Cubic and bicubic interpolation

On this page you can find explanation about cubic and bicubic interpolation and a Java implementation of bicubic interpolation.
Anything at this page may be copied and modified.
The Wikipedia articles of cubic and bicubic interpolation may also be interesting.

Cubic interpolation

If the values of a function *f*(x) and its derivative are known at x=0 and x=1, then the function can be interpolated on the interval [0,1] using a third degree polynomial. This is called cubic interpolation. The formula of this polynomial can be easily derived.

A third degree polynomial and its derivative:

$$f(x) = ax^3 + bx^2 + cx + d$$
$$f'(x) = 3ax^2 + 2bx + c$$

The values of the polynomial and its derivative at x=0 and x=1:

$$f(0) = d$$
$$f(1) = a + b + c + d$$
$$f'(0) = c$$
$$f'(1) = 3a + 2b + c$$

The four equations above can be rewritten to this:

$$a = 2f(0) - 2f(1) + f'(0) + f'(1)$$
$$b = -3f(0) + 3f(1) - 2f'(0) - f'(1)$$
$$c = f'(0)$$
$$d = f(0)$$

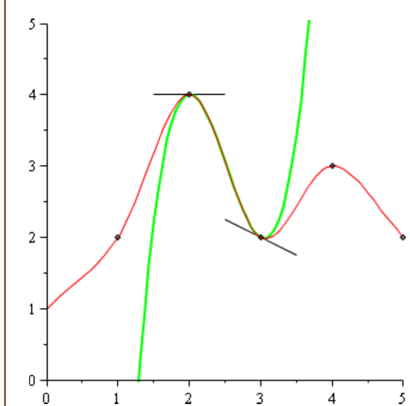
And there we have our cubic interpolation formula.

Interpolation is often used to interpolate between a list of values. In that case we don't know the derivative of the function. We could simply use derivative 0 at every point, but we obtain smoother curves when we use the slope of a line between the previous and the next point as the derivative at a point. Suppose you have the values p_0, p_1, p_2 and p_3 at respectively x=-1, x=0, x=1, and x=2. Then we can assign the values of *f*(0), *f*(1), *f'*(0) and *f'*(1) using the formulas below to interpolate between p_1 and p_2.

$$f(0) = p_1$$
$$f(1) = p_2$$
$$f'(0) = \frac{p_2 - p_0}{2}$$
$$f'(1) = \frac{p_3 - p_1}{2}$$

Combining the last four formulas and the preceding four, we get:

$$a = -\tfrac{1}{2}p_0 + \tfrac{3}{2}p_1 - \tfrac{3}{2}p_2 + \tfrac{1}{2}p_3$$
$$b = p_0 - \tfrac{5}{2}p_1 + 2p_2 - \tfrac{1}{2}p_3$$
$$c = -\tfrac{1}{2}p_0 + \tfrac{1}{2}p_2$$
$$d = p_1$$



For the green curve:

$$a = -\tfrac{1}{2}\cdot 2 + \tfrac{3}{2}\cdot 4 - \tfrac{3}{2}\cdot 2 + \tfrac{1}{2}\cdot 3 = \tfrac{7}{2}$$
$$b = 2 - \tfrac{5}{2}\cdot 4 + 2\cdot 2 - \tfrac{1}{2}\cdot 3 = -\tfrac{11}{2}$$
$$c = -\tfrac{1}{2}\cdot 2 + \tfrac{1}{2}\cdot 2 = 0$$
$$d = 4$$
$$f(x) = \tfrac{7}{2}(x-2)^3 - \tfrac{11}{2}(x-2)^2 + 4$$

Bicubic interpolation

Bicubic interpolation is cubic interpolation in two dimensions. I'll only consider the case where we want to interpolate a two dimensional grid. We can use the cubic interpolation formula to construct the bicubic interpolation formula. Here is the formula for cubic interpolation that we just derived, with a, b, c and d being the values of a function at respectively x=-1, x=0, x=1 and x=2:

$$f(x,a,b,c,d) = (-\tfrac{1}{2}a + \tfrac{3}{2}b - \tfrac{3}{2}c + \tfrac{1}{2}d)x^3 + (a - \tfrac{5}{2}b + 2c - \tfrac{1}{2}d)x^2 + (-\tfrac{1}{2}a + \tfrac{1}{2}c)x + b$$

Suppose we have the 16 points p_{ij}, with i and j going from 0 to 3 and with p_{ij} located at (i-1, j-1). Then we can interpolate the area [0,1] x [0,1] by first interpolating the four rows and then interpolating the results in the vertical direction. The formula becomes:

$$g(x,y) = f(y, f(x,p_{00},p_{10},p_{20},p_{30}), f(x,p_{01},p_{11},p_{21},p_{31}), f(x,p_{02},p_{12},p_{22},p_{32}), f(x,p_{03},p_{13},p_{23},p_{33}))$$

