Obliczenia naukowe
Sprawozdanie
Lista 4

Mateusz Laskowski

09.12.2018

1. Zadanie 1

1.1. Opis problemu

Napisać funkcję obliczająca ilorazy różnicowe. Zaimplementować funkcję bez użycia tablicy dwuwymiarowej, czyli macierzy.

1.2. Opis rozwiązania

Model funkcji:

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

Dane wejściowe:

x – wektor długości $n+1$ zawierający węzły $x_0, \dots, x_n$, gdzie
$x[1] = x_0, \dots, x[n+1] = x_n$,
f – wektor długości $n+1$ zawierających wartości interpolowanej funkcji w węzłach $f(x_0), \dots, f(x_n)$

Dane wyjściowe (fx):

fx – wektor długości $n+1$ zawierający obliczone ilorazy różnicowe
$fx[1] = f[x_0]$,
$fx[2] = f[x_0, x_1], \dots, fx[n] = f[x_0, \dots, x_{n-1}], fx[n+1] = f[x_0, \dots, x_n]$

Analiza:

Implementacja metody wyliczania ilorazów różnicowych to pierwszy krok do stworzenia algorytmu aproksymującego funkcję metodą Newtona. Podstawowy wzór rekurencyjny, który jest potrzebny do stworzenia odpowiedniego algorytmu:

$$f([x_k, x_{k+1}, \dots, x_{k+m}]) = \frac{f[x_{k+1}, x_{k+2}, \dots, x_{k+m}] - f[x_k, x_{k+1}, \dots, x_{k+m-1}]}{x_{k+m} - x_k}$$

Wyżej wypisany wzór rekurencyjny, rozwijamy do postaci ilorazu pojedynczego węzła, który jest znany i wynosi $f[x_0] = f(x_0)$. W tablicy trójkątnej można zobrazować wartości ilorazów potrzebne do wyliczenia interesującego nas ilorazu. Przykładowo dla $f[x_0, x_1, x_2, x_3]$, aby wyliczyć iloraz w komórce (x, y) potrzebne są wartości z komórek $(x-1, y)$ oraz $(x-1, y-1)$. Wartości $f[x_k] = f(x_k)$ znane są, więc postępując od lewej strony tablicy trójkątnej, można wypełnić całą tablicę.

x_0	$f[x_0]$	$f[x_0x_1]$	$f[x_0x_1x_2]$	$f[x_0x_1x_2x_3]$
x_1	$f[x_1]$	$f[x_1x_2]$	$f[x_1x_2x_3]$	
x_2	$f[x_2]$	$f[x_2x_3]$		
x_3	$f[x_3]$			

Z powyższej tablicy można rozpisać wielomian:

$$w(x) = f[x_0] + f[x_0x_1](x-x_0) + f[x_0x_1x_2](x-x_0)(x-x_1) + f[x_0x_1x_2x_3](x-x_0)$$
$$(x-x_1)(x-x_2)$$

W zadaniu trzeba zaimplementować funkcję, która będzie zwracała wektor ilorazów. Na wyżej podanym przykładzie z tablicy trójkątnej, powinniśmy zwrócić wektor $(f[x_0], f[x_0x_1], f[x_0x_1x_2], f[x_0x_1x_2x_3])$.

1.3. Implementacja

Założenie zadanie wymagało, aby przy wyliczeniach ilorazów nie używać tablicy dwuwymiarowej, więc w mojej implementacji posłużyłem się wzorem rekurencyjnym, wypisanym w analizie problemu.

2. Zadanie 2

2.1. Opis problemu

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 $O(n)$.

2.2. Rozwiązanie problemu

Model funkcji:

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

Dane wejściowe:

x – wektor długości $n + 1$ zawierający węzły $x_0, \dots, x_n$, gdzie
$x[1] = x_0, \dots, x[n + 1] = x_n$,
fx – wektor długości $n + 1$ zawierający obliczone ilorazy różnicowe
$fx[1] = f[x_0]$,
$fx[2] = f[x_0, x_1], \dots, fx[n] = f[x_0, \dots, x_{n-1}], fx[n + 1] = f[x_0, \dots, x_n]$
t – punkt, w którym należy obliczyć wartość wielomianu

Dane wyjściowe (nt):

nt – wartość wielomianu w punkcie t

Analiza:

Uogólniony algorytm Hornera pozwala na wyliczenie wartości wielomianu interpolacyjnego Newtona:

$$w_n(x) = f[x_0, x_1, \dots, x_n]$$
$$w_k(x) = f[x_0, x_1, \dots, x_k] + (x + x_k)w_{k+1}, \qquad gdzie(k = n - 1, \dots, 0)$$
$$N_n(x) = w_0(x)$$

Aby wyliczyć daną wartość $N_n(x)$ trzeba rozwinąć wzór rekurencyjny:

$$N_n(x) = f[x_0, x_1, \dots, x_k] + (x - x_k)(f[x_0, x_1, \dots, x_{k+1}] + (x - x_{k+1})w_{k+2})$$

$$\begin{aligned} &= f[x_0, x_1, \dots, x_k] \\ &\qquad + (x - x_k)\big(f[x_0, x_1, \dots, x_{k+1}] \\ &\qquad + (x - x_{k+1})(f[x_0, x_1, \dots, x_{k+2}] + (x - x_{k+2})w_{k+3})\big) \\ &= \ \dots \\ &= f[x_0] + f[x_0 x_1](x - x_0) + \cdots + f[x_0, \dots, x_n](x - x_0) \dots (x - x_n) \end{aligned}$$

który zakończy się, gdy $k = n$, czyli dla $w_n(x)$, za *x* podstawiając wybrany argument.

2.3. Implementacja

Algorytm trzeba było zaimplementować w czasie $O(n)$, gdzie na wejściu otrzymuje wektor węzłów x oraz wektor ilorazów różnicowych fx. Algorytm generuje tablicę W długości n i wypełnia jej ostatnią komórkę wartością $W[n] = f[x_0, x_1, \dots, x_n]$ (wartość brana z wektora fx). W każdym kolejnym kroku pętli przeskakuję do poprzednich komórek, wypełniając je według schematu $W[k] = f[x_0, x_1, \dots, x_k] + (x - x_k) * W[k+1]$. Aby wypełnić całą tabele W trzeba wykonać tylko jedno przejście więc złożoność algorytmu jest liniowa.

3. Zadanie 3

3.1. Opis problemu

Napisać funkcję obliczającą w czasie $O(n^2)$ współczynniki $a_0, \ \dots, a_n$ postaci naturalnej wielomianu interpolacyjnego, tzn. $a_n x^n + a_{n-1}x^{n-1} + \cdots + a_1 x + a_0$.

Funkcja ma działać dla zadanych współczynników 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 węzłów $x_0, x_1, x_2, \dots, x_n$.

3.2. Rozwiązanie problemu

Model funkcji:

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

Dane wejściowe:

x – wektor długości $n+1$ zawierający węzły $x_0, \dots, x_n$, gdzie
$x[1] = x_0, \dots, x[n+1] = x_n$,

fx – wektor długości $n+1$ zawierający obliczone ilorazy różnicowe
$fx[1] = f[x_0]$,
$fx[2] = f[x_0, x_1], \dots, fx[n] = f[x_0, \dots, x_{n-1}], fx[n+1] = f[x_0, \dots, x_n]$

Dane wyjściowe:

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

Analiza:

Zadanie bazuje na wynikach z zadania 9 z listy 4 z ćwiczeń, gdzie przedstawione zostało, że za pomocą wielomianu w postaci Newtona oraz użyciu wielomianów pomocniczych z metody Hornera, można wyznaczyć współczynniki postaci naturalnej. Ponieważ przy każdym kroku algorytmu Hornera, każdy kolejny zależy tylko od dwóch współczynników z poprzedniej iteracji algorytmu Hornera.

3.3. Implementacja

Na początku tworzona jest tablica A[n], w której będą trzymane kolejne współczynniki. Zaczyna się od końca, czyli od A[n] podstawiając iloraz różnicowy z n-tego węzła. Kolejne współczynniki obliczamy, biorąc wielomian dla wcześniejszego elementu i wyliczając wielomian pomocniczy dla tego, kolejnego współczynnika. W każdym następnym kroku współczynniki z wielomianu pomocniczego zestawiamy z dotychczas wyliczonymi współczynnikami. Powtarzane jest dopóki nie dojdziemy do współczynnika a_0. Po przejściu całego algorytmu otrzymujemy tabelę A[n] będącą wektorem współczynników wielomianu postaci normalnej. W każdej iteracji algorytmu należy dokonać zestawienia do n współczynników, co daje złożoność $O(n^2)$, bo $O(n) * O(n)$.

