Solving Geometric Problems using Subdivision Methods and Range Analysis

J. Estrada, D. Martínez, D. León, and H. Theisel

Abstract. In this paper we show that some important geometric modelling and geometric optimization problems related to implicitly defined algebraic plane curves, such as locating the singular points, the points of characteristic curvature, as well as computing the Euclidean distance from a point to a curve and computing the maximal/minimal distance between two curves, may be efficiently solved using subdivision schemes and range analysis. The proposed algorithms are efficient and robust using a finite arithmetic, even in the case that the curves considered have singular points of high multiplicity and/or very close branches.

§1. Introduction

Implicit curves are very useful in trimming operations on parametrically described shapes such as the representation of the intersection of two parametric surfaces or the representation of the silhouette edges of a parametric surface with respect to a given view (see [23]). There are also several CAD, CAGD and computer graphics problems where tracing an implicitly defined algebraic curve on a plane region is of great importance (see [1], [13], [14]).

During the nineties, several authors (see [4], [6], [23], [24]) introduced range analysis criteria to test if an implicitly defined algebraic curve does not pass through a rectangular plane region, and applied it to render implicitly defined algebraic curves.

In this paper we present efficient solutions to several interesting problems using these tests. First we consider the problem of locating the intersection points of implicitly defined algebraic curves. Later we show how this result may be used to solve other important geometric problems such as locating singular points and points of characteristic curvature, as well as

Mathematical Methods for Curves and Surfaces: Tromsø 2004
M. Dæhlen, K. Mørken, and L. L. Schumaker (eds.), pp. 101–114.

ISBN 0-0-9728482-4-X

computing the Euclidean distance from a point to an algebraic curve and computing the maximal/minimal distance between two algebraic curves.

The algorithms presented use a finite arithmetic and are simple, efficient and robust even in the case that the curves considered have singular points of high multiplicity and/or very close branches.

§2. Preliminaries

A plane algebraic curve is the set $Z(f)$ of points satisfying the polynomial equation

$$f(x,y) = \sum_{i=0}^{d} \sum_{j=0}^{d-i} f_{ij} x^i y^j = 0, \tag{1}$$

where d is the degree and f_{ij} the coefficients.

Rendering an implicit algebraic curve in a grid of pixels is of great interest in CAD, CAGD and computer graphics. A general subdivision procedure that provides a method for drawing an implicitly defined algebraic curve on a grid of pixels (or even higher resolution) may be described as follows.

Given any test to check if a curve does not pass through a square region (see [18] and the references therein), apply this test to the plot area as an initial cell, if it is not rejected, then subdivide the cell into smaller squares and apply recursively the test, until the lengths of the sides of the squares become smaller than any prescribed magnitude $\varepsilon > 0$ (for instance the length of a pixel side, in the computer graphics case).

Range analysis provides a general test for rejecting cells at each subdivision step: interval arithmetic, affine arithmetic, Berstein coefficient method, Taubin's method, Rivlin's method are some of the most common sources for range analysis. A comparison of the efficiency and performance of different function range interval methods for plotting implicitly defined algebraic curves is presented in [18].

§3. Intersection Points of Algebraic Curves

The problem of computing intersection points of algebraic curves has been previously considered in the more general setting of finding the solutions of a system of n polynomial equations (2).

$$\begin{cases} f_1(x,y) = 0 \\ f_2(x,y) = 0 \\ \vdots \\ f_n(x,y) = 0. \end{cases} \tag{2}$$

One approach, for instance, uses multivariate Stürm sequences (see [19], [21]). Another approach is based on U-Resultant schemes (see [15], [16]), but they are very expensive to compute for a set of equations of arbitrary high degree.

In this section we describe a procedure for computing the intersection points of n algebraic curves in a square region R, using subdivision schemes and range analysis.

