Using the Influence of Curve Tangent Vectors to Generate Approximately Uniformly Distributed Reference Points

Mohsen Madi
Department of Computer Science & GIM Research Group
University of Sharjah
Sharjah, UAE, POBox 27272
mmadi@sharjah.ac.ae

ABSTRACT

The need to systematically generate sets of reference points with prescribed arclengths along parametric curves, with accuracy and real-time performance usually arises in applications related to CNC machining, highway and railway design, manufacturing industry, and animation. Mechanisms to produce a parameter set that yield the coordinates of prescribed reference points along the curve $\mathbf{Q}(t) = \{x(t), y(t)\}$ are therefore sought. Arclength parameterizable expressions usually yield the parameter set that is necessary to generate the reference points; however, for typical design curves, such expressions are often not available in closed form. It is desirable to find efficient ways to compensate for lack of arclength parameterization. In this paper, several methods for approximating arclength parameterization are studied. These methods are examined for both accuracy and real-time processing requirements. The paper also introduces a numerical interpolation technique for a cubic interpolator function; the interpolator exploits the influence of end point tangent vectors to generate approximately uniformly distributed reference points as an example application.

Keywords:
Arclength Parameterization, Hermite Interpolation, Reference Points, Bézier Curves.

1 INTRODUCTION

The need to generate sets of reference points ($\mathbf{R}$'s) along paths of mechanical tools or parts is present in CAD and CAM applications. For convenience, reference points are referred to as $\mathbf{R}$'s and the i*th* reference point is designated as $\mathbf{R}_i$. As an example, in CNC machining, computers with CAD systems might be instructed to produce thousands of $\mathbf{R}$'s along the path of a manufactured part (such as the fuselage of an air- plane, where uniform spacing between adjacent reference points is desired to minimize tension) according to specific prescribed locations [Frk92a, Frk96a, Sha82a]. The manipulated part may be required to be affixed to other complementing parts by bolts through adjacent holes, in locations marked by reference points.

Conference proceedings, ISBN 80-86943-03-8
WSCG'2006, January 30-February 3, 2006
Plzen, Czech Republic.

A first step towards generating a set of N $\mathbf{R}$'s with prescribed arclengths is to abstract the physical paths along the object of interest by a *parametric* curve $\mathbf{Q}(t)$ in *Bernstein-Bézier* representation [Far93a]. 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 $\{s_{i=1,\ldots,N}\}$, $s_i \in (0, \mathcal{L}]$, at which the $\mathbf{R}$'s are to be generated.

(d) Re-parameterize $\mathbf{Q}(t)$ 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 design and analysis of curves and surfaces [Bur94a, Gug63a, You93a]. If a closed-form solution is available for items (a) and (d) above, the coordinates of an $\mathbf{R}$ that lie at arclength s_i along $\mathbf{Q}(t)$ may accurately be obtained by first evaluating $t_i = t(s_i)$, and then evaluating $\mathbf{Q}(t)$ at $t = t_i$.

In general, however, because of the non-linearity of the integral expression $s(t)$, 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 [Sha82a, Frk91a, Frk91b, You93a]. 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 to existing methods to actual arclength parameterization, is discussed.

A class of curves known as Pythagorean-hodographs have closed form expressions for their arclengths; however, they still require the solution of a non-linear equation to obtain the parameter as a function of the arclength [Frk91a]. This article is concerned with a more general class of polynomial curves.

The rest of the paper is organized as follows. Sections 2 and 3 present fundamental mathematical preliminaries and related work of several techniques that are geared to addressing approximation of arclength parameterization. Section 4 presents the Cubic Interpolator, an analytical method that embodies Hermite Interpolation to approximate arclength parameterization. This method is also introduced in [Mad04a] and is presented here for convenience. Section 5 shows Experimental Results of all presented techniques presented in a visual comparative fashion for the reader. In Section 6, a new cubic-spline Interpolation Function is derived to produce numerically accurate results when the role of real-time performance is down-played. The paper is concluded in Section 7.

