Symbolic and Numeric Computation in Curve Interrogation

Gershon Elber
Computer Science Department
Technion, Israel Institute of Technology
Haifa 32000
Israel

Abstract

The control of shape of curves is of great importance in computer aided geometric design. Determination of planar curves' convexity, the detection of inflection points, coincident regions, and self intersection points, the enclosed area of a closed curve, and the locations of extreme curvature are important features of curves that can affect the design, in modeling environments.

In this paper, we investigate the ability to robustly answer the above queries and related questions using an approach which exploits both symbolic computation and numeric analysis.

Key Words: Convexity, Curvature, Inflection points, Arc-length parameterization, Enclosed Area, Self-intersection.

1 Introduction

Freeform surfaces can be constructed in various ways. Predefined primitives and mechanical features can be represented using freeform surfaces. High level surface constructors, such as sweep, extrude, warp and bend [1], constraint based modeling [2, 3, 4, 5, 6], and low level direct manipulation of control points are all methods of surface construction and manipulation. In all those techniques, curves are used as intermediate representations in the process of a surface construction. Surfaces are frequently created by interpolating or approximating a set of curves. These curves can be explicitly supplied by the user, or can be implicitly created by the modeling system, when a high level surface constructor is invoked. It is therefore important to be able to create curves with shape controlled by constraints such as convexity, curvature, enclosed area, and no self intersections or inflection points.

The NURBs and Bézier representations are variation diminishing [7]. In general, if a control polygon is convex or has no inflection points, so is the curve itself. However, the control polygon can have inflection points while the curve is completely convex or concave [8, 9] (see Figure 1). Exact conditions whether

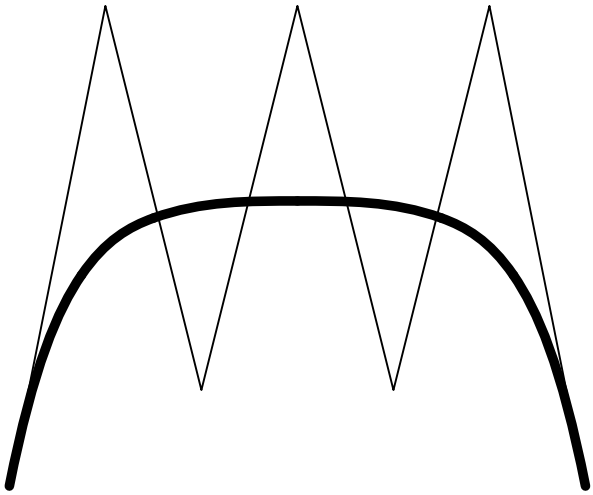
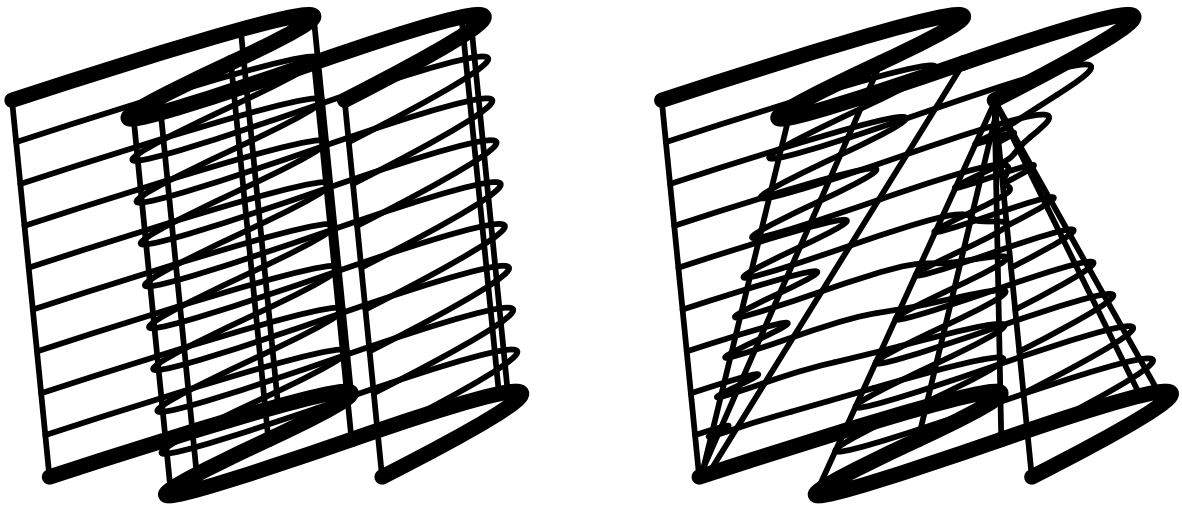


Figure 1: A Bézier curve can be completely convex while its control polygon has an inflection point.

a curve is convex or concave, or the ability to detect and subdivide it at all its inflection points into completely convex or concave regions, are of importance.

A parametric curve $C(t)$ is considered regular if $\|dC(t)/dt\| \neq 0$ [10]. A parametric curve $C(s)$ is considered arc-length parameterized if $\|dC(s)/ds\| = 1$. By reparameterizing a curve and creating an approximation to an arc-length parameterization, the correspondence between adjacent curves can be more easily established, which, in turn, can simplify the use of these curves in further manipulation and construction (see Figure 2). A constant step in s traces a constant distance along an arc-length parameterized curve, $C(s)$, a desired property for making sure that an end mill is moving at a steady speed during a machining operation or that the curve is rendered at every pixel the curve covers on the screen if curves are used as a valid coverage for surfaces [11]. Unfortunately, arc-length reparameterization is a difficult, and in most cases impossible in the polynomial and rational domains [10].

