Filtre digitale - Aplicații în procesarea audio

Ene Cristian Ștefan

1. Noțiuni teoretice

1.1 Transformata Z

În procesarea digitală a semnalelor este folosită des transformata Z, fiind echivalenta discretă a transformatei Laplace.

$$X(z) = \sum_{n=-\infty}^{\infty} x[n]z^{-n} \tag{1}$$

1.2 Convoluția semnalelor

Convoluția este o operație matematică, care combină două semnale pentru a genera un al treilea semnal.

$$y[n] = x[n] * h[n] = \sum_{k=-\infty}^{\infty} x[k]h[n-k] \tag{2}$$

1.3 Filtre digitale

Filtrele electronice sunt circuite care prezintă un comportament selectiv în frecvență. Ele sunt construite din componente pasive: rezistori, bobine, condensatoare sau componente active: tranzistori, amplificatoare operaționale.

Filtre digitale sunt algoritmi de calcul, care se aplică semnalelor discrete. Se pot proiecta algoritmi care reproduc caracteristicile celor patru tipuri de filtre : trece-jos (FTJ), trece-sus (FTS), trece-bandă (FTB), oprește bandă (FOB).

Avantaje:

- Performanțele nu sunt afectate de mediul înconjurător
- Atenuarea este mare în banda de oprire
- Se pot filtra mai multe semnale cu ajutorul multiplexării
- Proiectarea unor caracteristici imposibil de realizat cu filtre analogice
- Performanțele pot fi reproduse de diferite procesoare, fără să fie necesare anumite reglaje

Dezavantaje:

- Frevența de procesare este limitată de către procesor,dar și viteza cu care se realizează conversia analog-numerică și digital-analogică
- Prelucrarea se realizează pe lățime de bandă mai mică decât cea a filtrelor clasice

Frecvența de lucru a filtrelor digitale se numește frecvență normată. Filtrele digitale au o frecvență maximă de lucru, aceasta se numește frecvența Nyquist și este jumătate din frecvența de eșantionare.

$$f_{Nyquist} = \frac{f_e}{2} \tag{3}$$

1.4 Filtre cu răspuns finit la impuls (FIR)

Funcția de transfer este dată de formula de (4) de mai jos:

$$H[z] = \sum_{k=0}^{N-1} a_k z^{-k} \tag{4}$$

N = lungimea filtrului/ numărul de eșantioane necesare calculării unui nou filtru eșantionat

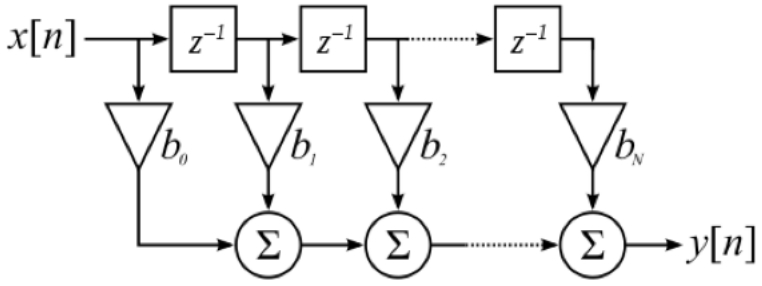

Figura 1. Structura logică a unui filtru FIR

Acest filtru se aplică asupra eșantionului curent și operațiilor precedente (de înmulțire /adunare). Z^{-1} se numește bloc de întârziere și are rolul să memoreze un eșantion.

1.5 Prelucrarea semnalelor audio

Prin intermediul limbajului de programare Python se poate înregistra, prelucra și reda semnale audio. Pentru a realiza acest lucru este necesară instalarea unui pachet numit "sounddevice". Se pot utiliza și mostre audio gata înregistrare pentru realizarea de diferite prelucrări. Asupra semnalelor audio se pot aplica filtre clasice, FIR sau operația de convuție.

