13.10.2019

Algorytmy numeryczne
Zadanie 1

Wojciech Rosiński
Nr indeksu: 240425

1. Opis zadania

Zadanie polegało na napisaniu programu wyliczającego wartości funkcji ln(1+x) z użyciem wzoru Taylora, a dokładnie rozwinięcia MacLaurina, który prezentuje się następująco:

$$\ln(1+x) = \sum_{n=1}^{\infty} \frac{(-1)^{n+1}}{n} * x^n$$

Ponadto, głównym założeniem zadania był podział sposobu wyliczania wartości wyżej wymienionej funkcji na 4 warianty i porównanie otrzymanych z nich wyników z wbudowaną funkcją biblioteczną. Wspomniane sposoby otrzymania wyniku prezentują się następująco:

1. sumując elementy szeregu potęgowego obliczane bezpośrednio ze wzoru Taylora w kolejności od początku,
2. sumując elementy szeregu potęgowego obliczane bezpośrednio ze wzoru Taylora w kolejności od końca,
3. sumując elementy szeregu potęgowego od początku, ale obliczając kolejny wyraz szeregu na podstawie poprzedniego,
4. sumując elementy szeregu potęgowego od końca, ale obliczając kolejny wyraz szeregu na podstawie poprzedniego.

Do obliczenia kolejnego wyrazu szeregu na podstawie poprzedniego wykorzystano wzór wyliczony z ilorazu dwóch, następujących po sobie elementów szeregu:

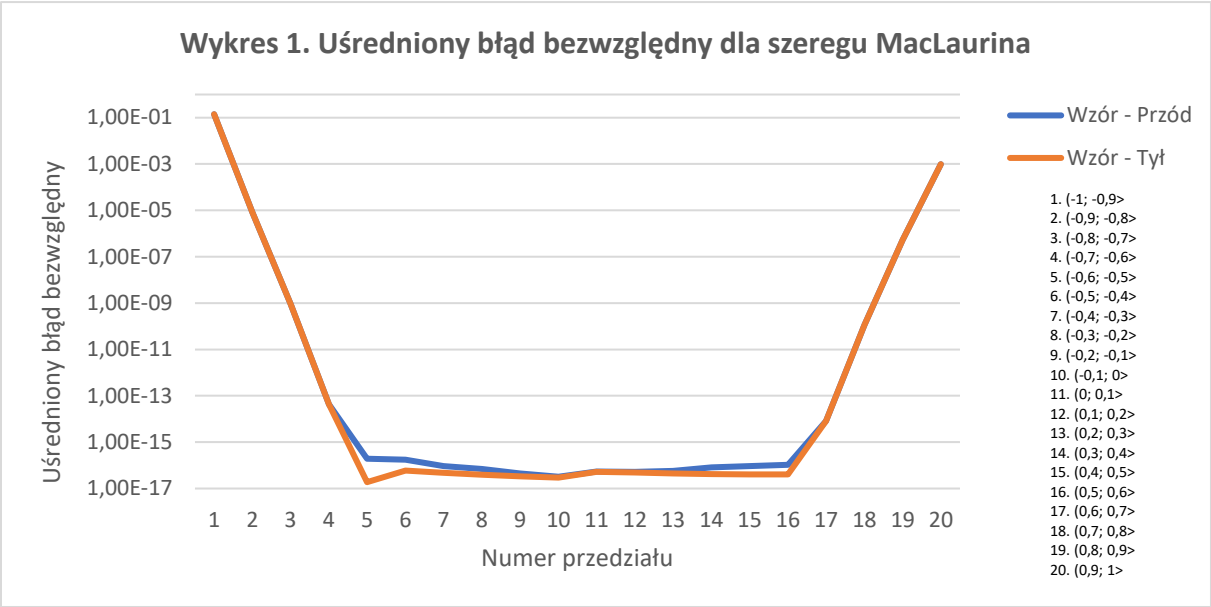
$$S_{n+1} = \frac{-S_n * x * n}{n+1}$$

2. Rozwiązanie zadania

Funkcje liczące zostały zaimplementowane w języku Java w postaci metod klasy o nazwie „lnx". Wszystkie metody realizują swe obliczenia na zmiennych typu double. Dodatkowo, zgodnie ze wskazówką, do działania potęgowania wykorzystano samodzielnie przygotowane rozwiązanie.

Program przyjmuje milion argumentów z przedziału $(-1;\ 1>$ ze względu na ograniczenie zmiennej x postaci $-1 < x \leq 1$. Szeroki zakres argumentów został podzielony na 20 przedziałów. W wyniku czego otrzymano 50 000 wartości pojedynczej funkcji na każdy przedział. Program wylicza różnicę pomiędzy każdą z czterech funkcji a funkcją biblioteczną. Różnice te są sumowane grupami i na podstawie takich sum liczony jest uśredniony błąd bezwzględny dla każdego przedziału.

