SFU
muraki

FRIDAY

- QUIZ TODAY: be at lecture on time + bring calculator
- computing workshop: FRI + MON @ $2^{30}$ (can use room next door)

LAST DAY

- closing comments on root-finding (CH 2.)
- introduction to INTERPOLATION
  - polynomial interpolation – exactly contains all points $(x_k, y_k)$, $k = 0 \to N$
  - linear algebra suggests degree of $P(x)$ is. $N$ (but can be less, if $a_N = 0$, $a_{N-1} = 0$ ... etc)

MONOMIAL BASIS POLYNOMIALS

- familiar polynomial $\quad P(x) = \underbrace{\sum_0^N a_j x^j}_{N+1 \text{ terms}}$

- Vandermonde linear solve

$$\begin{array}{c} k=0 \\ \\ k=N \end{array}\left[\begin{array}{ccc} & & \\ & (x_k)^j & \\ & & \end{array}\right]\begin{pmatrix} a_N \\ \vdots \\ a_j \\ \vdots \\ a_0 \end{pmatrix} = \begin{pmatrix} y_0 \\ \vdots \\ y_k \\ \vdots \\ y_N \end{pmatrix}$$
$$\qquad j=N \qquad\qquad j=0$$

gives a — linear solve! :(

# LAGRANGE BASIS POLYNOMIALS

- case $N=2$, points list: $\{(\ ,f_a),(\ ,f_b),(\ ,f_c)\}$
  $\quad 0 \qquad 1 \qquad 2$

- make 3 basis polynomials

$$L_0(x) = \frac{(\quad b)(\quad c)}{(\quad b)(\quad c)}$$

$L_0(b) = L_0(c) =$
& $L_0(a) =$
& quadratic poly.

$L_1(x)$ & $L_2(x)$

- LAGRANGE INTERPOLATING POLYNOMIAL

$$P_{lagr}(x) = \frac{(x-b)(x-c)}{(a-b)(a-c)}$$

$$+ \frac{(x-a)(x-c)}{(b-a)(b-c)} + \frac{(x-a)(x-b)}{(c-a)(c-b)}$$

& confirm $P_{lagr}(\ ) =$ , etc...

$$P_{lagr}(x) = \sum_{0}^{N} \qquad (x)$$

of $L_j$'s

- Lagrange $\{L_0(x), L_1(x) \ldots L_j(x) \ldots L_N(x)\}$

all

• is the Lagr. Interp Poly same as ?

, but !

• addresses poor-conditioning of monomial
but might still yield a of
function.

• costly in terms of

$$\begin{matrix}\text{op count} \\ \text{for } L_k(x)\end{matrix} = \begin{matrix}\text{op count} \\ \text{for}\end{matrix} \left\{ \sum_0^N \underbrace{y_k L_k(x)}_{} \right\} = O(\quad)$$

per

value of $x$ in $x_k$ -

$\Rightarrow$ terms adds

mult. + div

+ - for numerator

+ - for denominator

• KEY IDEA: -dependent basis $\rightarrow$

• what is the issue here? $a < x_k < b$

• say we pick a function & + sample points,
since this determines a $P(x)$, we
can ask how big: $(x) = |\ (x) - \ (x)\ |$
gets on $\le x \le$ ?

**Theorem 3.3** Suppose $\underbrace{x_0, x_1, \ldots, x_n}_{}$ values are distinct numbers in the interval $[a, b]$ and $f \in C^{n+1}[a, b]$. Then, for each $x$ in $[a, b]$, a number $\xi(x)$ (generally unknown) between $x_0, x_1, \ldots, x_n$, and hence in $(a, b)$, exists with

$$f(x) = P(x) + \boxed{\frac{f^{(n+1)}(\xi(x))}{(n+1)!}(x - x_0)(x - x_1) \cdots (x - x_n)}, \tag{3.3}$$

$\xi$

poly.

at all !!

this factor gives the

of given on question

(but we don't know $a < \xi(x) < b$.

however $|f^{(n+1)}(x)|$ is

$[a,b]$

worst case )

$\left(\frac{d^{n+1}}{dx^{n+1}} f\right)$

**(Taylor's Theorem)**

Suppose $f \in C^n[a, b]$, that $f^{(n+1)}$ exists on $[a, b]$, and $x_0 \in [a, b]$. For every $x \in [a, b]$, there exists a number $\xi(x)$ between $x_0$ and $x$ with

$$f(x) = P_n(x) + R_n(x),$$

where

$$P_n(x) = f(x_0) + f'(x_0)(x - x_0) + \frac{f''(x_0)}{2!}(x - x_0)^2 + \cdots + \frac{f^{(n)}(x_0)}{n!}(x - x_0)^n$$
$$= \sum_{k=0}^{n} \frac{f^{(k)}(x_0)}{k!}(x - x_0)^k$$

