Chapter 19

Hermite

Our objective in this chapter is to learn a method of interpolating a set of control points, in other words, finding a curve (or surface) that passes through each. Bézier curves, as we know, mandatorily interpolate only their first and last control points, while Bézier surfaces only the four corner control points. B-spline curves and surfaces of quadratic and higher degree do not necessarily interpolate any of their control points. Nevertheless, we learned in Section 18.2.5 how to force a B-spline curve to interpolate a control point by raising the multiplicity of a knot. In fact, the so-called standard knot vector, with repeated end knots, is often used to ensure the interpolation of end control points.

However, if a designer wishes to draw a curve or surface interpolating *all* its control points, then it's best to apply an intrinsically interpolating technique, rather than try to coax an approximating one like Bézier or B-spline into interpolating. A popular class of interpolating curves is that of the Hermite splines and this short chapter introduces this class, together with two special subclasses, that of the natural cubic splines and the cardinal splines. We discuss Hermite surface patches, as well, to interpolate 2D arrays of control points.

We begin with a discussion of general Hermite splines in Section 19.1. These curves, unfortunately, are guaranteed only to be piecewise smooth – they can have corners at control points. Moreover, the user is required to specify tangent vectors at all the control points. The subclass of natural cubic splines, the topic of Section 19.2, automatically determines these tangent vectors by imposing an additional C^2-continuity requirement. Cardinal splines, in Section 19.3, are based upon yet another scheme to automatically specify tangent vectors at control points.

We make a brief presentation of Hermite surfaces in Section 19.4 and conclude in Section 19.5.

19.1 Hermite Splines

A *Hermite spline*, also called a *cubic spline*, interpolating a sequence $P_0, P_1, \ldots, P_n$ of $n+1$ control points, is a *piecewise cubic* curve c passing through the control points. Each cubic arc of c joins successive pairs of control points, so that the entire spline comprises n cubic arcs joined end to end. Figure 19.1 shows a Hermite spline through four control points on a plane. There are corners at the middle two because the tangents of the cubics on either side don't agree.

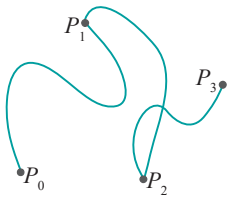

Figure 19.1: A (non-smooth) Hermite spline through four control points, composed of three cubic arcs.

Terminology: A *cubic arc* is a part of a cubic curve; e.g., an arc of the graph of $y = x^3$ is a cubic arc on the plane. Sometimes we'll loosen cubic to mean a polynomial of degree at most three, rather than exactly three.

Remark 19.1. Hermite splines are named after the nineteenth-century French mathematician Charles Hermite.

Remark 19.2. Curves of degree higher than three could be used to interpolate, or even lower, e.g., quadratic. However, three is a "Goldilocks" degree, high enough to assure flexibility, yet low enough to be computationally efficient.

Hermite interpolation for evident reasons is often called *cubic interpolation*.

We'll soon find a way to eliminate the corners in the interior and create a smooth Hermite spline through a given sequence of control points, but let's see first how to make a single cubic arc joining two arbitrary points P and Q.

Write the parametric equation of a general cubic curve c as

$$c(u) = A_3u^3 + A_2u^2 + A_1u + A_0 \qquad (0 \le u \le 1) \tag{19.1}$$

where each A_i, $0 \le i \le 3$, is a point – precisely, its vector of coordinates – in the ambient space. If you are wondering about polynomial coefficients which are vectors rather than scalars, then consider the following example.

Example 19.1. Suppose that we are interested in Hermite splines in the real world so that our ambient space is $\mathbb{R}^3$. Then the equation of a cubic curve is claimed to be

$$c(u) = A_3u^3 + A_2u^2 + A_1u + A_0 \qquad (0 \le u \le 1)$$

where each A_i, $0 \le i \le 3$, is a point in 3-space.

