Topic 9. Splines

Spring 2018

1 Introduction

A spline is a piecewise polynomial function.

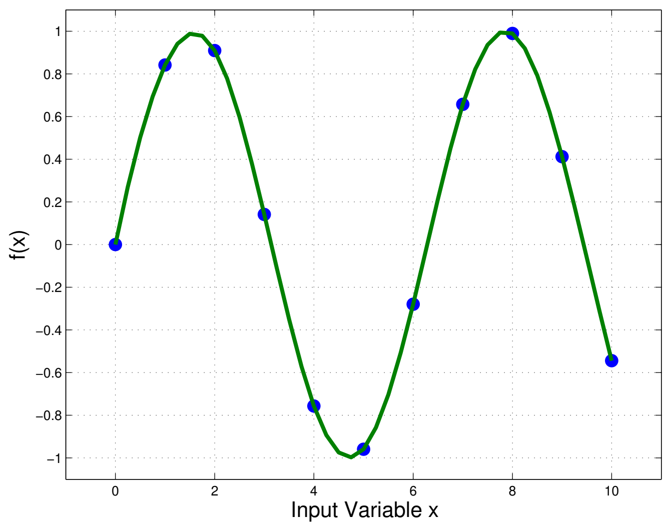


Figure 1: A spline example. x_i's: knots or nodes.

Definition: Assume $f(x_i) = f_i$ of the function $f(x)$ at the points $x_0 < x_1 < x_2 < ... < x_n$. A natural cubic interpolating spline, $s(x)$, is a function on the interval $[x_0, x_n]$ satisfying:

a. $s(x)$ is a cubic polynomial on each node-to-node interval $[x_i, x_{i+1}]$;

b. $s(x_i) = f_i$ at each node x_i;

c. the 2nd derivative $s''(x)$ exists and is continuous throughout the entire interval $[x_0, x_n]$;

d. at the terminal nodes, $s''(x_0) = s''(x_n) = 0$.

Proposition 1.1 (9.2.1. in Lange). *There is exactly one function $s(x)$ on $[x_0, x_n]$ satisfying the above properties*.

Proof. Skipped.
It is also intuitive that the above leads to a well-defined mathematical problem.

- There are n subintervals. Each cubic polynomial within a subinterval is determined by four parameters, imagining $c_3x^3 + c_2x^2 + c_1x + c_0$. So the total number of parameters can be $4n$.
- We have:
  - $2n$ 0-order conditions ($s(x_i) = f_i$),
  - $n-1$ 1-order conditions ($s'(x)$ is continuous at $x_i, 1 \le i \le 1$),
  - $n+1$ 2nd order conditions ($s''(x)$ is continuous at $x_i, 1 \le i \le 1$ and $s''(x_0) = s''(x_n) = 0$).

The number of conditions matches the number of parameters. The above problem is solvable *necessarily*. For *sufficiency*, we must study more. See the next section.

2 Computation for a Spline

Notations:

- $h_i = x_{i+1} - x_i$, interdistances.
- $\sigma_i = s''(x_i)$. (Second derivatives turn out to be a good way to parameterize the spline. More can be seen later.)
- $s_i(x) = s(x)$ for $x \in [x_i, x_{i+1}]$. Just a notation for later description.

$s_i''(x)$ is linear within $[x_i, x_{i+1}]$:

$$\begin{aligned} s_i''(x) &= \sigma_i \frac{x_{i+1} - x}{h_i} + \sigma_{i+1} \frac{x - x_i}{h_i} \\ \Rightarrow s_i(x) &= \frac{\sigma_i}{6h_i}(x_{i+1} - x)^3 + \frac{\sigma_{i+1}}{6h_i}(x - x_i)^3 + c_1(x - x_i) + c_2(x_{i+1} - x). \end{aligned}$$

Since

$$\begin{aligned} f_i &= s_i(x_i) = \frac{\sigma_i}{6}h_i^2 + c_2 h_i \\ f_{i+1} &= s_i(x_{i+1}) = \frac{\sigma_{i+1}}{6}h_i^2 + c_1 h_i. \end{aligned}$$

把c1，c2解出来

We have:

$$\begin{aligned} s_i(x) &= \frac{\sigma_i}{6h_i}(x_{i+1} - x)^3 + \frac{\sigma_{i+1}}{6h_i}(x - x_i)^3 + \left(\frac{f_{i+1}}{h_i} - \frac{\sigma_{i+1}h_i}{6}\right)(x - x_i) + \left(\frac{f_i}{h_i} - \frac{\sigma_i h_i}{6}\right)(x_{i+1} - x), \\ s_i'(x) &= -\frac{\sigma_i}{2h_i}(x_{i+1} - x)^2 + \frac{\sigma_{i+1}}{2h_i}(x - x_i)^2 + \frac{f_{i+1} - f_i}{h_i} - \frac{h_i}{6}(\sigma_{i+1} - \sigma_i). \end{aligned}$$

Since $s_{i-1}'(x_i) = s_i'(x_i)$, we have: for $1 \le i \le n-1$,

$$\frac{1}{6}h_{i-1}\sigma_{i-1} + \frac{1}{3}(h_{i-1} + h_i)\sigma_i + \frac{1}{6}h_i\sigma_{i+1} = \frac{f_{i+1} - f_i}{h_i} - \frac{f_i - f_{i-1}}{h_{i-1}}. \quad (2.1)$$

Equation (2.1) gives $n-1$ equations. Recall $\sigma_0 = \sigma_n = 0$, there are $n-1$ parameters. Solving the system of linear equations derived from (2.1) will give a set of $\sigma_1, \sigma_2, \cdots, \sigma_{n-1}$, which determine the spline function $s(x)$.

