5.7 Exercises

1. Let four points be given by

$$\mathbf{p}_0 = \begin{bmatrix} 0 \\ 0 \end{bmatrix}, \quad \mathbf{p}_1 = \begin{bmatrix} 1 \\ 1 \end{bmatrix}, \quad \mathbf{p}_2 = \begin{bmatrix} 2 \\ 4 \end{bmatrix}, \quad \mathbf{p}_3 = \begin{bmatrix} 3 \\ 9 \end{bmatrix}.$$

   Setting $t_i = i/3$, find the Bézier polygon of the interpolating cubic.

2. Using the same data as in Exercise 1, evaluate the interpolating cubic at $t = 0.5$ using Aitken's algorithm. (Be sure to sketch the spans as well as the intermediate points!) Verify that you get the same answer by applying the de Casteljau algorithm to the Bézier curve from the previous problem.

3. Sketch the interpolant to the three points

$$\begin{bmatrix} -1 \\ 0 \end{bmatrix} \quad \begin{bmatrix} 0 \\ 1 \end{bmatrix} \quad \begin{bmatrix} 1 \\ 0 \end{bmatrix}$$

   with parameters $t_i = (0, 1, 2)$. Use Aitken's algorithm to evaluate at $t = 0.5$ and $t = 1.5$ to guess the shape of the curve. Now repeat this exercise, but with parameters $t_i = (0, 0.25, 2)$. Explain the result.

4. Sketch (manually, if you like) the three quadratic Lagrange polynomials for $(t_0, t_1, t_2) = (0, 4, 5)$.

5. What are the Hermite forms of the three Bézier curves from Section 3.6?

6. The data points of Figure 5.3 are in the file `wing.dat` from the web site. Find and plot the least squares fit using degree three and seven curves with chord length parameters.

Bézier Patches

6

Figure 6.1.
One of the most famous objects in computer graphics is the "Utah teapot," made up of 32 Bézier patches.

In this chapter, we will encounter surfaces for the first time. We will cover the basic definitions and go on to extend the concept of Bézier curves to surfaces. A famous object which is composed of Bézier patches is the "Utah teapot," shown in Figure 6.1.[1]

6.1 Parametric Surfaces

A parametric curve is the result of a mapping of the real line into 2- or 3-space. A parametric surface is defined in a similar way: It is the result of a map of the real plane into 3-space. This "real plane" is called the *domain* of the surface. It is simply a plane with a

[1] Our teapot was created by Mary Zhu using 3D Studio Max.

coordinate system such that every point has coordinates (u, v). The corresponding 3D surface point is then a point:

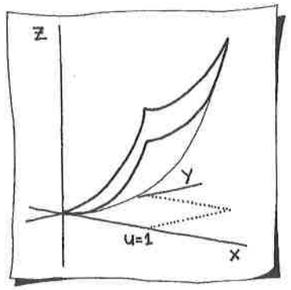
$$\mathbf{x}(u,v) = \begin{bmatrix} f(u,v) \\ g(u,v) \\ h(u,v) \end{bmatrix}. \tag{6.1}$$

EXAMPLE 6.1

The parametric surface given by

$$\mathbf{x}(u,v) = \begin{bmatrix} u \\ v \\ u^2+v^2 \end{bmatrix}$$

is illustrated in Sketch 39. Of course, only a portion of the surface is illustrated; the surface extends infinitely from each edge. This parametric surface happens to be a *functional surface* because two of the coordinate functions in (6.1) are simply u and v.



Sketch 39.
A parametric surface.

Just as for parametric curves, parametric surfaces may be rotated or moved around—they are much more general than bivariate functions of the form $z = f(x, y)$. See the analogous discussion in Section 3.1 for functional curves versus parametric curves.

6.2 Bilinear Patches

We will typically be interested in a finite piece of a parametric surface; It is the image of a rectangle in the domain.[2] The finite piece of surface will then be called a *patch*.

To get started, we pick a special rectangle: It is the *unit square*, defined by

$$\{(u,v) : 0 \le u, v \le 1\}.$$

We map it to a surface patch which is defined by four points $\mathbf{b}_{0,0}, \mathbf{b}_{0,1}, \mathbf{b}_{1,0}, \mathbf{b}_{1,1}$ in a very straightforward way: We simply set

$$\mathbf{x}(u,v) = \begin{bmatrix} 1-u & u \end{bmatrix} \begin{bmatrix} \mathbf{b}_{0,0} & \mathbf{b}_{0,1} \\ \mathbf{b}_{1,0} & \mathbf{b}_{1,1} \end{bmatrix} \begin{bmatrix} 1-v \\ v \end{bmatrix}. \tag{6.2}$$

[2] Shapes more complicated than rectangles are possible, but they are beyond the scope of this book.

This surface patch is linear in both the u and v parameters, hence the name *bilinear patch*.

Equation (6.2) gives a very nice concise expression for the bilinear patch, but it doesn't convey very much geometric information. By simply rewriting the bilinear patch as

$$\mathbf{x}(u,v) = (1-v)\mathbf{p}^u + v\mathbf{q}^u \tag{6.3}$$

where

$$\mathbf{p}^u = (1-u)\mathbf{b}_{0,0} + u\mathbf{b}_{1,0} \quad \text{and} \quad \mathbf{q}^u = (1-u)\mathbf{b}_{0,1} + u\mathbf{b}_{1,1},$$

we can gather a much better feeling for the shape of the bilinear patch.

EXAMPLE 6.2

Let four points $\mathbf{b}_{i,j}$ be given by

$$\mathbf{b}_{0,0} = \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix}, \quad \mathbf{b}_{1,0} = \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix}, \quad \mathbf{b}_{0,1} = \begin{bmatrix} 0 \\ 1 \\ 0 \end{bmatrix}, \quad \mathbf{b}_{1,1} = \begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix}.$$

First compute

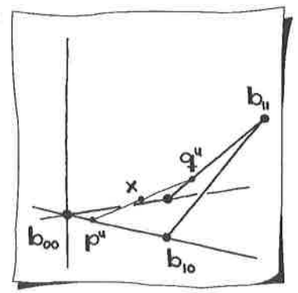
$$\mathbf{p}^u = 0.75 \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix} + 0.25 \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix} = \begin{bmatrix} 0.25 \\ 0 \\ 0 \end{bmatrix}$$

$$\mathbf{q}^u = 0.75 \begin{bmatrix} 0 \\ 1 \\ 0 \end{bmatrix} + 0.25 \begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix} = \begin{bmatrix} 0.25 \\ 1 \\ 0.25 \end{bmatrix}$$

as illustrated in Sketch 40. Then, the point on the patch is given by

$$\mathbf{x}(0.25, 0.5) = 0.5\mathbf{p}^u + 0.5\mathbf{q}^u = \begin{bmatrix} 0.25 \\ 0.5 \\ 0.125 \end{bmatrix}.$$

A rendered image of this patch is shown in Figure 6.2.



