Chapter 6

POLYNOMIAL INTERPOLATION

A curve that passes through its control points is said to interpolate those points. We have seen that Bézier curves interpolate only two control points and uniform B-splines interpolate none of the control points. Overhauser curves interpolate all of the control points, but do so in a piecewise manner: A new cubic curve is defined between each pair of points.

It is possible to find a *single* degree n polynomial curve that will interpolate $n+1$ points. This chapter discusses several methods for doing so. Most of the discussion will focus on function interpolation: Find a polynomial function $p(t)$ that satisfies $p(t_0) = y_0$, $p(t_1) = y_1$, ..., $p(t_n) = y_n$. To find a *curve* that passes through $n+1$ points at $n+1$ specified parameter values, one merely finds independent polynomials $x(t)$ and $y(t)$ using these methods.

Fundamental Theorem of Algebra A polynomial of degree n has exactly n complex roots. A special case is the *zero polynomial* ($f(t) \equiv 0$) which is zero for *all* values of t.

Theorem Given $n+1$ pairs (t_i, y_i) $i = 0, \ldots, n$ with no two t_i the same, there exists a unique polynomial of at most degree n that satisfies $p(t_i) = y_i$, $i = 0, \ldots, n$.
1.5ex] **Proof** Suppose two different degree n polynomials $f(t)$ and $g(t)$ satisfy the interpolation conditions. Then, $h(t) = f(t) - g(t)$ is a polynomial of at most degree n that has $n+1$ the roots t_i, $i = 0, \ldots, n$. But this is not possible, according to the fundamental theorem of algebra.

6.1 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.

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

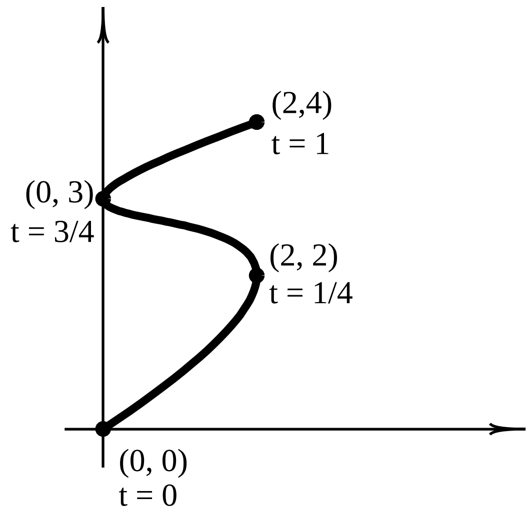

Figure 6.1: Interpolating Four Points

polynomials. For power basis polynomials, the parametric equations are of the form

$$x = a_0 + a_1 t + a_2 t^2 + a_3 t^3$$
$$y = b_0 + b_1 t + b_2 t^2 + b_3 t^3$$

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

$$0 = a_0 + a_1 0 + a_2 0^2 + a_3 0^3$$
$$2 = a_0 + a_1(\frac{1}{4}) + a_2(\frac{1}{4})^2 + a_3(\frac{1}{4})^3$$
$$0 = a_0 + a_1(\frac{3}{4}) + a_2(\frac{3}{4})^2 + a_3(\frac{3}{4})^3$$
$$2 = a_0 + a_1 1 + a_2 1^2 + a_3 1^3$$

In matrix form, we have

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

from which

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

Likewise for y, we solve the set of equations

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

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 $x = 3$, $t = \frac{3}{4}$ and again at $x = 3$, $t = 1$ to produce a set of equations:

$$0 = a_0(1-0)^3 + 3a_10(1-0)^2 + 3a_20^2(1-0) + a_30^3.$$

$$2 = a_0(1-\frac{1}{4})^3 + 3a_1\frac{1}{4}(1-\frac{1}{4})^2 + 3a_2\frac{1}{4}^2(1-\frac{1}{4}) + a_3\frac{1}{4}^3.$$

