Sprawozdanie z laboratorium Przetwarzanie Sygnałów i Obrazów

Ćwiczenie numer: 5

Temat: Systemy liniowe

Wykonujący ćwiczenie:

- Zaborowska Magda
- Wójtowicz Patryk

Studia dzienne I stopnia

Kierunek: Informatyka

Semestr: III Grupa zajęciowa: PS 12

Prowadzący ćwiczenie:
mgr inż. Dawid Najda

.....

OCENA

Data wykonania ćwiczenia

03.12.2021r.

.....

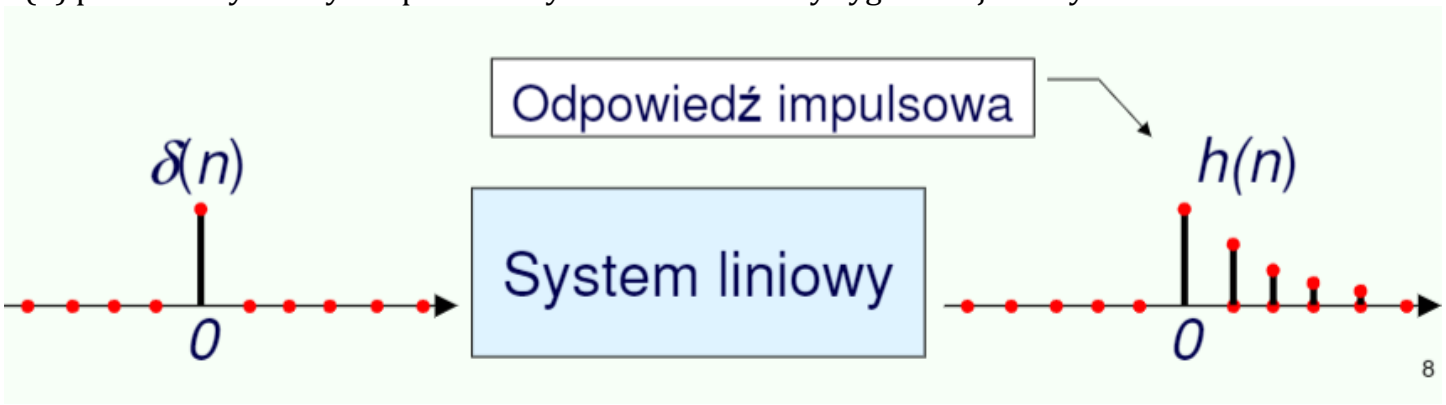
Data i podpis prowadzącego

Teoria

System LTI to system liniowy i niezmienny w czasie. Oznacza to, że jest liniowy ze względu na wszystkie swoje elementy w każdej chwili czasu. Systemy liniowe spełniają zasadę superpozycji:

$$H[ax_1(n)+bx_2(n)]=aH[x_1(n)]+bH[bx_2(n)]=ay_1(n)+by_2(n)$$

Liniowość oznacza, że odpowiedź systemu na sumę dwóch sygnałów będzie sumą odpowiedzi tego systemu na każdy z sygnałów podanych osobno, czyli dodanie do wejścia drugiego sygnału nie zakłóci przetwarzania w tym samym czasie pierwszego z nich. Dla systemów liniowych niezmiennych względem przesunięcia, znajomość odpowiedzi systemu na pobudzenie impulsowe d(n) pozwala wyznaczyć odpowiedź systemu na dowolny sygnał wejściowy.



Do najpowszechniejszych operacji przetwarzania sygnałów w dziedzinie czasu i przestrzeni należy obróbka sygnału wejściowego w celu poprawienia jego własności. Odbywa się to w procesie filtracji. Ogólnie, filtracja sprowadza się do wykonania pewnych operacji na zbiorze próbek wejściowych sąsiadujących z bieżącą próbką, a niekiedy także z wykorzystaniem pewnej ilości poprzednich próbek sygnału wyjściowego. Są różne sposoby charakteryzowania filtrów. Filtr „liniowy" jest liniowym przekształceniem próbek wejściowych, pozostałe filtry określane są jako „nieliniowe". Filtry liniowe spełniają zasadę superpozycji.

Rozmycie gaussowskie inaczej Gaussian blur jest to modyfikacja obrazu za pomocą filtru Gaussa. Używa się jego przeważnie w celu zmniejszenia szumów i zakłóceń w obrazie lub w celu zamazania detali. Wygładzanie gaussowskie jest również stosowane jako etap wstępnego przetwarzania obrazów w wizji komputerowej.

Podczas stosowania rozmycia gaussowskiego należy pamiętać, że większa intensywność rozmycia zmniejsza ostrość. Możesz przywrócić ostrość do obrazu, ograniczając promień rozmycia. Jest on mierzony w pikselach i określa liczbę sąsiadujących pikseli uwzględnianych przez funkcję gaussowską podczas obliczania rozmycia.
Standardowy algorytm wyostrzający:

$$I_p(x,y)=I_o(x,y)+A\cdot[I_o(x,y)-I_m(x,y)]$$

Io – obraz oryginalny
Im – obraz uśredniony, k*k – rozmiar maski filtru, np. 3x3
Ip – obraz przetworzony
A – współczynnik wyostrzenia

Zadanie 1

Treść zadania:

Zakładając, że x[n] – dowolna sekwencja wejściowa, sprawdzić analitycznie liniowość systemów opisanych następującymi równaniami:
a) S{x[n]} = 2x[n]
b) S{x[n]} = x[n] + 1
c) S{x[n]} = x[n + 1] – x[n]
Wygenerować dwa dowolne sygnały dyskretne x1[n], x2[n] (po 32 próbki każdy). Zweryfikować empirycznie liniowość (lub nieliniowość) systemów (a-c), porównując na wykresach odpowiedź sumy sygnałów S{x1[n] + x2[n]} z sumą odpowiedzi S{x1[n]} + S{x2[n]}. Co możesz powiedzieć o przyczynowości analizowanych systemów?

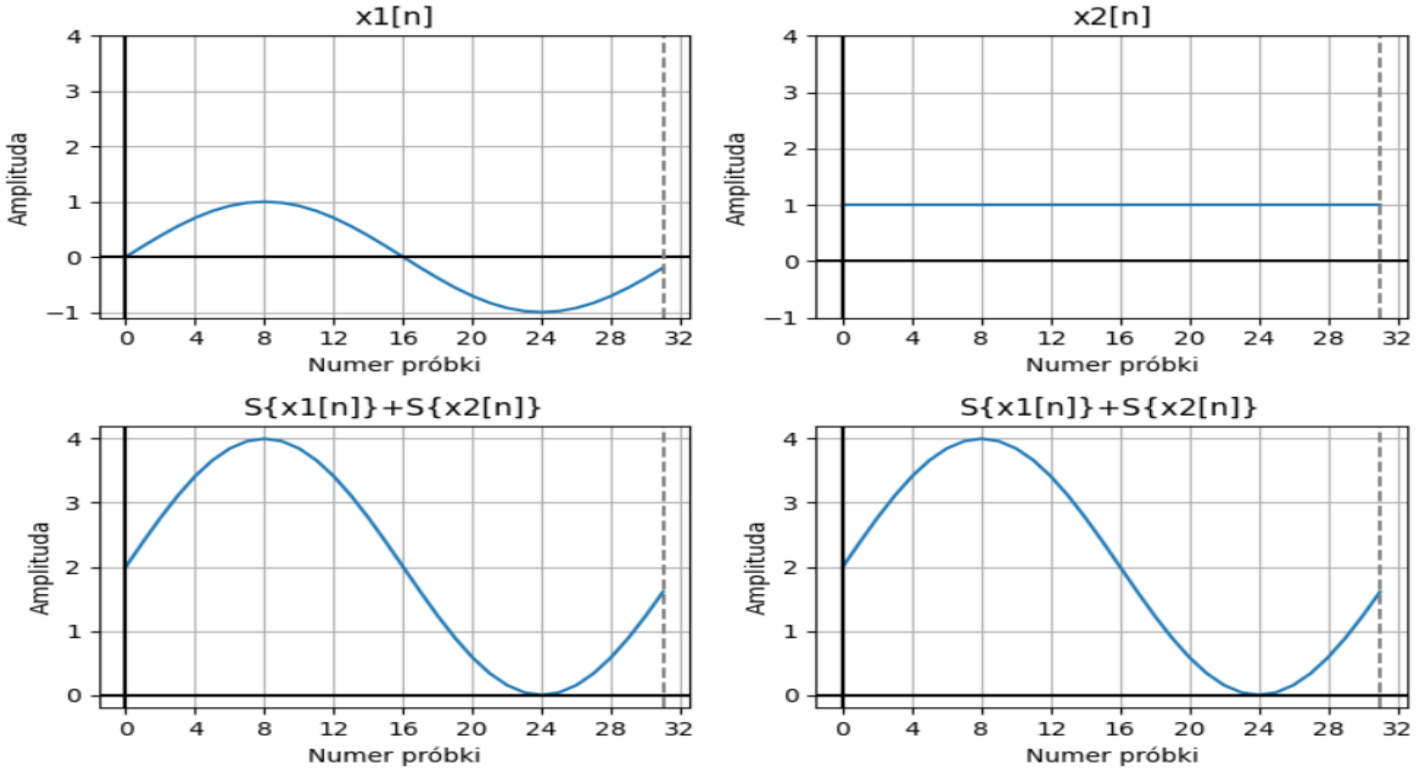
Realizacja w kodzie:

```python
import numpy as np
import matplotlib.pyplot as plt

def show(x1):
    plt.figure(figsize=(8,6),tight_layout=1)

    plt.subplot(2,2,1)
    plt.plot(x,y1)
    plt.title('x1[n]')
    plt.xlabel('Numer próbki')
    plt.ylabel('Amplituda')
    plt.axhline(y=0,color = "k")
    plt.axvline(x=0,color = "k")
    plt.axvline(x=0.97,color = "grey",linestyle='--')
    plt.grid(True,which='both')
    plt.yticks(np.arange(-1,5,1))
    plt.xticks([0.0,0.125,0.25,0.375,0.5,0.625,0.75,0.875,1.0],[0,4,8,12,16,20,24,28,32])

    plt.subplot(2,2,2)
    plt.plot(x,y2)
    plt.title('x2[n]')
    plt.xlabel('Numer próbki')
    plt.ylabel('Amplituda')
    plt.axhline(y=0,color = "k")
    plt.axvline(x=0,color = "k")
    plt.axvline(x=0.97,color = "grey",linestyle='--')
    plt.grid(True,which='both')
    plt.yticks(np.arange(-1,5,1))
    plt.xticks([0.0,0.125,0.25,0.375,0.5,0.625,0.75,0.875,1.0],[0,4,8,12,16,20,24,28,32])
    plt.subplot(2,2,3)
    plt.plot(x1,S1+S2)
    plt.title('S{x1[n]}+S{x2[n]}')
    plt.xlabel('Numer próbki')
    plt.ylabel('Amplituda')
    plt.axhline(y=0,color = "k")
    plt.axvline(x=0,color = "k")
    plt.axvline(x=0.97,color = "grey",linestyle='--')
    plt.grid(True,which='both')
    if i!=2: plt.yticks(np.arange(0,5,1))
    plt.xticks([0.0,0.125,0.25,0.375,0.5,0.625,0.75,0.875,1.0],[0,4,8,12,16,20,24,28,32])
```