Sketch 40.
A bilinear patch.

Notice that the bilinear patch as expressed in (6.3) could have also been written as

$$\mathbf{x}(u,v) = (1-u)\mathbf{p}^v + u\mathbf{q}^v \tag{6.4}$$

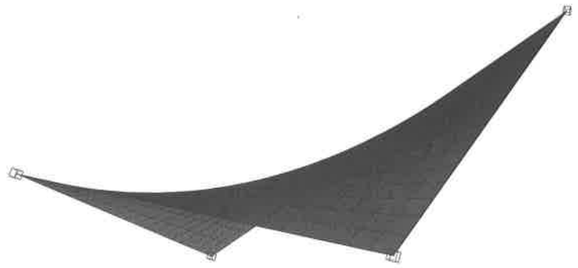


Figure 6.2.
A bilinear patch.

where

$$\mathbf{p}^v = (1-v)\mathbf{b}_{0,0} + v\mathbf{b}_{0,1} \quad \text{and} \quad \mathbf{q}^v = (1-v)\mathbf{b}_{1,0} + v\mathbf{b}_{1,1}.$$

Recompute the point on the patch from Example 6.2 to check that you get the same answer! Don't forget to create your own sketch with both computations.

A bilinear patch, also called a *hyperbolic paraboloid*, is covered by two families of straight lines. That's not so easy to see from (6.2), but it is easy to see from (6.3) or (6.4). Consider (parametric) lines in the domain that are parallel to the sides of the unit square. A line constant in u but varying in v would take the form $(\bar{u}, v)$; a line constant in v but varying in u would take the form $(u, \bar{v})$. These two families of lines on the patch correspond to the lines generated by (6.3) and (6.4), respectively. These are called *isoparametric curves* on the patch since only one parameter is allowed to vary. Four special isoparametric curves (lines) on the patch are the edges corresponding to

$$(u,0), \quad (u,1), \quad (0,v), \quad (1,v).$$

However, a hyperbolic paraboloid also contains *curves*, as we will now see. Consider the line $u = v$ in the domain, i.e., the diagonal of the unit square. In parametric form (in the domain), it may be written as $u(t) = t, v(t) = t$. This domain diagonal is mapped to the 3D curve

$$\mathbf{d}(t) = \mathbf{x}(t,t)$$

on the surface. In more detail:

$$\mathbf{d}(t) = \begin{bmatrix} 1-t & t \end{bmatrix} \begin{bmatrix} \mathbf{b}_{0,0} & \mathbf{b}_{0,1} \\ \mathbf{b}_{1,0} & \mathbf{b}_{1,1} \end{bmatrix} \begin{bmatrix} 1-t \\ t \end{bmatrix}.$$

Collecting terms now gives

$$\mathbf{d}(t) = (1-t)^2\mathbf{b}_{0,0} + 2(1-t)t[\frac{1}{2}\mathbf{b}_{0,1} + \frac{1}{2}\mathbf{b}_{1,0}] + t^2\mathbf{b}_{1,1}. \qquad (6.5)$$

This is a quadratic Bézier curve!

EXAMPLE 6.3

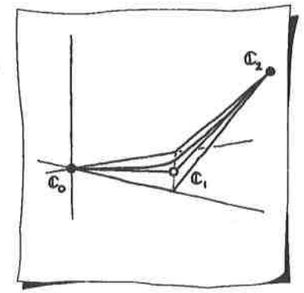
Using the bilinear patch from Example 6.2, let's compute the curve on the surface corresponding to the diagonal in the domain: $u(t) = t, v(t) = t$. From (6.5), we know this is a quadratic curve with Bézier points

$$\mathbf{c}_0 = \mathbf{b}_{0,0} = \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix} \quad \mathbf{c}_1 = \frac{1}{2}[\mathbf{b}_{1,0} + \mathbf{b}_{0,1}] = \begin{bmatrix} 0.5 \\ 0.5 \\ 0 \end{bmatrix} \quad \mathbf{c}_2 = \mathbf{b}_{1,1} = \begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix}.$$

Thus,

$$\mathbf{d}(t) = \mathbf{c}_0 B_0^2(t) + \mathbf{c}_1 B_1^2(t) + \mathbf{c}_2 B_2^2(t)$$

and Sketch 41 illustrates.



Sketch 41.
The diagonal of a bilinear patch.

6.3 Bézier Patches

Let's slightly rewrite the bilinear patch from (6.2) using linear Bernstein polynomials:

$$\mathbf{x}(u,v) = \begin{bmatrix} B_0^1(u) & B_1^1(u) \end{bmatrix} \begin{bmatrix} \mathbf{b}_{0,0} & \mathbf{b}_{0,1} \\ \mathbf{b}_{1,0} & \mathbf{b}_{1,1} \end{bmatrix} \begin{bmatrix} B_0^1(v) \\ B_1^1(v) \end{bmatrix}. \qquad (6.6)$$

This suggests the following generalization:

$$\mathbf{x}(u,v) = \begin{bmatrix} B_0^m(u) & \dots & B_m^m(u) \end{bmatrix} \begin{bmatrix} \mathbf{b}_{0,0} & \dots & \mathbf{b}_{0,n} \\ \vdots & & \vdots \\ \mathbf{b}_{m,0} & \dots & \mathbf{b}_{m,n} \end{bmatrix} \begin{bmatrix} B_0^n(v) \\ \vdots \\ B_n^n(v) \end{bmatrix}. \qquad (6.7)$$

For $m = n = 1$, we recover the bilinear case; for the special setting $n = m = 3$, we have *bicubic* Bézier patches. Expanding (6.7) would result in

$$\mathbf{x}(u,v) = \mathbf{b}_{0,0}B_0^m(u)B_0^n(v) + \ldots + \mathbf{b}_{i,j}B_i^m(u)B_j^n(v) + \ldots + \mathbf{b}_{m,n}B_m^m(u)B_n^n(v).$$

Equation (6.7) is conveniently abbreviated as

$$\mathbf{x}(u,v) = M^{\mathrm{T}}\mathbf{B}N. \tag{6.8}$$

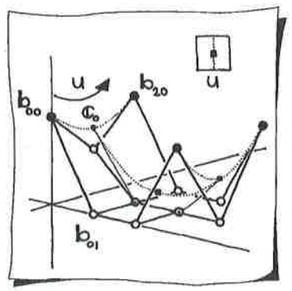
In this form, it is the surface generalization of the curve equation 4.13).

