Analiza błędów

Laboratorium 1

Jakub Ciszewski, Wiktor Smaga

7 marca 2024

1 Zadanie 1.

Celem ćwiczenia jest zbadanie błędów powstających przy obliczaniu przybliżonej wartości pochodnej funkcji $\mathrm{tg}(x)$ w punkcie $x = 1$.

1.1 Porównanie wartości przybliżonej i dokładnej

Przy pomocy tego wzoru obliczamy przybliżoną wartość pochodnej:

$$f'(x) \approx \frac{f(x+h) - f(x)}{h}$$

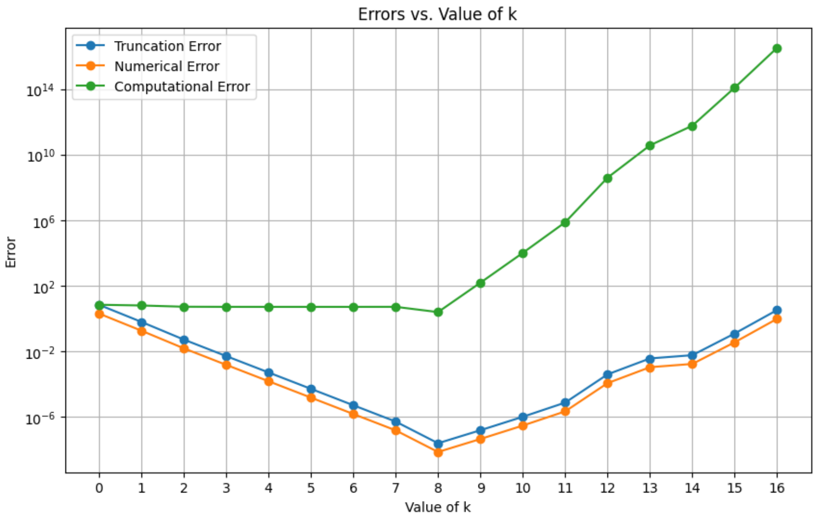
Następująca włanosć pozwala nam na obliczenie dokładnej wartości pochodnej:

$$f'(x) = 1 + \mathrm{tg}^2(x)$$

Szukane błędy:

- Błąd metody: Różnica między dokładną wartością pochodnej a jej przybliżoną wartością
$$|f'(x) - f'_{\text{approx}}(x)|$$
- Błąd numeryczny: Stosunek różnicy między dokładną wartością pochodnej a jej przybliżoną wartością do samej dokładnej wartości pochodnej
$$\left|\frac{f'(x) - f'_{\text{approx}}(x)}{f'(x)}\right|$$
- Błąd obliczeniowy: Stosunek różnicy między dokładną wartością pochodnej a jej przybliżoną wartością do wartości kroku h
$$\left|\frac{f'(x) - f'_{\text{approx}}(x)}{h}\right|$$

Porównując obie te wartości przy pomocy powyższych wzorów dla $h = 10^{-k}$, gdzie $k = 1, 2, 3, \ldots, 15$ otrzymujemy:



Wykres 1: Wykres zależności błędów od wartości k

1.2 Minimum wartości bezwzględnej błędu obliczeniowego

Minimum wartości bezwzględnej błędu obliczeniowego jest osiągane dla $h_{\text{min}} = 10^{-8}$. Wyznaczamy hmin przy pomocy wzoru:

$$h_{\text{min}} = \sqrt{\frac{2\epsilon_{\text{mach}}}{|f''(x)|}}$$

gdzie ϵ_{masz} to epsilon maszynowy, a $f''(x)$ to druga pochodna funkcji $\text{tg}(x)$.

