parimpar

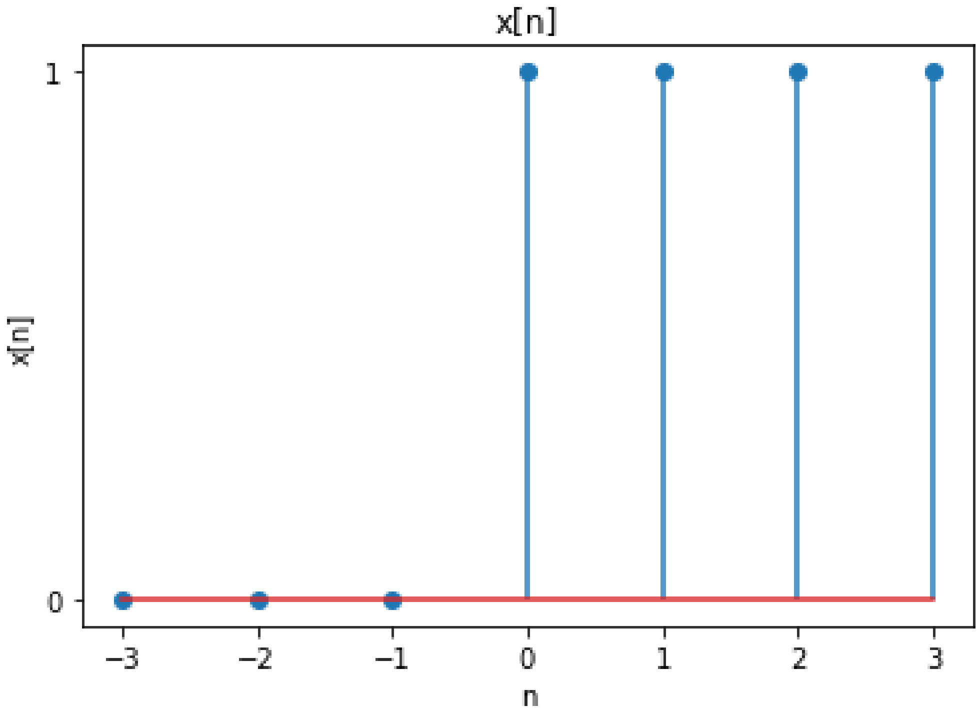
May 14, 2022

[1]:
```python
######Trabalho de Sinais e Sistemas#####

#Professor: André Braga

#Aluno: Antonio César de Andrade Júnior
#Matrícula: 473444

#Descrição: Recebe um sinal e o decompõe em uma parte par e outra ímpar, depois␣
 ↪faz a soma entre as duas partes.
#OBS: o código foi feito considerando que o intervalo do vetor n é simétrico (Ex:
 ↪ n_inicial = -7, n_final = 7)


import numpy as np
import matplotlib.pyplot as plt

#gerador do sinal x[n] = 1 para n>=1 e x[n] = 0 para n<0
def sinal1(n):
    x =[]
    for sample in n:
        if sample<0:
            x.append(0)
        else:
            x.append(1)
    return(x)

#gerador do sinal x[n] = -1 para n = -4 ou 3, x[n] = 2 para n = -3 ou -2 ou 1,␣
 ↪x[n] = 1 para n = -1 ou 0 ou 2, se não x[n] = 0
def sinal2(n):
    x =[]
    for sample in n:
        if sample == -4 or sample == 3:
            x.append(-1)
        elif sample == -3 or sample == -2 or sample == 1:
            x.append(2)
        elif sample == -1 or sample == 0 or sample == 2:
            x.append(1)
```

