SFU
muraki

MONDAY
- computing workshops @ 2^{30}

LAST DAY

- accuracy for Lagrange interpolating polynomial
  - error term formula.
  - DEMO: more points → smaller error
    for uniform & randomly-spaced $\{x_k\}$

- 2004: there are Lagrange interpolant algorithms
  that are efficient and robust.!

- general form MODIFIED LAGRANGE INTERPOLANT (MLI)

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

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

SFU
muraki

- 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)

  + (adds in sum

  + $(N+1)$ per term)

  $= 2(N+1) + (4N+3)$

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

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

  can - + for same $\{\quad\}$ values !

- as a bonus, if POINT x_{N+1} is ,

  the $\{w_k\}$-values can be in $O(\quad)$ flops

  $\Rightarrow$ NLI has an algorithm !

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

⇒ MLI has a algorithm

- there is another version: BARYCENTRIC FORMULA.

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

↳ version used in from repository

- although slightly in general, has advantages when there are large

- homework + testing will focus on MLI form.

POLYNOMIAL INTERPOLATION for VERY LARGE N

- for uniform spacing on $0 \le x \le 1$, even ML1
  fails for large N because

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

  and the range of to $\{w_k\}$
  reaches 10 around $N=$

- for chosen $\{x_k\}$ the w_j's grow

- BUT, there are special choices of $\{x_k\}$
  where the $\{w_j\}$ do GROW !!
  (example: Chebyshev nodes "cluster" near endpoints)

- WARNING: Matlab's "polyfit" for exact interpolation
  is for large N.

- next homework: evaluating $P_{lag}'(x)$ is $O(\;)$ per point

NEWTON INTERPOLATION ( )

- Newton basis polynomials for $\{\ \}_{k=0\to}$

$$N_0(x) = 1$$
$$N_1(x) = (x- \ )$$
$$\vdots$$
$$N_j(x) = (x-x_0)(x-x_1)\ldots(x- \ \ ) = \prod_{k=0} (x-x_k)$$
$$\vdots$$
$$N_N(x)$$

- key property: $N_j(x_k) = \quad$ for all $0 \le \quad < \quad$

- Newton Interpolating Polynomial

$$P_n(x) = \sum_{k=0}^{N} \quad N_k(x)$$

- TWO IDEAS

1) a_k's satisfy linear system

$$P_n(x_0) = a_0 N_0(x_0) = f_0$$
$$P_n(x_1) = a_0 N_0(x_1) + a_1 N_1(x_1) = f_1$$
$$P_n(x_2) = a_0 N_0(x_2) + a_1 N_1(x_2) + a_2 N(x_2) = f_2$$

- solving for $\{a_k\}$ is $O(\ \ )$

SFU
muraki

2) but you don't use to solve, instead use table

Table 3.9

x	$f(x)$	First divided differences	Second divided differences	Third divided differences
x_0	$f[x_0] = a_0$			
		$f[x_0,x_1] = \dfrac{f[x_1]-f[x_0]}{x_1-x_0} = a_1$		
x_1	$f[x_1]$		$f[x_0,x_1,x_2] = \dfrac{f[x_1,x_2]-f[x_0,x_1]}{x_2-x_0} = a_2$	
		$f[x_1,x_2] = \dfrac{f[x_2]-f[x_1]}{x_2-x_1}$		$f[x_0,x_1,x_2,x_3] = \dfrac{f[x_1,x_2,x_3]-f[x_0,x_1,x_2]}{x_3-x_0} = a_3$
x_2	$f[x_2]$		$f[x_1,x_2,x_3] = \dfrac{f[x_2,x_3]-f[x_1,x_2]}{x_3-x_1}$	
		$f[x_2,x_3] = \dfrac{f[x_3]-f[x_2]}{x_3-x_2}$		$f[x_1,x_2,x_3,x_4] = \dfrac{f[x_2,x_3,x_4]-f[x_1,x_2,x_3]}{x_4-x_1}$
x_3	$f[x_3]$		$f[x_2,x_3,x_4] = \dfrac{f[x_3,x_4]-f[x_2,x_3]}{x_4-x_2}$	
		$f[x_3,x_4] = \dfrac{f[x_4]-f[x_3]}{x_4-x_3}$		$f[x_2,x_3,x_4,x_5] = \dfrac{f[x_3,x_4,x_5]-f[x_2,x_3,x_4]}{x_5-x_2}$
x_4	$f[x_4]$		$f[x_3,x_4,x_5] = \dfrac{f[x_4,x_5]-f[x_3,x_4]}{x_5-x_3}$	
		$f[x_4,x_5] = \dfrac{f[x_5]-f[x_4]}{x_5-x_4}$		
x_5	$f[x_5]$			

- obtaining { } is O( ), but involves the { } values

  so, NLI is superior since $O(N^2)$ does need { } & therefore allows

- BUT what do you do for sets of data points ?

