VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
FAKULTA INFORMAČNÍCH TECHNOLOGIÍ

Signály a systémy

PROJEKT 2021/22

Denis Horil (xhoril01)

31.12.2021

RIEŠENIE

Riešenie je implementované v programovacom jazyku *Python*. Všetky uvedené funkcie sú funkciami knižníc tohto jazyka.

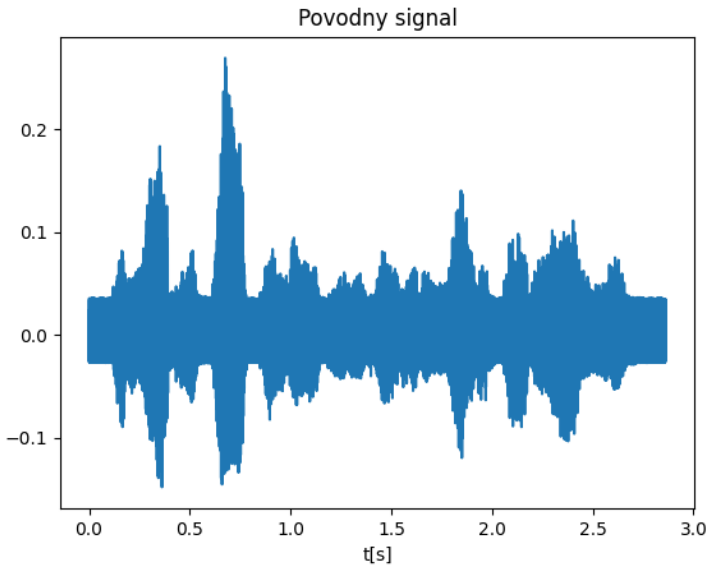
Pri implementácii som čerpal z prednášok a cvičení, z *Python* notebookov zo študijnej opory a z dokumentácie k *Python* knižniciam

Použité knižnice : *numpy, scipy, matplotlib, soundfile*

Všetky výpočty sú v zdrojovom kóde *xhoril01.py* a výsledky sa po spustení uložia do textového súboru *useful_info.txt*

Zadanie 4.1 – Základy

- Dĺžka signálu vo vzorkách je 45773
- Dĺžka signálu v sekundách je 2.8608125 s
- Maximálna hodnota signálu je 0.2692565918
- Minimálna hodnota signálu je -0.1477355957
- Signál som načítal pomocou funkcie *soundfile.read()*

