Lista 4

Obliczenia naukowe

Patryk Majewski
250134

Uwaga:
Implementacje zadań 1-4, zgodnie z zaleceniem z listy, umieszczone zostały w jednym pliku `interpolacja.jl` i opakowane w moduł `Interpolacja`. Testy tych implementacji znajdują się w pliku `testy.jl`.

Problem

Problem interpolacji wielomianowej formułowany jest następująco: dla danych $n+1$ par postaci (x_i, y_i), gdzie $x_i \neq x_j$, mamy za zadanie znaleźć wielomian $p(x)$ jak najniższego stopnia, że $\forall i \; p(x_i) = y_i$. Na podstawie twierdzenia przedstawionego na wykładzie wiemy, że istnieje dokładnie jeden wielomian stopnia co najwyżej n spełniający ten warunek. W indukcyjnym dowodzie istnienia takiego wielomianu pojawia się wzór

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

dzięki któremu konstruujemy wielomian stopnia o jeden wyższego niż poprzedni, zachowując jednocześnie wartości ustalone dla poprzednich x_i. Okazuje się, że ta postać użyteczna jest z numerycznego i algorytmicznego punktu widzenia – pozwala uniknąć kontaktu ze źle uwarunkowanymi macierzami Vandermonde'a i zmniejszyć złożoność obliczeniową wyznaczania poszczególnych współczynników. Formalnie, mamy

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

Przedstawienie w tej formie nazywane jest wzorem Newtona. W zadaniach wyjaśnimy i poczynimy kolejne kroki w celu uzyskania wielomianu tej postaci, a następnie na jej podstawie jawnie wyznaczymy współczynniki przy poszczególnych potęgach.

Zadanie 1 – Ilorazy różnicowe

Celem uproszczenia notacji, przyjmijmy oznaczenie

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

dookreślając $q_0(x) = 1$. Wówczas postać Newtona wielomianu p możemy zapisać jako

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

Wielomian p jest rozwiązaniem problemu interpolacji, jeśli spełnia

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

Uzyskany w ten sposób układ równań to

$$A \cdot C = Y$$

gdzie $a_{ij} = q_j(x_i)$, $C = [c_0, ..., c_n]^T$, a $Y = [y_0, ..., y_n]^T$. Zauważmy ponadto, że $\forall i < j \; q_j(x_i) = 0$, co czyni macierz A dolnotrójkątną. Ten fakt umożliwia nam łatwiejsze znalezienie c_i poprzez rozwiązywanie układu z góry do dołu. Łatwo wtedy zauważyć, że c_0 zależy od $y_0 = f(x_0)$, c_1 od y_0 i y_1, itd. Będziemy oznaczać tę zależność jako

$$c_i = f[x_0, ..., x_i]$$

nazywając ten czynnik ilorazem różnicowym funkcji f opartym na węzłach $x_0, ..., x_i$. Na wykładzie udało się dowieść, że ilorazy różnicowe spełniają zależność rekurencyjną

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

gdzie $f[x_i] = y_i$. W najprostszym podejściu do wyznaczenia każdego moglibyśmy zatem wykorzystać tablicę dwuwymiarową C, gdzie $C[i, j] = f[x_i, ..., x_{i+j}]$. Zauważmy jednak, że po wyznaczeniu wszystkich ilorazów opartych na k węzłach, częściowe ilorazy oparte na $k-1$ węzłach stają się bezużyteczne (z wyjątkiem $f[x_0, ..., x_{k-1}]$, który jest jednym z szukanych współczynników). Wydaje się zatem, że można w jakiś sposób oszczędzić pamięć potrzebną do rozwiązania zadania. Odpowiedzią na ten niepokój jest algorytm zaprezentowany poniżej.

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

$$d_1' = f[x_0, x_1] = \frac{f[x_1] - f[x_0]}{x_1 - x_0} = \frac{d_1 - d_0}{x_1 - x_0}$$

Zauważmy również, że po wyznaczeniu $f[x_{n-1}, x_n]$ nie użyjemy już do niczego ilorazu $f[x_n]$, możemy zatem nadpisać go tą nową wartością, tymczasem na przykład $f[x_1]$ potrzebny będzie jeszcze 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. Wykonujemy kolejną rundę, ponownie od końca, tym razem zatrzymując się na wyliczeniu d_3. Po n takich rundach nasz wektor wynikowy przyjmie już ostateczną postać. Poglądowy rysunek i pseudokod metody umieszczone zostały poniżej.

