

Obliczenia Naukowe

Sprawozdanie z laboratorium nr 4

Piotr Kawa

0.Wprowadzenie

Lista obejmowała zagadnienia związane z interpolacją. Należało stworzyć i umieścić w module 4 funkcje. Następnie należało przetestować utworzone funkcje na podanych w treści zadania przykładach oraz wyciągnąć wnioski.

1. Zadanie pierwsze

1.1. Cel zadania

Celem zadania pierwszego było napisanie funkcji liczącej ilorazy różnicowe. Jest to wielkość, która opisuje przyrost funkcji na danym przedziale. Jest ona wykorzystywana przy zagadnieniu interpolacji, czyli jednej z metod numerycznych, która zajmuje się przybliżaniem funkcji. Uzyskiwane jest to dzięki wyznaczaniu w zadanym przedziale tak zwanej funkcji interpolacyjnej.

Algorytm obliczania ilorazów różnicowych prezentuje się następująco. Aby uzyskać wartości rzędu zerowego i pierwszego wykorzystuje się następujące wzory – odpowiednio $f[x_0] = f(x_0)$ oraz $f[x_0, x_1] = \frac{f(x_1) - f(x_0)}{x_1 - x_0}$. W celu obliczenia ilorazów wyższych rzędów używany jest wzór

$$f[x_0, x_1, \dots, x_k] = \frac{f[x_1, \dots, x_k] - f[x_0, \dots, x_{k-1}]}{x_k - x_0} \quad (1)$$

Powysze wzory można zinterpretować następująco. $f[x_0]$ jest współczynnikiem przy x^0 wielomianu stopnia 0 interpolującego zadaną funkcję w x_0 . Kolejna z podanych wartości to współczynnik przy x^1 wielomianu stopnia ≤ 1 interpolującego w x_0, x_1 . Ostatni wzór to uogólnienie – określa on współczynnik przy x^n .

Podana funkcja miała zostać utworzona bez korzystania z tablic dwuwymiarowych. Implementacja składała się z dwóch funkcji – pierwsza z nich nazwana *ilorazyRoznicowe*($x :: Vector\{Float64\}, f :: Vector\{Float64\}$) składała się z pętli, która każdej komórce zwracanej tabeli przypisywała w pętli wynik funkcji *iloraz*($x :: Vector\{Float64\}, f :: Vector\{Float64\}$).

Ta natomiast była reprezentacją powyższych wzorów. Gdy wektor miał długość 1 bądź 2 (czyli szukaną wartością było $f[x_0]$ bądź $f[x_0, x_1]$) zwracane były wyliczone wartości. W przypadku gdy długość była dłuższa wykorzystywany był rekurencyjnie wzór (1).

Utworzona funkcja miała mieć postać:

function ilorazyRoznicowe($x :: Vector\{Float64\}, f :: Vector\{Float64\}$)

gdzie:

x – wektor długości $n + 1$ zawierający węzły x_0, \dots, x_n

$x[1] = x_0, \dots, x[n + 1] = x_n$

f – wektor długości $n + 1$ zawierający wartości interpolowanej funkcji w węzłach $f(x_0), \dots, f(x_n)$

oraz zwracać:

fx – wektor długości $n + 1$ zawierający wartości obliczone ilorazy różnicowe

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

$$fx[2] = f[x_0, x_1], \dots, fx[n] = f[x_0, \dots, x_{n-1}], \quad fx[n+1] = f[x_0, \dots, x_n]$$

2. Zadanie drugie

2.1. Cel zadania

Celem zadania drugiego była implementacja algorytmu z zadania 4.8. Była to funkcja obliczająca wartość wielomianu interpolacyjnego stopnia n w postaci Newtona $N_n(x)$ w punkcie $x = t$. Wykorzystany w tym celu miał zostać uogólniony algorytm Hornera, a złożoność obliczeniowa tego algorytmu miała wynosić $O(n)$.

Do napisania funkcji służyć miał algorytm, który znaleźć można było na jednej z list ćwiczeniowych. Składa się on z trzech wzorów:

$$w_n(x) = f[x_0, x_1, \dots, x_n] \quad (1)$$

$$w_k(x) = f[x_0, x_1, \dots, x_k] + (x - x_k) w_{k+1}(x) \quad (k = n - 1, \dots, 0) \quad (2)$$

$$N_n(x) = w_0(x) \quad (3)$$

Interpretuje się je następująco – najpierw następuje przypisanie do w_n wektora ilorazów różnicowych $f[x_0, x_1, \dots, x_n]$. Krok ten jest wymagany ponieważ algorytm odwołuje się do poprzednio wyliczonych wartości w .

Następnie w pętli (od $n - 1$ do 0) wyliczane są kolejne wartości w_k . Następuje to poprzez obliczenie sumy ilorazu różnicowego $f[x_0, x_1, \dots, x_k]$ oraz różnicy zadanego punktu z wartością punktu x_k pomnożoną przez poprzednią wartość w (czyli $k + 1$).

Szukana wartość wielomianu ($N_n(x)$) jest równa wynikowi ostatniej iteracji, czyli $w_0(x)$.

Implementacyjnie funkcja składała się z następujących operacji. Najpierw następowało przypisanie do ostatniej komórki w (tablica równa długością tablicy fx) ostatniej komórki fx . Następnie w pętli od $n - 1$ do 1 następowało przypisanie do aktualnej komórki w wartości wyliczonej za pomocą wzoru (2). Ostatnim etapem było zwrócenie nt – wartości $w[1]$.

Wyliczenie szukanej wartości wymaga jednego przejścia pętli dla n węzłów, co jest dowodem na to, że jego złożoność obliczeniowa wynosi $O(n)$.

