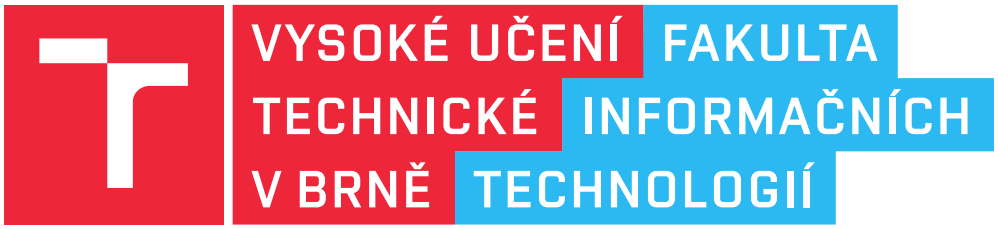


ISS Protokol

Peter Polóni

Login: xpolon03

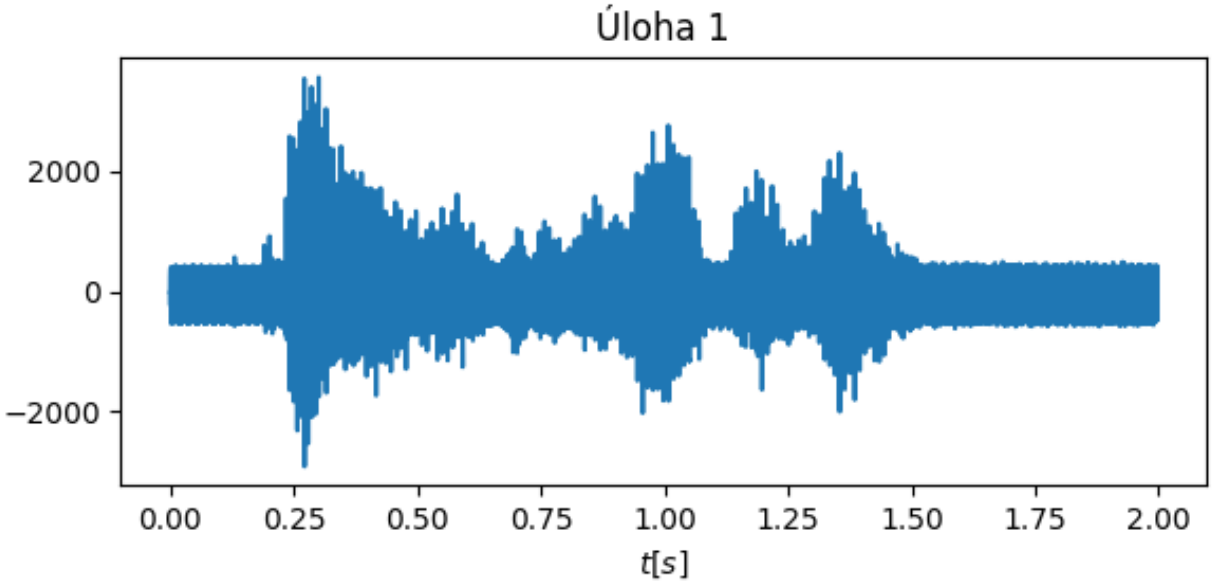
Obsah

1 Úloha 1

Vstupný signál som načítal pomocou funkcie *wavfile.read()*. Táto funkcia mi vrátila vzorkovaciu frekvenciu načítaného signálu a pole vzorkov, ktoré tvorí daný signál. Pomocou týchto údajov som vedel zistiť odpovede na požadované otázky. Dĺžku signálu vo vzorkoch reprezentovala veľkosť poľa vzorkov, bolo to **31949** vzorkov. Dĺžku signálu v sekundách som zistil jednoduchým výpočtom.

