Politechnika Wrocławska

Skalowanie Obrazów

Piotr Frac

November 2025

Spis treści

1 Interpolacja Funkcji

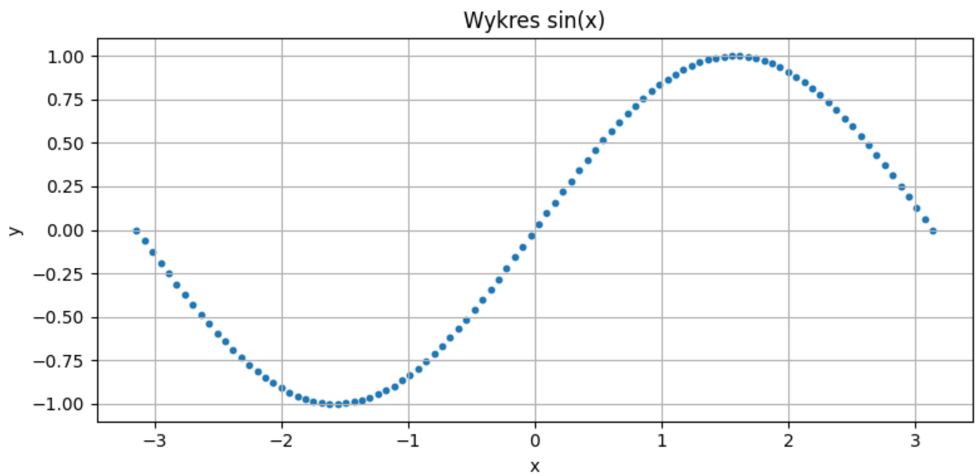
Zadanie polegało na wykonaniu interpolacji trzech funkcji:

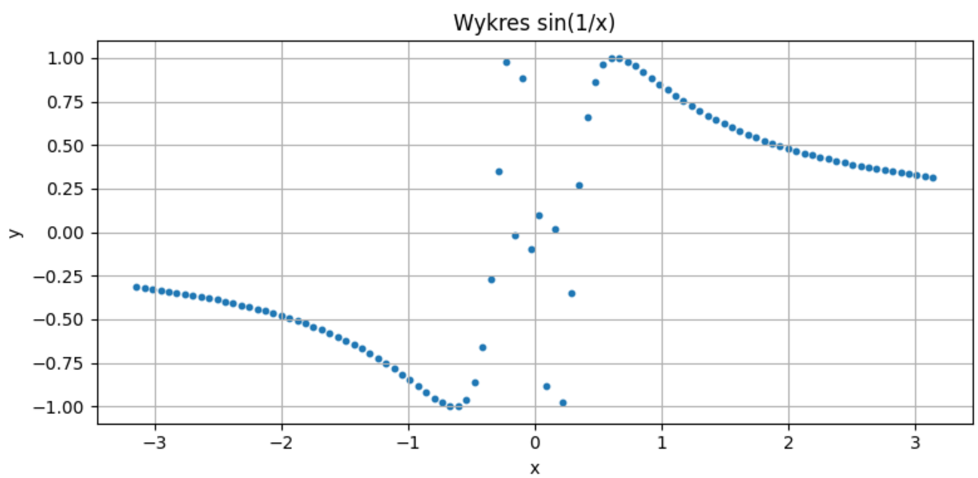
- $\sin(x)$
- $\sin\left(x^{-1}\right)$
- $sgn(\sin(8x))$

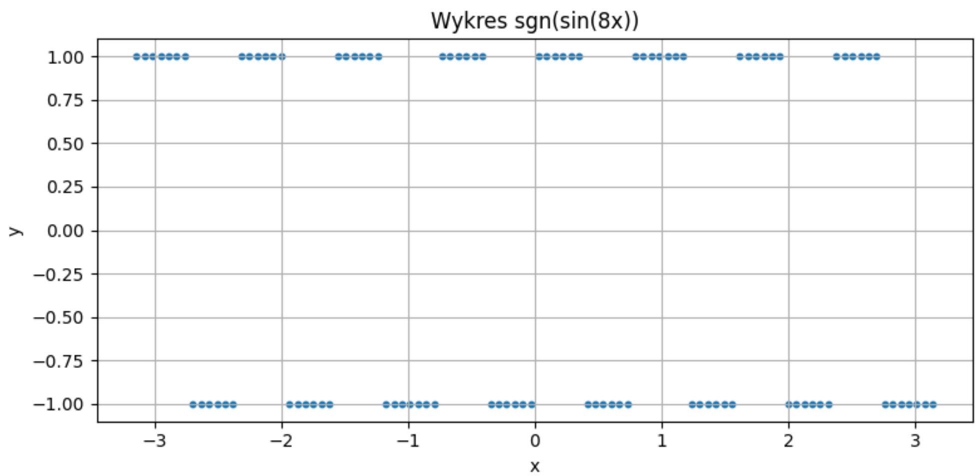
Generujac 2, 4, 10 razy wiecej punktow, dla x $\in [-\pi, \pi]$, wykorzystujac 3 wybrane kernele:

- One-sided rectangular (point-holder)
- Centered rectangular
- Triangular (tent)

