Interpolation*

Jonathan Goodman

March 3, 1999

1 Introduction

Interpolation is the process of estimating values of $f(x)$ from certain given values. Suppose we have points x_0, x_1, $\cdots$, x_n and known function values $f_0 = f(x_0)$, $f_1 = f(x_1)$, ..., $f_n = f(x_n)$. From this data, we want to find an approximating function[1], $\hat{f}(x)$, defined for x not one of the x_k. The hope is that if f and $\hat{f}$ are smooth and if $f(x_k) = \hat{f}(x_k)$, then $\hat{f}(x)$ will be close to $f(x)$ for other x values. Interpolation, then, is the process of finding a smooth function, $\hat{f}(x)$, satisfying the interpolation conditions $\hat{f}(x_k) = f_k$ for $k = 0$, ..., n.

We will discuss two good forms of interpolation, low order polynomial interpolation and spline interpolation. We will also discuss a possibly bad interpolation method, high order polynomial interpolation. The theoretical explination of the interpolation process uses ideas that are central to much of numerical analysis, the notions of *consistency* and *stability*. Most often, a computational method is useful only if it is both consistent and stable. All the forms of interpolation we will look at are consistent, but high order polynomial interpolation, as we will see, may be unstable.

Interpolation has many uses in scientific computing, beyond the obvious use of estimating specific function values. Many methods for integrating or solving ordinary differential equations make use of interpolating polynomials or other interpolating functions. Interpolation is another way to estimate derivatives of a function: use the derivative of the interpolating function. The list of known applications is much longer than this.

2 Basics of polynomial interpolation.

We begin by discussing the algebraic aspects of polynomial interpolation, the existence and uniqueness of interpolating polynomials. This will give us ways to construct the interpolating polynomial but will not tell whether this is a good idea. Later sections will explore the accuracy of polynomial interpolation.

Now, suppose we have $n+1$ points, x_0, x_1, ..., x_n, and $n+1$ function values, f_0, f_1, ..., f_n. We want a polynomial, $p_n(x)$, of degree n, that takes the specified values at the specified points:

$$p_n(x_k) = f_k \quad \text{for } k = 0, \ldots, k = n. \tag{1}$$

From now on, we suppose that the interpolating points are distinct, that is $x_j \neq x_k$ if $j \neq k$. In that case there is an interpolating polynomial. Furthermore, it is unique; every way to construct a solution leads to the same polynomial.

The construction of the interpolating polynomial most used in scientific computing is the Newton construction. This construction builds a sequence of polynomials, each interpolating one extra point. The first

*These are course notes for Scientific Computing, given Spring 1999 at the Courant Institute of Mathematical Sciences at New York University. Professor Goodman retains the copyright to these notes.

[1] Statisticians often write $\hat{q}$ for a statistical estimate of the unknown quantity, q. I am following a similar convention in writing $\hat{f}$ for an estimate of the unknown quantity $f(x)$, even though it is not a statistical estimate.

is $p_0(x)$, which has degree zero – that is, is a constant. We choose this constant so that p_0 satisfies the first interpolating condition (1) for $k = 0$:

$$p_0 = f_0 \quad .$$

Next, we find a polynomial, $p_1(x)$, of degree 1 that interpolates both f_0 and f_1. We choose $p_1(x)$ of the form

$$p_1(x) = p_0(x) + (x - x_0) \cdot c_1$$

so that the previous interpolation condition at x_0 is not disturbed, that is[2] $p_1(x_0) = p_0(x_0) = f_0$. Then we use the remaining interpolation condition, $p_1(x_1) = f_1$, to solve for c_1:

$$f_1 = f_0 + (x_1 - x_0) \cdot c_1 \quad ,$$

$$c_1 = \frac{f_1 - f_0}{x_1 - x_0} \quad , \tag{2}$$

$$p_1(x) = f_0 + \frac{f_1 - f_0}{x_1 - x_0}(x - x_0) \quad . \tag{3}$$

The formula (2) shows that the coefficient c_1 is a divided difference. The construction of p_2 will make the genereral pattern clear. First, choose a form for $p_2(x)$ that insures that the interpolation conditions at x_0 and x_1 are preserved:

$$p_2(x) = p_1(x) + (x - x_1)(x - x_0)c_2 \quad .$$

Then choose c_2 to respect the final interpolation condition $p_2(x_2) = f_2$:

$$c_2 = \frac{\dfrac{f_2 - f_0}{x_2 - x_0} - \dfrac{f_1 - f_0}{x_1 - x_0}}{x_2 - x_1} \quad .$$

The general construction proceeds by induction. If we have a polynomial of degree $j - 1$ that satisfies the interpolation conditions (1) for $k < j$, then we look for $p_k(x)$ of the form

$$p_j(x) = p_{j-1}(x) + (x - x_{j-1}) \cdots (x - x_0)c_j \quad , \tag{4}$$

and use the interpolation condition $p_j(x_j) = f_j$ to solve for c_j. If $p_{j-1}(x)$ has degree $j - 1$, then (4) gives p_j degree j because the second term has j factors involving x. If we continue in this way, eventually we construct $p_n(x)$ that satisfies all the conditions (1). This is essentially Newton's construction of the interpolating polynomial.

Another way to construct the interpolating polynomial is through the Lagrange interpolatin formula. For this purpose we use specific polynomials

$$m_k(x) = \prod_{j \neq k,\ j=0}^{n} (x - x_j) \quad , \tag{5}$$

$$l_k(x) = m_k(x)/m_k(x_k) \quad . \tag{6}$$

The denominator in (6) is not zero because the points x_j are never equal to x_k if $j \neq k$. The polynomials $m_k(x)$ and $l_k(x)$ have degree n because the products in (5) have n terms. Moreover, $m_k(x_j) = l_k(x_j) = 0$ if $j \neq k$. This is the purpose of the formula (5). The normalization (6) makes $l_k(x_k) = 1$. We express the interpolating polynomial $p_n(x)$ in terms of the $l_k(x)$ simply as

$$p_n(x) = \sum_{k=0}^{n} f_k l_k(x) \quad . \tag{7}$$

[2] Of course, p_0 does not actually depend on x. I hope there is no harm in pretending that it does. It makes the construction of $p_1(x)$ look more like the general case.

