CIS 515

Fundamentals of Linear Algebra and Optimization
Jean Gallier and Jocelyn Quaintance

Project 5: Curve Interpolation

The purpose of this project is to solve a curve interpolation problem using cubic splines. This type of problem arises frequently in computer graphics and in robotics (path planning).

Recall from Project 1A that polynomial curves of degree $\leq m$ can be defined in terms of control points and the Bézier polynomials. Cubic Bézier curves are often used because they are cheap to implement and give more flexibility than quadratic Bézier curves.

A *cubic Bézier curve* $C(t)$ (in $\mathbb{R}^2$ or $\mathbb{R}^3$) is specified by a list of four *control points* (b_0, b_2, b_2, b_3) and is given parametrically by the equation

$$C(t) = (1-t)^3\, b_0 + 3(1-t)^2 t\, b_1 + 3(1-t)t^2\, b_2 + t^3\, b_3.$$

Clearly, $C(0) = b_0$, $C(1) = b_3$, and for $t \in [0,1]$, the point $C(t)$ belongs to the convex hull of the control points b_0, b_1, b_2, b_3.

Interpolation problems require finding curves passing through some given data points and possibly satisfying some extra constraints.

It is known that Lagrange interpolation is not very satisfactory when $N \geq 5$ since Lagrange interpolants tend to oscillate in an undesirable manner. Thus, we turn to Bézier spline curves.

A *Bézier spline curve* F is a curve which is made up of curve segments which are Bézier curves, say $C_1, \ldots, C_m$ $(m \geq 2)$. We will assume that F is defined on $[0, m]$, so that for $i = 1, \ldots, m$,

$$F(t) = C_i(t - i + 1), \quad i - 1 \leq t \leq i.$$

Typically, some smoothness is required between any two junction points, that is, between any two points $C_i(1)$ and $C_{i+1}(0)$, for $i = 1, \ldots, m-1$. We require that $C_i(1) = C_{i+1}(0)$ (C^0*-continuity*), and typically that the derivatives of C_i at 1 and of C_{i+1} at 0 agree up to second order derivatives. This is called C^2*-continuity*, and it ensures that the tangents agree as well as the curvatures.

There are a number of interpolation problems, and we consider one of the most common problems which can be stated as follows:

Problem: Given $N+1$ data points $x_0, \ldots, x_N$, find a C^2 cubic spline curve F, such that $F(i) = x_i$, for all i, $0 \leq i \leq N$ $(N \geq 2)$.