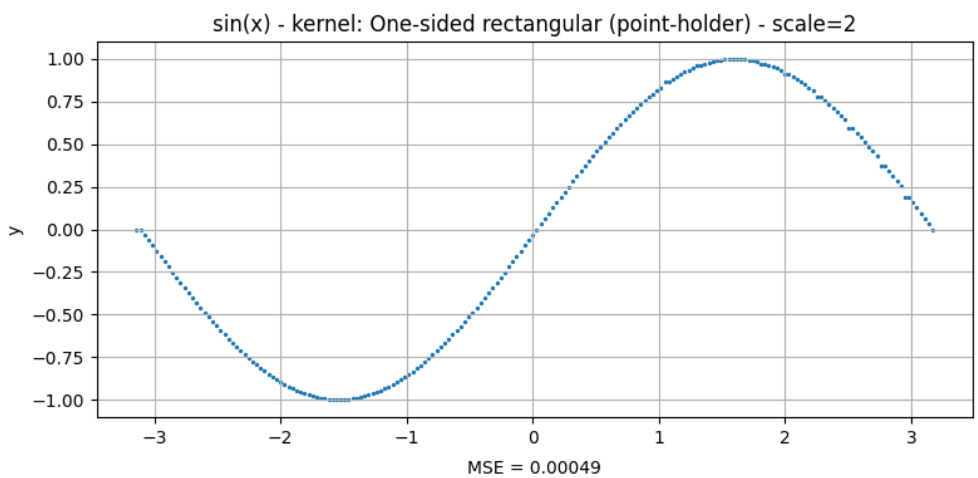
1.1 Wykresy z danych

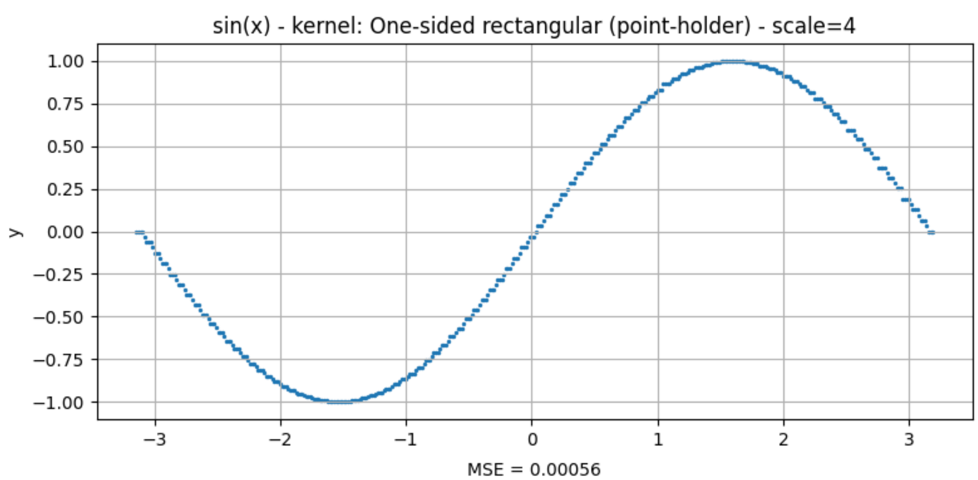


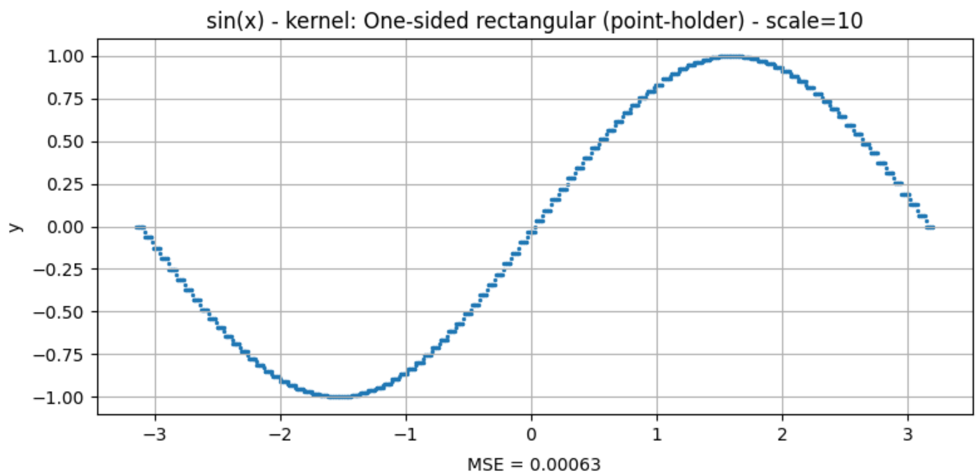




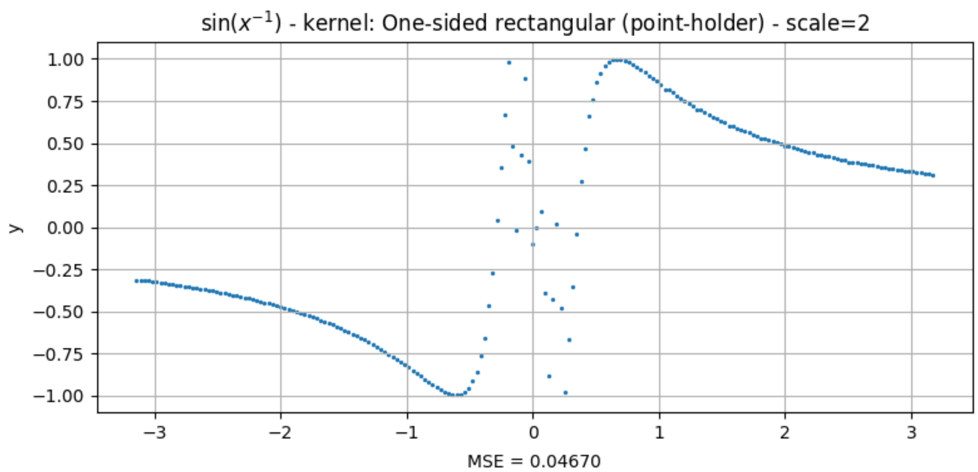
1.2 Wykresy dla Kernela: One-sided rectangular (point-holder)



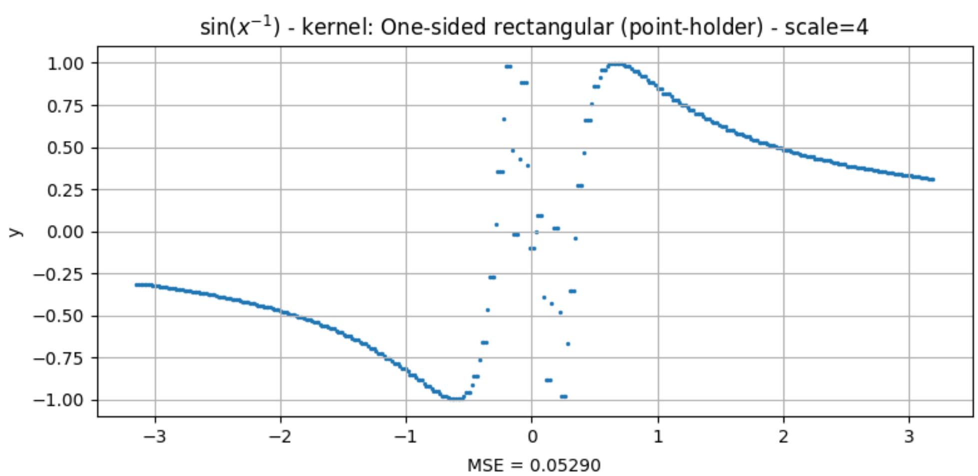




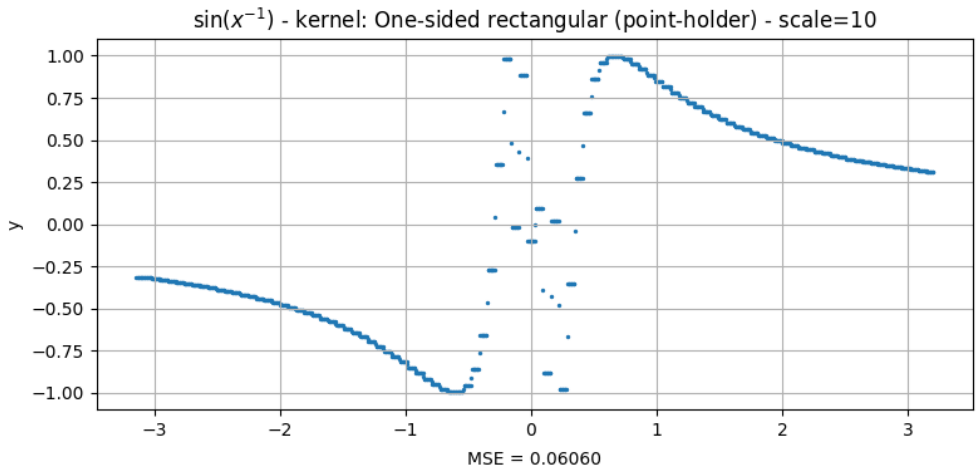
sin(x) - kernel: One-sided rectangular (point-holder) - scale=10
y
MSE = 0.00063

