Obliczenia naukowe

Lista 4

Arkadiusz Ziobrowski
229728

1 Zadanie pierwsze

1.1 Opis problemu

Celem zadania była implementacja funkcji obliczającej ilorazy różnicowe, bez użycia tablicy dwuwymiarowej (macierzy).

1.2 Rozwiązanie

W języku programowania `Julia` została zaimplementowana funkcja o następującej sygnaturze:

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

gdzie `x` to wektor długości $n+1$ zawierający węzły $x_0, x_1, ..., x_n$, a `f` to wektor długości $n+1$ zawierający wartości interpolowanej funkcji w węzłach $f(x_0), f(x_1), ..., f(x_n)$. Funkcja powinna zwracać wektor `fx` długości $n+1$, który zawiera obliczone ilorazy różnicowe w taki sposób, że wartość pod pierwszym indeksem odpowiada $f[x_0]$, zaś pod $(n+1)$-szym indeksem odpowiada $f[x_0, x_1, ..., x_n]$.

Ilorazy różnicowe spełniają zależność:

$$f[x_i, x_{i+1}, ..., x_{i+j}] = \frac{f[x_{i+1}, x_{i+2}, ..., x_{i+j}] - f[x_i, x_{i+1}, ..., x_{i+j-1}]}{x_{i+j} - x_i}$$

Dla powyższej zależności $f[x_i] = f(x_i)$ oraz $f[x_i, x_{i+1}] = \frac{f(x_{i+1}) - f(x_i)}{x_{i+1} - x_i}$. Znając węzły x_i oraz wartości funkcji funkcji interpolowanej w tych węzłach można za pomocą powyższej zależności utworzyć dwuwymiarową tablicę ilorazów różnicowych.

$$\begin{array}{llllll} x_0 & f[x_0] & f[x_0, x_1] & f[x_0, x_1, x_2] & \dots & f[x_0, x_1, ..., x_n] \\ x_1 & f[x_1] & f[x_1, x_2] & f[x_1, x_2, x_3] & \dots & \\ x_2 & f[x_2] & f[x_2, x_3] & \dots & & \\ & & \vdots & & & \\ x_{n-1} & f[x_{n-1}, x_n] & & & & \\ x_n & f[x_n] & & & & \end{array}$$

W powyższej dwuwymiarowej tablicy zostały umieszczone ilorazy różnicowe dla n węzłów. Jedynie pierwszy wiersz powyższej tablicy zawiera ilorazy, które powinny znaleźć się w wyznaczanym wektorze `fx`. Są to ilorazy wykorzystywane jako współczynniki we wzorze interpolacyjnym Newtona. Można użyć zatem tablicy jednowymiarowej, która będzie wypełniana zgodnie z podanym poniżej algorytmem.

```
1:  function ILORAZYROZNICOWE(x, f)
2:      n ← length(x)
3:      for i = 1 to n do
4:          fx[i] ← f[i]
5:      end for
6:      for j = 1 to n do
7:          for i = n down to j + 1 do
8:              fx[i] ← (fx[i] − fx[i − 1])/(x[i] − x[i − j])
9:          end for
10:     end for
11:     return fx
12: end function
```

Obliczanie ilorazów różnicowych poprzez stworzenie dwuwymiarowej tablicy można zastąpić stworzeniem tablicy jednowymiarowej z ilorazami różnicowymi. Jest to możliwe dzięki sposobowi w jaki taka tablica jest wypełniana. Tablicę tworzymy kolumnami i z dołu do góry. Dzięki takiej kolejności w naszej jednowymiarowej tablicy zawsze znajdują się wartości ilorazów różnicowych, które będa potrzebne do wyliczenia kolejnych wartości. W linijce 4 powyższego algorytmu tablica jest najpierw wypełniana wartościami $f(x_i)$, co odpowiada kolumnie drugiej z dwuwymiarowej tablicy ilorazów różnicowych. W linijce 8 w kroku j-tym wypełniamy tablicę wartościami, odpowiadającymi $(j+1)$-szej kolumnie dwuwymiarowej tablicy ilorazów różnicowych.

