CLOSED-FORM EXPRESSIONS FOR THE APPROXIMATION OF ARCLENGTH PARAMETERIZATION FOR BÉZIER CURVES

MOHSEN MADI*

* Department of Computer Science, University of Sharjah
Sharjah, P.O. Box 27272, United Arab Emirates
e-mail: mmadi@sharjah.ac.ae

In applications such as CNC machining, highway and railway design, manufacturing industry and animation, there is a need to systematically generate sets of reference points with prescribed arclengths along parametric curves, with sufficient accuracy and real-time performance. Thus, mechanisms to produce a parameter set that yields the coordinates of the reference points along the curve $\boldsymbol{Q}(t) = \{x(t), y(t)\}$ are sought. Arclength parameterizable expressions usually yield a parameter set that is necessary to generate reference points. However, for typical design curves, such expressions are not often available in closed form. It is thus desirable to find efficient ways to compensate for this lack of arclength parameterization. In this paper, several methods for approximating arclength parameterizations are studied. These methods are examined for both accuracy and real-time processing requirements. The application of generating reference points uniformly spaced along the paths of several curves is chosen for the illustration and comparison between the presented methods.

Keywords: arclength parameterization, Hermite interpolation, reference points

1. Introduction

A need to generate sets of reference points ($\boldsymbol{R}$s) along the paths of mechanical tools or parts is present in CAD and CAM applications. For convenience, reference points are referred to as $\boldsymbol{R}$s and the i-th reference point is designated as $\boldsymbol{R}_i$. As an example, in CNC machining, computers with CAD systems might be instructed to produce thousands of $\boldsymbol{R}$s along the path of a manufactured part (such as the fuselage of an airplane, where uniform spacing between adjacent reference points is desired to minimize tension) according to prescribed specific locations (Farouki, 1992; Farouki and Shah, 1996; Sharpe and Thorne, 1982). The manipulated part may be required to be affixed to other complementing parts by bolts through adjacent holes, in locations marked by reference points.

A first step towards generating a set of N $\boldsymbol{R}$s with prescribed arclengths is to abstract the physical paths along the object of interest by a *parametric* curve $\boldsymbol{Q}(t)$ in the *Bernstein-Bézier* representation (Farin, 1993; Su and Liu, 1989). Next, several problems have to be solved, usually in the following order:

(a) Obtain an expression for the arclength $s(t),\ t \in [0, 1]$.

(b) Compute the total arclength $\mathcal{L} = s(1)$.

(c) Determine the set of desired arclengths $\{s_{i=1,\dots,N}\}$, $s_i \in (0, \mathcal{L}]$, at which the $\boldsymbol{R}$s are to be generated.

(d) Re-parameterize $\boldsymbol{Q}(t)$ with the parameter set $\{t_i\}$ obtained from $t(s_i)$: the inverse function of the arclength expression obtained in (a).

Arclength parameterization is desirable because the arclength is an intrinsic quantity of the curve and arclength parameterization is an intrinsic property of the curve. It facilitates the design and analysis of curves and surfaces (Burchard *et al.*, 1994; Guggenheimer, 1963; Young, 1993). If a closed-form solution is available for the items (a) and (d) above, the coordinates of an $\boldsymbol{R}$ that lie at arclength s_i along $\boldsymbol{Q}(t)$ may accurately be obtained by first evaluating $t_i = t(s_i)$, and then evaluating $\boldsymbol{Q}(t)$ at $t = t_i$.

In general, however, because of the non-linearity of the integral expression $s(t)$ (see the next section), it is impossible to solve it in an analytic fashion, and even when this is possible, trying to derive an arclength parameterizable expression $t(s)$ from it usually fails (Farouki and Sakkalis, 1991; Sharpe and Thorne, 1982; Young, 1993). Because of this, several approaches are taken to approximate results that would be attained by arclength parameterization. In this paper, $s(t)$ is exploited to derive a cubic interpolating function to approximate $t(s)$, and the closeness of this function, in comparison with the existing methods of actual arclength parameterization, is discussed.

Although Pythagorean-hodograph curves (Farouki and Sakkalis, 1991) have closed form expressions for their arclengths, they still require the solution of a non-linear equation to obtain the parameter as a function of the arclength. This article is concerned with a more general class of polynomial curves.

2. Mathematical Preliminaries

For the purposes of this paper, a curve is represented by a parametric polynomial $\boldsymbol{Q}(t)$ that is in the Bernstein-Bézier representation:

$$\boldsymbol{Q}(t) = \sum_{i=0}^{n} \boldsymbol{p}_i \binom{n}{i} (1-t)^{n-i} t^i, \quad 0 \le t \le 1. \tag{1}$$

The properties and importance of such a representation for CAD/CAM are indicated in the literature (Farin, 1993; Su and Liu, 1989). In the above equation, n denotes the curve degree, and $\boldsymbol{p}_i \in \mathbb{E}^2$ are the Bézier points that constitute the control polygon of the curve.

The arclength $s(t)$ of $\boldsymbol{Q}(t)$ is determined by the following integral:

$$s(t) = \int_0^t \|\boldsymbol{Q}'(\tau)\| \, \mathrm{d}\tau, \tag{2}$$

where $\boldsymbol{Q}'(t)$ is the derivative of $\boldsymbol{Q}(t)$. The total arclength of $\boldsymbol{Q}(t)$ is therefore $\mathcal{L} = s(1)$.

