Numerik 1 Übungstunde, Gruppe 1 (Donnerstag)

Woche 02

Übungstunden werden registriert. Bitte informieren Sie den Assistenten, falls Sie nicht in der Aufzeichnung erscheinen möchten.

Streaming (öffentlich) und Aufzeichnungen (Anmeldung nötig):

https://seminarlive.mnf.uzh.ch/semlive/index.php?id=event_detail&module=mat801&semester=fs23&group=1

Notizen (diese Datei) und Python scripts:

https://user.math.uzh.ch/florian/stable/page/2023Mat801/notes.php

hw 02, 02

$S_g^{k,m}$ $\mathbb{P}_m(I)$

g a "mesh" on I

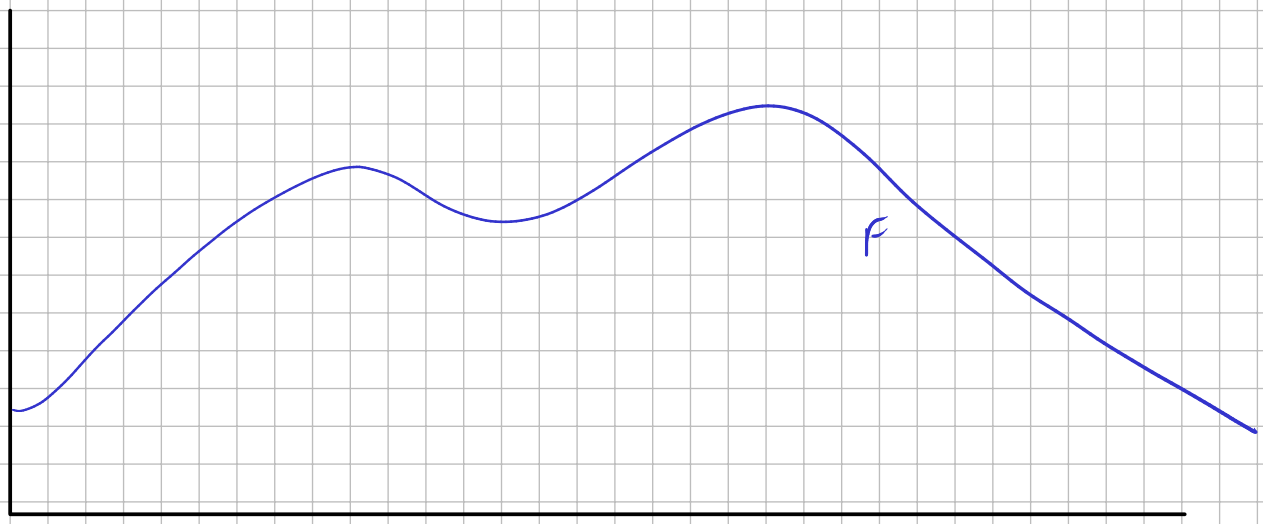
To prove for two sets A, B

$A = B$

you can prove

$A \subseteq B$, $B \subseteq A$

- Take $f \in S_g^{k,m}$ and prove $f \in \mathbb{P}_m(I)$,
- Take $h \in \mathbb{P}_m(I)$ " " $h \in S_g^{k,m}$



We want to interpolate f "in some way" with "sort-of" polynomials such that when increasing the number of nodes, the interpolant gets arbitrarily close to f

- It is clear that not always increasing the degree of the polynomial is enough
- So we use another strategy

→ decrease the interval length

Therefore we need to use more than one interval to cover the original domain

(⇒ Splines)

Horner's algorithm

$(x^n)_{n=0}^N$ monomial basis

$$p(x) = \sum_{n=0}^{N} c_n \cdot b_n(x) = \sum_{n=0}^{N} c_n x^n$$

Algorithm 0

Input: c (coefficients)

N

X point where you want to evaluate p

Output: p(X)

evaluation = 0 # this holds p(X)

loop on k (k=0 to N)

- compute X^k ⟶
- evaluation += $c_k \cdot X^k$

return evaluation

Problem: at step k we compute X^k

so at step k+1 we could reuse it *

to compute X^{n+1} as $X^n \cdot X$

↳ already computed

Algorithm 1

Input: c (coefficients)

N

X point where you want to evaluate p

Output: p(X)

evaluation = c_0

polynomial = 1

loop on k: (1 to N)

polynomial *= X # was $X^{n-1} \to X^n = X^{n-1} \cdot X$

evaluation += $c_k \cdot$ polynomial

return evaluation

* We want to compute x^3 (or x^{k+1})

1 (start)

$1 \cdot x \to x$

$x \cdot x \to x^2$

$x^2 \cdot x \to x^3$

Algorithm 2 (Horner)

$$\begin{pmatrix} c_0 x^0 + c_1 x^1 + c_2 x^2 + c_3 x^3 \\ c_0 + x \cdot (c_1 + x \cdot (c_2 + x \cdot c_3)) \end{pmatrix}$$

Input: c (coefficients)

N

x point where you want to evaluate p

Output: $p(x)$

evaluation $= c_N$

loop on n (backwards: $N-1$ to 0):

evaluation $* = x$

evaluation $+ = c_n$

return evaluation

Complexity:

	steps	complexity per step	total
Algorithm 0:	$O(N)$	$O(n)$	$O(N^2)$ $\left(\sum_{n=0}^{N} n\right)$
1	$O(N)$	$O(1)$ (2)	$O(N)$

2 $O(N)$ $O(1)$ $O(N)$

(2)

Algorithm 2b

Input: N

x (point where you want to evaluate p)

s (nodes)

b (divided differences)

Output: $p(x)$

$$p(x) = \sum_{n=0}^{N} b_n w_{k-1}(x)$$

$$= \sum_{n=0}^{N} b_n \prod_{\ell=0}^{k-1} (x - s_\ell)$$

(monomial basis: $\prod_{\ell=0}^{k-1} x$)

evaluation $= b_N$

loop on k $(N-1 \to 0)$

evaluation $*= x - s_k$

evaluation $+= b_k$

return evaluation

$$p(x) = b_0 \cdot 1 + b_1 (x - s_0) + b_2 (x - s_0)(x - s_1)$$

$$b_0 + (x - s_0)(b_1 + (x - s_1) \cdot b_2)$$