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 $x \neq 0$:

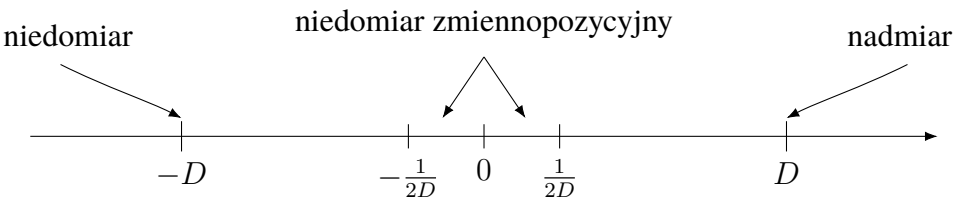
$$x \approx \text{chop}(x) := s \cdot m_t^c \cdot 2^c \qquad \text{albo} \qquad x \approx \text{rd}(x) := s \cdot m_t^r \cdot 2^c$$

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:

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

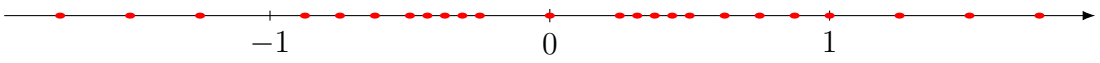
a cecha:

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

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

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

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

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

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

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.

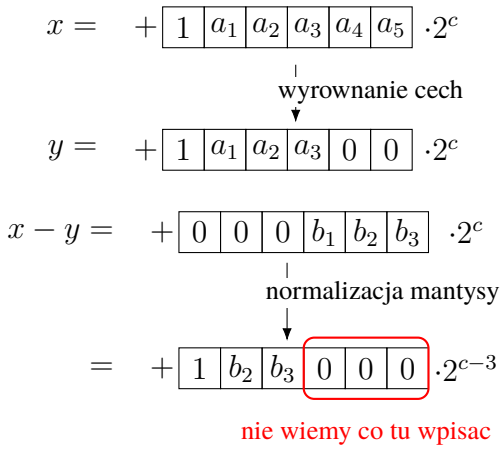
2.3 Zjawisko utraty cyfr znaczących

Problem utraty cyfr znaczących prześledźmy na przykładzie.
Niech

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

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.



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

3.1 Przykłady wrażliwości zadań

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 ε (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.

3.2 Uwarunkowanie i wskaźniki

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

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

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 $i = 1, \ldots, n$, 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.

3.3 Algorytmy numerycznie poprawne

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

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

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:

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

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

Wykład 4 Rozwiązywanie równań nieliniowych

Zadanie: dla danej funkcji $f : \mathbb{R} \to \mathbb{R}$ 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 $k = 0, 1, 2, \ldots,$
- środek przedziału $m_{k+1} := \dfrac{a_k + b_k}{2}$,
- 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:

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

- Metoda bisekcji jest zbieżna i praktycznie niezawodna (dla spełnionych założeń).
- Aby otrzymać przedział długości co najwyżej 2ε, 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:

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

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:

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

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

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

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

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

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

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

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

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:

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

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:

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

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

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

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

gdzie

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

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).

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

Wtedy $w(x) = w_0$.

c) Postać Czebyszewa. Wielomiany Czebyszewa definiujemy rekurencyjnie:

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

Własności (podstawowe).

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

Postać Czebyszewa wielomianu:

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

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 $O(n)$):

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

Wtedy

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

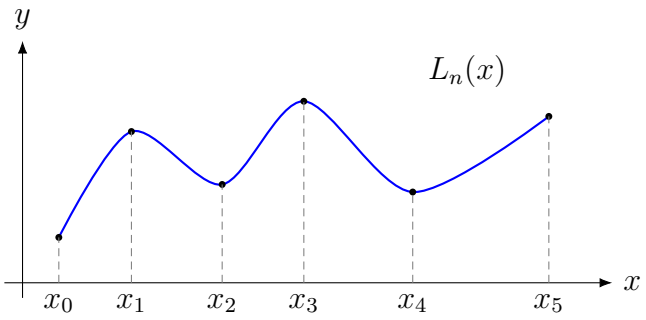
5.2 Interpolacja wielomianowa

Mamy dane pomiary:

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

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

$$L_n(x_k) = y_k, \qquad k = 0, 1, \ldots, n.$$



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

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

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:

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

Otrzymane przybliżenie:

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

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

Przykłady

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

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

  i porównujemy z interpolantem $L_n(x)$.

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

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

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

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

$$x_k = -2 + \frac{4k}{n}, \qquad 0 \le k \le n, \qquad n = 1, \ldots, 6.$$

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

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

Wykład 6 Interpolacja wielomianowa – część II

Plan wykładu:

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

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

$$L_n(x_k) = y_k, \qquad 0 \le k \le n.$$

W postaci Lagrange'a:

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

6.1 Postać Newtona wielomianu interpolacyjnego

Niech

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

Szukamy

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

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

Wzór jawny:

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

6.2 Ilorazy różnicowe

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

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

oraz rekurencyjnie

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

Przykład (rząd 2):

