Fall 2016 CIS515

Fundamentals of Linear Algebra and Optimization
Jean Gallier

Project 1

September 15, 2016; Due September 29, 2016

P1 (120 points). The purpose of this project is to design a `Matlab` program to plot a cubic Bézier spline curve given by a sequence of de Boor control points.

Recall that a *cubic Bézier spline* $F(t)$ (in $\mathbb{R}^2$ or $\mathbb{R}^3$) is specified by a list of *de Boor control points* $(d_0, d_1, \ldots, d_N)$, with $N \geq 7$, and consists of $N-2$ Bézier cubic segments $C_1, \ldots, C_{N-2}$, such that if the control points of C_i are $(b_0^i, b_1^i, b_2^i, b_3^i)$, then they are determined by the following equations:

For C_1, we have

$$\begin{aligned}
b_0^1 &= d_0 \\
b_1^1 &= d_1 \\
b_2^1 &= \frac{1}{2}d_1 + \frac{1}{2}d_2 \\
b_3^1 &= \frac{1}{2}b_2^1 + \frac{1}{2}b_1^2 = \frac{1}{4}d_1 + \frac{7}{12}d_2 + \frac{1}{6}d_3.
\end{aligned}$$

The curve segment C_2 is given by

$$\begin{aligned}
b_0^2 &= \frac{1}{2}b_2^1 + \frac{1}{2}b_1^2 = \frac{1}{4}d_1 + \frac{7}{12}d_2 + \frac{1}{6}d_3 \\
b_1^2 &= \frac{2}{3}d_2 + \frac{1}{3}d_3 \\
b_2^2 &= \frac{1}{3}d_2 + \frac{2}{3}d_3 \\
b_3^2 &= \frac{1}{2}b_2^2 + \frac{1}{2}b_1^3 = \frac{1}{6}d_2 + \frac{4}{6}d_3 + \frac{1}{6}d_4.
\end{aligned}$$

For $i = 3, \ldots, N-4$, the curve segment C_i is specified by the "one third two third rule:"

$$
\begin{aligned}
b_0^i &= \frac{1}{2}b_2^{i-1} + \frac{1}{2}b_1^i = \frac{1}{6}d_{i-1} + \frac{4}{6}d_i + \frac{1}{6}d_{i+1} \\
b_1^i &= \frac{2}{3}d_i + \frac{1}{3}d_{i+1} \\
b_2^i &= \frac{1}{3}d_i + \frac{2}{3}d_{i+1} \\
b_3^i &= \frac{1}{2}b_2^i + \frac{1}{2}b_1^{i+1} = \frac{1}{6}d_i + \frac{4}{6}d_{i+1} + \frac{1}{6}d_{i+2}.
\end{aligned}
$$

This generic case is illustrated in Figure 1.

