

# Laboratorium 2 - Arytmetyka Komputerowa(cd.)

Piotr Karamon

17.03.2024r.

## Treści zadań

1. Napisać algorytm do obliczenia funkcji wykładniczej  $e^x$  przy pomocy nieskończonych szeregów  $e^x = 1 + x + x^2/2! + x^3/3! + \dots$ 
  - (1a) Wykonując sumowanie w naturalnej kolejności, jakie kryterium zakończenia obliczeń przyjmiesz ?
  - (1b) Proszę przetestować algorytm dla:  $x = \pm 1, \pm 5, \pm 10$  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  $(x - y) \cdot (x + y)$  może być obliczone dokładniej w arytmetyce zmienneo-przecinkowej ? Dlaczego ?
3. Dla jakich wartości  $x$  i  $y$ , względem siebie, istnieje wyraźna różnica w dokładności dwóch wyrażeń ?
4. Zakładamy że rozwiązujemy równanie kwadratowe  $ax^2 + bx + c = 0$ , z  $a = 1.22$ ,  $b = 3.34$  i  $c = 2.28$ , wykorzystując znormalizowany system zmienneo-przecinkowy z podstawą  $\beta = 10$  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 ?

## Zadanie 1.

$$e^x = 1 + x + \frac{x^2}{2!} + \frac{x^3}{3!} + \frac{x^4}{4!} + \frac{x^5}{5!} + \dots$$

(1a) Wykonując sumowanie w naturalnej kolejności, jako kryterium zakończenia obliczeń przyjmujemy sytuację, gdy kolejny wyraz szeregu jest wystarczająco mały, tzn. mniejszy od ustalonej z góry wartości granicznej  $10^{-10}$ . Algorytm ten będzie napisany przy użyciu języka Rust. Z racji na jego specyfikę zamiast `exp(3.14)` używa się składni `(3.14f64).exp()`.

(1b)

```
fn my_exp(x: f64) -> f64 {
    let mut sum = 1.0;
    let mut term: f64 = 1.0; let mut n = 1.0;

    while term.abs() >= 1e-10 {
        term *= x / n;
        sum += term;
        n += 1.0;
    }
    sum
}

fn main() {
    let nums = [-1.0, 1.0, -5.0, 5.0, 10.0, -10.0];
    for x in nums {
        println!("x = {}", x);
        println!("my_exp({}) = {:e}", x, my_exp(x));
        println!("exp({}) = {:e}", x, x.exp());
        println!("różnica = {:e}", (my_exp(x) - x.exp()).abs());
        println!();
    }

    println!("my_exp(-50) = {:e}", my_exp(-50f64));
    println!("exp(-50) = {:e}", (-50f64).exp());
}
```

```
x = -1
my_exp(-1) = 3.6787944117216204e-1
exp(-1) = 3.6787944117144233e-1
różnica = 7.197020757132577e-13
```

```
x = 1
my_exp(1) = 2.71828182845823e0
```

```

exp(1) = 2.718281828459045e0
różnica = 8.149037000748649e-13

x = -5
my_exp(-5) = 6.7379469960630635e-3
exp(-5) = 6.737946999085467e-3
różnica = 3.022403508023963e-12

x = 5
my_exp(5) = 1.4841315910257242e2
exp(5) = 1.484131591025766e2
różnica = 4.177991286269389e-12

x = 10
my_exp(10) = 2.2026465794806707e4
exp(10) = 2.2026465794806718e4
różnica = 1.0913936421275139e-11

x = -10
my_exp(-10) = 4.539993653253598e-5
exp(-10) = 4.5399929762484854e-5
różnica = 6.770051126691115e-12

my_exp(-50) = 2.0418329628976137e3
exp(-50) = 1.9287498479639178e-22

```

Wyniki dla podanych liczb w zadaniu są bardzo zbliżone. Ogromną natomiast różnicę widać dla wartości  $-50$ . Dla bardzo dużych wartości ujemnych  $x$  sumowanie wielu wyrazów szeregu może prowadzić do wzrostu błędu numerycznego. Przyczyna to *catastrophic cancellation*. Gdy odejmujemy bliskie sobie liczby to wynikiem jest mała liczba, która posiada dużo zer tam gdzie jej "poprzednicy" mieli cyfry znaczące. Ta liczba jest normalizowana, zera zapisywane są w wykładniku a mantysa przesuwana jest "w lewo". Nie wiadomo czym zappełnić pojawiające się w mantysie po prawej stronie miejsca (zera lub losowe wartości).

