CAPITOLUL V

Transformate Discrete

A. Aspecte Teoretice și Relații de Calcul

A1. Eșantionarea semnalelor analogice

Semnalele prezentate și utilizate în capitolele anterioare sunt semnale analogice, ce prezintă o variație continuă în timp, la nivel teoretic cu suport finit sau infinit, la nivel practic întotdeauna cu suport finit. Întrebarea care se pune, cum introducem aceste informații, modelate în mediul analogic, în mediul digital? Cum putem modela o variație continuă a unei mărimi fizice folosind un set de numere? În digital putem lucra doar cu biți, ”1” și ”0”, prin șiruri de astfel de caractere putem coda orice număr, căruia printr-un tabel de codare îi putem aloca o anumită semnificație. Spre exemplu, numărul binar ”10110011” (pe 8 biți) în zecimal este 179, iar acest număr într-un tabel de codare poate fi corespondent pentru litera ”a”, sau ”z” sau ”**Ω**”, totul ține de modul în care a fost definită tabela de codare. Iar un lucru pe care trebuie să îl avem în vedere este că indiferent de forma semnalelor cu care lucrăm, analogică sau digitală, este că în final ne interesează ca reprezentarea pe care o folosim să păstreze inteligibilă informația pe care o stochează de fapt acestea.

Exercițiu 1: Câte simboluri putem reprezenta printr-o astfel de tabelă de codare, bazată pe 8 biți?

Metoda cea mai simplă prin care putem obține o reprezentare discretă a unui semnal analogic este prin eșantionarea periodică [1]:

$$x(n) = x_a(nT), -\infty < n < \infty. \qquad \text{(V.1)}$$

Unde $x_a(t)$, reprezintă un semnal analogic continuu ce poate fi transformat într-un şir de numere care stochează ”mare parte” din informaţia originală printr-un circuit special construit astfel în cât să funcţioneze conform relaţiei V.1. Prin exemplele pe care le vom analiza în continuare vom vedea ce înseamnă acest ”mare parte”.

Studiu de caz *Transformata Fourier pentru un semnal de tip poartă, varianta analogică şi varianta discretă*

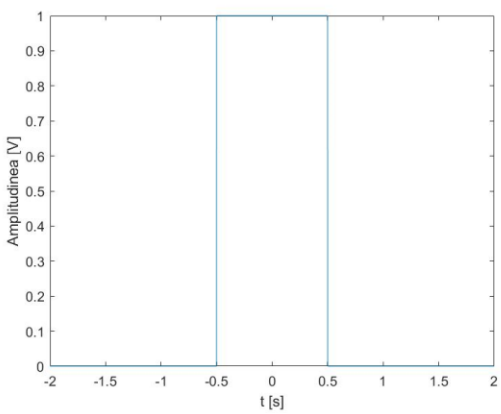

Fig.V.1a Exemplu semnal poartă durată 1s

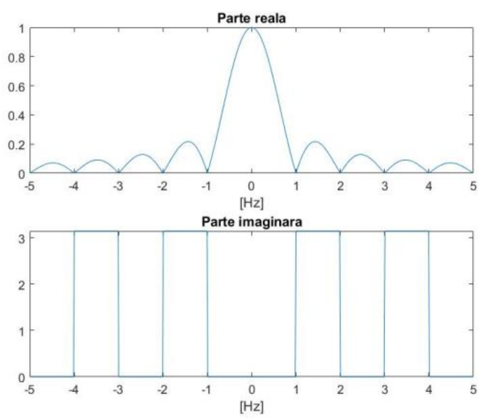

Fig.V.1b Transformata Fourier a semnalului tip poartă

În Fig. V.1 putem vedea un semnal de tip poarta şi transformata Fourier asociată care ne arată componentele spectrale importante. O observaţie pe care putem să o facem este că un semnal de durată finită, aparent banal, în frecvenţă devine chiar complex deoarece are o bandă largă, ce tinde spre infinit.

