High Fidelity Conversion of NURBS Curves for Data Exchange

Les A. Piegl, Khairan Rajab and Volha Smarodzinava

University of South Florida, {lpiegl,khairanr,olyagrove}@gmail.com

ABSTRACT

This paper presents a method for converting a NURBS curve of any degree to a non-rational B-spline curve of any other degree. The conversion is parametric rather than geometric, i.e. the geometry as well as the parametrization are approximated. The algorithm contains straightforward computations only, i.e. it does not rely on discrete sampling, guess points or iterative techniques. The error is computed precisely using symbolic operators allowing the application of maximum as well as average error conditions. High fidelity approximations up to 10^{-10} relative tolerance have been computed in real time using standard windows work stations.

Keywords: NURBS, spline conversion, data exchange.

1. INTRODUCTION

The proliferation of CAD/CAM systems in manufacturing opened up unlimited opportunities for design and manufacturing in all segments in industry. The high level of enthusiasm about system design in the late seventies and early eighties resulted in the development of a large number of CAD systems, a dozen of which have penetrated practically all segments of the manufacturing enterprise. The sad part of the story is that although there are only a few systems with a large market share, they are largely incompatible, often times to the level of geometry as well as topology. One may find that a simple entity like the cylinder may have quite different representations making it impossible to pass data from one CAD vendor to the other, hence conversion of geometry from any format to any other format has become a common practice. This paper addresses the following problem. A NURBS curve is designed in system A and this is to be processed in system B. Convert the curve into a B-spline curve so that system B can process it without loosing accuracy as well as other relevant information, e.g. parametrization or continuity, that was introduced as part of the original design.

The conversion must satisfy the following requirements:

- The conversion must be parametrization based. A NURBS curve is a parametric entity whose geometry cannot be separated from its parametrization, e.g. a badly parametrized NURBS curve can cause severe problems down-stream when used to generate ruled or lofted surfaces, or when used for offsetting based on point sampling.
- The process must work for degree reduction as well as degree elevation. That is, even though degree elevation is a precise process, it may not be suitable for data conversion (see note below).
- The approximating curve is non-rational irrespective of whether the original curve was rational or non-rational. Rational curves tend to cause problems as mild as undesirable parametrization and as severe as discontinuities in subsequent surface constructions.
- The approximating curve of degree p must have full parametric continuity of C^{p-1}.
- The error must be guaranteed while maintaining a low number of control points. This means that upper bound techniques, that may result in a huge number of control points, and discrete sampling, that do not guarantee the quality of the approximation, are not allowed.

A few words about degree elevation. While it is a precise process, it suffers from two major shortcomings:

- It changes the form of the curve only keeping the geometry as well as the parametrization intact. As ironic as it may, this property that motivated degree elevation research is a shortcoming when it comes to data exchange. When, say, a degree three curve is elevated to degree five, it has the *form* of a degree five curve, however, its *behavior* is degree three only. Data exchange may require form as well as behavior!

- The process introduces multiple knots which limit the continuity and can produce data explosion when a set of curves are merged together for subsequent (surface) constructions.

Degree elevation which generates multiple knots can be quite dangerous, as illustrated using the example of the circle. Assume that system B represents this crucial entity as a degree five NURBS curve for the very purpose of avoiding multiple knots and to have adequate continuity for subsequent operations. When the degree two NURBS circle, generated in system A, is read into system B, it must be elevated to degree five, producing knots with multiplicity five and discontinuities in the homogeneous space. That degree elevated circle is, of course, completely useless in system B and must be converted so that both *form* and *behavior* are satisfied in this system.

Spline conversion research goes back to the early eighties with most of the seminal works published in the late eighties and early nineties [1-6,7-14,17-20]. Nearly all published papers fall into one of the categories of sampling, polynomial base changes and upper bound techniques [7,9]. The primary concerns about these methods revolve around guess work, over-sampling, very restrictive theories, iterations that are not guaranteed to converge, and the error control is based on large upper bounds or on discrete sampling. The method presented in this paper has the following major components: (1) the original curve is decomposed based on the requirements of the output curve (not on the intrinsic characteristics of the original curve), (2) curves are generated interpolating points as well as (some) derivatives, and (3) the error is measured precisely based on symbolic operators [16].

