6
Bézier Approximation

Bézier methods for curves and surfaces are popular, are commonly used in practical work, and are described here in detail. Two approaches to the design of a Bézier curve are described, one using Bernstein polynomials and the other using the mediation operator. Both rectangular and triangular Bézier surface patches are discussed, with examples.

Historical Notes

Pierre Etienne Bézier (pronounced "Bez-yea" or "bez-ee-ay") was an applied mathematician with the French car manufacturer Renault. In the early 1960s, encouraged by his employer, he began searching for ways to automate the process of designing cars. His methods have been the basis of the modern field of Computer Aided Geometric Design (CAGD), a field with practical applications in many areas.

It is interesting to note that Paul de Faget de Casteljau, an applied mathematician with Citroën, was the first, in 1959, to develop the various Bézier methods but—because of the secretiveness of his employer—never published it (except for two internal technical memos that were discovered in 1975). This is why the entire field is named after the second person, Bézier, who developed it.

Bézier and de Casteljau did their work while working for car manufacturers. It is little known that Steven Anson Coons of MIT did most of his work on surfaces (around 1967) while a consultant for Ford. Another mathematician, William J. Gordon, has generalized the Coons surfaces, in 1969, as part of his work for General Motors research labs. In addition, airplane designer James Ferguson also came up with the same ideas for the construction of curves and surfaces. It seems that car and airplane manufacturers have been very innovative in the CAGD field. Detailed historical surveys of CAGD can be found in [Farin 04] and [Schumaker 81].

6.1 The Bézier Curve

The Bézier curve is a parametric curve $\mathbf{P}(t)$ that is a polynomial function of the parameter t. The degree of the polynomial depends on the number of points used to define the curve. The method employs *control points* and produces an approximating curve (note the title of this chapter). The curve does not pass through the interior points but is attracted by them (however, see Exercise 6.7 for an exception). It is as if the points exert a pull on the curve. Each point influences the direction of the curve by pulling it toward itself, and that influence is strongest when the curve gets nearest the point. Figure 6.1 shows some examples of cubic Bézier curves. Such a curve is defined by four points and is a cubic polynomial. Notice that one has a cusp and another one has a loop. The fact that the curve does not pass through the points implies that the points are not "set in stone" and can be moved. This makes it easy to edit, modify and reshape the curve, which is one reason for its popularity. The curve can also be edited by adding new points, or deleting points. These techniques are discussed in Sections 6.8 and 6.9, but they are cumbersome because the mathematical expression of the curve depends on the number of points, not just on the points themselves.

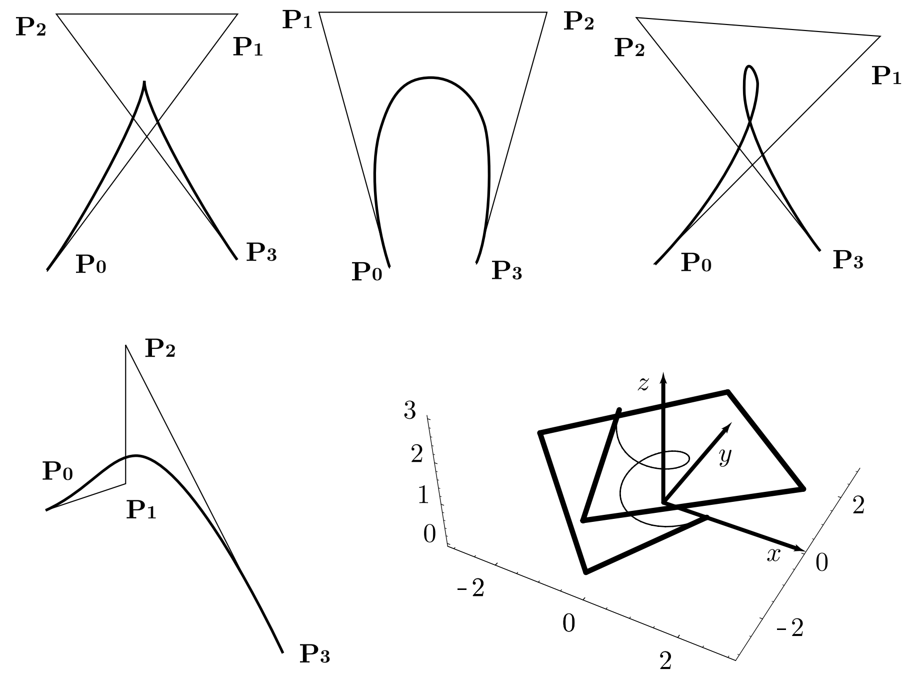


Figure 6.1: Four Plane Cubic and One Space Bézier Curves With Their Control Points and Polygons.

The *control polygon* of the Bézier curve is the polygon obtained when the control points are connected, in their natural order, with straight segments.

How does one go about deriving such a curve? We describe two approaches to the design—a weighted sum and a linear interpolation—and show that they are identical.

6.1.1 Pascal Triangle and the Binomial Theorem

The Pascal triangle and the binomial theorem are related because both employ the same numbers. The Pascal triangle is an infinite triangular matrix that's built from the edges inside

$$\begin{array}{ccccccccccc}
 & & & & & 1 & & & & & \\
 & & & & 1 & & 1 & & & & \\
 & & & 1 & & 2 & & 1 & & & \\
 & & 1 & & 3 & & 3 & & 1 & & \\
 & 1 & & 4 & & 6 & & 4 & & 1 & \\
1 & & 5 & & 10 & & 10 & & 5 & & 1 \\
 \dots & & & & & \dots & & & & & \dots
\end{array}$$

We first fill the left and right edges with ones, then compute each interior element as the sum of the two elements directly above it. As can be expected, it is not hard to obtain an explicit expression for the general element of the Pascal triangle. We first number the rows from 0 starting at the top, and the columns from 0 starting on the left. A general element is denoted by $\binom{i}{j}$. We then observe that the top two rows (corresponding to $i = 0, 1$) consist of 1's and that every other row can be obtained as the sum of its predecessor and a shifted version of its predecessor. For example,

$$\begin{array}{rccccc}
 & 1 & 3 & 3 & 1 & \\
+ & & 1 & 3 & 3 & 1 \\
\hline
 & 1 & 4 & 6 & 4 & 1
\end{array}$$

This shows that the elements of the triangle satisfy

$$\begin{aligned}
&\binom{i}{0} = \binom{i}{i} = 1, \qquad i = 0, 1, \dots, \\
&\binom{i}{j} = \binom{i-1}{j-1} + \binom{i-1}{j}, \qquad i = 2, 3, \dots, \quad j = 1, 2, \dots, (i-1).
\end{aligned}$$

From this it is easy to derive the explicit expression

$$\begin{aligned}
\binom{i}{j} &= \binom{i-1}{j-1} + \binom{i-1}{j} \\
&= \frac{(i-1)!}{(j-1)!(i-j)!} + \frac{(i-1)!}{j!(i-1-j)!} \\
&= \frac{j(i-1)!}{j!(i-j)!} + \frac{(i-j)(i-1)!}{j!(i-j)!} \\
&= \frac{i!}{j!(i-j)!}.
\end{aligned}$$

Thus, the general element of the Pascal triangle is the well-known *binomial coefficient*

$$\binom{i}{j} = \frac{i!}{j!(i-j)!}.$$

The binomial coefficient is one of Newton's many contributions to mathematics. His binomial theorem states that

$$(a+b)^n = \sum_{i=0}^{n} \binom{n}{i} a^i b^{n-i}. \tag{6.1}$$

This equation can be written in a symmetric way by denoting $j = n - i$. The result is

$$(a+b)^n = \sum_{i,j\geq 0}^{i+j=n} \frac{(i+j)!}{i!j!} a^i b^j, \tag{6.2}$$

from which we can easily guess the *trinomial theorem* (which is used in Section 6.23)

$$(a+b+c)^n = \sum_{i,j,k\geq 0}^{i+j+k=n} \frac{(i+j+k)!}{i!j!k!} a^i b^j c^k. \tag{6.3}$$

6.2 The Bernstein Form of the Bézier Curve

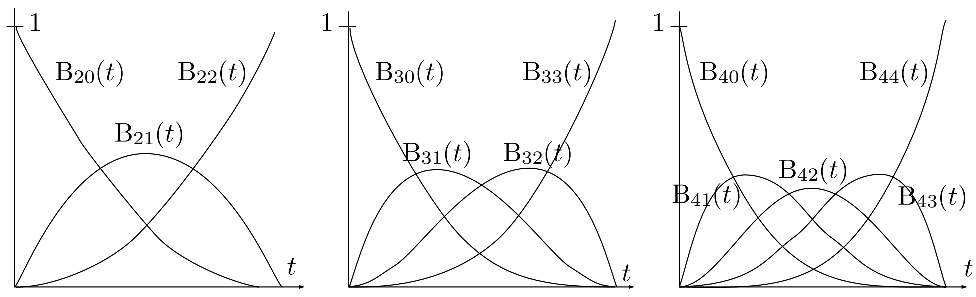
The first approach to the Bézier curve expresses it as a weighted sum of the points (with, of course, barycentric weights). Each control point is multiplied by a weight and the products are added. We denote the control points by $\mathbf{P}_0$, $\mathbf{P}_1, \ldots, \mathbf{P}_n$ (n is therefore defined as 1 less than the number of points) and the weights by B_i. The expression of weighted sum is

$$\mathbf{P}(t) = \sum_{i=0}^{n} \mathbf{P}_i B_i, \quad 0 \leq t \leq 1.$$

The result, $\mathbf{P}(t)$, depends on the parameter t. Since the points are given by the user, they are fixed, so it is the weights that must depend on t. We therefore denote them by $B_i(t)$. How should $B_i(t)$ behave as a function of t?

We first examine $B_0(t)$, the weight associated with the first point $\mathbf{P}_0$. We want that point to affect the curve mostly at the beginning, i.e., when t is close to 0. Thus, as t grows toward 1 (i.e., as the curve moves away from $\mathbf{P}_0$), $B_0(t)$ should drop down to 0. When $B_0(t) = 0$, the first point no longer influences the shape of the curve.

Next, we turn to $B_1(t)$. This weight function should start small, should have a maximum when the curve approaches the second point $\mathbf{P}_1$, and should then start dropping until it reaches zero. A natural question is: When (for what value of t) does the curve reach its closest approach to the second point? The answer is: It depends on the number of points. For three points (the case $n = 2$), the Bézier curve passes closest to the second point (the interior point) when $t = 0.5$. For four points, the curve is nearest the second point when $t = 1/3$. It is now clear that the weight functions must also depend on n and we denote them by $B_{n,i}(t)$. Hence, $B_{3,1}(t)$ should start at 0, have a maximum at $t = 1/3$, and go down to 0 from there. Figure 6.2 shows the desired behavior of $B_{n,i}(t)$



```
(* Just the base functions bern. Note how "pwr" handles 0^0 *)
Clear[pwr,bern];
pwr[x_,y_]:=If[x==0 && y==0, 1, x^y];
bern[n_,i_,t_]:=Binomial[n,i]pwr[t,i]pwr[1-t,n-i] (* t^i x (1-t)^(n-i) *)
Plot[Evaluate[Table[bern[5,i,t], {i,0,5}]], {t,0,1}, DefaultFont->{"cmr10", 10}];
```

Figure 6.2: The Bernstein Polynomials for $n = 2, 3, 4$.

for $n = 2$, 3, and 4. The five different weights $B_{4,i}(t)$ have their maxima at $t = 0$, 1/4, 1/2, 3/4, and 1.

The functions chosen by Bézier (and also by de Casteljau) were derived by the Russian mathematician Sergeĭ Natanovich Bernshteĭn in 1912, as part of his work on approximation theory (see Chapter 6 of [Davis 63]). They are known as the Bernstein polynomials and are defined by

$$B_{n,i}(t) = \binom{n}{i} t^i (1-t)^{n-i}, \quad \text{where} \quad \binom{n}{i} = \frac{n!}{i!(n-i)!} \tag{6.4}$$

are the binomial coefficients. These polynomials feature the desired behavior and have a few more useful properties that are discussed here. (In calculating the curve, we assume that the quantity 0^0, which is normally undefined, equals 1.)

The Bézier curve is now defined as

$$\mathbf{P}(t) = \sum_{i=0}^{n} \mathbf{P}_i B_{n,i}(t), \text{ where } B_{n,i}(t) = \binom{n}{i} t^i (1-t)^{n-i} \text{ and } 0 \le t \le 1. \tag{6.5}$$

Each control point (a pair or a triplet of coordinates) is multiplied by its weight, which is in the range $[0, 1]$. The weights act as *blending functions* that blend the contributions of the different points.

Here is *Mathematica* code to calculate and plot the Bernstein polynomials and the Bézier curve:

```
(* Just the base functions bern. Note how "pwr" handles 0^0 *)
Clear[pwr,bern,n,i,t]
pwr[x_,y_]:=If[x==0 && y==0, 1, x^y];
bern[n_,i_,t_]:=Binomial[n,i]pwr[t,i]pwr[1-t,n-i]
 (* t^i \[Times] (1-t)^(n-i) *)
Plot[Evaluate[Table[bern[5,i,t], {i,0,5}]], {t,0,1},
 DefaultFont->{"cmr10", 10}]
```

```
Clear[i,t,pnts,pwr,bern,bzCurve,g1,g2]; (* Cubic Bezier curve *)
(* either read points from file
pnts=ReadList["DataPoints",{Number,Number}]; *)
(* or enter them explicitly *)
pnts={{0,0},{.7,1},{.3,1},{1,0}}; (* 4 points for a cubic curve *)
pwr[x_,y_]:=If[x==0 && y==0, 1, x^y];
bern[n_,i_,t_]:=Binomial[n,i]pwr[t,i]pwr[1-t,n-i]
bzCurve[t_]:=Sum[pnts[[i+1]]bern[3,i,t], {i,0,3}]
g1=ListPlot[pnts, Prolog->AbsolutePointSize[4], PlotRange->All,
 AspectRatio->Automatic, DisplayFunction->Identity]
g2=ParametricPlot[bzCurve[t], {t,0,1}, DisplayFunction->Identity]
Show[g1,g2, DisplayFunction->$DisplayFunction]
```

Next is similar code for a three-dimensional Bézier curve. It was used to draw the space curve of Figure 6.1.

```
Clear[pnts,pwr,bern,bzCurve,g1,g2,g3]; (* General 3D Bezier curve *)
pnts={{1,0,0},{0,-3,0.5},{-3,0,0.75},{0,3,1},{3,0,1.5},{0,-3,1.75},{-1,0,2}};
n=Length[pnts]-1;
pwr[x_,y_]:=If[x==0 && y==0, 1, x^y];
bern[n_,i_,t_]:=Binomial[n,i]pwr[t,i]pwr[1-t,n-i] (* t^i x (1-t)^(n-i) *)
bzCurve[t_]:=Sum[pnts[[i+1]]bern[n,i,t], {i,0,n}];
g1=ParametricPlot3D[bzCurve[t], {t,0,1}, Compiled->False,
 DisplayFunction->Identity];
g2=Graphics3D[{AbsolutePointSize[2], Map[Point,pnts]}];
g3=Graphics3D[{AbsoluteThickness[2], (* control polygon *)
 Table[Line[{pnts[[j]],pnts[[j+1]]}], {j,1,n}]}];
g4=Graphics3D[{AbsoluteThickness[1.5], (* the coordinate axes *)
 Line[{{0,0,3},{0,0,0},{3,0,0},{0,0,0},{0,3,0}}]}];
Show[g1,g2,g3,g4, AspectRatio->Automatic, PlotRange->All, DefaultFont->{"cmr10", 10},
 Boxed->False, DisplayFunction->$DisplayFunction];
```

◇ **Exercise 6.1:** Design a heart-shaped Bézier curve based on nine control points.

When Bézier started searching for such functions in the early 1960s, he set the following requirements [Bézier 86]:

1. The functions should be such that the curve passes through the first and last control points.

2. The tangent to the curve at the start point should be $\mathbf{P}_1 - \mathbf{P}_0$, i.e., the curve should start at point $\mathbf{P}_0$ moving toward $\mathbf{P}_1$. A similar property should hold at the last point.

3. The same requirement is generalized for higher derivatives of the curve at the two extreme endpoints. Hence, $\mathbf{P}^{tt}(0)$ should depend only on the first point $\mathbf{P}_0$ and its two neighbors $\mathbf{P}_1$ and $\mathbf{P}_2$. In general, $\mathbf{P}^{(k)}(0)$ should only depend on $\mathbf{P}_0$ and its k neighbors $\mathbf{P}_1$ through $\mathbf{P}_k$. This feature provides complete control over the continuity at the joints between separate Bézier curve segments (Section 6.5).

4. The weight functions should be symmetric with respect to t and $(1 - t)$. This means that a reversal of the sequence of control points would not affect the shape of the curve.

5. The weights should be barycentric, to guarantee that the shape of the curve is independent of the coordinate system.

6. The entire curve lies within the convex hull of the set of control points. (See property 8 of Section 6.4 for a discussion of this point.)

The definition shown in Equation (6.5), using Bernstein polynomials as the weights, satisfies all these requirements. In particular, requirement 5 is proved when Equation (6.1) is written in the form $[t+(1-t)]^n = \cdots$ (see Equation (6.12) if you cannot figure this out). Following are the explicit expressions of these polynomials for $n = 2$, 3, and 4.

Example: For $n = 2$ (three control points), the weights are

$$\begin{aligned}
B_{2,0}(t) &= \tbinom{2}{0}t^0(1-t)^{2-0} = (1-t)^2,\\
B_{2,1}(t) &= \tbinom{2}{1}t^1(1-t)^{2-1} = 2t(1-t),\\
B_{2,2}(t) &= \tbinom{2}{2}t^2(1-t)^{2-2} = t^2,
\end{aligned}$$

and the curve is

$$\begin{aligned}
\mathbf{P}(t) &= (1-t)^2\mathbf{P}_0 + 2t(1-t)\mathbf{P}_1 + t^2\mathbf{P}_2\\
&= \left((1-t)^2, 2t(1-t), t^2\right)(\mathbf{P}_0, \mathbf{P}_1, \mathbf{P}_2)^T\\
&= (t^2, t, 1)\begin{pmatrix} 1 & -2 & 1\\ -2 & 2 & 0\\ 1 & 0 & 0\end{pmatrix}\begin{pmatrix}\mathbf{P}_0\\ \mathbf{P}_1\\ \mathbf{P}_2\end{pmatrix}. \qquad (6.6)
\end{aligned}$$

This is the quadratic Bézier curve.

◇ **Exercise 6.2:** Given three points $\mathbf{P}_1$, $\mathbf{P}_2$, and $\mathbf{P}_3$, calculate the parabola that goes from $\mathbf{P}_1$ to $\mathbf{P}_3$ and whose start and end tangent vectors point in directions $\mathbf{P}_2 - \mathbf{P}_1$ and $\mathbf{P}_3 - \mathbf{P}_2$, respectively.

In the special case $n = 3$, the four weight functions are

$$\begin{aligned}
B_{3,0}(t) &= \tbinom{3}{0}t^0(1-t)^{3-0} = (1-t)^3,\\
B_{3,1}(t) &= \tbinom{3}{1}t^1(1-t)^{3-1} = 3t(1-t)^2,\\
B_{3,2}(t) &= \tbinom{3}{2}t^2(1-t)^{3-2} = 3t^2(1-t),\\
B_{3,3}(t) &= \tbinom{3}{3}t^3(1-t)^{3-3} = t^3,
\end{aligned}$$

and the curve is

$$\begin{aligned}
\mathbf{P}(t) &= (1-t)^3\mathbf{P}_0 + 3t(1-t)^2\mathbf{P}_1 + 3t^2(1-t)\mathbf{P}_2 + t^3\mathbf{P}_3 \qquad (6.7)\\
&= \left[(1-t)^3, 3t(1-t)^2, 3t^2(1-t), t^3\right]\left[\mathbf{P}_0, \mathbf{P}_1, \mathbf{P}_2, \mathbf{P}_3\right]^T\\
&= \left[(1-3t+3t^2-t^3), (3t-6t^2+3t^3), (3t^2-3t^3), t^3\right]\left[\mathbf{P}_0, \mathbf{P}_1, \mathbf{P}_2, \mathbf{P}_3\right]^{\mathbf{T}}\\
&= (t^3, t^2, t, 1)\begin{pmatrix} -1 & 3 & -3 & 1\\ 3 & -6 & 3 & 0\\ -3 & 3 & 0 & 0\\ 1 & 0 & 0 & 0\end{pmatrix}\begin{pmatrix}\mathbf{P}_0\\ \mathbf{P}_1\\ \mathbf{P}_2\\ \mathbf{P}_3\end{pmatrix}. \qquad (6.8)
\end{aligned}$$

It is clear that $\mathbf{P}(t)$ is a cubic polynomial in t. It is the cubic Bézier curve. In general, the Bézier curve for points $\mathbf{P}_0$, $\mathbf{P}_1$,..., $\mathbf{P}_n$ is a polynomial of degree n.

◇ **Exercise 6.3:** Given the curve $\mathbf{P}(t) = (1 + t + t^2, t^3)$, find its control points.

◇ **Exercise 6.4:** The cubic curve of Equation (6.8) is drawn when the parameter t varies in the interval $[0, 1]$. Show how to substitute t with a new parameter u such that the curve will be drawn when $-1 \le u \le +1$.

◇ **Exercise 6.5:** Calculate the Bernstein polynomials for $n = 4$.

It can be proved by induction that the general, $(n + 1)$-point Bézier curve can be represented by

$$\mathbf{P}(t) = (t^n, t^{n-1}, \ldots, t, 1)\mathbf{N}\begin{pmatrix} \mathbf{P}_0 \\ \mathbf{P}_1 \\ \vdots \\ \mathbf{P}_{n-1} \\ \mathbf{P}_n \end{pmatrix} = \mathbf{T}(t)\cdot\mathbf{N}\cdot\mathbf{P}, \tag{6.9}$$

where

$$\mathbf{N} = \begin{pmatrix} \binom{n}{0}\binom{n}{n}(-1)^n & \binom{n}{1}\binom{n-1}{n-1}(-1)^{n-1} & \cdots & \binom{n}{n}\binom{n-n}{n-n}(-1)^0 \\ \binom{n}{0}\binom{n}{n-1}(-1)^{n-1} & \binom{n}{1}\binom{n-1}{n-2}(-1)^{n-2} & \cdots & 0 \\ \vdots & \vdots & \cdots & 0 \\ \binom{n}{0}\binom{n}{1}(-1)^1 & \binom{n}{1}\binom{n-1}{0}(-1)^0 & \cdots & 0 \\ \binom{n}{0}\binom{n}{0}(-1)^0 & 0 & \cdots & 0 \end{pmatrix}. \tag{6.10}$$

Matrix $\mathbf{N}$ is symmetric and its elements below the second diagonal are all zeros. Its determinant therefore equals (up to a sign) the product of the diagonal elements, which are all nonzero. A nonzero determinant implies a nonsingular matrix. Thus, matrix $\mathbf{N}$ always has an inverse. $\mathbf{N}$ can also be written as the product $\mathbf{AB}$, where

$$\mathbf{A} = \begin{pmatrix} \binom{n}{n}(-1)^n & \binom{n}{1}\binom{n-1}{n-1}(-1)^{n-1} & \cdots & \binom{n}{n}\binom{n-n}{n-n}(-1)^0 \\ \binom{n}{n-1}(-1)^{n-1} & \binom{n}{1}\binom{n-1}{n-2}(-1)^{n-2} & \cdots & 0 \\ \vdots & \vdots & \cdots & 0 \\ \binom{n}{1}(-1)^1 & \binom{n}{1}\binom{n-1}{0}(-1)^0 & \cdots & 0 \\ \binom{n}{0}(-1)^0 & 0 & \cdots & 0 \end{pmatrix}$$

and

$$\mathbf{B} = \begin{pmatrix} \binom{n}{0} & 0 & \cdots & 0 \\ 0 & \binom{n}{1} & \cdots & 0 \\ \vdots & & \ddots & \vdots \\ 0 & 0 & \cdots & \binom{n}{n} \end{pmatrix}.$$

Figure 6.3 shows the Bézier $\mathbf{N}$ matrices for $n = 1, 2, \ldots, 7$.

◇ **Exercise 6.6:** Calculate the Bézier curve for the case $n = 1$ (two control points). What kind of a curve is it?

$$\mathbf{N}_1 = \begin{pmatrix} -1 & 1 \\ 1 & 0 \end{pmatrix},$$

$$\mathbf{N}_2 = \begin{pmatrix} 1 & -2 & 1 \\ -2 & 2 & 0 \\ 1 & 0 & 0 \end{pmatrix},$$

$$\mathbf{N}_3 = \begin{pmatrix} -1 & 3 & -3 & 1 \\ 3 & -6 & 3 & 0 \\ -3 & 3 & 0 & 0 \\ 1 & 0 & 0 & 0 \end{pmatrix},$$

$$\mathbf{N}_4 = \begin{pmatrix} 1 & -4 & 6 & -4 & 1 \\ -4 & 12 & -12 & 4 & 0 \\ 6 & -12 & 6 & 0 & 0 \\ -4 & 4 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 \end{pmatrix},$$

$$\mathbf{N}_5 = \begin{pmatrix} -1 & 5 & -10 & 10 & -5 & 1 \\ 5 & -20 & 30 & -20 & 5 & 0 \\ -10 & 30 & -30 & 10 & 0 & 0 \\ 10 & -20 & 10 & 0 & 0 & 0 \\ -5 & 5 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 \end{pmatrix},$$

$$\mathbf{N}_6 = \begin{pmatrix} 1 & -6 & 15 & -20 & 15 & -6 & 1 \\ -6 & 30 & -60 & 60 & -30 & 6 & 0 \\ 15 & -60 & 90 & -60 & 15 & 0 & 0 \\ -20 & 60 & -60 & 20 & 0 & 0 & 0 \\ 15 & -30 & 15 & 0 & 0 & 0 & 0 \\ -6 & 6 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 \end{pmatrix},$$

$$\mathbf{N}_7 = \begin{pmatrix} -1 & 7 & -21 & 35 & -35 & 21 & -7 & 1 \\ 7 & -42 & 105 & -140 & 105 & -42 & 7 & 0 \\ -21 & 105 & -210 & 210 & -105 & 21 & 0 & 0 \\ 35 & -140 & 210 & -140 & 35 & 0 & 0 & 0 \\ -35 & 105 & -105 & 35 & 0 & 0 & 0 & 0 \\ 21 & -42 & 21 & 0 & 0 & 0 & 0 & 0 \\ -7 & 7 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{pmatrix}.$$

Figure 6.3: The First Seven Bézier Basis Matrices.

◇ **Exercise 6.7:** Generally, the Bézier curve passes through the first and last control points, but not through the intermediate points. Consider the case of three points $\mathbf{P}_0$, $\mathbf{P}_1$, and $\mathbf{P}_2$ on a straight line. Intuitively, it seems that the curve will be a straight line and would therefore pass through the interior point $\mathbf{P}_1$. Is that so?

The Bézier curve can also be represented in a very compact and elegant way as $\mathbf{P}(t) = (1-t+tE)^n\mathbf{P}_0$, where E is the shift operator defined by $E\mathbf{P}_i = \mathbf{P}_{i+1}$ (i.e., applying E to point $\mathbf{P}_i$ produces point $\mathbf{P}_{i+1}$). The definition of E implies $E\mathbf{P}_0 = \mathbf{P}_1$, $E^2\mathbf{P}_0 = \mathbf{P}_2$, and $E^i\mathbf{P}_0 = \mathbf{P}_i$.

The Bézier curve can now be written

$$\begin{aligned}\mathbf{P}(t) &= \sum_{i=0}^{n}\binom{n}{i}t^i(1-t)^{n-i}\mathbf{P}_i = \sum_{i=0}^{n}\binom{n}{i}t^i(1-t)^{n-i}E^i\mathbf{P}_0\\ &= \sum_{i=0}^{n}\binom{n}{i}(tE)^i(1-t)^{n-i}\mathbf{P}_0 = \bigl(tE+(1-t)\bigr)^n\mathbf{P}_0,\end{aligned}$$

where the last step is an application of the binomial theorem, Equation (6.1).

Example: For $n=1$, this representation amounts to

$$\mathbf{P}(t) = (1-t+tE)\mathbf{P}_0 = \mathbf{P}_0(1-t)+\mathbf{P}_1 t.$$

For $n=2$, we get

$$\begin{aligned}\mathbf{P}(t) &= (1-t+tE)^2\mathbf{P}_0\\ &= (1-t+tE-t+t^2-t^2E+tE-t^2E+t^2E^2)\mathbf{P}_0\\ &= \mathbf{P}_0(1-2t+t^2)+\mathbf{P}_1(2t-2t^2)+\mathbf{P}_2t^2\\ &= \mathbf{P}_0(1+t)^2+\mathbf{P}_1 2t(1-t)+\mathbf{P}_2t^2.\end{aligned}$$

Given $n+1$ control points $\mathbf{P}_0$ through $\mathbf{P}_n$, we can represent the Bézier curve for the points by $\mathbf{P}_n^{(n)}(t)$, where the quantity $\mathbf{P}_i^{(j)}(t)$ is defined recursively by

$$\mathbf{P}_i^{(j)}(t) = \begin{cases}(1-t)\mathbf{P}_{i-1}^{(j-1)}(t)+t\mathbf{P}_i^{(j-1)}(t), & \text{for } j>0,\\ \mathbf{P}_i, & \text{for } j=0.\end{cases} \tag{6.11}$$

The following examples show how the definition above is used to generate the quantities $\mathbf{P}_i^{(j)}(t)$ and why $\mathbf{P}_n^{(n)}(t)$ is the degree-n curve:

$$\begin{aligned}&\mathbf{P}_0^{(0)}(t) = \mathbf{P}_0,\quad \mathbf{P}_1^{(0)}(t) = \mathbf{P}_1,\quad \mathbf{P}_2^{(0)}(t) = \mathbf{P}_2,\ldots,\mathbf{P}_n^{(0)}(t) = \mathbf{P}_n,\\ &\mathbf{P}_1^{(1)}(t) = (1-t)\mathbf{P}_0^{(0)}(t)+t\mathbf{P}_1^{(0)}(t) = (1-t)\mathbf{P}_0+t\mathbf{P}_1,\\ &\mathbf{P}_2^{(2)}(t) = (1-t)\mathbf{P}_1^{(1)}(t)+t\mathbf{P}_2^{(1)}(t)\\ &\qquad = (1-t)\bigl((1-t)\mathbf{P}_0+t\mathbf{P}_1\bigr)+t\bigl((1-t)\mathbf{P}_1+t\mathbf{P}_2\bigr)\\ &\qquad = (1-t)^2\mathbf{P}_0+2t(1-t)\mathbf{P}_1+t^2\mathbf{P}_2,\\ &\mathbf{P}_3^{(3)}(t) = (1-t)\mathbf{P}_2^{(2)}(t)+t\mathbf{P}_3^{(2)}(t)\end{aligned}$$

$$
\begin{aligned}
&= (1-t)\big((1-t)\mathbf{P}_1^{(1)}(t) + t\mathbf{P}_2^{(1)}(t)\big) + t\big((1-t)\mathbf{P}_2^{(1)}(t) + t\mathbf{P}_3^{(1)}(t)\big)\\
&= (1-t)^2\mathbf{P}_1^{(1)}(t) + 2t(1-t)\mathbf{P}_2^{(1)}(t) + t^2\mathbf{P}_3^{(1)}(t)\\
&= (1-t)^2\big((1-t)\mathbf{P}_0 + t\mathbf{P}_1\big) + 2t(1-t)\big((1-t)\mathbf{P}_1 + t\mathbf{P}_2\big)\\
&\quad + t^2\big((1-t)\mathbf{P}_2 + t\mathbf{P}_3\big)\\
&= (1-t)^3\mathbf{P}_0 + 3t(1-t)^2\mathbf{P}_1 + 3t^2(1-t)\mathbf{P}_2 + t^3\mathbf{P}_3.
\end{aligned}
$$

6.3 Fast Calculation of the Curve

Calculating the Bézier curve is straightforward but slow. However, with a little thinking, it can be speeded up considerably, a feature that makes this curve very useful in practice. This section discusses three methods.

Method 1: We notice the following:

- The calculation involves the binomials $\binom{n}{i}$ for $i = 0, 1, \ldots, n$, which, in turn, require the factorials $0!, 1!, \ldots, n!$. The factorials can be precalculated once (each one from its predecessor) and stored in a table. They can then be used to calculate all the necessary binomials and those can also be stored in a table.

- The calculation involves terms of the form t^i for $i = 0, 1, \ldots, n$ and for many t values in the interval $[0, 1]$. These can also be precalculated and stored in a two-dimensional table where they can be accessed later, using t and i as indexes. This has the advantage that the values of $(1-t)^{n-i}$ can be read from the same table (using $1-t$ and $n-i$ as row and column indexes).

The calculation now reduces to a sum where each term is a product of four quantities, one control point and three numbers from tables. Instead of computing

$$\sum_{i=0}^{n} \binom{n}{i} t^i (1-t)^{n-i}\mathbf{P}_i,$$

we need to compute the simple sum

$$\sum_{i=0}^{n} \text{Table}_1[i,n]\cdot \text{Table}_2[t,i]\cdot \text{Table}_2[1-t,n-i]\cdot \mathbf{P}_i.$$

The parameter t is a real number that varies from 0 to 1, so a practical implementation of this method should use an integer T related to t. For example, if we increment t in 100 steps, then T should be the integer $100t$.

Method 2: Once n is known, each of the $n+1$ Bernstein polynomials $B_{n,i}(t)$, $i = 0, 1, \ldots, n$, can be precalculated for all the necessary values of t and stored in a table. The curve can now be calculated as the sum

$$\sum_{i=0}^{n} \text{Table}[t,i]\mathbf{P}_i,$$

indicating that each point on the computed curve requires $n+1$ table lookups, $n+1$ multiplications, and n additions. Again, an integer index T should be used instead of t.

