



**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.

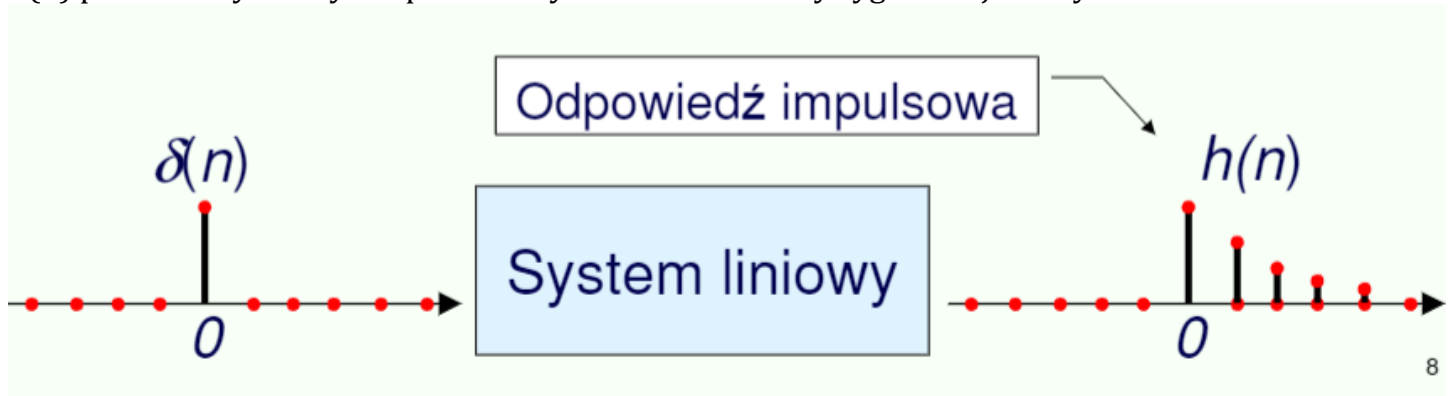
.....
Data i podpis prowadzącego

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 $\delta(n)$ 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:

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

I_o – obraz oryginalny

I_m – obraz uśredniony, $k \times k$ – rozmiar maski filtru, np. 3×3

I_p – 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:

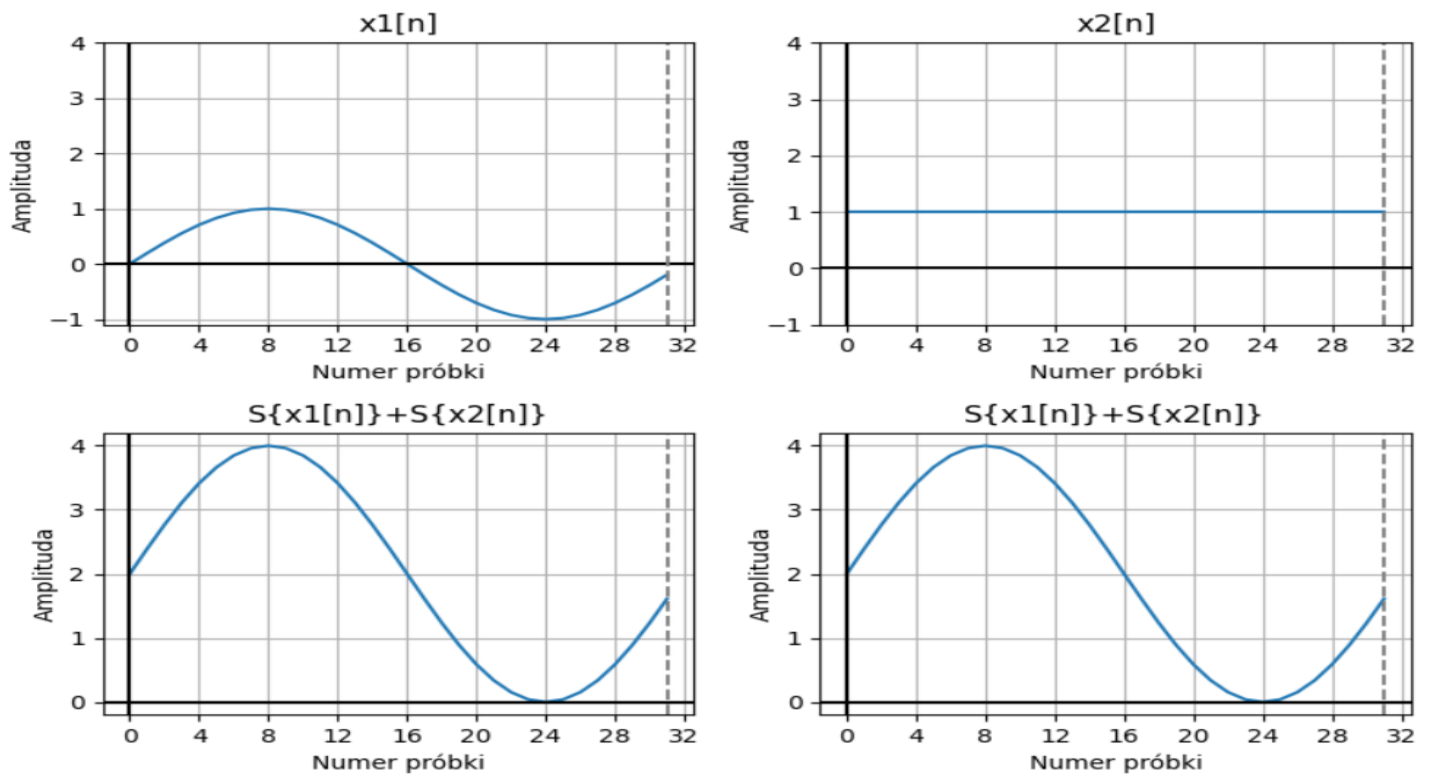
```
1 import numpy as np
2 import matplotlib.pyplot as plt
3
4 def show(x1):
5     plt.figure(figsize=(8,6),tight_layout=1)
6
7     plt.subplot(2,2,1)
8     plt.plot(x,y1)
9     plt.title('x1[n]')
10    plt.xlabel('Numer próbki')
11    plt.ylabel('Amplituda')
12    plt.axhline(y=0,color = "k")
13    plt.axvline(x=0,color = "k")
14    plt.axvline(x=0.97,color = "grey",linestyle='--')
15    plt.grid(True,which='both')
16    plt.yticks(np.arange(-1,5,1))
17    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])
18
19    plt.subplot(2,2,2)
20    plt.plot(x,y2)
21    plt.title('x2[n]')
22    plt.xlabel('Numer próbki')
23    plt.ylabel('Amplituda')
24    plt.axhline(y=0,color = "k")
25    plt.axvline(x=0,color = "k")
26    plt.axvline(x=0.97,color = "grey",linestyle='--')
27    plt.grid(True,which='both')
28    plt.yticks(np.arange(-1,5,1))
29    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])
31    plt.subplot(2,2,3)
32    plt.plot(x1,S1+S2)
33    plt.title('S{x1[n]}+S{x2[n]}')
34    plt.xlabel('Numer próbki')
35    plt.ylabel('Amplituda')
36    plt.axhline(y=0,color = "k")
37    plt.axvline(x=0,color = "k")
38    plt.axvline(x=0.97,color = "grey",linestyle='--')
39    plt.grid(True,which='both')
40    if i!=2: plt.yticks(np.arange(0,5,1))
41    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])
```

```

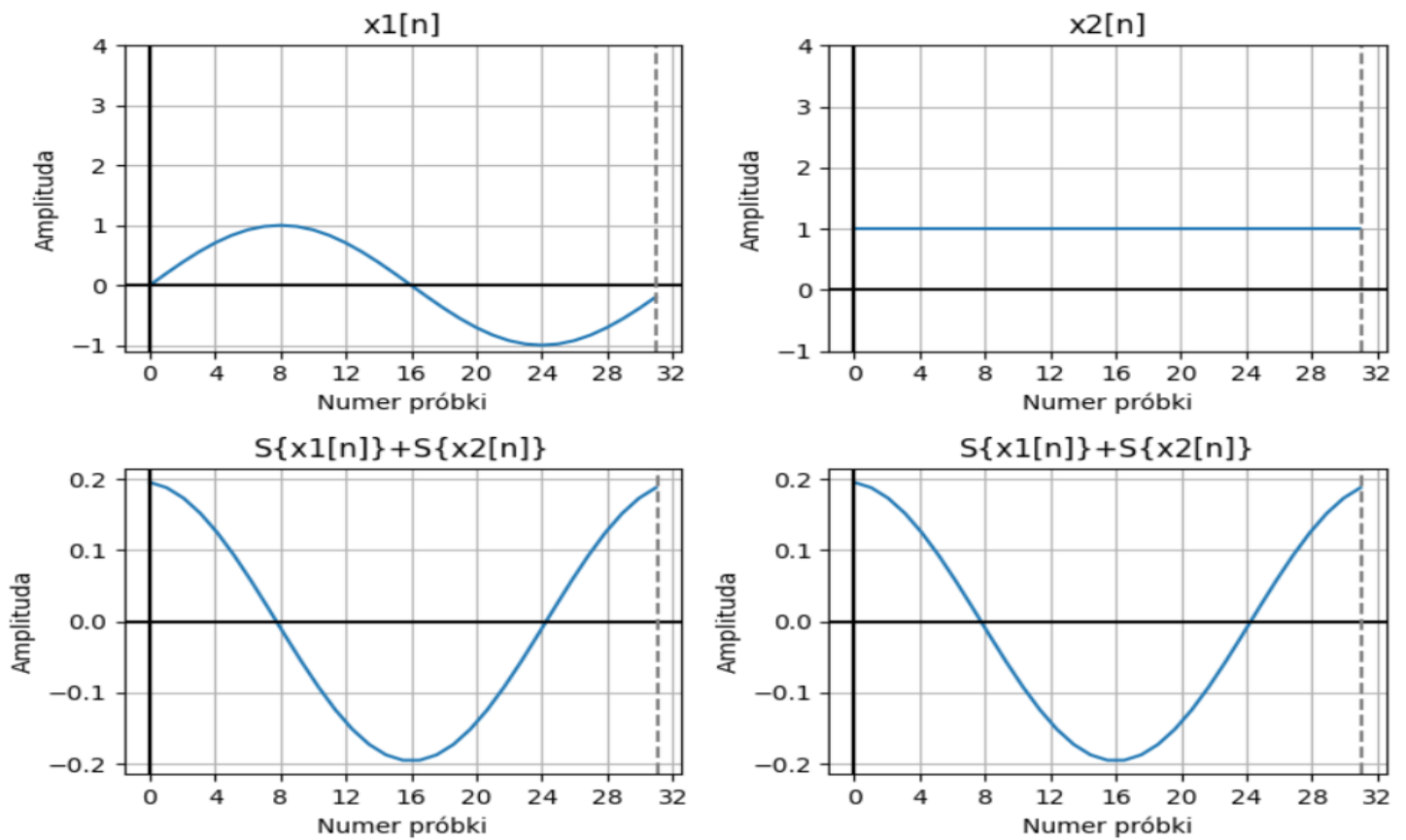
43     plt.subplot(2,2,4)
44     if i==0: plt.plot(x1,fun_a(y1+y2))
45     if i==1: plt.plot(x1,fun_b(y1+y2))
46     if i==2: plt.plot(x1,fun_c(y1+y2))
47     plt.title('S{x1[n]}+S{x2[n]}')
48     plt.xlabel('Numer próbki')
49     plt.ylabel('Amplituda')
50     plt.axhline(y=0,color = "k")
51     plt.axvline(x=0,color = "k")
52     plt.axvline(x=0.97,color = "grey",linestyle='--')
53     plt.grid(True,which='both')
54     if i!=2: plt.yticks(np.arange(0,5,1))
55     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])
56
57     plt.show()
58
59 def fun_a(f):
60     f = [f[i]*2
61           for i in range (len(f))]
62     return np.array(f)
63
64 def fun_b(f):
65     f = [f[i]+1
66           for i in range (len(f))]
67     return np.array(f)
68
69 def fun_c(f):
70     f = [f[i+1]-f[i]
71           for i in range (len(f)-1)]
72     return np.array(f)
73
74 N = 32
75 x = np.arange(N)/N
76 y1 = np.sin(2*np.pi*x)
77 y2 = np.ones(32)
78
79 i=0
80 #Podpunkt A
81 S1 = fun_a(y1)
82 S2 = fun_a(y2)
83 show(x)
84
85 i+=1
86 #Podpunkt B
87 S1 = fun_b(y1)
88 S2 = fun_b(y2)
89 show(x)
90
91 i+=1
92 #Podpunkt C
93 x1 = np.arange(31)/31
94 S1 = fun_c(y1)
95 S2 = fun_c(y2)
96 show(x1)

