DigiPen Institute of Technology, Bilbao

MAT300 Curves & Surfaces
Spring, 2020
Project 3. Deadline: 1/4/2020

---

1 Main description of the project

The project consists of coding routines to obtain Bezier curves in 2D and 3D.

Given $n+1$ control points P_0, P_1, ..., $P_n \in \mathbb{R}^2$, a planar Bezier curve is a polynomial map $\gamma : [0,1] \to \mathbb{R}^2$ defined as

$$\gamma(t) = \sum_{i=0}^{n} P_i B_i^n(t) \tag{1}$$

where B_i^n is the ith Bernstein polynomial of degree n defined as

$$B_i^n(t) = \begin{pmatrix} n \\ i \end{pmatrix} (1-t)^{n-i} t^i \tag{2}$$

for $i = 0, 1, \ldots, n$. If the points P_0, P_1, ..., $P_n \in \mathbb{R}^3$ the graph of the curve will be in 3D and the map given by (1) will be $\gamma : [0,1] \to \mathbb{R}^3$.

The idea is to select $n+1$ points in $\mathbb{R}^2$ or $\mathbb{R}^3$, then define a mesh of nodes over the interval $[0,1]$, apply a method for evaluating γ at that mesh of nodes, and plot the result. The different methods that you have to implement are the direct evaluation, the De Casteljau algorithm, and the midpoint subdivision.

1.1 Input data

The input data will be the following:

- Control points P_0, P_1, ..., P_n.
- Dimension (2 or 3).
- Desired method for Bezier evaluation.
- Number of nodes for the output mesh in the case of De Casteljau algorithm and direct evaluation.
- Index for the node for which shells have to be computed in the case of De Casteljau algorithm.
- Number of iterations in the case of midpoint subdivision.

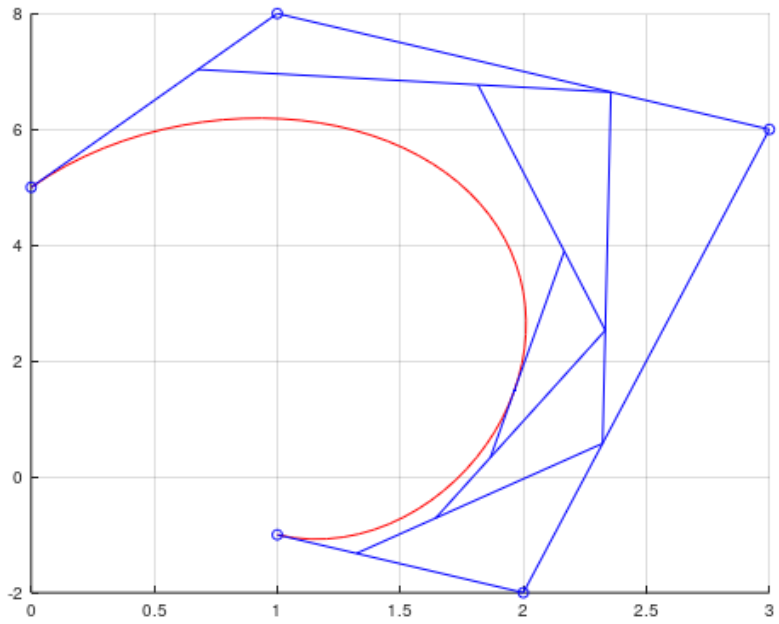
The data will be introduced in a script file like this:

data1.m

```
%-------------------------------------------------------------
% MAT300 Curves and surfaces
% Julia Sánchez Sanz
% julia.sanchez@digipen.edu
% 18/6/2019
%
% Script input data project 3
%-------------------------------------------------------------

PX=[1 2 3 1 0];  % x-coordinate Bezier control points
PY=[-1 -2 6 8 5]; % y-coordinate Bezier control points
%PZ=[2 1 -1 4 2]; % z-coordinate Bezier control points (un-comment)

Dimension=2; % dimension 2 or 3 (consistent with PZ)

methoddigit=1; % 0-direct evaluation, 1-De Casteljau, 2-midpoint subdivision

outputnodes=60; % number of nodes for the output mesh if methoddigit=0,1

shellindex=20; % node for which compute shells if methoddigit=1

miditerations=6; % number of iterations if methoddigit=2
```

1.2 Expected output

When the user runs the program, a figure should appear with the graph of the Bezier curve and the polygonal line linking the control points. Moreover, if the selected method is the De Casteljau algorithms, the shells for a selected point should appear as well. The plot will be a 2D or a 3D plot depending on the input data.



1.3 Main function

The main function of the project will be **beziercurve.m**

The function will check which method for obtaining the Bezier curve is the one to use, and will call the functions **directevaluation.m**, **decasteljau.m**, or **midsubdivision.m** for representing the curve.

1.4 The meshes of nodes

In the case of selecting the De Casteljau algorithm or the direct evaluation, the mesh will be regular in $[0, 1]$ with the number of output nodes introduced in the input file.

In the case of the midpoint subdivision, the number of iterations determine the number of output points, as we saw in class.

