Politechnika Wrocławska

Wydział Podstawowych Problemów Techniki

Obliczenia naukowe

Sprawozdanie – Lista nr 1

Autor: Tomasz Kulik

Wrocław 2015r.

**Zadanie 1.**

1. **Opis problemu**

Wyznaczenie liczb macheps i eta w sposób iteracyjny oraz porównanie wyników z wbudowanymi w języku Julia za pomocą wbudowanych metod. Przeprowadzenie eksperymentu na arytmetykach Float16, Float32 i Float64.

1. **Rozwiązanie**

Algorytm iteracyjnego wyznaczania liczby macheps polega na dzieleniu liczby x początkowo równej 1.0 przez 2 do czasu, gdy spełniony zostanie warunek „1.0+x=1.0”. Kiedy nastąpi taka sytuacja należy zwrócić ostatnią wartość przed wystąpieniem wartości „0.0” w zmiennej x. W przypadku wyznaczania liczby eta algorytm jest analogiczny z tą różnicą, że warunkiem końca pętli jest „0.0+x=0.0”.

1. **Wynik**

|  |
| --- |
| Wyznaczenie maszynowego epsilonu.  Typ Float16  Wynik funkcji iteracyjnej: 0.00097656  Wynik funkcji eps(Float16): 0.00097656  Typ Float32  Wynik funkcji iteracyjnej: 1.1920929e-7  Wynik funkcji eps(Float32): 1.1920929e-7  Typ Float64  Wynik funkcji iteracyjnej: 2.220446049250313e-16  Wynik funkcji eps(Float64): 2.220446049250313e-16  Wyznaczenie wartości eta.  Typ Float16  Wynik funkcji iteracyjnej: 5.9605e-8  Wynik funkcji nextfloat(float16(0.0)): 5.9605e-8  Typ Float32  Wynik funkcji iteracyjnej: 1.0e-45  Wynik funkcji nextfloat(float32(0.0)): 1.0e-45  Typ Float64  Wynik funkcji iteracyjnej: 5.0e-324  Wynik funkcji nextfloat(float64(0.0)): 5.0e-324 |

1. **Wnioski**Epsilon maszynowy oraz liczbę eta można policzyć w sposób iteracyjny. Epsilon maszynowy określa precyzję obliczeń danej arytmetyki.

**Zadanie 2.**

1. **Opis problemu**

Sprawdzenie w języku Julia twierdzenia Kahan’a: „macheps = 3(4/3−1)−1” w arytmetykach Float16, Float32 i Float64.

1. **Rozwiązanie**

Obliczenie wyrażenia 3(4/3-1)-1 rzutując każdą z liczb ze względu na przyjętą arytmetykę, czyli Float16, Float32 lub Float64.

1. **Wynik**

|  |
| --- |
| Wyznaczenie maszynowego epsilonu.  Typ Float16  Wynik funkcji Kahana: 0.00097656  Wynik funkcji eps(Float16): 0.00097656  Typ Float32  Wynik funkcji Kahana: 1.1920929e-7  Wynik funkcji eps(Float32): 1.1920929e-7  Typ Float64  Wynik funkcji Kahana: 2.220446049250313e-16  Wynik funkcji eps(Float64): 2.220446049250313e-16 |

1. **Wnioski**

Jest możliwe wyznaczenie maszynowego epsilonu przy pomocy jednego wyrażenia.

**Zadanie 3.**

1. **Opis problemu**

Sprawdzenie rozmieszczenia liczb w przedziałach [1,2], [0.5,1], [2,4].

1. **Rozwiązanie**

Rozwiązanie można przedstawić w postaci wydruku procedury bits dla liczb 1, 0.5, 2 oraz tych liczb powiększonych o odpowiednią delte.

1. **Wynik**

|  |
| --- |
| Dla przedziału[1,2]:  delta = 2^-52  0011111111110000000000000000000000000000000000000000000000000000  0011111111110000000000000000000000000000000000000000000000000001  0011111111110000000000000000000000000000000000000000000000000010  0011111111110000000000000000000000000000000000000000000000000011  0011111111110000000000000000000000000000000000000000000000000100  Dla przedziału[0.5,1]:  delta = 2^-52 / 2  0011111111100000000000000000000000000000000000000000000000000000  0011111111100000000000000000000000000000000000000000000000000001  0011111111100000000000000000000000000000000000000000000000000010  0011111111100000000000000000000000000000000000000000000000000011  0011111111100000000000000000000000000000000000000000000000000100  Dla przedziału[2,4]:  delta = 2^-52 \* 2  0100000000000000000000000000000000000000000000000000000000000000  0100000000000000000000000000000000000000000000000000000000000001  0100000000000000000000000000000000000000000000000000000000000010  0100000000000000000000000000000000000000000000000000000000000011  0100000000000000000000000000000000000000000000000000000000000100 |

1. **Wnioski**

Jeśli przyjrzeć się wynikom można stwierdzić, że ilość różnych liczb w tych trzech przedziałach jest równa i wynosi 2^52. Zmianie ulega jedynie delta – dla pierwszego przedziału wynosi 2^-52, drugiego 2^-52 / 2, a dla trzeciego 2^-52 \* 2.

