Analyzing and Enhancing the Robustness of Implicit Representations

Martin Aigner and Bert Jüttler
Institute of Applied Geometry
Johannes Kepler University Linz, Austria
martin.aigner@students.jku.at
bert.juettler@jku.at

Myung-Soo Kim
School of Computer Science and Engineering
Seoul National University
Seoul, South Korea
mskim@cse.snu.ac.kr

Abstract

We introduce a robustness measure which allows to analyze implicitly defined curves and surfaces with respect to their stability. It can be used to bound the maximal position error, which is introduced by small perturbations of the coefficients of a curve or surface. It is shown that the robustness of an implicitly defined curve or surface can be enhanced by multiplying it with auxiliary factors.

1. Introduction

Curves and surfaces can be described by various different representations: piecewise polynomial or rational parametric representations [7] (NURBS – Non–Uniform Rational B–Splines), implicitly defined curves and surfaces [1] or level sets [11], and triangular meshes / subdivision surfaces [19]). Traditionally, the different possible representations are tied to certain specific applications, such as Computer Aided Design (parametric representations), numerical simulation and Computer Graphics (meshes), or image processing (level sets). In this paper, we will focus on (piecewise) algebraic representations. More precisely, the free–form curves and surfaces will mainly be described by zero contours of bi– and trivariate spline functions. The use of these algebraic spline curves and surfaces offers several computational advantages.

- The problem of fitting curves and surfaces to scattered data, which is a fundamental task in various applications (e.g., in reverse engineering of geometric objects), can be solved without generating auxiliary triangulations and parameterizations of the data (cf. [9]).
- Algorithms for certain fundamental geometric operations, including intersections with lines and foot point generation, do not need suitable initial values which are often difficult to generate.
- Algebraic curves and surfaces are closed under certain important geometric operations, such as intersection or offsetting.

However, the problems associated with implicitly defined curves and surfaces are also well documented.

- The implicit representation of a curve or surface may contain unwanted branches, or even singular points, in the region of interest. Even if a regular parametric curve or surface (e.g., a cubic) is given, its implicit representation is not guaranteed to be free of these problems.
- Exact implicit representations of low–degree curves or surfaces may have huge data volumes. E.g., the implicit form of a bicubic Bézier patch (which has 16 control points) is a trivariate polynomial of degree 18, which is equipped with 1330 scalar coefficients.
- Small perturbations of the coefficients of an implicitly defined curve or surface may entail huge changes of the shape and the topology. This is different from B–spline or Bernstein–Bézier representations of parametric curves, where the maximum position error is bounded by the coefficient error, due to the convex hull property.

Recently, several approximate techniques for generating implicit representations have emerged [4, 2, 15, 10, 9]. Instead of exact techniques (relying on symbolic tools such as resultants [3], Gröbner bases, or moving curves and surfaces (syzygies) [14]), these methods approximate a given curve or surface by the zero contour of a bi– or trivariate (piecewise) polynomial. Using these techniques helps to avoid the problems associated with implicit representations.

In this paper we will concentrate on the issue of robustness. After summarizing the situation in the parametric case (Section 2), Section 3 introduces a robustness measure for implicit representations, leading to a classification of a given curve or surface with respect to its stability. Section 4 discusses a method for enhancing the robustness of a given representation, by multiplying it with suitable auxiliary factors. Finally we conclude this paper.

2. Parametric curves

For the convenience of the reader we recall how to bound the position error of a polynomial curve. Consider the Bézier curve

$$\mathbf{p}(t) := \sum_{i=0}^{n} \mathbf{b}_i B_i^n(t), \quad t \in [0,1] \tag{1}$$

with the Bernstein polynomials $B_i^n(t) := \binom{n}{i}(1-t)^{n-i}t^i$. We collect the coefficients $\mathbf{b}_i$ in a vector. This coefficient vector is supposed to be subject to a relative error of maximum magnitude ϵ. This results in a perturbed curve $\mathbf{p}_\epsilon(t)$ with control points $\mathbf{b}_i^*$, where $\max_i |\mathbf{b}_i - \mathbf{b}_i^*| \leq \epsilon \max_i |\mathbf{b}_i|$.

The perturbation leads to the position error

$$\delta_{\text{pos}} := \max_{t\in[0,1]} |p(t) - p_\epsilon(t)|$$

which can be bounded by

$$\begin{aligned}\delta_{\text{pos}} &= \max_{t\in[0,1]} \left| \sum_{i=0}^{n} \mathbf{b}_i B_i^n(t) - \sum_{i=0}^{n} \mathbf{b}_i^* B_i^n(t) \right| \\ &\leq \max_i |\mathbf{b}_i - \mathbf{b}_i^*| \leq \epsilon \max_i |\mathbf{b}_i|\end{aligned}$$

Similar to [5], we define a *robustness measure*

$$R_\mathbf{p} := \frac{\delta_{\text{pos}}}{\epsilon}, \tag{2}$$

where δ_{pos} is the maximum position error of all possible perturbed curves. For polynomials in the form (1) it follows immediately from (2) that

$$R_\mathbf{p} \leq \max_i |\mathbf{b}_i|. \tag{3}$$

This ratio allows us to classify a given curve: The smaller the robustness measure, the more robust is the representation of the curve. Multiplying $R_\mathbf{p}$ with the maximal relative error gives a bound on the position error of the curve. The result (3) has an obvious geometric interpretation: in order to obtain a robust representation, the origin of the system of coordinates should be chosen in the vicinity of the curve.

3. Implicitly defined curves and surfaces

Consider an algebraic curve segment $\mathcal{F}$ of order n,

$$\mathcal{F} = \{\mathbf{x} \mid \mathbf{x} \in \Omega \subset \mathbb{R}^2 \ \wedge \ f(\mathbf{x}) = 0\}$$

which is defined in a certain planar domain $\Omega \subset \mathbb{R}^2$ by the zero contour of a bivariate polynomial of degree n. This polynomial is given by its Bernstein–Bézier–representation (cf. [7, 13])

$$f(\mathbf{x}) := \sum_{\substack{i,j,k\in\mathbb{Z}_+ \\ i+j+k=n}} b_{ijk}\, B_{ijk}^n(u,v,w) \tag{4}$$

with $B_{ijk}^n(u,v,w) := \frac{n!}{i!j!k!}u^i v^j w^k$, where (u,v,w) are the barycentric coordinates of a point $\mathbf{x}$ with respect to a non-degenerated triangle $\triangle ABC$.

