# Sprawozdanie
Obliczenia naukowe - Lista 4

Witold Karaś

## 1 zadanie 1

Zadanie polegało na implementacji funkcji obliczającej ilorazy różnicowe.

### 1.1 Iloraz różnicowy:

Jeśli $f : X \to Y$ oraz $x_0, x_1 \in X$ to ilorazem różnicowym nazywamy

$$f[x_0] = f(x_0)$$

$$f[x_0, x_1] = \frac{f(x_0) - f(x_1)}{x_0 - x_1}$$

Ponadto ilorazy różnicowe spełniają równość:

$$f[x_0, x_1, \ldots, x_n] = \frac{f[x_1, x_2, \ldots, x_{n-1}] - f[x_0, x_1, \ldots, x_n]}{x_n - x_0}$$

Warto zauważyć że jeśli $\Delta x = x_0 - x_1 \to 0$ to iloraz różnicowy $f[x_0, x_1]$ odpowiada wartości pochodnej pierwszego stopnia w punkcie $x_0$.

### 1.2 Opis algorytmu:

Algorytm na wejściu dostaje wektory długości $n + 1$ argumentów i wartości pewnej funkcji.
Wyjściem jest wektor ilorazów różnicowych, długości $n + 1$, w postaci:

$$[f[x_0], f[x_0, x_1], \ldots, f[x_0, x_1, \ldots, x_{n-1}], f[x_0, x_1, \ldots, x_n]]$$

Funkcja została oprogramowana bez użycia macierzy 2-wymiarowej. Aby to osiągnąć korzystamy z zagnieżdżonej pętli oraz z równości obliczania ilorazów różnicowych.

$$
\begin{array}{rrcr}
f[x_0] \searrow & & & f[x_0] \\
f[x_1] \begin{array}{c}\rightarrow\\ \searrow\end{array} & f[x_0, x_1] \searrow & & f[x_0, x_1] \\
\vdots & & \cdots & \vdots \\
f[x_{n-1}] \begin{array}{c}\rightarrow\\ \searrow\end{array} & f[x_{n-2}, x_{n-1}] \begin{array}{c}\rightarrow\\ \searrow\end{array} & \cdots \begin{array}{c}\rightarrow\\ \searrow\end{array} & f[x_0, x_1, \ldots, x_{n-1}] \\
f[x_n] \rightarrow & f[x_{n-1}, x_n] \rightarrow & \cdots \rightarrow & f[x_0, x_1, \ldots, x_n]
\end{array}
$$

# 2 zadanie 2

Zadanie polegało na implementacji funkcji obliczającej wartość wielomianu interolacyjnego stopnia $n$ w postaci Newtona w punkcie $x = t$ za pomocą uogólionego algorytmu Hornera w czasie liniowym.
Algorytm na wejściu dostaje wektory długości argumentów $n+1$ oraz wektor ilorazów różnicowych. Wyjściem jest wartość wielomianu w punkcie $x = t$

## 2.1 Opis algorytmu:

Wartość wielomianu interpolacyjnego Newtona w punkcie jest równa:

$$N_n(x) = f[x_0]+f[x_0,x_1](x-x_0)+\ldots+f[x_0,x_1,\ldots,x_n](x-x_0)(x-x_1)\ldots(x-x_{n-1})$$

można ją obliczać za pomocą uogólnionego schematu Hornera

$$\begin{aligned} w_n(x) &:= f[x_0,x_1,\ldots,x_n] \\ w_k(x) &:= f[x_0,x_1,\ldots,x_k] + (x-x_k)w_k(x)\forall(k=n-1,\ldots,0) \\ N_n(x) &:= w_0(x) \end{aligned}$$

Algorytm wykorzystuje jedną pętlę dlatego jego złożoność obliczeniowa wynosi $O(n)$.

# 3 zadanie 3

Zadanie polegało na implementacji funkcji obliczającej, w czasie kwadratowym, współczynniki wielomianu w postaci naturalnej $a_0, a_1, \ldots, a_n$.