**Zadanie 4.**

1. **Opis problemu**

Znalezienie najmniejszej liczby x z przedziału [1; 2] nie spełniającej równania x\*(1/x) = 1 w arytmetyce Float64 (double).

1. **Rozwiązanie**

Warunkiem wstępnym jest wyznaczenie maszynowego epsilonu „macheps” oraz zainicjalizowanie zmiennej x z wartością 1. Następnie w pętli należy zwiększać wartość zmiennej x o wartość macheps do czasu, gdy warunek x\*(1/x) = 1 nie będzie spełniony.

1. **Wynik**

|  |
| --- |
| Najmniejsza liczba, która nie spełnia równania x\*(1/x)=1 w arytmetyce zmiennoprzecinkowej Float64 to: 1.000000057228997 |

1. **Wnioski**

Prowadząc obliczenia w arytmetykach zmiennoprzecinkowych należy uważać na tego typu trudne do wykrycia przypadki niezgodności pomiędzy wynikami wyliczonymi przez maszynę, a tymi faktycznymi.

**Zadanie 5.**

1. **Opis problemu**

Obliczenie na cztery 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]

1. **Rozwiązanie**
   1. Obliczenie iloczynu skalarnego w pętli i = 1 do 5:  
      S = S + x[i] \* y[i]
   2. Obliczenie iloczynu skalarnego w pętli i = 5 do 1:  
      S = S + x[i] \* y[i]
   3. Od największego do najmniejszego (dodanie dodatnich liczb w porządku od największej do najmniejszej, dodaj ujemne liczby w porządku od najmniejszej do największej, dodanie do siebie obliczone sumy częściowe),
      1. Posortowanie iloczynów kolejnych wartości wektorów w pętli i=1 do 5:  
         z[i] = x[i] \* y[i]  
         z = sort(z)
      2. Zsumowanie wartości ujemnych rosnąco w pętli i=1 do ostatniej wartości ujemnej:  
         S\_plus = S\_plus + z[i]
      3. Zsumowanie wartości dodatnich malejąco w pętli i=5 w dół do ostatniej wartości dodatniej:  
         S\_minus = S\_minus + z[i]
      4. Dodanie do siebie sum częściowych:

Result = S\_minus + S\_plus

* 1. Od najmniejszego do największego (dodanie dodatnich liczb w porządku od najmniejszej do największej, ujemnych liczb w porządku od największej do najmniejszej, dodanie do siebie obliczonych sum częściowych),
     1. Posortowanie iloczynów kolejnych wartości wektorów w pętli i=1 do 5:  
        z[i] = x[i] \* y[i]  
        z = sort(z)
     2. Zsumowanie wartości ujemnych rosnąco w pętli   
        i=od największej wartości ujemnej do 1:  
        S\_plus = S\_plus + z[i]
     3. Zsumowanie wartości dodatnich malejąco w pętli i=od najmniejszej wartości dodatniej do 5:  
        S\_minus = S\_minus + z[i]
     4. Dodanie do siebie sum częściowych:  
        Result = S\_minus + S\_plus

1. **Wynik**

|  |
| --- |
| Dokładność Float64  Pierwsza wersja iloczynu: 1.0251881368296672e-10 Różnica z wartością prawdziwą: 1.1258452438296672e-10  Druga wersja iloczynu: -1.5643308870494366e-10 Różnica z wartością prawdziwą: 1.4636737800494365e-10  Trzecia wersja iloczynu: -5.519005839236119e6 Różnica z wartością prawdziwą: 5.519005839236119e6  Czwarta wersja iloczynu: -5.519005839236119e6 Różnica z wartością prawdziwą: 5.519005839236119e6  Dokładność Float32  Pierwsza wersja iloczynu: -0.4999443 Różnica z wartością prawdziwą: 0.4999443  Druga wersja iloczynu: -0.4543457 Różnica z wartością prawdziwą: 0.4543457  Trzecia wersja iloczynu: -5.519006e6 Różnica z wartością prawdziwą: 5.519006e6  Czwarta wersja iloczynu: -5.519006e6 Różnica z wartością prawdziwą: 5.519006e6 |

1. **Wnioski**Wynikiem najbliższym prawdy okazał się sposób pierwszy dla Float64 i drugi dla Float32

