Obliczenia Naukowe
Lista 1

Paweł Polerowicz

254626

Październik 2021

Zadanie 1.
Zadanie składało się z kilku podpunktów, każdy odpowiadający oddzielnemu programowi. Każdy z nich wyznaczał pewną liczbę dla typów Float16, Float32, Float64.

1. Celem pierwszego programu było znalezienie *macheps*, czyli najmniejszej liczby takiej, że *macheps* $>$ 0, *fl(1.0+macheps)* $>$ 1.0 i *fl(1.0+macheps)* $=$ 1 + *macheps*. W tym celu zastosowano podejście iteracyjnie, dodając do 1.0 stopniowo coraz mniejszą liczbę i sprawdzając, czy nadal spełnione są warunki z założeń. Ostatnia liczba spełniająca warunki to *macheps*.

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

Dane są spójne dla wszystkich trzech źródeł, z zastrzeżeniem, że float.h nie zawiera informacji o epsilonie dla 16 bitowego typu. Ponadto, możemy zauważyć $macheps = 2 * \epsilon$ (precyzja arytmetyki)

typ	ϵ
Float16	4.88×10^{-4}
Float32	5.96×10^{-8}
Float64	1.11×10^{-16}

2. Kolejny program wyznacza liczbę η, czyli najmniejszą liczbę taką, że $\eta > 0$. Ponownie zastosowano podejście iteracyjne, zaczynając od 1 i dzieląc w pętli przez 2 dopóki warunek był spełniony.
Znalezione wartości pozwalają wysnuć wniosek: $\eta \approx Min_{sub}$

typ	η	nextfloat(typ(0.0))	Min_{sub}
Float16	5.96×10^{-8}	5.96×10^{-8}	
Float32	1.40×10^{-45}	1.40×10^{-45}	1.4×10^{-45}
Float64	4.94×10^{-324}	4.94×10^{-324}	4.9×10^{-325}

3. Sprawdziliśmy również wyniki zwracane przez funkcję floatmin
Znalezione wartości Min_{nor} są mniej dokładne, ale pozwalają wysnuć wniosek: $floatmin \approx Min_{nor}$

typ	floatmin(typ)	Min_{nor}
Float32	$1.1754944 \times 10^{-38}$	1.2×10^{-38}
Float64	$2.2250738585072014 \times 10^{-308}$	2.2×10^{-308}

4. Kolejny program znajduje iteracyjnie liczbę MAX dla wszystkich trzech typów. Algorytm próbuje dotrzeć do MAX poprzez stopniowe dodawanie do dużej liczby liczb coraz mniejszych, monitorując, czy otrzymujemy już nieskończoność oraz czy reprezentacja liczby zmienia się po wykonaniu dodawania. Reprezentacja binarna znalezionych liczb to 0 (bit znaku) i same 1.

typ	$\boldsymbol{MAX}$	**floatmax(typ)**	**float.h**
Float16	6.55×10^{4}	6.55×10^{4}	-
Float32	3.4028235×10^{38}	3.4028235×10^{38}	3.402823×10^{38}
Float64	$1.7976931348623157 \times 10^{308}$	$1.7976931348623157 \times 10^{308}$	1.797693×10^{308}

Zadanie 2.

Obliczyliśmy wyrażenie $3(4/3 - 1) - 1$, aby spradzić czy, zgodnie ze stwierdzeniem Kahana, otrzymamy *macheps*.

typ	$fl(3(4/3 - 1) - 1)$
Float16	-0.000977
Float32	1.1920929×10^{-7}
Float64	$-2.220446049250313 \times 10^{-16}$

Rzeczywiście, otrzymane wartości zgadzają się (z dokładnością do wartości bezwględnej) z *macheps*.

Zadanie 3.

Naszym celem było potwierdzenie, że w arytmetyce Float64 liczby na przedziale $[1, 2]$ są rozmieszczone równomiernie z krokiem $\delta = 2^{-52}$. Przedstawimy wybrane reprezentacje liczb, zwrócone przez fukncję bitstring.

liczba	**reprezentacja**
δ	0011110010110000000000000000000000000000000000000000000000000000
1.0	0011111111110000000000000000000000000000000000000000000000000000
$1.0 + \delta$	0011111111110000000000000000000000000000000000000000000000000001
2.0	0100000000000000000000000000000000000000000000000000000000000000
$2.0 - \delta$	0011111111111111111111111111111111111111111111111111111111111111

