Obliczenia Naukowe - Laboratorium 4

Michał Waluś 279695

Grudzień 2025

1 Zadanie 1

1.1 Opis problemu

W języku Julia napisanie funkcji obliczającej ilorazy różnicowe.
`function ilorazyRoznicowe (x::VectorFloat64, f::VectorFloat64)` gdzie:

- `x` - wektor zawierający węzły
- `f` - wektor zawierający wartości w węzłach

1.2 Algorytm

Wiemy, że ilorazy różnicowe spełniają następującą równość:

$$f[x_0, \ldots, x_n] = \frac{f[x_1, \ldots, x_n] - f[x_0, \ldots, x_{n-1}]}{x_n - x_0} \tag{1}$$

Zatem możemy prosto konstruować ilorazy różnicowe używając tabeli trójkątnej:

$$\begin{array}{cccccc}
f[x_n] \searrow & & & & & \\
f[x_{n-1}] \rightarrow & f[x_{n-1}, x_n] \searrow & & & & \\
f[x_{n-2}] \rightarrow & f[x_{n-2}, x_{n-1}] \rightarrow & f[x_{n-2}, x_{n-1}, x_n] \searrow & & & \\
\vdots & \vdots & \vdots & \ddots & & \\
f[x_0] \rightarrow & f[x_0, x_1] \rightarrow & f[x_0, x_1, x_2] \rightarrow & \cdots & f[x_0 \ldots, x_n]
\end{array}$$

Tabela 1: Tabela trójkątna zależności między ilorazami różnicowymi.

W powyższej tabeli strzałki obrazują zależności, do wyznaczenia których ilorazów różnicowych, ze wzoru (1), potrzebna jest dana wartość. Ponieważ $f[x_i] = f(x_i)$, to widzimy, że do wyznaczenia wartości w danym rzędzie potrzebujemy tylko tych z poprzedniego, a wszystkie wcześniejsze możemy zapomnieć. Korzystając z tego możemy stworzyć algorytm liczący ilorazy różnicowe w miejscu: (zakładamy indeksowanie od 1)

Oba argumenty oraz zmienne wewnątrz algorytmu są wielkości liniowej względem liczby podanych wartości, zatem algorytm ma liniową złożoność pamięciową ($O(n)$).

Operacje przypisania wartości oraz odczytania wartości z tablicy ma stałą złożoność czasową, a operacja przekopiowania tablicy liniową, zatem algorytm ma kwadratową złożoność czasową ($O(n^2)$).

```
Function ilorazyRoznicowe(x[n], f[n]):
    Define cur[n], prev[n];
    for i = 1 to n do
        cur[1] ← f[n − i + 1];
        for j ← 2 to i do
            cur[j] ← (cur[j − 1] − prev[j − i])/(x[n − i + 1] − x[n − i + j]);
        end
        prev ← cur;
    end
    Return cur;
end
```

Algorithm 1: Wyznaczanie ilorazów różnicowych

2 Zadanie 2

2.1 Opis problemu

W języku Julia napisanie funkcji obliczającej wartości wielomianu interpolacyjnego stopnia n w postaci Newtona.

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

- `x` - wektor zawierający węzły
- `fx` - wektor zawierający ilorazy różnicowe, `fx[i]` $= f[x_0, \ldots, x_{i-1}]$
- `t` - punkt, w którym należy obliczyć wartość wielomianu

2.2 Algorytm

Dla uproszczenia zapisu przymujemy:

$$X_k = \prod_{j=0}^{k-1}(x - x_j) \quad \text{oraz} \quad a_n = f[x_0, \ldots, x_k]$$

Dany wielomian p jest zapisany w postaci Newtona, zatem:

$$p(x) = \sum_{k=0}^{n} \left(f[x_0, \ldots, x_k] \prod_{j=0}^{k-1} (x - x_j) \right) = \sum_{k=0}^{n} a_k X_k$$