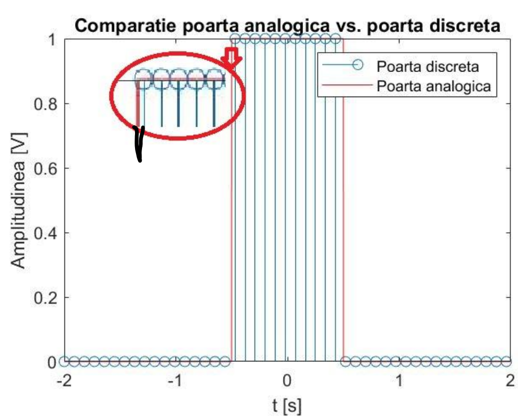

Fig.V.2a Comparaţie poartă discretă în timp vs. poartă analogică

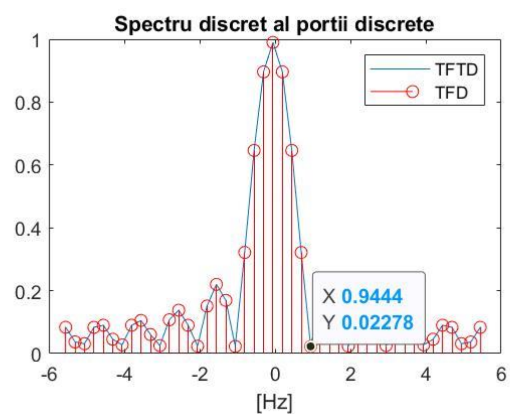

Fig.V.2b Transformata Fourier în timp discret vs. Transformata Fourier discretă

În Fig.V.2a putem vedea suprapuse un semnal analogic de tip poartă cu suportul 1 secundă în comparație cu varianta discretă obținută folosind un bloc cu frecvența de eșantionare de 11.11 Hz. Dacă ne uităm mai atent la colțul din stânga sus al porții vedem ca o parte din informație s-a pierdut, deoarece blocul de eșantionare nu începe să stocheze valori exact în momentul în care poarta ia valori diferite de 0, în final varianta digitală a porții noastre nu va mai avea 1 secundă suportul ci doar 0.94 secunde. Prin urmare am pierdut 0.06 secunde de informație importantă. De asemenea în Fig.V.2b se poate observa spectrul discret în comparație cu spectrul continuu al acestui semnal. Deși teoretic putem lucra și cu spectrul continuu modelat grafic, în realitate și în frecvență vom putea lucra efectiv pe calculator cu spectre discrete (șiruri de numere nu funcții abstracte). O diferență notabilă între spectrul din Fig.V.1b și spectrul din Fig.V.2b este că în timp ce în cazul analogic spectrul este infinit, în cazul discret spectrul este limitat de frecvența de eșantionare folosită, astfel, banda semnalului discret va fi între -Fs/2 și Fs/2 (Fs-frecvența de eșantionare folosită), în exemplul particular -5.55 Hz și 5.55 Hz.

Codul îl regăsiți în tabelul de la sfârșitul introducerii,

Tema 1: Încercați să rulați codul corespunzător exemplului anterior. Modificați durata semnalului tip poartă, modificați frecvența de eșantionare.

Tema 2 Refaceți analiza pentru un semnal de tip sinus. În următoarele setări particulare:

a. frecvența sinusului de 10 Hz, frecvența de eșantionare de 20 Hz
b. frecvența sinusului de 10 Hz, frecvența de eșantionare de 100 Hz
c. frecvența sinusului de 10 Hz, frecvența de eșantionare de 7 Hz

Salvați spectrele din cele 3 situații și comparați frecvențele ce se pot observa pe spectre. Din ce cauză apar diferențe? Care este banda în frecvența a unui semnal cu suport finit (poarta), în comparație cu banda unui semnal cu suport infinit (sinusul)? Cum putem evita diferențele ce se pot observa? Căutați Teorema lui Nyquist!

