Numerical Analysis

Interpolation and Polynomial Approximation

OL Say

ol.say@itc.edu.kh

Institute of Technology of Cambodia

March 9, 2023

Outline

1. Taylor Polynomial
2. Polynomial Determination
3. Polynomial Evaluation
4. Lagrange Interpolation
5. Newton's Method
6. Neville's Method
7. Cubic Spline Interpolation

1. Many problems arising in engineering require evaluation of real-valued functions.
2. Evaluation of some of the commonly used functions, such as $\cos(x)$, on a computer usually utilizes the so-called lookup tables, which store a finite set of numbers describing the function of interest in a way that allows one to approximate the value of $f(x)$ for any given x with a high level of accuracy.
3. For example, a finite set of coefficients in the power series decomposition of $f(x)$ could be used for this purpose.

Example 1

Consider $f(x) = \cos x, x \in [-1, 1]$. The Taylor series approximation of this function about the origin is

$$\cos x = 1 - \frac{x^2}{2!} + \frac{x^4}{4!} - \frac{x^6}{6!} + \cdots + (-1)^n \frac{x^{2n}}{(2n)!} + \cdots ,$$

and for a given n, the set of $2n + 1$ coefficients

$$1, 0, -\frac{1}{2!}, 0, \frac{1}{4!}, 0, \ldots, (-1)^n \frac{1}{(2n)!}$$

could be used as the table representing $f(x)$ for the values of x *close to* 0. Given such a table, the approximation to $f(x)$ is computed as

$$\cos x \approx 1 - \frac{x^2}{2!} + \frac{x^4}{4!} - \frac{x^6}{6!} + \cdots + (-1)^n \frac{x^{2n}}{(2n)!}$$

In this formula, a better approximation is obtained for a larger n.

Table: Approximation of $\cos(0.01) = 0.99995000041666653$ by Taylor Polynomial of Degree n

n	Taylor	Error
0	1.00000000000000000	0.00004999995833347
1	0.99995000000000000	0.00000000004166653
2	0.99995000004166667	0.00000000000000014
3	0.99995000004166653	0.00000000000000000

1. However, in many practical situations, the function $f(x)$ of interest is not known explicitly,
2. and the only available information about the function is a table of measured or computed values

$$y_0 = f(x_0), y_1 = f(x_1), \ldots, y_n = f(x_n),$$

   for a set of points $x_0 < x_1 < \cdots < x_n$ in some interval $[a, b]$:
3. Given such a table, our goal is to approximate $f(x)$ with a function, say a polynomial $p(x)$, such that

$$p(x_i) = f(x_i), i = 0, 1, \ldots, n.$$

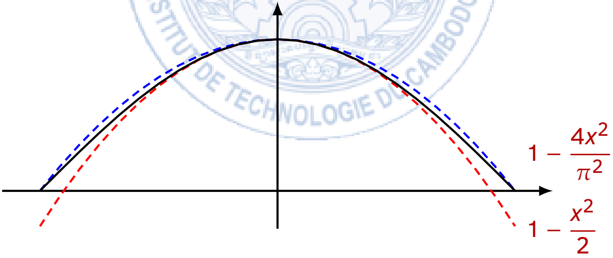
4. Then for $x \in [a, b]$ we can approximate the value $f(x)$ with $p(x)$.
5. If $x_0 < x < x_n$, the approximation $p(x)$ is called an *interpolated value,*
6. otherwise it is an *extrapolated value.*

Example 2

1. Taylor polynomial $p(x) = 1 - \dfrac{x^2}{2}$ of degree 2
2. and the quadratic polynomial $q(x) = 1 - \dfrac{4}{\pi^2}x^2$ passing through points

$-\pi/2$	0	$\pi/2$
0	1	0

both approximate $f(x) = \cos(x)$.



Polynomial Determination

Given $n+1$ pairs $(x_0, y_0), \dots, (x_n, y_n)$ of numbers, where $x_i \neq x_j$ if $i \neq j$, our goal is to find a polynomial $p(x)$ such that $P(x_i) = y_i, i = 0, \dots, n$.