Method 3: Use *forward differences* in combination with the Taylor series representation, to speed up the calculation significantly. The Bézier curve, which we denote by $\mathbf{B}(t)$, is drawn pixel by pixel in a loop where t is incremented from 0 to 1 in fixed, small steps of Δt. The principle of forward differences (Section 1.5.1) is to find a quantity $\mathbf{dB}$ such that $\mathbf{B}(t+\Delta t)=\mathbf{B}(t)+\mathbf{dB}$ for any value of t. If such a $\mathbf{dB}$ can be found, then it is enough to calculate $\mathbf{B}(0)$ (which, as we know, is simply $\mathbf{P}_0$) and use forward differences to calculate

$$\begin{aligned}\mathbf{B}(0+\Delta t)&=\mathbf{B}(0)+\mathbf{dB},\\ \mathbf{B}(2\Delta t)=\mathbf{B}(\Delta t)+\mathbf{dB}&=\mathbf{B}(0)+2\mathbf{dB},\end{aligned}$$

and, in general,

$$\mathbf{B}(i\Delta t)=\mathbf{B}\big((i-1)\Delta t\big)+\mathbf{dB}=\mathbf{B}(0)+i\,\mathbf{dB}.$$

The point is that $\mathbf{dB}$ should not depend on t. If $\mathbf{dB}$ turns out to depend on t, then as we advance t from 0 to 1, we would have to use different values of $\mathbf{dB}$, slowing down the calculations. The fastest way to calculate the curve is to precalculate $\mathbf{dB}$ before the loop starts and to repeatedly add this precalculated value to $\mathbf{B}(t)$ inside the loop.

We calculate $\mathbf{dB}$ by using the *Taylor series* representation of the Bézier curve. In general, the Taylor series representation of a function $f(t)$ at a point $f(t+\Delta t)$ is the infinite sum

$$f(t+\Delta t)=f(t)+f'(t)\Delta t+\frac{f''(t)\Delta^2 t}{2!}+\frac{f'''(t)\Delta^3 t}{3!}+\cdots.$$

In order to avoid dealing with an infinite sum, we limit our discussion to cubic Bézier curves. These are the most common Bézier curves and are used by many popular graphics applications. They are defined by four control points and are given by Equations (6.7) and (6.8):

$$\begin{aligned}\mathbf{B}(t)&=(1-t)^3\mathbf{P}_0+3t(1-t)^2\mathbf{P}_1+3t^2(1-t)\mathbf{P}_2+t^3\mathbf{P}_3\\ &=(t^3,t^2,t,1)\begin{pmatrix}-1&3&-3&1\\3&-6&3&0\\-3&3&0&0\\1&0&0&0\end{pmatrix}\begin{pmatrix}\mathbf{P}_0\\\mathbf{P}_1\\\mathbf{P}_2\\\mathbf{P}_3\end{pmatrix}.\end{aligned}$$

These curves are cubic polynomials in t, implying that only their first three derivatives are nonzero. In order to simplify the calculation of their derivatives, we need to express these curves in the form $\mathbf{B}(t)=\mathbf{a}t^3+\mathbf{b}t^2+\mathbf{c}t+\mathbf{d}$ [Equation (3.1)]. This is done by

$$\begin{aligned}\mathbf{B}(t)&=(1-t)^3\mathbf{P}_0+3t(1-t)^2\mathbf{P}_1+3t^2(1-t)\mathbf{P}_2+t^3\mathbf{P}_3\\ &=\big(3(\mathbf{P}_1-\mathbf{P}_2)-\mathbf{P}_0+\mathbf{P}_3\big)t^3+\big(3(\mathbf{P}_0+\mathbf{P}_2)-6\mathbf{P}_1\big)t^2+3(\mathbf{P}_1-\mathbf{P}_0)t+\mathbf{P}_0\\ &=\mathbf{a}t^3+\mathbf{b}t^2+\mathbf{c}t+\mathbf{d},\end{aligned}$$

so $\mathbf{a} = 3(\mathbf{P}_1 - \mathbf{P}_2) - \mathbf{P}_0 + \mathbf{P}_3$, $\mathbf{b} = 3(\mathbf{P}_0 + \mathbf{P}_2) - 6\mathbf{P}_1$, $\mathbf{c} = 3(\mathbf{P}_1 - \mathbf{P}_0)$, and $\mathbf{d} = \mathbf{P}_0$. These relations can also be expressed in matrix notation

$$\begin{pmatrix}\mathbf{a}\\ \mathbf{b}\\ \mathbf{c}\\ \mathbf{d}\end{pmatrix} = \begin{pmatrix}-1 & 3 & -3 & 1\\ 3 & -6 & 3 & 0\\ -3 & 3 & 0 & 0\\ 1 & 0 & 0 & 0\end{pmatrix}\begin{pmatrix}\mathbf{P}_0\\ \mathbf{P}_1\\ \mathbf{P}_2\\ \mathbf{P}_3\end{pmatrix}.$$

The curve is now easy to differentiate

$$\mathbf{B}^t(t) = 3\mathbf{a}t^2 + 2\mathbf{b}t + \mathbf{c}, \quad \mathbf{B}^{tt}(t) = 6\mathbf{a}t + 2\mathbf{b}, \quad \mathbf{B}^{ttt}(t) = 6\mathbf{a};$$

and the Taylor series representation yields

$$\begin{aligned}\mathbf{dB} &= \mathbf{B}(t+\Delta t) - \mathbf{B}(t)\\ &= \mathbf{B}^t(t)\Delta t + \frac{\mathbf{B}^{tt}(t)\Delta^2 t}{2} + \frac{\mathbf{B}^{ttt}(t)\Delta^3 t}{6}\\ &= 3\mathbf{a}\,t^2\Delta t + 2\mathbf{b}\,t\Delta t + \mathbf{c}\Delta t + 3\mathbf{a}\,t\Delta^2 t + \mathbf{b}\Delta^2 t + \mathbf{a}\Delta^3 t.\end{aligned}$$

This seems like a failure since the value obtained for $\mathbf{dB}$ is a function of t (it should be denoted by $\mathbf{dB}(t)$ instead of just $\mathbf{dB}$) and is also slow to calculate. However, the original cubic curve $\mathbf{B}(t)$ is a degree-3 polynomial in t, whereas $\mathbf{dB}(t)$ is only a degree-2 polynomial. This suggests a way out of our dilemma. We can try to express $\mathbf{dB}(t)$ by means of the Taylor series, similar to what we did with the original curve $\mathbf{B}(t)$. This should result in a forward difference $\mathbf{ddB}(t)$ that's a polynomial of degree 1 in t. The quantity $\mathbf{ddB}(t)$ can, in turn, be represented by another Taylor series to produce a forward difference $\mathbf{dddB}$ that's a degree-0 polynomial, i.e., a constant. Once we do that, we will end up with an algorithm of the form

```
precalculate certain quantities;
$\mathbf{B} = \mathbf{P}_0$;
for t:=0 to 1 step $\Delta t$ do
PlotPixel(B);
B:=B+dB; dB:=dB+ddB; ddB:=ddB+dddB;
endfor;
```

The quantity $\mathbf{ddB}(t)$ is obtained by

$$\mathbf{dB}(t+\Delta t) = \mathbf{dB}(t) + \mathbf{ddB}(t) = \mathbf{dB}(t) + \mathbf{dB}^t(t)\Delta t + \frac{\mathbf{dB}(t)^{tt}\Delta^2 t}{2},$$

yielding

$$\begin{aligned}\mathbf{ddB}(t) &= \mathbf{dB}^t(t)\Delta t + \frac{\mathbf{dB}(t)^{tt}\Delta^2 t}{2}\\ &= (6\mathbf{a}\,t\Delta t + 2\mathbf{b}\Delta t + 3\mathbf{a}\Delta^2 t)\Delta t + \frac{6\mathbf{a}\Delta t\Delta^2 t}{2}\\ &= 6\mathbf{a}\,t\Delta^2 t + 2\mathbf{b}\Delta^2 t + 6\mathbf{a}\Delta^3 t.\end{aligned}$$

Finally, the constant **dddB** is similarly obtained by

$$\mathbf{ddB}(t+\Delta t)=\mathbf{ddB}(t)+\mathbf{dddB}=\mathbf{ddB}(t)+\mathbf{ddB}^t(t)\Delta t,$$

yielding $\mathbf{dddB}=\mathbf{ddB}^t(t)\Delta t=6\mathbf{a}\Delta^3 t$.

The four quantities involved in the calculation of the curve are therefore

$$\begin{aligned}
\mathbf{B}(t)&=\mathbf{a}t^3+\mathbf{b}t^2+\mathbf{c}t+\mathbf{d},\\
\mathbf{dB}(t)&=3\mathbf{a}\,t^2\Delta t+2\mathbf{b}\,t\Delta t+\mathbf{c}\Delta t+3\mathbf{a}\,t\Delta^2 t+\mathbf{b}\Delta^2 t+\mathbf{a}\Delta^3 t,\\
\mathbf{ddB}(t)&=6\mathbf{a}\,t\Delta^2 t+2\mathbf{b}\Delta^2 t+6\mathbf{a}\Delta^3 t,\\
\mathbf{dddB}&=6\mathbf{a}\Delta^3 t.
\end{aligned}$$

They all have to be calculated at $t=0$, as functions of the four control points $\mathbf{P}_i$, before the loop starts:

$$\begin{aligned}
\mathbf{B}(0)&=\mathbf{d}=\mathbf{P}_0,\\
\mathbf{dB}(0)&=\mathbf{c}\Delta t+\mathbf{b}\Delta^2 t+\mathbf{a}\Delta^3 t\\
&=3\Delta t(\mathbf{P}_1-\mathbf{P}_0)+\Delta^2 t\big(3(\mathbf{P}_0+\mathbf{P}_2)-6\mathbf{P}_1\big)\\
&\quad+\Delta^3 t\big(3(\mathbf{P}_1-\mathbf{P}_2)-\mathbf{P}_0+\mathbf{P}_3\big)\\
&=3\Delta t(\mathbf{P}_1-\mathbf{P}_0)+3\Delta^2 t(\mathbf{P}_0-2\mathbf{P}_1+\mathbf{P}_2)\\
&\quad+\Delta^3 t\big(3(\mathbf{P}_1-\mathbf{P}_2)-\mathbf{P}_0+\mathbf{P}_3\big),\\
\mathbf{ddB}(0)&=2\mathbf{b}\Delta^2 t+6\mathbf{a}\Delta^3 t\\
&=2\Delta^2 t\big(3(\mathbf{P}_0+\mathbf{P}_2)-6\mathbf{P}_1\big)+6\Delta^3 t\big(3(\mathbf{P}_1-\mathbf{P}_2)-\mathbf{P}_0+\mathbf{P}_3\big)\\
&=6\Delta^2 t(\mathbf{P}_0-2\mathbf{P}_1+\mathbf{P}_2)+6\Delta^3 t\big(3(\mathbf{P}_1-\mathbf{P}_2)-\mathbf{P}_0+\mathbf{P}_3\big),\\
\mathbf{dddB}&=6\mathbf{a}\Delta^3 t=6\Delta^3 t\big(3(\mathbf{P}_1-\mathbf{P}_2)-\mathbf{P}_0+\mathbf{P}_3\big).
\end{aligned}$$

The above relations can be expressed in matrix notation as follows:

$$\begin{aligned}
\begin{pmatrix}\mathbf{dddB}\\ \mathbf{ddB}(0)\\ \mathbf{dB}(0)\\ \mathbf{B}(0)\end{pmatrix}
&=\begin{pmatrix}6&0&0&0\\6&2&0&0\\1&1&1&0\\0&0&0&1\end{pmatrix}
\begin{pmatrix}\Delta^3 t&0&0&0\\0&\Delta^2 t&0&0\\0&0&\Delta t&0\\0&0&0&1\end{pmatrix}
\begin{pmatrix}\mathbf{a}\\ \mathbf{b}\\ \mathbf{c}\\ \mathbf{d}\end{pmatrix}\\
&=\begin{pmatrix}6&0&0&0\\6&2&0&0\\1&1&1&0\\0&0&0&1\end{pmatrix}
\begin{pmatrix}\Delta^3 t&0&0&0\\0&\Delta^2 t&0&0\\0&0&\Delta t&0\\0&0&0&1\end{pmatrix}
\begin{pmatrix}-1&3&-3&1\\3&-6&3&0\\-3&3&0&0\\1&0&0&0\end{pmatrix}
\begin{pmatrix}\mathbf{P}_0\\ \mathbf{P}_1\\ \mathbf{P}_2\\ \mathbf{P}_3\end{pmatrix}\\
&=\begin{pmatrix}-6\Delta^3 t&18\Delta^3 t&-18\Delta^3 t&6\Delta^3 t\\
6\Delta^2 t-6\Delta^3 t&-12\Delta^2 t+18\Delta^3 t&6\Delta^2 t-18\Delta^3 t&6\Delta^3 t\\
3\Delta^2 t-\Delta^3 t-3\Delta^t&-6\Delta^2 t+3\Delta^3 t+3\Delta^t&3\Delta^2 t-3\Delta^3 t&\Delta^3 t\\
1&0&0&0\end{pmatrix}
\begin{pmatrix}\mathbf{P}_0\\ \mathbf{P}_1\\ \mathbf{P}_2\\ \mathbf{P}_3\end{pmatrix}
\end{aligned}$$

$$= \mathbf{Q}\begin{pmatrix}\mathbf{P}_0\\ \mathbf{P}_1\\ \mathbf{P}_2\\ \mathbf{P}_3\end{pmatrix},$$

where $\mathbf{Q}$ is a 4×4 matrix that can be calculated once Δt is known.

A detailed examination of the above expressions shows that the following quantities have to be precalculated: $3\Delta t$, $3\Delta^2 t$, $\Delta^3 t$, $6\Delta^2 t$, $6\Delta^3 t$, $\mathbf{P}_0-2\mathbf{P}_1+\mathbf{P}_2$, and $3(\mathbf{P}_1-\mathbf{P}_2)-\mathbf{P}_0+\mathbf{P}_3$. We therefore end up with the simple, fast algorithm shown in Figure 6.4. For those interested in a quick test, the corresponding *Mathematica* code is also included.

```
Q1:=3$\Delta t$;
Q2:=Q1$\times\Delta t$; // $3\Delta^2 t$
Q3:=$\Delta^3 t$;
Q4:=2Q2; // $6\Delta^2 t$
Q5:=6Q3; // $6\Delta^3 t$
Q6:=$\mathbf{P}_0-2\mathbf{P}_1+\mathbf{P}_2$;
Q7:=$3(\mathbf{P}_1-\mathbf{P}_2)-\mathbf{P}_0+\mathbf{P}_3$;
B:=$\mathbf{P}_0$;
dB:=$(\mathbf{P}_1-\mathbf{P}_0)$Q1+Q6$\times$Q2+Q7$\times$Q3;
ddB:=Q6$\times$Q4+Q7$\times$Q5;
dddB:=Q7$\times$Q5;
for t:=0 to 1 step $\Delta t$ do
Pixel(B);
B:=B+dB; dB:=dB+ddB; ddB:=ddB+dddB;
endfor;
```

```
n=3; Clear[q1,q2,q3,q4,q5,Q6,Q7,B,dB,ddB,dddB,p0,p1,p2,p3,tabl];
p0={0,1}; p1={5,.5}; p2={0,.5}; p3={0,1}; (* Four points *)
dt=.01; q1=3dt; q2=3dt^2; q3=dt^3; q4=2q2; q5=6q3;
Q6=p0-2p1+p2; Q7=3(p1-p2)-p0+p3;
B=p0; dB=(p1-p0) q1+Q6 q2+Q7 q3; (* space indicates *)
ddB=Q6 q4+Q7 q5; dddB=Q7 q5;     (* multiplication *)
tabl={};
Do[{tabl=Append[tabl,B], B=B+dB, dB=dB+ddB, ddB=ddB+dddB},
                                                    {t,0,1,dt}];
ListPlot[tabl];
```

Figure 6.4: A Fast Bézier Curve Algorithm.

Each point of the curve (i.e., each pixel in the loop) is calculated by three additions and three assignments only. There are no multiplications and no table lookups. This is a very fast algorithm indeed!

6.4 Properties of the Curve

The following useful properties are discussed in this section:

1. The weights add up to 1 (they are barycentric). This is easily shown from Newton's binomial theorem $(a+b)^n = \sum_{i=0}^{n} \binom{n}{i} a^i b^{n-i}$:

$$1 = \bigl(t + (1-t)\bigr)^n = \sum_{i=0}^{n} \binom{n}{i} t^i (1-t)^{n-i} = \sum_{i=0}^{n} B_{n,i}(t). \tag{6.12}$$

2. The curve passes through the two endpoints $\mathbf{P}_0$ and $\mathbf{P}_n$. We assume that $0^0 = 1$ and observe that

$$B_{n,0}(0) = \tbinom{n}{0} 0^0 (1-0)^{n-0} = 1 \cdot 1 \cdot 1^n = 1,$$

which implies

$$\mathbf{P}(0) = \sum_{i=0}^{n} \mathbf{P}_i B_{n,i}(0) = \mathbf{P}_0 B_{n,0}(0) = \mathbf{P}_0.$$

Also, the relation

$$B_{n,n}(1) = \tbinom{n}{n} 1^n (1-1)^{(n-n)} = 1 \cdot 1 \cdot 0^0 = 1,$$

implies

$$\mathbf{P}(1) = \sum_{i=0}^{n} \mathbf{P}_i B_{n,i}(1) = \mathbf{P}_n B_{n,n}(1) = \mathbf{P}_n.$$

3. Another interesting property of the Bézier curve is its symmetry with respect to the numbering of the control points. If we number the points $\mathbf{P}_n, \mathbf{P}_{n-1}, \ldots, \mathbf{P}_0$, we end up with the same curve, except that it proceeds from right (point $\mathbf{P}_0$) to left (point $\mathbf{P}_n$). The Bernstein polynomials satisfy the identity $B_{n,j}(t) = B_{n,n-j}(1-t)$, which can be proved directly and which can be used to prove the symmetry

$$\sum_{j=0}^{n} \mathbf{P}_j B_{n,j}(t) = \sum_{j=0}^{n} \mathbf{P}_{n-j} B_{n,j}(1-t).$$

4. The first derivative (the tangent vector) of the curve is straightforward to derive

$$\begin{aligned}
\mathbf{P}^t(t) &= \sum_{i=0}^{n} \mathbf{P}_i B'_{n,i}(t) \\
&= \sum_{0}^{n} \mathbf{P}_i \tbinom{n}{i} \left[i\, t^{i-1} (1-t)^{n-i} + t^i (n-i)(1-t)^{n-i-1}(-1) \right] \\
&= \sum_{0}^{n} \mathbf{P}_i \tbinom{n}{i} i\, t^{i-1} (1-t)^{n-i} - \sum_{0}^{n-1} \mathbf{P}_i \tbinom{n}{i} t^i (n-i)(1-t)^{n-1-i}
\end{aligned}$$

(using the identity $n\binom{n-1}{i-1} = i\binom{n}{i}$, we get)

$$= n\sum_{1}^{n}\mathbf{P}_i\tbinom{n-1}{i-1}t^{i-1}(1-t)^{(n-1)-(i-1)} - n\sum_{0}^{n-1}\mathbf{P}_i\tbinom{n-1}{i}t^{i}(1-t)^{n-1-i}$$

(but $\tbinom{n-1}{i-1}t^{i-1}(1-t)^{(n-1)-(i-1)} = B_{n-1,i-1}(t)$, so)

$$= n\sum_{0}^{n-1}\mathbf{P}_{i+1}B_{n-1,i}(t) - n\sum_{0}^{n-1}\mathbf{P}_iB_{n-1,i}(t)$$

$$= n\sum_{0}^{n-1}[\mathbf{P}_{i+1}-\mathbf{P}_i]B_{n-1,i}(t)$$

$$= n\sum_{0}^{n-1}\Delta\mathbf{P}_iB_{n-1,i}(t), \quad \text{where} \quad \Delta\mathbf{P}_i = \mathbf{P}_{i+1}-\mathbf{P}_i. \tag{6.13}$$

Note that the tangent vector is a Bézier weighted sum (of n terms) where each Bernstein polynomial is the weight of a "control point" $\Delta\mathbf{P}_i$ ($\Delta\mathbf{P}_i$ is the difference of two points, hence it is a vector, but since it is represented by a pair or a triplet, we can conveniently consider it a point). As a result, the second derivative is obviously another Bézier sum based on the $n-1$ "control points" $\Delta^2\mathbf{P}_i = \Delta\mathbf{P}_{i+1} - \Delta\mathbf{P}_i = \mathbf{P}_{i+2} - 2\mathbf{P}_{i+1} + \mathbf{P}_i$.

5. The weight functions $B_{n,i}(t)$ have a maximum at $t = i/n$. To see this, we first differentiate the weights

$$\begin{aligned} B'_{n,i}(t) &= \tbinom{n}{i}\left[i\,t^{i-1}(1-t)^{n-i} + t^i(n-i)(1-t)^{n-i-1}(-1)\right]\\ &= \tbinom{n}{i}i\,t^{i-1}(1-t)^{n-i} - \tbinom{n}{i}t^i(n-i)(1-t)^{n-1-i}, \end{aligned}$$

then equate the derivative to zero $\tbinom{n}{i}i\,t^{i-1}(1-t)^{n-i} - \tbinom{n}{i}t^i(n-i)(1-t)^{n-1-i} = 0$. Dividing by $t^{i-1}(1-t)^{n-i-1}$ yields $i(1-t) - t(n-i) = 0$ or $t = i/n$.

6. The two derivatives $\mathbf{P}^t(0)$ and $\mathbf{P}^t(1)$ are easy to derive from Equation (6.13) and are used to reshape the curve. They are $\mathbf{P}^t(0) = n(\mathbf{P}_1 - \mathbf{P}_0)$ and $\mathbf{P}^t(1) = n(\mathbf{P}_n - \mathbf{P}_{n-1})$. Since n is always positive, we conclude that $\mathbf{P}^t(0)$, the initial tangent of the curve, points in the direction from $\mathbf{P}_0$ to $\mathbf{P}_1$. This initial tangent can easily be controlled by moving point $\mathbf{P}_1$. The situation for the final tangent is similar.

7. The Bézier curve features global control. This means that moving one control point $\mathbf{P}_i$ modifies the entire curve. Most of the change, however, occurs at the vicinity of $\mathbf{P}_i$. This feature stems from the fact that the weight functions $B_{n,i}(t)$ are nonzero for all values of t except $t = 0$ and $t = 1$. Thus, any change in a control point $\mathbf{P}_i$ affects the contribution of the term $\mathbf{P}_iB_{n,i}(t)$ for all values of t. The behavior of the global control of the Bézier curve is easy to analyze. When a control point $\mathbf{P}_k$ is moved by a vector (α, β) to a new location $\mathbf{P}_k + (\alpha, \beta)$, the curve $\mathbf{P}(t)$ is changed from the original sum $\sum B_{ni}(t)\mathbf{P}_i$ to

$$\sum_{i=0}^{n} B_{ni}(t)\mathbf{P}_i + B_{nk}(t)(\alpha,\beta) = \mathbf{P}(t) + B_{nk}(t)(\alpha,\beta).$$

Thus, every point $\mathbf{P}(t_0)$ on the curve is moved by the vector $B_{nk}(t_0)(\alpha, \beta)$. The points are all moved in the same direction, but by different amounts, depending on t_0. This behavior is demonstrated by Figure 6.19b. (In principle, the figure is for a rational curve, but the particular choice of weights in the figure results in a standard curve.)

8. The concept of the *convex hull* of a set of points was introduced in Section 2.2.5. Here, we show a connection between the Bézier curve and the convex hull. Let $\mathbf{P}_1$, $\mathbf{P}_2$,..., $\mathbf{P}_n$ be a given set of points and let a point $\mathbf{P}$ be constructed as a barycentric sum of these points with nonnegative weights, i.e.,

$$\mathbf{P}=\sum_{i=1}^{n} a_i\mathbf{P}_i, \quad \text{where } \sum a_i = 1 \text{ and } a_i \geq 0. \tag{6.14}$$

It can be shown that the set of all points $\mathbf{P}$ satisfying Equation (6.14) lies in the convex hull of $\mathbf{P}_1$, $\mathbf{P}_2$ through $\mathbf{P}_n$. The Bézier curve, Equation (6.5), satisfies Equation (6.14) for all values of t, so all its points lie in the convex hull of the set of control points. Thus, the curve is said to have the *convex hull property.* The significance of this property is that it makes the Bézier curve more predictable. A designer specifying a set of control points needs just a little experience to visualize the shape of the curve, since the convex hull property guarantees that the curve will not "stray" far from the control points.

9. The control polygon of a Bézier curve intersects the curve at the first and the last points and in general may intersect the curve at a certain number m, of points (Figure 6.1, where m is 2, 3, or 4, may help to visualize this). If we take a straight segment and maneuver it to intersect the curve as many times as possible, we find that the number of intersection points is always less than or equal m. This property of the Bézier curve may be termed variation diminution.

10. Imagine that each control point is moved 10 units to the left. Such a transformation will move every point on the curve to the left by the same amount. Similarly, if the control points are rotated, reflected, or are subject to any other *affine* transformation, the entire curve will be transformed in the same way. We say that the Bézier curve is invariant under affine transformations. However, the curve is not invariant under projections. If we compute a three-dimensional Bézier curve and project every point on the curve by a perspective projection, we end up with a two-dimensional curve $\mathbf{P}(t)$. If we then project the three-dimensional control points and compute a two-dimensional Bézier curve $\mathbf{Q}(t)$ from the projected, two-dimensional points, the two curves $\mathbf{P}(t)$ and $\mathbf{Q}(t)$ will be different. Invariance under projections can be achieved by switching from the standard Bézier curve to the rational Bézier curve (Section 6.15).

6.5 Connecting Bézier Curves

The Bézier curve is a polynomial of degree n, which makes it slow to compute for large values of n. It is therefore preferable to connect several Bézier segments, each defined by a few points, typically four to six, into one smooth curve. The condition for smooth connection of two such segments is easy to derive. We assume that the control points are divided into two sets $\mathbf{P}_0, \mathbf{P}_1, \ldots, \mathbf{P}_n$ and $\mathbf{Q}_0, \mathbf{Q}_1, \ldots, \mathbf{Q}_m$. In order for the two segments to connect, $\mathbf{P}_n$ must equal $\mathbf{Q}_0$. We already know that the extreme tangent vectors of the Bézier curve satisfy

$$\mathbf{Q}^t(0) = m(\mathbf{Q}_1 - \mathbf{Q}_0) \quad \text{and} \quad \mathbf{P}^t(1) = n(\mathbf{P}_n - \mathbf{P}_{n-1}).$$

The condition for a smooth connection is $\mathbf{Q}^t(0) = \mathbf{P}^t(1)$ or $m\mathbf{Q}_1 - m\mathbf{Q}_0 = n\mathbf{P}_n - n\mathbf{P}_{n-1}$. Substituting $\mathbf{Q}_0 = \mathbf{P}_n$ yields

$$\mathbf{P}_n = \frac{m}{m+n}\mathbf{Q}_1 + \frac{n}{m+n}\mathbf{P}_{n-1}. \tag{6.15}$$

The three points $\mathbf{P}_{n-1}$, $\mathbf{P}_n$, and $\mathbf{Q}_1$ must therefore be dependent. Hence, the condition for smooth linking is that the three points $\mathbf{P}_{n-1}$, $\mathbf{P}_n$, and $\mathbf{Q}_1$ be *collinear*. In the special case where $n = m$, Equation (6.15) reduces to $\mathbf{P}_n = 0.5\mathbf{Q}_1 + 0.5\mathbf{P}_{n-1}$, implying that $\mathbf{P}_n$ should be the midpoint between $\mathbf{Q}_1$ and $\mathbf{P}_{n-1}$.

Example: Given that $\mathbf{P}_4 = \mathbf{Q}_0 = (6, -1)$, $\mathbf{Q}_1 = (7, 0)$, and $m = 5$, we compute $\mathbf{P}_3$ by

$$(6, -1) = \frac{5}{4+5}(7, 0) + \frac{4}{4+5}\mathbf{P}_3,$$

which yields $\mathbf{P}_3 = (21/4, -9/4)$.

◇ **Exercise 6.8:** A more general condition for a smooth connection of two curve segments is $\alpha\mathbf{Q}^t(0) = \mathbf{P}^t(1)$. The two tangents at the connection point are in the same direction, but have different magnitudes. Discuss this condition and what it means for the three control points $\mathbf{P}_{n-1}$, $\mathbf{P}_n = \mathbf{Q}_0$, and $\mathbf{Q}_1$.

Breaking large curves into short segments has the additional advantage of easy control. The Bézier curve offers only global control, but if it is constructed of separate segments, a change in the control points in one segment will not affect the other segments. Figure 6.5 is an example of two Bézier segments connected smoothly.

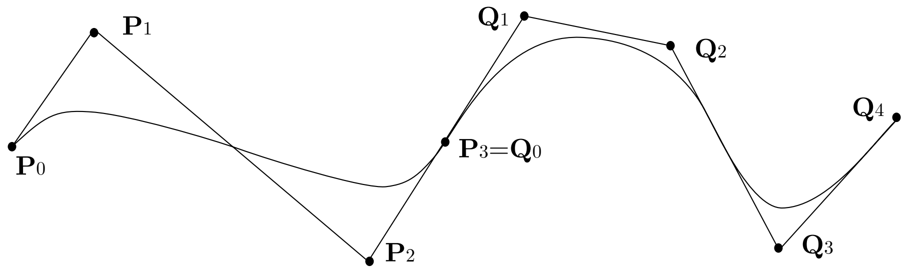


Figure 6.5: Connecting Bézier Segments.

6.6 The Bézier Curve as a Linear Interpolation

The original form of the Bézier curve, as developed by de Casteljau in 1959, is based on an approach entirely different from that of Bézier. Specifically, it employs linear interpolation and the *mediation operator*. Before we start, Figure 6.6 captures the essence of the concepts discussed here. The figure shows how a set of straight segments (or, equivalently, a single segment that slides along the base lines) creates the illusion (some would say, the magic) of a curve. Such a curve is called the envelope of the set, and the linear interpolation method of this section shows how to extend this simple construction to more than three points and two segments.

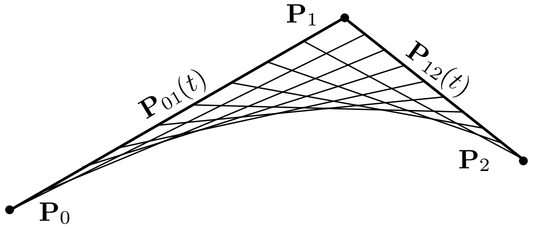


Figure 6.6: A Curve as an Envelope of Straight Segments.

Figure 6.6 involves only three points, which makes it easy to derive the expression of the envelope. The equation of the straight segment from $\mathbf{P}_0$ to $\mathbf{P}_1$ is $\mathbf{P}_{01}(t) = (1-t)\mathbf{P}_0 + t\,\mathbf{P}_1$ and the equation of the segment between $\mathbf{P}_1$ and $\mathbf{P}_2$ is similarly $\mathbf{P}_{12}(t) = (1-t)\mathbf{P}_1 + t\,\mathbf{P}_2$. If we fix t at a certain value, then $\mathbf{P}_{01}(t)$ and $\mathbf{P}_{12}(t)$ become points on the two segments. The straight segment connecting these points has the familiar form

$$\mathbf{P}(t) = (1-t)\mathbf{P}_{01}(t) + t\,\mathbf{P}_{12}(t) = (1-t)^2\mathbf{P}_0 + 2t(1-t)\mathbf{P}_1 + t^2\mathbf{P}_2.$$

For a fixed t, this is a point on the Bézier curve defined by $\mathbf{P}_0$, $\mathbf{P}_1$, and $\mathbf{P}_2$. When t is varied, the entire curve segment is obtained. Thus, the magical envelope has become a familiar curve. We can call this envelope a multilinear curve. *Linear*, because it is constructed from straight segments, and *multi*, because several such segments are required.

In order to extend this method to more than three points, we need appropriate notation. We start with a simple definition. The mediation operator $t[\![\mathbf{P}_0, \mathbf{P}_1]\!]$ between two points $\mathbf{P}_0$ and $\mathbf{P}_1$ is defined as the familiar linear interpolation*

$$t[\![\mathbf{P}_0, \mathbf{P}_1]\!] = (1-t)\mathbf{P}_0 + t\mathbf{P}_1 = t(\mathbf{P}_1 - \mathbf{P}_0) + \mathbf{P}_0, \quad \text{where} \quad 0 \le t \le 1.$$

The general definition, for any number of points, is recursive. The mediation operator

* The term "mediation" seems to have originated in [Knuth 86].

can be applied to any number of points according to

$$\begin{aligned}
t[\![\mathbf{P}_0,\ldots,\mathbf{P}_n]\!] &= t[\![\,t[\![\mathbf{P}_0,\ldots,\mathbf{P}_{n-1}]\!],t[\![\mathbf{P}_1,\ldots,\mathbf{P}_n]\!]\,]\!],\\
&\ \vdots\\
t[\![\mathbf{P}_0,\mathbf{P}_1,\mathbf{P}_2,\mathbf{P}_3]\!] &= t[\![\,t[\![\mathbf{P}_0,\mathbf{P}_1,\mathbf{P}_2]\!],t[\![\mathbf{P}_1,\mathbf{P}_2,\mathbf{P}_3]\!]\,]\!],\\
t[\![\mathbf{P}_0,\mathbf{P}_1,\mathbf{P}_2]\!] &= t[\![\,t[\![\mathbf{P}_0,\mathbf{P}_1]\!],t[\![\mathbf{P}_1,\mathbf{P}_2]\!]\,]\!],\\
t[\![\mathbf{P}_0,\mathbf{P}_1]\!] &= (1-t)\mathbf{P}_0+t\mathbf{P}_1 = t(\mathbf{P}_1-\mathbf{P}_0)+\mathbf{P}_0,\quad \text{where}\quad 0\le t\le 1.
\end{aligned}$$

This operator creates curves that interpolate between the points. It has the advantages of being a simple mathematical function (and therefore fast to calculate) and of producing interpolation curves whose shape can easily be predicted. We examine cases involving more and more points.

Case 1. Two points. Given the two points $\mathbf{P}_0$ and $\mathbf{P}_1$, we denote the straight segment connecting them by $\mathbf{L}_{01}$. It is easy to see that $\mathbf{L}_{01} = t[\![\mathbf{P}_0,\mathbf{P}_1]\!]$, because the mediation operator is a linear function of t and because $0[\![\mathbf{P}_0,\mathbf{P}_1]\!] = \mathbf{P}_0$ and $1[\![\mathbf{P}_0,\mathbf{P}_1]\!] = \mathbf{P}_1$. Notice that values of t below 0 or above 1 correspond to those parts of the line that don't lie between the two points. Such values may be of interest in certain cases but not in the present context. The interpolation curve between the two points is denoted by $\mathbf{P}_1(t)$ and is simply selected as the line $\mathbf{L}_{01}$ connecting the points. Hence, $\mathbf{P}_1(t) = \mathbf{L}_{01} = t[\![\mathbf{P}_0,\mathbf{P}_1]\!]$. Notice that a straight line is also a polynomial of degree 1.

