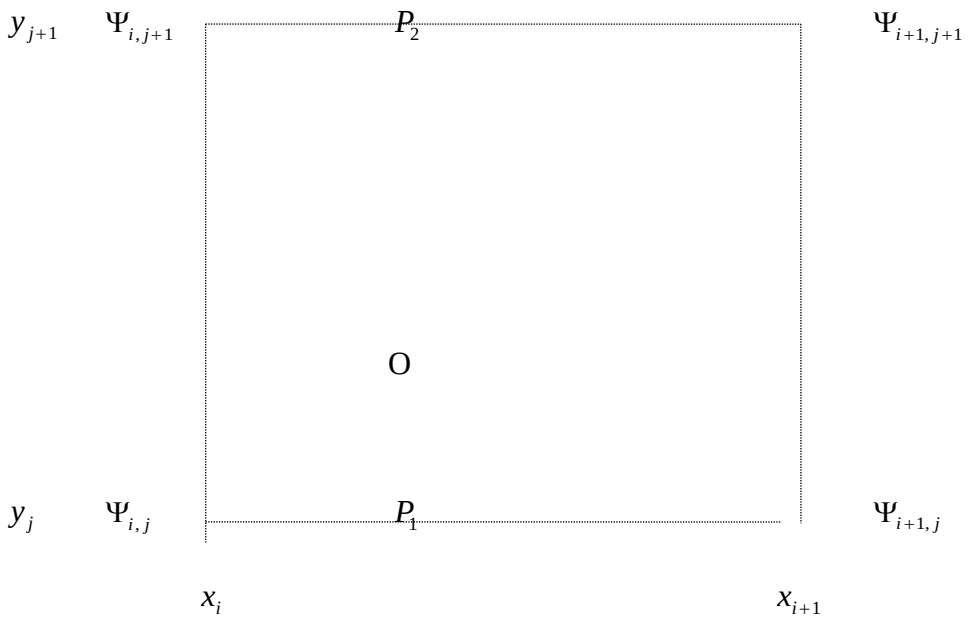
Interpolation between grid points

Bilinear interpolation

In data analysis, it is often necessary to estimate values between 4 grid points to a specific location. For example, one may want to estimate between model grid point values to a fixed point where meteorology measurements are taken, such as at a National Weather Service office. One way to achieve this is with *bilinear interpolation*. This should only be used for fields that are reasonably spatially smooth with linear gradients.

Consider the following 4 grid points with a variable Ψ at grid points (x_i, y_j), (x_{i+1}, y_j), (x_i, y_{j+1}), (x_{i+1}, y_{j+1})



Where $x_i = (i-1)\Delta x + x_o$, $y_j = (j-1)\Delta y + y_o$, and Ψ is some meteorology variable such as temperature, meridional wind, zonal wind, mixing ratio, etc. "O" is the point being estimated between the 4 grid points. x and y are a distance marker - often longitude and latitude.

Bilinear interpolation "weights" the value inversely proportionally to its distance from each of the four surrounding grid points. A derivation follows.

First, expressions for interpolating to points 1 and 2 may be written as:

$$P_1 = (1 - W_x)\Psi_{i,j} + W_x\Psi_{i+1,j} \qquad P_2 = (1 - W_x)\Psi_{i,j+1} + W_x\Psi_{i+1,j+1}$$

where the" weight" along the x axis may be expressed as:

$$W_x = \frac{x - x_i}{x_{i+1} - x_i}$$

Likewise, along the y axis, the weight is:

$$W_y = \frac{y - y_j}{y_{j+1} - y_{ji}}$$

Then one may interpolate to point "O" using the following equation:

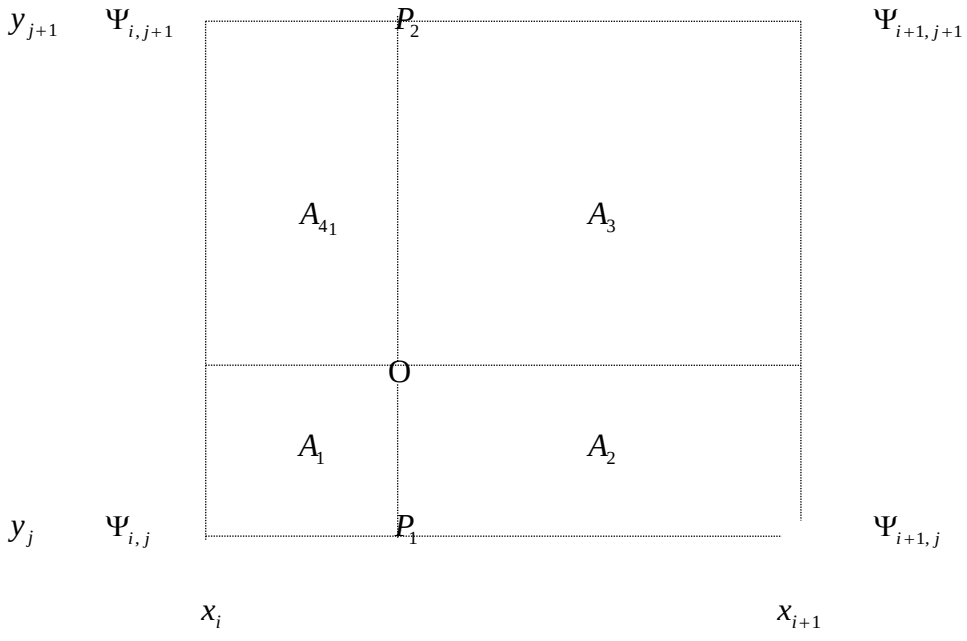
$$O = (1 - W_y) P_1 + W_y P_2$$

By substituting all the appropriate terms, this equation may also be written as

$$O = (1-W_y)(1-W_x)\Psi_{i,j} + (1-W_y)W_x\Psi_{i+1,j} + W_y(1-W_x)\Psi_{i,j+1} + W_yW_x\Psi_{i+1,j+1} \quad \text{[Eq. 1]}$$

Bilinear interpolation using area weights

An equivalent expression for Eq. 1 can be written by breaking the grid into 4 smaller rectangular points around "O" and computing the area for each of these rectangles:



$$O = \frac{A_3\Psi_{i,j} + A_4\Psi_{i+1,j} + A_2\Psi_{i,j+1} + A_1\Psi_{i+1,j+1}}{A_1 + A_2 + A_3 + A_4} \quad \text{[Eq. 2]}$$

Bilinear interpolation using grid indexes

There are two disadvantages of using Eq. 1 or Eq. 2.

1) To use these expressions, one must search for the 4 surrounding grid points using two do loops. This can be very slow if lots of interpolation is required on large datasets.
2) One assumes that the grid points are equally distributed. Most models use map projections which contains unequally spaced grid points. This also complicates the searching algorithms.

A method around these is to compute the *i* and *j* grid point location directly. This avoids searching for the 4 surrounding grid points, and also improves (but does not totally eliminate) the unequal spacing problem.

Computing i and j on map projections involves cumbersome trigonometric equations. These are sometimes documented on the model website. However, often a request has to be made to the model developer for a subroutine. The subroutine will be called something like "xy2ij".

For the simple rectangular grid shown above, we can rewrite the grid equations to solve for the interpolation point x and y as

$$x = (float\ i - 1)\Delta x + x_o \qquad y = (float\ j - 1)\Delta y + y_o$$

Then, the floating point i and j are computed from the interpolated location at x and y as:

$$float\ i = \frac{x - x_o}{\Delta x} + 1 \quad float\ j = \frac{y - y_o}{\Delta y} + 1$$

where *float* indicates that this value is a floating point number (i.e., has decimal points), and is not an integer.

Then the weights simply become:

$$W_x = float\ i\ - integer(float\ i) \qquad W_y = float\ j\ - integer(float\ j)$$

after canceling terms, and since the denominator becomes unity. The integer function simply truncates the floating point value. For example, a 3.4 becomes a 3, and a 3.9 also becomes a 3. It does not round up!

All four points are also known now. They are:

$$i = integer(float\ i) \qquad i + 1 = integer(float\ i) + 1$$
$$j = integer(float\ j) \qquad j + 1 = integer(float\ j) + 1$$

Now Eq. 1 may be used as before!

Nearest neighbor

In situations where data is spatially discontinuous, such as radar or satellite data, linear interpolation is invalid. If the grids points are reasonably dense, using the nearest neighbor technique is sometimes best. Nearest neighbor search, also known as proximity search, simply finds the closest grid point and sets that value to the new point.

One could search for the closest point, but again in large datasets this will run slow. But, just as in bilinear interpolation, if the index values are known, the calculation runs fast. Simply calculate the floating i and j, then round to the nearest whole number. For example, if the floating j is 3.4, round down to j=3. If the floating j is 3.6, round up to j=4. In computer languages, a common function is called *anint*. In spreadsheets, the function is called *round*. Check your software documentation for the appropriate function.

Below is an example using *anint*.

$$\Psi = \Psi_{anint(float\ i),anint(float\ j)}$$

So, for example,

$$\Psi = \Psi_{anint(4.3),anint(10.7)} = \Psi_{4,11}$$