Robust Computation of Foot Points on Implicitly Defined Curves

Martin Aigner and Bert Jüttler

Abstract. We introduce a new method for computing the foot point of a given point on an implicitly defined curve and compare it with existing local and global methods. As demonstrated in this paper, the implicit representation of curves and surfaces is useful not only for solving intersection problems, but also for foot point computation.

§1. Introduction

Computing the distance from a given point $\mathbf{p}$ to a curve or surface is a problem that arises in many applications (e.g., collision detection, registration in computer vision, curve and surface fitting etc). This problem is closely related to the computation of the foot points, i.e., the nearest point on the curve or surface, see [1] and the references cited therein.

In this paper we focus on the case of curves which are defined by an implicit representation (as the zero contour of some function f). Most of the results can be generalized to implicitly defined surfaces.

Given a planar curve $\mathcal{C}$, we solve

$$\mathbf{p}_c = \arg\min_{\mathbf{x}\in\mathcal{C}} \|\mathbf{p}-\mathbf{x}\| \tag{1}$$

If the curve $\mathcal{C}$ is the zero contour of a C^1 function $f : \mathbb{R}^2 \to \mathbb{R}$, then the foot point satisfies the two equations

$$f(\mathbf{p}_c) = 0 \quad \text{and} \quad \nabla f(\mathbf{p}_c) \wedge (\mathbf{p}-\mathbf{p}_c) = 0, \tag{2}$$

where $(a_1, a_2) \wedge (b_1, b_2) = a_1 b_2 - a_2 b_1$.

This paper is organized as follows. First we briefly discuss the existing techniques for solving this problem. Then we describe a new method,

xxx and xxx (eds.), pp. 1–4.

ISBN 0-9728482-x-x

which can be seen as a compromise between local and global methods, and demonstrate its properties by several examples. Finally we conclude this paper, addressing – among other issues – the suitability of implicitly defined curves and surfaces for foot point computation.

§2. Existing methods for foot point computation

The existing approaches can be classified as *local* and *global* methods.

2.1. Local methods

The nonlinear system (2) can be solved using a Newton method or modified versions of it (e.g., a damped Newton method). Clearly, this approach ignores the geometrical background of the problem, which causes some difficulties.

- A good initial guess in the vicinity of the solution is required. For instance, the point $\mathbf{p}$ itself may be used. Alternatively, one may use the point on $\mathcal{C}$ found by following the curve of steepest descent emanating from $\mathbf{p}$. (Note, however, that this curve is not guaranteed to hit the curve, since it may lead to a local minimum of f.)
- Newton's method finds one solution of the system of equations. Clearly, there may exist several points that fulfill (2), but not all of them are closest points; some of them may even be local maxima of the distance function.
- This approach needs the first derivative of the equations which are to be solved. In our case, these equations contain first derivatives, therefore $f(x, y)$ has to be C^2 with certain regularity conditions for the second derivative in order to guarantee a solution.

A geometrically motivated method has been proposed by Hartmann [2]. It is a two-step iteration process, which is based on the local approximation of the curve by its tangent , see Fig. 1. In a later version [3], the tangent has been replaced with a tangent parabola.

Algorithm 1. *1. Follow the line of steepest descent to find an initial foot point.*

2. Compute the foot point on the tangent (or tangent parabola) at the previous point.

3. Follow the curve of steepest descent from the foot point on the tangent (parabola) to the curve and calculate the tangent (parabola) in this new approximation of the foot point. Continue with Step 2.

Clearly, this method has problems with singular points (e.g., cusps or double points) of the curve, where no well–defined tangent (or tangent parabola) exists.

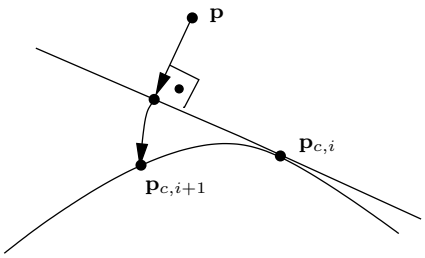

Fig. 1. Hartmann's method.

