פונקציות שבנית והשתמשת בהם במהלך המטלה

```python
# Lior Cohen 314818345

# importing required libraries
import math
import cmath
import matplotlib.pyplot as plt
import numpy as np
```

הספריות שהשתמשת בהן ←

```python
# function that calculates the Fourier series and creating its array of coefficients, time domain -> frequency domain
3 usages  Lior Cohen *
def fourier_series(signal):
    num = len(signal)                                   # calculating the signals length
    coef = [0] * num                                    # creating the array of coefficients
    for ik in range(- num // 2, num//2 + 1):        # calculating the coefficients with the formula give on the task
        for elem in range(- num // 2, num//2 + 1):   # technically it also does the Fourier transform to the signal
            coef[ik + num // 2] += (1 / num) * signal[elem + num // 2] * np.exp(-1j * ik * 2 * math.pi *
                                                                                elem / num)
    return coef                                         # returning the coefficients array
```

פונקציה של עצמות בעקבי פוריה (שם של התמרת פוריה) ←

```python
# function the finds specific coefficient and plots its signal in the time domain
1 usage  Lior Cohen *
def find_coef_and_plot(coef, signal, samples, sig_length, title='Coefficient', xlabel='n', ylabel='Amplitude'):
    idx = coef + sig_length // 2                    # finding the actual given index in the array
    sig_sp_coef = signal[idx]                       # saving the coefficient to a variable
    x_coef = sig_sp_coef * np.exp(1j * 2 * coef * samples * math.pi / sig_length)   # calculating it in the time domain
    plot_signal_smth(np.real(x_coef), samples, title, xlabel, ylabel)       # plotting the specific coefficient signal
```

פונקציה עם עצמות איבר והוצאת גרף שלו ←

```python
# function that does the inverse Fourier transform, creates its array of coefficients, frequency domain -> time domain
4 usages  Lior Cohen *
def re_transform(signal):
    num = len(signal)                           # calculating the length of the given signal
    signal_t = np.zeros(num, dtype=complex)     # creating the array of zeroes for the inversed coefficients
    for rk in range(num):                       # rk indexes on the array
        rk_val = rk - num // 2                  # rk_val is the actual k index in Fourier math
        for mi in range(num):                   # mi indexes on the array
            m_val = mi - num // 2               # m_val is the actual m index in Fourier math
            signal_t[rk] += signal[mi] * cmath.exp(1j * rk_val * 2 * math.pi * m_val / num)  # inverse Fourier transform
    return np.real(signal_t)                    # returning the time domain signal
```

פונקציה של התמרת פוריה הפוכה ←

```python
# function that plots the discrete signal, here it is being used mainly for fourier series plots (ak, bk, ck, etc...)
5 usages  Lior Cohen *
def plot_disct(signal, samples, title='Discrete signal', xlabel='n', ylabel='Amplitude'):
    plt.figure(figsize=(8, 4))                                  # figure window & figure size formating
    plt.stem( *args: samples, np.real(signal), linefmt='blue', markerfmt='bo', basefmt=" ")     # plot discrete signal
    plt.axhline(y=0, color='black', linewidth=1.15)             # plotting the axis lines
    plt.axvline(x=0, color='black', linewidth=1.15)             # x-axis first, then y-axis
    plt.title(title)                                            # plot title
    plt.xlabel(xlabel)                                          # x-axis title
    plt.ylabel(ylabel)                                          # y-axis title
    plt.grid(True)                                              # adding grid
    plt.show()                                                  # showing plot
```

הוצאת גרף דיסקרטי של כל בעקבים ←

```python
# function to plot discrete plots of the magnitude and the phase of a signal in the same figure window
2 usages  Lior Cohen *
def plot_mag_and_phase(magnitude, signal_phase, samples, title='Mag & Phase of Signal', mag_xlabel='n',
                       mag_ylabel='Amplitude', p_xlabel='n', p_ylabel='Amplitude'):
    plt.figure(figsize=(10, 8))                                 # figure window & figure size formating
    plt.subplot( *args: 2, 1, 1)                                # creating subplot 1 (top) for the magnitude
    plt.stem( *args: samples, magnitude, linefmt='blue', markerfmt='bo', basefmt=" ")   # plotting the signal magnitude
    plt.axhline(y=0, color='black', linewidth=1.15)             # axis lines formeting
    plt.axvline(x=0, color='black', linewidth=1.15)
    plt.xlim( *args: -100, 100)                                 # zooming in on the area between x = -100 to x =100
    plt.title(title)                                            # plot title
    plt.xlabel(mag_xlabel)                                      # magnitude subplot x-axis title
    plt.ylabel(mag_ylabel)                                      # magnitude subplot y-axis title
    plt.grid(True)                                              # adding grid

    plt.subplot( *args: 2, 1, 2)                                # creating subplot 2 (bottom) for the phase
    plt.stem( *args: samples, signal_phase, linefmt='blue', markerfmt='bo', basefmt=" ")    # plotting the signal phase
    plt.axhline(y=0, color='black', linewidth=1.15)             # axis lines formeting
    plt.axvline(x=0, color='black', linewidth=1.15)
    plt.xlabel(p_xlabel)                                        # phase subplot x-axis title
    plt.ylabel(p_ylabel)                                        # phase subplot y-axis title
    plt.grid(True)                                              # adding grid

    plt.tight_layout()                              # making sure the plots spaced right and not overlapping each over
    plt.show()                                      # showing plot
```

הוצאת גרפים לגשנ'ודה והפאזה ←

```python
# creating functions to plot the graphs throughout the assignment

# function that plots smooth continuous graph of given signal
8 usages  Lior Cohen *
def plot_signal_smth(signal, time_samples, title='signal', xlabel='n', ylabel='Amplitude'):
    plt.figure(figsize=(10, 4))                                 # figure window & figure size formating
    plt.plot( *args: time_samples, signal, color='blue', zorder=2)    # plotting the signal
    plt.axhline(y=0, color='black', linewidth=1.15, zorder=0)   # plotting the axis lines, x-axis first then y-axis
    plt.axvline(x=0, color='black', linewidth=1.15, zorder=0)   # zorder makes sure they are under the signal
    plt.title(title)                                            # plot title
    plt.xlabel(xlabel)                                          # x-axis title
    plt.ylabel(ylabel)                                          # y-axis title
    plt.grid(True)                                              # adding grid
    plt.show()                                                  # showing the plot
```

הוצאת גרף רציף ←

```python
# same as the discrete plot function but with one added line to zoom on the signal for better sight of its behavior
3 usages  Lior Cohen *
def plot_disct_zoom(signal, samples, zoom, title='Discrete signal (Zoom)', xlabel='n', ylabel='Amplitude'):
    plt.figure()
    plt.stem( *args: samples, np.real(signal), linefmt='blue', markerfmt='bo', basefmt=" ")
    plt.axhline(y=0, color='black', linewidth=1.15)
    plt.axvline(x=0, color='black', linewidth=1.15)
    plt.xlim( *args: -zoom, zoom)                   # zooms in on symmetric area on the x-axis
    plt.title(title)
    plt.xlabel(xlabel)
    plt.ylabel(ylabel)
    plt.grid(True)
    plt.show()
```

