CHAPTER 13

Methods of Data Analysis: Spatial Data

Numerical Methods in the Hydrological Sciences
Special Publication Series 57

doi:10.1029/057SP18

13. Methods of Data Analysis: Spatial Data

13.1. Background

One-dimensional spatial data are much like time series, and many of the same techniques can be used. Often, however, the spatial data we are interested in are two or three-dimensional. In hydrology, these include topographic data, measurements of soil properties such as hydraulic conductivity and soil moisture, groundwater heads, and contaminant concentrations. A common step in analysis of spatial data is to contour the data. In some cases the data may be evenly spaced, but more often they are irregularly spaced. Contouring provides information about the distribution and gradients of a spatially distributed parameter, but typically not about the uncertainty in those distributions or gradients. A variety of geostatistical methods, such as kriging, have been developed that can provide confidence limits on spatially interpolated data. In this chapter, we introduce the basics of contouring and geostatistics. Much more sophisticated formulations of these methods are available in freely or commercially available geographic information system (GIS) software packages. Our goal is to introduce the general concepts and some of the issues that must be considered when working with spatial data.

To illustrate the methods and functions presented in this chapter, we will use measurements of land-surface and water-table elevation from Oak Ridge National Laboratory (data file ornlgw.dat). The data, which are not regularly spaced, include the "Easting" (x) and "Northing" (y) for each pair of ground (z_g) and water-table (z_w) elevations (all in feet).

13.2. Interpolating irregularly spaced data onto a regular grid

There are typically three steps involved in contouring data. First, irregularly spaced points are interpolated onto a regular grid. Second, the locations of points corresponding to the contour values is determined. Finally, a technique must be chosen for connecting the points defining each contour.

MATLAB's contouring program assumes that the variable to be contoured is specified on a regular grid. We can use the *MATLAB* function `griddata` to interpolate irregularly spaced data onto a regular grid, but first we have to specify the grid. We can use the *MATLAB* function `meshgrid` to do this[1].

```
MESHGRID   X and Y arrays for 3-D plots.
   [X,Y] = MESHGRID(x,y) transforms the domain specified by vectors
   x and y into arrays X and Y that can be used for the evaluation
   of functions of two variables and 3-D surface plots.
   The rows of the output array X are copies of the vector x and
   the columns of the output array Y are copies of the vector y.

   [X,Y] = MESHGRID(x) is an abbreviation for [X,Y] = MESHGRID(x,x).
   [X,Y,Z] = MESHGRID(x,y,z) produces 3-D arrays that can be used to
   evaluate functions of three variables and 3-D volumetric plots.

   For example, to evaluate the function  x*exp(-x^2-y^2) over the
   range  -2 < x < 2,  -2 < y < 2,
```

[1] Reprinted with permission of The Mathworks, Inc.

```
[X,Y] = meshgrid(-2:.2:2, -2:.2:2);
Z = X .* exp(-X.^2 - Y.^2);
mesh(Z)
```

The following m-file uses the `meshgrid` function to generate a grid for the Oak Ridge data.

```
ornlgw=tblread('ornlgw.dat');
e=ornlgw(:,1);   %Easting (ft)
n=ornlgw(:,2);   %Northing (ft)
gse=ornlgw(:,3);  %ground surface elevation (ft)
wse=ornlgw(:,4);  %water table elevation (ft)
minx=min(e); maxx=max(e);
miny=min(n); maxy=max(n);
dx=100; dy=100;   %grid spacing
[ei,ni]=meshgrid(minx-dx:dx:maxx+dx,miny-dy:dy:maxy+dy);
plot(ei,ni,'c',ei',ni','c',e,n,'ro')
axis ('equal')   %makes the x and y axis intervals the same
```

