Obliczenia naukowe. Sprawozdanie z listy 4.

Maciej Bazela 261743

9 grudnia 2022

Spis treści

Opis problemu interpolacji wielomianowej

Dla n punktów $(x_0, f(x_0)), (x_1, f(x_1)), \ldots, (x_n, f(x_n))$ interpolacja wielomianowa polega na znalezieniu wielomianu $p(x)$ stopnia co najwyżej n spełniającego $p(x_i) = f(x_i)$ dla $i = 0, 1, \ldots, n$. Istnieje dokładnie jeden wielomian spełniający ten warunek (tw. 1 na wykładzie 6.).

Wielomian p można przedstawić jako kombinację liniową wielomianów q_i dla $i = 0, 1, \ldots, n$:

$$\begin{gathered} q_0(x) = 1, \\ q_1(x) = (x - x_0), \\ \ldots, \\ q_n(x) = (x - x_0) \ldots (x - x_{n-1}) \end{gathered}$$

Takie przedstawienie prowadzi do układu z macierzą dolno-trójkątną:

$$\begin{bmatrix} 1 & 0 & \ldots & 0 \\ 1 & q_1(x_0) & \ldots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 1 & q_1(x_n) & \ldots & q_n(x_n) \end{bmatrix} \begin{bmatrix} c_0 \\ c_1 \\ \vdots \\ c_n \end{bmatrix} = \begin{bmatrix} f(x_0) \\ f(x_1) \\ \vdots \\ f(x_n) \end{bmatrix} \tag{1}$$

Wtedy możemy zapisać wielomian p(x) jako:

$$p(x) = \sum_{i=0}^{n} c_i q_i(x) \tag{2}$$

Łatwo zauważyć, że c_0 jest zależne od $f(x_0)$, c_1 jest zależne od $f(x_0)$ i $f(x_1)$, c_2 jest zależne od $f(x_0)$, $f(x_1)$ i $f(x_2)$, itd. Wprowadźmy notację $c_n = f[x_0, x_1, \ldots, x_n]$, która będzie oznaczać współczynnik przy x^n wielomianu stopnia co najwyżej n interpolującego f w $x_0, x_1, ...x_n$. $f[x_0, x_1, \ldots, x_n]$ nazywamy **ilorazem różnicowym**.

Zastępując c_i z poprzedniego układu równań notacją $f[x_0, \ldots, x_i]$ otrzymujemy **postać Newtona** dla wielomianu interpolacyjnego:

$$p(x) = \sum_{i=0}^{n} c_i q_i(x) = \sum_{i=0}^{n} f[x_0, x_1, \ldots, x_i] \prod_{j=0}^{i-1} (x - x_j) \tag{3}$$

Ponadto ilorazy różnicowe są zależne od siebie rekurencyjnie:

$$f[x_0, x_1, \ldots, x_i] = \frac{f[x_1, x_2, \ldots, x_i] - f[x_0, x_1, \ldots, x_{i-1}]}{x_i - x_0} \tag{4}$$

Naszym zadaniem na tej liście jest wykorzystanie powyższych zależności do wyznaczania wielomianu interpolującego daną funkcję f.

1 Zadanie 1

1.1 Opis problemu

Zadanie 1. polegało na zaimplementowaniu funkcji obliczającej ilorazy różnicowe.

Jako parametry wejściowe przyjmujemy wektor x długości $n+1$, który zawiera węzły $x_0, x_1, \ldots, x_n$ oraz wektor f długości $n+1$, który zawiera wartości funkcji f w węzłach $x_0, x_1, \ldots, x_n$.

Jako wynik zwracamy wektor c długości $n+1$, który zawiera wyliczone ilorazy różnicowe.

Funkcja nie powinna korzystać z tablic dwuwymiarowych (macierzy).

1.2 Rozwiązanie

Kod źródłowy rozwiązań do tego zadania znajduje się w pliku interpolation.jl w funkcji ilorazyRoznicowe.

Możemy zauważyć, że obliczanie ilorazów różnicowych można wykonać konstruując tablicę trójkątną (przykład dla 4 węzłów):

$$\begin{array}{llllll} x_0 & f[x_0] & \rightarrow & f[x_0,x_1] & \rightarrow & f[x_0,x_1,x_2] \rightarrow f[x_0,x_1,x_2,x_3] \\ x_1 & f[x_1] & \nearrow & f[x_1,x_2] & \nearrow & f[x_1,x_2,x_3] \nearrow \\ x_2 & f[x_2] & \nearrow & f[x_2,x_3] & \nearrow & \\ x_3 & f[x_3] & \nearrow & & & \end{array}$$

Aby wykonać obliczenia na jednowymiarowej tablicy, musimy zacząć od wpisania wszystkich wartości $f[x_i]$ jako początkowe wartości wektora wynikowego. Następnie w pętli będziemy "od tyłu" obliczać następne wartości, nadpisując w każdym kroku o jeden element mniej w tablicy:

$$\begin{array}{llll} f[x_0] & f[x_1] & f[x_2] & f[x_3] \\ f[x_0] & f[x_0,x_1] & f[x_1,x_2] & f[x_2,x_3] \\ f[x_0] & f[x_0,x_1] & f[x_0,x_1,x_2] & f[x_1,x_2,x_3] \\ f[x_0] & f[x_0,x_1] & f[x_0,x_1,x_2] & f[x_0,x_1,x_2,x_3] \end{array}$$

Powyższy sposób można zapisać w postaci podwójnej pętli:

```
function ilorazyRoznicowe(x, f)
  fx = [val for val in f]

  for i 1:length(f)
    for j in length(f):-1:i+1
      fx[j] = (fx[j] - fx[j-1]) / (x[j] - x[j-i])
    end
  end

  return fx
end
```