2 MATHEMATICAL PRELIMINARIES

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

$$\mathbf{Q}(t_{0\leq t\leq 1}) = \sum_{i=0}^{n} \mathbf{p}_i \begin{pmatrix} n \\ i \end{pmatrix} (1-t)^{n-i} t^i. \qquad (1)$$

Properties and importance of such a representation to CAD/CAM are mentioned in the literature [Far93a, Qin89a]. In the above equation, n denotes curve degree, and $\mathbf{p}_i \in I\!E^2$ are the Bézier points that constitute the control polygon of the curve.

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

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

where $\mathbf{Q}'(t)$ is the derivative of $\mathbf{Q}(t)$. The total arclength of $\mathbf{Q}(t)$ 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 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 uniformly spaced intervals for smooth appearance, it is the performance rather than the accuracy that is of importance [Mad96a]. In the remainder of this section, an overview of existing methods is presented.

3.1 Basic parametric Flow (BPF)

The simplest method for $\mathbf{R}$'s generation may be called the basic parametric flow (BPF), since a number of N points are produced by uniform parameter spacing (e.g., $t_{i=0,\ldots,N} = i/N$). Although simple and fast, it is well known that this method is not suitable for generating points along the arclength of a curve [Frk97a, Mad96a].

3.2 Sharpe & Thorne (ST)

The method described by Sharpe and Thorne can accurately produce $\mathbf{R}$'s at prescribed arclengths [Sha82a].

However, it has a high computational cost associated with "extracting" the corresponding parametric value for each $\mathbf{R}$ to be generated. Consider the following non-linear equation used to find t_i, the parametric value needed to generate $\mathbf{R}_i$:

$$M(t) = \int_{t_{i-1}}^{t} \sqrt{\mathbf{Q}'(\tau) \cdot \mathbf{Q}'(\tau)}\, d\tau - s_i = 0, \quad (3)$$

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

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

where $j = 1, 2, \ldots, k$, and $M'(\tau_j)$ 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 $\mathbf{R}$'s, this method may be impractical.

3.3 Optimal Parameterization (OP)

Farouki's OP is mathematically a rather intricate process [Frk97a]. The given polynomial curve $\mathbf{Q}(t)$ is first transformed into an equivalent rational form by transforming the parameter t in (1) (by applying a Möbius transformation) as follows:

$$t = \frac{(1-\alpha)u}{\alpha(1-u) + (1-\alpha)u}, \quad (5)$$

with $0 < \alpha < 1$ and $0 \leq u \leq 1$. Substituting (5) into (1) results in the following rational form:

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

where $w_i = (1-\alpha)^i \alpha^{n-i}$. 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 [Mad96a].

3.4 Cumulative Chordlength (CC)

Cumulative chordlength is a straightforward method to approximate the arclength of a curve. This method can be exploited to generate $\mathbf{R}$'s that visually seem to be uniformly spaced.

The algorithm is as follows: while computing the arclength, the set $\{s_k \,|\, k = 0, 1, \ldots, \eta\}$ (η being the number of chords used to approximate the curve) keeps track of the cumulative chordlength thus far. $\mathbf{R}$'s at distances $\{i\Delta d \,|\, i = 0, \ldots, N;\ \ \Delta d = s_\eta / N\}$, 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 linear interpolation. That is, t_i, the parametric value corresponding to $\mathbf{R}_i$, at distance $i\Delta d$, is approximated by the function $A(i,k)$ as follows:

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

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

While increasing the number of chords approximating $\mathbf{Q}(t)$ *may* increase accuracy, the size of the curve-dependent data-table that has to be maintained also increases [Mad96a].

4 A Cubic Interpolator (CI)

Hermite interpolation [Dav63a, Fol92, Far93a] is used to approximate $t(s)$ for any parametric curve $\mathbf{Q}(t)$. The cubic interpolator (CI) is defined as follows:

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

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

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

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

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

Upon integration of (10), the following results:

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

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 $t(s)$ are obtained as follows:

$$\begin{bmatrix} t(0) \\ t'(0) \\ t(\mathcal{L}) \\ t'(\mathcal{L}) \end{bmatrix} = \begin{bmatrix} 0 \\ \frac{1}{\|\mathbf{Q}'(0)\|} \\ 1 \\ \frac{1}{\|\mathbf{Q}'(1)\|} \end{bmatrix}. \qquad (12)$$