In evaluating (6.7) at a parameter pair (u,v), it is natural to first multiply the first two factors and then multiply the result with the last factor. We would thus define

$$\mathbf{C} = M^{\mathrm{T}}\mathbf{B} = [\mathbf{c}_0, \ldots, \mathbf{c}_n] \tag{6.9}$$

and then write the final result as

$$\mathbf{x}(u,v) = \mathbf{C}N. \tag{6.10}$$



Sketch 42.
Evaluation of a Bézier patch via a u = constant isocurve.

Let's call this the *2-stage explicit* evaluation method. The term 'explicit' refers to the fact that the Bernstein polynomials are explicitly evaluated.

In (6.9), the elements $\mathbf{c}_0, \ldots, \mathbf{c}_n$ of $\mathbf{C}$ do not depend on the parameter value v; thus, after having computed $\mathbf{C}$, we could use it to compute several points $\mathbf{x}(u,v_1), \mathbf{x}(u,v_2), \ldots$. Thus, $\mathbf{C}$ contains the Bézier points which define the curve $\mathbf{C}N$ with constant u and variable v; such a curve is known as an *isoparametric curve*, or isocurve for short. Unlike the bilinear patch in Section 6.2, the isoparametric curves are, in general, not straight lines.

We now give an example, which is shown in Sketch 42.

EXAMPLE 6.4

Let $m = 2$ and $n = 3$; the Bézier patch is given by the control net

$$\mathbf{B} = \begin{bmatrix} \begin{bmatrix} 0 \\ 0 \\ 6 \end{bmatrix} & \begin{bmatrix} 3 \\ 0 \\ 0 \end{bmatrix} & \begin{bmatrix} 6 \\ 0 \\ 0 \end{bmatrix} & \begin{bmatrix} 9 \\ 0 \\ 6 \end{bmatrix} \\ \begin{bmatrix} 0 \\ 3 \\ 3 \end{bmatrix} & \begin{bmatrix} 3 \\ 3 \\ 0 \end{bmatrix} & \begin{bmatrix} 6 \\ 3 \\ 0 \end{bmatrix} & \begin{bmatrix} 9 \\ 3 \\ 0 \end{bmatrix} \\ \begin{bmatrix} 0 \\ 6 \\ 6 \end{bmatrix} & \begin{bmatrix} 3 \\ 6 \\ 0 \end{bmatrix} & \begin{bmatrix} 6 \\ 6 \\ 0 \end{bmatrix} & \begin{bmatrix} 9 \\ 6 \\ 6 \end{bmatrix} \end{bmatrix}$$

We select $(u,v) = (0.5, 0.5)$. First, we compute the quadratic Bernstein basis functions with respect to $u = 0.5$:

$$M^{\mathrm{T}} = \begin{bmatrix} 0.25 & 0.5 & 0.25 \end{bmatrix}.$$

Keep in mind from (4.23) that they sum to one! This results in intermediate control points

$$\mathbf{C} = M^{\mathrm{T}}\mathbf{B} = \left[\begin{bmatrix} 0 \\ 3 \\ 4.5 \end{bmatrix} \begin{bmatrix} 3 \\ 3 \\ 0 \end{bmatrix} \begin{bmatrix} 6 \\ 3 \\ 0 \end{bmatrix} \begin{bmatrix} 9 \\ 3 \\ 3 \end{bmatrix} \right].$$

As you can see, the points in $\mathbf{C}$ are the Bézier points of an isoparametric curve containing $\mathbf{x}(0.5, 0.5)$.

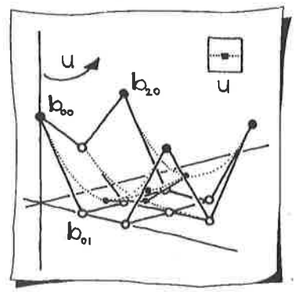
Next, we compute the cubic Bernstein basis functions with respect to $v = 0.5$:

$$N = \begin{bmatrix} 0.125 \\ 0.375 \\ 0.375 \\ 0.125 \end{bmatrix}$$

and we have

$$\mathbf{x}(0.5, 0.5) = \mathbf{C}N = \begin{bmatrix} 4.5 \\ 3 \\ 0.9375 \end{bmatrix}$$

In developing the matrix form of a Bézier patch, we chose to first compute $\mathbf{C} = M^{\mathrm{T}}\mathbf{B}$. Of course, we could have started the other way: Set $\mathbf{D} = \mathbf{B}N$ and then $\mathbf{x} = M^{\mathrm{T}}\mathbf{D}$. The result is the same; Sketch 43 illustrates this alternative approach to Example 6.4 with the v-isoparametric curve labeled $\mathbf{d}_0, \ldots, \mathbf{d}_m$.



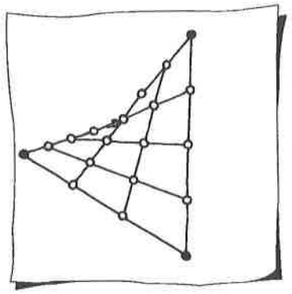
Sketch 43.
Evaluation of a Bézier patch via a v = constant isocurve.

6.4 Properties of Bézier Patches

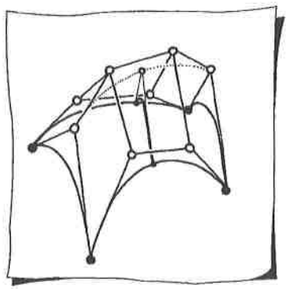
Bézier patches have many properties that are essentially carbon copies of the curve ones.

1. *Endpoint interpolation:* Analogous to the curve case, the patch passes through the four corner control points, that is

$$\mathbf{x}(0,0) = \mathbf{b}_{0,0} \quad \mathbf{x}(1,0) = \mathbf{b}_{m,0}$$
$$\mathbf{x}(0,1) = \mathbf{b}_{0,n} \quad \mathbf{x}(1,1) = \mathbf{b}_{m,n}.$$



Sketch 44.
A degree 3×4 control net with bilinear precision.



Sketch 45.
Bézier patch as locus of a moving and deforming template.

However, this property is more powerful for the surface case than for the curve case. We also have that control polygon boundaries are the control points of the patch boundary curves. For example: The curve $\mathbf{x}(u, 1)$ has the control polygon $\mathbf{b}_{0,n}, \ldots, \mathbf{b}_{m,n}$.

2. *Symmetry*: We could re-index the control net so that any of the corners corresponds to $\mathbf{b}_{0,0}$, and evaluation would result in a patch with the same shape as the original one.

