First International Conference
Modelling and Development of Intelligent Systems
Sibiu - Romania, 22-25 October, 2009

E-learning solutions in curve and surface design

Corina Simian

Abstract

The aim of this paper is to present a tool which can be integrated in an e-learning system, allowing students to learn advanced topics in curve and surface design. The tool is designed using MATLAB platform. The design of this tool implies advanced knowledge related to CAGD. We used Bezier curves and surfaces and introduced a new interpolation algorithm for obtaining Bezier spline curves of C^1 class.

1 Introduction

The design of curves and surfaces is a topic of great interest in CAGD. A large number of courses from bachelor and master level are dedicated to this topic. The wide area of applications made this topic more attractive. The theory of Bezier curves and surfaces and Coons patches, together with B-spline methods were the reference points in CAGD. Many interactive applications for construction of Bezier curves and surfaces, most of them implemented in Java, can be found on Internet. The aim of this article is to present a tool which can be used for accomplish many e-learning tasks in the field of Bezier curves and surfaces. For implementation we used MATLAB platform.

The paper is organized as follows: in chapter 2 we briefly present the problem of approximation of curves and surfaces using spline curves and surfaces in Bezier form and few algorithmic aspects. Section 3 presents the advantages of MATLAB for implementation. The main result, MathTool, is presented in section 4. Section 5 contains conclusions and further developments.

2. Interpolation using Bezier curves and surfaces

A Bezier curve of degree n is defined using a set of control points, b_i, i=0,...,n. The parametric form of a Bezier curve is given using the Berstein basis.

$$f(t) = (x(t), y(t), z(t)) = \sum_{i=0}^{n} b_i B_i^n(t), t \in [0,1], b_i = (x_i, y_i, z_i) \quad (1)$$

The Bernstein polynomials are defined as:

$$B_i^n(t) = \binom{n}{i} (1-t)^{n-i} t^i; t \in [0,1]; i = 0,1,...,n \quad (2)$$

The most important case, from a practical point of view is the case of cubic Bezier curves, obtained for n=3.
More details regarding the Bezier curves can be found in [3].
The problem of interpolation using Bezier curves is the following one: given *n+1* points, P_i, *i=0,...,n*, find a curve *C*, in Bezier form, such that $P_i \in C, \forall i = 0,...,n$, satisfying given conditions of continuity and derivability, .

A kind of "pseudo-interpolation" consists in finding the control points of the curve *C* (see [6], [7]). Let consider the cubic case. We have known 4 points from a curve and find the 4 control points of a cubic Bezier curve which contains these 4 points. We suppose that the given points correspond to equidistant values of parameter *t*. The interpolation conditions lead to:

$$c(0) = b_0 = P_0; \qquad c(\frac{1}{3}) = \frac{8}{27}b_0 + \frac{12}{27}b_1 + \frac{6}{27}b_2 + \frac{1}{27}b_3 = P_1; \quad (3)$$

$$c(\frac{2}{3}) = \frac{1}{27}b_0 + \frac{6}{27}b_1 + \frac{12}{27}b_2 + \frac{8}{27}b_3 = P_2; \qquad c(1) = b_3 = P_3$$

The equalities (3) can be written in the form (4):

$$b = A^{-1} * p, \qquad (4)$$

$$A = \frac{1}{27}\begin{pmatrix} 27 & 0 & 0 & 0 \\ 8 & 12 & 6 & 1 \\ 1 & 6 & 12 & 8 \\ 0 & 0 & 0 & 27 \end{pmatrix}; P = \begin{pmatrix} P_0 \\ P_1 \\ P_2 \\ P_3 \end{pmatrix} = \begin{pmatrix} Px_0 & Py_0 & Pz_0 \\ Px_1 & Py_1 & Pz_1 \\ Px_2 & Py_2 & Pz_3 \\ Px_4 & Py_4 & Pz_4 \end{pmatrix}; b = \begin{pmatrix} b_0 \\ b_1 \\ b_2 \\ b_3 \end{pmatrix} = \begin{pmatrix} x_0 & y_0 & z_0 \\ x_1 & y_1 & z_1 \\ x_2 & y_2 & z_2 \\ x_3 & y_3 & z_3 \end{pmatrix}$$

