DS 288: NUMERICAL METHODS

SEP-7-2021

ITERATIVE INTERPOLATION

- USE LOWER ORDER POLYNOMIALS TO FORM HIGHER ORDER POLYNOMIALS

$P_0(x) = f(x_0)\, b_0(x) = f(x_0)$

$b \rightarrow$ LAGRANGE POLYNOMIAL

$\rightarrow$ NOT ORDER

$$P_{0,1}(x) = f(x_0)\, b_0(x) + f(x_1)\, b_1(x)$$

$$= f(x_0)\,\frac{(x-x_1)}{x_0-x_1} + f(x_1)\,\frac{(x-x_0)}{x_1-x_0}$$

$$P_{0,1}(x) = \frac{P_1(x)(x-x_0) - P_0(x)(x-x_1)}{x_1-x_0}$$

$\hookrightarrow$ NEVILLE'S METHOD

NOW LOOK AT

$$P_{0,1,2}(x) = f(x_0)\,b_0(x) + f(x_1)\,b_1(x) + f(x_2)\,b_2(x)$$

WRITE OUT POLYNOMIALS

$$= f(x_0)\frac{(x-x_1)(x-x_2)}{(x_0-x_1)(x_0-x_2)} + f(x_1)\frac{(x-x_0)(x-x_2)}{(x_1-x_0)(x_1-x_2)} + f(x_2)\frac{(x-x_0)(x-x_1)}{(x_2-x_0)(x_2-x_1)}$$

[EXERCISE] CAN SHOW THAT

$$P_{0,1,2}(x) = \frac{P_{1,2}(x)(x-x_0) - P_{0,1}(x)(x-x_2)}{(x_2-x_0)}$$

ITERATE TO GET HIGHER ORDER POLYNOMIALS

$$P_{1,2}(x) = \frac{f(x_1)(x-x_2)}{(x_1-x_2)} + \frac{f(x_2)(x-x_1)}{(x_2-x_1)}$$

NEUILLE'S METHOD SCHEMATIC

[[$P \neq 1$ IN HW#3]]

$f(x_0)$ = P_0

x_0 P_0 $P_{0,1}$

x_1 P_1 $P_{0,1,2}$ $P_{0,1,2,3}$

$P_{1,2}$

x_2 P_2 $P_{1,2,3}$ $P_{0,1,2,3,4}$

$P_{2,3}$ $P_{1,2,3,4}$

x_3 P_3 $P_{2,3,4}$

$P_{3,4}$

x_4 P_4

↑ CUBIC ↑ FOURTH ORDER

↑ QUADRATIC POLY

↑ SCALAR ↑ LINEAR POLYNOMIAL

HIGHER ORDER POLYNOMIAL →

DIVIDED DIFFERENCE [§3.3]

NOTATION THAT ALLOWS US TO CONSTRUCT A TABLE CONTAINING THE COEFFICIENTS OF A POLYNOMIAL

$$P_N(x) = a_0 + a_1(x - x_0) + a_2(x-x_0)(x-x_1) + \dots + a_N(x-x_0)(x-x_1)\cdots(x-x_N)$$

$\{a_i\}$ coefficients OF A LAGRANGE POLYNOMIAL

ZEROTH DIVIDED DIFFERENCE]: $f[x_i] = f(x_i)$

FIRST DIVIDED DIFFERENCE : $f[x_i, x_{i+1}]$

$$= \frac{f[x_{i+1}] - f[x_i]}{x_{i+1} - x_i}$$

SECOND DIVIDED DIFFERENCE

$$f[x_i, x_{i+1}, x_{i+2}] = \frac{f[x_{i+1}, x_{i+2}] - f[x_i, x_{i+1}]}{x_{i+2} - x_i}$$

NOTE:

$$P_N(x_0) = a_0 = f[x_0]$$

$$\Rightarrow a_0 = f[x_0]$$

$$P_N(x_1) = a_0 + a_1(x_1 - x_0) = f[x_1]$$

$$f[x_0] + a_1(x_1 - x_0) = f[x_1]$$

$$a_1 = \frac{f[x_1] - f[x_0]}{x_1 - x_0}$$

$$\Rightarrow a_1 = f[x_0, x_1]$$

IN GENERAL

$$a_k = f[x_0, x_1, \ldots x_k]$$

K^{TH} DIVIDED DIFFERENCE

* SIMILAR NEVILLE'S METHOD (EXCEPT WE DO NOT USE LOWER ORDER POLYNOMIALS)

