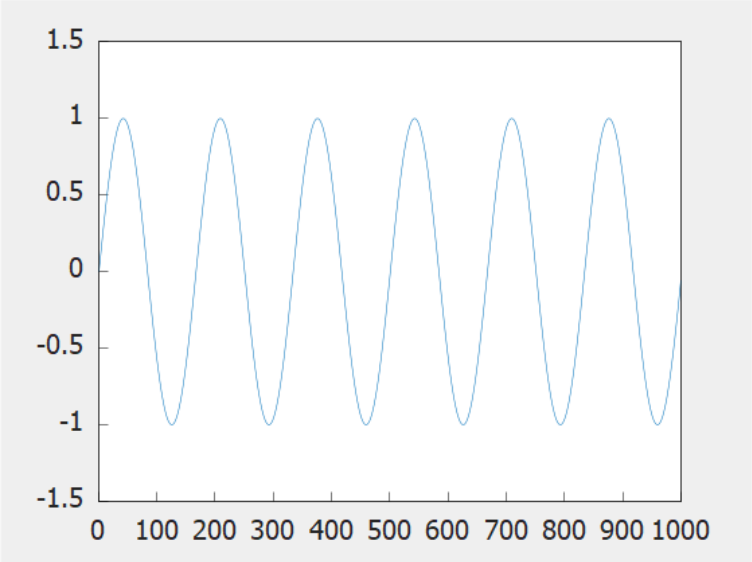
Alicja Morawska 203168

Aleksandra Stosio 203792

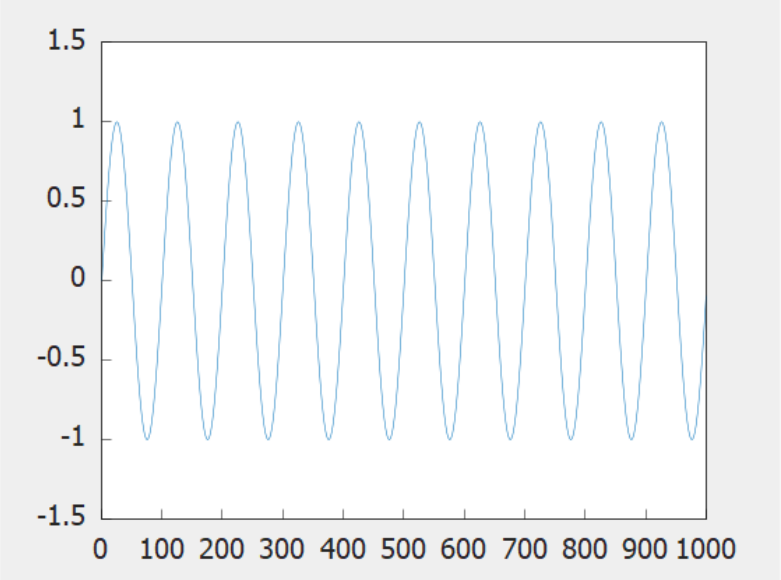
Techniki Programowania – projekt 3

1. Wykres sinus ze zmianą częstotliwości



result = scikit_build_example.generate_sin_signal(1000, 6)

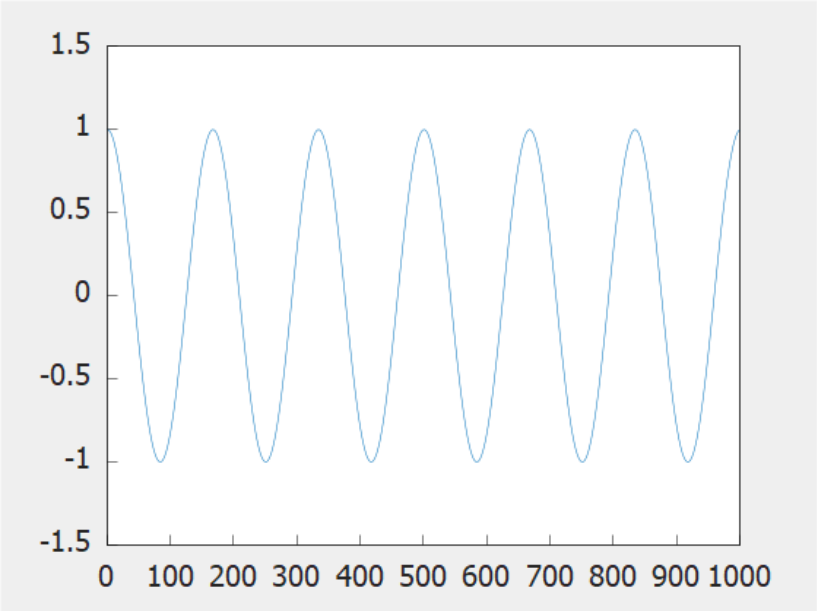
scikit_build_example.plot_signal(result)



result = scikit_build_example.generate_sin_signal(1000, 10)