Algorytm na wejściu dostaje wektor argumentów długości $n+1$ oraz wektor ilorazów różnicowych. Wyjściem jest wektor współczyników postaci naturalnej wielomianu, długości $n+1$, w postaci:

$$[a_0, a_1, \ldots, a_n]$$

## 3.1 Opis algorytmu:

Naszym celem jest zamiana wielomianu interpolacyjnego w postaci Newtona:

$$N_n(x) = f[x_0]+f[x_0,x_1](x-x_0)+\ldots+f[x_0,x_1,\ldots,x_n](x-x_0)(x-x_1)\ldots(x-x_{n-1})$$

do postaci normalnej:

$$p(x) = a_n x^n + a_{n-1}x^{n-1} + \ldots + a_1 x^1 + a_0$$

dla skrócenia zapisu będziemy posługiwać się notacją $c_0 = f[x_0], c_1 = f[x_0,x_1], \ldots, c_n = f[x_0,x_1,\ldots,x_n]$ wtedy:

$$N_n(x) = c_0 + c_1(x-x_0) + \ldots + c_n(x-x_0)(x-x_1)\ldots(x-x_{n-1})$$

Zauważmy że czynnik przy $a_n$ znajdujący się przy $x^n$ jest równy $c_n$. Idea algorytmu polega na wyznaczeniu początkowej wartości współczynnika przy aktualnie rozważanej potędze $x^k$ ($a_k = c_k - x_k a_{i+1}$). Następnie zaktualizujemy wartości $a_i$ dla $k < i < n$ w taki sposób że $\forall_{i\in(k+1,n-1)} a_i = a_i - x_i a_{i+1}$. Po kolejnych potęgach będziemy poruszać się iteracyjnie "w dół" ponieważ znamy wartość współczynnika stojącego przy $x^n$

## 3.2 Złożoność obliczeniowa algorytmu:

Algorytm wykorzystuje dwie zagnieżdżone pętle, z czego zewnętrzna wykona się $n$ razy, a wewnętrzna maksymalnie $n-1$ razy. Daje nam to złożoność obliczeniową wynoszącą $O(n^2)$

# 4 zadanie 4

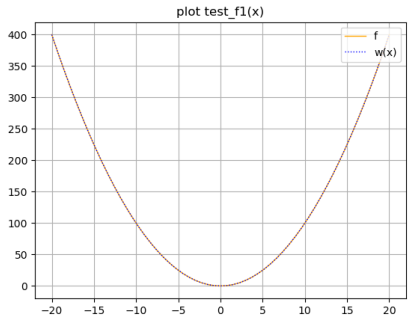
Zadanie polegało na napisanui funkcji która zinterpoluje zadaną funkcję $f(x)$ na zadanym przedziale $[a, b]$ za pomocą wielomianu interpolacyjnego stopnia n w postaci Newton oraz narysuje wielomian interpolacyjny i interpolowaną funkcję. Do rysowania użyłem pakiety **PyPlot**.

Algorytm na wejściu dostaje funkcję $f(x)$ zadaną jako anonimowa, $a, b$ - krańce przedziału oraz $n \in \mathbb{N}$ - stopień wielomianu interpolacyjnego. Wynikiem działania funkcji jest obrazek zapisany do pliku na dysku.

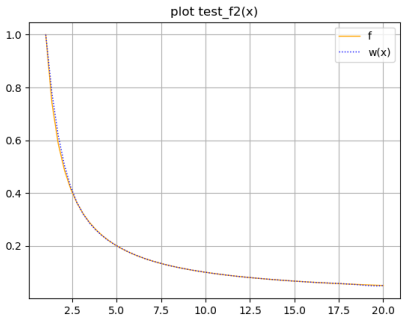
## 4.1 Opis działania:

Pierwszym krokiem jest podzielenie przedziału $[a, b]$ na $n$ równych podprzedziałów. Końce przedziału zapisujemy do tablicy argumentów $x$, a wartości funkcji w tych miejscach do tablicy wartości funkcji $y$. Następnie obliczamy ilorazy różnicowe dla uprzednio obliczonych argumentów i wartości $yN$.
Kolejnym krokiem jest zwiększenie 'rozdzielczości', wykonujemy to przez przemnożenie stopnia wielomian przez stałą $factor$. Tak jak poprzednio obliczamy nowe argumenty oraz wartości funkcji jednak dodatkowo na podstawie poprzednio policzonych argumentów i wartości korzystamy z funkcji **warNewton** aby policzyć wierzchołki wielomianu interpolacyjnego.
końcowym etapem jest wykorzystanie pakietu **PyPlot** do naniesienia danych na wykresy oraz zapisanie obrazków.

