ORCA – Online Research @ Cardiff

Modified Affine Arithmetic in Tensor Form for Trivariate Polynomial Evaluation and Algebraic Surface Plotting [⋆]

Huahao Shou [a,b], Hongwei Lin [a,∗], Ralph R. Martin [c], Guojin Wang [a]

[a] *State Key Lab of CAD & CG, Zhejiang University, Hangzhou 310027, China.*

[b] *Department of Applied Mathematics, Zhejiang University of Technology, Hangzhou 310014, China.*

[c] *School of Computer Science, Cardiff University, Cardiff, CF24 3AA, UK.*

Abstract

This paper extends the *modified affine arithmetic in matrix form* method for bivariate polynomial evaluation and algebraic curve plotting in 2D to *modified affine arithmetic in tensor form* for trivariate polynomial evaluation and algebraic surface plotting in 3D. Experimental comparison shows that modified affine arithmetic in tensor form is not only more accurate but also much faster than standard affine arithmetic when evaluating trivariate polynomials.

Key words: Interval arithmetic, Affine arithmetic, Algebraic surfaces

1 Introduction

Affine arithmetic (AA) was first introduced by Comba and Stolfi in 1993 [4] as an improvement to interval arithmetic (IA). Due to its ability to keep track of correlations between variables in subexpressions, AA is often more resistant to over-conservatism when evaluating estimates of ranges of expressions

⋆ Project supported jointly by the National Natural Science Foundation of China (No. 60373033 & No. 60333010), the National Natural Science Foundation for Innovative Research Groups (No. 60021201) and the Foundation of State Key Basic Research 973 Item (No. 2004CB719400).

∗ Corresponding author.
Email address: `hwlin@cad.zju.edu.cn` (Hongwei Lin).

over intervals. AA has been successfully applied as a replacement for IA in many geometric and computer graphics applications such as surface intersection [6], adaptive enumeration of implicit surfaces [7], ray tracing procedural displacement shaders [9], sampling procedural shaders [10], ray casting implicit surfaces [5], linear interval estimations for parametric objects [3] and a CSG geometric modeller [2].

However standard AA still has an over-conservatism problem because it uses approximation when multiplying affine forms, and thus it can be further improved to give so-called *modified affine arithmetic* (MAA). This uses a matrix form for bivariate polynomial evaluation which keeps all noise terms without any approximation; we have used it for algebraic curve plotting in 2D [11,14]. In this paper we extend MAA to tensor form for trivariate polynomial evaluation, and illustrate its use for algebraic surface plotting in 3D. In a previous paper [13] we proved theoretically that MAA is more accurate than both interval arithmetic on centered form (IAC), and standard AA. Clearly, we also expect MAA in tensor form to also be more efficient than standard AA, and we validate this expectation with an efficiency comparison.

2 Implicit surface plotting algorithm

The quadtree subdivision algorithm described in [11] for plotting an implicit curve $f(x, y) = 0$ in a given rectangle $[\underline{x}, \overline{x}] \times [\underline{y}, \overline{y}]$ can be easily generalized to an octree subdivision algorithm for plotting an implicit surface $f(x, y, z) = 0$ in a given box $[\underline{x}, \overline{x}] \times [\underline{y}, \overline{y}] \times [\underline{z}, \overline{z}]$. The basic idea is to evaluate $f(x, y, z)$ over the desired box using a range analysis estimation method (such as IA, AA or MAA) giving a range $[\underline{F}, \overline{F}]$ which is guaranteed to contain the exact range of f over the box. If the resulting interval does not contain 0, the surface cannot be present. If it does contain 0, we subdivide the box into eight sub-boxes at its mid point, and consider the pieces in turn. The process stops when a box reaches a single voxel in size. In such a case we plot the voxel. This may result in a "thick" surface if the test is too conservative, i.e. voxels may be plotted which do not actually contain the surface.

In a more sophisticated approach, instead of simply plotting the voxel, we may investigate further whether the function actually crosses the voxel, and if so, produce an appropriate linear approximant to the surface within the voxel, for example. In this paper, we concentrate on the problem of finding this set of candidate voxels, rather than on such subsequent processing.