Detection of self intersection of curves, intersection points of curves, and regions of coincidence in different curves is another set of important but difficult problems [12, 13, 14]. A curve that intersect itself cannot, in general, be used to construct a meaningful surface. The set of (self) intersection points of curves is of importance in several applications from Boolean operations and machining toolpath generation to hidden curve removal of freeform surfaces [15]. The detection of regions of coincidence of two curves with different speeds that are tracing the same curve in space is difficult. The two curves can have different polynomial representation, even if they are being elevated to the same function space, that is the same



(a) **(b)**

Figure 2: A ruled surface between two curves that do share the same speed (a) and a ruled surface between two curves that do not share the same speed (b). The two sets of curves trace the same curves in space.

order and the same continuity (knot vectors).

In this paper, we present methods to examine the above problems robustly using a combination of symbolical and numerical analysis. Scalar and vector fields are symbolically computed and analyzed with the aid of numerical techniques. The ability to symbolically represent differentiation, integration, addition, subtraction, product and composition of (piecewise) scalar polynomial and rational curves and represent the result in the same (piecewise) polynomial or rational domains is utilized. In [7, 16, 17, 18, 19], techniques to compute and represent these operations using the Bézier and NURBs basis function in the (piecewise) polynomial and rational domains are discussed. Constant sets of given scalar fields represented as Bézier or NURBs curves or surfaces will be computed using contouring techniques similar to the ones used in [16].

In section 2, curvature properties of planar curves are discussed. Queries on the existence of inflection points and the convexity or concavity of a curve are answered, as well as the ability to detect and isolate regions of curves with curvature values larger than prescribed, or the points of extreme curvature. In section 3, we attempt to approximate, to within a prescribed tolerance, a unit length vector field curve from an unnormalized vector field curve. This technique is also used to approximate the arc length $s(t)$ of a regular curve $C(t)$. In section 4, we use Green's theorem to represent the area enclosed by a

closed polynomial parametric curve symbolically. In section 5, the square of the distance between two curves $C_1(u)$ and $C_2(v)$, $\delta(u,v)$, is computed symbolically. By analyzing $\delta(u,v)$ we will be able to detect intersection points, self intersection points, and even coincident regions between the two curves.

All the figures in this paper were created using the experimental IRIT solid modeler, currently being developed at the Technion.

2 Curvature Analysis of Curves

Curvature properties of curves are important for proper design. Most existing research attempts to derive conditions on the behavior of the curve such as inflection points, cusps, and convexity by deriving the necessary and sufficient conditions on the control polygon of the curve. Unfortunately, deriving these condition is difficult and is usually for restricted cases only. In [20, 21] only cubic polynomial curves are considered, whereas in [22] only uniform B-spline cubic curves are discussed. A different approach is taken herein by deriving the curvature scalar field symbolically, so it can be directly analyzed.

Let $C(t)$ be a regular curve, that is $\|dC(t)/dt\| > 0$. The curvature $\kappa(t)$ of a $C(t)$ is equal to [10],

$$\kappa(t)N(t) = \frac{\frac{dC(t)}{dt} \times \frac{d^2C(t)}{dt^2}}{\left\|\frac{dC(t)}{dt}\right\|^3}, \tag{1}$$

where $\|\cdot\|$ denotes the magnitude of a vector (L^2 norm) and $N(t)$ is a unit vector in the curve's normal direction.

If $C(t) = (x(t), y(t))$ is a planar curve, the magnitude of equation (1) can be simplified into,

$$\kappa(t) = \frac{\frac{dx(t)}{dt}\frac{d^2y(t)}{dt^2} - \frac{d^2x(t)}{dt^2}\frac{dy(t)}{dt}}{\left(\left(\frac{dx(t)}{dt}\right)^2 + \left(\frac{dy(t)}{dt}\right)^2\right)^{\frac{3}{2}}}. \tag{2}$$

$\kappa(t)$ is not representable as a rational because of the square root term in the denominators of both equations (1) and (2). However, the square of both equation (1) and equation (2) is computable and representable as a piecewise rational scalar curve, provided $C(t)$ is a Bézier or a NURBs curve. Therefore, we consider $\chi(t) = \kappa^2(t)$. The sign of $\kappa(t)$ directly indicates the convexity or concavity of the curve and is the same as the sign of the numerator in both equations (1) and (2), because the denominators are strictly positive for a regular curve. We define the sign function,

$$\sigma(t) = \frac{dx(t)}{dt}\frac{d^2y(t)}{dt^2} - \frac{d^2x(t)}{dt^2}\frac{dy(t)}{dt}, \tag{3}$$

