Obliczenia Naukowe - Labolatoria 1

Krzysztof Zając

October 23, 2025

1 Zadanie 1 - rozpoznanie arytmetyki

1.1 Opis problemu

W arytmetyce zmiennoprzecinkowej próbujemy przedstawić liczbę rzeczywistą w komputerze, przez próbę zapisania jej w skończonej liczbie bitów istnieją poniższe ograniczenia: najmniejsza możliwa liczba dodatnia (MIN_{NOR}), największa możliwa liczba (MAX_{NOR}) i epsilon maszynowy (ϵ), czyli miara precyzji, pokazująca zagęszczenie liczb wokół jedynki.

MIN_{NOR} to najmniejsza liczba, którą można zapisać z pełną, standardową precyzją mantysy, dzięki niejawnemu bitowi '1', tzw. liczba znormalizowana . Poniżej tej wartości istnieje jednak "luka" do zera, którą wypełniają liczby zdenormalizowane (subnormalne, takie, dla których wykładnik to same zera), aby uniknąć gwałtownego zaokrąglenia do zera. MIN_{SUB} jest najmniejszą z tych liczb zdenormalizowanych i jednocześnie absolutnie najmniejszą dodatnią liczbą, jaką można zapisać w danym formacie.

Chcemy sprawdzić wartości MIN_{SUB} (zwanym w treści *eta*), MAX_{NOR} i ϵ metodą iteracyjną.

1.2 Rozwiązanie

Można sprawdzić wartości poszególnych wartości przez odpowiednie iteracyjne mnożenie przez 2 lub $\frac{1}{2}$.

1.3 Wyniki

Table 1: Porównanie Epsilona Maszynowego (ϵ)

Typ	Wartość iteracyjna (ϵ)	Wartość wbudowana (eps())
Float16	0.000977	0.000977
Float32	1.1920929e-7	1.1920929e-7
Float64	2.220446049250313e-16	2.220446049250313e-16

Table 2: Porównanie najmniejszej liczby dodatniej (η/MIN_sub)

Typ	Wartość iteracyjna (η)	Wartość wbudowana (nextfloat(0.0))
Float16	6.0e-8	6.0e-8
Float32	1.0e-45	1.0e-45
Float64	5.0e-324	5.0e-324

Table 3: Porównanie maksymalnej liczby (MAX)

Typ	Wartość iteracyjna (MAX)	Wartość wbudowana (floatmax())
Float16	6.55e4	6.55e4
Float32	3.4028235e38	3.4028235e38
Float64	1.7976931348623157e308	1.7976931348623157e308

Tablice wygenerowane przez Gemini Pro, na podstawie danych zwróconych przez program `ex1.jl`.

1.4 Wnioski

Nasza metoda iteracyjna jest poprawna i osiąga te same wartości, które mamy w Julii i w C.

1. Liczba **macheps** (ϵ_m) jest **miarą precyzji arytmetyki (ϵ)**, ponieważ określa największy błąd względny zaokrąglenia i jest to najmniejsza dodatnia liczba zmiennoprzecinkowa x taka, że $\mathrm{fl}(1+x) > 1$.
2. Liczba **eta** (η) jest **synonimem** (innym oznaczeniem) dla liczby **MINsub**, która jest najmniejszą dodatnią (zdenormalizowaną) liczbą zmiennoprzecinkową możliwą do zapisania w danym formacie.
3. Funkcje floatmin(Float32) i floatmin(Float64) zwracają odpowiednio MINnor_{32} (2^{-126}) i MINnor_{64} (2^{-1022}), czyli **najmniejsze dodatnie znormalizowane liczby maszynowe** (MINnor) w danym formacie.

2 Zadanie 2 - wzór Kahan'a

2.1 Opis problemu

Kahan postawił tezę, że można sprawdzić wartość macheps za pomocą wzoru: $3 * (\frac{4}{3} - 1) - 1$. Chcemy to sprawdzić ekperymentalnie dla typów `Float16`, `Float32` i `Float64`.

