Sprawozdanie

P2.16 z analizy numerycznej

obliczanie i badanie wielomianu $s_n(x) = {\sum_{k=0}^{n}}' c_k T_k(x)$

Mateusz Markiewicz

16 grudnia 2018

Spis treści

1 Wstęp

1.1 Cel zadania

Celem zadania jest skonstruowanie szybkiego oraz numerycznie poprawnego algorytmu obliczającego wartość wielomianu s_n w punkcie x, gdzie
$s_n(x) := \sum_{k=0}^{\prime n} c_k T_k(x)$, a $T_k(x)$ oznacza k-ty wielomian Czebyszewa.

1.2 Streszczenie sprawozdania

Po wstępie teoretycznym do omawianego zagadnienia przedstawię algorytm Clenshowa oraz jego wersję dla wielomianów Czebyszewa oraz pokażę, jak można wykorzystać go do wyznaczenia wartości sumy $s_n(x) = \sum_{k=0}^{\prime n} c_k T_k(x)$. Następie pokażę przykłady dokładności oraz numerycznej poprawności zaproponowanego rozwiązania.

2 Opis teoretyczny problemu

2.1 Wprowadzenie do zagadnienia

Dowolny wielomian $w_n(x)$ możemy przedstawić w bazie wielomianów Czebyszewa, czyli jako kombinację liniową w postaci $\sum_{k=0}^{\prime n} c_k T_k(x)$.
Wielomiany Czebyszewa są układem wielomianów ortogonalnych, stąd tworzą one bazę dla przestrzeni wielomianów. Zdefiniowane są w następujący rekurencyjny sposób:

$$T_0(x) := 1,\ T_1(x) := x$$
$$T_k(x) := 2xT_{k-1}(x) - T_{k-2}(x)$$

Korzystając jednak z powyższej zależności rekurencyjnej do obliczenia wartości wielomianu w_n punkcie x (zakładając, że korzystamy z definicji w_n w postaci kombinacji liniowej wielomianów Czebyszewa) byłoby możliwe w czasie wykładniczym względem n.
Powyższe obserwacje prowadzą do wniosku, że potrzebny jest inny algorytm służący do obliczania wartości sumy $\sum_{k=0}^{\prime n} c_k T_k(x)$. Algorytm ten musi spełniać nie tylko założenia czasowe, ale musi być również numerycznie poprawny.

Zgodnie z książką *„Analiza Numeryczna", D.Kincaid, W.Cheney* algorytm jest numerycznie poprawny (numerycznie stabilny), jeśli dla lekko zaburzonych danych zwraca lekko zaburzony wynik.

Zapamiętując 2 poprzednie wielomiany Czebyszewa $T_{k-1}(x)$, $T_{k-2}(x)$ wielomian T_k możemy obliczyć w czasie stałym, stąd wielomian T_n możemy obliczyć w czasie liniowym względem n. W celu uzyskania wartości sumy $\sum_{k=0}^{\prime n} c_k T_k(x)$ uzyskane wielomiany Czebyszewa trzeba jednak mnożyć przez znane wartości

c_k. Zwiększenie ilości wykonywanych operacji wpływa negatywnie na numeryczną poprawność algorytmu.
Wykorzystując algorytm Clenshawa możemy uzyskać algorytm obliczający wartość sumy ${\sum'}_{k=0}^{n} c_k T_k(x)$ w czasie liniowym względem n, w którym wykonujemy stosunkowo mało operacji, dzięki czemu błędy numeryczne będą stosunkowo niewielkie.

2.2 Algorytm Clenshawa

Dla dowolnego ciągu $V_k(x)$ zdefiniowanego w sposób rekurencyjny, który można zapisać w postaci:

$$V_k(x) = -\alpha V_{k-1}(x) - \beta V_{k-2}(x)$$

oraz dla dowolnego skończonego ciągu c_k możemy zdefiniować następującą wzór rekurencyjny:

$$\begin{gathered} B_{n+2}(x) := B_{n+1}(x) := 0 \\ B_k(x) := -\alpha_k(x) B_{k+1}(x) - \beta_k(x) B_{k+2}(x) + c_k \end{gathered}$$

dla którego zachodzi równość:

$$\textstyle\sum_{k=0}^{n} c_k V_k(x) = B_0(x) V_0(x) + B_1(x)(V_1(x) + \alpha_0(x) V_0(x))$$

Dla wielomianów Czebyszewa definiujemy:

$$\begin{gathered} \alpha_k(x) := -2x, B_k := 1, \text{ stąd:} \\ B_k(x) := 2x B_{k+1}(x) - B_{k+2}(x) + c_k, \text{ stąd:} \\ \textstyle{\sum'}_{k=0}^{n} c_k T_k(x) = -\frac{1}{2} c_0 T_0 + \sum_{k=0}^{n} c_k T_k(x) = \\ -\frac{1}{2} c_0 + B_0(x) T_0(x) + B_1(x)(T_1(x) - 2x T_0(x)) = \\ -\frac{1}{2} c_0 + B_0(x) - x B_1(x) = \\ -\frac{1}{2} c_0 + 2x B_1(x) - B_2(x) + c_0 - x B_1(x) = \\ x B_1(x) - B_2(x) + \frac{1}{2} c_0 = \\ \frac{1}{2}(2x B_1(x) - 2 B_2(x) + c_0) = \\ \frac{1}{2}(2x B_1(x) - B_2(x) + c_0 - B_2(x)) = \frac{1}{2}(B_0(x) - B_2(x)) \end{gathered}$$

