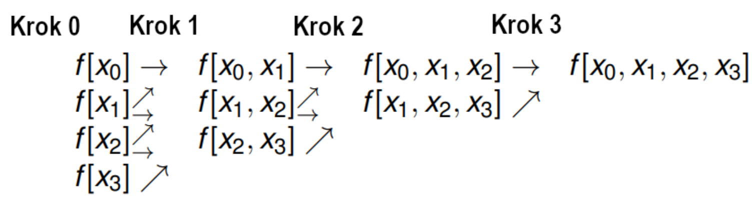
Obliczenia Naukowe - Lista nr 4

Maksymilian Piotrowski

1 Zadanie 1

Opis problemu: Należy napisać funkcję obliczającą ilorazy różnicowe nie używając tablicy dwuwymiarowej. Jako dane wejściowe otrzymujemy $x_0, x_1, ..., x_n$ oraz $f(x_0), f(x_1), ..., f(x_n)$.

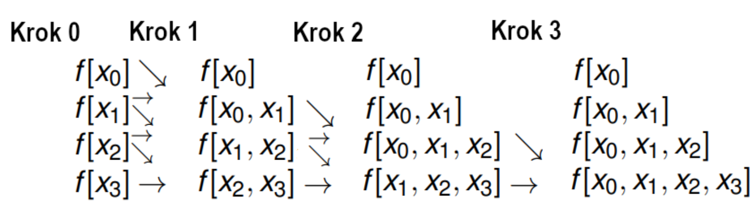
Rozwiązanie:
Zachodzi $f[x_i] = f(x_i)$. Ponadto na wykładzie zostało udowodnione, że $f[x_0, x_1, ..., x_n] = \frac{f[x_1,x_2,...,x_n]-f[x_0,x_1,...,x_{n-1}]}{x_n-x_0}$.
Został też pokazany algorytm wyznaczania ilorazów różnicowych rzędów 0,..,n (przykład dla n=3):



Rysunek 1: Przykład działania alogrytmu z wykładu.

Algorytm używa tablicy trójkątnej, aby obliczać kolejne $f[x_k, ..., x_l]$ i $f[x_{k+1}, ..., x_{l+1}]$, a na ich postawie $f[x_k, ..., x_{l+1}]$ za pomocą wzoru $f[x_0, x_1, ..., x_n] = \frac{f[x_1,x_2,...,x_n]-f[x_0,x_1,...,x_{n-1}]}{x_n-x_0}$, jak pokazują strzałki na Rysunku 1.

Jak widać, aby obliczyć wszystkie wielomiany różnicowe w algorytmie z Rysunku 1 musimy w danym kroku i obliczyć $f[x_0, ..., x_i]$, $f[x_1, ..., x_{i+1}]$, ..., $f[x_{n-i}, ..., x_n]$.
Czyli w i-tym kroku trzeba obliczyć $n + 1 - i$ wartości, czyli w każdym kolejnym kroku o jedną mniej. Ponadto po wyliczeniu wartości w kroku $i+1$, możemy zapomnieć wszystkie wartości wyliczone w i-tym kroku, oprócz jednej - ilorazu różnicowego $f[x_0, ..., x_i]$. Możemy więc nadpisywać wartości w pojedynczej tablicy o n+1 komórkach, zamiast używać tablicy trójkątnej. Daje nam to złożoność pamięciową $O(n)$.



Rysunek 2: Przykład działania zaimplementowanego algorytmu. Kolumny reprezentują stan jednowymiarowej tablicy po wykonaniu kroku.

Z Rysunku 2 widać, że kolejność działań w krokach jest ważna, bo jeśli nie wykonamy ich "od dołu w górę", to nadpiszemy potrzebne do dalszych obliczeń wartości.

Łącząc powyższe obserwacje, dla danych
$x[1] = x_0$, ..., $x[n+1] = x_n$
$f[1] = f(x_0)$, ..., $f[n+1] = f(x_n)$
otrzymujemy algorytm zwracający tablicę ilorazów różnicowych:

Algorytm 1 Ilorazy różnicowe

$n \leftarrow$ length(x) (wcześniej oznaczane jako n+1)
$arr \leftarrow$ pusta tablica n-elementowa
for $i \leftarrow 1$ to n **do**
  $arr[i] \leftarrow f[i]$
end for
for $i \leftarrow 2$ to n **do**
  for $j \leftarrow n$ down to i **do**
    $arr[j] \leftarrow \frac{arr[j]-arr[j-1]}{x[j]-x[j-i+1]}$
  end for
end for
return arr

Implementacja algorytmu jako funkcja ilorazyRoznicowe(x::VectorFloat64, f::VectorFloat64) znajduje się w l4_z1234.jl. Testy znajdują się w l4_z1234tests.jl

2 Zadanie 2

Opis problemu: Należy napisać funkcję obliczającą wartość wielomianu interpolacyjnego Newtona $N_n(x)$ w punkcie $x = t$ za pomocą uogólnionego algorytmu Hornera, w czasie $O(n)$.