Because of the non-linearity and the integral term present in $s(t)$, an arclength parameterizable expression $t(s)$ usually has to be approximated rather then derived directly from (2).

3. Related Work

Several methods have been developed to approximate arclength parameterization, some of which are based on curve dependent tables of data, while others are not. The former class of approximators have the advantage of being adaptable for a prescribed accuracy by several numerical techniques. It is difficult to obtain a meaningful comparison for methods which are not of the same class. In some applications such as graphical simulation and animation, where the animated object is to appear at approximately evenly spaced intervals for smooth appearance, it is the performance rather than the accuracy that is of importance (Madi, 1996). In the remainder of this section, an overview of the existing methods is presented.

3.1. Basic Parametric Flow (BPF)

The simplest method for $\boldsymbol{R}$s generation may be called the basic parametric flow (BPF), since a number of N points are produced by uniform parameter spacing (e.g., $t_{i=0,\dots,N} = i/N$). Although the method is simple and fast, it is well known that it is not suitable for generating points along the arclength of a curve (Farouki, 1997; Madi, 1996).

3.2. Sharpe and Thorne's (ST) Method

The method described by Sharpe and Thorne can accurately produce $\boldsymbol{R}$s at prescribed arclengths (Sharpe and Thorne, 1982). However, it has a high computational cost associated with "extracting" the corresponding parametric value for each $\boldsymbol{R}$ to be generated. Consider the following non-linear equation used to find t_i, the parametric value needed to generate $\boldsymbol{R}_i$:

$$M(t) = \int_{t_{i-1}}^{t} \sqrt{\boldsymbol{Q}'(\tau)\boldsymbol{Q}'(\tau)} \, \mathrm{d}\tau - s_i = 0, \quad i = 1, \dots, N, \tag{3}$$

where t_{i-1} is the parametric value corresponding to $\boldsymbol{R}_{i-1}$, the solution $t = t_i$ is the value corresponding to the next reference-point $\boldsymbol{R}_i$, and s_i is the arclength from $\boldsymbol{R}_{i-1}$ to $\boldsymbol{R}_i$. In order to obtain t_i, a few Newton-Raphson iterations are applied:

$$\tau_j = \tau_{j-1} - \frac{M(\tau_{j-1})}{M'(\tau_{j-1})}, \quad \tau_0 = t_{i-1}, \quad j = 1, 2, \dots, k, \tag{4}$$

where $M'(\tau_j)$ is the derivative of $M(\tau_j)$. The value of t_i is given by τ_k, where k is the number of iterations required for convergence to an acceptable accuracy.

For applications requiring real-time processing, or those not requiring very accurate spacing of $\boldsymbol{R}$s, this method may be impractical.

3.3. Optimal Parameterization (OP)

Farouki's OP is mathematically a rather intricate process (Farouki, 1997). The given polynomial curve $\boldsymbol{Q}(t)$ is first transformed into an equivalent rational form by transforming the parameter t in (1) (by applying a Möbius transformation) as follows:

$$t = \frac{(1-\alpha)u}{\alpha(1-u) + (1-\alpha)u}, \quad 0 < \alpha < 1, \quad 0 \le u \le 1. \tag{5}$$

Substituting (5) into (1) results in the following rational form:

$$\tilde{\boldsymbol{Q}}(u) = \frac{\sum_{i=0}^{n} w_i \boldsymbol{p}_i \binom{n}{i} (1-u)^{n-i} u^i}{\sum_{i=0}^{n} w_i \binom{n}{i} (1-u)^{n-i} u^i}, \tag{6}$$
$$w_i = (1-\alpha)^i \alpha^{n-i}.$$

The objective is to find the set of weights $\{w_i\}$ so that u approximates an arclength parameter. The problem is thus to find the "best" α for (6).

While the cost of obtaining the "right" α may be high, this method is better suited for applications requiring real-time processing than ST (Madi, 1996).

3.4. Cumulative Chordlength (CC)

The cumulative chordlength is a straightforward method to approximate the arclength of a curve. This method can be exploited to generate $\boldsymbol{R}$s that visually seem to be uniformly spaced. The algorithm is as follows: while computing the arclength, the set $\{s_k \mid k = 0, 1, \ldots, \eta\}$ (η being the number of the chords used to approximate the curve) keeps track of the cumulative chordlength obtained so far. $\boldsymbol{R}$s at distances $\{i\Delta d \mid i = 0, \ldots, N;\ \Delta d = s_\eta/N\}$, where s_η is the total chordlength, may then be located by searching for their closest values in $\{s_k\}$, and then further refining those values by means of a linear interpolation. That is, t_i, the parametric value corresponding to $\boldsymbol{R}_i$, at distance $i\Delta d$, is approximated by the function $A(i, k)$ as follows:

$$t_i = A(i,k) = \Delta u \left(k - 1 + \frac{i\Delta d - s_{k-1}}{s_k - s_{k-1}} \right), \tag{7}$$

where $s_{k-1} \le i\Delta d < s_k$, and $\Delta u = 1/\eta$.

While increasing the number of the chords approximating $\boldsymbol{Q}(t)$ *may* increase accuracy, the size of the curve-dependent data-table that has to be maintained also increases (Madi, 1996).