Algorithm 1: ilorazy różnicowe

Input: $\bar{x}$ – wektor węzłów , $\bar{y}$ – wektor wartości, n – długość wektorów
Output: $\bar{c}$ – wektor ilorazów różnicowych $f[x_0, ..., x_i]$
$\bar{c} = \bar{y}$;
for j ***from*** *1* ***to*** n **do**
  for i ***from*** n ***down to*** j **do**
    $c_i = \frac{c_i - c_{i-1}}{x_i - x_{i-j}}$;
return $\bar{c}$

← kierunek pracy

$$\begin{array}{ccccc}
f[x_0] & & f[x_1] & & f[x_2] \\
 & \searrow & \downarrow & \searrow & \downarrow \\
 & & f[x_0, x_1] & & f[x_1, x_2] \\
 & & & \searrow & \downarrow \\
 & & & & f[x_0, x_1, x_2]
\end{array}$$

Zadanie 2 – Uogólniony schemat Hornera

Za zadanie mamy teraz wyznaczenie wartości uzyskanego wielomianu postaci Newtona w danym punkcie. Łatwo zauważyć, że standardowa metoda liczenia "według wzoru" pozwala nam na dokonanie tego z kwadratową złożonością. Okazuje się jednak, że możemy rozłożyć nasz wielomian w sposób, który umożliwi złożoność liniową. Mamy bowiem

$$\begin{aligned} p(x) &= \sum_{i=0}^{n} f[x_0, ..., x_i]q_i(x) = f[x_0] + \sum_{i=1}^{n} f[x_0, ..., x_i](x - x_0)...(x - x_i) = \\ &= f[x_0] + (x - x_0)(f[x_0, x_1] + \sum_{i=2}^{n} f[x_0, ..., x_i](x - x_1)...(x - x_i)) = ... \\ &= f[x_0] + (x - x_0)(f[x_0, x_1] + (x - x_1)(...(f[x_0, ..., x_{n-1}] + (x - x_{n-1})f[x_0, ..., x_n])...)) \end{aligned}$$

Będziemy zatem obliczać wartość naszego wielomianu "od środka", zaczynając od najbardziej zagnieżdżonych elementów. Taka metodologia to zasadniczo schemat Hornera w bazie $\{q_i(x) : 0 \leqslant i \leqslant n\}$ zamiast tradycyjnej $\{1, x, ..., x^n\}$. Formalizując, mamy

$$\begin{aligned} w_n(x) &= f[x_0, ..., x_n] \\ w_k(x) &= f[x_0, ..., x_k] + (x - x_k)w_{k+1} \quad \text{dla} \quad k < n \\ p(x) &= w_0(x) \end{aligned}$$

Nietrudno dostrzec, że taki sposób pozwala nam na wyznaczenie poszukiwanej wartości z użyciem jednej pętli, a zatem w czasie liniowym. Pseudokod metody umieszczony został poniżej.

Algorithm 2: uogólniony schemat Hornera

Input: $\bar{x}$ – wektor węzłów , $\bar{c}$ – wektor ilorazów różnicowych $f[x_0, ..., x_i]$, n – długość wektorów, t – punkt, w którym należy obliczyć wartość wielomianu
Output: v – wartość wielomianu w punkcie t
$v = c_n$;
for i ***from*** $n - 1$ ***down to*** 0 **do**
  $v = c_i + (t - x_i) \cdot v$;
return v

Zadanie 3 – Postać naturalna

Postacią naturalną wielomianu nazywamy jego przedstawienie w bazie $\{1, x, x^2, x^3, ...\}$, to znaczy