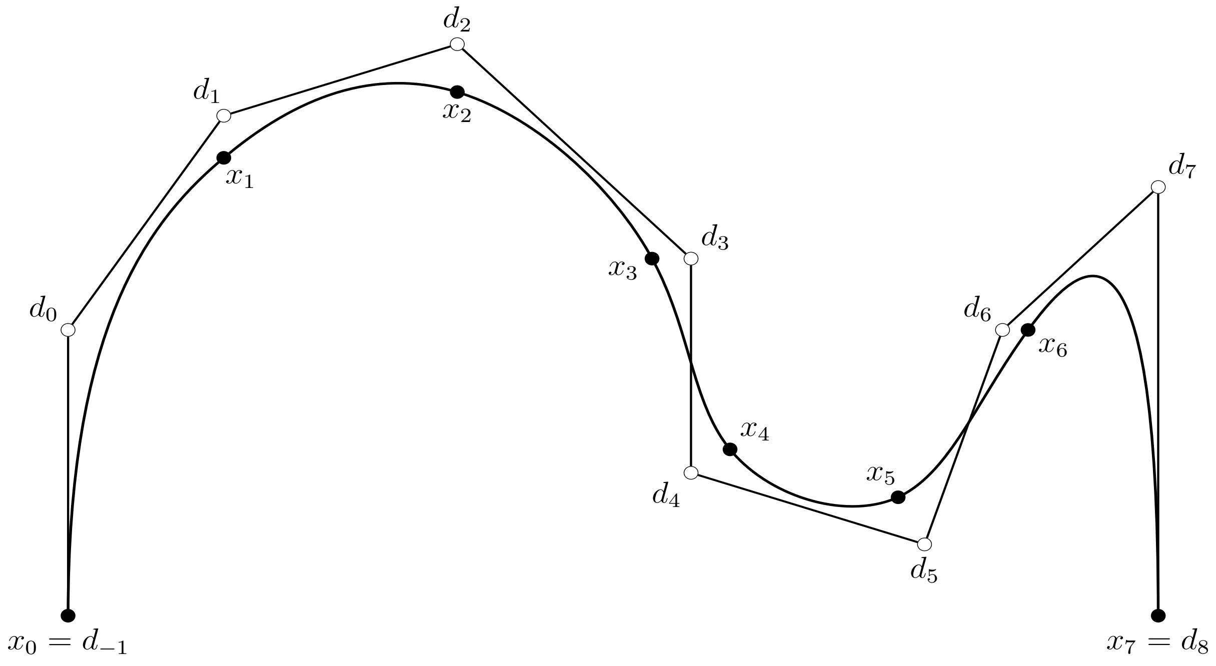

Figure 1: A C^2 cubic interpolation spline curve passing through the points $x_0, x_1,\ x_2, x_3,\ x_4, x_5,\ x_6, x_7$

A way to solve this problem is to find $N+3$ auxiliary points $d_{-1}, \ldots, d_{N+1}$ called *de Boor control points* from which N Bézier curves can be found. Actually,

$$d_{-1} = x_0 \quad \text{and} \quad d_{N+1} = x_N$$

so we only need to find $N+1$ points $d_0, \ldots, d_N$ (See figure 2).

It turns out that the C^2-continuity constraints on the N Bézier curves yield only $N-1$ equations, so d_0 and d_N can be chosen arbitrarily. In practice, d_0 and d_N are chosen according to various “end conditions,” such as prescribed velocities at x_0 and x_N. For the time being, we will assume that d_0 and d_N are given.

Figures 1 and 2 illustrates an interpolation problem involving $N+1 = 7+1 = 8$ data points. The control points d_0 and d_7 were chosen arbitrarily.

It can be shown that $d_1, \ldots, d_{N-1}$ are given by the linear system

$$\begin{pmatrix} \frac{7}{2} & 1 & & & \\ 1 & 4 & 1 & & 0 \\ & \ddots & \ddots & \ddots & \\ 0 & & 1 & 4 & 1 \\ & & & 1 & \frac{7}{2} \end{pmatrix} \begin{pmatrix} d_1 \\ d_2 \\ \vdots \\ d_{N-2} \\ d_{N-1} \end{pmatrix} = \begin{pmatrix} 6x_1 - \frac{3}{2}d_0 \\ 6x_2 \\ \vdots \\ 6x_{N-2} \\ 6x_{N-1} - \frac{3}{2}d_N \end{pmatrix}. \tag{1}$$

