数式処理 *J.JSSAC (1998)*
Vol. 6, No. 2, pp. 2 - 21

論文

Displaying Real Solution of Mathematical Equations

齋藤友克　近藤祐史　三好善彦　竹島卓

上智大学 *　詫間電波高専 †　埼玉女子短期大学‡　富士通研究所 §

(Received 1997/3/27　Revised 1997/12/17)

Abstract. Displaying the figure of real zeros of a function is a classical but not easy problem. It is still very hard if we want to obtain the real solution curves of a general bivariate function exactly. In this paper we propose a new concept for displaying real solution of mathematical equations. We define some display functions which have concrete mathematical background. We construct new algorithms for the displaying problem. The presented algorithms are implemented as alternatives of the implicit function plotting package for Risa/Asir, a non-commercial computer algebra system developed at Fujitsu Labs.

1 序言

数学的に定義された関数の形状を忠実に表示する問題は、古くから存在しかつその困難さが軽視されてきた問題である [5]。ここで関数の形状とは、関数の実零点の分布を言う。一般的な関数の形状表示は、例えば 2 変数関数 $f(x,y)=0$ の場合、一般的にまず関数を各枝ごとに $y=g(x)$ の形態に変形する。次に関数上の幾つかの実零点の xy 座標を関数の陽的な計算により求める。得られたそれらの点を結ぶ。このように、グラフ表示の問題は比較的安易な考えによりアルゴリズムが構築されてきた。この場合、表示のアルゴリズムを主たる問題とするのではなく、数値計算をいかに軽微にかつ精度良く計算するかということが主要問題として認識されてきた。

我々はこのような考え方では次の問題点が軽視されていると考えると。

*saito@mm.sophia.ac.jp
†kondoh@dc.takuma-ct.ac.jp
‡miyoshi@saitama.email.ne.jp
§tak@para.flab.fujitsu.co.jp

- グラフ上の点は理想的な理論上の連続均質な空間の中にある。一方それを表示する媒体は現実の３次元空間の中にある物理的な実体空間である。よってグラフの表示は表示媒体の持つ有限性、離散性等の制約を免れない。数学的性質の異なるこれらの空間の間のマッピングは慎重に取り扱われるべきである。

  このことを具体的に言うと、関数の実零点は理論上は大きさや幅を持たない。しかし「点」や「線」を表示するには、それらは有限の大きさや幅を持つ必要がある。

本論文においては、関数の形状を表示することは如何なることであるか、基本概念を提案し、その概念にしたがって関数を表示するにはどのような方法が存在するかを示す。

1.1 現状の研究状況

現在使用されている関数の形状を表示するアルゴリズムとして我々の観点からは、2 つの異なる方法 Draw Method と Plot Method が存在する。現在までに主に研究されてきた関数表示アルゴリズムは、Draw Method にほとんど限られている。以下にその概略ならびに利害得失を述べる。

1.1.1 Draw Method

Draw Method とは、表示する関数の実零点の中から幾つかの代表点の座標を求める。それらの代表点の座標に基づき、一定の接続アルゴリズムによって点を接続していく方法である。最も簡単なアルゴリズムとしては、各点の間を直線で結ぶ直線近似の手法がある。より進んだアルゴリズムとしては、所与の関係式から微分方程式等を作り、接続する次の代表点の予測をする方法等のアルゴリズムが存在する。これらの計算はまず理想的な数学的空間の上で行なわれ、ついで物理媒体上にマッピングされることが特徴である。参考文献としては、[2, 3, 4, 10, 11, 14] などがある。

Draw Method の利点としては、比較的実装が容易であることである。複雑な接続アルゴリズムを使用しない場合、かつ、陽的な計算で済む場合には計算量的に軽い手法である。現在実際に使われているほとんどの関数表示アルゴリズムは、Draw Method を採用している。

Draw Method の欠点としては、グラフ上の幾つかの実零点の座標をいかに精密に求めても、得られた点の間を一定のアルゴリズムにより補間し接続することになり、常に近似的な表示にならざるを得ないことである。また、精密に求める点の密度はできるだけ一様であることが望ましいが、そのコントロールが困難であることである。更に特異点の付近での描画や微細構造の描画を要求された場合、計算量の増加を許容したとしても描画が非常に困難を究め、そのためのアルゴリズムが複雑化することである。グラフ上の点 (一般には非有理点)

