Bernstein Based Arithmetic Featuring de Casteljau

Dominique Michelucci Sebti Foufou* Loïc Lamarque
David Ménegaux

Lab. Le2i, UMR CNRS 5158, Université de Bourgogne
BP 47870, 21078 Dijon, France
{dmichel, sfoufou, loic.lamarque, david.menegaux}@u-bourgogne.fr

Abstract

Bernstein based interval analysis permits to trace algebraic curves and surfaces. In this paper, we propose to use the classical de Casteljau algorithm to improve the efficiency of the Bernstein based method. The proposed tracing method gives significant results with functions of high degree. These results are illustrated and compared with other interval analysis approaches.

1 Introduction

Interval Analysis (IA), or Interval Arithmetic, has been used in a variety of domains for reliable computations [6], but its excessive conservatism severely restricts its benefits, more particularly for CAD-CAM applications. This paper aims at showing how IA can be beneficially used and proposes a set of tools to enable this use. The proposed IA is based on ideas by Bernstein, Bézier and de Casteljau. We call it the Bernstein based arithmetic. Examples of algebraic curves rendering using this arithmetic will be given.

We use the classical subdivision method in section 2 to trace algebraic curves with several interval arithmetics, which permits to visually compare the performances of these arithmetics, and to visually prove the superiority of the Bernstein based arithmetic. The subdivision method can be combined with continuation methods (*e.g.* marching cubes) to get piecewise linear approximations; these approximations ensure not to forget any component, which is the main problem with continuation methods. Section 2 makes a short recall about the main existing methods for the plotting of algebraic curves and surfaces: we choose to focus on standard IA, Taubin's method and affine arithmetic. Then the Bernstein based arithmetic is presented in section 3. Section 4 illustrates and compares the performances of different interval arithmetic methods with the recursive subdivision method for plotting algebraic curves $f(x, y) = 0$. This subdivision method can also be used in 3D, for voxelization or boundary evaluation of implicit surfaces or CSG trees. It can also be used beyond 3D, for instance to solve systems of polynomial equations.

*Corresponding author. Currently guest researcher at the National Institute of Standards and Technology, Gaithersburg, MD 20899, USA. sfoufou@cme.nist.gov

2 Related works

2.1 The naive arithmetic

The main principle of interval arithmetic consists in replacing a given value by an enclosing interval of floating-point numbers. The interval is defined by $X = [a, b] = \{x \mid a \leq x \leq b\}$ as a set of real numbers. Standards operations can also be defined [8], given two intervals X and Y: $X \odot Y = \{x \odot y \mid x \in X, y \in Y\}$, with $\odot = +, -, \times$ or $\div$.
To be fully conservative, lower bounds are rounded towards $-\infty$, while upper bounds are rounded towards $+\infty$. In practice, it is difficult to control, in a portable way, the rounding mode with high level programming languages and typically this precaution is not employed. Usually it has no consequence because the interval width is huge in front of the ULP (Unit in the Last Place: the difference between two contiguous floating point numbers), and because of the consequences of the wrapping effect [9, 11].

In order to plot the implicit algebraic curve $f(x, y) = 0$, the subdivision method evaluates $f(x, y)$ for $x \in X = \{x^-, x^+\}, y \in Y = \{y^-, y^+\}$, which gives $F = [F^-, F^+]$. If F does not contain 0, the curve is not in the box $X \times Y$, otherwise the box is subdivided in four equal parts, considering the middles of X and Y. This operation is recursively applied until a threshold is reached; then remaining boxes are drawn. Since IA is conservative, no box is forgotten, *i.e.* displayed boxes cover the curve. Conversely, IA can be too much conservative, especially with naive interval arithmetic: some of drawn boxes are not cut by the curve. It typically happens close to singular points and points with high curvature (Figure 1).