In detail, we use the following procedure for the simple approach:

```
PROCEDURE  Octree($\underline{x}, \overline{x}, \underline{y}, \overline{y}, \underline{z}, \overline{z}$):
  [$\underline{F}, \overline{F}$]=RangeEvaluation($\underline{x}, \overline{x}, \underline{y}, \overline{y}, \underline{z}, \overline{z}$);
  IF $\underline{F} \leq 0 \leq \overline{F}$ THEN
  IF $\overline{x} - \underline{x} \leq$VoxelSize AND $\overline{y} - \underline{y} \leq$VoxelSize
     AND $\overline{z} - \underline{z} \leq$VoxelSize THEN
     PlotVoxel($\underline{x}, \overline{x}, \underline{y}, \overline{y}, \underline{z}, \overline{z}$)
  ELSE Subdivide($\underline{x}, \overline{x}, \underline{y}, \overline{y}, \underline{z}, \overline{z}$).

PROCEDURE  Subdivide($\underline{x}, \overline{x}, \underline{y}, \overline{y}, \underline{z}, \overline{z}$):
  $x_0$=($\underline{x} + \overline{x}$)/2;    $y_0$=($\underline{y} + \overline{y}$)/2;    $z_0$=($\underline{z} + \overline{z}$)/2;
  Octree($\underline{x}, x_0, \underline{y}, y_0, \underline{z}, z_0$);    Octree($x_0, \overline{x}, \underline{y}, y_0, \underline{z}, z_0$);
  Octree($x_0, \overline{x}, y_0, \overline{y}, \underline{z}, z_0$);    Octree($\underline{x}, x_0, y_0, \overline{y}, \underline{z}, z_0$);
  Octree($\underline{x}, x_0, y_0, \overline{y}, z_0, \overline{z}$);    Octree($\underline{x}, x_0, \underline{y}, y_0, z_0, \overline{z}$);
  Octree($x_0, \overline{x}, \underline{y}, y_0, z_0, \overline{z}$);    Octree($x_0, \overline{x}, y_0, \overline{y}, z_0, \overline{z}$).
```

Here $[\underline{F}, \overline{F}]$=RangeEvaluation($\underline{x}, \overline{x}, \underline{y}, \overline{y}, \underline{z}, \overline{z}$) is a conservative interval containing all values of $f(x, y, z)$ over the box $[\underline{x}, \overline{x}] \times [\underline{y}, \overline{y}] \times [\underline{z}, \overline{z}]$, computed using a chosen range analysis method such as IA, AA or MAA. (x_0,y_0,z_0) is the mid-point of the box $[\underline{x}, \overline{x}] \times [\underline{y}, \overline{y}] \times [\underline{z}, \overline{z}]$.

3 Modified Affine Arithmetic in Tensor Form

Let $f(x, y, z)$ be a polynomial in three variables expressed in power form and Ω be a box-shaped interval of interest:

$$f(x, y, z) = \sum_{i=0}^{n} \sum_{j=0}^{m} \sum_{k=0}^{l} A_{ijk} x^i y^j z^k, \quad (x, y, z) \in \Omega = [\underline{x}, \overline{x}] \times [\underline{y}, \overline{y}] \times [\underline{z}, \overline{z}].$$

We rewrite $f(x, y, z)$ in tensor representation:

$$f(x, y, z) = X \otimes_x (Z \otimes_z A) \otimes_y Y,$$

where $X = (1, x, \cdots, x^n), \quad Y = (1, y, \cdots, y^m)^T, \quad Z = (1, z, \cdots, z^l)$ are power vectors, and A_{ijk} is the coefficient tensor.

Let us now convert the interval forms $[\underline{x}, \overline{x}]$, $[\underline{y}, \overline{y}]$ and $[\underline{z}, \overline{z}]$ to affine forms:

$$\hat{x} = x_0 + x_1\varepsilon_x, \quad \hat{y} = y_0 + y_1\varepsilon_y, \quad \hat{z} = z_0 + z_1\varepsilon_z,$$

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

$$x_0 = (\overline{x} + \underline{x})/2, \quad y_0 = (\overline{y} + \underline{y})/2, \quad z_0 = (\overline{z} + \underline{z})/2$$

$$x_1 = (\overline{x} - \underline{x})/2, \quad y_1 = (\overline{y} - \underline{y})/2, \quad z_1 = (\overline{z} - \underline{z})/2.$$

We also define power vectors in the noise symbols:

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

We now define three further matrices B, C and D as follows. Firstly,

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

in detail

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

Secondly,

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

in detail

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

Finally

$$D = \begin{bmatrix} 1 & z_0 & \cdots & z_0^{l-1} & z_0^l \\ 0 & z_1 & \cdots & (l-1)z_0^{l-2}z_1 & lz_0^{l-1}z_1 \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & \cdots & z_1^{l-1} & lz_0 z_1^{l-1} \\ 0 & 0 & \cdots & 0 & z_1^l \end{bmatrix};$$

in detail

$$D_{ij} = \begin{cases} \binom{j}{i} z_0^{j-i} z_1^i, & i \le j \\ 0, & i > j \end{cases}, \quad i = 0, 1, \cdots, l; \quad j = 0, 1, \cdots, l.$$