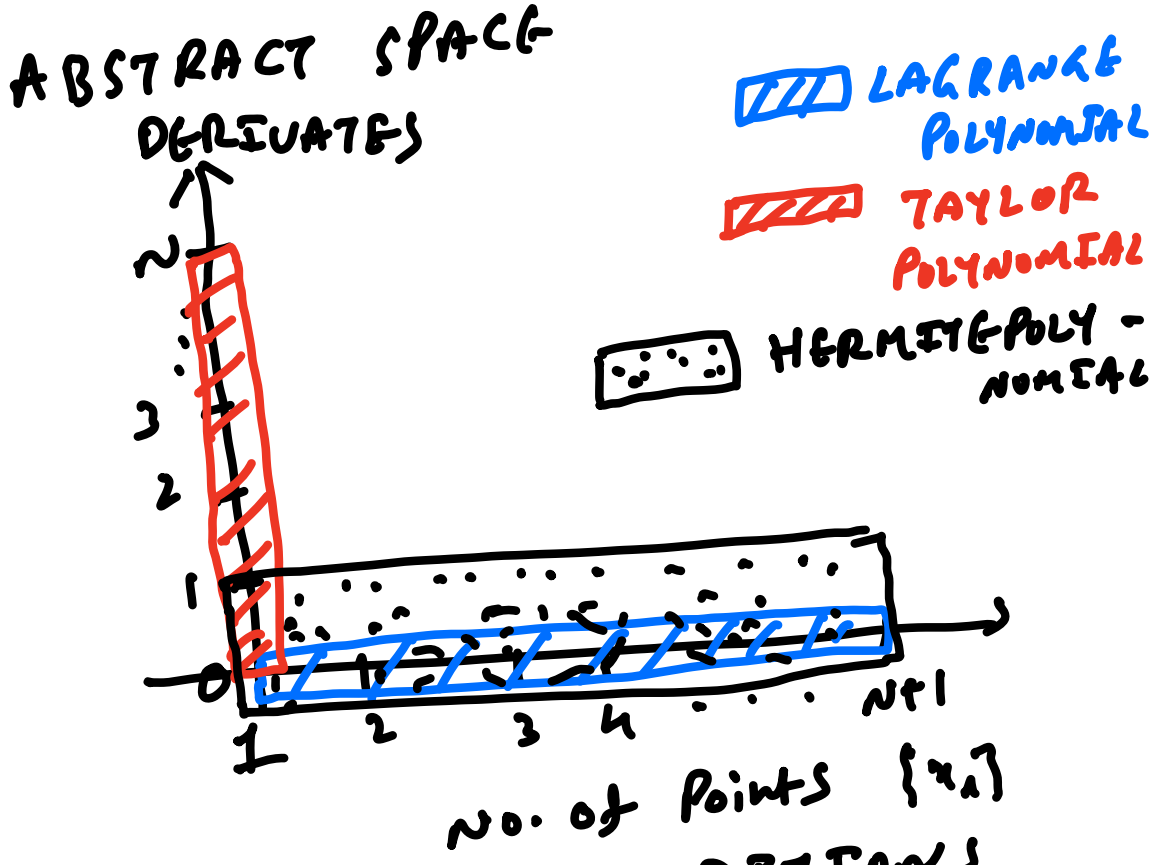
INTRODUCE ANOTHER TYPE POLYNOMIAL

N+1 → TABLUATED VALUES $\{x_i, f(x_i)\}$

MAXIMUM ORDER OF POLYNOMIAL - 'N'

HERMITE POLYNOMIAL

- MATCHES f & f' AT N+1 POINTS $\{x_i\}$



* $2(N+1) = 2N+2$ CONDITIONS

$P_N(x_j) = f(x_j)$ & $P_N'(x_j) = f'(x_j)$

$j = 0, 1, 2, \ldots N$

UNIQUE POLYNOMIAL OF ORDER

: $2N+1$

WRITE IN TERMS OF BASIS

$$P_{2N+1}(x) = \sum_{j=0}^{N+1} f(x_j) H_{2N+1,j}(x) + \sum_{j=0}^{N+1} f'(x_j) \hat{H}_{2N+1,j}(x)$$

