TSINGHUA SCIENCE AND TECHNOLOGY
ISSN 1007-0214 07/21 pp316-322
Volume 10, Number 3, June 2005

Approximation of NURBS Curves and Surfaces Using Adaptive Equidistant Parameterizations*

Aziguli Wulamu (阿孜古丽·吾拉木)[1,2,**], GOETTING Marc[3], ZECKZER Dirk[3]

1. School of Information Engineering, University of Science and Technology Beijing, Beijing 100083, China;
2. School of Information Science and Engineering, University of Xinjiang, Urumq 830046, China;
3. Department of Computer Science, University of Kaiserslautern, D-6753, Germany

Abstract: Non-uniform rational B-spline (NURBS) curves and surfaces are very important tools for modelling curves and surfaces. Several important details, such as the choice of the sample points, of the parameterization, and of the termination condition, are however not well described. These details have a great influence on the performance of the approximation algorithm, both in terms of quality as well as time and space usage. This paper described how to sample points, examining two standard parameterizations: equidistant and chordal. A new and local parameterization, namely an adaptive equidistant model, was proposed, which enhances the equidistant model. Localization can also be used to enhance the chordal parameterization. For NURBS surfaces, one must choose which direction will be approximated first and must pay special attention to surfaces of degree 1 which have to be handled as a special case.

Key words: computer graphics; computer-aided design (CAD); computer-aided geometric design (CAGD); NURBS curves; NURBS surfaces; approximation; implementation

Introduction

Curves and surfaces can be represented using coefficients and a set of basis functions[1]. In the following, only curves are considered, but all statements apply equally to surfaces. Different representations use different sets of basis functions. These basis functions can be, for example, polynomials for polynomial curves, Bernstein-polynomials for Bézier curves, or B-spline basis functions for B-spline curves. As long as the general representation is of the form:

$$C(u)=\sum_{i=0}^{n} a_i \cdot p_i(u),$$

where a_i is the coefficient and $p_i(u)$ the basis function. One of these representations can be converted to another in a more or less straightforward manner.

Unfortunately, these representations are not sufficient for all cases. Consequently, rational Bézier and B-spline curves were developed. These curves can be used to represent such curves as circles, which is otherwise not possible using representations without a rational part. Non-uniform rational B-spline (NURBS) curves are generalized rational B-spline curves, while NUBS curves are B-spline curves without rational parts, i.e., all weights are equal to 1.

NURBS curves and surfaces are used in computer-aided design (CAD) and computer-aided geometric design (CAGD) not only to describe mechanical parts but also to determine offset curves and surfaces[2] and curves and surfaces bases for various measurements[3], e.g., in the automotive industry. NURBS curves and surfaces are also used for approximating various other

Received: 2004-06-06
* Supported by the Company ProCAEss GmbH, Landau in der Pfalz, Germany
** To whom correspondence should be addressed.
E-mail: azgl2003@sohu.com

curves, e.g., circles[4] or multiple curves given by B-spline curves having different knot vectors[5]. Further, NURBS curves and surfaces are used in computer animation to describe such things as objects and camera trajectories.

In CAD, different CAD-systems use different representations and thus converting data from one CAD-system to another implies converting one curve representation to another. In fact, there are also several independent data formats used to exchange data between CAD-systems, each of which uses different representations for curves.

Converting any curve without rational part to a NURBS curve is quite straightforward. First, the curve is converted into a non-uniform B-spline (NUBS) curve using a basis transformation. Then, the weights are added such that all weights are equal to 1.

To convert NURBS curves to polynomial curves, two cases must be considered. The first case is where all weights of the NURBS curve are equal to 1. The curve is in fact a NUBS curve which can directly be converted using well-known algorithms[1,6-9]. However, if there is at least one weight having a value different from 1, the curve can no longer be converted directly. In such case, the curve must then be approximated[1,10].

While there exist several algorithms which deal with this approximation case, only the cores of the algorithms are described. Several important details are not usually described and must be determined by those trying to implement the approximation algorithm. The following will analyze these difficulties, describe how the required details can be chosen, and analyze what impact those choices have on the behaviour of the algorithm.

