Přednáška 2 - Interpolace

interpolace ~ interpolation ≃ $\underbrace{\text{inter}}_{\text{mezi}}$ - $\underbrace{\text{polare}}_{\equiv \text{polish}}$ (latina)

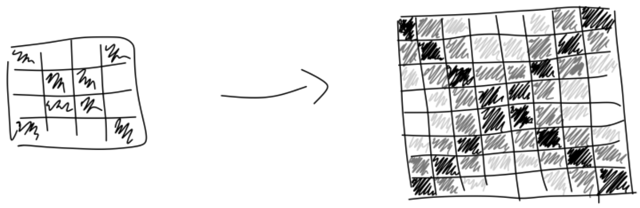
$\equiv$ polish $\equiv$ vyprecizovat / vybrousit / vyleštit / uhladit

$\Rightarrow$ interpolace funkce/dat „zhlazuje"

$\equiv$ výstupem by měly být „hladké" funkce

Motivace: image processing

- obrázek $\equiv \left\{ [r_{ij}]_{ij}, [g_{ij}]_{ij}, [b_{ij}]_{ij} \right\}$ 300 × 300 pixelů
- display $\equiv \left\{ [s_{ij}]_{ij}, [h_{ij}]_{ij}, [c_{ij}]_{ij} \right\}$ 600 × 600 pixelů
- jak zobrazit obrázek na display?
  $\rightsquigarrow$ „rozsadíme" pixely obrázku a ty pixely „mezi" interpolujeme $\equiv$ vyhladíme



(opak komprese, která nás čeká na konci semestru)

PythonDemo: image processing

terminologie:

- interpolace $\equiv$ doplň mezi
- extrapolace $\equiv$ doplň vně

Formulace problému v 1D

Mám:
- interval (a,b)
- body ($\equiv$ uzly $\equiv$ nodes) $x_0, \dots, x_n \in (a,b)$
- hodnoty v uzlech $f_0, \dots, f_n \in \mathbb{R}$

Chci: interpolující funkci, tj. $P_f(x) : (a,b) \longrightarrow \mathbb{R}$ t., že $\boxed{\forall i : P_f(x_i) = f_i}$

$\equiv$ tzv. interpolation property / condition

Poznámka: rozšíření do 2D/3D jde různě, např. $P_f(x,y) = P_f^{(x)}(x) \cdot P_f^{(y)}(y)$
nebo přes jinou geometrii, viz demo / googlecolab.

Krok 1: Typ interpolace

→ my se omezíme na „klasickou" interpolaci → polynomiální

⇝ $P_f(x) = \sum_{i=0}^{n} \alpha_i x^i = \alpha_0 + \alpha_1 x + \alpha_2 x^2 + \dots + \alpha_n x^n$

Pozorování – jednu metodu polynomiální aproximace už známe – Taylorův polynom. Ten ale splňuje $P_f(x_i) = f_i$ pouze pro jeden bod → nestačí

pouze

→ V praxi se běžně setkáváme i s jinými typy interpolací – místo polynomů lze uvažovat trigonometrické polynomy → $\alpha_0 + \alpha_1 \cos(\pi x) + \dots + \alpha_n \cos(n\pi x)$
$\beta_1 \sin(\pi x) + \dots + \beta_n \sin(n\pi x)$ $\quad x \in (-1,1)$

⇝ to ale jde nad rámec přednášky.

Krok 2: naivní formulace

Rozepíšeme si interpolační podmínky:

$$\forall i = 0, \dots, n: \quad \alpha_0 + \alpha_1 x_i + \alpha_2 (x_i)^2 + \dots + \alpha_n (x_i)^n = f_i$$

⇕

$$\forall i = 0, \dots, n: \quad \begin{bmatrix} 1 & x & (x_i)^2 & \dots & (x_i)^n \end{bmatrix} \begin{bmatrix} \alpha_0 \\ \vdots \\ \alpha_n \end{bmatrix} = f_i$$

⇕

$$\underbrace{\begin{bmatrix} 1 & x_0 & \dots & (x_0)^n \\ \vdots & \ddots & & \vdots \\ 1 & x_n & \dots & (x_n)^n \end{bmatrix}}_{\text{tzv. Vandermondova matice}} \begin{bmatrix} \alpha_0 \\ \vdots \\ \alpha_n \end{bmatrix} = \begin{bmatrix} f_0 \\ \vdots \\ f_n \end{bmatrix}$$

→ Když vyřešíme soustavu $n+1$ lineárních algebraických rovnic, máme $P_f(x)$

- Nevýhoda 1: pokud mi někdo zítra přidá (x_{n+1}, f_{n+1}) ⇝ vše počítám znova

PythonDemo: zkusíme vyřešit jako v Lingebra 1

- Nevýhoda 2: „asi těžký problém" → nespecializované metody (G.E.) můžou selhat

Krok 3: Lagrangeova interpolace

V čem je problém? Hledáme $p_f(x)$ jako lineární kombinaci x^i - monických polynomů

Existuje lepší báze! Definujeme polynomy $l_0(x), \dots, l_n(x)$ stupně $\underline{n}$: $\forall i$: $\boxed{l_i(x_j) = \delta_{ij} \text{ pro všechny } x_j = x_0, \dots, x_n}$

$\equiv$ n+1 podmínek určuje polynom stupně $\underline{n}$ jednoznačně

Z definice dostaneme: $f_0 \cdot l_0(x_i) = \begin{cases} 0 & \text{pro } i \neq 0 \\ f_0 & \text{pro } i = 0 \end{cases}$ ..... odtud je přirozená myšlenka na přímé řešení našeho problému

Pokud definujeme $p_f(x) := f_0 \cdot l_0(x) + f_1 \cdot l_1(x) + \dots + f_n \cdot l_n(x)$, pak:
- $p_f(x)$ je polynom stupně $\underline{n}$
- $p_f(x_i) = f_i \quad \forall i$

Pozorování $\rightarrow$ libovolný polynom stupně $\underline{n}$ je dán právě n+1 body. My jsme ukázali, že jsme schopni pro daných n+1 bodů (různých) napsat jimi určený polynom stupně $\underline{n}$ jako lineární kombinaci $l_0(x), \dots, l_n(x)$. Tedy jsme ukázali, že $\{l_0(x), \dots, l_n(x)\}$ je báze vektorového prostoru polynomů stupně $\underline{n}$.

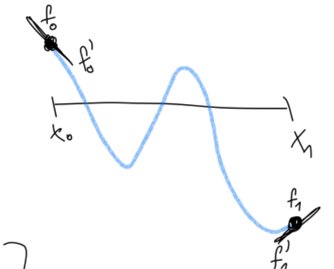
Umíme sestrojit $l_i(x)$? Ano: $$l_i(x) = \prod_{j \neq i} \frac{x - x_j}{x_i - x_j}$$ .... tzv. Lagrangeovy bazické polynomy

Python Demo: Rungeho jev

Krok 4: - Hermite

$\rightarrow$ co když chceme aby $\frac{d}{dx} p_f(x_i) = f_i'$?

zadané hodnoty $\in \mathbb{R}$ nikoliv derivace konstant



$\rightsquigarrow$ lze analogicky přes systém lin. rovnic:

- $p_f(x) = \sum_0^n \alpha_i x^i$

$\Downarrow$

$\frac{d}{dx} p_f(x) = \sum_0^{n-1} (i+1) \cdot \alpha_{i+1} \cdot x^i$

$\Rightarrow$ n+1 rovnic „$p_f(x_i) = f_i$“, n+1 rovnic „$\frac{d}{dx} p_f(x_i) = f_i'$“ ---> pouze větší systém (a tedy vyšší stupeň $p_f(\cdot)$)

$\rightsquigarrow$ lze analogicky jako u Lagrangeových polynomů:

$$h_i(x) := \left(1 - 2(x - x_i) \cdot l_i'(x_i)\right) \cdot l_i^2(x) \quad \text{-----} \quad h_i(x_j) = \delta_{ij}, \quad \frac{d}{dx} h_i(x_j) = 0$$

$$g_i(x) := (x - x_i) \cdot l_i^2(x) \quad \text{------} \quad g_i(x_j) = 0, \quad \frac{d}{dx} g_i(x_j) = \delta_{ij}$$

$$\Longrightarrow \quad p_f(x) := \sum_0^n \alpha_i h_i(x) + \sum_0^n \beta_i g_i(x)$$