2 Zadanie drugie

2.1 Opis problemu

Celem zadania była implementacja funkcji obliczającej wartość wielomianu interpolacyjnego stopnia n w postaci Newtona $N_n(x)$ w punkcie t za pomocą uogólnionego algorytmu Hornera w czasie $O(n)$.

2.2 Rozwiązanie

W języku programowania `Julia` została zaimplementowana funkcja o następującej sygnaturze:

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

gdzie `x` to wektor długości $n+1$ zawierający węzły $x_0, x_1, ..., x_n$, `fx` to wektor długości $n+1$ zawierający ilorazy różnicowe $f[x_0], f[x_0, x_1], ..., f[x_0, x_1, ..., x_n]$, zaś `t` to punkt, w którym należy obliczyć wartość wielomianu. Funkcja powinna zwracać `nt`, będące wartością wielomianu w punkcie t.

Do obliczenia wartości wielomianu w punkcie t zostanie zastosowany uogólniony algorytm Hornera, który można wyrazić za pomocą następujących wzorów:

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

Wzory te można wyznaczyć z wielomianu interpolacyjnego w postaci Newtona. Wielomian Newtona $N_n(x) = f[x_0] + f[x_0, x_1](x - x_0) + ... + f[x_0, x_1, ..., x_n](x - x_0)...(x - x_{n-1})$. Wielomian ten można zapisać w alternatywnej postaci $N_n(x) = (...(f[x_0, x_1, ..., x_n](x - x_{n-1}) + f[x_0, x_1, ..., x_{n-1}])(x - x_{n-2}) + ...f[x_0, x_1])(x - x_0) + f[x_0]$. Stąd też mamy:

$$w_{n-k}(x) = \sum_{i=n-k}^{n} f[x_0, x_1, ..., x_i] \prod_{j=n-k}^{i-1} (x - x_j)$$

Zaczynając od $w_n(t)$ i iterując w dół do $w_0(t)$ dostaniemy po n krokach wartość wielomianu interpolacyjnego w postaci Newtona w punkcie t.

```
1: function warNewton(x, fx, t)
2:     n ← length(x)
3:     w ← fx[n]
4:     for i = n − 1 down to 1 do
5:         w ← fx[i] + w ∗ (t − x[i])
6:     end for
7:     return w
8: end function
```

Powyższy pseudokod prezentuje zastosowanie uogólnionego algorytmu Hornera do wyliczenia wartości wielomianu Newtona stopnia n w punkcie t. W linijce 3 rozpoczynamy iteracje od zastosowania wzoru $w_n(x) = f[x_0, x_1, ..., x_n]$. Następnie w pętli stosujemy wzór na $w_k(x)$ aż dojdziemy do w_0. Pętla wykonuje się $n - 1$ razy, więc asymptotyczna złożoność tego algorytmu wynosi $O(n)$.

3 Zadanie trzecie

3.1 Opis problemu

Celem zadania była implementacja funkcji wyliczającej współczynniki postaci naturalnej wielomianu interpolacyjnego w postaci Newtona, przy znanych współczynnikach wielomianu interpolacyjnego oraz węzłach $x_0, x_1, ..., x_n$.

3.2 Rozwiązanie

W języku programowania `Julia` została zaimplementowana funkcja o następującej sygnaturze:

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

gdzie `x` to wektor długości $n+1$ zawierający węzły $x_0, x_1, ..., x_n$, a `fx` to wektor długości $n+1$ zawierający ilorazy różnicowe $f[x_0], f[x_0, x_1], ..., f[x_0, x_1, ..., x_n]$. Funkcja powinna zwracać wektor `a` długości $n+1$, zawierający obliczone współczynniki postaci naturalnej.

