

## **Lucrarea 6. Eșantionarea și cuantizarea semnalelor**

Semnalele pot fi clasificate atât după evoluția acestora în timp, cât și după evoluția acestora în amplitudine. În funcție de evoluția acestora în timp semnalele pot fi semnale continue, dacă evoluția este dată de o funcție continuă, și semnale discrete, dacă amplitudinea semnalului este cunoscută doar pentru momente discrete de timp. În funcție de evoluția semnalelor în amplitudine acestea pot fi semnale cuantificate continuu, a căror evoluție în timp a amplitudinii este continuă, amplitudinea putând lua orice valoare, și semnale cuantificate discret, a căror valori ale amplitudinii pot apartine unui set finit de valori.

*Tabel 1 – Eșantionarea semnalelor [3]*

		Timp	
		Continuu	Discret
Amplitudine	Continuă	Semnale continue (semnale analogice)	Semnale discrete
	Discretă	Semnale continue cuantificate	Semnale discrete cuantificate (semnale numerice)

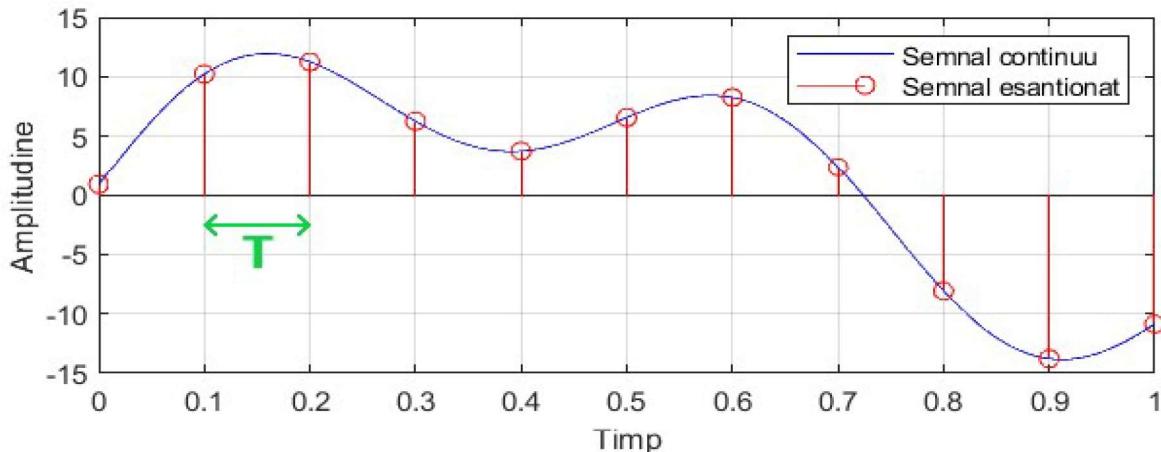
O reprezentare grafică a tipurilor de semnale exemplificate se poate vedea în Lucrarea 1.

### **5.1 Eșantionarea semnalelor**

Eșantionarea reprezintă discretizarea în timp a unui semnal în timp continuu [4].

*Tabel 2 – Eșantionarea semnalelor*

		Timp	
		Continuu	Discret
Amplitudine	Continuă	Semnale continue (semnale analogice)	 Eșantionare
	Discretă	Semnale continue cuantificate	 Eșantionare



*Figura 1 – Eșantionarea semnalelor*

Rezultatul eșantionării unui semnal continuu  $x(t)$  este un sir de eșantioane  $x[n]$ , un eșantion fiind valoarea semnalului  $x(t)$  la momentul  $t = n \cdot T$  [5].

Timpul dintre două eșantioane succesive (notat cu  $T$ ) se numește **perioadă de eșantionare** [5].

Numărul de eșantioane obținute într-o unitate de timp (secundă) se numește **frecvență de eșantionare** și se notează cu  $f_s = \frac{1}{T}$  [Hz] [5].

Dacă dorim să avem o acuratețe cât mai mare de informație, atunci este recomandat să alegem un număr cât mai mare de eșantioane (deci o frecvență de eșantionare mai mare).

### 5.1.1 Generarea eșantioanelor unui semnal în Python

Pentru a rezolva probleme de procesare a datelor, pentru semnale deterministe, putem genera, folosind Python, eșantioane ale acestui semnal. Pentru acest lucru, avem nevoie să cunoaștem trei aspecte: durata semnalului, frecvența sau pasul de eșantionare și expresia matematică ce descrie semnalul.

