Sprawozdanie

Obliczenia naukowe

Lista 4

Piotr Gałwiaczek

Numer indeksu: 221495

1. **Zadanie 1** – funkcja obliczająca ilorazy różnicowe

1.1. Opis problemu

W zadaniu tym mamy stworzyć funkcję obliczającą ilorazy różnicowe, oraz przetestować jej działanie.

1.2. Rozwiązanie

Do rozwiązania tego zadania została utworzona poniższa funkcja:

```
ilorazyRoznicowe(x, f)
        n = length(x)
        IF  n ≠length(f)
                Zakończ i zwróć komunikat o błędzie
        d - tablica, która zawierać będzie obliczone ilorazy różnicowe
        FOR i=1 TO n
                d[i] = f[i]
        FOR  j=2 TO n
                FOR i=n TO j DOWNTO
                        d[i] = (d[i]−d[i−1]) / (x[i]−x[i−j+1])
        return d
```

1.3. Wyniki oraz ich interpretacja

Funkcja przetestowana została na następujących danych:

x	3	1	5	6
$\mathbf{f(x)}$	1	-3	2	4

Poprawne rozwiązanie jest następujące:

$$
\begin{array}{cc|cccc}
3 & \boxed{1} & \boxed{2} & \boxed{-\frac{3}{8}} & \boxed{\frac{7}{40}} \\
1 & -3 & \frac{5}{4} & \frac{3}{20} & \\
5 & 2 & 2 & & \\
6 & 4 & & &
\end{array}
$$

Zaś nasza funkcja zwróciła wartości:

1.0	2.0	-0.375	0.17500000000000002

1.4. Wnioski

Funkcja zwróciła poprawne rozwiązanie. Została ona zaimplementowana bez użycia tablicy dwuwymiarowej dzięki czemu oszczędzamy przy tym pamięć.

2. **Zadanie 2** – uogólniony algorytm Hornera

2.1. Opis problemu

W zadaniu tym należy stworzyć funkcję obliczającą wartość wielomianu interpolacyjnego stopnia n w postaci Newtona w punkcie za pomocą uogólnionego algorytmu Hornera, oraz następnie przetestować jego działanie.

2.2. Rozwiązanie

Utworzyliśmy następującą funkcję:

```
warNewton(x, fx, t)
        n – długość tablicy x
        w[n] = fx[n]
        FOR i=n-1 TO 1 DOWNTO
                w[i] = fx[i]+(t-x[i])*w[i+1]
        return w[1]
```

2.3. Wyniki oraz ich interpretacja

Funkcja przetestowana została na tych samych danych których użyliśmy w zadaniu 1. Obliczyliśmy wartość tego wielomianu w punkcie $t = \frac{1}{2}$.

Wiemy, że funkcja w zadaniu poprzednim utworzyła wielomian postaci:

$$p(x) = 1 + 2(x-3) - \frac{3}{8}(x-3)(x-1) + \frac{7}{40}(x-3)(x-1)(x-5)$$

więc poprawnym rozwiązaniem powinna być wartość:

$$p\left(\frac{1}{2}\right) = 1 + 2\left(\frac{1}{2}-3\right) - \frac{3}{8}\left(\frac{1}{2}-3\right)\left(\frac{1}{2}-1\right) + \frac{7}{40}\left(\frac{1}{2}-3\right)\left(\frac{1}{2}-1\right)\left(\frac{1}{2}-5\right) = -5.453125$$

Nasza funkcja zwróciła właśnie taką wartość.

2.4. Wnioski

Funkcja zwróciła poprawne rozwiązanie. Dzięki zastosowaniu uogólnionego schematu Hornera, czas działania naszego algorytmu skraca się do $O(n)$.

3. **Zadanie 3 –** Funkcja obliczająca w czasie O(n^2)

3.1. Opis problemu

W zadaniu należało, znając współczynniki wielomianu interpolacyjnego Newtona oraz węzły, napisać funkcję obliczającą w czasie $O(n^2)$ współczynniki jego postaci naturalnej.