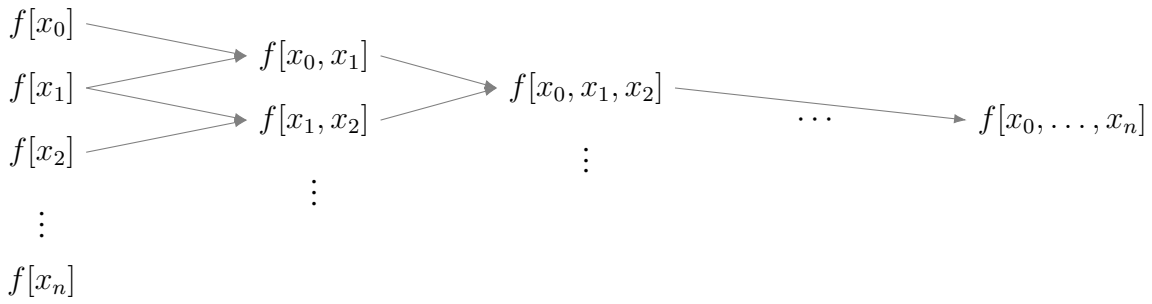
$$f[x_0, x_1, x_2] = \frac{f[x_1, x_2] - f[x_0, x_1]}{x_2 - x_0}.$$

Twierdzenie (postać Newtona przez ilorazy różnicowe).

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

czyli

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



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 $z_0, \ldots, z_M$:

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

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:

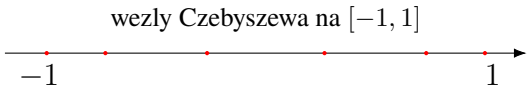
$$\max_{x\in[a,b]} |f(x) - L_n(x)| \leq \max_{x\in[a,b]} \left| \frac{f^{(n+1)}(x)}{(n+1)!} \right| \cdot \max_{x\in[a,b]} \left| \prod_{k=0}^{n} (x - x_k) \right|.$$

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:

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



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

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

Wykład 7 Naturalne interpolacyjne funkcje sklejane 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.

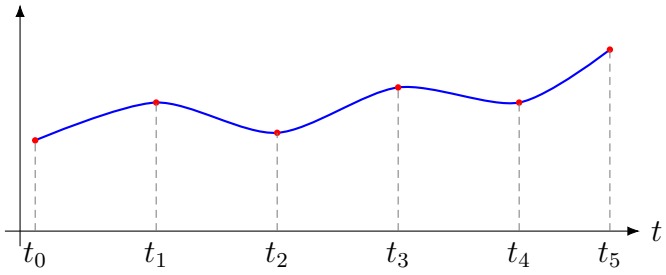
7.1 Definicja NIFS3

Niech

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

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

- $s(t_k) = y_k$ dla $k = 0, \ldots, n$,
- 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 klamrowa 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: 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 klamrowych jest mało efektywne dla dużych n.

7.3 Momenty i układ trójprzekątniowy

Wprowadzamy momenty:

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

Niech

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

Momenty spełniają układ:

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

Jest to układ liniowy trójprzekątniowy, 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)

Obliczamy pomocnicze wielkości rekurencyjnie:

$$q_0 := 0, \qquad u_0 := 0,$$

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

Następnie:

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

Po wyznaczeniu momentów rekonstruujemy każdy segment splajnu.

7.5 Istnienie i jednoznaczność

Twierdzenie. Dla dowolnych danych

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

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

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

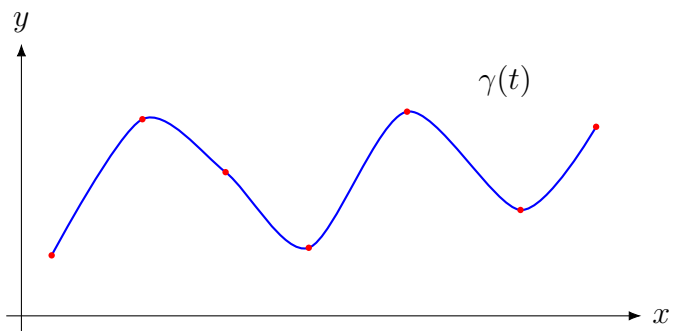
gdzie $x(t)$ i $y(t)$ są niezależnymi NIFS3 budowanymi dla tych samych parametrów t_k.
Praktyczny schemat:

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

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

- otrzymujemy krzywą

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



Wykład 8 Krzywe Béziera i ich zastosowania w grafice komputerowej

Powtórka z NIFS3.

- Dla danych (x_k, y_k), $k = 0, \ldots, n$, 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 $\{W_0, \ldots, W_n\}$.

8.2 Wielomiany Bernsteina

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

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

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

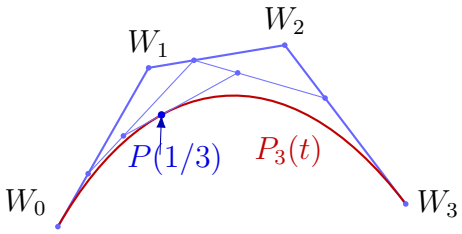
Wielomiany $B_0^n, \ldots, B_n^n$ tworzą bazę przestrzeni Π_n:
$$\operatorname{lin}\{B_0^n, \ldots, B_n^n\} = \Pi_n.$$

8.3 Krzywa Béziera

Niech punkty kontrolne $W_0, \ldots, W_n \in E^2$. Krzywa Béziera stopnia n jest dana wzorem
$$P_n(t) = \sum_{k=0}^{n} B_k^n(t)\, W_k, \qquad t \in [0,1].$$

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

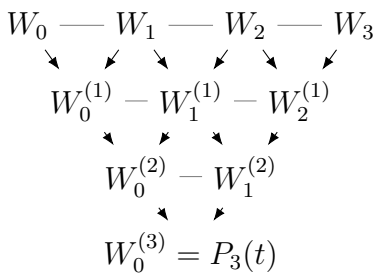


8.4 Algorytm de Casteljau

Dla ustalonego $t \in [0,1]$ definiujemy:
$$W_k^{(0)} := W_k, \qquad k = 0, \ldots, n,$$
$$W_k^{(i)} := (1-t)W_k^{(i-1)} + tW_{k+1}^{(i-1)}, \qquad i = 1, \ldots, n, \quad k = 0, \ldots, n-i.$$

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.

