On a Problem of Fitting Data Using Bézier Curves

Carmen-Violeta Muraru
Department of Mathematics and Computer Science,
Faculty of Sciences, "Vasile Alecsandri" University of Bacău
Calea Mărăşeşti, 157, Bacău, 600115, Romania
carmen_7419@yahoo.com, cmuraru@ub.ro

Abstract

The paper's aim is to study old and new problems regarding the Bézier curves, which are important tools in the geometric modelling of shapes. We use the Matlab software to study the estimation error of fitting data using Bézier curves least square fitting and to find new methods within a study that is currently under development about minimizing the distance between the curve and the approximated data.

Keywords: Bézier curve, fitting data

1. Introduction

Computer-Aided Geometric Design (CAGD) is the basis for modern design in most branches of industry, from naval and aeronautic to textile industry and medical imaging. The technology using 3*D* graphics, virtual reality, animation techniques, etc, requires storing and processing complex images and complex geometric models of shapes (face, limbs, organs, etc). In this context, it is necessary to understand better how to *discretize* geometric objects such as curves, surfaces, and volumes. Some of the design problems are handled by breaking the curves or surfaces into simpler pieces and then specifying how these pieces are joined together with some degree of smoothness. For a detailed study, we would recommend authors such as Gallier [1], Khan [2], Farin [5], Recktenwald [6], Gravesen [7], etc. Usually, the S shape is modeled by a system of equations and we view these equations as defining a certain function. Polynomials are incredibly useful mathematical tools because they can be differentiated and integrated easily, and can be pieced together to form *spline curves* that can approximate any function to any accuracy desired. Parametric curves and surfaces representation form the base on which the theory of differential geometry rests and they serve as the point of departure for many CAGD practical applications. On the other hand because they offer more flexibility, rational functions are often preferred to polynomial functionals to model curves and surfaces.

2. Applications of Bézier curves in graphic design

The Bernstein basis occurs frequently both in Computer Aided Geometric Design and in Approximation Theory for many applications in CAGD. Usually, the curves in the CAGD problems will be defined by a sequence of control points. The curves that are useful in geometric modeling should have a relatively smooth shape and should be intuitively connected with the path of the sequence of control points. One family of curves satisfying this requirement is represented by the Bézier curves which use the Bernstein polynomials as basis functions. The basis formed by the Bernstein functions, $\{B_0^n(t), B_1^n(t), ..., B_n^n(t)\}$ spans also the linear space P^n, the space of all polynomials of maximal degree n, where (1)

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

Given N+1 control points P_i with i =0 to n, the Bézier parametric curve is given by B(t) as follows (2):

$$B(t) = \sum_{i=0}^{n} B_i^n(t) P_i, \ 0 \le t \le 1 \qquad (2)$$

In formula (2) we can easily detect the form of the Bernstein basis.

Bézier curves are parametric curves, meaning that the formula above is applied independently to the *x* and *y* coordinates of the points for a 2D curve, resulting in (3).

$$\begin{aligned} B(t).x &= \sum_{i=0}^{n} P_i.x \cdot B_i^n(t) \\ B(t).y &= \sum_{i=0}^{n} P_i.y \cdot B_i^n(t) \end{aligned} \qquad (3)$$

The Bézier curve passes through its first and last control points, i.e. *P*0 and *P*3. The *middle control points*, i.e. *P*1 and *P*2 determine the shape of the curve. Rational Bézier curves are fundamental for geometric modeling. It is widely known that rational Bézier curves of degree two are conics. The curve scheme has been based on mathematical representations of the Bernstein Bézier, B-spline and NURBS curves. The construction of the Bézier curves makes use of the configuration of given control points and polynomial functions, the results of which are curves segments residing within the convex hull of those control polygons. One of the disadvantages of this type of curve is that when the number of control points is increased, the degree of the curve equation will be higher. The computational time required will be of quadratic complexity even though the de Casteljau algorithm has been used for the evaluation of the curve segments so the utility in real world application is reduced. Each point of the curve is calculated from a parametric mathematical function which uses the coordinates of the control points as parameters. Bézier curves were the adopted solution to describe an image in terms of its mathematical representation. With this type of curves, the way graphics was also changed by introducing the concept of *vector graphics* as opposed to raster images. The vector graphic file will contain the coordinates of control points to describe a series of curves. *The problem of zoom in became a matter of increasing the space between the control points and by redrawing a perfectly smooth curve again.* There are a lot of very interesting properties of the Bézier curves. Below are the few that are relevant to our current article:

- At t=0, B(t) is P0 and at t=1, B(t) is P_N. This means that the Bézier curves begin at the first control point (i=0) and finish at the last (i=N).
- At t=0 the curve is tangent to the line (P_0, P_1) and at t=1, the curve is tangent to the line (P_N, P_{N-1})
- The curve is always contained within the convex shape described by the control points.

As we have mentioned above, the Bézier curve uses the Bernstein basis function. Below are two examples of Bézier curves with the associated polygons, in Matlab.(Figure 1).

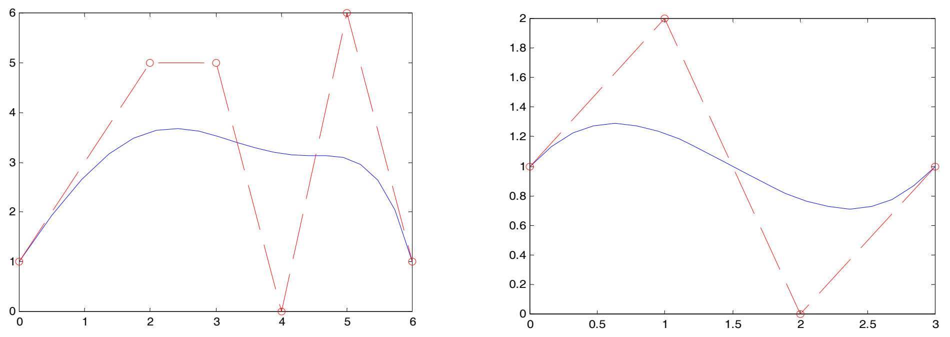

Figure 1. Examples of cubic Bézier curves ([4])

Another choice for basis polynomials could be the so called Hermite polynomials where the input to the computer would now be the coordinates with respect to the Hermite basis. This representation is indeed in use in industry, and the cubic polynomial curve in the Hermite representation was introduced and published in 1964 [9] by James Ferguson at Boeing, being thus known as the *Ferguson curve*.

3. A problem of fitting data using Bézier curve least square fitting

A common problem that often needs to be solved, for example in many branches of industry or in medical imaging, is fitting a curve or a surface through a set of data points.

Problem: Given $N+1$ data points $x0, \dots, xN$ and a sequence of $N+1$ reals $u_0, u_1, \dots, u_N$, with $u_i \le u_{i+1}$ for all i, $0 \le i \le N - 1$, find a $C2$-continuous curve F, such that $F(ui) = xi$, for all i, $0 \le i \le N$.

There are many different types of curves that can solve the above problem (defined by Fourier series, Lagrange interpolants, spline functions, etc), and we need to be more specific as to what kind of curve we would like to use. In most cases, efficiency is the dominant factor, and it turns out that piecewise polynomial curves are usually the best choice because these are the easiest functions to manipulate. Even then, the problem is still underdetermined.

However, the problem is no longer underdetermined if we specify some "end conditions", for instance the tangents at $x0$ and xN. In this case, it can be shown that there is a unique Bézier curve solving the above problem if we choose this type of curve. With this type of problem, we also want to minimize the error approximation term.

As an input we specify the value of error and provide initial set of breakpoints, i.e., the first point and the last point of the original data. Input data is divided into segments based on the initial set of breakpoints. A segment is set of all points between two consecutive breakpoints. We have to fit each segment using cubic Bézier curve(s). By means of the fitting process we generate n points (approximated data) using Bézier interpolation so that the Bézier curve(s) passes through the breakpoints. Then we measure the error between the original and the approximated (fitted) data.