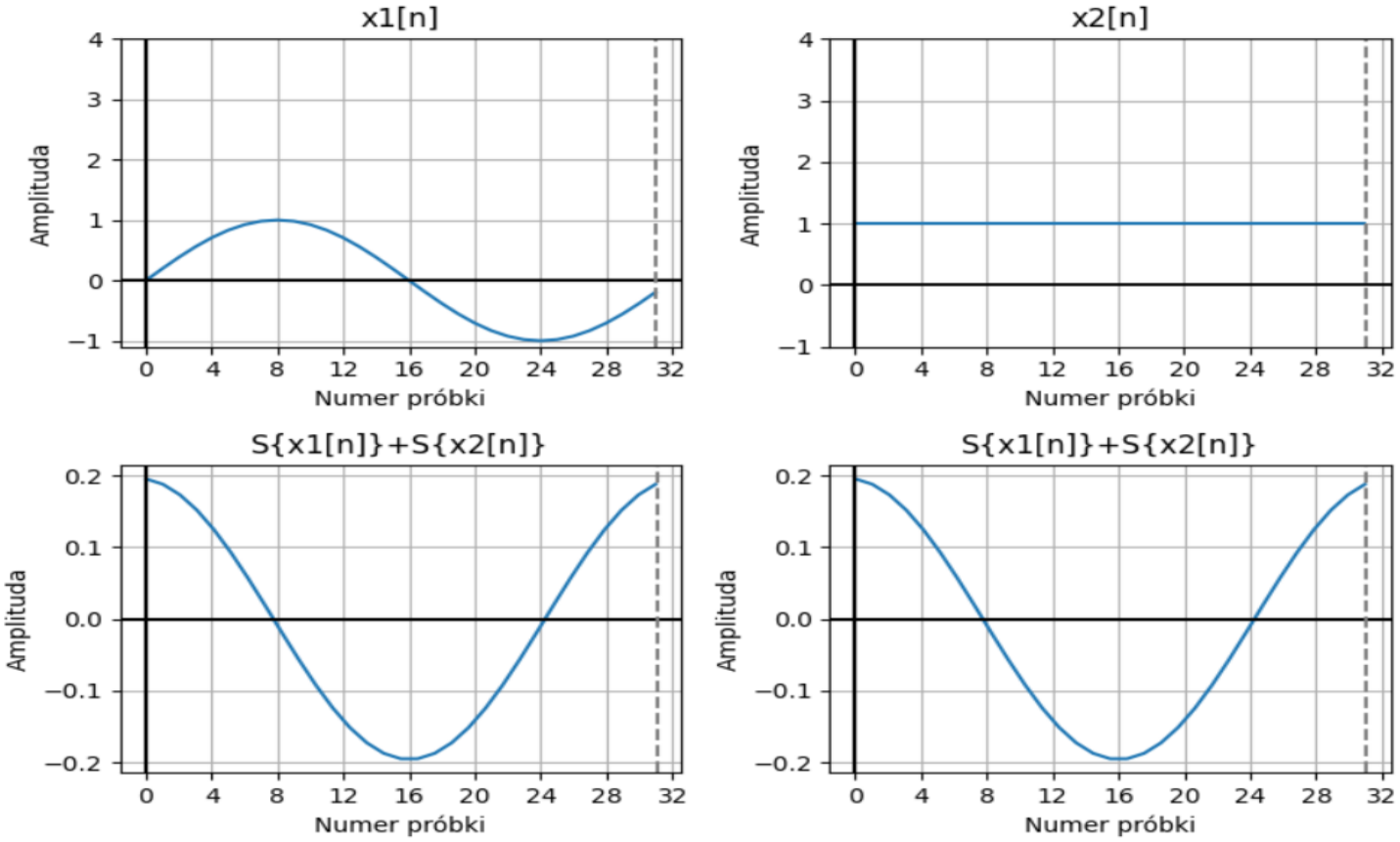
```
    plt.subplot(2,2,4)
    if i==0: plt.plot(x1,fun_a(y1+y2))
    if i==1: plt.plot(x1,fun_b(y1+y2))
    if i==2: plt.plot(x1,fun_c(y1+y2))
    plt.title('S{x1[n]}+S{x2[n]}')
    plt.xlabel('Numer próbki')
    plt.ylabel('Amplituda')
    plt.axhline(y=0,color = "k")
    plt.axvline(x=0,color = "k")
    plt.axvline(x=0.97,color = "grey",linestyle='--')
    plt.grid(True,which='both')
    if i!=2: plt.yticks(np.arange(0,5,1))
    plt.xticks([0.0,0.125,0.25,0.375,0.5,0.625,0.75,0.875,1.0],[0,4,8,12,16,20,24,28,32])

    plt.show()

def fun_a(f):
    f = [f[i]*2
        for i in range (len(f))]
    return np.array(f)

def fun_b(f):
    f = [f[i]+1
        for i in range (len(f))]
    return np.array(f)

def fun_c(f):
    f = [f[i+1]-f[i]
        for i in range (len(f)-1)]
    return np.array(f)

N = 32
x = np.arange(N)/N
y1 = np.sin(2*np.pi*x)
y2 = np.ones(32)
```

```
i=0
#Podpunkt A
S1 = fun_a(y1)
S2 = fun_a(y2)
show(x)

i+=1
#Podpunkt B
S1 = fun_b(y1)
S2 = fun_b(y2)
show(x)

i+=1
#Podpunkt C
x1 = np.arange(31)/31
S1 = fun_c(y1)
S2 = fun_c(y2)
show(x1)
```

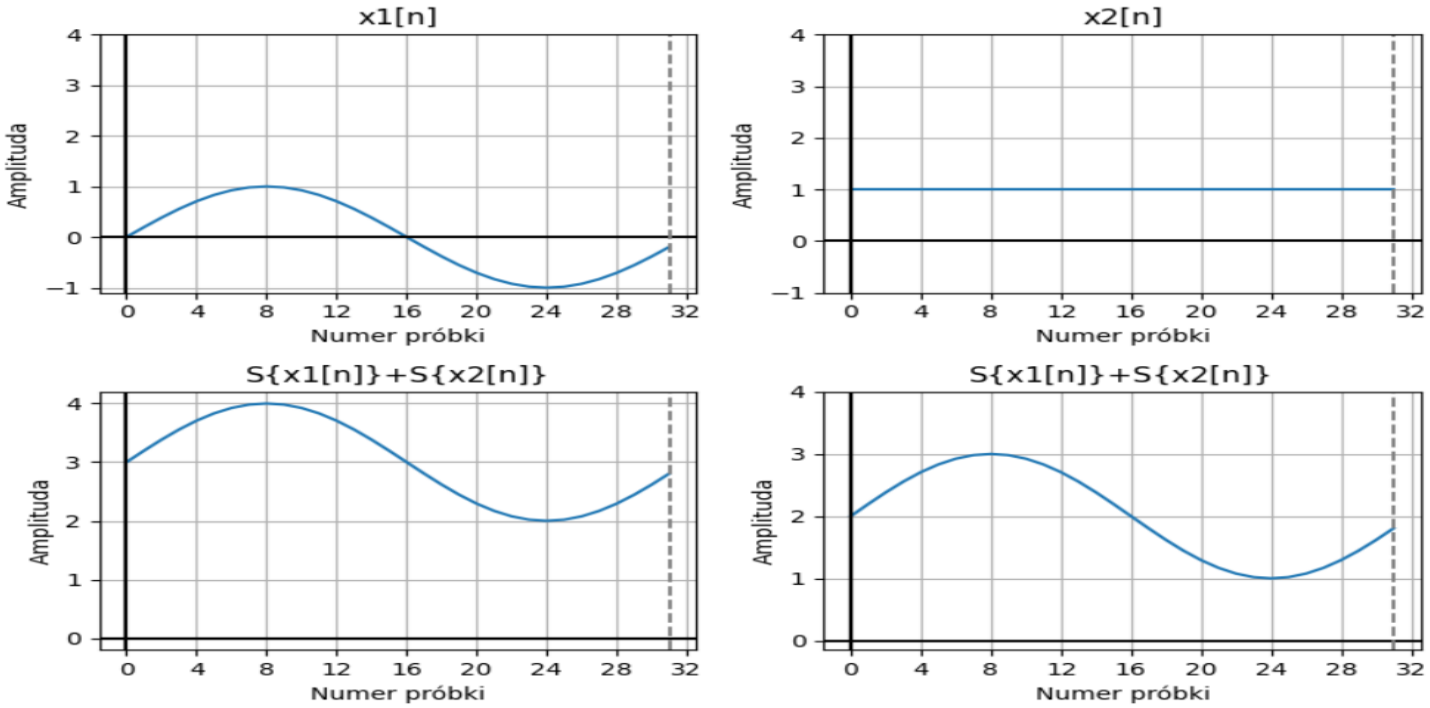
Wyniki:



Wykres 2 Wyniki 1.a



Wykres 1 Wyniki 1.b



Wykres 3 Wyniki 1.c

Zadanie 2

Treść zadania:

Wygenerować sygnały x1[n] = sin(2πn/N), x2[n] = sin(4πn/N) oraz h[n] = δ[n − k], gdzie k = {0, 16, 32}, N = 64, (założyć, że 0 ¬ n < N). Wyznaczyć splot liniowy sygnałów x1[n], x2[n] z sygnałem h[n] oraz samych ze sobą. Sporządzić wykresy, sprawdzić czy operacje splotu są przemienne oraz liniowe (dla ustalonego sygnału h[n])