The organization of the paper is as follows. Section 2 introduces some NURBS notations followed by an overview of the algorithm in Section 3. Decomposition details are given in Section 4, interpolation details are in Section 5 and the error control mechanism is covered in Section 6. Examples are presented in Section 7 followed by some conclusions.

2. NURBS NOTATION

For notational convenience, curve definition is introduced first. A NURBS curve of degree p will be defined as follows [15]:

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

where $P_i^w = (w_i x_i, w_i y_i, w_i z_i, w_i)$ are the control points in homogeneous space, and $N_{i,p}(u)$ are the normalized B-splines defined over the non-uniform and clamped knot vector

$$U = \{\underbrace{u_0 = \cdots = u_p}_{p+1}, u_{p+1}, \ldots, u_n, \underbrace{u_{m-p} = \cdots = u_m}_{p+1}\}$$

Unless otherwise stated, the knot vector is always clamped. We are seeking the approximating curve as a non-rational curve of degree q in the form:

$$A(u) = \sum_{i=0}^{k} N_{i,q}(u) Q_i$$

where $Q_i = (x_i, y_i, z_i)$ are the control points in the Euclidean space. Although there is no restriction on the approximating degree q, it is a good idea to impose a maximum. Once the degree goes over five or six, both the speed as well as the accuracy of NURBS algorithms begin to deteriorate quite fast.

3. ALGORITHM OVERVIEW

The major steps of the algorithm are explained below.

```
decompose the curve based on the tolerance and the required degree
interpolate through the decomposition points and the derivatives at these points

while not done do
    compute the error curve
    get the maximum or the average errors
    if the error is within tolerance → done
    for all decomposed segments do
        check if the interpolant is within tolerance on this segment
        if not, insert a new point
    end
    interpolate the decomposition points, the derivatives at these points and the newly inserted points
end
```

The items underlined in the above algorithm are the major steps requiring further explanations and/or details. Fig. 1 shows an example. On the left an initial interpolation is generated through the decomposition points and the derivatives at these points. Since the error turned out to be larger than allowed on the last three segments, extra points have been inserted and a new interpolant has been computed. The dashed curve is a degree three curve, whereas the solid one is a degree four curve.

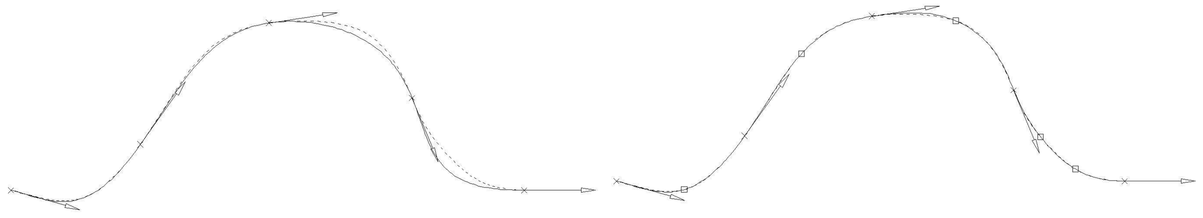

Fig. 1: Interpolation through decomposition points and derivatives (left), adding extra points to the segments with large deviation from the original curve (right). Notation: ($\times$) decomposition points, ($\square$) extra points.

4. CURVE DECOMPOSITION

A critical element of the spline conversion method is the decomposition of the original curve into segments that are simple enough to be approximated locally to within a tolerance. Given a degree p NURBS curve, the required approximating degree q, a set of parameter values are sought that represent a decomposition. The algorithm is outlined below.

```
initialize a stack with the start and end knots
while stack is not empty do
    $(u_l, u_r)$ ← pop the stack
    extract segment $C(u \mid [u_l, u_r])$
    get end derivatives
    compute a Bezier approximation of degree $q$
    compute the error curve
    $u_m$ ← compute the maximum error
    if the error is not within tolerance
        $(u_m, u_r)$ → push the stack
        $(u_l, u_m)$ → push the stack
    else
        save parameter $u_m$
    end
end
```

There are two types of Bezier curve approximations, one is for degree reduction and the other is for degree elevation (it should be emphasized that these Bezier curves are intermediate approximations used to get the decomposition points only). The curve is sought in the following format:

$$C(u) = \sum_{i=0}^{q} B_{i,q}(u) P_i$$

where $B_{i,q}(u)$ denote the Bernstein polynomials of degree q. There are three ways of determining a Bezier approximation from a given curve: (1) using end derivatives, (2) using points along the curve, or (3) the combination of the two. The subsections below provide the necessary details.