The algorithm begins by applying a recursive subdivision method based on range analysis (see Section 2) simultaneously to the curves $\{f_1, f_2, \ldots, f_n\}$ in the square region R. In each step, only those square regions satisfying the test for all curves are considered for the next subdivision step. The output is a set of small square regions with side length smaller than a previously prescribed accuracy $\varepsilon > 0$ (pixels, in the computer graphic case). These small squares contain all the intersection points and their centers provide us approximations to the exact intersection points of the curves, since with an error no greater than $\frac{\sqrt{2}\varepsilon}{2}$, these centers are on all the curves $\{f_1, f_2, \ldots, f_n\}$. Hence, an approximate solution to the problem of computing the intersection points in a square R of a set of n algebraic curves may be the following: given a prescribed accuracy $\varepsilon > 0$ (ε equal to the length of a pixel side, in the computer graphics case), apply a recursive subdivision method based on range analysis simultaneously to the curves $\{f_1, f_2, \ldots, f_n\}$ in the square region R, obtaining a set of small square regions with side length no greater than ε containing all their intersection points in R, and propose the centers of these small squares as approximations to the exact intersection points. In the generic case, the number of subdivisions k required to obtain approximations to the exact intersection points with an error smaller than ε is

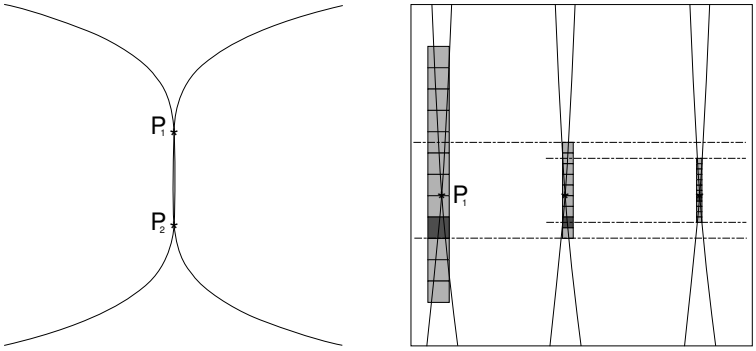
$$k = [log_2(l) + 1 - \log_2(\varepsilon)]$$

if the length of the side of the square R is equal to l.

Unfortunately, applying this simple method does not always lead to good approximations to the exact solution. When the curves do not intersect transversely, or if they have very close branches, more that one small square is found representing each intersection point hence, the solution must be refined (see Figure 1).

Clusters of small squares may appear around intersection points, and since the intersections points of a generic set of algebraic curves are isolated, we propose a simple strategy to select good approximations to the exact intersection points contained in each cluster.

Considering the centers of the small squares obtained as result of the subdivision process, we construct a set of clusters $\{CC_i\}$ as follows: One center of the small squares will belong to one cluster CC_j if its distance to at least one of the other points in CC_j is less or equal to 2ε, where ε is the previously prescribed accuracy.



(a) (b)

Fig. 1. Curve intersection (a), zooms in the neighborhood of P_1 with $k = 8$, 9 and 10 (b). As k increases, the size of the clusters diminishes.

Then, the sum of the Euclidean distances Sd_{ik} from the centers of the small squares p_{ik} contained in each cluster CC_i to each of the involved curves $\{f_1, \ldots, f_n\}$ is computed. Finally, the center of the small squares $p_{i\hat{k}}$ which minimizes Sd_{ik} is selected as the best approximation to the solution. Assuming that the set of curves $\{f_1, f_2, \ldots, f_n\}$ does not have a common component, after Bezout's Theorem for $\varepsilon \to 0$ the obtained approximations to the intersection points converge to the exact solutions.

It is clear that if ε is sufficiently small, then even using approximations to the Euclidean distance to compute Sd_{ik}, the points $p_{i\hat{k}}$ that minimize Sd_{ik} are good approximations to the exact solution.

There exist in the literature some approximations to the Euclidean distance (see Section 5). As the points p_{ik} are very close to the curves, we may use the one proposed by G. Taubin [24] given by the expression

$$d = \frac{|f(x,y)|}{||\nabla f(x,y)||}.$$

Our numerical experiments show that this is a good choice in most cases.

Of course, the more precision is required, the greater number of subdivisions are needed and the amount of points in each cluster grows. In all numerical examples we found that, within the usual precision required for CAD and CAGD problems, the computational cost is reasonable (see Sect. 6). Alternatively, we may use the approximations obtained in this way as a good initial approximations for any robust method to solve the nonlinear system (2).

§4. Locating Characteristic Points

Each curve has a set of characteristic points where it has a distinguished geometric behavior. Among these points are the singular points, the points with curvature 0 and the points attaining extreme values of the curvature. In this section we show how the problem of finding these points may be presented as a curve intersection problem, and thereby solved using the algorithm described in Section 3.