To illustrate, say,

$$A_3 = [-1\ 2\ 0]^T, \quad A_2 = [3\ 0\ -2]^T, \quad A_1 = [4\ 3\ 4]^T, \quad \text{and} \quad A_0 = [0\ 8\ 7]^T$$

Then,

$$\begin{aligned} c(u) &= [-1\ 2\ 0]^T u^3 + [3\ 0\ -2]^T u^2 + [4\ 3\ 4]^T u + [0\ 8\ 7]^T \\ &= [-u^3 + 3u^2 + 4u \qquad 2u^3 + 3u + 8 \qquad -2u^2 + 4u + 7]^T \end{aligned}$$

over the interval $[0, 1]$. As one would expect, the cubic c in $\mathbb{R}^3$ is simply a scalar cubic in *each* of its three coordinates.

Example 19.2. Express in the form (19.1) the twisted cubic given parametrically by

$$x = t,\ y = t^2,\ z = t^3$$

Answer:

$$c(t) = [t\ t^2\ t^3]^T = [0\ 0\ 1]^T t^3 + [0\ 1\ 0]^T t^2 + [1\ 0\ 0]^T t$$

Returning to the general form (19.1) of the cubic, rewrite it as a matrix equation:

$$c(u) = [u^3\ u^2\ u\ 1] \begin{bmatrix} A_3 \\ A_2 \\ A_1 \\ A_0 \end{bmatrix} \qquad (0 \le u \le 1) \tag{19.2}$$

Note: The RHS is a product of a 1×4 matrix of scalars with a 4×1 matrix of vectors, but this is not a problem if we appropriately multiply a vector by a scalar while following the usual rules of matrix multiplication.

Differentiating (19.2) one obtains the derivative of c as

$$c'(u) = [3u^2 \quad 2u \quad 1 \quad 0] \begin{bmatrix} A_3 \\ A_2 \\ A_1 \\ A_0 \end{bmatrix} \qquad (0 \le u \le 1) \tag{19.3}$$

Substitute 0 and 1 for u in Equations (19.2) and (19.3) to find that

$$c(0) = A_0, \quad c(1) = A_3 + A_2 + A_1 + A_0, \quad c'(0) = A_1, \quad c'(1) = 3A_3 + 2A_2 + A_1 \quad (19.4)$$

It seems that if one could specify $c(0)$, $c(1)$, $c'(0)$ and $c'(1)$, then one would have four equations in the four unknowns A_0, A_1, A_2 and A_3, which should solve to find these coefficients and specify c (*alert*: that's four vector equations in four *vector* unknowns, so, e.g., if we are in 3-space, we'll have actually twelve equations in twelve *scalar* unknowns).

Since we're looking for a cubic arc c from P to Q, we know at least that $c(0) = P$ and $c(1) = Q$; as for the tangent vectors $c'(0)$ and $c'(1)$, we have freedom to specify them as we please. Let's choose them to be two vectors denoted P' and Q', respectively. See Figure 19.2.

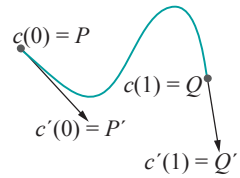

Figure 19.2: Four boundary constraints on a cubic curve c.

Accordingly, write (19.4) as

$$P = A_0, \qquad Q = A_3 + A_2 + A_1 + A_0, \qquad P' = A_1, \qquad Q' = 3A_3 + 2A_2 + A_1 \quad (19.5)$$

which in matrix form is the equation

$$\begin{bmatrix} P \\ Q \\ P' \\ Q' \end{bmatrix} = \begin{bmatrix} 0 & 0 & 0 & 1 \\ 1 & 1 & 1 & 1 \\ 0 & 0 & 1 & 0 \\ 3 & 2 & 1 & 0 \end{bmatrix} \begin{bmatrix} A_3 \\ A_2 \\ A_1 \\ A_0 \end{bmatrix} \quad (19.6)$$

