C^2-interpolating curves

Reading

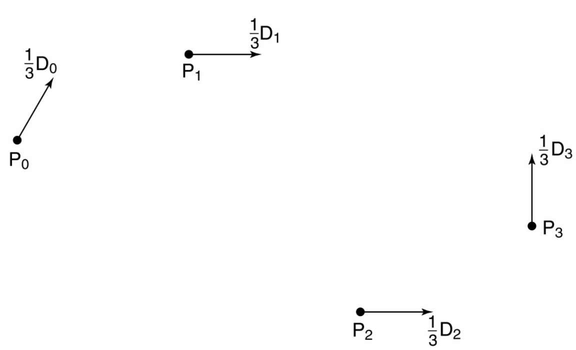
Optional

- Bartels, Beatty, and Barsky. *An Introduction to Splines for use in Computer Graphics and Geometric Modeling*, 1987. (Handout)

C² interpolating splines

How can we keep the C^2 continuity we get with B-splines but get interpolation, too?

Here's the idea behind ***C*² interpolating splines**. Suppose we had cubic Béziers connecting our control points P_0, P_1, P_2, …, P_m and that we somehow knew the first derivative of the spline at each point.



Let's say (V_0,V_1,V_2,V_3) are the first set of control points, and (W_0,W_1,W_2,W_3) are the second set. What are the *V*'s and *W's* in terms of *P*'s and *D*'s?

Finding the derivatives

We can write out these relationships as:

$$V_0 = P_0 \qquad W_0 = P_1$$
$$V_1 = P_0 + \tfrac{1}{3}D_0 \qquad W_1 = P_1 + \tfrac{1}{3}D_1$$
$$V_2 = P_1 - \tfrac{1}{3}D_1 \qquad W_2 = P_2 - \tfrac{1}{3}D_2$$
$$V_3 = P_1 \qquad W_3 = P_2$$

Now what we need to do is solve for the derivatives. These equations already imply C^0 and C^1 continuity.

Now we'll add C^2 continuity :

$$Q''_V(1) = Q''_W(0)$$

$$6(V_1 - 2V_2 + V_3) = 6(W_0 - 2W_1 + W_2)$$

Substituting the top set of equations into this last equation, we find:

$$D_0 + 4D_1 + D_2 = 3(P_2 - P_0)$$

Finding the derivatives, cont.

We can repeat this analysis for every pair of neighboring Bezier curve segments, giving us:

$$\begin{aligned} D_0 + 4D_1 + D_2 &= 3(P_2 - P_0) \\ D_1 + 4D_2 + D_3 &= 3(P_3 - P_1) \\ &\vdots \\ D_{m-2} + 4D_{m-1} + D_m &= 3(P_m - P_{m-2}) \end{aligned}$$

How many equations is this? m-1

How many unknowns are we solving for? m+1

Not quite done yet

We have two additional degrees of freedom, which we can nail down by imposing more conditions on the curve.

There are various ways to do this. We'll use the variant called **natural C^2 interpolating splines**, which requires the second derivative to be zero at the endpoints.

This condition gives us the two additional equations we need. At the P_0 endpoint, it is:

$$Q_V''(0) = 6(V_0 - 2V_1 + V_2) = 0$$

Let's say that the last set of control points are (U_0,U_1,U_2,U_3). Then, at the P_m endpoint, we have:

$$Q_U''(1) = 6(U_1 - 2U_2 + U_3) = 0$$

These constraints imply:

$$\begin{aligned} 2D_0 + D_1 &= 3(P_1 - P_0) \\ D_{m-1} + 2D_m &= 3(P_m - P_{m-1}) \end{aligned}$$

Solving for the derivatives

Let's collect our m+1 equations into a single linear system:

$$\begin{bmatrix} 2 & 1 & & & & \\ 1 & 4 & 1 & & & \\ & 1 & 4 & 1 & & \\ & & & \ddots & & \\ & & & 1 & 4 & 1 \\ & & & & 1 & 2 \end{bmatrix} \begin{bmatrix} D_0^T \\ D_1^T \\ D_2^T \\ \vdots \\ D_{m-1}^T \\ D_m^T \end{bmatrix} = \begin{bmatrix} 3(P_1 - P_0)^T \\ 3(P_2 - P_0)^T \\ 3(P_3 - P_1)^T \\ \vdots \\ 3(P_m - P_{m-2})^T \\ 3(P_m - P_{m-1})^T \end{bmatrix}$$

It's easier to solve than it looks. [Note: the elements in the vectors are each points which are represented with their transposes to make the math work out.]