1 Approximation of NURBS Curves

1.1 Definition

A detailed introduction to non-uniform rational B-spline curves can be found in Ref. [1]. Non-uniformity is a property of the knot vector where two knots need not have the same distance; B-spline refers to the kind of basis functions used (see Ref. [1]).

A B-spline curve of degree p is defined by n+1 control points P_i, i=0, …, n with knot vector $\boldsymbol{U}$ of length $m = n + p + 2$:

$$\boldsymbol{U}=(0, \dots, 0, u_{p+1}, \dots, u_n, 1, \dots, 1).$$

In fact, we have p+1 zeros and p+1 ones at the beginning and the end of the knot vector, respectively. This leads to end point interpolation, i.e., P_0 is the first point and P_n is the last of the resulting curve. The knots determine which control points influence which part of the curves and thus determine the boundaries of different segments. The control points P_i form the control polygon which has a bounding box property.

Let $N_{i,p}$ be the respective B-spline basis function. Then, the NUBS curve $C(u)$ is given by

$$C(u)=\sum_{i=0}^{n} P_i \bullet N_{i,p}(u),$$

while the NURBS curve $C^w(u)$ is given by

$$C^w(u)=\frac{\sum_{i=0}^{n} P_i \bullet N_{i,p}(u)}{\sum_{i=0}^{n} w_i \bullet N_{i,p}(u)},$$

where w_i is the weight.

1.2 Approximation

There are two obvious possibilities for the choice of the basis functions used to approximate a given NURBS curve. The first one is to use a curve representation where the basis is the same as that in the final representation, e.g., polynomials. Another one is to use NUBS to approximate NURBS. The second one has several advantages: 1) the NUBS representation gives a lot of control over the shape of the resulting curve; 2) there exist already some algorithms which can be used to perform this approximation; 3) we do not need to worry about segments or the degree of the polynomials; 4) we can transform a NUBS curve directly into a number of Bézier curves of a given degree and these Bézier curves can then be converted to the polynomial curves. Thus, the second approach is chosen here.

Further, we have the choice between two types of approximation algorithms, namely local and global ones. Local ones give better results for local segments of curves that are difficult to handle. Furthermore, the amount of data used for the approximations is smaller. However, special handling is needed to join the local curves with given constraints. Global methods require the handling of larger sets of data, but they

can treat the curve as a whole. Moreover, as will be shown later, carefully chosen parameterizations can be used to recover a local influence as well.

Thus, the least-squares method was chosen as the approximation algorithm to be implemented. A detailed description of this algorithm can be found in Ref. [1].

1.3 Determination of the sample points and the parameterization

The least-squares method is a discrete algorithm based on a number of sample points from the original curve and tries to minimize the difference between these sample points and the respective points on the approximating curve.

First, a certain number of sample points together with an initial parameterization are chosen. As the original curve is in parameter form, in fact, a set of parameters t_i, $i = 1, \ldots, m$ are chosen first. Evaluating the curve at these parameters yields the control points $Q_i = C(t_i)$ on the curve $C(t_i)$. The least-squares algorithm then works by determining the approximating curve $C'(t_i)$ such that the following sum is minimized:

$$\sum_{i=0}^{m}(Q_i' - Q_i)^2 ,$$

where $Q_i' = C'(t_i)$.

Therefore, first of all, the initial parameters have to be chosen. In order to solve the minimization problem, a matrix is used. A sufficient number of parameters is needed to obtain a nonsingular matrix. If too many parameters are used, the algorithm becomes too computationally expensive. Thus, the minimum number of parameters required to obtain good results should be used to keep the computational costs low.

The number of initial parameters was empirically determined as $12n$. Further, these parameters were chosen to be equidistant.

The second problem is the type of parameterization. After having computed the sample points from the initial set of parameters, there are two possibilities for the parameterization. One is the equidistant parameterization, in which case we can use the initial parameterization. The other is the chordal parameterization, and is preferred in many CAD applications. In this case, we have to compute the chordal parameterization.

Next, one approximation step is performed using the least-squares algorithm. If the approximation is not good enough after this step, new control points are inserted. The addition of new control points can lead to a singular matrix; in this case, new sample points and parameters also have to be added. New parameters are only required for those parts of the curve where new control points are inserted: if we simply increase the number of the initial parameters and spread them equidistantly over the curve, parts of the curve which are not influenced by the new control points will also get more parameters, although they are not needed there. This is also the case if afterwards we compute the chordal parameterization of the sample points associated with this new initial parameterization.

