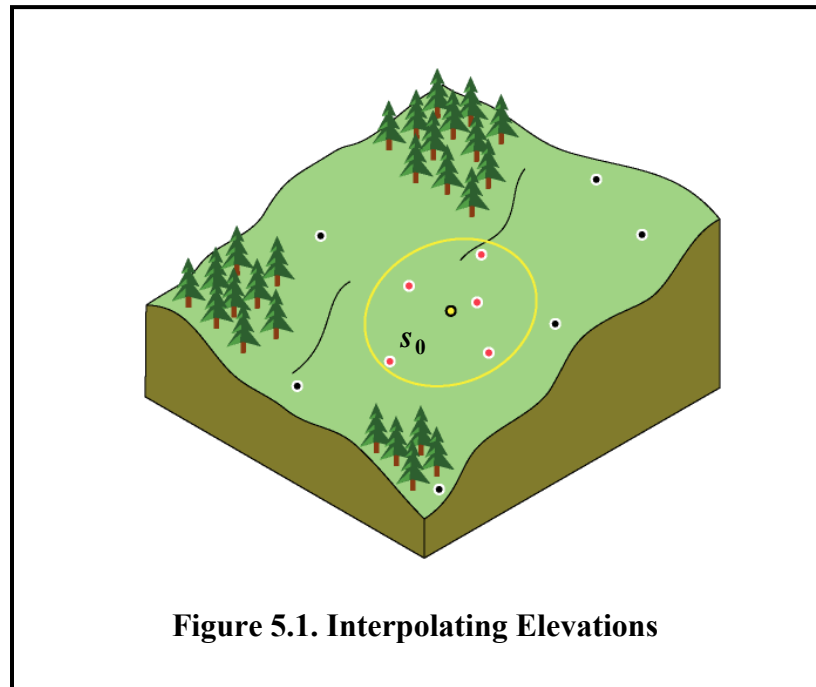


5. Spatial Interpolation Models

Given the above model of stationary random spatial effects $\{\varepsilon(s) : s \in R\}$, our ultimate objective is to apply these concepts to spatial models involving global trends, $\mu(s)$, i.e., to spatial stochastic models of the form, $Y(s) = \mu(s) + \varepsilon(s)$, $s \in R$. In continuous spatial data analysis, the most fully developed models of this type focus on *spatial prediction*, where values of spatial variables observed at certain locations are used to predict values at other locations. But it is important to emphasize here that many such models are in fact completely deterministic in nature [i.e., implicitly assume that $\varepsilon(s) \equiv 0$]. Such models are typically referred to as *spatial interpolation* (or *smoothing*) models [so we reserve the term *spatial prediction* for stochastic models of this type, as discussed later]. Indeed the *Inverse Distance Weighting* (IDW) model used for the Sudan Rainfall example in Section 2.1 above is an interpolation model. Moreover, a variety of other such models are in common use, and indeed, are also available in ARCMAP. So before developing the spatial prediction models that are of central interest for our purposes, it is appropriate to begin with selected examples of these interpolation models. In Section 6 below, we shall then consider the simplest types of spatial prediction models in which the global trend is constant, i.e., with $\mu(s) \equiv \mu$ for all $s \in R$. This will be followed in Section 7 with a development of more general prediction models in which the global trend, $\mu(s)$, is allowed to vary over space, and takes on a more important role.

5.1 A Simple Example of Spatial Interpolation

The basic idea of spatial interpolation is well illustrated by the “elevation” example shown in Figure 5.1 below (taken from the ESRI Desktop Help documentation)



Here it is assumed that elevations, $y(s)$, have been measured at a set of spatial locations $\{s_i : i = 1, \dots, n\}$ in some relevant region, R , as shown by the dots outlined in white. Given these measurements, one would like to estimate the elevation, $y(s_0)$, at some new location $s_0 \in R$, shown in the figure (outlined in black). Given the typical continuity properties of elevation, it is clear that those measurement locations closest to s_0 are the most relevant ones for estimating $y(s_0)$, as illustrated by the red dots lying in the neighborhood of s_0 denoted by the yellow circle. While it is not obvious how large this neighborhood should be, let us suppose for the moment that it has somehow been determined (we return to this question in Section ?? below). Then the question is how to use this set of five elevations at locations, say s_1, \dots, s_5 , to estimate $y(s_0)$. These locations are displayed in more detail in Figure 5.2 below, where $d_{0i} = \|s_0 - s_i\|$ denotes the distance from s_0 to each point s_i , $i = 1, \dots, 5$.

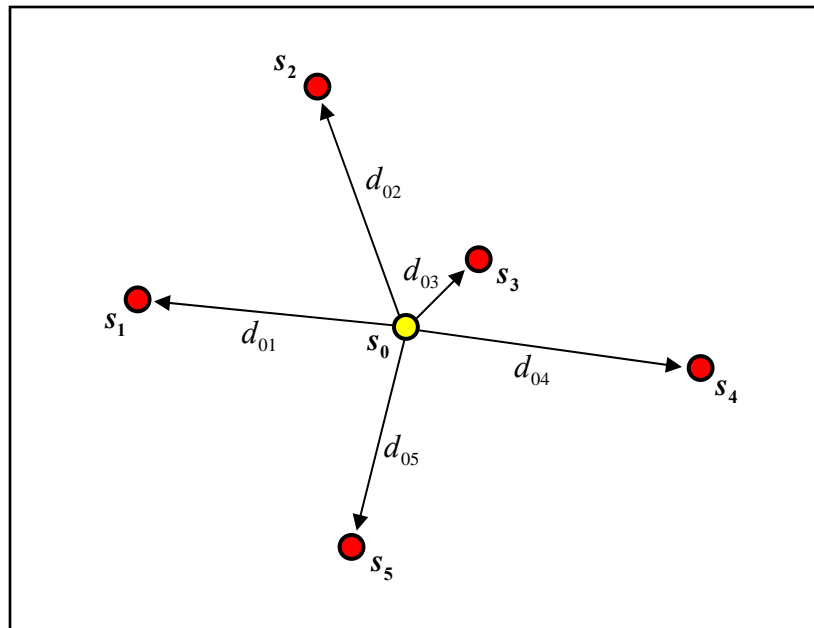


Figure 5.2. Neighborhood of Point s_0

5.2 Kernel Smoothing Models

Intuitively, those points closer to s_0 should have more influence in this estimate. For example, it is seen in the figure that point s_3 is considerably closer to s_0 than is point s_4 . So it is reasonable to assume that $y(s_3)$ is more influential in the estimation of $y(s_0)$ than is $y(s_4)$. Hence if we now designate the set of points used for estimation at s_0 as the *interpolation set*, $S(s_0)$, [so that in the example above, $S(s_0) = \{s_1, \dots, s_5\}$] then it is natural to consider estimates, $\hat{y}(s_0)$, of the form,

