Comparison of Interval Methods for Plotting Algebraic Curves

Ralph Martin[1], Huahao Shou[2,3], Irina Voiculescu[4], Adrian Bowyer[5], Guojin Wang[2]

1 Department of Computer Science, Cardiff University, Cardiff, UK.
e-mail: Ralph.Martin@cs.cf.ac.uk

2 Department of Mathematics, Zhejiang University, Hangzhou, China.

3 Department of Applied Mathematics, Zhejiang University of Technology, Hangzhou, China.

4 Computing Laboratory, Oxford University, Oxford, UK.

5 Department of Mechanical Engineering, University of Bath, Bath, UK.

Abstract

This paper compares the performance and efficiency of different function range interval methods for plotting $f(x, y) = 0$ on a rectangular region based on a subdivision scheme, where $f(x, y)$ is a polynomial. The solution of this problem has many applications in CAGD. The methods considered are interval arithmetic methods (using the power basis, Bernstein basis, Horner form and centred form), affine arithmetic method, Bernstein coefficient method, Taubin's method, Rivlin's method, Gopalsamy's method, and related methods which also take into account derivative information. Our experimental results show that the affine arithmetic method, interval arithmetic using centred form method, the Bernstein coefficient method, Taubin's method, Rivlin's method, and their related derivative methods have similar performance, and generally they are more accurate and efficient than Gopalsamy's method and interval arithmetic using the power basis, the Bernstein basis, and Horner form methods.

Keywords: Subdivision, Interval Analysis, Range Analysis, Algebraic Curves

1 Introduction

Implicit curves are extremely useful in geometric modelling, especially in CSG, and also for trimming operations on parametrically described shapes. They can represent, for example, the intersection of two parametric surfaces in $\mathbf{R}^3$, or the silhouette edges of a parametric surface in $\mathbf{R}^3$ with respect to a given view [28].

Tracing the implicit curve $f(x, y) = 0$ in a rectangular region $[\underline{x}, \overline{x}] \times [\underline{y}, \overline{y}]$, where f is a polynomial in two variables, is of great interest in CAD, CAGD and computer graphics. We consider here as an example the problem of drawing the curve on a rectangular grid of pixels, but the same methods can be used at a higher resolution for many other applications.

A straightforward solution to the implicit curve plotting problem is to exhaustively test whether the curve passes through each pixel. Such a test can be performed by evaluating the approximate Euclidean distance from the center of each pixel to the curve [31] or by point sampling [14]. Clearly such methods are not efficient.

Continuation methods [7] are usually efficient because they use one or more seed pixels on a curve and then trace the curve continuously. However these methods have one fundamental difficulty, that of finding a complete set of initial seed pixels.

Subdivision methods [9, 28, 29, 30, 31, 32] start with the plot area itself as an initial cell. If a cell is proved to be empty, it is ignored; otherwise, it is subdivided into smaller cells, which are then visited recursively, until the cells reach pixel size (or a desired accuracy). In this way large portions of the plot area can be discarded quickly and reliably at an early stage, again leading to efficient methods.

Range analysis [23] provides a general test procedure for reliably rejecting certain cells at each subdivision step. In range analysis, a conservative interval is computed for the range of function values within a cell. If this interval does not contain zero, then the curve does not intersect the cell. However, if the interval does contains zero, we cannot conclude that the cell intersects the curve because the function range interval is not required to be exact. Therefore the cell must be subdivided for further investigation. In the approach we take in this paper, once the cells reach pixel size, we stop further analysis (although in principle, a pixel may be further subdivided into subpixels), and just plot the cell as if the curve passed through it. This may result in a "fat" curve, as some plotted pixels do not actually contain the curve—the curve is fatter than it should be. This is clearly unacceptable for certain applications, which may prefer to approximate the curve in some other way in pixel-sized cells.

The classical technique of interval arithmetic (IA) [21, 22] provides a natural tool for range analysis [23]; an overview is given in Section 3.1. Subdivision methods based on IA have been proposed for rasterization of implicit curves and surfaces in computer graphics applications [9, 28, 30]. IA also has been used in computer graphics applications such as fast ray tracing and robust solid modelling [1, 18, 19].

Because of the way arithmetic operators work in IA (for example, the distributive law no longer holds), the form used to express the polynomial $f(x, y)$ affects the range for the function output by an IA evaluation. A previous paper [35] showed that IA using the Bernstein basis is generally more accurate than IA using the power basis. In the current paper two other polynomial forms—Horner form and centred form are added for comparison, in addition to other approaches we also consider.

The main weakness of IA is that it tends to be too conservative [8, 13, 14], i.e. the range output for the function by IA is sometimes *much* wider than the actual range of values the function takes over a given interval. To solve this problem, Comba and Stolfi [8] proposed a new model for numerical computation, called affine arithmetic (AA); an overview is given in Section 3.2. AA has been used as a replacement for IA in various computer graphics applications, such as ray tracing, intersection testing, enumeration of implicit curves and surfaces, and sampling for procedural shaders [8, 13, 14, 17]. As AA usually computes tighter intervals than IA, it is possible to draw algebraic curves using AA more efficiently and with higher quality than using IA [14, 36]. However AA is still too conservative sometimes, and it does not obey the distributive law either. To solve these problems, in this paper we use a modified Matrix AA (MAA) method proposed in [27].

Another well known method for range analysis is the Bernstein coefficient (BC) method based on the Bernstein convex hull property [10]. This method relies on the simple idea that if a polynomial is written in the Bernstein basis, the range of the polynomial is bounded by the values of the minimum and maximum Bernstein coefficients. A modification of this method elevates the degree of the Bernstein polynomials before using the coefficients to find the range. It is known that as the degree is elevated, the bounds become tighter, but an additional computational cost is involved [10].

Based on a simple polynomial inequality, Taubin [32] introduced a test that is a sufficient condition for a polynomial in two variables to not have roots inside a box. His approach is a particularly efficient way to construct inclusion functions for polynomials.

A further method for bounding the range of a polynomial over an interval for the univariate case by Cargo [6] and Rivlin [25] is based on a simple estimate of the second derivative in a Taylor expansion of the polynomial. Garloff [15] extended the idea to the bivariate case. The bounds are found from the values of the polynomial at points of a regular grid subdividing the unit square.

Gopalsamy et al. [16] proposed a method of evaluating compact geometric bounds for both univariate and bivariate polynomials by simply sampling the polynomial. The optimal sampling positions depend only on the degree of the polynomial.

In addition to the ideas above, we notice that derivative information can help to make the determination of bounds more precise and faster. The basic idea is that if the derivative has a single sign over an interval, then the function is bounded by its value at the ends of the interval.

By computing bounds on the derivative in the same way as on the function itself, this idea may be applied recursively. Each of the above mentioned methods thus has a companion family of derivative versions.

In summary, many approaches exist in the literature for conservatively solving the problem of whether the curve passes through a given cell or not. In the rest of this paper we compare these approaches in terms of arithmetic operations involved and localisation of the result, and also suggest some modifications to these approaches and new approaches of our own.

2 Implicit curve drawing algorithm

The basic recursive strategy as presented in [28, 29, 31, 36] for drawing an implicit curve $f(x, y) = 0$ in a given rectangular interval $[\underline{x}, \overline{x}] \times [\underline{y}, \overline{y}]$ is to evaluate $f(x, y)$ over the desired interval using some range analysis evaluation method (such as IA, AA or BC) giving a range $\mathbf{F} = [\underline{F}, \overline{F}]$. If the resulting interval does not contain 0, the curve cannot be present. If it does contain 0, we subdivide the interval horizontally and vertically at its mid point, and consider the pieces in turn. The process stops when an interval consisting of a single pixel is left. In such a case we fill the pixel. This may result in a "fat" curve if the test is too conservative, i.e. pixels may be filled which do not actually contain the curve. In detail, we use the following procedure:

```
PROCEDURE    Quadtree($\underline{x}, \overline{x}, \underline{y}, \overline{y}$):
    $\mathbf{F}$=RangeEvaluation($\underline{x}, \overline{x}, \underline{y}, \overline{y}$);
    if $\underline{F} \leq 0 \leq \overline{F}$ then
        if $\overline{x} - \underline{x}$ <PixelSize AND $\overline{y} - \underline{y}$ <PixelSize then
            PlotPixel($\underline{x}, \overline{x}, \underline{y}, \overline{y}$)
        else Subdivide($\underline{x}, \overline{x}, \underline{y}, \overline{y}$).

PROCEDURE    Subdivide($\underline{x}, \overline{x}, \underline{y}, \overline{y}$):
    $\check{x}$=$(\underline{x} + \overline{x})/2$;
    $\check{y}$=$(\underline{y} + \overline{y})/2$;
    Quadtree($\underline{x}, \check{x}, \underline{y}, \check{y}$);
    Quadtree($\underline{x}, \check{x}, \check{y}, \overline{y}$);
    Quadtree($\check{x}, \overline{x}, \check{y}, \overline{y}$);
    Quadtree($\check{x}, \overline{x}, \underline{y}, \check{y}$).
```

Here $\mathbf{F}$=RangeEvaluation($\underline{x}, \overline{x}, \underline{y}, \overline{y}$) is the conservative interval containing all values of $f(x, y)$ over $[\underline{x}, \overline{x}] \times [\underline{y}, \overline{y}]$, computed using a chosen range analysis method such as IA or AA; ($\check{x}$,$\check{y}$) is the mid-point of $[\underline{x}, \overline{x}] \times [\underline{y}, \overline{y}]$.

As far as IA is concerned, the natural interval extension $\mathbf{f}$ depends on the given algorithmic expression of f; different expressions will lead to different interval results. For example, a polynomial remains the same whether it is expressed in the power basis or in the Bernstein basis, but the natural interval extensions in the two cases are different. Some methods like IA on centred form, the Bernstein coefficient method, Rivlin's method and Gopalsamy's method need to perform further work to also update the algorithmic form of $f(x, y)$ as well as $[\underline{x}, \overline{x}] \times [\underline{y}, \overline{y}]$ during each subdivision.

We should point out that the actual bounds $[\underline{F}, \overline{F}]$ of f in a cell are unnecessary for the classification of the cell. It suffices to test any condition equivalent to $\underline{F} \leq 0 \leq \overline{F}$. This may be used to optimize some procedures; for example, given the coefficients of f in the Bernstein basis. the test may be terminated after finding any two coefficients of opposite sign.

Two techniques can be used to improve the graphical quality, for the curve drawing problem. One is to use subpixel techniques: the subdivision can go down to subpixels and combine the results on the way back up the recursion, which can help to further remove some pixels which do not actually intersect the curve. The other is to use a sign testing (also known as point sampling) technique, which can greatly reduce the "fatness" of the approximation. The algorithm given can be easily adapted to receive the values of f at the corners of the box and pass them along, computing f exactly once at each corner. The overhead of sign testing is thus quite low. Using sign testing, we can compute a drawing that has pixels of 3 colors: white, when the curve does not cross the pixel, black when the curve definitely crosses it, as shown by sign testing, and gray, when sign testing fails to give information, but 0 is still in the computed interval. These two techniques significantly improve the results produced by the poorer methods described later in the paper, but for the better methods these two techniques do not help much, and only increase the number of calculations. For this reason, we do not use consider these techniques further in this paper.

3 Description of the methods

There are many methods for conservatively solving $f(x,y)=0$ on $[\underline{x},\overline{x}]\times[\underline{y},\overline{y}]$ by means of range analysis. Some methods just need function values, some also need derivatives, and some are based on a change of basis or rewriting f in a particular form. We consider here IA methods, an AA method, Bernstein coefficient methods, Taubin's method, Rivlin's method, Gopalsamy's method, and corresponding derivative versions.

In this section we first review IA, then explain the polynomial power form, Horner form, Bernstein form and centred form, all using IA, in detail. After that we review AA methods, followed by descriptions of the Bernstein coefficient methods, Taubin's method, Rivlin's method and Gopalsamy's method. Finally we describe their derivative versions.

3.1 Interval arithmetic methods

Here we outline traditional IA methods, which are widely used in scientific computation. Interval arithmetic (or interval analysis) is a technique for numerical computation where each uncertain quantity is represented by an interval of floating-point numbers. These intervals are added, subtracted, multiplied, etc. in such a way that each computed interval is *guaranteed* to contain the unknown value of the quantity it represents [21, 22].

An interval $\mathbf{x}=[a,b]$, $a\le b$ is a set of real numbers defined by $[a,b]=\{x \mid a\le x\le b\}$. If $\mathbf{x}$ and $\mathbf{y}$ are intervals and $\odot$ denotes one of the arithmetic operators $+,-,\times$ and $/$, then $\mathbf{x}\odot\mathbf{y}$ is defined by

$$\mathbf{x}\odot\mathbf{y}=\{x\odot y \mid x\in\mathbf{x}, y\in\mathbf{y}\}.$$

Any real number a is considered to be an interval $a=[a,a]$, which means that expressions such as $a\mathbf{x}$, $a+\mathbf{x}$, $\mathbf{x}/a$, and $(-1)\mathbf{x}=-\mathbf{x}$ are well defined. Moore [21] proved that the above definition is equivalent to the following set of constructive rules:

$$\begin{aligned}
[a,b]+[c,d] &= [a+c,b+d]\,,\\
[a,b]-[c,d] &= [a-d,b-c]\,,\\
[a,b]\times[c,d] &= [\min(ac,ad,bc,bd),\max(ac,ad,bc,bd)]\,,\\
[a,b]\,/\,[c,d] &= [a,b]\times[1/d,1/c] \quad \text{provided } 0\notin[c,d]\,.
\end{aligned}$$

A treatment of interval division for intervals containing 0 can be found in [20].