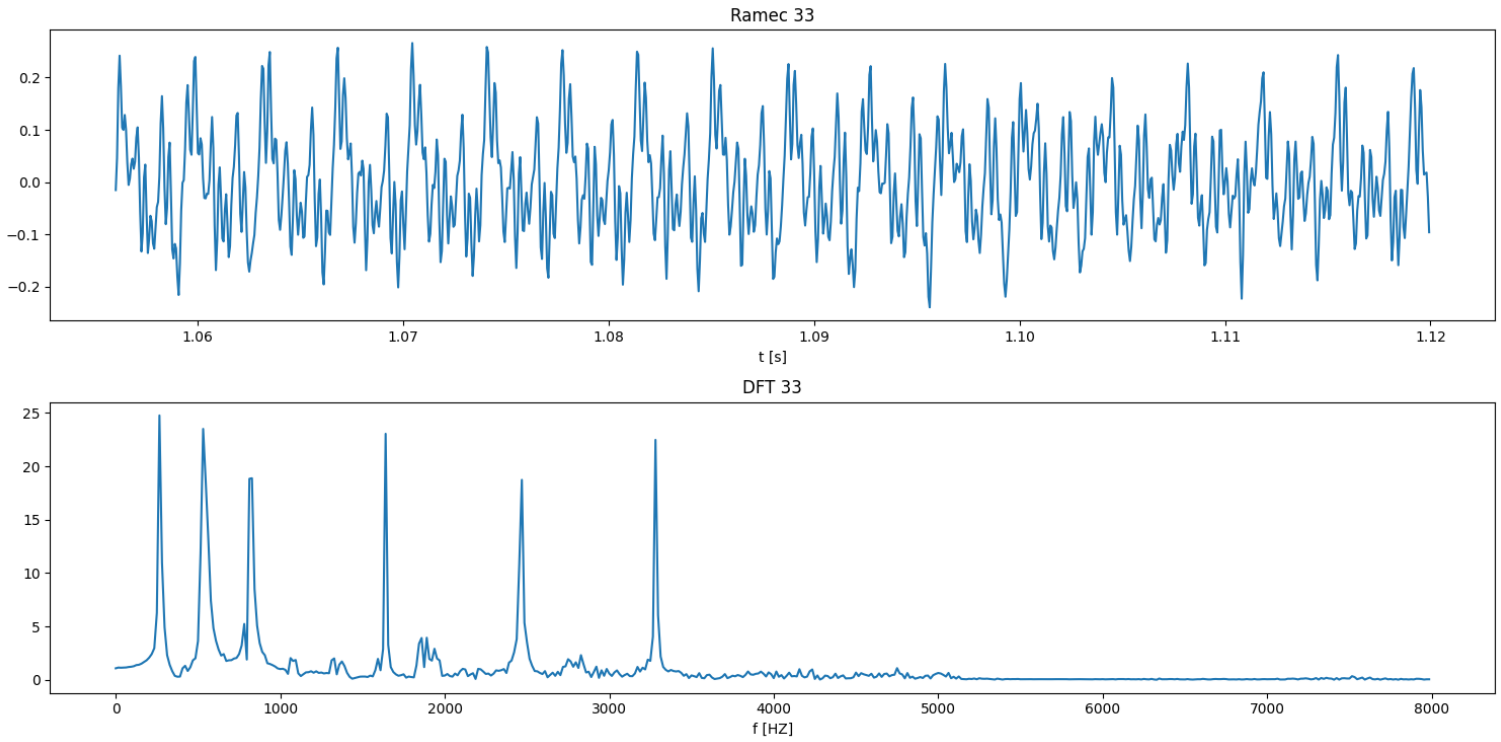


Zadanie 4.2 – Predzpracovanie a rámce

- Ustrednenie signálu – odčítanie strednej hodnoty pôvodného signálu, strednú hodnotu som dostal pomocou funkcie *np.mean()*
- Normalizovanie signálu – vydelenie absolútnou hodnotou maxima signálu, túto hodnotu som dostal pomocou funkcie *np.max()* a *np.abs()*
- Rozdelenie signálu na rámce – s dĺžkou rámca 1024 vzoriek a prekrytím 512 vzoriek vznikne 89 rámcov, posledný rámec treba doplniť nulami, aby bolo možné vytvoriť maticu *(1024 * počet rámcov)*

