Obliczenia naukowe

Lista 4

Aleksandra Czarniecka (272385)

grudzień 2024

Wstęp: Przedstawienie problemu interpolacji

Interpolacja jest kluczowym zagadnieniem w analizie numerycznej, które polega na przybliżeniu funkcji za pomocą wielomianów, bazując na wartościach funkcji w wybranych punktach, zwanych węzłami.

Istnieje $n+1$ par $(x_i, y_i = f(x_i))$, gdzie $\forall_{i,j} x_i \neq x_j$ i $i \in \{0, \ldots, n\}$. Problem interpolacji polega na znalezieniu wielomianu $p_n(x)$ stopnia co najwyżej n, który przechodzi przez zadane węzły interpolacyjne (x_i, y_i), gdzie x_i są parami różne, a $y_i = f(x_i)$. Z teorii wiadomo, że taki wielomian istnieje i jest dokładnie jeden.

Naturalna postać wielomianu

Wielomian $p_n(x)$ można zapisać w postaci naturalnej:

$$p_n(x) = a_0 + a_1 x + a_2 x^2 + \ldots + a_n x^n,$$

gdzie współczynniki $a_0, a_1, \ldots, a_n$ wyznacza się, rozwiązując układ równań z macierzą Vandermonde'a. Niestety, taka metoda jest numerycznie niestabilna, dlatego korzysta się z innych reprezentacji wielomianu.

Postać Newtona

Aby uniknąć problemów numerycznych, zapisujemy wielomian w postaci Newtona, korzystając z bazy wielomianów:

$$q_0(x) = 1, \quad q_1(x) = (x - x_0), \quad q_2(x) = (x - x_0)(x - x_1), \quad \ldots, \quad q_n(x) = \prod_{j=0}^{n-1} (x - x_j).$$

Wówczas wielomian $p_n(x)$ można wyrazić jako:

$$p_n(x) = \sum_{i=0}^{n} c_i \, q_i(x),$$

gdzie współczynniki c_i wyznacza się, korzystając z tzw. ilorazów różnicowych, czym zajmiemy się w zadaniu 1. Następnie w zadaniu 2 wyliczymy wartość wielomianu interpolacyjnego w danym punkcie, a w zadaniu 3 przekształcimy ten wielomian do postaci normalnej odkrywając jego stopień. Ostatecznie w zadaniu 5 przedstawiony zostanie wykres, na którym umieszczona zostanie wielomian interpolacyjny oraz funkcja, którą interpoluje.

1 Ilorazy różnicowe

1.1 Opis problemu

Problemem zadania jest napisanie funkcji obliczającej ilorazy różnicowe bez użycia tablicy dwuwymiarowej.

`function ilorazyRoznicowe (x::Vector{Float64}, f::Vector{Float64})`
Dane:

`x` - wektor długości $n+1$ zawierający węzły $x_0, \ldots, x_n$

`x[1]` = x_0, ..., `x[n+1]` = x_n

`f` - wektor długości $n+1$ zawierający wartości interpolowanej funkcji w węzłach $f(x_0), \ldots, f(x_n)$

Wyniki:

`fx` - wektor długości $n+1$ zawierający obliczone ilorazy różnicowe

`fx[1]` = $f[x_0]$,
`fx[2]` = $f[x_0, x_1]$, ..., `fx[n]` = $f[x_0, \ldots, x_{n-1}]$, `fx[n+1]` = $f[x_0, \ldots, x_n]$.

1.2 Rozwiązanie

Ilorazy różnicowe są podstawą interpolacji Newtona w postaci ilorazowej. Rozwiązanie tego problemu polega na zaimplementowaniu algorytmu, który oblicza ilorazy różnicowe mając dane pary $(x_i, y_i = f(x_i))$. Wiadomo, że:

$$f[x_i] = f(x_i)$$

Pozostałe ilorazy różnicowe spełniają równość:

$$f[x_i, x_{i+1}, \ldots, x_k] = \frac{f[x_{i+1}, x_{i+2}, \ldots, x_k] - f[x_i, x_{i+1}, \ldots, x_{k-1}]}{x_k - x_i}$$

.
Chcąc uniknąć tworzenia macierzy dwuwymiarowej można zauważyć, że po wyliczeniu wszystkich ilorazów różnicowych zależnych od k węzłów, to wszystkie poza $f[x_0, x_1, \ldots, x_k]$ stają się dla nas zbędne. Dzięki tej obserwacji algorytm można wykonać na jednej tablicy $n+1$ - elementowej.