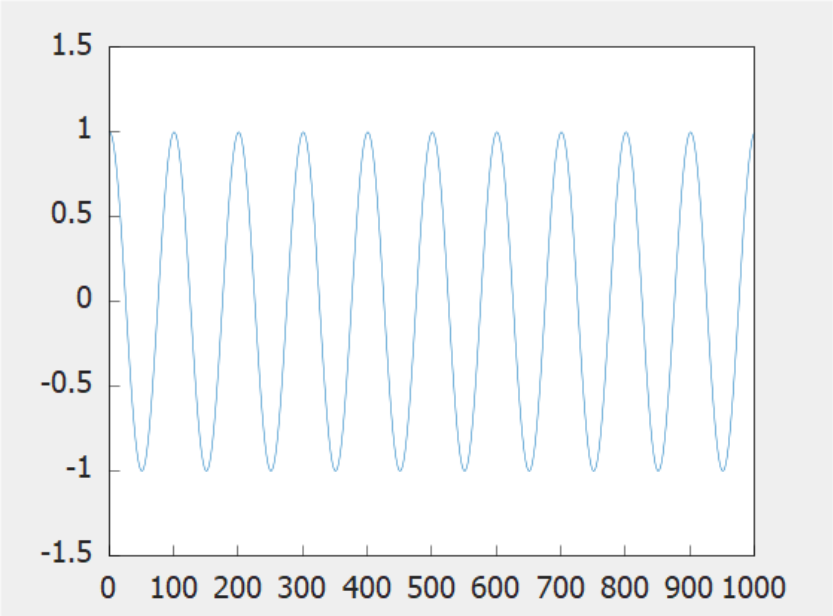
scikit_build_example.plot_signal(result)

2. Wykres cosinus ze zmianą częstotliwości



result = scikit_build_example.generate_cos_signal(1000, 6)

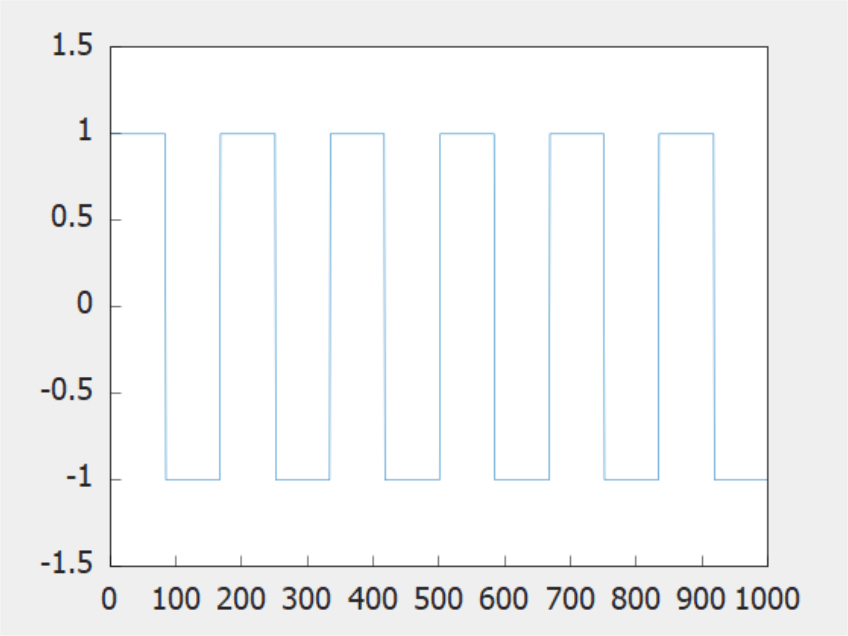
scikit_build_example.plot_signal(result)



result = scikit_build_example.generate_cos_signal(1000, 10)

