H.W Rom Hirsch -313288763

1.להן לפונקציה שיצרתי בpython לקונבולוציה :

"""

custom function Convoltion

"""

**def** custom\_cov**(**x**,**y**):**

m1**=len(**x**)**

m2**=len(**y**)**

#padded zeros to array

y **=** padded\_zeros**(**y**,**m2**+**m1**-**1**)**

x **=** padded\_zeros**(**x**,**m1**+**m2**-**1**)**

h **=** np**.**zeros**(**m1**+**m2**-**1**)** # check h[n] vector

**for** n **in** **range(**m2**+**m1**-**1**):**

**print(**n**)**

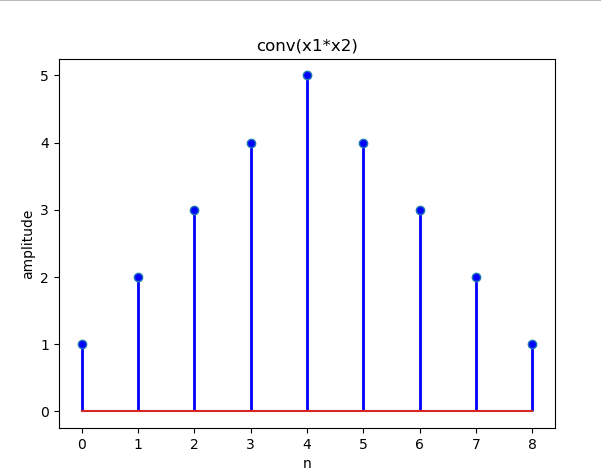
**for** k **in** **range(**m1**):**

h**[**n**]+=**x**[**k**]\***y**[**n**-**k**]**

**print(**h**[**n**])**

**return** h

2.plot x1\*x2 - קונבולוציה של x1 ו x2 בעזרת הפונקציה שבניתי



3. להן הפונקציה שיצרתי python לDFT :

"""

Function discrete Fourier Transform

"""

**def** custom\_DFT**(**f**):**

N **=** **len(**f**)**

F**=**np**.**zeros**(**N**,**dtype**=(complex))**

**for** r **in** **range(**N**):**

**for** n **in** **range(**N**):**

F**[**r**]** **+=** f**[**n**]** **\*** np**.**exp**(-**2j **\*** np**.**pi **\*** r **\*** n **/** N**)**

**return** F

4. להן לפונקציה שיצרתי python invert DFT :

"""

invert Function discrete Fourier Transform

"""

**def** custom\_IDFT**(**F**):**

N **=** **len(**F**)**

f**=**np**.**zeros**(**N**,**dtype**=(complex))**

**for** n **in** **range(**N**):**

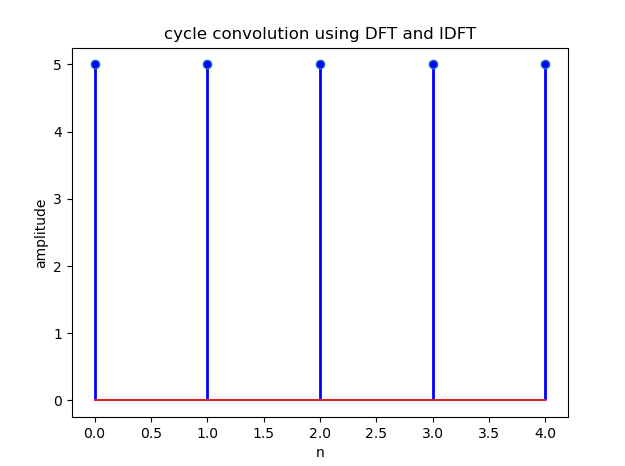
**for** r **in** **range(**N**):**

f**[**n**]** **+=(**1**/**N**)\*** F**[**r**]** **\*** np**.**exp**(**2j **\*** np**.**pi **\*** r **\*** n **/** N**)**

**return** f

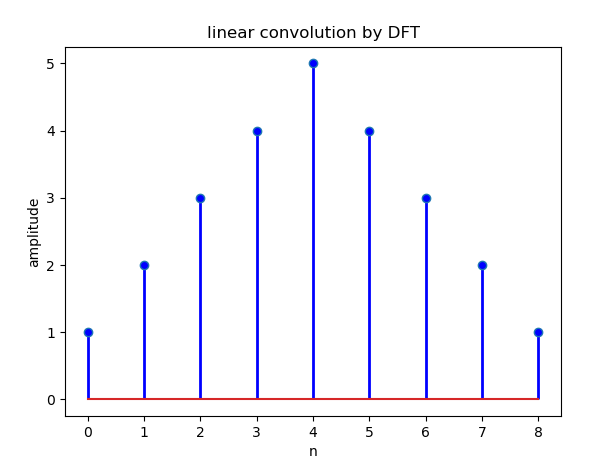
5. ביצעתי cyclic convolution אחרי בשימוש התוצאות של שאלה 3 – 4 :

Y=iDFT(DFT(x1)\*DFT(x2) (קונבולוציה במרחב היא הכפלה בתדר )