4.1 Bezier Curves from End Derivatives

This method is appropriate if degree reduction is sought, i.e. the end derivatives of the base curve over-determine the Bezier curve. Given end derivatives

$$D_s^0 = C^{(0)}(u_l),...,D_s^k = C^{(k)}(u_l) \qquad D_e^0 = C^{(0)}(u_r),...,D_s^l = C^{(l)}(u_r)$$

a Bezier curve is sought that satisfies the following conditions:

$$C^{(\alpha)}(0) = D_s^\alpha,\, \alpha \le k \qquad C^{(\beta)}(1) = D_e^\beta,\, \beta \le l$$

Relying on the end derivative formula for Bezier curves [15], the control points from the left are computed as follows:

$$P_0 = D_s^0$$
$$P_r = \frac{1}{\prod_{j=0}^{r-1}(q-j)} D_s^r + \sum_{i=1}^{r}(-1)^{(i-1)} BC_i^r P_{r-i} \qquad r = 1,...,k$$

where BC_i^r denotes the i^{th} binomial coefficient of degree r. Now, the control points from the right are given as:

$$P_{q-s} = (-1)^s \left[\frac{1}{\prod_{j=0}^{s-1}(q-j)} D_e^s + \sum_{i=1}^{s}(-1)^i BC_{i-1}^s P_{q-i+1} \right] \qquad s = 1,...,l$$
$$P_q = D_e^0$$

These formulas may be used to compute the control points from the left and the right until all of them are computed, or the left and right set of control points may overlap in which case an averaging may be applied. For example, reducing the degree from four to three using first and second derivatives, produces two sets of inner control points of the approximating degree three curve.

4.2 Bezier Curves from End Derivatives and Internal Points

In case degree elevation is required, the base curve does not provide enough derivatives to determine the Bezier curve. In this case some internal points are also selected in addition to the end derivatives above:

$$Q_0 = D_s^0, Q_1, ..., Q_{n-1}, Q_n = D_e^0$$

and the Bezier curve must now satisfy the following conditions:

$$C^{(r)}(0) = D_s^r,\, r = 0,...,k \qquad C(t_i) = Q_i,\, i = 0,...,n \qquad C^{(s)}(1) = D_e^s,\, s = 0,...,l$$

where the interpolation of the internal points is assumed at some parameters $t_0,...,t_n$. In order to solve this interpolation problem, the derivatives of the Bernstein polynomials are required. Based on the well known recursive definition [15]:

$$B_{i,q}^{(k)} = q\left[B_{i-1,q-1}^{(k-1)} - B_{i,q-1}^{(k-1)}\right]$$

a more efficient, non-recursive, version of the derivative formula is as follows:

$$B_{i,q}^{(k)} = \prod_{l=0}^{k-1}(q-l)\sum_{j=0}^{k}(-1)^j BC_j^k B_{i-k+j,q-k}$$

Given the end derivatives, the internal points and the derivatives of the Bernstein polynomials, the interpolation problem is solved by the following matrix equation:

$$\begin{bmatrix} 1 & 0 & & & & \cdots & & & 0 & 0 \\ B_0^{(1)}(0) & B_1^{(1)}(0) & & & & & & & & 0 \\ \cdots & & & & & & & & & 0 \\ B_0^{(k)}(0) & B_1^{(k)}(0) & B_2^{(k)}(0) & \cdots & B_k^{(k)}(0) & & & & & 0 \\ B_0(t_1) & B_1(t_1) & & & & \cdots & & & B_{q-1}(t_1) & B_q(t_1) \\ \cdots & & & & & \cdots & & & & \cdots \\ B_0(t_{n-1}) & B_1(t_{n-1}) & & & & \cdots & & & B_{q-1}(t_{n-1}) & B_q(t_{n-1}) \\ 0 & & & & & B_{q-l}^{(l)}(1) & B_{q-l-1}^{(l)}(1) & \cdots & B_{q-1}^{(l)}(1) & B_q^{(l)}(1) \\ 0 & & & & & \cdots & & & & \cdots \\ 0 & & & & & \cdots & & & B_{q-1}^{(1)}(1) & B_q^{(1)}(1) \\ 0 & 0 & & & & \cdots & & & 0 & 1 \end{bmatrix} \begin{bmatrix} P_0 \\ P_1 \\ \vdots \\ P_{q-1} \\ P_1 \end{bmatrix} = \begin{bmatrix} Q_0 \\ D_s^{(1)} \\ \vdots \\ D_s^{(k)} \\ Q_1 \\ \vdots \\ Q_{n-1} \\ D_e^{(l)} \\ \vdots \\ D_e^{(1)} \\ Q_n \end{bmatrix}$$

