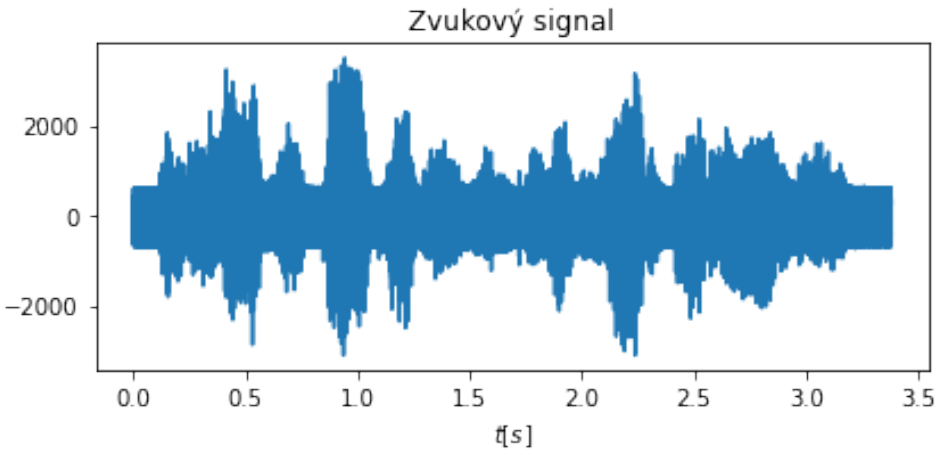
# 1 Základy

Projekt byl řešen v programovacím jazyce `Python` s použitím `Jupyter Notebook`, byly využity knihovny `Numpy`, `Scipy` a `Matplotlib`. Signál byl načten pomocí funkce `scipy.wavfile.read` z knihovny jako `Numpy ndarray`. O signálu byly pomocí knihovních funkcí zjištěny následující údaje:

```
Samplerate: 16000
Number of samples: 54068
Length: 3.37925s
Minimal sample: -3101, Maximal sample: 3496
```




Obrázek 1: Zobrazení zvukového signálu

# 2 Předzpracování a rámce

Signál byl ustředněn a normalizován následovně:

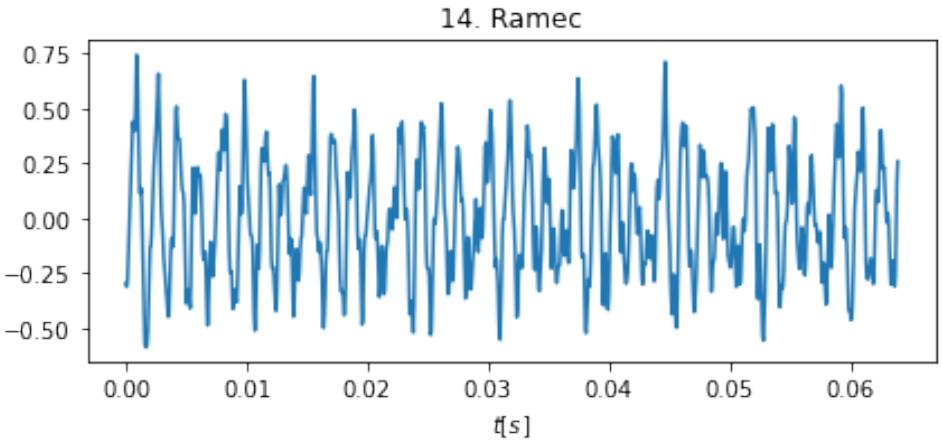
```
s = s - np.mean(s) #ustřednění
s = s / max(s.min(), s.max(), key=abs) #normalizace
```

Následně byl signál rozdělen do rámců o délce 1024 vzorků a uložen do dvojrozměrného pole, přičemž jednotlivé rámce byly uloženy do sloupců. Rozdělení do rámců proběhlo pomocí cyklu, ve kterém bylo do každého sloupce uloženo 1024 vzorků z aktuální polohy, která byla posunuta vždy o 512 vzorků, dokuď nebylo rozděleno celé pole.