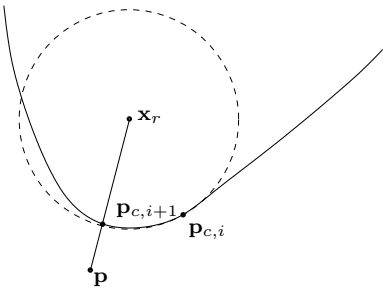

Fig. 2. Redding's method.

A similar method, which relies on the osculating circle instead of a tangent parabola, has been proposed by Redding [5], see Fig. 2. As observed by the author, it may happen in some cases that there is no intersection point, or that the new estimate $\mathbf{p}_{i+1}$ is further away from $\mathbf{p}$ than the previous one. In order to resolve these problems, the algorithm proceeds as follows. On the line connecting $\mathbf{p}_{i+1}$ and $\mathbf{p}_i$ a series of points is taken by halving the distance to $\mathbf{p}_i$ each time. Successively the line from $\mathbf{x}_r$ through each of these points is intersected with the curve. This procedure stops if an intersection point that is closer to $\mathbf{p}$ than $\mathbf{p}_{c,i}$ can be determined.

As another potential problem, the evaluation of the radius of curvature is possible only for C^2 functions f. In addition, the circle of curvature may shrink to a point, if singular points are present.

2.2. A global method: Bézier clipping

Robust global methods for computing foot points can be formulated by exploiting the convex hull property of Bernstein–Bézier representations for solving the non–linear system of equations (2), using the technique of Bézier clipping [4, 7, 8].

First, the two polynomial equations have to be transformed into Bernstein-Bézier form. Due to the convex hull property, parts of the domain $U \subset \mathbb{R}^2$ (typically a box) that do not contain a solution can be identified. These are iteratively clipped away using the de Casteljau subdivision algorithm. When a certain precision is reached, the algorithm stops.

This method finds all solutions of the system (2), which include stationary points of the distance function and singular points on the algebraic curves. Among those candidate solutions, the point having the shortest distance to the given point is the foot point.

While this method avoids the potential convergence problems of the local algorithms, the computations are relatively expensive and it may need many subdivision steps if singular points are present (and the result may be a region instead of a single point, due to numerical noise). Also it

is restricted to functions f belonging to refinable spaces which have bases enjoying a convex-hull property.

§3. Circle Shrinking

This technique, which can be seen as a compromise between local and global methods, is based on the following simple observation:

Consider a curve $\mathcal{C}$, a point $\mathbf{p}$ and the associated closest point on the curve $\mathbf{p}_c$, see Fig. 3a. The line segment connecting $\mathbf{p}$ and $\mathbf{p}_c$ is perpendicular to $\mathcal{C}$ in $\mathbf{p}_c$. Hence it can be used as the radius of a circle with center $\mathbf{p}$ that is tangent to the curve. Since $\mathbf{p}_c$ is the closest point on the curve, no other curve point will be inside the circle. Consequently, the function values in and on the circle either vanish or they have the same sign as the point $\mathbf{p}$.

Based on this fact we formulate an algorithm for calculating the closest point to a given data point.

Algorithm 2. *The point* $\mathbf{p}$ *and a point* $\mathbf{p}_{c,i}$ *on the curve are given. These two points define a circle with center* $\mathbf{p}$*. Without loss of generality we assume that* $f(\mathbf{p}) < 0$*.*