If the number of interpolation points is greater, we use a spline Bezier curve. The j-th component of a spline curve of degree *n*, composed by *m* pieces, has the equation:

$$c_j(t) = \begin{pmatrix} x(t) \\ y(t) \\ z(t) \end{pmatrix} = \sum_{i=0}^{n} b_i^j B_i(t - j + 1), t \in [j-1, j], j = 1,...,m \qquad (5)$$

A spline Bezier curve is of class G^0 (see [3]), iff $b_n^j = b_0^{j+1}, \forall j = 1, \ldots, m-1$.

A spline Bezier curve is of class G^1, if the control points $b_{n-1}^j, b_n^j = b_0^{j+1}, b_1^{j+1}$ are collinear. There are different approaches for reaching G^1 class of a spline Bezier curve (see [3]).

The tensor product Bezier surface patch is controlled using a mesh of control points, b_{ij} and is defined as:

$$s(u,v) = \begin{pmatrix} x(u,v) \\ y(u,v) \\ z(u,v) \end{pmatrix} = \sum_{i=0}^{n}\sum_{j=0}^{m} b_{ij} B_j^m(u) B_i^n(v); (u,v) \in [0,1] \qquad (6)$$

The bi cubic tensor product Bezier surface is given by the relations:

$$s(u,v) = Bv' \cdot b \cdot Bu \qquad (7)$$

with *Bv* the column matrix of Bernstein polynomials in variable *v*, *Bv'* the transpose matrix of *Bv*, *Bu* the column matrix of Bernstein polynomials in variable *u* and $b=[b_{ij}]$, the 4x4 matrix, with the coordinates of the 16points of the control mesh of the Bezier surface.

3. MATLAB in curve and surface design

MATLAB offers many advantages in implementation of the algorithms from CAGD and in particular for algorithms which handle Bezier curves and surfaces:

a) Easy matrix formulation of problems in MATLAB. Best performance of the programs which use specific operations with matrix in MATLAB
b) Graphical capabilities. MATLAB provides high-level functions for displaying graphics.
c) Specific toolboxes can be developed from users.
d) MATLAB provides powerful tools for Graphical User Interfaces design.

The most important advantage is the possibility of matrix formulation of problems in MATLAB. All the relations presented in section 2, can be expressed in a matrix form, which enable to write performing programs and easy to understand. The representation of a Bezier curve using (1), is illustrated forward

```
function CurbaBezierf(b)
%computation
t=0:0.01:1;%parameter
B0=(1-t).^3
B1=3*(1-t).^2.*t
B2=3*(1-t).*(t.^2)
B3=t.^3
%Bernstein polynomial matrix
B=[B0;B1;B2;B3]
%Bezier curve
f=b*B;
%graphical representation
hold on
title(Bezier curve');
plot(b(1,:),b(2,:),'r-')
plot(f(1,:),f(2,:),'b')
hold off
```

The representation of a bi cubic tensor product Bezier surface patch is made very easy using the matrix form given in (7). The function *bez3surf*, uses the coordinates of the control points of the surface and return the coordinates of a mesh of points from the Bezier surface. Using this mesh, the surface can be represented using the MATLAB graphical function surf, or mesh.

```
function [x,y,z] = bez3surf(bx,by,bz)
u=0:0.01:1; %parameter
v=0:0.01:1;
B0u=(1-u).^3;

B1u=3.*(1-u).^2.*u;
B2u=3.*(1-u).*u.^2;
B3u=u.^3;
%matrix of Bernstein polynomial in variable u
Bu=[B0u;B1u;B2u;B3u];

B0v=(1-v).^3;
B1v=3.*(1-v).^2.*v;
B2v=3.*(1-v).*v.^2;
B3v=v.^3;
%matrix of Bernstein polynomial in variable v
Bv=[B0v;B1v;B2v;B3v];
```

```
%coordinates of points from the Bezier surface
x=Bv'*bx*Bu;
y=Bv'*by*Bu;
z=Bv'*bz*Bu;
```

