

Appendix 2: Interpolation

Interpolating to Find a Value Between Two Known Values

One of the most common computational tricks in meteorology—and in science in general, for that matter—is interpolation. Interpolation is a technique by which you estimate the value of some variable at a location, given the values of the variable at locations on either side of the desired location.

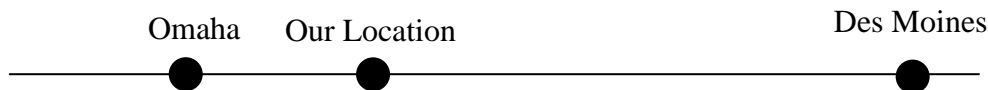
For example, suppose that we have the following situation. Suppose that we have the temperature in Omaha and the temperature F in Des Moines. We want to solve for the temperature at some location between Omaha and Des Moines. In order to do this, we are going to need the following pieces of information:

- the temperature in Omaha
- the temperature in Des Moines
- the location of Omaha
- the location of Des Moines
- *the location where we want to find the temperature*

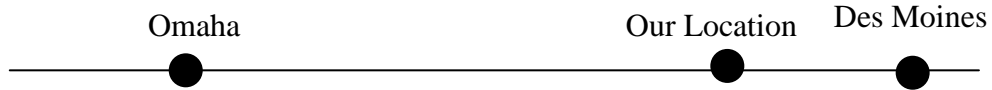
It's pretty obvious why we need the temperature in Omaha and Des Moines, but maybe less obvious why we need to know the locations of Omaha, Des Moines and the spot where we want to find the temperature. Consider this map:



In this situation, the temperature at “Our Location” should clearly be between the values at Omaha and Des Moines. You might be tempted to just take the average of the temperatures at these two stations and call that “good enough”, but we want to solve for the temperature at our location by interpolation. That way, if our location is closer to Omaha than it is to Des Moines, the temperature at our location should be closer to the temperature in Omaha than it is to the temperature at Des Moines:



Similarly, if our location is closer to Des Moines than it is to Omaha, we want the temperature at our location to be closer to the temperature in Des Moines than it is to the temperature in Omaha:



Clearly, we are going to need some kind of number that expresses where our location is relative to the locations of Omaha and Des Moines. To do this, we need the locations of Omaha and Des Moines, expressed as values along some kind of axis (in this case, the x axis). Therefore, the x coordinate of Omaha's location will be x_{omaha} the x coordinate of Des Moines' location will be $x_{\text{desmoines}}$, and the x coordinate of our location will be $x_{\text{ourlocation}}$. Then the distance between Omaha and Des Moines would be:

$$x_{\text{desmoines}} - x_{\text{omaha}}$$

and the distance between Omaha and our location will be:

$$x_{\text{ourlocation}} - x_{\text{omaha}}$$

The quotient of these two numbers:

$$(x_{\text{ourlocation}} - x_{\text{omaha}}) / (x_{\text{desmoines}} - x_{\text{omaha}})$$

is then the fraction of the distance between Omaha and Des Moines that our location is. If you do the math, you'll see that this ratio is 0 if our location is at Omaha, 1 if our location is in Des Moines, and something between 0 and 1 if the location is between Omaha and Des Moines.

Given the temperatures at Omaha and Des Moines, we can use this quotient to interpolate for a temperature at our location. Clearly, if our location is near Omaha, we want the temperature at our location to be close to the temperature at Omaha. The same is true of Des Moines. Thus, clearly the *difference between the temperatures at Omaha and Des Moines* is important:

$$T_{\text{desmoines}} - T_{\text{omaha}}$$

Thus, to interpolate for the temperature at our location, we will use the following expression:

$$T_{\text{ourlocation}} = T_{\text{omaha}} - (T_{\text{desmoines}} - T_{\text{omaha}}) * (x_{\text{ourlocation}} - x_{\text{omaha}}) / (x_{\text{desmoines}} - x_{\text{omaha}})$$