We now have that

$$X = \hat{X}B, \quad Y = C\hat{Y}, \quad Z = \hat{Z}D.$$

We compute the tensor G from matrices B, C and D, and the original coefficient tensor A as follows:

$$G = B \otimes_x (D \otimes_z A) \otimes_y C,$$

giving

$$f(\hat{x}, \hat{y}, \hat{z}) = \hat{X} \otimes_x (\hat{Z} \otimes_z G) \otimes_y \hat{Y} = \sum_{i=0}^{n} \sum_{j=0}^{m} \sum_{k=0}^{l} G_{ijk} \varepsilon_x^i \varepsilon_y^j \varepsilon_z^k.$$

Up to now the calculation is exact and does not involve any approximation. Furthermore, it can be seen that this polynomial is actually the centered form of the original polynomial [12]. In the next step we wish to convert this result back to interval form $[\underline{F}, \overline{F}]$. This can be done by standard IA. However standard IA can be seen not to be the best choice, if we take into account the following observations: if all of i, j and k are even, then $\varepsilon_x^i \varepsilon_y^j \varepsilon_z^k \in [0, 1]$, otherwise $\varepsilon_x^i \varepsilon_y^j \varepsilon_z^k \in [-1, 1]$. Thus a tighter interval $[\underline{F}, \overline{F}]$ can be obtained, compared to the result given by standard IA on this centered form, as follows

$$\begin{aligned} \overline{F} \quad = \quad G_{000} \quad + \quad & \sum_{k=1}^{l} \left\{ \begin{matrix} \max(0, G_{00k}), \text{ if } k \text{ is even} \\ |G_{00k}|, \qquad \text{otherwise} \end{matrix} \right\} \quad + \\ & \sum_{j=1}^{m} \sum_{k=0}^{l} \left\{ \begin{matrix} \max(0, G_{0jk}), \text{ if } j, k \text{ are both even} \\ |G_{0jk}|, \qquad \text{otherwise} \end{matrix} \right\} \quad + \end{aligned}$$

$$\sum_{i=1}^{n}\sum_{j=0}^{m}\sum_{k=0}^{l}\left\{\begin{array}{ll} \max(0, G_{ijk}), & \text{if } i,j,k \text{ are all even} \\ |G_{ijk}|, & \text{otherwise} \end{array}\right\},$$

and

$$\begin{aligned} \underline{F} &= G_{000} + \sum_{k=1}^{l}\left\{\begin{array}{ll} \min(0, G_{00k}), & \text{if } k \text{ is even} \\ -|G_{00k}|, & \text{otherwise} \end{array}\right\} + \\ &\sum_{j=1}^{m}\sum_{k=0}^{l}\left\{\begin{array}{ll} \min(0, G_{0jk}), & \text{if } j,k \text{ are both even} \\ -|G_{0jk}|, & \text{otherwise} \end{array}\right\} + \\ &\sum_{i=1}^{n}\sum_{j=0}^{m}\sum_{k=0}^{l}\left\{\begin{array}{ll} \min(0, G_{ijk}), & \text{if } i,j,k \text{ are all even} \\ -|G_{ijk}|, & \text{otherwise} \end{array}\right\}. \end{aligned}$$

In this way, the trivariate polynomial $f(x, y, z)$ thus takes on values guaranteed to be in the interval $[\underline{F}, \overline{F}]$ over the box $[\underline{x}, \overline{x}] \times [\underline{y}, \overline{y}] \times [\underline{z}, \overline{z}]$.

4 Comparison of AA and MAA in Tensor Form by Examples

To demonstrate and validate the results in the previous Section, and to compare the relative performance of AA and MAA in tensor form, we now consider some examples. We plot 10 carefully chosen example algebraic surfaces to compare the accuracy and speed of these two methods. Each example surface is represented by a trivariate polynomial equation $f(x, y, z) = 0$; it is plotted using the algorithm given in Section 2 on a grid of $128 \times 128 \times 128$ voxels. We used *Visual C++ 6.0* and *Windows 2000* on a personal computer with an *Intel Xeon* 2.8GHz CPU and 2GB RAM for all the tests.

