Obliczenia naukowe lista 4

Stanisław Tomkowiak

5 grudnia 2024

Zadanie 1

Opis zadania

Napisać funkcję obliczającą ilorazy różnicowe.

```
function ilorazyRoznicowe(x::VectorFloat64, f::VectorFloat64)
```

Dane wejściowe

- `x` – wektor długości $n+1$ zawierający węzły $x_0, \ldots, x_n$ `x[1]`$=x_0$`,...,` `x[n+1]`$=x_n$
- `f` – wektor długości $n+1$ zawierający wartości interpolowanej funkcji w węzłach $f(x_0), \ldots, f(x_n)$

Dane wyjściowe

- `fx` – wektor długości $n+1$ zawierający obliczone ilorazy różnicowe

$$\texttt{fx[1]} = f[x_0]$$

$$\texttt{fx[2]} = f[x_0, x_1], \ldots, \texttt{fx[n]} = f[x_0, \ldots, x_{n-1}], \texttt{fx[n+1]} = f[x_0, \ldots, x_n]$$

Opis metody

Kluczową częścią rozwiązania zadania jest użycie wzoru na ilorazy różnicowe wyższych rzędów:

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

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

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

Łatwo zrozumieć sens działania algorytmu patrząc na poniższą tabelę.

x_0	$f[x_0]$	$f[x_0, x_1]$	$f[x_0, x_1, x_2]$	$f[x_0, x_1, x_2, x_3]$
x_1	$f[x_1]$	$f[x_1, x_2]$	$f[x_1, x_2, x_3]$	
x_2	$f[x_2]$	$f[x_2, x_3]$		
x_3	$f[x_3]$			

Modyfikujemy wektor wejściowy `f` w trakcie działania algorytmu, stopniowo obliczając ilorazy różnicowe wyższych rzędów i zapisując je w odpowiednich pozycjach w `fx`. W pierwszej iteracji obliczamy wartości z drugiej kolumny tabeli, czyli `fx` staje się wektorem $[f[x_0], f[x_0, x_1], f[x_1, x_2], f[x_2, x_3]]$. W kolejnych iteracjach (dla $i = 2, 3, \ldots$) obliczamy wyższe rzędy ilorazów różnicowych, zapisując je na kolejnych miejscach w `fx`.

W i-tej iteracji obliczamy wartości ilorazów różnicowych $f[x_0, \ldots, x_i], f[x_1, \ldots, x_{i+1}], \ldots$, które odpowiadają kolejnej kolumnie tabeli. Ostatecznie, po zakończeniu wszystkich iteracji, wektor `fx` zawiera wyniki z pierwszego wiersza tabeli, od drugiej kolumny aż do końca. Są to ilorazy różnicowe wyższych rzędów, wyliczone zgodnie z podanym schematem.

Pseudokod

1: **function** IlorazyRóżnicowe(x,f)
2: $fx \leftarrow$ kopiuj(f)
3: $len \leftarrow$ długość(x)
4: **for** $j \leftarrow 1$ **to** len **do**
5: **for** $i \leftarrow len$ **downto** $j+1$ **do**
6: $fx[i] \leftarrow \frac{fx[i]-fx[i-1]}{x[i]-x[i-j]}$
7: **end for**
8: **end for**
9: **return** fx

Zadanie 2

Opis zadania

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 $O(n)$.

```
function warNewton(x::VectorFloat64, fx::VectorFloat64, t::Float64)
```

Dane wejściowe

- `x` – wektor długości $n+1$ zawierający węzły $x_0, \ldots, x_n$ `x[1]`=x_0`, ..., x[n+1]`=x_n
- `fx` – wektor długości $n+1$ zawierający ilorazy różnicowe

$$\texttt{fx[1]} = f[x_0]$$

$$\texttt{fx[2]} = f[x_0, x_1], \ldots, \texttt{fx[n]} = f[x_0, \ldots, x_{n-1}] \texttt{fx[n+1]} = f[x_0, \ldots, x_n]$$