Spre exemplu, dacă dorim generarea și afișarea grafică a eșantioanelor unui semnal sinusoidal continuu descris de  $x(t) = 5 \cdot \sin\left(2 \cdot \pi \cdot t + \frac{\pi}{2}\right)$ , eșantionat cu frecvența de eșantionare  $f_s = 10$  Hz, pe o perioadă de 2 secunde, vom proceda astfel:

1. generăm vectorul de timp discret  $t$ ;
2. generăm eșantioanele semnalului  $x$  la momentele de timp descrise de vectorul  $t$ ;
3. afișăm grafic rezultatul.

### **main.py**

```

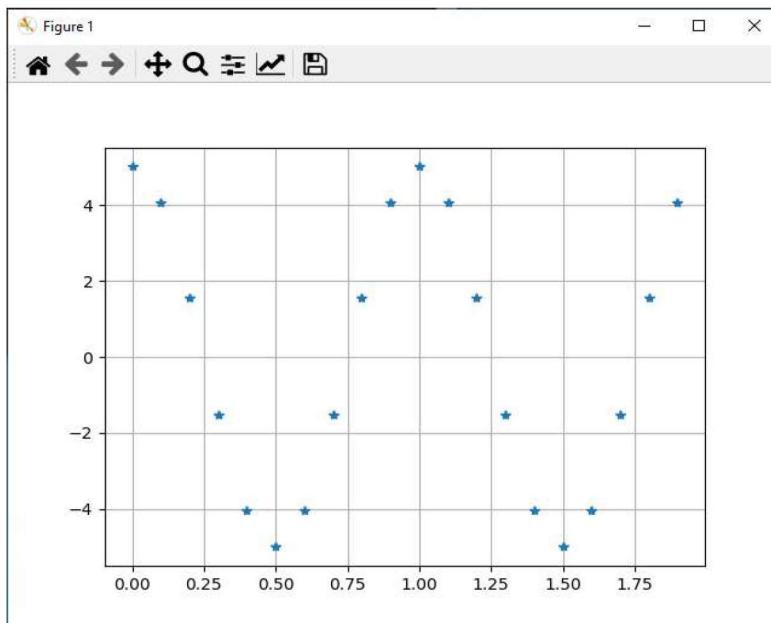
import numpy as np
import matplotlib.pyplot as plt

tf = 2
fs = 10

t = np.arange(0, tf, 1/fs)
x = 5 * np.sin(2 * np.pi * t + np.pi / 2)

plt.plot(t, x, "*")
plt.grid()
plt.show()

