# Analiza Numeryczna

Skrypt z wykładów
Semestr zimowy 2023/2024

Uniwersytet Wrocławski
Wydział Matematyki i Informatyki
Instytut Informatyki

# Spis treści

## Wykład 1

## Wykład 2 Arytmetyka zmiennopozycyjna, podstawy teorii błędów

# 2.1 Błąd bezwzględny i blad względny

Weźmy liczby:

$$x = 1.23456789 \quad \tilde{x} = 1.2345679$$
$$y = 10^{50} + 1 \quad \tilde{y} = 10^{50}$$

Wtedy błąd bezwzględny to:

$$|x - \tilde{x}| = 10^{-8} \quad |y - \tilde{y}| = 1$$

Błąd względny to:

$$\frac{|x - \tilde{x}|}{|x|} = 0.8 \cdot 10^{-8} \quad \frac{|y - \tilde{y}|}{|y|} = 10^{-50}$$

Błąd względny jest lepszą miarą błędu.
Dodatkowo zdefiniujmy liczbę cyfr dokładnych:

$$acc(v, \tilde{v}) = -\log_{10}\left(\left|1 - \frac{\tilde{v}}{v}\right|\right)$$

Wtedy:

$$acc(x, \tilde{x}) \approx 8.091 \quad acc(y, \tilde{y}) = 50$$

# 2.2 Reprezentacja liczb w komputerze

Potrafimy reprezentować wszystkie liczby całkowite z pewnego zakresu, ale nie potrafimy reprezentować wszystkich liczb rzeczywistych. Dlatego musimy wybrać pewną reprezentację, która będzie przybliżać liczby rzeczywiste.

a) $l \in \mathbb{Z}$,

$$l = \pm \sum_{i=0}^{n} e_i 2^i, \quad e_i \in \{0, 1\}, \quad e_n = 1$$

Jeśli $n < d$ to OK, a jeśli $n \geq d$ to przepełnienie.

b) $x \in \mathbb{R} \setminus \{0\}$

**TW.** Dla każdej liczby rzeczywistej $x \neq 0$ istnieje trójka:

$$m \in \left[\frac{1}{2}, 1\right) \quad \text{(mantysa)}$$

$$c \in \mathbb{Z} \quad \text{(cecha)}$$

$$s \in \{+1, -1\} \quad \text{(znak liczby)}$$

dla których

$$x = s \cdot m \cdot 2^c$$

Trójka $(s, m, c)$ jest wyznaczona jednoznacznie.

Reprezentacja mantysy:

$$m \in \left[\frac{1}{2}, 1\right), \quad m = \sum_{i=1}^{\infty} e_{-i} 2^{-i}, \quad e_{-i} \in \{0, 1\}, \quad e_{-1} = 1$$

Sposoby zaokrąglania mantysy:
i) obcięcie

$$m_t^c := \sum_{i=1}^{t} e_{-i} 2^{-i}$$

ii) zaokrąglanie symetryczne

$$m_t^r := \sum_{i=1}^{t} e_{-i} 2^{-i} + e_{-(t+1)} 2^{-t}$$

Model zapisu liczby:

$$[\pm] \underbrace{[\text{bity mantysy}]}_{t \text{ bitów}} \quad \underbrace{[\text{cecha ze znakiem}]}_{(d-t) \text{ bitów}}$$

Łącznie: $d + 1$ bitów na liczbę rzeczywistą ze znakiem.
Ten model reprezentacji jest teoretyczny. W praktyce stosujemy standard IEEE 754.

**TW.**

$$|m - m_t^c| \leq 2^{-t}, \qquad |m - m_t^r| \leq \frac{1}{2} \cdot 2^{-t}$$

Reprezentacja zmiennopozycyjna liczby rzeczywistej $x \neq 0$:

$$x \approx \text{chop}(x) := s \cdot m_t^c \cdot 2^c \qquad \text{albo} \qquad x \approx \text{rd}(x) := s \cdot m_t^r \cdot 2^c$$

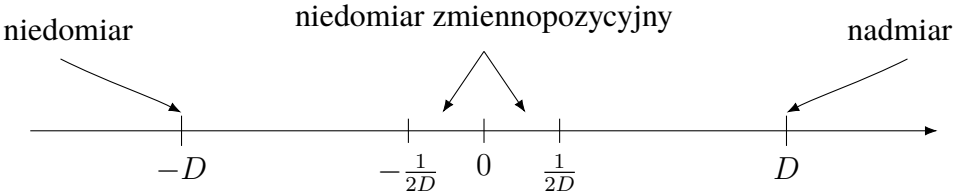
Pytanie: Które liczby rzeczywiste można dokładnie reprezentować w komputerze? Jaką one mają postać?