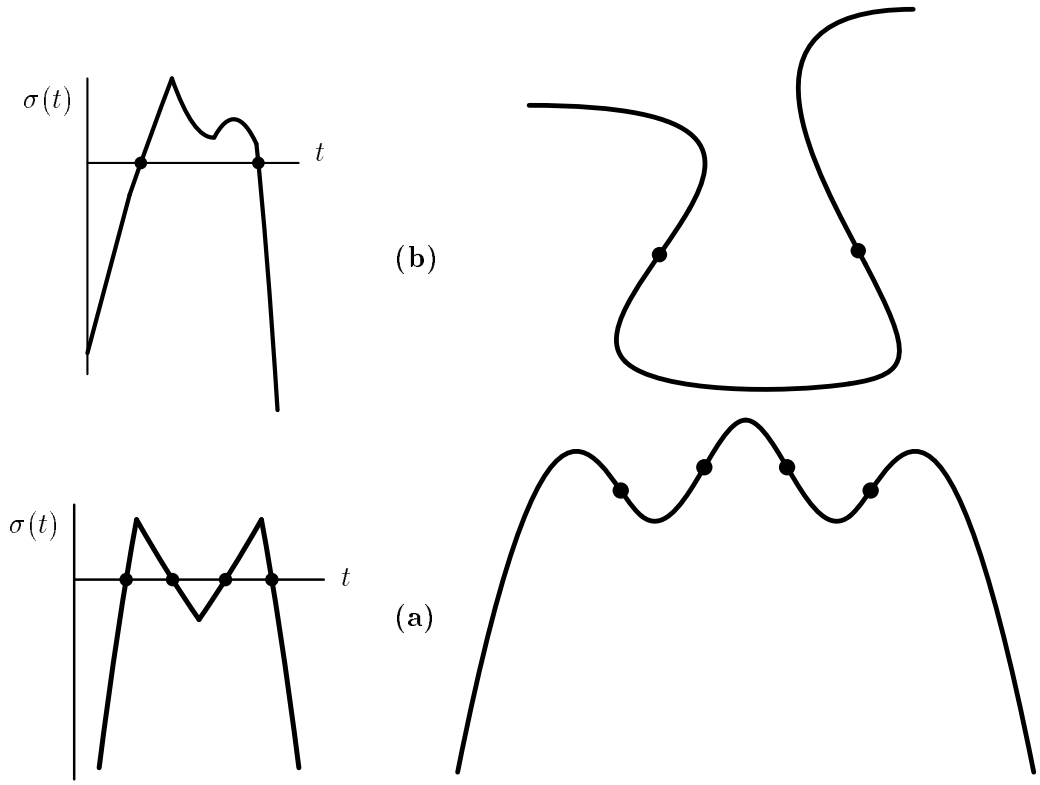


Figure 3: Two examples of detection of inflection points of cubic planar Bspline curves with interior knots, using symbolic approach. $\sigma(t)$, the curvature sign scalar field is also shown for the respective curves. $\sigma(t)$ is C^0 at the interior knots (of multiplicity one) of the cubic curve.

as a scalar field that preserves the sign of the curvature. $\sigma(t)$ is computable and representable as a polynomial or a rational scalar curve, as a difference between two products of scalar curves.

The zero set of $\sigma(t)$ provides the points of zero curvature of the planar curve $C(t)$. These points are the inflection points but also straight line segments of curve $C(t)$. Figure 3 shows examples of computing $\sigma(t)$ and using its zero set to isolate points of zero curvature.

The zero set of $d\chi(t)/dt$ provides all points of extreme curvature on the curve $C(t)$. Figure 4 shows the points of extreme curvature, for a cross section of a pawn using in [23], computed as the extrema points of $\chi(t)$.

Furthermore, by computing and representing $\chi(t)$ as a rational scalar field, one can not only detect the existence of regions of high curvature in the curve, but can also isolate them. Let $\mathcal{C}_d(t)$ be the offset curve of curve $C(t)$ by a distance d [23]. Clearly, if there exist regions in $C(t)$ such that $\kappa(t) > 1/d$, $\mathcal{C}_d(t)$

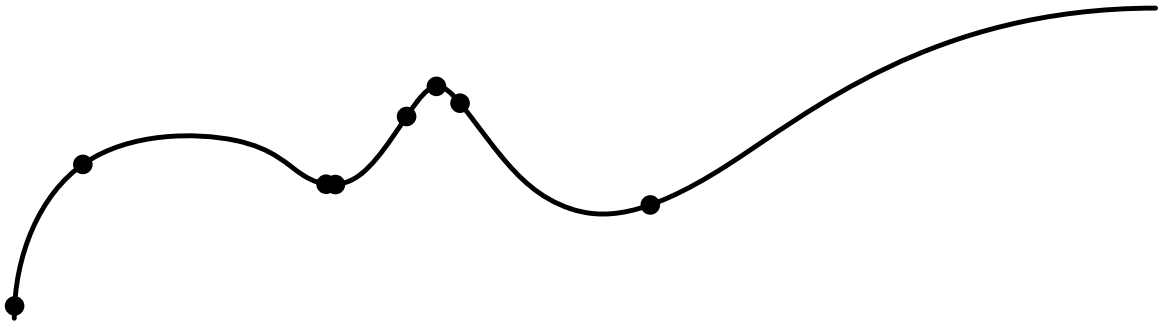


Figure 4: The points of extrema of $\chi(t) = \kappa^2(t)$ computed as the zero set of $\frac{d\chi(t)}{dt}$. This includes both points of maximum curvature and points of minimum curvature, including inflection points.

will self intersect [24, 25]. The $1/d^2$ constant set of $\chi(t)$ can be used to isolate the regions in $C(t)$ for which $\kappa(t) > 1/d$, if they exist. One can differentiate the $-1/d$ concave curvature points from the $1/d$ convex curvature points by inspecting the sign of $\sigma(t)$ at the points for which $\chi(t) = 1/d^2$. Figure 5 demonstrates this process on a cross section of a pawn used to demonstrate an adaptive offset approximation algorithm with global error bound that is described in [23]. By precomputing and trimming away the regions for which $\kappa(t) > 1/d$, the cusps formed by the offset are eliminated. Most of the computation in the adaptive offset approximation algorithm described in [23] is devoted to the approximation of these cusps, regions that are eventually eliminated. By preprocessing and eliminating the regions with cusps, the whole process of constructing an offset approximation with a globally bounded error, as described in [23], can be computationally improved. After the highly curved regions are detected and eliminated, the remaining curve segments should be further intersected and trimmed against each other using planar curve–curve intersection techniques [12, 13, 14].

3 Vector Field Normalization and Arc-length Reparameterization

Vector fields that are computed symbolically can be used to define tangent information in high precision. In [18, 26, 27], tangent information computed symbolically is used to construct cubic Hermite blend surfaces that are tangent to the blended surfaces with accuracy bounded only by the machine precision. The tangent vector field is easier to interpret geometrically if the vector field is normalized. Let $V(t)$ be an unnormalized vector field. We would like to be able to approximate the unit length vector field of