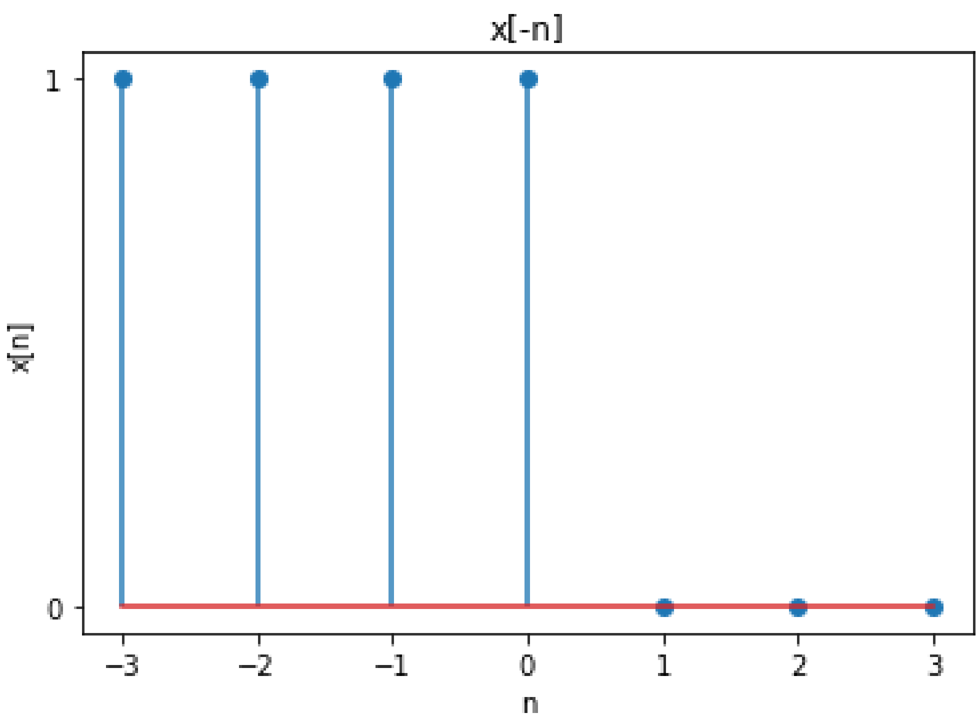
```
        else:
            x.append(0)
    return(x)


def parimpar(x, n0):
    print("n0 = ", n0)

    #plotar o sinal x[n] gerado. O motivo do ponto final ser -n0+1 é np.arange␣
 ↪gerar um array com o ponto final
    #uma posição a menos do valor passado como parâmetro, assim, para o array␣
 ↪ser simétrico, é preciso passar o parâmetro somado
    #a 1.
    print("x[n] =",x)
    n = np.arange(n0, -n0+1, 1)
    plt.stem(n, x)
    plt.xlabel('n')
    plt.xticks(np.arange(n0, -n0+1, 1))
    plt.yticks([0, 1])
    plt.ylabel('x[n]')
    plt.title('x[n]')
    plt.show()
    plt.savefig('x_exemplo.png')
    plt.clf()


    #reverte no tempo o sinal gerado usando a função np.flip()
    x_reverso = np.flip(x)
    print("x[-n] =",x_reverso)
    plt.stem(n, x_reverso)
    plt.xlabel('n')
    plt.xticks(np.arange(n0, -n0+1, 1))
    plt.yticks([0, 1])
    plt.ylabel('x[n]')
    plt.title('x[-n]')
    plt.show()
    plt.savefig('x_r_exemplo.png')
    plt.clf()


    #gera a parte par do sinal x[n] utilizando a fórmula xp[n] = (x[n]+x[-n])/2.␣
 ↪Para isso, é utilizada a função zip que cria
    #uma tupla para cada par de valores.
    xp = []
    for A, B in zip(x, x_reverso):
        xp.append((A+B)/2)
    print("xp[n] =",xp)
```

```python
    plt.stem(n, xp)
    plt.xlabel('n')
    plt.xticks(np.arange(n0, -n0+1, 1))
    plt.yticks([0, 1])
    plt.ylabel('x[n]')
    plt.title('xp[n]')
    plt.show()
    plt.savefig('par_exemplo.png')
    plt.clf()

    #gera a parte ímpar do sinal x[n] utilizando a fórmula xp[n] = (x[n]-x[-n])/
 ↪2.
    xi = []
    for A, B in zip(x, x_reverso):
        xi.append((A-B)/2)
    print("xi[n] =",xi)
    plt.stem(n, xi)
    plt.xlabel('n')
    plt.xticks(np.arange(n0, -n0+1, 1))
    plt.yticks([0, 1])
    plt.ylabel('x[n]')
    plt.title('xi[n]')
    plt.show()
    plt.savefig('impar_exemplo.png')
    plt.clf()


    #gera a soma das parte par r ímpar do sinal x[n].
    soma = []
    for A, B in zip(xp, xi):
        soma.append((A+B))
    print("soma[n] =",soma)
    plt.stem(n, soma)
    plt.xlabel('n')
    plt.xticks(np.arange(n0, -n0+1, 1))
    plt.yticks([0, 1])
    plt.ylabel('x[n]')
    plt.title('xp+xi')
    plt.savefig('soma_exemplo.png')

n0 = -3
n = np.arange(n0, -n0+1, 1)
x = sinal1(n)

parimpar(x, n0)
```