Again, we assume that the (now scalar–valued) coefficients b_{ijk} are subject to a relative error of maximum magnitude ϵ, resulting in perturbed coefficients b_{ijk}^* such that $\max_{ijk} |b_{ijk} - b_{ijk}^*| \leq \epsilon \max_{ijk} |b_{ijk}|$. Let $\mathcal{F}_\epsilon$ be the perturbed curve, i.e., the zero contour of the perturbed polynomial f_ϵ.

Before discussing the resulting position error, we introduce the following technical notion:

Definition 3.1. *Consider a vector field in $\Omega \subset \mathbb{R}^2$. Two algebraic curves $\mathcal{F}, \mathcal{G} \subset \Omega \subset \mathbb{R}^2$ satisfy the **"I–property"** with respect to the vector field if any integral curve of the vector field which starts from one of them hits the other curve.*

3.1 Bounding the Hausdorff distance

Since no parameterization of the algebraic curves is available, we have to use the more involved concept of the Hausdorff distance in order to quantify the difference between two curves (cf. [8]). The one–sided Hausdorff distance[1] is given by

$$\text{HD}(\mathcal{F}, \mathcal{F}_\epsilon) = \sup_{\mathbf{y}\in\mathcal{F}_\epsilon} \inf_{\mathbf{x}\in\mathcal{F}} ||\mathbf{x} - \mathbf{y}||.$$

Lemma 3.2. *We assume that the gradient fields ∇f and ∇f_ϵ do not vanish in Ω, and that the two curves $\mathcal{F}$ and $\mathcal{F}_\epsilon$ satisfy the I–property with respect to ∇f. If*

$$c \leq ||\nabla f(\mathbf{x})|| \ \text{ and } \ |f_\epsilon(\mathbf{x}) - f(\mathbf{y})| \leq \eta$$

holds for all $\mathbf{x}, \mathbf{y} \in \Omega$, where c and η are certain positive constants, then the one–sided Hausdorff distance can be bounded by

$$\text{HD}(\mathcal{F}, \mathcal{F}_\epsilon) \leq \frac{\eta}{c}.$$

The proof results immediately by applying the mean value theorem to the integral curves which connect the points of both curves. For the sake of brevity, we omit the details (cf. [20]).

We assume that the domain Ω is contained in $\triangle ABC$. Due to the convex–hull property, the maximal "algebraic

[1]The symmetric version is $\max(\text{HD}(\mathcal{F}, \mathcal{F}_\epsilon), \text{HD}(\mathcal{F}_\epsilon, \mathcal{F}))$.

distance" η in Lemma 3.2 can be bounded by

$$\begin{aligned}\eta &= \max_{(x,y)\in\triangle ABC} |f(x,y) - f_\epsilon(x,y)| \\ &\leq \max_{ijk} |b_{ijk} - b^*_{ijk}| \leq \epsilon \max_{ijk} |b_{ijk}|\end{aligned}$$

Analogously to (2), we define a *robustness measure* for algebraic curve segments,

$$R_f = \frac{\mathrm{HD}(\mathcal{F}, \mathcal{F}_\epsilon)}{\epsilon} \tag{5}$$

which can be bounded by[2]

$$R_f \leq \frac{\max |b_{ijk}|}{\min_{(x,y)\in\triangle ABC} ||\nabla f(x,y)||}. \tag{6}$$

Clearly, this upper bound is invariant with respect to scaling: When the coefficients b_{ijk} are multiplied by an arbitrary real constant $\lambda \neq 0$, the upper bound of the robustness constant remains unchanged.

Remark 3.3. Note that the robustness measure is tied to the domain triangle of the curve, since it leads to bounds of the absolute distance error. Consequently, this measure can be used for comparing two curves which are defined with respect to the same domain, but not for two curves with different domains.

In order to compare curves with different domain triangles, a *relative robustness measure* should be used. This could be defined as the ratio between the diameter of the domain and the (absolute) robustness measure.

3.2 Bounding the minimal gradient

In order to bound the robustness measure, the minimal gradient – or at least a lower bound of it – is needed. Hence, we have to compute the two partial derivatives with respect to x and y. Recall that the derivative of a triangular Bézier patch at a point X in the direction R is given by

$$\begin{aligned}&\frac{d}{d\tau} f(X + \tau R) \\ &= n \sum_{\substack{i,j,k\in\mathbb{Z}_+ \\ i+j+k=n-1}} \underbrace{(rb_{i+1jk} + sb_{ij+1k} + tb_{ijk+1})}_{= c_{ijk}} B^{n-1}_{ijk}(u,v,w)\end{aligned}$$

where $r, s, t, (r + s + t = 0)$, are the barycentric coordinates of the direction R with respect to the domain triangle $\triangle ABC = \triangle[(A_1, A_2), (B_1, B_2), (C_1, C_2)]$. For the case of the direction x (set $i = 1, j = 2$), resp. y (with $i = 2, j = 1$) we obtain

$$r = \frac{C_i - B_i}{\Delta_{ij}} \qquad s = \frac{B_i - C_i}{\Delta_{ij}} \qquad t = \frac{B_i - A_i}{\Delta_{ij}}$$

[2]In the case of polynomials with vanishing gradient in $\triangle ABC$, the robustness constant is set to ∞.