Do rozwiązania zadania został użyty algorytm z zadania drugiego z wprowadzonymi modyfikacjami. Dla wielomianu interpolacyjnego w postaci Newtona $N_n(x) = f[x_0] + f[x_0, x_1](x - x_0) + ... + f[x_0, x_1, ..., x_n](x - x_0)...(x - x_{n-1})$ wyliczamy:

$$\begin{aligned} b_n &= f[x_0, x_1, ..., x_n] \\ b_{n-1} &= f[x_0, x_1, ..., x_{n-1}] + (z - x_{n-1})b_n \\ &\dots \\ b_0 &= f[x_0] + (z - x_0)b_1 \end{aligned}$$

Dla wartości z wyliczone b_i dla $i = 1, ..., n$ są teraz współczynnikami nowego wielomianu w postaci: $N_n(x) = b_0 + b_1(x - z) + ... + b_n(x - z)...(x - x_{n-2})$. Wartości węzłów x_i przesunęły się o jedno w prawo w nowym wielomianie na mocy twierdzenia Bézouta.

Twierdzenie Bézouta. *Reszta z dzielenia wielomianu $f(x)$ przez $x - a$ jest równa $f(a)$.*

Wielomian $f(x)$ podzielony przez $x - a$ możemy zapisać jako $f(x) = q(x)(x - a) + r$, gdzie r to reszta z dzielenia wielomianu przez dwumian $x - a$. Stąd też bierze się przesunięcie węzłów x_i w nowym wielomianie, powstałym z wielomianu $N_n(x)$ po zastosowaniu jednej iteracji algorytmu. Naszym celem jest dzielenie wielomianu $N_n(x)$ przez $(x - 0)$ tak długo, aż każdy nowy węzeł x_i będzie zerem, a zatem wielomian będzie w postaci naturalnej. Dlatego we wzorze na b_i jako wartość z bierzemy zero.

Przypomnijmy teraz postać naszego wielomianu po pierwszym przejściu algorytmu: $N_n(x) = b_0 + b_1(x - z) + ... + b_n(x - z)...(x - x_{n-2})$. Wielomian ten można zapisać alternatywnie jako $N_n(x) = b_0 + (x - z)q(x)$, gdzie $q(x) = b_1 + ... + b_n(x - x0)...(x - x_{n-2})$. Kolejne przejścia algorytmu można wykonywać jedynie dla $q(x)$, gdyż b_0 nie ulegnie już zmianie.

```
1:  function NATURALNA(x, fx)
2:      n ← length(x)
3:      a ← fx
4:      for i = 1 to n do
5:          for j = n down to i do
6:              a[j] ← a[j] + a[j + 1] ∗ (z − x[j])
7:          end for
8:          shift x_i right one position
9:          x[1] ← z
10:     end for
11:     return a
12: end function
```

Powyższy pseudokod jest realizacją opisanego powyżej algorytmu. Pętla zewnętrzna odpowiada n przesunięciom węzłów, aby sprowadzić wielomian do postaci naturalnej poprzez zastępowanie węzła $x_0 = 0$, a $x_{i+1} = x_i$ dla $i \neq 0$. Pętla wewnętrzna wykorzystuje zmodyfikowany uogólniony algorytm Hornera do wyznaczenia b_i w i-tym przejściu pętli zewnętrznej. Jego asymptotyczna złożoność obliczeniowa wynosi $O(n^2)$, gdyż pętla zewnętrzna wykonuje się n razy, a pętla wewnętrzna $O(n)$ razy.

4 Zadanie czwarte

4.1 Opis problemu

Celem zadania była implementacja funkcji interpolującej zadaną funkcję $f(x)$ w przedziale $[a, b]$ za pomocą wielomianu interpolacyjnego stopnia n w postaci Newtona. Zaimplementowana funkcja powinna również narysować wielomian interpolacyjny i interpolowaną funkcję.

4.2 Rozwiązanie

W języku programowania `Julia` została zaimplementowana funkcja o następującej sygnaturze:

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

gdzie `f` to funkcja $f(x)$ zadana jako funkcja anonimowa, a oraz b to krańce przedziały interpolacji, zaś n to stopień wielomianu interpolacyjnego. Funkcja nie zwraca żadnej wartości, ale rysuje wielomian interpolacyjny i interpolowaną funkcję w przedziale $[a, b]$.