Celem otrzymania wniosków potrzebnych do rozwikłania hipotezy nr 3, program porównuje pojedyncze błędy bezwzględne między funkcjami 1 i 3 oraz 2 i 4. W przypadku większego błędu dla funkcji nr 1 program uznaje licznik dokładniejszych obliczeń funkcji nr 3 (analogiczna sytuacja jest z funkcjami 2 i 4).



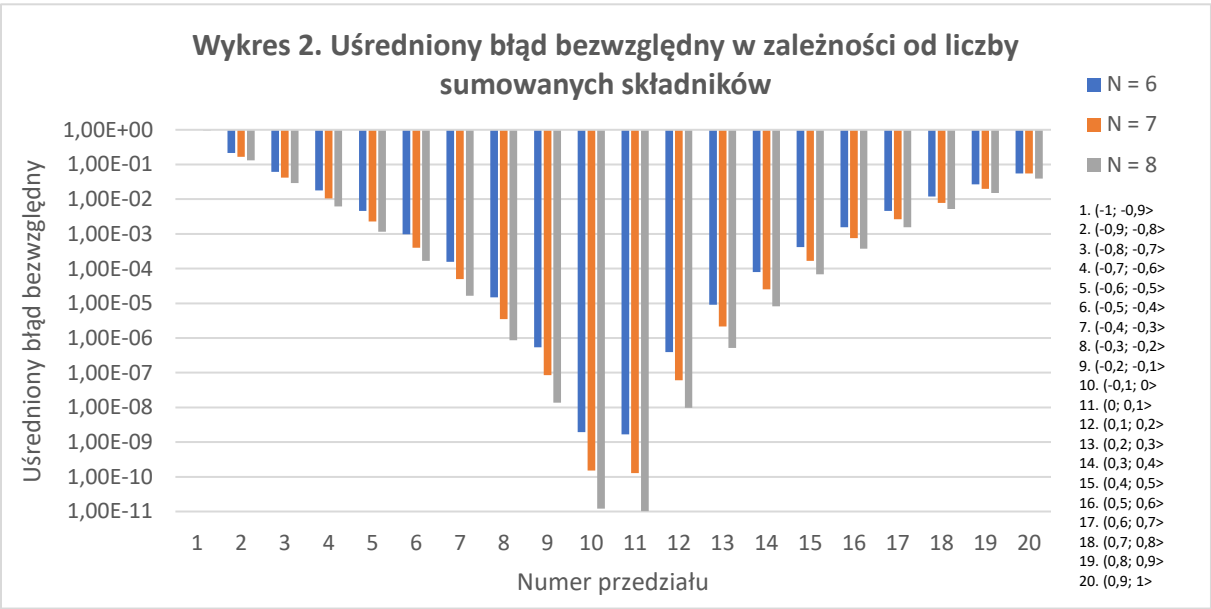


Tabela 1. Która funkcja wypadła lepiej, gdy występowała różnica?

Wzór - Przód	Poprzedni - Przód	Wzór - Tył	Poprzedni - Tył
48659	48688	24765	24952

Hipoteza nr 1 - Sumowanie od końca daje dokładniejsze wyniki niż sumowanie od początku, ponieważ zgodnie z wykresem nr 1 uśredniony błąd bezwzględny dla sumowania od tyłu jest mniejszy lub równy uśrednionemu błędowi bezwzględnemu sumowania od przodu. Zależność ta zachodzi dla obu sposobów wyliczania kolejnych elementów szeregu. Jest to potwierdzenie hipotezy.

Hipoteza nr 2 – Na wykresie nr 1 widać, że dla mniejszych argumentów uśredniony błąd bezwzględny jest również mniejszy. Przykładowo, dla przedziałów o numerach od 5 do 16, błąd jest zdecydowanie mniejszy niż dla przedziałów od 1 do 4 oraz od 17 do 20, gdzie znajdują się większe argumenty.

Hipoteza nr 3 – Sumowanie elementów obliczanych na podstawie poprzedniego daje dokładniejsze wyniki niż obliczanych bezpośrednio ze wzoru. Potwierdza to tabela nr 1, która przedstawia ilość razy, w których dana funkcja była dokładniejsza, w przypadku gdy wystąpiła różnica w błędach bezwzględnych.

Pytanie: Jak zależy dokładność obliczeń (błąd) od liczby sumowanych składników?

- Na to pytanie odpowiada wykres nr 2, który przedstawia uśredniony błąd bezwzględny w zależności od ilości sumowanych składników. Widać, że im większa liczba takich składników, tym błąd jest mniejszy.