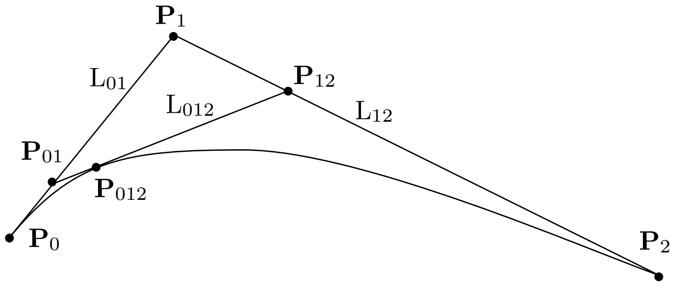


Figure 6.7: Repeated Linear Interpolation.

Case 2. Three points. Given the three points $\mathbf{P}_0$, $\mathbf{P}_1$, and $\mathbf{P}_2$ (Figure 6.7), the mediation operator can be used to construct an interpolation curve between them in the following steps:

1. Construct the two lines $\mathbf{L}_{01} = t[\![\mathbf{P}_0,\mathbf{P}_1]\!]$ and $\mathbf{L}_{12} = t[\![\mathbf{P}_1,\mathbf{P}_2]\!]$.

2. For some $0 \le t_0 \le 1$, consider the two points $\mathbf{P}_{01} = t_0[\![\mathbf{P}_0,\mathbf{P}_1]\!]$ and $\mathbf{P}_{12} = t_0[\![\mathbf{P}_1,\mathbf{P}_2]\!]$. Connect the points with a line $\mathbf{L}_{012}$. The equation of this line is, of course, $t[\![\mathbf{P}_{01},\mathbf{P}_{12}]\!]$ and it equals

$$\mathbf{L}_{012} = t[\![\mathbf{P}_{01},\mathbf{P}_{12}]\!] = t[\![\,t[\![\mathbf{P}_0,\mathbf{P}_1]\!],t[\![\mathbf{P}_1,\mathbf{P}_2]\!]\,]\!] = t[\![\mathbf{P}_0,\mathbf{P}_1,\mathbf{P}_2]\!].$$

3. For the same t_0, select point $\mathbf{P}_{012} = t_0[\![\mathbf{P}_0, \mathbf{P}_1, \mathbf{P}_2]\!]$ on $\mathbf{L_{012}}$. The point can be expressed as

$$\mathbf{P}_{012} = t_0[\![\mathbf{P}_0, \mathbf{P}_1, \mathbf{P}_2]\!] = t_0[\![\mathbf{P}_{01}, \mathbf{P}_{12}]\!] = t_0[\![\, t_0[\![\mathbf{P}_0, \mathbf{P}_1]\!], t_0[\![\mathbf{P}_1, \mathbf{P}_2]\!]\,]\!].$$

Now, release t_0 and let it vary from 0 to 1. Point $\mathbf{P}_{012}$ slides along the line $\mathbf{L}_{012}$, whose endpoints will, in turn, slide along $\mathbf{L}_{01}$ and $\mathbf{L}_{12}$. The curve described by point $\mathbf{P}_{012}$ as it is sliding is the interpolation curve for $\mathbf{P}_0$, $\mathbf{P}_1$, and $\mathbf{P}_2$ that we are seeking. It is the equivalent of the envelope curve of Figure 6.6. We denote it by $\mathbf{P}_2(t)$ and its expression is easy to calculate, using the definition of $t[\![\mathbf{P}_i, \mathbf{P}_j]\!]$:

$$\begin{aligned}
\mathbf{P}_2(t) &= t[\![\mathbf{P}_0, \mathbf{P}_1, \mathbf{P}_2]\!]\\
&= t[\![\, t[\![\mathbf{P}_0, \mathbf{P}_1]\!], t[\![\mathbf{P}_1, \mathbf{P}_2]\!]\,]\!]\\
&= t[\![t\mathbf{P}_1 + (1-t)\mathbf{P}_0, t\mathbf{P}_2 + (1-t)\mathbf{P}_1]\!]\\
&= t[t\mathbf{P}_2 + (1-t)\mathbf{P}_1] + (1-t)[t\mathbf{P}_1 + (1-t)\mathbf{P}_0]\\
&= \mathbf{P}_0(1-t)^2 + 2\mathbf{P}_1 t(1-t) + \mathbf{P}_2 t^2.
\end{aligned}$$

$\mathbf{P}_2(t)$ is therefore the Bézier curve for three points.

Case 3. Four points. Given the four points $\mathbf{P}_0$, $\mathbf{P}_1$, $\mathbf{P}_2$, and $\mathbf{P}_3$, we follow similar steps:

1. Construct the three lines $\mathbf{L}_{01} = t[\![\mathbf{P}_0, \mathbf{P}_1]\!]$, $\mathbf{L}_{12} = t[\![\mathbf{P}_1, \mathbf{P}_2]\!]$, and $\mathbf{L}_{23} = t[\![\mathbf{P}_2, \mathbf{P}_3]\!]$.

2. Select three points, $\mathbf{P}_{01} = t_0[\![\mathbf{P}_0, \mathbf{P}_1]\!]$, $\mathbf{P}_{12} = t_0[\![\mathbf{P}_1, \mathbf{P}_2]\!]$, and $\mathbf{P}_{23} = t_0[\![\mathbf{P}_2, \mathbf{P}_3]\!]$, and construct lines $\mathbf{L}_{012} = t[\![\mathbf{P}_0, \mathbf{P}_1, \mathbf{P}_2]\!] = t[\![\mathbf{P}_{01}, \mathbf{P}_{12}]\!]$ and $\mathbf{L}_{123} = t[\![\mathbf{P}_1, \mathbf{P}_2, \mathbf{P}_3]\!] = t[\![\mathbf{P}_{12}, \mathbf{P}_{23}]\!]$.

3. Select two points, $\mathbf{P}_{012} = t_0[\![\mathbf{P}_{01}, \mathbf{P}_{12}]\!]$ on segment $\mathbf{L}_{012}$ and $\mathbf{P}_{123} = t_0[\![\mathbf{P}_{12}, \mathbf{P}_{23}]\!]$ on segment $\mathbf{L}_{123}$. Construct a new segment $\mathbf{L}_{0123}$ as the mediation $t[\![\mathbf{P}_0, \mathbf{P}_1, \mathbf{P}_2, \mathbf{P}_3]\!] = t[\![\mathbf{P}_{012}, \mathbf{P}_{123}]\!]$.

4. Select point $\mathbf{P}_{0123} = t_0[\![\mathbf{P}_{012}, \mathbf{P}_{123}]\!]$ on $\mathbf{L}_{0123}$.

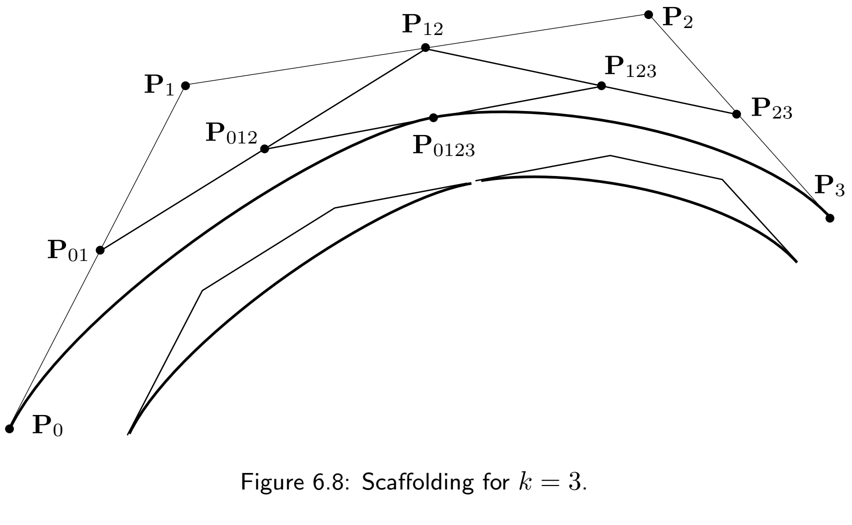


Figure 6.8: Scaffolding for $k = 3$.

When t_0 varies from 0 to 1, point $\mathbf{P}_{0123}$ slides along $\mathbf{L}_{0123}$, whose endpoints, in turn, slide along $\mathbf{L}_{012}$ and $\mathbf{L}_{123}$, which also slide. The entire structure, which resembles a *scaffolding* (Figure 6.8), slides along the original three lines. The interpolation curve for the four original points is denoted by $\mathbf{P}_3(t)$ and its expression is not hard to calculate, using the expression for $\mathbf{P}_2(t) = t[\![\mathbf{P}_0, \mathbf{P}_1, \mathbf{P}_2]\!]$:

$$\begin{aligned}\mathbf{P}_3(t) &= t[\![\mathbf{P}_0, \mathbf{P}_1, \mathbf{P}_2, \mathbf{P}_3]\!] = t[\![\, t[\![\mathbf{P}_0, \mathbf{P}_1, \mathbf{P}_2]\!], t[\![\mathbf{P}_1, \mathbf{P}_2, \mathbf{P}_3]\!]\,]\!]\\ &= t[t^2\mathbf{P}_3 + 2t(1-t)\mathbf{P}_2 + (1-t)^2\mathbf{P}_1]\\ &\quad + (1-t)[t^2\mathbf{P}_2 + 2t(1-t)\mathbf{P}_1 + (1-t)^2\mathbf{P}_0]\\ &= t^3\mathbf{P}_3 + 3t^2(1-t)\mathbf{P}_2 + 3t(1-t)^2\mathbf{P}_1 + (1-t)^3\mathbf{P}_0.\end{aligned}$$

$\mathbf{P}_3(t)$ is therefore the Bézier curve for four points.

Case 4. In the general case, $n+1$ points $\mathbf{P}_0$, $\mathbf{P}_1, \ldots$, $\mathbf{P}_n$ are given. The interpolation curve is, similarly, $t[\![\mathbf{P}_0, \mathbf{P}_1, \ldots, \mathbf{P}_n]\!] = t[\![\mathbf{P}_{01\ldots n-1}, \mathbf{P}_{12\ldots n}]\!]$. It can be proved by induction that its value is the degree-n polynomial

$$\mathbf{P}_n(t) = \sum_{i=0}^{n} \mathbf{P}_i B_{n,i}(t), \quad \text{where} \quad B_{n,i}(t) = \binom{n}{i} t^i (1-t)^{n-i},$$

that is the Bézier curve for $n+1$ points. The two approaches to curve construction, using Bernstein polynomials and using scaffolding, are therefore equivalent.

◇ **Exercise 6.9:** The scaffolding algorithm illustrated in Figure 6.8 is easy to understand because of the special placement of the four control points. The resulting curve is similar to a circular arc and doesn't have an inflection point (Section 1.6.8). Prove your grasp of this algorithm by performing it on the curve of Figure 6.9. Try to select the intermediate points so as to end up with the inflection point.

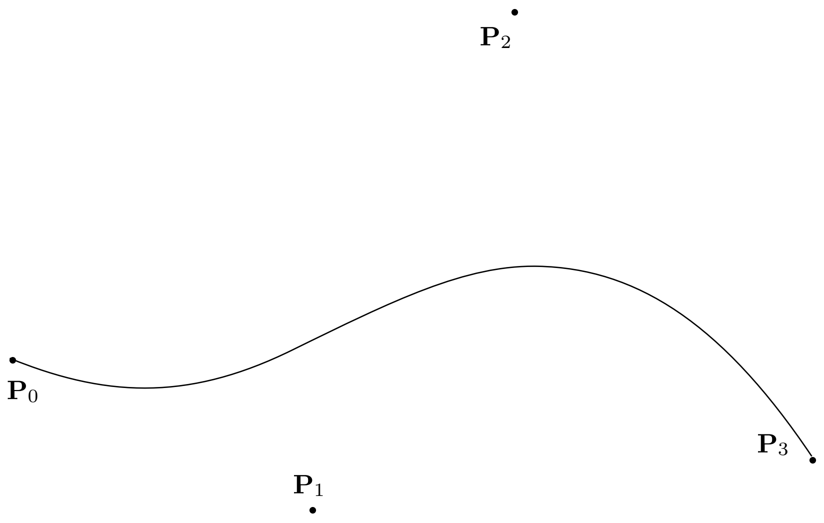


Figure 6.9: Scaffolding With an Inflection Point.

Figure 6.10 summarizes the process of scaffolding in the general case. The process takes n steps. In the first step, n new points are constructed between the original $n+1$ control points. In the second step, $n-1$ new points are constructed, between the n points of step 1 and so on, up to step n, where one point is constructed. The total number of points constructed during the entire process is therefore

$$n + (n-1) + (n-2) + \cdots + 2 + 1 = n(n+1)/2.$$

Step	Points constructed	# of points
1	$\mathbf{P}_{01}\,\mathbf{P}_{12}\,\mathbf{P}_{23}\ldots\mathbf{P}_{n-1,n}$	n
2	$\mathbf{P}_{012}\,\mathbf{P}_{123}\,\mathbf{P}_{234}\ldots\mathbf{P}_{n-2,n-1,n}$	$n-1$
3	$\mathbf{P}_{0123}\,\mathbf{P}_{1234}\,\mathbf{P}_{2345}\ldots\mathbf{P}_{n-3,n-2,n-1,n}$	$n-2$
$\vdots$	$\vdots$	$\vdots$
n	$\mathbf{P}_{0123\ldots n}$	1

$$\begin{tikzcd}
\mathbf{P}_0 \arrow[dr, dash] & & \mathbf{P}_1 \arrow[dl, dash] \arrow[dr, dash] & & \mathbf{P}_2 \arrow[dl, dash] \arrow[dr, dash] & & \mathbf{P}_3 \arrow[dl, dash] \\
& \mathbf{P}_{01} \arrow[dr, dash] & & \mathbf{P}_{12} \arrow[dl, dash] \arrow[dr, dash] & & \mathbf{P}_{23} \arrow[dl, dash] & \\
& & \mathbf{P}_{012} \arrow[dr, dash] & & \mathbf{P}_{123} \arrow[dl, dash] & & \\
& & & \mathbf{P}_{0123} & & &
\end{tikzcd}$$

Figure 6.10: The n Steps of Scaffolding.

6.7 Blossoming

The curves derived and discussed in the preceding chapters are based on polynomials. A typical curve is a pair or a triplet of polynomials of a certain degree n in t. Mathematicians know that a degree-n polynomial $P_n(t)$ of a single variable can be associated with a function $f(u_1, u_2, \ldots, u_n)$ in n variables that's linear (i.e., degree-1) in each variable and is symmetric with respect to the order of its variables. Such functions were named *blossom* by Lyle Ramshaw in [Ramshaw 87] to denote arrival at a promising stage. (The term *pole* was originally used by de Casteljau for those functions.) [Gallier 00] is a general, detailed reference for this topic.

Given a Bézier curve, this section shows how to derive its blossom and how to use the blossom to label the intermediate points obtained in the scaffolding construction. Other sections show how to apply blossoms to curve algorithms, such as curve subdivision (Section 6.8) and degree elevation (Section 6.9).

Dictionary definitions

Blossom:
Noun: The period of greatest prosperity or productivity.
Verb: To develop or come to a promising stage (Youth blossomed into maturity).

Blossoming: The process of budding and unfolding of blossoms.

We start by developing a special notation for use with blossoms. The equation of the straight segment from point $\mathbf{P}_0$ to point $\mathbf{P}_1$ is the familiar linear interpolation $\mathbf{P}(u) = (1-u)\mathbf{P}_0 + u\mathbf{P}_1$. Its start point is $\mathbf{P}(0)$, its end point is $\mathbf{P}(1)$, and a general point on this segment is $\mathbf{P}(u)$ for $0 \le u \le 1$. Because a straight segment has zero curvature, parameter values indicate arc lengths. Thus, the distance between $\mathbf{P}(0)$ and $\mathbf{P}(u)$ is proportional to u and the distance between $\mathbf{P}(u)$ and $\mathbf{P}(1)$ is proportional to $1-u$. We can therefore consider parameter values u in the interval $[0,1]$ a measure of distance (called affine distance) from the start of the segment. We introduce the symbol $\langle\mathbf{u}\rangle$ to denote point $\mathbf{P}(u)$. Similarly, points $\mathbf{P}(0)$ and $\mathbf{P}(1)$ are denoted by $\langle\mathbf{0}\rangle$ and $\langle\mathbf{1}\rangle$, respectively (Figure 6.11a).

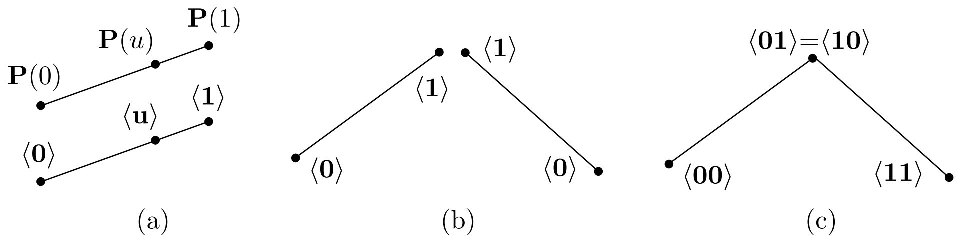

$\mathbf{P}(1)$
$\mathbf{P}(u)$
$\mathbf{P}(0)$
$\langle\mathbf{u}\rangle$
$\langle\mathbf{1}\rangle$
$\langle\mathbf{0}\rangle$
(a)
$\langle\mathbf{1}\rangle$
$\langle\mathbf{1}\rangle$
$\langle\mathbf{0}\rangle$
$\langle\mathbf{0}\rangle$
(b)
$\langle\mathbf{01}\rangle=\langle\mathbf{10}\rangle$
$\langle\mathbf{00}\rangle$
$\langle\mathbf{11}\rangle$
(c)

Figure 6.11: Blossom Notation For Points (Two Segments).

A spline consists of segments connected at the interior points, so we consider two straight segments connected at a common point. The endpoints of each segment are denoted by $\langle\mathbf{0}\rangle$ and $\langle\mathbf{1}\rangle$, but this creates an ambiguity. There are now two points labeled $\langle\mathbf{0}\rangle$ (Figure 6.11b). We distinguish between them by appending a bit to the symbol of each point. The two endpoints of one segment are now denoted by $\langle\mathbf{00}\rangle$ and $\langle\mathbf{01}\rangle$, while the two endpoints of the other segment are denoted by $\langle\mathbf{10}\rangle$ and $\langle\mathbf{11}\rangle$ (Figure 6.11c). The common point can be denoted by either $\langle\mathbf{01}\rangle$ or $\langle\mathbf{10}\rangle$. So far, it seems that the order of the individual indexes, 01 or 10, is immaterial. The new notation is symmetric with respect to the order of point indexes.

We now select a point with a parameter value u on each segment. The two new points are denoted by $\langle\mathbf{0u}\rangle$ and $\langle\mathbf{1u}\rangle$ (Figure 6.12a), but they can also be denoted by $\langle\mathbf{u0}\rangle$ and $\langle\mathbf{u1}\rangle$, respectively. The two points are now connected by a segment and a new point selected at affine distance u on that segment (Figure 6.12b). The new point deserves the label $\langle\mathbf{uu}\rangle$ because the endpoints of its segment have the common index $\mathbf{u}$.

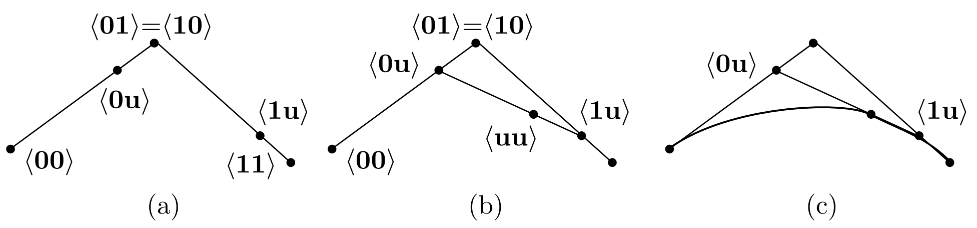

$\langle\mathbf{01}\rangle=\langle\mathbf{10}\rangle$
$\langle\mathbf{0u}\rangle$
$\langle\mathbf{1u}\rangle$
$\langle\mathbf{00}\rangle$
$\langle\mathbf{11}\rangle$
(a)
$\langle\mathbf{01}\rangle=\langle\mathbf{10}\rangle$
$\langle\mathbf{0u}\rangle$
$\langle\mathbf{1u}\rangle$
$\langle\mathbf{uu}\rangle$
$\langle\mathbf{00}\rangle$
(b)
$\langle\mathbf{0u}\rangle$
$\langle\mathbf{1u}\rangle$
(c)

Figure 6.12: Blossom Notation For Points (Two Segments).

At this point it is clear that the simple scaffolding construction of Figure 6.12b is identical to the de Casteljau algorithm of Section 6.6, which implies that point $\langle\mathbf{uu}\rangle$ is located on the Bézier curve defined by the three points $\langle\mathbf{00}\rangle$, $\langle\mathbf{01}\rangle$, and $\langle\mathbf{11}\rangle$ (Figure 6.12c).

To illustrate this process for more points, it is applied to three line segments in Figure 6.13. Two bits are appended to each point in order to distinguish between the segments. Thus, a point is denoted by a triplet of the form $\langle\mathbf{00x}\rangle$, $\langle\mathbf{01x}\rangle$, or $\langle\mathbf{11x}\rangle$. Notice that our indexes are symmetric, so $\langle\mathbf{01x}\rangle = \langle\mathbf{10x}\rangle$, which is why we use $\langle\mathbf{11x}\rangle$ instead of $\langle\mathbf{10x}\rangle$ to identify the third segment.

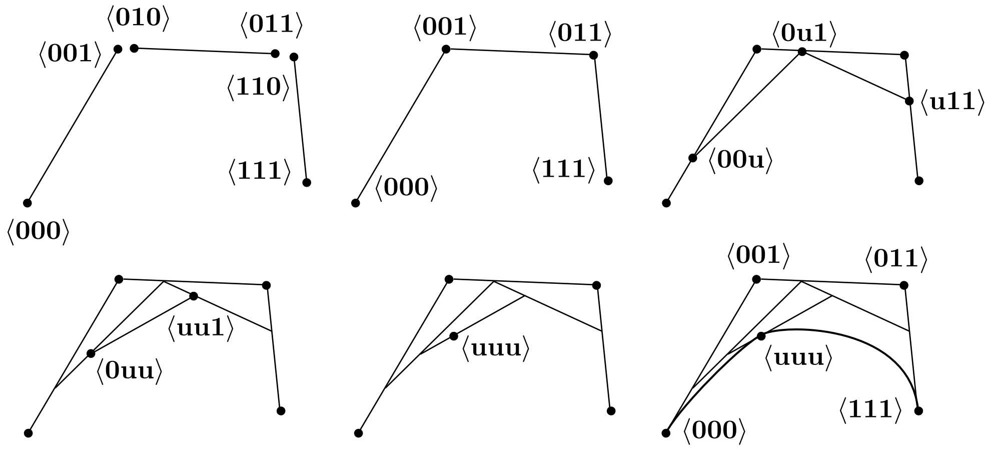


Figure 6.13: Blossom Notation For Points (Three Segments).

Again, our familiarity with the Bézier curve and the de Casteljau algorithm indicates intuitively that point $\langle\mathbf{uuu}\rangle$ is located on the Bézier curve defined by the four control points $\langle\mathbf{000}\rangle$, $\langle\mathbf{001}\rangle$, $\langle\mathbf{011}\rangle$, and $\langle\mathbf{111}\rangle$.

Let us be grateful to people who make us happy, they are the charming gardeners who make our souls blossom.

—Marcel Proust.

An actual construction of the scaffolding for this case verifies our intuitive feeling. Given points $\langle\mathbf{0uu}\rangle$ and $\langle\mathbf{uu1}\rangle$, we can write them as $\langle\mathbf{0uu}\rangle$ and $\langle\mathbf{1uu}\rangle$, which immediately produces point $\langle\mathbf{uuu}\rangle$ (it's located an affine distance u from $\langle\mathbf{0uu}\rangle$). Similarly, given points $\langle\mathbf{00u}\rangle$ and $\langle\mathbf{0u1}\rangle$, we can write them as $\langle\mathbf{00u}\rangle$ and $\langle\mathbf{01u}\rangle$, which immediately produces point $\langle\mathbf{0uu}\rangle$. A similar step produces $\langle\mathbf{00u}\rangle$ if points $\langle\mathbf{000}\rangle$ and $\langle\mathbf{001}\rangle$ are given. Thus, we conclude that knowledge of the four control points can produce all the intermediate points in the scaffolding construction and lead to one point $\langle\mathbf{uuu}\rangle$ that's located on the Bézier curve defined by the control points. This is an informal statement of the *blossoming principle*.

This principle can be illustrated in a different way. We know that point $\langle\mathbf{0u1}\rangle$ is obtained from points $\langle\mathbf{001}\rangle$ and $\langle\mathbf{011}\rangle$ as the linear interpolation $\langle\mathbf{0u1}\rangle = (1-u)\langle\mathbf{001}\rangle + u\langle\mathbf{011}\rangle$. We can therefore start from point $\langle\mathbf{uuu}\rangle$ and figure out its dependence on the four original points $\langle\mathbf{000}\rangle$, $\langle\mathbf{001}\rangle$, $\langle\mathbf{011}\rangle$, and $\langle\mathbf{111}\rangle$ as follows:

$$\begin{aligned}\langle\mathbf{uuu}\rangle &= (1-u)\langle\mathbf{0uu}\rangle + u\langle\mathbf{1uu}\rangle\\ &= (1-u)\big[(1-u)\langle\mathbf{00u}\rangle + u\langle\mathbf{01u}\rangle\big] + u\big[(1-u)\langle\mathbf{10u}\rangle + u\langle\mathbf{11u}\rangle\big]\\ &= (1-u)^2\langle\mathbf{00u}\rangle + 2u(1-u)\langle\mathbf{01u}\rangle + u^2\langle\mathbf{11u}\rangle\\ &= (1-u)^2\big[(1-u)\langle\mathbf{000}\rangle + u\langle\mathbf{001}\rangle\big] + 2u(1-u)\big[(1-u)\langle\mathbf{010}\rangle + u\langle\mathbf{011}\rangle\big]\\ &\quad + u^2\big[(1-u)\langle\mathbf{110}\rangle + u\langle\mathbf{111}\rangle\big]\\ &= (1-u)^3\langle\mathbf{000}\rangle + 3u(1-u)^2\langle\mathbf{001}\rangle + 3u^2(1-u)\langle\mathbf{011}\rangle + u^3\langle\mathbf{111}\rangle\\ &= B_{3,0}(u)\langle\mathbf{000}\rangle + B_{3,1}(u)\langle\mathbf{001}\rangle + B_{3,2}(u)\langle\mathbf{011}\rangle + B_{3,3}(u)\langle\mathbf{111}\rangle,\end{aligned}$$

where $B_{3,i}$ are the Bernstein polynomials for $n = 3$. This again shows that point $\langle\mathbf{uuu}\rangle$ lies on the Bézier curve whose control points are $\langle\mathbf{000}\rangle$, $\langle\mathbf{001}\rangle$, $\langle\mathbf{011}\rangle$, and $\langle\mathbf{111}\rangle$.

So far, blossoming has been used to assign labels to the control points and to the intermediate points. Even this simple application illustrates some of the power and elegance of the blossoming approach. Section 6.6 employs the notation $\mathbf{P}_{234}$, while various authors denote intermediate point i of scaffolding step j by d_i^j. The blossom labels $\langle\mathbf{u_1 u_2 \ldots u_n}\rangle$ are much more natural and useful.

We are now ready to see the actual blossom associated with the degree-n polynomial $P_n(t)$ as given by [Ramshaw 87]. The blossom of $P_n(t)$ is a function $f(u_1, u_2, \ldots, u_n)$ that satisfies the following:

1. f is linear in each variable u_i.
2. f is symmetric; the order of variables is irrelevant. Thus, $f(u_1, u_2, \ldots, u_n) = f(u_2, u_1, \ldots, u_n)$ or any other permutation of the n variables.
3. The diagonal $f(u, u, \ldots, u)$ of f equals $P_n(u)$.

Requirement 1 suggests the name "multilinear function" but [Ramshaw 87] explains why the term "multiaffine" is more appropriate.

Given $P_n(t)$, such a multiaffine function is easy to derive and is also unique. Here is an example for $n = 3$. Given the cubic polynomial $P(t) = -3t^3 + 6t^2 + 3t$, we are looking for a function $f(u, v, w)$ that's linear in each of its three parameters and is symmetric with respect to their order. The general form of such a function is

$$f(u, v, w) = a_1 uvw + a_2 uv + a_3 uw + a_4 vw + a_5 u + a_6 v + a_7 w + a_8.$$

If we also require that $f(u, v, w)$ satisfies $f(t, t, t) = P(t)$ for any t, it becomes obvious that a_1 must equal the coefficient of t^3. Because of the required symmetry, the sum $a_2 + a_3 + a_4$ must equal the coefficient of t^2 and the sum $a_5 + a_6 + a_7$ must equal the coefficient of t. Finally, a_8 must equal the free term of $P(t)$. Thus, we end up with the blossom $f(u, v, w) = -3uvw + 2(uv + uw + vw) + (u + v + w) + 0$. This blossom is unique.

In general, given an n-degree polynomial, the corresponding multiaffine blossom function is easy to construct in this way. Here are some examples.

Degree-0. $P(t) = a \to f(u, v, w) = a,$

Degree-1. $P(t) = at \to f(u, v, w) = \dfrac{a}{3}(u + v + w),$

$$\text{Degree-2. } P(t) = at^2 \to f(u,v,w) = \frac{a}{3}(uv + uw + vw), \tag{6.16}$$

$$\text{Degree-3. } P(t) = a_3t^3 + a_2t^2 + a_1 + a_0$$
$$\to f(u,v,w) = a_3uvw + \frac{a_2}{3}(uv + uw + vw) + \frac{a_1}{3}(u + v + w) + a_0.$$

The discussion above shows that the kth control point of the degree-n polynomial is associated with blossom value $f(\underbrace{00\ldots0}_{n-k}\underbrace{11\ldots1}_{k})$. Notice that there are $n+1$ such values, corresponding to the $n+1$ control points, and that blossom symmetry implies $f(011) = f(101) = f(110)$. If t varies in the general interval $[a,b]$ instead of in $[0,1]$, then the kth control point is associated with the blossom value $f(\underbrace{aa\ldots a}_{n-k}\underbrace{bb\ldots b}_{k})$.

◇ **Exercise 6.10:** Given the four points $\mathbf{P}_0 = (0,1,1)$, $\mathbf{P}_1 = (1,1,0)$, $\mathbf{P}_2 = (4,2,0)$, and $\mathbf{P}_3 = (6,1,1)$, compute the Bézier curve defined by them, construct the three blossoms associated with this curve, and show that the four blossom values $f(0,0,0)$, $f(0,0,1)$, $f(0,1,1)$, and $f(1,1,1)$ yield the control points.

6.8 Subdividing the Bézier Curve

Bézier methods are interactive. It is possible to control the shape of the curve by moving the control points and by smoothly connecting individual segments. Imagine a situation where the points are moved and maneuvered for a while, but the curve "refuses" to get the right shape. This indicates that there are not enough points. There are two ways to increase the number of points. One is to add a point to a segment while increasing its degree. This is called *degree elevation* and is discussed in Section 6.9.

An alternative is to subdivide a Bézier curve segment into two segments such that there is no change in the shape of the curve. If the original segment is of degree n (i.e., based on $n+1$ control points), this is done by adding $2n-1$ new control points and deleting $n-1$ of the original points, bringing the number of points to $(n+1)+(2n-1)-(n-1) = 2n+1$. Each new segment is based on $n+1$ points and they share one of the new points. With more points, it is now possible to manipulate the control points of the two segments in order to fine-tune the shape of the segments. The advantage of this approach is that both the original and the new curves are based on $n+1$ points, so only one set of Bernstein polynomials is needed.

The new points being added consist of some of the ones constructed in the last k steps of the scaffolding process. For the case $k=2$ (quadratic curve segments), the three points $\mathbf{P}_{01}$, $\mathbf{P}_{12}$, and $\mathbf{P}_{012}$ are added and the single point $\mathbf{P}_1$ is deleted (Figure 6.7). The two new segments consist of points $\mathbf{P}_0$, $\mathbf{P}_{01}$, and $\mathbf{P}_{012}$, and $\mathbf{P}_{012}$, $\mathbf{P}_{12}$, and $\mathbf{P}_2$. For the case $k=3$ (cubic segments), the five points $\mathbf{P}_{01}$, $\mathbf{P}_{23}$, $\mathbf{P}_{012}$, $\mathbf{P}_{123}$, and $\mathbf{P}_{0123}$ are added and the two points $\mathbf{P}_1$ and $\mathbf{P}_2$ are deleted (Figure 6.8, duplicated here, where the inset shows the two segments with their control polygons). The two new segments consist of points $\mathbf{P}_0$, $\mathbf{P}_{01}$, $\mathbf{P}_{012}$, and $\mathbf{P}_{0123}$ and $\mathbf{P}_{0123}$, $\mathbf{P}_{123}$, $\mathbf{P}_{23}$, and $\mathbf{P}_3$.

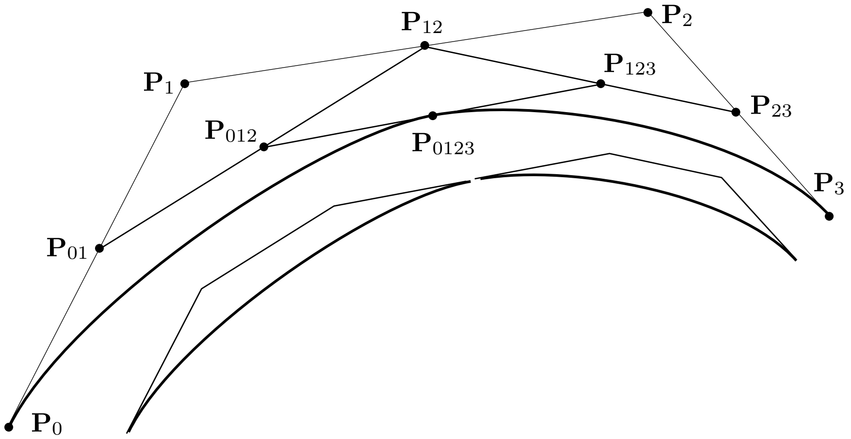


Figure 6.8: Scaffolding and Subdivision for $k = 3$ (Duplicate).