Realizacja w kodzie

```
def show():
    plt.figure(figsize=(8,6),tight_layout=1)

    plt.subplot(3,3,1)
    plt.plot(x,y1)
    plt.title('y1')
    plt.xlabel('Numer próbki')
    plt.ylabel('Amplituda')
    plt.axhline(y=0,color = "grey",linestyle='--')
    plt.axvline(x=0,color = "grey",linestyle='--')
    plt.axvline(x=63,color = "grey",linestyle='--')
    plt.grid(True,which='both')
    plt.yticks(np.arange(-1,1.1,0.5))
    plt.xlim(0,80)

    plt.subplot(3,3,2)
    plt.plot(np.convolve(y1,h))
    plt.title('Splot Liniowy y1*h')
    plt.yticks(np.arange(-1,1.1,0.5))
    plt.xlabel('Numer próbki')
    plt.ylabel('Amplituda')
    plt.axhline(y=0,color = "grey",linestyle='--')
    plt.axvline(x=0,color = "grey",linestyle='--')
    plt.axvline(x=126,color = "grey",linestyle='--')
    plt.grid(True,which='both')
    plt.xlim(0,150)

    plt.subplot(3,3,3)
    plt.plot(np.convolve(h,y1))
    plt.title('Splot Liniowy h*y1')
    plt.xlabel('Numer próbki')
    plt.ylabel('Amplituda')
    plt.axhline(y=0,color = "grey",linestyle='--')
    plt.axvline(x=0,color = "grey",linestyle='--')
    plt.axvline(x=126,color = "grey",linestyle='--')
    plt.grid(True,which='both')
    plt.yticks(np.arange(-1,1.1,0.5))
    plt.xlim(0,150)

    plt.subplot(3,3,4)
    plt.plot(x,y2)
    plt.title('y2')
    plt.xlabel('Numer próbki')
    plt.ylabel('Amplituda')
    plt.axhline(y=0,color = "grey",linestyle='--')
    plt.axvline(x=0,color = "grey",linestyle='--')
    plt.axvline(x=63,color = "grey",linestyle='--')
    plt.grid(True,which='both')
    plt.yticks(np.arange(-1,1.1,0.5))
    plt.xlim(0,80)
```

```
    plt.subplot(3,3,5)
    plt.plot(np.convolve(y2,h))
    plt.title('Splot Liniowy y2*h')
    plt.xlabel('Numer próbki')
    plt.ylabel('Amplituda')
    plt.axhline(y=0,color = "grey",linestyle='--')
    plt.axvline(x=0,color = "grey",linestyle='--')
    plt.axvline(x=126,color = "grey",linestyle='--')
    plt.grid(True,which='both')
    plt.yticks(np.arange(-1,1.1,0.5))
    plt.xlim(0,150)

    plt.subplot(3,3,6)
    plt.plot(np.convolve(h,y2))
    plt.title('Splot Liniowy h*y2')
    plt.xlabel('Numer próbki')
    plt.ylabel('Amplituda')
    plt.axhline(y=0,color = "grey",linestyle='--')
    plt.axvline(x=0,color = "grey",linestyle='--')
    plt.axvline(x=126,color = "grey",linestyle='--')
    plt.grid(True,which='both')
    plt.yticks(np.arange(-1,1.1,0.5))
    plt.xlim(0,150)


    plt.subplot(3,3,7)
    plt.stem(h,use_line_collection='true')
    plt.title('h')
    plt.xlabel('Numer próbki')
    plt.ylabel('Amplituda')
    plt.axhline(y=0,color = "grey",linestyle='--')
    plt.axvline(x=0,color = "grey",linestyle='--')
    plt.axvline(x=63,color = "grey",linestyle='--')
    plt.grid(True,which='both')
    plt.yticks(np.arange(-1,1.1,0.5))
    plt.xlim(-1,80)
```

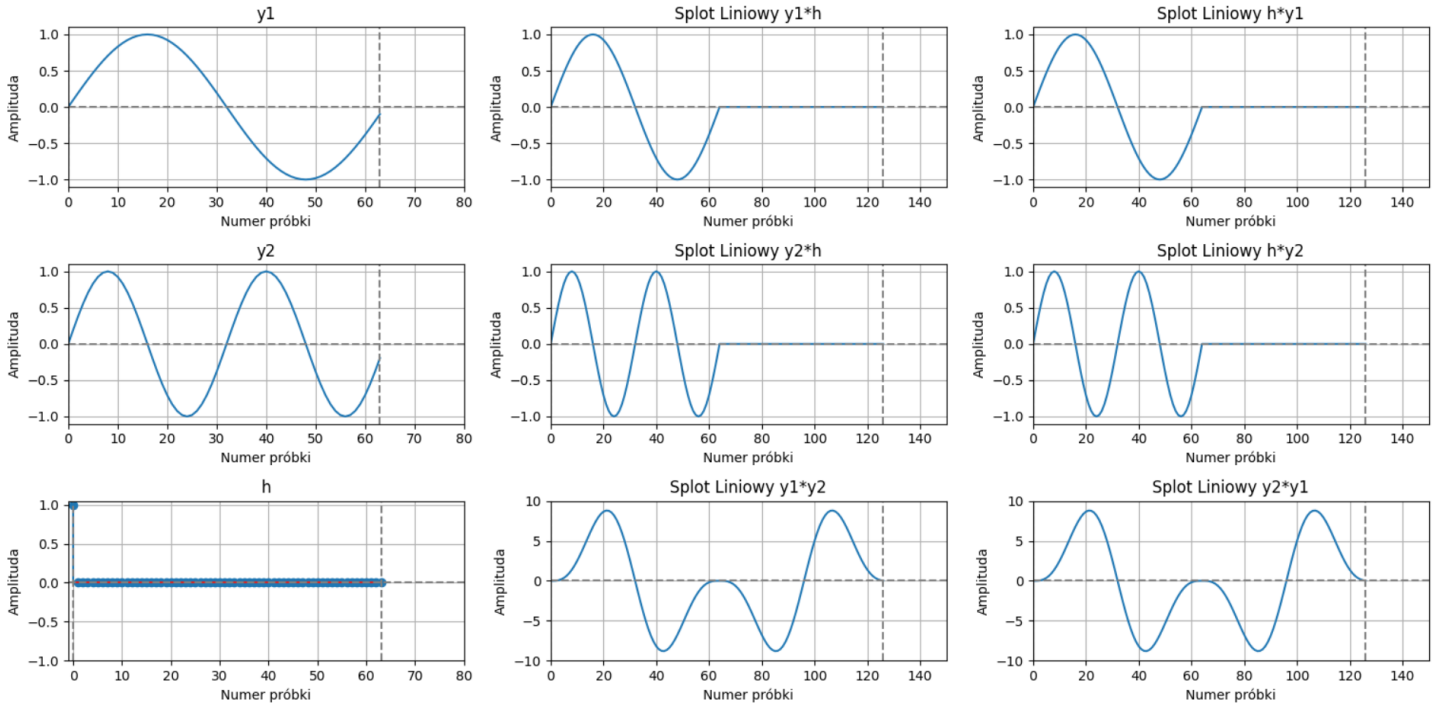
```
    plt.subplot(3,3,8)
    plt.plot(np.convolve(y1,y2))
    plt.title('Splot Liniowy y1*y2')
    plt.xlabel('Numer próbki')
    plt.ylabel('Amplituda')
    plt.axhline(y=0,color = "grey",linestyle='--')
    plt.axvline(x=0,color = "grey",linestyle='--')
    plt.axvline(x=126,color = "grey",linestyle='--')
    plt.grid(True,which='both')
    plt.yticks(np.arange(-10,10.1,5))
    plt.xlim(0,150)

    plt.subplot(3,3,9)
    plt.plot(np.convolve(y2,y1))
    plt.title('Splot Liniowy y2*y1')
    plt.xlabel('Numer próbki')
    plt.ylabel('Amplituda')
    plt.axhline(y=0,color = "grey",linestyle='--')
    plt.axvline(x=0,color = "grey",linestyle='--')
    plt.axvline(x=126,color = "grey",linestyle='--')
    plt.grid(True,which='both')
    plt.yticks(np.arange(-10,10.1,5))
    plt.xlim(0,150)

    plt.show()
```

```
def show2():
    plt.figure(figsize=(8,6),tight_layout=1)

    plt.subplot(3,2,1)
    plt.plot(x,y1)
    plt.title('y1')
    plt.xlabel('Numer próbki')
    plt.ylabel('Amplituda')
    plt.axhline(y=0,color = "grey",linestyle='--')
    plt.axvline(x=0,color = "grey",linestyle='--')
    plt.axvline(x=63,color = "grey",linestyle='--')
    plt.grid(True,which='both')
    plt.yticks(np.arange(-1,1.1,0.5))
    plt.xlim(0,80)

    plt.subplot(3,2,2)
    plt.plot(np.convolve(y1+y2,h))
    plt.title('Splot sumy (y1+y2)*h')
    plt.xlabel('Numer próbki')
    plt.ylabel('Amplituda')
    plt.axhline(y=0,color = "grey",linestyle='--')
    plt.axvline(x=0,color = "grey",linestyle='--')
    plt.axvline(x=126,color = "grey",linestyle='--')
    plt.grid(True,which='both')
    plt.yticks(np.arange(-2,2.1,1))
    plt.xlim(0,150)

    plt.subplot(3,2,3)
    plt.plot(x,y2)
    plt.title('x2')
    plt.xlabel('Numer próbki')
    plt.ylabel('Amplituda')
    plt.axhline(y=0,color = "grey",linestyle='--')
    plt.axvline(x=0,color = "grey",linestyle='--')
    plt.axvline(x=63,color = "grey",linestyle='--')
    plt.grid(True,which='both')
    plt.yticks(np.arange(-1,1.1,0.5))
```

