

## Chapter 10

# Polynomial Interpolation

### 10.1 Undetermined Coefficients

The method of undetermined coefficients provides a solution to the problem of finding a parametric curve which passes through a set of points. For example, suppose we wish to find a cubic parametric curve which passes through the points  $(0,0)$ ,  $(2,2)$ ,  $(0,3)$ , and  $(2,4)$ . We must first specify at what parameter value the curve will interpolate each point. Lets say we want  $(0,0)$  to have a parameter value of 0,  $(2,2)$  is to have a parameter value of  $1/4$ ,  $(0,3)$  should correspond to parameter  $t = 3/4$ , and  $(2,4)$  should have a parameter value of 1.

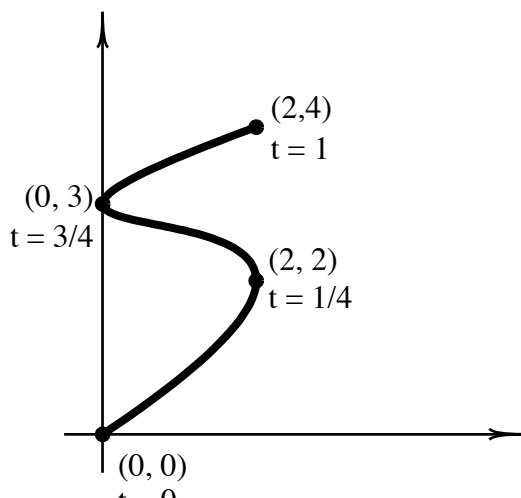


Figure 10.1: Interpolating Four Points

We can now pick any form we wish for the parametric equations. Let's first see how it is done for standard power basis polynomials, and then we will solve the same problem using Bernstein polynomials. For power basis polynomials, the parametric equations are of the form

$$x = a_0 + a_1t + a_2t^2 + a_3t^3$$

$$y = b_0 + b_1t + b_2t^2 + b_3t^3$$

To solve for the  $a_i$ , we set up four linear equations:

$$\begin{aligned} 0 &= a_0 + a_1 \cdot 0 + a_2 \cdot 0^2 + a_3 \cdot 0^3 \\ 2 &= a_0 + a_1\left(\frac{1}{4}\right) + a_2\left(\frac{1}{4}\right)^2 + a_3\left(\frac{1}{4}\right)^3 \\ 0 &= a_0 + a_1\left(\frac{3}{4}\right) + a_2\left(\frac{3}{4}\right)^2 + a_3\left(\frac{3}{4}\right)^3 \\ 2 &= a_0 + a_1 \cdot 1 + a_2 \cdot 1^2 + a_3 \cdot 1^3 \end{aligned}$$

In matrix form, we have

$$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 1 & \frac{1}{4} & \left(\frac{1}{4}\right)^2 & \left(\frac{1}{4}\right)^3 \\ 1 & \frac{3}{4} & \left(\frac{3}{4}\right)^2 & \left(\frac{3}{4}\right)^3 \\ 1 & 1 & 1 & 1 \end{bmatrix} \begin{Bmatrix} a_0 \\ a_1 \\ a_2 \\ a_3 \end{Bmatrix} = \begin{Bmatrix} 0 \\ 2 \\ 0 \\ 2 \end{Bmatrix}.$$

from which

$$x = 18t - 48t^2 + 32t^3$$

Likewise for  $y$ , we solve the set of equations

$$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 1 & \frac{1}{4} & \left(\frac{1}{4}\right)^2 & \left(\frac{1}{4}\right)^3 \\ 1 & \frac{3}{4} & \left(\frac{3}{4}\right)^2 & \left(\frac{3}{4}\right)^3 \\ 1 & 1 & 1 & 1 \end{bmatrix} \begin{Bmatrix} b_0 \\ b_1 \\ b_2 \\ b_3 \end{Bmatrix} = \begin{Bmatrix} 0 \\ 2 \\ 3 \\ 4 \end{Bmatrix}.$$