The matrix consists of an upper left and a lower right diagonal components as well as a full $(q+1)\times(n-1)$ band. It is safely solved by LU-decomposition followed by forward-backward substitution. Fig. 2 shows an example of using Bezier curves to approximate parts of a given NURBS curve. Note how the Bezier pieces (that can be considered as the localizations of the NURBS curve) capture the shape information on the local NURBS segment. The end points of the local Bezier segment are used as the decomposition points necessary for the global interpolation process detailed in the next section.

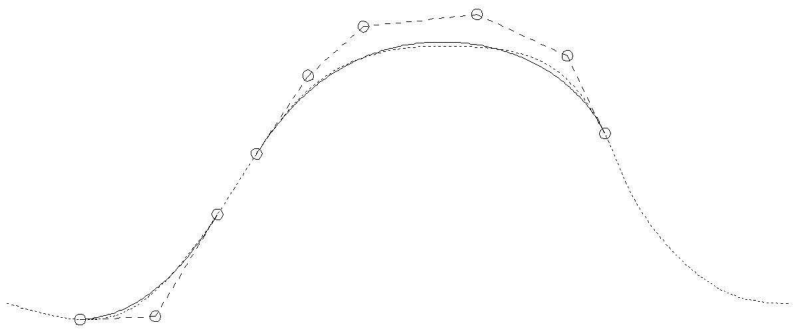

Fig. 2: Approximations of NURBS curve segment by Bezier curves of degree two (left part) and five (top part). Only first derivatives are used and for degree five two extra points have been selected.

5. GLOBAL CURVE INTERPOLATION

The algorithm described so far is able to generate decomposition points along the curve, derivatives at these points, and additional points in each segment. This set of data is more than enough to generate an interpolant that assumes the data points as well as the derivatives. Experience has shown that the quality of the approximation is better if more internal points are chosen with only first end derivatives than using higher end derivatives with a smaller set of internal points. Therefore the global interpolation problem is formulated as follows. Given a set of data points (some of which are decomposition points) and some derivatives at selected points (represented by an index set)

$$Q_0,\ldots Q_n \quad D_0,\ldots,D_k \quad I_0,\ldots,I_k$$

compute a B-spline curve that interpolates the points and the derivatives, i.e.

$$C(t_i) = Q_i,\ i = 0,\ldots,n \quad C'(t_{I_j}) = D_j,\ j = 0,\ldots,l$$

To solve this problem, the required entities are the parameters, the knots and a properly set-up system of equations. The parameters are chosen by one of the standard methods, e.g. chord length. The system of equations may be set up as follows:

$$\begin{bmatrix} 1 &&&&&&&&&& \\ -1 & 1 &&&&&&&&& \\ & N_1(t_1) & N_2(t_1) & \cdots & N_{p+1}(t_1) &&&&&& \\ &&& \ddots &&&&&&& \\ &&& N_i(t_i) & N_{i+1}(t_i) & \cdots & N_{i+p+1}(t_i) &&&& \\ &&& N_i'(t_i) & N_{i+1}'(t_i) & \cdots & N_{i+p+1}'(t_i) &&&& \\ &&&&& \ddots &&&&& \\ &&&& N_j(t_j) & N_{j+1}(t_j) & \cdots & N_{j+p+1}(t_j) &&& \\ &&&&&&&& \ddots && \\ &&&&&&&&& -1 & 1 \\ &&&&&&&&&& 1 \end{bmatrix} \times \begin{bmatrix} P_0 \\ P_1 \\ \vdots \\ P_{r-1} \\ P_r \end{bmatrix} = \begin{bmatrix} Q_0 \\ \gamma_0 D_0 \\ Q_1 \\ \vdots \\ Q_i \\ D_i \\ \vdots \\ Q_j \\ \vdots \\ \gamma_n D_n \\ Q_n \end{bmatrix}$$

