

# **Analiza Numeryczna**

Skrypt z wykładów  
Semestr zimowy 2023/2024

Uniwersytet Wrocławski  
Wydział Matematyki i Informatyki  
Instytut Informatyki



# Spis treści

<b>1</b>	<b>2</b>
<b>2 Arytmetyka zmiennopozycyjna, podstawy teorii błędów</b>	<b>3</b>
2.1 Błąd bezwzględny i błąd względny . . . . .	3
2.2 Reprezentacja liczb w komputerze . . . . .	3
2.3 Zjawisko utraty cyfr znaczących . . . . .	7
<b>3 Uwarunkowanie zadania, algorytmy numerycznie poprawne</b>	<b>9</b>
<b>4 Rozwiązywanie równań nieliniowych</b>	<b>13</b>
4.1 Metoda bisekcji . . . . .	13
4.2 Metoda Newtona . . . . .	14
4.3 Metoda siecznych . . . . .	14
4.4 Warunki stopu . . . . .	14
4.5 Wykładnik Zbieżności (Rząd Metody) . . . . .	15
4.6 Twierdzenie (metody jednokrokowe) . . . . .	15
<b>5 Interpolacja wielomianowa – część I</b>	<b>16</b>
5.1 Postaci wielomianu . . . . .	16
5.2 Interpolacja wielomianowa . . . . .	17

# **Wykład 1**

## Wykład 2 Arytmetyka zmiennopozycyjna, podstawy teorii błędów

### 2.1 Błąd bezwzględny i błąd 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:

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

Łącznie:  $d + 1$  bitów na liczbę rzeczywistą ze znakiem.

Ten model reprezentacji jest teoretyczny. W praktyce stosujemy standard IEEE 754.

**TW.**

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

Reprezentacja zmiennopozycyjna liczby rzeczywistej  $x \neq 0$ :

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

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

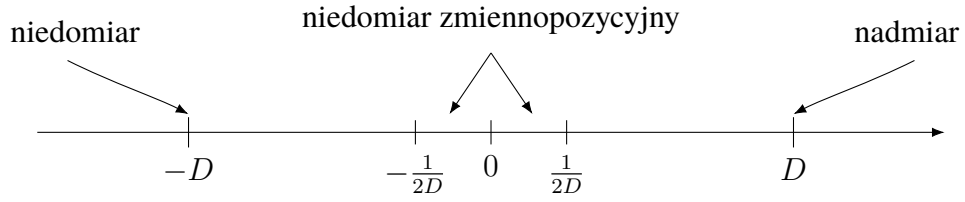
**TW.**

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

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

Niech

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



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

**Przykład.** Rozważmy arytmetykę dla

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

Wtedy:

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

Możliwe mantysy:

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

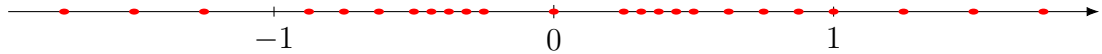
a cecha:

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

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:

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

Błąd względny pojedynczej operacji spełnia:

$$|\varepsilon_{xoy}| \leq 2^{-t}.$$

Równoważnie:

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

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

**TW. (o kumulacji błędów)**

Niech zachodzi

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

oraz niech

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

Wtedy

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

Jeśli dodatkowo

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

to

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

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 := x1
for i = 2 to 5
    S := S + xi
return S

```

Jak wygląda zmiennopozycyjna realizacja programu?

Dla uproszczenia przyjmijmy, że

$$\text{rd}(x_i) = x_i, \quad 1 \leq i \leq 5$$

tj. dane wejściowe są liczbami maszynowymi.

Wtedy realizacja zmiennopozycyjna programu ma postać:

$$\text{fl}(P) = \left( \left( \left( (x_1 + x_2)(1 + \xi_2) + x_3 \right)(1 + \xi_3) + x_4 \right)(1 + \xi_4) + x_5 \right)(1 + \xi_5),$$

gdzie

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

Po uporządkowaniu względem  $x_i$ :

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



Oznaczmy:

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

Wówczas

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

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

$$\begin{aligned} |E_2| &\leq \gamma_4 \lesssim 4 \cdot 2^{-t}, & |E_3| &\leq \gamma_3 \lesssim 3 \cdot 2^{-t}, \\ |E_4| &\leq \gamma_2 \lesssim 2 \cdot 2^{-t}, & |E_5| &= |\xi_5| \leq 2^{-t}. \end{aligned}$$

Badamy błąd względny:

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

Stąd

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

Wprowadzamy oznaczenie:

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

Wtedy

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

**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

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

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.

$$\begin{aligned}
 x &= + \boxed{1 \mid a_1 \mid a_2 \mid a_3 \mid a_4 \mid a_5} \cdot 2^c \\
 &\quad \downarrow \text{wyrównanie cech} \\
 y &= + \boxed{1 \mid a_1 \mid a_2 \mid a_3 \mid 0 \mid 0} \cdot 2^c \\
 x - y &= + \boxed{0 \mid 0 \mid 0 \mid b_1 \mid b_2 \mid b_3} \cdot 2^c \\
 &\quad \downarrow \text{normalizacja mantysy} \\
 &= + \boxed{1 \mid b_2 \mid b_3 \mid 0 \mid 0 \mid 0} \cdot 2^{c-3}
 \end{aligned}$$

nie wiemy co tu wpisać

### 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 dokładny = %1.30e\n\n",evalf(f1(x),64));

x = 2.7185536700e+00
f1(x) = 1.0000000000e-04 dla Digits:=10
Wynik dokładny = 9.999991401555060459381913048956e-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 dokładny = %1.30e\n\n",evalf(f1(x),64));

x = 2.7185536700e+00
f2(x) = 9.9999999830e-05 dla Digits:=10
Wynik dokładny = 9.999991401555060459381913048956e-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

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

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

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

**Przykład (macierz Hilberta).**

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

Badamy:

$$\det(H_n), \quad \det(\tilde{H}_n).$$

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

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

oraz układ zaburzony

$$\tilde{H}_n \tilde{x} = \tilde{b}_n, \quad \tilde{b}_n = \text{fl} \left( H_n \begin{bmatrix} 1 \\ 1 \\ \vdots \\ 1 \end{bmatrix} \right).$$

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  $f : \mathbb{R} \rightarrow \mathbb{R}$  badamy wpływ małej zmiany argumentu  $x \rightarrow x + h$  na wartość funkcji, np. przez iloraz:

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

Korzystając z

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

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

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

Wielkość

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

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

**Przykład (iloczyn skalarny).** Rozważmy

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

oraz funkcję

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

Niech dane będą względnie zaburzone składowo:

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

czyli

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

Wtedy

$$\begin{aligned}
\left| \frac{S(a, b) - S(\tilde{a}, \tilde{b})}{S(a, b)} \right| &= \left| \frac{\sum_{i=1}^n a_i b_i - \sum_{i=1}^n a_i b_i (1 + \varepsilon_i)(1 + \delta_i)}{\sum_{i=1}^n a_i b_i} \right| \\
&\approx \left| \frac{\sum_{i=1}^n a_i b_i (\varepsilon_i + \delta_i)}{\sum_{i=1}^n a_i b_i} \right| \\
&\leq \frac{\sum_{i=1}^n |a_i b_i| |\varepsilon_i + \delta_i|}{|\sum_{i=1}^n a_i b_i|} \\
&\leq \max_i |\varepsilon_i + \delta_i| \frac{\sum_{i=1}^n |a_i b_i|}{|\sum_{i=1}^n a_i b_i|}.
\end{aligned}$$

Definiujemy wskaźnik:

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

**Wnioski.**

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

**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  $x_1, x_2, \dots, x_n$  i rozważmy algorytm:

```

S := x1
for i = 2 to n
    S := S + xi
return S

```

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

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

Wtedy

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

Wprowadzamy oznaczenie:

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

Otrzymujemy

$$\text{fl}(S) = \sum_{i=1}^n x_i(1 + E_i) = \sum_{i=1}^n \hat{x}_i, \quad \hat{x}_i := x_i(1 + E_i).$$

Z twierdzenia o kumulacji błędów:

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

A więc wynik obliczony jest dokładną sumą lekko zaburzonych danych  $\hat{x}_i$ .

**Konkluzja.** Ten algorytm jest numerycznie poprawny.

## Wykład 4 Rozwiązywanie równań nieliniowych

Zadanie: dla danej funkcji  $f : \mathbb{R} \rightarrow \mathbb{R}$  znaleźć takie  $\alpha \in \mathbb{R}$ , dla którego

$$f(\alpha) = 0.$$

W przypadku równań nieliniowych rozwiązanie analityczne najczęściej nie jest możliwe, dlatego stosujemy metody przybliżone (iteracyjne), zwykle z użyciem komputera.

Omówimy trzy podstawowe metody:

- metodę bisekcji,
- metodę Newtona,
- metodę siecznych.

### 4.1 Metoda bisekcji

Założenia:

- $f$  jest ciągła na przedziale  $(a_0, b_0)$ ,
- istnieje dokładnie jedno miejsce zerowe  $\alpha \in (a_0, b_0)$ ,
- $f(a_0)f(b_0) < 0$  (np.  $f(a_0) < 0 < f(b_0)$ ).

Konstrukcja:

- $I_k = [a_k, b_k]$  dla  $k = 0, 1, 2, \dots$ ,
- środek przedziału  $m_{k+1} := \frac{a_k + b_k}{2}$ ,
- jeśli  $f(m_{k+1}) = 0$ , to  $\alpha = m_{k+1}$ ,
- w przeciwnym razie:

$$I_{k+1} = \begin{cases} [a_k, m_{k+1}], & \text{gdy } f(a_k)f(m_{k+1}) < 0, \\ [m_{k+1}, b_k], & \text{gdy } f(m_{k+1})f(b_k) < 0. \end{cases}$$

**Obserwacje.**

- Długość przedziału po  $k$  krokach:

$$|I_k| = \frac{b_0 - a_0}{2^k} \xrightarrow{k \rightarrow \infty} 0.$$

- Metoda bisekcji jest zbieżna i praktycznie niezawodna (dla spełnionych założeń).
- Aby otrzymać przedział długości co najwyżej  $2\varepsilon$ , wystarczy wykonać

$$N = \left\lceil \log_2 \left( \frac{b_0 - a_0}{2\varepsilon} \right) \right\rceil$$

kroków (często zapisywane także jako  $\lfloor \cdot \rfloor + 1$ ).

## 4.2 Metoda Newtona

Punkt startowy  $x_0$  dany, a iteracje:

$$x_{n+1} = x_n - \frac{f(x_n)}{f'(x_n)}, \quad n = 0, 1, 2, \dots$$

Interpretacja geometryczna:  $x_{n+1}$  jest miejscem zerowym stycznej do wykresu  $y = f(x)$  w punkcie  $(x_n, f(x_n))$ .

## 4.3 Metoda siecznych

Punkty startowe  $x_0, x_1$  dane, a iteracje:

$$x_{n+1} = x_n - \frac{f(x_n)(x_n - x_{n-1})}{f(x_n) - f(x_{n-1})}, \quad n = 1, 2, 3, \dots$$

Interpretacja geometryczna:  $x_{n+1}$  jest miejscem zerowym prostej przechodzącej przez punkty  $(x_n, f(x_n))$  oraz  $(x_{n-1}, f(x_{n-1}))$ .

Metoda siecznych jest odpowiedzią na wadę metody Newtona polegającą na konieczności liczenia pochodnej.