and we solve for

$$y = 12t - \frac{56}{3}t^2 + \frac{32}{3}t^3.$$

Let's next look at how we could solve directly for the Bézier control points of a Bézier curve which interpolates those same points at those same parameter values. The only difference between this problem and the one we just solved is the form of the polynomial. In this case, we want to solve for the coefficients of a Bernstein polynomial:

$$x = a_0(1-t)^3 + 3a_1t(1-t)^2 + 3a_2t^2(1-t) + a_3t^3.$$

We evaluate this expression at  $x = 0, t = 0$ , again at  $x = 2, t = \frac{1}{4}$ , again at  $x = 0, t = \frac{3}{4}$  and again at  $x = 2, t = 1$  to produce a set of equations:

$$\begin{aligned} 0 &= a_0(1-0)^3 + 3a_1 \cdot 0(1-0)^2 + 3a_2 \cdot 0^2(1-0) + a_3 \cdot 0^3 \\ 2 &= a_0\left(1-\frac{1}{4}\right)^3 + 3a_1\frac{1}{4}\left(1-\frac{1}{4}\right)^2 + 3a_2\frac{1^2}{4}\left(1-\frac{1}{4}\right) + a_3\frac{1^3}{4} \\ 0 &= a_0\left(1-\frac{3}{4}\right)^3 + 3a_1\frac{3}{4}\left(1-\frac{3}{4}\right)^2 + 3a_2\frac{3^2}{4}\left(1-\frac{3}{4}\right) + a_3\frac{3^3}{4} \\ 2 &= a_0(1-1)^3 + 3a_1 \cdot 1(1-1)^2 + 3a_2 \cdot 1^2(1-1) + a_3 \cdot 1^3 \end{aligned}$$

In matrix form, we have

$$\begin{bmatrix} 1 & 0 & 0 & 0 \\ \frac{27}{64} & \frac{27}{64} & \frac{9}{64} & \frac{1}{64} \\ \frac{1}{64} & \frac{64}{64} & \frac{27}{64} & \frac{64}{64} \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} a_0 \\ a_1 \\ a_2 \\ a_3 \end{bmatrix} = \begin{bmatrix} 0 \\ 2 \\ 0 \\ 2 \end{bmatrix}.$$

The  $x$  coordinates  $a_i$  of the Bézier control points work out to be  $a_0 = 0$ ,  $a_1 = 6$ ,  $a_2 = -4$ , and  $a_3 = 2$ . We can perform a similar computation to compute the  $y$  coordinates of the Bézier control points. They work out to be 0, 4,  $\frac{16}{9}$ , and 4.

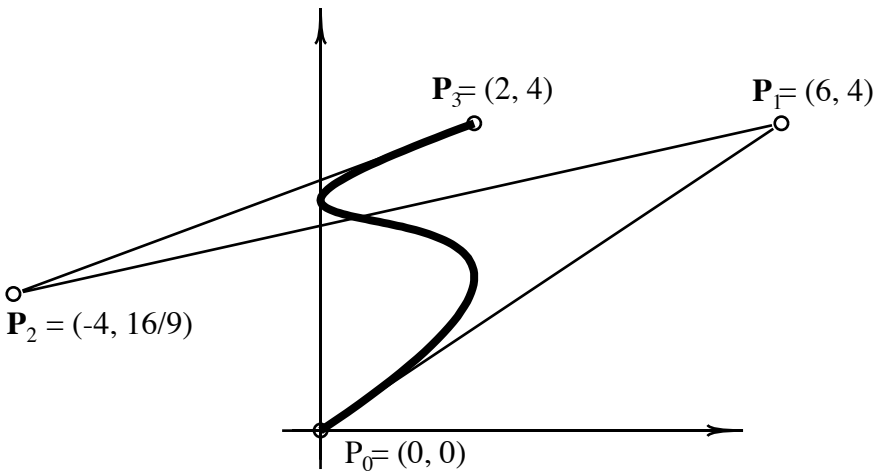


Figure 10.2: Interpolating Four Points

## 10.2 Lagrange Interpolation

There exists a clever set of basis polynomials which enable us to interpolate a set of points without having to solve a set of linear equations. These are known as Lagrange polynomials and are denoted  $L_i^n(t)$ . The purpose of Lagrange polynomials is to enable us, with virtually no computation, to find a degree  $n$  parametric curve which interpolates  $n + 1$  points  $\mathbf{P}_0, \mathbf{P}_1, \dots, \mathbf{P}_n$  at parameter values  $t_0, t_1, \dots, t_n$ . The curve is defined by

$$\mathbf{P}(t) = \mathbf{P}_0 L_0^n(t) + \mathbf{P}_1 L_1^n(t) + \dots + \mathbf{P}_n L_n^n(t).$$