```

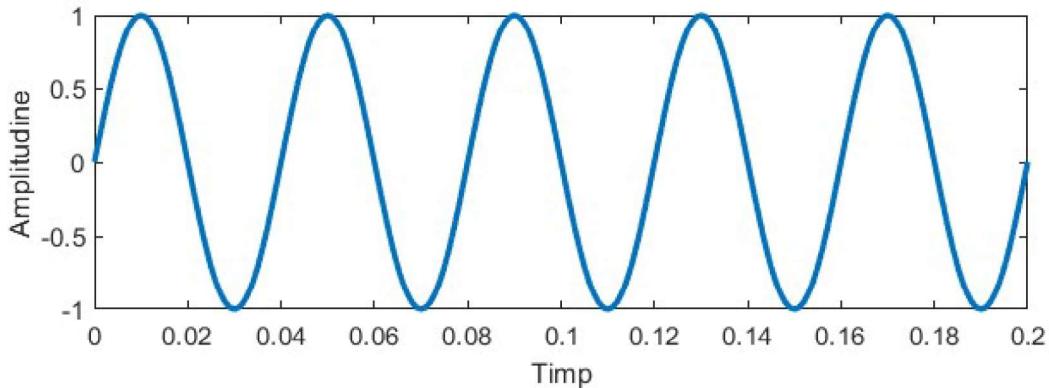


*Figura 2 – Exemplu de grafic afișat*

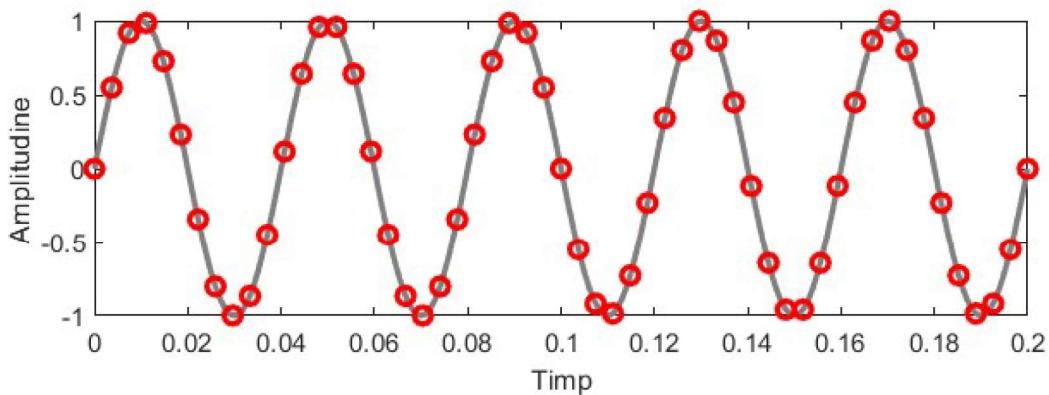
#### **5.1.2 Teorema eșantionării (Shannon)**

**Teorema eșantionării:** Pentru ca un semnal  $x(t)$  de spectru mărginit să poată fi reconstituit complet din eșantioanele sale, eșantionarea trebuie să respecte condiția Nyquist și anume, frecvența de eșantionare  $f_s$  trebuie să fie cel puțin dublul frecvenței maxime din spectrul semnalului:  $f_s \geq 2 \cdot f_{max}$  [8].

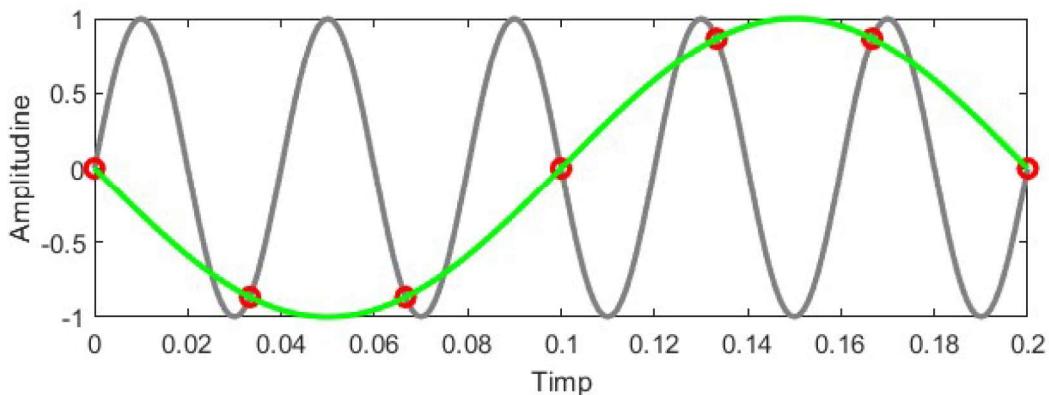
Exemplu: Fie un semnal sinusoidal  $x(t)$  având frecvență de  $25 \text{ Hz}$  (Figura 3). Semnalul a fost eșantionat cu o frecvență  $f_{s_1} = 270 \text{ Hz}$  (Figura 4) și cu o frecvență  $f_{s_2} = 30 \text{ Hz}$  (Figura 5).



*Figura 3 – Semnalul continuu  $x(t)$*



*Figura 4 – Semnalul  $x(t)$  eșantionat cu frecvența de eșantionare de  $270 \text{ Hz}$*



*Figura 5 – Semnalul  $x(t)$  eșantionat cu frecvența de eșantionare de  $30 \text{ Hz}$*

