Obliczenia Naukowe

Lista IV

Kacper Pieniążek, 236606

Wtorek TP 15:15

9 grudnia 2018

Spis treści

1 Zadanie 1

1.1 Cel zadania

Celem zadania jest napisanie funkcji obliczającej ilorazy różnicowe $f[x_0], f[x_0, x_1], \cdots, f[x_0, x_1, \cdots, x_n]$ funkcji interpolowanej f. Zużycie pamięci jest $O(n)$

1.2 Rozwiązanie

Do obliczenia ilorazów różnicowych funkcji f wykorzystano wzór:

$$f[x_0, x_1, \cdots, x_n] = \frac{f[x_1, \cdots, x_n] - f[x_0, \cdots, x_{n-1}]}{x_n - x_0}, f[x_i] = f(x_i)$$

Ilorazy różnicowe przechowywane są w tablicy rozmiaru $n + 1$, kolejne obliczane wartości zastępują poprzednie; do obliczenia ilorazu różnicowego potrzebne są tylko dwie poprzednie wartości.
Dane:
x - wektor węzłów funkcji f
f - wektor wartości kolejnych węzłów f

Algorithm 1 Obliczanie ilorazów różnicowych

function DIVIDEDDIFFERENCES(x, f)
    $len \leftarrow length(fx)$
    for k = 1 to $len - 1$ **do**
        for i = len downto $(k + 1)$ **do**
            $fx[i] \leftarrow (fx[i] - fx[i-1])/(x[i] - (x[i-k]))$
        end for
    end for
    return fx

1.3 Wnioski

Obliczenie ilorazów różnicowych odbywa się liniowo względem zarówno czasu, jak i pamięci. Pomaga w tym fakt, że każdy kolejny iloraz potrzebuje dokładnie dwóch poprzednich wartości, poza przypadkami bazowymi.

2 Zadanie 2

2.1 Cel zadania

Celem 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.

2.2 Rozwiązanie

Algorytm działa w czasie $O(n)$, do jego realizacji wykorzystano język Julia. Zaimplementowano uogólniony algorytm Hornera dla $k = n - 1, \cdots, 0$:

$$w_n(x) = f[x_0, x_1, \cdots, x_n]$$

$$w_k(x) = f[x_0, x_1, \cdots, x_k] + (x - x_k) \cdot w_{k+1}(x)$$

$$N_n(x) = w_0(x)$$

Dane:
x - wektor kolejnych węzłów

fx - wektor zawierający ilorazy różnicowe funkcji f
t - punkt, w którym należy obliczyć wartość wielomianu

Algorithm 2 Obliczanie wielomianu w punkcie

```
function valNewton(x, fx, t)
    result ← last(fx)
    for i = length(fx) − 1 downto 1 do
        result ← result ∗ (t − x[i]) + fx[i]
    end for
    return result
```

2.3 Wnioski

Podczas obliczeń wygodnie jest iterować po wielomianie ód końca", dzięki temu obliczenia są $O(n)$. Algorytm Hornera optymalizuje ilość potrzebnych działań do wyliczenia wartości; $n-1$ mnożeń oraz $n-1$ operacji dodawania.

3 Zadanie 3

3.1 Cel zadania

Celem zadania jest wyznaczenie współczynników postaci naturalnej wielomianu interpolacyjnego w postaci Newtona tj. wyznaczenie $a_0, \cdots, a_n$ w postaci $a_n x_n + \cdots + a_1 x_1 + a_0$. Funkcja ma działać w czasie $O(n^2)$.

3.2 Rozwiązanie

Rozwiązanie w języku Julia, podejście jest podobne do poprzedniego zadania.