1.3 Testy poprawności implementacji

Aby sprawdzić poprawność zaimplementowanej funkcji, wykorzystałem przykład z wykładu:

x_i	$f[x_i]$			
3	1	2	$\frac{-3}{8}$	$\frac{7}{40}$
1	-3	$\frac{5}{4}$	$\frac{3}{20}$	
5	2	2		
6	4			

Wynik: `[1, 2, -3/8, 7/40]`

Oraz wymyśliłem własny:

x_i	$f[x_i]$			
1	-5	$\frac{7}{4}$	$\frac{5}{4}$	$\frac{-53}{24}$
5	2	8	-12	
6	10	-16		
7	-6			

Wynik: `[-5, 7/4, 5/4, -53/24]`

2 Zadanie 2

2.1 Opis problemu

Zadanie 2. polegało na napisaniu funkcji obliczającej wartość wielomianu interpolacyjnego stopnia n w postani Newtona ($N_n(x)$) w punkcie $x = t$. Rozwiązanie powinno korzystać z uogólnionej wersji algorytmu Hornera, działającego w czasie liniowym.

Jako parametry funkcja przyjmuje wektor x zawierający $n+1$ węzłów, wektor fx zawierający wyliczone ilorazy różnicowe funkcji f oraz t - punkt, w którym ma zostać obliczona wartość wielomianu.

Zwracamy wartość wielomianu w punkcie t.

2.2 Rozwiązanie

Kod źródłowy rozwiązań do tego zadania znajduje się w pliku interpolation.jl w funkcji warNewton.

Wielomian interpolacyjny stopnia n w postaci Newtona można zapisać jako:

$$N_n(x) \overset{(3)}{=} \sum_{i=0}^{n} f[x_0, x_1, \ldots, x_i] \prod_{j=0}^{i-1}(x - x_j) = \sum_{i=0}^{n} c_i \prod_{j=0}^{i-1}(x - x_j)$$

Korzystając z uogólnionej wersji algorytmu Hornera, możemy zapisać powyższy wielomian w postaci:

$$w_n(x) = f[x_0, x_1, \ldots, x_n]$$
$$w_{n-1}(x) = f[x_0, x_1, \ldots, x_{n-1}] + (x - x_{n-1})w_n(x)$$
$$N_n(x) = w_0(x)$$

Powyższe równania można zapisać w postaci jednej pętli:

```
function warNewton(x, fx, t)
  w = fx[length(fx)]
  for i in length(fx)-1:-1:1
    w = fx[i] + (t - x[i]) * w
  end
  return w
end
```

2.3 Testy poprawności implementacji

Sprawdzenie poprawności implementacji polegało na ręcznym obliczeniu wartości wielomianu w postaci Newtona dla przykładów z zadania 1 i porównaniu ich z wynikami zwracanymi przez funkcję *warNewton*.

3 Zadanie 3

3.1 Opis problemu

W zadaniu 3. należało napisać funkcję, która dla danego wielomianu interpolacyjnego w postaci Newtona oraz węzłów $x_0, x_1, \ldots, x_n$ oblicza współczynniki tego wielomianu w postaci naturalnej, tj. $a_n x^n + a_{n-1}x^{n-1} + \cdots + a_1 x + a_0$.

Jako parametry funkcja przyjmuje wektor x zawierający $n+1$ równoodległych punktów oraz wektor fx zawierający wyliczone ilorazy różnicowe funkcji f. Zwracamy wektor a zawierający współczynniki wielomianu w postaci naturalnej.

Złożonośc czasowa powinna wynosić $O(n^2)$.

3.2 Rozwiązanie

Kod źródłowy rozwiązań do tego zadania znajduje się w pliku interpolation.jl w funkcji naturalna.

Korzystając ze wzorów z poprzedniego zadania, możemy zauważyć pewną zależność:

$$\begin{aligned}
w_n(x) &= c_n = a_n \\
w_{n-1}(x) &= c_{n-1} + (x - x_{n-1})w_n(x) = c_{n-1} + (x - x_{n-1})c_n = xc_n + c_{n-1} - x_{n-1}c_n = \\
&\quad xa_n + a_{n-1} \\
w_{n-2}(x) &= c_{n-2} + (x - x_{n-2})w_{n-1}(x) = c_{n-2} + (x - x_{n-2})c_{n-1} + (x - x_{n-2})(x - x_{n-1})c_n = \\
&\quad c_{n-2} + xc_{n-1} + -x_{n-2}c_{n-1} + x^2c_n - xx_{n-1}c_n - xx_{n-2}c_n + x_{n-2}x_{n-1}c_n = \\
&\quad x^2c_n + x(c_{n-1} - x_{n-1}c_n - x_{n-2}c_n) + c_{n-2} - x_{n-2}c_{n-1} + x_{n-2}x_{n-1}c_n = \\
&\quad x^2c_n + x(c_{n-1} - x_{n-1}c_n - x_{n-2}c_n) + c_{n-2} - x_{n-2}(c_{n-1} - x_{n-1}c_n) = \\
&\quad x^2a_n + xa_{n-1} + a_{n-2}
\end{aligned}$$

Algorytm będzie opierał się na wyliczaniu kolejnych wartości $w_i(x)$, a następnie zapisywaniu ich w wektorze a. Wyliczanie zaczniemy od $w_n(x)$, ponieważ $w_n(x) = c_n$. Następne wartości będziemy budować na podstawie poprzednio wyliczonej wartości.

Przy schodzeniu o jeden stopień w dół "odkrywamy" nowe składniki "wyższych" współczynników. Np. przy rozwinięciu $w_{n-2}(x)$ dostajemy $\cdots + x(c_{n-1} - x_{n-1}c_n - x_{n-2}c_n) + \ldots$, czyli do współczynnika a_{n-1} musimy dodać $-x_{n-2}c_n$.