Algorytm działa iteracyjnie, gdzie w każdym kroku oblicza kolejne ilorazy różnicowe. Dla każdego poziomu, różnice są obliczane od końca wektora `c`, co umożliwia bezpośrednią modyfikację danych. Na początku do tablicy zapisać wartości funkcji w węzłach interpolacji, ponieważ $f[x_i] = f(x_i)$. Następnie elementy, które nie są już potrzebne w kolejnych iteracjach, nadpisywane są nowymi wynikami, a wynikowe ilorazy różnicowe są tworzone w miejscu, co redukuje wymagania pamięciowe.

Metoda ma złożoność obliczeniową $O(n^2)$ (dla każdego rzędu k wykonujemy $n-k$ obliczeń) i złożoność pamięciową $O(n)$ (tablica jednowymiarowa).

Algorithm 1 Funkcja obliczająca ilorazy różnicowe

Input: $\overline{x}$ – wektor węzłów, $\overline{f}$ – wektor wartości funkcji w punktach z $\overline{x}$
Output: $\overline{c}$ – wektor ilorazów różnicowych $f[x_0, ..., x_i]$

```
c̄ ← f̄;
for j from 1 to n do
    for i from n down to j do
        c_i ← (c_i − c_{i−1}) / (x_i − x_{i−j})
    end
end
return c̄
```

1.3 Testy

Wszystkie przeprowadzone testy przechodzą, zatem implementacja jest poprawna.

2 Wartość wielomianu interpolacyjny w punkcie- algorytm Hornera

2.1 Opis problemu

Problemem zadania jest napisanie funkcji obliczającej wartość wielomianu interpolacyjnego stopnia n w postaci Newtona $N_n(x)$ w punkcie $x = t$ za pomocą uogólnionego algorytmu Hornera, w czasie $O(n)$.

`function warNewton (x::Vector{Float64}, fx::Vector{Float64}, t::Float64)`
Dane:

`x` - wektor długości $n+1$ zawierający węzły $x_0, \ldots, x_n$

`x[1]` $= x_0$, ..., `x[n+1]` $= x_n$

`fx` - wektor długości $n+1$ zawierający ilorazy różnicowe

`fx[1]` $= f[x_0]$

`fx[2]` $= f[x_0, x_1]$, ..., `fx[n]` $= f[x_0, \ldots, x_{n-1}]$, `fx[n+1]` $= f[x_0, \ldots, x_n]$

`t` - punkt, w którym należy obliczyć wartość wielomianu

Wyniki:

`nt` - wartość wielomianu w punkcie t

2.2 Rozwiązanie

Rozwiązanie tego problemu polega na zaimplementowaniu algorytmu, który oblicza wartości wielomianu interpolacyjnego stopnia n w postaci Newtona w zadanym punkcie t w czasie $O(n)$. Licząc w sposób bezpośrednio ze wzoru

$$p_n(x) = \sum_{i=0}^{n} c_i \, q_i(x),$$

uzyskamy złożoność $O(n^2)$.

Przekształcając wielomian na postać rekurencyjną, wykonujemy:

$$p_n(x) = \sum_{k=0}^{n} c_k \cdot q_k(x) = f[x_0] + \sum_{k=1}^{n} f[x_0, \ldots, x_k](x - x_0) \ldots (x - x_k) =$$

$$= f[x_0] + (x - x_0)(f[x_0, x_1] + \sum_{k=2}^{n} f[x_0, \ldots, x_k](x - x_1) \ldots (x - x_k)) =$$

$$\vdots$$

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

. Rekurencyjnie otrzymujemy:

$$w_n(x) = f[x0, \ldots, x_n]$$

$$w_k(x) = f[x_0, \ldots, x_k] + (x - x_k) w_{k+1}, \text{ gdzie } k \in \{0, \ldots, n-1\}$$

. Wówczas:

$$p_n(x) = w_0(x)$$

.