3 Obliczanie wartości sumy ${\sum'}_{k=0}^{n} c_k T_k(x)$

3.1 Algorytm Clenshawa dla wielomianów Czebyszewa

Do obliczenia wartości sumy ${\sum'}_{k=0}^{n} c_k T_k(x)$ możemy użyć wersji algorytmu Clenshawa dla wielomianów Czebyszewa.

$$\begin{gathered} \text{Niech: } B_{n+2}(x) := B_{n+1}(x) := 0 \\ B_k := 2x B_{k+1} - B_{k+2} + c_k, \\ \text{wówczas: } c_k = B_k - 2x B_{k+1} + B_{k+2} \end{gathered}$$

$$\sum_{k=0}^{\prime n} c_k T_k(x) = \sum_{k=0}^{\prime n}(B_k - 2xB_{k+1} + B_{k+2})T_k(x) =$$
$$\sum_{k=0}^{\prime n} B_k T_k(x) - \sum_{k=0}^{\prime n} 2xB_{k+1}T_k(x) + \sum_{k=0}^{\prime n} B_{k+2}T_k(x) =$$
$$\sum_{k=0}^{\prime n} B_k T_k(x) - \sum_{k=1}^{\prime n+1} 2xB_k T_{k-1}(x) + \sum_{k=2}^{\prime n+2} B_k T_{k-2}(x)$$
Ponieważ $B_{n+2} = B_{n+1} = 0$, stąd
$B_{n+1}T_n(x) = B_{n+1}T_{n-1}(x) = B_{n+2}T_n(x) = 0$, stąd:
$$\sum_{k=0}^{\prime n} B_k T_k(x) - \sum_{k=1}^{\prime n+1} 2xB_k T_{k-1}(x) + \sum_{k=2}^{\prime n+2} B_k T_{k-2}(x) =$$
$$\sum_{k=0}^{\prime n} B_k T_k(x) - \sum_{k=1}^{\prime n} 2xB_k T_{k-1}(x) + \sum_{k=2}^{\prime n} B_k T_{k-2}(x) =$$
$$\frac{1}{2}B_0T_0(x) + B_1T_1(x) + \sum_{k=2}^{n} B_k T_k(x) - \frac{1}{2}2xB_1T_0(x) -$$
$$\sum_{k=2}^{n} 2xB_k T_{k-1}(x) - \frac{1}{2}B_2T_0(x) + \sum_{k=2}^{n} B_k T_{k-2}(x)$$
Ponieważ $T_0(x) = 0$, $T_1(x) = x$, stąd:
$$\frac{1}{2}B_0T_0(x) + B_1T_1(x) - \frac{1}{2}2xB_1T_0(x) - \frac{1}{2}B_2T_0(x) +$$
$$\sum_{k=2}^{n} B_k T_k(x) - \sum_{k=2}^{n} 2xB_k T_{k-1}(x) + \sum_{k=2}^{n} B_k T_{k-2}(x) =$$
$$\frac{1}{2}B_0 + xB_1 - xB_1 - \frac{1}{2}B_2 + \sum_{k=2}^{n}(B_kT_k(x) - 2xB_kT_{k-1}(x) + B_kT_{k-2}(x)) =$$
$$\frac{1}{2}(B_0 - B_2) + \sum_{k=2}^{n} B_k(T_k(x) - 2xT_{k-1}(x) + T_{k-2}(x))$$
Ponieważ $T_k(x) = 2xT_{k-1}(x) - T_{k-2}(x)$, stąd
$T_k(x) - 2xT_{k-1}(x) + T_{k-2}(x) = 0$, stąd:
$$\frac{1}{2}(B_0 - B_2) + \sum_{k=2}^{n} B_k(T_k(x) - 2xT_{k-1}(x) + T_{k-2}(x)) =$$
$$\frac{1}{2}(B_0 - B_2) + \sum_{k=2}^{n} B_k \cdot 0 = \frac{1}{2}(B_0 - B_2)$$

Z zależności rekurencyjnej:

$$B_k := 2xB_{k+1} - B_{k+2} + c_k$$

widać, że zapamiętując 2 następne wyrazy B_{k+1} oraz B_{k+2} możemy obliczyć B_k wykonując mnożenie razy $2x$ oraz odejmowanie i dodawanie. Mnożenie razy 2 jest jedynie zwiększeniem cechy naszej liczby, a czas odejmowania oraz dodawania jest zaniedbywany względem czasu mnożenia, stąd przyjmując operację mnożenia razy x za operację jednostkową możemy obliczyć B_0 w czasie liniowym względem n, wyraz B_2 będzie już również w pamięci naszego algorytmu, stąd policzenie wartości sumy: $\sum_{k=0}^{\prime n} c_k T_k(x) = \frac{1}{2}(B_0 - B_2)$ możliwe jest w czasie $O(n)$.

