Obliczenia Naukowe - laboratorium 3

Łukasz Machnik

3 grudnia 2023

1 Zadanie 1

Należy napisać funkcję obliczającą ilorazy różnicowe. Funkcja jest zadeklarowana następująco:

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

Dane:

x	-	wektor długości $n+1$ zawierający węzły $x_0, \dots, x_n$ x[1]=$x_0, \dots,$x[n+1]=x_n
f	-	wektor długości $n+1$ zawierający wartości interpolowanej funkcji w węzłach $f(x_0), \dots, f(x_n)$

Zwracany wynik:

fx	-	wektor długości $n+1$ zawierający obliczone ilorazy różnicowe fx[1]=$f[x_0]$, fx[2]=$f[x_0, x_1]$, ..., fx[n]=$f[x_0, \dots, x_{n-1}]$, fx[n+1]=$f[x_0, \dots, x_n]$

Iloraz różnicowy ma następującą własność dla $0 \leqslant i \leqslant n$:

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

Iloraz różnicowe wyższych rzędów można przedstawić za pomocą następującego rekurencyjnego wzoru (dla $0 < k \leqslant n$):

$$f[x_0, x_1, \dots, x_k] = \frac{f[x_1, x_2, \dots, x_k] - f[x_0, x_1, \dots, x_{k-1}]}{x_k - x_0}$$

Wykorzystując te dwie własności łatwo można policzyć wszystkie ilorazy różnicowe $f[x_0]$, $f[x_0, x_1]$, $\dots, f[x_0, x_1, \dots, x_n]$.

Polecenie mówi żeby w celu wykonania zadania nie używać w algorytmie tablicy dwuwymiarowej, a zatem można to zrobić za pomocą następującego algorytmu:

```
function ilorazyRoznicowe(x::Vector{Float64}, f::Vector{Float64})
    n = length(x)
    if length(f) != n
        throw("ilorazyRoznicowe: Wektory muszą być tego samego rozmiaru")
    end

    result = Vector{Float64}(undef, n)
    for i in 1:n
        result[i] = f[i]
    end
    for i in 2:n
        for j in n:-1:i
            result[j] = (result[j] - result[j - 1]) / (x[j] - x[j - i + 1])
        end
    end
    return result
end
```

Na początku (*3*) sprawdzamy czy oba wektory są tej samej długości aby uniknąć ewentualnych błędów. Jeśli wszystko jest w porządku to tworzymy wektor (*7*) z wynikami (tak jak w specyfikacji zadania). Na początku (*8*) wypełniamy go wartościami pojedynczych ilorazów (result[i]= $f[x_{i-1}] = f(x_{i-1})$) Następnie (*11*) korzystając z drugiej z zaprezentowanych wcześniej własności liczymy kolejne ilorazy różnicowe wypełniając wektor poprawnymi wartościami "od przodu". result[1]=$f[x_0]$ jest już poprawnie przypisanym wynikiem. result[2] powinien być równy $f[x_0, x_1]$ zatem liczymy go korzystając z już obliczonych wartości $f[x_0]$ i $f[x_1]$. Sposób obliczania kolejnych ilorazów różnicowych dla wektora długości 4 prezentuje poniższy schemat gdzie każda kolejna kolumna od lewej do prawej prezentuje wektor result[] w kolejnych iteracjach zewnętrznej pętli (*11*) a wiersz (od dołu do góry) to wartości obliczane w wewnętrznej pętli (*12*)

$$\begin{array}{c|ccccccc}
\text{i} & & & & \text{result[i]} & & & \\
\hline
1 & f[x_0] & \searrow & & & & & \\
2 & f[x_1] & \overset{\rightarrow}{\searrow} & f[x_0,x_1] & & & & \\
3 & f[x_2] & \overset{\rightarrow}{\searrow} & f[x_1,x_2] & \overset{\rightarrow}{\searrow} & f[x_0,x_1,x_2] & & \\
4 & f[x_3] & \rightarrow & f[x_2,x_3] & \rightarrow & f[x_1,x_2,x_3] & \rightarrow & f[x_0,x_1,x_2,x_3]
\end{array}$$

W ten sposób otrzymujemy i zwracamy wektor ilorazów różnicowych. Te ilorazy przydadzą nam się również w następnych zadaniach. Ten jak i pozostałe algorytmy testowałem dla kilku funkcji (dla każdej wybierałem 3 węzły równomiernie z przedziału $[a; b]$:

Funkcja			$[a;b]$
$f(x)$	$=$	$(x-1)(x+1)$	$[-1;1]$
$g(x)$	$=$	$(x-1)(x-2)(x-3)(x-4)$	$[1;4]$
$h(x)$	$=$	$\frac{1}{x}$	$[4;16]$
$i(x)$	$=$	$sin(x)$	$[0;2]$

Poniżej prezentuję wygenerowane ilorazy różnicowe ($c_i = f[x_0, \ldots, x_i]$) zaokrąglone do 5 cyfr znaczących(3. kolumna), oraz ich porównanie z wynikiem oczekiwanym (również w zaokrągleniu - 4. kolumna):

	c_0	0	0
	c_1	-1	-1
$f(x)$	c_2	1	1

	c_0	0	0
	c_1	0.375	0.375
$g(x)$	c_2	-0.25	-0.25

	c_0	0.25	0.25
	c_1	-0.025	-0.025
$h(x)$	c_2	0.0015625	0.0015625

	c_0	0	0
	c_1	0.63662	$\frac{2}{\pi} \approx 0.63662$
$i(x)$	c_2	-0.40528	$-\frac{4}{\pi^2} \approx -0.40528$

Jak widać mój algorytm poprawnie liczy ilorazy różnicowe. W kolejnych zadaniach będę testował moje algorytmy dla tych samych funkcji ale innych zakresów, oraz generując 5 węzłów zamiast 3.

Zakresy używane w pozostałych zadaniach:

Funkcja			$[a;b]$
$f(x)$	$=$	$(x-1)(x+1)$	$[-1;1]$
$g(x)$	$=$	$(x-1)(x-2)(x-3)(x-4)$	$[1;4]$
$h(x)$	$=$	$\frac{1}{x}$	$[1;2]$
$i(x)$	$=$	$sin(x)$	$[0;2]$

2 Zadanie 2

Kolejnym zadaniem jest napisanie funkcji 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 jest zdefiniowana następująco:

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

Dane:

x	-	wektor długości $n+1$ zawierający węzły $x_0, \ldots, x_n$ x[1]=$x_0, \ldots,$x[n+1]=x_n
f	-	wektor długości $n+1$ zawierający wartości interpolowanej funkcji w węzłach $f(x_0), \ldots, f(x_n)$
t	-	punkt, w którym należy obliczyć wartość wielomianu

Zwracany wynik:

nt	-	wartość wielomianu interpolacyjnego w punkcie t

Algorytm korzysta z funkcji z poprzedniego zadania aby wyznaczyć wektor ilorazów różnicowych. Następnie korzysta z uogólnionego algorytmu Hornera aby wyznaczyć wartość funkcji w punkcie. Wielomian interpolacyjny w postaci Newtona można zapisać w postaci:

$$\sum_{k=0}^{n} c_k p_k(x)$$

gdzie c_k to iloraz różnicowy $f[x_0, \ldots, x_k]$, a $p_k(x)$ to wielomian $(x - x_0)(x - x_1)\ldots(x - x_{k-1})$ $(p_0(x) = 1)$. Zatem:

$$(1) N_n(x) = c_0 + c_1(x - x_0) + c_2(x - x_0)(x - x_1) + \ldots + c_n(x - x_0)(x - x_1)\ldots(x - x_{n-1})$$

$$(2) N_n(x) = c_0 + (x - x_0)(c_1 + (x - x_1)(c_2 + (x - x_2)(\ldots(c_{n-1} + (x - x_{n-1})c_n))))$$

Korzystając z algorytmu Hornera każdy wielomian $p(z) = a_n x^n + a_{n-1}x^{n-1} + \ldots + a_1 x + a_0$ możemy zapisać w postaci $p(z) = (z - z_0)(b_{n-1}x^{n-1} + \ldots + b_1 x + x_0) + p(z_0)$ i to właśnie robimy wielokrotnie biorąc za z_0 kolejne węzły x_i od x_0 do x_{n-1} przekształcając *(1)* w *(2)*. Za każdym razem reszta z dzielenia przez x_i, czyli $p(x_i)$ wynosi c_i ponieważ iloczyny przy pozostałych współczynnikach się wyzerują.

