Krzysztof Dobrucki
Numer albumu 268507
Informatyka Algorytmiczna
Semestr Zimowy 2023/2024
Politechnika Wrocławska
05.12.2023r.

Lista 4

Obliczenia Naukowe

Ogólny opis problemu

Rozważamy problem interpolacji wielomianowej, który polega na znalezieniu wielomianu $p(x)$ stopnia nie wyższego niż n, taki że dla danych $n+1$ par (x_i, y_i), gdzie $x_i \neq x_j$, spełnione jest $\forall i\ p(x_i) = y_i$. Zgodnie z twierdzeniem przedstawionym na wykładzie, istnieje dokładnie jeden taki wielomian interpolacyjny stopnia co najwyżej n.

W procesie konstrukcji tego wielomianu wykorzystuje się wzór rekurencyjny:

$$p_{k+1}(x) = p_k(x) + c(x - x_0) \cdot \ldots \cdot (x - x_k)$$

dzięki któremu tworzymy wielomian stopnia o jeden wyższego niż poprzedni, jednocześnie zachowując ustalone wartości dla wcześniejszych x_i. Okazuje się, że ta forma jest użyteczna z perspektywy numerycznej i algorytmicznej, umożliwiając uniknięcie problemów z źle uwarunkowanymi macierzami Vandermonde'a i redukcję złożoności obliczeniowej w wyznaczaniu poszczególnych współczynników.

Formalnie, wielomian interpolacyjny ma postać:

$$p(x) = \sum_{i=0}^{n} c_i \prod_{j=0}^{i-1}(x - x_j)$$

To przedstawienie, znane jako wzór Newtona, jest używane w celu uzyskania wielomianu interpolacyjnego. W kolejnych krokach wyjaśnimy i wykonujemy operacje, aby uzyskać wielomian w tej postaci, a następnie jawnie wyznaczamy współczynniki przy poszczególnych potęgach.

Zadanie 1. - Ilorazy Różnicowe

Opis zadania

Aby uprościć notację, wprowadźmy oznaczenie:

$q_i(x) = \prod_{j=0}^{i-1}(x - x_j)$

z definicją $q_0(x) = 1$. Wówczas postać Newtona wielomianu p można zapisać jako:

$p(x) = \sum_{i=0}^{n} c_i q_i(x)$

Wielomian p jest rozwiązaniem problemu interpolacji, jeśli spełnia warunek:

$\forall k \in \{0, \ldots, n\} \quad \sum_{i=0}^{n} c_i q_i(x_k) = y_k$

Otrzymany w ten sposób układ równań ma postać:

$A \cdot C = Y$

gdzie $a_{ij} = q_j(x_i)$, $C = [c_0, \ldots, c_n]^T$, a $Y = [y_0, \ldots, y_n]^T$. Ponadto zauważmy, że $\forall i < j \; q_j(x_i) = 0$, co sprawia, że macierz A jest dolnotrójkątna. Ten fakt ułatwia znalezienie współczynników c_i poprzez rozwiązywanie układu od góry do dołu.

Wówczas łatwo zauważyć, że c_0 zależy od $y_0 = f(x_0)$, c_1 od y_0 i y_1, itd. Tę zależność oznaczymy jako $c_i = f[x_0, \ldots, x_i]$, nazywając ten czynnik ilorazem różnicowym funkcji f opartym na węzłach $x_0, \ldots, x_i$. Na wykładzie udało się udowodnić, że ilorazy różnicowe spełniają rekurencyjną zależność:

$f[x_i, \ldots, x_k] = f[x_i + 1, \ldots, x_k] - \frac{f[x_i, \ldots, x_k - 1]}{x_k - x_i}$

gdzie $f[x_i] = y_i$. W najprostszym podejściu do wyznaczania każdego ilorazu moglibyśmy użyć tablicy dwuwymiarowej C, gdzie $C[i, j] = f[x_i, \ldots, x_i + j]$. Jednak zauważmy, że po obliczeniu wszystkich ilorazów opartych na k węzłach, częściowe ilorazy oparte na $k - 1$ węzłach stają się niepotrzebne (z wyjątkiem $f[x_0, \ldots, x_k - 1]$, który jest jednym z poszukiwanych współczynników). Wydaje się zatem, że można w jakiś sposób zaoszczędzić pamięć potrzebną do rozwiązania zadania. Odpowiedzią na to wyzwanie jest algorytm zaprezentowany poniżej.

Zaczynamy z wektorem $\bar{d}$ wypełnionym wartościami interpolowanej funkcji w zadanych węzłach. Na wyjściu chcemy uzyskać wektor ilorazów różnicowych postaci $f[x_0, \ldots, x_i]$ będących współczynnikami wielomianu w wzorze Newtona. Zauważmy, że element na pierwszym miejscu w $\bar{d}$, czyli d_0, jest już w odpowiedniej postaci. Reszta elementów jest za to postaci $f[x_i]$, zatem możemy wyznaczyć z ich pomocą wszystkie ilorazy zależne od dwóch węzłów – na przykład:

$d_{0,1} = f[x_0, x_1] = \frac{f[x_1] - f[x_0]}{x_1 - x_0} = \frac{d_1 - d_0}{x_1 - x_0}$

Zauważmy również, że po obliczeniu $f[x_{n-1}, x_n]$ nie użyjemy już do niczego ilorazu $f[x_n]$, zatem możemy go nadpisać tą nową wartością. Tymczasem, na przykład $f[x_1]$ będzie jeszcze potrzebny do wyznaczenia $f[x_1, x_2]$. Będziemy zatem szli od końca, nadpisując d_i po obliczeniu jego nowej wartości. Po jednej takiej rundzie uzyskamy wszystkie ilorazy oparte na dwóch węzłach. Wówczas $d_1 = f[x_0, x_1]$ przyjmie już swoją ostateczną postać. Kontynuujemy kolejne rundy, ponownie od końca, zatrzymując się na

wyliczeniu d_3. Po n takich rundach, nasz wektor wynikowy będzie już miał ostateczną postać. Poglądowy rysunek i pseudokod metody znajdują się poniżej.