WITH PROPERTIES

$$H_{2N+1,j}(x_i) = \delta_{ij} \qquad \hat{H}_{2N+1,j}(x_i) = 0$$

$$H'_{2N+1,j}(x_i) = 0 \qquad \hat{H}'_{2N+1,j}(x_i) = \delta_{ij}$$

$$\Rightarrow P_{2N+1,j}(x_i) = f(x_i) \quad \& \quad P'_{2N+1,j}(x_i) = f'(x_i)$$

CAN WRITE IN TERMS OF LAGRANGE'S POLYNOMIALS

(SHOWN IN TEXT).

HERMITE INTERPOLATING POLYNOMIALS HANDOUT

Hermite Interpolating Polynomials

Definition:

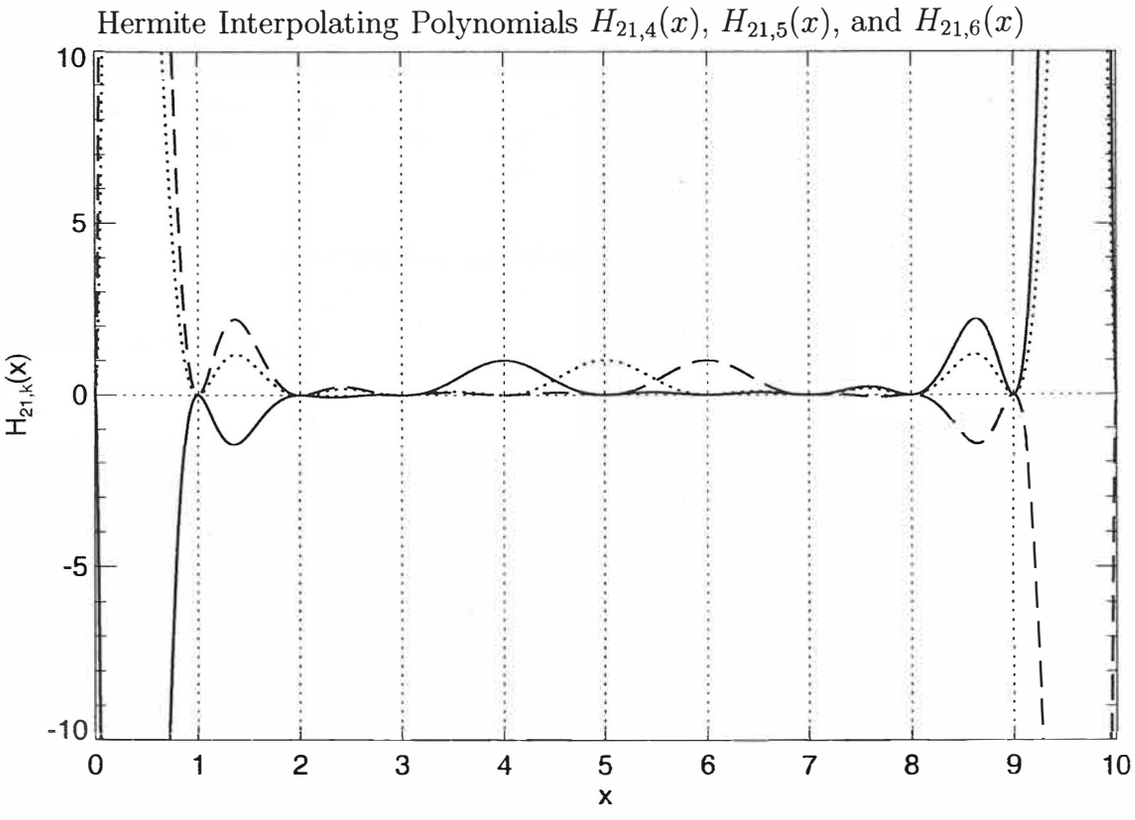
$$P_{2N+1}(x) = \sum_{j=0}^{N} f(x_j) H_{2N+1,j}(x) + \sum_{j=0}^{N} f'(x_j) \hat{H}_{2N+1,j}(x)$$