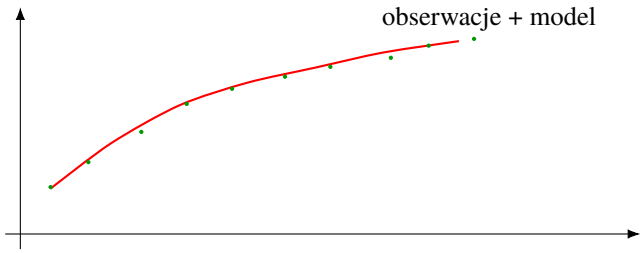
Wykład 9 Aproksymacja średniokwadratowa na zbiorze dyskretnym

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 $x_0 < x_1 < \cdots < x_N$ i funkcja f określona w tych punktach. Definiujemy dyskretną normę:

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

Odległość funkcji:

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

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 minimaksowego. 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 $x_0, \ldots, x_N$) i ustalonego modelu $\mathcal{X}$ szukamy elementu optymalnego $w^* \in \mathcal{X}$:

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

9.4 Model jednoparametrowy: funkcje stałe

Niech

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

Minimalizujemy

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

Warunek konieczny:

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

Zatem element optymalny:

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

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

Niech

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

Minimalizujemy

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

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

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

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

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

Warunki stacjonarności:

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

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

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

Wtedy

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

czyli klasyczna prosta regresji liniowej.

9.8 Model ogólny

Niech

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

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

Szukamy minimum funkcji błędu

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

Otrzymujemy układ równań normalnych:

$$\frac{\partial E}{\partial a_i} = 0, \qquad i = 0, 1, \ldots, m.$$

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.

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

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 $x_0 < \cdots < x_N$ używamy normy

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

- 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:

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

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. Rozwiązywanie zadania w bazie potęgowej bywa numerycznie niestabilne i kosztowne. Lepsza droga: baza ortogonalna.

10.2 Dyskretny iloczyn skalarny

Definiujemy

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

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 $f_0, \ldots, f_m$ nazywamy ortogonalnym względem $(\cdot,\cdot)_N$, jeśli

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

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

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

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:

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

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

gdzie

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

Koszt konstrukcji: liniowy względem m (poza kosztami obliczania iloczynów skalarnych).

10.5 Postać rozwiązania optymalnego

Twierdzenie. Wielomian optymalny ma postać

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

gdzie

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

To odpowiednik rozwinięcia Fouriera w bazie ortogonalnej.

10.6 Obliczanie wartości wielomianu

Jeśli wielomiany bazowe spełniają rekurencję ogólną

$$Q_0(x) = \alpha_0, \qquad Q_1(x) = (\alpha_1 x - \beta_1)Q_0(x),$$

$$Q_k(x) = (\alpha_k x - \beta_k)Q_{k-1}(x) - \gamma_{k-2}Q_{k-2}(x), \qquad k \geq 2,$$

to dla

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

wartość dla ustalonego x obliczamy algorytmem Clenshawa.

Wersja praktyczna (backward):

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

dla $k = m, m-1, \ldots, 0$, 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.

Wykład 11 Kwadratury, czyli całkowanie numeryczne

Powtórka (z wykładu 10).

- Szukamy $w_n^* \in \Pi_n$ minimalizującego błąd w normie dyskretnej:
$$\|f - w_n^*\|_2 = \min_{w \in \Pi_n} \|f - w\|_2.$$
- W bazie ortogonalnej $\{P_k\}$ mamy postać
$$w_n^*(x) = \sum_{k=0}^{n} a_k P_k(x), \qquad a_k = \frac{(f, P_k)_N}{(P_k, P_k)_N}.$$
- Współczynniki wyznaczamy projekcyjnie, jak w dyskretnym rozwinięciu Fouriera.

11.1 Całki nieoznaczone i oznaczone – przypomnienie

Dla pochodnej odwrotnej (funkcji pierwotnej):

$$\int f(x)\,dx = F(x) + C, \qquad F'(x) = f(x).$$

Przykłady:

$$\int x^n\,dx = \frac{x^{n+1}}{n+1} + C \quad (n \neq -1), \qquad \int \frac{dx}{x} = \ln|x| + C,$$

$$\int e^x\,dx = e^x + C, \qquad \int \cos x\,dx = \sin x + C, \qquad \int \frac{dx}{x^2+1} = \arctan x + C.$$

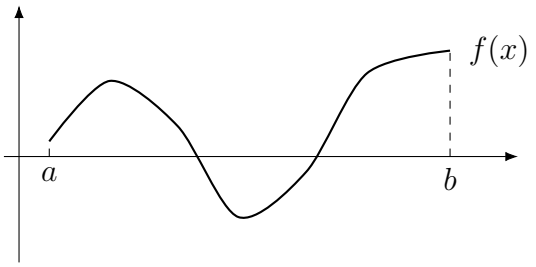
Nie każda całka ma postać elementarną, np.

$$\int e^{-x^2}\,dx, \qquad \int \frac{\sin x}{x}\,dx.$$

Całka oznaczona:

$$\int_a^b f(x)\,dx = F(b) - F(a), \qquad F' = f,$$

co interpretujemy jako pole obszaru zorientowane pomiędzy wykresem i osią OX.