Note the following about the  $L_i^n(t)$ :  $L_i^n(t_j) = 1$  whenever  $i = j$  and  $L_i^n(t_j) = 0$  whenever  $i \neq j$ . This must be so in order for the curve to interpolate point  $\mathbf{P}_i$  at parameter value  $t_i$ . You can easily verify that the following choice for  $L_i^n(t)$  satisfies those conditions:

$$L_i^n(t) = \frac{(t - t_0)(t - t_1) \dots (t - t_{i-1})(t - t_{i+1}) \dots (t - t_n)}{(t_i - t_0)(t_i - t_1) \dots (t_i - t_{i-1})(t_i - t_{i+1}) \dots (t_i - t_n)}$$

or

$$L_i^n(t) = \prod_{j=0, j \neq i}^n \frac{(t - t_j)}{(t_i - t_j)}, \quad j \neq i$$

In the example from the previous section, we have

$$L_0^3(t) = \frac{(t - \frac{1}{4})(t - \frac{3}{4})(t - 1)}{(0 - \frac{1}{4})(0 - \frac{3}{4})(0 - 1)} = -\frac{16}{3}(t - \frac{1}{4})(t - \frac{3}{4})(t - 1)$$

$$L_1^3(t) = \frac{(t - 0)(t - \frac{3}{4})(t - 1)}{(\frac{1}{4} - 0)(\frac{1}{4} - \frac{3}{4})(\frac{1}{4} - 1)} = \frac{32}{3}t(t - \frac{3}{4})(t - 1)$$

$$L_2^3(t) = \frac{(t - 0)(t - \frac{1}{4})(t - 1)}{(\frac{3}{4} - 0)(\frac{3}{4} - \frac{1}{4})(\frac{3}{4} - 1)} = -\frac{32}{3}t(t - \frac{1}{4})(t - 1)$$

$$L_3^3(t) = \frac{(t - 0)(t - \frac{1}{4})(t - \frac{3}{4})}{(1 - 0)(1 - \frac{1}{4})(1 - \frac{3}{4})} = \frac{16}{3}t(t - \frac{1}{4})(t - \frac{3}{4})$$

The interpolating curve is thus

$$\mathbf{P}(t) = \begin{Bmatrix} x(t) \\ y(t) \end{Bmatrix} = \begin{Bmatrix} 0 \\ 0 \end{Bmatrix} L_0^3(t) + \begin{Bmatrix} 2 \\ 2 \end{Bmatrix} L_1^3(t) + \begin{Bmatrix} 0 \\ 3 \end{Bmatrix} L_2^3(t) + \begin{Bmatrix} 2 \\ 4 \end{Bmatrix} L_3^3(t)$$

If this expression is expanded out, it is seen to be identical to the equation we obtained using the method of undetermined coefficients.

It can be shown that the Lagrange blending functions are coordinate system independent, but do not obey the convex-hull property.

### 10.3 Newton Polynomials

Newton polynomials provide an additional method for finding a degree- $n$  polynomial  $p(t)$  that interpolates  $n + 1$  points:

$$p(t_i) = y_i, \quad i = 0, \dots, n.$$

Newton polynomials are defined

$$\begin{aligned} p_0(t) &= a_0 \\ p_1(t) &= a_0 + a_1(t - t_0) \\ p_2(t) &= a_0 + a_1(t - t_0) + a_2(t - t_0)(t - t_1) \quad \dots \\ p_n(t) &= p_{n-1}(t) + a_n(t - t_0)(t - t_1) \cdots (t - t_{n-1}). \end{aligned}$$