Algorithm 2 Funkcja obliczająca wartość wielomianu interpolacyjnego w danym punkcie

Input: $\overline{x}$ – wektor punktów, $\overline{c}$ – wektor ilorazów różnicowych, t – punkt, dla którego szukamy wartości wielomianu
Output: nt – wartość wielomianu w punkcie t

```
8  nt ← c_n;
9  for i from n − 1 down to 0 do
10 |  nt ← c_i + (t − x_i) · nt
11 end
12 return nt
```

2.3 Testy

Wszystkie przeprowadzone testy przechodzą, zatem implementacja jest poprawna.

3 Obliczanie współczynników postaci naturalnej wielomianu interpolacyjnego

3.1 Opis problemu

Problemem zadania jest, znając współczynniki wielomianu interpolacyjnego w postaci Newtona $c_0 = f[x_0], c_1 = f[x_0, x_1], \ldots, c_n = f[x_0, x_1, \ldots, x_n]$ (ilorazy różnicowe) oraz węzły $x_0, x_1, \ldots, x_n$ napisać funkcję obliczającą, w czasie $O(n^2)$ współczynniki jego postaci naturalnej $a_0, a_1, \ldots, a_n$ tzn. $a_n x^n + a_{n-1} x^{n-1} + \cdots + a_1 x + a_0$.

`function naturalna (x::Vector{Float64}, fx::Vector{Float64}`
Dane:

`x` - wektor długości $n+1$ zawierający węzły $x_0, \ldots, x_n$

`x[1]` $= x_0$, ..., `x[n+1]` $= x_n$

`f` - wektor długości $n+1$ zawierający ilorazy różnicowe

`fx[1]` $= f[x_0]$
`fx[2]` $= f[x_0, x_1]$, ..., `fx[n]` $= f[x_0, \ldots, x_{n-1}]$, `fx[n+1]` $= f[x_0, \ldots, x_n]$

Wyniki:

`a` - wektor długości $n+1$ zawierający obliczone współczynniki postaci natrualnej

`a[1]` $= a_0$
`a[2]` $= a_1$, ..., `a[n]` $= a_{n-1}$, `a[n+1]` $= a_n$

3.2 Rozwiązanie

Rozwiązaniem problemu jest zaimplementowanie algorytmu sprowadzającego wielomian w postaci Newtona:

$$\sum_{k=0}^{n} c_k \prod_{j=0}^{k-1} (x - x_j),$$

do wielomianu w postaci naturalnej:

$$\sum_{i=0}^{n} a_i x^i$$

w czasie $O(n^2)$. W ten sposób otrzymamy współczynniki a_i każdej kolejnej potęgi x^i.

Algorytm przechodzi przez dwie pętle (złożoność obliczeniowa $O(n^2)$) zaczynając od $w_n(x)$ i obliczając kolejne wartości częściowe tak jak w algorytmie Hornera, tylko zapisując je w postaci naturalnej. Wówczas obliczone już wartości można użyć do obliczania wartości kolejnych współczynników. Użyta jest tutaj rekurencja z poprzedniego zadania.

$$w_n = c_n$$

Przy x^n współczynnik będzie wynosił c_n.

$$w_{n-1} = c_{n-1} + (x - x_{n-1}) w_n = c_{n-1} + (x - x_{n-1}) c_n = c_{n-1} + c_n x - c_n x_{n-1}$$

Przy x^{n-1} współczynnik będzie wynosił $c_{n-1} - x_{n-1} c_n$. .

Algorithm 3 Funkcja obliczająca współczynniki postaci naturalnej wielomianu interpolacyjnego

Input: $\overline{x}$ – wektor punktów, $\overline{c}$ – wektor ilorazów różnicowych
Output: $\overline{a}$ – wektor współczynników postaci naturalnej wielomianu interpolacyjnego

$a_n \leftarrow c_n$;
for i ***from*** $n-1$ ***down to*** 0 **do**
  $a_i \leftarrow c_i - x_i \cdot a_{i+1}$ **for** j ***from*** $i+1$ ***to*** $n-1$ **do**
    $a_j \leftarrow a_j - x_i \cdot a_{j+1}$
  end
end
return $\overline{a}$

3.3 Testy

Wszystkie przeprowadzone testy przechodzą, zatem implementacja jest poprawna.