4. A Cubic Interpolator (CI)

Hermite interpolation (Davis, 1963; Farin, 1993; Foley *et al.*, 1992) is used to approximate $t(s)$ for any parametric curve $\boldsymbol{Q}(t)$. The cubic interpolator (CI) is defined by

$$\mathcal{F}(s) = as^3 + bs^2 + cs + d \approx t(s). \tag{8}$$

An approximation to the inverse function of $s(t) = \int_0^t \|\boldsymbol{Q}'(\tau)\|\,\mathrm{d}\tau$ may be derived as follows: First, $s(t)$ is differentiated to give

$$s'(t) = \frac{\mathrm{d}s}{\mathrm{d}t} = \|\boldsymbol{Q}'(t)\|. \tag{9}$$

From (9), $t'(s)$ can be written as follows:

$$t'(s) = \frac{\mathrm{d}t}{\mathrm{d}s} = \frac{1}{\|\boldsymbol{Q}'(t)\|}. \tag{10}$$

Integrating (10) produces

$$t(s) = \int_0^s \frac{1}{\|\boldsymbol{Q}'(\tau)\|}\,\mathrm{d}\tau. \tag{11}$$

The value of $t(s)$ at two parametric values is already known, namely, at $s = 0$ and $s = \mathcal{L}$. Further, for a cubic interpolator, two more items of data are needed to determine values for the four coefficients of $\mathcal{F}(s)$ in (8): $a,\ b,\ c$ and d. The geometry vector at the boundaries of the curve $t(s)$ is obtained as follows:

$$\begin{bmatrix} t(0) \\ t'(0) \\ t(\mathcal{L}) \\ t'(\mathcal{L}) \end{bmatrix} = \begin{bmatrix} 0 \\ \dfrac{1}{\|\boldsymbol{Q}'(0)\|} \\ 1 \\ \dfrac{1}{\|\boldsymbol{Q}'(1)\|} \end{bmatrix}. \tag{12}$$

Further, by requiring that $\mathcal{F}(s) = t(s)$ and $\mathcal{F}'(s) = t'(s)$ at the boundaries, we can solve it for the coefficients of (8) to get the following solution vector:

$$\begin{bmatrix} a \\ b \\ c \\ d \end{bmatrix} = \begin{bmatrix} \dfrac{1}{\mathcal{L}^2}\left(c + \dfrac{1}{\|\boldsymbol{Q}'(1)\|}\right) - \dfrac{2}{\mathcal{L}^3} \\ \dfrac{1}{\mathcal{L}^2} - \dfrac{c}{\mathcal{L}} - a\mathcal{L} \\ \dfrac{1}{\|\boldsymbol{Q}'(0)\|} \\ 0 \end{bmatrix}. \tag{13}$$

To illustrate the low cost of generating $\boldsymbol{R}$s using the CI method, the CI algorithm shown next is an implementation of the method described above (Madi (1996) gives details regarding the implementation and the cost of all other algorithms described here). The idea is to generate a set $\{\mathcal{F}_i \mid i = 0, \ldots, N\}$ by the approximating interpolating function, such that the evaluation of $\{\boldsymbol{Q}(\mathcal{F}_i)\}$ renders reference points that are approximately uniformly spaced.

The CI algorithm starts by calculating the coefficients $a, b,$ and c of the cubic interpolating function. The function $\mathcal{F}(s)$ in (8) is evaluated at $\{i\Delta L\}$ to yield $\{\mathcal{L}_i\}$.

```
CI()
    compute $\mathcal{L}$, scale $\|\boldsymbol{Q}'(0)\|$ and $\|\boldsymbol{Q}'(1)\|$
                                        {this scales $\boldsymbol{Q}(t)$}
    $c \leftarrow 1/\|\boldsymbol{Q}'(0)\|$
    $a \leftarrow c + 1/\|\boldsymbol{Q}'(1)\| - 2$
    $b \leftarrow 1 - c - a$
    $\Delta L \leftarrow 1/N,\ \ \ell_0 \leftarrow 0$
    for $i \leftarrow 1$ to $N$
        $\ell_i \leftarrow \ell_{i-1} + \Delta L$
        $\mathcal{F}_i \leftarrow ((a\ell_i + b)\ell_i + c)\ell_i$
                            {Eqn. (8) using Horner's method}
        $\boldsymbol{R}_i \leftarrow \boldsymbol{Q}(\mathcal{F}_i)$
END
```

Scaling $\|\boldsymbol{Q}'(0)\|$ and $\|\boldsymbol{Q}'(1)\|$ implies dividing them by $\mathcal{L}$; this scales the whole curve such that $\mathcal{L} = 1$.

Note that a precise measure of the deviation from a true uniform distribution is obtainable. Consider, e.g., the quintic PH curve of Fig. 4, where $N = 80$ reference points are generated along its path. By calculating the distance between every pair of reference points and accumulating the deviation from a true uniform distribution, an exact measure of the deviation from a true uniform distribution is obtained. In this case, the deviation is 5.87%. In the *CI()* function above, such a deviation can be computed by first initializing *deviation* to 0, and by adding the following two lines to the for-loop:

```
$d_i \leftarrow \sqrt{(x_i - x_{i-1})^2 + (y_i - y_{i-1})^2}$
$deviation \leftarrow deviation + |d_i - \Delta L|$.
```

Inversely, the distribution is 94.13% accurate, which is a huge achievement over the basic parametric flow (BPF()) parameterization yielding only 69.32%, at approximately the same cost (see Table 4).