Każde takie zejście do w_{k-1} wpłynie na współczyniki a_i o dokładnie $-x_k w_{i+1}$ dla $i = k, k+1, \ldots, n-1$.

Powyższe rozumowanie można zwięźle zapisać za pomocą podwójnej pętli:

```
function naturalna(x, fx)
  a = [0 for _ in fx]
  a[end] = fx[end]
  for i in length(fx)-1:-1:1
    a[i] = fx[i] - a[i+1] * x[i]
    for j in i+1:length(fx)-1
      a[j] = a[j] - a[j+1] * x[i]
    end
  end
  return a
end
```

3.3 Testy poprawności implementacji

Sprawdzenie poprawności polegało po raz kolejny na ręcznym przeliczeniu przykładów z zadania 1. i 2. na kartce i porównaniu wyników z wynikami algorytmu. Aby oszczędzić masy obliczeń, postanowiłem pominąć wpisywanie ich w to sprawozdanie.

4 Zadanie 4

4.1 Opis problemu

W zadaniu 4. mieliśmy, wykorzystując zaimplementowanie w zadaniach 1-3 funkcje, napisać program *rysujNnfx(f, a, b, n)*, który rysuje wykres funkcji f i interpolującej jej wielomianu $p(x)$ dla zadanego przedziału $[a, b]$ i n węzłów.

W interpolacji powinniśmy użyć węzłów równoodległych:

$$x_k = a + kh$$
$$h = \frac{b-a}{n}$$
$$k = 0, 1, \ldots, n$$

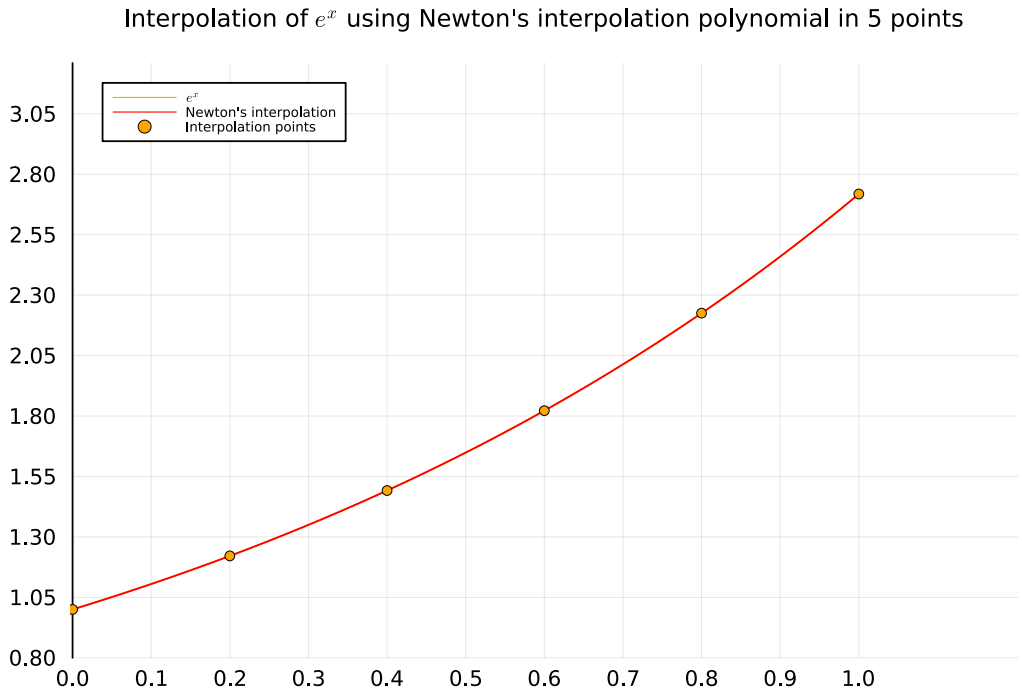
Wyniki działania tego programu są widoczne w zadaniach 5. i 6.

5 Zadanie 5

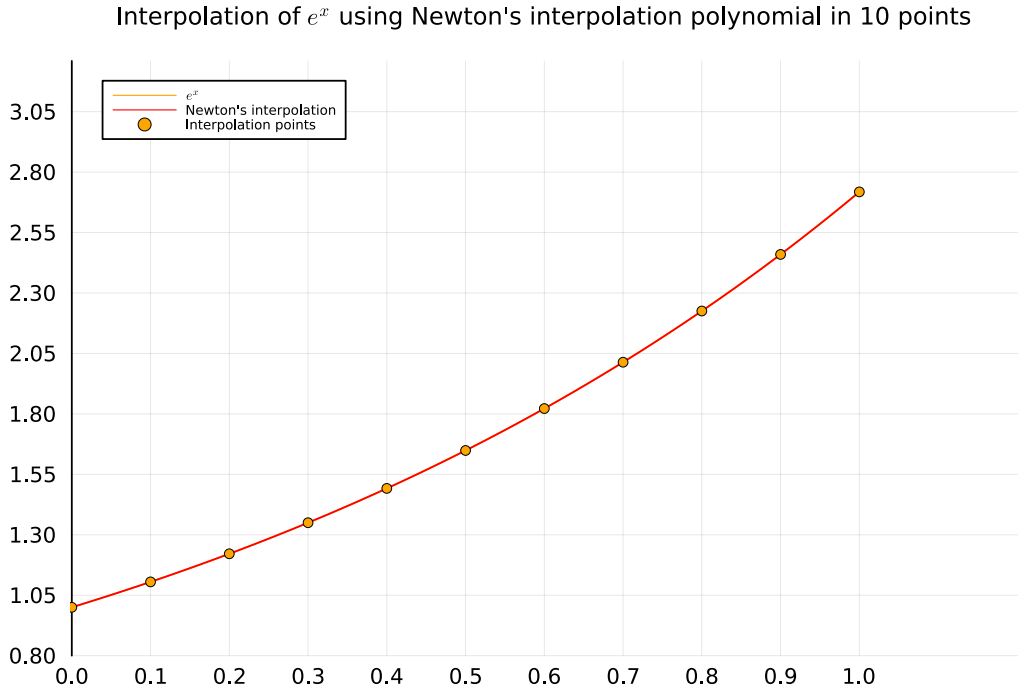
Zadanie 5. polegało na przetestowaniu funkcji *rysujNnfx* na poniższych przykładach:

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

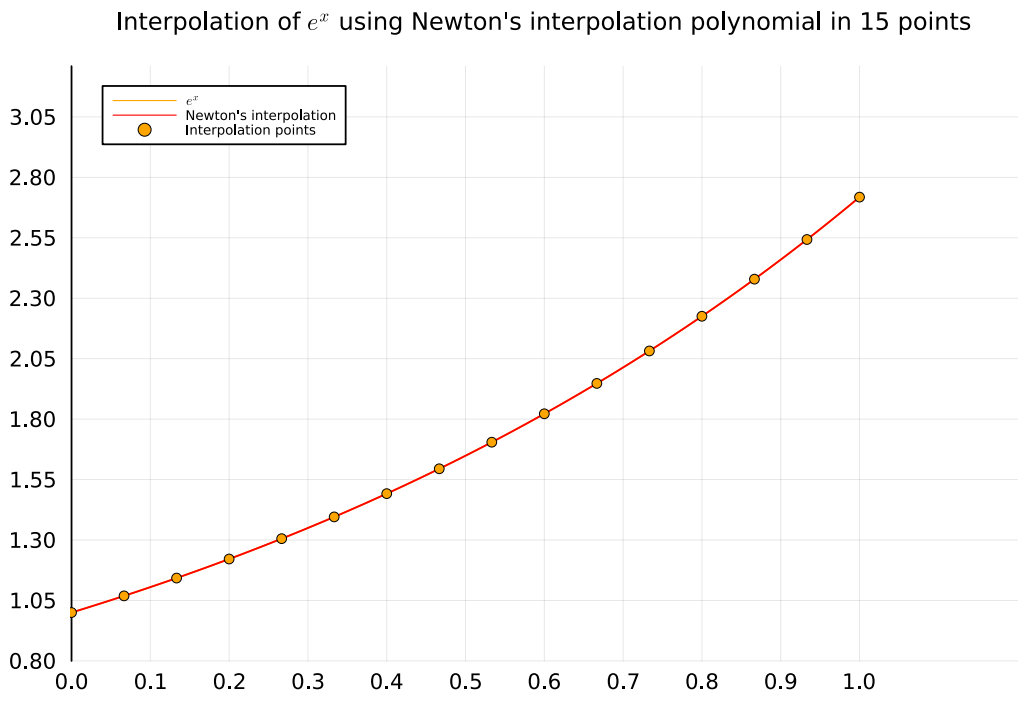
5.1 Wyniki



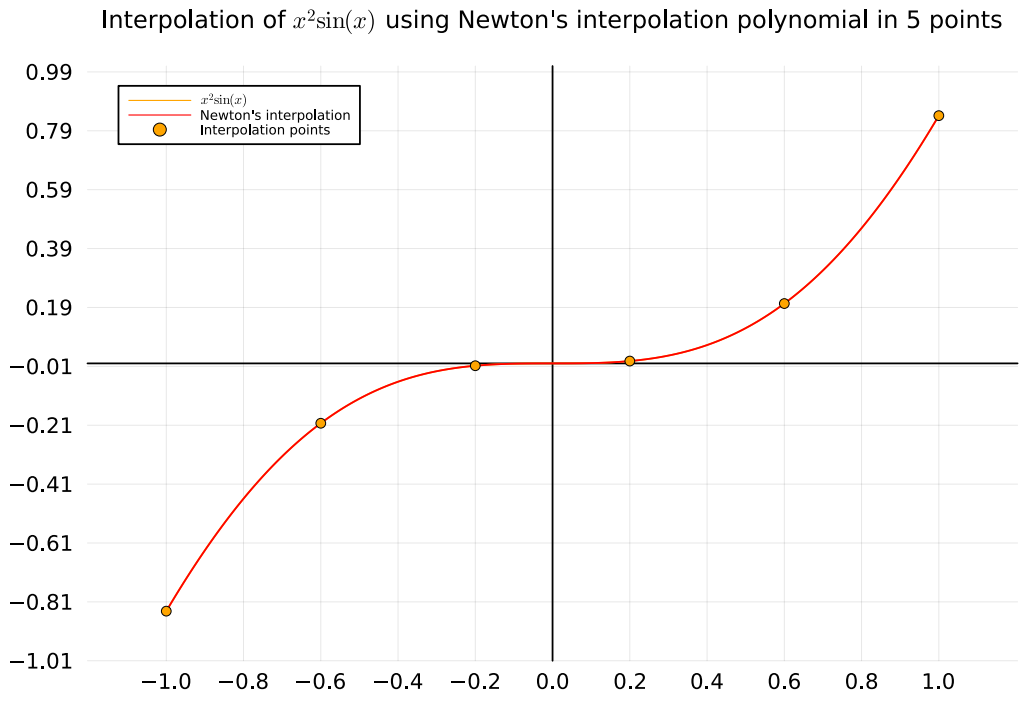
Rysunek 1: $f(x) = e^x$, $[a, b] = [0, 1]$, $n = 5$



