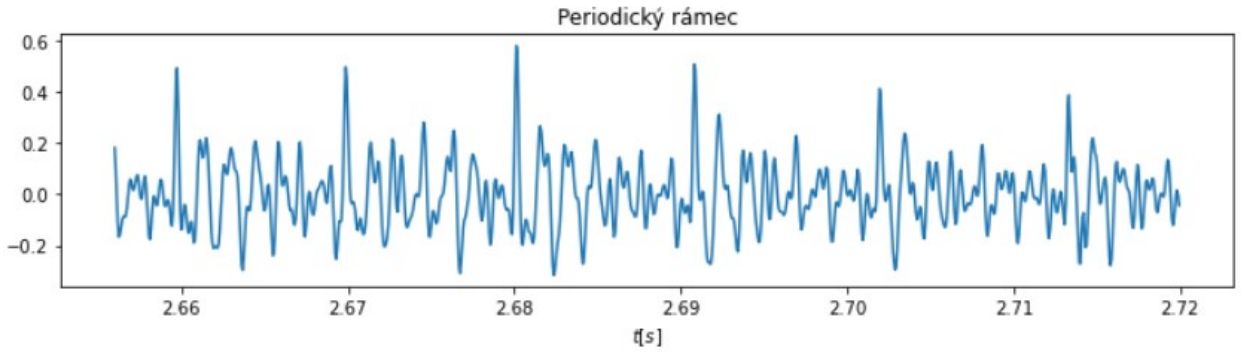
Signál sme normalizovali do dynamického rozsahu -1 a 1 vydelením všetkých vzorkov maximom absolútnej hodnoty signálu.

```
data = data / max(abs(data))
```

Signál sme rozdelili na rámce, ktoré sme uložili do matice.

```
matrix = []
for i in range(0, len(data), 1024-512):
 d = data[i:i+1024]
 matrix.append(np.full((1024), np.pad(d, (0, 1024 - d.size), 'constant')))
```

Vybrali sme nasledujúci rámec s periodickým charakterom:



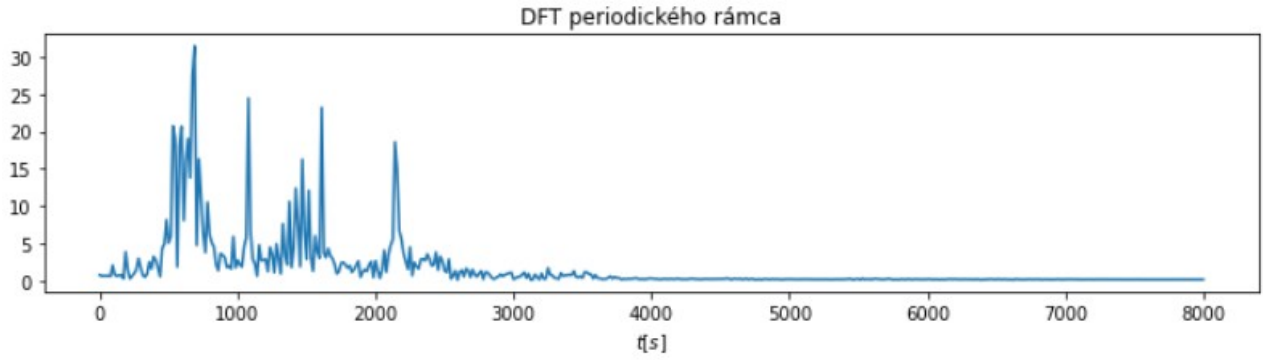
4.3 DFT

Implementovali sme vlastnú funkciu na výpočet DFT.

```
# incializácia matice bází
def DFT_matrix(N):
   i, j = np.meshgrid(np.arange(N), np.arange(N))
   omega = np.exp(-2 * np.pi * 1j / N )
   W = np.power( omega, i * j )
   return W


# výpočet DFT signálu
def DFT(signal, fs):
   N = signal.size
   W = DFT_matrix(N)
   # Nasobenie matice bazi s vektorom signalu
   G = W.dot(signal)
   f = np.arange(G.size) / N * fs
   return f, G
```