If x is one of the interpolating points, say $x = x_m$, then only the term on the right side of (7) with $k = m$ is different from zero. Thus, (7) satisfies the interpolation conditions (1). The formulae (1), (2), and (3) together are called the *Lagrange interpolation formula.*

The third approach we will discuss is the least elegant but the most general. It is a method of last resort when elegant solutions are not available[3]. We simply pose the interpolation problem as a system of linear equations and use general theorems from linear algebra. The fact that the interpolating polynomial is unique falls out immediately as a consequence of what we already know.

Write the polynomial $p_n(x)$ in terms of its coefficients:

$$p_n(x) = a_0 + a_1 x + a_2 x^2 + \cdots + a_n x^n \quad .$$

The interpolation conditions (1) then take the form

$$\begin{aligned} a_0 + a_1 x_0 + \cdots + a_n x_0^n &= f_0 \\ a_0 + a_1 x_1 + \cdots + a_n x_1^n &= f_1 \\ &\vdots \\ a_0 + a_1 x_n + \cdots + a_n x_n^n &= f_n \quad . \end{aligned}$$

This system may be written in matrix form as

$$Va = f \quad , \tag{8}$$

where

$$V = \begin{pmatrix} 1 & x_0 & . & . & . & x_0^n \\ 1 & x_1 & . & . & . & x_1^n \\ . & . & . & . & . & . \\ . & . & . & . & . & . \\ . & . & . & . & . & . \\ 1 & x_n & . & . & . & x_n^n \end{pmatrix} \quad ,$$

$$a = \begin{pmatrix} a_0 \\ a_1 \\ . \\ . \\ . \\ a_n \end{pmatrix} \quad , \quad f = \begin{pmatrix} f_0 \\ f_1 \\ . \\ . \\ . \\ f_n \end{pmatrix} \quad .$$

The matrix V is the *vander Monde* matrix. It is $(n+1)\times(n+1)$. This is just a reformulation of the polynomial interpolatin problem. Interpolating polynomials exist and are unique if the soltuion of the system of linear equations (8) exists and is unique.

Now we can use linear algebra. Since V is a square matrix, there are only two possibilities for (8). Either there is a unique solution for every f, or there is some f with no solution at all. The second possibility cannot happen for our interpolation problem, because we already know that there is some interpolating polynomial for any data, f. This is from the Newton or Lagrange construction. The insight of linear algebra is that because the equations are linear and there are the right number of equations ($n+1$ equations, one for each interpolation condition) and unknowns ($n+1$ coefficients in the degree n interpolating polynomial), we know the polynomial is unique if it always exists.

By the way, the vander Monde matrix generally is very badly conditioned. This makes the linear algebra approach to interpolation the worst of the three from a computational point of view.

[3]multidimensional interpolation is often done in this way.

3 Accuracy of low order polynomial interpolation

Suppose we have points x_0, x_1, ..., x_n, corresponding values f_0, ..., f_n, and a point x which is not one of the given points. For this analysis, we make use of the underlying but unknown function, $f(x)$, whose values are the f_k. Assume that the points are distinct and ordered: $x_0 < x_1 < \cdots < x_n$. Polynomial interpolation of degree $p \leq n$ would select a *stencil* of $p+1$ points x_k, x_{k+1}, ..., x_{k+p}, and define $\hat{f}(x)$ to be the value of the polynomial (at x) that interpolates the values f_j at the stencil points x_j for $j = k$, ..., $j = k+p$. The stencil may be different for different x and may be chosen in any way. The analysis will assume that x is inside the stencil, that is $x_k < x < x_{k+p}$, although this is not strictly necessary in every case[4]. In order to simplify notations, I will take $k = 0$ throughout the analysis below.

Common examples are degree $p = 1$ interpolation, which is linear interpolation. The Newton interpolation formula for linear interpolation is

$$\hat{f}(x) = f_k + \frac{f_{k+1} - f_k}{x_{k+1} - x_k}(x - x_k) \quad .$$

Degree $p = 2$ interpolation uses a three point stencil and quadratic interpolating polynomial. The stencil must be asymmetric in the sense that x is either in the left "interpolation panel" ($x_0 < x < x_1$) or x is in the right panel ($x_1 < x < x_2$). Local cubic interpolation can be symmetric in that x can be in the central panel. Please remember to call this "degree p interpolation" rather than "order p interpolation". We will see that "degree p interpolation" has accuracy of order $p+1$. Linear interpolation is second order accurate, and so on.

I will present two ways to understand the accuracy of local polynomial interpolation. The first is much simpler but "too clever by half". The second is clumsy but it illustrates one of the most significant mathematical ideas in scientific computing, *stability*. Many (though not all, in my survey) experienced numerical analysists find the second form of the analysis more natural.

The first analysis has two parts, each of which is a repeated application of the mean value theorem. Suppose that $u(x)$ is a differentiable function with a continuous derivative in the interval $[a, b]$. The mean (or average) value of the derivative, $u'(x)$, in the interval is

$$\frac{1}{b-a}\int_a^b u'(x)dx = \frac{u(b) - u(a)}{b-a} \quad .$$

The mean value theorem states that there is some $x_* \in (a, b)$ where u' is equal to its mean value:

$$u'(x_*) = \frac{u(b) - u(a)}{b-a} \quad \text{for some } x \in (a, b).$$

If this were not true, then u' would have to be larger than its average value (or smaller) thoughout the interval, which is impossible.

The first stage of the first argument shows that interpolating f automatically leads to interpolating derivatives f', f'', and so one up to the degree of the interpolating polynomial $\hat{f}$. We do not learn exactly where these interpolating points are, only that they "interlace" the original interpolating points.

Let $g(x) = \hat{f}(x) - f(x)$ be the interpolation error. From the interpolation conditions, we know that $g(x_j) = 0$ for $j = 0$, ..., p. We suppose that $0 \leq j \leq p-1$ and apply the mean value theorem with $a = x_j$, $b = x_{j+1}$, and $u(x) = g(x)$. We learn that there is a point, $x'_j \in (x_j, x_{j+1})$ so that

$$g'(x'_j) = \frac{g(x_{j+1}) - g(x_j)}{x_{j+1} - x_j} = 0 \quad .$$

The notation x'_j indicates a point where $\hat{f}'$ interpolates f'; the prime on x_j does not refer to a derivative of x with respect to some other variable. In other words, $\hat{f}'$ is the degree $p-1$ polynomial that interpolates $\prime f$ at the points $x'_0 < x'_1 < \cdots < x'_{p-1}$. The derivative interpolating points interlace the function interpolation points in the sense that

$$x_0 < x'_0 < x_1 < x'_1 < \cdots < x'_{p-1} < x_p \quad .$$

[4]Evaluating the interpolating polynomial, $\hat{f}$, outside the interpolation stencil is called polynomial *extrapolation*.