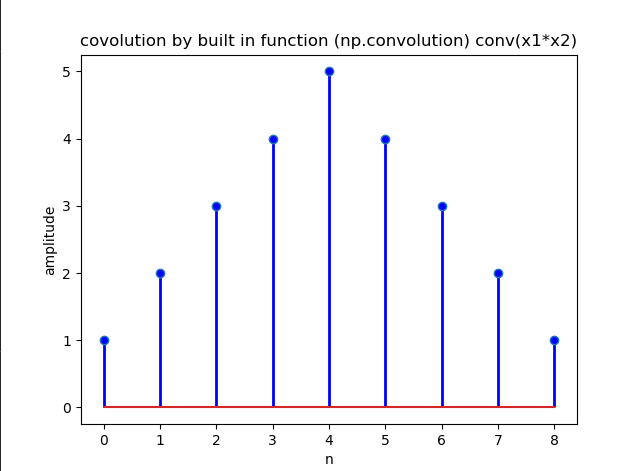


6.linear convolution by DFT

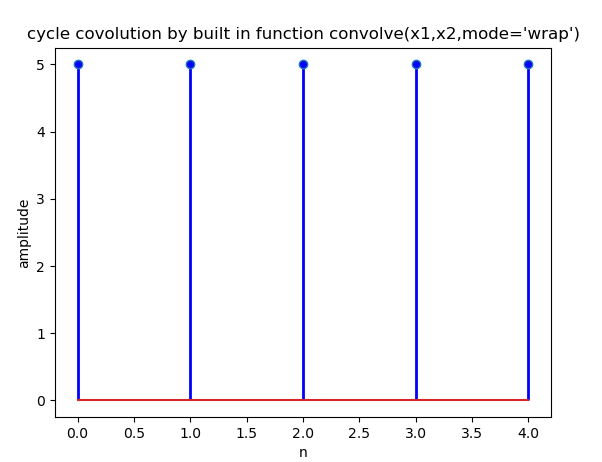
לביצוע ליניאר קונבולוציה נדרש להאריך את גודל הוקטורים לסכום אורי שתי הפולסים פחות אחד ומלא באפסים את החלק המורחב בווקטורים.



7. קונבולוציה ע"י שימוש בפונקציה מובנת np.convolution של python:



8. plot בשימוש בפונקציה מובנת בpython : scipy.ndimage.convolve(x1,x2,mode='wrap')



9. שימוש בפונקציה מובנית . "cconv" לקונבולוציה לינארית .

הקוד matlab :

%Q9

x1 = [1,1,1,1,1]

x2 = [1,1,1,1,1]

N = (0:1:8);

X3=cconv(x1,x2,9);%linear convolution by use build cycle convolution

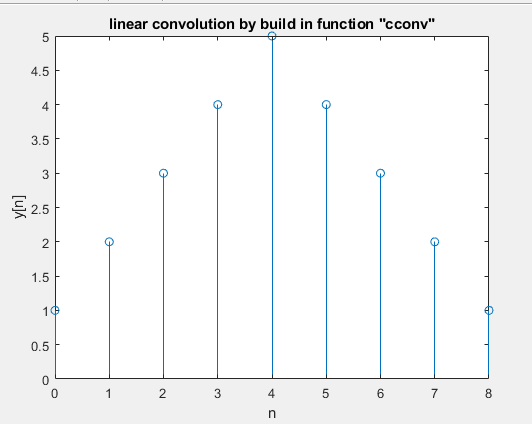
figure;

stem(N,X3);

xlabel('n');

ylabel('y[n]');

title('linear convolution by build in function "cconv"');



10 .ניתן לראות כי קונבולוציה מחזורית בעלת זמן מחזור גדול תניב לנו את אותה תוצאה שהקונבולוציה הלינארית מביאה לנו עד כי הוספה של אפסים לוקטור המורחב של הפונקציה,בנוסף ניתן לבצע קונבולוציה גם ע"י המשוואה עצמה וגם ע"י DFT וIDFT .

הקוד בpython :

# -\*- coding: utf-8 -\*-

"""

Created on Fri Aug 6 10:42:14 2021

@author: rom Hirsch - 313288763

"""

**import** scipy

**import** matplotlib**.**pyplot **as** plt

**from** scipy **import** signal

**import** numpy **as** np

#%%

#padded zeros to array , params (array ,size of the array after the padded)

**def** padded\_zeros**(**arr**,**N**):** # padded zeros

padded\_array **=** np**.**zeros**(**N**)**

padded\_array**[:len(**arr**)]+=**arr

**return** padded\_array

"""

custom function Convoltion

"""

**def** custom\_cov**(**x**,**y**):**

m1**=len(**x**)**

m2**=len(**y**)**

#padded zeros to array

y **=** padded\_zeros**(**y**,**m2**+**m1**-**1**)**

x **=** padded\_zeros**(**x**,**m1**+**m2**-**1**)**

h **=** np**.**zeros**(**m1**+**m2**-**1**)** # check h[n] vector

**for** n **in** **range(**m2**+**m1**-**1**):**

**print(**n**)**

**for** k **in** **range(**m1**):**

h**[**n**]+=**x**[**k**]\***y**[**n**-**k**]**

**print(**h**[**n**])**

**return** h

"""