$$0 = a_0(1-\frac{3}{4})^3 + 3a_1\frac{3}{4}(1-\frac{3}{4})^2 + 3a_2\frac{3}{4}^2(1-\frac{3}{4}) + a_3\frac{3}{4}^3.$$

$$2 = a_0(1-1)^3 + 3a_11(1-1)^2 + 3a_21^2(1-1) + a_31^3.$$

In matrix form, we have

$$\begin{bmatrix} 1 & 0 & 0 & 0 \\ \frac{27}{64} & \frac{27}{64} & \frac{9}{64} & \frac{1}{64} \\ \frac{1}{64} & \frac{9}{64} & \frac{27}{64} & \frac{27}{64} \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{Bmatrix} a_0 \\ a_1 \\ a_2 \\ a_3 \end{Bmatrix} = \begin{Bmatrix} 0 \\ 2 \\ 3 \\ 3 \end{Bmatrix}.$$

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.

6.2 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

$$\mathbf{P}(t) = \mathbf{P}_0L_0^n(t) + \mathbf{P}_1L_1^n(t) + \dots + \mathbf{P}_nL_n^n(t).$$

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

$$L_i^n(t) =$$

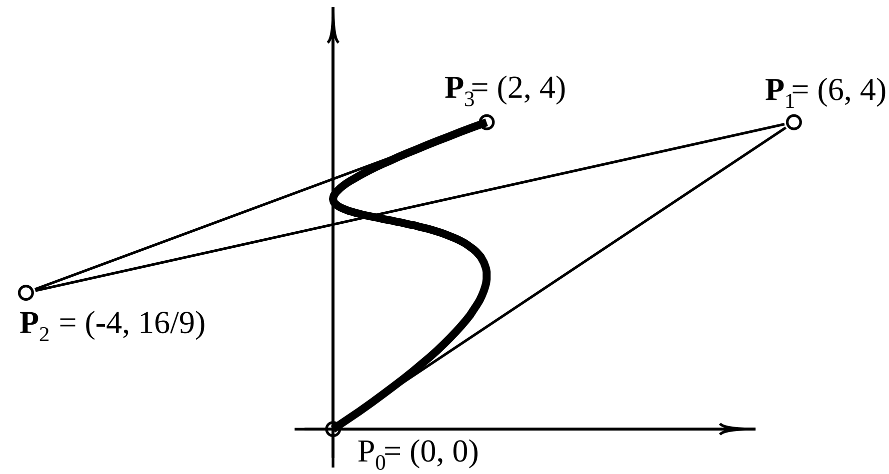

Figure 6.2: Interpolating Four Points

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

In the example from the previous section, we have

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

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

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

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

The interpolating curve is thus

$$\mathbf{P}(t) \;=\; \left\{ \begin{array}{c} x(t) \\ y(t) \end{array} \right\} \;=$$

$$\left\{ \begin{array}{c} 0 \\ 0 \end{array} \right\} L_0^3(t) \;+\; \left\{ \begin{array}{c} 2 \\ 2 \end{array} \right\} L_1^3(t) \;+\; \left\{ \begin{array}{c} 0 \\ 3 \end{array} \right\} L_2^3(t) \;+\; \left\{ \begin{array}{c} 2 \\ 4 \end{array} \right\} L_3^3(t)$$

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

6.3 Newton Polynomials

LIke Lagrange polynomials, Newton polynomials solve the problem of finding a degree n polynomial $p(t)$ that satisfies:

(6.1) $$p(t_0) = y_0; \quad p(t_1) = y_1; \quad \dots p(t_n) = y_n.$$

Newton polynomials are defined:

(6.2)
$$\begin{aligned} p_0(t) &= a_0 \\ p_1(t) &= a_0 + a_1(t - t_0) \\ p_2(t) &= a_0 + a_1(t - t_0) + a_2(t - t_0)(t - t_1) \\ &\cdots \end{aligned}$$
$$p_n(t) = p_{n-1}(t) + a_n(t - t_0)(t - t_1)\cdots(t - t_n)$$

The coefficients a_i are easily found by means of the divided differences $f_{i,\dots,k}$, which are defined as follows:

$$\begin{aligned} f_i &= f(t_i) \\ f_{i-1,i} &= \frac{f_i - f_{i-1}}{t_i - t_{i-1}} \\ &\cdots \\ f_{i,\dots,k} &= \frac{f_{i+1,\dots,k} - f_{i,\dots,k-1}}{t_k - t_i} \end{aligned}$$

Then, $a_0 = f_0$, $a_1 = f_{0,1}$, $a_2 = f_{0,1,2}$ etc.