The first stage of the first analysis continues with the application of this argument to g'. We take $a = x'_j$, $b = x'_{j+1}$, $u(x) = g'(x)$ and get an $x''_j \in (x'_j, x'_{j+1})$ with

$$\hat{f}''(x''_j) = f''(x''_j) \text{ for } j = 0, \ldots, p-2.$$

The second derivative interpolating points interlace the first derivative points:

$$x'_0 < x''_0 < x'_1 < x''_1 < \cdots < x''_{p-2} < x'_{p-1} \ .$$

We eventually stop with a single point $x_0^{(p)}$ where the p^{th} derivative is exact. We do not know where in the original stencil $x_0^{(p)}$ falls, but we do know (combining all the interlacing conditions) that it is inside the original stencil:

$$x_0 < x_0^{(p)} < x_p \ .$$

For the second stage of our first analysis of local interpolation, we remark that the $p+1$ derivative of $\hat{f}$ (a polynomial of degree p) is zero. Therefore $g^{(p+1)} = f^{(p+1)}$. We want to know that $g(x)$ is very small. We are supposed to conclude this based on the fact that g crosses the x axis many times. The trick is to use the mean value a second time (or set of times) to write a formula for $g(x)$ as a product of $p+1$ factors each of size H. To start this process, apply the mean value theorem with $u = g$, $a = x_0$, and $b = x$ to get

$$\frac{g(x) - g(x_0)}{x - x_0} = g'(x') \cdot (x - x_0) \ .$$

Since $g(x_0) = 0$, this gives

$$g(x) = g'(x') \cdot (x - x_0) \ , \tag{9}$$

which is the first step in this process. Next, take $u = g'$, $a = x'_0$, and $b = x'$. This gives

$$g'(x') = g''(x'') \cdot (x' - x'_0) \ . \tag{10}$$

Here, x'' is somewhere between x'_0 and x'. We do not know whether $x'_0 < x'_0$ or $x'_0 > x'_0$but (10) is true in either case. Substituting (10) into (9) leads to

$$g(x) = g''(x'') \cdot (x - x_0)(x' - x'_0) \ .$$

This formula has two factors of order H in the right. Continuing in this way leads to

$$g(x) = g^{(p+1)}(x^{(p+1)}) \cdot (x - x_0)(x' - x'_0) \cdots (x^{(p)} - x_0^{(p)}) \ . \tag{11}$$

This is the desired formula. Remembering that $g^{(p+1)} = f^{(p+1)}$, we see that

$$|g(x)| \leq const \cdot H^{(p+1)} \ , \tag{12}$$

where the constant depends on the maximum of the $(p+1)^{st}$ derivative of f but now on the interpolation points. The bound (12) tells us that degree p interpolation has an error of order H^{p+1}. This says that linear interpolation is second order accurate, cubic interpolation is fourth order, and so on.

The second way to look at low order interpolation may seem clumsy and roundabout, but it is more general and illustrates a fundamental idea in numerical analysis, stability. To avoid some technicalities, I reformulate the local interpolation problem slightly. Suppose $x_0 < x_1 < \cdots < x_k$ are $k+1$ points for interpolation, and that x is within the interpolation stencil: $x_0 < x < x_k$. The points are supposed to be "quasi uniform" in that thre are numbers $0 < \alpha < beta$ with

$$\alpha h \leq |x_{j+1} - x_j| \leq \beta h \quad \text{for } j = 0, \ldots, k-1. \tag{13}$$

The constants in the estimates below depend on α, β, k, and on the maximum of $f^{(k+1)}(x)$ within the interpolation stencil, but they do not depend on the specific points or function otherwise.

The goal is to get an estimate on the difference between $\hat{f}(x)$ and $f(x)$, where, as before, $\hat{c}$ is the degree interpolating polynomial. Our approach will be to view $\hat{f}$ as thge solution to the interpolation conditions, (1). We study this solution in two stages, called *consistency*, and *stability* respectively. This approach makes great use of the miracle of linearity, as will be clear soon.

We want to find a solution to the interpolation conditions that is close to f (remember that this solution is *the* solution). The consistency step finds a near solution, $\tilde{f}$, which is a polynomial of degree k that is near f but does not satisfy the interpolation conditions exactly. This is the easy part. In the stability step we show that $\hat{f}$ is close to $\tilde{f}$. We do this by noting that $g(x) = \tilde{f} - \hat{f}$ is small at the interpolation points, because $\hat{f}$ satisfies the interpolation equations exactly and $\tilde{f}$ satisfies them approximately. The stability step is to show that $g(x)$ is small for all x inside the stencil, provided that g is small at the interpolation points.

Let x_* be any point within the interpolation stencil ($x_0 \le x_* \le x_k$). Define $\tilde{f}$ to be the Taylor series expansion of x about x_* up to degree k:

$$\tilde{f}(x) = f(x_*) + (x - x_*)f'(x_*) \cdots + (x - x_*)^k f^{(k)}(x_*)/k! \ . \tag{14}$$

Because x_* and x are both within the interpolation stencil, and in view of (13), $|x - x_*| \le k\beta h$. Therefore the error for Taylor series approximation is

$$\left| f(x) - \tilde{f}(x) \right| \le \left(\frac{\max_{y \in [x_0, x_k]} f^{(k+1)}(y)}{(k+1)!} \right) h^{k+1} \ . \tag{15}$$

In particular, take g to be degree k polynomial which is the difference between $\hat{f}$ and $\tilde{f}$, then g satisfies the interpolation conditions

$$g(x_j) = r_j \ \text{ for } j = 0, \ldots, j = k, \text{ with } |r_j| \le const \cdot h^{k+1}. \tag{16}$$

This is the consistency step; we found a simple polynomial that at the same time is a good approximation for f and almost satisfies the interpolation conditions (1). The *residuals*, r_j are the extent to which $\tilde{f}$ fails to satisfy the interpolation equations that define $\hat{f}$.

The stability step is to show that small residuals imply small errors. Numerical methods fail because of instability more than they fail from inconsistency. An example of this is the failure of high order polynomial interpolation discussed in the next section.