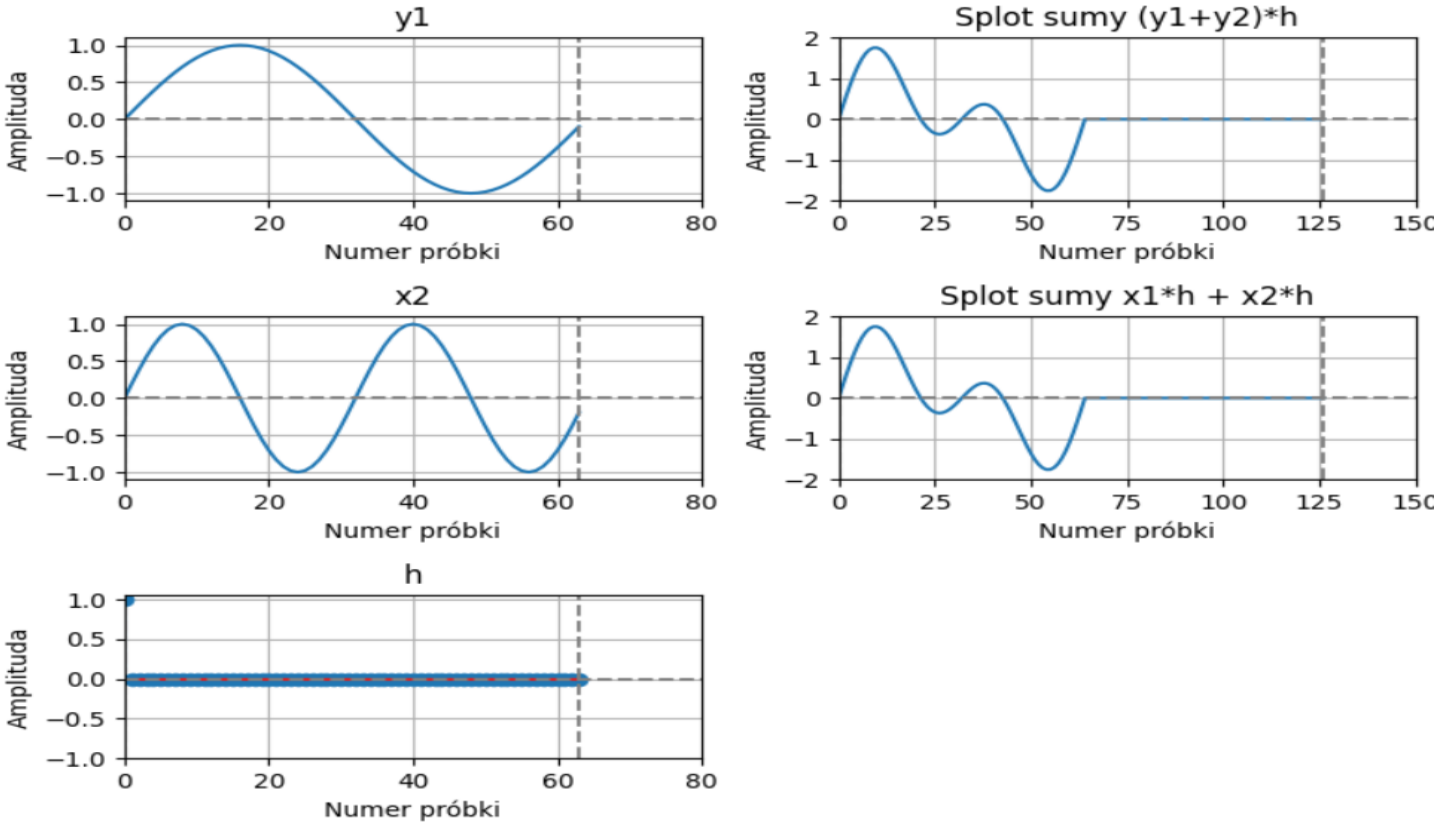
```
    plt.subplot(3,2,4)
    plt.plot(np.convolve(y1,h)+np.convolve(y2,h))
    plt.title('Splot sumy x1*h + x2*h')
    plt.xlabel('Numer próbki')
    plt.ylabel('Amplituda')
    plt.axhline(y=0,color = "grey",linestyle='--')
    plt.axvline(x=0,color = "grey",linestyle='--')
    plt.axvline(x=126,color = "grey",linestyle='--')
    plt.grid(True,which='both')
    plt.yticks(np.arange(-2,2.1,1))
    plt.xlim(0,150)

    plt.subplot(3,2,5)
    plt.stem(h,use_line_collection='true')
    plt.title('h')
    plt.xlabel('Numer próbki')
    plt.ylabel('Amplituda')
    plt.axhline(y=0,color = "grey",linestyle='--')
    plt.axvline(x=0,color = "grey",linestyle='--')
    plt.axvline(x=63,color = "grey",linestyle='--')
    plt.grid(True,which='both')
    plt.yticks(np.arange(-1,1.1,0.5))
    plt.xlim(0,80)

    plt.show()

N = 64
k = 0
x = np.arange(N)
y1 = np.sin(2*np.pi*x/N)
y2 = np.sin(4*np.pi*x/N)
h = np.zeros(N)/N
h[k] = 1
```

```
195    show()
196    show2()
197
198    h[k] = 0
199    k=16
200    h[k] = 1
201
202    show()
203    show2()
204
205    h[k] = 0
206    k=32
207    h[k] = 1
208
209    show()
210    show2()
```

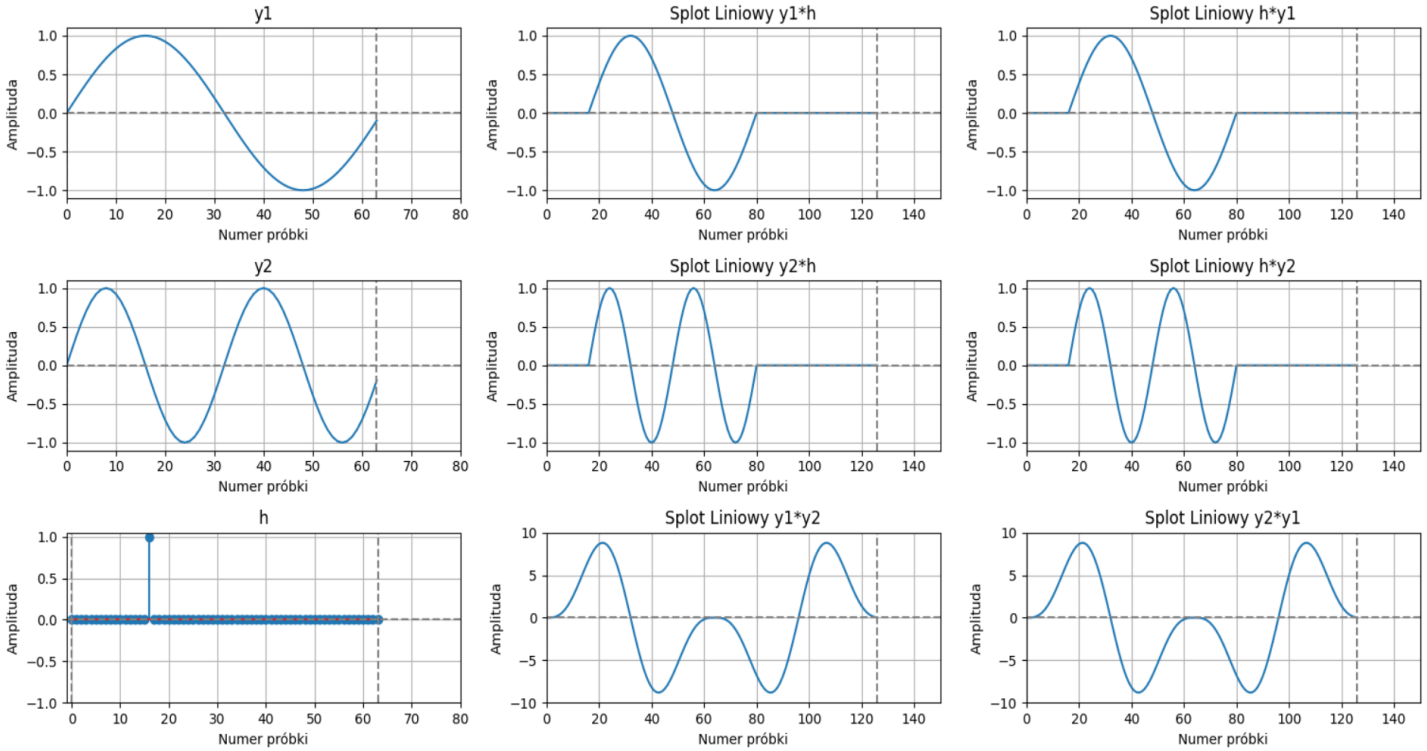
Wyniki



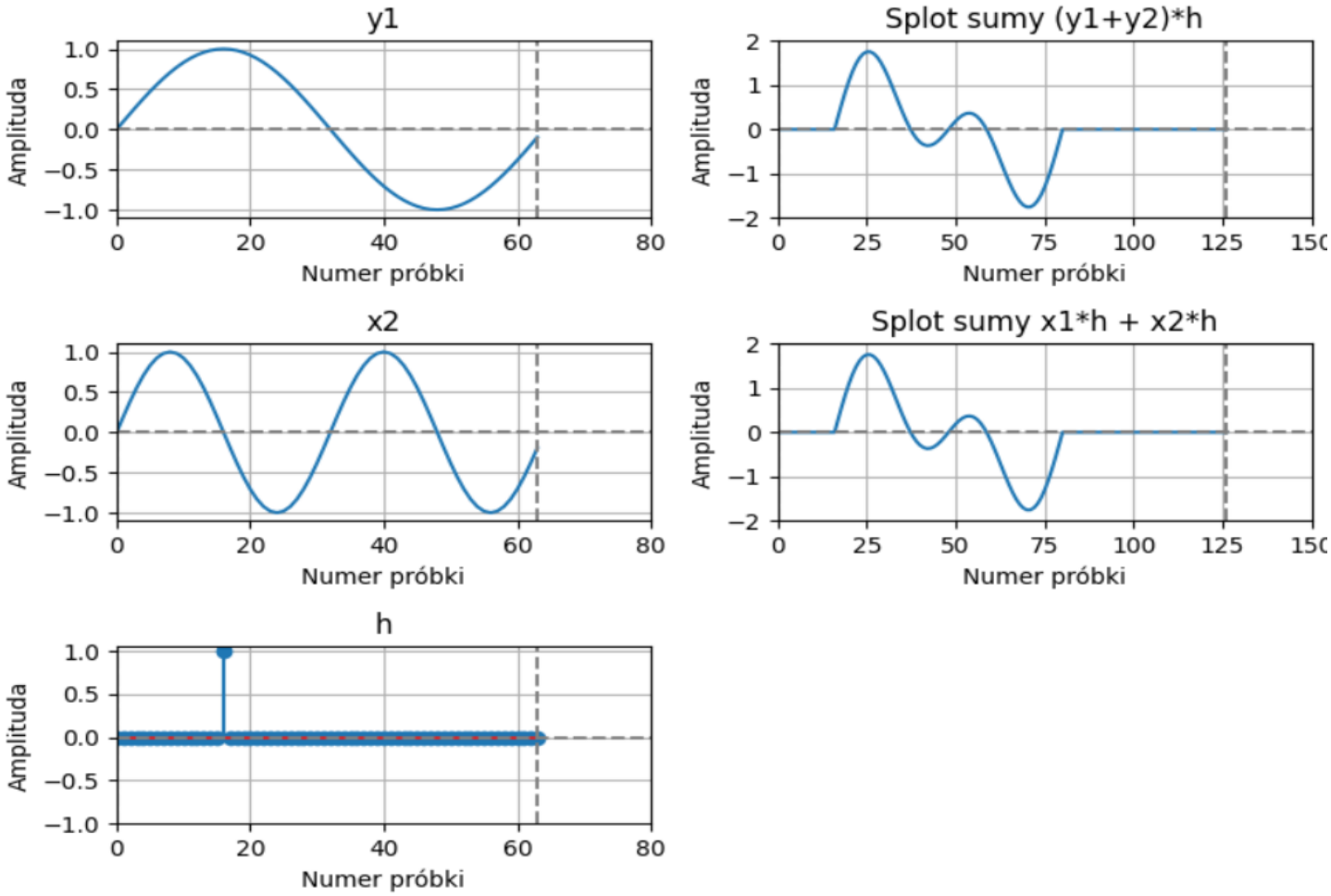
Wykres 4 Operacje splotu dla zmiennej k = 0 oraz dla ustalonego sygnału h[n]



