

# **Analiza Numeryczna**

Skrypt z wykładów  
Semestr zimowy 2023/2024

Uniwersytet Wrocławski  
Wydział Matematyki i Informatyki  
Instytut Informatyki



# Spis treści

<b>1</b>	<b>3</b>
<b>2 Arytmetyka zmiennopozycyjna, podstawy teorii błędów</b>	<b>4</b>
2.1 Błąd bezwzględny i błąd względny . . . . .	4
2.2 Reprezentacja liczb w komputerze . . . . .	4
2.3 Zjawisko utraty cyfr znaczących . . . . .	8
<b>3 Uwarunkowanie zadania, algorytmy numerycznie poprawne</b>	<b>10</b>
<b>4 Rozwiązywanie równań nieliniowych</b>	<b>14</b>
4.1 Metoda bisekcji . . . . .	14
4.2 Metoda Newtona . . . . .	15
4.3 Metoda siecznych . . . . .	15
4.4 Warunki stopu . . . . .	15
4.5 Wykładnik Zbieżności (Rząd Metody) . . . . .	16
4.6 Twierdzenie (metody jednokrokowe) . . . . .	16
<b>5 Interpolacja wielomianowa – część I</b>	<b>17</b>
5.1 Postacie wielomianu . . . . .	17
5.2 Interpolacja wielomianowa . . . . .	18
<b>6 Interpolacja wielomianowa – część II</b>	<b>21</b>
6.1 Postać Newtona wielomianu interpolacyjnego . . . . .	21
6.2 Ilorazy różnicowe . . . . .	21
6.3 Koszty i aktualizacja . . . . .	22
6.4 Uwagi numeryczne . . . . .	22
6.5 Błąd interpolacji i wybór węzłów . . . . .	23
<b>7 Naturalne interpolacyjne funkcje sklepane 3-go stopnia i ich zastosowania w grafice komputerowej</b>	<b>24</b>
7.1 Definicja NIFS3 . . . . .	24
7.2 Postać klamrowa i przykład . . . . .	25
7.3 Momenty i układ trójprzekątniowy . . . . .	25
7.4 Algorytm $O(n)$ (schemat z tablicy) . . . . .	25
7.5 Istnienie i jednoznaczność . . . . .	26
7.6 Zastosowanie w grafice komputerowej . . . . .	26
<b>8 Krzywe Béziera i ich zastosowania w grafice komputerowej</b>	<b>27</b>
8.1 Oznaczenia i kombinacje wypukłe . . . . .	27
8.2 Wielomiany Bernsteina . . . . .	27

8.3	Krzywa Béziera . . . . .	28
8.4	Algorytm de Casteljau . . . . .	28
8.5	Uwagi praktyczne . . . . .	29
<b>9</b>	<b>Aproksymacja średniokwadratowa na zbiorze dyskretnym</b>	<b>30</b>
9.1	Idea aproksymacji . . . . .	30
9.2	Dyskretna norma średniokwadratowa . . . . .	30
9.3	Zadanie aproksymacji . . . . .	31
9.4	Model jednoparametrowy: funkcje stałe . . . . .	31
9.5	Model jednoparametrowy: $\{ax^2\}$ . . . . .	31
9.6	Model jednoparametrowy: $\{ae^x\}$ . . . . .	32
9.7	Model dwuparametrowy: prosta regresji . . . . .	32
9.8	Model ogólny . . . . .	33
9.9	Pochodne cząstkowe – uwaga techniczna . . . . .	33
<b>10</b>	<b>Wielomianowa aproksymacja średniokwadratowa na zbiorze dyskretnym</b>	<b>34</b>
10.1	Zadanie wykładu 10 . . . . .	34
10.2	Dyskretny iloczyn skalarny . . . . .	34
10.3	Układ i ciąg ortogonalny . . . . .	35
10.4	Rekurencja trójczłonowa . . . . .	35
10.5	Postać rozwiązania optymalnego . . . . .	35
10.6	Obliczanie wartości wielomianu . . . . .	36
10.7	Podsumowanie . . . . .	36

## **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?

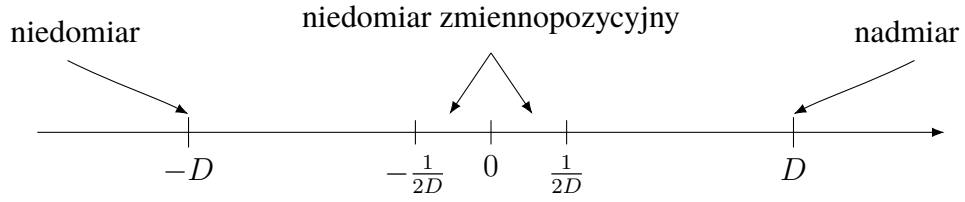
**TW.**

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

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:

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

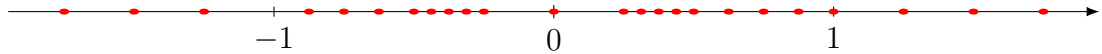
a cecha:

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

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:

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

Błąd względny pojedynczej operacji spełnia:

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

Równoważnie:

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

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

**TW. (o kumulacji błędów)**

Niech zachodzi

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



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) = 9.999999830e-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 Postacie 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

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

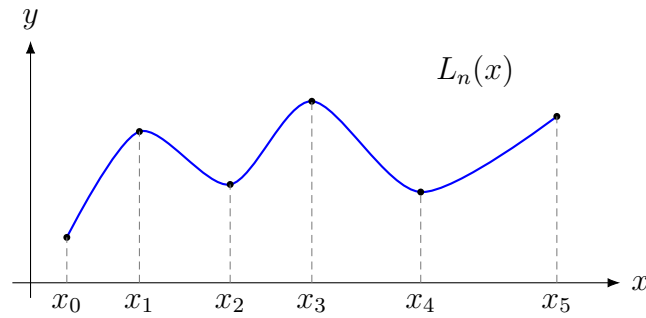
## 5.2 Interpolacja wielomianowa

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

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

## Wykład 6 Interpolacja wielomianowa – część II

### Plan wykładu:

- postać Newtona,
- ilorazy różnicowe,
- „dobry” wybór węzłów interpolacji.

**Przypomnienie.** Dla par parami różnych węzłów  $x_0, \dots, x_n$  oraz danych  $y_0, \dots, y_n$  szukamy  $L_n \in \Pi_n$  takiego, że

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

W postaci Lagrange’a:

$$L_n(x) = \sum_{k=0}^n y_k \lambda_k(x), \quad \lambda_k(x) = \prod_{\substack{j=0 \\ j \neq k}}^n \frac{x - x_j}{x_k - x_j}.$$

### 6.1 Postać Newtona wielomianu interpolacyjnego

Niech

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

Szukamy

$$L_n(x) = b_0 P_0(x) + b_1 P_1(x) + \dots + b_n P_n(x).$$

Współczynniki  $b_k$  wyznacza się z warunków interpolacji, co prowadzi do układu trójkątnego (koszt  $O(n^2)$ ).

**Wzór jawny (z tablicy):**

$$b_k = \sum_{i=0}^k \frac{y_i}{\prod_{\substack{j=0 \\ j \neq i}}^k (x_i - x_j)}, \quad k = 0, 1, \dots, n.$$

### 6.2 Ilorazy różnicowe

**Definicja.** Niech  $x_0, \dots, x_n$  będą parami różne i niech  $f(x_k) = y_k$ . Definiujemy:

$$f[x_i] = f(x_i) = y_i,$$

oraz rekurencyjnie

$$f[x_i, \dots, x_k] := \frac{f[x_{i+1}, \dots, x_k] - f[x_i, \dots, x_{k-1}]}{x_k - x_i}, \quad i < k.$$

**Przykład (rzęd 2):**

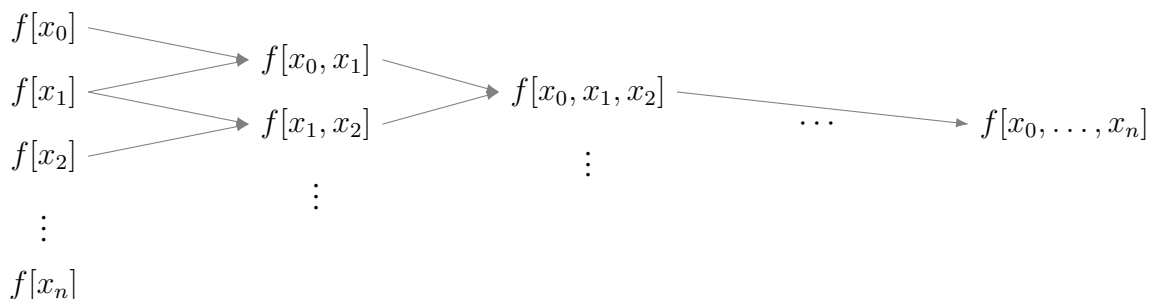
$$f[x_0, x_1, x_2] = \frac{f[x_1, x_2] - f[x_0, x_1]}{x_2 - x_0}.$$

**Twierdzenie (postać Newtona przez ilorazy różnicowe).**

$$L_n(x) = \sum_{k=0}^n f[x_0, x_1, \dots, x_k] P_k(x),$$

czyli

$$b_k = f[x_0, \dots, x_k].$$



## 6.3 Koszty i aktualizacja

**Koszt jednej konstrukcji:**

- tablica ilorazów różnicowych:  $O(n^2)$ ,
- obliczenie  $L_n(x)$  dla ustalonego  $x$ :  $O(n)$  (uogólniony Horner).