Podsumowując: niech $w_n(x) = f[x_0, \ldots, x_n]$, $w_k(x) = f[x_0, \ldots, x_k] + (x - x_k)w_{k+1}(x)$ dla $k = n - 1, \ldots, 0$. Wartość wielomianu interpolacyjnego $N_n(x)$ w punkcie t wynosi $N_n(t) = w_0(x)$. Stosując taki rekurencyjny wzór korzystamy pośrednio z algorytmu Hornera do obliczenia wartości funkcji w punkcie. Poniżej znajduje się implementacja tego algorytmu:

```
1  function warNewton(x::Vector{Float64}, fx::Vector{Float64}, t::Float64)
2      n = length(x)
3      w = Vector{Float64}(undef, n)
4      ir = ilorazyRoznicowe(x, fx)
5      w[n] = ir[n]
6      for i = (n - 1):-1:1
7          w[i] = ir[i] + (t - x[i]) * w[i + 1]
8      end
9      return w[1]
10 end
```

Na początku (*3*) inicjalizujemy wektor przechowujący wartości $w_i(x)$ oraz generujemy ilorazy różnicowe (*3*) dla zadanych węzłów. Następnie zaczynamy wypełniać wektor w (*5-8*) (od końca gdyż do obliczenia w_i potrzebujemy x_{i+1} dla $i < n$. Na końcu (*9*) zwracamy wartość $w_0(x)$.

Poniżej - analogicznie jak w zadaniu 1 - znajdują się wyniki przeprowadzonych testów poprawności wykonywanych obliczeń (dla tych samych 4 funkcji). W celu sprawdzenia poprawności wybrałem 1000 równomiernie rozmieszczonych punktów i sprawdziłem maksymalny występujący błąd bezwzględny pomiędzy wielomianem interpolacyjnym a właściwą funkcją (zaokrąglony do 5 cyfr znaczących).

f-cja	Δ_{max}
$f(x)$	$2.2204 \cdot 10^{-16}$
$g(x)$	$1.6342 \cdot 10^{-13}$
$h(x)$	0.00049745
$i(x)$	0.00059353

Jak widać nawet dla tylko 5 węzłów wielomian interpolacyjny jest bardzo bliski wyjściowej funkcji.

3 Zadanie 3

Znając współczynniki wielomianu interpolacyjnego w postaci Newtona $c_0 = f[x_0]$, $c_1 = f[x_0, x_1]$, $\ldots$, $c_n = f[x_0, \ldots, x_n]$ oraz węzły $x_0, x_1, \ldots, x_n$ należy napisać funkcję obliczającą współczynniki jego postaci naturalnej $a_0, \ldots, a_n$ (takie że $p(x) = a_n x^n + \ldots + a_1 x + a_0$). Funkcja ma być zdefiniowana następująco:

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

Dane:

x	-	wektor długości $n+1$ zawierający węzły $x_0, \ldots, x_n$ x[1]=$x_0, \ldots$,x[n+1]=x_n
fx	-	wektor długości $n+1$ zawierający wartości interpolowanej funkcji w węzłach $f(x_0), \ldots, f(x_n)$

Zwracany wynik:

a	-	wektor długości $n+1$ zawierający obliczone współczynniki postaci naturalnej a[1]=a_0, a[2]=a_1, …, a[n]=a_{n-1}, a[n+1]=a_n]

W celu wymyślenia odpowiedniego algorytmu skorzystałem z obserwacji dla wielomianu rzędu 3:

$$N_3(x) = c_0 + c_1(x - x_0) + c_2(x - x_0)(x - x_1) + c_3(x - x_0)(x - x_1)(x - x_2)$$

$$N_3(x) = (c_0 - c_1x_0 + c_2(x_0x_1) - c_3(x_0x_1x_2)) + (c_1 - c_2(x_0 + x_1) + c_3(x_0x_1 + x_0x_2 + x_1x_2))x+$$
$$+(c_2 - c_3(x_0 + x_1 + x_2))x^2 + c_3x^3$$

Zatem:

$$a_0 = c_0 - c_1x_0 + c_2(x_0x_1) - c3(x_0x_1x_2)$$

$$a_1 = c_1 - c_2(x_0 + x_1) + c_3(x_0x_1 + x_0x_2 + x_1x_2)$$

$$a_2 = c_2 - c_3(x_0 + x_1 + x_2)$$

$$a_3 = c_3$$

Zatem w ogólności:

$$a_i = c_i - c_{i+1}z_{i+1} + c_{i+2}z_{i+2} - \ldots + c_nz_n$$

gdzie żeby otrzymać z_{i+j} bierzemy wszystkie j-elementowe podzbiory zbioru $\{x_k : 0 \leqslant k < i\}$. W każdym takim podzbiorze mnożymy przez siebie wszystkie elementy po czym sumujemy iloczyny wszystkich takich zbiorów. To zadanie zrealizowałem przy pomocy poniższego kodu:

```
1  function naturalna(x::Vector{Float64}, fx::Vector{Float64})
2      ir = ilorazyRoznicowe(x, fx)
3      n = length(x)
4      a = Vector{Float64}(undef, n)
5      for k in 1:n
6          a[k] = ir[k]
7          sign = -1
8          for i in (k + 1):n
9              a[k] += sign * ir[i] * sumOfProducts(combinations(x[1:(i - 1)], (i - k)))
10             sign *= -1
11         end
12     end
13     return a
14 end
```

Na początku (*2*) generuje on ilorazy różnicowe na podstawie podanych węzłów. Następnie (*4*) inicjalizuję wektor przechowujący współczynniki postaci naturalnej. Następnie (*5-12*) po kolei wypełniam ten wektor. Funkcja `combinations(X, k)` zwraca zbiór wszystkich k-elementowych podzbiorów zbioru X. Funkcja `sumOfProducts(Z)` sumuje iloczyny wszystkich tych podzbiorów. `sign` to zmienna kontrolująca to że w podanym wcześniej wzorze na a_i na zmianę jest $+$ i $-$.

Poniżej znajdują się wyniki przeprowadzonych testów poprawności wykonywanych obliczeń (porównałem wygenerowane współczynniki z realnymi współczynnikami dwóch testowanych w zadaniu 1 wielomianów). Wygenerowane współczynniki zaokrągliłem do 5 miejsc znaczących. W 3. kolumnie są wygenerowane współczynniki, a w 4. prawdziwe.

	a_0	-1	-1
	a_1	0	0
$f(x)$	a_2	1	1

	a_0	24	24
	a_1	-50	-50
$g(x)$	a_2	35	35
	a_3	-10	-10
	a_4	1	1

Jak widać w obu przypadkach funkcja wygenerowała dokładnie te współczynniki które powinna.