Bazując na algorytmie Hornera dzielimy p przez $(x - t)$, żeby wyznaczyć resztę równą $p(t)$. Otrzymujemy w ten sposób:

$$\begin{aligned}
p(x) &= (x - t) \left(\sum_{k=0}^{n-1} b_k X_k \right) + p(x_0) \\
&= \sum_{k=0}^{n-1} (x - x_k + x_k - t) b_k X_k + p(x_0) \\
&= \sum_{k=0}^{n-1} \left(b_k X_{k+1} + (x_k - t) b_k X_k \right) + p(x_0) \\
&= b_{n-1} X_n + \sum_{k=1}^{n-1} \left(\left(b_k + (x_{k+1} - t) b_{k+1} \right) X_{k+1} + (p(x_0) + (x_0 - t) b_0 \right)
\end{aligned}$$

Porównując z oryginalną formą wielomianu otrzymujemy wzory na współczynniki $(b_k)_{k=0}^{n-1}$ oraz $p(x_0)$:

$$b_{n-1} = a_n$$
$$(\forall k \in \{-1, \ldots, n-2\}) b_k = a_{k+1} + (t - x_{k+1}) b_{k+1}$$
$$\text{gdzie } p(x_0) = b_{-1}$$

Korzystając z powyższych wzorów nietrudno zaproponować algorytm o liniowej złożności czasowej:

Function `warNewton`($x[n]$, $fx[n]$, t)**:**
  Define $b[n]$;
  $b[n] \leftarrow fx[n]$;
  for $k \leftarrow n-1$ **down to** 1 **do**
    $b[k] \leftarrow fx[k] + (t - x[k]) \cdot b[k+1]$;
  end
  Return $b[1]$;
end

Algorithm 2: Wyznaczanie wartości wielomianu

Nietrudno też zauważyć, że algorytm ma również liniową złożoność pamięciową.

3 Zadanie 3

3.1 Opis problemu

W języku Julia napisanie funkcji wyliczającej współczynniki $(a_k)_{k=0}^{n}$ wielomianu w formie normalnej, mając dany wielomian interpolacyjny w formie Newtona.

`function naturalna (x::VectorFloat64, fx::VectorFloat64)`, `x` oraz `fx` są takie same jak w Zadaniu 2.

3.2 Algorytm

Wykonując te same operacje co w poprzednim zadaniu, dla $t = 0$ otrzymujemy wartość wielomianu w 0 (czyli a_0) oraz współczynniki $(b_k)_{k=0}^{n-1}$ wyznaczające wielomian $p_1(x)$ stopnia (n - 1) spełniający $p(x) = xp_1(x) + a_0$. Stosując tą samą metodę na p_1 otrzymamy kolejny wielomian p_2 oraz a_1. Powtarzamy to łącznie $n + 1$ i otrzymujemy wszystkie współczynniki.

Function `naturalna`($x[n]$, $fx[n]$)**:**
  Define $cur[n]$;
  $cur \leftarrow fx$;
  for $m \leftarrow 1$ **to** n **do**
    for $k \leftarrow n-1$ **down to** m **do**
      $cur[k] \leftarrow cur[k] - x[k-m+1] \cdot cur[k+1]$;
    end
  end
  Return cur;
end

Algorithm 3: Wyznaczanie postaci naturalnej

W algorytmie dostosowaliśmy współczynniki, by w każdej pętli poprawnie wykorzystywał pierwsze $n-(m-1)$ elementów x, a fakt, pod koniec każdej kolejnej iteracji tablica *cur* zawiera współczynniki b_i na ostatnich miejscach, pozwala nam pracować tylko na jednej tablicy pomocniczej.

Wewnętrzna pętla oraz operacja kopiawania tablic mają liniowe złożoności czasowe, zatem cały algorytm ma złożoność czasową kwadratową ($O(n^2)$) oraz liniową pamięciową ($O(n)$).

4 Zadanie 4

4.1 Opis problemu

W języku Julia napisać funkcje interpolującą wielomian oraz rysującą jego wykres oraz rysującą wykres oryginalnej funkcji, nie przekształcając do formy naturalnej.

