Parabola Volume 47, Issue 3 (2011)

Large Splines

Bill McKee [1]

This article is fundamentally about the calculations behind the ways in which computers draw graphs. In the era before computers (unknown to most of you, but very familiar to me!) graphs were drawn on paper. Typically, the data points were plotted. Next, pins were placed in the paper at these data points and a thin flexible piece of wood was threaded around these points to produce a nice smooth shape which was then traced by hand. This piece of wood was known as a *spline*. The word itself apparently derives from a dialect word from the East Anglia region of England for a strip of wood and is related to the word *splinter*. This article will present a computational method which approximates the behaviour of a spline. This method, and generalisations thereof, underlie much of computer graphics.

An earlier article in *Parabola Incorporating Function* (Volume 44, Number 3) showed how to construct a polynomial which passed exactly through some data points. Splines provide a different way of constructing a smooth curve which also passes exactly through the given data points and avoids some of the pitfalls which can be sometimes associated with the earlier procedure.

The Problem of Interpolation

Suppose that we have $n+1$ data points (x_i, y_i) for $i = 0, 1, \ldots, n$ where the x_i are all different. The x_i need not be equally spaced, but are assumed here to be in increasing order. The *interpolation problem* is one of finding a function which passes exactly through the given data points. There are an infinite number of such functions but we obviously aim to find a simple interpolating function which is a good approximation to the underlying function of which the data points are point values. The article mentioned above showed how to construct a polynomial of degree n which passed exactly through all of these points. It also showed that there was only one such polynomial of degree n. As pointed out in the earlier article, it would probably be better to refer to this polynomial as having degree at most n since, for example, three points might just happen to lie on a straight line rather than a parabola. So, when we refer to the interpolating polynomial as being of degree n, it is to be understood that the degree is at most n. One of the main reasons for interpolation is to estimate the value of quantities between the given data points. For example, we might have temperature measurements taken every hour and want to estimate the temperature at some intermediate time. Often the use of this interpolating polynomial gives quite satisfactory results. This is demonstrated in Figure 1 for the sine function. The x points are $x_0 = 0$, $x_1 = 0.5$,

[1]Dr Bill McKee is a Visiting Fellow in the School of Mathematics and Statistics at the University of New South Wales.

$x_2 = 1.0, \ldots, x_{10} = 5$ and the y points are $y_i = \sin\ x_i$ for $i = 0, 1, \ldots, 10$. (Remember that x is measured in radians not degrees). The sine function itself is not shown since it is virtually indistinguishable from the interpolating polynomial of degree 10 at the scale depicted.

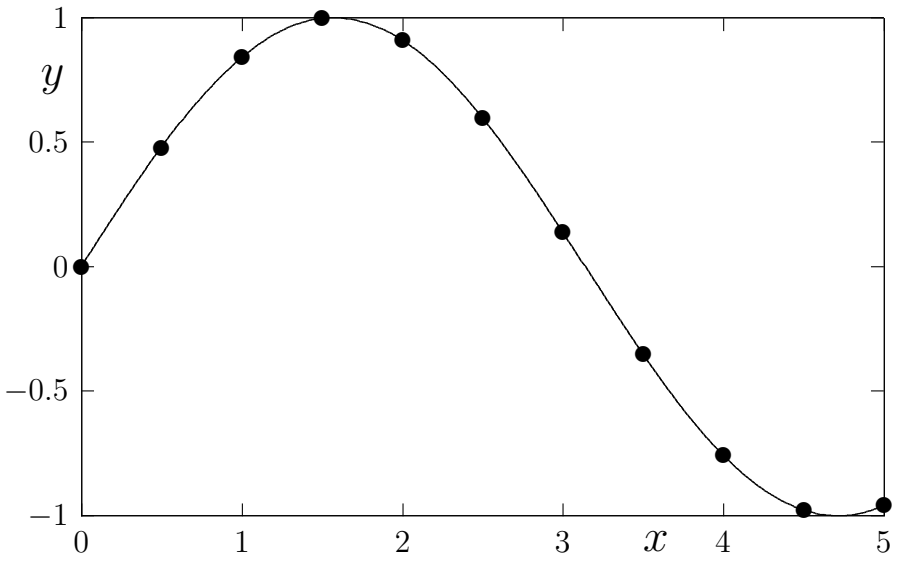

Figure 1: The dots represent the data points for the sine function. The line is the interpolating polynomial of degree 10 passing exactly through these points.