Using the mediation operator to express the new points in the scaffolding in terms of the original control points produces, for the quadratic case

$$\mathbf{P}_{01} = \alpha\mathbf{P}_0+(1-\alpha)\mathbf{P}_1, \mathbf{P}_{12} = \alpha\mathbf{P}_1+(1-\alpha)\mathbf{P}_2, \mathbf{P}_{012} = \alpha^2\mathbf{P}_0+2\alpha(1-\alpha)\mathbf{P}_1+(1-\alpha)^2\mathbf{P}_2,$$

where α is any value in the range $[0,1]$. We can therefore write

$$\begin{pmatrix}\mathbf{P}_0\\ \mathbf{P}_{01}\\ \mathbf{P}_{012}\end{pmatrix} = \begin{pmatrix}1 & 0 & 0\\ \alpha & 1-\alpha & 0\\ \alpha^2 & 2\alpha(1-\alpha) & (1-\alpha)^2\end{pmatrix}\begin{pmatrix}\mathbf{P}_0\\ \mathbf{P}_1\\ \mathbf{P}_2\end{pmatrix},$$
$$\begin{pmatrix}\mathbf{P}_{012}\\ \mathbf{P}_{12}\\ \mathbf{P}_2\end{pmatrix} = \begin{pmatrix}\alpha^2 & 2\alpha(1-\alpha) & (1-\alpha)^2\\ 0 & \alpha & 1-\alpha\\ 0 & 0 & 1\end{pmatrix}\begin{pmatrix}\mathbf{P}_0\\ \mathbf{P}_1\\ \mathbf{P}_2\end{pmatrix},$$

for the left and right segments, respectively.

◇ **Exercise 6.11:** Use the mediation operator to calculate the scaffolding for the cubic case (four control points). Use $\alpha = 1/2$ and write the results in terms of matrices, as above.

In the general case where an $(n + 1)$-point Bézier curve is subdivided, the $n - 1$ points being deleted are $\mathbf{P}_1$, $\mathbf{P}_2$,..., $\mathbf{P}_{n-1}$ (the original $n - 1$ interior control points). The $2n - 1$ points added are the first and last points constructed in each scaffolding step (except the last step, where only one point is constructed). Figure 6.10 shows that these are points $\mathbf{P}_{01}$, $\mathbf{P}_{n-1,n}$ (from step 1), $\mathbf{P}_{012}$, $\mathbf{P}_{n-2,n-1,n}$ (from step 2), $\mathbf{P}_{0123}$, $\mathbf{P}_{n-3,n-2,n-1,n}$ (from step 3), up to $\mathbf{P}_{0123\ldots n}$ from step n.

The $2n - 1$ points being added are therefore

$$\mathbf{P}_{01}, \mathbf{P}_{012}, \mathbf{P}_{0123}, \ldots, \mathbf{P}_{0123\ldots n}, \mathbf{P}_{123\ldots n}, \mathbf{P}_{23\ldots n}, \ldots, \mathbf{P}_{n-1,n}.$$

These points can be computed in two ways as follows:

1. Perform the entire scaffolding procedure and save all the points, then use only the appropriate $2n-1$ points.

2. Compute just the required points. This is done by means of the two relations

$$\text{(a) } \mathbf{P}_{0123\ldots k}=\sum_{j=0}^{k}B_{k,j}(t)\mathbf{P}_j,\quad \text{and}\quad \text{(b) } \mathbf{P}_{n-k,n-k+1,\ldots,n}=\sum_{j=0}^{k}B_{k,j}(t)\mathbf{P}_{n-k+j}. \qquad (6.17)$$

(These expressions can be proved by induction.)

The first decision that has to be made when subdividing a curve, is at what point (what value of t) to break the original curve into two segments. Breaking a curve $\mathbf{P}(t)$ into two segments at $t=0.1$ will result in a short segment followed by a long segment, each defined by $n+1$ control points. Obviously, the first segment will be easier to edit. Once the value of t has been determined, the software computes the $2n-1$ new points. The original $n-1$ interior control points are easy to delete, and the set of $2n+1$ points is partitioned into two sets. The procedure that computed the original curve is now invoked twice, to compute and display the two segments.

◇ **Exercise 6.12:** Given the four points $\mathbf{P}_0=(0,1,1)$, $\mathbf{P}_1=(1,1,0)$, $\mathbf{P}_2=(4,2,0)$, and $\mathbf{P}_3=(6,1,1)$, apply Equation (6.17)a,b to subdivide the Bézier curve $\sum B_{3,i}(t)\mathbf{P}_i$ at $t=1/3$.

Figure 6.14 illustrates how blossoms are applied to the problem of curve subdivision. The points on the left edge of the triangle become the control points of the first segment. In blossom notation these are points $\langle\underbrace{\mathbf{00\ldots0}}_{n-k}\underbrace{\mathbf{tt\ldots t}}_{k}\rangle$. Similarly, the points on the right edge of the triangle become the control points of the second segment. In blossom notation these are points $\langle\underbrace{\mathbf{11\ldots1}}_{n-k}\underbrace{\mathbf{tt\ldots t}}_{k}\rangle$. There are $n+1$ points on each edge, but the total is $2n-1$ because the top of the triangle has just one point, namely $\langle\mathbf{ttt}\rangle$.

$$\begin{tikzcd}
 & & & \langle\mathbf{ttt}\rangle \arrow[dl, dash] \arrow[dr, dash] & & & \\
 & & \langle\mathbf{0tt}\rangle \arrow[dl, dash] \arrow[dr, dash] & & \langle\mathbf{1tt}\rangle \arrow[dl, dash] \arrow[dr, dash] & & \\
 & \langle\mathbf{00t}\rangle \arrow[dl, dash] \arrow[dr, dash] & & \langle\mathbf{01t}\rangle \arrow[dl, dash] \arrow[dr, dash] & & \langle\mathbf{11t}\rangle \arrow[dl, dash] \arrow[dr, dash] & \\
\langle\mathbf{000}\rangle & & \langle\mathbf{001}\rangle & & \langle\mathbf{011}\rangle & & \langle\mathbf{111}\rangle
\end{tikzcd}$$

Figure 6.14: Blossoming for Subdivision.

6.9 Degree Elevation

Degree elevation of the Bézier curve is a process that starts with a Bézier curve $\mathbf{P}_n(t)$ of degree n (i.e., defined by $n+1$ control points) and adds a control point, thereby ending up with a curve $\mathbf{P}_{n+1}(t)$.

The advantage of degree elevation is that the new curve is based on more control points and is therefore easier to edit by maneuvering the points. Its shape can be better fine-tuned than that of the original curve.

Just adding a control point is not very useful because the new point will change the shape of the curve globally. Degree elevation is useful only if it is done without modifying the shape of the curve. The principle of degree elevation is therefore to compute a new set of $n+2$ control points $\mathbf{Q}_i$ from the original set of $n+1$ points $\mathbf{P}_i$, such that the Bézier curve $\mathbf{P}_{n+1}(t)$ defined by the new points will have the same shape as the original curve $\mathbf{P}_n(t)$.

We start with the innocuous identity that's true for any Bézier curve $\mathbf{P}(t)$

$$\mathbf{P}(t) = \big(t + (1-t)\big)\mathbf{P}(t) = t\mathbf{P}(t) + (1-t)\mathbf{P}(t).$$

The two Bézier curves on the right-hand side are polynomials of degree n, but because each is multiplied by t, the polynomial on the left-hand side is of degree $n+1$. Thus, we can represent a degree-$(n+1)$ curve as the weighted sum of two degree-n curves and write the identity in the form $\mathbf{P}_{n+1}(t) = (1-t)\mathbf{P}_n(t) + t\mathbf{P}_n(t)$. We use the notation

$$\mathbf{P}_n(t) = \sum_{i=0}^{n} \tbinom{n}{i} t^i (1-t)^{n-i}\mathbf{P}_i \overset{\text{def}}{=} \langle\langle \mathbf{P}_0, \mathbf{P}_1, \ldots, \mathbf{P}_n \rangle\rangle.$$

(Recall that the angle bracket notation indicates blossoms. The double-angle bracket notation used here implies that each point should be multiplied by the corresponding Bernstein polynomial and the products summed.)

The first step is to express $t\mathbf{P}_n(t)$ in the new notation

$$\begin{aligned} t\mathbf{P}_n(t) &= \sum_{i=0}^{n} \tbinom{n}{i} t^{i+1}(1-t)^{n-i}\mathbf{P}_i = \sum_{k=1}^{m} \binom{m-1}{k-1} t^k (1-t)^{m-k}\mathbf{P}_{k-1} \\ &= \sum_{k=0}^{m} \binom{m}{k} t^k (1-t)^{m-k} \frac{k}{m}\mathbf{P}_{k-1} = \left\langle\!\!\left\langle 0, \frac{\mathbf{P}_0}{n+1}, \frac{2\mathbf{P}_1}{n+1}, \ldots, \frac{n\mathbf{P}_{n-1}}{n+1}, \mathbf{P}_n \right\rangle\!\!\right\rangle. \end{aligned}$$

Here, we first use the substitutions $k = i+1$ and $m = n+1$, and then the identity

$$\binom{m-1}{k-1} = \frac{k}{m}\binom{m}{k}.$$

The next step is to similarly express $(1-t)\mathbf{P}_n(t)$ in the new notation:

$$(1-t)\mathbf{P}_n(t) = \left\langle\!\!\left\langle \mathbf{P}_0, \frac{n\mathbf{P}_1}{n+1}, \frac{(n-1)\mathbf{P}_2}{n+1}, \ldots, \frac{\mathbf{P}_n}{n+1}, 0 \right\rangle\!\!\right\rangle.$$

Adding the two expressions produces

$$\begin{aligned}
\mathbf{P}_{n+1}(t) &= (1-t)\mathbf{P}_n(t) + t\mathbf{P}_n(t) \\
&= \left\langle\!\!\left\langle 0, \frac{\mathbf{P}_0}{n+1}, \frac{2\mathbf{P}_1}{n+1}, \ldots, \frac{n\mathbf{P}_{n-1}}{n+1}, \mathbf{P}_n \right\rangle\!\!\right\rangle \\
&+ \left\langle\!\!\left\langle \mathbf{P}_0, \frac{n\mathbf{P}_1}{n+1}, \frac{(n-1)\mathbf{P}_2}{n+1}, \ldots, \frac{\mathbf{P}_n}{n+1}, 0 \right\rangle\!\!\right\rangle \\
&= \left\langle\!\!\left\langle \mathbf{P}_0, \frac{\mathbf{P}_0+n\mathbf{P}_1}{n+1}, \frac{2\mathbf{P}_1+(n-1)\mathbf{P}_2}{n+1}, \ldots, \frac{n\mathbf{P}_{n-1}+\mathbf{P}_n}{n+1}, \mathbf{P}_n \right\rangle\!\!\right\rangle, \qquad (6.18)
\end{aligned}$$

which shows the $n+2$ control points that define the new, degree-elevated Bézier curve.

If the new control points are denoted by $\mathbf{Q}_i$, then the expression above can be summarized by the following notation:

$$\begin{aligned}
&\mathbf{Q}_0 = \mathbf{P}_0, \\
&\mathbf{Q}_i = a_i\mathbf{P}_{i-1} + (1-a_i)\mathbf{P}_i, \quad \text{where} \quad a_i = \frac{i}{n+1}, \quad i = 1, 2, \ldots, n, \qquad (6.19) \\
&\mathbf{Q}_{n+1} = \mathbf{P}_n.
\end{aligned}$$

◇ **Exercise 6.13:** Given the quadratic Bézier curve defined by the three control points $\mathbf{P}_0$, $\mathbf{P}_1$, and $\mathbf{P}_2$, elevate its degree twice and list the five new control points.

It is possible to elevate the degree of a curve many times. Each time the degree is elevated, the new set of control points grows by one point and also approaches the curve. At the limit, the set consists of infinitely many points that are located on the curve.

◇ **Exercise 6.14:** Given the four control points $\mathbf{P}_0 = (0,0)$, $\mathbf{P}_1 = (1,2)$, $\mathbf{P}_2 = (3,2)$, and $\mathbf{P}_3 = (2,0)$, elevate the degree of the Bézier curve defined by them.

The degree elevation algorithm summarized by Equation (6.19) can also be derived as an application of blossoms. We define a three-parameter function $f_?(u_1, u_2, u_3)$ as a sum of blossoms of two parameters

$$\begin{aligned}
f_?(u_1,u_2,u_3) &= \frac{1}{3}\left[f_2(u_1,u_2) + f_2(u_1,u_3) + f_2(u_2,u_3)\right] \\
&= \frac{1}{3}\Big[\big[a_2u_1u_2 + \frac{a_1}{2}(u_1+u_2) + a_0\big] + \big[a_2u_1u_3 + \frac{a_1}{2}(u_1+u_3) + a_0\big] \\
&\quad + \big[a_2u_2u_3 + \frac{a_1}{2}(u_2+u_3) + a_0\big]\Big] \\
&= \frac{a_2}{3}(u_1u_2 + u_1u_3 + u_2u_3) + a_1(u_1+u_2+u_3) + a_0. \qquad (6.20)
\end{aligned}$$

We notice that $f_?(u_1, u_2, u_3)$ satisfies the following three conditions

1. It is linear in each of its three parameters.
2. It is symmetric with respect to the order of the parameters.
3. Its diagonal, $f_?(u, u, u)$, yields the polynomial $P_2(t) = a_2t^2 + a_1t + a_0$.

We therefore conclude that $f_?(u_1, u_2, u_3)$ is the $(n+1)$-blossom of $P_2(t)$. It should be denoted by $f_3(u_1, u_2, u_3)$. It can be shown that the extension of Equation (6.20) to any $f_{n+1}(u_1, u_2, \ldots, u_{n+1})$ is

$$f_{n+1}(u_1, \ldots, u_{n+1}) = \frac{1}{n+1} \sum_{i=1}^{n+1} f_n(u_1, \ldots, \underline{u}_i, \ldots, u_{n+1}). \tag{6.21}$$

(where the underline indicates a missing parameter).

Section 6.7 shows that control point $\mathbf{P}_k$ of a Bézier curve $\mathbf{P}_n(t)$ is given by the blossom $f(\underbrace{0\ldots0}_{n-k}\underbrace{1\ldots1}_{k})$. Equation (6.21) implies that the same control point $\mathbf{Q}_k$ of a Bézier curve $\mathbf{P}_{n+1}(t)$ is given as the sum

$$\mathbf{Q}_k = \frac{n+1-k}{n+1}\mathbf{P}_k + \frac{k}{n+1}\mathbf{P}_{k-1},$$

which is identical to Equation (6.19).

6.10 Reparametrizing the Curve

The parameter t varies normally in the range $[0, 1]$. It is, however, easy to reparametrize the Bézier curve such that its parameter varies in an arbitrary range $[a, b]$, where a and b are real and $a \le b$. The new curve is denoted by $\mathbf{P}_{ab}(t)$ and is simply the original curve with a different parameter:

$$\mathbf{P}_{ab}(t) = \mathbf{P}\left(\frac{t-a}{b-a}\right).$$

The two functions $\mathbf{P}_{ab}(t)$ and $\mathbf{P}(t)$ produce the same curve when t varies from a to b in the former and from 0 to 1 in the latter. Notice that the new curve has tangent vector

$$\mathbf{P}^t_{ab}(t) = \frac{1}{b-a}\mathbf{P}^t\left(\frac{t-a}{b-a}\right).$$

Reparametrization can also be used to answer the question: Given a Bézier curve $\mathbf{P}(t)$ where $0 \le t \le 1$, how can we calculate a curve $\mathbf{Q}(t)$ that's defined on an arbitrary part of $\mathbf{P}(t)$? More specifically, if $\mathbf{P}(t)$ is defined by control points $\mathbf{P}_i$ and if we select an interval $[a, b]$, how can we calculate control points $\mathbf{Q}_i$ such that the curve $\mathbf{Q}(t)$ based on them will go from $\mathbf{P}(a)$ to $\mathbf{P}(b)$ [i.e., $\mathbf{Q}(0) = \mathbf{P}(a)$ and $\mathbf{Q}(1) = \mathbf{P}(b)$] and will be identical in shape to $\mathbf{P}(t)$ in that interval? As an example, if $[a, b] = [0, 0.5]$, then $\mathbf{Q}(t)$ will be identical to the first half of $\mathbf{P}(t)$. The point is that the interval $[a, b]$ does not have to be inside $[0, 1]$. We may select, for example, $[a, b] = [0.9, 1.5]$ and end up with a curve $\mathbf{Q}(t)$ that will go from $\mathbf{P}(0.9)$ to $\mathbf{P}(1.5)$ as t varies from 0 to 1. Even though the Bézier curve was originally designed with $0 \le t \le 1$ in mind, it can still be calculated for t values outside this range. If we like its shape in the range $[0.2, 1.1]$, we may want to calculate new control points $\mathbf{Q}_i$ and obtain a new curve $\mathbf{Q}(t)$ that has this shape when its parameter varies in the standard range $[0,1]$.

Our approach is to define the new curve $\mathbf{Q}(t)$ as $\mathbf{P}([b-a]t+a)$ and express the control points $\mathbf{Q}_i$ of $\mathbf{Q}(t)$ in terms of the control points $\mathbf{P}_i$ and a and b. We illustrate this technique with the cubic Bézier curve. This curve is given by Equation (6.8) and we can therefore write

$$\begin{aligned}
\mathbf{Q}(t) &= \mathbf{P}([b-a]t+a)\\
&= \big(([b-a]t+a)^3, ([b-a]t+a)^2, ([b-a]t+a), 1\big)\begin{pmatrix}-1&3&-3&1\\3&-6&3&0\\-3&3&0&0\\1&0&0&0\end{pmatrix}\begin{pmatrix}\mathbf{P}_0\\\mathbf{P}_1\\\mathbf{P}_2\\\mathbf{P}_3\end{pmatrix}\\
&= (t^3,t^2,t,1)\begin{pmatrix}(b-a)^3&0&0&0\\3a(b-a)^2&(b-a)^2&0&0\\3a^2(b-a)&2a(b-a)&b-a&0\\a^3&a^2&a&1\end{pmatrix}\begin{pmatrix}-1&3&-3&1\\3&-6&3&0\\-3&3&0&0\\1&0&0&0\end{pmatrix}\begin{pmatrix}\mathbf{P}_0\\\mathbf{P}_1\\\mathbf{P}_2\\\mathbf{P}_3\end{pmatrix}\\
&= \mathbf{T}(t)\cdot\mathbf{A}\cdot\mathbf{M}\cdot\mathbf{P}\\
&= \mathbf{T}(t)\cdot\mathbf{M}\cdot\mathbf{M}^{-1}\cdot\mathbf{A}\cdot\mathbf{M}\cdot\mathbf{P}\\
&= \mathbf{T}(t)\cdot\mathbf{M}\cdot(\mathbf{M}^{-1}\cdot\mathbf{A}\cdot\mathbf{M})\cdot\mathbf{P}\\
&= \mathbf{T}(t)\cdot\mathbf{M}\cdot\mathbf{B}\cdot\mathbf{P}\\
&= \mathbf{T}(t)\cdot\mathbf{M}\cdot\mathbf{Q},
\end{aligned}$$

where

$$\begin{aligned}
\mathbf{B} &= \mathbf{M}^{-1}\cdot\mathbf{A}\cdot\mathbf{M}\\
&= \begin{pmatrix}(1-a)^3 & 3(a-1)^2a & 3(1-a)a^2 & a^3\\
(a-1)^2(1-b) & (a-1)(-2a-b+3ab) & a(a+2b-3ab) & a^2b\\
(1-a)(-1+b)^2 & (b-1)(-a-2b+3ab) & b(2a+b-3ab) & ab^2\\
(1-b)^3 & 3(b-1)^2b & 3(1-b)b^2 & b^3\end{pmatrix}.
\end{aligned}\tag{6.22}$$

The four new control points $\mathbf{Q}_i$, $i=0,1,2,3$ are therefore obtained by selecting specific values for a and b, calculating matrix $\mathbf{B}$, and multiplying it by the column $\mathbf{P} = (\mathbf{P}_0,\mathbf{P}_1,\mathbf{P}_2,\mathbf{P}_3)^T$.

◇ **Exercise 6.15:** Show that the new curve $\mathbf{Q}(t)$ is independent of the particular coordinate system used.

Example: We select values $b=2$ and $a=1$. The new curve $\mathbf{Q}(t)$ will be identical to the part of $\mathbf{P}(t)$ from $\mathbf{P}(1)$ to $\mathbf{P}(2)$ (normally, of course, we don't calculate this part, but this example assumes that we are interested in it). Matrix $\mathbf{B}$ becomes, in this case

$$\mathbf{B} = \begin{pmatrix}0&0&0&1\\0&0&-1&2\\0&1&-4&4\\-1&6&-12&8\end{pmatrix}$$

(it is easy to verify that each row sums up to 1) and the new control points are

$$\begin{pmatrix}\mathbf{Q}_0\\ \mathbf{Q}_1\\ \mathbf{Q}_2\\ \mathbf{Q}_3\end{pmatrix} = \mathbf{B}\begin{pmatrix}\mathbf{P}_0\\ \mathbf{P}_1\\ \mathbf{P}_2\\ \mathbf{P}_3\end{pmatrix} = \begin{pmatrix}\mathbf{P}_3\\ -\mathbf{P}_2+2\mathbf{P}_3\\ \mathbf{P}_1-4\mathbf{P}_2+4\mathbf{P}_3\\ -\mathbf{P}_0+6\mathbf{P}_1-12\mathbf{P}_2+8\mathbf{P}_3\end{pmatrix}.$$

To understand the geometrical meaning of these points, we define three auxiliary points $\mathbf{R}_i$ as follows:

$$\begin{aligned}\mathbf{R}_1 &= \mathbf{P}_1 + (\mathbf{P}_1 - \mathbf{P}_0),\\ \mathbf{R}_2 &= \mathbf{P}_2 + (\mathbf{P}_2 - \mathbf{P}_1),\\ \mathbf{R}_3 &= \mathbf{R}_2 + (\mathbf{R}_2 - \mathbf{R}_1) = \mathbf{P}_0 - 4\mathbf{P}_1 + 4\mathbf{P}_2,\end{aligned}$$

and write the $\mathbf{Q}_i$'s in the form

$$\begin{aligned}\mathbf{Q}_0 &= \mathbf{P}_3,\\ \mathbf{Q}_1 &= \mathbf{P}_3 + (\mathbf{P}_3 - \mathbf{P}_2),\\ \mathbf{Q}_2 &= \mathbf{Q}_1 + (\mathbf{Q}_1 - \mathbf{R}_2) = \mathbf{P}_1 - 4\mathbf{P}_2 + 4\mathbf{P}_3,\\ \mathbf{Q}_3 &= \mathbf{Q}_2 + (\mathbf{Q}_2 - \mathbf{R}_3) = -\mathbf{P}_0 + 6\mathbf{P}_1 - 12\mathbf{P}_2 + 8\mathbf{P}_3.\end{aligned}$$

Figure 6.15 illustrates how the four new points $\mathbf{Q}_i$ are obtained from the four original points $\mathbf{P}_i$.

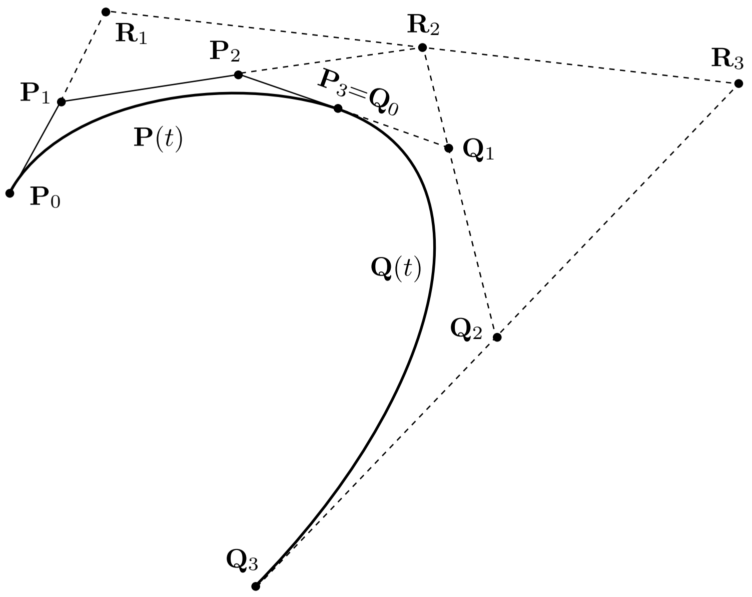


Figure 6.15: Control Points for the Case $[a, b] = [1, 2]$.

Example: We select $b = 2$ and $a = 0$. The new curve $\mathbf{Q}(t)$ will be identical to $\mathbf{P}(t)$ from $\mathbf{P}(0)$ to $\mathbf{P}(2)$. Matrix $\mathbf{B}$ becomes

$$\mathbf{B} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ -1 & 2 & 0 & 0 \\ 1 & -4 & 4 & 0 \\ -1 & 6 & -12 & 8 \end{pmatrix},$$

and the new control points $\mathbf{V}_i$ are

$$\begin{pmatrix} \mathbf{V}_0 \\ \mathbf{V}_1 \\ \mathbf{V}_2 \\ \mathbf{V}_3 \end{pmatrix} = \mathbf{B} \begin{pmatrix} \mathbf{P}_0 \\ \mathbf{P}_1 \\ \mathbf{P}_2 \\ \mathbf{P}_3 \end{pmatrix} = \begin{pmatrix} \mathbf{P}_0 \\ -\mathbf{P}_0 + 2\mathbf{P}_1 \\ \mathbf{P}_0 - 4\mathbf{P}_1 + 4\mathbf{P}_2 \\ -\mathbf{P}_0 + 6\mathbf{P}_1 - 12\mathbf{P}_2 + 8\mathbf{P}_3 \end{pmatrix},$$

and it is easy to see that they satisfy $\mathbf{V}_0 = \mathbf{P}_0$, $\mathbf{V}_1 = \mathbf{R}_1$, $\mathbf{V}_2 = \mathbf{R}_3$, and $\mathbf{V}_3 = \mathbf{Q}_3$.

◇ **Exercise 6.16:** (1) Calculate matrix $\mathbf{B}$ for $a = 1$ and $b = a + x$ (where x is positive); (2) calculate the four new control points $\mathbf{Q}_i$ as functions of the $\mathbf{P}_i$'s and of b; and (3) recalculate them for $x = 0.75$.

◇ **Exercise 6.17:** Calculate matrix $\mathbf{B}$ and the four new control points $\mathbf{Q}_i$ for $a = 0$ and $b = 0.5$ (the first half of the curve).

6.11 Cubic Bézier Segments with Tension

Adding a tension parameter to a cubic Bézier segment is done by manipulating tangent vectors similar to how tension is added to the Cardinal spline (Section 5.4). We use Hermite interpolation [Equation (4.7)] to calculate a PC segment that starts at point $\mathbf{P}_0$ and ends at point $\mathbf{P}_3$ and whose extreme tangent vectors are $s(\mathbf{P}_1 - \mathbf{P}_0)$ and $s(\mathbf{P}_3 - \mathbf{P}_2)$ [see Equation (6.23).]

◇ **Exercise 6.18:** Any set of four given control points $\mathbf{P}_0$, $\mathbf{P}_1$, $\mathbf{P}_2$, and $\mathbf{P}_3$ determines a unique (cubic) Bézier curve. Show that there is a Hermite curve that has an identical shape and is determined by the 4-tuple

$$(\mathbf{P}_0, \mathbf{P}_3, 3(\mathbf{P}_1 - \mathbf{P}_0), 3(\mathbf{P}_3 - \mathbf{P}_2)). \tag{6.23}$$

Substituting these values in Equation (4.7), we manipulate it so that it ends up looking like a cubic Bézier segment, Equation (6.8)

$$\begin{aligned} \mathbf{P}(t) &= (t^3, t^2, t, 1) \begin{pmatrix} 2 & -2 & 1 & 1 \\ -3 & 3 & -2 & -1 \\ 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 0 \end{pmatrix} \begin{pmatrix} \mathbf{P}_0 \\ \mathbf{P}_3 \\ s(\mathbf{P}_1 - \mathbf{P}_0) \\ s(\mathbf{P}_3 - \mathbf{P}_2) \end{pmatrix} \\ &= (t^3, t^2, t, 1) \begin{pmatrix} 2-s & s & -s & s-2 \\ 2s-3 & -2s & s & 3-s \\ -s & s & 0 & 0 \\ 1 & 0 & 0 & 0 \end{pmatrix} \begin{pmatrix} \mathbf{P}_0 \\ \mathbf{P}_1 \\ \mathbf{P}_2 \\ \mathbf{P}_3 \end{pmatrix}. \end{aligned} \tag{6.24}$$

A quick check verifies that Equation (6.24) reduces to the cubic Bézier segment, Equation (6.8), for $s=3$. This value is therefore considered the "neutral" or "standard" value of the tension parameter s. Since s controls the length of the tangent vectors, small values of s should produce the effects of higher tension and, in the extreme, the value $s=0$ should result in indefinite tangent vectors and in the curve segment becoming a straight line. To show this, we rewrite Equation (6.24) for $s=0$:

$$\begin{aligned}\mathbf{P}(t) &= (t^3,t^2,t,1)\begin{pmatrix}2&0&0&-2\\-3&0&0&3\\0&0&0&0\\1&0&0&0\end{pmatrix}\begin{pmatrix}\mathbf{P}_0\\\mathbf{P}_1\\\mathbf{P}_2\\\mathbf{P}_3\end{pmatrix}\\ &= (2t^3-3t^2+1)\mathbf{P}_0+(-2t^3+3t^2)\mathbf{P}_3.\end{aligned}$$

Substituting $T=3t^2-2t^3$ for t changes the expression above to the form $\mathbf{P}(T)=(\mathbf{P}_3-\mathbf{P}_0)T+\mathbf{P}_0$, i.e., a straight line from $\mathbf{P}(0)=\mathbf{P}_0$ to $\mathbf{P}(1)=\mathbf{P}_3$.

The tangent vector of Equation (6.24) is

$$\begin{aligned}\mathbf{P}^t(t) &= (3t^2,2t,1,0)\begin{pmatrix}2-s&s&-s&s-2\\2s-3&-2s&s&3-s\\-s&s&0&0\\1&0&0&0\end{pmatrix}\begin{pmatrix}\mathbf{P}_0\\\mathbf{P}_1\\\mathbf{P}_2\\\mathbf{P}_3\end{pmatrix} \\ &= \bigl(3t^2(2-s)+2t(2s-3)-s\bigr)\mathbf{P}_0+\bigl(3st^2-4st+s\bigr)\mathbf{P}_1\\ &\quad+\bigl(-3st^2+2st\bigr)\mathbf{P}_2+\bigl(3t^2(s-2)+2t(3-s)\bigr)\mathbf{P}_3.\end{aligned}\tag{6.25}$$

The extreme tangents are $\mathbf{P}^t(0)=s(\mathbf{P}_1-\mathbf{P}_0)$ and $\mathbf{P}^t(1)=s(\mathbf{P}_3-\mathbf{P}_2)$. Substituting $s=0$ in Equation (6.25) yields the tangent vector for the case of infinite tension (compare with Exercise 5.8)

$$\mathbf{P}^t(t)=6(t^2-t)\mathbf{P}_0-6(t^2-t)\mathbf{P}_3=6(t-t^2)(\mathbf{P}_3-\mathbf{P}_0). \tag{6.26}$$

◇ **Exercise 6.19:** Since the spline segment is a straight line in this case, its tangent vector should always point in the same direction. Use Equation (6.26) to show that this is so.

See also Section 7.4 for a discussion of cubic B-spline with tension.

We interrupt this program to increase dramatic tension.
—Joe Leahy (as the Announcer) in *Freakazoid!* (1995).

6.12 An Interpolating Bézier Curve: I

Any set of four control points $\mathbf{P}_1$, $\mathbf{P}_2$, $\mathbf{P}_3$, and $\mathbf{P}_4$ determines a unique Catmull–Rom segment that's a cubic polynomial going from point $\mathbf{P}_2$ to point $\mathbf{P}_3$. It turns out that such a segment can also be written as a four-point Bézier curve from $\mathbf{P}_2$ to $\mathbf{P}_3$. All that we have to do is find two points, $\mathbf{X}$ and $\mathbf{Y}$, located between $\mathbf{P}_2$ and $\mathbf{P}_3$, such that the Bézier curve based on $\mathbf{P}_2$, $\mathbf{X}$, $\mathbf{Y}$, and $\mathbf{P}_3$ will be identical to the Catmull–Rom segment. This turns out to be an easy task. We start with the expressions for a Catmull–Rom segment defined by $\mathbf{P}_1$, $\mathbf{P}_2$, $\mathbf{P}_3$, and $\mathbf{P}_4$, and for a four-point Bézier curve defined by $\mathbf{P}_2$, $\mathbf{X}$, $\mathbf{Y}$, and $\mathbf{P}_3$ [Equations (5.33) and (6.8)]:

$$(t^3,t^2,t,1)\begin{pmatrix}-0.5 & 1.5 & -1.5 & 0.5\\ 1 & -2.5 & 2 & -0.5\\ -0.5 & 0 & 0.5 & 0\\ 0 & 1 & 0 & 0\end{pmatrix}\begin{pmatrix}\mathbf{P}_1\\ \mathbf{P}_2\\ \mathbf{P}_3\\ \mathbf{P}_4\end{pmatrix},$$

$$(t^3,t^2,t,1)\begin{pmatrix}-1 & 3 & -3 & 1\\ 3 & -6 & 3 & 0\\ -3 & 3 & 0 & 0\\ 1 & 0 & 0 & 0\end{pmatrix}\begin{pmatrix}\mathbf{P}_2\\ \mathbf{X}\\ \mathbf{Y}\\ \mathbf{P}_3\end{pmatrix}.$$

These have to be equal for each power of t, which yields the four equations

$$\begin{aligned}
-0.5\mathbf{P}_1+1.5\mathbf{P}_2-1.5\mathbf{P}_3+0.5\mathbf{P}_4 &= -\mathbf{P}_2+3\mathbf{X}-3\mathbf{Y}+\mathbf{P}_3,\\
\mathbf{P}_1-2.5\mathbf{P}_2+2.0\mathbf{P}_3-0.5\mathbf{P}_4 &= 3\mathbf{P}_2-6\mathbf{X}+3\mathbf{Y},\\
-0.5\mathbf{P}_1\qquad +0.5\mathbf{P}_3\qquad &= -3\mathbf{P}_2+3\mathbf{X},\\
\mathbf{P}_2\qquad &= \mathbf{P}_2.
\end{aligned}$$

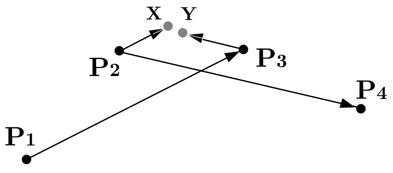


Figure 6.16: Calculating Points $\mathbf{X}$ and $\mathbf{Y}$.