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

Celem oszczędności znaków, przyjmiemy z powrotem oznaczenie $c_i = f[x_0, ..., x_i]$. Pierwszym spostrzeżeniem potrzebnym nam do rozwiązania problemu jest fakt, że a_n – współczynnik przy x^n – jest równy c_n. Spróbujemy teraz przeżyć kilka pierwszych iteracji algorytmu Hornera z poprzedniej sekcji, skupiając 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 $-x_{n-1}c_n$. Pójdźmy krok dalej:

$$w_{n-2} = c_{n-2} + (x - x_{n_2})w_{n-1}$$

Tutaj sytuacja trochę się zmienia, bo 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:

$$x \ \cdot w_{n-1} = \underbrace{x^2 c_n}_{a_n} + \underbrace{x(c_{n-1} - x_{n-1}c_n)}_{\text{"stare" } a_{n-1}}$$

$$-x_{n-2} \cdot w_{n-1} = \underbrace{-x_{n-2}c_n \cdot x}_{\text{"nowe" do } a_{n-1}} - \underbrace{x_{n-2}(c_{n-1} - x_{n-1}c_n)}_{\text{część startowego } a_{n-2}}$$

dostajemy $w_{n-2} = c_{n-2} - x_{n-2}(c_{n-1} - x_{n-1}c_n) + x(c_{n-1} - (x_{n-1} + x_{n-2})c_n) + x^2c_n$. Kolejna iteracja to mnóstwo znaków, ale pozwoli nam upewnić się w dotychczasowych intuicjach.

$$x \cdot w_{n-2} = \underbrace{x^3c_n}_{a_n} + \underbrace{x^2(c_{n-1} - (x_{n-1} + x_{n-2})c_n)}_{\text{"nowsze stare" } a_{n-1}} + \underbrace{x(c_{n-2} - x_{n-2}(c_{n-1} - x_{n-1}c_n))}_{\text{"stare" } a_{n-2}}$$

$$-x_{n-3} \cdot w_{n-2} = \underbrace{-x_{n-3}c_nx^2}_{\text{"nowe" do } a_{n-1}} + \underbrace{-x_{n-3}(c_{n-1} - (x_{n-1} + x_{n-2})c_n)x}_{\text{"nowe" do } a_{n-2}} + \underbrace{-x_{n-3}(c_{n-2} - x_{n-2}(c_{n-1} - x_{n-1}c_n))}_{\text{część startowego } a_{n-3}}$$

Okazuje się zatem, że w każdej iteracji (cofając się jak w algorytmie Hornera, poza pierwszą) wyznaczamy bazową wartość przy obecnej potędze (dla x^i będzie to $c_i - x_ia_{i+1}$), a następnie musimy jeszcze zaktualizować współczynniki przy wyższych potęgach o "nowo odkryty" składnik. Z powyższych rozważań można zauważyć, że do każdego a_j, że $i < j < n$ dodajemy w i-tej iteracji składnik postaci $-x_{n-i}a_{j+1}$, gdzie a_{j+1} odpowiada obecnemu stanowi naszej wiedzy (widać to w przykładach – "nowe" części dla a_{n-1} i a_{n-2}). Zaprojektujemy zatem algorytm oparty o dokładnie taką technikę. Jego złożoność jest kwadratowa, ponieważ dla każdego kroku "odwinięcia" (kroku algorytmu Hornera) musimy zaktualizować wszystkie współczynniki dla wyższych potęg.

Algorithm 3: postać naturalna wielomianu

Input: $\bar{x}$ – wektor węzłów, $\bar{c}$ – wektor ilorazów różnicowych, n – długość wektorów
Output: $\bar{a}$ – wektor współczynników wielomianu w postaci naturalnej
$a_n = c_n$;
for i ***from*** $n-1$ ***down to*** 0 **do**
  $a_i = c_i - x_i \cdot a_{i+1}$;
  for j ***from*** $i+1$ ***to*** $n-1$ **do**
    $a_j = a_j - x_i \cdot a_{j+1}$;
return $\bar{a}$

Zadanie 4 – Wykresy wielomianów

Celem zadania jest połączenie zaimplementowanych metod w jedną – umożliwiającą graficzne porównanie otrzymanego wielomianu z interpolowaną funkcją. W tym celu na wskazanym przez użytkownika przedziale $[a, b]$ wydzielamy $n+1$ równoodległych węzłów i obliczamy dla nich wartości funkcji. Następnie wyznaczamy ilorazy różnicowe, dzięki którym możemy już ustalać wartość wielomianu w punkcie. Dyskretyzujemy przedział w taki sposób, żeby móc ujrzeć wartości wielomianu także poza węzłami (najlepiej $N \cdot (n+1)$ punktów na przedziale $[a, b]$, gdzie $N > 1$ całkowite, wtedy wśród nich znajdą się nasze węzły). Dla każdego z punktów obliczamy wartość funkcji i wielomianu, a następnie uzyskane wyniki umieszczamy na wykresie.