Depending on the objective, the same polynomial can be represented in several different forms. A polynomial $p(x)$ in the

1. power form: $p(x) = a_0 + a_1 x + \cdots + a_n x^n$
2. shifted power form: $p(x) = b_0 + b_1(x - x_0) + \cdots + b_n(x - x_0)^n$
3. Newton form:
   $p(x) = c_0 + c_1(x - x_0) + c_2(x - x_0)(x - x_1) + \cdots + c_n(x - x_0)\cdots(x - x_{n-1})$
4. Lagrange form: $p(x) = p(x_0)L_{n,0}(x) + \cdots p(x_n)L_{n,n}(x)$ where $x_0, \dots, x_n$ are $n+1$ pairwise distinct nodes and $L_{n,i}$ be polynomial of degree n satisfying $L_{n,i}(x_j) = \begin{cases} 0, & i \neq j \\ 1, & i = j \end{cases}$.

Example 3

Our goal is to find a polynomial of degree at most 3 say $P(x)$ passing through the points

x_i	0	1	2	3
y_i	3	6	11	18

.

1. Determine the polynomial using the method of undetermined coefficients.
2. Determine the polynomial using Lagrange's method.
3. Determine the polynomial using Newton's method with $x_0 = 1, x_1 = 2, x_2 = 3, x_3 = 4$.
4. Write computer programming code to compute $P(5)$ using Lagrange's and Newton's methods.

- To satisfy $L_{n,j}(x_i) = 0$ for each $i \neq j$, it is required $L_{n,j}(x) = k(x - x_0)\cdots(x - x_{j-1})(x - x_{j+1})\cdots(x - x_n)$.
- To satisfy $L_{n,j}(x_i) = 1$ for each $i = j$, it is required $k = [(x_k - x_0)\cdots(x_k - x_{j-1})(x_k - x_{j+1})\cdots(x_k - x_n)]^{-1}$.

Evaluate a polynomial $P(x)$ at x_0 with the curve of P passing through the set of points $(x_1, y_1), (x_2, y_2), \dots, (x_n, y_n)$.
INPUT: Data set $(x_1, y_1), (x_2, y_2), \dots, (x_n, y_n)$. and x_0.
OUTPUT: $P(x_0)$

1. Initialize $P = 0$.
   a. For i from 1 to n do
   b. Set $L = y_i$
      i. For k from 1 to n do
         If $k \neq i$ then set $L = L * (x_0 - x_k)/(x_i - x_k)$
   c. Set $P = P + L$
2. OUTPUT P.

The Newton form can be written as nested form

$$p(x) = a_0 + (x - x_0)[a_1 + (x - x_1)[a_2 + \cdots [a_{n-1} + (x - x_{n-1})a_n] \cdots]]$$

Evaluating a polynomial $p(x)$ in the Newton form.

INPUT $a_0, a_1, \ldots, a_n; x_0, x_1, \ldots, x_{n-1}; x$

OUTPUT $p(x) = a_0 + \sum_{k=1}^{n} a_k \prod_{i=0}^{k-1}(x - x_i)$

1. Set $p_n = a_n$
2. For i from $n-1$ to 0 do

   $p_i = a_i + (x - x_i)p_{i+1}$
3. OUTPUT p_0

Theorem 4 ((Weierstrass Approximation Theorem)

Suppose that f is defined and continuous on $[a, b]$. For each $\varepsilon > 0$, there exists a polynomial $P(x)$, with the property that
$|f(x) - P(x)| < \varepsilon,$ *for all x in $[a, b]$.*

Theorem 5 (The n-th Lagrange Interpolating Polynomial)

If $x_0, x_1, \ldots, x_n$ are $n+1$ distinct numbers and f is a function whose values are given at these numbers, then a unique polynomial $P(x)$ of degree at most n exists with

$$f(x_k) = P(x_k), \text{ for each } k = 0, 1, \ldots, n.$$

This polynomial is given by

$$P(x) = f(x_0)L_{n,0}(x) + \cdots + f(x_n)L_{n,n}(x) = \sum_{k=0}^{n} f(x_k)L_{n,k}(x),$$

where, for each $k = 0, 1, \ldots, n$,

$$L_{n,k}(x) = \frac{(x-x_0)(x-x_1)\cdots(x-x_{k-1})(x-x_{k+1})\cdots(x-x_n)}{(x_k-x_0)(x_k-x_1)\cdots(x_k-x_{k-1})(x_k-x_{k+1})\cdots(x-x_n)}$$

Theorem 6

Suppose $x_0, x_1, \ldots, x_n$ 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

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

where $P(x)$ is the n-th Lagrange interpolating polynomial.

Example 7

1. Use the numbers (nodes) $x_0 = 2, x_1 = 2.75$, and $x_2 = 4$ to find the second Lagrange interpolating polynomial for $f(x) = 1/x$.
2. Using this polynomial to approximate $f(3) = 1/3$.
3. Determine the error form for this polynomial, and the maximum error when the polynomial is used to approximate $f(x)$ for $x \in [2, 4]$.