2. Rezolvarea cerințelor

1. Definiți vectorul z ca fiind convoluția a doi vectori x și y;

```
      import numpy as np
from scipy import signal
import matplotlib.pyplot as plt

x=np.array([3,4,-2])
y=np.array([1,1,5,4,3])
z=signal.convolve(x,y) #Convolutia semnalelor x si h n=np.arange(0,7)
n=np.arange(0,7)

#Reprezentare grafica
plt.subplot(2,2,1)
plt.stem(np.arange(0,3),x,basefmt="b") #schimbare a culorii pentru linia de
baza
plt.xlim([0,7]) #definirea limitelor pentru axa Ox

plt.ylim([-3,20]) #definirea limitelor pentru axa Oy plt.xlabel('n')
plt.xlabel('n')
plt.ylabel('x[n]')

plt.subplot(2,2,2)
plt.stem(np.arange(0,5),y,basefmt="b")
plt.xlim([0,10])
plt.ylim([0,10])
plt.xlabel('n')
plt.ylabel('y[n]')

plt.subplot(2,1,2)
plt.stem(n,z,basefmt="b")
plt.xlim([0,6])
plt.ylim([-2,40])
plt.xlabel('n')
plt.ylabel('z[n]')

plt.tight_layout()
plt.show()
```

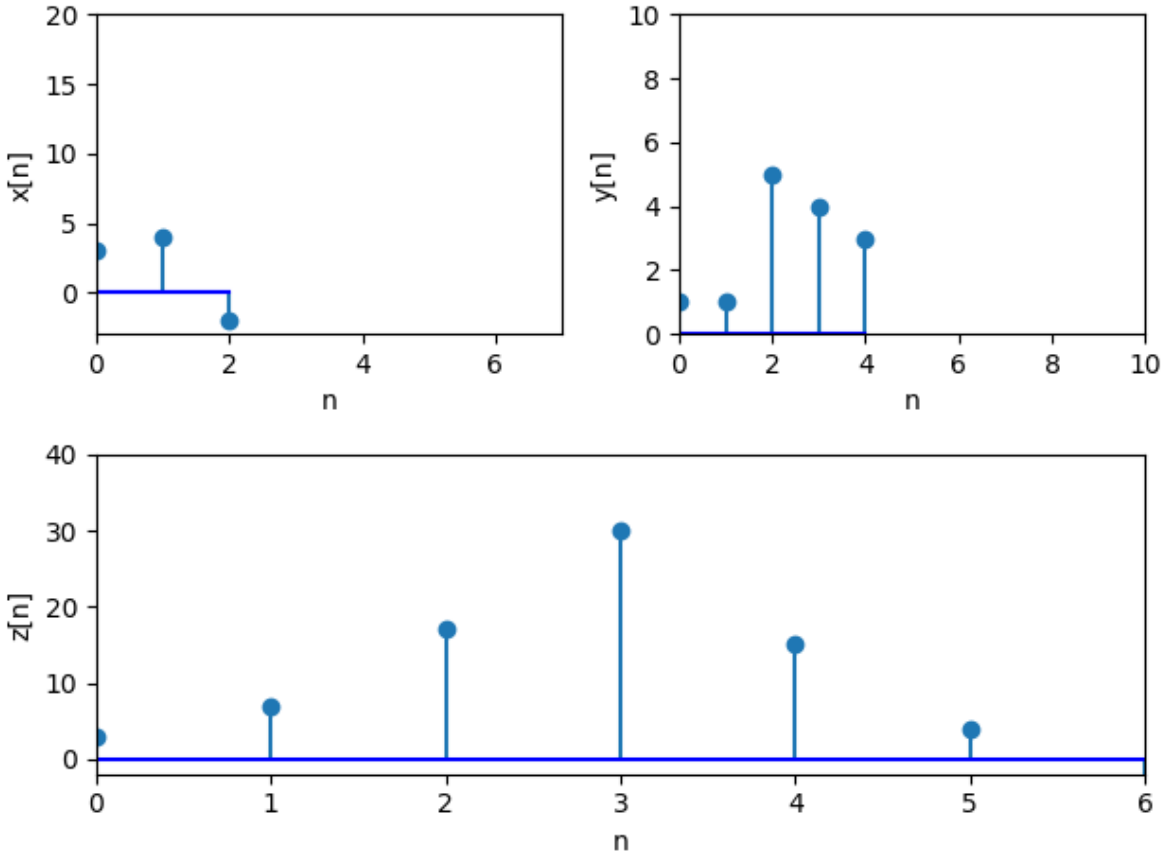

Figura 2. Reprezentarea convoluției

2. Reprezentanți caracteristicile de amplitudine si fază pentru un filtru FIR cu 32 de termeni de tip FTJ și frecvența critică 0.6;

```
import numpy as np
from scipy import signal
import matplotlib.pyplot as plt
f=signal.firwin(32,0.6,pass_zero='lowpass')
#Afisare
w,h=signal.freqz(f) #Generare componente amplitudine si faza pentru FIR
anterior
w=w/np.pi
plt.subplot(2,1,1)
plt.plot(w,20*np.log10(abs(h)),'b')
plt.xlabel('Frecventa normata [x$\pi$ rad/esantion]')
plt.ylabel('Amplitudine [dB]')
plt.xlim([0,1])
plt.grid()
plt.subplot(2,1,2)
angles=np.unwrap(np.angle(h))
plt.plot(w,angles,'g')
plt.xlabel('Frecventa normata [x$\pi$ rad/esantion]')
plt.ylabel('Faza [grade]')
plt.xlim([0,1])
plt.grid()
```

```
plt.tight_layout()
plt.show()
```