- `t` – punkt, w którym należy obliczyć wartość wielomianu

Dane wyjściowe

- `nt` – wartość wielomianu w punkcie t

Opis metody

Aby obliczyć wartość wielomianu interpolacyjnego stopnia n w punkcie $x = t$ za pomocą uogólnionego algorytmu Hornera pokażmy jak wygląda taki wielomian w postaci Newtona $N_n(x)$:

$N_n(x) = f[x_0]+f[x_0,x_1](x-x_0)+f[x_0,x_1,x_2](x-x_0)(x-x_1)+\ldots+f[x_0,\ldots,x_n](x-x_0)(x-x_1)\ldots(x-x_{n-1})$

Tak otrzymany wielomian kolejno zwijamy aby ograniczyć liczbę operacji:

$$N_n(x) = f[x_0] + (x-x_0)\left(f[x_0,x_1] + (x-x_1)\left(f[x_0,x_1,x_2] + \ldots + (x-x_{n-1})f[x_0,\ldots,x_n]\right)\right)$$

W ogólności schemat rozwiązania wygląda tak:

$$\omega_n(x) = f[x_0, \ldots, x_n]$$
$$\omega_k(x) = f[x_0, \ldots, x_k] + (x - x_k) \cdot \omega_{k+1}(x)$$
$$N_n(x) = \omega_0(x)$$

Pseudokod

1: **function** warNewton(x, fx, t)
2: len $\leftarrow$ length(x)
3: $nt \leftarrow fx$[len]
4: **for** $i \leftarrow$ len $- 1, \ldots, 1$ **do**
5: $nt \leftarrow nt \cdot (t - x[i]) + fx[i]$
6: **end for**
7: **return** nt

Zadanie 3

Opis zadania

Znając współczynniki wielomianu interpolacyjnego w postaci Newtona $c_0 = f[x_0], c_1 = f[x_0, x_1], c2 = f[x_0, x_1, x_2], \ldots, c_n = f[x_0, \ldots, x_n]$ (ilorazy różnicowe) oraz węzły $x_0, x_2, \ldots, x_n$ napisać funkcję obliczającą, w czasie $O(n^2)$, współczynniki jego postaci naturalnej $a_0, \ldots, a_n$ tzn. $a_n x^n + a_{n-1} x^{n-1} + \ldots + a_1 x + a_0$.

`function naturalna(x::VectorFloat64, fx::VectorFloat64))`

Dane wejściowe

- `x` – wektor długości $n+1$ zawierający węzły $x_0, \ldots, x_n$ `x[1]`$=x_0$`,..., x[n+1]`$=x_n$
- `fx` – wektor długości $n+1$ zawierający ilorazy różnicowe

$$\texttt{fx[1]} = f[x_0]$$

$$\texttt{fx[2]} = f[x_0, x_1], \ldots, \texttt{fx[n]} = f[x_0, \ldots, x_{n-1}] \texttt{fx[n+1]} = f[x_0, \ldots, x_n]$$

Dane wyjściowe

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

$$\texttt{a[1]} = a_0,$$

$$\texttt{a[2]} = a_1, \ldots, \texttt{a[n]} = a_{n-1}, \texttt{a[n+1]} = a_n$$

Opis metody

Aby obliczyć wartość współczynników wielomianu interpolacyjnego w postaci Newtona stopnia n należy spojrzeć na wzór wielomianu postaci Newtona:

$$N_n(x) = f[x_0] + f[x_0, x_1](x - x_0) + f[x_0, x_1, x_2](x - x_0)(x - x_1) + \ldots + f[x_0, \ldots, x_n](x - x_0)(x - x_1)\ldots(x - x_{n-1})$$

Mamy go przekształcić do postaci:

$$N_n(x) = a_0 + a_1 x + a_2 x^2 + \cdots + a_n x^n$$

Aby to osiągnąć podejmujemy takie kroki przed główną częścią algorytmu:

- kopiujemy tablicę `fx` do tablicy `a`, to ona będzie przechowywać wyniki dla poszczególnych współczynników.
- sprawdzamy długość tablicy `fx` i zapisujemy ją do zmiennej `length`.