הוצאת גרף דיסקרטי עם זום על איזור שנים ←

נתון אות חלון הבא :

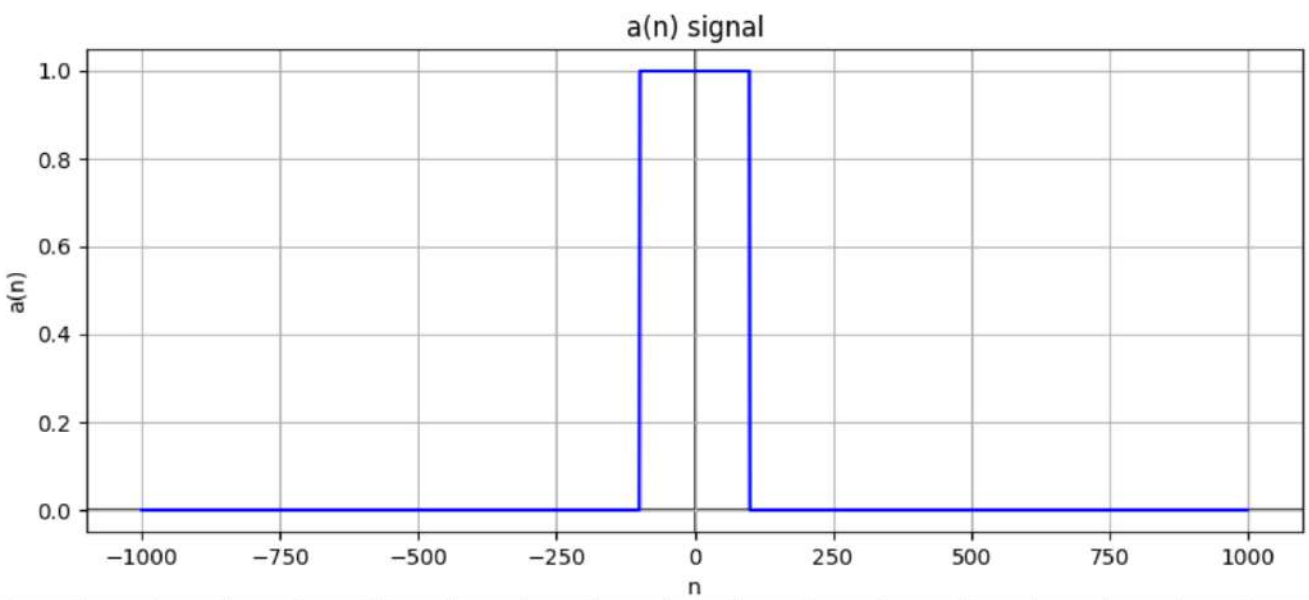
$$a(n) = \begin{cases} 1 & |n| < 100 \\ 0 & \text{אחרת} \end{cases}$$

א. קבעו וקטור זמנים, $n = -1000:1:1000$, וציירו את האות.

הקוד לסעיף א'

```python
# ====Q1=====
# creating the time vector n= -1000:1:1000.

# start value is -1000 and the stop value is 1001 (1001 not included).
n = np.arange(-1000, 1001)

# creating the Window signal as required.
# np.abs(n) < 100 make sure that the condition is only for what between -99 and 99 (99 and -99 included)
a = np.where(np.abs(n) < 100, 1, 0)
N = len(a)          # calculating the signals length for future usage

plot_signal_smth(a, n, title='a(n) signal', xlabel='n', ylabel='a(n)')    # plotting the a(n) signal
```

האות הרצוי אות חלון $x \in (-100, 100)$



תשובות שהדפסתי בהתאם לדרישה

```
C:\Users\USER\AppData\Local\Programs\Python\Python310\python.exe "C:\Users\USER\Documents\University\EE\2nd Year\Signals and Systems\code projects\part
 1\SignalsAndSystemsP1\Assignment.py"
Max real difference: 0.0084936648469998034
a_k is Symmetric
Max imaginary value of ak: 5.4752209710517974e-18
1.000493664238575
Max value of d(n): 199.001462961002
Parseval in the frequency domain over d(k): 2625.586710213759
Parseval in the time domain over d(n): 2625.586709143633
max error: 0.0004963204926038184

Process finished with exit code 0
```

ב. התייחסו אל האות בסעיף א' כאל מחזור יחיד של אות מחזורי במחזור: $N = 2001$.

עתה נרצה להציג את מקדמי טור פורייה של האות

$$a_k = \frac{1}{N} \sum_{n=<N>} x(n) e^{-jk2\pi n/N}$$

הציגו את מקדמי פורייה של האות . כיון ש $a(n)$ הוא ממשי וסימטרי בזמן הראו כי a_k הוא ממשי וסימטרי בתדר.

```python
# ====Q2====
# here we want to find out the Fourier series coefficients of a(n) and plotting it
# and also checking if the signal in the frequency domain is real and symmetric

# function that calculates the Fourier series and creating its array of coefficients, time domain -> frequency domain
3 usages  Lior Cohen *
def fourier_series(signal):...

# function that does the inverse Fourier transform, creates its array of coefficients, frequency domain -> time domain
4 usages  Lior Cohen *
def re_transform(signal):...


# function the finds specific coefficient and plots its signal in the time domain
1 usage  Lior Cohen *
def find_coef_and_plot(coef, signal, samples, sig_length, title='Coefficient', xlabel='n', ylabel='Amplitude'):...


ak = fourier_series(a)            # calling for the function to calculate and create the Fourier series of given signal
phase = np.angle(ak)              # calculating the phase of the Fourier series coefficients phase of ak
mag = np.abs(ak)                  # calculating the ak coefficients magnitude

plot_disct(ak, n, title='a(k), Fourier Series Coefficients', xlabel='k', ylabel='ak')     # plot discrete plot of ak
plot_disct_zoom(ak, n, zoom=75, title='a(k), Fourier Series (Zoom)', xlabel='k', ylabel='ak')  # zooms on the discrete
                                                                                              # ak plot
plot_mag_and_phase(mag, phase, n, title='Magnitude and Phase of a(k)', mag_xlabel='k',
                   mag_ylabel='|ak|', p_xlabel='k', p_ylabel='Phase')              # plot magnitude and phase plot

# symmetry check

diffs = []                                # creating array for the max difference between the sides if the signal
for i in range(1, N // 2 + 1):            # running on the positive indexes of the signal
    diff = abs(np.real(ak[i + N//2]) - np.real(ak[-i + N//2]))      # calculating the difference between the value
                                                                    # of the positive and negative indexes
    diffs.append(diff)                    # adding the value of the difference to the array

print("Max real difference:", max(diffs))       # finding the max difference value in the array and printing it

flag = True                           # declaring the flag before the loop
for i in range(1, N//2 + 1):          # running over the positive side of the array
    right = np.real(ak[i + N//2])     # index on the positive side
    left = np.real(ak[-i + N//2])     # index on the negative side
    if not np.isclose(left, right, atol=5e-4):  # checking if the difference is not greater than 5e-4
        flag = False                  # if the difference exceeds the tolerance value the flag will be updated

if flag:                              # checking if the flag is on True
    print('a_k is Symmetric')         # printing that the signal is symmetric
if not flag:                          # checking if the flag is on False
    print('a_k is Asymmetric')        # printing that the signal is asymmetric

# real check

max_imag = np.max(np.abs(np.imag(ak)))      # finding the max value of the imaginary part of the signal
print('Max imaginary value of ak:', max_imag)                          # printing the max value
```