(1c) Wadą korzystania z szeregów gdy chcemy policzyć  $e^{-x}$  dla bardzo dużych wartości negatywnych jest to, że napotykamy na *catastrophic cancellation*. Możemy spróbować zredukować ilość odejmowań przez które to tracimy precyzję i policzyć dwa szeregi o wartościach dodatnich. Dla  $x < 0$ ,

szereg przekształcamy w następujący sposób.

$$e^{-x} = 1 - x + \frac{x^2}{2!} - \frac{x^3}{3!} + \frac{x^4}{4!} - \frac{x^5}{5!} + \dots$$

$$e^{-x} = \sum_{i=0}^{\infty} \frac{x^{2i}}{(2i)!} - \sum_{i=0}^{\infty} \frac{x^{2i+1}}{(2i+1)!}$$

Obie sumy można policzyć osobno a potem odjąć jedną od drugiej. Takie rozwiązanie niestety nie przynosi efektów:

```
fn main() {
  for x in [-3.0, -15.0, -30.0, -50.0, -100.0] {
    println!("x = {x}");
    println!("my_exp_neg({}) = {:.e}", x, my_exp_neg(x));
    println!("exp({}) = {:.e}", x, (x).exp());
    println!();
  }
}

const EPS: f64 = 1e-10;
fn my_exp_neg(mut x: f64) -> f64 {
  x = -x;
  let mut pos_total = 0.0;
  let mut neg_total = 0.0;
  let mut n = 1.0;
  let mut term = 1.0;

  while term > EPS {
    pos_total += term;
    term *= x / n;
    n += 1.0;
    neg_total += term;

    term *= x / n;
    n += 1.0;
  }

  pos_total - neg_total
}
```

```
x = -3
my_exp_neg(-3) = 4.9787068343178476e-2
exp(-3) = 4.9787068367863944e-2
```

```
x = -15
```

```
my_exp_neg(-15) = 3.059394657611847e-7
exp(-15) = 3.059023205018258e-7
```

```
x = -30
my_exp_neg(-30) = -9.765625e-4
exp(-30) = 9.357622968840175e-14
```

```
x = -50
my_exp_neg(-50) = 5.24288e5
exp(-50) = 1.9287498479639178e-22
```

```
x = -100
my_exp_neg(-100) = -9.903520314283042e27
exp(-100) = 3.720075976020836e-44
```

To nie działa ponieważ na ogół `pos_total` i `neg_total` osiągają bardzo duże wartości, zatem by dokładnie obliczyć wartość  $e^{-x}$  musielibyśmy je znać z bardzo dużą liczbą cyfr znaczących. Dokładność oferowana przez typ `f64` (double w innych językach) to zdecydowanie za mało.

(1d) Dużo prostszym rozwiązaniem jest zastosowanie wzoru:

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

Zatem finalny algorytm przedstawia się następująco:

```
fn my_exp(x: f64) -> f64 {
  if x < 0.0 {
    return 1.0/my_exp(-x);
  }

  let mut sum = 1.0;
  let mut term: f64 = 1.0; let mut n = 1.0;

  while term.abs() >= 1e-10 {
    term *= x / n;
    sum += term;
    n += 1.0;
  }
  sum
}

fn main() {
  let nums = [30.0, -10.0, -30.0, -50.0, -100.0];
```

```

for x in nums {
    println!("x = {}",x);
    println!("my_exp({}) = {:e}",x, my_exp(x));
    println!("exp({}) = {:e}",x, x.exp());
    println!("różnica = {:e}",(my_exp(x) - x.exp()).abs());
    println!();
}
}

```

```

x = 30
my_exp(30) = 1.0686474581524465e13
exp(30) = 1.0686474581524463e13
różnica = 1.953125e-3

```

```

x = -10
my_exp(-10) = 4.5399929762484875e-5
exp(-10) = 4.5399929762484854e-5
różnica = 2.0328790734103208e-20

```

```

x = -30
my_exp(-30) = 9.357622968840172e-14
exp(-30) = 9.357622968840175e-14
różnica = 2.524354896707238e-29

```

```

x = -50
my_exp(-50) = 1.9287498479639145e-22
exp(-50) = 1.9287498479639178e-22
różnica = 3.291384182302405e-37

```

```

x = -100
my_exp(-100) = 3.7200759760208336e-44
exp(-100) = 3.720075976020836e-44
różnica = 2.4892061111444567e-59

```

Ta wersja algorytmu daje wyniki bardzo zbliżone do funkcji bibliotecznej.

**Wnioski:** W przypadku liczenia wartości szeregów często możemy natrafić na *catastrophic cancellation*, przez które nasze obliczenia stają się bardzo niedokładne a wyniki absurdalne. By poradzić sobie z tym problemem najlepiej jest szukać innych, ale równoważnych matematycznie sposób liczenia, jak np.  $e^{-x} = \frac{1}{e^x}$ .

