Michał Budnik – Sprawozdanie 4

Wstęp

Czwarta lista pokazuje w jaki sposób należy zaimplementować algorytmy pozwalające na wyliczanie wartości wielomianu interpolacyjnego, jak również ukazuje naturę tych wielomianów.

Zadania

Zadanie 1

1.1 Opis zagadnienia

Zadanie polega na napisaniu funkcji, która w efektywny sposób (nie korzystając z tablicy dwuwymiarowej), wylicza ilorazy różnicowe na podstawie podanych węzłów, oraz odpowiadających im wartości funkcji.

1.2 Rozwiązanie

Znajomość węzłów x_i oraz wartości funkcji $f(x_i)$ (a zatem wartości ilorazów różnicowych $f[x_i] = f(x_i)$ pozwala na stworzenie trójkątnej tablicy ilorazów różnicowych.

$$
\begin{array}{ccc}
f[x_0] \rightarrow f[x_0, x_1] \rightarrow \cdots & f[x_0, x_1, \ldots, x_n] \\
f[x_1] \nearrow \rightarrow f[x_1, x_2] \nearrow \rightarrow \ldots & \\
\ldots \qquad \ldots & \\
f[x_n] \nearrow &
\end{array}
$$

Jest to możliwe poprzez wykorzystanie wzoru rekurencyjnego na iloraz różnicowy.

$$
\begin{aligned}
&(i)\ \ dla\ k = 0 : f[x_i] = f(x_i) \\
&(ii)\ dla\ k > 0 : f[x_i, x_{i+1}, \ldots, x_{i+k}] = \frac{f[x_{i+1}, x_{i+2}, \ldots, x_{i+k}] - f[x_i, x_{i+1}, \ldots, x_{i+k-1}]}{x_k - x_i}
\end{aligned}
$$

W zadaniu należy zwrócić wektor wynikowy fx w formie $fx = [f[x_0], f[x_0, x_1], \ldots, f[x_0, x_1, \ldots, x_n]]$. Można zauważyć, że w celu obliczenia każdego kolejnego elementu wektora należy użyć $n + 1 - k$ wyrażeń, które mogą być tymczasowo przechowywane w niewyznaczonych jeszcze miejscach wektora wynikowego. W ten sposób w każdej kolejnej iteracji algorytmu mogą być od końca nadpisywane odpowiednie wyliczone ilorazy różnicowe. Muszą być one nadpisywane od końca, ponieważ ostatni iloraz różnicowy jest potrzebny tylko do wyznaczenia jednego ilorazu różnicowego rzędu o jeden większego.

1.3 Algorytm

Prezentowany algorytm dla tego zadania opierający się na rozwiązaniu 1.2:

```
function ilorazyRoznicowe(x::Array{Float64}, f::Array{Float64})
    n = size(x,1)
    fx = Array{Float64}(n)
    for i = 1:n
        fx[i] = f[i]
    end
    for i = 1:n
        for j = n:-1:i+1
            fx[j] = (fx[j]-fx[j-1])/(x[j] - x[j-i])
        end
    end
    return fx
end
```

Zadanie 2

2.1 Opis zagadnienia

Zadanie polega na napisaniu funkcji obliczającej wartość wielomianu interpolacyjnego stopnia n w postaci Newtona $N_n(x)$ w punkcie $x = t$. Funkcja ma działać w czasie liniowym na podstawie uogólnionego algorytmu Hornera.

2.2 Rozwiązanie

Wzór wielomianu interpolacyjnego Newtona można przedstawić używając ilorazów różnicowych. Ukazuje on zależność wielomianu interpolacyjnego od funkcji f.

