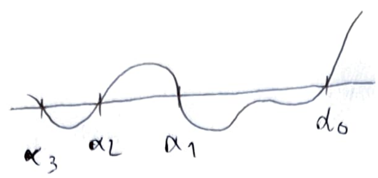
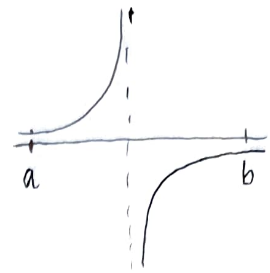
Powtórka:

- metoda bisekcji $(p=2)$
- metoda Newtona: $(p=1)$

$$\begin{cases} x_0 \text{ - dane (jak znaleźć?)} \\ x_{n+1} = x_n - \dfrac{f(x_n)}{f'(x_n)} \end{cases}$$

co jeśli nie znamy pochodnej?





- p - rząd zbieżności (wykładnik metody)

$$\lim_{n\to\infty} x_n = \alpha : \lim_{n\to\infty} \frac{|x_{n+1}-\alpha|}{|x_n-\alpha|^p} = C > 0$$

im większe p, tym szybciej x_n dąży do α

- metoda siecznych $(p \approx 1.6)$

x_0, x_1 - dane

$$x_{n+1} = x_n - \frac{f(x_n)}{\dfrac{f(x_n)-f(x_{n-1})}{x_n - x_{n-1}}}$$

Interpolacja wielomianowa

Oznaczenia

Π_n - zbiór wielomianów stopnia $\leq n$, $n \in \mathbb{N}$

$\Pi_n \setminus \Pi_{n-1}$ - zbiór wielomianów stopnia dokładnie n, $n\in\mathbb{N}$,

$\Pi_{-1} := \emptyset$

Postaci wielomianów:

(a) postać naturalna potęgowa:

$$w \in \Pi_n : w(x) = \sum_{k=0}^{n} a_k x^k$$ (a_k - współczynnik postaci potęgowej wielomianu w)

Jak dla danego $x \in \mathbb{R}$ obliczyć $w(x)$?

Schemat Hornera!

$$\begin{aligned} w(x) &= a_n x^n + a_{n-1} x^{n-1} + \ldots + a_1 x + a_0 = \\ &= x(a_n x^{n-1} + a_{n-1}x^{n-2} + \ldots + a_1) + a_0 = \ldots = \\ &= x(x(\ldots x(a_n x + a_{n-1}) + a_{n-2}) + a_{n+3}) + \ldots + a_1) + a_0 = \\ &= x(x(\ldots \underbrace{x(\underbrace{x \cdot \underbrace{(a_n)}_{w_n} + a_{n-1}}_{w_{n-1}}) + a_{n-2}}_{w_{n-2}}) + a_{n+3}) + \ldots + a_1) + a_0 \end{aligned}$$

Algorytm Hornera Czas: $O(n)$

$$\begin{cases} w_n := a_n \\ w_k := w_{k+1} \cdot x + a_k \quad (k = n-1, n-2, \dots, 0) \end{cases}$$

Wtedy $w(x) = w_0$

Twierdzenie: Algorytm Hornera jest algorytmem numerycznie poprawnym.

(b) postać Newtona

Dla danych liczb $x_0, x_1, x_2, \dots$ określamy wielomiany

$$p_0(x) \equiv 1, \quad p_k(x) := (x - x_{k-1})\, p_{k-1}(x) \quad (k = 1, 2, \dots)$$

Obserwacja: $p_k(x) = (x-x_0)(x-x_1)\dots(x-x_{k-1}), \quad k \geq 1$

$$p_k = \prod_{j=0}^{k-1} (x - x_j), \quad p_k \in \Pi_k \setminus \Pi_{k-1}$$

$w \in \Pi_n : w(x) = \sum_{k=0}^{n} b_k p_k(x)$ (b_k - współczynniki postaci Newtona wielomianu w)

Jak dla danego $x \in \mathbb{R}$ obliczyć $w(x)$?