3. *Affine invariance*: Apply an affine map to the control net, and then evaluate the patch. This surface will be identical to the surface created by applying the same affine map to the original patch.

4. *Convex hull property*: For $(u, v) \in [0, 1] \times [0, 1]$, the patch $\mathbf{x}(u, v)$ is in the convex hull of the the control net.

5. *Bilinear precision*: Sketch 44 illustrates a degree $m \times n$ patch with boundary control points which are evenly spaced on lines connecting the corner control points, and the interior control points are evenly-spaced on lines connecting boundary control points on adjacent edges. This patch is identical to the bilinear interpolant to the four corner control points.

6. *Tensor product*: Bézier patches are in the class of tensor product surfaces. This property allows Bézier patches to be dealt with in terms of isoparametric curves, which in turn simplifies evaluation and other operations. The breakdown of (6.8) into (6.9) and (6.10) illustrates this point nicely.

The tensor product property is a very powerful conceptual tool for understanding Bézier patches. Sketch 45 illustrates how the shape of a Bézier patch can be thought of as a record of the shape of a template moving through space. This template can change shape as it moves, and its shape and position is guided by "columns" of Bézier control points.

6.5 Derivatives

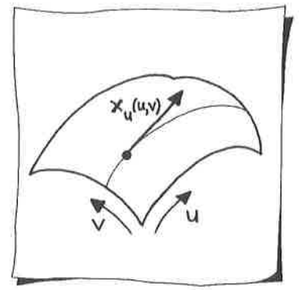
Derivatives for parametric curves are tangent vectors. For surfaces, the same is true: A derivative is the tangent vector of a curve on the surface.

To be more specific, let $\mathbf{x}(u, v)$ be a point on a parametric surface (think Bézier patch for simplicity). There are two isoparametric curves through this point; let's focus on the $v = \textit{constant}$ one,

shown in Sketch 46. This is a parametric curve which happens to be restricted to lie on the surface; its parameter is u. We may thus differentiate it with respect to u. The resulting tangent vector $\mathbf{x}_u$ is called a *partial derivative* and is denoted

$$\mathbf{x}_u(u, v) = \frac{\partial \mathbf{x}(u, v)}{\partial u}.$$

It is also called a u-partial. A simple example will explain.



Sketch 46.
A partial.

EXAMPLE 6.5

Let's take the patch from Example 6.4. What is the partial $\mathbf{x}_v(0.5, 0.5)$? The control polygon C for the $u = 0.5$ isoparametric curve is illustrated in Sketch 42. Its derivative curve is

$$\mathbf{x}_v(0.5, v) = 3(\Delta\mathbf{c}_0 B_0^2(v) + \Delta\mathbf{c}_1 B_1^2(v) + \Delta\mathbf{c}_2 B_2^2(v)),$$

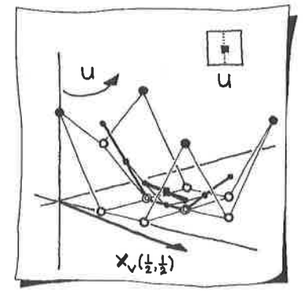
where

$$\Delta\mathbf{c}_0 = \begin{bmatrix} 3 \\ 0 \\ -4.5 \end{bmatrix} \quad \Delta\mathbf{c}_1 = \begin{bmatrix} 3 \\ 0 \\ 0 \end{bmatrix} \quad \Delta\mathbf{c}_2 = \begin{bmatrix} 3 \\ 0 \\ 3 \end{bmatrix}.$$

Evaluate this quadratic Bézier curve at $v = 0.5$, and you have

$$\mathbf{x}_v(0.5, 0.5) = \begin{bmatrix} 9 \\ 0 \\ -1.125 \end{bmatrix}.$$

This partial is shown in Sketch 47.



Sketch 47.
A u-partial.

In an analogous way, we may define u-partials. We just differentiate the isoparametric curve with control points $\mathbf{D} = \mathbf{B}N$.

Another possibility for computing derivatives is to find a closed-form expression. For it, we differentiate (6.7) with respect to u:

$$\mathbf{x}_u(u, v) = m\begin{bmatrix} B_0^{m-1}(u) & \ldots & B_{m-1}^{m-1}(u) \end{bmatrix} \begin{bmatrix} \Delta^{1,0}\mathbf{b}_{0,0} & \ldots & \Delta^{1,0}\mathbf{b}_{0,n} \\ \vdots & & \vdots \\ \Delta^{1,0}\mathbf{b}_{m-1,0} & \ldots & \Delta^{1,0}\mathbf{b}_{m-1,n} \end{bmatrix} \begin{bmatrix} B_0^n(v) \\ \vdots \\ B_n^n(v) \end{bmatrix} \quad (6.11)$$

The terms $\Delta^{1,0}\mathbf{b}_{i,j}$ denote forward differences:

$$\Delta^{1,0}\mathbf{b}_{i,j} = \mathbf{b}_{i+1,j} - \mathbf{b}_{i,j}.$$

Thus the closed-form u-partial derivative expression is a degree $(m-1)\times n$ patch with a control net consisting of vectors rather than points.

EXAMPLE 6.6

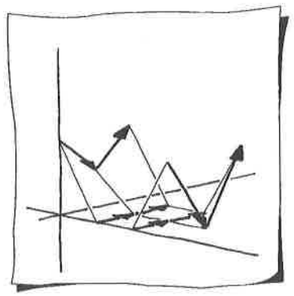
For the patch from Example 6.4, the u-partial is given by forming differences of the control points of the original patch in the u-direction; see Sketch 48. The u-partial at any (u,v) may be determined by evaluating

$$\mathbf{x}_u(u,v) = 2\left[B_0^1(u)\; B_1^1(u)\right]\begin{bmatrix}\begin{bmatrix}0\\3\\-3\end{bmatrix}&\begin{bmatrix}0\\3\\0\end{bmatrix}&\begin{bmatrix}0\\3\\0\end{bmatrix}&\begin{bmatrix}0\\3\\-6\end{bmatrix}\\ \begin{bmatrix}0\\3\\3\end{bmatrix}&\begin{bmatrix}0\\3\\0\end{bmatrix}&\begin{bmatrix}0\\3\\0\end{bmatrix}&\begin{bmatrix}0\\3\\6\end{bmatrix}\end{bmatrix}\begin{bmatrix}B_0^3(v)\\B_1^3(v)\\B_2^3(v)\\B_3^3(v)\end{bmatrix}$$

Evaluation of this patch at $(u,v) = (0.5, 0.5)$ yields

$$\mathbf{x}_u(0.5,0.5) = \begin{bmatrix}0\\6\\0\end{bmatrix}.$$

Notice that all x- and y-coordinates of the control vectors are identical. This is due to the fact that our initial control net is evenly spaced in both the x- and y-directions. See Section 6.12 for more information on this type of surface.