1. *At the point* $\mathbf{p}_{c,i}$ *we evaluate the derivative in the tangent direction of the circle,*
$$\frac{(\mathbf{p}_{c,i} - \mathbf{p}) \wedge \nabla f(\mathbf{p}_{c,i})}{||\mathbf{p}_{c,i} - \mathbf{p}||} \tag{3}$$
If its absolute value is smaller than a certain threshold, we continue with step 5 since $\mathbf{p}_{c,i}$ *is a candidate for the closest point on the curve.*
2. *Otherwise there exists a point* $\mathbf{p}^+$ *in the vicinity of* $\mathbf{p}_{c,i}$ *which lies on the circle and whose function value is positive, see figure 3b. Since the intermediate value theorem holds, the curve intersects the circle between* $\mathbf{p}$ *and* $\mathbf{p}^+$*.*

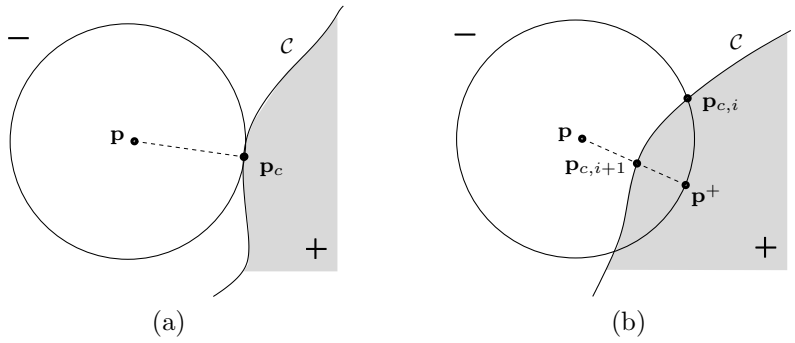

Fig. 3. Circle Shrinking.

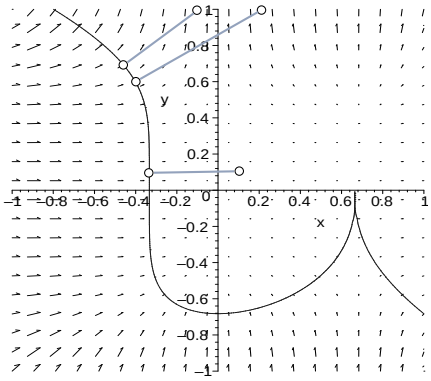

Fig. 4. Algebraic curve with a singularity and its gradient field.

3. *By intersecting the line between* $\mathbf{p}^+$ *and* $\mathbf{p}$ *with* $\mathcal{C}$ *we obtain a point* $\mathbf{p}_{c,i+1}$ *on the curve which is closer to* $\mathbf{p}$ *than* $\mathbf{p}_{c,i}$.
4. *Continue with step 1 until the radius is not reduced by at least a certain threshold.*
5. *Finally, it is verified that no additional branches of the curve inside the circle exist. Otherwise a point* $\mathbf{p}^+$ *can be found inside the circle and the algorithm continues with step 3.*

Often, the initial point $\mathbf{p}_{c,0}$ can be found by tracing the curve of steepest descent emanating from $\mathbf{p}$ until it hits the curve. This may fail, however, if local minima of f are present. In this case, one may adjust the initial radius of the circle via binary search. (Starting with a small initial value, the radius is doubled until the curve is hit.)

The point $\mathbf{p}^+$ (needed in step 2 of the algorithm) is found by computing the (local) maximum of f along the current circle, starting from $\mathbf{p}_{c,i}$. It can be computed by a simple binary search, or using a damped Newton algorithm.

Example. We consider the algebraic curve defined by the polynomial

$$f(x,y) := (y^5 + x^3 - x^2 + \frac{4}{27})(\frac{x}{2} + 1) \tag{4}$$

which has a singular point at $(\frac{2}{3}, 0)$, see figure 4.