Cod MATLAB/Octave	*Cod Python*

<pre>clear all;close all;clc;
pas = 1/1000;
limita=2;
t=-limita:pas:limita;
t0 = 0; %deplasarea porÈ›ii
A = 1;
x = poarta(-A/2,A/2,1,t-t0);
figure(1);
plot(t,x);
xlabel('t [s]');
ylabel('Amplitudinea [V]');
k=10;
omega = -k*pi/A:1/10:k*pi/A;
 	
X = zeros(1,length(omega));
for i=1:length(omega)
    X(i) = quad(@(t)poarta(-A/2,A/2,1,t-t0).*exp(-
1i*omega(i)*t),-10,10);
    re = real(X(i));
    im = imag(X(i));
    if abs(re)<10^-10
        re = 0;
    end
    if abs(im)<10^-10
        im = 0;
    end
    X(i) = re+1i*im;
end
 	
figure(2);
subplot(2,1,1);
plot(omega/(2*pi),abs(X)), title('Parte reala');
xlabel('[Hz]');
subplot(2,1,2);
plot(omega/(2*pi),angle(X)), title('Parte imagi-
nara');
xlabel('[Hz]');</pre>
	<pre>import numpy as np
	import matplotlib.pyplot as plt
	from mpldatacursor import datacursor
	import scipy.integrate as integrate
	
	#importuri necesare pentru ca graficele sa se deschida
	#in fereastra separata si sa putem folosi zoom si datacursor
	from IPython import get_ipython
	get_ipython().run_line_magic('matplotlib', 'qt')
	
	def poarta(a,b,amp,t):
	# functia returneaza o treapta in intervalul [a,b], conform cu
	baza de timp t
	    y=0;
	    if t>=a and t<=b:
	        y = amp;
	    return y;
	
	pas=1/1000;
	limita=2;
	t=np.arange(-limita, limita, pas)
	t0 = 0; #deplasarea portii
	A = 1;
	i=0;
	x=np.zeros(len(t));
	for val in (t-t0):
	    x[i] = poarta(-A/2,A/2,1,val);
	    i=i+1;
	
	f,(ax1)=plt.subplots(1,1);
	ax1.plot(t,x) ;
	ax1.set_xlabel("Timpul, t (s)") ;
	ax1.set_ylabel("x(t) [V]") ;
	ax1.set_ylim(-0.5, 1.5)
	
	k=10;
	omega = np.arange(-k*np.pi/A,k*np.pi/A,1/10);
	
	X = np.zeros(len(omega));
	i=0;</pre>

<pre>figure(3);
plot3(omega,real(X),imag(X));
xlabel('Frecventa unghiulara');
ylabel('Partea reala');
zlabel('Partea imaginara');
 	
%%% esantionare semnale poarta
T=0.09;
N=limita/T;
for n=-N:N
    xd(n+N+1)=poarta(-A/2,A/2,1,n*T);
end
%x reprezeninta varianta discreta a semnalului de
tip poarta
%obtinut prin esantionare
X=fft(xd);
n=-N:N;
figure, stem(n*T,xd);
hold on
plot(t,x,'r-');
hold off
legend('Poarta discreta', 'Poarta analogica');
title('Comparatie poarta analogica vs. poarta dis-
creta');
xlabel('t [s]');
ylabel('Amplitudinea [V]');
freq=n/(N*2*T);
figure, plot(freq,2*fftshift(abs(X)/N));
hold on
stem(freq,2*fftshift(abs(X)/N),'r');
xlabel('[Hz]');
title('Spectru discret al portii discrete');
legend('TFTD', 'TFD');
hold off</pre>
	<pre># esantionare semnale poarta
	
	T=0.09;
	
	N=np.round(limita/T);
	
	valori=np.arange(-N,N,1);
	
	xd=np.zeros(len(valori));
	
	for n in valori:
	
	    xd[int(n+N)]=poarta(-A/2,A/2,1,n*T);
	
