Łukasz Kluza, 05.03.2024
„Laboratorium” 01

Arytmetyka Komputerowa

Treści zadań

Zadanie 1.
Znaleźć *"maszynowe epsilon"*, czyli najmniejszą liczbę *a*, taką że *a*+1>1

Zadanie 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:

- Ocenić błąd bezwzględny przy ewaluacji *sin(x)*
- Ocenić błąd względny przy ewaluacji *sin(x)*
- Ocenić uwarunkowanie dla tego problemu
- Dla jakich wartości argumentu x problem jest bardzo czuły ?

Zadanie 3.

Funkcja sinus zadana jest nieskończonym ciągiem

$$sin(x) = x - \frac{x^3}{3!} + \frac{x^5}{5!} - \frac{x^7}{7!} + \dots$$

- Jakie są błędy progresywny i wsteczny jeśli przybliżamy funkcję sinus biorąc tylko pierwszy człon rozwinięcią, tj. $sin(x) \approx x$, dla x = 0.1, 0.5 i 1.0 ?
- Jakie są błędy progresywny i wsteczny jeśli przybliżamy funkcję sinus biorąc pierwsze dwa człony rozwinięcią, tj. $sin(x) \approx x - \frac{x^3}{3!}$, dla x = 0.1, 0.5 i 1.0 ?

Zadanie 4.

Zakładamy że mamy znormalizowany system zmiennoprzecinkowy z β = 10, p = 3, L = -98

- Jaka jest wartość poziomu UFL (underflow) dla takiego systemu ?
- Jeśli x = 6.87 x 10^(-97) i y = 6.81 x 10^(-97), jaki jest wynik operacji x – y ?

Rozwiązania

Zadanie 1.
Aby wyznaczyć *maszynowe epsilon* wystarczy zauważyć że musi on mieć taki sam wykładnik jak liczba 1.0 oraz jak najmniejszą mantysę, też równą 1.

Dla typu *double*, który ma 53 bity na mantysę, możemy to oszacować na $2^{-M}, (M \rightarrow mantysa)$. Jednak, ponieważ w standardzie IEEE 754 pierwszy bit mantysy zawsze jest równy 1, więc w naszym przypadku *maszynowy epsilon* wyniesie: $\epsilon = 2^{-52}, (2^{1-M})$.

Dla przykładu w języku programowanie *python* łatwo możemy sprawdzć ile wynosi *"maszynowe epsilon"* użwyając wbudwoanych fuknkcji:

```
print(sys.float_info.epsilon)
#2.220446049250313e-16
```

Lub możemy też zdefiniować własną funkcję:

```
def show_info():
    number = 1.0
    ONE = 1.0
    while number + ONE > ONE:
        epsilon = number
        number /= 2

    print(f'Maszynowy epsilon: {epsilon}')

show_info()
# Maszynowy epsilon: 2.220446049250313e-16
```

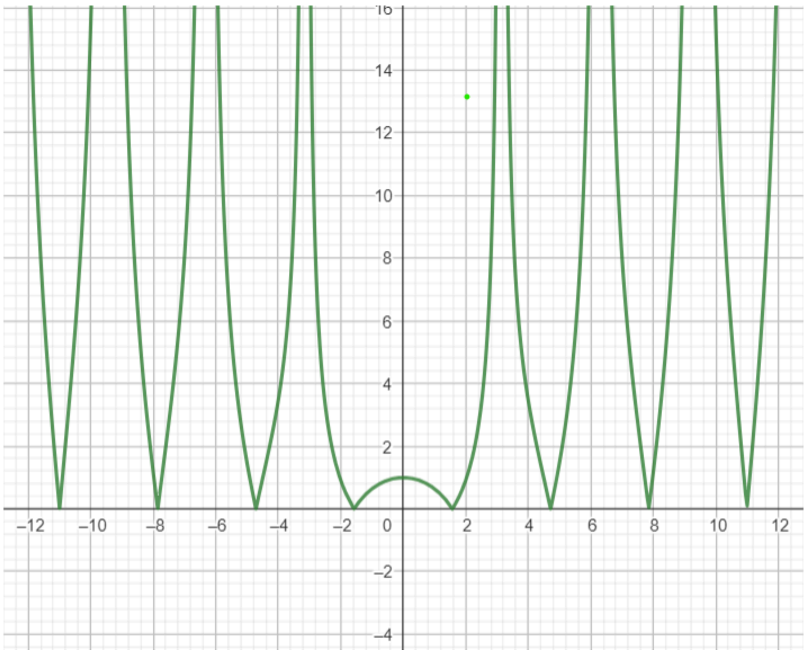
Jak widać powyższe wyniki uzyskane algebraicznie oraz numerycznie się pokrywają.

Zadanie 2.

- Błąd względny
  $\Delta f(x) = |sin(x) \cdot (1 + \epsilon) - sin(x)|$

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

- Uwarunkowanie i czułość
  $cond \approx |\frac{x \cdot f'(x)}{f(x)}| = |\frac{x \cdot cos(x)}{sin(x)}| = |x \cdot cot(x)|$



Problem jest najlepiej uwarunkowany w miejscach kiedy funkcja $y = |x \cdot cot(x)|$ się zeruje, czyli $x = \frac{\pi}{2} + k\pi, k \in \mathbb{Z} - \{0\}$.
Natomiast problen ten jest czuły (tj. najgorzej uwarunkowany), kiedy ta funkja ($y = |x \cdot cot(x)|$) zmierza do nieskończoności, a wiec dla $x = k\pi, k \in \mathbb{Z} - \{0\}$

Zadanie 3.