```

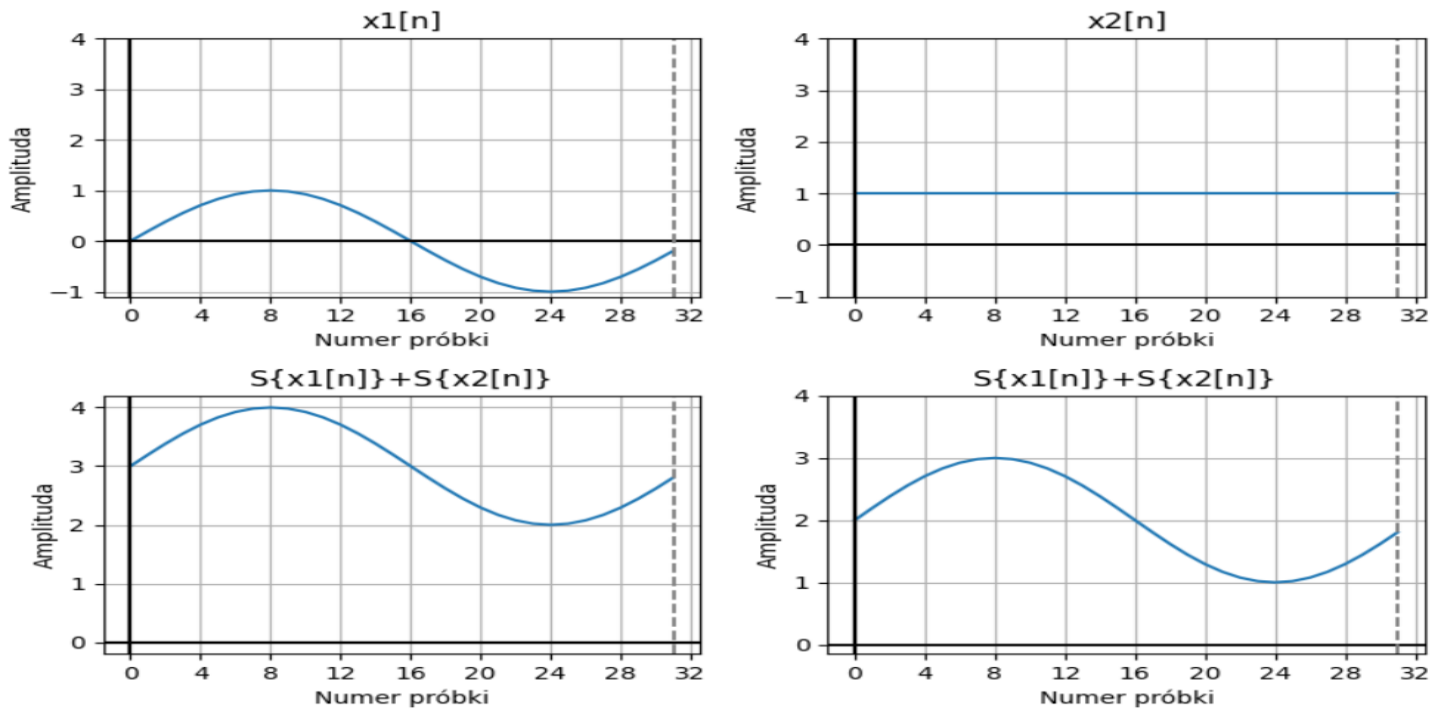
Wyniki:



Wykres 2 Wyniki 1.a



Wykres 1 Wyniki 1.b



Wykres 3 Wyniki 1.c

Zadanie 2

Treść zadania:

Wygenerować sygnały $x_1[n] = \sin(2\pi n/N)$, $x_2[n] = \sin(4\pi n/N)$ oraz $h[n] = \delta[n - k]$, gdzie $k = \{0, 16, 32\}$, $N = 64$, (założyć, że $0 \leq n < N$). Wyznaczyć spłot liniowy sygnałów $x_1[n]$, $x_2[n]$ z sygnałem $h[n]$ oraz samych ze sobą. Sporządzić wykresy, sprawdzić czy operacje spłotu są przemienne oraz liniowe (dla ustalonego sygnału $h[n]$)

Realizacja w kodzie

```

4 def show():
5     plt.figure(figsize=(8,6),tight_layout=1)
6
7     plt.subplot(3,3,1)
8     plt.plot(x,y1)
9     plt.title('y1')
10    plt.xlabel('Numer próbki')
11    plt.ylabel('Amplituda')
12    plt.axhline(y=0,color = "grey",linestyle='--')
13    plt.axvline(x=0,color = "grey",linestyle='--')
14    plt.axvline(x=63,color = "grey",linestyle='--')
15    plt.grid(True,which='both')
16    plt.yticks(np.arange(-1,1.1,0.5))
17    plt.xlim(0,80)
18
19    plt.subplot(3,3,2)
20    plt.plot(np.convolve(y1,h))
21    plt.title('Splot liniowy y1*h')
22    plt.yticks(np.arange(-1,1.1,0.5))
23    plt.xlabel('Numer próbki')
24    plt.ylabel('Amplituda')
25    plt.axhline(y=0,color = "grey",linestyle='--')
26    plt.axvline(x=0,color = "grey",linestyle='--')
27    plt.axvline(x=126,color = "grey",linestyle='--')
28    plt.grid(True,which='both')
29    plt.xlim(0,150)
30
31    plt.subplot(3,3,3)
32    plt.plot(np.convolve(h,y1))
33    plt.title('Splot liniowy h*y1')
34    plt.xlabel('Numer próbki')
35    plt.ylabel('Amplituda')
36    plt.axhline(y=0,color = "grey",linestyle='--')
37    plt.axvline(x=0,color = "grey",linestyle='--')
38    plt.axvline(x=126,color = "grey",linestyle='--')
39    plt.grid(True,which='both')
40    plt.yticks(np.arange(-1,1.1,0.5))
41    plt.xlim(0,150)
42
43    plt.subplot(3,3,4)
44    plt.plot(x,y2)
45    plt.title('y2')
46    plt.xlabel('Numer próbki')
47    plt.ylabel('Amplituda')
48    plt.axhline(y=0,color = "grey",linestyle='--')
49    plt.axvline(x=0,color = "grey",linestyle='--')
50    plt.axvline(x=63,color = "grey",linestyle='--')
51    plt.grid(True,which='both')
52    plt.yticks(np.arange(-1,1.1,0.5))
53    plt.xlim(0,80)
54