According to our numerical experiments, the method has linear convergence. Table 1 shows the convergence quotients $||x_{k+1} - x^*||/||x_k - x^*||$, where x_k is an approximation to the closest point obtained in the k-th step and x^* is the exact value, for three examples (cf. Fig. 4). The computations were stopped at a precision of $1.e-5$.

point	(-0.1,1)		(0.2,1)		(0.1,0.1)	
step	distance	quotient	distance	quotient	distance	quotient
0	0.734026		1.064710		0.569524	
1	0.550519	0.299696	0.798530	0.227745	0.498334	0.477237
2	0.481704	0.123731	0.748622	0.364210	0.436042	0.041529
3	0.474178	0.225404	0.725228	0.181716	0.434339	0.368924
4	0.472325	0.154303	0.722395	0.454704	0.433491	0.148047
5	0.472095	0.317582	0.720984	0.402727	0.433385	0.282078
6	0.472037	0.463064	0.720280	0.259912	0.433358	0.363876
7	0.472008	0.420305	0.720104	0.288833	0.433352	0.562980
8	0.471994	0.310431	0.720060	0.384598		
9	0.471990	0.444685	0.720049	0.599995		
10			0.720038	0.333330		
11			0.720032	6.26508e-08		

Tab. 1. convergence quotients for some data points and $1.e-5$ precision

§4. Detecting additional branches

As a well–known problem, implicitly defined curves (and surfaces) may have additional branches or closed loops. In order to certify the result of the closest point computation, such additional branches inside the circle in step 5 have to be detected.

We assume that f which defines the curve is C^1 and that $||\nabla f||$ can be bounded – at least within the region of interest. The following observation is a simple consequence of the mean value theorem:

Lemma 1. *Let $f(x,y) : \Omega := [a,b] \times [c,d] \to \mathbb{R}$ be a bivariate function and $\mathbf{p}$ such that $f(\mathbf{p}) < 0$. Assume that an upper bound $M \in \mathbb{R}^+$ of $||\nabla f(x,y)||$ exists. Then $f(\mathbf{q}) < 0$ holds for all points $\mathbf{q} \in \Omega$ satisfying*

$$||\mathbf{p} - \mathbf{q}|| \leq \frac{||f(\mathbf{p})||}{M}.$$

Consequently, possible additional branches in step 5 can be detected by sampling sufficiently many points, as follows (see also Fig.).

Algorithm 3. *Consider the last circle of the circle shrinking process.*

1. *Restrict the domain to the bounding square containing the last circle.*
2. *Subdivide this square in to four sub–squares.*
3. *Evaluate the function at the centers of the squares.*
4. *For each evaluation we obtain a ball not containing any additional branch, see Fig. 5a. If these balls are big enough (grey circles), the according squares of the grid can be discarded; otherwise (white circles) they have to be split in 4 smaller ones (see Figure 5b) and the algorithm continues with step 3.*

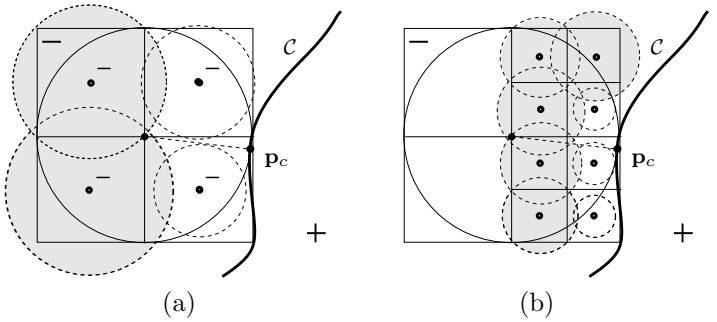

Fig. 5. Excluding additional branches.

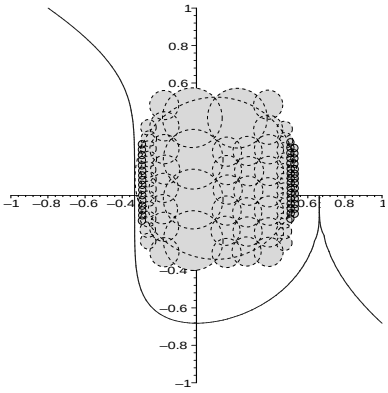

Fig. 6. Example: Excluding additional branches.

