Obliczenia naukowe

Wojciech Maziarz

Październik 2022

1 Zadanie 1

1.1 Wyznaczanie macheps

Arytmetyka	Macheps	eps()	float.h
Float16	0.000 977	0.000 977	brak
Float32	$1.192\,092\,9 \times 10^{-7}$	$1.192\,092\,9 \times 10^{-7}$	$1.192\,093 \times 10^{-7}$
Float64	$2.220\,446\,049\,250\,313 \times 10^{-16}$	$2.220\,446\,049\,250\,313 \times 10^{-16}$	$2.220\,446 \times 10^{-16}$

1.2 Związek macheps z precyzją arytmetyki

Precyzja arytmetyki to połowa machepsu.

1.3 Wyznaczanie eta

Arytmetyka	Eta	nextfloat(0)
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}

1.4 Związek eta z MIN_{sub}

Eta jest równy MIN_{sub}, czyli najmniejszej liczbie nieznormalizowanej.

1.5 Związek floatmin() a MIN_{nor}

Funkcja floatmin() zwraca najmniejszą liczbę znormalizowaną, czyli MIN_{nor}.

1.6 Wyznaczanie MAX

Arytmetyka	max	floatmax()	float.h
Float16	6.55×10^{4}	6.55×10^{4}	brak
Float32	$3.402\,823\,5 \times 10^{38}$	$3.402\,823\,5 \times 10^{38}$	3.4×10^{38}
Float64	$1.797\,693\,134\,862\,315\,7 \times 10^{308}$	$1.797\,693\,134\,862\,315\,7 \times 10^{308}$	1.8×10^{308}

2 Zadanie 2

2.1 Wzór Kahana

Kahan stwierdził, że epsilon maszynowy (macheps) można otrzymać obliczając wyrażenie: $macheps = 3 \cdot (\frac{4}{3} - 1) - 1$. Mamy eksperymentalnie sprawdzić poprawność tego wyrażenia.

2.2 Wyniki

Arytmetyka	Wzór Kahana	Macheps
Float16	$-0.000\,977$	$0.000\,977$
Float32	$1.192\,092\,9 \times 10^{-7}$	$1.192\,092\,9 \times 10^{-7}$
Float64	$-2.220\,446\,049\,250\,313 \times 10^{-16}$	$2.220\,446\,049\,250\,313 \times 10^{-16}$

2.3 Wnioski

Wzór Kahana wynosi tyle samo co macheps z dokładnością do znaku.

3 Zadanie 3

3.1 Problem

Eksperymentalnie sprawdzić w arytmetyce Float64 równomierność rozmieszczenia liczb w przedziałach $(1, 2)$, $(\frac{1}{2}, 1)$ oraz $(2, 4)$

3.2 Rozwiązanie

Dla danego x możemy obliczyć różnicę $x - nextfloat(x)$.

wyrażenie	wartość
x	1.0
$bitstring(x)$	0011111111110000000000000000000000000000000000000000000000000000
$bitstring(nextfloat(x))$	0011111111110000000000000000000000000000000000000000000000000001
$x - nextfloat(x)$	$2.220\,446\,049\,250\,313 \times 10^{-16}$
2^{-52}	$2.220\,446\,049\,250\,313 \times 10^{-16}$

2^{-52} jest odległością pomiędzy kolejnymi wartościami w przedziale $(1, 2)$

wyrażenie	wartość
x	2.0
$bitstring(x)$	0100000000000000000000000000000000000000000000000000000000000000
$bitstring(nextfloat(x))$	0100000000000000000000000000000000000000000000000000000000000001
$x - nextfloat(x)$	$4.440\,892\,098\,500\,626 \times 10^{-16}$
2^{-51}	$4.440\,892\,098\,500\,626 \times 10^{-16}$

2^{-51} jest odległością pomiędzy kolejnymi wartościami w przedziale $(2, 4)$

wyrażenie	wartość
x	2.0
$bitstring(x)$	0011111111110000000000000000000000000000000000000000000000000000
$bitstring(nextfloat(x))$	0011111111110000000000000000000000000000000000000000000000000001
$x - nextfloat(x)$	$1.110\,223\,024\,625\,156\,5 \times 10^{-16}$
2^{-53}	$1.110\,223\,024\,625\,156\,5 \times 10^{-16}$

2^{-53} jest odległością pomiędzy kolejnymi wartościami w przedziale $(\frac{1}{2}, 1)$

3.3 Wnioski

Dla danego przedziału $(2^n, 2^{n+1})$, odległość pomiędzy kolejnymi liczbami wynosi 2^{52-n}

4 Zadanie 4

4.1 Problem

Należy eksperymentalnie znaleźć liczbę zmiennopozycyjną $x \in (1, 2)$ taką że $x \cdot \frac{1}{x} \neq 1$ w arytmetyce Float64.

