Norwegian University of Science
and Technology
Department of Mathematical
Sciences

TMA4215 Numerical mathematics Autumn 2013

Project 2

Last update: November 4, 2013

Instructions

This project counts for 20% of the final grade in the course.

Deadline: November 18, 16:00.

Group size: ≤ 2.

To be handed in (no more, no less)

- A report of maximum 6 pages, using the given LaTeX-template (or as similar as possible if another word-processing system is used). The report should be submitted as a pdf-file!
- One (and only one) executable, well structured and well documented, self-contained MATLAB file.

The MATLAB-file and the pdf-file should be sent to Eirik within the deadline.

Use the **student number** (no name, no candidate number) in the report, and make sure that we can identify your student numbers from the MATLAB-file.

Failure to meet the instruction above may cause the project to be dismissed!

Objective of the project

Develop, implement, test and document an algorithm for fitting shapes defined by a discrete set of data points with a paramtric piecewise cubic polynomial curve.

Some comments and advices

- In this project, you will use Bezier curves (in particular the cubic ones) and you will have to solve least square problems. These topics are part the curriculum, but has not been lectured. A very brief description of Bezier curves is provided, but you are supposed to find out more by using whatever is available of resources (web, the library, etc). Notice that NTNU has electronic access to a lot of books.
- In the evaluation, quite some emphasis will be given to the presentation. If it turns out that you are not able to fulfill the project completely, or your final program do not work, then focus on what you have done. In this case, the MATLAB file you enclose may be one you have used for one of the subtasks. But the same rule apply, it should be self-contained and well documented.

  We advice you to stop doing new stuff on Friday 15th, and spend the weekend and Monday on fine-tuning the report and the (one and only) Matlab-file to be handed in.
- With a well-documented and self-contained MATLAB file we mean that the file
  - includes sufficient information in the initial comment lines to make it clear for the user what the program do, and how it should be used. This information should be available by writing `help filename`.
  - executess and provide the expected results without any problem.
- The tasks described below are there to help you to learn and master the material. They should be done, but the results should in general not be included in the report as **Task 1**, **Task 2**, etc. But you may want to include some of them as examples, or to justify that the described procedure work.

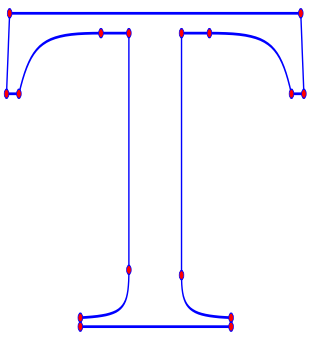
The LaTeX template, including some instruction on how to write the report, as well as some data files for for the tasks, will be provided at the project webpage.

1 Bezier curves and Bernstein polynomials

Bezier curves are used in computer graphics to construct smooth curves. The idea is to use parametrized functions, e.g.

$$\mathbf{S}(t) = \Big(x(t), y(t)\Big), \qquad 0 \le t \le 1$$

and to construct pictures by gluing several of these together. The letter T to the left is constructed by 16 Bezier curves, of which some are just straight lines.



Bezier curves are constructed from the Bernstein polynomials, given by

$$B_i^n(t) = \binom{n}{i} t^i (1-t)^{(n-i)}, \qquad i = 0, 1, \ldots, n, \qquad t \in [0, 1].$$

A planar Bezier curve of degree n is a parametrized curve in $\mathbb{R}^2$, defined by

$$\mathbf{S}(t) = \sum_{i=0}^{n} \mathbf{P}_i B_i^n(t), \qquad 0 \le t \le 1$$

where $\mathbf{P}_i \in \mathbb{R}^2$, $i = 0, 1, \ldots n$ are called *control points*. Notice that $\mathbf{S}(0) = \mathbf{P}_0$ and $\mathbf{S}(1) = \mathbf{P}_n$ are interpolation points, the others are not. In this project, we only consider *cubic* Bezier curves, thus $n = 3$, and for convenience, we will ignore the superscript. So unless otherwise stated, $B_i(t)$, $i = 0, 1, 2, 3$ refer to the cubic Bernstein polynomials.

