Laboratorijska vježba 7

Digitalni filteri -uvod

U ovim vježbama prikazat ćemo primjenu jednog od digitalnih filtera. Digitalni filter koji smo izabrali je FIR filtera. Za ulazni signal koristi ćemo .wav file iz kojeg treba da uklonimo neželjene frekvencije.

FIR filter je digitalni filter koji za razliku od IIR filtera ima konačni impulsni odziv.

Izlaz FIR filtera ovisi o trenutnim i prošlim ulazima te njegov impulsni odziv opada prema nuli. FIR filter postiže linearni prikaz faznog odziva i propušta signal bez iskrivljenja faze.

Veoma bitna karakteristika filtera je linearna fazna karakteristika, a to FIR filter ima da bi se izbjegao gubitak bitnih informacija.

Dizajn FIR filtera

Pa da počnemo :).
Grafik polova i nula prikladan je za vizualizaciju odnosa između z-domene i karakteristika frekventnog odziva. Na auditornim vježbama smo spominjali da se frekventni odziv sistema $H(e^{j\omega})$ može izračunati procjenom prenosne funkcije H(z) pri vrijednostima kada je $z = e^{j\omega}$. Pošto je frekventni odziv periodičan sa periodom 2π dovoljno je procijeniti frekvenciju za $-\pi < \omega < \pi$. Taj period odgovara jediničnom z-krugu, počinje u -1 i završava tu.

FIR filteri sadrže samo nule i nemaju polove na svom dijagramu. Za FIR filtere mjesto nula H(z) na jediničnoj kružnici poništava određene frekvencije. Na osnovu toga da bismo dizajnirali FIR filter za poništavanje određene frekvencije ω, samo je potrebno postaviti nule na odgovarajuće mjesto na jediničnom krugu gdje dobitak filtera treba biti nula.

To ćemo sada i vidjeti.

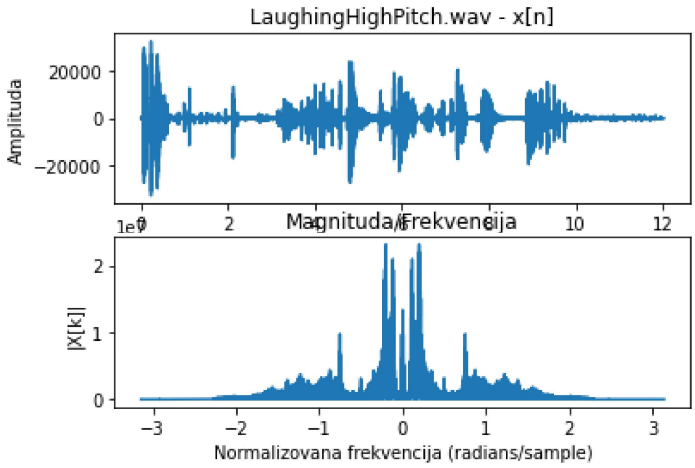
U našem slučaju, kao što smo već naveli, koristit ćemo audio datoteku kao ulaz. Koristit ćemo DTFT kao alat za analizu karakteristika frekventnog područja našeg signala.

```
# -*- coding: utf-8 -*-
"""
Kako dizajnirati jednostavan digitalni FIR filter za odbijanje
ne eljenih frekvencija u dolaznom signalu?
"""

import numpy as np
import matplotlib.pyplot as plt

from math import ceil,log,pi,cos
from scipy.fftpack import fft,fftfreq,fftshift

```

```
def compute_DTFT(x,M=0):
    """
    Sljede a se funkcija koristi za izra unavanje DTFT niza iz audio datoteke.
    """
    N = max(M,len(x)) #M je  eljena  du ina za prora un DTFT
    N = 2**(ceil(log(N)/log(2))) #N broj samplova

    X = fftshift(fft(x,N)) #fftshit - pomak komponente nulte frekvencije u
                           #sredi te spektra.
    w = 2*pi*fftshift(fftfreq(N))
    return (X,w) #Vra a DTFT X[k] i odgovaraju e frekvencije w(omega) poredane
                 #od -pi do pi

##############################################################################
# U frekventnom podru ju vrijednosti zauzimaju pozitivnu i negativnu
# frekvencijsku osu.
# Da bi se DFT vrijednosti prikazale na frekventnoj osi s pozitivnim i
# negativnim vrijednostima,
# DFT vrijednost na indeksu uzorka 0 mora biti centrirana na sredini polja.
# To se posti e kori tenjem funkcije FFTshift u Scipy Pythonu.
##############################################################################
```