W zaimplementowanej funkcji `rysujNnfx` na początku wyznaczana jest wartość $h = (b - a)/n$, która posłuży do obliczenia węzłów równoodległych. Następnie dwa wektory x i y o długości $n + 1$ są zapełniane kolejno wartościami węzłów równoodległych i wartości funkcji f w węźle. Jako kolejny krok są wyliczane ilorazy różnicowe przy użyciu funkcji `ilorazyRoznicowe` zaimplementowanej w zadaniu pierwszym. Następnie ustalana jest liczba punktów, które na wykresie będą odpowiadać wartościom wielomianu interpolacyjnego. Aby wykres wielomianu interpolacyjnego był wygładzony ilość punktów jest sto razy większa niż stopień wielomianu interpolacyjnego. Kolejny fragment funkcji odpowiada za rysowanie wykresów. Początkowo są wyznaczane nowe równoodległe punkty, które będą wielomian interpolacyjny i interpolowana funkcja będą przyjmować jako wartości. Wartość funkcji interpolowanej w punkcie t wynosi $f(t)$, zaś wartość wielomianu interpolacyjnego w punkcie t jest wynikiem wywołania funkcji `warNewton` zaimplementowanej w zadaniu drugim. Wykresy są rysowane przy użyciu biblioteki `PyPlot`.

5 Zadanie piąte

5.1 Opis problemu

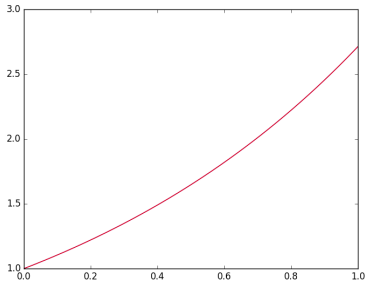
Celem zadania było przetestowanie funkcji `rysujNnfx(f, a, b, n)` dla zadanych przykładów.

5.2 Rozwiązanie

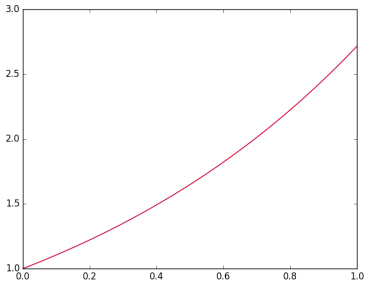
Rozwiązanie zadania polegało na zaimportowaniu modułu stworzonego w ramach zadań od pierwszego do czwartego, a następnie wywołaniu funkcji `rysujNnfx(f, a, b, n)` z następującymi argumentami:

- Funkcja e^x w przedziale $[0, 1]$ i stopniem wielomianu interpolacyjnego $n = 5, 10, 15$.
- Funkcja $x^2 sin(x)$ w przedziale $[-1, 1]$ i stopniem wielomianu interpolacyjnego $n = 5, 10, 15$.

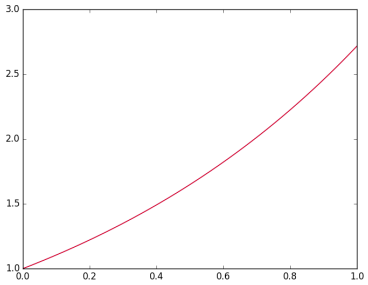
5.3 Wyniki i interpretacja



Rysunek 1: Wielomian interpolacyjny funkcji e^x w przedziale $[0, 1]$ i stopniu $n = 5$.

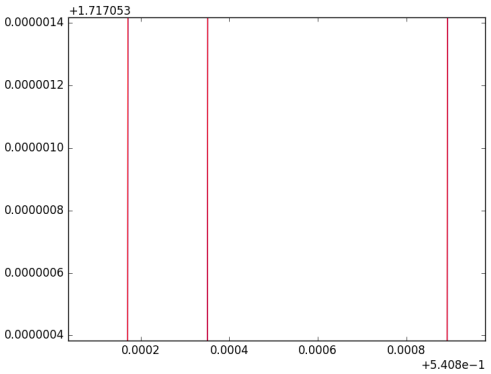


Rysunek 2: Wielomian interpolacyjny funkcji e^x w przedziale $[0, 1]$ i stopniu $n = 10$.