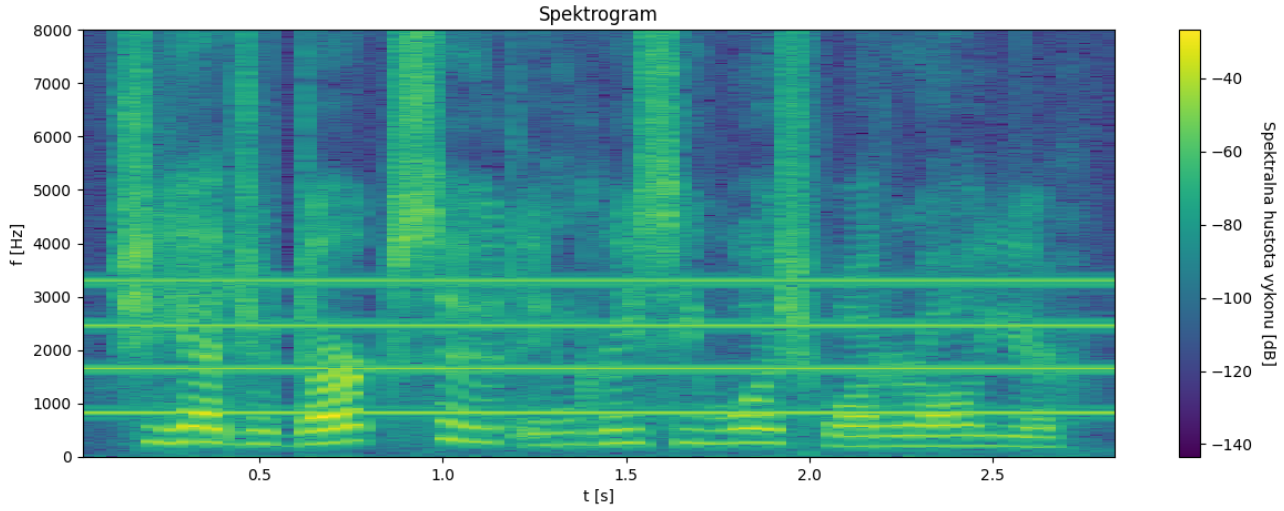
Zadanie 4.3 – DFT

- Pre výpočet DFT som implementoval vlastnú funkciu *custom_DFT(frame)*, kde '*frame*' je vybraný „pekný" znelý rámec – vybral som rámec 33
- Transformáciu som uskutočnil pomocou vzorca pre výpočet DFT prebraný na prednáškach a cvičeniach
- Pre výpočet súčinu bázovej matice a vektora signálu (rámca) som využil funkciu np.dot()
- Pre porovnanie môjho výsledku s knižnicovou implementáciou som využil funkcie *np.fft.fft()* a *np.allclose()*



Zadanie 4.4 – Spektrogram

- Na vygenerovanie spektrogramu som využil funkciu *scipy.signal.spectrogram()* a potom funkciu *matplotlib.pyplot.pcolormesh()*
- Hodnoty koeficientov DFT spektrogramu som následne upravil pomocou vzorca v zadaní
- Aby sa v spektrograme zobrazil signál rozdelený na rámce, využil som voliteľné parametre funkcie *spectrogram()* – *nperseg=1024* (dĺžka okna) a *noverlap=512* (prekrytie)



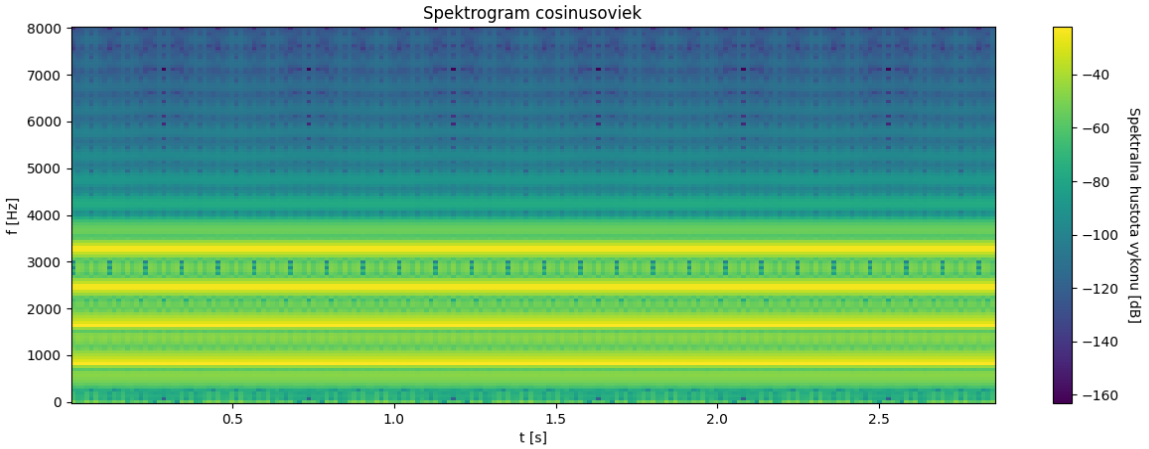
Zadanie 4.5 – Určenie rušivých frekvencií

- Na určenie rušivých frekvencií som si vybral nultý rámec, aby boli rušivé frekvencie ľahšie rozpoznateľné a transformoval pomocou funkcie *custom_DFT(frame)*, na zistenie frekvencií nám stačí len polovica vzorkov , teda 512
- Pomocou funkcie *scipy.signal.find_peaks()* som našiel koeficienty rušivých frekvencií a vytvoril som pole hodnôt týchto frekvencií
- Vypočítal som jednotlivé koeficienty ako podiel frekvencií *f2,f3,f4* a *f1* a možnú odchýlku týchto koeficientov *(16000/1024 = 15,625 Hz)*
- Zistil som závislosť týchto frekvencií, teda či sú násobkami frekvencie $f1 \pm 15{,}625\ Hz$

Zadanie 4.6 – Generovanie signálu

- Pomocou funkcie *np.cos()* som vytvoril 4 kosínusovky a spojil ich dohromady

- Pomocou funkcií *soundfile.write()*, *astype()* a *np.float32()* som vytvoril signál ***‘4cos.wav‘***