11.2 Idea całkowania numerycznego

Jeśli f jest trudna do scałkowania analitycznie, szukamy funkcji g dobrze przybliżającej f na $[a, b]$, takiej że

$$\int_a^b f(x)\,dx \approx \int_a^b g(x)\,dx,$$

a prawa strona daje się policzyć jawnie.

11.3 Kwadratury liniowe

Definicja. Kwadraturą liniową nazywamy wyrażenie

$$Q_n(f) := \sum_{k=0}^{n} A_k^{(n)}\, f\big(x_k^{(n)}\big),$$

gdzie $x_0^{(n)}, \ldots, x_n^{(n)}$ to węzły, a $A_0^{(n)}, \ldots, A_n^{(n)}$ to współczynniki kwadratury.

Zadanie. Dobrać węzły i współczynniki tak, aby

$$\int_a^b f(x)\,dx \approx Q_n(f)$$

dla możliwie szerokiej klasy funkcji f.

Mamy równanie z resztą:

$$\int_a^b f(x)\,dx = Q_n(f) + R_n(f),$$

gdzie $R_n(f)$ to błąd kwadratury.

11.4 Rząd kwadratury

Definicja. Mówimy, że kwadratura liniowa Q_n ma rząd $r \in \mathbb{N}$, gdy:

- dla każdego $w \in \Pi_{r-1}$ zachodzi

$$\int_a^b w(x)\,dx = Q_n(w),$$

- istnieje $v \in \Pi_r$ takie, że

$$\int_a^b v(x)\,dx \neq Q_n(v).$$

Uwaga. Kwadratury chcemy konstruować tak, aby ich rząd był możliwie wysoki.

Twierdzenie.

$$\operatorname{rzad}(Q_n) \leq 2n + 2.$$

11.5 Kwadratura interpolacyjna

Niech $L_n \in \Pi_n$ będzie wielomianem interpolacyjnym funkcji f w węzłach $x_0, \ldots, x_n$:

$$L_n(x_k) = f(x_k), \qquad k = 0, 1, \ldots, n.$$

W postaci Lagrange'a:

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

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

Po scałkowaniu:

$$\int_a^b L_n(x)\, dx = \sum_{k=0}^{n} \left(\int_a^b \lambda_k(x)\, dx \right) f(x_k) =: \sum_{k=0}^{n} A_k f(x_k) = Q_n(f),$$

czyli

$$A_k = \int_a^b \lambda_k(x)\, dx.$$

To jest właśnie kwadratura interpolacyjna.

Przypomnienie (błąd interpolacji). Jeśli $f \in C^{n+1}[a, b]$, to dla $x \in [a, b]$:

$$f(x) - L_n(x) = \frac{f^{(n+1)}(\eta_x)}{(n+1)!} \prod_{k=0}^{n} (x - x_k), \qquad \eta_x \in (a, b).$$

Stąd po scałkowaniu:

$$\int_a^b f(x)\, dx - \int_a^b L_n(x)\, dx = \frac{1}{(n+1)!} \int_a^b f^{(n+1)}(\eta_x) \prod_{k=0}^{n} (x - x_k)\, dx.$$

11.6 Przykład kwadratury interpolacyjnej

Dla $[a, b] = [0, 1]$, $n = 4$, węzły:

$$x_0 = 0, \quad x_1 = \frac{1}{3}, \quad x_2 = \frac{1}{2}, \quad x_3 = \frac{2}{3}, \quad x_4 = 1.$$

Dla tej siatki:

$$A_0 = A_4 = \frac{11}{120}, \qquad A_1 = A_3 = \frac{27}{40}, \qquad A_2 = -\frac{8}{15}.$$

Zatem

$$Q_4(f) = \frac{11}{120} f(0) + \frac{27}{40} f\left(\frac{1}{3}\right) - \frac{8}{15} f\left(\frac{1}{2}\right) + \frac{27}{40} f\left(\frac{2}{3}\right) + \frac{11}{120} f(1).$$

Dla $f(x) = \sin(\pi x)$:

$$\int_0^1 \sin(\pi x)\,dx = \frac{2}{\pi} \approx 0.6366,$$

$$Q_4(f) = \frac{27\sqrt{3}}{40} - \frac{8}{15} \approx 0.6358.$$

Dla $f(x) = e^{-x^2}$:

$$\int_0^1 e^{-x^2}\,dx \approx 0.7468241, \qquad Q_4(f) \approx 0.746841.$$

11.7 Rząd a interpolacja

Twierdzenie. Kwadratura liniowa ma rząd co najmniej $n+1$ wtedy i tylko wtedy, gdy jest kwadraturą interpolacyjną.

W szczególności dla kwadratur interpolacyjnych:

$$n+1 \le \mathrm{rzad}(Q_n) \le 2n+2.$$

Pytanie naturalne: jak dobrać węzły i współczynniki, by osiągnąć rząd dokładnie $2n+2$? Odpowiedzią są kwadratury Gaussa.

11.8 Kwadratury Newtona–Cotesa

Bierzemy węzły równoodległe:

$$x_k = a + kh, \qquad k = 0, 1, \ldots, n, \qquad h := \frac{b-a}{n}.$$

Dla tych węzłów współczynniki kwadratury interpolacyjnej można zapisać jawnie:

$$A_k = \frac{(-1)^{n-k}}{k!(n-k)!}\, h \int_0^n \prod_{\substack{j=0 \\ j\neq k}}^{n} (t-j)\,dt,$$

a ponadto zachodzi symetria

$$A_k = A_{n-k}.$$

Fakt (rząd kwadratur Newtona–Cotesa).