Ulepszony (uogólniony) schemat Hornera:

$$\begin{aligned} w(x) &= b_n p_n(x) + b_{n-1} p_{n-1}(x) + \dots + b_1 p_1(x) + p_0 = \\ &= b_n (x-x_0)(x-x_1)\dots(x-x_{n-2})(x-x_{n-1}) + \\ &+ b_{n-1}(x-x_0)(x-x_1)\dots(x-x_{n-2}) + \\ &+ b_{n-2}(x-x_0)(x-x_1)\dots(x-x_{n-3}) + \dots + \\ &+ b_1 (x-x_0) \\ &+ b_0 \cdot 1 = \end{aligned}$$

$$= \overbrace{\Big(\dots\underbrace{\Big(\underbrace{\big(\underbrace{b_n}_{w_n}(x-x_{n-1}) + b_{n-1}\big)}_{w_{n-1}}(x-x_{n-2}) + b_{n-2}\Big)}_{w_{n-2}}(x-x_{n-3}) + \dots + b_1\Big)(x-x_0) + b_0 \cdot 1}^{w_0}$$

Uogólniony algorytm Hornera   czas: $O(n)$

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

Wtedy $w(x) = w_0$

Dane:
$x, x_0, x_1, \dots, x_{n-1}$,
$b_0, b_1, \dots, b_n$
(czyli używamy więcej pamięci)

Twierdzenie: Uogólniony schemat Hornera to algorytm numerycznie poprawny.

(c) postać Czebyszewa

Wprowadzamy następujący ciąg wielomianów $T_0, T_1, T_2, \dots$ zdefiniowany rekurencyjnie:

- $T_0(x) \equiv 1$, $T_1(x) \equiv x$
- $T_k(x) = 2x\,T_{k-1}(x) - T_{k-2}(x) \quad (k = 2, 3, \dots)$

Podstawowe własności wielomianów Czebyszewa:

1° $T_k \in \Pi_k \setminus \Pi_{k-1}$ $T_k(x) = 2^{k-1}x^k + 0 \cdot x^{k-1} + \dots \quad (k \geq 1)$

2° T_{2n} – funkcja parzysta, T_{2k+1} – funkcja nieparzysta

3° $\text{lin}\{T_0, T_1, \dots, T_n\} = \Pi_n \Rightarrow$ każdy wielomian można zapisać jako kombinację liniową wielomianów Czebyszewa (baza Π_n)

4° T_k ma dokładnie <u>k miejsc zerowych</u> należących do przedziału $(-1, 1)$
ISTNIEJE JAWNY WZÓR

5° Dla $x \in [-1, 1]$ mamy $T_k(x) = \cos(k \cdot \arccos x)$

Stąd wynika, że zbiór wartości wielomianu Czebyszewa to $[-1, 1]$

to wynika z szeregu Fouriera

$$w \in \Pi_n: \quad w(x) = \frac{1}{2} c_0 T_0(x) + c_1 T_1(x) + \ldots + c_n T_n(x) =: \sum_{k=0}^{n}{}' c_k T_k(x)$$

(c_k - współczynniki postaci Czebyszewa wielomianu w)

prim oznacza mnożenie pierwszego składnika przez $\frac{1}{2}$,
$\sum_{k=0}^{n}{}' a_k = \frac{1}{2} a_0 + a_1 + a_2 + \ldots$

Jak dla danego $x \in \mathbb{R}$ obliczyć $w(x)$ (dla wielomianu w podanego w postaci Czebyszewa)?

Algorytm Clenshawa: ze względów numerycznych wartość wielomianu podanego w postaci Czebyszewa zaleca się obliczać przy pomocy następującego algorytmu:

czas: $O(n)$

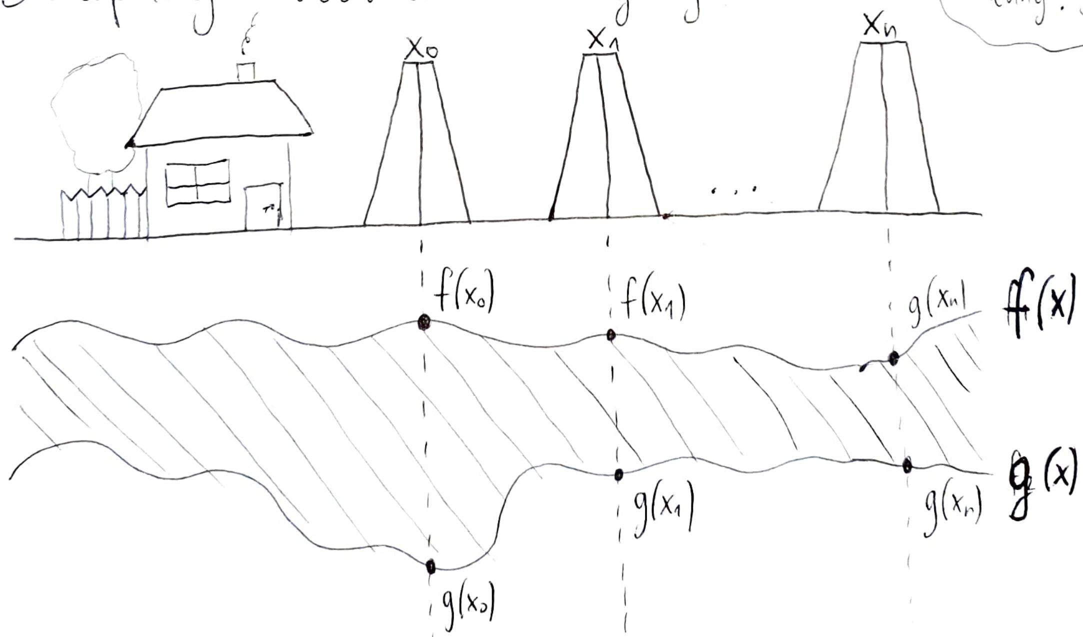
Dane: $x, c_0, c_1, \ldots, c_n$

$$\left[\begin{array}{l} B_{n+2} := 0, \; B_{n+1} := 0 \\ B_k := 2x B_{k+1} - B_{k+2} + c_k \quad (k = n, n-1, \ldots, 0) \end{array}\right.$$

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

Jak zamieniać jedną z postaci wielomianu na inną? Jaka jest złożoność?

Interpolacja wielomianowa Lagrange'a



Dane: $\left.\begin{array}{l} x_0, f(x_0) \\ x_1, f(x_1) \\ \vdots \\ x_n, f(x_n) \end{array}\right\} \rightarrow f(x) = ?$ tzn. jak „odtworzyć" funkcję f na podstawie skończonej liczby informacji

Zadanie (interpolacja Lagrange'a)

Dla danych parami różnych $x_0, x_1, \dots, x_n \in \mathbb{R}$ i odpowiadających im wartości $y_0, y_1, \dots, y_n$, znaleźć taki wielomian L_n, że:

$$\begin{cases} L_n \in \Pi_n \\ L_n(x_k) = y_k \quad (k = 0, 1, \dots, n) \end{cases}$$

Twierdzenie: Zadanie interpolacyjne Lagrange'a ma zawsze jednoznaczne rozwiązanie.

Twierdzenie:

$$L_n(x) = \sum_{k=0}^{n} y_k \lambda_k(x), \text{ gdzie } \lambda_k(x) := \prod_{\substack{i=0 \\ i \neq k}}^{n} \frac{x - x_i}{x_k - x_i}.$$

$x_0, x_1, \dots, x_n$ nazywamy węzłami interpolacji

Dowód: Oczywiście $L_n \in \Pi_n$. Zauważmy, że

$$\lambda_k(\overset{\text{j-ty węzeł}}{x_j}) = \begin{cases} 1 : & k = j \\ 0 : & k \neq j \end{cases}, \quad j = 0, 1, \dots, n$$

$\blacksquare$

Przykład: $f(x) = e^x$, $n = 3$, $\underbrace{x_0 = 0,\ x_1 = 0.2,\ x_2 = 0.6,\ x_3 = 0.8}_{\text{jak znaleźć } f(0.4)?}$

x_k	x_0	x_1	x_2	x_3
$f(x_k)$	$f(x_0)$	$f(x_1)$	$f(x_2)$	$f(x_3)$
	$\parallel$	$\parallel$	$\parallel$	$\parallel$
	1	$1.22\ldots$	$1.82\ldots$	$2.22\ldots$

$$L_3(x) = f(x_0)\lambda_0(x) + f(x_1)\lambda_1(x) + f(x_2)\lambda_2(x) + f(x_3)\lambda_3(x)$$

$$\lambda_0(x) = \frac{x - x_1}{x_0 - x_1} \cdot \frac{x - x_2}{x_0 - x_2} \cdot \frac{x - x_3}{x_0 - x_3} = -\frac{1}{0.096}(x - 0.2)(x - 0.6)(x - 0.8)$$

$$\lambda_1(x) = \frac{x - x_0}{x_1 - x_0} \cdot \frac{x - x_2}{x_1 - x_2} \cdot \frac{x - x_3}{x_1 - x_3} = \frac{1}{0.048}(x - 0.2)(x - 0.6)(x - 0.8)$$

$$\lambda_2(x) = \frac{x-x_0}{x_2-x_0}\cdot\frac{x-x_1}{x_2-x_1}\cdot\frac{x-x_3}{x_2-x_3} = -\frac{1}{0.048}(x-0.2)(x-0.6)(x-0.8)$$

$$\lambda_3(x) = \frac{x-x_0}{x_3-x_0}\cdot\frac{x-x_1}{x_3-x_1}\cdot\frac{x-x_2}{x_3-x_2} = \frac{1}{0.096}(x-0.2)(x-0.6)(x-0.8)$$

$$L_3(0.4) = \underbrace{1.491}42\ldots \qquad f(0.4) = e^{0.4} = \underbrace{1.491}82$$

$$|L_3(0.4) - f(0.4)| \approx 0.4\cdot 10^{-3}$$

$$\max_{0\le x\le 0.8} |L_3(x) - f(x)| \approx 0.4\cdot 10^{-3}$$

Przykłady:

(a) $f(x) = \sin(x)$, $[0, 2\pi]$, $x_k := \frac{2k\pi}{n}$ $(k = 0,1,\ldots,n;\ n = 1,2,\ldots)$

↑ węzły równoodległe w przedziale $[0, 2\pi]$

1° $L_1(x) = 0$

2° $L_2(x) = 0$

3° $L_3(x)$ zaczyna przypominać $\sin(x)$, podobnie $L_4(x)$

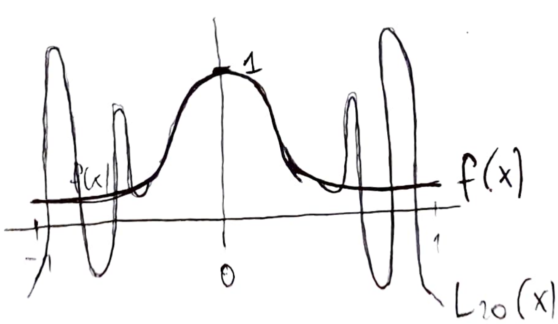
4° $L_5(x)$ jest bardzo dobrym przybliżeniem $\sin(x)$

5° $L_{14}(x)$ przedstawia funkcję $\sin(x)$ z błędem $7\cdot 10^{-6}$!!

(b) $f(x) = \frac{1}{25x^2+1}$, $[-1,1]$, $x_k = -1 + \frac{2k}{n}$ $(k = 0,1,\ldots,n;\ n = 1,2,\ldots)$

funkcja Rungego

Mieliśmy gładką funkcję w okolicach $x \in [0, 0.6]$, jednak później wartości osiągnęły błąd ~ 58.6. ← efekt Rungego



(c) $f(x) = x^6$, $[-1,1]$, $x_k = -1 + \frac{2k}{n}$ $(k = 0, 1, \ldots, n)$ $n = 1, 2, \ldots)$

$L_0(x), L_1(x), \ldots, L_5(x)$ musielibyśmy obliczyć, a dla $n \geq 6$ mamy $L_n(x) = x^6$, a więc $L_{2019}(x) = x^6$, numerycznie jednak nawet Maple ma problemy z obliczaniem kolejnych $L_k(x)$.