3.2 Algorytm obliczania sumy $s_n(x) = \sum_{k=0}^{\prime n} c_k T_k(x)$

Znając wartości c_k dla $k = 0, 1, ..., n$ oraz wykorzystując wartości pomocnicze B_{k+1}, B_{k+2} wynoszące początkowo 0 możemy obliczyć B_0 za pomocą pętli od n do 1 wykonując następujący algorytm:

for $i = n : 1$
$B_k := 2xB_{k+1} - B_{k+2} + c_i$
$B_{k+2} := B_{k+1}$
$B_{k+1} := B_k$

Po wykonaniu obliczeń w tej pętli wartości pomocnicze B_{k+1}, B_{k+2} wynoszą odpowiednio B_1, B_2, stąd wartość $\frac{1}{2}(B_0 - B_2)$ możemy obliczyć w następujący sposób:

$$s_n(x) = {\sum_{k=0}^{n}}' c_k T_k(x) = \frac{1}{2}(B_0 - B_2) = xB_{k+1} - B_{k+2} + \frac{1}{2}c_0$$

4 Numeryczna poprawność badanego algorytmu

4.1 Wstęp

Do obliczeń użyję programu w języku Julia w wersji 1.0.1. Liczby zmiennoprzecinkowe reprezentowane będą w podwójnej precyzji.

Dokładność wyników prezentowana jest jako ilość cyfr znaczących wyniku. Ilość cyfr znaczących obliczam za pomocą wzoru:

$$-log_{10}(|\frac{x-x0}{x0}|)$$, gdzie $x0$ - wartość dokładna, x - wartość przybliżona

Do wyznaczania błędu względnego używam standardowego wzoru:

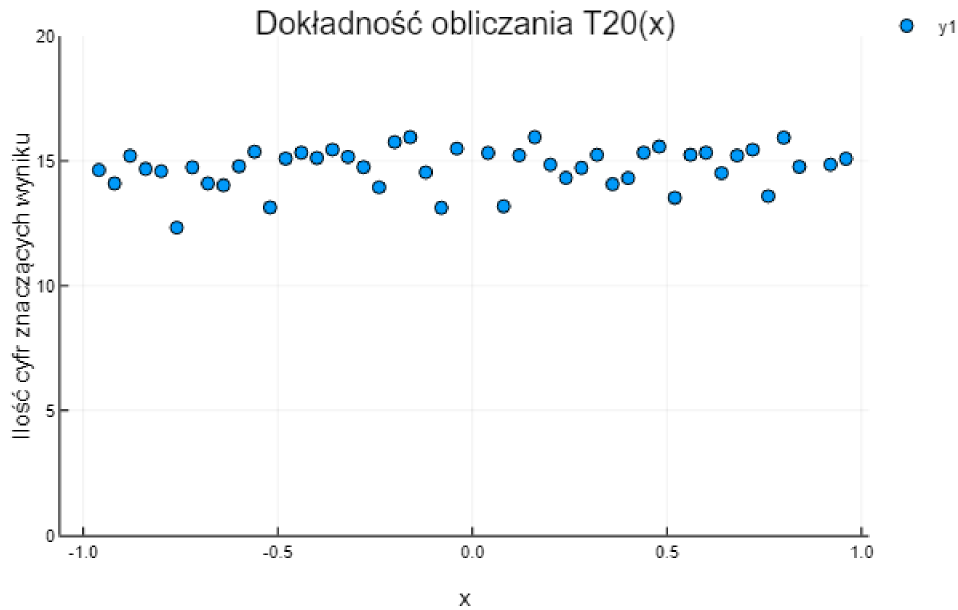
$$|\frac{x-x0}{x0}|$$, gdzie $x0$ - wartość dokładna, x - wartość przybliżona

4.2 Dokładność algorytmu Clenshawa

Wiemy, że dla $x \in [-1, 1]$ wartość n-tego wielomianu Czebyszewa w punkcie x możemy obliczyć za pomocą wzoru $T_n(x) = \cos(n \cdot \text{acos}(x))$. Dla

$$c_k = \begin{cases} 0 & \text{dla} \quad k = 0, 1, ..., n-1 \\ 1 & \text{dla} \quad k = n \end{cases}$$

spełniona jest równość ${\sum_{k=0}^{n}}' c_k T_k(x) = T_n(x)$ (dla $n > 0$). Korzystając z tego możemy zbadać z jaką dokładnością algorytm Clenshawa oblicza wartość $T_n(x)$, za wartość dokładną uznając tę obliczaną ze wzoru $T_n(x) = \cos(n \cdot \text{acos}(x))$. Dokładność przedstawiona będzie jako ilość cyfr znaczących wyniku, obliczenia zostaną wykonane dla $T_{20}(x)$, w 50 równoodległych punktach z przedziału $[-1, 1]$.