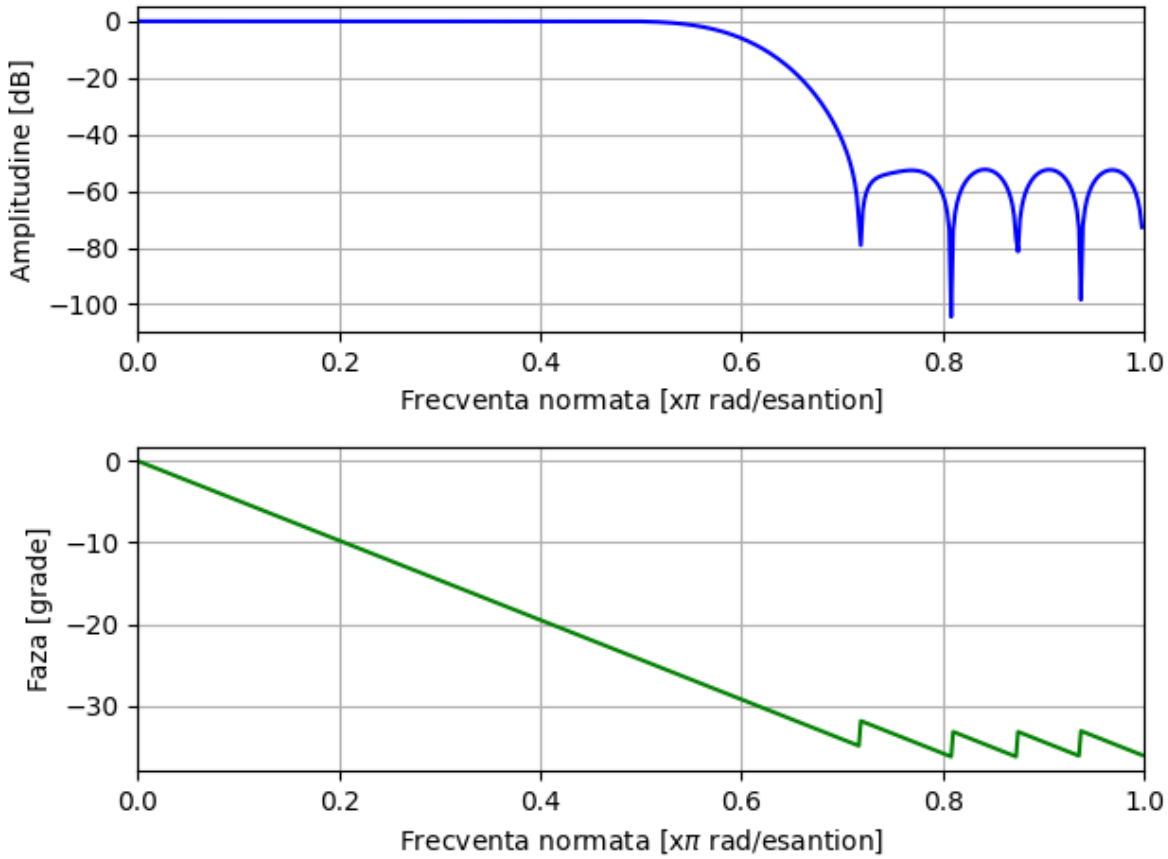

Figura 3. Caracteristicile de amplitudine si fază pentru un filtru FIR de tip FTJ

3. Reprezentanți caracteristicile de amplitudine și fază pentru un filtru FIR cu 53 de termeni de tip FTS și frecvența critică 0.45;

```
import numpy as np
from scipy import signal
import matplotlib.pyplot as plt
f=signal.firwin(53,0.45,pass_zero='highpass')
#Afisare
w,h=signal.freqz(f) #Generare componente amplitudine si faza pentru FIR
anterior
w=w/np.pi
plt.subplot(2,1,1)
plt.plot(w,20*np.log10(abs(h)),'b')
plt.xlabel('Frecventa normata [x$\pi$ rad/esantion]')
plt.ylabel('Amplitudine [dB]')
plt.xlim([0,1])
plt.grid()
plt.subplot(2,1,2)
angles=np.unwrap(np.angle(h))
plt.plot(w,angles,'g')
```

```
plt.xlabel('Frecventa normata [x$\pi$ rad/esantion]')
plt.ylabel('Faza [grade]')
plt.xlim([0,1])
plt.grid()
plt.tight_layout()
plt.show()
```