3.2. Rozwiązanie

W celu znalezienia współczynników dla naturalnej postaci wielomianu interpolacyjnego Newtona wykorzystano uogólnione wzory Hornera przedstawione w poprzednim zadaniu. Na początku korzystamy z faktu, że $a_n = c_n$. Następnie wyliczamy kolejne wartości częściowe dla wielomianu interpolacyjnego, ale podczas każdej iteracji każdy "składowy" wielomian doprowadzamy do postaci i w ten sposób otrzymujemy kolejne współczynniki dla postaci naturalnej wielomianu interpolacyjnego.

```
function NATURALNA(x, fx)

        a – tablica długości len z wyliczającymi współczynnikami
        x – wektor długości n+1 zawierający węzły X0, … ,  Xn.
        fx – wektor długości n+1 zawierający obliczone ilorazy różnicowe

        len ← LENGTH(x)
        a[len] ← fx[len]
        for i ← len -1 down to 1 do
            a[i] =  fx[i] – a[i+1] * x[i]
            for j ← i + 1 to len – 1 do
                a[j]← a[j] – a[j+1] * x[i]
            end for
        end for
        return a
end function
```

3.3. Wyniki oraz ich interpretacja

Na podstawie obserwacji pętli użytych w algorytmie oraz tego, ile razy i w jakiej kolejności każda z nich się wykona otrzymujemy złożoność przedstawionego algorytmu rzędu właśnie O(n^2).

4. **Zadanie 4** – interpolacja funkcji

4.1. Opis problemu

W zadaniu należy utworzyć funkcję która zinterpoluje podaną funkcję $f(x)$ w przedziale $[a, b]$ za pomocą wielomianu intepolacyjnego stopnia n w postaci Newtona.

4.2. Rozwiązanie

W celu rozwiązania tego zadania utworzyliśmy następującą funkcję:

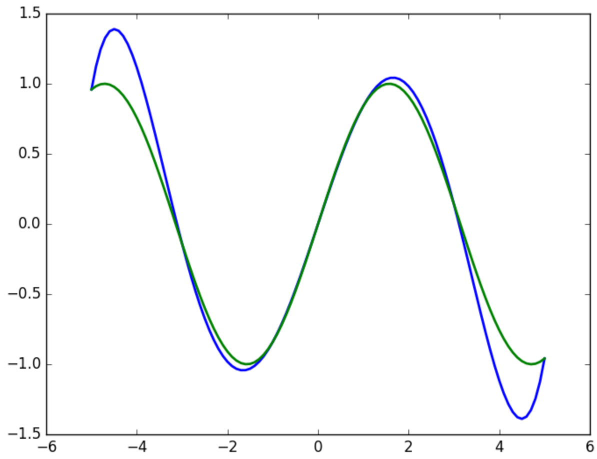
```
rysujNnfx(f, a, b, n)
        h = (b−a)/n
        FOR i=1 TO n+1
                x[i] = a+(i-1)*h
                y[i] = f(x[i])
        h = (b−a)/100
        il_roz = ilorazyRoznicowe(x, y)
        FOR i=1 TO n+1
                x2[i] = a+(i-1)*h
                y2[i] = warNewton(x, il_roz, x2[i])
                y3[i] = f(x2[i])
        rysuj wykresy dla funkcji:
                - z argumentami x2 i wartościami y2
                -z argumentami x2 i wartościami y3
```

4.3. Wyniki oraz ich interpretacja

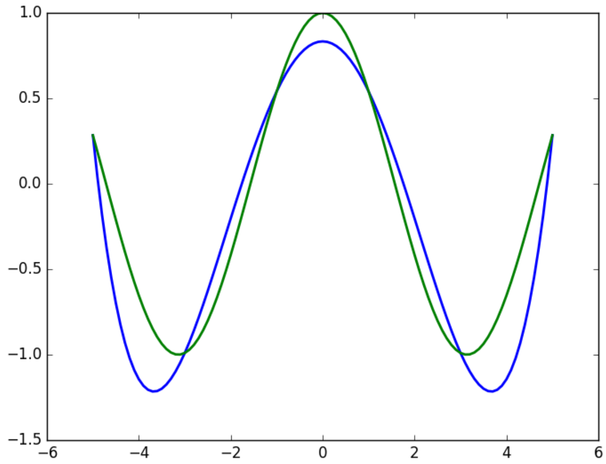
Funkcja przetestowana została na następujących danych:

$funkcja$	a	b	n
sin(x)	-5.0	5.0	5
cos(x)	-5.0	5.0	5

oraz zwróciła następujące ich wykresy:



Rysunek 1 Funkcja sin(x) i jej zinterpolowany wielomian



Rysunek 2 Funkcja cos(x) i jej zinterpolowany wielomian

zinterpolowany wielomian
prawdziwa funkcja

4.4. Wnioski

Jak widzimy funkcja zwróciła zbliżone wykresy. Możemy w tym miejscu zaobserwować jak wielkie możliwości daje nam interpolacja. Dzięki niej jesteśmy w stanie zaobserwować zachowanie funkcji na określonym przedziale, znając jedynie skończoną liczbę danych. Możemy również używać jej w celu uproszczenia skomplikowanych funkcji, np. podczas całkowania.

5. **Zadanie 5** – interpolacja wielomianu na przykładach funkcji e^x, oraz $x^2 \sin x$

5.1. Opis problemu

W zadaniu tym mamy przetestować funkcję z poprzedniego zadania na danych:

1) $e^x, [0,1], \; n = 5, 10, 15$
2) $x^2 \sin x\, , [-1, 1], \; n = 5, 10, 15$

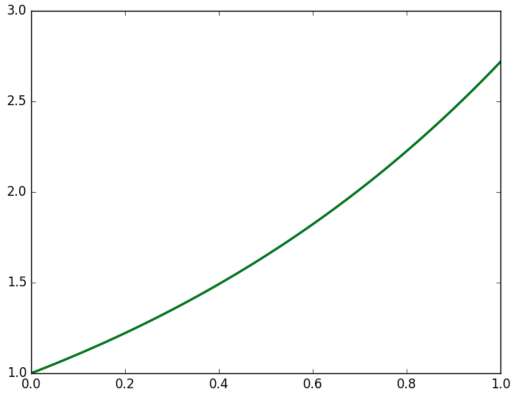
5.2. Rozwiązanie

Aby rozwiązać dane zadnie uruchomiliśmy wcześniej utworzoną funkcję na podanych danych.

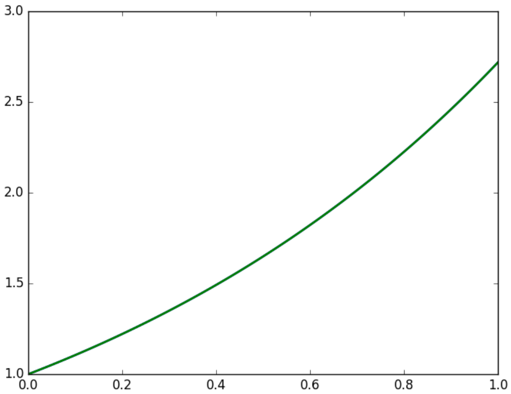
5.3. Wyniki oraz ich interpretacja

Po uruchomieniu funkcji **rysujNnfx** zwróciła ona następujące wykresy:

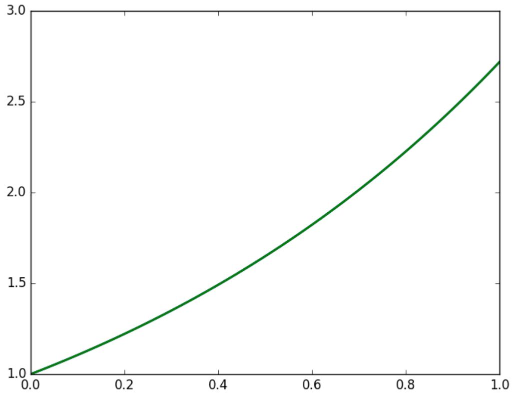
e^x



Rysunek 3 Interpolacja dla n=5



Rysunek 4 Interpolacja dla n=10

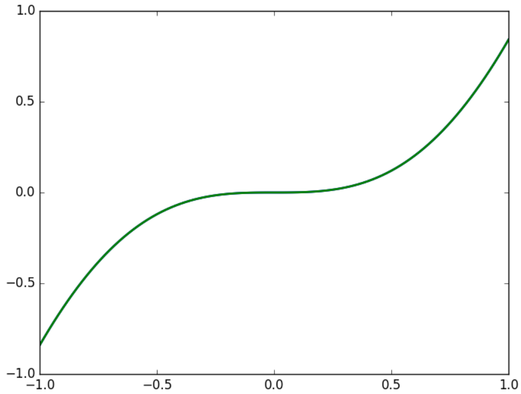


Rysunek 5 Interpolacja dla n=5

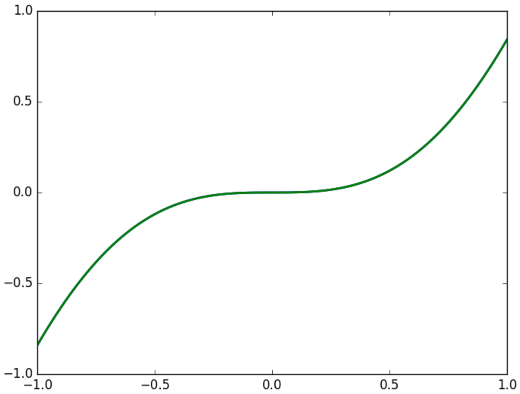
zinterpolowany wielomian

prawdziwa funkcja

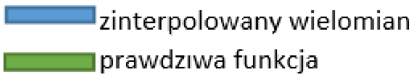
$x^2 \sin x$

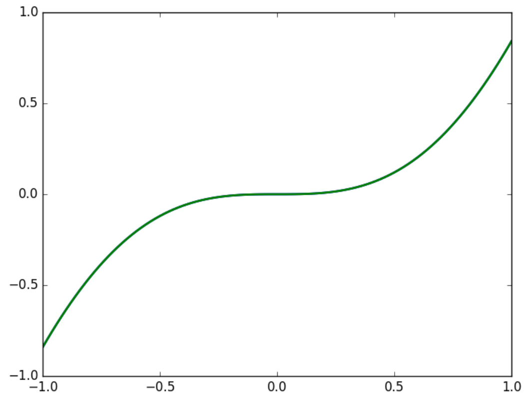


Rysunek 6 Interpolacja dla n=5



Rysunek 7 Interpolacja dla n=10





Rysunek 8 Interpolacja dla n=15

5.4. Wnioski

Jak możemy zaobserwować funkcja dla wskazanych danych zwraca niesamowicie dokładne wyniki. Wykresy zinterpolowanych wielomianów niemal całkowicie pokrywają się z prawdziwymi wykresami podanych funkcji. Nie zmienia tego również zwiększanie stopnia wielomianu.

6. **Zadanie 6** – interpolacja wielomianu na przykładach funkcji |x|, oraz $\frac{1}{1+x^2}$

6.1. Opis problemu

W zadaniu tym również mamy przetestować funkcję **rysujNnfx**na danych:

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

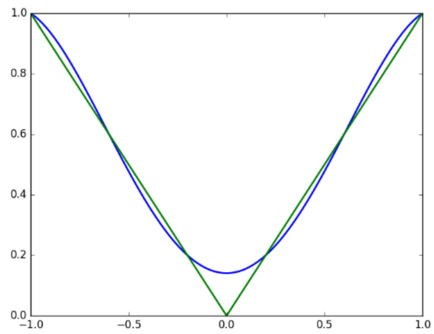
6.2. Rozwiązanie

Podobnie jak w zadaniu 4 uruchomiliśmy funkcję na podanych danych.