The coefficients  $a_i$  can be computed using *divided differences*, which can be found using a table that is reminiscent of forward differences. The notation for a divided difference is  $f_{i, \dots, k}$ , and the coefficients  $a_i$  turn out to be:

$$a_0 = f_0, \quad a_1 = f_{0,1}, \quad a_2 = f_{0,1,2}, \dots, \quad a_n = f_{0,1, \dots, n}.$$

The divided differences are computed using a recurrence relation:

$$f_i = y_i$$

$$f_{i,1+1} = \frac{f_{i+1} - f_i}{t_{i+1} - t_i}$$

$$f_{i,\dots,k} = \frac{f_{i+1,\dots,k} - f_{i,\dots,k-1}}{t_k - t_i}$$

These divided differences are easily calculated by means of a divided difference table:

$t_0$	$y_0 = f_0$					
$t_1$	$y_1 = f_1$	$f_{0,1}$				
$t_2$	$y_2 = f_2$	$f_{1,2}$	$f_{0,1,2}$			
$t_3$	$y_3 = f_3$	$f_{2,3}$	$f_{1,2,3}$	$f_{0,1,2,3}$		
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	
$t_n$	$y_n = f_n$	$f_{n-1,n}$	$\cdots$	$\cdots$	$\cdots$	$f_{0,1,\dots,n}$

## 10.4 Neville's Scheme

Neville's scheme addresses the problem: Given a set of  $n + 1$  points and a parameter value assigned to each point

$$p(t_i) = p_i, \quad i = 0, \dots, n.$$

find the point  $p(t)$  for any other point on the unique degree  $n$  polynomial curve that interpolates those points. Note that while Lagrange and Newton polynomials give us an actual curve equation, Neville's scheme provides us with a *geometric construction* for finding any point on the curve, without formally using an equation for the curve.

Neville's scheme is based on Aitken's Lemma. Denote by  $p_{i,\dots,k}(t)$  the degree  $k - 1$  polynomial curve that interpolates points  $p_i, \dots, p_k$ . Aitken's Lemma states:

$$p_{i,\dots,k}(t) = \frac{(t_k - t)p_{i,\dots,k-1}(t) - (t_i - t)p_{i+1,\dots,k}(t)}{t_k - t_i}.$$

Neville's scheme applies Aitken's Lemma as follows. Suppose we wish to find the point  $p(t)$ . Begin by labeling the interpolated points as  $p_0, \dots, p_n$ . Then compute the  $n$  points

$$p_{i,i+1} = \frac{(t_{i+1} - t)p_i - (t_i - t)p_{i+1}}{t_{i+1} - t_i}, \quad i = 0, \dots, n - 1.$$

Next compute the  $n - 1$  points

$$p_{i,i+1,i+2} = \frac{(t_{i+2} - t)p_{i,i+1} - (t_i - t)p_{i+1,i+2}}{t_{i+2} - t_i}, \quad i = 0, \dots, n - 2.$$

Then likewise compute the  $n - 2$  points  $p_{i,i+1,i+2,i+3}$  and so forth until  $p_{i,\dots,n} = p(t)$  is computed.

## 10.5 Error Bounds

We can use degree  $n$  Lagrange polynomials to approximate any function by interpolating  $n+1$  points  $f(x_0), f(x_1), \dots, f(x_n)$  on the function. In the following equation,  $f(x)$  is the function we wish to approximate and  $p(x)$  is the Lagrange approximation.

$$f(x) - p(x) = \frac{f^{(n+1)}(\xi)}{(n+1)!} (x - x_0)(x - x_1) \cdots (x - x_n) \quad (10.1)$$

The value  $f^{(n+1)}(\xi)$  is the  $(n+1)^{th}$  derivative of  $f(x)$  evaluated at some value  $\xi$  where  $\min(x, x_0, \dots, x_n) \leq \xi \leq \max(x, x_0, \dots, x_n)$ . If we can determine a bound on this derivative, then equation 10.1 can be used to provide an error bound on our approximation.