Algorithm 3 Obliczanie współczynników a_i

```
function natural(x,fx)
    n ← length(x)
    result ← Array(n)
    for i = n downto 1 do
        result[i] ← fx[i]
        for j = i to (n − 1) do
            result[j]− = x[i] ∗ result[j + 1]
        end for
    end for
    return result
```

3.3 Wnioski

Algorytm ma złożoność $O(n^2)$, ponieważ dla każdego ilorazu różnicowego musimy obliczać a_i na podstawie poprzednich wyników. Tak samo, jak podczas tworzenia wielomianu interpolacyjnego w postaci Newtona, najwygodniejsze jest działanie ód końca".

4 Zadanie 4

4.1 Cel zadania

Celem zadania jest napisanie funkcji, które zinterpoluje zadaną funkcję $f(x)$ za pomocą wielomianu interpolującego w postaci Newtona. Następnie, funkcja ma narysować $f(x)$ oraz wielomian interpolujący.

4.2 Rozwiązanie

Do rozwiązania wykorzystano PyPlot w języku Julia. Rozwiązanie polega na wykorzystaniu funkcji zaimplementowanych w poprzednich zadaniach. Szczegóły implementacji rysowania dla funkcji zostaną pominięte w sprawozdaniu.
Do obliczania ilorazów różnicowych wykorzystano węzły równoodległe. Wielomian jest wyznaczany za pomocą funkcji **dividedDifferences** i **valNewton**.
Dane:
f - funkcja $f(x)$ do zinterpolowania
a, b - przedział interpolacji n - stopień wielomianu interpolacyjnego

5 Zadanie 5

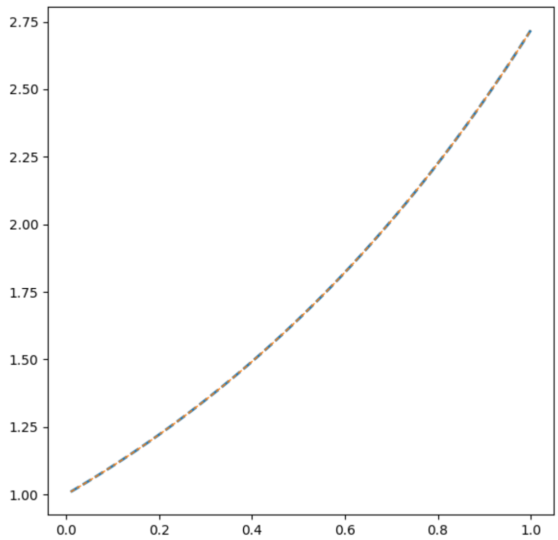
5.1 Cel zadania

Zadanie polega na przetestowaniu funkcji rysującej wielomian interpolujący funkcję $f(x)$ na podanych przykładach.

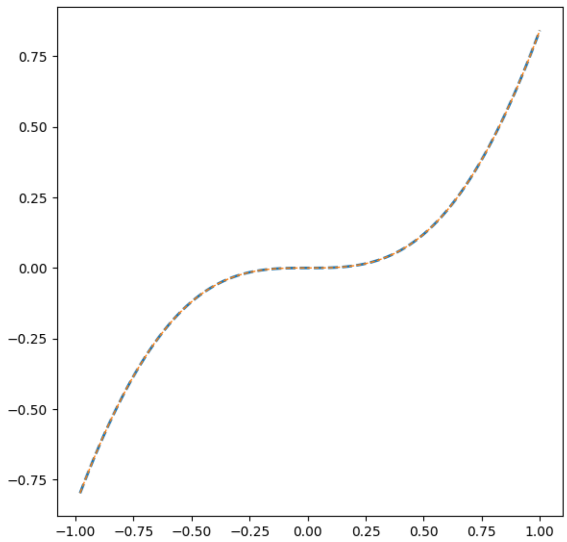
5.2 Rozwiązanie

W celu realizacji zadania, wywołano funkcję z poprzedniego zadania z odpowiednimi parametrami.
Kolor niebieski - wielomian interpolujący
Kolor pomarańczowy - funkcja $f(x)$

f(x) $= e^x$
(a, b) = (0,1)
n = 5, 10, 15



f(x) $= x^2 sin(x)$
(a, b) = (-1,1)
n = 5, 10, 15



Dla narysowanych wykresów $n = 5$

5.3 Wnioski

Jak widać na zaprezentowanych wykresach, wartości funkcji dokładnie pokrywają się z wartościami wielomianu interpolującego. Co więcej, dla wszystkich n wykresy pozostają takie same; wielomian dobrze interpoluje funkcje $f(x)$, zachowuje ich własności na zadanych odcinkach.