We not only show the graphical output generated for these examples, but also present a tabular comparison of accuracy and computation times in each case; we also show the results obtained by IAC for comparison. To compare the performance and efficiency of AA, IAC and MAA in tensor form methods, a number of quantities were measured:

- *The number of voxels plotted:* the fewer the better, as plotted voxels may or may not contain the surface in practice.
- *The CPU time used:* the lower the better.
- *The number of subdivisions involved:* the lower the better, due to additional time used for stack overheads.

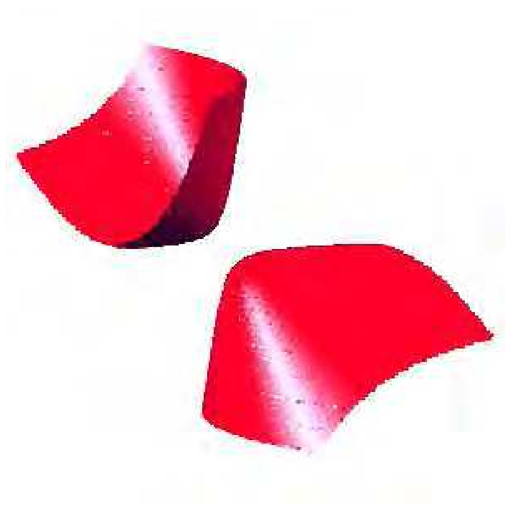
Fig. 1. Example 1. Hyperboloid of two sheets plotted using AA.

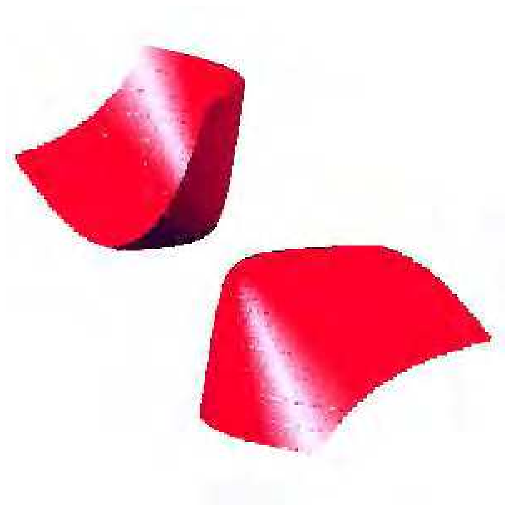
Fig. 2. Example 1. Hyperboloid of two sheets plotted using MAA in tensor form .

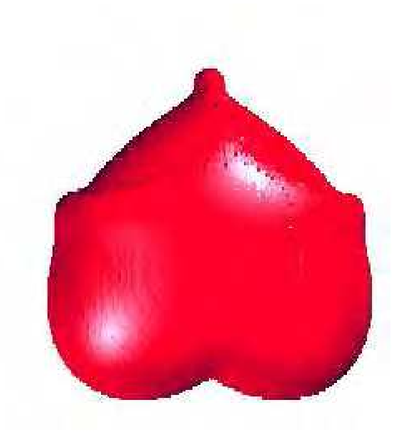
Fig. 3. Example 2. Heart surface plotted using AA.

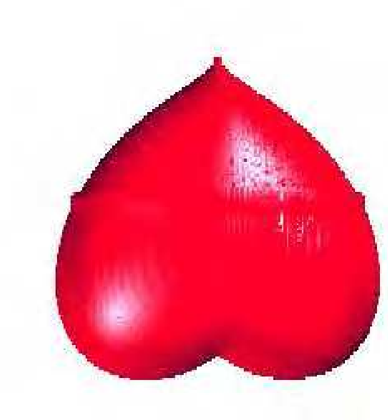
Fig. 4. Example 2. Heart surface plotted using MAA in tensor form.

The examples we used in this paper are as follows:

Example 1: $0.06(x^2 - x + 4y - xy + 2yz + 3) = 0$ on $[-10, 10] \times [-10, 10] \times [-10, 10]$. This is a degree 2 polynomial equation representing a hyperboloid of two sheets, taken from [2].

Example 2: $(2x^2 + y^2 + z^2 - 1)^3 - 0.1x^2z^3 - y^2z^3 = 0$ on $[-1.25, 1.25] \times [-1.25, 1.25] \times [-1.25, 1.25]$. This is a degree 6 polynomial equation representing a heart-shaped surface, taken from [15].