Algorytm

Input:

$\bar{x}$ – wektor węzłów

$\bar{y}$ – wektor wartości

n – długość wektorów

Output:

$\bar{c}$ – wektor ilorazów różnicowych $f[x_0, \ldots, x_i]$

Code:

```
1 |  c̄ ← ȳ
2 |  for j from 1 to n do
3 |    for i from n down to j do
4 |      c_i ← (c_i − c_{i−1}) / (x_i − x_{i−j})
5 |  return c̄
```

Zadanie 2. - Uogólniony schemat Hornera

Opis zadania

Teraz naszym celem jest obliczenie wartości wielomianu Newtona w danym punkcie. Standardowa metoda, znana jako "według wzoru", pozwala na to z kwadratową złożonością obliczeniową. Jednakże istnieje sposób, aby rozłożyć nasz wielomian w taki sposób, który umożliwi obliczenia liniowe. Mamy następującą reprezentację wielomianu:

$$p(x) = \sum_{i=0}^{n} f[x_0, \ldots, x_i] \cdot q_i(x) = f[x_0] + \sum_{i=1}^{n} f[x_0, \ldots, x_i] \cdot (x - x_0) \cdot \ldots \cdot (x - x_i) =$$

$$= f[x_0] + (x - x_0)(f[x_0, x_1] + \sum_{i=2}^{n} f[x_0, \ldots, x_i] \cdot (x - x_1) \cdot \ldots \cdot (x - x_i)) =$$

$$= \ldots =$$

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

Możemy zauważyć, że obliczając wartość wielomianu, możemy postępować "od środka", zaczynając od najbardziej zagnieżdżonych elementów. To podejście jest zasadniczo schematem Hornera w bazie $\{q_i(x) : 0 \leq i \leq n\}$, zamiast tradycyjnej bazy $\{1, x, \ldots, x^n\}$. Formalnie, definiujemy:

$$w_n(x) = f[x_0, \ldots, x_n]$$

$$w_k(x) = f[x_0, \ldots, x_k] + (x - x_k) \cdot w_{k+1} \quad \text{dla} \quad k < n$$

$$p(x) = w_0(x)$$

Ten sposób pozwala nam efektywnie obliczyć wartość wielomianu używając tylko jednej pętli, co przekłada się na liniową złożoność czasową. Poniżej znajduje się pseudokod tej metody.

Algorytm

Input:

$\bar{x}$ – wektor węzłów $\bar{c}$ – wektor ilorazów różnicowych $f[x_0, \ldots, x_i]$ n – długość wektorów t – punkt, w którym należy obliczyć wartość wielomianu

Output:

v – wartość wielomianu w punkcie t

Code:

```
1 |  v ← c_n
2 |  for i from n − 1 down to 0 do
3 |      v ← c_i + (t − x_i) · v
4 |  return v
```

Zadanie 3. - Postać naturalna wielomianu

Opis zadania

Terminem "postać naturalna" wielomianu nazywamy jego reprezentację w bazie {1, x, x^2, x^3, ...}, czyli $p(x) = \sum_{i=0}^{n} a_i x^i$

W celu oszczędności znaków, przyjmujemy ponownie oznaczenie $c_i = f[x_0, \ldots, x_i]$. Kluczowym spostrzeżeniem niezbędnym do rozwiązania problemu jest fakt, że a_n – współczynnik przy x^n – jest równy c_n. Próbujemy teraz prześledzić kilka pierwszych iteracji algorytmu Hornera z poprzedniej sekcji, koncentrując się na współczynnikach przy konkretnych potęgach. Mamy:

$$w_{n-1} = c_{n-1} + (x - x_{n-1})w_n$$

Skąd otrzymujemy pierwsze składowe współczynnika przy x^{n-1}: c_{n-1} i $-x_{n-1}c_n$. Przechodzimy krok dalej:

$$w_{n-2} = c_{n-2} + (x - x_{n-2})w_{n-1}$$

Tutaj sytuacja nieco się zmienia, ponieważ w_{n-1}, w odróżnieniu od w_n, jest wielomianem stopnia większego niż 0. Przyjrzyjmy się dokładniej drugiemu składnikowi tej sumy,

rozbijając go na dwie części:

$$x \cdot w_{n-1} = x^2 c_n + x(c_{n-1} - x_{n-1}c_n)$$

$$-x_{n-2} \cdot w_{n-1} = -x_{n-2}c_n \cdot x + x_{n-2}(c_{n-1} - x_{n-1}c_n)$$

Stąd otrzymujemy:

$$w_{n-2} = c_{n-2} - x_{n-2}(c_{n-1} - x_{n-1}c_n) + x(c_{n-1} - (x_{n-1} + x_{n-2})c_n) + x^2 c_n$$

Kolejne iteracje wprowadzają mnóstwo znaków, ale pozwalają nam upewnić się co do dotychczasowych przypuszczeń.

$$x \cdot w_{n-2} = x^3 c_n + x^2(c_{n-1} - (x_{n-1} + x_{n-2})c_n) + x(c_{n-2} - x_{n-2}(c_{n-1} - x_{n-1}c_n))$$

$$-x_{n-3} \cdot w_{n-2} = -x_{n-3}c_n \cdot x^2 + x_{n-3}(c_{n-1} - (x_{n-1} + x_{n-2})c_n) \cdot x +$$

$$+ x_{n-3}(c_{n-2} - x_{n-2}(c_{n-1} - x_{n-1}c_n))$$

Wnioskiem z tego jest, że w każdej iteracji (cofając się, jak w algorytmie Hornera, oprócz pierwszej) wyznaczamy bazową wartość dla obecnej potęgi (dla x^i będzie to $c_i - x_{n-i}c_{n-i+1}$), a następnie musimy jeszcze zaktualizować współczynniki przy wyższych potęgach o "nowo odkryte" składniki. Z powyższych rozważań wynika, że do każdego a_j, gdzie $i < j < n$, dodajemy w i-tej iteracji składnik postaci $-x_{n-i}a_{j+1}$, gdzie a_{j+1} odpowiada obecnemu stanowi naszej wiedzy (co jest widoczne w przykładach – "nowe" części dla a_{n-1} i a_{n-2}). Projektujemy zatem algorytm oparty na dokładnie takiej technice. Jego złożoność wynosi kwadratowa, ponieważ dla każdego kroku "odwinięcia" (kroku algorytmu Hornera) musimy zaktualizować wszystkie współczynniki dla wyższych potęg.