Przy wielu punktach  $z_0, \dots, z_M$ :

$$\text{koszt} \approx O(n^2) + (M + 1) O(n),$$

co jest tańsze niż wielokrotne liczenie postaci Lagrange'a od zera.

Jeśli dodamy jedną obserwację  $(x_{n+1}, y_{n+1})$  i mamy zapamiętaną ostatnią kolumnę tablicy ilorazów różnicowych, aktualizacja do  $L_{n+1}$  kosztuje  $O(n)$ .

## 6.4 Uwagi numeryczne

- Przed budową tablicy ilorazów różnicowych zaleca się uporządkować węzły.
- Naiwny algorytm wypełniania tablicy ilorazów różnicowych nie jest numerycznie poprawny.
- Dla dużej liczby węzłów (na tablicy: rzędu  $n \gtrsim 30$ ) mogą pojawiać się istotne problemy numeryczne.



## 6.5 Błąd interpolacji i wybór węzłów

**Twierdzenie (reszta interpolacji).** Jeśli  $f \in C^{n+1}([a, b])$ , to dla każdego  $x \in (a, b)$  istnieje  $\xi_x \in (a, b)$  takie, że

$$f(x) - L_n(x) = \frac{f^{(n+1)}(\xi_x)}{(n+1)!} \prod_{k=0}^n (x - x_k).$$

Stąd oszacowanie:

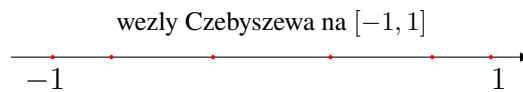
$$\max_{x \in [a, b]} |f(x) - L_n(x)| \leq \max_{x \in [a, b]} \left| \frac{f^{(n+1)}(x)}{(n+1)!} \right| \cdot \max_{x \in [a, b]} \left| \prod_{k=0}^n (x - x_k) \right|.$$

Żeby zmniejszyć błąd, chcemy minimalizować

$$\max_{x \in [a, b]} \left| \prod_{k=0}^n (x - x_k) \right|.$$

**Fakt.** Dla  $[a, b] = [-1, 1]$  minimum osiągają węzły Czebyszewa:

$$x_k = \cos\left(\frac{2k+1}{2n+2}\pi\right), \quad 0 \leq k \leq n.$$



Dla ogólnego przedziału  $[a, b]$  stosujemy przeskalowanie:

$$t_k = \frac{a+b}{2} + \frac{b-a}{2} \cos\left(\frac{2k+1}{2n+2}\pi\right), \quad 0 \leq k \leq n.$$

## Wykład 7 Naturalne interpolacyjne funkcje sklepane 3-go stopnia i ich zastosowania w grafice komputerowej

### Powtórka.

- Postać Newtona + ilorazy różnicowe: budowa w  $O(n^2)$ .
- Obliczanie wartości wielomianu Newtona:  $O(n)$  (uogólniony schemat Hornera).
- Błąd interpolacji:

$$|f(x) - L_n(x)| \leq \max_{t \in [a,b]} \left| \frac{f^{(n+1)}(t)}{(n+1)!} \right| \cdot \max_{x \in [a,b]} \left| \prod_{k=0}^n (x - x_k) \right|.$$

- Dobór „dobrych” węzłów: węzły Czebyszewa.

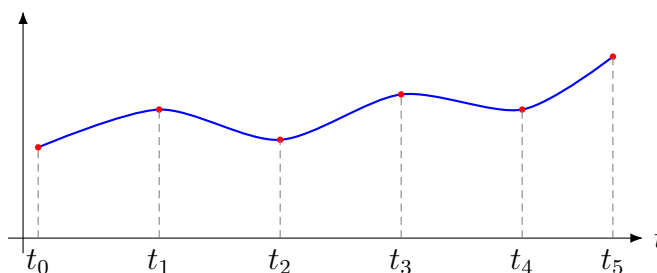
### 7.1 Definicja NIFS3

Niech

$$a = t_0 < t_1 < \dots < t_n = b, \quad y_k = f(t_k).$$

Funkcję  $s$  nazywamy **NIFS3** (naturalną interpolacyjną funkcją sklejaną stopnia 3), jeśli:

- $s(t_k) = y_k$  dla  $k = 0, \dots, n$ ,
- na każdym  $[t_{k-1}, t_k]$  funkcja  $s$  jest wielomianem stopnia co najwyżej 3,
- $s, s', s'' \in C[a, b]$ ,
- warunki naturalności:  $s''(a) = s''(b) = 0$ .