In the case of interpolation using a bi cubic tensor product Bezier surface patch, we use 16 interpolation points and find the 16 points from the control mesh of the Bezier surface. We repeat first on lines and then on columns the procedure for interpolation of a curve using 4 interpolation points, given in relation (4). The function *bsplines* computes the coordinates of the control points of the Bezier interpolation curve. The function *bezinterpsurf* applies the tensor product method for obtaioning the coordinates of the control points of interpolation Bezier surface.

```
function[bx]= bsplines(px)
%px is 1*4 matrix with a coordinate (x,y or z) of 4 points from the surface
%bx is a matrix with the same coordinate of the control points of Bezier
%curve
A=(1/27)*[27. 0 0 0; 8. 12. 6. 1.; 1. 6. 12. 8.; 0 0 0 27.];
bx = inv(A)*px';
bx=bx';
```

```
function [bx] = bezinterpsurf(px)
%px is a 16x16 matrix, containing one of the coordinates (x,y or z) of the
%interpolation points.
%function bsplines is applied on lines
b1=ones(4,4);
b11 = bsplines(px(1,:));
b12 = bsplines(px(2,:));
b13 = bsplines(px(3,:));
b14 = bsplines(px(4,:));
b1=[b11; b12; b13; b14]
%function bsplines is applied on columns
b=ones(4,4);
bc1=(b1(:,1))';
bc2=(b1(:,2))';
bc3=(b1(:,3))';
bc4=(b1(:,4))';
b21 =bsplines(bc1);
b22 =bsplines(bc2);
b23 =bsplines(bc3 );
b24 =bsplines(bc4);
bx=[b21; b22; b23; b24];
bx=bx';
```

The functions presented before are called from the function *beziertensor*, which realizes the graphical representation of the interpolation surface.

```
function beziertensor(px,py,pz)
bx= bezinterpsurf(px);
by= bezinterpsurf(py);
bz=bezinterpsurf(pz);
[sx,sy,sz] = bez3surf(bx,by,bz);
%bz=bz'
surf(sx,sy,sz);hold on
shading interp;
```

```
colormap cool;
alpha 0.5;
plot3(px,py,pz,'ro');
hold on
```

The examples presented in this section proved the advantages of MATLAB in curve and surface design using Bezier curves and surfaces.

4 Main results - MathTool

4.1 General presentation

Our aim was to allow a collaborative learning in the field of curve and surface design (using Bezier curves and surfaces) and to develop algorithmic and programming skills. Our tool, named MathTool, is a web based tool, which includes many sections. Theoretical elements are grouped in 4 sections, containing many items and providing a guide in curve and surface design. The sections are:
1) Theoretical fundamentals: contains basic mathematical elements regarding multivariate polynomial interpolation, spline interpolation, Bernstein polynomials.
2) Curves and surfaces in CAGD: presents a short history of CAGD, parametric representation of curves and surfaces and different types of curves and surfaces used in CAGD.
3) Bezier curves: includes theoretical, algorithmic and computational aspects related to Bezier curves and interpolation using Bezier curves.
4) Bezier surfaces: contains theoretical, algorithmic and computational aspects related to Bezier surfaces and interpolation using spline surfaces in Bezier form. It is presented the tensor product method. The ruled surfaces, cylindrical surfaces, sweep surfaces are taken into account.

Every section has a corresponding MATLAB GUI, to illustrate the presented notions.
The user has access to the source code, after registration in the MathTool interface. The call-back functions from the GUI are easy to change and to adapt such that new applications can be obtained. We included a guide which contains a detailed description of the algorithms used, implementation details and instructions to use the MATLAB GUIs.
References for a deeper research in the domain of curve and surface design are given in a special section.
For exchanging ideas or posing questions, a discussions forum is included and can be used.