	#x reprezeninta varianta discreta a semnalului de
	tip poarta
	
	#obtinut prin esantionare
	
	X=np.fft.fft(xd);
	
	n=valori;
	
	f,(ax1)=plt.subplots(1,1);
	
	ax1.stem(n*T,xd, label='Poarta discreta');
	
	ax1.set_xlabel("t [s]") ;
	
	ax1.set_ylabel("Amplitudinea [V]") ;
	
	ax1.set_title('Comparatie poarta analogica vs.
	poarta discreta');
	
	ax1.plot(t,x,'r-', label='Poarta analogica');
	
	legend = ax1.legend(loc='upper right', shadow=True,
	fontsize='x-large')
	
	freq=n/(N*2*T);</pre>

<pre>function y = poarta(a ,b, amp, t)
% functia returneaza o treapta in intervalul [a,b], conform cu baza de timp
% t
 	
y = zeros(1,length(t));
 	
for i=1:length(t)
    if t(i)>=a && t(i)<=b
        y(i) = amp;
    end
end</pre>
	<pre>f,(ax1)=plt.subplots(1,1);
	
	ax1.stem(freq,2*np.fft.fftshift(np.abs(X)/N),'r', label='TFD');
	
	ax1.set_xlabel("[Hz]") ;
	
	ax1.set_title('Spectru discret al portii discrete');
	
	ax1.plot(freq,2*np.fft.fftshift(np.abs(X)/N),'r-', label='TFTD');
	
	legend = ax1.legend(loc='upper right', shadow=True, fontsize='x-large')</pre>

A2. Transformata Z

Pentru semnalele analogice am văzut că putem obține analize și prelucrări mult mai complexe cu ajutorul transformatelor Fourier și Laplace. Echivalentul acestor două transformate în spațiul semnalelor discrete sunt Transformata Fourier Discretă și Transformata Z.

Transformata Z a unei secvențe discrete x(n) se definești prin:

$$X(z) = \sum_{n=-\infty}^{\infty} x(n) z^{-n}, r_1 < |z| < r_2. \quad \text{(V.2)}$$

A3. Transformata Fourier Discretă

Transformata Fourier ne permite ssă analizăm conținutul spectral al semnalelor discrete (digitale). Pe baza acestui utilitar putem analiza conținutul informațional al unui semnal într-o bază de frecvență în loc să folosim o bază de timp. Transformata Fourier Discretă are la bază perechea de analiză/sinteză [1]:

$$\text{Analiză: } X(k) = \sum_{n=0}^{N-1} x(n) W_N^{kn}, \quad \text{(V.3)}$$

$$\text{Sinteză: } x(n) = \frac{1}{N}\sum_{n=0}^{N-1} X(k) W_N^{-kn}, \qquad \text{(V.4)}$$

unde, $W_N = e^{-j\left(\frac{2\pi}{N}\right)}$ reprezintă o exponențială complexă de perioadă 2π.

Unde a dispărut noțiunea de frecvență? În spațiul discret se lucrează cu frecvențe normate, datorită frecvenței de eșantionare folosite F_s semnalul discret folosit, după cum am văzut la punctul ***A1***, devine un șir de numere (aparent se abstractizează noțiunea de timp). Doar în pereche cu perioada de eșantionare index-ul numerelor n capată semnificația temporală de nT_s. Dacă ne uităm ecuația de analiză V.3 putem observa că analiza Fourier într-un final se transformă dintr-un șir de numere într-un alt șir de numere, indexate dup n, index corespunzător unor frecvențe normate $\frac{2\pi n}{N}$, care la rândul lor corespunzătoare frecvențelor de $\frac{n}{N} F_s$ [Hz], cu $n = -\frac{N}{2} .. \frac{N}{2}$. În Fig. V.3 putem vedea analiza în timp și în frecvență a unui semnal discret obținut prin eșantionarea cu $F_s = 50Hz$ a unui semnal analogic ce conține 2 componente tonale de $5Hz$ și $12Hz$ de durată 1 secundă.