4. Zadanie 4

4.1. Opis problemu

Napisać funkcję, która zinterpoluje zadaną funkcję $f(x)$ w przedziale $[a, b]$, za pomocą wielomianu interpolacyjnego stopnia n w postaci Newtona. Następnie wygenerować wielomian interpolacyjny i interpolowaną funkcję, przy pomocy pakietu Plots, PyPlot lub Gadfly. Do interpolacji należy użyć węzłów równoodległych, tzn. $x_k = a + kh$,
$k = \frac{b-a}{n}$, $k = 0, 1, \dots, n$.

4.2. Rozwiązanie problemu

Model funkcji:

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

Dane wejściowe:

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

Dane wyjściowe:

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

4.3. Implementacja

Kroki, które wykonuje algorytm przy interpolacji:

1) Generowanie wektora węzłów W
2) Wyliczanie wartości funkcji w węzłach, wpisywanie do tablicy T_{wynik}
3) Użycie funkcji ilorazyRoznicowe (W, T_{wynik}), które generuje wektor ilorazów $I_{ilorazy}$
4) Użycie funkcji function warNewton (W, $I_{ilorazy}$), które generuje tablicę N_{wynik}, wartości funkcji w zakresie (w którym zostanie wygenerowany wykres)
5) Za pomocą tablicy N_{wynik} generowanie wykresu wielomianu interpolacyjnego
6) Zestawienie generowanego wykresu z rzeczywistym wykresem interpolowanej funkcji

5. Zadanie 5

5.1. Opis problemu

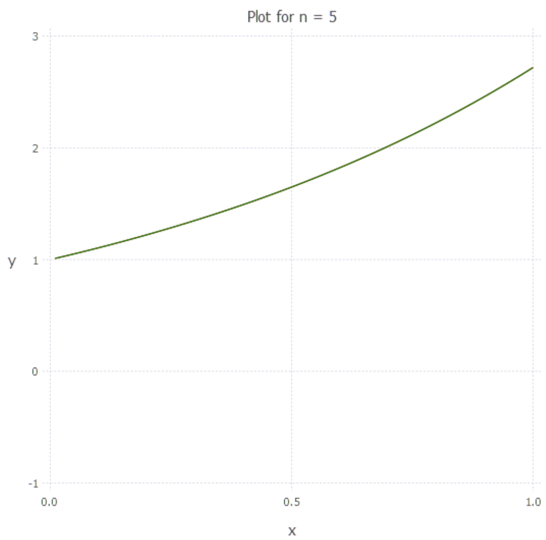
Wygenerować wykresy wielomianów interpolowanych (za pomocą wcześniej zaimplementowanych funkcji) z poniższych funkcji:

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

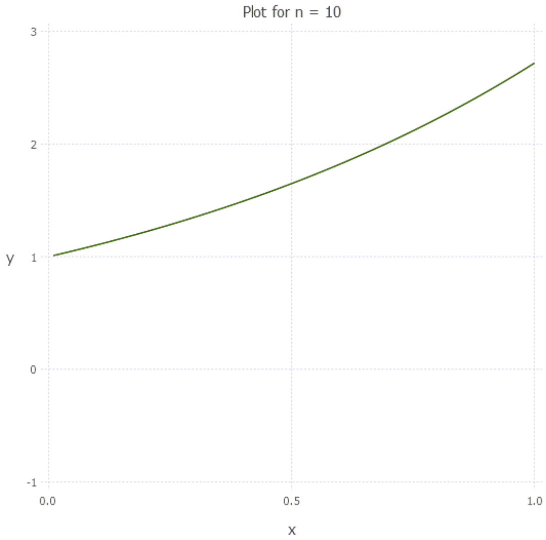
5.2. Rozwiązanie problemu

Wygenerowałem wykresy za pomocą funkcji rysujNnfx().