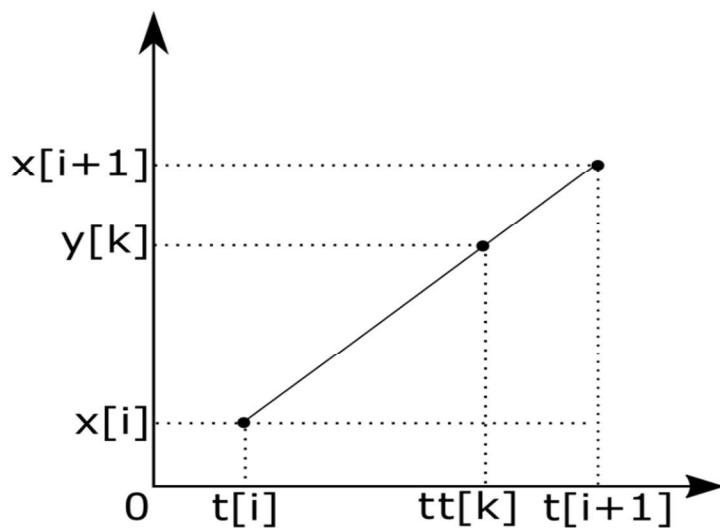
Se observă că în cazul  $f_{s_1} = 270 \text{ Hz}$  se respectă criteriul lui Nyquist ( $270 \text{ Hz} \geq 2 \cdot 25 \text{ Hz}$ ) și semnalul  $x(t)$  poate fi refăcut din eșantioanele sale. În cazul  $f_{s_2} = 30 \text{ Hz}$  nu se respectă criteriul lui Nyquist ( $30 \text{ Hz} \ngeq 2 \cdot 25 \text{ Hz}$ ) iar semnalul  $x(t)$  nu poate fi refăcut din eșantioanele sale (se observă că aceleași eșantioane rezultă atât din eșantionarea semnalului  $x(t)$ , cât și din eșantionarea unui alt semnal sinusoidal de frecvență 5 Hz – apare fenomenul de „aliere” – „aliasing”).

### **5.1.3 Reeșantionarea**

Schimbarea frecvenței de eșantionare (reeșantionare – resampling) de la  $f_{s_1}$  la  $f_{s_2}$  se realizează prin calcularea valorilor noilor eșantioane folosind o metodă de interpolare (liniară, pătratică, Lagrange etc). Trebuie avut grijă ca și noua frecvență de eșantionare să respecte criteriul Nyquist pentru a evita fenomenul de aliere.

#### **Interpolatorul liniar**

Având date două puncte  $(t[i], x[i])$  și  $(t[i + 1], x[i + 1])$ , interpolatorul liniar este dreapta ce trece prin aceste două puncte [9]. Pentru un  $tt[k]$  în intervalul  $[t[i], t[i + 1]]$  valoarea lui  $y[k]$  va fi dată de ecuația  $\frac{y[k]-x[i]}{x[i+1]-x[i]} = \frac{tt[k]-t[i]}{t[i+1]-t[i]}$  [9].



*Figura 6 – Interpolatorul liniar*

$$\text{Rezolvând ecuația, } y[k] = x[i] + (x[i + 1] - x[i]) \cdot \frac{tt[k] - t[i]}{t[i + 1] - t[i]}$$

Vom defini o funcție Python ce implementează metoda de interpolare liniară pentru creșterea frecvenței de eșantionare de la  $f_{s_1}$  la  $f_{s_2}$ . Funcția va avea ca intrări:

- $t$  – vectorul de timp al semnalului eșantionat cu  $f_{s_1}$
- $t = \text{np.arange}(0, tf, 1/fs1)$
- $x$  – eșantioanele semnalului eșantionat cu  $f_{s_1}$
- $tt$  – vectorul de timp al semnalului eșantionat cu  $f_{s_2}$
- $tt = \text{np.arange}(0, tf, 1/fs2)$

și ca ieșire  $y$  – eșantioanele semnalului eșantionat cu  $f_{s_2}$ .

### **main.py**

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

def interpolate_linear(t, x, tt):
    N = len(t)
    NN = len(tt)
    y = np.zeros((NN,))

