Laboratorium 2

Metody obliczeniowe w nauce i technice

Mateusz Ryś

18 Marca 2025

1.Treści zadań

1. Napisać algorytm do obliczenia funkcji wykładniczej e^x przy pomocy nieskończonych szeregów

    $e^x = 1 + x + \frac{x^2}{2!} + \frac{x^3}{3!} + \dots$

(1a) Wykonując sumowanie w naturalnej kolejności, jakie kryterium zakończenia obliczeń przyjjmiesz ?

(1b) Proszę przetestować algorytm dla:

x= +−**1**, +−**5**, +−**10**

i porównać wyniki z wynikami wykonania standardowej funkcji exp(x)

(1c) Czy można posłużyć się szeregami w tej postaci do uzyskania dokładnych wyników dla x < 0 ?

(1d) Czy możesz zmienić wygląd szeregu lub w jakiś sposób przegrupować składowe żeby uzyskać dokładniejsze wyniki dla x < 0 ?

2. Które z dwóch matematycznie ekwiwalentnych wyrażeń $x ** 2 - y ** 2$ oraz $(x - y) * (x + y)$ może być obliczone dokładniej w arytmetyce zmienno-przecinkowej ? Dlaczego ?
3. Dla jakich wartości x i y, względem siebie, istnieje wyraźna różnica w dokładności dwóch wyrażeń ?

Zakładamy że rozwiązujemy równanie kwadratowe $ax^2 + bx + c = 0$, z a = 1.22, b = 3.34 i c = 2.28, wykorzystując znormalizowany system zmienno-przecinkowy z podstawa beta = 10 i dokładnością p = 3.

(a) ile wyniesie obliczona wartość $b^2 - 4ac$?

(b) jaka jest dokładna wartość wyróżnika w rzeczywistej (dokładnej) arytmetyce ?

(c) jaki jest względny błąd w obliczonej wartości wyróżnika ?

2. Rozwiązania

2.1 Zadanie 1

Zacznijmy od przedstawienia samego algorytmu na e^x:

```
def machine_epsilon():
    epsilon = 1
    epsilon_last = epsilon

    while epsilon + 1 > 1:
        epsilon_last = epsilon
        epsilon = epsilon/2
    return epsilon_last

def exponent(x):
    val = 1
    to_add = 1
    index = 1
    epsilon = machine_epsilon()

    while abs(to_add)>epsilon:
        to_add *= x/index
        val += to_add
        index += 1

    return val
```

a. Ogólnie powinniśmy przyjąć koniec sumowania w momencie gdy wartość bezwzględna wartości dodawanej będzie mniejsza od pewnej wartości. Problemem jest wybór owej wartości. Tutaj idealnym rozwiązaniem tej zagadki jest maszynowy epsilon. Dlaczego? Z poprzednich laboratorium wiemy, że jest to wartość której dodanie do jakiejś innej liczby nie zmienia nam jej. Więc ta wartość tutaj będzie idealna. Wykorzystuję oczywiście kod z poprzednich zajęć do obliczenia naszego epsilona.

b. Poniżej mamy kod który porównuje wyniki mojej implementacji z wynikami funkcji bibliotecznej exp

```
if __name__ == "__main__":
    arr = [1,-1,5,-5,10,-10]

    for element in arr:
        print("Dla x = ",element)
        print("Wartosc zwrocona z exponent: ",exponent(element))
        print("Wartosc zwrocna z exp: ",exp(element))
        print("----------------------------------------------------")
```

Oraz output

```
Dla x =  1
Wartosc zwrocona z exponent:  2.7182818284590455
Wartosc zwrocna z exp:  2.718281828459045
-----------------------------------------------------
Dla x =  -1
Wartosc zwrocona z exponent:  0.36787944117144245
Wartosc zwrocna z exp:  0.36787944117144233
-----------------------------------------------------
Dla x =  5
Wartosc zwrocona z exponent:  148.4131591025766
Wartosc zwrocna z exp:  148.4131591025766
-----------------------------------------------------
Dla x =  -5
Wartosc zwrocona z exponent:  0.006737946999084642
Wartosc zwrocna z exp:  0.006737946999085467
-----------------------------------------------------
Dla x =  10
Wartosc zwrocona z exponent:  22026.465794806714
Wartosc zwrocna z exp:  22026.465794806718
-----------------------------------------------------
Dla x =  -10
Wartosc zwrocona z exponent:  4.5399929670419935e-05
Wartosc zwrocna z exp:  4.5399929762484854e-05
-----------------------------------------------------
```

c. Nie można się posłużyć tym szeregiem ponieważ na przemian dodajemy i odejmujemy wartości. Może to doprowadzić do *catastrophic cancellation*. Zjawisko to polega na tym, że odjęcie dwóch bliskich siebie liczb może nam dać złe przybliżenie ich różnicy.

