Laboratorium 2

Arytmetyka komputerowa (cd.)

Krzysztof Solecki

7 Marca 2023

1 Treści zadań

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

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

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

(b) Proszę przetestować algorytm dla: x= +-1, +-5, +-10 i porównać wyniki z wynikami wykonania standardowej funkcji exp(x)

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

(d) 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 $(x - y) \times (x + y)$ 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

1.

a) W poniższym algorytmie określam ilość wyrazów sumowanych jako parametr funkcji calculate_e. Im większą wartość n określę przy wywołaniu funkcji tym dokładniejszy wynik otrzymam dla wartości x > 0.

Listing 1: Python example

```
# 1a) Wykonujac sumowanie w naturalnej kolejnosci, jakie kryterium zakonczenia obliczen
    przyjmiesz?

def calculate_e(x, n):
    sum = 1.0
    for i in range(n,0,-1):
        sum = 1 + x*sum/i
    return sum


x = int(input("Enter an argument x to exp(x) calculate: "))
n = int(input("Enter a number of sum elements: "))
exp = calculate_e(x, n)
print("The exponential value of {} is {}".format(x, exp))
```

b) Poniższy kod programu prezentuje wyniki zwrócone przez funkcję dla x= +-1,+-5,+-10. Zostały one zestawione z wynikami funkcji bibliotecznej exp(x) dla tych samym argumentów x. Można zauważyć, że dla x<0 te wartości się różnią. Największa różnica jaką można zauważyć występuje dla x = -10. Wówczas błąd względny jest na poziomie $\approx 97\%$.

Listing 2: Python example

```
# 1b) Prosz przetestowa algorytm dla:
# x= +-1, +-5, +-10
# i  porwna  wyniki z wynikami wykonania standardowej funkcji exp(x)

from math import exp

n = 30

exp_1p = calculate_e(1,n)
exp_1m = calculate_e(-1,n)
exp_5p = calculate_e(5,n)
exp_5m = calculate_e(-5,n)
exp_10p = calculate_e(10,n)
exp_10m = calculate_e(-10,n)

print("Argument x: {}\nValue of my function: {}\nValue of python exp function:
    {}".format(1,exp_1p,exp(1)))
print("Argument x: {}\nValue of my function: {}\nValue of python exp function:
    {}".format(-1,exp_1m,exp(-1)))
print("Argument x: {}\nValue of my function: {}\nValue of python exp function:
    {}".format(5,exp_5p,exp(5)))
```

```
print("Argument x: {}\nValue of my function: {}\nValue of python exp function:
    {}".format(-5,exp_5m,exp(-5)))
print("Argument x: {}\nValue of my function: {}\nValue of python exp function:
    {}".format(10,exp_10p,exp(10)))
print("Argument x: {}\nValue of my function: {}\nValue of python exp function:
    {}".format(-10,exp_10m,exp(-10)))
```

```
Argument x: 1
Value of my function: 2.718281828459045
Value of python exp function: 2.718281828459045
Argument x: -1
Value of my function: 0.36787944117144233
Value of python exp function: 0.36787944117144233
Argument x: 5
Value of my function: 148.41315910257592
Value of python exp function: 148.4131591025766
Argument x: -5
Value of my function: 0.006737946999571198
Value of python exp function: 0.006737946999085467
Argument x: 10
Value of my function: 22026.46403625892
Value of python exp function: 22026.465794806718
Argument x: -10
Value of my function: 0.000970341580183387
Value of python exp function: 4.5399929762484854e-05
```

c)Podczas testowania powyższego programu dla argumentów x $<$ 0 można zauważyć znaczące różnice między wartością aproksymowaną z użyciem szeregu Maclaurina, a wartością zwróconą przez funkcję exp(x) z biblioteki math, szczególnie widać to dla x= -10. W poniższym przykładzie wartość sumy była obliczana dla 30 wyrazów szeregu Maclaurina. Dlatego powyższy algorytm w tej postaci nie jest poprawny dla obliczania wartości dla x$<$0.

d) Dla x$<$0 algorytm można zmodyfikować w celu uzyskania dokładniejszych wyników. Należy w tym celu skorzystać z prostego przekształcenia $e^{-x} = \frac{1}{e^x}$. Wówczas nasz algorytm stanie się algorytmem stabilnym.

