MCIT 515

Fundamentals of Linear Algebra and Optimization
Jean Gallier and Jocelyn Quaintance

Project 1B: Drawing Cubic Bézier Spline Curves

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:"

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

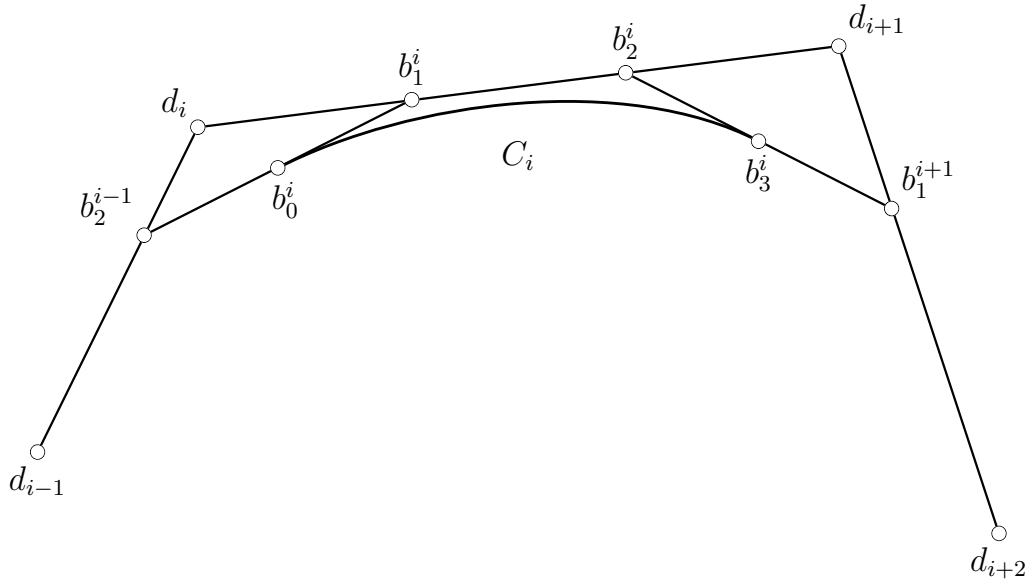

Figure 1: Computing Bézier control points from de Boor control points

This generic case is illustrated in Figure 1.

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

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

Using the above equations, 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

$$b_3^1 = b_0^2 = \frac{1}{4}d_1 + \frac{1}{2}d_2 + \frac{1}{4}d_3.$$

(Part 1) (**50 points**) Implement a `Matlab` program to compute the Bézier control points of the Bézier segments $C_1, \ldots, C_{N-2}$ constituting the spline specified by a sequence of de Boor control points $d_0, d_1, \ldots, d_N$ (for $N \geq 4$). The input to your program should be two column vectors dx and dy of dimension $N + 1$ consisting of the x-coordinates and the y-coordinates of the $N + 1$ de Boor control points $d_0, d_1, \ldots, d_N$. The output should be two $(N - 2) \times 4$ matrices Bx and By where $Bx(i, :)$ consists of the x-coordinates the control points of the Bézier segment C_i and $By(i, :)$ consists of the y-coordinates the control points of the Bézier segment C_i $(i = 1, \ldots, N - 2)$. For example, on input

```
dx = [4.2173; 1.5849; 2.1301; 4.7625; 7.7531; 8.3606; 5.4322]
dy = [1.8424; 3.2603; 6.0028; 7.6446; 6.3013; 2.5886; 4.0065]
```

(seven de Boor control points, $N = 6$) we get

```
Bx =
    4.2173    1.5849    1.8575    2.4325
    2.4325    3.0075    3.8850    4.8222
    4.8222    5.7593    6.7562    7.4065
    7.4065    8.0569    8.3606    5.4322
By =
    1.8424    3.2603    4.6315    5.5908
    5.5908    6.5501    7.0973    7.1471
    7.1471    7.1968    6.7491    5.5970
    5.5970    4.4450    2.5886    4.0065
```

See Figures 2 and 3.

Use the function `show_bspline2b` (shown in figure 5), in which you need to supply the function `bspline2b` (shown in figure 6 and also in project zip file).