However, things are not always so rosy and polynomial interpolation can sometimes go spectacularly wrong. The basic reason for this is that if we have a large number of data points the interpolating polynomial will be of high degree. Now a polynomial of degree n can have as many as $n - 1$ local maxima and minima; that is to say it may be quite 'wiggly'. Some of these wiggles may lie between x_0 and x_n which can sometimes lead to inaccurate results. For example, consider the simple function

$$f(x) = \frac{1}{1 + x^2} \tag{0.1}$$

and suppose we choose our x points to be $x_0 = -3$, $x_1 = -2.5$, $x_2 = -2$, $\ldots$, $x_{12} = 3$ and the corresponding y values to be $y_i = f(x_i)$ for $i = 0, 1, \ldots, 12$. The interpolating polynomial will be of degree 12 and is shown in Figure 2 as a thin solid line along with the data points shown by dots and $f(x)$ shown by a thick solid line. As you can see, the interpolating polynomial is not providing a good approximation to the original function from which the data points were obtained, particularly closer to the end points of the region. We might try to remedy this by using more points. For example, in Figure 3 we use 19 equally spaced points rather than 13. The problem is getting worse!

If all we know are the data points then any interpolating function will satisfy the requirement of passing through the data points but may not seem reasonable to us.

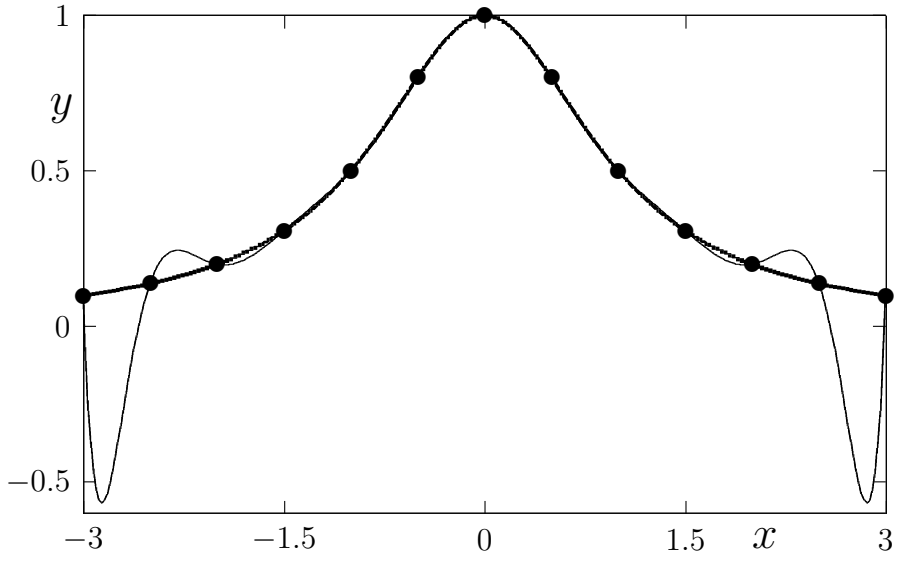

Figure 2: The dots represent the data points. The thin line is the interpolating polynomial of degree 12 passing exactly through these points and the thick line is the function defined by equation (0.1).

Suppose, for example, that the data points in Figure 2 represented hourly temperature measurements. It is extremely unlikely that the real, but unmeasured, temperature exhibited such huge fluctuations away from the data points and any sensible meteorologist would reject this graph as a useful tool for estimating the temperature at times between data points. Another problem which may arise is that the form of the interpolating polynomial could be quite sensitive to small changes in the data. For example, suppose we had three points which happened to lie in a straight line. Altering one of the points slightly would turn that straight line into a parabola.