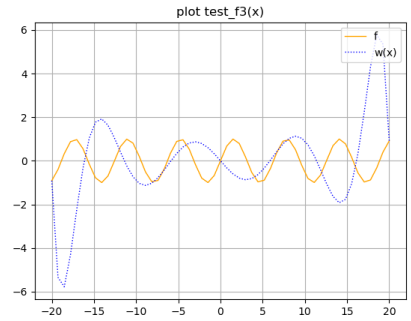
## 4.2 Dla przykładowych funkcji:



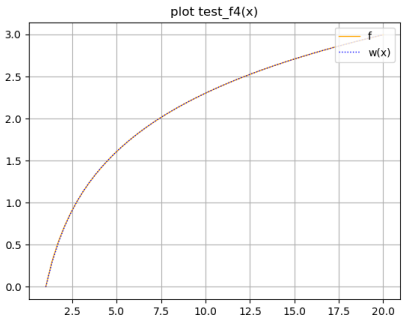

(a) $f(x) = x^2, n = 10$




(b) $f(x) = 1/x, n = 10$




(c) $f(x) = \sin(x), n = 10$



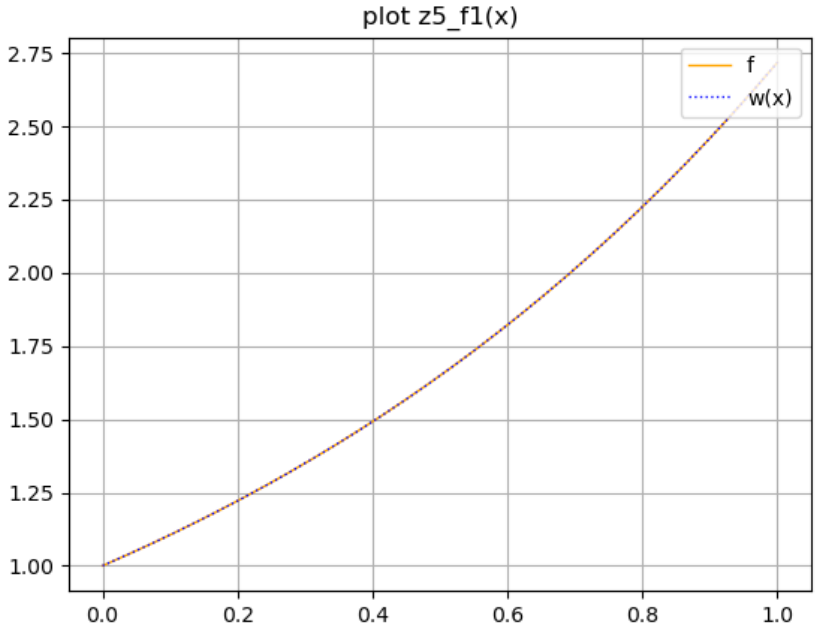

(d) $f(x) = \log(x), n = 10$

# 5 zadanie 5

Zadanie polegało na przetestowaniu funkcji /(rysujNnfx/) przygotowanej na potrzeby wcześniejszego zadania. Testy zostały wykonane na następujących danych wejściowych:

1. $f(x) = e^x; [a, b] = [0, 1], n = 5, 10, 15$
2. $f(x) = x^2 \sin(x); [a, b] = [-1, 1], n = 5, 10, 15$