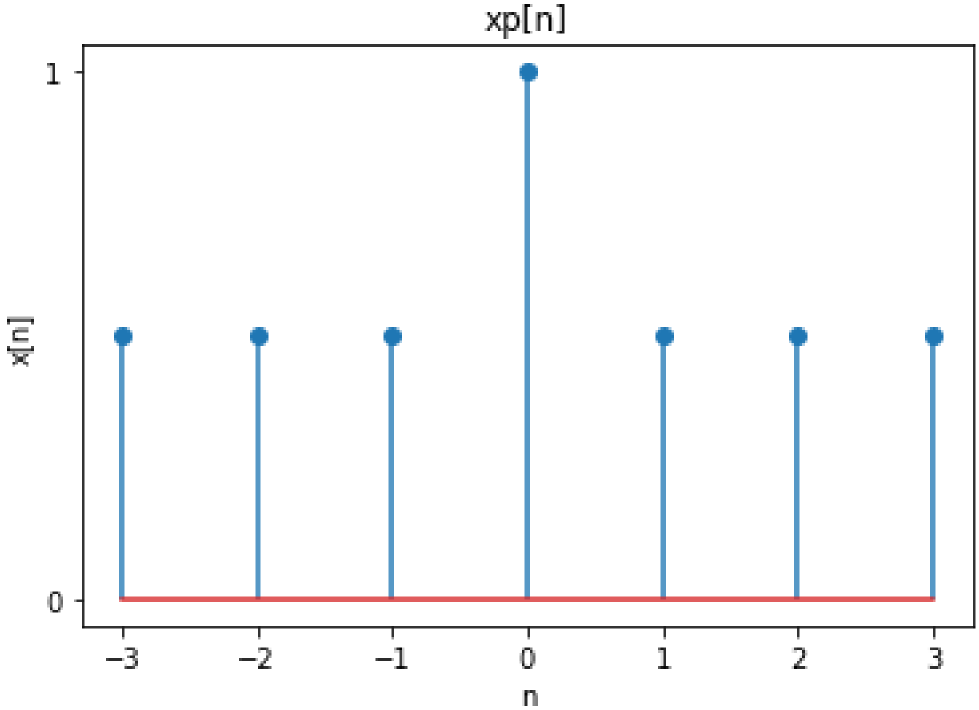
```
n0 =  -3
x[n] = [0, 0, 0, 1, 1, 1, 1]
```