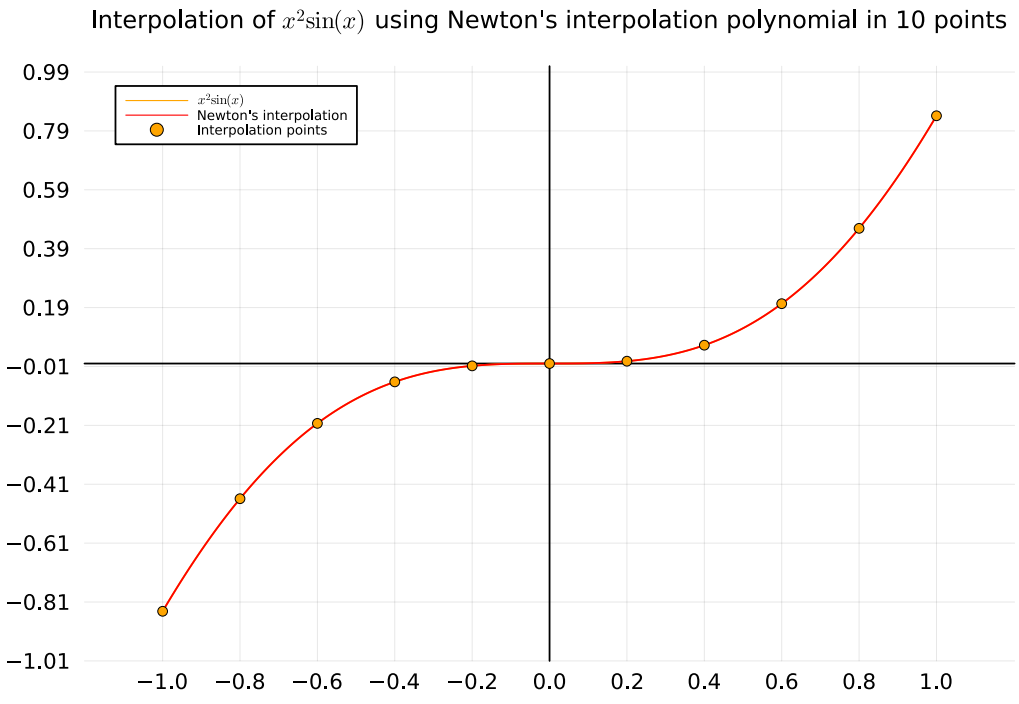
Rysunek 2: $f(x) = e^x$, $[a, b] = [0, 1]$, $n = 10$



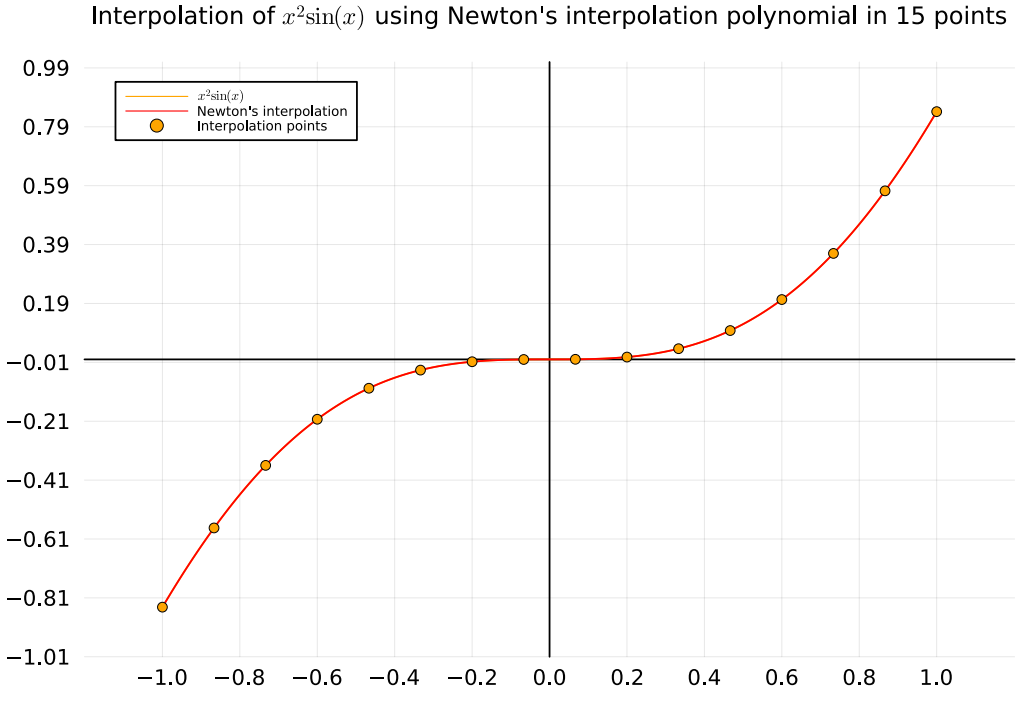
Rysunek 3: $f(x) = e^x$, $[a, b] = [0, 1]$, $n = 15$



Rysunek 4: $f(x) = x^2 \sin(x)$, $a, b = [-1, 1]$, $n = 5$



Rysunek 5: $f(x) = x^2 \sin(x)$, $a, b = [-1, 1]$, $n = 10$



Rysunek 6: $f(x) = x^2 \sin(x)$, $a, b = [-1, 1]$, $n = 15$

5.2 Wnioski

Interpolacja funkcji w podanych zakresach całkowicie pokrywa się z zadaną funkcją. Dzieje się tak, ponieważ podane funkcje nie zmieniają się gwałtownie oraz są na zadanym przedziale rosnące (pochodna nie zmienia znaku). Dzięki temu możemy uzyskać bardzo dokładne przybliżenie tych funkcji z wykorzystaniem wielomianów interpolacyjnych (które są funkcjami gładkimi).