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

t_0	f_0					
t_1	f_1	$f_{0,1}$				
t_2	f_2	$f_{1,2}$	$f_{0,1,2}$			
t_3	f_3	$f_{2,3}$	$f_{1,2,3}$	$f_{0,1,2,3}$		
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	
t_n	f_n	$f_{n-1,b}$	$\cdots$	$\cdots$	$\cdots$	$f_{0,1,\dots,n}$

6.4 Aitken's Lemma

Some variations on Newton's scheme have been proposed. Suppose polynomial $p_{i,\dots,k-1}(t)$ interpolates points $t_i, \dots, t_{k-1}$ and polynomial $p_{i+1,\dots,k}(t)$ interpolates points $t_{i+1}, \dots, t_k$. Then a polynomial that interpolates $t_i, \dots, t_k$ can easily be found:

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

6.5 Neville's Scheme

Imagine a parametric curve $\mathbf{P}(t) = \mathbf{P}_{0,1,\dots,n}(t)$ that interpolates $n+1$ points $\mathbf{P}_0, \dots, \mathbf{P}_n$ at parameter values $t_0, \dots, t_n$. Aitken's Lemma carries with it some nice geometric intuition that makes it possible to create a sort of geometric construction algorithm for evaluating $\mathbf{P}(t)$ at any parameter value.

The idea is reminiscent of using polar values to evaluate a B-spline. Denote by $\hat{t}$ the parameter value at which we want to evaluate the curve. Begin by labelling the interpolated points as $\mathbf{P}_0, \ldots, \mathbf{P}_n$. Then, compute the n points

$$\mathbf{P}_{i,i+1} = \frac{(t_{i+1} - \hat{t})\mathbf{P}_i - (t_i - \hat{t})\mathbf{P}_{i+1}}{t_{i+1} - t_i}, \quad i = 0, \ldots, n-1. \tag{6.4}$$

Then, compute the $n-1$ points

$$\mathbf{P}_{i,i+1,i+2} = \frac{(t_{i+2} - \hat{t})\mathbf{P}_{i,i+1} - (t_i - \hat{t})\mathbf{P}_{i+1.i+2}}{t_{i+2} - t_i}, \quad i = 0, \ldots, n-2. \tag{6.5}$$

Then, compute the $n-2$ points

$$\mathbf{P}_{i,i+1,i+2,i+3} = \frac{(t_{i+3} - \hat{t})\mathbf{P}_{i,i+1,i+2} - (t_i - \hat{t})\mathbf{P}_{i+1.i+2,i+3}}{t_{i+3} - t_i}, \quad i = 0, \ldots, n-3, \tag{6.6}$$

and so forth until $\mathbf{P}_{i,\ldots,n}$ is computed.

Note the close similarity of these computations to the affine map property of polar labels.

6.6 Error Bounds

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

$$f(x) - p(x) = \frac{f^{(n+1)}(\xi)}{(n+1)!}(x - x_0)(x - x_1)\cdots(x - x_n) \tag{6.7}$$

The value $f^{(n+1)}(\xi)$ is the $(n+1)^{th}$ derivative of $f(x)$ evaluated at some value ξ where $\min(x, x_0, \ldots, x_n) \leq \xi \leq \max(x, x_0, \ldots, x_n)$. If we can determine a bound on this derivative, then equation 6.7 can be used to provide an error bound on our approximation.