$$(5.2.1) \quad \hat{y}(s_0) = \frac{\sum_{s_i \in S(s_0)} w(d_{0i}) y(s_i)}{\sum_{s_j \in S(s_0)} w(d_{0j})} = \sum_{s_i \in S(s_0)} \left(\frac{w(d_{0i})}{\sum_{s_j \in S(s_0)} w(d_{0j})} \right) y(s_i)$$

where the *weight function*, $w(d)$, is a *positive decreasing* function of distance, d . Interpolation models of this type are often referred to as *kernel smoothers*. The reason for the ratio form is that the *effective weights* on each $y(s)$ value, [as defined by the bracketed expression in (5.1.1)] must then sum to one, i.e.,

$$(5.2.2) \quad \sum_{s_i \in S(s_0)} \left(\frac{w(d_{0i})}{\sum_{s_j \in S(s_0)} w(d_{0j})} \right) = \frac{\sum_{s_i \in S(s_0)} w(d_{0i})}{\sum_{s_j \in S(s_0)} w(d_{0j})} = 1$$

and are thus interpretable as the “fractional contribution” of each $y(s)$ to the estimate, $\hat{y}(s_0)$. Thus points closer to s_0 in $S(s_0)$ will have higher fractional contributions to $\hat{y}(s_0)$ since for all $s_i, s_j \in S(s_0)$

$$(5.2.3) \quad d_{0i} < d_{0j} \Rightarrow w(d_{0i}) > w(d_{0j}) \\ \Rightarrow \frac{w(d_{0i})}{\sum_{s_k \in S(s_0)} w(d_{0k})} > \frac{w(d_{0j})}{\sum_{s_k \in S(s_0)} w(d_{0k})}$$

We have already seen an example of a kernel smoother, namely the *inverse distance weighting* (IDW) smoother in Section 2.1 above. In this case, the weight function is a simple *inverse power function* of the form,

$$(5.2.4) \quad w(d) = d^{-\alpha}$$

where α is a positive constant (typically $\alpha = 1$ or $\alpha = 2$). While this is the only kernel smoother available in ARCMAP, it worthwhile mentioning one other, namely the *exponential smoother*, in which the weights are given by a *negative exponential function* of the form

$$(5.2.5) \quad w(d) = e^{-\theta d}$$

for some positive constant, $\theta > 0$. To compare these two smoothers, it is instructive to plot typical values of these weight functions. In Figure 5.3 below, an inverse power function with $\alpha = 2$ (shown in blue) is compared to a negative exponential function with $\theta = 1$ (shown in red). As seen in the figure, the most important difference between these functions is near the origin where $d \rightarrow 0$ implies that $e^{-\theta d} \rightarrow e^0 = 1$, but where $d^{-\alpha} \rightarrow \infty$. So for inverse power smoothers (like IDW), one necessarily obtains very “peaked” interpolation surfaces near data points where the effective weight of that data point approaches one, and totally dominates all other data points. (This is precisely the reason for the “rainfall peaks” around data points seen for Sudan in Figure 2.2.)

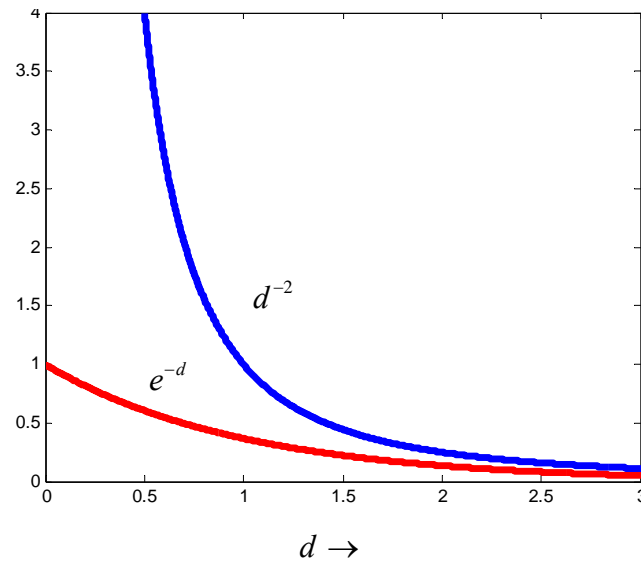


Figure 5.3. Comparison of Exponential and IDW Smoothers

In this respect, exponential smoothers may be preferred. However, if one requires *exact interpolation* at data points [i.e., $\hat{y}(s_i) \equiv y(s_i)$] then this is only possible if $w(d) \rightarrow \infty$ as $d \rightarrow 0$. So kernel smoothers like the exponential yield results that are actually *smoother than the data itself*. In summary, it is important to be aware of such differences between possible kernel smoothers, and to employ one that is most suitable for the purpose at hand.

5.3 Local Polynomial Models

A second type of spatial interpolator available in the *Geostatistical Analyst* (GA) extension of ARCMAP is a *local polynomial interpolator*. Here it is assumed explicitly that the value, $y(s_0)$, lies on the same (smooth) surface as the observed values, $\{y(s_i) : i = 1, \dots, n\}$, and that the local curvature of this surface is well approximated by polynomials of a given order. The simplest polynomial (of order one) is a *linear* function. So here the value, $y(s_0)$, is estimated by finding the linear function which best fits the y -values on the coordinate values of the points in the interpolation set, $S(s_0) = \{s_1, \dots, s_n\}$. More specifically, if the y -values of these points are denoted by $(y_i : i = 1, \dots, n)$ and their coordinate values by $[(s_{i1}, s_{i2}) : i = 1, \dots, n]$, then estimation is done by the same least squares procedure as in linear regression, i.e., by finding the beta estimates $(\hat{\beta}_0, \hat{\beta}_1, \hat{\beta}_2)$, that minimize the sum-of-squared deviations:¹

¹ It should be emphasized that while this estimation procedure is the same as in regression, there is no appeal to a linear statistical model, and in particular, no random error model.

$$(5.3.1) \quad \sum_{i=1}^n [y_i - (\beta_0 + \beta_1 s_{i1} + \beta_2 s_{i2})]^2$$

Using these estimates, the interpolated value of $y(s_0)$ at point $s_0 = (s_{01}, s_{02})$ is given by

$$(5.3.2) \quad \hat{y}(s_0) = \hat{\beta}_0 + \hat{\beta}_1 s_{01} + \hat{\beta}_2 s_{02}$$

A *one-dimensional* illustration of local linear interpolation is shown in Figure 5.4 below, where in this case it is assumed that the interpolation set, $S(s_0) = \{s_1, s_2, s_3, s_4\} \subset \mathbb{R}$, is given by the four points shown in red (where s_i is the coordinate of point i on the line, \mathbb{R}). The dashed red line is a plot of the linear interpolation function obtained from these four data points, so that the interpolated value, $\hat{y}(s_0) = \hat{\beta}_0 + \hat{\beta}_1 s_0$, is shown by the white dot in the figure.