を計算するには浮動小数点数による計算しか実用上の手段がないことなどが挙げられる。

このような特性から、Draw Method の研究主題はいかに軽いアルゴリズムによってより精密な関数近似をするかに主眼が置かれている。

1.1.2 Plot Method

Plot Method は、空間上の定められた領域を直接与え、その領域を変数とする指標関数の値により、表示すべきグラフ上の点か否かを決定する方法である。この方法は計算量が大きいと予想されたのみならず、数学的に解の存在範囲を確定することが困難であると考えられていたことから従来顧みられなかった方法である。

この方法の利点は、確実に (すなわち有限的有理的な方法で) 指標関数の関数値を求めることができるのならば、表示された図形がある絶対誤差の範囲で常に正しいことが保証される点である。Draw Method が実用的でない特異点の近傍や、微細構造に対しても安定に表示できることである。

欠点としては、計算量的に多大な計算を必要とする点である。さらに、この方法による指標関数の数学的な理論解析がいまだ不十分であるため、厳密な正確さを要求した場合に扱える関数が代数関数に限られることも欠点として挙げられる。

1.2 目的

本論文は、今まで関数表示アルゴリズムとしてあまり考慮されていない Plot Method の概念を明確にし、研究の出発点を与えることを主目的としている。

2 Plot Mehod における基本概念

関数を表示するとは、実際問題としては、ディスプレイやプリンタなどの物理媒体に何らかの意味を持った出力を与えることである。つまり、与えられた関数 $f(x_1, \cdots, x_n) = 0$ の実零点の像を物理媒体上に射影することである。このことを考慮すると、『点を表示』するときの『点』は数学の定議する『点』とは異なり常に大きさを持つ。以下混乱が生じないよう点を Cell と呼ぶことにする。通常 Cell の大きさは、出力の解像度と呼ばれるものである。正確な Cell の定義は後程与えるが、 Cell が必要とする基本的な概念および性質を以下に示す。

2.1 Cell に対する要請

表示空間は Cell (微小であるかどうかは表示の求める必要性により決定される) により覆い尽くされていると考えることは自然である。よって次の基本概念が要請される。

要請 1

- Cell は大きさを持つ。
- 表現空間はこの Cell により覆い尽くされている。

ここで大きさとは、物理的なボリュームを持つことである。厳密な定義は後程述べる。また表現空間が Cell により覆い尽くされていなければ表示に穴が空くことになり、特に必要性が無い限りこの要請を外すことは不自然である。

2.2 Cell の選択に対する要請

一般に関数の実零点を表示するということを、『関数の実零点を含む Cell を求めること』と考えることは自然な発想である。しかし、我々はその概念の逆の立場である『関数の実零点を含まない Cell を除外する』という概念から出発し論理を構成すべきであると考える。その理由としては、実零点を含まない可能性をコントロールすることにより指標関数の強弱を得ることができることである。指標関数の強弱は、計算量の増減をもたらす。よって必要に応じた計算量の指標関数を構築できる。言い替えるとこの方法は、関数の実零点を表示するのではなく、実零点を含まない Cell を除外し表示空間から除去して行く方法である。

そこで、表示されている図が与えられた関数の零点であることを最低限に保証する概念として、次の要請は必要である。

要請 2

- 関数を表示するとは、Cell を選択することである。
- 選択されない Cell は関数の実零点と共通部分を持たない。

3 指標関数の定義および性質

要請を実現するための必要な定義を示す。以下取り扱う関数 $f(x_1, \cdots, x_n) = 0$ は $\mathbb{R}^n$ の連結集合 $\mathcal{D}$ 上で定義されている連続関数とする。また表示する領域 D は、$\mathcal{D}$ の部分集合であって一般的な距離位相のもとで閉集合かつ Jordan 測度が 0 でないとする。

3.1 基本定義

定義 1 (Cell の定義)
$\mathcal{C}_k$ $(k = 1, \cdots, m)$ が D 上定義された *Cell* であるとは、

1. $D = \bigcup_{k=1}^{m} \mathcal{C}_k, \qquad \mathcal{C}_k^i \bigcap \mathcal{C}_j^i = \phi, \qquad k \neq j,$