1. $P(x) = \frac{1}{22}x^2 - \frac{35}{88}x + \frac{49}{44}$
2. $P(3) = \frac{29}{88} \approx 0.32955$
3. The maximum error is

$$\begin{aligned}\left|\frac{f'''(\xi(x))}{3!}\right| \cdot |(x-x_0)(x-x_1)(x-x_2)| &= |-[\xi(x)]^{-4}| \cdot |(x-2)(x-2.75)(x-4)| \\ &\leq |-2^{-4}| \cdot |-9/16| = 2/512 \approx 0.00586.\end{aligned}$$

1. Suppose that $P_n(x)$ is the n-th Lagrange polynomial that agrees with the function f at the distinct numbers $x_0, x_1, \ldots, x_n$.
2. Although this polynomial is unique, there are alternate algebraic representations that are useful in certain situations.
3. The divided differences of f with respect to $x_0, x_1, \ldots, x_n$. are used to express $P_n(x)$ in the form

$$P_n(x) = a_0 + a_1(x - x_0) + \cdots + a_n(x - x_0)\cdots(x - x_{n-1}),$$

   for appropriate constants $a_0, a_1, \cdots, a_n$.
4. $a_0 = P_n(x_0) = f(x_0)$
5. $a_1 = \dfrac{P_n(x_1) - a_0}{x_1 - x_0} = \dfrac{f(x_1) - f(x_0)}{x_1 - x_0}$
6. $a_2 = \dfrac{P(x_2) - a_0 - a_1(x_2 - x_0)}{(x_2 - x_0)(x_2 - x_1)} = \dfrac{f(x_2) - f(x_0)}{x_2 - x_0} - \dfrac{f(x_1) - f(x_0)}{x_1 - x_0}$
7. To find the formula a_k, we introduce the following notations:

- a The *zeroth divided difference* of the function f with respect to x_i, denoted $f[x_i]$, is simply the value of f at $x_i : f[x_i] = f(x_i)$.
- b The *first divided difference* of f with respect to x_i and x_{i+1} is denoted $f[x_i, x_{i+1}]$ and defined as $f[x_i, x_{i+1}] = \dfrac{f[x_{i+1}] - f[x_i]}{x_{i+1} - x_i}$.
- c The *second divided difference*, $f[x_i, x_{i+1}, x_{i+2}]$, is defined as $f[x_i, x_{i+1}, x_{i+2}] = \dfrac{f[x_{i+1}, x_{i+2}] - f[x_i, x_{i+1}]}{x_{i+2} - x_i}$.
- d The process ends with the single *nth divided difference*,

$$f[x_0, x_1, \ldots, x_n] = \frac{f[x_1, x_2, \ldots, x_n] - f[x_0, x_1, \ldots, x_{n-1}]}{x_n - x_0}$$

From the notations, we have

$$\begin{aligned} P_n(x) = f[x_0] &+ f[x_0, x_1](x - x_0) + f[x_0, x_1, x_2](x - x_0)(x - x_1) \\ &+ a_3(x - x_0)(x - x_1)(x - x_2) + \cdots \\ &+ a_n(x - x_0)(x - x_1)\cdots(x - x_{n-1}) \end{aligned}$$

To obtain a_3, set $x = x_3$:

$$\begin{aligned} P_n(x_2) &= f[x_0] + f[x_0, x_1](x_2 - x_0) + f[x_0, x_1, x_2](x_2 - x_0)(x_2 - x_1) \\ P_n(x_3) &= f[x_0] + f[x_0, x_1](x_3 - x_0) + f[x_0, x_1, x_2](x_3 - x_0)(x_3 - x_1) \\ &\quad + a_3(x_3 - x_0)(x_3 - x_1)(x_3 - x_2) \end{aligned}$$

As expected it has been verified that

$$a_k = f[x_0, x_1, \ldots, x_k]; k = 0, 1, \ldots, n.$$

So, we obtain the Newton's Divided-Difference Formula:

$$P_n(x) = f[x_0] + \sum_{k=1}^{n} f[x_0, x_1, \ldots, x_k](x - x_0)(x - x_1)\cdots(x - x_{k-1}).$$