Also, we can achieve a local control if we do not spread the new initial parameters over the whole curve. When introducing new control points, we only add new parameters there, if necessary and then only to avoid a singular matrix. Thus, the parts posing no problems with respect to the approximation will be unaltered, whereas the parts where new control points were added will be given new parameters if necessary. We call this type of parameter insertion adaptive equidistant parameterization.

One problem for the adaptive equidistant parameterization is the choice of the number of parameters to be inserted. While the number of iterations might be smaller if more parameters are inserted, the total number of parameters increases. Thus, even with a smaller number of iterations, the costs in terms of computation time and space needed might increase. However, more parameters inserted do not mean less iterations. Experiments have shown that the optimal number of parameters to be inserted to get a minimal number of iterations varies from curve to curve. Values of around 8 seem, however, to yield good results both with respect to the number of iterations and the number of parameters (sample points).

Unfortunately, adding parameters locally is not as easily possible for chordal parameterizations. In fact, for these, we always have two parameterizations: the initial, equidistant one to determine the sample points, and the resulting chordal parameterization of these sample points. For each parameter interval indicating where to insert new sample points, we have to map this parameter interval back to the initial parameterization

used before the approximation step to determine the new sample points and the new initial parameterization for the next step in this case. Therefore, the algorithm needed for local chordal parameterization is much more complicated. In fact, chordal parameterization already yields the best results. The second best method is adaptive equidistant (local), while the equidistant parameterization is the worst.

As an example, a curve is given in Fig. 1, together with its control polygon. In Fig. 2, the same curve is shown together with the sample points required based on an equidistant initial parameterization.

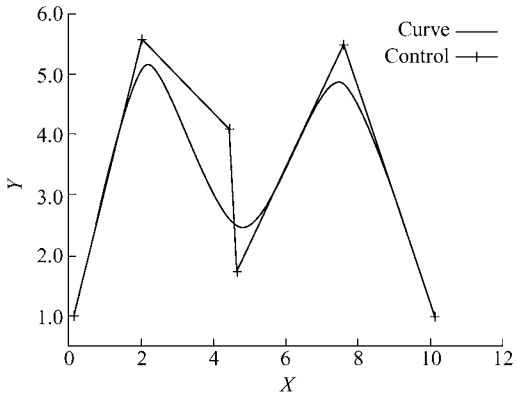

Fig. 1 An example curve together with its control polygon

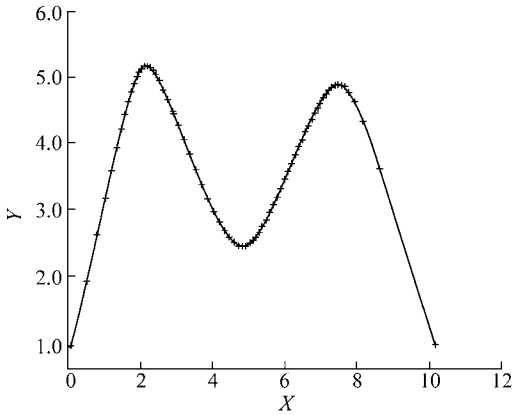

Fig. 2 Sample points based on an equidistant initial parameterization

Figure 3 gives the maximal error after each iteration for the curve presented in Fig. 1 using the three parameterizations presented in this section. A logarithmic scale is used to display better the differences between the parameterizations. The termination condition was chosen to be: maximal error $<1\times10^{-4}$. Using the chordal parameterization, the algorithm terminates after 7 iterations; using the adaptive equidistant one, it terminates after 8 iterations; using the equidistant parameterization, it only terminates after 12 iterations. For the adaptive equidistant parameterization, 8 new parameters and control points were added where needed to prevent the matrix from becoming singular. The maximal error after each iteration is the smallest using the chordal parameterization, except for at the starting point. The adaptive equidistant parameterization is the second best, while using the equidistant parameterization is worst. This observation holds for all examples tested. In Fig. 4, results are shown using a normal scale for the maximal error.