2. 各 $\mathcal{C}_k$ の *Jordan* 測度はゼロではない。

とする。ここで $\mathcal{C}_k^i$ は、$\mathcal{C}_k$ の内点の全体とする。

定義 2 (指標関数の定義)
D 上定義されている関数 f の *Cell* の族 $\{\mathcal{C}_k\}$ による指標関数 χ とは、

1. $\chi : \{\mathcal{C}_k\} \to \{0, 1\}$

2. $\chi(\mathcal{C}_k) = 0$ ならば $\mathcal{C}_k$ の任意の元 x に対し $f(x) \neq 0$

とする。

Cell の大きさとは、$\mathcal{C}_k$ の Jordan 測度のことである (要請 1)。関数を表示するとは、この定義域において定義される部分集合の族 $\{\mathcal{C}_k\}$ とその上で定義される指標関数 χ を求めることである。

この指標関数の定義によれば、$\chi(\mathcal{C}_k)$ が値 0 を持てば、与えられた関数 f はその領域において零点を持たないことを保証している (要請 2 を定式化している)。

指標関数間の関係を記述するため順序関係を定義する。

定義 3 (順序の定義)
χ_1, χ_2 を関数 $f(x_1, \cdots, x_n) = 0$ に対する *Cell* の族 $\{\mathcal{C}_k\}$ 上の指標関数とする。このとき $\chi_1 \succ \chi_2$ であるとは、

$$\chi_2(\mathcal{C}_i) = 1 \qquad \text{ならば} \qquad \chi_1(\mathcal{C}_i) = 1$$

が成り立つことである。この順序を指標関数の順序とする。

3.2 指標関数の性質

以下前節で定義された指標関数の諸性質を述べる。

補題 4 (順序の公理)
指標関数の順序は順序の公理系を満たす。

補題 5
関数 f に関する *Cell* 族 $\{\mathcal{C}_k\}$ 上の任意の指標関数 χ_1, χ_2 に対し、次のような指標関数 χ_u, χ_b が存在する。

1. $\chi_1 \succ \chi_b$ かつ $\chi_2 \succ \chi_b$

2. $\chi_1 \prec \chi_u$ かつ $\chi_2 \prec \chi_u$

証明

次のような関数 $\chi_1 \cap \chi_2$ と $\chi_1 \cup \chi_2$ を考える。

$$\begin{aligned}(\chi_1 \cap \chi_2)(\mathcal{C}_i) &= \chi_1(\mathcal{C}_i)\chi_2(\mathcal{C}_i) \\ (\chi_1 \cup \chi_2)(\mathcal{C}_i) &= \begin{cases} 1 & \chi_1(\mathcal{C}_i) = 1 \text{ または } \chi_2(\mathcal{C}_i) = 1 \\ 0 & otherwise \end{cases}\end{aligned}$$

$(\chi_1 \cap \chi_2)(\mathcal{C}_i) = 0$, であるとは $\chi_1(\mathcal{C}_i) = 0$ であるか $\chi_2(\mathcal{C}_i) = 0$ である。よって χ_1 と χ_2 がともに指標関数であることより、$\mathcal{C}_i$ 内に $f = 0$ となる解は存在しない。故に $\chi_1 \cap \chi_2$ は、関数 f の Cell 族 $\{\mathcal{C}_i\}$ の指標関数である。

また、

$$\chi_1 \succ \chi_1 \cap \chi_2 \text{ かつ } \chi_2 \succ \chi_1 \cap \chi_2$$

は構成法から明らかである。よって、$\chi_1 \cap \chi_2$ は χ_b の一つである。

$\chi_1 \cup \chi_2$ も同様に f に関する族 $\{\mathcal{C}_k\}$ 上の指標関数となることは容易に示すことができる。$\chi_1 \cup \chi_2$ が χ_u の一つとなる。 ∎

系 6 (最小上界、最大下界)

$\chi_1 \cap \chi_2$, $\chi_1 \cup \chi_2$ はそれぞれ χ_1 と χ_2 の最大下界、最小上界である。

証明

χ_1 と χ_2 の共通下界として χ を考える。このとき $\chi_1 \cap \chi_2 \succ \chi$ を示せば良い。そこで $\chi_1 \cap \chi_2 \not\succ \chi$ と仮定すると $\chi(\mathcal{C}_i) = 1$ かつ $(\chi_1 \cap \chi_2)(\mathcal{C}_i) = 0$ となる $\mathcal{C}_i$ が存在する。χ が χ_1 と χ_2 の共通下界であることより $\chi_1(\mathcal{C}_i) = \chi_2(\mathcal{C}_2) = 1$ が成り立つ。これは $\chi_1 \cap \chi_2$ の構成法から矛盾である。

$\chi_1 \cup \chi_2$ の場合も同様に示すことができる。 ∎

定理 7 (指標関数のなす束)