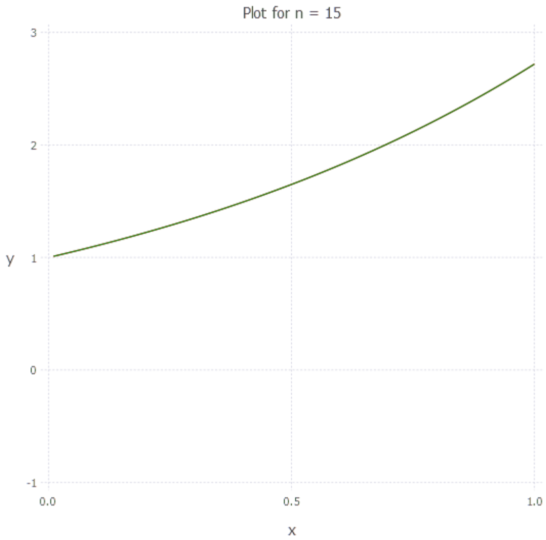
5.3. Wyniki



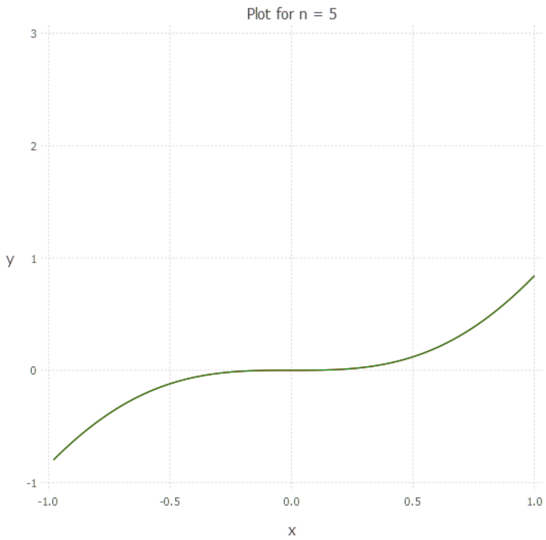
Ryc. 5.1 – $f(x) = e^x, [0,1], n = 5$



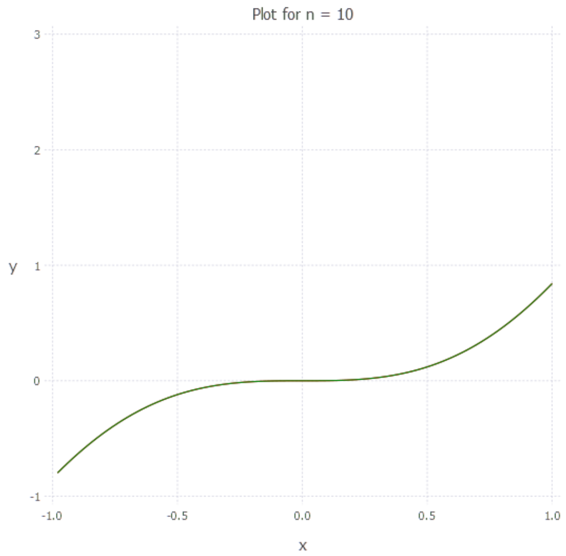
Ryc. 5.2 – $f(x) = e^x, [0,1], n = 10$



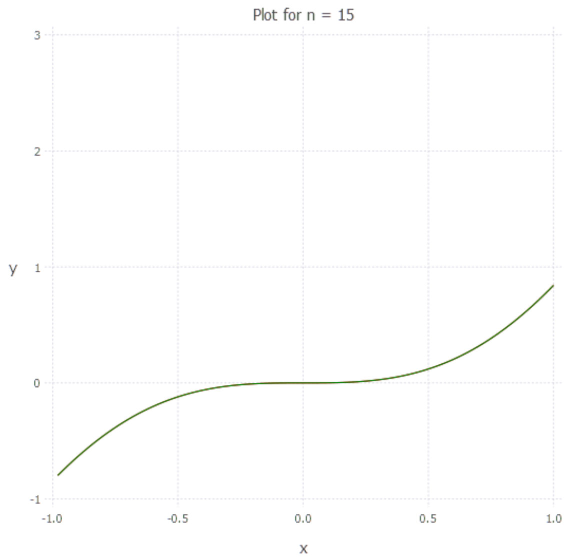
Ryc. 5.3 – $f(x) = e^x, [0,1], n = 15$



Ryc. 5.4 – $g(x) = x^2 sin(x), [-1, 1], n = 5$



Ryc. 5.5 – $g(x) = x^2 sin(x), [-1, 1], n = 10$



Ryc. 5.6 – $g(x) = x^2 sin(x), [-1, 1], n = 15$

5.4. Wnioski

Wykresy wielomianów będących interpolacją funkcji, pokryły się z wykresami tych funkcji. Interpolacja bardzo dobrze odwzorowała funkcje. Oczywiście trzeba mieć na uwadze, że wyliczenia są obarczone pewnymi małymi błędami obliczeń.