The natural interval extension of a bivariate polynomial $f(x,y)$, denoted by $\mathbf{f}(\mathbf{x},\mathbf{y})$, is obtained by replacing each occurrence of x and y in $f(x,y)$ by intervals $\mathbf{x}$ and $\mathbf{y}$, and evaluating the resulting interval expression using the above definitions. The result is itself an interval. As already noted, interval extensions depend on the specific order of evaluation of the intermediate results. In the case of a polynomial, the usual order is that first one computes powers of x and y, multiplies them together and then multiplies the result by the corresponding coefficient, and the resulting monomials are added. While the rules for $+,-,\times,/$ are exact, more generally, the resulting interval is too large. To see this, consider $\mathbf{x}=[-1,2]$. Computing $\mathbf{x}\times\mathbf{x}$ gives $[-2,4]$, but the exact range for $\mathbf{x}^2$ is $[0,4]$. This happens here because the two quantities being multiplied are not independent.

Note that even the computation of powers can be done in several ways. In this paper we use the exact interval result for powers.

The primary motivation for using interval analysis in almost all applications is that the interval extension of a function provides bounds for the variation of the function [21]. This comes from the fundamental property of interval arithmetic: $x \in \mathbf{x} \Rightarrow f(x) \in \mathbf{f}(\mathbf{x})$.

In the current application, intervals are used to represent (large) *regions of interest* in curves and surfaces [26, 33]. Other types of application use intervals to represent (small) *errors* or *uncertainty*, using intervals for the *coefficients* of the polynomials in a suitable basis, rather than the variables.

A significant property of IA already noted is that the form in which the polynomial is expressed can affect the result [5]. For example, clearly $f(u) = 1 + 2u - u^2 = 1 + u(2 - u)$, giving the power form of the polynomial, and the Horner form respectively. Supposing $\mathbf{u} = [0, 1]$, using the power form to evaluate $\mathbf{f}(\mathbf{u})$ gives $[0, 3]$ as the answer, while the Horner form gives $[1, 3]$. Both answers are correct, in the sense that the interval obtained is guaranteed to contain the actual range of the function (but neither is exact: the exact range is $[1, 2]$). Rearranging the function can give tighter bounds on the result, as does the Horner form in this case. We thus now consider several different ways of expressing polynomials for evaluation in IA.

3.1.1 IA using the power basis

Here we describe how to use IA to evaluate a polynomial in two variables written in the power basis, i.e. in which the terms are of the form $a_{ij}x^iy^j$. Let

$$f(x,y) = \sum_{i=0}^{n}\sum_{j=0}^{m} a_{ij}x^iy^j, \quad (x,y) \in \Omega = [\underline{x}, \overline{x}] \times [\underline{y}, \overline{y}]$$

be a polynomial of two variables in power form. It is helpful to rewrite $f(x, y)$ in matrix representation:

$$f(x,y) = XAY,$$

where

$$X = (1, x, ..., x^n), \quad Y = (1, y, \cdots, y^m)^T, \quad A_{ij} = a_{ij}.$$

Example 1. This example is from [8] (but differs by an affine change of coordinates):

$$f(x,y) = 15/4 + 8x - 16x^2 + 8y - 112xy + 128x^2y - 16y^2 + 128xy^2 - 128x^2y^2,$$

and $(x, y) \in [0, 1] \times [0, 1]$. $f(x, y)$ can be rewritten as $f(x, y) = P(x, y) = XAY$, where

$$X = (1, x, x^2), \quad Y = (1, y, y^2)^T, \quad A = \begin{bmatrix} 15/4 & 8 & -16 \\ 8 & -112 & 128 \\ -16 & 128 & -128 \end{bmatrix}.$$

To evaluate $f(x, y)$, X and Y are first computed using special IA rules for powers, and then the matrix product is found.

3.1.2 IA using Horner form

The Horner form of a polynomial involves nested brackets. Successively higher powers are computed working outwards from the innermost bracket. E.g., in the univariate case, the Horner form of the polynomial function $x^3 + 2x^2 + 3x + 4$ is $x(x(x + 2) + 3) + 4$. Several versions of Horner form exist in the bivariate case: one can nest in x first then y, or nest in y first then x, or one can nest alternatively x, y, x, y, etc.

Two such Horner forms for Example 1 are:

$$\begin{aligned} f(x,y) = h_x(x,y) &= 15/4 + (8 - 16y)y + (8 + (-112 + 128y)y + (-16 + (128 - 128y)y)x)x \\ = h_y(x,y) &= 15/4 + (8 - 16x)x + (8 + (-112 + 128x)x + (-16 + (128 - 128x)x)y)y. \end{aligned}$$

In this paper we only consider x first then y and y first then x Horner forms. Usually they produce different graphical results, unless the function $f(x, y)$ is symmetric, as is the one in Example 1.

3.1.3 IA using the Bernstein basis

Bernstein polynomials are widely used for generating Bézier, B-spline and NURBS curves and surfaces [10]. The Bernstein basis $B_j^i(u) = \binom{i}{j} u^j (1-u)^{i-j}, \quad j = 0, 1, \cdots, i$ has been shown to be numerically more stable and better conditioned for finding roots than the power basis [11, 12].

Bowyer et al. [2, 3, 4, 5, 35] have extensively considered IA applied to multivariate Bernstein-form polynomials.

Conversion between the power basis and Bernstein basis for multivariate polynomials is discussed in [2, 3]. We just give an example here. The Bernstein form for Example 1 is:

$$f(x,y) = b(x,y) = (\frac{15}{4}(1-x)^2 + \frac{31}{2}(1-x)x - \frac{17}{4}x^2)(1-y)^2 + 2(\frac{31}{4}(1-x)^2 - \frac{65}{2}(1-x)x$$

$$+\frac{31}{4}x^2)(1-y)y + (-\frac{17}{4}(1-x)^2 + \frac{31}{2}(1-x)x + \frac{15}{4}x^2)y^2, \quad \text{where} \quad (x,y) \in [0,1] \times [0,1].$$

As seen in this example, the Bernstein form is usually more complicated (i.e. less sparse) and may have many terms. Furthermore it contains repeated subexpressions of $x, (1-x), y$ and $(1-y)$. Knowing that repeated expressions can lead to excessive conservativeness in IA, one might doubt the desirability of using the Bernstein basis with IA. However, practical results show that surprisingly the Bernstein form not only does well, but it usually does better than the simpler power basis.

Two approaches may be taken to conversion to Bernstein basis. One is to perform the conversion once only, at the start of the process. The other is to reconvert the polynomial to a new local Bernstein form every time subdivision is done, in an attempt to localize the curve further. However, the latter approach is completely unsuccessful, as the resulting interval on evaluating the function contains zero in *every* case, as can easily be seen. Consider the univariate case. After changing the coordinates to new local ones for the interval under consideration, $x = [0,1]$, so $1-x$ is also $[0,1]$, and $x^i(1-x)^j = [0,1]$. Thus, $a_{ij}x^i(1-x)^j$ contains 0, and a sum of such terms needed to evaluate the function also contains 0. Because of this, we only consider *initial* conversion to the Bernstein basis in the rest of the paper.

3.1.4 IA using centred form

The centred form was introduced by Moore in [21]. It has been shown to be an effective tool for computing function ranges using IA [23].

The centred form of a bivariate polynomial $f(x,y)$ on $[\underline{x}, \overline{x}] \times [\underline{y}, \overline{y}]$ can be obtained by translating the coordinate origin to the centre of the rectangle $[\underline{x}, \overline{x}] \times [\underline{y}, \overline{y}]$, using a variable transformation $x = \tilde{x} + (\underline{x} + \overline{x})/2, y = \tilde{y} + (\underline{y} + \overline{y})/2$, where $\tilde{x}$ and $\tilde{y}$ are new variables.

Unlike when using the power basis, Horner form or Bernstein basis methods with IA, when using the centred form, each time subdivision is performed, the centred form of the function must be updated. Although this requires some extra work, it helps to restrict the computed range of the function on each rectangle, and overall this approach gives very good results, as we show in Section 4.2.

3.2 Affine arithmetic method

Affine arithmetic (AA) [8] is an alternative approach to IA that can be more resistant to over-conservatism due to its ability of keeping track of correlations between computed and input quantities.

In affine arithmetic an uncertain quantity x (such as an interval) is represented by an affine form $\hat{x}$ that is a linear expression in a set of noise symbols ε_i:

$$\hat{x} = x_0 + x_1\varepsilon_1 + \cdots + x_m\varepsilon_m = x_0 + \sum_{i=1}^{m} x_i\varepsilon_i.$$

Here the values of the noise symbols ε_i are unknown but are assumed to be in the range $[-1, 1]$. The corresponding coefficient x_i is a real number that determines the magnitude and sign of ε_i. Each ε_i stands for an independent source of error or uncertainty which contributes to the total uncertainty in the quantity x. One may make the number m as large as necessary in order to represent all the sources of uncertainty. These may be input data uncertainty, formula truncation errors, arithmetic rounding errors, and so on. If the same noise symbol ε_i appears in two or more affine forms (e.g. in both $\hat{x}$ and $\hat{y}$), it indicates that dependencies and correlations exist between the underlying quantities x and y.

Conversions between affine forms and intervals are defined in [8]: given an ordinary interval $[\underline{x}, \overline{x}]$ representing a quantity x, the corresponding affine form can be written as

$$\hat{x} = x_0 + x_1\varepsilon_x, \qquad \text{where we set} \quad x_0 = (\underline{x} + \overline{x})/2, \quad x_1 = (\overline{x} - \underline{x})/2.$$

Conversely, given an affine form $\hat{x} = x_0 + x_1\varepsilon_1 + \cdots + x_m\varepsilon_m$, the corresponding interval is

$$[\underline{x}, \overline{x}] = [x_0 - \xi, x_0 + \xi], \qquad \text{where} \quad \xi = \sum_{i=1}^{m} |x_i|.$$

Given two affine forms

$$\hat{x} = x_0 + x_1\varepsilon_1 + \cdots + x_n\varepsilon_n, \quad \hat{y} = y_0 + y_1\varepsilon_1 + \cdots + y_n\varepsilon_n,$$

some simple operations are defined in [8] as below:

$$\begin{aligned} \hat{x} \pm \hat{y} &= (x_0 \pm y_0) + (x_1 \pm y_1)\varepsilon_1 + \cdots + (x_n \pm y_n)\varepsilon_n, \\ \alpha \pm \hat{x} &= (\alpha \pm x_0) + x_1\varepsilon_1 + \cdots + x_n\varepsilon_n, \\ \alpha\hat{x} &= (\alpha x_0) + (\alpha x_1)\varepsilon_1 + \cdots + (\alpha x_n)\varepsilon_n. \end{aligned}$$

From the above equations, it is clear that, if $\hat{x} = [-1, 1] = 0 + 1 \cdot \varepsilon_1, \hat{y} = [-1, 1] = 0 + 1 \cdot \varepsilon_2$, in AA, $\hat{x} - \hat{x} = 0$ and $(2\hat{x} + \hat{y}) - \hat{x} = \hat{x} + \hat{y} = 0 + \varepsilon_1 + \varepsilon_2 = [-2, 2]$, whereas in IA, the results computed are $[-2, 2]$ and $[-4, 4]$, respectively.

Multiplication of two affine forms $\hat{x} \times \hat{y}$ produces a quadratic polynomial in the noise symbols ε_i:

$$\hat{x} \times \hat{y} = (x_0 + \sum_{i=1}^{n} x_i\varepsilon_i) \times (y_0 + \sum_{i=1}^{n} y_i\varepsilon_i).$$

Comba and Stolfi [8] show how to reduce the result to a new affine form. By expanding, we get

$$\hat{x} \times \hat{y} = x_0y_0 + \sum_{i=1}^{n}(x_0y_i + y_0x_i)\varepsilon_i + (\sum_{i=1}^{n} x_i\varepsilon_i) \times (\sum_{i=1}^{n} y_i\varepsilon_i).$$

To produce a new linear expression, the last term, which is quadratic in the ε_i, is replaced by another new noise symbol ε_k with coefficient uv, where

$$u = \sum_{i=1}^{n} |x_i|, \quad v = \sum_{i=1}^{n} |y_i|.$$

Thus $\hat{x} \times \hat{y}$ can be expressed as an affine combination of first-degree polynomials on ε_i plus a new noise symbol ε_k whose value is still between $[-1, 1]$:

$$\hat{x} \times \hat{y} = x_0y_0 + (x_0y_1 + x_1y_0)\varepsilon_1 + \cdots + (x_0y_n + x_ny_0)\varepsilon_n + uv\varepsilon_k.$$

However AA still has a over-conservatism problem. For example, let $\hat{x} = 0 + \varepsilon_1 + \varepsilon_2, \hat{y} = 0 + \varepsilon_1 - \varepsilon_2$. The exact range of $\hat{x} \times \hat{y}$ is $\varepsilon_1^2 - \varepsilon_2^2 = [0, 1] - [0, 1] = [-1, 1]$, while using AA gives $[-4, 4]$. Besides, AA does not obey the distributive law. For example, in AA, $\hat{x} \times (\hat{y} - \hat{y})$ is zero , but $\hat{x} \times \hat{y} - \hat{x} \times \hat{y}$ is not zero. To avoid these problems, in this paper, we use the modified Matrix AA polynomial evaluation method (MAA) we proposed in [27], reproduced below.

