Zadanie numeryczne 1

Pola Dudek

14 października 2024

1 Wstęp

Różniczkowanie numeryczne znajduję istotne zastosowanie w sytuacjach, które nie pozwalają na analityczne obliczenie pochodnej funkcji. Jednym z takich przypadków jest wyznaczanie pochodnych na komputerze, który z uwagi na ograniczenia w obliczeniach symbolicznych nie może bezpośrednio obliczać pochodnych analitycznych - właśnie w takich sytuacjach stosuje się metody aproksymacji numerycznej, które pozwalają na przybliżenie wartości pochodnej.

Celem zadania jest analiza błędu pomiędzy pochodną obliczoną przy pomocy metod numerycznych, a pochodną obliczoną analitycznie. Do badań wykorzystano funkcje:

- $f(x) = \sin(x^3)$, dla $x = 0.2$ – zgodnie z poleceniem zadania, o pochodnej analitycznej $f'(x) = 3x^2 \cdot \cos(x^3)$
- $g(x) = x^3$, dla $x = 0.6$ – zgodnie z poleceniem "eksperymentu" i w celach porównawczych, o pochodnej analitycznej $g'(x) = 3x^2$

Podczas obliczeń użyto dwóch typów reprezentacji liczb zmiennoprzecinkowych: *float* (pojedyncza precyzja) i *double* (podwójna precyzja) - typy te różnią się liczbą bitów używanych do reprezentacji wartości, co bezpośrednio wpływa na dokładność i zasięg liczb, jakie mogą reprezentować.

Dodatkowo przeanalizowano dwie metody różniczkowania numerycznego:

1. Różnicę w przód:
$$f'(x) \approx \frac{f(x+h) - f(x)}{h} \tag{1}$$
2. Różnicę centralną:
$$f'(x) \approx \frac{f(x+h) - f(x-h)}{2h} \tag{2}$$

dla różnych wartości zmiennej h, gdzie h przyjmuje wartości w zakresie $10^{-16} \leqslant h \leqslant 10^0$.

Wartość błędu obliczono przy użyciu następującego wzoru:

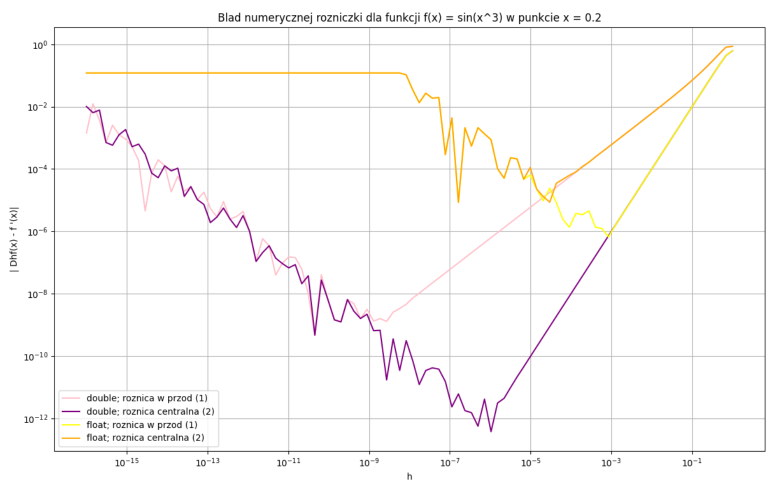
$$|D_h f(x) - f'(x)| \tag{3}$$

gdzie $D_h f(x)$ oznacza różniczkę numeryczną, a $f'(x)$ – różniczkę analityczną.

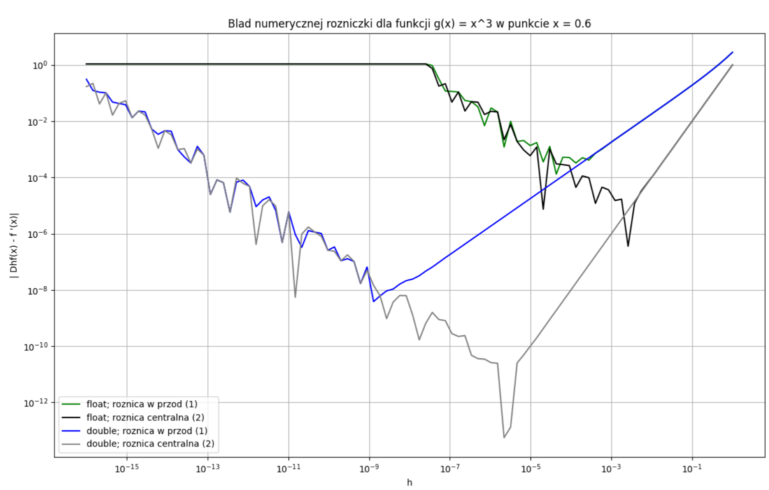
Program napisany w celu analizy problemu powstał w języku Python, przy pomocy bibliotek NumPy oraz Matplotlib.

