Laboratorijska vježba 4

Konvolucija diskretnih vremenskih funkcija

Konvolucija dvije funkcije x[n] i h[n] analitički se označava kao y[n]=x[n]*h[n], gdje x[n] predstavlja pobudu, a y[n] odziv signala. Definicija konvolucije za diskretni signal, predstavljena je sljedećim izrazom:

$$y[n] = \sum_{k=-\infty}^{\infty} x[k]h[n-k]$$

Ukoliko se uvede smjena m=n-k, dobija se izraz:

$$y[n] = -\sum_{k=-\infty}^{\infty} x[m-n]h[m] = \sum_{k=-\infty}^{\infty} h[m]x[n-m] = h[n] * x[n]$$

Što znači da za konvoluciju važe slijedeće osobine:
Komutacija tj: x[n]*h[n]=h[n]*x[n]
Asocijacija tj: (x[n]*h[n])*g[n]=x[n]*(h[n]*g[n])

Distributivna je sa operacijom sabiranja signala:
$x[n] * (h_1[n] + h_2[n] = x[n] * h_1[n] + x[n] * h_2[n]$

Koraci u konvoluciji

1. Signal h[n] se prvo invertuje i pomjeri u vremenu da bi se dobio signal h[n-k] što postaje funkcija od k gdje je n konkretan parametar
2. Signali x[k] i h[n-k] se izmnože za sve moguće vrijednosti k za neko fiksno n
3. Proizvod $x[n] \times [n-k]$ se sumira za sve vrijednosti k i tako se dobija vrijednost y[n] za neko fiksno n
4. Ponove se koraci 1, 2, 3 za sve vrijednosti n iz skupa $(-\infty, \infty)$ kako bi se dobila kompletna funkcija y[n].

Zadatak 4.1

Koristeći programski jezik Python, izračunati konvoluciju dva pravougaona impulsa $x_1[n] = u[n-1] - u[n-3], ax_2[n] = u[n-1] - u[n-2]$

```
# -*- coding: utf-8 -*-
"""
Koriste i programski jezik Python, izra unati konvoluciju dva
pravougaona impulsa x1[n]=u[n-1]-u[n-3], a x2[n]=u[n-1]-u[n-2]
"""

import numpy as np
import matplotlib.pyplot as plt

x=np.array([0,1,1,1,0]) #jo  jedan na in predstavljanja niza

plt.stem(x)
plt.show()

h=np.array([0,1,1,0,0])
plt.stem(h)
plt.show()

lx=len(x)
lh=len(h)

zerosh=np.zeros(lh)
x=np.concatenate((x,zerosh),axis=0) #spajanje dva niza

zerosx=np.zeros(lx)
h=np.concatenate((h,zerosx),axis=0) #spajanje dva niza

y=np.zeros(lx+lh-1)

for n in range(1,lx+lh-1):
    y[n]=0
    for k in range(1,lx):
        if(n-k+1>0):
            y[n]=y[n]+x[k]*h[n-k+1]

plt.figure(3)
plt.stem(y)
```

Zadatak 4.2

Koristeći programski jezik Python i funkciju convolve za računanje konvolucije, nacrtati sljedeće signale dužine N=10:

1. $x_1[n]$->jedinični impuls
2. $x_2[n]$->Hevisajdova funckija
3. $h[n] = e^{0.5n}$

Zatim nacrtati signal y2=(x1*x2)*h gdje * predstavlja konvoluciju signala.

```
# -*- coding: utf-8 -*-
"""
Koriste i programski jezik Python i funkciju convolve za ra unanje konvolucije,
nacrtati sljede e signale du ine N=10:
    a) x1[n]->jedini ni impuls
    b) x2[n]->Hevisajdova funckija
    c) h[n]=e^0.5n

Zatim nacrtati signal y2=(x1*x2)*h gdje * predstavlja konvoluciju signala
"""
```

```
11
12 import matplotlib.pyplot as plt
13 import scipy as sp
14 from scipy import signal
15
16 n=sp.arange(-5,5,1)
17
18 x1=sp.zeros(10)
19 x1[4]=1 #Dirakov impuls
20
21 x2=1.*(n>=0) #Hevisajdova funkcija
22
23 h=sp.exp(0.5*n)
24
25 y1=signal.convolve(x1,x2)
26 plt.stem(y1)
27 plt.show()
28
29 y2=signal.convolve(y1,h)
30 plt.stem(y2)
31 plt.show()
```