```

```

55 plt.subplot(3,3,5)
56 plt.plot(np.convolve(y2,h))
57 plt.title('Splot Liniowy y2*h')
58 plt.xlabel('Numer próbki')
59 plt.ylabel('Amplituda')
60 plt.axhline(y=0,color = "grey",linestyle='--')
61 plt.axvline(x=0,color = "grey",linestyle='--')
62 plt.axvline(x=126,color = "grey",linestyle='--')
63 plt.grid(True,which='both')
64 plt.yticks(np.arange(-1,1.1,0.5))
65 plt.xlim(0,150)
66
67 plt.subplot(3,3,6)
68 plt.plot(np.convolve(h,y2))
69 plt.title('Splot Liniowy h*y2')
70 plt.xlabel('Numer próbki')
71 plt.ylabel('Amplituda')
72 plt.axhline(y=0,color = "grey",linestyle='--')
73 plt.axvline(x=0,color = "grey",linestyle='--')
74 plt.axvline(x=126,color = "grey",linestyle='--')
75 plt.grid(True,which='both')
76 plt.yticks(np.arange(-1,1.1,0.5))
77 plt.xlim(0,150)
78
79
80 plt.subplot(3,3,7)
81 plt.stem(h,use_line_collection='true')
82 plt.title('h')
83 plt.xlabel('Numer próbki')
84 plt.ylabel('Amplituda')
85 plt.axhline(y=0,color = "grey",linestyle='--')
86 plt.axvline(x=0,color = "grey",linestyle='--')
87 plt.axvline(x=63,color = "grey",linestyle='--')
88 plt.grid(True,which='both')
89 plt.yticks(np.arange(-1,1.1,0.5))
90 plt.xlim(-1,80)
91
92 plt.subplot(3,3,8)
93 plt.plot(np.convolve(y1,y2))
94 plt.title('Splot Liniowy y1*y2')
95 plt.xlabel('Numer próbki')
96 plt.ylabel('Amplituda')
97 plt.axhline(y=0,color = "grey",linestyle='--')
98 plt.axvline(x=0,color = "grey",linestyle='--')
99 plt.axvline(x=126,color = "grey",linestyle='--')
100 plt.grid(True,which='both')
101 plt.yticks(np.arange(-10,10.1,5))
102 plt.xlim(0,150)
103
104 plt.subplot(3,3,9)
105 plt.plot(np.convolve(y2,y1))
106 plt.title('Splot Liniowy y2*y1')
107 plt.xlabel('Numer próbki')
108 plt.ylabel('Amplituda')
109 plt.axhline(y=0,color = "grey",linestyle='--')
110 plt.axvline(x=0,color = "grey",linestyle='--')
111 plt.axvline(x=126,color = "grey",linestyle='--')
112 plt.grid(True,which='both')
113 plt.yticks(np.arange(-10,10.1,5))
114 plt.xlim(0,150)
115
116 plt.show()
117
118
119
120
121
122
123
124
125
126
127
128
129
130
131
132
133
134
135
136
137
138
139
140
141
142
143
144
145
146
147
148
149
150
151
152
153
154
155
156
157
158
159
160
161
162
163
164
165
166
167
168
169
170
171
172
173
174
175
176
177
178
179
180
181
182
183
184
185
186
187
188
189
190
191
192
193

```

```

118 def show2():
119     plt.figure(figsize=(8,6),tight_layout=1)
120
121     plt.subplot(3,2,1)
122     plt.plot(x,y1)
123     plt.title('y1')
124     plt.xlabel('Numer próbki')
125     plt.ylabel('Amplituda')
126     plt.axhline(y=0,color = "grey",linestyle='--')
127     plt.axvline(x=0,color = "grey",linestyle='--')
128     plt.axvline(x=63,color = "grey",linestyle='--')
129     plt.grid(True,which='both')
130     plt.yticks(np.arange(-1,1.1,0.5))
131     plt.xlim(0,80)
132
133     plt.subplot(3,2,2)
134     plt.plot(np.convolve(y1+y2,h))
135     plt.title('Splot sumy (y1+y2)*h')
136     plt.xlabel('Numer próbki')
137     plt.ylabel('Amplituda')
138     plt.axhline(y=0,color = "grey",linestyle='--')
139     plt.axvline(x=0,color = "grey",linestyle='--')
140     plt.axvline(x=126,color = "grey",linestyle='--')
141     plt.grid(True,which='both')
142     plt.yticks(np.arange(-2,2.1,1))
143     plt.xlim(0,150)
144
145     plt.subplot(3,2,3)
146     plt.plot(x,y2)
147     plt.title('x2')
148     plt.xlabel('Numer próbki')
149     plt.ylabel('Amplituda')
150     plt.axhline(y=0,color = "grey",linestyle='--')
151     plt.axvline(x=0,color = "grey",linestyle='--')
152     plt.axvline(x=63,color = "grey",linestyle='--')
153     plt.grid(True,which='both')
154     plt.yticks(np.arange(-1,1.1,0.5))
155
156     plt.subplot(3,2,4)
157     plt.plot(np.convolve(y1,h)+np.convolve(y2,h))
158     plt.title('Splot sumy x1*h + x2*h')
159     plt.xlabel('Numer próbki')
160     plt.ylabel('Amplituda')
161     plt.axhline(y=0,color = "grey",linestyle='--')
162     plt.axvline(x=0,color = "grey",linestyle='--')
163     plt.axvline(x=126,color = "grey",linestyle='--')
164     plt.grid(True,which='both')
165     plt.yticks(np.arange(-2,2.1,1))
166     plt.xlim(0,150)
167
168     plt.subplot(3,2,5)
169     plt.stem(h,use_line_collection='true')
170     plt.title('h')
171     plt.xlabel('Numer próbki')
172     plt.ylabel('Amplituda')
173     plt.axhline(y=0,color = "grey",linestyle='--')
174     plt.axvline(x=0,color = "grey",linestyle='--')
175     plt.axvline(x=63,color = "grey",linestyle='--')
176     plt.grid(True,which='both')
177     plt.yticks(np.arange(-1,1.1,0.5))
178     plt.xlim(0,80)
179
180     plt.show()
181
182
183 N = 64
184 k = 0
185 x = np.arange(N)
186 y1 = np.sin(2*np.pi*x/N)
187 y2 = np.sin(4*np.pi*x/N)
188 h = np.zeros(N)/N
189 h[k] = 1
190
191
192
193