We write out the above system of linear equations. We hope to identify a fast numerical approach to solve it. The system of linear equations is:

$$\begin{array}{rlll}
 & \frac{1}{3}(h_0+h_1)\sigma_1 & +\dfrac{1}{6}h_1\sigma_2 = & \dfrac{f_2-f_1}{h_1} - \dfrac{f_1-f_0}{h_0}, \\
\dfrac{1}{6}h_1\sigma_1 + & \frac{1}{3}(h_1+h_2)\sigma_2 & +\dfrac{1}{6}h_2\sigma_3 = & \dfrac{f_3-f_2}{h_2} - \dfrac{f_2-f_1}{h_1}, \\
\dfrac{1}{6}h_2\sigma_2 + & \frac{1}{3}(h_2+h_3)\sigma_3 & +\dfrac{1}{6}h_3\sigma_4 = & \dfrac{f_4-f_3}{h_3} - \dfrac{f_3-f_2}{h_2}, \\
 & & \vdots & \\
\dfrac{1}{6}h_{n-3}\sigma_{n-3} + & \frac{1}{3}(h_{n-3}+h_{n-2})\sigma_{n-2} & +\dfrac{1}{6}h_{n-2}\sigma_{n-1} = & \dfrac{f_{n-1}-f_{n-2}}{h_{n-2}} - \dfrac{f_{n-2}-f_{n-3}}{h_{n-3}}, \\
\dfrac{1}{6}h_{n-2}\sigma_{n-2} + & \frac{1}{3}(h_{n-2}+h_{n-1})\sigma_{n-1} & = & \dfrac{f_n-f_{n-1}}{h_{n-1}} - \dfrac{f_{n-1}-f_{n-2}}{h_{n-2}}.
\end{array}$$

The above can be rewritten as

$$\mathbf{M} \cdot \sigma = \mathbf{y},$$

where vectors σ and $\mathbf{y}$ are

$$\sigma = \begin{pmatrix} \sigma_1 \\ \vdots \\ \sigma_{n-1} \end{pmatrix}, \quad \mathbf{y} = \begin{pmatrix} \frac{f_2-f_1}{h_1} - \frac{f_1-f_0}{h_0} \\ \vdots \\ \frac{f_n-f_{n-1}}{h_{n-1}} - \frac{f_{n-1}-f_{n-2}}{h_{n-2}} \end{pmatrix},$$

and matrix $\mathbf{M}$ is defined as

$$\mathbf{M} = \begin{pmatrix}
\frac{1}{3}(h_0+h_1) & h_1/6 & 0 & \cdots & 0 & 0 \\
h_1/6 & \frac{1}{3}(h_1+h_2) & h_2/6 & \cdots & 0 & 0 \\
0 & h_2/6 & \frac{1}{3}(h_2+h_3) & \cdots & 0 & 0 \\
\vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\
0 & 0 & 0 & \cdots & \frac{1}{3}(h_{n-3}+h_{n-2}) & h_{n-2}/6 \\
0 & 0 & 0 & \cdots & h_{n-2}/6 & \frac{1}{3}(h_{n-2}+h_{n-1})
\end{pmatrix}_{(n-1)*(n-1)}$$

Matrix $\mathbf{M}$ is symmetric and positive definite. Moreover, $\mathbf{M}$ is tridiagonal. It is known in linear algebra that $\mathbf{M}$ can be decomposed as $\mathbf{M} = \mathbf{L}\mathbf{D}\mathbf{L}^T$ (Cholesky decomposition), where $\mathbf{L}$ has the form

$$\mathbf{L} = \begin{pmatrix}
1 & & & \cdots & 0 \\
a_1 & 1 & & & \vdots \\
 & \ddots & \ddots & & \\
\vdots & & \ddots & & \\
0 & \cdots & & a_{n-2} & 1
\end{pmatrix}$$

and $\mathbf{D}$ is diagonal

$$\mathbf{D} = \begin{pmatrix} d_1 & & \\ & \ddots & \\ & & d_{n-1} \end{pmatrix}.$$

To make the above hold, we need the following equations:

$$\begin{aligned} \frac{1}{3}(h_0 + h_1) = \mathbf{M}_{11} &= d_1, \\ h_i/6 = \mathbf{M}_{i,i+1} &= d_i a_i, \qquad i = 1, 2, \ldots, n-2, \\ \frac{1}{3}(h_{i-1} + h_i) = \mathbf{M}_{ii} &= d_{i-1}a_{i-1}^2 + d_i, \qquad i = 2, 3, \ldots, n-1. \end{aligned}$$

Hence we can compute for a_i's and d_i's iteratively:

```
$d_1 = \frac{1}{3}(h_0 + h_1)$;
For $i = 1, 2, \ldots, n-2$,
    $a_i = \mathbf{M}_{i,i+1}/d_i = \frac{h_i}{6d_i}$;
    $d_{i+1} = \mathbf{M}_{i+1,i+1} - d_i a_i^2 = \frac{1}{3}(h_i + h_{i+1}) - \frac{h_i^2}{36d_i}$;
end;
```

According to the above, the matrices $\mathbf{L}$ and $\mathbf{D}$ can be computed at order $O(n)$. Note that

$$\sigma = \mathbf{M}^{-1}\mathbf{y} = \underline{(\mathbf{L}^T)}^{-1}\mathbf{D}^{-1}\overbrace{\mathbf{L}^{-1}\mathbf{y}}^{O(n)}.$$

Solving a system involved with matrix $\mathbf{L}$ or $\mathbf{D}$ can be done easily, with $O(n)$. Overall, the complexity of computing for a spline is $O(n)$, i.e., the same order as the data size.

