Obliczenia naukowe
Sprawozdanie lista 4

Wojciech Wróblewski 250349

October 2020

Zad1

Opis problemu oraz terminologia

Zaimplementować funkcję obliczającą ilorazy różnicowe.

Dane

x – wektor długości n + 1, zawierający węzły $x_0, ..., x_n$.($x\,[1] = x_0, ..., x\,[n+1] = x_n$)

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

Wyniki

fx– wektor długości n + 1 zawierający obliczone ilorazy różnicowe

Rozwiązanie

Ilorazy różnicowe rzędu zerowego oraz pierwszego obliczamy następująco.
$f\,[x_0] = f\,(x_0)$
$f\,[x_0, x_1] = \frac{f[x_1]-f[x_0]}{x_1-x0}$
Ogólny wzór rekurancyjny pozwalający obliczanie ilorazów różnicowych jest postaci.
$f\,[x_i, x_{i+1}, ..., x_{i+k}] = \frac{f[x_{i+1},x_{i+2},...,x_{i+k}]-f[x_i,x_{i+1},...,x_{i+k-1}]}{x_{i+k}-x_i}$

Funkcja **ilorazyRoznicowe()**,która została zaimplementowana w module **Lista4**, bazuje na poniższym pseudokodzie.Wszystkie obliczenia zostały przeprowadzone w arytmetyce Float64.

Algorithm 1: Pseudokod funkcji obliczającej ilorazy różnicowe.

```
   Input  : x,f
   Output: fx
 1
 2 Function ilorazyRoznicowe (x ::Vector, f ::Vector)
 3
 4 j = 1
 5 while j ⩽ length(x) do
 6 |   fx[j] = f[j]
 7 |   j++
 8 end
 9
10 j = 2
11 while j ⩽ length(x) do
12 |
13 |   k = length(x)
14 |   while k ⩾ j do
15 |   |
16 |   |   fx[j] = (fx[k]−fx[k−1]) / (x[k]−x[k−j+1])
17 |   |   k - -
18 |   end
19 |   j++
20 end
21
22 return fx;
```

Zad2

Opis problemu

Napisać funkcję obliczającą wartość wielomianu interpolacyjnego stopnia n w postaci Newtona tj. Nn(x), w punkcie x = t. Rozwiązanie należy oprzeć na podstawie uogólnionego algorytmu Hornera.

Dane

x – wektor długości n + 1 zawierający węzły $x_0, ..., x_n$
($x\,[1] = x_0, ..., x\,[n+1] = x_n$)

fx – wektor długości n + 1 zawierający ilorazy różnicowe

t – punkt, w którym należy obliczyć wartość wielomianu

Wyniki

nt – wartość wielomianu w punkcie t

Rozwiązanie

Postać Newtona wzoru interpolacyjnego możemy zapisać jako:
$p(x) = \sum_{k=0}^{n} c_k q_k(x) = \sum_{k=0}^{n} f\,[x_0, x_1, ..., x_k] \prod_{j=0}^{k-1}(x - x_j)$
Obliczenie wartości wieomianu umożliwi nam podstawienie wykorzystujące uogólniony algorytm Hornera.Wykorzystujemy :
$w_n := f\,[x_0, x_1, ..., x_n]$
$w_k(x) := f\,[x_0, x_1, ..., x_k] + (x - x_k)w_{k+1}(x)$ gdzie$k \in \{n-1, ..., 0\}$
$N_n(x) := w_0(x)$
Wobec tego, aby uzyskać wartość wielomianu w postaci Newtona w punkcie t (tj. $N_n(t)$), będziemi iterowali w pętli po wartościach od w_n do w_0. Na poniższym pseudokodzie obserwujemy, że wykonamy mniej niż n iteracji w pętli, operacje podczas każdej z iteracji wykonamy w czasie O(c) gdzie jest stałą. Wobec tego zaproponowany algorytm wykona się w czasie O(n). Funkcja **warNewton()**,która została zaimplementowana w module **Lista4**, bazuje na poniższym pseudokodzie.Wszystkie obliczenia zostały przeprowadzone w arytmetyce Float64.

Algorithm 2: Pseudokod funkcji obliczającej wartości wielomianu interpolacyjnego w postaci Netwona.