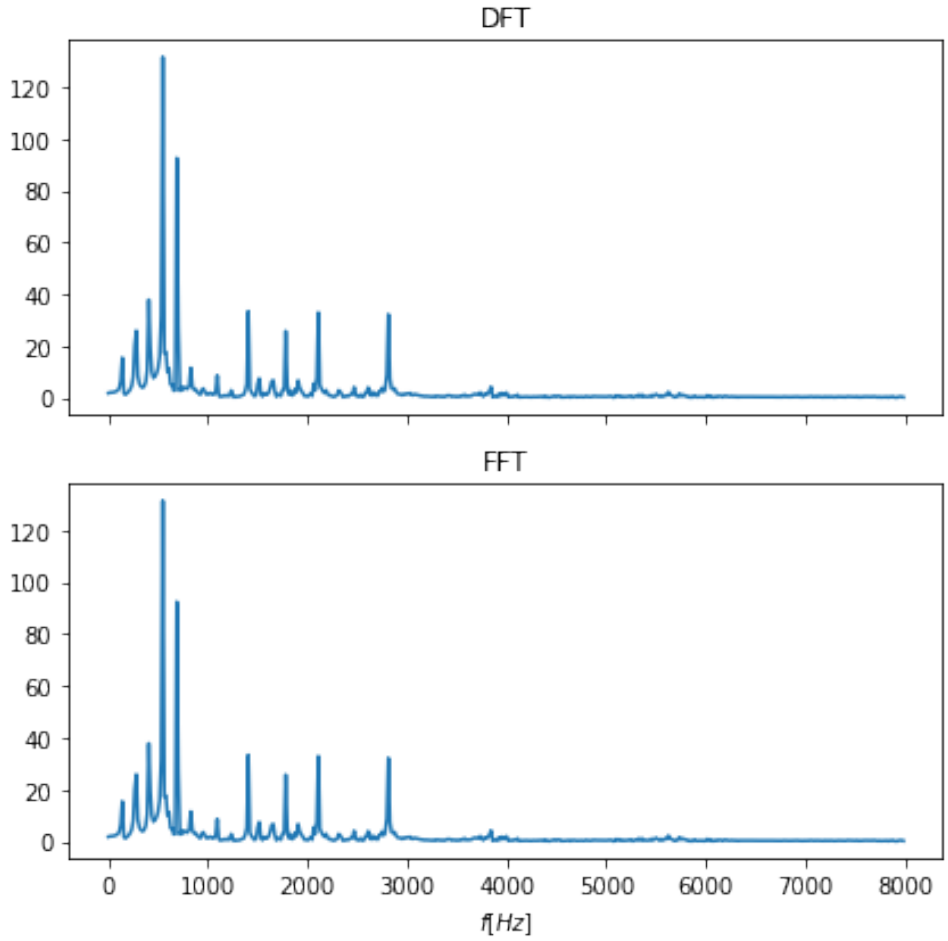
Pro zobrazení byl vybrán 14. rámec, ve kterém je vyslovována samohláska „e“.

# 3 DFT

Vlastní funkce pro diskrétní fourierovu transformaci byla vytvořena pomocí funkce `scipy.linalg.dft`, která vytvoří matici pro dft o zadané velikosti (v tomto případě 1024) a následného maticového násobení této matice s rámcem signálu. Je zobrazena má implementace dft zároveň s funkcí `numpy.fft.fft` na 14. rámci signálu, oba grafy jsou vizuálně shodné.




Obrázek 2: Zobrazení 14. rámce signálu

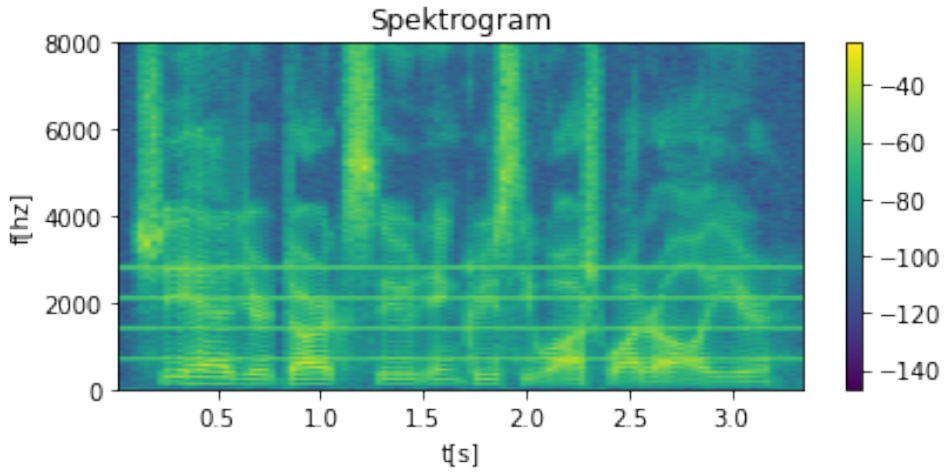



Obrázek 3: Porovnání mé implementace dft a knihovní fft

## 4 Spektrogram

Spektrogram s velikostí okna 1024 a překrytím 512 byl vytvořen pomocí funkce

```
plt.specgram(s, window=None, Fs=fs, NFFT=1024, noverlap=512)
```