We can use **forward elimination** to zero out everything below the diagonal, then **back substitution** to compute each D value.

Note: technically speaking, we need to put the transposes of D and P vectors in the matrices. We'll omit this for ease of reading.

Forward elimination

First, for notational convenience, we set re-label the righthand side. Then, we eliminate the elements below the diagonal:

* (-1/2) + (applied on both sides, from row 0 to row 1)

$$\begin{bmatrix} 2 & 1 & & & & \\ 1 & 4 & 1 & & & \\ & 1 & 4 & 1 & & \\ & & & \ddots & & \\ & & & 1 & 4 & 1 \\ & & & & 1 & 2 \end{bmatrix} \begin{bmatrix} D_0^T \\ D_1^T \\ D_2^T \\ \vdots \\ D_{m-1}^T \\ D_m^T \end{bmatrix} = \begin{bmatrix} E_0^T \\ E_1^T \\ E_2^T \\ \vdots \\ E_{m-1}^T \\ E_m^T \end{bmatrix}$$

$$\begin{bmatrix} 2 & 1 & & & & \\ 0 & 7/2 & 1 & & & \\ & 1 & 4 & 1 & & \\ & & & \ddots & & \\ & & & 1 & 4 & 1 \\ & & & & 1 & 2 \end{bmatrix} \begin{bmatrix} D_0^T \\ D_1^T \\ D_2^T \\ \vdots \\ D_{m-1}^T \\ D_m^T \end{bmatrix} = \begin{bmatrix} F_0^T = E_0^T \\ F_1^T = E_1^T - (1/2)E_0^T \\ E_2^T \\ \vdots \\ E_{m-1}^T \\ E_m^T \end{bmatrix}$$

Back subsitution

The resulting matrix is **upper diagonal**:

$$\mathbf{UD} = \mathbf{F}$$

$$\begin{bmatrix} u_{11} & \cdots & u_{1m} \\ & \ddots & \vdots \\ & & u_{mm} \end{bmatrix} \begin{bmatrix} D_0^T \\ D_1^T \\ D_2^T \\ \vdots \\ D_{m-1}^T \\ D_m^T \end{bmatrix} = \begin{bmatrix} F_0^T \\ F_1^T \\ F_2^T \\ \vdots \\ F_{m-1}^T \\ F_m^T \end{bmatrix}$$

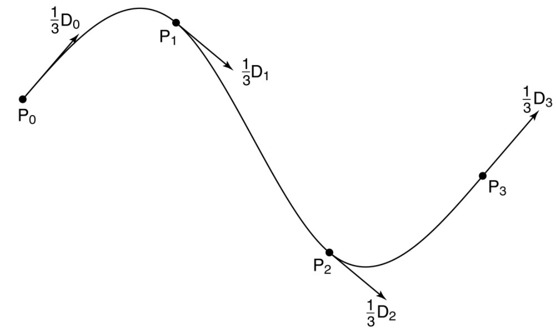
We can now solve for the unknowns by back substitution (where we can drop the transposes for the moment):

$$u_{mm} D_m = F_m$$
$$u_{m\text{-}1m\text{-}1} D_{m\text{-}1} + u_{m\text{-}1m} D_m = F_{m\text{-}1}$$

See the notes from Bartels, Beatty, and Barsky for more implementation details.

C^2 interpolating spline

Once we've solved for the real D_is, we can plug them in to find our Bézier control points and draw the final spline:



Have we lost anything?

=> Yes, local control.