3 Basic Properties

Proposition 3.1 (9.2.2. in Lange). *Let $s(x)$ be the spline interpolating the function $f(x)$ at the nodes $x_0 < x_1 < \cdots < x_n$. If $g(x)$ is any other twice continuously differentiable function interpolating $f(x)$ at these nodes, then*

$$\int_{x_0}^{x_n} g''(x)^2 \geq \int_{x_0}^{x_n} s''(x)^2$$

with equality only if $g(x) = s(x)$ throughout $[x_0, x_n]$.

Proof. Notice that we have the following.

$$\begin{aligned} \int_{x_0}^{x_n} [g''(x)]^2 - \int_{x_0}^{x_n} [s''(x)]^2 &= \int_{x_0}^{x_n} \left\{(g''(x) - s''(x))^2 + 2g''(x)s''(x) - 2[s''(x)]^2\right\} \\ &= \int_{x_0}^{x_n} [g''(x) - s''(x)]^2 dx + 2\int_{x_0}^{x_n} s''(x)[g''(x) - s''(x)]. \quad (3.2) \end{aligned}$$

If we can show that the second term in the above expression is zero, then the proposition holds. To do so, we consider the following term. We adopt the integration by part.

$$\int_{x_i}^{x_{i+1}} s''(x)[g''(x) - s''(x)] = s''(x)[g'(x) - s'(x)] \mid_{x_i}^{x_{i+1}} - \int_{x_i}^{x_{i+1}} s'''(x)[g'(x) - s'(x)]. \tag{3.3}$$

For the second term in (3.3), recall that $s'''(x)$ is a constant in $[x_i, x_{i+1}]$, we have

$$\begin{aligned}\int_{x_i}^{x_{i+1}} s'''(x)[g'(x) - s'(x)] &= \text{constant } \cdot \int_{x_i}^{x_{i+1}} [g'(x) - s'(x)] \\ &= \text{constant } \cdot [g(x) - s(x)] \mid_{x_i}^{x_{i+1}} \\ &= \text{constant } \cdot \{[g(x_{i+1}) - s(x_{i+1})] - [g(x_i) - s(x_i)]\} \\ &= 0. \end{aligned} \tag{3.4}$$

On the other hand, we have

$$\begin{aligned}\int_{x_0}^{x_n} s''(x)[g''(x) - s''(x)] &= \sum_{i=0}^{n-1} \int_{x_i}^{x_{i+1}} s''(x)[g''(x) - s''(x)] \\ &\overset{(3.3),(3.4)}{=} \sum_{i=0}^{n-1} s''(x)[g'(x) - s'(x)] \mid_{x_i}^{x_{i+1}} \\ &= \sum_{i=0}^{n-1} s''(x_{i+1})[g'(x_{i+1}) - s'(x_{i+1})] - s''(x_i)[g'(x_i) - s'(x_i)] \\ &= s''(x_n)[g'(x_n) - s'(x_n)] - s''(x_0)[g'(x_0) - s'(x_0)] \\ &= 0.\end{aligned}$$

The last equation is due to $s''(x_n) = s''(x_0) = 0$.

From all the above, we have

$$\int_{x_0}^{x_n} [g''(x)]^2 - \int_{x_0}^{x_n} [s''(x)]^2 = \int_{x_0}^{x_n} [g''(x) - s''(x)]^2 dx \geq 0,$$

and the equality holds if and only if $g''(x) = s''(x)$ for $x \in [x_0, x_n]$, which is equivalent to $g(x) = s(x)$ throughout $[x_0, x_n]$. □

Proposition 3.2. *Suppose that $f(x)$ is twice continuously differentiable and $s(x)$ is the spline interpolating $f(x)$ at the nodes $x_0 < x_1 < \cdots < x_n$. If*

$$h = \max_{0 \leq i \leq n-1} (x_{i+1} - x_i)$$

then

$$\max_{x_0 \leq x \leq x_n} |f(x) - s(x)| \leq h^{\frac{3}{2}} \left[\int_{x_0}^{x_n} f''(y)^2 dy\right]^{\frac{1}{2}},$$

and

$$\max_{x \epsilon [x_0, x_n]} |f'(x) - s'(x)| \leq h^{\frac{1}{2}} \left[\int_{x_0}^{x_n} f''(y)^2 dy\right]^{\frac{1}{2}}.$$

Proof. It happens that the second inequality is relatively easy to establish. Let's assume that $x \in (x_i, x_{i+1})$ for $0 \le i \le n-1$. Because $f(x_i) - s(x_i) = f(x_{i+1}) - s(x_{i+1}) = 0$, we must have $z \in (x_i, x_{i+1})$ such that $f'(z) - s'(z) = 0$ (Rolle's theorem). Hence we have

$$f'(x) - s'(x) = \int_z^x (f''(y) - s''(y))dy.$$

On the other hand, we have

$$\begin{aligned}
|f'(x) - s'(x)| &\le \int_z^x |f''(y) - s''(y)|dy \\
&\overset{\text{Cauchy}}{\le} \sqrt{\int_z^x (f''(y) - s''(y))^2 dy}\sqrt{\int_z^x 1dy} \\
&\le \{\int_{x_0}^{x_n} (f''(y) - s''(y))^2\}^{1/2} \cdot h^{1/2}. \qquad (3.5)
\end{aligned}$$

Recall in the proof of the last proposition, we proved that

$$\int_{x_0}^{x_n} [f''(y) - s''(y)]^2 dy = \int_{x_0}^{x_n} [f''(y)]^2 - \int_{x_0}^{x_n} [s''(y)]^2.$$

From the above, it is evident that

$$\left\{\int_{x_0}^{x_n} (f''(y) - s''(y))^2\right\}^{1/2} \le \left[\int_{x_0}^{x_n} f''(y)^2 dy\right]^{\frac{1}{2}}.$$

Considering the x is arbitrarily chosen, given the above inequality and (3.5), we established the second inequality in the proposition.

