Obliczenia naukowe - sprawozdanie lista nr 1

Piotr Kołodziejczyk

Październik 2019

Spis treści

1 Zadanie 1

1.1 Opis problemu

Zadanie polegało na empirycznym rozpoznaniu arytmetyki komputera, w szczególności na poznaniu wartości liczb:

- Macheps – (epsilon maszynowy) – najmniejsza dodatnia liczba, taka że $fl(1.0 + macheps) > 1.0$,
- Eta – najmniejsza liczba dodatnia,
- Max – liczba maksymalna,

dla każdego z typów Float16, Float32, Float64.

1.2 Rozwiązanie

Liczby wyliczone zostały następująco:

- Epsilon maszynowy – iteracyjne połowienie liczby, dopóki ta dodana do jedności daje w wyniku wartość większą,
- Eta – iteracyjne połowienie liczby, dopóki wynik jest większy od zera,
- Max – podwajanie liczby do momentu gdy dwukrotność owej liczby zwróci nieskończoność, a następnie dodawanie coraz mniejszych liczb (kolejno dzielonych przez 2),

za każdym razem zaczynając od jedności w danym typie, co realizuje skrypt 1.jl.

1.3 Wyniki

Tabele poniżej prezentują uzyskane wyniki.

Typ	wyliczony epsilon	wartość $eps()$	float.h
Float16	0.000977	0.000977	-
Float32	1.1920929e-7	1.1920929e-7	1.192093e-07
Float64	2.220446049250313e-16	2.220446049250313e-16	2.220446e-16

Tabela 1: Porównanie wartości epsilonu maszynowego

Typ	wyliczona eta	$nextfloat(0.0)$
Float16	6.0e-8	6.0e-8
Float32	1.0e-45	1.0e-45
Float64	5.0e-324	5.0e-324

Tabela 2: Porównanie wartości eta

Typ	wyliczony max	floatmax()	float.h
Float16	6.55e4	6.55e4	-
Float32	3.4028235e38	3.4028235e38	3.402823e38
Float64	1.7976931348623157e308	1.7976931348623157e308	1.797693e308

Tabela 3: Porównanie wartości maksymalnych

Wyniki zwracane przez napisane funkcje pokrywają się z faktycznymi wartościami macheps, eta i max. Wartość epsilonu maszynowego stanowi dwukrotność precyzji arytmetyki, wynoszącej $\epsilon = 2^{-t}$ (czyli dla Float32 i Float64 odpowiednio $5.96 \cdot 10^{-8}$ oraz $1.11 \cdot 10^{-16}$). Liczba MIN_{sub} wyraża się wzorem $2^{-t-1}2^{c_{min}}$ ($c_{min} = 2^{d-1} + 2$, d i t to ilość bitów przeznaczona odpowiednio na eksponentę i mantysę), co w wyniku daje wartość eta dla obu arytmetyk. Funkcje floatmin() dla obu z typów zwraca wartość MIN_{nor} równą $2^{c_{min}}$ (odpowiednio $1.18 \cdot 10^{-38}$ i $2.2 \cdot 10^{-308}$).

1.4 Wnioski

Wyliczone wartości pokazują, że arytmetyka IEEE754 nie odzwierciedla liczb rzeczywistych - jesteśmy ograniczeni zarówno co do dokładności liczb zmiennoprzecinkowych, jak i ich zakresu.

2 Zadanie 2

2.1 Opis problemu

Zadanie polegało na sprawdzeniu hipotezy, jakoby epsilon maszynowy dało się wyliczyć z wzoru

$$3(\frac{4}{3} - 1) - 1,$$

co postulował Kahan.

2.2 Rozwiązanie

W celu potwierdzenia lub obalenia poprawności stwierdzenia, dla każdego z typów Float16, Float32, Float64 wyliczam wartość tego wyrażenia.

2.3 Wyniki

Tabela 4 przedstawia wartości wyrażenia w danym typie, które równają się (z dokładnością do wartości bezwzględnej) epsilonowi maszynowemu.

2.4 Wnioski

Kahan słusznie stwierdził, że wyrażenie 3(4/3-1)-1 pozwala poznać wartość epsilonu maszynowego danego typu, jednak z dokładnością do znaku.

