**Reprezentacja liczb i sprawdzanie obliczeń dla różnych arytmetyk, lista nr 1**

Obliczenia naukowe

Filip Rurak, 244960

Laboratorium 7:30, poniedziałek

**Zadanie nr 1**

* Celem zadania jest policzenie iteracyjne epsilonów maszynowych, liczb eta i liczby MAX, dla arytmetyk Float16, Float32 i Float64
* Zostały stworzone trzy funkcje
  + feps – licząca epsilon. Dzieli liczbę macheps na 2 i dodaje ją do 1 aż nie będzie spełnione 1 + macheps > 1
  + feta – licząca liczbę eta. Dzieli liczbę eta na 2, dopóki będzie ona większa od 0
  + fmax – licząca liczbę MAX. Mnoży liczbę max razy 2, aż nie będzie spełnione !isinf(max\*2). Potem w pętli dzieli max na 2 i dodaje do max, dopóki !isinf(max + delta)

Następnie policzone wartości są odpowiednio porównywane z funkcjami eps(), nextfloat() i floatmax()

function feps(t::Type)

macheps = t(1)

while 1 + macheps/2 > 1

macheps = macheps/2

end

return macheps

end

function feta(t::Type)

eta = t(1)

while eta/2 > 0

eta = eta/2

end

return eta

end

function fmax(t::Type)

max = t(1)

while !isinf(max\*2)

max = max \*2

end

delta = max/2

while !isinf(max + delta)

max += delta

delta = delta/2

end

return max

end

* Po uruchomieniu programu zostają wyświetlone następujące wyniki

Float16, macheps: 0.000977 vs eps(func): 0.000977

Float32, macheps: 1.1920929e-7 vs eps(func): 1.1920929e-7

Float64, macheps: 2.220446049250313e-16 vs eps(func): 2.220446049250313e-16

Float16, eta: 6.0e-8 vs nextfloat: 6.0e-8

Float32, eta: 1.0e-45 vs nextfloat: 1.0e-45

Float64, eta: 5.0e-324 vs nextfloat: 5.0e-324

Float16, MAX: 6.55e4 vs floatmax: 6.55e4

Float32, MAX: 3.4028235e38 vs floatmax: 3.4028235e38

Float64, MAX: 1.7976931348623157e308 vs floatmax: 1.7976931348623157e308

**Zadanie nr 2**

* Celem zadania jest sprawdzenie poprawności wyrażenia 3(4/3−1)−1 pozwalającego wyznaczyć epsilon maszynowy
* Została stworzona funkcja f, która przyjmuje za parametr typ zmiennej, gdzie wyrażenie 4/3 będzie danego typu oraz tablica typów dla których ma być testowana funkcja

function f(t::Type)

3(t(4/3)-1)-1

end

types=[Float16, Float32, Float64]

* Po uruchomieniu programu wyświetlone zostały wartości poniżej

Float16, Kahan: -0.000977 vs eps: 0.000977

Float32, Kahan: -1.1920929e-7 vs eps: 1.1920929e-7

Float64, Kahan: -2.220446049250313e-16 vs eps: 2.220446049250313e-16

* Jak widać wyniki funkcji są poprawne co do liczb, jednak ujemnych wartości, dlatego należy edytować ją poprzez umieszczenie wyrażenia w wartości bezwzględnej abs()

**Zadanie nr 3**

* Celem zadania jest sprawdzenie czy liczby z podanego przedziału znajdują się w takiej samej odległości
* Do rozwiązania podanego zadania została stworzona funkcja f, która porównuje cechy liczb x i y, po przejściu odpowiednio na następną i ostatnią, a następnie porównuje podaną deltę z różnicą między liczbami

function f(x, y, delta)

y1 = prevfloat(y)

x1 = nextfloat(x)

xexp = SubString(bitstring(x1), 2:12)

yexp = SubString(bitstring(y1), 2:12)

if xexp != yexp

return false

end

if y1 + delta != y

return false

end

return true

end

* Po uruchomieniu programu dla zmiennych x = 1 i y = 2 konsola wyświetla true, co oznacza, że liczby w tym przedziale są w takiej samej odległości
* Program został przetestowany dla liczb z przedziału (0; 1), z oczekiwanym wynikiem false

