Norwegian University of Science
and Technology
Department of Mathematical
Sciences

TMA4215 Numerical mathematics Autumn 2014

Project 2

Last update: November 6, 2014

Instructions

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

Deadline: November 18, 23:59.

Group size: 3–4 students.

To be handed in

- 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 well structured, well documented and self-contained MATLAB file.
- Optionally, you can submit one image file that is used by your program.

Send the files (maximum 3) to Lars 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 parametric 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 have 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. However, 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 last days 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 does, and how to use it. This information should be available by writing `help filename`.
  - executes and provides the expected results without any problems.
- The tasks described below are there to help you 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. However, you may want to include some of them as examples, or to justify that the described procedure works.

  We do not want you to present all the tasks in your MATLAB file. What we want is to be able to reproduce the last result(s) you present in your report with that program.

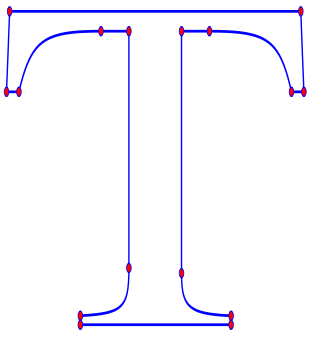
The LaTeX template, including some instruction on how to write the report, as well as some data files for the tasks, will be provided on the project webpage.

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 right 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

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

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 $n = 3$. 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 down 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. Similarly, the straight line between the last two control points is tangential to the curve in P_3. Prove that this is true in general. What impact does this have if you want to glue several Bezier curves into one smooth curve?

Task 2 *Write a Matlab program which reads 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 can we approximate this curve by one Bezier curve? To be more specific, how can we find the control points?

2.1 With a given, discrete parametrization

Assume we are 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 $\mathbf{S}$, 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 along the tangents of the curve at the endpoints. To be 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. Thus $E = E(\alpha_1, \alpha_2)$.

Task 3 *Set up the least square problem, that is, minimize $E(\alpha_1, \alpha_2)$. Use the data from the test problem given on the webpage 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 influence the result.
Change the choices of parametrization points t_i and see the impact of that on the result.

2.2 Choice of discrete parametrization

So far, we have assumed that the discrete parametrization $\{t_j\}_{j=1}^m$ is given. However, this is not 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. Use, for instance, the test problem from Task 3, but with the new parametrization.*

2.3 Code verification

To verify an implementation, it is common to apply it to a problem where the analytical solution is known.

Task 6 *Verify your code by testing it on the curve*

$$y = \sin(\pi x), \quad x \in [0, 1].$$

Discretize the curve into a discrete set of points, $(x_i, \sin(\pi x_i)$*, where* $x_i = i\Delta x$*, for* $i = 0, 1, \ldots, 100$*, and* $\Delta x = 0.01$*. Start by approximating the discrete curve with one Bezier curve. Then, repeatedly divide the curve in two, and construct a Bezier curve for each half. Make sure that the approximated curve is sufficiently smooth, that is* C^1*. Use the analytic expression to calculate the tangent vectors. Plot the total error against the number of curve segments in a log-log plot and comment on the result. Include this figure in your report.*

3 Finally

Write a MATLAB-program which converts shapes defined by a discrete set of data points to a set of Bezier curves. MATLAB-routines for the detection of edges and corners of a given bitmap picture is provided on the webpage. If the error measured by (1) is too large for a segment, cut the segment in two and construct a Bezier curve for each half. Remember to keep the curve smooth, that is C^1, everywhere except at the corner points. You will have to come up with a procedure for approximating the tangent vectors at start- and endpoints of the Bezier curves.

Test your program on the bitmap picture given on the webpage and on at least one picture of your own choice. Your program should plot the original curve and the Bezier curves in the same figure. Furthermore, mark all corner points, and color adjacent curve segments with two different colors.

References

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