After much rewriting, done by mathematical software, we discover that we can write the formula this way:

$$g(x,y) = \sum_{i=0}^{3}\sum_{j=0}^{3} a_{ij}x^i y^j$$

With these values for a_{ij}:

$$a_{00} = p_{11}$$
$$a_{01} = -\tfrac{1}{2}p_{10} + \tfrac{1}{2}p_{12}$$
$$a_{02} = p_{10} - \tfrac{5}{2}p_{11} + 2p_{12} - \tfrac{1}{2}p_{13}$$
$$a_{03} = -\tfrac{1}{2}p_{10} + \tfrac{3}{2}p_{11} - \tfrac{3}{2}p_{12} + \tfrac{1}{2}p_{13}$$
$$a_{10} = -\tfrac{1}{2}p_{01} + \tfrac{1}{2}p_{21}$$
$$a_{11} = \tfrac{1}{4}p_{00} - \tfrac{1}{4}p_{02} - \tfrac{1}{4}p_{20} + \tfrac{1}{4}p_{22}$$
$$a_{12} = -\tfrac{1}{2}p_{00} + \tfrac{5}{4}p_{01} - p_{02} + \tfrac{1}{4}p_{03} + \tfrac{1}{2}p_{20} - \tfrac{5}{4}p_{21} + p_{22} - \tfrac{1}{4}p_{23}$$
$$a_{13} = \tfrac{1}{4}p_{00} - \tfrac{3}{4}p_{01} + \tfrac{3}{4}p_{02} - \tfrac{1}{4}p_{03} - \tfrac{1}{4}p_{20} + \tfrac{3}{4}p_{21} - \tfrac{3}{4}p_{22} + \tfrac{1}{4}p_{23}$$
$$a_{20} = p_{01} - \tfrac{5}{2}p_{11} + 2p_{21} - \tfrac{1}{2}p_{31}$$
$$a_{21} = -\tfrac{1}{2}p_{00} + \tfrac{1}{2}p_{02} + \tfrac{5}{4}p_{10} - \tfrac{5}{4}p_{12} - p_{20} + p_{22} + \tfrac{1}{4}p_{30} - \tfrac{1}{4}p_{32}$$
$$a_{22} = p_{00} - \tfrac{5}{2}p_{01} + 2p_{02} - \tfrac{1}{2}p_{03} - \tfrac{5}{2}p_{10} + \tfrac{25}{4}p_{11} - 5p_{12} + \tfrac{5}{4}p_{13} + 2p_{20} - 5p_{21} + 4p_{22} - p_{23} - \tfrac{1}{2}p_{30} + \tfrac{5}{4}p_{31} - p_{32} + \tfrac{1}{4}p_{33}$$
$$a_{23} = -\tfrac{1}{2}p_{00} + \tfrac{3}{2}p_{01} - \tfrac{3}{2}p_{02} + \tfrac{1}{2}p_{03} + \tfrac{5}{4}p_{10} - \tfrac{15}{4}p_{11} + \tfrac{15}{4}p_{12} - \tfrac{5}{4}p_{13} - p_{20} + 3p_{21} - 3p_{22} + p_{23} + \tfrac{1}{4}p_{30} - \tfrac{3}{4}p_{31} + \tfrac{3}{4}p_{32} - \tfrac{1}{4}p_{33}$$
$$a_{30} = -\tfrac{1}{2}p_{01} + \tfrac{3}{2}p_{11} - \tfrac{3}{2}p_{21} + \tfrac{1}{2}p_{31}$$
$$a_{31} = \tfrac{1}{4}p_{00} - \tfrac{1}{4}p_{02} - \tfrac{3}{4}p_{10} + \tfrac{3}{4}p_{12} + \tfrac{3}{4}p_{20} - \tfrac{3}{4}p_{22} - \tfrac{1}{4}p_{30} + \tfrac{1}{4}p_{32}$$
$$a_{32} = -\tfrac{1}{2}p_{00} + \tfrac{5}{4}p_{01} - p_{02} + \tfrac{1}{4}p_{03} + \tfrac{3}{2}p_{10} - \tfrac{15}{4}p_{11} + 3p_{12} - \tfrac{3}{4}p_{13} - \tfrac{3}{2}p_{20} + \tfrac{15}{4}p_{21} - 3p_{22} + \tfrac{3}{4}p_{23} + \tfrac{1}{2}p_{30} - \tfrac{5}{4}p_{31} + p_{32} - \tfrac{1}{4}p_{33}$$
$$a_{33} = \tfrac{1}{4}p_{00} - \tfrac{3}{4}p_{01} + \tfrac{3}{4}p_{02} - \tfrac{1}{4}p_{03} - \tfrac{3}{4}p_{10} + \tfrac{9}{4}p_{11} - \tfrac{9}{4}p_{12} + \tfrac{3}{4}p_{13} + \tfrac{3}{4}p_{20} - \tfrac{9}{4}p_{21} + \tfrac{9}{4}p_{22} - \tfrac{3}{4}p_{23} - \tfrac{1}{4}p_{30} + \tfrac{3}{4}p_{31} - \tfrac{3}{4}p_{32} + \tfrac{1}{4}p_{33}$$

