1 Spline lnterpolation (Section 6.4)

A **spline function** consists of polynomial pieces on subintervals joined together with certain continuity conditions. Formally, suppose we have $m+1$ **ordered** points, called **knots**, $t_0, t_1, \dots, t_m$ (i.e., we know the values of each t_i and $t_i < t_{i+1}$). Thus, a **spline function of degree** k having knots $t_0, t_1, \dots, t_m$ is a function S such that

1. On each interval $[t_{i-1}, t_i)$, S is a polynomial of degree $\leq k$.
2. On $[t_0, t_n]$, S has a continuous $(k-1)$th derivative[1].

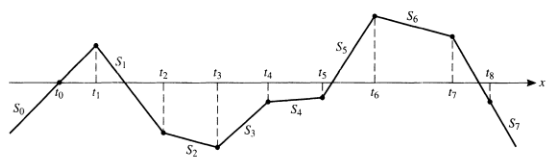
Basically, S is a piecewise polynomial of degree at most k having continuous derivatives of all orders up to $k-1$.

1.1 Degree 1 Spline Functions

Let $k=1$ so that we have a degree one spline function. Suppose we have coefficients a_i, b_i. Then, we can define the spline function S as

$$S = \begin{cases} S_0(x) = a_0 x + b_0 & x \in [t_0, t_1) \\ S_1(x) = a_1 x + b_1 & x \in [t_1, t_2) \\ \vdots & \\ S_{m-1}(x) = a_{m-1} x + b_{m-1} & x \in [t_{m-1}, t] \end{cases}.$$

From the second property, $S(x)$ is continuous, so the piecewise polynomials match up at the nodes. That is, $\boxed{S_i(t_{i+1}) = S_{i+1}(t_{i+1}).}$



Remark: This typically extends the knots. In other words, we might see

$$S = \begin{cases} S_0(x) & x \in (-\infty, t_1] \\ S_{m-1}(x) & x \in [t_{m-1}, \infty) \end{cases}.$$

1.1.1 Algorithm for Degree 1 Spline Functions

We can write some code to evaluate a **degree 1 spline**. The inputs are the coefficients $\{a_i\}$, $\{b_i\}$, the knot values $\{t_j\}$, and x such that $0 \leq i \leq m-1$ and $0 \leq j \leq m$.

Algorithm 1 Degree 1 Spline

function DegOneSpline($\{a_i\}, \{b_i\}, \{t_j\}, x$)
    $s \leftarrow a_{m-1} x + b_{m-1}$
    for $i \leftarrow 1$ to $m-1$ **do**
        if $x \leq t_i$ **then**
            $s \leftarrow a_{i-1} x + b_{i-1}$ ▷ Search into which interval x falls into.
            break
        end if
    end for
end function

[1]The wording here confused me. So, as a note to myself, here's an example: if we have $k=1$ (i.e., a linear spline function), then will S have a continuous 0th derivative? This is just the real function $f(x)$. So, essentially, if we have a linear spline function, we expect $f(x)$ to be continuous.

1.2 Cubic Spline Functions

We will now consider spline functions of degree 3, i.e., $k = 3$. Given the data points

x	t_1	t_2	t_3	$\ldots$	t_m
y	y_1	y_2	y_3	$\ldots$	y_m

we want to construct an interpolating cubic spline,

$$S(x) = \begin{cases} S_0(x) & x \in [t_0, t_1] \\ S_1(x) & x \in [t_1, t_2] \\ S_2(x) & x \in [t_2, t_3] \\ \vdots & \\ S_{m-1}(x) & x \in [t_{m-1}, t_m] \end{cases}.$$

Each piece of $S(x)$ will be cubic polynomials. There are $4m$ unknown coefficients[2].

1.2.1 Evaluation Conditions

The conditions for evaluating degree 3 polynomials are conditions for interpolation and continuity.

