Interpolation

Written by Dave Pawlowski

Introduction

One of the most common tasks faced by computational physicists, in particular when working with data, is interpolation. There are many different types of interpolation, but we'll just focus on a couple of those here. The goal of interpolation is to evaluate the values of some unknown function at points where we don't have any data. Say for example you had data on a grid:

x	$f(x)$
0.0	1.0
0.1	0.95
0.2	0.80
0.3	0.58
0.4	0.30
0.5	0.1
0.6	-0.31
0.7	-0.58
0.8	-0.80
0.9	-0.95
1.0	-1.0

Table 1: Data on a regularly spaced grid

Say, for example, we needed the data at the point $x = 0.36$; we have to interpolate.

1 Piecewise interpolation

Piecewise interpolation is incredibly simple. Say we wanted the value of $f(x)$ at $x = 0.36$ given the data in Table . In this case, we would simply use the value obtained at the nearest point. So, we would assign $f(x = 0.36) = f(x = 0.4) = 0.30$. As simple as this method is, it isn't particularly useful, and we could do significantly better by using linear interpolation, which doesn't turn out to be that much more work (computationally speaking).

2 Linear Interpolation

Linear interpolation is really the simplest, useful, method that we can employ. Linear interpolation approximates the function that we don't know as a straight line between neighboring points.

What this means is that if we want to determine a value for $f(x = 0.36)$, we would choose a value that is somewhere between 0.58 and 0.30, but slightly closer to 0.30. Formally, we can derive a straight forward algorithm for determining the value y, given $y_i, y_{i-1}, x, x_i, x_{i-1}$:

$$\frac{y - y_{i-1}}{y_i - y_{i-1}} = \frac{x - x_{i-1}}{x_i - x_{i-1}} \tag{1}$$

$$y - y_{i-1} = (y_i - y_{i-1})\frac{x - x_{i-1}}{x_i - x_{i-1}} \tag{2}$$

$$y = y_{i-1} + (y_i - y_{i-1})\frac{x - x_{i-1}}{x_i - x_{i-1}} \tag{3}$$

While relatively simple and not particularly computationally intensive, linear interpolation does suffer because it results in a function that is not continuously differentiable at our given grid points. This is not always and issue though. Also, in general, linear interpolation is not very precise.

3 Exponential Interpolation

In the situation where you know that the function, $f(x)$ behaves, at least to some degree, as an exponential function of x, then exponential interpolation is much more accurate than linear interpolation. That said, the two really utilize the same algorithm, in that in order to perform an exponential interpolation, we really perform a linear interpolation in log space:

$$y = exp[ln(y_{i-1}) + [ln(y_i) - ln(y_{i-1})]\frac{x - x_{i-1}}{x_i - x_{i-1}}] \tag{4}$$

The only difference between this equation and the previous one is that before doing the linear interpolation, we take the natural log of our ”y” points, then afterwards, we take the exponential.

4 Polynomial Interpolation

After linear interpolation, the logical next step to improve accuracy is to approximate the function a polynomial. In general, if you have n data points, there is one polynomial of order n-1 that passes through all of the data points.

Compared with linear interpolation, polynomial interpolation is much more computationally expensive. However, since a polynomail is being fit to the data, it is infintiely differentiable. On the other hand, polynomial interpolation can often exhibit oscillatory behavior (Runge’s phenomenon), as anyone that has used Excel to do such interpolation can attest to, especially when using polynomials of high degree.

5 Spline interpolation

Instead of fitting a polynomial that passes through all of the data points, we can limit ourselves to low order polynomials, which require less computational effort to determine. Spline interpolation fits low order polynomials to each of the data intervals and then chooses the polynomials such that they fit together nicely. The most common example, the cubic spline, ensures that the function is twice differentialble everywhere.