6. Zadanie 6

6.1. Opis problemu

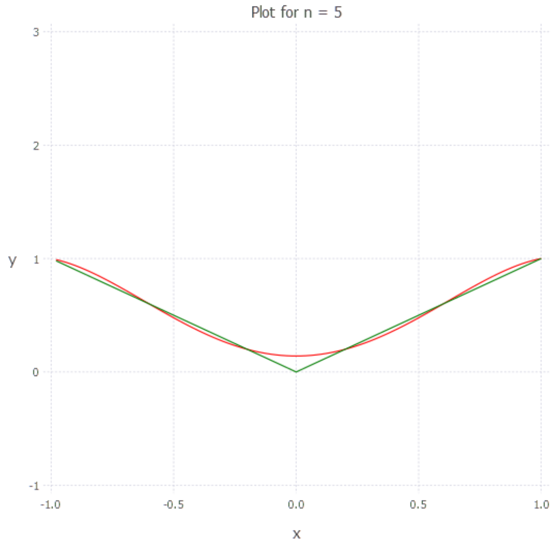
Wygenerować wykresy wielomianów interpolowanych (za pomocą wcześniej zaimplementowanych funkcji) z poniższych funkcji:

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

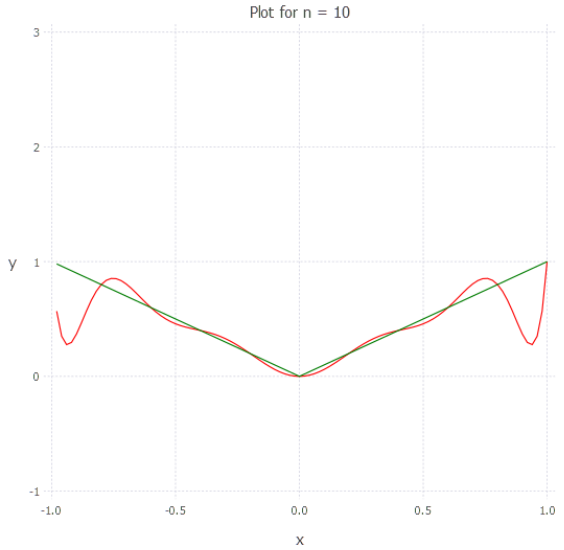
6.2. Rozwiązanie problemu

Wygenerowałem wykresy za pomocą funkcji rysujNnfx().