These are easily solved to produce

$$\mathbf{X}=\mathbf{P}_2+\frac{1}{6}(\mathbf{P}_3-\mathbf{P}_1)\quad\text{and}\quad\mathbf{Y}=\mathbf{P}_3-\frac{1}{6}(\mathbf{P}_4-\mathbf{P}_2).\tag{6.27}$$

The difference $(\mathbf{P}_3-\mathbf{P}_1)$ is the vector from $\mathbf{P}_1$ to $\mathbf{P}_3$. Thus, point $\mathbf{X}$ is obtained by adding 1/6 of this vector to point $\mathbf{P}_2$ (Figure 6.16). Similarly, $\mathbf{Y}$ is obtained by subtracting 1/6 of the difference $(\mathbf{P}_4-\mathbf{P}_2)$ from point $\mathbf{P}_3$.

This simple result suggests a novel approach to the problem of interactive curve design, an approach that combines the useful features of both cubic splines and Bézier curves. A cubic spline passes through the (data) points but is not highly interactive. It can be edited only by modifying the two extreme tangent vectors. A Bézier curve does not pass through the (control) points, but it is easy to manipulate and edit by moving the points. The new approach constructs an *interpolating* Bézier curve in the following steps:

1. The user is asked to input n points, through which the final curve will pass.

2. The program divides the points into overlapping groups of four points each and applies Equation (6.27) to compute two auxiliary points $\mathbf{X}$ and $\mathbf{Y}$ for each group.

3. A Bézier segment is then drawn from the second to the third point of each group, using points $\mathbf{X}$ and $\mathbf{Y}$ as its other two control points. Note that points $\mathbf{Y}$ and $\mathbf{P}_3$ of a group are on a straight line with point $\mathbf{X}$ of the next group. This guarantees that the individual segments will connect smoothly.

4. It is also possible to draw a Bézier segment from $\mathbf{P}_1$ to $\mathbf{P}_2$ (and, similarly, from $\mathbf{P}_{n-1}$ to $\mathbf{P}_n$). This segment uses the two auxiliary control points $\mathbf{X} = \mathbf{P}_1 + \frac{1}{6}(\mathbf{P}_2 - \mathbf{P}_1)$ and $\mathbf{Y} = \mathbf{P}_2 - \frac{1}{6}(\mathbf{P}_3 - \mathbf{P}_1)$.

Users find it natural to specify such a curve, because they don't have to worry about the positions of the control points. The curve consists of $n - 1$ segments and the two auxiliary control points of each segment are calculated automatically.

Such a curve is usually pleasing to the eye and rarely needs to be edited. However, if it is not satisfactory, it can be modified by moving the auxiliary control points. There are $2(n - 1)$ of them, which allows for flexible control. A good program should display the auxiliary points and should make it easy for the user to grab and move any of them.

The well-known drawing program *Adobe Illustrator* [Adobe 04] uses a similar approach. The user specifies points with the mouse. At each point $\mathbf{P}_i$, the user presses the mouse button to fix $\mathbf{P}_i$, then drags the mouse before releasing the button, which defines two symmetrical points, $\mathbf{X}$ (following $\mathbf{P}_i$) and $\mathbf{Y}$ (preceding it). Releasing the button is a signal to the program to draw the segment from $\mathbf{P}_{i-1}$ to $\mathbf{P}_i$ (Figure 6.17).

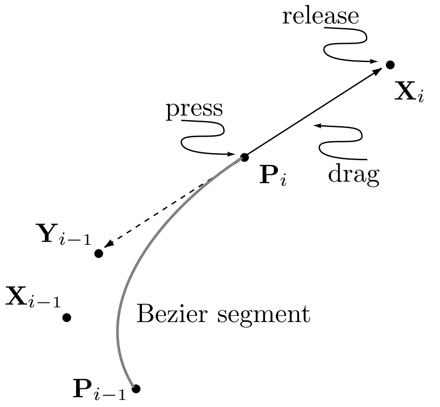


Figure 6.17: Construction of $\mathbf{X}_i$ and $\mathbf{Y}_i$ by Click and Drag.

Example: We apply this method to the six points $\mathbf{P}_0 = (1/2, 0)$, $\mathbf{P}_1 = (1/2, 1/2)$, $\mathbf{P}_2 = (0, 1)$, $\mathbf{P}_3 = (1, 3/2)$, $\mathbf{P}_4 = (3/2, 1)$, and $\mathbf{P}_5 = (1, 1/2)$. The six points yield three curve segments and the main step is to calculate the two intermediate points for each of the three segments. This is trivial and it results in:

$$\mathbf{X}_1 = \mathbf{P}_1 + (\mathbf{P}_2 - \mathbf{P}_0)/6 = (5/12, 2/3), \quad \mathbf{Y}1 = \mathbf{P}_2 - (\mathbf{P}_3 - \mathbf{P}_1)/6 = (-1/12, 5/6),$$
$$\mathbf{X}_2 = \mathbf{P}_2 + (\mathbf{P}_3 - \mathbf{P}_1)/6 = (1/12, 7/6), \quad \mathbf{Y}2 = \mathbf{P}_3 - (\mathbf{P}_4 - \mathbf{P}_2)/6 = (3/4, 3/2),$$
$$\mathbf{X}_3 = \mathbf{P}_3 + (\mathbf{P}_4 - \mathbf{P}_2)/6 = (5/4, 3/2), \quad \mathbf{Y}3 = \mathbf{P}_4 - (\mathbf{P}_5 - \mathbf{P}_3)/6 = (3/2, 7/6).$$

Once the points are available, the three segments can easily be calculated. Each is a cubic Bézier segment based on a group of four points. The groups are

$$[\mathbf{P}_1, \mathbf{X}_1, \mathbf{Y}_1, \mathbf{P}_2], \quad [\mathbf{P}_2, \mathbf{X}_2, \mathbf{Y}_2, \mathbf{P}_3], \quad [\mathbf{P}_3, \mathbf{X}_3, \mathbf{Y}_3, \mathbf{P}_4],$$

and the three curve segments are

$$\begin{aligned}
\mathbf{P}_1(t) &= (1-t)^3\mathbf{P}_1 + 3t(1-t)^2\mathbf{X}_1 + 3t^2(1-t)\mathbf{Y}_1 + t^3\mathbf{P}_2\\
&= \big((2 - t - 5t^2 + 4t^3)/4, (1+t)/2\big),\\
\mathbf{P}_2(t) &= (1-t)^3\mathbf{P}_2 + 3t(1-t)^2\mathbf{X}_2 + 3t^2(1-t)\mathbf{Y}_2 + t^3\mathbf{P}_3\\
&= \big((t + 7t^2 - 4t^3)/4, (2 + t + t^2 - t^3)/2\big),\\
\mathbf{P}_3(t) &= (1-t)^3\mathbf{P}_3 + 3t(1-t)^2\mathbf{X}_3 + 3t^2(1-t)\mathbf{Y}_3 + t^3\mathbf{P}_4\\
&= \big((4 + 3t - t^3)/4, (3 - 2t^2 + t^3)/2\big).
\end{aligned}$$

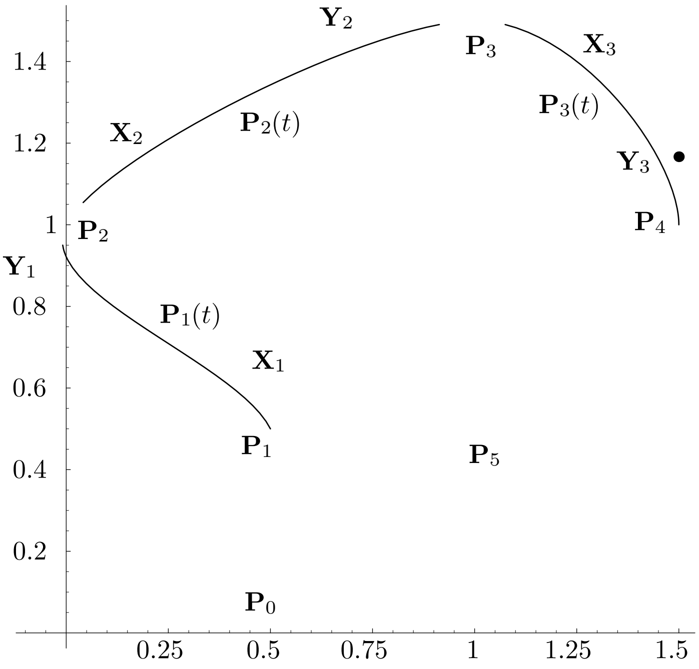
The 12 points and the three segments are shown in Figure 6.18 (where the segments have been separated intentionally), as well as the code for the entire example.

6.13 An Interpolating Bézier Curve: II

The approach outlined in this section calculates an interpolating Bézier curve by solving equations. Given a set of $n+1$ data points $\mathbf{Q}_0$, $\mathbf{Q}_1$,..., $\mathbf{Q}_n$, we select $n+1$ values t_i such that $\mathbf{P}(t_i) = \mathbf{Q}_i$. We require that whenever t reaches one of the values t_i, the curve will pass through a point $\mathbf{Q}_i$. The values t_i don't have to be equally spaced, which provides control over the "speed" of the curve. All that's needed to calculate the curve is to compute the right set of $n+1$ control points $\mathbf{P}_i$. This is done by setting and solving the set of $n+1$ linear equations $\mathbf{P}(t_0) = \mathbf{Q}_0$, $\mathbf{P}(t_1) = \mathbf{Q}_1$,..., $\mathbf{P}(t_n) = \mathbf{Q}_n$ that's expressed in matrix notation as follows:

$$\begin{pmatrix} B_{n,0}(t_0) & B_{n,1}(t_0) & \dots & B_{n,n}(t_0)\\ B_{n,0}(t_1) & B_{n,1}(t_1) & \dots & B_{n,n}(t_1)\\ \vdots & \vdots & \ddots & \vdots\\ B_{n,0}(t_n) & B_{n,1}(t_n) & \dots & B_{n,n}(t_n) \end{pmatrix} \begin{pmatrix} \mathbf{P}_0\\ \mathbf{P}_1\\ \vdots\\ \mathbf{P}_n \end{pmatrix} = \begin{pmatrix} \mathbf{Q}_0\\ \mathbf{Q}_1\\ \vdots\\ \mathbf{Q}_n \end{pmatrix}. \tag{6.28}$$

This set of equations can be expressed as $\mathbf{MP} = \mathbf{Q}$ and it is easily solved by inverting $\mathbf{M}$ numerically. The solution is $\mathbf{P} = \mathbf{M}^{-1}\mathbf{Q}$. If we select $t_0 = 0$, the top row of Equation (6.28) yields $\mathbf{P}_0 = \mathbf{Q}_0$. Similarly, if we select $t_n = 1$, the bottom row of



```
(* Interpolating Bezier Curve: I *)
Clear[p0,p1,p2,p3,p4,p5,x1,x2,x3,y1,y2,y3,c1,c2,c3,g1,g2,g3,g4];
p0={1/2,0}; p1={1/2,1/2}; p2={0,1};
 p3={1,3/2}; p4={3/2,1}; p5={1,1/2};
x1=p1+(p2-p0)/6;
x2=p2+(p3-p1)/6;
x3=p3+(p4-p2)/6;
y1=p2-(p3-p1)/6;
y2=p3-(p4-p2)/6;
y3=p4-(p5-p3)/6;
c1[t_]:=Simplify[(1-t)^3 p1+3t(1-t)^2 x1+3t^2(1-t) y1+t^3 p2]
c2[t_]:=Simplify[(1-t)^3 p2+3t(1-t)^2 x2+3t^2(1-t) y2+t^3 p3]
c3[t_]:=Simplify[(1-t)^3 p3+3t(1-t)^2 x3+3t^2(1-t) y3+t^3 p4]
g1=ListPlot[{p0,p1,p2,p3,p4,p5,x1,x2,x3,y1,y2,y3},
 Prolog->AbsolutePointSize[4], PlotRange->All,
 AspectRatio->Automatic, DisplayFunction->Identity]
g2=ParametricPlot[c1[t], {t,0,.9}, DisplayFunction->Identity]
g3=ParametricPlot[c2[t], {t,0.1,.9}, DisplayFunction->Identity]
g4=ParametricPlot[c3[t], {t,0.1,1}, DisplayFunction->Identity]
Show[g1,g2,g3,g4, DisplayFunction->$DisplayFunction]
```

Figure 6.18: An Interpolating Bézier Curve.

Equation (6.28) yields $\mathbf{P}_n = \mathbf{Q}_n$. This decreases the number of equations from $n+1$ to $n-1$.

The disadvantage of this approach is that any changes in the t_i's require a recalculation of $\mathbf{M}$ and, consequently, of $\mathbf{M}^{-1}$.

If controlling the speed of the curve is not important, we can select the $n+1$ equally-spaced values $t_i = i/n$. Equation (6.28) can now be written

$$\begin{pmatrix} B_{n,0}(0/n) & B_{n,1}(0/n) & \ldots & B_{n,n}(0/n) \\ B_{n,0}(1/n) & B_{n,1}(1/n) & \ldots & B_{n,n}(1/n) \\ \vdots & \vdots & \ddots & \vdots \\ B_{n,0}(n/n) & B_{n,1}(n/n) & \ldots & B_{n,n}(n/n) \end{pmatrix} \begin{pmatrix} \mathbf{P}_0 \\ \mathbf{P}_1 \\ \vdots \\ \mathbf{P}_n \end{pmatrix} = \begin{pmatrix} \mathbf{Q}_0 \\ \mathbf{Q}_1 \\ \vdots \\ \mathbf{Q}_n \end{pmatrix}. \tag{6.29}$$

Now, if the data points $\mathbf{Q}_i$ are moved, matrix $\mathbf{M}$ (or, rather, $\mathbf{M}^{-1}$) doesn't have to be recalculated. If we number the rows and columns of $\mathbf{M}$ 0 through n, then a general element of $\mathbf{M}$ is given by

$$M_{ij} = B_{n,j}(i/n) = \binom{n}{j}(i/n)^j(1-i/n)^{n-j} = \frac{n!(n-i)^{n-j}i^j}{j!(n-j)!n^n}.$$

Such elements can be calculated, if desired, as exact rational integers, instead of (approximate) floating-point numbers.

Example: We use Equation (6.29) to compute the interpolating Bézier curve that passes through the four points $\mathbf{Q}_0 = (0,0)$, $\mathbf{Q}_1 = (1,1)$, $\mathbf{Q}_2 = (2,1)$, and $\mathbf{Q}_3 = (3,0)$. Since the curve has to pass through the first and last point, we get $\mathbf{P}_0 = \mathbf{Q}_0 = (0,0)$ and $\mathbf{P}_3 = \mathbf{Q}_3 = (3,0)$. Since the four given points are equally spaced, it makes sense to assume that $\mathbf{P}(1/3) = \mathbf{Q}_1$ and $\mathbf{P}(2/3) = \mathbf{Q}_2$. We therefore end up with the two equations

$$\begin{aligned} 3(1/3)(1-1/3)^2\mathbf{P}_1 + 3(1/3)^2(1-1/3)\mathbf{P}_2 + (1/3)^3(3,0) &= (1,1), \\ 3(2/3)(1-2/3)^2\mathbf{P}_1 + 3(2/3)^2(1-2/3)\mathbf{P}_2 + (2/3)^3(3,0) &= (2,1), \end{aligned}$$

that are solved to yield $\mathbf{P}_1 = (1,3/2)$ and $\mathbf{P}_2 = (2,3/2)$. The curve is

$$\mathbf{P}(t) = (1-t)^3(0,0) + 3t(1-t)^2(1,3/2) + 3t^2(1-t)(2,3/2) + t^3(3,0).$$

◇ **Exercise 6.20:** Plot the curve and the eight points.

6.14 Nonparametric Bézier Curves

The explicit representation of a curve (Section 1.3) has the familiar form $y = f(x)$. The Bézier curve is, of course, parametric, but it can be represented in a nonparametric form, similar to explicit curves. Given $n+1$ real values (not points) P_i, we start with the polynomial $c(t) = \sum P_i B_{ni}(t)$ and employ the identity

$$\sum_{i=0}^{n} (i/n) B_{ni}(t) = t \tag{6.30}$$

to create the curve

$$\mathbf{P}(t) = \big(t, c(t)\big) = \sum_{i=0}^{n} (i/n, P_i) B_{ni}(t).$$

(This identity is satisfied by the Bernstein polynomials and can be proved by induction.) It is clear that this version of the curve is defined by the control points $(i/n, P_i)$ which are equally-spaced on the x axis.

This version of the Bézier curve exists only for two-dimensional curves. In the general case, where t varies in the interval $[a, b]$, the control points are $\big((a + i(b - a))/n, P_i\big)$.

6.15 Rational Bézier Curves

The rational Bézier curve is an extension of the original Bézier curve [Equation (6.5)] to

$$\mathbf{P}(t) = \frac{\sum_{i=0}^{n} w_i \mathbf{P}_i B_{n,i}(t)}{\sum_{j=0}^{n} w_j B_{n,j}(t)} = \sum_{i=0}^{n} \mathbf{P}_i \left[\frac{w_i B_{n,i}(t)}{\sum_{j=0}^{n} w_j B_{n,j}(t)} \right] = \sum_{i=0}^{n} \mathbf{P}_i R_{n,i}(t), \qquad 0 \le t \le 1.$$

The new weight functions $R_{n,i}(t)$ are ratios of polynomials (which is the reason for the term *rational*) and they also depend on weights w_i that act as additional parameters that control the shape of the curve. Note that negative weights might lead to a zero denominator, which is why nonnegative weights are normally used. A rational curve seems unnecessarily complicated (and for many applications, it is), but it has the following advantages:

1. It is invariant under projections. Section 6.4 mentions that the Bézier curve is invariant under affine transformations. If we want to rotate, reflect, scale, or shear such a curve, we can apply the affine transformation to the control points, then use the new points to compute the transformed curve. The Bézier curve, however, is not invariant under projections. If we compute a three-dimensional Bézier curve and project every point of the curve by a perspective projection, we end up with a plane curve $\mathbf{P}(t)$. If we then project the three-dimensional control points and compute a plane Bézier curve $\mathbf{Q}(t)$ from the projected, two-dimensional points, the two curves $\mathbf{P}(t)$ and $\mathbf{Q}(t)$ will generally be different. One advantage of the rational Bézier curve is its invariance under projections.

2. The rational Bézier curve provides for accurate control of curve shape, such as precise representation of conic sections (Appendix A).

Section 7.5 shows that the Bézier curve is a special case of the B-spline curve. As a result, many current software systems use the rational B-spline (Section 7.14) when rational curves are required. Such a system can produce the rational Bézier curve as a special case.

Here is a quick example showing how the rational Bézier curve can be useful. Given the three points $\mathbf{P}_0 = (1,0)$, $\mathbf{P}_1 = (1,1)$, and $\mathbf{P}_2 = (0,1)$, The Bézier curve defined by the points is quadratic and is therefore a parabola $\mathbf{P}(t) = (1-t)^2\mathbf{P}_0 + 2t(1-t)\mathbf{P}_1 + t^2\mathbf{P}_2 = (1-t^2, 2t(1-t))$, but the rational Bézier curve with weights $w_0 = w_1 = 1$ and $w_2 = 2$ results in the more complex expression

$$\mathbf{P}(t) = \frac{(1-t)^2\mathbf{P}_0 + 2t(1-t)\mathbf{P}_1 + 2t^2\mathbf{P}_2}{(1-t)^2 + 2t(1-t) + 2t^2} = \left(\frac{1-t^2}{1+t^2}, \frac{2t}{1+t^2}\right)$$

which is a circle, as illustrated by Figure 1.6a.

In general, a quadratic rational Bézier curve with weights $w_0 = w_2 = 1$ is a parabola when $w_1 = 1$, an ellipse for $w_1 < 1$, and a hyperbola for $w_1 > 1$. A quarter circle is obtained when $w_1 = \cos(\alpha/2)$ where α is the angle formed by the three control points $\mathbf{P}_0$, $\mathbf{P}_1$, and $\mathbf{P}_2$ (the control points must also be placed as the three corners of an isosceles triangle). Page 261 of [Beach 91] proves this construction for the special case $\alpha = 90°$.

Appendix A shows, among other features, that the canonical ellipse is represented as the rational expression

$$\left(a\frac{1-t^2}{1+t^2}, b\frac{2t}{1+t^2}\right), \qquad -\infty < t < \infty, \tag{A.7}$$

and the canonical hyperbola is represented as the rational

$$\left(a\frac{1+t^2}{1-t^2}, b\frac{2t}{1-t^2}\right), \qquad -\infty < t < \infty. \tag{A.8}$$

Accurate control of the shape of the curve is provided by either moving the control points or varying the weights, and Figure 6.19 illustrates the different responses of the curve to these changes. Part (a) of the figure shows four curves where weight w_1 is increased from 1 to 4. The curve is pulled toward $\mathbf{P}_1$ in such a way that individual points on the curve *converge* at $\mathbf{P}_1$. In contrast, part (b) of the figure illustrates how the curve behaves when $\mathbf{P}_1$ itself is moved (while all the weights remain set to 1). The curve is again pulled toward $\mathbf{P}_1$, but in such a way that every point on the curve moves in the same direction as $\mathbf{P}_1$ itself.

◇ **Exercise 6.21:** Use mathematical software to compute Figure 6.19 or a similar illustration.

Section 6.22 extends the techniques presented here to rectangular Bézier surface patches.

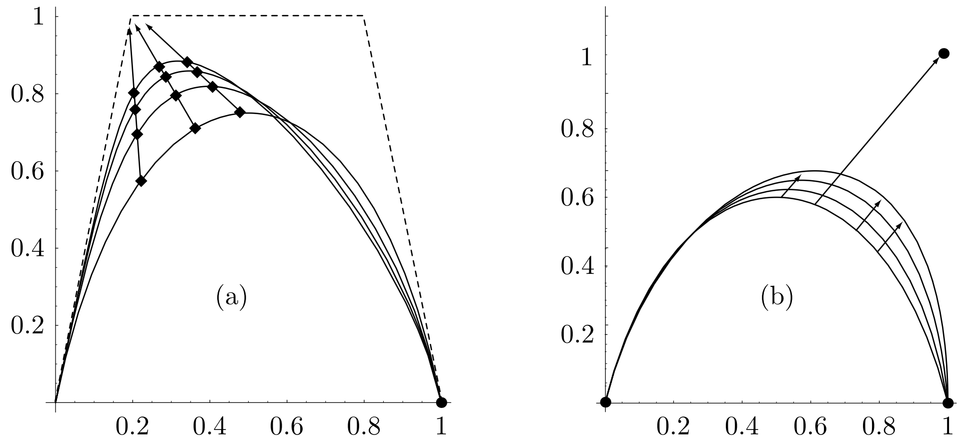


Figure 6.19: (a) Varying Weights and (b) Moving Points in a Rational Bézier Curve.

6.16 Rectangular Bézier Surfaces

The Bézier surface patch, like its relative the Bézier curve, is popular and is commonly used in practice. We discuss the rectangular and the triangular Bézier surface methods, and this section covers the former.

We start with an $(m+1)\times(n+1)$ grid of control points arranged in a roughly rectangular grid

$$\begin{matrix} \mathbf{P}_{m,0} & \mathbf{P}_{m,1} & \dots & \mathbf{P}_{m,n} \\ \vdots & \vdots & & \vdots \\ \mathbf{P}_{1,0} & \mathbf{P}_{1,1} & \dots & \mathbf{P}_{1,n} \\ \mathbf{P}_{0,0} & \mathbf{P}_{0,1} & \dots & \mathbf{P}_{0,n} \end{matrix}$$

and construct the rectangular Bézier surface patch for the points by applying the technique of Cartesian product (Section 1.9) to the Bézier curve. Equation (1.28) produces

$$\begin{aligned} \mathbf{P}(u,w) &= \sum_{i=0}^{m}\sum_{j=0}^{n} B_{m,i}(u)\mathbf{P}_{i,j}B_{n,j}(w) \\ &= (B_{m,0}(u), B_{m,1}(u), \dots, B_{m,m}(u))\,\mathbf{P}\begin{pmatrix} B_{n,0}(w) \\ B_{n,1}(w) \\ \vdots \\ B_{n,n}(w) \end{pmatrix} \\ &= \mathbf{B}_m(u)\,\mathbf{P}\,\mathbf{B}_n(w), \end{aligned} \tag{6.31}$$

where $$\mathbf{P} = \begin{pmatrix} \mathbf{P}_{0,0} & \mathbf{P}_{0,1} & \dots & \mathbf{P}_{0,n} \\ \mathbf{P}_{1,0} & \mathbf{P}_{1,1} & \dots & \mathbf{P}_{1,n} \\ \vdots & \vdots & \ddots & \vdots \\ \mathbf{P}_{m,0} & \mathbf{P}_{m,1} & \dots & \mathbf{P}_{m,n} \end{pmatrix}.$$

The surface can also be expressed, by analogy with Equation (6.9), as

$$\mathbf{P}(u,w)=\mathbf{UNPN}^T\mathbf{W}^T, \tag{6.32}$$

where $\mathbf{U}=(u^m,u^{m-1},\ldots,u,1)$, $\mathbf{W}=(w^n,w^{n-1},\ldots,w,1)$, and $\mathbf{N}$ is defined by Equation (6.10).

Notice that both $\mathbf{P}(u_0,w)$ and $\mathbf{P}(u,w_0)$ (for constants u_0 and w_0) are Bézier curves on the surface. A Bézier curve is defined by $n+1$ control points, it passes through the two extreme points, and employs the interior points to determine its shape. Similarly, a rectangular Bézier surface patch is defined by a rectangular grid of $(m+1)\times(n+1)$ control points, it is anchored at the four corner points and employs the other grid points to determine its shape.

Figure 6.20 is an example of a biquadratic Bézier surface patch with the *Mathematica* code that generated it. Notice how the surface is anchored at the four corner points and how the other control points pull the surface toward them.

Example: Given the six three-dimensional points

$$\begin{matrix}\mathbf{P}_{10} & \mathbf{P}_{11} & \mathbf{P}_{12}\\ \mathbf{P}_{00} & \mathbf{P}_{01} & \mathbf{P}_{02}\end{matrix}$$

the corresponding Bézier surface is generated in the following three steps:

1. Find the orders m and n of the surface. Since the points are numbered starting from 0, the two orders of the surface are $m=1$ and $n=2$.

2. Calculate the weight functions $B_{1i}(w)$ and $B_{2j}(u)$. For $m=1$, we get

$$B_{1i}(w)=\binom{1}{i}w^i(1-w)^{1-i},$$

which yields the two functions

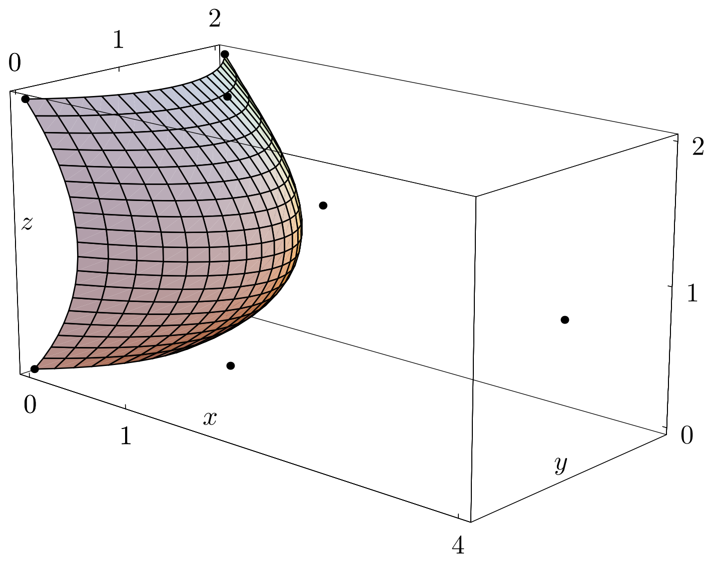
$$B_{10}(w)=\binom{1}{0}w^0(1-w)^{1-0}=1-w,\quad B_{11}(w)=\binom{1}{1}w^1(1-w)^{1-1}=w.$$

For $n=2$, we get

$$B_{2j}(u)=\binom{2}{j}u^j(1-u)^{2-j},$$

which yields the three functions

$$\begin{aligned}B_{20}(u)=&\binom{2}{0}u^0(1-u)^{2-0}=(1-u)^2,\\ B_{21}(u)=&\binom{2}{1}u^1(1-u)^{2-1}=2u(1-u),\\ B_{22}(u)=&\binom{2}{2}u^2(1-u)^{2-2}=u^2.\end{aligned}$$



```
(* biquadratic bezier surface patch *)
Clear[pwr,bern,spnts,n,bzSurf,g1,g2];
n=2;
<<:Graphics:ParametricPlot3D.m
spnts={{{0,0,0},{1,0,1},{0,0,2}},
 {{1,1,0},{4,1,1},{1,1,2}}, {{0,2,0},{1,2,1},{0,2,2}}};
(* Handle Indeterminate condition *)
pwr[x_,y_]:=If[x==0 && y==0, 1, x^y];
bern[n_,i_,u_]:=Binomial[n,i]pwr[u,i]pwr[1-u,n-i]
bzSurf[u_,w_]:=Sum[bern[n,i,u] spnts[[i+1,j+1]] bern[n,j,w],
 {i,0,n}, {j,0,n}]
g1=ParametricPlot3D[bzSurf[u,w],{u,0,1}, {w,0,1},
 Ticks->{{0,1,4},{0,1,2},{0,1,2}},
 Compiled->False, DisplayFunction->Identity];
g2=Graphics3D[{AbsolutePointSize[3],
 Table[Point[spnts[[i,j]]],{i,1,n+1},{j,1,n+1}]}];
Show[g1,g2, ViewPoint->{2.783, -3.090, 1.243}, PlotRange->All,
 DefaultFont->{"cmr10", 10}, DisplayFunction->$DisplayFunction];
```

Figure 6.20: A Biquadratic Bézier Surface Patch.

3. Substitute the weight functions in the general expression for the surface [Equation (6.31)]:

$$\begin{aligned}\mathbf{P}(u,w) &= \sum_{i=0}^{1}\sum_{j=0}^{2} B_{1i}(w)\mathbf{P}_{ij}B_{2j}(u)\\ &= B_{10}(w)\sum_{j=0}^{2}\mathbf{P}_{0j}B_{2j}(u) + B_{11}(w)\sum_{j=0}^{2}\mathbf{P}_{1j}B_{2j}(u)\\ &= (1-w)\left[\mathbf{P}_{00}B_{20}(u) + \mathbf{P}_{01}B_{21}(u) + \mathbf{P}_{02}B_{22}(u)\right]\\ &\quad + w\left[\mathbf{P}_{10}B_{20}(u) + \mathbf{P}_{11}B_{21}(u) + \mathbf{P}_{12}B_{22}(u)\right]\end{aligned}$$

$$\begin{aligned}
&= (1-w)\left[\mathbf{P}_{00}(1-u)^2 + \mathbf{P}_{01}2u(1-u) + \mathbf{P}_{02}u^2\right]\\
&\quad + w\left[\mathbf{P}_{10}(1-u)^2 + \mathbf{P}_{11}2u(1-u) + \mathbf{P}_{12}u^2\right]\\
&= \mathbf{P}_{00}(1-w)(1-u)^2 + \mathbf{P}_{01}(1-w)2u(1-u) + \mathbf{P}_{02}(1-w)u^2\\
&\quad + \mathbf{P}_{10}w(1-u)^2 + \mathbf{P}_{11}w2u(1-u) + \mathbf{P}_{12}wu^2. \qquad (6.33)
\end{aligned}$$

The final expression is linear in w since the surface is defined by just two points in the w direction. Surface lines in this direction are straight. In the u direction, where the surface is defined by three points, each line is a polynomial of degree 2 in u. This expression can also be written in the form

$$(1-w)\sum B_{2,i}(u)\mathbf{P}_{0i} + w\sum B_{2,i}(u)\mathbf{P}_{1i} = (1-w)\mathbf{P}(u,0) + w\mathbf{P}(u,1),$$

which is a lofted surface [Equation (2.14)].

A good technique to check the final expression is to calculate it for the four values $(u,w) = (0,0)$, $(0,1)$, $(1,0)$, and $(1,1)$. This should yield the coordinates of the four original corner points.

The entire surface can now be easily displayed, as a wire frame, by performing two loops. One draws curves in the u direction and the other draws the curves in the w direction. Notice that the expression of the patch is the same regardless of the particular points used. The user may change the points to modify the surface, and the new surface can be displayed (Figure 6.21) by calculating Equation (6.33).

