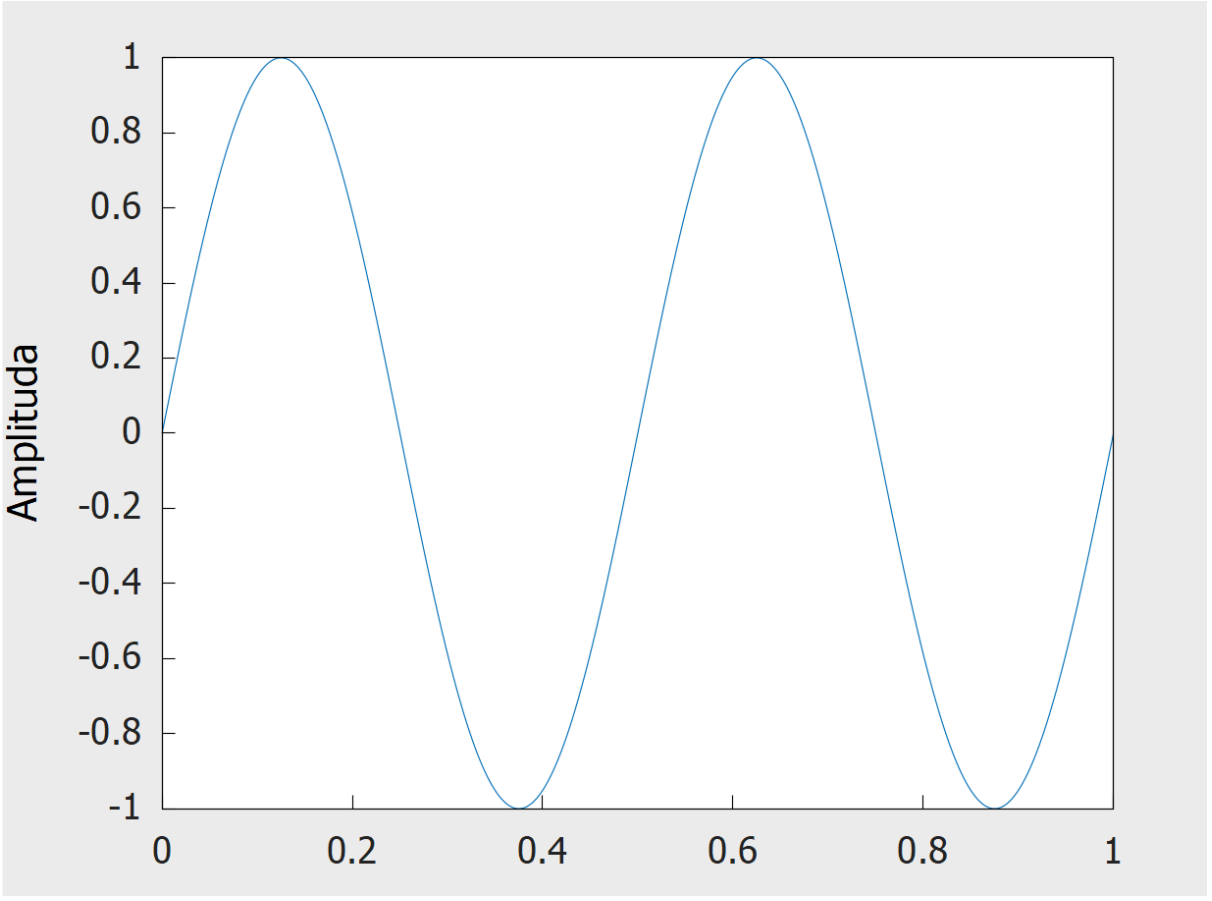
Techniki Programowania

Raport z projektu 3

Oskar Duda 203830
Rafał Zaczek 203450

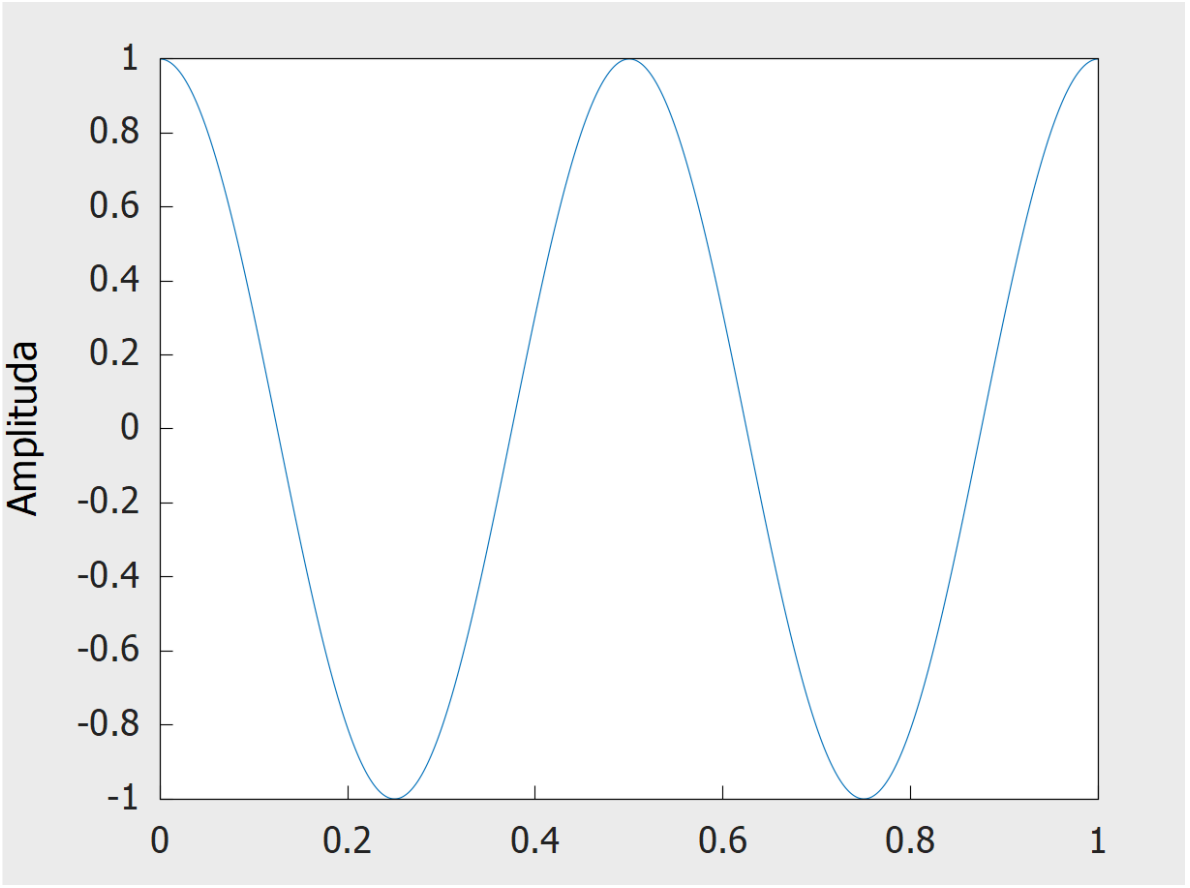
