ONE METHOD FOR REPRESENTING AN ARC OF ELLIPSE BY A NURBS CURVE

Conference Paper · January 2005

DOI: 10.13140/RG.2.1.4541.6403

2 authors, including:

Emiliyan Petkov
University of Veliko Tarnovo
31 PUBLICATIONS **33** CITATIONS

SEE PROFILE

ONE METHOD FOR REPRESENTING AN ARC OF ELLIPSE BY A NURBS CURVE

Emiliyan Petkov, Liuben Cekov

University of Veliko Turnovo, Computer Systems and Technologies Department, Veliko Turnovo 5000, 2 Teodosii Turnovski str., tel.062 649831, e-mail: epetkov@abv.bg
Technical University - Gabrovo, Computer Systems and Technologies Department, Gabrovo 5300, 4 Hadgi Dimitar str., tel. 066 223456, e-mail: cekov@tugab.bg

Abstract: This paper offers one method for constructing an arc of ellipse – one of the conic sections, by NURBS – one of the most popular mathematical model for representing curves and surfaces in CAGD. It also gives the analytical equations of this conic and explains its relation with a quadratic rational Bezier curve. The developed technique is given by mathematically proved algorithms.

Key words: Computer Graphics, CAD/CAM, CAGD, Approximation, Bezier, NURBS

Introduction

Computer-Aided Geometric Design (CAGD) is the base of projection and construction in a number of industrial settings. CAGD is concerned with representations, constructions, deformations and approximations of curves and surfaces. There is an aspiration for opening new and more efficient tools for expanding the functionality of the graphics systems.

We have a goal to provide new possibilities and bigger flexibility in designing graphic models of real objects by investigation of new methods, algorithms and graphics tools. Because very often the borders of the objects have forms of conic sections, we concentrate our work on building methods allowing interactive modeling by ellipses, parabolas and hyperbolas, as well as ellipsoids, paraboloids, hyperboloids and etc. So, this task is a part of a bigger research for representing and constructing the all quadratic curves and surfaces by appropriate models and techniques. They should be prepared for implanting in graphics systems.

In this report we examine one of these curves – *the ellipse*, and with parabola and hyperbola we do the same in other papers.

One of the most popular mathematical model in CAGD for representing curves and surfaces is NURBS (Non-Uniform Rational B-Spline). That is why we want to represent the ellipse by a NURBS curve.

The mathematical base of the problem is given in section 1 and 2 and it is based on the references [3,5,7]. In section 3 we publish the result of our research.

1. Mathematics background

NURBS curves is a modification of the rational *B*-spline curves. They are obtained when special set of knots are chosen. The knots form a numerical vector (known as *knot-vector*) and they are necessary to define the *B*-spline function. Every curve approximates points in 2D or 3D. These points are called *control points* and form the *control polygon*. There is a relation between Bezier-Bernstein basis and *B*-spline basis as well.

1.1 General model of a rational Bezier curve

Here is the parametric equation of one rational Bezier curve of degree r when $r+1$ control points $M_i\left(x_i, y_i\right)$ and their weights w_i are given:

$$\beta(t)=\frac{\sum_{i=0}^{r} M_i w_i B_{i,r}(t)}{\sum_{j=0}^{r} w_j B_{j,r}(t)} \tag{1.1}$$

and $B_{i,r}(t)=\binom{n}{i} t^i\left(1-t\right)^{r-1}, i=0,\ldots,n$ and $t\in[0,1]$ is the polynomial of Bernstein.

$\beta(t)$ passes trough the first and the last control points and the tangents in these points have the directions of the first and the last lines of the control polygon [4].

1.2 General model of a NURBS curve

In [5, p. 55] is given definition of a *B*-spline function of degree r. But the most often used are the *normalized B-splines*:

$$N_{i,r}(t)=\frac{t-t_i}{t_{i+r}-t_i}N_{i,r-1}(t)+\frac{t_{i+r+1}-t}{t_{i+r+1}-t_{i+1}}N_{i+1,r-1}(t),$$

$$N_{i,0}(t)=\begin{cases}1 & t_i\le t\le t_{i+1}\\ 0 & t\notin[t_i,t_{i+1}]\end{cases} \tag{1.2}$$

Now one rational B-spline curve defined upon non-uniform knot-vector in homogenous coordinates is given by

$$\gamma(t)=\frac{\sum_{i=0}^{r-1} M_i w_i N_{i,r}(t)}{\sum_{j=0}^{n-1} w_j N_{j,r}(t)} \tag{1.3}$$

where $M_i\left(x_i, y_i\right)$ are control points with weights $w_i, i=0,\ldots,n$, t_j are $m+1$ knots for $t_j\le t_{j+1}, j=0,\ldots,m-1$ and $N_{i,r}(t)$ is the B-spline function given by Eq. (1.2).