Algorithm 4: wykres funkcji i wielomianu

Input: f – interpolowana funkcja, $[a, b]$ – przedział interpolacji, n – stopień wielomianu
$h = \frac{1}{n}(b-a)$;
for k ***from*** 0 ***to*** n **do**
  $x_k = a + k \cdot h$;
  $y_k = f(x_k)$;
$\bar{c} =$ ilorazy_różnicowe($\bar{x}$, $\bar{y}$);
$pt = N \cdot (n+1)$;
$dx = \frac{1}{pt-1}(b-a)$;
for i ***from*** 0 ***to*** pt **do**
  $X_i = a + i \cdot dx$;
  $W_i =$ wartość_wielomianu($\bar{x}$, $\bar{c}$, X_i);
  $F_i = f(X_i)$;
wykres($x = \bar{X}$, $y = [\bar{W}, \bar{F}]$);

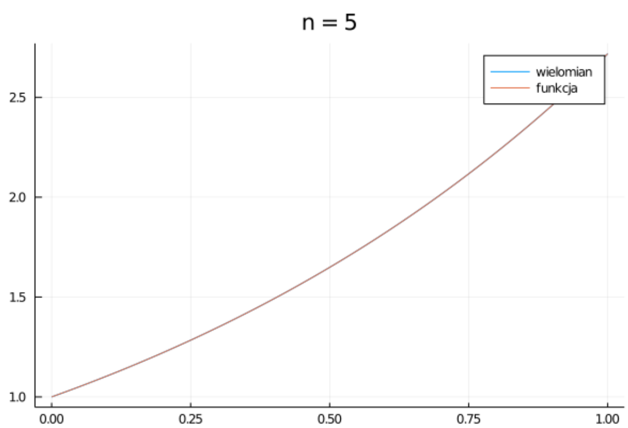
Zadanie 5

Celem zadania było użycie narzędzia skonstruowanego w poprzednim zadaniu na funkcjach

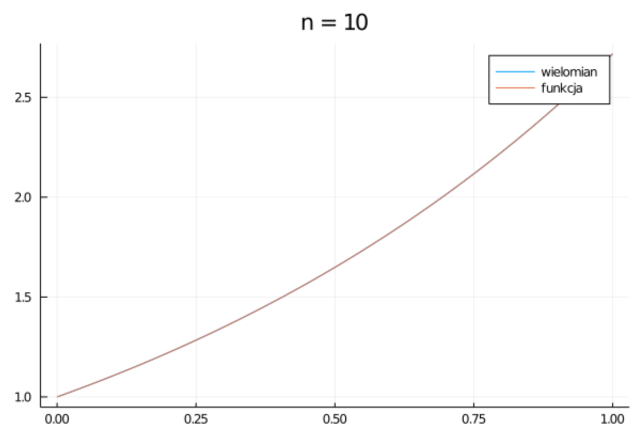
- $f(x) = e^x$ na przedziale $[0, 1]$
- $f(x) = x^2 \cdot \sin x$ na przedziale $[-1, 1]$

dla stopni wielomianu $n = 5, 10, 15$. Wyniki zaprezentowane zostały na rysunkach 1 (e^x) i 2 ($x^2 \cdot \sin x$).