Closing the loop

With C^2 interpolating splines, we have to modify the matrix for closed loops:

$$\begin{bmatrix} 4 & 1 & & & & 1 \\ 1 & 4 & 1 & & & \\ & 1 & 4 & 1 & & \\ & & & \ddots & & \\ & & & 1 & 4 & 1 \\ 1 & & & & 1 & 4 \end{bmatrix} \begin{bmatrix} D_0^T \\ D_1^T \\ D_2^T \\ \vdots \\ D_{m-1}^T \\ D_m^T \end{bmatrix} = \begin{bmatrix} 3(P_1 - P_m)^T \\ 3(P_2 - P_0)^T \\ 3(P_3 - P_1)^T \\ \vdots \\ 3(P_m - P_{m-2})^T \\ 3(P_0 - P_{m-1})^T \end{bmatrix}$$

We can use a *modified* forward elimination to zero out everything below the diagonal, then back substitution to compute each D value.

See the notes from Bartels, Beatty, and Barsky for more implementation details.

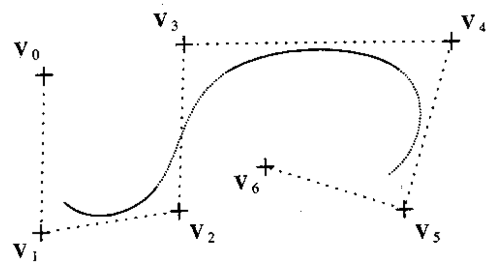


Figure 2.3. An example of a curve that approximates a sequence of points, represented here by "+" signs. The lightly dotted line connecting the points indicates the order in which they are to be approximated. The solid and heavily dotted curves represent distinct curve *segments*. Each is a single parametric cubic. The point at which two successive segments meet is called a *joint*. The value of the parameter $\bar{u}$ which corresponds to a joint is called a *knot*.

3

Hermite and Cubic Spline Interpolation

Suppose that we have $m + 1$ data points $\mathbf{P}_0, \ldots, \mathbf{P}_m$ through which we wish to draw a curve such as that shown in Figure 3.1 (in which $m = 6$).

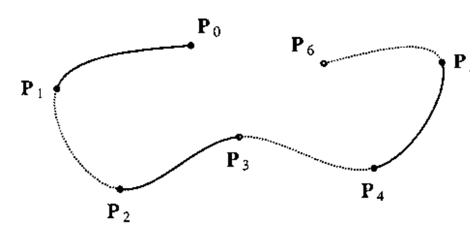


Figure 3.1. An interpolating cubic spline.

Each successive pair of data points is connected by a distinct curve segment. The i^{th} segment runs from $\mathbf{P}_i$ to $\mathbf{P}_{i+1}$, and we will assume that the parameter $\bar{u}$ runs correspondingly from the knot $\bar{u}_i$ to the knot $\bar{u}_{i+1}$ to generate this segment. This corresponds to the knot sequence and parameter range outlined in Chapter 2 with the special choices $\bar{u}_0 = \bar{u}_j = \bar{u}_{min}$ and $\bar{u}_{max} = \bar{u}_\ell = \bar{u}_m = \bar{u}_{last}$. Since each such segment $\mathbf{Q}_i(\bar{u})$ is represented parametrically as $(X_i(\bar{u}), Y_i(\bar{u}))$, we are really concerned with how the $X_i(\bar{u})$ and $Y_i(\bar{u})$ are determined by the points

$$\mathbf{P}_i = (x_i, y_i).$$

In general, the x-coordinates $X(\bar{u})$ of points on a curve are determined solely by the x-coordinates $x_0, \ldots, x_m$ of the data points, and similarly $Y(\bar{u})$ is determined solely by the y-coordinates of the data points. Since both $X(\bar{u})$ and $Y(\bar{u})$ are treated in the same way we will discuss only $Y(\bar{u})$; indeed, to obtain curves in three dimensions we simply define a $Z(\bar{u})$ as well and let $\mathbf{Q}_i(u)$ be given by $(X_i(u), Y_i(u), Z_i(u))$.