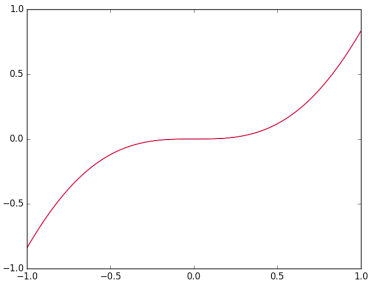
Rysunek 3: Wielomian interpolacyjny funkcji e^x w przedziale $[0, 1]$ i stopniu $n = 15$.

Widoczne powyżej trzy wykresy niemal pokrywają się ze sobą. To niewielkie odchylenie jest widoczne dopiero po znacznym powiększeniu wykresu, na który zostały naniesione wszystkie trzy funkcje.

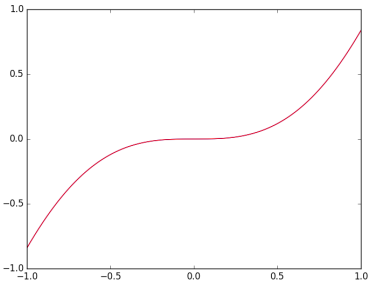


Rysunek 4: Wielomiany interpolacyjne funkcji e^x w przedziale $[0, 1]$ i stopniach $n = 5, 10, 15$.

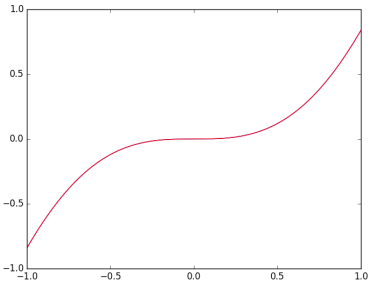
Na powyższym wykresie widoczne są wykresy wielomianów interpolacyjnych dla zadanych stopni wielomianu. Jak widać odchylenie jest niewielkie, bo widoczne dopiero po znacznym powiększeniu wykresu.



Rysunek 5: Wielomian interpolacyjny funkcji $x^2 sinx(x)$ w przedziale $[-1, 1]$ i stopniu $n = 5$.

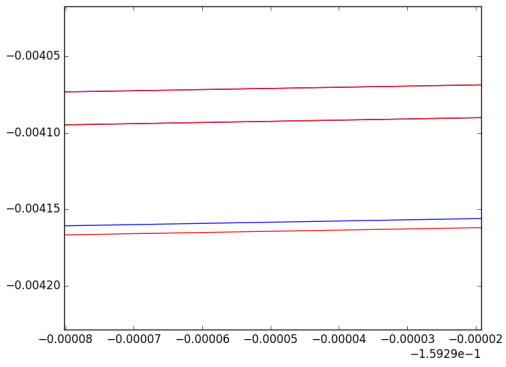


Rysunek 6: Wielomian interpolacyjny funkcji $x^2 sinx(x)$ w przedziale $[-1, 1]$ i stopniu $n = 10$.



Rysunek 7: Wielomian interpolacyjny funkcji $x^2 sinx(x)$ w przedziale $[-1, 1]$ i stopniu $n = 15$.

Również dla drugiej funkcji wielomiany interpolacyjne nie różnią się znacząco. Dopiero po powiększeniu wykresu, na który zostały naniesione trzy wielomiany interpolacyjne, są widoczne odchylenia.



Rysunek 8: Wielomiany interpolacyjne funkcji $x^2 sinx(x)$ w przedziale $[-1, 1]$ i stopniach $n = 5, 10, 15$.

Na powyższym wykresie w kolorze czerwonym zostały narysowane wielomiany interpolacyjne funkcji $f(x) = x^2 sinx(x)$, a kolorem niebieskim dokładne wartości tej funkcji. Jak widać odchylenia są niewielkie i widoczne dopiero po znacznym powiększeniu. Wielomianem, który jest najbardziej odchylony od wartości dokładnych jest wielomian o najniższym stopniu.

5.4 Wnioski

Dla wszystkich funkcji z zadanymi parametrami można było wyznaczyć wielomiany interpolacyjne, które niemal pokrywały się z dokładnymi wartościami funkcji. Małe odchylenia widoczne były dopiero po powiększeniu wykresów. Dla przykładów w zadaniu można również zauważyć, iż im większy stopień wielomianu interpolacyjnego, tym mniejsze odchylenie od wartości dokładnej.