A little experimentation will show you that when our location is at Omaha, the quotient is 0 and the equation reduces to:

$$T_{\text{ourlocation}} = T_{\text{omaha}} - (T_{\text{desmoines}} - T_{\text{omaha}}) * 0 = T_{\text{omaha}}$$

Similarly, if the location is at Des Moines, the quotient is 1 and the equation reduces to:

$$T_{\text{ourlocation}} = T_{\text{omaha}} - (T_{\text{desmoines}} - T_{\text{omaha}}) * 1 = T_{\text{desmoines}}$$

We can generalize this technique now to interpolate a value at some location, given known values at two known locations. If the known locations are numbered “1” and “3”, with the location of the unknown value numbered “2” (because “2” is between “1” and “3”), then the locations of these three places will be x_1 , x_2 , and x_3 . We know the values at locations “1” and “3”, which I’ll indicate at v_1 and v_3 . Therefore, we can solve for the unknown value, v_2 , by:

$$v_2 = v_1 + (v_3 - v_1) * (x_2 - x_1) / (x_3 - x_1)$$

With a little experimentation, you can show that it doesn’t matter whether location “1” is “west of” location “3” or “east of” location “3” or anything like that—as long as you are consistent about making sure that v_1 is the value at location “1” and v_2 is the value at location “3”, it’s all good.

Your Interpolation Function

Interpolation is hugely useful—we have several programs in this course that need to use interpolation. Therefore, I would recommend writing an interpolation function, making sure that it works appropriately, and then just keeping it around and using it over and over. As you might have guessed, such a function will have a *lot* of arguments—5 or 6, depending on exactly how you do this. On the other hand, the function itself is pretty short; in fact, you can probably do this in as few as 1 line inside of the curly braces of your interpolation function.

How you set up your interpolation function is completely up to you, as long as it works. I typically make my function work like this:

```
interp(v1, &v2, v3, x1, x2, x3);
```

where `interp` is a void function, and the value at location “2”, v_2 , is passed by address (or “pointer”). The only reason I tend to do it this way is that I find it easier to keep track of all of the argument if they are in order like this: 1-2-3 for v , 1-2-3 for x . However, in some ways it would be better to set up a function that worked like this:

```
v2 = interp(v1, v3, x1, x2, x3);
```

where now `interp` is a float function that returns the interpolated value v_2 . Either way is fine, as long as you get it to work.

Interpolating to Find a Location Between Two Known Locations

Another common way to use interpolation is to find the location of some value, given two known locations. For example, let’s think again about the problems of interpolating between Omaha and Des Moines. Suppose that you knew the temperatures

at Omaha and Des Moines, and now you want to solve for the *location* between Omaha and Des Moines at which the temperature is some value. For example, maybe the temperature in Omaha is 57°F and the temperature in Des Moines is 62°F; now you need to solve for the location where the temperature is 60°F. In other words, you want to know the location along the x axis where the temperature *is* your temperature, let's say $T_{\text{mytemperature}}$. (This is going to be important when we draw contour lines—clearly contouring is all about figuring out *where* the temperature is equal to the current value of the isotherm that you are drawing.)

Somewhat surprisingly, it works out that the same equation for interpolation will work to solve this problem, but the roles of the v and x variables are reversed:

$$x_{\text{mytemperature}} = x_{\text{omaha}} + (x_{\text{desmoines}} - x_{\text{omaha}}) * (T_{\text{mytemperature}} - T_{\text{omaha}}) / (T_{\text{desmoines}} - T_{\text{omaha}})$$

Even better, all that you need to do in a C program is reuse the interpolation code that you have already written, just reversing the x and v values:

```
interp (x1, &x2, x3, v1, v2, v3);
```

Interpolation vs. Extrapolation

Extrapolation is a lot like interpolation, and it uses the same exact equation. The only difference between the two techniques is that the unknown location is *not between* the two known locations. For example, given the temperatures at Omaha and Des Moines, you can *extrapolate* a temperature at Chicago. Use exactly the same function, and it will work just fine.

However, extrapolation is a considerably riskier proposition than interpolation is. When you extrapolate, there is a greater chance of serious error. You can read all about the risks of this error online.