Sinais e Sistemas - SS

LAB001 - Python em SS:

Vamos usar a Linguagem Python para visualizar e compreender os conceitos de SS

Bibliotecas de Python::

- Numpy e Scipy - https://scipy.org/
- Matplotlib - https://matplotlib.org/

Vamos representar os sinais no Python através de vetores (arrays):

Representação de um sinal discreto no tempo:

```
import numpy as np
n =  np.arange(0,11,1)
yD = np.sin(n*3.1415/8)
```

Cria um vetor *n* = [0,1,2,3,4,5,6,7,8,9,10] e um sinal senoidal discreto no tempo *yD*.

Representação de um sinal contínuo no tempo:

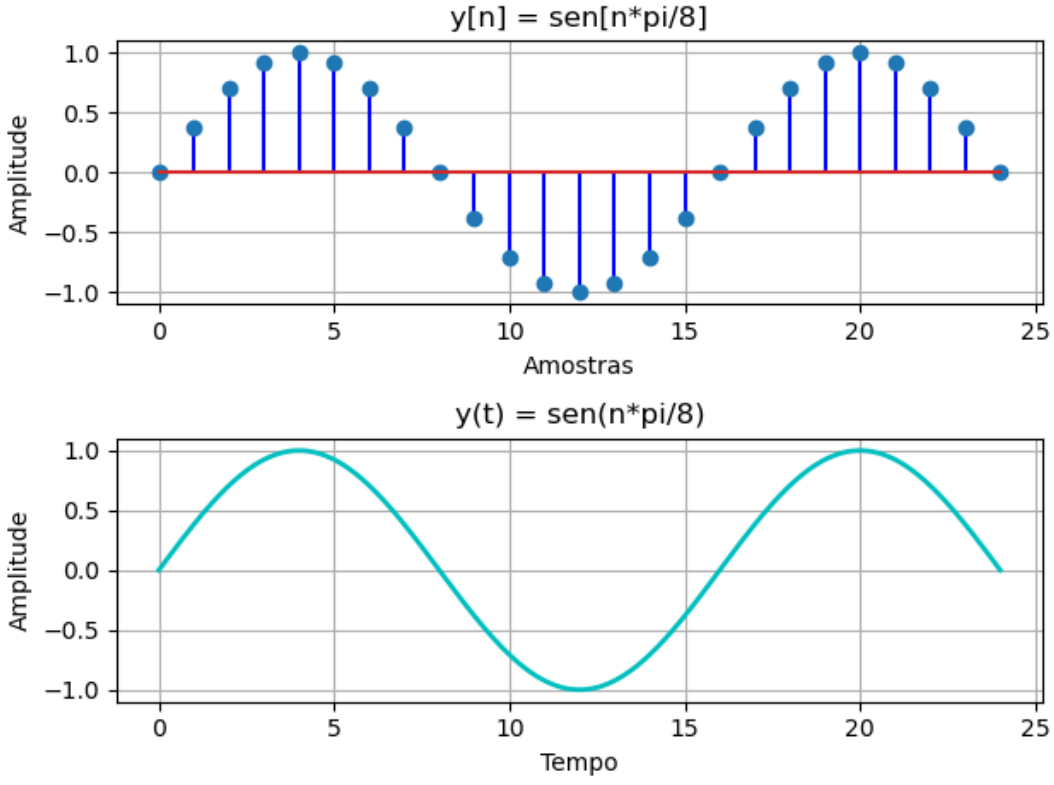
```
import numpy as np
t = np.arange(0, 2*3.1415, 0.00001)
yC = np.sin(t*3.1415/8)
```

Cria um vetor *t* = [0, 0.00001, 0.00002, 0.00003,...,((2*3.1415)-0.0001)] e um sinal senoidal *contínuo* no tempo *yC*. É um falso *contínuo* porque o sinal *yC* continua um sinal discreto porem com muitos pontos entre os valores inteiros de *t*. Nós vamos utilizar esta representação para os números contínuos nos nossos estudos porem com muito cuidado e sempre lembrando que ainda são números discretos.

Para visualizar os sinais criados na forma de gráficos vamos usar a biblioteca Matplotlib.

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

n = np.arange(25)
t = np.arange(0, 24, 0.0001)

yD = np.sin(n*pi/8)
yC = np.sin(t*pi/8)

#criando os gráficos
fig, ax = plt.subplots(2,1)

ax[0].stem(n, yD, linefmt='b-',use_line_collection=True)
ax[0].set_xlabel("Amostras")
ax[0].set_ylabel("Amplitude")
ax[0].grid(True)
ax[0].set_title('y[n] = sen[n*pi/8]')

ax[1].plot(t, yC, 'c-', linewidth=2, label="função seno Continua")
ax[1].set_xlabel("Tempo")
ax[1].set_ylabel("Amplitude")
ax[1].grid(True)
ax[1].set_title('y(t) = sen(n*pi/8)')

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

Produz o gráfico da figura abaixo:



Podemos criar os sinais de outros modos:

```
import numpy as np
a = np.array( [2, 3, 4, 5.01, 7.304, 45, 334])
```

```
a
>> array([  2.   ,   3.   ,   4.   ,   5.01 ,   7.304,  45.   , 334.   ])
```