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

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

To illustrate the low cost of generating $\mathbf{R}$'s using the CI method, the CI algorithm shown next is an implementation of the method described above ([Mad96a] gives details on implementation and 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 evaluation of $\{\mathbf{Q}(\mathcal{F}_i)\}$ 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 L, scale ||Q'(0)|| and ||Q'(1)||
    c ← 1/||Q'(0)||
    a ← c + 1/||Q'(1)|| − 2
    b ← 1 − c − a
    ΔL ← 1/N,  ℓ_0 ← 0
    for i ← 1 to N
        ℓ_i ← ℓ_{i−1} + ΔL
        F_i ← ((aℓ_i + b)ℓ_i + c)ℓ_i
        R_i ← Q(F_i)
END
```

Scaling of $\|\mathbf{Q}'(0)\|$ and $\|\mathbf{Q}'(1)\|$ implies dividing them by $\mathcal{L}$; this scales the whole curve such that $\mathcal{L} = 1$. Note that $\mathcal{F}_i$ above is simply Eq. (8) evaluated using Horner's less costly method.

Note that a precise measure of the deviation from a true uniform distribution is obtainable. Consider, for example, the quintic PH curve of Figure 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 the following two lines to the for-loop.

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

Inversely, the distribution is 94.13% accurate, which is a huge achievement over the basic parametric flow (BPF()) parameterization yielding only 69.32%, at more or less the same cost.

5 Experimental Results

The objective of this section is to show how close the $\mathbf{R}$'s generated by the various methods are to the exact $\mathbf{R}$'s (note that 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 $\mathbf{R}$'s. Figures 1 to 4 show the result of applying the algorithms discussed to a sample of curves; a much larger sample of curves is found at [Mad96a].

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

The ST reference points are considered to be exact (6 to 8 digits of accuracy) and thus are used to compare other results with. Because CC (cumulative chord-length) plots are *visually* indistinguishable from ST plots, they are not shown ([Mad96a] discusses CC and CC plots in more detail).

For symmetric curves (e.g., Figures 1 and 4), the value of α in OP's algorithm is $1/2$, thereby producing results exact to those of BPF [Mad96a, Frk97a]. These and other figures show the closeness of CI results to those of ST, obtained at a considerably lower cost than required by ST.