**Zadanie nr 4**

* Celem zadania jest znalezienie eksperymentalnie w arytmetyce Float64 liczby z przedziału (1; 2), która nie spełnia następującej równości x ∗ (1/x) = 1, a następnie znalezienia najmniejszej takiej liczby
* Do rozwiązania zadania została stworzona funkcja f, która iteruje przez przedział (1; 2) za pomocą funkcji nextfloat(), do momentu znalezienia liczby spełniającej t(x\*t(1/x)) != t(1). Funkcja zaczyna rosnąco od liczby 1, dlatego pierwsza znaleziona liczba będzie jednocześnie najmniejsza

function f(t::Type)

x=t(1)

while x != t(2) && x < t(2)

x = nextfloat(x)

if t(x\*t(1/x)) != t(1)

return x

end

end

return t(2)

end

* Po w miarę długim działaniu programu zostaje wyświetlony wynik x = 1.000000057228997
* Jest to liczba w stosunkowo małej odległości od 1, można stwierdzić więc, że istnieje wiele liczb niespełniających x ∗ (1/x) = 1

**Zadanie nr 5**

* Celem zadania jest obliczenie iloczynu skalarnego dwóch wektorów używając różnych funkcji i precyzji
* Zostały stworzone cztery funkcje
  + f1 – licząca sumę „w przód”
  + f2 – licząca sumę „w tył”
  + f3 – suma od największego do najmniejszego
  + f4 – suma od najmniejszego do największego

function f1(x::Vector, y::Vector)

s=0

for i=1:length(x)

s+= x[i] \* y[i]

end

return s

end

function f2(x::Vector, y::Vector)

s=0

for i=length(x):-1:1

s+= x[i] \* y[i]

end

return s

end

function f3(x::Vector, y::Vector)

xy=[]

for i=length(x):-1:1

push!(xy, x[i] \* y[i])

end

positive = sum(sort(filter(x -> x>=0, xy), rev=true))

negative = sum(sort(filter(x -> x<0, xy)))

return positive + negative

end

function f4(x::Vector, y::Vector)

xy=[]

for i=length(x):-1:1

push!(xy, x[i] \* y[i])

end

positive = sum(sort(filter(x -> x>=0, xy)))

negative = sum(sort(filter(x -> x<0, xy), rev=true))

return positive + negative

end

* Po uruchomieniu programu zostały wyświetlone następujące wyniki

Float32 f1: -0.4999443

Float32 f2: -0.4543457

Float32 f3: -0.5

Float32 f4: -0.5

Float64 f1: 1.0251881368296672e-10

Float64 f2: -1.5643308870494366e-10

Float64 f3: 0.0

Float64 f4: 0.0

* Można z tego wnioskować, że kolejność dodawania liczb w różnych arytmetykach wpływa na wynik końcowy

**Zadanie nr 6**

* Celem zadania jest stworzenie dwóch funkcji f(x) i g(x) pokazujących takie samo wyrażenie, jednak w funkcji f usunięto pierwiastek z mianownika oraz przetestowanie ich dla wartości x = 8−1 , 8−2 , 8−3 , …
* Poniższy listing przedstawia zaimplementowane funkcje

function f(x::Float64)

return sqrt(x^2 + 1)-1

end

function g(x::Float64)

return (x^2) / (sqrt(x^2 + 1)+1)