The Concept of a Spline

We now abandon the idea of using a single polynomial to interpolate the data for all x between x_0 and x_n. Instead we use a different and simpler polynomial between each of the data points and then match them up somehow at the data points. Indeed, you will already be familiar with the technique of connecting adjacent data points by straight lines, as shown in Figure 4. This is sometimes called a *linear spline*. In this example, the use of a linear spline has avoided the huge errors associated with the high-degree interpolating polynomial but is not a particularly accurate approximation to $f(x)$ and has a discontinuous slope at each of the data points. We could try using parabolic arcs rather than straight lines (this gives rise to *quadratic splines*) but it turns out that it is better to use cubics rather than quadratics which gives us the *cubic splines* to which we will turn our attention after first briefly discussing linear splines.

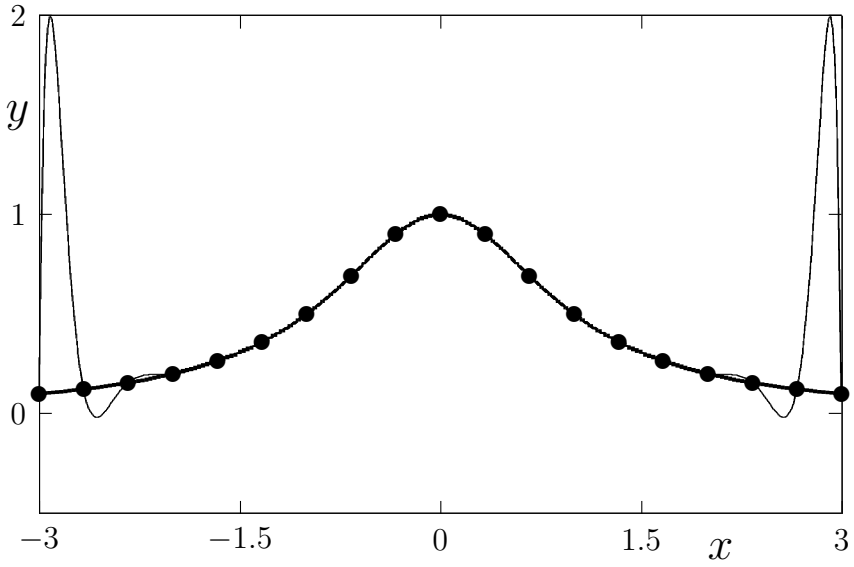

Figure 3: As in Figure 2 except that we have used 19 equally-spaced points which means that the interpolating polynomial is of degree 18.

Linear Splines

The idea here is to use straight lines to interpolate our data. Thus between x_j and x_{j+1} for $j = 0, 1, \ldots, n-1$ we define our linear spline interpolant to be

$$S_j(x) = \lambda_j + \mu_j x$$

upon which we impose the conditions $S_j(x_j) = y_j$ and $S_j(x_{j+1}) = y_{j+1}$. As you know, the equation of the straight line passing through the two given points can be written as

$$y = S_j(x) = y_j + \frac{(y_{j+1} - y_j)}{(x_{j+1} - x_j)}(x - x_j) \tag{0.2}$$

from which λ_j and μ_j could be written down if desired.

Cubic Splines

As outlined above, we now use cubics rather than straight lines to interpolate our data. Thus between x_j and x_{j+1} for $j = 0, 1, \ldots, n-1$ we define our cubic spline interpolant to be

$$S_j(x) = \alpha_j + \beta_j x + \gamma_j x^2 + \delta_j x^3. \tag{0.3}$$

It is important to note that the coefficients α_j, β_j, γ_j and δ_j vary from one interval to the next since we are using different cubics in each of the intervals.

