Obliczenia naukowe - lista 1

Sandra Szwed - 261719

23 października 2022

Zadanie 1 (Rozpoznanie arytmetyki)

Cel

Należało napisać program w języku Julia wyznaczający iteracyjnie epsilony maszynowe, liczbę maszynową eta i liczbę (MAX) dla wszystkich dostępnych typów zmiennopozycyjnych Float16, Float32, Float64, a następnie porównać z wartościami zwracanymi przez funkcje eps(), nextfloat() oraz floatmax().

1. Epsilon maszynowy (macheps)

$$fl(1.0 + macheps) > 1.0 \text{ oraz } fl(1.0 + macheps) = 1 + macheps$$

Macheps jest odległością od liczby 1.0 do kolejnej liczby x, $x > 1$, która jest reprezentowana w arytmetyce zmiennopozycyjnej.

Typy zmiennopozycyjne	Moje wyniki	Wyniki funkcji eps()	Wyniki z pliku float.h
Float16	0.000977	0.000977	nie istnieje
Float32	1.1920929e-7	1.1920929e-7	1.192093e-07
Float64	2.220446049250313e-16	2.220446049250313e-16	2.220446e-16

2. Liczba eta

Liczba maszynowa eta to najmniejsza liczba > 0.0.

Typy zmiennopozycyjne	Moje wyniki	Wyniki funkcji nextfloat()
Float16	6.0e-8	6.0e-8
Float32	1.0e-45	1.0e-45
Float64	5.0e-324	5.0e-324

Jaki związek ma liczba macheps z precyzją arytmetyki (oznaczaną na wykładzie przez ϵ)?

Precyzja arytmetyki to połowa machepsa: macheps $= \beta^{1-t}$, $\epsilon = \frac{1}{2} * \beta^{1-t}$. Z czego wynika, że $\epsilon = \frac{macheps}{2}$, gdzie $\beta = 2$ oraz t jest z przedziału $[\frac{1}{\beta}, 1)$.

Jaki związek ma liczba eta z liczbą MIN_{sub}?

$MIN_{sub} = 2^{1-t} * 2^{c_{min}}$ oraz $c_{min} = -2^{d-1} + 2$
Wyliczając ze wzoru otrzymujemy:
Dla Float16: $float16(MIN_{sub} = 2^{-24}) =$ 6.0e-8
Dla Float32: $float32(MIN_{sub} = 2^{-149}) =$ 1.0e-45
Dla Float64: $float64(MIN_{sub} = 2^{-1074}) =$ 5.0e-324

Zatem eta $= MIN_{sub}$

Co zwracają funkcje floatmin(Float32) i floatmin(Float64) i jaki jest związek zwracanych wartości z liczbą MIN_{nor}?

$MIN_{nor} = 2^{c_{min}}$
Wyliczając ze wzoru otrzymujemy:
Dla Float16: $float16(MIN_{nor} = 2^{-14}) = 6.0\text{e-}8$
Dla Float32: $float32(MIN_{nor} = 2^{-126}) = 1.0\text{e-}45$
Dla Float64: $float64(MIN_{nor} = 2^{-1022}) = 5.0\text{e-}324$

Zestawiając z wartościami zwracanymi przez floatmin():

Typy zmiennopozycyjne	Wyniki funkcji floatmin()	MIN_{nor}
Float32	1.1754944e-38	1.1754944e-38
Float64	2.2250738585072014e-308	2.2250738585072014e-308

Zatem wartości zwracane przez funkcje floatmin() są równe MIN_{nor}.

3. Liczba MAX

Cel

Należało znaleźć liczbę MAX dla wszystkich typów zmiennopozycyjnych Float16, Float32, Float64 i porównać z wynikami zwracanymi przez funkcję floatmax().

Typy zmiennopozycyjne	Moje wyniki	Wyniki funkcji floatmax()	Wyniki z pliku float.h
Float16	6.55e4	6.55e4	-
Float32	3.4028235e38	3.4028235e38	3.402823e+38
Float64	1.7976931348623157e308	1.7976931348623157e308	1.797693e+308

Wnioski

Znaleziona liczba MAX jest równa z wynikami zwracanymi przez funkcję floatmax() oraz z wynikami z pliku float.h.

Zadanie 2

Cel

Należało sprawdzić czy jest możliwe otrzymanie epsilona maszynowego obliczając wyrażenie $3(\frac{4}{3}-1)-1$ w arytmetyce zmiennopozycyjnej.

Wyniki

Typy zmiennopozycyjne	Wyniki wyrażenia Kahana	Wyniki funkcji eps()
Float16	-0.000977	0.000977
Float32	1.1920929e-7	1.1920929e-7
Float64	-2.220446049250313e-16	2.220446049250313e-16

Wnioski

Wyrażenie Kahana daje te same wyniki co funkcja eps() (nie patrząc na znak), zatem można otrzymać w ten sposób epsilon maszynowy..

Zadanie 3

Cel

Należało sprawdzić eksperymentalnie, że w arytmetyce Float64 liczby zmiennopozycyjne są równomiernie rozmieszczone w [1,2] z krokiem $\delta = 2^{-52}$.