4 Graficzne przedstawienie wielomianu interpolacyjnego i interpolowanej funkcji

4.1 Opis problemu

Problemem zadania jest napisanie funkcji, która zinterpoluje zadaną funkcję $f(x)$ w przedziale $[a, b]$ za pomocą wielomianu interpolacyjnego stopnia n w postaci Newtona. Następnie narysuje wielomian interpolacyjny i interpolowaną funkcję. Do rysowania użyty został pakiet Plots. W interpolacji należało użyć węzłów równoodległych, tj. $x_k = a + kh, h = (ba)/n, k = 0, 1, ..., n$.

`function rysujNnfx (f, a::Float64, b::Float64, n::Int`
Dane:

`f` - funkcja $f(x)$ zadana jako anonimowa funkcja,

`a, b` - przedział interpolacji,

`n` - stopień wielomianu interpolacyjnego.

Wyniki:

- funkcja rysuje wielomian interpolacyjny i interpolowaną funkcję w przedziale $[a, b]$

4.2 Rozwiązanie

Rozwiązaniem problemu jest połączenie zaimplementowanych wcześniej metod w jedną, umożliwiającą graficzne porównanie otrzymanego wielomianu interpolacyjnego z dokładną funkcją.

Algorithm 4 Funkcja przedstawiająca graficznie wykresy funkcji interpolowanej i wielomianu interpolującego

Input: f – funkcja, którą chcemy interpolować, a, b – końce przedziału interpolacji, n – stopień wielomianu interpolacyjnego

Output: wykres wielomianu interpolującego i funkcji interpolowanej

1: Wyznacz $n+1$ równoodległych punktów x, gdzie $x_i = a + i \cdot \frac{b-a}{n}$, $i = 0, \ldots, n$
2: Oblicz wartości funkcji w tych punktach: $fx[i] = f(x_i)$
3: Oblicz ilorazy różnicowe: $\overline{c} \leftarrow$ `ilorazyRoznicowe`$(\overline{x}, \overline{y})$
4: Stwórz gęstą siatkę punktów xs w przedziale $[a, b]$, np. length(xs) $= 1000$
5: Oblicz:

- $y_{\text{original}} \leftarrow [f(t)$ dla $t \in$ xs$]$
- $y_{\text{interpolated}} \leftarrow [$`warNewton`$(x, c, t)$ dla $t \in$ xs$]$

6: Utwórz wykres:

- niebieska linia jako funkcja oryginalna: $(\text{xs}, y_{\text{original}})$
- czerwona linia jako wielomian interpolacyjny: $(\text{xs}, y_{\text{interpolated}})$

7: Zapisz wykres do pliku: `savefig(filename)`

4.3 Testy

Wszystkie przeprowadzone testy przechodzą, zatem implementacja jest poprawna.

5 Analiza wykresów funkcji i wielomianu interpolacyjnego

5.1 Opis problemu

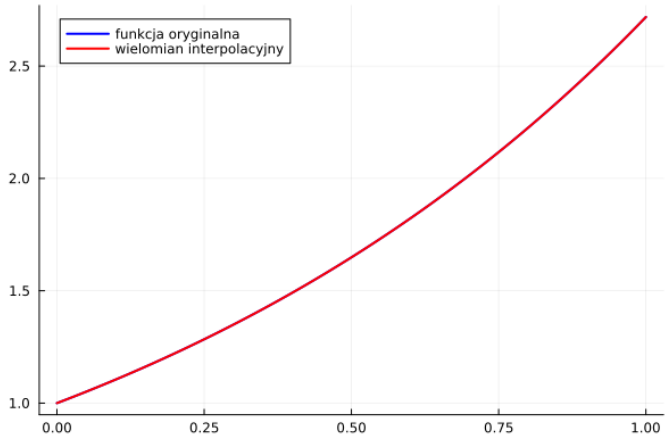
Problemem zadania jest przetestowanie funkcji `rysujNnfx(f, a, b, n)` na nastepujących przykładach:

1. $e^x, [0, 1], n \in \{5, 10, 15\}$
2. $x^2 \sin x, [-1, 1], n \in \{5, 10, 15\}$

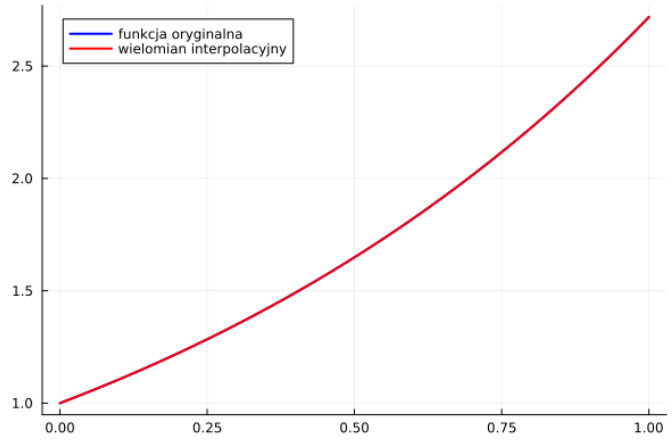
5.2 Rozwiązanie

Rozwiązaniem problemu jest użycie metody `rysujNnfx(f, a, b, n)` dla odpowiednich danych. Wykresy wygenerowane przez program są umieszczone poniżej.

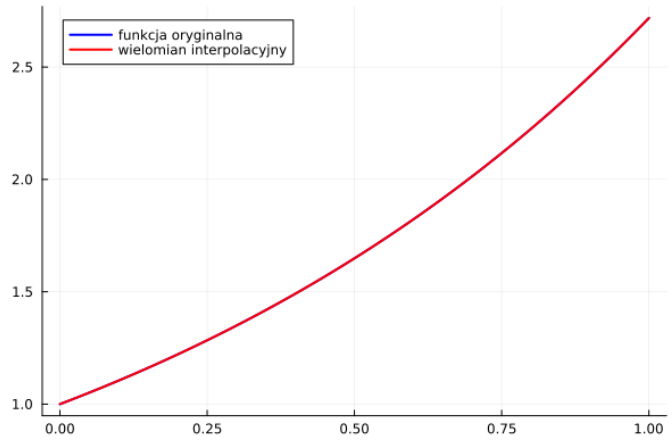
5.3 Wyniki i ich interpretacja



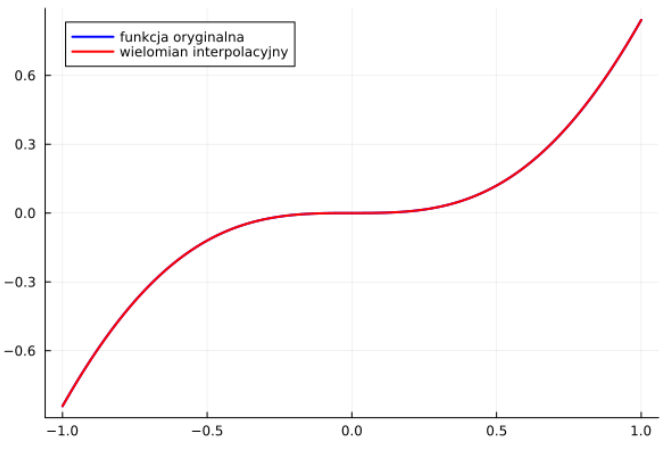
Rysunek 1: Wykres wielomianu interpolacyjnego stopnia ≤ 5 i funkcji interpolowanej $f(x) = e^x$ na przedziale $[0, 1]$.



Rysunek 2: Wykres wielomianu interpolacyjnego stopnia ≤ 10 i funkcji interpolowanej $f(x) = e^x$ na przedziale $[0, 1]$.