Guided by equation (0.2) we re-write equation (0.3) as

$$S_j(x) = a_j + b_j(x - x_j) + c_j(x - x_j)^2 + d_j(x - x_j)^3. \tag{0.4}$$

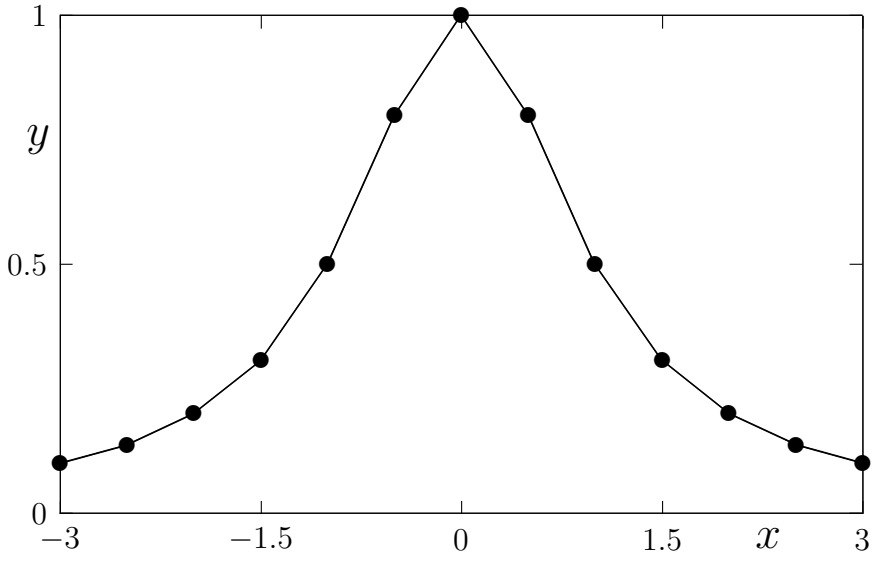

Figure 4: As in Figure 2 except that we have connected adjacent points by straight lines and thus constructed a *linear spline*. The function defined by equation (0.1) is not shown.

What we are doing here is referring the spline to the left-hand endpoint of the interval over which it is defined, rather than to $x = 0$. This is sensible because equation (0.4) only applies for $x_j \leq x \leq x_{j+1}$. Notice also that setting $c_j = d_j = 0$ gives our linear spline equation (0.2) with appropriate choice of a_j and b_j.

There are four unknowns in (0.4) and hence in total there are $4n$ unknowns and hence we will need $4n$ conditions from which to determine them. Clearly we want our cubic to agree with the data at each endpoint. In contrast with the case of linear splines we can also require that the slope (i.e. the first derivative) of the spline in one interval at the endpoint matches with that of the spline in the next interval at that point. Clearly, we can only do this at the interior points and not at x_0 or x_n. We can also impose a similar condition on the second derivatives at the interior data points. These two conditions together mean that the curvatures of the spline also match at these points. This gives rise to a nice smooth curve. Thus, the requirement that the spline passes exactly through all the data points requires:

$$S_j(x_j) = y_j \quad \text{for} \quad j = 0, 1, \ldots, n-1 \tag{0.5}$$

$$S_j(x_{j+1}) = y_{j+1} \quad \text{for} \quad j = 0, 1, \ldots, n-1 \tag{0.6}$$

Similarly, the requirement that the first and second derivatives of the spline segments match at each of the interior points requires:

$$S_j'(x_{j+1}) = S_{j+1}'(x_{j+1}) \quad \text{for} \quad j = 0, 1, \ldots, n-2 \tag{0.7}$$

$$S_j''(x_{j+1}) = S_{j+1}''(x_{j+1}) \quad \text{for} \quad j = 0, 1, \ldots, n-2 \tag{0.8}$$

where $'$ denotes one differentiation with respect to x and hence $''$ denotes two differentiations with respect to x.

So far, we have $4n-2$ conditions imposed and $4n$ coefficients to be found. Two more conditions are needed. Many conditions are possible but two of the most most commonly used conditions are:

1. To specify the derivative at x_0 and x_n or
2. To require that the second derivative is zero at x_0 and at x_n.

The first of these would generally be used if we were trying to interpolate a known function such as the one defined by (0.1). The required derivative values are then known numbers. This produces what is known as a *clamped spline*. The second is generally used to interpolate data such as hourly temperature measurements. Physically, it is saying that the interpolation function has zero curvature (i.e. is straight) at the two endpoints. This corresponds to the situation which existed at the endpoints when a wooden strip was passed through the data points and protruded unconstrained beyond the first and last points. It produces what is known as a *natural cubic spline*.