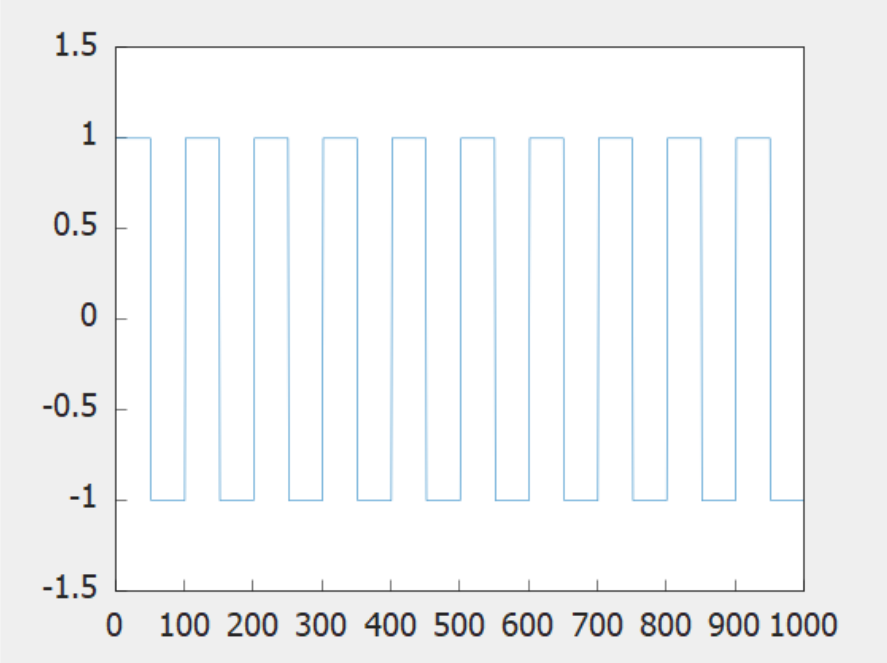
scikit_build_example.plot_signal(result)

3. Wykres prostokątny ze zmianą częstotliwości



result = scikit_build_example.generate_square_signal(1000, 6)

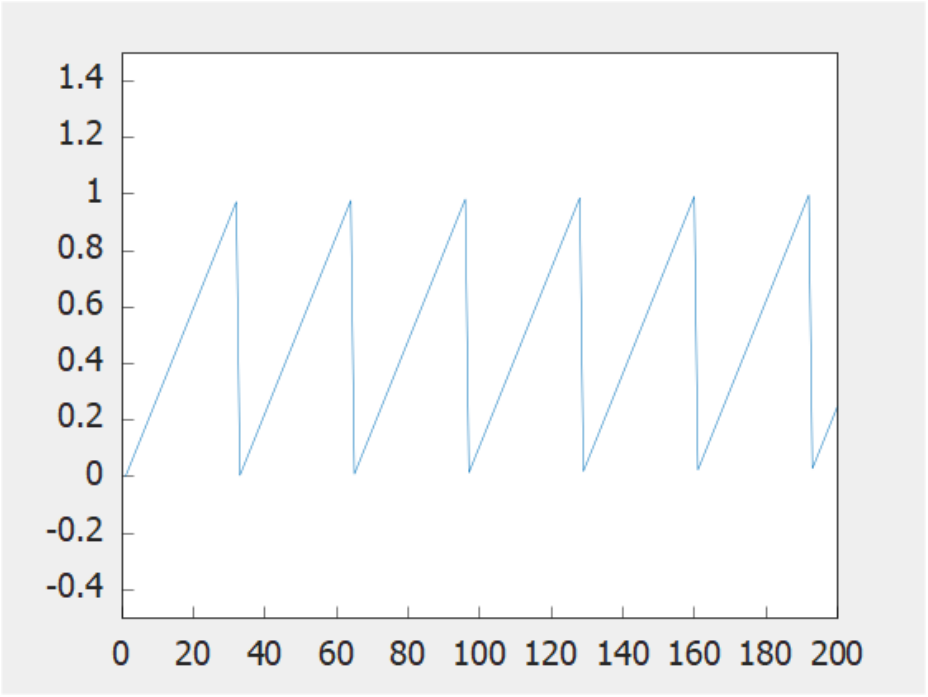
scikit_build_example.plot_signal(result)



result = scikit_build_example.generate_square_signal(1000, 10)

