Sprawozdanie z Laboratorium
Obliczenia Naukowe - Lista 1

Wojciech Typer

12 października 2025

Zadanie 1

0.1 Epsilon maszynowy (*macheps*)

Epsilonem maszynowym *macheps* nazywamy najmniejszą liczbę dodatnią taką, że w arytmetyce zmiennoprzecinkowej zachodzi $1.0 + \text{macheps} > 1.0$. Jest to miara precyzji obliczeń, która określa odległość od liczby 1.0 do następnej reprezentowalnej liczby maszynowej. Im mniejszy epsilon, tym większa precyzja arytmetyki, co jest bezpośrednio związane z liczbą bitów przeznaczonych na mantysę w danym typie zmiennoprzecinkowym.

Poniżej przedstawiono porównanie wartości *macheps* uzyskanych iteracyjnie, wartości zwracanych przez funkcję `eps()` w Julii oraz wartości zdefiniowanych w pliku nagłówkowym `float.h` kompilatora C (GCC 13).

Tabela 1: Porównanie wartości epsilona maszynowego.

Typ danych	Wartość z `float.h` (GCC)	Wartość z `eps(T)` (Julia)	Wartość wyznaczona iteracyjnie
`Float16`	$9.7656e{-4}$	$9.77e{-4}$	$9.77e{-4}$
`Float32`	$1.192\,092\,90e{-7}$	$1.192\,092\,9e{-7}$	$1.192\,092\,9e{-7}$
`Float64`	$2.220\,446\,049\,250\,313\,1e{-16}$	$2.220\,446\,049\,250\,313e{-16}$	$2.220\,446\,049\,250\,313e{-16}$

Jak widać w tabeli 1, wartości uzyskane eksperymentalnie są zgodne z wartościami referencyjnymi.

0.2 Najmniejsza dodatnia liczba maszynowa (*eta*)

Liczba *eta* (η) to najmniejsza dodatnia wartość, jaką można reprezentować w danym standardzie zmiennoprzecinkowym. Wartość ta jest związana z liczbami subnormalnymi (denormalizowanymi), które pozwalają na płynne "wypełnienie"luki między zerem a najmniejszą dodatnią liczbą znormalizowaną.

- **Związek z MIN_{sub}**: Liczba *eta* jest tożsama z MIN_{sub}, czyli najmniejszą możliwą do reprezentowania dodatnią liczbą subnormalną. W języku Julia wartość tę można uzyskać za pomocą funkcji `nextfloat(T(0.0))`.
- **Związek z MIN_{nor}**: Funkcja `floatmin(T)` zwraca najmniejszą dodatnią liczbę **znormalizowaną**, znaną jako MIN_{nor}. Jest to wartość większa od *eta*.

Wartości *eta* wyznaczone iteracyjnie (poprzez dzielenie 1.0 przez 2 aż do uzyskania 0.0) są zgodne z wynikami funkcji `nextfloat(T(0.0))`. Porównanie tych wartości przedstawiono w tabeli 2.

Tabela 2: Porównanie wartości *eta* (η).

Typ danych	**Wartość z `nextfloat(T(0.0))`**	**Wartość wyznaczona iteracyjnie**
`Float16`	$6.0e{-8}$	$6.0e{-8}$
`Float32`	$1.4e{-45}$	$1.0e{-45}$
`Float64`	$5.0e{-324}$	$5.0e{-324}$

0.3 Największa wartość skończona (*MAX*)

Liczba MAX to największa skończona wartość, jaką można zapisać w danym typie zmiennoprzecinkowym. Próba reprezentacji liczby większej niż MAX prowadzi do uzyskania wartości nieskończonej (`Inf`). Doświadczalne wyznaczenie tej wartości polega na iteracyjnym mnożeniu liczby przez 2, aż do momentu, gdy stanie się ona nieskończona, a następnie cofnięciu ostatniej operacji.

Tabela 3: Porównanie maksymalnych wartości zmiennoprzecinkowych.

Typ danych	**Wartość z `float.h` (GCC)**	**Wartość wyznaczona iteracyjnie**
`Float16`	$6.550\,40e4$	$6.55e4$
`Float32`	$3.402\,823\,47e38$	$3.402\,823\,5e38$
`Float64`	$1.797\,693\,134\,862\,315\,7e308$	$1.797\,693\,134\,862\,315\,7e308$

Zadanie 2

W tabeli poniżej znajdują się wartości epsilona maszynowego, obliczone metodą Kahana i te, zwrócone przez funkcję `eps()` w Julii.