◇ **Exercise 6.22:** Given the 3×4 array of control points

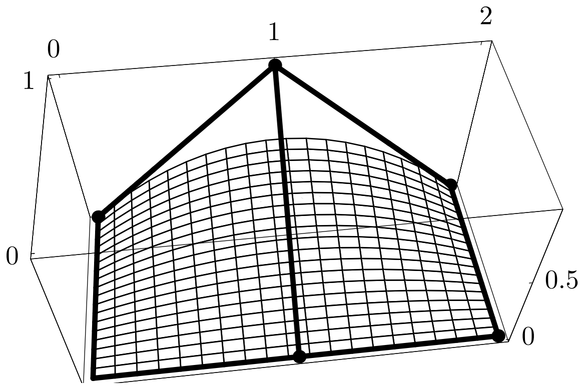
$$\begin{array}{llll}
\mathbf{P}_{20} = (0,2,0) & \mathbf{P}_{21} = (1,2,1) & \mathbf{P}_{22} = (2,2,1) & \mathbf{P}_{23} = (3,2,0)\\
\mathbf{P}_{10} = (0,1,0) & \mathbf{P}_{11} = (1,1,1) & \mathbf{P}_{12} = (2,1,1) & \mathbf{P}_{13} = (3,1,0)\\
\mathbf{P}_{00} = (0,0,0) & \mathbf{P}_{01} = (1,0,1) & \mathbf{P}_{02} = (2,0,1) & \mathbf{P}_{03} = (3,0,0),
\end{array}$$

calculate the order-2×3 Bézier surface patch defined by them.

Notice that the order-2×2 Bézier surface patch defined by only four control points is a bilinear patch. Its form is given by Equation (2.8).

6.16.1 Scaffolding Construction

The scaffolding construction (or de Casteljau algorithm) of Section 6.6 can be directly extended to the rectangular Bézier patch. Figure 6.22 illustrates the principle. Part (a) of the figure shows a rectangular Bézier patch defined by 3×4 control points (the circles). The de Casteljau algorithm for curves is applied to each row of three points to compute two intermediate points (the squares), followed by a final point (the triangle). The final point is located on the Bézier curve defined by the row of three points. The result of applying the de Casteljau algorithm to the four rows is four points (the triangles). The algorithm is now applied to those four points (Figure 6.22b) to compute one point (the heavy circle) that's located both on the curve defined by the four (black triangle) points and on the Bézier surface patch defined by the 3×4 control points. (This is one of the many curve algorithms that can be directly extended to surfaces.)



```
(* A Bezier surface example. Given the six two-dimensional... *)
Clear[pnts,b1,b2,g1,g2,vlines,hlines];
pnts={{{0,1,0},{1,1,1},{2,1,0}},{{0,0,0},{1,0,0},{2,0,0}}};
b1[w_]:={1-w,w}; b2[u_]:={(1-u)^2,2u(1-u),u^2};
comb[i_]:=(b1[w].pnts)[[i]] b2[u][[i]];
g1=ParametricPlot3D[comb[1]+comb[2]+comb[3], {u,0,1},{w,0,1}, Compiled->False,
DefaultFont->{"cmr10", 10}, DisplayFunction->Identity,
 AspectRatio->Automatic, Ticks->{{0,1,2},{0,1},{0,.5}}];
g2=Graphics3D[{AbsolutePointSize[5],
 Table[Point[pnts[[i,j]]],{i,1,2},{j,1,3}]}];
vlines=Graphics3D[{AbsoluteThickness[2],
 Table[Line[{pnts[[1,j]],pnts[[2,j]]}], {j,1,3}]}];
hlines=Graphics3D[{AbsoluteThickness[2],
 Table[Line[{pnts[[i,j]],pnts[[i,j+1]]}], {i,1,2}, {j,1,2}]}];
Show[g1,g2,vlines,hlines, ViewPoint->{-0.139, -1.179, 1.475},
 DisplayFunction->$DisplayFunction, PlotRange->All, Shading->False,
 DefaultFont->{"cmr10", 10}];
```

Figure 6.21: A Lofted Bézier Surface Patch.

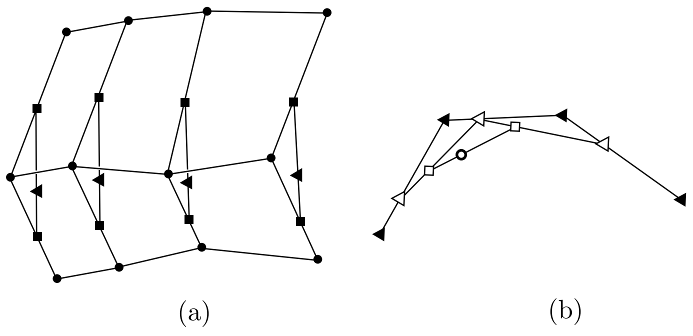


Figure 6.22: Scaffolding in a Rectangular Bézier Patch.

Referring to Equation (6.31), we can summarize this process as follows:

1. Construct the $n+1$ curves

$$\mathbf{P}_j(u) = \sum_{i=0}^{m} B_{mi}(u)\mathbf{P}_{ij}, \quad j = 0, 1, \ldots, n.$$

2. Apply the de Casteljau algorithm to each curve to end up with $n+1$ points, one on each curve.

3. Apply the same algorithm to the $n+1$ points to end up with one point.

Alternatively, we can first construct the $m+1$ curves

$$\mathbf{P}_i(w) = \sum_{j=0}^{n} \mathbf{P}_{ij} B_{nj}(w), \quad i = 0, 1, \ldots, m,$$

then apply the de Casteljau algorithm to each curve to end up with $m+1$ points, and finally apply the same algorithm to the $m+1$ points, and end up with one point.

6.17 Subdividing Rectangular Patches

A rectangular Bézier patch is computed from a given rectangular array of $m \times n$ control points. If there are not enough points, the patch may not have the right shape. Just adding points is not a good solution because this changes the shape of the surface, forcing the designer to start reshaping it from the beginning. A better solution is to subdivide the patch into four connected surface patches, each based on $m \times n$ control points. The technique described here is similar to that presented in Section 6.8 for subdividing the Bézier curve. It employs the scaffolding construction of Section 6.6.

Figure 6.22a shows a grid of 4×3 control points. The first step in subdividing the surface patch defined by this grid is for the user to select values for u and w. This determines a point on the surface, a point that will be common to the four new patches. The de Casteljau algorithm is then applied to each of the three columns of control points (the black circles of Figure 6.23a) separately. Each column of four control points $\mathbf{P}_0$, $\mathbf{P}_1$, $\mathbf{P}_2$, and $\mathbf{P}_3$ results in several points, of which the following seven are used for the subdivision (refer to Figure 6.8) $\mathbf{P}_0$, $\mathbf{P}_{01}$, $\mathbf{P}_{012}$, $\mathbf{P}_{0123}$, $\mathbf{P}_{123}$, $\mathbf{P}_{23}$, and $\mathbf{P}_3$. The result of this step is three columns of seven points each (Figure 6.23b where the black circles indicate original control points).

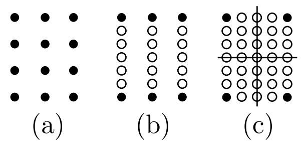


Figure 6.23: Subdividing a Rectangular 3×4 Bézier Patch.

The next step is to apply the de Casteljau algorithm to each of the seven rows of three points, to obtain five points (refer to Figure 6.7). The resulting grid of 7×5 is shown in Figure 6.23c. This grid is divided into four overlapping subgrids of 4×3 control points each, and each subgrid serves to compute a new rectangular Bézier patch.

6.18 Degree Elevation

Degree elevation of the rectangular Bézier surface is similar to elevating the degree of the Bézier curve (Section 6.9). Specifically, Equation (6.19) is extended in the following way. Given a rectangular Bézier patch of degree $m\times n$ (i.e., defined by $(m+1)\times(n+1)$ control points), expressed as a double-polynomial by Equation (6.31)

$$\mathbf{P}_{mn}(u,w)=\sum_{i=0}^{m}\sum_{j=0}^{n}B_{m,i}(u)\mathbf{P}_{i,j}B_{n,j}(w), \tag{6.31}$$

we first write the patch as a double polynomial of degree $(m+1)\times n$ defined by intermediate control points $\mathbf{R}_{ij}$

$$\sum_{j=0}^{n}\left[\sum_{i=0}^{m+1}B_{m+1,i}(u)\mathbf{R}_{i,j}\right]B_{n,j}(w).$$

Based on the result of Section 6.9 the intermediate points are given by

$$\mathbf{R}_{ij}=\frac{i}{m+1}\mathbf{P}_{i-1,j}+(1-\frac{i}{m+1})\mathbf{P}_{i,j}. \tag{6.34}$$

We then repeat this process to increase the degree to $(m+1)\times(n+1)$ and write

$$\mathbf{P}_{m+1,n+1}(u,w)=\sum_{i=0}^{m+1}\sum_{j=0}^{n+1}B_{m+1,i}(u)\mathbf{Q}_{i,j}B_{n+1,j}(w),$$

where the new $(m+2)\times(n+2)$ control points $\mathbf{Q}_{ij}$ can be obtained either from the intermediate points $\mathbf{R}_{ij}$ by an expression similar to Equation (6.34) or directly from the original control points $\mathbf{P}_{ij}$ by a bilinear interpolation

$$\mathbf{Q}_{ij}=\left(\frac{i}{m+1},1-\frac{i}{m+1}\right)\begin{bmatrix}\mathbf{P}_{i-1,j-1} & \mathbf{P}_{i-1,j}\\ \mathbf{P}_{i,j-1} & \mathbf{P}_{i,j}\end{bmatrix}\begin{bmatrix}\frac{j}{n+1}\\ 1-\frac{j}{n+1}\end{bmatrix}, \tag{6.35}$$
$$\text{for}\quad i=0,1,\ldots,m+1,\quad\text{and}\quad j=0,1,\ldots,n+1.$$

If $i=0$ or $j=0$, indexes of the form $i-1$ or $j-1$ are negative, but (the nonexistent) points with such indexes are multiplied by zero, which is why this bilinear interpolation works well in this case. Similarly, when $i=m+1$, point $\mathbf{P}_{i,j}$ does not exist, but the factor $1-i/(m+1)$ that multiplies it is zero and when $j=n+1$, point $\mathbf{P}_{i,j}$ does not exist, but the factor $1 - j/(n+1)$ that multiplies it is also zero. Thus, Equation (6.35) always works.

Example: Starting with the 2×3 control points

$$\begin{matrix}\mathbf{P}_{10} & \mathbf{P}_{11} & \mathbf{P}_{12}\\ \mathbf{P}_{00} & \mathbf{P}_{01} & \mathbf{P}_{02}\end{matrix},$$

(this implies that $m = 1$ and $n = 2$), we perform two steps to elevate the degree of the rectangular patch defined by them from 1×2 to 2×3. The first step is to elevate the degree of each of the three columns from 1 (two control points $\mathbf{P}_{0i}$ and $\mathbf{P}_{1i}$) to 2 (three intermediate points $\mathbf{R}_{0i}$, $\mathbf{R}_{1i}$, and $\mathbf{R}_{2i}$). This step produces the nine intermediate points

$$\begin{matrix}\mathbf{R}_{20} & \mathbf{R}_{21} & \mathbf{R}_{22}\\ \mathbf{R}_{10} & \mathbf{R}_{11} & \mathbf{R}_{12}\\ \mathbf{R}_{00} & \mathbf{R}_{01} & \mathbf{R}_{02}\end{matrix}.$$

For the leftmost column, the two extreme points $\mathbf{R}_{00}$ and $\mathbf{R}_{20}$ equal the two original control points $\mathbf{P}_{00}$ and $\mathbf{P}_{10}$, respectively. The middle point $\mathbf{R}_{10}$ is computed from Equation (6.34) as

$$\mathbf{R}_{10} = \tfrac{1}{2}\mathbf{P}_{00} + (1 - \tfrac{1}{2})\mathbf{P}_{10}.$$

Similarly, the middle column yields

$$\mathbf{R}_{01} = \mathbf{P}_{01}, \quad \mathbf{R}_{21} = \mathbf{P}_{11}, \quad \mathbf{R}_{11} = \tfrac{1}{2}\mathbf{P}_{01} + (1 - \tfrac{1}{2})\mathbf{P}_{11}$$

and the rightmost column results in

$$\mathbf{R}_{02} = \mathbf{P}_{02}, \quad \mathbf{R}_{22} = \mathbf{P}_{12}, \quad \mathbf{R}_{12} = \tfrac{1}{2}\mathbf{P}_{02} + (1 - \tfrac{1}{2})\mathbf{P}_{12}.$$

The second step is to elevate the degree of each of the three rows from 2 (three points $\mathbf{R}_{i0}$, $\mathbf{R}_{i1}$, and $\mathbf{R}_{i2}$) to 3 (four new points $\mathbf{Q}_{i0}$, $\mathbf{Q}_{i1}$, $\mathbf{Q}_{i2}$, and $\mathbf{Q}_{i3}$). This step produces the 12 new control points

$$\begin{matrix}\mathbf{Q}_{20} & \mathbf{Q}_{21} & \mathbf{Q}_{22} & \mathbf{Q}_{23}\\ \mathbf{Q}_{10} & \mathbf{Q}_{11} & \mathbf{Q}_{12} & \mathbf{Q}_{13}\\ \mathbf{Q}_{00} & \mathbf{Q}_{01} & \mathbf{Q}_{02} & \mathbf{Q}_{03}\end{matrix}.$$

For the bottom row, the two extreme points $\mathbf{Q}_{00}$ and $\mathbf{Q}_{03}$ equal the two intermediate control points $\mathbf{R}_{00}$ and $\mathbf{R}_{02}$, respectively. These, together with the two interior points $\mathbf{Q}_{01}$ and $\mathbf{Q}_{02}$ are computed from Equations (6.34) and (6.35) as

$$\begin{aligned}
\mathbf{Q}_{00} &= \mathbf{R}_{00} = \mathbf{P}_{00} = (0, 1-0)\begin{pmatrix}\mathbf{P}_{-1,-1} & \mathbf{P}_{-1,0}\\ \mathbf{P}_{0,-1} & \mathbf{P}_{00}\end{pmatrix}\begin{pmatrix}0\\1\end{pmatrix},\\
\mathbf{Q}_{01} &= \tfrac{1}{3}\mathbf{R}_{00} + \tfrac{2}{3}\mathbf{R}_{01} = \tfrac{1}{3}\mathbf{P}_{00} + \tfrac{2}{3}\mathbf{P}_{01} = (0,1)\begin{pmatrix}\mathbf{P}_{-1,0} & \mathbf{P}_{-1,1}\\ \mathbf{P}_{0,-1} & \mathbf{P}_{01}\end{pmatrix}\begin{pmatrix}1/3\\1-1/3\end{pmatrix},\\
\mathbf{Q}_{02} &= \tfrac{2}{3}\mathbf{R}_{01} + \tfrac{1}{3}\mathbf{R}_{02} = \tfrac{2}{3}\mathbf{P}_{01} + \tfrac{1}{3}\mathbf{P}_{02} = (0,1)\begin{pmatrix}\mathbf{P}_{-1,1} & \mathbf{P}_{-1,2}\\ \mathbf{P}_{01} & \mathbf{P}_{02}\end{pmatrix}\begin{pmatrix}2/3\\1-2/3\end{pmatrix},\\
\mathbf{Q}_{03} &= \mathbf{R}_{02} = \mathbf{P}_{02} = (0, 1-0)\begin{pmatrix}\mathbf{P}_{-1,2} & \mathbf{P}_{-1,3}\\ \mathbf{P}_{0,2} & \mathbf{P}_{03}\end{pmatrix}\begin{pmatrix}1\\0\end{pmatrix}.
\end{aligned}$$

The middle row yields

$$\begin{aligned}
\mathbf{Q}_{10} &= \mathbf{R}_{10} = \tfrac{1}{2}\mathbf{P}_{00} + (1-\tfrac{1}{2})\mathbf{P}_{10} = (\tfrac{1}{2}, 1-\tfrac{1}{2})\begin{pmatrix}\mathbf{P}_{0,-1} & \mathbf{P}_{00}\\ \mathbf{P}_{1,-1} & \mathbf{P}_{10}\end{pmatrix}\begin{pmatrix}0\\1\end{pmatrix},\\
\mathbf{Q}_{11} &= \tfrac{1}{3}\mathbf{R}_{10} + \tfrac{2}{3}\mathbf{R}_{11} = \tfrac{1}{3}(\tfrac{1}{2}\mathbf{P}_{00} + \tfrac{1}{2}\mathbf{P}_{10}) + \tfrac{2}{3}(\tfrac{1}{2}\mathbf{P}_{01} + \tfrac{1}{2}\mathbf{P}_{11})\\
&= (\tfrac{1}{2}, 1-\tfrac{1}{2})\begin{pmatrix}\mathbf{P}_{00} & \mathbf{P}_{01}\\ \mathbf{P}_{10} & \mathbf{P}_{11}\end{pmatrix}\begin{pmatrix}1/3\\1-1/3\end{pmatrix},\\
\mathbf{Q}_{12} &= \tfrac{2}{3}\mathbf{R}_{11} + \tfrac{1}{3}\mathbf{R}_{12} = \tfrac{2}{3}(\tfrac{1}{2}\mathbf{P}_{01} + \tfrac{1}{2}\mathbf{P}_{11}) + \tfrac{1}{3}(\tfrac{1}{2}\mathbf{P}_{02} + \tfrac{1}{2}\mathbf{P}_{12})\\
&= (\tfrac{1}{2}, 1-\tfrac{1}{2})\begin{pmatrix}\mathbf{P}_{01} & \mathbf{P}_{02}\\ \mathbf{P}_{11} & \mathbf{P}_{12}\end{pmatrix}\begin{pmatrix}2/3\\1-2/3\end{pmatrix},\\
\mathbf{Q}_{13} &= \mathbf{R}_{12} = \tfrac{1}{2}\mathbf{P}_{02} + (1-\tfrac{1}{2})\mathbf{P}_{12} = (\tfrac{1}{2}, 1-\tfrac{1}{2})\begin{pmatrix}\mathbf{P}_{02} & \mathbf{P}_{03}\\ \mathbf{P}_{12} & \mathbf{P}_{13}\end{pmatrix}\begin{pmatrix}1\\0\end{pmatrix}.
\end{aligned}$$

Finally, the third row of intermediate points produces the four new control points

$$\begin{aligned}
\mathbf{Q}_{20} &= \mathbf{R}_{20} = \mathbf{P}_{10} = (1,0)\begin{pmatrix}\mathbf{P}_{1,-1} & \mathbf{P}_{10}\\ \mathbf{P}_{2,-1} & \mathbf{P}_{20}\end{pmatrix}\begin{pmatrix}0\\1\end{pmatrix},\\
\mathbf{Q}_{21} &= \tfrac{1}{3}\mathbf{R}_{20} + \tfrac{2}{3}\mathbf{R}_{21} = \tfrac{1}{3}\mathbf{P}_{10} + \tfrac{2}{3}\mathbf{P}_{11} = (1,0)\begin{pmatrix}\mathbf{P}_{10} & \mathbf{P}_{11}\\ \mathbf{P}_{20} & \mathbf{P}_{21}\end{pmatrix}\begin{pmatrix}1/3\\1-1/3\end{pmatrix},\\
\mathbf{Q}_{22} &= \tfrac{2}{3}\mathbf{R}_{21} + \tfrac{1}{3}\mathbf{R}_{22} = \tfrac{2}{3}\mathbf{P}_{11} + \tfrac{1}{3}\mathbf{P}_{12} = (1,0)\begin{pmatrix}\mathbf{P}_{11} & \mathbf{P}_{12}\\ \mathbf{P}_{21} & \mathbf{P}_{22}\end{pmatrix}\begin{pmatrix}2/3\\1/3\end{pmatrix},\\
\mathbf{Q}_{23} &= \mathbf{R}_{22} = \mathbf{P}_{12} = (1,0)\begin{pmatrix}\mathbf{P}_{12} & \mathbf{P}_{13}\\ \mathbf{P}_{22} & \mathbf{P}_{23}\end{pmatrix}\begin{pmatrix}1\\0\end{pmatrix}.
\end{aligned}$$

Figure 6.24 lists code for elevating the degree of a rectangular Bézier patch based on 2×3 control points. In part (a) of the figure each point is a symbol, such as `p00`, and in part (b) each point is a triplet of coordinates. The points are stored in a 2×3 array `p` and are transferred to a 4×5 array `r`, parts of which remain undefined.

6.19 Nonparametric Rectangular Patches

The explicit representation of a surface (Section 1.8) is $z = f(x,y)$. The rectangular Bézier surface is, of course, parametric, but it can be represented in a nonparametric form, similar to explicit surfaces. The derivation in this section is similar to that of Section 6.14. Given $(n+1)\times(m+1)$ real values (not points) P_{ij}, we start with the double polynomial

$$s(u,w) = \sum_{i=0}^{n}\sum_{j=0}^{m} B_{ni}(u)P_{ij}B_{mj}(w)$$

and employ the identity of Equation (6.30) twice, for u and for w, to create the surface patch

$$\mathbf{P}(u,w) = \big(u, w, s(u,w)\big) = \sum_{i=0}^{n}\sum_{j=0}^{m} B_{ni}(u)(i/m, j/n, P_{ij})B_{mj}(w).$$

```
(* Degree elevation of a rect Bezier surface from 2x3 to 4x5 *)
Clear[p,q,r];
m=1; n=2;
p={{p00,p01,p02},{p10,p11,p12}}; (* array of points *)
r=Array[a, {m+3,n+3}]; (* extended array, still undefined *)
Part[r,1]=Table[a, {i,-1,m+2}];
Part[r,2]=Append[Prepend[Part[p,1],a],a];
Part[r,3]=Append[Prepend[Part[p,2],a],a];
Part[r,n+2]=Table[a, {i,-1,m+2}];
MatrixForm[r] (* display extended array *)
q[i_,j_]:=({i/(m+1),1-i/(m+1)}. (* dot product *)
 {{r[[i+1,j+1]],r[[i+1,j+2]]},{r[[i+2,j+1]],r[[i+2,j+2]]}}).
 {j/(n+1),1-j/(n+1)}
q[2,3] (* test *)
```

(a)

```
(* Degree elevation of a rect Bezier surface from 2x3 to 4x5 *)
Clear[p,r,comb];
m=1; n=2; (* set p to an array of 3D points *)
p={{{0,0,0},{1,0,1},{2,0,0}},{{0,1,0},{1,1,.5},{2,1,0}}};
r=Array[a, {m+3,n+3}]; (* extended array, still undefined *)
Part[r,1]=Table[{a,a,a}, {i,-1,m+2}];
Part[r,2]=Append[Prepend[Part[p,1],{a,a,a}],{a,a,a}];
Part[r,3]=Append[Prepend[Part[p,2],{a,a,a}],{a,a,a}];
Part[r,n+2]=Table[{a,a,a}, {i,-1,m+2}];
MatrixForm[r] (* display extended array *)
comb[i_,j_]:=({i/(m+1),1-i/(m+1)}.
{{r[[i+1,j+1]],r[[i+1,j+2]]},{r[[i+2,j+1]],r[[i+2,j+2]]}})[[1]]{j/(n+1),1-j/(n+1)}[[1]]+
({i/(m+1),1-i/(m+1)}.
{{r[[i+1,j+1]],r[[i+1,j+2]]},{r[[i+2,j+1]],r[[i+2,j+2]]}})[[2]]{j/(n+1),1-j/(n+1)}[[2]];
MatrixForm[Table[comb[i,j], {i,0,2},{j,0,3}]]
```

(b)

Figure 6.24: Code for Degree Elevation of a Rectangular Bézier Surface.

This version of the Bézier surface is defined by the control points $(i/m, j/n, P_{ij})$ which form a regular grid on the xy plane.

6.20 Joining Rectangular Bézier Patches

It is easy, although tedious, to explore the conditions for the smooth joining of two Bézier surface patches. Figure 6.25 shows a typical example of this problem. It shows parts of two patches **P** and **Q**. It is not difficult to see that the former is based on 4×5 control points and the latter on $4 \times n$ points, where $n \geq 2$. It is also easy to see that they are joined such that the eight control points along the joint satisfy $\mathbf{P}_{i4} = \mathbf{Q}_{i0}$ for $i = 0, 1, 2, 3$.

The condition for smooth joining of the two surface patches is that the two tangent vectors at the common boundary are in the same direction, although they may have different magnitudes. This condition is expressed as

$$\left.\frac{\partial \mathbf{P}(u,w)}{\partial w}\right|_{w=1} = \alpha \left.\frac{\partial \mathbf{Q}(u,w)}{\partial w}\right|_{w=0}.$$

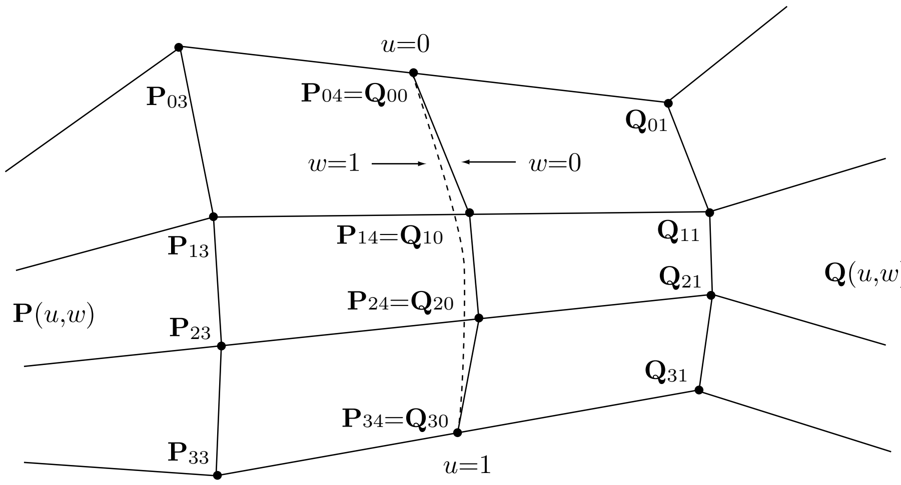


Figure 6.25: Smoothly Joining Rectangular Bézier Patches.

The two tangents are calculated from Equation (6.32) (and the $\mathbf{B}_3$ and $\mathbf{B}_4$ matrices given by Figure 6.3). For the first patch, we have

$$\left.\frac{\partial \mathbf{P}(u,w)}{\partial w}\right|_{w=1} = (u^3,u^2,u,1)\mathbf{B}_3 \begin{pmatrix} \mathbf{P}_{00} & \mathbf{P}_{01} & \mathbf{P}_{02} & \mathbf{P}_{03} & \mathbf{P}_{04} \\ \mathbf{P}_{10} & \mathbf{P}_{11} & \mathbf{P}_{12} & \mathbf{P}_{13} & \mathbf{P}_{14} \\ \mathbf{P}_{20} & \mathbf{P}_{21} & \mathbf{P}_{22} & \mathbf{P}_{23} & \mathbf{P}_{24} \\ \mathbf{P}_{30} & \mathbf{P}_{31} & \mathbf{P}_{32} & \mathbf{P}_{33} & \mathbf{P}_{34} \end{pmatrix} \mathbf{B}_4^T \left.\begin{pmatrix} 4w^3 \\ 3w^2 \\ 2w \\ 1 \\ 0 \end{pmatrix}\right|_{w=1}$$

$$= 4(u^3,u^2,u,1)\mathbf{B}_3 \begin{pmatrix} \mathbf{P}_{04}-\mathbf{P}_{03} \\ \mathbf{P}_{14}-\mathbf{P}_{13} \\ \mathbf{P}_{24}-\mathbf{P}_{23} \\ \mathbf{P}_{34}-\mathbf{P}_{33} \end{pmatrix}.$$

Similarly, for the second patch,

$$\left.\frac{\partial \mathbf{Q}(u,w)}{\partial w}\right|_{w=0} = 4(u^3,u^2,u,1)\mathbf{B}_3 \begin{pmatrix} \mathbf{Q}_{01}-\mathbf{Q}_{00} \\ \mathbf{Q}_{11}-\mathbf{Q}_{10} \\ \mathbf{Q}_{21}-\mathbf{Q}_{20} \\ \mathbf{Q}_{31}-\mathbf{Q}_{30} \end{pmatrix}.$$

The conditions for a smooth join are therefore

$$\begin{pmatrix} \mathbf{P}_{04}-\mathbf{P}_{03} \\ \mathbf{P}_{14}-\mathbf{P}_{13} \\ \mathbf{P}_{24}-\mathbf{P}_{23} \\ \mathbf{P}_{34}-\mathbf{P}_{33} \end{pmatrix} = \alpha \begin{pmatrix} \mathbf{Q}_{01}-\mathbf{Q}_{00} \\ \mathbf{Q}_{11}-\mathbf{Q}_{10} \\ \mathbf{Q}_{21}-\mathbf{Q}_{20} \\ \mathbf{Q}_{31}-\mathbf{Q}_{30} \end{pmatrix},$$

or $\mathbf{P}_{i4}-\mathbf{P}_{i3} = \alpha(\mathbf{Q}_{i1}-\mathbf{Q}_{i0})$ for $i = 0,1,2,$ and 3. This can also be expressed by saying that the three points $\mathbf{P}_{i3}$, $\mathbf{P}_{i4} = \mathbf{Q}_{i0}$, and $\mathbf{Q}_{i1}$ should be on a straight line, although not necessarily equally spaced.

Example: Each of the two patches in Figure 6.26 is based on 3×3 points ($n = 2$). The patches are smoothly connected along the curve defined by the common points $(0, 2, 0)$, $(0, 0, 0)$, and $(0, -2, 0)$. Note that in the diagram they are slightly separated, but this was done intentionally. The smooth connection is obtained by making sure that the points $(-2, 2, 0)$, $(0, 2, 0)$, and $(2, 2, 0)$ are collinear (find the other two collinear triplets). The coordinates of the points are

$$\begin{array}{ccc@{\qquad}ccc}
-2,2,2 & -2,2,0 & 0,2,0 & 0,2,0 & 2,2,0 & 2,2,-2\\
-4,0,2 & -4,0,0 & 0,0,0 & 0,0,0 & 4,0,0 & 4,0,-2\\
-2,-2,2 & -2,-2,0 & 0,-2,0 & 0,-2,0 & 2,-2,0 & 2,-2,-2
\end{array}$$

The famous Utah teapot was designed in the 1960s at the University of Utah by digitizing a real teapot (now at the computer museum in Boston) and creating 32 smoothly-connected Bézier patches defined by a total of 306 control points. [Crow 87] has a detailed description. The coordinates of the points are publicly available, as is a program to display the entire surface. The program is part of a public-domain general three-dimensional graphics package called SIPP (SImple Polygon Processor). SIPP was originally written in Sweden and is distributed by the Free Software Foundation [Free 04]. It can be downloaded anonymously from several sources and for different platforms. A more recent source for this important surface is a *Mathematica* notebook by Jan Mangaldan, available at [MathSource 05].

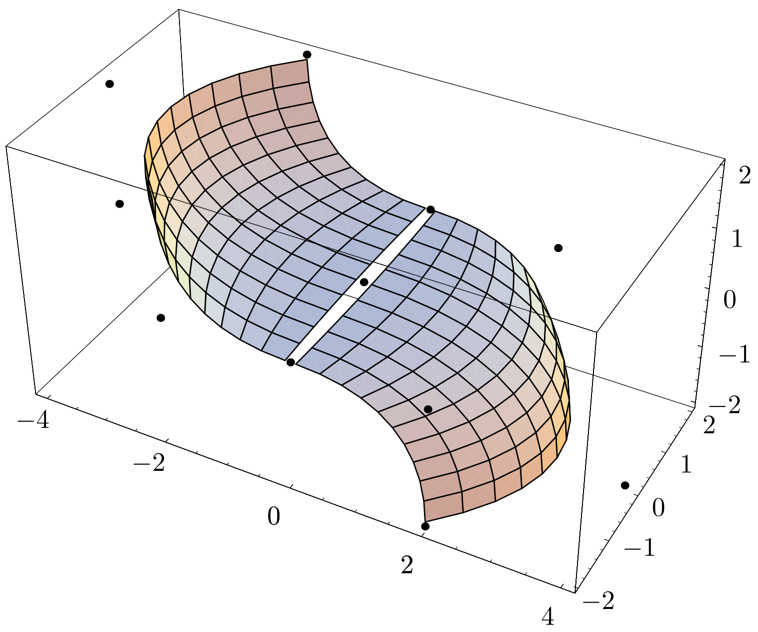
She finished pouring the tea and put down the pot.
"That's an old teapot," remarked Harold.
"Sterling silver," said Maude wistfully. "It was my dear mother-in-law's, part of a dinner set of fifty pieces. It was sent to me, one of the few things that survived." Her voice trailed off and she absently sipped her tea.

—Colin Higgins, *Harold and Maude* (1971).

6.21 An Interpolating Bézier Surface Patch

An interpolating rectangular Bézier surface patch solves the following problem. Given a set of $(m + 1)\times(n + 1)$ data points $\mathbf{Q}_{kl}$, compute a set of $(m + 1)\times(n + 1)$ control points $\mathbf{P}_{ij}$, such that the rectangular Bézier surface patch $\mathbf{P}(u, w)$ defined by the $\mathbf{P}_{ij}$'s will pass through all the data points $\mathbf{Q}_{kl}$.