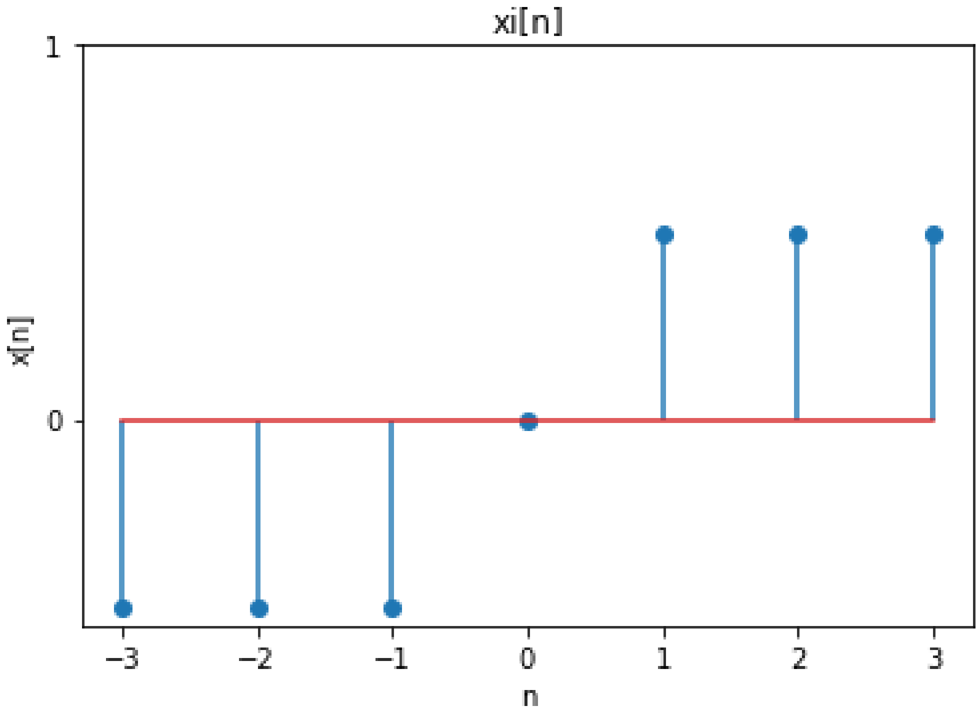

```
x[-n] = [1 1 1 1 0 0 0]
```



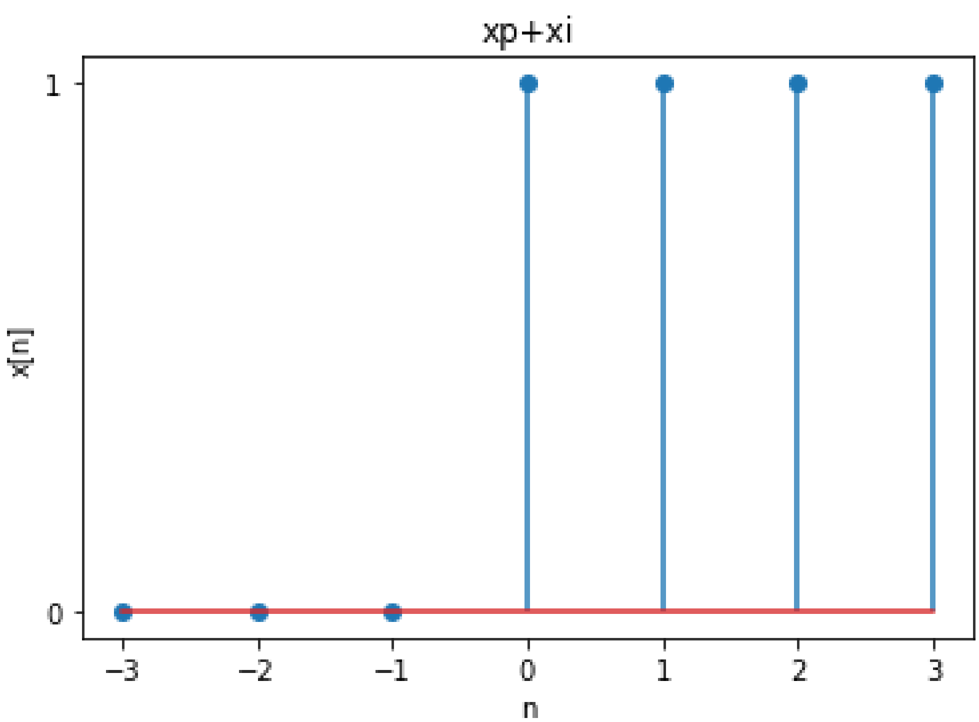
```
xp[n] = [0.5, 0.5, 0.5, 1.0, 0.5, 0.5, 0.5]
```




```
xi[n] = [-0.5, -0.5, -0.5, 0.0, 0.5, 0.5, 0.5]
```