Sada ćemo učitati audio datoteku, uzorke ćemo učitati kao signalni niz x i iscrtati u vremenskoj domeni / frekventnoj domeni (koristeći DTFT).

```
from scipy.io.wavfile import read # read za u itavanje wav file-a

samplerate, x = read('LaughingHighPitch.wav') #vra a brzinu uzorkovanja i
                                              #uzorke podataka.
duration = len(x)/samplerate #trajanje i vremenski vektor zvu nog uzorka
time = np.arange(0,duration,1/samplerate)

#Iscrtavanje signala (time domain)
fig1,(ax1,ax2) = plt.subplots(nrows=2,ncols=1)

ax1.plot(time,x)
ax1.set_xlabel('Vrijeme (s)')
ax1.set_ylabel('Amplituda')
ax1.set_title('LaughingHighPitch.wav - x[n]')

#Iscrtavanje DTFT na eg signala (frequency domain)
(X,w)= compute_DTFT(x)
ax2.plot(w,abs(X))
ax2.set_xlabel('Normalizovana frekvencija (radians/sample)')
ax2.set_ylabel('|X[k]|')
ax2.set_title('Magnituda/Frekvencija')
```



Drugi grafikon prikazuje skokove, te peek-ove možemo zabilježiti koristeći numpy.argmax()

```
maxIndex = np.argmax(X)
theta = w[maxIndex]
print(theta)
```

Dobijamo vrijednost $\theta = -0.19814717458516556$

Kosinusni signal može se predstaviti matematički kao:

$$x[n] = cos(\theta n) = \frac{1}{2}\left(e^{j\theta n} + e^{-j\theta n}\right)$$

Na malom dijelu druge slike je naš "čisti" signal kojeg trebamo otkriti upotrebnom FIR filtera i odbijanjem naše dobijene θ vrijednosti.

Za dizajn FIR filtra imamo dvije nule:

$$z_1 = e^{j\theta} \qquad z_2 = e^{-j\theta}$$

Prenosna funkcija FIR filtera je data kao:

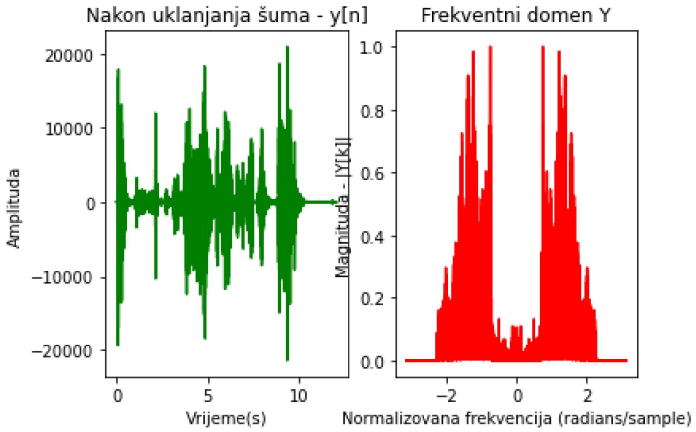
$$H_f(z) = (1 - z_1 z^{-1})(1 - z_2 z^{-1}) = 1 - 2\cos(\theta)z^{-1} + z^{-2}$$

, odakle su koeficijenti filtera:

$$b_0 = 1, b_1 = -2cos(\theta), b_2 = 1 \text{ i } a_0 = 1$$

Sada ćemo filtrirati ulazni audio signal kroz dizajnirani filter i filtrirati filtrirani izlaz u vremenskoj i frekvencijskoj domeni. Za postupak filtriranja koristi se funkcija lfilter iz scipy.signal paketa.

```
from scipy.signal import lfilter

b = [1,-2*cos(theta),1] #Koeficijenti filtera
a = [1]
y_signal = lfilter(b,a,x)
fig3, (ax3,ax4) = plt.subplots(nrows=1,ncols=2)
ax3.plot(time,y_signal,'g')
ax3.set(title='Nakon uklanjanja  uma  - y[n]',xlabel='Vrijeme(s)',ylabel='Amplituda')

(Y,w)= compute_DTFT(y_signal)
ax4.plot(w,abs(Y)/max(abs(Y)),'r')
ax4.set(title='Frekventni domen Y',xlabel='Normalizovana frekvencija (radians/sample)'
        ,ylabel='Magnituda - |Y[k]|')
```