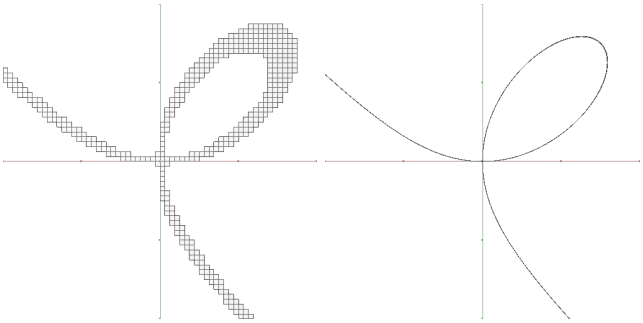

Figure 1: $F_{-1}(x,y) = 0$ in the square $[-2,2] \times [-2,2]$, with $F_k(x,y) = x^3 + 3kxy + y^3$. On the left, the folium is covered with a set of boxes using naive IA and the subdivision method.

2.2 The Taubin's method

Taubin [15, 16] expresses the polynomial $f(x,y)$ as:

$$f(x,y) = f_0 + f_1(x,y) + f_2(x,y) + \dots f_n(x,y)$$

where $f_i(x,y)$ are degree i homogeneous polynomials. Let F_i be the sum of absolute values of coefficients of an homogeneous polynomial f_i, then:

$$f([-a,a]) \in [f_0 - \sum_{i=1}^{i=n} F_i a^i, f_0 + \sum_{i=1}^{i=n} F_i a^i]$$

Taubin's method finds out that for $f(x) = x(1-x)$, $f([0,1])$ lies in $[0, 0.25]$, which is sharper than the $[0,1]$ that results from the naive IA. Taubin uses this homogeneous polynomials based expression for the tracing of 2D implicit curves $f(x,y) = 0$ and the voxelization of 3D implicit surfaces $f(x,y,z) = 0$.

2.3 The affine arithmetic

The affine arithmetic was proposed by Comba and Stolfi [1], and later by Figueiredo *et al.* [2, 4] to decrease the large overestimation observed in classical IA (*e.g.* error explosion problem). Intervals are made even smaller by taking into account the correlations between variables, a quantity x is represented by its affine form:

$$\hat{x} = x_0 + x_1\varepsilon_1 + x_2\varepsilon_2 + \dots + x_n\varepsilon_n$$

where the x_i are finite floating-point numbers and the ε_i are unknown and independent real numbers in $[-1,1]$. x_0 is the central value of $\hat{x}$, other x_i are its partial derivations and the ε_i are the noise symbols. Given a quantity x in standard IA within the interval $[a,b]$, the corresponding affine form of x is $\hat{x} = x_0 + x_1\varepsilon_1$ where $x_0 = (b+a)/2$ and $x_1 = (b-a)/2$. The same noise symbol may appear in two or more quantities, which indicates their dependencies.

3 The Bernstein method

3.1 Bernstein based intervals

The polynomial curve $f(x,y) = 0$ inside the square $[0,1] \times [0,1]$ is considered as the intersection curve between the plane $z = 0$ and the Bézier patch: $x = x, y = y, z = f(x,y)$. If f has degrees m in x and n in y, the control points of the Bézier patch are $P_{i,j} = (i/(m), j/(n), z_{i,j}), i = 0 \dots m, j = 0 \dots n$, and the coefficients $z_{i,j}$ are computed as follows: $f(x,y) = XFY^t, X = (1, x, x^2, \dots x^m)), Y = (1, y, y^2, \dots y^n)$

The Bernstein base for degree n polynomials is

$$B_t = (B_0^n(t), B_1^n(t), \dots B_d^n(t)).$$

The canonical base is: $T = (1, t, t^2 \dots t^n)$. The power base and the Bernstein base are related by:

$$t^i = \sum_{j=i}^{n} \frac{\binom{j}{i}}{\binom{n}{i}} B_j^n(t)$$

and

$$B_i^n(t) = \sum_{j=i}^{n} (-1)^{j-i} \binom{n}{j} \binom{j}{i} t^j.$$

Since $f(x,y) = XFY^t = B_x B B_y^t$ in the Bernstein base, the matrix B contains the control points of the surface. The convex hull property of the Bernstein-Bézier base (Figure 2) guarantees that $f(x,y)$ lies inside the convex hull of its control points; here the convex hull for $z = f(x,y), x \in [0,1], y \in [0,1]$ is just the interval $[\min(z_{i,j}), \max(z_{i,j})]$.