Realizacja punktów 1) i 4) projektu

- **Sygnał sinusoidalny**



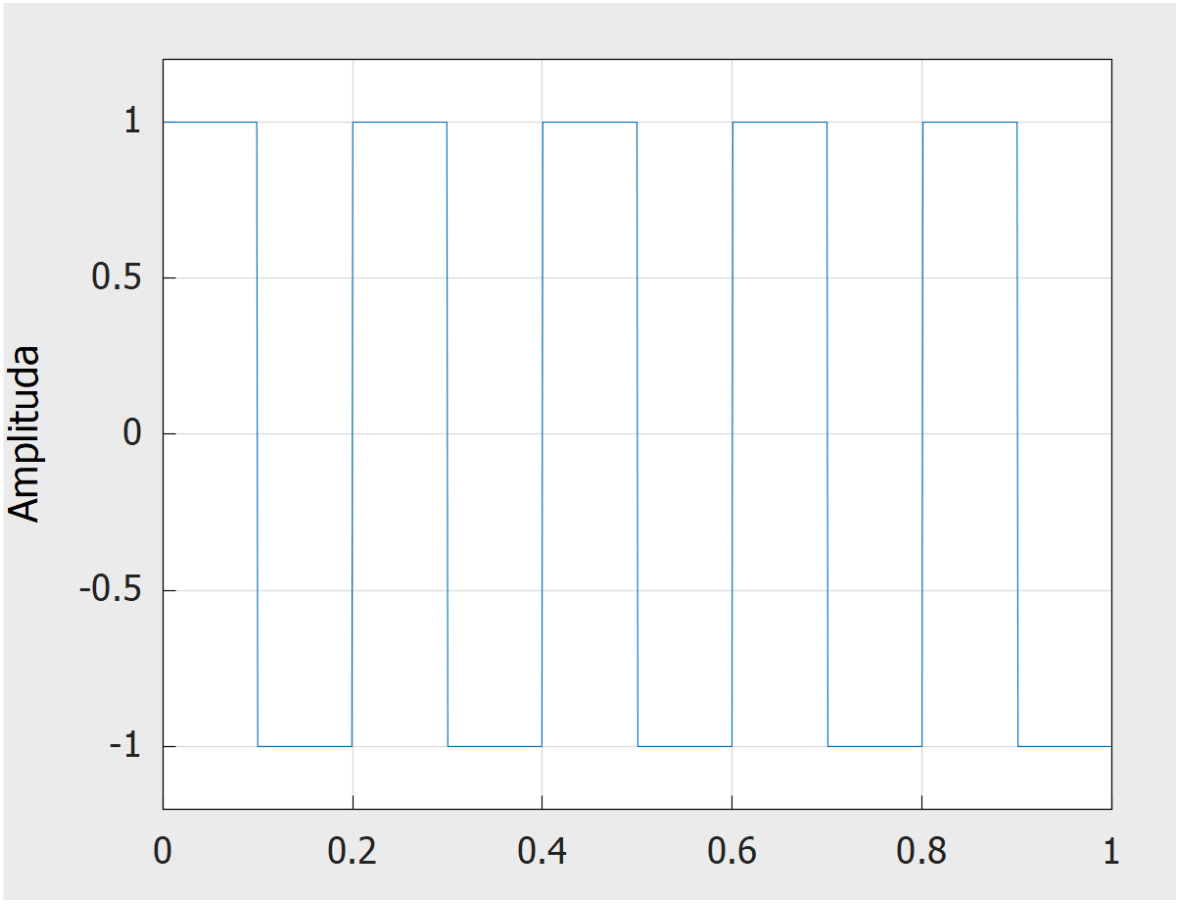
Dane:
freq=2.0
start=0.0