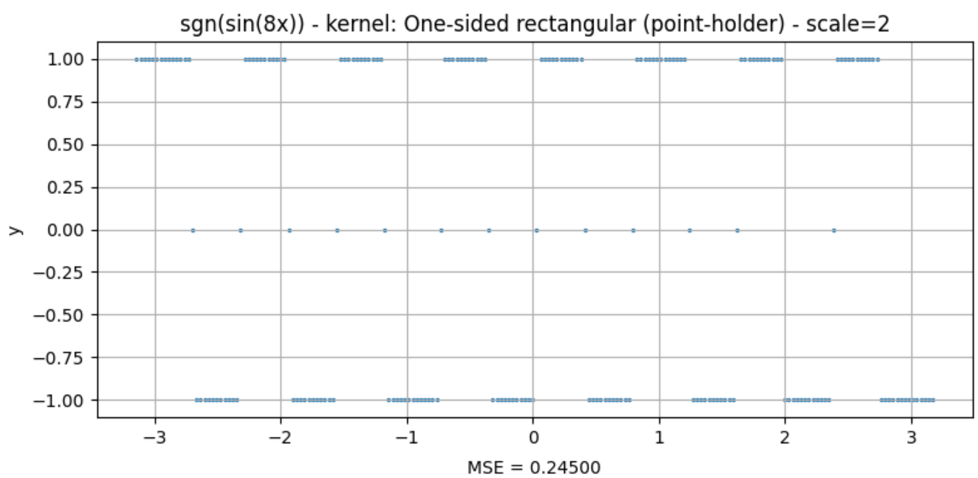


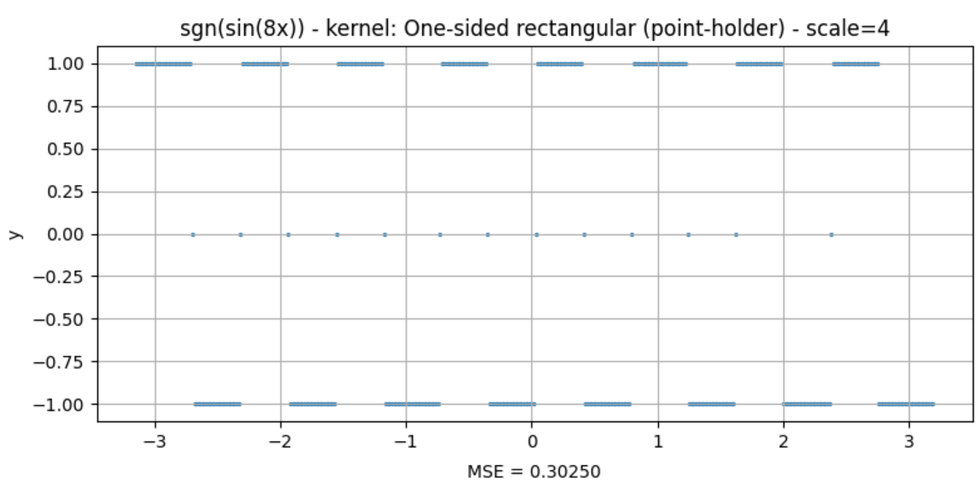
sin(x^{-1}) - kernel: One-sided rectangular (point-holder) - scale=2
y
MSE = 0.04670

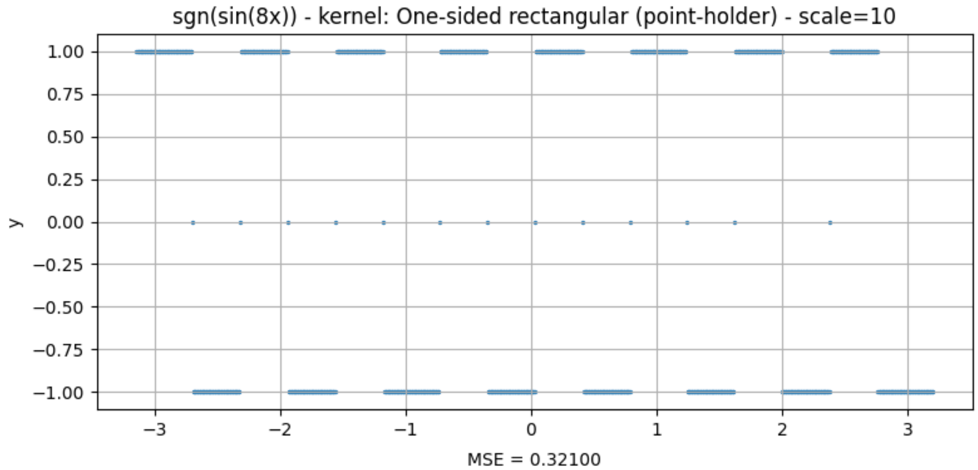


sin(x^{-1}) - kernel: One-sided rectangular (point-holder) - scale=4
y
MSE = 0.05290

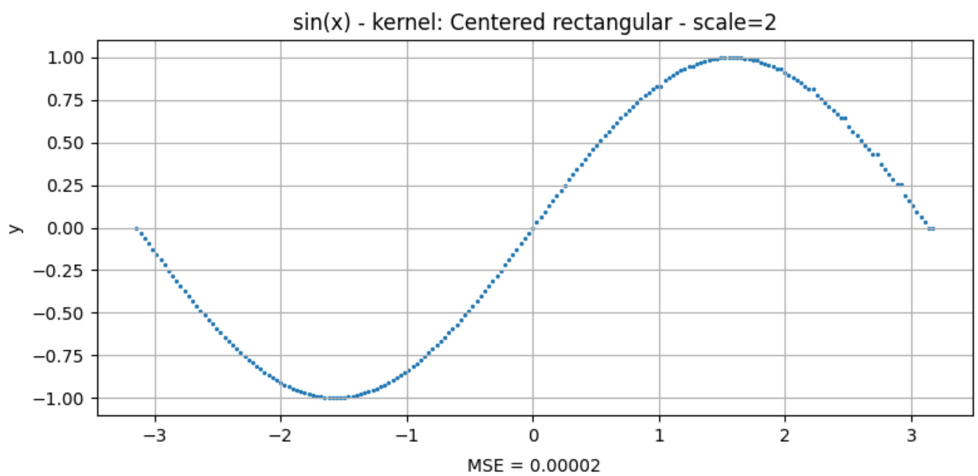

sin(x^{-1}) - kernel: One-sided rectangular (point-holder) - scale=10
y
MSE = 0.06060

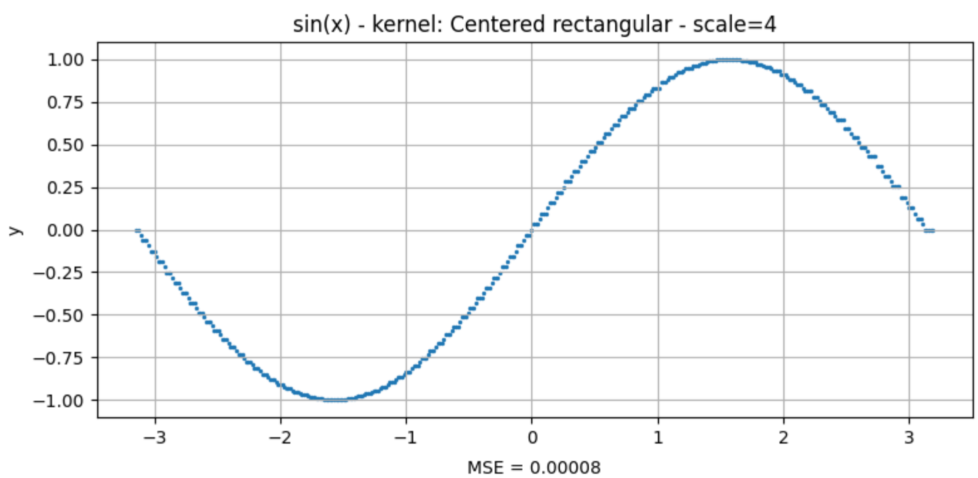

sgn(sin(8x)) - kernel: One-sided rectangular (point-holder) - scale=2
y
MSE = 0.24500

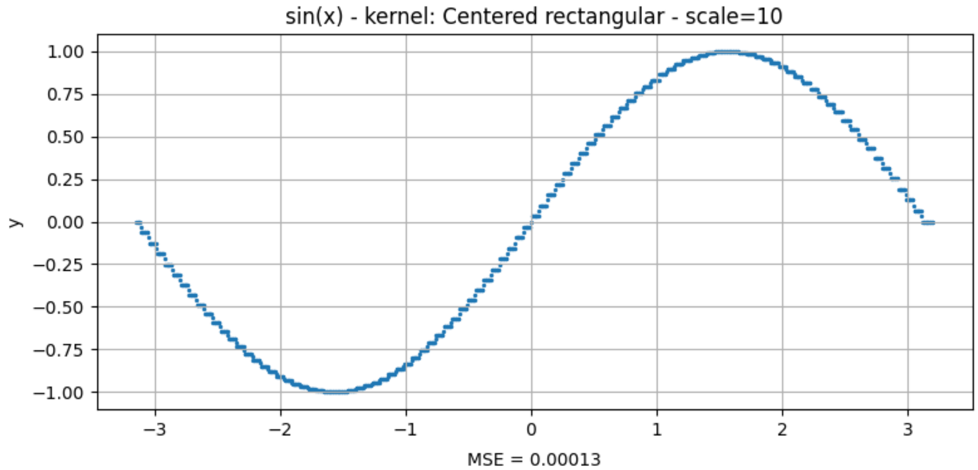

sgn(sin(8x)) - kernel: One-sided rectangular (point-holder) - scale=4
y
MSE = 0.30250