- dla n parzystych: $\mathrm{rzad}(Q_n^{NC}) = n+2$,
- dla n nieparzystych: $\mathrm{rzad}(Q_n^{NC}) = n+1$.

11.9 Dwa klasyczne wzory NC

Wzór trapezów ($n = 1$):

$$Q_1^{NC}(f) = \frac{b-a}{2}\,[f(a) + f(b)].$$

Reszta ma rząd $O(h^3)$ (dla $h = b - a$).

Wzór Simpsona ($n = 2$):

$$Q_2^{NC}(f) = \frac{b-a}{6}\left[f(a) + 4f\left(\frac{a+b}{2}\right) + f(b)\right],$$

$$R_2^{NC}(f) = \frac{1}{90}h^5 f^{(4)}(\alpha), \qquad \alpha \in (a, b),$$

co daje rząd $O(h^5)$.

Uwaga. Na zdjęciu ze wzorem trapezów współczynnik przy dokładnej postaci reszty jest nieczytelny; zachowałem pewną informację o rzędzie błędu.

Wykład 12 Kwadratury złożone, metoda Romberga, kwadratury Gaussa

Powtórka (z wykładu 11).

- Kwadratura liniowa ma postać

$$Q_n(f) = \sum_{k=0}^{n} A_k^{(n)} f\big(x_k^{(n)}\big).$$

- Rząd kwadratury:

$$\operatorname{rzad}(Q_n) = r \iff \begin{cases} Q_n(w) = \displaystyle\int_a^b w(x)\,dx & \forall w \in \Pi_{r-1}, \\ \exists v \in \Pi_r : \; Q_n(v) \neq \displaystyle\int_a^b v(x)\,dx. \end{cases}$$

- Dla kwadratur interpolacyjnych:

$$n+1 \leq \operatorname{rzad}(Q_n) \leq 2n+2.$$

12.1 Kwadratura interpolacyjna – postać współczynników

Dla zadanych węzłów $x_0, \ldots, x_n$ i bazy Lagrange'a λ_k:

$$Q_n(f) = \sum_{k=0}^{n} A_k f(x_k), \qquad A_k = \int_a^b \lambda_k(x)\,dx,$$

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

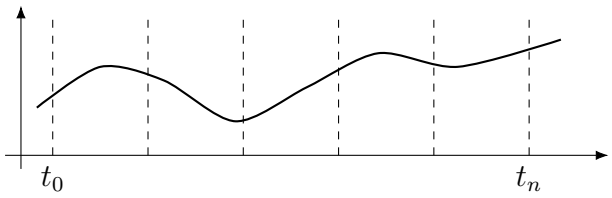
Dla węzłów równoodległych otrzymujemy kwadratury Newtona–Cotesa.

12.2 Kwadratury złożone – idea

Dzielimy przedział całkowania na podprzedziały:

$$a = t_0 < t_1 < \cdots < t_n = b, \qquad \int_a^b f(x)\,dx = \sum_{k=0}^{n-1} \int_{t_k}^{t_{k+1}} f(x)\,dx.$$

Na każdym małym przedziale stosujemy prostą kwadraturę (np. trapezy lub Simpsona), a następnie sumujemy.



12.3 Złożony wzór trapezów

Niech siatka będzie równomierna:

$$t_k = a + kh_n, \qquad h_n := \frac{b-a}{n}, \qquad k = 0, 1, \ldots, n.$$

Dla pojedynczego przedziału $[t_k, t_{k+1}]$:

$$\int_{t_k}^{t_{k+1}} f(x)\,dx = \frac{h_n}{2}\big(f(t_k) + f(t_{k+1})\big) - \frac{h_n^3}{12} f''(\eta_k), \qquad \eta_k \in (t_k, t_{k+1}).$$

Po zsumowaniu:

$$\int_a^b f(x)\,dx = T_n(f) + R_n^T(f),$$

$$T_n(f) := h_n \left[\frac{f(a)+f(b)}{2} + \sum_{k=1}^{n-1} f(t_k) \right],$$

$$R_n^T(f) = -\frac{b-a}{12} h_n^2 f''(\xi), \qquad \xi \in (a,b), \quad f \in C^2[a,b].$$

Twierdzenie. Jeśli $f \in C[a,b]$, to

$$\lim_{n\to\infty} T_n(f) = \int_a^b f(x)\,dx.$$

12.4 Złożony wzór Simpsona

Dzielimy $[a,b]$ na parzystą liczbę podprzedziałów:

$$n = 2m, \qquad h_n = \frac{b-a}{n}.$$

Dla każdej pary przedziałów $[t_{2k}, t_{2k+2}]$:

$$\int_{t_{2k}}^{t_{2k+2}} f(x)\,dx = \frac{h_n}{3}\Big(f(t_{2k}) + 4f(t_{2k+1}) + f(t_{2k+2})\Big) + \frac{h_n^5}{90} f^{(4)}(\alpha_k),$$

$$\alpha_k \in (t_{2k}, t_{2k+2}), \qquad f \in C^4[t_{2k}, t_{2k+2}].$$

Po zsumowaniu:

$$\int_a^b f(x)\,dx = S_n(f) + R_n^S(f),$$

$$S_n(f) := \frac{h_n}{3} \left[f(t_0) + 2\sum_{k=1}^{m-1} f(t_{2k}) + 4\sum_{k=1}^{m} f(t_{2k-1}) + f(t_{2m}) \right],$$

$$R_n^S(f) = \frac{a-b}{180} h_n^4 f^{(4)}(\alpha), \qquad \alpha \in (a,b), \quad f \in C^4[a,b].$$