`function rysujNnfx(f,a::Float64,b::Float64,n::Int; wezly::Symbol = :rownoodlegle)`, gdzie:

- `f` - funkcja $f(x)$.
- `a`, `b` - krańce przedziału.
- `n` - stopień wielomianu interpolacyjnego.
- `wezly` - rodzaj użytych węzłów:
  - `:rownoodlegle` - $(n+1)$ równoodległych węzłów z przedziału $[a,b]$.
  - `:czebyszew` - pierwiastki $(n+1)$-ego wielomianu Czebyszewa.

4.2 Algorytm

Węzły interpolacyjne wyznaczamy następująco:

- Równoodległe:
$$x_i = a + \frac{b-a}{n} \cdot k, \text{ dla } k = 0, \ldots, n$$
- Pierwiastki wielomianu Czebyszewa:
$$x_i = \frac{a+b}{2} + \frac{b-a}{2} \cdot \cos\left(\frac{2k+1}{2n}\pi\right), \text{ dla } k = 0, \ldots, n$$

Postać pierwiastków wielomianu Czebyszewa, wynika z tego, że n-ty wielomian Czebyszewa jest $(n-1)$ stopnia i możemy go zapisać jako

$$T_n(x) = \cos(n \arc\cos(x)) \tag{2}$$

oraz z faktu, że $\cos(\frac{\pi}{2}) = 0$, a także że wielomian k-tego stopnia ma co najwyżej k pierwiastków rzeczywistych, zatem widzimy, że wartości postaci $\cos\left(\frac{2k+1}{2n}\pi\right)$ są jedynymi pierwiastkami. Ponieważ w przedziale $[-1,1]$ wielomiany Czebyszewa przyjmują tylko wartości z tego samego przedziału, to mnożąc przez czynnik $\frac{b-a}{2}$ oraz dodając składnik $\frac{a+b}{2}$ "przesuwamy"te wartości, tak by wszystkie znajdowały się w przedziale $[a,b]$, ponieważ w przeciwnym wypadku wyniki interpolacji byłyby bardziej oddalone od oryginalnej funkcji na zadanym przedziale.

Wartości w węzłach interpolacyjnych wyznaczamy, korzystając z podanej funkcji, a współczynniki obliczamy korzystając z funkcji z Zadania 1 - `ilorazyRoznicowe`.

Do narysowania wykresu korzystamy z pakietu `Plots`, jako argumenty przyjmujemy 10000 równoodległych liczb z przedziału $[a,b]$, a do obliczenia wartości wielomianu interpolacyjnego w tych punktach używamy funkcji z Zadania 2 - `warNewton`.

5 Zadanie 5

5.1 Opis problemu

Przetestować funkcję `RysujNnfx` dla węzłów równoległych dla następujących parametrów:

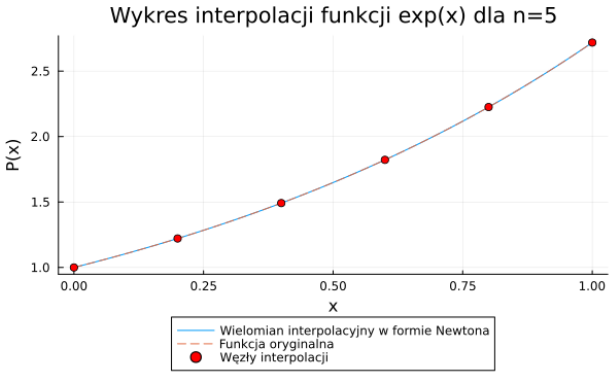
1. $f(x) = e^x$, $[a, b] = [0, 1]$, $n = 5, 10, 15$.
2. $f(x) = x^2 \sin(x)$, $[a, b] = [-1, 1]$, $n = 5, 10, 15$.

5.2 Rozwiązanie

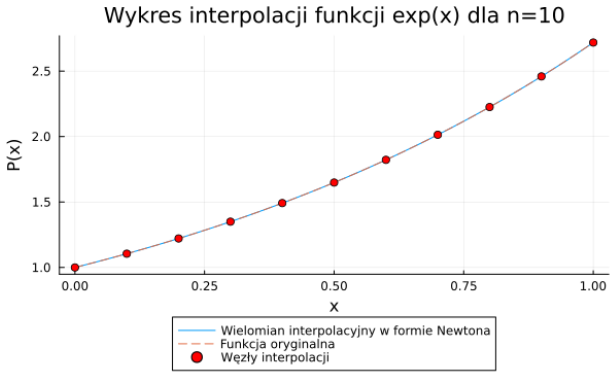
Wykonujemy funkcję 6 razy dla podanych parametrów.