When the approximated data is not close enough to the original data, an existing segment is split into two segments at a point called new breakpoint. After the splitting number of segments increases by one (a split segment is replaced by two new segments), the number of breakpoints also increases by one (a new breakpoint is added to the set of existing breakpoints). The point where the error is maximum between the original and the approximated data is selected as a new breakpoint and this point is added to the set of breakpoints.

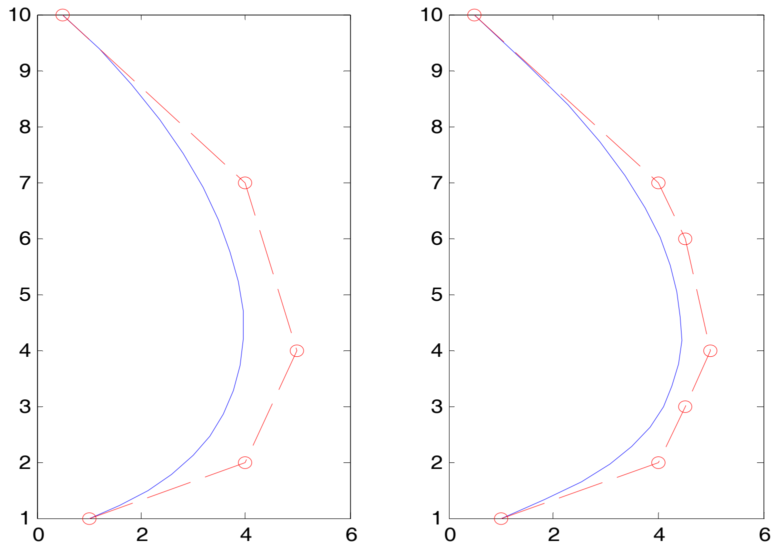

Figure 2. The process of splitting segments by inserting new points

After splitting, the same fitting procedure using an updated set of segments and breakpoints is repeated until error is less than or equal to the limit of error. This technique of fitting is called the *break- and-fit* strategy (see [2]). As we can see in the following figure, this results in replacing the old sequence P_1, P_2, P_3 with a new one $P_1, P_1', P_2, P_2', P_3$ by adding two new points. It is obvious that the new curve is now closer to the control polygon (Figure 2).

We take into account that on the Bézier curve there will be found n+1 points obtained for the next values of parameter

t: $0/n, 1/n, 2/n, ... k/n, ..., n/n$ corresponding to n+1 control points.

In Figure 3 we represent two control polygons, P0-P1-P2-P3-P4-P5 and P0-P1-P2-P3'-P4-P5, and the coordinates are:

CP: (1, 0.5), (0.25, 2.00), (1.25, 3.75), (2.50, 4.00), (4.00, 3.25), (5.00, 1.00) and

CP: (1.00, 0.5), (0.25, 2.00), (1.25, 3.75), (2.00, 0.25), (4.00, 3.25), (5.00, 1.00).

As it can be observed, P3' is out of place in a new position and the new Bézier curve changed its shape. If the control point P_i is moving with D_i then the new control point will have the coordinates

$$P_i^{'} = P_i + D_i, \quad \begin{bmatrix} x_i \\ y_i \end{bmatrix}^{'} = \begin{bmatrix} x_i \\ y_i \end{bmatrix} + \begin{bmatrix} d_x \\ d_y \end{bmatrix} \quad (4)$$

where d_x, d_y represent the coordinates of displacement.

In the next figure, based on the Matlab code, it is obvious that the positions of all the points of the Bézier curve will change.