First we convert the interval forms $[\underline{x}, \overline{x}]$ and $[\underline{y}, \overline{y}]$ to affine forms $\hat{x} = x_0 + x_1\varepsilon_x, \quad \hat{y} = y_0 + y_1\varepsilon_y$, as explained earlier. Then let

$$\hat{X} = (1, \varepsilon_x, \cdots, \varepsilon_x^n), \quad \hat{Y} = (1, \varepsilon_y, \cdots, \varepsilon_y^m)^T.$$

Let

$$B = \begin{bmatrix} 1 & x_0 & \cdots & x_0^{n-1} & x_0^n \\ 0 & x_1 & \cdots & (n-1)x_0^{n-2}x_1 & nx_0^{n-1}x_1 \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & \cdots & x_1^{n-1} & nx_0x_1^{n-1} \\ 0 & 0 & \cdots & 0 & x_1^n \end{bmatrix};$$

in detail

$$B_{ij} = \begin{cases} \binom{j}{i} x_0^{j-i} x_1^i, & i \le j \\ 0, & i > j \end{cases}, \quad i = 0, 1, \cdots, n; \quad j = 0, 1, \cdots, n.$$

Also, let

$$C = \begin{bmatrix} 1 & 0 & \cdots & 0 & 0 \\ y_0 & y_1 & \cdots & 0 & 0 \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ y_0^{m-1} & (m-1)y_0^{m-2}y_1 & \cdots & y_1^{m-1} & 0 \\ y_0^m & my_0^{m-1}y_1 & \cdots & my_0y_1^{m-1} & y_1^m \end{bmatrix};$$

in detail

$$C_{ij} = \begin{cases} 0, & i < j \\ \binom{i}{j} y_0^{i-j} y_1^j, & i \ge j \end{cases}, \quad i = 0, 1, \cdots, m; \quad j = 0, 1, \cdots, m.$$

Now, if we compute D from matrices B and C, and the original coefficient matrix A, as follows

$$D = BAC,$$

we obtain

$$f(\hat{x}, \hat{y}) = \hat{X} D \hat{Y} = \sum_{i=0}^{n} \sum_{j=0}^{m} D_{ij} \varepsilon_x^i \varepsilon_y^j.$$

Up to now the calculation is exact; in the next step we convert this result back to interval form $[\underline{F}, \overline{F}]$, as follows. If i is even and j is even, then $\varepsilon_x^i \varepsilon_y^j \in [0, 1]$, otherwise $\varepsilon_x^i \varepsilon_y^j \in [-1, 1]$. Thus,

$$\overline{F} = D_{00} + \sum_{j=1}^{m} \left\{ \begin{array}{cc} \max(0, D_{0j}), & \text{if } j \text{ is even} \\ |D_{0j}|, & \text{otherwise} \end{array} \right\} + \sum_{i=1}^{n} \left\{ \begin{array}{cc} \max(0, D_{i0}), & \text{if } i \text{ is even} \\ |D_{i0}|, & \text{otherwise} \end{array} \right\}$$
$$+ \sum_{i=1}^{n} \sum_{j=1}^{m} \left\{ \begin{array}{cc} \max(0, D_{ij}), & \text{if } i, j \text{ are both even} \\ |D_{ij}|, & \text{otherwise} \end{array} \right\},$$

and

$$\underline{F} = D_{00} + \sum_{j=1}^{m} \left\{ \begin{array}{cc} \min(0, D_{0j}), & \text{if } j \text{ is even} \\ -|D_{0j}|, & \text{otherwise} \end{array} \right\} + \sum_{i=1}^{n} \left\{ \begin{array}{cc} \min(0, D_{i0}), & \text{if } i \text{ is even} \\ -|D_{i0}|, & \text{otherwise} \end{array} \right\}$$
$$+ \sum_{i=1}^{n} \sum_{j=1}^{m} \left\{ \begin{array}{cc} \min(0, D_{ij}), & \text{if } i, j \text{ are both even} \\ -|D_{ij}|, & \text{otherwise} \end{array} \right\}.$$

This gives tighter bounds on $f(x, y)$ over the range $[\underline{x}, \overline{x}] \times [\underline{y}, \overline{y}]$ than straightforward AA.

Unlike IA and AA, which do not obey the distributive law, MAA satisfies all the commutative, associative and distributive laws because it keeps all powers of noise symbols without approximation. In this respect, there is no difference between MAA and real arithmetic. Because of this, using different algorithmic expressions for a polynomial function does nothing other than rearranging the terms, and does not affect the result of evaluation of the polynomial in MAA. For example, consider $f(x) = 4x^2 - 12x + 9$ over the interval $\mathbf{x} = [0, 1]$. Its Bernstein form is $b(x) = 9(1 - x)^2 + 6x(1 - x) + x^2$. Its affine form is $\hat{x} = \frac{1}{2}(1 - \varepsilon_1)$. In IA, $f([0, 1]) = [-3, 13]$, while $b([0, 1]) = [0, 16]$: they are different. In MAA, $f(\hat{x}) = 4\hat{x}^2 - 12\hat{x} + 9 = 4 + 4\varepsilon_1 + \varepsilon_1^2$, while $b(\hat{x}) = 9(1 - \hat{x})^2 + 6\hat{x}(1 - \hat{x}) + \hat{x}^2 = 4 + 4\varepsilon_1 + \varepsilon_1^2 = f(\hat{x})$: they are the same. Therefore, when MAA is involved, only the power basis needs to be considered.

The above argument relies on the commutativity, associativity and distributivity of real numbers. More subtly, one may argue that in practice this is not true because of the non-commutativity, associativity and distributivity of computer floating point arithmetic used in MAA operations. However, machine precision is negligible when compared with the widths of the intervals used in solving the curve drawing problem. Although in principle there may be tiny differences in results computed using MAA with different forms or bases on a computer, they are not significant and can be ignored for the purposes of this paper as pixel sizes considered here are much larger than machine precision. (Careful choice of rounding directions should be used to ensure that output intervals are still guaranteed to contain the exact range of the function.)

3.3 Bernstein coefficient methods

Another family of methods for bounding the range of a polynomial over an interval depends on the Bernstein convex hull property [10], which guarantees that the value of a polynomial over the interval $[0, 1]$ is bounded by the values of the minimum and maximum Bernstein coefficients when the polynomial is written in the Bernstein basis.

To utilize this property for the evaluation of $f(x, y)$ over the region $[\underline{x}, \overline{x}] \times [\underline{y}, \overline{y}]$, we must first convert the range $[\underline{x}, \overline{x}] \times [\underline{y}, \overline{y}]$ to $[0, 1] \times [0, 1]$. This can be done by a change of variables:

$$x = \underline{x} + (\overline{x} - \underline{x})\tilde{x}, \quad y = \underline{y} + (\overline{y} - \underline{y})\tilde{y},$$

where $\tilde{x}$ and $\tilde{y}$ are new variables. Then

$$f(x, y) = \tilde{X}(EAR^T)\tilde{Y}^T,$$

where

$$\tilde{X} = (1, \tilde{x}, \cdots, \tilde{x}^n), \quad \tilde{Y} = (1, \tilde{y}, \cdots, \tilde{y}^n),$$

$$E_{ij} = \begin{cases} \binom{j}{i}\underline{x}^{j-i}(\overline{x} - \underline{x})^i, & i \le j \\ 0, & i > j \end{cases}, \quad i = 0, 1, \cdots, n; \quad j = 0, 1, \cdots, n,$$

and

$$R_{ij} = \begin{cases} \binom{j}{i}\underline{y}^{j-i}(\overline{y} - \underline{y})^i, & i \le j \\ 0, & i > j \end{cases}, \quad i = 0, 1, \cdots, m; \quad j = 0, 1, \cdots, m.$$

Let

$$G = EAR^T.$$

Then

$$\tilde{f}(\tilde{x}, \tilde{y}) = \tilde{X}G\tilde{Y}^T, \quad (\tilde{x}, \tilde{y}) \in [0, 1] \times [0, 1],$$

completing the range conversion.

Next we need to convert the above polynomial from the power basis to the Bernstein basis. Let

$$\tilde{B}^n(\tilde{X}) = (B_0^n(\tilde{x}), B_1^n(\tilde{x}), \cdots, B_n^n(\tilde{x})), \quad \tilde{B}^m(\tilde{Y}) = (B_0^m(\tilde{y}), B_1^m(\tilde{y}), \cdots, B_m^m(\tilde{y})),$$

where $B_j^i(u) = \binom{i}{j}u^j(1 - u)^{i-j}$ are the Bernstein basis functions. Then

$$\tilde{B}^n(\tilde{X}) = \tilde{X}H, \quad \tilde{B}^m(\tilde{Y}) = \tilde{Y}P,$$

where

$$H_{ij} = \left\{ \begin{array}{cc} 0, & i < j \\ (-1)^{i-j}\binom{n}{j}\binom{n-j}{i-j}, & i \geq j \end{array} \right. , \quad i = 0, 1, \cdots, n; \quad j = 0, 1, \cdots, n,$$

$$P_{ij} = \left\{ \begin{array}{cc} 0, & i < j \\ (-1)^{i-j}\binom{m}{j}\binom{m-j}{i-j}, & i \geq j \end{array} \right. , \quad i = 0, 1, \cdots, m; \quad j = 0, 1, \cdots, m.$$

Then

$$\tilde{f}(\tilde{x}, \tilde{y}) = \tilde{B}^n(\tilde{X})H^{-1}G(P^T)^{-1}\tilde{B}^m(\tilde{Y})^T,$$

so, letting

$$Q = H^{-1}G(P^T)^{-1},$$

we obtain

$$\tilde{f}(\tilde{x}, \tilde{y}) = \tilde{B}^n(\tilde{X})Q\tilde{B}^m(\tilde{Y})^T, \quad (\tilde{x}, \tilde{y}) \in [0, 1] \times [0, 1].$$

The conversion from the power basis to Bernstein basis is completed.

Let

$$\underline{F} = \min_{i,j}\{Q_{ij}\}, \quad \overline{F} = \max_{i,j}\{Q_{ij}\}, \quad i \in \{0, 1, \cdots, n\}, \quad j \in \{0, 1, \cdots, m\}$$

By the Bernstein convex hull property [10] we know that

$$\underline{F} \leq \tilde{f}(\tilde{x}, \tilde{y}) \leq \overline{F}, \quad (\tilde{x}, \tilde{y}) \in [0, 1] \times [0, 1],$$

and so

$$\underline{F} \leq f(x, y) \leq \overline{F}, \quad (x, y) \in [\underline{x}, \overline{x}] \times [\underline{y}, \overline{y}],$$

giving the desired bounds on $f(x, y)$ over the range $[\underline{x}, \overline{x}] \times [\underline{y}, \overline{y}]$.

To make the bounds on $\tilde{f}(\tilde{x}, \tilde{y})$ tighter over the range $[0, 1] \times [0, 1]$, we can elevate the degree of Bernstein polynomial $\tilde{f}(\tilde{x}, \tilde{y})$ from degree (n, m) to degree $(n+1, m+1)$. The same polynomial written as a higher degree Bernstein polynomial has a tighter Bernstein hull, but there is an additional computational cost, not only due to the conversion itself, but also because the higher degree polynomial generally has more terms. Since

$$\tilde{B}^n(\tilde{X}) = \tilde{B}^{n+1}(\tilde{X})W, \quad \tilde{B}^m(\tilde{Y}) = \tilde{B}^{m+1}(\tilde{Y})V,$$

where

$$W_{ij} = \left\{ \begin{array}{cc} \frac{n-i+1}{n+1}, & i = j \\ \frac{i}{n+1}, & i = j+1 \\ 0, & \text{otherwise} \end{array} \right. , \quad i = 0, 1, \cdots, n+1; \quad j = 0, 1, \cdots, n,$$

$$V_{ij} = \left\{ \begin{array}{cc} \frac{m-i+1}{m+1}, & i = j \\ \frac{i}{m+1}, & i = j+1 \\ 0, & \text{otherwise} \end{array} \right. , \quad i = 0, 1, \cdots, m+1; \quad j = 0, 1, \cdots, m.$$

Then

$$\tilde{f}(\tilde{x}, \tilde{y}) = \tilde{B}^{n+1}(\tilde{X})WQV^T\tilde{B}^{m+1}(\tilde{Y})^T,$$

so, letting

$$S = WQV^T,$$

we obtain

$$\tilde{f}(\tilde{x}, \tilde{y}) = \tilde{B}^{n+1}(\tilde{X})S\tilde{B}^{m+1}(\tilde{Y})^T, \quad (\tilde{x}, \tilde{y}) \in [0, 1] \times [0, 1],$$

and

$$\underline{F} = \min_{i,j}\{S_{ij}\}, \quad \overline{F} = \max_{i,j}\{S_{ij}\}, \quad i \in \{0, 1, \cdots, n+1\}, \quad j \in \{0, 1, \cdots, m+1\}.$$

This process may be repeated as many times as one wishes. Table 1 shows the effect of degree elevation up to five times when drawing the curve $15/4 + 8x - 16x^2 + 8y - 112xy + 128x^2y - 16y^2 + 128xy^2 - 128x^2y^2 = 0$ over $[0, 1] \times [0, 1]$. 'Pixels plotted' shows the number of pixels plotted. The other columns show the number of subdivisions needed and how much arithmetic was required using the corresponding amount of degree elevation. From Table 1 we can see that degree elevation does not help to classify the curve further, and simply increases the amount of operations involved. This is probably because the Bernstein coefficient method without degree elevation already reaches the best possible graphical result at the given resolution. Similar results were observed for other curves. We therefore conclude that degree elevation is not worthwhile, and it is not considered in the rest of this paper.

Degree elevation	Pixels plotted	Subdivisions	Additions	Multiplications
1	522	535	508720	265660
2	522	535	680164	402700
3	522	535	911618	616820
4	522	535	1211654	925148
5	522	535	1588844	1344812

Table 1: Effect of degree elevation when drawing the curve $15/4 + 8x - 16x^2 + 8y - 112xy + 128x^2y - 16y^2 + 128xy^2 - 128x^2y^2 = 0$ using Bernstein coefficient method.

