Sprawozdanie z Listy 1
Obliczenia Naukowe

Tomasz Hałas

25 października 2021

Spis treści

1 Zadanie 1.

1.1 Opis problemu

Zadanie polega na wyliczeniu:

- epsilona maszynowego „*macheps*", czyli najmniejszej liczby spełniającej nierówność $fl(1.0 + macheps) > 1.0$ i $fl(1 + macheps) == 1$.
- liczby maszynowej „*eta*", czyli najmniejszej reprezentowalnej liczby.
- liczby „MAX", czyli największej reprezentowalnej liczby.

dla różnych typów zmiennoprzecinkowych w standardzie IEEE 754.

1.2 Rozwiązanie

Rozwiązania znajdują się w pliku zad1L1.jl. Polegają one na wyliczeniu podanych wyżej liczb iteracyjnie za pomocą funkcji w odpowiednim typie zmiennoprzecinkowym:

1. w funkcji $find\_macheps()$ zaczynając od 1.0, dziele zmienną przez dwa do momentu otrzymania *macheps*, czyli takiej liczby, że $fl(1 + macheps) == 1$ i $fl(1.0 + macheps) > 1.0$.
2. w funkcji $find\_eta()$ zaczynając od 1.0, dziele zmienną $current\_number$ przez dwa do momentu otrzymania *eta*, czyli gdy $current\_number/2 \neq 0$.
3. w funkcji $find\_max()$ mnoże zmienną przez 2 do momenu otrzymania $MAX \neq \infty$.

W przypadku, gdy osiągne szukana liczbę, przerywam pętle i zwracam wynik.

1.3 Wyniki

Artmetyka	**Mój** *macheps*	**eps()**	**float.h**
Float16	0.000977	0.000977	brak
Float32	1.1920929e-7	1.1920929e-7	1.19209290E-07F
Float64	2.220446049250313e-16	2.220446049250313e-16	2.2204460492503131E-016

Tabela 1: Tabela wartości dla epsliona maszynowego.

Widzimy pewien związek pomiędzy wartością epsilona maszynowego, a precyzją. Mianowicie im mniejsza jest wartość epsilona maszynowego, tym większa jest względna precyzja naszych obliczeń.

Artmetyka	**Moja** *eta*	**nextfloat**
Float16	6.0e-8	6.0e-8
Float32	1.0e-45	1.0e-45
Float64	5.0e-324	5.0e-324

Tabela 2: Tabela wartości dla liczby maszynowej.

Zależność pomiędzy liczbą eta i MIN_{sub} jest, taka że liczba eta jest równa liczbie MIN_{sub} i jest najmiejszą zdenormalizowaną liczbą reprezentowalną w danej aremtyce.

Artmetyka	**Mój** MAX	**floatmax()**	**float.h**
Float16	6.55e4	6.55e4	Brak
Float32	3.4028235e38	3.4028235e38	3.40282347E+38F
Float64	1.7976931348623157e308	1.7976931348623157e308	1.7976931348623157E+308

Tabela 3: Tabela wartości dla MAX.

Na podstawie swoich wyników stwierdzam, że moje funkcje zostały poprawnie zaimplementowane, gdyż zwróciły oczekiwane przez nas wartości.

1.4 Wniosek

Artmetyka IEEE 754 posiada skończoną precyzję, jak i zakres liczb, które jest w stanie „reprezentować".
Istnieją też dwa rodzaje bardzo małych liczb. Mowa tu o liczbach znormalizowanych i zdenormalizowanych. Liczba *eta* (uzyskana przeze mnie) lub za pomocą nextfloat() to liczba zdenormalizowana. Natomiast liczby uzyskane przez komende:

Artmetyka	**floatmin()**
Float32	1.1754944e-38
Float64	2.2250738585072014e-308

Tabela 4: Tabela wartości dla floatmin().

to liczby znormalizowane. Są to tak naprawdę MIN_{nor} dla danej artmeytyki zmiennopozycyjnej.

2 Zadanie 2.

2.1 Opis problemu

Kahan stwierdził, że epsilon maszynowy (*macheps*) można otrzymać, obliczając wyrażenie $3 * (4/3 - 1) - 1$ w arytmetyce zmiennopozycyjnej.