The MathTool Interface is presented in fig. 1. In fig. 2 is presented a list of the sources which can be accessed by a user

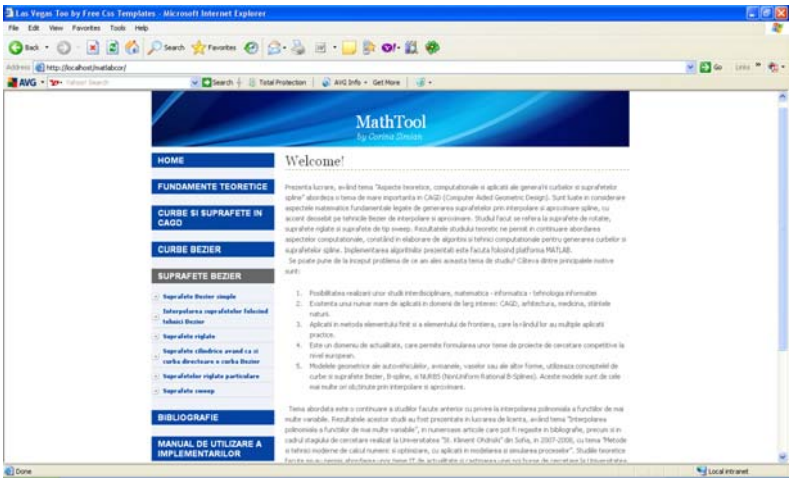

Fig. 1 – MathTool Interface

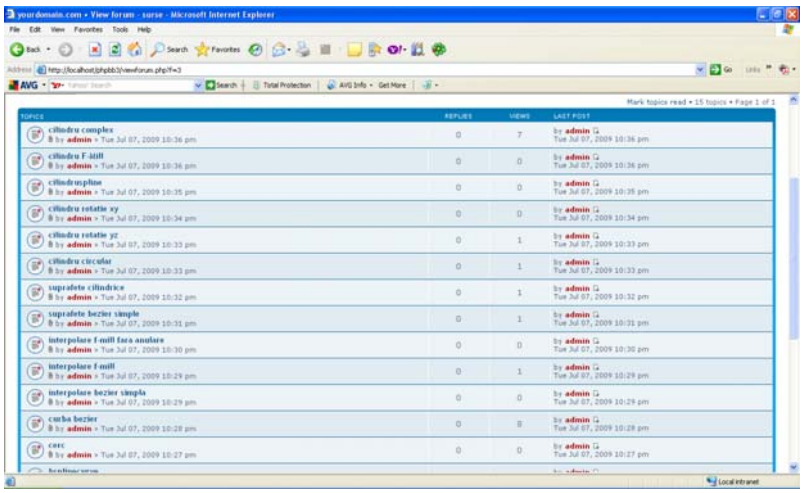

Fig. 2 –List of the MATLAB files

4.2 MATLAB GUIs

The aim of this subsection is to present the most important implementations included in MathTool. The implementations are made in MATLAB and offer a friendly Graphic User Interface (GUI).

4.2.1 Interactive Bezier Curve

A special pop-up menu created in the MATLAB figure windows allows different representation of a cubic Bezier curve.

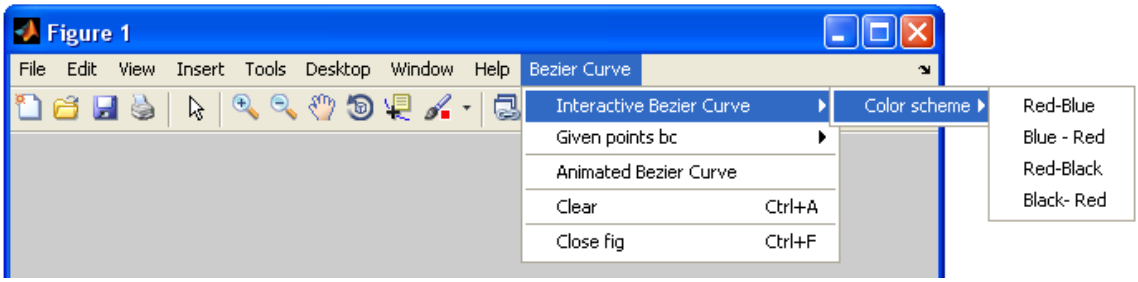

Fig. 3 Pop–up Bezier curve menu

Interactive representation of a cubic Bezier curve is presented in fig. 4.