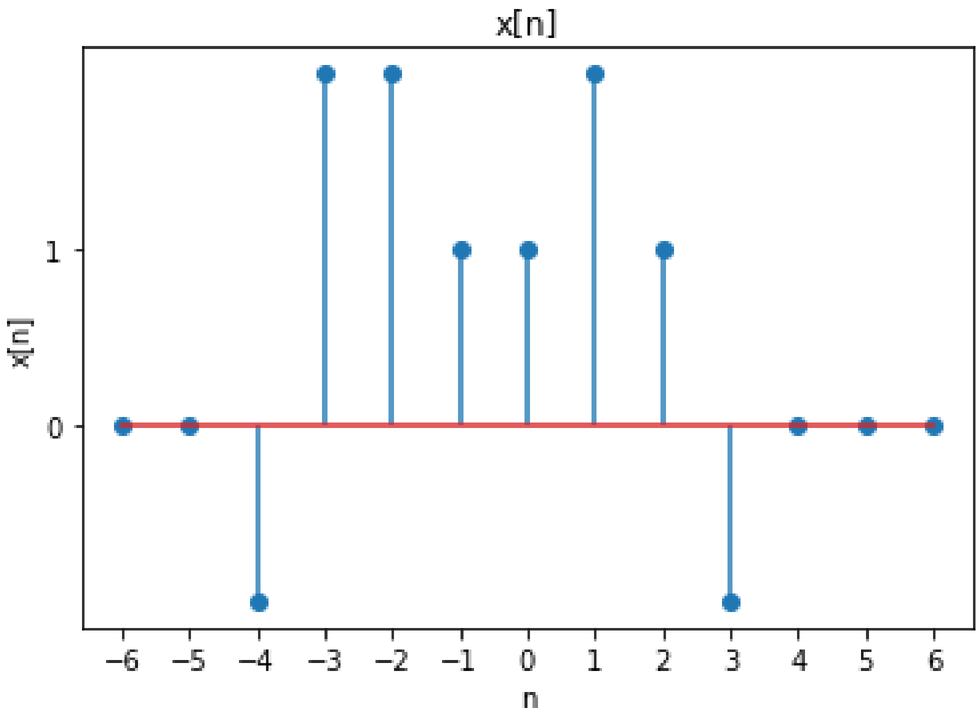



```
soma[n] = [0.0, 0.0, 0.0, 1.0, 1.0, 1.0, 1.0]
```

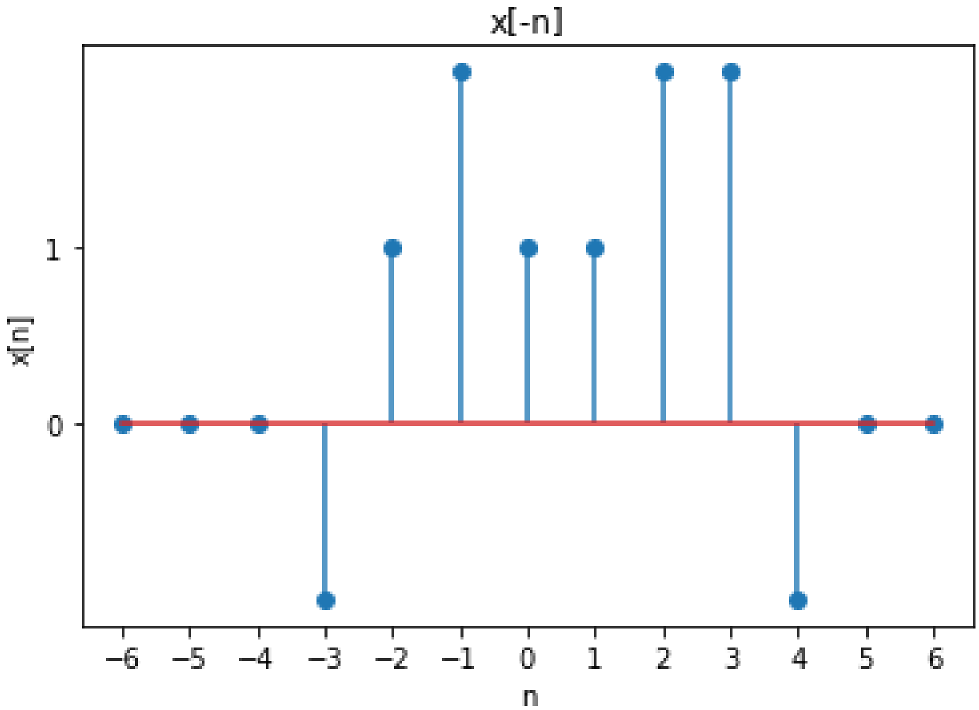


[2]:
```python
n0 = -6
n = np.arange(n0, -n0+1, 1)
x = sinal2(n)

parimpar(x, n0)
```