2.2 Rozwiązanie

Rozwiązania znajdują się w pliku zad2L1.jl. Polegają one na obliczeniu wyrażenia
$3 * (4/3 - 1) - 1$ w odpowiedniej artmetyce zmiennopozycyjnej.

2.3 Wyniki

Artmetyka	**Moje obliczenia**
Float16	-0.000977
Float32	1.1920929f-7
Float64	-2.220446049250313e-16

Tabela 5: Tabela wartości dla wyrażenia $3 * (4/3 - 1) - 1$.

Porównując wyniki do tabeli z epsilonem maszynowym (*macheps*), widzimy że bezwględna wartość otrzymanych wyników jest identyczna z tymi w tabeli.
Różnica pod względem znaku pojawiła się zapewne przez reprezentację liczby 4/3 w systemie binarnym. W Float32 doszło do zaokrąlgenia z nadmiarem. Natomiast w Float16 i Float64 z niedomiarem, co spowodowało negatywne wyniki.

2.4 Wniosek

Przez ograniczoną dokładność reprezentacji dochodzi bardzo często to błedów w artmetyce zmiennopozycyjnej, wynikających między innymi z używania zaokrągleń, co bezpośrednio może wpływać na nasz wynik końcowy.

3 Zadanie 3.

3.1 Opis problemu

Zadanie polega na sprawdzeniu rozmieszczenia liczb w określonych przedziałach liczbowych w artmetyce Float64.

3.2 Rozwiązanie

Rozwiązanie znajduje się w pliku zad3L1.jl. Za pomoca funkcji $maunal()$ sprawdziłem iteracyjnie, czy liczby są równomiernie rozmieszczone w przedziale [1,2] dla $\delta = 2^{-52}$.

Natomiast zgodnie z standardem IEEE 754 napisałem funkcje $spread\_numbers()$, która zwraca odległości pomiędzy liczbami (ich rozmieszczenie) z określonego przedziału za pomocą wzoru:

$$2.0^{eksponenta-(2^{10}-1)} * 2.0^{-52}.$$

3.3 Wyniki

Przedzial	Moje obliczenia
[0.5-1]	1.1102230246251565e-16
[1-2]	2.220446049250313e-16
[2-4]	4.440892098500626e-16

Tabela 6: Tabela odległości liczb na danym przedziale.

Widzimy, że powiększenie eksponenty o jeden na podstawie wzoru $(2.0^{eksponenta-(2^{10}-1)} * 2.0^{-52})$ powoduje wzrost rozmieszczenia między liczbami dwukrotnie. Wynika to bezpośrednio z reprezentacji danej liczby w IEEE 754.

3.4 Wniosek

W standardzie IEEE 754 liczby są równomiernie rozmieszczone na określonym przedziale.

4 Zadanie 4.

4.1 Opis problemu

Musimy znaleść:

- liczbę zmiennopozycyjną x w przedziale $1 < x < 2$, taką że $x * 1/x \neq= 1$
- najmniejszą liczbę, taką że $x * 1/x \neq= 1$ i $0 < x$

4.2 Rozwiązanie

Rozwiązania znajdują się w pliku zad4L1.jl. Rozwiązanie podpunktu 1) polega na wzięciu dolnego ograniczenia przedziału i powiększeniu go o kolejne liczby maszynowe. Następnie w pętli sprawdzam, dla której liczby równanie nie będzie spełnione.

Natomiast w przypadku podpunktu 2) biorę dolne ograniczenie przedziału (w tym przypadku 0) i powiększam o kolejne liczby maszynowe. Wykonuje operacje jak w 1) do momentu, gdy równanie nie będzie spełnione.

4.3 Wyniki

Podpunkt	**Wynik**
1	1.0000000572289973
2	$1.0e-323$

Tabela 7: Tabela wyników

Dla tych liczb nie zachodzą już odpowiednio równania z podpunktu 1) i 2).

4.4 Wniosek

Artmetyka zmiennopozycyjna z powodu ograniczonej dokładność i precyzji nie zawsze zwraca poprawny wynik.

5 Zadanie 5.

5.1 Opis problemu

Problem opiera się na policzeniu iloczynu skalarnego dwóch znanych wektorów na kilka sposobów:

- w przód.
- w tył.
- od największego do najmniejszego.
- od najmniejszego do największego.