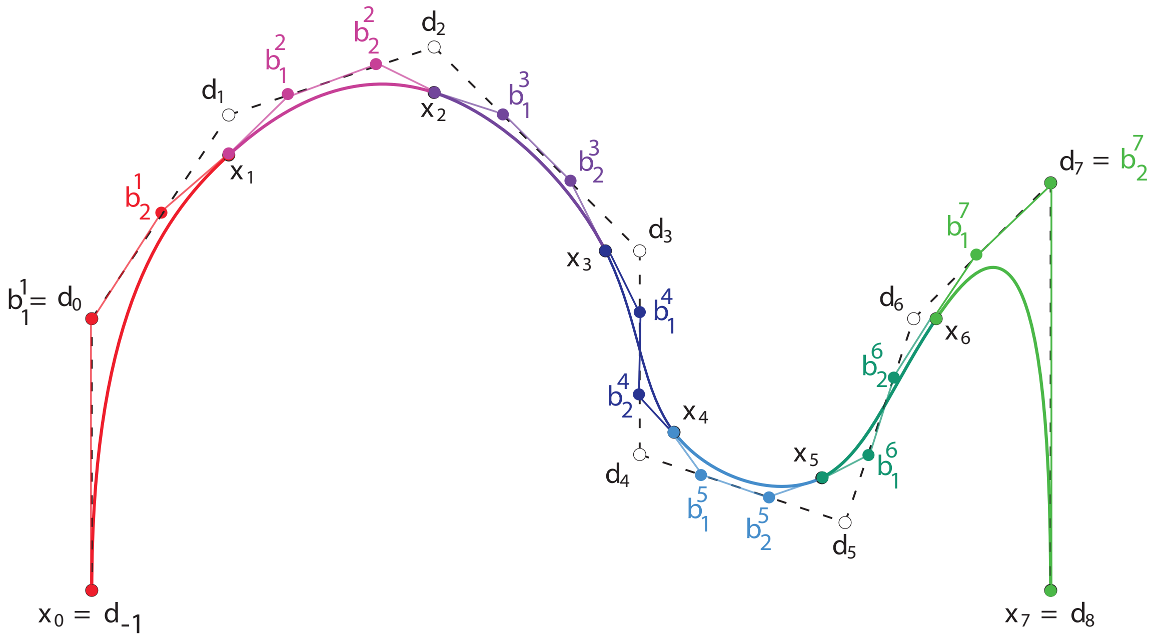

Figure 2: Figure 1 with color coding added and the Bezier control points plotted

The derivation of the above system assumes that $N \geq 4$. If $N = 3$, this system reduces to

$$\begin{pmatrix} \frac{7}{2} & 1 \\ 1 & \frac{7}{2} \end{pmatrix} \begin{pmatrix} d_1 \\ d_2 \end{pmatrix} = \begin{pmatrix} 6x_1 - \frac{3}{2}d_0 \\ 6x_2 - \frac{3}{2}d_3 \end{pmatrix} \tag{2}$$

When $N = 2$, it can be shown that d_1 is given by

$$d_1 = 2x_1 - \frac{1}{2}d_0 - \frac{1}{2}d_2. \tag{3}$$

Observe that when $N \geq 3$, the above matrix is strictly diagonally dominant, so it is invertible. Actually, the above system needs to be solved for the x-coordinates and for the y-coordinates of the d_is (and also for the z-coordinates, if the points are in $\mathbb{R}^3$). Once the above system is solved, the Bézier cubics $C_1, \ldots, C_N$ are determined as follows (we assume $N \geq 2$): For $2 \leq i \leq N - 1$, the control points $(b_0^i, b_1^i, b_2^i, b_3^i)$ of C_i are given by

$$\begin{aligned} b_0^i &= x_{i-1} \\ b_1^i &= \frac{2}{3}d_{i-1} + \frac{1}{3}d_i \\ b_2^i &= \frac{1}{3}d_{i-1} + \frac{2}{3}d_i \\ b_3^i &= x_i. \end{aligned} \tag{4}$$

The control points $(b_0^1, b_1^1, b_2^1, b_3^1)$ of C_1 are given by

$$\begin{aligned}
b_0^1 &= x_0 \\
b_1^1 &= d_0 \\
b_2^1 &= \frac{1}{2}d_0 + \frac{1}{2}d_1 \\
b_3^1 &= x_1,
\end{aligned} \tag{5}$$

and the control points $(b_0^N, b_1^N, b_2^N, b_3^N)$ of C_N are given by

$$\begin{aligned}
b_0^N &= x_{N-1} \\
b_1^N &= \frac{1}{2}d_{N-1} + \frac{1}{2}d_N \\
b_2^N &= d_N \\
b_3^N &= x_N.
\end{aligned} \tag{6}$$

See figure 2 for an illustration of these points.

It is easy to show that the tangent vectors m_0 at x_0 and m_N at x_N are given by

$$\begin{aligned}
m_0 &= 3(d_0 - x_0) \\
m_N &= 3(x_N - d_N).
\end{aligned} \tag{7}$$

End Conditions.

