

Obliczenia naukowe

Sprawozdanie 4

Józef Piechaczek

7 grudnia 2019

1 Zadanie 1

Celem zadania 1 jest napisanie funkcji obliczającej ilorazy różnicowe.

Definicja funkcji:

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

Dane:

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 $f(x_0), \dots, f(x_n)$

Wyniki:

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

$fx[1]=f[x_0]$

$fx[2]=f[x_0, x_1], \dots, fx[n+1]=f[x_0, \dots, x_n]$

Funkcję należy zaprojektować bez użycia tablicy dwuwymiarowej.

Opis algorytmu:

W celu obliczenia ilorazów korzystamy z następujących zależności:

I. $f[x_i] = f(x_i)$ dla $0 \leq i \leq n$

II. $f[x_i, \dots, x_{i+j+1}] = \frac{f[x_{i+1}, \dots, x_{i+j+1}] - f[x_i, \dots, x_{i+j}]}{x_{i+j+1} - x_i}$

Ilorazy różnicowe możemy łatwo policzyć korzystając z tablicy trójkątnej tworzonej w następujący sposób:

x_0	$f[x_0]$	$f[x_0, x_1]$	$f[x_0, x_1, x_2]$	\dots	$f[x_0, \dots, x_n]$
x_1	$f[x_1]$	$f[x_1, x_2]$	$f[x_1, x_2, x_3]$		
x_2	$f[x_2]$	$f[x_2, x_3]$	$f[x_2, x_3, x_4]$		
\vdots	\vdots	\vdots	\vdots		
x_{n-2}	$f[x_{n-2}]$	$f[x_{n-2}, x_{n-1}]$	$f[x_{n-2}, x_{n-1}, x_n]$		
x_{n-1}	$f[x_{n-1}]$	$f[x_{n-1}, x_n]$			
x_n	$f[x_n]$				

Do przechowywania wartości ilorazów wystarczy zwykła tabela, której początkowymi wartościami, będą wartości funkcji w interpolowanych węzłach. W i -tej iteracji zewnętrznej pętli algorytmu będziemy modyfikować wartości w tablicy od n do i .

Algorytm:

```
l <- length(x)
fx <- f
for j=2 to l do
    for i=l downto j do
        fx[i] <- (fx[i] - fx[i - 1])/(x[i] - x[i - j + 1])
    end for
end for
return fx
```

2 Zadanie 2

Napisać funkcję obliczającą wartość wielomianu interpolacyjnego stopnia n w postaci Newtona $N_n(x)$ w punkcie $x = t$ za pomocą uogólnionego algorytmu Hornera, w czasie $O(n)$.

Definicja funkcji:

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

Dane:

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

$fx[1]=f[x_0]$

$fx[2]=f[x_0, x_1], \dots, fx[n+1]=f[x_0, \dots, x_n]$

Wyniki:

nt wartość wielomianu w punkcie t

Opis algorytmu:

W celu obliczenia wartości wielomianu interpolacyjnego w punkcie użyjemy uogólnionego algorytmu Hornera.

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

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

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

Podane wzory wynikają z następującej postaci Newtona wielomianu interpolacyjnego

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

Algorytm działa w czasie $O(n)$.

Algorytm:

```
l <- length(x)
nt = fx[1]
for i=l-1 downto 1
    nt = fx[i] + (t - x[i]) * nt
return nt
```

3 Zadanie 3

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

Definicja funkcji:

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

Dane:

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
 $fx[1] = f[x_0]$
 $fx[2] = f[x_0, x_1], \dots, fx[n+1] = f[x_0, \dots, x_n]$

Wyniki:

a wektor długości $n + 1$ zawierający obliczone współczynniki postaci naturalnej
 $a[1] = a_0$
 $a[2] = a_1, \dots, a[n] = a_{n-1}, a[n+1] = a_n$

Opis algorytmu:

Algorytm korzysta z uogólnionego algorytmu Hornera z poprzedniego zadania. Jako a_n przyjmujemy wartość $f[x_0, \dots, x_n]$, co da się łatwo wykazać. Następnie w każdej iteracji zewnętrznej pętli obliczamy częściową wartość dla wielomianu, podobnie jak w zadaniu 2, a w wewnętrznej pętli aktualizujemy wartości współczynników tak, aby częściowy wielomian w_k był w postaci naturalnej. Algorytm działa w czasie $O(n^2)$.