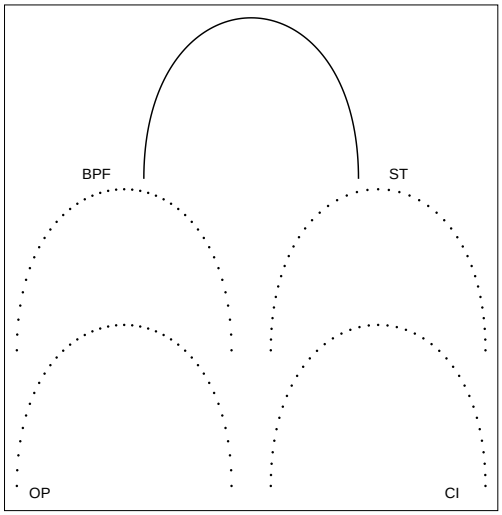

Figure 1: A cubic PH curve.

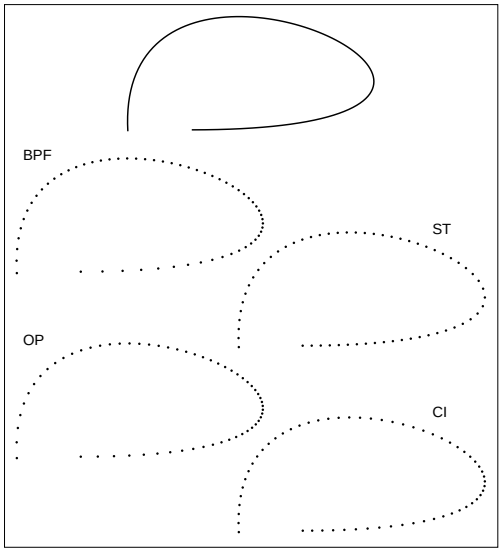

Figure 2: A cubic Bézier with high curvature regions.

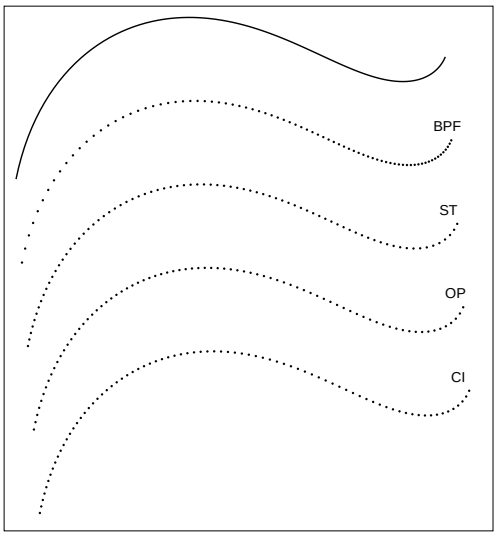

Figure 3: A quintic PH curve.

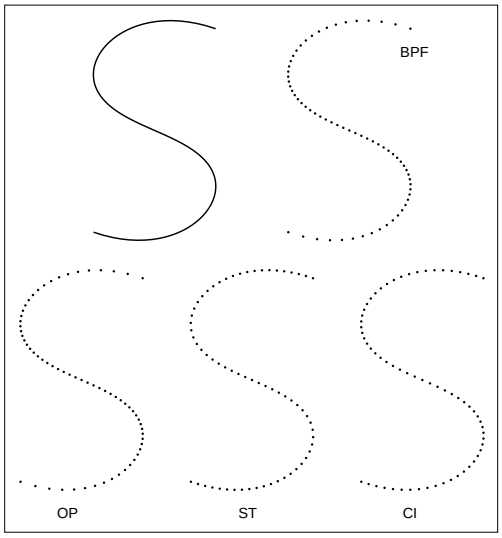

Figure 4: A quintic S-curve.

Refer to [Mad96a, Mad04a] for detailed analysis and tabulated statistics about the run-time cost of each algorithm.

6 USING THE CUBIC-SPLINE INTERPOLATION FUNCTION

In developing the Cubic Interpolator in Section 4, the emphasis was on obtaining an expression for $t(s)$ (that is, t, the parameter of the curve $\mathbf{Q}(t)$, as a function of s, the arclength of the curve) in an analytical fashion, such that, when evaluated at $s_k \in [0, \mathcal{L}]$, the corresponding $t_k \in [0, 1]$ results. It is desirable, for manipulation in mathematical experiments, that it is a single function rather than a piecewise function.

The objective of CI is to make available an approximation to $t(s)$. This is similar to some of the objectives sought in [Sha82a, Gue90a, Frk92a, Frk97a]. However, it is noted that some of these methods depend on analytical expressions, such as the CI or the F-OP algorithms, whereas other methods, such as the ST algorithm, depend on numerical methods to approximate $t(s)$. The first kind of approximators has the advantage of not having to compute and maintain arrays of numbers, or use quadrature techniques to reach a satisfactory result; it is not equitable to compare their accuracy to those that depend on numerical methods. The latter have the advantage that any prescribed accuracy can be obtained by increasing the number of approximating segments to refine results in accordance with some prescribed tolerance, or increasing the number of Newton-Raphson iterations.

In this section, it is shown how the proposed method may be modified so that it also uses numerical techniques to generate reference points (RPs) along parametric curves, thereby making use of the advantages that numerical methods have. The numerical version of the proposed method is called NCI, or, the numerical CI. Its objective is to generate RPs with accuracy and performance comparable to methods which depend on numerical techniques.

6.1 The NCI Derivation & Algorithm