For ease of computation we will limit ourselves to the use of polynomials in defining $X_i(u)$, $Y_i(u)$ and $Z_i(u)$. Indeed cubic polynomials usually provide sufficient flexibility for many applications at reasonable cost. For the curve in Figure 3.1, then, $Y(\bar{u})$ is shown in Figure 3.2.

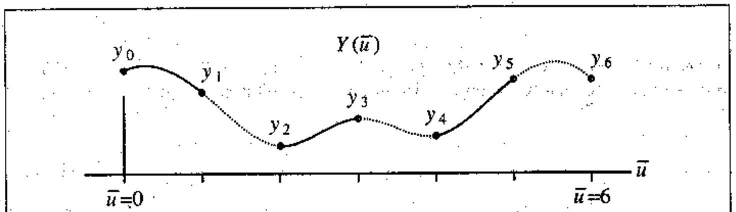


Figure 3.2. $Y(\bar{u})$ for the curve shown in Figure 3.1 above. In this example we have rather arbitrarily chosen to use uniform knot spacing, so that the knot sequence is (0,1,2,3,4,5,6).

It will be easiest to continue the discussion by reparametrizing each segment Y_i separately by substituting u for $\bar{u}$ as was described earlier. This means that $u = \bar{u}_i - i$ for the knot sequence given in Figure 3.2. Each $Y_i(u)$ is a cubic polynomial in the parameter u. We know two things in particular about

$$Y_i(u) = a_i + b_i u + c_i u^2 + d_i u^3,$$

namely that

$$Y_i(0) = y_i = a_i$$

$$Y_i(1) = y_{i+1} = a_i + b_i + c_i + d_i.$$

Because we have four coefficients to determine, we need two other constraints to completely determine a particular $Y_i(u)$. One easy way to do this is to simply pick, arbitrarily, first derivatives D_i of $Y(u)$ at each knot $\bar{u}_i$, so that

$$Y_i^{(1)}(0) = D_i = b_i$$

$$Y_i^{(1)}(1) = D_{i+1} = b_i + 2c_i + 3d_i.$$

These four equations can be solved symbolically, once and for all, to yield

$$a_i = y_i$$

$$b_i = D_i$$

$$c_i = 3(y_{i+1} - y_i) - 2D_i - D_{i+1} \qquad (3.1)$$

$$d_i = 2(y_i - y_{i+1}) + D_i + D_{i+1}.$$

Since we use D_i as the derivative at the left end of the i^{th} segment (i.e., as $Y_i^{(1)}(0)$) and at the right end of the $(i-1)^{th}$ segment (as $Y_{i-1}^{(1)}(1)$), $Y(u)$ has a continuous first derivative.

This technique is called *Hermite interpolation*. It can be generalized to higher-order polynomials.

How are the D_i specified? One possibility is to compute them automatically, perhaps by fitting a parabola through y_{i-1}, y_i, and y_{i+1}, and using its derivative at y_i as D_i; arbitrary values (such as 0) can be used at the end points [Kochanek/et al.82]. Or one can use for D_i the y component of a weighted average of the vector from P_{i-1} to P_i and the vector from P_{i+1} to P_i [Kochanek/Bartels84]. Or the user may specify derivative vectors directly. Some of these possibilities are discussed later in Chapter 21.

It is possible to arrange that successive segments match second as well as first derivatives at joints, using only cubic polynomials. Suppose, as above, that we want to interpolate the $(m+1)$ points $\mathbf{P}_0, \ldots, \mathbf{P}_m$ by such a curve. Each of the m segments $Y_0(u), \ldots, Y_{m-1}(u)$ is a cubic polynomial determined by four coefficients. Hence we have $4m$ unknown values to determine. At each of the $(m-1)$ interior knots $\bar{u}_1, \ldots, \bar{u}_{m-1}$ (where two segments meet) we have four conditions:

$$Y_{i-1}(1) = y_i, \quad Y_{i-1}^{(1)}(1) = Y_i^{(1)}(0)$$

$$Y_i(0) = y_i, \quad Y_{i-1}^{(2)}(1) = Y_i^{(2)}(0).$$

Since we also require that

$$Y_0(0) = y_0$$

$$Y_{m-1}(1) = y_m$$

we have a total of $4(m-1)+2 = 4m-2$ conditions from which to determine our $4m$ unknowns. Thus, we need two more conditions. These may be chosen in a variety of ways. A common choice is simply to require that the second derivatives at the endpoints $\bar{u}_0$ and $\bar{u}_m$ both be zero; these conditions yield what is called a *natural cubic spline*. Figure 3.7 is actually a natural cubic spline.

3.1 Practical Considerations — Computing Natural Cubic Splines

We do not need to solve $4m$ equations directly — the problem can be simplified. Notice that a natural cubic spline is actually a special case of Hermite interpolation; we may simply choose first derivative vectors so as to match second derivatives as well. If we can compute the needed D_i, we have already obtained definitions of the a_i, b_i, c_i and d_i in terms of the D_i.

Thus at each internal joint we want to choose D_i so that

$$Y_{i-1}^{(2)}(1) = Y_i^{(2)}(0)$$

or

$$2c_{i-1} + 6d_{i-1} = 2c_i .$$

Substituting in our earlier solutions (3.1) for c_{i-1}, d_{i-1} and c_i, we have

$$2[3(y_i - y_{i-1}) - 2D_{i-1} - D_i] + 6[2(y_{i-1} - y_i) + D_{i-1} + D_i]$$

$$= 2[3(y_{i+1} - y_i) - 2D_i - D_{i+1}] .$$

Simplifying, and moving the unknowns to the left, we have

$$D_{i-1} + 4D_i + D_{i+1} = 3(y_{i+1} - y_{i-1}) . \qquad (3.2)$$

Since there are $m-1$ internal joints, there are $m-1$ such equations. Requiring that the second derivative at the beginning of the curve be zero implies that

$$2c_0 = 0$$

$$2[3(y_1 - y_0) - 2D_0 - D_1] = 0$$

$$2D_0 + D_1 = 3(y_1 - y_0) .$$

Requiring that the second derivative at the end of the curve be zero similarly results in

$$D_{m-1} + 2D_m = 3(y_m - y_{m-1}) .$$

We now have $m+1$ equations in $m+1$ unknowns. Representing them in matrix form we have

$$\begin{bmatrix} 2 & 1 & & & & & \\ 1 & 4 & 1 & & & & \\ & 1 & 4 & 1 & & & \\ & & 1 & 4 & 1 & & \\ & & & \cdots & \cdots & & \\ & & & & 1 & 4 & 1 \\ & & & & & 1 & 2 \end{bmatrix} \begin{bmatrix} D_0 \\ D_1 \\ \cdot \\ \cdot \\ \cdot \\ \cdot \\ D_m \end{bmatrix} = \begin{bmatrix} 3(y_1 - y_0) \\ 3(y_2 - y_0) \\ \cdot \\ \cdot \\ \cdot \\ 3(y_m - y_{m-2}) \\ 3(y_m - y_{m-1}) \end{bmatrix} .$$

Beginning at the top, the first 1 in each row is eliminated using the row immediately above and the diagonal is scaled:

$\gamma_0 \leftarrow 1/2$

for $i \leftarrow 1$ **step 1 until** $m-1$ **do**

  $\gamma_i \leftarrow 1/(4-\gamma_{i-1})$

endfor

$\gamma_m \leftarrow 1/(2-\gamma_{m-1})$.

Corresponding operations are carried out on the right-hand-side entries; e.g., for the y components shown above:

$\delta_0 \leftarrow 3(y_1 - y_0)\gamma_0$

for $i \leftarrow 1$ **step 1 until** $m-1$ **do**

  $\delta_i \leftarrow (3(y_{i+1} - y_{i-1}) - \delta_{i-1})\gamma_i$

endfor

$\delta_m \leftarrow (3(y_m - y_{m-1}) - \delta_{m-1})\gamma_m$.