## 4.4 Warunki stopu

W praktyce zamiast warunku idealnego  $f(\alpha) = 0$  stosuje się kryteria przybliżone:

- mała wartość reszty:  $|f(x_n)|$  małe,
- mały krok iteracji (np. względny):

$$\left| \frac{x_n - x_{n-1}}{x_n} \right| \text{ małe,}$$

- ograniczenie liczby iteracji:  $n \leq N_{\max}$ .



## 4.5 Wykładnik Zbieżności (Rząd Metody)

Niech  $x_n \rightarrow \alpha$ . Mówimy, że ciąg  $(x_n)$  ma asymptotyczny rząd zbieżności  $p$ , jeśli istnieją  $C > 0$  oraz  $p \geq 1$  takie, że

$$\lim_{n \rightarrow \infty} \frac{|x_{n+1} - \alpha|}{|x_n - \alpha|^p} = C.$$

Wtedy (dla dużych  $n$ ):

$$|x_{n+1} - \alpha| \approx C |x_n - \alpha|^p.$$

Przypadki szczególne:

- $p = 1, C \in (0, 1)$ : zbieżność liniowa,
- $p = 2$ : zbieżność kwadratowa,
- $p = 3$ : zbieżność sześcienna.

Im większe  $p$ , tym szybciej (asymptotycznie) maleje błąd.

**Uwaga.** Porównywanie metod wyłącznie przez rząd zbieżności ma sens przede wszystkim dla metod jednokrokowych.

## 4.6 Twierdzenie (metody jednokrokowe)

Niech metoda jednokrokowa ma postać

$$x_0 \text{ dane,} \quad x_{n+1} = F(x_n), \quad n \geq 0,$$

i niech  $x_n \rightarrow \alpha$ , gdzie  $\alpha$  jest miejscem zerowym  $f$ .

Wtedy rząd zbieżności  $p$  tej metody jest liczbą naturalną i zachodzi równoważność:

$$p \text{ jest rzędem metody} \iff \begin{cases} F(\alpha) = \alpha, \\ F'(\alpha) = F''(\alpha) = \dots = F^{(p-1)}(\alpha) = 0, \\ F^{(p)}(\alpha) \neq 0. \end{cases}$$

Dodatkowo stała asymptotyczna wynosi

$$C = \frac{|F^{(p)}(\alpha)|}{p!}.$$

## Wykład 5 Interpolacja wielomianowa – część I

### Powtórka z wykładu 4.

- Bisekcja: rząd zbieżności  $p = 1$ .
- Newton: rząd zbieżności  $p = 2$  (lokalnie, przy spełnionych założeniach).
- Sieczne: rząd zbieżności  $p \approx 1.618$ .

## 5.1 Postaci wielomianu

Oznaczenie:

$$\Pi_n := \{w : w \text{ jest wielomianem stopnia } \leq n\}.$$

**a) Postać naturalna (potęgowa).** Dla  $w \in \Pi_n$ :

$$w(x) = \sum_{i=0}^n a_i x^i.$$

**Schemat Hornera.**

$$w(x) = (((a_n x + a_{n-1})x + a_{n-2})x + \cdots + a_1)x + a_0.$$

Algorytm:

$$\begin{aligned} w_n &:= a_n, \\ w_k &:= w_{k+1}x + a_k, \quad k = n-1, n-2, \dots, 0. \end{aligned}$$

Wtedy  $w(x) = w_0$ . Koszt obliczeniowy:  $O(n)$ .

**b) Postać Newtona.** Niech  $x_0, x_1, \dots, x_n$  będą ustalonymi punktami. Wtedy

$$w(x) = \sum_{k=0}^n b_k P_k(x),$$

gdzie

$$P_0(x) = 1, \quad P_k(x) = \prod_{i=0}^{k-1} (x - x_i) \quad (k \geq 1).$$

Czyli:

$$w(x) = b_0 + b_1(x - x_0) + b_2(x - x_0)(x - x_1) + \cdots + b_n \prod_{i=0}^{n-1} (x - x_i).$$

### Uogólniony schemat Hornera (dla postaci Newtona).

$$\begin{aligned}w_n &:= b_n, \\w_k &:= w_{k+1}(x - x_k) + b_k, \quad k = n-1, n-2, \dots, 0.\end{aligned}$$