Zadanie 4.7 – Čistiace filtre

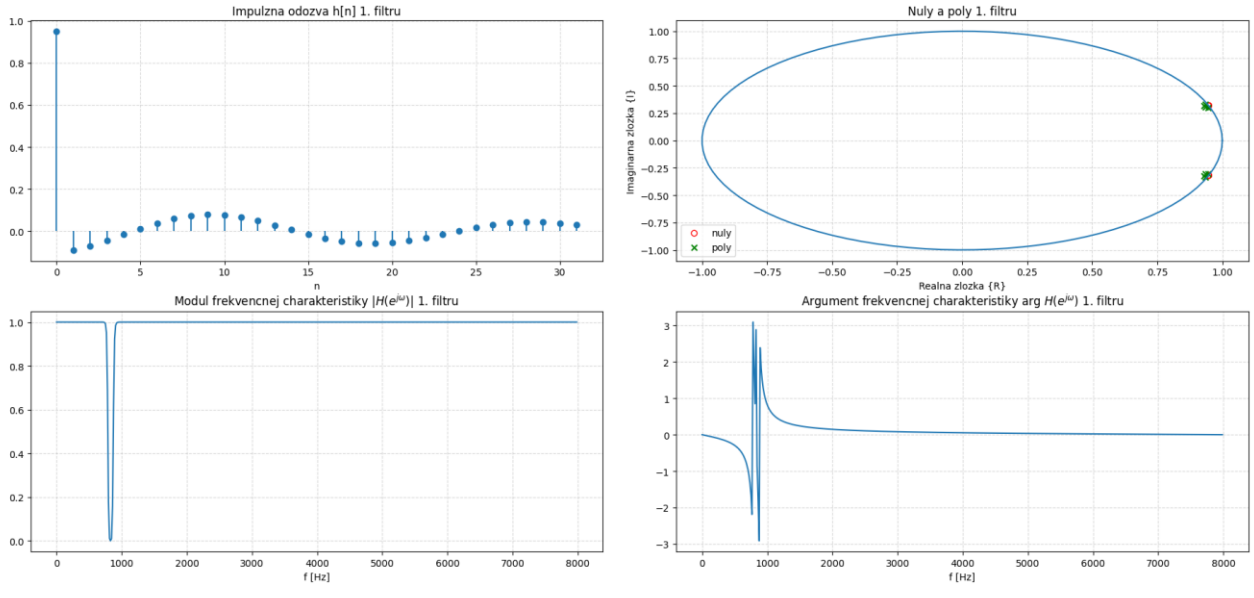
- Pomocou funkcií *scipy.signal.buttord()* a *scipy.signal.butter()* som z rušivých frekvencií vytvoril polynomický čitateľ a menovateľ filtra so šírkou záverného pásma 30Hz, šírkou prechodu 50Hz, zvlnením 3dB a potlačením v závernom pásme -40dB
- Pomocou funkcie *scipy.signal.lfilter()* som vytvoril daný filter a zobrazil jeho impulznú odozvu
- Tento spôsob som aplikoval na všetky rušivé frekvencie

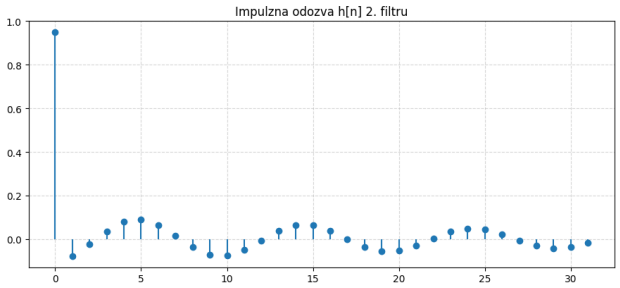
Zadanie 4.8 – Nuly a póly

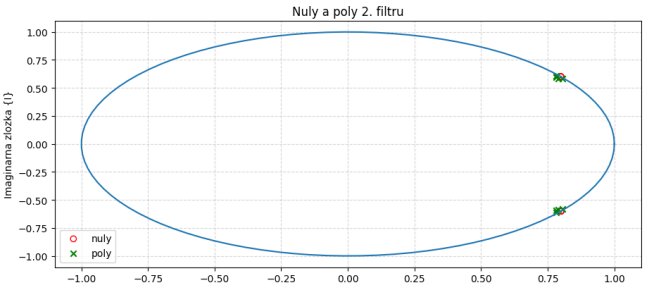
- Nuly a póly pre jednotlivé filtre som získal pomocou funkcie *scipy.signal.tf2zpk()*