The result of this *forward elimination* process will be

$$\begin{bmatrix} 1 & \gamma_0 & & & & \\ & 1 & \gamma_1 & & & \\ & & 1 & \gamma_2 & & \\ & & & \ddots & & \\ & & & & \gamma_{m-2} & \\ & & & & 1 & \gamma_{m-1} \\ & & & & & 1 \end{bmatrix} \begin{bmatrix} D_0 \\ D_1 \\ \cdot \\ \cdot \\ \cdot \\ \cdot \\ D_m \end{bmatrix} = \begin{bmatrix} \delta_0 \\ \delta_1 \\ \cdot \\ \cdot \\ \cdot \\ \cdot \\ \delta_m \end{bmatrix}.$$

This directly yields the value of D_m, and it is then a simple matter to solve successively for $D_{m-1}, \ldots, D_0$ in a process of *backward substitution*:

$D_m \leftarrow \delta_m$

for $i \leftarrow m-1$ **step -1 until** 0 **do**

$D_i \leftarrow \delta_i - \gamma_i D_{i+1}$

endfor.

The multiplicative factors γ_i that accomplish the forward substitution need only be computed once. The δ_i's must be computed and the backward substitution performed separately for each coordinate. When a data point is moved, the values $\delta_j, \ldots, \delta_m$ must be recomputed and the entire backward substitution again performed.

3.2 Other End Conditions For Cubic Interpolating Splines

There are many other ways in which to determine the additional two constraints needed to define a C^2 continuous interpolating cubic spline fully. These conditions are most commonly applied to the ends of a curve, hence the name *end conditions*; the natural cubic splines offer an example of this. However, all that is really necessary is to provide the missing two conditions. Any two linear equations that are independent of those provided by the interpolation conditions could be used. They could involve data points or derivatives interior to the curve as well as at the ends. Whatever conditions are used, they will have some influence over the shape of the entire curve. For example, instead of fixing the second derivatives at the first and last knot to zero, we may fix the first derivatives there to be zero.

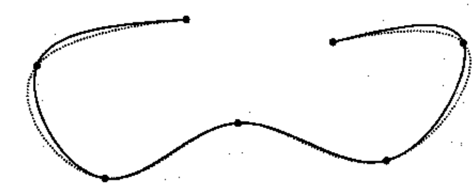


Figure 3.3. The solid line is a natural cubic interpolating spline; that is, the second parametric derivatives at the ends of the solid curve are zero. For the dotted curve the first derivatives at the ends have been set to zero instead.

Another possibility, which de Boor calls the *not-a-knot* condition [de Boor78], is to require C^3 continuity at the second and next-to-last knots u_1 and u_{m-1}. In effect the first two segments are a single polynomial, as are the last two.

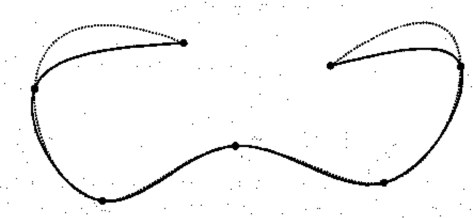


Figure 3.4. The solid line is a natural cubic interpolating spline. For the dotted curve, C^3 continuity has been forced between the first and second segments, and between the last and the next-to-last segments.

Yet another alternative, suggested by Forsythe, Malcolm and Moler [Forsythe/et al.77], is to use the third derivatives of the cubic polynomials that interpolate the first and last four points as the third derivatives of the first and last segments.

One might allow the user to explicitly supply any two of the first, second, or third derivative vectors at the ends. In any case, we can construct and solve a set of equations very much as we did for the natural cubic splines. Additional discussion of how this can be done, and algorithms, are given in Chapter 4 of [Forsythe/et al.77] and in Chapter 4 of [de Boor78]. For a uniform knot vector, and indeed for any reasonable strictly increasing sequence of knots, these equations are well conditioned and can be solved easily and accurately.