Typ	Wynik funkcji	Wartość faktyczna
Float16	-0.000977	0.000977
Float32	1.1920929e-7	1.1920929e-7
Float64	-2.220446049250313e-16	2.220446049250313e-16

Tabela 4: Porównanie wartości wyrażenia $3(4/3-1)-1$ z wartością macheps

3 Zadanie 3

3.1 Opis problemu

Zadanie polegało na eksperymentalnym sprawdzeniu, że w przedziale [1,2] liczby zmiennopozycyjnie w arytmetyce Float64 są rozłożone równomiernie z krokiem $\delta = 2^{-52}$. Ponadto sprawdzić jak wygląda rozmieszczenie w przedziałach $[\frac{1}{2}, 1]$ oraz [2,4].

3.2 Rozwiązanie

Najprostsza, ale jednocześnie najbardziej czasochłonna wersja rozwiązania polegałaby na iteracji po wszystkich liczbach arytmetyki, i sprawdzeniu czy różnica pomiędzy każdymi kolejnymi dwoma wynosi δ. Wymaga to jednak wykonania $2^{52} = 4.5 \cdot 10^{15}$ porównań.

Alternatywne rozwiązanie polega na wykorzystaniu podstawowej wiedzy na temat standardu IEEE754. W przedziale [1,2] pierwszych 12 bitów każdej liczby jest jednakowe (bit znaku i 11 bitów eksponenty). Pozostają 52 bity - widać więc że możliwych kombinacji jest 2^{52}. Podobne sprawdzenie można wykonać dla przedziałów $[\frac{1}{2}, 1]$ oraz [2,4], zwracając uwagę na liczbę bitów potrzebnych do zakodowania części całkowitej liczby.

3.3 Wyniki

Przedział	Odległość
[1,2]	$2.22 \cdot 10^{-16}$
$[\frac{1}{2}, 1]$	$1.11 \cdot 10^{-16}$
[2,4]	$4.44 \cdot 10^{-16}$

Tabela 5: Odległości pomiędzy liczbami w danym przedziale

Różnica dwu kolejnych liczb wyznacza odległość pomiędzy liczbami w danym przedziale, a podstawowa wiedza na temat IEEE754 pozwala być pewnym, że odległość ta jest jednakowa w całym przedziale. Różnica pomiędzy przedziałami wynika z większej liczby bitów potrzebnych na zakodowanie części całkowitej liczby.

3.4 Wnioski

Odległość pomiędzy kolejnymi liczbami w standardzie IEEE754 zależy od przedziału, w jakim się te liczby znajdują i jest przedziałami stała. Przedziały te są zależne od kolejnych potęg dwójki, gdyż one wpływają na liczbę bitów potrzebnych do zakodowania części całkowitej liczby, kosztem bitów części ułamkowej.

4 Zadanie 4

4.1 Opis problemu

Celem zadania było eksperymentalne wyznaczenie liczby x, takiej że $1 < x < 2$ i $fl(x * fl(\frac{1}{x})) \neq 1$ w arytmetyce Float64. Pomimo że w arytmetyce liczb rzeczywistych równość $x * (\frac{1}{x}) = 1$ jest zawsze prawdziwa, w standardzie IEEE754 – nie.

4.2 Rozwiązanie

Rozwiązanie zrealizowane w skrypcie 4.jl polega na iteracji po wszystkich liczbach standardu Float64 począwszy od 1.0, do znalezienia liczby spełniającej nierówność. W celu znalezienia najmniejszej takiej liczby najpierw podwajana jest podstawiana wartość, dopóki wyrażenie nie przestanie zwracać nieskończoności (co dzieje się dla liczb bliskich zeru), aby dalej iterować jak w pierwszej części.

4.3 Wyniki

Znalezione wartości x to:

- w przedziale $(1, 2)$: $x = 1.000000057228997$,
- w przedziale $(0, 1)$: $x = 1.1125369929229734e - 308$, a dopuszczając nieskończoność jako wynik: $x = 5.0 \cdot 10^{-324}$,

4.4 Wnioski

W standardzie IEEE754 istnieją liczby, które nie spełniają rzeczywistych równości, co implikuje potrzebę ostrożności, szczególnie w przypadku sprawdzania warunków równości w programach operujących na liczbach float.

5 Zadanie 5

5.1 Opis problemu

Celem zadania było obliczenie iloczynu skalarnego dwu wektorów na kilka sposobów w różnych precyzjach i zaobserwowanie różnic w wynikach oraz różnicy z wartością dokładną.