```

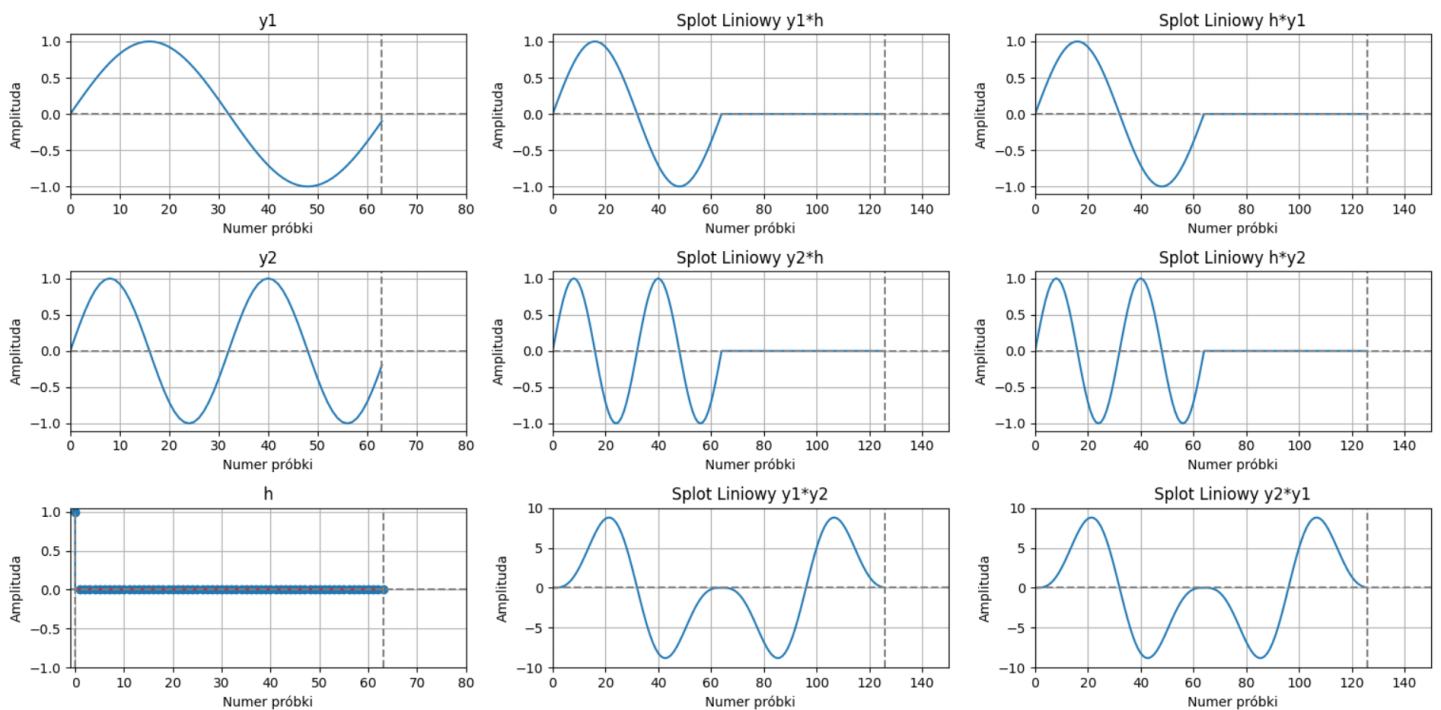


```

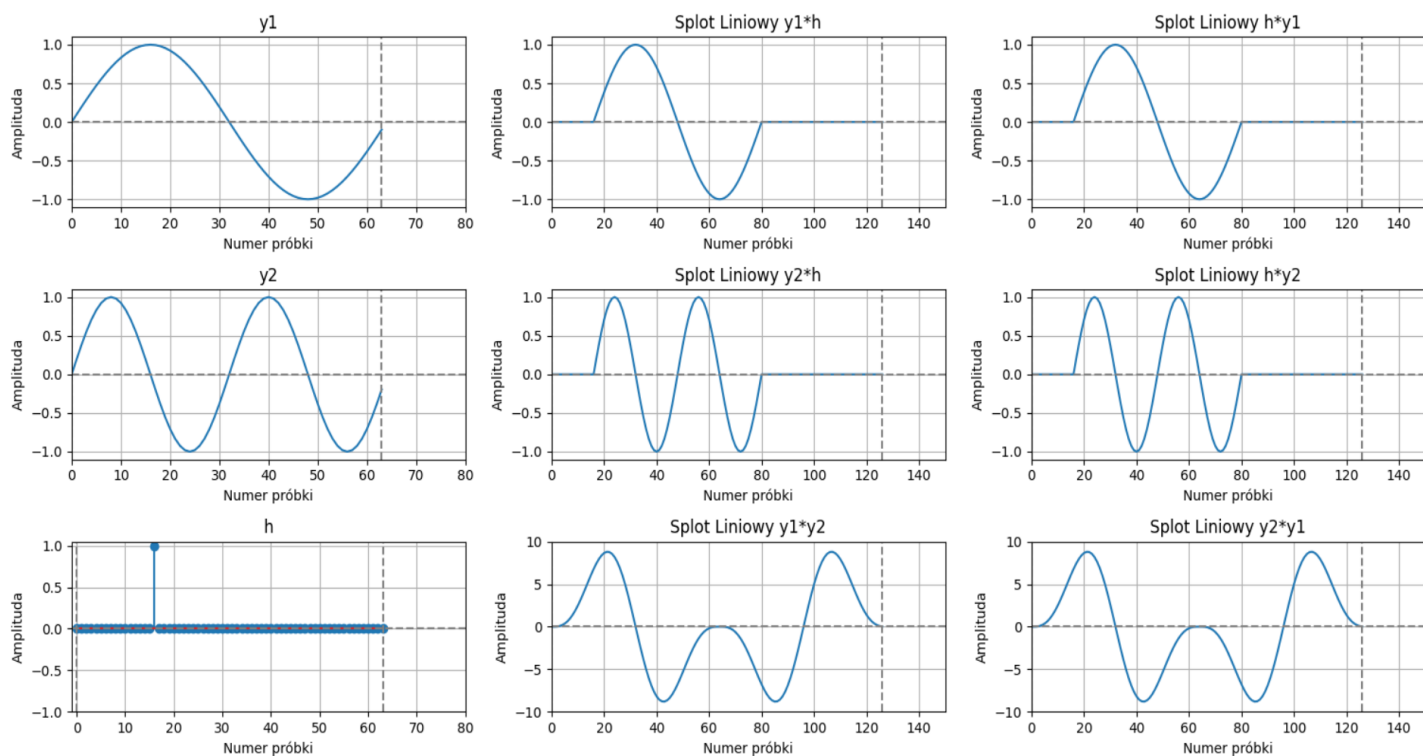
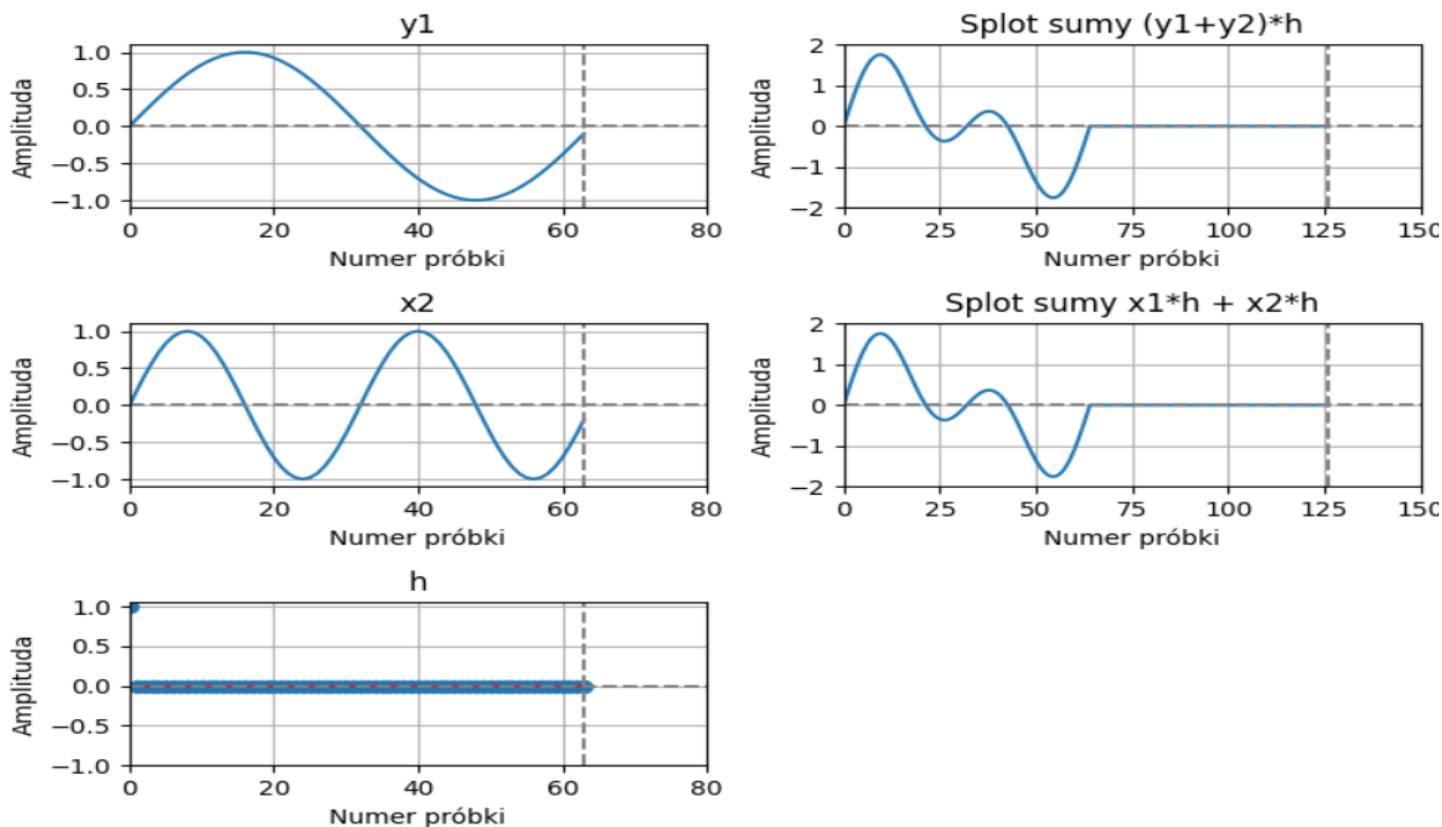
195     show()
196     show2()
197
198     h[k] = 0
199     k=16
200     h[k] = 1
201
202     show()
203     show2()
204
205     h[k] = 0
206     k=32
207     h[k] = 1
208
209     show()
210     show2()

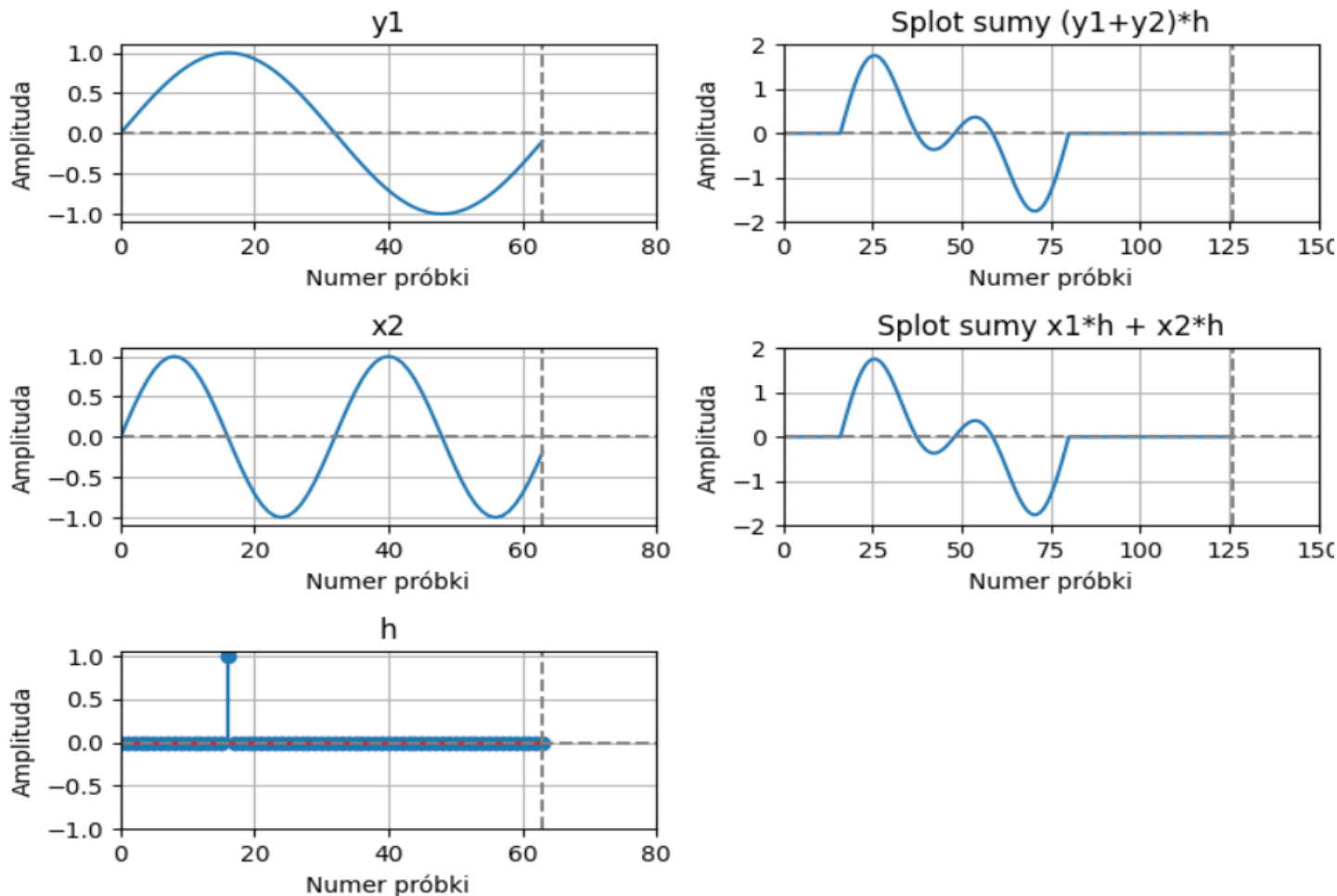
```