In our situation, the stability step is to show that the small residuals in (**??**) imply that $g(x)$ is small throughout the interpolation stencil. Because $\tilde{f}$ is already known to be close to f, the fact that $\hat{f} - \tilde{f}$ is small will imply that $\hat{f}$ is close to f, as desired. This is formalized in the

Lemma 1. *Suppose that the $k+1$ interpolation points are quasi uniform in the sense of (13), then there is a constant, C, that depends only on α, β, and k, so that if*

$$|g(x_j)| = |r_j| \le \epsilon \ \textit{for } j = 0, \ldots, k,$$

then

$$|g(x)| \le C\epsilon \ \textit{for } x_0 \le x \le x_k.$$

Ultimately, the proof of this lemma is about the norm of a matrix[5], the inverse of the vanderMonde matrix. To get to that point, we do a preliminary reparametrization and a rescaling. The reparametrization is to put 0 inside the interpolation stencil. Choose any x_* in the interpolation interval and let $y = x - x_*$, and $y_j = x_j - x_*$. The rescaling removes the h: $y = h \cdot z$, $y_j = h \cdot z_j$ (defining z in terms of y).

In these variables, Lemma 1 may be reformulated as follows. Suppose $z_0 < z_1 < \cdots < z_k$ and $\alpha \le z_{k+1} - z_k \le \beta$ for $j = 0, \ldots, j = k-1$, and $z_0 < 0 < z_k$. Suppose that $g(z)$ is a polynomial of degree k in z and that

$$g(z_j) = r_j \ , \quad |r_j| \le \epsilon \ \text{ for } j = 0, \ldots, j = k.$$

[5] Stability issues almost always boil down to understanding the norm of something.

Then

$$|g(z)| \leq C\epsilon \text{ for all } z \in [z_0, z_k]. \tag{17}$$

Now we get to the matrix at the heart of the matter. Consider the polynomial representation of $g(z)$:

$$g(z) = b_0 + b_1 z + \cdots + b_k z^k \ .$$

We will prove (17) if we show that

$$|b_j| \leq C\epsilon \text{ for } j = 0, \ldots, j = k. \tag{18}$$

But the coefficients, b_j, are given in terms of the r_j using the vanderMonde matrix: $Vb = r$, $b = V^{-1}r$. From this, (18) is the statement that

$$\left\| V^{-1} \right\| \leq C \ , \tag{19}$$

where this C depends on α, β, and k, but not otherwise on specific placement of the z_j. This is what mathematicians call a "compactness argument". Since $\left\| V^{-1} \right\|$ is not infinite for any of the parameters, z_j, and the z_j range in a bounded set (This is where α and β come in.). There is a maximum possible $\left\| V^{-1} \right\|$. This is all (19) says. Therefore, we are done proving Lemma 1. We get another proof of the h^{k+1} accuracy of local degree k polynomial interpolation once we remember that the ϵ of Lemma 1 is $O(h^{k+1}$, as was established in the consistency step.

Assignment 4 gives a situation where this consistency/stability analysis works while the clever one dimensional arguments do not, as far as I can tell.

4 High order polynomial interpolation

So far we have discussed "local" polynomial interpolation. The degree of the interpolating polynomial is fixed. As h gets smaller, the number of points in the interpolating stencil remains the same while the stencil also shrinks. We showed that interpolating from $k+1$ nearby points whose spacing is on the order of h leads to $O(h^{k+1})$ accuracy in the interpolant, if f is smooth enough.

The situation is completely different in high order polynomial interpolation. There is a deep mathematical theory of this, so I want just to discuss a particular example that has the features of the general situation. In this example, we fix the size of the interpolation stencil, use equispaced interpolation points with spacing h, and let the number of interpolation points, together with the degree of the interpolating polynomial, go to infinity. We will see that as $h \to 0$, the maximal difference between $f(x)$ and $\hat{f}(x)$ inside the interpolation stencil goes to infinity. That is, the approximation error from high order interpolation goes to infinity as the interpolation mesh is refined. This makes high order polynomial interpolation very delicate.

The specific example I want to discuss is the Runge example, given by the German applied mathematician Karl Runge around the turn of the twentieth century. In this example,

$$f(x) = \frac{1}{1 + 20x^2} \ .$$

This function is as smooth as any. The interpolation stencil is the interval $[-1, 1]$. There are $n+1$ interpolation points $x_0 = -1 < x_1 < \cdots < x_n = 1$ with uniform grid spacing $x_{j+1} - x_j = h = 2/n$ for $j = 0, \ldots, n-1$. Figures 1 through 5 illustrate the Runge phenomenon. As the degree of the interpolating polynomial increases, the fit first gets better, then starts to get very bad near the ends of the interval.

One interesting thing about this example is that it is perfectly possible to approximate f by high order polynomials. In fact, in our example, there is a family of approximating polynomials, $p_n(x)$, of degree n so that

$$|p_n(x) - f(x)| \leq e^{-\alpha n} \ , \tag{20}$$

for some positive constant, α. However, $p_n(x)$ is not the interpolating polynomial on the uniformly spaced interpolation points given. One such family of polynomials, though not the absolutely most accurate, is the polynomials that interpolate the Runge function at the Thcebychev points. Figures 6 - 9 illustrate the Thcebechev interpolating polynomials and interpolation points. Note the the points are more closely

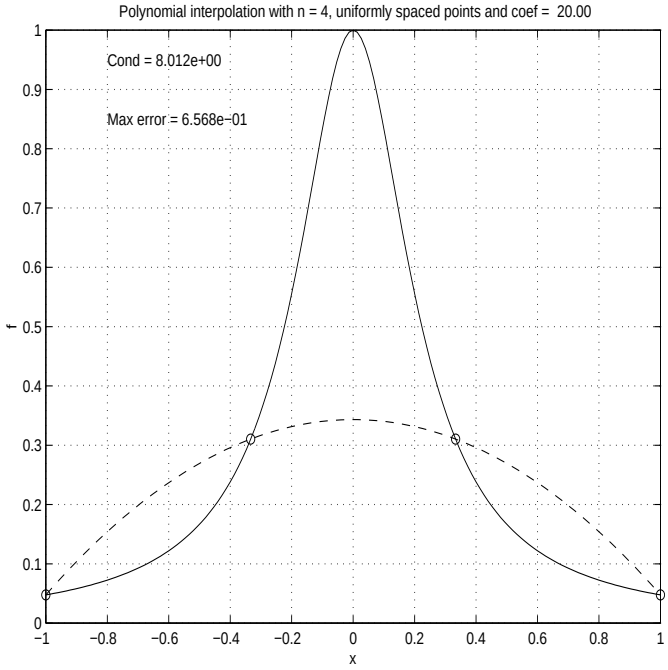

Figure 1: Interpolation of the Runge function by a polynomial of degree 4 at uniformly spaced interpolation points.