Function discrete Fourier Transform

"""

**def** custom\_DFT**(**f**):**

N **=** **len(**f**)**

F**=**np**.**zeros**(**N**,**dtype**=(complex))**

**for** r **in** **range(**N**):**

**for** n **in** **range(**N**):**

F**[**r**]** **+=** f**[**n**]** **\*** np**.**exp**(-**2j **\*** np**.**pi **\*** r **\*** n **/** N**)**

**return** F

"""

invert Function discrete Fourier Transform

"""

**def** custom\_IDFT**(**F**):**

N **=** **len(**F**)**

f**=**np**.**zeros**(**N**,**dtype**=(complex))**

**for** n **in** **range(**N**):**

**for** r **in** **range(**N**):**

f**[**n**]** **+=(**1**/**N**)\*** F**[**r**]** **\*** np**.**exp**(**2j **\*** np**.**pi **\*** r **\*** n **/** N**)**

**return** f

#plot stem with color

**def** stem\_plot**(**n**,**val**,**color**):**

markerline1**,** stemlines1**,** baseline1 **=** plt**.**stem**(**n**,**val**)**

plt**.**setp**(**markerline1**,** 'markerfacecolor'**,** color**)**

plt**.**setp**(**stemlines1**,** linestyle**=**"-"**,** color**=**color**,** linewidth**=**2 **)**

#Create Stem plot with color

**def** plotStem**(**title**,**ylabel**,**xlabel**,**color**,**x**,**y**):**

plt**.**figure**()**

plt**.**title**(**title**)**

plt**.**ylabel**(**ylabel**)**

plt**.**xlabel**(**xlabel**)**

stem\_plot**(**x**,**y**,**color**)**

plt**.**show**()**

#%%%

"""

Q2 -

"""

x1 **=** **[**1**,**1**,**1**,**1**,**1**]**

x2 **=** **[**1**,**1**,**1**,**1**,**1**]**

h **=** custom\_cov**(**x1**,**x2**)**

n **=** np**.**arange**(len(**h**))**#get the axis x for plot

plotStem**(**"conv(x1\*x2)"**,**"amplitude"**,**"n"**,**'blue'**,**n**,**h**)**

#%%

"""

Q5 -

"""

x1 **=** **[**1**,**1**,**1**,**1**,**1**]**

x2 **=** **[**1**,**1**,**1**,**1**,**1**]**

x1 **=** custom\_DFT**(**x1**)**

x2 **=** custom\_DFT**(**x2**)**

h **=** x1**\***x2

h**=** custom\_IDFT**(**h**)**

n **=** np**.**arange**(len(**h**))**#get the axis x for plot

plotStem**(**"cycle convolution using DFT and IDFT"**,**"amplitude"**,**"n"**,**'blue'**,**n**,**h**)**

#%%

"""

Q6 -

"""

x1 **=** **[**1**,**1**,**1**,**1**,**1**]**

x2 **=** **[**1**,**1**,**1**,**1**,**1**]**

#padded with zero to length vector of (len(x1)+len(x2)-1)

m1 **=** **len(**x1**)**

m2 **=** **len(**x2**)**

x1 **=** padded\_zeros**(**x1**,**m2**+**m1**-**1**)**

x2 **=** padded\_zeros**(**x2**,**m2**+**m1**-**1**)**

x1 **=** custom\_DFT**(**x1**)**

x2 **=** custom\_DFT**(**x2**)**

h **=** x1**\***x2

h**=** custom\_IDFT**(**h**)**

n **=** np**.**arange**(len(**h**))**#get the axis x for plot

plotStem**(**"linear convolution by DFT"**,**"amplitude"**,**"n"**,**'blue'**,**n**,**h**)**

#%%

"""

Q7 -

"""

x1 **=** **[**1**,**1**,**1**,**1**,**1**]**

x2 **=** **[**1**,**1**,**1**,**1**,**1**]**

h **=** np**.**convolve**(**x1**,**x2**)**

n **=** np**.**arange**(len(**h**))**

plotStem**(**"covolution by built in function (np.convolution) conv(x1\*x2)"**,**"amplitude"**,**"n"**,**'blue'**,**n**,**h**)**

#%%

"""

Q8-

"""

x1 **=** **[**1**,**1**,**1**,**1**,**1**]**

x2 **=** **[**1**,**1**,**1**,**1**,**1**]**

h **=** scipy**.**ndimage**.**convolve**(**x1**,**x2**,**mode**=**'wrap'**)**#cycle conv

n **=** np**.**arange**(len(**h**))**#get the axis x for plot

plotStem**(**"cycle covolution by built in function convolve(x1,x2,mode='wrap')"**,**"amplitude"**,**"n"**,**'blue'**,**n**,**h**)**