Wyniki

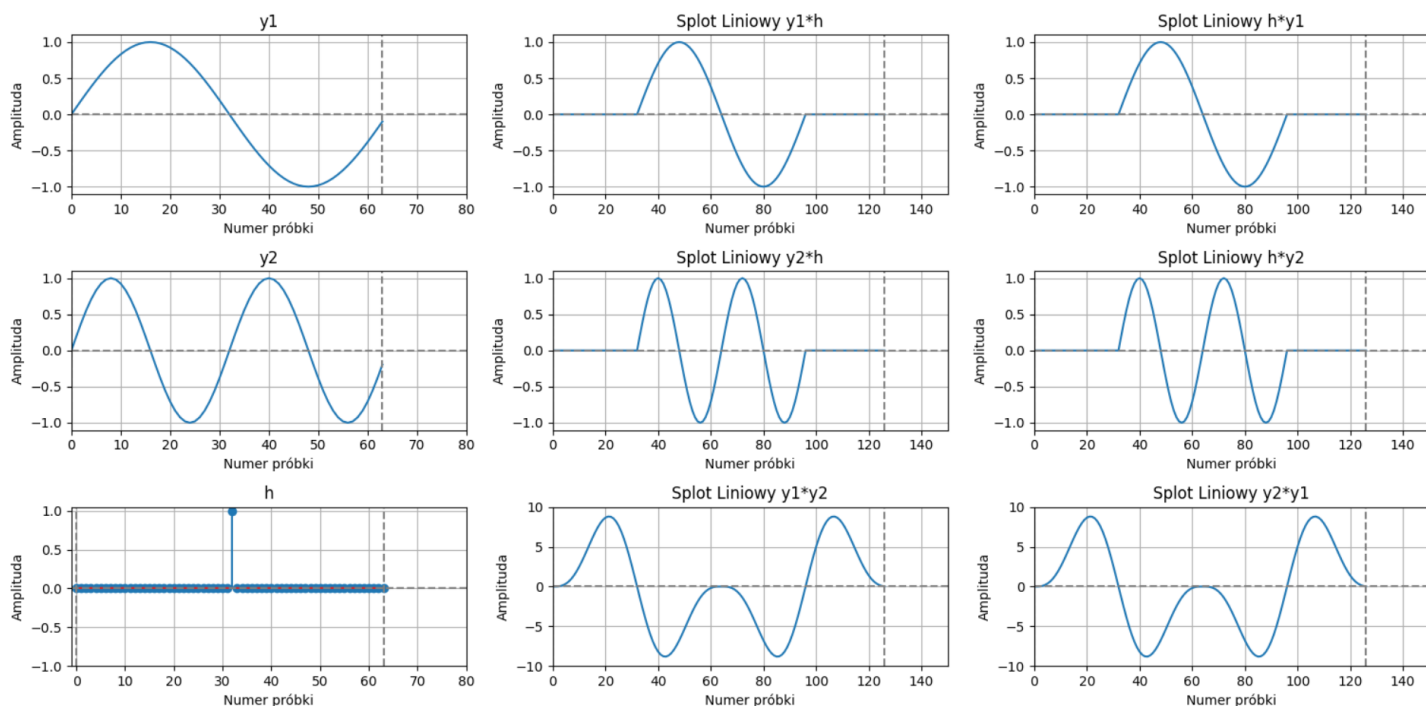


Wykres 4 Operacje splotu dla zmiennej $k = 0$ oraz dla ustalonego sygnału $h[n]$

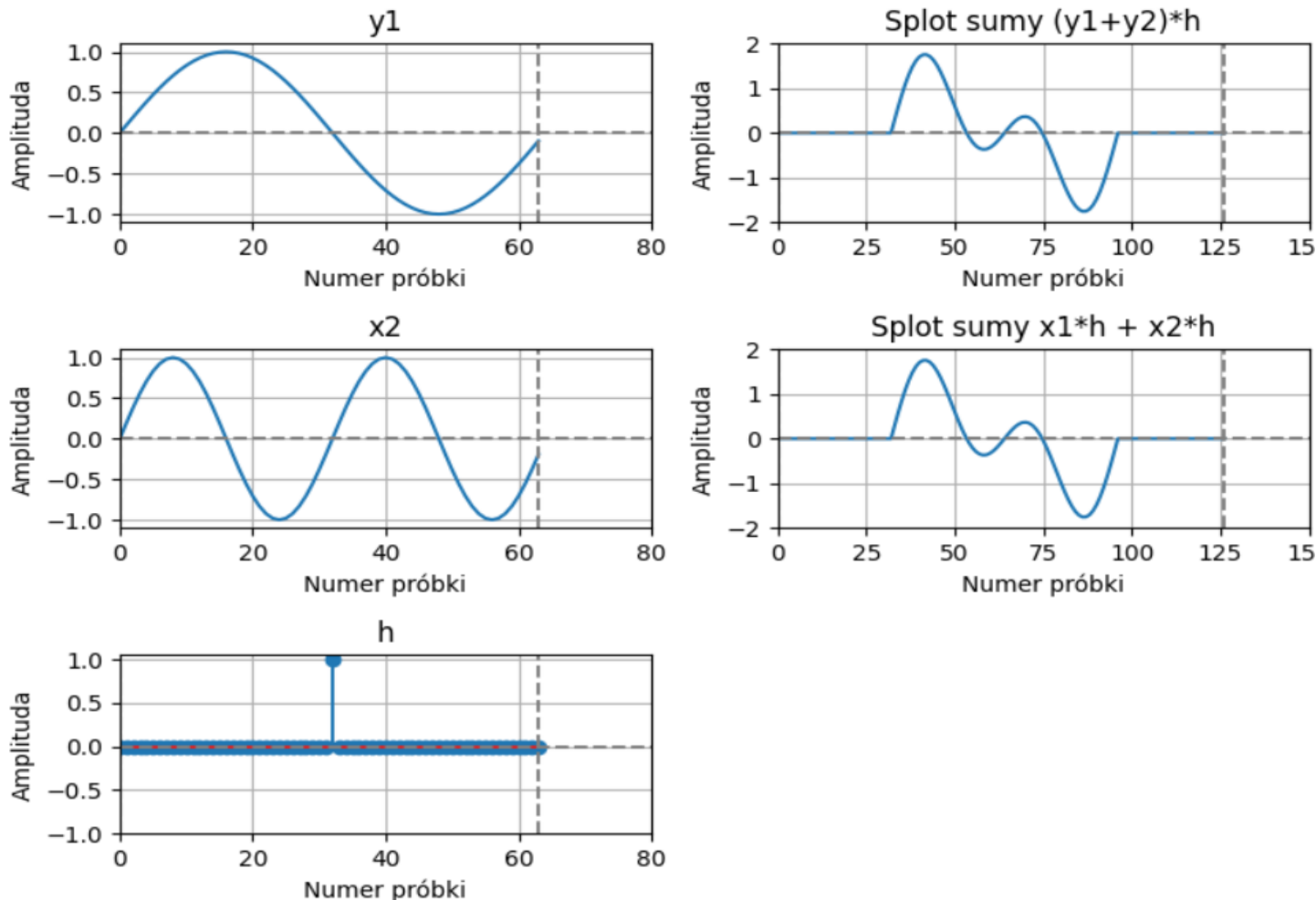




Wykres 7 Dla zmiennej $k = 16$, oraz dla ustalonego sygnału $h[n]$



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. $G(k) = X1(k) \cdot H(k)$ dla $k = 0, 1, \dots, 63$), wyznaczyć IDFT iloczynu $G(k)$. Uzyskany wynik porównać ze splotem liniowym (64-punktowym) sygnałów $x1[n]$ i $h[n]$.