4.1. Locating Singular Points

A point p on an algebraic curve $f(x, y) = 0$ is called singular if both partial derivatives $f_x(x, y)$ and $f_y(x, y)$ vanish at p. To find the singular points of an algebraic curve $f(x, y) = 0$ we must be able solve efficiently the nonlinear system of equations

$$\begin{cases} f(x,y) = 0 \\ f_x(x,y) = 0 \\ f_y(x,y) = 0. \end{cases} \tag{3}$$

We propose to find the solutions of system (3) locating the intersection points among the algebraic curves $\{f = 0, f_x = 0, f_y = 0\}$ (see Section 3). Observe that in the case of singular points with high multiplicity, these three curves may not intersect transversely, nevertheless, the proposed method shows a good performance (see Figure 2).

4.2. Locating Points of Characteristic Curvature

A regular point p on an algebraic curve $f(x, y) = 0$ is considered as a point with characteristic curvature if the curvature $k(x, y)$ given by the expression

$$k(x,y) = \frac{f_{xx}f_y^2 - 2f_xf_yf_{xy} + f_x^2f_{yy}}{(f_x^2 + f_y^2)^{3/2}}$$

is 0 or attains a relative extreme value in this point.

The points on a curve $f(x, y) = 0$ where the curvature $k(x, y)$ is 0 may be computed as the solutions of the nonlinear system of equations

$$\begin{cases} f(x,y) = 0, \\ k_n(x,y) = 0, \end{cases} \tag{4}$$

where $k_n(x, y)$ is the numerator of $k(x, y)$.

On the other hand, the points attaining extreme values of the curvature are the solutions of the nonlinear system

$$\begin{cases} f(x,y) = 0, \\ dk_n(x,y) = 0, \end{cases} \tag{5}$$

where $dk_n(x,y)$ is the numerator of the differential of the curvature $k(x,y)$ given by

$$dk(x,y) = f_x k_y - f_y k_x,$$

where f_x, f_y denote the partial derivatives of $f(x,y)$ and k_x, k_y the partial derivatives of the curvature $k(x,y)$.

The curves $k_n(x,y) = 0$ and $dk_n(x,y) = 0$ are also algebraic curves of degree $O(3d)$ and $O(6d)$, where d is the degree of f. Though, to compute the points with curvature 0 and the points attaining extreme values of the curvature in a region R is equivalent to locate the intersection points between the algebraic curves $\{f = 0, k_n = 0\}$ and $\{f = 0, dk_n = 0\}$ respectively, in that region (see Section 3). In both cases, as a result we will obtain all the points on $f = 0$ contained in R with characteristic curvature, as well as the singular points (see Figure 2). Observe that some of the curves involved do not intersect transversely or possess very close branches. Nevertheless, the proposed method performs well (see Figure 2).

§5. Euclidean Distance

There are several practical problems arising from computer graphics, computer vision, pattern recognition and computational mechanics, where it is necessary to compute the Euclidean distance from a point to an arbitrary algebraic plane curve, as well as the coordinates of the footpoints.

Another interesting problem is to compute the maximal/minimal distance between two algebraic curves and the coordinates of the points where it is attained.

5.1. Computing the Euclidean Distance from a Point to an Algebraic Curve

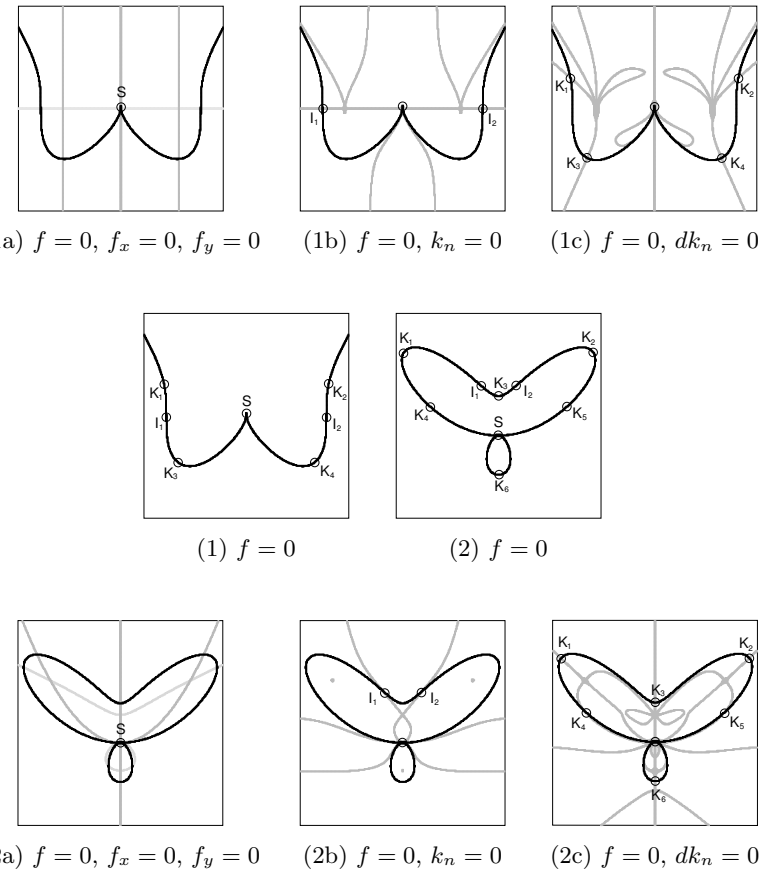
The problem of computing the Euclidean distance from a point q to a curve $\mathcal{C}$ may be stated as the following constrained nonlinear global minimization problem:

$$d(q,\mathcal{C}) = min\{\|p - q\| : f(p) = 0\} \tag{6}$$

More geometrically, the Euclidean distance from q to $\mathcal{C}$ is attained at a point p on $\mathcal{C}$, such that the normal of $\mathcal{C}$ at p passes through q (see Figure 3).

This problem is, in general, very difficult to solve. The already known works on this subject can be classified into two groups, those which offer a closed formula to compute approximations to the Euclidean distance ([24]) and those which provide an algorithm to compute the footpoints ([8],[9],[10],[12]).

Some of this methods have the disadvantage that they do not give good results if the external point is not close to the curve, or if it has more than



(1a) $f = 0$, $f_x = 0$, $f_y = 0$ (1b) $f = 0$, $k_n = 0$ (1c) $f = 0$, $dk_n = 0$

(1) $f = 0$ (2) $f = 0$

(2a) $f = 0$, $f_x = 0$, $f_y = 0$ (2b) $f = 0$, $k_n = 0$ (2c) $f = 0$, $dk_n = 0$

S - Singular points of $f = 0$.
I - Points on $f = 0$ where the curvature is 0.
K - Points on $f = 0$ attaining extreme values of the curvature.

Fig. 2. Locating Characteristic Points.

one footpoint. Also some of them just compute an approximation of the distance without giving us the coordinates of the footpoints, which may be also useful in many situations.

Now we describe a simple procedure to compute the footpoints that applies to any algebraic curve, and does not requires initial conditions related to the location of the external point.

If we have the curve represented by its implicit equation $f(x, y) = 0$, then the coordinates (x, y) of the points on f attaining the relative extremes values of the distance from an external point $q(x_0, y_0)$ to the

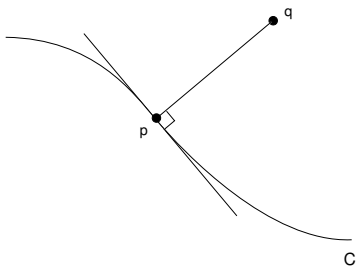


Fig. 3. The orthogonal projection of q.

curve as well as the singular points, may be computed as the solution of the following nonlinear system of polynomial equations

$$\begin{cases} f(x,y) = 0 \\ f_N(x,y) = f_y(x,y)(x-x_0) - f_x(x,y)(y-y_0) = 0. \end{cases} \tag{7}$$

Then it is easy to select among these points those that give us the global minimum (6) of the distance from each point to the external point q.

In order to compute the solutions of system (7) we propose to locate the intersection points between the algebraic curves f and f_N (see Section 3) in a square region $R(q, 2D)$ centered in the external point q with side length twice any upper bound of the Euclidean distance from q to f (see Figure 4). This upper bound D may be easily obtained, for example, computing the distance from q to any point on f.