Example 3: $(x^2 + y^2 + z^2 - r^2)^2 - 2(x^2 + r^2)(f^2 + a^2) - 2(y^2 - z^2)(a^2 - f^2) + 8afrx + (a^2 - f^2)^2 = 0$ where $a = 15, r = 3, f = 5$ on $[-20, 20] \times [-20, 20] \times$

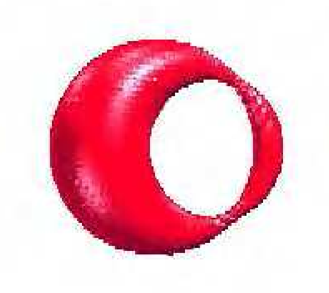

Fig. 5. Example 3. Dupin cyclide plotted using AA.

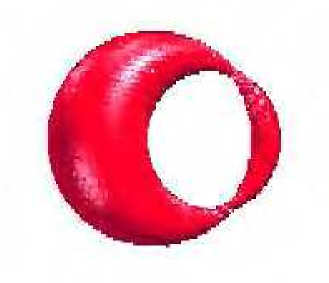

Fig. 6. Example 3. Dupin cyclide plotted using MAA in tensor form.

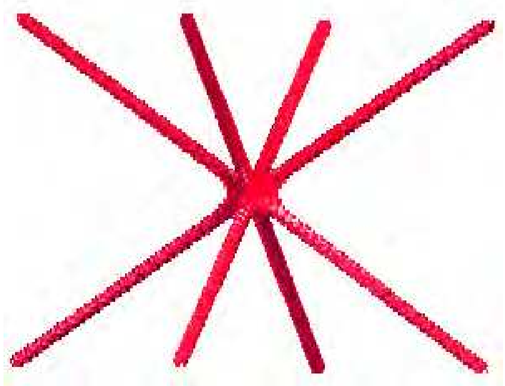

Fig. 7. Example 4. Spiky surface plotted using AA.

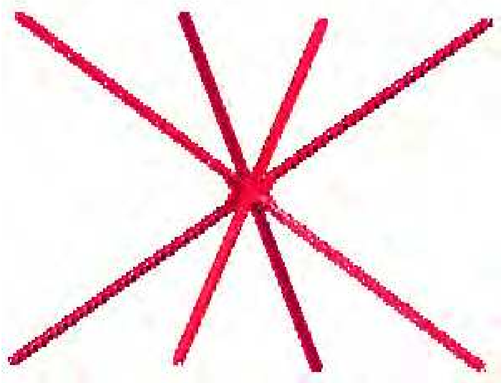

Fig. 8. Example 4. Spiky surface plotted using MAA in tensor form.

$[-20, 20]$. This is a degree 4 polynomial equation representing a Dupin cyclide, taken from [1].

Example 4: $x^2n + y^2n + z^2n - x^ny^n - x^nz^n - y^nz^n = 0$ where $n = 4$ on $[-1.5, 1.5] \times [-1.5, 1.5] \times [-1.5, 1.5]$. This is a degree 8 polynomial equation representing a spiky surface, chosen from [1].

Example 5: $x^2y^2 + y^2z^2 + x^2z^2 + xyz = 0$ on $[-0.5, 0.5] \times [-0.5, 0.5] \times [-0.5, 0.5]$. This is a degree 4 polynomial equation representing Steiner's Roman surface, which has self-intersections, taken from [1].

Example 6: $(x^2 + y^2 - 4)(x^2 + z^2 - 4)(y^2 + z^2 - 4) - 4.0078 = 0$ on $[-6, 6] \times [-6, 6] \times [-6, 6]$. This is a degree 6 polynomial equation representing an implicit

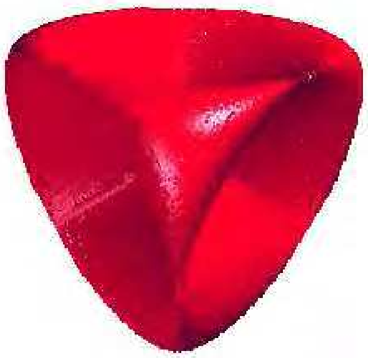

Fig. 9. Example 5. Steiner's Roman surface plotted using AA.

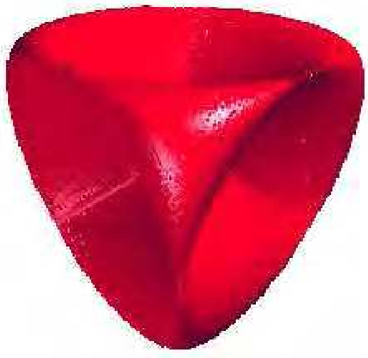

Fig. 10. Example 5. Steiner's Roman surface plotted using MAA in tensor form.