3.4 Taubin's method

Taubin's method [32] can be seen as a specialized form of interval method for polynomials. Although it is not explicitly given in his paper, Taubin actually proved that the bounds of $f(x,y) = \sum_{i=0}^{n}\sum_{j=0}^{m} a_{ij}x^iy^j$ on the box $[-\delta,\delta]\times[-\delta,\delta]$ are

$$\underline{F} = a_{00} - \sum_{h=1}^{n+m} F_h\delta^h, \quad \overline{F} = a_{00} + \sum_{h=1}^{n+m} F_h\delta^h$$

where

$$F_h = \sum_{i+j=h} |a_{ij}|.$$

Since this formula is valid only for the region $[-\delta,\delta]\times[-\delta,\delta]$, to apply this method to a general region $[\underline{x},\overline{x}]\times[\underline{y},\overline{y}]$ we must first translate the coordinate origin to the centre of $[\underline{x},\overline{x}]\times[\underline{y},\overline{y}]$ as in the IAC method and set $\delta = \max((\overline{x}-\underline{x})/2, (\overline{y}-\underline{y})/2)$.

3.5 Rivlin's method

Rivlin [25] proposed a method for computing bounds on a univariate polynomial over the interval $[0,1]$ by evaluating the function at several points in its range. Garloff [15] extended the idea to the bivariate case. For an integer k we define $K = \{(i,j), i = 0,1,\cdots,k; j = 0,1,\cdots,k\}$. The bounds $[\underline{F},\overline{F}]$ involve the function values of the polynomial on a regular grid dividing the unit square $[0,1]\times[0,1]$ at the points $(i/k, j/k), (i,j)\in K$. Garloff showed that

$$\underline{F} = \min_{(i,j)\in K} f(\frac{i}{k},\frac{j}{k}) - \alpha_k, \quad \overline{F} = \max_{(i,j)\in K} f(\frac{i}{k},\frac{j}{k}) + \alpha_k,$$

where

$$\alpha_k = \frac{1}{8k^2}\sum_{i=0}^{n}\sum_{j=0}^{m}(i+j)(i+j-1)|a_{ij}|.$$

Since this formula is valid only for the region $[0,1]\times[0,1]$, to apply this method to a general region $[\underline{x},\overline{x}]\times[\underline{y},\overline{y}]$ we first convert $[\underline{x},\overline{x}]\times[\underline{y},\overline{y}]$ to $[0,1]\times[0,1]$ using the approach described in Section 3.3.

The smallest value of k which may be used is 1. One can choose bigger k in order to increase the accuracy of the bounds, but at the expense of increasing the amount of calculation involved. Table 2 shows the effect of changing k when drawing the curve $15/4 + 8x - 16x^2 + 8y - 112xy + 128x^2y - 16y^2 + 128xy^2 - 128x^2y^2 = 0$ over $[0,1] \times [0,1]$. We can see that increasing k only slightly improves the accuracy of the curve drawn while the number of operations involved steadily increases. We observed similar results for other curves. As a result, we choose the value of k to be 1 wherever we consider Rivlin's method in the rest of the paper.

k	Pixels plotted	Subdivisions	Additions	Multiplications
1	534	585	669849	427489
2	526	541	796379	545808
3	522	539	1038289	753681
4	522	535	1342021	1016180
5	522	535	1721557	1343661

Table 2: Effect of changing k when drawing the curve $15/4 + 8x - 16x^2 + 8y - 112xy + 128x^2y - 16y^2 + 128xy^2 - 128x^2y^2 = 0$ using Rivlin's method.

3.6 Gopalsamy's method

Gopalsamy [16] also proposed a method for computing bounds on an nth degree bivariate polynomial by sampling the polynomial. The optimal sampling positions are calculated once and for all for various degrees of polynomial ($2 \le n \le 9$) using numerical methods, and depend only on the degree of the polynomial.

Gopalsamy's bounds on a degree n polynomial $f(x,y)$ over $[0,1] \times [0,1]$ are calculated as

$$\underline{F} = T - U_g, \quad \overline{F} = T + U_g,$$

where

$$T = \frac{1}{(n+1)^2} \sum_{i=0}^{n} \sum_{j=0}^{n} f(u_i, v_j),$$

and

$$U_g = [\sum_{i=0}^{n} \sum_{j=0}^{n} g^2(u_i, v_j)]^{\frac{1}{2}},$$

where

$$g(u_i, v_j) = f(u_i, v_j) - T.$$

Here u_i and v_i, $0 \le i \le n$, are the optimal parameter values for sampling given in [16] for various values of n ($2 \le n \le 9$).

Unlike any of the previous methods, direct use of Gopalsamy's method requires the computation of square roots, rather than just the four basic arithmetic operations. This may be avoided as follows: having T and U^2 such that $\underline{f} = T - U$ and $\overline{f} = T + U$, instead of testing $\underline{f}\overline{f} \le 0$, the program may test the equivalent inequality $T^2 \le U^2$. However we cannot avoid computing square roots in Gopalsamy's derivative version methods described later, and therefore we do not use this improvement in this paper. In practice the number of square root operations is small, anyhow.

3.7 Derivative versions

Using the derivative of $f(x,y)$ can provide extra information, which can help to make the determination of the bounds for $f(x,y)$ on $[\underline{x}, \overline{x}] \times [\underline{y}, \overline{y}]$ more precise.

The idea is that before evaluating $f(x,y)$ over $[\underline{x}, \overline{x}] \times [\underline{y}, \overline{y}]$ using *any* range analysis method, we first evaluate two further functions $\partial f/\partial x$ and $\partial f/\partial y$ over $[\underline{x}, \overline{x}] \times [\underline{y}, \overline{y}]$ using the same range analysis method as used to evaluate f itself. If both resulting derivative intervals do not straddle 0, then f increases or decreases monotonically on going across the interval in x and y. Thus, *exact* bounds of $f(x,y)$ over $[\underline{x},\overline{x}] \times [\underline{y},\overline{y}]$ can be obtained immediately as shown below:

- If $\frac{\partial f}{\partial x} > 0$ and $\frac{\partial f}{\partial y} > 0$, then $\underline{F} = f(\underline{x},\underline{y})$, $\overline{F} = f(\overline{x},\overline{y})$;
- If $\frac{\partial f}{\partial x} > 0$ and $\frac{\partial f}{\partial y} < 0$, then $\underline{F} = f(\underline{x},\overline{y})$, $\overline{F} = f(\overline{x},\underline{y})$;
- If $\frac{\partial f}{\partial x} < 0$ and $\frac{\partial f}{\partial y} > 0$, then $\underline{F} = f(\overline{x},\underline{y})$, $\overline{F} = f(\underline{x},\overline{y})$;
- If $\frac{\partial f}{\partial x} < 0$ and $\frac{\partial f}{\partial y} < 0$, then $\underline{F} = f(\overline{x},\overline{y})$, $\overline{F} = f(\underline{x},\underline{y})$;

The same idea can also be used recursively — to get the bounds on $\partial f/\partial x$, one can use its derivatives, i.e. $\partial^2 f/\partial x^2$, $\partial^2 f/\partial x \partial y$, and so on. This process must terminate whenever a derivative is a constant function.

The recursive derivative methods use not only first derivatives but all higher derivative information possible in trying to find the exact bounds of $f(x,y)$ over $[\underline{x},\overline{x}] \times [\underline{y},\overline{y}]$, and therefore they are more accurate. Since the recursive derivative methods require evaluation of higher derivatives, one might expect that these methods need more operations than first derivative only methods. However, our results in Tables 3–12 show that the recursive derivative methods generally not only generate more accurate results but also need less total operations than the first derivative methods. The definite classification of some areas using higher derivatives means that they do not need to be subdivided, which generally seems to outweigh the extra operations required in other areas. One disadvantage of the recursive derivative methods is that they use a lot of stack operations which are not counted here, and which add to the execution time.

4 Experiments

4.1 Test cases

We have now outlined various methods for estimating the range of a function over a rectangle. The rest of this paper considers a series of examples of plotting various functions using the different methods described earlier, in an attempt to understand which methods work best.

First, we briefly describe our set of test cases used to compare the accuracy and speed of each of the methods described. Each test comprises plotting a polynomial $f(x,y) = 0$ using the curve drawing algorithm given in Section 2 on a grid of 256×256 pixels. We used *Mathematica 4.1* as a convenient test bed.

The first example which differs only by a affine change of coordinates from the one in [8] is: $\frac{15}{4} + 8x - 16x^2 + 8y - 112xy + 128x^2y - 16y^2 + 128xy^2 - 128x^2y^2 = 0$ on $[0,1] \times [0,1]$. This is a symmetric low degree polynomial. The curve as shown in Figure 1 consists of three components over the region of interest; two of these meet the boundary and the other is a closed loop. Drawings of the curve produced by various range analysis methods are shown in Figures 11–26. Note that the graphical results for IA on centred form, for AA, for the Bernstein coefficient method, for Taubin's method, and for Rivlin's method (all with or without derivative information) are virtually identical visually, and in the interests of space we have just shown one representative picture in Figure 23.

At first sight it may appear strange that some of the computed results are not merely 'fat' curves, but incorporate lines transversal to the actual curves, especially appearing as isolated short segments (e.g. see Figure 22). This is can be understood by thinking of the output as the intersection of two figures, one drawn by the range analysis method itself, and the other one drawn by the derivative information method. Both of them are continuous, but their intersection may not be continuous. The transversal lines are the curves where the derivatives are equal to zero.

The second example (Figure 2) is from [36]: $20160x^5 - 30176x^4 + 14156x^3 - 2344x^2 + 151x + 237 - 480y = 0$ on $[0,1] \times [0,1]$. This is a strongly asymmetric medium degree polynomial.

The third example is from [35]: $0.945xy - 9.43214x^2y^3 + 7.4554x^3y^2 + y^4 - x^3 = 0$ on $[0,1] \times [0,1]$. This is an asymmetric medium degree polynomial. The curve (Figure 3) consists of two components over the region of interest; each meets the boundary.

The fourth example is from [36]: $x^9 - x^7y + 3x^2y^6 - y^3 + y^5 + y^4x - 4y^4x^3 = 0$ on $[0,1] \times [0,1]$. This is an asymmetric high degree polynomial. The curve (Figure 4) consists of two components over the region of interest; each meets the boundary.

The fifth example (Figure 5) which differs only by a affine change of coordinates from the one in [36] is: $-\frac{1801}{50} + 280x - 816x^2 + 1056x^3 - 512x^4 + \frac{1601}{25}y - 512xy + 1536x^2y - 2048x^3y + 1024x^4y = 0$ on $[0,1] \times [0,1]$. This is an antisymmetric medium degree polynomial.

The sixth example which differs only by a affine change of coordinates from the one in [34] is: $\frac{601}{9} - \frac{872}{3}x + 544x^2 - 512x^3 + 256x^4 - \frac{2728}{9}y + \frac{2384}{3}xy - 768x^2y + \frac{5104}{9}y^2 - \frac{2432}{3}xy^2 + 768x^2y^2 - 512y^3 + 256y^4 = 0$ on $[0,1] \times [0,1]$. This is an asymmetric medium degree polynomial. The curve (Figure 6) consists of two components over the region of interest; each is a closed loop.

The seventh example which also differs only by a affine change of coordinates from the one in [34] is: $-13 + 32x - 288x^2 + 512x^3 - 256x^4 + 64y - 112y^2 + 256xy^2 - 256x^2y^2 = 0$ on $[0,1] \times [0,1]$. This is an asymmetric medium degree polynomial. The curve (Figure 7) comprises a single closed loop containing two cusps.

The eighth example is $-\frac{169}{64} + \frac{51}{8}x - 11x^2 + 8x^3 + 9y - 8xy - 9y^2 + 8xy^2 = 0$ on $[0,1] \times [0,1]$. This is an asymmetric medium degree polynomial. The curve (Figure 8) contains a single compnent with a self-intersection point.

The ninth example which differs only by a affine change of coordinates from the one in [24] is: $47.6 - 220.8x + 476.8x^2 - 512x^3 + 256x^4 - 220.8y + 512xy - 512x^2y + 476.8y^2 - 512xy^2 + 512x^2y^2 - 512y^3 + 256y^4 = 0$ on $[0,1] \times [0,1]$. This is a symmetric medium degree polynomial. The curve (Figure 9) consists of two concentric circles with a relatively small distance between them.

The tenth example is $\frac{55}{256} - x + 2x^2 - 2x^3 + x^4 - \frac{55}{64}y + 2xy - 2x^2y + \frac{119}{64}y^2 - 2xy^2 + 2x^2y^2 - 2y^3 + y^4 = 0$ on $[0,1] \times [0,1]$. This is an asymmetric medium degree polynomial. The curve (Figure 10) consists of two circles which meet tangentially.

These examples have been chosen to illustrate polynomials of varying degrees, with differing numbers of components which may be closed or open, and include curves with cusps, self-intersections and tangencies as special cases. Obviously, no finite set of test cases can establish general truths, but we have aimed to capture a range of curve behaviour with these test cases, including well-known problem cases. This at least gives some hope that any conclusions we draw are not specific to any particular example.

Although our implementations can handle any rectangular plot range, not only the unit square, for comparison purposes we need to use the same plot range for each method. As many methods such as IAB, BC, R, G are only defined on the unit square, we would need to transform the original rectangular plot range to the unit square when these methods are involved. For these reasons we confine the plot range for testing to the unit square $[0,1] \times [0,1]$. This does not lose generality because any other rectangular range can be mapped onto the unit square by a linear variable transformation.

We only show graphical output for the first example for reasons of space. Instead, we present a numerical summary of accuracy and computational expense for each example in table form. When comparing the performance and efficiency of the different range analysis based subdivision methods, a number of quantities were measured:

- The number of pixels filled, the fewer the better: plotted pixels may or may not contain the curve in practice.
- The number of additions (including subtractions) and multiplications (including divisions) needed, the fewer the better.
- The number of subdivisions involved, the lower the better, due to overheads in recursion (stack push and pop operations). Also, less subdivisions result in a smaller number of rectangles used to describe the output, which may improve the speed of any subsequent processing to be performed on the output.

Numbers of operations $(+, -, \times, \div)$ were recorded, rather than CPU times because these quantities are independent of processor or implementation; a *C* language implementation would have been much quicker than the *Mathematica* implementation we used for testing[1]. The numbers of square root operations involved in Gopalsamy's method and its related derivative methods were also recorded for completeness. However, these numbers were insignificant compared to the numbers of additions and multiplications.