Solve this equation by inverting the coefficient matrix as follows

$$\begin{aligned} \begin{bmatrix} A_3 \\ A_2 \\ A_1 \\ A_0 \end{bmatrix} &= \begin{bmatrix} 0 & 0 & 0 & 1 \\ 1 & 1 & 1 & 1 \\ 0 & 0 & 1 & 0 \\ 3 & 2 & 1 & 0 \end{bmatrix}^{-1} \begin{bmatrix} P \\ Q \\ P' \\ Q' \end{bmatrix} \\ &= \begin{bmatrix} 2 & -2 & 1 & 1 \\ -3 & 3 & -2 & -1 \\ 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} P \\ Q \\ P' \\ Q' \end{bmatrix} \end{aligned} \quad (19.7)$$

to see that the four coefficients A_0, A_1, A_2 and A_3 can indeed be derived from the four boundary constraints P, Q, P' and Q'. The 4×4 matrix in the second line of the equation is called the *Hermite matrix* and denoted M_H, so (19.7) is written concisely as

$$[A_3 \ A_2 \ A_1 \ A_0]^T = M_H \, [P \ Q \ P' \ Q']^T \quad (19.8)$$

Finally, let's use (19.2) to write c's equation in terms of its boundary constraints:

$$\begin{aligned} c(u) &= [u^3 \ u^2 \ u \ 1] \, [A_3 \ A_2 \ A_1 \ A_0]^T \\ &= [u^3 \ u^2 \ u \ 1] \, M_H \, [P \ Q \ P' \ Q']^T \\ &= (2u^3 - 3u^2 + 1) \, P + (-2u^3 + 3u^2) \, Q + \\ &\qquad (u^3 - 2u^2 + u) \, P' + (u^3 - u^2) \, Q' \end{aligned} \quad (19.9)$$

in $0 \le u \le 1$, after performing the matrix multiplications in the second line.

Therefore,

$$c(u) = H_0(u) \, P + H_1(u) \, Q + H_2(u) \, P' + H_3(u) \, Q' \qquad (0 \le u \le 1) \quad (19.10)$$

where the polynomials

$$\begin{aligned} H_0(u) &= 2u^3 - 3u^2 + 1, \qquad H_1(u) = -2u^3 + 3u^2, \\ H_2(u) &= u^3 - 2u^2 + u, \qquad H_3(u) = u^3 - u^2 \end{aligned}$$

are called *Hermite blending polynomials*, which, of course, are blending functions, but very different clearly from those used earlier in Bézier and B-spline theory; moreover, they blend not just control points, but tangent vectors as well, as one sees in (19.10). Their graphs are sketched in Figure 19.3. Certain symmetries are evident. Observe, as well, that $H_3(u)$ is non-positive in $0 \leq u \leq 1$, reaching a minimum value of nearly -0.15.

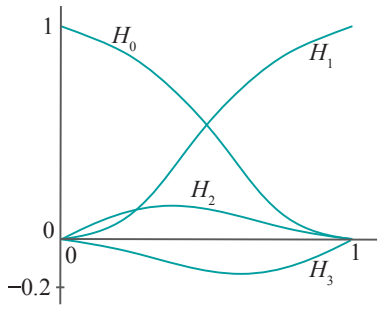

Figure 19.3: Hermite blending polynomials (not exact plots).

The curve $c(u)$ itself is called a *Hermite cubic*. Equation (19.10) is called the *geometric* form of the cubic because its expression is in terms of c's boundary constraints, while (19.1) is its *algebraic* form.

Remark 19.3. Readers familiar with the popular Adobe Illustrator drawing package will recognize that its pen tool, in fact, is used to draw Hermite cubics by specifying endpoints and editing tangents there.

Exercise 19.1. Use calculus to determine the maximum value of $H_2(u)$ and the minimum value of $H_3(u)$ in the interval $[0, 1]$.

