Johnson and Riess (1982)

5.2.7 Interpolation in Several Variables and Bicubic Splines

The problem of interpolating in a two-way table is fairly common in many engineering and scientific applications. Specifically if $f(x, y)$ is a function of two variables that has been tabulated at the points (x_i, y_j), $0 \le i \le n$, $0 \le j \le m$, then we can construct a two-way table that represents this information:

	x_0	x_1	$\cdots$	x_n
y_0	$f(x_0, y_0)$	$f(x_1, y_0)$	$\cdots$	$f(x_n, y_0)$
y_1	$f(x_0, y_1)$	$f(x_1, y_1)$	$\cdots$	$f(x_n, y_1)$
$\vdots$	$\vdots$			$\vdots$
y_m	$f(x_0, y_m)$	$f(x_1, y_m)$	$\cdots$	$f(x_n, y_m)$

If we want to estimate $f(x, y)$ at a point $(\hat{x}, \hat{y})$, which is not in the table, we are faced with an interpolation problem. We will see momentarily that we can construct a polynomial in x and y of the form

$$p(x, y) = \sum_{r=0}^{n} \sum_{s=0}^{m} a_{rs} x^r y^s,$$

which satisfies $p(x_i, y_j) = f(x_i, y_j)$, $0 \le i \le n$, $0 \le j \le m$. Given this result, we can estimate $f(x, y)$ at $(\hat{x}, \hat{y})$ by $f(\hat{x}, \hat{y}) \approx p(\hat{x}, \hat{y})$.

A variation of the problem of interpolating in a two-way table is the problem of approximating a function $f(x, y)$ of two variables; this problem is sometimes called ''surface fitting.'' The graph of the function $z = f(x, y)$ is a surface in three-space, and we can ask for a simple function $p(x, y)$ such that $p(x, y) \simeq f(x, y)$. As an application, if we want the normal to the surface $z = f(x, y)$ at the point $(\hat{x}, \hat{y}, f(\hat{x}, \hat{y}))$, we can estimate the normal by finding the normal to $z = p(x, y)$.

If we restrict our approximation problem and consider how we might approximate only functions $f(x, y)$ that are defined over a *rectangular* region in the xy-plane, then we can very easily extend our previous results for polynomial and spline approximation. So for simplicity we consider how we can approximate $f(x, y)$ only for (x, y) in R where

$$R = \{(x, y)\colon a \le x \le b,\ c \le y \le d\}.$$

We first treat the problem of interpolating $f(x, y)$ by polynomials in two variables and then consider the case of bicubic spline approximation. To begin, suppose $a \le x_0 < x_1 < \cdots < x_n \le b$ and $c \le y_0 < y_1 < \cdots < y_m \le d$; and let $\pi_{nm} = \{(x_i, y_j) : 0 \le i \le n, 0 \le j \le m\}$; that is, π_{nm} is a rectangular grid of $(n + 1)(m + 1)$ points in R. Given a function $f(x, y)$, we would like to find a polynomial in two variables, $p(x, y)$, such that $p(x_i, y_j) = f(x_i, y_j)$ for all (x_i, y_j) in the grid π_{nm}. Now if we define $\mathcal{P}_{nm}$ by

$$\mathcal{P}_{nm} = \{p(x, y)\ 1{:}p(x, y) = \sum_{i=0}^{n} \sum_{j=0}^{m} a_{ij}x^i y^j, \text{ for all real } a_{ij}\},$$

then each $p(x, y)$ has $(n + 1)(m + 1)$ coefficients. We then hope that we could choose these coefficients to satisfy the $(n + 1)(m + 1)$ interpolation constraints $p(x_i, y_j) = f(x_i, y_j)$, $0 \le i \le n$, $0 \le j \le m$.

Following the lines of one-variable polynomial interpolation in Section 5.2, we define the Lagrange polynomials of two variables by

$$\ell_{ij}(x, y) = \ell_i(x)\tilde{\ell}_j(y), \qquad 0 \le i \le n,\ 0 \le j \le m$$

where $\ell_i(x)$ is given in (5.2) and $\tilde{\ell}_j(y)$ is defined similarly for the points $c = y_0 < y_1 < \cdots < y_m = d$. Thus we have $\ell_i(x_k) = \delta_{ik}$ and $\tilde{\ell}_j(y_k) = \delta_{jk}$; and so $\ell_{ij}(x_r, y_s) = 1$ if $i = r$ and $j = s$; and $\ell_{ij}(x_r, y_s) = 0$ otherwise. From this we see that

$$p(x, y) = \sum_{i=0}^{n} \sum_{j=0}^{m} f(x_i, y_j)\ell_{ij}(x, y)$$

is a polynomial in $\mathcal{P}_{nm}$ that interpolates $f(x, y)$ on the set of points π_{nm}, and we will call this form of $p(x, y)$ the Lagrange form of the interpolating polynomial.