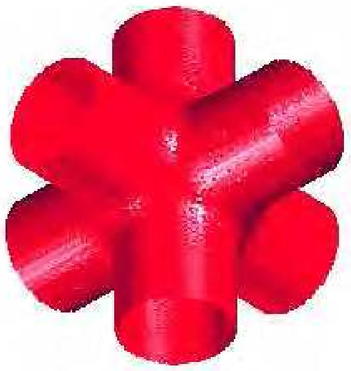

Fig. 11. Example 6. Implicit blending surface plotted using AA.

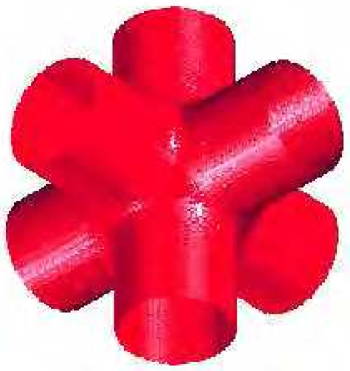

Fig. 12. Example 6. Implicit blending surface plotted using MAA in tensor form.

blending surface, chosen from [1].

Example 7: $(x^4 + y^4 + z^4 + 1) - (x^2 + y^2 + z^2 + y^2z^2 + z^2x^2 + x^2y^2) = 0$ on $[-2, 2] \times [-2, 2] \times [-2, 2]$. This is a degree 4 polynomial equation representing the quadruple Kummer's surface, taken from [8].

Example 8: $z^3 + xz + y = 0$ on $[-5, 5] \times [-5, 5] \times [-5, 5]$. This is a degree 3 polynomial equation representing a cubic cusp catastrophe, taken from [8].

Example 9: $-\frac{1801}{50} + 280x - 816x^2 + 1056x^3 - 512x^4 + \frac{1601}{25}y - 512xy + 1536x^2y - 2048x^3y + 1024x^4y = 0$ on $[0, 1] \times [0, 1] \times [0, 1]$. This is a degree 5 polynomial

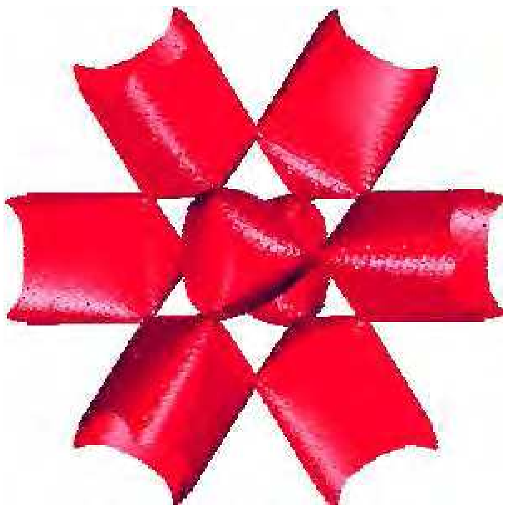
Fig. 13. Example 7. Quadruple Kummer's surface plotted using AA.

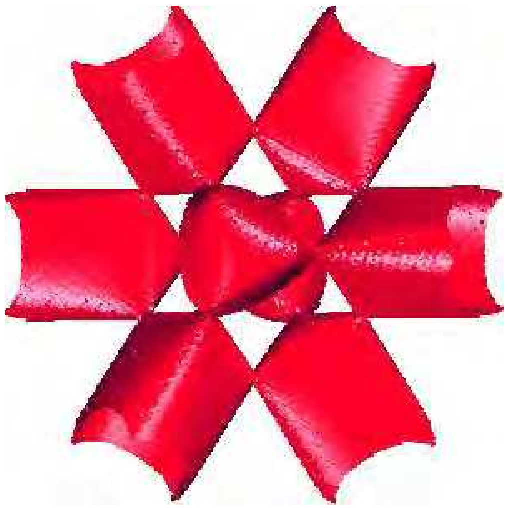
Fig. 14. Example 7. Quadruple Kummer's surface plotted using MAA in tensor form.

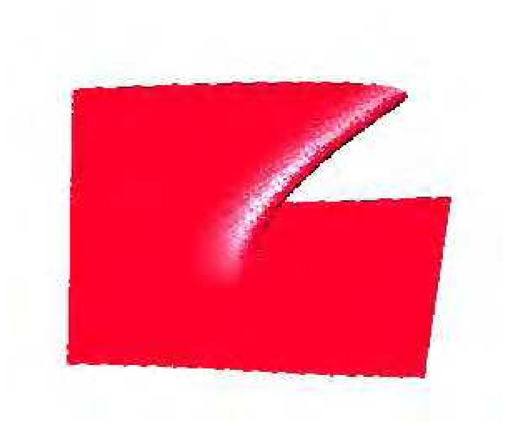
Fig. 15. Example 8. Cubic cusp catastrophe plotted using AA.