$$t = \frac{N_v}{F_s}$$

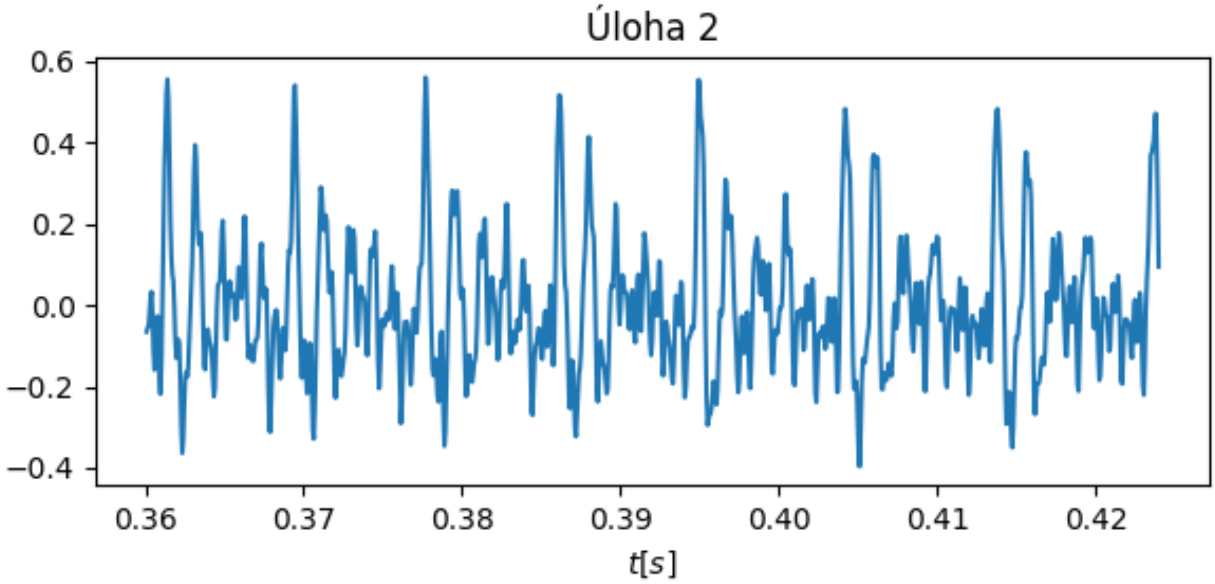
V tejto rovnici t predstavuje čas v sekundách, N_v je počet vzorkov a F_s je vzorkovacia frekvencia. Vypočítal som, že dĺžka signálu v sekundách je **1.9968125** sekundy. Minimálnu a maximálnu hodnotu signálu som zistil pomocou funkcií *.min()* a *.max()*, najmenšia hodnota bola **-2918**, najväčšia hodnota bola **3567**.



Obr. 1: Môj načítaný signál

2 Úloha 2

Ustrednenie som vykonal pomocou funkcie *np.mean()*, ktorá mi zistila strednú hodnotu signálu, túto hodnotu som následne odčítal od hodnôt signálu. Normalizáciu som vykonal predelením hodnôt signálu najväčšou absolútnou hodnotou signálu. Signál som rozdelil na rámce s dĺžkou 1024 vzorkov s prekrytím 512 vzorkov. To znamená že prvý rámec obsahuje hodnoty od 0 do 1023, druhý rámec obsahuje hodnoty od 511 do 1535 a tak pokračujú ďalej, celkovo mám vytvorených 61 rámcov. "Pekný rámec"som hľadal ručne, bol to rámec, v ktorom sa vyslovuje samohláska.