    for k in range(0, NN):
        for i in range(0, N - 1):
            if (t[i] <= tt[k] and tt[k] <= t[i+1]) or i == N - 2:
                y[k] = x[i] + (x[i+1] - x[i]) / (t[i+1] - t[i]) * (tt[k] - t[i])
                break
    return y
```

Exemplu de utilizare a funcției implementate:

### **main.py**

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

def interpolate_linear(t, x, tt):
    ...
    # continutul functiei

tf = 2
fs1, fs2 = 10, 18

t = np.arange(0, tf, 1/fs1)
x = 5 * np.sin(2 * np.pi * t + np.pi / 2)

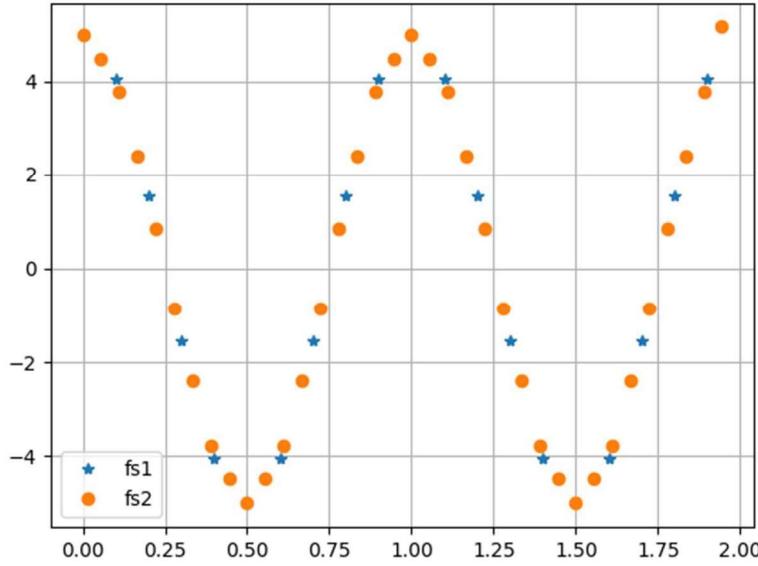
tt = np.arange(0, tf, 1/fs2)
```

```

y = interpolate_linear(t, x, tt)

plt.plot(t, x, "*", tt, y, 'o')
plt.grid()
plt.legend(["fs1", "fs2"])
plt.show()

```



*Figura 8 – Exemplu de grafic afișat (2)*

### Interpolatorul Lagrange

**Teorema interpolării:** Se dau  $N$  puncte distințte  $(t[0], x[0]) \dots (t[N - 1], x[N - 1])$  [9]. Există o ecuație polinomială de gradul maxim  $N - 1$  care interpolează punctele date. Ecuația de interpolare polinomială în forma Lagrange este se definește ca combinația liniară  $y[k] = \sum_{i=0}^{N-1} x[i] \cdot \ell(tt[k])$  de ecuații polinomiale Lagrange  $\ell(tt[k]) = \prod_{m=0, m \neq i}^{N-1} \frac{tt[k] - t[m]}{t[i] - t[m]}$  [9].

## 5.2 Cuantizarea semnalelor

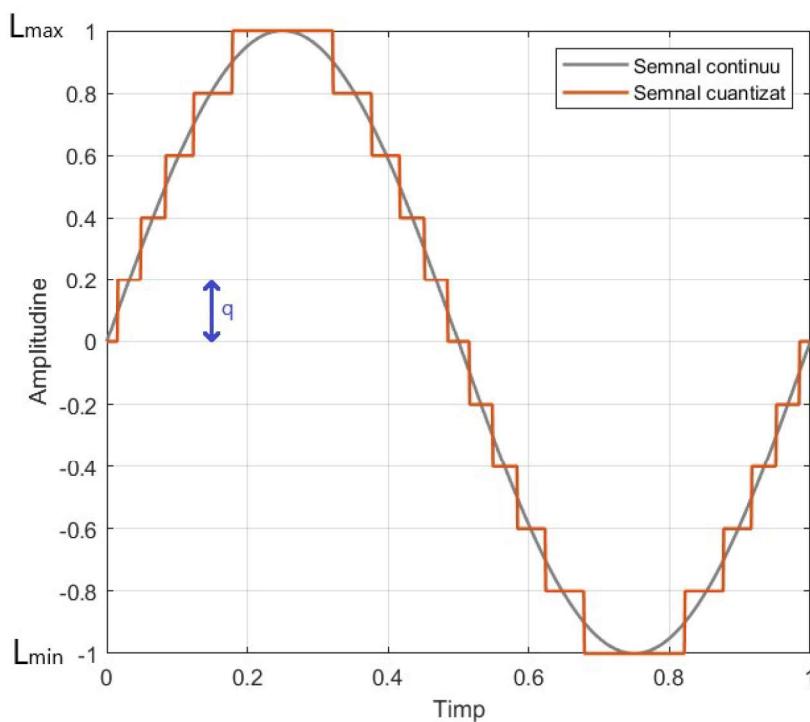
În urma eșantionării unui semnal analogic rezultă un semnal care este definit doar în anumite momente de timp a cărui amplitudine poate lua orice valoare reală din

domeniul de valori [3]. Totuși, întrucât sistemele digitale nu pot prelucra semnale cu amplitudine continuă, aceasta trebuie discretizată prin procesul de cuantizare.

