OBLICZENIA NAUKOWE
Lista nr 1 (laboratorium)[1]

zad. 0

- Zainstalować język `Julia` (http://julialang.org) i IDE `Juno` (http://junolab.org).
- Przeczytać rozdział 1 z dokumentacji [1].

zad. 1 (Rozpoznanie arytmetyki)

Epsilonem maszynowym *macheps* (ang. machine epsilon) nazywamy najmniejszą liczbę $macheps > 0$ taką, że $1.0 + macheps > 1.0$.

Napisać program w języku `Julia` wyznaczający iteracyjnie epsilony maszynowe dla wszystkich dostępnych typów zmiennopozycyjnych `Float16, Float32, Float64`, zgodnych ze standardem IEEE 754 (`half, single, double`), i porównać z wartościami zwracanymi przez funkcje: `eps(Float16)`, `eps(Float32)`, `eps(Float64)` oraz z danymi zawartymi w pliku nagłówkowym `float.h` dowolnej instalacji języka `C`.

Napisać program w języku `Julia` wyznaczający **iteracyjnie** liczbę *eta* taką, że $eta > 0.0$ dla wszystkich typów zmiennopozycyjnych `Float16, Float32, Float64`, zgodnych ze standardem IEEE 754 (`half, single, double`), i porównać z wartościami zwracanymi przez funkcje: `nextfloat(float16(0.0))`, `nextfloat(float32(0.0))`, `nextfloat(float64(0.0))`

Wsk. Rozpocząć od jedynki i dzielić w pętli przez dwa. Weź pod uwagę konwersję typów (zob. rozdział 4 dokumentacji [1]).

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

Jaki związek ma liczba *eta* z liczbą MIN_{sub} (zob. wykład lub raport [2])?

Napisać program w języku `Julia` wyznaczający **iteracyjnie** liczbę (MAX) dla wszystkich typów zmiennopozycyjnych `Float16, Float32, Float64`, zgodnych ze standardem IEEE 754 (`half, single, double`), i porównać z wartościami zwracanymi przez funkcje: `realmax(Float16)`, `realmax(Float32)`, `realmax(Float64)` oraz z danymi zawartymi w pliku nagłówkowym `float.h` dowolnej instalacji języka `C` lub z danymi z wykładu lub zob. raport [2].

Wsk. Skorzystać z funkcji `isinf`. Weź również pod uwagę konwersję typów (zob. rozdział 4 dokumentacji [1]).

zad. 2 Kahan stwierdził, że epsilon maszynowy (*macheps*) można otrzymać obliczając wyrażenie $3(4/3-1)-1$ w arytmetyce zmiennopozycyjnej. Sprawdzić eksperymentalnie w języku `Julia` słuszność tego stwierdzenia dla wszystkich typów zmiennopozycyjnych `Float16, Float32, Float64`.

zad. 3 Sprawdź eksperymentalnie w języku `Julia`, że w arytmetyce `Float64` (arytmetyce `double` w standarcie IEEE 754) liczby zmiennopozycyjne są równomiernie rozmieszczone w $[1, 2]$ z krokiem $\delta = 2^{-52}$. Innymi słowy, każda liczba zmiennopozycyjna x pomiędzy 1 i 2 może być przedstawione następująco $x = 1 + k\delta$ w tej arytmetyce, gdzie $k = 1, 2, \ldots, 2^{52} - 1$ i $\delta = 2^{-52}$.

Jak rozmieszczone są liczby zmiennopozycyjne w przedziale $[\frac{1}{2}, 1]$, jak w przedziale$[2, 4]$ i jak mogą być przedstawione dla rozpatrywanego przedziału?

Wsk. Skorzystać z funkcji `bits`.

zad. 4

[1] Większość zadań pochodzi z książki: D. Kincaid, W. Cheney, Analiza numeryczna, WNT, 2005.