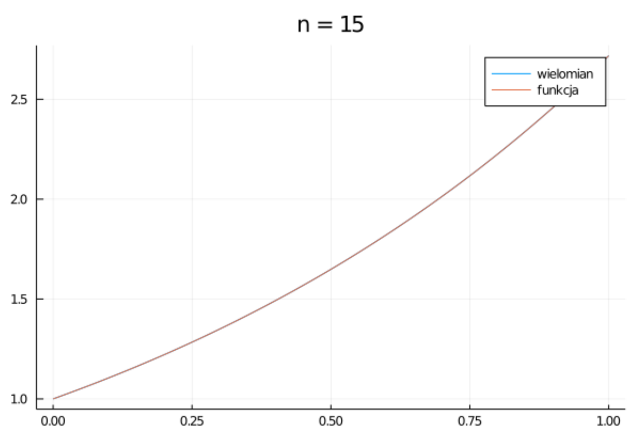
Możemy zaobserwować, że obie testowane funkcje dają się bardzo dokładnie interpolować, to znaczy dla obu wartości wielomianów interpolacyjnych dowolnego ze sprawdzanych stopni niemal pokrywają się z wartościami funkcji na całym zadanym przedziale.



(a)

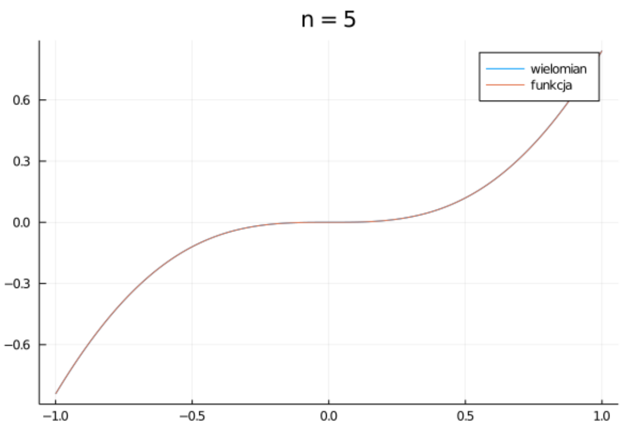


(b)

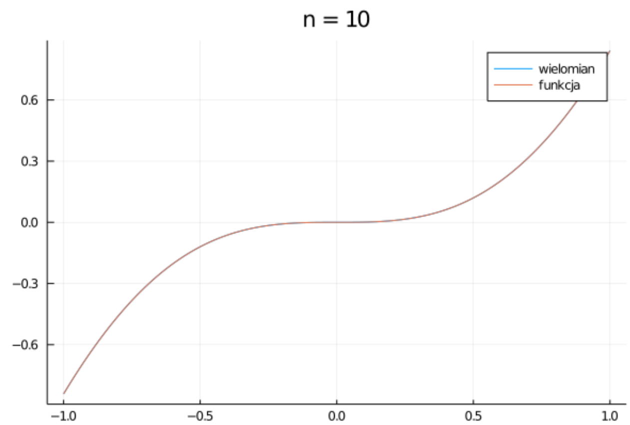


(c)

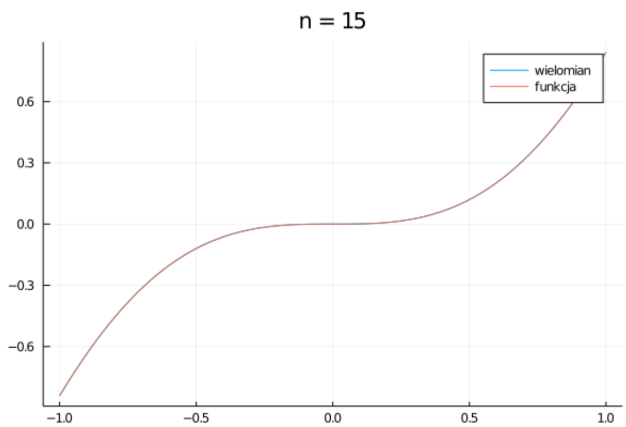
Rysunek 1: Wykresy narysowane przez metodę z zadania 4 dla funkcji $f = e^x$ na przedziale $[0, 1]$ i wielomianów o stopniach 5, 10, 15.



(a)



(b)



(c)

Rysunek 2: Wykresy narysowane przez metodę z zadania 4 dla funkcji $f = x^2 \cdot \sin x$ na przedziale $[-1, 1]$ i wielomianów o stopniach 5, 10, 15.