To establish the first inequality, we use the following fact:

$$f(x) - s(x) = f(x) - s(x) - [f(x_i) - s(x_i)] = \int_{x_i}^x [f'(y) - s'(y)]dy.$$

The above leads to the following:

$$\begin{aligned}
|f(x) - s(x)| &\le \int_{x_i}^x |f'(y) - s'(y)|dy \\
&\le \max_y |f'(y) - s'(y)| \cdot \int_{x_i}^x 1dy \\
&\le h^{\frac{1}{2}} \left[\int_{x_0}^{x_n} f''(y)^2 dy\right]^{\frac{1}{2}} \cdot h.
\end{aligned}$$

The above is equivalent to the first inequality in the proposition. □

Corollary 3.3. *If $h \to 0$, then $s(x) \to f(x)$ and $s'(x) \to f'(x)$ uniformly.*

4 Spline Basis Functions

Recall that basis is a linear independent set, which span a linear space. Suppose we consider all splines with fixed knots $x_0 < x_1 < \cdots < x_n, n \geq 1$. Let $\mathcal{S}$ denote these splines. We determine a basis of $\mathcal{S}$. It is evident that if we have $s, t \in \mathcal{S}$, then $s + t \in \mathcal{S}$. As a matter of fact, this can be generalized to any linear combinations of splines: any linear combination of splines in $\mathcal{S}$ still belongs to $\mathcal{S}$. So $\mathcal{S}$ is a linear space. On the other hand, let s_i denote a special spline that satisfies the following:

$$\begin{cases} s_i(x_i) = 1, \text{ and} \\ s_i(x_j) = 0, \text{ for } j \neq i, 0 \leq j \leq n; \end{cases} \quad i = 0, 1, 2, \ldots, n.$$

Note that these s_i are redefined. (We used to use them to denote the spline function restricted in interval $[x_i, x_{i+1}]$.) It is easy to verify the following:

$$s(x) = \sum_{i=0}^{n} s(x_i) s_i(x), \quad x \in [x_0, x_n].$$

You can verify that the above is the unique linear combination. This indicates that the set $\{s_i(x) : i = 0, \ldots, n\}$ is a basis of $\mathcal{S}$.

An immediate application of the above analysis is that we can introduce another parametrization of splines. Recall that we have computed a spline function via its second derivatives at interior knots: $\sigma_1, \ldots, \sigma_{n-1}, \sigma_i = s''(x_i), 1 \leq i \leq n-1$. The advantage of doing so is that the computational complexity can be controlled at $O(n)$. On the hand hand, based on the above basis analysis, we can parameterize a spline by its interpolating values: $f_0, f_1, \ldots, f_n$, where $f_i = f(x_i), 0 \leq i \leq n$; i.e., f_i's are the functional values at those knots. Recall that in Section 2, we established

$$\mathbf{M} \cdot \sigma = \mathbf{y},$$

where $\mathbf{M}, \sigma$, and $\mathbf{y}$ were defined there. We have

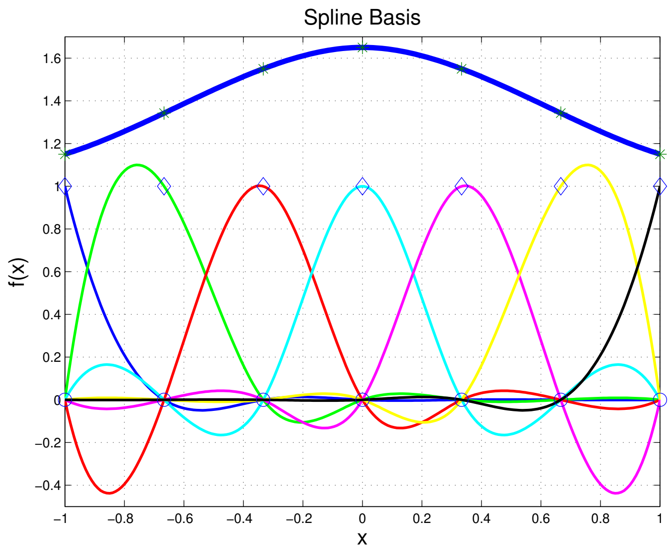
$$\mathbf{y} = \mathbf{Q} \cdot \mathbf{f},$$

where

$$\mathbf{Q} = \begin{pmatrix} \frac{1}{h_0} & -\frac{1}{h_0} - \frac{1}{h_1} & \frac{1}{h_1} & & & \\ & \frac{1}{h_1} & -\frac{1}{h_1} - \frac{1}{h_2} & \frac{1}{h_2} & & \\ & & \ddots & \ddots & \ddots & \\ & & & \frac{1}{h_{n-2}} & -\frac{1}{h_{n-2}} - \frac{1}{h_{n-1}} & \frac{1}{h_{n-1}} \end{pmatrix}_{(n-1)*(n+1)} \text{ and } \mathbf{f} = \begin{pmatrix} f_0 \\ f_1 \\ \vdots \\ f_n \end{pmatrix}.$$

The above is evident from the definition of $\mathbf{y}$. Note that $\mathbf{Q} \in \mathbb{R}^{(n-1)\times(n+1)}$, so $\mathbf{Q}$ can not have full column rank. We will come back to this when we fit noisy observations, seeing **5-2**.

Below is an illustration of spline basis.