where for computational convenience the following notation has been used:

$$\left.\begin{aligned} C'(u_0) &= \frac{p}{u_{p+1}-u_0}(P_1 - P_0) \\ C'(u_m) &= \frac{p}{u_m - u_n}(P_n - P_{n-1}) \end{aligned}\right\} \Rightarrow \left\{\begin{aligned} \gamma_0 D_0 &= -P_0 + P_1 \quad \gamma_0 = \frac{u_{p+1}-u_0}{p} \\ \gamma_n D_n &= -P_{n-1} + P_n \quad \gamma_n = \frac{u_m - u_n}{p} \end{aligned}\right.$$

The most difficult part of the above method (and all interpolation methods) is to choose the right knots. They are chosen in two steps: (1) first knots are computed without derivatives using the De Boor knots [15]. Then extra knots are inserted depending on at which point derivatives are specified. The key element here is to keep the matrix banded by localizing the knot insertion (note that derivatives generate an extra row of basis functions which push the diagonal element one position to the right). More precisely, given parameters $t_0,\ldots,t_n$, the required degree q and an index set $I_0,\ldots,I_k$, compute the knots $u_0,\ldots,u_m,\ m = n + k + p + 2$. The details of the algorithm are explained below.

```
(* Compute the knots without derivatives present *)
**if** $I_0 = 0$  $is = 0$  **else** $is = 1$
**if** $I_k = n$  $ie = n - p + 1$ **else** $ie = n - p$
$a_0 = t_0$   $r = 0$
**for** $i = is$ **to** $ie$ **do**
    $sum = 0$   $r = r + 1$
    **for** $j = i$ **to** $i + p - 1$ **do** $sum = sum + t_j$
    $a_r = sum / p$
**end**
```

```
(* Add extra knots where derivatives are present *)
$s = 0$
for $i = 0$ to $r-1$ do
    $c_0 = a_i$
    $c_1,...,c_{l-1} \leftarrow$ parameters inside $[a_i, a_{i+1})$
    $c_l = a_{i+1}$
    $II_1,...,II_{l-1} \leftarrow$ indexes of the parameters
    for $j = 0$ to $l-2$ do
        if at $II_{j+1}$ derivatives are present
            $s = s+1$
            $b_s = (c_j + c_{j+1} + c_{j+2})/3$
        end
    end
end
$U \leftarrow \mathrm{Merge}(a_0,...a_r; b_0,...,b_s)$
```

Some relevant notes worth mentioning are: (1) the start and end indexes of initial averaging depends on whether there are end derivatives; (2) the extra knots are anchored to the initial knots as well as to the parameters (extra knots are added to the spans where the parameters are found that belong to the derivatives); and (3) the initial averaging process, based on p consecutive parameters, cannot produce more than $p-2$ consecutive empty spans. Adding extra knots to the non-zero spans can cause the number to increase on each end of the zero spans by one, potentially producing p consecutive empty spans. However, this can only happen if derivatives are present which introduce extra rows to the interpolation matrix, keeping the matrix banded.

6. ERROR CURVE

The algorithm for spline conversion must compute errors on the entire segment as well as on the local, decomposed ones. The error is computed by symbolically computing the difference of two curves, the original curve $C(u)$ and its approximation $A(u)$:

$$D(u) = C(u) - A(u)$$

There are two cases to consider: (1) at least one of the curves is rational, or (2) both are non-rational. If one of the curves is rational, the computation requires the evaluation of the following form:

$$C(u) = \frac{N_C(u)}{d_C(u)} \quad A(u) = \frac{N_A(u)}{d_A(u)} \Rightarrow D(u) = \frac{N_C(u)d_A(u) - N_A(u)d_C(u)}{d_C(u)d_A(u)}$$

To compute the difference curve, the following utilities are required: (1) decomposing a NURBS curve into numerator curve and denominator function, (2) to compute the product of a B-spline curve and a function, (3) to compute the product of two B-spline functions, (4) to compute the difference of two B-spline curves, and (5) to form a NURBS curve from numerator and denominator. If the curves happen to be non-rational, the difference curve is computed as follows: (1) degree elevate the lower degree curve (note that this is precise degree elevation, not an approximation), (2) merge the knots to obtain a definition of the two curves on a common knot vector, and (3) compute the difference of the control points. Fig. 3 shows the difference curves computed from the data shown in Fig. 1. Notice how much more uniform the error distribution is after point insertion into the last three segments.