## Zadanie 2

Matematycznie równoważne wyrażenia

1.  $x^2 - y^2$
2.  $(x - y) \cdot (x + y)$

By obliczyć pierwsze wyrażenie musimy wykonać dwa mnożenia. Mnożenie jest bardziej niedokładne na liczbach zmiennoprzecinkowych niż dodawanie. Ponad to w przypadku mnożenia *overflow* i *underflow* są bardziej prawdopodobne. W przypadku obliczania pierwszego wyrażenia możemy spotkać się z *catastrophic cancellation*, gdy  $x$  i  $y$  są bliskie siebie.

W drugim wyrażeniu wykonujemy tylko jedno mnożenie, ale za to mamy dwa dodawania. Jednakże na ogół błędy dodawania są mniejsze od błędów związanych z mnożeniem.

Z powodów przedstawionych powyżej na ogół wyrażenie  $(x - y) \cdot (x + y)$  będzie dokładniej obliczane w systemach zmiennoprzecinkowych.

## Zadanie 3

Aby pokazać różnicę w obliczaniu tych dwóch wyrażeń użyjemy prostego systemu zmiennoprzecinkowego w którym podstawa  $\beta = 10$  a mantysa ma długość równą 3. Niech  $x = 123 = 1.23 \cdot 10^2$  i  $y = 122 = 1.22 \cdot 10^2$ .

1.  $x^2 - y^2$ 
  - $x^2 = 15129$ , czyli w naszym systemie:  $x^2 = 1.51 \cdot 10^4$
  - $y^2 = 14884$ , czyli w naszym systemie:  $y^2 = 1.49 \cdot 10^4$
  - $x^2 - y^2 = 1.51 \cdot 10^4 - 1.49 \cdot 10^4 = 0.02 \cdot 10^4 = 2.00 \cdot 10^2 = 200$
2.  $(x - y) \cdot (x + y)$ 
  - $x - y = 123 - 122$ , czyli w naszym systemie:  $x - y = 1.0 \cdot 10^0$   
warto tutaj zwrócić uwagę na to, że wynik odejmowania należy do naszego systemu.
  - $x + y = 123 + 122 = 245$ , czyli w naszym systemie:  $x + y = 2.45 \cdot 10^2$   
wynik dodawania również mieści się w naszym systemie.
  - $(x - y) \cdot (x + y) = 1.0 \cdot 10^0 \cdot 2.45 \cdot 10^2 = 2.45 \cdot 10^2 = 245$

Wyniki uzyskany poprzez drugi sposób jest zgodny z prawdą.

**Wnioski:** W przypadku obliczeń numerycznych warto przeanalizować każdy składnik rozwiązania i zastanowić się, czy istnieje równoważny matematycznie sposób policzenia go, który jednak będzie *przyjaźniejszy* arytmetyce zmiennoprzecinkowej. Jak pokazuje przykład nawet tak proste matematycznie równoważne wyrażania jak  $x^2 - y^2$  i  $(x - y) \cdot (x + y)$  przy arytmetyce zmiennoprzecinkowej nie są już równoważne. Wybranie  $(x - y) \cdot (x + y)$  sprawi że nasze obliczenia będą dokładniejsze.

## Zadanie 4

$$1.22x^2 + 3.34x + 2.28 = 0$$

System  $\beta = 10$  i  $p = 3$ .

1. Ile wyniesie obliczona wartość  $b^2 - 4ac$ ?

$$fl(b^2) = fl(3.34 \cdot 3.34) = fl(11.1556) = 11.2$$

$$fl(4a) = fl(4 \cdot 1.22) = fl(4.88) = 4.88$$

$$fl(4ac) = fl(fl(4 \cdot a) \cdot c) = fl(4.88 \cdot 2.28) = fl(11.1264) = 11.1$$

$$fl(b^2 - 4ac) = fl(fl(b^2) - fl(4ac)) = fl(11.2 - 11.1) = 0.1$$

Wartość tego wyrażenia wyniesie  $\hat{\Delta} = 0.1$ .

2. Jaka jest dokładna wartość wyróżnika w jaka jest dokładna wartość wyróżnika w rzeczywistej (dokładnej) arytmetyce ?

$$\Delta = b^2 - 4ac = 11.1556 - 11.1264 = 0.0292$$

3. Jaki jest względny błąd w obliczonej wartości wyróżnika?

$$\left| \frac{\hat{\Delta} - \Delta}{\Delta} \right| = \left| \frac{0.1 - 0.0292}{0.0292} \right| \approx 2.4266 = 242.66\%$$

**Wnioski:** Błąd względny jest bardzo duży. Choć wartości współczynników należały do systemu zmiennoprzecinkowego to jednak w wyniku obliczeń nasz wynik bardzo odbiegł od tego w rzeczywistej arytmetyce.

## Bibliografia

- Marian Bubak, Katarzyna Rycerz: *Metody Obliczeniowe w Nauce i Technice. Wprowadzenie, arytmetyka komputerowa*