Algorytm

Input:

$\bar{x}$ – wektor węzłów $\bar{c}$ – wektor ilorazów różnicowych $f[x_0, \ldots, x_i]$ n – długość wektorów

Output:

$\bar{a}$ – wektor współczynników wielomianu w postaci naturalnej

Code:

```
1 | a_n ← c_n
2 | for i from n − 1 down to 0 do
3 |     a_i ← c_i − x_i · a_{i+1}
4 |     for j from i + 1 to n − 1 do
5 |         a_j ← a_j − x_i · a_{j+1}
6 | return ā
```

Zadanie 4. - Wykres funkcji i wielomianu

Opis zadania

Celem tego zadania jest stworzenie jednej metody, która połączy wcześniej zaimplementowane techniki, umożliwiając graficzne porównanie otrzymanego wielomianu z interpolowaną funkcją. W tym celu użytkownik wskazuje przedział $[a, b]$, na którym zostanie utworzonych $n + 1$ równoodległych węzłów. Następnie obliczane są wartości funkcji dla tych węzłów. Ilorazy różnicowe są wyznaczane na podstawie tych wartości, co umożliwia określenie wartości wielomianu w dowolnym punkcie. Przedział jest dyskretyzowany tak, aby wartości wielomianu były widoczne również poza węzłami. Sugerowane jest użycie $N \cdot (n + 1)$ punktów na przedziale $[a, b]$, gdzie $N > 1$ jest liczbą całkowitą, aby uwzględnić również wartości węzłów. Dla każdego punktu obliczane są wartości funkcji i wielomianu, a następnie uzyskane wyniki są przedstawiane na wykresie.

Algorytm

Input:

f – interpolowana funkcja $[a, b]$ – przedział interpolacji n – stopień wielomianu

Code:

$h \leftarrow \frac{1}{n}(b - a)$
for k from 0 to n **do**
  $x_k \leftarrow a + k \cdot h$
  $y_k \leftarrow f(x_k)$
$\bar{c} \leftarrow \text{ilorazy różnicowe}(\bar{x}, \bar{y})$
$pt \leftarrow N \cdot (n + 1)$
$dx \leftarrow \frac{1}{pt-1}(b - a)$
for i from 0 to pt **do**
  $X_i \leftarrow a + i \cdot dx$
  $W_i \leftarrow \text{wartość wielomianu}(\bar{x}, \bar{c}, X_i)$
  $F_i \leftarrow f(X_i)$
$\text{wykres}(x = \bar{X}, y = [\bar{W}, \bar{F}])$

Uwagi do zadań 1. - 4.

Wszytskie algorytmy zostały zaimplementowane w języku programowania *julia*. Stworzony został moduł *Interpolation* w pliku *interpolation.jl*, zawiera on:

- **differencesQuotients()** - ilorazy różnicowe,

- **NewtonPolynomial()** - uogólniony schemat Hornera,
- **naturalForm()** - postać naturalna wielomianu,
- **plotNnfx()** - wizualizacja wykresu funkcji i wielomianu.

Wszystkie funkcje posiadają komentarze dokumentujący ogólny cel funkcji, dane wejściowe i dane wyjściowe. Szczegóły w pliku *interpolation.jl*.

Zadanie 5.

Opis zadania

Zadaniem było wykorzystanie narzędzia opracowanego w poprzednim etapie na funkcjach:

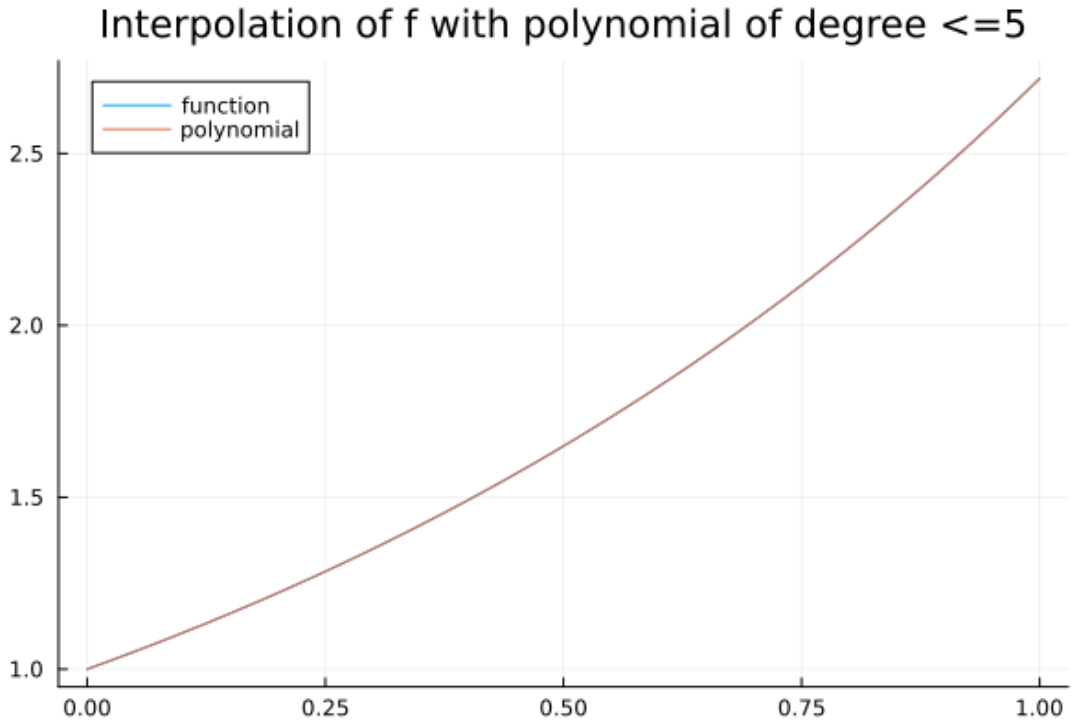
- $f(x) = e^x$ w przedziale $[0, 1]$
- $f(x) = x^2 \cdot \sin(x)$ w przedziale $[-1, 1]$