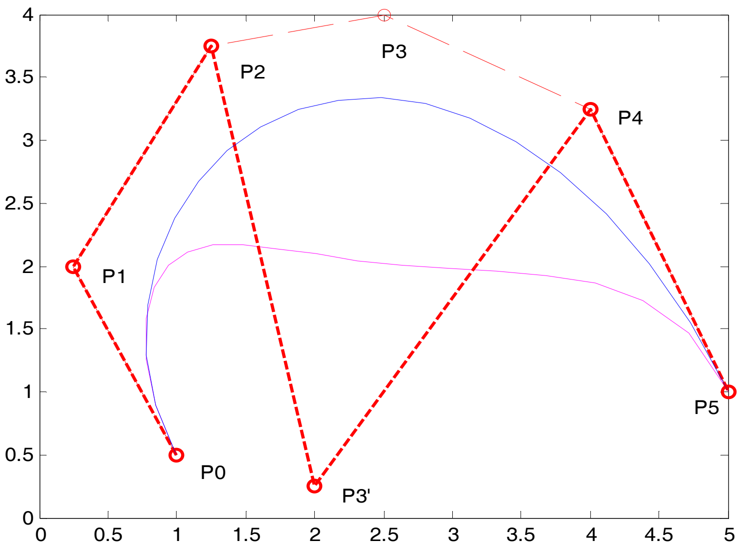

Figure 3. Two control polygons

For the following sequence of parameters values $\left\{\frac{0}{10}, \frac{1}{10}, \frac{2}{10}, ..., \frac{9}{10}, \frac{10}{10}\right\}$, we have the coordinates for the point of the Bézier curve in Table 1.

The maximum distance d=1.2960 corresponds to the point P_3 of the control polygon which corresponds to $t = 0.60$. In a matter of fitting data it is important to estimate the error approximation. One of the conditions for bringing the Bézier curve closer to the approximated data is to minimize the next sum (5):

$$S = \sum_{i=0}^{n} (P_i - B(u_i))^2 \qquad (5)$$

Table 1. Coordinates for the control polygon of the Bézier curve

Parameter *t*	**Coordinates of the Bézier curve for the first control polygon**		**Coordinates of the Bézier curve for the second control polygon**	
0.00	1.0000	0.5000	1.0000	0.5000
0.10	0.7857	1.2586	0.7817	1.2282
0.20	0.8413	1.9770	0.8157	1.7850
0.30	1.1003	2.5857	1.0341	2.0896
0.40	1.5090	3.0347	1.3938	2.1707
0.50	2.0234	3.2891	1.8672	2.1172
0.60	2.6070	3.3253	2.4342	2.0293
0.70	3.2276	3.1274	3.0732	1.9698
0.80	3.8547	2.6830	3.7523	1.9150
0.90	4.4571	1.9795	4.4207	1.7062
1.00	5.0000	1.0000	5.0000	1.0000

In this case of a Bézier curve which fits the control points, there is a certain position of the middle point of the curve for minimizing the squared distance between the original and the fitted data and it is well suited for data approximation. As we have mentioned above, by inserting a new breakpoint the curve gets closer to the polygon and we obtain some numerical results which are presented in Table 2.

Table 2. Some numerical results

The control polygon	$\sum_{i=0}^{n}(P_i - B(u_i))^2$
CP=[1.00, 0.50; 0.25 ,2.00; 1.25, 3.75; 2.50, 4.00; 4.00, 3.25; 5.00, 1.00];	10.9797
CP=[1.00, 0.50; 0.25, 2.00; 1.25, 3.75; 2.50, 4.00;3.00, 3.75; 4.00, 3.25; 4.50, 2.50; 5.00, 1.00];	7.9833
CP=[1.00, 0.50; 0.25, 2.00; 1.25, 3.75; 2.50, 4.00;3.00, 3.60 ;4.00, 3.25; 4.00, 2.50; 5.00, 1.00];	7.7241
CP=[1.00, 0.50;0.50, 1.00; 0.25, 2.00;0.50, 2.80; 1.25, 3.75; 2.50, 4.00; 3.00, 3.75 ;4.00, 3.25; 4.50, 2.50;5.00,1.00]	2.4137
CP=[1.00, 0.50;0.50, 1.00; 0.25, 2.00;0.50, 2.80; 0.70, 3.30;1.25, 3.75; 2.50, 4.00; 3.00, 3.75 ;4.00, 3.25; 4.50, 2.50;5.00, 1.00];	0.6438

According to Table 2, it is obvious that the distance between the Bézier curve and the approximated data is getting smaller.