5.2 Rozwiązanie

Rozwiązania znajdują się w pliku zad5L1.jl. Zaimplementowałem wyżej wymienione sposoby liczenia iloczynu skalarnego za pomocą czterech funkcji odpowiadającym podpunktom w zadaniu.

5.3 Wyniki

Artmetyka	**Wynik a)**	**Wynik b)**	**Wynik c)**	**Wynik d)**
Float32	-0.3472038161853561	-0.3472038162872195	-0.5	-0.5
Float64	1.0251881368296672e-10	-1.5643308870494366e-10	0.0	0.0

Tabela 8: Wyniki iloczynu skalaranego.

Otrzymane wyniki różnią się od wyniku poprawnego $-1.00657107000000 * 10^{-11}$.

5.4 Wniosek

Kolejność w jakiej wykonywane są obliczenia, może mieć diametralny wpływ na końcowy wynik.

6 Zadanie 6.

6.1 Opis problemu

Musimy policzyć wartości dla 2 równoważnych matematycznie funkcji:

- $f(x) = \sqrt{x^2 + 1} - 1$.
- $g(x) = x^2/(\sqrt{x^2 + 1} + 1)$.

dla $x = 8^{-1}, 8^{-2}...$

6.2 Rozwiązanie

Rozwiązanie znajduje się w pliku zad6L1.jl. Polega ono na wywołaniu iteracyjnie funkcji z określonym x.

6.3 Wyniki

wartosc x	$f(x)$	$g(x)$
1	0.0077822185373186414	0.0077822185373187065
2	0.00012206286282867573 5	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
100	0.0	1.204959932551442e-181
179	0.0	0.0

Tabela 9: Tabela dla $f(x)$ i $g(x)$.

Widzimy, że funckja $f(x)$ bardzo szybko dąży do 0. Natomiast $g(x)$ pomimo, że matematycznie oznacza to samo, to numerycznie wykonuje dokładne obliczenia na ile Float64 pozwala. Powodem dla którego funckja $f(x)$ nie poradziła sobie, jest fakt, że działa ona na wartościach bliskich zeru. Oblicza ona róznice pierwiastka, którego wartość jest bliska 1, z jedynką.

6.4 Wniosek

Wykonując obliczenia musimy się starać, aby liczba cyfr znaczących podczas następnych działań nie różniła się diametralnie. Poprawi to precyzję naszych obliczeń.

7 Zadanie 7.

7.1 Opis problemu

Problem polega na wyliczeniu przybliżonej wartości pochodnej w punkcie x za pomocą:

$$f'(x) \approx \widetilde{f}'(x) = \frac{f(x+h_0)+x}{h}$$

dla $f(x) = sin(x) + cos(3x)$ w punkcie $x_0 = 1$ oraz błędów $f'(x) - \widetilde{f}'(x)$ dla h=2^{-n} dla $(n = 0, 1, 2, 3..., 54)$.

7.2 Rozwiązanie

Rozwiązana znajdują się w pliku zad7L1.jl. Funkcja $f(x)$ liczy pochodną według wzoru, natomiast funkcja $def\_f(x)$ liczy „gotowa pochodną” funkcji $f(x)$.

7.3 Wyniki

Pochodną obliczam według wzoru $f'(x) \approx \widetilde{f}'(x) = \frac{f(x+h_0)+x}{h}$, natomiast różnice uzyskuje iteracyjnie w pętli, gdzie niezmienkiem jest „i”, poprzez: $\frac{f(1.0+2.0^{-i})-f(1.0)}{2.0^{-i}} - der\_f(1.0)$

wartosc h	**Pochodna**	**Róznica**
2^0	2.0179892252685967	1.9010469435800585
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

Dla malejącego h widzimy poprawe w przybliżaniu wartość naszych obliczeń do momentu gdy h=28. Błąd wtedy jest najmnieszy i spada do wielkości rzędu 10^{-9}. Jednak dla h $>$ 28 błąd rośnie, aż do momentu gdy osiagnie wartość 100%. Z coraz to mniejszymi liczbami tracimy liczby znaczące, co w konsekwencji przyczynia się do wzrostu błędu i pogorszenia dokładności.

7.4 Wniosek

Dokładność artmeyki IEEE 754 jest ograniczona, dlatego trzeba unikać liczb bardzo bliskich 0.