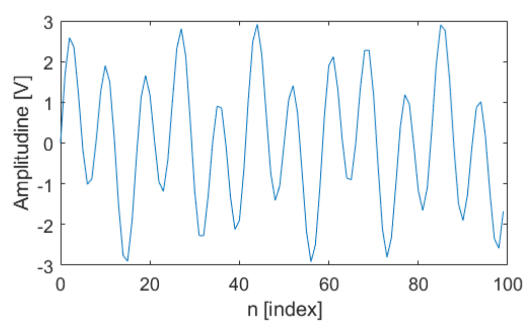

Fig.V.3a Semnal discret, șir de numere

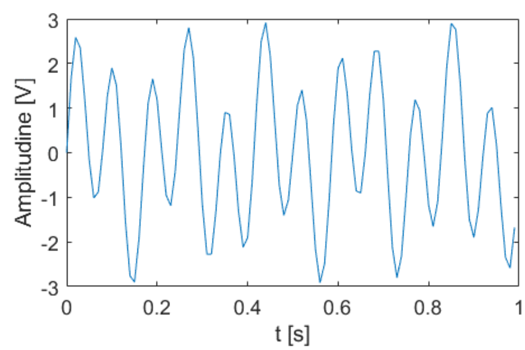

Fig.V.3b Semnal discret cu refacere referință temporală

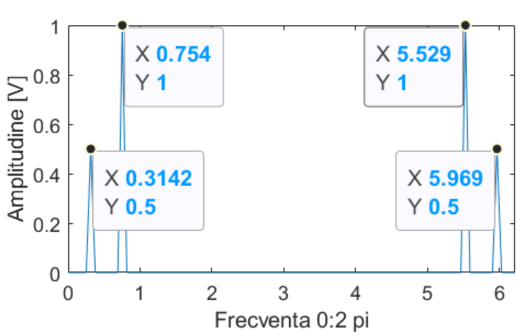

Fig.V.3c Analiza în frecvență normată

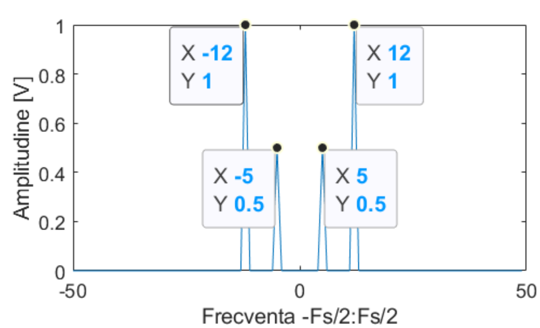

Fig.V.3d Analiză în frecvență de normată

<table>
<tr><th>Cod MATLAB/Octave</th><th>Cod Python</th></tr>
<tr><td>

```
clear all
close all
clc
Fs=100; %Hz Frecventa esantionare
N=100; %numar esantioane
durata=N/Fs;
n=0:N-1; %definire index esantioane
F1=5;% Hz
F2=12;% Hz
A1=1; % V
A2=2; % V
semnal=A1*sin(2*pi*F1*n/Fs)+A2*sin(2*pi*F2*n/Fs);
%generare semnal
figure, plot(n,semnal);xlabel('n [index]');yla-
bel('Amplitudine [V]');
figure, plot(n/Fs,semnal);xlabel('t [s]');yla-
bel('Amplitudine [V]');
figure, plot(2*pi*n/N,1/N*abs(fft(semnal))); xla-
bel('Frecventa 0:2 pi');ylabel('Amplitudine [V]');
figure, plot((n-N/2)/N*Fs,1/N*abs(fftshift(fft(sem-
nal))));xlabel('Frecventa -Fs/2:Fs/2');ylabel('Am-
plitudine [V]');
```