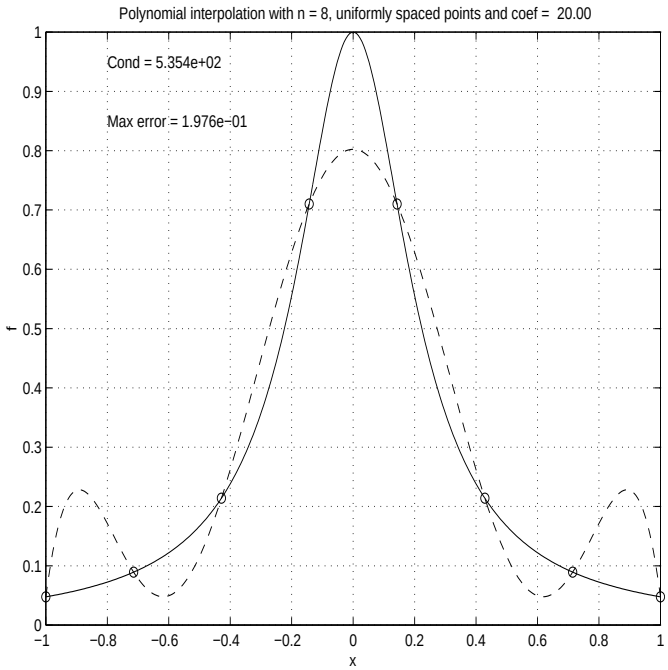

Figure 2: Interpolation of the Runge function by a polynomial of degree 8 at uniformly spaced interpolation points.

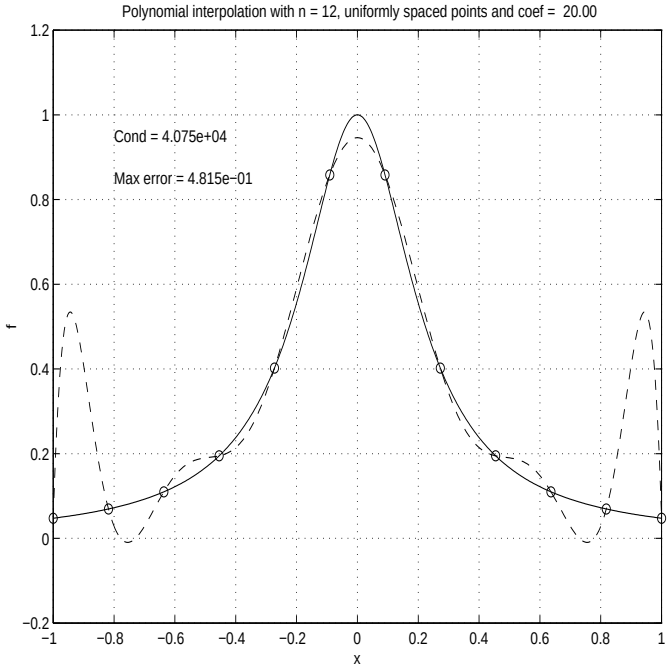

Figure 3: Interpolation of the Runge function by a polynomial of degree 12 at uniformly spaced interpolation points.

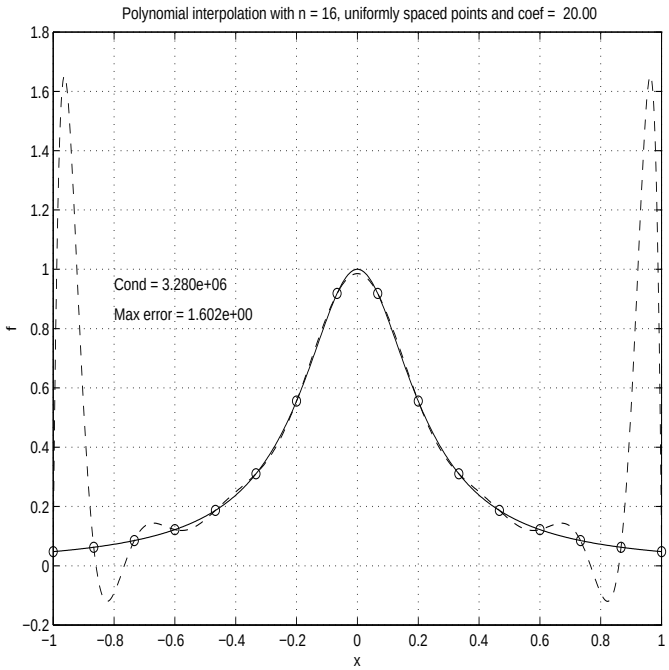

Figure 4: Interpolation of the Runge function by a polynomial of degree 16 at uniformly spaced interpolation points.

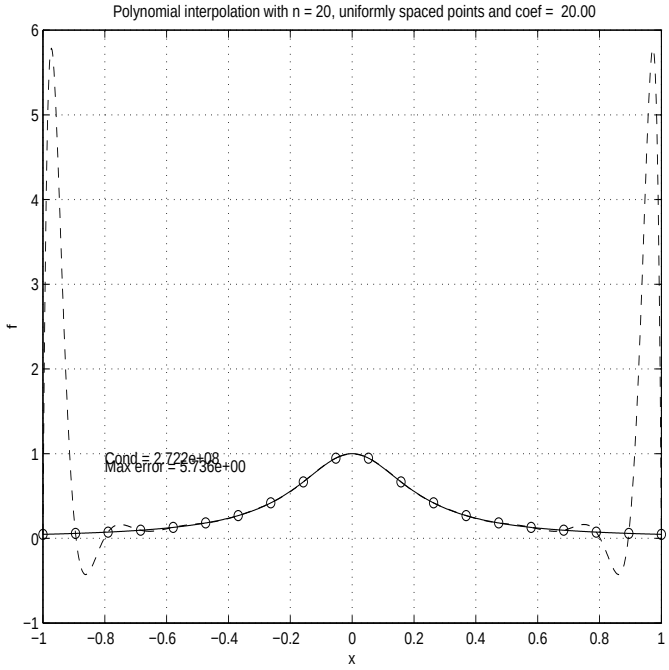

Figure 5: Interpolation of the Runge function by a polynomial of degree 20 at uniformly spaced interpolation points.