4.2 Rozwiązanie

Wiemy z poprzedniego zadania że w przedziale $(1, 2)$ liczby są równomiernie rozmieszczone z krokiem $\delta = 2^{-52}$. Korzystając z tej wiedzy możemy kolejnych $x = 1 + k \cdot \delta$ sprawdzać nierówność $x \cdot \frac{1}{x} \neq 1$.

4.3 Wyniki

x	$x \cdot (1/x) \neq 1$
1.000 000 057 228 997	0.999 999 999 999 999 9
1.000 000 066 222 211	0.999 999 999 999 999 9
1.000 000 069 494 391 8	0.999 999 999 999 999 9
1.000 000 071 074 011 6	0.999 999 999 999 999 9
1.000 000 083 300 026 9	0.999 999 999 999 999 9
1.000 000 099 123 532 7	0.999 999 999 999 999 9
1.000 000 105 103 379	0.999 999 999 999 999 9
1.000 000 107 195 193 6	0.999 999 999 999 999 9
1.000 000 110 258 53	0.999 999 999 999 999 9
1.000 000 115 183 187 4	0.999 999 999 999 999 9
...	...

Najmniejszą znalezioną liczbą jest: 1.000 000 057 228 997

5 Zadanie 5

5.1 Problem

Mamy zaimplementować 4 rodzaje algorytmów obliczających iloczyn skalarny podanych wektorów
$x = [2.718281828, -3.141592654, 1.414213562, 0.5772156649, 0.3010299957]$
$y = [1486.2497, 878366.9879, -22.37492, 4773714.647, 0.000185049]$
(a) w przód
(b) w tył
(c) 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)
(d) od najmniejszego do największego (przeciwnie do metody (c)).

5.2 Wyniki

Algorytm	Float32	Float64
A	−0.499 944 3	$1.025\,188\,136\,829\,667\,2 \times 10^{-10}$
B	−0.454 345 7	$-1.564\,330\,887\,049\,436\,6 \times 10^{-10}$
C	-0.5	0.0
D	-0.5	0.0

5.3 Wnioski

Różne algorytmy iloczynu skalarnego dają różniące się wyniki.

6 Zadanie 6

6.1 Problem

Mamy podane dwie funkcje $f(x) = \sqrt{x^2 + 1} - 1$ $g(x) = \frac{x^2}{\sqrt{x^2+1}+1}$, gdzie f(x) = g(x), mamy dla kolejnych wartości argumentów $x = 8^{-1}, 8^{-2}, 8^{-3}$...

6.2 Wyniki

Wyniki zostały obliczone w arytmetyce $Float64$.

x	f(x)	g(x)
8^{-1}	0.007 782 218 537 318 641 4	0.007 782 218 537 318 706 5
8^{-2}	0.000 122 062 862 828 675 73	0.000 122 062 862 828 759 01
8^{-3}	$1.907\,346\,813\,823\,096\,5 \times 10^{-6}$	$1.907\,346\,813\,826\,566 \times 10^{-6}$
8^{-4}	$2.980\,232\,194\,360\,610\,3 \times 10^{-8}$	$2.980\,232\,194\,360\,611\,6 \times 10^{-8}$
8^{-5}	$4.656\,612\,873\,077\,393 \times 10^{-10}$	$4.656\,612\,871\,993\,190\,4 \times 10^{-10}$
8^{-6}	$7.275\,957\,614\,183\,426 \times 10^{-12}$	$7.275\,957\,614\,156\,956 \times 10^{-12}$
8^{-7}	$1.136\,868\,377\,216\,160\,3 \times 10^{-13}$	$1.136\,868\,377\,216\,095\,7 \times 10^{-13}$
8^{-8}	$1.776\,356\,839\,400\,250\,5 \times 10^{-15}$	$1.776\,356\,839\,400\,248\,9 \times 10^{-15}$
8^{-9}	0.0	$2.775\,557\,561\,562\,891\,4 \times 10^{-17}$
8^{-10}	0.0	$4.336\,808\,689\,942\,018 \times 10^{-19}$
...	...	...
8^{-175}	0.0	$4.144\,523 \times 10^{-317}$
8^{-176}	0.0	6.4758×10^{-319}
8^{-177}	0.0	1.012×10^{-320}
8^{-178}	0.0	1.6×10^{-322}
8^{-179}	0.0	0.0
8^{-180}	0.0	0.0
...	...	...