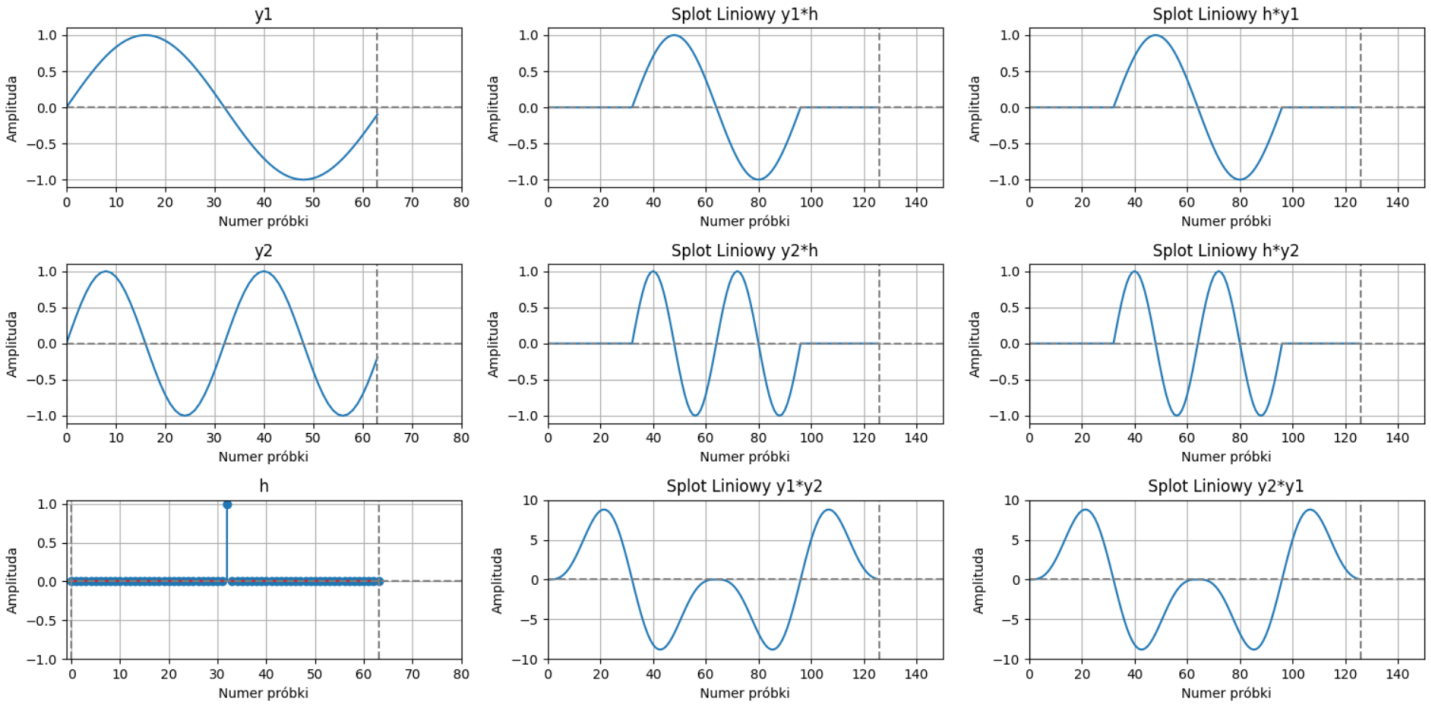
Wykres 5 Dla zmiennej k = 0, oraz dla ustalonego sygnału h[n]



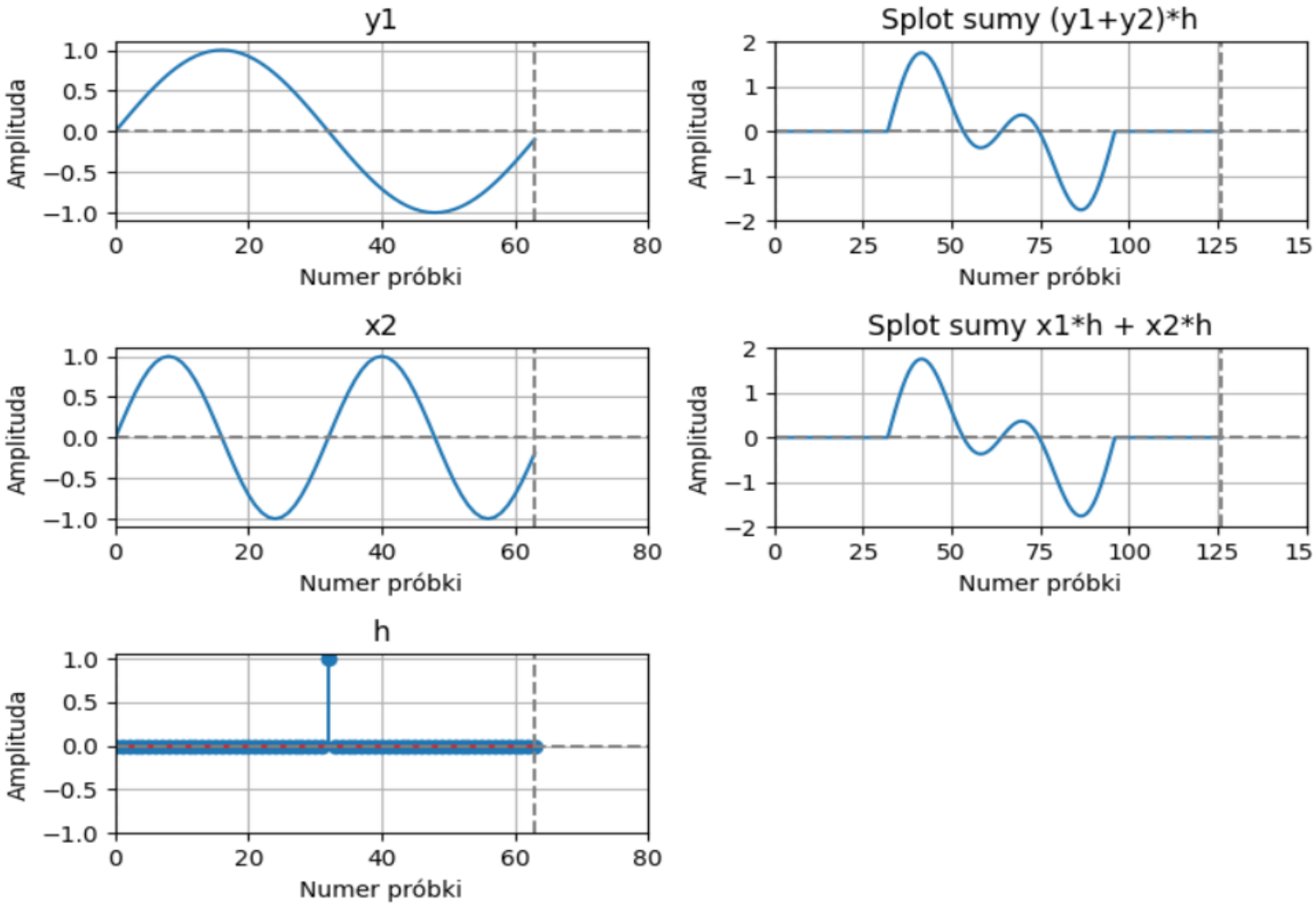
Wykres 6 Operacje splotu dla zmiennej k = 16 oraz dla ustalonego sygnału h[n]



Wykres 7 Dla zmiennej k = 16, oraz dla ustalonego sygnału h[n]



Wykres 8 Operacje splotu dla zmiennej k = 32 oraz dla ustalonego sygnału h[n]



Wykres 9 : Dla zmiennej k = 32, oraz dla ustalonego sygnału h[n]

Zadanie 3

Treść zadania:

Wyznaczyć 64-punktowe DFT sygnału x1[n] z zadania 5.2 oraz sygnału h[n] = exp(−n/10), obliczyć iloczyn widm zespolonych (tj. G(k) = X1(k)· H(k) dla k = 0, 1, . . . , 63), wyznaczyć IDFT iloczynu G(k). Uzyskany wynik porównać ze splotem liniowym (64-punktowym) sygnałów x1[n] i h[n].

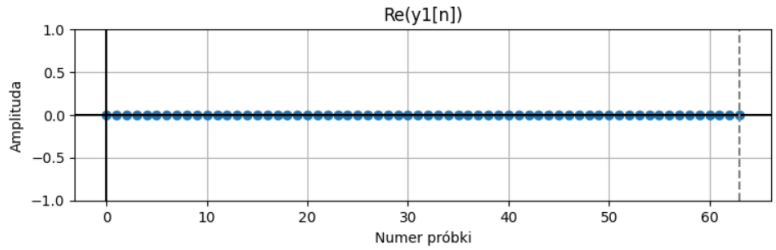
Realizacja w kodzie

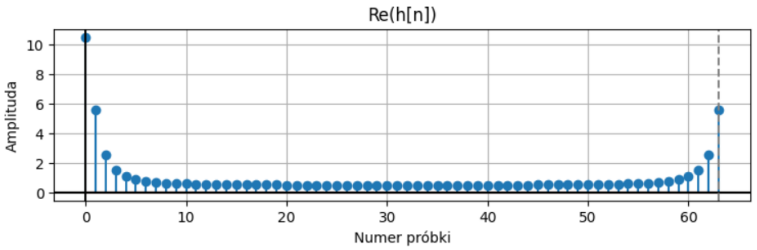
```
import numpy as np
import matplotlib.pyplot as plt

N = 64
x = np.arange(N)
y1 = np.sin(2*np.pi*x/N)
h = np.exp(-x/10)
fy1 = np.fft.fft(y1)
fh = np.fft.fft(h)
z = fy1 * fh
fz = np.fft.fft(z)
```

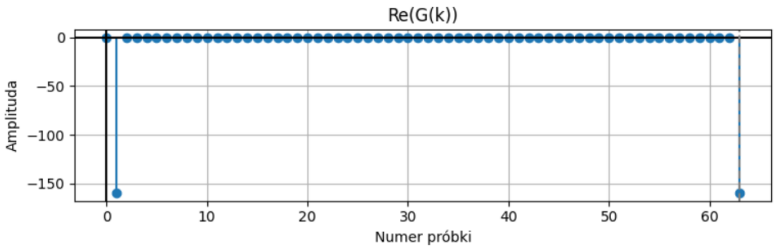
```
plt.subplot(3,2,1)
plt.stem(np.real(fy1),use_line_collection='true')
plt.title('Re(y1[n])')
plt.xlabel('Numer próbki')
plt.ylabel('Amplituda')
plt.axhline(y=0,color = "k",)
plt.axvline(x=0,color = "k")
plt.axvline(x=63,color = "grey",linestyle='--')
plt.grid(True,which='both')
plt.ylim(-1,1)

plt.subplot(3,2,2)
plt.stem(np.real(fh),use_line_collection='true')
plt.title('Re(h[n])')
plt.xlabel('Numer próbki')
plt.ylabel('Amplituda')
plt.axhline(y=0,color = "k",)
plt.axvline(x=0,color = "k")
plt.axvline(x=63,color = "grey",linestyle='--')
plt.grid(True,which='both')

plt.subplot(3,2,3)
plt.stem(np.real(z),use_line_collection='true')
plt.title('Re(G(k))')
plt.xlabel('Numer próbki')
plt.ylabel('Amplituda')
plt.axhline(y=0,color = "k",)
plt.axvline(x=0,color = "k")
plt.axvline(x=63,color = "grey",linestyle='--')
plt.grid(True,which='both')
```