The function `bspline2b` computes the $(N - 2) \times 4$ matrices Bx and By specified above.

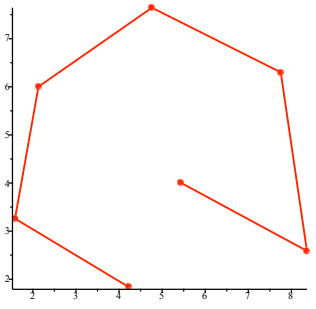

Figure 1

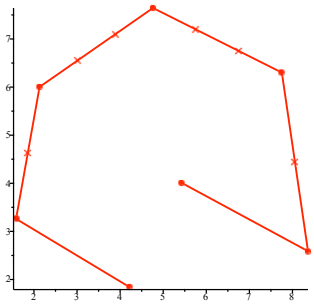

Figure 2

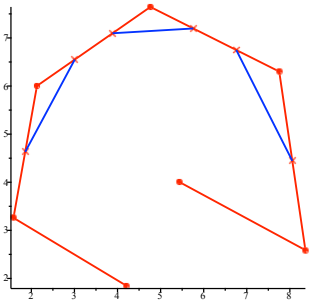

Figure 3

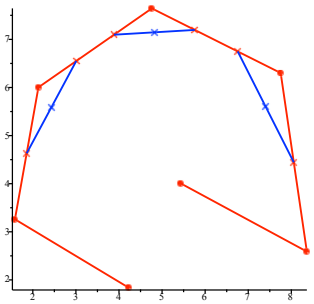

Figure 4

Figure 2: Figure 1 is the polygonal curve determined by the seven de Boor control points of dx and dy. Figure 2 applies the "one third two third rule" and plots the following six points from Bx and By: $[1.8575, 4.6315]$, $[3.0075, 6.5501]$, $[3.8850, 7.0973]$, $[5.7593, 7.1968]$, $[6.7562, 6.7491]$, $[8.0569, 4.4450]$. Figure 3 connects the adjacent points found in Step 2 via blue line segments. Step 4 computes the midpoint of these blue line segment to get the remaining three points of Bx and By with coordinates $[2.4325, 5.5908]$, $[4.8222, 7.1471]$, $[7.4065, 5.5970]$.

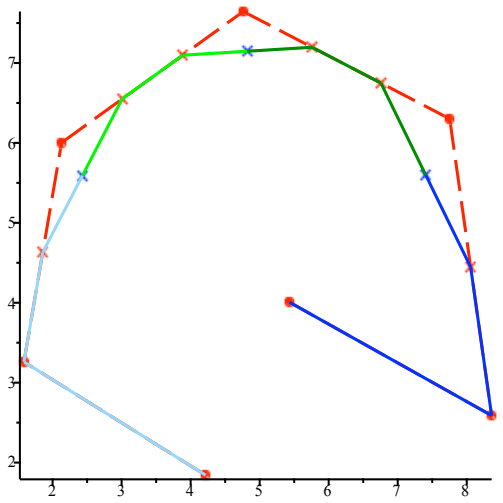

Figure 3: A continuation of the example illustrated by Figure 2 in which we color code how the points of Bx and By form four Bézier control polygons.

In the output script the following control points will be tested:

$$\begin{aligned}
\texttt{dx1} &= [3.6942; 1.3690; 2.9865; 5.8509; 8.1929; 8.2098; 6.8281] \\
\texttt{dy1} &= [1.2144; 3.5925; 7.3933; 7.9217; 6.9665; 4.0396; 1.5600] \\
\texttt{dx2} &= [3.9806; 2.2789; 3.6942; 6.8618; 7.1820] \\
\texttt{dy2} &= [2.1087; 4.2429; 7.0884; 6.9461; 4.3852] \\
\texttt{dx3} &= [4.2334; 1.0826; 1.3016; 4.9579; 8.2435; 4.8062] \\
\texttt{dy3} &= [1.0315; 3.6941; 5.6250; 7.9624; 5.5640; 5.8486] \\
\texttt{dx4} &= [4.6040; 2.2283; 3.3741; 2.1609; 7.2494; 6.8955; 9.1702] \\
\texttt{dy4} &= [1.3364; 1.6616; 3.5722; 6.8242; 8.6535; 3.7957; 2.8608] \\
\texttt{dx5} &= [5.4297; 5.2275; 2.9865; 1.4532; 2.1778; 3.2898; 6.8113; \\
&\quad 9.0691; 7.2999; 7.2157; 9.2713; 7.4853; 6.4575] \\
\texttt{dy5} &= [4.6494; 1.9055; 1.7429; 3.8974; 8.0030; 6.7429; 9.1209 \\
&\quad 7.2917; 6.4380; 2.8404; 2.7795; 0.9502; 1.3974]
\end{aligned}$$

(These are also in written the output script).

