12. C^2-interpolating curves

Reading

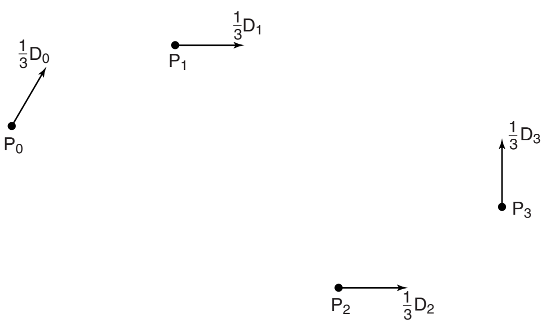
Optional

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

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:

$$
\text{* (-1/2) +}\curvearrowright\;
\begin{bmatrix}
2 & 1 & & & & \\
1 & 4 & 1 & & & \\
 & 1 & 4 & 1 & & \\
 & & & \ddots & & \\
 & & & 1 & 4 & 1 \\
 & & & & 1 & 2
\end{bmatrix}
\begin{bmatrix} D_0^T \\ D_1^T \\ D_2^T \\ \vdots \\ D_{m-1}^T \\ D_m^T \end{bmatrix}
=
\begin{bmatrix} E_0^T \\ E_1^T \\ E_2^T \\ \vdots \\ E_{m-1}^T \\ E_m^T \end{bmatrix}
\;\curvearrowleft\text{* (-1/2) +}
$$

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

Back subsitution

The resulting matrix is **upper diagonal**:

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

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

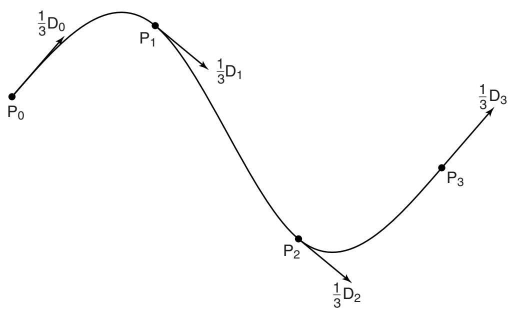
We can now solve for the unknowns by back substitution (where we can drop the transposes for the moment):

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

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?

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.