Realizacja w kodzie

```
1 import numpy as np
2 import matplotlib.pyplot as plt
3
4 N = 64
5 x = np.arange(N)
6 y1 = np.sin(2*np.pi*x/N)
7 h = np.exp(-x/10)
8 fy1 = np.fft.fft(y1)
9 fh = np.fft.fft(h)
10 z = fy1 * fh
11 fz = np.fft.fft(z)
```

```

15 plt.subplot(3,2,1)
16 plt.stem(np.real(fy1),use_line_collection='true')
17 plt.title('Re(y1[n])')
18 plt.xlabel('Numer próbek')
19 plt.ylabel('Amplituda')
20 plt.axhline(y=0,color = "k",)
21 plt.axvline(x=0,color = "k")
22 plt.axvline(x=63,color = "grey",linestyle='--')
23 plt.grid(True,which='both')
24 plt.ylim(-1,1)
25
26 plt.subplot(3,2,2)
27 plt.stem(np.real(fh),use_line_collection='true')
28 plt.title('Re(h[n])')
29 plt.xlabel('Numer próbek')
30 plt.ylabel('Amplituda')
31 plt.axhline(y=0,color = "k",)
32 plt.axvline(x=0,color = "k")
33 plt.axvline(x=63,color = "grey",linestyle='--')
34 plt.grid(True,which='both')
35
36 plt.subplot(3,2,3)
37 plt.stem(np.real(z),use_line_collection='true')
38 plt.title('Re(G(k))')
39 plt.xlabel('Numer próbek')
40 plt.ylabel('Amplituda')
41 plt.axhline(y=0,color = "k",)
42 plt.axvline(x=0,color = "k")
43 plt.axvline(x=63,color = "grey",linestyle='--')
44 plt.grid(True,which='both')

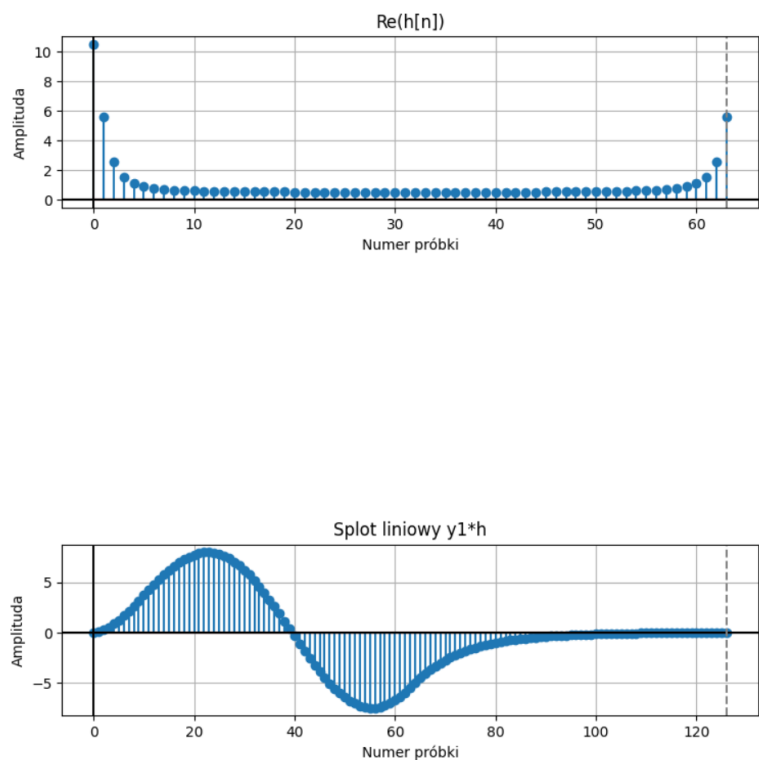
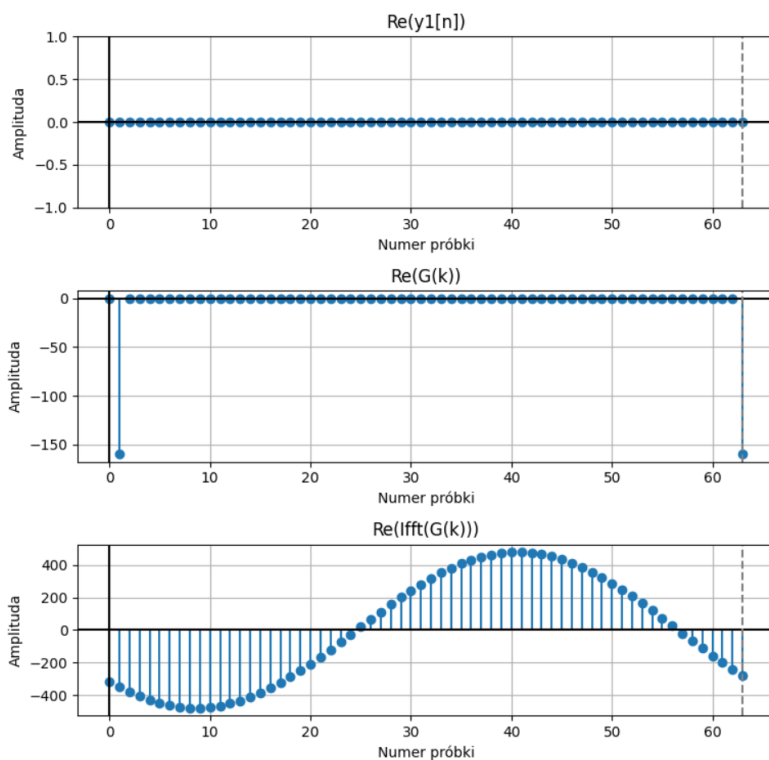
```

```

46 plt.subplot(3,2,5)
47 plt.stem(np.real(fz),use_line_collection='true')
48 plt.title('Re(Ifft(G(k)))')
49 plt.xlabel('Numer próbek')
50 plt.ylabel('Amplituda')
51 plt.axhline(y=0,color = "k",)
52 plt.axvline(x=0,color = "k")
53 plt.axvline(x=63,color = "grey",linestyle='--')
54 plt.grid(True,which='both')
55
56 plt.subplot(3,2,6)
57 plt.stem(np.convolve(y1,h),use_line_collection='true')
58 plt.title('Spłot liniowy y1*h')
59 plt.xlabel('Numer próbek')
60 plt.ylabel('Amplituda')
61 plt.axhline(y=0,color = "k",)
62 plt.axvline(x=0,color = "k")
63 plt.axvline(x=126,color = "grey",linestyle='--')
64 plt.grid(True,which='both')
65
66 plt.show()