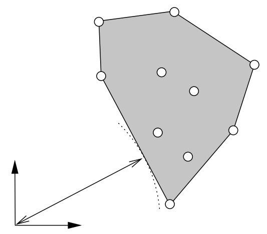


Figure 1. Bounding the minimal gradient

where

$$\Delta_{ij} = (B_j - C_j)(A_i - C_i) - (B_i - C_i)(A_j - C_j).$$

A derivative of a triangular degree n Bézier patch is a triangular Bézier patch of order $n - 1$. Let c^x_{ijk} and c^y_{ijk} be the Bernstein–Bézier coefficients of the derivatives with respect to x and y ($i, j, k \in \mathbb{Z}_+$, $i + j + k = n - 1$). These coefficients are certain linear combinations of the b_{ijk}.

We collect these coefficients to control points

$$\mathbf{d}_{ijk} = (c^x_{ijk}, c^y_{ijk}).$$

This system of control points is associated with the triangular Bézier patch $\nabla f : \triangle ABC \to \mathbb{R}^2$ of degree $n - 1$. Clearly, due to the convex hull property, the gradient $\nabla f = ((\partial/\partial x)f, (\partial/\partial y)f)$ in each point of the domain triangle lies within the convex hull of the control points $\mathbf{d}_{ijk}$.

The minimum gradient length can now be bounded by

$$\min ||\nabla f(x,y)|| \geq \min_{\substack{\mathbf{x}\in \text{ convex hull}\{\mathbf{d}_{ijk}| \\ i+j+k=n-1; i,j,k\in\mathbb{Z}_+\}}} ||\mathbf{x}|| \,. \tag{7}$$

Consequently, we search for the minimal distance of the convex hull of the $\mathbf{d}_{ijk}$ to the origin, see Figure 1. If the origin lies within the convex hull, then the robustness constant is chosen as ∞. Otherwise, this distance is used to bound the denominator in (6). This leads to a lower bound on the robustness measure, expressed in terms of the control points.

Remark 3.4. While the convex hull of a (finite) point set in the plane can easily be generated exactly (e.g., using Graham's algorithm), the computation in the three–dimensional situation (which is needed for bounding the robustness measure for implicitly defined surfaces) is slightly more involved. However, there exist several approximative techniques which can be used instead.

Remark 3.5. The assumptions of Lemma 3.2 include the "I–property". Consequently, the distance bounds are only valid for those segments of the curve, which satisfy this property with respect to the perturbed curve. Since this may not be valid at the boundaries, the original curve segment should be slightly “blown up”, by enlarging the domain triangle. Otherwise, the bounds would be valid only for infinitely small perturbations.

In practice, the relative coefficient error should be very small (much smaller than the 5% used throughout this paper), and the effects of the possible violation of the I–property at the segment boundaries be neglected.

3.3 Two examples

Example 1: For a polynomial f whose gradient vanishes somewhere in the domain, the robustness measure R_f is formally ∞. Since

$$\delta_{\text{pos}} = \epsilon R_f,$$

it is no longer possible to bound the displacement δ_{pos}. In this situation, the perturbed curve is not only subject to a displacement, but some branches of the zero contour may vanish or even new ones may be created. We illustrate this fact by a simple example.

Figure 2a shows the surface given by a polynomial in Bernstein-Bézier form. The three vertical lines represent the vertices of the domain triangle. Figure 2b shows the gradient field of the polynomial, the zero contour of f, and two zero contours after admitting a relative coefficient error of 5%.

The surface has a saddle point, thus the gradient vanishes in a point, as shown in Figure 2b. In addition, the perturbed curves have totally different shapes. Since the displacement of the curve does not depend continuously on the error, we have to exclude zero contours of polynomials with vanishing gradients; no robustness measure can be defined in this situation.

Example 2: In the case of polynomials with non–vanishing gradient, an upper bound of the robustness measure can be obtained from (6). However, if there exists a high diversity in the gradients and in the control points, we may get poor bounds for the robustness measure.

Similar to the previous example, Figure 3a shows the surface given by a polynomial in Bernstein-Bézier form. Figure 3b shows the gradient field of the polynomial, the zero contour of f, and two zero contours after introducing a relative coefficient error of 5%.

Since the gradient does not vanish within the domain triangle, we are now able to bound the minimum gradient length by $\min ||\nabla f|| \geq 0.36$, which leads to the robustness constant $R_f \leq 22$. After introducing 5% relative coefficient error, the maximal position error, which occurs at $x = 5$, equals 0.264. Using the robustness measure, it can be bounded by $5\% \times R_f = 1.1$. Clearly, this is a relatively poor upper bound.

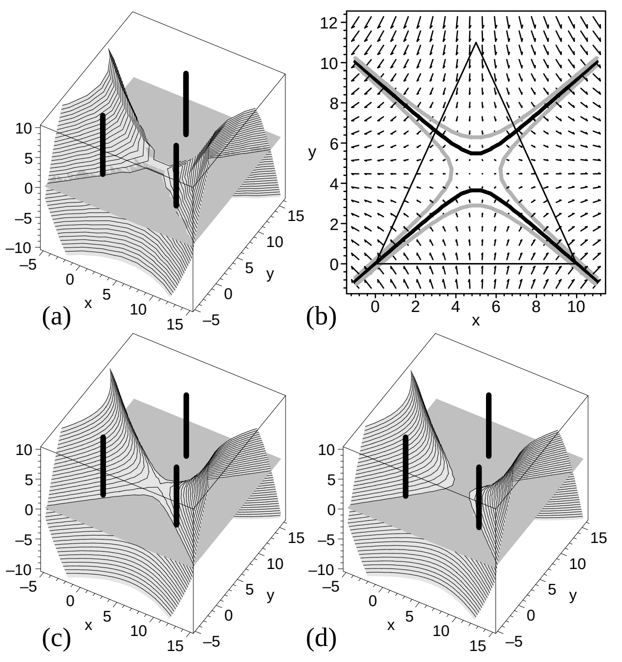


Figure 2. Original bivariate polynomial with vanishing gradient (a). Perturbation with 5% relative error gives the polynomials shown in (c,d). The gradient field and the zero contours before (black) and after (grey) perturbing the Bernstein–Bézier coefficients are visualized in (b).