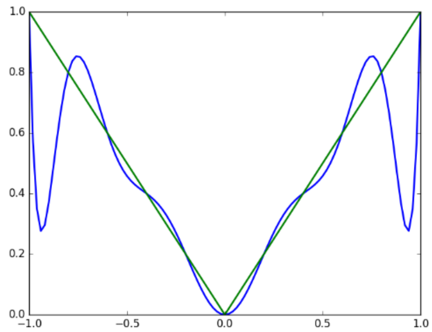
6.3. Wyniki oraz ich interpretacja

Funkcja **rysujNnfx**zwróciła następujące wykresy:

$$|x|$$



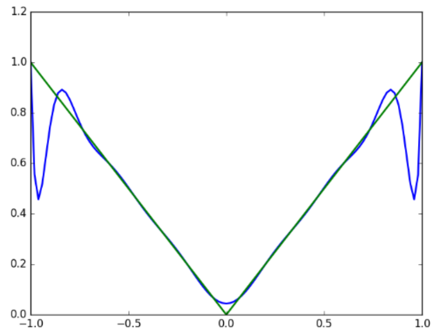
Rysunek 9 Interpolacja dla n=5



Rysunek 10 Interpolacja n=10

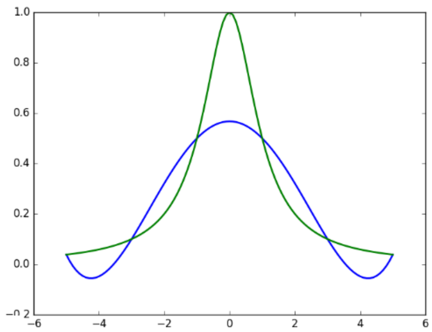
zinterpolowany wielomian

prawdziwa funkcja

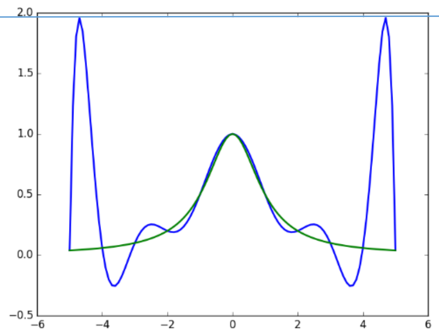


Rysunek 11 Interpolacja dla n=15

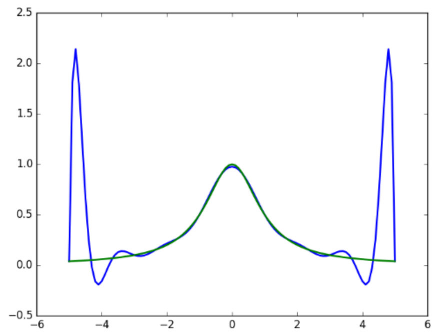
$$\frac{1}{1+x^2}$$



Rysunek 12 Interpolacja dla n=5



Rysunek 13 Interpolacja dla n=10



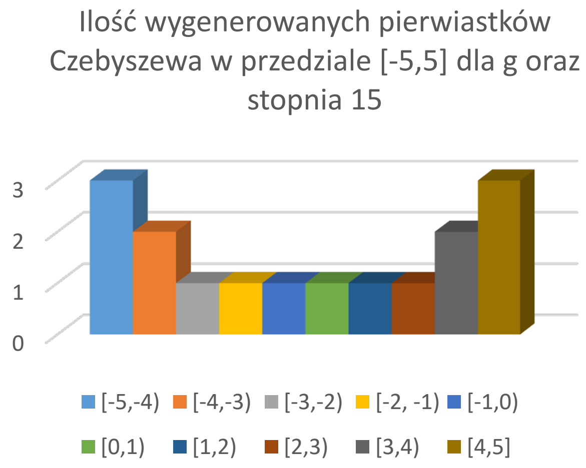
Rysunek 14 Interpolacja dla n=15

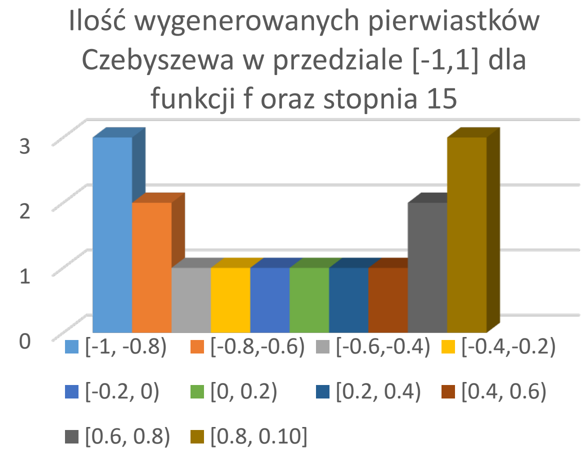
6.4. Wnioski

Początkowo, dla $n = 5$, przybliżenie jest satysfakcjonujące, lecz jak widzimy, funkcja dla coraz większej ilości węzłów zwraca wykresy, na których końcach przedziałów przyjmowane są niepożądane, zaskakujące nas wartości. Dla tych funkcji rozpatrywanych w danych przedziałach ciąg wielomianów interpolacyjnych z węzłami rozmieszczonymi tam w równych odstępach nie dąży do f. Okazuje się, że ma to związek ze **zjawiskiem Rugne'go**, które mówi, że w przypadku węzłów równoodległych, wraz ze wzrostem liczby węzłów na obrzeżach przedziału w którym przeprowadzamy interpolację pojawiają się znaczące różnice pomiędzy wartością wielomianu interpolacyjnego, a wartością żądaną.