5 Applications

5-1: Function Integration.
Recall that

$$s_i(x) = \frac{\sigma_i}{6h_i}(x_{i+1}-x)^3 + \frac{\sigma_{i+1}}{6h_i}(x-x_i)^3 + \left(\frac{f_{i+1}}{h_i} - \frac{\sigma_{i+1}h_i}{6}\right)(x-x_i) + \left(\frac{f_i}{h_i} - \frac{\sigma_i h_i}{6}\right)(x_{i+1}-x). \quad (5.6)$$

We can easily verify the following:

$$\begin{aligned}
\int_{x_i}^{x_{i+1}} (x_{i+1}-x)^3 dx &= -\int_{h_i}^{0} y^3 dy = \frac{1}{4}h_i^4, \\
\int_{x_i}^{x_{i+1}} (x-x_i)^3 dx &= \int_{0}^{h_i} y^3 dy = \frac{1}{4}h_i^4, \\
\int_{x_i}^{x_{i+1}} (x-x_i) dx &= \int_{0}^{h_i} y dy = \frac{1}{2}h_i^2, \\
\int_{x_i}^{x_{i+1}} (x_{i+1}-x) dx &= -\int_{h_i}^{0} y dy = \frac{1}{2}h_i^2.
\end{aligned}$$

Hence, after simplification, we have

$$\int_{x_i}^{x_{i+1}} s_i(x) dx = \frac{f_i + f_{i+1}}{2}h_i - \frac{\sigma_i + \sigma_{i+1}}{24}h_i^3.$$

It follows that

$$\begin{aligned}\int_{x_0}^{x_n} f(x)dx &\approx \int_{x_0}^{x_n} s(x)dx \\ &= \sum_{i=0}^{n-1}\left[\frac{f_i+f_{i+1}}{2}h_i - \frac{\sigma_i+\sigma_{i+1}}{24}h_i^3\right].\end{aligned}$$

Note that we must compute σ_i's in order to use the above formula. We can also establish the following theoretical result.

Proposition 5.1.

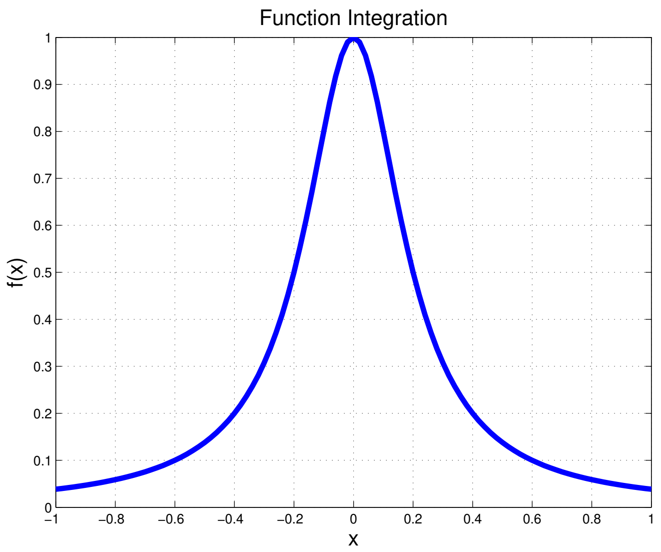
$$\left|\int_{x_0}^{x_n} f(x)dx - \int_{x_0}^{x_n} s(x)dx\right| \le \frac{h^4}{16}(x_n - x_0)\max_{x_0\le x\le x_n}|f^{(4)}(x)|.$$

We are not going to talk about its proof.

Example 5.2. *For function*

$$f(x) = \frac{1}{1+25x^2}, \quad x \in [-1,1],$$

Find $\int_{-1}^{1} f(x)dx$ numerically.



In this case, we can compute for the true value analytically:

$$\int_{-1}^{1}\frac{1}{1+25x^2}dx = \left[\frac{1}{5}\arctan(5x)\right]_{-1}^{1} = (\arctan(5)-\arctan(-5))/5 = 0.549360306778006.$$

We will use the above as a reference value. Notice that the output of Matlab function `spline` provides coefficients of a polynomial $(s_i(x))$ at interval $(x_i, x_{i+1}), 0 \le i \le n-1$. Suppose $s_i(x) =$

$\beta_3 x^3 + \beta_2 x^2 + \beta_1 x + \beta_0$. Values $\beta_0, \beta_1, \beta_2, \beta_3$ are in the output of `spline`. We need to compute for σ_i's. From (5.6), we have

$$\beta_3 = \frac{\sigma_{i+1} - \sigma_i}{6h_i}, \quad \beta_2 = \frac{\sigma_i x_{i+1} - \sigma_{i+1} x_i}{2h_i}.$$

The above is equivalent to a system of linear equations:

$$\begin{cases} \sigma_{i+1} - \sigma_i = 6h_i\beta_3, \\ -x_i\sigma_{i+1} + x_{i+1}\sigma_i = 2h_i\beta_2. \end{cases}$$

Solving the above gives us the formula for σ_i's:

$$\sigma_i = 2\beta_2 + 6\beta_3 x_i, \quad i = 0, 1, 2, \ldots, n-1; \text{ and } \sigma_{i+1} = 2\beta_2 + 6\beta_3 x_{i+1}, \quad i = n-1.$$

The above can also be obtained by $\sigma_i = s''(x_i), i = 0, 1, \ldots, n$. The following Matlab script illustrate how to implement the above.

```
ns = 101;
h = 2/(ns-1);
x = linspace(-1, 1, ns);
f = 1./(1+25.*x.*x);
pp = spline(x,f);
sigma = 2*pp.coefs(:,2)+6.*pp.coefs(:,1).*(x(1:ns-1)');
sigma = [sigma; 2*pp.coefs(end,2)+6.*pp.coefs(end,1).*(x(end))];
int1 = h.*sum(f(1:end-1)+f(2:end))/2;
int2 = h.^3.*sum(sigma(1:end-1)+sigma(2:end))/24;
int = int1 - int2;
fprintf('ns= %7d: int = %12.5e \n', ns, int);
```

The output from running the above script is:

```
ns=     101: int = 5.49341e-001
```

Readers can compare it with the value that was obtained analytically.