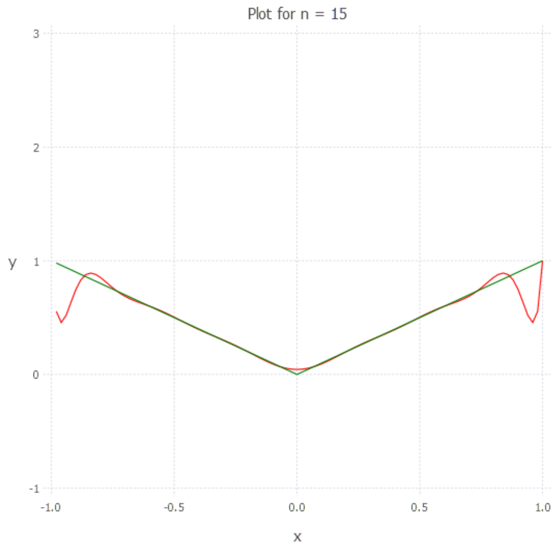
6.3. Wyniki



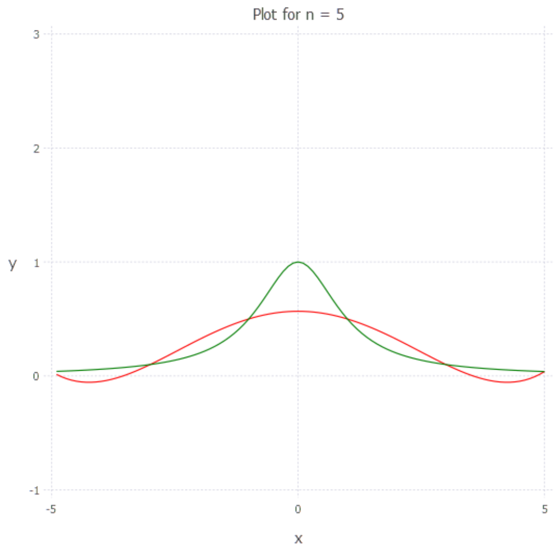
Ryc. 6.1 - $f(x) = |x|, [-1, 1], n = 5$



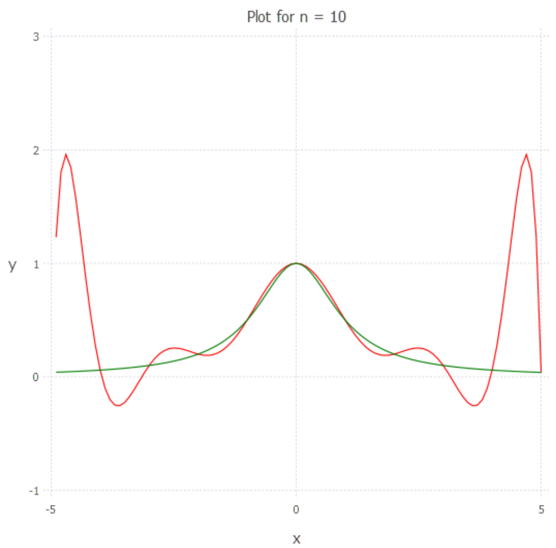
Ryc. 6.2 - $f(x) = |x|, [-1, 1], n = 10$



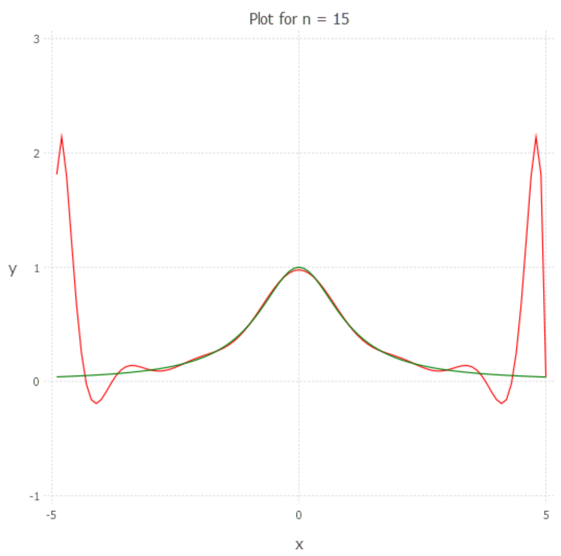
Ryc. 6.3 - $\mathrm{f(x)} = |\mathrm{x}|, [-1, 1], \mathrm{n} = 15$



Ryc. 6.4 - $g(x) = \frac{1}{1+\mathrm{x}^2}, [-5,5], n = 5$



Ryc. 6.5 - $g(x) = \frac{1}{1+\mathrm{x}^2}, [-5,5], n = 10$



Ryc. 6.6 - $g(x) = \frac{1}{1+x^2}, [-5{,}5], n = 15$

6.4. Wnioski

W tym zadaniu są funkcje, dla których wielomiany interpolacyjne są podatne na tzw. Efekt Rungego, czyli pogorszenie jakości interpolacji mimo zwiększania ilości węzłów. Efekt Ryngego występuje gdy:

- Interpolowana funkcja jest nieciągła
- Funkcja odbiega znacząco od funkcji gładkiej (funkcja gładka – funkcja ciągła mająca pochodne wszystkich rzędów)
- Wielomiany interpolujące mają wysokie stopnie, a odległość między węzłami jest stała

Funkcja $f(x) = |x|$ dla $n = 5$ interpolacyjna nie była dokładna w okolicach $x = 0$, lecz bliska $f(x)$ na krańcach przedziału. Po zwiększeniu ilości węzłów, wygenerowany wielomian był bardziej dokładny w centrum, lecz na krańcach pojawił się efekt Rungego, który pomimo ponownego zwiększenia ilości węzłów nie został zniwelowany. Głównym powodem wystąpienia efektu Rungego w tej funkcji jest brak ciągłości funkcji oraz równe odstępy między poszczególnymi węzłami.

W przypadku wielomianu będący interpolacją funkcji $g(x) = \frac{1}{1+x^2}$ dla $n = 5$ jest różny od rzeczywistego wykresu $g(x)$. Przy zwiększeniu ilości węzłów, interpolacja staje się bliższa rzeczywistej funkcji w okolicach $x = 0$, lecz na krańcach przedziału pojawia się jak w poprzednim przykładzie efekt Rungego. Zwiększając ilość węzłów efekt znów nie zniknął, jak w przykładzie funkcji $f(x)$. Powodem wystąpienia efektu Rungego w funkcji $g(x)$ jest nieciągłość funkcji oraz takie same odstępy między węzłami.

Można zniwelować błędy wynikające z efektu Rungego. Należy gęściej uwzględnić węzły na krańcach przedziału interpolowanej funkcji, wykorzystując wielomiany Czybyszczewa, których miejsca zerowe zagęszczają się na krańcach.