Metody obliczeniowe w nauce i technice

Adam Naumiec
Marzec 2023

Laboratorium 1

Arytmetyka komputerowa

Spis treści

1. Treść zadań

1. Znaleźć maszynowy epsilon, czyli najmniejszą liczbę a taką, że:

$$a + 1 > 1.$$

2. Rozważamy problem ewaluacji funkcji $\sin x$ m.in. propagację błędu danych wejściowych, tj. błąd wartości funkcji ze względu na zakłócenie h w argumencie x:
   2.1. Ocenić błąd bezwzględny przy ewaluacji $\sin x$;
   2.2. Ocenić błąd względny przy ewaluacji $\sin x$;
   2.3. Ocenić uwarunkowanie dla tego problemu;
   2.4. Dla jakich wartości argumentu x problem jest bardzo czuły?

3. Funkcja sinus zadana jest nieskończonym ciągiem:

$$sin\, x = x - \frac{x^3}{3!} + \frac{x^5}{5!} - \frac{x^7}{7!} + \cdots.$$

   3.1. Jakie są błędy progresywny i wsteczny, jeśli przybliżamy funkcję sinus biorąc tylko pierwszy człon rozwinięcia, tj.:

$$sin\, x \approx x,$$

   dla:
   3.1.1. $x = 0.1$,
   3.1.2. $x = 0.5$,
   3.1.3. $x = 1.0$.

   3.2. Jakie są błędy progresywny i wsteczny, jeśli przybliżamy funkcję sinus biorąc pierwsze dwa człony rozwinięcia, tj.:

$$sin\, x \approx x - \frac{x^3}{6},$$

   dla:
   3.2.1. $x = 0.1$,
   3.2.2. $x = 0.5$,
   3.2.3. $x = 1.0$.

4. Zakładamy, że mamy znormalizowany system zmiennoprzecinkowy z:

$$\beta = 10,$$
$$p = 3,$$
$$L = -98.$$

   4.1. Jaka jest wartość poziomu UFL (underflow) dla takiego systemu?
   4.2. Jeśli:

$$x = 6.87 \cdot 10^{-97},$$
$$y = 6.81 \cdot 10^{-97},$$

jaki jest wynik operacji: $x - y$?

2. Rozwiązania

2.1. Zadanie 1

2.1.1. Maszynowy epsilon

Maszynowy epsilon, oznaczany ϵ jest równy:

$$\epsilon = B^{-(p-1)} = B^{1-p},$$

gdzie:

- B oznacza podstawę systemu liczbowego,
- p oznacza precyzję.

Wartość maszynowego epsilon jest równa najmniejszej wartości, jaką można dodać do liczby 1 w systemie reprezentacji zmiennoprzecinkowej, aby uzyskać inną wartość. Ma on taką samą wartość wykładnika jak liczba 1 i najmniejszą możliwą mantysę, która w przypadku systemów normalizowanych wynosi 1.

Dlatego wartość maszynowego epsilon zależy wyłącznie od podstawy systemu liczbowego B i precyzji p.

W języku Python 3.11:

```
from sys import float_info
from numpy import float32, finfo



def epsilon():
    eps_double = 1.0
    while (1.0 + eps_double / 2.0) != 1.0:
        eps_double /= 2.0

    eps_float = float32(1.0)
    while (float32(1.0) + eps_float / float32(2.0)) != float32(1.0):
        eps_float /= float32(2.0)

    print("Obliczone maszynowe epsilon dla double:", eps_double)
    print("Wartość rzeczywista (double) =", float_info.epsilon, "\n")

    print("Obliczone maszynowe epsilon dla float:", eps_float)
    print("Wartość rzeczywista (float) =", finfo(float32).eps)


if __name__ == "__main__":
    epsilon()
```

Wynikiem jest:

```
Obliczone maszynowe epsilon dla double: 2.220446049250313e-16
Wartość rzeczywista (double) = 2.220446049250313e-16

Obliczone maszynowe epsilon dla float: 1.1920929e-07
Wartość rzeczywista (float) = 1.1920929e-07
```

Należy pamiętać, że w języku Python nie ma typu *double*, a typ $float$ jest 64 bitowy i działa jak typ *double* w języku C. Typ $float$ (32 bitowy) można uzyskać za pomocą biblioteki $NumPy$ i funkcji $float32$.

