Procesarea Semnalelor

Laboratorul 1

Introducere

1 Semnale continue și semnale discrete

Majoritatea semnalelor ce descriu fenomene fizice (mecanice, optice, electrice, chimice, etc) sunt semnale continue și pot fi reprezentate matematic ca o funcție $x_a : \mathbb{R} \to \mathbb{R}$. Valoarea semnalului la momentul t este $x_a(t)$.

Eșantionare. Pentru a putea fi prelucrate numeric, este nevoie ca aceste semnale să fie discretizate. De regulă, variația mărimii fizice măsurate de elementul sensibil al senzorului este întâi tradusă în variația unei alte mărimi fizice mai convenabilă prelucrării, de cele mai multe ori electrică. Apoi, semnalul este eșantionat, adică transformat într-un semnal discret, prin înregistrarea valorilor acestuia la momente distincte de timp.

Un semnal discret este o funcție $x : \mathbb{Z} \to \mathbb{R}$. Relația dintre semnalul continuu și cel discretizat este $x[n] = x_a(nT)$, unde T se numește perioada de eșantionare. Frecvența de eșantionare, $f_s = \frac{2\pi}{T}$ măsoară numărul de eșantioane pe secundă și are unitatea de măsură Hz.

Figura 1 prezintă câteva tipuri de semnale sintetice de forme diferite (*waveform* în engleză) utilizate des.

2 Ghid Python

Pentru a rezolva exercițiile din laboratorul de astăzi, aveți nevoie de biblioteca `numpy` și modulul `matplotlib.pyplot`.

Funcții utilizate:

```
numpy.pi
numpy.cos(), numpy.sin()
numpy.floor, numpy.mod, numpy.sign
numpy.random.rand(d0,d1,...,dn)
numpy.ones(shape), numpy.zeros(shape)
numpy.linspace(start, end, no_samples)
numpy.round(number)
```

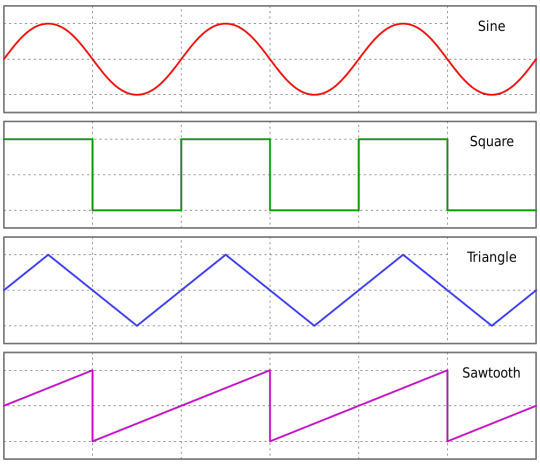

Figure 1: Semnale sintetice

Valori discrete se pot afișa grafic folosind funcția `stem()`. Dacă ați importat modulul `matplotlib.pyplot`cu numele `plt`, sintaxa acesteia este `plt.stem(x,y)`.

Pentru a genera un grafic cu n subplot-uri, utilizați următoarea secvență:

```
fig, axs = plt.subplots(n)
fig.suptitle('Titlu principal')
axs[0].plot(x0,y0)
axs[1].plot(x1,y1)
...
```

Dacă doriți să setați aceași proprietate pentru toate subploturile (spre exemplu limitele axelor sau eticheta lor),

```
for ax in axs.flat:
ax.set_xlim([xmin, xmax])
```

3 Exerciții

1. Fie semnalele continue $x(t) = \cos(520\pi t + \pi/3)$, $y(t) = \cos(280\pi t - \pi/3)$ și $z(t) = \cos(120\pi t + \pi/3)$.
   (a) În Python, simulați axa reală de timp printr-un șir de numere suficient de apropiate, spre exemplu $[0 : 0.0005 : 0.03]$.
   (b) Construiți semnalele $x(t)$, $y(t)$ și $z(t)$ și afișați-le grafic, în cate un subplot.
   (c) Eșantionați semnalele cu o frecvență de 200 Hz pentru a obține $x[n]$, $y[n]$ și $z[n]$ și afișați-le grafic, în câte un subplot.

2. Generați următoarele semnale și afișați-le grafic, fiecare într-un plot:

   (a) Un semnal sinusoidal de frecvență 400 Hz, care să conțină 1600 de eșantioane.

   (b) Un semnal sinusoidal de frecvență 800 Hz, care să dureze 3 secunde.

   (c) Un semnal de tip **sawtooth** de frecvență 240 Hz (puteți folosi funcțiile `numpy.floor` sau `numpy.mod`).

   (d) Un semnal de tip **square** de frecvență 300 Hz (puteți folosi funcția `numpy.sign`).

   (e) Un semnal 2D aleator. Creați un `numpy.array` de dimensiune 128x128 și inițializați-l aleator, folosind `numpy.random.rand(x,y)`, unde x și y reprezintă numărul de linii respectiv de coloane. Afișați semnalul generat folosind funcția `imshow(I)` din `matplotlib`.

   (f) Un semnal 2D la alegerea voastră. Creați un `numpy.array` de dimensiune 128x128 și inițializați-l folosind o procedură creată de voi. Utilizați, spre exemplu, funcțiile `numpy.zeros()` și `numpy.ones()`.

3. Un semnal este digitizat cu o frecvență de eșantionare de 2000 Hz.

   (a) Care este intervalul de timp între două eșantioane?

   (b) Dacă un eșantion este memorat pe 4 biți, câți bytes vor ocupa 1 oră de achiziție?

Surse imagini

Figura 1: `https://en.wikipedia.org/wiki/Sawtooth_wave#/media/File:Waveforms.svg`