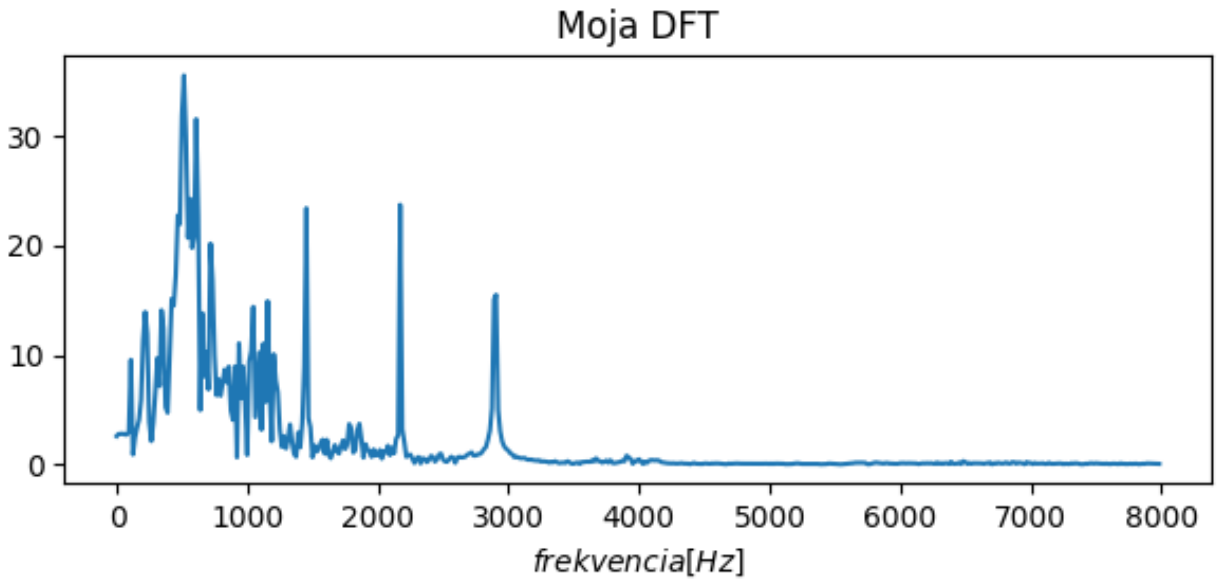
Obr. 2: Ustrednený, normalizovaný periodický rámec

3 Úloha 3

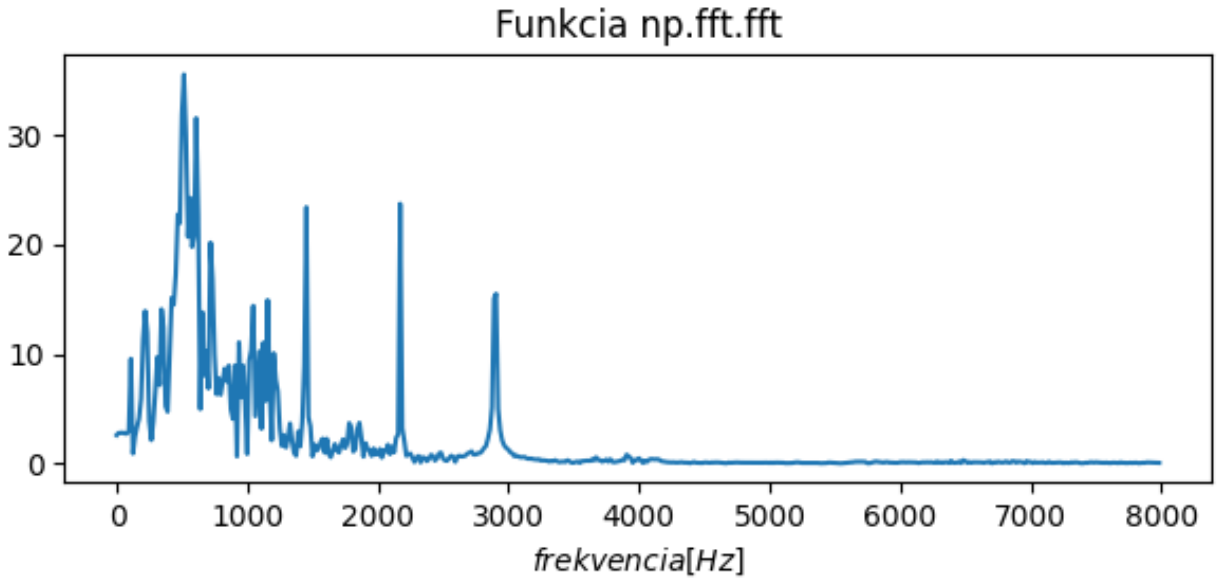
Ako rámec na ktorom sa vykonala DFT som použil rovnaký rámec ako v minulej úlohe. Generovanie hodnôt pre násobenie robím pomocou funkcie *np.exp()* v nej používam vzorec:

$$\frac{-1j \times 2 \times \pi \times k \times n}{N}$$

Hodnoty použijem vo funkcií *np.exp()*, ktorá vykoná umocnenie. Následne hodnoty vynásobím maticou, v ktorej mám uložený signál. Pre porovnanie a kontrolu som použil aj funkciu *np.fft.fft()* aj *np.allclose()*, ktorej výtupom bola hondota true teda sa výsledky funkcií rovnajú.



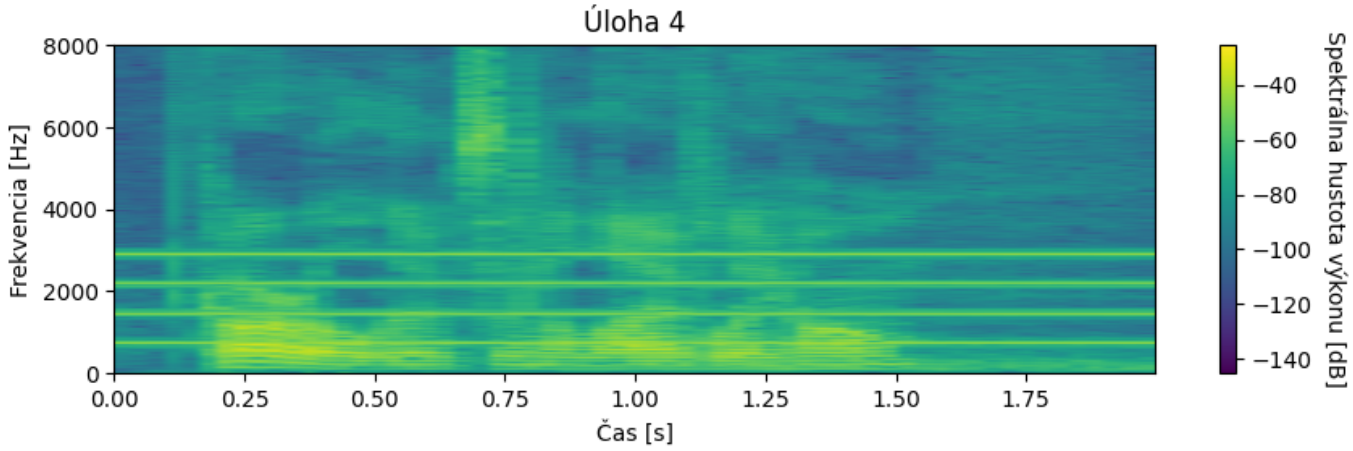
Obr. 3: DFT vykonaná mojou funkciou



Obr. 4: DFT vykonaná funkciou np.fft.fft

4 Úloha 4

Spektrogram som implementoval pomocou knižnicovej funkcie *np.spectrogram()*.Táto funkcia dokázala zabezpčiť dĺžku okna 1024 vzorkov s prekrytím 512 vzorkov. Taktiež vykonala nad signálom DFT a vrátila hodnoty umocnené a v absolútnej hodnote. Hodnoty, ktoré mi vrátila funkcia spectrogram som prenásobil $10 \times \log_{10} X$, kde X sú hodnoty z funkcie *np.spectrogram()*.