5. Experimental Results

This section is divided into two subsections. The first subsection presents visual results on a sample of Bézier curves of several degrees. The objective is to show how close the $\boldsymbol{R}$s generated by various methods are to the exact $\boldsymbol{R}$s (note that a uniform spacing is desired), and how the results of the method presented in the previous section compare to those of other methods. In the last subsection, attention is turned to the cost of using the algorithms discussed to generate $\boldsymbol{R}$s.

5.1. Visual Results

Figures 1–5 show the result of applying the algorithms (methods) discussed to a sample of curves (the algorithms have been applied to a much larger sample of curves (Madi, 1996) for a much larger sample of curves).

Each of the figures is organized as follows: The actual curve is shown first, followed by plots showing only $\boldsymbol{R}$s along their translated paths. In each case, four $\boldsymbol{R}$ plots are given: those resulting from BPF (basic parametric flow), ST (Sharpe and Thorne), OP (Farouki's optimal parameterization), and from CI (cubic interpolator).

The ST reference points are considered to be exact (6–8 digits of accuracy (Madi, 1996)) and thus they are used to compare other results with. Because CC (cumulative chordlength) plots are *visually* indistinguishable from ST plots, they are not shown (Madi (1996) discusses CC and CC plots in greater detail).

In Fig. 1 we observe the closeness of OP and CI to ST results, whereas the results produced by the BPF are unsatisfactory.

For symmetric curves (e.g., Figs. 2 and 5), the value of α in the OP algorithm is $1/2$, thereby producing results exact to those of BPF (Farouki, 1997; Madi, 1996). These and other figures show the closeness of CI results to those of ST, obtained at a considerably lower cost than required by ST.

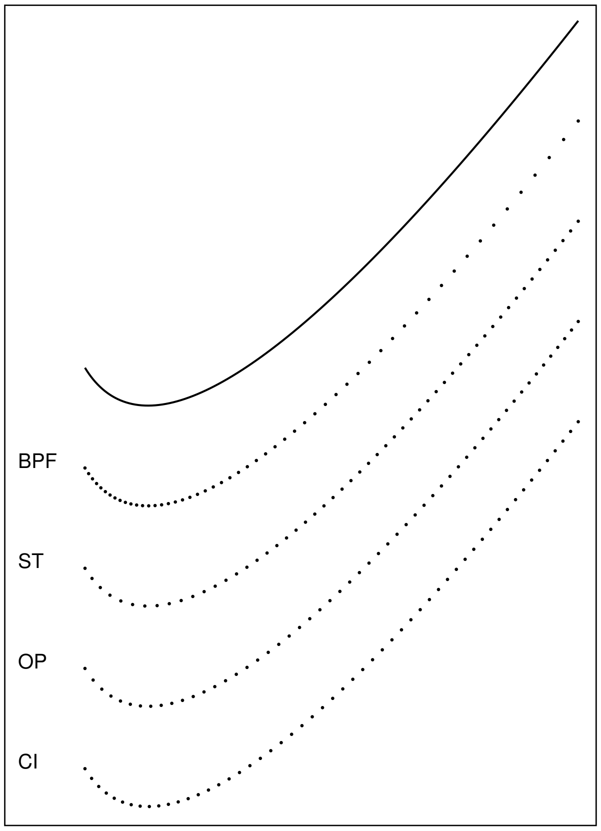

Fig. 1. Quadratic Bézier curve used by Farouki (1996).

5.2. Numerical Results

In this subsection, the costs of the different RPG algorithms are presented in tabular format to facilitate a comparison. Table 1 lists the cost expressions of all the algorithms, for both the setup time and the $\boldsymbol{R}$ generation time. Tables 2 and 3 show the setup-time cost, whereas Tables 4 and 5 consider the cost of generating $\boldsymbol{R}$s.

Because of the wide variation in the techniques used by various methods, a straightforward comparison is difficult. The following conventions are introduced to obtain meaningful comparisons (Madi, 1996):

(a) The degree of the curve, n, assumes values of 3, 5, and 9.

(b) N, the number of $\boldsymbol{R}$s to be generated, is fixed at 100.

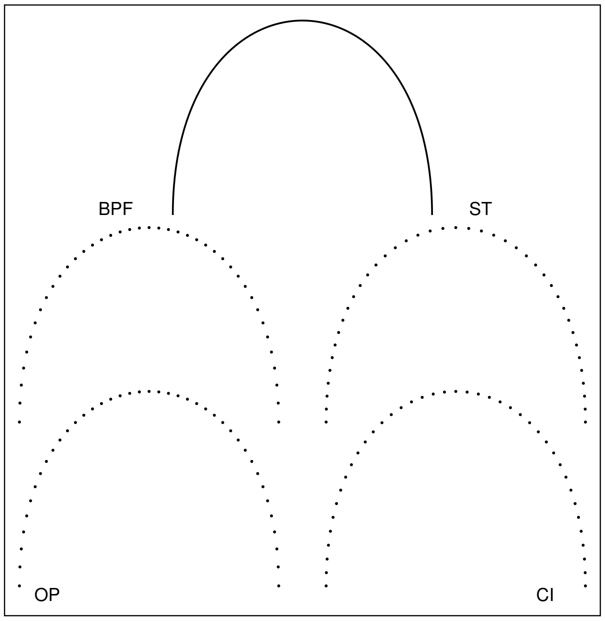

Fig. 2. Cubic PH curve.