3.4 Enhancing the gradient bound

In order to get a tighter bound we would need to know the minimal gradient in the vicinity of the curve. With the method described in Section 3.2, we bound the minimal gradient for the whole domain. We may adapt this bound by subdividing the domain triangle into four smaller ones, see Figure 3b.

If the subdivision is realized using de Casteljau’s algorithm, then an extrapolation is needed for the inner triangle. Hence, the computations are no longer numerically stable. Therefore, one should a priori compute the splitting rules symbolically, by combining three consecutive de Casteljau algorithms.

Remark 3.6. Special attention should be paid when computing the position error of the smaller triangles. Multiplying the robustness constant of a subtriangle with the maximal relative error of the original patch leads to a wrong result. Wallner et al. [18] point out that the de Casteljau algorithm propagates the absolute error of the control points. So it is clear that the relative error can become arbitrary high. Thus one must not use the relative error of the original control points, but the relative error of the control net(s)

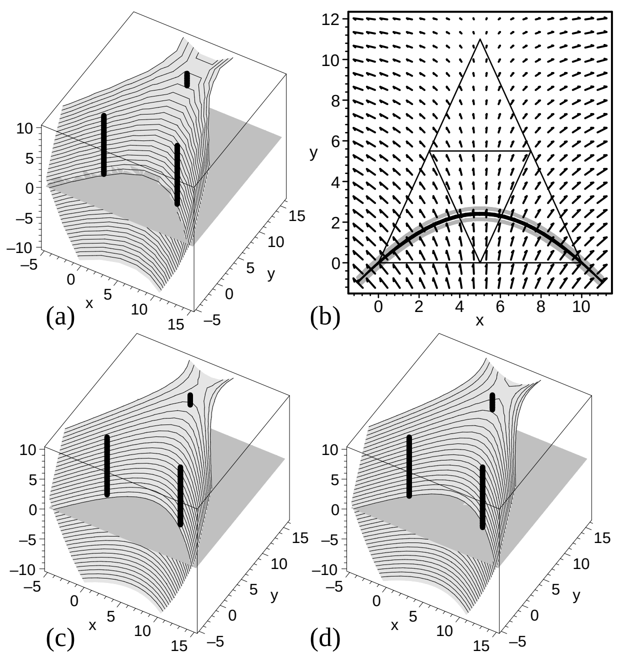


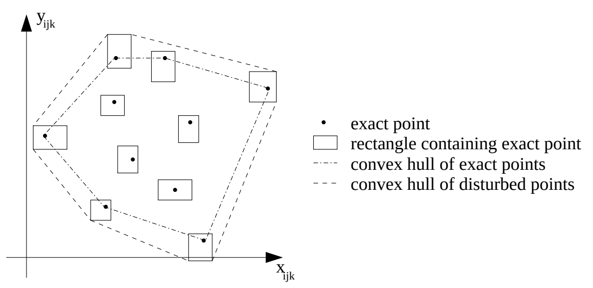
Figure 3. The original bivariate polynomial (a). Perturbation with 5% relative error gives the polynomials shown in (c,d). The gradient field and the zero contours before (black) and after (grey) perturbing the Bernstein–Bézier coefficients are visualized in (b).

of the subtriangle(s).

Example 2 (continued): Recall that the original distance bound (for 5% coefficient error) was 1.1. We split the domain into four triangles and recompute the robustness measure for the innermost triangle (since the biggest error occurs there). This leads to a distance bound of 0.37. This bound is far better, but it still overestimates the position error by about 40%. If we split once more, we get a distance bound of 0.275 which is now within 4.3% of the exact error (0.264).

Remark 3.7. 1. Note that a direct comparison of the robustness measures for the original triangle and the smaller ones (see previous example) does not make much sense, cf. Remark 3.3(2). Therefore, we have compared the absolute error bounds obtained for an relative error of 5% in the original coefficients.

2. In practice, one does not know in which part of the triangle the biggest error occurs and so the minimal gradient has to be computed for every subtriangle which is hit by the curve. As a necessary condition, these triangles are characterized by alternating signs of the Bernstein–Bézier coefficients.



y_{ijk}
x_{ijk}
exact point
rectangle containing exact point
convex hull of exact points
convex hull of disturbed points

Figure 4. Bounding the the minimum gradient length for exact and interval coefficients

3.5 Inexact control points

So far, we bounded the position error of the zero contour of bivariate polynomials in Bernstein–Bézier representation with *exact* control points. However, in practice the coefficients are the result of some numerical computation, e.g. one interpolates some points with a polynomial. In this case it is not possible to specify the points exactly but only an interval which contains the exact value. Note that every floating point number is only reliable within the machine accuracy. Hence, each floating point number is in fact an interval.

So interval arithmetic is needed to compute the derivatives of the Bézier patch. Finally we do not obtain the c_{ijk} as points in $\mathbb{R}^2$, but as certain rectangles which contain the exact points. See [16] for more information on interval arithmetics and related techniques.

Still, we can proceed similar to the case of exact points. We only have to split each rectangle up into 4 points and find the convex hull for these corner points. As an example, Figure 4 shows the gradient patch of a quartic bivariate polynomial. Since the derivative patch is cubic, it has 10 control points.

Example 2 (continued): We consider again the previous curve segment, but now with interval coefficients with accuracy ± 0.01 (the maximum absolute value of the coefficients is equal to 8). In the interval case, the minimum gradient length can be bounded by 0.36, and the robustness measure is equal to 22.3. In the exact situation, these figures were equal to 0.37 and 22.0, respectively.

After subdividing the triangle twice, the innermost triangle has the robustness constant 2.79 (which was 2.75 in the exact case).

