

Approximation By Spline Functions

Def. A function S is called a spline of degree k if

- The domain of S is an interval $[a, b]$.
- $S, S', \dots, S^{(k-1)}$ are continuous on $[a, b]$.
- There are points t_i (the knots of S) such that $a = t_0 < t_1 < \dots < t_n = b$ and such that S is a polynomial of degree at most k on each $[t_i, t_{i+1}]$.

When $k = 1$, the splines are called linear splines, when $k = 2$, the splines are called quadratic splines, when $k = 3$, the splines are called cubic splines. Here we are mainly interested in linear splines and cubic splines.

Interpolation by Linear Splines

Problem: Given $n + 1$ points $(t_0, y_0), (t_1, y_1), \dots, (t_n, y_n)$, where without loss of generality we assume $t_0 < t_1 < \dots < t_n$. Seek a linear spline $S(x)$ such that $S(t_i) = y_i$ for $0 \leq i \leq n$ and t_i are the knots of $S(x)$.

Obviously we can write

$$S(x) = \begin{cases} S_0(x), & t_0 \leq x \leq t_1 \\ S_1(x), & t_1 \leq x \leq t_2 \\ \vdots \\ S_{n-1}(x), & t_{n-1} \leq x \leq t_n \end{cases}$$

where $S_i(x) = y_i + m_i(x - t_i)$, $t_i \leq x \leq t_{i+1}$, with slope $m_i = (y_{i+1} - y_i)/(t_{i+1} - t_i)$. So $S(x)$ is a piecewise linear polynomial.

Algorithm for evaluating $S(x)$ (given x, t_i, y_i and $m_i, i = 0, 1, \dots, n$):

```
for  $i = 0 : n - 1$ 
  if  $x - t_{i+1} \leq 0$ ,
    exit loop
  end
end
 $S \leftarrow y_i + m_i(x - t_i)$ 
```

Remarks:

- When $x < t_0$, the algorithm gives $S = y_0 + m_0(x - t_0)$; when $x > t_n$, it gives $S = y_{n-1} + m_{n-1}(x - t_{n-1})$.
- A *binary search* can be used to find the desired interval which consists of x . Averagely, this is more efficient.

Interpolation by Cubic Splines

For a linear spline, generally S' is not continuous, so its graph lacks of smoothness. For a quadratic spline, generally S'' is not continuous, so the curvature of its graph changes abruptly at each knot. So in practice, the most frequently used splines are cubic splines.

Problem: Given $n + 1$ points $(t_0, y_0), (t_1, y_1), \dots, (t_n, y_n)$, where without loss of generality we assume $t_0 < t_1 < \dots < t_n$. Seek a cubic spline $S(x)$ such that $S(t_i) = y_i$ for $0 \leq i \leq n$ and t_i are the knots of $S(x)$.

Obviously we can write

$$S(x) = \begin{cases} S_0(x), & t_0 \leq x \leq t_1 \\ S_1(x), & t_1 \leq x \leq t_2 \\ \vdots \\ S_{n-1}(x), & t_{n-1} \leq x \leq t_n \end{cases}$$

where S_i is a cubic polynomial on $[t_i, t_{i+1}]$.

Number of unknowns:

Each S_i has 4 unknowns. So there is a total of $4n$ unknowns.

Number of conditions:

$S(t_i) = y_i$ for $i = 0, 1, \dots, n$ result in $n + 1$ conditions. $S_{i-1}^{(k)}(t_i) = S_i^{(k)}(t_i)$ for $k = 0, 1, 2$ and $i = 1, \dots, n - 1$ lead to $3(n - 1)$ conditions. So there is a total of $4n - 2$ conditions.

In order to get a unique solution, we need 2 more extra conditions. Here we impose the following two conditions:

$$S''(t_0) = S''(t_n) = 0.$$

The resulting spline function is called a *natural cubic spline*.