Input : x,fx,t
Output: nt

Function warNewton (x ::Vector, fx ::Vector, t ::Float)

$nt = fx\,[length(x)]$

j =length(x) − 1
while $j \geqslant 1$ **do**
  $nt = fx\,[j] + (t - x\,[j]) * nt$
end

return nt

Zad3

Opis problemu

Znając współczynniki wielomianu interpolacyjnego w postaci Newtona
$c_0 = f\,[x_0]$,
$c_1 = f\,[x_0, x_1]$,
$c_2 = f\,[x_0, x_1, x_2]$,
.
.
.
$c_n = f\,[x_0, ..., x_n]$
oraz węzły $x_0, x_1, x_2, ..., x_n$.Należy napisać funkcję obliczającą, w czasie $O(n^2)$, współczynniki jego postaci naturalnej $a_0, ..., a_n$ tzn. $a_n x^n + a_{n-1} x^{n-1} + ... + a_1 x + a_0$

Dane

x – wektor długości n + 1 zawierający węzły $x_0, ..., x_n$
$x\,[1] = x_0$,
.
.
.
$x\,[n+1] = x_n$

fx – wektor długości n + 1 zawierający ilorazy różnicowe

Wyniki

a – wektor długości n + 1 zawierający obliczone współczynniki postaci naturalnej
$a[1] = a_0$,
$a[2] = a_1, ..., a[n] = a_{n1}, a[n+1] = a_n$.

Rozwiązanie

Algorithm 3: Pseudokod funkcji obliczającej wartości współczynników wielomianu interpolacyjnego w postaci naturalnej

```
   Input  : x,fx
   Output: a
1
2  Function naturalna (x::Vector, fx::Vector)
     j = 1
     while j ⩽ length(x) do
3  |    a[j] = undefined
4  end
5  a[length(x)] = fx[length(x)]
6
7  j = length(x) − 1
8  while  j ⩾ 1 do
9  |
10 |    a[j] = fx[j] − a[j + 1] ∗ x[j]
   |
11 |    k=j+1
   |     while  k ⩽ length(x) - 1 do
12 |    |    a[k] = a[k] − a[k + 1] ∗ x[j]
   |    |
13 |    end
14 end
15 return a
```

W celu znalezienia współczynników wielomianu interpolacyjnego w postaci naturalnej ponownie korzystamy z uogólnionego algorytmu Hornera. Zauważamy, że współczynnik a_n (współczynnik przy najwyższej potędze w postaci Newtona) jest równy współczynnikowi c_n. Wobec tego rozpoczynamy algorytm od przypisania danych wartości, a nastepnie wykorzystując 2 pętle obliczamy wartości a_i, zależne od współczynników a_{i+1}. W pierwszej pętli iterujemy generujemy kolejne współczynniki a_i, a w pętli wewnętrznej doprowadzamy aktualny stan wielomianu do postaci naturalnej. Każda z pętli wykona się maksymalnie n razy, a operacje wykonywane podczas każdej z iteracji wykonająsię w czasie

O(c) gdzie c to stała. Wobec tego cały algorytm będzie miał złożoność czasową $O(n^2)$.
Funkcja **naturalna()**,która została zaimplementowana w module **Lista4**, bazuje na powyższym pseudokodzie.Wszystkie obliczenia zostały przeprowadzone w arytmetyce Float64.

Zad4

Opis problemu

Celem zadania jest napisanie funkcji, która zinterpoluje zadaną funkcję f(x) w przedziale domkniętym [a, b] za pomocą wielomianu interpolacyjnego stopnia n w postaci Newtona. Następnie narysuje wielomian interpolacyjny i interpolowaną funkcję. W zadaniu wykorzystujemy algorytmy **ilorazyRoznicowe()** oraz **warNewton()** z poprzednich zadań.

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].

Rozwiązanie

Algorithm 4: Pseudokod funkcji, która interpoluje zadaną funkcję f(x) w przedziale [a, b] za pomocą wielomianu interpolacyjnego stopnia n w postaci Newtona i wyświetla graficzną reprezentację interpolacji.