Given the error curve as a NURBS curve

$$D(u) = \sum_{i=0}^{n} R_{i,p}(u)P_i \quad R_{i,p}(u) = \frac{N_{i,p}(u)w_i}{\sum_{j=0}^{n} N_{j,p}(u)w_j}$$

where $R_{i,p}(u)$ are rational basis functions or the B-splines, maximum or average errors are computed using the (refined) control polygon:

$$\varepsilon_{\max} = \max(|P_i|),\ i = 0,\ldots,n \qquad \varepsilon_{ave} = \frac{1}{n}\sum_{i=0}^{n}|P_i|$$

The parameter for the maximum error is computed to be the node corresponding to the control point with maximum distance from the origin.

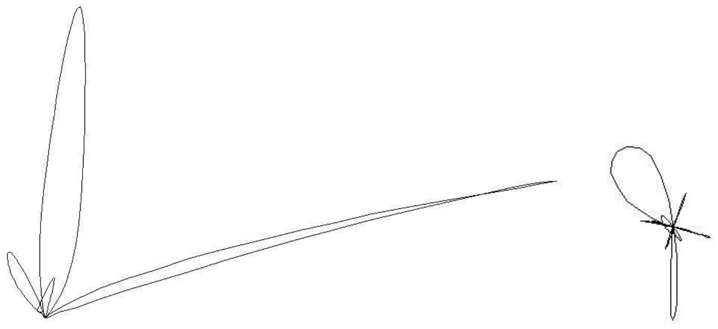

Fig. 3: Left: difference curve for Fig. 1 left; right: difference curve between the curves in Fig. 1 right.

To make the above formula a useful error measure, a proper knot refinement is required. Knot refinements that place knots into the middle of each non-zero span may not work for proper error control as large spans may be under-refined whereas small ones over-refined. The algorithm implemented is explained below.

```
$sa \leftarrow$ the average of non-zero span lengths
**for** all non-zero spans **do**
    $d = \frac{u_{k+1} - u_k}{sa}$
    $n = \lfloor d + 1.0 \rfloor$
    insert $n$ number of knots uniformly into the span $[u_k, u_{k+1})$
**end**
```

Using this simple knot refinement process, the knot vector of the difference curve is successively refined until the relative maximum (or average) bounds do not change much. Let L denote the approximate arc length of the difference curve (the length of the initial control polygon), then knot refinement stops when

$$\frac{\varepsilon_{i+1} - \varepsilon_i}{L} < \delta$$

where δ is a relative tolerance (set in our system as 0.1%), and ε_i denotes the error after the i^{th} knot refinement. Practical experience shows that after about three knot refinements the control polygon approximates the error curve quite well, giving rise to a very precise error measure.

7. EXAMPLES

The first set of examples is shown in Fig. 4, elevating the degree of a degree two circle to three, four and five. Note that the degree-two circle is defined by a double knot and therefore it is discontinuous in homogeneous space. The conversions to degree three, four and five are all non-rational and have full continuity of C^2, C^3 and C^4, respectively. Tab. 1 (columns 2-4) provides numerical data on how many control points are required to achieve the desired level of approximation. The asterisk over some numbers means that the decomposition returned too few segments for global interpolation and the algorithm had to insert some extra points.

The second set of examples, shown in Fig. 5, illustrate degree reduction of a degree five circle to degrees four, three and two. Note that the distribution of the control points in Figs. 4 and 5 is not uniform, as may be expected from the circle, however, it is the reflection of the parametrization of the original curve. As indicated above, the conversion algorithm is parametrization based not geometry based. The required number of control points is shown in Tab. 1 (columns 5-7).