2 Wykresy

Poniżej znajdują się wykresy narysowane przez program (biblioteka Matplotlib), w skali logarytmicznej:



Rysunek 1: Wykres 1: $f(x) = \sin(x^3)$, dla $x = 0.2$



Rysunek 2: Wykres 2: $g(x) = x^3$, dla $x = 0.6$

3 Podstawowa analiza wykresów

Analizując wykresy, można zauważyć wspólne schematy dla wszystkich czterech kombinacji formatów zmiennoprzecinkowych i metod różniczkowania:

- Obserwując wykres przeciwnie z kierunkiem osi h, błąd początkowo maleje w sposób liniowy, aż osiąga najmniejszą wartość. Następnie, przy dalszym zmniejszaniu wartości h, błąd ponownie rośnie, tym razem w sposób chaotyczny.
- Dla każdej linii reprezentującej jedną z czterech kombinacji można wyznaczyć wartość h, dla którego błąd jest najmniejszy - nazwijmy tą wartość $h_{\text{optymalne}}$.

Poniższa dyskusja będzie szczególnie przeprowadzona dla wykresu pierwszego, natomiast dla wykresu drugiego może być sformułowana bardzo analogicznie, ponieważ z obserwacji wynika, że pomimo użycia innej funkcji oraz punktu x wykres zachowywuje się w sposób bardzo podobny.

4 Dyskusja

1. Dla bardzo małych wartości h, błąd znacząco rośnie, co jest efektem działania błędu zaokrąglenia $\epsilon^{*}_{\text{zaokrąglenia}}$. Obraz tego błędu jest chaotyczny - może być problematyczny pod względem oszacowania przewidywalności. Zaobserwować można także ciekawą sytuację, w której obliczone wartości stają się nierozróżnialne – można to zauważyć dla niezwykle małych wartości h, gdy typem reprezentacji jest *float* - błąd przestaje się zmieniać, a wykres przyjmuje postać linii poziomej.
2. Dla większych wartości h, wzrost błędu jest proporcjonalny i co za tym idzie przewidywalny - wynika z dominacji błędu obcięcia $\epsilon^{**}_{\text{obcięcia}}$.

Przypisy:

- * Błąd zaokrąglenia to błąd, który powstaje w wyniku ograniczonej reprezentacji liczb w komputerach, szczególnie w arytmetyce zmiennoprzecinkowej. Ponieważ komputery mają ograniczoną liczbę bitów do przechowywania wartości, mogą wystąpić sytuacje, w których reprezentacja liczby będzie nieprecyzyjna .
- ** Błąd obcięcia to błąd charakterystyczny dla różniczki numerycznej. Powstaje ponieważ zamiast dokładnej pochodnej obliczamy jej przybliżenie "obcinając" kolejne składniki nieskończonej sumy wzoru Taylora.

3. Dalsza analiza wykresu pokazuje, że ze wszystkich czterech analizowanych kombinacji, najniższe wartości błędu osiąga metoda różnicy centralnej realizowana z typem reprezentacji liczb zmiennoprzecinkowych *double* dla $h_{\text{optymalnego}} \approx 10^{-6}$. Wynika to z większej precyzji reprezentacji *double* oraz dokładniejszego przybliżenia różnicy centralnej. Z kolei

najgorszy wynik uzyskano dla formatu *float* i metody różnicy w przód ($h_{\text{optymalne}} \approx 10^{-5} - 10^{-4}$).

4. Innym zjawiskiem nad którym warto się zastanowić jest fakt, że kombinacje metody różnicy w przód / różnicy centralnej z typem *double* osiągają niższe wartości błędu od kombinacji metody różnicy w przód / różnicy centralnej z typem *float* -można skłaniać się opini, że wybór typu reprezentacji liczby jest istotniejszy od wybranej metody numercznej, jednak wymagałoby to analizy większej ilości wzorów i algorytmów.

5 Wnioski

Podsumowując, najbardziej zbliżoną wartość różniczki numerycznej do wartości analitycznej uzyskano przy użyciu metody różnicy centralnej oraz typu *double*, dla wartości $h_{\text{optymalnego}} \approx 10^{-6}$.

Wyniki wykazują, że metody numeryczne są mniej tolerancyjne na błędy zaokrąglenia przy bardzo małych wartościach h, zwracając nieuporządkowane wartości i bardziej przewidywalne przy większych wartościach, gdzie dominuje błąd obcięcia. Ostatecznie, obserwacje sugerują, że wybór wartości zmiennej h oraz typu reprezentacji liczb zmiennoprzecinkowych ma kluczowe znaczenie dla minimalizacji błędów numerycznych.

Pomimo, że kombinacja wykorzystujące typ *double* wydają się być najbardziej efektywne w kontekście uzyskiwania wyników najbliższych wartościom analitycznym, warto wspomnieć, że gdy precyzyjność jest mniej istotna od szybkości wykonywanych obliczeń, bardziej sensowne będzie użycie typu zmiennoprzecinkowego *float* - na przyklad podczas optymalizacji algorytmu mającego zastosowanie w grach komputerowych. Podobnie metoda różnicy centralnej - choć jest znacznie bardziej dokładna, wymaga dwóch wywołań funkcji f co może mieć negatywny wpływ na wydajność dla kosztownych obliczeniowo funkcji.