Obliczenia naukowe
Lista 1

Joanna Szołomicka

2019-10-20

1 Zadanie 1

1.1 Epsilon maszynowy

1.1.1 Opis problemu

Zadanie polega na iteracyjnym wyznaczeniu epsilonu maszynowego dla wszystkich typów zmiennopozycyjnych w standardzie IEEE 754. Wyniki należy porównać z wartościami funkcji eps w języku Julia oraz danymi zawartymi w pliku nagłówkowym float.h języka C.

1.1.2 Rozwiązanie

Epsilon maszynowy określa precyzję obliczeń numerycznych wykonywanych na liczbach zmiennopozycyjnych. Macheps jest to najmniejsza liczba $macheps > 0$ taka, że $fl(1.0 + macheps) > 1.0$.

Algorytm zaczynam od przypisania $macheps = 1$, a następnie dzielenia go przez 2 do momentu aż powyższa nierówność przestanie być prawdziwa. Otrzymana liczba to epsilon maszynowy.

1.1.3 Wyniki

typ	**macheps**	**eps(type)**	**float.h**
Float16	0.000977	0.000977	-
Float32	$1.1920929 * 10^{-7}$	$1.1920929 * 10^{-7}$	$1.1920929 * 10^{7}$
Float64	$2.220446049250313 * 10^{-16}$	$2.220446049250313 * 10^{-16}$	$2.220446049250313 * 10^{-16}$

Tabela 1: Wartości macheps.

1.1.4 Wnioski

Algorytm wyznaczania epsilonu maszynowego jest poprawny, gdyż wyniki pokrywają się z prawidłowymi dla wszystkich typów. Można zauważyć, że im większa precyzja obliczeń tym mniejsza jest wartość macheps. Precyzja obliczeń zadana wzorem $\varepsilon = 2^{-t-1}$, gdzie t jest równe liczbie bitów do zapisania mantysy, jest dwa razy mniejsza niż epsilon maszynowy 2^{-t}. Im większa precyzja arytmetyki tym liczby są dokładniej reprezentowane.

1.2 Liczba eta

1.2.1 Opis problemu

Zadanie polega na iteracyjnym wyznaczeniu liczby $eta > 0.0$ dla wszystkich typów zmiennopozycyjnych w standardzie IEEE 754. Wyniki należy porównać z wartościami funkcji nextfloat w języku Julia.

1.2.2 Rozwiązanie

Algorytm zaczynam od przypisania $eta = 1$, a następnie dzielenia jej przez 2 do momentu aż powyższa nierówność przestanie być prawdziwa. Otrzymana liczba to eta.

1.2.3 Wyniki

typ	**eta**	**nextfloat(type)**
Float16	$6.0 * 10^{-8}$	$6.0 * 10^{-8}$
Float32	$1.0 * 10^{-45}$	$1.0 * 10^{-45}$
Float64	$5.0 * 10^{-324}$	$5.0 * 10^{-324}$

Tabela 2: Wartości eta.

typ	**floatmin(type)**
Float32	1.175494^{-38}
Float64	$2.2250738585072014^{-308}$

Tabela 3: Wartości floatmin.

1.2.4 Wnioski

Algorytm wyznaczania eta jest poprawny dla wszystkich typów zmiennopozycyjnych. Liczba eta jest najmniejszą dodatnią liczbą w IEEE 754. Jest nieznormalizowana, równa MIN_{sub}, co można zauważyć po zerowych bitach jej cechy: 0 00000000 00000000000000000000001

Funkcja floatmin zwraca MIN_{nor}, czyli najmniejszą znormalizowaną liczbę w danej arytmetyce.

1.3 Liczba MAX

1.3.1 Opis problemu

Zadanie polega na iteracyjnym wyznaczeniu liczby MAX dla wszystkich typów zmiennopozycyjnych w standardzie IEEE 754. Wyniki należy porównać z wartościami funkcji floatmax w języku Julia.

1.3.2 Rozwiązanie

Liczba MAX jest największą możliwą do zapisania liczbą. Algorytm zaczynam od przypisania $max = 1$, a następnie mnożenia max razy 2 do momentu aż nierówność $MAX * 2 < inf$ nie przestanie być prawdziwa. Otrzymana liczba ma 0 w mantysie, a więc nie jest największa. Następnie dodaje do max: $max/2^1$, $max/2^2$, $max/2^3$ dopóki nie otrzymam liczby MAX.