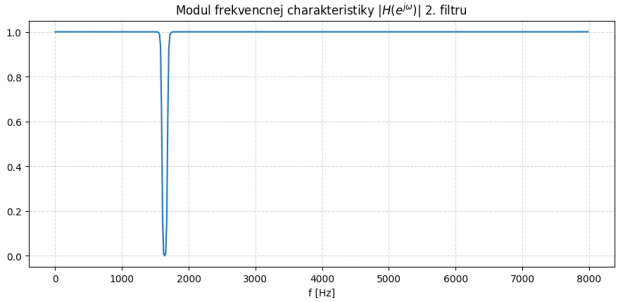
Zadanie 4.9 – Frekvenčná charakteristika

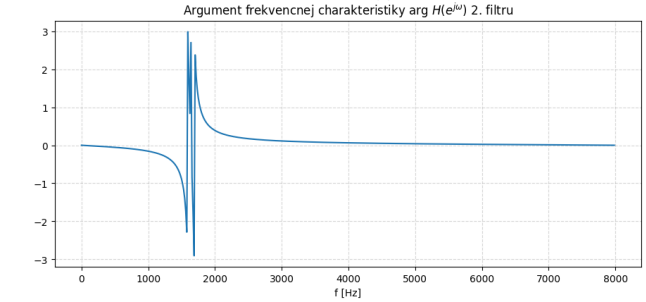
- Frekvenčnú charakteristiku pre jednotlivé filtre som získal pomocou funkcie *scipy.signal.freqz()*

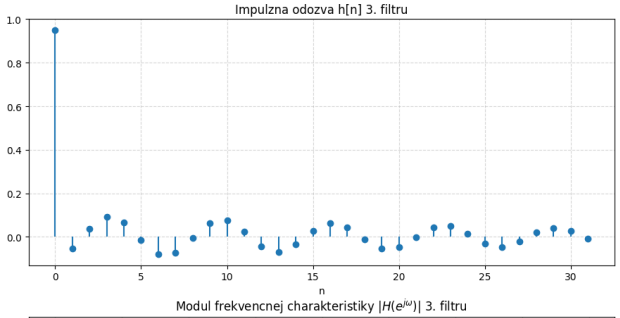


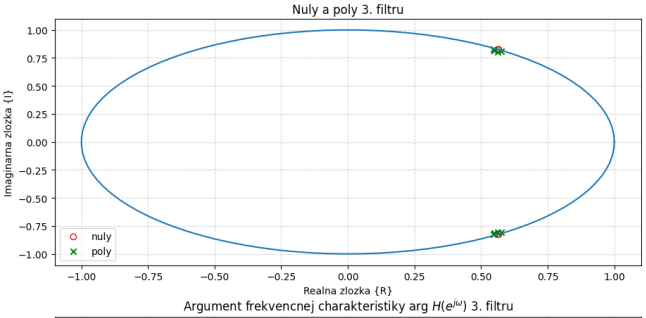

Impulzna odozva h[n] 2. filtru
n

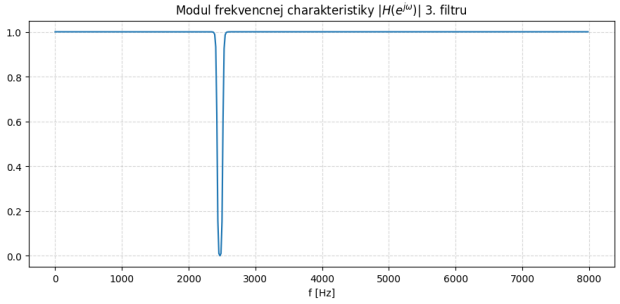

Nuly a poly 2. filtru
Imaginarna zlozka {I}
Realna zlozka {R}
nuly
poly

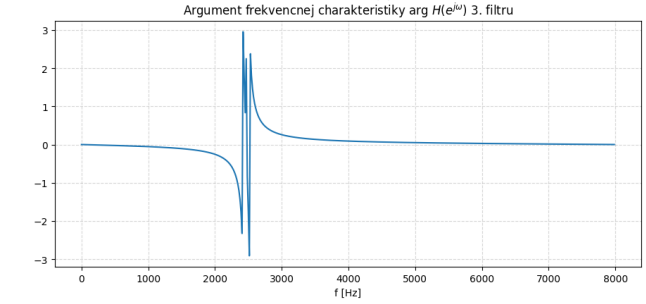

Modul frekvencnej charakteristiky |H(e^jω)| 2. filtru
f [Hz]