Input : f,a,b,n
Output: reprezentacja graficzna interpolacji

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

//**constants definitions**

EXPAND_CONST= 25
MAX_NODES = n + 1
LIMIT = EXPAND_CONST * MAX_NODES

//**vector definitions**

x :: Vector
y :: Vector
fx :: Vector

data_x :: Vector
data_y :: Vector
data_y_interpolated :: Vector

kh =0
h = $\frac{b-a}{MAX\_NODES-1}$

i = 1
while $i \leqslant MAX\_NODES$ **do**
    x[i] = $a + kh$
    y[i] = $f(x\,[i])$
    kh += h
    i ++
end

kh = 0
fx = ilorazyRoznicowe(x, y)
h = $\frac{b-a}{LIMIT-1}$

i = 1
while $i \leqslant LIMIT$ **do**
    data_x[i] = $a + kh$
    data_y_interpolated[i] = $warNewton(x, fx, data\_x\,[i])$
    data_y[i] = $f(data\_x\,[i])$
    kh += h
    i++
end
draw(data_x,data_y,data_y_interpolated)

W rozwiązaniu podczas interpolacji używamy węzłów równoodległych, tj.
$x_k = a + kh, h = \frac{b-a}{n}, k \in \{0, 1, ..., n\}$
Początkowo definiujemy stałe oraz wektory. Maksymalna liczba węzłów to n+1 zdefiniowana jako MAX_NODES.Definiujemy wektor węzłów interpolacji. Odległość między węzłami to$\frac{b-a}{n}$. Następnie obliczamy wartości funkcji w węzłach i przechowujemy je w wektorze. Kolejnym krokiem jest wykorzystanie funkcji **ilorazyRoznicowe()** w celu wyznaczenia ilorazów różnicowych dla stowrzonych węzłów. Definiujemy zmienną LIMIT przechowującą liczbę punktów dla których tworzone będą wykresy.Stałą limit możemy zmieniać, w celu obserwacji zachowania interpolacji dla liczby próbkowań. Na podstawie węzłów ilości testów oraz funkcji **warNewton()** generujemy wektory przechowujące rzeczywiste jak i interpolowane wartości funkcji. Otrzymane wyniki reprezentujemy graficznie za pomocą pakietu *Plots*.

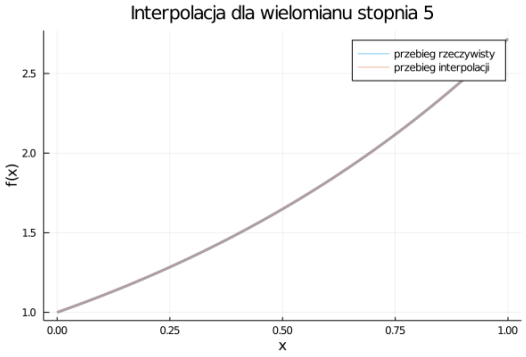
Zad5

Opis problemu

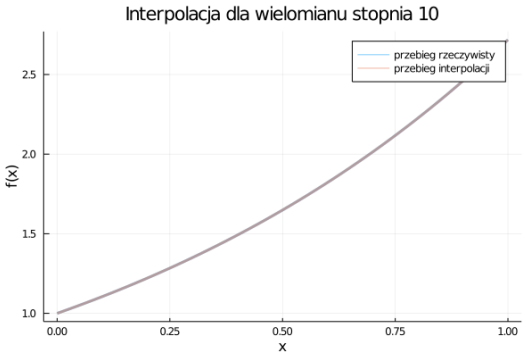
Przetestować funkcję rysujNnfx(f,a,b,n) na następujących przykładach gdzie:

- Funkcja zdefiniowana jest jako e^x. Zadany przedział domknięty to $[0, 1]$. Stopnie n wielomianu do próbkowania $\in \{5, 10, 15\}$.
- Funkcja zdefiniowana jest jako $x^2 sin(x)$. Zadany przedział domknięty to $[-1, 1]$. Stopnie n wielomianu do próbkowania $\in \{5, 10, 15\}$.

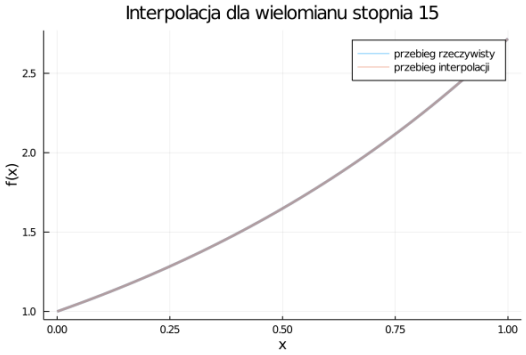
Wyniki



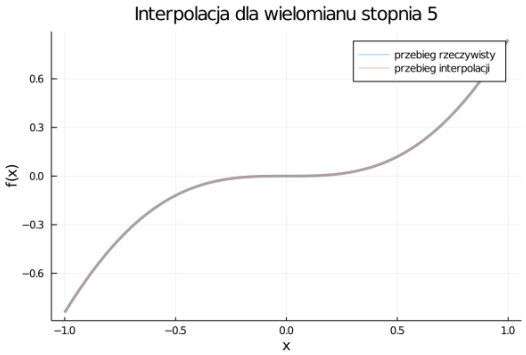
Rysunek 1: interpolacja dla funkcji e^x dla wielomianu stopnia 5