Obrázek 4: Spektrogram signálu

## 5 Určení rušivých frekvencí

Rušivé frekvence byly určeny pomocí funkce `scipy.signal.find_peaks` použité na první rámec signálu po fourierově transformaci. Funkce vrátila následující hodnoty:

```
peaks = [ 703.125, 1406.25, 2109.375, 2812.5 ]
```

Ověření, jestli jsou frekvence harmonicky vztažené bylo provedeno vydělením všech frekvencí tou nejnižší, výsledek byla celá čísla, frekvence jsou tedy násobky nejnižší.

## 6 Generování signálu

Signál byl vygenerovám pomocí sečtení 4 cosinusovek generovaných následovně:

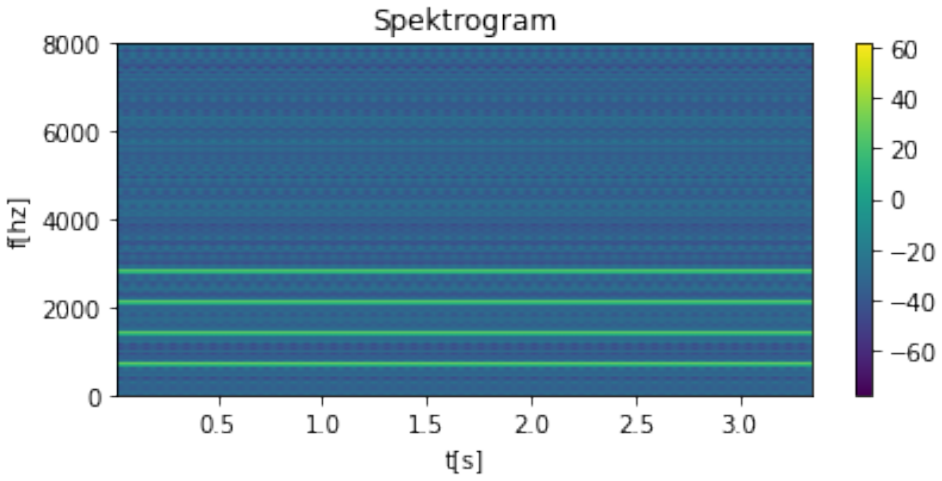
```
samples = np.arange(s.size) / fs
for i in range(4):
    cos += np.cos(2 * np.pi * peaks[i] * samples)
```

Signál byl následovně uložen do souboru pomocí funkce `scipy.io.wavfile.write`. Poslechem a pomocí spektrogramu bylo ověřeno, že byly rušivé frekvence určeny správně.

## 7 Čistící filtr

Byly vytvořeny 4 bandstop filtry f0 až f3 (od nejnížší frekvence k nejvyšší) s šíří závěrného pásma 30 Hz a rozptylem přechodu do propustného pásma 50 Hz na každé straně. K návrhu filtrů byla použita funkce `scipy.signal.buttord`, která vrátila správný řád filtru (4) a kritické frekvence, které byly předány funkci `scipy.signal.butter`, která vrátila samotné koeficienty filtrů:

```
f0: a[  1.    -7.6   25.57 -49.69  61.02 -48.47  24.33  -7.05   0.91]
    b[  0.95  -7.32  24.94 -49.09  61.03 -49.09  24.94  -7.32   0.95]
f1: a[  1.    -6.72  20.86 -38.68  46.76 -37.71  19.83  -6.23   0.9 ]
    b[  0.95  -6.48  20.34 -38.2   46.77 -38.2   20.34  -6.48   0.95]
f2: a[  1.    -5.34  14.59 -25.13  29.73 -24.5   13.87  -4.95   0.9 ]
    b[  0.95  -5.14  14.23 -24.82  29.73 -24.82  14.23  -5.14   0.95]
f3: a[  1.    -3.55   8.63 -13.18  15.54 -12.85   8.2   -3.29   0.9 ]
    b[  0.95  -3.42   8.41 -13.02  15.54 -13.02   8.41  -3.42   0.95]
```




Obrázek 5: Spektrogram signálu s rušivými cosinusovkami

Impulzní odezvy byly vytvořeny aplikováním filtrů pomocí funkce `scipy.signal.lfilter` na signál s jedničkovým impulzem na začátku o délce 75 vzorků.