One method to determine the points d_0 and d_N is the *natural end condition*, which consists in setting the second derivatives at x_0 and at x_N to be zero, that is,

$$C_1''(0) = 0, \quad C_N''(1) = 0.$$

(Part 1) (**10 points**) Prove that the second derivative at b_0 of a Bézier cubic specified by the control points (b_0, b_1, b_2, b_3) is

$$6(b_0 - 2b_1 + b_2),$$

and the second derivative at b_3 is

$$6(b_1 - 2b_2 + b_3).$$

Prove that our system becomes

$$\begin{pmatrix}
4 & 1 & & & \\
1 & 4 & 1 & & 0 \\
 & \ddots & \ddots & \ddots & \\
0 & & 1 & 4 & 1 \\
 & & & 1 & 4
\end{pmatrix}
\begin{pmatrix} d_1 \\ d_2 \\ \vdots \\ d_{N-2} \\ d_{N-1} \end{pmatrix}
=
\begin{pmatrix} 6x_1 - x_0 \\ 6x_2 \\ \vdots \\ 6x_{N-2} \\ 6x_{N-1} - x_N \end{pmatrix},$$

where d_0, d_N are given by

$$\begin{aligned} d_0 &= \frac{2}{3}x_0 + \frac{1}{3}d_1 \\ d_N &= \frac{1}{3}d_{N-1} + \frac{2}{3}x_N. \end{aligned}$$

Note that d_0 is on the line segment (x_0, d_1) (1/3 of the way from x_0) and d_N is on the line segment (d_{N-1}, x_N) (1/3 of the way from x_N). For this proof, you may use Equation (1).

In the above derivation we assumed that $N \geq 4$. If $N = 3$, show that this system reduces to

$$\begin{pmatrix} 4 & 1 \\ 1 & 4 \end{pmatrix} \begin{pmatrix} d_1 \\ d_2 \end{pmatrix} = \begin{pmatrix} 6x_1 - x_0 \\ 6x_2 - x_3 \end{pmatrix}.$$

(2) After computing $d_1, \ldots, d_{N-1}$ by solving the linear system in (1) using the `Matlab` command \, use your program from Project 1B to compute the control points for the Bézier curves $C_1, \ldots, C_N$ and to plot these Bézier segments (for $t \in [0, 1]$) to visualize the interpolating spline.

(Part 2)(i) (**50 points**) Code a first version that takes as input two column vectors x and y of dimension $N + 1$ containing the x and y coordinates of the $N + 1$ data points $x_0, x_1, \ldots, x_N$ ($N \geq 2$). The ouput should be two column vectors dx and dy of dimension $N + 3$ consisting of the de Boor control points obtained by solving the linear system

$$\begin{pmatrix} 4 & 1 & & & \\ 1 & 4 & 1 & & 0 \\ & \ddots & \ddots & \ddots & \\ 0 & & 1 & 4 & 1 \\ & & & 1 & 4 \end{pmatrix} \begin{pmatrix} d_1 \\ d_2 \\ \vdots \\ d_{N-2} \\ d_{N-1} \end{pmatrix} = \begin{pmatrix} 6x_1 - x_0 \\ 6x_2 \\ \vdots \\ 6x_{N-2} \\ 6x_{N-1} - x_N \end{pmatrix},$$

with $d_{-1} = x_0$, $d_{N+1} = x_N$, and

$$\begin{aligned} d_0 &= \frac{2}{3}x_0 + \frac{1}{3}d_1 \\ d_N &= \frac{1}{3}d_{N-1} + \frac{2}{3}x_N. \end{aligned}$$

This should be a function of the form:

```
function [dx, dy] = interpnatxy(x,y)
```

(provided in project zip file).

(Part 2)(ii) (**20 points**) Compute the x and the y coordinates Bx and By of the N Bézier segments $C_1, \ldots, C_N$ computed by the method of Project 1B, as $N \times 4$ matrices. For visual correctness, also plot the curve. To achieve this, code the function `interpnatxy` modified from 2(i) to make use of the function `bspline2b` from Project 1B.