5.2 Rozwiązanie

Program 5.jl oblicza na cztery sposoby iloczyn skalarny wektorów:

a) w przód - $\sum_{i=1}^{n} x_i y_i$,

b) w tył - $\sum_{i=n}^{1} x_i y_i$,

c) dodać osobno iloczyny dodatnie od najmniejszego do największego i ujemne od największego do najmniejszego,

d) odwrotnie jak w c).

5.3 Wyniki

Wyniki działania programu przedstawia tabela, podczas gdy faktyczna wartość iloczynu wynosi $-1.00657107000000 \cdot 10^{-11}$.

Sposób	Float32	Float64
a)	-0.4999443	1.0251881e-10
b)	-0.4543457	-1.5643309e-10
c)	-0.5	0.0
d)	-0.5	0.0

Tabela 6: Porównanie wyników iloczynu skalarnego wyliczonego na różne sposoby

5.4 Wnioski

Tylko wyniki w podwójnej precyzji są bliskie wartości rzeczywistej iloczynu skalarnego. Wartości są bliskie zeru, zatem wektory są prawie prostopadłe. Ze względu na znaczne różnice w wynikach w tego typu obliczeniach, powinno używać się liczb zmiennoprzecinkowych podwójnej precyzji (nawet jeśli pozostałe obliczenia wykonywane są w pojedynczej). Co więcej, zadanie pokazuje, że w standardzie IEEE754 zmiana kolejności dodawania może dać różne wyniki.

6 Zadanie 6

6.1 Opis problemu

Zadanie polegało na implementacji dwu funkcji $f(x)$ i $g(x)$, które z matematycznego punktu widzenia są równe, ale w standardzie IEEE754 mogą dawać różne wyniki, oraz ocenie wiarygodności wyników dla kolejnych ujemnych potęg ósemki.

6.2 Rozwiązanie

Skrypt 6.jl implementuje funkcje f oraz g jak napisane.

6.3 Wyniki

Tabela przedstawia wyliczone wartości funkcji f oraz g dla argumentów postaci 8^{-i}

i	$f(8^{-i})$	$f(8^{-i})$
1	0.0077822185373186414	0.0077822185373187065
2	0.00012206286282867573	0.00012206286282875901
3	1.9073468138230965e-6	1.907346813826566e-6
4	2.9802321943606103e-8	2.9802321943606116e-8
5	4.656612873077393e-10	4.6566128719931904e-10
6	7.275957614183426e-12	7.275957614156956e-12
7	1.1368683772161603e-13	1.1368683772160957e-13
8	1.7763568394002505e-15	1.7763568394002489e-15
9	0.0	2.7755575615628914e-17
10	0.0	4.336808689942018e-19
20	0.0	3.76158192263132e-37
30	0.0	3.2626522339992623e-55
40	0.0	2.8298997121333476e-73
50	0.0	2.4545467326488633e-91
60	0.0	2.1289799200040754e-109
70	0.0	1.8465957235571472e-127
80	0.0	1.6016664761464807e-145
90	0.0	1.3892242184281734e-163
100	0.0	1.204959932551442e-181

Tabela 7: Porównanie wyników funkcji f i g

6.4 Wnioski

Funkcja g z ilorazem daje dokładniejsze wyniki, ponieważ skutecznie omija problem napotykany przez funkcję f – odejmowanie bliskich sobie liczb.

7 Zadanie 7

7.1 Opis problemu

Zadanie polegało przeprowadzeniu eksperymentów z obliczaniem wartości pochodnej funkcji w punkcie wzorem:

$$f'(x_0) \approx \widetilde{f}'(x_0) = \frac{f(x_0 + h) - f(x_0)}{h},$$

porównanie z faktyczną wartością pochodnej oraz analizie różnicy wyników w zależności od wartości parametru h, na przykładzie funkcji $f(x) = sinx + cos3x$.

7.2 Rozwiązanie

Dla funkcji f wyliczona została pochodna funkcji w punkcie $x_0 = 1$ dla każdego $h \in \{2^0, 2^{-1}, ..2^{-54}\}$ oraz błąd $|f'(x) - \widetilde{f'}(x)|$, gdzie $f'(x) = cosx - 3sin3x$.