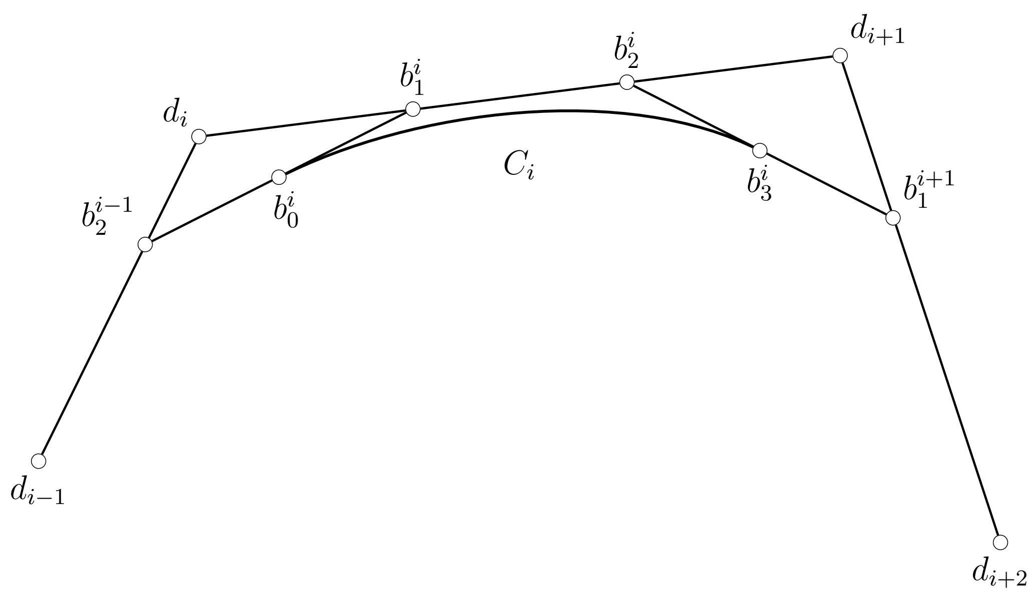

Figure 1: Computing Bézier control points from de Boor control points

The curve segment C_{N-3} is given by

$$
\begin{aligned}
b_0^{N-3} &= \frac{1}{2}b_2^{N-4} + \frac{1}{2}b_1^{N-3} = \frac{1}{6}d_{N-4} + \frac{4}{6}d_{N-3} + \frac{1}{6}d_{N-2} \\
b_1^{N-3} &= \frac{2}{3}d_{N-3} + \frac{1}{3}d_{N-2} \\
b_2^{N-3} &= \frac{1}{3}d_{N-3} + \frac{2}{3}d_{N-2} \\
b_3^{N-3} &= \frac{1}{2}b_2^{N-3} + \frac{1}{2}b_1^{N-2} = \frac{1}{6}d_{N-3} + \frac{7}{12}d_{N-2} + \frac{1}{4}d_{N-1}.
\end{aligned}
$$

Finally, C_{N-2} is specified by

$$
\begin{aligned}
b_0^{N-2} &= \frac{1}{2}b_2^{N-3} + \frac{1}{2}b_1^{N-2} = \frac{1}{6}d_{N-3} + \frac{7}{12}d_{N-2} + \frac{1}{4}d_{N-1}\\
b_1^{N-2} &= \frac{1}{2}d_{N-2} + \frac{1}{2}d_{N-1}\\
b_2^{N-2} &= d_{N-1}\\
b_3^{N-2} &= d_N
\end{aligned}
$$

Observe that

$$b_0^{i+1} = b_3^i, \quad 1 \leq i \leq N-3.$$

Using the above equation, the cases $N = 5, 6$ are easily adapted from the general case: compute the control points for C_1, $C_2, \ldots, C_{N-4}, C_{N-2}$, and C_{N-3}. When $N = 4$, use the formulae for C_1 and $C_{N-2} = C_2$ with

$$b_3^1 = b_0^2 = \frac{1}{4}b_1 + \frac{1}{2}b_2 + \frac{1}{4}b_3.$$

(1) Implement a `Matlab` program to plot a cubic spline specified by a sequence of de Boor control points (for $N \geq 5$). Make use of a function that allows you to specify the control points by clicking on the mouse (screen input).

(2) Referring back to the interpolation problem defined in the notes (and the slides), can you explain why the linear system giving $d_1, \ldots, d_{N-1}$ in terms of the data points $x_0, \ldots, x_N$ and d_0 and d_N ($N \geq 4$) is:

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

Beware that the N in the interpolation problem is *not* the same N as in (1)!

P2 (80 points). The purpose of this project is to implement the subdivision version of the de Casteljau algorithm for approximating a Bézier curve by a polygonal line.

Given a cubic Bézier curve C specified by its control points (b_0, b_1, b_2, b_3), for any t, the de Casteljau algorithm constructs points

$$
\begin{aligned}
&b_0^1, b_1^1, b_2^1\\
&b_0^2, b_1^2\\
&b_0^3,
\end{aligned}
$$

using the equations

$$
\begin{aligned}
b_i^1 &= (1-t)b_i + tb_{i+1} && i = 0, 1, 2 \\
b_i^2 &= (1-t)b_i^1 + tb_{i+1}^1 && i = 0, 1 \\
b_i^3 &= (1-t)b_0^2 + tb_1^2 && i = 0.
\end{aligned}
$$

This process is conveniently depicted as follows:

$$
\begin{array}{cccc}
0 & 1 & 2 & 3 \\
b_0 = b_0^0 & & & \\
 & b_0^1 & & \\
b_1 = b_1^0 & & b_{0,}^2 & \\
 & b_1^1 & & b_0^3 \\
b_2 = b_2^0 & & b_1^2 & \\
 & b_2^1 & & \\
b_3 = b_3^0 & & &
\end{array}
$$

Then, the point $C(t)$ is given by

$$C(t) = b_0^3.$$

The cubic curve is tangent to the line segment (b_0^2, b_1^2) at b_0^3; see Figure 2.

It turns out that the two sequences of points

$$ud = (b_0, b_0^1, b_0^2, b_0^3)$$

and

$$ld = (b_0^3, b_1^2, b_2^1, b_3)$$

are also control points for the curve C; see Figure 2.