and produces this result

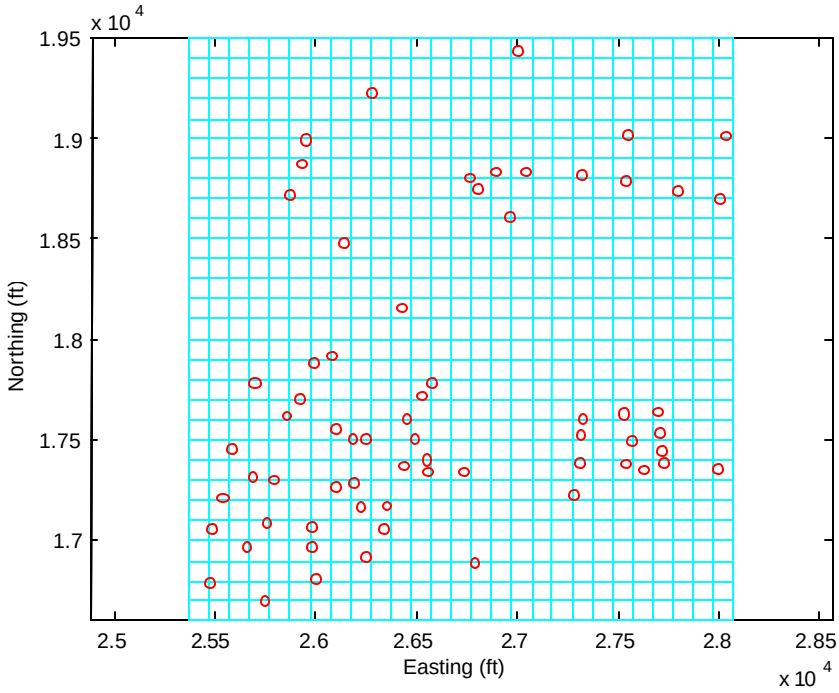

Figure 13.1. Distribution of Oak Ridge sampling points and proposed grid.

Once we have generated the grid, we can use `griddata` to interpolate the data onto the grid. The simplest gridding algorithms average the data values at the *n* control (data) points closest to a grid point. However, when the control points are unevenly distributed, this method can yield poor results. The grid interpolation methods in *MATLAB* (version 5 and later) are based on Delaunay triangulation. In this method, a triangular mesh is formed from the control points by connecting each point to its "natural neighbors". The resulting mesh of lines forms a triangular tesselation of the region with no intersecting lines except at the control points.

There are many ways to triangulate a region. Which are the "right" points to connect? Consider the subset of 9 Oak Ridge sampling locations indicated by the blue ×'s and the red ∗ in Figure 13.2. There are 28 triangles we can draw that include the location indicated by the ∗. The secret of Delaunay triangulation lies in the fact that each set of 3 points that defines a triangle also defines a circle. When all possible circles are formed, the circles with no interior data points (i.e., no data (control) points lie within the circles) identify the points that should be connected into triangles. For example, Figure 13.2 shows the 9 smallest circles that go through the red ∗ and two of the remaining 8 points (×'s). The circles shown with solid red lines are the smallest three. None of the red circles contains another point while each of the light blue circles does. The triangles (blue lines) are defined by these natural neighbors of the ∗ point. There will also be triangles that connect to the point on the far right of region shown in Figure 13.2; these will be larger than the ones on the left because of the larger distance between the points. One of the properties of a Delaunay triangulation is that the triangles are as close to equilateral as is possible given the locations of the data points. This helps to improve the accuracy of interpolations based on the triangulation. See Watson (1992) or Davis (2002) for more details of Delaunay and other triangulation schemes.

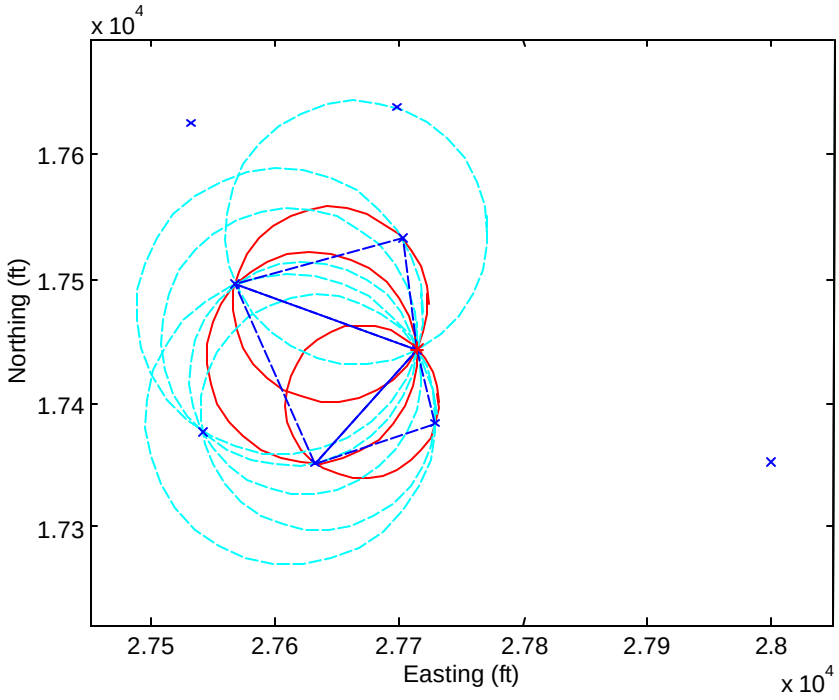

Figure 13.2. Delaunay triangulation of a subset of the Oak Ridge sampling sites.