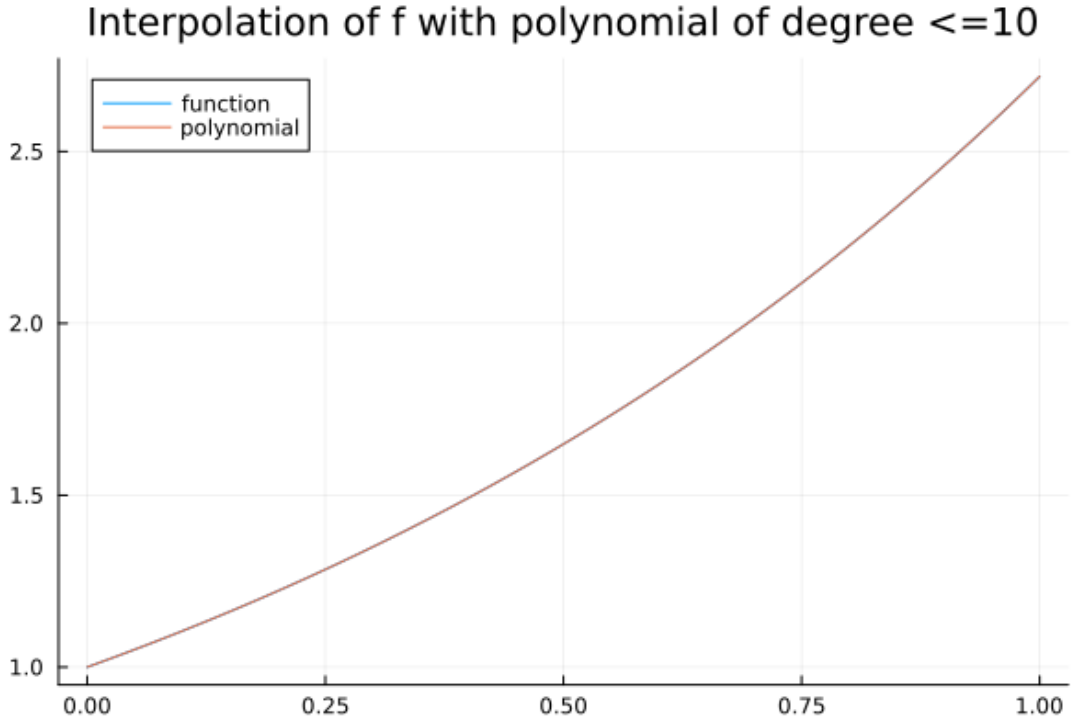
dla stopni wielomianu $n = \{5, 10, 15\}$.

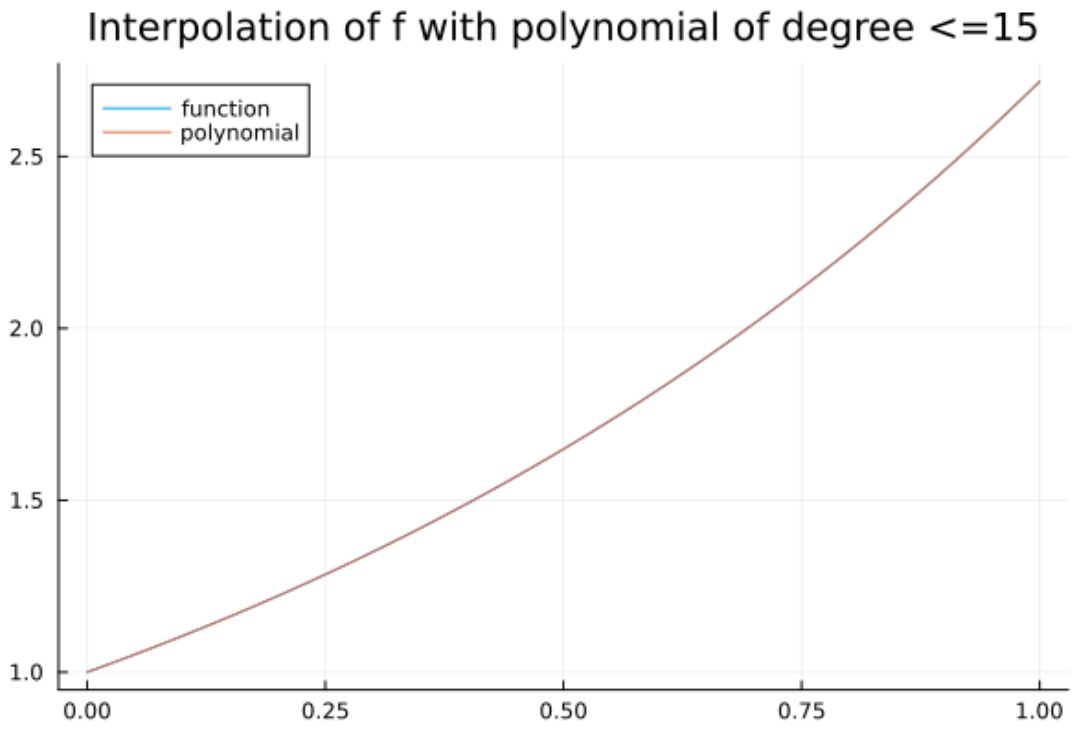
Wyniki

Wyniki zaostały przedstawione na poniższych wykresach zrobionych za pomocą funkcji *plotNnfx* z modułu *Interpolation* - zadania 1.-4.

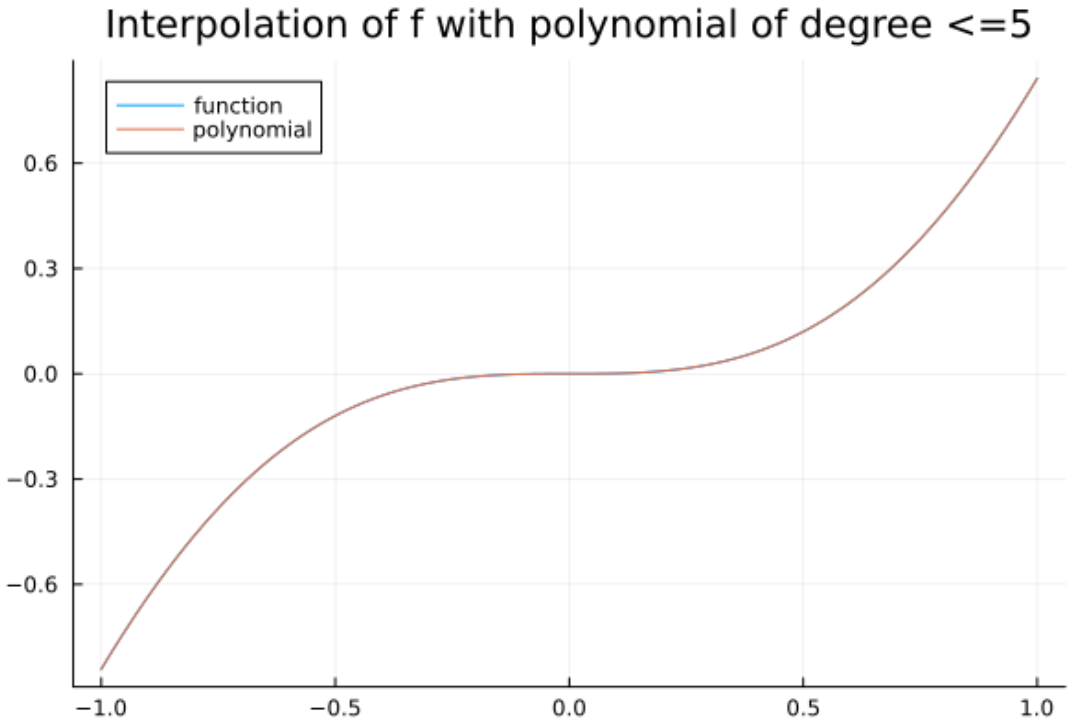
Wykresy dla $f(x) = e^x$ w przedziale $[0, 1]$, kolejno dla $n = \{5, 10, 15\}$:

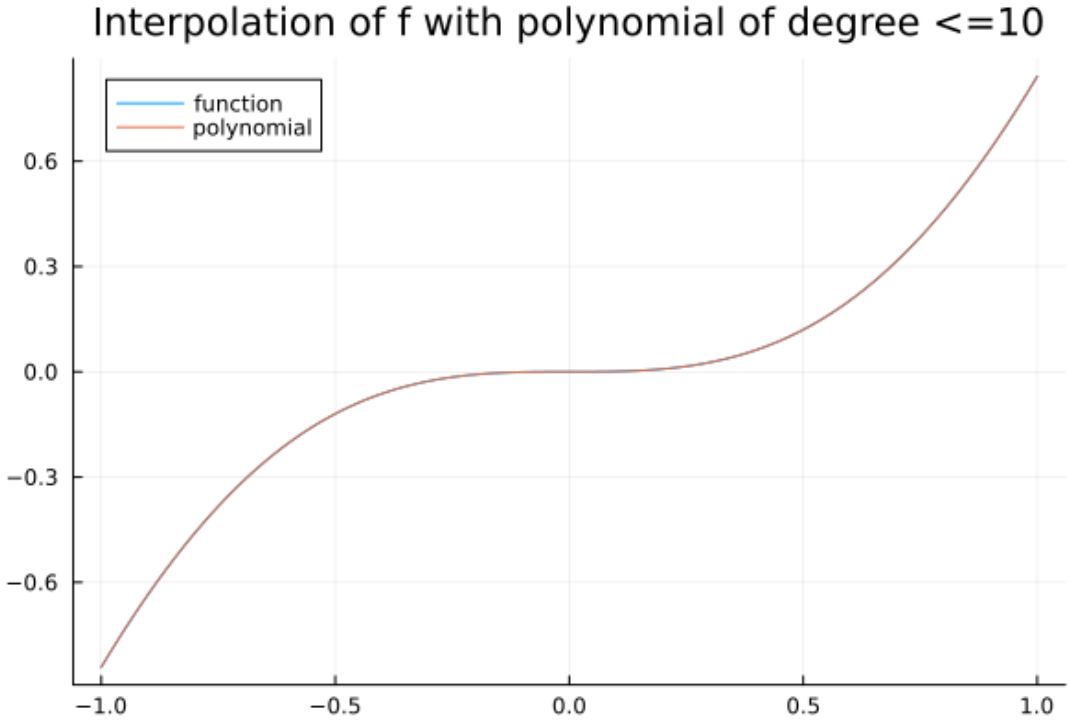

Interpolation of f with polynomial of degree <=5
function
polynomial
2.5
2.0
1.5
1.0
0.00
0.25
0.50
0.75
1.00

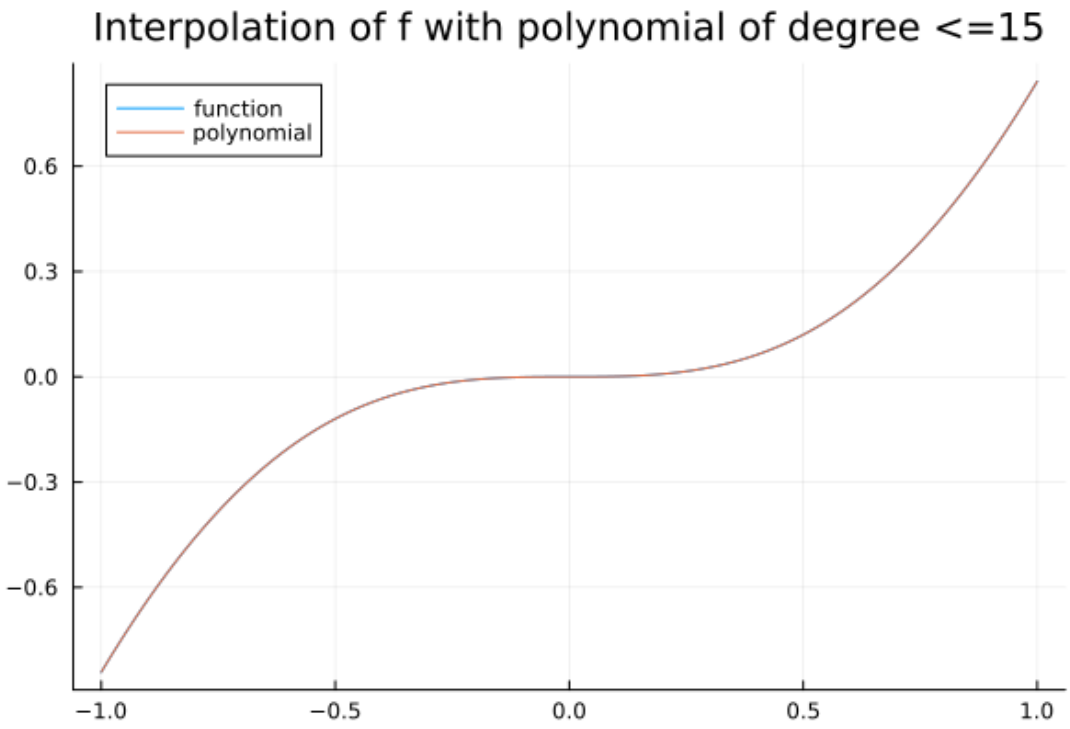

Interpolation of f with polynomial of degree <=10
function
polynomial
2.5
2.0
1.5
1.0
0.00
0.25
0.50
0.75
1.00



Wykresy dla $f(x) = x^2 \cdot \sin(x)$ w przedziale $[-1, 1]$, kolejno dla $n = \{5, 10, 15\}$:


Interpolation of f with polynomial of degree <=5
function
polynomial
0.6
0.3
0.0
−0.3
−0.6
−1.0
−0.5
0.0
0.5
1.0


Interpolation of f with polynomial of degree <=10
function
polynomial
0.6
0.3
0.0
−0.3
−0.6
−1.0
−0.5
0.0
0.5
1.0



Wnioski

W przypadku obu funkcji, nawet dla wielomianu interpolacyjnego stopnia 5, nie obserwujemy znaczących różnic na wykresie między tym wielomianem a interpolowaną funkcją. Wynika z tego, że te funkcje są łatwe do interpolacji przy użyciu wielomianów, a zaimplementowana metoda działa poprawnie.