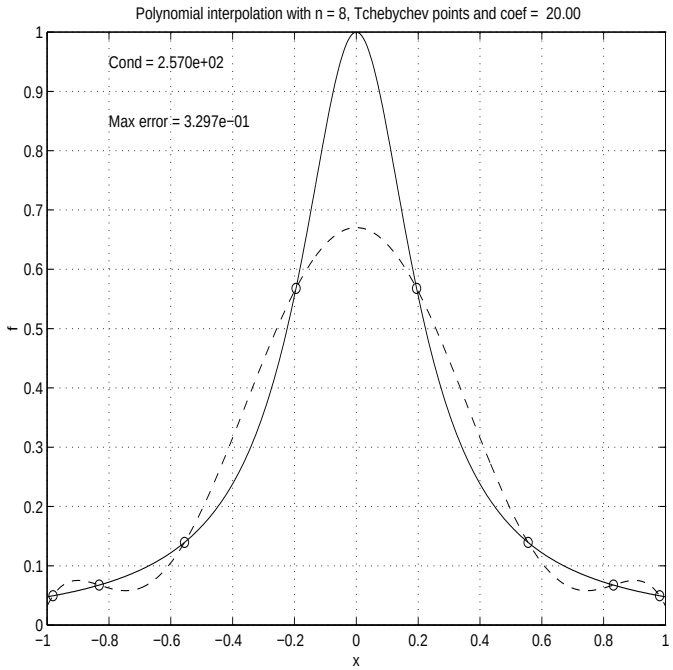

Figure 6: Interpolation of the Runge function by a polynomial of degree 8 at Tchebychev interpolation points.

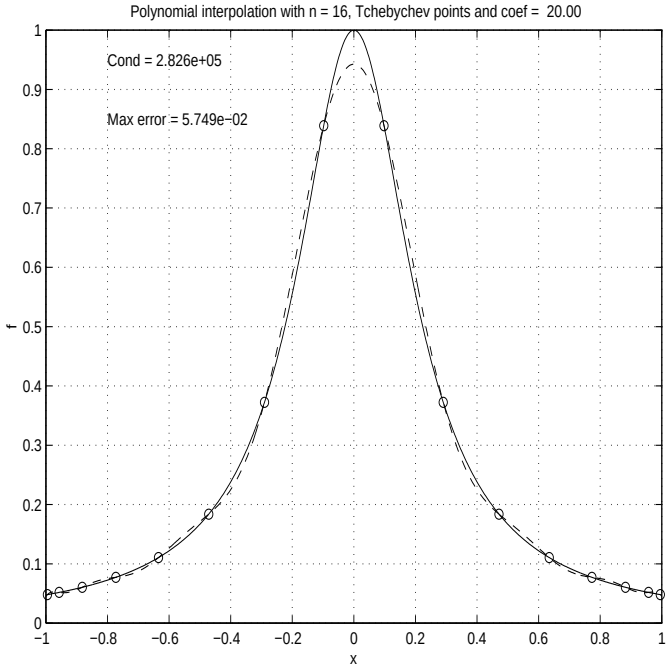

Figure 7: Interpolation of the Runge function by a polynomial of degree 16 at Tchebychev interpolation points.

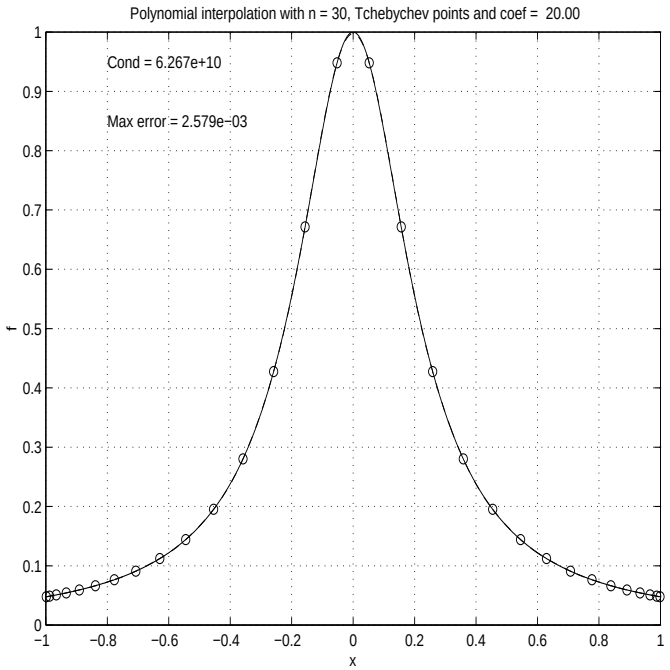

Figure 8: Interpolation of the Runge function by a polynomial of degree 30 at Tchebychev interpolation points.

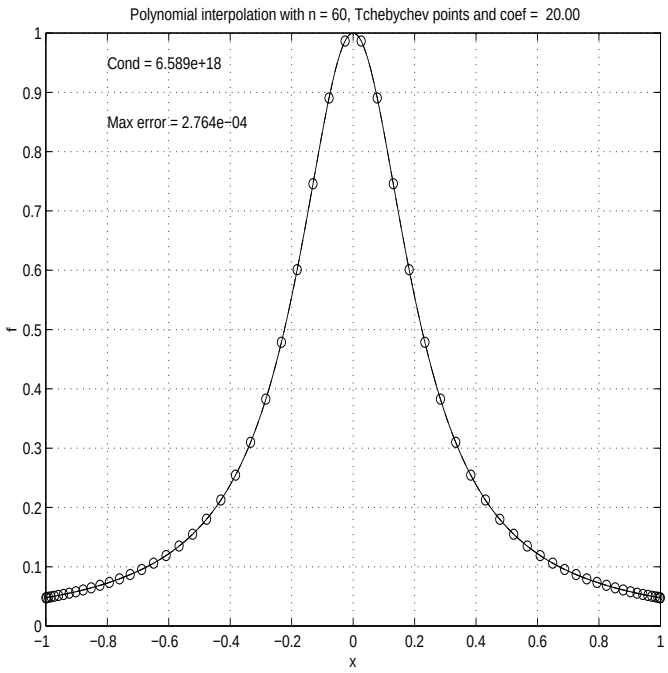

Figure 9: Interpolation of the Runge function by a polynomial of degree 60 at Tchebychev interpolation points. The interpolating polynomial still looks reasonable even though the condition number of the vanderMonde matrix is so large that the coefficients are not computed accurately.

spaced near the endpoints. This prevents the interpolating polynomial from oscillating so wildly at near the endpoints.