3.6 The case of tensor-product functions

The previous results can immediately be generalized to the case of curves and surfaces defined by tensor-product

(TP) polynomials. In the curve case, the implicitly defined curve is described by the zero contour of the polynomial

$$f(x,y) := \sum_{i=0}^{m}\sum_{j=0}^{n} b_{ij} B_i^m(x)\hat{B}_j^n(y), \quad (x,y)\in[0,1]^2, \tag{8}$$

with coefficients b_{ij}, where $B_i^m(x), \hat{B}_j^n(y)$ are the Bernstein polynomials with respect to x and y, respectively, as defined in (1). Again, we admit a maximal relative error ϵ in the maximum norm of the coefficient space,

$$\max_{ij} |b_{ij} - b_{ij}^*| \le \epsilon \max_{ij} |b_{ij}|.$$

Due to the convex hull property, the perturbation of the values of f can be bounded by

$$\max_{(x,y)} |f(x,y) - f_\epsilon(x,y)| \le \epsilon \max_{i,j} |b_{ij}|$$

The partial derivatives with respect to x and y of a TP polynomial are TP polynomials of degree $(m-1,n)$ and $(m,n-1)$, respectively. Thus, in order to represent the gradient as a parametric TP surface patch $[0,1]^2 \to \mathbb{R}^2$ of degree (m,n), degree elevation of the derivatives is needed, see [7]. We can now proceed in the same way as in the case of Bézier triangles; the minimal gradient can be bounded as the minimal distance of the convex hull of the $\mathbf{d}_{ij}$ of the gradient patch to the origin.

3.7 Implicitly defined surfaces

The concept of the robustness measure can easily be generalized to implicitly defined surfaces. Adopting again the Bernstein–Bézier technique, such surfaces are given by the zero contours of polynomials of the form

$$f(x,y,z) \;:=\; \sum_{\substack{i,j,k,l\in\mathbb{Z}_+\\ i+j+k+l=n}} b_{ijkl} B_{ijkl}^n(r,s,t,u)$$

with trivariate Bernstein polynomials B_{ijkl}^n, see [1, 7]. These polynomials are now defined with respect to a domain tetrahedron $\triangle ABCD$.

Lemma 3.2 can easily be generalized to the surface case. As in the bivariate case we define the robustness measure, which can be bounded as follows:

$$R_f := \frac{\delta_{\text{pos}}}{\epsilon} \le \frac{\max_{i+j+k+l=n} |b_{ijkl}|}{\min_{(x,y,z)\in\triangle ABCD} ||\nabla f||}.$$

The gradient field of f now defines a trivariate vector–valued polynomial. The minimal gradient can again be bounded by the minimum distance of the convex hull of the control points from the origin. Clearly, the computation of this lower bound now involves convex hull computations in three–dimensional space, where approximative techniques will be preferred.

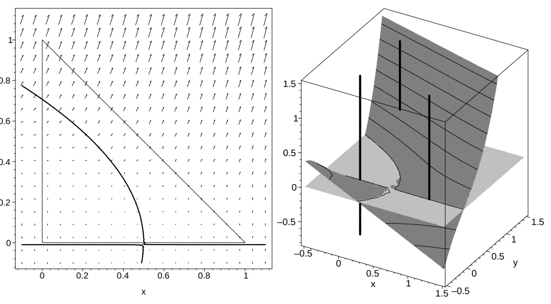


Figure 5. Example 3 – product of parabola and straight line

4 Robustness enhancement

After analyzing the robustness, we develop a method for improving the robustness of a given implicitly defined curve or surface.

4.1 Motivation

As observed in the previous section, the robustness can often be improved by subdividing the domain. However, as demonstrated by the following example, this approach will not work in all cases.

Example 3: We consider a the cubic polynomial

$$f(x,y) = (y^2 + x - 0.5)(y + 0.01), \tag{9}$$

whose zero contour is a parabola and a straight line, see Figure 5. The domain triangle is shown in the figure. The singular point is not contained in the domain, but close to it. As to be expected, the robustness measure is rather poor, $R_f \le 50.5051$. After a subdivision step we get the tighter bound $R_f \le 17.0825$, which seems to be an important improvement. However, this improved robustness measure does not give better bounds on the position error: for the original triangle (with 5% coefficient error) the bound was

$$\delta_{\text{pos}} \le 50.5051 \cdot 0.05 = 2.525255.$$

After the subdivision, the position error can be bounded by

$$\delta_{\text{pos}} \le R_f \epsilon_2 \le 17.0825 \cdot 0.147 \approx 2.52488.$$

(Here, $\epsilon_2 = 0.147$ is the relative error with respect to one of the four triangles obtained after the subdivision). Consequently, the robustness of this curve representation is rather poor. This is due to the singular point which is close to the domain triangle.

4.2 Multiplication by an auxiliary factor

The representation of an implicitly defined curve or surface is not unique, since the defining function $f(x, y)$ can be multiplied by any bivariate function $\lambda(x, y)$. Clearly, while this multiplication does not change the original zero contour, it might introduce additional branches, which have to be kept outside the domain triangle. Hence, λ has to be strictly positive (or negative) on the entire domain.

We will now exploit this observation in order to improve the robustness of the representation. Clearly, if one already has a perturbed representation (with some coefficient error), then this multiplication will not improve matters, as it cannot eliminate the noise from the coefficients. On the other hand, if a highly precise implicit representation is given (e.g., based on extended arithmetic, or even exact arithmetic, or just double precision), but it has to be converted into to less precise format (such as standard floating point numbers), then an enhancement of the robustness can be highly useful for improving the accuracy of the representation.

High robustness measures (which lead to large position errors) are caused by high variations in the gradient field. Consequently, we choose the auxiliary factor such that the gradient field becomes more uniform ($\|\nabla f\| \approx 1$). As a heuristic observation, uniform gradient fields can be expected to give a more robust representation. This is similar to the idea of "reinitialization" in the level set framework [11, 17].

If a bivariate function has uniform gradients, $\|\nabla f\| \equiv 1$, its graph forms a so–called *surface of constant slope*. These surfaces, which are special ruled surfaces, have thoroughly been studied in classical differential geometry. Due material properties of the material, these surfaces are naturally generated by piling material like sand or grains.

Consider the surface of constant slope which is associated with a given curve. Generally, this surface does not exist in the entire domain, but only in a certain neighbourhood of the given curve, which is bounded by the curve's *evolute*. As an example, we consider the curve in Figure 6, whose evolute is the unit circle. The associated surface of constant slope (right) is defined only in the outer part of the evolute (visualized by the cylinder).