end=1.0
n_samples=1000

- **Sygnał cosinusoidalny**



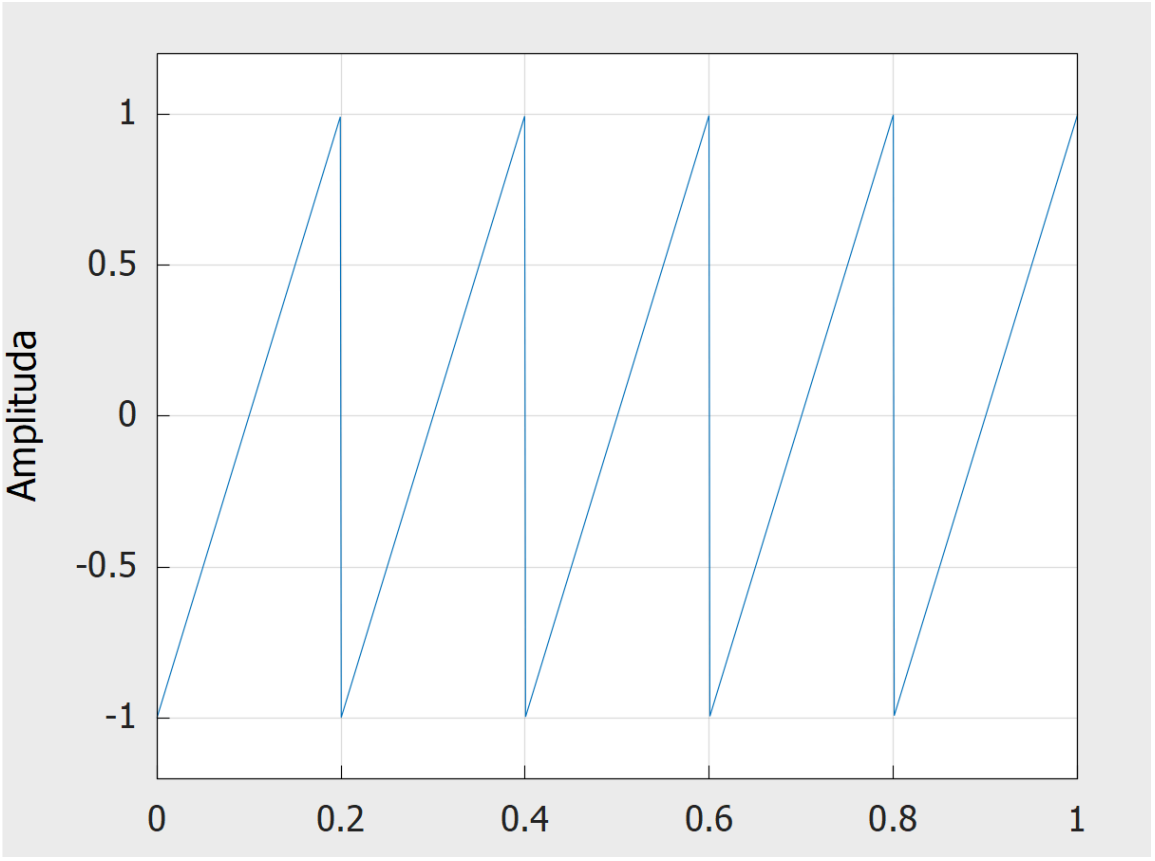
Dane:
freq=2.0
start=0.0
end=1.0
n_samples=1000