$\{\chi_i\}$ を *Cell* の族 $\{\mathcal{C}_k\}$ 上の指標関数全体とする。このとき指標関数全体は、指標関数の順序によって束をなす。

証明

補題 5 より指標関数全体は、束の公理系を満たすことは明らかである。 ∎

定理 8 (最大最小元の存在)

関数 $f(x_1, \cdots, x_n) = 0$ に関する *Cell* の族 $\{\mathcal{C}_k\}$ 上の全ての指標関数に対し、最大元と最小元がただ一つ存在する。

証明

指標関数の全体は有限個であり、かつ束をなすことより最大最小元が存在することは明らかである。 ∎

4 代表的な指標関数

4.1 Faithful Character

前節で示したことをまとめると次の定理が得られる。

定理 9 (Faithful Character)
D 上定義されている関数 f の *Cell* の族 $\{\mathcal{C}_k\}$ による指標関数 χ が最小元となる必要十分条件は $\chi(\mathcal{C}_k) = 1$ ならば、

$$\mathcal{C}_k \text{ に含まれるある元 } x \text{ が存在し } f(x) = 0 \text{ である。}$$

この最小指標関数 χ を *Faithful Character* と呼ぶ。

一般の関数に対し Faithful Character を求めるアルゴリズムは現在知られていない。また Faithful Character アルゴリズムに関する基礎的な理論は、有理係数 2 変数代数関数に関してのみである [12] 。

このことより実用上必要なことは、どの様な指標関数が目的に対し有効かつ実用的かという問題に帰着する。関数を表示するとは、有効な指標関数を定め計算アルゴリズムを決定し、かつ計算することが関数表示問題と考えることができる。

4.2 Interval Character

Interval Character [6, 7] は、指標関数を計算する場合 Cell を区間数として扱うことを特徴とする。以下、次の仮定を追加する。

- Cell $\mathcal{C}_k$ は境界を含む矩形領域とする。

よって $\mathcal{C}_k$ は、

$$\mathcal{C}_k = \{(x_1, \cdots, x_n) | \quad a_{k,l} \leq x_l \leq b_{k,l}, \quad 1 \leq l \leq n\}$$

と表すことができる。また区間数 $I_{k,l}$ を $[a_{k,l}, b_{k,l}]$ とする。

定義 10 (Interval Character)
χ が *Interval Character* であるとは、

1. $\chi : \{\mathcal{C}_k\} \to \{0, 1\}$
2. $\chi(\mathcal{C}_k) = \begin{cases} 1 & f(I_{k,1}, \cdots, I_{k,n}) \ni 0 \\ 0 & f(I_{k,1}, \cdots, I_{k,n}) \not\ni 0 \end{cases}$

と定義する。ここで $f(I_{k,1}, \cdots, I_{k,n})$ は、関数を区間数 $I_{k,i}$ $(i = 1, \cdots, n)$ を用いて区間数演算 *[1]* をすることにより得られた区間数である。

この Interval Character 計算アルゴリズムは、演算を全て区間数として取り扱うため、区間演算の性質により Cell $\mathcal{C}_k$ の中に関数 $f = 0$ の零点が含まれていれば、必ず $\chi(\mathcal{C}_k)$ は 1 である (零点の存在を見逃すことはない)。 よって χ は、Cell の族 $\{\mathcal{C}_k\}$ 上の f の指標関数であることは容易に検証できる [1]。

しかし、区間演算の性質である区間膨張のため解が存在しない Cell であっても表示することがある。

利点としては、他の知られている指標関数計算アルゴリズムとは異なり一般の n 変数連続関数に対して計算できる点である。しかし、 Interval Character は、区間演算の特徴 (区間膨張) より解が存在しない Cell に対し実零点があるかどうか判定不能、つまり表示されてしまう性質がある。(図 2 における塗りつぶされた領域)。この黒く塗りつぶされた領域は、実零点が含まれていないかも知れない領域である。しかし実零点を含む可能性がある領域はこれ以外に存在しないことを示している。また、区間演算の区間を縮小することによりアルゴリズムをなんら改良すること無く必要な精度で関数の解を求め表示できる点は有用と考えられる。

