Obliczenia Naukowe 3

Arian Herda

3 December 2023

1 Ilorazy Różnicowe

Zadanie pierwsze polegało na napisaniu funkcji wyliczającej ilorazy różnicowe oparte na podanych węzłach nie korzystając z tablicy dwuwymiarowej. Funkcja miała pobierać dwa parametry:

- x - wektor o długości $n+1$ zawierający węzły $x_0, ..., x_n$
- f - wektor o długości $n+1$ zawierający wartości interpolowanej funkcji w węzłach $f(x_0), ..., f(x_n)$

a zwracać:

- fx - wektor o długości $n+1$ zawierający obliczone ilorazy różnicowe
  $fx[1] = f[x_0]$,
  $fx[2] = f[x_0, x_1]$,
  ...
  $fx[n+1] = f[x_0, ..., x_n]$

1.1 Rozwiązanie

Ilorazy różnicowe można obliczyć na podstawie następującego wzoru rekurencyjnego:

$$f[x_0] = f(x_0)$$

$$f[x_i, x_j] = \frac{f(x_j) - f(x_i)}{x_j - x_i}$$

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

Opierając się na tym wzorze można łatwo utworzyć algorytm korzystający z tablicy dwuwymiarowej, trójkątnej. Możliwe jest jednak wykonanie obliczeń w tablicy jednowymiarowej. Wystarczy zacząć od tablicy wypełnionej wartościami węzłów fukcji, a przy kolejnych iteracjach "rzędów"tablicy dwuwymiarowej aktualizować obliczone ilorazy przeliczając je kolejny raz od dołu. W ten sposób uwzględniają one wcześniej wyliczone ilorazy różnicowe, od których są zależne. Oto pseudokod tego sposobu:

Algorithm 1 Ilorazy Różnicowe

```
function ilorazyRoznicowe(x̂, f̂)
    ĉ ← f̂
    n ← length of x̂
    for j = 1, ..., n do
        for i = n, ..., j do
            c_i ← (c_i − c_{i−1}) / (x_i − x_{i−j})
        end for
    end for
    return ĉ
end function
```

2 Uogólniony Algorytm Hornera

Zadanie drugie polegało na napisaniu funkcji wyliczającej wartość wielomianu interpolacyjnego stopnia n w postaci Newtona w podanym punkcie za pomocą uogólnionego sposobu Hornera w czasie liniowym. Wartością takiej funkcji w punkcie x jest:

$$N_n(x) = \sum_{i=0}^{n} q_i \prod_{j=0}^{i-1}(x - x_j)$$

gdzie q_i jest ilorazem różnicowym a x_j jest węzłem interpolacji. Funkcja miała przyjmować:

- x - wektor o długości $n+1$ zawierający węzły $x_0, ..., x_n$
- f - wektor o długości $n+1$ zawierający ilorazy różnicowe
- t - punkt, w którym należy obliczyć wartość wielomianu

a zwracać:

- nt - wartość wielomianu w punkcie t

2.1 Rozwiązanie

Powyższą zależność można przedstawić używając uogólnionych wzorów Hornera

$$w_n(x) = f[x_0, ..., x_n]$$

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

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

Na podstawie powyższych wzorów tworzymy algorytm o podanym pseudokodzie:

Algorithm 2 Uogólniony Schemat Hornera

```
function warNewton(x̂, ĉ, t)
    n ← length of x̂f
    nt ← c_n
    for i = n − 1, . . . , 0 do
        nt ← c_i + (t − x_i) · nt
    end for
    return nt
end function
```

3 Postać Naturalna

Zadanie trzecie polegało na napisaniu funkcji obliczającej współczynniki postaci naturalnej wielomianu interpolacyjnego w postaci Newtona. Funkcja za parametry miała przyjmować:

- x - wektor o długości $n+1$ zawierający węzły $x_0, ..., x_n$
- fx - wektor o długości $n+1$ zawierający ilorazy różnicowe

a zwracać:

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

3.1 Rozwiązanie

W wielomianie interpolacyjnym, dla korego ilorazy różnicowe to $c_0, c_1, ...$ współczynnikiem a_n stojącym przy najwyższej potędze, x_n, jest c_n. Opierając się na tym fakcie możemy wyliczać kolejne współczynniki zaczynając od najwyższych potęg i przy kolejnych przejściach aktualizując współczynniki przy obejmowanych aktualnie potęgach, w taki sposób aby były one na tę chwilę w postaciach naturalnych. Przy kolejnych iteracjach, aż do jedynki, będziemy wykorzystywali obliczone wcześniej współczynniki "częściowe"postaci naturalnej aby zaaktualizować je o nowe potęgi.

