The cubic spline

As always our goal in this interpolation task is to define a curve $s(x)$ which interpolates the n data points

$$(x_1, y_1), (x_2, y_2), \ldots, (x_n, y_n) \quad (\text{where } x_1 < x_2 < \cdots < x_n)$$

In the fashion of piecewise polynomials we will define $s(x)$ as a <u>different</u> cubic polynomial $s_k(x)$ at each sub-interval $I_k = [x_k, x_{k+1}]$, i.e:

$$s(x) = \begin{cases} s_1(x) & x \in I_1 \\ s_2(x) & x \in I_2 \\ \vdots & \vdots \\ s_k(x) & x \in I_k \\ \vdots & \vdots \\ s_{n-1}(x) & x \in I_{n-1} \end{cases}$$

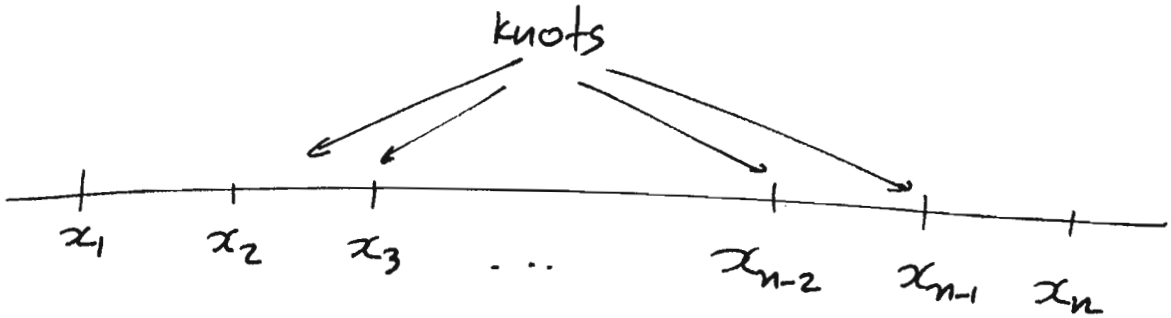
Each of the s_k's is a cubic polynomial

$$s_k(x) = \underline{a_0^{(k)}} + \underline{a_1^{(k)}} x + \underline{a_2^{(k)}} x^2 + \underline{a_3^{(k)}} x^3$$

Unknown coefficients

Since we have $(n-1)$ piecewise polynomials, in total we shall have to determine $4(n-1) = 4n-4$ unknown coefficients.

The points $(x_2, x_3, \ldots, x_{n-1})$ where the formula for $s(x)$ changes from one cubic polynomial (S_k) to another (S_{k+1}) are called <u>knots</u>



<u>Note</u>: In some textbooks, the extreme points x_1 & x_n are also included in the definition of what a knot is. We will stick with the definition we stated previously...

The piecewise polynomial interpolation method described as "cubic spline" also requires the neighboring polynomials S_k & S_{k+1} to be joined at x_{k+1} with a certain degree of smoothness. In detail:

The curve should be continuous: $S_k(x_{k+1}) = S_{k+1}(x_{k+1})$

The derivative (slope) should be continuous: $S_k'(x_{k+1}) = S_{k+1}'(x_{k+1})$

The 2nd derivative, as well: $S_k''(x_{k+1}) = S_{k+1}''(x_{k+1})$

(Note: If we force the next (3rd) derivative to match, this will force S_k & S_{k+1} to be exactly identical).

When determining the unknown coefficients $\{a_i^{(j)}\}$, each of these 3 smoothness constraints (for knots $k=2,3,\ldots,n-1$) needs to be satisfied, for a total of $3(n-2) = 3n-6$ constraint equations. We should not forget that we additionally want to <u>interpolate</u> all <u>n</u> data points i.e

$$s(x_i) = y_i \quad \text{for } i=1,2,\ldots,n \quad (n \text{ equation}$$

In total we have $(3n-6)+n = 4n-6$ total equations to satisfy, and $4n-4$ unknowns... consequently we will need <u>2</u> more equations to ensure that the unknown coefficients will be uniquely determined. Several plausible options exist on how to do that:

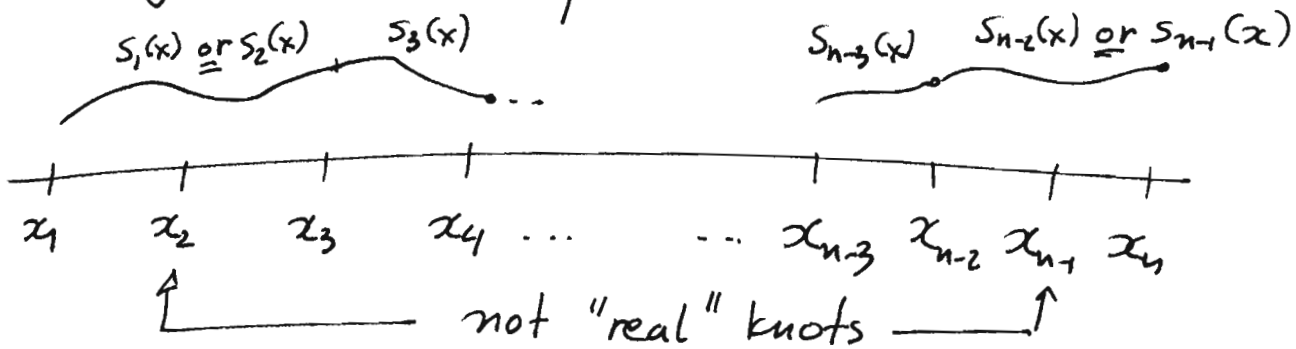
1. The "not-a-knot" approach: We stipulate that at the locations of the first (x_2) and last knot (x_{n-1}) the <u>third</u> derivative of $s(x)$ should also be continuous.