Utworzona funkcja miała mieć postać:

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

gdzie:

x – wektor długości $n + 1$ zawierający węzły x_0, \dots, x_n

$$x[1] = x_0, \dots, x[n + 1] = x_n$$

fx – wektor długości $n + 1$ zawierający ilorazy różnicowe

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

$$fx[2] = f[x_0, x_1], \dots, fx[n] = f[x_0, \dots, x_{n-1}], \quad fx[n + 1] = f[x_0, \dots, x_n]$$

t – punkt, w którym należy obliczyć wartość wielomianu

oraz zwracać:

nt – wartość wielomianu w punkcie t .

3. Zadanie trzecie

3.1. Cel zadania

Celem zadania trzeciego było napisanie funkcji obliczającej współczynniki postaci naturalnej wielomianu interpolacyjnego w postaci Newtona. Posiadanymi informacjami były jego współczynniki ($c_0 = f[x_0], c_1 = f[x_0, x_1], \dots, c_n = f[x_0, \dots, x_n]$) oraz węzły x_0, x_1, \dots, x_n . Algorytm miał mieć złożoność obliczeniową $O(n^2)$.

Implementacja wygląda następująco. Podobnie jak w przypadku algorytmu z zadania drugiego kolejne wartości wyliczane były dzięki przejściu pętli od końca do początku tablicy korzystając z wcześniej obliczonych wartości.

W tym celu do ostatniej komórki tablicy wynikowej (a) przypisana została wartość ilorazu różnicowego węzła na pozycji n . Algorytm dla każdej kolejnej komórki tablicy wynikowej wykonuje następujące obliczenia – począwszy od aktualnej komórki j aż do komórki $n - 1$ wartości wyliczane w pętli są za pomocą wzoru $a[j] = a[j] - (a[j + 1] * x[i])$.

Ostatnim krokiem jest zwrócenie żądanej wartości – tablicy a zawierającej współczynniki wielomianu postaci normalnej.

Algorytm składa się z dwóch pętli (każda wykonuje się po n razy) – złożoność wynosi $O(n^2)$.

Utworzona funkcja miała mieć postać:

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

gdzie:

x – wektor długości $n + 1$ zawierający węzły x_0, \dots, x_n

$$x[1] = x_0, \dots, x[n + 1] = x_n$$

fx – wektor długości $n + 1$ zawierający ilorazy różnicowe

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

$$fx[2] = f[x_0, x_1], \dots, fx[n] = f[x_0, \dots, x_{n-1}], \quad fx[n+1] = f[x_0, \dots, x_n]$$

oraz zwracać:

a – wektor długości $n + 1$ zawierający obliczone współczynniki postaci naturalnej

$$a[1] = a_0$$

$$a[2] = a_1, \dots, a[n] = a_{n-1}, \quad a[n+1] = a_n$$

4. Zadanie czwarte

4.1. Cel zadania

Celem zadania czwartego było napisanie programu, który zinterpoluje funkcję $f(x)$ na pewnym przedziale $[a, b]$ przy wykorzystaniu wielomianu interpolacyjnego stopnia n (w postaci Newtona). Operacja miała wykorzystywać funkcje *ilorazyRoznicowe()* oraz *warNewton()* oraz węzły równoodległe w postaci

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

Następnie zarówno wielomian interpolacyjny jak i interpolowana funkcja miała zostać przedstawiona na wykresie.

4.2 Rozwiążanie

rysujNnfx() interpoluje podaną funkcję w odpowiednim przedziale, wynik tejże operacji (wraz z samą funkcją) jest następnie przedstawiony na wykresie.

Implementacja wygląda następująco. Korzystając z podanych argumentów (końce przedziałów oraz stopień wielomianu interpolacyjnego) następuje wyliczenie wartości węzłów równoodległych oraz wartości funkcji w tychże punktach. Są one następnie wykorzystane do obliczenia ilorazów różnicowych za pomocą funkcji *ilorazyRoznicowe()*.

Wyliczenie danych, które zostaną przedstawione na wykresie odbywa się za pomocą *warNewton()* oraz *f()* (czyli funkcji podanej jako argument).

Wygenerowanie wykresów reprezentujących wyliczone dane odbywa się za pomocą biblioteki *Plots*.

Utworzona funkcja miała mieć postać:

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

gdzie:

f – funkcja $f(x)$ zadana jako anonimowa funkcja

a, b – przedział interpolacji

n – stopień wielomianu interpolacyjnego

oraz zwracać:

- narysowany wielomian interpolacyjny oraz interpolowaną funkcję w przedziale $[a, b]$

Przedstawione powyżej funkcje znajdowały się w module języka Julia nazwanym *Interpolation*. Do tego modułu zostały dopisane testy, które sprawdzały poprawność funkcji. Znajdowały się tam między innymi przykłady z wykładu, w innych przypadkach poprawność wyników była weryfikowana za pomocą zewnętrznych źródeł, bądź wyliczeniami ręcznymi. W zależności od wyniku testu wyświetlany był odpowiedni komunikat.