קיבלנו כי האות סימטרי כמצר בהנחה שקיבלנו בין שתי הגבולות על כאשר נוכל מהחיסור הנומרי והאיבוד של חלק מהתשובים בגבלך ההרבה

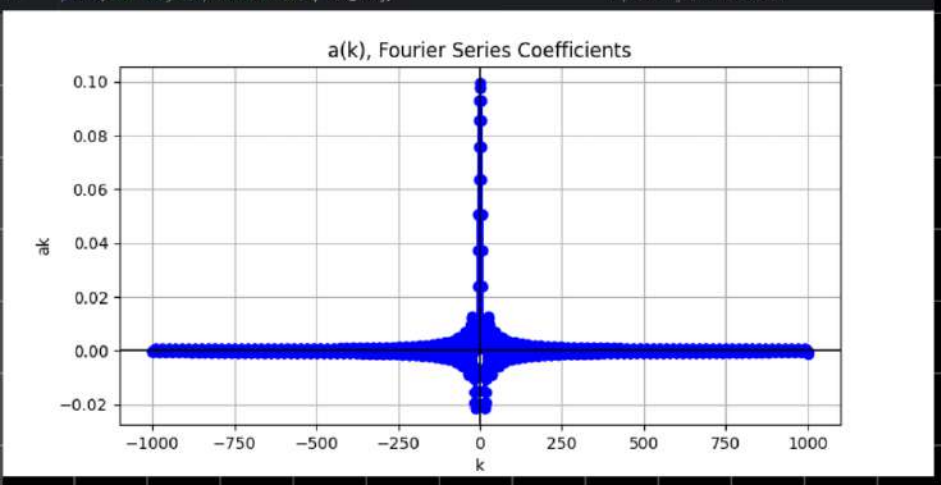
```
Max real difference: 0.0004936648469998034
a_k is Symmetric
Max imaginary value of ak: 5.4752209710517974e-18
```

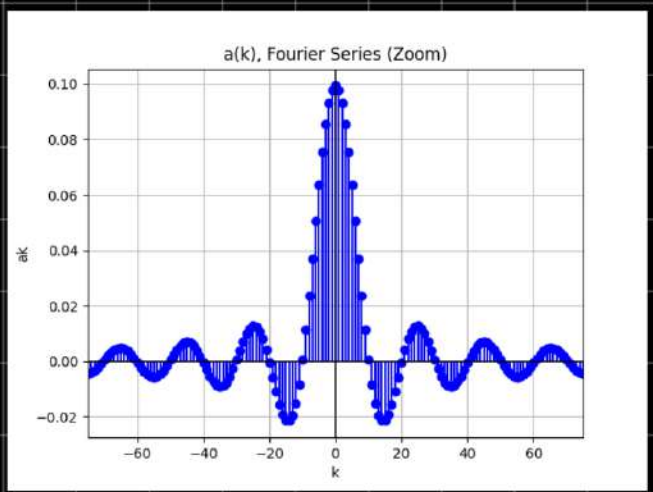
האות ממש, קיבלנו חלק מדומה מאוד קטן בגלל התשובים הנומרים

וגם כלום 5.5×10^{-18} זה מספר קטן מטמטמות וקרוב מאוד לאפס

הקוד הבא לסעיף ב' ←

גרף של מקדמי טור פורייה a_k ←





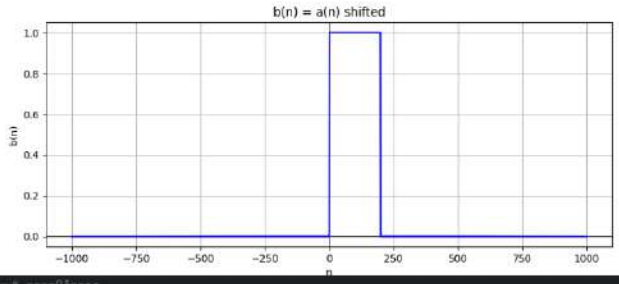
צוף על הגרף לגלה בתחום $x\in[-75,75]$

ג. נרצה עתה לבחון את הזהות של הזזה בזמן ⇔ הכפלה באקפוננט בתדר

חשב את : $b_k = a_k e^{-jk2\pi 100/N}$

חשבו והציגו את האות בזמן $b(n)$ ע"י ההתמרה ההופכית של b_k ואשרו את הזהות הנ"ל.

- **הסבר את ההבדל בין הגרפים a_k, b_k.**
- **שרטט את גרף הפאזה של a_k, b_k.**
- **צייר את המקדם b_{10} במרחב הזמן. מהו הביטוי המתמטי?**



קיבלנו את b(n) שהוא בעצם a(n) שהוזז ב- 100 יחידות ימינה,

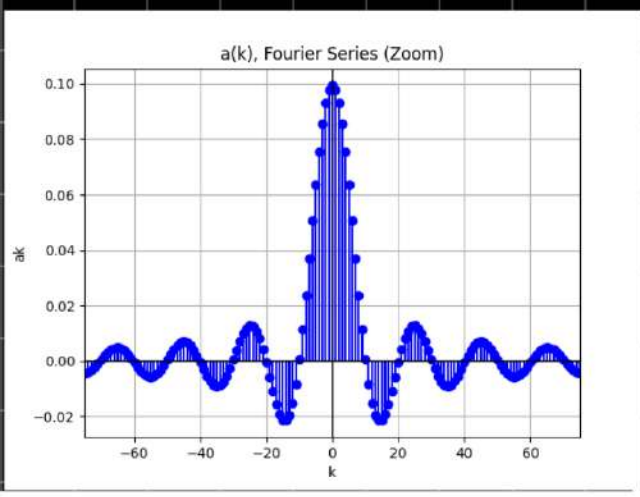
$$b(n) = \begin{cases} 1 & 0<n<200 \\ 0 & else \end{cases}$$