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

Takie przedstawienie powoduje, że dodanie nowych punktów (x_i, y_i) nie narusza wcześniej obliczonych współczynników. Stosując uogólniony algorytm Hornera możliwe jest wyznaczenie wartości tak wyrażonego wielomianu wyliczając odpowiednie współczynniki w_k:

$$\begin{aligned} w_n(x) &= f[x_0, x_1, \dots, x_n] \\ w_k(x) &= w_{k+1}(x - x_k) + f[x_0, x_1, \dots, x_k] \qquad k \in [n-1] \\ N_n(x) &= w_0(x) \end{aligned}$$

2.3 Algorytm

Prezentowany algorytm dla tego zadania opierający się na rozwiązaniu 2.2:

```
function warNewton(x::Array{Float64}, fx::Array{Float64}, t::Float64)
    n = size(fx, 1)
    nt = fx[n]
    for i = n-1:-1:1
        nt = fx[i] + (t - x[i]) * nt
    end
    return nt
end
```

Zadanie 3

3.1 Opis zagadnienia

Zadanie polega na stworzeniu funkcji, która znajduje współczynniki naturalne wielomianu interpolacyjnego wyrażonego w postaci Newtona.

3.2 Rozwiązanie

W celu rozwiązania tego zadania zastosowany został uogólniony algorytm Hornera. Początkowo przypisujemy a_n wartość w_n z metody Hornera. Jest to możliwe dzięki wykorzystaniu faktu, że współczynnik a_n jest równy współczynnikowi c_n z wielomianu w postaci Newtona. Dla kolejnych wartości a_i znamy zależność rekurencyjną:

$$a_i = f[x_0, x_1, \dots, x_i] - x_i w_{i+1}(x)$$

3.3 Algorytm

Prezentowany algorytm dla tego zadania opierający się na rozwiązaniu 3.2:

```
function naturalna(x::Array{Float64}, fx::Array{Float64})
    n = length(fx)
    a = Array{Float64}(n)
    a[n] = fx[n]
    for i = n-1:-1:1
        a[i] = fx[i]
        for j = i:n-1
            a[j] = a[j] - x[i] * a[j+1]
        end
    end
    return a
end
```

Zadanie 4

4.1 Opis zagadnienia

Zadanie polega na napisaniu funkcji interpolującej zadaną funkcję $f(x)$ na przedziale $[a, b]$ za pomocą wielomianu interpolacyjnego stopnia n w postaci Newtona. Następnie otrzymany wielomian interpolacyjny wraz z faktyczną funkcją powinny zostać przedstawione na wykresie. Przy interpolacji należy użyć węzłów równoodległych.

4.2 Rozwiązanie

Funkcja $rysujNnfx$ składa się z kilku kroków:

- wyznaczenie węzłów $(x_1, x_2, \dots, x_{n+1})$, każdy kolejno oddalony o wartość $\frac{b-a}{n}$,
- wyliczenie wartości funkcji $f(x)$ w wyznaczonych węzłach,
- wyliczenie ilorazów różnicowych $(f[x])$ dla wyznaczonych węzłów oraz ich wartości w funkcji,
- wyliczenie z dokładnością $20 * n$ równoległych punktów na osi x, z których wyliczane były faktyczne wartości funkcji oraz wartości wielomianu interpolacyjnego,
- rysowanie wykresu funkcji oraz wielomianu interpolacyjnego.

4.3 Algorytm

Prezentowany algorytm dla tego zadania opierający się na rozwiązaniu 4.2:

```
function rysujNnfx(f, a::Float64, b::Float64, n::Int, name::String = "plot")
    x = Array{Float64}(n+1)
    x2 = Array{Float64}(20*n+1)
    wartX = Array{Float64}(n+1)
    wartX2 = Array{Float64}(20*n+1)
    nt = Array{Float64}(20*n+1)
    for i = 1:n+1
        x[i] = a+(i-1)*((b-a)/n)
        wartX[i] = f(x[i])
    end
    fx = ilorazyRoznicowe(x, wartX)
    for i = 1:20*n+1
        x2[i] = a+(i-1)*((b-a)/(20*n))
        wartX2[i] = f(x2[i])
        nt[i] = warNewton(x, fx, x2[i])
    end
    plot(x2, [wartX2, nt])
    savefig("$name.png")
end
```