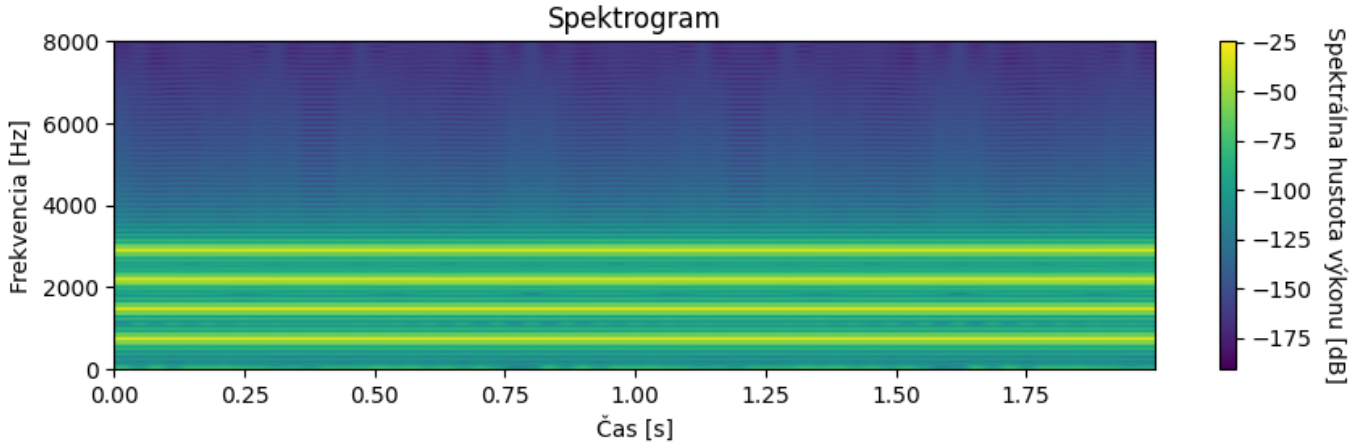
Obr. 5: Spektrogram

5 Úloha 5

Frekvencie f_1, f_2, f_3, f_4 som odčítal priamo zo spektrogramu. Našiel som, že sú to hodnotnoty 725, 1460, 2175, 2900 čo sú približne násobky 725, teda cosinusovky sú harmonicky vzťažené.

6 Úloha 6

Pre vytvorenie cosinusovky som využil funkciu *np.cos()*, musel som však vytvoriť pole časových úsekov, toto pole predstavuje hodnotu cosinusovky v danom čase, teda v čase nula má hodnotu $\frac{0}{16000}$ potom má hodnotu $\frac{1}{16000}$, prechádzam takto každý vzorok až po hodnotu 31948, teda celkovo 31949 vzorkov. Pomocou toho som dokázal vytvoriť štyri cosinusovky, na frekvenciách f_1, f_2, f_3, f_4. Aby som vytvoril výsledný signál tieto cosinusovky som sčítal. Výsledný signál, ktorý som zapisoval som normalizoval a ustrednil, pred zápisom som musel jeho hodnotu upraviť, tak, aby z neho bol 16bitový integer. Odposluchom výstupného signálu som overil, že sa jedná o rušivý zvuk z nahrávky, potvrdil to aj výsledný spektrogram.