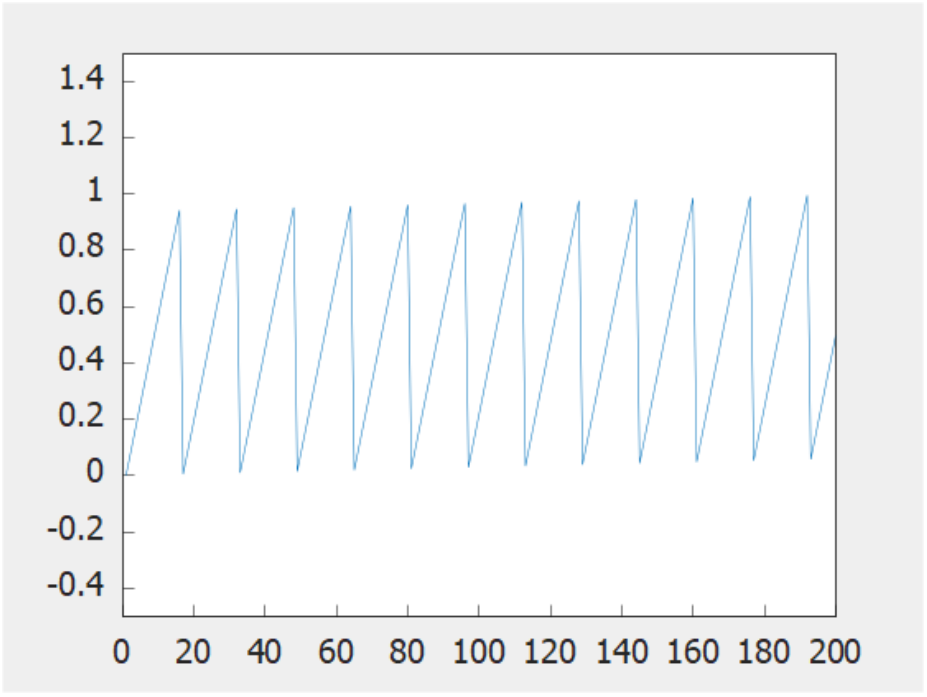
scikit_build_example.plot_signal(result)

4. Wykres piłokształtny ze zmianą częstotliwości



result = scikit_build_example.generate_pilo_signal(200, 1)

scikit_build_example.plot_signal(result)

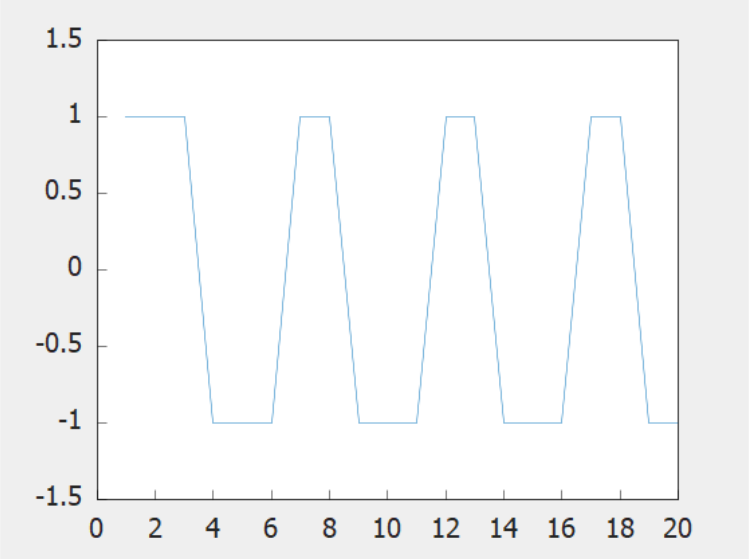


result = scikit_build_example.generate_pilo_signal(200, 2)

scikit_build_example.plot_signal(result)

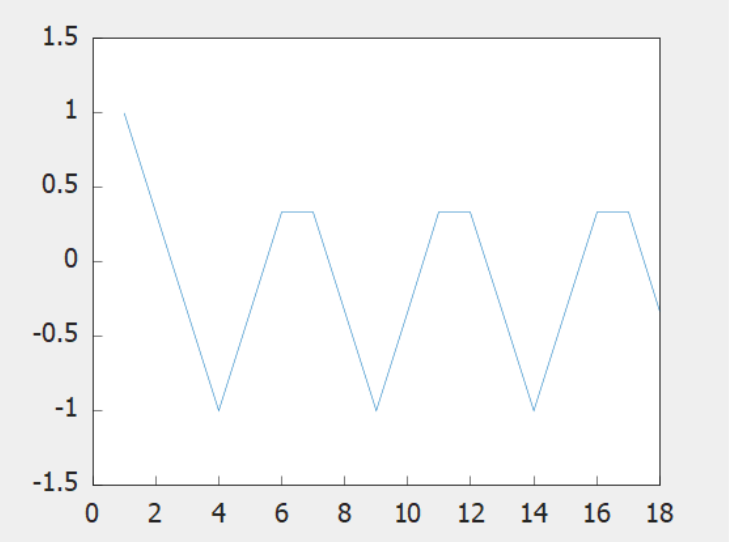
5. Filtracja 1D na bazie wykresu prostokątnego