The *MATLAB* command `delaunay` provides an implementation of this method. The following continuation of the gridding m-file uses this function to calculate the Delaunay triangulation of the Oak Ridge sampling locations.

```
plot(e,n,'o'); axis('equal'); hold on
dtm=delaunay(e,n);
trimesh (dtm,e,n,gse);
hold off; hidden off
```

The resulting triangulation is shown in Figure 13.3. Consult the *MATLAB* help file for more information about the function `delaunay`.

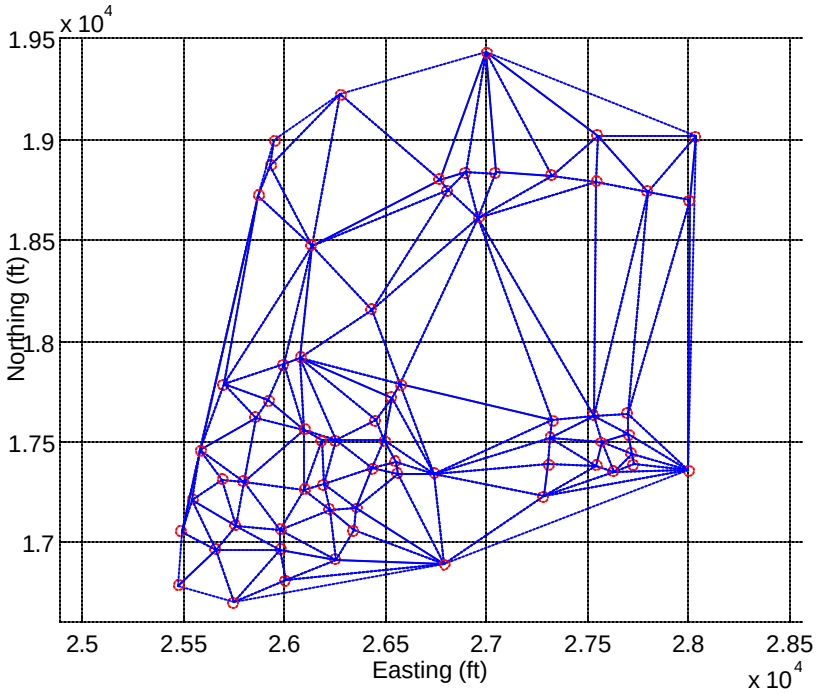

Figure 13.3. Delaunay triangulation of Oak Ridge sampling sites.

Now we have to consider how to use the known data values (control points) to interpolate the data onto our grid. One possibility is to assign to each point on the grid the value of the nearest control point. We wouldn't even need to triangulate the region for this method, but, as you might expect, the resulting surface is very blocky and not acceptable for most purposes. Another possibility is to average the three natural neighbors for a given grid location – the three control points defining the triangle in which the grid point falls. However, if a grid point is much closer to one of the control points than the other two, an equal weighting of the 3 control points might not give the best value. Intuitively, it seems that we might get the best results if we used some sort of weighted average of the natural neighbors. One way of weighting the nearest control points is by a function that depends inversely on the distance from the grid point.

The *MATLAB* `griddata` function performs the necessary interpolation. The help file for `griddata` is given below[2]. There are 3 options for how to do the interpolation: 'linear', 'cubic', and 'nearest', as well as a method carried over from an older version of *MATLAB*. 'Linear' and 'cubic' use weighted averages, while 'nearest' assigns the value of the nearest control point; 'cubic' produces a smooth surface

```
GRIDDATA Data gridding and surface fitting.
    ZI = GRIDDATA(X,Y,Z,XI,YI) fits a surface of the form Z = F(X,Y)
```

[2] Reprinted with permission of The Mathworks, Inc.

```
to the data in the (usually) nonuniformly-spaced vectors (X,Y,Z)
GRIDDATA interpolates this surface at the points specified by
(XI,YI) to produce ZI.  The surface always goes through the data
points.  XI and YI are usually a uniform grid (as produced by
MESHGRID) and is where GRIDDATA gets its name.

XI can be a row vector, in which case it specifies a matrix with
constant columns. Similarly, YI can be a column vector and it
specifies a matrix with constant rows.

[XI,YI,ZI] = GRIDDATA(X,Y,Z,XI,YI) also returns the XI and YI
formed this way (the results of [XI,YI] = MESHGRID(XI,YI)).

[...] = GRIDDATA(...,'method') where 'method' is one of
    'linear'    - Triangle-based linear interpolation (default).
    'cubic'     - Triangle-based cubic interpolation.
    'nearest'   - Nearest neighbor interpolation.
    'v4'        - MATLAB 4 griddata method.
defines the type of surface fit to the data. The 'cubic' and 'v4'
methods produce smooth surfaces while 'linear' and 'nearest' have
discontinuities in the first and zero-th derivative respectively.  All
the methods except 'v4' are based on a Delaunay triangulation of the
data.

See also INTERP2, DELAUNAY, MESHGRID.
```

The gridded surface calculated with `zgi=griddata(e,n,gse,ei,ni,'linear')` is plotted in Figure 13.4.