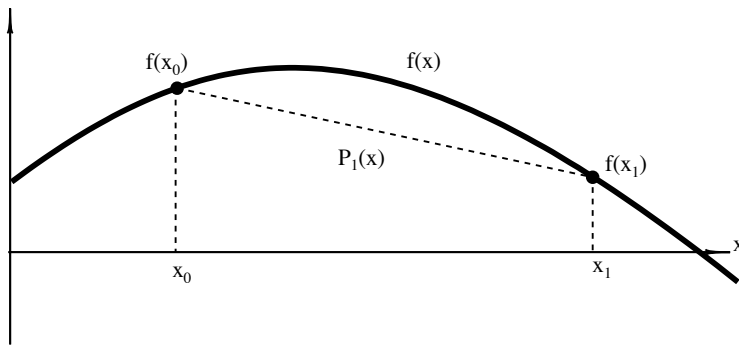


Figure 10.3: Error Bounds

A particularly useful case for equation 10.1 is  $n = 1$ , or linear approximation. In this case, we can compute the maximum distance that a polynomial deviates from a straight line. If the straight line is

$$p_1(x) = \frac{f(x_0)(x_1 - x) + f(x_1)(x - x_0)}{x_1 - x_0}$$

then

$$|f(x) - p_1(x)| \leq \frac{\max_{(\min(x, x_0, x_1) \leq x \leq \max(x, x_0, x_1))} (f''(x))}{2} (x - x_0)(x - x_1).$$

Let  $\delta = x_1 - x_0$  and  $L = \max_{(\min(x, x_0, x_1) \leq x \leq \max(x, x_0, x_1))} (f''(x))$ . Since the expression  $(x - x_0)(x - x_1)$  has a maximum value at  $x = \frac{x_0 + x_1}{2}$  of  $\frac{(x_1 - x_0)^2}{4} = \frac{\delta^2}{4}$ ,

$$|f(x) - p_1(x)| \leq L \frac{\delta^2}{8}.$$

We can assure that the approximation error will be less than a specified tolerance  $\epsilon$  by using  $m$  line segments whose endpoints are evenly spaced in  $x$ , where

$$m \geq \sqrt{\frac{L}{8\epsilon}} (x_1 - x_0).$$

A useful application of this idea is to determine how many line segments are needed for plotting a Bézier curve so that the maximum distance between the curve and the set of line segments is less