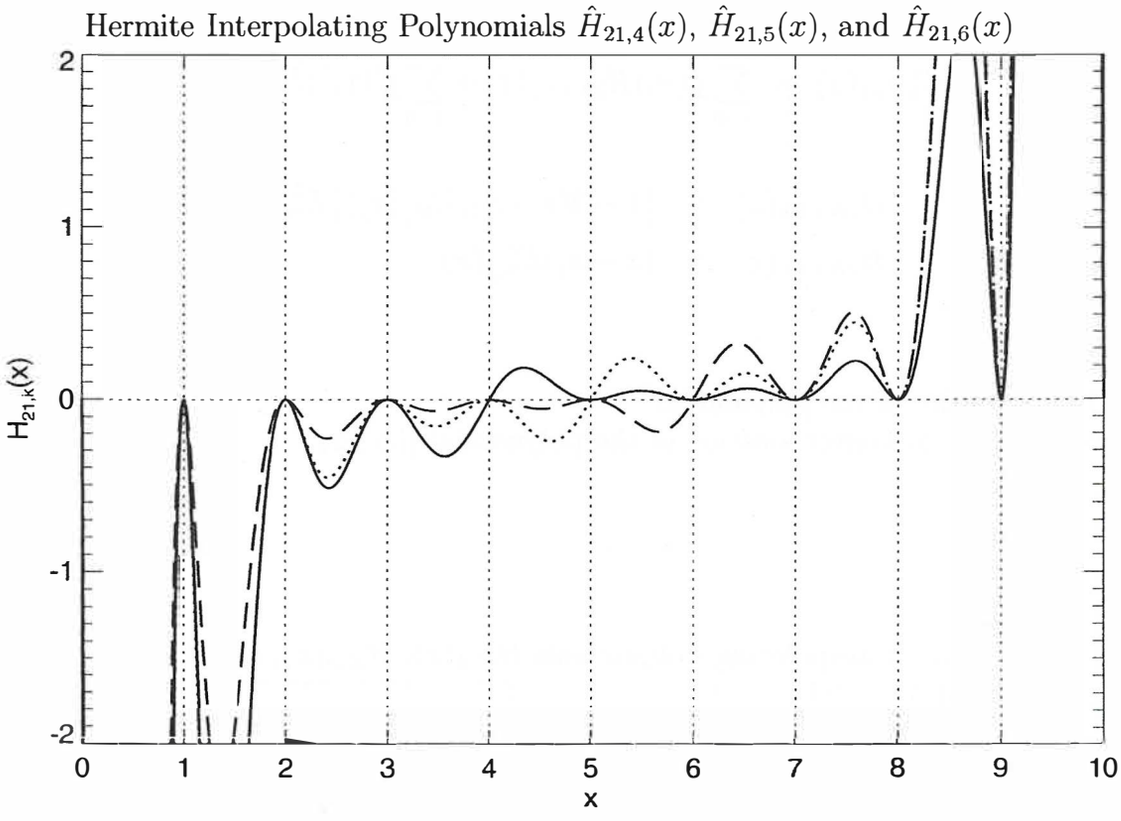
$$H_{2N+1,j}(x) = \left[1 - 2(x - x_j) L'_{N,j}(x_j)\right] L^2_{N,j}(x)$$
$$\hat{H}_{2N+1,j}(x) = (x - x_j) L^2_{N,j}(x)$$

IN TERMS OF LAGRANGE POLYNOMIAL (L).

$2N + 1$ is the order of the polynomial
j is the index of the center location of the polynomial (i.e., x_j)

Example:





Note the desirable behavior of Hermite interpolating polynomials:

$$H_{2N+1,k}(x_i) = \delta_{ik} \equiv \begin{cases} 1 & i = k \\ 0 & i \neq k \end{cases} \qquad H'_{2N+1,k}(x_i) = 0$$

$$\hat{H}_{2N+1,k}(x_i) = 0 \qquad \hat{H}'_{2N+1,k}(x_i) = \delta_{ik}$$

Error term:

$$f(x) - P_{2N+1}(x) = \frac{f^{(2N+2)}(\xi)}{(2N+2)!} \prod_{i=0}^{N} (x - x_i)^2 \qquad \xi \in [x_0, x_N]$$

BETTER CONVERGENCE

CAVEAT: YOU REQUIRE $f(x_i)$'s & $f'(x_i)$'s.