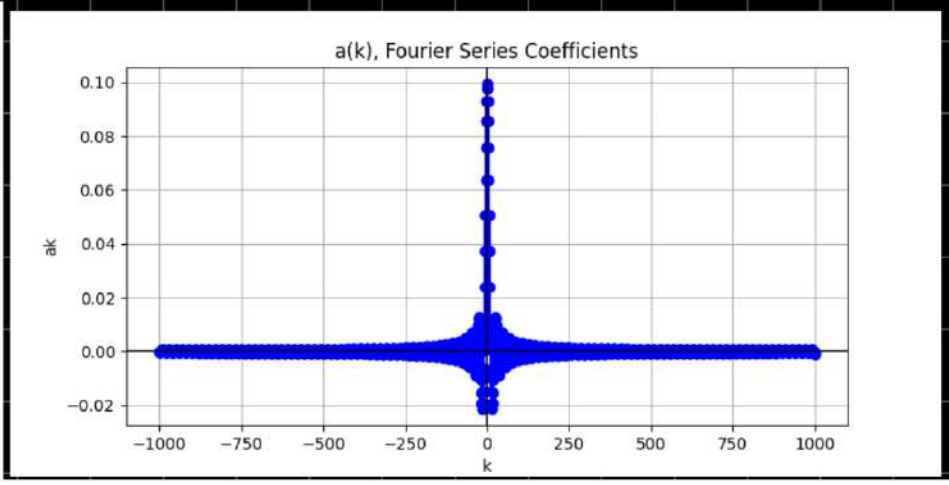
הקוד לסעיף ג' ←

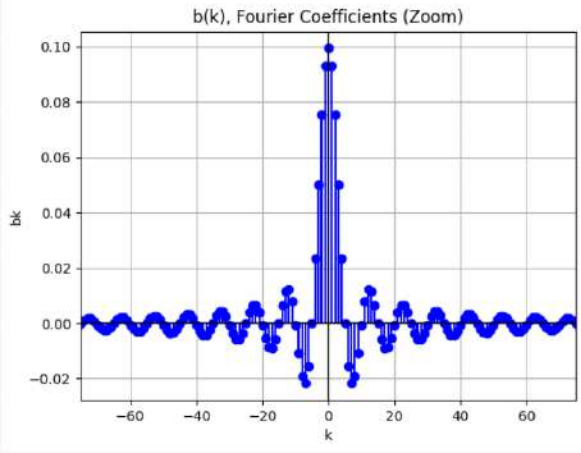
```python
# ====Q3====
# here we want to see the identity that says:
# multiplication by exponent in the frequency domain <==> shifting in the time domain

bk = np.zeros(N, dtype=complex)          # creating the array for the bk signal
for r in range(N):                       # running on all the 2001 indexes
    k_val = r - N//2                     # the actual index on the centered signal
    bk[r] = ak[r] * cmath.exp(-1j * 2 * np.pi * k_val * 100 / N)    # multiplying the index value with exponent

plot_disct(bk, n, title='b(k)', xlabel='k', ylabel='bk')    # plotting bk discrete signal
plot_disct_zoom(bk, n, zoom:75, title='b(k), Fourier Coefficients (Zoom)', xlabel='k', ylabel='bk')    # zoom on the plot

b = re_transform(bk)                     # creating b(n) with the inverse Fourier transform function
b_real = np.real(b)                      # taking the real part of b(n)
plot_signal_smth(b_real, n, title='b(n) = a(n) shifted', xlabel='n', ylabel='b(n)')      # plotting b(n) (shifted a(n))
print(np.max(b_real))                    # printing the max value of b(n) to see if the signal transformed correctly

bk_mag = np.abs(bk)                      # creating the signal for the magnitude of b(k)
bk_phase = np.angle(bk)                  # creating the signal for the phase of b(k)
plot_mag_and_phase(bk_mag, bk_phase, n, title='Magnitude and Phase of b(k)', mag_xlabel='k',
                   mag_ylabel='|bk|', p_xlabel='k', p_ylabel='Phase')       # plotting the magnitude & phase signals

# finding a specific coefficient contribution and plotting it
find_coef_and_plot( coef: 10, bk, n, N, title='b_10 coefficient in time', xlabel='n', ylabel='b(10)')
```

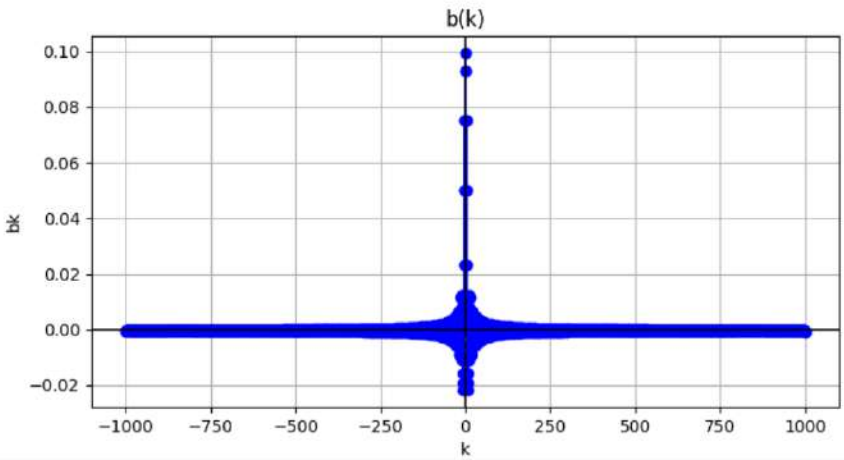
ניתן לראות שלכי בגרפים בזוגלים של ak ו-bk בנקודים הרבה יותר גבוהים כתוצאה מההכפלה באקספוננט, אבל את הבדל באמת ניתן לראות בגרפים של הפאזות של ak ו-bk