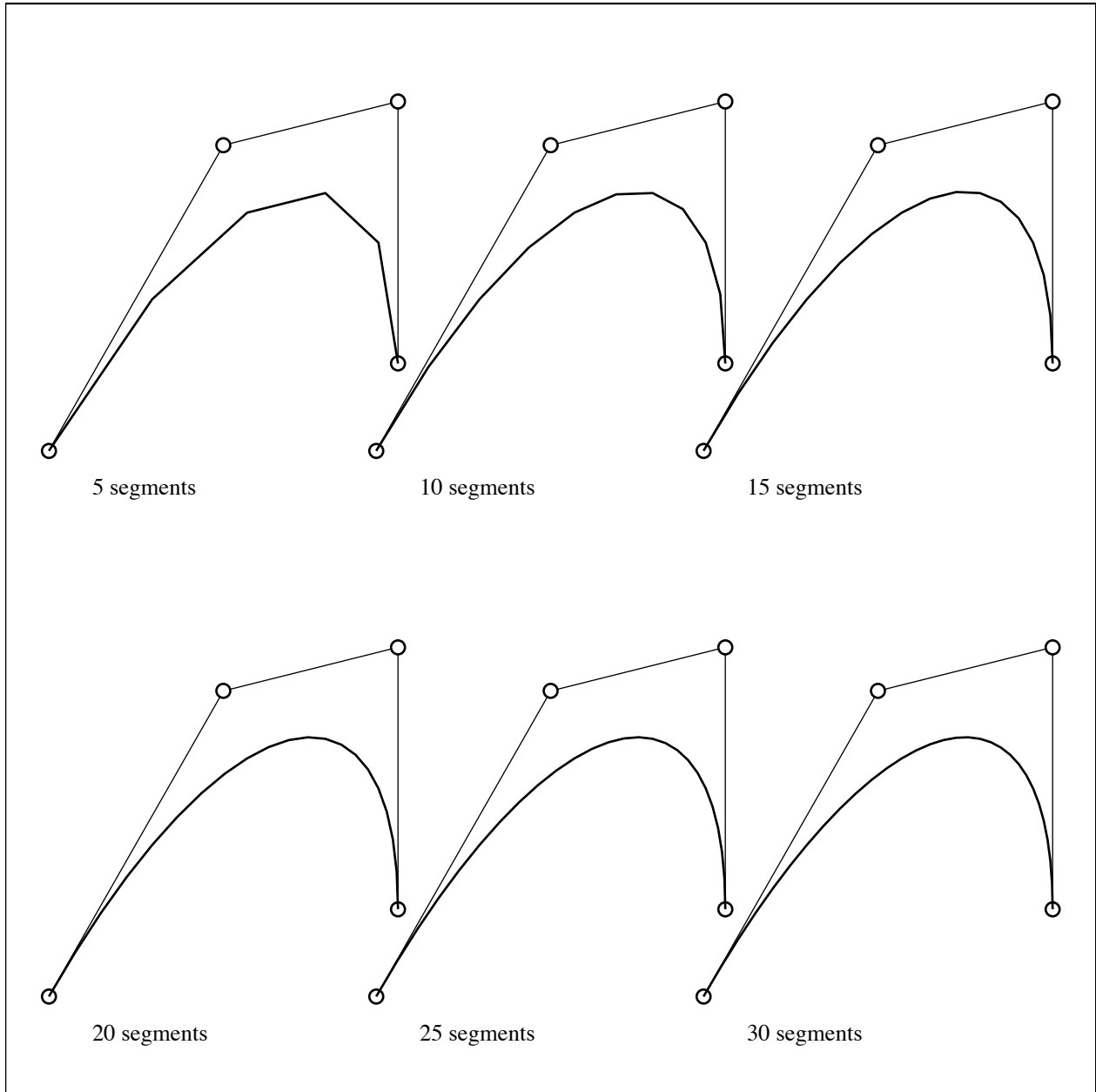


Figure 10.4: Piecewise linear approximation of a Béziercurve

than  $\epsilon$ . Figure 10.4 shows a cubic Béziercurve approximated with various numbers of line segments. In this case, we simply get a bound on the error in  $x$  coordinates and  $y$  coordinates independently, and apply the Pythagorean theorem. Bounds  $(L_x, L_y)$  on the second derivatives of Bézier curves are easily obtained by computing the second hodograph:

$$L_x = n(n-1) \max_{0 \leq i \leq n-2} |x_{i+2} - 2x_{i+1} + x_i| \quad (10.2)$$

and

$$L_y = n(n-1) \max_{0 \leq i \leq n-2} |y_{i+2} - 2y_{i+1} + y_i|, \quad (10.3)$$

Now, if we use  $m$  line segments for approximating the curve where

$$m \geq \sqrt{\frac{\sqrt{L_x^2 + L_y^2}}{8\epsilon}},$$

the maximum error will be less than  $\epsilon$ .

We can likewise determine how many times  $r$  a Béziercurve must be recursively bisected until each curve segment is assured to deviate from a straight line segment by no more than  $\epsilon$  [GZ84]:

$$r = \log_2 \sqrt{\frac{\sqrt{L_x^2 + L_y^2}}{8\epsilon}} \quad (10.4)$$

This form of the approximation error equation is useful for applications such as computing the intersection between two curves.

### 10.5.1 Chebyshev Polynomials

If we wish to use a Lagrange polynomial of degree greater than one to perform approximation, we see from equation 10.1 that a wise choice of the interpolation points  $x_i$  can improve the approximation error. From equation 10.1,

$$|f(x) - p(x)| \leq \max_{a \leq x \leq b} \frac{f^{(n+1)}(x)}{(n+1)!} \max_{a \leq x \leq b} |(x-x_0)(x-x_1) \cdots (x-x_n)| \quad (10.5)$$

where  $x, x_0, \dots, x_n \in [a, b]$ . If we want to decrease the approximation error, we have no control over  $\max_{a \leq x \leq b} \frac{f^{(n+1)}(x)}{(n+1)!}$ . However, there is an optimal choice for the  $x_i$  which will minimize  $\max_{a \leq x \leq b} |(x-x_0)(x-x_1) \cdots (x-x_n)|$ .