Section 6.13 discusses the same problem for the Bézier curve, and here we apply the same approach to the rectangular Bézier surface. We select $m + 1$ values u_k and $n + 1$ values w_l and require that the $(m + 1)\times(n + 1)$ surface points $\mathbf{P}(u_k, w_l)$ equal the data points $\mathbf{Q}_{kl}$ for $k = 0, 1, \ldots, m$ and $l = 0, 1, \ldots, n$. This results in a set of $(m+1)\times(n+1)$ equations with the control points $\mathbf{P}_{ij}$ as the unknowns. Such a set of equations may be



```
n=2; Clear[n,bern,p1,p2,g3,bzSurf,patch];
<<:Graphics:ParametricPlot3D.m
p1={{{-2,2,2},{-2,2,0},{0,2,0}},
 {{-4,0,2},{-4,0,0},{0,0,0}},
 {{-2,-2,2},{-2,-2,0},{0,-2,0}}};
p2={{{0,2,0},{2,2,0},{2,2,-2}},
 {{0,0,0},{4,0,0},{4,0,-2}},
 {{0,-2,0},{2,-2,0},{2,-2,-2}}};
pwr[x_,y_]:=If[x==0 && y==0, 1, x^y];
bern[n_,i_,u_]:=Binomial[n,i]pwr[u,i]pwr[1-u,n-i]
bzSurf[p_]:={Sum[p[[i+1,j+1,1]]bern[n,i,u]bern[n,j,w],
 {i,0,n,1}, {j,0,n,1}],
 Sum[p[[i+1,j+1,2]]bern[n,i,u]bern[n,j,w],
 {i,0,n,1}, {j,0,n,1}],
 Sum[p[[i+1,j+1,3]]bern[n,i,u]bern[n,j,w],
 {i,0,n,1}, {j,0,n,1}]};
patch[s_]:=
ParametricPlot3D[bzSurf[s],{u,0,1,.1}, {w,0.02,.98,.1}];
g3=Graphics3D[{AbsolutePointSize[3],
 Table[Point[p1[[i,j]]],{i,1,n+1},{j,1,n+1}]}]
g4=Graphics3D[{AbsolutePointSize[3],
 Table[Point[p2[[i,j]]],{i,1,n+1},{j,1,n+1}]}]
Show[patch[p1],patch[p2],g3,g4,
 DisplayFunction->$DisplayFunction]
```

Figure 6.26: Two Bézier Surface Patches.

big, but is easy to solve with appropriate mathematical software. A general equation in this set is

$$\mathbf{P}(u_k, w_l) = \mathbf{B}_m(u_k)\,\mathbf{P}\,\mathbf{B}_n(w_l) = \mathbf{Q}_{kl} \quad \text{for} \quad k = 0, 1, \ldots, m \quad \text{and} \quad l = 0, 1, \ldots, n.$$

Example: We choose $m = 3$ and $n = 2$. The system of equations becomes

$$\left[(1-u_k)^3, 3u_k(1-u_k)^2, 3u_k^2(1-u_k), u_k^3\right] \begin{bmatrix} \mathbf{P}_{00} & \mathbf{P}_{01} & \mathbf{P}_{02} \\ \mathbf{P}_{10} & \mathbf{P}_{11} & \mathbf{P}_{12} \\ \mathbf{P}_{20} & \mathbf{P}_{21} & \mathbf{P}_{22} \\ \mathbf{P}_{30} & \mathbf{P}_{31} & \mathbf{P}_{32} \end{bmatrix} \begin{bmatrix} (1-w_l)^2 \\ 2w_l(1-w_l) \\ w_l^2 \end{bmatrix} = \mathbf{Q}_{kl},$$

for $k = 0, 1, 2, 3$ and $l = 0, 1, 2$. This is a system of 12 equations in the 12 unknowns $\mathbf{P}_{ij}$. In most cases the u_k values can be equally spaced between 0 and 1 (in our case 0, 0.25, 0.5, 0.75, and 1), and the same for the w_l values (in our case, 0, 0.5, and 1).

6.22 Rational Bézier Surfaces

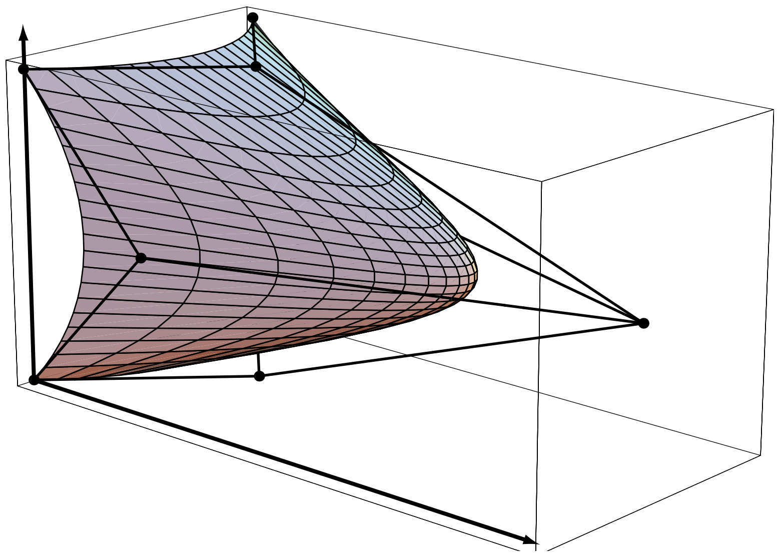
Section 6.15 describe the rational Bézier curve. The principle of this type of curve can be extended to surfaces, and this section discusses the rational rectangular Bézier surface patch. This type of surface is expressed by

$$\mathbf{P}(u, w) = \frac{\sum_{i=0}^{n}\sum_{j=0}^{m} w_{ij} B_{n,i}(u)\mathbf{P}_{ij} B_{m,j}(w)}{\sum_{k=0}^{n}\sum_{l=0}^{m} w_{kl} B_{n,k}(u) B_{m,l}(w)} \qquad 0 \le u, w \le 1. \tag{6.36}$$

When all the weights w_{ij} are set to 1, Equation (6.36) reduces to the original rectangular Bézier surface patch. The weights serve as additional parameters and provide fine, accurate control of the shape of the surface. Figure 6.27 shows how the surface patch of Figure 6.20 can be pulled toward the center point [point $(4, 1, 1)$] by assigning $w_{22} = 5$, while keeping the other weights set to 1.

Note that weights of 0 and negative weights can also be used, as long as the denominator of Equation (6.36) is not zero.

◇ **Exercise 6.23:** Use the code of Figure 6.27 to construct a closed rational Bézier surface patch based on a grid of 2×4 control points.



```
(* A Rational Bezier Surface *)
Clear[pwr,bern,spnts,n,m,wt,bzSurf,cpnts,patch,vlines,hlines,axes];
<<:Graphics:ParametricPlot3D.m
spnts={{{0,0,0},{1,0,1},{0,0,2}},
 {{1,1,0},{4,1,1},{1,1,2}}, {{0,2,0},{1,2,1},{0,2,2}}};
m=Length[spnts[[1]]]-1; n=Length[Transpose[spnts][[1]]]-1;
wt=Table[1, {i,1,n+1},{j,1,m+1}];
wt[[2,2]]=5;
pwr[x_,y_]:=If[x==0 && y==0, 1, x^y];
bern[n_,i_,u_]:=Binomial[n,i]pwr[u,i]pwr[1-u,n-i]
bzSurf[u_,w_]:=
 Sum[wt[[i+1,j+1]]spnts[[i+1,j+1]]bern[n,i,u]bern[m,j,w], {i,0,n}, {j,0,m}]/
 Sum[wt[[i+1,j+1]]bern[n,i,u]bern[m,j,w], {i,0,n}, {j,0,m}];
patch=ParametricPlot3D[bzSurf[u,w],{u,0,1}, {w,0,1},
 Compiled->False, DisplayFunction->Identity];
cpnts=Graphics3D[{AbsolutePointSize[4], (* control points *)
 Table[Point[spnts[[i,j]]], {i,1,n+1},{j,1,m+1}]}];
vlines=Graphics3D[{AbsoluteThickness[1], (* control polygon *)
 Table[Line[{spnts[[i,j]],spnts[[i+1,j]]}], {i,1,n}, {j,1,m+1}]}];
hlines=Graphics3D[{AbsoluteThickness[1],
 Table[Line[{spnts[[i,j]],spnts[[i,j+1]]}], {i,1,n+1}, {j,1,m}]}];
maxx=Max[Flatten[Table[Part[spnts[[i,j]], 1], {i,1,n+1}, {j,1,m+1}]]];
maxy=Max[Flatten[Table[Part[spnts[[i,j]], 2], {i,1,n+1}, {j,1,m+1}]]];
maxz=Max[Flatten[Table[Part[spnts[[i,j]], 3], {i,1,n+1}, {j,1,m+1}]]];
axes=Graphics3D[{AbsoluteThickness[1.5], (* the coordinate axes *)
 Line[{{0,0,maxz},{0,0,0},{maxx,0,0},{0,0,0},{0,maxy,0}}]}];
Show[cpnts,hlines,vlines,axes,patch, PlotRange->All, DefaultFont->{"cmr10",10},
 DisplayFunction->$DisplayFunction, ViewPoint->{2.783, -3.090, 1.243}];
```

Figure 6.27: A Rational Bézier Surface Patch.

6.23 Triangular Bézier Surfaces

The first surface to be derived with Bézier methods was the triangular patch, not the rectangular. It was developed in 1959 by de Casteljau at Citroën. The triangular Bézier patch, and its properties, is the topic of this section, but it should be noted that the ideas and techniques described here can be extended to Bézier surface patches with any number of edges. [DeRose and Loop 89] discusses one approach, termed *S-patch*, to this problem.

The triangular Bézier patch is based on control points $\mathbf{P}_{ijk}$ arranged in a roughly triangular shape. Each control point is three-dimensional and is assigned three indexes ijk such that $0 \le i, j, k \le n$ and $i + j + k = n$. The value of n is selected by the user depending on how large and complex the patch should be and how many points are given. Generally, a large n allows for a finer control of surface details but involves more computations. The following convention is used here. The first index, i, corresponds to the left side of the triangle, the second index, j, corresponds to the base, and the third index, k, corresponds to the right side. The indexing convention for $n = 1$, 2, 3, and 4 are shown in Figure 6.28. There are $n+1$ points on each side of the triangle and because of the way the points are arranged there is a total of $\frac{1}{2}(n+1)(n+2)$ control points:

$$
\begin{array}{c}
\mathbf{P}_{010}\\
\mathbf{P}_{001}\,\mathbf{P}_{100}
\end{array}
\quad
\begin{array}{c}
\mathbf{P}_{020}\\
\mathbf{P}_{011}\,\mathbf{P}_{110}\\
\mathbf{P}_{002}\,\mathbf{P}_{101}\,\mathbf{P}_{200}
\end{array}
\quad
\begin{array}{c}
\mathbf{P}_{030}\\
\mathbf{P}_{021}\,\mathbf{P}_{120}\\
\mathbf{P}_{012}\,\mathbf{P}_{111}\,\mathbf{P}_{210}\\
\mathbf{P}_{003}\,\mathbf{P}_{102}\,\mathbf{P}_{201}\,\mathbf{P}_{300}
\end{array}
\quad
\begin{array}{c}
\mathbf{P}_{040}\\
\mathbf{P}_{031}\,\mathbf{P}_{130}\\
\mathbf{P}_{022}\,\mathbf{P}_{121}\,\mathbf{P}_{220}\\
\mathbf{P}_{013}\,\mathbf{P}_{112}\,\mathbf{P}_{211}\,\mathbf{P}_{310}\\
\mathbf{P}_{004}\,\mathbf{P}_{103}\,\mathbf{P}_{202}\,\mathbf{P}_{301}\,\mathbf{P}_{400}
\end{array}
$$

Figure 6.28: Control Points for Four Triangular Bézier Patches.

The surface patch itself is defined by the trinomial theorem [Equation (6.3)] as

$$
\mathbf{P}(u,v,w) = \sum_{i+j+k=n} \mathbf{P}_{ijk} \frac{n!}{i!\,j!\,k!} u^i v^j w^k = \sum_{i+j+k=n} \mathbf{P}_{ijk} B^n_{ijk}(u,v,w), \qquad (6.37)
$$

where $u + v + w = 1$. Note that even though $\mathbf{P}(u,v,w)$ seems to depend on three parameters, it only depends on two since their sum is constant. The quantities

$$
B^n_{ijk}(u,v,w) = \frac{n!}{i!\,j!\,k!} u^i v^j w^k
$$

are the Bernstein polynomials in two variables (bivariate). They are listed here for $n = 1$, 2, 3, and 4

$$
\begin{array}{c}
v\\
w\;u
\end{array}
\quad
\begin{array}{c}
v^2\\
2vw\;2uv\\
w^2\;2uw\;u^2
\end{array}
\quad
\begin{array}{c}
v^3\\
3v^2w\;3uv^2\\
3vw^2\;6uvw\;3u^2v\\
w^3\;\;3uw^2\;3u^2w\;\;u^3
\end{array}
\quad
\begin{array}{c}
v^4\\
4v^3w\;4uv^3\\
6v^2w^2\;12uv^2w\;6u^2v^2\\
4vw^3\;12uvw^2\;12u^2vw\;4u^3v\\
w^4\;\;4uw^3\;\;6u^2w^2\;\;4u^3w\;\;u^4
\end{array}
$$

The three boundary curves are obtained from Equation (6.37) by setting each of the three parameters in turn to zero. Setting, for example, $u = 0$ causes all terms of Equation (6.37) except those with $i = 0$ to vanish. The result is

$$\mathbf{P}(0,v,w) = \sum_{j+k=n} \mathbf{P}_{0jk} \frac{n!}{j!\,k!} v^j w^k, \quad \text{where} \quad v+w=1. \tag{6.38}$$

Since $v + w = 1$, Equation (6.38) can be written

$$\mathbf{P}(v) = \sum_{j+k=n} \mathbf{P}_{0jk} \frac{n!}{j!\,k!} v^j (1-v)^k = \sum_{j=0}^{n} \mathbf{P}_{0j,n-j} \frac{n!}{j!\,(n-j)!} v^j (1-v)^{n-j}, \tag{6.39}$$

and this is a Bézier curve.

Example: We illustrate the case $n = 2$. There should be three control points on each side of the triangle, for a total of $\frac{1}{2}(2+1)(2+2) = 6$ points. We select simple coordinates:

$$\begin{array}{c} (1,3,1) \\ (0.5,1,0)\,(1.5,1,0) \\ (0,0,0)\,(1,0,-1)\,(2,0,0) \end{array}$$

Note that four points have $z = 0$ and are therefore on the same plane. It is only the other two points, with $z = \pm 1$, that cause this surface to be nonflat.

The expression of the surface is

$$\begin{aligned}
\mathbf{P}(u,v,w) &= \sum_{i+j+k=2} \mathbf{P}_{ijk} \frac{n!}{i!\,j!\,k!} u^i v^j w^k \\
&= \mathbf{P}_{002} \frac{2!}{0!\,0!\,2!} w^2 + \mathbf{P}_{101} \frac{2!}{1!\,0!\,1!} uw + \mathbf{P}_{200} \frac{2!}{2!\,0!\,0!} u^2 \\
&\quad + \mathbf{P}_{011} \frac{2!}{0!\,1!\,1!} vw + \mathbf{P}_{110} \frac{2!}{1!\,1!\,0!} uv + \mathbf{P}_{020} \frac{2!}{0!\,2!\,0!} v^2 \\
&= (0,0,0)w^2 + (1,0,-1)2uw + (2,0,0)u^2 \\
&\quad + (0.5,1,0)2vw + (1.5,1,0)2uv + (1,3,1)v^2 \\
&= (2uw + 2u^2 + vw + 3uv + v^2, 2vw + 2uv + 3v^2, -2uw + v^2).
\end{aligned}$$

It is now easy to verify that the following special values of u, v, and w produce the three corner points:

u	v	w	point
0	0	1	(0,0,0)
0	1	0	(1,3,1)
1	0	0	(2,0,0)

But the most important feature of this triangular surface patch is the way it is displayed as a wireframe. The principle is to display this surface as a mesh of *three* families of curves (compare this with the two families in the case of a rectangular surface patch).

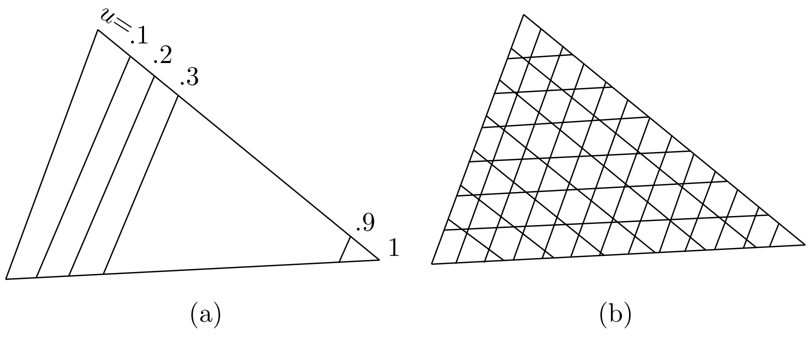


Figure 6.29: (a) Lines in the u Direction. (b) The Complete Surface Patch.

Each family consists of curves that are roughly parallel to one side of the triangle (Figure 6.29a,b).

◇ **Exercise 6.24:** Write pseudo-code to draw the three families of curves.

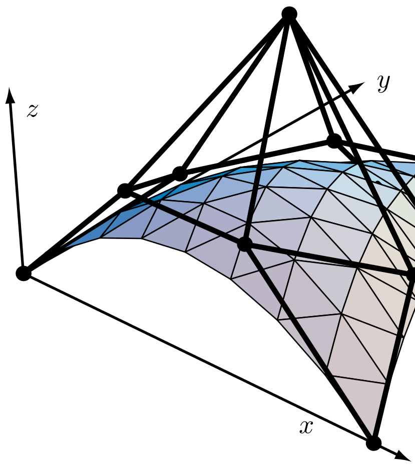
A triangle of points can be stored in a one-dimensional array in computer memory. A simple way of doing this is to store the top point $\mathbf{P}_{0n0}$ at the beginning of the array, followed by a short segment consisting of the two points $\mathbf{P}_{0,n-1,1}$ and $\mathbf{P}_{1,n-1,0}$ of the next row down, followed by a longer segment with three points, and so on, ending with a segment with the $n+1$ points $\mathbf{P}_{00n}$, $\mathbf{P}_{1,0,n-1}$, through $\mathbf{P}_{n00}$ of the bottom row of the triangle. A direct check verifies that the points $\mathbf{P}_{ijk}$ of triangle row j, where $0 \le j \le n$, start at location $j(j+1)/2+1$ of the array, so they can be indexed by $j(j+1)/2+1+i$.

Figure 6.30 lists *Mathematica* code to compute one point on such a surface patch. Note that j is incremented from 0 to n (from the bottom to the top of the triangle), so the first iteration needs the points in the last segment of the array and the last iteration needs the single point at the start of the array. This is why the index to array `pnts` depends on j as $(n-j)(n-j+1)/2+1$ instead of as $j(j+1)/2+1$.

```
(* Triangular Bezier surface patch *)
pnts={{3,3,0}, {2,2,0},{4,2,1}, {1,1,0},{3,1,1},{5,1,2},
 {0,0,0},{2,0,1},{4,0,2},{6,0,3}};
B[i_,j_,k_]:=(n!/(i! j! k!))u^i v^j w^k;
n=3; u=1/6; v=2/6; w=3/6; Tsrpt={0,0,0};
indx:=(n-j)(n-j+1)/2+1+i;
Do[{k=n-i-j, Tsrpt=Tsrpt+B[i,j,k] pnts[[indx]]}, {j,0,n}, {i,0,n-j}];
Tsrpt
```

Figure 6.30: Code for One Point in a Triangular Bézier Patch.

Figure 6.31 shows a triangular Bézier surface patch for $n = 3$. Note how the wireframe consists of three sets of curves and how the curves remain roughly parallel and don't converge toward the three corners. (This should be compared with the triangular Coons patch of Figure 3.14 and with the lofted sweep surface of Figure 9.3. Each of these surfaces is displayed as two families of curves and has one dark corner as a result.) The control points and control polygon are also shown. The *Mathematica* code for this type



```
(* Triangular Bezier patch by Garry Helzer *)
rules=Solve[{u{a1,b1}+v{a2,b2}+w{a3,b3}=={x,y},u+v+w==1},{u,v,w}]
BarycentricCoordinates[Polygon[{{a1_,b1_},{a2_,b2_},{a3_,b3_}}]] \
 [{x_,y_}]={u,v,w}/.rules//Flatten
Subdivide[l_]:=l/. Polygon[{p_,q_,r_}] :> Polygon /@ \
 ({{p+p,p+q,p+r},{p+q,q+q,q+r},{p+r,q+r,r+r},{p+q,q+r,r+p}}/2)
Transform[F_][L_]:= L /. Polygon[l_] :> Polygon[F /@ l]
P[L_][{u_,v_,w_}]:=
Module[{x,y,z,n=(Sqrt[8Length[L]+1]-3)/2},
 ((List @@ Expand[(x+y+z)^n]) /. {x->u,y->v,z->w}).L]
Param[T_,L_][{x_,y_}]:=With[{p=BarycentricCoordinates[T][{x, y}]},P[L][p]]
```

Run the code below in a separate cell

```
(* Triangular bezier patch for n=3 *)
T=Polygon[{{1, 0}, {0, 1}, {0, 0}}];
L={P300,P210,P120,P030, P201,P111,P021, P102,P012, P003} \
 ={{3,0,0},{2.5,1,.5},{2,2,0},{1.5,3,0},
  {2,0,1},{1.5,1,2},{1,2,.5}, {1,0,1},{.5,1,.5}, {0,0,0}};
SubT=Nest[Subdivide, T, 3];
Patch=Transform[Param[T, L]][SubT];
cpts={PointSize[0.02], Point/@L};
coord={AbsoluteThickness[1],
 Line/@{{{0,0,0},{3.2,0,0}},{{0,0,0},{0,3.4,0}},{{0,0,0},{0,0,1.3}}}};
cpolygon={AbsoluteThickness[2],
 Line[{P300,P210,P120,P030,P021,P012,P003,P102,P201,P300}],
 Line[{P012,P102,P111,P120,P021,P111,P201,P210,P111,P012}]};
Show[Graphics3D[{cpolygon,cpts,coord,Patch}], Boxed->False, PlotRange->All,
 ViewPoint->{2.620, -3.176, 2.236}];
```

Figure 6.31: A Triangular Bézier Surface Patch For $n = 3$.

When an object is digitized mechanically, the result is a large set of points. Such a set can be converted to a set of triangles by the Delaunay triangulation algorithm. This method produces a collection of edges that satisfy the following property: For each edge we can find a circle containing the edge's endpoints but not containing any other points.

of surface is due to Garry Helzer and it works by recursively subdividing the triangular patch into subtriangles. Figure C.2 shows two triangular Bézier patches for $n = 2$ and $n = 4$.

6.23.1 Scaffolding Construction

The scaffolding construction (or de Casteljau algorithm) of Section 6.6 can be directly extended to triangular Bézier patches. The bivariate Bernstein polynomials that are the basis of this type of surface are given by Equation (6.3), rewritten here

$$B^n_{i,j,k}(u,v,w) = \sum_{i,j,k\geq 0}^{i+j+k=n} \frac{(i+j+k)!}{i!j!k!} u^i v^j w^k = \sum_{i,j,k\geq 0}^{i+j+k=n} \frac{n!}{i!j!k!} u^i v^j w^k. \qquad (6.3)$$

Direct checking verifies that these polynomials satisfy the recursion relation

$$B^n_{i,j,k}(u,v,w) = uB^{n-1}_{i-1,jk}(u,v,w) + vB^{n-1}_{i,j-1,k}(u,v,w) + wB^{n-1}_{i,j,k-1}(u,v,w), \qquad (6.40)$$

and this relation is the basis of the de Casteljau algorithm for the triangular Bézier patch.

The algorithm starts with the original control points $\mathbf{P}_{ijk}$ which are labeled $\mathbf{P}^0_{ijk}$. The user selects a triplet (u,v,w) where $u+v+w=1$ and performs the following step n times to compute intermediate points $\mathbf{P}^r_{i,j,k}$ for $r = 1,\ldots,n$ and $i+j+k = n-r$

$$\mathbf{P}^r_{i,j,k} = u\mathbf{P}^{r-1}_{i+1,j,k} + v\mathbf{P}^{r-1}_{i,j+1,k} + w\mathbf{P}^{r-1}_{i,j,k+1}.$$

The last step produces the single point $\mathbf{P}^n_{000}$ that's also the point produced by the selected triplet (u,v,w) on the triangular Bézier patch.

The algorithm is illustrated here for $n = 3$. Figure 6.28 shows the 10 control points. Assuming that the user has selected appropriate values for the parameter triplet (u,v,w), the first step of the algorithm produces the six intermediate points for $n = 2$ (Figure 6.32)

$$\begin{aligned}
\mathbf{P}^1_{002} &= u\mathbf{P}^0_{102} + v\mathbf{P}^0_{012} + w\mathbf{P}^0_{003}, & \mathbf{P}^1_{101} &= u\mathbf{P}^0_{201} + v\mathbf{P}^0_{111} + w\mathbf{P}^0_{102},\\
\mathbf{P}^1_{200} &= u\mathbf{P}^0_{300} + v\mathbf{P}^0_{210} + w\mathbf{P}^0_{201}, & \mathbf{P}^1_{011} &= u\mathbf{P}^0_{111} + v\mathbf{P}^0_{021} + w\mathbf{P}^0_{012},\\
\mathbf{P}^1_{110} &= u\mathbf{P}^0_{210} + v\mathbf{P}^0_{120} + w\mathbf{P}^0_{111}, & \mathbf{P}^1_{020} &= u\mathbf{P}^0_{120} + v\mathbf{P}^0_{030} + w\mathbf{P}^0_{021}.
\end{aligned}$$

The second step produces the three intermediate points for $n = 1$

$$\begin{aligned}
\mathbf{P}^2_{001} &= u\mathbf{P}^1_{101} + v\mathbf{P}^1_{011} + w\mathbf{P}^1_{002},\\
\mathbf{P}^2_{100} &= u\mathbf{P}^1_{200} + v\mathbf{P}^1_{110} + w\mathbf{P}^1_{101},\\
\mathbf{P}^2_{010} &= u\mathbf{P}^1_{110} + v\mathbf{P}^1_{020} + w\mathbf{P}^1_{011}.
\end{aligned}$$

And the third step produces the single point

$$\mathbf{P}^3_{000} = u\mathbf{P}^2_{100} + v\mathbf{P}^2_{010} + w\mathbf{P}^2_{001}.$$

This is the point that corresponds to the particular triplet (u,v,w) on the triangular patch defined by the 10 original control points.

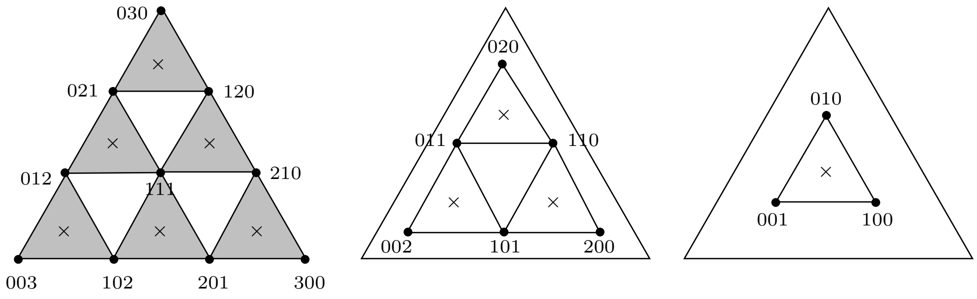


Figure 6.32: Scaffolding in a Triangular Bézier Patch.

◇ **Exercise 6.25:** Illustrate this algorithm for $n = 4$. Start with the 15 original control points and list the four steps of the scaffolding. The final result should be the single point $\mathbf{P}^4_{000}$. Assume that the user has selected appropriate values for the parameter triplet (u, v, w),

◇ **Exercise 6.26:** Assuming the values $u = 1/6$, $v = 2/6$, and $w = 3/6$, and the 10 control points

$$\begin{array}{c}(3,3,0)\\(2,2,0)\,(4,2,1)\\(1,1,0)\,(3,1,1)\,(5,1,2)\\(0,0,0)\,(2,0,1)\,(4,0,2)\,(6,0,3)\end{array}$$

apply the de Casteljau algorithm to compute point $\mathbf{P}^3_{000}$, then use Equation (6.37) to compute surface point $\mathbf{P}(1/6, 2/6/3/6)$ and show that the two points are identical.

It can be shown that a general intermediate point $\mathbf{P}^r_{i,j,k}(u, v, w)$ obtained in the scaffolding process can be computed directly from the control points without having to go through the intermediate steps of the scaffolding construction, as follows

$$\mathbf{P}^r_{ijk}(u,v,w) = \sum_{a+b+c=r} B^r_{abc}(u,v,w)\mathbf{P}_{i+a,j+b,k+c}.$$

Example: For $n = 3$ and $r = 1$, point $\mathbf{P}^1_{002}$ is computed directly from the control points as the sum

$$\mathbf{P}^1_{002} = \sum_{a+b+c=1} B^1_{abc}(u,v,w)\mathbf{P}_{0+a,0+b,2+c} = u\mathbf{P}_{102} + v\mathbf{P}_{012} + w\mathbf{P}_{003}.$$

For $n = 3$ and $r = 2$, point $\mathbf{P}^2_{001}$ is computed directly as the sum

$$\begin{aligned}\mathbf{P}^2_{001} &= \sum_{a+b+c=2} B^2_{abc}(u,v,w)\mathbf{P}_{0+a,0+b,1+c}\\ &= v^2\mathbf{P}_{021} + 2vw\mathbf{P}_{012} + 2uv\mathbf{P}_{111} + w^2\mathbf{P}_{003} + 2uw\mathbf{P}_{102} + u^2\mathbf{P}_{201}.\end{aligned}$$

◇ **Exercise 6.27:** For $n = 4$, compute intermediate points $\mathbf{P}^3_{001}$ and $\mathbf{P}^1_{111}$ directly from the control points.

6.23.2 Subdivision

A triangular Bézier patch can be subdivided into three triangular Bézier patches by a process similar to the one described in Section 6.8 for the Bézier curve. New control points for the three new patches are computed in two steps. First, all the intermediate points generated in the scaffolding steps are computed, then the original interior control points are deleted. We illustrate this process first for $n = 3$ and $n = 4$, then for the general case.

A triangular Bézier patch for $n = 3$ is defined by 10 control points, of which nine are exterior. The user first selects the point inside the surface patch where the three new triangles will meet. This is done by selecting a barycentric triplet (u, v, w). The user then executes three steps of the scaffolding process to generate $6+3+1 = 10$ new intermediate points. The new points are added to the nine exterior control points and the single interior point $\mathbf{P}_{111}$ is deleted. The resulting 19 points are divided into three overlapping sets of 10 points each (Figure 6.33) that define three adjacent triangular Bézier patches inside the original patch.

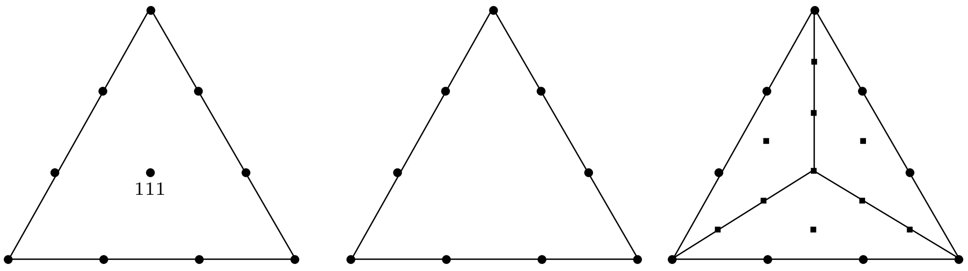


Figure 6.33: Subdividing the Triangular Bézier Patch for $n = 3$.

A triangular Bézier patch for $n = 4$ is defined by 15 control points, of which 12 are exterior. The user selects a barycentric triplet (u, v, w) and executes four steps of the scaffolding process to generate $9 + 6 + 3 + 1 = 19$ new intermediate points. The new points are added to the 12 exterior control points and the three interior points are deleted. The resulting 31 points are divided into three overlapping sets of 15 points each that define three adjacent triangular Bézier patches inside the original patch.

◇ **Exercise 6.28:** Draw a diagram for this case, similar to Figure 6.33.

In general, a triangular Bézier patch is defined by $\frac{1}{2}(n+1)(n+2)$ control points, of which $1+2+\underbrace{2+\cdots+2}_{n-2}+(n+1) = 3n$ points are exterior. The scaffolding construction is then performed, creating $3(n-1)$ points in step 1, $3(n-2)$ points in step 2, and so on, down to $3[n-(n-1)] = 3$ points in step $n-1$ and one point in step n, for a total of $\frac{3n}{2}(n-1)+1$ points. For $n = 3$ through 7, these numbers are 10, 19, 31, 46, and 64. (Note that there are no interior points for $n = 1$ and $n = 2$.) These new points, added to the original exterior points, provide $\frac{3n}{2}(n-1)+1+3n = \frac{3n}{2}(n+1)+1$ points. For $n = 3$ through 7, these numbers are 19, 31, 46, 64, and 84. These numbers are enough to construct three adjacent triangular Bézier patches defined by $\frac{1}{2}(n+1)(n+2)$ control points each.

The user always starts a subdivision by selecting a surface point $\mathbf{P}(u,v,w)$ where the three new triangular patches will meet. A special case occurs if this point is located on an edge of the original triangular patch (i.e., if one of u, v, or w is zero). In such a case, the original triangle is subdivided into two, instead of three triangular patches. This may be useful in cases where only a few extra points are required to reshape the surface.

6.23.3 Degree Elevation

Section 6.9 describes how to elevate the degree of a Bézier curve. This section employs the same ideas to elevate the degree of a triangular Bézier patch. Given a triangular patch of order n defined by $\frac{1}{2}(n+1)(n+2)$ control points $\mathbf{P}_{ijk}$, it is easy to compute a new set of control points $\mathbf{Q}_{ijk}$ that represent the same surface as a triangular patch of order $n+1$. The basic relation is

$$\sum_{i,j,k\geq 0}^{i+j+k=n} \mathbf{P}_{ijk} B^n_{i,j,k}(u,v,w) = \sum_{i+j+k=n+1} \mathbf{Q}_{ijk} B^{n+1}_{i,j,k}(u,v,w).$$

It can be shown, employing methods similar to those of Section 6.9, that the new points $\mathbf{Q}_{ijk}$ are obtained from the original control points $\mathbf{P}_{ijk}$ by

$$\mathbf{Q}_{ijk} = \frac{1}{n+1}\left[i\mathbf{P}_{i-1,j,k} + j\mathbf{P}_{i,j-1,k} + k\mathbf{P}_{i,j,k-1}\right].$$

Example: We elevate the degree of a triangular Bézier patch from $n=2$ to $n=3$. The 10 new control points are obtained from the six original ones by

$$\begin{gathered}
\mathbf{Q}_{003} = \mathbf{P}_{002}, \quad \mathbf{Q}_{102} = \tfrac{1}{3}(\mathbf{P}_{002} + 2\mathbf{P}_{101}), \quad \mathbf{Q}_{201} = \tfrac{1}{3}(2\mathbf{P}_{101} + \mathbf{P}_{200}), \quad \mathbf{Q}_{300} = \mathbf{P}_{200},\\
\mathbf{Q}_{012} = \tfrac{1}{3}(\mathbf{P}_{002} + 2\mathbf{P}_{011}), \quad \mathbf{Q}_{111} = \tfrac{1}{3}(\mathbf{P}_{011} + \mathbf{P}_{101} + \mathbf{P}_{110}), \quad \mathbf{Q}_{210} = \tfrac{1}{3}(2\mathbf{P}_{110} + \mathbf{P}_{200}),\\
\mathbf{Q}_{021} = \tfrac{1}{3}(2\mathbf{P}_{011} + \mathbf{P}_{020}), \quad \mathbf{Q}_{120} = \tfrac{1}{3}(\mathbf{P}_{020} + 2\mathbf{P}_{110}), \quad \mathbf{Q}_{030} = \mathbf{P}_{020}.
\end{gathered}$$

It is possible to elevate the degree of a patch repeatedly. Each degree elevation increases the number of control points and moves them closer to the actual surface. At the limit, the number of control points approaches infinity and the net of points approaches the surface patch.