7.3 Wyniki

Tabela poniżej zestawia obliczoną wartość pochodnej funkcji w punkcie ilorazem różnicowym z błędem bezwzględnym wyniku.

h	$\widetilde{f'}(x)$	$\lvert f'(x) - \widetilde{f'}(x)\rvert$
2^{-1}	1.8704413979316472	1.753499116243109
2^{-2}	1.1077870952342974	0.9908448135457593
2^{-3}	0.6232412792975817	0.5062989976090435
2^{-4}	0.3704000662035192	0.253457784514981
2^{-5}	0.24344307439754687	0.1265007927090087
2^{-6}	0.18009756330732785	0.0631552816187897
2^{-7}	0.1484913953710958	0.03154911368255764
2^{-8}	0.1327091142805159	0.015766832591977753
2^{-9}	0.1248236929407085	0.007881411252170345
2^{-10}	0.12088247681106168	0.0039401951225235265
2^{-11}	0.11891225046883847	0.001969968780300313
2^{-12}	0.11792723373901026	0.0009849520504721099
2^{-13}	0.11743474961076572	0.0004924679222275685
2^{-14}	0.11718851362093119	0.0002462319323930373
2^{-15}	0.11706539714577957	0.00012311545724141837
2^{-16}	0.11700383928837255	6.155759983439424e-5
2^{-17}	0.11697306045971345	3.077877117529937e-5
2^{-18}	0.11695767106721178	1.5389378673624776e-5
2^{-19}	0.11694997636368498	7.694675146829866e-6
2^{-20}	0.11694612901192158	3.8473233834324105e-6
2^{-21}	0.1169442052487284	1.9235601902423127e-6
2^{-22}	0.11694324295967817	9.612711400208696e-7
2^{-23}	0.11694276239722967	4.807086915192826e-7
2^{-24}	0.11694252118468285	2.394961446938737e-7
2^{-25}	0.116942398250103	1.1656156484463054e-7
2^{-26}	0.11694233864545822	5.6956920069239914e-8
2^{-27}	0.11694231629371643	3.460517827846843e-8
2^{-28}	0.11694228649139404	4.802855890773117e-9
2^{-29}	0.11694222688674927	5.480178888461751e-8
2^{-30}	0.11694216728210449	1.1440643366000813e-7
2^{-31}	0.11694216728210449	1.1440643366000813e-7
2^{-32}	0.11694192886352539	3.5282501276157063e-7
2^{-33}	0.11694145202636719	8.296621709646956e-7

2^{-34}	0.11694145202636719	8.296621709646956e-7
2^{-35}	0.11693954467773438	2.7370108037771956e-6
2^{-36}	0.116943359375	1.0776864618478044e-6
2^{-37}	0.1169281005859375	1.4181102600652196e-5
2^{-38}	0.116943359375	1.0776864618478044e-6
2^{-39}	0.11688232421875	5.9957469788152196e-5
2^{-40}	0.1168212890625	0.0001209926260381522
2^{-41}	0.116943359375	1.0776864618478044e-6
2^{-42}	0.11669921875	0.0002430629385381522
2^{-43}	0.1162109375	0.0007313441885381522
2^{-44}	0.1171875	0.0002452183114618478
2^{-45}	0.11328125	0.003661031688538152
2^{-46}	0.109375	0.007567281688538152
2^{-47}	0.109375	0.007567281688538152
2^{-48}	0.09375	0.023192281688538152
2^{-49}	0.125	0.008057718311461848
2^{-50}	0.0	0.11694228168853815
2^{-51}	0.0	0.11694228168853815
2^{-52}	-0.5	0.6169422816885382
2^{-53}	0.0	0.11694228168853815
2^{-54}	0.0	0.11694228168853815

Tabela 8: Porównanie wyników funkcji f i g

Możemy zauważyć, że błąd bezwzględny maleje razem z wartością h, aż do momentu gdy $h = 2^{-28}$, odkąd zaczyna rosnąć. Dla bardzo małych h w liczniku odejmujemy bardzo bliskie sobie liczby, co skutkuje niską dokładnością. Wartości 1+h dla ostatnich przypadków wynoszą 1.0, gdyż h^{-i} jest mniejsze od epsilonu maszynowego.

7.4 Wnioski

Pomimo matematycznego modelu, w którym w celu policznia pochodnej funkcji w punkcie ilorazem różnicowym $h \to 0$, w arytmetyce komputera poniżej pewnego poziomu mniejsza wartość h nie zwiększa dokładności obliczeń - przeciwnie.