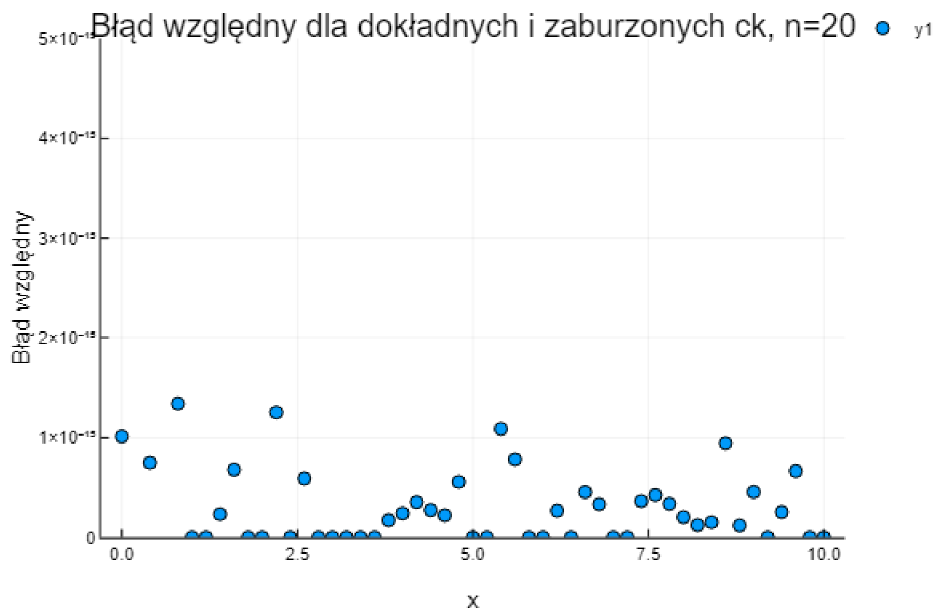


Na wykresie widać, że algorytm Clenshawa obliczył wartość $T_{20}(x)$, dla $x \in [-1, 1]$ ze średnią dokładnością 15 cyfr znaczących wyniku, co jest zadowalające.

4.3 Poprawność numeryczna algorytmu Clenshawa

Współczynniki $c_k := \sqrt{k}$

Niech $c_k := \sqrt{k}$, oraz $\overline{c_k} := c_k(1 + \epsilon_k)$, dla $\epsilon_k \in [-2^{-48}, 2^{-48}]$. Algorytm Clenshawa jest numerycznie poprawny, jeśli wartości sumy $\sum_{k=0}^{\prime n} c_k T_k(x)$ dla dokładnych (c_k) oraz lekko zaburzonych $(\overline{c_k})$ danych będą do siebie zbliżone, czyli jeśli błąd względny tych wartości będzie niewielki. Wykres przedstawia obliczenia dla $n = 20$ oraz $x \in [0, 10]$, wyniki dla innych wartości n oraz x nie różnią się znacznie od tych przedstawionych poniżej.



Z wykresu widzimy, że błąd względny wyniku jest nie większy, niż 10^{-14}, a dla niektórych argumentów jest on mniejszy niż precyzja Float64, stąd reprezentowany jest jako 0. Należy również zauważyć, że wartość zaburzenia współczynników c_k jest losowa, a więc i błąd względny wyniku zależy od pewnej losowej wartości.

Poprawność numeryczna na przykładzie wielomianów interpolacyjnych

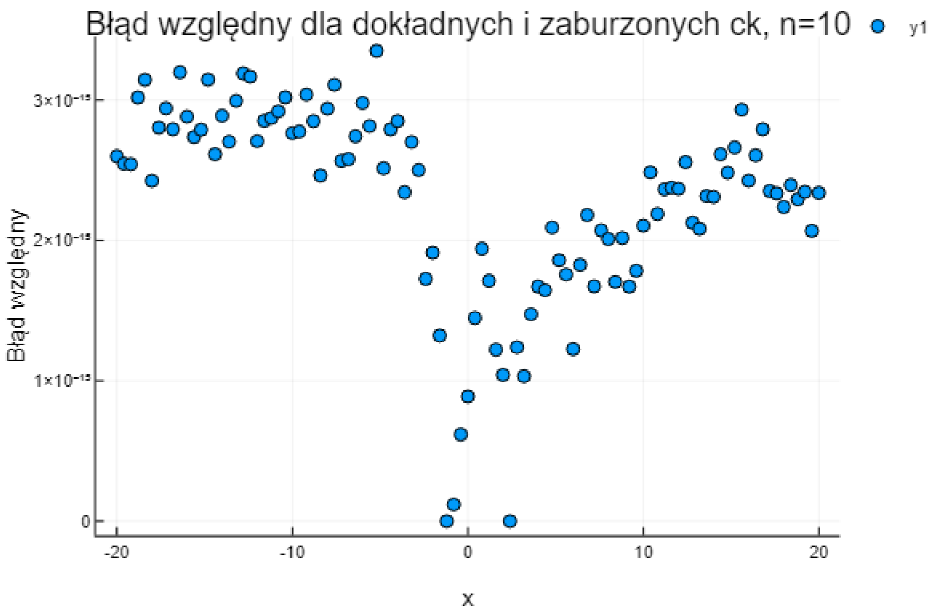
Powyższe badanie powtórzymy dla sumy z innymi współczynnikami. Niech:

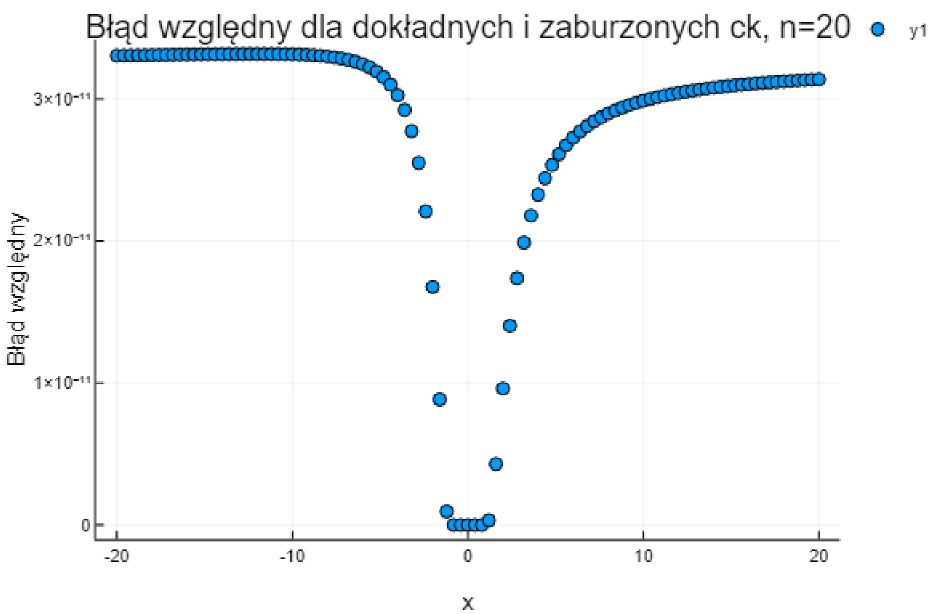
$$t_{n+1,k} = \cos\tfrac{2k+1}{2n+2}\pi$$
$$I_n(x) = \tfrac{2}{n+1} {\sum_{k=0}^{n}}' \left(\sum_{i=0}^{n} f(t_{n+1,i}) T_k(t_{n+1,i})\right) T_k(x)$$
$$u_{n-1,k} = \cos\tfrac{k}{n}\pi$$
$$J_n(x) = \tfrac{2}{n} {\sum_{k=0}^{n}}'' \left({\sum_{i=0}^{n}}'' f(u_{n-1,k}) T_i(u_{n-1,k})\right) T_k(x)$$

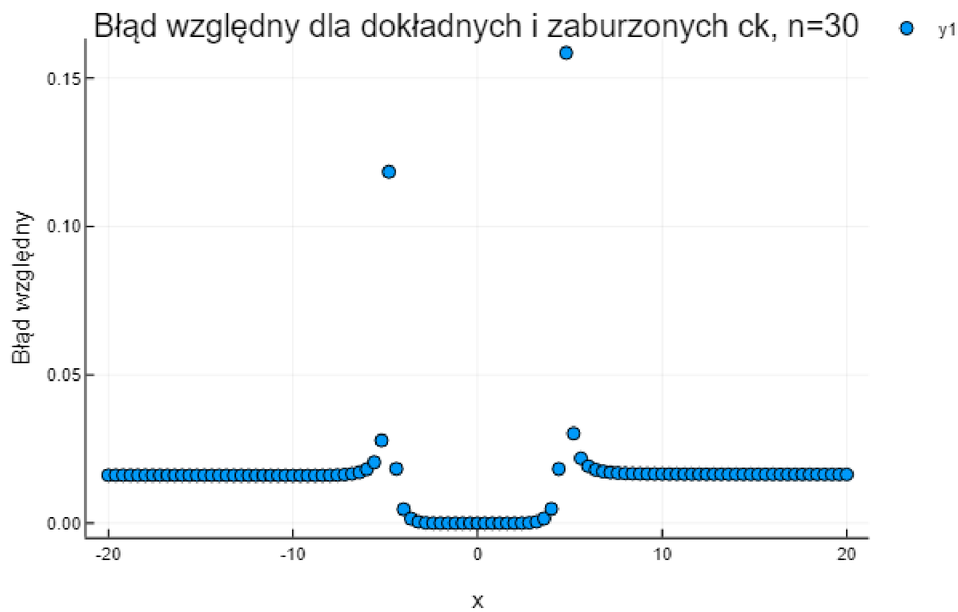
Wielomiany $I_n(x)$ oraz $J_n(x)$ interpolują funkcję $f(x)$ w węzłach Czebyszewa.