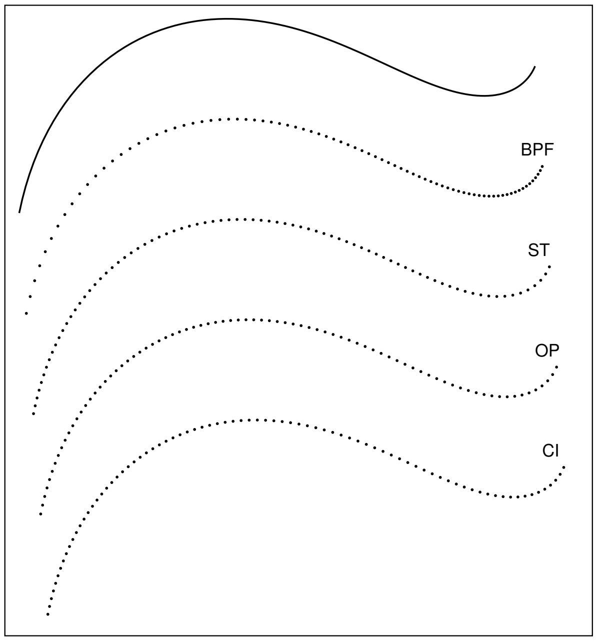

Fig. 4. Quintic PH curve.

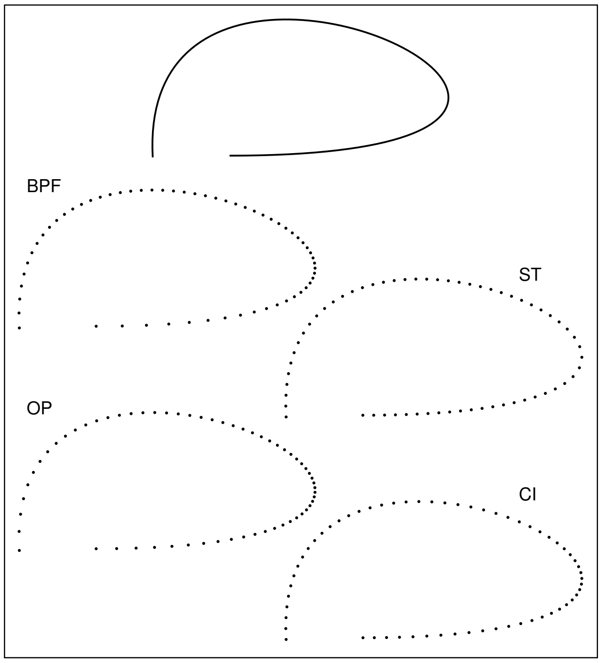

Fig. 3. Cubic Bézier exhibiting high curvature regions.

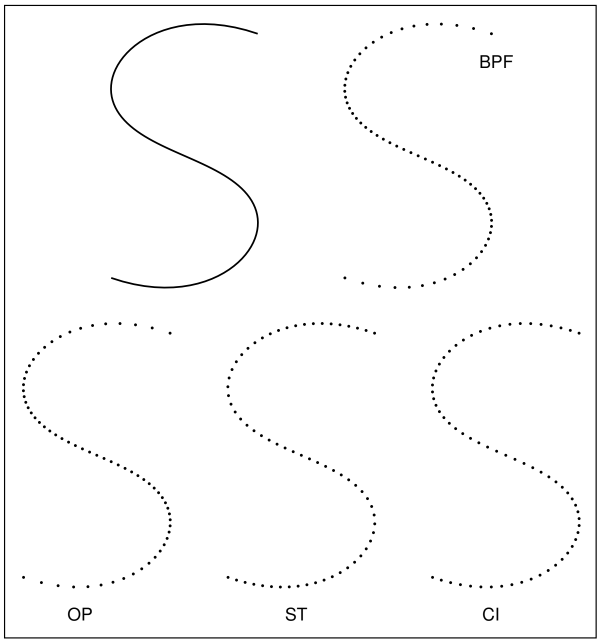

Fig. 5. Quintic S-curve.

(c) The number of chords that are used in the CC algorithm, η (only applicable to the CC algorithm), is set to 48 (note that this may be an underestimate for the number of chords required in practice).

(d) The number of Simpson intervals, I, required to compute arclengths is set as follows: to compute $\mathcal{L}$, the total arclength of $\boldsymbol{Q}(t)$, I is set to eight; to compute the arclength of a segment of $\boldsymbol{Q}(t)$ (as required by ST, for example), I is set to four.

(e) Algorithms with entries containing zero in every column are omitted.

(f) Entries are in the form $m_1 + m_2 + \cdots = \mathcal{S}$, where each term corresponds to the equivalent term in the cost expressions listed in Table 1, and $\mathcal{S}$ is their sum.

Table 1. List of cost expressions.