The definition, all characteristics and geometric properties of the NURBS curves can be seen in [3]. Only the most

important property concerning this research will be given here.

Property 1.1 NURBS curve with no interior knots is a rational Bezier curve, since the $N_{i,r}$ reduce to $B_{i,r}$. NURBS curves contain non-rational *B*-spline and non-rational Bezier curves as special cases.

The advantage of using rational curves is that they interpolate conic sections correctly [1,2,3,4]. It means that by these curves correct representation of ellipses, hyperbolas and parabolas is possible.

1.3 Analytical representation of an ellipse

The *conic sections* are the non-degenerate curves generated by the intersections of a plane with one or two nappes of a cone. These objects are very good examined [7]. They are specified in three types of curves: *elliptical*, *parabolic* and *hyperbolic*.

The analytical representation of an *ellipse* in standard position into orthogonal coordinate system $Oxy = \left\{O, \vec{e_1}, \vec{e_2}\right\}$ is given by the following equations

$$\varepsilon : \frac{x^2}{a^2} + \frac{y^2}{b^2} = 1, \quad a, b > 0, \tag{1.4}$$

$$\varepsilon : \begin{cases} x = a.\cos\alpha \\ y = b.\sin\alpha \end{cases}, \ \alpha \in [0; 2\pi]. \tag{1.5}$$

2. Representing an arc of ellipse by a Bezier curve

There are a lot of techniques for building an arc of ellipse in CAGD, but one of the most popular belongs to Farin Gerald [1] and Les Piegl [3]. They represent the arc of conic (that is defined by its endpoints and the tangents at them plus an intermediate point) by rational Bezier curve.

They examine the quadratic rational Bezier curve for representing an arc of conic by looking at Eq. (1.1). When $n = 2$ all conic sections could be represented because they are quadratic curves as well. The parameters which influence the form of the cure (when $n = 2$) are $w_i, i = 0,1,2$. Usually, $w_0 = w_2 = 1$ are selected. It is called *normal parameterisation*. Now, the form of the curve depends on $w_1 \in (-\infty; +\infty)$. If $-1 < w_1 < 1$ the form of the curve is an arc of ellipse. For the prove of it, you can look at [3, p. 293].

Varying w_1 yields a family of conic arcs and one more convenient way to select a conic form is to specify a third point on the conic, which is attained at some parameter value $t = 0.5$. This point is called the *shoulder point* of the conic (Figure 1a). Substitution of $t = 0.5$ into Eq. (1.1) yields

$$S = \frac{1}{1+w_1} P + \frac{w_1}{1+w_1} M_1 \quad , \tag{2.1}$$

where P is the midpoint of the cord M_0M_2. Let s be a new parameter that gives a linear interpolation between P and M_1. Then for some value of s we have

$$S = (1-s)P + sM_1 \quad \Rightarrow \quad s = \frac{w_1}{1+w_1} \quad \Rightarrow \quad w_1 = \frac{s}{1-s} \tag{2.2}$$

A circular arc of sweep angle les than 180° is also represented by Eq. (1.1). For symmetry $M_0M_1M_2$ (Figure 1a) must be an isosceles triangle, with $|M_0M_1| = |M_1M_2|$. Let $\theta = \measuredangle M_1M_2P$. From symmetry the arc M_2S is the same as SM_0, hence the angle $\measuredangle SM_2P$ bisects θ. From Eq. (2.2) and the properties of bisectors it follows that

$$w_1 = \frac{s}{1-s} = \frac{|PS|}{|SM_1|} = \frac{|PM_2|}{|M_1M_2|} = \frac{e}{f} = \cos\theta \tag{2.3}$$

3. Building the arc of ellipse by a NURBS curve

On the base of the mathematics background we did above, the representation of an arc of ellipse we shall do by finding the right place and order of the control points in the quadratic NURBS model of a curve.