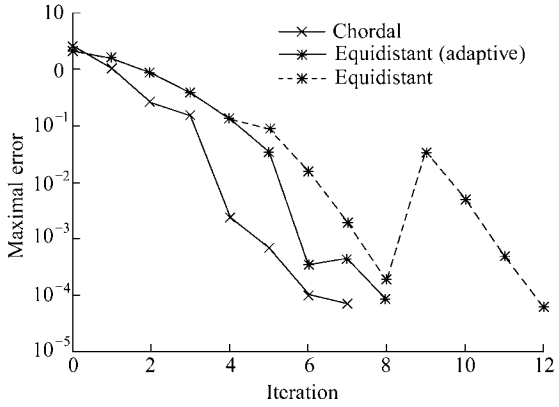

Fig. 3 Maximal error for different parameterizations (logarithmic scale)

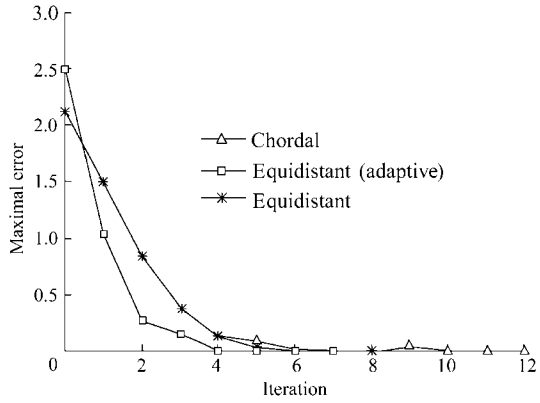

Fig. 4 Maximal error for different parameterizations (normal scale)

1.4 Determination of the error

Another important choice is to determine the error between the original and the approximated curve. The simplest approach is to choose an oversampling parameterization, for example, choose three times as many parameters equidistantly, determine the respective points for both curves, and then compute the sum of the squared distances (section 0). In fact, any

number of parameters can be chosen, but at least as many are required as for the least-squares algorithm.

Another method uses a projection[1]. This method is more difficult to implement and requires more resources, but yields better results. However, because of the complexity of its implementation, the projection method is not considered further here. The crucial choice in both cases is the number of points on the curve required to determine the error between the original and the approximating curves.

1.5 Termination

Perhaps the most difficult thing to determine is when to terminate the approximation process. A simple approach would be to choose some predefined limit for the error: if the error determined (section 0) is less than this chosen limit, the approximation will be terminated and the approximating curve will be considered to be good enough. Unfortunately, the choice of limit is not obvious, due to the fact that we do not know beforehand which scale is used. If we choose a limit of d_1 = 0.001, for example, this might be appropriate for curves in a [−5, 5] × [−5, 5] × [−5, 5] space. If, however, the scale is larger, e.g., [−100,100] × [−100,100] × [−100,100], this limit may no longer be appropriate and may be far too small. Conversely, if a smaller scale is used, e.g., $[-1\times10^{-9}, 1\times10^{-9}] \times [-1\times10^{-9}, 1\times10^{-9}] \times [-1\times10^{-9}, 1\times10^{-9}]$, the limit may be much too large. Thus, we have to consider the limit as a parameter in the approximation algorithm.

Other possible termination conditions include:

- When the number of iterations is greater than N_I.
- When the number of control points is greater than N_C.
- When $\frac{d_i}{d_{i+1}}<1$, where d_i is the maximal error in step i.

Unfortunately, none of these termination conditions can be considered as realistic alternatives. The determination of N_I and N_C is even more difficult than the determination of d_i. Moreover, the third condition, which depends on the maximal error ratio between two consecutive steps, might be fulfilled even if it would be better to continue (Fig. 3). For chordal parameterization, the maximal error decreases monotonously. However, this is not the case for both equidistant parameterizations. In the example previously given for the adaptive equidistant parameterization, the maximal error after iteration 7 is greater than after iteration 6. Nevertheless, after iteration 8, it falls below the limit chosen. For the equidistant parameterization, the maximal error after iteration 8 is worse for all iterations but the last one. In fact, only after the last iteration is the maximal error below the limit chosen. Thus, no better termination condition than using d_1 as a limit was found in the literature.