Badanie własności funkcji *sin(x)* wyprowadzonej za pomocą wzoru *Taylora*

$$sin(x) = x - \frac{x^3}{3!} + \frac{x^5}{5!} - \frac{x^7}{7!} + \ldots$$

Wzory:
y = f(x), gdzie $f : \mathbb{R} \rightarrow \mathbb{R}$
Błąd progresywny: $\Delta y = |\hat{y} - y|$
Błąd wsteczny: $\Delta x = |\hat{x} - x|$

- Błąd progresywny i wsteczny dla przybliżonego wzorem: *sin(x)* ≈ *x* gdzie $x = 0.1, 0.5, 1.0$.

$y = sin(x)$
$\hat{y} = x, \hat{x} = arcsin(\hat{y})$
Błąd progresywny: $\Delta y = |\hat{y} - y| = |x - sin(x)|$
Błąd wsteczny: $\Delta x = |\hat{x} - x| = |arcsin(\hat{y}) - x| = |arcsin(x) - x|$

Wyniki/ Dane	$\hat{y} = x$	$\hat{x} = arcsin(\hat{y})$	$sin(x)$	**Błąd progresywny**	**Błąd wsteczny**
x=0.1	*0.1*	$arcsin(0.1) = 0.1001674$	0.0998334	0.0001666	0.0001674
x=0.5	*0.5*	$arcsin(0.5) = 0.5235987$	0.4794255	0.0205745	0.0235987
x=1.0	*1.0*	$arcsin(1.0) = 1.5707963$	0.8414709	0.1585291	0.5707963

- Błąd progresywny i wsteczny dla $sin(x)$ przybliżonego wzorem: $sin(x) \approx x - \frac{x^3}{3!}$ gdzie $x = 0.1, 0.5, 1.0$.

$y = sin(x)$
$\hat{y} = x - \frac{x^3}{3!}, \hat{x} = arcsin(\hat{y})$
Błąd progresywny: $\Delta y = |\hat{y} - y| = |(x - \frac{x^3}{3!}) - sin(x)|$
Błąd wsteczny: $\Delta x = |\hat{x} - x| = |arcsin(\hat{y}) - x| = |arcsin(x - \frac{x^3}{3!}) - x|$

Wyniki/ Dane	$\hat{y} = x - \frac{x^3}{3!}$	$\hat{x} = arcsin(\hat{y})$	$sin(x)$	**Błąd progresywny**	**Błąd wsteczny**
x=0.1	0.0998333	$arcsin(0.0998333) = 0.0999998$	0.0998334	0.0000001	0.0000002
x=0.5	0.4791666	$arcsin(0.4791666) = 0.4997049$	0.4794255	0.0002589	0.0002951
x=1.0	0.8333333	$arcsin(0.8333333) = 0.9851107$	0.8414709	0.0081376	0.0148893

Zadanie 4.

Dane: Znormalizowany system zmienno przecinowy z: $\beta = 10, p = 3, L = -98$

- Underflow tego systemu:

$UFL = \beta^L$, czyli: $UFL = 10^{-98}$

- Jaki jest wynik operacji $x - y$, jeśli $x = 6.87 \cdot 10^{-97}$ a $y = 6.81 \cdot 10^{-97}$

$x - y = 6.87 \cdot 10^{-97} - 6.81 \cdot 10^{-97} = 0.06 \cdot 10^{-98} = 6.0 \cdot 10^{-99} < 10^{-98} (UFL)$

Ponieważ wartość takiej $x - y$ jest mniejsza niż wartość *UFL*, więc wynik będzie równy *0*.

Wnioski

Analizując powyższe, bardzo łatwo można zauważyć, że w informatyce konieczne jest zachowanie szczególnej ostrożności podczas operacji na liczbach zmiennoprzecinkowych. Te liczby generują wiele problemów, zwłaszcza gdy są używane do reprezentowania bardzo małych wartości lub gdy odejmujemy od siebie dwie bardzo bliskie sobie liczby.

Jednym z głównych problemów jest utrata precyzji, która może wystąpić przy operacjach arytmetycznych na liczbach zmiennoprzecinkowych. Na przykład, gdy dodajemy do siebie bardzo małą wartość do bardzo dużej, wynik może być zaokrąglany do wartości większej niż oczekiwano, co prowadzi do błędnego wyniku.

Aby uniknąć tych problemów, należy brać pod uwagę kilka kwestii:

- *Unikanie operacji na liczbach o zbliżonych wartościach:* W miarę możliwości, należy unikać odejmowania dwóch bardzo bliskich siebie liczb zmiennoprzecinkowych, ponieważ może to prowadzić do utraty precyzji.
- *Używanie większej precyzji:* W niektórych przypadkach, zwłaszcza gdy operujemy na bardzo małych lub bardzo dużych wartościach, korzystanie z większej precyzji zmiennoprzecinkowej może pomóc uniknąć problemów związanych z utratą precyzji.

Dbanie o te kwestie pomoże nam uniknąc wielu problemów związanych z operacjami na liczbach zmiennoprzecinkowych i zapewni dokładnejsze i wiarygodniejsze wyniki obliczeń.

Biografia:

- https://pl.wikipedia.org/wiki/IEEE_754

- Prof. Michael T. Heath, Scientific Computing: An Introductory SurveyChapter 1 – Scientific Computing
- dr. inż. Katarzyna Rycerz, Metody Obliczeniowe w Nauce i Technice - wykład