Convolução

May 14, 2022

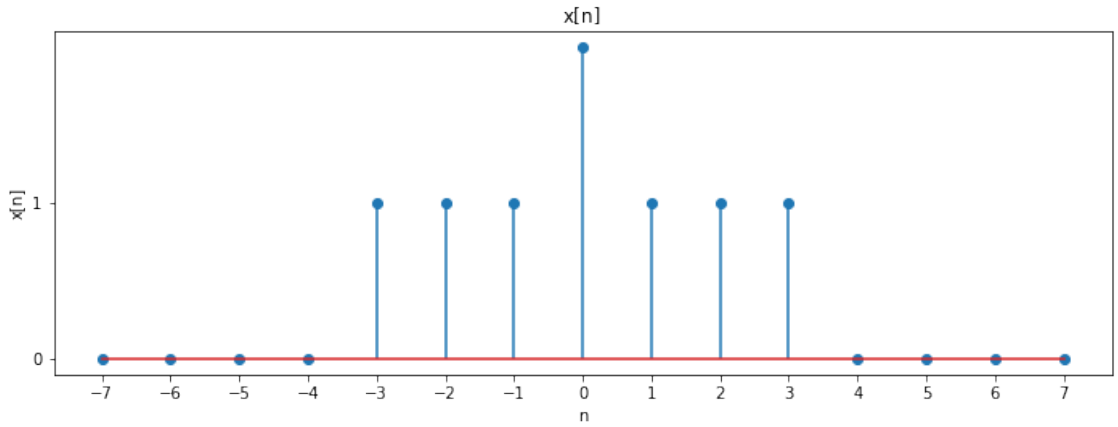
[1]:
```
######Trabalho de Sinais e Sistemas#####

#Professor: André Braga

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

#Descrição: Recebe um sinal x e um sinal h e faz convolução entre os dois.
#OBS: o código foi feito considerando que o intervalo do vetor n é simétrico (Ex:
 ↪ n_inicial = -7, n_final = 7)
#Referência: https://www.geeksforgeeks.org/linear-convolution-using-c-and-matlab/


import numpy as np
import matplotlib.pyplot as plt

#gerador do sinal x[n]=2 para n=0, x[n]=1 para n>=-3 e n<=3 e x[n]=0 para os␣
 ↪demais n's
def sinalx1(n):
    x =[]
    for sample in n:
        if sample==0:
            x.append(2)
        elif sample>=-3 and sample<=3:
            x.append(1)
        else:
            x.append(0)
    return(x)

#gerador do sinal x[n]=1 para n>=0 e n<=4 e x[n]=0 para os demais casos
def sinalx2(n):
    x=[]
    for sample in n:
        if sample>=0 and sample<=4:
            x.append(1)
        else:
            x.append(0)
```

```
    return x

#gerador do sinal h[n]=1 para n=1 ou n=-1, h[n]=2 para n=0 e h[n]=0 para os␣
 ↪demais casos
def sinalh1(n):
    x=[]
    for sample in n:
        if sample == 1 or sample == -1:
            x.append(1)
        elif sample == 0:
            x.append(2)
        else:
            x.append(0)

    return x

#gerador do sinal h[n]=1 para n>=2 e n<=7 ou n>=11 e n<=16, e h[n]=0 para os␣
 ↪demais casos
def sinalh2(n):
    x=[]
    for sample in n:
        if (sample>=2 and sample<=7) or (sample>=11 and sample<=16):
            x.append(1)
        else:
            x.append(0)

    return x

#faz deslocamento de n a partir de um determinado valor k, por exemplo, com k=-1␣
 ↪o sinal será adiantado
def deslocamento(k, n):
    for i in range(len(n)):
        n[i]=n[i]+k
    return n

#faz a soma entre os termos da convolução (y=+x[k]*h[n-k]), em que x e h são os␣
 ↪sinais originais, n é o vetor gerado a partir
#de n0 e h_temp é o n deslocado com k.
def mult(h,x,n,h_temp):
    n = list(n)
    h_temp = list(h_temp)
    y = 0
    for A,B in zip(x,n):
        if B in h_temp:
            indexBn = n.index(B)
            indexBh_temp = h_temp.index(B)
```