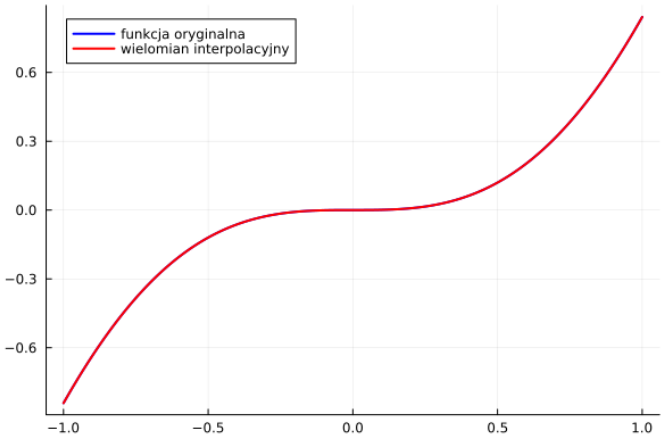


Rysunek 3: Wykres wielomianu interpolacyjnego stopnia ≤ 15 i funkcji interpolowanej $f(x) = e^x$ na przedziale $[0, 1]$.

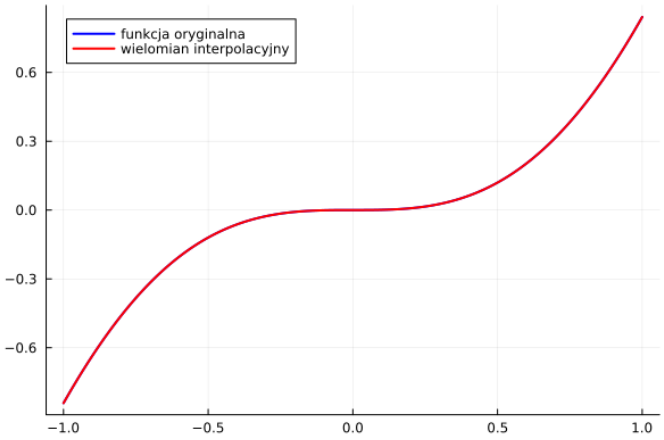


.png

Rysunek 4: Wykres wielomianu interpolacyjnego stopnia ≤ 5 i funkcji interpolowanej $f(x) = x^2 \sin x$ na przedziale $[-1, 1]$.



Rysunek 5: Wykres wielomianu interpolacyjnego stopnia ≤ 10 i funkcji interpolowanej $f(x) = x^2 \sin x$ na przedziale $[-1, 1]$.



Rysunek 6: Wykres wielomianu interpolacyjnego stopnia ≤ 15 i funkcji interpolowanej $f(x) = x^2 \sin x$ na przedziale $[-1, 1]$.

Na wszystkich wykresach funkcja oryginalna pokrywa się z wielomianem interpolacyjnym, niezależnie od stopnia wielomianu.

5.4 Wnioski

Dla funkcji $f(x) = e^x$ i $f(x) = x^2 sinx$ interpolacja wielomianem sprawdza się bardzo dobrze. Nawet dla wielomianów niskiego stopnia nie ma widocznych różnic. Funkcje te są łatwo interpolowane, a algorytm działa poprawnie.

6 Analiza wykresów funkcji i wielomianu interpolacyjnego

6.1 Opis problemu

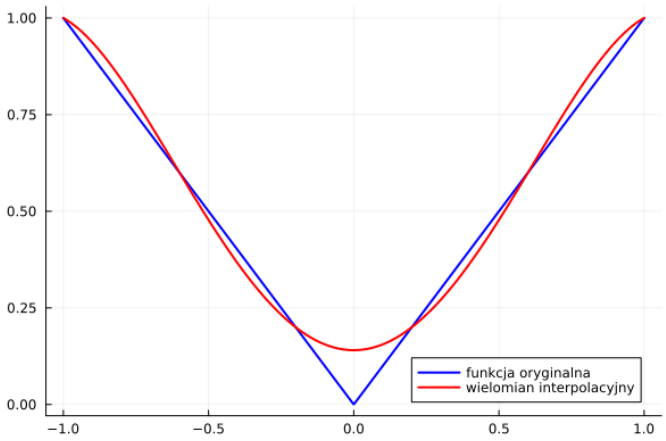
Problemem zadania jest przetestowanie funkcji `rysujNnfx(f, a, b, n)` na nastepujących przykładach:

1. $|x|, [-1, 1], n \in \{5, 10, 15\}$
2. $\frac{1}{1+x^2}, [-5, 5], n \in \{5, 10, 15\}$