</td><td>

```
import numpy as np
import matplotlib.pyplot as plt

#importuri necesare  pentru ca graficele sa se deschida
#in fereastra separata si sa putem folosi zoom si
datacursor
from IPython import get_ipython
get_ipython().run_line_magic('matplotlib', 'qt')

Fs=100; #Hz Frecventa esantionare
N=100; #numar esantioane
durata=N/Fs;
n=np.arange(0,N-1);#definire index esantioane
F1=5;# Hz
F2=12;# Hz
A1=1; # V
A2=2; # V
semnal=A1*np.sin(2*np.pi*F1*n/Fs)+A2*np.sin(2*np.pi*F2*
n/Fs); #generare semnal
f,(ax1)=plt.subplots(1,1);
ax1.plot(n,semnal);
ax1.set_xlabel('n [in-dex]') ;
ax1.set_ylabel("Amplitudinea [V]") ;

f,(ax1)=plt.subplots(1,1);
ax1.plot(n/Fs,semnal);
ax1.set_xlabel('t [s]') ;
ax1.set_ylabel("Amplitudinea [V]") ;

f,(ax1)=plt.subplots(1,1);
ax1.plot(2*np.pi*n/N,1/N*np.abs(np.fft.fft(semnal)));
ax1.set_xlabel('Frecventa 0:2 pi') ;
ax1.set_ylabel("Amplitudinea [V]") ;

f,(ax1)=plt.subplots(1,1);
ax1.plot((n-
N/2)/N*Fs,1/N*np.abs(np.fft.fftshift(np.fft.fft(semnal)
)))
ax1.set_xlabel('Frecventa -Fs/2:Fs/2') ;
ax1.set_ylabel("Amplitudinea [V]") ;
```

</td></tr>
</table>

B. Aspecte practice de rezolvare a mini-proiectelor

Obiectivele lucrării curente sunt următoarele:

Familiarizarea cu librăriile de tip symbolic math ce pot fi folosite pentru a calcula Transformata Z cu ajutorul calculatorului.

Analiză spectru semnal înregistrat și evidență frecvență fundamentală.

B1. Symbolic math

Calculul Transformatei Z poate fi realizat ușor folosind comanda **symsum** în MATLAB și Octave și **sumation** în Python (din librăria SymPy).

Observație: Deși Octave folosește comenzile din MATLAB, în realitate librăria folosită este SymPy din Python, prin urmare ca să rulați codul de mai jos în Octave va trebui să aveți Anaconda instalat și să parcurgeți următorii pași [2]:

- `pkg install -forge symbolic`
- Setare cale python

  `setenv PYTHON C:\Users\valen\anaconda3\python (aveți grijă, calea poate să fie diferită la dumneavoastră);`

- `pkg load symbolic`
- încercați `syms x` și `factor(x^2-x-6)`

Debugging: încercați `sympref reset` și `sympref diagnose`.

Exemplu: Calculați Transformata Z pentru secvența

$$x(n) = \begin{cases} a^n, n \geq 0 \\ 0, n < 0 \end{cases}$$

Cod MATLAB/Octave	*Cod Python*

`syms k a z`
`trans = symsum( a^k * z^(-k),k,0,inf)`
`pretty(trans)`
	`from sympy import *`
	
	`a, k, z = symbols('a k z')`
	
	`pprint(summation(1*z**-k, (k, 0, oo)))`

Tema3: Calculați Transformata Z pentru următoarele semnale și identificați domeniile de convergență.

Semnalul x(n).
$x(n) = \begin{cases} 1, n \geq 0 \\ 0, n < 0 \end{cases}$
$x(n) = \begin{cases} 0, n \geq 0 \\ -1, n < 0 \end{cases}$
$x(n) = \begin{cases} n, n \geq 0 \\ 0, n < 0 \end{cases}$
$x(n) = \begin{cases} a^n, n \geq 0 \\ 0, n < 0 \end{cases}$
$x(n) = \begin{cases} 0, n \geq 0 \\ -a^n, n < 0 \end{cases}$
$x(n) = \begin{cases} 1, n \geq 0 \\ 0, n < 0 \end{cases}$
$x(n) = \begin{cases} \cos(\omega_0 n), n \geq 0 \\ 0, n < 0 \end{cases}$
$x(n) = \begin{cases} \sin(\omega_0 n), n \geq 0 \\ 0, n < 0 \end{cases}$
$x(n) = \begin{cases} a^n \cos(\omega_0 n), n \geq 0 \\ 0, n < 0 \end{cases}$
$x(n) = \begin{cases} a^n \sin(\omega_0 n), n \geq 0 \\ 0, n < 0 \end{cases}$

B2. Spectru semnal discret

Fișierul 'vocale.wav', găsit aici, conține pronunția vocalelor a, e, i, o, u, folosiți codul de mai jos pentru a analiza transformata Fourier a întregului semnal și transformata Fourier a unui cadru din semnal ce conține doar o anumită vocală.

