Obliczenia Naukowe - sprawozdanie nr 1

Wiktor Bachta

Październik 2024

1 Zadanie 1

1.1 Opis

Wyznaczanie iteracyjne liczb *macheps, eta, max* dla różnych typów arytmetyki zmiennoprzecinkowej.

1.2 Rozwiązanie

Liczby *eta* oraz *macheps* wyznaczam przez iteracyjne dzielenie przez 2. Liczbę *max* wyznaczam przez mnożenie przez 2, a następnie powrót funkcją *prevfloat*

1.3 Wynik

typ	*macheps*	*eps(typ)*	wartość z 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

typ	*eta*	*nextfloat(typ(0))*
Float16	6.0e-8	6.0e-8
Float32	1.0e-45	1.0e-45
Float64	5.0e-324	5.0e-324

typ	*max*	*floatmax(typ)*	wartość z float.h
Float16	6.55e4	6.55e4	-
Float32	3.4028235e38	3.4028235e38	3.402823e+38
Float64	1.7976931348623157e308	1.7976931348623157e308	1.797693e+308

typ	*floatmin(typ)*	Min_{nor}
Float32	1.1754944e-38	1.1754944e-38
Float64	2.2250738585072014e-308	2.2250738585072014e-308

1.4 Wnioski

- Iteracyjne obliczenia dają prawidłowe wartości dla *macheps, eta, max*
- Jaki związek ma liczba *macheps* z precyzją arytmetyki? $macheps = 2\epsilon$ - maksymalna odległość od reprezentowalnej liczby jest w środku dwóch kolejnych liczb
- Jaki związek ma liczba eta z liczbą MIN_{sub}? Są tożsame - najmniejsza liczba zdenormalizowana.
- Co zwracają funkcje $floatmin(Float32)$ i $floatmin(Float64)$ i jaki jest związek zwracanych wartości z liczbą MIN_{nor}? Są tożsame - najmniejsza liczba znormalizowana.

2 Zadanie 2

2.1 Opis

Wyznaczenie epsilonu maszynowego *macheps* na podstawie wzoru

$$macheps = 3 * (4/3 - 1) - 1$$

2.2 Wynik

Typ	*macheps*	*macheps* wg wzoru Kahana
Float16	0.000977	-0.000977
Float32	1.1920929e-7	1.1920929e-7
Float64	2.220446049250313e-16	-2.220446049250313e-16

2.3 Wnioski

Wzór Kahana może być wykorzystany do obliczenia *macheps*, przy zastosowaniu wartości bezwzględnej. Różnice znaków wynikają z miejsca "ucięcia" liczby $4/3$ w odpowiedniej reprezentacji. Dla Float16 oraz Float64 jest ona zaokrąglana w dół (...01), natomiast dla Float32 w górę (...11).

3 Zadanie 3

3.1 Opis

Sprawdzenie rozmieszczenia liczb typu Float64 w przedziale $[1, 2]$. Każda liczba x w tym przedziale może być przedstawiona jako $x + \delta k$, gdzie $k = 1, 2, ..., 2^{52} - 1$ i $\delta = 2^{-52}$ Podobne rozważania dla przedziałów $[1/2, 1]$ oraz $[2, 4]$

3.2 Rozwiązanie

Przenalalizuję reprezentację bitową liczb w przedziale, z najmniejszym możliwym krokiem $\delta = nextfloat(x)$

3.3 Wynik

liczba	*bitstring* (liczba)
1.0	0011111111110000000000000000000000000000000000000000000000000000
1.0 + δ	0011111111110000000000000000000000000000000000000000000000000001
1.0 + 2 δ	0011111111110000000000000000000000000000000000000000000000000010
1.0 + 3 δ	0011111111110000000000000000000000000000000000000000000000000011
1.0 + 4 δ	0011111111110000000000000000000000000000000000000000000000000100

Table 1: $\delta = 2^{-52}$

liczba	*bitstring* (liczba)
0.5	0011111111100000000000000000000000000000000000000000000000000000
0.5 + δ	0011111111100000000000000000000000000000000000000000000000000001
0.5 + 2 δ	0011111111100000000000000000000000000000000000000000000000000010
0.5 + 3 δ	0011111111100000000000000000000000000000000000000000000000000011
0.5 + 4 δ	0011111111100000000000000000000000000000000000000000000000000100