Výsledná DFT pre vybraný periodický signál:

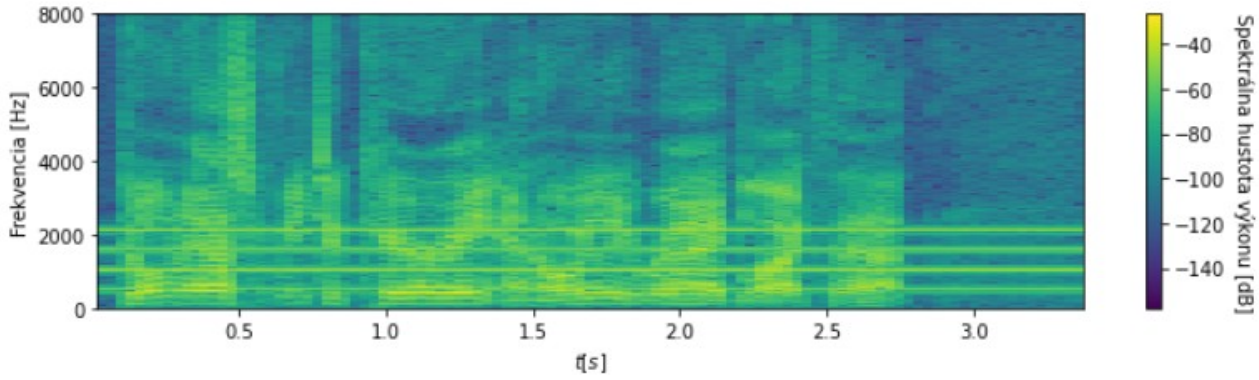


4.4 Spektogram

Na výpočet spektogramu sme použili funkciu *spectogram* z knižnice *scipy.signal.*Na výsledok sme aplikovali vzorec zo zadania.

```
f, time, sgr = spectrogram(data, fs, nperseg=1024, noverlap=512)
result = 10 * np.log10(sgr+1e-20)
```

Výsledný spektogram:



4.5 Určenie rušivých frekvencií

Zo spektogramu sme jasne určile nasledujúce rušivé frekvencie:

- f_1 = 535Hz
- f_2 = 1070Hz = 2 * f_1
- f_3 = 1605Hz = 3 * f_2
- f_4 = 2140Hz = 4 * f_3

Keďže všetky frekvencie sú násobkami f_1, cosinusovky sú harmonicky vzťažené.

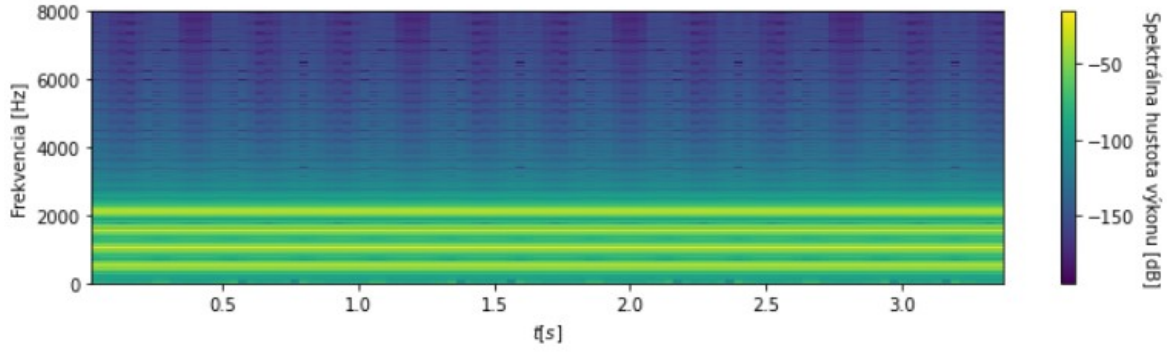
4.6 Generovanie signálu

Signál zložený zo 4 cosinusoviek s frekvenciami f_{1-4} sme vygenerovali nasledovne:

```
cosSignal = np.cos(t * np.pi * 2 * F[0])
cosSignal += np.cos(t * np.pi * 2 * F[1])
cosSignal += np.cos(t * np.pi * 2 * F[2])
cosSignal += np.cos(t * np.pi * 2 * F[3])
```