Wniosek (rząd zbieżności).

$$R_n^T(f) = O(n^{-2}), \qquad R_n^S(f) = O(n^{-4}).$$

12.5 Metoda Romberga

Bierzemy kolejne zagęszczenia siatki:

$$n = 2^k, \qquad h_k := \frac{b-a}{2^k}, \qquad x_i^{(k)} = a + ih_k \ (i = 0, 1, \ldots, 2^k).$$

Pierwszy wiersz tablicy Romberga to wartości złożonego trapezu:

$$T_{0k} := T_{2^k}(f) = h_k \left[\frac{f(a)+f(b)}{2} + \sum_{i=1}^{2^k-1} f\big(x_i^{(k)}\big) \right].$$

Następnie wykonujemy ekstrapolację Richardsona:

$$T_{mk} := \frac{4^m T_{m-1,k+1} - T_{m-1,k}}{4^m - 1}, \qquad k = 0, 1, \ldots, \quad m = 1, 2, \ldots$$

Własności (z tablicy Romberga):

- $T_{mk} = \int_a^b f(x)\,dx + O\big(h_k^{2m+2}\big)$,
- każde T_{mk} jest kwadraturą liniową,
- ciąg na przekątnej $T_{00}, T_{11}, T_{22}, \ldots$ daje coraz wyższe rzędy.

12.6 Informacja o kwadraturach Gaussa

Problem. Dla kwadratury

$$Q_n(f) = \sum_{k=0}^{n} A_k^{(n)} f\big(x_k^{(n)}\big)$$

chcemy dobrać węzły i współczynniki tak, aby

$$\mathrm{rzad}(Q_n) = 2n + 2$$

(maksymalny możliwy).

Taka kwadratura musi być interpolacyjna, więc

$$A_k^{(n)} = \int_a^b \prod_{\substack{i=0 \\ i \neq k}}^{n} \frac{x - x_i^{(n)}}{x_k^{(n)} - x_i^{(n)}}\,dx.$$

Kluczowy problem to dobór węzłów $x_k^{(n)}$.

Twierdzenie (Gauss–Legendre, przypadek $[a, b] = [-1, 1]$). Węzły kwadratury rzędu $2n + 2$ są miejscami zerowymi wielomianu Legendre'a P_{n+1}.

Wielomiany Legendre'a spełniają rekurencję:

$$P_0(x) = 1, \qquad P_1(x) = x,$$

$$P_k(x) = \frac{2k-1}{k} x P_{k-1}(x) - \frac{k-1}{k} P_{k-2}(x), \qquad k = 2, 3, \ldots$$

Uwagi.

- Nie ma prostych wzorów jawnych na zera P_{n+1}.
- W praktyce zera i współczynniki są tablicowane lub liczone numerycznie z dużą dokładnością.
- Dzięki temu kwadratury Gaussa można stabilnie liczyć także dla dużych n.

Wykład 13 Podstawowe algorytmy numeryczne algebry liniowej

Powtórka (z wykładu 12).

- Złożony wzór trapezów: błąd rzędu $O(h^2) = O(n^{-2})$.
- Złożony wzór Simpsona: błąd rzędu $O(h^4) = O(n^{-4})$.
- Metoda Romberga: ekstrapolacja Richardsona dla kolejnych przybliżeń trapezowych.
- Kwadratury Gaussa: maksymalny rząd $2n + 2$ dla $n + 1$ węzłów.

13.1 Macierze – przypomnienie

Macierz rzeczywista $m \times n$:

$$A = [a_{ij}] \in \mathbb{R}^{m\times n}, \qquad b = \begin{bmatrix} b_1 \\ \vdots \\ b_m \end{bmatrix} \in \mathbb{R}^m.$$

Dla macierzy tego samego rozmiaru:

$$A \pm B = [a_{ij} \pm b_{ij}], \qquad \lambda A = [\lambda a_{ij}],\ \lambda \in \mathbb{R}, \qquad A^T = [a_{ji}].$$

Macierz jednostkowa:

$$I_n = \operatorname{diag}(1, \ldots, 1) \in \mathbb{R}^{n\times n}.$$

13.2 Mnożenie macierzy i macierz odwrotna

Jeśli

$$A = [a_{ij}] \in \mathbb{R}^{m\times k}, \qquad B = [b_{ij}] \in \mathbb{R}^{k\times n},$$

to

$$AB = [c_{ij}] \in \mathbb{R}^{m\times n}, \qquad c_{ij} = \sum_{\ell=1}^{k} a_{i\ell} b_{\ell j}.$$

Koszt klasycznego mnożenia: $O(n^3)$ (dla macierzy kwadratowych). Mnożenie macierzy nie jest przemienne.

Dla macierzy kwadratowej $A \in \mathbb{R}^{n\times n}$:

$$A^{-1} \text{ spełnia } AA^{-1} = A^{-1}A = I_n.$$

13.3 Wyznacznik i odwracalność

Wyznacznik:

$$\det : \mathbb{R}^{n\times n} \to \mathbb{R}.$$

Własności:

$$\det(AB) = \det(A)\det(B), \qquad \det(A^T) = \det(A).$$

Twierdzenie. Macierz A jest odwracalna wtedy i tylko wtedy, gdy

$$\det(A) \neq 0.$$

13.4 Układ równań liniowych

Układ

$$\begin{cases} a_{11}x_1 + \cdots + a_{1n}x_n = b_1, \\ \vdots \\ a_{n1}x_1 + \cdots + a_{nn}x_n = b_n \end{cases} \qquad \Longleftrightarrow \qquad Ax = b,$$

z

$$A \in \mathbb{R}^{n\times n}, \quad x \in \mathbb{R}^n, \quad b \in \mathbb{R}^n.$$

Fakt. Układ $Ax = b$ ma dokładnie jedno rozwiązanie wtedy i tylko wtedy, gdy $\det(A) \neq 0$.