Courtesy: Prof. Simon Shepard

UP TO THIS POINT, WE HAVE BEEN PERFORMING GLOBAL POLYNOMIAL INTERPOLATION.

- SINGLE POLYNOMIAL TO APPROXIMATE $f(x)$ $x \in [a,b]$
- ADVANTAGES
  - STORAGE, COMPUTATIONS, ETC.

- HAND OUT
  GLOBAL INTERPOLATING POLYNOMIAL

Interpolation: Global Polynomial Fitting

From Text Book: Page-158

Figure 3.11

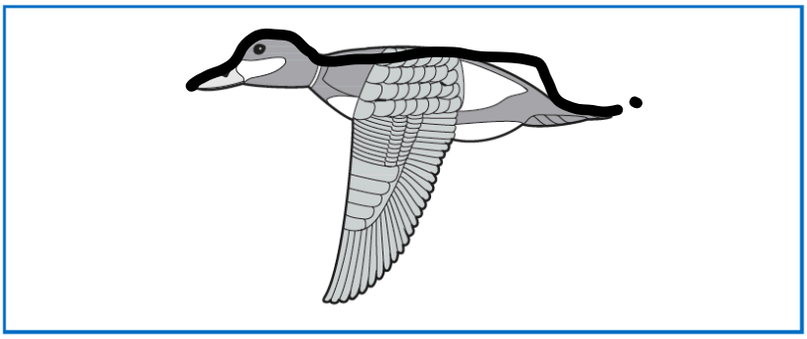
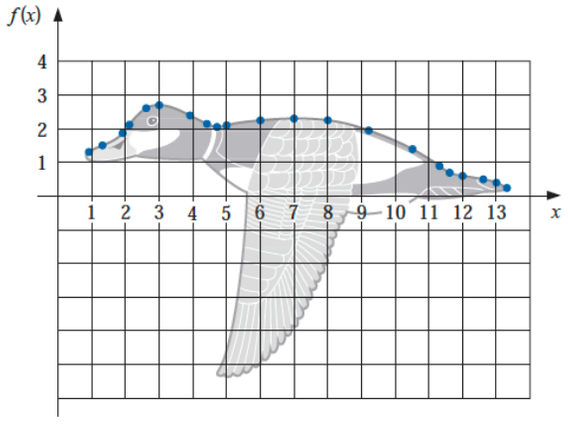


Table 3.18

x	0.9	1.3	1.9	2.1	2.6	3.0	3.9	4.4	4.7	5.0	6.0	7.0	8.0	9.2	10.5	11.3	11.6	12.0	12.6	13.0	13.3
$f(x)$	1.3	1.5	1.85	2.1	2.6	2.7	2.4	2.15	2.05	2.1	2.25	2.3	2.25	1.95	1.4	0.9	0.7	0.6	0.5	0.4	0.25

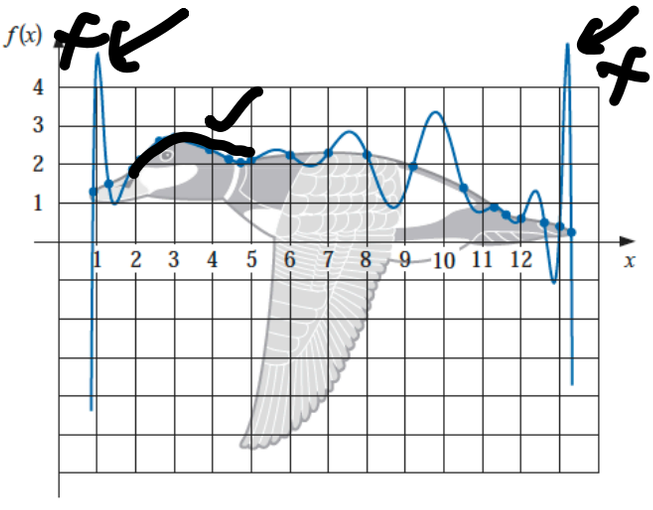
Data Points: 21 ✔

Figure 3.12



Lagrange Interpolating Polynomial: Degree – 20 ✔

Figure 3.14



CHEBYSHEV SAMPLING HELPS

(SAMPLING MAY NOT BE POSSIBLE)