Ile warunków?

- interpolacja w węzłach:  $n + 1$ ,
- ciągłość  $s'$  i  $s''$  w węzłach wewnętrznych:  $2(n - 1)$ ,
- naturalność: 2.

Razem:  $4n$  warunków (tyle co współczynników dla  $n$  wielomianów kubicznych).

## 7.2 Postać kłamrowa i przykład

Na przedziale  $[t_{k-1}, t_k]$ :

$$s(t) = s_k(t) = A_k t^3 + B_k t^2 + C_k t + D_k.$$

**Przykład (z tablicy):** dla danych  $(t_0, y_0) = (-1, 1)$ ,  $(t_1, y_1) = (0, -1)$ ,  $(t_2, y_2) = (1, 1)$  otrzymujemy

$$s(t) = \begin{cases} t^3 + 3t^2 - 1, & t \in [-1, 0], \\ -t^3 + 3t^2 - 1, & t \in [0, 1]. \end{cases}$$

**Wniosek.** Bezpośrednie rozwiązywanie pełnego układu dla współczynników kłamrowych jest mało efektywne dla dużych  $n$ .

## 7.3 Momenty i układ trójpzekątniowy

Wprowadzamy momenty:

$$M_k := s''(t_k), \quad k = 0, 1, \dots, n, \quad M_0 = M_n = 0.$$

Niech

$$h_k := t_k - t_{k-1}, \quad \lambda_k := \frac{h_k}{h_k + h_{k+1}}, \quad d_k := 6 f[t_{k-1}, t_k, t_{k+1}], \quad k = 1, \dots, n-1.$$

Momenty spełniają układ:

$$\lambda_k M_{k-1} + 2M_k + (1 - \lambda_k) M_{k+1} = d_k, \quad k = 1, \dots, n-1.$$

Jest to układ liniowy trójpzekątniowy, więc można go rozwiązać w czasie  $O(n)$ .

**Uwaga.** Na egzaminie nie używamy bezpośrednio gotowego wzoru jawnego na segment  $s_k(t)$ .

## 7.4 Algorytm $O(n)$ (schemat z tablicy)

Obliczamy pomocnicze wielkości rekurencyjnie:

$$\begin{aligned} q_0 &:= 0, & u_0 &:= 0, \\ p_k &:= \lambda_k q_{k-1} + 2, \\ q_k &:= \frac{\lambda_k - 1}{p_k}, & k &= 1, 2, \dots, n-1. \\ u_k &:= \frac{d_k - \lambda_k u_{k-1}}{p_k}, \end{aligned}$$

Następnie:

$$M_{n-1} = u_{n-1}, \quad M_k = u_k + q_k M_{k+1}, \quad k = n-2, n-3, \dots, 1.$$

Po wyznaczeniu momentów rekonstruujemy każdy segment splajnu.

## 7.5 Istnienie i jednoznaczność

**Twierdzenie.** Dla dowolnych danych

$$n \in \mathbb{N}, \quad a = t_0 < t_1 < \dots < t_n = b, \quad y_k = f(t_k),$$

istnieje dokładnie jedna naturalna interpolacyjna funkcja sklejona 3-go stopnia  $s$  spełniająca

$$s''(a) = s''(b) = 0.$$

## 7.6 Zastosowanie w grafice komputerowej

W grafice komputerowej często buduje się krzywą parametryczną

$$\gamma(t) = (x(t), y(t)), \quad t \in [t_0, t_n],$$

gdzie  $x(t)$  i  $y(t)$  są niezależnymi NIFS3 budowanymi dla tych samych parametrów  $t_k$ .