If this interval contains 0, the classical de Casteljau algorithm permits to divide the patch in 2, or in 4 sub - patches - no translation to the canonical base is needed. Numerically, the matrix B of control points is divided with de Casteljau algorithm to obtain new control points. Further recursive applications of the algorithm to each new patch give a better approximation of the curve.

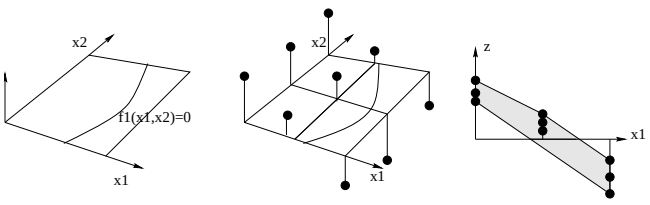

Figure 2: Convex hull property of the Bernstein-Bézier base.

3.2 Shrinking intervals with linear programming

Bernstein based intervals can be shrunken with linear programming, as already remarked by N. Patrikalakis *et al.* [5, 12, 14]. This optimization uses properties of the Bernstein base and is not possible with the naive IA and with Taubin's method.

A Bézier surface patch lies inside the convex hull of its control points. The intersection between this convex hull and the plane $z = 0$ is a convex polygon, which encloses the curve arc to trace. Finding the smallest box $[x_0, x_1] \times [y_0, y_1]$ enclosing this polygon reduces to 4 linear programming problems. For instance x_0 is the solution of this linear programming problem:

$$\begin{aligned} X &= \sum_{i=0}^{i=m} \sum_{j=0}^{j=n} \lambda_{i,j} P_{i,j} \\ 1 &= \sum_{i=0}^{i=m} \sum_{j=0}^{j=n} \lambda_{i,j} \qquad (1) \\ X &= (x, y, z), \quad z = 0, \quad 0 \leq \lambda_{i,j} \\ x_0 &= \min x \end{aligned}$$

Actually, the problem is even simpler than that. For any dimension d, if there is only one equation $z = f(x_1, x_2 \dots x_d) = 0$, then finding the smallest enclosing box $[x_1^-, x_1^+] \times [x_2^-, x_2^+] \times \dots [x_d^-, x_d^+]$ reduces to d very simple 2D convex hull problems: to find $[x_1^-, x_1^+]$, one should project all control points on the plane O, x_1, z, compute the 2D convex hull of these 2D points, and compute the intersection with the line $z = 0$: it is $[x_1^-, x_1^+]$ (see Figure 2 right).

The shrinking method extends to all dimension, and also to any number of equations. This shrinking method can also be used to solve systems of polynomial equations. An important optimization consists in preprocessing the system $F(X) = 0$ to be solved as follows: $F(X) = 0$ has the same roots as $F(X)F'(X)^{-1} = 0$. A single step of this shrinking method may be sufficient to solve the system. It ensures quadratic convergence near a root.

4 Comparison

It is well known that the naive IA is optimal when each variable occurs once in the evaluated expression, although this case rarely happens. Clearly the Taubin's method is much better than the naive one, and the Bernstein method is better than the Taubin's one (see Figures 4, 5 and 6), more details can be found in [3] pages 418–421. Martin *et al.* [7] did not reach the same conclusion, probably because they consider only low degree or sparse polynomials, and their method may be different than ours (*e.g.* they don't use the de Casteljau algorithm). In all cases, the Bernstein method gives a nearly optimal subdivision.