In passing, let us see what would happen if we tried to use quadratics rather than cubics for our spline. This would be accomplished by setting $d_j = 0$ in equation (0.4) leaving now only $3n$ coefficients to be found. Equations (0.5) and (0.6) together provide $2n$ conditions while equation (0.7) provides another $n-1$ which makes $3n-1$ in total leaving us with only one more condition to impose. Since there are two end points we can impose a condition at one of them only. This is unsatisfactory since one end point is being treated differently from the other. This is the basic problem with attempts to use quadratic splines. There are ways around this difficulty using a slightly different approach but cubic splines generally give smoother curves anyway and have some nice mathematical properties which are too complicated to discuss here.

Before treating the general case for a cubic spline let us look at a simple example.

Only two points

Suppose we have only two points (x_0, y_0) and (x_1, y_1). Our cubic spline would then be

$$S_0(x) = a_0 + b_0(x-x_0) + c_0(x-x_0)^2 + d_0(x-x_0)^3. \tag{0.9}$$

From this we see that

$$S_0'(x) = b_0 + 2c_0(x-x_0) + 3d_0(x-x_0)^2$$

and

$$S_0''(x) = 2c_0 + 6d_0(x-x_0)$$

The condition that $S_0(x_0) = y_0$ thus requires that $a_0 = y_0$ and this plus the condition that $S_0(x_1) = y_1$ requires

$$y_1 = y_0 + b_0h_0 + c_0h_0^2 + d_0h_0^3 \tag{0.10}$$

where $h_0 = x_1 - x_0$ is the spacing between the two points. Here there are no interior points and we have only the two end points at which to impose conditions to determine c_0 and d_0. It is an easy matter to check that imposing the condition that the second

derivative vanish at x_0 and at x_1 leads to $c_0 = d_0 = 0$ which means, not surprisingly, that we are back to our linear spline. However, now let us suppose that instead we imposed the conditions

$$S_0'(x_0) = K_0 \quad \text{and} \quad S_0'(x_1) = K_1$$

in order to find a clamped spline. Then

$$K_0 = b_0 \quad \text{and} \quad K_1 = K_0 + 2c_0h_0 + 3d_0h_0^2$$

thus using equation (0.10) we see that we have to solve

$$\begin{aligned} c_0h_0^2 + d_0h_0^3 &= y_1 - y_0 - K_0h_0 && (0.11)\\ 2c_0h_0 + 3d_0h_0^2 &= K_1 - K_0 && (0.12) \end{aligned}$$

for the only two remaining unknowns c_0 and d_0. You probably remember how to do this but, in case you have forgotten, one way would be as follows. We multiply the first by $2/h_0$. The coefficient of c_0 in both equations is then $2h_0$. So, subtracting the two equations gives a new equation in which d_0 is the only unknown and so is readily solved. This value of d_0 is then substituted back into either of the original equations which thus becomes an equation in which c_0 is the only unknown and so is easily solved for c_0. You should try this for yourself and should obtain the answers

$$c_0 = \frac{3(y_1 - y_0) - h_0(K_1 + 2K_0)}{h_0^2} \qquad (0.13)$$

$$d_0 = \frac{h_0(K_1 + K_0) - 2(y_1 - y_0)}{h_0^3} \qquad (0.14)$$

As an example, consider the function defined by equation (0.1). For this function

$$f'(x) = \frac{-2x}{(1+x^2)^2}$$

Let us take $x_0 = 0$ and $x_1 = 1$. Then $h_0 = 1$, $y_0 = f(0) = 1$ and $y_1 = f(1) = 0.5$. Furthermore, $K_0 = f'(0) = 0$ and $K_1 = f'(1) = -0.5$. Using the above formulæ we find $a_0 = 1$, $b_0 = 0$, $c_0 = -1$ and $d_0 = 0.5$. Hence our spline is

$$S_0(x) = 1 - x^2 + \frac{x^3}{2}$$

which is plotted in Figure 5. As you can see, this is giving a much better approximation to the original function than a linear spline would do. This is hardly surprising since it incorporates four pieces of information rather than just two.