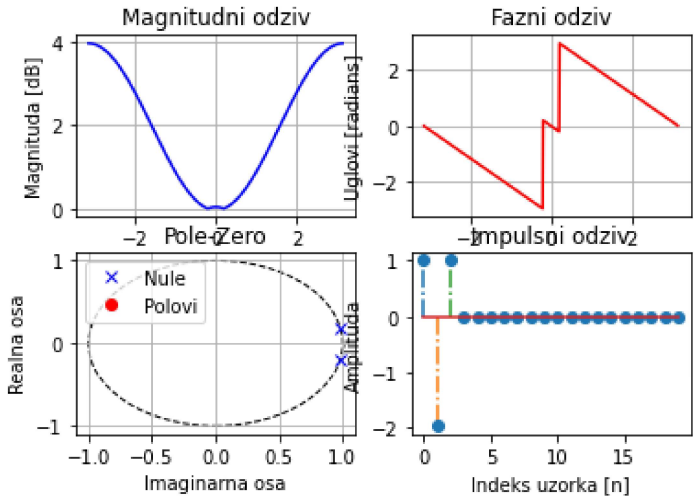


Snimimo izlaznu audio datoteku.

```
from scipy.io.wavfile import write
output_data = np.asarray(y_signal, dtype=np.int16)#
write("izlaz_bez_uma.wav", samplerate, output_data)
```

Karakteristike FIR filtera

```
from scipy.signal import freqz

w, h = freqz(b,a,whole=True)#frekventni odziv h[e^(jw)]
#whole = za cijeli period 0 do 2*pi
#Za iscrtavanje dvostranog odziva - fftshift
w = w - 2*np.pi*(w>=np.pi) #convert to range -pi to pi
w = fftshift(w)
h = fftshift(h)


#############################################################################
#Potrebno je nacrtati odziv magnitude, fazni odziv, grafikon polova i nula i
#impulsni odziv dizajniranog filtra.
#############################################################################
#Magnitudno fazni odziv
fig2, (ax) = plt.subplots(nrows=2,ncols=2)
ax[0,0].plot(w,abs(h),'b')
ax[0,0].set(title='Magnitudni odziv',xlabel='Frekvencija [radians/sample]',ylabel='
    Magnituda [dB]')
ax[0,0].grid();ax[0,0].axis('tight');

#Fazni odziv
angles = np.unwrap(np.angle(h))
ax[0,1].plot(w,angles,'r')
ax[0,1].set(title='Fazni odziv',xlabel='Frekvencija [radians/sample]',ylabel='Uglovi [
    radians]')
ax[0,1].grid();ax[0,1].axis('tight');

# Prenosna funkcija Pole-Zero reprezentacije
from scipy.signal import tf2zpk
z, p, k = tf2zpk(b, a)

# Iscrtavanje polova i nula na z-osi
from  matplotlib import patches
patch = patches.Circle((0,0), radius=1, fill=False,
                    color='black', ls='dashed')
ax[1,0].add_patch(patch)
ax[1,0].plot(np.real(z), np.imag(z), 'xb',label='Nule')
ax[1,0].plot(np.real(p), np.imag(p), 'or',label='Polovi')
ax[1,0].legend(loc=2)
ax[1,0].set(title='Pole-Zero',ylabel='Realna osa',xlabel='Imaginarna osa')
ax[1,0].grid()

#Impulsni odziv
#Kreira se impulsni signal
imp = np.zeros(20)
imp[0] = 1

from scipy.signal import lfilter
y_imp = lfilter(b,a,imp) #impulsni signal kroz filter
ax[1,1].stem(y_imp,linefmt='-.')
ax[1,1].set(title='Impulsni odziv',xlabel='Indeks uzorka [n]',ylabel='Amplituda')
ax[1,1].axis('tight')
```



Primjer našeg signala nema baš pretjerano izražen šum, ali vi možete probati sa nekim drugim proizvoljnim signalom visokih ili niskih frekvencija. Kako budete primjenjivali različite signale i koristili filter nad tim signalima, vidjet ćete razliku u izlaznom signalu odnosno kvalitetu izlaznog signala.