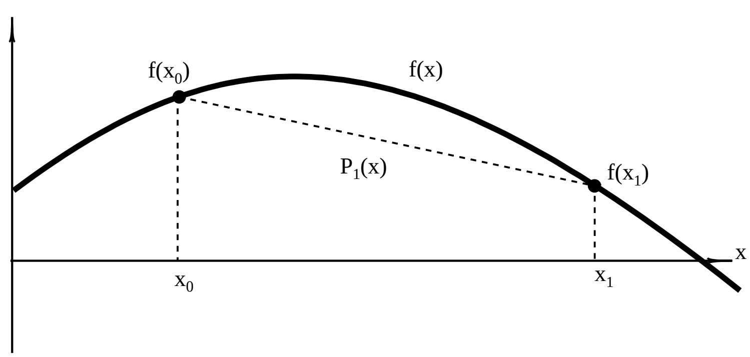

Figure 6.3: Error Bounds

A particularly useful case for equation 6.7 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

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

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

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

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

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

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 than ϵ. Figure 6.4 shows a cubic Bézier curve 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:

(6.8)
$$L_x = n(n-1)\max_{0\le i\le n-2}|x_{i+2} - 2x_{i+1} + x_i|$$

and

(6.9)
$$L_y = n(n-1)\max_{0\le i\le n-2}|y_{i+2} - 2y_{i+1} + y_i|,$$

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

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

the maximum error will be less than ϵ.

We can likewise determine how many times r a Bézier curve must be recursively bisected until each curve segment is assured to deviate from a straight line segment by no more than ϵ [17]:

(6.10)
$$r = \log_2\sqrt{\frac{\sqrt{L_x^2 + L_y^2}}{8\epsilon}}$$

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

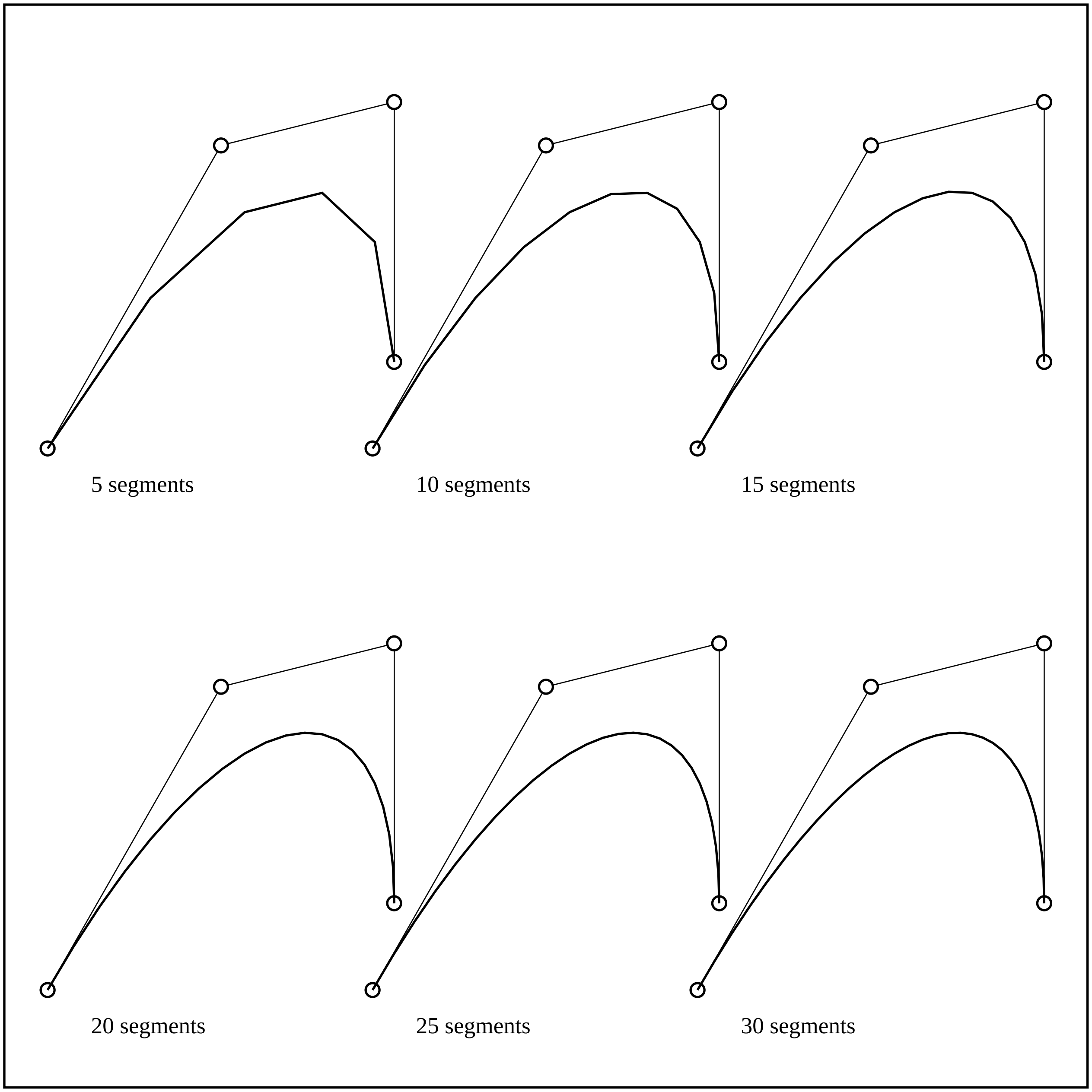

Figure 6.4: Piecewise linear approximation of a Bézier curve

6.6.1 Chebyshev Polynomials

If we wish to use a Lagrange polynomial of degree greater than one to perform approximation, we see from equation 6.7 that a wise choice of the interpolation points x_i can improve the approximation error. From equation 6.7,

$$|f(x) - p(x)| \leq \max_{a \leq x \leq b} \frac{f^{(n+1)}(x)}{(n+1)!} \max_{a \leq x \leq b} |(x - x_0)(x - x_1) \cdots (x - x_n)| \tag{6.11}$$

where $x, x_0, \ldots, x_n \in [a, b]$. 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 $4 \leq n \leq 20$, $\max_{a \leq x \leq b} |(x - x_0)(x - x_1) \cdots (x - x_n)| \approx \left(\frac{b-a}{3}\right)^n$.

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

$$x_i = \frac{a+b}{2} + \frac{b-a}{2} \cos \frac{2i+1}{2n+2}\pi, \quad i = 0, \ldots, n-1. \tag{6.12}$$

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

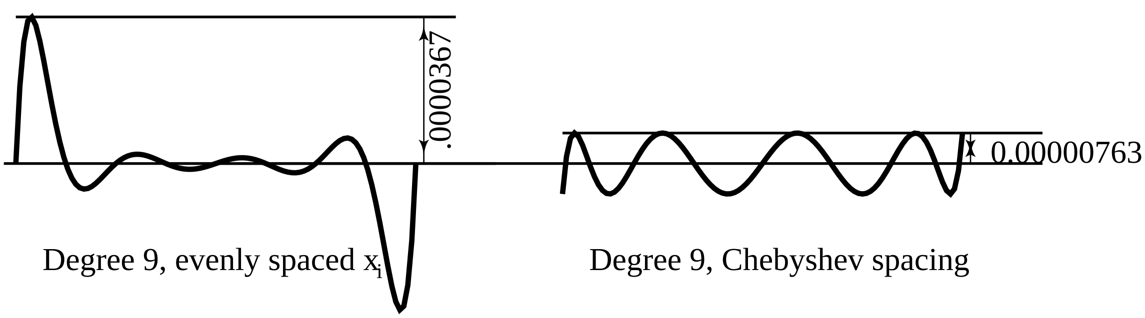

Figure 6.5: Two cases of $(x - x_0)(x - x_1) \cdots (x - x_9)$ for $0 \leq x \leq 1$.

Figure 6.5 shows the Chebyshev and uniform spacing for degree nine polynomials.

6.7 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 $\mathbf{P}'(t_4)$. Alternately, 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:

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