Wyniki

Dla [1,2]:

0011111111110000000000000000000000000000000000000000000000000000**1**
0011111111110000000000000000000000000000000000000000000000000000**10**
0011111111110000000000000000000000000000000000000000000000000000**11**
0011111111110000000000000000000000000000000000000000000000000000**100**
0011111111110000000000000000000000000000000000000000000000000000**101**
0011111111110000000000000000000000000000000000000000000000000000**110**
0011111111110000000000000000000000000000000000000000000000000000**111**
0011111111110000000000000000000000000000000000000000000000000000**1000**

W [1,2] liczby są równomiernie rozmieszczone z krokiem $\delta = 2^{-52}$ ponieważ zauważamy, że bity kolejnych liczb zmieniają sie o 1.

Dla $[\frac{1}{2}, 1]$:

0011111111100000000000000000000000000000000000000000000000000000**10**
0011111111100000000000000000000000000000000000000000000000000000**100**
0011111111100000000000000000000000000000000000000000000000000000**110**
0011111111100000000000000000000000000000000000000000000000000000**1000**
0011111111100000000000000000000000000000000000000000000000000000**1010**
0011111111100000000000000000000000000000000000000000000000000000**1100**
0011111111100000000000000000000000000000000000000000000000000000**1110**
0011111111100000000000000000000000000000000000000000000000000000**10000**

Dla $[\frac{1}{2}, 1]$ i kroku $\delta = 2^{-52}$ końcowe bity zmieniają się o 2, zatem spróbujmy dla $\delta = 2^{-53}$:

0011111111100000000000000000000000000000000000000000000000000000**1**
0011111111100000000000000000000000000000000000000000000000000000**10**
0011111111100000000000000000000000000000000000000000000000000000**11**
0011111111100000000000000000000000000000000000000000000000000000**100**
0011111111100000000000000000000000000000000000000000000000000000**101**
0011111111100000000000000000000000000000000000000000000000000000**110**
0011111111100000000000000000000000000000000000000000000000000000**111**
0011111111100000000000000000000000000000000000000000000000000000**1000**

Zmieniają się o 1, zatem dla $[\frac{1}{2}, 1]$ krok wynosi $\delta = 2^{-53}$.

Dla [2,4]:

0100000000000000000000000000000000000000000000000000000000000000
0100000000000000000000000000000000000000000000000000000000000000
0100000000000000000000000000000000000000000000000000000000000000
0100000000000000000000000000000000000000000000000000000000000000
0100000000000000000000000000000000000000000000000000000000000000
0100000000000000000000000000000000000000000000000000000000000000
0100000000000000000000000000000000000000000000000000000000000000
0100000000000000000000000000000000000000000000000000000000000000

Dla przedziału [2,4] dla delty 2^{-51}:

0100000000000000000000000000000000000000000000000000000000000001
0100000000000000000000000000000000000000000000000000000000000010
0100000000000000000000000000000000000000000000000000000000000011
0100000000000000000000000000000000000000000000000000000000000100
0100000000000000000000000000000000000000000000000000000000000101
0100000000000000000000000000000000000000000000000000000000000110
0100000000000000000000000000000000000000000000000000000000000111
0100000000000000000000000000000000000000000000000000000000001000

Dla przedziału [2,4] dla delty 2^{-53}:

0100000000000000000000000000000000000000000000000000000000000000
0100000000000000000000000000000000000000000000000000000000000000
0100000000000000000000000000000000000000000000000000000000000000
0100000000000000000000000000000000000000000000000000000000000000
0100000000000000000000000000000000000000000000000000000000000000
0100000000000000000000000000000000000000000000000000000000000000
0100000000000000000000000000000000000000000000000000000000000000
0100000000000000000000000000000000000000000000000000000000000000

Zatem dla przedziału [2,4] liczby są rozmieszczone z krokiem 2^{-51}.

Dla rozpatrywanego przedziału liczby mogą być przedstawione jako x + $k * \delta$, gdzie x to początek przedziału, k to liczba kroków, a δ to krok.

Zadanie 4

Cel

Należało znaleźć eksperymentalnie w arytmetyce Float64 zgodnej ze standardem IEEE 754 liczbę zmiennopozycyjną x w przedziale $1 < x < 2$, taką, że $x * \frac{1}{x} \neq 1$; tj. $fl(xfl(\frac{1}{x})) \neq 1$ oraz należało znaleźć też najmniejszą taką liczbę.

Wyniki

a) Liczba z podanego przedziału, która spełnia warunek $x * \frac{1}{x} \neq 1$ została znaleziona poprzez losowanie. Taka liczba to przykładowo x = 1.7213617909187509. Podstawiając znalezioną liczbę do warunku zauważamy, że spełnia ona go ponieważ dostajemy wynik $0.9999999999999999 \neq 1$.

b) Najmniejsza liczba: 1.000000057228997

Zadanie 5

Cel

Należało przeprowadzić eksperyment obliczania iloczynu skalarnego dwóch iloczynów na cztery sposoby:

a) "w przód": $\sum_{i=1}^{n} x_i y_i$
b) "w tył": $\sum_{i=n}^{1} x_i y_i$
c) od największego do najmniejszego
d) od najmniejszego do największego

Wyniki

Prawidłowy wynik to $1.00657107000000 \cdot 10^{11}$

Typy zmiennopozycyjne	podpunkt a	podpunkt b	podpunkt c	podpunkt d
Float32	-0.4999443	-0.4543457	-0.5	0.0
Float64	1.0251881368296672e-10	-1.5643308870494366e-10	-0.5	0.0

Wnioski

Każdy sposób obliczania iloczynu skalarnego daje złe wyniki dla liczb zmiennoprzecinkowych.