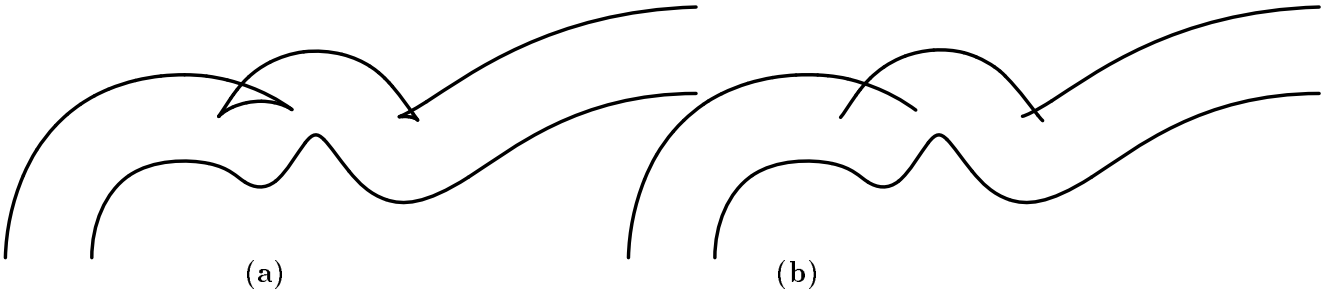


Figure 5: An offset to this cross section of a chess pawn has regions that self intersect due to large curvature (a). These regions can be detected and eliminated using symbolic methods that computes the curvature of the curve (b).

$V(t)$, $\hat{V}(t)$, to within a given tolerance, such that $\hat{V}(t)$ and $V(t)$ are collinear and $\|\hat{V}(t)\| \approx 1$. $V(t)$ is unit length if $\|V(t)\| = 1$. Although $\|V(t)\|$ is not representable as a polynomial or rational, in general, $\|V(t)\|^2$ is and is equal to,

$$m(t) = \|V(t)\|^2 = v_x^2(t) + v_y^2(t) + v_z^2(t). \tag{4}$$

Let $M(t)$ be a scalar field defined using $m(t)$, with the same order and continuity (knot vector), by substituting each coefficient k in the control polygon of $m(t)$ as $1/\sqrt{k}$ in the control polygon of $M(t)$.

The error between a C^2 continuous function and its Schoenberg variation diminishing spline approximation [28] over a knot vector $\{t_i\}$ is $O(|\{t_i\}|^2)$, where $|\{t_i\}| = max_i\{t_{i+1} - t_i\}$. By using a sequence of Schoenberg variation diminishing spline approximations to $\hat{V}(t)$, each one based on a knot vector that is a refinement of the previous one, and a sequence, $\{V^i(t)\}$, of refined representations of $V(t)$, based on the same sequence of knot vectors, we form a converging sequence of approximations to $\hat{V}(t)$. Using the refinement sequence $\{m^i(t)\}$, $m^i(t) = ||V^i(t)||^2$, the sequence $\{M^i(t)\}$ converges to $M(t) = 1/\|V(t)\|$. Let $\hat{V}^i(t) = V^i(t)M^i(t)$. By suitable refinement on $V(t)$, $\hat{V}^i(t)$ can converge as closely as needed to the exact unit length vector field. By inspecting $\hat{V}^i(t)$ we can not only determine the quality of the approximation but also figure out the regions in the curve that has error larger than allowed and apply refinement to only these regions.

An iterative algorithm that adaptively refines a vector field, $V(t)$, at the regions with large error and approximates a unit length vector field, $\hat{V}(t)$, with the desired tolerance τ so that $1+\tau > \left|\left|\hat{V}^i(t)\right|\right| > 1-\tau$, can now be derived.

Algorithm 1

Input:

vector field $V(t)$;

Tolerance τ;

Output:

$\hat{V}^i(t),\ 1+\tau > \left|\left|\hat{V}^i(t)\right|\right| > 1-\tau$;

Algorithm:

$i \Leftarrow 1$;

$V^i(t) \Leftarrow V(t)$;

do

$m(t) \Leftarrow ||V^i(t)||^2$, a scalar field computed symbolically;

(1) $M(t) \Leftarrow \frac{1}{\sqrt{m(t)}}$, a scalar field approximated numerically;

$\hat{V}^i(t) \Leftarrow V^i(t)M(t)$, computed symbolically;

$\xi(t) \Leftarrow \left|\left|\hat{V}^i(t)\right|\right|^2$, a scalar field computed symbolically;

(2) if (exist t, such that $(\xi(t) > (1+\tau)^2$ or $\xi(t) < (1-\tau)^2)$)

$V^{i+1}(t) \Leftarrow V^i(t)$ refined at the neighborhood of any such t;

$i \Leftarrow i+1$;

while (exist t, such that $(\xi(t) > (1+\tau)^2$ or $\xi(t) < (1-\tau)^2)$);

Curve (Figure 6)	Maximum error
Original $V(t)$ (a)	1.3714
3 new knots (b)	0.1561
6 new knots (c)	0.0385
12 new knots (d)	Below tolerance (0.01)

Table 1: Convergence of $\|\hat{V}^i(t)\|$ to a unit length using refinement.

The construction of $M(t)$ from $m(t)$ is the only numeric stage (1) in Algorithm 1. $M(t)$ can be constructed by sampling $m(t)$ at the node values of $M(t)$ and setting the coefficients of $M(t)$ to be $1/\sqrt{m(t)}$ at these points. As a Schoenberg variation diminishing spline, $M(t)$ will converge using refinement to the inverse of the square root of the magnitude, guaranteeing the convergence of the entire algorithm.

In step (2) of Algorithm 1, the convex hull property of the Bézier and NURBs representations can be exploited to compute bounds on $\xi(t)$, by inspecting the coefficients of $\xi(t)$

Figure 6 shows an example of this approach and demonstrates the effect of the refinement on the convergence. $M^i(t)$ were computed from the original quadratic Bspline curve, $V(t)$, which has one interior knot. In Figures 6(b) to (d), 3, 6, and 12 new knots were inserted, respectively, into the parametric domain of $V(t)$.