2 Comparison and Examples

In this section, we present the example results for the evaluation of the three parameterizations presented in Section 1.3: equidistant, adaptive equidistant, and chordal.

The first curve chosen is depicted in Fig.1. The knot vector is (0, 0, 0, 0, 0.3125, 0.625, 1, 1, 1, 1); the six control points have the weights (1, 1.82, 0.1, 1.16, 1.9, 0.16). The effects of the weights can be seen in Fig. 2: if an equidistant parameterization is used, then the respective points on the curve are very dense if the weight is high and they are very sparse if the weight is low. In particular, between the 4th and the 5th control points there are many samples, whereas between the 5th and the 6th there are only two.

Table 1 summarizes the results of an approximation of this curve using the equidistant and the adaptive equidistant parameterizations. The termination condition was chosen to be: maximal error $< 1\times10^{-4}$. The number of control points after each iteration is given for each parameterization.

Table 1 shows that the adaptive equidistant parameterization needs fewer iterations, namely 8 compared to the 12 iterations needed for the equidistant parameterization. Furthermore, the number of control points is much lower: 69 compared to 97. A total of only 120 parameters are generated for the adaptive equidistant parameterization compared to 972 parameters for the equidistant parameterization. Therefore, the computation of the approximating curve is much faster using the adaptive equidistant parameterization compared to the equidistant parameterization. During the first 5 iterations, no additional parameters were needed.

Table 1 Approximation results using the equidistant and the adaptive equidistant parameterizations

Iteration	Equidistant	Adaptive equidistant
1	9	9
2	15	15
3	27	27
4	50	50
5	56	56
6	69	65
7	76	67
8	79	69
9	81	
10	90	
11	94	
12	97	

Table 2 shows results for the adaptive equidistant parameterization when adding different numbers of parameters after each iteration. The entries give the number of control points.

Table 2 Adaptive equidistant parameterization with different number of parameters

Iteration	Number of parameters added			
	4	8	16	128
1	9	9	9	9
2	15	15	15	15
3	27	27	27	27
4	50	50	50	50
5	56	56	56	56
6	65	66	67	69
7	67	70	70	72
8	69		72	76

During the first 5 iterations, no new parameters are needed and thus the number of control points is the same for all four cases. After the 5th iteration, new parameters were added and the number of control points differs after the 6th iteration. Adding 8 new parameters results in the fewest number of iterations, but 70 control points are needed. Adding 4 new parameters results in 8 iterations, but only 69 control points. Those two are the best cases, as adding 16 or 128 new parameters both yield 8 iterations and 72 or 76 control points. The numbers of added parameters in total are 120 (4), 160 (8), 264 (16), and 1736 (128). Thus, adding only 4 parameters if an empty knot span is encountered, is the best one.

Finally, Table 3 shows the results for approximating a circle using all three parameterizations. The error bound was again chosen to be 10^{-4}. The entries give the number of control points.

Table 3 Approximating a circle using all three parameterizations

Iteration	Equidistant	Adaptive equidistant	Chordal
1	16	16	16
2	30	30	30
3	58	58	36
4	63	63	38
5	70		

Table 3 shows that the chordal parameterization is best, creating an approximating NUBS curve with only 38 control points, while the adaptive equidistant parameterization generates an approximating NUBS curve with more than 65% more control points; the approximating NUBS curve generated by the equidistant parameterization has 7 more control points. The chordal parameterization generates 108 new parameters compared to 116 generated by the adaptive equidistant one and 756 by the equidistant one. Thus, the chordal parameterization produces the best results compared to both the adaptive equidistant parameterization and the equidistant parameterization in this case.

3 Approximation of NURBS Surfaces

3.1 Definition

A NURBS surface is given with respect to a two-dimensional parameter space (u, v). Let p be the degree in the u-direction and q be the degree in the v-direction. For each dimension there is a separate knot vector. We can then consider a NURBS surface as a product of two NURBS curves. In fact, there are $(n+1) \times (m+1)$ control points $P_{i,j}$, and the basis functions are the products of the basis functions of the respective curves. We obtain isoparametric curves in the v-direction for fixed u, and in the u-direction for fixed v.