In Java code we can write this as:

```java
/**
 * Interpolates the value of a point in a two dimensional surface using bicubic interpolation.
 * The value is calculated using the position of the point and the values of the 16 surrounding points.
 * Note that the returned value can be more or less than any of the values of the surrounding points.
 *
 * @param p A 4x4 array containing the values of the 16 surrounding points
 * @param x The horizontal distance between the point and the four points left of it, between 0 and 1
 * @param y The vertical distance between the point and the four points below it, between 0 and 1
 * @return the interpolated value
 */
public static double bicubicInterpolate (double[][] p, double x, double y) {
```

```
        double a00 = p[1][1];
        double a01 = -.5*p[1][0] + .5*p[1][2];
        double a02 = p[1][0] - 2.5*p[1][1] + 2*p[1][2] - .5*p[1][3];
        double a03 = -.5*p[1][0] + 1.5*p[1][1] - 1.5*p[1][2] + .5*p[1][3];
        double a10 = -.5*p[0][1] + .5*p[2][1];
        double a11 = .25*p[0][0] - .25*p[0][2] - .25*p[2][0] + .25*p[2][2];
        double a12 = -.5*p[0][0] + 1.25*p[0][1] - p[0][2] + .25*p[0][3] + .5*p[2][0] - 1.25*p[2][1] + p[2][2] - .25*p[2][3];
        double a13 = .25*p[0][0] - .75*p[0][1] + .75*p[0][2] - .25*p[0][3] - .25*p[2][0] + .75*p[2][1] - .75*p[2][2] + .25*p[2][3];
        double a20 = p[0][1] - 2.5*p[1][1] + 2*p[2][1] - .5*p[3][1];
        double a21 = -.5*p[0][0] + .5*p[0][2] + 1.25*p[1][0] - 1.25*p[1][2] - p[2][0] + p[2][2] + .25*p[3][0] - .25*p[3][2];
        double a22 = p[0][0] - 2.5*p[0][1] + 2*p[0][2] - .5*p[0][3] - 2.5*p[1][0] + 6.25*p[1][1] - 5*p[1][2] + 1.25*p[1][3] + 2*p[2][0] - 5*p[2][1] + 4*p[2][2] -
        double a23 = -.5*p[0][0] + 1.5*p[0][1] - 1.5*p[0][2] + .5*p[0][3] + 1.25*p[1][0] - 3.75*p[1][1] + 3.75*p[1][2] - 1.25*p[1][3] - p[2][0] + 3*p[2][1] - 3*p
        double a30 = -.5*p[0][1] + 1.5*p[1][1] - 1.5*p[2][1] + .5*p[3][1];
        double a31 = .25*p[0][0] - .25*p[0][2] - .75*p[1][0] + .75*p[1][2] + .75*p[2][0] - .75*p[2][2] - .25*p[3][0] + .25*p[3][2];
        double a32 = -.5*p[0][0] + 1.25*p[0][1] - p[0][2] + .25*p[0][3] + 1.5*p[1][0] - 3.75*p[1][1] + 3*p[1][2] - .75*p[1][3] - 1.5*p[2][0] + 3.75*p[2][1] - 3*p
        double a33 = .25*p[0][0] - .75*p[0][1] + .75*p[0][2] - .25*p[0][3] - .75*p[1][0] + 2.25*p[1][1] - 2.25*p[1][2] + .75*p[1][3] + .75*p[2][0] - 2.25*p[2][1

        double x2 = x * x;
        double x3 = x2 * x;
        double y2 = y * y;
        double y3 = y2 * y;

        return a00 + a01 * y + a02 * y2 + a03 * y3 +
               a10 * x + a11 * x * y + a12 * x * y2 + a13 * x * y3 +
               a20 * x2 + a21 * x2 * y + a22 * x2 * y2 + a23 * x2 * y3 +
               a30 * x3 + a31 * x3 * y + a32 * x3 * y2 + a33 * x3 * y3;
}
```

This code changed at 22 Februari and 28 April 2010, thanks to Dan Walmsley and Laura Spoldi for notifying me about mistakes I made.