$C(\boldsymbol{Q}(t), N)$	$\left(N(5n-1)+2\left\{n+\left\lfloor\frac{n}{2}\right\rfloor-2\right\},0\right)$	
$C(s(t), I)$	$\left((5I+7)n+2\left\lfloor\frac{n-1}{2}\right\rfloor-2I, I+1\right)$	
$C(\text{Alg.}, 1)$	Setup	$\boldsymbol{R}$ Gen.
BPF	—	$C(\boldsymbol{Q}(t), N)$
CI	$C(s(t),1)+(4,0)$	$C(\boldsymbol{Q}(t),N)+(3N,0)$
OP	$C(s(t),1)+(16n^2+36n+10,1)$	$C(\boldsymbol{Q}(t),N)+(3N,0)$
ST	$C(s(t),1)$	$C(s(t),\gamma N)+(\gamma N,0)+C(\boldsymbol{Q}(t),N)$
CC	$C(\boldsymbol{Q}(t),\eta)+(2\eta,\eta)$	$C(\boldsymbol{Q}(t),N)+(2N,0)$

Table 2. Multiplication count during the setup.

	$n=3$	$n=5$	$n=9$
CI	$4+127=131$	$4+223=227$	$4+415=419$
OP	$262+127=389$	$590+223=813$	$1630+415=2045$
ST	127	223	415
CC	$676+96=772$	$1162+96=1258$	$2134+96=2230$

Table 3. Function-call count during the setup.

	$n=3$	$n=5$	$n=9$
CI	$9+0=9$	$9+0=9$	$9+0=9$
OP	$9+1=10$	$9+1=10$	$9+1=10$
ST	9	9	9
CC	$0+48$	$0+48$	$0+48$

Table 4. Multiplication count during $\boldsymbol{R}$ generation.

	$n=3$	$n=5$	$n=9$
BPF	1404	2410	4422
CI	$1404+300=1704$	$2410+300=2710$	$4422+300=4722$
OP	$1404+300=1704$	$2410+300=2710$	$4422+300=4722$
ST	$30000+400+1404=31804$	$52400+400+2410=55210$	$97200+400+4422=102022$
CC	$1404+200=1604$	$2410+200=2610$	$4422+200=4622$

Table 5. Function-call count during $\boldsymbol{R}$ generation.

	$n=3$	$n=5$	$n=9$
ST	2000	2000	2000

(g) The cost expressions is represented as follows:

$$C(\Psi,\beta) = (\text{\# of multiplications},\ \text{\# of function calls}),$$

where $C(\Psi,\beta)$ reads: *the cost of evaluating* β *times the expression* Ψ. The divisions required in the evaluation of Ψ are counted as multiplications; the number of function calls is a count of calls made to functions such as the square-root function, logarithmic functions, etc. The operations of additions and subtractions are not considered.

From the above tables, the following is concluded:

(a) The CI algorithm has a smaller setup time when compared with the others. In the cases where the arclength has been pre-computed, the setup time is negligible, whereas the OP and the CC have relatively larger setup times, depending on the values of n and η, respectively, as indicated in Table 2.

(b) From Tables 4 and 5, it is evident that the ST algorithm is the most computationally intensive algorithm to be used for locating points, both for the number of multiplications, and the number of function calls that it makes. By comparison, the other algorithms have low and similar operation counts, especially as n increases.

6. Discussion on Curves of Different Orders

In this section, the justification behind the choice of cubic degree polynomials represented as Bézier curves is presented. The approximation of the arclength parameterization of higher order curves is also shed light on.

6.1. Cubic Bézier Curves

Many real-world objects are inherently both complex and smooth, and much of computer graphics involve modeling either already existing real-world objects (such as faces, mountains, maps) or designing objects “from scratch” (such as fuselages, highways, etc.).

Because the classes of curves that are generated by an n-th-degree polynomial are a superset of those generated by a lower degree polynomial, additional flexibility can be attained by using higher-degree curves. Figure 6 illustrates this flexibility concept with some cubic Bézier curves. It should be noted that it will take *at least* two quadratic Bézier curves in order to obtain any of the shapes in Fig. 6. Generally, parabolic third-order curves cannot model spatial data that contain loops, cusps, or inflexion points inherent to them.

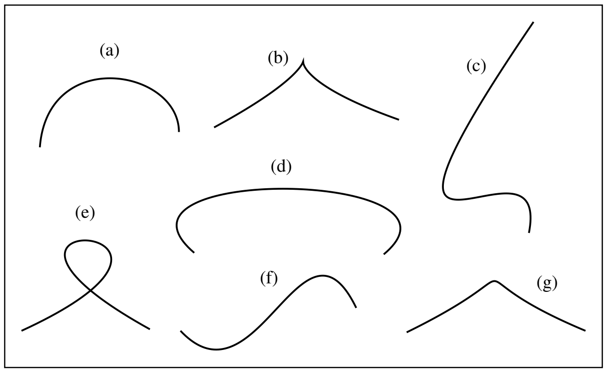

Fig. 6. One-segment cubic Bézier curves.

All the curves in Fig. 6 are one-segment cubic Bézier curves. These curves reflect the flexibility and *long-reach* of the modeling capability of cubic, Bézier curves. Figures 6(a) and (d) show different C-curves; Figs. 6(b) and (e) exhibit a cusp and a loop, respectively; Fig. 6(c) shows how an object like a nose may be modeled; Figs. 6(c) and (f) show forms of S-curves and therefore each has an inflection point; and finally, Fig. 6(g) shows a mountain-shaped object that has two inflection points.