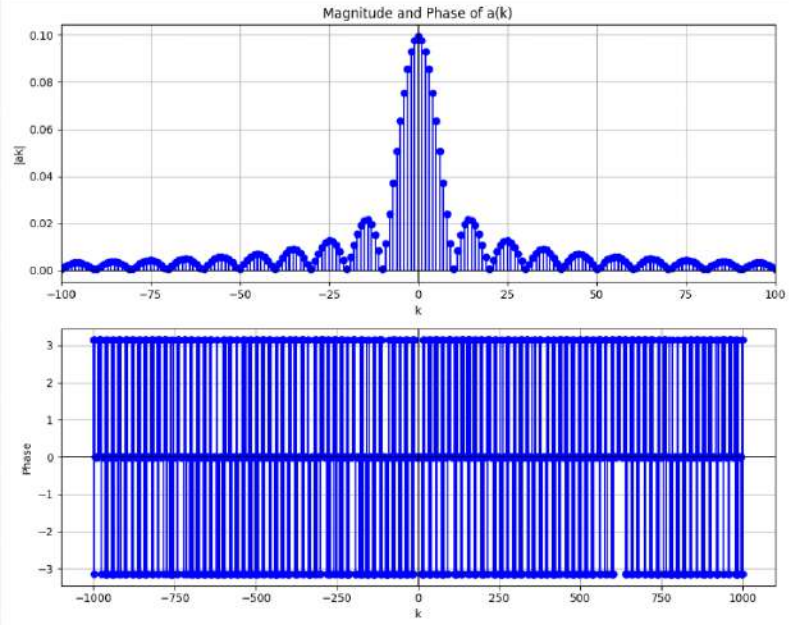
zoom

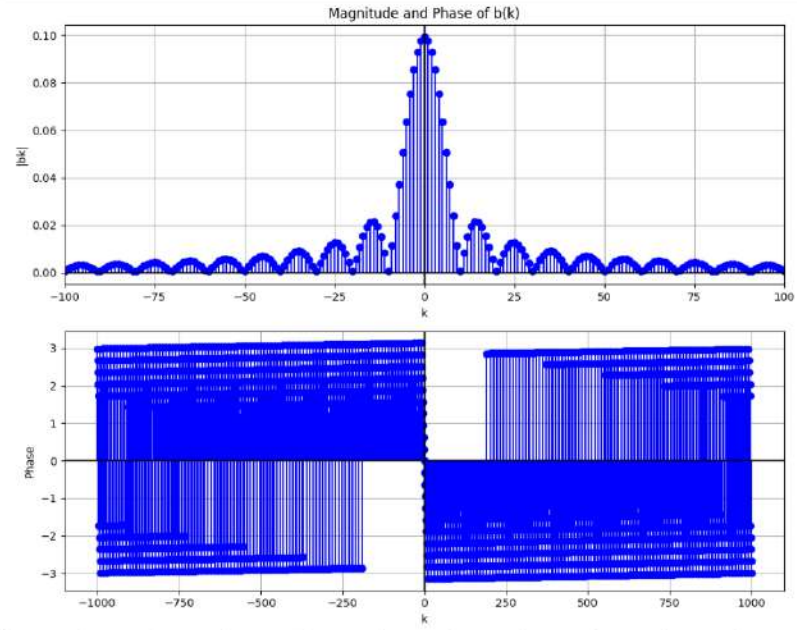












$a(k)$ $b(k)$

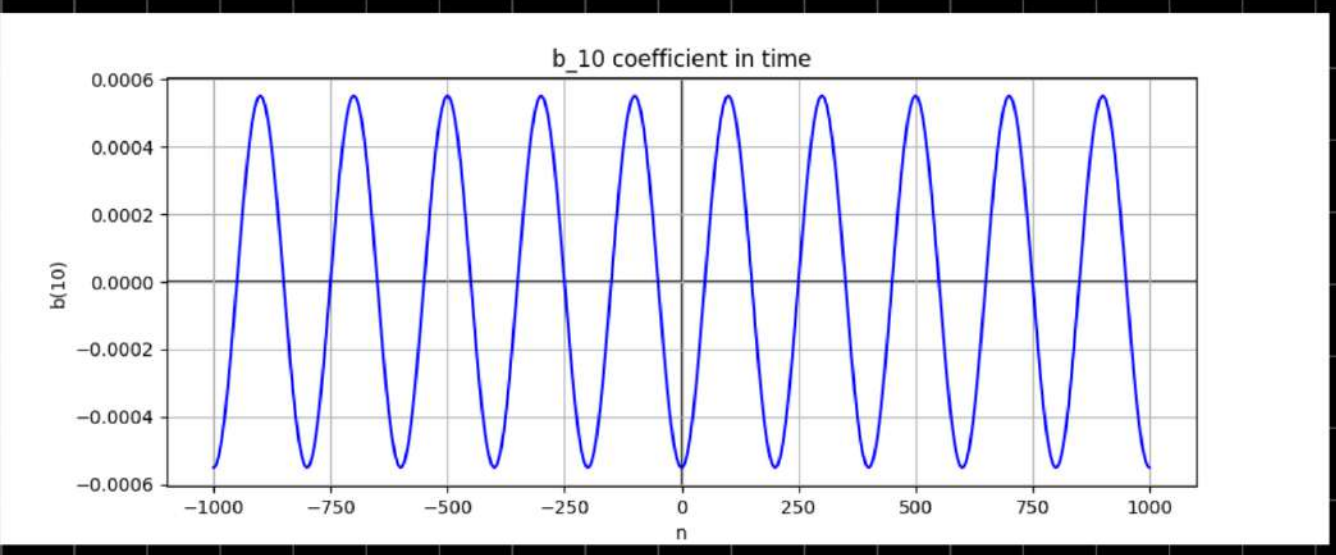
גרפי המגניטודה בין האליונים
גרפי הפאזה הן התחתונים

רואים שבהבלה בתדר פונט לא נשתנה את המגניטודה אלא רק את הפאזה

$$b_k = a_k \cdot e^{-j\frac{2\pi k 100}{N}}$$

$$b_{10} = a_{10} \cdot e^{-j\frac{2\pi 10000}{N}}$$

$$\boxed{b_{10} = \frac{1}{N}\sum_{n=-100}^{100} a(n)\, e^{-j\frac{2\pi}{N}10\cdot n} \cdot e^{-j\frac{2\pi 10000}{N}}}$$



ד. נרצה עתה לבחון את הזהות של גזירה בזמן ⇔ "הכפלה ב k " בתדר.

חשבו את : $c_k = a_k\left(1 - e^{-jk2\pi/N}\right)$

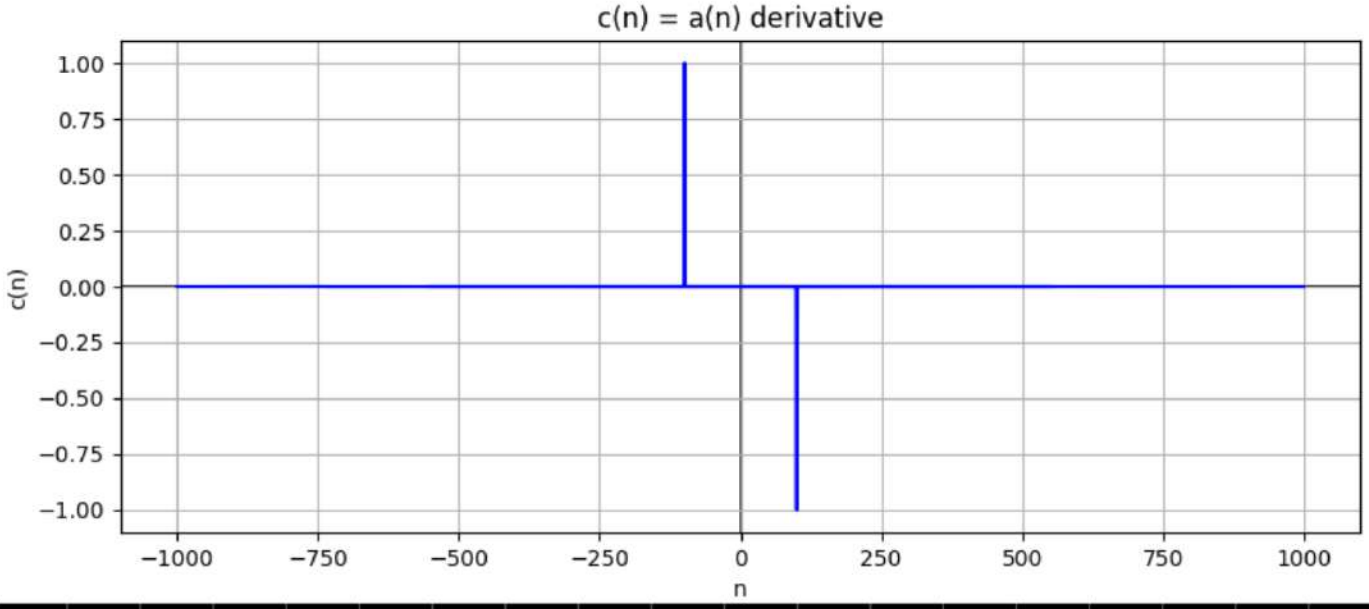
חשבו והציגו את האות בזמן $c(n)$ ע"י ההתמרה ההופכית של c_k . האם קבלתם גזירה בזמן?