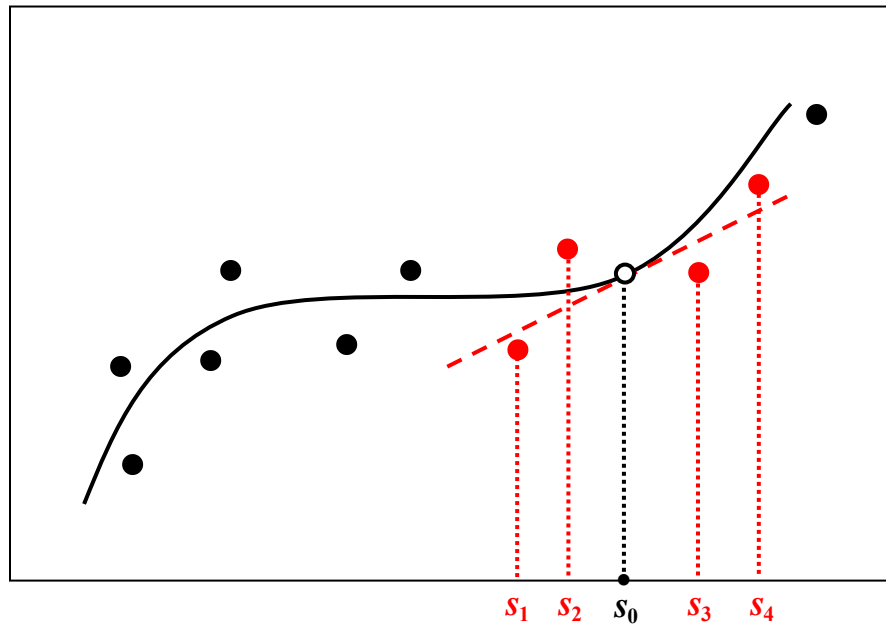


Figure 5.4. Local Linear Interpolation

More generally, this defines a single point on the *interpolation function* defined for all points, as shown schematically by the solid curve in the figure.² In practice this function would not be so smooth, and in fact would not even be continuous. In particular there would be jumps in this function at each location, s_0 , where a data point, s_i , either enters or leaves the current interpolation set, $S(s_0)$. Such discontinuities can be removed by fixing the diameter (*bandwidth*) of interpolation sets, say

$$(5.3.3) \quad S(s_0) = \{s_i : \|s_0 - s_i\| \leq d_0\}$$

² A particularly good discussion of this *local linear polynomial* case is given in ESRI Desktop Help at http://webhelp.esri.com/arcgisdesktop/9.2/index.cfm?TopicName=How_Local_Polynomial_interpolation_works.

and introducing *kernel smoothing weight function*, w , similar to (5.2.1) which falls to zero at distance, d_0 . One then modifies the local least squares in (5.3.1) to a *local weighted least squares* of the form,³

$$(5.3.4) \quad \sum_{i=1}^n w_{0i} [y_i - (\beta_0 + \beta_1 s_{i1} + \beta_2 s_{i2})]^2$$

where $w_{0i} = w(\|s_0 - s_i\|)$. Hence points in $S(s_0)$ at greater distances from s_0 will have less weight in the interpolation (and will have *no weight* at distance, d_0). This implies in particular that as s_0 moves along the axis in Figure 5.4, data points entering or leaving the interpolation set will initially have zero weight, thus preserving *continuity* of the interpolation function. An actual example of such a *locally weighted linear interpolation function* is shown (in red) on the left in Figure 5.5 below.

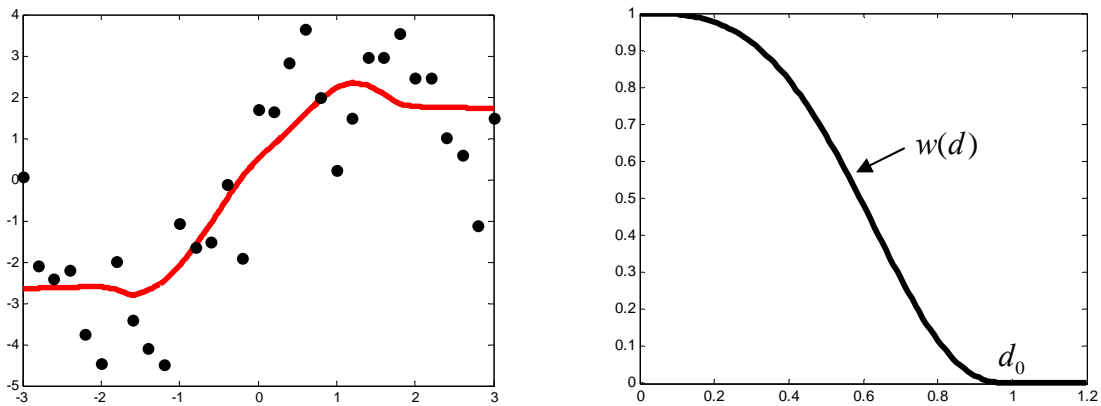


Figure 5.5 One-Dimensional Geographic Weighted Regression

Here the kernel smoothing function used is the popular “tri-cube” function which has the mathematical form:

$$(5.3.5) \quad w(d) = \begin{cases} \left(1 - (d/d_0)^3\right)^3 & , d \leq d_0 \\ 0 & , d > d_0 \end{cases}$$

where in this case a bandwidth of $d_0 = 1$ was used. The shape of this function is shown on the right in Figure 5.5, where the distance scale has been increased for visual clarity.

This type of linear interpolation is formally related to *Geographic Weighted Regression* (GWR), which is also available in the ArcToolBox of ARCMAP [**Spatial Statistics Tools** → **Modeling Spatial Relationships** → **GWR**]. Since GWR allows the use of explanatory variables other than coordinate locations, and also includes stochastic random effects, it is much more than a simple interpolation tool. But the interpolation

³ This is essentially a linear version of the *nonlinear weighted least squares* in expression (4.7.9) of the Variograms section.

example above nonetheless serves to illustrate the basic mechanism of *locally weighted least squares* used in GWR.⁴

Finally it should be mentioned that local polynomial interpolation can of course involve high-order polynomials. As one illustration, suppose we consider local polynomial interpolation with second-order (quadratic) polynomials. Then the sum-of-squared deviations in (5.3.1) would now be replaced by the *quadratic* version,

$$(5.3.6) \quad \sum_{i=1}^n [y_i - (\beta_0 + \beta_1 s_{i1} + \beta_2 s_{i2} + \beta_3 s_{i1}^2 + \beta_4 s_{i2}^2 + \beta_5 s_{i1} s_{i2})]^2$$

and would in turn yield *local quadratic interpolations* of the form:

$$(5.3.7) \quad \hat{y}(s_0) = \hat{\beta}_0 + \hat{\beta}_1 s_{01} + \hat{\beta}_2 s_{02} + \hat{\beta}_3 s_{01}^2 + \hat{\beta}_4 s_{02}^2 + \hat{\beta}_5 s_{01} s_{02}$$

A schematic of such an interpolation paralleling Figure 5.4 is shown in Figure 5.6 below.