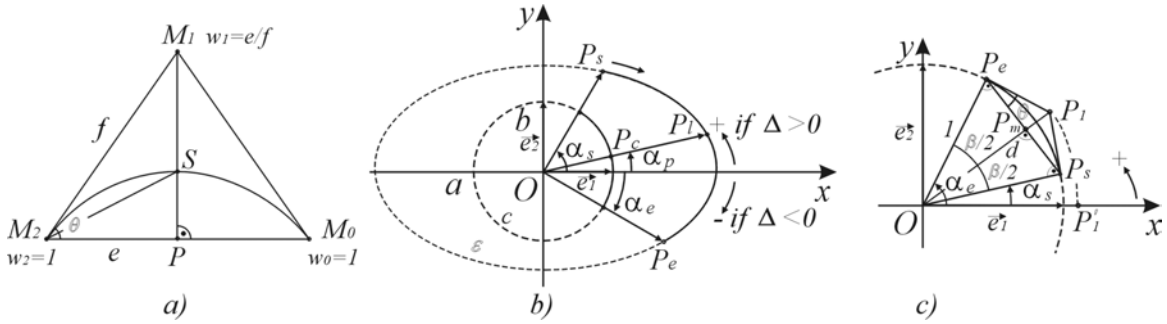

Fig. 1 – a) representing an arc of circle by a Bezier curve; b)an arc of the ellipse ε; c)an arc of the circle *c* represented by a Bezier curve

The conics always lie in one plain. So, let l be a plain and $K_l = \left\{O, \vec{e_1}, \vec{e_2}\right\} \left(\vec{e_1} = \vec{e_2}\right)$ a right oriented orthogonal coordinate system in it. Therefore, the couple $\left(\vec{e_1}, \vec{e_2}\right)$ characterizes positive direction of visiting the points of l. Let ε given by Eq. (1.4) be in standard position (Figure 1b).

3.1 Determining the direction of the arc

For Eq. (1.5) we have a requirement $\alpha \in [0; 2\pi]$, but we could release α to vary in the range $(-\infty; +\infty)$.

Therefore, varying α in the range $[\alpha_s;\alpha_e]$, for $\alpha_s,\alpha_e \in \mathbb{R}$, yields (repeatedly) turning the radius-vector (when $|\alpha_s-\alpha_e|>2\pi$) that builds the ellipse when α changes its value from α_s to α_e.

Let α_s determine a point $P_s(x_s,y_s)$ from ε with a radius-vector $\vec{r_s}$, and α_e determine a point $P_e(x_e,y_e)$ from ε with a radius-vector $\vec{r_e}$. We know (from [7], pp. 22-24), that 1) the vectors $\vec{r_s}$ and $\vec{r_e}$ are lineal independent $\Leftrightarrow \Delta=\begin{vmatrix} x_s & y_s \\ x_e & y_e \end{vmatrix}\neq 0 \Leftrightarrow \operatorname{rang}\begin{pmatrix} x_s & y_s \\ x_e & y_e \end{pmatrix}=2$ and 2) the couple $(\vec{r_s},\vec{r_e})$ is right and characterizes positive direction of visiting $\Leftrightarrow \Delta>0$, and it is left and characterizes negative direction of visiting $\Leftrightarrow \Delta<0$.

Therefore, if $\alpha_s<\alpha_e$ we have positive direction of visiting the points of the arc, if $\alpha_s>\alpha_e$ we have negative direction, and if $\alpha_s=\alpha_e$ - the arc does not exist. P_s we shall call *start point*, P_e - *end point,* α_s - *start angle* and α_e - *end angle*.

Heuristic 3.1 *Positive arc* from ε (Figure 1b) we shall call the curve obtained by the movement of radius-vector $\vec{r}$ in positive direction from P_s to P_e when $\Delta>0$ for $\alpha\in[\alpha_s;\alpha_e]$. *Negative arc* from ε we shall call the curve obtained by the movement of radius-vector $\vec{r}$ in negative direction from P_s to P_e when $\Delta<0$ for $\alpha\in[\alpha_s;\alpha_e]$.

3.2 The problem

A continuous arc γ_ε from ε to be represented by a NURBS curve and to be determined by *a*, *b*, α_s and α_e. If $\alpha_s<\alpha_e$ the arc to be built in positive direction, if $\alpha_s>\alpha_e$ - in negative direction, and if $\alpha_s=\alpha_e$ - the arc is not built.

3.3 The main idea for solving the problem

First, we shall examine an arc γ_c given by α_s and α_e from the unit circle c (Figure 1b). We shall find its representation by NURBS curve. After that, by a suitable affine transformation of this curve, we shall find the correct representation of γ_ε which is also determined by α_s and α_e, but it belongs to the ellipse ε.