Exercițiu 2: Comparații spectrele și încercați să măsurați și să modificați frecvența fundamentală a vorbitorului.

<table>
<tr><th>Cod MATLAB/Octave</th><th>Cod Python</th></tr>
<tr><td>

```
clear all
close all
clc
[semnal,Fs]=audioread('vocale.wav');
figure,plot(semnal);
%%selectati doar o anumita vocala din semnal
semnal=semnal(1000:2000);
%%lungime semnal analizat
L=length(semnal);
%%axa timp
t=(1:length(semnal))/Fs;
%%transformata Fourier rapida
Y = fft(semnal);
%%normalizare la lungimea semnalului
P2 = abs(Y/L);
%%vizualizare spectrul in functie de frecvente
pozitive
P1 = P2(1:L/2+1);
P1(2:end-1) = 2*P1(2:end-1);
f = Fs*(0:(L/2))/L;
figure,plot(f,P1)
title('Spectru semnal analizat')
xlabel('f (Hz)')
ylabel('|P1(f)|')
```

</td><td>

```
import numpy as np
import matplotlib.pyplot as plt
import soundfile as sf

plt.close('all')

semnal, Fs = sf.read('vocale.wav');

f,(ax1)=plt.subplots(1,1);
ax1.plot(semnal);

#selectati doar o anumita vocala din semnal
semnal=semnal[1000:2000];
#lungime semnal analizat
L=len(semnal);
#axa timp
t=np.arange(1,len(semnal))/Fs;
##transformata Fourier rapida
Y = np.fft.fft(semnal);
#normalizare la lungimea semnalului
P2 = np.abs(Y/L);
#vizualizare spectrul in functie de frecvente
pozitive
P1 = P2[0:int(L/2)];
P1[1:len(P1)] = 2*P1[1:len(P1)];
freq = Fs*np.arange(0,int(L/2))/L;
f,(ax1)=plt.subplots(1,1);
ax1.plot(freq,P1);
ax1.set_xlabel('f (Hz)');
ax1.set_ylabel('|P1(f)|');
ax1.set_title('Spectru semnal analizat');
```

</td></tr>
</table>

Temă 4

Realizați un simulator în Python/Matlab al unui dispozitiv care realizează un bloc ce modelează modul în care sunt apelate numerele de telefon în sistemul de comunicație analogic folosind dual-tone și a modulului care decodează impulsurile primite. Pentru explicații suplimentare vedeți [3].

Formarea unei cifre va crea un semnal sonor ce se bazează pe o combinație de două tonuri sonore similare cu ceea ce putem regăsi în tabelul de mai jos. ***Durata semnalului corespunzător unui simbol este de: 0.1s, 0.5s și 1s.***

	1209 Hz	1336 Hz	1477 Hz	1633 Hz
697 Hz	1	2	3	A
770 Hz	4	5	6	B
852 Hz	7	8	9	C
941 Hz	*	0	#	D

D. Bibliografie

[1] A. V. Oppenheim și R. W. Schafer, Discrete-Time Signal Processing, New Jersey: Prentice-Hall, 1989, pp. 149-201.

[2] C. B. Macdonald, „Installing Python and SymPy with Anaconda,” GitHub, [Interactiv]. Available: https://github.com/cbm755/octsympy/wiki/Notes-on-Windows-installation. [Accesat 28 07 1989].

[3] Wikipedia, „Dual-tone multi-frequency signaling,” Wikipedia, 15 07 2020. [Interactiv]. Available: https://en.wikipedia.org/wiki/Dual-tone_multi-frequency_signaling. [Accesat 27 07 2020].