## 8 Nulové body a póly

Nulové body a póly filtrů byly taktéž vypočítány pomocí funkce `scipy.signal.butter`, ale s parametrem `output='zpk'`.

## 9 Frekvenční charakteristika

Frekvenční charakteristika byla vypočtena pomocí funkce `scipy.signal.freqz`.

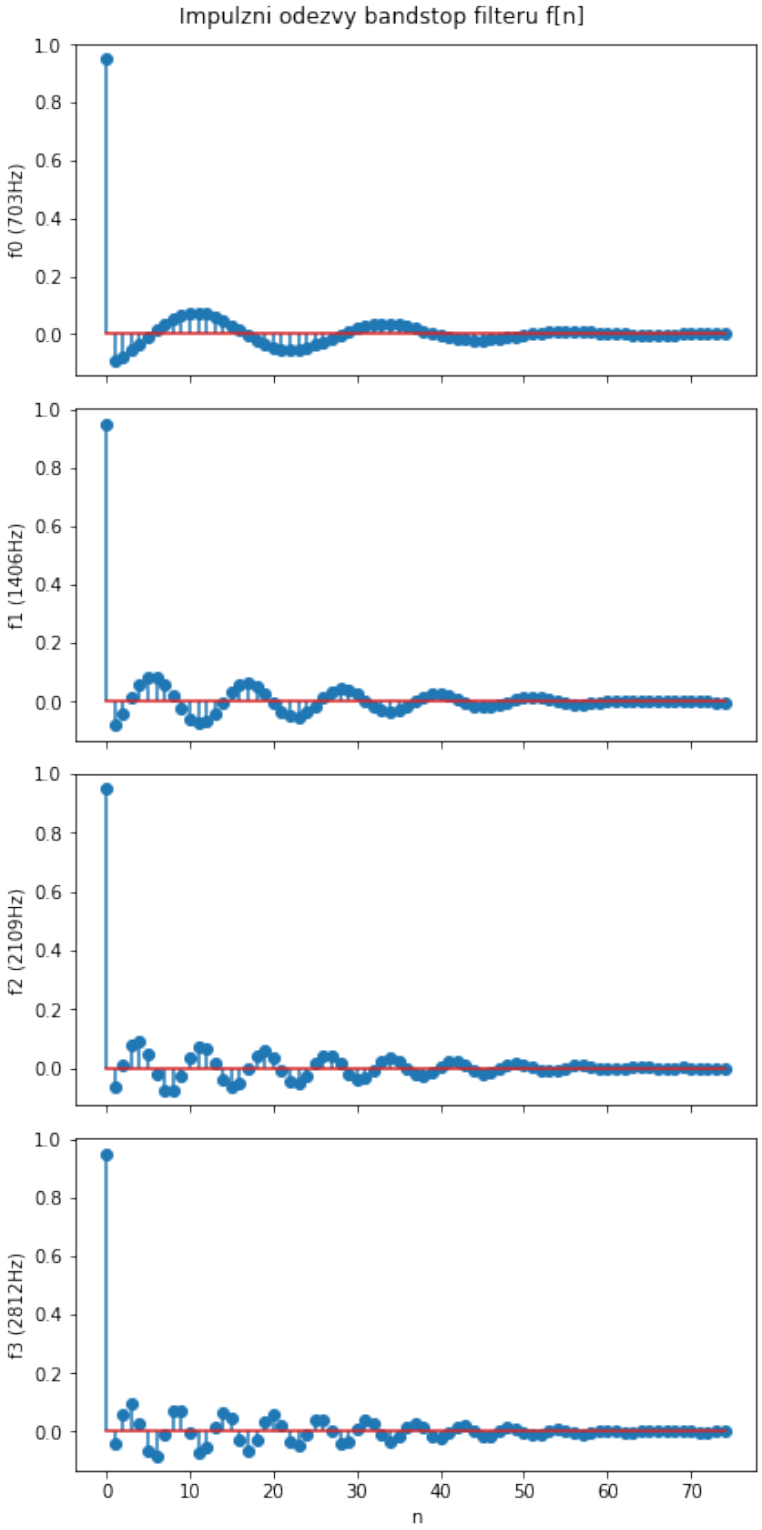
## 10 Filtrace

Filtrace byla provedena použitím funkce `scipy.signal.lfilter` na signál pro každý filtr. Vypsáním největšího a nejmenšího vzorku bylo zjištěno, že je signál ve správném rozsahu $[-1, 1]$. Výsledný signál byl uložen do souboru pomocí `scipy.io.wavfile`.