The idea behind developing the NCI is similar to that behind the CC method in Subsection 3.4: the curve $\mathbf{Q}(t)$ is first approximated by $\Im$ segments. The cumulative arclength is calculated at the end of the kth segment, $k = 1, 2, \ldots, \Im$, along with the coefficients a_k, b_k, c_k, and d_k. Because each segment is approximated over two Simpson intervals (using Simpson's rule), $\Im$ is always $I/2$, where I is an even number of Simpson intervals used to approximate the $\mathbf{Q}(t)$. This is feasible because Simpson's rule may give the intermediate arclengths at every second function. That is, the arclength of the kth segment may be determined by

$$s_k = \frac{1}{3I}(\alpha_{2k-2} + 4\alpha_{2k-1} + \alpha_{2k}) + s_{k-1} \qquad (14)$$

where $k = 1, 2, \ldots, \Im$, $s_0 = 0$, and $\alpha_j = \|\mathbf{Q}'(j/I)\|$. Note that $\mathcal{L} = s_{\Im}$. To determine the coefficients for each segment, Equation (3.27) is rewritten in the following manner:

$$f_k(s) = a_k s^3 + b_k s^2 + c_k s + d_k, \qquad (15)$$

with derivative

$$f'_k(s) = 3a_k s^2 + 2b_k s + c_k, \qquad (16)$$

where $s_{k-1} < s \leq s_k$. For each segment, four equations are required to solve for the four unknowns. Following the same methodology used in Section 3.3.1 (i.e., to formulate four equations by equating each of $f_k(s)$ and $f'_k(s)$ at $s = s_{k-1}$, and $s = s_k$, with the appropriate values of $t(s)$, and $t'(s)$, respectively). Let

$$\begin{aligned} f_k(s_{k-1}) &= \frac{k-1}{\Im}, \\ f_k(s_k) &= \frac{k}{\Im}, \end{aligned}$$

and

$$\begin{aligned} f'_k(s_{k-1}) &= \frac{1}{\alpha_{2k-2}}, \\ f'_k(s_k) &= \frac{1}{\alpha_{2k}}, \end{aligned}$$

the four equations are formulated as follows:

$$\begin{aligned} a_k s_{k-1}^3 + b_k s_{k-1}^2 + c_k s_{k-1} + d_k &= (k-1)/\Im, \\ a_k s_k^3 + b_k s_k^2 + c_k s_k + d_k &= k/\Im, \\ 3a_k s_{k-1}^2 + 2b_k s_{k-1} + c_k &= 1/\alpha_{2k-2}, \\ 3a_k s_k^2 + 2b_k s_k + c_k &= 1/\alpha_{2k}. \end{aligned}$$

The coefficients a_k, b_k, c_k, and d_k may now be solved for, and are as follows:

$$\begin{aligned} a_k &= \frac{1}{(s_k - s_{k-1})^2}\left(\frac{\alpha_{2k-2} + \alpha_{2k}}{\alpha_{2k}\alpha_{2k-2}} - \frac{2}{\Im(s_k - s_{k-1})}\right), \\ b_k &= \frac{\alpha_{2k-2} - \alpha_{2k}}{2\alpha_{2k}\alpha_{2k-2}(s_k - s_{k-1})} - \frac{3}{2}a_k(s_k + s_{k-1}), \\ c_k &= \frac{1}{\alpha_{2k}} - 3a_k s_k^2 - 2b_k s_k, \\ d_k &= \frac{k}{\Im} - (a_k s_k^3 + b_k s_k^2 + c_k s_k). \end{aligned}$$

The complete algorithm follows.