1.3.3 Wyniki

typ	**MAX**	**floatmax(type)**
Float16	$6.55 * 10^4$	$6.55 * 10^4$
Float32	$3.4028235 * 10^{38}$	$3.4028235 * 10^{38}$
Float64	$1.7976931348623157 * 10^{308}$	$1.7976931348623157 * 10^{308}$

Tabela 4: Wartości MAX.

1.3.4 Wnioski

Wyniki iteracyjnego algorytmu pokrywają się z wynikami funkcji floatmax wyznaczającej największą liczbę w danej arytmetyce.

2 Zadanie 2

2.1 Epsilon maszynowy Kahana

2.1.1 Opis problemu

Zadanie polega na eksperymentalnym sprawdzeniu słuszności twierdzenia Kahana dla wszystkich typów zmiennopozycyjnych.

2.1.2 Rozwiązanie

Program sprawdza, czy obliczając wyrażenie $(4/3 - 1) - 1$ można otrzymać epsilon maszynowy.

2.1.3 Wyniki

typ	kahan	eps(type)
Float16	-0.000977	0.000977
Float32	$1.1920929 * 10^-7$	$1.1920929 * 10^-7$
Float64	$-2.220446049250313 * 10^{-16}$	$2.220446049250313 * 10^{-16}$

Tabela 5: Wartości epsilonu maszynowego Kahana.

2.1.4 Wnioski

Wyniki uzyskane z twierdzenia Kahana są równe co do wartości z epsilonem maszynowym. Dla Float16 i Float64 różnią się jedynie znakiem. Metoda Kahana jest poprawna, gdyż w systemie dwójkowym nie da się przedstawić dokładnie liczby 4/3. Niedokładność zaokręglenia będzie więc równa liczbie macheps. Błąd znaku jest spowodowany zaokrąglaniem liczby z niedomiarem.

3 Zadanie 3

3.1 Rozmieszczenie liczb maszynowych

3.1.1 Opis problemu

Zadanie polega na eksperymentalnym sprawdzeniu, jak są rozmieszczone liczby w arytmetyce zmiennopozycyjnej.

3.1.2 Rozwiązanie

Program sprawdza, czy liczby z przedziału $[1, 2]$ są równomiernie rozmieszczone tzn, czy można je przedstawić w postaci $x = 1 + k\delta$, gdzie $k = 1, 2, ..., 2^{52} - 1$, $\delta = 2^{-52}$. Sprawdza także, jakie jest rozmieszczenie liczb w przedziale $[1/2, 1]$ i $[2, 4]$.

3.1.3 Wyniki

$[1, 2]$ $\delta = 2^{-52}$
0011111111110000000000000000000000000000000000000000000000000000
0011111111110000000000000000000000000000000000000000000000000001
0011111111110000000000000000000000000000000000000000000000000010
⋮
0011111111111111111111111111111111111111111111111111111111111101
0011111111111111111111111111111111111111111111111111111111111110
0011111111111111111111111111111111111111111111111111111111111111
$[1/2, 1]$ $\delta = 2^{-53}$
0011111111110000000000000000000000000000000000000000000000000000
0011111111110000000000000000000000000000000000000000000000000001
0011111111110000000000000000000000000000000000000000000000000010
⋮
0011111111111111111111111111111111111111111111111111111111111101
0011111111111111111111111111111111111111111111111111111111111110
0011111111111111111111111111111111111111111111111111111111111111
$[2, 4]$ $\delta = 2^{-51}$
0011111111110000000000000000000000000000000000000000000000000000
0011111111110000000000000000000000000000000000000000000000000001
0011111111110000000000000000000000000000000000000000000000000010
⋮
0011111111111111111111111111111111111111111111111111111111111101
0011111111111111111111111111111111111111111111111111111111111110
0011111111111111111111111111111111111111111111111111111111111111

3.1.4 Wnioski

Można zauważyć, że liczby zmiennopozycyjne są równomiernie rozmieszczone. Liczby pomiędzy kolejnymi potęgami dwójki posiadają tą samą cechę, zmienia się jedynie mantysa. Z tego wyniki, że liczb pomiędzy kolejnymi potęgami dwójki jest 2^t. Wraz ze zwiększaniem się przedziału odległość między nimi rośnie, a gęstość maleje.

