Implementacja rozwinięć wybranych funkcji w szereg Maclaurina

Grams, Stanisław
251000 MFI UG
Algorytmy Numeryczne

17 października 2018

1 Sumowanie szeregów potęgowych

1.1 Rozwinięcia funkcji w szereg Maclaurina

$$\forall x \sin x = \sum_{n=0}^{\infty} \frac{(-1)^n}{(2n+1)!} x^{2n+1} = x - \frac{x^3}{3!} + \frac{x^5}{5!} - ...$$

$$\forall x \exp x = \sum_{n=0}^{\infty} \frac{x^n}{n!} = 1 + x + \frac{x^2}{2!} + \frac{x^3}{3!} + ...$$

1.2 O implementacji

Program *„taylor”* został napisany w języku C z użyciem bibliotek:

- *„quadmath.h”* — pozwalającej na użycie zmiennych typu $\_\_float128$,
- *„pthread.h”* — do celów obsługi wielowątkowości,
- *„glib.h”* — ułatwiającej pracę z tablicami znaków.

Wyniki działania programu zapisywane są do poszczególnych plików **.csv*.

1.3 Wyniki

Program uruchamiano na przedziale $x \in [-10; 9.99999]$ dla kolejnych parametrów $M = 4, 8, 16, 32, 64$ z krokiem 0.00001 (2×10^6 elementów). Z powodu braku wspomagania koprocesora zmiennoprzecinkowego dla typu poczwórnej precyzji, obliczenia zajmowały bardzo dużo czasu (dla $M = 256$ nawet **16** godzin!).

1.4 Konkluzje po implementacji i przeprowadzeniu doświadczeń

1.4.1 Odpowiedzi na postawione hipotezy

Hipoteza 1. Sumowanie od końca daje dokładniejsze wyniki niż sumowanie od początku.

Hipoteza dla zaimplementowanych funkcji, na przedziale $x \in [-10; 9.99999]$, jest prawdziwa **jedynie** w pierwszym przypadku, tj. sumując elementy szeregu potęgowego obliczane bezpośrednio ze wzoru Taylora w kolejności od końca.
Dowód: patrz *Hipoteza 3*.

Hipoteza 2. Używając rozwinięcia wokół 0 (szereg Maclaurina), przy tej samej liczbie składników szeregu dokładniejsze wyniki uzyskujemy przy małych argumentach.

Zaimplementowane funkcje potwierdzają hipotezę na przedziale $x \in [-10; 9.99999]$, co możemy łatwo zauważyć na wykresie *„01_functions_m002.svg"*. Funkcje 1, 2, 3 oraz 4 zbiegają się z wykresem funkcji 0 tylko w okolicach zera oraz w całkowitych wielokrotnościach liczby 2π (ze względu na użyte sprowadzenie do przedziału $[-2\pi; 2\pi]$).

Hipoteza 3. Sumowanie elementów obliczanych na podstawie poprzedniego daje dokładniejsze wyniki niż obliczanych bezpośrednio ze wzoru.

Zaimplementowane funkcje pokazują, że na przedziale $x \in [-10; 9.99999]$ hipoteza jest prawdziwa, co więcej, możemy powiedzieć, że w przypadku użycia typu $\_\_float128$, funkcje te mają taką samą dokładność, jak funkcja 2 (szereg Maclaurina, sumowanie od końca).
Dowodem niech będą wyniki średnich arytmetycznych różnic w stosunku do obliczeń popełnionych funkcjami wbudowanymi:
`func1() avg. absolute diff to func0() with M=16 is`
0.0000000000016539521796360597985907078303108796753612877507503184
`func2() avg. absolute diff to func0() with M=16 is`
0.0000000000016539521796360597985908280055981357440636938642850951
`func3() avg. absolute diff to func0() with M=16 is`
0.0000000000016539521796360597985908280055981357440636938642850951
`func4() avg. absolute diff to func0() with M=16 is`
0.0000000000016539521796360597985908280055981357440636938642850951
Na powyższych wynikach, dla parametru $M = 16$, zauważyć możemy dokładność co do 33 miejsc po przecinku.

1.4.2 Odpowiedzi na postawione pytania

Pytanie 1. Jak zależy dokładność obliczeń (błąd) od liczby sumowanych składników?

Im wyższa liczba sumowanych składników (parametr M), tym większa dokładność obliczeń (niższy błąd bezwzględny do obliczeń wykonanych przez funkcje wbudowane), oraz tym dłuższy czas wykonania programu.

1.5 Spostrzeżenia

1. Użycie __float128 — typu nie wspieranego przez wbudowany koprocesor znacznie zwiększa czas działania programu.

2. Spośród zbadanych parametrów wystarczająca dokładność uzyskiwana jest już od $M = 32$.

3. Sprowadzanie argumentu do przedziału $[-2\pi; 2\pi]$ zwiększa dokładność obliczeń, przyspiesza ich czas oraz zwiększa stabilność programu.

4. Znaczący przyrost wydajności można uzyskać przez podzielenie obliczeń pomiędzy kilka wątków.

1.6 Wykresy

Wykresy (w postaci wektorowej) załączono w katalogu *„report/plots/”*, natomiast wyniki średnich arytmetycznych w pliku *„report/diffs.txt”*
Do narysowania wykresów użyto oprogramowania *„GNU Octave”* wraz z pakietem *„gnuplot”*. Skrypty załączono w katalogu *„scripts/”*.