6 Zadanie szóste

6.1 Opis problemu

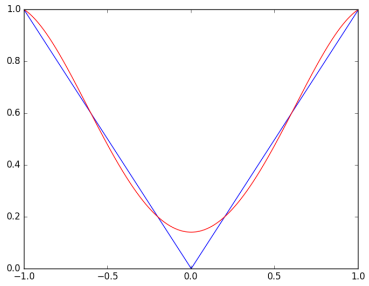
Celem zadania było przetestowanie funkcji `rysujNnfx(f, a, b, n)` dla zadanych przykładów.

6.2 Rozwiązanie

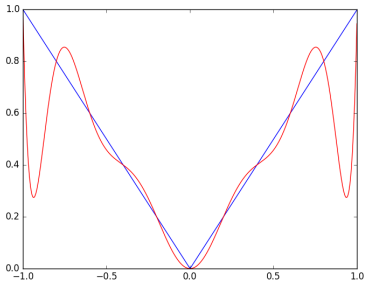
Rozwiązanie zadania polegało na zaimportowaniu modułu stworzonego w ramach zadań od pierwszego do czwartego, a następnie wywołaniu funkcji `rysujNnfx(f, a, b, n)` z następującymi argumentami:

- Funkcja $|x|$ w przedziale $[-1, 1]$ i stopniem wielomianu interpolacyjnego $n = 5, 10, 15$.
- Funkcja $\frac{1}{1+x^2}$ w przedziale $[-5, 5]$ i stopniem wielomianu interpolacyjnego $n = 5, 10, 15$.

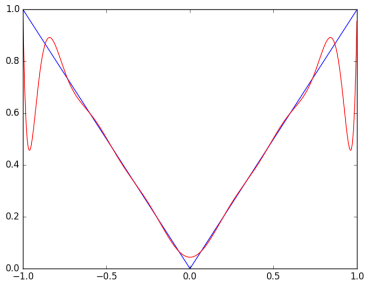
6.3 Wyniki i interpretacja



Rysunek 9: Wielomian interpolacyjny funkcji $|x|$ w przedziale $[-1, 1]$ i stopniu $n = 5$.

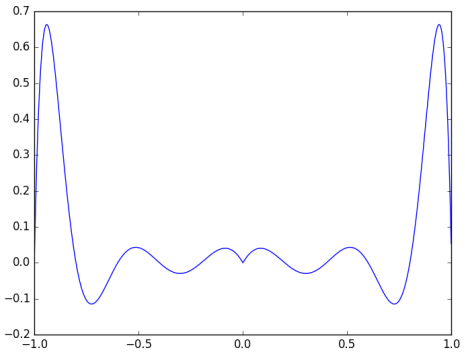


Rysunek 10: Wielomian interpolacyjny funkcji $|x|$ w przedziale $[-1, 1]$ i stopniu $n = 10$.



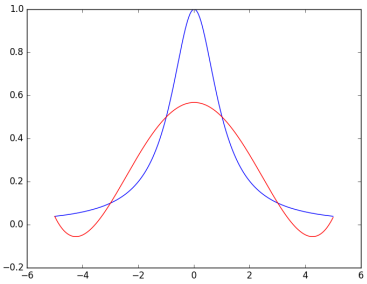
Rysunek 11: Wielomian interpolacyjny funkcji $|x|$ w przedziale $[-1, 1]$ i stopniu $n = 15$.

Na powyższych wykresach można zaobserwować, że wielomiany interpolacyjne nie są zbieżne do funkcji f. Odchylenie jest znacznie bardziej widoczne wraz ze zbliżaniem się wielomianu interpolacyjnego do krańców przedziału $[a, b]$.

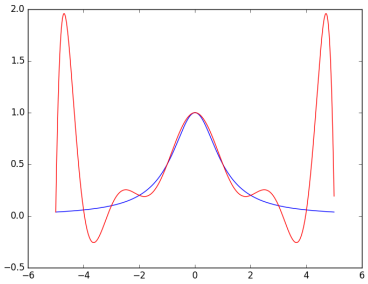


Rysunek 12: Wykres błędu $e = f(x) - p(x)$, gdzie $f(x) = |x|$, a $p(x)$ to wielomian interpolacyjny stopnia $n = 10$.