Typ danych	**Wartość z metody Kahana**	**Wartość z `eps(T)` (Julia)**
`Float16`	$-9.77e{-4}$	$9.77e{-4}$
`Float32`	$1.192\,092\,9e{-7}$	$1.192\,092\,9e{-7}$
`Float64`	$-2.220\,446\,049\,250\,313e{-16}$	$2.220\,446\,049\,250\,313e{-16}$

Zadanie 3

W zadaniu przeanalizowano rozkład liczb zmiennoprzecinkowych w arytmetyce `double` w różnych przedziałach. Gęstość rozmieszczenia tych liczb, czyli odległość między dwiema kolejnymi reprezentacjami, zależy od przedziału, w którym się znajdujemy. Sprawdzono to eksperymentalnie dla trzech przypadków.

- **Przedział** $[1, 2]$**:** W arytmetyce `double`, liczby zmiennoprzecinkowe są rozmieszczone równomiernie na przedziale $[1, 2]$ z krokiem równym $\delta = 2^{-52}$. Oznacza to, że każda kolejna liczba na tym przedziale różni się od poprzedniej o dokładnie δ. Sprawdzono to eksperymentalnie: 1000-krotnie generując losową liczbę z tego przedziału i wyznaczając kolejną liczbę maszynową za pomocą funkcji `nextfloat()`, różnica między nimi zawsze była równa δ.

- **Przedział** $[0.5, 1]$**:** Dla tego przedziału krok wynosi $\delta = 2^{-53}$. Każda liczba może być przedstawiona jako: $x = 1 + k \cdot \delta$, gdzie k jest liczbą całkowitą, a $\delta = 2^{-53}$.

- **Przedział** $[2, 4]$**:** W tym przypadku krok jest większy i wynosi $\delta = 2^{-51}$. Dla tego przedziału każda liczba może być przedstawiona jako: $x = 2 + k \cdot \delta$, gdzie $\delta = 2^{-51}$.

Powyższe eksperymenty potwierdzają, że w arytmetyce zmiennoprzecinkowej liczby są rozmieszczone gęściej bliżej zera i rzadziej w miarę oddalania się od niego.

Zadanie 4

- Liczba znaleziona eksperymentalnie: 1.7935706239891005 - generujemy losową liczbę z przedziału $[1.0, 2.0]$ i sprawdzamy, czy $x \cdot \frac{1}{x} \neq 1$.
- Najmniejsza liczba z przedziału $[1.0, 2.0]$, dla której zachodzi $x \cdot \frac{1}{x} \neq 1$ to 1.000000057228997. Iterujemy od 1.0 w górę, aż do momentu, gdy warunek będzie spełniony.

Zadanie 5

W zadaniu porównano cztery różne metody obliczania iloczynu skalarnego wektorów, aby zbadać ich stabilność numeryczną i wpływ na błędy zaokrągleń. Wyniki dla precyzji 32-bitowej (`Float32`) oraz 64-bitowej (`Float64`) przedstawiono w tabeli 0.3.

Liczba bitów	**Metoda**	**Wynik**
Precyzja 32-bitowa (`Float32`)		
	Metoda 1	-0.4999443
	Metoda 2	-0.4543457
	Metoda 3	-0.5
	Metoda 4	-0.5
Precyzja 64-bitowa (`Float64`)		
	Metoda 1	1.0251881368296672e-10
	Metoda 2	-1.5643308870494366e-10
	Metoda 3	0.0
	Metoda 4	0.0

Zadanie 6

Wartość x	Wynik metody 1		Wynik metody 2	
8^{-1}	7.782 218 537 318 641 4	$e-3$	7.782 218 537 318 706 5	$e-3$
8^{-2}	1.220 628 628 286 757 3	$e-4$	1.220 628 628 287 590 1	$e-4$
8^{-3}	1.907 346 813 823 096 5	$e-6$	1.907 346 813 826 566	$e-6$
8^{-4}	2.980 232 194 360 610 3	$e-8$	2.980 232 194 360 611 6	$e-8$
8^{-5}	4.656 612 873 077 393	$e-10$	4.656 612 871 993 190 4	$e-10$
8^{-6}	7.275 957 614 183 426	$e-12$	7.275 957 614 156 956	$e-12$
8^{-7}	1.136 868 377 216 160 3	$e-13$	1.136 868 377 216 095 7	$e-13$
8^{-8}	1.776 356 839 400 250 5	$e-15$	1.776 356 839 400 248 9	$e-15$
8^{-9}	0.0		2.775 557 561 562 891 4	$e-17$
8^{-10}	0.0		4.336 808 689 942 018	$e-19$

Metoda nr 2 daje bardziej precyzyjne wyniki, ponieważ w jej formule nie ma żadnych operacji, które prowadziłyby do utraty precyzji

Zadanie 7