4.3 Computing the auxiliary factor

Starting from the observations in the previous section, we aim at improving the robustness measure of a given function $f(x, y)$ by multiplying it with another bivariate polynomial $\lambda(x, y)$, called the *mollifying function*. The goal of this procedure is to obtain gradients of constant length.

The absolute length of $||\nabla f||$ is of no importance, since one can always multiply the defining function with a con-

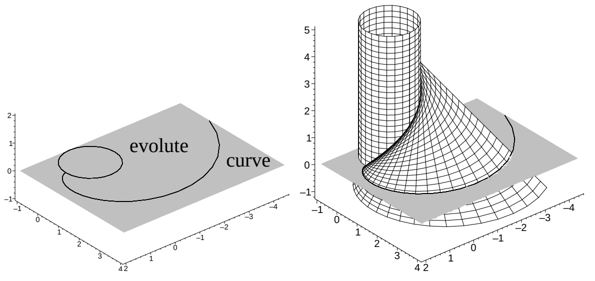


Figure 6. A planar curve and its evolute (left). Surface of constant slope through the given curve (right).

stant factor without changing the robustness of the curve. For the sake of simplicity, we search for a unit gradient field. As explained earlier, this cannot be fulfilled on the entire domain for an arbitrary surface. However, in the vicinity of the curve we may get a representation similar to a surface of constant slope. In order to preserve the shape of the original zero contour, this polynomial has to be non-zero on the entire domain. Thus we search for a so–called "mollifying function" $\lambda(x, y)$ taking positive values on $\triangle ABC$ such that

$$\begin{aligned} & ||\nabla(\lambda(x,y)f(x,y))|| \\ = \; & ||f(x,y)\nabla\lambda(x,y) + \lambda(x,y)\nabla f(x,y)|| \overset{!}{=} 1 \end{aligned} \tag{10}$$

We have to distinguish between several parts of the domain:

1) points on the zero contour,
2) points in the vicinity of the curve,
3) points far away from the curve.

First we apply condition (10) to all points $(\bar{x}, \bar{y})$ on the zero contour and obtain

$$\begin{aligned} & ||f(\bar{x},\bar{y})\nabla\lambda(\bar{x},\bar{y}) + \lambda(\bar{x},\bar{y})\nabla f(\bar{x},\bar{y})|| \\ = \; & ||\lambda(\bar{x},\bar{y})\nabla f(\bar{x},\bar{y})|| \overset{!}{=} 1 \end{aligned} \tag{11}$$

Since we discuss only curves without singularities (i.e. $||\nabla f(\bar{x}, \bar{y})|| \neq 0$), this gives

$$\lambda(\bar{x},\bar{y}) = \frac{1}{||\nabla f(\bar{x},\bar{y})||} \tag{12}$$

For all $(\bar{x}, \bar{y}) \in \mathcal{F}$ we get $||\nabla(\lambda(\bar{x}, \bar{y})f(\bar{x}, \bar{y}))|| \equiv 1$.

Second we consider the neighbourhood of $\mathcal{F}$. Here,

$$\lambda(x,y) \approx \frac{1}{||\nabla f(x,y)||} \tag{13}$$

is still reliable, since $f(x, y)$ is rather small and thus

$$||f(x,y)\nabla\lambda(x,y) + \lambda(x,y)\nabla f(x,y)|| \approx 1. \tag{14}$$

Third, for points further away from the zero contour, the first term in (14) may become big and so the approximation in (13) is no longer valid, so we need another condition to calculate $\lambda(x, y)$. If the gradients of the surface were all unit vectors, then

$$\text{dist}_{\mathcal{F}}(x, y) = \lambda(x, y)|f(x, y)|$$

would hold, where $\text{dist}_{\mathcal{F}}(x, y)$ is the footpoint distance of the point (x, y). (For details on footpoint calculation see [6].) So, at all points which are further away from the zero contour, the polynomial $\lambda(x, y)$ should fulfill

$$\lambda(x, y) = \frac{\text{dist}_{\mathcal{F}}(x, y)}{|f(x, y)|}. \tag{15}$$

In order to compute the mollifying function, we sample the expected values λ_i at a sufficiently high number of points (x_i, y_i) $(i = 1 \dots N)$ in the domain. The values are determined from (12) if the points are close to the zero contour, or from (15) otherwise. Then, by fitting a bivariate polynomial to these points in the least squares, we obtain the mollifying function $\lambda(x, y)$.

When computing the points and fitting the surface there arise some issues that have to be addressed:

- First, a bound $c_1 \in \mathbb{R}^+$ has to be introduced in order to decide where to switch between condition (12) and (15). Furthermore we may exclude points which are too far away from the zero contour. More precisely, condition (12) is used whenever $c_1 < |f(x, y)| \le c_2$.

- Another problem is the degree of the λ. Surfaces with a high degree provide a better approximation of the points. On the other hand, if there is a high variation in the values of $\lambda(x, y)$, they tend to oscillate and so additional zero contours, or even points with vanishing gradients, may be created.

  This problem can be addressed by using a series of linear enhancements. That is, instead of multiplying the original representation with one polynomial of degree n, it is multiplied by n linear ones. See the example in the next section.

- According to our numerical experiments, it is better to exclude sample points ('outliers') where the estimated value of λ exceeds a certain threshold, i.e., $|\lambda_i| > C, C \in \mathbb{R}^+$.

- Finally, in order to compute the enhanced representation, the Bernstein–Bézier representations of the product $h(x, y) = f(x, y)\lambda(x, y)$ is needed. If $f(x, y)$ and $\lambda(x, y)$ have degrees m and n, respectively, their product is a bivariate polynomial of degree $m+n$. The new coefficients can be generated with the help of suitable product formulas for polynomials in Bernstein form, cf. [7].

Summing up, it is difficult to decide which combination of parameters provides the best enhancement. Some numerical experiences are reported in the next section.