Oczywiście łatwo zauważyć, że 1 i 2 mają taką samą mantysę a ich cecha różni się o 1. Stąd wiemy, że pomiędzy nimi musi znajdować się $2^{52} - 1$ innych liczb, bo mantysa ma 52 cyfry. Zauważmy ponadto, że dodanie δ do 1.0 powoduje zmianę reprezentacji na ostatnim bicie, natomiast obliczając takie wyrażenie w większej precyzji zauważylibyśmy, że jest to wynik dokładny. Eksperymenty potwierdzają, że dodając do 1.0 kolejne wielokrotności δ otrzymamy wszystkie możliwe liczby w tym przedziale. Na podobnej zasadzie, spradziliśmy, że dla przedziału $[0.5, 1]$ krok wynosi 2^{-53} a dla $[2, 4]$ wynosi on 2^{-51}, ponieważ ponownie mamy $2^{-52} - 1$ liczb pomiędzy, ale są one rozmieszczone odpowiednio dwukrotnie gęściej i dwuktrotnie rzadziej.

Zadanie 4.

W tym zadaniu próbowaliśmy znaleźć najmniejszą liczbę $x, 1 < x < 2$ taką, że $x(1/x) \neq 1$. Wykorzystaliśmy wcześniej potwierdzoną równomierność rozmieszczenia liczb na $[1, 2]$, iteracyjnie dodając 2^{-52} i sprawdzając warunek. Znależliśmy liczbę **1.000000057228997**, dla której zwracany wynik wyrażenia to 0.9999999999999999.

Zadanie 5.
Naszym zadaniem było obliczenie na różne sposoby iloczynu skalarnego wektorów:
$x = [2.718281828, -3.141592654, 1.414213562, 0.5772156649, 0.3010299957]$
$y = [1486.2497, 878366.9879, -22.37492, 4773714.647, 0.000185049]$

typ	"w przód"	"w tył"	najw. - najmn.	najmn. - najw.
Float32	-0.4999443	1.1920929×10^{-7}	-0.5	-0.5
Float64	$1.0251881368296672 \times 10^{-10}$	$-1.5643308870494366 \times 10^{-10}$	0.0	0.0

Niestety, wszystkie metody dają wyniki wyraźnie różne od dokładnej wartości $-1.00657107000000 \times 10^{-11}$. W przypadku dwóch ostatnich metod wynika to prawdopodobnie z faktu, że dodajemy do siebie duże sumy częściowe o przeciwnych znakach, bliskie co do wartości bezwzględnej. W pierwszych dwóch metodach dodajemy do siebie liczby o dużej i małej wartości bezględnej, co również wprowadza niedokładność.

Zadanie 6.
Naszym zadaniem było policzenie wartości funkcji
$f(x) = \sqrt{x^2+1} - 1$
$g(x) = x^2/(\sqrt{x^2+1} + 1)$
dla $x = 8^-1, 8^-2...$ Okazało się, że funkcje f i g dają zbliżone wyniki dla wykładnika i: $i >= -8$. Dla mniejszych i f daje wynik 0, natomiast g daje wiarygodne wyniki aż do $i = 178$. Wynika to z redukcji cyfr znaczących przy odejmowaniu w funkcji f

Zadanie 7.
Zadanie polegało na obliczeniu przybliżonej wartości pochodnej wyrażenia $f(x) = \sin x + \cos 3x$ za pomocą wzoru $f'(x) \approx \frac{f(x+h)-f(x)}{h}$ dla $h = 2^{-n}(n = 0, 1, \ldots, 54)$. Dla przejrzystości przedstawimy tylko wybrane wartości n.

n	przybliżenie	błąd
5	0.24344307439754687	0.1265007927090087
10	0.12088247681106168	0.0039401951225235265
20	0.11694612901192158	$3.8473233834324105 \times 10^{-6}$
27	0.11694231629371643	$3.460517827846843 \times 10^{-8}$
28	0.11694228649139404	$4.802855890773117 \times 10^{-9}$
29	0.11694222688674927	$5.480178888461751 \times 10^{-8}$
40	0.1168212890625	0.0001209926260381522
54	0.0	0.11694228168853815

Najmniejszy błąd uzyskano dla n = 28. Powyżej tej wartości musimy liczyć się z utratą precyzji przy odejmowaniu $f(x+h) - f(x)$, gdyż ich wartości są bliskie siebie, co wpływa na dokładność całego wyniku.