Zadanie 6

Tym razem za cel mieliśmy zbadanie funkcji

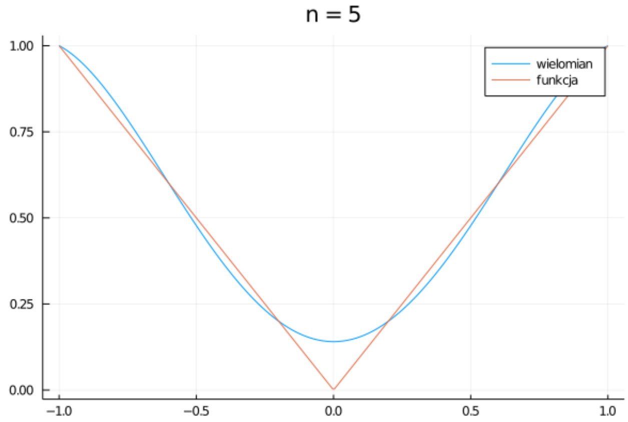
- $f(x) = |x|$ na przedziale $[-1, 1]$
- $f(x) = \frac{1}{1+x^2}$ na przedziale $[-5, 5]$

dla stopni wielomianu $n = 5, 10, 15$. Wyniki zaprezentowane zostały na rysunkach 3 ($|x|$) i 4 ($\frac{1}{1+x^2}$).

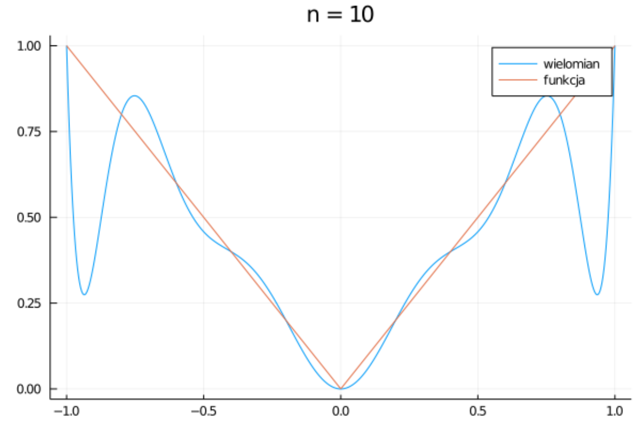
W przeciwieństwie do poprzedniego zadania, tym razem funkcje nie interpolują się za dobrze. Co więcej, wzrost stopnia wielomianu nie niesie za sobą poprawy dokładności.

W przypadku funkcji $|x|$ problemem jest jej nieróżniczkowalność. Intuicyjnie można powiedzieć, że wielomiany są raczej okrągłe, dlatego wierzchołek wykresu stanowi dla nich trudność.

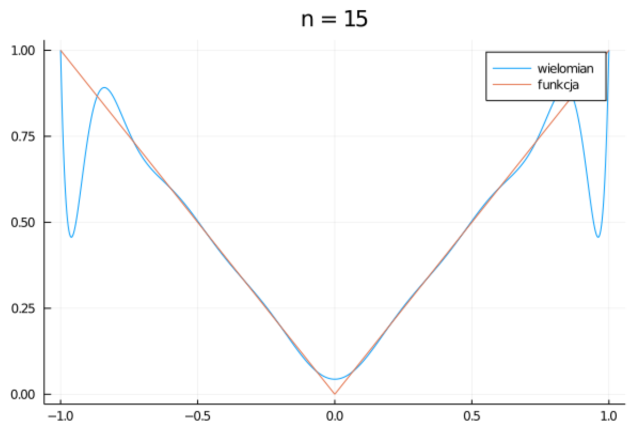
Dla funkcji $\frac{1}{1+x^2}$ obserwujemy zjawisko Rungego, które polega na zwiększaniu się rozbieżności na końcach przedziału wraz ze zwiększaniem stopnia wielomianu interpolacyjnego. Pojawia się ono, gdy węzły interpolacji są równoodległe, co ma miejsce w przypadku naszej implementacji. Rozwiązaniem tego problemu mógłby być na przykład dobór punktów w taki sposób, żeby na przy krańcach przedziału występowało ich większe zagęszczenie.