Thus, we can iterate the above method using the control points in ud and ld, to obtain a sequence of four control polygons, and if we iterate this process n times, we obtain 2^n control polygons which linked together yield a polygonal line that approximates very closely the segment of Bézier curve $C(t)$ for $t \in [0, 1]$. Usually, we perform subdivision for $t = 1/2$. This method is called the *subdivision version of the de Casteljau algorithm*.

(1) Implement the subdivision version of the de Casteljau algorithm in `Matlab`, for a cubic specified by its control points (b_0, b_1, b_2, b_3). Your program should take as input the control polygon (b_0, b_1, b_2, b_3) and the number of times n that your program subdivides.

Try various control polygons, and for each one, show of the result of subdividing at least (six times $n = 1, 2, \ldots, 6$).

Make use of a function that allows you to specify the control points by clicking on the mouse (screen input).

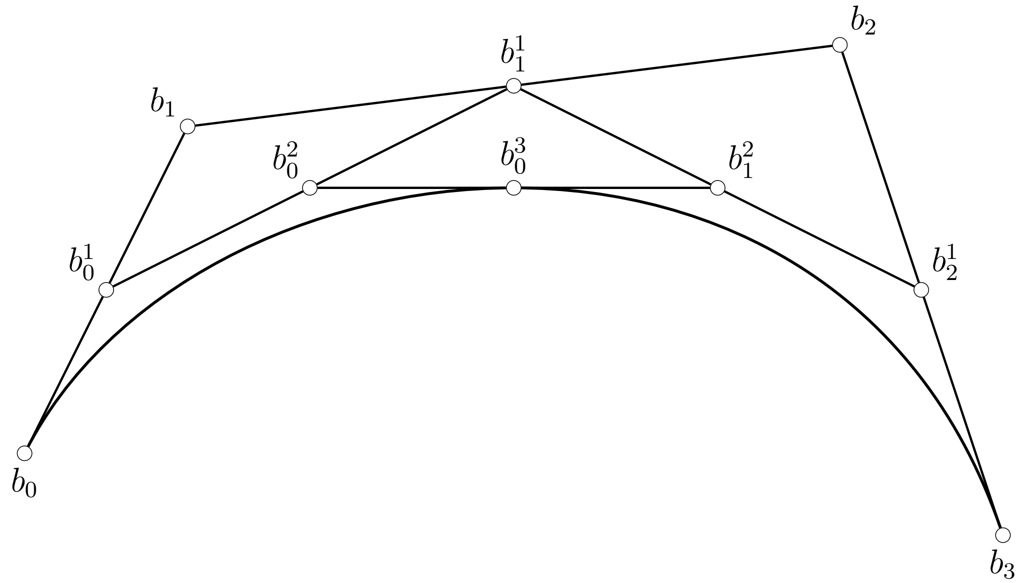

Figure 2: de Casteljau subdivision

(2) Given a Bézier curve C of degree m specified by its control points $(b_0, b_1, \ldots, b_m)$, for any t, the de Casteljau algorithm constructs points b_i^k in m stages

$$\begin{array}{l} b_0^1, b_1^1, \ldots, b_{m-2}^1, b_{m-1}^1 \\ b_0^2, b_1^2, \ldots, b_{m-2}^2 \\ \vdots \\ b_0^{m-1}, b_1^{m-1} \\ b_0^m. \end{array}$$

If we write $b_i^0 = b_i$ for $i = 0, \ldots, m$, then the b_i^k are given by the following equations:

$$b_i^{k+1} = (1-t)b_i^k + tb_{i+1}^k \quad k = 0, \ldots, m-1,\ i = 0, \ldots, m-k-1,$$

and as in the case $m = 3$, the point on the curve is

$$C(t) = b_0^m.$$

As in the case of cubic curves, the two sequences of points

$$ud = (b_0, b_0^1, \ldots, b_0^{m-1}, b_0^m)$$

and

$$ld = (b_0^m, b_1^{m-1}, \ldots, b_{m-1}^1, b_m)$$

are also control points for the curve C, so we can iterate the above method using the control points in ud and ld, and we obtain a subdivision method that yields a polygonal line that approximates very closely the segment of Bézier curve for $t \in [0, 1]$.

Implement the subdivision version of the de Casteljau algorithm in `Matlab`, for a Bézier cutrve of degree m specified by its control points $(b_0, b_1, \ldots, b_m)$. Your program should take as input the control polygon $(b_0, b_1, \ldots, b_m)$, and the number of times n that your program subdivides.

Try various control polygons, and for each one, show of the result of subdividing at least six times ($n = 1, 2, \ldots, 6$).

Make use of a function that allows you to specify the control points by clicking on the mouse (screen input).

TOTAL: 200 points.