e.g.

$$S_1'''(x_2) = S_2'''(x_2)$$

and, $$S_{n-2}'''(x_{n-1}) = S_{n-1}'''(x_{n-1})$$

As we discussed before, these 2 additional constraints will effectively cause $S_1(x)$ to be identical with $S_2(x)$, and $S_{n-2}(x)$ to coincide with $S_{n-1}(x)$. In this sense, x_2 and x_{n-1} are no longer "knots" in the sense that the formula for $S(x)$ "changes" at these points: (hence the name).



2. Complete spline: If we have access to the derivative f' of the function being sampled by the y_i's (i.e $y_i = f(x_i)$), we can formulate the 2 additional constraints as:

$$s'(x_1) = f'(x_1)$$
$$s'(x_n) = f'(x_n)$$

$$\left(\text{or} \quad \begin{array}{l} s_1'(x_1) = f'(x_1) \\ s_{n-1}'(x_n) = f'(x_n) \end{array} \right)$$

Note that, qualitatively, using the complete spline approach is a better utilization of the flexibility of the spline curve in matching yet one more property of f; in contrast, the not-a-knot makes the spline "less flexible" by two degrees of freedom, in order to obtain a unique solution. However, we cannot always assume knowledge of f'

An additional 2 methodologies are:

3. The <u>natural cubic spline</u>: we use the following 2 constraints

$$s''(x_1) = 0$$
$$s''(x_n) = 0$$

Thus, $s(x)$ reaches the endpoints looking like a straight line (instead of a curved one).

4. Periodic spline:

$$s'(x_1) = s'(x_n)$$
$$s''(x_1) = s''(x_n)$$

This is useful when the underlying function f is also known to be periodical over $[a, b]$.

We will not discuss the analytic derivation of the cubic spline coefficients; instead we describe how to access this functionality within matlab, through the built-in functions spline and ppval

- The function spline is called as

$$s = \text{spline}(x, y)$$

x: The vector containing the x_i values

$$x = (x_1, x_2, \ldots, x_n)$$

y: A corresponding vector of y_i values

s: A specially encoded result, containing the necessary information for the generated spline. This is only used indirectly, by other MATLAB functions

The spline function can be used to implement either the not-a-knot, or the complete spline method.

→ If length(x) = length(y), then y is assumed to contain the values $y = (y_1, y_2, \ldots, y_n)$ and the spline is generated using the not-a-knot approach.

→ To implement the complete spline approach, we provide 2 additional values in the vector y, starting with $y_1' = f'(x_1)$ and ending with $y_n' = f'(x_n)$, i.e.

$$y = (y_1', y_1, y_2, \ldots, y_{n-2}, y_{n-1}, y_n, y_n')$$

In this case we obviously have

$$\text{length}(y) = \text{length}(x) + 2.$$

This triggers MATLAB to implement the complete spline approach.

The ppval function takes in the information encoded in $\leq$ (the output of spline) and evaluates the spline curve at a number of different locations.

Syntax $\quad v = \text{ppval}(s, u)$

u : A vector of m new x-locations where we want the spline to be interpolated/evaluated

$$u = (u_1, u_2, \ldots, u_m).$$

v : The corresponding y-values of these u_i locations

$$v = (v_1, v_2, \ldots, v_m).$$

Example:

```
x = 0 : pi/5 : 2*pi ;
y = sin (x) ;
s = spline (x,y) ;
u = 0 : pi/100 : 2*pi ;
v = ppval (s,u) ;
plot (u,v) ;
plot (x,y,u,v) ;
w = sin (u)
plot (u,v,u,w) ;
```

Error analysis:

for simplicity, we will again assume that

$$h_1 = h_2 = \cdots = h_{n-1} = h \qquad (h_k = x_{k+1} - x_k)$$

For the not-a-knot method, we have

$$|f(x) - s(x)| \leq \frac{5}{384} \|f^{(4)}\|_\infty h^4$$

The "approximate" inequality is used because the interpolation error can be slightly larger near the endpoints of the interval $[a,b]$.

This is a very comparable result with the (non-smooth) piecewise cubic polynomial method:

$$|f(x)-s(x)| \leq \frac{9}{384} \|f''''\|_\infty h^4$$

Note though that the computation of the piecewise cubic method was <u>very local</u> and simple (every interval could be independently evaluated) while the computation of the coefficients of the cubic spline is more elaborate.

<u>Hermite spline</u>

Let us assume a number of x-locations

$$x_1 < x_2 < \cdots < x_{n-1} < x_n$$

and let us make the hypothesis that we know both f <u>and</u> f' at every location x_i. We denote these values by

$$\left.\begin{aligned} y_i &= f(x_i) \\ y_i' &= f'(x_i) \end{aligned}\right\} \quad i = 1, 2, \ldots, n$$