```
NCI()
    compute α_{i←0,1,...,I},  s_{k=0,1,...,ℑ}
    scale α_i, s_k  so that  s_ℑ  is unity
    compute a_k, b_k, c_k, and d_k, k = 0, 1, ..., ℑ
    ΔL ← 1/N,  k ← 0,  ℓ_0 ← 0
    for i ← 1 to N − 1
        ℓ_i ← ℓ_{i−1} + ΔL
        while ℓ_i > s_k
            k ← k + 1
        f_i ← ((a_k ℓ_i + b_k)ℓ_i + c_k)ℓ_i + d_k
        r_i ← Q(f_i)
END
```

6.2 Visual Results

In this subsection, some RP plots are shown for both the NCI and the CC method, where both will be compared against RPs generated by the ST method. The purpose is to show how the NCI produces better results with fewer segments than is usually required for the CC, and does less computation than is required by the CC, or the ST method. In Figure 5, the circles denote RPs generated by the ST algorithm, while the dots denote RPs generated by the CC and the NCI algorithms. The number next to the algorithm name on each plot indicates the number of chords, segments used by the CC, and the NCI algorithms, respectively.

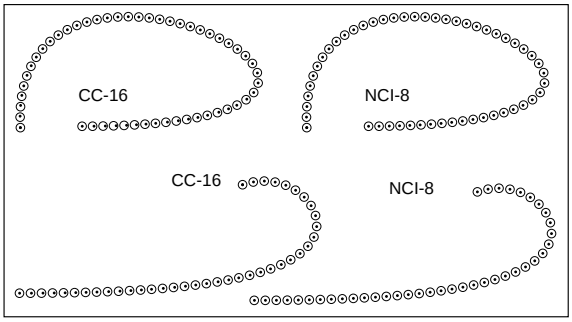

Figure 5: Comparative plots with CC and NCI.

7 Conclusion & Future Research

A survey of methods for approximating the intrinsic arclength parameterization for parametric curves has been presented. The main property of one of the feature methods, the CI, is that it depends on analytical expressions influenced by the tangent vectors at the curve end points, as opposed to methods which depend on numerical techniques. The main advantage of CI is that it is suitable for real-time applications. When the prescribed accuracy to generating the Reference Points is sought over the speed in generating the reference points, a new numerical method, NCI, which depends largely on numerical techniques is derived out of CI and is shown to produce accurate results in a competitive number of iterations.

Work currently in progress iterates over the following points.

- Determine the number of Simpson intervals needed to achieve acceptable accuracy in obtaining the arclength of a curve segment.
- 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.

References

[Bur94a] Burchard, H.G., J.A. Ayers, W.H. Frey and N.S. Sapidis, Approximating with Aesthetic Constraints, In *Designing Fair Curves and surfaces: Shape Quality in Geometric Modeling and Computer-Aided Design*, (N.S. Sapidis, ed.), SIAM, Philadelphia, pp.3–28, 1994.

[Dav63a] Davis, P.J., *Interpolation and Approximation*, Blaisdell Publishing Company, New York, 1963.

[Frk91a] Farouki, R.T. and T. Sakkalis, Pythagorean Hodographs, In *IBM Journal of Research and Development*, Vol. 34, No. 5, pp.736–752, 1991.

[Frk91b] Farouki, R.T. and T. Sakkalis, Real rational curves are not ”unit speed”, In *Computer Aided Geometric Design*, Vol. 8, No. 2, pp.151–157, 1991.

[Frk92a] Farouki, R.T., Pythagorean-hodograph Curves in Practical Use, In *Geometry Processing for Design and Manufacturing*, (R.E. Barnhill, ed.), SIAM, Philadelphia, pp.3–33, 1992.

[Far93a] Farin, G., *Curves and Surfaces for Computer-Aided Geometric Design: A Practical Guide*, Academic Press, Boston, 1993.

[Frk96a] Farouki, R.T. and S. Shah, Real-time CNC interpolators for Pythagorean-hodograph curves, In *Computer Aided Geometric Design*, Vol. 13, No. 7, pp.583–600, 1996.

[Frk97a] Farouki, R.T., Optimal Parameterizations, In *Computer Aided Geometric Design*, Vol. 14, No. 2, pp.153–168, 1997.

[Fol92] Foley, J.D., A. Van Dam, S.K. Feiner and J.F. Hughes, *Computer Graphics: Principles and Practice*, Addison Wesley, Reading, Massachusetts, 1992.

[Gue90a] Guenter, B. and R. Parent, Computing the Arc Length of Parametric Curves, In *IEEE Computer Graphics and Applications*, Vol. 10, No. 3, pp.72–78, 1990.

[Gug63a] Guggenheimer, H.W., *Differential Geometry*, McGraw-Hill, New York, 1963.

[Mad96a] Madi, M.M., *Arclength Approximation for Reference-Point Generation*, Master’s thesis, Dept. of Computer Science, University of Manitoba, June 1996.

[Mad04a] Madi, M.M., Closed-Form Expressions for Approximation of Arclength Parameterization for Bézier Curves, In *Int. J. Appl. Math. Comput. Sci.*, Vol. 14, No. 1, pp.33–41, 2004.

[Sha82a] Sharpe, R.J. and Richard W. Thorne, Numerical Method for Extracting an Arclength Parameterization from Parametric Curves, In *Computer-Aided Design*, Vol. 14, No 2, pp.79–81, 1982.

[Qin89a] Su, B-Q. and D-Y. Liu, *Computational Geometry: Curve and Surface Modeling*, Academic Press, Boston, 1989.

[You93a] Young, E.C., *Vector and Tensor Analysis*, Marcel Dekker, New York, 1993.