Figure 7 shows a cubic Bézier curve with its control polygon shown as thin lines. By shifting the position of any of the control polygon points, different shapes are obtained as illustrated in Fig. 7.

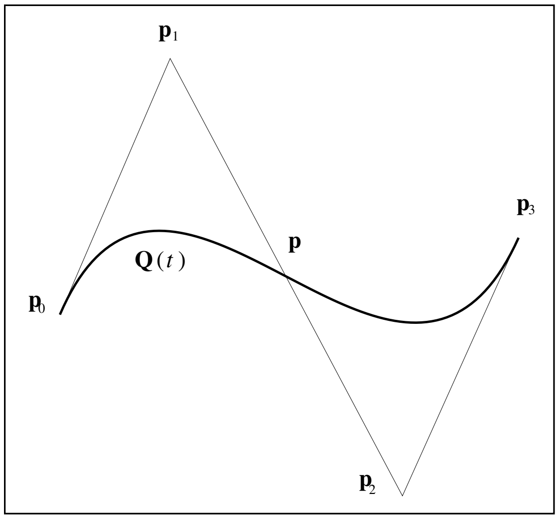

Fig. 7. Cubic Bézier S-curve.

6.2. Approximation of High-Degree Curves

Since the CI depends largely on the tangent vectors at the end points to perform its approximation of arclength parameterization, it is expected to produce good visual results for quadratic and cubic curves. However, it also seems to perform reasonably well for quartic and quintic curves (as shown in the previous figures), perhaps because the behavior of the curve away from the end tangent vectors is still sufficiently influenced by the information at the end points. Figures 8 and 9 illustrate these observations for yet higher degree curves. It is emphasized, however, that typical design curves do not usually exceed a quintic degree.

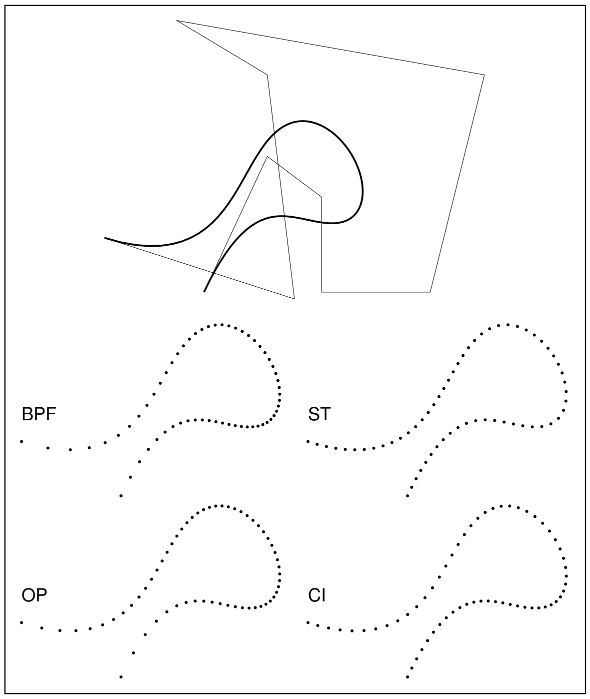

Fig. 8. Ninth-degree curve.

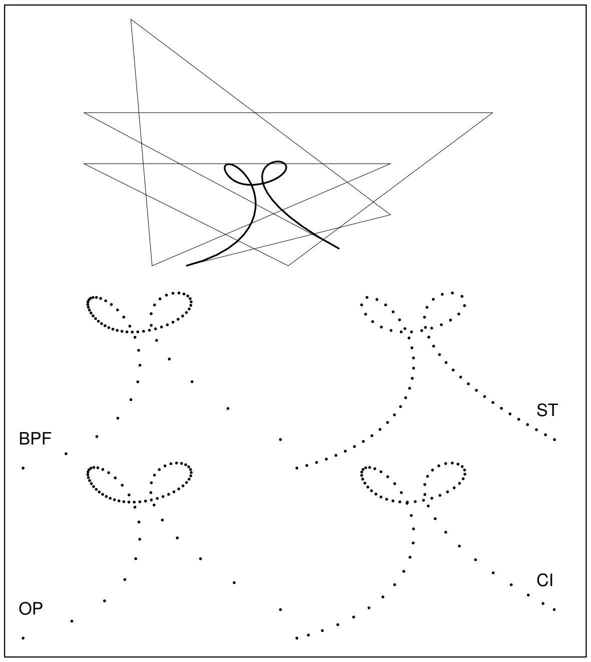

Fig. 9. Another ninth-degree curve exhibiting two loops.

Note that for curves that exhibit one or more regions of high curvature, interpolating with one cubic curve segment may not yield satisfactory results. Depending on the application and its precision requirements, one of the following two remedies may be applied:

- Use more than one cubic interpolator to approximate the whole curve. In this case, the curve may be divided into n segments using the basic parametric flow parameter t, and then each segment is approximated by a single cubic interpolator.
- The curve is subdivided into n segments at regions of high curvature to produce segments that are close in shape to linear curves. Each segment is then approximated by a single cubic interpolator. This method should produce far more better results since from both analytic and visual results the interpolator works best when the curves do not possess regions of high curvature, thereby producing reference points that are highly uniformly distributed across a curve. Further remarks about this method are provided in the concluding section.