- Interpolation: for $0 \leq i \leq m-1$, we have

$$S_i(t_i) = y_i$$

$$S_i(t_{i+1}) = y_{i+1}.$$

  There are a total of $2m$ conditions here.

- Continuity: for $0 \leq i \leq m-2$, we have

$$S_i'(t_{i+1}) = S_{i+1}'(t_{i+1})$$

$$S_i''(t_{i+1}) = S_{i+1}''(t_{i+1})$$

  There are $2(m-1)$ conditions here.

In total, there are $2m + 2(m-1)$ conditions.

1.2.2 Finding $S(x)$

Define the coefficients as $z_i = S_i''(t_i)$ for $1 \leq i \leq m-1$. We know that $S_i''(x)$ is a linear function[3] on $[t_i, t_{i+1}]$. Hence, we can write

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

Then,

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

Likewise, we have

$$S_i''(t_{i+1}) = z_{i+1}.$$

We now want to think about integrating to obtain $S(x)$. For this, let $h_i = t_{i+1} - t_i$. Then,

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

[2] Recall that a cubic function looks like $ax^3 + bx^2 + cx + d$, with four coefficients.

[3] Since it's the *second* derivative of a cubic function.

Integrating yields

$$S_i'(x) = -\frac{z_i}{2h_i}(x - t_{i+1})^2 + \frac{z_{i+1}}{2h_i}(x - t_i)^2 + A_1,$$

where A_1 is an arbitrary constant. Integrating again yields

$$S_i(x) = -\frac{z_i}{6h_i}(x - t_{i+1})^3 + \frac{z_{i+1}}{6h_i}(x - t_i)^3 + A_1 x + A_2$$

for some arbitrary A_2. For easier computation, we can write

$$A_1 x + A_2 = C(x - t_i) + D(t_{i+1} - x)$$

for some arbitrary A_1, A_2, C, D. Then,

$$\boxed{S_i(x) = -\frac{z_i}{6h_i}(x - t_{i+1})^3 + \frac{z_{i+1}}{6h_i}(x - t_i)^3 + C_i(x - t_i) + D_i(t_{i+1} - x)},$$

where the first term can be rewritten as $-\frac{z_i}{6h_i}(x - t_{i+1})^3 = \frac{z_i}{6h_i}(t_{i+1} - x)^3$. We know that, from the interpolation condition,

$$S_i(t_i) = y_i = -\frac{z_i}{6h_i}(t_i - t_{i+1})^3 + D_i(t_{i+1} - t_i),$$

$$S_i(t_{i+1}) = y_{i+1} = \frac{z_{i+1}}{6h_i}\underbrace{(t_{i+1} - t_i)}_{h_i} + C_i(t_{i+1} - t_i),$$

where $C_i = \frac{y_{i+1}}{h_i} - \frac{z_{i+1}h_i}{6}$ and $D_i = \frac{y_i}{h_i} - \frac{z_i h_i}{6}$. Recall, from one of the conditions, that $S_{i-1}'(t_i) = S_i'(t_i) = z_i$ for $1 \leq i \leq m - 1$ and $S_i(t_i) = y_i$.

1.2.3 Finding z_i

We now want to determine the values of z_i for these $k = 3$ polynomials. To do so, we note that

$$S_i'(x) = -\frac{z_i}{2h_i}(t_{i+1} - x)^2 + \frac{z_{i+1}}{2h_i}(x - t_i)^2 + C_i - D_i.$$

$$\begin{aligned} S_{i-1}'(t_i) &= -\frac{z_{i-1}}{2h_{i-1}}(t_i - t_i)^2 + \frac{z_i}{2h_{i-1}}(t_i - t_{i-1})^2 + C_{i-1} - D_{i-1} \\ &= 0 + \frac{z_i}{2}h_{i-1} + C_{i-1} - D_{i-1} \\ &= -\frac{z_i}{2}h_i + C_i - D_i \\ &= S_i'(t_i) \end{aligned}$$

and

$$\frac{z_i}{2}h_{i-1} + \left(\frac{y_i}{h_{i-1}} - \frac{z_i h_{i-1}}{6} - \frac{y_{i-1}}{h_{i-1}} + \frac{z_{i-1}h_{i-1}}{6}\right) = -\frac{z_i}{2}h_i + \left(\frac{y_{i+1}}{h_i} - \frac{z_{i+1}h_i}{6} - \frac{y_i}{h_i} + \frac{z_i h_i}{6}\right).$$