(a) Znajdź eksperymentalnie w arytmetyce `Float64` zgodnej ze standardem IEEE 754 (`double`) liczbę zmiennopozycyjną x w przedziale $1 < x < 2$, taką, że $x * (1/x) \neq 1$; tj. $fl(xfl(1/x)) \neq 1$ (napisz program w języku `Julia` znajdujący tę liczbę).

(b) Znajdź najmniejszą taką liczbę.

zad. 5 Napisz program w języku `Julia` realizujący następujący eksperyment obliczania iloczynu skalarnego dwóch wektorów:

$$\begin{aligned} x &= [2.718281828, -3.141592654, 1.414213562, 0.5772156649, 0.3010299957] \\ y &= [1486.2497, 878366.9879, -22.37492, 4773714.647, 0.000185049]. \end{aligned}$$

Zaimplementuj poniższe algorytmy i policz sumę na cztery sposoby dla $n = 5$:

(a) "w przód" $\sum_{i=1}^{n} x_i y_i$, tj. algorytm
$S := 0$
for $i := 1$ to n **do**
  $S := S + x_i * y_i$
end for

(b) "w tył" $\sum_{i=n}^{1} x_i y_i$, , tj. algorytm
$S := 0$
for $i := n$ **downto** 1 **do**
  $S := S + x_i * y_i$
end for

(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)).

Użyj pojedynczej i podwójnej precyzji (typy `Float32` i `Float64` w języku `Julia`). Porównaj wyniki z prawidłową wartością (dokładność do 15 cyfr) $-1.00657107000000_{10} - 11$.

zad. 6 Policz w języku `Julia` w arytmetyce `Float64` wartości następujących funkcji

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

dla kolejnych wartości argumentu $x = 8^{-1}, 8^{-2}, 8^{-3}, \ldots$. Chociaż $f = g$ komputer daje różne wyniki. Które z nich są wiarygodne, a które nie?

zad. 7 Przybliżoną wartość pochodnej $f(x)$ w punkcie x można obliczyć za pomocą następującego wzoru

$$f'(x_0) \approx \widetilde{f}'(x_0) = \frac{f(x_0 + h) - f(x_0)}{h}.$$

Skorzystać z tego wzoru do obliczenia w języku `Julia` w arytmetyce `Float64` przybliżonej wartości pochodnej funkcji $f(x) = \sin x + \cos 3x$ w punkcie $x_0 = 1$ oraz błędów $|f'(x_0) - \widetilde{f}'(x_0)|$ dla $h = 2^{-n}$ $(n = 0, 1, 2, \ldots, 54)$.

Jak wytłumaczyć, że od pewnego momentu zmniejszanie wartości h nie poprawia przybliżenia wartości pochodnej? Jak zachowują się wartości $1 + h$? Obliczone przybliżenia pochodnej porównać z dokładną wartością pochodnej, tj. zwróć uwagę na błędy $|f'(x_0) - \widetilde{f}'(x_0)|$ dla $h = 2^{-n}$ $(n = 0, 1, 2, \ldots, 54)$.

Rozwiązania zadań przedstawić w sprawozdaniu, plik pdf + **wydruk**, które powinno zawierać:

1. krótki opis problemu,

2. rozwiązanie,
3. wyniki oraz ich interpretację,
4. wnioski.

Do sprawozdania należy dołączyć pliki z kodem (`*.jl`). Pliki powinny być skomentowane: imię i nazwisko autora (**anonimy nie będą sprawdzane**), opisane parametry formalne funkcji, komentarze zmiennych. Spakowane pliki wraz ze sprawozdaniem (`*.zip`) należy przesłać e-mailem prowadzącemu. Natomiast wydruk sprawozdania należy oddać prowadzącemu na laboratorium.
Uwaga: Ostateczną wersję programów proszę przetestować pod linuksem.

Literatura

[1] J. Bezanson, S. Karpinski, V. Shah, A. Edelman, et al., Julia Language Documentation

[2] W. Kahan, Lecture Notes on the Status of IEEE Standard 754 for Binary Floating-Point Arithmetic