**TW.**

$$\left|\frac{\text{chop}(x) - x}{x}\right| \leq 2 \cdot 2^{-t}, \qquad \left|\frac{\text{rd}(x) - x}{x}\right| \leq 2^{-t}$$

Jakie liczby, tzn. z jakiego zakresu, 'zna' komputer?
Niech

$$C_{\max} = 2^{d-t-1} - 1, \qquad D := 2^{C_{\max}}.$$



W rzeczywistości w komputerze możemy reprezentować liczby ze zbioru $X' := (-D, D)$. W pamięci pojawiają się liczby ze zbioru dyskretnego $X_{fl} := \mathrm{rd}(X')$.

**Przykład.** Rozważmy arytmetykę dla

$$d = 5, \qquad t = 3.$$

Wtedy:

$$C_{\max} = 2^{d-t-1} - 1 = 2^1 - 1 = 1, \qquad D = 2^{C_{\max}} = 2.$$

Możliwe mantysy:

$$m \in \left\{ \frac{1}{2}, \frac{5}{8}, \frac{3}{4}, \frac{7}{8} \right\},$$

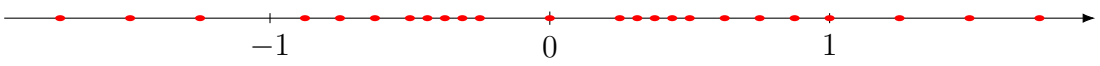
a cecha:

$$c \in \{-1, 0, 1\}.$$

Dodatnie liczby znormalizowane:

$$\left\{ \frac{1}{4}, \frac{5}{16}, \frac{3}{8}, \frac{7}{16}, \frac{1}{2}, \frac{5}{8}, \frac{3}{4}, \frac{7}{8}, 1, \frac{5}{4}, \frac{3}{2}, \frac{7}{4} \right\}.$$

Zbiór $X_{fl}$ jest symetryczny względem zera (dochodzi też liczba $0$).




Zmiennopozycyjna realizacja działań arytmetycznych:

$$\mathrm{fl}(x \circ y) := (x \circ y)\,(1 + \varepsilon_{x \circ y}), \qquad \circ \in \{+, -, *, /\}, \qquad x, y \in X_{fl}.$$

Błąd względny pojedynczej operacji spełnia:

$$|\varepsilon_{x \circ y}| \leq 2^{-t}.$$

Równoważnie:

$$\left| \frac{x \circ y - \mathrm{fl}(x \circ y)}{x \circ y} \right| = |\varepsilon_{x \circ y}|.$$

Aby analizować (symulować) działanie algorytmów w arytmetyce zmiennopozycyjnej, będziemy często posługiwać się tzw. twierdzeniem o kumulacji błędów.

**TW. (o kumulacji błędów)**
Niech zachodzi

$$|\delta_i| \leq 2^{-t}, \qquad i = 1, 2, 3, \ldots, n,$$

oraz niech

$$1 + \sigma_n := \prod_{i=1}^{n} (1 + \delta_i).$$

Wtedy

$$\sigma_n = \sum_{i=1}^{n} \delta_i + O\big(2^{-2t}\big).$$

Jeśli dodatkowo

$$n \cdot 2^{-t} < 2,$$

to

$$|\sigma_n| \le \gamma_n := \frac{n \cdot 2^{-t}}{1 - \frac{1}{2} n \cdot 2^{-t}} \approx n \cdot 2^{-t}.$$

Użyjemy twierdzenia o kumulacji błędów do analizy zmiennopozycyjnej realizacji prostego programu komputerowego.

**Przykład.** Rozważmy zadanie obliczenia sumy liczb $x_1, x_2, x_3, x_4, x_5$, tzn.

$$S = \sum_{i=1}^{5} x_i.$$

Wartość $S$ wyznaczamy przy pomocy następującego ('naturalnego') programu:

$S := x_1$
**for** $i = 2$ **to** $5$
  $S := S + x_i$
**return** $S$

Jak wygląda zmiennopozycyjna realizacja programu?

Dla uproszczenia przyjmijmy, że

$$\mathrm{rd}(x_i) = x_i, \qquad 1 \le i \le 5$$

tj. dane wejściowe są liczbami maszynowymi.