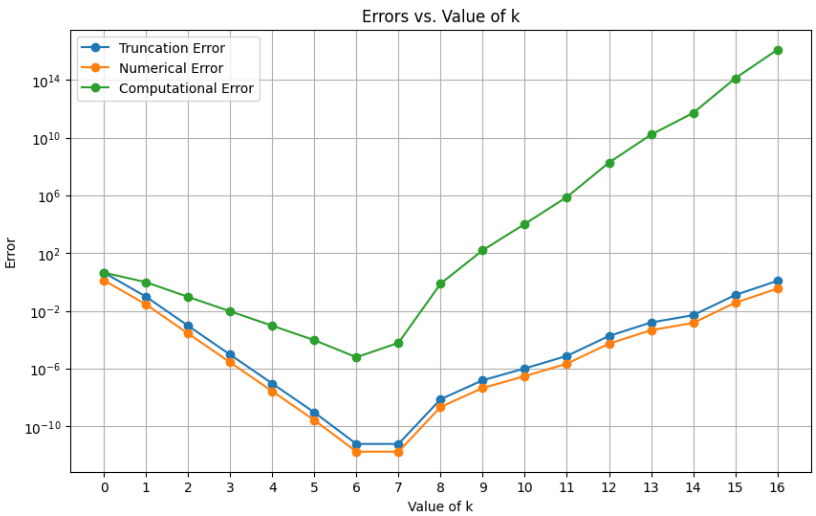
Wartość h_{min} obliczona ze wzoru:	9.123695225180455e-09
Wartość h_{min} odczytana z wykresu:	1e-08

1.3 Wzór różnic centralnych

Wzór różnic centralnych:

$$f'(x) \approx \frac{f(x+h) - f(x-h)}{2h}$$

Powtarzamy obliczenia dla wzoru różnic centralnych:



Wykres 2: Wykres zależności błędów od wartości k

1.4 Minimum wartości bezwzględnej błędu obliczeniowego dla wzoru różnic centralnych

Minimum wartości bezwzględnej błędu obliczeniowego dla wzoru różnic centralnych jest osiągane dla $h = 10^{-6}$.

Wzór na obliczenie h_{min} dla wzoru różnic centralnych:

$$h_{\text{min}} = \sqrt[3]{\frac{3\epsilon_{\text{mach}}}{|f'''(x)|}}$$

gdzie ϵ_{masz} to epsilon maszynowy, a $f'''(x)$ to trzecia pochodna funkcji $\text{tg}(x)$.

Wartość h_{min} obliczona ze wzoru:	1.8246730729023089e-06
Wartość h_{min} odczytana z wykresu:	1e-06

1.5 Wnioski

- Dla pierwszego wzoru najmniejsze błędy otrzymujemy dla $h = 10^{-8}$. Natomiast dla wzoru różnic centralnych dla $h = 10^{-6}$.
- Wzór różnic centralnych daje mniejsze błędy niż wzór różnic skończonych.
- Wartości h_{min} dla obu metod obliczone ze wzorów są zbliżone o rząd wielkości do wartości h_{min} odczytanych z wykresów.

2 Zadanie 2.

Celem ćwiczenia jest porównanie dokładności obliczeń dla pojedynczej precyzji - `float32`, podwójnej precyzji - `float64` oraz dla klasy `Fraction` z biblioteki `fractions`.

2.1 Definicja ciągu

W ramach ćwiczenia opracowano program generujący n wyrazów ciągu zadanego równanienm różnicowym:

$$x_{k+1} = 2.25x_k - 0.5x_{k-1} \tag{1}$$

z wyrazami początkowymi:

$$x_0 = \frac{1}{3} \quad x_1 = \frac{1}{12}$$

Dokładne rozwiązanie równania:

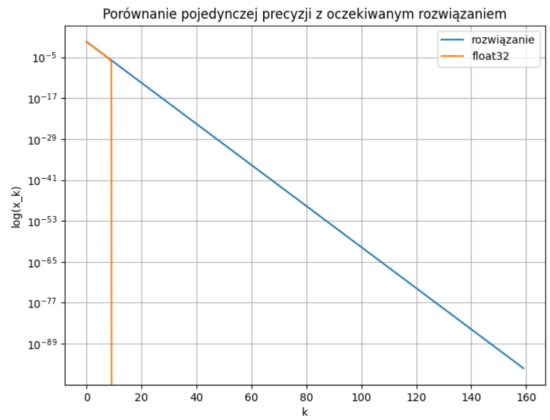
$$x_k = \frac{4^{-k}}{3} \tag{2}$$

2.2 Wygenerowane ciągi

Wygenerowano 5 ciągów:

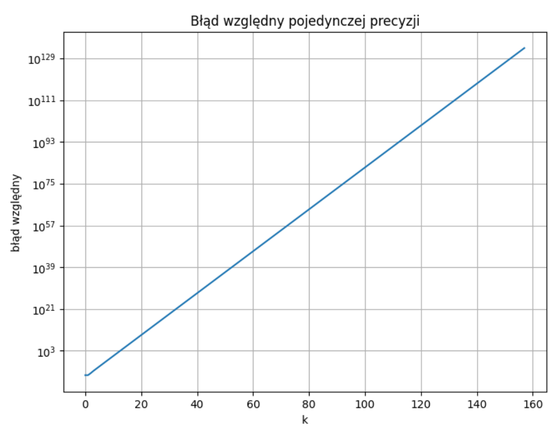
- Ciąg A - 225-cio wyrazowy, wygenerowany wzorem 1, dla pojedynczej precyzji.
- Ciąg B - 60-cio wyrazowy, wygenerowany wzorem 1, dla podwójnej precyzji.
- Ciąg C - 225-cio wyrazowy, wygenerowany wzorem 1, dla reprezentacji `Fraction` z biblioteki `fractions`.
- Ciąg D - 225-cio wyrazowy, wygenerowany wzorem 2, dla podwójnej precyzji, przygotowany do porównań
- Ciąg E - 60-cio wyrazowy, wygenerowany wzorem 2, dla podwójnej precyzji, przygotowany do porównań

2.3 Analiza ciągu A



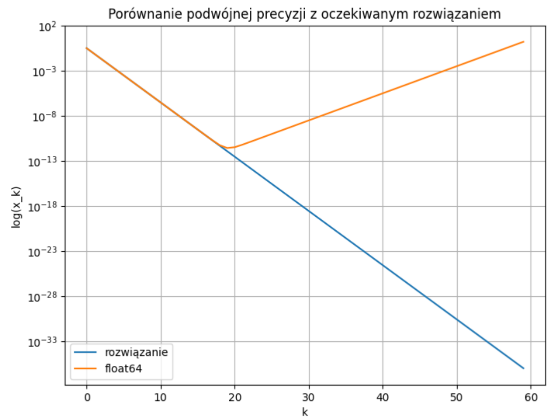
Wykres 3: Logarytmiczna zależność wartości ciągów A i D w zależności od wyrazu ciągu k

Na wykresie 3 możemy zauważyć, że wartości ciągu A pokrywają się z oczekiwanym rozwiązaniem do momentu przekroczenia granicy 10^{-7}. Dzieje się tak, ponieważ typ danych `float32` może przechowywać około 7 cyfr po przecinku. Dalej następuje przepełnienie zakresu wartości, przez co `python` ewaluuje przepełnione wartości na bardzo małe liczby, $-\infty$ lub `NaN`. Przy 160 wyrazie ciągu następuje ewaluacja na `NaN`, stąd zmniejszony zakres na osi OX.