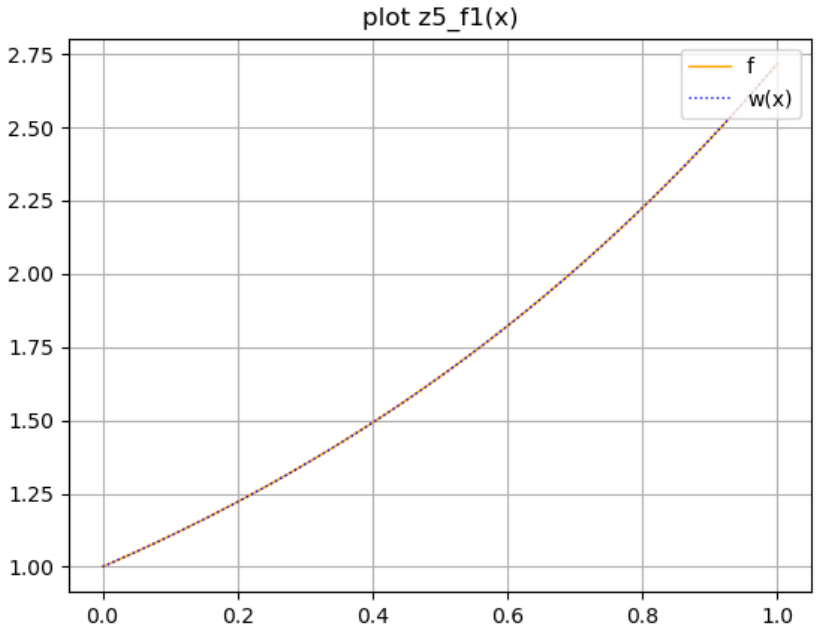
## 5.1 Wyniki:



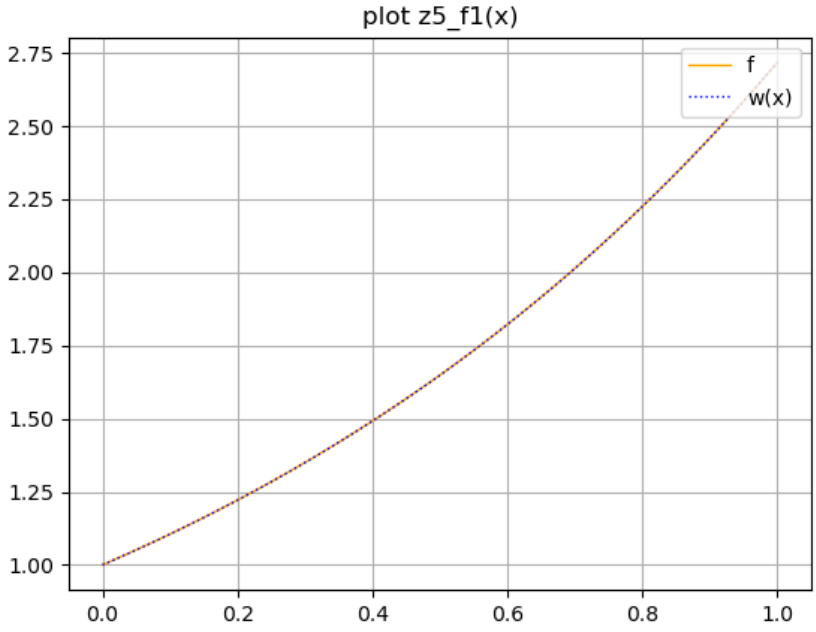

Rysunek 2: Wykresy funkcji $f(x) = e^x, n = 5$

## 5.2 Wnioski:

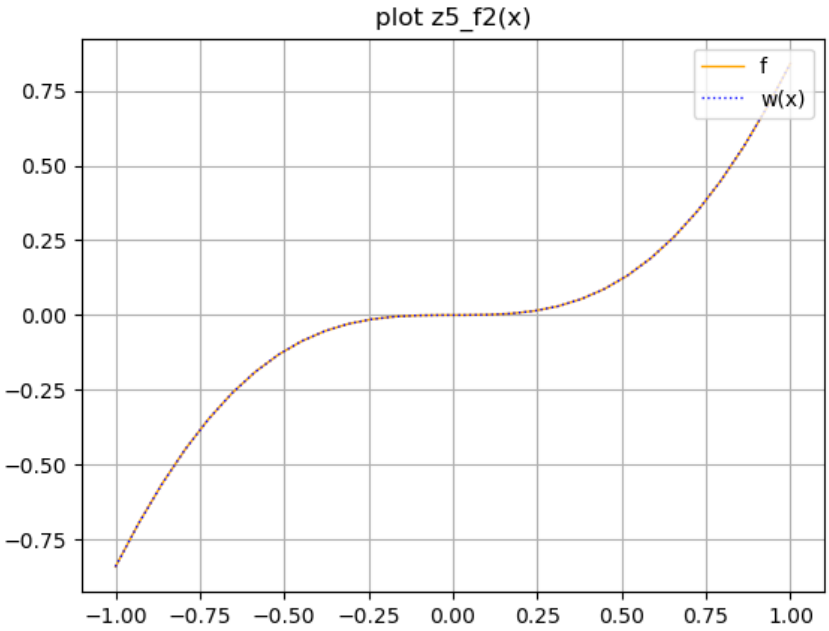
Funkcje wydają się być 'porządne' tj. ich wielomianowe przybliżenie nie odbiega znacząco od spodziewanych rezultatów na zadanych przedziałach.



