A large component of research in the Earth sciences is devoted to collecting data, say \( z_i \), that is tied to individual locations \((x_i, y_i)\). Examples range from small (localized) studies (e.g., \( z \) is the surface roughness on a fault surface in a laboratory specimen) to regional investigations of the variations in topography, the earth’s gravity and magnetic field, heat flow, ocean surface temperatures, etc. The most common end product of the data analysis will often be a map that displays the variation in the observed quantity over the area in question. At other times we are more interested in predicting values at intermediate locations where no observation exists. There is often considerable overlap between these goals and we will see that we will often need to follow both paths. For simplicity we will only consider the making of contour maps. Other map forms (shaded or colored images) share most of the same underlying principles and assumptions.

While most contouring techniques require the data to be evenly distributed on a grid (one reason to consider gridding) there are other methods. One such is triangulation, which we will look at today.

**Gridding** is the name of any operation that re-samples a data set given at arbitrary locations onto an equidistant grid or lattice. There are many methods available for achieving this, and we will discuss the most useful. The following introduces some notation:

```
gridlines
node
control point
```

While the gridding algorithms may differ substantially in how they work, they all share some underlying assumptions:

1. Function is single-valued
2. Function is continuous inside region
3. Function is positively auto-correlated over a length \( \geq \) distance between observation points.

Assumption 1 almost always applies without any sacrifice. For the most part, assumption 2 is reasonable. In those cases that clearly violate the assumption, e.g., data with steps or faults, special routines that subdivide the region into sub-regions where assumption 2 is valid may be used. Almost all gridding methods estimate the function using weighted averages of nearby data points, a procedure justified by assumption 3 in particular.

Although rarely pointed out, some methods also require that the control points do not containing information at wavelengths shorter than twice the grid spacing so that spatial aliasing will not occur. Pre-filtering the data will avoid this problem. Thus, one should
keep in mind what the preferred Nyquist interval should be when deciding on the gridding interval. There are several areas of concern in evaluating a gridding algorithm, and their relative importance will depend on the application.

The most common gridding techniques include

1. Triangulation
2. Moving averages and other spatial filters
3. Cubic spline interpolations and their siblings
4. Statistical methods (Kriging)
5. Imposed conditions methods (Projections onto Convex sets (POCS))

While only some of these techniques can be found in Matlab (often donated by Matlab users), they are implemented in many commercial and public domain software packages. Rather than designing your own program, you will likely be using someone else’s implementation. It is therefore extremely important that you fully understand what the black box operation does with your data. Such knowledge will allow you to distinguish between true and artificial features in the final product (e.g., a contour map), be aware of the limitations of the method, help you choose the best technique for your situation, and guide you when interpreting the results. In all cases it is important to show the underlying data distribution so that viewers may judge which contours have data constraints to support them.

**Triangulation**

The earliest implementations of computer contouring relied on triangulation. One can always draw lines between the neighboring points and obtain a triangular mesh of variable size:

Once the triangles have been identified, it is relatively straightforward to contour the surface: Each triangle defines a plane, whose equation \( z = ax + by + c \) we may determine from the \( z \)-values at the vertices. To contour a particular level, simply fix the \( z \)-value in the plane-equation and draw the resulting \( y = f(x) \) line within the boundaries of each separate triangle.

All this sounds simple enough, until we realize that there are many ways to connect the vertices and vastly different triangle-configurations will result. How to select the best configuration? It turns out that one can find optimal networks that are unique for a given set of points. These networks result in triangles that are as equilateral as possible. Such triangles are called Delauney triangles.

One can imagine that every point is surrounded by an irregular polygon such that any location within the polygon is closer to the enclosed point than it is to any other control point. This is the most compact way to divide space. Such polygons are often associated with the names Thiessen, Dirichlet, and Voronoi, and arise in many fields. For instance,
soap bubbles form easily recognizable network of polyhedra, the 3-D equivalent of polygons. The Voronoi polygons are found by connecting the chosen point with all others by a straight line, then finding the line’s midpoint. The innermost segments of these lines and their intersections define the Voronoi polygon. Once all the Voronoi polygons have been found we simply connect the central points in neighboring polygons. This operation results in drawing the Delauney triangles. The neighboring points determined in this way are called the Thiessen neighbors. Triangulation methods consist of finding the Thiessen neighbors of successive points on a map. The geometric solution is easily programmed on a computer and can of course be combined with the \( z \) values to give a contour map of the arbitrarily positioned observations. Davis explains the implementation of such a solution.