- **Sygnał prostokątny**



Dane:
freq=5.0
start=0.0
end=1.0
n_samples=1000

- **Sygnał piłokształtny**

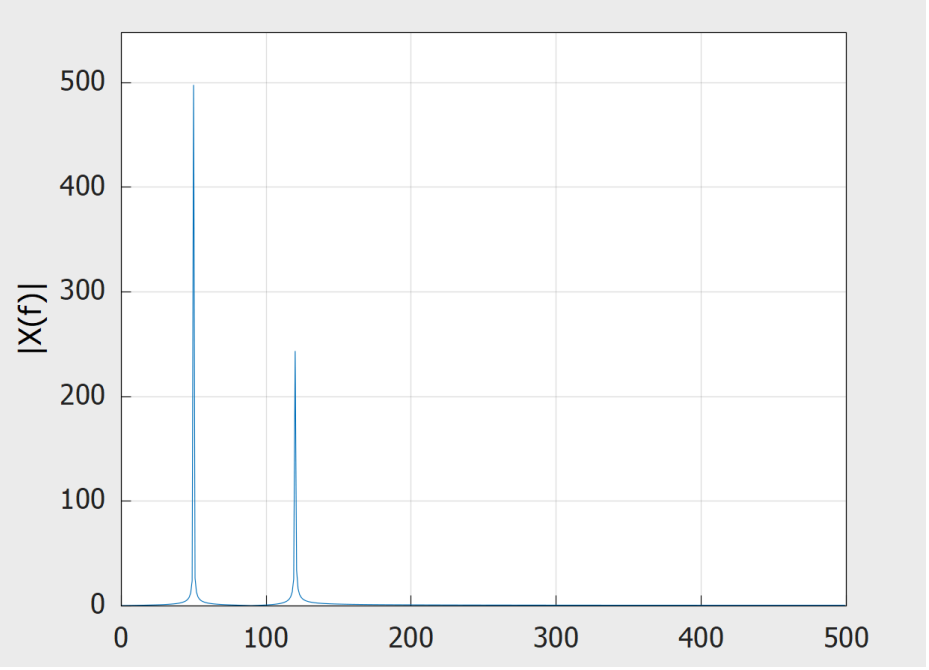


Dane:
freq=5.0
start=0.0
end=1.0
n_samples=1000

Realizacja punktu 2)

- **DFT**

W celu weryfikacji działania transformaty DFT, wygenerowano sygnał będący sumą dwóch sinusoid o częstotliwościach 50 Hz i 120 Hz. Na wykresie widoczne są wyraźne składowe odpowiadające tym częstotliwościom.