Obr. 6: Spektrogram pre vygenerovanú cosinusovku

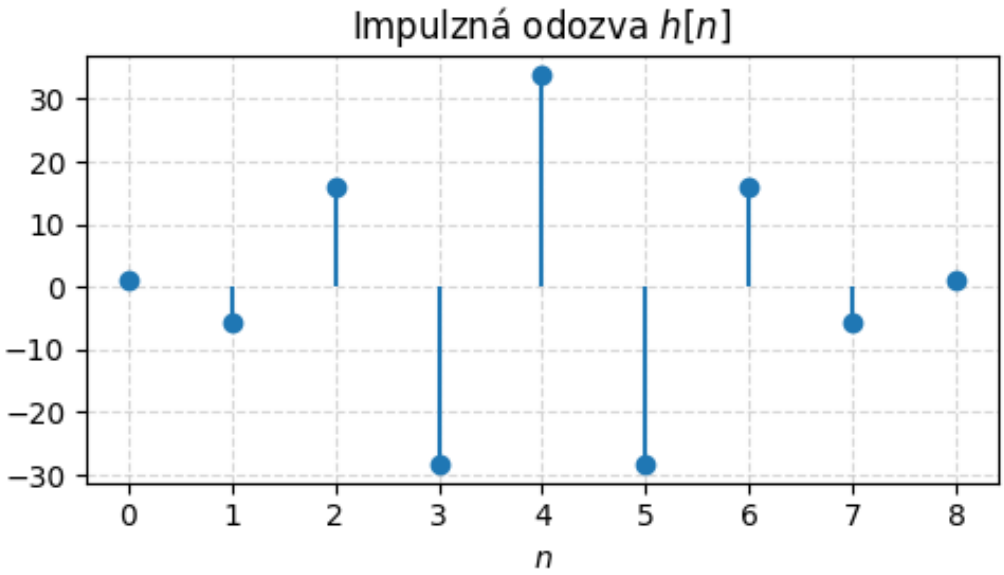
7 Úloha 7

Ako typ filtru som si vybral filter v z–rovine, najprv som vypočítal štyri nulové body podľa vzorcov

$$\omega_k = 2\pi \frac{f_k}{F_s}$$

$$n_k = e^j \omega_k$$

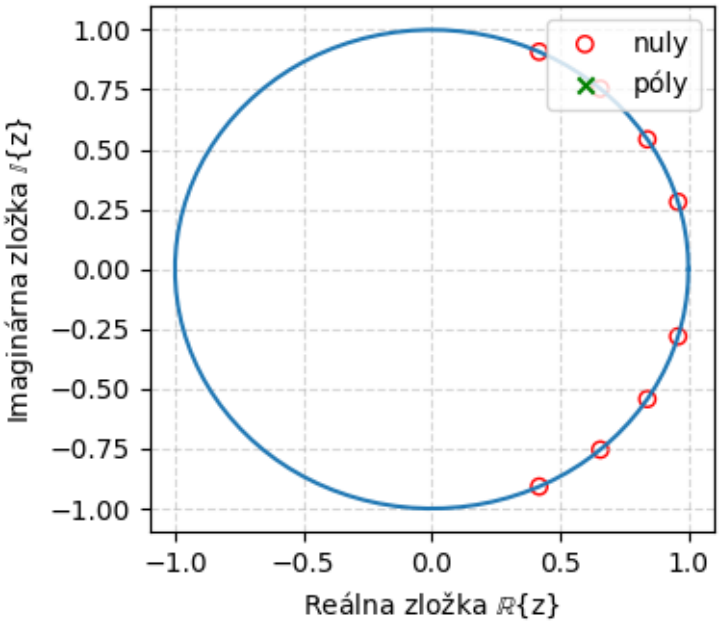
Ďalej som pokračoval podľa zadania a dostal som FIR–filter s deviatimi koeficientami. Výsledný filter síce vyfiltroval rušivé frekvencie, no veľmi zoslabil hlasitosť reči v nahrávke, spôsobilo to to, že filter zosilnil vysoké frekvencie, a po normalizovaní je reč slabšia. Koeficienty môjho filtra sú **1, -5.75075846, 16.0683402, -28.14951886, 33.68441848, -28.14951886, 16.0683402, -5.75075846, 1**. Impulznú odozvu som zobrazil na deviatich bodoch, pomocou funkcie *lfilter()*.