ANOTHER POSSIBILITY IS TO USE

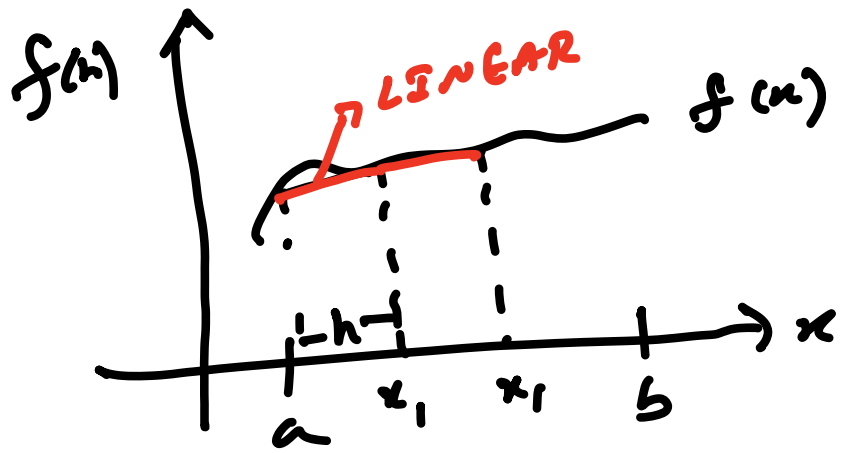
PIECEWISE POLYNOMIALS

DIVIDE $[a, b]$ INTO SEGMENTS

$[x_i, x_{i+1}]$

FOR EACH i in $[x_i, x_{i+1}]$ → WE FIT A POLYNOMIAL.

EASY ONE: LINEAR POLYNOMIAL FOR EACH SEGMENT.



SIMPLE: $|P(x) - f(x)| \sim \Theta(h^2)$

$h = x_{j+1} - x_j$

$$f(x) - P_1(x) = \frac{f''(\xi)}{2!} \underbrace{(x-x_0)(x-x_1)}_{O(h^2)}$$

MUST HAVE 'h' → SMALL
FOR SMALLER ERROR.