Zadanie 6

Opis zadania

Zadaniem było wykorzystanie narzędzia opracowanego w poprzednim etapie na funkcjach:

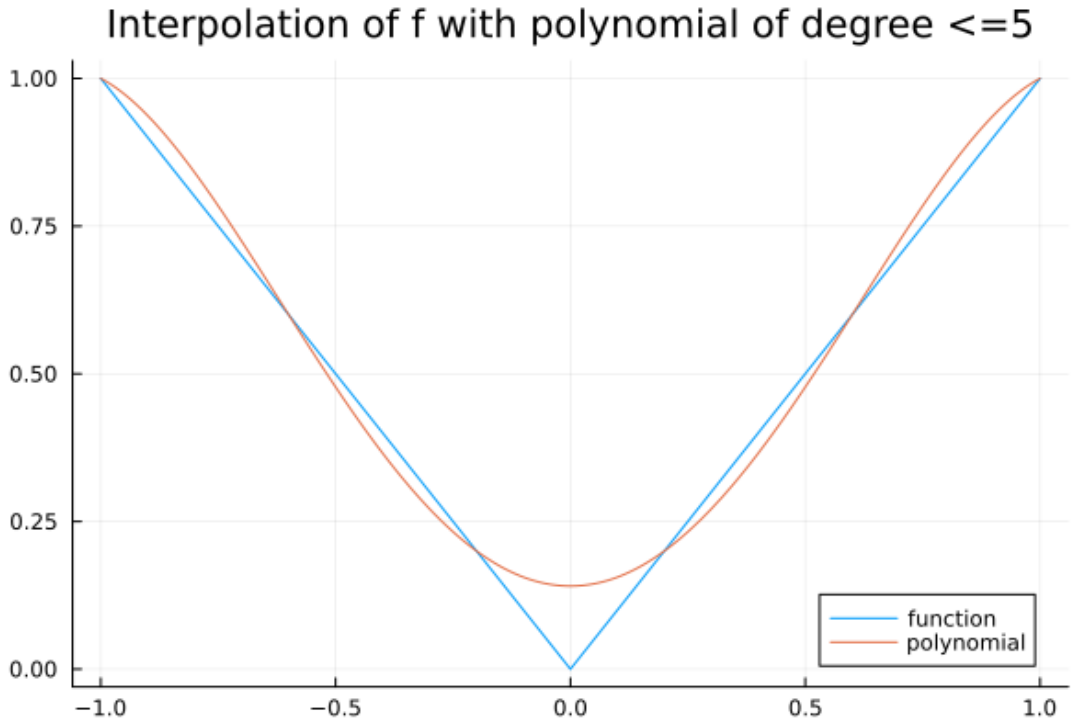
- $f(x) = |x|$ w przedziale $[-1, 1]$
- $f(x) = \frac{1}{1+x^2}$ w przedziale $[-5, 5]$

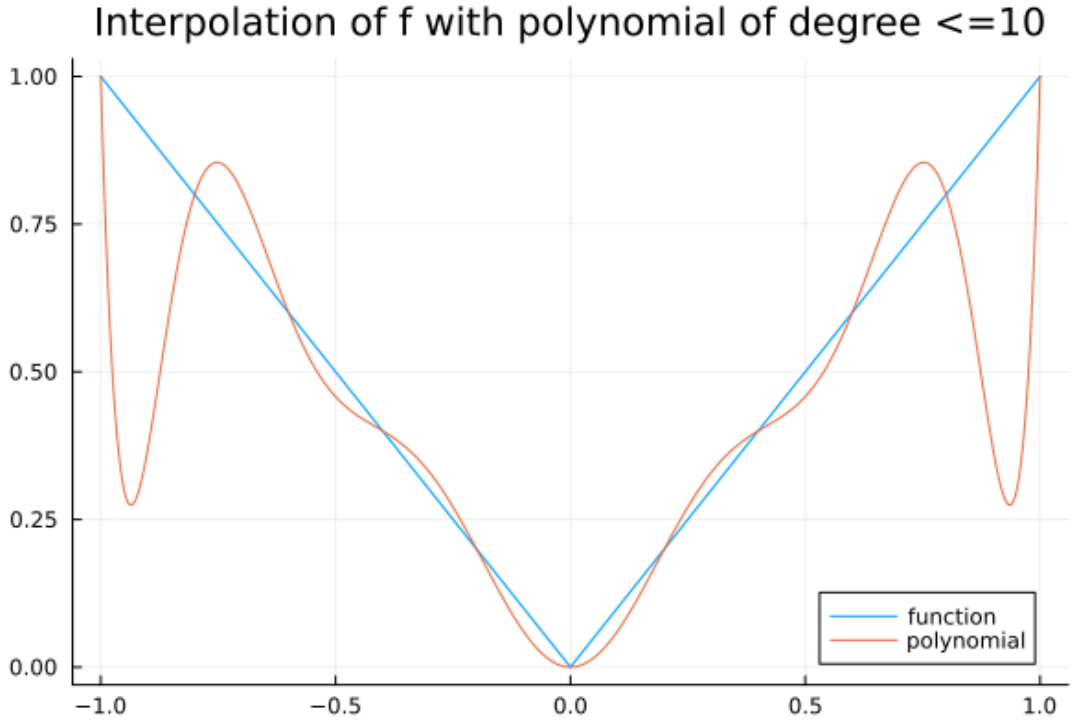
dla stopni wielomianu $n = \{5, 10, 15\}$.