Pseudokod tego algorytmu prezentuje się następująco:

Algorithm 3 Postać Naturalna

```
function NATURALNA(x̂, f̂)
    n ← length of x̂
    a_n ← f_n
    for i = n − 1, . . . , 0 do
        a_i ← f_i − x_i · a_{i+1}
        for j = i + 1, . . . , n − 1 do
            a_j ← a_j − x_i · a_{j+1}
        end for
    end for
    return â
end function
```

4 Wykresy

Zadanie czwarte polegało na zaimplementowaniu funkcji umożliwiającej graficzne porównanie interpolacji, stopnia maksymalnie n, funkcji f, w przedziale $[a, b]$, z jej prawdziwym wykresem. Funkcja za parametry miała przyjmować:

- f - funkcja $f(x)$
- a, b = przedział interpolacji
- n - stopień wielomianu interpolacyjnego

Funkcja miała tworzyć obraz z graficznym porównaniem interpolacji z funkcją interpolowaną.

4.1 Rozwiązanie

Implementacja zadania polegała na równoległym rozmieszczeniu $n + 1$ węzłów oraz na obliczeniu ich wartości w funkcji. Następnie, za pomocą funkcji z zadania pierwszego, obliczane są ilorazy różnicowe, posłużą one nam do następnego kroku. Po tym, wykorzystując funkcje z zadania drugiego, analizujemy wartości wielomianu nie tylko w miejscach, gdzie mamy ustalone węzły, ale także między nimi. Aby to zrobić, dzielimy przedział na wiele punktów (najlepiej $N \cdot (n + 1)$ punktów na danym przedziale $[a, b]$, gdzie $N > 1$ jest całkowite). Dla każdego z tych punktów obliczamy wartość funkcji oraz wartość wielomianu. Ostatecznie uzyskane wyniki przedstawiamy na wykresie. W skrócie, chodzi o bardziej szczegółową analizę zachowania wielomianu w różnych punktach przedziału aby jego wykres był "płynny".

Do tworzenia wykresu użyłem pakietu PyPlot.

Pseudokod tłumaczonego algorytmu wygląda następująco:

Algorithm 4 Generator Wykresów

function RYSUJNNFX(f, a, b, n)
  $h \leftarrow \frac{b-a}{n}$
  for $k = 0, \ldots, n$ **do**
    $x_k \leftarrow a + k \cdot h$
    $y_k \leftarrow f(x_k)$
  end for
  $\hat{c} \leftarrow ilorazyRoznicowe(\hat{x}, \hat{y})$
  $pt \leftarrow N \cdot (n + 1)$
  $dx \leftarrow \frac{b-a}{pt-1}$
  for $i = 0, \ldots, pt$ **do**
    $X_i \leftarrow sa + i \cdot dx$
    $W_i = warNewton(\hat{x}, \hat{c}, X_i)$
    $F_i = f(X_i)$
  end for
  $plot(x = H, y = [W, F])$
end function

5 Zadanie 5

Zadanie 5 polegało na przetestowaniu funkcji z zadania czwartego dla dwóch przypadków:

- $f(x) = e^x$ na przedziale $[0, 1]$
- $f(x) = x^2 \cdot sin(x)$ na przedziale $[-1, 1]$

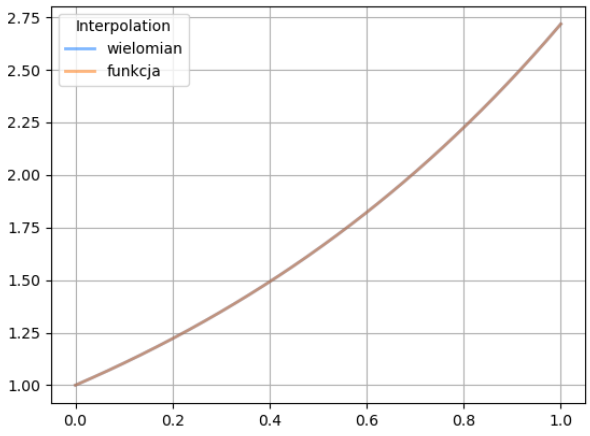
Dla wielomianów stopnia $n \in \{5, 10, 15\}$