Constructing a Natural Cubic Spline

Let $z_i = S''(t_i)$ ($0 \leq i \leq n$). On $[t_i, t_{i+1}]$, $S_i''(x)$ is a linear polynomial and $S_i''(t_i) = z_i$ and $S_i''(t_{i+1}) = z_{i+1}$. So we can write

$$S_i''(x) = \frac{x - t_{i+1}}{t_i - t_{i+1}} z_i + \frac{x - t_i}{t_{i+1} - t_i} z_{i+1}.$$

Integrating $S_i''(x)$ twice, we obtain

$$S_i(x) = (t_{i+1} - x)^3 \frac{z_i}{6h_i} + (x - t_i)^3 \frac{z_{i+1}}{6h_i} + cx + d, \quad (1)$$

where $h_i \equiv t_{i+1} - t_i$, and c and d are constants of integration. We impose the conditions $S_i(t_i) = y_i$ and $S_i(t_{i+1}) = y_{i+1}$. Then we have

$$\begin{aligned} h_i^3 \frac{z_i}{6h_i} + ct_i + d &= y_i, \\ h_i^3 \frac{z_{i+1}}{6h_i} + ct_{i+1} + d &= y_{i+1}, \end{aligned}$$

Algorithm for finding z_i , $i = 0, \dots, n$ (given t_i, y_i , $i = 0, \dots, n$):

```

for  $i = 0 : n - 1$ 
     $h_i \leftarrow t_{i+1} - t_i$ 
     $b_i \leftarrow (y_{i+1} - y_i)/h_i$ 
end
% Forward elimination
 $u_1 \leftarrow 2(h_0 + h_1)$ 
 $v_1 \leftarrow 6(b_1 - b_0)$ 
for  $i = 2 : n - 1$ 
     $u_i \leftarrow 2(h_{i-1} + h_i) - h_{i-1}^2/u_{i-1}$ 
     $v_i \leftarrow 6(b_i - b_{i-1}) - h_{i-1}v_{i-1}/u_{i-1}$ 
end
% Back substitution
 $z_n \leftarrow 0$ 
for  $i = n - 1 : -1 : 1$ 
     $z_i \leftarrow (v_i - h_i z_{i+1})/u_i$ 
end
 $z_0 \leftarrow 0$ 

```

Evaluation of $S(x)$

$$\begin{aligned}
 S_i(x) = & (t_{i+1} - x)^3 \frac{z_i}{6h_i} + (x - t_i)^3 \frac{z_{i+1}}{6h_i} + \left[\frac{y_{i+1} - y_i}{h_i} - \frac{h_i}{6}(z_{i+1} - z_i) \right] x \\
 & + \frac{y_i t_{i+1} - y_{i+1} t_i}{h_i} + \frac{h_i}{6}(t_i z_{i+1} - t_{i+1} z_i).
 \end{aligned}$$

This is not the best computational form. As we want to utilize nested multiplication, we write

$$S_i(x) = A_i + B_i(x - t_i) + C_i(x - t_i)^2 + D_i(x - t_i)^3.$$

Notice $A_i = S_i(t_i)$, $B_i = S'_i(t_i)$, $C_i = \frac{1}{2}S''_i(t_i)$, $D_i = \frac{1}{6}S'''_i(t_i)$. Then we can obtain

$$\begin{aligned}
 A_i &= y_i \\
 B_i &= -h_i z_{i+1}/6 - h_i z_i/3 + (y_{i+1} - y_i)/h_i \\
 C_i &= z_i/2 \\
 D_i &= (z_{i+1} - z_i)/(6h_i)
 \end{aligned}$$

$$S_i(x) = A_i + (x - t_i)(B_i + (x - t_i)(C_i + (x - t_i)D_i)).$$

Algorithm for evaluating $S(x)$ (given x, t_i, y_i and z_i for $i = 0, 1, \dots, n$):

```
for  $i = 0 : n - 1$ 
  if  $x - t_{i+1} \leq 0$ 
    exit loop
  end
end
 $h \leftarrow t_{i+1} - t_i$ 
 $B \leftarrow -hz_{i+1}/6 - hz_i/3 + (y_{i+1} - y_i)/h$ 
 $D \leftarrow (z_{i+1} - z_i)/(6h)$ 
 $S \leftarrow y_i + (x - t_i)(B + (x - t_i)(z_i/2 + (x - t_i)D))$ 
```

Note. You can use a binary search to find the desired interval consisting of x .

MATLAB Spline Tools: spline