Exercise 19.2. Determine the symmetries among the Hermite blending polynomials. For example, that $H_0(u)$ and $H_1(u)$ are mirror images across the vertical line $u = \frac{1}{2}$ down the middle of the parameter interval $[0, 1]$ can be seen by substituting $(1 - u)$ for u in the equation of one to obtain that of the other.

How about the relationship between $H_2(u)$ and $H_3(u)$? Do you see any symmetries?

Exercise 19.3. Prove the affine invariance of the cubic curve c given by Equation (19.10).

Note: Keeping in mind that an affine transformation is a linear transformation followed by a translation, we'll want its linear transformation part applied to all four boundary constraints P, Q, P' and Q', while the translation should apply only to P and Q.

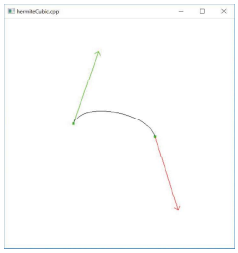

Figure 19.4: Screenshot of `hermiteCubic.cpp`.

Experiment 19.1. Run `hermiteCubic.cpp`, which implements Equation (19.10) to draw a Hermite cubic on a plane. Press space to select either a control point or tangent vector and the arrow keys to change it. Figure 19.4 is a screenshot. The actual cubic is simple to draw, but as you can see in the program we invested many lines of code to get the arrow heads right! End

Exercise 19.4. What sort of curve is c if the two boundary constraints P' and Q' are both zero (i.e., if the two end velocities vanish)? Determine this from the geometric form of the Hermite cubic and verify it in the preceding program.

It's interesting to contrast (19.10) with the equation of the cubic Bézier curve (Equation (17.8)):

$$c(u) = B_{0,3}(u)P_0 + B_{1,3}(u)P_1 + B_{2,3}(u)P_2 + B_{3,3}(u)P_3 \qquad (0 \leq u \leq 1)$$

In the case of the Bézier curve, the control points are blended with weights equal to the Bernstein polynomials of degree 3; in the case of the Hermite cubic, the two end control points and their respective tangents are blended with weights equal to the Hermite blending polynomials, which are of degree 3 as well.

Remark 19.4. Since a Hermite cubic interpolates not only its two specified control points, but also the specified tangents there, it's said to make a first-order interpolation (versus a zeroth-order one which would interpolate merely control points).

Let's return to the original problem of joining successive pairs of the $n+1$ control points $P_0, P_1, \ldots, P_n$ by means of cubic arcs so that the resulting Hermite spline is smooth. A strategy that comes to mind from the discussion above is to ask the designer to specify, in addition to the $n+1$ control points, the tangent vectors $P'_0, P'_1, \ldots, P'_n$ at each, as indicated in Figure 19.5.

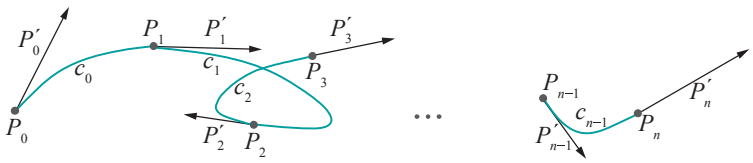

Figure 19.5: Specifying a Hermite spline by specifying the tangent vector at each control point.

Then, using (19.10) to manufacture each of the n successive Hermite cubic arcs c_i, $0 \le i \le n$, subject to the respective boundary constraints P_i, P_{i+1}, P'_i and P'_{i+1} yields a C^1-continuous Hermite spline, as the derivatives on either side of each internal control point agree.

However, asking the designer for $n+1$ tangent values, in addition to the control points themselves, may be a bit much. It would be nice to have an *automatic* way to deduce these tangent values from other constraints, *transparently* to the user. In fact, there is and we'll discuss next two popular types of Hermite splines arising from particular sets of constraints. These are the natural cubic and cardinal splines.