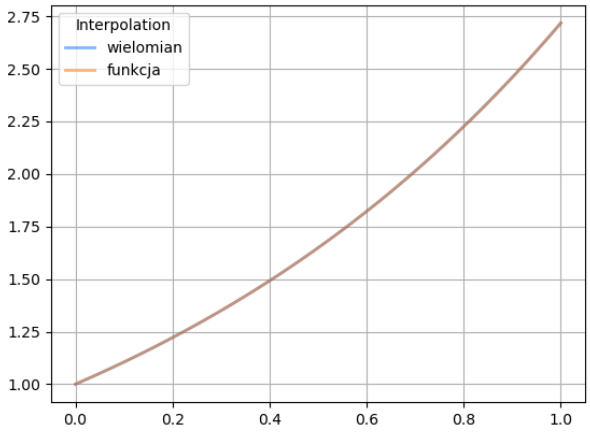
5.1 Rozwiązanie

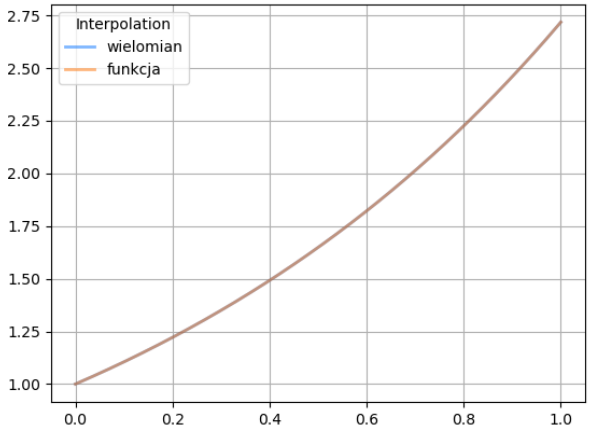
Zadanie zostało rozwiązane za pomocą funkcji opisanych w tym dokumencie zaimplementowanych w języku Julia. Rozwiązanie znajduje się w pliku *"zad5.jl"*.