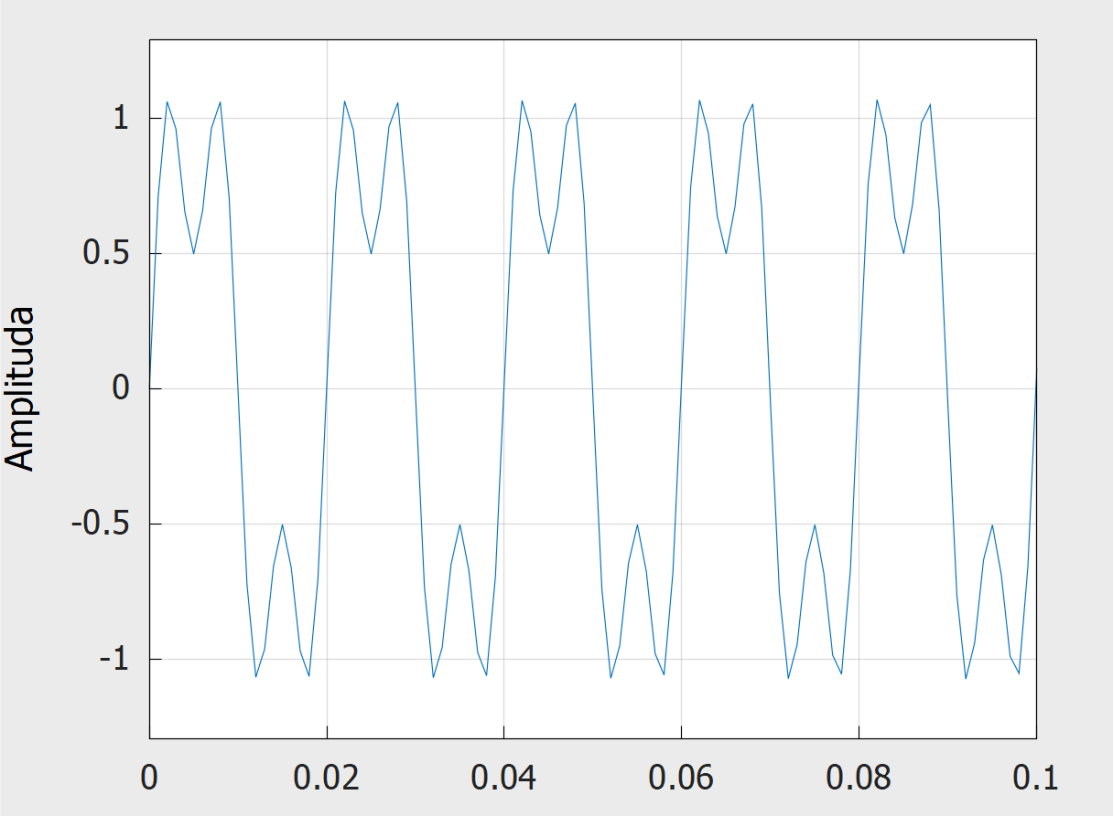


Dane wejściowe i wywołanie funkcji plot_dft() w interpreterze Pythona.

```
>>> sampling_rate = 1000.0
>>> duration = 1.0
>>> n_samples = int(sampling_rate * duration)
>>> signal1 = scikit_build_example.generate_sin(50.0, 0.0, duration, n_samples)
>>> signal2 = scikit_build_example.generate_sin(120.0, 0.0, duration, n_samples)
>>> signal = [s1 + 0.5 * s2 for s1, s2 in zip(signal1, signal2)]
>>> scikit_build_example.plot_dft(signal, sampling_rate)
```