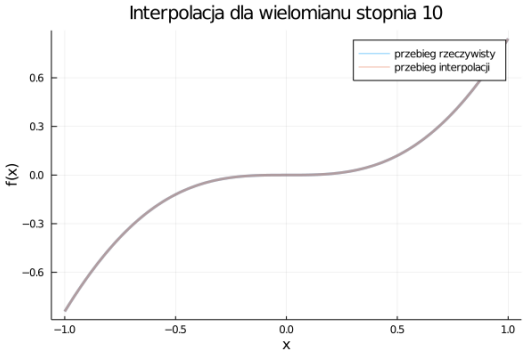
Rysunek 2: interpolacja dla funkcji e^x dla wielomianu stopnia 10



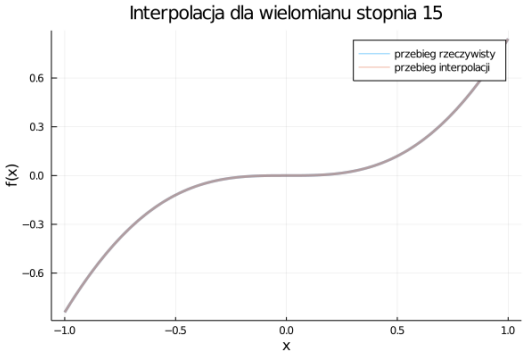
Rysunek 3: interpolacja dla funkcji e^x dla wielomianu stopnia 15



Rysunek 4: interpolacja dla funkcji $x^2 sin(x)$ dla wielomianu stopnia 5



Rysunek 5: interpolacja dla funkcji $x^2 sin(x)$ dla wielomianu stopnia 10



Rysunek 6: interpolacja dla funkcji $x^2 sin(x)$ dla wielomianu stopnia 15

Wnioski

Interpolacja przeprowadzona na powyższych funkcjach dla węzłów równoodległych i ich stałego zagęszczenia jest dokładna. Błąd, czyli potencjane odchylenia interpolacji względem funkcji bazowej jest niezauważalny, pokazuje to pokrycie przebiegu rzeczywistego z przebiegiem interpolacji. Zauważamy, że dla zadanych przykładów interpolacja już dla wielomianu 5 stopnia zwraca dokładne przybliżenie przebiegu funkcji bazowej.

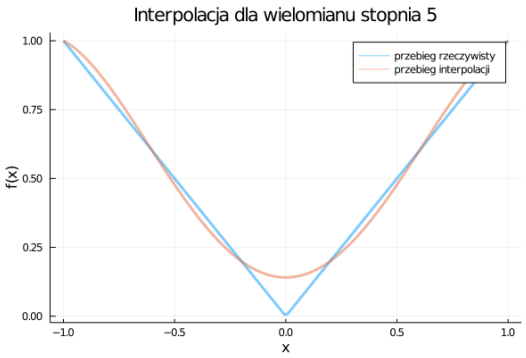
Zad6

Opis problemu

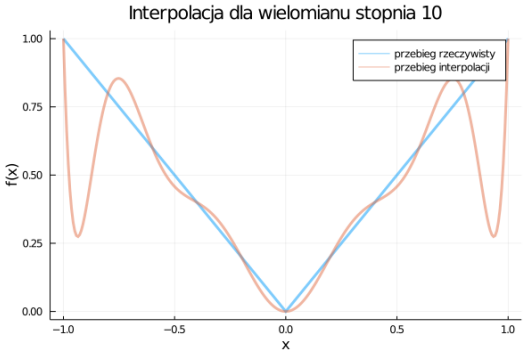
Przetestować funkcję rysujNnfx(f,a,b,n) na następujących przykładach gdzie:

- Funkcja zdefiniowana jest jako $|x|$. Zadany przedział domknięty to $[-1, 1]$. Stopnie n wielomianu do próbkowania $\in \{5, 10, 15\}$.
- Funkcja zdefiniowana jest jako $\frac{1}{1+x^2}$. Zadany przedział domknięty to $[-5, 5]$. Stopnie n wielomianu do próbkowania $\in \{5, 10, 15\}$.

