Sprawozdanie

Obliczenia naukowe, lista 4

Aleksandra Malinowska, 244925, WPPT INF, semestr 5, grudzień 2019

1. Wstęp

Sprawozdanie to zawiera analizę problemów oraz wyniki ich rozwiązań z zadań z listy nr 4 doktora Pawła Zielińskiego z przedmiotu Obliczenia naukowe. Wszystkie programy potrzebne do rozwiązania zadań zostały napisane w języku Julia w arytmetyce Float64.

2. Lista zadań

2.1. Zadanie 1

2.1.1. Opis problemu

W tym zadaniu należało zaimplementować funkcję `ilorazyRoznicowe(x,f)` obliczającą ilorazy różnicowe, gdzie:

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

`f` – wektor długości $n + 1$ zawierający wartości interpolowanej funkcji w węzłach

Wynikiem działania funkcji jest wektor długości $n + 1$ zawierający obliczone ilorazy różnicowe. Zastrzeżeniem w zadaniu było, aby do obliczeń nie używać tablicy dwuwymiarowej.

2.1.2. Algorytm

Algorytm będący rozwiązaniem tego zadania pochodzi z książki „Analiza numeryczna” (D. Kincaid, W. Cheney).

Ilorazy różnicowe spełniają zależność

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

dzięki której możemy obliczyć ich wartości. Znając węzły x_i oraz wartości funkcji $f(x_i)$, czyli ilorazy $f[x_i]$ zerowego rzędu, można za pomocą powyższego wzoru stworzyć trójkątną tablicę ilorazów różnicowych wyższych rzędów.

Przy założeniu, że

$$c_{ij} = f[x_i, x_{x+1}, \dots, x_{i+j}],$$

możemy ową tablicę przedstawić następująco

x_0	c_{00}	c_{01}	c_{02}	...	$c_{0(n-1)}$	c_{0n}
x_1	c_{10}	c_{11}	c_{12}	...	$c_{1(n-1)}$	
...	...	...	...			
x_{n-1}	$c_{(n-1)0}$	$c_{(n-1)1}$				
x_n	c_{n0}					

a wtedy wzór rekurencyjny przyjmuje postać

$$c_{ij} = \frac{c_{(i+1)(j-1)} - c_{i(j-1)}}{x_{i+j} - x_i}$$

Algorytm konstrukcji powyższej tablicy opiera się na tym wzorze. Na początku przepisujemy drugą kolumnę tablicy do wektora wynikowego, a następnie zapisujemy kolejne wartości w kolejności od dołu do góry, aby zawsze zachowywać tylko te wartości, które będą jeszcze potrzebne. Dokładny algorytm jest opisany jest pseudokodem poniżej (n – liczba węzłów, d_i – wartość c_{0i}).

for $i = 0$ **to** n **do**
    $d_i \leftarrow f(x_i)$
end do
for $j = 0$ **to** n **do**
    for $i = n$ **downto** j **do**
        $d_i \leftarrow \frac{d_i - d_{i-1}}{x_i - x_{i-j}}$
    end do
end do

2.1.3. Rozwiązanie

Rozwiązaniem tego zadania jest implementacja powyższego pseudokodu w języku Julia w funkcji `ilorazyRoznicowe(x,f)` w module `Interpolacja`.

2.2. Zadanie 2

2.2.1. Opis problemu

Problemem tego zadania jest zaimplementowanie funkcji `warNewton(x,fx,t)` obliczającej 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 na wejściu:

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

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

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

oraz zwraca na wyjściu wartość wielomianu w punkcie t.

2.2.2. Algorytm

Wielomian interpolacyjny stopnia n w postaci Newtona można, przy pomocy ilorazów różnicowych, przedstawić następująco:

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

Użycie uogólnionego algorytmu Hornera do obliczenia wartości tego wielomianu w punkcie t sprowadza się do zastosowania poniższych wzorów:

$$w_n(t) = f[x_0, x_1, \dots, x_n]$$

$$w_k(t) = f[x_0, x_1, \dots, x_k] + (t - x_k) w_{k+1}(t), \text{ gdzie } k = n-1, \dots, 0$$

$$N_n(t) = w_0(t)$$

Algorytm obliczania $N_n(t)$ polega na początkowym przypisaniu wartości $f[x_0, \dots, x_n]$ do zmiennej n_t, a następnie obliczaniu jej kolejnych wartości zgodnie z powyższym wzorem w pętli $n-1, \dots, 0$. Ostatecznie algorytm zwraca wynik, który jest jednocześnie wartością $w_0(t)$. Dokładny przebieg algorytmu przedstawia poniższy pseudokod.