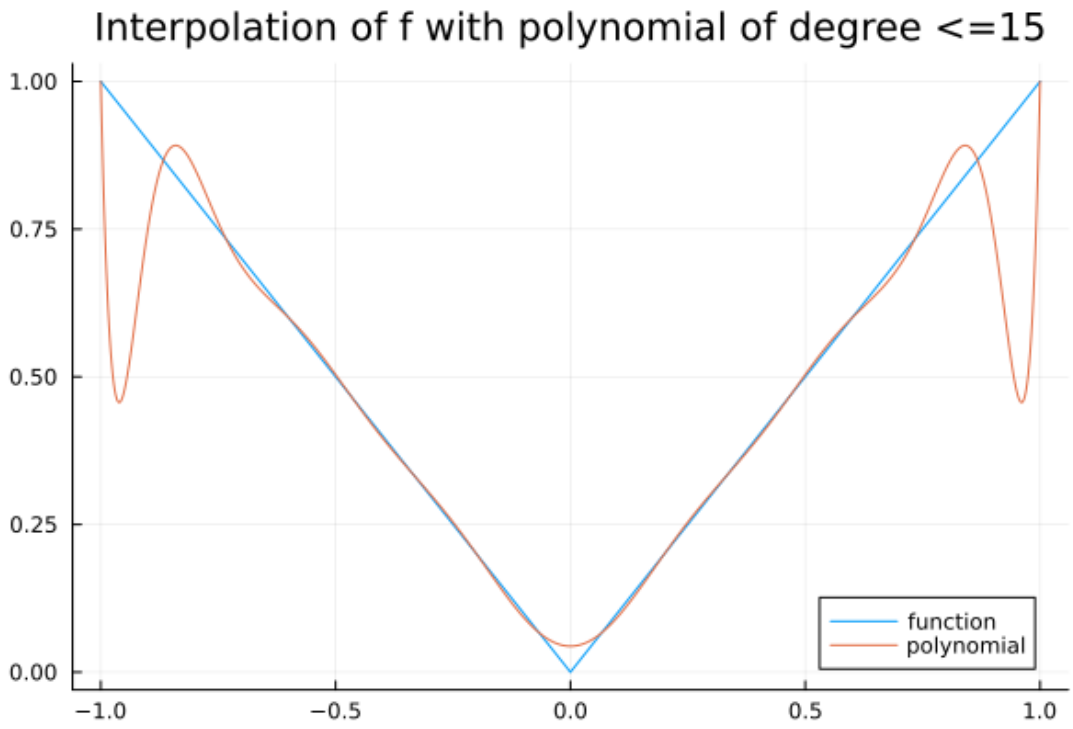
Wyniki

Wyniki zaostały przedstawione na poniższych wykresach zrobionych za pomocą funkcji *plotNnfx* z modułu *Interpolation* - zadania 1.-4.

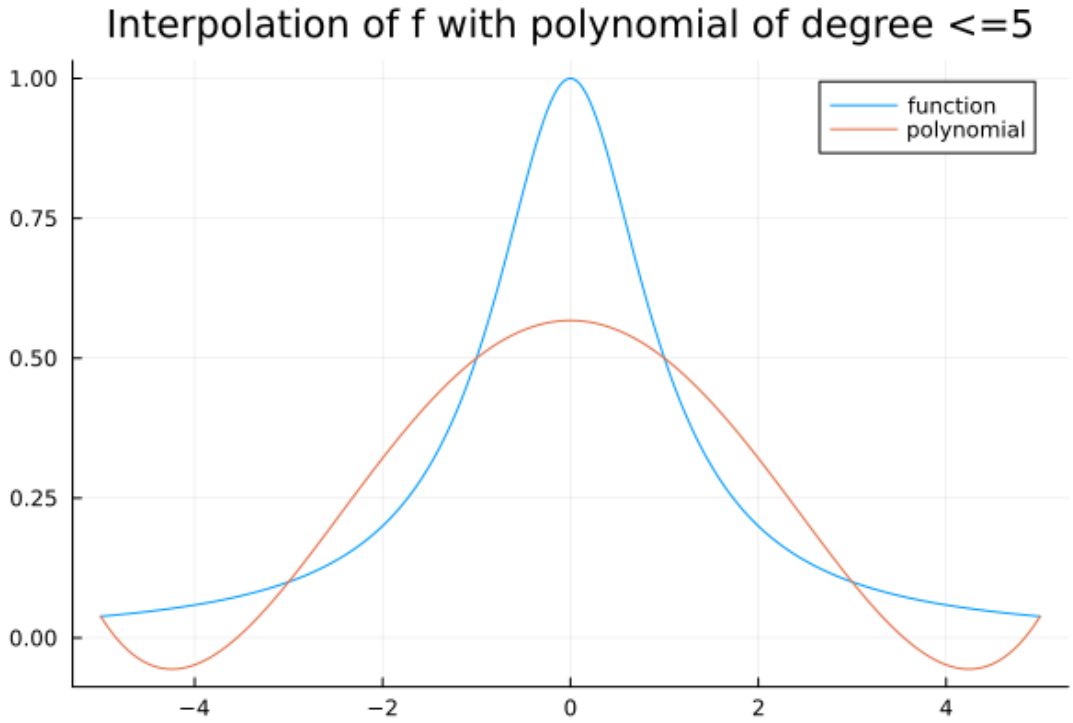
Wykresy dla $f(x) = |x|$ w przedziale $[-1, 1]$, kolejno dla $n = \{5, 10, 15\}$:

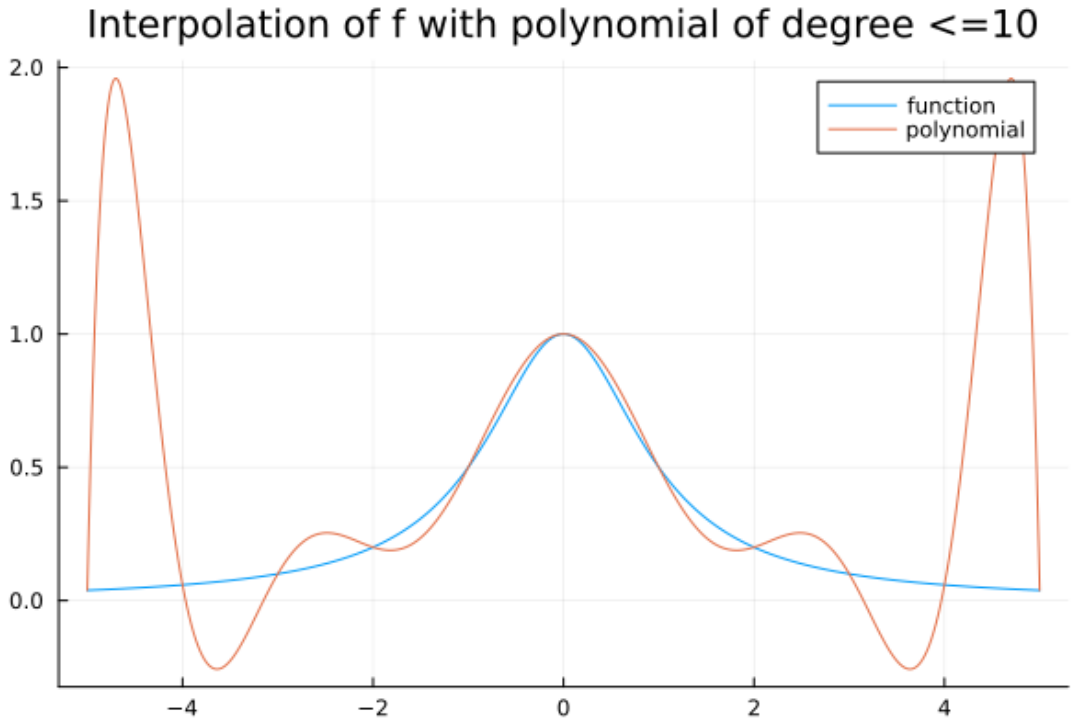

Interpolation of f with polynomial of degree <=5
1.00
0.75
0.50
0.25
0.00
−1.0
−0.5
0.0
0.5
1.0
function
polynomial

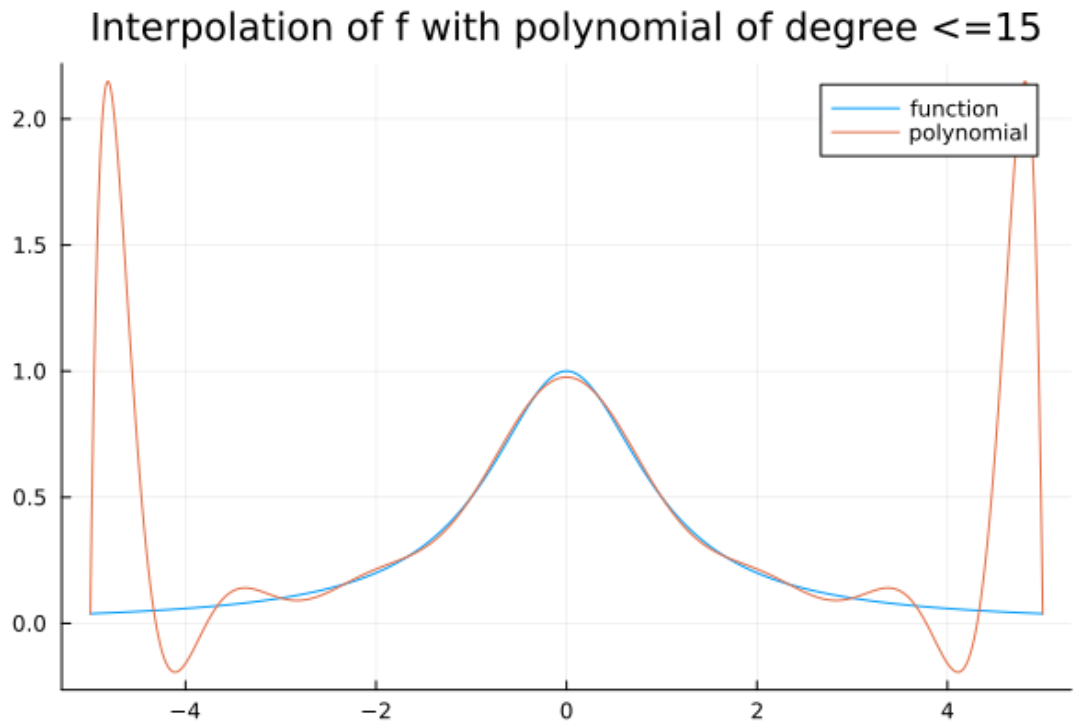

Interpolation of f with polynomial of degree <=10
1.00
0.75
0.50
0.25
0.00
−1.0
−0.5
0.0
0.5
1.0
function
polynomial



Wykresy dla $f(x) = \frac{1}{1+x^2}$ w przedziale $[-5, 5]$, kolejno dla $n = \{5, 10, 15\}$:


Interpolation of f with polynomial of degree <=5
function
polynomial
1.00
0.75
0.50
0.25
0.00
−4
−2
0
2
4


Interpolation of f with polynomial of degree <=10
function
polynomial
2.0
1.5
1.0
0.5
0.0
−4
−2
0
2
4



Wnioski

W przypadku tych funkcji obserwujemy istotne rozbieżności między wartościami funkcji a interpolującym je wielomianem. Dodatkowo, zwiększenie stopnia wielomianu nie przynosi poprawy błędu interpolacji na całym przedziale.

W przypadku funkcji $f_1(x) = |x|$ istotnym problemem jest brak ciągłej pochodnej na całym przedziale interpolacji. Na wykresie dostrzegamy "czubek" dla $x = 0$; przedstawienie go dokładnie za pomocą (gładkich z zasady) wielomianów jest bardzo trudne i prowadzi do silnych zniekształceń na krańcach przedziału.

Dla funkcji $f_2(x) = \frac{1}{1+x^2}$ sytuacja komplikuje się jeszcze bardziej. Na wykresach zauważamy bardzo duży wzrost błędu interpolacji na krańcach przedziału wraz ze zwiększaniem stopnia wielomianu interpolującego. Zjawisko to nazywane jest Efektem Rungego (od nazwiska Carla Rungego, niemieckiego matematyka).

W omówionym na wykładzie twierdzeniu o błędzie interpolacji zostało określone następujące równanie:

$$f(x) - p_n(x) = \frac{f^{(n+1)}(\zeta x)}{(n+1)!} \prod_{i=0}^{n} (x - x_i),$$

gdzie $\zeta \in (a, b)$, $f \in C^{n+1}[a, b]$, a p_n to wielomian interpolacyjny stopnia n na węzłach $x_0, x_1, \ldots, x_n \in [a, b]$.

Warto zauważyć, że dla f_2 wartość pochodnej w punkcie szybko rośnie rzędowo wraz ze wzrostem n. Wzrost ten jest tak szybki, że przewyższa znajdujące się w mianowniku $(n+1)!$ oraz zmniejszające się odległości $\prod_{i=0}^{n}(x - x_i)$ dla węzłów równoodległych. Teoretycznie można pokazać, że górne ograniczenie błędu interpolacji wynosi:

$$\lim_{n\to\infty}\left(\max_{a<x<b}|f(x)-p_n(x)|\right)=+\infty,$$

co praktycznie obserwujemy na wykresach. Najprostszym rozwiązaniem problemu rosnącego błędu interpolacji dla f_2 byłaby zmiana węzłów interpolacji z równoodległych na np. te wyznaczane z pomocą wielomianów Czebyszewa.