Affine arithmetic in matrix form for polynomial evaluation and algebraic curve drawing*

SHOU Huahao[1]**, Ralph MARTIN[2], Irina VOICULESCU[3], Adrian BOWYER[4] and WANG Guojin[1]

(1. Department of Mathematics, Zhejiang University, Hangzhou 310027, China; 2. Department of Computer Science, Cardiff University, Cardiff, UK; 3. Computing Laboratory, Oxford University, Oxford, UK; 4. Department of Mechanical Engineering, University of Bath, Bath, UK)

Received March 12, 2001; revised June 6, 2001

Abstract This paper shows how tight bounds for the range of a bivariate polynomial can be found using a matrix method based on affine arithmetic. Then, this method is applied to drawing an algebraic curve with a hierarchical algorithm, which demonstrates that more accurate answers can be obtained more rapidly than using conventional interval arithmetic.

Keywords: affine arithmetic, interval arithmetic, algebraic curve drawing.

Solving $f(x, y) = 0$ in a rectangular area $[\underline{x}, \bar{x}] \times [\underline{y}, \bar{y}]$ is a problem with many practical applications in CAD and computer graphics. One such example is drawing algebraic curves; other applications include surface-surface intersection and computation of the silhouette curve of a parametric surface[1].

Let C be an algebraic curve defined implicitly by the equation $f(x, y) = 0$, where $f(x, y)$ is a polynomial in two variables. A simple and general technique for computing an approximation of C on a rectangular region Ω as described in Ref. [2] is: (i) decompose Ω into small cells; (ii) identify which cells intersect C; (iii) approximate C within each intersecting cell. Typically, the cellular decomposition of Ω is a regular grid of rectangles, so identifying the intersecting cells is usually the most expensive step in this method. In the simplest schema, the cells that intersect C are identified by point sampling of each cell. However, this method may cause aliasing and is extremely inefficient.

Continuation methods[3] sample the curve only in the immediate neighbourhood of known intersecting cells, so they are generally more efficient. However, these methods have one fundamental difficulty besides aliasing: finding a complete set of initial seed cells intersecting every connected component of C in Ω.

Hierarchical decomposition methods[4,5] rely on range analysis[6] to explore Ω recursively in order to discard large portions of Ω quickly and reliably. The classical technique of interval arithmetic (IA)[7] provides a natural tool for range analysis. However, the main weakness of IA is that it tends to be too conservative. To solve this problem, Comba[8] proposed a new model for numerical computation, called affine arithmetic (AA).

Like IA, AA can be used to manipulate imprecise values and to evaluate functions over intervals. It can also keep track of truncation and round-off errors. In contrast to IA, AA also maintains dependencies among the sources of error arising from common variables in different subexpressions, and thus manages to compute significantly tighter error bounds. AA has been used as a replacement for IA in various computer graphical applications, such as ray tracing, intersection testing, enumeration of implicit curves and surfaces, and sampling for procedural shaders[2, 8~10]. However, Comba's simplified formula for affine form multiplication given in Ref. [8] is still conservative. Its range estimate may be four times wider than the exact range. This is not desirable, especially in a chain of calculations needed to evaluate a polynomial. A new more efficient AA technique for polynomial evaluation was proposed in a pre-

* Supported by the China Scholarship Council and the National Natural Science Foundation of China (Grant No. 69973041)

** To whom correspondence should be addressed. E-mail: h.shou@cs.cf.ac.uk

vious paper[11], but it failed to give correct results in certain circumstances.

Thus, this paper proposes a novel matrix formulation using AA for polynomial evaluation, which is mathematically correct and produces better estimation of the range of a bivariate polynomial over a box, because the affine form thus computed is exact. A comparison of the performance and efficiency between IA and matrix AA versions applied to the algebraic curve drawing algorithms is then performed with selected examples.

1 Affine arithmetic bivariate polynomial evaluation in matrix form

In this section we give a novel matrix formulation for bivariate polynomial evaluation using affine arithmetic.