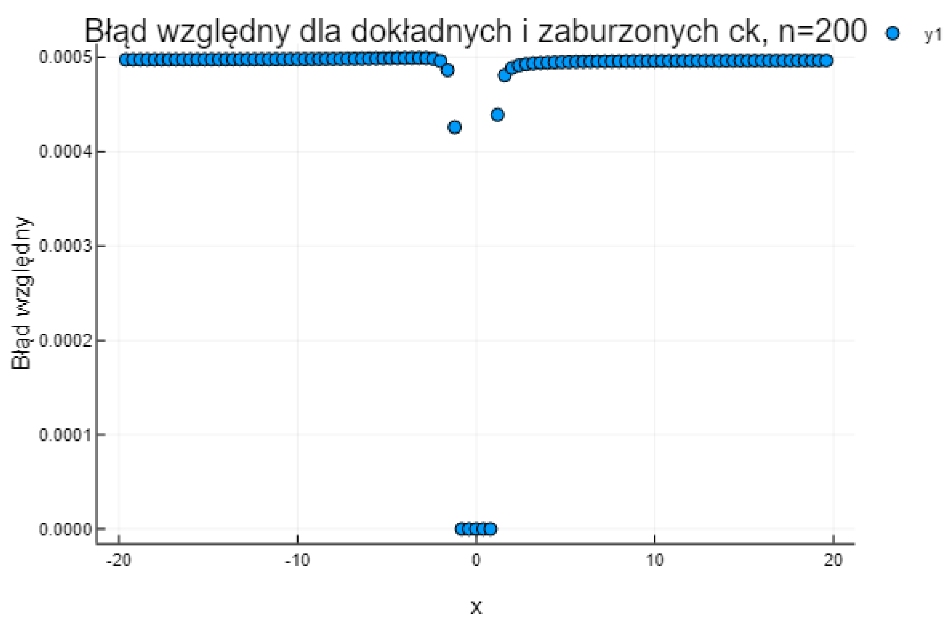
Wielomian $I_n(x)$

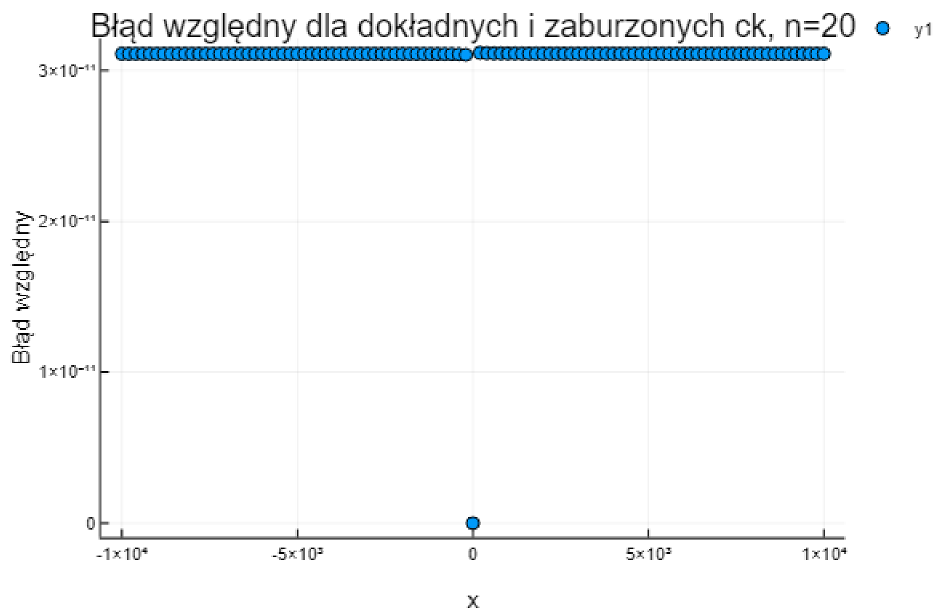
Niech $f(x) := x^{22} + x^7 + e^x$. Obliczymy wielomian I_n za pomocą algorytmu Clenshawa ze współczynnikami $c_k = \sum_{i=0}^{n} f(t_{n+1,i}) T_k(t_{n+1,i})$ dla różnych wartości n oraz dla x z różnych przedziałów. Obliczenia powtórzymy dla zaburzonych współczynników c_k i sprawdzimy błąd względny tych wartości.


Błąd względny dla dokładnych i zaburzonych ck, n=10
y1
Błąd względny
x


Błąd względny dla dokładnych i zaburzonych ck, n=20
y1
Błąd względny
x


Błąd względny dla dokładnych i zaburzonych ck, n=30
y1
Błąd względny
0.15
0.10
0.05
0.00
-20
-10
0
10
20
x


Błąd względny dla dokładnych i zaburzonych ck, n=200
y1
Błąd względny
0.0005
0.0004
0.0003
0.0002
0.0001
0.0000
-20
-10
0
10
20
x



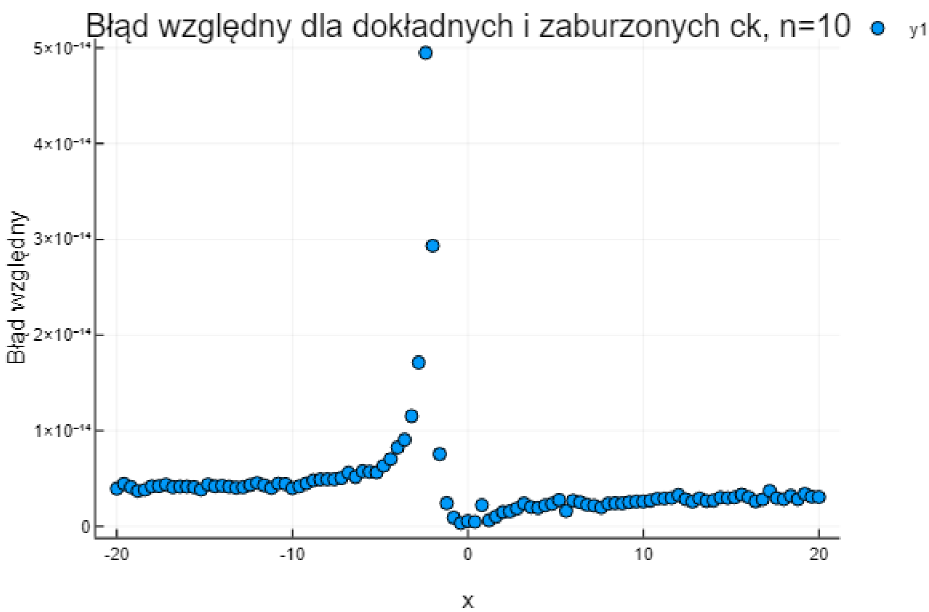
Z wykresów widzimy, że niezależnie od wartości n błąd względny obliczeń jest najmniejszy dla $x \in [-5, 5]$, dla x spoza tego przedziału błąd względny wyniku nie zmienia się, niezależnie od zmian x (dla zadanej wartości n).
Większe znaczenie dla błędu względnego wyniku ma zmiana wartości n. Dla $n \in [1, 20]$ błąd względny wyniku nie przekracza wartości 10^{-10}, dla $n \in [21, 200]$ błąd względny przyjmuje wartości z przedziału $[10^{-4}, 10^{-1}]$, dla $n > 200$ obliczenie wartości wielomianu $I_n(x)$ przestaje być możliwe (otrzymywane wartości są zbyt duże, by na nich operować). Wyniki te utrzymują się dla innych funkcji $f(x)$.