Figures 10 and 11 give the error in the Tchebychev interpolating polynomial for various n. The first two illustrate the main features of this error; it is an even function of x, and it is oscillatory with amplitude largest in the center. Figure 12 demonstrates that the interpolating polynomial is no longer calculated accurately. The error, though small, is no longer a symmertic function of x.

The claim for the expolential accuracy of the Tchebychev interpolating polynomial is backed up in figure 13. There the maximum error over the interval is plotted as a function of n. Where the curve is computed correctly, it is a straight line. Beyond a certain point, the condition number of the vanderMonde matrix is so large that the interpolating polynomial is not computed correctly by the method I used, which was to form and invert that matrix. Figure 14 is a plot of the condition number as a function of n. Clearly, this condition number is exponentially increasing.

Let us conclude with a summary of the consistence/stability "spin" on low and high order polynomial interpolation. The interpolating polynomial is found by solving the interpolation conditions (1). For a smooth f, it is possible to find a polynomial that is a good approximation to f and therefore provides an approximate solution to the interpolating conditions. For low degree, the condition number of V is such that a the approximate solution of (1) is close to the exact solution, and therefore, the exact interpolating polynomial is close to the approximate interpolating polynomial. Since the approximate interpolating polynomial is a close to f, the exact interpolating polynomial will also be close to f. However, for high degree interpolation, the condition number is very large (Figure 14). Therefore the exact interpolating polynomial need not be close to the approximate interpolating polynomial, and therefore need not be close to f.

In the mathematical theory of high order interpolation, the poor approximation behavior of the polynomial that interpolates at uniformly spaced points is attributed to the fact that the Runge function has a pole in the complex plane that is close to the real axis. Why the Tchebychev interpolating polynomial does not suffer is a topic for another time.

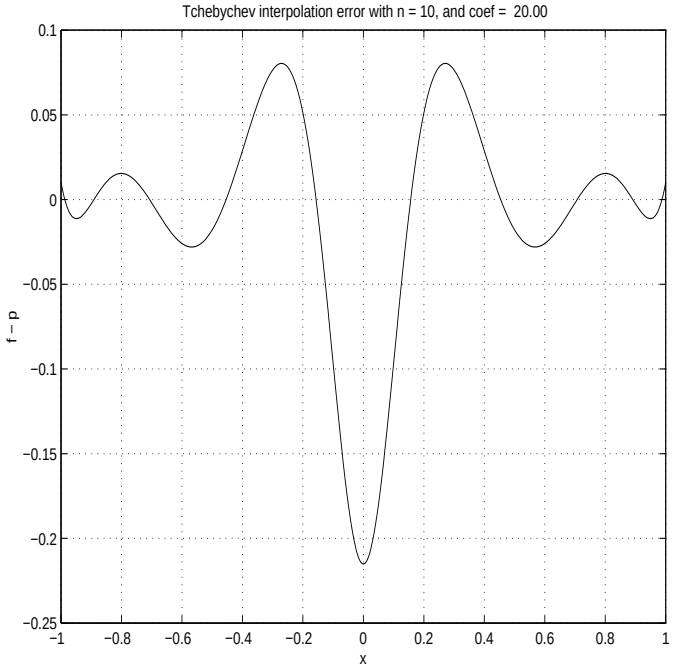

Figure 10: The difference between the Runge function and the Tchebychev interpolating polynomial for $n = 10$ interpolating points.

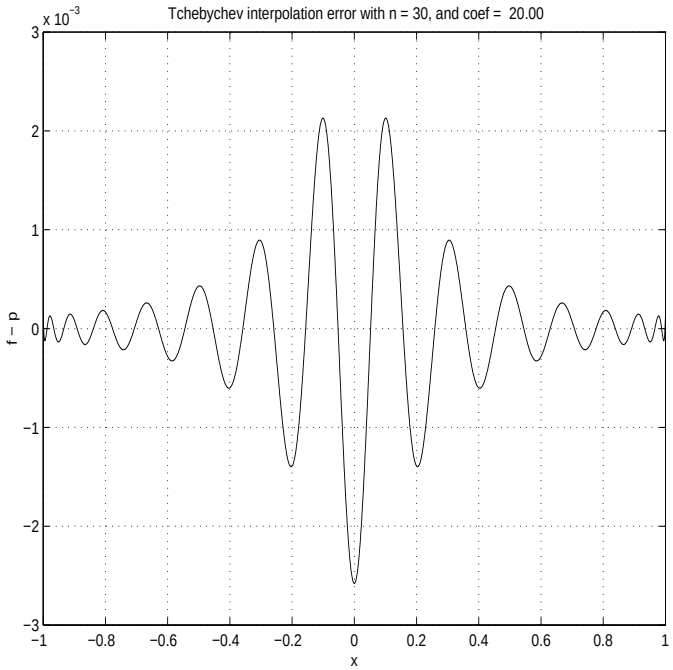

Figure 11: The difference between the Runge function and the Tchebychev interpolating polynomial for $n = 30$ interpolating points.

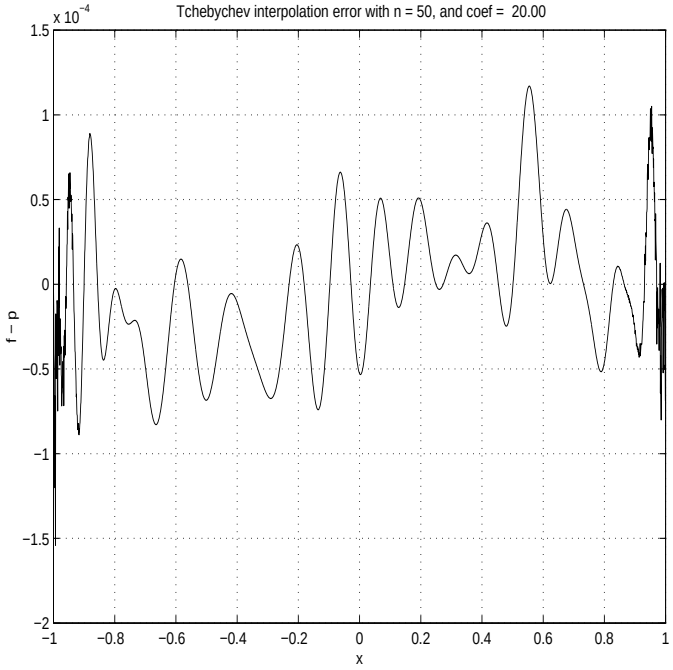

Figure 12: The difference between the Runge function and the Tchebychev interpolating polynomial for $n = 50$ interpolating points. Note that this graph is no longer symmetric in x, because the polynomial is not computed accurately.