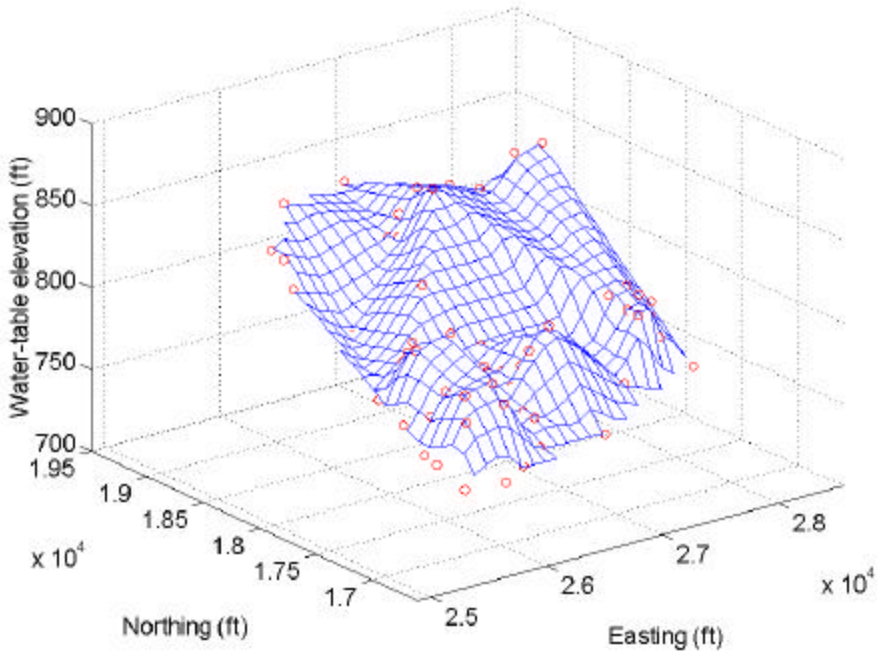

Figure 13.4. Surface elevations of the Oak Ridge sampling region calculated using linear interpolation.

13.3. Contouring

Once the data have been interpolated onto a regular grid, we can consider how to contour them. When we contour data, we are connecting points of equal value of some spatially distributed variable. The resulting curves are constrained by some contouring rules (after Middleton, 2000):

1. Contour lines don't cross or merge with other contour lines.
2. Contour lines pass between known points having higher and lower values than the contour value.
3. Contour lines are repeated to indicate a reversal of slope.
4. Contour lines must close or end at the boundaries of the region.

In addition, there are some conventions commonly adopted in contouring, such as use of a standard datum and equal contour intervals.

MATLAB has a built-in contouring routine, `contour`, that works with gridded data. The function finds the contour points and connects them with straight lines. Smoother contours require a finer grid. The command

```
[c,h]=contour(ei,ni,zi,750:10:850)
```

followed by

```
clabel (c,h,[760 780 800 820 840])
```

produces the contour plot shown in Figure 13.5a. The gridding upon which these contours were drawn is based on *MATLAB*'s default linear weighting. If instead we grid the data using the cubic weighting function, we obtain the contours plotted in Figure 13.5b. While the two contour maps are generally similar, they do differ in some details as a result of the different interpolation methods. See Davis (2002) for a more thorough discussion of interpolation and contouring. It is worth noting that manual contouring was typically done directly from a triangulation without gridding the data. The resulting contours are similar to those found from gridded data except in places where the triangles are very acute and differences in slope of the faces of the triangles is relatively large (Davis, 2002).