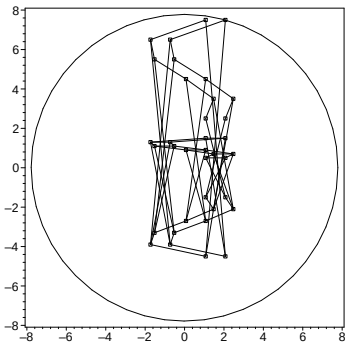

Fig. 7. Generating a gradient bound for Eq. (4).

This is iterated until the whole circle is filled up with grey circles (excluding a part of the boundary, according to the desired accuracy) or until a point with a positive function value is found.

As an example, we applied this procedure to the curve of the previous example (data point (0.1,0.1), see Fig. 6). The number of circles which were generated in the different subdivision levels are 5, 28, 24, 44, 24, 34, 48 and 68. Due to the presence of the neighbouring singular point, a relatively large number of circles was needed.

Remarks.

1. If the function f is a (piecewise) polynomial of degree d, then the gradient can be represented as a planar polynomial patch, or a collection of such patches. For instance, if the region of interested is the box $[a, b] \times [c, d]$, then the gradient has a representation of the

form

$$\nabla f(x,y) = \sum_{i=0}^{d} \sum_{j=0}^{d} A_i(x) B_j(y) \mathbf{b}_{i,j} \tag{5}$$

with certain control points $\mathbf{b}_{i,j} \in \mathbb{R}^2$, where $A_i(x)$ and $B_j(y)$ are the Bernstein polynomials of degree d with respect to the intervals $[a,b]$ and $[c,d]$, respectively. Due to the convex hull property, the norm of the gradient can be bounded by

$$||\nabla f(x,y)|| \leq M = \max_{i,j=0,\ldots,d} ||\mathbf{b}_{i,j}|| \tag{6}$$

In order to obtain a tight bound, the domain of the given polynomial is restricted to the smallest square containing the last circle of the circle shrinking process.

As an example, Fig. 7 shows the control net of the gradient of the polynomial (4) transformed in Bernstein-Bézier basis and restricted to the square containing the last circle about the point $(0.1, 0.1)$. The gradient bound is the radius of the bounding circle (centered at the origin).

2. If the function f is a (possibly piecewise) *polynomial*, the positivity can also be guaranteed by checking the signs of the coefficients of the Bernstein–Bézier representation, again via recursive subdivision. In this situation, this is a valuable alternative to the previously described sampling-based technique.
3. Lemma 1 can also be used for checking for points with a positive function value along the boundary of the last circle, simply by sampling sufficiently many points along the boundary.

§5. Comparison

Finally we compare the different methods. We consider again the curve defined by equation 4. We computed the closest points for 676 points in the unit square. Newton's method gave the correct answer for 341 points (50.4%), Hartmann's algorithm for 637 (94.2%) and Redding's method for 315 (46.6%). Both Bézier Clipping and Circle Shrinking were successful for all points (cf. Fig. 8. The points with a correct foot point computation are marked by squares).

In the case of Hartmann's method we identified two regions where the algorithm did not converge. In region 1 the problems are due to the high curvature of the curve. Following the gradient field from a point in this area leads to an initial point on the branch in the 4th quadrant. Accordingly, a local minimum for the distance is found in this part of the curve. The algorithm is not able to find the global minimum which lies

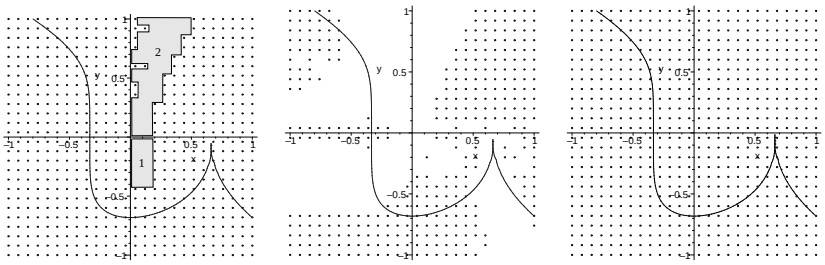

Fig. 8. Comparison of Hartmann's method, Newton iteration and Circle Shrink.

for points of this region in the 3rd quadrant. In region 2 the singularity causes the algorithm to fail. The integral curves emanating from points in region 2 arrive at the singular point, while the closest points lie in the second quadrant.