Filtruje sygnał 1D, wykonując średnią ruchomą (po 3 próbki) w celu wygładzenia szumu lub sygnału.



sig = scikit_build_example.generate_square_signal(20, 4)

scikit_build_example.plot_signal(sig)



sig = scikit_build_example.generate_square_signal(20, 4)

scikit_build_example.plot_signal(sig)

sig2 = scikit_build_example.filter_1d(sig)

scikit_build_example.plot_signal(sig2)

6. Filtracja 2D

Filtracja 2D dla każdego elementu sygnału (z pominięciem brzegów) wyznacza średnią arytmetyczną z wartości tego elementu oraz jego ośmiu sąsiadów, a następnie przypisuje ją do nowej macierzy wynikowej.

```
list_2d = []
N = 10
for i in range(N):
    row = []
    for j in range(N):
        random_number = random.randint(1, 5)
        row.append(random_number)
    list_2d.append(row)

for row in list_2d:
    print(row)

print("-------------------------------------")

filtered_list = scikit_build_example.filter_2d(list_2d)
for row in filtered_list:
    print(row)
```

[4, 4, 1, 1, 2, 4, 4, 5, 2, 4]

[3, 3, 1, 2, 2, 2, 1, 4, 1, 3]

[2, 1, 3, 2, 4, 4, 5, 3, 2, 1]

[4, 2, 3, 2, 1, 1, 3, 3, 5, 5]

[2, 2, 2, 4, 2, 3, 3, 3, 4, 4]

[3, 3, 5, 4, 3, 3, 2, 4, 5, 4]

[3, 1, 1, 4, 1, 1, 1, 3, 1, 2]

[1, 4, 5, 4, 3, 2, 2, 1, 2, 5]

[1, 4, 5, 3, 3, 3, 2, 5, 4, 2]

[4, 5, 2, 1, 3, 3, 3, 3, 3, 4]

[2.4444444444444446, 2.0, 2.0, 2.5555555555555554, 3.111111111111111, 3.5555555555555554, 3.0, 2.7777777777777777]

[2.4444444444444446, 2.111111111111111, 2.2222222222222223, 2.2222222222222223, 2.5555555555555554, 2.888888888888889, 3.0, 3.0]

[2.3333333333333335, 2.3333333333333335, 2.5555555555555554, 2.5555555555555554, 2.888888888888889, 3.111111111111111, 3.4444444444444446, 3.3333333333333335]

[2.888888888888889, 3.0, 2.888888888888889, 2.5555555555555554, 2.3333333333333335, 2.7777777777777777, 3.5555555555555554, 4.111111111111111]

[2.4444444444444446, 2.888888888888889, 2.888888888888889, 2.7777777777777777, 2.111111111111111, 2.5555555555555554, 2.888888888888889, 3.3333333333333335]

[2.888888888888889, 3.4444444444444446, 3.3333333333333335, 2.7777777777777777, 2.0, 2.111111111111111, 2.3333333333333335, 3.0]

[2.7777777777777777, 3.4444444444444446, 3.2222222222222223, 2.6666666666666665, 2.0, 2.2222222222222223, 2.3333333333333335, 2.7777777777777777]

[3.4444444444444446, 3.6666666666666665, 3.2222222222222223, 2.7777777777777777, 2.6666666666666665, 2.6666666666666665, 2.7777777777777777, 3.2222222222222223]

7. DFT

Oblicza Dyskretną Transformatę Fouriera (DFT) dla sygnału 1D, zwracając wektor liczb zespolonych reprezentujących amplitudę i fazę sygnału w dziedzinie częstotliwości. To przykład wykresu sinusaidalnego przekształconego na transformatę DFT. Wynik otrzymuje na przykładzie macierzy.

```
result = scikit_build_example.generate_sin_signal(1000, 2)
result_dft = scikit_build_example.dft(result)
print(result_dft)
```