Table: Newton's Forward Divided Difference

x	$f(x)$	First	Second	Third
x_0	$f[x_0]$			
x_1	$f[x_1]$	$f[x_0,x_1] = \dfrac{f[x_1]-f[x_0]}{x_1-x_0}$		
x_2	$f[x_2]$	$f[x_1,x_2] = \dfrac{f[x_2]-f[x_1]}{x_2-x_1}$	$f[x_0,x_1,x_2] = \dfrac{f[x_1,x_2]-f[x_0,x_1]}{x_2-x_0}$	
x_3	$f[x_3]$	$f[x_2,x_3] = \dfrac{f[x_3]-f[x_2]}{x_3-x_2}$	$f[x_1,x_2,x_3] = \dfrac{f[x_2,x_3]-f[x_1,x_2]}{x_3-x_1}$	$f[x_0,\ldots,x_3]$

where $f[x_0,\ldots,x_3] = \dfrac{f[x_1,x_2,x_3]-f[x_0,x_1,x_2]}{x_3-x_0}$.

To obtain the divided-difference coefficients of the interpolatory polynomial P on the $(n+1)$ distinct numbers $x_0, x_1, \dots, x_n$ for the function f :
INPUT numbers $x_0, x_1, \dots, x_n$;
$f(x_0), f(x_1), \dots, f(x_n)$ as $F_{0,0}, F_{1,0}, \dots F_{n,0}$.
OUTPUT the numbers $F_{0,0}, F_{1,1}, \dots F_{n,n}$ where $F_{i,i} = f[x_0, x_1, \dots, x_i]$.

1. For i from 1 to n do
   a. For j from 1 to i do
      $$F_{i,j} = \frac{F_{i,j-1} - F_{i-1,j-1}}{x_i - x_{i-j}}.$$
      (where $F_{i,j} = f[x_{i-j}, \dots, x_i]$.)
2. OUTPUT $F_{0,0}, F_{1,1}, \dots F_{n,n}$.

Example 8

Find a polynomial of degree at most 3 say $P(x)$ passing through the points

x_i	0	1	2	3
y_i	3	6	11	18

using Newton's Divided Difference.
Then, use the interpolating polynomial to approximate $f(1.5)$.

i	x_i	$f[x_i]$	$f[x_i, x_{i+1}]$	$f[x_i, x_{i+1}, x_{i+2}]$	$f[x_i, x_{i+1}, x_{i+2}, x_{i+3}]$
0	0	3			
1	1	6	3		
2	2	11	5	1	
3	3	18	7	1	0

$P(x) = 3 + 3(x-0) + 1(x-0)(x-1) + 0(x-0)(x-1)(x-2) = 3 + 2x + x^2$.
Then, $P(1.5) = 8.25$.

1. In the previous section we found an explicit representation for Lagrange polynomials and their error when approximating a function on an interval.
2. A frequent use of these polynomials involves the interpolation of tabulated data.
3. In this case an explicit representation of the polynomial might not be needed,
4. only the values of the polynomial at specified points.
5. In this situation the function underlying the data might not be known so the explicit form of the error cannot be used.
6. Newton's method of interpolation involved two steps: computation of the coefficients, followed by evaluation of the polynomial.
7. It works well if the interpolationis carried out repeatedly at different values of x using the same polynomial.
8. If only one point is to interpolated, a method that computes the interpolant in a single step, we use Neville's algorithm.

Definition 9

Let f be a function defined at $x_0, x_1, x_2, \ldots, x_n$, and suppose that $m_1, m_2, \ldots, m_k$ are k distinct integers, with $0 \leq m_i \leq n$ for each i. The Lagrange polynomial that agrees with $f(x)$ at the k points $x_{m_1}, x_{m_2}, \ldots, x_{m_k}$ is denoted $P_{m_1, m_2, \ldots, m_k}(x)$.

Theorem 10

Let f be defined at $x_0, x_1, \ldots, x_k$, and let x_i and x_j be two distinct numbers in this set. Then

$$P(x) = \frac{(x - x_i)P_{0,1,\ldots,i-1,i+1} - (x - x_j)P_{0,1,\ldots,j-1,j+1}}{x_j - x_i}$$

is the k-th Lagrange polynomial that interpolates f at the $k+1$ points $x_0, x_1, \ldots, x_k$.

For example,

1. $$P_{0,1} = \frac{(x-x_0)P_1 - (x-x_1)P_0}{x_1 - x_0}$$
2. $$P_{1,2} = \frac{(x-x_1)P_2 - (x-x_2)P_1}{x_2 - x_1}$$
3. $$\begin{aligned} P_{0,1,2} &= \frac{(x-x_0)P_{1,2} - (x-x_2)P_{0,1}}{x_2 - x_0} \\ &= \frac{(x-x_0)P_{1,2} - (x-x_1)P_{0,2}}{x_1 - x_0} \\ &= \frac{(x-x_1)P_{0,2} - (x-x_2)P_{0,1}}{x_2 - x_1} \end{aligned}$$
4. $$P_{2,3,4,5} = \frac{(x-x_2)P_{3,4,5} - (x-x_5)P_{2,3,4}}{x_5 - x_2}$$

and so on.