Zadanie 6

Cel

Należało obliczyć w arytmetyce Float64 wartości funkcji:

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

dla kolejnych wartości elementu x = $8^{-1}, 8^{-2}, 8^{-3}$...

Wyniki

x	f(x)	g(x)
8^{-1}	0.0077822185373186414	0.0077822185373187065
8^{-2}	0.00012206286282867573	0.00012206286282875901
8^{-3}	1.9073468138230965e-6	1.907346813826566e-6
8^{-4}	2.9802321943606103e-8	2.9802321943606116e-8
8^{-5}	4.656612873077393e-10	4.6566128719931904e-10
8^{-6}	7.275957614183426e-12	7.275957614156956e-12
8^{-7}	1.1368683772161603e-13	1.1368683772160957e-13
8^{-8}	1.7763568394002505e-15	1.7763568394002489e-15
8^{-9}	0.0	2.7755575615628914e-17
8^{-10}	0.0	4.336808689942018e-19
8^{-11}	0.0	6.776263578034403e-21

Wnioski

Funkcje f i g są sobie równe, jednak to funkcja g daje dokładniejsze wyniki ponieważ w funckji f odejmujemy od pewnego momentu liczbę 1 od bardzo małej liczby.

Zadanie 7

Cel

Należało wyznaczyć przybliżoną wartość pochodnej funkcji f(x) = sin(x) + cos(3x) w punkcie $x_0 = 1$ korzystając ze wzoru $f'(x_0) \approx \widetilde{f}'(x_0) = \frac{f(x_0+h)-f(x_0)}{h}$ oraz błędów $|f'(x_0) - \widetilde{f}(x_0)|$ dla h = 2^{-n} dla n = 0,1,2,...,54.

Wyniki

h	przybliżona wartość pochodnej f(x)	błąd	1 + h
2^0	2.0179892252685967	1.9010469435800585	2.0
2^{-1}	1.8704413979316472	1.753499116243109	1.5
2^{-2}	1.1077870952342974	0.9908448135457593	1.25
2^{-3}	0.6232412792975817	0.5062989976090435	1.125
2^{-4}	0.3704000662035192	0.253457784514981	1.0625
2^{-5}	0.24344307439754687	0.1265007927090087	1.03125
2^{-6}	0.18009756330732785	0.0631552816187897	1.015625
2^{-7}	0.1484913953710958	0.03154911368255764	1.0078125
2^{-8}	0.1327091142805159	0.015766832591977753	1.00390625
2^{-9}	0.1248236929407085	0.007881411252170345	1.001953125
2^{-10}	0.12088247681106168	0.0039401951225235265	1.0009765625
⋮	⋮	⋮	
2^{-25}	0.116942398250103	1.1656156484463054e-7	1.0000000298023224
2^{-26}	0.11694233864545822	5.6956920069239914e-8	1.0000000149011612
2^{-27}	0.11694231629371643	3.460517827846843e-8	1.0000000074505806
2^{-28}	0.11694228649139404	4.802855890773117e-9	1.0000000037252903
2^{-29}	0.11694222688674927	5.480178888461751e-8	1.0000000018626451
2^{-30}	0.11694216728210449	1.1440643366000813e-7	1.0000000009313226
⋮	⋮	⋮	
2^{-48}	0.09375	0.023192281688538152	1.0000000000000036
2^{-49}	0.125	0.008057718311461848	1.0000000000000018
2^{-50}	0.0	0.11694228168853815	1.0000000000000009
2^{-51}	0.0	0.11694228168853815	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

Wnioski

Biorąc pod uwagę przedział h $\in \langle 2^0, 2^{-28} \rangle$ można zauważyć, że błąd maleje dając tym samym dla h $= 2^{-28}$ najmniejszy błąd dla wszystkich rozpatrywanych przez nas wartości h, zatem jest to wynik najbliższy prawdzie. Od h $= 2^{-29}$ błąd się tylko zwiększa.

Jak wytłumaczyć, że od pewnego momentu zmniejszanie wartości h nie poprawia przybliżenia wartości pochodnej?

Prawdopodobnie dlatego, że h staje się tak małe, że wykonywane działania sprawiają, że występuje utrata cyfr znaczących. A od pewnego momentu h jest tak małe, że zaczyna przyjmować wartość równą 0, mimo, że we wzorze h powinno dążyć do 0, ale nigdy go nie osiągnąć. Zatem nie osiągniemy lepszego przybliżenia wartości pochodnej.

Jak zachowują sie wartości 1+h?

Maleją, jako że h maleje.