3.4 Transforming the unit circle c to the ellipse ε

Let α_p determine the points: P_c from the unit circle c and P_l from the ellipse ε. Therefore, relying on Eq. (1.5), $P_c:\begin{cases} x_c=\cos\alpha_p \\ y_c=\sin\alpha_p \end{cases}$, $P_l:\begin{cases} x_l=a\cos\alpha_p \\ y_l=b\sin\alpha_p \end{cases}$ and the point P_l could be obtained from the point P_c by the following transformation:

$$P_l=TP_c,\quad T=\begin{pmatrix} a & 0 \\ 0 & b \end{pmatrix} \tag{3.1}$$

3.5 Building the arc γ_c by using the unit circle c

We shall give a technique for constructing an arc defined by α_s and α_e ($\alpha_s,\alpha_e\in(-\infty;+\infty)$) of the unit circle c. We set the condition $\alpha_s<\alpha_e$, i.e. for building only a positive arc. The whole arc will be divided into segments, everyone of them will be not more than 90°.

Lets first examine the special case when $|\alpha_s-\alpha_e|\le\pi/2$, looking at a Figure 1c. We know that $|OP_s|=|OP_e|=1$. Let the tangents at P_s and P_e intersects themselves at P_1. Then P_s,P_1,P_e are the control points of the quadratic rational Bezier curve that interpolates the arc precisely. $P_s,P_e\in\varepsilon \Rightarrow P_s:\begin{cases} x_s=a\cos\alpha_s \\ y_s=b\sin\alpha_s \end{cases}$ and $P_e:\begin{cases} x_e=a\cos\alpha_e \\ y_e=b\sin\alpha_e \end{cases}$. Let w_s,w_1,w_e be the weights of those three points. We want $w_s=w_e=1$ - for normal parameterisation. Now, it is easy to calculate P_1. We obtain $P_1:\begin{cases} x_1=\cos(\alpha_s+\beta/2).d \\ y_1=\sin(\alpha_s+\beta/2).d \end{cases}$. For finding w_1 we do the following:

Let $P_sP_e\cap OP_1=P_m$ and $\sphericalangle P_1P_eP_m=\theta$ (Figure 1c), then from Eq. (2.3) $\Rightarrow w_1=\dfrac{|P_eP_m|}{|P_eP_1|}=\cos\theta$. From $\triangle OP_eP_1$ and $\triangle P_eP_mP_1$ we have: $\sphericalangle OP_eP_1=\sphericalangle P_eP_mP_1=90^\circ$ and $\sphericalangle OP_1P_e=\sphericalangle P_eP_1P_m \Rightarrow$ $\sphericalangle P_eOP_1=\sphericalangle P_1P_eP_m=\beta/2=\theta$. Therefore, $w_1=\cos\beta/2$.

P_s,P_1,P_e are the control points with the weights w_s,w_1,w_e that we pass to Eq. (1.1) for $n=2$ to obtain the rational Bezier curve.

Now, we give the Algorithm 3.1 for building the whole arc γ_c by NURBS curve, using several small segments which is not more than 90°. We could do that relying on Property 1.1.

Algorithm 3.1

ComputeNURBSCircularArc(α_s, α_e, segments, aP_c [], nNP, aK [], nNK) {

```
//input: α_s – start angle, α_e – end engle, segments – number of
segments of the whole arc
//output: aP_c – array of the control points, nNP – number of the
control points,
//aK – knot-vector, nNK - number of knot-vector values
//narcs –number of the arcs of the whole arc
//w1 - weight of the mid-points (P1 in fig. 3.1b) of every segment
buf : = α := |α_s − α_e| ; narcs := 0;
while( buf > 0.0 ) {        narcs++;        buf -= π / 2;                }
if( 0 < narcs && narcs < segments )  narcs := segments;
if( narcs > 0 ) {
   β := α / narcs; nNP := 2 * narcs + 1; w1 := cos( β / 2 );
   aP_c [ 0 ].x := cos( α_s ); aP_c [ 0 ].y := sin( α_s );
   aP_c [ 0 ].w := 1.0;  index := 0; angle := α_s ;
   d := 1 / cos( β / 2 );
   //computes every tree control points of every segment
   for( int i := 1; i <= narcs; i++ ) {
      aP_c [index+1].x := cos(angle+ β / 2 ) * d;
      aP_c [index+1].y := sin(angle + β / 2 ) * d;
      aP_c [index+1].w:=w1; aP_c [index+2].x:=cos(angle+ β );
      aP_c [index+2].y := sin(angle + β );
      aP_c [index+2].w := 1;
      index += 2; angle += β ; }
   nNK := nNP + 1; step := 1.0; g := 0.0;
   //computes the step of the knot-vector values:
   if( nNK > 4 ) step := 1 / ( (nNK - 4) / 2 + 1 );
   //computes the knot-vector values:
   for( i := 0; i <= nNK-1 ; i+=2 ) {
      aK [ i ] := aK [ i+1 ] := g; g += step; }          }
};
```

After executing the Algorithm 3.1 the control points are obtained in $aP_c[]$ and the values of the knot-vector in $aK[]$, which are necessary for building the NURBS curve by Eq. (1.3). That curve interpolates γ_c precisely.