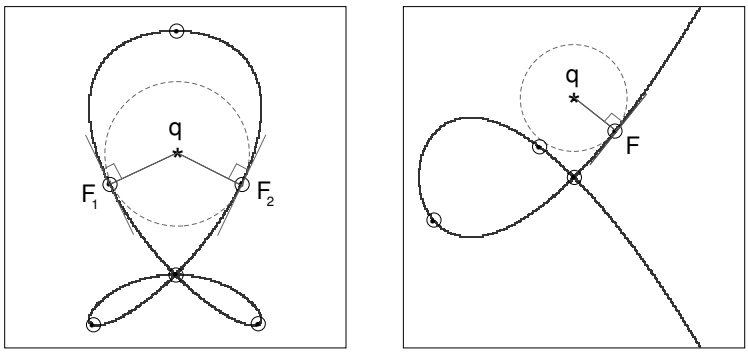
5.2. Computing the Extremal Distance Between Two Curves.

In this section an algorithm for detecting pairs of points attaining the maximal/minimal distance between two algebraic curves non intersecting each other in a square region R is described.

Given two algebraic curves $f(x,y) = 0$, $g(x,y) = 0$ and a square region R, detect the pairs of points $P_f, P_g \in R$, where P_f is a point on $f(x,y) = 0$ and P_g is a point on $g(x,y) = 0$, such that the normal line of $f(x,y) = 0$ at P_f coincides with the normal line of $g(x,y) = 0$ at P_g.

Algorithm. We subdivide R in 4 square cells $R_k,\ k = 1,..,4$. Given two square cells $R_i = I_x^i \times I_y^i$ and $R_j = I_x^j \times I_y^j$:

1. Test if $f(x,y) = 0$ crosses R_i and if $g(x,y) = 0$ crosses R_j
2. Compute the range intervals $I_{f_x}^i$ (respectively $I_{f_y}^i$) for the function $f_x(x,y)$ (resp. $f_y(x,y)$) on R_i.
3. Compute the range intervals $I_{g_x}^j$ (respectively $I_{g_y}^j$) for the function $g_x(x,y) = 0$ (resp. $g_y(x,y) = 0$) on R_j.



(1) (2)

F - Footpoints of the point q on $f = 0$.
O - Orthogonal projections of the point q on $f = 0$
and singular points.

Fig. 4. Locating Footpoints.

4. Using interval arithmetic, test if 0 is contained in all the following intervals
   (a) $I^i_{f_x} * (I^i_y - I^j_y) - I^i_{f_y} * (I^i_x - I^j_x)$
   (b) $I^j_{g_x} * (I^j_y - I^i_y) - I^j_{g_y} * (I^j_x - I^i_x)$
   (c) $I^j_{f_y} * I^i_{g_x} - I^j_{f_x} * I^i_{g_y}$.

Remark. Note that the first two conditions (a) and (b) correspond to the requirement that the interval of normal lines to $f(x, y) = 0$ at points in R_i intersects R_j and the same permuting f with g and i with j. The third condition corresponds with the requirement that in both intervals of normal lines exist pairs of parallel lines. Observe that the first two conditions may hold if R_i and R_j are very small and also very close, while (c) may not necessarily hold.

If all answers from 1 to 4 were "yes" then subdivide both R_i and R_j into four square cells and apply the same tests to all possible pairs of square cells (16 cases), until the size of the sides of a square cells is not greater than a previously prescribed accuracy $\varepsilon > 0$.

The output are the pairs of the centers of the small squares associated to each curve, whose corresponding intervals of normal lines approximately coincide. Since this output is a dense sample and many pairs of centers of the small squares are very near to each other, we apply the following method for selecting representative pairs of points from the clusters of pairs of centers of small squares. First, we consider that two pairs of

centers of the small squares (P_f^i, P_g^i) and (P_f^j, P_g^j) are near each other if $||P_f^i - P_f^j|| \leq 2 * \varepsilon$ and $||P_g^i - P_g^j|| \leq 2 * \varepsilon$. Second, we construct a graph where there is an edge between two pairs of centers of the small squares (P_f^i, P_g^i) and (P_f^j, P_g^j) if they are near each other. Then, the connected components of the graph are computed and a representative pair for each connected component is selected as follows.

Given a cluster of pairs of centers of small squares $C = \{(P_f^i, P_g^i)\}$ we compute:

1. r_i as the line that joins P_f^i and P_g^i
2. n_f^i the normal line of $f = 0$ at P_f^i and the angle θ_f^i between r_i and n_f^i
3. n_g^i the normal line of $g = 0$ at P_g^i and the angle θ_g^i between r_i and n_g^i

The pair of points minimizing the the deviation among the normal line of each curve at each point and the line joining the pair of points $(\cos\theta_f^i)^2 + (\cos\theta_g^i)^2$ is selected as the representative of the cluster.