Sketch 48.
Taking differences of a control net.

Let's compare the v-partial derivative computed using the closed-form to the method which isolates an isoparametric curve. For the closed-form v-partial, we differentiate (6.7) with respect to v:

$$\mathbf{x}_v(u,v) = n\begin{bmatrix}B_0^m(u)\;\ldots\;B_m^m(u)\end{bmatrix}\begin{bmatrix}\Delta^{0,1}\mathbf{b}_{0,0}&\ldots&\Delta^{0,1}\mathbf{b}_{0,n-1}\\ \vdots&&\vdots\\ \Delta^{0,1}\mathbf{b}_{m,0}&\ldots&\Delta^{0,1}\mathbf{b}_{m,n-1}\end{bmatrix}\begin{bmatrix}B_0^{n-1}(v)\\ \vdots\\ B_{n-1}^{n-1}(v)\end{bmatrix}. \quad (6.12)$$

The terms $\Delta^{0,1}\mathbf{b}_{i,j}$ denote forward differences:

$$\Delta^{0,1}\mathbf{b}_{i,j} = \mathbf{b}_{i,j+1} - \mathbf{b}_{i,j}.$$

Thus the closed-form v-partial derivative expression is a degree $m \times (n-1)$ patch.

EXAMPLE 6.7

For the patch from Example 6.4, the v-partial is given by forming differences of the control points of the original patch in the v-direction: $\Delta^{0,1}\mathbf{b}_{i,j}$. The v-partial at any (u,v) may be determined by evaluating

$$\mathbf{x}_v(u,v) = 3\left[B_0^2(u)\; B_1^2(u)\; B_2^2(u)\right]\begin{bmatrix}\begin{bmatrix}3\\0\\-6\end{bmatrix}&\begin{bmatrix}3\\0\\0\end{bmatrix}&\begin{bmatrix}3\\0\\6\end{bmatrix}\\ \begin{bmatrix}3\\0\\-3\end{bmatrix}&\begin{bmatrix}3\\0\\0\end{bmatrix}&\begin{bmatrix}3\\0\\0\end{bmatrix}\\ \begin{bmatrix}3\\0\\-6\end{bmatrix}&\begin{bmatrix}3\\0\\0\end{bmatrix}&\begin{bmatrix}3\\0\\6\end{bmatrix}\end{bmatrix}\begin{bmatrix}B_0^2(v)\\B_1^2(v)\\B_2^2(v)\end{bmatrix}.$$

Evaluation of this patch at $(u,v) = (0.5, 0.5)$ yields the same result as in Example 6.5.

6.6 Higher Order Derivatives

A Bézier patch may be differentiated several times. The resulting derivatives of order k are the k^{th} partials of the patch. Focusing on v-partials for now, we obtain

$$\mathbf{x}_v^{(k)}(u,v) = \frac{n!}{(n-k)!}\begin{bmatrix}B_0^m(u)\ldots B_m^m(u)\end{bmatrix}\begin{bmatrix}\Delta^{0,k}\mathbf{b}_{0,0}&\ldots&\Delta^{0,k}\mathbf{b}_{0,n-k}\\ \vdots&&\vdots\\ \Delta^{0,k}\mathbf{b}_{m,0}&\ldots&\Delta^{0,k}\mathbf{b}_{m,n-k}\end{bmatrix}\begin{bmatrix}B_0^{n-k}(v)\\ \vdots\\ B_{n-k}^{n-k}(v)\end{bmatrix} \quad (6.13)$$

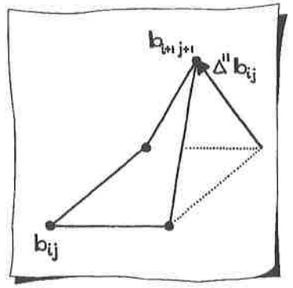
The terms $\Delta^{0,k}\mathbf{b}_{i,j}$ are k^{th} forward differences in the v-direction, acting only on the second subscripts; recall the definition from (4.6). Of course, the k^{th} u-partial follows similarly.

Another commonly used derivative is the mixed partial, or *twist vector*. It is denoted by $\mathbf{x}_{u,v}(u,v)$ and is obtained in either of two ways:

$$\mathbf{x}_{u,v}(u,v) = \frac{\partial \mathbf{x}_u(u,v)}{\partial v} \quad \text{or} \quad \frac{\partial \mathbf{x}_v(u,v)}{\partial u}.$$

If we take the second option, we simply differentiate (6.12) with respect to u, which amounts to taking differences in the u-direction:

$$\mathbf{x}_{u,v}(u,v) = mn\Big[B_0^{m-1}(u)\ldots B_{m-1}^{m-1}(u)\Big]\begin{bmatrix}\Delta^{1,1}\mathbf{b}_{0,0} & \ldots & \Delta^{1,1}\mathbf{b}_{0,n-1}\\ \vdots & & \vdots\\ \Delta^{1,1}\mathbf{b}_{m-1,0} & \ldots & \Delta^{1,1}\mathbf{b}_{m-1,n-1}\end{bmatrix}\begin{bmatrix}B_0^{n-1}(v)\\ \vdots\\ B_{n-1}^{n-1}(v)\end{bmatrix} \quad (6.14)$$



Sketch 49.
A twist coefficient.

The terms $\Delta^{1,1}\mathbf{b}_{i,j}$ are obtained by differencing the control net first in the u-direction

$$\Delta^{1,1}\mathbf{b}_{i,j} = \Delta^{0,1}(\mathbf{b}_{i+1,j} - \mathbf{b}_{i,j}) = \Delta^{0,1}\mathbf{b}_{i+1,j} - \Delta^{0,1}\mathbf{b}_{i,j},$$

and then in the v-direction. They are explicitly given by

$$\Delta^{1,1}\mathbf{b}_{i,j} = \mathbf{b}_{i+1,j+1} - \mathbf{b}_{i+1,j} - \mathbf{b}_{i,j+1} + \mathbf{b}_{i,j}. \quad (6.15)$$

These coefficients have a simple geometric meaning: They indicate how much the quadrilateral defined by the four points in (6.15) deviates from a parallelogram. See Sketch 49.

EXAMPLE 6.8

Revisiting Example 6.2, we compute

$$\mathbf{x}_{u,v}(u,v) = \Delta^{1,1}\mathbf{b}_{0,0} = \begin{bmatrix}0\\0\\1\end{bmatrix},$$

since $B_0^0(u) = 1$ for all u. Thus, a bilinear patch has a *constant* twist vector.