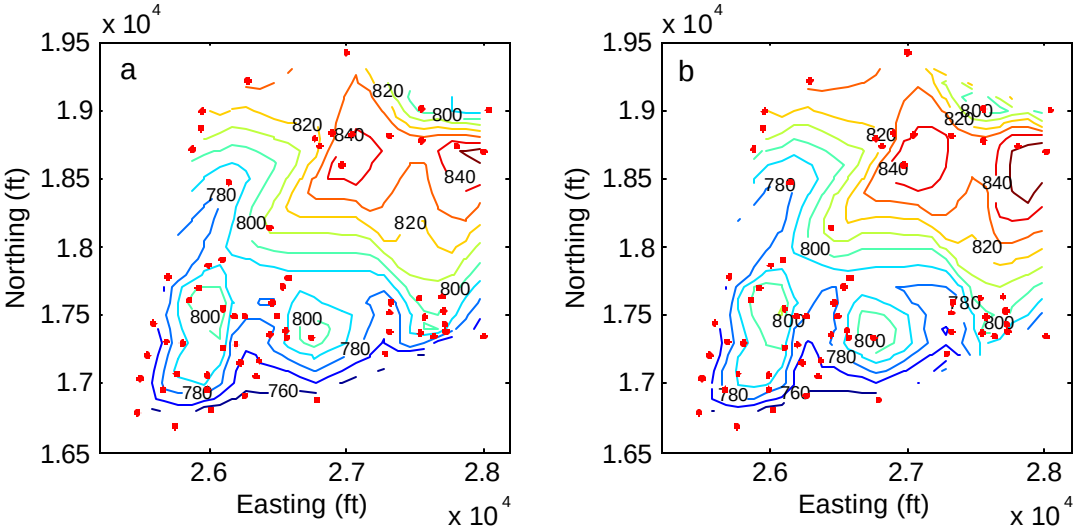

Figure 13.5. *MATLAB*-generated contours using (a) linear and (b) cubic weighting to grid the data.

13.4. Semivariance

Surfaces can be smooth, bumpy, ridged, or some combination. On a smoothly varying surface, a control point quite a distance from a point of interest might be useful in estimating the value of a variable at that point. In contrast, a nearby control point on a bumpy surface may not be useful at all. Intuitively, we might expect that if we could define the distances over which spatially distributed data were correlated, that some length scale related to the correlation distances might provide the basis for improved gridding and contouring algorithms.

We can characterize temporal correlations in a time series by calculating the autocovariance. This is done by taking two representations of a time series, sliding them past each other in time-step increments (lags), and calculating the covariance of the two time series. The autocorrelation is the autocovariance divided by the variance of the time series. The autocovariance of a time series at zero lag is equal to the variance, so the autocorrelation of a time series is 1 for lag = 0 and generally decreases with increasing lag. If the time series is periodic, then the autocorrelation will rise again at lags equal to the periodicities contained in the time series.

For spatial data, a related function called the semivariance typically is used to characterize the spatial correlation. The semivariance for a one-dimensional spatial series of equally spaced values is given by

$$g_h = \frac{1}{2n}\sum_{i=1}^{n-h}(z_i - z_{i+h})^2 \tag{13.1}$$

where z represents the data values, h is the lag index and n is the length of the series (see, e.g., Middleton, 2000; Davis, 2002). In general, the semivariance is smallest when the lag is close to 0 and increases as the lag increases up to some maximum value (termed the 'sill'), after which the semivariance remains roughly constant at a value near the variance of the whole data series. The sill is reached at a lag distance $h = a$, where a is termed the 'range'. At separations greater than the range, data values are essentially uncorrelated. Several models for the general form of a semivariogram are shown in Figure 13.6. In these models, the semivariance is zero at a lag of zero and rises to a normalized semivariance of 1 (g_h = var(z)) at a normalized lag h/a near 1. A semivariance model is typically fit to a semivariogram calculated from experimental data to provide a continuous description of the semivariance for use in interpolation methods such as those described in the next section. A linear semivariogram model has the form

$$\begin{aligned} g_h &= a\,h \quad \text{for } h < a \\ g_h &= \text{var}(z) \quad \text{for } h \geq a \end{aligned} \tag{13.2}$$

while a spherical model is described by

$$\begin{aligned} g_h &= \text{var}(z)\left(\frac{3h}{2a} - \frac{h^3}{2a^3}\right) \quad \text{for } h < a \\ g_h &= \text{var}(z) \quad \text{for } h \geq a \end{aligned} \tag{13.3}$$

(Davis, 2002).

Equations for the exponential model and more information about semivariance models in general can be found in Davis (2002) or Middleton (2000).