For a better understanding of the numerical results from Table 2, Figure 3 shows how the shape of the Bézier curve changes according to the last two control polygons.

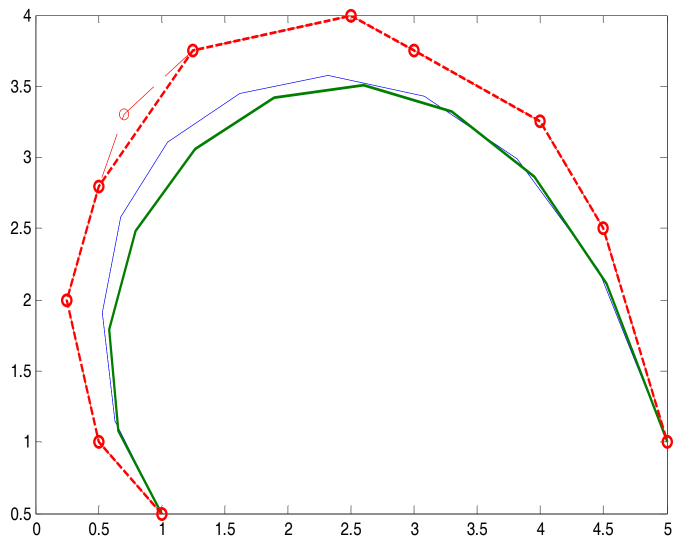

Figure 3. The changing of the shape of the Bézier curve

4. Conclusions

Geometric modeling and computer graphics have been interesting and important subjects for many years from the point of view of scientists and engineers. One of the main and useful applications of these concepts is the treatment of curves and surfaces in terms of control points, a tool extensively used in CAGD. The paper is part of a study under development regarding the efficiency and advantages of using Bézier curves in the process of fitting data in fields ranging from the automative and aeronautic to textile industry or medical imaging.

5. References

[1] Gallier, J. (2008). *Curves and Surfaces In Geometric Modeling Theory And Algorithms*.
[2] Khan, M. *Approximation of data using cubic Bézier curve least square fitting*. Retrieved from http://docs.google.com/viewer?a=v&q=cache:WgF5avMgoeUJ:obberon.com/documents/cubicbezierleastsquarefit.pdf+Khan+Approximation+of+data+using+cubic+B%C3%A9zier+curve+least+square+fitting&hl=ro&gl=ro&pid=bl&srcid=ADGEESiFwpI5EMrCjPPG7IoHXDkzEfImLY_NQBgWS-riQd_sDgrXBRSv_oI_BtFPPyQrFK0bQpHG4nseteXbxqajChI-XC2K3MN18tFcVHhYmih6RS9z_Ng6Ibk07BeILlfX4KgwU_F6&sig=AHIEtbRnlJ7i7zc8skiKho6PGmWbSzVpuw
[3] Ursu-Fischer, N., Ursu, M. (2003). *Numerical Methods in Technique and Programs in C/C++*. Cluj-Napoca: Ed. Casa Cărții de Stiință.
[4] Nimineț, V., Muraru, C.V. (2009).*Computational Geometry with applications in Matlab.* Iaşi: Ed. PIM.
[5] Farin, G. (1997) *Curves and Surfaces for Computer - Aided Geometric Design. A Practical Guide*, Fourth Edition, Academic Press.
[6] Recktenwald, G. (2007). *Numerical Methods with Matlab: Implemantation and Applications,* Ed. Prentice-Hall.
[7] Gravesen, J. (2002). *Differential Geometry and Design of Shape and Motion.*