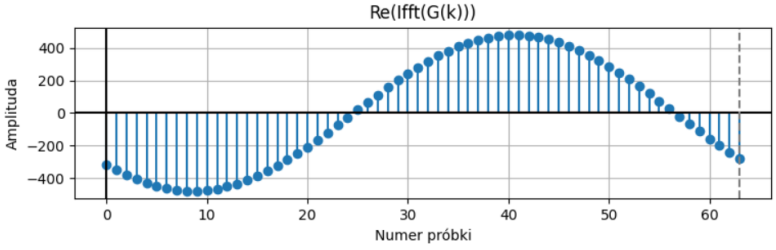
```
plt.subplot(3,2,5)
plt.stem(np.real(fz),use_line_collection='true')
plt.title('Re(Ifft(G(k)))')
plt.xlabel('Numer próbki')
plt.ylabel('Amplituda')
plt.axhline(y=0,color = "k",)
plt.axvline(x=0,color = "k")
plt.axvline(x=63,color = "grey",linestyle='--')
plt.grid(True,which='both')

plt.subplot(3,2,6)
plt.stem(np.convolve(y1,h),use_line_collection='true')
plt.title('Splot liniowy y1*h')
plt.xlabel('Numer próbki')
plt.ylabel('Amplituda')
plt.axhline(y=0,color = "k",)
plt.axvline(x=0,color = "k")
plt.axvline(x=126,color = "grey",linestyle='--')
plt.grid(True,which='both')

plt.show()
```

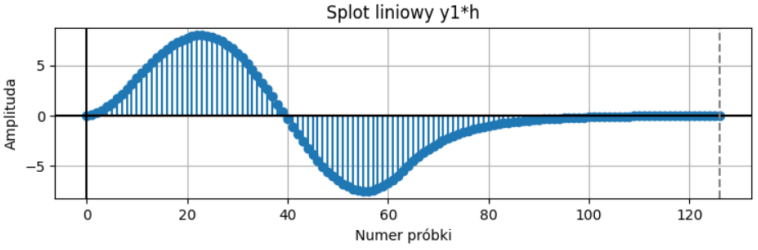
Wyniki











Zadanie 4

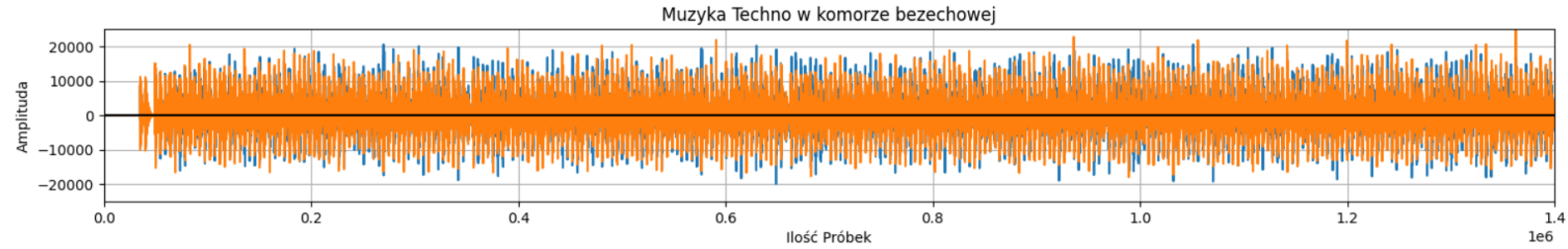
Treść zadania:

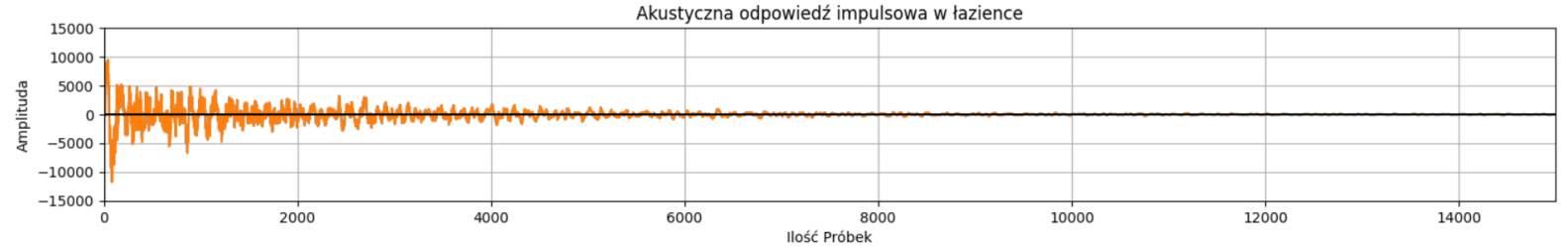
Dokonać splotu nagrania dźwiękowego dokonanego w komorze bezechowej x[n] (ang. anechoic chamber) z akustyczną odpowiedzią impulsową dwóch dowolnie wybranych pomieszczeń h[n] (sala koncertowa, korytarz, itp.). Jak różnią się słuchowo poszczególne sygnały (x[n], h[n] oraz ich splot)?

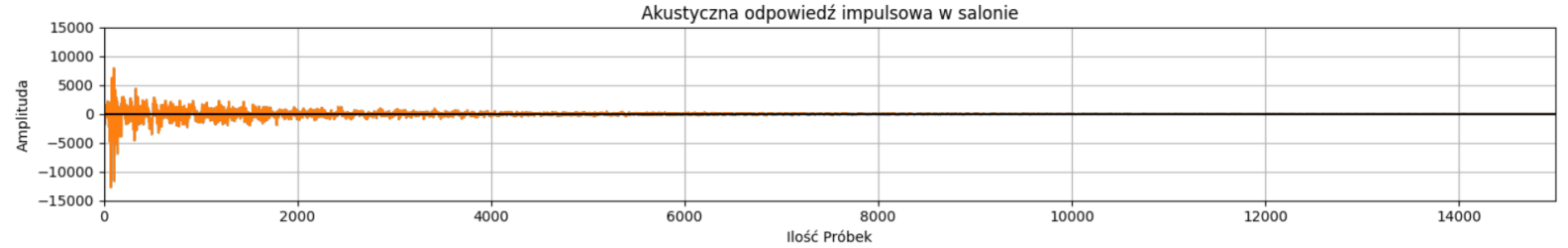
Realizacja w kodzie:

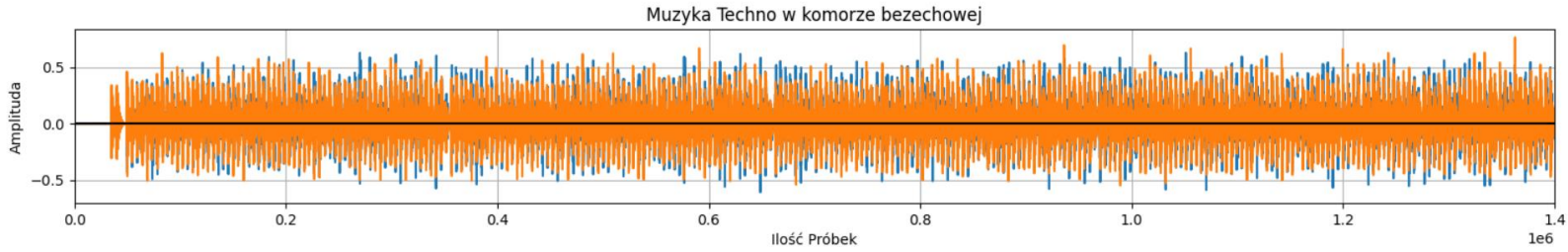
```python
import numpy as np
import matplotlib.pyplot as plt
import scipy.io.wavfile as siw
import scipy.signal as ss
import soundfile as sf
import utility as u

fs, a = siw.read('Universal Linear Accelerator.wav')
fs, h1 = siw.read('BathRoom.wav')
fs, h2 = siw.read('Living Room.wav')

plt.figure(figsize=(8,6),tight_layout=1)

plt.subplot(3,1,1)
plt.plot(a)
plt.title('Muzyka Techno w komorze bezechowej')
plt.ylabel('Amplituda')
plt.xlabel('Ilość Próbek')
plt.axhline(y=0,color = "k",)
plt.axvline(x=0,color = "k")
plt.grid(True,which='both')
plt.xlim(0,1400000)
plt.ylim(-25000,25000)

plt.subplot(3,1,2)
plt.plot(h1)
plt.title('Akustyczna odpowiedź impulsowa w łazience')
plt.ylabel('Amplituda')
plt.xlabel('Ilość Próbek')
plt.axhline(y=0,color = "k",)
plt.axvline(x=0,color = "k")
plt.grid(True,which='both')
plt.xlim(0,15000)
plt.ylim(-15000,15000)

plt.subplot(3,1,3)
plt.plot(h2)
plt.title('Akustyczna odpowiedź impulsowa w salonie')
plt.ylabel('Amplituda')
plt.xlabel('Ilość Próbek')
plt.axhline(y=0,color = "k",)
plt.axvline(x=0,color = "k")
plt.grid(True,which='both')
plt.xlim(0,15000)
plt.ylim(-15000,15000)

plt.show()

a = u.pcm2float(a,'float32')
h1 = u.pcm2float(h1,'float32')
h2 = u.pcm2float(h2,'float32')

y1 = ss.convolve(a,h1)
y2 = ss.convolve(a,h2)

plt.figure(figsize=(8,6),tight_layout=1)

plt.subplot(3,1,1)
plt.plot(a)
plt.title('Muzyka Techno w komorze bezechowej')
plt.ylabel('Amplituda')
plt.xlabel('Ilość Próbek')
plt.axhline(y=0,color = "k",)
plt.axvline(x=0,color = "k")
plt.grid(True,which='both')
plt.xlim(0,1400000)


plt.subplot(3,1,2)
plt.plot(y1)
plt.title('Muzyka Techno w komorze
plt.ylabel('Amplituda')
plt.xlabel('Ilość Próbek')
plt.axhline(y=0,color = "k",)
plt.axvline(x=0,color = "k")
plt.grid(True,which='both')
plt.xlim(0,1400000)


plt.subplot(3,1,3)
plt.plot(y2)
plt.title('Muzyka Techno w komorze bezechowej + Akustyczna odpowiedź impulsowa w salonie')
plt.ylabel('Amplituda')
plt.xlabel('Ilość Próbek')
plt.axhline(y=0,color = "k",)
plt.axvline(x=0,color = "k")
plt.grid(True,which='both')
plt.xlim(0,1400000)

plt.show()

sf.write("jeden.wav",y1,fs)
sf.write("dwa.wav",y2,fs)
```

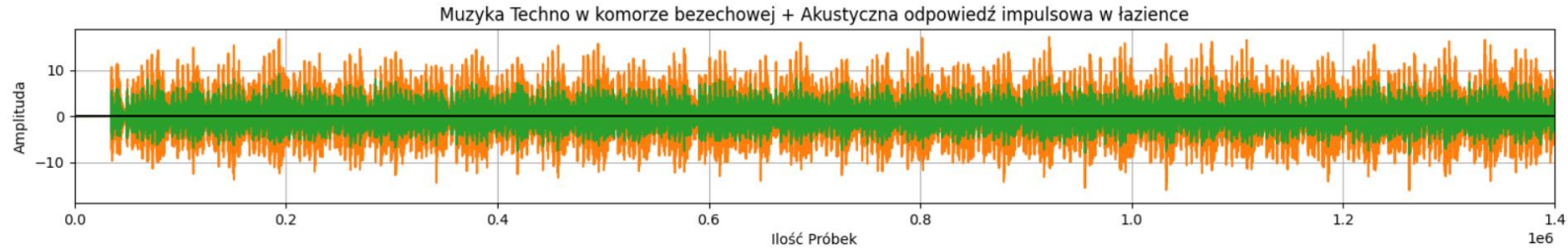
Wyniki:

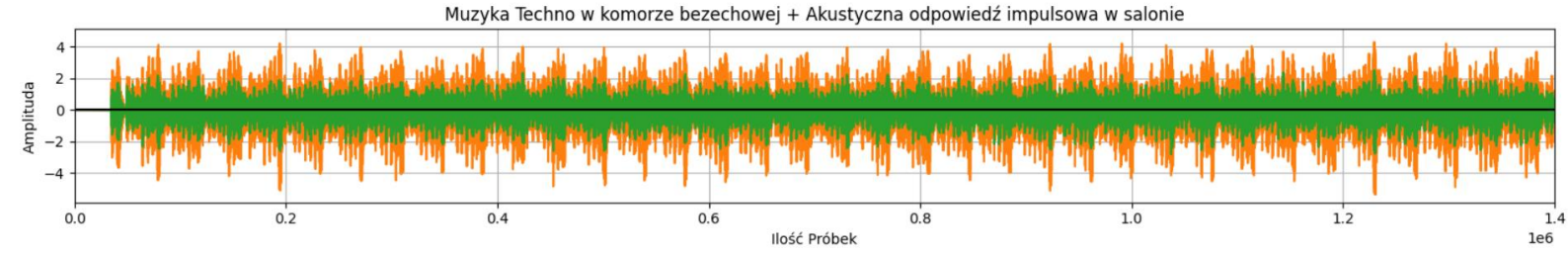












Zadanie 5

Treść zadania:

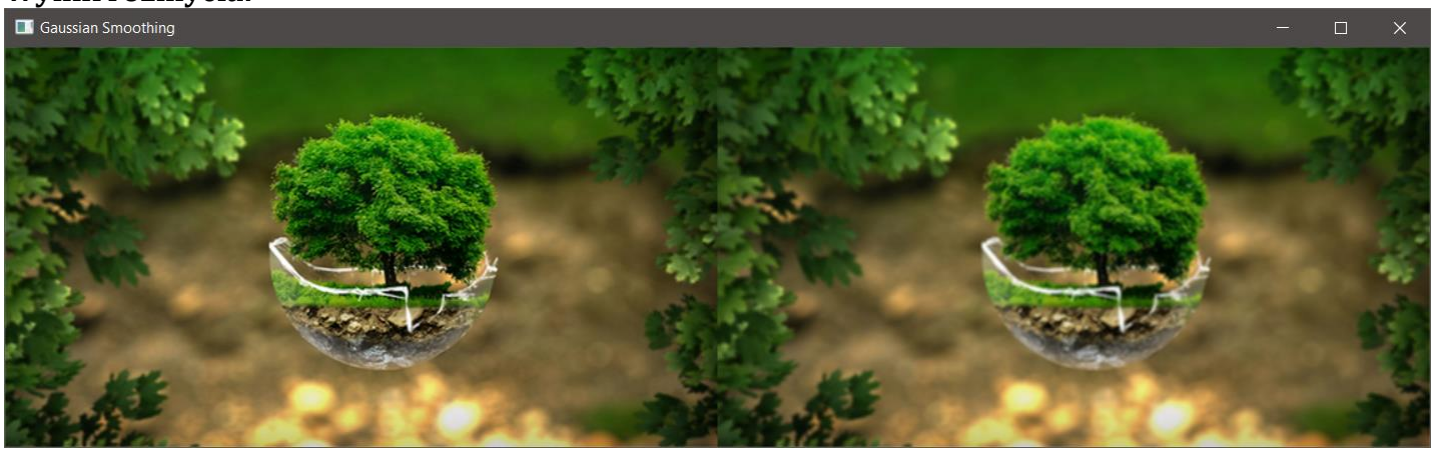
Przeprowadzić rozmycie Gaussa oraz wyostrzenie na dowolnym obrazie za pomocą operacji splotu na oknach 3 na 3.

• W jaki sposób można rozwiązać problem wartości brzegowych?
• Jakie obrazy należy wstępnie przetwarzać przed operacjami splotu?
• Przeprowadzić wykrywanie krawędzi za pomocą splotu z dowolną maską wykrywającą krawędzie. (na przykład Sobel, Prewitt, Laplaciany)
• Jak wyostrzanie, rozmywanie i wykrywanie krawędzi wpływają na składowe FFT obrazu?

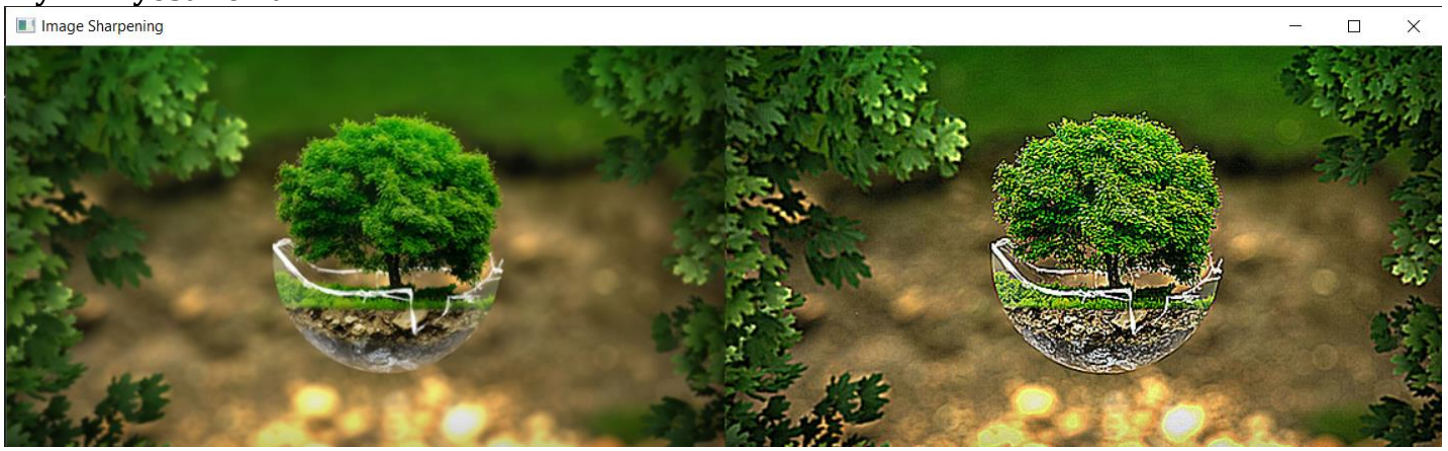
Realizacja w kodzie:

```python
import cv2
import numpy as np

src = cv2.imread('obrazek.png', cv2.IMREAD_UNCHANGED)
dst = cv2.GaussianBlur(src, (3, 3), cv2.BORDER_DEFAULT)

kernel = np.array([[-1,-1,-1],
                   [-1, 9,-1],
                   [-1,-1,-1]])
sharpened = cv2.filter2D(src, -1, kernel)
cv2.imshow('Image Sharpening',np.hstack((src, sharpened)))

cv2.imshow("Gaussian Smoothing", np.hstack((src, dst,)))
cv2.waitKey(0)
cv2.destroyAllWindows()
```

Wynik rozmycia:



Wynik wyostrzenia:



Zadanie 6

Treść zadania:

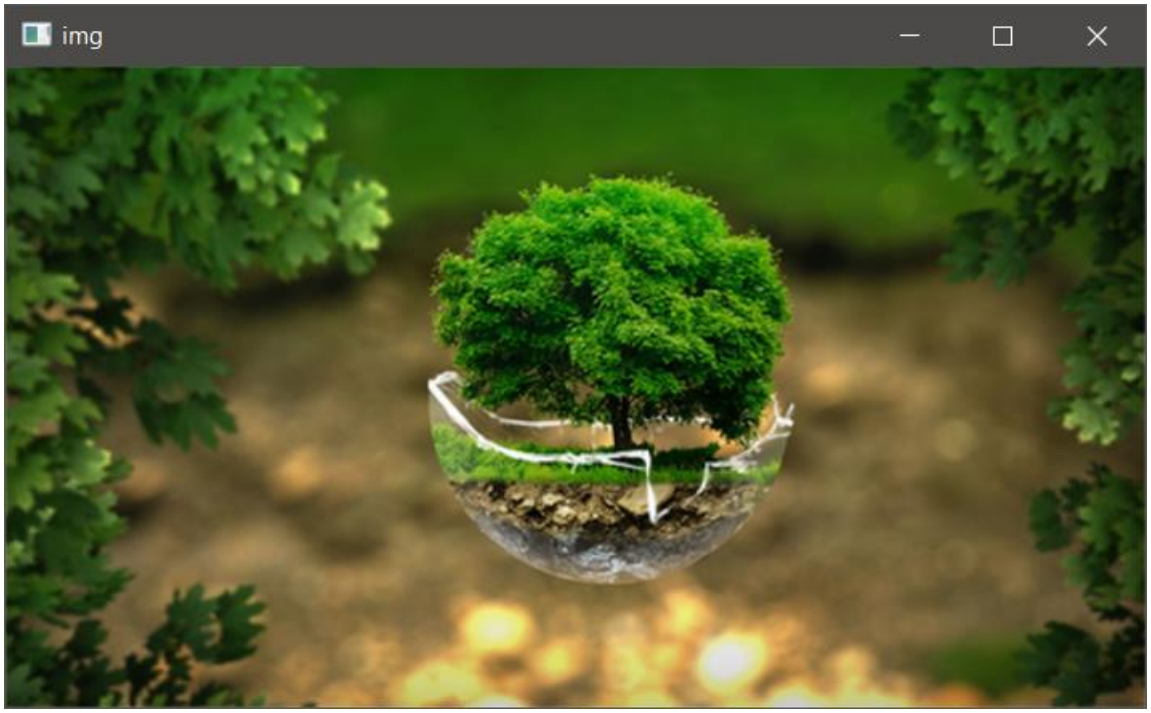
Zaimplementować rozmycie Gaussa na okno o dowolnym rozmiarze. Przetestować okna o różnych rozmiarach.
• Jakie obserwujemy różnice w sile rozmycia na różnych oknach?
• Przetestować okna na białym obrazie z czarną prostą linią o grubości jednego piksela przechodzącą przez górny i dolny środek obrazu.
• Przetestować wykrywanie krawędzi przed i po rozmyciach. Co można zaobserwować w ilości wykrytych krawędzi?