Przykładowe testy:

- dla *ilorazyRoznicowe()*
 $f(x) = [1.0, -3.0, 2.0, 4.0]$, $x = [3.0, 1.0, 5.0, 6.0]$
wynik= [1.0, 2.0, -0.375, 0.175]
- dla *warNewton()*
 $f(x) = [1.0, -23.0, -54.0, -954.0]$, $x = [5.0, -7.0, -6.0, 0.0]$, $t = 6.0$
wynik= 666.0
- dla *naturalna()*
 $f(x) = [1.0, 2.0, 3.0, 4.0]$, $x = [2.0, 2.0, 5.0, 1.0]$,
wynik= [-71.0, 86.0, -33.0, 4.0]

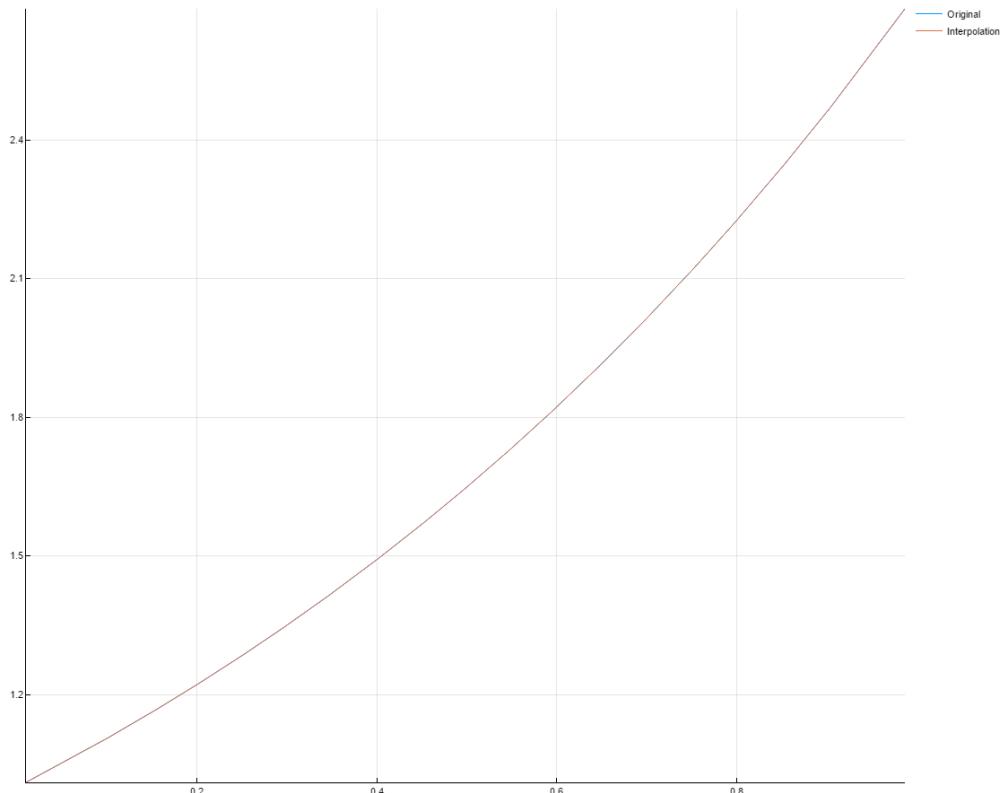
5. Zadanie piąte

5.1 Cel zadania

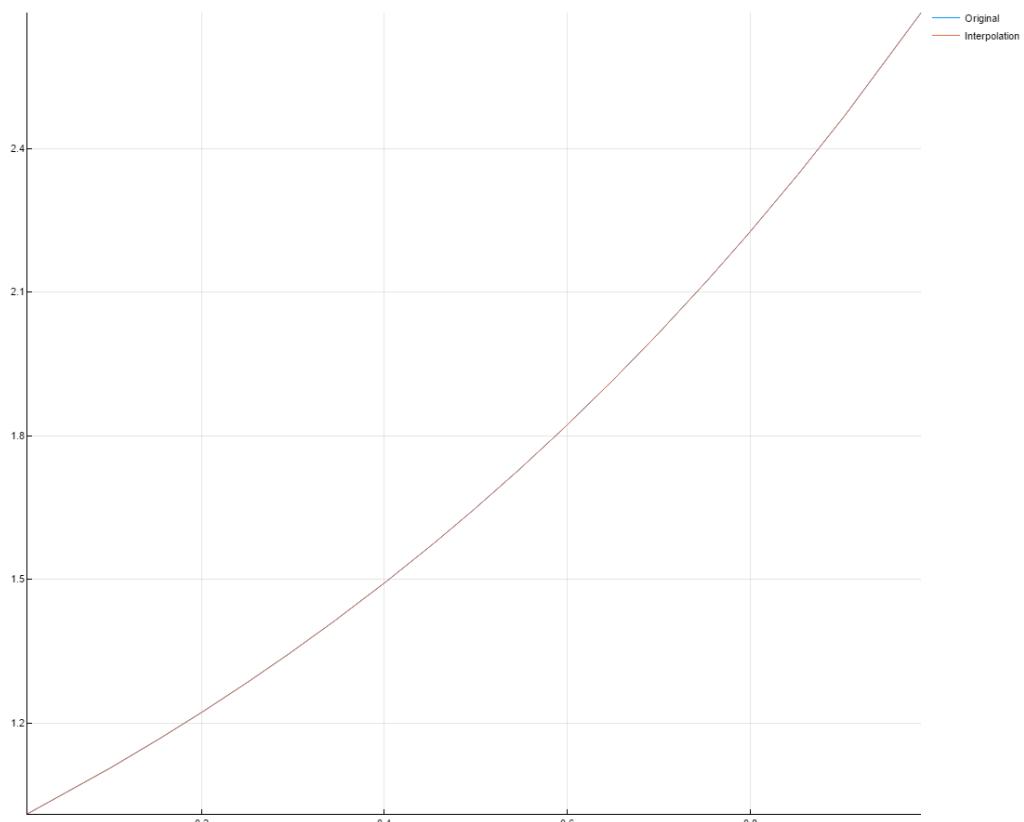
Celem zadania piątego było przetestowanie funkcji *rysujNnf(x(f, a, b, n))* na dwóch przykładach:

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

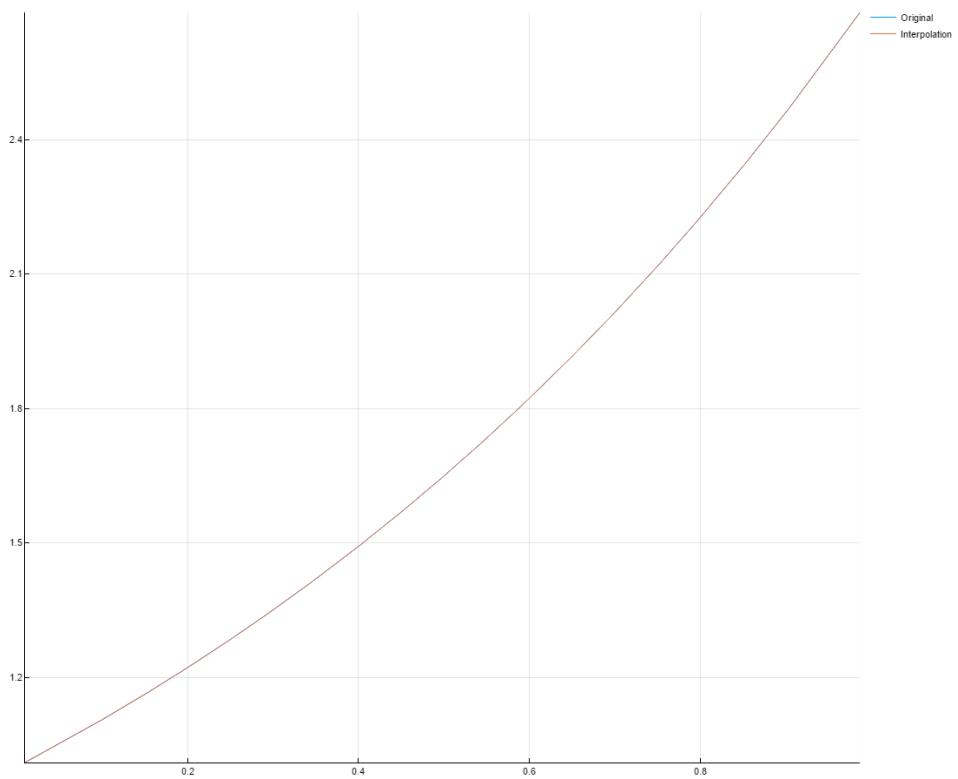
5.2 e^x



$n = 5$

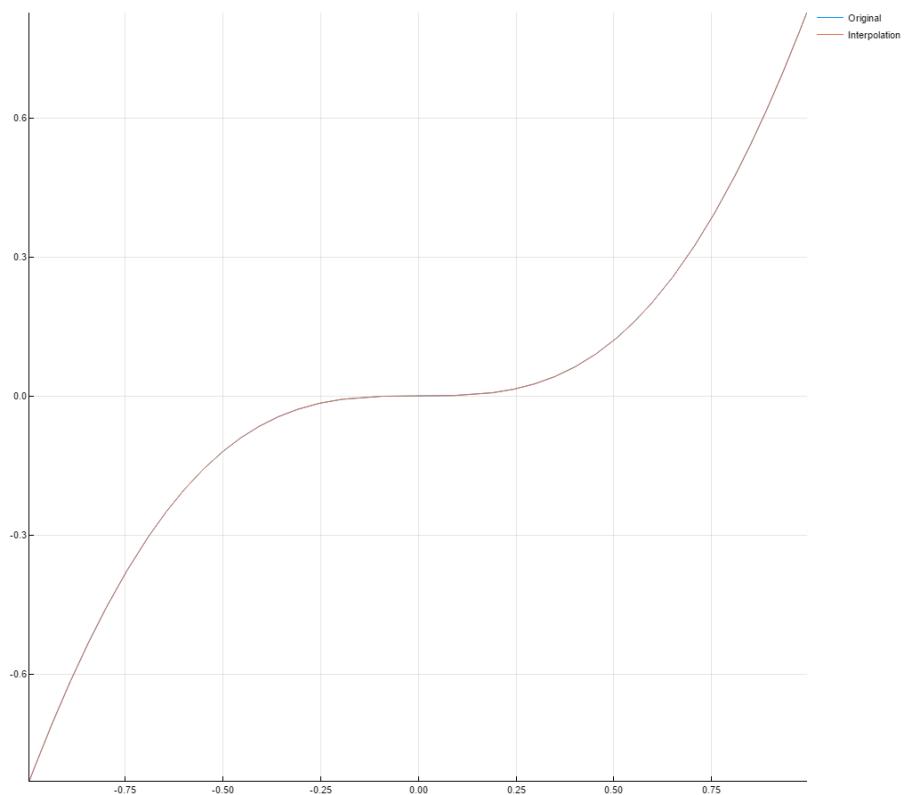


$n = 10$

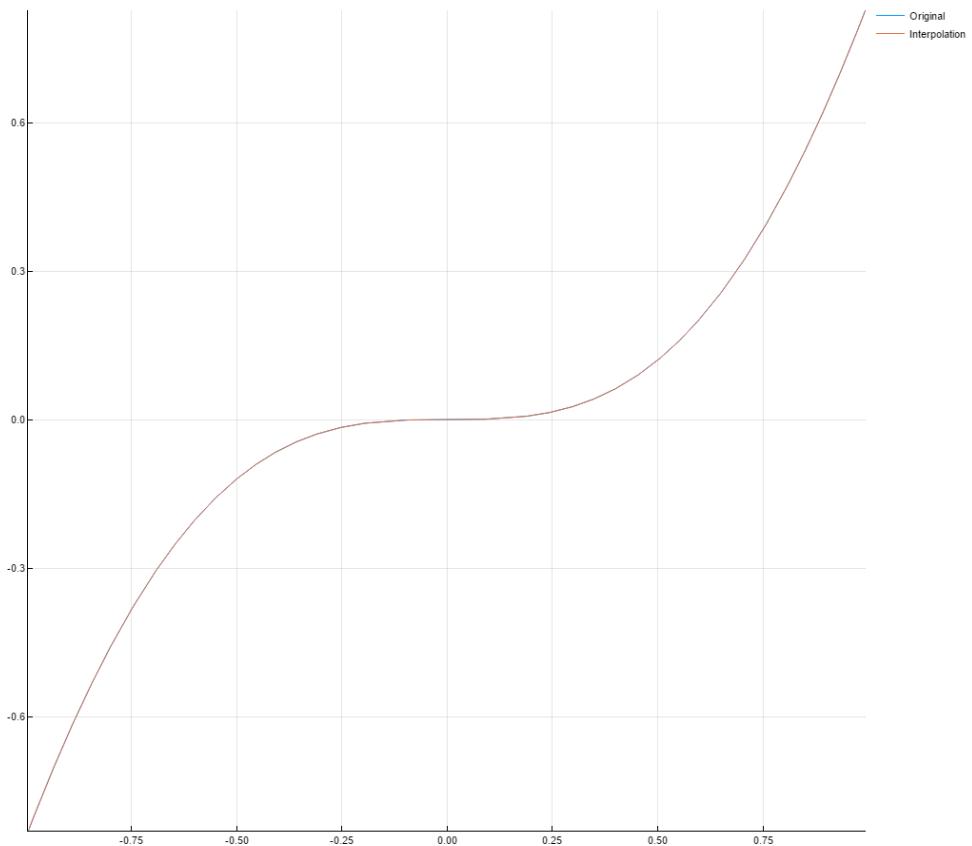


$n = 15$

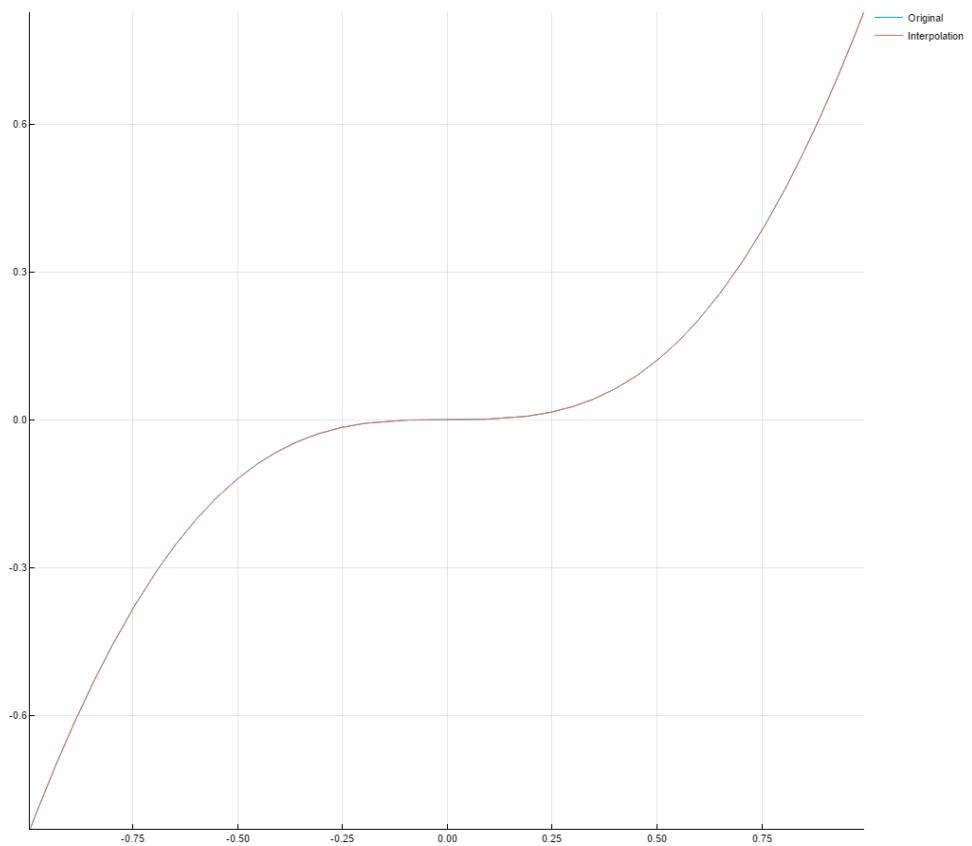
5.3 $x^2 \sin(x)$



$n = 5$



$n = 10$



$n = 15$

5.3 Wnioski

Analizując przedstawione powyżej wykresy można dojść do dwóch wniosków – wspólnych zarówno dla funkcji e^x jak i $x^2 \sin(x)$. Pierwszy z nich dotyczy samych funkcji interpolacyjnych – mają one takie same wartości. Drugi natomiast związany jest z funkcją oryginalną i interpolacyjną - wywnioskować można, że one również składają się z tych samych danych.

Są to jednak mylne wnioski. Jeśli przybliży się wykres wystarczajco mocno widoczne są pewne rozbieżności pomiędzy wartościami zarówno funkcji interpolacyjnych oraz funkcji oryginalnej jak i pomiędzy samym wyglądem funkcji interpolacyjnych - w przypadku zastosowania wyższych stopni widoczna jest większa dokładność (mniejsze odchylenie od oryginalnej funkcji).

Nie zmienia to jednak faktu, że przybliżenie funkcji, które zostało otrzymane dzięki temu algorytmowi jest bardzo bliskie prawdziwym wartościom.