Wtedy realizacja zmiennopozycyjna programu ma postać:

$$\mathrm{fl}(P) = \Big(\Big(\big((x_1 + x_2)(1+\xi_2) + x_3\big)(1+\xi_3) + x_4\Big)(1+\xi_4) + x_5\Big)(1+\xi_5),$$

gdzie

$$|\xi_2|, |\xi_3|, |\xi_4|, |\xi_5| \le 2^{-t}.$$

Po uporządkowaniu względem $x_i$:

$$\begin{aligned}\mathrm{fl}(P) = {} & x_1(1+\xi_2)(1+\xi_3)(1+\xi_4)(1+\xi_5) \\ & + x_2(1+\xi_2)(1+\xi_3)(1+\xi_4)(1+\xi_5) \\ & + x_3(1+\xi_3)(1+\xi_4)(1+\xi_5) \\ & + x_4(1+\xi_4)(1+\xi_5) + x_5(1+\xi_5).\end{aligned}$$

Oznaczmy:

$$1 + E_i := \prod_{j=i}^{5}(1+\xi_j), \qquad i = 2,3,4,5, \qquad E_1 := E_2.$$

Wówczas

$$\mathrm{fl}(P) = \sum_{i=1}^{5} x_i(1+E_i).$$

Z twierdzenia o kumulacji błędów otrzymujemy (w pierwszym rzędzie):

$$|E_2| \le \gamma_4 \lesssim 4 \cdot 2^{-t}, \qquad |E_3| \le \gamma_3 \lesssim 3 \cdot 2^{-t},$$

$$|E_4| \le \gamma_2 \lesssim 2 \cdot 2^{-t}, \qquad |E_5| = |\xi_5| \le 2^{-t}.$$

Badamy błąd względny:

$$\left|\frac{S - \mathrm{fl}(P)}{S}\right| = \left|\frac{\sum_{i=1}^5 x_i - \sum_{i=1}^5 x_i(1+E_i)}{\sum_{i=1}^5 x_i}\right| = \left|\frac{\sum_{i=1}^5 x_i E_i}{\sum_{i=1}^5 x_i}\right|.$$

Stąd

$$\left|\frac{S - \mathrm{fl}(P)}{S}\right| \le \frac{\sum_{i=1}^5 |x_i||E_i|}{\left|\sum_{i=1}^5 x_i\right|} \le \left(\frac{\sum_{i=1}^5 |x_i|}{\left|\sum_{i=1}^5 x_i\right|}\right) \cdot 4 \cdot 2^{-t}.$$

Wprowadzamy oznaczenie:

$$K := \frac{\sum_{i=1}^5 |x_i|}{\left|\sum_{i=1}^5 x_i\right|}.$$

Wtedy

$$\left|\frac{S - \mathrm{fl}(P)}{S}\right| \lesssim K \cdot 4 \cdot 2^{-t}.$$

**Wniosek.**

- Jeśli sumujemy liczby dodatnie, to warto je najpierw posortować.
- Jeśli wszystkie $x_i$ mają ten sam znak, to $K = 1$.
- Może jednak być tak, że $K$ jest dowolnie duże.

## 2.3 Zjawisko utraty cyfr znaczących

Problem utraty cyfr znaczących prześledźmy na przykładzie.
Niech

$$x, y \in X_{fl}, \qquad x > y > 0, \qquad x \approx y.$$

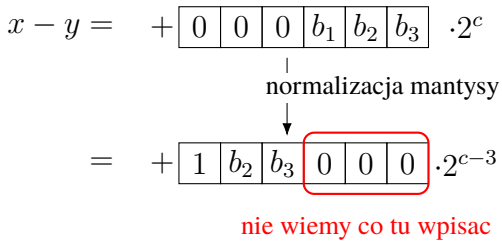
Przy odejmowaniu $x - y$ najpierw wyrównujemy cechy (przesuwamy mantysę jednej z liczb), a następnie odejmujemy mantysy. Gdy liczby są bliskie, najstarsze cyfry (bity) mantysy się redukują i wynik zaczyna się od wielu zer.
W efekcie:

- w wyniku zostaje mało cyfr znaczących,
- względny błąd wyniku może istotnie wzrosnąć.

$$x = \quad + \boxed{1}\boxed{a_1}\boxed{a_2}\boxed{a_3}\boxed{a_4}\boxed{a_5} \cdot 2^c$$

wyrownanie cech

$$y = \quad + \boxed{1}\boxed{a_1}\boxed{a_2}\boxed{a_3}\boxed{0}\boxed{0} \cdot 2^c$$