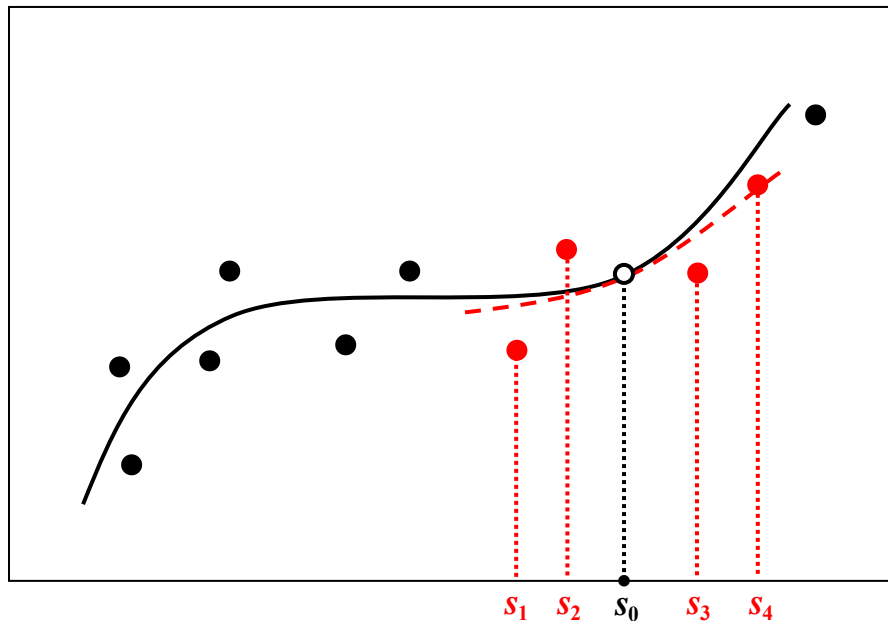


Figure 5.6. Local Quadratic Interpolation

In this schematic example, the red dashed curve is now the quadratic function in (5.3.7) evaluated at all points on the line, include s_0 . Given the obvious nonlinear nature of this data, it would appear that a quadratic polynomial interpolation would yield a better fit to this data.

However, it should again be emphasized that the actual locus of interpolations obtained would still be *discontinuous* for exactly the same reasons as in the linear interpolation

⁴ Here it should be noted that the type of one-dimensional example shown in Figure 5.5 is not readily implemented using GWR in ARCMAP. Rather this example was computed using the MATLAB program, `gwr.m`, in the suite of programs by James LeSage, available at <http://www.spatial-econometrics.com/>.

case above. So when using these interpolators in GA, be aware that implicit smoothing procedures are being used, in a manner similar to the kernel smoothing procedure outlined above. Hence the numerical values obtained will differ slightly from the simple interpolations, $\hat{y}(s_0)$, in (5.3.2) and (5.3.7) above, depending the spacing of actual data points. Also be aware that these smoothing procedures used are *not* documented in ARCMAP help. As with essentially all commercial GIS software, there are often hidden layers of calculations being done that are not fully documented.

5.4 Radial Basis Function Models

In view of the continuity problem inherent in local polynomial interpolations, it is of interest to consider interpolation methods that are guaranteed to yield interpolation surfaces that are not only continuous but are in fact everywhere *smooth* (in contrast to IDW which is continuous but not smooth at data points). The simplest of these are the so called *radial basis function interpolators* also available in GA. Here the basic idea is to choose a family of radially symmetric functions, $f(s)$, about the origin that (typically) fall to zero as distance from the origin increases. We have already seen one such function, namely the *standard bivariate normal density functions* in Figure 3.2 of the Spatially-Dependent Random Effects section above. Each data point, (s_i, y_i) , is then associated with a member of this family where the origin is set at s_i . So for the bivariate normal case one would have⁵

$$(5.4.1) \quad f_i(s) = e^{-\frac{1}{2}\|s-s_i\|^2}, \quad s \in \mathbb{R}^2$$

where the normalizing factor $(2\pi)^{-1/2}$ plays no role here, and has been removed. One then defines the *interpolation function* for this model to be a weighted combination of these basis functions:

$$(5.4.2) \quad \hat{y}(s) = \sum_{i=1}^n a_i f_i(s)$$

To choose an appropriate set of weights, one typically requires *exact interpolation* at each data point (s_i, y_i) , i.e.,

$$(5.4.3) \quad y_i = \hat{y}(s_i) = \sum_{j=1}^n a_j f_j(s_i), \quad i = 1, \dots, n$$

Since this is simply a system of linear equations in the unknown weight vector, $a = (a_1, \dots, a_n)'$, one can easily solve for these weights. In particular, if we let $y = (y_1, \dots, y_n)'$ denote the vector of observe y -values, and let the n -square *function matrix*, F , be defined by

$$(5.4.4) \quad F_{ij} = f_i(s_j), \quad i, j = 1, \dots, n$$

⁵ Recall that *Euclidean distance* between vectors x and y is denoted by $d(x, y) = \|x - y\|$.

then it follows at once from (5.4.3) that

$$(5.4.5) \quad \begin{pmatrix} y_1 \\ \vdots \\ y_n \end{pmatrix} = \begin{bmatrix} F_{11} & \cdots & F_{1n} \\ \vdots & \ddots & \vdots \\ F_{n1} & \cdots & F_{nn} \end{bmatrix} \begin{pmatrix} a_1 \\ \vdots \\ a_n \end{pmatrix} \Rightarrow y = F a$$

Hence the desired weight vector is uniquely defined by⁶

$$(5.4.6) \quad a = F^{-1}y$$

This set of weights necessarily yields a *smooth interpolation function* that passes through all of the data points.

Although the normal density above is a very common choice for radial basis functions, this option is not available in GA. However, one option that is available which looks very similar to the standard bivariate normal density is the so called *inverse multiquadratic* function defined for all $s \in \mathbb{R}^2$ by⁷

$$(5.4.7) \quad f(s) = \frac{1}{1 + \|s\|^2}$$

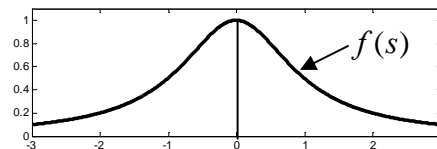
so that in this case, (5.1.13) is replaced by

$$(5.4.8) \quad f_i(s) = \frac{1}{1 + \|s - s_i\|^2}$$

While this function appears to be mathematically very different from the bivariate normal density, a two-dimensional plot of (5.4.7) shows that it is virtually indistinguishable from Figure 3.2 in a qualitative sense. About the only significant difference is that it falls zero much more *slowly* than the normal density.

To gain some feeling for this type of interpolation, it is again convenient to develop a *one-dimensional* example paralleling the example for local polynomial interpolations above. In this case, (5.4.7) reduces to the function

$$(5.4.9) \quad f(s) = \frac{1}{1 + s^2}, \quad s \in \mathbb{R}$$



which is now seen to be qualitatively similar to the univariate normal density in expression (3.1.11) of Section 3. Interpolation with these radial basis functions can be

⁶ This of course assumes that F is *nonsingular*, which will hold in all but the most degenerate cases.