4.2.1 実装

次の関数 $\mathrm{Knot}(x, y) = 0$ と $\mathrm{Heart}(x, y) = 0$ に対して Interval Character を適用した実例をそれぞれ図 1,2 (大きさは印刷の都合上縦横 0.4 倍してある。) に示す。ここで D は、$(2, 2)$, $(-2, 2)$, $(-2, -2)$, $(2, -2)$ によって囲まれた正方形領域である。さらに各 Cell は縦横 1/600 の大きさの正方形である。

$$
\begin{aligned}
\mathrm{Knot}(x, y) &= x^5 - 2x^2y + y^5 \\
\mathrm{Heart}(x, y) &= \tfrac{93392896}{15625}x^6 + (\tfrac{94359552}{625}y^2 + \tfrac{91521024}{625}y - \tfrac{249088}{125})x^4 \\
&\quad + (\tfrac{1032192}{25}y^4 - 36864y^3 - \tfrac{7732224}{25}y^2 - 207360y + \tfrac{770048}{25})x^2 \\
&\quad + 65536y^6 + 49152y^5 - 135168y^4 - 72704y^3 \\
&\quad + 101376y^2 + 27648y - 27648
\end{aligned}
$$

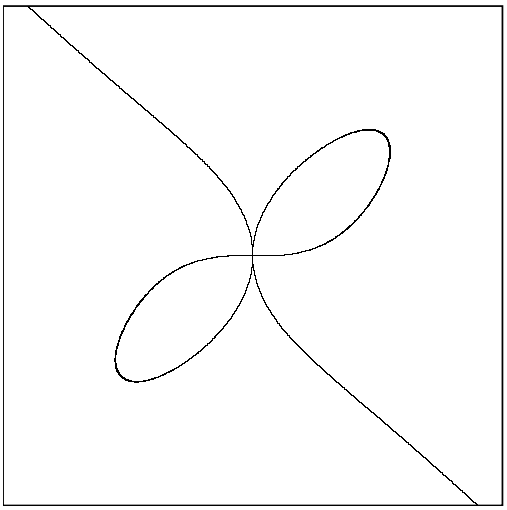

図 1 Interval Character による
Knot$(x, y) = 0$
解像度 1/600

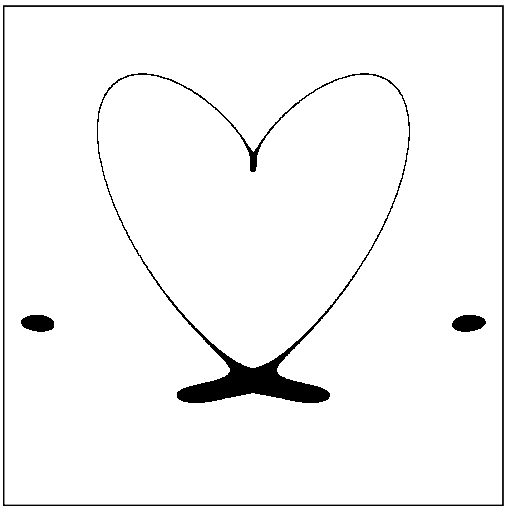

図 2 Interval Character による
Heart$(x, y) = 0$
解像度 1/600

特に Heart$(x, y) = 0$ は $(\pm 35\sqrt{17}/351, -386/351)$ と $(\pm 35\sqrt{14}/76, -41/76)$ に実孤立特異点が存在するため図 2 中の黒い領域が出現する。この Character は、Risa/Asir [8] の上のユーザー言語をもちい記述されている。

5 Weak Character

Faithful Character を一般的に求めることができないとすれば、次善の方式として指標関数の条件を緩和して計算できる弱い指標関数 (Weak Character) を構成する問題が発生する。それは、どのような指標関数が有用かという問題に帰着する。弱い Character として領域の次元を下げた Boundary Character ともっぱら実用性を重視した Sign Weak Character を提案する。Sign Weak Character は、Character と呼ぶには**要請 2** に抵触し問題があるが実用性を考えると多項式以外に拡張が容易である点を考慮して構成されている。

5.1 Boundary Character

Weak Character の事例として Cell の境界のみに注目した Boundary Character を次のように定める。

定義 11 (Boundary Character)
D 上の *Cell* 族 $\{\mathcal{C}_k\}$ による関数 $f(x, y) = 0$ に対する *Boundary Character* β とは、

1. $\beta : \{\mathcal{C}_k\} \to \{0, 1\}$
2. $\beta(\mathcal{C}_k) = \begin{cases} 1 & \{(x, y) \in \mathcal{C}_k | f(x, y) = 0\} \cap \partial\mathcal{C}_k \neq \phi \\ 0 & otherwise \end{cases}$