The triangulation process will make up a network that is circumscribed by a convex hull. To complete a rectangular region we must add a few guiding points along the map border. By evaluating the triangular planes at equidistant points on a lattice we effectively “grid” the observations. However, using triangular, plane segments as a model for the underlying function may not give the best result. It is also possible to fit cubic functions for each triangles, yielding a smoother interpolation.

**Moving averages**

The next gridding algorithm we will consider is called moving averages. What do we mean by MA gridding? It is any operation that involves taking into account only those control points that lie inside a circle of specified radius centered on the grid node. Points outside this region will have no influence on the estimated nodal value; hence such methods are local. The local methods differ in how they estimate the nodal value.

The simplest approach calculates a weighted average of all the control points inside the area of influence. It seems reasonable to use the distance \( D_{ik} \) from each control point \( i \) to the node \( k \) when designing the weights. A simple form of the weighting function is

\[
W_{ik} = \frac{1}{D_{ik}}
\]

Then, the node estimate becomes

\[
z_k = \frac{\sum_{i=1}^{n} z_i W_{ik}}{\sum_{i=1}^{n} W_{ik}}
\]

(24.2)

where \( n \) is the number of points inside the search radius \( R \). As we apply this operation to
all the nodes we see why it is called a moving average. Looking closer at the weights we see that they are hyperbolic. In particular, if a data point is very close to the node, it will be given tremendous weight. This can lead to short-wavelength noise since the radius of the influence region effectively approaches zero. To avoid this and preserve some of the correlation with nearby points, we can redefine the weights as

$$w_{ik} = \frac{1}{1 + c(D_{ik} R)^2},$$

(24.3)

where $R$ is radius of influence and $c$ is a constant (typically 9). This function approximates the slightly more appropriate but slower-to-compute Gaussian weights.

Other weighting schemes may also be considered. Some methods use the data inside the zone of influence and fit a plane to the data. The value of the plane at the node is used as the node estimate. If the $n$ points inside the area have coordinates $x_i, y_i, z_i$, we find the plane by least-squares:

$$E = \sum_{i=1}^{n} (ax_i + by_i + c - z_i)^2 = \text{minimum}$$

(24.4)

Setting the partial derivatives to zero gives the normal equations

$$2 \sum (ax_i + by_i + c - z_i)x_i = 0$$
$$2 \sum (ax_i + by_i + c - z_i)y_i = 0$$
$$2 \sum (ax_i + by_i + c - z_i) = 0$$

(24.5)

or

$$\begin{bmatrix}
\Sigma x_i^2 & \Sigma x_i y_i & \Sigma x_i \\
\Sigma x_i y_i & \Sigma y_i^2 & \Sigma y_i \\
\Sigma x_i & \Sigma y_i & n
\end{bmatrix}
\begin{bmatrix}
a \\
b \\
c
\end{bmatrix}
= 
\begin{bmatrix}
\Sigma x_i z_i \\
\Sigma y_i z_i \\
\Sigma z_i
\end{bmatrix}$$

(24.6)

which can be solved by a number of standard techniques. The estimated value of the node $(\hat{a}, \hat{b})$ is then simply

$$\hat{z} = a \hat{x} + b \hat{y} + c$$

(24.7)

We can call this method "piecewise linear least squares." However, more complicated surfaces than a plane could also be used, like quadratic or cubic surfaces. Their development is similar to that of the plane. While such methods work reasonably well when enough control points are present (about anything works well when there are enough data!), they produce wild estimates in unconstrained areas.

Variants on this theme exist. A common method is to fit planes centered on all control points rather than the nodes. The planes are constrained to go through the control point they are centered on. Thus for each control point inside the circle of influence we must construct another circle of influence to find the control points to be used for each plane. All the planes can then be evaluated at the single node position, yielding several estimates for $\hat{z}$. The final estimate is selected as a weighted average of these individual plane projections, using a distance weight.