Run your program on the sequences of data points given by their x and y coordinates as the column vector x and y specified below:

$$\begin{aligned}
\texttt{x1} &= [3.6942; 1.3690; 2.9865; 5.8509; 8.1929; 8.2098; 6.8281]\\
\texttt{y1} &= [1.2144; 3.5925; 7.3933; 7.9217; 6.9665; 4.0396; 1.5600]\\
\texttt{x2} &= [3.9806; 2.2789; 3.6942; 6.8618; 7.1820]\\
\texttt{y2} &= [2.1087; 4.2429; 7.0884; 6.9461; 4.3852]\\
\texttt{x3} &= [4.2334; 1.0826; 1.3016; 4.9579; 8.2435; 4.8062]\\
\texttt{y3} &= [1.0315; 3.6941; 5.6250; 7.9624; 5.5640; 5.8486]\\
\texttt{x4} &= [4.6040; 2.2283; 3.3741; 2.1609; 7.2494; 6.8955; 9.1702]\\
\texttt{y4} &= [1.3364; 1.6616; 3.5722; 6.8242; 8.6535; 3.7957; 2.8608]\\
\texttt{x5} &= [2.7310; 1.3599; 1.1662; 1.9709; 4.7876; 7.0827; 6.2630;\\
&\quad 4.2809; 3.9232; 4.9367; 8.3048; 9.0052; 7.6639]\\
\texttt{y5} &= [1.1106; 2.4716; 5.3639; 7.3677; 8.3129; 7.6134; 5.3072;\\
&\quad 5.0614; 2.8875; 1.0539; 0.8648; 2.5095; 3.7760]
\end{aligned}$$

Running your program on the data points specified by $x5$ and $y5$ you should get the interpolating curve shown in Figure 3.

(Part 2)(iii) (**20 points**) Then code a second version that takes the data points $x_0, \ldots, x_N$ as sceen input using the function `getpoints` of Project 1B. This version does not need to output dx, dy, Bx and By, but should plot the interpolating curve.

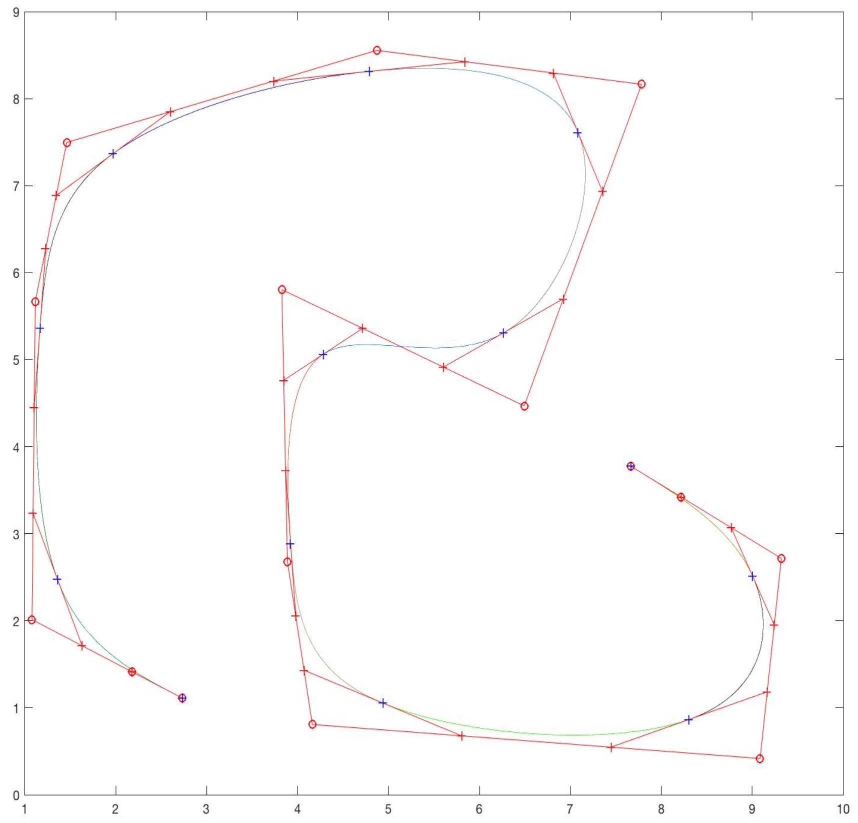

Figure 3: The interpolating curve specified by $x5$ and $y5$.