Funkcja *epsilon* oblicza maszynowe epsilon dla liczb pojedynczej (32 bity) i podwójnej (64 bity) precyzji. Program zwraca także wartość maszynowego epsilon zapisaną w bibliotece *sys* za pomocą $float\_info.epsilon$ (64 bity) i w bibliotece $NumPy$ za pomocą $finfo(float32).eps$ (32 bity). Wartości te są równe wartościom bibliotecznym, zatem można twierdzić, że program działa poprawnie.

IEEE 754 - 2008	Common name	C++ data type	Base b	Precision p	Machine epsilon[a] $b^{-(p-1)}/2$	Machine epsilon[b] $b^{-(p-1)}$
binary16	half precision	*N/A*	2	11 (one bit is implicit)	$2^{-11} \approx 4.88e\text{-}04$	$2^{-10} \approx 9.77e\text{-}04$
binary32	single precision	float	2	24 (one bit is implicit)	$2^{-24} \approx 5.96e\text{-}08$	$2^{-23} \approx 1.19e\text{-}07$
binary64	double precision	double	2	53 (one bit is implicit)	$2^{-53} \approx 1.11e\text{-}16$	$2^{-52} \approx 2.22e\text{-}16$

Tabela 1. Standard IEEE 754-2008
(źródło: https://en.wikipedia.org/wiki/Machine_epsilon)

2.1.2. Wnioski

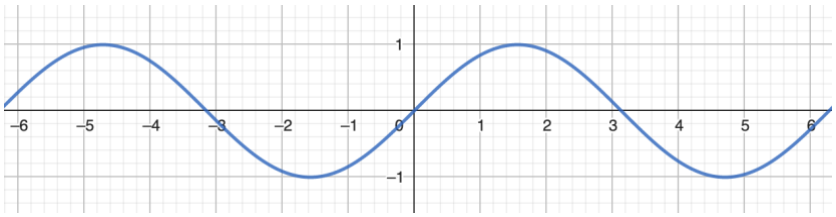
Przy dokonywaniu obliczeń komputerowych należy mieć świadomość o istniejących ograniczeniach. Wartość maszynowego epsilon zależy od systemu liczbowego i precyzji obliczeń, wraz ze wzrostem precyzji zmniejsza się maszynowy epsilon, trzeba o tym pamiętać i dobierać odpowiednią precyzję do danego systemu liczbowego (np. binarnego – stosowanego w komputerach) oraz do specyfiki problemu.

2.2. Zadanie 2

2.2.1. Błąd bezwzględny

Błąd bezwzględny obliczony może zostać za pomocą wzoru:

$$\Delta \sin x = |\sin x(1+\epsilon) - \sin x|.$$



Rys. 1. Wykres funkcji $\sin x$.

2.2.2. Błąd względny

Błąd względny obliczony może zostać za pomocą wzoru:

$$\frac{\Delta f(x)}{f(x)} = \frac{\Delta \sin x}{\sin x} = \frac{|\sin x(1+\epsilon) - \sin x|}{\sin x}.$$

2.2.3. Uwarunkowanie problemu

Uwarunkowanie jest miarą wrażliwości wyniku obliczeń na zmiany danych wejściowych. W uproszczeniu, mówi nam, jak bardzo wynik zmienia się w odpowiedzi na niewielkie zmiany danych wejściowych. Im mniejsze uwarunkowanie, tym bardziej stabilne są obliczenia.

Uwarunkowanie spełnia:

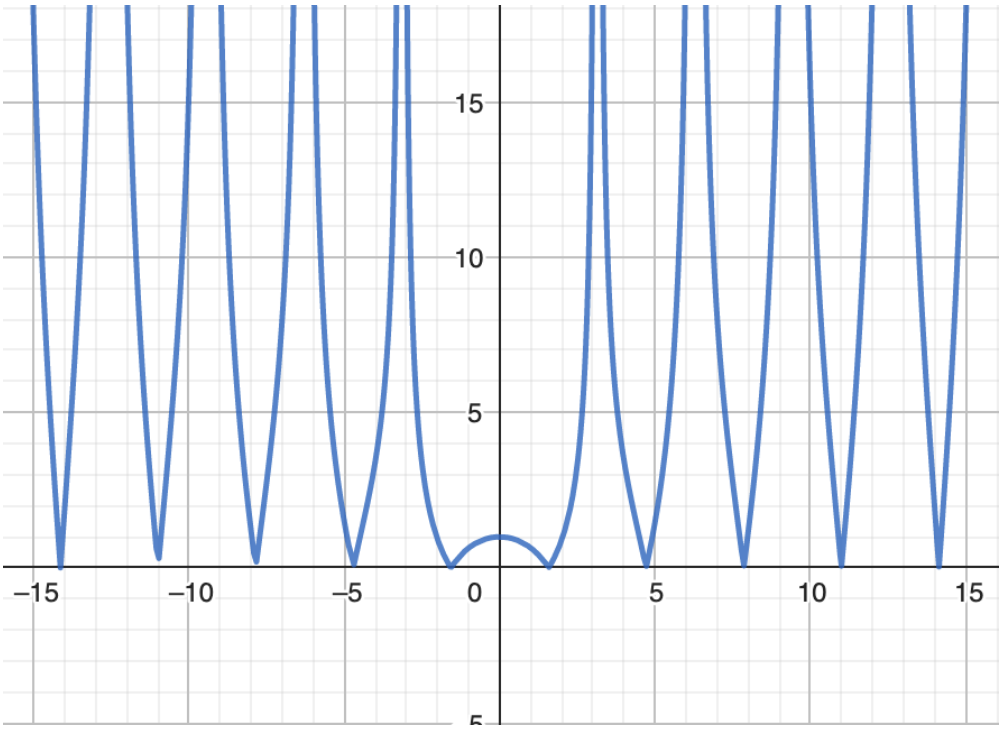
$$cond(f(x)) = cond(\sin x) = \left|\frac{xf'(x)}{f(x)}\right| = \left|\frac{x\cos x}{\sin x}\right| = |x\,ctg\,x|.$$

2.2.4. Czułość problemu dla wartości argumentów

Rozważany problem dotyczy błędów numerycznych związanych z obliczaniem wartości funkcji będzie czuły w miejscach, gdzie funkcja cotangens zmierza do nieskończoności

Istnieją pewne wartości argumentów, dla których funkcja ta ma osobliwości i nie jest określona, a mianowicie dla $x = k\pi$, gdzie $k \in \mathbb{Z}$, poza $k = 0$. Dlatego w tych miejscach błędy numeryczne będą szczególnie duże, ponieważ wartość funkcji cotangens zbliża się do nieskończoności. Wraz ze zbliżaniem się do tych punktów, błędy numeryczne będą się gwałtownie zwiększać, co może prowadzić do niedokładnych wyników.

Najlepiej uwarunkowany problem będzie natomiast w miejscach, gdzie funkcja cosinus zeruje się, czyli dla $x = k\pi + \frac{\pi}{2}$, gdzie $k \in \mathbb{Z}$ (wtedy zeruje się licznik, a mianownik jest niezerowy co oznacza, że wartość całego wyrażenia jest równa 0). W tych miejscach funkcja cotangens osiąga swoje skrajne wartości, dlatego ma ona dużą pochodną, a błędy numeryczne mogą się bardzo szybko kumulować. Jednocześnie jednak, ze względu na charakterystykę funkcji cotangens, zwiększenie błędu w tych miejscach nie prowadzi bezpośrednio do nieskończoności lub nieokreśloności, dlatego uwarunkowanie problemu jest lepsze niż w przypadku osobliwości funkcji cotangens.



Rys. 2. Wykres funkcji $f(x) = |x\ ctg\ x|$.

Funkcja zmierza do nieskończoności dla argumentów zbliżających się do całkowitych wielokrotności wartości π oprócz 0. W tych miejscach problem jest bardzo czuły.

2.2.5. Wnioski

Zgodnie z przewidywaniami, funkcja sinus jest najgorzej uwarunkowana w pobliżu swoich miejsc zerowych, ponieważ w tych punktach ma bardzo małe wartości, co oznacza, że nawet małe błędy w obliczeniach mogą prowadzić do dużych błędów względnych. Ponadto, funkcja sinus ma w tych punktach swoją największą pochodną, co oznacza, że błędy numeryczne w obliczeniach pochodnych mogą znacznie wpłynąć na wynik końcowy.