Rozwiązanie: Rozważany wielomian ma postać:
$N_n(x) = f[x_0] + f[x_0, x_1](x - x_0) + ... + f[x_0, x_1, ..., x_n](x - x_0)(x - x_1)...(x - x_{n-1})$

Możemy go równoważnie zapisać jako:
$N_n(x) = (...((f[x_0, ..., x_n])(x - x_{n-1}) + (f[x_0, ..., x_{n-1}]))(x - x_{n-2}) + ... + (f[x_0, x_1])...)(x - x_0) + f[x_0]$

Wtedy wykonując działania zgodnie z kolejnością wykonywania działań, dla danych
$x_0, x_1, ..., x_n$ oraz $f[x_0], f[x_0, x_1], ..., f[x_0, x_1, ..., x_n]$,
otrzymujemy algorytm obliczający wartość wielomianu interpolacyjnego Newtona w punkcie:

Algorytm 2 Wartość wielomianu interpolacyjnego w punkcie t

$x \leftarrow t$
$r \leftarrow f[x_0, ..., x_n]$
for $i \leftarrow n-1$ down to 0 **do**
  $r \leftarrow r * (x - x_i)$
  $r \leftarrow r + f[x_0, .., x_i]$
end for
return r

Implementacja algorytmu jako funkcja warNewton(x::VectorFloat64, fx::VectorFloat64, t::Float64) znajduje się w l4_z1234.jl. Testy znajdują się w l4_z1234tests.jl

3 Zadanie 3

Opis problemu: Należy napisać funkcję obliczającą wartości współczynników postaci naturalnej a_0, a_1, ..., a_n wielomianu interpolacyjnego w postaci Newtona, znając jego ilorazy różnicowe $f[x_0]$, $f[x_0, x_1]$, ...,$f[x_0, ..., x_n]$ oraz węzły x_0, x_1, ..., x_n.

Rozwiązanie: Z wykładu wiemy, że $f[x_0, x_1, ..., x_n] = a_n$. Algorytm będzie obliczał pozostałe współczynniki a_i i aktualizował te już obliczone dla kolejnych wielomianów:
$W_0(x) = a_n = f[x_0, ..., x_n]$, następnie
$W_1(x) = a_{n-1} * x^0 + a_n * x^1 = f[x_0, ..., x_{n-1}] + f[x_0, ..., x_n](x - x_{n-1})$, następnie
$W_2(x) = a_{n-2} * x^0 + a_{n-1} * x^1 + a_n * x^2 =$
$= f[x_0, ..., x_{n-2}] + f[x_0, ..., x_{n-1}](x - x_{n-2}) + f[x_0, ..., x_n](x - x_{n-1})(x - x_{n-2})$, aż dojdziemy do szukanego wielomianu:
$W_n(x) = a_0 * x^0 + a_1 * x^1 + ... + a_n * x^n =$
$= f[x_0] + f[x_0, x_1](x - x_0) + ... + f[x_0, x_1, ..., x_n](x - x_0)(x - x_1)...(x - x_{n-1})$

Załóżmy, że mamy obliczone współczynniki dla
$W_i(x) = a_{n-i} * x^0 + a_{n-i+1} * x^1 + ... + a_n * x^i =$
$= f[x_0, .., x_{n-i}] + f[x_0, ..., x_{n-i+1}](x - x_{n-i}) + ... + f[x_0, x_1, ..., x_n](x - x_{n-i})(x - x_{n-i+1})...(x - x_{n-1})$.
Wtedy możemy obliczyć współczynniki wielomianu
$W_{i+1}(x) = a_{n-(i+1)} * x^0 + a_{n-i} * x^1 + a_{n-i+1} * x^2 + ... + a_n * x^{i+1} =$
$= f[x_0, .., x_{n-(i+1)}] + f[x_0, .., x_{n-i}](x - x_{n-(i+1)}) + f[x_0, ..., x_{n-i+1}](x - x_{n-i})(x - x_{n-(i+1)}) + ... +$
$f[x_0, x_1, ..., x_n](x - x_{n-(i+1)})(x - x_{n-i})(x - x_{n-i+1})...(x - x_{n-1})$
w następujący sposób:

Zauważmy, że a_{n-i} stojący przy x^0 w W_i został pomnożony przez $(x - x_{n-(i+1)})$ w W_{i+1}, oraz dodano iloraz różnicowy $f[x_0, .., x_{n-(i+1)}]$.
Zatem współczynnik stojący przy x^0 w W_{i+1} jest równy: $a_{n-(i+1)} = f[x_0, .., x_{n-(i+1)}] - a_{n-i} * x_{n-(i+1)}$.
Następnie trzeba zaktualizować obliczone już współczynniki $a_{n-i}, ..., a_n$. Każdy współczynnik a_j z W_i został pomnożony przez $(x - x_{n-(i+1)})$ w W_{i+1}. Zatem zaktualizowane współczynniki, stojące przy potędze x o jeden większej niż pierwotnie, mają wartość $a_j = a_j - a_{j+1} * x_{n-(i+1)}$.

Zapętlając wyżej opisany krok dla danych
$x[1] = x_0$, ..., $x[n+1] = x_n$
$fx[1] = f[x_0]$, ..., $fx[n+1] = f[x_0, ..., x_n]$
otrzymujemy algorytm:

Algorytm 3 Wartości współczynników naturalnych

```
n ← length(x) (wcześniej oznaczane jako n+1)
a ← pusta tablica n-elementowa
a[n] ← fx[n]
for i ← 0 to n − 1 do
    a[n − i] ← fx[n − i] − a[n − (i − 1)] ∗ x[n − i]
    for j ← n − (i + 1) to n − 1 do
        a[j] ← a[j] − a[j + 1] ∗ x[n − i]
    end for
end for
return a
```

Implementacja algorytmu jako funkcja naturalna(x::VectorFloat64, fx::VectorFloat64) znajduje się w l4_z1234.jl. Testy znajdują się w l4_z1234tests.jl

4 Zadanie 4

Opis problemu: Należy napisać funkcję, która zinterpoluje zadane f w przedziale $[a, b]$ używając wielomianu interpolacyjnego zadanego stopnia n. Następnie funkcja ma narysować f i uzyskany wielomian interpolacyjny W.

Rozwiązanie:
1. Tworzymy tablicę n+1 równoodległych węzłów x_k, k = 0,1,...,n, z przedziału $[a, b]$.
2. Obliczamy i zapamiętujemy $f(x_k)$ dla każdego x_k z kroku 1.
3. Używając funkcji ilorazyRoznicowe z zadania 1 obliczamy ilorazy różnicowe i_k dla x_k, $f(x_k)$.

4. Rysujemy f na przedziale $[a, b]$ obliczając $f(c_l)$ dla równoodległych wartości c_l, l = 1,2,...,1000 z przedziału $[a, b]$. Obliczone wartości naniosłem na wykres przy pomocy modułu Plots z Julii.
5. Rysujemy wielomian interpolacyjny Newtona W na przedziale $[a, b]$ używając 1000 wartości z kroku nr. 4. Aby obliczać $W(c_l)$, l = 1,2,...,1000, używamy funkcji warNewton z Zadania 2 z argumentami x_k, i_k.

Implementacja algorytmu jako funkcja rysujNnfx(f,a::Float64,b::Float64,n::Int) znajduje się w l4_z1234.jl. Testy znajdują się w l4_z1234tests.jl

5 Zadanie 5

Opis problemu: Należy przetestować zaimplementowaną w Zadaniu 4 funkcję rysujNnfx(f,a,b,n) na przykładach:

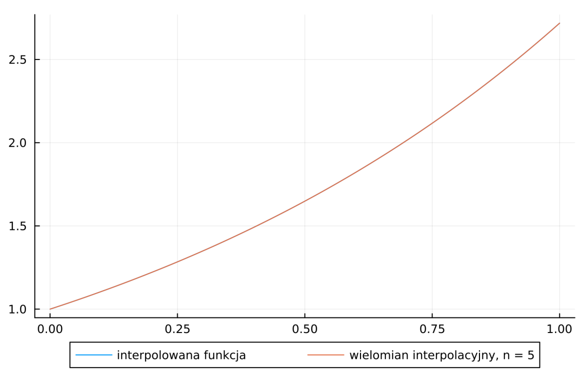
(a) $f(x) = e^x$, $[a, b] = [0, 1]$, n = 5,10,15