7. Conclusion and Future Research

A method for approximating the intrinsic arclength parameterization for parametric curves has been proposed. The main property of the proposed method is that it depends on analytical expressions, as opposed to methods which depend on numerical techniques.

The accuracy and performance of the proposed method for approximating arclength parameterization, along with those of several other methods, were tested on the application of reference point generation. The task was to generate reference points that would be uniformly spaced along the path of parametric curves. It was shown that a cubic interpolation method produced results practical enough for most design curves, with a computational performance that is acceptable for real-time processing requirements.

Although the objective was to use a single entity in approximating when approximating the arclength parameterizable expression $t(s)$, a spline function (composed of several segments of the same definition of CI) can be used if both high accuracy, and real-time processing are required. Madi (1996) shows how an approximating spline function is constructed, and how accurate results are obtained with no more than eight segments.

Work in progress includes the following topics:

- Determine the number of Simpson intervals needed to achieve an acceptable accuracy in obtaining the arclength of a curve segment.
- Determine analytically how close the resulting $\boldsymbol{R}$ spacing is to the spacing that would result from an exact arclength parameterization. For example, in approximating $t(s)$ of the curve shown in Fig. 4, Fig. 10 shows a visual expression for the deviation of the approximating function $\mathcal{F}(s)$ from the true function $t(s)$, but this is just particular to that figure.
- Develop cost effective methods to recursively subdivide a curve at the center of a region of high curvature before having uniformly distributed reference points generated for it. This requires the analysis of the parametric flow behavior before and after the point along the curve at the center of the curvature region of interest.
- Investigate the performance of an interpolating function of a higher degree. For this, a quintic interpolator will be developed and analyzed against the cubic interpolator to determine whether there is a pay off by using more information at the end points.

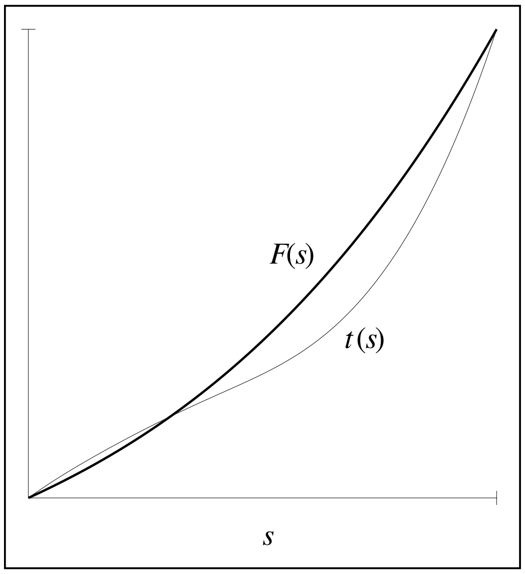

Fig. 10. Error in the approximation of $t(s)$ by $\mathcal{F}(s)$.

References

Burchard H.G., Ayers J.A., Frey W.H. and Sapidis N.S. (1994): *Approximating with aesthetic constraints*, In: Designing Fair Curves and Surfaces: Shape Quality in Geometric Modeling and Computer-Aided Design (N.S. Sapidis, Ed.). — Philadelphia: SIAM, pp. 3–28.

Davis P.J. (1963): *Interpolation and Approximation*. — New York: Blaisdell Publishing Company.

Farin G. (1993): *Curves and Surfaces for Computer-Aided Geometric Design: A Practical Guide*. — Boston: Academic Press.

Farouki R.T. (1992): *Pythagorean-hodograph curves in practical Use*, In: Geometry Processing for Design and Manufacturing, (R.E. Barnhill, Ed.). — Philadelphia: SIAM, pp. 3–33.

Farouki R.T. (1997): *Optimal Parameterizations*. — Computer Aided Geometric Design, Vol. 14, No. 2, pp. 153–168.

Farouki R.T. and Sakkalis T. (1991): *Pythagorean hodographs*. — IBM J. Res. Development, Vol. 34, No. 5, pp. 736–752.

Farouki R.T. and Shah S. (1996): *Real-time CNC interpolators for Pythagorean-hodograph curves*. — Computer Aided Geometric Design, Vol. 13, No. 7, pp. 583–600.

Foley J.D., Van Dam A., Feiner S.K. and Hughes J.F. (1992): *Computer Graphics: Principles and Practice*. — Reading: Addison Wesley.

Guggenheimer H.W. (1963): *Differential Geometry*. — New York: McGraw-Hill, pp. 15–17.

Madi M.M. (1996): *Arclength Approximation for Reference-Point Generation*. — M.Sc. Thesis, Dept. of Computer Science, University of Manitoba.

Sharpe R.J. and Thorne R.W. (1982): *Numerical method for extracting an arclength parameterization from parametric curves*. — Computer-Aided Design, Vol. 14, No. 2, pp. 79–81.

Su B-Q. and Liu D-Y. (1989): *Computational Geometry: Curve and Surface Modeling*. — Boston: Academic Press.

Young E.C. (1993): *Vector and Tensor Analysis*. — New York: Marcel Dekker.

Received: 21 January 2003
Revised: 7 December 2003
Re-revised: 17 February 2004