Algorytm:

```
l <- length(x)
a[l] <- fx[l]
for i = (l-1) downto 1
    a[i] = fx[i] - a[i+1] * x[i]
    for j = (i+1) to (l-1)
        a[j] = a[j] - a[j+1] * x[i]
return a
```

4 Zadanie 4

Napisać funkcję, która zinterpoluje zadaną funkcję $f(x)$ w przedziale $[a, b]$ za pomocą wielomianu interpolacyjnego stopnia n w postaci Newtona. Następnie narysuje wielomian interpolacyjny i interpolowaną funkcję. W interpolacji użyć węzłów równoodległych, tj. $x_k = a + kh, h = \frac{b-a}{n}, k = 0, 1, \dots, n$. Nie wyznaczać wielomianu interpolacyjnego w jawnej postaci. Należy skorzystać z funkcji `ilorazyRoznicowe` i `warNewton`.

Definicja funkcji:

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

Dane:

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

a, b przedział interpolacji

n stopień wielomianu interpolacyjnego

Wyniki:

wykres funkcji interpolowanej i wielomianu interpolacyjnego na przedziale $[a, b]$

Opis rozwiązania:

Algorytm dzieli przedział $[a, b]$ na $n + 1$ równo odległych punktów i oblicza wartość funkcji w tych punktach. Następnie korzysta z funkcji `ilorazyRoznicowe` w celu obliczenia ilorazów różnicowych. Następnie program dzieli zadany przedział na równo odległe punkty z większym zagęszczeniem (w celu większej dokładności wykresu) i oblicza wartości funkcji interpolowanej i wielomianu interpolującego (za pomocą `warNewton`) w tych punktach. Wykresy są tworzone za pomocą biblioteki **Plots** i zapisywane do pliku.