Let

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

where

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

Let us express the intervals $[\underline{x},\bar{x}]$ and $[\underline{y},\bar{y}]$ in affine form:

$$\hat{x}=x_0+x_1\varepsilon_x,\quad \hat{y}=y_0+y_1\varepsilon_y,$$

where ε_x and ε_y are noise symbols whose values are unknown but each is assumed to be in the range $[-1,1]$, and

$$x_0=(\bar{x}+\underline{x})/2,\quad x_1=(\bar{x}-\underline{x})/2,$$
$$y_0=(\bar{y}+\underline{y})/2,\quad y_1=(\bar{y}-\underline{y})/2.$$

Now define power vectors in the error symbols:

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

We now define two further matrices B and C as follows. Let

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

where

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

and let

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

where

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

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

$$D=BAC,$$

we get

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

This is the exact affine form which we now wish to convert back to interval form $[\underline{F},\bar{F}]$. The conversion procedure takes the following form: if i is even and j is also even, then $\varepsilon_x^i\varepsilon_y^j\in[0,1]$; otherwise $\varepsilon_x^i\varepsilon_y^j\in[-1,1]$. So we find that:

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

and

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

The bivariate polynomial $f(x,y)$ thus takes on values guaranteed to be in the interval $[\underline{F},\bar{F}]$ over the box $[\underline{x},\bar{x}]\times[\underline{y},\bar{y}]$.

2 Method and algorithm

We now outline further the hierarchical curve drawing algorithm used in our examples. The basic strategy as presented in Ref. [11] to draw an algebraic curve $f(x,y)=0$ in a given rectangular interval is that we evaluate $f(x,y)$ over the desired interval using IA or AA, starting with the whole interval. If the resulting interval does not contain 0, no curve is 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. In detail, we use the following procedure:

```
PROCEDURE Quadtree(x, x̄, y, ȳ):
  F = IA or AA Evaluation(x, x̄, y, ȳ);
  if F ≤ 0 ≤ F̄ then
    if x̄ − x < 1 AND ȳ − y < 1 then
      PlotPixel(Round((x + x̄)/2), Round((y + ȳ)/2))
    else Subdivide(x, x̄, y, ȳ).

PROCEDURE Subdivide(x, x̄, y, ȳ):
  x̆ = Round((x + x̄)/2);
  y̆ = Round((y + ȳ)/2);
  Quadtree(x, x̆, y, y̆);
  Quadtree(x, x̆, y̆ + 1, ȳ);
  Quadtree(x̆ + 1, x̄, y̆ + 1, ȳ);
  Quadtree(x̆ + 1, x̄, y, y̆).
```

Here $\mathbf{F}$ = IA or AA Evaluation$(\underline{x}, \bar{x}, \underline{y}, \bar{y})$ is the conservative interval containing all values of $f(x, y)$ over $[\underline{x}, \bar{x}] \times [\underline{y}, \bar{y}]$ computed using IA or AA. The pair $(\breve{x}, \breve{y})$ denotes the mid-point of $[\underline{x}, \bar{x}] \times [\underline{y}, \bar{y}]$.

3 Examples

We now illustrate through two examples that affine arithmetic in matrix form usually not only gives much better graphical results but also is more efficient than interval arithmetic. Each example consists in plotting a function $f(x, y) = 0$ using the above algorithm on a grid of 256×256 pixels.

The first example, which is from Ref. [12], is shown in Fig. 1(a) and (b), and is a plot of the curve $0.945xy - 9.43214x^2y^3 + 7.4554x^3y^2 + y^4 - x^3 = 0$ on $[0, 1] \times [0, 1]$ using interval arithmetic and affine arithmetic in matrix form respectively.

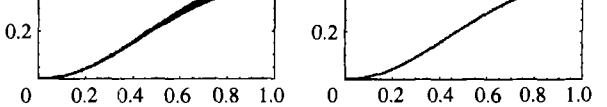

Fig. 1. Drawing curve $0.945xy - 9.43214x^2y^3 + 7.4554x^3y^2 + y^4 - x^3 = 0$. (a) Using IA; (b) using AA.

The second example from Ref. [11] is shown in Fig. 2(a) and (b), and is a plot of $20160x^5 - 30176x^4 + 14156x^3 - 2344x^2 + 151x + 237 - 480y = 0$ on $[0, 1] \times [0, 1]$ using interval arithmetic and affine arithmetic in matrix form respectively.