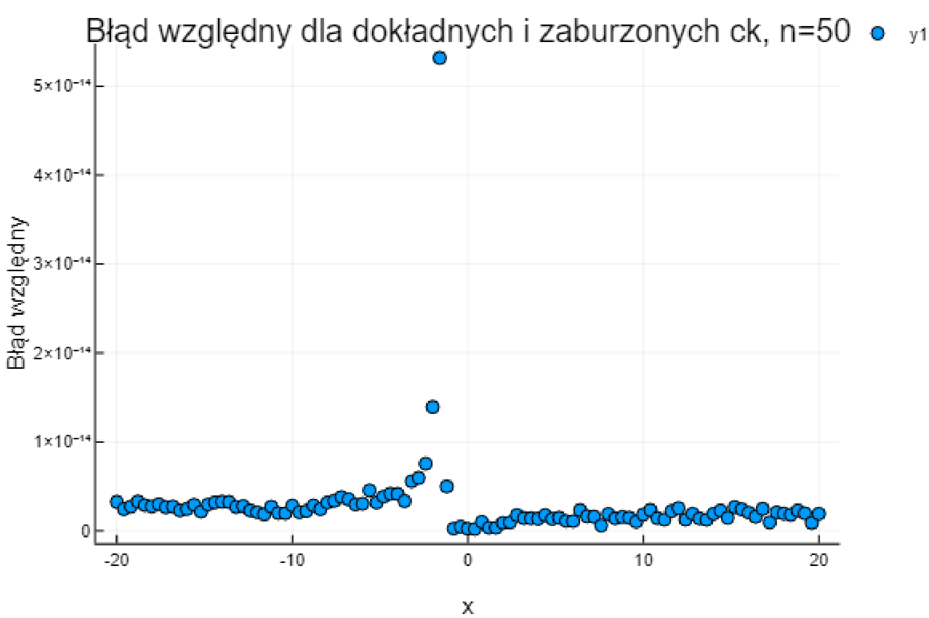
Wielomian $J_n(x)$

Niech $f(x) := x^{22} + x^7 + e^x$. Obliczymy wielomian J_n za pomocą algorytmu Clenshawa ze współczynnikami

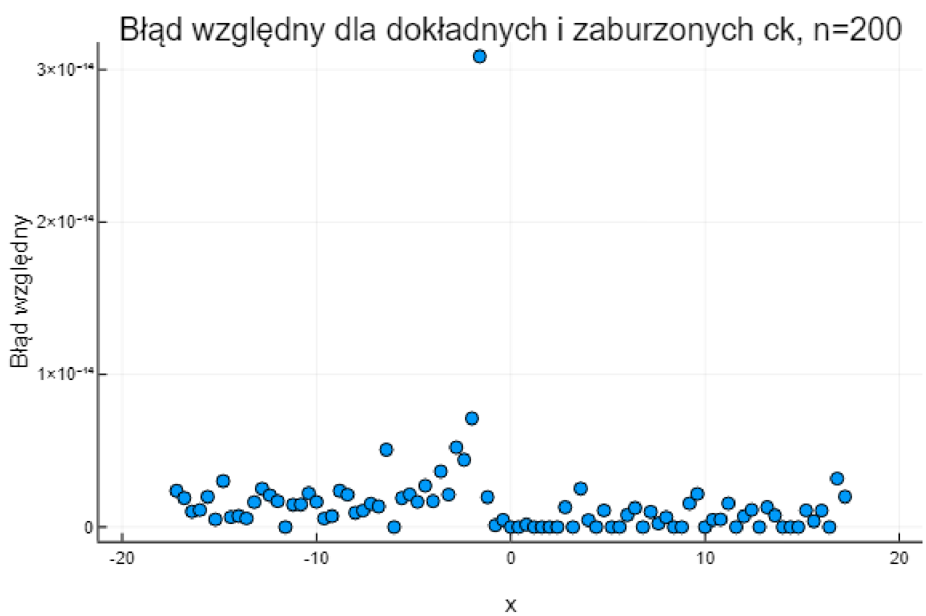
$$c_k = \sum_{i=0}^{\prime\prime n} f(u_{n-1,k}) T_i(u_{n-1,k}) =$$
$$\sum_{i=0}^{\prime n} f(u_{n-1,k}) T_i(u_{n-1,k}) - \frac{1}{2} f(u_{n-1,n}) T_i(u_{n-1,n})$$