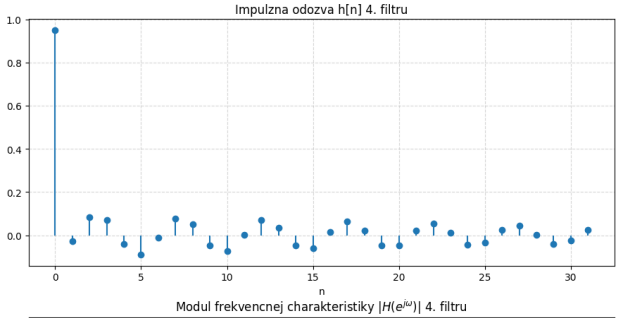

Argument frekvencnej charakteristiky arg H(e^jω) 2. filtru
f [Hz]

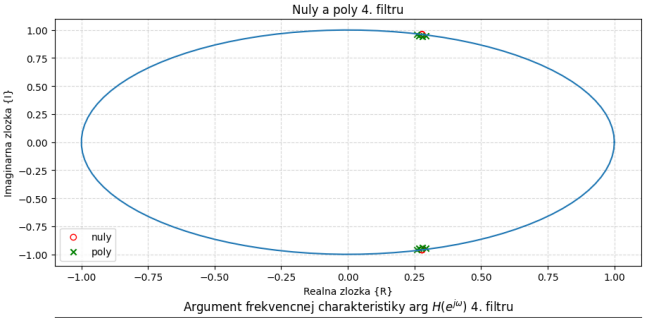

Impulzna odozva h[n] 3. filtru
n

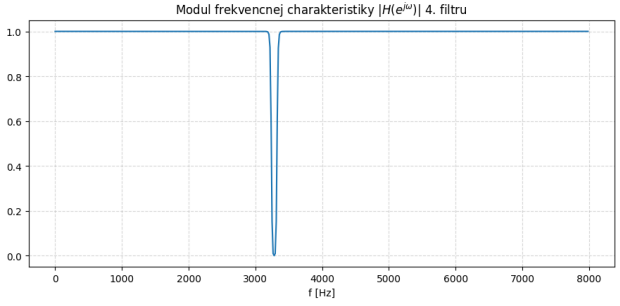

Nuly a poly 3. filtru
Imaginarna zlozka {I}
Realna zlozka {R}
nuly
poly

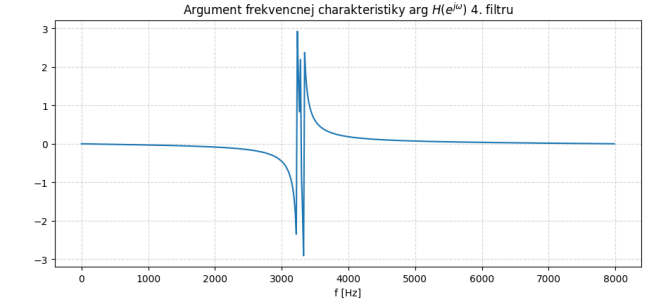

Modul frekvencnej charakteristiky |H(e^jω)| 3. filtru
f [Hz]


Argument frekvencnej charakteristiky arg H(e^jω) 3. filtru
f [Hz]


Impulzna odozva h[n] 4. filtru
n


Nuly a poly 4. filtru
Imaginarna zlozka {I}
Realna zlozka {R}
nuly
poly


Modul frekvencnej charakteristiky |H(e^jω)| 4. filtru
f [Hz]


Argument frekvencnej charakteristiky arg H(e^jω) 4. filtru
f [Hz]

Zadanie 4.10 – Filtrácia

- Pomocou vytvorených filtrov som odfiltroval signál, normalizoval ho do dynamického rozsahu <-1,1> a vygeneroval výsledný signál ***'clean_bandstop.wav'***
- Ako môžeme vidieť na spektrograme výsledného signálu, filtrácia sa podarila takmer na celom signále (v prvých 0,015s počuť ešte zvyšok rušivých frekvencií, čo je možné zanedbať)