Algorytm działa iteracyjne od końca z zagnieżdżoną pętlą działającą od aktualnie rozpatrywanego $i = 1$ do końca tablicy `a`. W każdym kroku algorytm wykonuje następujące kroki:

- Dla każdego i, współczynnik a[i] jest aktualizowany według wzoru:

  $$\texttt{a[i]} = \texttt{fx[i]} - \texttt{a[i+1]} * \texttt{x[i]},$$

  gdzie `a[i+1]` to współczynnik już obliczony w poprzednich iteracjach.

- Następnie, dla każdego j od $i + 1$ do $n - 1$, aktualizowane są pozostałe współczynniki `a[j]`.

Po przejściu przez wszystkie iteracje algorytm zwraca tablicę `a`.

Ogólne założenie algorytmu polega na tym, że współczynniki są aktualizowane, uwzględniając poprzednie wartości. Dla każdej pary współczynników `a[i]` i `a[i+1]` wylicza się nową wartość `a[i]` jako różnicę między aktualną wartością funkcji `fx[i]` a poprzednim współczynnikiem `a[i+1]` pomnożonym przez `x[i]`. Zagnieżdżona funkcja służy natomiast do tego, aby uwzględnić wpływ węzłów poprzednich. W ten sposób, stopniowo, algorytm przekształca współczynniki z postaci Newtona na postać naturalną.

Pseudokod

Przekształcenie wielomianu Newtona na postać naturalną

```
1: function NATURALNA(x, fx)
2:     len ← length(fx)
3:     a ← fx
4:     for i ← len − 1 downto 1 do
5:         a[i] ← fx[i] − a[i + 1] · x[i]
6:         for j ← i + 1 to len - 1 do
7:             a[j] ← a[j] − a[j + 1] · x[i]
8:         end for
9:     end for
10:    return a
```

Zadanie 4

Opis zadania

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. Do rysowania zainstaluj pakiet `Plots`.

W interpolacji użyć węzłów równoodległych, tj.

$$x_k = a + k * h, h = (b - a)/n, k = 0, 1, \ldots, n.$$

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

Dane wejściowe

- `f` - zadana jako anonimowa funkcja $f(x)$
- `a,b` – przedział interpolacji
- `n`– stopień wielomianu interpolacyjnego

Dane wyjściowe

- `plot` – funkcja rysuje wielomian interpolacyjny i interpolowaną funkcję w przedziale $[a, b]$.

Pseudokod

Rysowanie wykresu funkcji i wielomianu interpolacyjnego Newtona

```
1: function RYSUJNNFX(f, a,b,n)
2:     step ← (b−a)/n
3:     xnodes ← [a + i · step | i = 0, 1, . . . , n]
4:     ynodes ← f(xnodes)
5:     coefnodes ← ilorazyRoznicowe(xnodes, ynodes)
6:     nodes ← [warNewton(xnodes, coefnodes, x) | x ∈ xnodes]
7:     step ← (b−a)/500
8:     xstep ← [a + i · step | i = 0, 1, . . . , 500]
9:     ystep_real ← f(xstep)
10:    ystep_inter ← [warNewton(xnodes, coefnodes, x) | x ∈ xstep]
11:    plot(xstep, ystep_real, label = "f(x) (dokładna)")
12:    plot!(xstep, ystep_inter, label = "Wielomian interpolacyjny")
13:    scatter!(xnodes, nodes, label = "Węzły interpolacyjne")
14:    savefig()
```

