

Derivation of cubic spline equations

Paht Juangphanich

I. Spline Properties

1. $S(x) = S_{k,0} + S_{k,1}(x - x_k) + S_{k,2}(x - x_k)^2 + S_{k,3}(x - x_k)^3$
2. $S(x_k) = y_k$
3. $S_k(x_{k+1}) = S_{k+1}(x_{k+1})$
4. $S'_k(x_{k+1}) = S'_{k+1}(x_{k+1})$
5. $S''_k(x_{k+1}) = S''_{k+1}(x_{k+1})$

All five of these properties must be satisfied for a spline to exist

II. Derivation

General Equation for a spline

$$S(x) = S_{k,0} + S_{k,1}(x - x_k) + S_{k,2}(x - x_k)^2 + S_{k,3}(x - x_k)^3 \quad (1)$$

$$S''_k(x) = S''(x_k) \frac{x - x_{k+1}}{x_k - x_{k+1}} + S''(x_{k+1}) \frac{x - x_k}{x_{k+1} - x_k}$$

Let $x_{k+1} - x_k$ equal to h_k

$$S''_k(x) = -S''(x_k) \frac{x - x_{k+1}}{h_k} + S''(x_{k+1}) \frac{x - x_k}{h_k}$$

Let $S''_k = m_k$ Integrating twice results in:

$$S_k(x) = -\frac{m_k}{6h_k}(x - x_{k+1})^3 + \frac{m_{k+1}}{6h_k}(x - x_k)^3 + p_k(x - x_{k+1}) + q_k(x - x_k) \quad (2)$$

If $S_k(x_k) = y_k$ and $S_k(x_{k+1}) = y_{k+1}$ then the values of p_k and q_k can be solved for.

$$y_k = \frac{m_k}{6}h_k^2 - p_k h_k \quad (3)$$

$$y_{k+1} = \frac{m_{k+1}}{6}h_k^2 + q_k h_k \quad (4)$$

$$S_k(x) = -\frac{m_k}{6h_k}(x - x_{k+1})^3 + \frac{m_{k+1}}{6h_k}(x - x_k)^3 + \left(-\frac{y_k}{h_k} + \frac{m_k h_k}{6}\right)(x - x_{k+1}) + \left(\frac{y_{k+1}}{h_k} - \frac{m_{k+1} h_k}{6}\right)(x - x_k) \quad (5)$$

Taking the first derivative of equation 5 results in:

$$S'_k(x) = -\frac{m_k}{2h_k}(x - x_{k+1})^2 + \frac{m_{k+1}}{2h_k}(x - x_k)^2 + \frac{y_{k+1} - y_k}{h_k} + \frac{h_k}{6}(m_k - m_{k+1}) \quad (6)$$

We have to satisfy condition 4. Lets define $S'_k(x_k)$ and $S'_{k-1}(x_k)$ let $d_k = \frac{y_{k+1}-y_k}{h_k}$

$$S'_k(x_k) = -\frac{m_k}{3} - \frac{m_{k+1}}{6}h_k + d_k \quad (7)$$

$$S'_{k-1}(x_k) = -\frac{m_{k-1}}{3} - \frac{m_k}{6}h_{k-1} + d_{k-1} \quad (8)$$

Setting equations 7 and 8 equal to each other satisfying requirement 4 results in the following

$$h_{k-1}m_{k-1} + 2(h_{k-1} + h_k)m_k + h_k m_{k+1} = 6(d_k - d_{k-1}) \quad (9)$$

For $k = 1:n-1$

III. Clamped Spline

If a spline is clamped it means the derivatives at the end points are known. In this case forward difference may be used to determine a relationship for m_0 and m_n .

$$m_0 = \frac{3}{h_0}(d_0 - S'(x_0)) - \frac{m_1}{2} \quad (10)$$

$$m_N = \frac{3}{h_{N-1}}(S'(x_N) - d_{N-1}) - \frac{m_{N-1}}{2} \quad (11)$$

For natural splines m_0 and $m_N = 0$

IV. Code

The code below is used to solve natural and clamped cubic splines.