- **Transformata odwrotna(IDFT)**

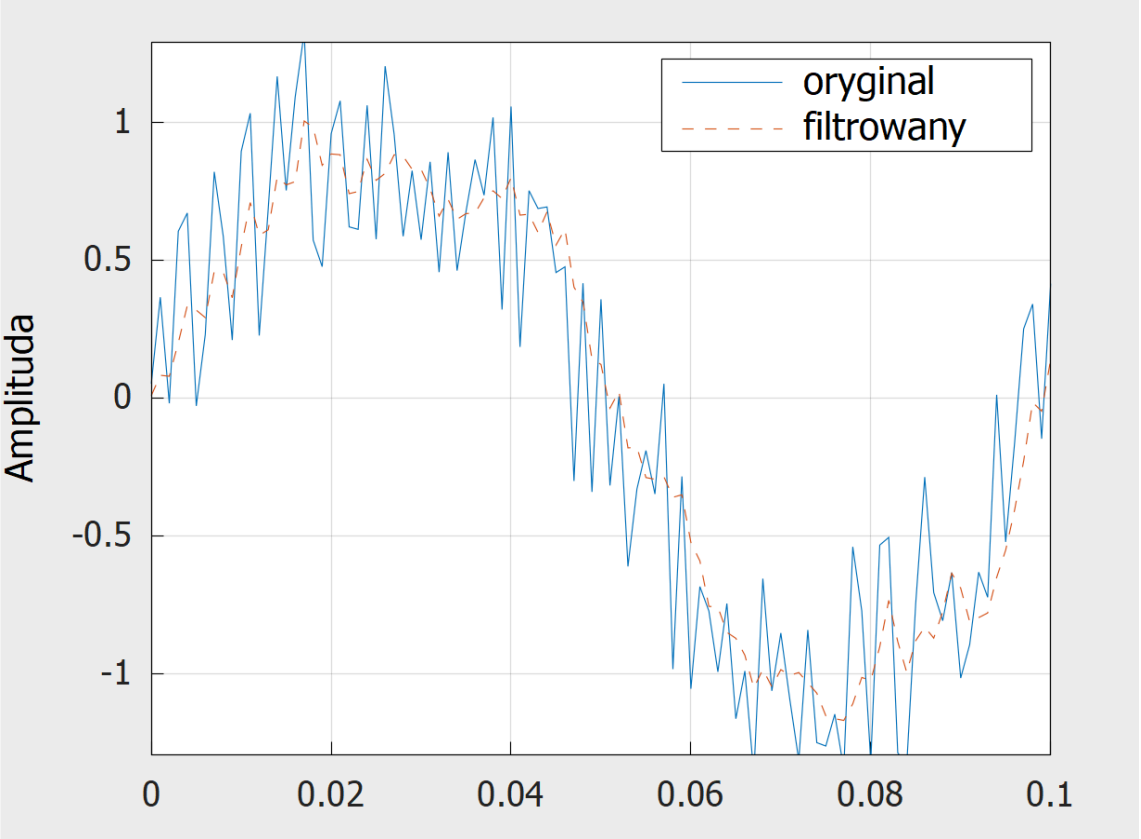
Zrekonstruowany sygnał po odwrotnej transformacie Fouriera (IDFT). Widoczna suma dwóch składowych sinusoidalnych o częstotliwościach 50 Hz i 120 Hz. Zakres czasu ograniczono do 0.1 s w celu poprawy czytelności.



Realizacja punktu 3)

- **Filtracja 1D**

Przykład filtracji sygnału zaszumionego. Zastosowano filtr dolnoprzepustowy (uśredniający). Widoczna znaczna redukcja zakłóceń przy zachowaniu głównego kształtu sygnału.



Fragment kodu w języku Python generujący sygnał sinusoidalny o częstotliwości 10 Hz, dodający do niego losowy szum oraz definiujący współczynniki filtru dolnoprzepustowego typu MA (średnia ruchoma z 5 próbek).

```
>>> sampling_rate = 1000.0
>>> duration = 1.0
>>> n_samples = int(sampling_rate * duration)
>>>
>>> # Sygnał bazowy (sinus) #
>>> clean = scikit_build_example.generate_sin(10.0, 0.0, duration, n_samples)
>>>
>>> # Dodanie szumu #
>>> noisy = [c + 0.5 * random.uniform(-1, 1) for c in clean]
>>>
>>> # Filtr dolnoprzepustowy (np. średnia z 5 próbek) #
>>> b = [1/5] * 5
>>> a = [1.0]
```

- **Filtracja 2D**

W celu zobrazowania działania filtracji dwuwymiarowej zastosowano sztucznie utworzoną macierz wejściową zawierającą prostokątny obszar o wysokich wartościach. Użyto filtru uśredniającego 3×3 (tzw. box blur), którego celem jest wygładzenie ostrych przejść między pikselami.