Zadanie 5

Opis zadania

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

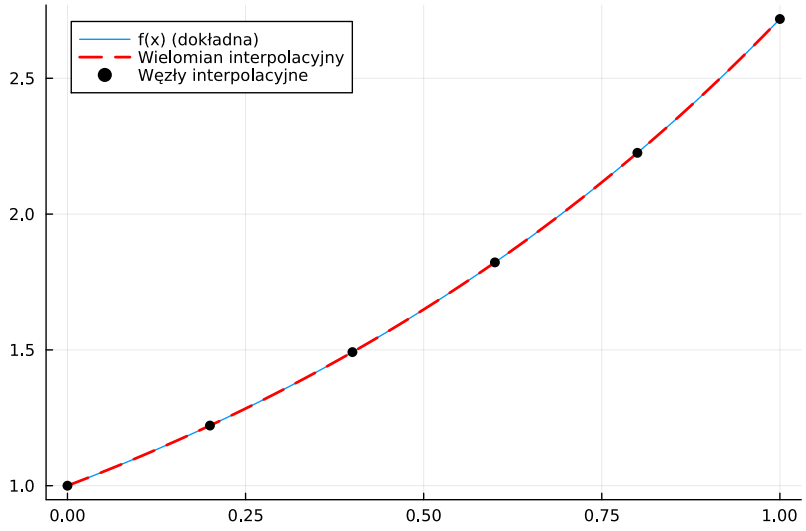
- e^x, $[0,1]$, $n = 5, 10, 15$
- $x^2 * sin(x)$, $[1,1]$, $n = 5, 10, 15$

Rozwiązanie

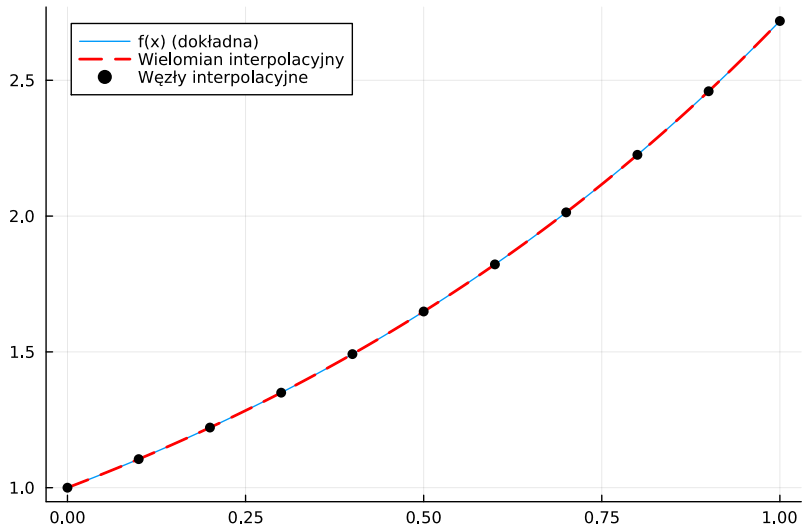
Należy uruchomić funkcję `rysujNnfx(f,a,b,n)` z odpowiednio zadanymi parametrami.