19.2 Natural Cubic Splines

A *natural cubic spline* is a Hermite spline with two constraints: (a) it is C^2-continuous, i.e., its second derivative is continuous, and (b) its second derivative vanishes at its two end control points. It turns out, as we'll see, that these two constraints are enough to uniquely determine the spline.

Assume that the $n+1$ control points through which a natural cubic spline passes are $P_0, P_1, \ldots, P_n$. Because of C^1-continuity – mind that C^2-continuity implies C^1-continuity – one assumes that the tangents at the control points are well-defined, in particular, that there are no corners and the value of the tangent is the same on either side of a control point. Say the tangent values at $P_0, P_1, \ldots, P_n$ are $P'_0, P'_1, \ldots, P'_n$, respectively. We'll compute these values from the constraints given.

Rewrite (19.9) as the equation of the cubic arc c_i from P_i to P_{i+1}:

$$\begin{aligned} c_i(u) \;\; = \;\; & (2u^3 - 3u^2 + 1)\, P_i \; + \; (-2u^3 + 3u^2)\, P_{i+1} \; + \\ & (u^3 - 2u^2 + u)\, P'_i \; + \; (u^3 - u^2)\, P'_{i+1} \quad (0 \le u \le 1) \end{aligned}$$

Differentiating twice one finds the second derivative

$$c''_i(u) = (12u-6)\, P_i + (-12u+6)\, P_{i+1} + (6u-4)\, P'_i + (6u-2)\, P'_{i+1} \quad (0 \le u \le 1) \quad (19.11)$$

Observe now that the constraints on a natural cubic spline through $P_0, P_1, \ldots, P_n$ can be written as the $n+1$ equations:

$$c''_0(0) = 0, \qquad c''_{i-1}(1) = c''_i(0), \quad \text{for} \;\; 1 \le i \le n-1, \qquad c''_{n-1}(1) = 0$$

the middle equations saying that the values of the second derivative on either side of each internal control point are equal, assuring C^2-continuity. Expand the constraint equations using (19.11):

$$\begin{aligned} -6P_0 + 6P_1 - 4P'_0 - 2P'_1 &= 0 \\ 6P_{i-1} - 6P_i + 2P'_{i-1} + 4P'_i &= -6P_i + 6P_{i+1} - 4P'_i - 2P'_{i+1}, 1 \le i \le n-1 \\ 6P_{n-1} - 6P_n + 2P'_{n-1} + 4P'_n &= 0 \end{aligned}$$

Simplifying and rearranging, we have the system

$$\begin{aligned} 2P'_0 + P'_1 &= -3P_0 + 3P_1 \\ P'_{i-1} + 4P'_i + P'_{i+1} &= -3P_{i-1} + 3P_{i+1}, \qquad 1 \le i \le n-1 \\ P'_{n-1} + 2P'_n &= -3P_{n-1} + 3P_n \end{aligned} \tag{19.12}$$

of $n+1$ equations in $n+1$ unknowns, which can be solved for the P'_i in terms of the P_i. In fact, writing out the system (19.12) in matrix form one obtains

$$\begin{bmatrix} 2 & 1 & 0 & 0 & 0 & 0 & \dots & 0 & 0 & 0 & 0 \\ 1 & 4 & 1 & 0 & 0 & 0 & \dots & 0 & 0 & 0 & 0 \\ 0 & 1 & 4 & 1 & 0 & 0 & \dots & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 4 & 1 & 0 & \dots & 0 & 0 & 0 & 0 \\ \dots & & \dots & & \dots & & & & \dots & & \\ 0 & 0 & 0 & 0 & 0 & 0 & \dots & 0 & 1 & 4 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & \dots & 0 & 0 & 1 & 2 \end{bmatrix} \begin{bmatrix} P'_0 \\ P'_1 \\ P'_2 \\ P'_3 \\ \dots \\ P'_{n-1} \\ P'_n \end{bmatrix} = \begin{bmatrix} -3P_0 + 3P_1 \\ -3P_0 + 3P_2 \\ -3P_1 + 3P_3 \\ -3P_2 + 3P_4 \\ \dots \\ -3P_{n-2} + 3P_n \\ -3P_{n-1} + 3P_n \end{bmatrix} \tag{19.13}$$

where the coefficient matrix is *tridiagonal* because it has non-zero entries only along the principal diagonal and its two neighboring diagonals. Tridiagonal matrices are particularly efficient to invert [116]; accordingly, equation systems with a tridiagonal coefficient matrix are efficiently solvable .