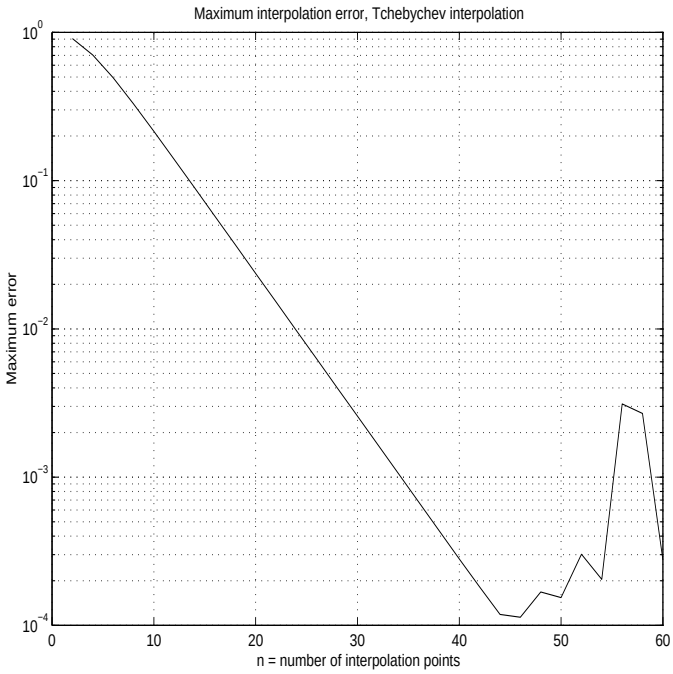

Figure 13: Error in the polynomial that interpolates the Runge function at the Thcebychev points, as computed in double precision by MATLAB. The fact that the graph is linear as a function of n on a log scale indicates that the maximum error is a decreasing exponential function of n. Beyond $n \approx 43$, the condition number of the vanderMonde matrix is so large that the interpolating polynomial is not computed accurately.

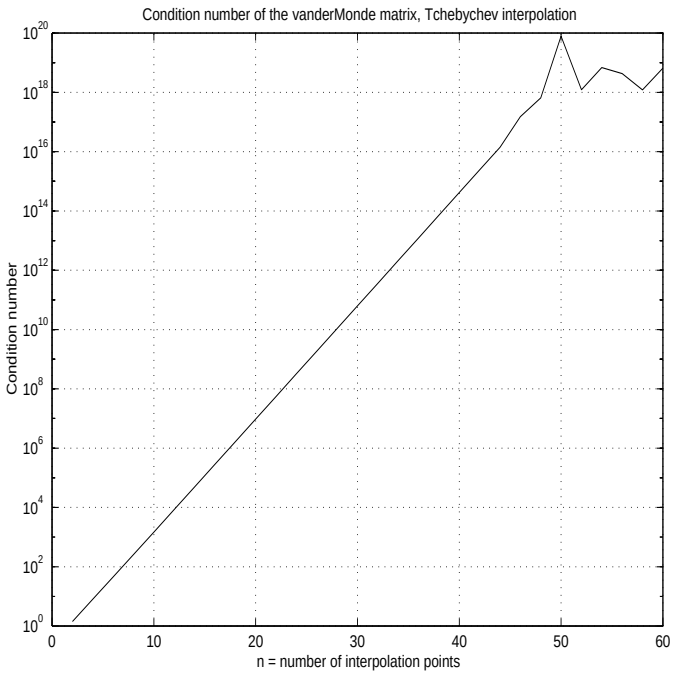

Figure 14: Condition number of the vanderMonde matrix for polynomial interpoation at the Tchebychev points, as computed in double precision by MATLAB. The fact that the graph is linear as a function of n on a log scale indicates that the condition number is an increasing exponential function of n. Beyond $n \approx 45$, the condition number of the vanderMonde matrix is so large that it is not computed accurately.

5 A word about splines

Spline interpolation differs from local polynomial interpolation to produce smoother approximating curves. If we are interpolating a smooth function, the value of $\hat{f}(x)$ gotten from different stencils are very close to $f(x)$, and to each other. Still they are not equal. Therefore, the *curve* $\hat{f}(x)$ may have discontinuities at points where the stencil changes. If the stencil changes only at interpolation points, then $\hat{f}$ will be continuous. However, then $\hat{f}'$ will be discontinuous. The interpolation points will be clearly visible on a plot of $\hat{f}$ as places where the slope of $\hat{f}$ jumps.

Splines produce smoother curves by allowing a greater degree of smoothness at the points where the polynomial changes. We describe this in the terminology usually used for splines. We have a sequence of *knot* points $x_0 < x_1 < \cdots < x_n$. We say that the function $\hat{f}(x)$ is a degree p spline with continuity of order q if

1. For each interval $[x_k, x_{k+1}]$ there is a polynomial[6], $p_k(x)$, of degree $\leq p$ with $\hat{f}(x) = p_k(x)$ for all $x \in [x_k, x_{x+1}]$.

2. All derivatives of $\hat{f}$ up to and including order q are continuous at the know points. This includes continuity of $\hat{f}$ itself, which is considered to be the dervative of order zero.

Example: Piecewise linear interpolation produces a spline of degree 1 with continuity of order 0.
Example: Splines with degree 3 and continuity of order 2 are **cubic b–splines**. Some very important cubic b–splines are the b–spline basis functions. Suppose that $k \geq 3$ and $k \leq n-3$, then there is a spline, $b_k(x)$, that takes the value 1 at x_k and is equal to zero at outside the interval $[x_{k-3}, x_{k+3}]$. The construction of $b_k(x)$ requires us to find 6 polynomials of degree 3 that satisfy continuity conditions up to order 2 at the knot points. This is not too hard when the knot poitns are equally spaced.

[6]Before this, $p_k(x)$ would have been a polynomial of degree k. Now, k refers to the interval, not the degree of p.