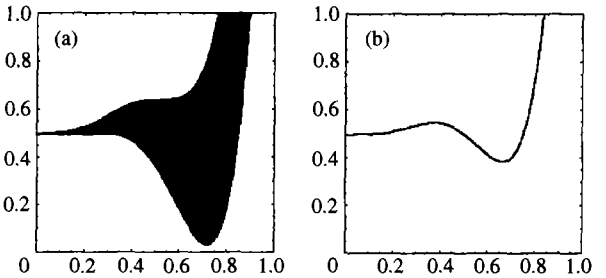

Fig. 2. Drawing curve $20160x^5 - 30176x^4 + 14156x^3 - 2344x^2 + 151x + 237 - 480y = 0$. (a) Using IA; (b) using AA.

To compare the performance and efficiency of our affine arithmetic matrix form bivariate polynomial evaluator against the direct use of interval arithmetic, a number of quantities were measured in both cases (CPU times themselves are not given, as a C language implementation would have been much quicker than the Mathematica implementation we used for testing):

(i) The percentage of the overall area definitely classified as not containing the curve: the bigger the better.

(ii) The number of subdivisions needed: the lower the better, because of overheads incurred in recursion.

(iii) The number of additions (and subtractions), and multiplications needed: the lower the better.

Tables 1 and 2 give the details of these quantities for both test cases.

As can be clearly seen from these figures and tables, the affine arithmetic in matrix form gives much better results than the interval arithmetic in terms of the area correctly classified, the number of subdivisions, and the number of operations needed. These results are usually true for various other examples we tested.

Table 1. Comparison between AA and IA methods for the first example

Method	Area classified (%)	Subdivisions	Additions	Multiplications
AA	99.0723	634	1355919	870189
IA	93.8568	3909	1493798	1876438

Table 2. Comparison between AA and IA methods for the second example

Method	Area classified (%)	Subdivisions	Additions	Multiplications
AA	99.3393	459	667642	510684
IA	73.0225	11458	3485276	4308300

4 Conclusions

Usually affine arithmetic is more complex than interval arithmetic, hence is harder to implement because of the need to keep track of the various sources of error. It might be supposed that affine arithmetic would need more operations during curve drawing. However, the examples show that this is not generally the case because many fewer rectangular regions need to be considered. Due to the tighter intervals it produces, the affine arithmetic method usually not only yields much better graphical results than the interval arithmetic method, but also is more efficient in the number of arithmetic operations required because fewer subdivisions are needed. Expressing the affine operations using matrix forms raises the affine operations to a higher level of abstraction. The small cost to pay for these AA improvements is the addition of extra storage.

Acknowledgement The first author would like to thank the Department of Computer Science of Cardiff University for hosting his one year visit.

References

1 Snyder, J. M. Interval analysis for computer graphics. Computer Graphics (SIGGRAPH'92 Proceedings), 1992, 26(2): 121.
2 de Figueiredo, L. H. et al. Adaptive enumeration of implicit surfaces with affine arithmetic. Computer Graphics Forum, 1996, 15 (5): 287.
3 Chandler, R. E. A tracking algorithm for implicitly defined curves. IEEE Computer Graphics & Applications, 1988, 8(2): 83.
4 Suffern, K. G. Quadtree algorithms for contouring functions of two variables. The Computer Journal, 1990, 33(5): 402.
5 Taubin, G. Rasterizing algebraic curves and surfaces. IEEE Computer Graphics and Applications, 1994, 14(2): 14.
6 Ratschek, H. et al. Computer Methods for the Range of Functions, New York: Ellis Horwood Ltd., 1984.
7 Moore, R. E. Methods and Applications of Interval Analysis, Philadelphia: Society for Industrial and Applied Mathematics, 1979.
8 Comba, J. L. D. et al. Affine arithmetic and its applications to computer graphics. In: Proceedings of Anais do VII SIBGRAPI (Brazilian Symposium on Computer Graphics and Image Processing), Recife, Brazil, 1993, 9.
9 de Figueiredo, L. H. Surface intersection using affine arithmetic. In: Proceedings of Graphics Interface, San Francisco: Morgan Kaufmann publishers, 1996, 168.
10 Heidrich, W. et al. Sampling of procedural shaders using affine arithmetic. ACM Transactions on Graphics, 1998, 17(3): 158.
11 Zhang, Q. et al. Polynomial evaluation using affine arithmetic for curve drawing. In: Proceedings of Eurographics UK 2000, Abingdon: Eurographics UK, 2000, 49.
12 Voiculescu, I. et al. Interval and affine arithmetic for surface location of power-and Bernstein-form polynomials. In: The Mathematics of Surfaces IX (eds. Cipolla, R. et al.), London: Springer-verlag, 2000, 410.