4.4 Three examples

Example 4: Figure 7 shows a bivariate polynomial of degree 4 and its gradient field. The gradient length can be bounded from below by $\frac{1}{15}$, which leads to the robustness measure $R_f \le 30$. As one can see, the lowest gradients occur on the zero contour, which is simply a straight line. In order to enhance the robustness, we tested all combinations of the following parameters: $C = 1(1)5$, $c_1 = 0(0.05)c_2$, $c_2 = 0(0.05)1$ and degree=1(1)5. The best result is shown in Figure 8, which was obtained by choosing a quartic polynomial λ, and the parameters $C = 4$, $c_2 = 1$, $c_1 = 0.25$. Figure 8a shows f, along with the two sets of sampled points and the mollifying function λ, while Figure 8b visualizes the enhanced representation.

In order to compare the product $\lambda(x, y)f(x, y)$ with the original representation $f(x, y)$, we have to raise the degree of $f(x, y)$, in order to get Bernstein–Bézier representations of the same degree. The graph of the product λf is almost a plane, see Figure 8b. While the robustness measure of the original surface could be bounded by $R_f \le 30$, the enhanced representation gives the improved value $R_{\lambda f} \le 4.67$. Consequently, the new representations is about six times more robust than the original one.

In this example, the mollifying function has degree 4. However, this high degree was only possible because of the simple shape of the curve. According to our numerical experiments, often only a degree 1 enhancement is possible for more complicated shapes, since mollifying functions of higher degrees introduced new zero contours or points with vanishing gradients. This problem can be overcome by applying iteratively a series of linear enhancements.

Example 3 (continued): We consider again the cubic polynomial, whose zero contour factors into a parabola and a straight line. For the original polynomial we found $R_f \le 50.50$, which is rather poor. The reason for this is the singular point which is close to the domain. Hence the lower bound for the gradient was 0.01. Now we apply several steps of enhancement by linear polynomials. The results are shown in table 1. The upper bounds for R_f decrease successively with each step. Figure 9 compares the gradient field of the original patch and the gradient field of the enhanced representation after 5 iterations. Of course, the effect of the singular point cannot be eliminated, since the zero contour remains unchanged. Still, the robustness can be improved significantly. Clearly, this is a rather pathological example, due to the singularity which is close to the

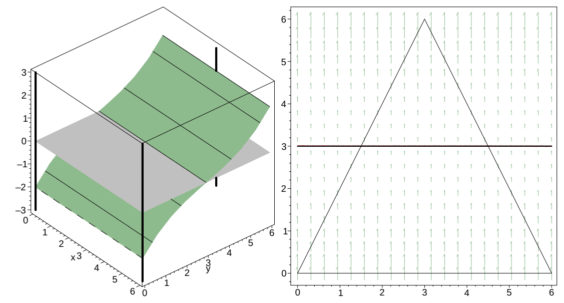


Figure 7. Bivariate polynomial of degree 4.

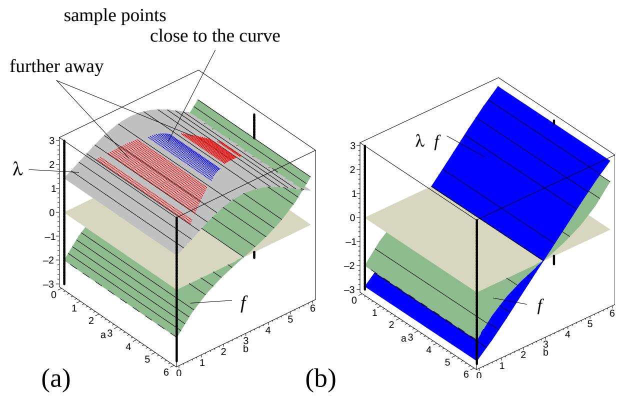


Figure 8. sampled points and mollifying function (a). Enhanced surface with more uniform gradients (b).

boundary of the domain. Still, the method leads to a significant improvement of the robustness.

Example 5: Finally, we have applied the method to data provided by one of our industrial partners, see Figure 10. First, an approximate implicit representation of the parametric surface shown in the figure has been generated using a variant of the method described in [9]; is a tensor–product polynomial of degree $6 \times 6 \times 6$. Among other applications, this implicit representation is useful for detecting the self–intersection (red curve). After restricting the polynomial to a suitable box, we obtained the grey surface patch which has very poor robustness measure (3,582). This is due to

degree	R_f	$\max \lvert b_{ijk} \rvert$	$\min \lvert\lvert \nabla f \rvert\rvert$
3	50.50	0.51	0.01
4	16.45	0.45	0.027
5	10.99	0.56	0.051
6	9.89	0.55	0.056
7	9.28	0.51	0.055
8	8.70	0.47	0.054

Table 1. Example 3 – Iterative enhancement by linear factors

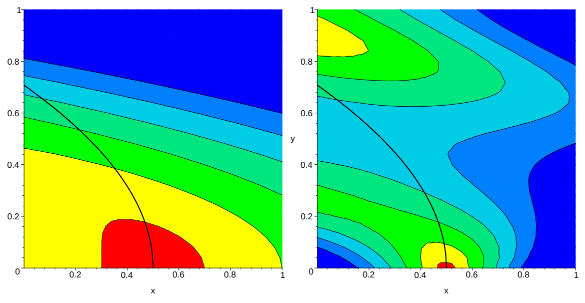


Figure 9. Example 3 – original (left) and enhanced (right) gradient field. The plots visualize the level curves $||\nabla f|| =$ constant. Note that the size of the critical region (red) has shrunk a lot.

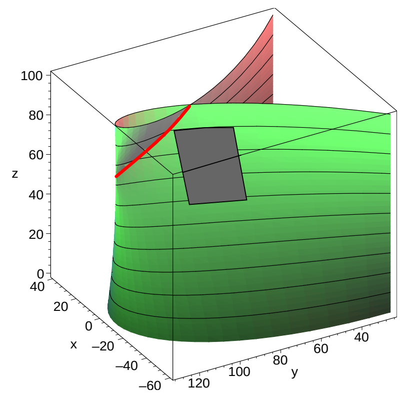


Figure 10. (Industrial) Example 5 (data courtesy of think3): The robustness measure of the grey surface patch has been improved from 3,582 to 2,358.

the neighbouring singularity of the surface. In this case, the mollifying factor was chosen as a tri–linear polynomial. The enhanced representation had a robustness measure of 2,358. Consequently, the geometrical errors caused by coefficient errors have been reduced to approximately 66%.