```

Wyniki



Zadanie 4

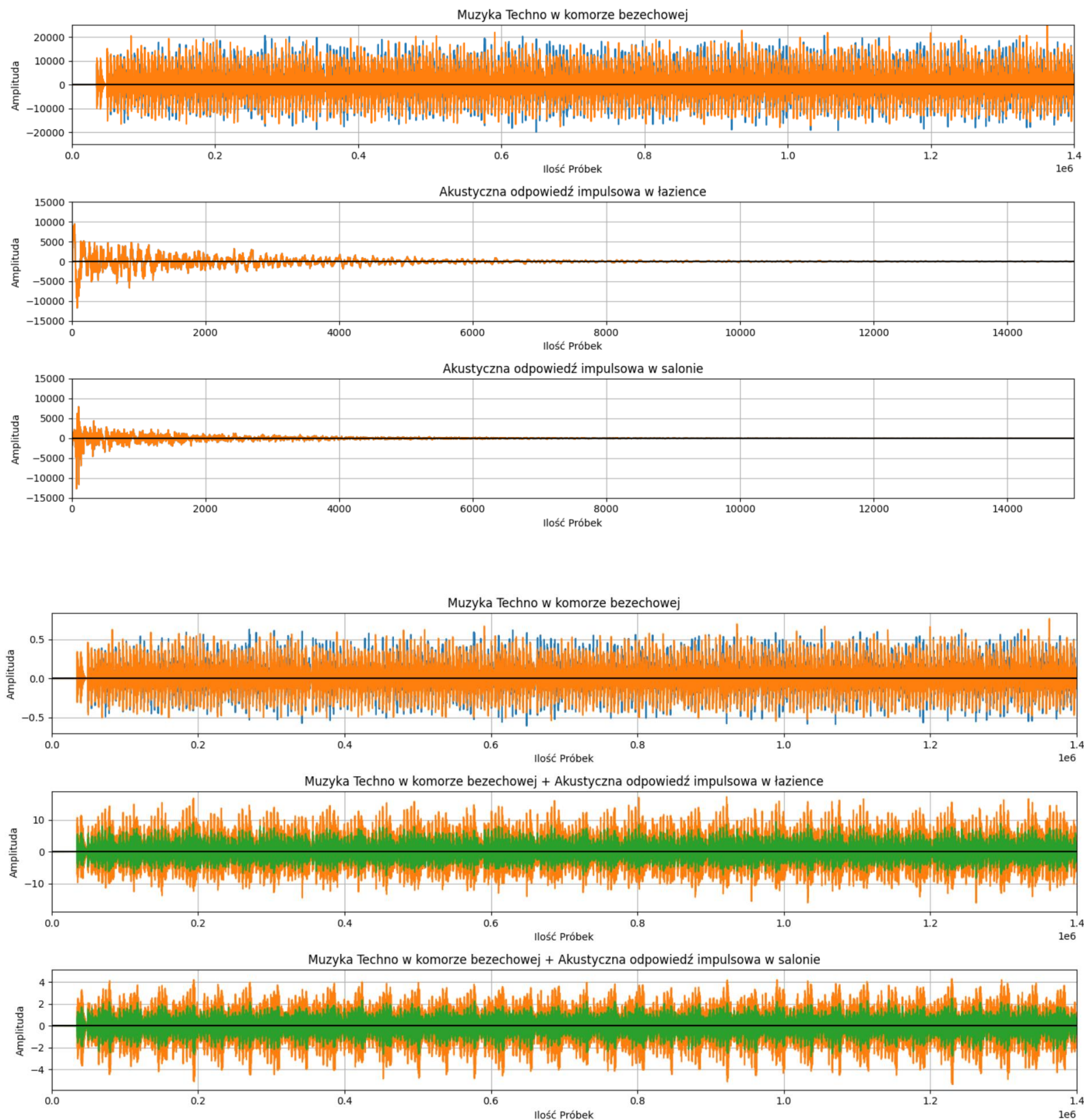
Treść zadania:

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 sploty?

Realizacja w kodzie:

```
1 import numpy as np
2 import matplotlib.pyplot as plt
3 import scipy.io.wavfile as siw
4 import scipy.signal as ss
5 import soundfile as sf
6 import utility as u
7
8 fs, a = siw.read('Universal Linear Accelerator.wav')
9 fs, h1 = siw.read('BathRoom.wav')
10 fs, h2 = siw.read('Living Room.wav')
11
12 plt.figure(figsize=(8,6),tight_layout=1)
13
14 plt.subplot(3,1,1)
15 plt.plot(a)
16 plt.title('Muzyka Techno w komorze bezechowej')
17 plt.ylabel('Amplituda')
18 plt.xlabel('Ilość Próbek')
19 plt.axhline(y=0,color = "k",)
20 plt.axvline(x=0,color = "k")
21 plt.grid(True,which='both')
22 plt.xlim(0,1400000)
23 plt.ylim(-25000,25000)
24
25 plt.subplot(3,1,2)
26 plt.plot(h1)
27 plt.title('Akustyczna odpowiedź impulsowa w łazience')
28 plt.ylabel('Amplituda')
29 plt.xlabel('Ilość Próbek')
30 plt.axhline(y=0,color = "k",)
31 plt.axvline(x=0,color = "k")
32 plt.grid(True,which='both')
33 plt.xlim(0,15000)
34 plt.ylim(-15000,15000)
35
36 plt.subplot(3,1,3)
37 plt.plot(h2)
38 plt.title('Akustyczna odpowiedź impulsowa w salonie')
39 plt.ylabel('Amplituda')
40 plt.xlabel('Ilość Próbek')
41 plt.axhline(y=0,color = "k",)
42 plt.axvline(x=0,color = "k")
43 plt.grid(True,which='both')
44 plt.xlim(0,15000)
45 plt.ylim(-15000,15000)
46
47 plt.show()
48
49 a = u.pcm2float(a,'float32')
50 h1 = u.pcm2float(h1,'float32')
51 h2 = u.pcm2float(h2,'float32')
52
53 y1 = ss.convolve(a,h1)
54 y2 = ss.convolve(a,h2)
55
56 plt.figure(figsize=(8,6),tight_layout=1)
57
58 plt.subplot(3,1,1)
59 plt.plot(a)
60 plt.title('Muzyka Techno w komorze bezechowej')
61 plt.ylabel('Amplituda')
62 plt.xlabel('Ilość Próbek')
63 plt.axhline(y=0,color = "k",)
64 plt.axvline(x=0,color = "k")
65 plt.grid(True,which='both')
66 plt.xlim(0,1400000)
67
68
69
70 plt.subplot(3,1,2)
71 plt.plot(y1)
72 plt.title('Muzyka Techno w komorze bezechowej + Akustyczna odpowiedź impulsowa w łazience')
73 plt.ylabel('Amplituda')
74 plt.xlabel('Ilość Próbek')
75 plt.axhline(y=0,color = "k",)
76 plt.axvline(x=0,color = "k")
77 plt.grid(True,which='both')
78 plt.xlim(0,1400000)
79
80
81
82
83
84
85 plt.subplot(3,1,3)
86 plt.plot(y2)
87 plt.title('Muzyka Techno w komorze bezechowej + Akustyczna odpowiedź impulsowa w salonie')
88 plt.ylabel('Amplituda')
89 plt.xlabel('Ilość Próbek')
90 plt.axhline(y=0,color = "k",)
91 plt.axvline(x=0,color = "k")
92 plt.grid(True,which='both')
93 plt.xlim(0,1400000)
94
95
96 plt.show()
97
98 sf.write("jeden.wav",y1,fs)
99 sf.write("dwa.wav",y2,fs)
```

Wyniki:



Zadanie 5

Treść zadania:

Przeprowadzić rozmycie Gaussa oraz wyostrenie 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 wyostrażanie, rozmywanie i wykrywanie krawędzi wpływają na składowe FFT obrazu?

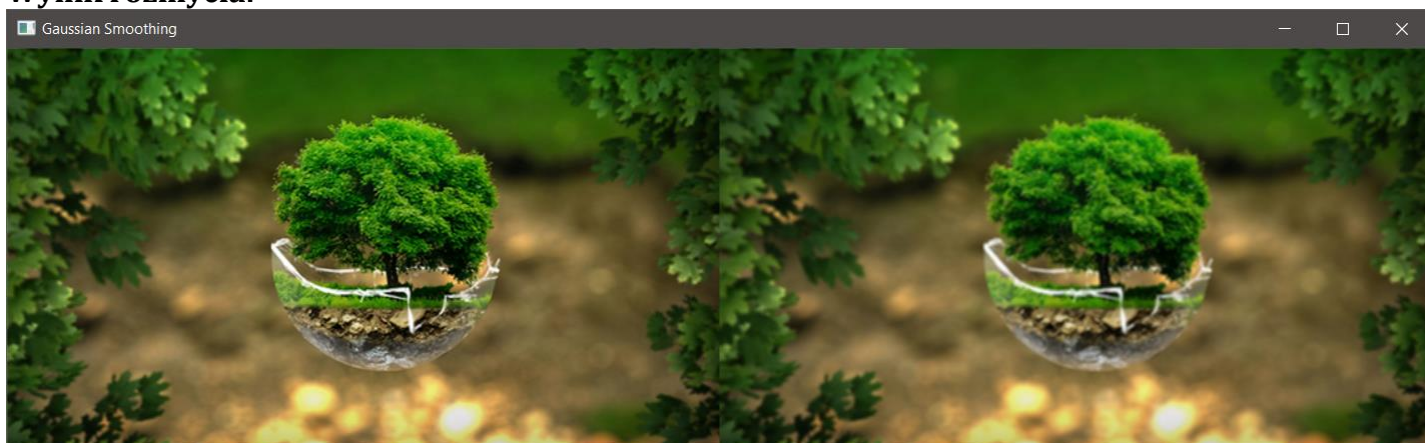
Realizacja w kodzie:

```

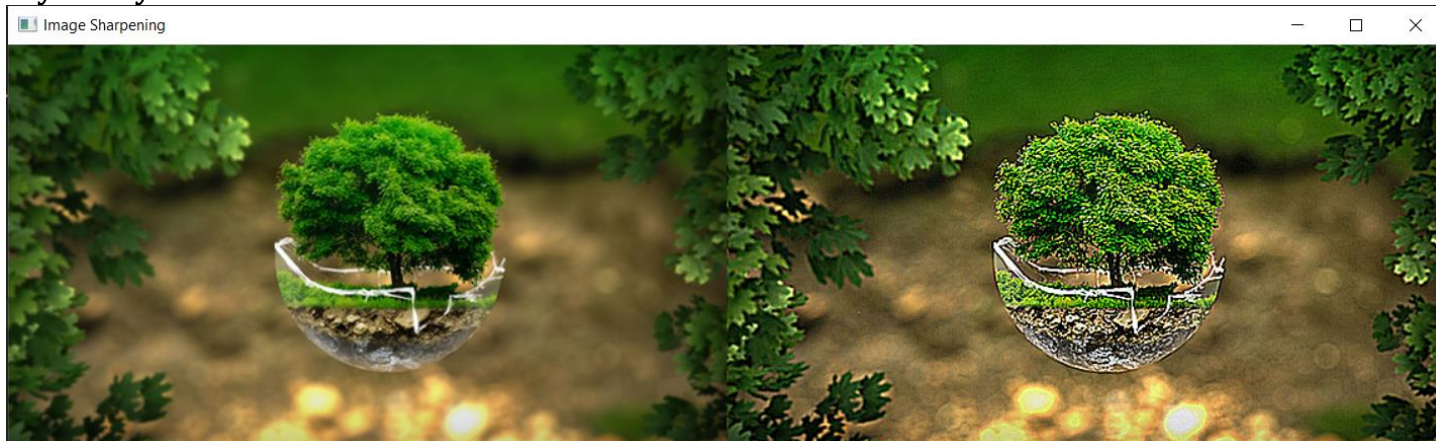
1  import cv2
2  import numpy as np
3
4  src = cv2.imread('obrazek.png', cv2.IMREAD_UNCHANGED)
5  dst = cv2.GaussianBlur(src, (3, 3), cv2.BORDER_DEFAULT)
6
7  kernel = np.array([[ -1, -1, -1],
8                    [ -1,  9, -1],
9                    [ -1, -1, -1]])
10 sharpened = cv2.filter2D(src, -1, kernel)
11 cv2.imshow('Image Sharpening', np.hstack((src, sharpened)))
12
13 cv2.imshow("Gaussian Smoothing", np.hstack((src, dst,)))
14 cv2.waitKey(0)
15 cv2.destroyAllWindows()