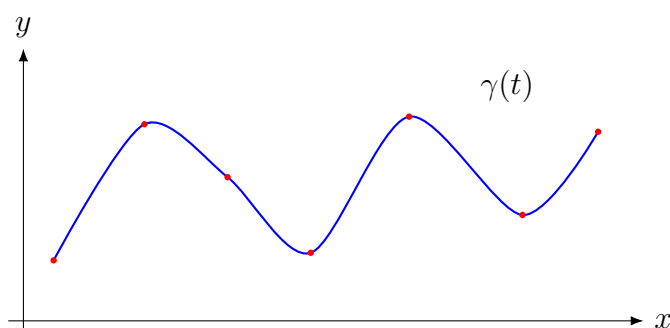
Praktyczny schemat:

- wybieramy punkty kontrolne  $(x_k, y_k)$ ,
- dobieramy parametry  $t_k$  (często prawie równoległe długości łuków),
- interpolujemy osobno:

$$s_x(t_k) = x_k, \quad s_y(t_k) = y_k,$$

- otrzymujemy krzywą

$$\gamma(t) = (s_x(t), s_y(t)).$$



## Wykład 8 Krzywe Béziera i ich zastosowania w grafice komputerowej

### Powtórka z NIFS3.

- Dla danych  $(x_k, y_k)$ ,  $k = 0, \dots, n$ , istnieje dokładnie jedna naturalna funkcja sklejana 3-go stopnia.
- Układ dla momentów jest trójprzekątniowy i można go rozwiązać w czasie  $O(n)$ .
- Splajny są praktyczne w grafice, ale dziś przechodzimy do krzywych Béziera.

### 8.1 Oznaczenia i kombinacje wypukłe

- $\mathbb{R}^d$  – przestrzeń wektorów  $d$ -wymiarowych.
- $E^d$  – punkty w przestrzeni  $d$ -wymiarowej (np.  $E^2$  na płaszczyźnie).

Jeśli  $W_i \in E^d$ ,  $\alpha_i \in \mathbb{R}$  i

$$\sum_{i=0}^n \alpha_i = 1,$$

to punkt

$$\sum_{i=0}^n \alpha_i W_i \in E^d$$

nazywamy kombinacją barycentryczną punktów  $W_i$ .

Jeśli dodatkowo  $\alpha_i \geq 0$ , to jest to kombinacja wypukła i punkt należy do otoczki wypukłej zbioru  $\{W_0, \dots, W_n\}$ .

### 8.2 Wielomiany Bernsteina

Dla  $n \in \mathbb{N}$  i  $k = 0, \dots, n$ :

$$B_k^n(t) := \binom{n}{k} t^k (1-t)^{n-k}.$$

#### Własności podstawowe.

- $B_k^n(t) \geq 0$  dla  $t \in [0, 1]$ .
- $\sum_{k=0}^n B_k^n(t) = 1$ .

- Rekurencja:

$$B_k^n(t) = (1-t)B_k^{n-1}(t) + tB_{k-1}^{n-1}(t).$$

- Pochodna:

$$(B_k^n)'(t) = n(B_k^{n-1}(t) - B_{k-1}^{n-1}(t)).$$

Wielomiany  $B_0^n, \dots, B_n^n$  tworzą bazę przestrzeni  $\Pi_n$ :

$$\text{lin}\{B_0^n, \dots, B_n^n\} = \Pi_n.$$

### 8.3 Krzywa Béziera

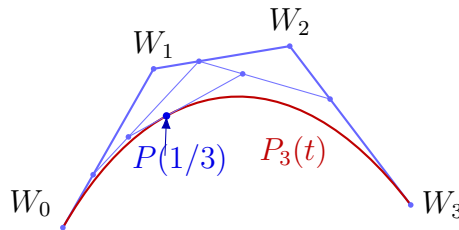
Niech punkty kontrolne  $W_0, \dots, W_n \in E^2$ . Krzywa Béziera stopnia  $n$  jest dana wzorem

$$P_n(t) = \sum_{k=0}^n B_k^n(t) W_k, \quad t \in [0, 1].$$

**Własności.**

- $P_n(0) = W_0, P_n(1) = W_n$ .
- Dla  $t \in [0, 1]$  punkt  $P_n(t)$  leży w otoczce wypukłej punktów kontrolnych.
- 

$$P_n'(0) = n(W_1 - W_0), \quad P_n'(1) = n(W_n - W_{n-1}).$$



### 8.4 Algorytm de Casteljau

Dla ustalonego  $t \in [0, 1]$  definiujemy:

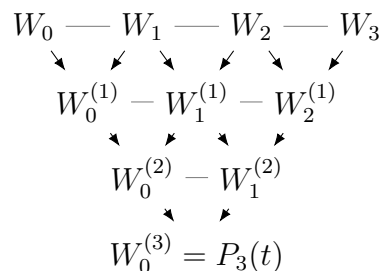
$$W_k^{(0)} := W_k, \quad k = 0, \dots, n,$$

$$W_k^{(i)} := (1-t)W_k^{(i-1)} + tW_{k+1}^{(i-1)}, \quad i = 1, \dots, n, \quad k = 0, \dots, n-i.$$

Wtedy

$$P_n(t) = W_0^{(n)}.$$

Koszt obliczeń punktu  $P_n(t)$ :  $O(n^2)$ .