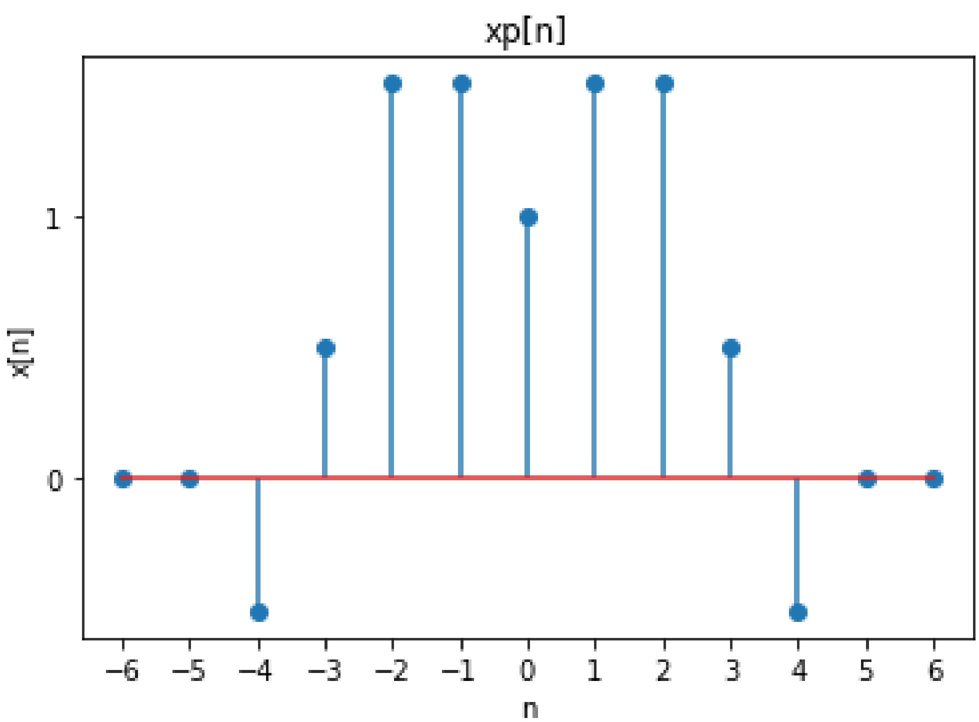
```
n0 =  -6
x[n] = [0, 0, -1, 2, 2, 1, 1, 2, 1, -1, 0, 0, 0]
```



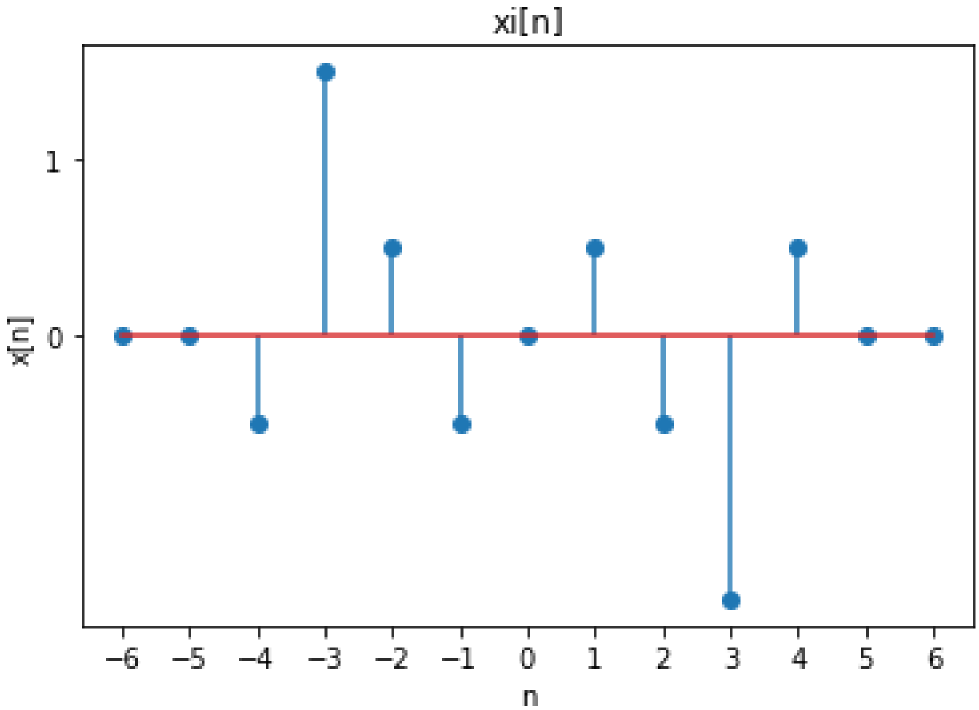

```
x[-n] = [ 0  0  0 -1  1  2  1  1  2  2 -1  0  0]
```