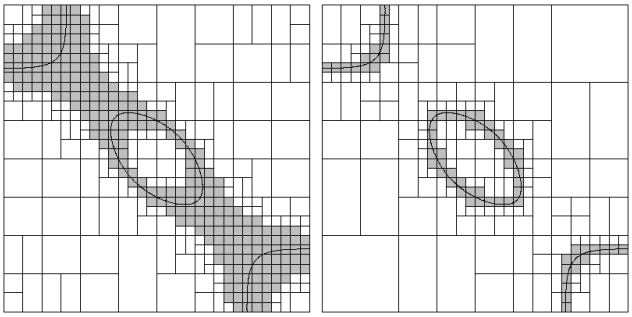

Figure 3: Figueiredo *et al.*'s comparison by between naive IA (left) and affine arithmetic (right) with curve $f(x, y) = x^2 + y^2 + xy - (xy)^2/2 - 1/4$, $(x, y) \in [-2, 2] \times [-2, 2]$.

Figueiredo *et al.* [2,4] results appear to be similar to the Bernstein based method with low degree polynomials (see Figure 3). Though, we lack of material to make any further comparison with other arithmetics.

The Bernstein based method has limitations:

- It is a dense representation. A function $f(x_1, x_2 \dots x_n)$ with degree d_i in x_i is represented by a grid of $D = (d_1 + 1)(d_2 + 1) \dots (d_n + 1)$ coefficients, even when f is sparse in the canonical base. If $n = 10$ and $d_i = 2$, then $N = 3^{10}$ coefficients are needed. Maybe a solution is to use the simplicial Bézier form [3, 13] instead of the tensorial product form, but we do not yet investigate this track.
- For high degree (20 or more), the condition number of the conversion matrix between the canonical and the Bernstein base becomes bad.
- It does not apply for transcendent functions like $\cos, \exp$. It is possible to bound these functions between two polynomials: this method is used in computer arithmetic for computing transcendent functions, another possibility relies on the Poisson's base [10].

5 Conclusion

On this paper we show that we could experiment better results using Bernstein based arithmetic with the de Casteljau algorithm than using naive interval arithmetic, Taubin's method and the Bernstein method. It would be very interesting to compare this method with affine arithmetic since Figueiredo *et al.* [4] seem to obtain quite similar results. Our interest would be to focus on high degree polynomials to complete our results.

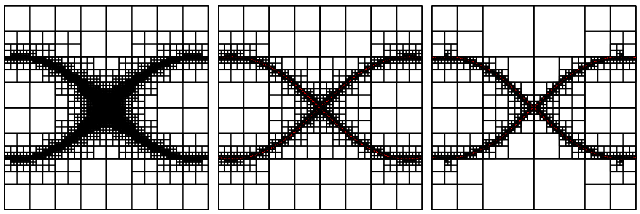

Figure 4: Left: naive IA. Middle: Taubin. Right: de Casteljau. Plots of Cassini's oval: $C_{2,2}(x,y) = 0$ in $[-2,2] \times [-2,2]$, where $C_{a,b}(x,y) = ((x+a)^2 + y^2) \times ((x-)^2 + y^2) - b^4$.

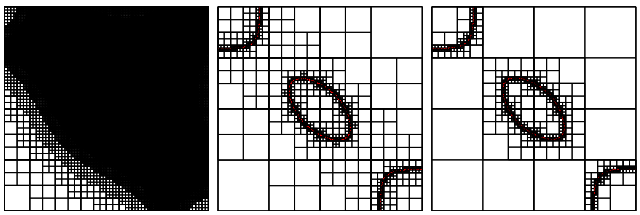

Figure 5: Left: naive IA. Middle: Taubin. Right: de Casteljau. The equation is: $f(x,y) = 15/4 + 8x - 16x^2 + 8y - 112xy + 128x^2y - 16y^2 + 128xy^2 - 128x^2y^2 = 0$ on the square $[0,1] \times [0,1]$. The same curve is drawn in Martin *et al.* [7]. The image we obtain with the de Casteljau method is different from the one obtained in [7]: the latter is polluted with isolated short segments.

