5
Spline Interpolation

Given a set of points, it is easy to compute a polynomial that passes through the points. The LP of Section 3.2 is an example of such a polynomial. However, as the discussion in Section 1.5 (especially exercise 1.20) illustrates, a curve based on a high-degree polynomial may wiggle wildly and its shape may be far from what the user has in mind. In practical work we are normally interested in a smooth, tight curve that proceeds from point to point such that each segment between two points is a smooth arc. The spline approach to curve design, discussed in this chapter, constructs such a curve from individual segments, each a simple curve, generally a parametric cubic (PC). This chapter illustrates spline interpolation with three examples, cubic splines (Section 5.1), cardinal splines (Section 5.4), and Kochanek–Bartels splines (Section 5.6). Another important type, the B-spline, is the topic of Chapter 7. Other types of splines are known and are discussed in the scientific literature. A short history of splines can be found in [Schumaker 81] and [Farin 04].

Definition: A spline is a set of polynomials of degree k that are smoothly connected at certain data points. At each data point, two polynomials connect, and their first derivatives (tangent vectors) have the same values. The definition also requires that all their derivatives up to the $(k-1)$st be the same at the point.

5.1 The Cubic Spline Curve

The cubic spline was originally introduced by James Ferguson in [Ferguson 64]. Given n data points that are numbered $\mathbf{P}_1$ through $\mathbf{P}_n$, there are infinitely many curves that pass through all the points in order of their numbers (Figure 5.1a), but the eye often tends to trace *one* imaginary smooth curve through the points, especially if the points are arranged in a familiar pattern. It is therefore useful to have an algorithm that does the same. Since the computer does not recognize familiar patterns the way humans do,

such a method should be interactive, thereby allowing the user to create the desired curve.

The cubic spline method is such an algorithm. Given n data points, it constructs a smooth curve that passes through the points (see definition of data points in Section 1.3). The curve consists of $n-1$ individual Hermite segments that are smoothly connected at the $n-2$ interior points and that are easy to calculate and display. For the segments to meet at the interior points, their tangent vectors (first derivatives) must be the same at each interior point. An added feature of cubic splines is that their second derivatives are also the same at the interior points. The cubic spline method is interactive. The user can control the shape of the curve by varying the two extreme tangent vectors at the beginning and the end of the curve.

Given the n data points $\mathbf{P}_1$, $\mathbf{P}_2$, through $\mathbf{P}_n$, we look for $n-1$ parametric cubics $\mathbf{P}_1(t)$, $\mathbf{P}_2(t), \ldots, \mathbf{P}_{n-1}(t)$ such that $\mathbf{P}_k(t)$ is the polynomial segment from point $\mathbf{P}_k$ to point $\mathbf{P}_{k+1}$ (Figure 5.1b). The PCs will have to be smoothly connected at the $n-2$ interior points $\mathbf{P}_2, \mathbf{P}_3, \ldots, \mathbf{P}_{n-1}$, which means that their first derivatives will have to match at every interior point. The definition of a spline requires that their second derivatives match too. This requirement (the boundary condition of the cubic spline) is important because it provides the necessary equations and also results in a tight curve in the sense that once the curve is drawn, the eye can no longer detect the positions of the original data points.

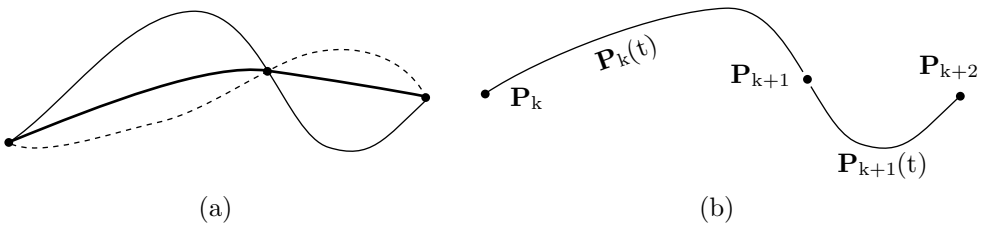

Figure 5.1: (a) Three Different Curves. (b) Two Segments.

The principle of cubic splines is to divide the set of n points into $n-1$ overlapping pairs of two points each and to fit a Hermite segment [Equations (4.4) and (4.5)] to each pair. The pairs are $(\mathbf{P}_1, \mathbf{P}_2)$, $(\mathbf{P}_2, \mathbf{P}_3)$, and so on, up to $(\mathbf{P}_{n-1}, \mathbf{P}_n)$. Recall that a Hermite curve segment is specified by two points and two tangents. In our case, all the points are given, so the only unknowns are the tangent vectors. In order for segments $\mathbf{P}_k(t)$ and $\mathbf{P}_{k+1}(t)$ to connect smoothly at point $\mathbf{P}_{k+1}$, the end tangent of $\mathbf{P}_k(t)$ has to equal the start tangent of $\mathbf{P}_{k+1}(t)$. Thus, there is only one tangent vector per point, for a total of n unknowns.

The unknown tangent vectors are computed as the solutions of a system of n equations. The equations are derived from the requirement that the second derivatives of the individual segments match at every interior point. However, there are only $n-2$ interior points, so we can only have $n-2$ equations, enough to solve for only $n-2$ unknowns.

The key to resolving this shortage of equations is to ask the user to provide the software with the values of two tangent vectors (normally the first and last ones). Once this is done, the equations can easily be solved, yielding the remaining $n-2$ tangents.