4.2.2 Interpolation using Bezier curves

Many GUI were designed for interpolation using Bezier curves: F-Mill interpolation, interpolation using spline curves in Bezier form of class G^0 and G^1(see [1]-[3], [6] for more details of these methods).

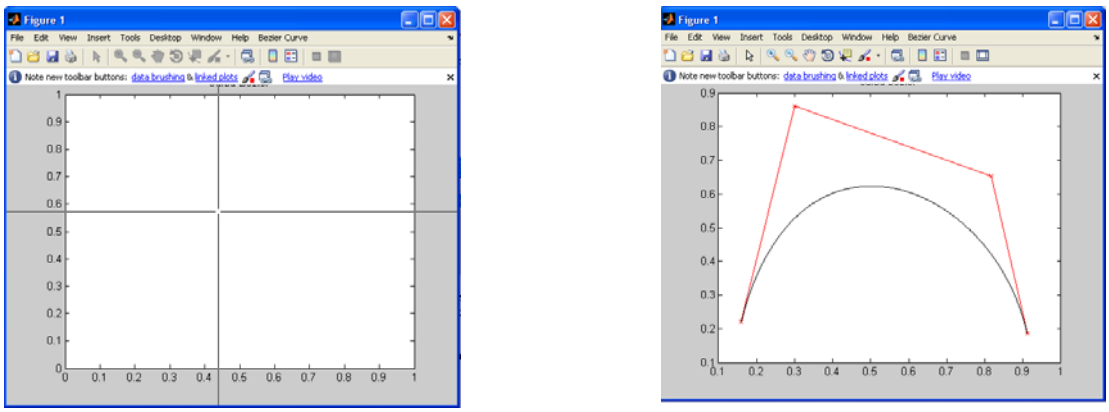

Fig. 4 Interactive representation of a cubic Bezier curve

All GUIs allow the input from the graphical interface of the interpolation points coordinates and provide the spline interpolation curve, the control polygon and the interpolation points. As an application, a GUI for approximation of one or two circular arcs of given sweep is realized.

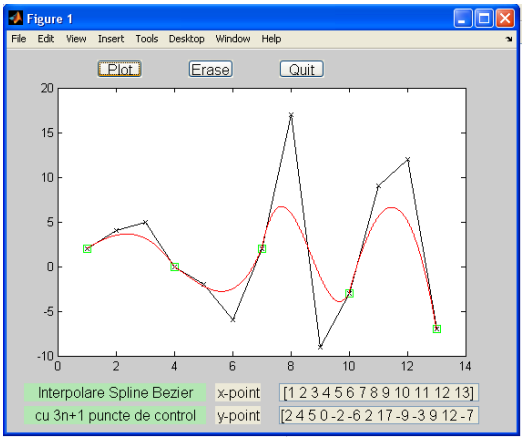

Fig. 5 Bezier spline curve with 3n+1 control points

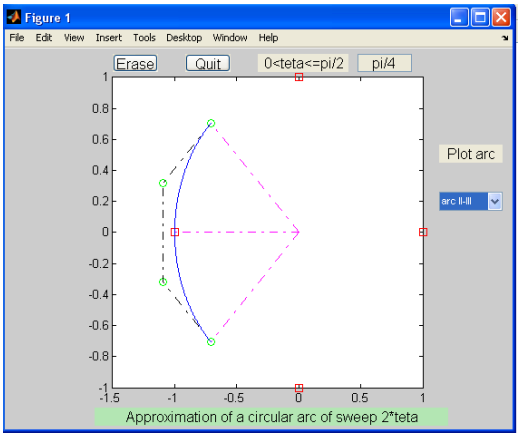

Fig. 6 Approximation of circular arc

4.2.3 Interpolation surfaces in Bezier form

Many GUIs for obtaining interpolation surfaces in Bezier form were designed: interpolation bicubic Bezier surfaces, cylindrical surfaces, rotation surfaces.