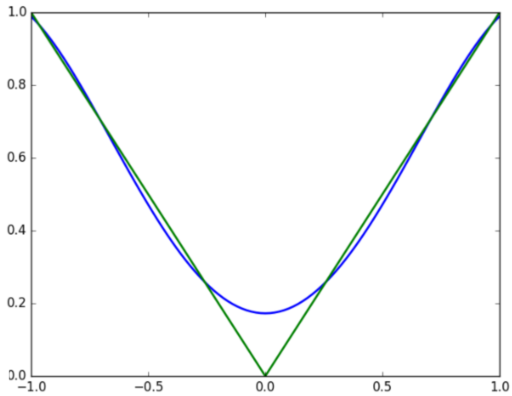
Można poradzić sobie z tym problemem biorąc do interpolacji inne węzły, tj. **węzły Czebyszewa**.

Pierwiastki Czebyszewa dla n = 15	
$\lvert x \rvert$	$\dfrac{1}{1+x^2}$
0.9951847266721969	4.975923633360985
0.9569403357322088	4.784701678661044
0.881921264348355	4.409606321741776
0.773010453362737	3.865052266813685
0.6343932841636455	3.1719664208182277
0.4713967368259978	2.356983684129989
0.29028467725446233	1.4514233862723116
0.09801714032956077	0.49008570164780385
-0.09801714032956065	-0.49008570164780324
-0.29028467725446216	-1.4514233862723107
-0.4713967368259977	-2.3569836841299887
-0.6343932841636454	-3.1719664208182268
-0.773010453362737	-3.865052266813685
-0.8819212643483549	-4.409606321741775
-0.9569403357322088	-4.784701678661044
-0.9951847266721968	-4.975923633360984

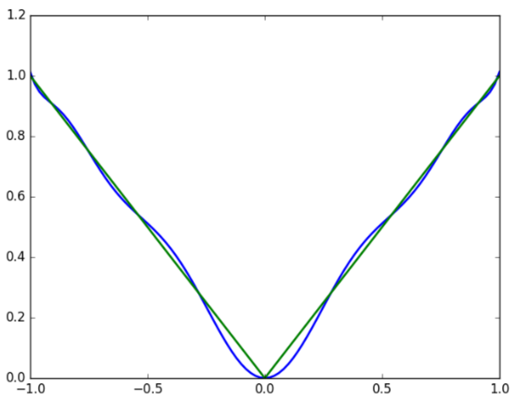




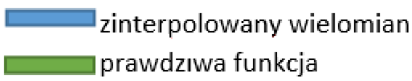
Jak możemy zaobserwować wygenerowane pierwiastki dla obu funkcji w większości skumulowane są na końcach przedziałów. Używając w interpolacji węzłów Czebyszewa funkcja wygenerowała następujące wykresy:

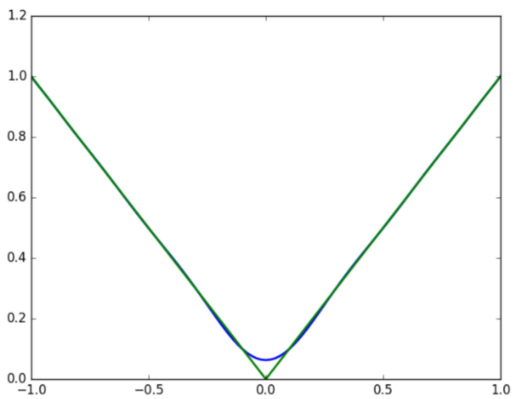


Rysunek 15 Interpolacja dla n=5



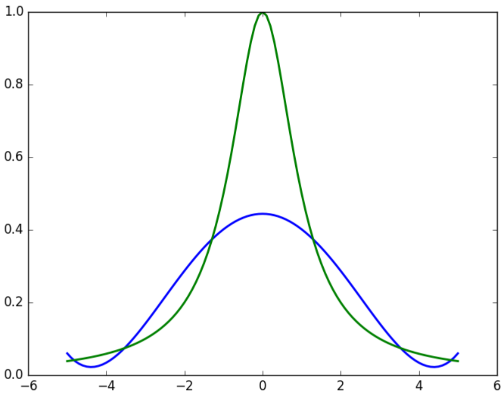
Rysunek 16 Interpolacja dla n=10



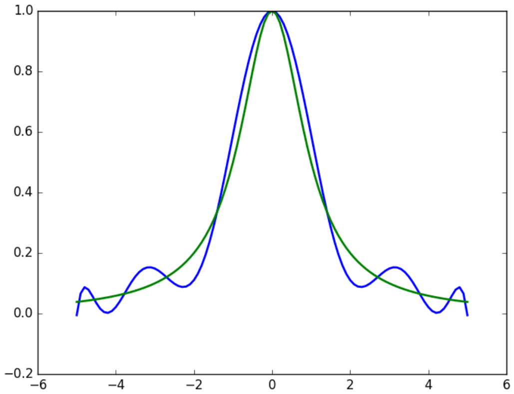


Rysunek 17 Interpolacja dla n=15

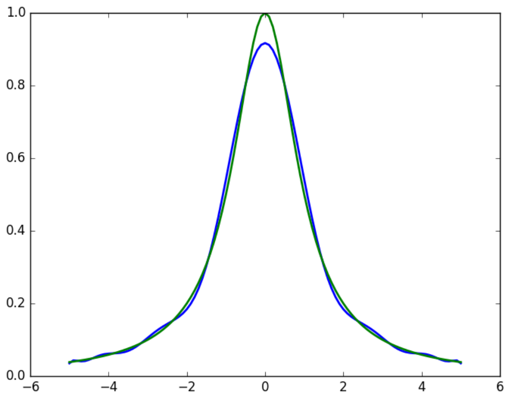
$$\frac{1}{1+x^2}$$



Rysunek 18 Interpolacja dla n=5



Rysunek 19 Interpolacja dla n=10



Rysunek 20 Interpolacja dla n=15

Jak widzimy w przypadku użycia węzłów Czebyszewa utworzone wielomiany są w dużo lepszej postaci. Tak prosta zamiana danych może w dużym stopniu poprawić wydajność naszej funkcji. Dzięki temu udało nam się uniknąć efektu Rugne'go.

zinterpolowany wielomian
prawdziwa funkcja