Wyniki i interpretacja

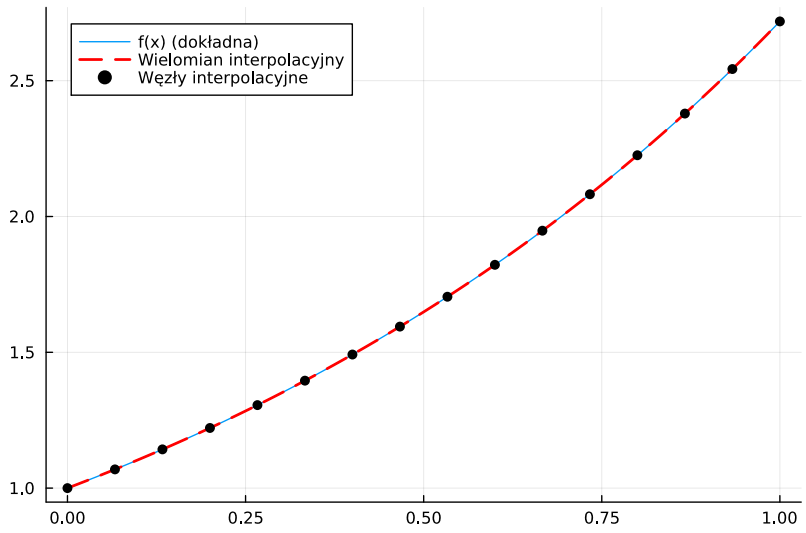
Podpunkt 1



Rysunek 1: Interpolacja funkcji $f(x) = e^x$ dla $n = 5$

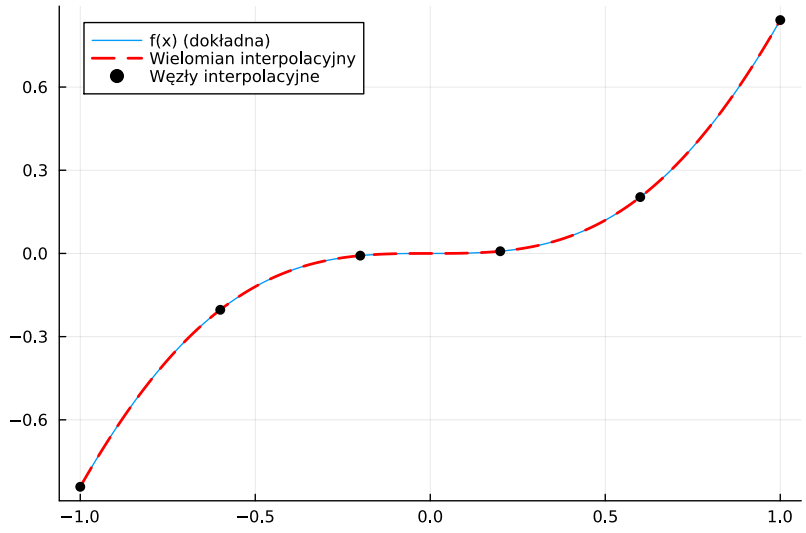


Rysunek 2: Interpolacja funkcji $f(x) = e^x$ dla $n = 10$

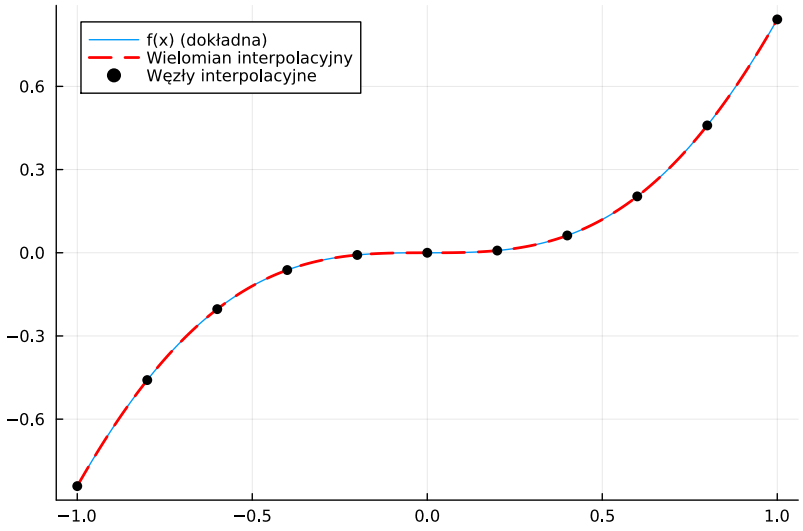


Rysunek 3: Interpolacja funkcji $f(x) = e^x$ dla $n = 15$

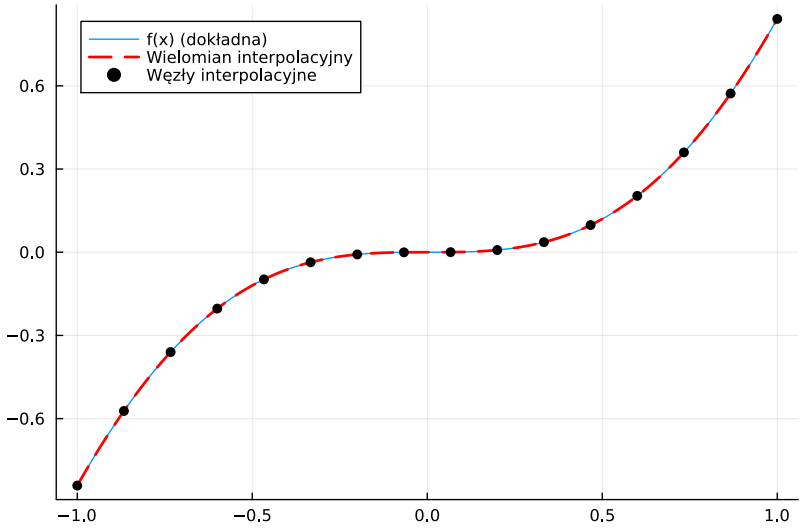
Podpunkt 2



Rysunek 4: Interpolacja funkcji $f(x) = x^2 * sin(x)$ dla $n = 5$



Rysunek 5: Interpolacja funkcji $f(x) = x^2 * sin(x)$ dla $n = 10$



Rysunek 6: Interpolacja funkcji $f(x) = x^2 * sin(x)$ dla $n = 15$

Interpretacja