5.3 Wyniki

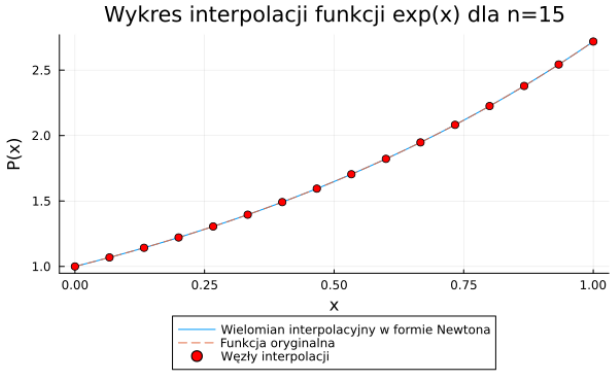
Wykresy dla przypadków 1 i 2, przedstawione na odpowiednio Rysunku 1 i 2.



(a) $n = 5$



(b) $n = 10$



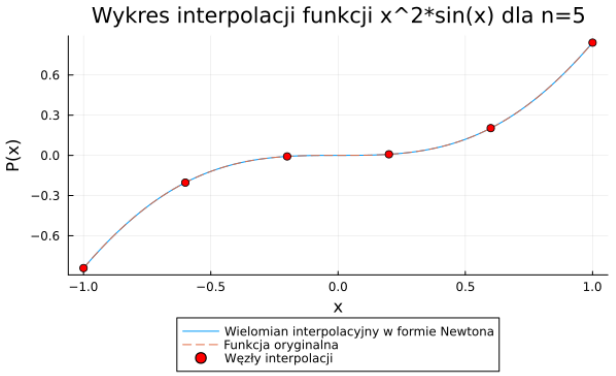
(c) $n = 15$

Rysunek 1: Wykresy dla e^x

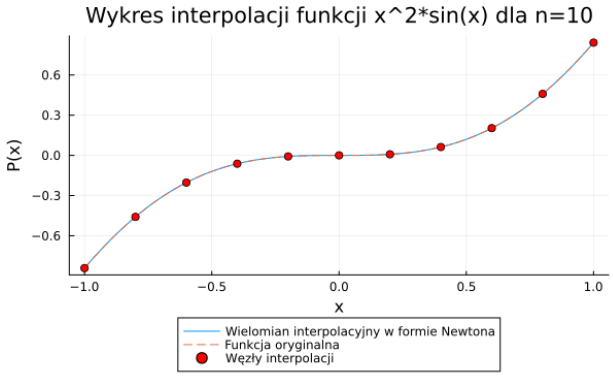
Na wszystkich wykresach funkcja oryginalna (zaznaczona czerowną przerywaną linią), pokrywa się z wielomianem interpolacyjnym.

5.4 Wnioski

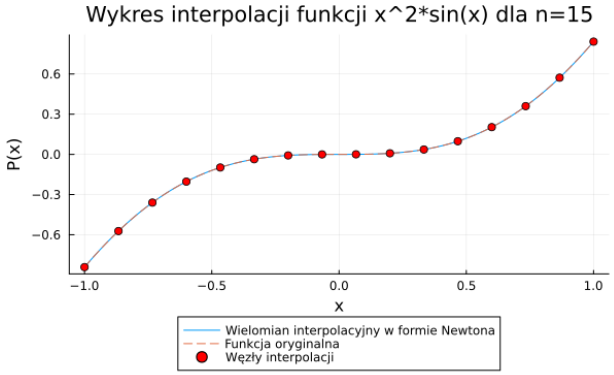
Dla prostych przykładów na małym przedziale nasza funkcja poprawnie interpoluje, nawet dla małej liczby węzłów.