6.3 Wnioski

Funkcja $g(x)$ daje bardziej wiarygodne wyniki, dzieje się tak ponieważ w funkcji $f(x)$ występuje odejmowanie liczb $\sqrt{x^2+1}$ i 1, które dla bardzo małego x są bardzo zbliżone, co zwiększa błąd obliczeń poprzez utratę znaczących bitów.

7 Zadanie 7

7.1 Problem

Mamy obliczyć w arytmetyce Float64 przybliżoną wartość pochodnej funkcji f korzystając za pomocą następującego wzoru: $f^{'}(x_0) \approx f^{'}(x_0) = \frac{f(x_0+h)-f(x_0)}{h}$, gdzie
$f(x) = sin(x) + cos(3 \cdot x)$
$f^{'}(x) = g(x) = cos(x) - 3 \cdot sin(3 \cdot x)$

7.2 Wyniki

i	wzór	$f^{'}(x)$	$\lvert f^{'}(x) - g(x)\rvert$
0.0	2.0	2.017 989 225 268 596 7	1.901 046 943 580 058 5
−1.0	1.5	1.870 441 397 931 647 2	1.753 499 116 243 109
−2.0	1.25	1.107 787 095 234 297 4	0.990 844 813 545 759 3
−3.0	1.125	0.623 241 279 297 581 7	0.506 298 997 609 043 5
−4.0	1.0625	0.370 400 066 203 519 2	0.253 457 784 514 981
−5.0	1.031 25	0.243 443 074 397 546 87	0.126 500 792 709 008 7
−6.0	1.015 625	0.180 097 563 307 327 85	0.063 155 281 618 789 7
−7.0	1.007 812 5	0.148 491 395 371 095 8	0.031 549 113 682 557 64
−8.0	1.003 906 25	0.132 709 114 280 515 9	0.015 766 832 591 977 753
−9.0	1.001 953 125	0.124 823 692 940 708 5	0.007 881 411 252 170 345
−10.0	1.000 976 562 5	0.120 882 476 811 061 68	0.003 940 195 122 523 526 5
−11.0	1.000 488 281 25	0.118 912 250 468 838 47	0.001 969 968 780 300 313
−12.0	1.000 244 140 625	0.117 927 233 739 010 26	0.000 984 952 050 472 109 9
−13.0	1.000 122 070 312 5	0.117 434 749 610 765 72	0.000 492 467 922 227 568 5
−14.0	1.000 061 035 156 25	0.117 188 513 620 931 19	0.000 246 231 932 393 037 3
−15.0	1.000 030 517 578 125	0.117 065 397 145 779 57	0.000 123 115 457 241 418 37
−16.0	1.000 015 258 789 062 5	0.117 003 839 288 372 55	$6.155\,759\,983\,439\,424 \times 10^{-5}$
−17.0	1.000 007 629 394 531 2	0.116 973 060 459 713 45	$3.077\,877\,117\,529\,937 \times 10^{-5}$
−18.0	1.000 003 814 697 265 6	0.116 957 671 067 211 78	$1.538\,937\,867\,362\,477\,6 \times 10^{-5}$
−19.0	1.000 001 907 348 632 8	0.116 949 976 363 684 98	$7.694\,675\,146\,829\,866 \times 10^{-6}$
−20.0	1.000 000 953 674 316 4	0.116 946 129 011 921 58	$3.847\,323\,383\,432\,410\,5 \times 10^{-6}$
−21.0	1.000 000 476 837 158 2	0.116 944 205 248 728 4	$1.923\,560\,190\,242\,312\,7 \times 10^{-6}$
−22.0	1.000 000 238 418 579	0.116 943 242 959 678 17	$9.612\,711\,400\,208\,696 \times 10^{-7}$
−23.0	1.000 000 119 209 289 6	0.116 942 762 397 229 67	$4.807\,086\,915\,192\,826 \times 10^{-7}$
−24.0	1.000 000 059 604 644 8	0.116 942 521 184 682 85	$2.394\,961\,446\,938\,737 \times 10^{-7}$
−25.0	1.000 000 029 802 322 4	0.116 942 398 250 103	$1.165\,615\,648\,446\,305\,4 \times 10^{-7}$
−26.0	1.000 000 014 901 161 2	0.116 942 338 645 458 22	$5.695\,692\,006\,923\,991\,4 \times 10^{-8}$