6 Zadanie 6

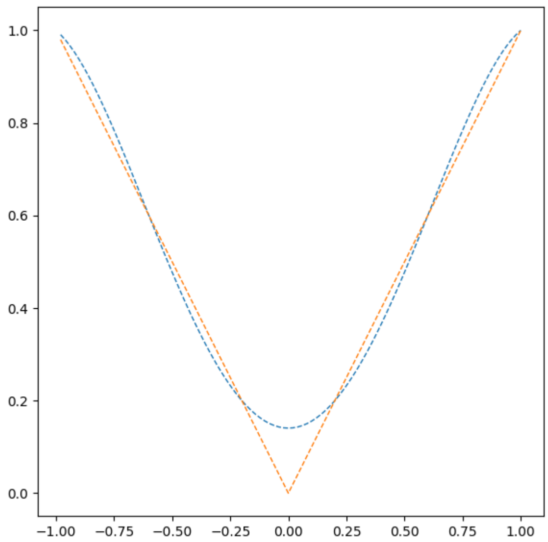
6.1 Cel zadania

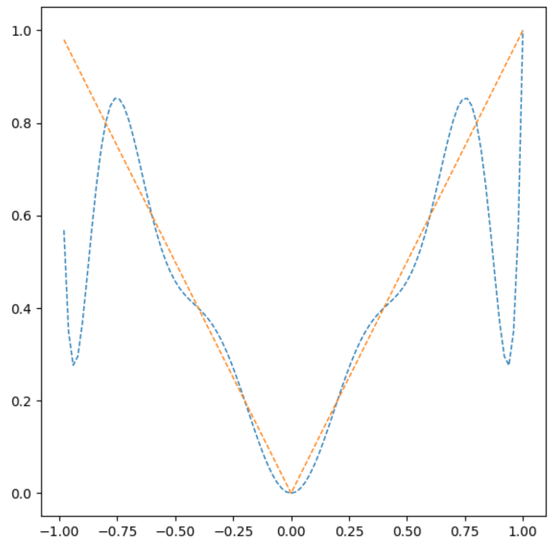
Celem zadania jest przetestowanie funkcji rysującej wielomian interpolujący funkcję $f(x)$ na podanych przykładach oraz zbadanie zjawiska rozbieżności.

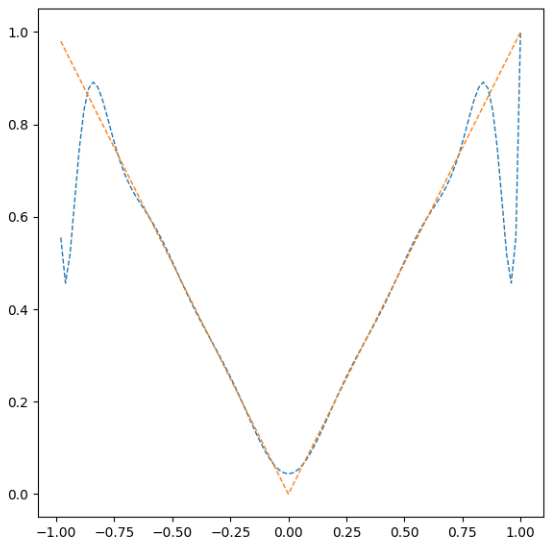
6.2 Rozwiązanie

W celu realizacji zadania, wywołano funkcję z poprzedniego zadania z odpowiednimi parametrami.
Kolor niebieski - wielomian interpolujący
Kolor pomarańczowy - funkcja $f(x)$