Listing 1. Descriptive Caption Text

```
1 function [S x2] = pspline(x,y,dydx,npoints)
   % Inputs
3  %   x = x1, x2, x3, x4, ... , xn as vector
   %   y = same as x
5  %   dydx = derivatives at end points
   %       dydx(0) = const then clamped at start
7  %       dydx(n) = const then clamped at end
   h = diff(x); % h =1 to n-1 % h(0) = x(1)-x(0) h(n) = x(n)-x(n-1)
9  d = diff(y)./h; % d = 1 to n-1
   u = 6.*diff(d); % u = 1 to n-1
11 n = length(x);
   a = zeros(1,n-2);
13 b = a;
   c = a;
15 m=a;
   e=a;
17 phi = a;
   % if k ==1 evaluate below
19 if dydx(1) ~= 0 % Clamped Spline
       a(1) = 0;
21     b(1) = (3/2*h(1)+2*h(2));
       c(1) = h(1);
23     e(1) = u(1)-3*(d(1)-dydx(1));
   else % Natural Spline
25     a(1) = 0;
       b(1) = 2*(h(1)+h(2));
27     c(1) = h(1);
       e(1) = u(1);
```

```

29     m(1)=0;
    end
31 % if k == n-1 evaluate below
    if dydx(2)~=0 % Clamped Spline
33         a(end) = h(end-1);
        b(end) = 2*h(end-1)+3/2*h(end);
35         c(end) = 0;
        e(end) = u(end)-3*(dydx(end)-d(end));
37     else % Natural Spline
        a(end) = h(end-1);
39         b(end) = 2*(h(end-1)+h(end));
        c(end) = 0;
41         e(end) = u(1);
        m(end) = 0;
43     end
    % loop for inner nodes k =2 to n-2
45     if (2 ~= (n-2))
        for k = 2:(n-2)
47             a(k) = h(k-1);
            b(k) = 2*(h(k-1)+h(k));
49             c(k) = h(k);
            e(k) = u(k);
51         end
        end
53 % Call thomas algorithm and solve
    m = zeros(1,n);
55 m(2:end-1) = thomas(a,b,c,e);
    if dydx(1) ~=0 % if clamped spline
57         m(1) = 3/h(1)*(d(1)-dydx(1)) -m(2)/2;%m0
        end
59     if dydx(2) ~= 0
        m(n) = 3/h(n-1)*(dydx(2)-d(n-1))-m(n-1)/2;
61     end
    % Substitute back into
63 S = zeros(length(m),npoints); w = S; x2=w;
    for k = 1:n-1 % 1 to number of points
65         s(k,1) = y(k); %s(k,0)
            s(k,2) = d(k)-(h(k)*(2*m(k)+m(k+1)))/6;
67         s(k,3) = m(k)/2;
            s(k,4) = (m(k+1)-m(k))/(6*h(k));
69         x2(k,:) = linspace(x(k),x(k+1),npoints);
            w(k,:) = x2(k,:)-x(k);
71         S(k,:) = ((s(k,4).*w(k,:)+s(k,3)).*w(k,:)+s(k,2)).*w(k,:)+s(k,1);
    end

```

V. Thomas Algorithm

Listing 2. Descriptive Caption Text

```

function [phi]= thomas(a,b,c,d)
2 % Thomas Algorithm
% Inputs: n = Size of system
4 % b = vector of sub diagonal elements
% d = vector of diagonal elements
6 % a = vector of superdiagonal elements
% c = right hand side vector

```

```

8  % Note: b(1) and a(n) are not used
   % Outputs: phi = solution vector
10
   %Written by Paht Juangphanich
12  n = length(a);
   cp=zeros(1,n);
14  dp=cp;
   phi=cp;
16
   for i=1:n
18       if (i==1)
           cp(i) = c(i)/b(i);
           dp(i) = d(i)/b(i);
20       else
           cp(i) = c(i)/(b(i)-cp(i-1)*a(i));
           dp(i) = (d(i)-dp(i-1)*a(i))/(b(i)-cp(i-1)*a(i));
22       end
24   end
   end
26  phi(n) = dp(n);
   for i=n-1:-1:1
28       phi(i) = dp(i)-cp(i)*phi(i+1);
   end

```

VI. Sample Problem

Generate a spline that goes through the following points. Solution and code posted below.