**Przykład numeryczny (kod demonstracyjny).**

```
f1:=z->ln(z)-1;

x:=exp(1.0001):

printf("         x = %1.10e\n\n",x);
printf("      f1(x) = %1.10e dla Digits:=%d\n ",f1(x),Digits);
printf(" Wynik dokladny = %1.30e\n\n",evalf(f1(x),64));

x = 2.7185536700e+00
f1(x) = 1.0000000000e-04 dla Digits:=10
Wynik dokladny = 9.99999140155506045938191304895 6e-05

f2:=z->ln(z/exp(1.0));
```

```
x:=exp(1.0001):

printf("         x = %1.10e\n\n",x);
printf("      f2(x) = %1.10e dla Digits:=%d\n ",f2(x),Digits);
printf(" Wynik dokladny = %1.30e\n\n",evalf(f1(x),64));

x = 2.7185536700e+00
f2(x) = 9.9999999830e-05 dla Digits:=10
Wynik dokladny = 9.99999140155506045938191304895 6e-05
```

## Wykład 3 Uwarunkowanie zadania, algorytmy numerycznie poprawne

**Przykład.** Rozważmy wielomian

$$\omega(x) = (x-1)(x-2)\cdots(x-20) = \sum_{i=0}^{20} a_i x^i = x^{20} - 210x^{19} + \dots$$

oraz jego zaburzenie

$$\widetilde{\omega}(x) := \omega(x) - \varepsilon x^{19}, \qquad \varepsilon \text{ małe.}$$

Przy bardzo małym $\varepsilon$ (np. $\varepsilon = 2^{-30}$) można zaobserwować:

- minimalna zmiana danych (współczynników),
- duża zmiana wyniku (położenia miejsc zerowych).

**Przykład (macierz Hilberta).**

$$H_n = \left[\frac{1}{i+j-1}\right] \in \mathbb{R}^{n\times n}, \qquad \widetilde{H}_n = \left[\mathrm{rd}\left(\frac{1}{i+j-1}\right)\right] \in \mathbb{R}^{n\times n}.$$

Badamy:

$$\det(H_n), \qquad \det(\widetilde{H}_n).$$

Dla większych $n$ (na tablicy: $n = 15$) wartości te mogą mieć nawet przeciwne znaki, co oznacza błąd względny rzędu $100\%$.

**Przykład (układ równań).** Rozważmy układ

$$H_n x = b_n, \qquad b_n = H_n \begin{bmatrix} 1 \\ 1 \\ \vdots \\ 1 \end{bmatrix},$$

oraz układ zaburzony

$$\widetilde{H}_n \widetilde{x} = \widetilde{b}_n, \qquad \widetilde{b}_n = \mathrm{fl}\left( H_n \begin{bmatrix} 1 \\ 1 \\ \vdots \\ 1 \end{bmatrix} \right).$$

W praktyce rozwiązania mogą się istotnie różnić, mimo małej zmiany danych.

**Definicja (uwarunkowanie zadania).** Zadanie nazywamy *źle uwarunkowanym*, jeśli mała względna zmiana danych powoduje dużą względną zmianę wyniku.

**Uwaga.**

- Dla zadań źle uwarunkowanych obliczenia komputerowe wymagają szczególnej ostrożności.
- Sprawdzenie, czy zadanie jest źle uwarunkowane, bywa trudne.

**Przykład (wartość funkcji).** Dla $f : \mathbb{R} \to \mathbb{R}$ badamy wpływ małej zmiany argumentu $x \to x + h$ na wartość funkcji, np. przez iloraz:

$$\left| \frac{f(x) - f(x+h)}{f(x)} \right|.$$

Korzystając z

$$f'(x) = \lim_{h \to 0} \frac{f(x+h) - f(x)}{h},$$

otrzymujemy dla małego $h$:

$$\left| \frac{f(x) - f(x+h)}{f(x)} \right| \approx \left| \frac{f(x+h) - f(x)}{h} \right| \frac{|h|}{|f(x)|} \approx |f'(x)| \frac{|h|}{|f(x)|}.$$

Ponieważ

$$\left| \frac{x - (x+h)}{x} \right| = \left| \frac{h}{x} \right|,$$

to

$$\left| \frac{f(x) - f(x+h)}{f(x)} \right| \approx \left| \frac{x f'(x)}{f(x)} \right| \left| \frac{h}{x} \right|.$$

Wielkość

$$\left| \frac{x f'(x)}{f(x)} \right|$$

można zatem przyjąć za miarę tego, jak względna zmiana danych wpływa na względną zmianę wyniku w zadaniu obliczania wartości funkcji.