The Bernstein basis functions have the property that for a given degree $B_0^n(0) = 1$ and all other $B_i^n(0) = 0$. A similar situation exists with $B_n^n(1)$; see Figure 4.2. This allows for a simple form of the twist (6.14) at the corners of the patch:

$$\mathbf{x}_{u,v}(0,0) = mn\Delta^{1,1}\mathbf{b}_{0,0} \qquad \mathbf{x}_{u,v}(0,1) = mn\Delta^{1,1}\mathbf{b}_{0,n-1} \quad (6.16)$$

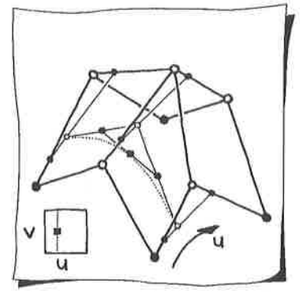
$$\mathbf{x}_{u,v}(1,0) = mn\Delta^{1,1}\mathbf{b}_{m-1,0} \qquad \mathbf{x}_{u,v}(1,1) = mn\Delta^{1,1}\mathbf{b}_{m-1,n-1}. \quad (6.17)$$

6.7 The de Casteljau Algorithm

When we evaluated a patch in Section 6.3, we defined an intermediate set of points, constituting $\mathbf{C} = M^{\mathrm{T}}\mathbf{B}$. Writing out the computation for each individual $\mathbf{c}_i$ yields

$$\begin{aligned}\mathbf{c}_0 &= B_0^m(u)\mathbf{b}_{0,0} + \ldots + B_m^m(u)\mathbf{b}_{m,0},\\ \mathbf{c}_1 &= B_0^m(u)\mathbf{b}_{0,1} + \ldots + B_m^m(u)\mathbf{b}_{m,1},\\ &\ldots\\ \mathbf{c}_n &= B_0^m(u)\mathbf{b}_{0,n} + \ldots + B_m^m(u)\mathbf{b}_{m,n}.\end{aligned}$$

Each of these computations is the evaluation of a degree m Bézier curve. Instead of using the explicit Bernstein polynomial form, we might also use the de Casteljau algorithm for each of the $\mathbf{c}_i$.[3] The resulting geometry is shown in Sketch 50.



Sketch 50.
Evaluating a patch using de Casteljau algorithms.

The final evaluation step, $\mathbf{x}(u,v) = \mathbf{C}N$, is again the evaluation of a Bézier curve. This time it is of degree n. Again, we may use the de Casteljau algorithm. Thus, the de Casteljau algorithm for Bézier patches, let's call this the *2-stage de Casteljau* evaluation method, simply consists of repeated calls to the de Casteljau algorithm for curves.

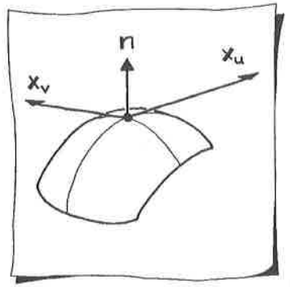
The advantage of this geometric approach is that it allows computation of a derivative along with computation of a point. Once we have the control polygon $\mathbf{C}$, then we evaluate point and tangent just as described for the curve case in Section 3.4. This tangent, for the algorithm above would correspond to $\mathbf{x}_v$. It even has another advantage, as outlined in Section 6.8.

The roles of u and v can be switched in the 2-stage de Casteljau evaluation method. We can also evaluate a Bézier patch by first computing $\mathbf{D} = \mathbf{B}N$ and then $\mathbf{x} = M^{\mathrm{T}}\mathbf{D}$. The tangent to the curve with control polygon $\mathbf{D}$ results in $\mathbf{x}_u$.

6.8 Normals

The *normal vector*, or *normal*, is a fundamental geometric concept which is used throughout computer graphics and CAD/CAM. At a given point $\mathbf{x}(u,v)$ on a patch, the normal is *perpendicular* to the surface at $\mathbf{x}$. At $\mathbf{x}$, the surface has a *tangent plane*; it is defined

[3] This algorithm was described for cubics in Section 3.4 and for degree n Bézier curves in Section 4.3.



Sketch 51.
Tangent plane and normal.

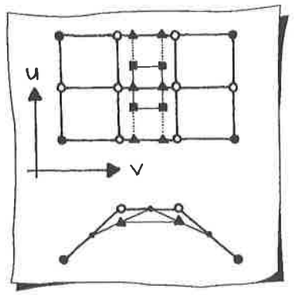
by $\mathbf{x}, \mathbf{x}_u, \mathbf{x}_v$—it is a point and two vectors. The normal $\mathbf{n}$ is a unit vector,[4] defined by

$$\mathbf{n} = \frac{\mathbf{x}_u \wedge \mathbf{x}_v}{\|\mathbf{x}_u \wedge \mathbf{x}_v\|}. \tag{6.18}$$

The geometry is explained in Sketch 51.

A note of caution: Since we are dividing by $\|\mathbf{x}_u \wedge \mathbf{x}_v\|$ in order to normalize, this term should not be zero! In some degenerate cases it can be, so a zero division check is usually a good idea. If this term is within tolerance of zero, you could use local geometry to make an informed guess. Returning the zero vector may or may not be an acceptable solution; depending on the application.

The main ingredients for $\mathbf{n}$ are $\mathbf{x}, \mathbf{x}_u$, and $\mathbf{x}_v$. The de Casteljau algorithm is the evaluation technique normally used for Bézier patches. As presented in Section 6.7, computing $\mathbf{x}_u$ and $\mathbf{x}_v$ would require two patch evaluations. However, there is a trick to compute these three ingredients almost simultaneously. Referring to the schematic diagram of a degree $m = 2 \times n = 3$ patch in Sketch 52, the algorithm proceeds as follows. First, compute $n - 1$ levels of the de Casteljau algorithm for all $m + 1$ rows of control points; these are all with respect to v, and depicted by triangles in the sketch. You now have two columns with $m + 1$ points each; compute $m - 1$ levels of the de Casteljau algorithm for each of them with parameter u. You are now left with four points, depicted by squares in the sketch, which form a bilinear patch. Its tangent plane at (u, v) agrees with the surface's tangent plane at (u, v). Thus, we evaluate and compute the partials of this bilinear patch at (u, v). Keep in mind that the partials of the bilinear patch must be scaled by a factor that reflects the degree of the original patch. Let's call this the *3-stage de Casteljau* evaluation method: a slightly modified version of the 2-stage de Casteljau plus a bilinear patch evaluation.



Sketch 52.
A schematic description of the 3-stage algorithm.