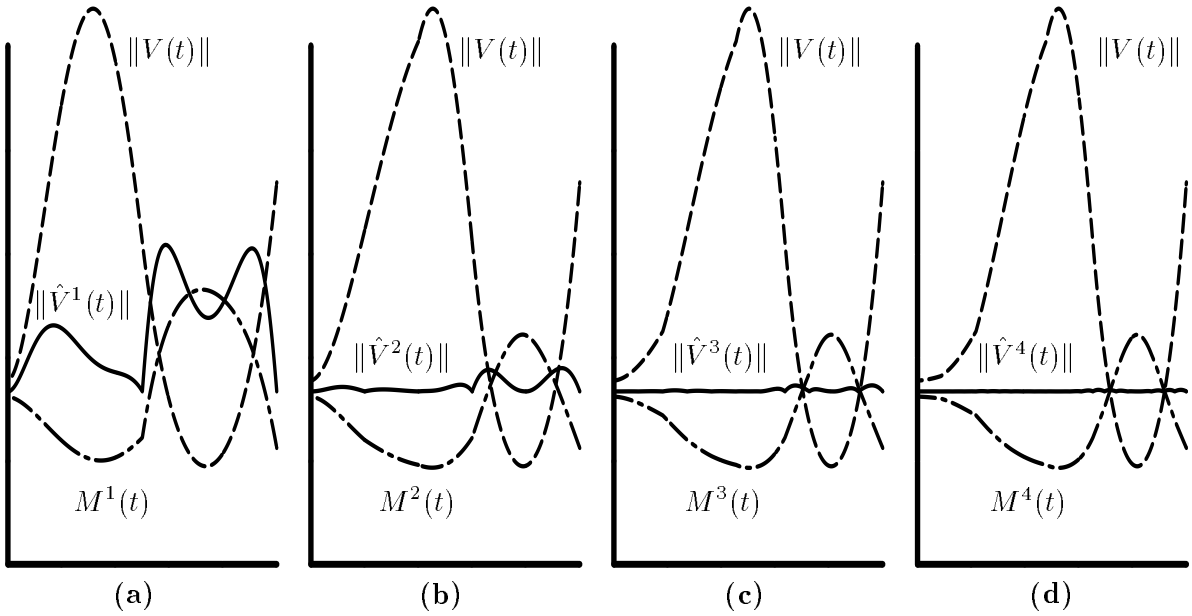


Figure 6: Few steps in a vector field curve normalization. Shown are: in dashed lines the magnitude of the unnormalized vector field, $\|V(t)\|$, in dotted-dashed lines the approximation to the inverse of the magnitude, $M(t)$, and in solid lines the magnitude of the approximation to the unit length vector field, $\|\hat{V}(t)\|$.

If $\|dC(s)/ds\| = 1$ we say that s is the curve's arc-length parameterization. It is evident that there always exists a change of parameter $t(s)$ so that any regular curve $C(t)$ can be written as $C(t(s)) = C(s)$ and s is the curve's arc-length parameterization [10]. By differentiating $C(t(s))$:

$$\frac{dC(t(s))}{ds} = \frac{dC(t)}{dt}\frac{dt(s)}{ds}, \tag{5}$$

or, by integrating along the arc-length,

$$t(s) = \int \frac{\left|\left|\frac{dC(t(s))}{ds}\right|\right|}{\left|\left|\frac{dC(t)}{dt}\right|\right|} ds = \int \frac{||C'(s)||}{||C'(t)||} ds = \int \frac{1}{||C'(t)||} ds, \tag{6}$$

because $||C'(s)|| = 1$. Because $C(t)$ is regular, $||C'(t)|| > 0$, The integration of a strictly positive function yields a strictly increasing function, $t(s)$.

Unfortunately, equation (6) is an integration with respect to the arc length parameter s which is unknown. Instead, the following approximation approach to arc length is used.

$$s(t) = \int \frac{\left|\left|\frac{dC(t)}{dt}\right|\right|}{\left|\left|\frac{dC(t(s))}{ds}\right|\right|} dt = \int \frac{||C'(t)||}{||C'(s)||} dt = \int ||C'(t)||\, dt. \tag{7}$$

Algorithm 2

Input:

curve $C(t)$, $t \in [T_{min}, T_{max}]$;
Tolerance τ;
Arc-length step δ;

Output:

$\mathcal{T}$, list of t parameter values of distance δ apart along $C(t)$;

Algorithm:

$m(t) \Leftarrow ||C'(t)||^2$, a scalar field computed symbolically;
$M(t) \Leftarrow$ approximation to $\sqrt{m(t)}$, a scalar field computed numerically;
$s(t) \Leftarrow \int M(t)dt$, a scalar field computed symbolically;
$\mathcal{T} \Leftarrow \emptyset$;
$\mathcal{D} \Leftarrow s(T_{max}) - s(T_{min})$;
$\Delta \Leftarrow s(T_{min}) + \delta$;
do
    $\mathcal{T} \Leftarrow \mathcal{T} \cup ZeroSet(s(t) - \Delta, \tau)$
    $\Delta \Leftarrow \Delta + \delta$;
while ($\Delta < \mathcal{D}$);

$||C'(t)||^2$ can be computed symbolically. $||C'(t)||$ can be numerically approximated from $||C'(t)||^2$ using the same technique that is described in Algorithm 1. Integration can also be computed symbolically for polynomial Bézier and Bspline curves (see [7]).

An algorithm to step along an arbitrary regular curve with constant arc-length steps can now be derived:

where $ZeroSet(c(t), \tau)$ uses numerical methods to find the zero set of $c(t)$ to within τ.

Figure 7 show some examples of the approach taken in Algorithm 2 to create constant arc-length steps along curves.