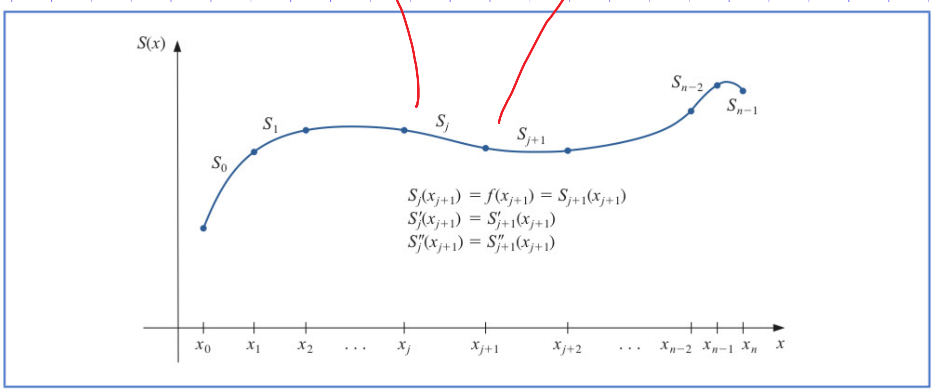
  (or in more ?)

CUBIC SPLINES (§3.5)

- NLI eventually fails if there are points, the basic problem is that asking FUNCTION to exactly go through points is too much.

- NEW IDEA – – POLYNOMIAL INTERPOLATION

  use a different in each interval



Definition 3.10 Given a function f defined on $[a, b]$ and a set of nodes $a = x_0 < x_1 < \cdots < x_n = b$, a **cubic spline interpolant** S for f is a function that satisfies the following conditions:

A natural spline has no conditions imposed for the direction at its endpoints, so the curve takes the shape of a straight line after it passes through the interpolation points nearest its endpoints. The name derives from the fact that this is the natural shape a flexible strip assumes if forced to pass through specified interpolation points with no additional constraints. (See Figure 3.9.)

(a) $S(x)$ is a cubic polynomial, denoted $S_j(x)$, on the subinterval $[x_j, x_{j+1}]$ for each $j = 0, 1, \ldots, n-1$;

(b) $S_j(x_j) = f(x_j)$ and $S_j(x_{j+1}) = f(x_{j+1})$ for each $j = 0, 1, \ldots, n-1$;

(c) $S_{j+1}(x_{j+1}) = S_j(x_{j+1})$ for each $j = 0, 1, \ldots, n-2$; (*Implied by* **(b)**.)

(d) $S'_{j+1}(x_{j+1}) = S'_j(x_{j+1})$ for each $j = 0, 1, \ldots, n-2$;

(e) $S''_{j+1}(x_{j+1}) = S''_j(x_{j+1})$ for each $j = 0, 1, \ldots, n-2$;

(f) One of the following sets of boundary conditions is satisfied:

  (i) $S''(x_0) = S''(x_n) = 0$ (**natural** *(or free)* **boundary**);

  (ii) $S'(x_0) = f'(x_0)$ and $S'(x_n) = f'(x_n)$ (**clamped boundary**). ■

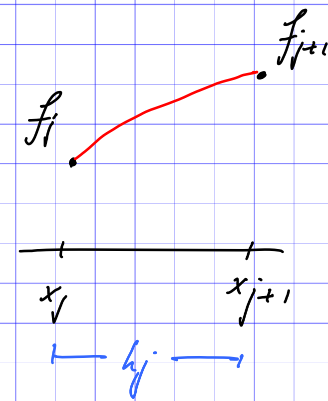
- each SPLINE $S_j(x)$ determined by values

  and continuity of

SFU

muraki

- consider one interval



- left-endpoint polynomial

$$S_j(x) = \underset{f_j}{\underline{\quad}} + \underline{\quad}(x-x_j) + \underline{\quad}(x-x_j) + \underline{\quad}(x-x_j)$$

a) $S_j(x_{j+1}) = f_j + b_j \cdot h_j + c_j h_j^2 + d_j h_j^3 = f_{j+1}$

b) $S_j'(x_{j+1}) = b_j + 2c_j h_j + 3d_j h_j^2 = (x_j)$

$=$

c) $S_j''(x_{j+1}) = 2c_j + 6d_j h_j = (x_j)$

$=$

- eliminate d_j using $d_j h_j = \frac{1}{3}(c_{j+1} - c_j)$ (eqn 3.17)

b') $b_{j+1} - b_j = (c_{j+1} + c_j) h_j$ (eqn. 3.19)

a') $f_{j+1} = f_j + b_j h_j + \frac{1}{3}(2c_j + c_{j+1}) h_j^2$ (eqn 3.18)

$f_j = f_{j-1} + b_{j-1} h_{j-1} + \frac{1}{3}(2c_{j-1} + c_j) h_{j-1}^2$

SFU
muraki

MAIN CUBIC SPLINE EQUATION

$f_{j+1} - f_j$ $\quad$ $f_j - f_{j-1}$

c')
$$\underbrace{h_{j-1}c_{j-1}}_{?} + 2(h_{j-1}+h_j)\underbrace{c_j}_{?} + h_j\underbrace{c_{j+1}}_{?} = \frac{3}{h_j}(a_{j+1}-a_j) - \frac{3}{h_{j-1}}(a_j - a_{j-1}), \qquad (3.21)$$

unknowns,

$j = 1 \to N-2 \quad \to \quad N-2$ equations muraki