## 8.5 Uwagi praktyczne

- Krzywe Béziera są podstawowym narzędziem modelowania krzywych w grafice komputerowej.
- Punkty kontrolne sterują kształtem krzywej w intuicyjny sposób.
- Dla złożonych kształtów używa się łączenia wielu segmentów Béziera albo baz B-sklejanych.

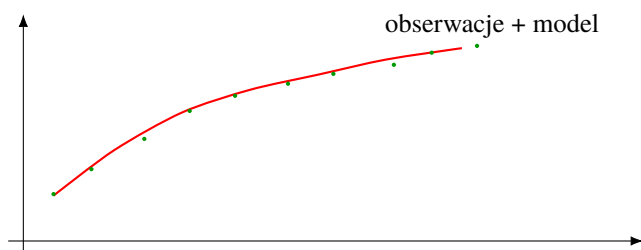
## Wykład 9 Aproksymacja średniokwadratowa na zbiorze dyskretnym

### Powtórka.

- punkty  $(E^d)$  i wektory  $(\mathbb{R}^d)$ ,
- kombinacje barycentryczne i wypukłe,
- wielomiany Bernsteina oraz krzywe Béziera,
- algorytm de Casteljau ( $O(n^2)$ ).

### 9.1 Idea aproksymacji

Mamy obserwacje (“chmurę” punktów) i chcemy dobrać funkcję z ustalonego modelu tak, aby była do nich możliwie bliska.



### 9.2 Dyskretna norma średniokwadratowa

Niech ustalone będą liczby  $x_0 < x_1 < \dots < x_N$  i funkcja  $f$  określona w tych punktach. Definiujemy dyskretną normę:

$$\|f\|_2 := \sqrt{\sum_{k=0}^N (f(x_k))^2}.$$

Odległość funkcji:

$$\|f - g\|_2 = \sqrt{\sum_{k=0}^N (f(x_k) - g(x_k))^2}.$$



**Uwaga (motywacja).** W praktyce nie definiujemy “najlepszego dopasowania” przez normę jednostajną

$$\max_{x \in [a, b]} |f(x) - g(x)|,$$

bo prowadzi to zwykle do trudnego numerycznie zadania minimaxowego. W tej części używamy normy dyskretnej średniokwadratowej.

**Własności normy  $\|\cdot\|_2$ :**

- $\|f\|_2 \geq 0$ , a  $\|f\|_2 = 0 \Leftrightarrow f(x_k) = 0$  dla wszystkich  $k$ ,
- $\|\alpha f\|_2 = |\alpha| \|f\|_2$ ,
- $\|f + g\|_2 \leq \|f\|_2 + \|g\|_2$ .

## 9.3 Zadanie aproksymacji

Dla danej funkcji  $f$  (określonej w  $x_0, \dots, x_N$ ) i ustalonego modelu  $\mathcal{X}$  szukamy elementu optymalnego  $w^* \in \mathcal{X}$ :

$$\|f - w^*\|_2 = \min_{w \in \mathcal{X}} \|f - w\|_2 = \min_{w \in \mathcal{X}} \sqrt{\sum_{k=0}^N (f(x_k) - w(x_k))^2}.$$

## 9.4 Model jednoparametrowy: funkcje stałe

Niech

$$\mathcal{X} = \{a : a \in \mathbb{R}\} = \Pi_0, \quad w(x) = a.$$

Minimalizujemy

$$E(a) := \sum_{k=0}^N (f(x_k) - a)^2.$$

Warunek konieczny:

$$E'(a) = 0 \quad \implies \quad a^* = \frac{1}{N+1} \sum_{k=0}^N f(x_k) = \frac{1}{N+1} \sum_{k=0}^N y_k.$$