−27.0	1.000 000 007 450 580 6	0.116 942 316 293 716 43	$3.460\,517\,827\,846\,843 \times 10^{-8}$
−28.0	1.000 000 003 725 290 3	0.116 942 286 491 394 04	$4.802\,855\,890\,773\,117 \times 10^{-9}$
−29.0	1.000 000 001 862 645 1	0.116 942 226 886 749 27	$5.480\,178\,888\,461\,751 \times 10^{-8}$
−30.0	1.000 000 000 931 322 6	0.116 942 167 282 104 49	$1.144\,064\,336\,600\,081\,3 \times 10^{-7}$
−31.0	1.000 000 000 465 661 3	0.116 942 167 282 104 49	$1.144\,064\,336\,600\,081\,3 \times 10^{-7}$
−32.0	1.000 000 000 232 830 6	0.116 941 928 863 525 39	$3.528\,250\,127\,615\,706\,3 \times 10^{-7}$
−33.0	1.000 000 000 116 415 3	0.116 941 452 026 367 19	$8.296\,621\,709\,646\,956 \times 10^{-7}$
−34.0	1.000 000 000 058 207 7	0.116 941 452 026 367 19	$8.296\,621\,709\,646\,956 \times 10^{-7}$
−35.0	1.000 000 000 029 103 8	0.116 939 544 677 734 38	$2.737\,010\,803\,777\,195\,6 \times 10^{-6}$
−36.0	1.000 000 000 014 552	0.116 943 359 375	$1.077\,686\,461\,847\,804\,4 \times 10^{-6}$
−37.0	1.000 000 000 007 276	0.116 928 100 585 937 5	$1.418\,110\,260\,065\,219\,6 \times 10^{-5}$
−38.0	1.000 000 000 003 638	0.116 943 359 375	$1.077\,686\,461\,847\,804\,4 \times 10^{-6}$
−39.0	1.000 000 000 001 819	0.116 882 324 218 75	$5.995\,746\,978\,815\,219\,6 \times 10^{-5}$
−40.0	1.000 000 000 000 909 5	0.116 821 289 062 5	0.000 120 992 626 038 152 2
−41.0	1.000 000 000 000 454 7	0.116 943 359 375	$1.077\,686\,461\,847\,804\,4 \times 10^{-6}$
−42.0	1.000 000 000 000 227 4	0.116 699 218 75	0.000 243 062 938 538 152 2
−43.0	1.000 000 000 000 113 7	0.116 210 937 5	0.000 731 344 188 538 152 2
−44.0	1.000 000 000 000 056 8	0.117 187 5	0.000 245 218 311 461 847 8
−45.0	1.000 000 000 000 028 4	0.113 281 25	0.003 661 031 688 538 152
−46.0	1.000 000 000 000 014 2	0.109 375	0.007 567 281 688 538 152
−47.0	1.000 000 000 000 007	0.109 375	0.007 567 281 688 538 152
−48.0	1.000 000 000 000 003 6	0.093 75	0.023 192 281 688 538 152
−49.0	1.000 000 000 000 001 8	0.125	0.008 057 718 311 461 848
−50.0	1.000 000 000 000 000 9	0.0	0.116 942 281 688 538 15
−51.0	1.000 000 000 000 000 4	0.0	0.116 942 281 688 538 15
−52.0	1.000 000 000 000 000 2	−0.5	0.616 942 281 688 538 2
−53.0	1.0	0.0	0.116 942 281 688 538 15
−54.0	1.0	0.0	0.116 942 281 688 538 15

Funkcja do obliczania pochodnej dla danej funkcji jest najdokładniejsza dla $i = -28$, a jej błąd wynosi $4.802\,855\,890\,773\,117 \times 10^{-9}$.

7.3 Wnioski

Zmniejszając wartość h wartość pochodnej staje się coraz dokładniejsza, ale ponieważ wraz ze zmniejszaniem h liczby $f(x_0+h)$ i $f(x_0)$ stają się bardzo bliskie siebie przez co błąd w ich odejmowaniu zwiększa się. $h = 2^{-28}$ jest kompromisem pomiędzy błędami wynikającymi ze zbyt dużego i zbyt małego h.