NO. OF SEGMENTS → INCREASE.
↙
MORE STORAGE

ALSO $P'(x_i)$ IS DISCONTINUOUS
FOR $\{x_i\}$ [NOT DESIRABLE]
↙ SUCCESSIVE
ACROSS × SEGMENTS

HERMITE CUBIC FOR LOCAL INTERPOLATION

$N=1$ $\quad 2N+1=3$

x_j $\quad x_{j+1}$

REQUIRES $f(x_j)$, $f'(x_j)$, $f(x_{j+1})$, $f'(x_{j+1})$

CONTINUOUS FIRST DERIVATIVE

BUT NEED TO KNOW $f'(x)$

AT LEAST $f'(x_i)$'s for

$i = 0, 1, 2, \ldots N$

A USEFUL COMPROMISE IS TO

USE A SPLINE

- DON'T INSIST THAT DERIVATIVES MATCH FUNCTION

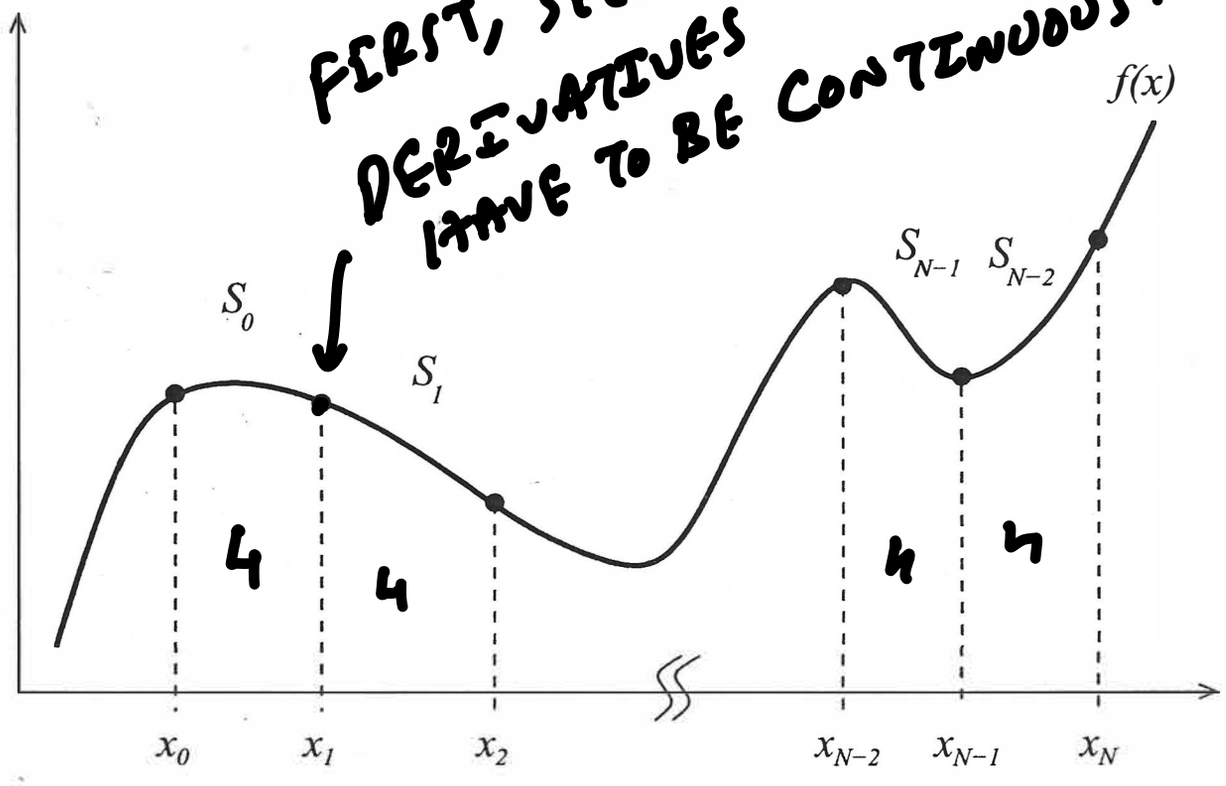
*ONLY THAT DERIVATIVES ARE CONTINUOUS.

CUBIC SPLINE [§ 3.5]

-USE CUBIC POLYNOMIAL ON EACH INTERVAL

-MAKE FIRST & SECOND ORDER DERIVATIVES CONTINUOUS ACROSS INTERVALS [HANDOUT]

Cubic Spline Interpolation



FIRST, SECOND DERIVATIVES HAVE TO BE CONTINUOUS

$$S_j(x) = a_j + b_j(x - x_j) + c_j(x - x_j)^2 + d_j(x - x_j)^3 \quad x \in [x_j, x_{j+1}$$

FUNCTION MATCHING

Constraints:	(i)	$S_j(x_j) = f(x_j)$	$j = 0, 1, \ldots N-1$	$\longrightarrow N$	conditions
	(ii)	$S_j(x_{j+1}) = f(x_{j+1})$	$j = 0, 1, \ldots N-1$	$\longrightarrow N$	conditions
	(iii)	$S'_j(x_{j+1}) = S'_{j+1}(x_{j+1})$	$j = 0, 1, \ldots N-2$	$\longrightarrow N-1$	conditions
	(iv)	$S''_j(x_{j+1}) = S''_{j+1}(x_{j+1})$	$j = 0, 1, \ldots N-2$	$\longrightarrow N-1$	conditions

These constraints total $4N - 2$ conditions. We need $4N$ to get a unique solution.

The solution is to impose one condition at each endpoint, x_0 and x_N.