Note that several of the methods use matrix manipulations including multiplication and inverse. Since some of the matrices involved are in triangular form, and almost half of the elements are zero, we use "shortened multiplication" which avoids operations involving addition of zeros, in order to reduce the number of of basic operations involved. In a similar way, the number of basic operations involved in computing inverses can also be reduced. The number of operations depends on the choice of algorithm for computing inverse, which we have based on Gaussian elimination. Transposition does not require any operations.

4.2 Results

4.2.1 Description

Tables 3–12 list the measured numbers of arithmetic operations for Examples 1–10 respectively, using each of the methods described before.

For convenience, we use the following notation:

IAP interval arithmetic on power form;

IAPD interval arithmetic on power form plus first derivative information;

IAPRD interval arithmetic on power form plus recursive derivative information;

IAHX interval arithmetic on Horner form x first;

IAHXD interval arithmetic on Horner form, x first, plus first derivative information;

IAHXRD interval arithmetic on Horner form, x first, plus recursive derivative information;

IAHY interval arithmetic on Horner form y first;

IAHYD interval arithmetic on Horner form, y first, plus first derivative information;

IAHYRD interval arithmetic on Horner form, y first, plus recursive derivative information;

IAB interval arithmetic on Bernstein form;

IABD interval arithmetic on Bernstein form plus first derivative information;

IABRD interval arithmetic on Bernstein form plus recursive derivative information;

IAC interval arithmetic on centred form;

IACD interval arithmetic on centred form plus first derivative information;

IACRD interval arithmetic on centred form plus recursive derivative information;

AA affine arithmetic method;

AAD affine arithmetic method plus first derivative information;

AARD affine arithmetic method plus recursive derivative information;

BC Bernstein coefficient method;

[1]Our code is can be obtained by emailing the authors at shh@math.zju.edu.cn

BCD Bernstein coefficient method plus first derivative information;

BCRD Bernstein coefficient method plus recursive derivative information;

T Taubin's method;

TD Taubin's method plus first derivative information.

TRD Taubin's method plus recursive derivative information.

R Rivlin's method;

RD Rivlin's method plus first derivative information.

RRD Rivlin's method plus recursive derivative information.

G Gopalsamy's method;

GD Gopalsamy's method plus first derivative information;

GRD Gopalsamy's method plus recursive derivative information;

4.2.2 Analysis

Our observations from the tabulated results concerning the relative accuracy and speed of these methods are:

- Generally, IAP, IAB, IAHX, IAHY methods are not as accurate as IAC, AA, BC, T, R and G.
- IAP, IAB, IAHX, IAHY methods generally, but not invariably, need more arithmetic operations than the IAC, AA, BC, T and R (Table 6 shows a counterexample).
- IAHX and IAHY generally need less arithmetic operations than IAP or IAB; generally, but not invariably, they are more accurate than IAP but less so than IAB.
- The accuracies of IACRD, AARD, BCRD, TRD and RRD are very similar, but BCRD is usually slightly more accurate.
- The number of arithmetic operations involved in IACRD, AARD, BCRD, TRD and RRD are very similar, but AARD usually needs slightly less arithmetic operations.
- G is less accurate than IAC, AA, BC, T or R, but is usually more accurate than IAP, IAB, IAHX, IAHY methods.
- G needs more arithmetic operations than IAC, AA, BC, T or R, and generally, but not invariably, needs more arithmetic operations than IAP, IAB, IAHX, IAHY methods.
- Including first derivative information improves classification of IAP, IAB, IAHX, IAHY and G methods, but not necessarily the numbers of arithmetic operations.
- Including first derivative information slightly improves classification of IAC, AA, T and R methods, but not necessarily the numbers of arithmetic operations.
- Including first derivative information does not affect the classification of the BC method, but only increases the numbers of arithmetic operations.
- Including recursive derivative information greatly improves classification of IAP, IAB, IAHX, IAHY and G methods and often greatly reduces the numbers of arithmetic operations as well.

- Including recursive derivative information slightly improves the classification of IAC, AA, T and R methods and generally reduces the numbers of arithmetic operations as well.
- Including recursive derivative information does not improve the classification of the BC method but generally reduces the numbers of arithmetic operations.
- Function range analysis based subdivision methods have no difficulties in handling cusps, tangencies, self-intersections and multiple closed loops, unlike continuation methods.

5 Conclusions

Clearly, a small set of examples cannot lead to definitive statements about the relative merits of each method in every possible case. Indeed, we can see that for differing test cases, differing methods are best. Nevertheless, these carefully chosen test cases do demonstrate some general conclusions, which we believe will be generally useful.

Summarizing the experimental observations above, we note that the IAC, AA, BC, T, and R methods, and their related derivative methods, are roughly similar in performance, and are generally better than the G, IAP, IAB, IAHX, and IAHY methods.

Our recommendation for practical applications is that the AARD (affine arithmetic with recursive derivative information) method is probably the best overall choice for accuracy and computational efficiency, but the IACRD, BCRD, TRD and RRD methods are very similar in terms of accuracy and computational efficiency. However, the IACRD method is easiest to implement, while AARD, BCRD, TRD and RRD are a little more difficult.

Acknowledgements

We thank the China Scholarship Council for funding Huahao Shou, Bath University and the ORS award scheme for funding Irina Voiculescu, and the anonymous referees for valuable comments and suggestions which have improved the paper.

References

[1] Barth, W., Lieger, R., Schindler, M., Ray Tracing General Parametric Surfaces Using Interval Arithmetic, *The Visual Computer*, 1994, 10: 363–371.

[2] Berchtold, J., *The Bernstein Form in Set-Theoretic Geometric Modelling*, PhD Thesis, University of Bath, 2000.

[3] Berchtold, J., Bowyer, A., Robust Arithmetic for Multivariate Bernstein-Form Polynomials, *Computer Aided Design*, 2000, 32(11): 681–689.

[4] Berchtold, J., Voiculescu I., Bowyer, A., *Interval Arithmetic Applied to Multivariate Bernstein-Form Polynomials*, Technical Report Number 31/98, School of Mechanical Engineering, University of Bath, 1998. Avaible at http://www.bath.ac.uk/∼ensab/G_ mod/Bernstein/interval.html

[5] Bowyer, A., Berchtold, J., Eisenthal, D., Voiculescu, I., and Wise, K., Interval Methods in Geometric Modelling, In: Martin, R., Wang, W., (Eds.), *Geometric Modeling and Processing 2000*, IEEE Computer Society Press, 2000, 321–327.

[6] Cargo, G. T. and Shisha, O., The Bernstein Form of a Polynomial, *Journal of Research of the National Bureau of Standards–B. Mathematical Sciences*, 1966, 70B: 79–81.

[7] Chandler, R. E., A Tracking Algorithm for Implicitly Defined Curves, *IEEE Computer Graphics & Applications*, 1988, 8(2): 83–89.

[8] Comba, J. L. D., Stolfi, J., Affine Arithmetic and its Applications to Computer Graphics, *Anais do VII SIBGRAPI*, 1993, 9–18. Available at http://www.dcc.unicamp.br/∼stolfi/.

[9] Duff, T., Interval Arithmetic and Recursive Subdivision for Implicit Functions and Constructive Solid Geometry, *Computer Graphics (SIGGRAPH'92 Proceedings)*, 1992, 26(2): 131–138.

[10] Farin, G., *Curves and Surfaces in Computer Aided Geometric Design*, 3rd ed., Academic Press, San Diego,1993.

[11] Farouki, R. T., Rajan, V. T., On the Numerical Condition of Polynomials in Bernstein Form, *Computer Aided Geometric Design*, 1987, 4(3): 191–216.

[12] Farouki, R. T., Rajan, V. T., Algorithms for Polynomials in Bernstein Form, *Computer Aided Geometric Design*, 1988, 5(1): 1–26.

[13] de Figueiredo, L. H., Surface Intersection Using Affine Arithmetic, *Proceedings of Graphics Interface*, 1996, 168–175.

[14] de Figueiredo, L. H., Stolfi, J., Adaptive Enumeration of Implicit Surfaces with Affine Arithmetic, *Computer Graphics Forum*, 1996, 15(5): 287–296.

[15] Garloff, J., Convergent Bounds for the Range of Multivariate polynomials, *Interval Mathematics 1985, Lecture Notes in Computer Science*, 1985, 212: 37–56.

[16] Gopalsamy, S., Khandekar, D., Mudur, S. P., A New Method of Evaluating Compact Geometric Bounds for Use in Subdivision Algorithms, *Computer Aided Geometric Design*, 1991, 8(5): 337–356.

[17] Heidrich, W., Slusallek, P., Seidel, H. P., Sampling of Procedural Shaders Using Affine Arithmetic, *ACM Transaction on Graphics*, 1998, 17(3): 158–176.

[18] Hu, C. Y., Patrikalakis, N. M., Ye, X., Robust Interval Solid Modeling: Part I, Representations, *Computer Aided Design*, 1996, 28(10): 807–817.

[19] Hu, C. Y., Patrikalakis, N. M., Ye, X., Robust Interval Solid Modeling: Part II, Boundary Evaluation, *Computer Aided Design*, 1996, 28(10): 819–830.

[20] Milne, P. S., *On the Algorithms and Implementation of a Geometric Algebra System*, University of Bath Computer Science Technical Report 90–40, 1990 (e-mail: tech-report@maths.bath.ac.uk).

[21] Moore, R. E., *Interval Analysis*, Prentice-Hall, 1966.

[22] Moore, R. E., *Methods and Applications of Interval Analysis*, Society for Industrial and Applied Mathematics, 1979.

[23] Ratschek, H., Rokne, J., *Computer Methods for the Range of Functions*, Ellis Horwood Ltd., 1984.

[24] Ratschek, H., Rokne, J., SCCI-Hybrid Methods for 2D-Curve Tracing, submitted for publication.

[25] Rivlin, T. J., Bounds on a Polynomial, *Journal of Research of the National Bureau of Standards–B. Mathematical Sciences*, 1970, 74B(1): 47–54.

[26] Sederberg, T. W., Farouki, R. T., Approximation by Interval Bézier Curves, *IEEE Computer Graphics & Applications*, 1992, 12(5): 87–95.

[27] Shou H., Martin R., Voiculescu I., Bowyer A., Wang G., Affine Arithmetic in Matrix Form for Algebraic Curve Drawing, *Progress in Natural Science (China)*, 2002, 12(1): 77–80.

[28] Snyder, J. M., Interval Analysis for Computer Graphics, *Computer Graphics (SIGGRAPH'92 Proceedings)*, 1992, 26(2): 121–130.

[29] Suffern, K. G., Quadtree Algorithms for Contouring Functions of Two Variables, *The Computer Journal*, 1990, 33: 402–407.

[30] Suffern, K. G., Fackerell, E. D., Interval Methods in Computer Graphics, *Computers & Graphics*, 1991, 15: 331–340.

[31] Taubin, G., Distance Approximations for Rasterizing Implicit Curves, *ACM Transactions on Graphics*, 1994, 13(1): 3–42.

[32] Taubin, G., Rasterizing Algebraic Curves and Surfaces, *IEEE Computer Graphics and Applications*, 1994, 14: 14–23.

[33] Tuohy, S. T., Maekawa, T., Shen, G., Patrikalakis, N. M., Approximation of Measured Data with Interval B-Splines, *Computer Aided Design*, 1997, 29(11): 791–799.

[34] Voiculescu, I., Implicit Function Algebra in Set-theoretic Modelling, University of Bath Ph.D. thesis (submitted, to be examined)

[35] Voiculescu, I., Berchtold, J. Bowyer, A. Martin, R. R., Zhang, Q., Interval and Affine Arithmetic for Surface Location of Power- and Bernstein-form Polynomials, In: Cipolla, R., Martin, R. R., (Eds.) *Mathematics of Surfaces IX*, 2000, 410–423.

[36] Zhang, Q., Martin, R. R., Polynomial Evaluation using Affine Arithmetic for Curve Drawing, *Proceedings Eurographics UK Conference*, 2000, 49–56.

Methods	Pixels plotted	Subdivisions	Additions	Multiplications	Square roots
IAP	25567	15332	4916592	6010240	
IAPD	8891	8652	4354562	5406276	
IAPRD	1282	1662	1439351	924556	
IAHX	10733	6852	1372266	1315630	
IAHXD	3112	3225	1149422	1099988	
IAHXRD	1011	1187	955775	453688	
IAHY	10733	6852	1372266	1315630	
IAHYD	3112	3225	1148630	1099988	
IAHYRD	1011	1187	955633	453688	
IAB	3946	3467	1742472	1969396	
IABD	888	1535	1671154	1919602	
IABRD	662	974	916045	638662	
IAC	526	563	435790	378420	
IACD	522	545	644162	542603	
IACRD	522	545	487349	301389	
AA	526	563	404262	171226	
AAD	522	545	575822	283096	
AARD	522	545	478337	264396	
BC	522	535	388714	188572	
BCD	522	535	1175398	807960	
BCRD	522	535	538651	327622	
T	530	587	473100	143287	
TD	522	545	723215	243205	
TRD	522	545	492432	258433	
R	534	585	669849	427489	
RD	522	553	1025493	614414	
RRD	522	553	543568	316112	
G	1072	1031	1430258	1368583	4125
GD	630	723	1764041	1435000	6627
GRD	574	609	647386	433080	609

Table 3: Example 1. Comparison of range analysis based subdivision methods for drawing the curve $\frac{15}{4} + 8x - 16x^2 + 8y - 112xy + 128x^2y - 16y^2 + 128xy^2 - 128x^2y^2 = 0$.