4 Zadanie 4

Wykorzystując funkcje z poprzednich zadań oraz pakiet `Plots` należało zaimplementować funkcję

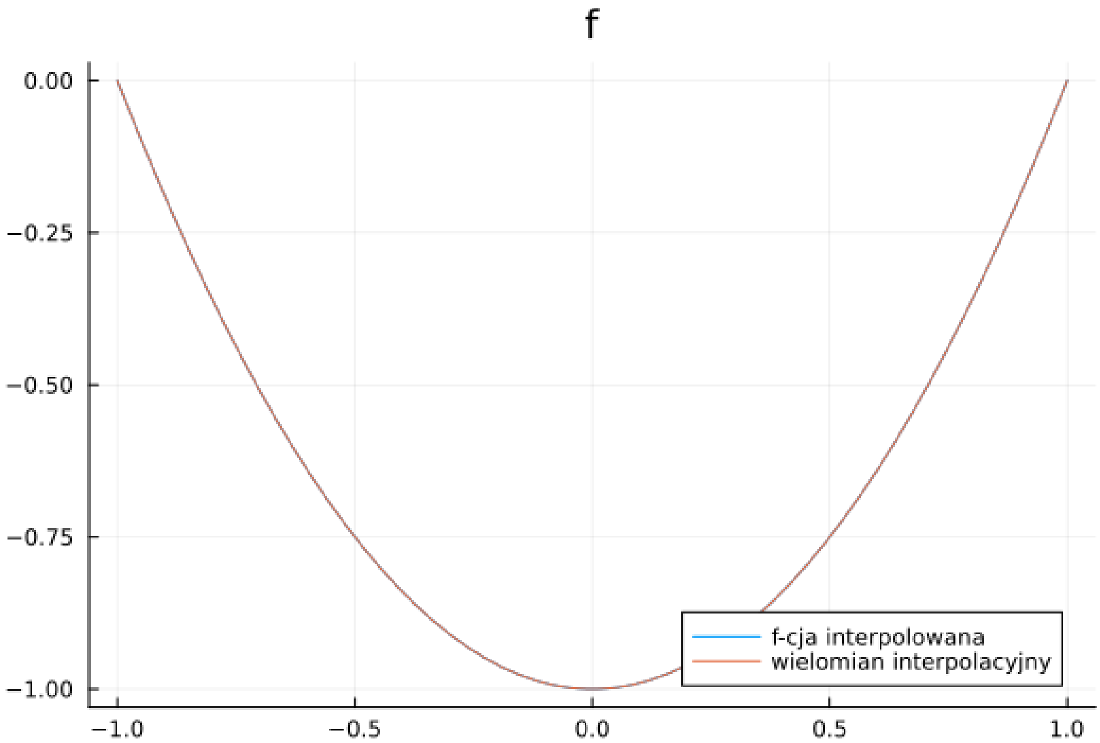
```
function rysujNnfx(f, a::Float64, b::Float64, n::Integer)
```

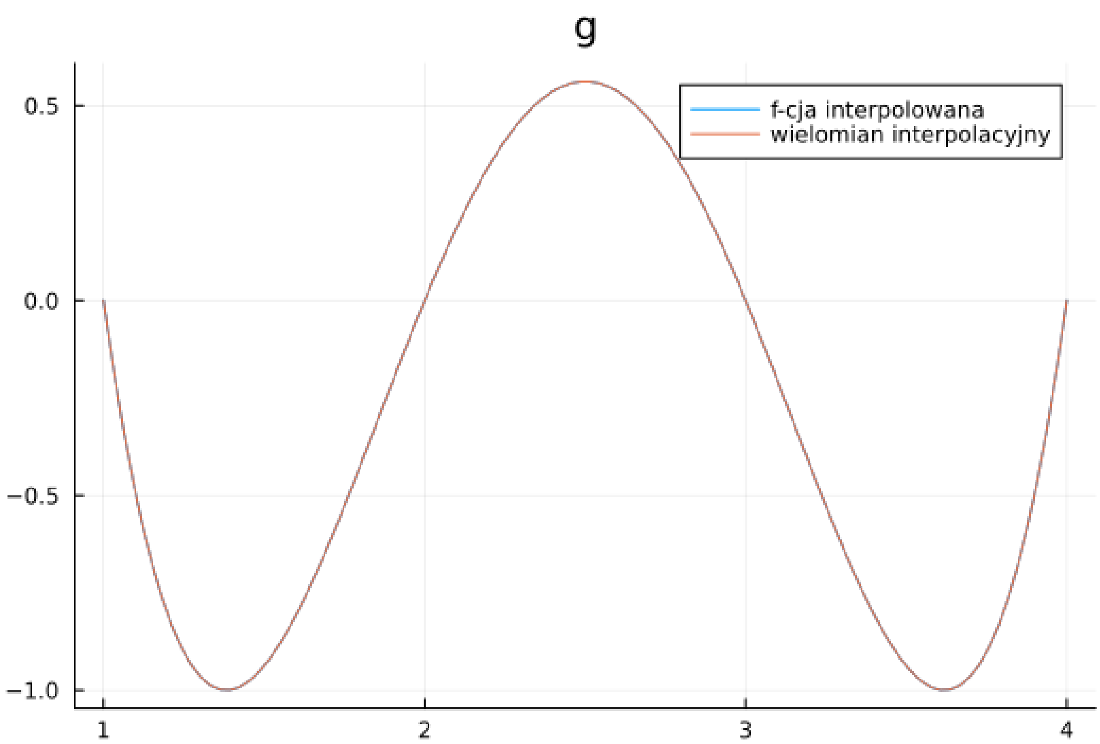
Dane:

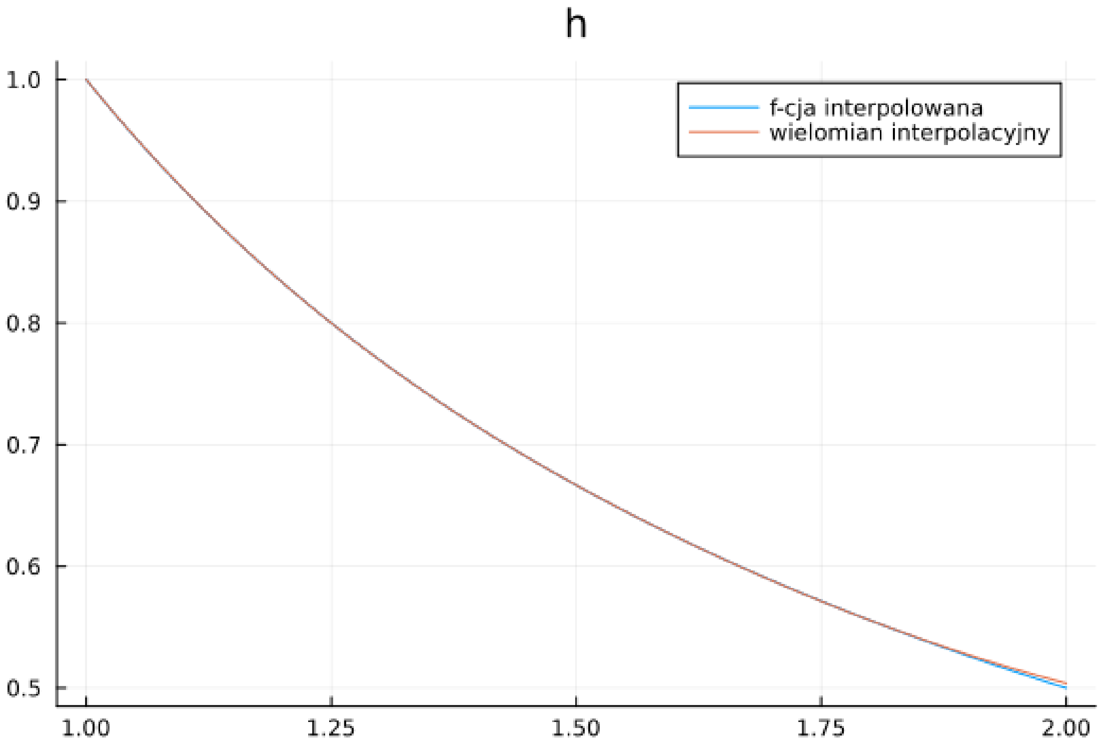
f	-	funkcja $f(x)$ zadana jako anonimowa funkcja
a, b	-	przedział interpolacji
n	-	stopień wielomianu interpolacyjnego

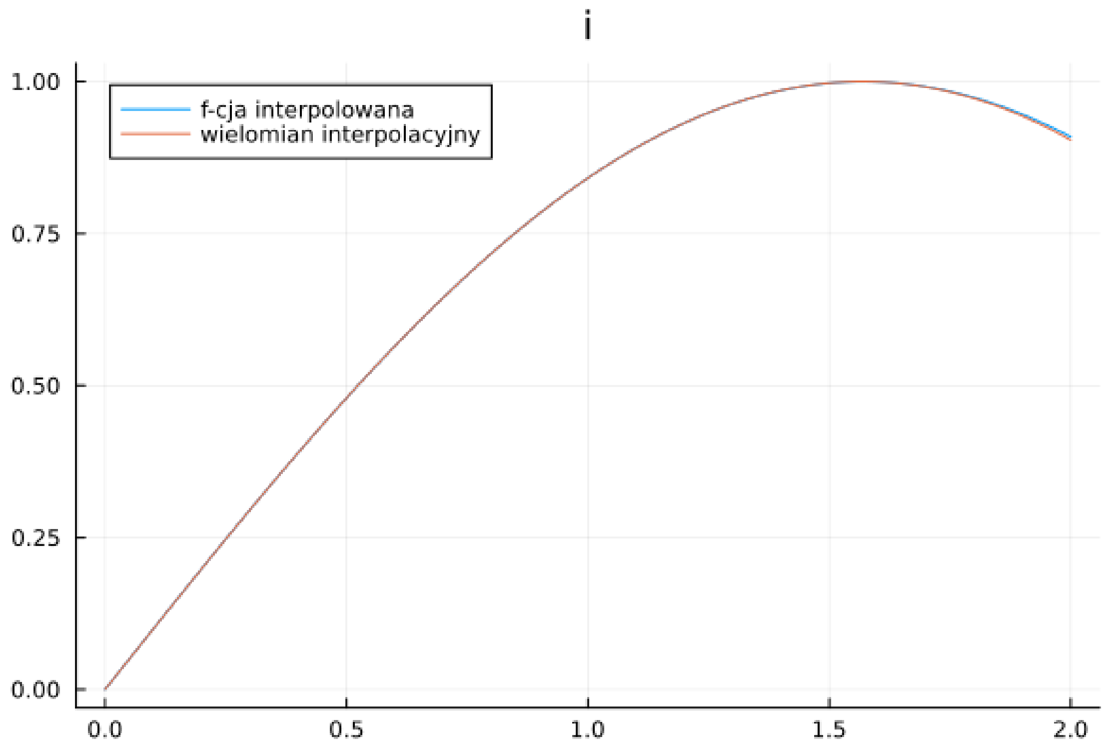
Wynik: funkcja rysuje wielomian interpolacyjny i interpolowaną funkcję w przedziale $[a; b]$

Funkcja na zadanym przedziale równomiernie wybiera $n + 1$ punktów ($x_k = a + (k - 1)\frac{b-a}{n}$ dla $k = 0, 1, \ldots, n$), sprawdza wartość funkcji w tych punktach i na podstawie tych dwóch wektorów generuje wielomian interpolacyjny. Następnie na podstawie 100 równomiernie wybranych punktów z przedziału $[a; b]$ generuje wykres funkcji interpolowanej oraz wykres wielomianu interpolacyjnego na przedziale $[a; b]$. Poniżej przykłady wygenerowanych wykresów dla 4 testowych funkcji z zadania 1:




g
f-cja interpolowana
wielomian interpolacyjny
0.5
0.0
−0.5
−1.0
1
2
3
4


h
f-cja interpolowana
wielomian interpolacyjny
1.0
0.9
0.8
0.7
0.6
0.5
1.00
1.25
1.50
1.75
2.00



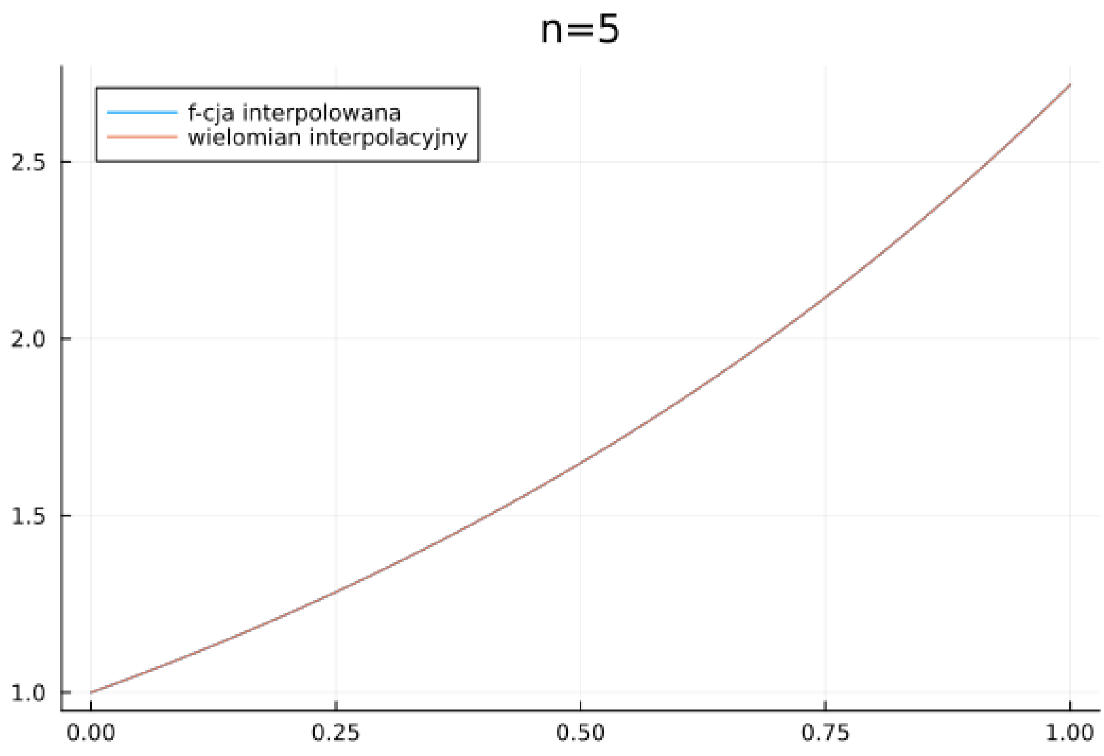
Na wykresach widać że wygenerowane wielomiany są zbieżne z funkcją interpolowaną.

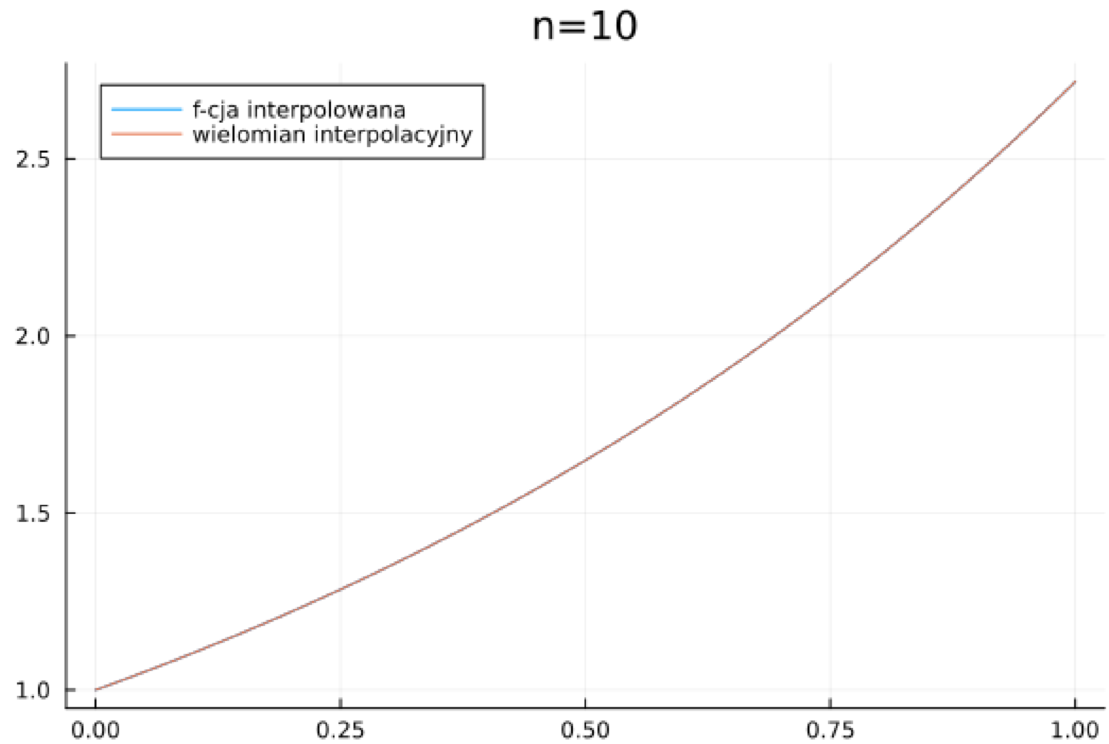
5 Zadanie 5

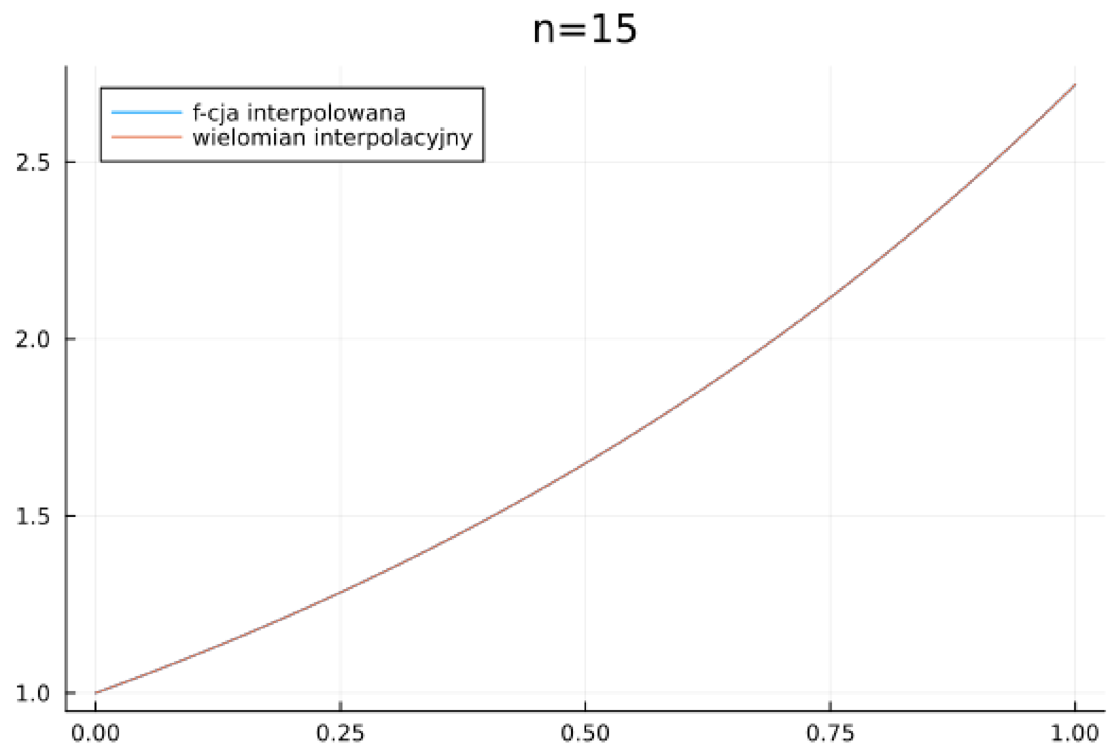
W tym zadaniu mamy podane dwie "porządne" funkcje. "Porządne" czyli takie które w miarę łatwo jest przybliżyć za pomocą wielomianu interpolacyjnego o ile dobrze (równomiernie) dobierze się węzły. Testujemy funkcję `rysujNnfx(f, a, b, n)` dla obu tych funkcji i $n = 5, 10, 15$.

5.1

Niech $f(x) = e^x$, $[a; b] = [0; 1]$ poniżej znajdują się wykresy wygenerowane przez mój program



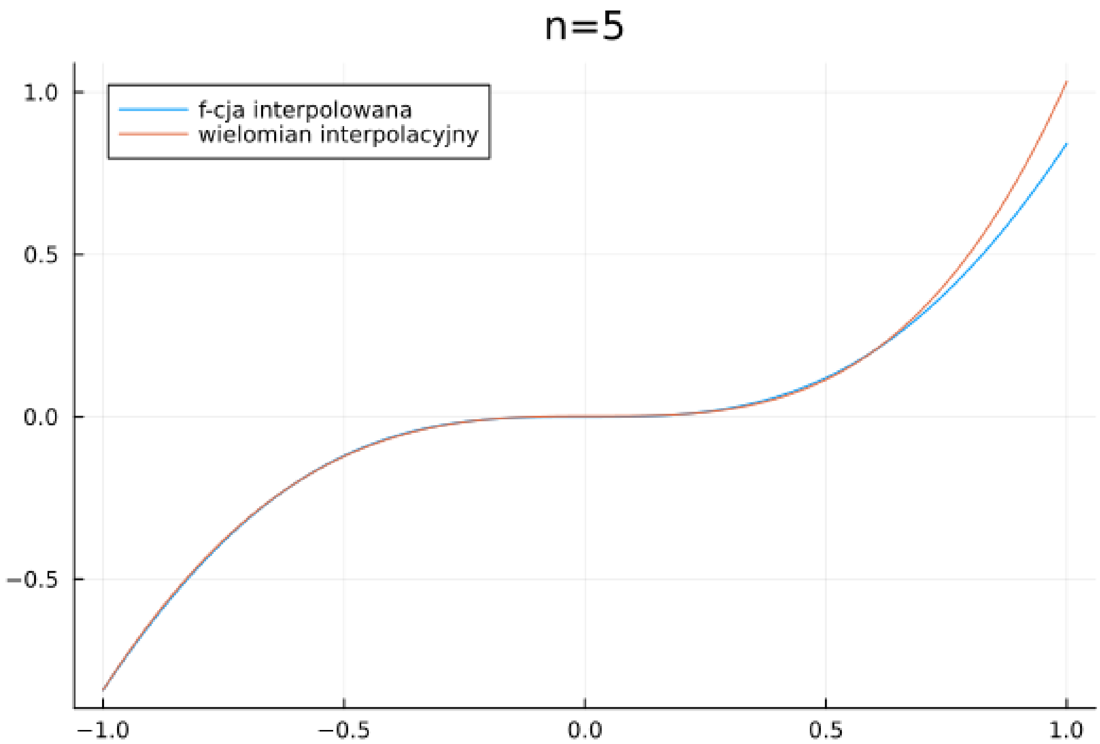


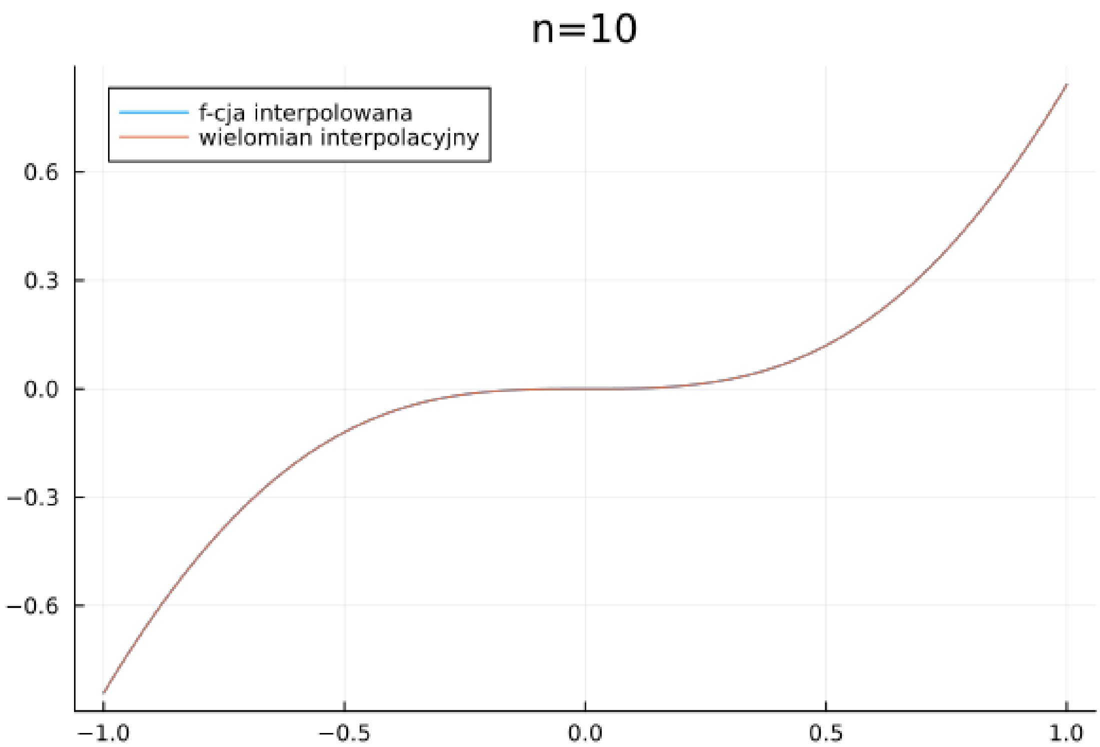


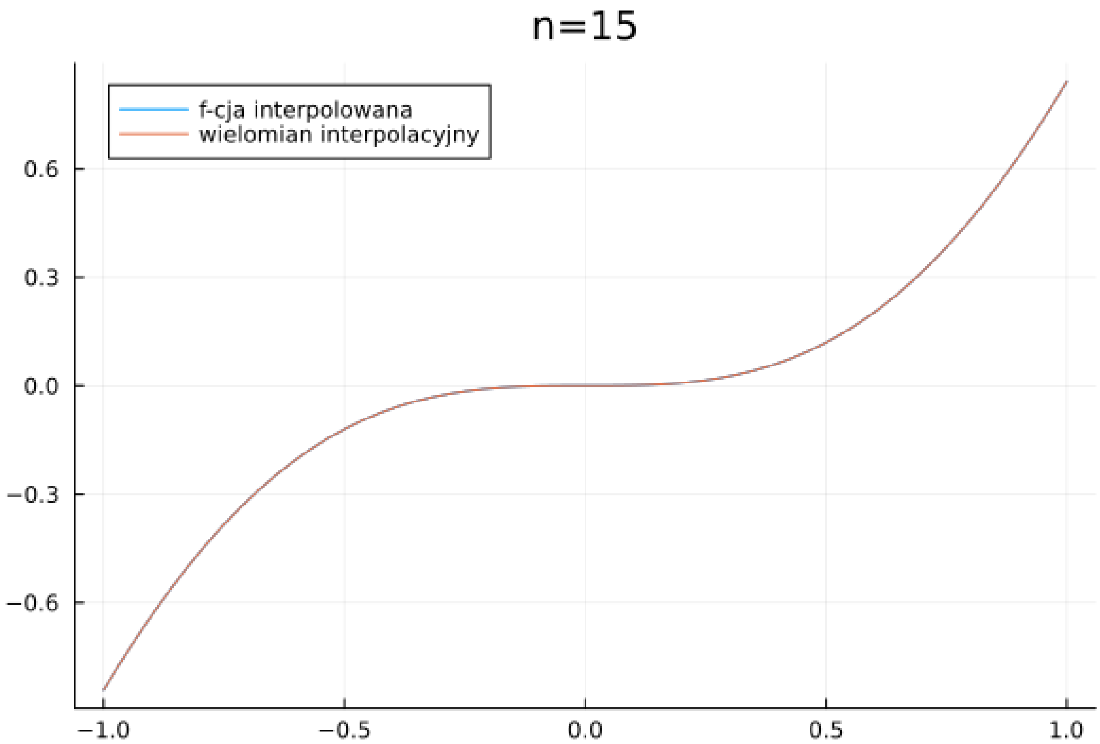
Dla funkcji eksponencjalnej nawet dla $n = 5$ nie widać różnicy między wielomianem interpolacyjnym a funkcją interpolowaną.

5.2

Niech $f(x) = x^2 sin(x)$, $[a; b] = [-1; 1]$ poniżej znajdują się wykresy wygenerowane przez mój program


n=5
f-cja interpolowana
wielomian interpolacyjny
1.0
0.5
0.0
−0.5
−1.0
−0.5
0.0
0.5
1.0


n=10
f-cja interpolowana
wielomian interpolacyjny
0.6
0.3
0.0
−0.3
−0.6
−1.0
−0.5
0.0
0.5
1.0



Dla $n = 5$ widać pewne rozbieżności między zadaną funkcją a wygenerowanym wielomianem interpolacyjnym - zwłaszcza przy prawej granicy przedziału. Jednak dla $n = 10$ i $n = 15$ gołym okiem praktycznie nie widać już różnicy między tymi dwoma funkcjami na tym przedziale.

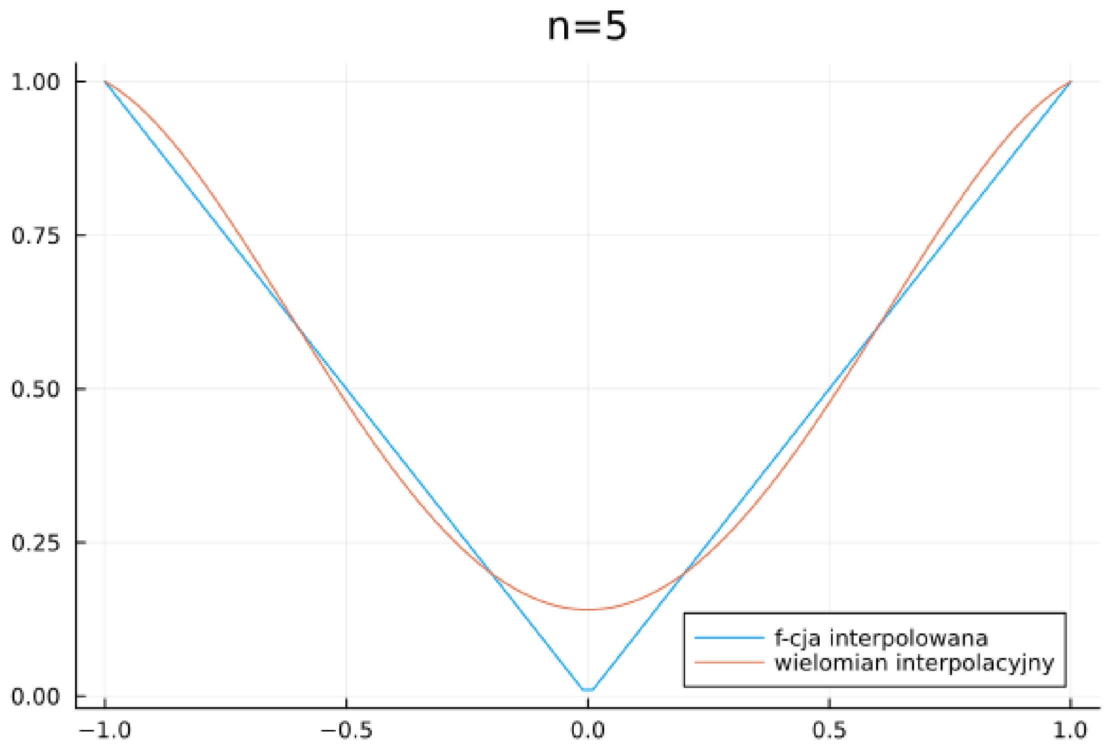
Wniosek jest następujący: obie te funkcje są dobrze uwarunkowane i praktycznie równoważnie z nimi można się posługiwać ich interpolacjami bez znacznej utraty precyzji.

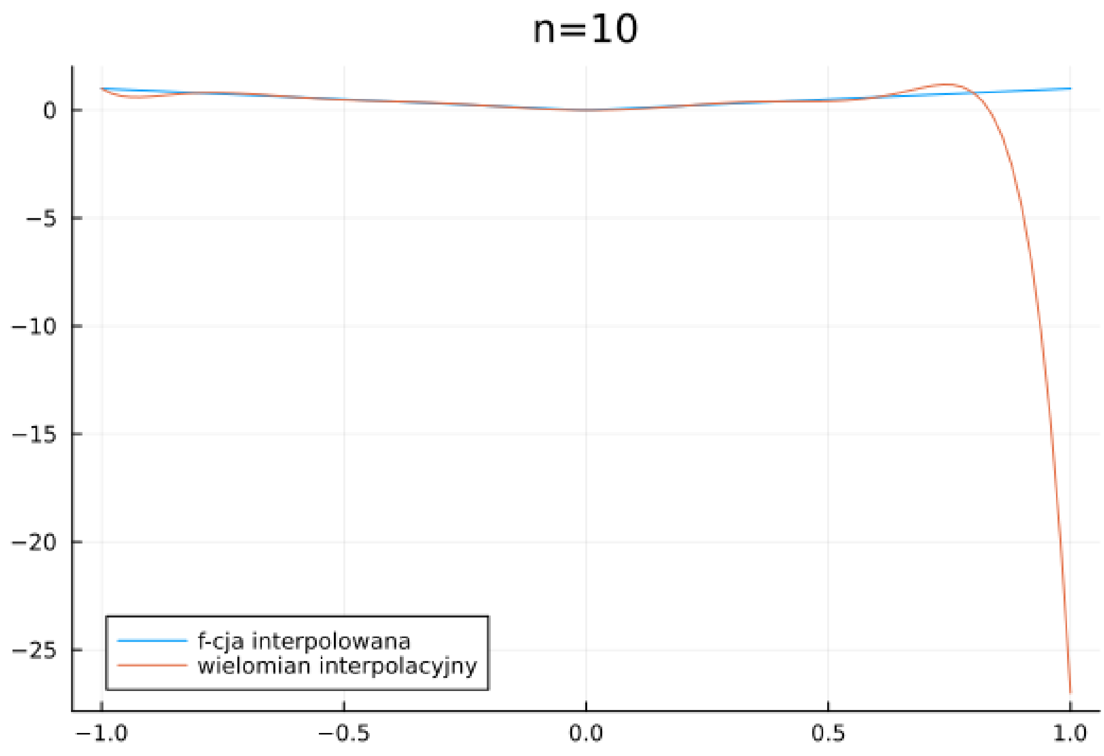
6 Zadanie 6

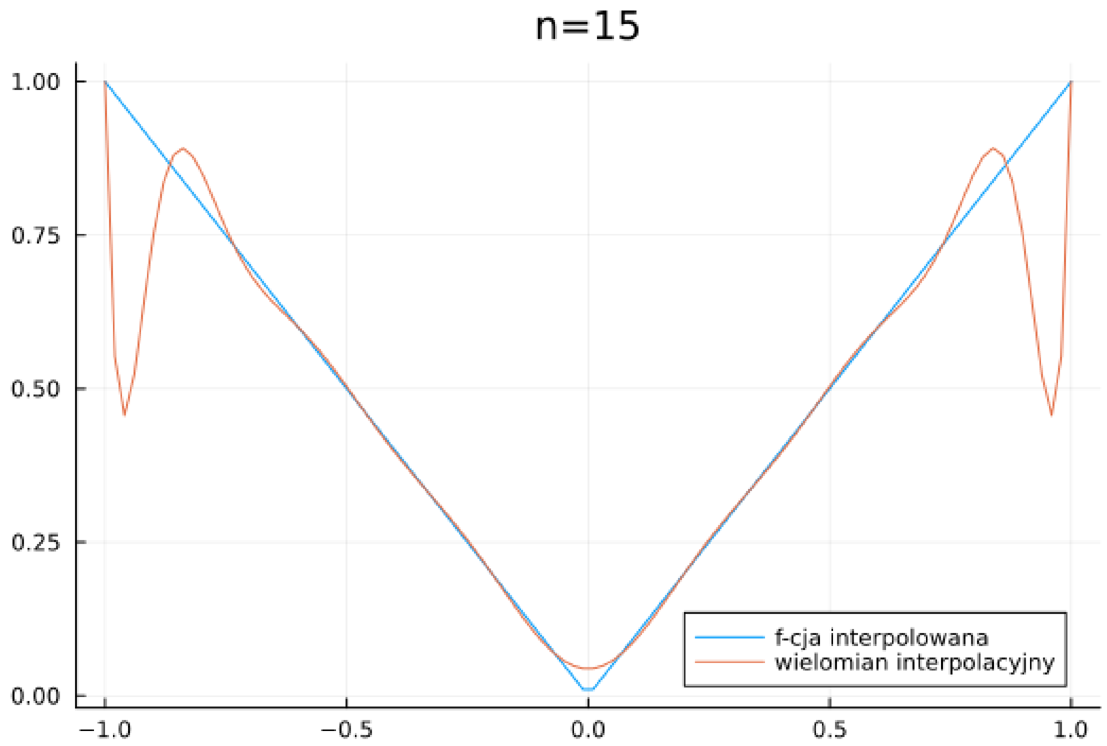
Ostatnie zadanie jest analogiczne do poprzedniego tylko tym razem testujemy nasz program dla dużo "gorszych" funkcji.

6.1

Niech $f(x) = |x|$, $[a; b] = [-1; 1]$ poniżej znajdują się wykresy wygenerowane przez mój program


n=5
1.00
0.75
0.50
0.25
0.00
−1.0
−0.5
0.0
0.5
1.0
f-cja interpolowana
wielomian interpolacyjny

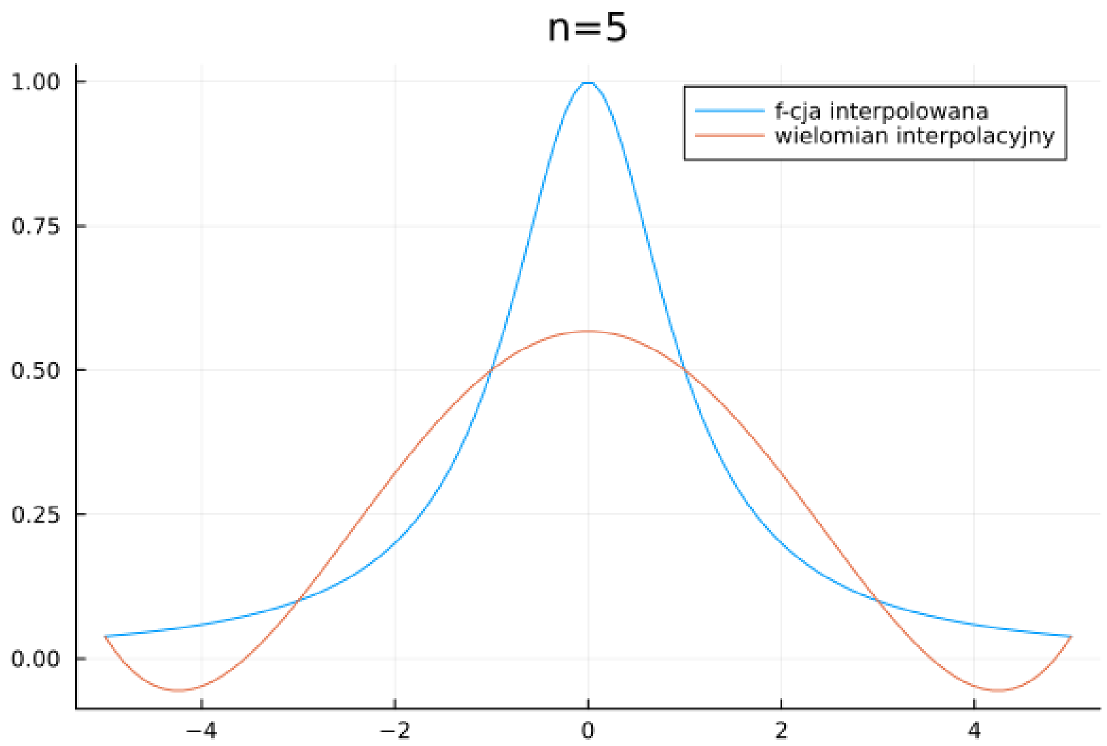

n=10
0
−5
−10
−15
−20
−25
−1.0
−0.5
0.0
0.5
1.0
f-cja interpolowana
wielomian interpolacyjny