Wielomiany interpolujące pokrywają się z podanymi funkcjami na wykresach. Wynika to z faktu, że funkcje te są ciągłe oraz gładkie czyli wszystkie ich pochodne także są ciągłe. Dzięki zwiększeniu n błąd interpolacji spada. Błędy tych samych rzędów wychodzą dla wszystkich punktów w przedziale interpolacji.

Wnioski

Wielomiany interpolujące w bardzo dobry sposób przybliżają funkcje ciągłe, gładkie, różniczkowalne z dobrze zachowującą się pochodną na zadanym przedziale.

Zadanie 6

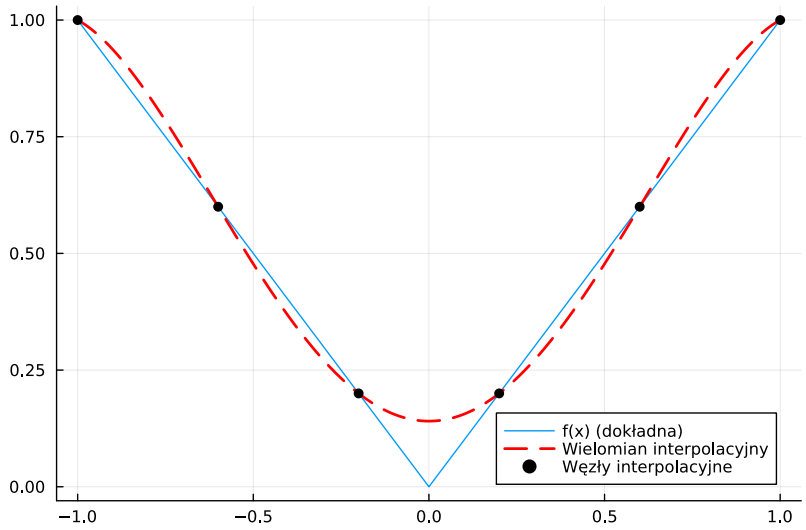
Opis zadania

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

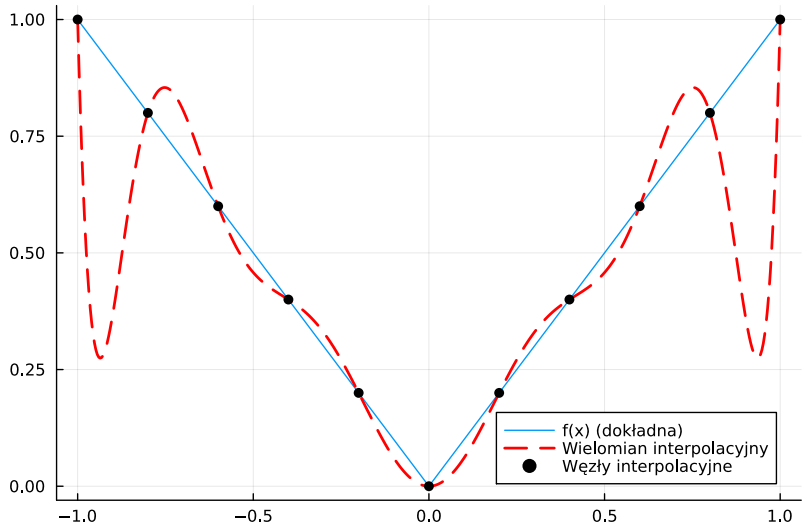
- $|x|$, $[1,1]$, $n = 5, 10, 15$
- $\frac{1}{1+x^2}$, $[5,5]$, $n = 5, 10, 15$ (zjawisko Runge'go)