4 Enclosed Area of Closed Curves

It is common to attempt and design a complicated surface, such as a fuselage of a plane or a ship hull, by defining a set of cross sections that the surface interpolates or approximates. It is important to be able to compute the enclosed area of such a cross section, or even guarantee that all cross sections have certain areas, since these areas directly affect the volume that will be confined by the designed surface, and its hydro/aerodynamic behavior. If $C(t) = (x(t), y(t))$ is a polynomial planar curve, one can exploit Green's theorem and the symbolic tools defined herein to yield a symbolic closed form formulation of the

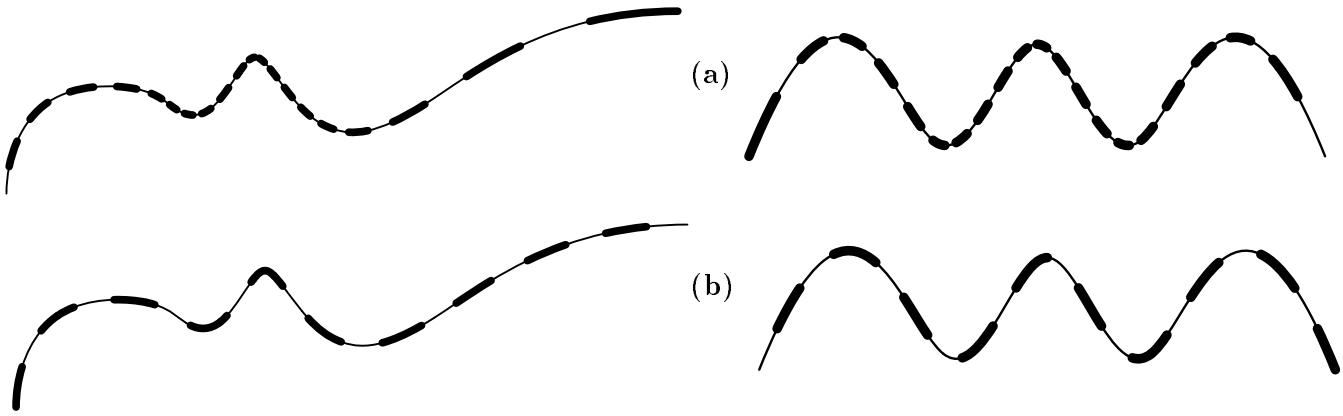


Figure 7: Two examples of an approximation of arc-lenth reparameterization of curves. In (a), constant steps along the original parameterization are shown while in (b) the arc-length steps computed using Algorithm 2 are displayed.

enclosed area, represented as a scalar field.

Given continuous functions $P(x,y)$, $Q(x,y)$, $\frac{\partial P}{\partial y}$, $\frac{\partial Q}{\partial x}$, the Green's theorem for a simple closed curve C,

$$\oint_C P dx + Q dy = \int\int_{\mathcal{S}} \left(\frac{\partial Q}{\partial x} - \frac{\partial P}{\partial y} \right) dx dy, \tag{8}$$

where $\mathcal{S}$ is the enclosed area by the closed curve $C(t)$.

Let $P(x,y) = -y$ and let $Q(x,y) = x$. Then the right hand side of equation (8) becomes $\int\int_{\mathcal{S}} 2 dx dy = 2\mathcal{S}$ or,

$$\mathcal{S} = \frac{1}{2} \oint_C x dy - y dx. \tag{9}$$

In parametric form equation (9) is equal to,

$$\mathcal{S} = \frac{1}{2} \oint_C \left(x(t) \frac{dy(t)}{dt} - y(t) \frac{dx(t)}{dt} \right) dt. \tag{10}$$

Equation (10) is computable and representable using the symbolic tools assumed in section 1. One can compute and represent the (area) integral of a closed polynomial curve as a polynomial [7], while the area itself is the definite integral evaluated at the two parameter values of the curve's end points.

5 Curve Curve Intersection and Coincidence

The need to find the intersection points of two curves rises in numerous instances, including modeling, Boolean operations, machining toolpath generation, and hidden curve removal of freeform surfaces [15]. Moreover, the detection of coincidence of regions of curves that trace the same geometry in a different speed is a difficult problem. By representing the distance function symbolically, one can alleviate the difficulties in solving these problems.

Let $C_1(u)$ and $C_2(v)$ be two regular curves. The distance between two points on the two curves is equal to,

$$d(u,v) = ||C_1(u) - C_2(v)|| \,, \tag{11}$$

but because of the square root in equation (11), we evaluate,

$$\begin{aligned} \delta(u,v) &= d^2(u,v) \\ &= (x_1(u) - x_2(v))^2 + (y_1(u) - y_2(v))^2 + (z_1(u) - z_2(v))^2 \,. \end{aligned} \tag{12}$$

Clearly $\delta(u,v)$ is computable and representable as a polynomial or rational scalar field because the x_i, y_i and z_i coefficients can be rewritten as $x_i(u,v)$, $y_i(u,v)$, and $z_i(u,v)$, respectively.

Figures 8 and 9 provide two examples of the $\delta(u,v)$ distance function.

By using bivariate minimization methods, solutions to the intersection points can be robustly found. Furthermore, not only one can find the curve's intersection points, but the closest points or the points with largest distance on the two curves can be isolated in the same way, by finding the local minima or maxima of $\delta(u,v)$. A search for local extrema of an explicit surface, as $\delta(u,v)$ is, can be mapped to a zero set search on a different explicit surface,

$$\zeta(u,v) = \left(\frac{\partial \delta(u,v)}{\partial u}\right)^2 + \left(\frac{\partial \delta(u,v)}{\partial v}\right)^2 . \tag{13}$$

In general $\delta(u,v) \geq 0$ and at an intersection point, $\delta(u,v) = 0$. If $\delta(u,v)$ is C^1 continuous, and using the convex hull property of the Bézier and NURBs representations, it is clear that at the neighborhood of an intersection point, there exists a non positive coefficient in the control mesh of $\delta(u,v)$.

In Figures 8 and 9, a simple subdivision based algorithm is used and which subdivides a surface if one of its coefficients is non positive, until a termination criteria is reached, usually that the size of the parametric domain is sufficiently small. This simple algorithm was found reasonably fast and very robust.