Z drugiej strony, funkcja sinus jest bardzo dobrze uwarunkowana w miejscach, gdzie osiąga swoje ekstrema, czyli wartości maksymalne i minimalne. W tych punktach funkcja ma największą wartość, co oznacza, że błędy numeryczne będą

miały mniejszy wpływ na wynik końcowy. Ponadto, w tych punktach pochodna funkcji sinus wynosi 0, co oznacza, że błędy numeryczne w obliczeniach pochodnych nie będą miały wpływu na wynik końcowy.

Obliczenia komputerowe obarczone są błędami obliczeniowymi. Dla każdego problemu należy ocenić jego uwarunkowanie by zachować odpowiednią precyzje obliczeń.

2.3. Zadanie 3

2.3.1. Błąd progresywny i błąd wsteczny

Błąd progresywny to moduł różnicy między wartością rzeczywistą a wartością uzyskaną przez przybliżenie lub eksperyment. Oznacza to, że błąd progresywny informuje nas o tym, jak bardzo nasze wyniki lub oszacowania odbiegają od prawdziwej wartości.

Błąd wsteczny (lub błąd odwrotny) to moduł różnicy między wartością argumentu, dla którego obliczona wartość funkcji jest równa wartości przybliżonej, a wartością używaną do uzyskania tej wartości przybliżonej. Oznacza to, że błąd wsteczny informuje nas o tym, jak bardzo nasze przybliżenie lub oszacowanie argumentu odbiega od rzeczywistej wartości argumentu.

Błąd progresywny: $|\hat{y} - y| = |y - \hat{y}|$.
Błąd wsteczny: $|\hat{x} - x| = |x - \hat{x}|$.

2.3.2. A

- $y = sin\, x$
- $\hat{y} = sin\, x \approx x$
- Błąd progresywny: $|sin\, x - x|$

- $x \in \{0{,}1;\ 0{,}5;\ 1\}$
- $\hat{x} = sin^{-1}\, \hat{y} = arcsin\, \hat{y} = arcsin\, x$
- Błąd wsteczny: $|arcsin\, \hat{y} - x| = |arcsin\, x - x|$

x	$y = sin\, x$	$\hat{y} = sin\, x \approx x$	*Błąd progresywny:* $\|sin\, x - x\|$	$\hat{x} = arcsin\, x$	*Błąd wsteczny:* $\|arcsin\, x - x\|$
0,1	0,0998334	0,1	0,0001666	0,1001674	0,0001674
0,5	0,4794255	0,5	0,0205745	0,5235987	0,0235987
1	0,8414709	1	0,1585290	1,5707963	0,5707963

Tabela 2. Wartości dla $sin(x) \approx x$

2.3.3. B

- $y = sin\, x$
- $\hat{y} = sin\, x \approx x - \frac{x^3}{6}$
- Błąd progresywny: $\left|sin\, x - (x - \frac{x^3}{6})\right| = \left|sin\, x - x + \frac{x^3}{6}\right|$

- $x \in \{0{,}1;\ 0{,}5;\ 1\}$
- $\hat{x} = sin^{-1}\, \hat{y} = arcsin\, \hat{y} = arcsin(x - \frac{x^3}{6})$
- Błąd wsteczny: $|arcsin\, \hat{y} - x| = \left|arcsin\, (x - \frac{x^3}{6}) - x\right|$

x	$y = sin\, x$	$\hat{y} = sin\, x \approx x - \frac{x^3}{6}$	*Błąd progresywny:* $\left\|sin\, x - x + \frac{x^3}{6}\right\|$	$\hat{x} = arcsin(x - \frac{x^3}{6})$	*Błąd wsteczny:* $\left\|arcsin\, (x - \frac{x^3}{6}) - x\right\|$
0,1	0,0998334	0.0998333	0,0000001	0.0999999	0,0000001
0,5	0,4794255	0.4791667	0.0002588	0.4997050	0,0002949
1	0,8414709	0.8333333	0.0081376	0.9851108	0.0148892

Tabela 3. Wartości dla $sin\, x \approx x - \frac{x^3}{6}$

2.3.4. Wnioski

Wykorzystanie większej liczby wyrazów szeregu Taylora zadanego dla funkcji sinus znacząco zwiększa dokładność otrzymanych wyników, zauważalne jest znaczne zmniejszenie błędów (zarówno progresywnego jak i wstecznego).