1.5 Methods for Bezier evaluation

1.5.1 Direct evaluation

The scheme for the direct evaluation method is the following:

- Construct a regular mesh of t values in $[0, 1]$.
- Evaluate the $n + 1$ Bernstein polynomials

$$B_i^n(t) = \begin{pmatrix} n \\ i \end{pmatrix} (1-t)^{n-i} t^i$$

  at the values of the mesh.

- Evaluate the Bezier curve

$$\gamma(t) = \sum_{i=0}^{n} P_i B_i^n(t)$$

  at the values of the mesh.

- Plot the result.

1.5.2 The De Casteljau algorithm

The scheme for the De Casteljau algorithm is the following:

- Construct a regular mesh of t values in $[0, 1]$.
- Apply recursively linear interpolation to obtain $\gamma(t_j)$ at the values of the mesh:
  - Start with the initial control point $P_i^0 = P_i$ for $i = 0, 1, \ldots, n$.

  - for $k = 1 : n$ and for $i = 0 : n - k$ the recursion is given by

$$P_i^k(t_j) = t_j P_{i+1}^{k-1}(t_j) + (1 - t_j) P_i^{k-1}(t_j) \tag{3}$$

  - $\gamma(t_j) = P_0^n$

- Store the values in an array and plot the result.

For obtaining the shells for a particular point the idea is to save the intermediate control points at each stage and plot the resulting polygonal lines, i.e.

- $P_0 - P_1 - \ldots - P_n$ is the first line,
- $P_0^1 - P_1^1 - \ldots - P_{n-1}^1$ is the second line,
- $P_0^2 - P_1^2 - \ldots - P_{n-2}^2$ is the third line and so on.

1.5.3 Midpoint subdivision

The scheme for the midpoint subdivision algorithm is the following:

- Knowing that we will apply k iterations, construct an array Γ for the outputs with length $2^k + 1$.
- for $j = 1 : k$ do:
  - for $m = 1 : 2^{j-1}$ do: (for each curve do midpoint subdivision)
    * apply recursively linear interpolation to obtain $\gamma_{(j,m)}(\frac{1}{2})$
    * take the above intermediate control points for the curve $\gamma_{(j+1,2m-1)}$ and the below control points for the curve $\gamma_{(j+1,2m)}$

  Take the initial, midpoint and final points of each curve for the output as follows:

- $\Gamma(1) = \gamma_{(k,1)}(0)$
- for $m = 1 : 2^{k-1}$ do:
  - $\Gamma(2m) = \gamma_{(k,m)}(\frac{1}{2})$
  - $\Gamma(2m + 1) = \gamma_{(k,m)}(1)$
- plot the result.

1.6 Graphs

In Octave 2D plots are computed with the command plot. 3D plots are obtained with plot3, sometimes they appear planar plots that you have to rotate.

2 Submission instructions

The programming projects will be done in groups of three people (and one of two because you are eleven students in class).

Please, submit all project parts on the Moodle page for MAT300 in a zip. The name of the zip will be

MAT300_ project3_ "surnames of all members of group"

You should include in the zip:

- Octave codes for each part of the project.
- A **readme.text** file specifying which programs the zip contains, in which program is each task of the project, the authors and instructions for running the code.
- A **explanations.pdf** file with the mathematical background. This file will contain at least 4 and at most 7 pages. In this pdf you should present:
  - **the description of the problem:** what is the mathematical problem that the code solves (one sentence or two), what are the inputs and the expected outputs.
  - **the explanation of the numerical methods that you use to tackle the problem:** here you have to present the theory of every method you use, step by step, using the proper terminology, a correct notation and understandable mathematical formulation (you can use latex). The explanations should be carried out for the general case, and not for particular examples. You can use the slides, the books of reference in the syllabus, or internet for information, but do not forget to include the references in the bibliography. If you look at Wikipedia as a source of information, be sure you understand the material, notation and formulation before doing copy-paste, and be sure that material is consistent with the contents of the course. The notation has to be consistent along the whole document, and if possible the same as the used in class. Please, use a common form along the document (same font, same size and so on).
  - **the relation between the above methods and your code:** you should link the above descriptions to the files and lines of code (what the code does and in which order, what is done-where is done).
  - **examples showing that your code indeed works:** you should compare your obtained results with analytical solutions. It worth to compute for a short number of nodes Bezier curves by hand and evaluate them in some points and compare with the computations.

  - **observations:** behavior of curves and shells, problems of computations if any, time of computation, comparison among different methods, and other additional information that you may consider relevant.

  - **bibliography:** include the references that you used as literature.

3 Grading policy

The total grade of this programming assignment will be computed as follows:

- Codes 60%:
  - Direct evaluation 8%
  - De Casteljau 11%
  - Shells for De Casteljau 4%
  - Midpoint subdivision 12%
  - Meshes are correct 5%
  - Outputs in 2D and 3D are correct 5%
  - Codes are structured and clean. They implement Octave matrix and vector algebra 5%
  - Codes are well commented with headers and mathematical explanations with correct terminology in each line 10%
- Readme 4%
- Document 36%:
  - Description of the problem 3%
  - Mathematical explanation of the numerical methods that you use to tackle the problem: 20%
  - The relation between the above methods and your code: 5%
  - Examples: 3%
  - Observations: 5%

Concerning late submissions I will restrict to the late policy included in the syllabus.