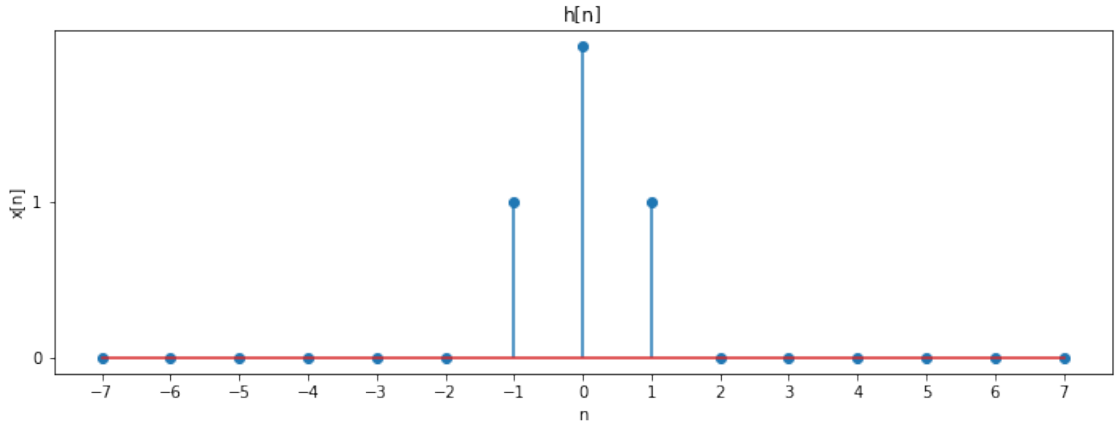
```
            y += x[indexBn]*h[indexBh_temp]

    return y

#função que gera a convolução entre os sinais x e h, em que  n0 é o ponto␣
 ↪inicial dos sinais.
def convolucao(x,h,n0):
    print("n0 = ", n0)

    # 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.
    n = np.arange(n0, -n0+1, 1)
    h_temp = []
    y = []
    k = 0

    #dobra otamanho do vetor, já que a convolução pode ter pontos deferentes de␣
 ↪0 que não aparecem em x e h
    n_begin = 2*n[0]
    n_end = 2*n[-1]
    ny = np.arange(n_begin, n_end+1, 1)

    #reverte o sinal h no tempo, como n é simétrico, então só é preciso fazer␣
 ↪uma negação do mesmo para revertê-lo
    h_reverso = np.flip(h)
    n_neg = np.negative(n)
    n_neg.sort()

    #pega os índices do sinal x (k) para que seja feito o deslocamento do h␣
 ↪revertido (h[n-k]), depois faz as somas sos termos
    #do somatório
    for i in ny:
        k = i
        h_temp = deslocamento(k, n_neg.copy())
        y.append(mult(h_reverso,x,n,h_temp))


    print("x[n]=", x)
    cm = 1/2.54
    fig, ax = plt.subplots(figsize=(30*cm, 10*cm))
    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()

    print("h[n]=", h)
    cm = 1/2.54
    fig, ax = plt.subplots(figsize=(30*cm, 10*cm))
    plt.stem(n, h)
    plt.xlabel('n')
    plt.xticks(np.arange(n0, -n0 + 1, 1))
    plt.yticks([0, 1])
    plt.ylabel('x[n]')
    plt.title('h[n]')
    plt.show()
    plt.savefig('h_exemplo.png')
    plt.clf

    print("x[n]*h[n]=", y)
    cm = 1/2.54
    fig, ax = plt.subplots(figsize=(50*cm, 10*cm))
    plt.stem(ny, y)
    plt.xlabel('n')
    plt.xticks(ny)
    plt.ylabel('x[n]')
    plt.title('x[n]*h[n]')
    plt.savefig('xh_exemplo.png')



n0 = -7
n = np.arange(n0, -n0 + 1, 1)
x = sinalx1(n)
h = sinalh1(n)

convolucao(x,h,n0)
```

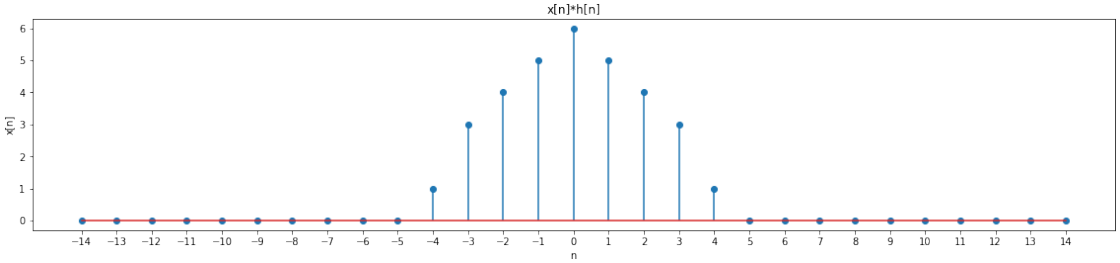
```
n0 =  -7
x[n]= [0, 0, 0, 0, 1, 1, 1, 2, 1, 1, 1, 0, 0, 0, 0]
```




```
h[n]= [0, 0, 0, 0, 0, 0, 1, 2, 1, 0, 0, 0, 0, 0, 0]

<Figure size 432x288 with 0 Axes>
```