Interpolacja wielomianem w postaci Newtona działa dobrze dla funkcji gładkich.

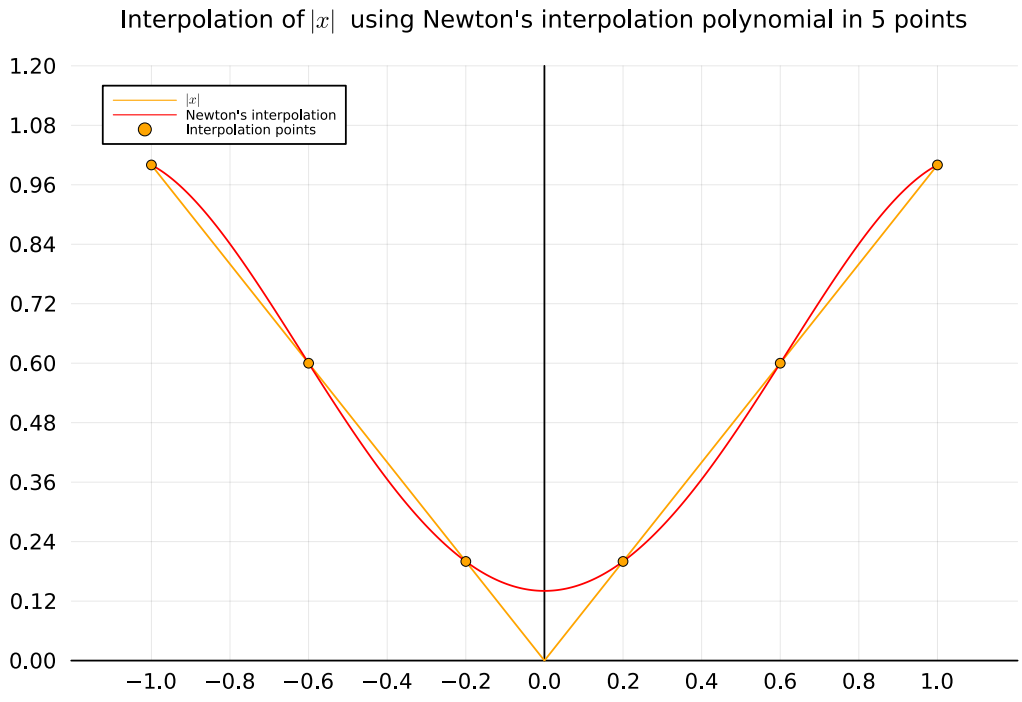
6 Zadanie 6

6.1 Opis problemu

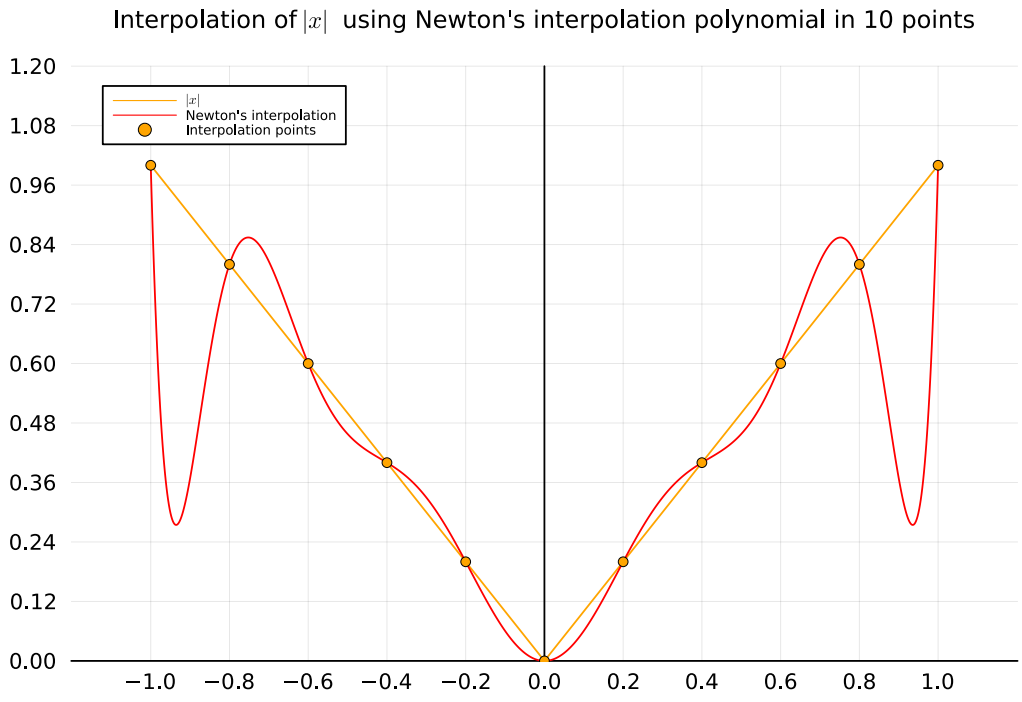
Zadanie 5. polegało na przetestowaniu funkcji *rysujNnfx* na poniższych przykładach:

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

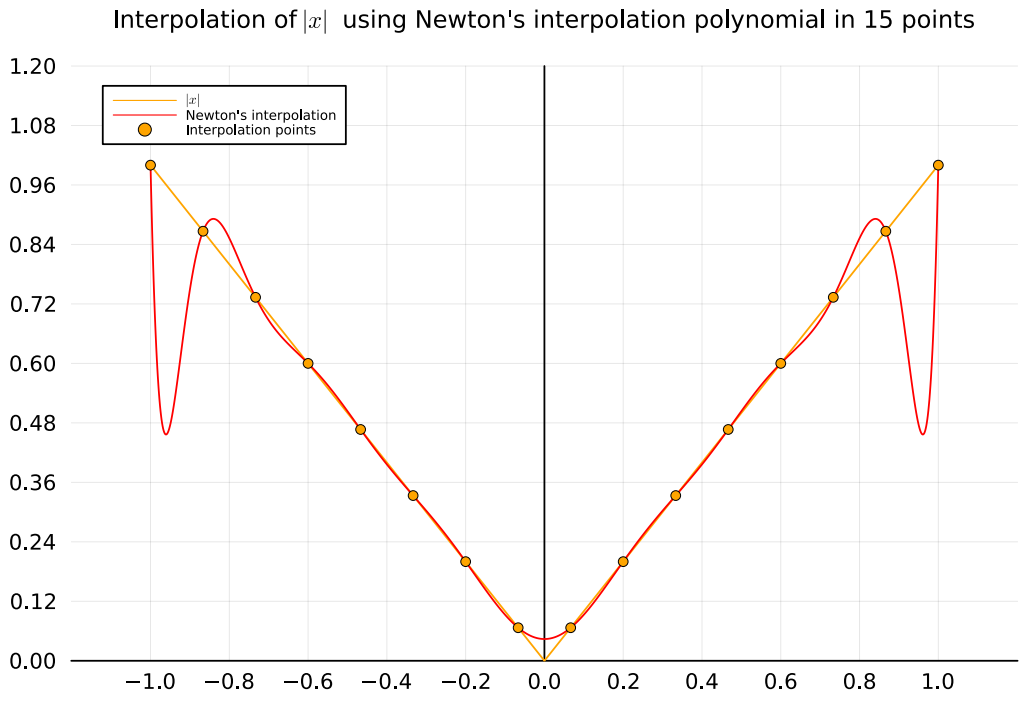
6.2 Wyniki



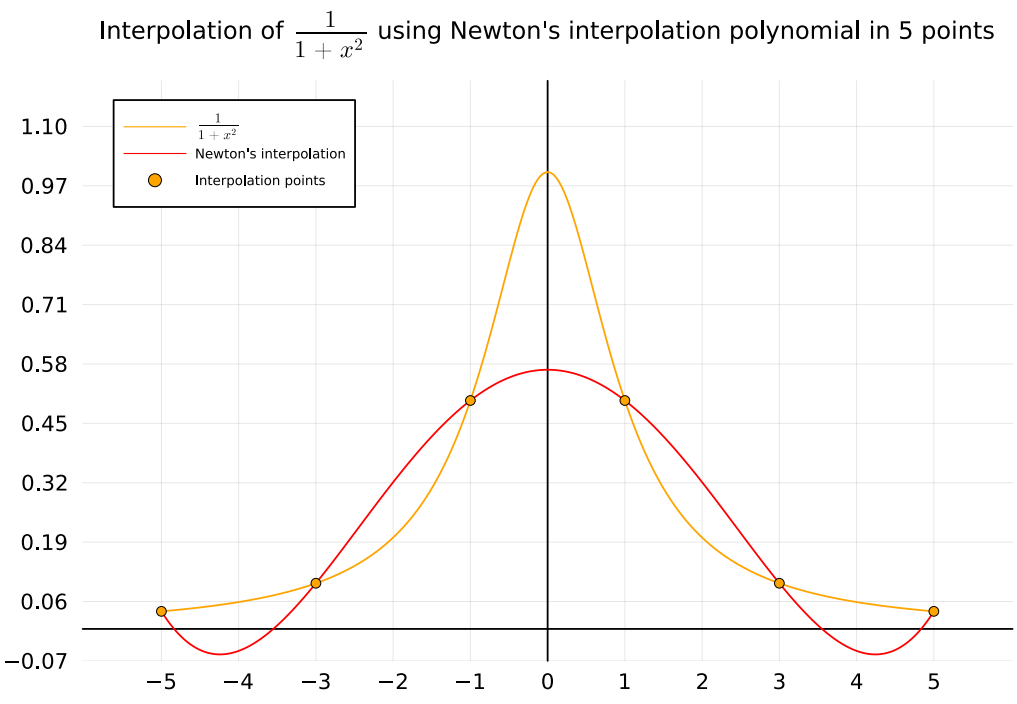
Rysunek 7: $f(x) = |x|$, $[a, b] = [-1, 1]$, $n = 5$