5 Concluding remarks

In the case of parametric curves and surfaces, bounds for the position errors of perturbed curves and surfaces can easily be generated. In this paper, we generalized this approach to the implicit case. More precisely, we introduced a *robustness measure* which allows to classify implicitly defined curves/surfaces with respect to their stability. The position error can be bounded by the relative error in the coefficients, multiplied with the robustness measure.

As a heuristic observation, zero contours of functions with a uniform gradient field ($||\nabla f|| \approx 1$) have optimal stability. Ideally, the graph of such a function would be a surface of constant slope. This observation leads to the idea of *robustness enhancement:* the implicit representation of a curve (or surface) can be made more robust by multiplying it with a suitable auxiliary factor (called the mollifying function). As demonstrated by the numerical example, this approach indeed improves the implicit representation, even in pathological cases.

Currently, the computation of the mollifying function relies on (1.) generating sample points and (2.) fitting a function to them. As a matter of future research, we will investigate other, more direct methods for computing the auxiliary factors. For instance, the auxiliary factor could be generated by comparing the first terms of the Taylor expansion of λf with the Taylor expansion of the distance function. This is also closely related to a more general question: Given a curve or surface, what is the "optimal" function f (in a suitable sense) such that the zero contour is the given curve or surface? Clearly, the distance function (which assigns – to each point – the distance from the curve) cannot play this role, since it is not smooth globally. Once a optimal global representation has been found, it could then be approximated by polynomials, leading to an optimal robust representation.

Acknowledgment. The first two authors have been supported by the European Commission through RTD project IST 2002-35512, entitled "Intersection algorithms for geometry based IT-applications using approximate algebraic methods.", which is funded by the Future and Emerging Technologies (FET) arm of the 5th Framework programme.

References

[1] J. Bloomenthal (ed.), *Introduction to implicit surfaces*, Morgan Kaufmann, San Francisco, 1997.

[2] R.M. Corless, M.W. Giesbrecht, I.S. Kotsireas and S.M. Watt, Numerical implicitization of parametric hypersurfaces with linear algebra, AISC'2000 Proceedings, Madrid, Spain. LNAI 1930, Springer.

[3] D. Cox, J. Little, and D. O'Shea, *Ideals, varieties and Algorithms*, Springer, New York, 1997.

[4] T. Dokken, Approximate implicitization, in: T. Lyche et al. (eds.), *Mathematical methods for curves and surface*, Vanderbilt University Press, Nashville, 2001, 81–102.

[5] R.T. Farouki, V.T. Rajan. *On the numerical condition of algebraic curves and surfaces 1.implicit equations.* Computer Aided Geometric Design 1988; 5:215-252.

[6] E. Hartmann. *The normal form of a planar curve and its application to curve design.* Mathematical Methods for Curves and Surfaces II, Morten Daelen, Tom Lyche, Larry L. Schumaker (eds), Vanderbilt University Press, Nashville & London, 1998, 237-244.

[7] J. Hoschek, and D. Lasser. *Fundamentals of Computer Aided Geometric Design.* AK Peters, Wellesly (Mass.), 1993.

[8] B. Jüttler, *Bounding the Hausdorff Distance of Implicitly Defined and/or Parametric Curves.* Mathematical Methods in CAGD, Tom Lyche and Larry L. Schumaker (eds), Vanderbilt University Press, Nashville & London, 2000, 223–232.

[9] B. Jüttler and A. Felis, Least–squares fitting of algebraic spline surfaces, *Advances in Computational Mathematics* **17** (2002), 135–152.

[10] B. Jüttler, J. Schicho and M. Shalaby, Spline implicitization of planar curves, in: T. Lyche et al. (eds.), Curve and Surface Design: St. Malo 2002, Nashboro Press, Brentwood 2003, 225–234.

[11] S.J. Osher and R. Fedkiw, Level Set Methods and Dynamic Implicit Surfaces, Springer, Berlin 2002.

[12] N.M. Patrikalakis, T. Maekawa. *Intersection Problems.* Handbook of Computer Aided Geometric Design, edited by G.Farin, J.Hoschek, M.-S.Kim, North-Holland, 2002; 623-649.

[13] T.W. Sederberg, Planar piecewise algebraic curves, Computer Aided Geom. Design **1** (1984), 241–255.

[14] T.W. Sederberg and F. Chen, *Implicitization using moving curves and surfaces*, Computer Graphics 29 (SIGGRAPH'95 Conference Proceedings), 301–308.

[15] T. Sederberg, J. Zheng, K. Klimaszewski, and T. Dokken, Approximate implicitization using monoid curves and surfaces, *Graphical Models and Image Processing*, **61** (1999), 177–198.

[16] H. Shou, H. Lin, R.R. Martin and G. Wang, Modified Affine Arithmetic is More Accurate than Centred Interval Arithmetic or Affine Arithmetic, in : The Mathematics of Surfaces X, Eds. M. J. Wilson, R. R. Martin, 355-365, Springer LNCS 2768, Heidelberg 2003.

[17] M. Sussman, P. Smereca, and S. Osher, A level set approach for computing solutions to Incompressible Two-phase flow, *J. Comput. Phys.* **114** (1994), 146-159.

[18] J. Wallner, R. Krasauskas, H. Pottmann. *Error propagation in geometric constructions.* Computer Aided Geometric Design 2000; 32:631-641.

[19] J. Warren, *Subdivision Methods For Geometric Design: A Constructive Approach*, Morgan Kaufmann, San Francsico, 2003.

[20] E. Wings and B. Jüttler, Generating tool paths on surfaces for a numerically controlled calotte cutting system, *Computer-Aided Design*, in press.