Wyniki i interpretacja

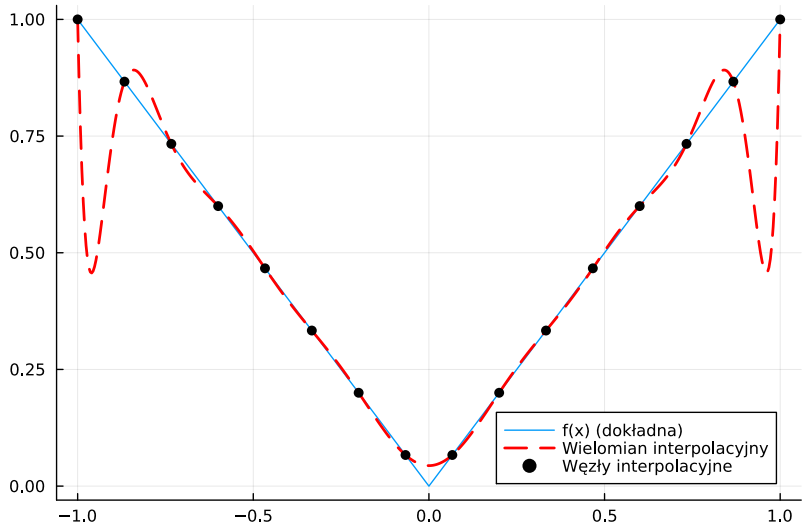
Podpunkt 1



Rysunek 7: Interpolacja funkcji $f(x) = |x|$ dla $n = 5$

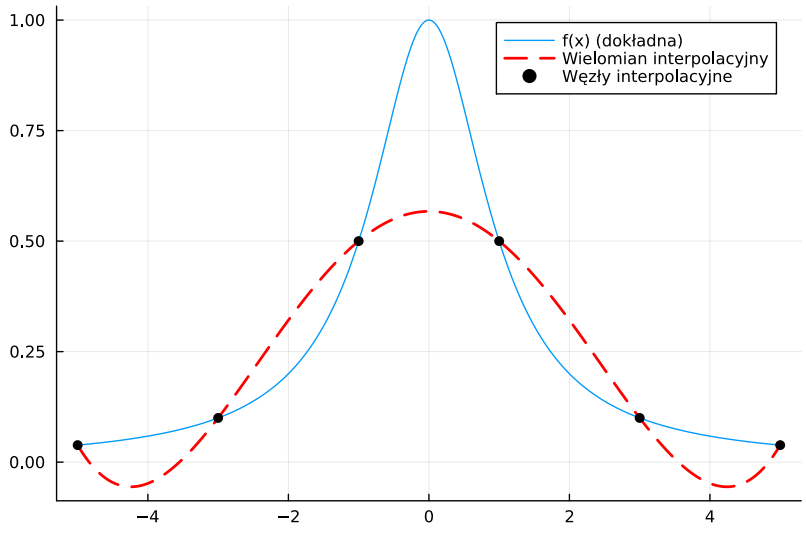


Rysunek 8: Interpolacja funkcji $f(x) = |x|$ dla $n = 10$

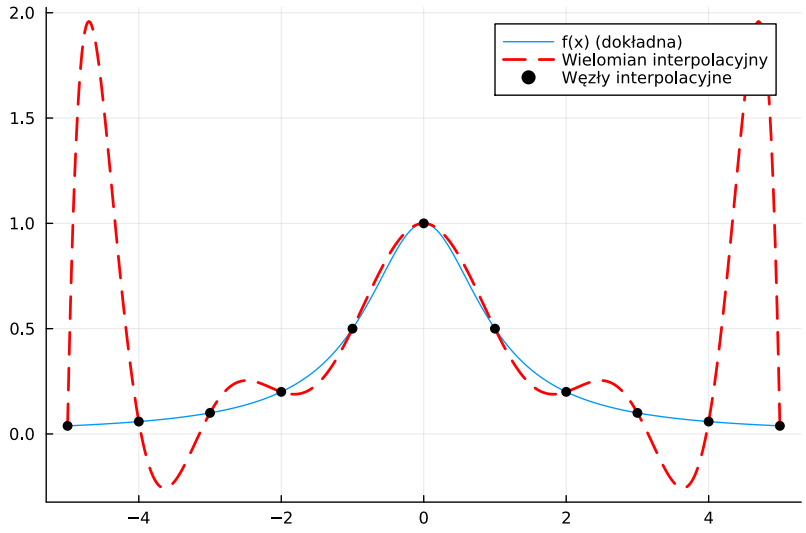


Rysunek 9: Interpolacja funkcji $f(x) = |x|$ dla $n = 15$

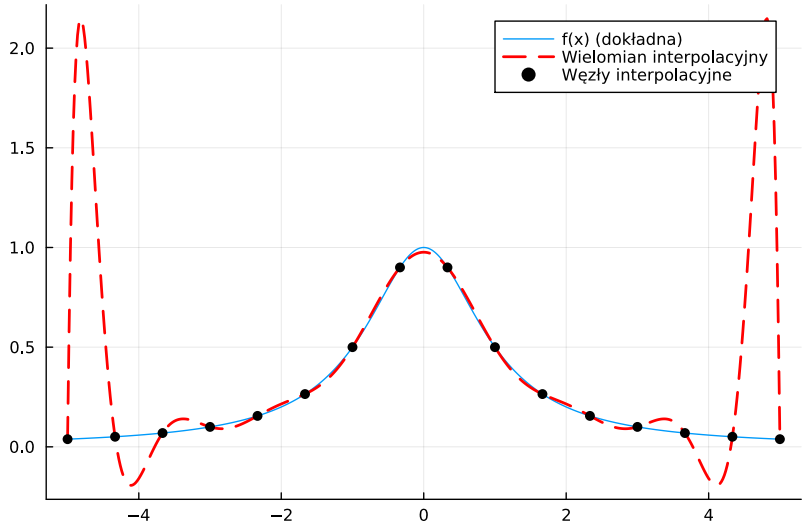
Podpunkt 2



Rysunek 10: Interpolacja funkcji $f(x) = \frac{1}{1+x^2}$ dla $n = 5$



Rysunek 11: Interpolacja funkcji $f(x) = \frac{1}{1+x^2}$ dla $n = 10$



Rysunek 12: Interpolacja funkcji $f(x) = \frac{1}{1+x^2}$ dla $n = 15$

Interpretacja

Wielomiany interpolacyjne w tym zadaniu bardzo słabo przybliżają zadane funkcje. Zwiększenie liczby punktów interpolujących na danym przedziale poprawia jedynie środkową część wykresu. Jednocześnie pogarsza to przybliżenia na końcach przedziałów. Jest to zjawisko Runge'go. Wynika to z zmiany znaku pochodnej na badanych przedziałach. Problemem jest także równa odległość pomiędzy punktami.

Wnioski

Aby poprawić wyniki przybliżeń w tym zadaniu należy zagęścić punkty na końcach przedziałów. Jednym z rozwiązań problemu jest użycie do przybliżeń wielomianów węzłów Czebyszewa, które mają więcej węzłów na końcach przedziałów.