§6. Concluding remarks

We introduced a new method for computing foot points on implicitly defined curves, which can be seen as a compromise between local iteration methods (which depend heavily on the choice of the initial solution) and the more expensive global methods. The method can be extended to functions f which are only continuous, by modifying the criterion used in the first step of Algorithm 2.

Using the *implicit* representation of a curve and surface (as the zero set of bivariate or trivariate function) may have some advantages when compared with the parametric representation. First, in many cases, a good initial solution can be found simply be tracing the curve of steepest descent emanating from the given point until it hits the curve or surface. Second, checks for collision with the auxiliary circles or spheres, as needed in our algorithm, can be reduced to sampling sufficiently many points, where the required density can easily be controlled.

Several methods of *approximate* implicitization [6], which are an alternative to exact methods (such as resultants) have been developed recently. Using these methods, the generation of an implicit representations is now feasible for general free-form curves and surfaces, not only to low–degree geometric primitives.

Clearly, the idea of circle shrinking can also be applied to curves and surfaces given by a rational *parametric* representation. In this situation, however, the collision check requires the composition of polynomials, which is more expensive.

As a matter of future research, we will generalize the idea of circle shrinking to the 3-dimensional case.

Acknowledgment. This work has been supported by the European Commission through project IST 2001-35512, entitled "Intersection algorithms for geometry based IT-applications using approximate algebraic methods" (GAIA II).

§7. References

1. Anderson, I.J., et al., An efficient and robust algorithm for solving the foot point problem. In: *Mathematical Methods for Curves and Surfaces II*, M. Dæhlen, T. Lyche, and L. L. Schumaker (eds.), Vanderbilt University Press, Nashville, 1998, 9–16.
2. Hartmann, E., The normal form of a planar curve and its application to curve design, *Mathematical Methods for Curves and Surfaces II*, M. Dæhlen, T. Lyche, and L. L. Schumaker (eds.), Vanderbilt University Press, Nashville, 1998, 237–244.
3. Hartmann, E., On the curvature of curves and surfaces defined by normal forms, Comput. Aided Geom. Design **16** (1999), 355–376.
4. Nishita, T., T. Sederberg and M. Kakimoto, Ray tracing trimmed rational surface patches, Computer Graphics (SIGGRAPH '90 proceedings), **24.4** (August 1990), 337–345.
5. Redding, N.J., Implicit polynomials, orthogonal distance regression, and the closest point on a curve, IEEE Trans. Pattern Anal. and Machine Intelligence, **22.2** (2000), 191–199.
6. Wurm, E., J. Thomassen, B. Jüttler and T. Dokken, Comparative Benchmarking of Methods for Approximate Implicitization, in: *Geometric Design and Computing*, M. Luciam and M. Neamtu, (eds.), to appear.
7. Elber, G., and M.-S. Kim, Geometric constraint solver using multivariate rational spline functions, Proceedings of the Sixth ACM Symposium on Solid Modeling and Applications. Ann Arbor, 2001, 1-10.
8. Sherbrooke, E.C., and N.M. Patrikalakis, Computation of the solutions of nonlinear polynomial systems, Comput. Aided Geom. Design **10** (1993), 379–405.

Martin Aigner and Bert Jüttler
Institute of Applied Geometry
Johannes Kepler University, Linz, AUSTRIA
{martin.aigner,bert.juettler}@jku.at
http://www.ag.jku.at