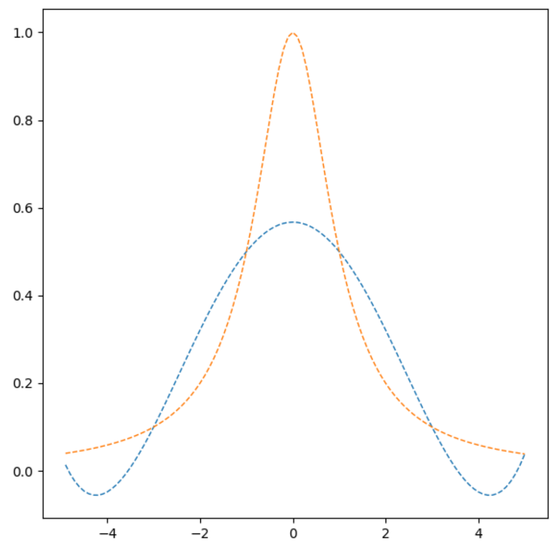
f(x) $= |x|$
(a, b) = (-1,1)
n = 5, 10, 15

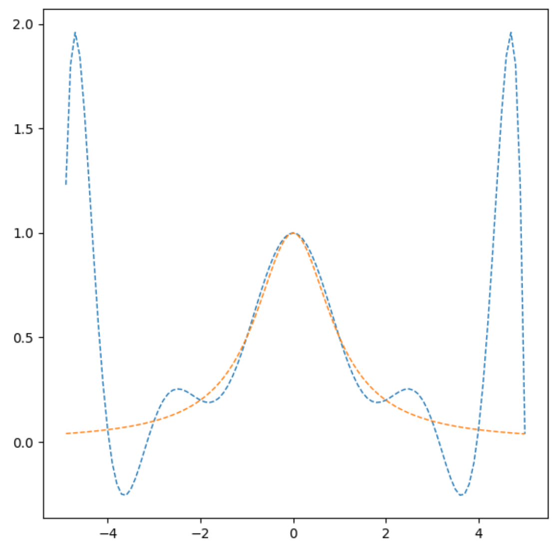


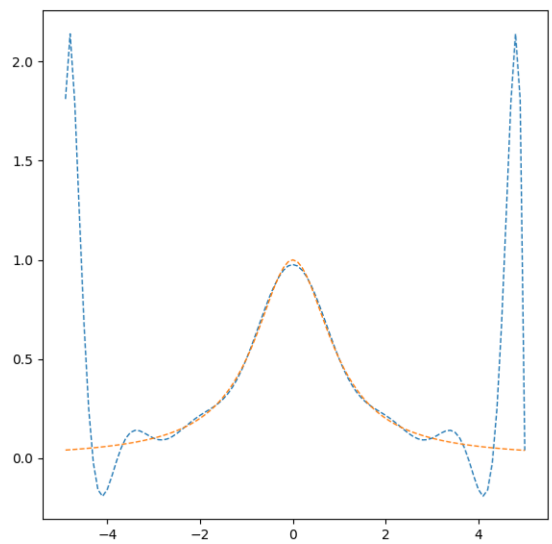




$\mathbf{f(x)} = \frac{1}{1+x^2}$
(a, b) = (-5,5)
n = 5, 10, 15







6.3 Wnioski

Dla niewielkich n, wielomian interpolujący jest niedokładny. Wyższe wartości n w środku przedziału są bardziej dokładnie, jednak coraz mocniej się rozbiegają na brzegach przedziałów. Zaobserwowane zjawisko nazywa się Efektem Rungego. Początkowo, dla większej ilości węzłów jakość interpolacji się poprawia, jednak od pewnego momentu ulega znacznemu pogorszeniu.
Może być to spowodowane nieciągłością funkcji oraz interpolacją przy pomocy wielomianów wysokiego stopnia przy stałych odległościach węzłów.
Uniknięcie takiego efektu może być możliwe, jeśli zastosuje się interpolację z węzłami występującymi gęściej przy granicach przedziałów.