Methods	Pixels plotted	Subdivisions	Additions	Multiplications	Square roots
IAP	17680	11458	3485276	4308300	
IAPD	9057	8220	3646732	4511428	
IAPRD	726	1216	503413	490700	
IAHX	2643	2530	400228	384596	
IAHXD	626	1108	486772	462324	
IAHXRD	463	621	387027	336266	
IAHY	2643	2530	400228	384596	
IAHYD	626	1108	486772	462324	
IAHYRD	463	621	387027	336266	
IAB	3087	2933	4077830	4998256	
IABD	837	1728	4150256	5274376	
IABRD	494	799	624347	826794	
IAC	433	459	656630	624578	
IACD	432	447	426520	392894	
IACRD	432	445	187933	172769	
AA	433	459	601510	407812	
AAD	432	447	314856	266244	
AARD	432	445	179407	156053	
BC	432	444	621548	448530	
BCD	432	444	1026684	1061312	
BCRD	432	444	263595	261641	
T	435	471	671867	350608	
TD	432	449	443947	235496	
TRD	432	447	191515	152692	
R	434	470	864116	697857	
RD	432	456	619102	458333	
RRD	432	454	226630	194686	
G	1581	1605	6147408	9387500	6421
GD	473	656	2065355	2742316	5913
GRD	446	510	444176	553800	414

Table 4: Example 2. Comparison of range analysis based subdivision methods for drawing the curve $20160x^5 - 30176x^4 + 14156x^3 - 2344x^2 + 151x + 237 - 480y = 0$.

Methods	Pixels plotted	Subdivisions	Additions	Multiplications	Square roots
IAP	4026	3909	1493798	1876438	
IAPD	821	1840	1140246	1542534	
IAPRD	623	951	1390208	1037968	
IAHX	2537	2468	646954	789838	
IAHXD	690	1131	668336	860594	
IAHXRD	610	792	1281666	1065100	
IAHY	2672	2609	621250	751462	
IAHYD	673	1108	687886	867346	
IAHYRD	610	798	1322760	1130234	
IAB	1579	1564	2149336	2615424	
IABD	613	780	2729162	3447920	
IABRD	599	708	4114797	3103070	
IAC	608	631	1546481	1573073	
IACD	593	594	1916664	1837020	
IACRD	592	589	1169674	896039	
AA	608	634	1178329	646933	
AAD	593	596	1549458	817927	
AARD	592	591	1103553	687932	
BC	592	585	1101958	738022	
BCD	592	585	3044980	2651225	
BCRD	592	585	1415699	1056651	
T	609	638	1193422	543787	
TD	593	597	1680410	688301	
TRD	592	592	1153440	661934	
R	610	640	1850213	1590386	
RD	593	604	2486531	1938248	
RRD	592	598	1371473	958173	
G	1799	1755	10858163	17110175	7021
GD	643	755	5695524	7547702	6587
GRD	606	669	2219246	2327926	2074

Table 5: Example 3. Comparison of range analysis based subdivision methods for drawing the curve $0.945xy - 9.43214x^2y^3 + 7.4554x^3y^2 + y^4 - x^3$.

Methods	Pixels plotted	Subdivisions	Additions	Multiplications	Square roots
IAP	2560	2430	2367580	3032950	
IAPD	801	1027	1620456	2298498	
IAPRD	778	886	5909984	5078928	
IAHX	1538	1496	930832	1137148	
IAHXD	784	865	1843724	2373058	
IAHXRD	776	822	7069814	8552740	
IAHY	1417	1370	852480	1052350	
IAHYD	773	808	1322608	1720094	
IAHYRD	772	791	8559296	10299586	
IAB	2156	2107	21860804	27933704	
IABD	798	1040	30356562	40904380	
IABRD	780	879	7305730	8824128	
IAC	816	857	9324996	11168193	
IACD	787	805	13491137	15508341	
IACRD	778	795	8094666	7467202	
AA	816	857	6773822	6302500	
AAD	787	805	10281617	8838443	
AARD	778	795	7421194	5923621	
BC	770	756	6701649	7001260	
BCD	770	756	21970123	28303068	
BCRD	770	756	8579701	8391015	
T	819	880	6508351	5510363	
TD	790	813	10091322	7698515	
TRD	782	798	7516455	5700579	
R	826	894	10642248	11835093	
RD	790	821	15869922	16651905	
RRD	779	807	9228983	8084017	
G	4586	4458	173931656	468575967	17833
GD	1305	2112	132033458	313476478	20266
GRD	889	1178	34030133	64049202	9873

Table 6: Example 4. Comparison of range analysis based subdivision methods for drawing the curve $x^9 - x^7y + 3x^2y^6 - y^3 + y^5 + y^4x - 4y^4x^3 = 0$.

Methods	Pixels plotted	Subdivisions	Additions	Multiplications	Square roots
IAP	55158	19562	9336638	11580850	
IAPD	47223	18431	16553602	20607928	
IAPRD	1813	1601	1731156	1534894	
IAHX	45896	17601	3316604	3097818	
IAHXD	33889	14603	9031358	8757640	
IAHXRD	1667	1522	2159716	1743454	
IAHY	43259	17442	4533596	4325676	
IAHYD	33192	14386	10791388	10403372	
IAHYRD	1718	1599	2346688	1857032	
IAB	30212	12735	11862746	14161596	
IABD	25604	11191	19972916	23964246	
IABRD	850	985	1032154	999998	
IAC	464	611	736514	686915	
IACD	456	535	997286	877847	
IACRD	456	477	479671	403475	
AA	464	611	599656	339853	
AAD	456	535	813646	518040	
AARD	456	477	465475	368272	
BC	456	465	483239	294463	
BCD	456	465	1776552	1701900	
BCRD	456	465	519443	440263	
T	470	659	720470	303253	
TD	456	589	1114931	496031	
TRD	456	477	478283	362330	
R	470	690	1014447	785072	
RD	456	598	1478852	1075782	
RRD	456	497	557549	442373	
G	1378	1973	6458419	9777212	7893
GD	485	1215	6099441	7626677	11188
GRD	464	583	879911	941563	640

Table 7: Example 5. Comparison of range analysis based subdivision methods for drawing the curve $-\frac{1801}{50}+280x-816x^2+1056x^3-512x^4+\frac{1601}{25}y-512xy+1536x^2y-2048x^3y+1024x^4y=0$.

Methods	Pixels plotted	Subdivisions	Additions	Multiplications	Square roots
IAP	48088	19132	11403848	14234392	
IAPD	29321	14039	13391336	16830580	
IAPRD	2819	2525	3007275	2539108	
IAHX	31099	14399	4415370	4262176	
IAHXD	19521	9617	8160076	7896454	
IAHXRD	2435	2417	3047763	2080756	
IAHY	31008	14373	4407258	4254480	
IAHYD	19179	9449	8014434	7758502	
IAHYRD	2461	2384	3005033	2051422	
IAB	14131	8315	22291500	27207496	
IABD	8535	6978	54778878	67714114	
IABRD	1979	2206	5090611	5848074	
IAC	460	558	1463312	1464788	
IACD	455	501	1346822	1244627	
IACRD	455	482	731951	605886	
AA	460	560	1329630	788830	
AAD	455	502	1266280	700824	
AARD	455	483	706751	468204	
BC	454	454	1104452	789386	
BCD	454	454	2209336	1916862	
BCRD	454	454	889325	747434	
T	466	596	1395910	694033	
TD	455	514	1422903	609968	
TRD	455	488	742887	447188	
R	472	601	2053509	1678435	
RD	457	525	2027672	1465521	
RRD	457	504	945500	699666	
G	1291	1596	8350370	11824383	6385
GD	512	779	4795051	5427360	7207
GRD	486	607	1525807	1650371	1314

Table 8: Example 6. Comparison of range analysis based subdivision methods for drawing the curve $\frac{601}{9}-\frac{872}{3}x+544x^2-512x^3+256x^4-\frac{2728}{9}y+\frac{2384}{3}xy-768x^2y+\frac{5104}{9}y^2-\frac{2432}{3}xy^2+768x^2y^2-512y^3+256y^4=0$.

Methods	Pixels plotted	Subdivisions	Additions	Multiplications	Square roots
IAP	25875	13140	4860268	5991952	
IAPD	11415	8326	4682586	5885352	
IAPRD	1257	1295	1159615	771280	
IAHX	14588	8358	1880448	1939112	
IAHXD	5453	4251	2283366	2421118	
IAHXRD	1114	1135	1072335	622900	
IAHY	11375	6538	1470836	1412260	
IAHYD	4025	3222	1618706	1595296	
IAHYRD	1074	1097	1023799	580636	
IAB	9042	5065	7421300	8955360	
IABD	3488	2619	9804078	12066954	
IABRD	1011	1014	1378353	1318774	
IAC	512	625	926157	860294	
IACD	450	499	975782	841732	
IACRD	450	497	536422	361290	
AA	512	627	873923	476708	
AAD	450	501	932182	479614	
AARD	450	497	523494	296746	
BC	426	437	623874	394011	
BCD	426	437	1575655	1270938	
BCRD	426	437	574694	416959	
T	532	675	983951	421354	
TD	456	521	1122203	423094	
TRD	456	513	562111	294457	
R	510	624	1243219	919788	
RD	456	521	1514864	990452	
RRD	456	513	629178	397463	
G	1414	1689	6419333	8270251	6757
GD	503	721	3269620	3387019	6795
GRD	472	609	1103949	1067172	729

Table 9: Example 7. Comparison of range analysis based subdivision methods for drawing the curve $-13 + 32x - 288x^2 + 512x^3 - 256x^4 + 64y - 112y^2 + 256xy^2 - 256x^2y^2 = 0$.

Methods	Pixels plotted	Subdivisions	Additions	Multiplications	Square roots
IAP	17223	10744	2583044	3094342	
IAPD	2444	3793	1106100	1328828	
IAPRD	967	1175	501229	309466	
IAHX	7512	5493	919952	878918	
IAHXD	1242	1707	515150	479292	
IAHXRD	878	991	460279	227248	
IAHY	7429	5483	918000	877318	
IAHYD	1220	1699	484126	462154	
IAHYRD	881	1001	433011	213700	
IAB	5292	4055	3695222	4347226	
IABD	1228	1671	3404828	4131398	
IABRD	862	960	564873	523404	
IAC	818	827	842101	727934	
IACD	808	813	864027	711250	
IACRD	806	803	347400	209274	
AA	818	827	855337	397078	
AAD	808	813	875785	429406	
AARD	806	803	347386	197538	
BC	804	791	827964	456045	
BCD	804	791	1652898	1135417	
BCRD	804	791	364427	222577	
T	838	853	947054	337885	
TD	808	817	1084939	366272	
TRD	806	805	353928	195778	
R	832	851	1277816	830144	
RD	808	817	1463993	846534	
RRD	806	807	369676	216376	
G	1888	1871	4016062	3997209	7485
GD	822	886	1612136	1269898	7455
GRD	812	843	439705	294257	230

Table 10: Example 8. Comparison of range analysis based subdivision methods for drawing the curve $-\frac{169}{64}+\frac{51}{8}x-11x^2+8x^3+9y-8xy-9y^2+8xy^2=0$.

Methods	Pixels plotted	Subdivisions	Additions	Multiplications	Square roots
IAP	49971	19545	11653398	14541664	
IAPD	33761	15283	14845250	18578276	
IAPRD	3284	2705	3024465	2524648	
IAHX	33026	14807	4544662	4382944	
IAHXD	22026	10832	9563102	9263614	
IAHXRD	2805	2635	3530831	2477374	
IAHY	33026	14807	4544662	4382944	
IAHYD	22026	10832	9474452	9176956	
IAHYRD	2805	2635	3446441	2393046	
IAB	18860	9629	25824052	31506904	
IABD	12956	8389	66581950	82125622	
IABRD	1710	1522	2597999	3065494	
IAC	1144	1261	4154102	4379058	
IACD	1080	1053	3303970	3184889	
IACRD	1080	1033	1571521	1361269	
AA	1144	1269	3012696	1787102	
AAD	1080	1057	2696170	1490676	
AARD	1080	1037	1397061	964716	
BC	1073	1000	2431524	1737242	
BCD	1073	1000	4983432	4336576	
BCRD	1073	1000	1785391	1545664	
T	1208	1373	3215504	1598461	
TD	1080	1085	3035159	1299883	
TRD	1080	1037	1413728	915445	
R	1208	1393	5028905	4653463	
RD	1080	1119	4572173	3584474	
RRD	1080	1099	1885110	1538142	
G	3160	3237	24751718	46111387	12949
GD	1120	1541	12396588	17507303	14326
GRD	1112	1305	3788300	5547762	1707

Table 11: Example 9. Comparison of range analysis based subdivision methods for drawing the curve $47.6 - 220.8x + 476.8x^2 - 512x^3 + 256x^4 - 220.8y + 512xy - 512x^2y + 476.8y^2 - 512xy^2 + 512x^2y^2 - 512y^3 + 256y^4 = 0$.

Methods	Pixels plotted	Subdivisions	Additions	Multiplications	Square roots
IAP	45865	18727	11157626	13483618	
IAPD	30651	14329	13858740	17001158	
IAPRD	1696	1553	1644149	1320658	
IAHX	28159	13157	3822978	3894544	
IAHXD	19496	9724	8000502	8184954	
IAHXRD	1514	1553	1865047	1294114	
IAHY	28077	13075	3799526	3870272	
IAHYD	19158	9610	7897612	8082810	
IAHYRD	1508	1533	1844109	1280858	
IAB	12680	7605	16493268	20138700	
IABD	8672	6461	42295318	51683386	
IABRD	920	1263	1813369	1937956	
IAC	784	841	2205466	2193876	
IACD	772	793	2144124	1962977	
IACRD	772	781	832283	657657	
AA	784	845	2006376	1190110	
AAD	772	797	2024514	1103620	
AARD	772	785	826231	600604	
BC	772	773	1879618	1343170	
BCD	772	773	3525360	3015788	
BCRD	772	773	923947	749912	
T	812	905	2119736	1053709	
TD	776	817	2290543	964535	
TRD	776	801	859980	604001	
R	804	890	3039295	2484151	
RD	780	827	3174385	2277012	
RRD	776	806	937889	714606	
G	2316	2385	12449158	17654749	9541
GD	860	1175	6944787	7798088	10713
GRD	792	875	1281869	1264587	488

Table 12: Example 10. Comparison of range analysis based subdivision methods for drawing the curve $\frac{55}{256} - x + 2x^2 - 2x^3 + x^4 - \frac{55}{64}y + 2xy - 2x^2y + \frac{119}{64}y^2 - 2xy^2 + 2x^2y^2 - 2y^3 + y^4 = 0$.