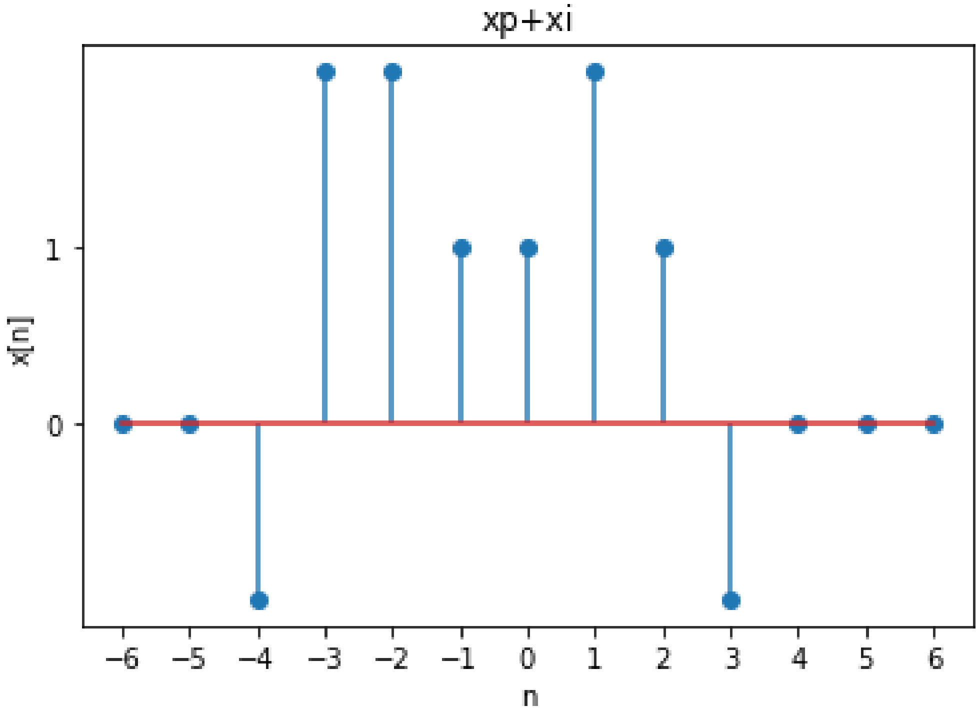

```
xp[n] = [0.0, 0.0, -0.5, 0.5, 1.5, 1.5, 1.0, 1.5, 1.5, 0.5, -0.5, 0.0, 0.0]
```



```
xi[n] = [0.0, 0.0, -0.5, 1.5, 0.5, -0.5, 0.0, 0.5, -0.5, -1.5, 0.5, 0.0, 0.0]
```




```
soma[n] = [0.0, 0.0, -1.0, 2.0, 2.0, 1.0, 1.0, 2.0, 1.0, -1.0, 0.0, 0.0, 0.0]
```


xp+xi
x[n]
n
1
0
−6
−5
−4
−3
−2
−1
0
1
2
3
4
5
6