(a)



(b)

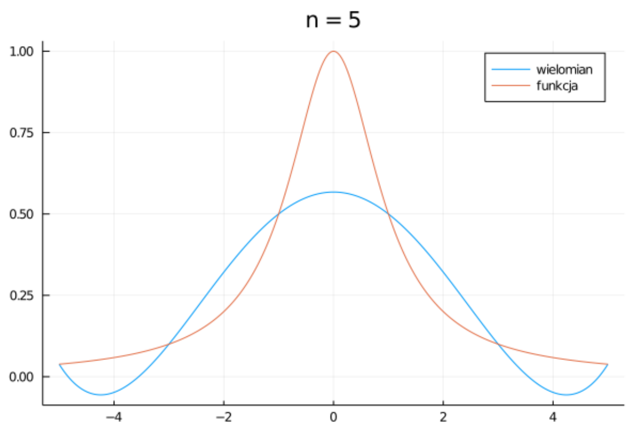


(c)

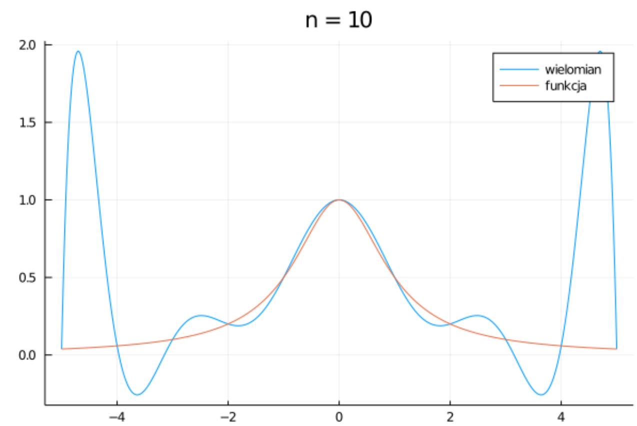
Rysunek 3: Wykresy narysowane przez metodę z zadania 4 dla funkcji $f = |x|$ na przedziale $[-1, 1]$ i wielomianów o stopniach 5, 10, 15.

Wnioski

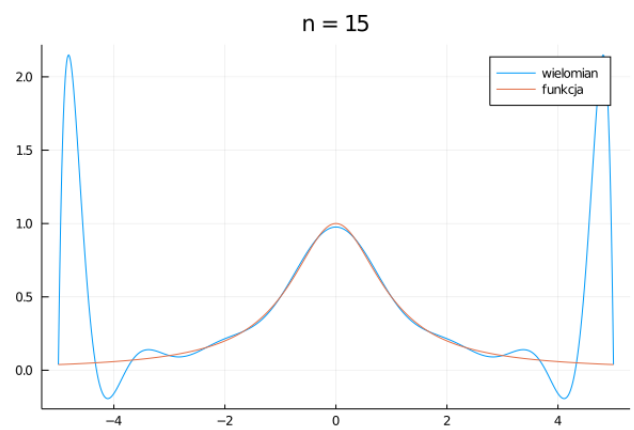
Interpolacja wielomianowa jest dosyć dobrą metodą przybliżania funkcji, gdy znamy jej wartości tylko w niektórych punktach, ale trzeba pamiętać o jej ograniczeniach. Świetnie radzi sobie z gładkimi, zaokrąglonymi funkcjami jak te z zadania 5. Taka intuicja może być jednak zgubna – druga funkcja z zadania 6 również może wydawać się bardzo porządna, a jednak natrafiliśmy na trudności. Należy mieć na uwadze, że bezmyślne zwiększanie stopnia wielomianu może przynieść więcej szkody niż pożytku. W niektórych przypadkach bardziej pomocne może okazać się z kolei przemyślane rozstawienie węzłów interpolacji. Narysowanie wykresu może dać nam pewną intuicję w kwestii tego rozstawu.



(a)



(b)



(c)

Rysunek 4: Wykresy narysowane przez metodę z zadania 4 dla funkcji $f = \frac{1}{1+x^2}$ na przedziale $[-5, 5]$ i wielomianów o stopniach 5, 10, 15.