4 Zadanie 4

4.1 Działania arytmetyczne

4.1.1 Opis problemu

Zadanie polega na eksperymentalnym znalezieniu liczby zmiennopozycyjnej x z przedziału $(1, 2)$ oraz najmniejszej takie liczby, że $x * (1/x) \neq 1$.

4.1.2 Rozwiązanie

Żeby znaleźć liczbę z przedziału $(1, 2)$ program zaczyna od następnej liczby większej od 1 i wypisuje pierwszą liczbę mniejszą od 2, która spełnia warunek $x * (1/x) \neq 1$. Najmniejsza liczba jest znajdowana w analogiczny sposób, ale program rozpoczyna szukanie od $-floatmax(type)$ i szuka do $floatmax(type)$.

4.1.3 Wyniki

Najmniejsza liczba z przedziału $(1, 2)$ to: 1.000000057228997.
Najmniejsza liczba dla Float64 to: $-1.7976931348623157 * 10^{308}$

4.1.4 Wnioski

Znalezienie takich liczb świadczy o tym, że działania arytmetyczne na liczbach zmiennopozycyjnych mogą generować błędy. Należy pamiętać, że przy takich obliczeniach mogą wystąpić błędy spowodowane zaokrągleniami.

5 Zadanie 5

5.1 Iloczyn skalarny wektorów

5.1.1 Opis problemu

Zadanie polega na obliczeniu iloczynu skalarnego dwóch wektorów
$x = [2.718281828, -3.141592654, 1.414213562, 0.5772156649, 0.3010299957]$
$y = [1486.2497, 878366.9879, -22.37492, 4773714.647, 0.000185049]$
na cztery sposoby dla typów Float32 i Float64.

5.1.2 Rozwiązanie

W programie użyto czterech algorytmów:

1. "w przód"
   $\sum_{i=1}^{n} x_i y_i$
2. "w tył"
   $\sum_{i=n}^{1} x_i y_i$
3. Dodanie liczb dodatnich od największej do najmniejszej, a liczb ujemnych od najmniejszej do największej.
4. Przeciwnie do agorytmu trzeciego.

5.1.3 Wyniki

typ	1	2	3	4
Float64	$1.0251881368296672 * 10^{-10}$	$-1.5643308870494366 * 10^{-10}$	-0.5	-0.5
Float32	-0.4999443	-0.4543457	0.0	0.0

Tabela 6: Wartości iloczynów skalarnych.

Prawidłowy wynik to: $-1.00657107000000 * 10^{-11}$

5.1.4 Wnioski

Kolejność wykonywania działań arytmetycznych w arytmetyce zmiennopozycyjnej jest ważna. Dodawanie do siebie liczb, z których jedna jest dużo większa od drugiej generuje błędy, gdyż większa z liczb pochłania mniejszą. Ponadto im więcej działań jest wykonanych tym błąd jest większy. Można także zauważyć, że użycie arytmetyki o większej precyzji pozwala na uzyskanie mniejszego błędu.

6 Zadanie 6

6.1 Wartości funkcji

6.1.1 Opis problemu

Zadanie polega na obliczeniu wartości funkcji
$f(x) = \sqrt{x^2 + 1} - 1$
$g(x) = \frac{x^2}{\sqrt{x^2+1}+1}$
dla kolejnych wartości $x = 8^{-1}, 8^{-2}, 8^{-3} \ldots$

6.1.2 Rozwiązanie

W programie zostały zaimplementowane funkcje f oraz g. Natępnie należało porównać ich wyniki dla kolejnych iteracji.