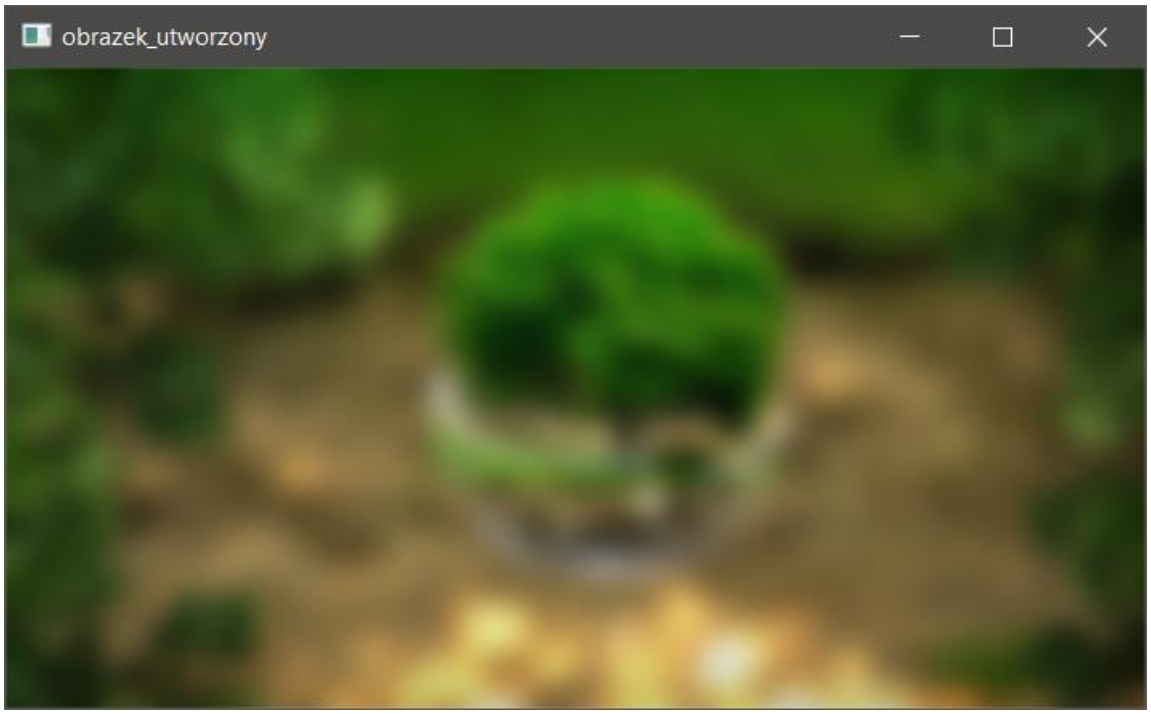
Realizacja w kodzie:

```python
import numpy as np
import imageio as imo
import math as m
from PIL import Image as im

def read_image(file):
    img = imo.imread(file)
    size = np.shape(img)

    return img, size

def make_gaussian_kernel(sigma, size):
    denum = 1 / (2 * np.pi * np.power(sigma, 2))
    dexp = 2 * m.pow(sigma, 2)
    kernel = [[0 for x in range(size)] for y in range(size)]

    for x in range(0, size):
        for y in range(0, size):
            xyval = (m.pow(x - 2, 2) + m.pow(y - 2, 2))
            befexp = (xyval / dexp) * (-1)
            exp = m.exp(befexp)
            val = denum * exp
            kernel[x][y] = val

    raiseval = kernel[2][2] / kernel[0][0]
    sumkernel = 0

    for x in range(0, size):
        for y in range(0, size):
            kernel[x][y] = np.around(kernel[x][y] * raiseval)
            sumkernel += round(kernel[x][y], 0)

    return kernel, sumkernel + 1
def smoothing(img, kernel, kernsum, time):
    row = len(img)
    col = len(img[0])
    blur = img

    for tm in range(time):
        for x in range(0, row):
            for y in range(0, col):
                if x == 0 or y == 0 or x == row - 1 or y == col - 1:
                    blur[x][y] = img[x][y]
                elif x == 1 or y == 1 or x == row - 2 or y == col - 2:
                    blur[x][y] = img[x][y]
                else:
```

```
48
49                      blur[x][y] = (
50                          img[x - 2][y - 2] * kernel[0][0] + img[x - 2][y - 1] * kernel[0][1] +
51                          img[x - 2][y] * kernel[0][2]
52                          + img[x - 2][y + 1] * kernel[0][3] + img[x - 2][y + 2] * kernel[0][4]
53                          + img[x - 1][y - 2] * kernel[1][0] + img[x - 1][y - 1] * kernel[1][1] +
54                          img[x - 1][y] * kernel[1][2]
55                          + img[x - 1][y + 1] * kernel[1][3] + img[x - 1][y + 2] * kernel[1][4]
56                          + img[x][y - 2] * kernel[2][0] + img[x][y - 1] * kernel[2][1] + img[x][y] *
57                          kernel[2][2]
58                          + img[x][y + 1] * kernel[2][3] + img[x][y + 2] * kernel[2][4]
59                          + img[x + 1][y] * kernel[3][0] + img[x + 1][y - 1] * kernel[3][1] + img[x + 1][
60                          y] * kernel[3][2]
61                          + img[x + 1][y + 1] * kernel[3][3] + img[x + 1][y + 2] * kernel[3][4]
62                          + img[x + 2][y - 2] * kernel[4][0] + img[x + 2][y - 1] * kernel[4][1] +
63                          img[x + 2][y] * kernel[4][2]
64                          + img[x + 2][y + 1] * kernel[4][3] + img[x + 2][y + 2] * kernel[4][4]) / kernsum
65      return blur
66
67  if __name__ == "__main__":
68
69      filename = 'obrazek.png'
70      imgori, ori_size = read_image(filename)
71      gauker, gausum = make_gaussian_kernel(1, 5)
72      times = 2
73      imgblur = smoothing(imgori, gauker, gausum, times)
74
75      save = im.fromarray(imgblur, mode=None)
76      save.save("obrazek_utworzony.png", "PNG")
```

Wynik





Interpretacja wyników i wnioski

Pierwszym wnioskiem, który nasuwa się po wykonaniu zadań jest to, że zadania wymagały dużo czasu. Nie mniej jednak za pomocą Pythona udało się rozwiązać wszystkie zadania.
Po wykonaniu pierwszego zadania stwierdzono, że:

a) Dowód analityczny liniowości wykazał, że addytywność i skalowalność są spełnione. Widać, że wykres dla wzoru: S{x1[n] + x2[n]} jest taki sam jak wykres dla wzoru S{x1[n]} + S{x2[n]}, więc addytywność jest spełniona. Zatem jest to system liniowy. System jest przyczynowy, ponieważ jest addytywny i skalowalny, czyli jest liniowy na całej dziedzinie rzeczywistej. Jest to system, w którym wynik zależy od wejścia bieżącego, a nie zależy od wejść przyszłych.

b) Dowód analityczny liniowości wykazał, że addytywność i skalowalność nie są spełnione. Widać, że wykres dla wzoru: S{x1[n] + x2[n]} jest inny niż wykres dla wzoru S{x1[n]} + S{x2[n]}, więc addytywność nie jest spełniona, zatem jest to system nieliniowy. Jest to system inkrementalnie liniowy, tzn. liniowy względem różnic sygnałów wejścia i wyjścia. System jest przyczynowy, ponieważ działa na całej dziedzinie liczb rzeczywistych. Jest to system, w którym wynik zależy od wejścia bieżącego, a nie zależy od wejść przyszłych.

c) Dowód analityczny liniowości wykazał, że addytywność jest zachowana. Wykresy pokazują, że wykres dla wzoru: S{x1[n] + x2[n]} jest taki sam jak wykres dla wzoru S{x1[n]} + S{x2[n]}, więc addytywność jest spełniona, dlatego jest to system liniowy. System nie jest przyczynowy, ponieważ, nie jest to system, w którym wyjścia zależą od wejść bieżących i przeszłych, a zależą od wejść przyszłych.

W kolejnym zadaniu po przeanalizowaniu wyników wywnioskowano, że: Na wykresie 4 Operacje splotu dla zmiennej k = 0 oraz dla ustalonego sygnału h[n] są przemienne. Na wykresie 5 dla zmiennej k = 0 oraz dla ustalonego sygnału h[n] operacje splotu są liniowe. Na wykresie 6: Operacje splotu dla zmiennej k = 16 oraz dla ustalonego sygnału h[n] są przemienne. Na wykresie 7: Dla zmiennej k = 16 oraz dla ustalonego sygnału h[n] operacje splotu są liniowe. Na wykresie 8: Operacje splotu dla zmiennej k = 32 oraz dla ustalonego sygnału h[n] są przemienne. Na wykresie 9: Dla zmiennej k = 32 oraz dla ustalonego sygnału h[n] operacje splotu są liniowe.

W zdaniu trzecim zauważono, że splot liniowy x1*h kształtem jest częściowo podobny do IDFT iloczynu G(k). Jednak obydwa wykresy różnią się znacząco. Parabola w części rzeczywistej IDFT iloczynu G(k) jest

częściowo przesunięta, poza tym, splot liniowy jest przedstawiony na dwukrotnie większej liczbie próbówek. Obydwa wykresy mają te same amplitudy.

W zadaniu czwartym porówna dźwięki z komory bezechowej z akustyczną odpowiedzią impulsową dwóch pomieszczeń oraz ze splotem. Dźwięk w komorze bezechowej nie posiada echa. Dźwięki nagrane w pomieszczeniach takich jak łazienka prezentują dźwięk echa. Akustyczna odpowiedź impulsowa salonu ma mniejsze echo niż ta w łazience, ponieważ oba obiekty różnią się przestrzenią. Po dokonaniu splotu dźwięku z komory bezechowej z akustyczną odpowiedzą impulsową salonu jak i łazienki, w pierwotnie „pustym" brzmieniu dźwięku pojawia się echo, które nadaje „rozgłosu" dla „pustego" dźwięku muzyki techno.

W zadaniu piątym wykorzystano gotową funkcję Gaussian blur. Pochodzi ona z pakietu CV. Znając ją, rozmycie obrazu okazuje się być bardzo trywialnym problemem. Argumentami funkcji są: wejściowy obraz, rozmiar, typ granic obrazu). Wykonano również wyostrzenie. Dowolny obraz to zbiór sygnałów o różnych częstotliwościach. Wyższe częstotliwości kontrolują krawędzie, a niższe częstotliwości kontrolują zawartość obrazu. Krawędzie są tworzone, gdy występuje ostre przejście od wartości jednego piksela do wartości drugiego piksela, np. 0 i 255 w sąsiedniej komórce. Oczywiście następuje gwałtowna zmiana, a tym samym krawędź i wysoka częstotliwość. W celu wyostrzenia obrazu przejścia te można dodatkowo wzmocnić. Czego też dokonaliśmy.

Jednym ze sposobów jest połączenie z obrazem samodzielnie utworzonego jądra filtru. W kolejnym zadaniu należało zaimplementować własną funkcję blurującą. Jest to złożony proces. Wzór jak widać jest bardzo skomplikowany. Koniec końców efekt jest ten sam. Zdjęcie zostało poprawnie rozmyte.

Źródła

[1] https://www.tutorialkart.com/opencv/python/opencv-python-gaussian-image-smoothing/
[2] https://multimed.org/student/pdio/pdio09_filtracja_obrazu_poprawa_jakosci_tekstu.pdf
[3] https://qa-stack.pl/programming/4993082/how-to-sharpen-an-image-in-opencv
[4] https://10-raisons.fr/pl/quest-ce-quun-flou-gaussien-exactement/