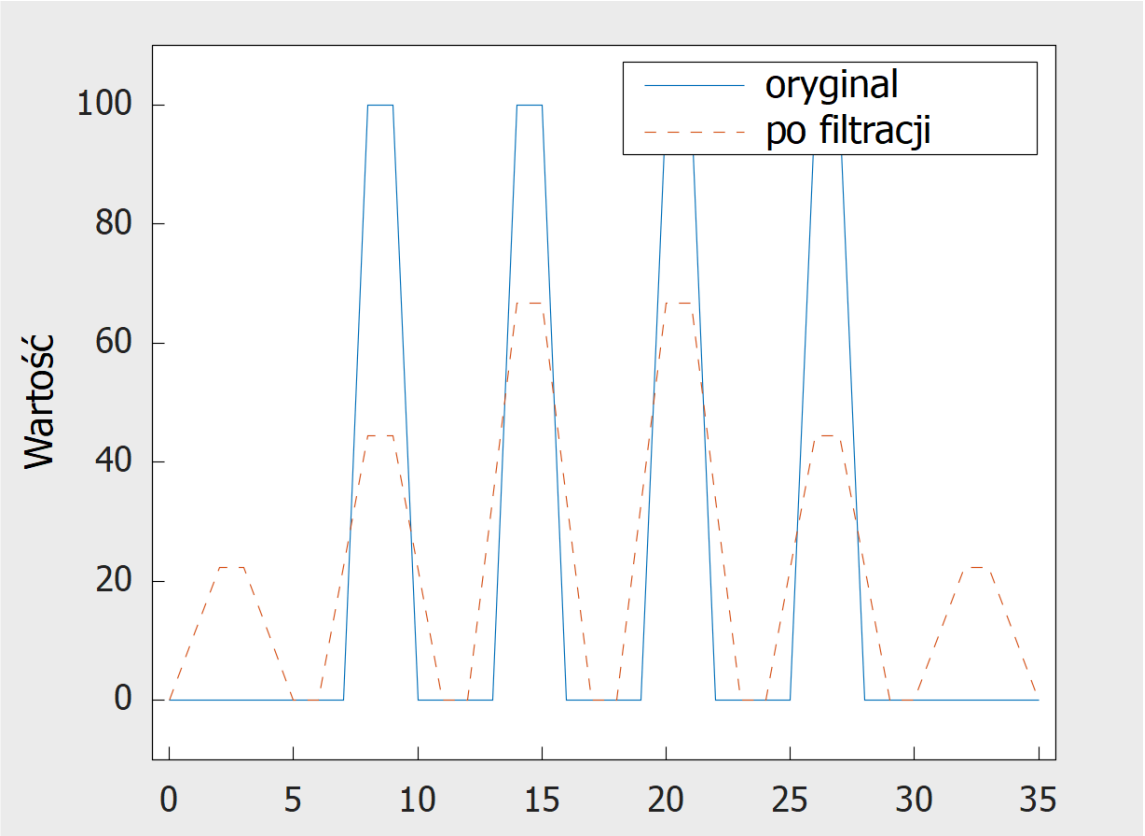
Przykładowa macierz wejściowa image oraz filtr kernel zastosowane do testu filtracji 2D.

```
>>> image = [
...     [0, 0, 0, 0, 0, 0],
...     [0, 0, 100, 100, 0, 0],
...     [0, 0, 100, 100, 0, 0],
...     [0, 0, 100, 100, 0, 0],
...     [0, 0, 100, 100, 0, 0],
...     [0, 0, 0, 0, 0, 0],
...           ]
>>>
>>>
>>> kernel = [
...     [1/9, 1/9, 1/9],
...     [1/9, 1/9, 1/9],
...     [1/9, 1/9, 1/9],
...         ]
```

Wynik działania filtru 2D wypisany w konsoli — wartości z rozmytym centrum i zmniejszonym kontrastem krawędzi.

```
Przefiltrowana macierz (Filtracja 2D):
0.0000  11.1111 22.2222 22.2222 11.1111 0.0000
0.0000  22.2222 44.4444 44.4444 22.2222 0.0000
0.0000  33.3333 66.6667 66.6667 33.3333 0.0000
0.0000  33.3333 66.6667 66.6667 33.3333 0.0000
0.0000  22.2222 44.4444 44.4444 22.2222 0.0000
0.0000  11.1111 22.2222 22.2222 11.1111 0.0000
```

Porównanie wartości spłaszczonych przed i po filtracji 2D. Wygładzenie wyraźnie zmniejsza skoki sygnału w miejscach przejść krawędziowych.