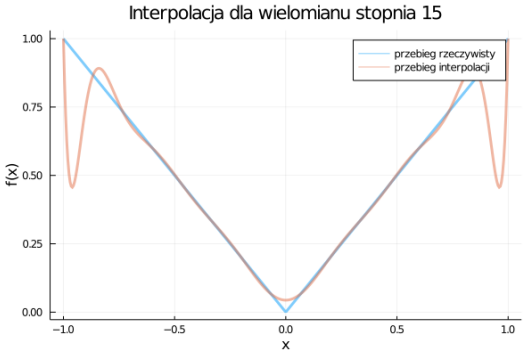
Wyniki



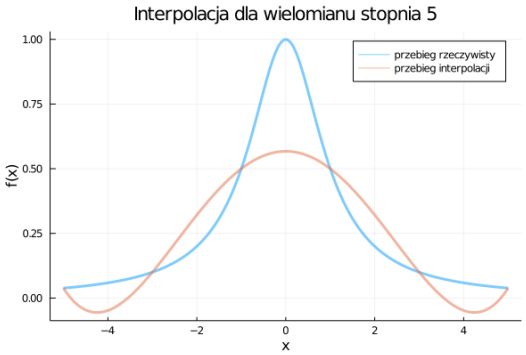
Rysunek 7: interpolacja dla funkcji $|x|$ dla wielomianu stopnia 5



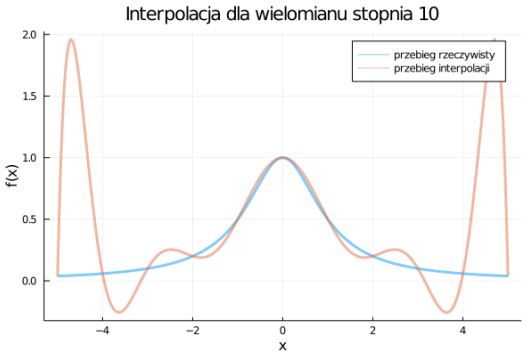
Rysunek 8: interpolacja dla funkcji $|x|$ dla wielomianu stopnia 10



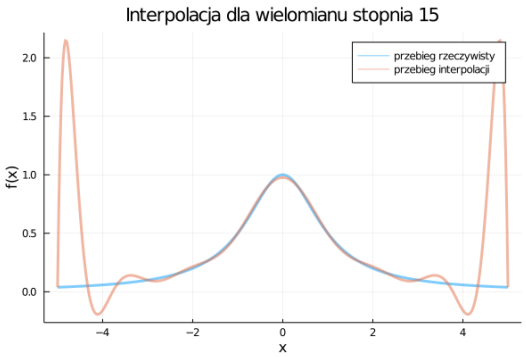
Rysunek 9: interpolacja dla funkcji $|x|$ dla wielomianu stopnia 15



Rysunek 10: interpolacja dla funkcji $\frac{1}{1+x^2}$ dla wielomianu stopnia 5



Rysunek 11: interpolacja dla funkcji $\frac{1}{1+x^2}$ dla wielomianu stopnia 10



Rysunek 12: interpolacja dla funkcji $\frac{1}{1+x^2}$ dla wielomianu stopnia 15

Wnioski

Badane przykłady pokazują, że zastosowane metody interpolacji z wykorzsytaniem węzłów równoodległych nie jest idealnym i ogólnym sposobem na przybliżanie każdej funkcji. Istanieją funkcje, dla których równoodległość rozmieszczenia węzłów zamiast zwiększać dokładność wraz ze wzrostem wielomianu ją pogarsza. Obserwujemy to na powyższych wykresach. Amplituda "oscylacji" wielomianu interpolacyjnego w pobliżu końców przedziału powiększa się ze wzrostem jego stopnia n. Błąd aproksymacji na krańcach przedziałów nie maleje, lecz wzrasta. Obserwowanego zjawisko to tzw. efekt Runge'ego.Pomocne w zniwelowaniu efektu jest użycie węzłów rozmieszczonych nierównomiernie. W takich przypadkach pomocne jest zastosowanie interpolacji lokalnej (na kilku podprzedziałach) wielomianami niskiego stopnia i późniejszym połączeniem ich poprzez sklejenie. Innym sposobem jest zastosowanie węzłów Czebyszewa (pierwiastów wielomianów Czebyszewa) jako węzły w interpolacji wielomianowej,ponieważ miejsca zerowe wielomianów Czebyszewa zagęszczają się ku krańcom przedziału,co pozwala lepiej związać wielomian zapobiegając oscylacjom na krańcach przedziałów.