Consequently, using the solved values $P'_0, P'_1, \dots, P'_n$ from (19.13) and the geometric form (19.10) of the Hermite cubic, one determines the n Hermite cubic arcs between successive pairs from $P_0, P_1, \dots, P_n$. These arcs then join end to end to give the natural cubic spline through these $n+1$ control points.

Exercise 19.5. (Programming) Solve (19.13) by hand for only three control points P_0, P_1 and P_2. Write a program to draw a natural cubic spline through three control points, each of which can be moved on a plane.

Exercise 19.6. Investigate the local control (or lack thereof) of natural cubic splines. In particular, which of the cubic arcs of a natural cubic spline are affected by moving only one control point?
Hint: Playing with a natural cubic spline applet (there are many on the web) should suggest an answer.

19.3 Cardinal Splines

A *cardinal spline* is a C^1 Hermite spline whose tangent vector at each internal control point is determined by the location of its two adjacent control points in the following simple manner. Say the control points through which a cardinal spline passes are $P_0, P_1, \dots, P_n$. The tangent vector P'_i at P_i, $1 \le i \le n-1$, then is specified to be *parallel* to the vector from P_{i-1} to P_{i+1} by the equation

$$P'_i = \frac{1}{2}(1 - t_i)(P_{i+1} - P_{i-1}) \tag{19.14}$$

See Figure 19.6.

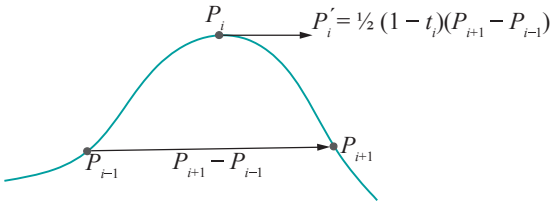

Figure 19.6: The tangent vector at an internal control point of a cardinal spline is parallel to the vector joining the adjacent control points – the tension parameter t_i is user-specified.

The constant of proportionality $\frac{1}{2}(1 - t_i)$ in (19.14) involves a designer-specified parameter t_i, called the *tension parameter*. The tension parameter is usually set between -1 and 1 at each internal control point, in turn setting $\frac{1}{2}(1 - t_i)$ between 1 and 0. If the tension parameter is set to 0 at *every* internal control point, one gets a popularly used special kind of cardinal spline called a *Catmull-Rom* spline. Specifically, the tangent vector at the internal control point P_i of a Catmull-Rom spline is

$$P_i' = \frac{1}{2}(P_{i+1} - P_{i-1}) \tag{19.15}$$

Now, from (19.14), $1 \leq i \leq n-1$, one has only $n-1$ equations in the $n+1$ unknowns P_i', $0 \leq i \leq n$. Therefore, two more are required to uniquely solve for these unknowns and determine the cardinal spline through P_i, $0 \leq i \leq n$. Typically, as in the case of a natural cubic spline, these are obtained from requiring the second derivatives to vanish at the two end control points.

Exercise 19.7. Write a matrix equation analogous to (19.13) relating P_i' to P_i for a cardinal spline, assuming the additional constraints that the second derivatives vanish at the terminal control points. Is the coefficient matrix tridiagonal?

Exercise 19.8. What can you say of local control in cardinal splines? In other words, which of the cubic arcs of a cardinal spline are affected by moving a specific control point?