EXAMPLE 6.9

Let's revisit our patch from Example 6.4, and let's set $(u, v) = (0.5, 0.5)$. Two levels of the de Casteljau algorithm for the three rows of control points gives

[4] Its length is one.

$$\begin{bmatrix} 0 \\ 0 \\ 6 \end{bmatrix}$$

$$\begin{bmatrix} 3 \\ 0 \\ 0 \end{bmatrix} \quad \begin{bmatrix} 1.5 \\ 0 \\ 3 \end{bmatrix}$$

$$\begin{bmatrix} 6 \\ 0 \\ 0 \end{bmatrix} \quad \begin{bmatrix} 4.5 \\ 0 \\ 0 \end{bmatrix} \quad \begin{bmatrix} 3 \\ 0 \\ 1.5 \end{bmatrix}$$

$$\begin{bmatrix} 9 \\ 0 \\ 6 \end{bmatrix} \quad \begin{bmatrix} 7.5 \\ 0 \\ 3 \end{bmatrix} \quad \begin{bmatrix} 6 \\ 0 \\ 1.5 \end{bmatrix}$$

$$\begin{bmatrix} 3 \\ 3 \\ 0 \end{bmatrix} \quad \begin{bmatrix} 1.5 \\ 3 \\ 1.5 \end{bmatrix}$$

$$\begin{bmatrix} 6 \\ 3 \\ 0 \end{bmatrix} \quad \begin{bmatrix} 4.5 \\ 3 \\ 0 \end{bmatrix} \quad \begin{bmatrix} 3 \\ 3 \\ 0.75 \end{bmatrix}$$

$$\begin{bmatrix} 9 \\ 3 \\ 0 \end{bmatrix} \quad \begin{bmatrix} 7.5 \\ 3 \\ 0 \end{bmatrix} \quad \begin{bmatrix} 6 \\ 3 \\ 0 \end{bmatrix}$$

$$\begin{bmatrix} 0 \\ 6 \\ 6 \end{bmatrix}$$

$$\begin{bmatrix} 3 \\ 6 \\ 0 \end{bmatrix} \quad \begin{bmatrix} 1.5 \\ 6 \\ 3 \end{bmatrix}$$

$$\begin{bmatrix} 6 \\ 6 \\ 0 \end{bmatrix} \quad \begin{bmatrix} 4.5 \\ 6 \\ 0 \end{bmatrix} \quad \begin{bmatrix} 3 \\ 6 \\ 1.5 \end{bmatrix}$$

$$\begin{bmatrix} 9 \\ 6 \\ 6 \end{bmatrix} \quad \begin{bmatrix} 7.5 \\ 6 \\ 3 \end{bmatrix} \quad \begin{bmatrix} 6 \\ 6 \\ 1.5 \end{bmatrix}$$

We now collect these results and perform one level of two more de Casteljau algorithms:

$$\begin{bmatrix} 3 \\ 0 \\ 1.5 \end{bmatrix}$$

$$\begin{bmatrix} 3 \\ 3 \\ 0.75 \end{bmatrix} \quad \begin{bmatrix} 3 \\ 1.5 \\ 1.125 \end{bmatrix}$$

$$\begin{bmatrix} 3 \\ 6 \\ 1.5 \end{bmatrix} \quad \begin{bmatrix} 3 \\ 4.5 \\ 1.125 \end{bmatrix}$$

$$\begin{bmatrix} 6 \\ 0 \\ 1.5 \end{bmatrix}$$

$$\begin{bmatrix} 6 \\ 3 \\ 0 \end{bmatrix} \quad \begin{bmatrix} 6 \\ 1.5 \\ 0.75 \end{bmatrix}$$

$$\begin{bmatrix} 6 \\ 6 \\ 1.5 \end{bmatrix} \quad \begin{bmatrix} 6 \\ 4.5 \\ 0.75 \end{bmatrix}$$

Thus, the bilinear patch is spanned by the four points

$$\begin{bmatrix} \begin{bmatrix} 3 \\ 1.5 \\ 1.125 \end{bmatrix} & \begin{bmatrix} 6 \\ 1.5 \\ 0.75 \end{bmatrix} \\ \begin{bmatrix} 3 \\ 4.5 \\ 1.125 \end{bmatrix} & \begin{bmatrix} 6 \\ 4.5 \\ 0.75 \end{bmatrix} \end{bmatrix}.$$

This bilinear patch has the same tangent plane as the original patch $\mathbf{x}$. To form $\mathbf{x}_u$, we simply use the 2-stage de Casteljau algorithm to find the u-partial of the bilinear patch, and then scale this result by the degree of $\mathbf{x}$ in the u-direction, $m = 2$:

$$\mathbf{x}_u = 2\left(\begin{bmatrix} 4.5 \\ 4.5 \\ 0.9375 \end{bmatrix} - \begin{bmatrix} 4.5 \\ 1.5 \\ 0.9375 \end{bmatrix} \right) = \begin{bmatrix} 0 \\ 6 \\ 0 \end{bmatrix}$$

(compare with Example 6.6!). The v-partial is found similarly:

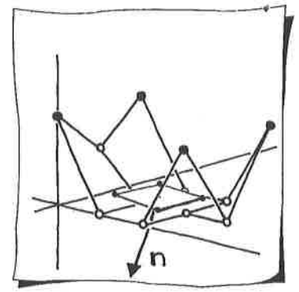
$$\mathbf{x}_v = 3\left(\begin{bmatrix} 6 \\ 3 \\ 0.75 \end{bmatrix} - \begin{bmatrix} 3 \\ 3 \\ 1.125 \end{bmatrix} \right) = \begin{bmatrix} 9 \\ 0 \\ -1.125 \end{bmatrix}$$

(compare with Example 6.5!). Then,

$$\mathbf{x}_u \wedge \mathbf{x}_v = \begin{bmatrix} -6.75 \\ 0 \\ -54 \end{bmatrix}.$$

After normalization, we finally have

$$\mathbf{n} = \begin{bmatrix} -0.1240 \\ 0 \\ -0.9922 \end{bmatrix}.$$



Sketch 53.
Computation of a normal vector.

The computations in Example (6.9) are illustrated in Sketch 53.

Of course, the role of the u and v directions could be switched in the 3-stage de Casteljau algorithm: Compute $m-1$ levels of the de Casteljau algorithm for all $n+1$ columns of control points; these are all with respect to u. You now have two rows with $n+1$ points each; compute $n-1$ levels of the de Casteljau algorithm for each of them with parameter v. You are now left with four points that define the same bilinear patch as above.