2.2 Rozwiązanie

Obliczono wartość wyrażenia $3 \times (\frac{4}{3} - 1) - 1$ dla każdego z typów. Działanie $\frac{4}{3}$ jest operacją, która w systemie binarnym ma nieskończone rozwinięcie, co wymusza zaokrąglenie.

2.3 Wyniki

Table 4: Wyniki wzoru Kahana

Typ	Wartość wzoru Kahana	Wartość `eps(T)`
Float16	0.0009766	0.0009766
Float32	1.1920929e-7	1.1920929e-7
Float64	2.220446049250313e-16	2.220446049250313e-16

2.4 Wnioski

Wzór Kahana jest poprawny i zwraca wartość epsilona maszynowego dla badanych typów.

3 Zadanie 3 - gęstość liczb

3.1 Opis problemu

Eksperymentalne sprawdzenie rozmieszczenia liczb `Float64` w przedziałach $[1, 2]$, $[\frac{1}{2}, 1]$ oraz $[2, 4]$. Weryfikacja tezy o stałym kroku $\delta = 2^{-52}$ w $[1, 2]$.

3.2 Rozwiązanie

Krok δ (odstęp) zależy od wykładnika E jako $\delta = 2^{-52} \times 2^{E-1023}$. W przedziałach $[2^n, 2^{n+1})$ wykładnik jest stały, a więc krok δ też jest stały. Do weryfikacji użyto funkcji `eps(x)` (zwracającej δ dla przedziału, do którego należy x) oraz `bitstring(x)` do obserwacji wykładnika.

3.3 Wyniki

- **Przedział [1, 2]**: Wykładnik $E - 1023 = 0$. Krok $\delta = \text{eps}(1.0) = 2^{-52} \times 2^0 = 2^{-52}$. Reprezentacja: $x_k = 1 + k \cdot 2^{-52}$.
- **Przedział [0.5, 1]**: Wykładnik $E - 1023 = -1$. Krok $\delta' = \text{eps}(0.5) = 2^{-52} \times 2^{-1} = 2^{-53}$. Reprezentacja: $x_k = 0.5 + k \cdot 2^{-53}$.
- **Przedział [2, 4]**: Wykładnik $E - 1023 = 1$. Krok $\delta'' = \text{eps}(2.0) = 2^{-52} \times 2^1 = 2^{-51}$. Reprezentacja: $x_k = 2 + k \cdot 2^{-51}$.

Wartości `eps()` były zgodne z teoretycznymi 2^{-52}, 2^{-53} i 2^{-51}. Analiza `bitstring` potwierdziła stałość wykładników wewnątrz przedziałów (odp. 1023, 1022, 1024).

3.4 Wnioski

Liczby są rozmieszczone równomiernie tylko w obrębie przedziałów o stałym wykładniku $[2^n, 2^{n+1})$. Krok (odstęp) podwaja się przy każdym przekroczeniu potęgi dwójki. W $[0.5, 1]$ gęstość jest 2x większa niż w $[1, 2]$, a w $[2, 4]$ jest 2x mniejsza.

4 Zadanie 4 - błąd odwrotności

4.1 Opis problemu

Znalezienie liczby $x \in (1, 2)$ w `Float64` takiej, że $\text{fl}(x \times \text{fl}(1/x)) \neq 1$, oraz znalezienie najmniejszej takiej liczby.

4.2 Rozwiązanie

Iterowano po liczbach $x = \text{nextfloat}(1.0)$ w górę, sprawdzając warunek $x \times (1/x) \neq 1.0$.

4.3 Wyniki

Najmniejszą liczbą `Float64` w przedziale $(1, 2)$ niespełniającą warunku jest $x = \text{nextfloat}(1.0) = 1.0 + 2^{-52}$. Dla tej wartości x, $fl(1/x)$ jest obliczane z błędem zaokrąglenia, a następnie $fl(x \times fl(1/x))$ wprowadza kolejny błąd zaokrąglenia. Wynik to $\text{prevfloat}(1.0) = 1.0 - 2^{-53}$.