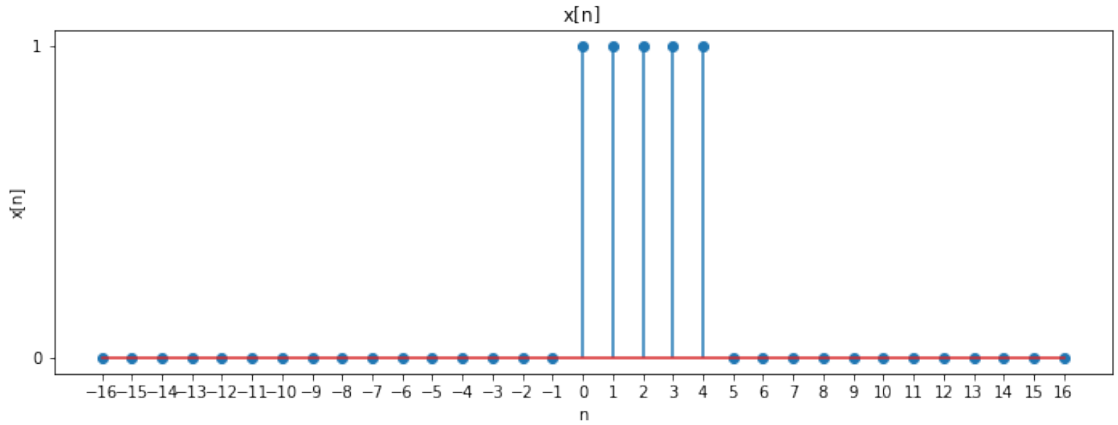



```
x[n]*h[n]= [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 3, 4, 5, 6, 5, 4, 3, 1, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0]

<Figure size 432x288 with 0 Axes>
```



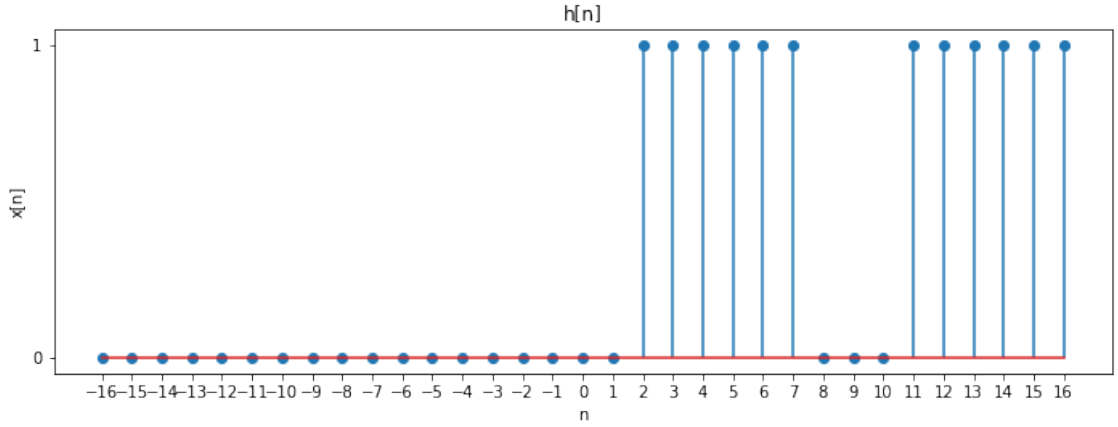
[2]:
```
n0 = -16
n = np.arange(n0, -n0 + 1, 1)
x = sinalx2(n)
h = sinalh2(n)

convolucao(x,h,n0)
```

```
n0 =  -16
x[n]= [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 1, 1, 1, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0]
```




```
h[n]= [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 1, 1, 1, 1,
0, 0, 0, 1, 1, 1, 1, 1, 1]
```

```
<Figure size 432x288 with 0 Axes>
```



```
x[n]*h[n]= [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 2, 3, 4, 5, 5, 4, 3, 2, 2, 2, 3, 4, 5, 5, 4,
3, 2, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0]
```

```
<Figure size 432x288 with 0 Axes>
```