Obliczenia naukowe

Laboratorium

Lista 4

Kinga Majcher
272354

Grudzień 2024

Problem interpolacji

Problem interpolacji sformułowany jest następująco. Istnieje $n+1$ par $(x_i, y_i = f(x_i))$, gdzie $\forall_{i,j} x_i \neq x_j$ i $i \in \{0, \ldots, n\}$. Chcemy wyznaczyć taki wielomian $p_n(x)$ stopnia co najwyżej n, dla którego $\forall_{i \in \{0,\ldots,n\}} \; p_n(x_i) = f(x_i) = y_i$. Wiemy, że istnieje dokładnie jeden taki wielomian.

Chcąc otrzymać wielomian interpolacyjny w postaci:

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

należało by wyznaczyć współczynniki $a_0, a_1, \ldots, a_n$, co prowadziło by do konieczności rozwiązania układu równań z macierzą Vandermonde'a, co jest zadaniem bardzo źle uwarunkowanym w kwestii arytmetyki zmiennoprzecinkowej.

Przedstawmy więc p_n w innej bazie. Niech:

$$q_0(x) = 1$$

$$q_1(x) = (x - x_0)$$

$$q_2(x) = (x - x_0)(x - x_1)$$

$$\vdots$$

$$q_n(x) = (x - x_0)(x - x_1) \cdots (x - x_{n-1})$$

Wówczas istnieje takie c_j, że:

$$p_n(x) = \sum_{j=0}^{n} c_j q_j(x)$$

Korzystamy z faktu, że wielomian spełnia warunki interpolacji i uzyskujemy układ równań, z którego wyznaczamy $c_0, c_1, \ldots, c_n$:

$$\forall_{i \in \{0,\ldots,n\}} \quad \sum_{j=0}^{n} c_j q_j(x_i) = f(x_i)$$

Zauważmy, że dla dowolnego k wartość c_k zależy od $f(x_0), f(x_1), \ldots, f(x_k)$. Wprowadźmy więc notację $c_k = f[x_0, x_1, \ldots, x_k]$. Wielkość tę nazywamy ilorazem różnicowym. Po podstawieniu do wzoru otrzymujemy postać Newtona wzoru interpolacyjnego:

$$p_n(x) = \sum_{k=0}^{n} c_k q_k(x) = \sum_{k=0}^{n} f[x_0, x_1, \ldots, x_k] \prod_{j=0}^{k-1} (x - x_j)$$

1 Zadanie 1

1.1 Opis problemu

Napisać funkcję obliczającą ilorazy różnicowe bez użycia tablicy dwuwymiarowej.

Funkcja przyjmuje parametry:

`x` - wektor długości $n+1$ zawierający węzły $x_0, \ldots, x_n$

$\texttt{x[1]} = x_0$

$\vdots$

$\texttt{x[n+1]} = x_n$

`f` - wektor długości $n+1$ zawierający wartości interpolowanej funkcji w węzłach $f(x_0), \ldots, f(x_n)$

Funkcja zwraca:

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

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

$\texttt{fx[2]} = f[x_0, x_1]$

$\vdots$

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

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

Zaprogramować funkcję bez użycia tablicy dwuwymiarowej.

1.2 Opis metody

Zadaniem algorytmu jest obliczenie ilorazów różnicowych mając dane pary $(x_i, y_i = f(x_i))$.

Wiemy, że:

$$f[x_i] = f(x_i)$$

a pozostałe ilorazy różnicowe spełniają równość:

$$f[x_i, x_{i+1}, \ldots, x_k] = \frac{f[x_{i+1}, x_{i+2}, \ldots, x_k] - f[x_i, x_{i+1}, \ldots, x_{k-1}]}{x_k - x_i}$$

Naturalnym było by więc zaczęcie tworzenia macierzy trójkątnej, jak poniżej i wykorzystywanie jej już wyliczonych elementów do dalszych obliczeń:

$$\begin{bmatrix}
f[x_0] & f[x_0,x_1] & \ldots & f[x_0,\ldots,x_{m-1}] & f[x_0,\ldots,x_m] & \ldots & f[x_0,\ldots,x_{n-1}] & f[x_0,\ldots,x_n] \\
f[x_1] & f[x_1,x_2] & \ldots & f[x_1,\ldots,x_m] & f[x_1,\ldots,x_{m+1}] & \ldots & f[x_1,\ldots,x_n] & \\
\vdots & \vdots & & \vdots & \vdots & \iddots & & \\
f[x_k] & f[x_k,x_{k+1}] & \ldots & f[x_k,\ldots,x_{k+m-1}] & f[x_k,\ldots,x_{k+m}] & & & \\
f[x_{k+1}] & f[x_{k+1},x_{k+2}] & \ldots & f[x_{k+1},\ldots,x_{k+m}] & & & & \\
\vdots & & \iddots & & & & & \\
f[x_{n-1}] & f[x_{n-1},x_n] & & & & & & \\
f[x_n] & & & & & & &
\end{bmatrix}$$

Chcemy jednak uniknąć używania tablicy dwuwymiarowej w implementacji. Zauważmy, że po wyliczeniu wszystkich ilorazów różnicowych zależnych od k węzłów, to wszystkie poza $f[x_0, x_1, \ldots, x_k]$ stają się dla nas zbędne. Dzięki tej obserwacji algorytm można wykonać na jednej tablicy $n+1$ - elementowej.

Na początku do tablicy zapisujemy wartości funkcji w węzłach interpolacji (ponieważ $f[x_i] = f(x_i)$). Zauważmy, że element na pierwszym miejscu tablicy już jest jednym z elementów, które chcemy zwrócić. Za pomocą pozostałych elementów możemy za to wyznaczyć wszystkie ilorazy zależne od dwóch węzłów. Wyznaczamy je i obliczonymi wartościami nadpisujemy pozostałe elementy tablicy. Analogicznie robimy dla ilorazów różnicowych wyższych rzędów. Stan tablicy wyjściowej po każdej iteracji algorytmu przedstawia poniższa grafika:

$$\begin{array}{cccccc}
\mathbf{f[x_0]} & \mathbf{f[x_0]} & \mathbf{f[x_0]} & \dots & \mathbf{f[x_0]} & \mathbf{f[x_0]} \\
f[x_1] & \mathbf{f[x_0, x_1]} & \mathbf{f[x_0, x_1]} & \dots & \mathbf{f[x_0, x_1]} & \mathbf{f[x_0, x_1]} \\
f[x_2] & f[x_1, x_2] & \mathbf{f[x_0, x_1, x_2]} & \dots & \mathbf{f[x_0, x_1, x_2]} & \mathbf{f[x_0, x_1, x_2]} \\
\vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\
f[x_{n-1}] & f[x_{n-2}, x_{n-1}] & f[x_{n-3}, x_{n-2}, x_{n-1}] & \dots & \mathbf{f[x_0, x_1, \dots, x_{n-1}]} & \mathbf{f[x_0, x_1, \dots, x_{n-1}]} \\
f[x_n] & f[x_{n-1}, x_n] & f[x_{n-2}, x_{n-1}, x_n] & \dots & & \mathbf{f[x_0, x_1, \dots, x_n]}
\end{array}$$

Tabela 1: Stany tablicy wyjściowej po kolejnych iteracjach. Kolumny pokazują stany po kolejnych iteracjach, ilorazy zapisane pogrubioną czcionką są ilorazami już wyjściowymi.

Przedstawiona metoda ma złożoność obliczeniową $O(n^2)$ i złożoność pamięciową $O(n)$.

1.3 Pseudokod

Algorithm 1: Funkcja obliczająca ilorazy różnicowe

Input: $\overline{x}$ – wektor punktów, $\overline{f}$ – wektor wartości funkcji w punktach z $\overline{x}$
Output: $\overline{c}$ – wektor ilorazów różnicowych

```
1 c̄ ← f̄;
2 for j from 1 to n do
3 |   for k from n downto j do
4 |   |   c_k ← (c_k − c_{k−1}) / (x_k − x_{k−j});
5 |   end
6 end
7 return c̄
```

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 $O(n)$.

Funkcja przyjmuje parametry:

`x` - wektor długości $n + 1$ zawierający węzły $x_0, \dots, x_n$

$\texttt{x[1]} = x_0$

$\vdots$

$\texttt{x[n+1]} = x_n$

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

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

$\texttt{fx[2]} = f[x_0, x_1]$

$\vdots$

$\texttt{fx[n]} = f[x_0, \dots, x_{n-1}]$

`fx[n+1]` $= f[x_0, \ldots, x_n]$

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

Funkcja zwraca:

`nt` - wartość wielomianu w punkcie t

2.2 Opis metody

Zadaniem algorytmu jest obliczenie wartości wielomianu interpolacyjnego stopnia n w postaci Newtona w punkcie t w czasie $O(n)$. Jesteśmy w stanie policzyć to prosto ze wzoru, jednakże wówczas otrzymujemy złożoność $O(n^2)$, a tego nie chcemy.

Przekształćmy wielomian:

$$p_n(x) = \sum_{k=0}^{n} c_k \cdot q_k(x) = f[x_0] + \sum_{k=1}^{n} f[x_0, \ldots, x_k](x - x_0) \ldots (x - x_k) =$$

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

$$\vdots$$

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

Zapiszmy więc rekurencyjnie:

$$w_n(x) = f[x0, \ldots, x_n]$$

$$w_k(x) = f[x_0, \ldots, x_k] + (x - x_k) w_{k+1}, \text{ gdzie } k \in \{0, \ldots, n-1\}$$

Wówczas:

$$p_n(x) = w_0(x)$$

Dzięki takim przekształceniom algorytm ma złożoność obliczeniową $O(n)$.

2.3 Pseudokod

Algorithm 2: Funkcja obliczająca wartość wielomianu interpolacyjnego w danym punkcie

Input: $\overline{x}$ – wektor punktów, $\overline{c}$ – wektor ilorazów różnicowych, t – punkt, dla którego szukamy wartości wielomianu

Output: v – wartość wielomianu w punkcie t

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

3 Zadanie 3

3.1 Opis problemu

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

Funkcja przyjmuje parametry:

`x` - wektor długości $n + 1$ zawierający węzły $x_0, \ldots, x_n$

`x[1]` $= x_0$

$\vdots$

`x[n+1]` $= x_n$

`f` - wektor długości $n+1$ zawierający ilorazy różnicowe

`fx[1]` $= f[x_0]$

`fx[2]` $= f[x_0, x_1]$

$\vdots$

`fx[n]` $= f[x_0, \ldots, x_{n-1}]$

`fx[n+1]` $= f[x_0, \ldots, x_n]$

Funkcja zwraca:

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

`a[1]` $= a_0$

`a[2]` $= a_1$

$\vdots$

`a[n]` $= a_{n-1}$

`a[n+1]` $= a_n$

3.2 Opis metody

Zadaniem algorytmu jest sprowadzenie wielomianu w postaci Newtona:

$$\sum_{k=0}^{n} c_k \prod_{j=0}^{k-1}(x - x_j)$$

do wielomianu w postaci naturalnej

$$\sum_{i=0}^{n} a_i x^i$$

w czasie $O(n^2)$.

Wykorzystajmy rekurencję z poprzedniego zadania:

$w_n = c_n$, skąd wiemy, że przy x^n współczynnik będzie wynosił c_n

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

skąd wiemy, że przy x^{n-1} współczynnik będzie wynosił $c_{n-1} - x_{n-1}c_n$

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

Z powyższych obliczeń można zauważyć, że kolejne współczynniki są zależne od tych poprzednio obliczonych. Wykorzystamy ten fakt w algorytmie.

Zaczynając od $w_n(x)$ możemy wyliczać kolejne wartości częściowe tak jak w algorytmie Hornera, z różnicą taką, że chcemy zapisywać je w postaci naturalnej. Wówczas obliczone już wartości można użyć do obliczania wartości kolejnych współczynników. Taki algorytm wykorzystuje dwie pętle, przez co jego złożoność obliczeniowa wynosi $O(n^2)$, czyli taka jaką chcieliśmy otrzymać.

3.3 Pseudokod

Algorithm 3: Funkcja obliczająca współczynniki postaci naturalnej wielomianu interpolacyjnego

Input: $\overline{x}$ – wektor punktów, $\overline{c}$ – wektor ilorazów różnicowych
Output: $\overline{a}$ – wektor współczynników postaci naturalnej wielomianu interpolacyjnego

```
1 a_n ← c_n;
2 for i from n − 1 downto 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 |   end
7 end
8 return ā
```

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 narysuje wielomian interpolacyjny i interpolowaną funkcję. Do rysowania zainstaluj np. pakiet Plots, PyPlot lub Gadfly. W interpolacji użyć węzłów równoodległych, tj. $x_k = a + kh, h = (b - a)/n, k = 0, 1, \ldots, n$.

Funkcja przyjmuje parametry:

`f` - funkcja $f(x)$ zadana jako anonimowa funkcja

`a, b` - przedział interpolacji

`n` - stopień wielomianu interpolacyjnego

Funkcja zwraca:

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

4.2 Opis metody

Celem zadania jest połączenie zaimplementowanych wcześniej metod w jedną, umożliwiającą graficzne porównanie otrzymanego wielomianu interpolacyjnego z dokładną funkcją.

Z racji tego, że chcemy interpolować za pomocą wielomianu stopnia n, to musimy w przedziale $[a, b]$ wyznaczyć $n + 1$ równoodległych węzłów, takich, że: $a = x_0 < x_2 < \cdots < x_{n-1} < x_n = b$, gdzie $x_i = a + i \cdot h, h = \frac{b-a}{n}$. Dla obliczonych x_i obliczamy wartości $f(x_i)$. Za pomocą metody `ilorazyRoznicowe (x::VectorFloat64, f::VectorFloat64)` obliczmy ilorazy różnicowe c_i dla tych punktów. Mając te dane możemy skorzystać z funkcji `warNewton (x::VectorFloat64, fx::VectorFloat64, t::Float64)` i obliczać wartości wielomianu interpolacyjnego dla dowolnego punktu.

Aby móc narysować sensowny wykres, potrzebujemy znacznie więcej punktów niż te wyznaczone węzły interpolacji. Dlatego w przedziale $[a, b]$ wyznaczmy $P \cdot n + 1, P \in \mathrm{N}$ punktów. Wówczas w każdym przedziale $[x_i, x_{i+1})$ znajduje się dokładnie P punktów, a ponadto punktem jest także wartość prawego końca przedziału.

Dla tak obliczonych punktów obliczamy wartości funkcji interpolowanej i wielomianu interpolującego i umieszczamy te dane na wykresie.

4.3 Pseudokod

Algorithm 4: Funkcja przedstawiająca graficznie wykresy funkcji interpolowanej i wielomianu interpolującego

Input: f – funkcja, którą chcemy interpolować, a, b – końce przedziału interpolacji, n – stopień wielomianu interpolacyjnego
Output: wykres wielomianu interpolującego i funkcji interpolowanej

$P \leftarrow 100$;
$h \leftarrow \frac{b-a}{n}$;
for i ***from*** 0 ***to*** n **do**
  $x_i \leftarrow a + i \cdot h$;
  $y_i \leftarrow f(x_i)$;
end
$\overline{c} \leftarrow$ `ilorazyRoznicowe`$(\overline{x}, \overline{y})$;
$z \leftarrow P \cdot n + 1$ // liczba punktów ;
$v \leftarrow \frac{b-a}{z-1}$ // odstęp między punktami ;
for i ***from*** 0 ***to*** z **do**
  $X_i \leftarrow a + i \cdot v$ // wartość x na wykresie;
  $W_i \leftarrow$ `warNewton`$(\overline{x}, \overline{c}, X_i)$ // wartość wielomianu w punkcie x;
  $Y_i \leftarrow f(X_i)$ // wartość funkcji f w punkcie x;
end
$wykres = (x = \overline{X}, y = \overline{W}, \overline{Y})$
return $wykres$

5 Zadanie 5

5.1 Opis problemu

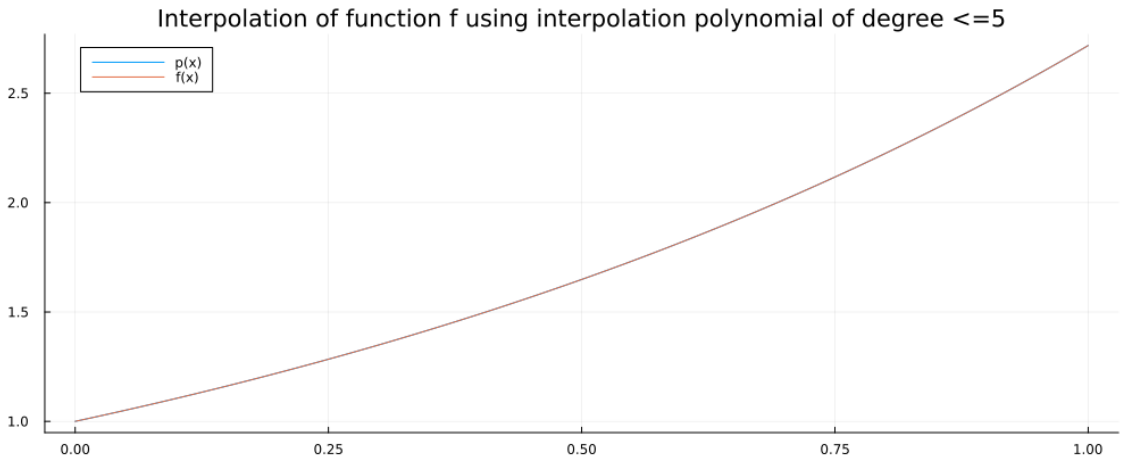
Przetestować funkcję `rysujNnfx(f, a, b, n)` na nastepujących przykładach:

1. $e^x, [0, 1], n \in \{5, 10, 15\}$
2. $x^2 \sin x, [-1, 1], n \in \{5, 10, 15\}$

5.2 Rozwiązanie

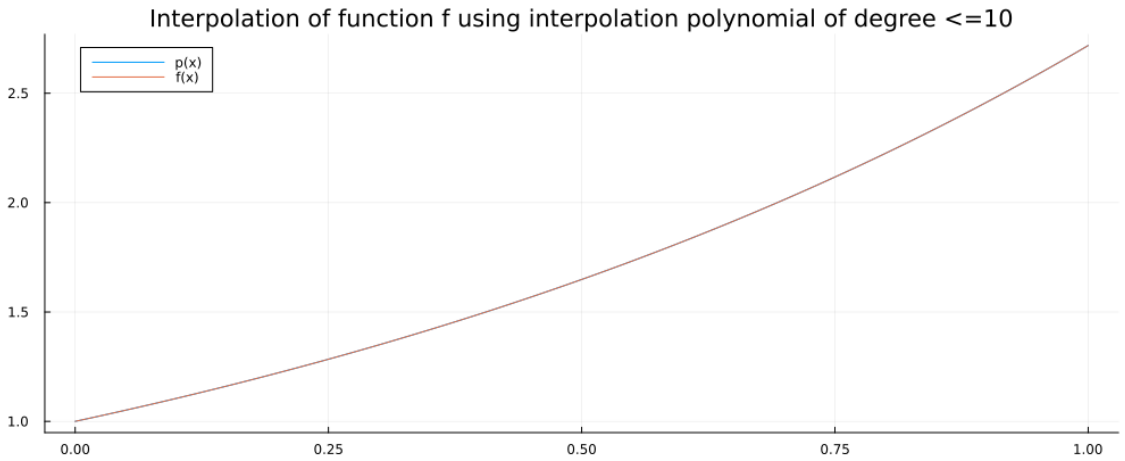
Do rozwiązania zadania została wykorzystana metoda `rysujNnfx(f, a, b, n)` uruchamiana dla odpowiednich danych. Wykresy generowane przez metodę zostały zapisane do plików i wykorzystane do interpretacji.

5.3 Wyniki

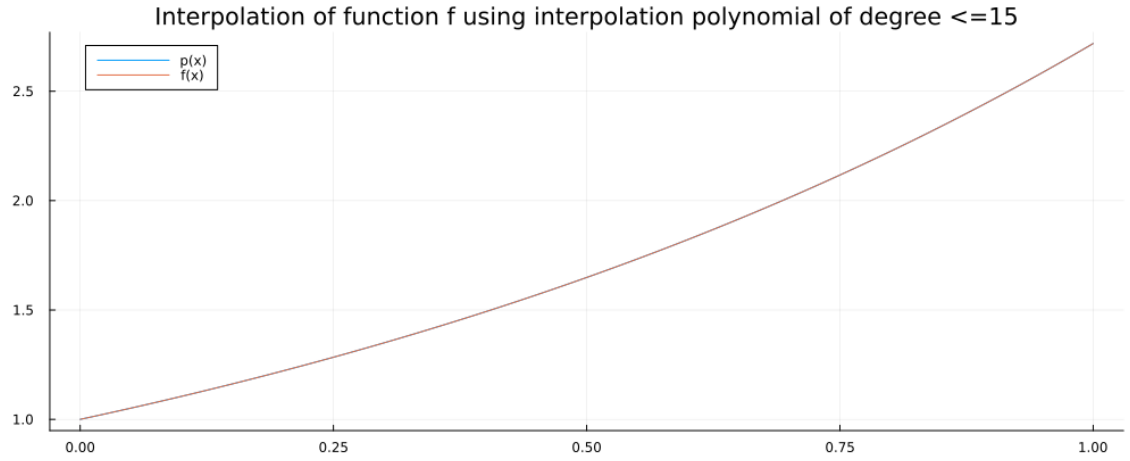


Rysunek 1: Wykres wielomianu interpolacyjnego stopnia ≤ 5 i funkcji interpolowanej $f(x) = e^x$ na przedziale $[0, 1]$

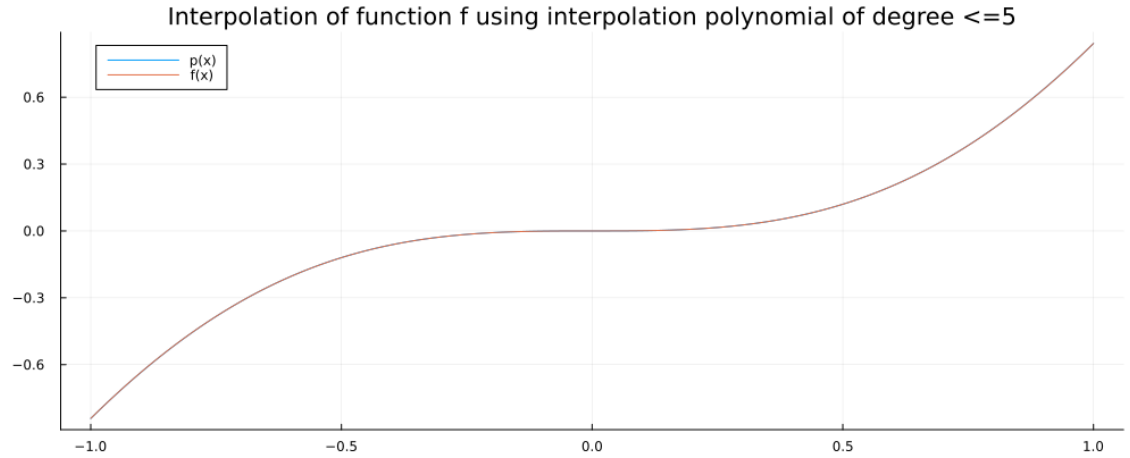
.



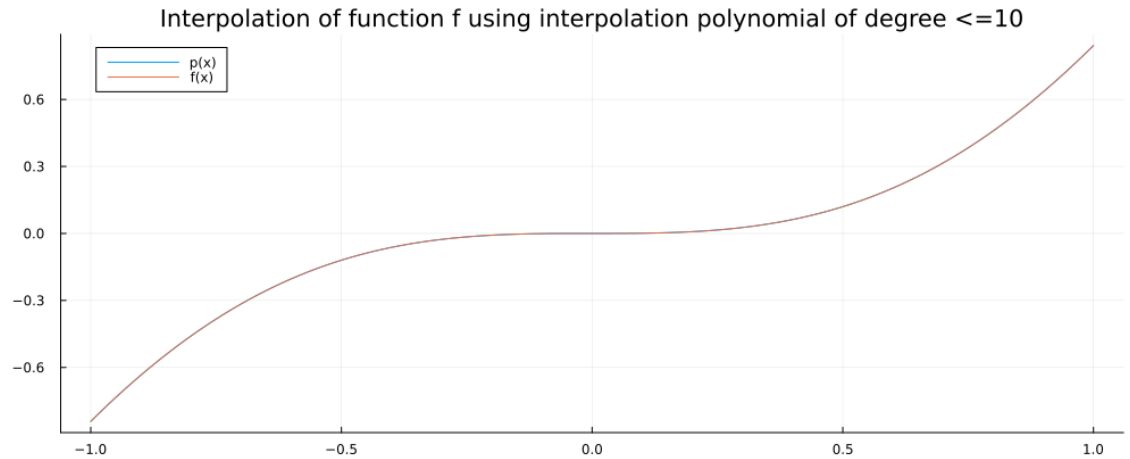
Rysunek 2: Wykres wielomianu interpolacyjnego stopnia ≤ 10 i funkcji interpolowanej $f(x) = e^x$ na przedziale $[0, 1]$.



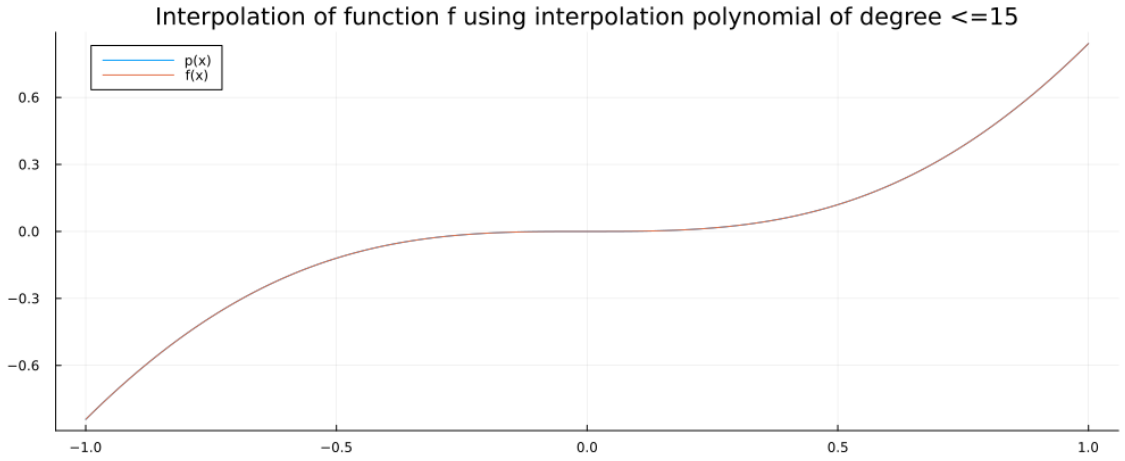
Rysunek 3: Wykres wielomianu interpolacyjnego stopnia ≤ 15 i funkcji interpolowanej $f(x) = e^x$ na przedziale $[0, 1]$.



Rysunek 4: Wykres wielomianu interpolacyjnego stopnia ≤ 5 i funkcji interpolowanej $f(x) = x^2 \sin x$ na przedziale $[-1, 1]$.



Rysunek 5: Wykres wielomianu interpolacyjnego stopnia ≤ 10 i funkcji interpolowanej $f(x) = x^2 \sin x$ na przedziale $[-1, 1]$.



Rysunek 6: Wykres wielomianu interpolacyjnego stopnia ≤ 15 i funkcji interpolowanej $f(x) = x^2 \sin x$ na przedziale $[-1, 1]$.

5.4 Interpretacja wyników oraz wnioski

Jak można wywnioskować po spojrzeniu na wykresy, dla funkcji $f(x) = e^x f(x) = x^2 \sin x$ interpolacja wielomianem sprawdza się bardzo dobrze. Nawet dla wielomianów niskiego stopnia nie widać znaczących różnic między wielomianem, a interpolowaną funkcją.

Funkcje te dają się łatwo interpolować, a zaimplementowana metoda działa poprawnie.

6 Zadanie 6

6.1 Opis problemu

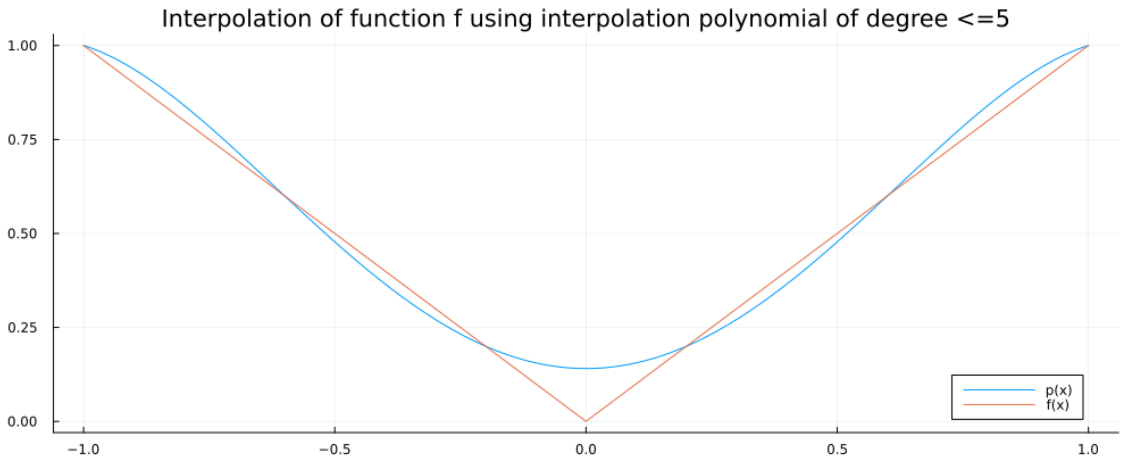
Przetestować funkcję `rysujNnfx(f, a, b, n)` na nastepujących przykładach:

1. $|x|, [-1, 1], n \in \{5, 10, 15\}$
2. $\frac{1}{1+x^2}, [-5, 5], n \in \{5, 10, 15\}$

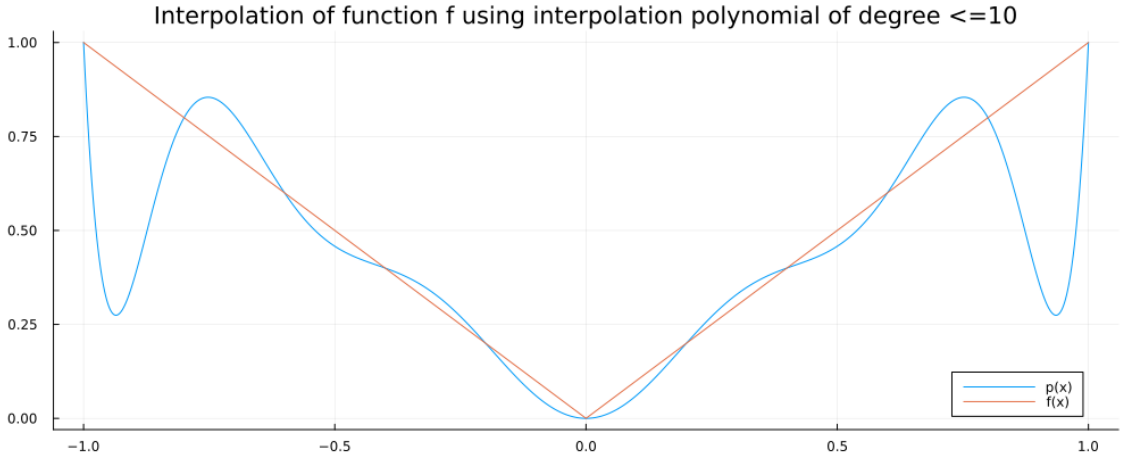
6.2 Rozwiązanie

Do rozwiązania zadania została wykorzystana metoda `rysujNnfx(f, a, b, n)` uruchamiana dla odpowiednich danych. Wykresy generowane przez metodę zostały zapisane do plików i wykorzystane do interpretacji.

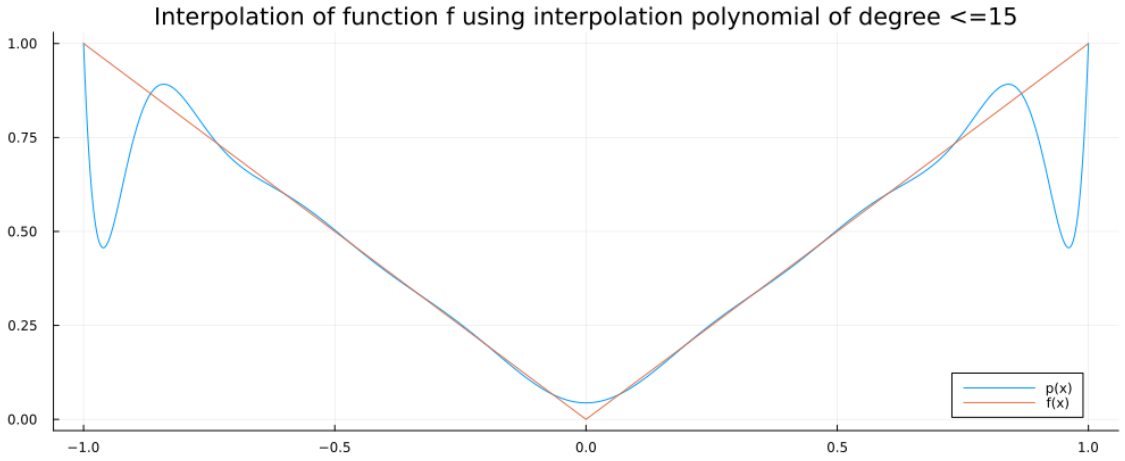
6.3 Wyniki



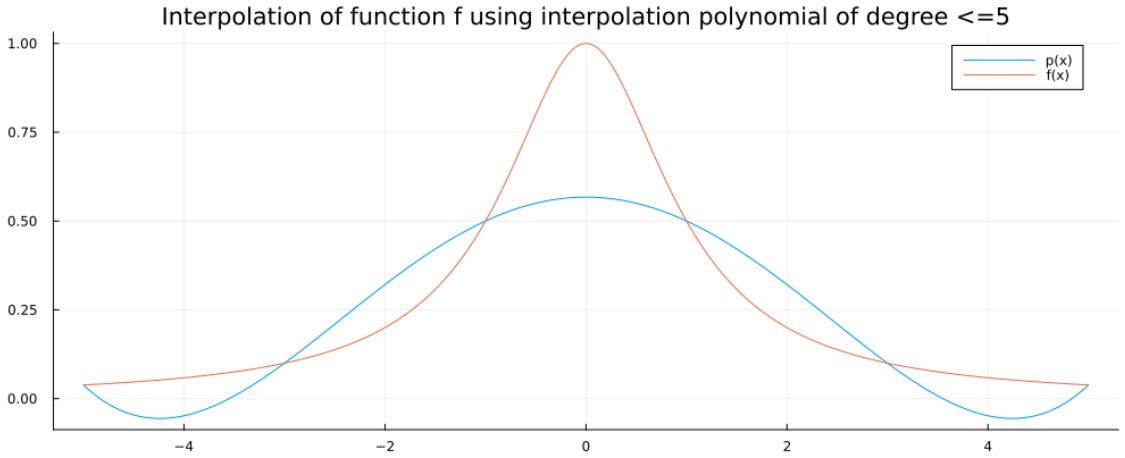
Rysunek 7: Wykres wielomianu interpolacyjnego stopnia ≤ 5 i funkcji interpolowanej $f(x) = |x|$ na przedziale $[-1, 1]$.



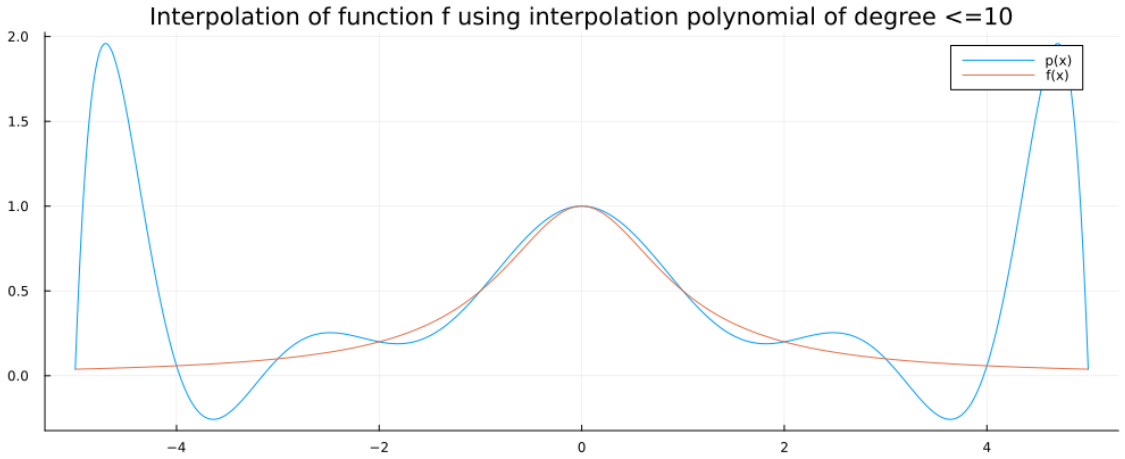
Rysunek 8: Wykres wielomianu interpolacyjnego stopnia ≤ 10 i funkcji interpolowanej $f(x) = |x|$ na przedziale $[-1, 1]$.



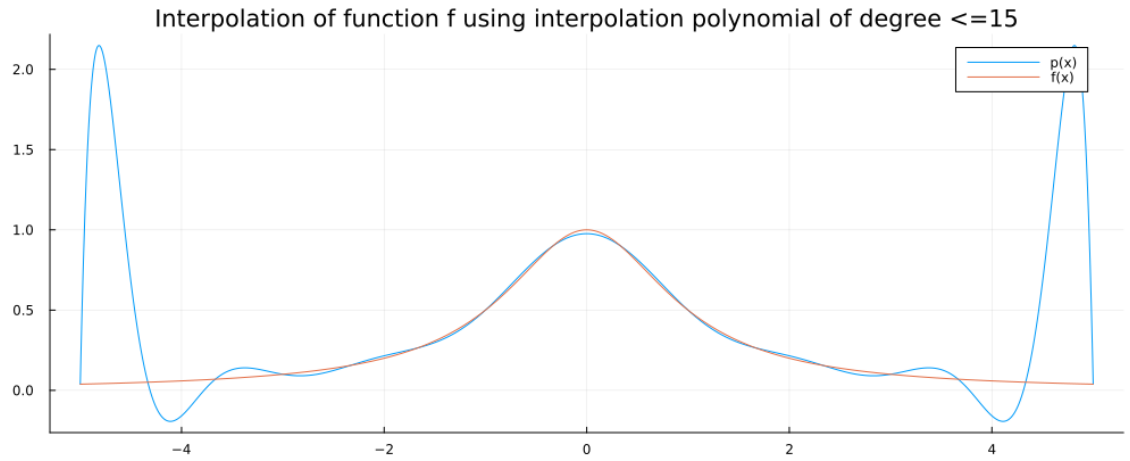
Rysunek 9: Wykres wielomianu interpolacyjnego stopnia ≤ 15 i funkcji interpolowanej $f(x) = |x|$ na przedziale $[-1, 1]$.



Rysunek 10: Wykres wielomianu interpolacyjnego stopnia ≤ 5 i funkcji interpolowanej $f(x) = \frac{1}{1+x^2}$ na przedziale $[-5, 5]$.



Rysunek 11: Wykres wielomianu interpolacyjnego stopnia ≤ 10 i funkcji interpolowanej $f(x) = \frac{1}{1+x^2}$ na przedziale $[-5, 5]$.



Rysunek 12: Wykres wielomianu interpolacyjnego stopnia ≤ 15 i funkcji interpolowanej $f(x) = \frac{1}{1+x^2}$ na przedziale $[-5, 5]$.

6.4 Interpretacja wyników oraz wnioski

W przeciwieństwie do poprzedniego zadania, tym razem interpolacja wielomianem nie działa za dobrze. Nawet zwiększenie stopnia wielomianu nie pomaga w uzyskaniu lepszych wyników.

W przypadku funkcji $|x|$ problemem jest to, że nie jest ona różniczkowalna w punkcie $x = 0$. Wielomiany są funkcjami ciągłymi i gładkimi, więc ciężko za ich pomocą dobrze zinterpolować funkcję, która tak jak $|x|$ ma "ostry"wierzchołek.

Dla funkcji $\frac{1}{1+x^2}$ wykres wielomianu interpolacyjnego zdaje się coraz bardziej odbiegać od rzeczywistego wraz ze zbliżaniem się do końców przedziału. Zjawisko to nazywa się zjawiskiem Runge'go. Pojawia się ono zwłaszcza w przypadku, gdy wykorzystujemy węzły równoodległe takie jak tutaj. Prostym rozwiązaniem tego problemu mogłoby być na przykład zrezygnowanie z równoodległych węzłów i użycie zamiast tego węzłów zagęszczonych w problematycznych miejscach lub zastosowanie węzłów wyznaczanych za pomocą wielomianów Czebyszewa.

Wnioski

Interpolacja wielomianowa jest dobrą i dość poprawnie działającą metodą przybliżania funkcji, gdy mamy dostępne tylko kilka jej wartości. Nie działa ona jednak idealnie. W przypadku funkcji nieróżniczkowalnych nawet miejscowo zwracane przez nią wartości mogą nie do końca być zbliżone do tych poprawnych. Czasem nawet dla funkcji ciągłych i gładkich, dla których powinna działać poprawnie uzyskujemy nieoczekiwane efekty. Nie zawsze zwiększanie stopnia wielomianu poprawi dokładność przybliżenia, czasem może nawet zaszkodzić.

Nie należy więc wierzyć na słowo zwracanym przez metodę wartościom, warto zastosować kilka różnych podziałów przedziału na części, zwłaszcza jeśli jakieś jego fragmenty wydają się problematyczne. Dobrą praktyką jest zbadanie przebiegu funkcji, aby wiedzieć w których momentach należy uważać i gdzie mogą pojawić się problemy. Jeśli mamy taką możliwość, warto porównać uzyskane wartości z dokładnym wykresem funkcji.