To see that $p(x, y)$ is unique in $\mathcal{P}_{nm}$, suppose that $q(x, y)$ in $\mathcal{P}_{nm}$ satisfies $q(x_i, y_j) = f(x_i, y_j)$, $0 \le i \le n$, $0 \le j \le m$, where $q(x, y) = \sum_{i=0}^{n} \sum_{j=0}^{m} b_{ij}x^i y^j$. Let us rewrite $q(x, y)$ as

$$q(x, y) = \sum_{i=0}^{n} x^i \sum_{j=0}^{m} b_{ij}y^j = \sum_{i=0}^{n} x^i q_i(y)$$

where

$$q_i(y) = \sum_{j=0}^{m} b_{ij}y^j.$$

If we set $y = y_s$ for a fixed value of s, $0 \le s \le m$, then $q(x, y_s)$ interpolates $f(x, y_s)$ at $x = x_0, x_1, \ldots, x_n$. Thus $q(x, y_s)$ is a polynomial only in x, and its coefficients $q_i(y_s)$ are uniquely determined (see Theorem 5.3) by the data $f(x_0, y_s), f(x_1, y_s) \ldots, f(x_n, y_s)$.

Knowing that each $q_i(y_s)$ is determined uniquely by the data $f(x_i, y_s)$, $0 \le i \le n$, $0 \le s \le m$, we can show that the b_{ij} are also uniquely determined. If we fix i and consider the system of $(m + 1)$ equations

$$q_i(y_s) = b_{i0} + b_{i1}y_s + \cdots + b_{im}y_s^m, \qquad 0 \le s \le m,$$

then it is clear that $b_{i0}, b_{i1}, \ldots, b_{im}$ are uniquely determined by $q_i(y_0)$, $q_i(y_1), \ldots, q_i(y_m)$ since the coefficient matrix for the system is a Vandermonde matrix. Thus we have shown that interpolation by $p(x, y)$ is unique.

For computational purposes, it is frequently useful to express the interpolating polynomial $p(x, y)$ in the form

$$p(x, y) = \sum_{i=0}^{n} \ell_i(x) \sum_{j=0}^{m} f(x_i, y_j)\tilde{\ell}_j(y) = \sum_{i=0}^{n} \ell_i(x) p_i(y)$$

where

$$p_i(y) = \sum_{j=0}^{m} f(x_i, y_j)\tilde{\ell}_j(y).$$

As we noted before, $p_i(y)$ is a one-variable polynomial (in y) interpolating $f(x, y)$ along the line $x = x_i$, at the points $(x_i, y_0), (x_i, y_1), \ldots, (x_i, y_m)$. This means that $p(x, y)$ can be built up using one-variable polynomial interpolation. For example, $p(\hat{x}, \hat{y})$ is given by

$$p(\hat{x}, \hat{y}) = \sum_{i=0}^{n} \ell_i(\hat{x}) p_i(\hat{y}).$$

From the expression above, $p(\hat{x}, \hat{y}) = q(\hat{x})$ where $q(x)$ is the polynomial interpolating the data $p_0(\hat{y}), p_1(\hat{y}), \ldots, p_n(\hat{y})$, at the points $x = x_i$, $0 \le i \le n$. To get the values $p_i(\hat{y})$, we interpolate the data $f(x_i, y_j)$, $0 \le j \le m$, by $p_i(y)$ and evaluate $p_i(y)$ at $y = \hat{y}$.

This two-dimensional interpolation scheme can be rephrased thus.

1. For each fixed grid line $x = x_i$, interpolate the data $f(x_i, y_j)$ in the y-direction at the knots $y_0, y_1, \ldots, y_m$ and evaluate the (one-variable) interpolating polynomial $p_i(y)$ at $y = \hat{y}$.
2. Interpolate the values $p_0(\hat{y}), p_1(\hat{y}), \ldots, p_n(\hat{y})$ in the x-direction at the knots $x_0, x_1, \ldots, x_n$ and evaluate the (one-variable) interpolating polynomial $q(x)$ at $x = \hat{x}$. The result, $q(\hat{x})$, is equal to $p(\hat{x}, \hat{y})$.