Zatem element optymalny:

$$w^*(x) = a^*.$$

## 9.5 Model jednoparametrowy: $\{ax^2\}$

Niech

$$\mathcal{X} = \{ax^2 : a \in \mathbb{R}\}.$$

Minimalizujemy

$$E(a) = \sum_{k=0}^N (f(x_k) - ax_k^2)^2.$$

Z warunku  $E'(a) = 0$ :

$$a^* = \frac{\sum_{k=0}^N f(x_k) x_k^2}{\sum_{k=0}^N x_k^4}.$$

Wtedy

$$w^*(x) = a^* x^2.$$

## 9.6 Model jednoparametrowy: $\{ae^x\}$

Niech

$$\mathcal{X} = \{ae^x : a \in \mathbb{R}\}.$$

Minimalizujemy

$$E(a) = \sum_{k=0}^N (f(x_k) - ae^{x_k})^2.$$

Warunek  $E'(a) = 0$  daje:

$$a^* = \frac{\sum_{k=0}^N f(x_k) e^{x_k}}{\sum_{k=0}^N e^{2x_k}}.$$

Stąd

$$w^*(x) = a^* e^x.$$

## 9.7 Model dwuparametrowy: prosta regresji

Niech

$$\mathcal{X} = \{ax + b : a, b \in \mathbb{R}\} = \Pi_1.$$

Minimalizujemy

$$E(a, b) = \sum_{k=0}^N (f(x_k) - ax_k - b)^2.$$

Warunki stacjonarności:

$$\frac{\partial E}{\partial a} = 0, \quad \frac{\partial E}{\partial b} = 0.$$

Dają układ:

$$\begin{cases} a \sum x_k^2 + b \sum x_k = \sum x_k f(x_k), \\ a \sum x_k + b(N+1) = \sum f(x_k). \end{cases}$$

Oznaczając

$$s_1 := \sum_{k=0}^N x_k, \quad s_2 := \sum_{k=0}^N x_k^2, \quad s_3 := \sum_{k=0}^N f(x_k), \quad s_4 := \sum_{k=0}^N x_k f(x_k),$$

otrzymujemy

$$a^* = \frac{(N+1)s_4 - s_1 s_3}{(N+1)s_2 - s_1^2}, \quad b^* = \frac{s_2 s_3 - s_1 s_4}{(N+1)s_2 - s_1^2}.$$

Wtedy

$$w^*(x) = a^* x + b^*,$$

czyli klasyczna prosta regresji liniowej.

## 9.8 Model ogólny

Niech

$$\mathcal{X} = \{a_0 g_0(x) + a_1 g_1(x) + \cdots + a_m g_m(x) : a_0, \dots, a_m \in \mathbb{R}\},$$

gdzie  $g_0, \dots, g_m$  są ustalone (typowo  $m + 1 \leq N + 1$ ).

Szukamy minimum funkcji błędu

$$E(a_0, \dots, a_m) = \sum_{k=0}^N \left( f(x_k) - \sum_{i=0}^m a_i g_i(x_k) \right)^2.$$

Otrzymujemy układ równań normalnych:

$$\frac{\partial E}{\partial a_i} = 0, \quad i = 0, 1, \dots, m.$$

Rozwiązanie tego układu daje parametry optymalne i funkcję  $w^* \in \mathcal{X}$ .

## 9.9 Pochodne cząstkowe – uwaga techniczna

Gdy model zależy od wielu parametrów, minimalizujemy funkcję błędu wielu zmiennych. Wtedy używamy pochodnych cząstkowych.

**Przykład.** Dla

$$F(x, y, z) = x^2 y + z \cos(x) - 4e^{2y+xz}$$

mamy np.

$$\frac{\partial F}{\partial x} = 2xy - z \sin(x) - 4e^{2y+xz} z, \quad \frac{\partial F}{\partial z} = \cos(x) - 4e^{2y+xz} x.$$

**Fakt.** Jeśli funkcja wielu zmiennych ma ekstremum wewnętrzne, to jej wszystkie pochodne cząstkowe pierwszego rzędu zerują się w tym punkcie.

## Wykład 10 Wielomianowa aproksymacja średniokwadratowa na zbiorze dyskretnym

### Powtórka (Wykład 9).

- Dla danych  $x_0 < \dots < x_N$  używamy normy

$$\|f\|_2 = \sqrt{\sum_{k=0}^N (f(x_k))^2}.$$

- Szukamy elementu optymalnego

$$w^* = \arg \min_{w \in \mathcal{X}} \|f - w\|_2.$$

- Dla modeli jednoparametrowych i liniowych (regresja) dostajemy wzory jawne po wyzerowaniu pochodnych funkcji błędu.

## 10.1 Zadanie wykładu 10

Zakładamy model wielomianowy:

$$\mathcal{X} := \Pi_m, \quad m \leq N.$$

Dla danych  $(x_k, y_k)$ , gdzie  $y_k = f(x_k)$ , szukamy wielomianu

$$w_m^* \in \Pi_m$$

takiego, że

$$\|f - w_m^*\|_2 = \min_{w \in \Pi_m} \|f - w\|_2.$$

**Uwaga praktyczna.** Rozwiązanie zadania w bazie potęgowej bywa numerycznie niestabilne i kosztowne. Lepsza droga: baza ortogonalna.