(Part 2) (**30 points**) Use your program for drawing a Bézier curve (see Project 1A) to display the Bézier segments $C_1, \ldots, C_{N-2}$ computed in Part (1) subdivided 6 times. To achieve this, modify the function `bspline2b` (see figure 6) so that it calls a function `drawbezier_dc` to display each spline segment C_i using the subdivision version of the de Casteljau algorithm from Project 1A. (The code is in the project zip file as well)

For example, running the de Casteljau subdivision algorithm (with $nn = 6$) on each of the four Bézier segments specified by

```
Bx =
    4.2173    1.5849    1.8575    2.4325
    2.4325    3.0075    3.8850    4.8222
    4.8222    5.7593    6.7562    7.4065
    7.4065    8.0569    8.3606    5.4322
By =
    1.8424    3.2603    4.6315    5.5908
    5.5908    6.5501    7.0973    7.1471
    7.1471    7.1968    6.7491    5.5970
    5.5970    4.4450    2.5886    4.0065
```

you get the spline shown in Figure 4.

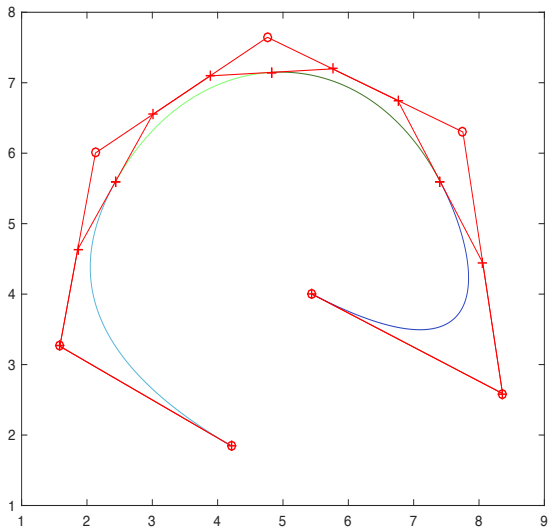

Figure 4: A continuation of Figure 3 in which we illustrate the cubic spline specified by its Boor control points. The coloring coding in the spline curve reflects the color coding of the four Bézier control polygons of Figure 3.

(Part 3) (**20 points**) Finally make use of the function `getpoints` to specify the control points by clicking on the mouse (screen input). This is done when you run the output script, make sure the images are saved properly.

```
function [Bx, By] = show_bspline2b2(dx,dy)
%
% This is an auxilary function designed to output Bx and By
% for the version that uses dx and dy as input instead of
%
%  drawb = 1, shows Bezier control polygons
%
% nn = subdivision level for de Casteljau
nn = 6;
N = size(dx,1)-1;
fprintf('N = %d \n', N)
drawb = 1;
[Bx, By] = bspline2b(dx,dy,N);
hold off
end
```

Figure 5: Code for function `show_bspline2b2`

```
function [Bx, By] = bspline2b(dx,dy,N,nn,drawb)
% Works if N >= 4.

%These will hold the four control points for each segment
Bx = zeros(N-2,4);
By = zeros(N-2,4);

%
% Code to compute the Bezier control points of the
% cuve segments C_1, ..., C_{N-2} from Part (1).
%

dim_data = 2;
B = zeros(dim_data,4);
plot(dx,dy,'or-');   % plots d's as red circles
hold on;
for i = 1:N-2
    B(1,:) = Bx(i,:); B(2,:) = By(i,:);
    drawbezier_dc(B,nn,drawb);
end
% hold off;
end
```

Figure 6: Code for function (bspline2b)