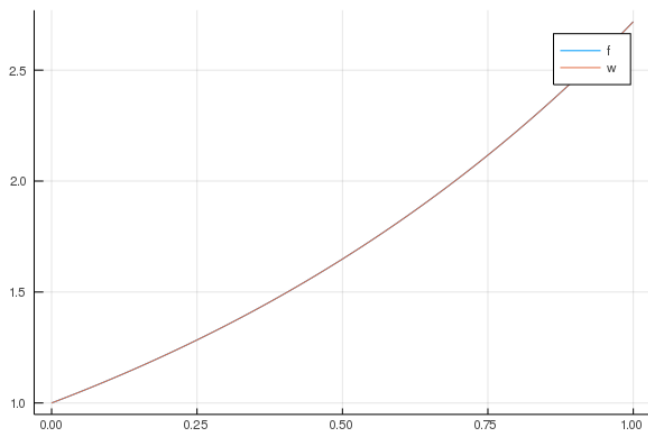
5 Zadanie 5

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

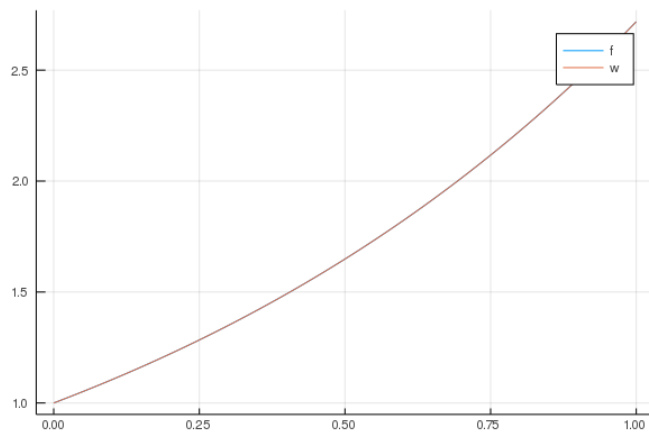
(a) e^x , $[0, 1]$, $n = 5, 10, 15$

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

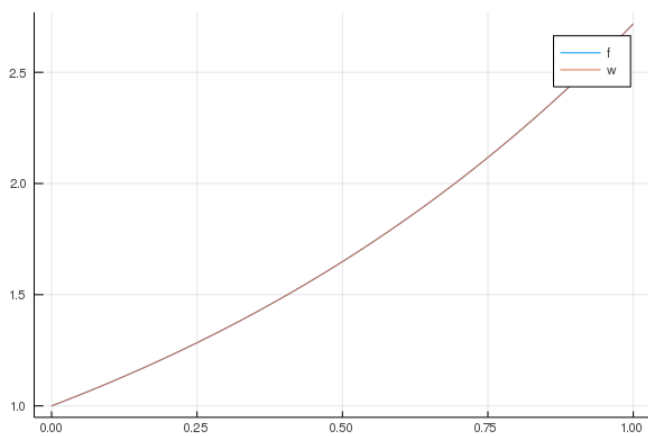
Rozwiązania:



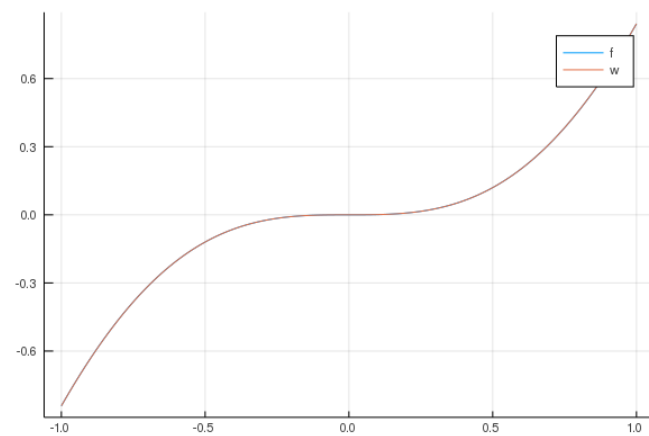
Rysunek 1: $e^x, n = 5$



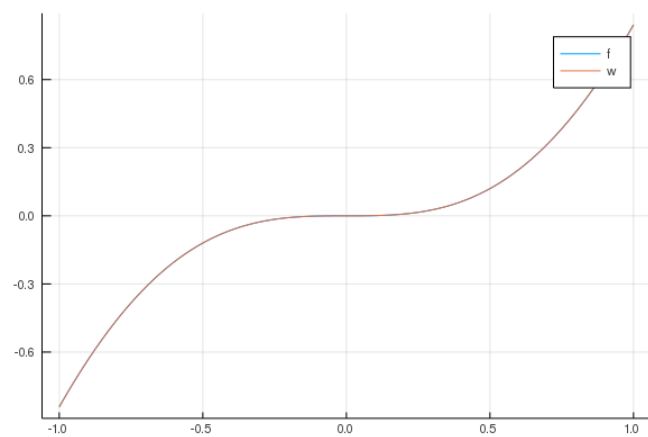
Rysunek 2: $e^x, n = 10$



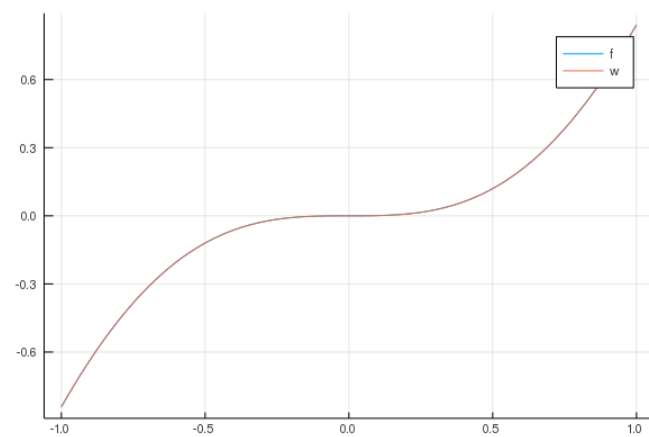
Rysunek 3: $e^x, n = 15$



Rysunek 4: $x^2 \sin x, n = 5$



Rysunek 5: $x^2 \sin x, n = 10$



Rysunek 6: $x^2 \sin x, n = 15$

Wnioski:

Patrząc na wykresy funkcji można zauważyć że zadane wielomiany bardzo dobrze interpolują podane funkcje. Nawet dla wielomianu stopnia $n = 5$ nie da się zobaczyć rozbieżności na wykresach.

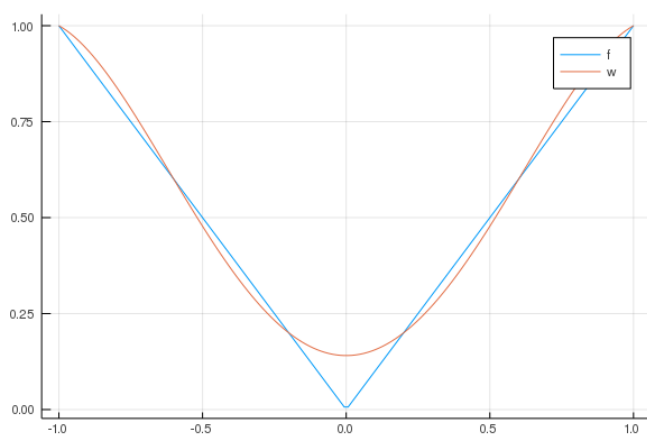
6 Zadanie 6

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

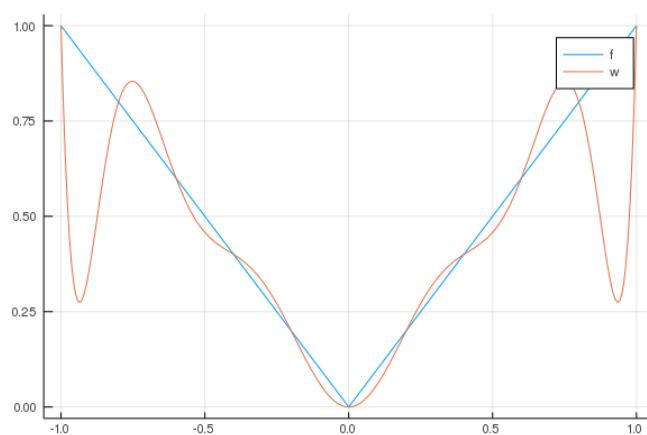
(a) $|x|$, $[-1, 1]$, $n = 5, 10, 15$

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

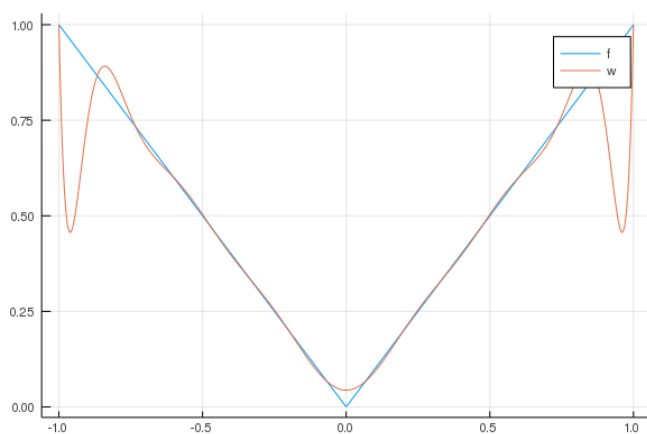
Rozwiązania:



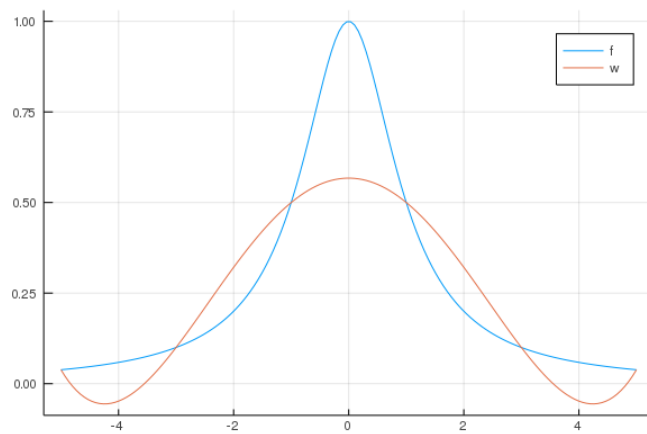
Rysunek 7: $|x|$, $n = 5$



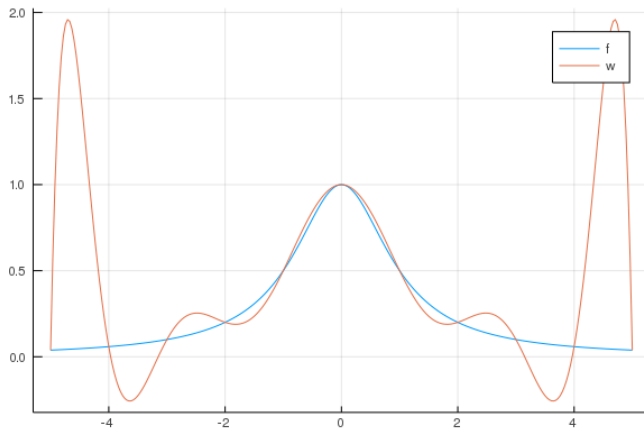
Rysunek 8: $|x|$, $n = 10$



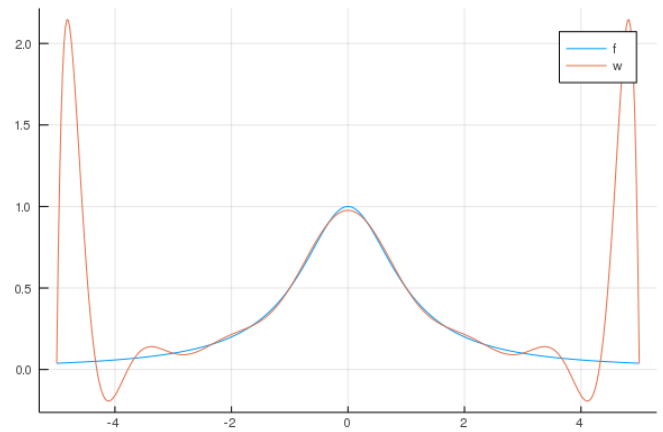
Rysunek 9: $|x|$, $n = 15$



Rysunek 10: $\frac{1}{1+x^2}$, $n = 5$



Rysunek 11: $\frac{1}{1+x^2}, n = 10$



Rysunek 12: $\frac{1}{1+x^2}, n = 15$

Wnioski:

W przypadku wielomianów interpolujących w zadaniu 6 możemy zauważyć, że wraz ze wzrostem stopnia wielomianu jakość interpolacji ulega pogorszeniu. Jest to szczególnie widoczne na końcach przedziałów. W wypadku pierwszym wynika to z faktu, że funkcja znacząco odbiega od funkcji gładkiej. W drugim przykładzie możemy zaobserwować efekt Rungego, który jest zjawiskiem typowym dla interpolacji za pomocą wielomianów wysokich stopni przy stałych odległościach węzłów. Aby uniknąć tego efektu, możemy stosować interpolację z węzłami upakowanymi gęściej na krańcach przedziału interpolacji.