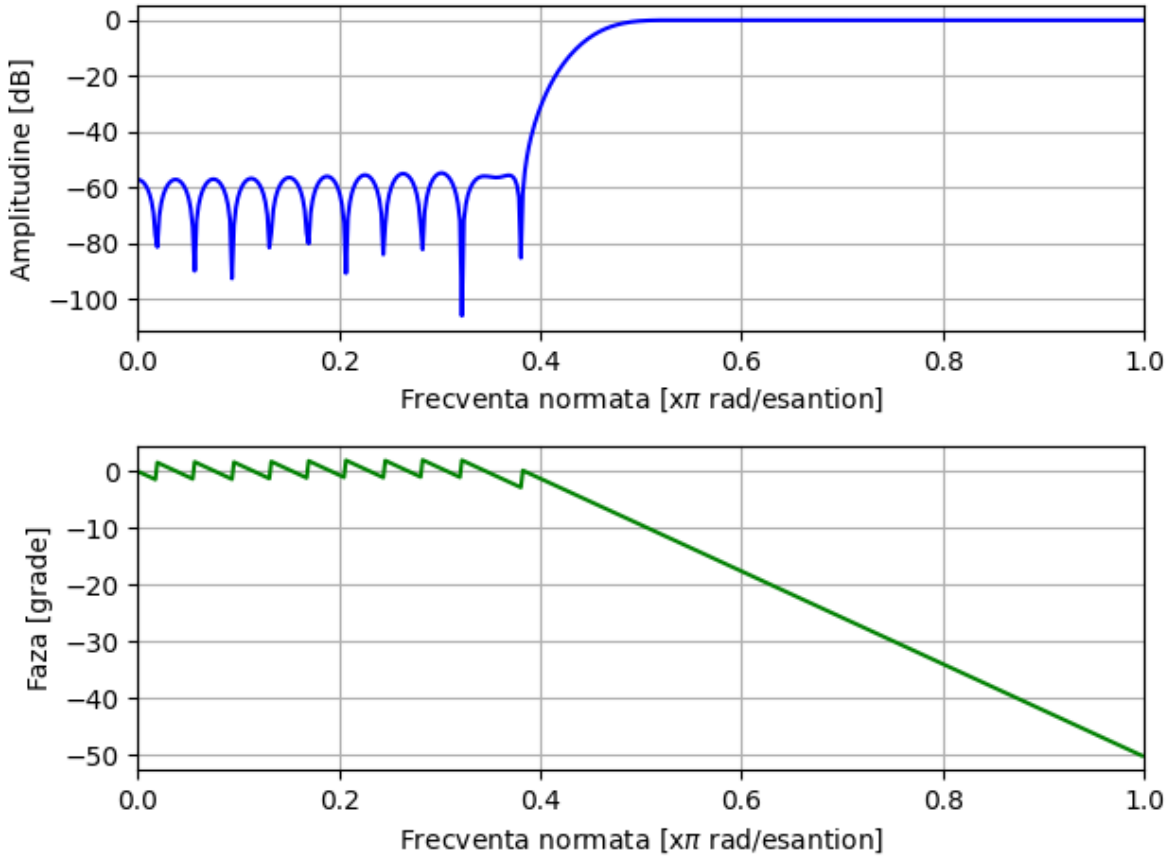

Figura 4. Caracteristicile de amplitudine si fază pentru un filtru FIR de tip FTS

4. Înregistrați un clip audio de 7 secunde și reprezentanți spectrograma acestuia

```
import sounddevice as sd
from scipy.io.wavfile import write
import matplotlib.pyplot as plt
fs = 12000 #Frecventa de esantionare
sec = 7 # Durata inregistrarii
#Inregistrare audio
rec = sd.rec(int(sec * fs), samplerate=fs, channels=1)
sd.wait()
rec=rec.flatten() #redimensionarea inregistrarii intr-un vector
write('inregistrare.wav', fs, rec) #salvare inregistrare
plt.specgram(rec,Fs=fs)
plt.xlabel('Timp [s]')
plt.ylabel('Frecventa')
plt.colorbar()#Afisaza legenda culorilor
```

```
plt.show()
```