Table 2: $\delta = 2^{-53}$

liczba	*bitstring* (liczba)
2.0	0100000000000000000000000000000000000000000000000000000000000000
2.0 + δ	0100000000000000000000000000000000000000000000000000000000000001
2.0 + 2 δ	0100000000000000000000000000000000000000000000000000000000000010
2.0 + 3 δ	0100000000000000000000000000000000000000000000000000000000000011
2.0 + 4 δ	0100000000000000000000000000000000000000000000000000000000000100

Table 3: $\delta = 2^{-51}$

3.4 Wnioski

Ze względu na to, że dane przedziały są w postaci $[2^t, 2^{t+1}]$ cecha dla każdej liczby z przedziału będzie taka sama. Czyli liczba jest tożsama z mantysą, którą możemy ułożyć na 2^{52} sposobów. Najmniejszy krok jest wtedy równy $2^{-(52+t)}$ i liczby są ułożone w przedziale jednostajnie. Oznacza to, że precyzja reprezentacji jest proporcjonalna do odległości liczby od zera - dokładniejsza dla mniejszych liczb.

4 Zadanie 4

4.1 Opis

Znaleźć najmniejszą liczbę w arytmetyce Float64 $1 < x < 2$, dla której $x(1/x) \neq 1$

4.2 Rozwiązanie

Iteracyjne przechodzenie po liczbach za pomocą *nextfloat* do odnalezienia liczby o szukanej własności

4.3 Wynik

Znaleziona liczba: 1.000000057228997
$x(1/x) = 0.9999999999999999$

4.4 Wnioski

Liczba $1/x$ nie ma dokładnej reprezentacji w arytmetyce Float, więc ulega zaokrągleniu do najbliższej możliwej liczby, co powoduje błąd. Nawet najprostrze dzielenie i monożenie wprowadza potencjalne błędy, a komputer nie potrafi "od tak" upraszczać wyrażeń.

5 Zadanie 5

5.1 Opis

Obliczanie iloczynu skalarnego dwóch wektorów za pomocą czterech metod

$$x = [2.718281828, -3.141592654, 1.414213562, 0.5772156649, 0.3010299957]$$

$$y = [1486.2497, 878366.9879, -22.37492, 4773714.647, 0.000185049]$$

- Obliczanie i sumowanie iloczynów po kolei
- Obliczanie i sumowanie iloczynów w odwrotnej kolejności
- Od największego do najmniejszego (dodaj dodatnie liczby w porządku od największego do najmniejszego, dodaj ujemne liczby w porządku od najmniejszego do największego, a następnie daj do siebie obliczone sumy częściowe)
- od najmniejszego do największego

5.2 Wynik

Typ	metoda 1	metoda 2	metoda 3	metoda 4
Float32	-0.4999443	-0.4543457	-0.5	-0.5
Float64	1.0251881368296672e-10	-1.5643308870494366e-10	0.0	0.0

5.3 Wnioski

Rzeczywista wartość iloczynu wynosi $-1.0065710700000010^{-11}$ Obie arytmetyki dały wyniki dalekie rzeczywitej wartości dla każdej z metod. Wszystkie błędy względne przekraczają tutaj 100%. Oznacza to, że nawet w podstawowych operacjach (dodawanie, mnożenie) kolejność wykonywanych działań ma znaczący wpływ na otrzymywane rezultaty. Należy stosować odpowiednie metody/algorytmy w zależności o wykonywanych obliczeń.