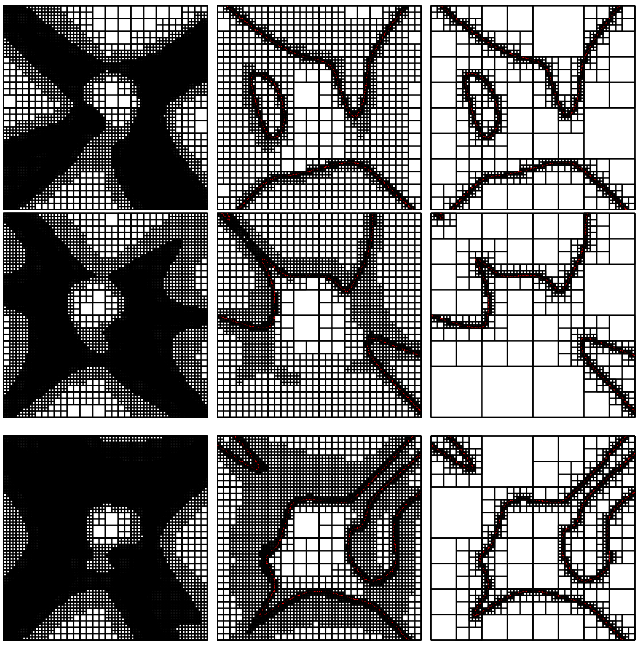

Figure 6: Left: naive IA. Middle: Taubin. Right: de Casteljau. Random curves with total degree 10, 14, 18.

References

[1] A. Andrade, D. Comba, J. Luiz, J. Stolfi, and M. Vinicius. Affine arithmetic, Dec. 06 1993.

[2] L. H. de Figueiredo and J. Stolfi. Affine arithmetic: Concepts and applications. *Numerical Algorithms*, 37(1–4):147–158, Dec. 2004.

[3] G. Farin. *Curves and Surfaces for Computer Aided Geometric Design*. Academic Press, Boston, 3 edition, 1993.

[4] L. Figueiredo and J. Solfi. Adaptive enumeration of implicit surfaces with affine arithmetic. In *Implicit Surfaces'95*, pages 161–170, Grenoble, France, Apr. 1995.

[5] C.-Y. Hu, N. Patrikalakis, and X. Ye. Robust Interval Solid Modelling. Part 1: Representations. Part 2: Boundary Evaluation. *CAD*, 28(10):807–817, 819–830, 1996.

[6] R. Kearfott. Interval computations: Introduction, uses, and resources. *Euromath Bulletin*, 1996. Also available at *http://interval.usl.edu/preprints.html*.

[7] R. Martin, H. Shou, I. Voiculescu, A. Bowyer, and G. Wang. Comparison of interval methods for plotting algebraic curves. *Comput. Aided Geom. Des.*, 19(7):553–587, 2002.

[8] R. Moore. *Interval Analysis*. Prentice Hall, Englewood Cliffs, N.J., 1966.

[9] R. E. Moore. The automatic analysis and control of error in digital computation based on the use of interval numbers. In L. B. Rall, editor, *Error in Digital Computation, Vol. I*, pages 61–130. Wiley, New York, 1965.

[10] G. Morin and R. Goldman. Trimming analytic functions using right sided Poisson subdivision. *Computer-Aided Design*, 33(11):813–824, 2001.

[11] A. Neumaier. The wrapping effect, ellipsoid arithmetic, stability and confidence regions. *Computing Supplementum*, 9:175–190, 1993.

[12] N. M. Patrikalakis and N. M. Patrikalakis. *Shape Interrogation for Computer Aided Design and Manufacturing*. Springer-Verlag New York, Inc., 2002.

[13] J. Peters. Evaluation and approximate evaluation of the multivariate bernstein-Bézier form on a regularly partitioned simplex. *ACM Trans. Math. Softw.*, 20(4):460–480, 1994.

[14] E. C. Sherbrooke and N. Patrikalakis. Computation of the solutions of nonlinear polynomial systems. *Computer Aided Geometric Design*, 10(5):379–405, 1993.

[15] G. Taubin. Rasterizing algebraic curves and surfaces. *IEEE Computer Graphics and Applications*, 14(2):14–23, mar 1994.

[16] G. Taubin. An Accurate Algorithm for Rasterizing Algebraic Curves. In *Second Symp. on Solid Modeling and Applications, ACM/IEEE*, pages 221–230, May 1993.