Dwie klasyczne postacie rozwiązania:

- przez macierz odwrotną:
$$x = A^{-1}b,$$
- wzory Cramera:
$$x_k = \frac{\det(A_k)}{\det(A)}, \qquad k = 1, \ldots, n,$$
gdzie A_k powstaje z A przez zastąpienie k-tej kolumny wektorem b.

13.5 Uwaga o uwarunkowaniu

Nawet mała perturbacja danych może silnie zmienić rozwiązanie. Przykład dla macierzy bliskiej osobliwej:

$$A = \begin{bmatrix} 1 & 0.99 \\ 0.99 & 0.98 \end{bmatrix},$$

porównujący rozwiązania układów $Ax = b$ oraz $A\tilde{x} = \tilde{b}$.

Uwaga. Część wartości liczbowych perturbacji na zdjęciu jest nieczytelna; sens przykładu: zadanie jest źle uwarunkowane.

13.6 Układy trójkątne

Układ dolnotrójkątny:

$$Lx = b, \qquad L = [\ell_{ij}],\ \ell_{ij} = 0 \text{ dla } j > i.$$

Rozwiązujemy podstawianiem w przód:

$$x_i = \frac{b_i - \sum_{j=1}^{i-1} \ell_{ij} x_j}{\ell_{ii}}, \qquad i = 1, \ldots, n,$$

przy $\ell_{ii} \neq 0$. Koszt: $O(n^2)$.

Układ górnotrójkątny:

$$Ux = b, \qquad U = [u_{ij}],\ u_{ij} = 0 \text{ dla } j < i.$$

Rozwiązujemy podstawianiem wstecz:

$$x_i = \frac{b_i - \sum_{j=i+1}^{n} u_{ij} x_j}{u_{ii}}, \qquad i = n, n-1, \ldots, 1,$$

przy $u_{ii} \neq 0$. Koszt: $O(n^2)$.

13.7 Metoda faktoryzacji (LU)

Niech A będzie odwracalna i załóżmy, że

$$A = LU,$$

gdzie L jest trójkątna dolna, a U trójkątna górna.

Wtedy

$$Ax = b \iff (LU)x = b \iff L(Ux) = b.$$

Wprowadzamy zmienną pomocniczą $y = Ux$ i rozwiązujemy dwa układy:

$$\begin{cases} Ly = b, \\ Ux = y. \end{cases}$$

Korzyść. Zamiast pełnego układu rozwiązujemy dwa układy trójkątne.

Koszt:

$$O(n^3) \text{ (wyznaczenie } L, U) + O(n^2) + O(n^2) = O(n^3).$$

13.8 Przykład

Rozwiązać $Ax = b$ metodą LU, gdzie

$$A = \begin{bmatrix} 1 & 2 & 3 \\ -3 & -2 & -4 \\ -5 & 18 & 26 \end{bmatrix}, \qquad b = \begin{bmatrix} 14 \\ -19 \\ 109 \end{bmatrix}.$$

Dla

$$L = \begin{bmatrix} 1 & 0 & 0 \\ -3 & 1 & 0 \\ -5 & 7 & 1 \end{bmatrix}, \qquad U = \begin{bmatrix} 1 & 2 & 3 \\ 0 & 4 & 5 \\ 0 & 0 & 6 \end{bmatrix}$$

mamy $A = LU$.

1) Rozwiązujemy $Ly = b$:

$$y = \begin{bmatrix} 14 \\ 23 \\ 18 \end{bmatrix}.$$

2) Rozwiązujemy $Ux = y$:

$$x = \begin{bmatrix} 1 \\ 2 \\ 3 \end{bmatrix}.$$

13.9 Warunek istnienia rozkładu LU (bez pivotingu)

Twierdzenie. Jeśli wszystkie minory główne wiodące macierzy $A = [a_{ij}] \in \mathbb{R}^{n\times n}$ są niezerowe,

$$\det \begin{bmatrix} a_{11} & \cdots & a_{1k} \\ \vdots & \ddots & \vdots \\ a_{k1} & \cdots & a_{kk} \end{bmatrix} \neq 0, \qquad k = 1, 2, \ldots, n,$$

to istnieje dokładnie jeden rozkład

$$A = LU,$$

gdzie L jest dolnotrójkątna z jedynkami na przekątnej, a U jest górnotrójkątna.

Współczynniki wyznaczamy rekurencyjnie:

$$u_{ij} = a_{ij} - \sum_{k=1}^{i-1} \ell_{ik} u_{kj}, \qquad i \leq j,$$

$$\ell_{ij} = \frac{a_{ij} - \sum_{k=1}^{j-1} \ell_{ik} u_{kj}}{u_{jj}}, \qquad i > j.$$

13.10 Zastosowania rozkładu LU

- Obliczanie wyznacznika:

  $$\det(A) = \det(L)\det(U) = u_{11}u_{22}\cdots u_{nn}$$

  (bo w tym wariancie $\det(L) = 1$).

- Obliczanie macierzy odwrotnej:

  $$A^{-1} = (LU)^{-1} = U^{-1}L^{-1},$$

  co sprowadza się do rozwiązywania układów trójkątnych.

Wykład 14 Eliminacja Gaussa w wersji numerycznej

Powtórka (z wykładu 13).