6.1.3 Wyniki

x	f	g
8^{-1}	0.0077822185373186414	0.0077822185373187065
8^{-2}	0.00012206286282867573	0.00012206286282875901
8^{-3}	$1.9073468138230965 * 10^{-6}$	$1.907346813826566 * 10^{-6}$
8^{-4}	$2.9802321943606103 * 10^{-8}$	$2.9802321943606116 * 10^{-8}$
8^{-5}	$4.656612873077393 * 10^{-10}$	$4.6566128719931904 * 10^{-10}$
8^{-6}	$7.275957614183426 * 10^{-12}$	$7.275957614156956 * 10^{-12}$
8^{-7}	$1.1368683772161603 * 10^{-13}$	$1.1368683772160957 * 10^{-13}$
8^{-8}	$1.7763568394002505 * 10^{-15}$	$1.7763568394002489 * 10^{-15}$
8^{-9}	0.0	$2.775557561562891 * 10^{-17}$
8^{-10}	0.0	$4.336808689942018 * 10^{-19}$
8^{-11}	0.0	$6.776263578034403 * 10^{-21}$
$\vdots$	$\vdots$	$\vdots$
8^{-100}	0.0	$1.204959932551442 * 10^{-181}$
8^{-101}	0.0	$1.8827498946116282 * 10^{-183}$
8^{-102}	0.0	$2.941796710330669 * 10^{-185}$

Tabela 7: Wartości $f(x)$ i $g(x)$.

6.1.4 Wnioski

Mimo, że funkcje g i f są sobie równe otrzymaliśmy różne wyniki. Funkcja f osiąga wartość 0 już dla $x = 8^{-9}$, co jest spowodowane odejmowaniem od siebie liczb, które mają zbliżone wartości. Wartość x zbliża się do 1. Funkcja g daje znacznie bardziej rzeczywiste wyniki. Odejmowanie prawie równych sobie liczb jest związane z utratą cyfr znaczących i przez to daje błędne wyniki.

7 Zadanie 7

7.1 Pochodne

7.1.1 Opis problemu

Zadanie polega na obliczeniu przybliżonej wartości pochodnej funkcji $f(x) = \sin x + \cos 3x$ w punkcie $x_0 = 1$ oraz błędów $|f'(x_0) - \tilde{f}'(x_0)|$ dla $h = 2^{-n}$ $(n = 0, 1, 2, \ldots, 54)$.

7.1.2 Rozwiązanie

Przybliżona wartość pochodnej została obliczona wzorem $\tilde{f}'(x_0) = \frac{f(x_0+h)-f(x_0)}{h}$. W celu określenia błędu została obliczona rzeczywista pochodna ze wzoru $f'(x) =$

$\cos(x) - 3 * \sin(3x)$. Dla kolejnych iteracji h program liczy wartość przybliżonej pochodnej, błąd oraz wartość $h + 1$.

7.1.3 Wyniki

h	$\tilde{f'}(1)$	$\lvert f'(1) - \tilde{f}'(1)\rvert$	$1 + h$
2^{0}	2.0179892252685967	1.90104694358005855	2.0
2^{-1}	1.8704413979316472	1.753499116243109	1.5
2^{-2}	1.1077870952342974	0.9908448135457593	1.25
$\vdots$	$\vdots$	$\vdots$	$\vdots$
2^{-20}	0.11694612901192158	$3.8473233834324105e-6$	1.0000009536743164
2^{-21}	0.1169442052487284	$1.9235601902423127e-6$	1.0000004768371582
2^{-22}	0.11694324295967817	$9.612711400208696e-7$	1.000000238418579
$\vdots$	$\vdots$	$\vdots$	$\vdots$
2^{-40}	0.1168212890625	0.0001209926260381522	1.0000000000009095
2^{-41}	0.116943359375	$1.0776864618478044e-65$	1.0000000000004547
2^{-42}	0.11669921875	0.0002430629385381522	1.0000000000002274
$\vdots$	$\vdots$	$\vdots$	$\vdots$
2^{-51}	0.0	0.116942281688538155	1.0000000000000004
2^{-52}	-0.5	0.6169422816885382	1.0000000000000002
2^{-53}	0.0	0.11694228168853815	1.0
2^{-54}	0.0	0.11694228168853815	1.0

Tabela 8: Wartości przybliżonej pochodnej, błędu oraz $h + 1$.

7.1.4 Wnioski

Wyniki iteracji $1 + h$ pokazują, że jeżeli dodajemy dwie liczby, z których jedna jest znacznie większa od drugiej to większa z liczb pochłania mniejszą. $1 + h$ od pewnego momentu jest równe 1.0. To zjawisko generuje błąd przy obliczaniu przybliżenia pochodnej. Dodatkowo błąd przy obliczaniu funkcji przybliżenia pochodnej może być spowodowany odejmowaniem dwóch zbliżonych do siebie wartości, co jest spowodowane utratą cyfr znaczących.