Wtedy  $w(x) = w_0$ .

**c) Postać Czebyszewa.** Wielomiany Czebyszewa definiujemy rekurencyjnie:

$$T_0(x) = 1, \quad T_1(x) = x, \quad T_k(x) = 2xT_{k-1}(x) - T_{k-2}(x) \quad (k \geq 2).$$

### Własności (podstawowe).

- $T_k \in \Pi_k \setminus \Pi_{k-1}$ .
- Dla  $n \geq 1$ :  $T_n(x) = 2^{n-1}x^n + \dots$
- Parzystość:  $T_n(-x) = (-1)^n T_n(x)$ .
- Dla  $x \in [-1, 1]$ :  $T_n(x) = \cos(n \arccos(x))$ .
- Wszystkie miejsca zerowe  $T_n$  są rzeczywiste, pojedyncze i należą do  $(-1, 1)$ .
- Dla  $x \in [-1, 1]$ :  $|T_n(x)| \leq 1$ .
- $\text{lin}\{T_0, T_1, \dots, T_n\} = \Pi_n$ .

Postać Czebyszewa wielomianu:

$$w(x) = \frac{1}{2}c_0T_0(x) + c_1T_1(x) + \dots + c_nT_n(x) = \sum_{k=0}^n {}'c_k T_k(x),$$

gdzie kreska przy sumie oznacza, że składnik dla  $k = 0$  jest liczony z czynnikiem  $\frac{1}{2}$ .

**Algorytm Clenshawa** (obliczanie wartości w postaci Czebyszewa, koszt  $O(n)$ ):

$$\begin{aligned}B_{n+2} &:= 0, \quad B_{n+1} := 0, \\B_k &:= 2xB_{k+1} - B_{k+2} + c_k, \quad k = n, n-1, \dots, 0.\end{aligned}$$

Wtedy

$$w(x) = \frac{B_0 - B_2}{2}.$$

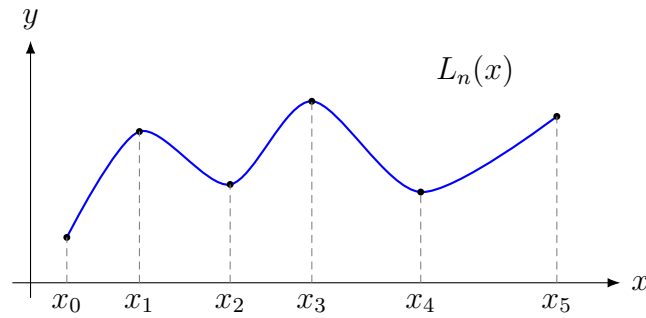
## 5.2 Interpolacja wielomianowa

Mamy dane pomiary:

$$x_0, x_1, \dots, x_n \in \mathbb{R}, \quad x_i \neq x_j \quad (i \neq j), \quad y_0, y_1, \dots, y_n \in \mathbb{R}.$$

Szukamy wielomianu  $L_n \in \Pi_n$  takiego, że

$$L_n(x_k) = y_k, \quad k = 0, 1, \dots, n.$$



**Twierdzenie.** Zadanie interpolacyjne Lagrange’a ma zawsze dokładnie jedno rozwiązanie.

**Postać Lagrange’a.**

$$L_n(x) = \sum_{k=0}^n \lambda_k(x) y_k,$$

gdzie

$$\lambda_k(x) := \prod_{\substack{i=0 \\ i \neq k}}^n \frac{x - x_i}{x_k - x_i}, \quad k = 0, 1, \dots, n.$$

Własność bazowa:

$$\lambda_k(x_i) = \begin{cases} 1, & i = k, \\ 0, & i \neq k. \end{cases}$$

**Przykład.** Niech  $f(x) = e^x$ ,  $n = 3$ , węzły:

$$x_0 = 0, \quad x_1 = 0.2, \quad x_2 = 0.6, \quad x_3 = 0.8.$$

Budujemy  $L_3$  i przybliżamy  $f(0.4)$ . W zapisie z tablicy:

$$L_3(x) = \lambda_0(x)y_0 + \lambda_1(x)y_1 + \lambda_2(x)y_2 + \lambda_3(x)y_3, \quad y_k = f(x_k).$$

Otrzymane przybliżenie:

$$L_3(0.4) \approx 1.49142, \quad f(0.4) \approx 1.49182.$$

Zatem błąd punktowy jest rzędu  $10^{-3}$ .

**Przykłady (jak na tablicy).**

- $f(x) = \sin(x)$ ,

$$x_k = \frac{2k\pi}{n}, \quad 0 \leq k \leq n, \quad n = 1, \dots, 15,$$

i porównujemy z interpolantem  $L_n(x)$ .

- $f(x) = |x|$ ,

$$x_k = -1 + \frac{2k}{n}, \quad 0 \leq k \leq n, \quad n = 1, \dots, 15,$$

i obserwujemy jakość przybliżenia przez  $L_n(x)$ .

- $f(x) = x^6$ ,

$$x_k = -2 + \frac{4k}{n}, \quad 0 \leq k \leq n, \quad n = 1, \dots, 6.$$

Ponieważ  $f \in \Pi_6$ , dla  $n \geq 6$  mamy dokładnie

$$L_n(x) = x^6 \quad (\text{w szczególności } L_6(x) = x^6).$$