6 Zadanie 6

6.1 Opis

Obliczenie i porównanie wartości dwóch funkcji, które są sobie równe, ale wykorzystują inne wzory

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

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

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

6.2 Wynik

x	$f(x)$	$g(x)$	$x^2/2$
8^{-1}	0.0077822185373186414	0.0077822185373187065	0.0078125
8^{-2}	0.00012206286282867573	0.00012206286282875901	0.0001220703125
8^{-3}	1.9073468138230965e-6	1.907346813826566e-6	1.9073486328125e-6
8^{-4}	2.9802321943606103e-8	2.9802321943606116e-8	2.9802322387695312e-8
8^{-5}	4.656612873077393e-10	4.6566128719931904e-10	4.656612873077393e-10
8^{-6}	7.275957614183426e-12	7.275957614156956e-12	7.275957614183426e-12
8^{-7}	1.1368683772161603e-13	1.1368683772160957e-13	1.1368683772161603e-13
8^{-8}	1.7763568394002505e-15	1.7763568394002489e-15	1.7763568394002505e-15
8^{-9}	0.0	2.7755575615628914e-17	2.7755575615628914e-17
8^{-10}	0.0	4.336808689942018e-19	4.336808689942018e-19
...	...	...	...
8^{-177}	0.0	1.012e-320	1.012e-320
8^{-178}	0.0	1.6e-322	1.6e-322
8^{-179}	0.0	0.0	0.0

6.3 Wnioski

Analiza liczby $\sqrt{x^2+1}$ pozwala na dokładniejsze zrozumienie obliczeń. Jako, że x jest bliski zeru, x^2 staje się bardzo mała i dodawanie z jedynką daje rezultat równy jeden. Cecha $(8^{-9})^2 = 2^{-54}$ to -54, cecha jedynki to 0, czyli ich różnica przekracza długość mantysy. Dlatego funkcja f daje wartości równe 0, a g w rzeczywistości zwraca $x^2/2$ Z tego powodu, wiarygodniejsza jest funkcja g. Wnioskiem jest fakt, że dla dużych różnic wielkości liczb $x + y = x$, na co należy uważać przy obliczeniach. Drugi wniosek to fakt, że dla pewnych obliczeń można znaleźć alternatywny wzór, który działa z większą dokładnością.

7 Zadanie 7

7.1 Opis

Przybliżanie pochodnej w punkcie za pomocą dedfinicji, dla coraz mniejszych h i porównanie z wartością rzeczywistą w punkcie $x = 1$.

$$f(x) = sin(x) + cos(3x)$$

$$f'(x) = cos(x) - 3sin(3x)$$

$$\widetilde{f}(x) = \frac{f(x+h) - f(x)}{h}$$

$$h = 2^{-n}, n = 0, 1, 2, ..., 54$$

7.2 Wynik

h	$h+1$	$\widetilde{f}(x)$	$\lvert f'(x) - \widetilde{f}(x)\rvert$
2^0	2.0	2.0179892252685967	1.9010469435800585
2^{-1}	1.5	1.8704413979316472	1.753499116243109
2^{-2}	1.25	1.1077870952342974	0.9908448135457593
...	...	...	...
2^{-26}	1.0000000149011612	0.11694233864545822	5.6956920069239914e-8
2^{-27}	1.0000000074505806	0.11694231629371643	3.460517827846843e-8
2^{-28}	1.0000000037252903	0.11694228649139404	4.802855890773117e-9
...	...	...	...
2^{-52}	1.0000000000000002	-0.5	0.6169422816885382
2^{-53}	1.0	0.0	0.11694228168853815
2^{-54}	1.0	0.0	0.11694228168853815

7.3 Wnioski

Najlepsze przybliżenie otrzymujemy dla $n = 28$, potem wartość błędu rośnie. Dla $h <= 2^{-53}, h+1 = 1$ co psuje przybliżenie pochodnej, dając wynik 0. Jak wytłumaczyć, że od pewnego momentu zmniejszanie wartości h nie poprawia przybliżenia wartości pochodnej? Błędy zaokrągleń: Gdy h jest zbyt małe, wyrażenie $f(x+h)-f(x)$ staje się podatne na błędy zaokrągleń. Różnica jest liczona na liczbach o małej różnicy wartości, co prowadzi do coraz większego

wpływu błędów numerycznych. Wniosek: obliczenia maszynowne rzadko zachowują się intuicyjnie, i dosłowne traktowanie matematycznych granic, może dawać gorsze rezultaty dla teoretycznie lepszych (dokładniejszych) wartości. Odejmownie liczb barzo bliskich powoduje dużą utratę dokładności.