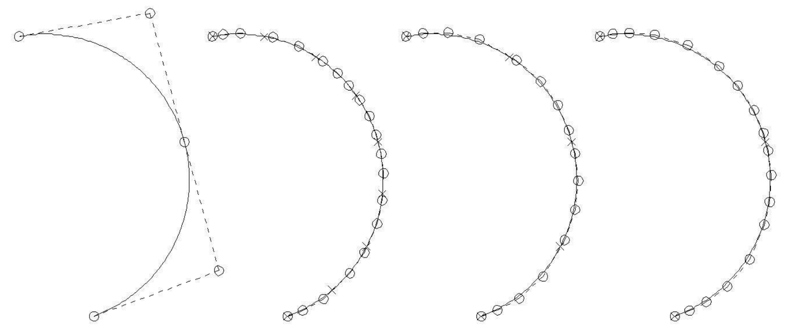

Fig. 4: Conversion of a degree-two circle to circles of degrees three, four and five; $\varepsilon = 10^{-4}$.

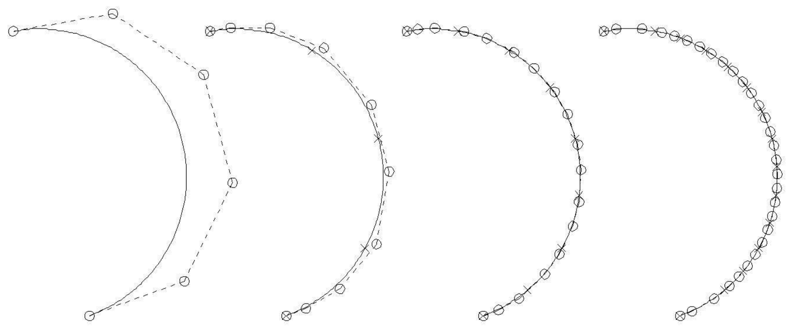

Fig. 5: Conversion of a degree-five circle to circles of degrees four, three and two; $\varepsilon = 10^{-4}$.

ε	**2→3**	**2→4**	**2→5**	**5→4**	**5→3**	**5→2**	**3→2**	**3→4**
10^{-2}	7*	9*	11*	9*	7	9*	11	12
10^{-3}	10	9*	11*	9*	9	9	23	14
10^{-4}	19	15	17*	9	17	29	57	29
10^{-5}	39	27	24*	13	33	65	121	52
10^{-6}	74	51	57	17	49	129	239	86
10^{-7}	92	77	70	33	65	257	513	132
10^{-8}	166	113	93	37	129	513	1025	179
10^{-9}	303	141	107	65	257	1025	2337	251
10^{-10}	563	185	157	129	513	2049	4981	300

Tab. 1: Number of control points required to convert the curves in Figs. 4-6.

The last example is shown in Fig. 6 where a degree three free-form curve has been degree reduced to two and degree elevated to four. Tab. 1 (columns 8-9) show the number of required control points.

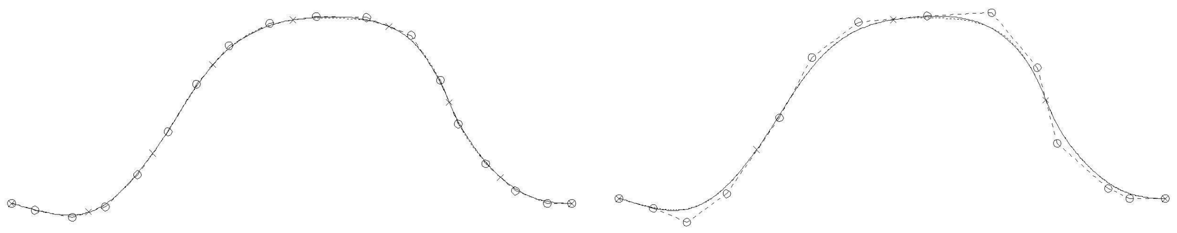

Fig. 6: Degree reduction (left) and degree elevation (right) of free-form curve; $\varepsilon = 5 \cdot 10^{-3}$.

8. CONCLUSIONS

An algorithm for converting NURBS curves to any degree B-spline curves has been presented. The method relies on straightforward computations used for curve decomposition, curve interpolation and symbolic operations on NURBS entities. The main advantages of the method are: (1) reliability (no iterations, guess parameters or discrete sampling are required), (2) speed (a relatively small number of control points are necessary to achieve approximations to within common engineering tolerances), (3) quality (both the geometry as well as the parametrization are approximated), and (4) precise error control (symbolic operators are used to compute the error curve). A slight disadvantage (discomfort) is that the method requires significant amount of code, i.e. it requires a full-blown NURBS kernel to support the various forms of interpolation techniques as well as the symbolic operators.