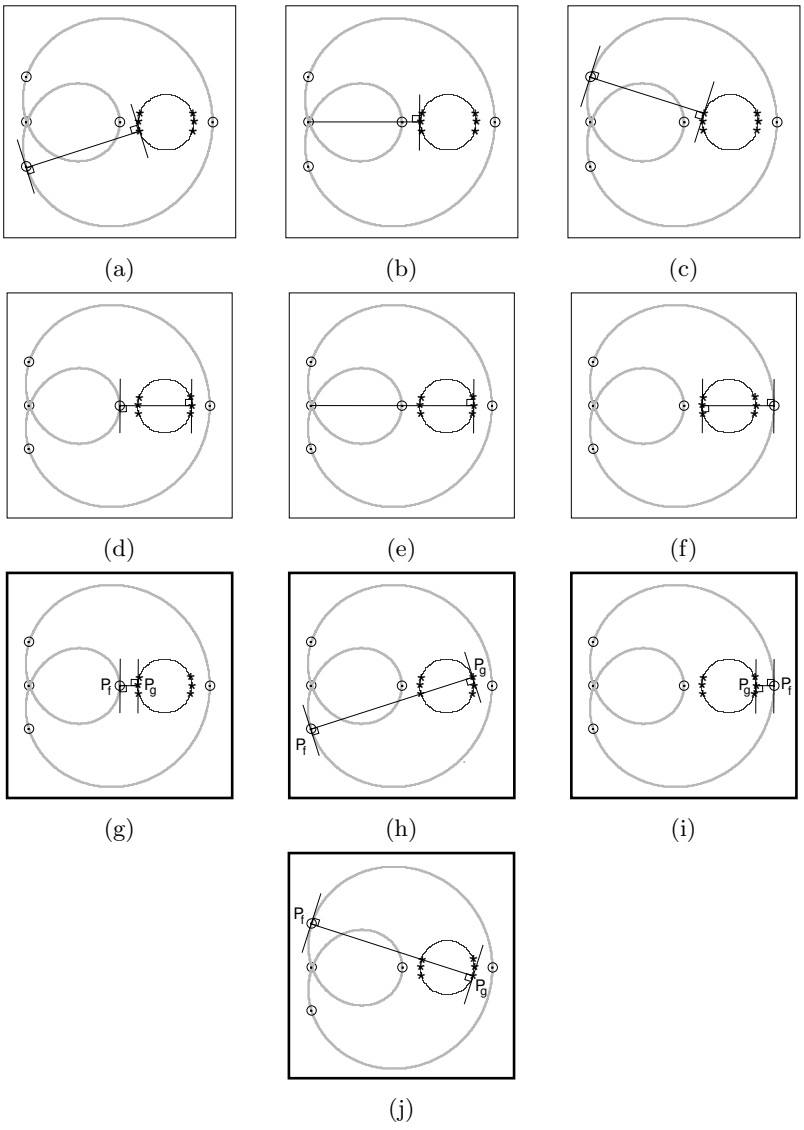
As a result of this algorithm, we obtain good approximations for the pairs of points attaining the relative extreme values of the distance between the algebraic curves f and g in a region R. Finally, we must select among them those that minimize/maximize the distance. It is guaranteed that if both curves are completely contained inside the region R then the global maximum and minimum are obtained (see Figure 5).

A faster alternative procedure (suggested by the anonymous referee) could be simply to compute all distances in the cluster of pairs of points, and then select the smallest/largest. For very small values of the accuracy ε, both algorithms provide very similar results, but for intermediate ε, the pairs of centers of small squares attaining a minimum/maximum distance may correspond to points which are relatively *far away* from the corresponding curves (for instance, to a distance approximately equal to $\frac{\sqrt{2}\varepsilon}{2}$) and in this situation, selecting the pair of points minimizing the deviation among the normal line of each curve at each point and the line joining the pair of points gives better results.

§6. Additional Figure Information

In this section we provide some additional information on the figures. All timings reported in the present paper are in CPU seconds on a 700 MHz Pentium III processor running Windows.