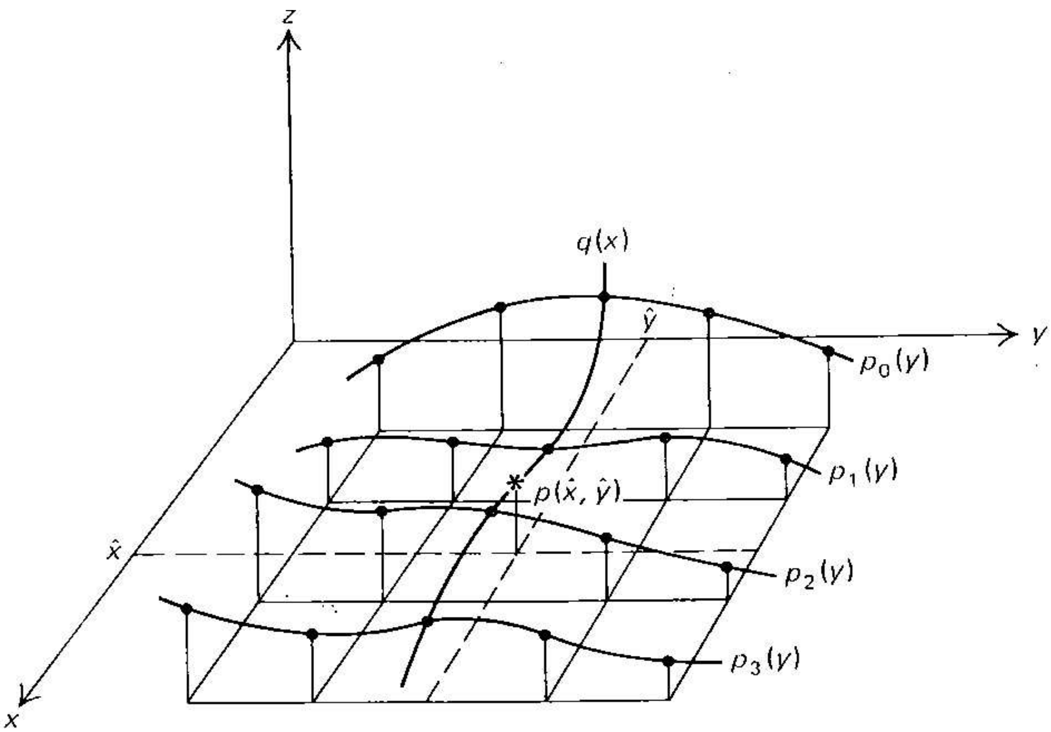

Figure 5.7 A wire-frame model illustrating the calculation $p(\hat{x}, \hat{y})$.

This algorithm can be illustrated as a "wire-frame" model as in Fig. 5.7. We also emphasize what is probably obvious: an algorithm to implement two-dimensional interpolation can be constructed from any routine for one-dimensional interpolation. For step (1) above, we call a polynomial interpolator $(n + 1)$ times to obtain the values $p_0(\hat{y}), p_1(\hat{y}), \ldots, p_n(\hat{y})$ and then call it once more to get $q(\hat{x}) = p(\hat{x}, \hat{y})$.

Writing $p(x, y)$ in the form

$$p(x, y) = \sum_{i=0}^{n} \sum_{j=0}^{m} \ell_i(x)\tilde{\ell}_j(y)f(x_i, y_i)$$

makes it clear (see Fig. 5.7) that we could equally well construct one-dimensional interpolators in the x-direction (at the grid lines $y = y_j$), evaluate these at $x = \hat{x}$, and then pass a one-dimensional interpolator through these data along the line $x = \hat{x}$ in the y-direction. In addition, from the form of $p(x, y)$ above, we see how to calculate quantities such as $p_{xy}(\hat{x}, \hat{y})$. In particular, the mixed partial p_{xy} is clearly given by

$$p_{xy}(x, y) = \sum_{i=0}^{n} \ell_i'(x) \sum_{j=0}^{m} \tilde{\ell}_j'(y)f(x_i, y_j).$$

Therefore,

$$p_{xy}(\hat{x}, \hat{y}) = \sum_{i=0}^{n} \ell_i'(\hat{x}) p_i'(\hat{y})$$

where

$$p_i'(\hat{y}) = \sum_{j=0}^{m} \tilde{\ell}_j'(\hat{y}) f(x_i, y_j).$$

By way of interpretation, we pass $p_i(y)$ through the data $f(x_i, y_j)$, $0 \leq j \leq m$ (along the grid line $x = x_i$), differentiate $p_i(y)$, and evaluate $p_i'(\hat{y})$. Given the values $p_i'(\hat{y})$, $0 \leq i \leq n$, we pass $q(x)$ through these values at the knots $x_0, x_1, \ldots, x_n$, differentiate $q(x)$, and evaluate $q'(\hat{x})$. The net result is $q'(\hat{x}) = p_{xy}(\hat{x}, \hat{y}) \approx f_{xy}(\hat{x}, \hat{y})$.