6.24 Joining Triangular Bézier Patches

The triangular Bézier surface patch is used in cases where a large surface happens to be easier to break up into triangular patches than into rectangular ones. It is therefore important to discover the conditions for smooth joining of these surface patches. The conditions should be expressed in terms of constraints on the control points.

These constraints are developed here for cubic surface patches, but the principles are the same for higher-degree patches. The idea is to calculate three vectors that are tangent to the surface at the common boundary curve. Intuitively, the condition for a smooth join is that these vectors be coplanar (although they can have different magnitudes). We proceed in three steps:

Step 1. Figure 6.34 shows two triangular Bézier cubic patches, $\mathbf{P}(u,v,w)$ and $\mathbf{Q}(u,v,w)$, joined at the common boundary curve $\mathbf{P}(0,v,w)=\mathbf{Q}(0,v,w)$. Equation (6.39) shows how the boundary curves can be expressed as Bézier curves. Based on this equation, our common boundary curve can be written

$$\mathbf{P}(v)=\sum_{j+k=3}\frac{3!}{j!\,k!}v^j(1-v)^{3-j}\mathbf{P}_{0jk}.$$

This is easy to differentiate with respect to v and the result is

$$\begin{aligned}\frac{d\mathbf{P}(v)}{dv}&=3v^2(\mathbf{P}_{030}-\mathbf{P}_{021})+6v(1-v)(\mathbf{P}_{021}-\mathbf{P}_{012})+3(1-v)^2(\mathbf{P}_{012}-\mathbf{P}_{003})\\&=3v^2\mathbf{B}_3+6v(1-v)\mathbf{B}_2+(1-v)^2\mathbf{B}_1,\end{aligned}\tag{6.41}$$

where each of the $\mathbf{B}_i$ vectors is defined as the difference of two control points. They can be seen in the figure as thick arrows going from $\mathbf{P}_{003}$ to $\mathbf{P}_{030}$.

Step 2. Another vector is computed that's tangent to the patch $\mathbf{P}(u,v,w)$ along the common boundary. This is done by calculating the tangent vector to the surface in the u direction and substituting $u=0$. We first write the expression for the surface patch without the parameter w (it can be eliminated because $w=1-u-v$):

$$\mathbf{P}(u,v)=\sum_{i+j+k=3}\frac{3!}{i!\,j!\,k!}u^iv^j(1-u-v)^k\mathbf{P}_{ijk}.$$

This is easy to differentiate with respect to u and it yields

$$\begin{aligned}\left.\frac{\partial\mathbf{P}(u,v)}{\partial u}\right|_{u=0}&=3v^2(\mathbf{P}_{120}-\mathbf{P}_{021})+6v(1-v)(\mathbf{P}_{111}-\mathbf{P}_{012})\\&\quad+3(1-v)^2(\mathbf{P}_{102}-\mathbf{P}_{003})\\&=3v^2\mathbf{A}_3+6v(1-v)\mathbf{A}_2+3(1-v)^2\mathbf{A}_1,\end{aligned}\tag{6.42}$$

where each of the $\mathbf{A}_i$ vectors is again defined as the difference of two control points. They can be seen in the figure as thick arrows going, for example, from $\mathbf{P}_{003}$ to $\mathbf{P}_{102}$.

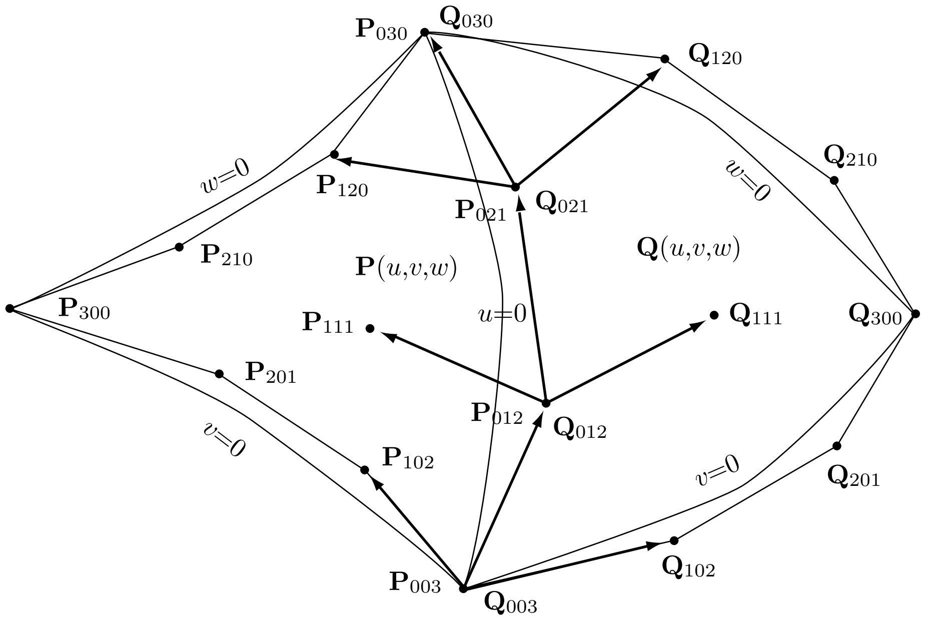


Figure 6.34: Joining Triangular Bézier Patches Smoothly.

Step 3. The third vector is the tangent to the other surface patch $\mathbf{Q}(u,v,w)$ along the common boundary. It is expressed as

$$\begin{aligned}\left.\frac{\partial \mathbf{Q}(u,v)}{\partial u}\right|_{u=0} &= 3v^2(\mathbf{Q}_{120}-\mathbf{Q}_{021}) + 6v(1-v)(\mathbf{Q}_{111}-\mathbf{Q}_{012}) \\ &\quad + 3(1-v)^2(\mathbf{Q}_{102}-\mathbf{Q}_{003}) \\ &= 3v^2\mathbf{C}_3 + 6v(1-v)\mathbf{C}_2 + 3(1-v)^2\mathbf{C}_1,\end{aligned} \tag{6.43}$$

where each of the $\mathbf{C}_i$ vectors is again defined as the difference of two control points. They can be seen in the figure as thick arrows going, for example, from $\mathbf{Q}_{003}$ to $\mathbf{Q}_{102}$.

The condition for smooth joining is that the vectors defined by Equations (6.41) through (6.43) be coplanar for any value of v. This can be expressed as

$$\begin{aligned}&3v^2\mathbf{B}_3 + 6v(1-v)\mathbf{B}_2 + (1-v)^2\mathbf{B}_1 \\ &\qquad = \alpha(3v^2\mathbf{A}_3 + 6v(1-v)\mathbf{A}_2 + 3(1-v)^2\mathbf{A}_1) \\ &\qquad + \beta(3v^2\mathbf{C}_3 + 6v(1-v)\mathbf{C}_2 + 3(1-v)^2\mathbf{C}_1),\end{aligned} \tag{6.44}$$

or, equivalently,

$$v^2(\mathbf{B}_3-\alpha\mathbf{A}_3-\beta\mathbf{C}_3) + 2v(1-v)(\mathbf{B}_2-\alpha\mathbf{A}_2-\beta\mathbf{C}_2) + (1-v)^2(\mathbf{B}_1-\alpha\mathbf{A}_1-\beta\mathbf{C}_1) = 0.$$

Since this should hold for any value of v, it can be written as the set of three equations:

$$\begin{aligned}\mathbf{B}_1&=\alpha\mathbf{A}_1+\beta\mathbf{C}_1,\\ \mathbf{B}_2&=\alpha\mathbf{A}_2+\beta\mathbf{C}_2,\\ \mathbf{B}_3&=\alpha\mathbf{A}_3+\beta\mathbf{C}_3.\end{aligned}\tag{6.45}$$

Each of the three sets of vectors $\mathbf{B}_i$, $\mathbf{A}_i$, and $\mathbf{C}_i$ ($i=1,2,3$) should therefore be coplanar. This condition can be expressed for the control points by saying that each of the three quadrilaterals given by

$$\begin{array}{llll}\mathbf{P}_{003}=\mathbf{Q}_{003}, & \mathbf{P}_{102}, & \mathbf{P}_{012}=\mathbf{Q}_{012}, & \mathbf{Q}_{102},\\ \mathbf{P}_{012}=\mathbf{Q}_{012}, & \mathbf{P}_{111}, & \mathbf{P}_{021}=\mathbf{Q}_{021}, & \mathbf{Q}_{111},\\ \mathbf{P}_{021}=\mathbf{Q}_{021}, & \mathbf{P}_{120}, & \mathbf{P}_{030}=\mathbf{Q}_{030}, & \mathbf{Q}_{120},\end{array}$$

should be planar. In the special case $\alpha=\beta=1$, each quadrilateral should be a square. Otherwise, each should have the same ratio of height to width.

The condition for such a set of three vectors to be coplanar is simple to derive. Figure 6.35 shows a quadrilateral with four corner points $\mathbf{A}$, $\mathbf{B}$, $\mathbf{C}$, and $\mathbf{D}$. Two dashed segments are shown, connecting $\mathbf{A}$ to $\mathbf{B}$ and $\mathbf{C}$ to $\mathbf{D}$. The condition for a flat quadrilateral (four coplanar corners) is that the two segments intersect. The first segment can be expressed parametrically as $(1-u)\mathbf{A}+u\mathbf{B}$ and the second segment can be similarly expressed as $(1-w)\mathbf{C}+w\mathbf{D}$. If there exist u and w in the interval $[0,1]$ such that $(1-u)\mathbf{A}+u\mathbf{B}=(1-w)\mathbf{C}+w\mathbf{D}$, then the quadrilateral is flat.

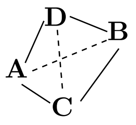


Figure 6.35: A Quadrilateral.

6.24.1 Joining Rectangular and Triangular Bézier Patches

A smooth joining of a rectangular and a triangular surface patches, both of order n, may be useful in many practical applications. Figure 6.36a shows the numbering of the control points for the case $n=4$. Points $\mathbf{Q}_{ijk}$ define the triangular patch and points $\mathbf{P}_{ij}$ define the rectangular patch. There are four pairs (in general, n pairs) of identical points. The problem of joining surface patches of such different topologies can be greatly simplified by elevating the degree (Section 6.23.3) of the two rightmost columns of control points of the triangular patch. The column of four points $\mathbf{Q}_{0jk}$ where $j+k=3$ is transformed to five points $\mathbf{R}_{0jk}$ where $j+k=4$, and the column of three points $\mathbf{Q}_{1jk}$ where $1+j+k=3$ is transformed to four points $\mathbf{R}_{1jk}$ where $1+j+k=4$. Figure 6.36b shows the new points and how, together with the column of four points $\mathbf{P}_{10}$ through $\mathbf{P}_{13}$, they create four quadrilaterals. The condition for smooth joining of the patches is that each quadrilateral be flat.

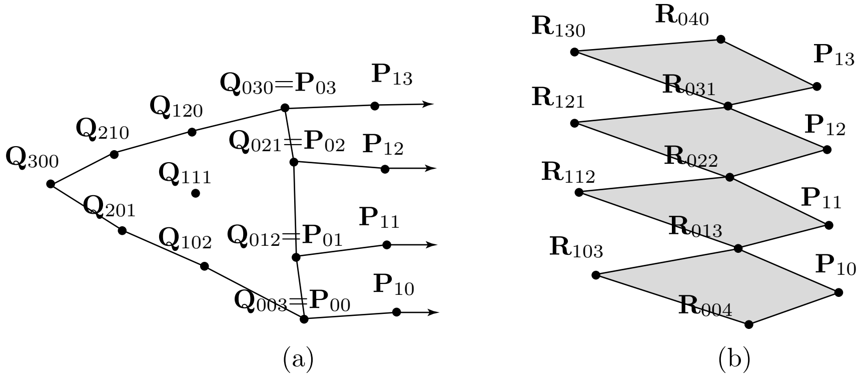


Figure 6.36: Smooth Joining of Triangular and Rectangular Bézier Surface Patches.

In general, there are $n+1$ such quadrilaterals, and each condition can be written explicitly, as an equation, in terms of some of the points $\mathbf{P}_{1i}$, $\mathbf{Q}_{0jk}$, and $\mathbf{Q}_{1jk}$. A general equation is

$$(1-\alpha)\mathbf{R}_{0,i,n-i} + \alpha\mathbf{R}_{0,i+1,n-i} = (1-\beta)\mathbf{R}_{1,i,n-i} + \beta\mathbf{P}_{1,i}, \quad \text{for} \quad i = 0, 1, \ldots, n.$$

When the $\mathbf{R}_{ijk}$ points are expressed in terms of the original $\mathbf{Q}_{ijk}$ points, this relation becomes

$$\begin{aligned}\frac{1-\alpha}{n}\left[i\mathbf{Q}_{0,i-1,n-i} + (n-i)\mathbf{Q}_{0,i,n-i}\right] &+ \frac{\alpha}{n}[(i+1)\mathbf{Q}_{0,i,n-i} + (n-i)\mathbf{Q}_{0,i+1,n-i-1}]\\ &= \frac{\beta}{n}\left[\mathbf{Q}_{0,i,n-i} + i\mathbf{Q}_{1,i-1,n-i} + (n-i)\mathbf{Q}_{1,i,n-i-1}\right] + \beta\mathbf{P}_{1i}.\end{aligned}$$

Note that the quantities α and β in these equations should be indexed by i. In general, each quadrilateral has its own α_i and β_i, but the surface designer can start by guessing values for these $2(n+1)$ quantities, then use them as parameters and vary them (while still keeping each quadrilateral flat), until the surface is molded to the desired shape.

If the rectangular patch is given and the triangular patch has to be designed and manipulated to connect smoothly to it, then the n points $\mathbf{Q}_{1jk}$ (the column to the left of the common boundary) are the unknowns. Conversely, if we start from the triangular patch and want to select control points for the rectangular patch, then the unknowns are the $n+1$ control points $\mathbf{P}_{1i}$ (the column to the right of the common boundary). [Liu and Hoschek 89] has a detailed analysis of the conditions for smooth connection of various types of Bézier surface patches.

6.25 Reparametrizing the Bézier Surface

We illustrate the method described here by applying it to the bicubic Bézier surface patch. The expression for this patch is given by Equations (6.32) and (6.31):

$$\begin{aligned}\mathbf{P}(u,w) &= \sum_{i=0}^{3}\sum_{j=0}^{3} B_{3,i}(u)\mathbf{P}_{i,j}B_{3,j}(w)\\ &= \sum_{i=0}^{3}\sum_{j=0}^{3}(u^3,u^2,u,1)\mathbf{MPM}^{-1}(w^3,w^2,w,1)^T,\end{aligned}$$

where $\mathbf{M}$ is the basis matrix

$$\mathbf{M} = \begin{pmatrix} -1 & 3 & -3 & 1\\ 3 & -6 & 3 & 0\\ -3 & 3 & 0 & 0\\ 1 & 0 & 0 & 0\end{pmatrix}$$

and $\mathbf{P}$ is the 4×4 matrix of control points

$$\begin{pmatrix} \mathbf{P}_{3,0} & \mathbf{P}_{3,1} & \mathbf{P}_{3,2} & \mathbf{P}_{3,3}\\ \mathbf{P}_{2,0} & \mathbf{P}_{2,1} & \mathbf{P}_{2,2} & \mathbf{P}_{2,3}\\ \mathbf{P}_{1,0} & \mathbf{P}_{1,1} & \mathbf{P}_{1,2} & \mathbf{P}_{1,3}\\ \mathbf{P}_{0,0} & \mathbf{P}_{0,1} & \mathbf{P}_{0,2} & \mathbf{P}_{0,3}\end{pmatrix}.$$

This surface patch can be reparametrized with the method of Section 6.10. We select part of patch $\mathbf{P}(u,w)$, e.g., the part where u varies from a to b, and define it as a new patch $\mathbf{Q}(u,w)$ where both u and w vary in the range $[0,1]$. The method discussed here shows how to obtain the control points $\mathbf{Q}_{ij}$ of patch $\mathbf{Q}(u,w)$ as functions of a, b and points $\mathbf{P}_{ij}$.

B-splines are the defacto standard that drives today's sophisticated computer graphics applications. This method is also responsible for the developments that have transformed computer-aided geometric design from the era of hand-built models and manual measurements to fast computations and three-dimensional renderings.

Suppose that we want to reparametrize the "left" part of $\mathbf{P}(u,w)$, i.e., the part where $0\le u\le 0.5$. Applying the methods of Section 6.10, we select $a=0$, $b=0.5$ and can write

$$\mathbf{P}(u/2,w) = (u^3,u^2,u,1)\mathbf{MBPM}^{-1}(w^3,w^2,w,1)^T,$$

where $\mathbf{B}$ is given by Equation (6.22)

$$\mathbf{B} = \begin{pmatrix} (1-a)^3 & 3(a-1)^2a & 3(1-a)a^2 & a^3\\ (a-1)^2(1-b) & (a-1)(-2a-b+3ab) & a(a+2b-3ab) & a^2b\\ (1-a)(-1+b)^2 & (b-1)(-a-2b+3ab) & b(2a+b-3ab) & ab^2\\ (1-b)^3 & 3(b-1)^2b & 3(1-b)b^2 & b^3\end{pmatrix}.$$

Exercise 6.17 shows that selecting $a = 0$ and $b = 0.5$ reduces matrix $\mathbf{B}$ to

$$\mathbf{B} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ \frac{1}{2} & \frac{1}{2} & 0 & 0 \\ \frac{1}{4} & \frac{1}{2} & \frac{1}{4} & 0 \\ \frac{1}{8} & \frac{3}{8} & \frac{3}{8} & \frac{1}{8} \end{pmatrix}.$$

The new control points for our surface patch are therefore

$$\begin{pmatrix} \mathbf{Q}_{3,0} & \mathbf{Q}_{3,1} & \mathbf{Q}_{3,2} & \mathbf{Q}_{3,3} \\ \mathbf{Q}_{2,0} & \mathbf{Q}_{2,1} & \mathbf{Q}_{2,2} & \mathbf{Q}_{2,3} \\ \mathbf{Q}_{1,0} & \mathbf{Q}_{1,1} & \mathbf{Q}_{1,2} & \mathbf{Q}_{1,3} \\ \mathbf{Q}_{0,0} & \mathbf{Q}_{0,1} & \mathbf{Q}_{0,2} & \mathbf{Q}_{0,3} \end{pmatrix} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ \frac{1}{2} & \frac{1}{2} & 0 & 0 \\ \frac{1}{4} & \frac{1}{2} & \frac{1}{4} & 0 \\ \frac{1}{8} & \frac{3}{8} & \frac{3}{8} & \frac{1}{8} \end{pmatrix} \begin{pmatrix} \mathbf{P}_{3,0} & \mathbf{P}_{3,1} & \mathbf{P}_{3,2} & \mathbf{P}_{3,3} \\ \mathbf{P}_{2,0} & \mathbf{P}_{2,1} & \mathbf{P}_{2,2} & \mathbf{P}_{2,3} \\ \mathbf{P}_{1,0} & \mathbf{P}_{1,1} & \mathbf{P}_{1,2} & \mathbf{P}_{1,3} \\ \mathbf{P}_{0,0} & \mathbf{P}_{0,1} & \mathbf{P}_{0,2} & \mathbf{P}_{0,3} \end{pmatrix}$$

$$= \left(\begin{matrix} \mathbf{P}_{3,0} & \mathbf{P}_{3,1} \\ \frac{1}{2}\mathbf{P}_{3,0} + \frac{1}{2}\mathbf{P}_{2,0} & \frac{1}{2}\mathbf{P}_{3,1} + \frac{1}{2}\mathbf{P}_{2,1} \\ \frac{1}{4}\mathbf{P}_{3,0} + \frac{1}{2}\mathbf{P}_{2,0} + \frac{1}{4}\mathbf{P}_{1,0} & \frac{1}{4}\mathbf{P}_{3,1} + \frac{1}{2}\mathbf{P}_{2,1} + \frac{1}{4}\mathbf{P}_{1,1} \\ \frac{1}{8}\mathbf{P}_{3,0} + \frac{3}{8}\mathbf{P}_{2,0} + \frac{3}{8}\mathbf{P}_{1,0} + \frac{1}{8}\mathbf{P}_{0,0} & \frac{1}{8}\mathbf{P}_{3,1} + \frac{3}{8}\mathbf{P}_{2,1} + \frac{3}{8}\mathbf{P}_{1,1} + \frac{1}{8}\mathbf{P}_{1,0} \end{matrix}\right.$$

$$\left.\begin{matrix} \mathbf{P}_{3,2} & \mathbf{P}_{3,3} \\ \frac{1}{2}\mathbf{P}_{3,2} + \frac{1}{2}\mathbf{P}_{2,2} & \frac{1}{2}\mathbf{P}_{3,3} + \frac{1}{2}\mathbf{P}_{2,3} \\ \frac{1}{4}\mathbf{P}_{3,2} + \frac{1}{2}\mathbf{P}_{2,2} + \frac{1}{4}\mathbf{P}_{1,2} & \frac{1}{4}\mathbf{P}_{3,3} + \frac{1}{2}\mathbf{P}_{2,3} + \frac{1}{4}\mathbf{P}_{1,3} \\ \frac{1}{8}\mathbf{P}_{3,2} + \frac{3}{8}\mathbf{P}_{2,2} + \frac{3}{8}\mathbf{P}_{1,2} + \frac{1}{8}\mathbf{P}_{2,0} & \frac{1}{8}\mathbf{P}_{3,3} + \frac{3}{8}\mathbf{P}_{2,3} + \frac{3}{8}\mathbf{P}_{1,3} + \frac{1}{8}\mathbf{P}_{3,0} \end{matrix}\right).$$

In general, suppose we want to reparametrize that portion of patch $\mathbf{P}(u, w)$ where $a \le u \le b$ and $c \le w \le d$. We can write

$$\begin{aligned} &\mathbf{Q}(u, w) \\ &\quad = \mathbf{P}([b-a]u + a, [d-c]w + c) \\ &\quad = \big(([b-a]u+a)^3, ([b-a]u+a)^2, ([b-a]u+a), 1\big)\mathbf{M} \cdot \mathbf{P} \cdot \mathbf{M}^{-1} \begin{pmatrix} ([d-c]w+c)^3 \\ ([d-c]w+c)^2 \\ [d-c]w+c \\ 1 \end{pmatrix} \\ &\quad = (u^3, u^2, u, 1)\mathbf{A}_{ab}\mathbf{M} \cdot \mathbf{P} \cdot \mathbf{M}^T \cdot \mathbf{A}_{cd}^T (w^3, w^2, w, 1)^T \\ &\quad = (u^3, u^2, u, 1)\mathbf{M}(\mathbf{M}^{-1} \cdot \mathbf{A}_{ab} \cdot \mathbf{M})\mathbf{P}(\mathbf{M}^T \cdot \mathbf{A}_{cd}^T \cdot (\mathbf{M}^T)^{-1})\mathbf{M}^T (w^3, w^2, w, 1)^T \\ &\quad = (u^3, u^2, u, 1)\mathbf{M} \cdot \mathbf{B}_{ab} \cdot \mathbf{P} \cdot \mathbf{B}_{cd}^T \cdot \mathbf{M}^T (w^3, w^2, w, 1)^T \\ &\quad = (u^3, u^2, u, 1)\mathbf{M} \cdot \mathbf{Q} \cdot \mathbf{M}^T (w^3, w^2, w, 1)^T, \end{aligned} \tag{6.46}$$

where $\mathbf{B}_{ab} = \mathbf{M}^{-1} \cdot \mathbf{A}_{ab} \cdot \mathbf{M}$, $\mathbf{B}_{cd}^T = \mathbf{M}^T \cdot \mathbf{A}_{cd}^T \cdot (\mathbf{M}^T)^{-1}$, $\mathbf{Q} = \mathbf{B}_{ab} \cdot \mathbf{P} \cdot \mathbf{B}_{cd}^T$, and

$$\mathbf{A}_{ab} = \begin{pmatrix} (b-a)^3 & 0 & 0 & 0 \\ 3a(b-a)^2 & (b-a)^2 & 0 & 0 \\ 3a^2(b-a) & 2a(b-a) & b-a & 0 \\ a^3 & a^2 & a & 1 \end{pmatrix}.$$

The elements of $\mathbf{Q}$ depend on a, b, c, and d, and the $\mathbf{P}_{ij}$'s and are quite complex. They can be produced by the following *Mathematica* code:

```
B={{(1 - a)^3, 3*(-1 + a)^2*a, 3*(1 - a)*a^2, a^3},
  {(-1 + a)^2*(1 - b), (-1 + a)*(-2*a - b + 3*a*b),
   a*(a + 2*b - 3*a*b),
   a^2*b}, {(1 - a)*(-1 + b)^2, (-1 + b)*(-a - 2*b + 3*a*b),
   b*(2*a + b - 3*a*b), a*b^2},
  {(1 - b)^3, 3*(-1 + b)^2*b, 3*(1 - b)*b^2, b^3}};
TB={{(1 - c)^3, (-1 + c)^2*(1 - d), (1 - c)*(-1 + d)^2,
   (1 - d)^3},
  {3*(-1 + c)^2*c, (-1 + c)*(-2*c - d + 3*c*d),
  (-1 + d)*(-c - 2*d + 3*c*d), 3*(-1 + d)^2*d},
  {3*(1 - c)*c^2, c*(c + 2*d - 3*c*d), d*(2*c + d - 3*c*d),
  3*(1 - d)*d^2},
  {c^3, c^2*d, c*d^2, d^3}};
P={{P30,P31,P32,P33},{P20,P21,P22,P23},
 {P10,P11,P12,P13},{P00,P01,P02,P03}};
Q=Simplify[B.P.TB]
```

6.26 The Gregory Patch

John A. Gregory developed this method to extend the Coons surface patch. The Gregory method, however, becomes very practical when it is applied to extend the bicubic Bézier patch. Recall that such a patch is based on $4\times 4 = 16$ control points (Figure 6.37a). We can divide the 16 points into two groups: the interior points, consisting of the four points $\mathbf{P}_{11}$, $\mathbf{P}_{12}$, $\mathbf{P}_{21}$, and $\mathbf{P}_{22}$, and the boundary points, consisting of the remaining 12 points. Experience shows that there are too few interior points to fine-tune the shape of the patch. Moving point $\mathbf{P}_{11}$, for example, affects both the direction from $\mathbf{P}_{01}$ to $\mathbf{P}_{11}$, and the direction from $\mathbf{P}_{10}$ to $\mathbf{P}_{11}$.

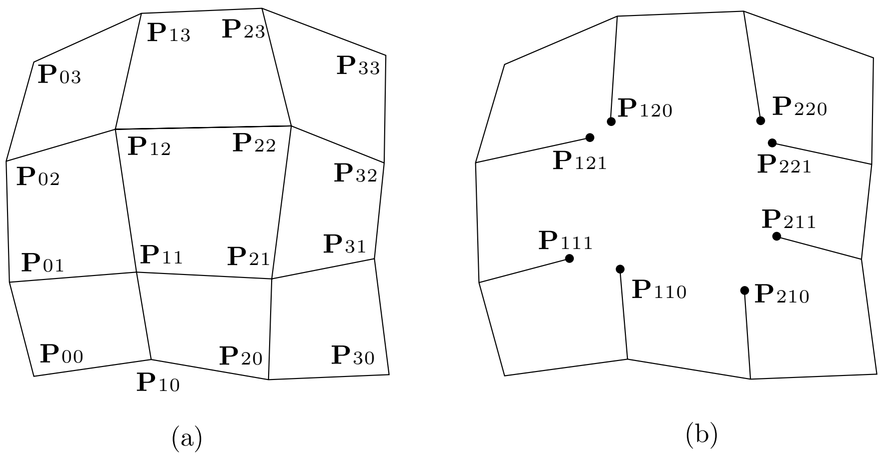


Figure 6.37: (a) A Bicubic Bézier Patch. (b) A Gregory Patch.

The idea in the Gregory patch is to split each of the four interior points into two points. Hence, instead of point $\mathbf{P}_{11}$, for example, there should be two points $\mathbf{P}_{110}$ and $\mathbf{P}_{111}$, both in the vicinity of the original $\mathbf{P}_{11}$. Moving $\mathbf{P}_{110}$ affects the shape of the patch only in the direction from $\mathbf{P}_{10}$ to $\mathbf{P}_{110}$. The shape of the patch around point $\mathbf{P}_{01}$ is not affected (at least, not significantly). Thus, the bicubic Gregory patch is defined by 20 points (Figure 6.37b), eight interior points and 12 boundary points. Points $\mathbf{P}_{110}$ and $\mathbf{P}_{111}$ can initially be set equal to $\mathbf{P}_{11}$, then moved interactively in different directions to obtain the right shape of the surface.

To calculate the surface, we first define 16 new points $\mathbf{Q}_{ij}$, then use Equation (6.31) with the new points as control points and with $n = m = 3$. Twelve of the $\mathbf{Q}$ points are boundary points and are identical to the boundary $\mathbf{P}$ points. The remaining four $\mathbf{Q}$ points are interior and each is calculated from a pair of interior $\mathbf{P}$ points. Their definitions are the following

$$\mathbf{Q}_{11}(u,w) = \frac{u\mathbf{P}_{110} + w\mathbf{P}_{111}}{u+w}, \qquad \mathbf{Q}_{21}(u,w) = \frac{(1-u)\mathbf{P}_{210} + w\mathbf{P}_{211}}{1-u+w},$$
$$\mathbf{Q}_{12}(u,w) = \frac{u\mathbf{P}_{120} + (1-w)\mathbf{P}_{121}}{u+1-w}, \qquad \mathbf{Q}_{22}(u,w) = \frac{(1-u)\mathbf{P}_{220} + (1-w)\mathbf{P}_{221}}{1-u+1-w}.$$

Note that $\mathbf{Q}_{11}(u,w)$ is a barycentric sum of two $\mathbf{P}$ points, so it is well defined. Even though u and w are independent and each is varied from 0 to 1 independently of the other, the sum is always a point on the straight segment connecting $\mathbf{P}_{110}$ to $\mathbf{P}_{111}$. The same is true for the other three interior $\mathbf{Q}$ points.

After calculating the new points, the Gregory patch is defined as the bicubic Bézier patch

$$\mathbf{P}(u,w) = \sum_{i=0}^{3}\sum_{j=0}^{3} B_{3,i}(w)\mathbf{Q}_{i,j}B_{3,j}(u).$$

(Note that four of the 16 points $\mathbf{Q}_{i,j}$ depend on the parameters u and w.)

6.26.1 The Gregory Tangent Vectors

The first derivatives of the Gregory patch are more complex than those of the bicubic Bézier patch, because four of the control points depend on the parameters u and w. The derivatives are

$$\frac{\partial \mathbf{P}(u,w)}{\partial u} = \sum_{i=0}^{3}\sum_{j=0}^{3} \frac{d\,B_{3,i}(u)}{du} B_{3,j}(w)\mathbf{Q}_{i,j}(u,w) + \sum_{i=0}^{3}\sum_{j=0}^{3} B_{3,i}(u)B_{3,j}(w)\frac{\partial \mathbf{Q}_{i,j}(u,w)}{\partial u},$$

$$\frac{\partial \mathbf{P}(u,w)}{\partial w} = \sum_{i=0}^{3}\sum_{j=0}^{3} B_{3,i}(u)\frac{d\,B_{3,j}(w)}{dw}\mathbf{Q}_{i,j}(u,w) + \sum_{i=0}^{3}\sum_{j=0}^{3} B_{3,i}(u)B_{3,j}(w)\frac{\partial \mathbf{Q}_{i,j}(u,w)}{\partial w}.$$

Each derivative is the sum of two similar terms, each of which has the same format as a derivative of the bicubic Bézier patch. Therefore, only one procedure is needed to calculate the derivatives numerically. This procedure is called twice for each partial derivative. The second call involves the derivatives of the control points, which are shown here.

The 12 boundary **Q** points don't depend on u or w, so their derivatives are zero. The eight derivatives of the four interior points are

$$\begin{aligned}
\frac{\partial \mathbf{Q}_{11}(u,w)}{\partial u} &= \frac{w(\mathbf{P}_{110}-\mathbf{P}_{111})}{(u+w)^2}, & \frac{\partial \mathbf{Q}_{11}(u,w)}{\partial w} &= \frac{u(\mathbf{P}_{110}-\mathbf{P}_{111})}{(u+w)^2},\\
\frac{\partial \mathbf{Q}_{21}(u,w)}{\partial u} &= \frac{w(\mathbf{P}_{210}-\mathbf{P}_{211})}{(1-u+w)^2}, & \frac{\partial \mathbf{Q}_{21}(u,w)}{\partial w} &= \frac{(1-u)(\mathbf{P}_{210}-\mathbf{P}_{211})}{(1-u+w)^2},\\
\frac{\partial \mathbf{Q}_{12}(u,w)}{\partial u} &= \frac{(1-w)(\mathbf{P}_{120}-\mathbf{P}_{121})}{(u+1-w)^2}, & \frac{\partial \mathbf{Q}_{12}(u,w)}{\partial w} &= \frac{u(\mathbf{P}_{120}-\mathbf{P}_{121})}{(u+1-w)^2},\\
\frac{\partial \mathbf{Q}_{22}(u,w)}{\partial u} &= \frac{(1-w)(\mathbf{P}_{220}-\mathbf{P}_{221})}{(1-u+1-w)^2}, & \frac{\partial \mathbf{Q}_{22}(u,w)}{\partial w} &= \frac{(1-u)(\mathbf{P}_{220}-\mathbf{P}_{221})}{(1-u+1-w)^2}.
\end{aligned}$$

After the first derivatives (the tangent vectors) have been calculated numerically at a point, they are used to numerically calculate the normal vector at the point.

It is interesting to observe that the Bernshteĭn polynomial of degree 1, i.e., the function $z(t) = (1-t)\,z_1 + t\,z_2$, is precisely the mediation operator $t[z_1, z_2]$ that we discussed in the previous chapter.

Donald Knuth, *The MetafontBook* (1986)