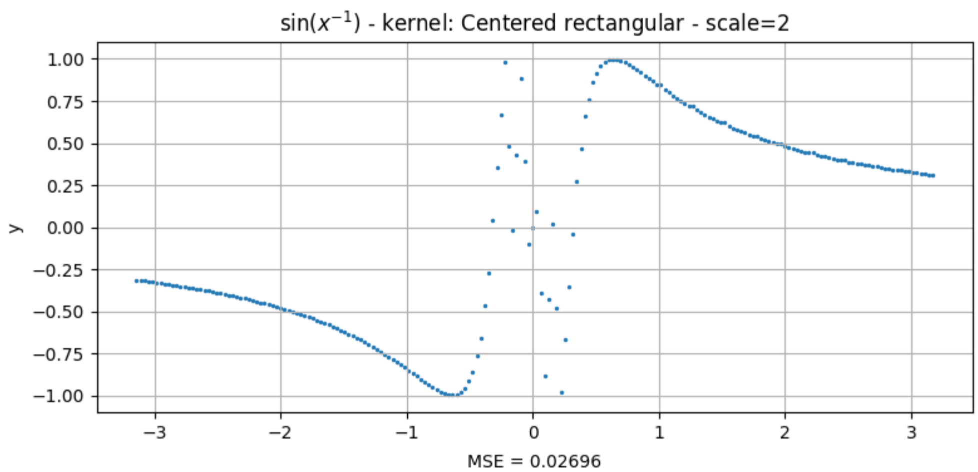


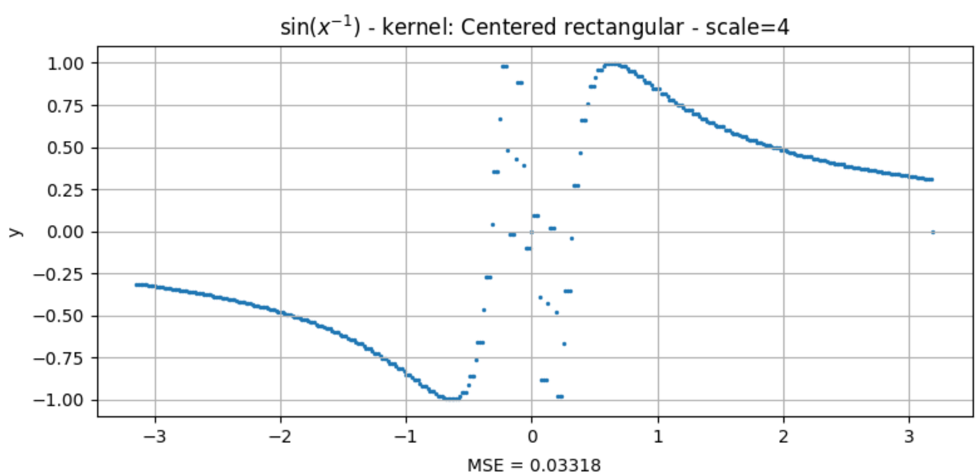
1.3 Wykresy dla Kernela: Centered rectangular