אשרו זאת אנליטית.

```python
# ===Q4===
# here we want to see the identity that says:
# "multiplicetion by k" in the frequency domain <==> derivative in the time domain

ck = np.zeros(N, dtype=complex)            # creating the array for c(k) signal
for i in range(N):                         # running on all the indexes of the signal
    k_val = i - N//2                       # the actual index of the centered signal
    ck[i] = ak[i] * (1 - cmath.exp(-1j * 2 * np.pi * k_val / N))   # implementing the formula given in the questions
                                                                   # sheet for multiplying by k in the frequency domain
                                                                   # to find the derivative in the time domain

c = re_transform(ck)                       # inverse transform on c(k) to find the c(n) signal
c_real = np.real(c)                        # taking the real part of the signal c(n)
plot_signal_smth(c_real, n, title='c(n) = a(n) derivative', xlabel='n', ylabel='c(n)')   # plotting c(n)
```

הקוד לסעיף ד' ←

הגרף של $c(n)$ ניתן לראות שלמעשה הנגזרת של $a(n)$



בזמן:

$$a(n) = u[n+100] - u(n-100] \xrightarrow{\frac{d}{dn}}$$

$$\boxed{a'(n) = c(n) = \delta[n+100] - \delta[n-100]}$$

בתדר:

$$c_k = \sum a_n e^{-j\frac{2\pi kn}{N}} - \sum a_n e^{-j\frac{2\pi k}{N}(n-1)} =$$

$$\sum a_n e^{-j\frac{2\pi kn}{N}}\left(1 - e^{-j\frac{2\pi k}{N}}\right) =$$

$$= \boxed{a_k\left(1 - e^{-j\frac{2\pi k}{N}}\right) = c_k}$$

ה. נרצה עתה לבחון את הזהות של קונבלוציה בזמן ⇔ הכפלה בתדר.

חשבו את : $d_k = Na_k^2$

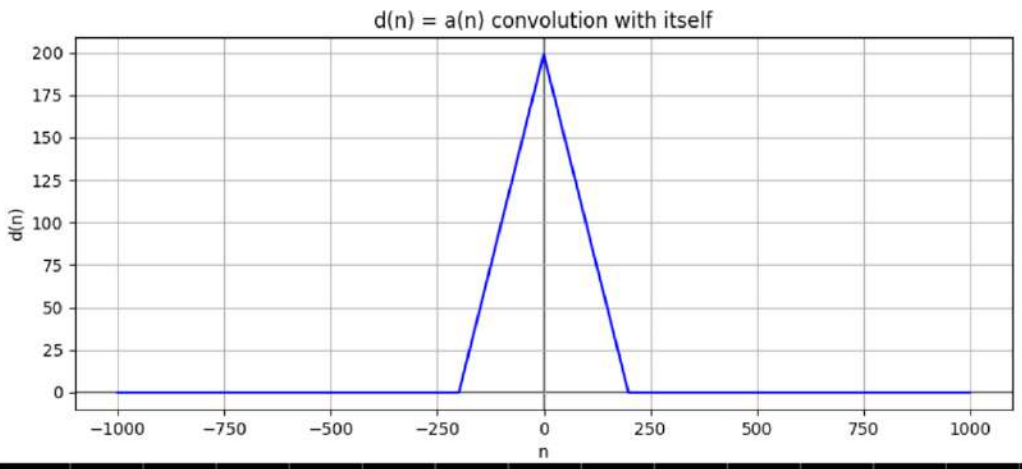
חשבו והציגו את האות בזמן $d(n)$ שהינו ההתמרה ההופכית של d_k . האם קבלתם קונבלוציה בזמן? אשרו זאת אנליטית.

חשב את $\max\{d_n\}$ והסבר את התוצאה.

הקוד של ה' ←

```python
# ====Q5====
# here we want to see the identity that says:
# multiplication in the frequency domain <==> convolution in the time domain

dk = np.zeros(N, dtype=complex)     # creating the array for d(k) signal
for i in range(N):                  # running on all the indexes of the signal
    k_val = i - N//2                # the actual index on the centered signal
    dk[i] = ak[i] * ak[i] * N       # multiplying the signal by its length and itself for convolution in the time domain

d = re_transform(dk)                # inverse transform on d(k) to find the d(n) signal
d_real = np.real(d)                 # taking the real part of d(n)
plot_signal_smth(d_real, n, title='d(n) = a(n) convolution with itself', xlabel='n', ylabel='d(n)')      # plotting d(n)
d_max = np.max(d_real)              # finding the max value of d(n)
print('Max value of d(n):', d_max)                          # printing the max value
```

הגרף ←




```
Max value of d(n): 199.001462961002
```

הערך המקס' של $d(n)$ ←

לדעתנו בכיתה שקונבולוציה בזמן זה הכפלה בתדר.
אם כאשר הכפלנו את הגל הריבועי מצאנו קיבלנו קונבולוציה של שתי גלים ריבועיים בגבים בזמן
שיודע לנו שמתרחשת קונבולוציה של שתי אותות ריבועיים בגבים נתון
גל נשולם שהגבולות שלו בציר ה-x הוא פי שתיים מהגבולות של כל
גל ריבועי בלבד $a(n) : n \in [-99, 99]$

$$\downarrow$$

$$c[n] : n \in [-198, 198]$$

והערך המקס' לי יתור להיות בשונה של פעמיים המקסימום של האות הריבועי
וקיבלנו $199 \approx$ וגלל שגיאות נומריות

ו. נרצה עתה לבחון את שוויון פרסיבל.

חשבו בנפרד את : $\frac{1}{N}\sum_{n=<N>}^{\square}|d(n)|^2$

ואת : $\sum_{k=<N>}^{\square}|d_k|^2$

והראו כי מתקיים שוויון.

הקוד לסעיף ו'

```python
# ====Q6====
# here we want to see the parseval identity in the time and frequency domains ,and see if it gives the same value

par_dn = 0                          # initializing the variable for the Parseval identity in the time domain
par_dk = 0                          # initializing the variable for the Parseval identity in the frequency domain
for i in range(N):
    par_dk += np.abs(dk[i] * dk[i])    # calculating Parsevals sum for the frequency domain signal

for i in range(N):
    par_dn += (1 / N) * np.abs(d[i] * d[i])     # calculating Parsevals sum for the time domain signal

# printing the values
print('Parseval in the frequency domain over d(k):', par_dk)
print('Parseval in the time domain over d(n):', par_dn)
```

ניתן לראות ששתי התוצאות נהיות עד כדי שגיאה

```
Parseval in the frequency domain over d(k): 2625.586710213759
Parseval in the time domain over d(n): 2625.586709143633
```

שגיאה בגודל של $\pm 10^{-5}$

ז. נרצה עתה לבחון את הזהות הכפלה בזמן ⇔ קונבלוציה בחדר.