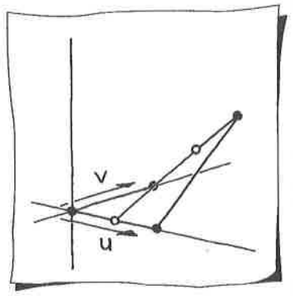
6.9 Changing Degrees

A Bézier patch has two degrees: m in the u-direction and n in the v-direction. Each of these degrees may be increased by a simple procedure. To be concrete, let's raise m to $m+1$. The resulting control net—still describing the same surface—will have $n+1$ columns of control points, each column containing $m+2$ control points. These columns are simply obtained from the original columns by the process of degree elevation for curves as presented in Section 4.5.

EXAMPLE 6.10

Let us write the bilinear patch from Example 6.2 as a patch of degree 2 in u. We obtain the control net

$$\begin{bmatrix} \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix} & \begin{bmatrix} 0 \\ 1 \\ 0 \end{bmatrix} \\ \begin{bmatrix} 0.5 \\ 0.0 \\ 0 \end{bmatrix} & \begin{bmatrix} 0.5 \\ 1 \\ 0.5 \end{bmatrix} \\ \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix} & \begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix} \end{bmatrix}.$$



Sketch 54.
Degree elevation of a bilinear patch.

It is shown in Sketch 54.

To degree elevate in the v-direction, things follow the same pattern: Each of the $m+1$ rows of the control net will be degree elevated to degree $n+1$. Degree elevation in both the u- and v-directions is independent of the order in which it is done.

Degree reduction is also performed on a row-by-row or column-by-column basis, while repeatedly applying the curve algorithm. (See Section 4.6.)

6.10 Subdivision

Another curve operation was subdivision: This is the task of splitting one curve segment into two segments; see (4.10). Similarly, a patch may be subdivided into two patches. The u-parameter u_0 splits the domain unit square into two rectangles as shown in Figure 6.3. The patch is split along this isoparametric curve into two patches, together being identical to the original patch. The recipe for doing this cannot be simpler: Perform curve subdivision for each degree m column of the control net at parameter u_0.

EXAMPLE 6.11

Let's revisit our Example 42 patch and subdivide it at $u_0 = 0.5$. We have to subdivide each column of control points and arrive at the two new control nets.

$$\begin{bmatrix} 0\\0\\6 \end{bmatrix} \quad \begin{bmatrix} 3\\0\\0 \end{bmatrix}$$
$$\begin{bmatrix} 0\\3\\3 \end{bmatrix} \quad \begin{bmatrix} 0\\1.5\\4.5 \end{bmatrix} \quad \begin{bmatrix} 3\\3\\0 \end{bmatrix} \quad \begin{bmatrix} 3\\1.5\\0 \end{bmatrix}$$
$$\begin{bmatrix} 0\\6\\6 \end{bmatrix} \quad \begin{bmatrix} 0\\4.5\\4.5 \end{bmatrix} \quad \begin{bmatrix} 0\\3\\4.5 \end{bmatrix} \quad \begin{bmatrix} 3\\6\\0 \end{bmatrix} \quad \begin{bmatrix} 3\\4.5\\0 \end{bmatrix} \quad \begin{bmatrix} 3\\3\\0 \end{bmatrix}$$

$$\begin{bmatrix} 6\\0\\0 \end{bmatrix} \quad \begin{bmatrix} 9\\0\\6 \end{bmatrix}$$
$$\begin{bmatrix} 6\\3\\0 \end{bmatrix} \quad \begin{bmatrix} 6\\1.5\\0 \end{bmatrix} \quad \begin{bmatrix} 9\\3\\0 \end{bmatrix} \quad \begin{bmatrix} 9\\1.5\\3 \end{bmatrix}$$
$$\begin{bmatrix} 6\\6\\0 \end{bmatrix} \quad \begin{bmatrix} 6\\4.5\\0 \end{bmatrix} \quad \begin{bmatrix} 6\\3\\0 \end{bmatrix} \quad \begin{bmatrix} 9\\6\\6 \end{bmatrix} \quad \begin{bmatrix} 9\\4.5\\3 \end{bmatrix} \quad \begin{bmatrix} 9\\3\\3 \end{bmatrix}$$

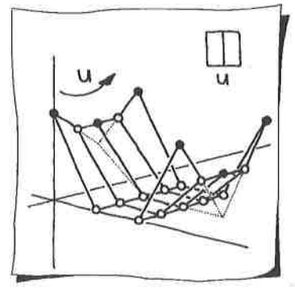
Gather the points along the diagonals of the schematic triangular diagrams of the de Casteljau algorithm to form the patch that lives over the domain $[0, 0.5] \times [0, 1]$ of the original patch:

$$\begin{bmatrix} \begin{bmatrix} 0\\0\\6 \end{bmatrix} & \begin{bmatrix} 3\\0\\0 \end{bmatrix} & \begin{bmatrix} 6\\0\\0 \end{bmatrix} & \begin{bmatrix} 9\\0\\6 \end{bmatrix} \\ \begin{bmatrix} 0\\1.5\\4.5 \end{bmatrix} & \begin{bmatrix} 3\\1.5\\0 \end{bmatrix} & \begin{bmatrix} 6\\1.5\\0 \end{bmatrix} & \begin{bmatrix} 9\\1.5\\3 \end{bmatrix} \\ \begin{bmatrix} 0\\3\\4.5 \end{bmatrix} & \begin{bmatrix} 3\\3\\0 \end{bmatrix} & \begin{bmatrix} 6\\3\\0 \end{bmatrix} & \begin{bmatrix} 9\\3\\3 \end{bmatrix} \end{bmatrix}.$$

Gather the points along the bases of the schematic triangular diagrams to form the patch that lives over $[0.5, 1.0] \times [0, 1]$ of the original patch.

$$\begin{bmatrix} \begin{bmatrix} 0\\3\\4.5 \end{bmatrix} & \begin{bmatrix} 3\\3\\0 \end{bmatrix} & \begin{bmatrix} 6\\3\\0 \end{bmatrix} & \begin{bmatrix} 9\\3\\3 \end{bmatrix} \\ \begin{bmatrix} 0\\4.5\\4.5 \end{bmatrix} & \begin{bmatrix} 3\\4.5\\0 \end{bmatrix} & \begin{bmatrix} 6\\4.5\\0 \end{bmatrix} & \begin{bmatrix} 9\\4.5\\3 \end{bmatrix} \\ \begin{bmatrix} 0\\6\\6 \end{bmatrix} & \begin{bmatrix} 3\\6\\0 \end{bmatrix} & \begin{bmatrix} 6\\6\\0 \end{bmatrix} & \begin{bmatrix} 9\\6\\6 \end{bmatrix} \end{bmatrix}.$$

This result is shown in Sketch 55.



Sketch 55.
Subdivision of a surface.