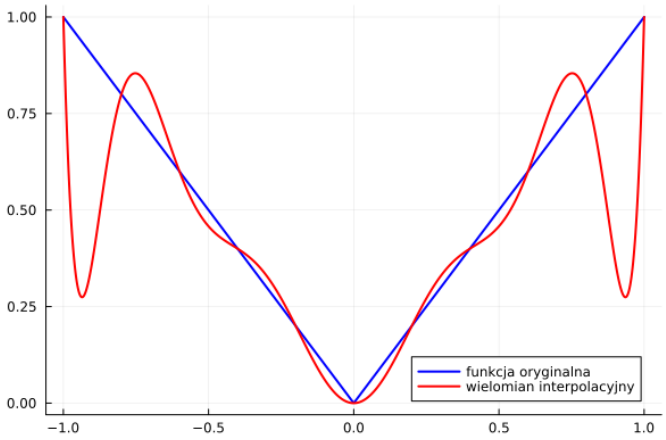
6.2 Rozwiązanie

Rozwiązaniem problemu jest użycie metody `rysujNnfx(f, a, b, n)` dla odpowiednich danych. Wykresy wygenerowane przez program są umieszczone poniżej.

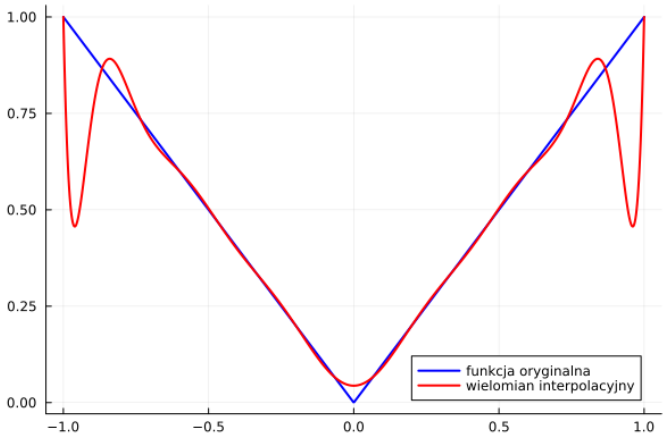
6.3 Wyniki i ich interpretacja



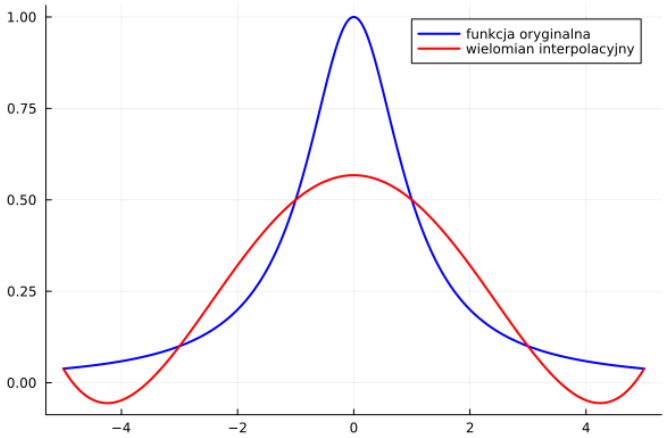
Rysunek 7: Wykres wielomianu interpolacyjnego stopnia ≤ 5 i funkcji interpolowanej $f(x) = |x|$ na przedziale $[-1, 1]$.



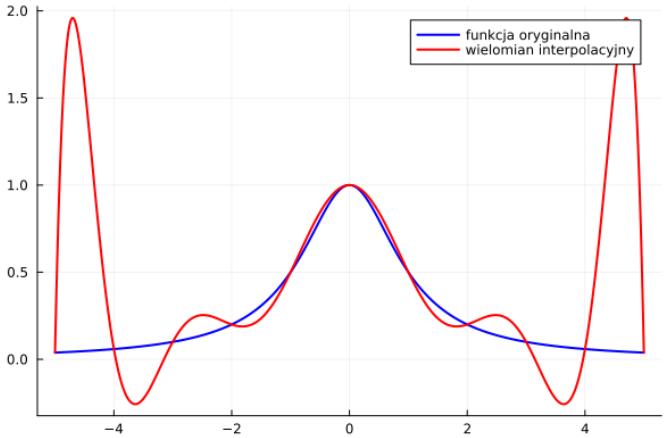
Rysunek 8: Wykres wielomianu interpolacyjnego stopnia ≤ 10 i funkcji interpolowanej $f(x) = |x|$ na przedziale $[-1, 1]$.