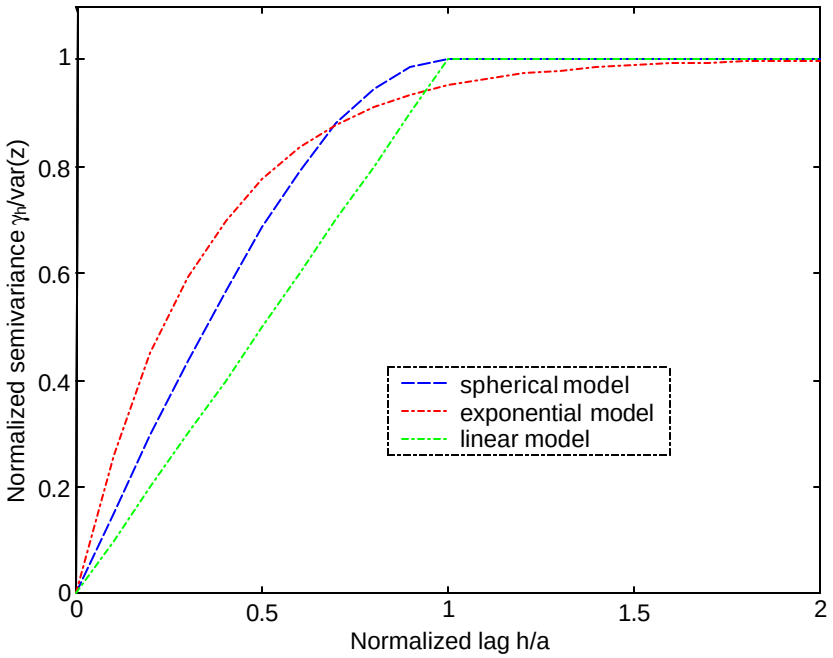

Figure 13.6. Theoretical semivariance models

To calculate the semivariogram for irregularly spaced experimental data, each point is taken pairwise with all of the other points in the region of interest, the distance between them (in the *x*-*y* plane) determined, and the squared difference in *z* values (e.g., surface elevation) calculated. Then, the results can be aggregated into distance bins (or directional bins) and averaged to obtain the semivariogram. The following m-file, `semivar.m`, does this using distance bins, but could be adapted for directional bins. By default, `semivar.m` used 20 distance bins; the code can easily be modified to change the number of bins. The surface should be detrended prior to calculating the semivariogram. An m-file (`detrendsurf.m`) to remove a 1st-order trend surface (plane) from a set of spatial data (input as an N×3 matrix of *x*, *y* (location) and *z* (data) values) is included with the m-files for this chapter.

```
%semivar.m
data=tblread('ornlgw.dat');
X=data(:,1:3)  %select x, y, and ground-surface elevation data
dtX=detrendsurf(X); %detrend surface
x=dtX(:,1); y=dtX(:,2); z=dtX(:,3); %assign x,y,z values
n=length(x); nn=n*n;
%calculate distance between each pair of points
xx=repmat(x,1,n);
yy=repmat(y,1,n);
zz=repmat(z,1,n);
dx=xx-xx'; dy=yy-yy';
h=sqrt(dx.^2+dy.^2);
%eliminate duplicate values in the lower triangular part of the matrix
% and sort values by distance
```

```
hv=reshape(triu(h),nn,1);
[nr,nc,nhv]=find(hv);  %nhv are the nonzero values of hv
[sh,ih]=sort(nhv);
%calculate the squared differences in z values for each pair of points
zvar=(zz-zz').^2; vz=reshape(zvar,nn,1); nvz=vz(nr); sv=nvz(ih);
%only compare distances that are half or less of the largest distance
hmax=max(sh)./2; nm=length(find(sh<=hmax));
nbin=20; hinc=hmax/nbin; %set the number of distance bins to 20
% adjust bin width to be round number
% The values for the magnitude of hinc (10, 100, 1000) may need to be
% adjusted depending on domain size and distance units.
if hinc<=10, hinc=floor(hinc);
   elseif hinc<=100, hinc=floor(hinc/10)*10;
   elseif hinc<=1000, hinc=floor(hinc/100)*100;
   end;
hvec=0:hinc:nbin*hinc; %vector of bin endpoints
for i=1:nbin-1;
   ib=[]; ib=find(sh>hvec(i)&sh<=hvec(i+1)); %find distances in each bin
   if ~isempty(ib),
      ni(i)=length(ib);
      gamh(i)=sum(sv(ib))./(2*ni(i)); %calculate semivariance for each bin
   else gamh(i)=0; ni(i)=0; %set semivariance to 0 if no data in a bin
   end;
end;
hvmn=(hvec(1:nbin-1)+hvec(2:nbin))./2; %find the mid-point of each bin
plot(hvmn,gamh,'bo') %plot the results
xlabel('Distance (ft)')
ylabel('Semivariance')
```