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

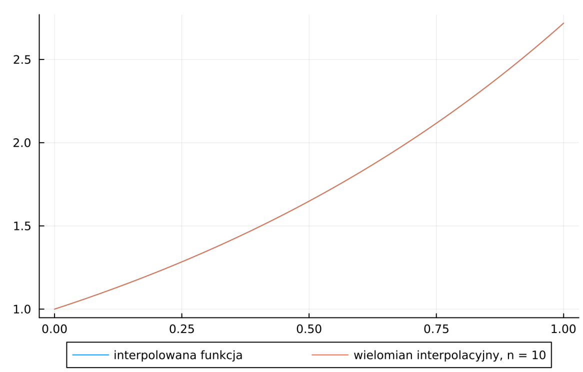
Rozwiązanie: Wywołano funkcję rysujNnfx dla zadanych przykładów.

Implementacja znajduje się w pliku l4_z5.jl.

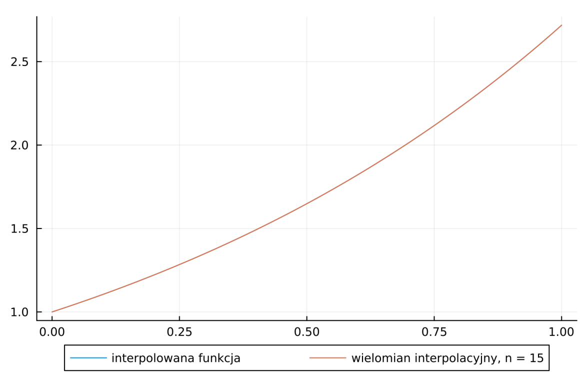
Wyniki i interpretacja:



n = 5

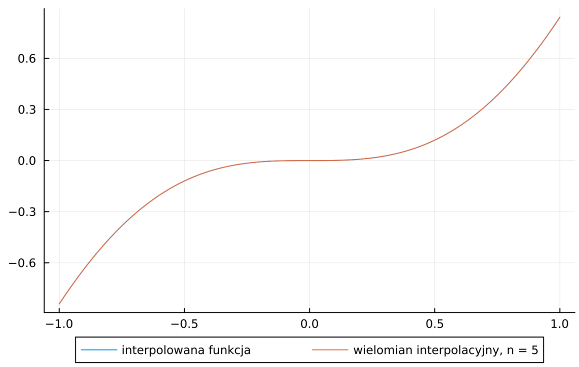


n = 10

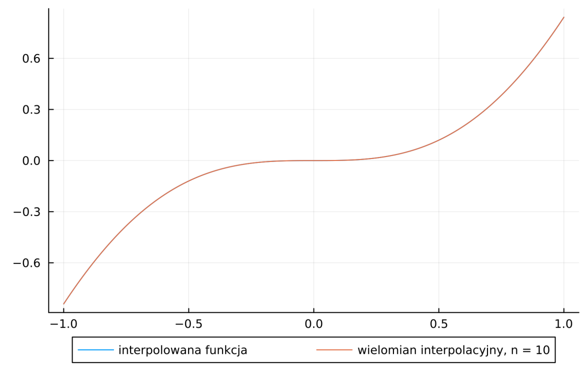


n = 15

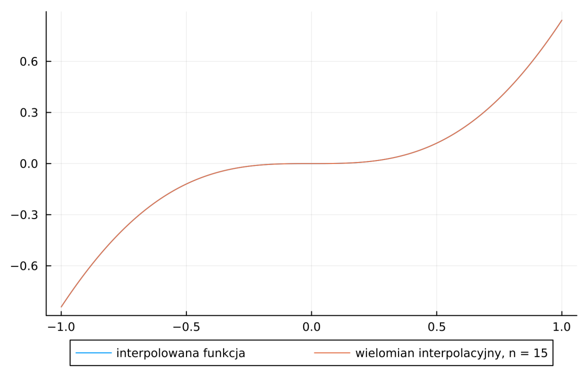
Rysunek 3: Wykresy dla przykładu (a): $f(x) = e^x$



n = 5



n = 10



n = 15

Rysunek 4: Wykresy dla przykładu (b): $f(x) = x^2 * sin(x)$

$f(x)$	n	avg($\|f(c_l) - W(c_l)\|$)
e^x	5	8.64572353878712e-7
e^x	10	2.3865354137342366e-14
e^x	15	2.2841728508637972e-15
$x^2 * sin(x)$	5	0.00012926152441936067
$x^2 * sin(x)$	10	3.1628890517901894e-9
$x^2 * sin(x)$	15	1.4848638405392179e-16