⁷ As with the standard normal density, this function can be generalized by adding a weight, θ , to s

yielding the one-parameter family, $f(s | \theta) = (\theta^2 + \|s\|^2)^{-1}$.

illustrated by the example shown in Figure 5.7 below, which involves only three data points, (s_i, y_i) , $i = 1, 2, 3$.

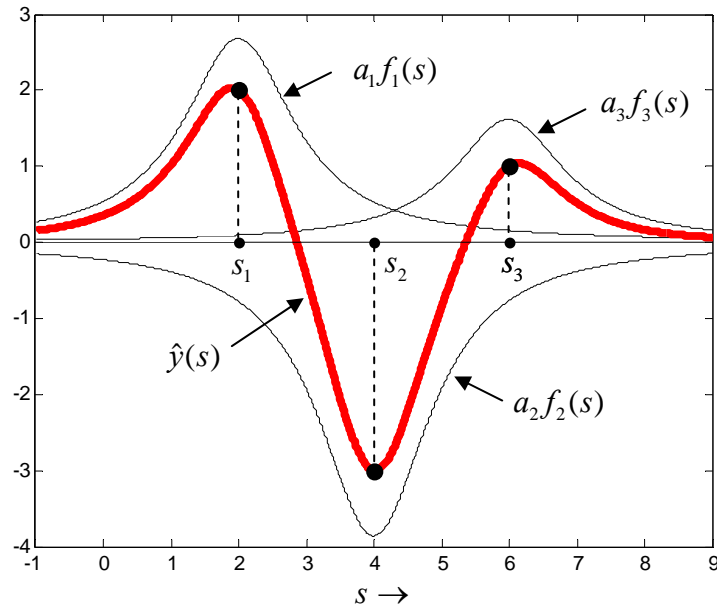


Figure 5.7. Interpolation with Radial Basis Functions

Here the fitted radial basis functions $[a_i f_i(s), i = 1, 2, 3]$ are shown in black, and the resulting interpolation function, $\hat{y}(s)$, is shown in red.

Notice that unlike the kernel smoothers above, there is no need for interpolation sets in this case. Here the entire interpolation function, $\hat{y}(s)$, is determined simultaneously at all point locations, s . Notice also that this function is necessarily smooth (since it is a sum of smooth functions). Finally, note from the figure that $\hat{y}(s)$ is indeed an *exact interpolator* at data points, i.e., it passes through each of the data points shown.

5.5 Spline Models

While the above procedure offers a remarkably simple way to obtain smooth and exact interpolations, it can be argued that the choice of basis functions is rather arbitrary. Moreover, it is difficult to regard this fitting procedure as “optimal” in any sense. Rather its weights are determined entirely by the *exact-interpolation condition* in (5.4.3) above. However, there is an alternative method of interpolation, known as *spline interpolation*, which is more appealing from a theoretical viewpoint. As with radial basis functions, this approach seeks to find a prediction function, $\hat{y}(s)$, that satisfies the *exact-interpolation condition*,

$$(5.5.1) \quad \hat{y}(s_i) = y_i, \quad i = 1, \dots, n$$

There are of course infinitely many smooth functions that could satisfy this condition. Hence the unique feature of spline interpolation is that rather than simply pre-selecting a given set of smooth candidate functions, this approach seeks to find the *smoothest possible function* satisfying (5.5.1). To characterize “smoothness”, recall that for one-dimensional functions, $f(s)$, the *second derivative*, $f''(s)$, measures the curvature of the function. In particular, linear functions, $f(s) = a + bs$, have *zero curvature* as reflected by the fact that $f''(s) \equiv 0$. More generally, if we ignore signs and define the curvature of f at s by $f''(s)^2$, then to compare the curvature of functions f on a given interval, say $[a, b]$, it is natural to consider their *total curvature*

$$(5.5.2) \quad C(f) = \int_a^b f''(s)^2 ds$$

as a measure of “smoothness”, where higher degrees of smoothness correspond to lower total curvature.⁸

For *two dimensions*, the idea is basically the same. Here “curvature” at point, $s = (s_1, s_2)$, is defined in terms of the *Hessian matrix* of second partial derivatives

$$(5.5.3) \quad H_f(s) = \begin{pmatrix} \frac{\partial^2 f}{\partial s_1^2} & \frac{\partial^2 f}{\partial s_1 \partial s_2} \\ \frac{\partial^2 f}{\partial s_1 \partial s_2} & \frac{\partial^2 f}{\partial s_2^2} \end{pmatrix}$$

Again to ignore signs, one can define the *size* of a matrix, $M = (m_{ij})$, by its squared distance from the origin (as a vector), i.e.,

$$(5.5.4) \quad \|M\|^2 = \sum_i \sum_j m_{ij}^2$$

In these terms, the *curvature* of a two-dimensional function, $f(s)$, at $s = (s_1, s_2)$ is defined to be the size of its Hessian at s , i.e.,

$$(5.5.5) \quad \|H_f(s)\|^2 = \left(\frac{\partial^2 f}{\partial s_1^2}\right)^2 + 2\left(\frac{\partial^2 f}{\partial s_1 \partial s_2}\right)^2 + \left(\frac{\partial^2 f}{\partial s_2^2}\right)^2$$

[which is seen to be the natural generalization of the one-dimensional case, $\|f''(s)\|^2 = f''(s)^2$]. Hence to compare the curvature of functions, f , on a (bounded) two-dimensional region, $R \subset \mathbb{R}^2$, the natural extension of (5.5.2) is to define *total curvature*, $C(f)$, by

⁸ While it might be conceptually more appropriate to use *average curvature*, $\bar{C}(f) = \frac{1}{b-a} C(f)$, this simple rescaling has no effect on the ordering of smoothness among functions, f .

$$(5.5.6) \quad C(f) = \int_R \|H_f(s)\|^2 ds = \int_R \left[\left(\frac{\partial^2 f}{\partial s_1^2} \right)^2 + 2 \left(\frac{\partial^2 f}{\partial s_1 \partial s_2} \right)^2 + \left(\frac{\partial^2 f}{\partial s_2^2} \right)^2 \right] ds_1 ds_2$$

Thus, for a given set of data points, $[(s_1, y_1), \dots, (s_n, y_n)]$ with $\{s_1, \dots, s_n\} \subset R \subset \mathbb{R}^2$, the corresponding *spline interpolation problem* is to find a function, $\hat{y}(s)$, on R which *minimizes total curvature* (5.5.6) subject to the *exact-interpolation condition* (5.5.1).

While these interpolation problems are relatively simple to state, they can only be solved by very sophisticated mathematical methods. Hence for our purposes, it suffices to say that these solutions are themselves remarkably simple, and lead to optimally smooth interpolation functions, $\hat{y}(s)$. To illustrate the basic ideas, it is again convenient to focus on the one dimensional case. Here it turns out that the basic interpolation functions are combinations of *cubic functions*,

$$(5.5.7) \quad f(s) = a_3 s^3 + a_2 s^2 + a_1 s + a_0$$

between every pair of adjacent data points. To gain some intuition here, suppose one considers a “partial” smooth curve with a gap between two data points, s_1 and s_2 , as shown in Figure 5.8a below. To “complete” this curve in a smooth way, one must match both the *end values* (shown as black dots) and the *end slopes* (shown as dashed lines).