One might make an initial guess that the best choice for the  $x_i$  to minimize  $\max_{a \leq x \leq b} |(x-x_0)(x-x_1) \cdots (x-x_n)|$  would be simply to space them evenly between  $x_0$  and  $x_n$ . This actually gives pretty good results. For the interval  $[a, b]$  with  $x_i = a + (b-a_i)/n$ , it is observed that at least for  $4 \leq n \leq 20$ ,  $\max_{a \leq x \leq b} |(x-x_0)(x-x_1) \cdots (x-x_n)| \approx (\frac{b-a}{3})^n$ .

The *best* choice for the  $x_i$  is the Chebyshev points:

$$x_i = \frac{a+b}{2} + \frac{b-a}{2} \cos \frac{2i+1}{2n+2} \pi, \quad i = 0, \dots, n-1. \quad (10.6)$$

In this case,  $\max_{a \leq x \leq b} |(x-x_0)(x-x_1) \cdots (x-x_n)| \equiv 2 \left(\frac{b-a}{4}\right)^n$ .

Figure 10.5 shows the Chebyshev and uniform spacing for degree nine polynomials.

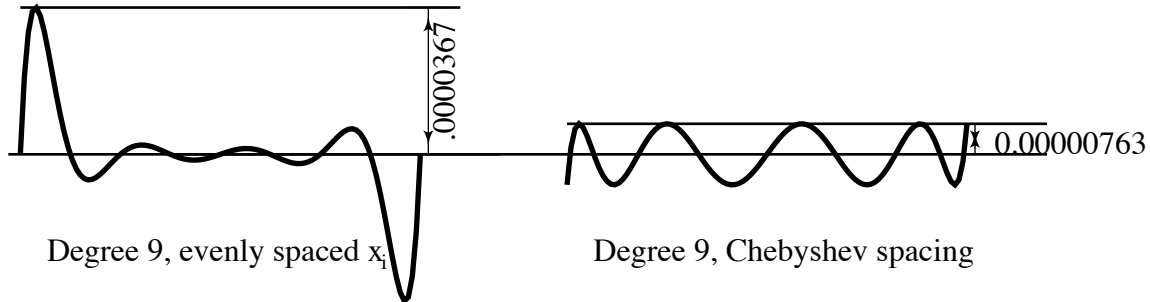


Figure 10.5: Two cases of  $(x - x_0)(x - x_1) \cdots (x - x_9)$  for  $0 \leq x \leq 1$ .

## 10.6 Interpolating Points and Normals

It is possible to use the method of undetermined coefficients to specify tangent vectors as well as interpolating points. For example, consider the cubic parametric curve

$$\mathbf{P}(t) = \mathbf{a}_0 + \mathbf{a}_1 t + \mathbf{a}_2 t^2 + \mathbf{a}_3 t^3.$$

The derivative of the curve is

$$\mathbf{P}'(t) = \mathbf{a}_1 + 2\mathbf{a}_2 t + 3\mathbf{a}_3 t^2$$

It should be obvious that we can now specify, for example,  $\mathbf{P}(t_1)$ ,  $\mathbf{P}(t_2)$ ,  $\mathbf{P}'(t_3)$  and  $\mathbf{P}'(t_4)$ . Alternatively, we can specify three interpolating points and one slope, or three slopes and one interpolating point, etc.

An important case of specifying points and tangent vectors is the Hermite blending functions. In this case, the curve is determined by specifying  $\mathbf{P}(0)$ ,  $\mathbf{P}(1)$ ,  $\mathbf{P}'(0)$ , and  $\mathbf{P}'(1)$  - that is, the two end points and the two end tangent vectors. Solving this case using the method of undetermined coefficients, we obtain the cubic Hermite curve:

$$\begin{aligned} \mathbf{P}(t) = & \mathbf{P}(0)(2t^3 - 3t^2 + 1) + \mathbf{P}(1)(-2t^3 + 3t^2) + \\ & \mathbf{P}'(0)(t^3 - 2t^2 + t) + \mathbf{P}'(1)(t^3 - t^2) \end{aligned}$$