One could try different number of intervals, and compare the performance with a more straightforward approximation formula (trapezoidal approximation)

$$\sum_{i=0}^{n-1} \frac{f_i + f_{i+1}}{2} h_i.$$

Can you tell why the above is another approximate?

The following table summarizes the differences between the reference value and the approximates based on spline or the above direct approximate formula. It seems that spline approximate does not significantly improve the approximation here.

number of knots $(n+1)$	Spline	Direct
101	$-1.88956 10^{-5}$	$-4.93062 10^{-6}$
1001	$-1.89646 10^{-7}$	$-4.93096 10^{-8}$
10001	$-1.89653 10^{-9}$	$-4.93098 10^{-10}$
100001	$-1.89659 10^{-11}$	$-4.93161 10^{-12}$
1000001	$-2.05613 10^{-13}$	$-6.52811 10^{-14}$

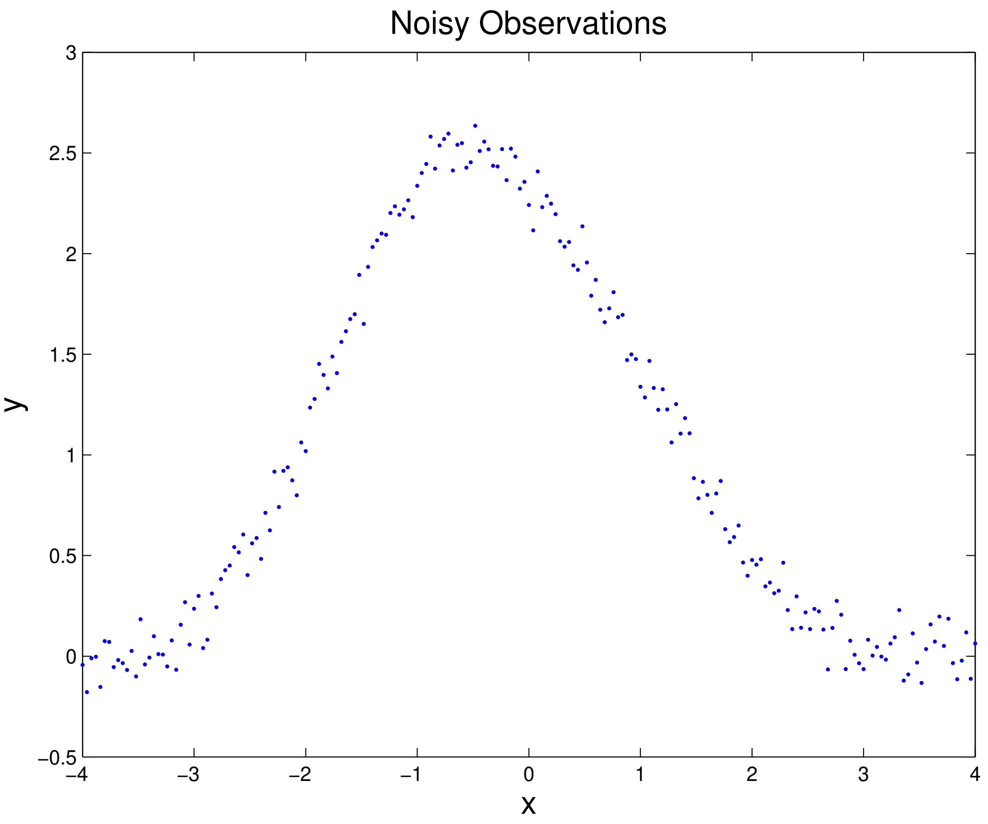
5-2: Application to Nonparametric Regression.

We sketch the main ideas.

- Given response $y_0, y_1, \cdots, y_n$ and weights $w_0, w_1, \cdots, w_n$, $w_i > 0$, we would like to minimize the weighted sum of squares:

$$\min_s \sum_{i=0}^{n} w_i [y_i - s(x_i)]^2,$$

where $x_0 < \cdots < x_n$ are knots, $s(x), x \in [x_0, x_n]$, is a spline function. For simplicity, we can imagine $w_i = 1, \forall i$. Below is a sample noisy data set.



At the same time, we would like to control the smoothness of the spline s. We define the smoothness of s as

$$\int_{x_0}^{x_n} s''(x)^2 dx.$$

- Tradeoff between smoothness of s and goodness of fit. Consider

$$J_\alpha(s) = \alpha \sum_{i=0}^{n} w_i [y_i - s(x_i)]^2 + (1-\alpha)\int_{x_0}^{x_n} s''(x)^2 dx.$$

- We derive formula for the spline $s(x)$ that minimizes $J_\alpha(g)$. Recall we have defined the following notations:

$$\sigma = \begin{pmatrix} \sigma_1 \\ \vdots \\ \sigma_n \end{pmatrix}, \mathbf{f} = \begin{pmatrix} f_0 \\ f_1 \\ \vdots \\ f_n \end{pmatrix} = \begin{pmatrix} s(x_0) \\ s(x_1) \\ \vdots \\ s(x_n) \end{pmatrix}, \mathbf{y} = \begin{pmatrix} y_0 \\ y_1 \\ \vdots \\ y_n \end{pmatrix}.$$

