Paper PT-204 Unit III

Interpolation

1 Spline-Interpolation

So far we have been considering interpolation by means of a single polynomial in the entire range.

Let's now consider interpolation using different polynomials (but of the same degree) at different intervals.

Let the function f(x) be defined at the nodes $a = x_0, x_1, x_2, ..., x_n = b$.

The problem now is to construct piecewise polynomials $S_j(x)$ on each interval $[x_j, x_{j+1}]$, $j = 0, 1, 2, ..., n-1$, so that the resulting function S(x) is an interpolant for the function f(x).

The simplest such polynomials are of course, linear polynomials (straight lines). The interpolating polynomial in this case is called **linear spline**. The two biggest disadvantages of a linear spline are

(i) the convergence is rather slow, and
(ii) not suitable for applications demanding smooth approximations, since these splines have corner at the knots.

Likewise the **quadratic splines** have also certain disadvantages.

The most common and widely used splines are cubic splines. Assume that the cubic polynomial $S_j(x)$, has the following form:

$$S_j(x) = a_j + b_j(x - x_j) + c_j(x - x_j)^2 + d_j(x - x_j)^3, \quad j = 0, 1, \ldots, n-1 \qquad (1)$$

Since $S_j(x)$ contains four unknowns, to construct n cubic polynomials $S_0(x), S_1(x), ..., S_{n-1}(x)$, we need 4n conditions. To have these 4n conditions, a **cubic spline S(x)** for the **function f(x)** can be conveniently defined as follows:

2. Cubic Spline Interpolant

A function $S(x)$, denoted by $S_j(x)$, over the interval $[x_j, x_{j+1}]$, $j = 0, 1, ..., n-1$ is called **a cubic spline interpolant** if the following **conditions** hold:

(i) $S_j(x_j) = f_j$, $\quad j = 0, 1, 2, ..., n.$

(Cubic spline value at nodes should match with the function value)

(ii) $S_{j+1}(x_{j+1}) = S_j(x_{j+1})$, $\quad j = 0, 1, 2, ..., n-2.$

(Cubic spline segments value at the common nodes should match with each other values)

(iii) $S'j+1(xJ+1) = S'j(xj+1$ $\quad j = 0, 1, 2, ..., n-2.$

(The value of gradients of the cubic spline segments at the common nodes should match with each other)

(iv) $S''j+1(xJ+1) = S''j(xj+1$ $\quad j = 0, 1, 2, ..., n-2.$

(The value of the gradient of gradients of the cubic spline segments at the common nodes should match with each other)

In equation (1) for j there are four unknowns: a_j, b_j, c_j, and d_j and there are n such polynomials to be determined, all together there are 4n unknowns. However, conditions (i)-(iv) above give only $(4n-2)$ equations:

Condition (i) gives $n+1$, and each of (ii)-(iv) gives $n-1$ options.

So, to completely determine the cubic spline, we must need two more equations. To obtain these two additional equations, we can invoke boundary conditions.

If the gradient at the boundary is assumed as constant then the second derivative of $S(x)$ can be approximated as :
$S''(x_o) = S''(x_n) = 0$ (free or natural boundary)

As an alternative ,on the other hand, if only the first derivative is estimated, we can use the following boundary conditions:

$S'(x_o) = f'(x_o)$ and $S'(x_n) = f'(x_n)$ (clamped boundary)

In this presentation we will discuss only the clamped cubic spline here.

3. Spline-Interpolation for four nodes.

Lets begin with following nodes and corresponding function values.

point	0	1	2	3
x	x0	x1	x2	x3
f(x)	f0	f1	f2	f3

From the condition (i), we immediately obtain using equation (1)

$a0 = f0$ (at the point 0) (2)
$a1 = f1$ (at the point 1) (3)
$a2 = f2$ (at the point 2) (4)
$a3 = f3$ (at the point 3) (5)

Setting $h_j = x_{j+1} - x_j$ and using conditions (ii) to (iv) we immediately get

(at the point 1)

