OBLICZENIA NAUKOWE

Lista nr 4.

Patrycja Paradowska
nr indeksu: 244952
Prowadzący laboratoria: Mgr inż. Marta Słowik

07.12.2019r.

Lista laboratoryjna nr 4. obejmowała zagadnienia związane z interpolacją. Funkcje do zadań 1-4 zostały zaimplementowane w języku `Julia` i umieszczone w module o nazwie PolynomialInterpolation. Zostały również stworzone programy testujące.

1 Zadanie 1. - Ilorazy różnicowe

1.1 Przedstawienie problemu

Celem zadania pierwszego było utworzenie funkcji obliczającej ilorazy różnicowe na podstawie danej tablicy węzłów i odpowiadających im wartości funkcji. Należało efektywnie wykorzystać pamięć i nie korzystać z tablicy dwuwymiarowej (macierzy).
Iloraz różnicowy n-tego rzędu obliczany jest według wzoru rekurencyjnego, $i, j \in 0, ..., n$. Można zauważyć, iż iloraz różnicowy nie jest zależny od kolejności węzłów (x_i), a właściwość ta znajduje zastosowanie przy praktycznym użyciu ilorazów różnicowych.

a) dla $n = 0$

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

b) dla $n = 1$

$$f[x_i, x_j] = \frac{f(x_j) - f(x_i)}{x_j - x_i},$$

c) dla $n > 1$

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

Iloraz różnicowy to wielkość opisująca przyrost funkcji na danym przedziale. Jest ona wykorzystywana przy zagadnieniu interpolacji, czyli jednej z metod numerycznych, która zajmuje się przybliżaniem funkcji. Uzyskiwane jest to dzięki wyznaczaniu w zadanym przedziale tak zwanej funkcji interpolacyjnej.

1.2 Opis algorytmu

Znając węzły x_i i wartości funkcji w tych węzłach $f(x_i)$ (będące jednocześnie ilorazami różnicowymi zerowego rzędu $f[x_0]$) można obliczyć za pomocą wzoru rekurencyjnego ilorazy różnicowe wyższych rzędów, tworząc następującą tablicę trojkątną:

$$\begin{array}{cc|cccccc}
x_0 & f[x_0] & f[x_0,x_1] & f[x_0,x_1,x_2] & \cdots & f[x_0,\ldots,x_{n-1}] & f[x_0,\ldots,x_n] \\
x_1 & f[x_1] & f[x_1,x_2] & f[x_1,x_2,x_3] & \cdots & f[x_1,\ldots,x_{n-1}] & \\
\cdots & \cdots & \cdots & \cdots & \cdots & & \\
x_{n-1} & f[x_{n-1}] & f[x_{n-1},x_n] & & & & \\
x_n & f[x_n] & & & & &
\end{array}$$

Przyjmując następujące oznaczenie $c_k = f[x_i, x_{i+1}, \ldots, x_{i+j}]$:

$$\begin{array}{cc|cccccc}
x_0 & c_{00} & c_{01} & c_{02} & \cdots & c_{0,n-1} & c_{0,n} \\
x_1 & c_{10} & c_{11} & c_{12} & \cdots & c_{1,n-1} & \\
\cdots & \cdots & \cdots & \cdots & & & \\
x_{n-1} & c_{n-1,0} & c_{n-1,1} & & & & \\
x_n & c_{n,0} & & & & &
\end{array}$$

Jesteśmy więc w stanie podać wzór na obliczenie czynników c_{ij} o następującej postaci:

$$c_{ij} = \frac{c_{i+1,j-1} - c_{i,j-1}}{x_{i+j} - x_i}.$$

Algorytm wynikający bezpośrednio z zastosowania wzoru rekurencyjnego, który konstruuje dwuwymiarową tablicę jest pamięciowo nieefektywny. Analiza algorytmu obliczania kolejnych ilorazów różnicowych zwraca uwagę, iż nie musimy wcale do ich wyliczenia używać tablicy dwuwymiarowej. Tablica jednowymiarowa z jednym wskaźnikiem (ozn. tę tablicę jako f_x) okazuje się wystarczająca.
Początkowymi wartościami zmiennych fx_i są $c_{i,0} = f(x_i)$, a następnymi wartościami $c_{i-1,1}, \ldots, c_{1,i-1}, c_{0,i}$ dla każdej kolejnej zmiennej fx_i. Można dostrzec, iż zmienne tablicy fx z każdym przejściem tworzą kolejne kolumny tablicy ilorazów różnicowych, a wewnątrz każdej kolumny aktualizowane są kolejno od dołu do góry. Zaimplementowany algorytm przedstawiony został poniżej:

Dane :
x — wektor długości $n+1$ zawierający węzły $x_0, \ldots, x_n$,
f — wektor długości $n+1$ zawierający wartości interpolowanej funkcji w węzłach $f(x_0), \ldots, f(x_n)$.
Wyniki:
fx — wektor długości $n+1$ zawierający obliczone ilorazy różnicowe.

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

Obliczenia są wykonywane w arytmetyce `Float64`. W każdej iteracji zaprezentowanego algorytmu otrzymywane są w ten sposób kolejne kolumny tablicy trójkątnej niezbędne do wyliczenia kolejnych ilorazów różnicowych. Kolumny są aktualizowane "od dołu" (linie 6-9), co zapewnia, iż tablica fx zawierać będzie w każdej kolejnej iteracji ilorazy potrzebne do wyliczenia nowych wartości w późniejszych krokach.

2 Zadanie 2. - Wielomian interpolacyjny w postaci Newtona

2.1 Przedstawienie problemu

Drugie zadanie polegało na implementacji algorytmu z zadania 8. z listy ćwiczeń nr 4., czyli na utworzeniu funkcji obliczającej wartość wielomianu interpolacyjnego stopnia n w postaci Newtona $N_n(x)$ w punkcie $x = t$ za pomocą uogólnionego algorytmu Hornera działającej w czasie liniowym ($O(n)$). Postać wielomianu interpolacyjnego Newtona można przedstawić za pomocą ilorazów różnicowych. Wyraża się on wzorem pokazującym zależność wielomianu interpolacyjnego N_n od funkcji f:

$$N_n(x) = \sum_{i=0}^{n} f[x_0, x_1, \ldots, x_i] \prod_{j=0}^{i-1} (x - x_j).$$

Przedstawiona powyżej metoda jest dobrze uwarunkowana pod wzgledem numerycznym. Zaletą jest to, iż dodanie nowych punktów (x_i, y_i) nie narusza obliczonych wcześniej współczynników $c_k = f[x_0, x_1, \ldots, x_k]$. Powyższą zależność można również łatwo zapisać za pomocą uogólnionych wzorów Hornera. Na zajęciach pokazaliśmy, że poniższa formuła (uogólniony algorytm Hornera) pozwala na wyliczenie wartości wielomianu interpolacyjnego Newtona:

$$\begin{array}{lcl} w_n(x) & = & f[x_0, x_1, \ldots, x_n] \\ w_k(x) & = & f[x_0, x_1, \ldots, x_k] + (x - x_k) w_{k+1} \quad (k = n-1, \ldots, 0) \\ N_n(x) & = & w_0(x) \end{array}$$

Rzeczywiście, starając się wyliczyć interesującą nas wartość $N_n(x)$ rozwijamy rekurencyjny wzór:
$N_n(x) = f[x_0, x_1, \ldots, x_k] + (x - x_k)(f[x_0, x_1, \ldots, x_{k+1}] + (x - x_{k+1}) w_{k+2})$
$= f[x_0, x_1, \ldots, x_k] + (x - x_k)(f[x_0, x_1, \ldots, x_{k+1}] + (x - x_{k+1})(f[x_0, x_1, \ldots, x_{k+2}] + (x - x_{k+2}) w_{k+3}))$
$= \ldots$
$= f[x_0] + f[x_0, x_1](x - x_1) + \ldots + f[x_0, \ldots, x_n](x - x_1) \ldots (x - x_n)$

który zakończy się w momencie, gdy dojdziemy do $k = n$, czyli dla $w_n(x)$, za x podstawiając nasz argument.

2.2 Opis algorytmu

Warunkiem postawionym w treści zadania było stworzenie algorytmu działającego w czasie $O(n)$, dostając na wstępie wektor węzłów `x` oraz wektor ilorazów różnicowych `fx`. Funkcja obliczająca wartość wielomianu interpolacyjnego stopnia n w postaci Newtona $N_n(x)$ w punkcie $x = t$ wykorzystującą podany algorytm Hornera przedstawiona została w poniższym pseudokodzie. Wszystkie obliczenia są wykonywane przy użyciu arytmetyki `Float64`.

Dane :

`x` — wektor długości $n+1$ zawierający węzły $x_0, \ldots, x_n$,
`fx` — wektor długości $n+1$ zawierający ilorazy różnicowe,
`t` — punkt, w którym należy obliczyć wartość wielomianu.

Wyniki:

`nt` — wartość wielomianu w punkcie `t`.

```
1 function warNewton(x, fx, t)
2 |  l ← length(fx)
3 |  nt ← fx[l]
4 |  for i ← l − 1 downto 1 do
5 |  |  nt ← fx[i] + (t − x[i]) · nt
6 |  end
7 |  return nt
```

Przypisalismy do l długość tablicy fx, a następnie do nt wartości fx[l]. Następnie w pętli od l-1 do 1 następowało przypisywanie do nt wartośći wyliczonej za pomocą podanego wczesniej przeze mnie wzoru. Ostatnim etapem było zwrócenie nt. Przedstawiony algorytm gwarantuje liniową złożoność (wyliczenie szukanej wartści wymaga jednego przejścia pętli dla n węzłów).

3 Zadanie 3. - Postać naturalna wielomianu interpolacyjnego

3.1 Przedstawienie problemu

Celem zadania 3. była implementacja algorytmu z zadania 9. z listy ćwiczeń nr 4., gdzie pokazaliśmy, że mając wielomian w postaci Newtona, jesteśmy w stanie, przy pomocy wielomianów pomocniczych z metody Hornera, wyznaczyć współczynniki postaci naturalnej. W tym zadaniu należało dla zadanych współczynników wielomianu interpolacyjnego w postaci Newtona $c_0 = f[x_0], c_1 = f[x_0, x_1], c_2 = f[x_0, x_1, x_2], \ldots, c_n = f[x_0, \ldots, x_n]$ (ilorazy różnicowe) oraz węzłów $x_0, x_1, \ldots, x_n$ napisać funkcję obliczającą w czasie $O(n^2)$ współczynniki $a_0, \ldots, a_n$ jego postaci naturalnej, tzn. $a_n x^n + a_{n-1} x^{n-1} + \ldots + a_1 x + a_0$

3.2 Opis algorytmu

Wszystkie obliczenia są wykonywane przy użyciu arytmetyki `Float64`. Aby odszukać współczynniki dla naturalnej postaci wielomianu interpolacyjnego Newtona, wykorzystano uogólnione wzory Hornera przedstawione w poprzednim zadaniu. Idea rozwiązania nie jest skomplikowana. Warto zauważyć, że współczynnik a_n przy najwyższej potędze wielomianu w postaci naturalnej jest równy współczynnikowi c_n przy najwyższej potędze wielomianu w postaci Newtona. Z tego faktu wynika również, iż w_n z uogólnionego algorytmu Hornera jest również równy a_n. Mając $a_n = w_n$ w następnych krokach algorytmu będą tworzone wartości a_i bazujące na współczynnikach a_{i+1}. Aby znaleźć zależności między kolejnymi a_i algorytm przechodzi po wszystkich w_i. Na początku algorytmu korzystamy jednak z faktu, że $a_n = c_n$ (linia 3). A jest tablicą długości n z wyliczanymi współczynnikami. Następnie, w sposób zbliżony do wcześniejszego zadania, wyznaczamy kolejne wartości częściowe dla wielomianu interpolacyjnego (odbywa się to w linii 5 pseudokodu). Różnica jest jednak taka, że podczas każdej iteracji każdy "składowy" wielomian doprowadzamy do postaci naturalnej (linie 6-8) i w ten sposób uzyskujemy kolejne współczynniki dla postaci naturalnej wielomianu interpolacyjnego. W każdej iteracji uaktualniane są wszystkie współczynniki przy potęgach wyższych niż aktualnie obliczana (każdy współczynnik a_j pomniejszany jest o wartość $a_{j+1}x_i$), tzn, każdy tymczasowy wielomian w_i doprowadzany jest do postaci naturalnej. Możemy zaobserwować, iż algorytm składa się z dwóch pętli. Pętla zewnętrzna wykona się n razy, podobnie jak i pętla wewnętrzna, w najgorszym przypadku wykona n iteracji.

Zatem otrzymujemy pożądaną złożoność algorytmu rzędu $\mathcal{O}(n^2)$.

Dane :
x — wektor długości $n+1$ zawierający węzły $x_0, \ldots, x_n$,
fx — wektor długości $n+1$ zawierający ilorazy różnicowe.
Wyniki:
a — wektor długości $n+1$ zawierający obliczone współczynniki postaci naturalnej.

```
1  function naturalna(x, fx)
2      n ← length(fx)
3      a[n] ← fx[n]
4      for i ← n − 1 downto 1 do
5          a[i] ← fx[i] − a[i + 1] · x[i]
6          for j ← i + 1 to n − 1 do
7              a[j] ← a[j] − a[j + 1] · x[i]
8          end
9      end
10     return a
```

4 Zadanie 4. - Interpolacja funkcji i jej wykres

4.1 Przedstawienie problemu

Celem zadania czwartego było napisanie, używając języka `Julia`, funkcji, która zinterpoluje zadaną funkcję $f(x)$ na konkretnym przedziale $[a, b]$. Miało się to odbywać przy wykorzystaniu wielomianu interpolacyjnego Newtona stopnia n. Wynikiem funkcji jest wykres, na którym są zobrazowane funkcja $f(x)$ oraz interpolujący ją wielomian. W iterpolacji należało użyć węzłów równoodległych, czyli $x_k = a + kh$, $h = (b-a)/n$, $k = 0, 1, \ldots, n$. Ponadto, należało skorzystać z zaimplementowanych w poprzednich zadaniach funkcji *ilorazyRoznicowe* oraz *warNewton*.

4.2 Opis algorytmu

Algorytm zaimplementowany w tym zadaniu nie jest skomplikowany i jego działanie wydaje się być oczywiste, ponieważ na podstawie otrzymanych danych i zaimplementowanych we wcześniejszych zadaniach funkcji, wyznacza on ilorazy różnicowe i wartość wielomianu interpolacyjnego. Na początku zostanie przedstawiony pseudokod algorytmu.

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

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

```
1: function RYSUJ-NNFX(f, a, b, n)
2:     maxlWezlow ← n + 1
3:     h ← (b−a)/n
4:     kh ← 0
5:     mnoznik ← 20
6:     for i ← 1 to maxlWezlow do
7:         x[i] ← a + kh
8:         y[i] ← f(x[i])
9:         kh ← kh + h
10:    end for
11:    f_x ← ILORAZY-RÓŻNICOWE(x, y)
12:    kh ← 0
13:    maxlWezlow ← maxlWezlow · mnoznik
14:    h ← (b−a)/(maxlWezlow−1)
15:    for i ← 1 to maxlWezlow do
16:        plotargs[i] ← a + kh
17:        plotip[i] ← WAR-NEWTON(x, f_x, plotargs[i])
18:        plotval[i] ← f(plotargs[i])
19:        kh ← kh + h
20:    end for
21:    DRAW-PLOT(plotargs, plotval, plotip)
22: end function
```

Wszystkie obliczenia są wykonywane przy użyciu arytmetyki `Float64`. Najpierw zostają zdefiniowane warunki początkowe (zmienne h i kh) dla późniejszego obliczania kolejnych, równoodległych od siebie węzłów wielomianu. Odbywa się to w liniach 3 oraz 4 naszego pseudokodu. Ustalona zostaje również maksymalna liczba węzłów użytych do wyznaczenia wielomianu przechowywana w zmniennej `maxlWezlow`. Następnie rozpoczynamy wyznaczanie kolejnych węzłów interpolacji $(x_1, \ldots, x_{n+1})$, które są rozmieszczone od siebie w odległości $\frac{b-a}{n}$ w przedziale $[a, b]$. Liczymy także wartości funkcji f w stworzonych węzłach – $(f(x_1), \ldots, f(x_{n+1}))$ (węzły są wyznaczane z krokiem h). Jest to widoczne w liniach 6-10. Następnie przy pomocy funkcji zaimplementowanej w zadaniu 1. o nazwie `ilorazyRoznicowe` obliczone zostały ilorazy różnicowe dla utworzonych węzłów - linia 11. Dla uzyskania większej dokładności wykresów zarówno wielomian interpolacyjny jak i funkcja f próbkowane są w $20 \times (n + 1)$ równoodległych punktach. Wartości wielomianu zostaną wyznaczone w $maxlWezlow \cdot mnoznik$ węzłach - linia 13, a mnożnikiem jest tutaj ustalona wartość 20, czyli nastąpi próbkowanie w $20 \cdot (n + 1)$ węzłach. Potem rozpoczyna się kolejna pętla for (15-19), gdzie następuje obliczenie wartości wielomianu za pomocą funkcji `warNewton` z zadania 2. Otrzymane tym sposobem wartości umożliwiają nam szybkie i łatwe narysowanie odpowiedniego wykresu funkcji f oraz jej wielomianu interpolacyjnego, a do tego zadania został wykorzystany pakiet `PyPlot` w języku `Julia`. W kolejnych dwóch zadaniach użyjemy i przetestujemy zaimplementowaną tutaj funkcję i zostaną zaprezentowane wykresy wygenerowane przy jej użyciu.

5 Zadnie 5. - Interpolacja funkcji e^x oraz $x^2 \sin x$

5.1 Przedstawienie problemu

Zadanie polegało na przetestowaniu funkcji `rysujNnfx(f,a,b,n)` z zadania 4. na następujących przykładach:

(a) $f(x) = e^x$, $[a, b] = [0, 1]$, $n \in \{5, 10, 15\}$,

(b) $f(x) = x^2 \sin x$, $[a, b] = [-1, 1]$, $n \in \{5, 10, 15\}$.

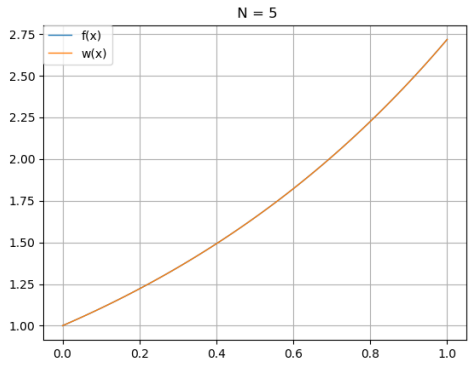
5.2 Opis algorytmu

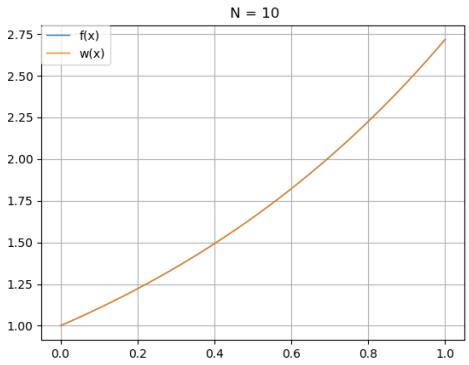
Do rozwiązania zadania użyto funkcji `rysujNnfx(f,a,b,n)`, której opis znajduje się w poprzednim zadaniu.

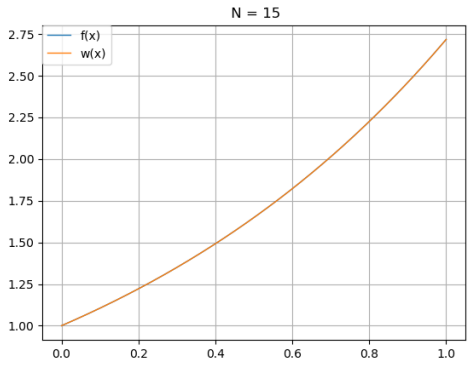
5.3 Uzyskane wyniki

Wykresy otrzymane za pomocą funkcji `rysujNnfx(f,a,b,n)` znajdują się poniżej.

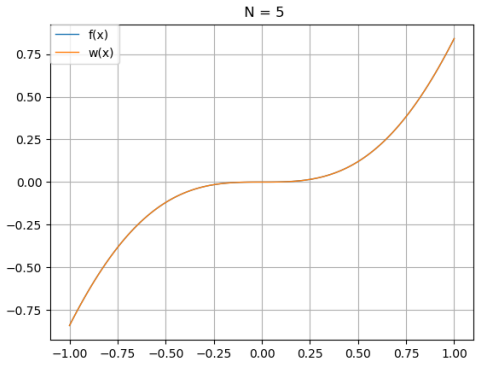
WYKRESY DLA FUNKCJI e^x i jej wielomianu interpolacyjnego dla danego stopnia n

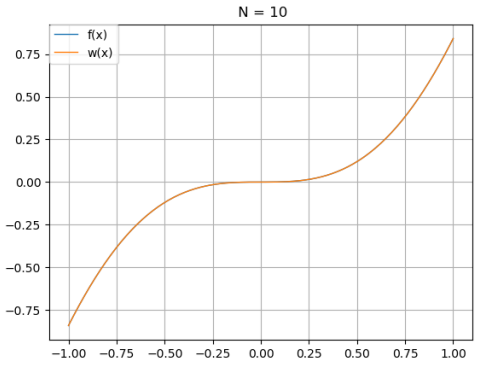


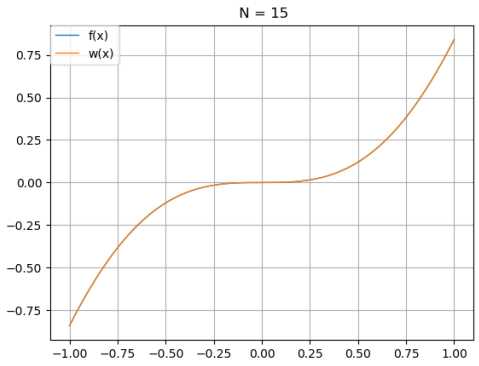




WYKRESY DLA FUNKCJI $x^2 \sin(x)$ i jej wielomianu interpolacyjnego dla danego stopnia n







5.4 Obserwacje i wnioski

Otrzymane wielomiany interpolacyjne pokrywają się z rzeczywistym przebiegiem obu badanych funkcji. Zarówno dla funkcji e^x jak i również $x^2 \sin x$ na zadanych przedziałach wielomiany interpolacyjne są bardzo bliskie interpolowanym funkcjom, na wykresach nie zaobserwowano rozbieżności. Dla wielomianu piątego stopnia różnice są rzędu 10^{-6}, a wraz ze zwiększeniem stopnia wielomianu jeszcze maleją. Można więc uznać, że w badanych przypadkach interpolacja wielomianowa z zastosowaniem węzłów równoodległych daje bardzo dobre wyniki. Spowodowane jest to wyliczaniem wartości dla równo odległych od siebie węzłów, co daje w konsekwencji otrzymanie bardzo dobrego przybliżenia rzeczywistych danych.

6 Zadanie 6. - Interpolowanie funkcji $|x|$ i $\frac{1}{1+x^2}$ – zjawisko rozbieżności

6.1 Przedstawienie problemu

Zadanie polegało na przetestowaniu funkcji `rysujNnfx(f,a,b,n)` z Zadania **??** na następujących przykładach:

(a) $f(x) = |x|$, $[a, b] = [-1, 1]$, $n \in \{5, 10, 15\}$,

(b) $f(x) = \frac{1}{1+x^2}$, $[a, b] = [-5, 5]$, $n \in \{5, 10, 15\}$.

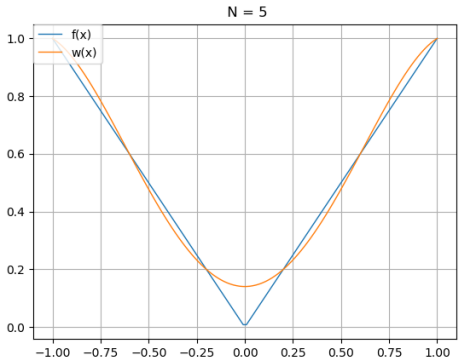
6.2 Opis algorytmu

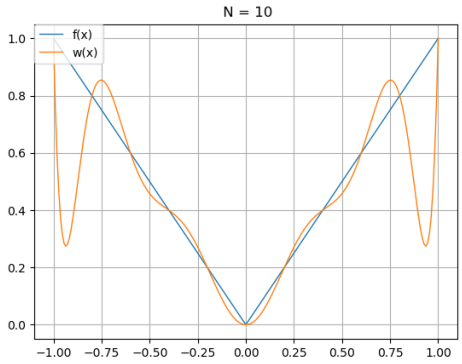
Do rozwiązania zadania użyto funkcji `rysujNnfx(f,a,b,n)` z zadania czwartego.

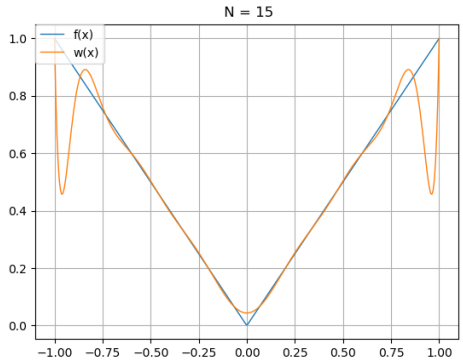
6.3 Uzyskane wyniki

Wygenerowane wykresy przedstawione zostały poniżej:

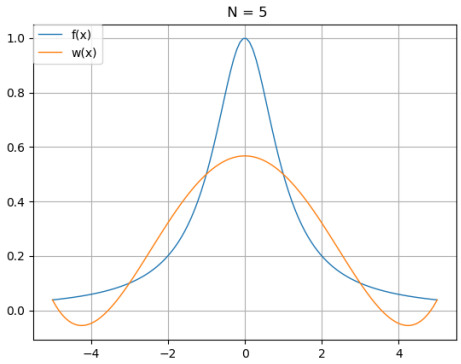
WYKRESY DLA FUNKCJI $|x|$ i jej wielomianu interpolacyjnego dla danego stopnia n

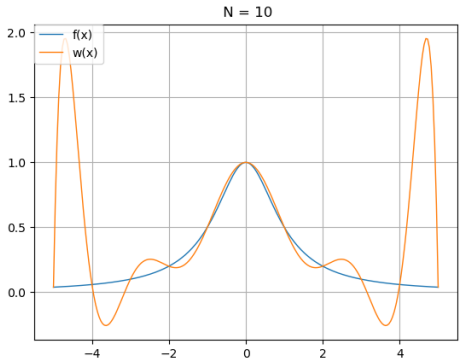


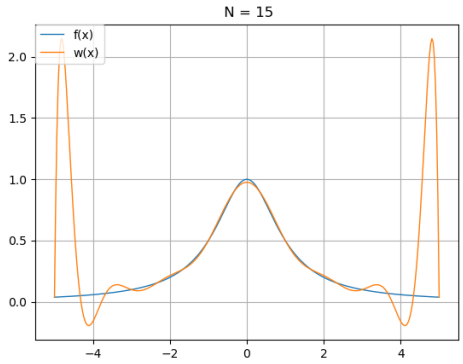




WYKRESY DLA FUNKCJI $\frac{1}{1+x^2}$ i jej wielomianu interpolacyjnego dla danego stopnia n







6.4 Obserwacje i wnioski

Intuicja nabyta dzięki zadaniu wcześniejszemu (nr 5.) mogłaby nam podpowiadać, iż wykresy funkcji interpolacyjnej znów będą przypominać oryginalne. Jednak otrzymane wykresy wyraźnie temu zaprzeczają. W przeciwieństwie do zadania poprzedniego widoczne są znaczne rozbieżności między wartościami funkcji i wartościami wielomianu interpolującego, które zmieniają się wraz ze stopniem wielomianu interpolującego.

W przypadku funkcji $f(x) = |x|$ widoczne odchylenia pojawiają się w punkcie $x = 0$, w którym funkcja jest nieróżniczkowalna. Nieróżniczkowalność tej funkcji w największej mierze odpowiada za odchylenia w tym przypadku. Największa niedokładność została zaobserwowana tym miejscu dla wielomianu stopnia 5. Odchylenia jednak są widoczne również dla wielomianów wyższych stopni, choć ich charakter różni się w zależności od parzystości stopnia wielomianu (ze względu na symetrię wykresu na badanym przedziale, przybliżenie w punkcie $x = 0$ będzie zależeć od tego, czy jakiś węzeł znajdzie się w tym punkcie, czy węzły będą symetrycznie rozmieszczone po obu jego stronach). Znaczne odchylenia od rzeczywistego przebiegu funkcji są dobrze zauważalne także na krańcach badanego przedziału, przy czym pojawiają się one dla wielomianów wyższych rzędów.

Przybliżenie funkcji $f(x) = \frac{1}{1+x^2}$ wielomianem 5. stopnia pokrywa się z jej rzeczywistym przebiegiem tylko w punktach węzłowych. Powiększanie stopnia wielomianu wpływa poprawnie na przybliżenie w środkowej części przedziału. Powoduje jednak znaczące zaburzenia na jego krańcach. Możemy więc zaobserować, iż zwiększanie stopnia wielomianu interpolacyjnego może powodować pogorszenie jakości aproksymacji funkcji, szczególnie na końcach przedziałów. Takie zjawisko jest nazywane *zjawiskiem Rungego*.

Polega ono na pogorszeniu jakości interpolacji wielomianowej mimo tego, iż ilość węzłów rośnie. Jest to typowe dla interpolacji posiadających stałe odległości między punktami - czyli takiej, która została zastosowana. Zauważyć można również, że wielomian interpolacyjny zachowuje się różnie w zależności od rozmieszczenia węzłów. Dzięki temu, odpowiednio wybierając węzły, można znacznie zredukować zjawisko Rungego. W sytuacji, jeśli chcielibyśmy polepszyć jakość interpolacji, należałoby w sposób gęstszy rozmieścić węzły w tych obszarach, gdzie interpolacja jest trudniejsza. Optymalną interpolację można uzyskać poprzez przyjęcie jako węzły zera odpowiedniego wielomianu Czebyszewa (jest to rosyjski matematyk, który sformułował problem dokładności interpolacji jako problem znalezienia wielomianu, który najlepiej przybliżałby zero na danym przedziale). Wielomiany oparte na węzłach Czebyszewa mają dużo mniejsze oscylacje, gdyż są oparte na węzłach mocno zagęszczonych przy krańcach przedziału.