  Recall we also have

$$\mathbf{M} \cdot \sigma = \mathbf{Q} \cdot \mathbf{f},$$

  where matrices $\mathbf{M}$ and $\mathbf{Q}$ are defined earlier. We are going to verify the following

  1. $\int_{x_0}^{x_n} s''(x)^2 dx = \sigma^T \mathbf{M} \sigma$;
  2. $J_\alpha(s) = \alpha(\mathbf{y}-\mathbf{f})^T \mathbf{W}(\mathbf{y}-\mathbf{f}) + (1-\alpha)\mathbf{f}^T \mathbf{Q}^T \mathbf{M}^{-1} \mathbf{Q}\mathbf{f}$, where

$$\mathbf{W} = \begin{pmatrix} w_0 & & & \\ & w_1 & & \\ & & \ddots & \\ & & & w_n \end{pmatrix}.$$

  Proof. (1) For the first equation, we have

$$\int_{x_0}^{x_n} s''(x)^2 dx = \sum_{i=0}^{n-1} \int_{x_i}^{x_{i+1}} s''(x)^2 dx.$$

  Recall that for $x \in [x_i, x_{i+1}]$, we have

$$s''(x) = \sigma_i \frac{x_{i+1}-x}{h_i} + \sigma_{i+1}\frac{x-x_i}{h_i}.$$

  Hence

$$\begin{aligned} & \int_{x_i}^{x_{i+1}} \left(\sigma_i \frac{x_{i+1}-x}{h_i} + \sigma_{i+1}\frac{x-x_i}{h_i}\right)^2 dx \\ = & \sigma_i^2 \int_{x_i}^{x_{i+1}} \left(\frac{x_{i+1}-x}{h_i}\right)^2 dx + \sigma_{i+1}^2 \int_{x_i}^{x_{i+1}} \left(\frac{x-x_i}{h_i}\right)^2 dx + 2\sigma_i\sigma_{i+1}\int_{x_i}^{x_{i+1}} \frac{x_{i+1}-x}{h_i}\frac{x-x_i}{h_i} dx \\ = & \sigma_i^2 h_i/3 + \sigma_{i+1}^2 h_i/3 + \sigma_i\sigma_{i+1}h_i/3. \end{aligned}$$

  Hence

$$\int_{x_0}^{x_n} s''(x)^2 dx = \sigma^T \cdot \mathbf{M} \cdot \sigma.$$

(2) For the second equation, we use the following facts:

$$\sum_{i=0}^{n} w_i[y_i - s(x_i)]^2 = (\mathbf{y}-\mathbf{f})^T\mathbf{W}(\mathbf{y}-\mathbf{f}), \tag{5.7}$$

$$\sigma = \mathbf{M}^{-1}\mathbf{Q}\cdot\mathbf{f}. \tag{5.8}$$

We skip the details. □

- To minimize $J_\alpha(S)$, we consider the first order condition. We give the result:

$$\hat{\mathbf{f}} = [\alpha\mathbf{W} + (1-\alpha)\mathbf{Q}^T\mathbf{M}^{-1}\mathbf{Q}]^{-1}\alpha\mathbf{W}\mathbf{y}.$$

You can also show that for σ:

$$\hat{\sigma} = [\alpha\mathbf{M} + (1-\alpha)\mathbf{Q}\mathbf{W}^{-1}\mathbf{Q}^T]^{-1}\alpha\mathbf{Q}\mathbf{y}.$$

Remark: the above can be implemented numerically.

We add some details here. Recall the first order condition is

$$\mathbf{0} = \frac{dJ_\alpha(s)}{d\mathbf{f}} = -2\alpha\mathbf{W}(\mathbf{y}-\mathbf{f}) + 2(1-\alpha)\mathbf{Q}^T\mathbf{M}^{-1}\mathbf{Q}\mathbf{f}.$$

The above leads to the formula for $\hat{\mathbf{f}}$. On the other hand, the above is equivalent to

$$\alpha\mathbf{W}(\mathbf{y}-\mathbf{f}) = (1-\alpha)\mathbf{Q}^T\sigma.$$

Multiplying both sides by $\mathbf{Q}\mathbf{W}^{-1}$, we have

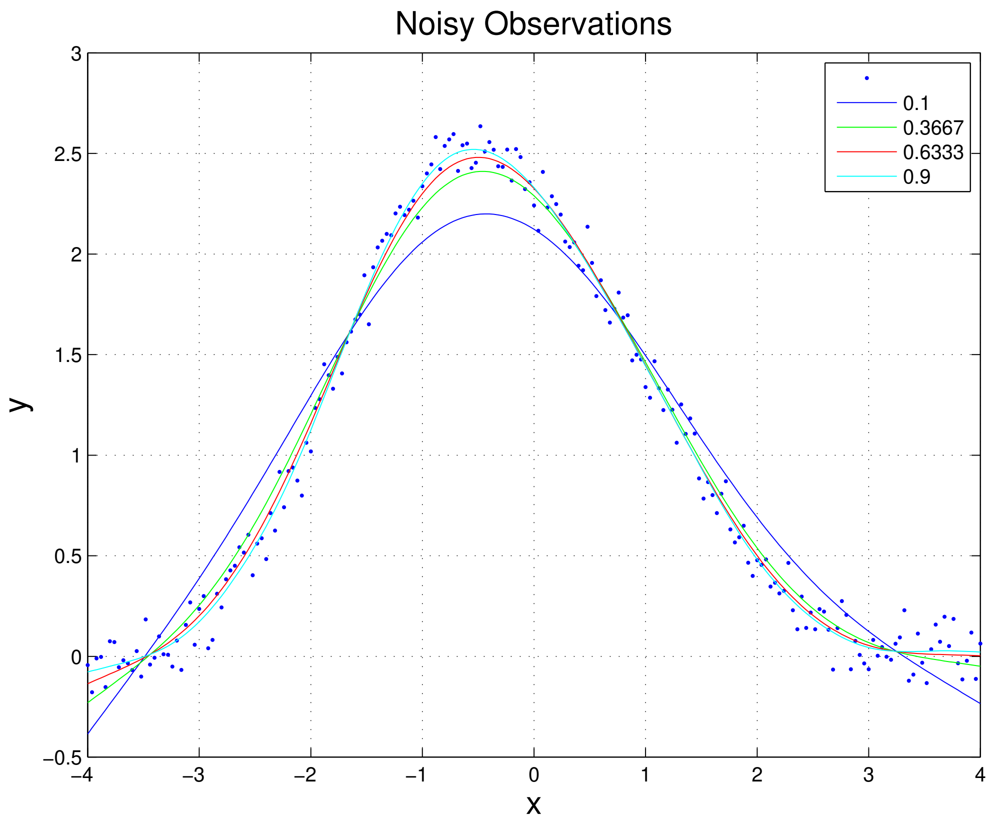
$$\alpha\mathbf{Q}(\mathbf{y}-\mathbf{f}) = (1-\alpha)\mathbf{Q}\mathbf{W}^{-1}\mathbf{Q}^T\sigma,$$