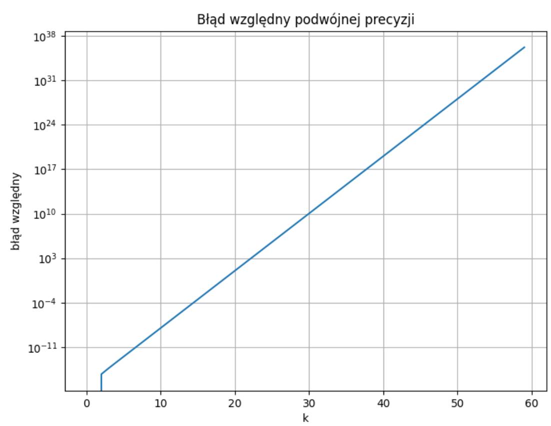
Wykres 4: Błąd względny w skali logarytmicznej wyrazów ciągu A w zależności od wyrazu ciągu k

2.4 Analiza ciągu B



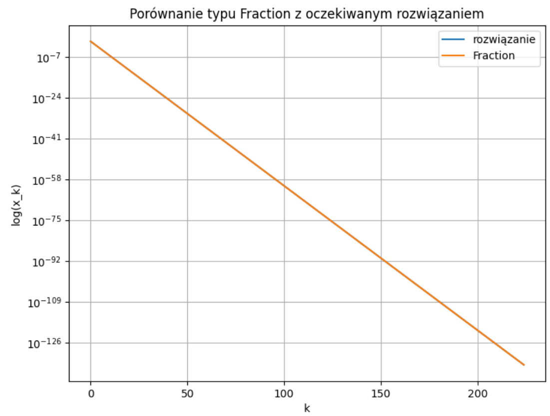
Wykres 5: Logarytmiczna zależność wartości ciągów B i E w zależności od wyrazu ciągu k

Na wykresie 5 można zauważyć, że podwójna wyrazy ciągu reprezentowane przy pomocy podwójnej precyzji pokrywają się z oczekiwanymi wynikami do około 18 wyrazu. Dalej wyrazy ciągu B rosną, odbiegając od oczekiwanego rozwiązania. Dzieje się tak przez akumulację błędów zaokrąglenia.



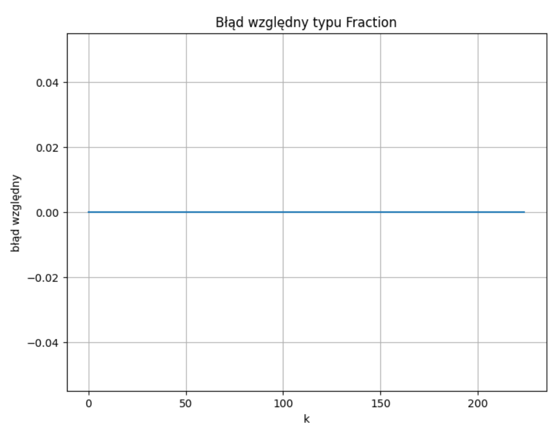
Wykres 6: Błąd względny w skali logarytmicznej wyrazów ciągu B w zależności od wyrazu ciągu k

2.5 Analiza ciągu C



Wykres 7: Logarytmiczna zależność wartości ciągów C i D w zależności od wyrazu ciągu k

Na wykresie 7 możemy zauważyć, że reprezentacja z biblioteki `fractions` pokrywa się całkowicie z oczekiwanym rezultatem. Dzieje się tak dlatego, że klasa `Fraction` używa dwóch liczb typu `int` do reprezentacji liczb wymiernych. W języku `python` typ danych `int` jest ograniczony tylko przez pamięć komputera co pozwala na reprezentację liczb zmiennoprzecinkowych z bardzo dużą dokładnością.



Wykres 8: Błąd względny w skali logarytmicznej wyrazów ciągu C w zależności od wyrazu ciągu k

2.6 Wnioski

- Najdokładniejszym sposobem reprezentacji liczb zmiennoprzecinkowych w języku `python` jest reprezentacja przy użyciu klasy `Fraction` z biblioteki `fractions`.
- Zwiększenie precyzji z pojedynczej na podwójną nie gwarantuję drastycznej poprawy dokładności

3 Bibliografia

- https://docs.python.org/3/library/fractions.html
- https://en.wikipedia.org/wiki/Double-precision_floating-point_format
- https://note.nkmk.me/en/python-int-max-value/
- https://en.wikipedia.org/wiki/Single-precision_floating-point_format