Listing 3: Python example

```
def calculate_e(x, n):
    reverse_result = False
    if x<0:
        x = abs(x)
        reverse_result = True

    result = float(1.0)
    for i in range(n,0,-1):
        result = 1 + x * result / i

    if reverse_result:
        return 1 / result
    else: return result
```

```
Argument x: 1
```

```
Value of my function: 2.718281828459045
Value of python exp function: 2.718281828459045
Argument x: -1
Value of my function: 0.36787944117144233
Value of python exp function: 0.36787944117144233
Argument x: 5
Value of my function: 148.41315910257592
Value of python exp function: 148.4131591025766
Argument x: -5
Value of my function: 0.006737946999085498
Value of python exp function: 0.006737946999085467
Argument x: 10
Value of my function: 22026.46403625892
Value of python exp function: 22026.465794806718
Argument x: -10
Value of my function: 4.53999333871223e-05
Value of python exp function: 4.5399929762484854e-05
```

Wnioski: Algorytm aproksymacji numerycznej wartości funkji $e(x)$ za pomocą szeregu Maclaurina jest algorytmem niestabilnym. Niestabilność ta widoczna jest, gdy chcemy policzyć wartość funkcji e^x dla x ujemnego. Spowodowane jest to błędem *catastrophic cancellation* , czyli odejmowanie bliskich liczb, którego wynikiem jest mała liczba. Ta liczba jest normalizowana, mantysę przesuwamy w lewo, a pojawiające się miejsca po prawej stronie zapełniane są zerami lub przypadkowymi wartościami. Aby nasz algorytm stał się algorytmem stabilnym należy zastosować przekształcenie (dla ujemnych x): $e^{-x} = \frac{1}{e^x}$

2.

To które z ww. wyrażeń będzie dokładniej obliczone w arytmetyce zmiennoprzecinkowej zależy od argumentów x i y. Może się zdarzyć sytuacja, w której x^2 lub y^2 bedą na tyle duże, że nie zmieszczą się w danej arytmetyce, wówczas różnice w wynikach będą bardzo duże. Dodatkowo w przypadku, gdy x ≈ y, to operacja $x^2 - y^2$ spowoduje dość duże błędy względne. W przypadku, gdy |x| » |y| wyrażenie $x^2 - y^2$ może zostać obliczone dokładniej, ponieważ będą dwa błędy zaokrogleń (błąd $fl(y^2)$ nie będzie miał znaczenia dla wyniku). Natomiast wyrażenie $(x - y)(x + y)$ będzie miało wówczas trzy błędy zaokrąglenia.

Wnioski: Moc obliczeniowa potrzebna do obliczenia obu wyrażeń jest w przybliżeniu taka sama. Ostatecznie najczęściej lepszym wyborem będzie obliczenie wartości $(x - y)(x + y)$.

3.

a) Ile wyniesie obliczona wartość b^2 – 4ac ?

$$fl(b * b) = fl(3.34 \times 3.34) = fl(11.1556) = 11.2$$
$$fl(4 \times a \times c) = fl(fl(4 \times 1.22) \times 2.28) = fl(fl(4.88) \times 2.28) = fl(4.88 \times 2.28) = fl(11.1264) = 11.1$$
$$fl(11.2 - 11.1) = 0.1$$

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

$$b^2 - 4\text{ac} = 0.0292$$

c) Jaki jest względny błąd w obliczonej wartości wyróżnika ?

$$b_w = \frac{0.1-0.0292}{0.0292} = \frac{0.0708}{0.0292} \approx 2.42466$$

3 Bibliografia

1. Katarzyna Rycerz: Materiały wykładowe z przedmiotu Metody Obliczeniowe w Nauce i Technice