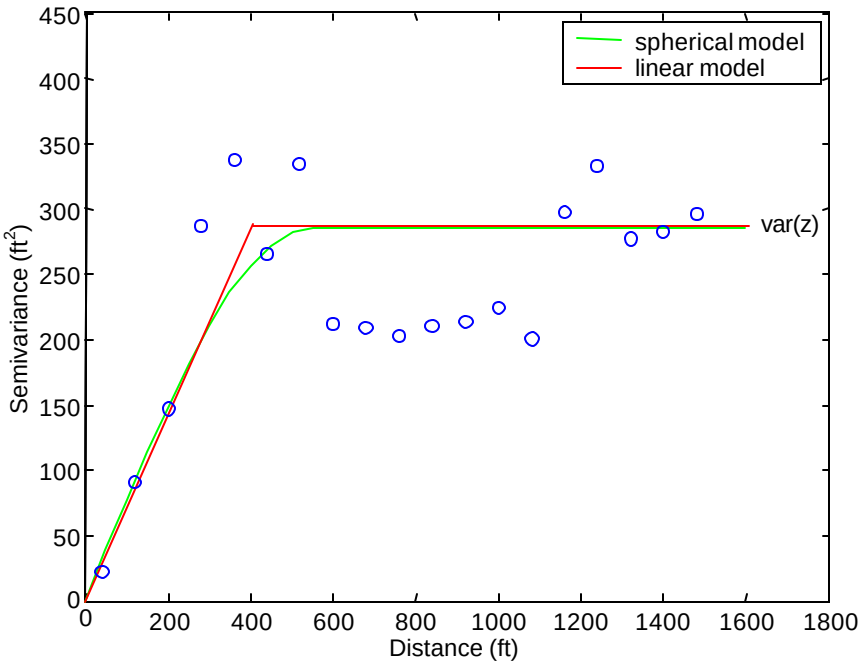

Figure 13.7. Calculated semivariance and fitted models for the detrended Oak Ridge elevation data.

The semivariogram shown in Figure 13.7 was calculated using `semivar.m` for the Oak Ridge ground surface elevation data. It doesn't look as neat as the model semivariograms shown previously, but the values appear to reach the sill (g_h = var(z)) at a range (distance) of about 400 ft. Conceptually, the range, a, indicates the maximum distance over which the data are spatially correlated. As a result, we might expect to obtain better results when performing spatial interpolations if we weight neighboring points within the range most heavily. This is the basis for the method known as *kriging*. Fitting the surface elevation semivariogram by eye using a linear model (equation 13.2) gives $\alpha = 0.7$ and $a = 400$ (Figure 13.7). Fitting a spherical model gives $a = 550$ (Figure 13.7). Middleton (2000, p. 154) notes that "the form of the semivariogram is generally not very well known, with the data showing large deviations from the theoretical models that are used."

13.5. Kriging

Kriging, a technique developed by D.G. Krige for mining purposes, provides an alternative approach to gridding irregularly spaced data. In kriging, the semivariogram is used to determine the best weights to apply to the control points when performing spatial interpolation. It also provides an estimate of the error associated with the predicted value. There are many kriging packages available as shareware or for purchase. A simple m-file is provided below that does punctual or ordinary kriging in which the mean of the variable to be interpolated is assumed to be constant over the region containing the points used to construct the semivariogram; a more general procedure called universal kriging does not make this assumption (Davis, 2002). The purpose of this m-file is mostly to illustrate the basic ideas of kriging; more sophisticated programs are needed for more general application.

The m-file `ordkrig.m` follows the mathematical formulation provided in Davis (2002). The basic idea is that we can estimate the value of z at a specified point (x_0, y_0) as:

$$z_0 = \sum_{i=1}^{k} l_i z_i \tag{13.4}$$

where l_i are weights, z_i are the values of z at the control points and k is the number of observations used in the estimate. The problem is solved using a Lagrange multiplier with the constraints that the weights result in minimum error variance and sum to 1 (see Davis, 2002). The weights are estimated by the matrix equation

$$\mathrm{L} = \mathbf{S}^{-1}\mathbf{B} \tag{13.5}$$

where L =[l_1, l_2, … l_k, m]' (column vector), m is the Lagrange multiplier, **S** is the matrix of semivariances for each of the pairs of control points being used, and **B** is a column vector of semivariances for each of the k control points paired with the point being interpolated, z_0. Using this, (13.4) can be rewritten as

$$z_0 = \mathbf{Z}\mathrm{L} = \mathbf{Z}\mathbf{S}^{-1}\mathbf{B} \tag{13.6}$$

where **Z**=[z_1, z_2, …z_k, 0] (row vector). The variance of the estimate is given by