(a) $n = 5$



(b) $n = 10$



(c) $n = 15$

Rysunek 2: Wykresy dla $x^2 \sin(x)$

6 Zadanie 6

6.1 Opis problemu

Przetestować funkcję `RysujNnfx` dla węzłów równoległych oraz będących pierwiastkami wielomianów Czebyszewa, dla następujących parametrów:

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

6.2 Rozwiązanie

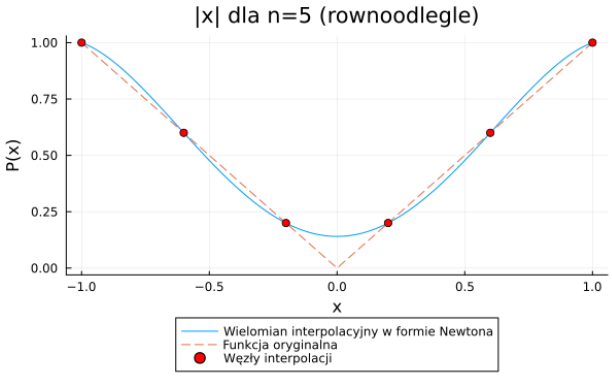
Wykonujemy funkcję łącznie 12 razy dla podanych parametrów.

6.3 Wyniki

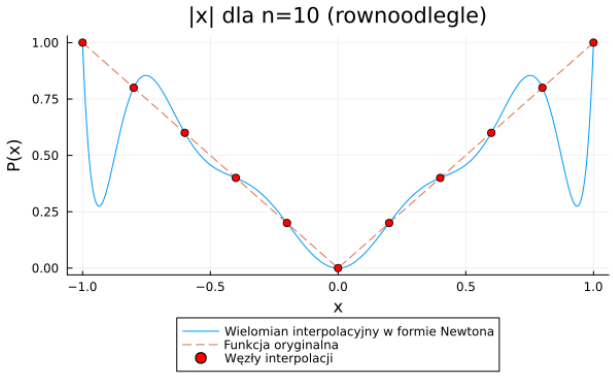
Wykresy dla przypadków 1 i 2 zostały przedstawione na Rysunkach 3 i 4 odpowiednio.

Widzimy, że dla obu tych funkcji metoda interpolacji, dla tak małej liczby węzłów, zwraca wielomiany oddalone od oryginalnej funkcji. Widzimy, że dla tych przykładów używanie pierwiastków wielomianów Czebyszewa daje lepsze wyniki, zdecydowanie bliższe oryginalnej funkcji.

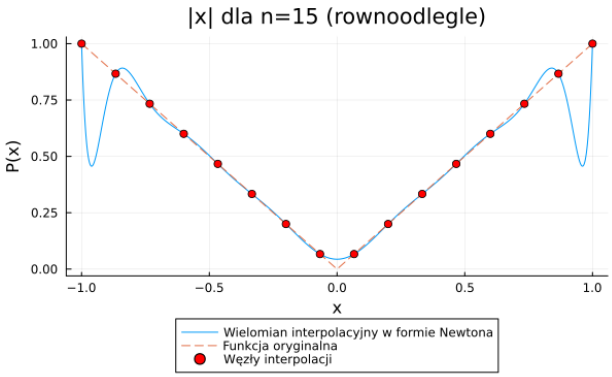
Dla równoodległych węzłów, widzimy, że dla funkcji $|x|$, najlepszy wykres uzyskaliśmy dla $n = 5$, a dla obu funkcji dla $n = 10$ i $n = 15$ otrzymaliśmy "wybrzuszenia"blisko końców przedziału.