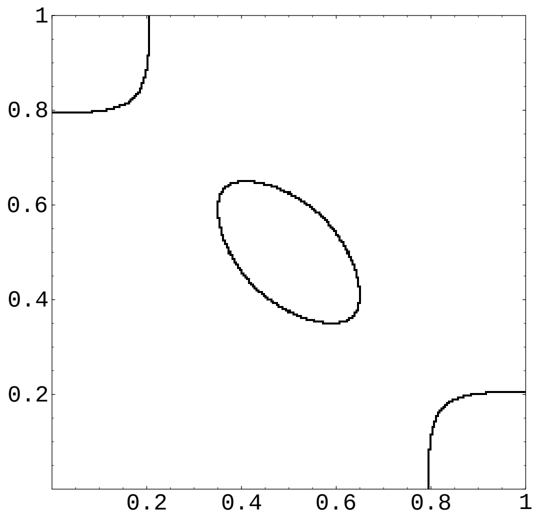

Figure 1: Example 1. $\frac{15}{4} + 8x - 16x^2 + 8y - 112xy + 128x^2y - 16y^2 + 128xy^2 - 128x^2y^2 = 0$.

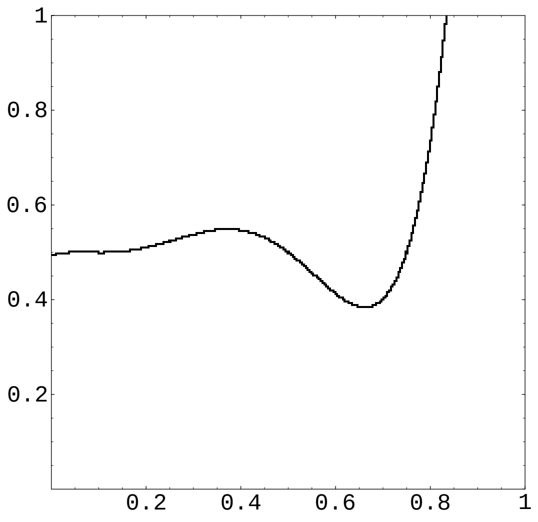

Figure 2: Example 2. $20160x^5 - 30176x^4 + 14156x^3 - 2344x^2 + 151x + 237 - 480y = 0$.

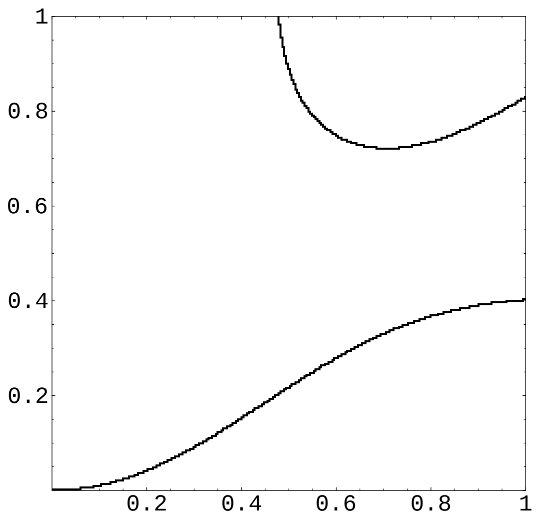

Figure 3: Example 3. $0.945xy - 9.43214x^2y^3 + 7.4554x^3y^2 + y^4 - x^3 = 0$.

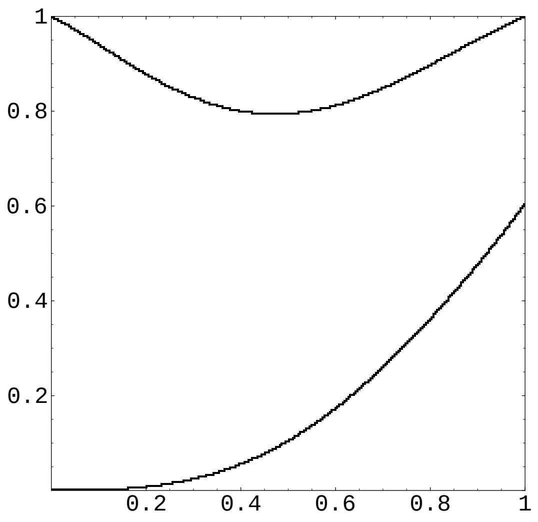

Figure 4: Example 4. $x^9 - x^7y + 3x^2y^6 - y^3 + y^5 + y^4x - 4y^4x^3 = 0$.

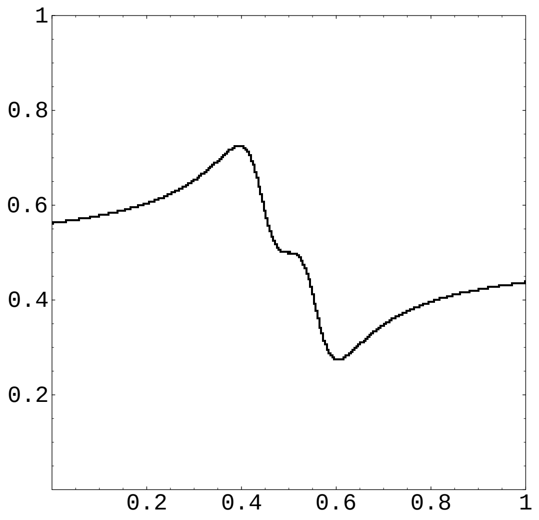

Figure 5: Example 5. $-\frac{1801}{50}+280x-816x^2+1056x^3-512x^4+\frac{1601}{25}y-512xy+1536x^2y-2048x^3y+1024x^4y=0$.

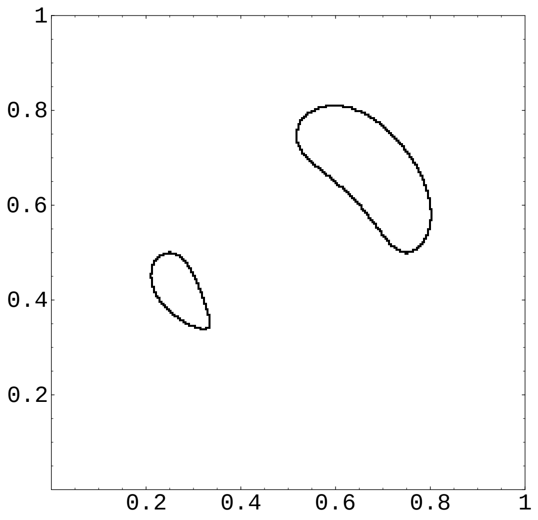

Figure 6: Example 6. $\frac{601}{9}-\frac{872}{3}x+544x^2-512x^3+256x^4-\frac{2728}{9}y+\frac{2384}{3}xy-768x^2y+\frac{5104}{9}y^2-\frac{2432}{3}xy^2+768x^2y^2-512y^3+256y^4=0$.

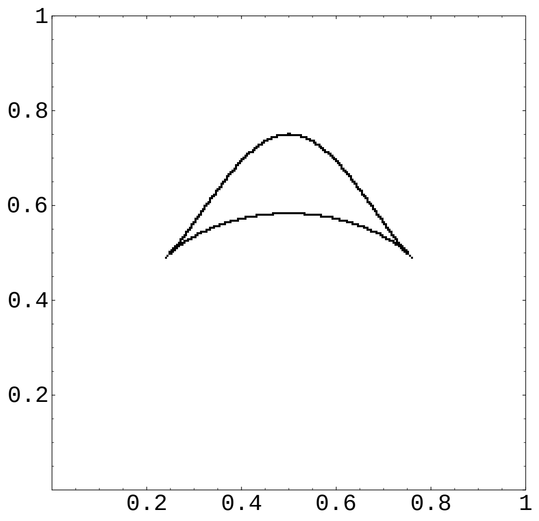

Figure 7: Example 7. $-13+32x-288x^2+512x^3-256x^4+64y-112y^2+256xy^2-256x^2y^2=0$.

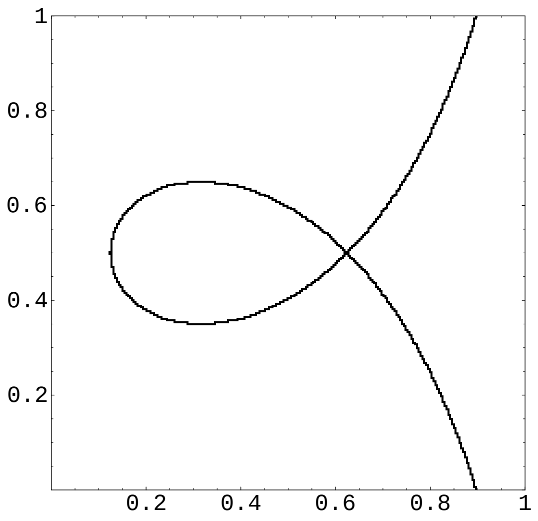

Figure 8: Example 8. $-\frac{169}{64}+\frac{51}{8}x-11x^2+8x^3+9y-8xy-9y^2+8xy^2=0$.

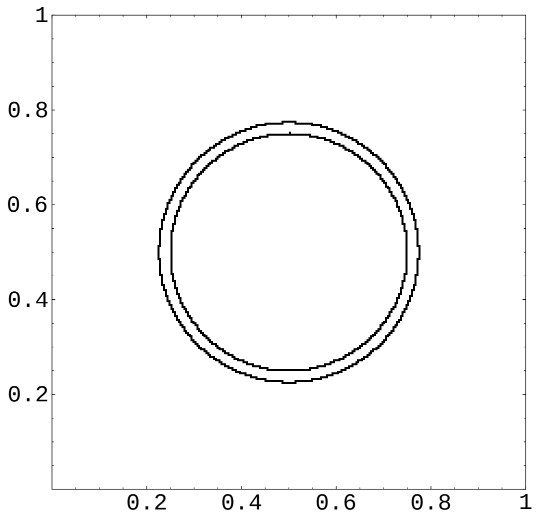

Figure 9: Example 9. $47.6-220.8x+476.8x^2-512x^3+256x^4-220.8y+512xy-512x^2y+476.8y^2-512xy^2+512x^2y^2-512y^3+256y^4=0$.

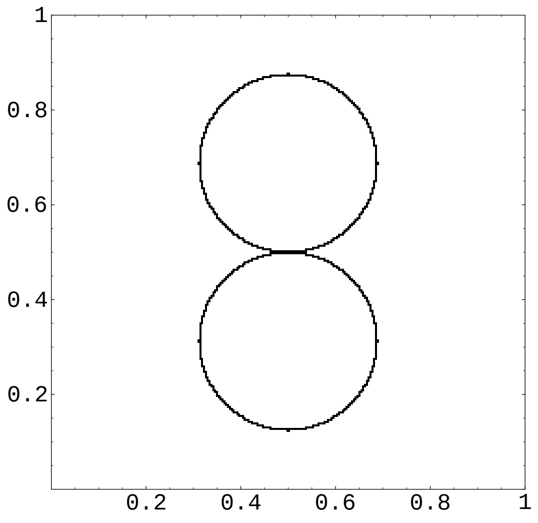

Figure 10: Example 10. $\frac{55}{256}-x+2x^2-2x^3+x^4-\frac{55}{64}y+2xy-2x^2y+\frac{119}{64}y^2-2xy^2+2x^2y^2-2y^3+y^4=0$.

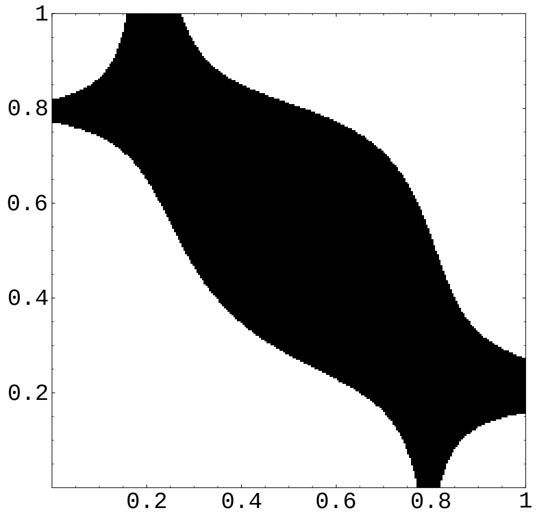

Figure 11: IA on power form.

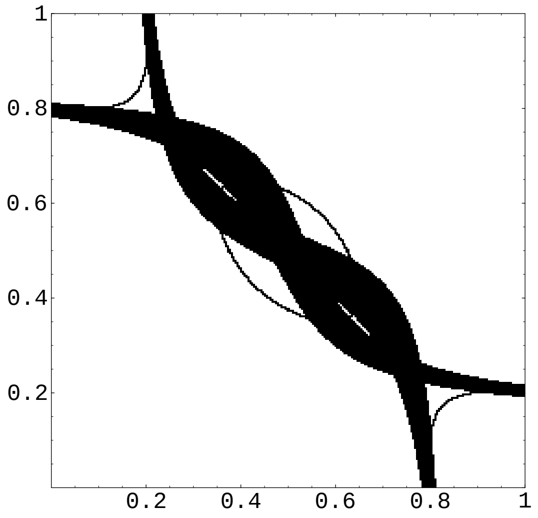

Figure 12: IA on power form plus first derivative information.