Listing 3. Descriptive Caption Text

```

1  x(1) = 0; y(1) = 0;
   x(2) = 1; y(2) = 0.5;
3  x(3) = 2; y(3) = 2;
   x(4) = 3; y(4) = 1.5;
5  dydx = [0 0];
   [S,w] = pspline(x(1:4),y(1:4),dydx,100);
7  [m n]= size(S);
   figure(1)
9  hold on
   for k = 1:m
11       plot(w(k,:),S(k,:));
   end
13 hold off;
   y=2;

```

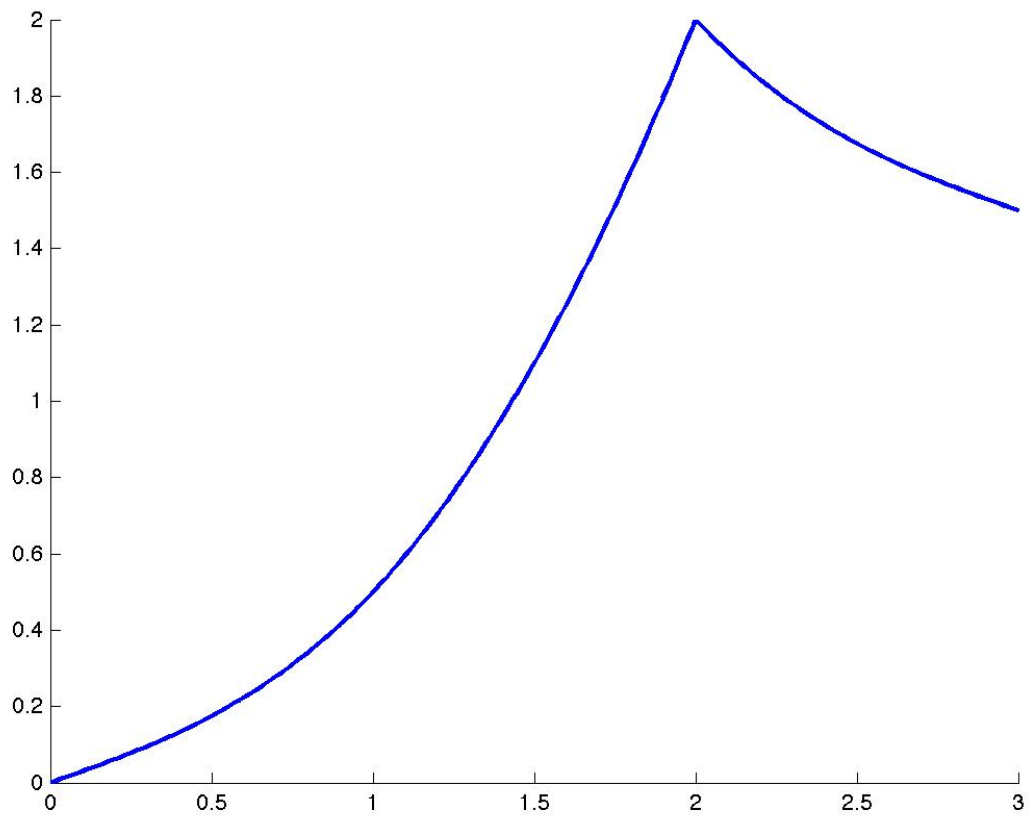


Figure 1. Result