$n_t = f_x[n]$
for $i \leftarrow n - 1$ ***downto*** 0 ***do***
    $n_t \leftarrow f_x[i] + (t - x[i]) \cdot n_t$
end for
return n_t

2.2.3. Rozwiązanie

Rozwiązaniem tego zadania jest implementacja powyższego pseudokodu w języku Julia w funkcji `warNewton(x,fx,t)` w module `Interpolacja`.

2.3. Zadanie 3

2.3.1. Opis problemu

W tym zadaniu należało napisać funkcję `naturalna(x,fx)`, która przyjmując na wejściu:

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

`fx` – wektor długości $n - 1$ zawierający ilorazy różnicowe (współczynniki wielomianu w postaci newtona)

ma zwrócić na wyjściu współczynniki postaci naturalnej wielomianu interpolacyjnego. Funkcja ta ma działać w czasie $O(n^2)$.

2.3.2. Algorytm

Algorytm wyznaczania współczynników postaci naturalnej wielomianu Newtona $w(x)$ opiera się również na uogólnionym schemacie Hornera przedstawionym w poprzednim zadaniu. Niech

$a_0, \dots, a_n$ – współczynniki postaci naturalnej wielomianu oraz

$c_0, \dots, c_n$ – współczynniki postaci Newtona.

Algorytm rozpoczynamy od faktu, że $a_n = c_n$. Następnie w pętli $i = n-1, \dots, 0$ obliczamy kolejne wartości a_i i dla każdej z nich doprowadzamy aktualny wielomian do postaci naturalnej, dzięki czemu uzyskujemy kolejne współczynniki wielomianu $w(x)$. Dokładny algorytm przedstawia pseudokod poniżej.

$a[n] = f_x[n]$
for $i \leftarrow n-1$ **downto** 0 **do**
    $a[i] \leftarrow f_x[i] - a[i+1] \cdot x[i]$
    for $j \leftarrow i+1$ **to** $n-1$ **do**
        $a[j] \leftarrow a[j] - a[j+1] \cdot x[i]$
    end for
end for
return a

Analizując przebieg tego algorytmu można stwierdzić, że pierwsza pętla wykona się n razy, druga również co najwyżej n razy, stąd złożoność powyższego algorytmu jest równa $O(n^2)$.

2.3.3. Rozwiązanie

Rozwiązaniem tego zadania jest implementacja powyższego pseudokodu w języku Julia w funkcji `naturalna(x,fx)` w module `Interpolacja`.

2.4. Zadanie 4

2.4.1. Opis problemu

Problemem tego zadania było zaimplementowanie funkcji `rysujNnfx(f,a,b,n)` interpolującą daną funkcję $f(x)$ w przedziale $[a, b]$ za pomocą wielomianu interpolacyjnego $w(x)$ stopnia n w postaci Newtona, a następnie rysującą na wykresie $w(x)$ oraz $f(x)$.

Zastrzeżeniami w tym zadaniu było:

1) użycie węzłów równoodległych tj. $x_k = a + kh, h = \frac{b-a}{h}, k = 0,1, \dots, n$
2) skorzystanie z funkcji `ilorazyRoznicowe` oraz `warNewton` (w celu niewyznaczania wielomianu $w(x)$ w jawnej postaci)

Wynikiem działania programu jest wykres $w(x)$ oraz $f(x)$.

2.4.2. Algorytm

Algorytm rozwiązujący to zadanie polega na wyznaczeniu węzłów, obliczeniu wartości funkcji $f(x)$ w węzłach, wyznaczeniu ilorazów różnicowych a następnie obliczeniu wartości wielomianu $w(x)$ w węzłach.

Rozpoczynamy zatem od ustalenia ilości węzłów $nodes \leftarrow n + 1$ oraz odstępu między nimi $h \leftarrow \frac{b-a}{n}$. Następnie w pętli obliczamy kolejne węzły oraz wartości funkcji $f(x)$ w tych węzłach. Mając te dane, korzystamy z funkcji `ilorazyRoznicowe(x,fx)`, aby obliczyć ilorazy różnicowe.

W tym momencie, dzięki zgromadzonym danym, moglibyśmy obliczyć wartości wielomianu $w(x)$, jednak najpierw, aby zwiększyć dokładność wykresu, zwielokrotnimy ilość punktów potrzebnych do jego narysowania. Dlatego ustalamy nową wartość $nodes \leftarrow nodes \cdot 15$ oraz $h \leftarrow \frac{b-a}{nodes-1}$. Następnie w pętli obliczamy nowe punkty $plot_x$, które posłużą do rysowania wykresu, oraz wartości funkcji w tych punktach. Dodatkowo obliczamy wartości wielomianu interpolacyjnego za pomocą funkcji `warNewton(x,fx,t)`. Tych danych używamy do narysowania wykresu $f(x)$ oraz $w(x)$.

Poniżej znajduje się dokładny przebieg tego algorytmu w postaci pseudokodu.

$nodes \leftarrow n + 1$
$h = \frac{b-a}{n}$
for $i \leftarrow 1$ **to** $nodes$ **do**
    $x[i] \leftarrow a + (i-1) \cdot h$
    $f_x[i] \leftarrow f(x[i])$
end for
$c \leftarrow ilorazyRoznicowe(x, f_x)$
$nodes \leftarrow nodes \cdot 15$
$h \leftarrow \frac{b-a}{nodes-1}$
for $i \leftarrow 1$ **to** $nodes$ **do**
    $plot_x[i] \leftarrow a + (i-1) \cdot h$
    $plot_{f_x}[i] \leftarrow f(plot_x[i])$
    $plot_{w_x}[i] \leftarrow warNewton(x, c, plot_x[i])$
end for
$drawPlot(plot_x, plot_{f_x}, plot_{w_x})$

Dzięki takiej implementacji algorytm ten spełnia wszystkie założenia zadania.

2.4.3. Rozwiązanie

Rozwiązaniem tego zadania jest implementacja powyższego algorytmu w funkcji `rysujNnfx(f,a,b,n)` w języku Julia w module `Interpolacja`. Implementacja funkcji `drawPlot(args, seriesA, seriesB)` wymaga użycia biblioteki `Plots`.

2.5. Zadanie 5

2.5.1. Opis problemu i rozwiązanie

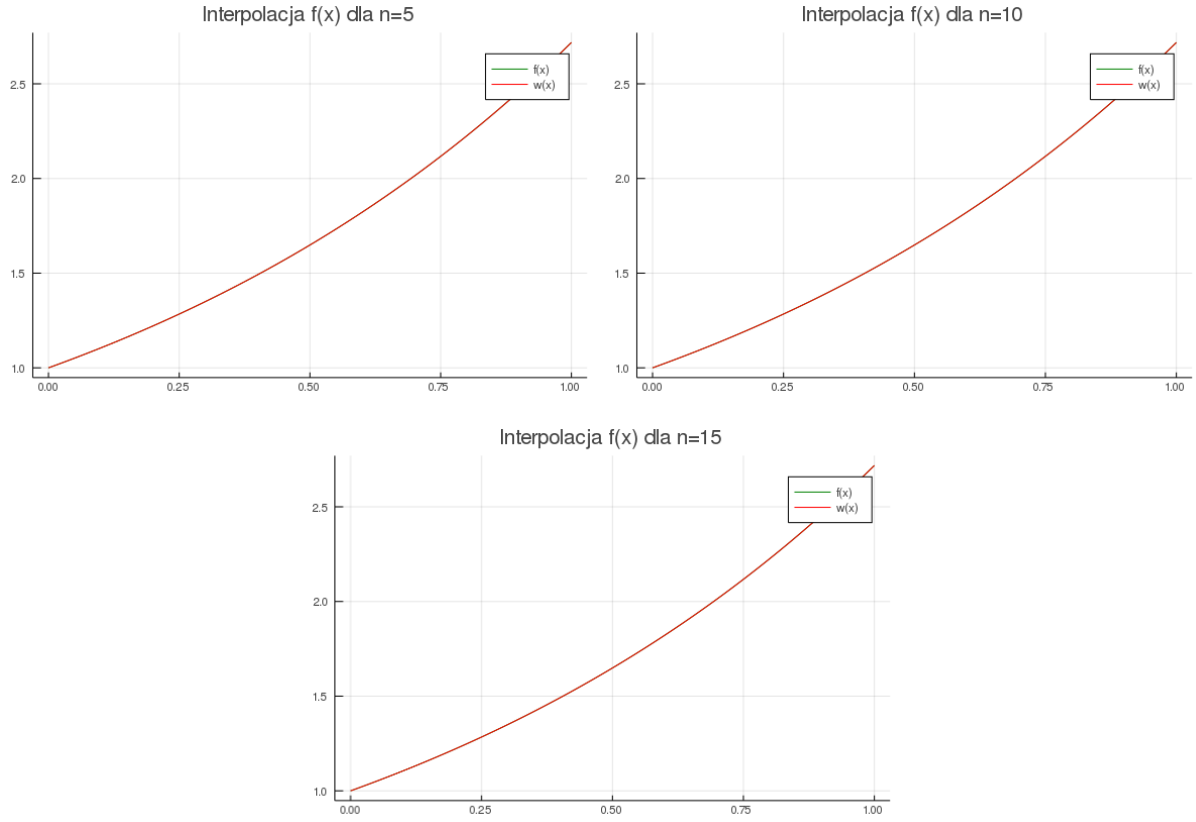
W tym zadaniu należało przetestować funkcję `rysujNnfx(f,a,b,n)` w dwóch przypadkach:

1) $f(x) = e^x$, w przedziale [0,1], dla $n = 5, 10, 15$
2) $g(x) = x^2 \sin(x)$, w przedziale $[-1,1]$, dla $n = 5, 10, 15$

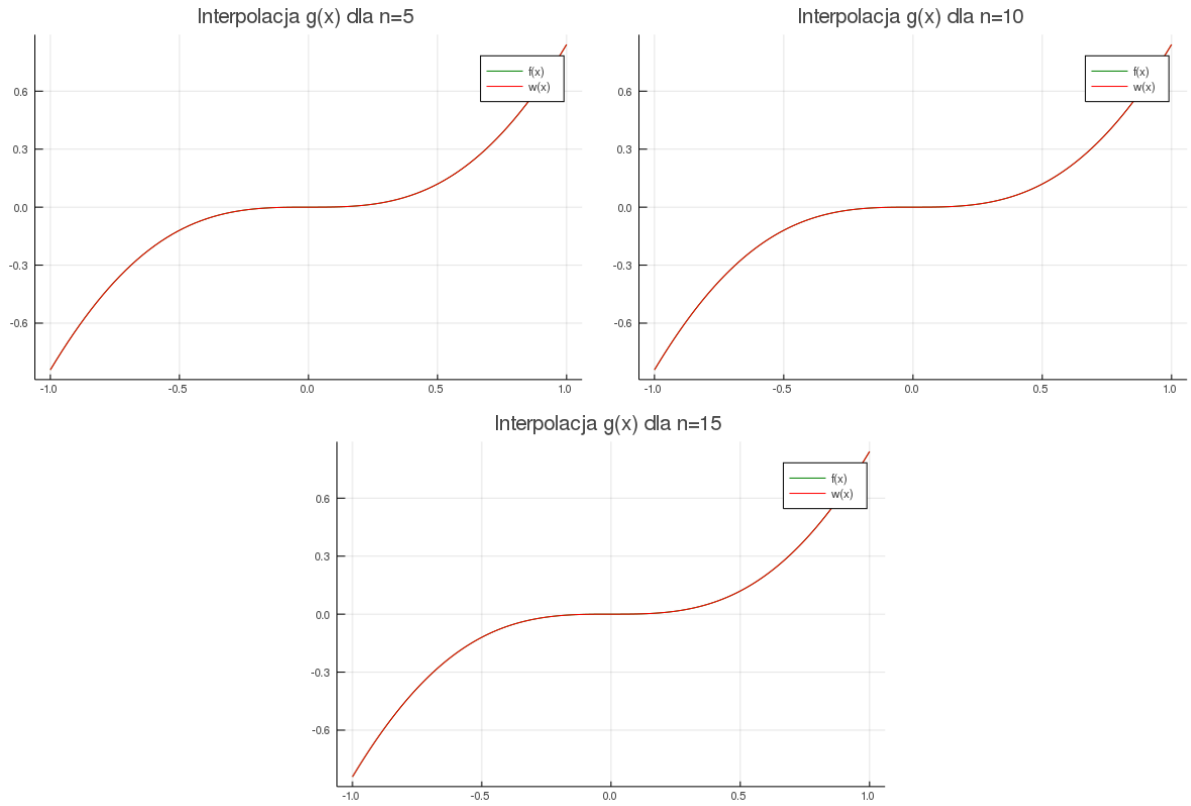
Rozwiązaniem tego zadania jest program używający modułu `Interpolacja`, w którym uruchomiona zostaje funkcja `rysujNnfx(f,a,b,n)` dla kolejnych sześciu przypadków (dwóch podpunktów powyżej oraz każdego n z osobna).

2.5.2. Wyniki

Wynik działania programu dla pierwszego przypadku przedstawiają wykresy poniżej.



Wynik działania programu dla przypadku (2) przedstawiają poniższe wykresy.



2.5.3. Wnioski

Na powyższych wykresach wyraźnie widać, że wartości funkcji interpolowanych oraz wielomianów interpolujących pokrywają się ze sobą. Stąd bierze się wniosek, że wartości $f(x)$ oraz $w(x)$ są prawie równe, jeśli nie identyczne. Wynika to rozstawu węzłów, które są oddalone od siebie w równych odległościach. Dzięki temu uzyskujemy dane, które są bardzo dobrym przybliżeniem danych rzeczywistych.

2.6. Zadanie 6

2.6.1. Opis problemu i rozwiązanie

W tym zadaniu, analogicznie jak w zadaniu 5, należało przetestować funkcję rysujNnfx(f,a,b,n) dla podanych dwóch przypadków:

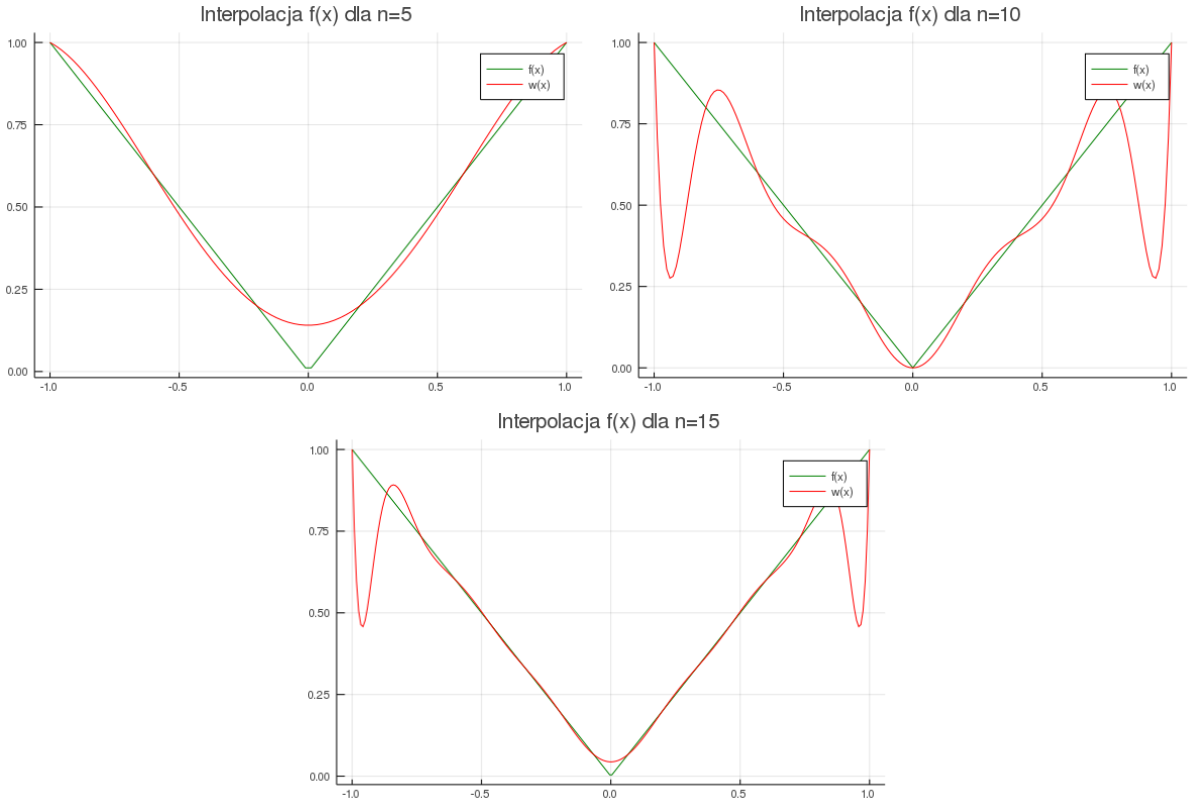
1) $f(x) = |x|$, w przedziale $[-1,1]$, dla $n = 5, 10, 15$
2) $g(x) = \frac{1}{1+x^2}$, w przedziale $[-5,5]$, dla $n = 5, 10, 15$

Należało tu zwrócić szczególną uwagę na zjawisko rozbieżności oraz zjawisko Runge'go.

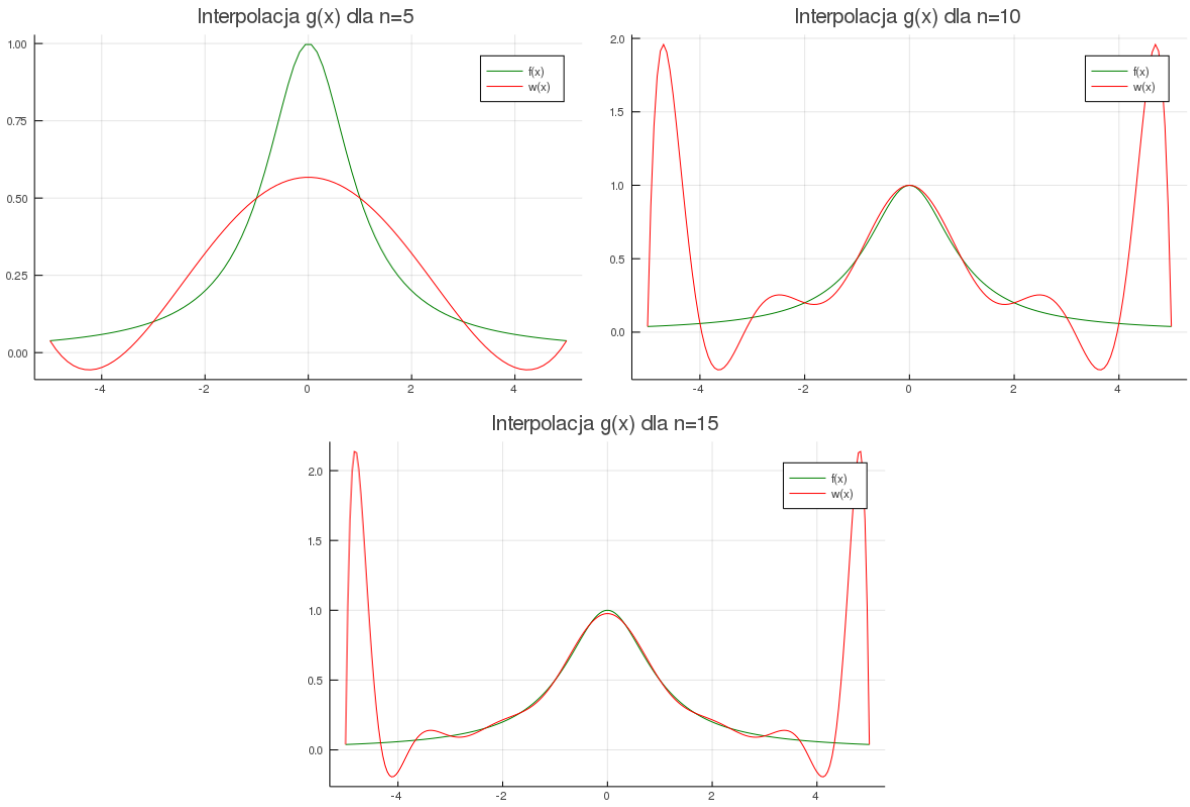
Rozwiązaniem tego zadania jest program analogiczny do programu w zadaniu 6.

2.6.2. Wyniki

Wyniki działania programu dla pierwszego przypadku znajdują się na poniższych wykresach.



Wyniki działania programu dla drugiego przypadku znajdują się na wykresach poniżej.



2.6.3. Wnioski

Na powyższych wykresach widać, że wraz ze wzrostem n przybliżenie funkcji $f(x)$ poprawia się z wyjątkiem końców przedziałów, gdzie znacznie się pogarsza. Prowadzi to do wniosku, że mamy tu do czynienia ze zjawiskiem nazywanym efektem Runge'go, które polega na pogorszeniu jakości interpolacji wielomianowej, mimo zwiększenia liczby jej węzłów. Ma na to wpływ równoodległość węzłów oraz interpolacja za pomocą wielomianu wysokiego stopnia. Zapobieganie temu zjawisku wiąże się ze zwiększeniem ilości węzłów na krańcach przedziałów, np. poprzez wykorzystanie wielomianu Czybyszewa n-tego stopnia.

3. Podsumowanie

Przeprowadzone eksperymenty miały na celu zapoznanie się z metodą interpolacji funkcji oraz zbadanie jej skuteczności w różnych przypadkach.