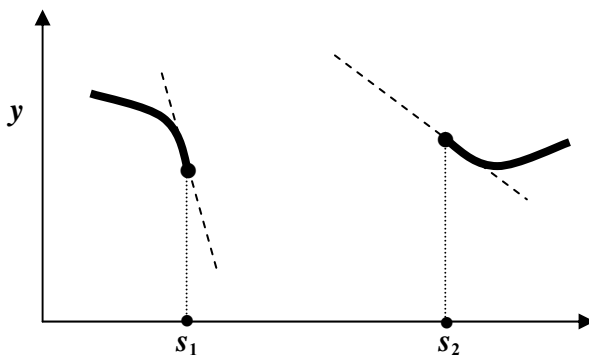


Figure 5.8a. Partial Smooth Curve

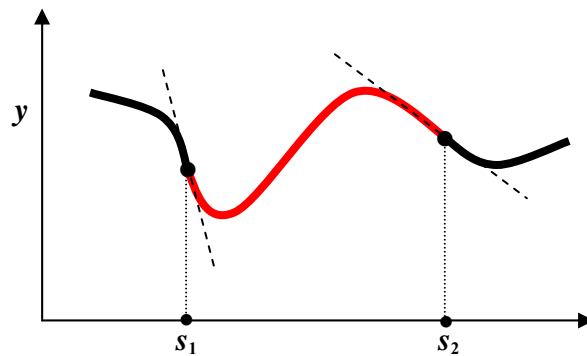


Figure 5.8b. Completed Smooth Curve

Since the cubic function is the smoothest function with an *inflection point* (where the second derivative changes sign), it should be clear from Figure 5.8b that this adds just enough flexibility to complete this curve in the smoothest possible way.⁹

As one example of this interpolation method, the data set in Figure 5.7 has been re-interpolated using cubic spline interpolation in Figure 5.9 below.¹⁰ Here there appears to be a dramatic difference between the two methods. But except for the slight scale

⁹ For a more general discussion of fitting cubic splines to (one-dimensional) data sets, see McKinley and Levine at <http://online.redwoods.cc.ca.us/instruct/darnold/laproj/fall98/skymeg/proj.pdf>.

¹⁰ This interpolation was computed in MATLAB using their package program, **spline.m**.

differences between these two figures, the qualitative “bowl” shape of $\hat{y}(s)$ within the interval defined by these three data points is roughly similar. Moreover, it should now be clear that this “bowl” is much smoother for the cubic spline case than for the (inverse quadratic) radial basis function case in Figure 5.7. Indeed, this cubic spline is the *smoothest possible* exact interpolator within this interval. But notice also that *outside* this interval, the two interpolations differ radically. Since the individual radial basis functions in Figure 5.7 all approach zero as distance from data points increase, the interpolation function also decreases to zero. But for the cubic spline case, this smooth “bowl” is only achieved by continuing the bowl shape *outside* the data interval.

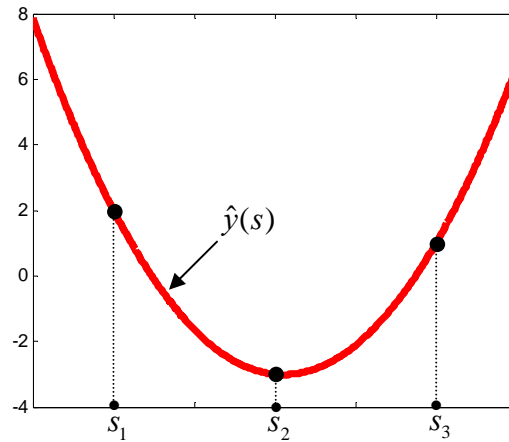


Figure 5.9 Cubic Spline Interpolation

More generally, these cubic spline interpolations tend to diverge at the data boundaries, and are much less trustworthy in this range. A better example of this is shown in Figure 5.10 below where the interpolation now involves five points. Here again the cubic spline interpolator, $\hat{y}(s)$, is seen to yield the smoothest possible exact interpolation of these five data points. But outside this range, $\hat{y}(s)$ now diverges downward on both sides in order to achieve this degree of smoothness.

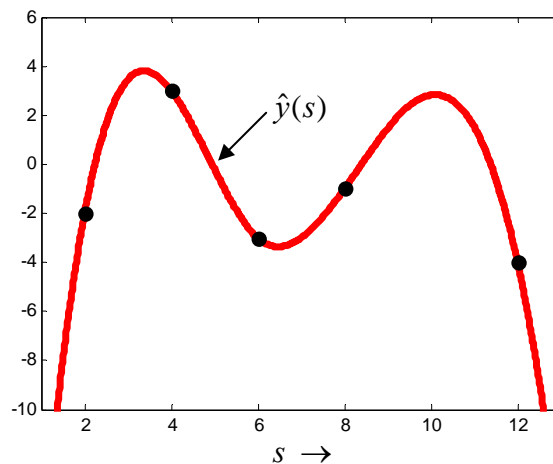


Figure 5.10 Cubic Spline Interpolation

While these one-dimensional examples serve to illustrate the main ideas of spline interpolation, the solution in *two dimensions* [i.e., the function $\hat{y}(s)$ minimizing (5.5.6) subject to (5.5.1)] is mathematically quite different than the one-dimensional case. From an intuitive viewpoint, the basic reason for this is that in two dimensions it is not possible to “piece together” solutions between data points. In fact, the solution in two dimensions is formally much closer to the radial basis function approach above. In particular, the optimal interpolation function, $\hat{y}(s)$, designated as a *thin-plate spline function*, takes essentially the same form as (5.4.2), namely

$$(5.5.8) \quad \hat{y}(s) = \hat{y}(s_1, s_2) = (\lambda_0 + \lambda_1 s_1 + \lambda_2 s_2) + \sum_{i=1}^n a_i f_i(s | \tau)$$

with “radial basis functions” (parameterized by, $\tau > 0$) given by

$$(5.5.9) \quad f_i(s | \tau) = f_\tau(d_{is}) = d_{is}^2 \log\left(\frac{d_{is}}{\tau}\right), \quad i = 1, \dots, n$$

where $d_{si} = d(s, s_i) = \|s - s_i\|$ for each data point, s_i . The linear part of $\hat{y}(s)$ is usually called the *trend function*, and is seen to have little influence on the detailed shape of $\hat{y}(s)$ relative to these radial basis functions.¹¹ The key feature of these functions is their flexible shape. In particular observe that for s closer to s_i than distance τ , i.e., $d_{is} < \tau$, the log expression is *negative*. Hence each function first “dips” and eventually rises as distance d_{is} increases. So, as with the one-dimensional cubic splines above, this again yields a *single inflection point* along rays in every direction from the origin, as shown by the one-dimensional profile of this function in Figure 5.11 below:

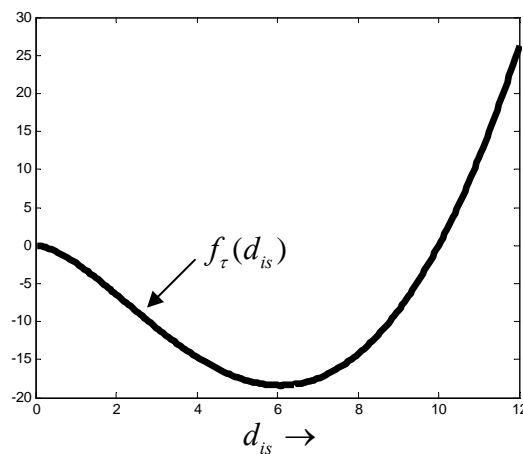


Figure 5.11 Radial Shape of Spline Basis Functions

¹¹ In fact, this linear part has *zero curvature* by definition, and hence has no effect on the curvature of $\hat{y}(s)$.

Here the value of $\tau = 10$ was used, so that by definition $f_{10}(d_{is})$ rises back up to zero at exactly $d_{is} = 10$. This also makes it clear that beyond radial distance τ from data points, s_i , these functions diverge rapidly, and can produce rapid changes in $\hat{y}(s)$. So larger values of τ tend to produce “stiffer” surfaces with less variation. This can also be seen in the full two-dimensional representation of $f_i(s|\tau)$ is shown in Figure 5.12 below [again with $\tau = 10$].

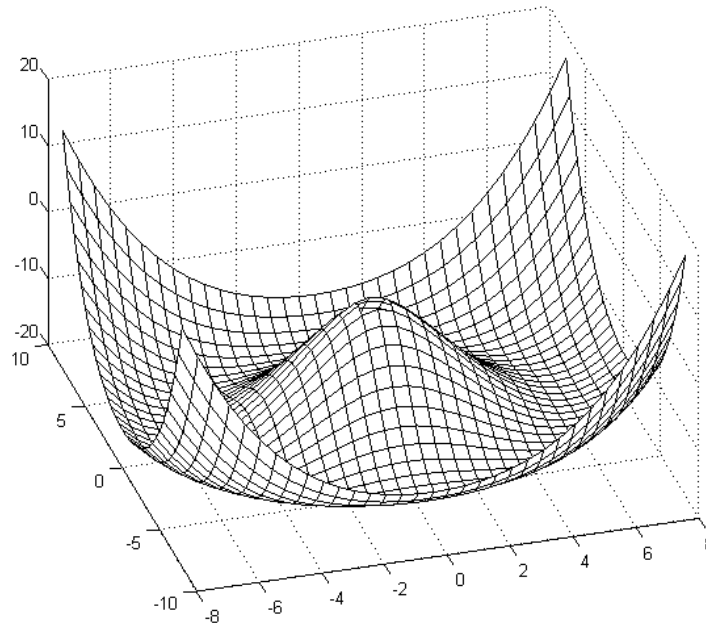


Figure 5.12. Two-Dimensional Spline Basis Functions

As in the one-dimensional case, the local flexibility of this “Mexican hat” function allows much more rapid changes in value than say the multi-quadratic basis function above. So for example in the case of elevations, these thin-plate splines will do a much better job of capturing rapid changes in elevation than will the multi-quadratic functions.

Finally, it should again be noted that (as mentioned for kernel smoothers at the end of Section 5.2 above) the two-dimensional spline interpolation methods employed in the *Spatial Analyst* (SA) extension of ARCMAP are considerably more complex than the basic thin-plate spline model developed above. To describe the main differences, we focus on the *regularized spline* option (rather than the less commonly used “tension” spline option). While this model is based essentially on thin-plate splines, the radial basis functions in (5.5.9) above are “augmented” by an additional term that reflects *third derivative* effects as well and second derivative (curvature) effects.¹² So the “ τ ”

¹² This use of third derivatives is appropriate if continuity of second derivatives is required for the prediction function [see Mitáš and Mitášová (1988, Section 6)]. But continuity of first derivatives is usually sufficient for adequate smoothness. So from a practical viewpoint, the simpler approach of thin-plate spline interpolation is in many ways more appealing. [See for example the constructive development of this method in Franke (1982)].

parameter in (5.5.9) plays a somewhat more complex role in these functions. However, its basic interpretation remains the same. In particular, larger values of τ still produce “stiffer” surfaces with less variation.

In addition to these more complex basis functions, the spline interpolation procedure in SA is “localized” by partitioning R into smaller cells to simplify the calibration procedure. So in addition to the τ parameter, the user is also asked to choose the “number of points”, say η , to be used in the interpolation of each cell. Again, larger numbers of points produce smoother interpolations with less variation.

5.6 A Comparison of Models using the Nickel Data

While the above models were motivated in terms of the “elevation” example in Figure 5.1, most spatial data sets tend to exhibit more local variation than elevations (or, say, the mean temperature levels studied in Assignments 3 and 5). A typical example is provided by the Nickel data displayed in Figure 4.18 of Section 4.9 above. Here we start with the *regularized spline tool* discussed above, and compare interpolations for two different parameter settings, (τ, η) :

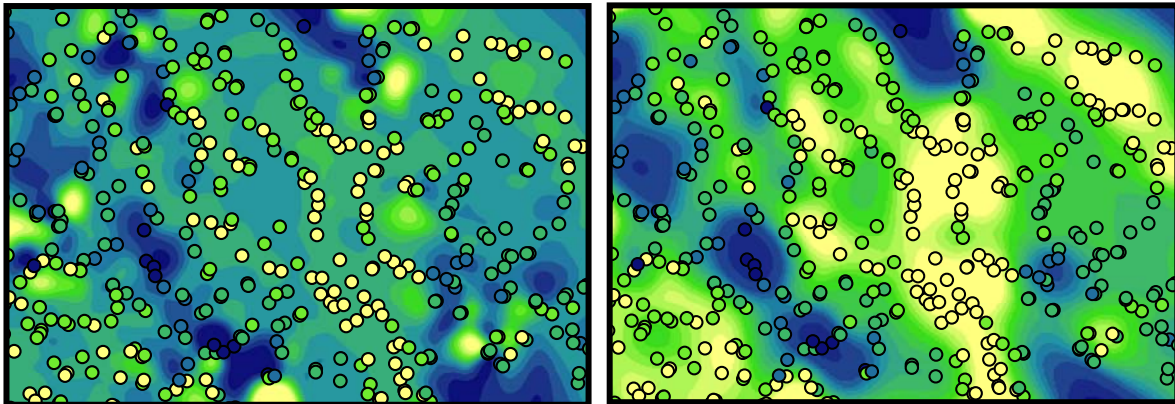


Figure 5.13a. Spline with $(\tau = 0.1, \eta = 12)$ Figure 5.13b. Spline with $(\tau = 10, \eta = 50)$