d. Można lekko zmienić ten szereg i dla ujemnych liczb, zamiast liczyć e^{-x}, byśmy policzyli $\frac{1}{e^x}$. Sprowadza się do tego, że obliczamy $e^{|x|}$ i dla $x < 0$ wynikiem będzie liczba odwrotna do otrzymanego wyniku.

```
def better_exponent(x):
    val = 1
    to_add = 1
    index = 1
    new_x = abs(x)
    epsilon = machine_epsilon()

    while abs(to_add)>epsilon:
        to_add *= new_x/index
        val += to_add
        index += 1

    if x >= 0:
        return val
    else:
        return 1/val
```

Wyniki dla wartości $x < 0$ prezentują się teraz lepiej

```
Dla x =  1
Wartosc zwrocona z exponent:  2.7182818284590455
Wartosc zwrocna z exp:  2.718281828459045
-----------------------------------------------------
Dla x =  -1
Wartosc zwrocona z exponent:  0.3678794411714423
Wartosc zwrocna z exp:  0.36787944117144233
-----------------------------------------------------
Dla x =  5
Wartosc zwrocona z exponent:  148.4131591025766
Wartosc zwrocna z exp:  148.4131591025766
-----------------------------------------------------
Dla x =  -5
Wartosc zwrocona z exponent:  0.006737946999085467
Wartosc zwrocna z exp:  0.006737946999085467
-----------------------------------------------------
Dla x =  10
Wartosc zwrocona z exponent:  22026.465794806714
Wartosc zwrocna z exp:  22026.465794806718
-----------------------------------------------------
Dla x =  -10
Wartosc zwrocona z exponent:  4.5399929762484854e-05
Wartosc zwrocna z exp:  4.5399929762484854e-05
-----------------------------------------------------
```

Wnioski:

Operacja odejmowania może prowadzić do niedokładności wyników. Jak widzieliśmy powyżej dla $x < 0$ dostawaliśmy wyniki lekko różniące się od siebie, a jak usunęliśmy opoeracje różnicy to uzyskaliśmy bliższe wyniki do oczekiwanych.

2.2 Zadanie 2

a. Mnożenie jest zazwyczaj obarczone większym błędem niż odejmowanie. Różnica może prowadzić do *catastrophic cancellation* który jeszcze bardziej zwiększa błąd. Również może pojawić się problem z kwadratami. Mianowicie mogą być tak duże, że nie będzie możliwe zapisanie ich w naszym systemie. W związku z tym $x^2 - y^2$ może prowadzić do bardziej odległych wyników względem prawdziwego niż $(x - y)(x + y)$.

b. Takowa różnica występuje między innymi dla wartości x i y które są blisko siebie, ale nie identyczne. Innymi słowy są to tak dobrane liczby, których kwadraty są podatne na *catastrophic cancellation*. Wartości oczywiście nie mogą przekraczać zakresu reprezentacji systemu.

2.3 Zadanie 3

Rozpatrujemy równanie kwadratowe:

$$1.22x^2 + 3.34x + 2.28$$

a) Po kolei będziemy teraz rozpatrywać kolejne działania i dostosowywać wyniki do przyjętego systemu (beta = 10, p = 3):

$b^2 = 3.34 * 3.34 = 11.1556 \approx 11.2$

$4a = 4 * 1.22 = 4.88$

$4ac = 4.88 * 2.28 = 11.1264 \approx 11.1$

$\Delta = b^2 - 4ac = 11.2 - 11.1 = 0.1$

b) Teraz liczymy dokładną wartość, więc dostajemy:

$\Delta = b^2 - 4ac = (3.34)^2 - 4 * 1.22 * 2.28 = 0.0292$

c) $\text{błąd względny} = \frac{\text{błąd bezwzględny}}{\text{wartość rzeczywista}} = \frac{|\text{wartość przybliżona} - \text{wartość rzeczywista}|}{\text{wartość rzeczywista}} = \frac{|0.1 - 0.0292|}{0.0292} \approx 2.4246575$

Wnioski:

Przez ograniczoną precyzję systemu wykonane obliczenia obciążone są błędem, który wynika z zaokrągleń, które były potrzebne do zapisania danej liczby.

3. Bibliografia

1. Katarzyna Rycerz, wykład z przedmiotu Metody Obliczeniowe w Nauce i Technice
2. Michael T. Heath, Scientific Computing: An Introductory Survey. Chapter 1 - Scientific Computing, wydawnictwo McGraw-Hill, 2001
3. https://en.wikipedia.org/wiki/Catastrophic_cancellation
4. Obliczenia wykonane w Wolfram|Alpha: https://www.wolframalpha.com/