Exercise 19.9. Natural cubic splines are C^2 by definition. How about cardinal splines – are they C^2?
Hint: The answer is no in general and we ask the reader to try and come up with a counter-example. A Catmull-Rom spline through three control points which loses C^2-continuity in the middle is probably easiest.

19.4 Hermite Surface Patches

We'll give a brief introduction to the 2D version of Hermite curves, namely, Hermite surfaces. Analogously to (19.1), one can write the parametric equation of a *Hermite surface patch* (or *bicubic surface patch*) in algebraic form as

$$\begin{aligned} s(u,v) &= \sum_{i=0}^{3}\sum_{j=0}^{3} A_{i,j} u^i v^j \\ &= A_{3,3}\,u^3v^3 + A_{3,2}\,u^3v^2 + A_{3,1}\,u^3v + A_{3,0}\,u^3 \\ &\quad + A_{2,3}\,u^2v^3 + A_{2,2}\,u^2v^2 + A_{2,1}\,u^2v + A_{2,0}\,u^2 \\ &\quad + A_{1,3}\,uv^3 + A_{1,2}\,uv^2 + A_{1,1}\,uv + A_{1,0}\,u \\ &\quad + A_{0,3}\,v^3 + A_{0,2}\,v^2 + A_{0,1}\,v + A_{0,0} \end{aligned} \tag{19.16}$$

for $0 \leq u, v \leq 1$. The expression after the second equality consists of 16 monomial summands, where A_{ij}, $0 \leq i \leq 3$, $0 \leq j \leq 3$, are points in the ambient space.

Going back to curves for a moment, observe that the geometric form (19.10), viz.,

$$c(u) = H_0(u)\,P + H_1(u)\,Q + H_2(u)\,P' + H_3(u)\,Q'$$

of the equation of a Hermite cubic is more useful than the algebraic (19.1), viz.,

$$c(u) = A_3u^3 + A_2u^2 + A_1u + A_0$$

because it gives an equation in terms of *perceptible* boundary constraints, in particular, the endpoints P and Q and the tangent vectors P' and Q' there. Moreover, we were able to derive the algebraic form from the geometric because these four boundary constraints were sufficient to uniquely recover the four coefficients A_i, $0 \le i \le 3$, of the algebraic form.

So what would be a suitable set of boundary constraints for a geometric form of the equation of a Hermite patch? Clearly, one would want sixteen constraints leading to a unique determination of the sixteen coefficients A_{ij}, $0 \le i \le 3$, $0 \le j \le 3$, on the RHS of (19.16).

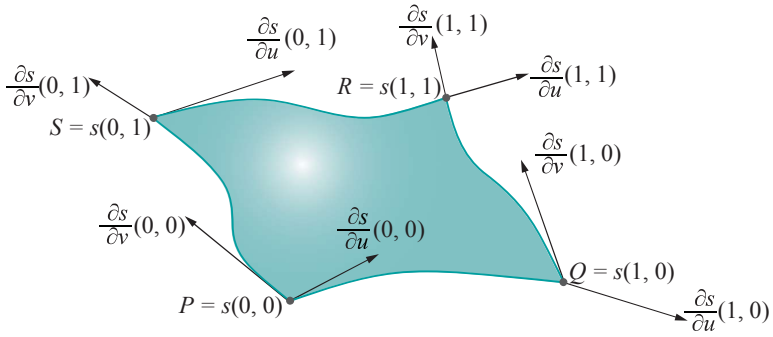

Figure 19.7: Twelve boundary constraints on a bicubic patch.