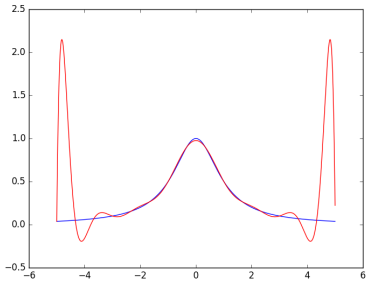
Wykres błędu interpolacji wielomianowej potwierdza wcześniejsze obserwacje. Wraz ze zbliżaniem się do krańców przedziału interpolacji błąd rośnie. Błąd jest symetryczny względem $x = 0$, co wynika z charakterystyki funkcji $f(x) = |x|$. Przykład ten pokazuje również, że zwiększenie stopnia wielomianu interpolacyjnego nie musi powodować zmnieszenia błędu.



Rysunek 13: Wielomian interpolacyjny funkcji $\frac{1}{1+x^2}$ w przedziale $[-1, 1]$ i stopniu $n = 5$.

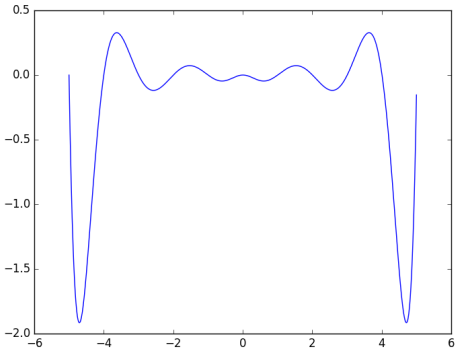


Rysunek 14: Wielomian interpolacyjny funkcji $\frac{1}{1+x^2}$ w przedziale $[-1, 1]$ i stopniu $n = 10$.



Rysunek 15: Wielomian interpolacyjny funkcji $\frac{1}{1+x^2}$ w przedziale $[-1, 1]$ i stopniu $n = 15$.

Na powyższych wykresach można zaobserwować znaczne odchylenia od wartości dokładnej funkcji. Wraz ze zbliżaniem się do krańców przedziału wielomian interpolacyjny zaczyna gwałtowniej oscylować. Jest to zjawisko *Rungego*. Początkowo wraz ze wzrostem liczby węzłów równoodległych dokładność zwiększa się, jednak dla większych n przybliżenie pogarsza się, co jest widoczne szczególnie na krańcach przedziałów.



Rysunek 16: Wykres błędu $e = f(x) - p(x)$, gdzie $f(x) = \frac{1}{1+x^2}$, a $p(x)$ to wielomian interpolacyjny stopnia $n = 10$.

Tak silne oscylowanie na krańcach przedziału jest związane z dużymi błędami interpolacji. Przy interpolacji funkcji za pomocą wielomianów chcemy, aby błąd interpolacji dążył do zera wraz z $n \to \text{inf}$. Dla powyższej funkcji $max_{a \leq \xi_x \leq b} f^{n+1}(\xi_x)$ bardzo szybko dąży do nieskończoności wraz ze wzrostem n. Z tego powodu błąd interpolacji nie jest ograniczony od góry i zwiększa się wraz ze wzrostem n. Jest to spowodowane użyciem węzłów równoodległych. Wielomian oparty na węzłach równoodległych silnie oscyluje na krańcach przedziału, gdyż wielomian ten jest wysokiego stopnia, więc musi mieć dużo zer, a zarazem chce szybko uciec do nieskończoności. Oscylację możemy zredukować poprzez zastosowanie węzłów Czebyszewa, które są zagęszczone na krańcach przedziału i z tego powodu redukują występującą tam oscylację.

6.4 Wnioski

W tym zadaniu, w przeciwieństwie do zadania piątego, zwiększenie stopnia wielomianu interpolacyjnego spowodowało generowanie większych odchyleń. Można było tutaj zaobserwować zjawiska rozbieżności wielomianów spowodowane doborem węzłów równoodległych lub samą charakterystyką funkcji.