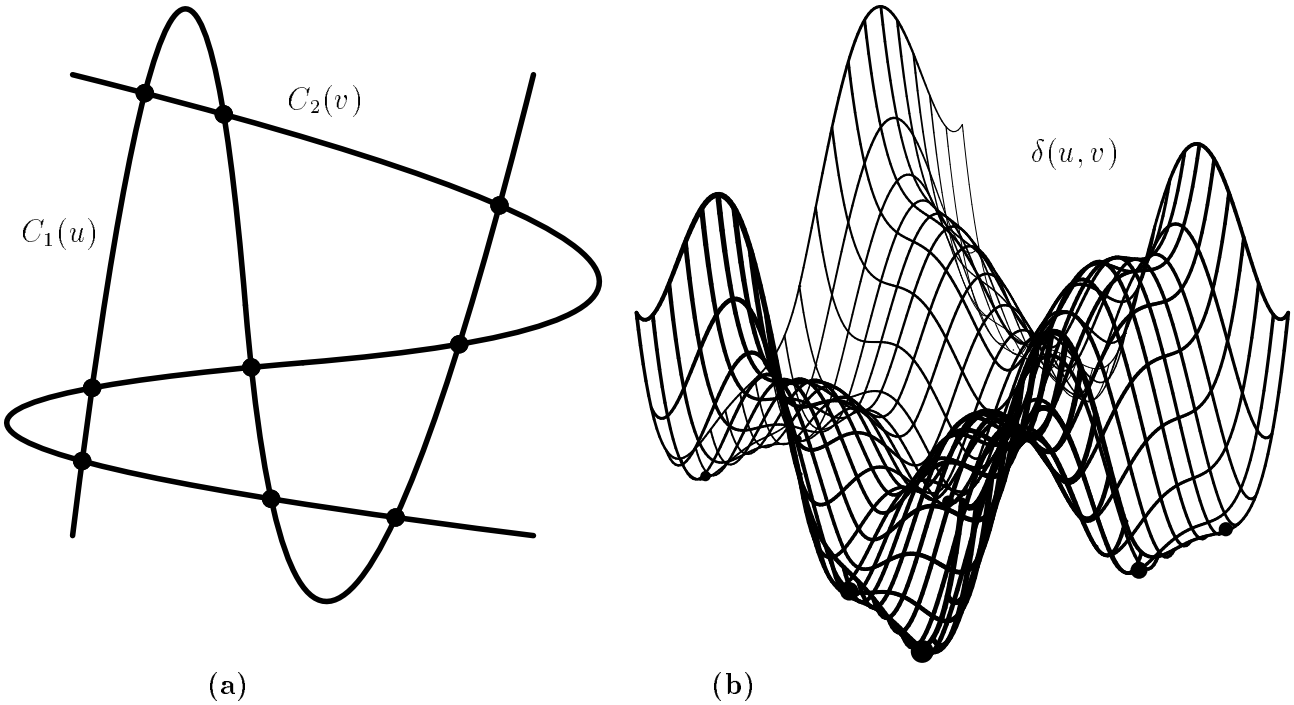


Figure 8: A bivariate distance square function, $\delta(u,v) = \|C_1(u) - C_2(v)\|$, is defined (b) for the given two curves $C_1(u)$ and $C_2(v)$ (a) and is used to robustly find their nine intersection points.

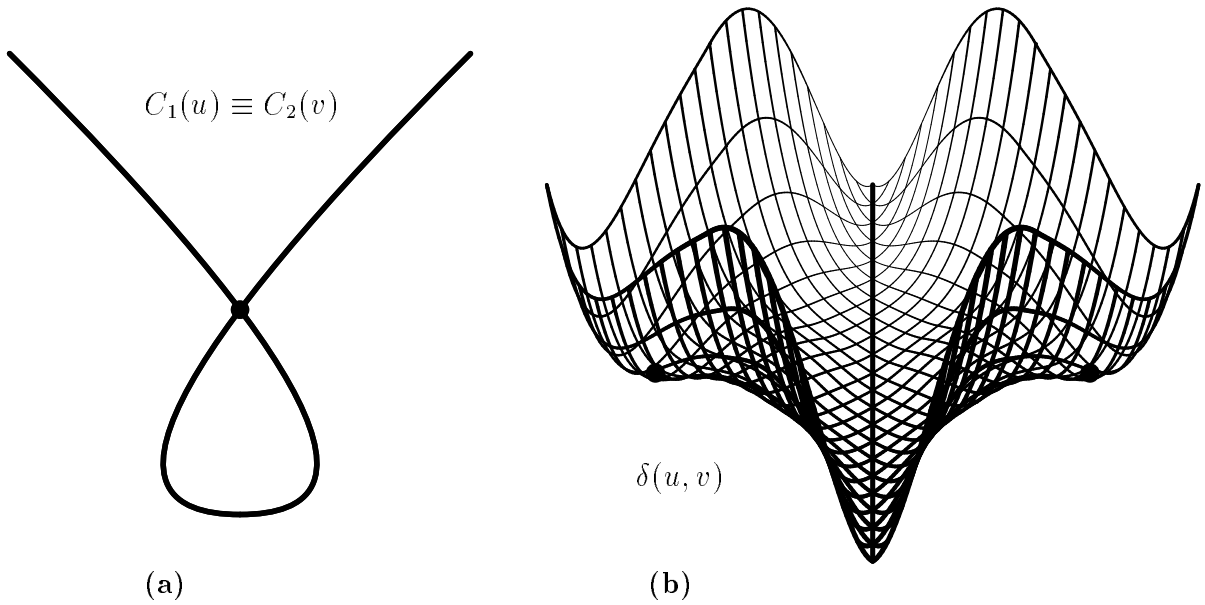


Figure 9: A bivariate distance square function, $\delta(u,v) = \|C_1(u) - C_1(v)\|$, is defined (b) from a curve (a) to itself. As expected the diagonal $u = v$ of the distance function is zero. However, since the curve is self intersecting, two more symmetric zeros corresponding to the self intersection point, are found.