חשבו את : $e(n) = a(n)b(n)$

חשבו והציגו את האות בתדר e_k ע"י ההתמרה של $e(n)$. חשבו והציגו גם את e_k המתקבל ע"י

קונבלוציה ציקלית בתדר דהיינו $\hat{e}_k = \sum_{l=<N>} a_l b_{k-l}$. אשרו כי האות e_k זהה ל $\hat{e}_k$.

```python
# ===Q7===
# here we want to see the identity that says:
# multiplication in the time domain <==> circular convolution in the frequency domain
# and checking how close the signals values to each other

e = np.zeros(N, dtype=complex)      # creating the array for e(n) signal
for i in range(N):
    e[i] = a[i] * b[i]              # calculating the e(n) signal by multiplying a(n) with b(n)

ek = fourier_series(e)              # finding the e(k) signal using the Fourier series function
e_real = np.real(e)                 # taking the real part of e(n)
plot_signal_smth(e_real, n, title='e(n) Signal', xlabel='n', ylabel='e(n)')     # plotting e(n)
plot_disct(ek, n, title='e(k), Regular Fourier Transform Samples', xlabel='k', ylabel='ek')   # plotting e(k)
plot_disct_zoom(ek, n, zoom=75, title='e(k), Regular Fourier Transform Samples (Zoom)', xlabel='k', ylabel='ek')
# zoom on e(k)

ek_hat = np.zeros(N, dtype=complex)     # creating the array for e_hat(k) signal
for i in range(N):          # running on all the indexes of the signal
    k = i - N // 2          # the actual k index in the centered signal
    for j in range(N):      # running on all the indexes of the signal
        m = j - N // 2      # the actual m index in the centered signal
        km = (k - m) % N    # gives index between 0 to N-1
        idx_km = (km + N // 2) % N  # adjusting the index to centered with the signal
        ek_hat[i] += ak[j] * bk[idx_km]  # preforming the circular convolution

e_hat = re_transform(ek_hat)        # inverse transform for e_hat(k) -> e_hat(n)
e_hat_real = np.real(e_hat)         # taking the real part of e_hat(n)
plot_signal_smth(e_hat_real, n, title='e(n) Hat', xlabel='n', ylabel='e(n)')    # plotting e_hat(n)
plot_disct(ek_hat, n, title='ek hat coefficients', xlabel='k', ylabel='ek_hat')  # plotting e_hat(k)
plot_disct_zoom(ek_hat, n, zoom=75, title='ek hat (Zoom)', xlabel='k', ylabel='ek_hat')  # zoom on e_hat(k)
max_error = np.max(np.abs(ek - ek_hat))     # finding the max difference between e(k) and e_hat(k)
print("max error:", max_error)              # printing the max difference
```

הקוד לסעיף ז' ←

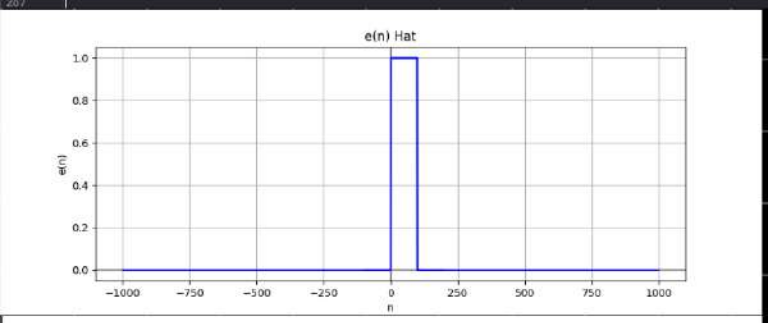
ההפרש המקסימלי בין שתי האותות

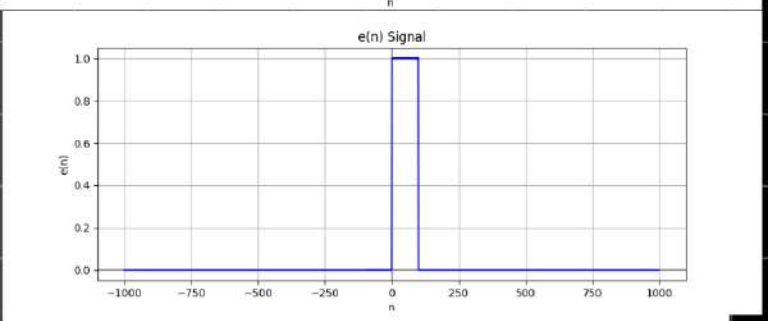
```
max error: 0.0004963204926038184
```