**Przykład (iloczyn skalarny).** Rozważmy

$$a = \begin{bmatrix} a_1 \\ a_2 \\ \vdots \\ a_n \end{bmatrix} \in \mathbb{R}^n, \qquad b = \begin{bmatrix} b_1 \\ b_2 \\ \vdots \\ b_n \end{bmatrix} \in \mathbb{R}^n,$$

oraz funkcję

$$S(a, b) = \sum_{i=1}^{n} a_i b_i, \qquad S : \mathbb{R}^n \times \mathbb{R}^n \to \mathbb{R}.$$

Niech dane będą względnie zaburzone składowo:

$$\widetilde{a}_i = a_i(1 + \varepsilon_i), \qquad \widetilde{b}_i = b_i(1 + \delta_i),$$

czyli

$$\widetilde{a} = \begin{bmatrix} a_1(1+\varepsilon_1) \\ \vdots \\ a_n(1+\varepsilon_n) \end{bmatrix}, \qquad \widetilde{b} = \begin{bmatrix} b_1(1+\delta_1) \\ \vdots \\ b_n(1+\delta_n) \end{bmatrix}.$$

Wtedy

$$\left|\frac{S(a,b)-S(\widetilde{a},\widetilde{b})}{S(a,b)}\right| = \left|\frac{\sum_{i=1}^n a_ib_i - \sum_{i=1}^n a_ib_i(1+\varepsilon_i)(1+\delta_i)}{\sum_{i=1}^n a_ib_i}\right|$$
$$\approx \left|\frac{\sum_{i=1}^n a_ib_i(\varepsilon_i+\delta_i)}{\sum_{i=1}^n a_ib_i}\right|$$
$$\leq \frac{\sum_{i=1}^n |a_ib_i|\,|\varepsilon_i+\delta_i|}{|\sum_{i=1}^n a_ib_i|}$$
$$\leq \max_i |\varepsilon_i+\delta_i| \frac{\sum_{i=1}^n |a_ib_i|}{|\sum_{i=1}^n a_ib_i|}.$$

Definiujemy wskaźnik:

$$K(a,b) := \frac{\sum_{i=1}^n |a_ib_i|}{|\sum_{i=1}^n a_ib_i|}.$$

**Wnioski.**

- Jeśli $a_ib_i > 0$ (albo $a_ib_i < 0$) dla wszystkich $i = 1,\ldots,n$, to $K(a,b) = 1$.
- W szczególnych sytuacjach $K(a,b)$ może być dowolnie duże (np. $\sum_i |a_ib_i| = 1$, a $\sum_i a_ib_i \approx 0$).

**Definicja (wskaźnik uwarunkowania zadania).** Wielkość (lub wielkości), które opisują, jak względna zmiana danych wpływa na względną zmianę wyniku, nazywamy wskaźnikiem (wskaźnikami) uwarunkowania.

Umowa:

- jeśli wskaźnik uwarunkowania jest ograniczony, to zadanie jest dobrze uwarunkowane,
- jeśli jest nieograniczony, to zadanie jest źle uwarunkowane.

**Algorytmy numerycznie poprawne.**

**Definicja (algorytm numerycznie poprawny).** Algorytm nazywamy numerycznie poprawnym, jeśli wynik uzyskany w arytmetyce zmiennopozycyjnej może być zinterpretowany jako mało zaburzony wynik dokładny dla mało zaburzonych danych.

**Przykład.** Niech dane będą liczby $x_1, x_2, \ldots, x_n$ i rozważmy algorytm:

$S := x_1$
**for** $i = 2$ **to** $n$
  $S := S + x_i$
**return** $S$

Zakładamy, że $x_i = \mathrm{rd}(x_i)$ (dane wejściowe są maszynowe) oraz

$$|\varepsilon_i| \leq 2^{-t} \quad (i = 2,\ldots,n), \qquad \varepsilon_1 := 0.$$

Wtedy

$$\mathrm{fl}(S) = \sum_{i=1}^n x_i \prod_{j=i}^n (1+\varepsilon_j).$$

Wprowadzamy oznaczenie:

$$1 + E_i := \prod_{j=i}^{n} (1 + \varepsilon_j), \qquad i = 2, \ldots, n, \qquad E_1 := E_2.$$

Otrzymujemy

$$\mathrm{fl}(S) = \sum_{i=1}^{n} x_i(1 + E_i) = \sum_{i=1}^{n} \widehat{x}_i, \qquad \widehat{x}_i := x_i(1 + E_i).$$

Z twierdzenia o kumulacji błędów:

$$|E_i| \leq (n - i + 1)\, 2^{-t} \quad (i = 2, \ldots, n), \qquad |E_1| \leq (n - 1)\, 2^{-t}.$$

A więc wynik obliczony jest dokładną sumą lekko zaburzonych danych $\widehat{x}_i$.
**Konkluzja.** Ten algorytm jest numerycznie poprawny.