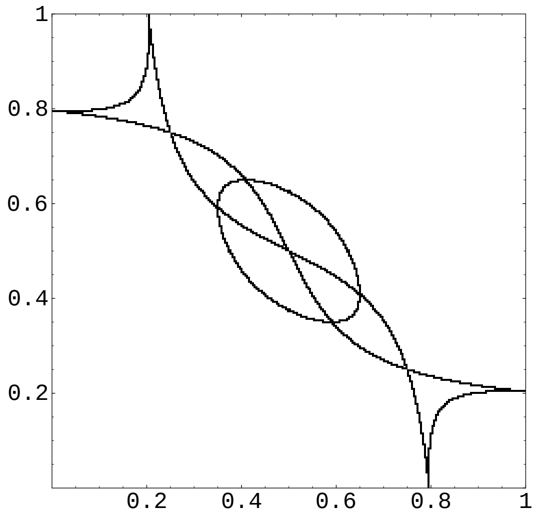

Figure 13: IA on power form plus recursive derivative information.

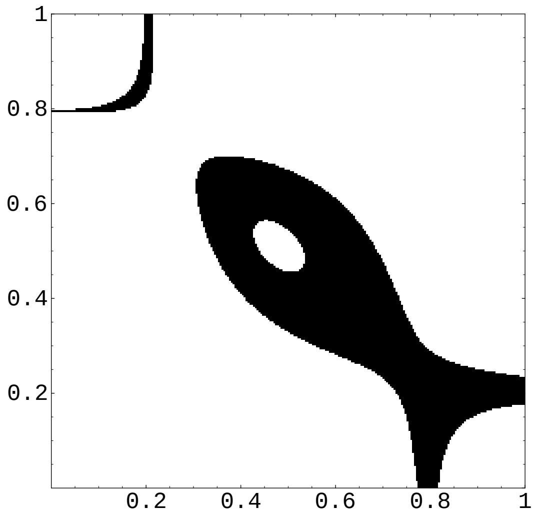

Figure 14: IA on Horner form x first.

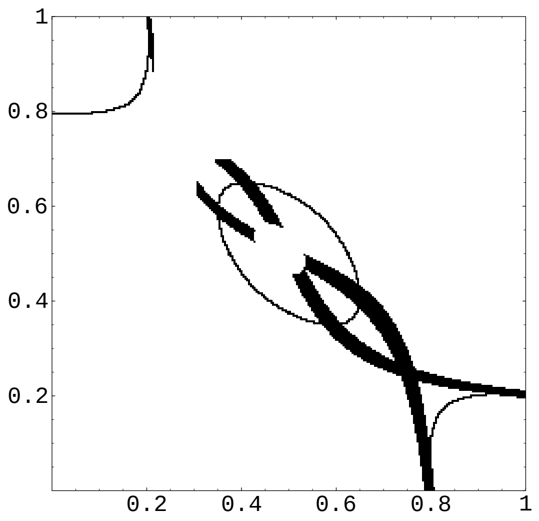

Figure 15: IA on Horner form x first, plus first derivative information.

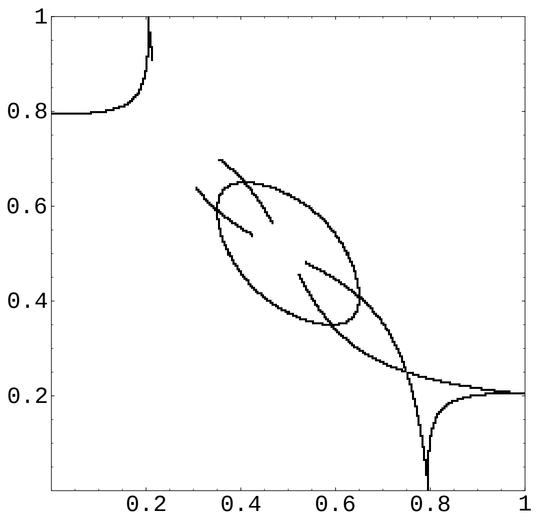

Figure 16: IA on Horner form x first, plus recursive derivative information.

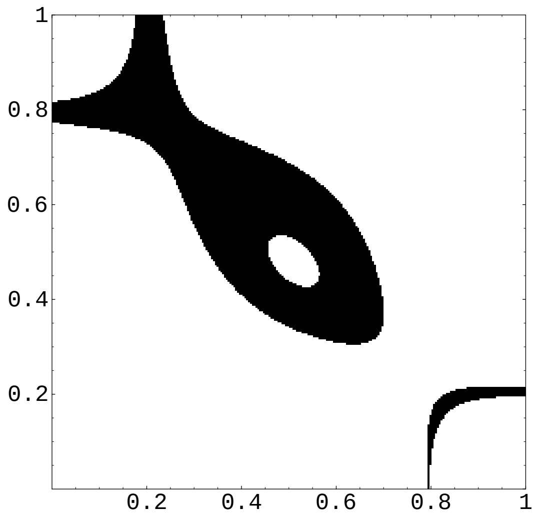

Figure 17: IA on Horner form y first.

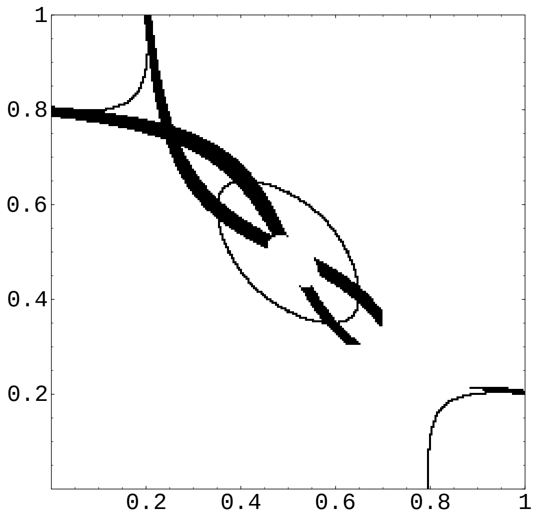

Figure 18: IA on Horner form y first, plus first derivative information.

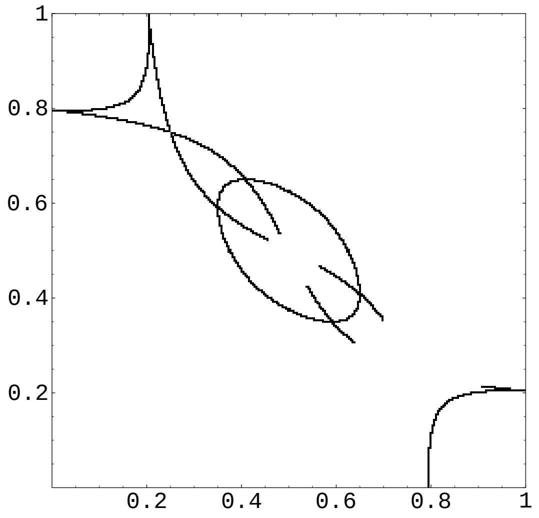

Figure 19: IA on Horner form y first, plus recursive derivative information.

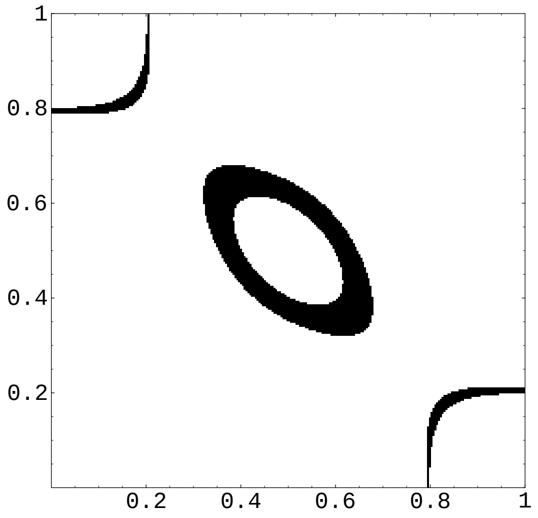

Figure 20: IA on Bernstein form.

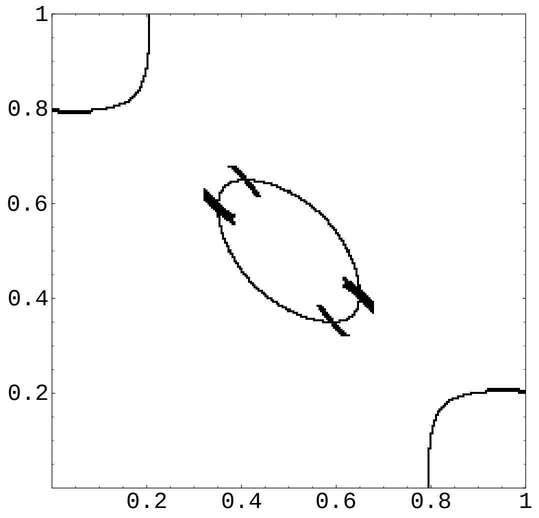

Figure 21: IA on Bernstein form plus first derivative information.

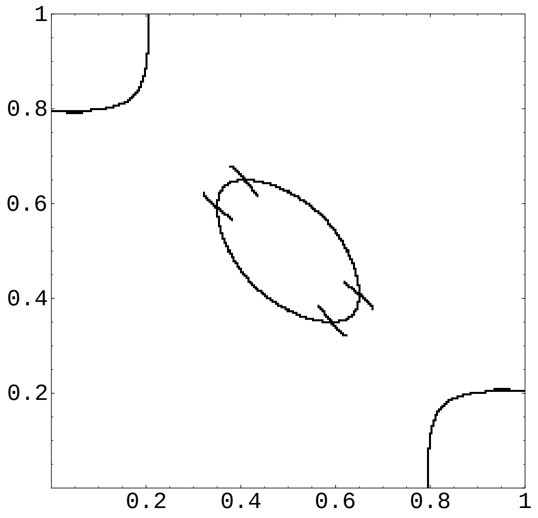

Figure 22: IA on Bernstein form plus recursive derivative information.

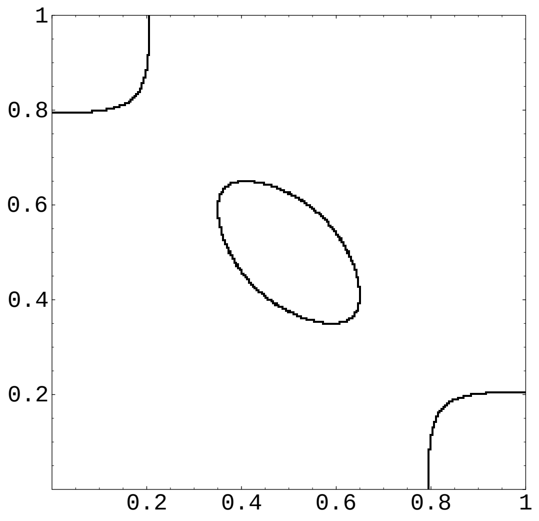

Figure 23: Representative picture for all 'good' methods.

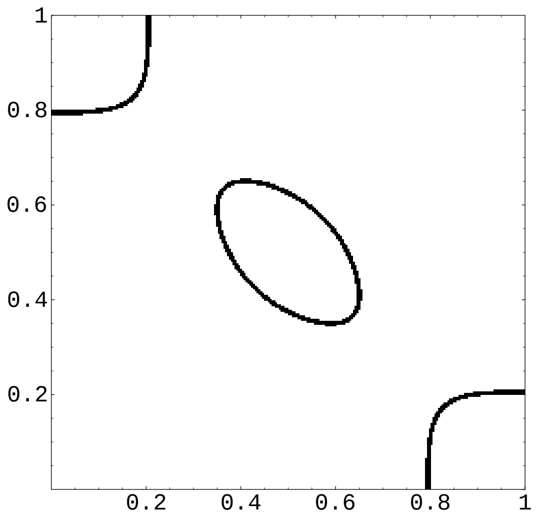

Figure 24: Gopalsamy's method.

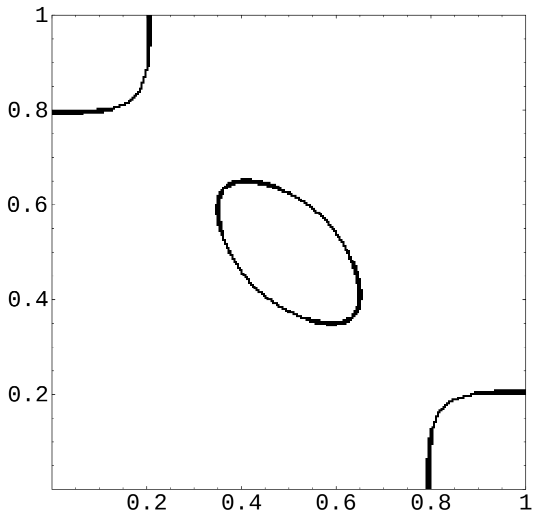

Figure 25: Gopalsamy's method plus first derivative information.

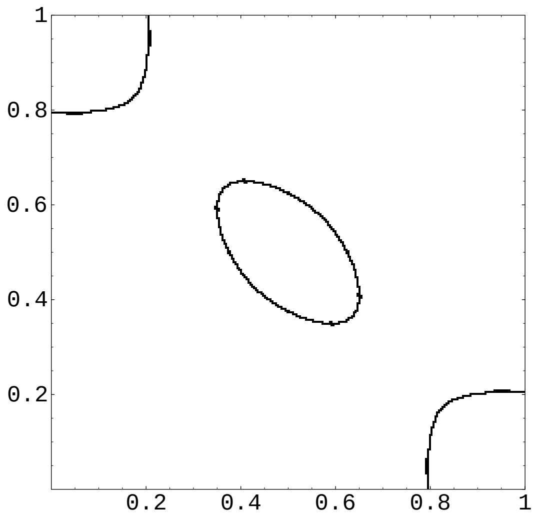

Figure 26: Gopalsamy's method plus recursive derivative information.