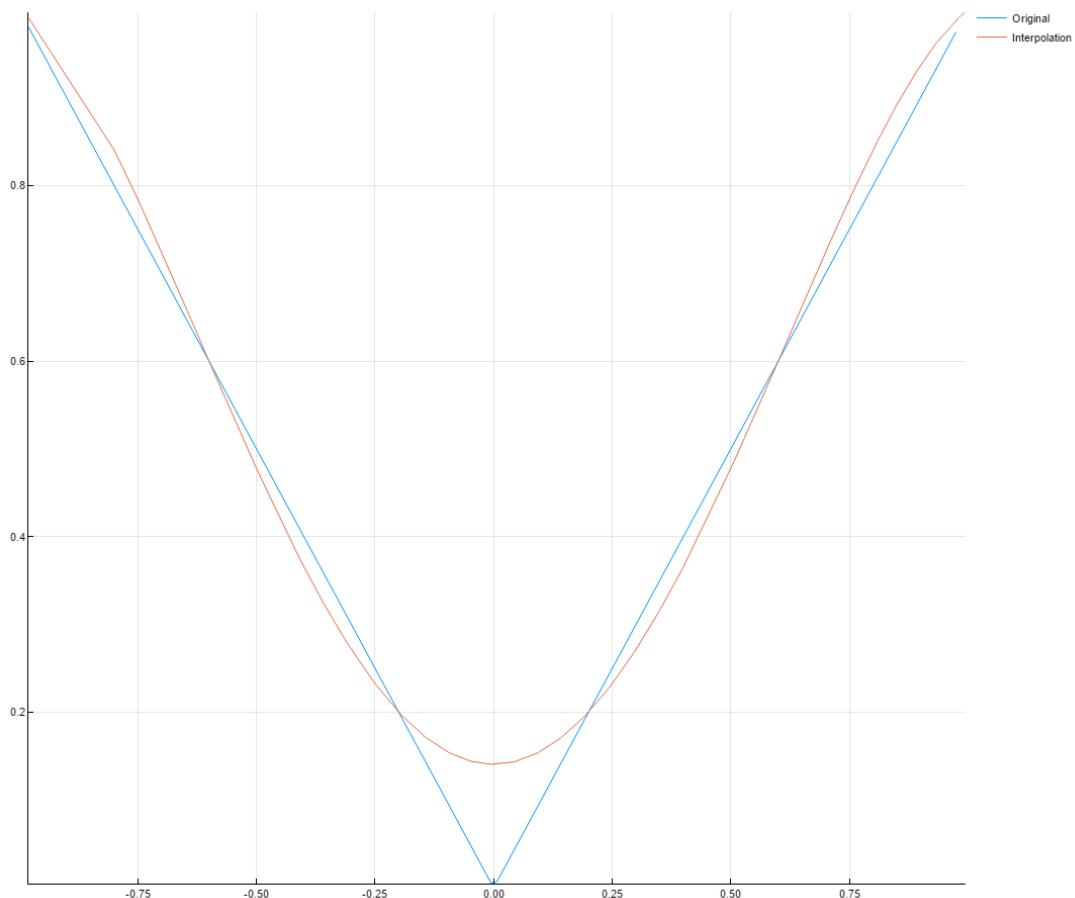
6. Zadanie szóste

Zadanie szóste podobnie jak zadanie piąte polegało na przetestowaniu funkcji $y = |x|$ na dwóch przykładach, tym razem:

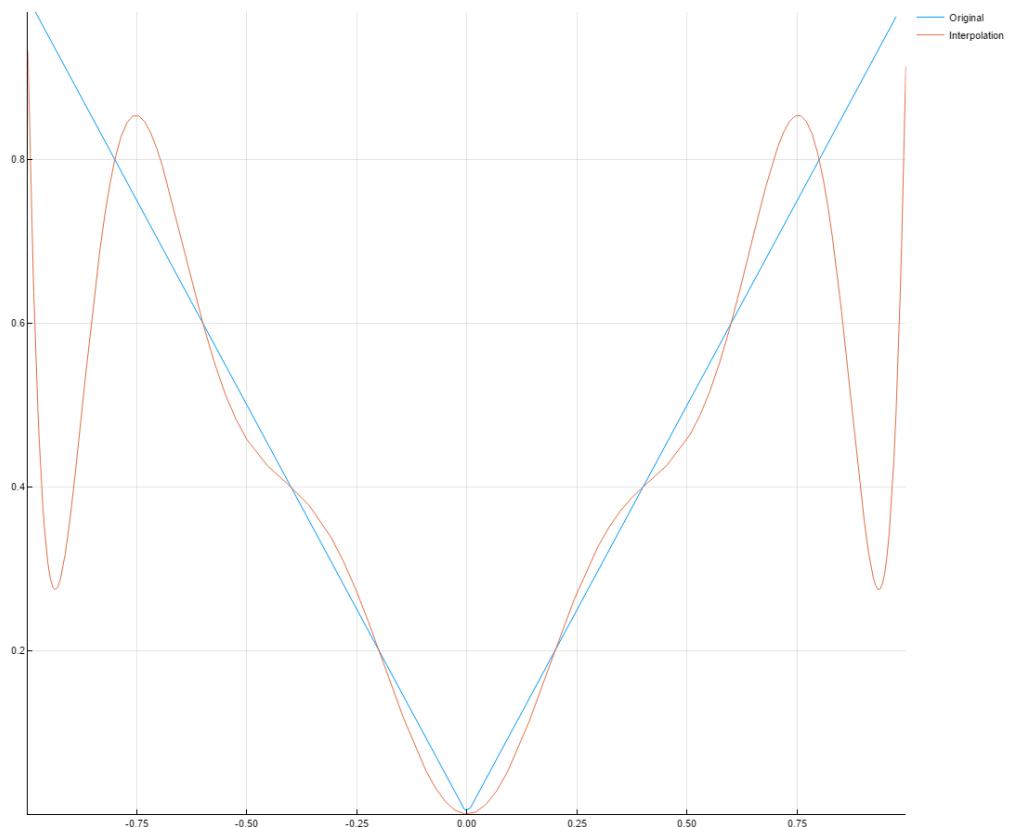
a) $|x|, [-1, 1], n = 5, 10, 15$

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

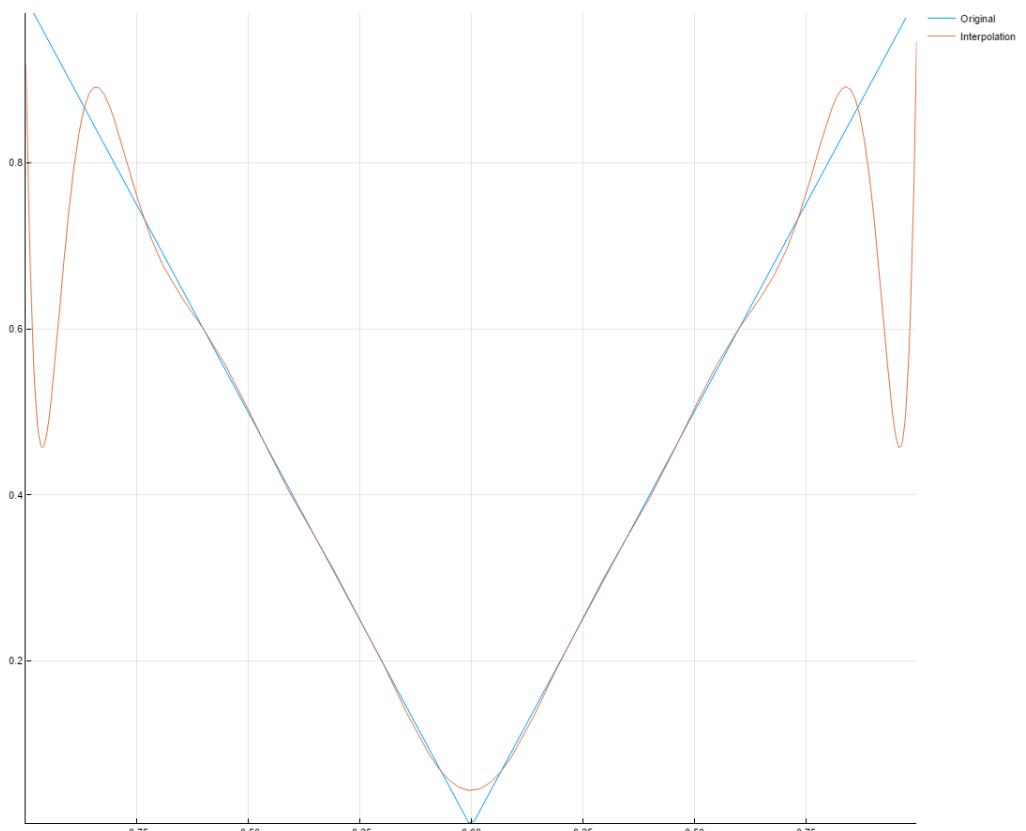
6.2 $|x|$



$n = 5$

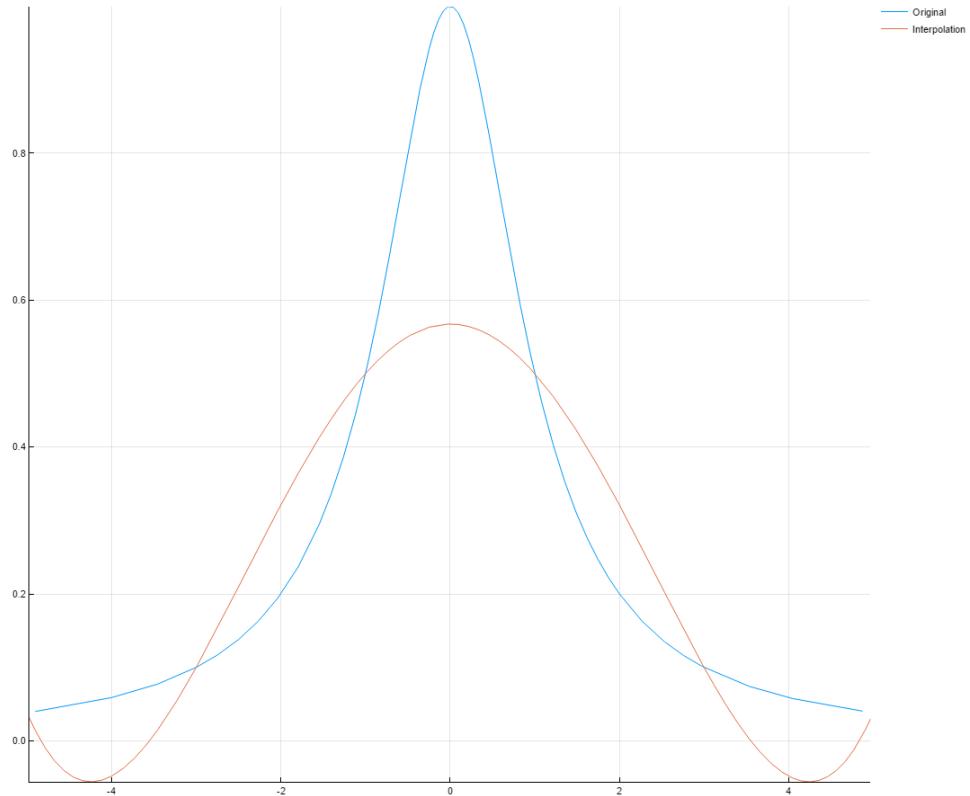


$n = 10$

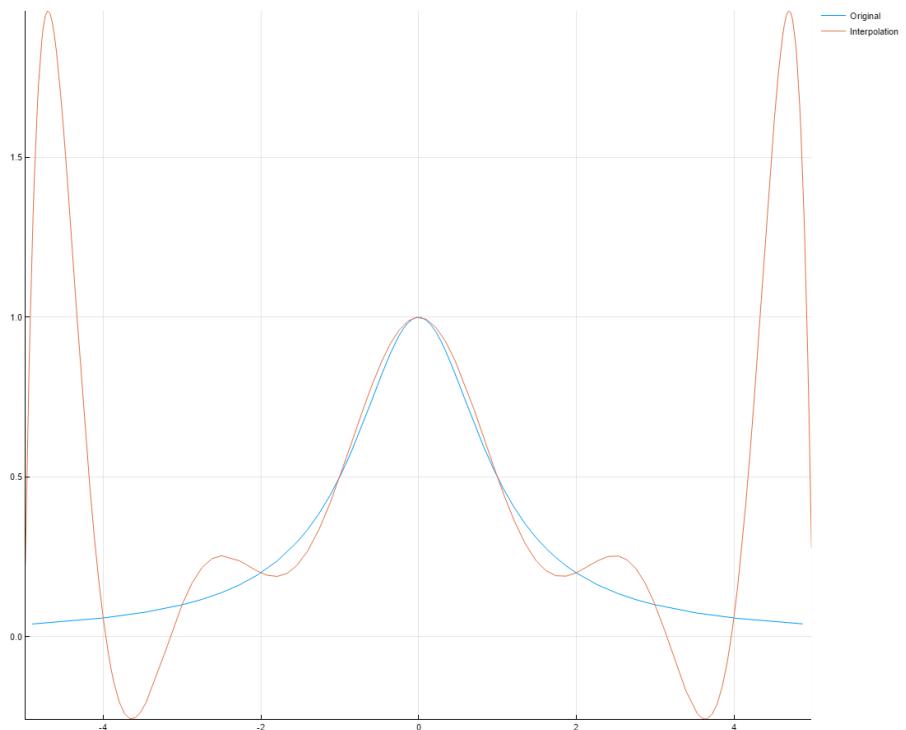


$n = 15$

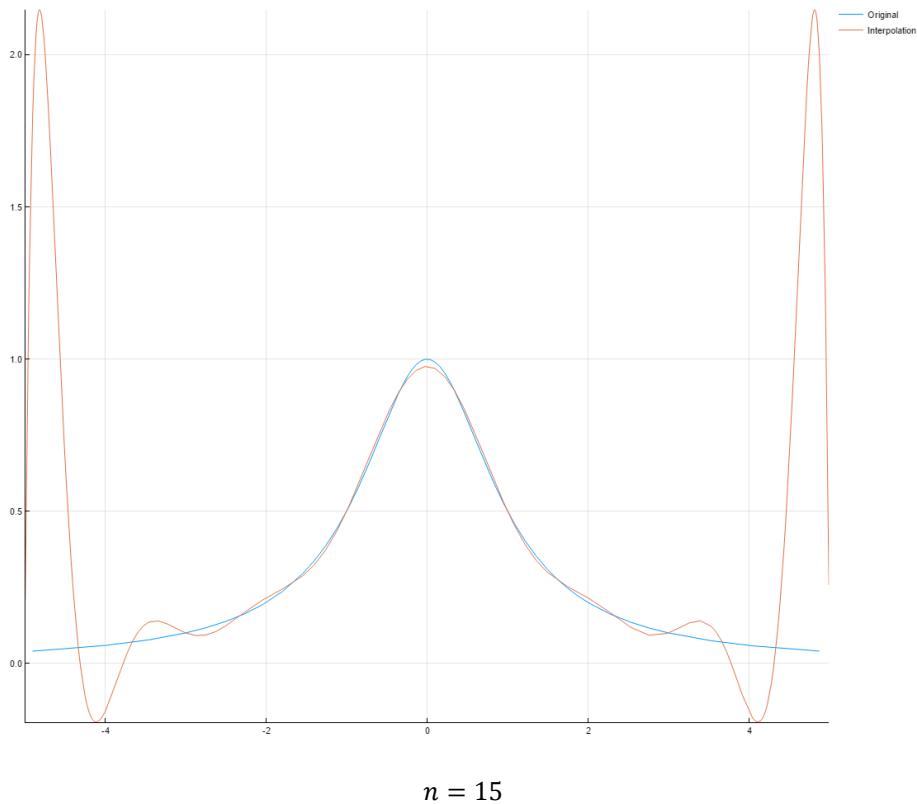
$$6.3 \frac{1}{1+x^2}$$



$$n = 5$$



$$n = 10$$



6.4 Wnioski

Intuicja nabyta dzięki zadaniu piątemu podpowiadała, że wykresy funkcji interpolacyjnej będą przypominały oryginalne. Powyższe przykłady jasno temu zaprzeczają. Największe różnice między wartościami funkcji zachodziły na krańcach badanych przedziałów.

Powodem tego stanu rzeczy jest tzw. zjawisko Rungego. Polega ono na pogorszeniu interpolacji mimo tego, że ilość węzłów rośnie. Jest to typowe dla interpolacji posiadających stałe odległości między punktami – czyli takiej, która została zastosowana. W celu przeciwdziałania temu efektowi wykorzystuje się interpolację, której węzły są gęściej porozmieszczane na krańcach interpolowanego przedziału – czyli tam, gdzie zachodzą największe różnice. Można to uzyskać między innymi dzięki zastosowaniu wielomianów Czebyszewa, a właściwie ich miejsc zerowych. W celu wygenerowania węzłów dla n -punktowej interpolacji należy znaleźć miejsca zerowe dla wielomianu n -tego stopnia.