Tabela 1: Średnia odległość wartości f i wielomianu interpolacyjnego W w rysowanych punktach c_l

Wykresy funkcji pokrywają się wizualnie z wielomianami interpolacyjnymi już dla małego n = 5. Zgodnie z oczekiwaniami wzrost n zmniejsza średnią odległość wartości f i wielomianu interpolacyjnego w punktach (Tabela 1).

Wnioski: Wielomiany interpolacyjne wyznaczane dla węzłów równoodległych dobrze przybliżają funkcje $x^2 * sin(x)$ oraz e^x już dla stopnia wielomianu interpolacyjnego równego 5. Zwiększenie stopnia daje jednak większą dokładność przybliżenia.

6 Zadanie 6

Opis problemu: Należy przetestować zaimplementowaną w Zadaniu 4 funkcję rysujNnfx(f,a,b,n) na przykładach:

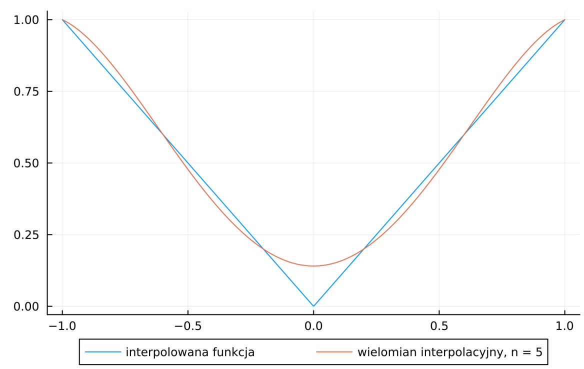
(a) $f(x) = |x|$, $[a, b] = [-1, 1]$, n = 5,10,15

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

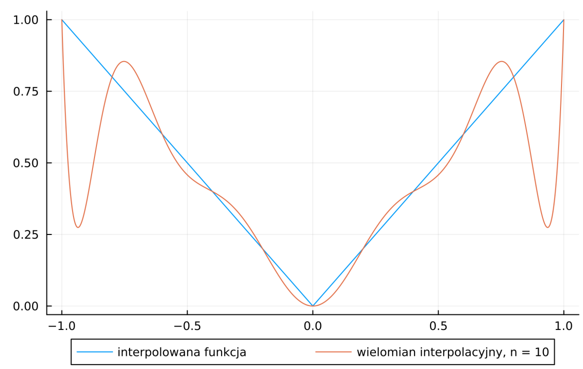
Rozwiązanie: Wywołano funkcję rysujNnfx dla zadanych przykładów.

Implementacja znajduje się w pliku l4_z5.jl.

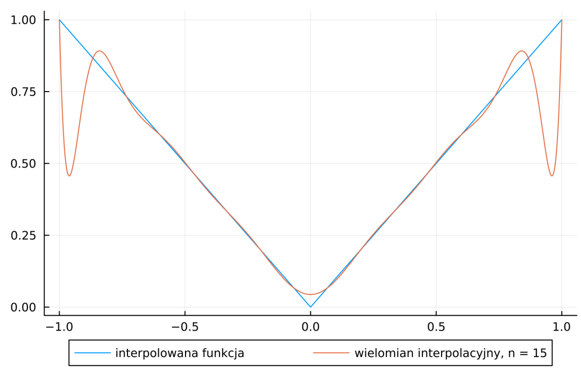
Wyniki i interpretacja:



n = 5

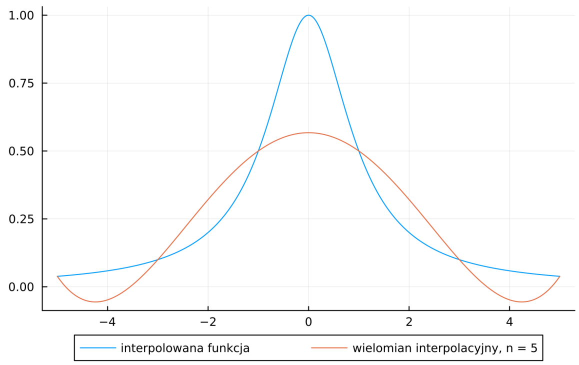


n = 10

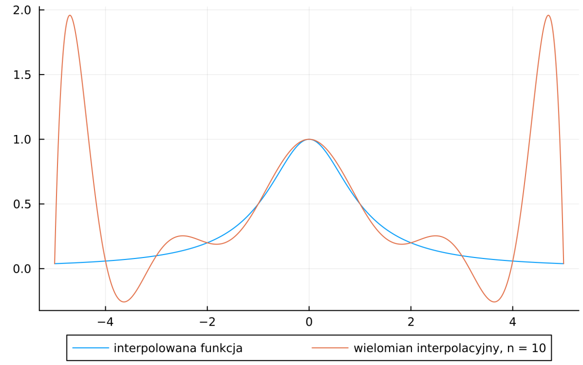


n = 15

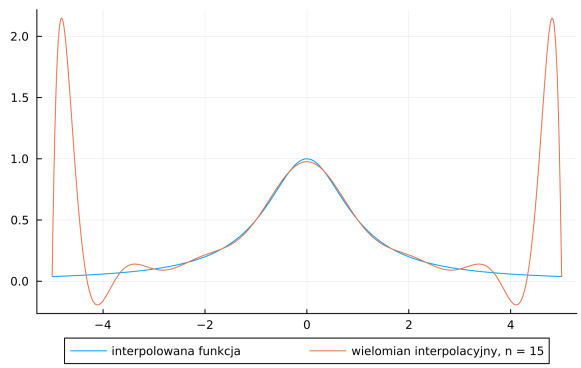
Rysunek 5: Wykresy dla przykładu (a): $f(x) = |x|$



n = 5



n = 10



n = 15

Rysunek 6: Wykresy dla przykładu (b): $f(x) = \frac{1}{1+x^2}$

$f(x)$	n	$\mathrm{avg}(\lvert f(c_l) - W(c_l)\rvert)$
$\lvert x\rvert$	5	0.03207875405183263
$\lvert x\rvert$	10	0.10565556807290731
$\lvert x\rvert$	15	0.04677707193749788
$\frac{1}{1+x^2}$	5	0.10941932077454027
$\frac{1}{1+x^2}$	10	0.291898518551083
$\frac{1}{1+x^2}$	15	0.18911227222615012

Tabela 2: Średnia odległość wartości f i wielomianu interpolacyjnego W w rysowanych punktach c_l

Wykresy wielomianów nie pokrywają się wizualnie z wykresami f nawet dla n=15. Ponadto przy zwiększaniu n średnia odległość wartości f i W w rysowanych punktach nie maleje (Tabela 2).

Wnioski: Wielomiany interpolacyjne wyznaczane dla węzłów równoodległych niedokładnie przybliżają funkcje $|x|$ oraz $\frac{1}{1+x^2}$. Wartości f i wielomianu interpolacyjnego są zauważalnie rozbieżne. Jest to szczególnie widoczne na końcach przedziału $[a, b]$.
Dla $f(x) = |x|$ na rozbieżność wpływa brak pochodnej w punkcie x=0. Funkcja różni się od funkcji gładkich, które łatwiej jest przybliżyć za pomocą wielomianu.
Dla $\frac{1}{1+x^2}$ rozbieżność wynika z wystąpienia zjawiska Runge'go - wraz ze zwiększeniem liczby węzłów, wielomian interpolacyjny gorzej przybliża f na końcach przedziałów. Zjawisko to zachodzi dla niektórych funkcji jeśli użyjemy równoodległych węzłów do generowania wielomianu interpolacyjnego, tak jak zrobiliśmy to w Zadaniu 4. Aby zminimalizować rozbieżności wynikające ze zjawiska Runge'go można użyć węzłów występujących gęściej na końcach przedziału.

Wielomiany interpolacyjne oparte na równoodległych węzłach nie przybliżają dobrze wszystkich funkcji. Wynika to z istnienia zjawiska Runge'go oraz faktu, że nie wszystkie funkcje są dobrze przybliżane przez wielomiany.