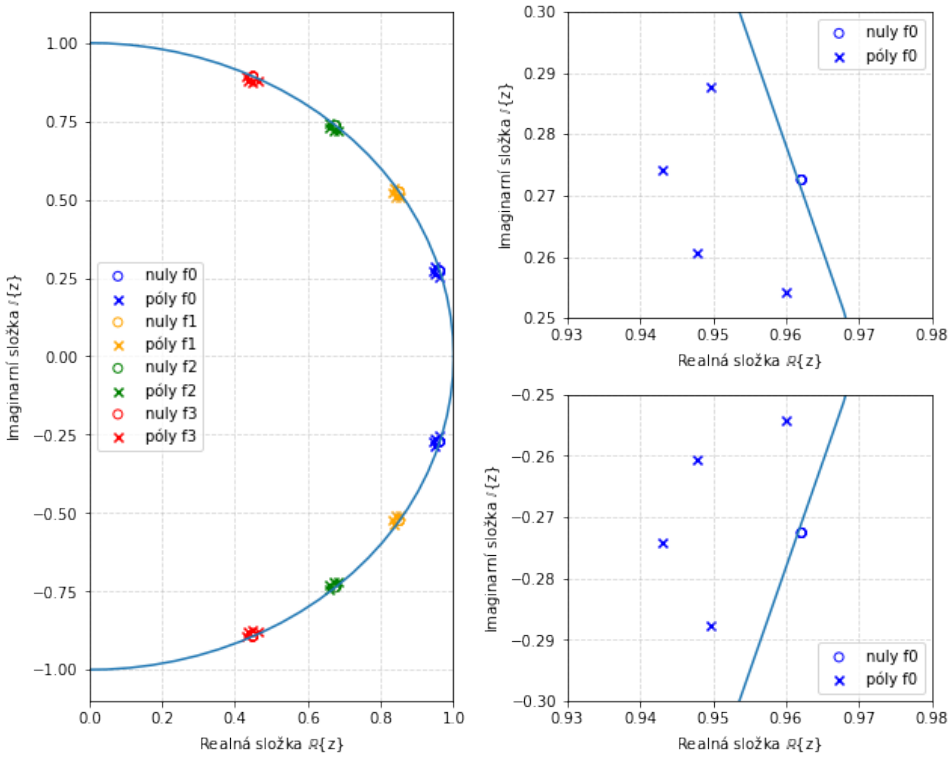
Poslechem bylo ověřeno, že rušivé frekvence byly odfiltrovány. Je ale slyšet, že se tyto frekvence z nahrávky ztratily a nahrávka je tedy mírně zkreslená.

## 11 Použité zdroje

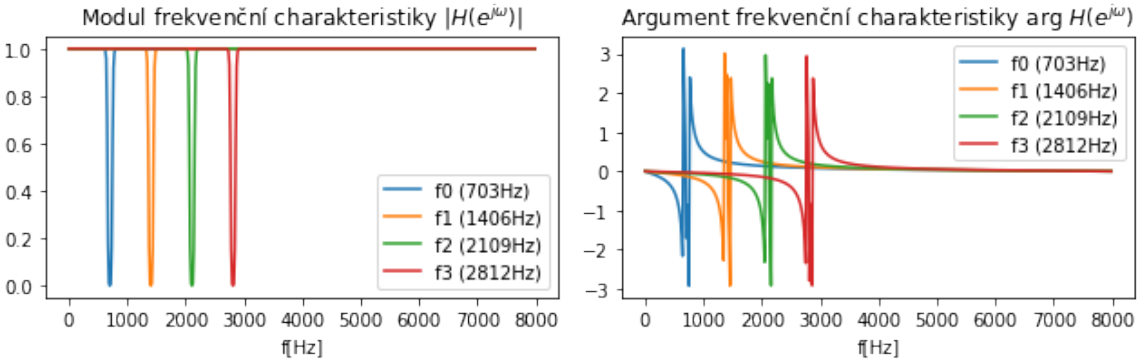
- Dokumentace numpy: `https://numpy.org/doc/1.21/`
- Dokumentace scipy: `https://docs.scipy.org/doc/scipy-1.7.0/reference/index.html`
- Dokumentace matplotlib: `https://matplotlib.org/stable/index.html`




Obrázek 6: Impulzní odezvy navržených filtrů




Obrázek 7: Nulové body a póly filtrů v komplexní rovině




Obrázek 8: Frekvenční charakteristika filtrů