Prin cuantizare fiecărui moment de timp  $i$  se alocă o valoare discretă dintr-un set finit de valori [3]. Distanța dintre două valori consecutive din setul de valori se numește pas de cuantizare [3].

*Tabel 3 – Cuantizarea semnalelor [3]*

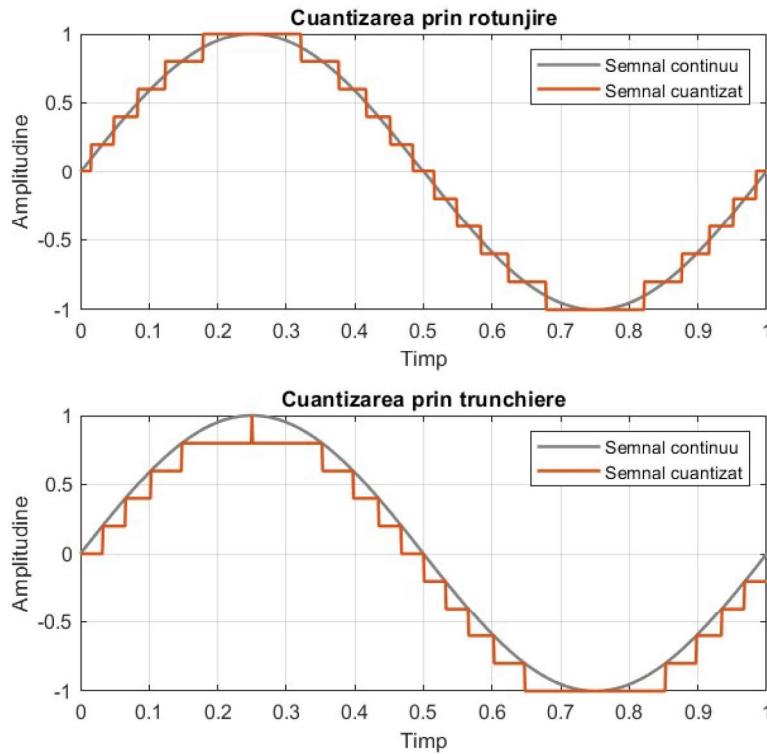
		Timp	
		Continuu	Discret
Amplitudine	Continuă	Semnale continue (semnale analogice)	Semnale discrete
	Discretă	Cuantizare	Cuantizare
		Semnale continue cuantificate	Semnale discrete cuantificate (semnale numerice)



*Figura 9 – Cuantizarea semnalelor*

Cele mai folosite metode de cuantizare sunt [3]:

1. Cuantizarea prin rotunjire – valoarea discretă va fi cel mai apropiat nivel de cuantizare disponibil de valoarea reală
2. Cuantizarea prin trunchiere – valoarea discretă va fi cel mai apropiat nivel de cuantizare disponibil valoric inferior valorii reale