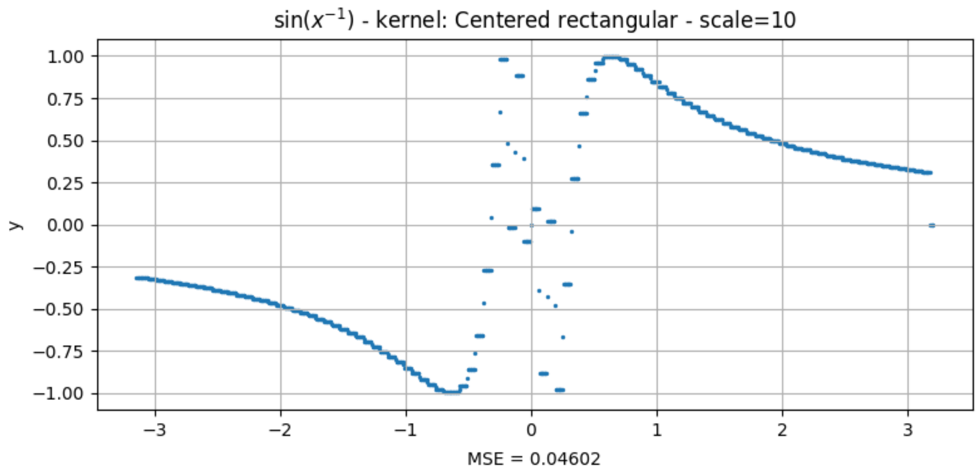


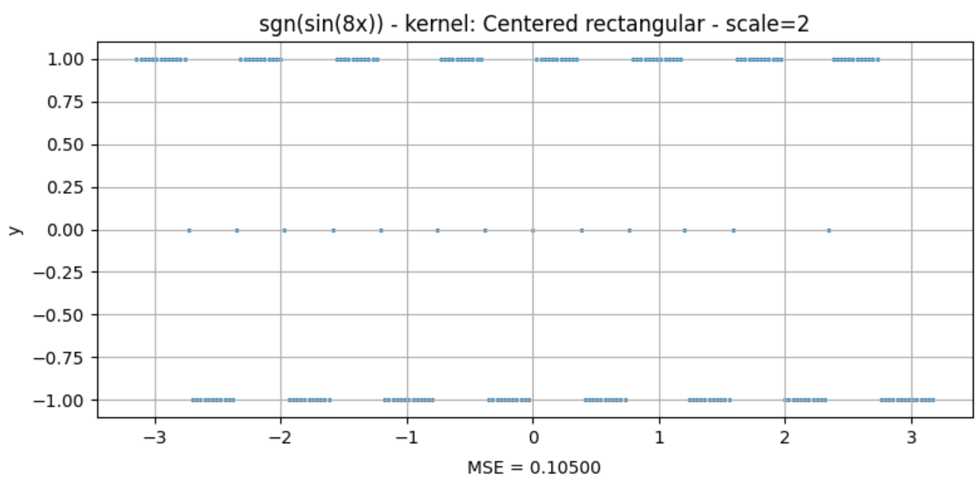


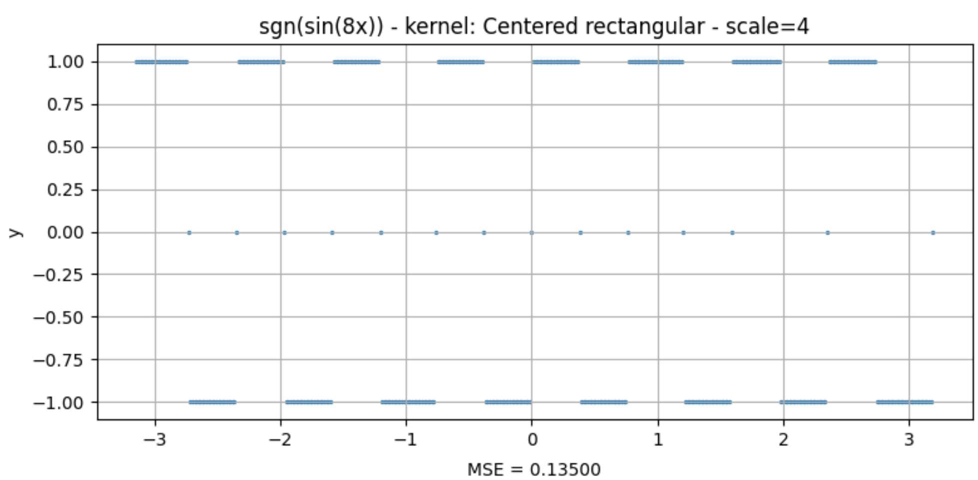

sin(x) - kernel: Centered rectangular - scale=10
MSE = 0.00013

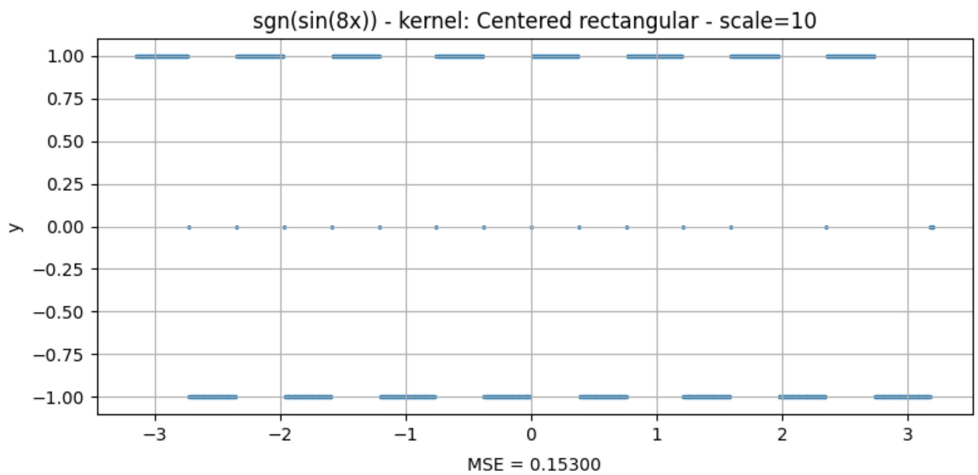

sin(x^{-1}) - kernel: Centered rectangular - scale=2
MSE = 0.02696


sin(x^{-1}) - kernel: Centered rectangular - scale=4
MSE = 0.03318

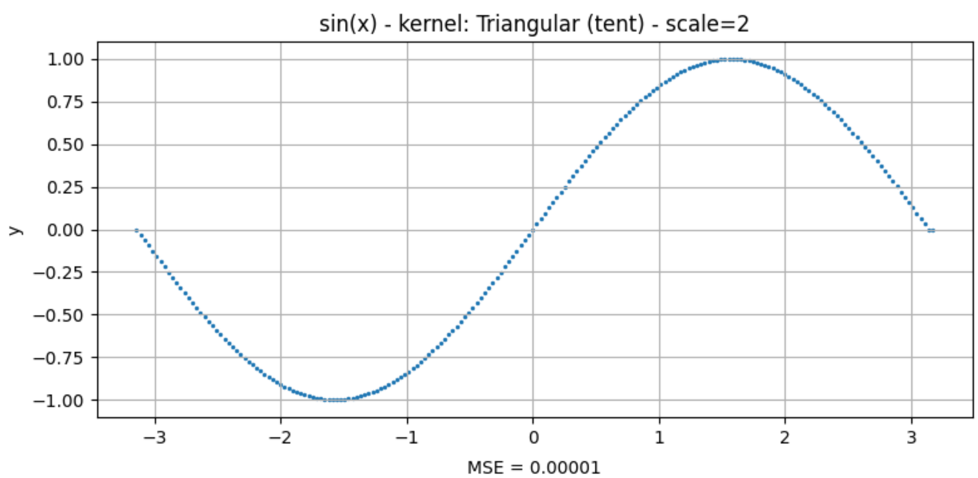

sin(x^{-1}) - kernel: Centered rectangular - scale=10
y
MSE = 0.04602

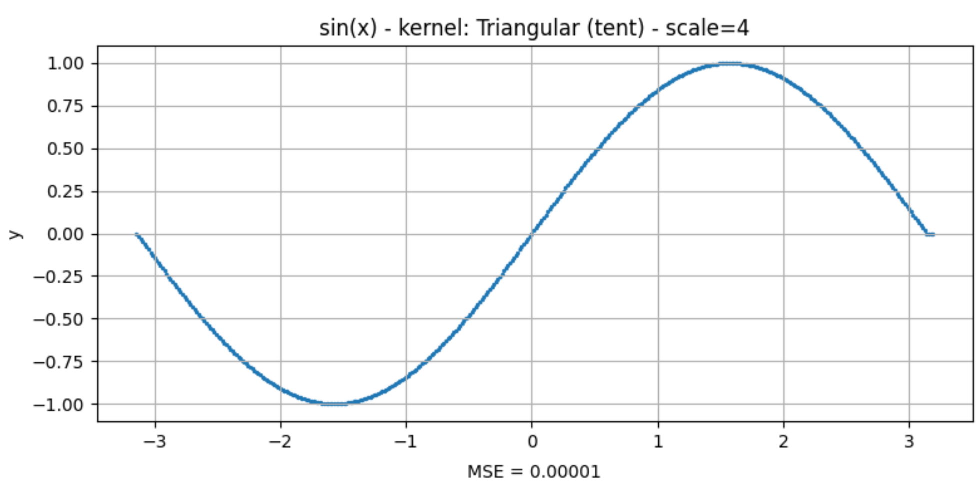

sgn(sin(8x)) - kernel: Centered rectangular - scale=2
y
MSE = 0.10500

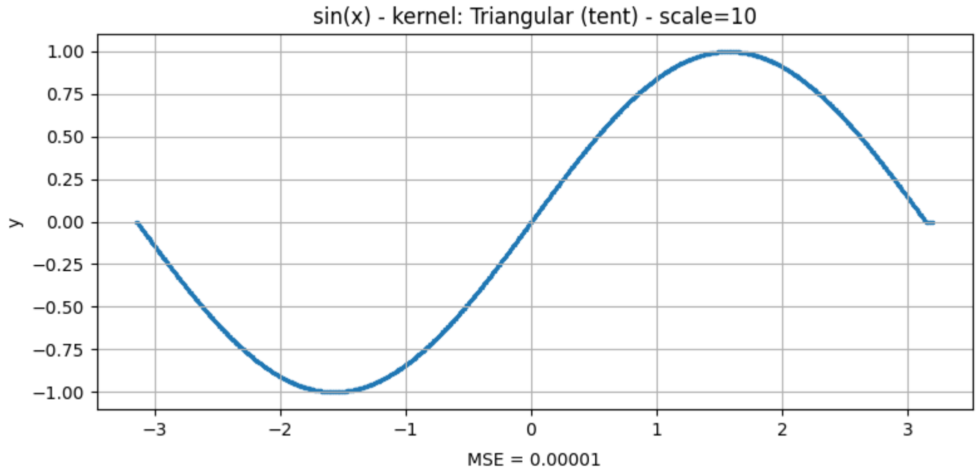

sgn(sin(8x)) - kernel: Centered rectangular - scale=4
y
MSE = 0.13500

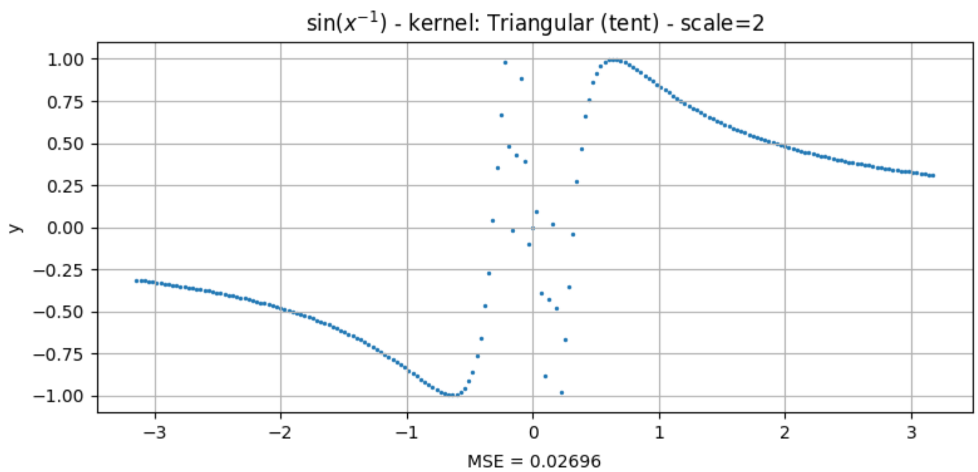


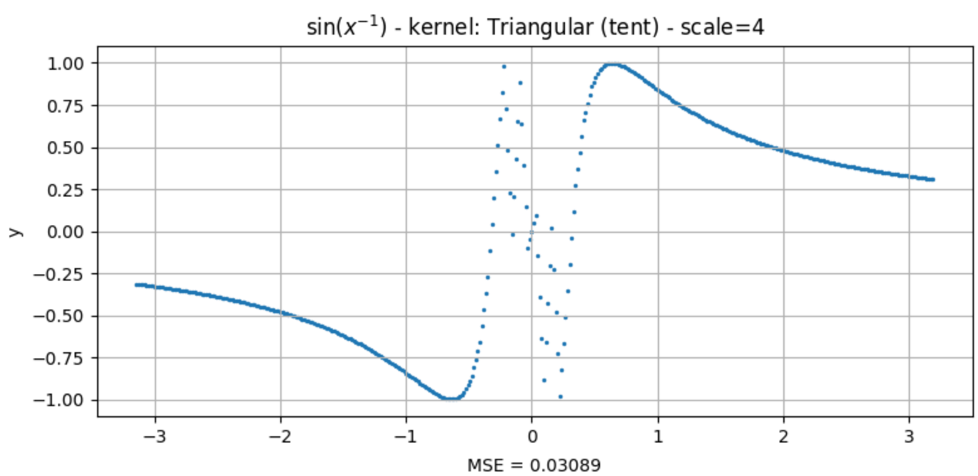
1.4 Wykresy dla Kernela:Triangular (tent)