```

Wynik rozmycia:



Wynik wyostżenia:



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:

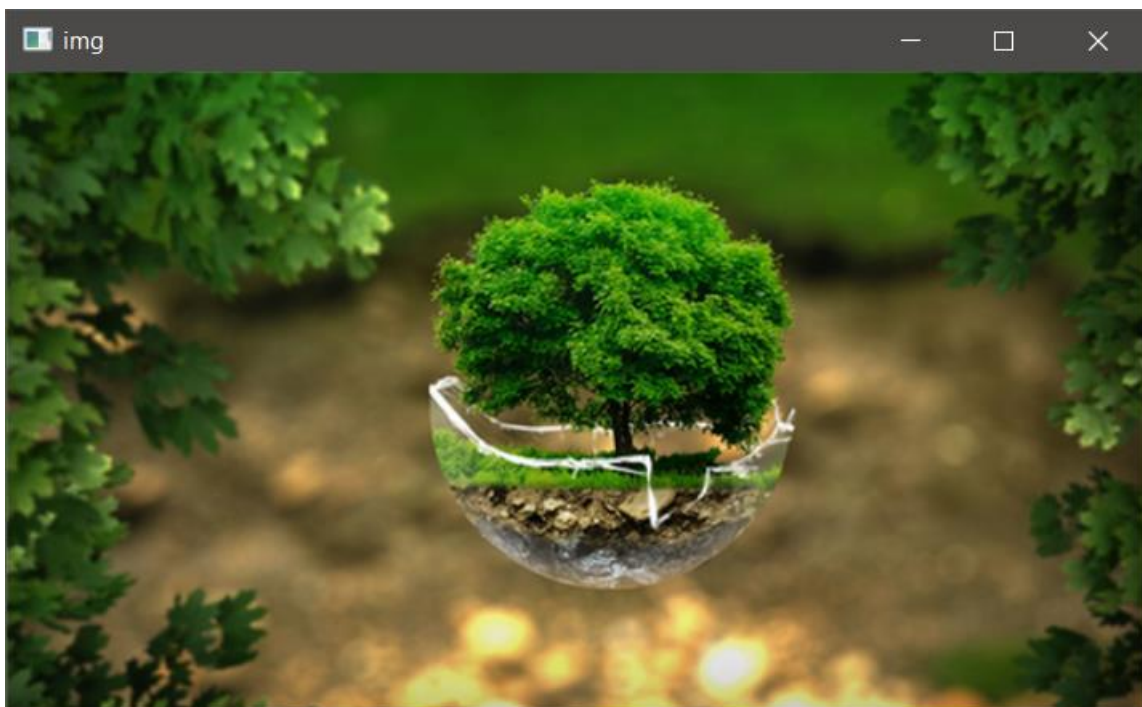
```
1  import numpy as np
2  import imageio as imo
3  import math as m
4  from PIL import Image as im
5
6  def read_image(file):
7      img = imo.imread(file)
8      size = np.shape(img)
9
10     return img, size
11
12 def make_gaussian_kernel(sigma, size):
13     denum = 1 / (2 * np.pi * np.power(sigma, 2))
14     dexp = 2 * m.pow(sigma, 2)
15     kernel = [[0 for x in range(size)] for y in range(size)]
16
17     for x in range(0, size):
18         for y in range(0, size):
19             xyval = (m.pow(x - 2, 2) + m.pow(y - 2, 2))
20             befexp = (xyval / dexp) * (-1)
21             exp = m.exp(befexp)
22             val = denum * exp
23             kernel[x][y] = val
24
25     raiseval = kernel[2][2] / kernel[0][0]
26     sumkernel = 0
27
28     for x in range(0, size):
29         for y in range(0, size):
30             kernel[x][y] = np.around(kernel[x][y] * raiseval)
31             sumkernel += round(kernel[x][y], 0)
32
33     return kernel, sumkernel + 1
34
35 def smoothing(img, kernel, kernsum, time):
36     row = len(img)
37     col = len(img[0])
38     blur = img
39
40     for tm in range(time):
41         for x in range(0, row):
42             for y in range(0, col):
43                 if x == 0 or y == 0 or x == row - 1 or y == col - 1:
44                     blur[x][y] = img[x][y]
45                 elif x == 1 or y == 1 or x == row - 2 or y == col - 2:
46                     blur[x][y] = img[x][y]
47                 else:
```

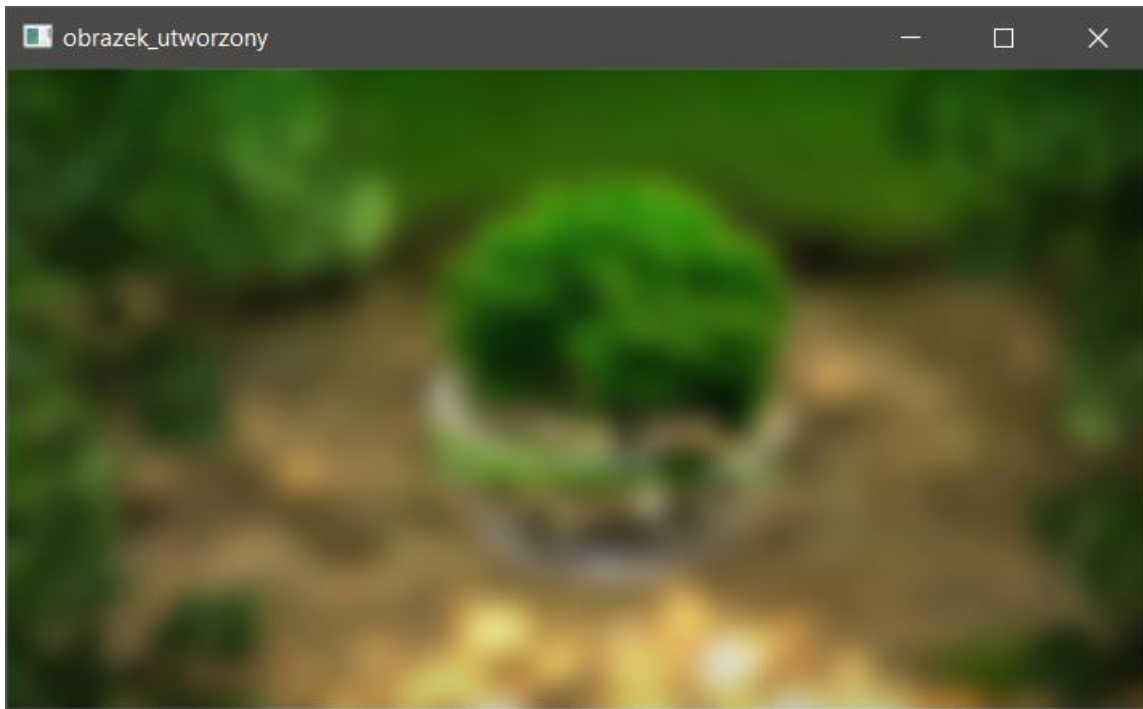
```

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: $S\{x_1[n] + x_2[n]\}$ jest taki sam jak wykres dla wzoru $S\{x_1[n]\} + S\{x_2[n]\}$, 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: $S\{x_1[n] + x_2[n]\}$ jest inny niż wykres dla wzoru $S\{x_1[n]\} + S\{x_2[n]\}$, 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: $S\{x_1[n] + x_2[n]\}$ jest taki sam jak wykres dla wzoru $S\{x_1[n]\} + S\{x_2[n]\}$, 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 $x_1 * h$ 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, spłot liniowy jest przedstawiony na dwukrotnie większej liczbie próbek. Obydwa wykresy mają te same amplitudy.

W zadaniu czwartym porówna dźwięki z komory bezechowej z akustyczną odpowiedzią impulsową dwóch pomieszczeń oraz ze spłotem. 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 spłotu dźwięku z komory bezechowej z akustyczną odpowiedzią 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 wyostwienie. 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 wyostwienia 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/>