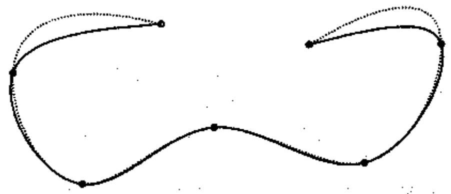


Figure 3.5. The solid line is a natural cubic interpolating spline. For the dotted curve the third derivative of the polynomial that interpolates the first four points is used as the (constant) third derivative of the first segment, and similarly for the last segment.

3.3 Knot Spacing

Although the end conditions discussed above affect the entire curve, their principal influence is felt at the endpoints. Gross changes to a curve's shape can be made anywhere, without moving the interpolation points, by varying the knot spacing. (See Figure 3.6.)

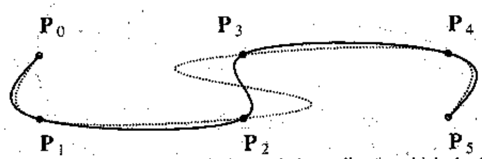


Figure 3.6. The solid line is a natural cubic interpolating spline in which the knots are spaced one unit apart. Unit knot spacing is used also in the dotted curve except for the parametric interval corresponding to the segment between $\mathbf{P}_2$ and $\mathbf{P}_3$, for which the knots are spaced four units apart.

With the single exception of Figure 3.6, we have used a uniform knot sequence in defining the interpolating cubic spline curves discussed above. The knot vector for the solid curve in Figure 3.6 is

0, 1, 2, 3, 4, 5

while the dotted curve interpolates the same data points, but for the knot vector

0, 1, 2, 6, 7, 8.

Thus knot spacing can be used to influence shape; the more difficult question is how that influence can be controlled intuitively.

Uniform knot spacing is one obvious way to define a knot sequence. The Euclidean distance between data points is a second natural choice for the length of the parametric interval over which u varies in defining a segment.

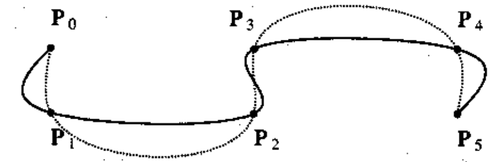


Figure 3.7. The solid line in the above figure is a natural cubic interpolating spline in which the knots are spaced a unit apart. In the case of the dotted curve, the knots corresponding to two successive data points differ in value by the Euclidean distance separating the two points.

3.4 Closed Curves

It is sometimes useful to generate closed curves such as in Figure 3.8. In this case, equation (3.2) applies at each of the m points, with the caveat that indices must be computed modulo $m+1$. The system of equations that results looks a little different:

$$\begin{bmatrix} 4 & 1 & & & & & 1 \\ 1 & 4 & 1 & & & & \\ & 1 & 4 & 1 & & & \\ & & 1 & 4 & 1 & & \\ & & & \cdot & \cdot & \cdot & \\ & & & & 1 & 4 & 1 \\ 1 & & & & & 1 & 4 \end{bmatrix} \begin{bmatrix} D_0 \\ D_1 \\ \cdot \\ \cdot \\ \cdot \\ \cdot \\ D_m \end{bmatrix} = \begin{bmatrix} 3(y_1-y_m) \\ 3(y_2-y_0) \\ \cdot \\ \cdot \\ \cdot \\ 3(y_m-y_{m-2}) \\ 3(y_0-y_{m-1}) \end{bmatrix}.$$

Basically one solves this system as one solved for the D_i for an open curve. During forward elimination, however, it is necessary to compute and save nonzero values for entries in the rightmost column and to successively cancel the leftmost nonzero value in the bottom row. The analogous change must be made to the back substitution process as well.

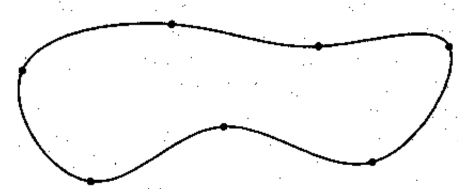


Figure 3.8. A closed interpolating cubic spline.