and

$$R_n(x) = \frac{f^{(n+1)}(\xi(x))}{(n+1)!}(x - x_0)^{n+1}.$$

in error terms! ■

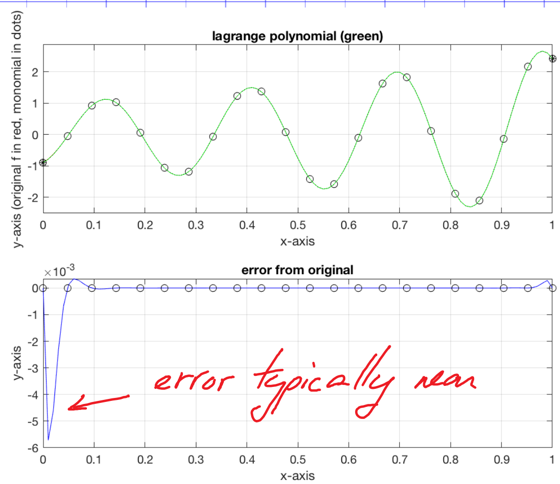
Here $P_n(x)$ is called the **$n$th Taylor polynomial** for $f$ about $x_0$, and $R_n(x)$ is called the **remainder term** (or **truncation error**) associated with $P_n(x)$. Since the number $\xi(x)$ in the truncation error $R_n(x)$ depends on the value of $x$ at which the polynomial $P_n(x)$ is being evaluated, it is a function of the variable $x$. However, we should not expect to be

• polynomial interpolation demo

• sample points
incl ( ,f )
& ( ,f )

a) -

b)




• Vandermonde solve has cond( )
for N around (dotted)

• works well
beyond Vandermonde warnings!

CHAPTER 3 IS SO LAST CENTURY!

• the textbook BF, chapter 3 reads as:
  • §3.1 – Lagrange interp, it works but is costly to use ...
  • §3.2 – so we learn Newton interpolation & divided differences.

SFU muraki

it turns out that in 2004, two papers* were published showing that forgotten ideas (from as far back as 1945) allow for both an efficient, and more robust (than Newton interpolation), numerical algorithm for Lagrange interpolation.

*links on lecture page

WHEN IS A NOT A ?

$$P_{lagr}(x) = f_a \cdot \frac{(x-b)(x-c)}{(a-b)(a-c)} + f_b \cdot \frac{(x-a)(x-c)}{(b-a)(b-c)} + f_c \frac{(x-a)(x-b)}{(c-a)(c-b)}$$

N=2 case

• form COMMON ATOR

$$= \underbrace{(x-a)(x-b)(x-c)}_{\text{degree poly}} \left\{ \underbrace{f_a}_{\text{-values}} \cdot \underbrace{\frac{1}{(a-b)(a-c)}}_{\text{numbers depending only on } \{\ \}} \underbrace{\frac{1}{(x-a)}}_{\text{funct.}} + f_b \cdot \frac{1}{(b-a)(b-c)} \frac{1}{(x-b)} + f_c \cdot \frac{1}{(c-a)(c-b)} \frac{1}{(x-c)} \right\}$$

WARNING! / limit as $x \to x_k$ !!

SFU

muraki

- general form MODIFIED LAGRANGE INTERPOLANT (MLI)

$$P_{lagr}(x) = \left( \prod_{0}^{N} (x - x_k) \right) \left\{ \sum_{0}^{N} f_k \cdot \frac{w_k}{(x - x_k)} \right\}$$

$( \; )$ $\qquad w_k = \left( \prod_{j \neq k} ( \; - \; ) \right)^{-1}$

- flops to compute $\{w_k\}_{0 \to N}$ =

  # of ops per w-value = ( subt ) + ( mult/div )

  =

- flops to compute one value of $P(x)$, $x \neq \{x_k\}$

  # of ops for $L(x)$ = ( subt + mult )

  + ( $N$ adds in sum

  + $3(N+1)$ per term )

  = ( + ) + ( + )

  = $O( \; )$ per x-value !

- also, since $\{w_k\}$ does not depend on $\{ \; \}$

  can – & for same $\{ \; \}$ values !

- despite , Higham 2004 shows MLI to well-beyond vandermonde failure

- there is another version: BARYCENTRIC FORMULA.

$$P_{bary}(x) = \frac{\sum_0^N f_k \cdot \frac{w_k}{(x-x_k)}}{\sum_0^N \frac{w_k}{(x-x_k)}}$$

→ version used in Matlab script from Mathworks repository

POLYNOMIAL INTERPOLATION for VERY LARGE $N$

- for uniform spacing on $0 \le x \le 1$, even MLI for $N$ because

$$w_j = (-1)^j \binom{N}{j} = (-1)^j \frac{N!}{j!\,(N-j)!}$$

and some $w_j$ reach at $N=$

SFU
muraki

- for randomly chosen $\{x_k\}$ the $w_j$'s grow
- BUT, there are special choices of $\{x_k\}$ where the $\{w_j\}$ GROW !!

SFU

muraki