- Rozkład $A = LU$ pozwala rozwiązywać układ $Ax = b$ przez dwa układy trójkątne:

$$Ly = b, \qquad Ux = y.$$

- Koszt: wyznaczenie L, U rzędu $O(n^3)$, a każde podstawianie trójkątne rzędu $O(n^2)$.
- Z rozkładu LU:

$$\det(A) = \det(L)\det(U), \qquad A^{-1} = (LU)^{-1} = U^{-1}L^{-1}.$$

14.1 Odwracanie macierzy przez układy z wektorami bazowymi

Niech $A \in \mathbb{R}^{n\times n}$, $\det(A) \neq 0$, oraz

$$X := A^{-1} = [x_1 \ \cdots \ x_n].$$

Z równania

$$AX = I_n = [e_1 \ \cdots \ e_n]$$

dostajemy niezależne układy:

$$Ax_1 = e_1, \quad Ax_2 = e_2, \ \ldots, \ Ax_n = e_n.$$

Jeśli mamy już rozkład $A = LU$, to dla każdego k rozwiązujemy:

$$Ly_k = e_k, \qquad Ux_k = y_k.$$

Koszt całkowity:

$$O(n^3) + n \cdot O(n^2) = O(n^3).$$

Ważna uwaga numeryczna. W praktyce numerycznej zwykle unikamy jawnego wyznaczania A^{-1}, gdy celem jest rozwiązanie $Ax = b$.

14.2 Eliminacja Gaussa – schemat

Rozważamy układ

$$A^{(1)}x = b^{(1)},$$

gdzie standardowo $A^{(1)} = A$, $b^{(1)} = b$.

Krok 1

Zakładamy $a_{11}^{(1)} \neq 0$ i dla $i = 2, 3, \ldots, n$ bierzemy mnożniki

$$m_{i1} := -\frac{a_{i1}^{(1)}}{a_{11}^{(1)}}.$$

Dodając m_{i1}-krotność pierwszego równania do i-tego, dostajemy:

$$a_{ij}^{(2)} = a_{ij}^{(1)} + m_{i1} a_{1j}^{(1)}, \qquad j = 2, 3, \ldots, n,$$

$$b_i^{(2)} = b_i^{(1)} + m_{i1} b_1^{(1)}, \qquad i = 2, 3, \ldots, n.$$

Krok ogólny

Po $(r-1)$ krokach mamy układ

$$A^{(r)} x = b^{(r)},$$

i eliminujemy zmienną x_{r-1} z równań $i = r, \ldots, n$. Zakładając $a_{r-1,r-1}^{(r-1)} \neq 0$, bierzemy

$$m_{i,r-1} := -\frac{a_{i,r-1}^{(r-1)}}{a_{r-1,r-1}^{(r-1)}}, \qquad i = r, \ldots, n,$$

po czym aktualizujemy

$$a_{ij}^{(r)} = a_{ij}^{(r-1)} + m_{i,r-1} a_{r-1,j}^{(r-1)}, \qquad i, j = r, \ldots, n,$$

$$b_i^{(r)} = b_i^{(r-1)} + m_{i,r-1} b_{r-1}^{(r-1)}, \qquad i = r, \ldots, n.$$

Po $n-1$ krokach otrzymujemy układ trójkątny górny:

$$\sum_{j=r}^{n} a_{rj}^{(n)} x_j = b_r^{(n)}, \qquad r = 1, 2, \ldots, n,$$

z elementami głównymi $a_{rr}^{(r)} \neq 0$.

14.3 Podstawianie wsteczne i koszt

Rozwiązanie układu trójkątnego górnego:

$$x_r = \frac{b_r^{(n)} - \sum_{j=r+1}^{n} a_{rj}^{(n)} x_j}{a_{rr}^{(r)}}, \qquad r = n, n-1, \ldots, 1.$$

Koszt:

- faza eliminacji: $\sim O\left(\frac{n^3}{3}\right)$,
- podstawianie wsteczne: $O(n^2)$,
- łącznie: $O(n^3)$.

14.4 Problem numeryczny: zanik elementu głównego

Może się zdarzyć, że mimo istnienia jednoznacznego rozwiązania, w pewnym kroku eliminacji dostajemy

$$a_{rr}^{(r)} = 0$$

lub bardzo małą wartość, co psuje obliczenia (dzielenie przez zero albo duże błędy zaokrągleń).

Aby temu zapobiec, stosujemy **wybór elementów głównych** (pivoting).

14.5 Wybór elementów głównych

1) Częściowy wybór elementów głównych (partial pivoting). W kroku r zamieniamy miejscami wiersz r z wierszem p, gdzie

$$|a_{pr}^{(r)}| = \max_{r \le i \le n} |a_{ir}^{(r)}|.$$

2) Pełny wybór elementów głównych (full pivoting). W kroku r wybieramy (p, q) takie, że

$$|a_{pq}^{(r)}| = \max_{r \le i,j \le n} |a_{ij}^{(r)}|,$$

a następnie zamieniamy zarówno wiersze, jak i kolumny (czyli także numerację niewiadomych).

14.6 Wniosek o rozkładzie z permutacją

Twierdzenie. Dla dowolnej macierzy $A \in \mathbb{R}^{n \times n}$ istnieją:

- macierz permutacji wierszy P,
- macierz dolnotrójkątna L z jedynkami na przekątnej,
- macierz górnotrójkątna U,

takie, że

$$PA = LU.$$

To jest algebraiczny zapis eliminacji Gaussa z częściowym wyborem elementów głównych.