NO DERIVATIVES AT x_0 & x_N

Common strategies:

(a)	$S'_0(x_0) = f'(x_0);\ S'_{N-1}(x_N) = f'(x_N)$	"Clamped Spline"
(b)	$S''_0(x_0) = S''_{N-1}(x_N) = 0$	"Natural Spline"
(c)	$S'''_0(x_1) = S'''_1(x_1);\ S'''_{N-1}(x_{N-1}) = S'''_{N-2}(x_{N-1})$	Other

→ LEADS TO MOST ACCURATE ESTIMATION BUT REQUIRES f' AT END POINTS

These constraints plus "end conditions" lead to a set of relationships which can be solved for the sets of $\{a_j\}$, $\{b_j\}$, $\{c_j\}$, and $\{d_j\}$:

- From (i): $S_j(x_j) = \boxed{f(x_j) = a_j}$ $\quad j = 0, 1, ...N-1$
- From (ii): $S_j(x_{j+1}) = f(x_{j+1}) = \boxed{a_j + b_j h_j + c_j h_j^2 + d_j h_j^3 = a_{j+1}}$ $\quad j = 0, 1, ...N-1$

  where $h_j \equiv x_{j+1} - x_j$
- From (iii): $S_j'(x_{j+1}) = S_{j+1}'(x_{j+1})$ $\quad$ where $S_j'(x) = b_j + 2c_j(x - x_j) + 3d_j(x - x_j)^2$

  $\Rightarrow \boxed{b_j + 2c_j h_j + 3d_j h_j^2 = b_{j+1}}$ $\quad j = 1, 2, ...N-2$
- From (iv): $S_j''(x_{j+1}) = S_{j+1}''(x_{j+1})$ $\quad$ where $S_j''(x) = 2c_j + 6d_j(x - x_j)$

  $\Rightarrow \boxed{2c_j + 6d_j h_j = 2c_{j+1}}$ $\quad j = 1, 2, ...N-2$

We can combine these four equations into a single equation involving on the set of $\{c_j\}$ and known quantities by various substitutions. Solve (iv) for d_j and plug into (ii), solve (ii) for b_j and plug into (iii).

$$h_{j-1}c_{j-1} + 2(h_{j-1} + h_j)c_j + h_j c_{j+1} = \frac{3}{h_j}(a_{j+1} - a_j) - \frac{3}{h_{j-1}}(a_j - a_{j-1})$$

holds for $j = 1, 2, ...N-1$, i.e., this is a system of $N-1$ equations involving $N+1$ $\{c_j\}$... must apply "end constraints" to determine a unique solution.

e.g., Natural Spline: $S_0''(x_0) = 0$; $S_N''(x_N) = S_{N-1}''(x_N) = 0$ by (iv)

$$S_j'' = 2c_j + 6d_j(x - x_j) \quad \Rightarrow \quad S_0''(x_0) = 2c_0 = 0$$
$$S_N''(x_N) = 2c_N = 0$$

In this case, $c_0 = c_N = 0$.

Now write as a matrix system: $\underline{\underline{A}} \cdot \underline{x} = \underline{b}$ ... (each j consitutes a row in a matrix).

$$\begin{bmatrix} 1 & 0 & 0 & 0 & \dots & 0 \\ h_0 & 2(h_0+h_1) & h_1 & 0 & \dots & 0 \\ 0 & h_1 & 2(h_1+h_2) & h_2 & \dots & 0 \\ \vdots & \vdots & & \ddots & & \vdots \\ 0 & 0 & \dots & h_{N-2} & 2(h_{N-2}+h_{N-1}) & h_{N-1} \\ 0 & 0 & \dots & 0 & 0 & 1 \end{bmatrix} \cdot \begin{pmatrix} c_0 \\ c_1 \\ c_2 \\ \vdots \\ c_{N-1} \\ c_N \end{pmatrix} = \begin{pmatrix} 0 \\ \frac{3}{h_1}(a_2 - a_1) - \frac{3}{h_0}(a_1 - a_0) \\ \frac{3}{h_2}(a_3 - a_2) - \frac{3}{h_1}(a_2 - a_1) \\ \vdots \\ \frac{3}{h_{N-1}}(a_N - a_{N-1}) - \frac{3}{h_{N-2}}(a_{N-1} - a_{N-} \\ 0 \end{pmatrix}$$

4N x 4N $\quad$ 4N x 1 $\quad$ 4N x 1

TRIDIAGONAL MATRIX

THOMAS ALGORITHM : O(4N) COMPUTATIONS

TEXT §3.6 TALKS ABOUT PARAMETRIC CURVES.

↳ WHICH ARE USEFUL FOR APPROXIMATING NON-FUNCTIONS

* PRACTISE IN HW #3 *

— END OF CH #3 —