By elevating the curve curve intersection problem a dimension, the problem is transformed into a zero set finding problem. One can use $\delta(u, v)$ to not only robustly find the (self) intersection points and shortest distance points, but also to find regions of coincidence manifested as local minima curves on $\delta(u, v)$ that are everywhere zero as well. Further, the isolation of the regions in $C_1(u)$ such that $\|C_1(u) - C_2(v)\| < \xi$, for some constant ξ, is reduced to the computation of the zero set of $\delta(u, v) - \xi^2$.

6 conclusion

We have shown that few symbolic tools can enhance the robustness of current analysis methods of freeform curves as well as introduce new analysis techniques altogether. It is clear that the examples that were demonstrated in this paper are only a small subset of problems that can benefit from this symbolic approach and further research in the area is to be carried out.

References

[1] **B. Cobb.** Design of Sculptured Surfaces Using The B-spline Representation. *Ph.D. Thesis*, University of Utah, Computer Science Department, June 1984.

[2] **G. Celniker.** Shape Wright: Finite Element Based Free-Form Shape Design. *Ph.D. Thesis*, Massachusetts Institute of Technology, Mechanical Engineering, September 1990.

[3] **G. Celniker and W. Welch.** Linear Constraints for Deformable B-Spline Surfaces. *Computer Graphics (Symposium on Interactive 3D graphics)*, Vol. 25, No. 2, pp 165-170, March 1992.

[4] **Y. Hel-Or, A. Rappoport, and M. Werman.** Relaxed Parametric Design with Probabilistic Constraints. *2nd ACM Solid Modeling conference,* pp 261-270, May 93, Montreal Canada.

[5] **M. Kallay.** Constrainted optimization in Surface Design. *Modeling in Computer Graphics,* B. Falcidieno and T. L. Kunii (Eds.), Working Conference on Geometric Modeling in Computer Graphics (IFIP TC 5/WG 5.10), Genova 1993.

[6] **W. Welch and A. Witkin.** Variational Surface Modeling. *Computer Graphics (Siggraph 1992)*, Vol. 26, No. 2, pp 157-166, July 1992.

[7] **G. Farin.** Curves and Surfaces for Computer Aided Geometric Design Academic Press, Inc. Third Edition 1993.

[8] **J. A. Roulier.** Bezier curves of Positive Curvature. *Computer Aided Geometric Design*, Vol. 5, pp 59-70, 1988.

[9] **N. S. Sapidis.** Controlling the Curvature of a Quadratic Bezier curve. *Computer Aided Geometric Design*, Vol. 9, pp 85-91, 1992.

[10] **M. P. DoCarmo.** Differential Geometry of Curves and Surfaces. *Prentice-Hall* 1976.

[11] **G. Elber and E. Cohen.** Adaptive Isocurves Based Rendering for Freeform Surfaces. *Technical Report UUCS-92-040*, University of Utah, 1992.

[12] **M. Markot and R. L. Magenson.** Solutions of Tangential Surface and Curve Intersections. *Computer Aided Design*, Vol. 21, No. 7, pp 421-429, September 1989.

[13] **T. W. Sederberg and S. Parry.** Comparison of Three Curve Intersection Algorithms. *Computer Aided Design*, Vol. 18, No. 1, pp 58-63, January/February 1986.

[14] **T. W. Sederberg and T. Nishita.** Curve Intersection Using Bezier Clipping. *Computer Aided Design*, Vol. 22, No. 9, pp 538-549, November 1990.

[15] **G. Elber and E. Cohen.** Hidden Curve Removal for Free Form Surfaces. *Computer Graphics (Siggraph 1990)*, Vol. 24, No. 4, pp 95-104, August 1990.

[16] **G. Elber.** Free Form Surface Analysis using a Hybrid of Symbolic and Numeric Computation. *Ph.D. thesis*, University of Utah, Computer Science Department, December 1992.

[17] **R. T. Farouki and V. T. Rajan.** Algorithms For Polynomials In Bernstein Form. *Computer Aided Geometric Design*, Vol. 5, pp 1-26, 1988.

[18] **K. Kim.** Blending Parametric Surfaces. *M.S. Thesis*, University of Utah, Computer Science Department, August 1992.

[19] **K. Morken.** Some Identities for Products and Degree Raising of Splines. To appear in the *Journal of Constructive Approximation.*

[20] **D. S. Kim.** Hodograph Approach to Geometric Characterization of Parametric Cubic Curves. *Computer Aided Design*, Vol. 25, No. 10, pp 644-654, October 1993.

[21] **M. C. Stone and T. D. DeRose.** A Geometric Characterization of Parametric Cubic Curves. *ACM Transactions on Graphics*, Vol. 8, No. 3, pp 147-163, 1989.

[22] **E. Kantorowitz. and Y. Schechner.** Managing the Shape of Planar Splines by Their Control Polygons. *Computer Aided Design*, Vol. 25, No. 6, pp 355-364, June 1993.

[23] **G. Elber and E. Cohen.** Error Bounded Variable Distance Offset Operator for Free Form Curves and Surfaces. *International Journal of Computational Geometry and Applications,* Vol. 1., No. 1, pp 67-78, March 1991.

[24] **G. Elber and E. Cohen.** Error Bounded Variable Distance Offset Operator for Free Form Curves and Surfaces. *Technical Report UUCS-91-001*, University of Utah, 1992.

[25] **R. T. Farouki and C. A. Neff.** Some Analytic and Algebraic Properties of Plane Offset Curves. *Research Report 14364 (#64329) 1/25/89*, IBM Research Division.

[26] **D. J. Fillip.** Blending Parametric Surfaces. *ACM Transaction on Graphics,* Vol. 8, No. 3, pp 165-173, July 1989.

[27] **P. Koparkar.** Parametric Blending using Fanout Surfaces. *Symposium on Solid Modeling Foundation and CAD/CAM Applications,* June 1991, Austin Texas, pp. 317-327.

[28] **M. Marsden and I. J. Schoenberg.** On Variation Diminishing Spline Approximation Methods. *Mathematica*, Vol. 8(31), No. 1, pp 61-82, 1966.