Gdyby zwiększano liczbę wykorzystanych wyrazów uzyskiwane byłyby wyniki bliższe wartościom dokładnym, wzrastałaby przy tym złożoność pamięciowa i czas obliczeń.

Błędy rosną zgodnie z przewidywaniami z uwarunkowania problemu z poprzedniego zadania.

2.4. Zadanie 4

2.4.1. UFL systemu

Poziom UFL (ang. Underflow Level) w systemie liczbowym zmiennoprzecinkowym to najmniejsza dodatnia liczba, która może być zapisana w tym systemie. W przypadku, gdy wynik operacji matematycznych jest mniejszy od poziomu UFL, następuje zjawisko denormalizacji - liczba ta musi być zapisana w sposób szczególny, co prowadzi do utraty dokładności obliczeń. Oznacza to, że UFL jest najmniejszą możliwą wartością reprezentowaną w danym systemie zmiennoprzecinkowym, a różnice mniejsze niż UFL między dwoma liczbami są traktowane jako błąd numeryczny i są zwykle wynikiem działań zaokrąglania.

Poziom UFL można wyrazić wzorem:

$$UFL = \beta^L,$$

gdzie:

- β to podstawa systemu liczbowego,
- L to liczba bitów przeznaczona na zapis wykładnika.

W podanym przypadku poziom UFL wynosi:

$$UFL = 10^{-98}.$$

2.4.2. Wynik działania

W zadaniu obliczana jest różnica bardzo małych liczb zmiennoprzecinkowych o bliskich sobie wartościach (liczby mają ten sam wykładnik i niewiele różniącą się mantysę) powoduje to, że mantysa wyniku jest znacznie zdenormalizowana, a renormalizacja wiąże się w takim wypadku z wprowadzeniem do reprezentacji wyniku dużej liczby nieznaczących zer, które będą pamiętane w komputerze na końcu mantysy.

Wykonując działanie odejmowania otrzymujemy:

$$\begin{gathered} x - y = \\ 6{,}87 \cdot 10^{-97} - 6{,}81 \cdot 10^{-97} = \\ (6{,}87 - 6{,}81) \cdot 10^{-97} = \\ 0{,}06 \cdot 10^{-97}. \end{gathered}$$

Analizując otrzymany wynik otrzymujemy:

$$0{,}06 \cdot 10^{-97} = 6 \cdot 10^{-99} < 6 \cdot 10^{-98},$$
$$x - y < UFL,$$

zatem wynik tej operacji w tym systemie będzie równy:

$$x - y = 0.$$

2.4.3. Wnioski

W przypadku, gdy system operuje na małych liczbach i dąży się do jak najdokładniejszych wyników, wartość parametru L powinna być jak najmniejsza, aby UFL była jak najmniejsza i uniknąć zjawiska denormalizacji.

3. Bibliografia

1. Wykłady dr inż. Katarzyny Rycerz z przedmiotu Metody obliczeniowe w nauce i technice na czwartym semestrze kierunku Informatyka w AGH w Krakowie
2. Wykresy kreślono za pomocą internetowego programu GeoGebra: https://www.geogebra.org/calculator
3. Obliczenia wykonywano za pomocą internetowego programu WolframAlpha: https://www.wolframalpha.com/ oraz programu Microsoft Excel: https://www.microsoft.com/pl-pl/microsoft-365/excel
4. Programy napisane zostały w języku Python w wersji 3.11: https://www.python.org/
5. Wykorzystano bibliotekę NumPy dla języka Python w wersji 1.24: https://numpy.org/doc/stable/index.html
6. https://en.wikipedia.org/wiki/Machine_epsilon
7. https://en.wikipedia.org/wiki/IEEE_754
8. https://irem.univ-reunion.fr/IMG/pdf/ieee-754-2008.pdf