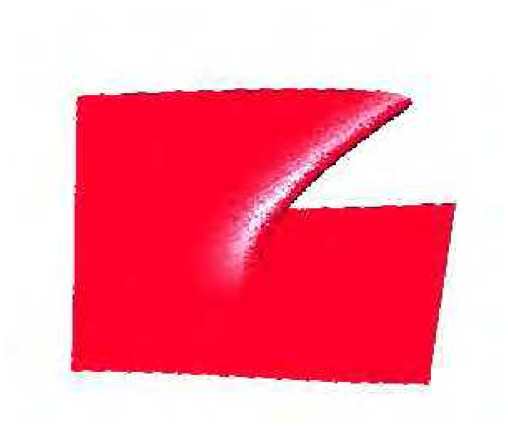
Fig. 16. Example 8. Cubic cusp catastrophe plotted using MAA in tensor form.

equation representing a curve swept along the z direction, taken from [11]. In the latter, the same polynomial equation represents a curve in 2D, but here represents a swept surface in 3D.

Example 10: $\frac{55}{256} - x + 2x^2 - 2x^3 + x^4 - \frac{55}{64}y + 2xy - 2x^2y + \frac{119}{64}y^2 - 2xy^2 + 2x^2y^2 - 2y^3 + y^4 = 0$ on $[0,1] \times [0,1] \times [0,1]$. This is a degree 4 polynomial equation representing a pair of tangent cylinders, chosen from [11]. Again, in the latter, the same polynomial equation represents a curve in 2D, but here represents a swept surface in 3D.

The graphical results for these examples using AA and MAA in tensor form

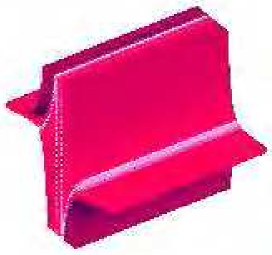
Fig. 17. Example 9. Translational surface plotted using AA.

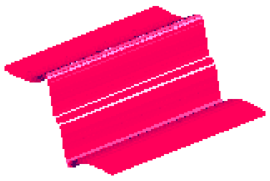
Fig. 18. Example 9. Translational surface plotted using MAA in tensor form.

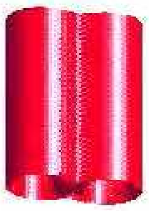
Fig. 19. Example 10. Two tangent cylinders plotted using AA.

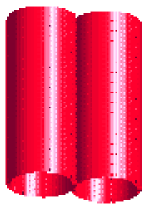
Fig. 20. Example 10. Two tangent cylinders plotted using MAA in tensor form.

methods respectively are shown in Figures 1–20, and the related quantities are recorded in Table 1. From Figures 1–20 and Table 1 we can see that in general the MAA in tensor form method is not only more accurate but also much quicker than the AA method. The AA method is particularly bad on Examples 2, 4, 9, and 10. In Example 9 AA totally fails to reveal the form of the surface while MAA in tensor form successfully reveals it. In Examples 2 and 4 the surfaces generated by AA are much thicker than these generated by MAA in tensor form. In Example 10, AA does badly near the tangency line of contact of the two cylinders, while MAA in tensor form does not have this problem. In terms of performance, AA is always clearly worst in terms of number of subdivisions and computational effort. MAA in tensor form is slightly more accurate than IAC, but MAA in tensor form may or may not take slightly less CPU time than IAC—the times are usually quite similar. Overall, it seems that the performance of MAA in tensor form is slightly better than IAC.

The reasons why MAA in tensor form is much faster than AA can be explained as follows. Firstly, the expressions evaluated by AA involve more arithmetic operations than those needed for IAC or MAA in tensor form. Secondly, AA is more conservative than MAA in tensor form, and therefore needs more subdivisions. Thirdly, MAA in tensor form contains only tensor manipulations, which are essentially loops and are easy to implement. On the other hand, AA requires a complicated data structure involving a dynamic list to represent an affine form and the associated error terms; arithmetic operations on affine forms are performed using insertion and deletion of elements of lists, which are not as efficient as the simple loops needed by MAA in tensor form.