There is plenty of information on Bezier curves on the web, see for instance the Wikipedia page.

Task 1 *Write down the 4 cubic Bernstein polynomials.*

Choose 4 control points, for instance

$$\mathbf{P}_0 = (0, 0), \quad \mathbf{P}_1 = (1, 2), \quad \mathbf{P}_2 = (2, -1), \quad \mathbf{P}_3 = (1, 0).$$

and write up and plot the corresponding Bezier curve. It may be useful to plot the control points and the straight line between them as well. Play around with other points and see what happens.

Notice that the straight line between the first two control points is tangential to the curve in P_0, similar is the straight line between the last two points tangential to the curve in P_3. Prove that this is true. What impact does it have if you want to glue several Bezier curves into one smooth curve?

Task 2 *Write a Matlab program which read a set of control points from a file, and draws the represented figure. Test the program on the test files on the webpage. One of them should give you the T.*

Each row in the datafile contains the control points of one Bezier curve on the format

$$P_{0,x} \;\; P_{0,y} \;\; P_{1,x} \;\; P_{1,y} \;\; P_{2,x} \;\; P_{2,y} \;\; P_{3,x} \;\; P_{3,y}$$

2 Curve fitting.

Given a curve γ in $\mathbf{R^2}$, how to approximate this by one Bezier curve? Or more specific, how to find the control points?

2.1 With a given, discrete parametrization

Assume we have given some points $\mathbf{x}_i$, $i = 1, \ldots, m$ on the curve γ, with $\mathbf{x}_1$ and $\mathbf{x}_m$ as the endpoints. Let $\mathbf{S}(t)$ be a Bezier curve approximating γ. Choose a discrete parametrization $0 = t_1 < t_2 < \cdots < t_m = 1$ and measure the distance between the curves by

$$E = \frac{1}{m} \sum_{i=1}^{m} \|\mathbf{x}_i - \mathbf{S}(t_i)\|_2^2. \tag{1}$$

The main task is now to find S, defined by its control points, so that E becomes as small as possible. We know two things which will be helpful:

- $\mathbf{P}_0$ and $\mathbf{P}_3$ are equal to the first and the last point on the curve.
- $\mathbf{P}_1$ and $\mathbf{P}_2$ are located somewhere at the tangents of the curve at the endpoints, or more specific, if $\mathbf{v}_0$ and $\mathbf{v}_3$ are the unit tangent vectors of γ at the endpoints, then

$$\mathbf{P}_1 = \mathbf{P}_0 + \alpha_1 \mathbf{v}_0, \qquad \mathbf{P}_2 = \mathbf{P}_3 + \alpha_2 \mathbf{v}_3.$$

We are left with two unknowns: α_1 and α_2, and $E = E(\alpha_1, \alpha_2)$.

Task 3 *Set up the mean square problem, that is, minimize $E(\alpha_1, \alpha_2)$. Use the data from one of the test problems and solve the problem in MATLAB.*
Plot the orignal curve as well as the Bezier curve, and compare them.

Task 4 *Do some experiments:*
Play a bit with the given data: Use less of them, and see how this will influence the result.
Change the choices of parametrizations points t_i and see the impact of that on the result.

2.2 Choice of discrete parametrization

So far, we have assumed the discrete parametrization $\{t_j\}_{j=1}^m$ for given. But this is not a part of the original curve γ, it has to be chosen somehow. In [1] a procedure for an initial parametrization is given, as well as a procedure for improving this choice.

Task 5 *Implement and test the procedure, both the initial parametrization and the improvements. You may or may not want to make your own modifications.*

2.3 Further improvements

If the distance between the curve and the Bezier curve is still too large, measured with (1), then it is possible to divide the curve in two, and construct a Bezier curve for each half. If so, make sure that the approximated curve is sufficiently smooth, that is C^1.

3 Finally

Write a MATLAB-program which will convert shapes defined by a discrete set of data points to a set of Bezier curves, and test it on some picture of your own choice. Techniques for the detection of edges of given bitmap picture in MATLAB will be provided.

References

[1] M. Plass and M. Stone, *Curve-fitting with piecewise parametric cubics*, Computer Graphics, 17 (1983) pp. 229–239.