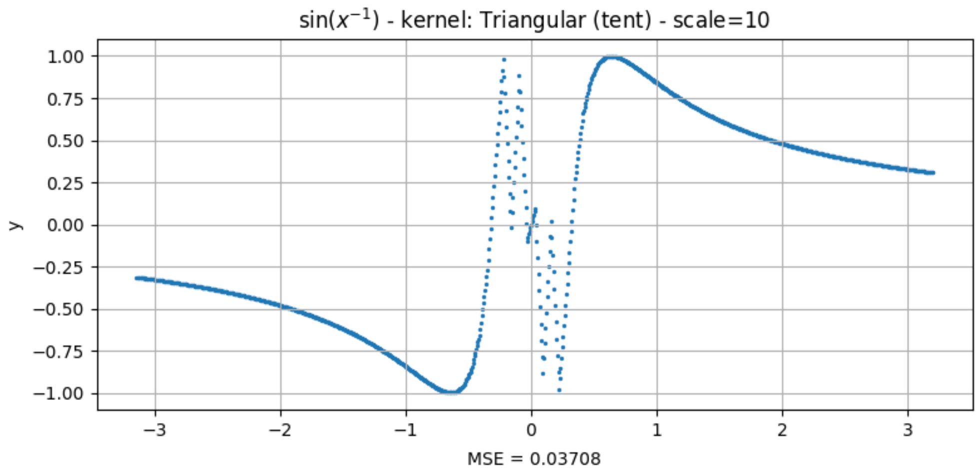




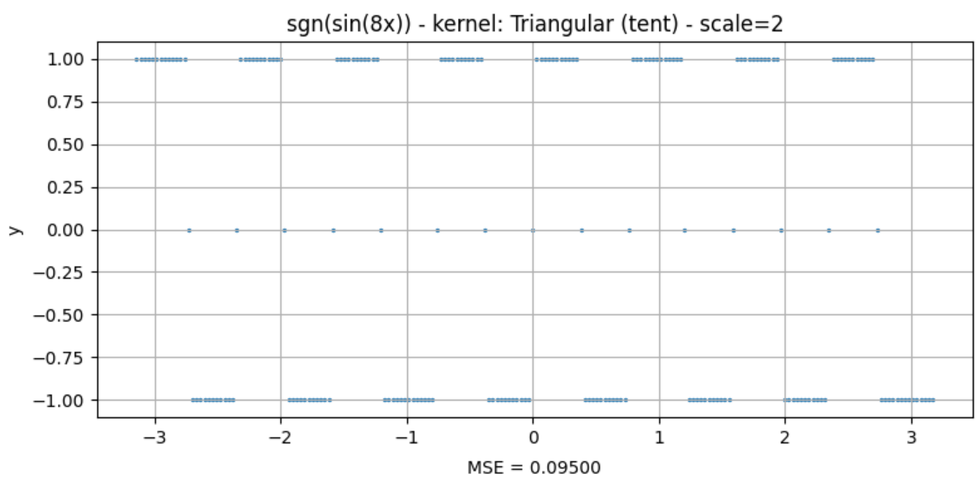

sin(x) - kernel: Triangular (tent) - scale=10
y
x
MSE = 0.00001


sin(x^{-1}) - kernel: Triangular (tent) - scale=2
y
MSE = 0.02696

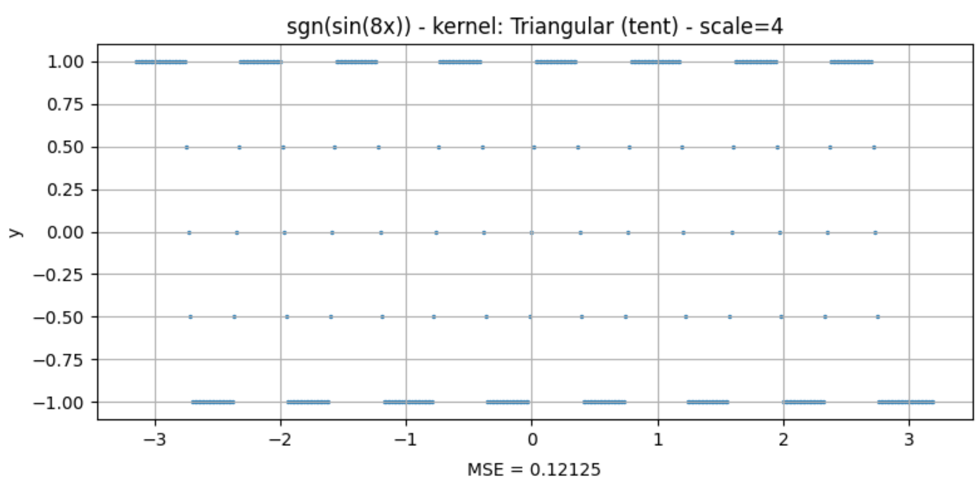

sin(x^{-1}) - kernel: Triangular (tent) - scale=4
y
MSE = 0.03089



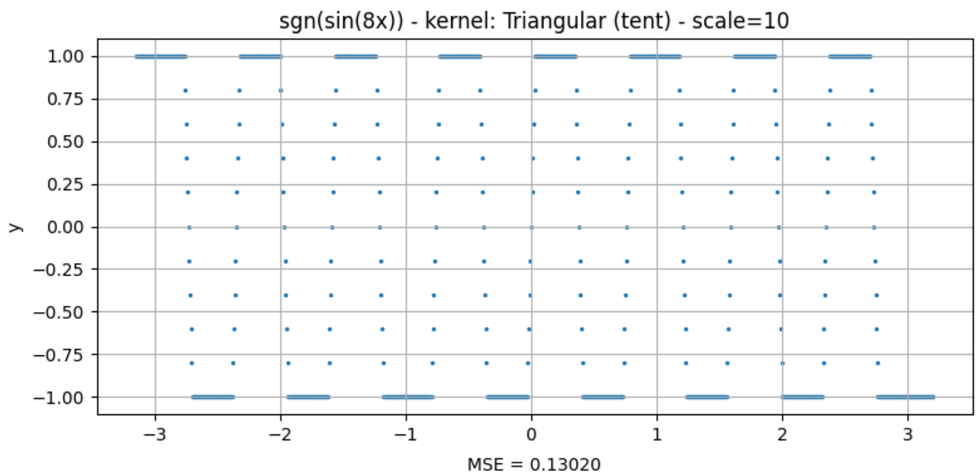
sin(x^{-1}) - kernel: Triangular (tent) - scale=10
y
MSE = 0.03708