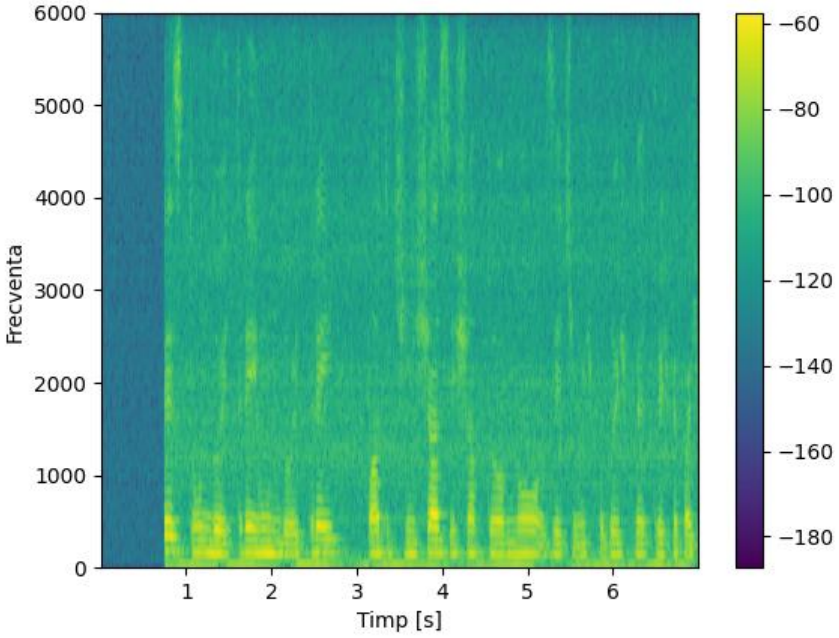

Figura 5. Spectogramă fisier audio înregistrat

5. Filtrați semnalul audio înregistrat anterior folosind un FTJ cu orice număr de termeni și cu frecvența de tăiere aleasă astfel încât să se filtreze cât mai bine frecvențele parazite înalte. Reprezentanți spectrograma semnalului filtrat.

```
import sounddevice as sd
from scipy.io.wavfile import write
from scipy import signal

from scipy.io.wavfile import read
import matplotlib.pyplot as plt
fs = 12000 #Frecventa de esantionare
sec = 7 # Durata inregistrarii
f=signal.firwin(32,0.8,pass_zero='lowpass')
fs,rec=read('inregistrare.wav') #Citire inregistrare
rec=rec.flatten()#redimensionarea inregistrarii intr-un vector
y=signal.convolve(rec,f)
plt.specgram(rec,Fs=fs)
plt.xlabel('Timp [s]')
plt.ylabel('Frecventa')
plt.colorbar()#Afisaza legenda culorilor
plt.show()
```

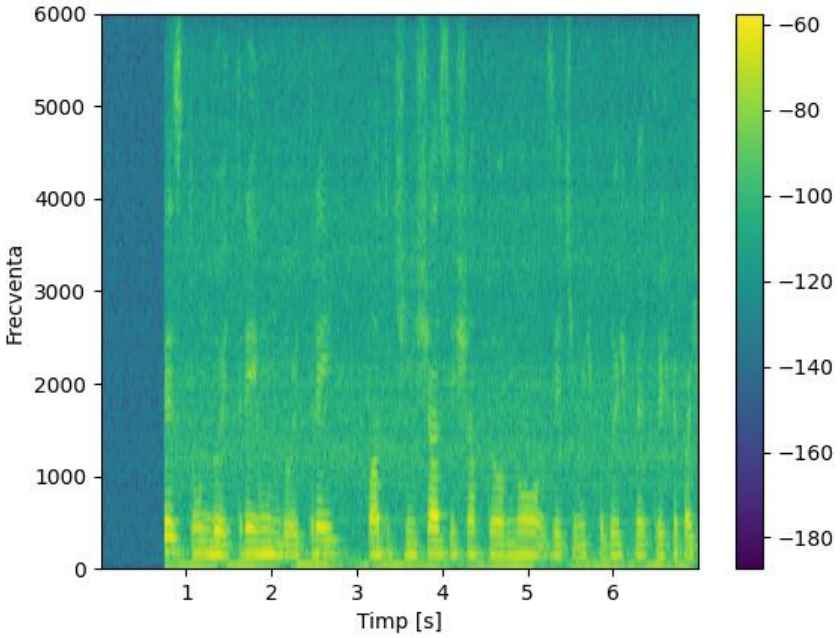

Figura 6. Spectogramă după filtrare