Zadanie 5

5.1 Opis zagadnienia

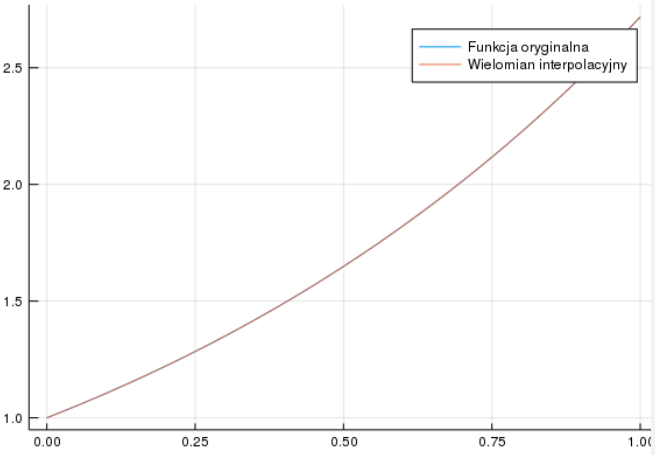
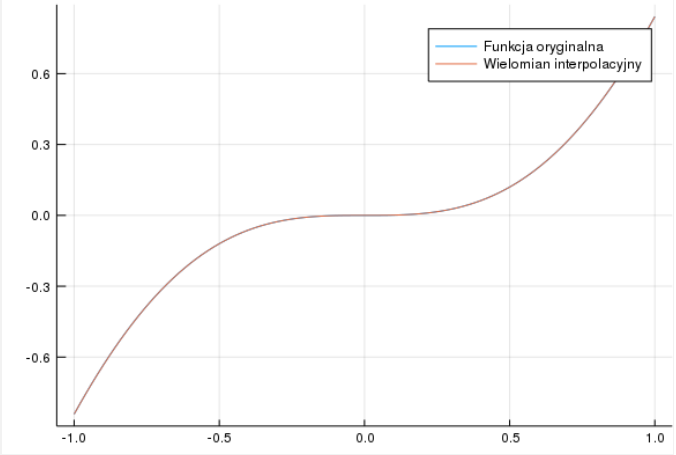
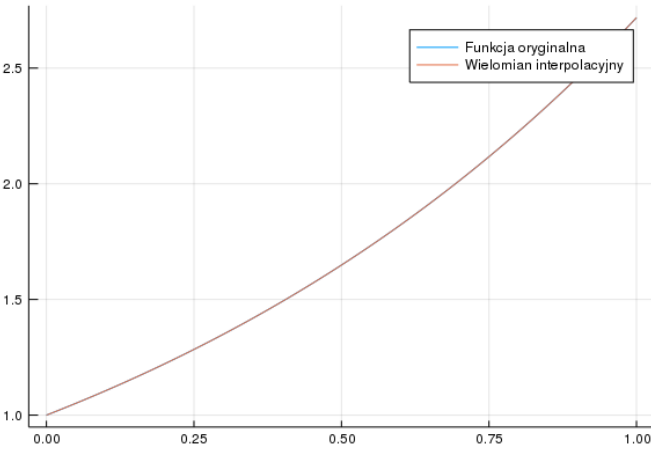
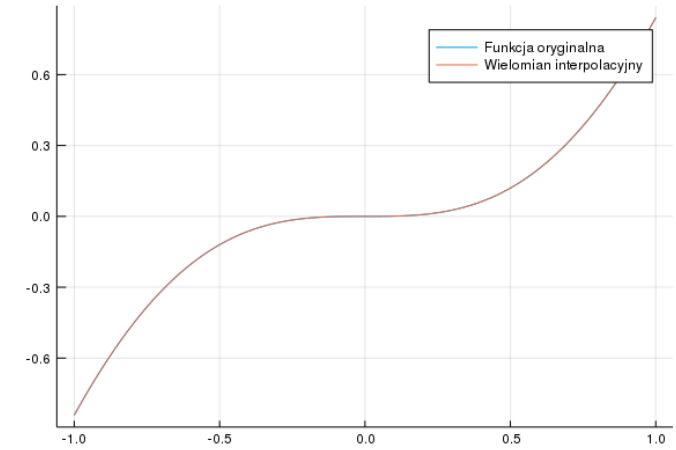
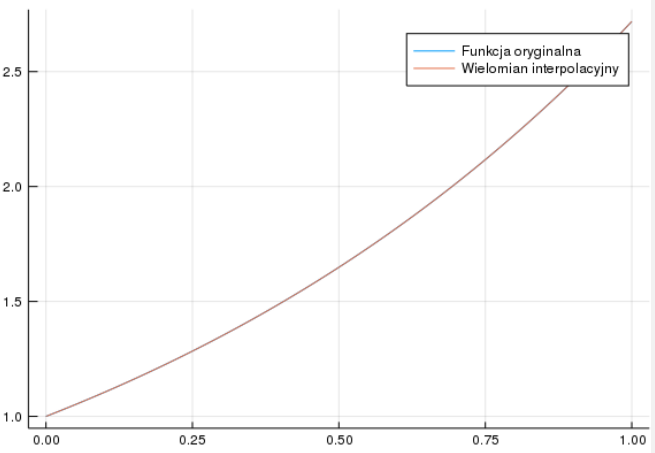
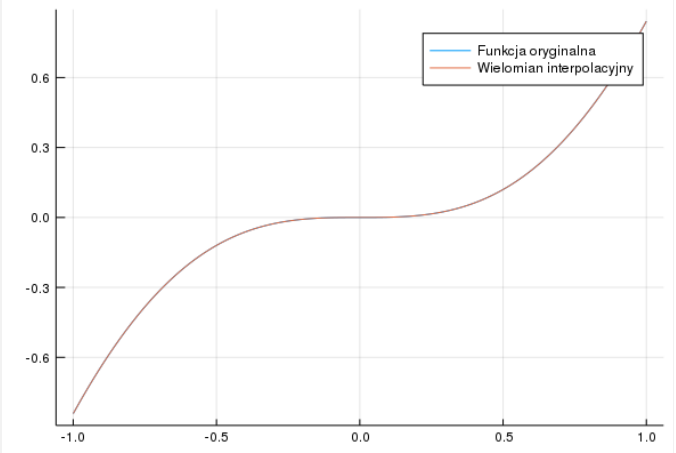
Zadanie polega na przetestowaniu funkcji $rysujNnfx(f, a, b, n)$ na następujących przykładach:

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

5.2 Rozwiązanie

Dla odpowiednich danych wywołano funkcję $rysujNnfx(f, a, b, n)$ opisaną w zadaniu 4.

5.3 Wyniki

n	e^x	$x^2*sin(x)$
5		
10		
15		

5.4 Wnioski

Zarówno dla funkcji e^x jak i $x^2 * sin(x)$ na zadanych przedziałach wielomiany interpolacyjne są bardzo zbliżone interpolowanym funkcjom. W testach, dla najniższej dokładności, pierwsza różnica pojawiła się dopiero na piątym miejscu po przecinku.

Zadanie 6

6.1 Opis zagadnienia

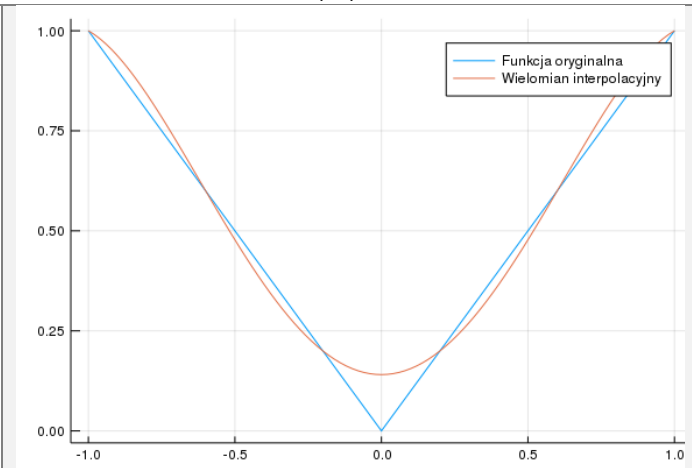
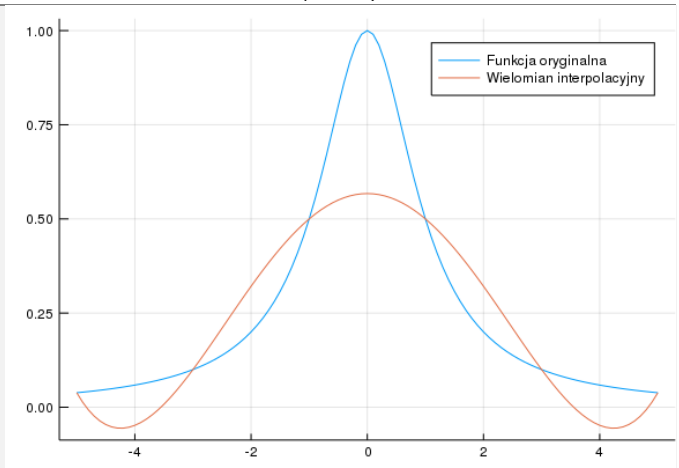
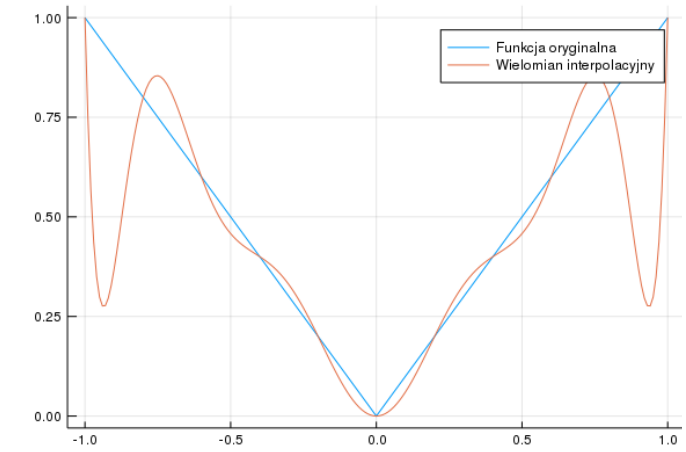
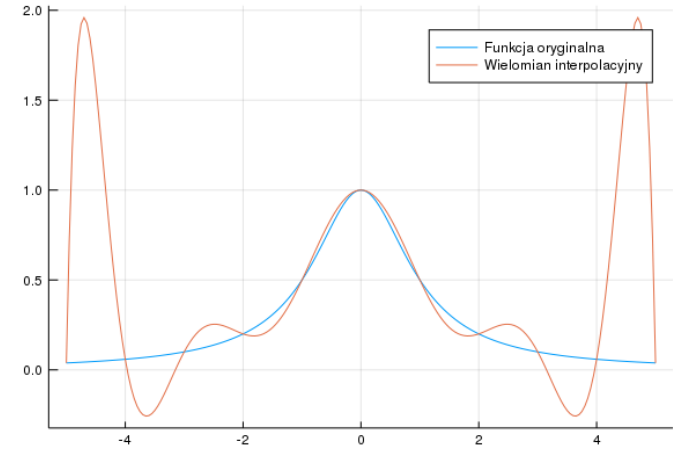
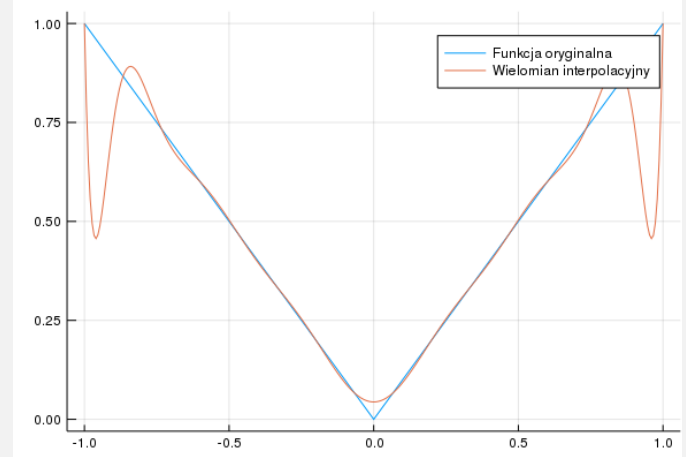
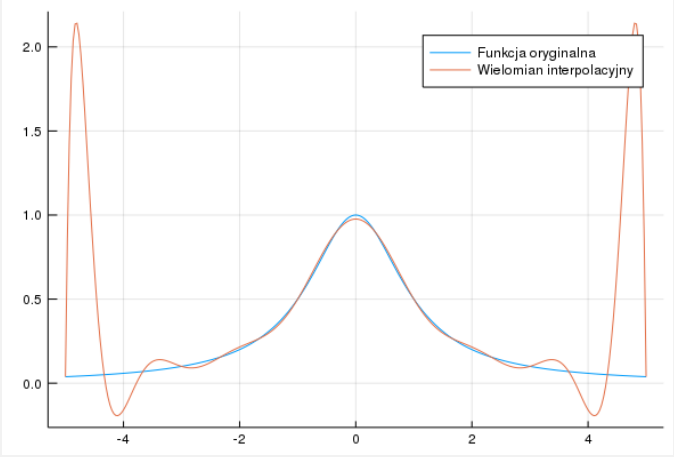
Zadanie polega na przetestowaniu funkcji $rysujNnfx(f, a, b, n)$ na następujących przykładach:

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

6.2 Rozwiązanie

Dla odpowiednich danych wywołano funkcję $rysujNnfx(f, a, b, n)$ opisaną w zadaniu 4.

6.3 Wyniki

n	*|x|*	*1/(1+x²)*
5		
10		
15		

6.4 Wnioski

Dla obu wielomianów interpolacyjnych można zauważyć znaczące rozbieżności w porównaniu z oryginalną funkcją, w szczególności na końcach przedziałów. Wyjaśnieniem dla funkcji $|x|$ jest jej nieróżniczkowalność. Natomiast dla funkcji $\frac{1}{1+x^2}$ zachodzi tzw. efekt Rungre'go – pogorszenie jakości interpolacji wielomianowej pomimo zwiększania liczby węzłów. Zauważyć można, że stosując równoodległe węzły sprawia, że w miejscach gdzie interpolacja funkcji jest trudniejsza przypada stosunkowo niewielka liczba punktów (tyle samo co w innych miejscach). Oczywiste jest również, że wielomian wysokiego stopnia ma dużą liczbę zer oraz szybko rozbiega do nieskończoności, dlatego na końcach przedziałów zaczyna przyjmować skrajne wartości. W celu rozwiązania tego problemu, do wyznaczenia węzłów które bardziej się nadają do wyliczania wielomianów interpolacyjnych takich funkcji, możnaby zastosować wielomiany oparte na węzłach Czybyszewa. Mają one dużo mniejsze oscylacje, gdyż następuje zagęszczenie węzłów przy końcach przedziału.