To avoid the multiple subscripts, we let $Q_{i,j}(x)$, for $0 \le j \le i$, denote the interpolating polynomial of degree j on the $(j+1)$ numbers $x_{i-j}, x_{i-j+1}, \ldots, x_{i-1}, x_i$; that is,

$$Q_{i,j} = P_{i-j,i-j+1,\ldots,i-1,i}.$$

Using this notation provides the Q notation array in below table

Table: Neville's Table

x_0	$P_0 = Q_{0,0}$			
x_1	$P_1 = Q_{1,0}$	$P_{0,1} = Q_{1,1}$		
x_2	$P_2 = Q_{2,0}$	$P_{1,2} = Q_{2,1}$	$P_{0,1,2} = Q_{2,2}$	
x_3	$P_3 = Q_{3,0}$	$P_{2,3} = Q_{3,1}$	$P_{1,2,3} = Q_{3,2}$	$P_{0,1,2,3} = Q_{3,3}$

To evaluate the interpolating polynomial P on the $n+1$ distinct numbers $x_0, \ldots, x_n$ at the number x for the function f :
INPUT numbers $x, x_0, x_1, \ldots, x_n$; values $f(x_0), f(x_1), \ldots, f(x_n)$ as the first column $Q_{0,0}, Q_{1,0}, \ldots, Q_{n,0}$ of Q.
OUTPUT the table Q with $P(x) = Q_{n,n}$.

1. For i from 1 to n do
   a. For j from 1 to i do

      Set $Q_{i,j} = \dfrac{(x - x_{i-j})Q_{i,j-1} - (x - x_i)Q_{i-1,j-1}}{x_i - x_{i-j}}$
2. OUTPUT Q.

Example 11

Find a polynomial of degree at most 3 say $P(x)$ passing through the points

x_i	0	1	2	3
y_i	3	6	11	18

using Newton's Divided Difference. Then, use the interpolating polynomial to approximate $f(1.5)$.

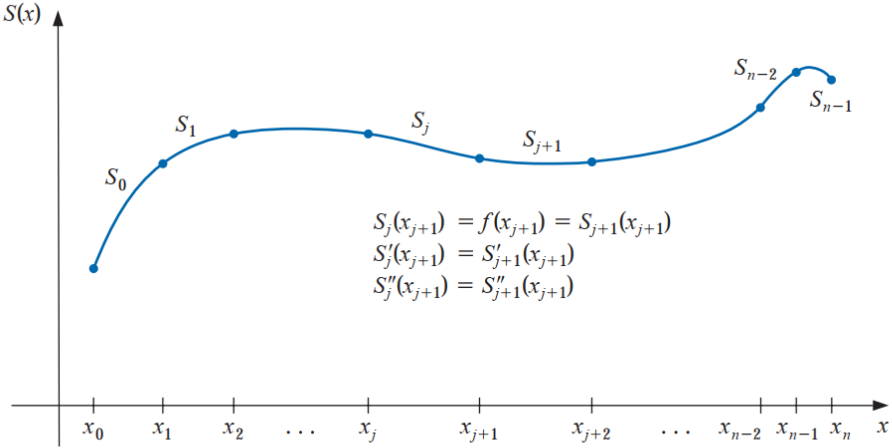
Table: Neville' Method $Q_{i,j}(1.5)$

i	x	$Q_{i,0}$	$Q_{i,1}$	$Q_{i,2}$	$Q_{i,3}$
0	0.00	3.00			
1	1.00	6.00	7.50		
2	2.00	11.00	8.50	8.25	
3	3.00	18.00	7.50	8.25	8.25

Thus, $P(1.5) = Q_{3,3}(1.5) = 8.25$.

1. The most common piecewise-polynomial approximation uses cubic polynomials between each successive pair of nodes and is called cubic spline interpolation.
2. A general cubic polynomial involves four constants, so there is sufficient flexibility in the cubic spline procedure to ensure that the interpolant is not only continuously differentiable on the interval, but also has a continuous second derivative.
3. The construction of the cubic spline does not,however, assume that the derivatives of the interpolant agree with those of the function it is approximating, even at the nodes.