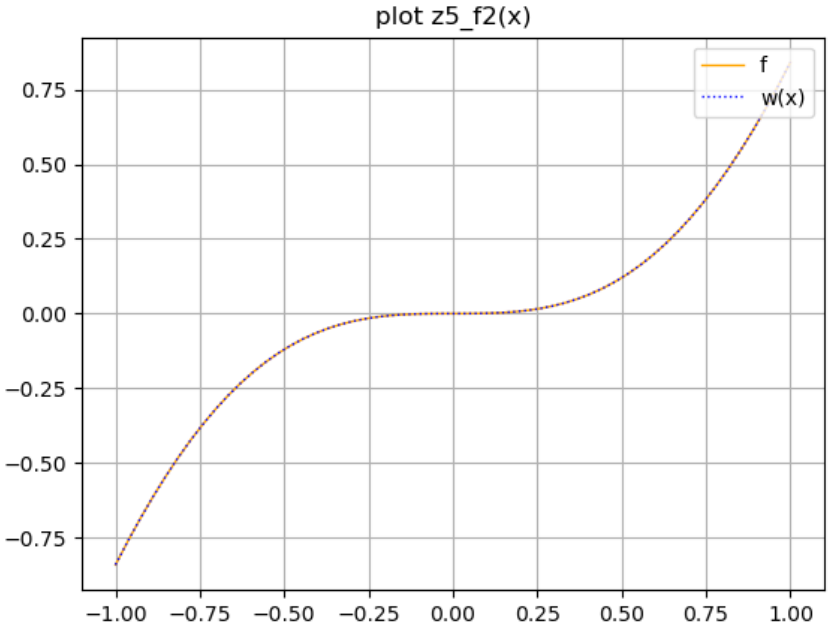

Rysunek 3: Wykresy funkcji $f(x) = e^x, n = 10$



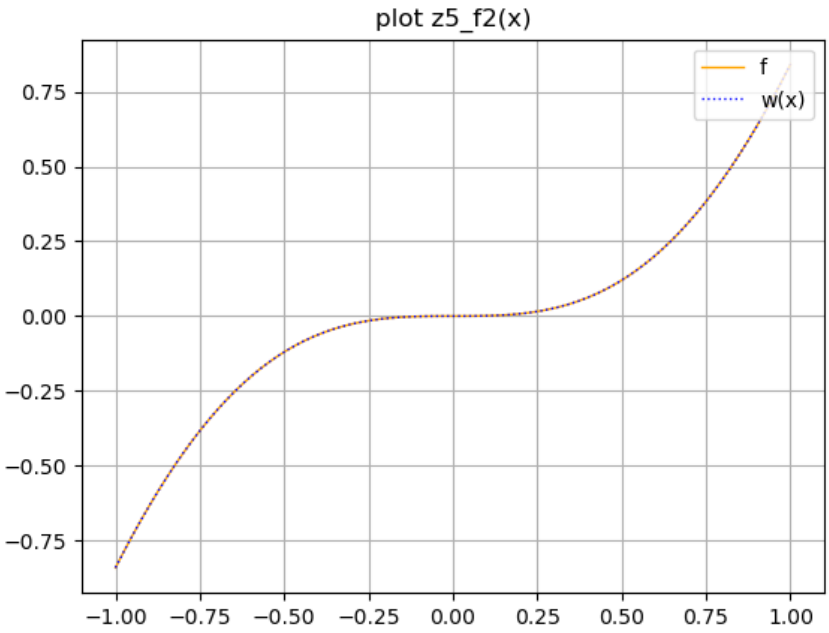

Rysunek 4: Wykresy funkcji $f(x) = e^x, n = 15$



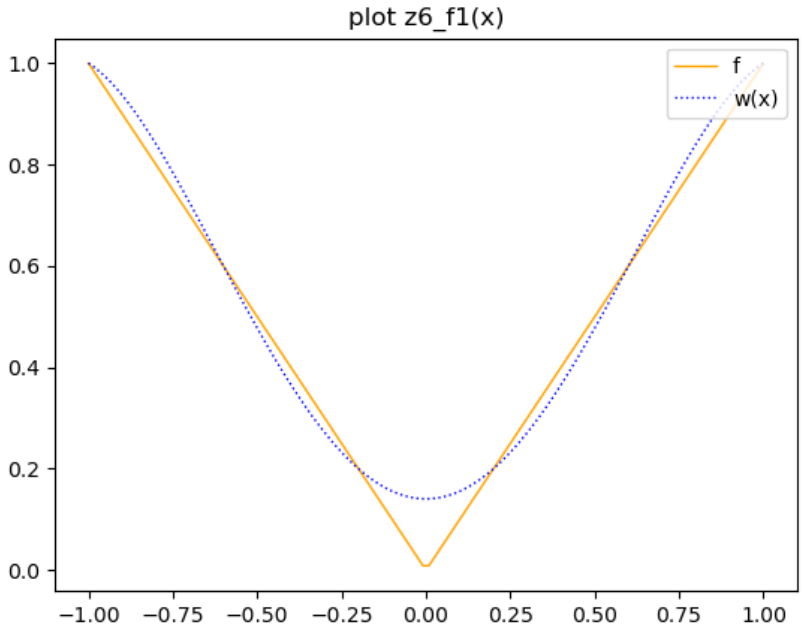

Rysunek 5: Wykresy funkcji $f(x) = x^2 sin(x), n = 5$




Rysunek 6: Wykresy funkcji $f(x) = x^2 sin(x), n = 10$




Rysunek 7: Wykresy funkcji $f(x) = x^2 sin(x), n = 15$




Rysunek 8: Wykres funkcji $f(x) = |x|, n = 5$

# 6 zadanie 6

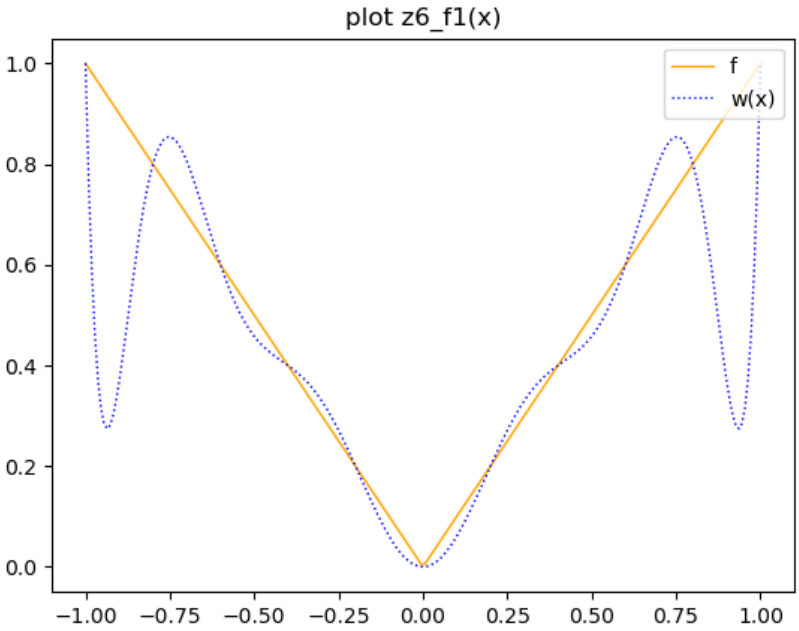
## 6.1 Wyniki:

Zadanie polegało na przetestowaniu funkcji /(rysujNnfx/) przygotowanej na potrzeby wcześniejszego zadania. Testy zostały wykonane na następujących danych wejściowych:

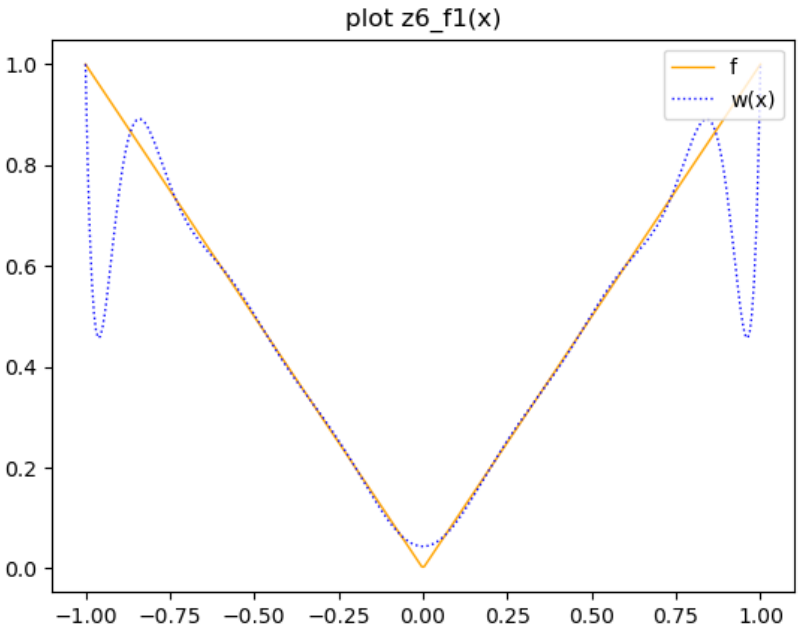
1. $f(x) = |x|; [a, b] = [-1, 1], n = 5, 10, 15$
2. $f(x) = \frac{1}{1+x^2}; [a, b] = [-5, 5], n = 5, 10, 15$

## 6.2 Wnioski:

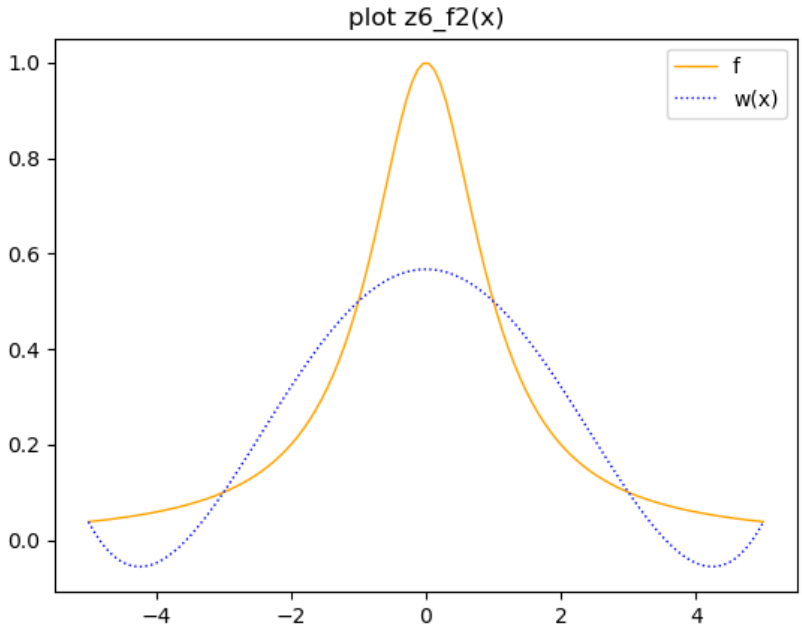
Dla testowanych funkcji przybliżenia niezbyt wiernie oddają spodziewane kształty. W tych przypadkach możemy zaobserwować efekt Rungego. Jest to zjawisko polegające na pogorszeniu się jakości interpolacji mimo zwiększenia liczby jej węzłów co jest efektem odwrotnym do zamierzonego. Efekt jest wywołany przez równomierne rozłożenie węzłów interpolacji. Można go uniknąć kładąc więcej węzłów interpolacji na końcach przedziału.