Figure 1. This figure shows the intersection points between the curves $x - y^4 = 0$ and $x + y^4 - 0.01 = 0$ inside the square centered in the origin with side length 2.0, for $k = 8$, 9 and 10 subdivision steps. The exact intersection point $P1$ has coordinates $(0.005, -0.2659147948)$, For $k = 8$, the approximate solution $P1 = (0.00390625, -0.277344)$ was computed in 1 ms running time and the cluster diameter is approximately $2 \cdot 10^{-2}$. For $k = 9$, the approximate solution $P1 = (0.00585938, -0.275391)$ was computed in



(P_f, P_g) - Pairs of points on the curves $f = 0$ and $g = 0$ where the global minimum value (Figs. (g), (i)) and the global maximum value (Figs. (h), (j)) of the distance between both curves is attained.

Fig. 5. Pairs of points attaining relative extreme values of the distance between two algebraic curves.

15 ms running time and the cluster diameter is approximately $7 \cdot 10^{-3}$. For $k = 10$, the approximate solution $P1 = (0.00488281, -0.266602)$ was computed in 16 ms running time and the cluster diameter is approximately $2 \cdot 10^{-3}$.

Figure 2. This figure shows two examples of computation of characteristic points.

Example (1): For the curve $f(x, y) = y^3 - x^4 + x^2 = 0$ we computed the singular point $S = (0.005, -0.005)$, the zero curvature points $I_1 = (-0.526, 1.339)$ and $I_2 = (0.526, 1.339)$, and the extreme of curvature points $K_1 = (-1.024, 0.375)$, $K_2 = (1.024, 0.375)$, $K_3 = (-0.819, -0.604)$ and $K_4 = (0.819, -0.604)$, in the square region of center $(0.0, 0.0)$, and side 2.5. Run times were 1ms for singular points, 40ms for inflexion points and 50ms for computing the points attaining extreme values of the curvature.

Example (2): For the curve Curve $f(x, y) = x^4 + y^4 - 4x^2y - y^2 = 0$ we computed the singular point $S = (0.010, 0.007)$, the zero curvature points $I_1 = (-0.100, -0.001)$ and $I_2 = (0.100, 0.001)$, and the extreme of curvature points $K_1 = (-2.379, 2.138)$, $K_2 = (2.379, 2.138)$, $K_3 = (-0.003, 1.000)$, $K_4 = (-1.658, 0.665)$ and $K_5 = (1.658, 0.665)$, for the square region of center $(0.0, 0.5)$, and side 5.2. Run times were 1ms for singular points, 100ms for inflexion points and 140ms for computing the points attaining extreme values of the curvature.

Figure 4. This figure shows some results of the Euclidean distance computation.

Example (1): The distance from the point $q = (0.0, 0.5)$ to the curve $f(x, y) = y^3 - x^2y - x^4 - y^4 = 0$ is attained at the footpoints $F_1 = (-0.267, 0.367)$, and $F_2 = (0.267, 0.367)$. Computations were performed for the square region of center $(0.0, 0.4)$ and side 1.5.

Example (2): The distance from the point $q = (0.0, 0.5)$ to the curve $f(x, y) = y^2 - x^3 - x^2 = 0$ is attained at footpoint $F = (0.259, 0.291)$. Computations were performed for the square region of center $(0.0, 0.0)$ and side 2.2.

Figure 5. Pairs of points attaining relative extreme values of the distance between two algebraic curves. For the curves $f(x, y) = 4x^4 + 8x^2y^2 - 12x^3 + 4y^4 - 12xy^2 + 8x^2 - y^2 = 0$ and $g(x, y) = 4x^2 - 12x + 8.64 + 4y^2 = 0$ were computed the Global Maxima (h)$\{$ $P_f = (-0.011, -0.483)$; $P_g = (1.786, 0.093)$ $\}$ and (j) $\{P_f = (-0.011, 0.483)$; $P_g = (1.786, -0.093)\}$. Also the Global Minima (g) $\{P_f = (1.005, -0.005)$; $P_g = (1.20, -0.005)\}$ and (i) $\{P_f = (2.001, -0.005)$; $P_g = (1.796, -0.005)\}$.

Acknowledgments. We would like to express our gratitude to Richard Patterson and Luiz Henrique de Figueiredo for their valuable suggestions, comments and careful reading. We wish also thank R.-P. Holzapfel for providing us with the nice examples of his gallery of singular algebraic curves.