Exemplos para números complexos:

```
import numpy as np
a = numpy.array( [ [2,3], [4,5] ], dtype=complex)
c = numpy.array ( [2.+1.j, 2, 4 ] )
```

```
>> a
array([[2.+0.j, 3.+0.j],
 [4.+0.j, 5.+0.j]])
>> c
array([2.+1.j , 2.+0.j , 4.+0.j])
```

Exercícios:

1. Crie uma função em Python que retorne a parte Par e a parte Impar de um sinal qualquer:

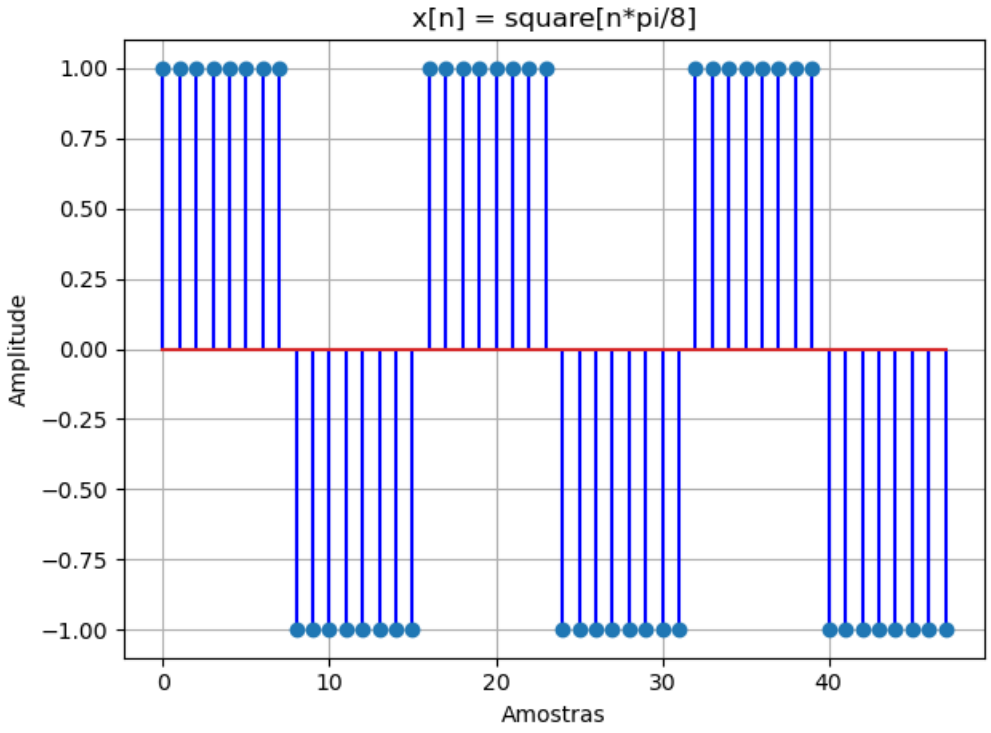
   Modelo da função:
   p,i = separa_ParImpar(x,n)
   - Entrada:
     - x – sinal de entrada
     - n – vetor de tempo do sinal de entrada
   - Saída:
     - p – parte par do sinal
     - i – parte impar do sinal

   Apresente um programa que teste todas as letras abaixo:

   a. s_1 = [2, 1, 0, 1, 2], n = [-2, -1, 0, 1, 2]
   b. s_2 = [-2,-1, 0, 1, 2], n = [-2, -1, 0, 1, 2]
   c. s_3 = [0, 0 ,0 , 2, 4], n = [-2, -1, 0, 1, 2]
   d. s_4 = [0 , -1 , -1, 3 , 2], n = [-2, -1, 0, 1, 2]
   e. s_5 = [0, 0 ,0 , 2, 4], n = [0, 1, 2, 3, 4]

2. Crie os sinais:
   a. $x_1[n]$, em que $x_1[n] = 2\cos\left(\frac{\pi}{4}n\right)$ e plote o sinal no intervalo de -10 a 10.
   b. $x_2(t)$, em que $x_2(t) = 2\cos\left(\frac{\pi}{4}t\right)$ e plote o sinal no intervalo de -10 a 10, com passos de 0.001.

3. Utilizando a função ***square*** da biblioteca “scipy.signal”, crie uma sinal de uma onda quadrada igual da figura abaixo:



4. Sendo
   - $x_1[n] = \cos\left(\frac{\pi}{6}n\right)$;
   - $x_2[n] = \begin{cases} \cos\left(\frac{\pi}{6}n\right), -10 \leq n \leq 20 \\ 0, caso\ contrario \end{cases}$
   - $u[n]$ uma função degrau unitário;

   Plote no intervalo de -10 a 20:

   a. $y_1[n] = x_1[n] + 0.9x_1[n-4]$
   b. $y_2[n] = x_2[n] + 0.9x_2[n-4]$

   Note que $y_1[n]$ é diferente de $y_2[n]$.

   c. $y_3[n] = x_1[-n]u[n+3]$
   d. $y_4[n] = x_2[-n]u[n+3]$

   Note que $y_3[n]$ é diferente de $y_4[n]$.

   e. $y_5[n] = x_1[n]u[n-5]$
   f. $y_6[n] = x_1[n]\{u[n-5] - u[n-10]\}$
   g. $y_7[n] = x_1[n] + u[n+1]$