4.4 Wnioski

Operacja $1/x$ dla $x = 1.0 + 2^{-52}$ nie jest dokładnie reprezentowalna i jej zaokrąglenie, a następnie ponowne pomnożenie przez x i kolejne zaokrąglenie, prowadzi do utraty dokładności, uniemożliwiając powrót do wartości 1.0.

5 Zadanie 5 - iloczyn skalarny

5.1 Opis problemu

Obliczenie iloczynu skalarnego $S = \sum x_i y_i$ dla $n = 5$ na cztery sposoby (w przód, w tył, sortowanie dodatnich/ujemnych od najw. do najmn., sortowanie od najmn. do najw.) dla typów `Float32` i `Float64`. Porównanie z dokładną wartością $S_{dokl} = -1.00657107 \times 10^{-11}$.

5.2 Rozwiązanie

Zaimplementowano cztery algorytmy sumowania iloczynów $x_i y_i$ dla podanych wektorów i obu precyzji.

5.3 Wyniki

Wartości iloczynów $x_i y_i$ mają bardzo różne rzędy wielkości. Dwie wartości są duże i przeciwnego znaku ($x_2 y_2 \approx -2.7 \times 10^6$ i $x_4 y_4 \approx +2.7 \times 10^6$), co stwarza ryzyko katastrofalnej redukcji cyfr.

Table 5: Wyniki obliczeń iloczynu skalarnego

Metoda (dla $n = 5$)	Wynik `Float32`	Wynik `Float64`
(a) "w przód"	-0.4999443	$1.0251881368296672 \times 10^{-10}$
(b) "w tył"	-0.4543457	$-1.5643308870494366 \times 10^{-10}$
(c) Sort (najw. → najmn.)	-0.5	0.0
(d) Sort (najmn. → najw.)	-0.5	0.0
Wartość dokładna	$-1.00657107 \times 10^{-11}$	

5.4 Wnioski

1. **Float32**: Precyzja jest całkowicie niewystarczająca. Żaden wynik nie jest nawet bliski poprawnemu. Różne wyniki dla metody (a) i (b) wyraźnie pokazują, że dodawanie zmiennoprzecinkowe nie jest łączne (asocjatywne).
2. **Float64**: Pomimo znacznie wyższej precyzji, **każda** z metod dała błędny wynik. Jest to spowodowane odejmowaniem dwóch bardzo bliskich sobie (co do modułu) liczb $x_2 y_2$ i $x_4 y_4$.
3. Metody (a) i (b) (w przód / w tył) dają różne błędne wyniki, ponownie pokazując brak łączności.
4. Metody (c) i (d) (sortowanie) doprowadziły do całkowitego wyzerowania wyniku. W tym przypadku algorytm sumowania (prawdopodobnie `sum()` w Julii) lub kolejność operacji spowodowały, że duże liczby o przeciwnych znakach zredukowały się do zera, gubiąc całkowicie mniejsze składniki (w tym ten, który niósł poprawny wynik 10^{-11}).
5. Jest to klasyczny przykład **katastrofalnej redukcji cyfr (anulowania)** i niestabilności numerycznej algorytmu sumowania. Żadna ze standardowych metod nie poradziła sobie z tym konkretnym zestawem danych.

6 Zadanie 6 - błąd odejmowania

6.1 Opis problemu

Porównanie wyników obliczeń dwóch matematycznie równoważnych funkcji $f(x) = \sqrt{x^2+1} - 1$ i $g(x) = \frac{x^2}{\sqrt{x^2+1}+1}$ dla $x = 8^{-n}$, $n = 1, 2, \ldots$ w arytmetyce `Float64`.

6.2 Rozwiązanie

Obliczono wartości $f(x)$ i $g(x)$ dla malejących $x \to 0$.