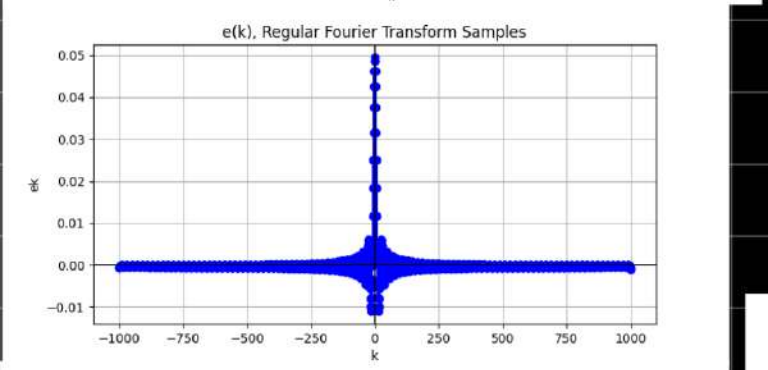
תואם כאן

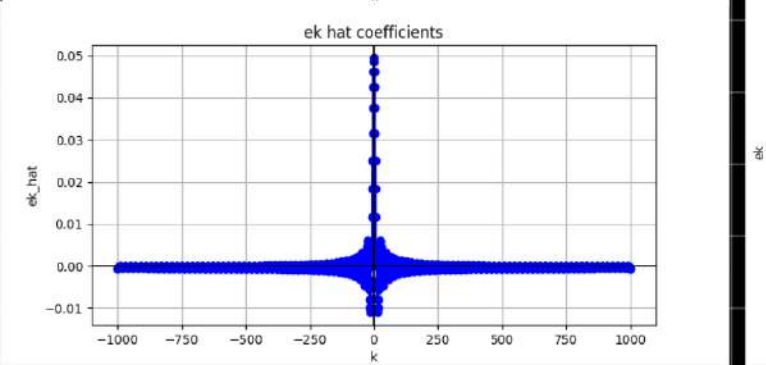
ניתן לראות שכל הגרפים נכון ובאמת

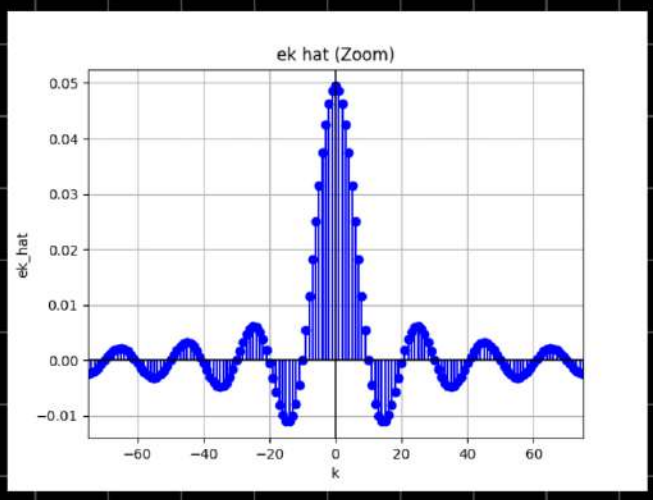
ושהשינוי הנורמלי אינו משמעותי

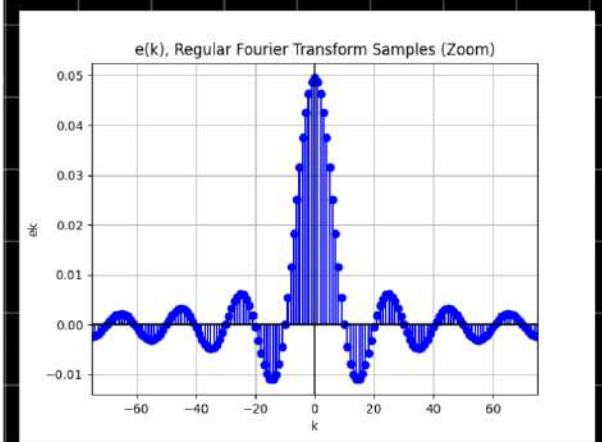












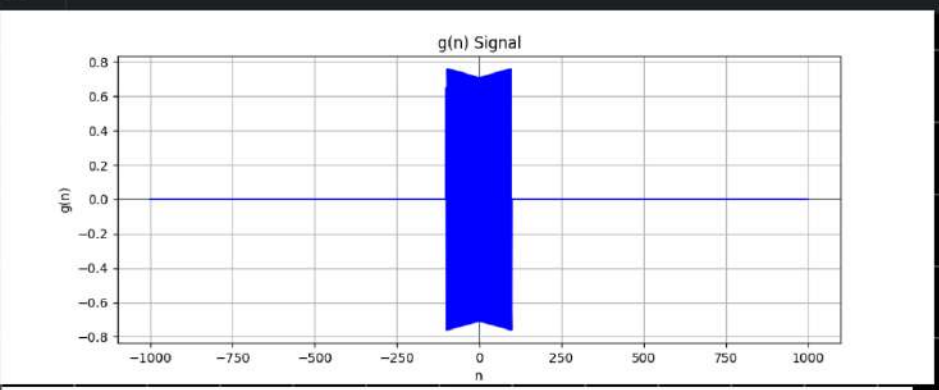
ח. נרצה עתה לבחון את ההכפלה בקוסינוס בזמן.

חשבו את : $g(n) = a(n)cos(2\pi \cdot 500 \cdot n/N)$

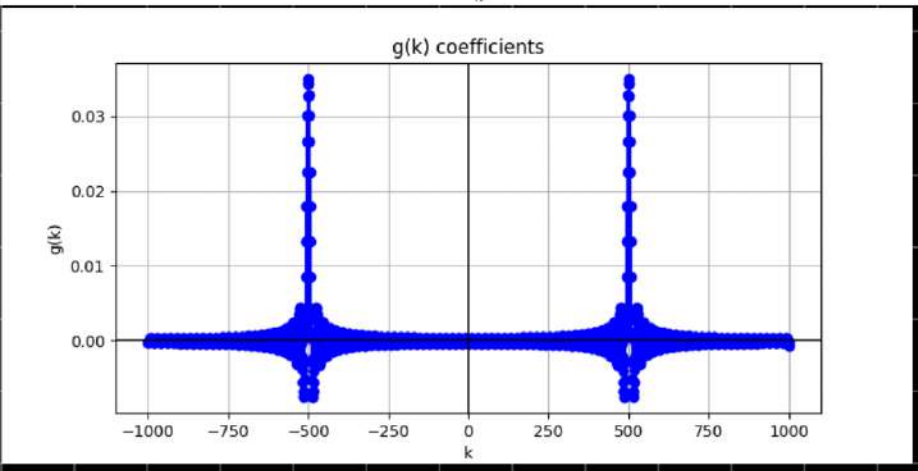
חשבו והציגו את האות בתדר g_k ע"י ההתמרה של $g(n)$. הצדיקו אנליטית את התוצאה.

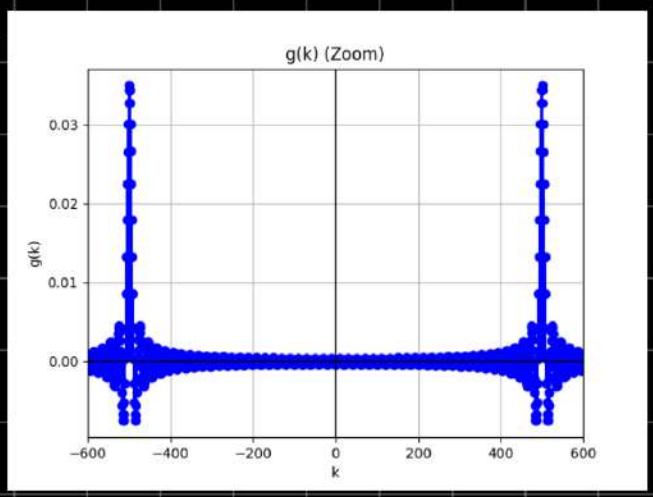
```python
# ====Q8====
# here we want to see what happens to a(n) if we multiply it by cosine in the time domain, and see it also it the
# frequency domain

g = np.zeros(N, dtype=complex)              # creating the array for g(n) signal
for i in range(N):                          # finding what happens to a(n) if we multiply it by cosine in the time domain
    g[i] = a[i] * math.cos(2 * np.pi * 500 * i / N)

g_real = np.real(g)                         # taking the real part of g(n)
gk = fourier_series(g)                      # finding g(k) with the inverse transform for g(n)
plot_signal_smth(g_real, n, title='g(n) Signal', xlabel='n', ylabel='g(n)')     # plotting g(n)
plot_disct(gk, n, title='g(k) coefficients', xlabel='k', ylabel='g(k)')         # plotting g(k)
plot_disct_zoom(gk, n, zoom:600, title='g(k) (Zoom)', xlabel='k', ylabel='g(k)')  # zoom on g(k)
```

הקוד של סעיף ח' ←



האות g(n) כאן ←





$$g(n) = a(n)\cos\left(\frac{2\pi 500 n}{N}\right) = a(n)\cdot\frac{1}{2}\left[e^{\frac{2\pi 500 n}{N}} + e^{-\frac{2\pi 500 n}{N}}\right]$$

$$g_n = \frac{1}{2}a(n)\, e^{\frac{2\pi 500 n}{N}} + \frac{1}{2}a(n)\, e^{-\frac{2\pi 500 n}{N}}$$

$$\Big\downarrow \mathcal{F}$$

$$\boxed{g_k = \frac{1}{2}a(k-500) + \frac{1}{2}a(k-500)}$$