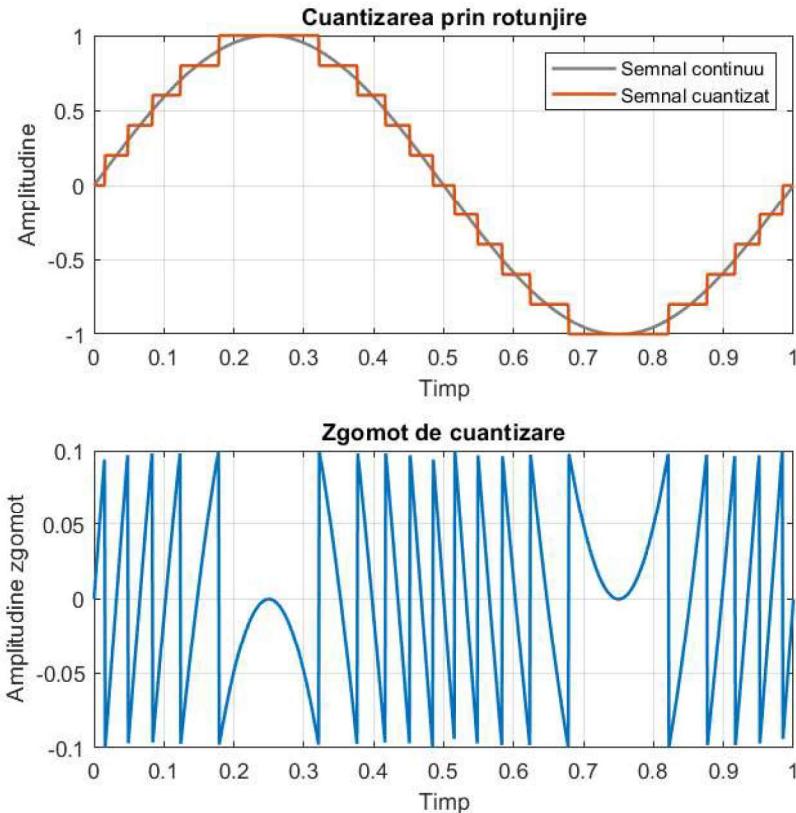
*Figura 10 – Metode de cuantizare*

**Zgomotul de cuantizare** este semnalul obținut făcând diferența dintre semnalul cuantizat și semnalul original [3]:  $\varepsilon[t] = x_c[t] - x[t]$ .

Pentru a evalua zgomotul de cuantizare vom folosi **eroarea pătratică medie** [3]:

$$\varepsilon_m = \frac{\sum_{t=1}^N \varepsilon[t]^2}{N}$$

Întrucât în urma cuantizării fiecărui eșantion al semnalului i se atribuie o valoare din setul finit de valori ai nivelelor de cuantizare, pentru a descrie un eșantion este suficientă dacă se precizează nivelul de cuantizare corespunzător.



*Figura 11 – Zgomotul de cuantizare*

Exemplu: Se dau următoarele niveluri de cuantizare:  $\{0, 0.5, 1, 1.5\}$ :

- Nivel de cuantizare 0 = 0
- Nivel de cuantizare 1 = 0.5
- Nivel de cuantizare 2 = 1
- Nivel de cuantizare 3 = 1.5

Valoare analogică $x(t)$	Valoare cuantizată prin rotunjire	Nivel de cuantizare	Nivel de cuantizare (binar)
$x(t) \leq 0.25$	0	Nivel 0	00
$0.25 < x(t) \leq 0.75$	0.5	Nivel 1	01
$0.75 < x(t) \leq 1.25$	1	Nivel 2	10
$1.25 < x(t)$	1.5	Nivel 3	11

Se poate observa în tabelul anterior faptul că nivelul de cuantizare poate fi reprezentat cu un număr mai mic de biți (2 biți în exemplu) astfel încât întregul semnal poate fi stocat într-un spațiu mic de memorie.

Pentru a calcula numărul minim de biți  $b$  prin care putem reprezenta nivelul de cuantizare putem folosi formula  $b = \lceil \log_2 N_q \rceil$  unde  $N_q$  reprezintă numărul de niveluri de cuantizare [3].

### 5.3 Exerciții

1. Reprezentați grafic în *Python* eșantioanele unui semnal sinusoidal de frecvență 25 Hz, defazaj  $\frac{\pi}{6}$  și amplitudine 1 pe un interval de timp de 0.1 secunde folosind o frecvență de eșantionare de 1 kHz. Reprezentați pe același grafic, dar cu altă culoare, același semnal eșantionat cu o frecvență de eșantionare de 40 Hz.
2. Se dă eșantioanele unui semnal sinusoidal  $x(t) = \sin(2 \cdot \pi \cdot t)$  eșantionat pe o perioadă de 1 secundă ( $t_f = 1$ ) cu o frecvență de eșantionare  $f_{s_1} = 5$  Hz. Pornind de la eșantioanele date, reeșantionați semnalul la frecvența de eșantionare  $f_{s_2} = 20$  Hz folosind funcția de interpolare liniară. Afipați grafic semnalul original și semnalul reeșantionat.
3. Implementați în *Python* o funcție pentru interpolatorul Lagrange și folosiți-o pentru reeșantionarea semnalului de la exercițiul anterior.
4. Reprezentați grafic eșantioanele unui semnal sinusoidal  $x(t) = 10 \cdot \sin(2 \cdot \pi \cdot t \cdot 5 + \pi/3)$  eșantionat cu frecvență de eșantionare de 1 kHz pe o perioadă de 0.25 secunde.
  - a. Cuantizați semnalul folosind un pas de cuantizare  $q = 2$  prin rotunjire și afipați grafic.
  - b. Cuantizați semnalul folosind un pas de cuantizare  $q = 2$  prin trunchiere și afipați grafic.
  - c. Reprezentați grafic zgomotele de cuantizare pentru semnalele cuantizate și comparați erorile pătratice medii.