Výsledný signál sme uložili do súboru *4cos.wav*

Spektogram vygenerovaného signálu:



4.7 Čistiaci filter

Pre návrh filtra sme zo zadania sme vybrali metódu 3. Navrhli sme teda 4 pásmové zádrže pomocou vstavných funkcií *buttord* a *butter* nasledovne:

```
filter_nom_denom = []

for idx, freq in enumerate(F):
  f_ord, wn = buttord([(F[idx] - 50), (F[idx] + 50)], [(F[idx] - 15),
                    (F[idx] + 15)], 3, 40, fs=fs)
  filter_nom_denom.append(butter(f_ord, wn, 'bandstop', fs=fs))
```

Koeficienty (hodnoty čitateľa a menovateľa):

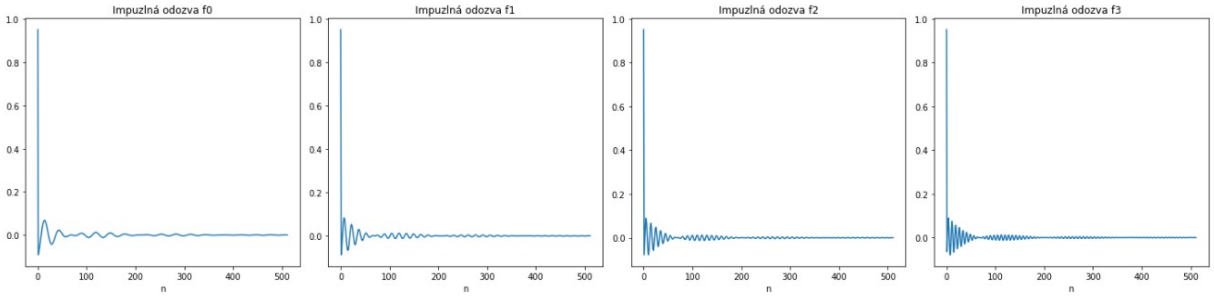
```
f0: [[ 0.95189292 -7.44783707 25.66019125 -50.8401951 63.35189954 -50.8401951
25.66019125 -7.44783707 0.95189292]
[ 1. -7.72780872 26.29717413 -51.46224151 63.34073227 -50.20909585
25.03206135 -7.17691825 0.90610013]]


f1: [[ 0.95095653 -6.94602188 22.82962474 -43.9995893 54.33093026 -43.9995893
22.82962474 -6.94602188 0.95095653]
[ 1. -7.21242986 23.4076437 -44.54827989 54.32050656 -43.44211536
22.25962422 -6.68839726 0.90431833]]


f2: [[ 0.95059571 -6.14372262 18.69251723 -34.47038317 41.96270736 -
34.47038317 18.69251723 -6.14372262 0.95059571]
[ 1. -6.38116764 19.16923229 -34.90299758 41.95389627 -34.0298822
18.22217246 -5.91416415 0.9036322 ]]


f3: [[ 0.95039741 -5.07282685 13.95533111 -24.25122341 29.02318191 -
24.25122341 13.95533111 -5.07282685 0.95039741]
[ 1. -5.26970452 14.31249191 -24.55632897 29.01634202 -23.93955223
13.60254979 -4.88251479 0.90325523]]
```

Impulzné odozvy filtrov:

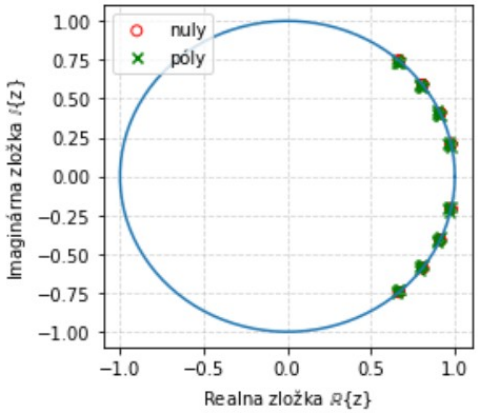


4.8 Nulové body a póly

Nulové body a póly sme vypočítali pomocou funkcie *tf2zpk* z kližnice *scipy* nasledovne:

```
z_arr = []
p_arr = []
for idx, nom_denom in enumerate(filter_nom_denom):
   z, p, k = tf2zpk(nom_denom[0], nom_denom[1])
   z_arr.append(z)
   p_arr.append(p)
```

Znázornenie nulových bodov a pólov:

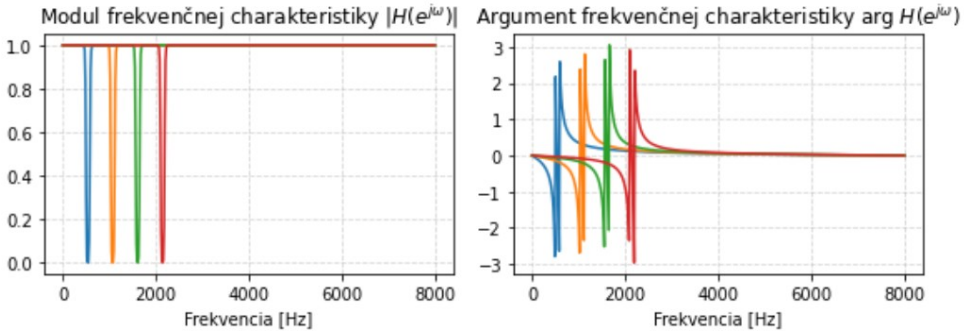


4.9 Frekvenčná charakteristika

Frekvenčnú charakteristiku filtrov sme vypočítali nasledovne:

```
for idx, nom_denom in enumerate(filter_nom_denom):
   w, H = freqz(nom_denom[0], nom_denom[1])
   ax[0].plot(w / 2 / np.pi * fs, np.abs(H))
   ax[1].plot(w / 2 / np.pi * fs, np.angle(H))
```

Zobrazenie frekvenčných charekteristík:



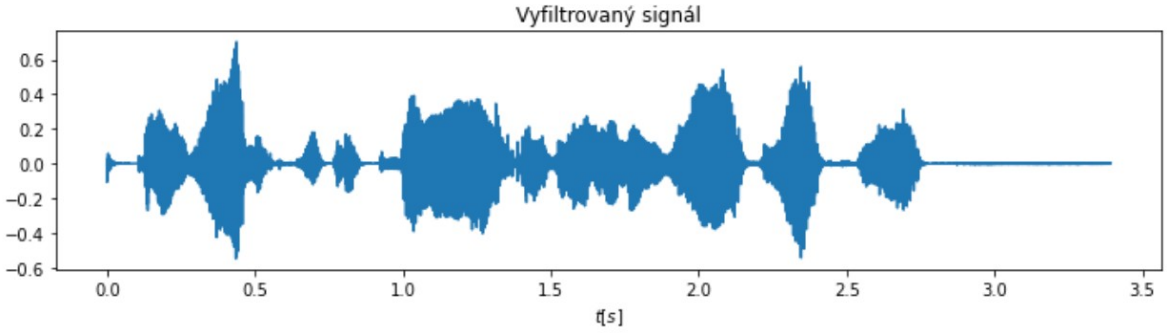
4.10 Filtrácia

Signál sme vyfiltrovali pomocou navrhnutých filtrov výsledný signál uložili do súboru *clean_bandstop.wav*

```
filtered_signal = data

for idx, fil in enumerate(filter_nom_denom):
  filtered_signal = lfilter(filter_nom_denom[idx][0], filter_nom_denom[idx]
                            [1], filtered_signal)
```

Zobrazenie výsledného signálu:



Spektogram výsledného signálu: