Přednáška 2 - Interpolace

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

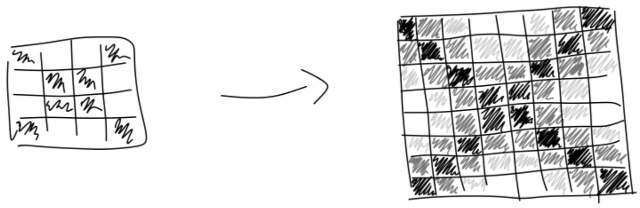
≡ polish ≡ vyprecizovat / vybrousit / vyleštit / uhladit

⇒ interpolace funkce/dat „zhlazuje"

≡ výstupem by měly být „hladké" funkce

Motivace: image processing

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



(opak komprese, která nás čeká na konci semestru)

PythonDemo: image processing

terminologie:

- interpolace ≡ doplň mezi
- extrapolace ≡ doplň vně

Formulace problému v 1D

Mám:
- interval (a,b)
- body (≡ uzly ≡ 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 $\boxed{\forall i : P_f(x_i) = f_i}$

≡ 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

⇝ $p_f(x) = \alpha_0 + \alpha_1 x + \alpha_2 x^2 + \dots + \alpha_p x^p$

Pozorování – jednu metodu polynomiální aproximace už známe → → Taylorův polynom. Ta by ale splňovala $p_f(x_i) = f_i$ pouze pro jeden bod ⇝ nestačí

→ V praxi se často setkáme i s jinými typy interpolací → → např. místo x^i můžeme brát $\cos(i \cdot \pi \cdot x)$ (nebo $\sin(\dots)$ nebo obojí) ⇝ to ale jde nad rámec přednášky

Krok 2: naivní formulace

Rozepíšeme si interpolační podmínky:

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

$$\Updownarrow$$

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

$$\Updownarrow$$

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

→ Když vyřešíme lineární soustavu 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 problém? Hledáme $p_f(x)$ jako LK monických pol. x^i

→ lépe: mějme polynomy $l_i(x)$ t.ž. $l_i(x_j) = \delta_{ij}$ $\quad i=0,\dots,n,\ j=0,\dots,n$

a zkusme $p_f(x) = \beta_0 l_0(x) + \dots + \beta_n l_n(x)$

⇒ zjevně $p_f(x_i) = \beta_i$ ⇒ volba $\beta_i = f(x_i)$

⇒ není nutné nic „počítat" → $f(x) \approx \sum_0^n l_i(x) f(x_i)$

ZMĚNA BÁZE PROSTORU POLYNOMŮ VE KTERÉ HLEDÁM MOJI APPROX.

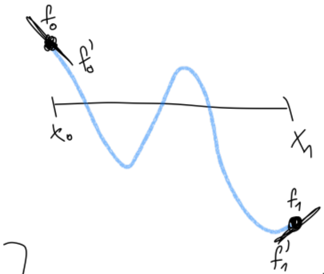
→ umíme spočítat $l_i(x)$?

$$l_i(x) = \prod_{j \neq i} \frac{x - x_j}{x_i - x_j}$$

PythonDemo: Rungeho jev

Krok 4: – Hermite

zadané hodnoty ∈ R nikoliv derivace konstant



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

⇝ lze analogicky přes systém lin. rovnic:

$$\left.\begin{aligned} \bullet\ p_f(x) &= \sum_0^n \alpha_i x^i \\ &\Downarrow \\ \frac{d}{dx}p_f(x) &= \sum_0^{n-1} (i+1)\cdot\alpha_{i+1}\cdot x^i \end{aligned}\right\} \Rightarrow \left.\begin{aligned} &n+1 \text{ rovnic } „p_f(x_i) = f_i" \\ &n+1 \text{ rovnic } „\tfrac{d}{dx}p_f(x_i) = f_i'" \end{aligned}\right\}$$

⇝ pouze větší systém (a tedy vyšší stupeň $p_f(\cdot)$)

⇝ lze analogicky jako u Lagrangeových polynomů:

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

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

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