sgn(sin(8x)) - kernel: Triangular (tent) - scale=2
y
MSE = 0.09500

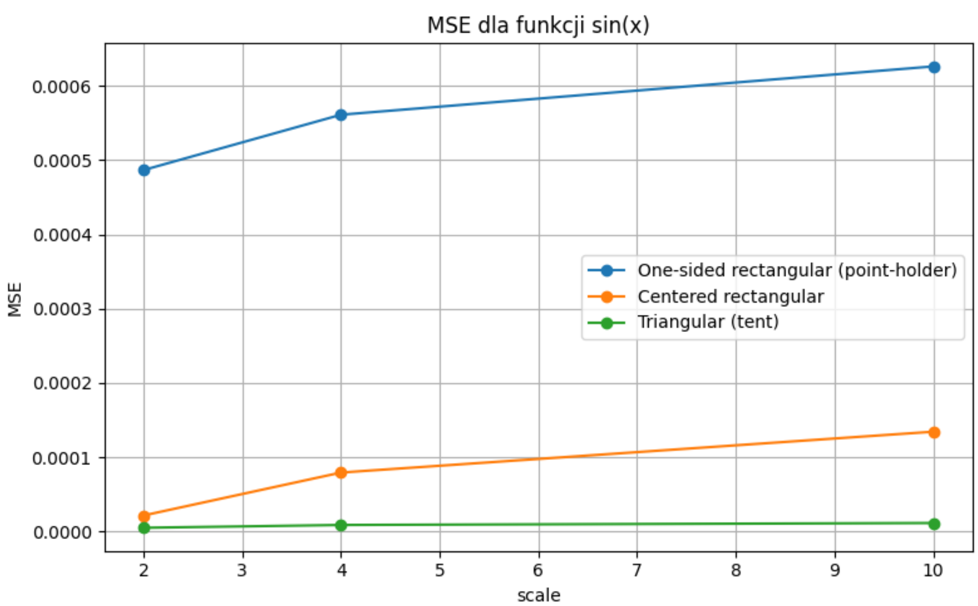


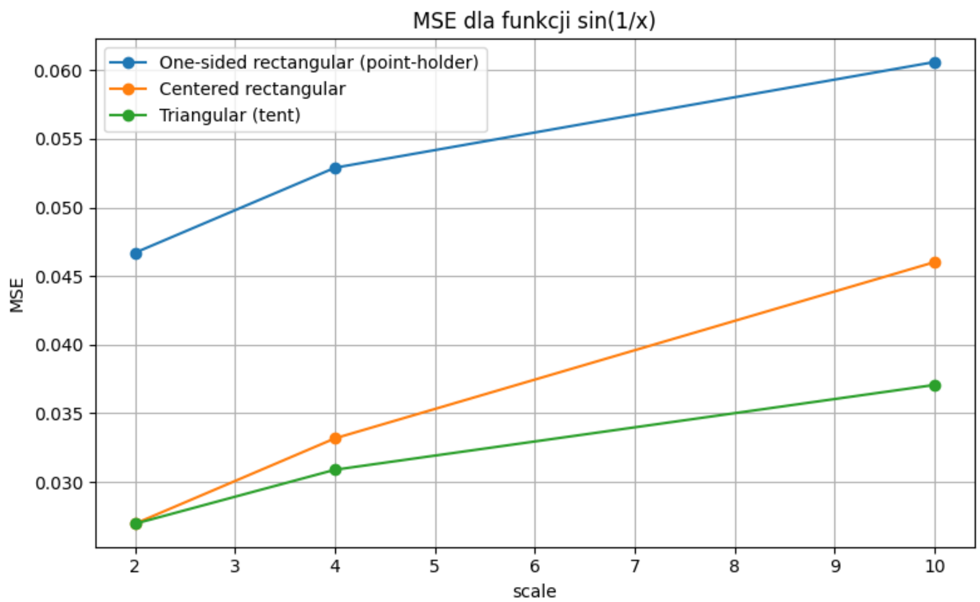
sgn(sin(8x)) - kernel: Triangular (tent) - scale=4
y
MSE = 0.12125

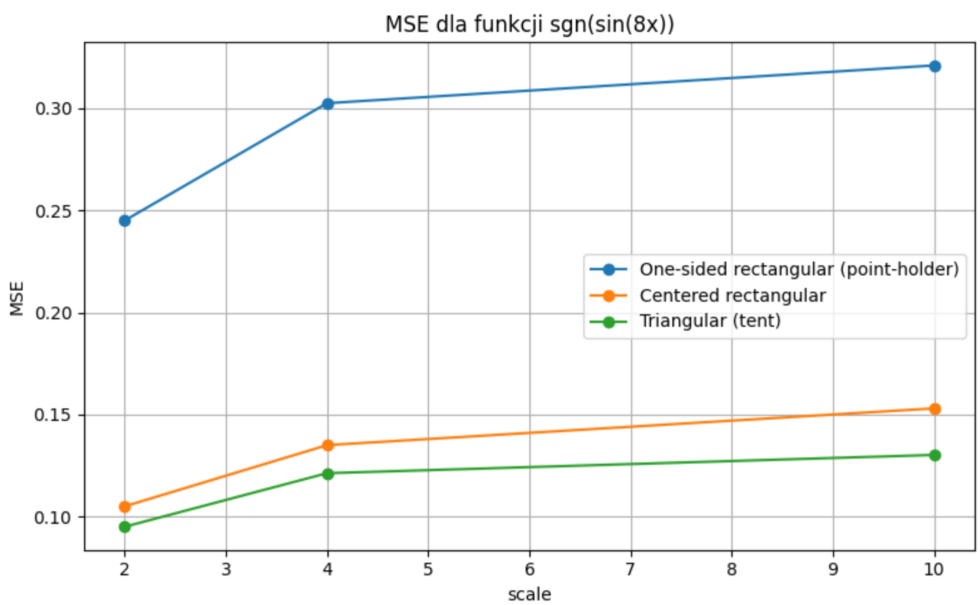


1.5 MSE

Na wykresach poniżej możemy zaobserowac trend rosnacy MSE wraz ze wzrostem skali, szegolnie pomiedzy skalowaniem 2-4 i, ze najlepszym z tych trzech kerneli jest Triangular Kernel





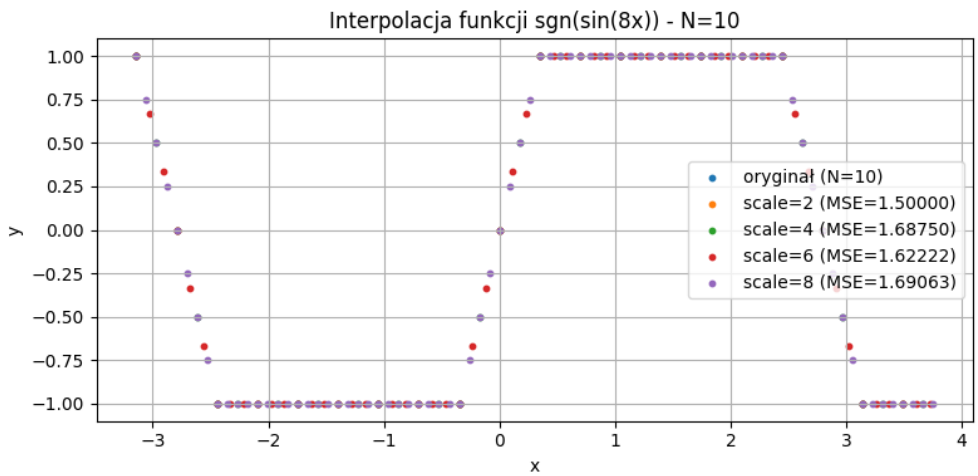


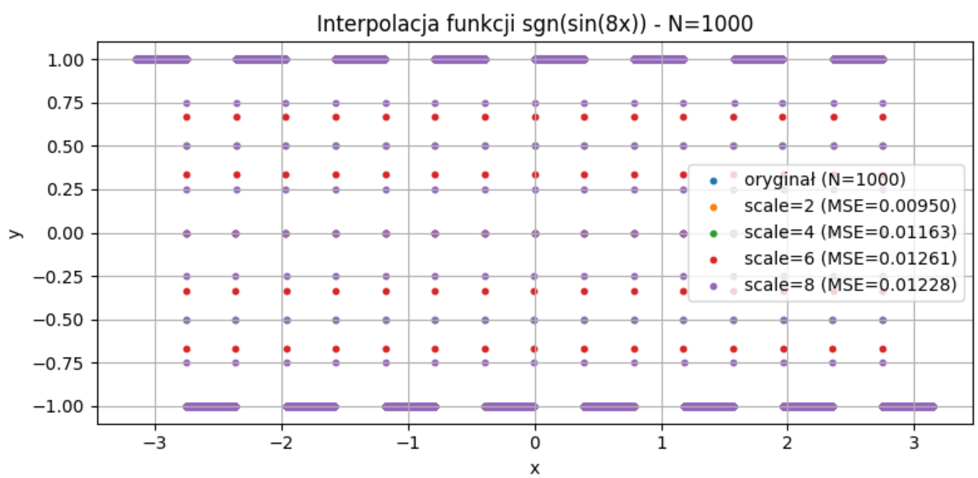
1.6 Ilosc liczby punktow na jakosc konwolucji

Do wykonania tego zadania wykorzystalem:

- Funkcje: sgn(sin(8x))
- Kernel: Triangular

Stworzyłem dwa zestawy punktow dla N = 10 i dla N = 1000 i odpowiednio przeskalowalem 2, 4, 6, 8 krotnie.