Recalling $\mathbf{M}\cdot\sigma = \mathbf{Q}\cdot\mathbf{f}$, we have

$$\alpha\mathbf{Q}\mathbf{y} = \alpha\mathbf{M}\cdot\sigma + (1-\alpha)\mathbf{Q}\mathbf{W}^{-1}\mathbf{Q}^T\sigma.$$

The above leads to the formula of $\widehat{\sigma}$.

The figure below shows the splines fitted with the noisy observations, with different values of α.



The Matlab codes are:

```
x = -4:4;
y = [0 .15 1.12 2.36 2.36 1.46 .49 .06 0];
cs = spline(x,y);
np1 = 201;
xx = linspace(-4,4,np1);
yy = ppval(cs,xx) + randn(size(xx)).*0.1;
plot(xx,yy,'.'); hold on;
title('Noisy Observations','fontsize',16)
xlabel('x','fontsize',16)
ylabel('y','fontsize',16)
print -depsc2 spline05

W = speye(np1);
h = 8/(np1-1);
bQ = zeros(np1-2,np1);
for i=1:np1-2,
    bQ(i,i+[0 1 2]) = [1 -2 1]./h;
end;
bM = 2/3*h.*diag(ones(np1-2,1));
```

```
for i=1:np1-3,
    bM(i,i+1) = h/6;
    bM(i+1,i) = h/6;
end;
colorst = ['b'; 'g'; 'r'; 'c'];
alpha_st = linspace(0.1, 0.9, 4);
for ind=1:4,
    alpha = alpha_st(ind);
    fhat = inv(alpha.*W + (1-alpha).*bQ'*inv(bM)*bQ)*alpha*W*(yy');
    plot(xx,fhat,colorst(ind,:));
end;
grid on; legend('','0.1','0.3667','0.6333','0.9')
print -depsc2 spline06
```

A1: Matlab Script for Figure 1

The following Matlab script generates a sine curve, then samples the spline over a finer mesh.

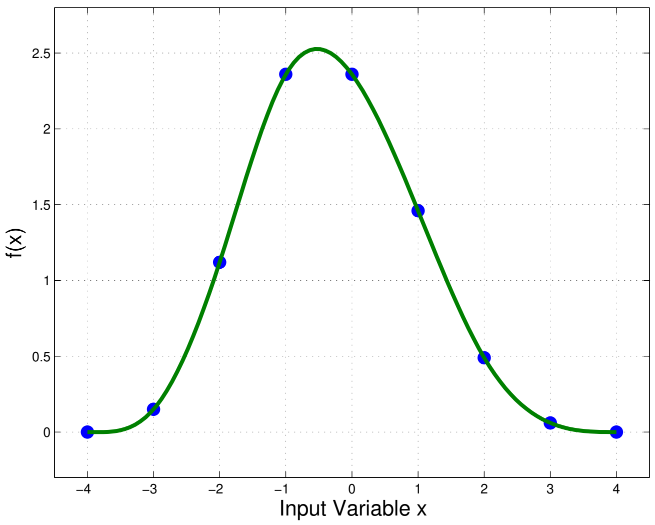
```
x = 0:10;
y = sin(x);
xx = 0:.25:10;
yy = spline(x,y,xx);
p1 = plot(x,y,'.',xx,yy);
set(p1(1),'markersize',30);
set(p1(2),'linewidth',3);
xlabel('Input Variable x','fontsize',16)
ylabel('f(x)','fontsize',16)
grid on;
xlim([-1 11])
ylim([-1.1 1.1])
print -depsc2 spline01
```

A2: Second Matlab Example

The following illustrates the use of clamped or complete spline interpolation where end slopes are prescribed. Zero slopes at the ends of an interpolant to the values of a certain distribution are enforced.

```
x = -4:4;
y = [0 .15 1.12 2.36 2.36 1.46 .49 .06 0];
cs = spline(x,[0 y 0]);
xx = linspace(-4,4,101);
p1 = plot(x,y,'.',xx,ppval(cs,xx),'-');
set(p1(1),'markersize',30);
set(p1(2),'linewidth',3);
xlabel('Input Variable x','fontsize',16)
ylabel('f(x)','fontsize',16)
grid on; xlim([-4.5 4.5]); ylim([-0.3 2.8])
print -depsc2 spline02
```

The generated figure is below.


f(x)
Input Variable x