$$\mathrm{var}(z_0) = \mathbf{B}'\mathrm{L} \ . \tag{13.7}$$

Calculated variances can be converted to 95% confidence intervals on the predicted value of z as $ci_{95} = \pm 1.96\sqrt{\mathrm{var}(z_0)}$ (Davis, 2002).

```
%ordkrig.m
%uses ordinary kriging and a spherical semivariance model to predict the
%elevation z0 and its associated variance at a specified location x0, y0.

data=tblread('ornlgw.dat');
X=data(:,1:3);  %select x, y, and ground-surface elevation data
[dtX,b]=detrendsurf(X); %detrend surface
x0=input('enter the x coordinate of the point to be estimated: ');
y0=input('enter the y coordinate of the point to be estimated: ');
mz0=b(1)+b(2)*x0+b(3)*y0; %z0 value on trend surface
svrange=input('enter the range for a spherical semivariance model \n fit to
the control data: ');
x=[dtX(:,1);x0];
y=[dtX(:,2);y0];
z=dtX(:,3);
n=length(x);
xx=repmat(x,1,n);
yy=repmat(y,1,n);
dx=xx-xx'; dy=yy-yy';
h=sqrt(dx.^2+dy.^2); %calculating the distances between points
h0=h(end,:); %distances from the estimation point
dx0=dx(end,:); dy0=dy(end,:);
hn=find(h0<svrange); %select points within range to use in estimate
dxn=dx0(hn); dyn=dy0(hn);
hh=sqrt(dxn.^2+dyn.^2); %dist. of selected points from estimation point
xi=x(hn); yi=y(hn); m=length(xi);
xxi=repmat(xi,1,m);
yyi=repmat(yi,1,m);
dxxi=xxi-xxi'; dyyi=yyi-yyi';
hhi=sqrt(dxxi.^2+dyyi.^2);
gamh=var(z)*(1.5*hhi/svrange-0.5*(hhi/svrange).^3);%calculate semivariance
S_top=[gamh(1:m-1,1:m-1) ones(m-1,1)];
S_last=[ones(1,m-1) 0];
S=[S_top;S_last]; %construct semivariance matrix S
B=[gamh(1:m-1,end);1]; %semivariance vector B
lamv=S\B; %lamv are the weights
zi=lamv(1:m-1)'*z(hn(1:m-1)); %calculate estimated z
z0=zi+mz0; %add in trend surface value for z0
var0=lamv'*B; %calculate the variance in estimate of z
fprintf(1,'Estimated z0 = %5.0f at x0 = %7.2f, y0 =%7.2f \n',[z0,x0,y0])
fprintf(1,'Variance of estimate of z0 = %5.0f',var0)
```

As an example, `ordkrig.m` was used to predict the value of surface elevation at $x_0 = 2.60\times10^4$ ft, $y_0 = 1.73\times10^4$ ft, an area of topographic variation but also many control points. The range from the fit of the spherical semivariance model to the Oak Ridge data (550 ft, Figure 13.7) was used in the calculation. The predicted elevation is $z_0 = 800 \pm 19$ ft (var(z_0) = 91 ft^2). If a location in a 'smoother' region is selected, e.g., $x_0 = 2.70\times10^4$ ft, $y_0 = 1.80\times10^4$ ft (see Figures 13.4 and 13.5), `ordkrig` gives $z_0 = 789 \pm 42$ ft (var(z_0) = 461 ft^2). There is a larger error associated with this estimate because there are fewer nearby control points, suggesting we should question whether the surface really does vary smoothly in the middle of the region or whether the smoothness is an artifact of the lack of control points in that area.

13.6. Problems

One of the most common and challenging applications of the contouring and geostatistical techniques considered in this chapter is contouring topographic data. A data set from Davis (1973) has become a common test set for contouring algorithms and is provided as file davistopo.dat. Note that the *x* and *y* values are given in terms of unit distances, where one unit is 50 ft with the origin at the southwest corner of the map. The unit distances should be converted to real distance for use in `semivar.m` (or the ranges of `hinc` in `semivar.m` adjusted to allow for `hinc` < 1).

1. Develop an m-file that plots the spatial distribution of points and the Delaunay triangulation of the surface. Then calculate gridded values and contour them. Investigate the effect of grid spacing and interpolation weighting on the resulting contour plots. There is a stream network through this region. Try to identify where the stream channels are based on your contour plot. Consult Davis (2002, p. 373; also shown in earlier editions of the book) for a map showing the channel locations.

2. Use `semivar.m` to calculate the semivariogram for this data set and fit it with a linear and spherical model (equations 13.2 and 13.3). Limit the maximum semivariance to the overall variance of the data set. Then use `ordkrig.m` to grid the values. [You may want to convert `ordkrig` to a function that returns the interpolated value and variance for given *x*,*y* values.] Determine not only the gridded elevations, but also the surface defined by the error variance at each interpolation point. What does this reveal about the interpolated elevations? Contour the resulting gridded values and compare the results to those obtained in Problem 1.

13.7. References

Davis, J.C., *Statistics and Data Analysis in Geology*, 638pp, Wiley, New York, 2002.

Middleton, G.V., *Data Analysis in the Earth Sciences Using Matlab*, 260 pp., Prentice Hall, Upper Saddle River, NJ, 2000.

Watson, D.F., *Contouring*, 321pp, Pergamon Press, Oxford, 1992.