The General Case

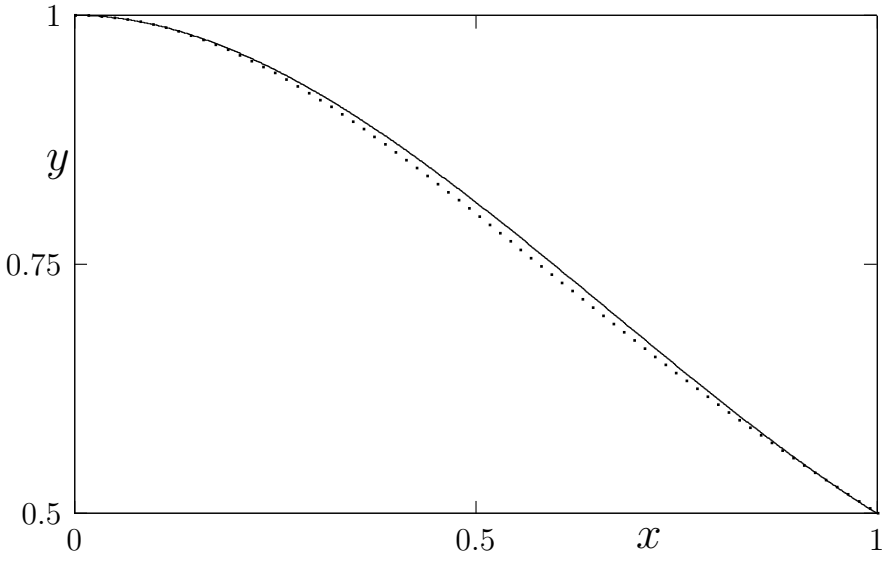

Figure 5: The solid line is the simple cubic spline passing exactly through the two end points and having the correct slope at these points while the dotted line is the function defined by equation (0.1).

Let us now return to the general case. The details are quite messy and you may prefer to skip over this section. If you do so, then it suffices for you to know that we can solve the above equations for the $4n$ coefficients a_j, b_j, c_j and d_j for $j = 0, 1, \ldots, n-1$.

Anyway, it is convenient to define:

$$h_j = x_{j+1} - x_j \quad \text{for} \quad j = 0, 1, \ldots, n-1$$

and

$$a_n = y_n$$

This definition plus equation (0.5) requires that

$$a_j = y_j \quad \text{for} \quad j = 0, 1, \ldots, n \tag{0.15}$$

while this plus equation (0.6) requires that

$$y_{j+1} = a_{j+1} = a_j + b_j h_j + c_j h_j^2 + d_j h_j^3 \quad \text{for} \quad j = 0, 1, \ldots, n-1 \tag{0.16}$$

Let us now match the derivatives at the interior points and introduce the definition $b_n = S'_{n-1}(x_n)$. Noting that

$$S'_j(x) = b_j + 2c_j(x - x_j) + 3d_j(x - x_j)^2$$

and using equation (0.7) we can see that

$$b_{j+1} = b_j + 2c_j h_j + 3d_j h_j^2 \quad \text{for} \quad j = 0, 1, \ldots, n-1 \tag{0.17}$$

Finally, let us now match the second derivatives at the interior points and introduce the definition $c_n = S''_{n-1}(x_n)/2$. Noting that

$$S''_j(x) = 2c_j + 6d_j(x - x_j)$$

and using equation (0.8) we can see that

$$c_{j+1} = c_j + 3d_j h_j \quad \text{for} \quad j = 0, 1, \ldots, n-1 \tag{0.18}$$

In the case of a natural cubic spline, which is the only one we shall consider here, the condition $S''_0(x_0) = 0$ requires that $c_0 = 0$ while the condition $S''_{n-1}(x_n) = 0$ requires that $c_n = 0$. (In the case of a clamped cubic spline these are replaced by conditions which I leave it to you to write down.)