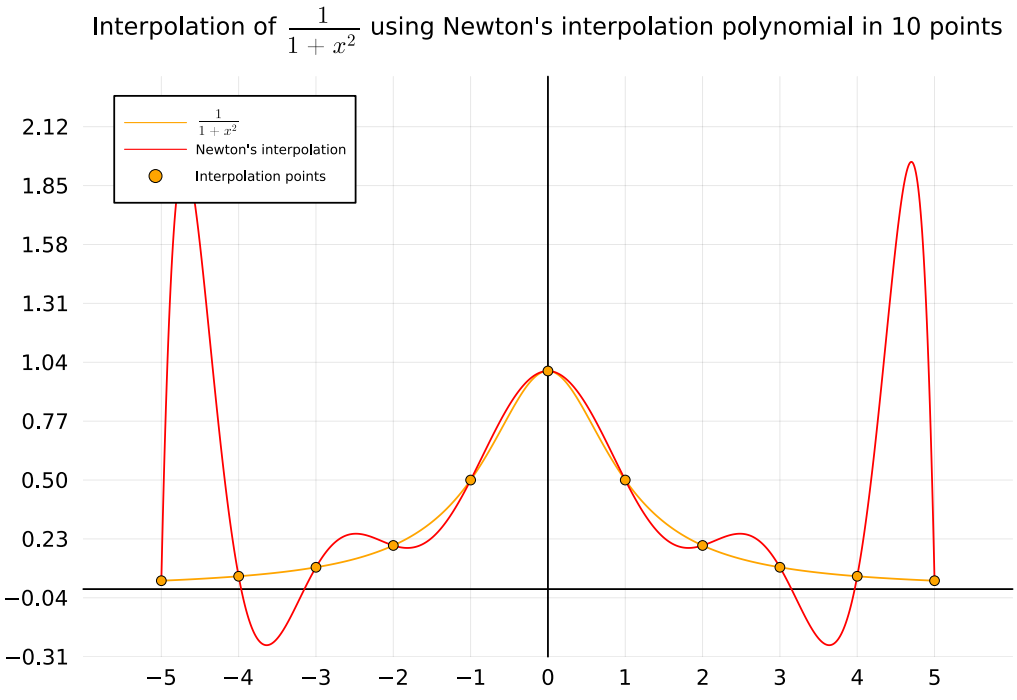
Rysunek 8: $f(x) = |x|$, $[a, b] = [-1, 1]$, $n = 10$



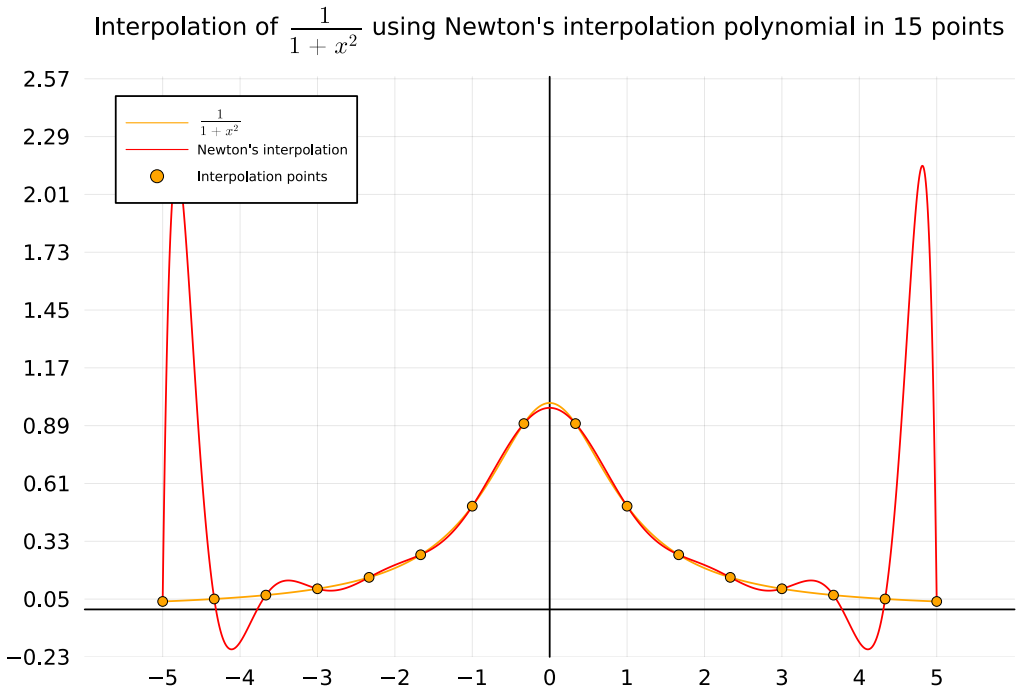
Rysunek 9: $f(x) = |x|$, $[a, b] = [-1, 1]$, $n = 15$



Rysunek 10: $f(x) = \frac{1}{1+x^2}$, $a, b = [-5, 5]$, $n = 5$



Rysunek 11: $f(x) = \frac{1}{1+x^2}$, $a, b = [-5, 5]$, $n = 10$



Rysunek 12: $f(x) = \frac{1}{1+x^2}$, $a, b = [-5, 5]$, $n = 15$

6.3 Wnioski

Interpolacje dla zadanych funkcji poradziły sobie dużo gorzej w porównaniu z poprzednim zadaniem.

W przypadku funkcji $f(x) = |x|$ problemem jest to, że $f(x) = |x|$ jest nieróżniczkowalna w $x = 0$ i bardzo ostra. Przy krańcach przedziału pojawiają się dziwne anomalie w postaci nagłych "skoków". Bardzo ciężko jest uzyskać wielomian, który będzie dobrze się dopasowywał do funkcji $|x|$. Dla większych wartości n interpolacja zachowuje się coraz gorzej na krańcach.

W przypadku funkcji $f(x) = \frac{1}{1+x^2}$ problemem jest wysoki stopień interpolującego wielomianu. Im bliżej krańców przedziału, tym większy jest błąd interpolacji. Dla większych wartości n interpolacja zachowuje się coraz gorzej na krańcach. Wbrew intuicji, mówiącej, że im większy stopień wielomianu, tym dokładniejsza interpolacja, w tym przypadku jest to nieprawda.

Ten fenomen został nazwany *efektem Rungego*. Jest to efekt, który występuje w przypadku interpolacji wielomianami w punktach równoodległych. Wielomiany interpolacyjne mają bardzo duży stopień w okolicach krańców przedziału, co powoduje, że interpolacja jest bardzo zanieczyszczona.

Efekt ten można zniwelować wybierając węzły, które są gęściej rozłożone przy krańcach przedziału lub poprzez zastosowanie wielomianów Czebyszewa. Wielomiany te mają stopień stały i są wypukłe w okolicach krańców przedziału, co akurat dla tego przykładu byłoby bardzo korzystne.