Zadatak 4.3

Odrediti linearnu konvoluciju sekvenci:
$x(n) = 2\delta(n) + 2\delta(n-3) + 4\delta(n-6)$ $h(n) = 4\delta(n) - 3\delta(n-1) - 1\delta(n-2) + 4\delta(n-4)$

I način:

```
1  # -*- coding: utf-8 -*-
2  """
3  Odrediti linearnu konvoluciju sekvenci
4
5  x(n)=2delta(n) + 2delta(n-3) + 4delta(n-6)
6  h(n)=4delta(n)-3delta(n-1)-1delta(n-2)+4delta(n-4)
7  """
8  import numpy as np
9  import matplotlib.pyplot as plt
10 import scipy as sp
11 from scipy import signal
12
13 def stepfun(t,n):
14     x=np.zeros(len(t))
15     l=0;
16     while l<len(t):
17         if t[l]<n:
18             x[l]=0
19         else:
20             x[l]=1
21         l=l+1
22     return x
23
24 t=np.arange(-10,11,1)
25 # ============================================================================
26 #  x(n)=2delta(n)
27 # ============================================================================
28 x1=stepfun(t,0)
29 x2=stepfun(t,0.01)
30 x3=2*(x1-x2)
31
32 # ============================================================================
33 #  x(n)=2delta(n-3)
34 # ============================================================================
35
36 x4=stepfun(t,3)
37 x5=stepfun(t,3.01)
38 x6=2*(x4-x5)
```

```
39
40 # ================================================================================
41 #  x(n)=4delta(n-6)
42 # ================================================================================
43
44 x7=stepfun(t,6)
45 x8=stepfun(t,6.01)
46 x9=4*(x7-x8)
47
48 # ================================================================================
49 #  x(n)=2delta(n) + 2delta(n-3) + 4delta(n-6)
50 # ================================================================================
51
52 X=x3+x6+x9
53
54 plt.stem(X)
55 plt.show()
56
57 # ================================================================================
58 #  h(n)=4delta(n)
59 # ================================================================================
60 h1=stepfun(t,0)
61 h2=stepfun(t,0.01)
62 h3=4*(h1-h2)
63
64 # ================================================================================
65 #  h(n)=3delta(n-1)
66 # ================================================================================
67
68 h4=stepfun(t,1)
69 h5=stepfun(t,1.01)
70 h6=3*(h4-h5)
71
72 # ================================================================================
73 #  h(n)=1delta(n-2)
74 # ================================================================================
75
76 h7=stepfun(t,2)
77 h8=stepfun(t,2.01)
78 h9=(h7-h8)
79
80 # ================================================================================
81 #  h(n)=4delta(n-4)
82 # ================================================================================
83
84 h10=stepfun(t,4)
85 h11=stepfun(t,4.01)
86 h12=4*(h10-h11)
87 # ================================================================================
88 #  h(n)=4delta(n)-3delta(n-1)-1delta(n-2)+4delta(n-4)
89 # ================================================================================
90
91 H=h3-h6-h9+h12
92
93 plt.stem(H)
94 plt.show()
95
96 Y=signal.convolve(H,X)
97 plt.stem(Y)
98 plt.show()
```

II način:

```
1 # -*- coding: utf-8 -*-
2 """
3 Odrediti linearnu konvoluciju sekvenci
4
5 x(n)=2delta(n) + 2delta(n-3) + 4delta(n-6)
```

```
h(n)=4delta(n)-3delta(n-1)-1delta(n-2)+4delta(n-4)
"""

import matplotlib.pyplot as plt
import scipy as sp
from scipy import signal

n=sp.arange(-7,7,1)

x1=sp.zeros(10)
x1[0]=2 #Dirakov impuls
x1[3]=2
x1[6]=4

plt.stem(x1)
plt.show()
x2=sp.zeros(10)
x2[0]=4
x2[1]=-3
x2[2]=-1
x2[4]=4

plt.stem(x2)
plt.show()


y1=signal.convolve(x1,x2)
plt.stem(y1)
plt.show()
```