We now systematically eliminate all variables in favour of the c_j. The result is a system of $n-1$ simultaneous linear equations for the $n-1$ coefficients $c_1, c_2, \ldots, c_{n-1}$. For $j = 1, \ldots, n-1$, the jth equation is

$$h_{j-1}c_{j-1} + 2(h_{j-1} + h_j)c_j + h_j c_{j+1} = 3(y_{j+1} - y_j)/h_j - 3(y_j - y_{j-1})/h_{j-1}$$

which obviously simplifies if the h_j are all equal, i.e. if the x_j are equally spaced. The system is solved by a systematic elimination procedure which is a generalisation of the method we used to solve equations (0.11) and (0.12). Don't worry too much about the technical details. Once the c_j are known, we can find the d_j from equation (0.18) and hence the b_j from (0.17). The a_j are already known from (0.15).

An Example

As an example, let us consider the data in Table 1 which has been generated in an approximately random manner. The x_i are not equally-spaced. To fix ideas, we might think of the x_i as time measured in seconds and the y_i as measurements of some quantity such as the temperature or pH of a solution undergoing a chemical reaction.

x_i	2.3	2.9	3.4	4.0	5.0	7.7	8.5	8.9	9.2	9.5
y_i	4.6	2.2	1.5	8.0	6.5	0.3	5.9	4.2	4.4	5.0

Table 1: A typical data set.

Figure 6 shows these data points together with the interpolating polynomial of degree 9 plus the natural cubic spline. The point of trying to interpolate the data here would be to estimate the temperature or pH at times between observations. Clearly, the natural cubic spline gives a much more credible result than the interpolating polynomial of degree 9.

In passing, let it be noted that if we were to use a natural cubic spline rather than a high-order interpolating polynomial as in Figure 2 or Figure 3 then the spline would be graphically indistinguishable from the original function at the scale depicted. For this reason, we will not present the graphs here.

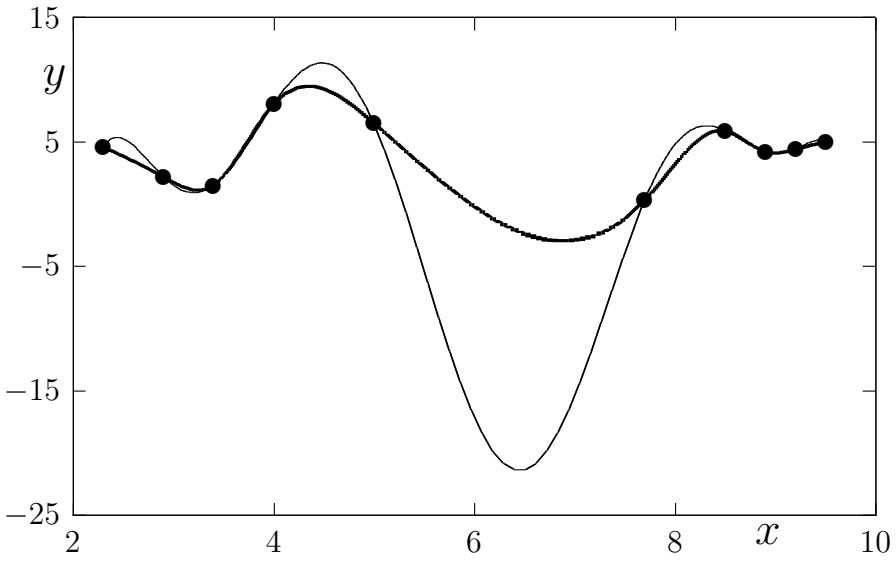

Figure 6: The thick line is the natural cubic spline passing exactly through the data points in Table 1 while the thin line is the interpolating polynomial of degree 9 passing exactly through these same points.

Discussion

We have seen that the use of cubic splines can avoid some of the overshooting sometimes associated with interpolating polynomials of high degree. They are widely used in industrial design and computer graphics to produce nice smooth curves. It is relatively easy to modify the procedure described above to handle cases where the curve being described has sharp interior corners. Rather than requiring that the first and second derivatives be continuous at that point, we specify the derivatives at either side of this point. It can also be generalised to use more general functions not just cubics. The ideas introduced here can be extended to two dimensions to plot surfaces. Virtually all computer graphics packages and applications use splines in one form or another.

As you might imagine, there is a large amount of material on the web about splines. You might like to start at

`http://en.wikipedia.org/wiki/Spline_(mathematics)`

and follow some of the links there.