9. ACKNOWLEDGEMENTS

The work reported in this paper was supported by the National Science Foundation under grant No: DMI-0200385, awarded to the University of South Florida. All opinions, findings, conclusions and recommendations expressed in this paper are those of the author and do not necessarily reflect the views of the National Science Foundation or the University of South Florida.

10. REFERENCES

[1] Cho, W.; Maekawa, T.; Patrikalakis, N. M.: Topologically reliable approximation of composite Bezier curves, Computer-Aided Geometric Design, 13(6), 1996, 497-520.

[2] Dannenberg, L.; Nowacki, H.: Approximate conversion of surface representations with polynomial bases, Computer-Aided Geometric Design, 2(1-3), 1985, 123-131.

[3] Degen, W. L. F.: Best approximation of parametric curves by splines, in Mathematical Methods in CAGD II, Lyche, T.; Schumaker, L. L. (eds.), Academic Press, New York, NY, 1992, 171-184.

[4] Dokken, T.; Lyche, T.: Spline conversion: existing solutions and open problems, in Curves and Surfaces in Geometric Design, Laurent, P.-J.; Le Mehaute, A.; Schumaker, L. L. (eds.), A. K. Peters, Wellesley, 1994, 121-130.

[5] Eck, M.: Degree reduction of Bezier curves, Computer-Aided Geometric Design, 10(3-4), 1993, 237-251.

[6] Eck, M.; Hadenfeld, J.: A stepwise algorithm for converting B-splines, in Curves and Surfaces in Geometric Design, Laurent, P.-J.; Le Mehaute, A.; Schumaker, L. L. (eds.), A. K. Peters, Wellesley, 1994, 131-138.

[7] Filip, D.; Magedson, R.; Markot, R.: Surface algorithms using bounds on derivatives, Computer-Aided Geometric Design, 3(4), 1986, 295-311.

[8] Floater, M.: High-order approximation of conic sections by quadratic splines, Computer-Aided Geometric Design, 12(6), 1995, 617-637.

[9] Holzle, G. E.: Knot placement for piecewise polynomial approximation of curves, Computer-Aided Design, 15(5), 1983, 295-296.

[10] Hoschek, J.: Approximate conversion of spline curves, Computer-Aided Geometric Design, 4(1-2), 1987, 59-66.

[11] Hoschek, J.; Wissel, N.: Optimal approximate conversion of spline curves and spline approximation of offset curves, Computer-Aided Design, 20(8), 1988, 475-483.

[12] Hoschek, J.; Schneider, F.-J.: Approximate conversion and data compression of integral and rational B-spline surfaces, in Curves and Surfaces in Geometric Design, Laurent, P.-J.; Le Mehaute, A.; Schumaker, L. L. (eds.), A. K. Peters, Wellesley, 1994, 241-250.

[13] Kallay, M.: Approximating a composite cubic curve by one with fewer pieces, Computer-Aided Design, 19(10), 1987, 536-543.
[14] Patrikalakis, N. M.: Approximate conversion of rational splines, Computer-Aided Geometric Design, 6(2), 1989, 155-165.
[15] Piegl, L.; Tiller, W.: The NURBS Book, Springer-Verlag, New York, NY, 1997.
[16] Piegl, L.; Tiller, W.: Symbolic operators for NURBS, Computer-Aided Design, 29(5), 1997, 361-368.
[17] Pratt, M. J.; Goult, R. J.; Ye, L.: On rational parametric curve approximation, Computer-Aided Geometric Design, 10(3-4), 1993, 363-377.
[18] Sederberg, T.; Kakimoto, M.: Approximating rational curves using polynomial curves, in NURBS for Curve and Surface Design, G. Farin (ed.), SIMA, Philadelphia, PA, 1991, 149-158.
[19] Wolter, F. E.: Approximation of high-degree and procedural curves, Engineering with Computers, 8(2), 1992, 61-80.
[20] Yang, H.; Wang, W.; Sun, J.: Control point adjustment for B-spline curve approximation, Computer-Aided Design, 36(7), 2004, 639-652.