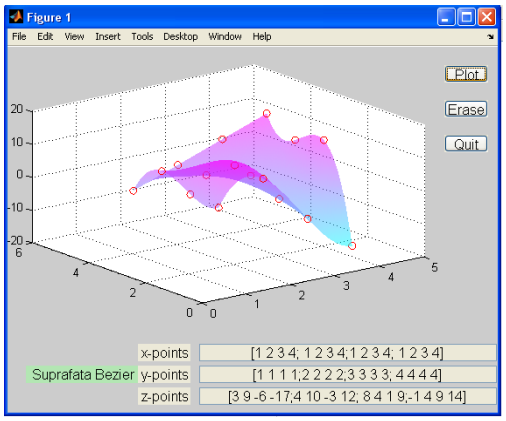

Fig. 7 Bicubic interpolation Bezier surface

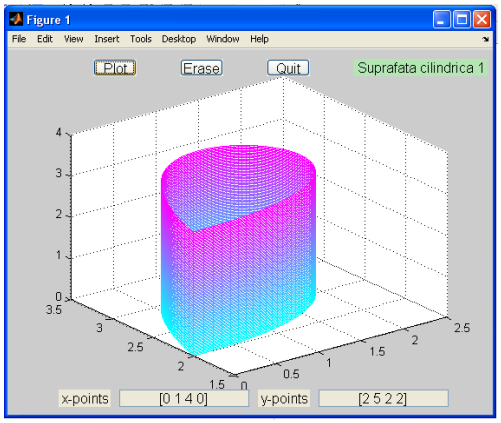

Fig 8- Cylindrical surface based on Bezier curve

4.3 E-learning goals of MathTool

MathTool was created in order to accomplish many e-learning tasks:
1. Computer-based learning of topics from Computational Geometry and especially from CAGD. Both the theoretical elements and the graphical interfaces from MathTool are useful in distance learning, or in blended learning (see [10]).
2. Computer-based learning of MATLAB programming and MATLAB GUI design. All implementations are relevant for MATLAB programming. The source can be visualized and all details about the algorithms and implementations are given in the User Implementation Guide.
3. Computer – based training in Bezier curves and surfaces and their applications in CAGD.
4. Computer-supported collaborative learning. To encourage students to work together and to cooperate, a forum of discussions was included in the MathTool.

5. Conclusions

In this article we present a web based tool which can be used in computer based-learning and training in topics from CAGD and MATLAB programming. It offers also a collaborative learning support. A great advantage is the modularization of the applications and the possibility of easy changing and adapting the modules for other goals.
A further development of our tool will offer the possibility to extended the set of programs and knowledge from the users and to introduce a section of open problems.

References

[1] W. Boehm, G. Farin and J. Kahmann, *A survey of curve and surface methods in CAGD Computer Aided Geometric Design*, 1–60, 1984.
[2] C. de Boor, K. Hollig and M. Sabin, High accuracy geometric Hermite interpolation. *Computer Aided Geometric Design*, 4:4,269–278, 1987.
[3] G. Farin, *Curves and Surfaces for Computer Aided Geometric Design. A practical guide*, Academic Press, Boston, 1996.
[4] G. Farin, Class A Bezier Curves,*Computer Aided Geometric Design*, 23, Elsevier Press, pp.573-581, 2006
[5] M. A. Khan, Approximation of Circle by Cubic Bezier Curves, http://www.mathworks.com/matlabcentral/fileexchange/6844
[6] D. E Ulmet., Swept surfaces in Computer Aided Geometric Design, *Proceedings of the International Workshop New approaches, Algorithms and Advanced Computational Techniques in Approximation Theory and its Applications*, Ed. Univ. Lucian Blaga, Sibiu, pp. 19-31, 2007.
[7] D. E. Ulmet, Customized Reflection Lines for Surface Interrogation in Car Body Design, *Proceedings of the SYNASC,* Timisoara, 2007
[8] eLearn Magazine, ACM, http://www.elearnmag.org/subpage.cfm?section=archives
[9] Journal of e-Learning, http://www.ejel.org/index.htm
[10] http://en.wikipedia.org/wiki/E-learning#Computer-based_learning

CORINA SIMIAN
Zurich University
Institute for Mathematics
Winterthurerstrasse 190, CH-8057 Zurich
SWITZERLAND
E-mail: corina.simian@math.uzh.ch,
corinafirst@yahoo.com