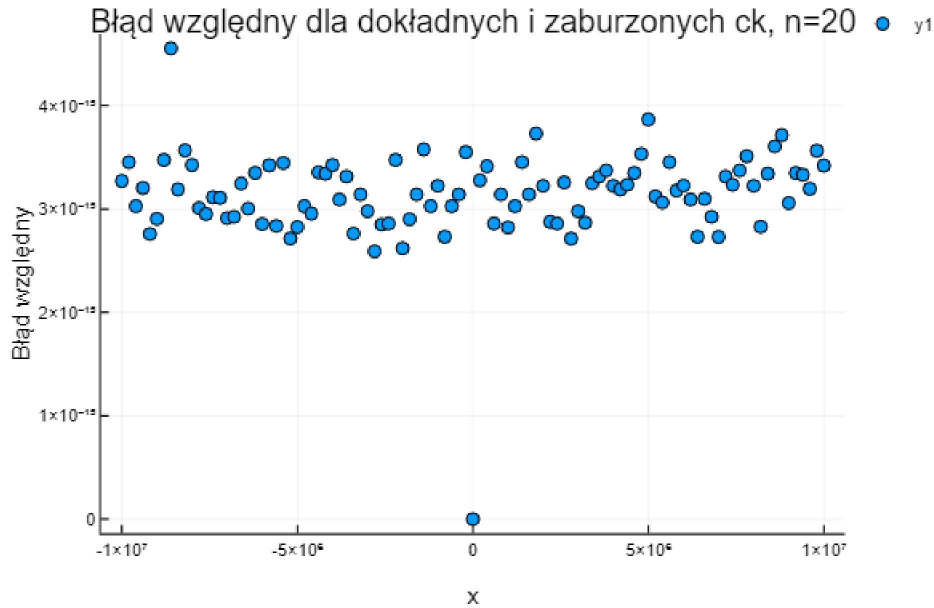
Współczynniki c_k obliczymy również za pomocą algorytmu Clenshawa. Obliczenia wykonamy dla różnych wartości n oraz dla x z różnych przedziałów. Obliczenia powtórzymy dla zaburzonych wartości współczynników c_k i sprawdzimy błąd względny tych wartości.


Błąd względny dla dokładnych i zaburzonych ck, n=10
y1
Błąd względny
x


Błąd względny dla dokładnych i zaburzonych ck, n=50
y1
Błąd względny
x


Błąd względny dla dokładnych i zaburzonych ck, n=100
y1
Błąd względny
5×10⁻¹⁴
4×10⁻¹⁴
3×10⁻¹⁴
2×10⁻¹⁴
1×10⁻¹⁴
0
-20
-10
0
10
20
x


Błąd względny dla dokładnych i zaburzonych ck, n=200
Błąd względny
3×10⁻¹⁴
2×10⁻¹⁴
1×10⁻¹⁴
0
-20
-10
0
10
20
x



Jak widać na wykresach dla $J_n(x)$ błąd względny sumy obliczonej za pomocą zaburzonych oraz niezaburzonych współczynników c_k nie przekracza 10^{-13}, niezależnie od n ani od przedziału x. Podobnie, jak dla wielomianu $I_n(x)$ dla dużych wartości n oraz x wartość wielomianu jest niemożliwa do policzenia. Wyniki te potwierdzają się dla innych funkcji $f(x)$.

4.4 Wnioski

Na powyższych przykładach widać, że istnieją takie współczynniki c_k, które nawet lekko zaburzone powodują, że błąd względny wyniku wynosi 10^{-1}. Dla wielu innych współczynników algorytm Clenshawa zachowywał poprawność numeryczną. Błąd względny pomiędzy sumą o współczynnikach prawidłowych, a sumą o lekko zaburzonych współczynnikach miał najczęściej wartość rzędu pomiędzy 10^{-15}, a 10^{-10}. Jest to zadowalający wynik. Warto również zaznaczyć, że nawet dla skrajnych wartości n, czy też x nie udało się uzyskać dużych wartości błędu względnego, co wskazuje na numeryczną poprawność tego algorytmu w większości przypadków.

5 Podsumowanie

Wzorując się na algorytmie Clenshawa skonstruowaliśmy skuteczny algorytm do obliczania wartości sum postaci $s_n(x) = \sum_{k=0}^{\prime n} c_k T_k(x)$. Algorytm ten pozwala wykonać jak najmniej operacji, co gwarantuje nie tylko szybkie działanie, ale również numeryczną poprawność dla większości współczynników c_k. Z przeprowadzonych eksperymentów widzimy, że błąd względny pomiędzy wartością tej sumy dla poprawnych oraz lekko zaburzonych wartości c_k w większości przypadków nie przekracza 10^{-10}, co jest bardzo dobrym wynikiem.

6 Literatura

1) D. Kincaid, W. Cheney, Analiza numeryczna, WNT, 2005.