The NUBS surface $S(u, v)$ is then given by

$$S(u,v)=\sum_{i=0}^{n}\sum_{j=0}^{m}P_{i,j}\cdot N_{i,p}(u)\cdot N_{j,q}(v),$$

while the NURBS surface $S^w(u, v)$ is given by

$$S^w(u,v)=\frac{\sum_{i=0}^{n}\sum_{j=0}^{m}P_{i,j}\cdot N_{i,p}(u)\cdot N_{j,q}(v)}{\sum_{i=0}^{n}\sum_{j=0}^{m}w_{i,j}\cdot N_{i,p}(u)\cdot N_{j,q}(v)},$$

where w_i is the weight.

3.2 Computation of the approximation

NURBS surfaces can be approximated using NUBS surfaces. In fact, the algorithm used for approximating curves can be used as parts of the surface approximation algorithm. The difficulty is to determine whether to approximate first in the u- or in the v-direction: the results will differ from each other in general. It is possible in fact to find examples where each of the two possibilities yields better results. A simple example will be a cylinder. In one direction, the curves are simply lines; and in the other direction, they are circles. If u and v are exchanged, the results will also change.

3.3 Boundaries

Another problem with NURBS surfaces is the boundaries. With NURBS curves, the starting and end points of the curve can be interpolated. Thus, a continuous transition between two adjacent curves is always possible for the approximated curves, provided this was already the case for the NURBS curves. However, for faces only the four corner points of the surface have this property. The boundary curves themselves cannot be interpolated. In the example of a cylinder, two of the boundary curves are circles. These circles can only be approximated within a certain error boundary, and therefore, topological information is needed to prevent holes between two adjacent surfaces.

4 NURBS Curves and Surfaces with Degree 1

NURBS curves of degree 1 cannot be handled by the approximation algorithm. In fact, such curves consist of adjacent line segments. An initial parameterization of the curve will sample points anywhere on the curve and the end points of the respective line segments might not be included in this set of parameters, though they could be added. Thus, running an approximation algorithm does not make sense in this case. A better way is to compute the end points of the line segments and compute the NUBS curve directly from the resulting points. The same holds for NURBS surfaces with degree 1.

5 Summary

This paper presented some possible choices for approximating NURBS curves and surfaces together with an analysis for: 1) the approximation algorithm to be used; 2) the sample points to start with; 3) how to add new sample points; 4) which parameterization to choose; 5) the determination of the error; and 6) the termination condition. For NURBS surfaces, one has to choose which direction will be approximated first, and special attention has to be paid to boundaries. Finally, NURBS curves and surfaces with degree 1 have to be handled as a special case. The choices presented in this paper represent only a small part of the possible choices but are considered sufficient to demonstrate the problems involved.

References

[1] Piegl L, Tiller W. The NURBS Book, Second Edition. Heidelberg, Berlin, New York: Springer-Verlag, 1997.

[2] Kulczyka M A, Nachman L J. Qualitative and quantitative comparisons of B-spline offset surface approximation methods. *CAD*, 2002, **34**(1): 19-26.

[3] Brujic D, Ristic M, Ainsworth I. Measurement-based modification of NURBS surfaces. *CAD*, 2002, **34**(3): 173-183.

[4] Piegl L, Tiller W. Circle approximation using integral B-splines. *CAD*, 2003, **35**(6): 601-606.

[5] Park H, Kim K, Lee S-C. A method for approximate NURBS curve compatibility based on multiple curve refitting. *CAD*, 2000, **32**(4): 237-252.

[6] Hoschek J, Lasser D. Grundlagen der geometrischen Datenverarbeitung (Basics of geometric data manipulation). B.G. Teubner, Stuttgart, 1989.

[7] De Boor C. A Practical Guide to Splines. Amsterdam, Netherlands: Springer-Verlag, 1978.

[8] Farin G. NURBS: From Projective Geometry to Practical Use, Second Edition. Natick, Massachusetts, USA: AK Peters Ltd., 1999.

[9] Farin G, Hoschek J, Kim M-S. Handbook of Computer Aided Geometric Design. Amsterdam, Netherlands: Elsevier, 2002.

[10] Yang Huaiping, Wang Wenping, Sun Jiaguang. Control point adjustment for B-spline curve approximation. *CAD*, 2004, **36**(7): 639-652.