6.3 Wyniki

Dla $n = 1$ do $n = 8$, obie funkcje dają niemal identyczne wyniki.

- Dla $n = 8$ ($x \approx 5.96 \times 10^{-8}$): $f(x) \approx 1.7763... \times 10^{-15}$ i $g(x) \approx 1.7763... \times 10^{-15}$.
- Dla $n = 9$ ($x \approx 7.45 \times 10^{-9}$): $f(x) = 0.0$, natomiast $g(x) \approx 2.7755... \times 10^{-17}$.
- Dla $n \geq 9$: $f(x)$ zwraca 0.0, podczas gdy $g(x)$ nadal zwraca poprawne, malejące, niezerowe wartości.

6.4 Wnioski

Wzór $f(x)$ cierpi na katastrofalną redukcję cyfr (błąd anulowania). Gdy $x \to 0$, $\sqrt{x^2+1}$ jest bardzo bliskie 1. Dla $n = 9$, $x^2 \approx 5.55 \times 10^{-17}$. Ta wartość jest mniejsza niż epsilon maszynowy ($\epsilon \approx 2.22 \times 10^{-16}$). W rezultacie $\mathrm{fl}(x^2 + 1)$ zwraca 1.0, co prowadzi do $\sqrt{1.0} - 1 = 0$. Wzór $g(x)$ (uzyskany przez przekształcenie $f(x)$ z użyciem sprzężenia) jest numerycznie stabilny, ponieważ zastępuje problematyczne odejmowanie stabilnym dodawaniem w mianowniku.

7 Zadanie 7 - błąd pochodnej

7.1 Opis problemu

Obliczenie przybliżonej pochodnej $f(x) = \sin(x) + \cos(3x)$ w $x_0 = 1$ za pomocą ilorazu różnicowego $\tilde{f}'(x_0) = \frac{f(x_0+h)-f(x_0)}{h}$ dla $h = 2^{-n}$ ($n = 0, \ldots, 54$). Analiza błędu $\left|f'(1) - \tilde{f}'(1)\right|$.

7.2 Rozwiązanie

Dokładna pochodna $f'(x) = \cos(x) - 3\sin(3x)$, co dla $x_0 = 1$ daje $f'(1) \approx 0.11694228168853815$. Obliczono $\tilde{f}'(1)$ i błąd dla kolejnych n.

7.3 Wyniki

- Dla $n \approx 0$ do $n \approx 27$: Błąd maleje proporcjonalnie do h (o $\approx 1/2$ co krok). Dominuje błąd obcięcia (błąd metody) $\mathcal{O}(h)$.
- Dla $n = 28$ ($h \approx 3.72 \times 10^{-9}$): Osiągnięto minimalny błąd rzędu $\approx 4.80 \times 10^{-9}$.
- Dla $n > 28$ (małe h): Błąd zaczyna gwałtownie rosnąć (oscylując). Dominuje błąd zaokrąglenia. W liczniku $f(x_0+h) - f(x_0)$ dochodzi do katastrofalnej redukcji cyfr (odejmowanie liczb bliskich sobie).
- Dla $n \geq 53$ ($h \leq 1.11 \times 10^{-16}$): Krok h staje się mniejszy niż epsilon maszynowy dla $x_0 = 1$, co powoduje $\mathrm{fl}(1.0 + h) = 1.0$. W efekcie $f(x_0 + h) = f(x_0)$, licznik staje się 0.0, a całe przybliżenie pochodnej wynosi 0.0, generując maksymalny błąd ≈ 0.1169.

7.4 Wnioski

Zmniejszanie kroku h w metodach różnicowych nie poprawia wyniku w nieskończoność. Istnieje optymalna wartość h (tutaj dla $n = 28$), gdzie błąd obcięcia metody i błąd zaokrąglenia arytmetyki równoważą się. Poniżej tej wartości błędy zaokrąglenia (wynikające z odejmowania bliskich liczb) zaczynają dominować i niszczą dokładność wyniku.