Obr. 7: Impulzná odozva filtru

8 Úloha 8

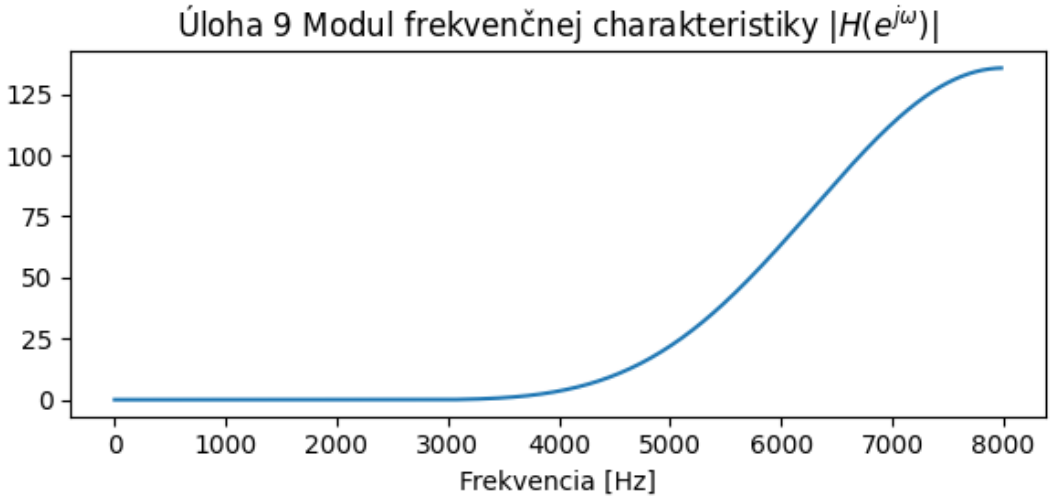
Pomocou funkcie *tf2zpk()* som zistil nuly a póly môjho filtru, nuly predstavujú nulové body, ktoré som pre filter počítal, póly filter nemá.



Obr. 8: Nuly a póly

9 Úloha 9

Frekvenčnú charakteristiku som vypočítal funkciou *freqz()*.

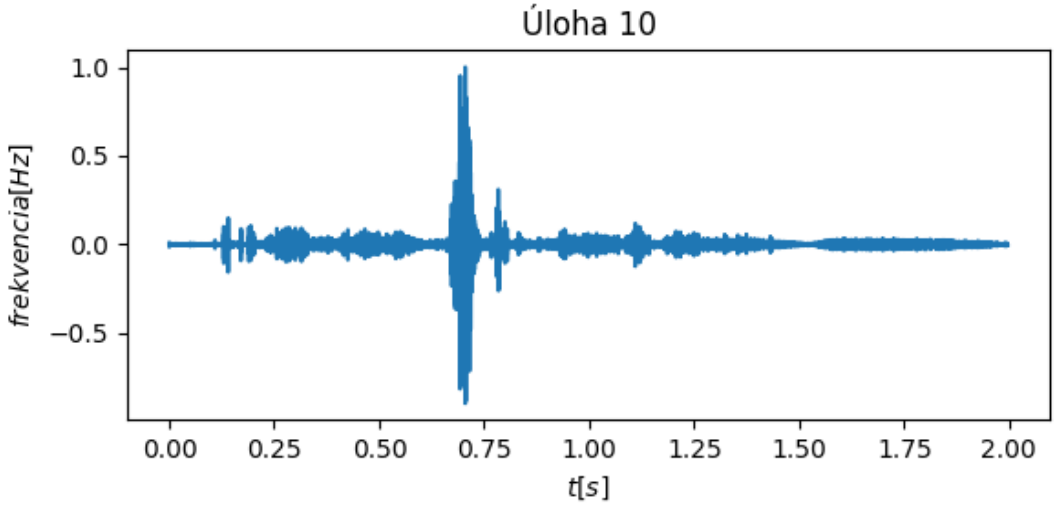


Obr. 9: Frekvenčná charakteristika

10 Úloha 10

Samotnú filtráciu vykonávam funkciou *lfiter()*, vyfiltrovaný signál ustredňujem a normalizujem, aby mal slušný dynamický rozsah. Tiež vyfiltrovaný signál pri zápise prevádzam na 16bitový integer.

Výsledný zvuk je zbavený rušivého pískania, no reč je veľmi tichá oproti pôvodnej nahrávke.



Obr. 10: Vyfiltrovaný signál

11 Zdroje

Väčšinu funkcií som našiel v príkladoch od Katky Žmolíkové, rovnako som funkcie čerpal aj zo zadania projektu a z dokumentácie knižnice numpy informácie o dft som čerpal zo stránky: `https://pythonnumericalmethods.berkeley.edu/notebooks/chapter24.02-Discrete-Fourier-Transform.html`