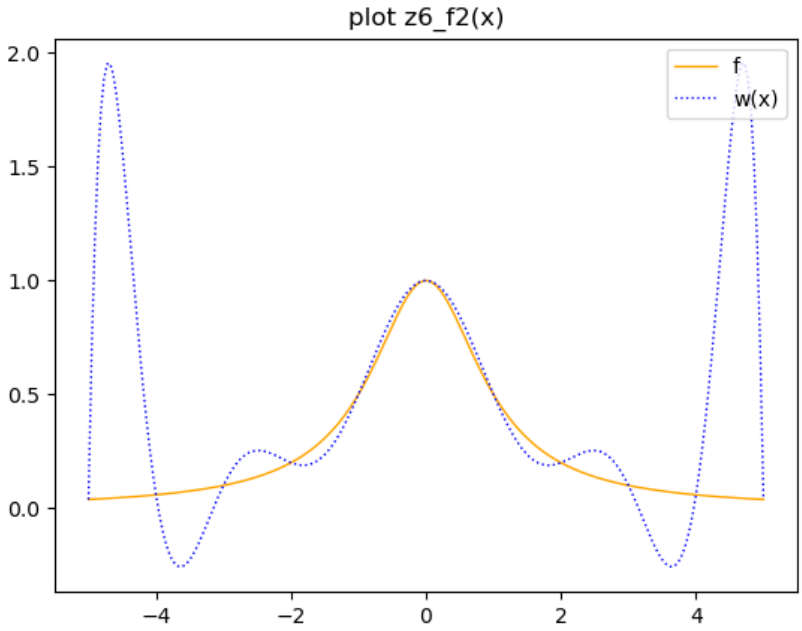

Rysunek 9: Wykres funkcji $f(x) = |x|, n = 10$



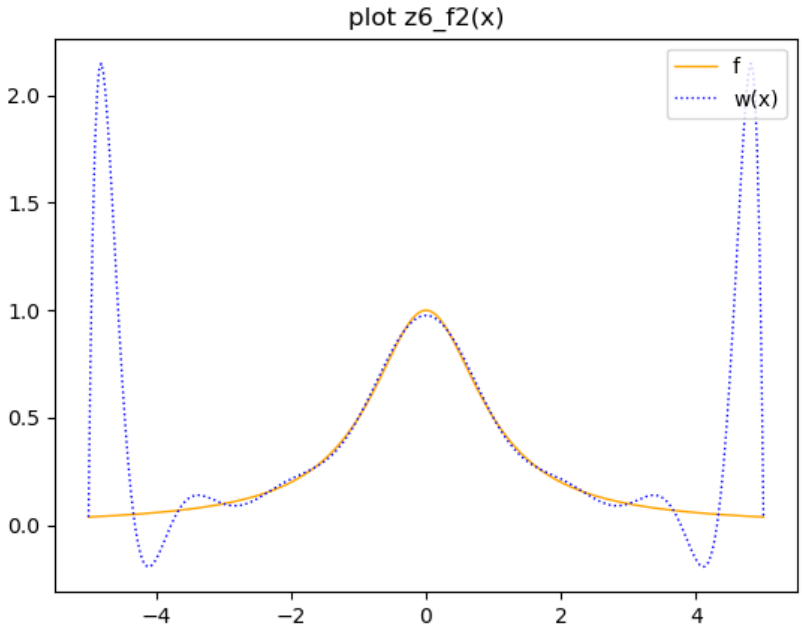

Rysunek 10: Wykres funkcji $f(x) = |x|, n = 15$




Rysunek 11: Wykres funkcji $f(x) = \frac{1}{1+x^2}, n = 5$




Rysunek 12: Wykres funkcji $f(x) = \frac{1}{1+x^2}, n = 10$




Rysunek 13: Wykres funkcji $f(x) = \frac{1}{1+x^2}, n = 15$