7. Cubic Spline Interpolation



Definition 12 (Cubic Spline Interpolation)

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:

1. $S(x)$ is a cubic polynomial, denoted $S_j(x)$, on the subinterval $[x_j, x_{j+1}]$ for each $j = 0, 1, \cdots, n-1$;
2. $S_j(x_j) = f(x_j)$ and $S_j(x_{j+1}) = f(x_{j+1})$ for each $j = 0, 1, \ldots, n-1$;
3. $S_{j+1}(x_{j+1}) = S_j(x_{j+1})$ for each $j = 0, 1, \ldots, n-2$;
4. $S'_{j+1}(x_{j+1}) = S'_j(x_{j+1})$ for each $j = 0, 1, \ldots, n-2$;
5. $S''_{j+1}(x_{j+1}) = S''_j(x_{j+1})$ for each $j = 0, 1, \ldots, n-2$;
6. One of the following sets of boundary conditions is satisfied:
   a. $S''(x_0) = S''(x_n) = 0$
   b. $S'(x_0) = f'(x_0)$ and $S'(x_n) = f'(x_n)$.

Example 13

Construct a cubic spline passing through $(1, 2), (2, 3)$, and $(3, 5)$.

ⓐ $S_0(x) = a_0 + b_0(x-1) + c_0(x-1)^2 + d_0(x-1)^3,$

$S_1(x) = a_1 + b_1(x-2) + c_1(x-2)^2 + d_1(x-2)^3.$

ⓑ $2 = f(1) = S_0(1) = a_0,$

$3 = f(2) = S_0(2) = a_0 + b_0 + c_0 + d_0,$

$3 = f(2) = S_1(2) = a_1,$

$5 = f(3) = S_1(3) = a_1 + b_1 + c_1 + d_1.$

ⓒ $b_0 + 2c_0 + 3d_0 = S_0'(2) = S_1'(2) = b_1,$

$2c_0 + 6d_0 = S_0''(2) = S_1''(2) = 2c_1.$

ⓓ $2c_0 = S_0''(1) = 0,$

$2c_1 + 6d_1 = S_1''(1) = 0.$

$$\Rightarrow S(x) = \begin{cases} 2 + \frac{3}{4}(x-1) + \frac{1}{4}(x-1)^3, & \text{for } x \in [1, 2] \\ 3 + \frac{3}{2}(x-2) + \frac{3}{4}(x-2)^2 - \frac{1}{4}(x-2)^3, & \text{for } x \in [2, 3]. \end{cases}$$

Theorem 14 (Natural Splines)

If f *is defined at* $a = x_0 < x_1 < \cdots < x_n = b$, *then* f *has a unique natural spline interpolant* S *on the nodes* $x_0, x_1, \ldots, x_n$; *that is, a spline interpolant that satisfies the natural boundary conditions* $S''(a) = 0$ *and* $S''(b) = 0$.

7. Cubic Spline Interpolation I

To construct the natural cubic spline interpolant S for the function f, defined at the numbers $x_0 < x_1 < \cdots < x_n$, satisfying $S''(x_0) = S''(x_n) = 0$:

INPUT $n; x_0, x_1, \dots, x_n; a_0 = f(x_0), \dots, a_n = f(x_n)$.

OUTPUT a_j, b_j, c_j, d_j for $j = 0, 1, \dots, n-1$.

1. For i from 0 to $n-1$ set $h = x_{i+1} - x_i$.
2. For i from 1 to $n-1$ set $\alpha_i = \dfrac{3}{h_i}(a_{i+1} - a_i) - \dfrac{3}{h_{i-1}}(a_i - a_{i-1})$.
3. Set $I_0 = 1; \mu_0 = 0; z_0 = 0$.
4. For i from 1 to $n-1$ set

   $l_i = 2(x_{i+1} - x_{i-1}) - h_{i-1}\mu_{i-1}$;

   $\mu_i = h_i / l_i$;

   $z_i = (\alpha_i - h_{i-1} z_{i-1}) / l_i$.
5. Set $I_n = 1; z_n = 0; c_n = 0$.

6. For j from $n-1$ to 0 set
   $c_j = z_j - \mu_j c_{j+1}$;
   $b_j = (a_{j+1} - a_j)/h_j - h_j(c_{j+1} + 2c_j)/3$;
   $d_j = (c_{j+1} - c_j)/(3h_j)$.
7. OUTPUT a_j, b_j, c_j, d_j for j from 0 to $n-1$.