**Zadanie 6.**

1. **Opis problemu**

Obliczenie wartości następujących funkcji z argumentami

1. **Rozwiązanie**

Implementacja w języku Julia funkcji *f* zwracającej *sqrt(x^2 + float64(1.0)) - float64(1.0)* oraz funkcji *g* zwracającej *x^2 / (sqrt(x^2 + float64(1.0)) + float64(1.0)).*

1. **Wynik**

|  |
| --- |
| Dla x = 0.125  f(0.125) = 0.0077822185373186414  g(0.125) = 0.0077822185373187065  Dla x = 0.015625  f(0.015625) = 0.00012206286282867573  g(0.015625) = 0.00012206286282875901  Dla x = 0.001953125  f(0.001953125) = 1.9073468138230965e-6  g(0.001953125) = 1.907346813826566e-6  Dla x = 0.000244140625  f(0.000244140625) = 2.9802321943606103e-8  g(0.000244140625) = 2.9802321943606116e-8  Dla x = 3.0517578125e-5  f(3.0517578125e-5) = 4.656612873077393e-10  g(3.0517578125e-5) = 4.6566128719931904e-10  Dla x = 3.814697265625e-6  f(3.814697265625e-6) = 7.275957614183426e-12  g(3.814697265625e-6) = 7.275957614156956e-12  Dla x = 4.76837158203125e-7  f(4.76837158203125e-7) = 1.1368683772161603e-13  g(4.76837158203125e-7) = 1.1368683772160957e-13 |

1. **Wnioski**

Mimo teoretycznie identycznych spodziewanych wyników, funkcje *g* i *h* zwracają minimalnie różne wartości.

**Zadanie 7.**

1. **Opis problemu**

Obliczenie pochodnej funkcji f(x):

Według wzoru:

Dla kolejnych wartości w punkcie x0 = 1.

1. **Rozwiązanie**

Stworzenie funkcji f(x) zwracającej wynik sin(x) + cos(3\*x) oraz funkcji derivative(h) zwracającej (f(1 + h) - f(1)) / h. Następnie obliczenie w pętli i=od 1 do 54 wartości result = derivative(1/2^i) oraz wyliczenie różnicy pomiędzy nią a rzeczywistą wartością pochodnej f(x).

1. **Wynik**

Fragmenty wydruku z programu:

|  |
| --- |
| Wartość h obliczana wg wzoru: h = 1/2^n  Wartość 1 + h:1.0000000000009095  Wynik pochodnej przy wartości n=41:0.116943359375, różnica z faktyczną wartością:1.3593749999957216e-6.  Wartość 1 + h:1.0000000000004547  Wynik pochodnej przy wartości n=42:0.11669921875, różnica z faktyczną wartością:0.00024278125000000428.  Wartość 1 + h:1.0000000000002274  Wynik pochodnej przy wartości n=43:0.1162109375, różnica z faktyczną wartością:0.0007310625000000043.  Wartość 1 + h:1.0000000000001137  Wynik pochodnej przy wartości n=44:0.1171875, różnica z faktyczną wartością:0.0002454999999999957.  Wartość 1 + h:1.0000000000000568  Wynik pochodnej przy wartości n=45:0.11328125, różnica z faktyczną wartością:0.0036607500000000043.  Wartość 1 + h:1.0000000000000284  Wynik pochodnej przy wartości n=46:0.109375, różnica z faktyczną wartością:0.007567000000000004.  Wartość 1 + h:1.0000000000000142  Wynik pochodnej przy wartości n=47:0.109375, różnica z faktyczną wartością:0.007567000000000004.  Wartość 1 + h:1.000000000000007  Wynik pochodnej przy wartości n=48:0.09375, różnica z faktyczną wartością:0.023192000000000004.  Wartość 1 + h:1.0000000000000036  Wynik pochodnej przy wartości n=49:0.125, różnica z faktyczną wartością:0.008057999999999996.  Wartość 1 + h:1.0000000000000018  Wynik pochodnej przy wartości n=50:0.0, różnica z faktyczną wartością:0.116942.  Wartość 1 + h:1.0000000000000009  Wynik pochodnej przy wartości n=51:0.0, różnica z faktyczną wartością:0.116942.  Wartość 1 + h:1.0000000000000004  Wynik pochodnej przy wartości n=52:-0.5, różnica z faktyczną wartością:0.616942.  Wartość 1 + h:1.0000000000000002  Wynik pochodnej przy wartości n=53:0.0, różnica z faktyczną wartością:0.116942.  Wartość 1 + h:1.0  Wynik pochodnej przy wartości n=54:0.0, różnica z faktyczną wartością:0.116942.  Wartość 1 + h:1.0 |

1. **Wnioski**

Im mniejsza wartość *h* tym dokładniejszy jest wynik funkcji obliczającej wartość pochodnej. Jednak od pewnego momentu następuje załamanie i mimo bardzo niskiej wartości *h* wynik szybko zaczyna odbiegać od wartości oczekiwanej.