5.2 Wyniki

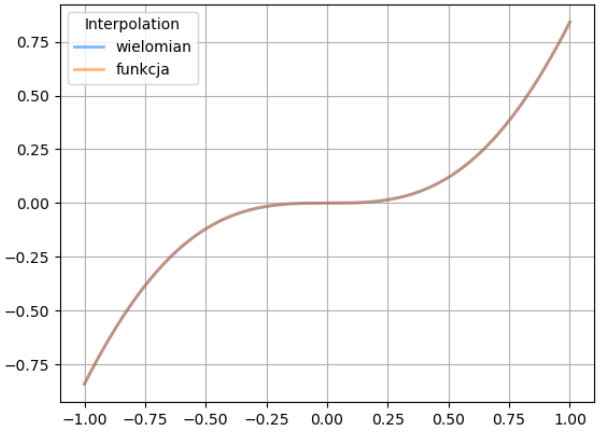
5.2.1 Funkcja e^x

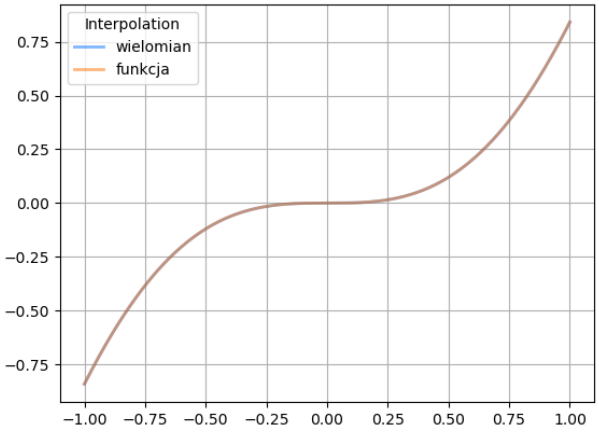


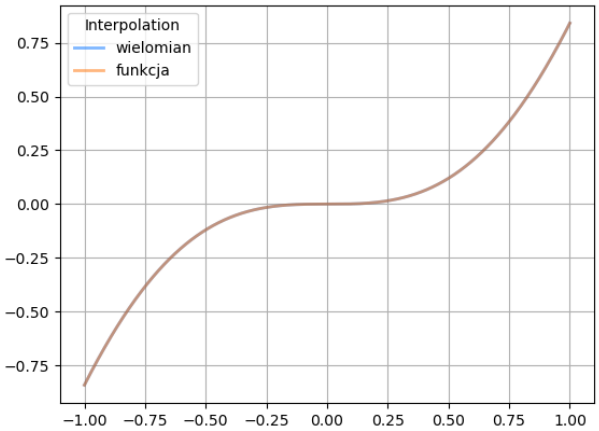




5.2.2 Funkcja $x^2 \cdot sin(x)$







Widać, że już wielomiany interpolacyjne o niskich stopniach dają bardzo dobre przybliżenie pierwotnej funkcji.

5.3 Wnioski

Interpolowanie funkcji na niewielkich zakresach oraz gdy jej pochodna nie zmienia znaku, daje bardzo dokładne przybliżenie.

6 Zadanie 6

Zadanie 5 polegało na przetestowaniu funkcji z zadania czwartego dla dwóch przypadków:

- $f(x) = |x|$ na przedziale $[-1, 1]$
- $f(x) = \frac{1}{1+x^2}$ na przedziale $[-5, 5]$

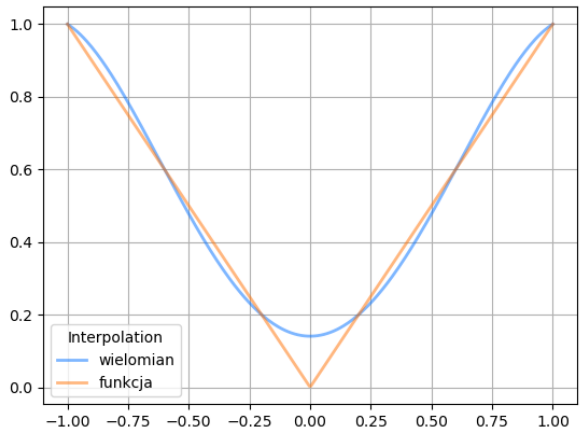
Dla wielomianów stopnia $n \in \{5, 10, 15\}$

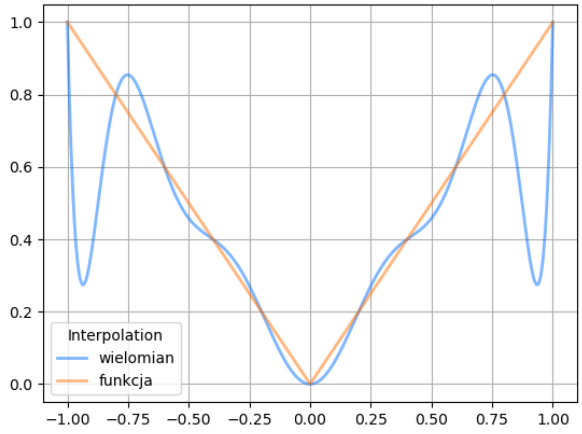
6.1 Rozwiązanie

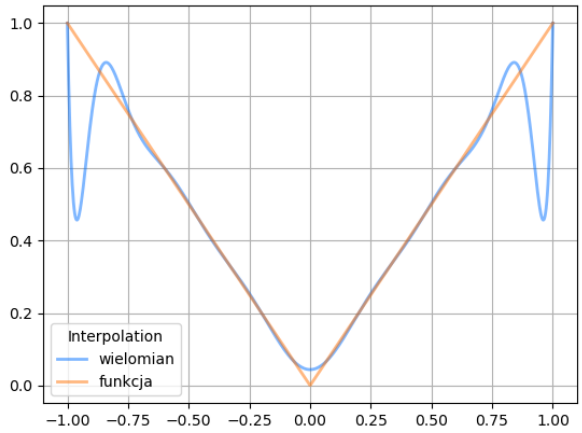
Zadanie zostało rozwiązane za pomocą funkcji opisanych w tym dokumencie zaimplementowanych w języku Julia. Rozwiązanie znajduje się w pliku *"zad6.jl"*.