end

* Po uruchomieniu programu wyświetlane zostają następujące wyniki

f(8^-1)= 0.0077822185373186414 g(8^-1)= 0.0077822185373187065

f(8^-2)= 0.00012206286282867573 g(8^-2)= 0.00012206286282875901

f(8^-3)= 1.9073468138230965e-6 g(8^-3)= 1.907346813826566e-6

f(8^-4)= 2.9802321943606103e-8 g(8^-4)= 2.9802321943606116e-8

f(8^-5)= 4.656612873077393e-10 g(8^-5)= 4.6566128719931904e-10

f(8^-6)= 7.275957614183426e-12 g(8^-6)= 7.275957614156956e-12

f(8^-7)= 1.1368683772161603e-13 g(8^-7)= 1.1368683772160957e-13

f(8^-8)= 1.7763568394002505e-15 g(8^-8)= 1.7763568394002489e-15

f(8^-9)= 0.0 g(8^-9)= 2.7755575615628914e-17

f(8^-10)= 0.0 g(8^-10)= 4.336808689942018e-19

f(8^-11)= 0.0 g(8^-11)= 6.776263578034403e-21

…

* Funkcje dają zbliżone do siebie wyniki, jednak funkcja f dla wartości mniejszych od 8-8 nie daje wartości równe 0

**Zadanie nr 7**

* Celem zadania jest obliczenie przybliżonej wartości pochodnej funkcji w punkcie, porównanie jej prawidłową wartością i obliczenie błędu wyników
* Zostały stworzone trzy funkcje:
  + f – funkcja podana w zadaniu
  + fp – pochodna funkcji f
  + fp\_rd – pochodna funkcji f, policzona według wzoru (f(x+h)-f(x))/h, dla h = 2−n (n = 0, 1, 2, ..., 54)

function f(x::Float64)

return sin(x) + cos(3x)

end

function fp(x::Float64)

return cos(x) - 3sin(3x)

end

function fp\_rd(f::Function, x::Float64, h::Float64)

return (f(x+h)-f(x))/h

end

* Po uruchomieniu programu wyświetlone zostają następujące wyniki

Dla h= 2^-0 wartosc wynosi 2.0179892252685967 blad 1.9010469435800585

Dla h= 2^-1 wartosc wynosi 1.8704413979316472 blad 1.753499116243109

Dla h= 2^-2 wartosc wynosi 1.1077870952342974 blad 0.9908448135457593

Dla h= 2^-3 wartosc wynosi 0.6232412792975817 blad 0.5062989976090435

Dla h= 2^-4 wartosc wynosi 0.3704000662035192 blad 0.253457784514981

Dla h= 2^-5 wartosc wynosi 0.24344307439754687 blad 0.1265007927090087

…

Dla h= 2^-41 wartosc wynosi 0.116943359375 blad 1.0776864618478044e-6

Dla h= 2^-43 wartosc wynosi 0.1162109375 blad 0.0007313441885381522

Dla h= 2^-44 wartosc wynosi 0.1171875 blad 0.0002452183114618478

Dla h= 2^-45 wartosc wynosi 0.11328125 blad 0.003661031688538152

Dla h= 2^-46 wartosc wynosi 0.109375 blad 0.007567281688538152

Dla h= 2^-47 wartosc wynosi 0.109375 blad 0.007567281688538152

Dla h= 2^-48 wartosc wynosi 0.09375 blad 0.023192281688538152

Dla h= 2^-49 wartosc wynosi 0.125 blad 0.008057718311461848

Dla h= 2^-50 wartosc wynosi 0.0 blad 0.11694228168853815

Dla h= 2^-51 wartosc wynosi 0.0 blad 0.11694228168853815

Dla h= 2^-52 wartosc wynosi -0.5 blad 0.6169422816885382

Dla h= 2^-53 wartosc wynosi 0.0 blad 0.11694228168853815

Dla h= 2^-54 wartosc wynosi 0.0 blad 0.11694228168853815

Wartosc pochodnej = 0.11694228168853815.

* Dla początkowych wyników wartości znacznie się od siebie różnią, jednak aż do h = 2-40 błąd wyniku jest rzędu 10-7. Dla kolejnych wartości h, wynik pochodnej zbliża się do 0