Realizacja wymagania dodatkowego - rozmycie gaussa

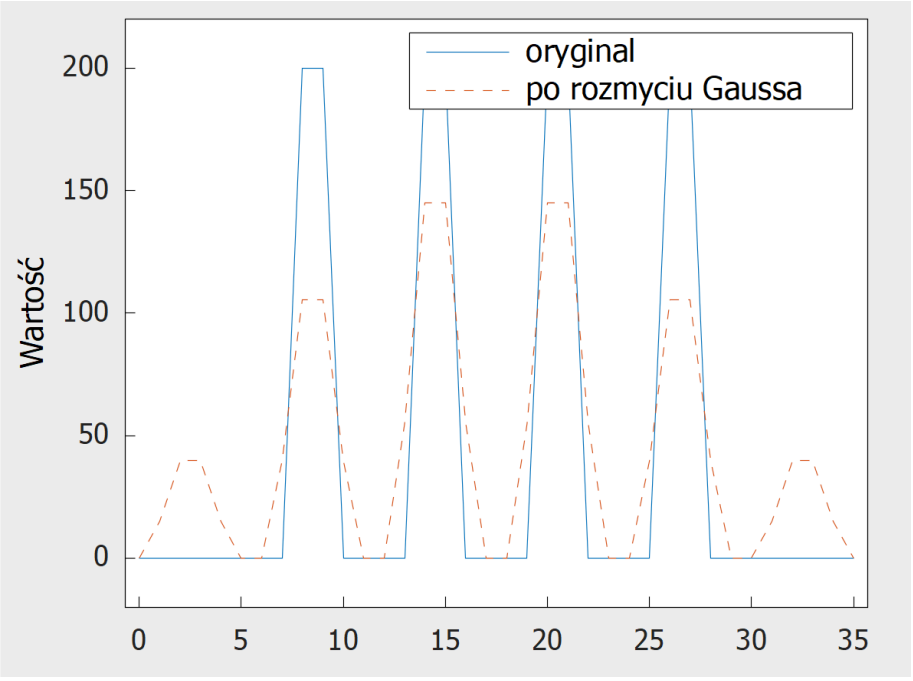
W celu porównania działania filtracji Gaussa zastosowano sztucznie przygotowaną macierz 6×6 zawierającą wyraźne przejścia pomiędzy zerami a wartościami 100. Zastosowany filtr to splot Gaussa z jądrem 3×3 oraz odchyleniem standardowym σ = 1.0. Celem było zredukowanie ostrych przejść, przy jednoczesnym zachowaniu ogólnego kształtu sygnału.

```
image = [
    [0, 0, 0, 0, 0, 0],
    [0, 0, 200, 200, 0, 0],
    [0, 0, 200, 200, 0, 0],
    [0, 0, 200, 200, 0, 0],
    [0, 0, 200, 200, 0, 0],
    [0, 0, 0, 0, 0, 0],
          ]

scikit_build_example.plot_gaussian_blur(image, 3, 1.0)
```

Przefiltrowana macierz wypisana w konsoli po zastosowaniu rozmycia Gaussa. Wartości na brzegach zostały złagodzone, a jasny obszar w centrum rozmyty.

```
Przefiltrowana macierz (rozmycie Gaussa):
    0.0000   15.0227   39.7910   39.7910   15.0227    0.0000
    0.0000   39.7910  105.3953  105.3953   39.7910    0.0000
    0.0000   54.8137  145.1863  145.1863   54.8137    0.0000
    0.0000   54.8137  145.1863  145.1863   54.8137    0.0000
    0.0000   39.7910  105.3953  105.3953   39.7910    0.0000
    0.0000   15.0227   39.7910   39.7910   15.0227    0.0000
```



Porównanie spłaszczonych wartości sygnału przed i po filtracji. Czerwony wykres przedstawia wygładzony wynik rozmycia Gaussa, natomiast niebieski pokazuje oryginalne, skokowe dane. Widoczna redukcja ostrych przejść i lokalne wygładzenie.