5 Conclusions

MAA in tensor form is not only more accurate but also much faster than standard AA. We have also demonstrated that MAA in tensor form is similar to IAC, but enhanced by a proper consideration of the signs of even or odd powers of polynomial terms. As a result, using MAA in tensor form is always slightly more accurate than IAC, while IAC is always more accurate than AA. In conclusion we recommend that the MAA in tensor form be used instead of AA or IAC in trivariate polynomial geometric computations.

References

[1] R.J. Balsys, K.G. Suffern, Visualisation of implicit surfaces, Computers & Graphics 25 (2001) 89–107.

[2] A. Bowyer, R. Martin, H. Shou, I. Voiculescu, Affine intervals in a CSG geometric modeller, in: J. Winkler, M. Niranjan (Eds.), Uncertainty in Geometric Computations, 2002, pp. 1-14.

[3] K. Bühler, Linear interval estimations for parametric objects theory and application, Computer Graphics Forum 20(3) (2001) 522–531.

[4] J.L.D. Comba, J. Stolfi, Affine arithmetic and its applications to computer graphics, Anais do VII SIBGRAPI, 1993, pp. 9–18. Available at http://www.dcc.unicamp.br/∼stolfi/.

[5] A.Jr. De Cusatis, L.H. De Figueiredo, M. Gattass, Interval Methods for Ray Casting Implicit Surfaces with Affine Arithmetic, XII Brazilian Symposium on Computer Graphics and Image Processing, 1999, pp. 65–71.

Table 1
Comparison of AA, IAC and MAA in tensor form for each example

Examples	Methods	Voxels plotted	Subdivisions involved	CPU time used
1	AA	39305	13440	167 sec
1	IAC	39305	13440	51 sec
1	MAA	39214	13343	52 sec
2	AA	143104	61901	2734 sec
2	IAC	63168	31669	147 sec
2	MAA	59104	28333	136 sec
3	AA	16716	7049	41 sec
3	IAC	16420	6881	10 sec
3	MAA	15956	6357	10 sec
4	AA	17480	17953	73 sec
4	IAC	11792	14665	15 sec
4	MAA	10256	10681	12 sec
5	AA	89396	34009	920 sec
5	IAC	86864	31897	255 sec
5	MAA	85448	31033	262 sec
6	AA	61512	35873	530 sec
6	IAC	53576	26017	101 sec
6	MAA	52544	24337	104 sec
7	AA	114320	42129	1608 sec
7	IAC	111536	40289	423 sec
7	MAA	109712	39209	428 sec
8	AA	33982	12063	174 sec
8	IAC	33982	12063	39 sec
8	MAA	33666	11683	40 sec
9	AA	748032	163881	61876 sec
9	IAC	32000	13673	35 sec
9	MAA	31744	12521	36 sec
10	AA	163072	69281	3250 sec
10	IAC	53248	20361	97 sec
10	MAA	50176	18601	89 sec

[6] L.H. De Figueiredo, Surface intersection using affine arithmetic, Proceedings of Graphics Interface, 1996, pp. 168–175.

[7] L.H. De Figueiredo, J. Stolfi, Adaptive enumeration of implicit surfaces with affine arithmetic, Computer Graphics Forum 15(5) (1996) 287–296.

[8] P. Hanrahan, Ray tracing algebraic surfaces, Computer Graphics 17(3) (1983) 83–90.

[9] W. Heidrich, H.P. Seidel, Ray tracing procedural displacement shaders, Proceedings of Graphics Interface, 1998, 8–16.

[10] W. Heidrich, P. Slusallek, H.P. Seidel, Sampling of procedural shaders using affine arithmetic, ACM Trans. on Graphics 17(3) (1998) 158–176.

[11] R. Martin, H. Shou, I. Voiculescu, A. Bowyer, G. Wang, Comparison of interval methods for plotting algebraic curves, Computer Aided Geometric Design 19(7) (2002) 553–587.

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

[13] H. Shou, H. Lin, R. Martin, G. Wang, Modified affine arithmetic is more accurate than centered interval arithmetic or affine arithmetic, in: M.J. Wilson, R.R. Martin (Eds.), Lecture Notes in Computer Science 2768, Mathematics of Surfaces, Springer-Verlag Berlin Heidelberg New York, 2003, pp. 355-365.

[14] H. Shou, R. Martin, I. Voiculescu, A. Bowyer, G. Wang, Affine arithmetic in matrix form for polynomial evaluation and algebraic curve drawing, Progress in Natural Science 12(1) (2002) 77–80.

[15] G. Taubin, Rasterizing algebraic curves and surfaces, IEEE Computer Graphics Appl. 14 (1994) 14–23.