Twelve choices are fairly clear. See Figure 19.7. Firstly, the four corners $s(0,0), s(1,0), s(1,1), s(0,1)$ of the patch s evidently evidently coincide with user-specified control points P, Q, R and S, respectively. Next, analogous to asking for end tangent vectors in the case of a curve, the values of the partial derivatives with respect to u and v at each corner provide eight more constraints. Observe that the two partial derivatives at each corner are nothing but the tangent vectors to the two boundary curves meeting there. Four remaining boundary constraints are up to the designer, but are usually taken to be values of the second-order mixed partial derivatives at the corners, namely,

$$\frac{\partial^2 s}{\partial u \partial v}(0,0), \qquad \frac{\partial^2 s}{\partial u \partial v}(1,0), \qquad \frac{\partial^2 s}{\partial u \partial v}(0,1), \qquad \frac{\partial^2 s}{\partial u \partial v}(1,1)$$

These four are called *twist vectors* and have geometric significance too – though not as straightforwardly as the first twelve – which we'll not go into here.

We'll conclude our discussion by saying that it turns out that, indeed, the four corner position vectors, the eight tangent vectors at the corners and the four twist vectors together provide sixteen boundary constraints which are sufficient to uniquely specify a Hermite patch. We'll not go further into the derivation ourselves, but refer the interested reader to the chapter on Hermite surfaces in the book by Mortenson [97].

Lagrange Interpolation

At the conclusion of this chapter, we'll briefly describe a method of polynomial (in fact, *entirely* polynomial, not piecewise like Hermite) interpolation, called *Lagrange interpolation*, actually of more theoretical interest than practical value in design.

The *Lagrange polynomial* $f_{i,n}$, where n is a positive integer and i is an integer between 0 and n, is defined by the equation

$$f_{i,n}(u) = \prod_{0 \le j \le n,\ j \ne i} \frac{u-j}{i-j}$$

For example,

$$\begin{aligned} f_{2,4}(u) &= \frac{(u-0)(u-1)(u-3)(u-4)}{(2-0)(2-1)(2-3)(2-4)} \\ &= \frac{1}{4}\,u(u-1)(u-3)(u-4) \end{aligned}$$

Lagrange polynomials have the easily verified property that

$$f_{i,n}(u) = \begin{cases} 1, & u = i \\ 0, & u \in \{0, 1, \ldots, n\},\ u \ne i \end{cases}$$

In other words, on the particular set of integers $\{0, 1, \ldots, n\}$, the Lagrange polynomial $f_{i,n}$ is 1 at exactly one point, namely i, and 0, elsewhere.

Exercise 19.10. Write the formula for $f_{0,4}(u)$ and check it for the above-mentioned property.

If, now, one uses the Lagrange polynomials as blending functions for $n+1$ control points P_i, $0 \le i \le n$, obtaining the curve

$$c(u) = f_{0,n}(u)P_0 + f_{1,n}(u)P_1 + \ldots + f_{n,n}(u)P_n \qquad (0 \le u \le n)$$

then c, called a *Lagrange curve*, is a polynomial curve of degree n. It's seen easily from its definition that c interpolates all its control points; in particular, c is equal to P_i at the point i of the parameter domain $[0, n]$, for $0 \le i \le n$.

Exercise 19.11. Write the formula for the Lagrange curve interpolating the four control points

$$[0\ \ -1\ 3]^T \qquad [1\ 2\ \ -3]^T \qquad [5\ \ -1\ 4]^T \qquad [2\ 0\ 8]^T$$

Remark 19.5. Lagrange interpolation is rarely used in practice because it suffers from the Bézier-like problem that the degree of the interpolating curve grows with its number of control points. It lacks local control as well.

19.5 Summary, Notes and More Reading

After a couple of chapters on Bézier and B-spline approximation of control points, we learned in this chapter practical methods to interpolate. These will come in handy in design applications that do require interpolation and most 3D modelers, in fact, offer at least a flavor or two of Hermite interpolation, such as natural cubic and Catmull-Rom splines. It's true, though, in the majority of real-life applications that the only known hard constraints on a curve or surface are at its boundary, e.g., by the way a surface patch meets its neighbors, so the designer typically prefers using internal control points as attractors *a la* Bézier or B-spline, rather than having them tightly latched to an interpolating curve or surface.

For more about Hermite interpolation the reader should consult Farin [45] and Mortenson [97].