3.6 Representing the arc γ_ε by a NURBS curve

After we have the representation of the arc γ_c by NURBS curve, we could transform (relying on Eq. (3.1)) the control points aP_c obtaining aP_ε (Figure 2a). Also, we have to conform the direction of the given arc. Algorithm 3.2 contains these steps and constructs the NURBS elliptical arc we are seeking for.

Algorithm 3.2

```
ComputeNURBSEllipticalArc( a, b, α_s , α_e , segments, aP_ε [],
                                   nNP, aK[], nNK ) {
//input: a, b, α_s , α_e , segments; //output: aP_ε , nNP, aK , nNK;
   if( α_s > α_e ) { β_s := α_e ; β_e := α_s ; bAnglesEx := true; }
   else { β_s := α_s ; β_e := α_e ; bAnglesEx := false;};
   ComputeNURBSCircularArc( β_s , β_e , segments, aP_c ,
                                   nNP, aK, nNK );
   if( bAnglesEx ) ReverseOrderOfTheControlPoints( aP_ε );
   //transforms every control point of c to a control point of ε :
   for( i := 0; i <= nNP-1 ; i++ )  {
      aP_ε [ i ].x *= a; aP_ε [ i ].y *= b;  }
};
```

After executing the Algorithm 3.2 the control points are obtained in $aP_\varepsilon[]$ and the values of the knot-vector in $aK[]$, which are necessary for building the NURBS elliptical arc by Eq. (1.3). That curve interpolates γ_ε precisely. Some examples could be seen in a Figure 2 b), c).

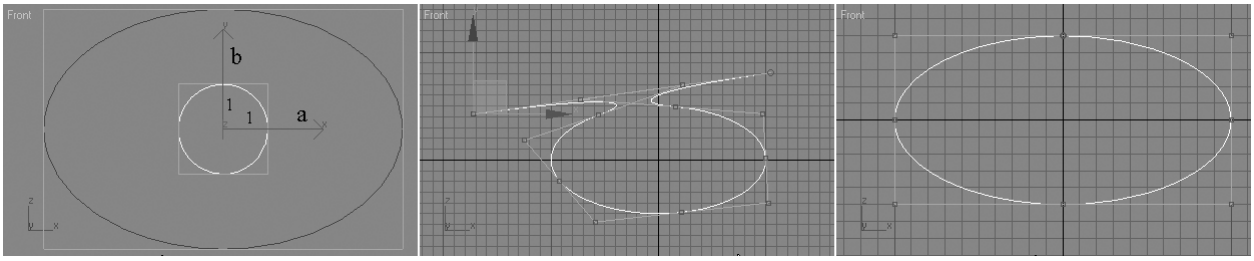

Fig. 2 – a) full ellipse; b) from -70° to 405° arc and the positions of the end points are changed; c) from 0° to 360° arc

Conclusion

We explained one technique for constructing NURBS elliptical arcs, that is useful for the computer-aided geometric design. It makes the computer algorithms for representing this kind of curves effective and fast. The algorithms are developed to represent the conic of the lowest degree NURBS curve and fewest control points. For more flexibility increasing the number of the control points and constructing arcs for angles less than 0° and more than 360° is possible. So designers could model elliptical forms with ease.

In addition, a programming model for NURBS conic curves could be seen in [6].

Our work is being expanded with developing new effective techniques for representing an arc of parabola and an arc of hyperbola by NURBS curves. In this way we shall have all three generatrix curves that we need for representing the nine quadratic surfaces by NURBS.

References

1. Farin Gerald. From Conies to NURBS: A Tutorial and Survey. IEEE CG&A, pp. 78-86, 1992.
2. Lee E. T. Y. Rational quadratic Bezier representation for conics. Geometric Modelling: Algorithms and New Trends. Philadelphia: SIAM, pp. 3-19, 1987.
3. Piegl Les, W. Tiller. The NURBS Book. Springer, USA, 1997.
4. Rogers David F., J. Alan Adams. Mathematical Elements for Computer Graphics, 2nd edition. McGraw-Hill, 1989.
5. Боянов Борислав. Лекции по числени методи. Дарба, С., 1995.
6. Петков Емилиян. NURBS коники – програмен модел. Международна научна конференция "УниТех'04", Габрово. Сборник доклади, Том 1, стр. 359-363, 2004.
7. Христов Милен. Аналитична геометрия. Астарта, Велико Търново, 2003.