## 10.2 Dyskretny iloczyn skalarny

Definiujemy

$$(f, g)_N := \sum_{k=0}^N f(x_k)g(x_k), \quad (\cdot, \cdot)_N : F \times F \rightarrow \mathbb{R}.$$

Własności:

- $(f, f)_N \geq 0$ , a  $(f, f)_N = 0 \Leftrightarrow f(x_k) = 0 \forall k$ ,
- $(f, g)_N = (g, f)_N$ ,
- $(f + g, h)_N = (f, h)_N + (g, h)_N$ ,
- $(\alpha f, g)_N = \alpha(f, g)_N$ .

Związek z normą:

$$\|f\|_2 = \sqrt{(f, f)_N}.$$

Funkcje  $f, g$  są ortogonalne względem  $(\cdot, \cdot)_N$ , gdy

$$(f, g)_N = 0.$$

### 10.3 Układ i ciąg ortogonalny

Układ funkcji  $f_0, \dots, f_m$  nazywamy ortogonalnym względem  $(\cdot, \cdot)_N$ , jeśli

$$(f_i, f_j)_N = 0 \quad (i \neq j), \quad (f_i, f_i)_N > 0.$$

Szczególny przypadek: ciąg wielomianów ortogonalnych  $P_0, \dots, P_m$ ,

$$P_k \in \Pi_k \setminus \Pi_{k-1}, \quad (P_i, P_j)_N = 0 \quad (i \neq j), \quad (P_i, P_i)_N > 0.$$

Można go dostać np. ortogonalizacją Grama–Schmidta, ale ta droga jest zwykle za droga numerycznie.

### 10.4 Rekurencja trójczłonowa

Dla ustalonego  $(\cdot, \cdot)_N$  ciąg wielomianów ortogonalnych spełnia rekurencję:

$$P_0(x) = 1, \quad P_1(x) = x - c_1,$$

$$P_k(x) = (x - c_k)P_{k-1}(x) - d_k P_{k-2}(x), \quad k = 2, \dots, m,$$

gdzie

$$c_k = \frac{(xP_{k-1}, P_{k-1})_N}{(P_{k-1}, P_{k-1})_N}, \quad d_k = \frac{(P_{k-1}, P_{k-1})_N}{(P_{k-2}, P_{k-2})_N}.$$

Koszt konstrukcji: liniowy względem  $m$  (poza kosztami obliczania iloczynów skalar-nych).

### 10.5 Postać rozwiązania optymalnego

**Twierdzenie.** Wielomian optymalny ma postać

$$w_m^*(x) = \sum_{k=0}^m a_k P_k(x),$$

gdzie

$$a_k = \frac{(f, P_k)_N}{(P_k, P_k)_N}, \quad k = 0, \dots, m.$$

To odpowiednik rozwinięcia Fouriera w bazie ortogonalnej.

## 10.6 Obliczanie wartości wielomianu

Jeśli wielomiany bazowe spełniają rekurencję ogólną

$$\begin{aligned} Q_0(x) &= \alpha_0, & Q_1(x) &= (\alpha_1 x - \beta_1) Q_0(x), \\ Q_k(x) &= (\alpha_k x - \beta_k) Q_{k-1}(x) - \gamma_{k-2} Q_{k-2}(x), & k &\geq 2, \end{aligned}$$

to dla

$$q_m(x) = \sum_{k=0}^m a_k Q_k(x)$$

wartość dla ustalonego  $x$  obliczamy algorytmem Clenshawa.

Wersja praktyczna (backward):

$$B_{m+2} = B_{m+1} = 0, \quad B_k = a_k + (\alpha_{k+1} x - \beta_{k+1}) B_{k+1} - \gamma_{k+1} B_{k+2},$$

dla  $k = m, m-1, \dots, 0$ , a następnie

$$q_m(x) = \alpha_0 B_0.$$

Koszt:  $O(m)$  dla jednego punktu.

## 10.7 Podsumowanie

- Wielomianową aproksymację średniokwadratową warto realizować w bazie ortogonalnej.
- Współczynniki optymalne liczymy projekcyjnie:

$$a_k = \frac{(f, P_k)_N}{(P_k, P_k)_N}.$$

- Rekurencja trójczłonowa i Clenshaw dają efektywny koszt obliczeń.