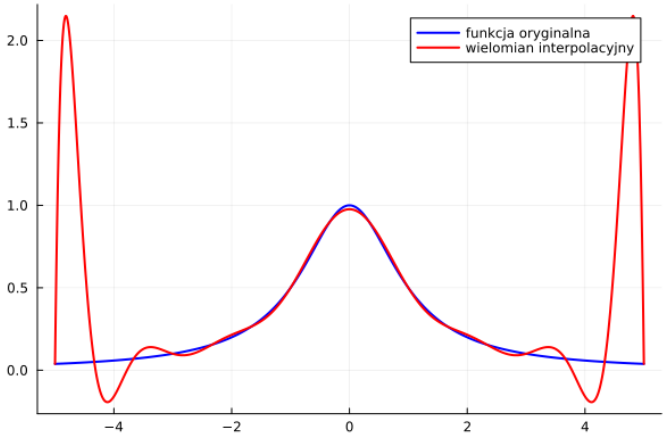
Rysunek 9: Wykres wielomianu interpolacyjnego stopnia ≤ 15 i funkcji interpolowanej $f(x) = |x|$ na przedziale $[-1, 1]$.



Rysunek 10: Wykres wielomianu interpolacyjnego stopnia ≤ 5 i funkcji interpolowanej $f(x) = \frac{1}{1+x^2}$ na przedziale $[-5, 5]$.



Rysunek 11: Wykres wielomianu interpolacyjnego stopnia ≤ 10 i funkcji interpolowanej $f(x) = \frac{1}{1+x^2}$ na przedziale $[-5, 5]$.



Rysunek 12: Wykres wielomianu interpolacyjnego stopnia ≤ 15 i funkcji interpolowanej $f(x) = \frac{1}{1+x^2}$ na przedziale $[-5, 5]$.

Interpolacja wielomianowa w przypadku funkcji $|x|$ oraz $\frac{1}{1+x^2}$ nie daje zadowalających wyników. Nie poprawia ich nawet zwiększenie stopnia wielomianu.

Dla funkcji $|x|$ spowodowane jest to tym, że nie jest ona różniczkowalna w punkcie $x = 0$. Wielomiany, jako funkcje ciągłe i gładkie, mają ograniczone możliwości przybliżenia funkcji o charakterystycznym, ostrym wierzchołku, takim jak w $|x|$.

Natomiast dla funkcji $\frac{1}{1+x^2}$, interpolacja wielomianowa prowadzi do coraz większych odchyleń od rzeczywistego przebiegu funkcji w okolicach końców przedziału. To zjawisko, znane jako efekt Runge'go, szczególnie wyraźnie występuje przy stosowaniu równoodległych węzłów, jak w tym przypadku. Rozwiązaniem mogłoby być użycie węzłów rozmieszczonych gęściej w obszarach sprawiających problemy lub zastosowanie węzłów opartych na zerach wielomianów Czebyszewa.

6.4 Wnioski

Interpolacja wielomianowa to użyteczna metoda do przybliżania funkcji, zwłaszcza gdy dysponujemy tylko kilkoma jej wartościami. Jednak ma swoje ograniczenia. Jeśli funkcja nie jest różniczkowalna, choćby na małych odcinkach, wyniki interpolacji mogą być dalekie od rzeczywistości. Nawet dla funkcji ciągłych i gładkich zwiększanie stopnia wielomianu nie zawsze daje lepsze rezultaty – czasami wręcz pogarsza dokładność.

Dlatego nie należy ślepo ufać wynikom interpolacji. Warto stosować różne podziały przedziału na węzły, zwłaszcza w miejscach, gdzie mogą pojawić się problemy. Pomocna jest analiza samego kształtu funkcji, aby zidentyfikować potencjalne trudności. Jeśli to możliwe, warto porównać wyniki interpolacji z dokładnym wykresem funkcji – dzięki temu łatwiej ocenić skuteczność metody.