[(0.004800892979420268+0j), (0.004800856792647606 5+2.0244779912199538e-05j),
(0.00480074823262662 7-500.2535667701684j),
(0.004800567299798473+6.0733513429155734e-05j),
(0.004800313995756204+8.097705421544819e-05j),
(0.004799988321943432+0.00010121976861478797j),
(0.004799590281131191+0.00012146145042620936j),
(0.004799119875604017+0.00014170189312174859j),
(0.004798577108829007+0.00016194089011209228j),
(0.004797961984591714+0.00018217823494632635j),
(0.004797274507125311+0.00020241372098910092j),
(0.004796514681257611+0.00022264714211621623j),
(0.004795682512127 09+0.00024287829172352704j),
(0.004794778005691995+0.00026310696336198 43j),
(0.004793801168173174+0.00028333295074519076j),
(0.004792752006350813+0.00030355604744861445j),
(0.004791630527532613+0.0003237760471448306j),
(0.004790436739473 8+0.0003439927436029596j),
(0.004789170650456937+0.00036420593052943 76j),
(0.004787832269348181 5+0.00038441540167489033j),
(0.004786421605478743+0.0004046209508394641j),
(0.004784938668460949+0.00042482237201612325j),
(0.004783383468706274+0.0004450194588514 4j),
(0.004781756017151955+0.00046521200545407 86j),
(0.004780056324910367+0.00048539980576430 78j)]

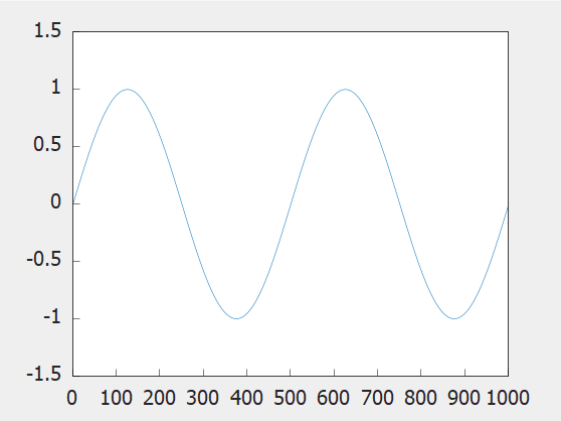
8. IDFT

Oblicza odwrotną DFT (IDFT), przekształcając sygnał z powrotem z dziedziny częstotliwości do dziedziny czasu.

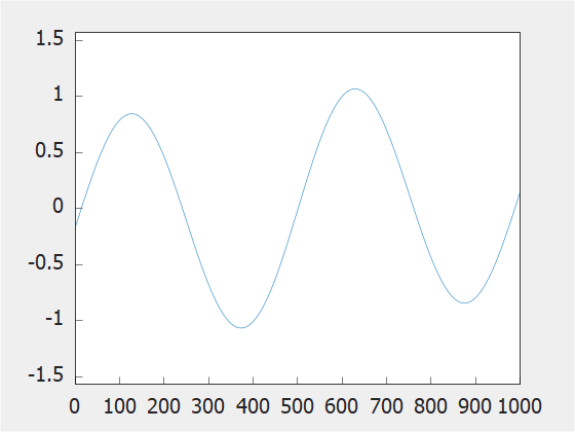
```
result = scikit_build_example.generate_sin_signal(1000, 2)
result_dft = scikit_build_example.dft(result)

idft = scikit_build_example.idft(result_dft)
print()
print(idft)
scikit_build_example.plot_signal(idft)
```

Przed transformatą



Po transformacie



9. Interpolacja dwuliniowa na siatce

Wykonuje dwuliniową interpolację sygnału 2D na zadanej macierzy wartośći do nowej rozdzielczości new_rows x new_cols.

Dane:

```
values = [
    [1, 2, 3],
    [4, 5, 6],
    [7, 8, 9]
]
result = scikit_build_example.interpolacja(values, 6, 6)
for row in result:
    print(row)
```

Wynik:

[1.0, 1.4, 1.8, 2.2, 2.6, 3.0]
[2.2, 2.6, 3.0, 3.4000000000000004, 3.8000000000000003, 4.2]
[3.4000000000000004, 3.8, 4.2, 4.6, 5.0, 5.4]
[4.6000000000000005, 5.0, 5.4, 5.800000000000001, 6.2, 6.6000000000000005]
[5.800000000000001, 6.200000000000001, 6.6, 7.000000000000001, 7.4, 7.8]
[7.0, 7.4, 7.8, 8.2, 8.6, 9.0]