$3n-6$ constraint equations. We should not forget that we additionally want to *interpolate* all n data points, i.e.:

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

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

Lecture of: 12 Mar 2013

1. The "not-a-knot" approach: We stipulate that at the locations of the first (x_2) and last knot (x_{n-1}) the third 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 (which explains the name for this approach).

2. The *Complete spline* method: 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:

$$\begin{aligned} S'(x_1) &= f'(x_1) \\ S'(x_n) &= f'(x_n) \\ &\text{or} \\ S_1'(x_1) &= f'(x_1) \\ S_{n-1}'(x_n) &= f'(x_n) \end{aligned} \tag{10}$$

   Note that, qualitatively, using the complete spline approach is a better utilization of the flexibility of the spline curve in matching yet one move 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'.

   Two additional methodologies are:

3. The *natural cubic spline*: We use the following 2 constraints:

$$\begin{aligned} S''(x_1) &= 0 \\ S''(x_n) &= 0 \end{aligned}$$

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

4. Periodic spline:

$$\begin{aligned} S'(x_1) &= S'(x_n) \\ S''(x_1) &= S''(x_n) \end{aligned}$$

   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:

  $S = \texttt{spline}(x, y)$

  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 generalized 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 $\texttt{length}(x) = \texttt{length}(y)$, 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 $\texttt{length}(y) = \texttt{length}(x){+}2$. This triggers MATLAB to implement the complete spline approach.

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

- Syntax:

  $v = \texttt{ppval}(S, u)$

  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 = 1: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 $h_1 = h_2 = \cdots = h_{n-1} = h$ ($h_k = x_{k+1} - x_k$). For the not-a-knot method, we have:

$$|f(x) - S(x)| \le \frac{5}{384} ||f''''||_\infty h^4$$

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

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

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

Note, though, that the computation of the piecewise cubic method was *very local* and simple (Every interval could be independently evaluated.) while the computation of the coefficients of the cubic spline is more elaborate.

Cubic Hermite Splines

We will now consider a different approach to piecewise cubic polynomial interpolation. In particular, given n x-values (in ascending order)

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

and n associated y-values (sampled from a function $f(x)$)

$$y_1, y_2, \ldots, y_{n-1}, y_n, \text{ where } y_k = f(x_k)$$

and assume we also know the derivative $f'(x)$ at the same locations, denoted by:

$$y'_1, y'_2, \ldots, y'_{n-1}, y'_n, \text{ where } y'_k = f'(x_k)$$

As with other methods based on piecewise polynomials, we construct the interpolant as

$$S(x) = \begin{cases} S_1(x), & x \in I_1 \\ S_2(x), & x \in I_2 \\ \quad\vdots & \\ S_k(x), & x \in I_k \\ \quad\vdots & \\ S_{n-1}(x), & x \in I_{n-1} \end{cases} \quad \text{where } I_k = [x_k, x_{k+1}]$$

In this case, each individual $S_k(x)$ is constructed to match both the function values y_k, y_{k+1} as well as the derivatives y'_k, y'_{k+1} at the endpoints of I_k. In detail:

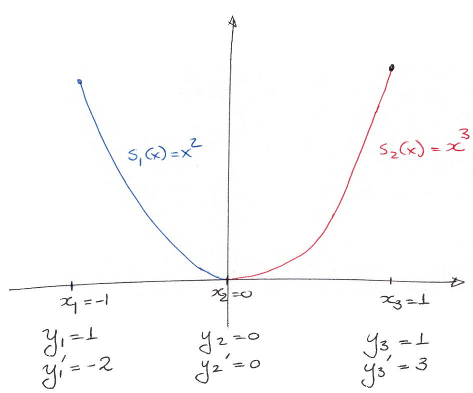
$$\left.\begin{aligned} S_k(x_k) &= y_k \\ S_k(x_{k+1}) &= y_{k+1} \\ S'_k(x_k) &= y'_k \\ S'_k(x_{k+1}) &= y'_{k+1} \end{aligned}\right\} (*)$$

Since $S_k(x) = a_3x^3 + a_2x^2 + a_1x + a_0$ has 4 unknown coefficients, the 4 equations $(*)$ uniquely define the appropriate values of $a_0, \ldots, a_3$. Also note that:

Lecture of: 14 Mar 2013

$$\begin{aligned} S_k(x_{k+1}) &= y_{k+1} = S_{k+1}(x_{k+1}) \\ \text{and } S'_k(x_{k+1}) &= y'_{k+1} = S'_{k+1}(x_{k+1}) \end{aligned}$$

Thus, the resulting interpolant $S(x)$ is *continuous* with continuous derivatives (e.g. a C^1 function). However, we do not strictly enforce that the *2nd derivative* should be continuous, and in fact, it generally will *not* be:



In this case $S''_1(0) = 2$, while $S''_2(0) = 0$.