6.2 Wyniki

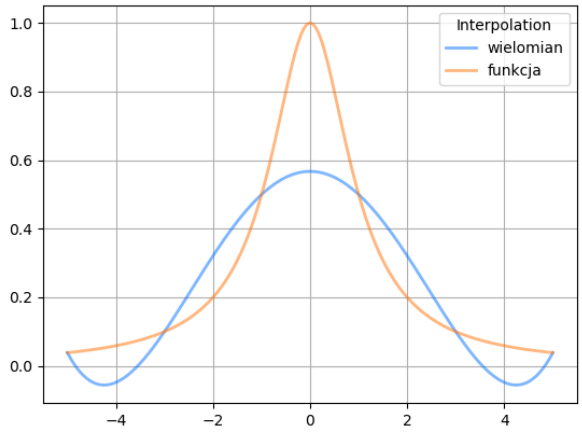
6.2.1 Funkcja $|x|$

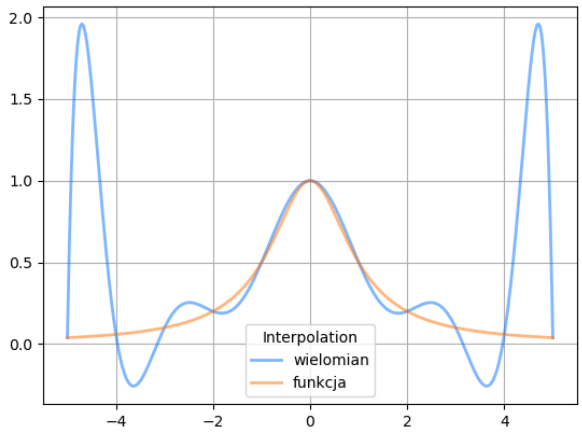


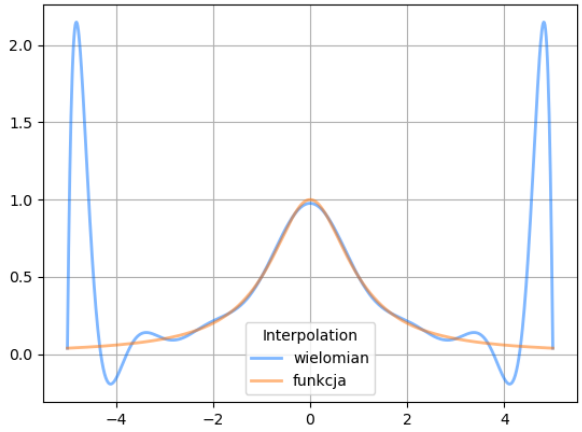

Interpolation
wielomian
funkcja
1.0
0.8
0.6
0.4
0.2
0.0
−1.00
−0.75
−0.50
−0.25
0.00
0.25
0.50
0.75
1.00


Interpolation
wielomian
funkcja
1.0
0.8
0.6
0.4
0.2
0.0
−1.00
−0.75
−0.50
−0.25
0.00
0.25
0.50
0.75
1.00

6.2.2 Funkcja $\frac{1}{1+x^2}$







Możemy zauważyć, że funkcje w tym zadaniu znacząco różnią się od funkcji z zadania piątego. Funckje nie są już gładkie i monotoniczne, to znaczy, że teraz mają ostrza (punkt w którym funkcja jest ciągłą ale nie posiada pochodnej, dodatkowo pochodna lewostronna i prawostronna mają inne znaki) oraz ich pochodne zmieniają znaki tworząc lokalne maksima i minima.

Dokładność przybliżeń w tym zadaniu już pozostawia wiele do życzenia. Wraz ze wzrostem stopnia, dokładność interpolacji w centrum przedziału rosła ale na bliżej jego końców malała.

7 Wnioski

Skuteczność wielomianu interpolacyjnego zależy w dużej mierze od funkcji, którą chcemy przy jego pomocy interpolować. Funkcje z ostrzami lub funkcje niewielomianowe z pochodną o zmieniającym się znaku są przybliżane znacząco gorzej. Precyzja interpolacji na brzegach też jest problemem o którym trzeba pamiętać. Takie wydarzenie jest spowodowane tym że wielomiany duzych stopni przybierają ogromne wartości im dalej od centrum przedziały jesteśmy zmiejszając dokładność interpolacji. Przydatna może się okazać zmiana rozmieszczenia węzłów.