An exactly analogous development can be given for spline interpolation. In particular, a function $S(x, y)$ is called a *bicubic spline* if $S(x, y)$ is a cubic spline in x for fixed y and a cubic spline in y for fixed x. The natural bicubic spline is easiest to treat, and so we restrict our attention to it. As with two-variable polynomial interpolation, we will see that a bicubic spline interpolator can be built up from one-dimensional cubic spline interpolators. The wire-frame representation in Fig. 5.7 is also valid for bicubic splines. We can run cubic spline interpolators along the grid lines $x = x_i$, through the data $f(x_i, y_j)$ for $0 \leq j \leq m$, and then evaluate them at $y = \hat{y}$. We then run a cubic spline interpolator along $y = \hat{y}$, through the values obtained above, and evaluate this cubic spline at $x = \hat{x}$.

To see that the construction described above is valid, we can use the idea of a *cardinal* natural spline. Given the knots $y_0, y_1, \ldots, y_m$, we say that $S_j(y)$ is a cardinal natural spline if $S_j(y)$ is a natural cubic spline on knots $y_0, y_1, \ldots, y_m$ and if $S_j(y_k) = \delta_{jk}$, $0 \leq k \leq m$. Clearly, these cardinal natural splines always exist; and if $g(y)$ is a function defined on $[y_0, y_m]$, then the natural cubic spline interpolator for $g(y)$ can be represented as

$$S(y) = \sum_{j=0}^{m} S_j(y) g(y_j).$$

[Note that the sum of natural cubic splines is a natural cubic spline; so $S(y)$ is obviously the cubic spline interpolator for $g(y)$; the form of the representation is similar to the Lagrange form of the interpolating polynomial.]

Given the concept of a cardinal natural spline, it is easy to see how to construct a natural bicubic spline $S(x, y)$. In particular, if $S_i(x)$, $0 \le i \le n$, denote the cardinal natural splines for $x_0, x_1, \ldots, x_n$, and $\tilde{S}_j(y)$ denote the cardinal natural splines for $y_0, y_1, \ldots, y_m$, then

$$S(x, y) = \sum_{i=0}^{n} \sum_{j=0}^{m} S_i(x)\tilde{S}_j(y)f(x_i, y_j)$$

is a bicubic spline that interpolates $f(x, y)$ on the grid π_{nm}. As before, we can write $S(x, y)$ as

$$S(x, y) = \sum_{i=0}^{n} S_i(x)\phi_i(y)$$

where

$$\phi_i(y) = \sum_{j=0}^{m} \tilde{S}_j(y)f(x_i, y_j).$$

From the representation of $S(x, y)$ above, we see that we can evaluate $S(x, y)$ and various partial derivatives of $S(x, y)$ by repeatedly calling a one-dimensional spline routine, $(n + 1)$ times in the y-direction and then once in the x-direction. [Note, we do not need to calculate $S_i(x)$ or $\tilde{S}_j(y)$.]

Bicubic spline approximation to functions $f(x, y)$ are quite effective. A number of applications should be evident; we can use these approximations to estimate mixed partials, estimate normals to a surface, calculate approximations to surface area, etc. Finally, cardinal splines can be used to obtain higher-dimensional analogs of cubic splines. Given $f(x, y, z)$ to approximate over a solid rectangular region that is gridded by (x_i, y_j, z_k), $0 \le i \le n$, $0 \le j \le m$, $0 \le k \le \ell$, we can form

$$S(x, y, z) = \sum_{i=0}^{n} \sum_{j=0}^{m} \sum_{k=0}^{\ell} S_i(x)\tilde{S}_j(y)S_k^*(z)f(x_i, y_j, z_k).$$

As before,

$$S(x, y, z) = \sum_{k=0}^{\ell} S_k^*(z)B_k(x, y)$$

where $B_k(x, y)$ is the bicubic spline interpolator to $f(x, y, z)$ at the level $z = z_k$. Again, the three-dimensional spline approximation to $f(x, y, z)$ can be built up by calling a one-dimensional spline routine repeatedly. To evaluate $f(\hat{x}, \hat{y}, \hat{z})$, we call the routine $(n + 2)$ times to get $B_k(\hat{x}, \hat{y})$, a total of $(\ell + 1)$ $(n + 2)$ calls. A final call in the z-direction interpolating $B_k(\hat{x}, \hat{y})$, $0 \le k \le \ell$ will produce $S(\hat{x}, \hat{y}, \hat{z})$.