$a1 = a0 + b0h0 + c0(h0)^2 + d0(h0)^3$ (6)
$b1 = b0 + 2c0h0 + 3d0(h0)^2$ (7)
$c1 = 2c0 + 6d0h0$ (8)

(at the point 2)

$a2 = a1 + b1h1 + c1(h1)^2 + d1(h1)^3$ (9)

$$b2 = b1 + 2c1h1 + 3d1(h1)^2 \qquad (10)$$
$$c2 = 2c1 + 6d1h1 \qquad (11)$$

Remaining two additional conditions are

$$S'(x0) \;=\; b0 \;=\; f'(a) \qquad \text{(at the initial point)} \qquad (12)$$

and

$$S'(x3) \;=\; b0 \;=\; f'(b) \qquad \text{(at the last point)} \qquad (13)$$

As a0, a1, a2 are known in terms of function values, now, eliminating b0, b1, d0 and d1 using equations (2) through (13), the obtained equations may be written in forms of matrix notation as follows:

$$\begin{bmatrix} 2h0 & h0 & 0 \\ h0 & 29h0 + h1) & h1 \\ 0 & h1 & 2h1 \end{bmatrix} \begin{pmatrix} c0 \\ c1 \\ c2 \end{pmatrix} = \begin{pmatrix} \frac{3}{h0}(a1 - a0) - 3f'(a) \\ \frac{3}{h1}(a2 - a1) - \frac{3}{h0}(a1 - a0) \\ 3f'(a) - \frac{3}{h1}(a2 - a1) \end{pmatrix} \qquad (14)$$

Once c0, c1 and c2 are known by solving the above system of equations, the quantities bj and dj can be computed as follows:

$$bj = \frac{(a(j-1) - aj)}{hj} - \frac{hj}{3}(c\,(j-1) - c\,j)\ , \ j = 2,1,0$$

$$dj = \frac{(c\,(j+1) - c\,j)}{3\,hj}\ , \qquad j = 2,1,0$$

4. Algorithm for Computing Clamped Cubic Spline

Inputs:

(a) The nodes $x_0, x_1, \ldots, x_n$.

(b) The functional values: $f(x_i) = f_i$, $i = 0, 1, \ldots, n$.
(Note that $a = x_0$ and $b = x_n$).

(c) $f'(x0) = f'(a)$ and $f'(xn) = f'(b)$

Outputs: The coefficients $a_0, \ldots, a_n$; $b_0, \ldots, b_n$; $c_0, c_1, \ldots, c_n$, and $d_0, d_1, \ldots, d_n$ of the n polynomials $S_0(x), S_1(x), \ldots, S_{n-1}(x)$ of which the cubic interpolant S(x) is composed.

Step 1. for i = 0, 1, ..., n −1 do
$h_j \leftarrow (x_{j+1} - x_j)$

Step 2. For i = 0, 1, ..., n do #Compute $a_0, a_1, \ldots, a_n$:
$a_i \leftarrow f(x_i)$

Step 3. Compute the coefficients $c_0, c_1, \ldots, c_n$ by solving the system Ax = r, where A is square matrix, x is column marix of c, and r is right hand side column matrix in equation (13).

Step 4. Compute the coefficients $b_0, \ldots, b_n$ and $d_0, d_1, \ldots, d_n$ as given in the equations (14) and (15) above.

Exercises

(1) Consider the following table

x	1	1.01	1.02	**1.025**	1.03
$f(x)$	0	0.01	0.0198	**?**	0.0296

(a) Find an approximation of $f(1.025)$ using Lagrangian interpolating polynomial of degree 3

(2) Suppose that a table is to be prepared for the function $f(x) = \ln x$ on $[1, 3.5]$ with equal spacing nodes such that the interpolation with third degree polynomial will give an accuracy of $\varepsilon = 5\text{x}10^{-8}$. Determine how small h has to be to guarantee the above accuracy.

(3) (a) Find an approximate value of $\log_{10}(5)$ using Newton's forward difference formula with $x_0 = 1$, $x_1 = 2.5$, $x_2 = 4$, and $x_3 = 5.5$.

(b) Repeat part (a) using Newton's backward differences.

(c) Compare the results.