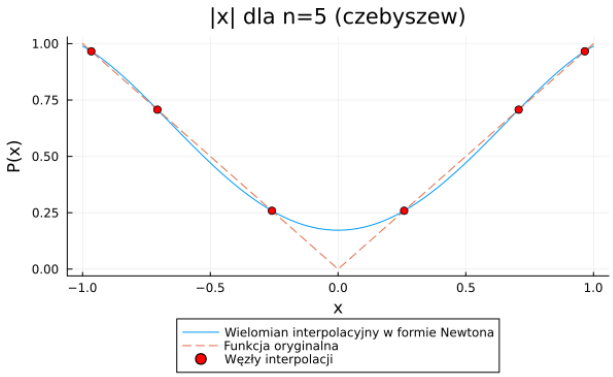
(a) $n = 5$, równoodległe



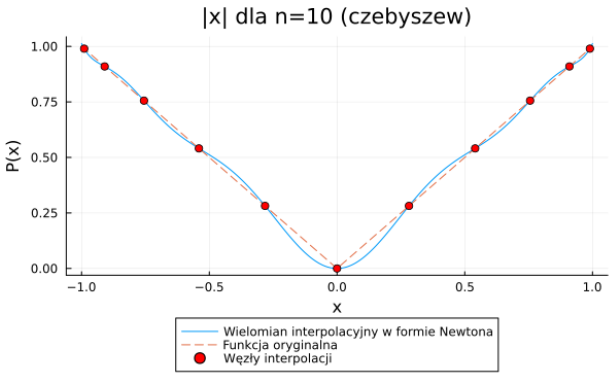
(b) $n = 10$, równoodległe



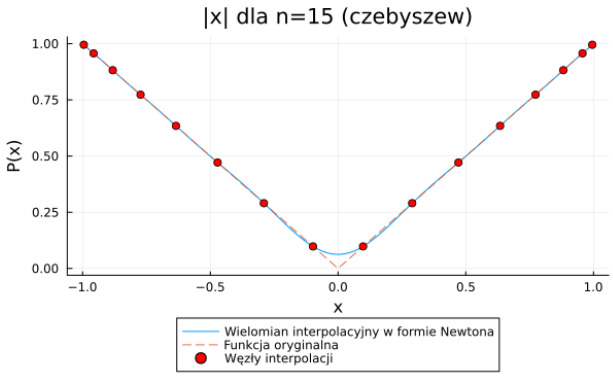
(c) $n = 15$, równoodległe



(d) $n = 5$, Czebyszew



(e) $n = 10$, Czebyszew



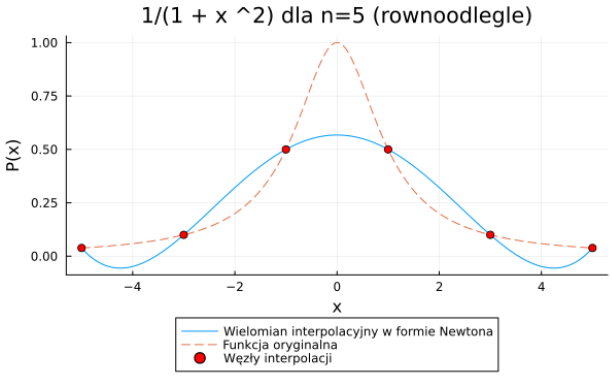
(f) $n = 15$, Czebyszew

Rysunek 3: Wykresy dla $|x|$

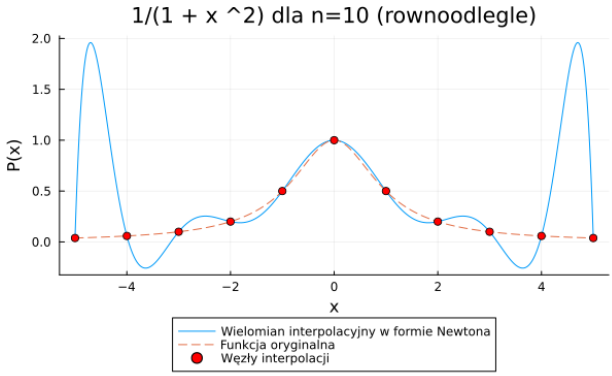
Dla funkcji $\frac{1}{x^2+1}$ dziwne zachowanie interpolacji jest związane z ograniczeniem górnym dokładności interpolacji z użyciem n punktów będącym zależnym od n-tej pochodnej oryginalnej funkcji:

$$|f(x) - p(x)| \leqslant \frac{1}{n!} f^n(\zeta_x) \prod_{i=0}^{n-1} (x - x_i)$$