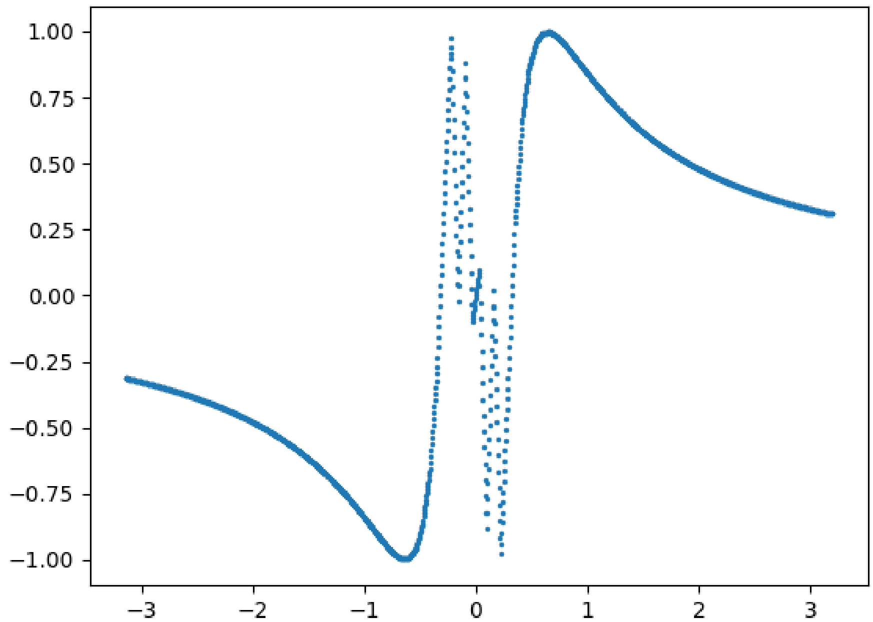
Jak widac na powyzszych wykresach mala ilosc punktow skutkuje, duzym odchyleniem od funkcji orginalnej, gdzie srednie MSE wynosilo 1.6251. Za to duza ilosc skutkuje bardzo zblizona funkcje przy srednim MSE = 0.0115

1.7 Sprawdzic roznice pomiedzy pojedyncza do wielokrotna interpolacja

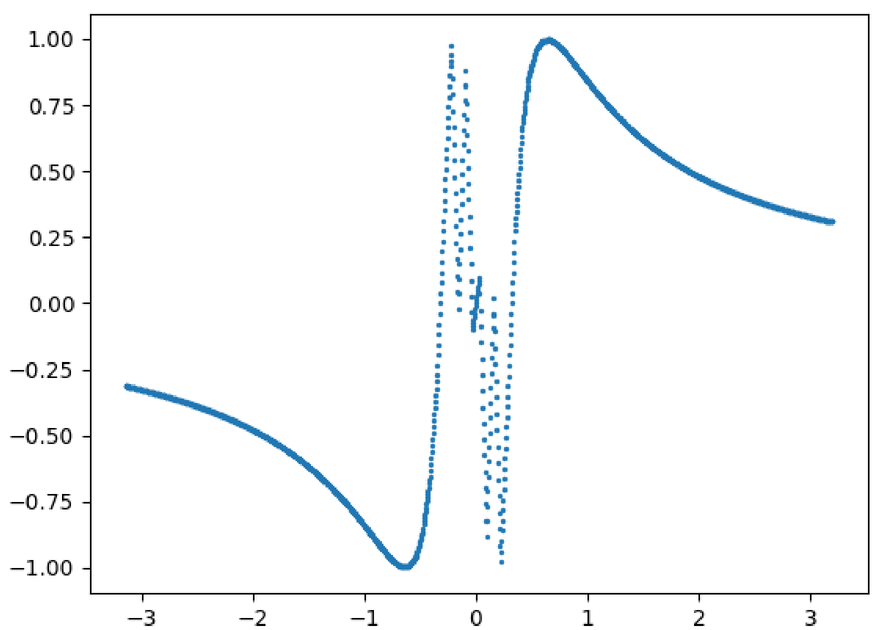
Do sprawdzenia wykorzystalem:

- Funkcje: $\sin(x^{-1})$
- Kernel: Triangular

Dla pierwszego wykresu uzylem interpolacji pojedynczej do skali 16



Dla wykresu ponizej zrobilem 4 interpolacje o skali 2 co daje $2^4 = 16$



Dla interpolacji pojedynczej:

$$\mathrm{MSE}_{\mathrm{single}} = 0.036252975643022475$$

Dla interpolacji wielokrotnej:

$$\mathrm{MSE}_{\mathrm{multi}} = 0.036252975643015654$$

Różnica pojawia się dopiero po 14 miejscu po przecinku!

Oznacza to, że istnieje subtelna, ale mierzalna różnica między interpolacją wielokrotną a pojedynczą.

2 Skalowanie Obrazów

Zadanie polegało na implementacji algorytmu skalowania obrazów cyfrowych:

2.1 Algorytm zmniejszania obrazu, za pomaca Jadra Usredniajacego

2.2 Operację powiększania obrazu za pomocą 2 interpolacji funkcji

2.3 Wykonanie obu operacji i ocenienie jakości algorytmów

2.1 Algorytm zmniejszania obrazu, za pomaca Jadra Usredniajacego

Rysunek 1: Obraz orginalny: 256x256px

Rysunek 2: Obraz pomniejszony 2 krotnie: 128x128px

Rysunek 3: Obraz pomniejszony 4 krotnie: 64x64px

2.2 Operacja powiększania obrazu za pomocą 2 interpolacji funkcji

Rysunek 4: Obraz pomniejszony 2 krotnie: 256x256px

Rysunek 5: Obraz powiekszony za pomoca kernela One-sided rectangular (point-holder)

Rysunek 6: Obraz powiekszony za pomoca kernela Centered rectangular

Rysunek 7: Obraz powiekszony za pomoca kernela Triangular (tent)

2.3 Ocena algorytmów

Do tego doświadczenia pomniejszylem obraz,
a potem go powiekszylem używając każdego z kernelów z zadania 1

Rysunek 8: Obraz zrekonstruowany za pomoca One-sided rectangular (point-holder)

Rysunek 9: Obraz zrekonstruowany za pomoca Centered rectangular

Rysunek 10: Obraz zrekonstruowany za pomoca Triangular (tent)

Do oceny tych algorytmow wykożystałem heat mapy MSE dla danego każdego kernela:

Rysunek 11: Heat mapa MSE dla One-sided rectangular (point-holder)

Rysunek 12: Heat mapa MSE dla Centered rectangular

Rysunek 13: Heat mapa MSE dla Triangular (tent)

2.4 Wnioski

Dzięki heat mapie jesteśmy wstanie zaobserwowac roznice w tych obrazach, gdzie w skali tej heat mapy tym blizej bieli tym wieksze MSE. Z czego wynika, ze najwiekszy blad jest dla kernela 1. Nastomiast roznica pomiedzy kernelem 2, i 3 nawet na heat mapie nie jest widoczna i sa znacznie blizej oreginalu niz kernel 1, wiec sa znacznie lepszymi kernelami