Here the spline interpolation shown in Figure 5.13a uses the default parameter settings for the spline tool, namely $\tau = 0.1$ and $\eta = 12$. However, a comparison with Figure 4.18 shows that this interpolation does a rather poor job of capturing overall variations in nickel deposits. Much like the IDW interpolation of rainfall in Figure 2.2 of Section 2 above, this interpolation shows far too many local extremes, both high and low. This can also be seen numerically by noting that while the actual values of nickel deposits are of course nonnegative (starting from 1.0 ppm) the values in this spline interpolation go down to values as low as -700 ppm, which are of course totally meaningless. As described above, the key problem here is that this spline function is attempting to interpolate between points that exhibit extreme local variation in values. Hence while the values, $\tau = 0.1$ and $\eta = 12$, are reasonable choices for very smooth surfaces, they are creating far too much variation between data locations in this case. In fact, to achieve an

interpolation that is sufficiently “stiff” to provide a reasonable overall approximation to this surface, it is necessary to use much larger settings, such as the values $\tau = 10$ and $\eta = 50$ shown in Figure 5.13b.

Turning now to a broader range of models, we present a graphical comparison of four different methods for interpolating this nickel data in Figure 5.14 below. The first three relate to the deterministic methods above.

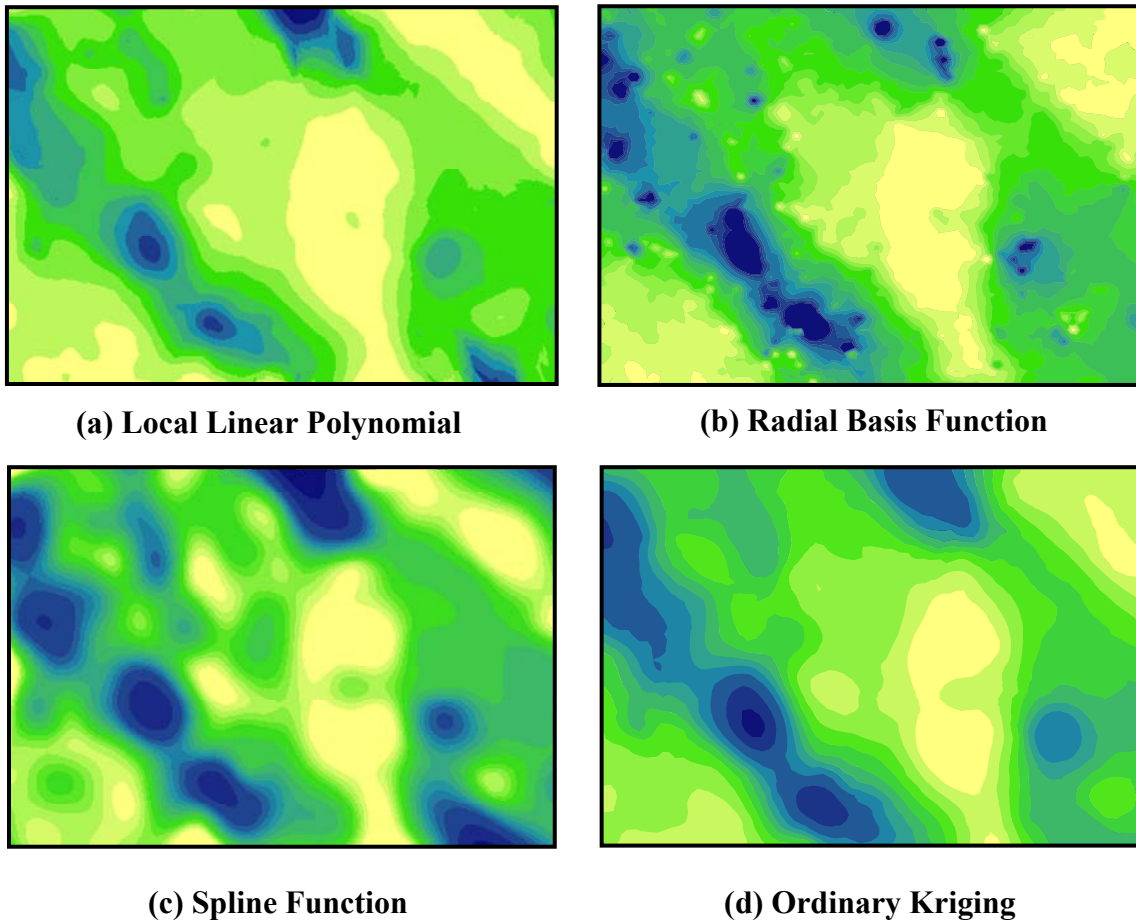


Figure 5.14 Interpolation Comparisons

Panel (a) shows the results of a *local linear polynomial* interpolation in Geostatistical Analyst (GA). Panel (b) shows a *radial basis function* interpolation in GA with the *inverse multiquadratic* option described above. Panel (c) shows the same regularized spline interpolation in Figure 5.13b above (minus the data points). Finally, the last panel compares these deterministic methods with the stochastic method of *ordinary kriging* in GA, which will be developed fully in Section 6.

For the present, the main purpose of this graphical comparison to show that in spite of their mathematical differences, the actual results of these different interpolation methods

are qualitatively quite similar. . However, it should be emphasized that considerable experimentation was required to find parameter settings for each method the yielded this degree of visual similarity. For example, as already mentioned above, the spline interpolation in panel (c) required the use of very large values of (τ, η) to achieve a sufficient degree of overall smoothness. As for panel (b), note that while the inverse multiquadratic function is itself very smooth, the variation in y -values here leads to the least smooth interpolation of the four panels shown. The key factor appears to be the lack of flexibility in fitting, which in this case is determined entirely by the exact-interpolation condition. Finally, as was pointed out at the end of Section 5.2, the local linear interpolation in panel (a) involves a number of internal smoothing procedures to remove the discontinuities created by shifting interpolation sets from point to point. So here it is not even clear how such smoothness was achieved. The same is in fact true of the ordinary kriging results shown in panel (d) . As we shall see below, this procedure involves “prediction sets” identical in spirit to the “interpolation sets” of local polynomial interpolations. So again, internal smoothing procedures have been employed in GA to obtain continuous results.

Finally, it is important to emphasize once again that in all interpolation models developed in this section are completely *deterministic*. In other words, the surface being approximated is treated simply as an unknown *function*, $y(s)$, on region R , rather than as the realization of an unknown *spatial stochastic process* $Y(s) = \mu(s) + \varepsilon(s)$, on R . The deterministic approach may be quite appropriate for applications such as “filling in” surface elevations from a subset of observed values, where such elevations can reasonably be assumed to vary continuously. But for spatial data such as nickel deposits example, where local variations can be quite substantial, it is generally more useful to treat such variations as random fluctuations, $\varepsilon(s)$, about a deterministic trend function, $\mu(s)$, representing the mean values of a stochastic process, $Y(s) = \mu(s) + \varepsilon(s)$. Hence we now turn to a consideration of this stochastic approach to spatial prediction.