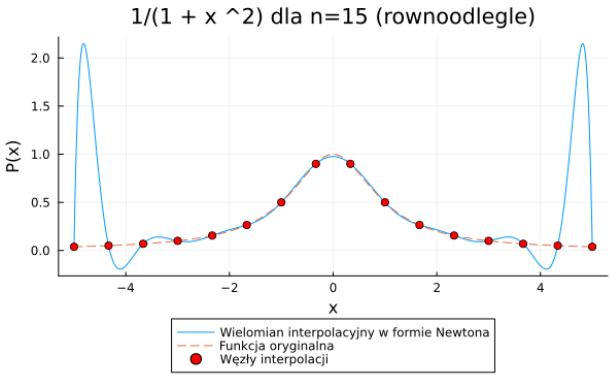
Dla tej funkcji maksymalna wartość pochodnej na przedziale $[-5, 5]$ szybko się zwiększa, dla $n = 10$ jest to około $2.4 \cdot 10^6$, a dla $n = 15$ około $1.2 \cdot 10^{12}$ (wartości uzyskane z użyciem wolfram alpha).



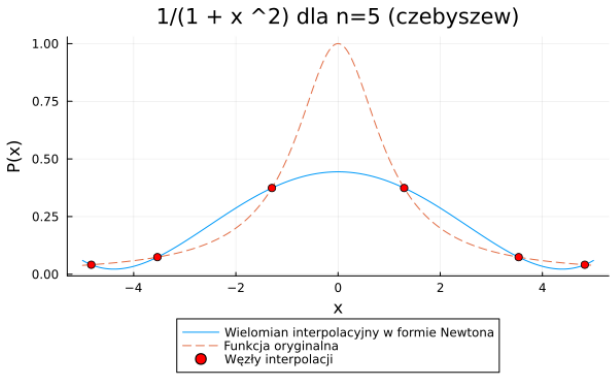
(a) $n = 5$, równoodległe



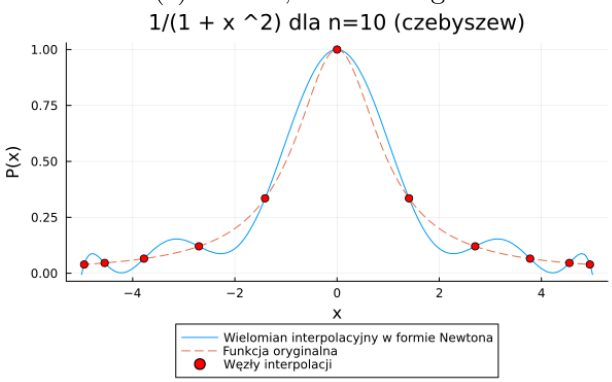
(b) $n = 10$, równoodległe



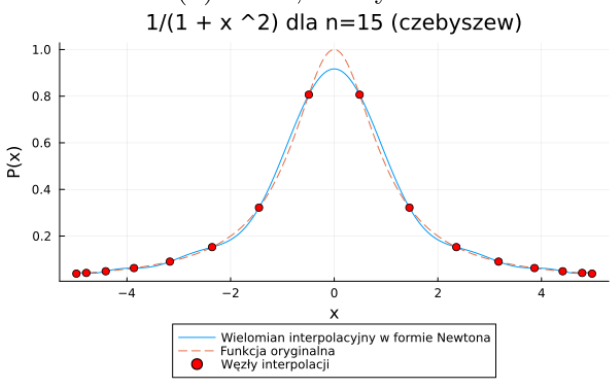
(c) $n = 15$, równoodległe



(d) $n = 5$, Czebyszew



(e) $n = 10$, Czebyszew



(f) $n = 15$, Czebyszew

Rysunek 4: Wykresy dla $\frac{1}{1+x^2}$

6.4 Wnioski

Dla funkcji trudnych do przybliżenia wielomianem używanie pierwiastków wielomianów Czebyszewa zdaje się dawać lepsze rezultaty. Używanie ich pozwala łagodzić zjawisko Runge'go, które występuje gdy wartość pochodnej szybko rośnie i zwiększanie liczby węzłów nie poprawia dokładności przybliżenia funkcji, a nawet może je pogorszyć.