References

1. S. Behar, V. Hernández and J. Estrada, Geometric design by means of a G2 continuous A-spline, Approximation and Optimization in the Caribbean, M. Lassonde (ed.), Physica-Verlag, Heidelberg, 2001, 133–145.
2. M. D. Bloomenthal, *Error bounded approximate reparametrization of NURBS curves*, M. Sc. Thesis, University of Utah, 1999.
3. G. Casciola and S. Morigi, Reparametrization of NURBS curves, Int. Journal of Shape Modeling **2** (1996), 103–116.
4. L. H. de Figueiredo, Surface intersection using affine arithmetic, Proceedings of Graphics Interface (1996), 168–175.
5. L. H. de Figueiredo, *Computational Morphology of Implicit Curves*, PhD Thesis, IMPA, Rio de Janeiro, 1992.
6. L. H. de Figueiredo and J. Stolfi, Adaptive enumeration of implicit surfaces with affine arithmetic, Computer Graphics Forum **15**(5) (1996), 287–296.
7. J. Gil and D. Keren, New approach to the arc length parametrization problem, 13th Spring Conference on Computer Graphics, W. Strasser (ed.), 1997, 27–42.
8. E. Hartman, The Normal Form of a Planar Curve and Its Applications to Curve Design, *Mathematical Methods for Curves and Surfaces II*, Dæhlen, Lyche and Schumaker (eds.), Nashville, 1998, 1–8.
9. E. Hartman, On the Curvature of Curves and Surfaces Defined by Normal forms, Comput. Aided Geom. Design **16** (1989), 355–376.
10. E. Hartman, Numerical Parameterization of curves and surfaces, Comput. Aided Geom. Design **17**, 2000, 251–266.
11. V. Hernández and J. Estrada, Generation of points on a curve with prescribed distribution of their arclength, Revista Investigación Operacional **17** (1996), 133–138.
12. V. Hernández, J. Estrada, P. Barrera, A new algorithm to compute the Euclidean distance from a point to a conic, Revista Investigación Operacional **23**(2) (2002), 164–174.
13. V. Guerra and V. Hernández, Numerical aspects in locating the corner of the L-curve, Approximation and Optimization in the Caribbean, M. Lassonde (ed.), Physica-Verlag, Heidelberg, 2001, 121–131.
14. V. Hernández and J. Estrada, Designing a G1 A-spline surface over a triangulation, Trends in Approximation Theory, K. Kopotun, T. Lyche,

M. Neamtu (eds.), Vanderbildt Univ. Press, 2001, 151–162, ISBN 0-8265-1379-4.
15. J. K. Johnstone and C. L. Bajaj, Sorting Points Along an Algebraic Curve, Siam J. Comput. **19**(5), 1990, 925–927.
16. J. Keyser J, T. Calver, D. Manocha, S. Krishnan, MAPC: A library for Efficient and Exact Manipulation of Algebraic Points and Curves, Proc. of the Fifteenth Annual Symposium on Computational Geometry, SCG'99, ACM Press, 1999, 360–369.
17. D. J. Kriegman, On recognizing and positioning curved 3D objects from image contours, IEEE Trans. Pattern Anal. Mach. Intell.-12, 1990, 1127–1137.
18. R. Martin, H. Shou, I. Voiculescu, A. Bowyer and G. Wang, Comparison of interval methods for plotting algebraic curves, Comput. Aided Geom. Design **19** (2002), 553–581.
19. P. S. Milne, On the solution of a set of polynomial equations. In Symbolic and Numerical Computation for Artificial Intelligence, 1992, 89–102
20. B. H. Ong, An extraction of almost arc length parameterization from parametric curves, Annals of Numerical Mathematics **3** (1996), 317–332.
21. P. Pedersen, Multivariate St urm sequences, In Proceedings of AAECC, Springer-Verlag, 1991, 318–332.
22. A. Ponce, J. Hoogs, and D. Kriegman, On using CAD models to compute the pose of curved 3D objects, Comput. Vision, Graphics and Image processing **55**(2), 1992, 184–197.
23. J. M. Snyder, Interval analysis for computer graphics, SIGGRAPH'92 Proceedings **26**(2), 1992, 121–130.
24. G. Taubin, Distance approximations for rasterizing implicit curves, ACM Trans. on Graphics, **13**(1), 1994.

J. Estrada, D. Martínez, and D. León
ICIMAF, Cuba
{jestrada, dimas, dionne}@icmf.inf.cu

H. Theisel
MPI Informatik
Saarbrücken, Germany
theisel@mpi-sb.mpg.de