We want to now solve for the z_i's on the left and then group. This gives us

$$\frac{z_{i-1}h_{i-1}}{6} + \frac{1}{6}\left(2(h_{i-1} + h_i)\right) z_i + \frac{h_i z_{i+1}}{6} = \left(\frac{y_{i+1}}{h_i} - \frac{y_i}{h_i}\right) - \left(\frac{y_i}{h_{i-1}} - \frac{y_{i-1}}{h_{i-1}}\right).$$

This represents a linear system with $m + 1$ unknowns and $m - 1$ equations. With $z_0 = z_m = 0$, we have

$$b_i = 6\left(\frac{y_{i+1}}{h_i} - \frac{y_i}{h_i}\right), \quad v_i = b_i - b_{i-1}.$$

$$u_i = 2(h_{i-1} + h_i), \quad h_i = t_{i+1} - t_i.$$

This defines the natural cubic spline. Rewriting yields

$$z_{i-1}h_{i-1} + z_i u_i + z_{i+1}h_{i+1} = v_i, \quad 1 \leq i \leq m - 1,$$

and thus the system looks like

$$\begin{bmatrix} u_1 & h_1 & 0 & \dots & 0 \\ h_1 & u_2 & h_2 & \dots & 0 \\ 0 & h_2 & u_3 & \dots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & \dots & u_{m-1} \end{bmatrix} \begin{bmatrix} z_1 \\ z_2 \\ z_3 \\ \vdots \\ z_{m-1} \end{bmatrix} = \begin{bmatrix} v_1 \\ v_2 \\ v_3 \\ \vdots \\ v_{m-1} \end{bmatrix}.$$

We can use Gauss elimination to solve for this system, specifically by reducing the tridiagonal matrix to a bidiagonal matrix. Then, we can back substitute to find the solutions.

1.2.4 Algorithm

We can write an algorithm to do this process for us. For $0 \leq i \leq m$, the algorithm takes in $\{t_i\}$ and $\{y_i\}$ and outputs $\{z_i\}$.

Algorithm 2 Cubic Spline

function $\textsc{CubicSpline}(\{t_i\}, \{y_i\})$
    for $i \leftarrow 0$ to $m-1$ **do**
        $h_i \leftarrow t_{i+1} - t_i$ ▷ Coefficients in system.
        $b_i \leftarrow 6(y_{i+1} - y_i)/h_i$
    end for
    $u_1 \leftarrow 2(h_1 + h_0)$
    $v_1 \leftarrow b_1 - b_0$
    for $i \leftarrow 2$ to $m-1$ **do**
        $u_i = 2(h_i + h_{i-1}) - h_{i-1}^2/u_{i-1}$ ▷ Reduce to bidiagonal.
        $v_i = b_i - b_{i-1} - h_{i-1}v_{i-1}/u_{i-1}$
    end for
    $z_m \leftarrow 0$
    for $i \leftarrow m-1$ to 1 step -1 **do**
        $z_i \leftarrow (v_i - h_i z_{i+1})/u_i$ ▷ Back substitution.
    end for
end function

Once the coefficients z are computed, then the spline $S_i(x)$ can be evaluated. That is, given an input x,

- we need to find the interval i such that $x \in [t_i, t_{i+1})$.
- we can use this index i to evaluate $S_i(x)$ for the coefficients z_{i-1}, z_i, z_{i+1}.