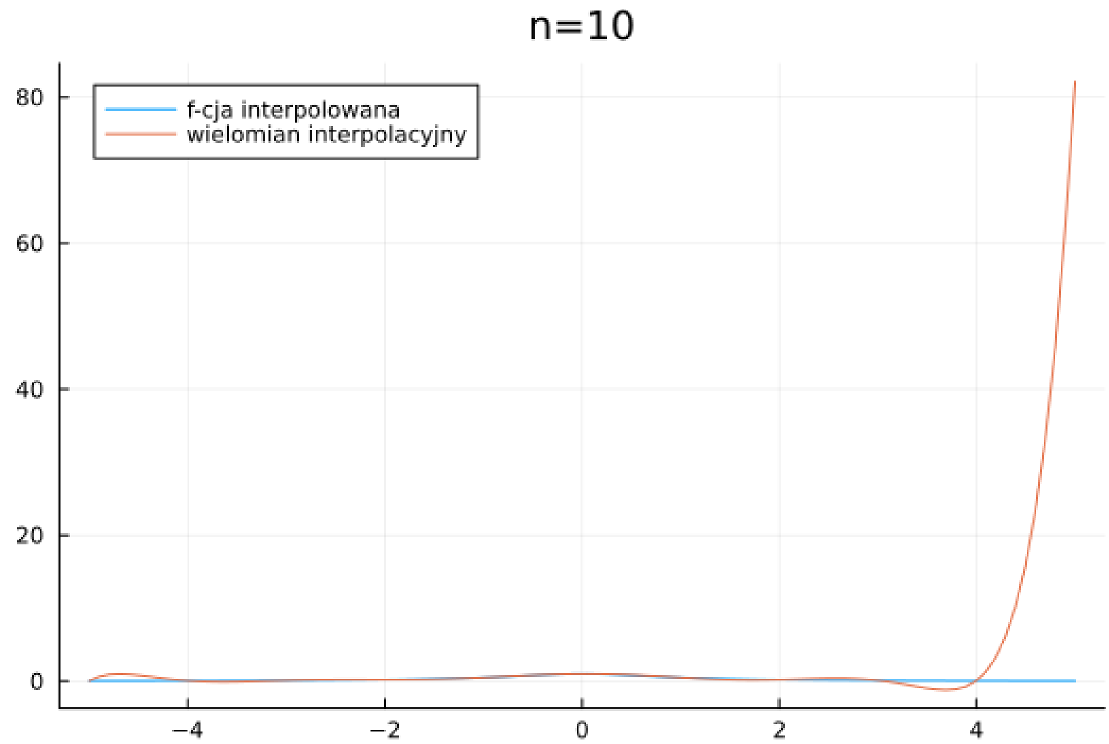


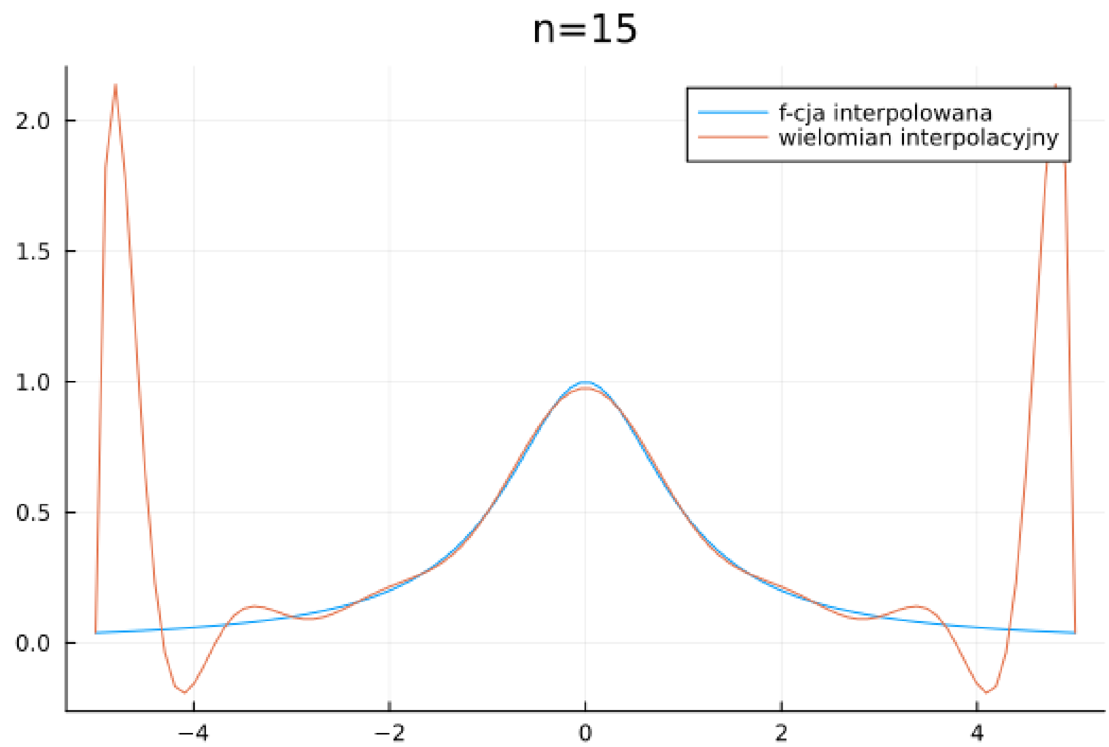
Dla $n = 5$ funkcja na zadanym przedziale jest całkiem dobrze aproksymowana. Dla $n = 10$ na pierwszy rzut oka wydaje się że w programie wystąpił jakiś błąd ale taki jest wynik interpolacji na przedziale $[-1; 1]$ - na przedziale $[-1; 0.75]$ funkcja jest aproksymowana całkiem dobrze, jednak przy prawej krawędzi przedziału funkcja jest bardzo ostro malejąca i dość mocno rozbiega od funkcji interpolowanej. Dla $n = 15$ znowu aproksymacja jest w miarę dobra oprócz obu końców przedziału gdzie występują spore oscylacje.

6.2

Niech $f(x) = \frac{1}{1+x^2}$, $[a; b] = [-5; 5]$ poniżej znajdują się wykresy wygenerowane przez mój program







Dla $n = 5$ mimo sporych rozbieżności między dwoma wykresami funkcja jest aproksymowana całkiem dobrze - ma przynajmniej podobny kształt co $f(x)$. Nie można tego powiedzieć o funkcji narysowanej dla $n = 10$. Tutaj - podobnie jak w poprzednim przykładzie - funkcja jest aproksymowana w porządku mniej więcej na przedziale $[-5; 4]$ Jednak przy prawym końcu zadanego przedziału funkcja drastycznie rośnie daleko odbiegając od poprawnych wartości. Dla $n = 15$ funkcja znowu jest przybliżona całkiem dobrze pomijając duże oscylacje na obu krańcach przedziału.

W obu powyższych funkcjach i ich interpolacjach można zauważyć tzw. zjawisko Runge'go. Polega ono na tym że mimo wzrostu liczby węzłów - co intuicyjnie powinno poskutkować większą precyzją interpolacji - dokładność wielomianu interpolacyjnego pogarsza się względem tych wygenerowanych dla mniejszych n.