この Character は、領域 D を $\{\mathcal{C}_k\}$ の境界領域に制限した場合の Faithfule Character となっている。Boundary Character は、Cell の形状が任意の場合計算するアルゴリズムは知られていない。しかし、次のような条件を付ければ求めることができる [13]。

定理 12
次の条件 [1)] を満足するならば、*Boundary Character* は有限手順で計算できる。

1. 各 *Cell* の境界は直線である。
2. 各 *Cell* の境界線分の端点は代数的数である。
3. 表示する関数は *2* 変数有理係数代数関数である。

証明
個々の Cell に関して求められることを示せれば良いから Cell 族に含まれる Cell は 1 つだけと仮定することができる。この Cell を

$$\mathcal{C} = \{(x, y) | \quad a \leq x \leq b, \quad c \leq y \leq d\}$$

とする。この Cell の境界線 (a, c)–(b, c), (b, c)–(b, d), (b, d)–(a, d), (a, d)–(a, c) に対し Sturm 列を求め解の存在を判定する。 ∎

5.1.1 実装

このアルゴリズムを実装した描画機能は、現在 Risa/Asir [8] の ifplot precise モードとして実装されている。実例として $\mathrm{Knot}(x, y) = 0$ と $\mathrm{Heart}(x, y) = 0$ の図形を Boundary Character で求めたものをそれぞれ図 3,5,4,6 (各々の図は実際の図を 0.4 倍したもの) に示す。このときの Cell の大きさは各辺の長さ 1/100 または 1/600 表示領域の範囲は $(2, 2)$, $(-2, 2)$, $(-2, -2)$, $(2, -2)$ で囲まれた範囲である。また得られた図形は、関数の構造を別途理論的に解析することにより孤立特異点を除いて Faithful Character と同一な図形であることは容易に証明することはできる。

タイミングデータとしては、OS として FreeBSD 計算機として Intel Pentium Pro 200MH による計算時間を表 1 に示す。

[1)] 厳密にいうと各境界が直線である必要性はなく、各境界が代数関数の零点で囲まれていればよい。しかし、計算量的かつ現実的な方法としては、Cell は矩形領域とするという仮定は不自然ではない。また端点が代数的数で無い場合有限手順で計算するアルゴリズムは知られていない。

表 1 Boundary Character の計算時間

	1/100	1/600
Heart	5.129 秒	46.59 秒
Knot	2.103 秒	22.96 秒

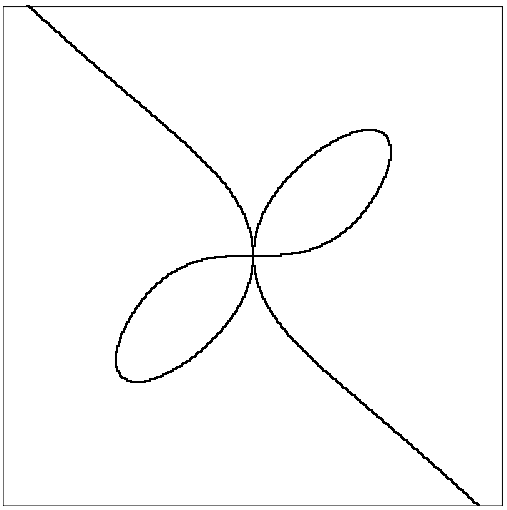

図 3 Boundary Character による
$\mathrm{Knot}(x, y) = 0$
解像度 1/100

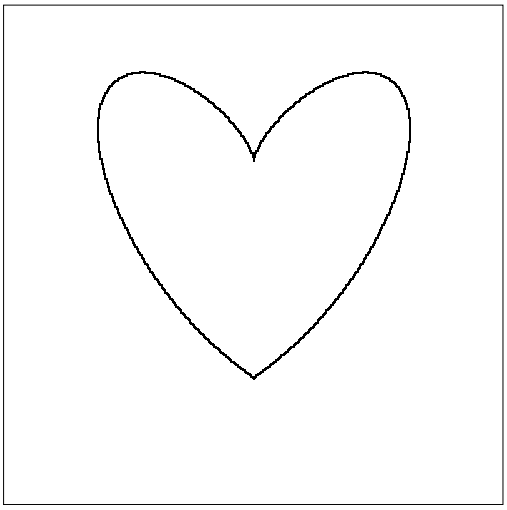

図 4 Boundary Character による
$\mathrm{Heart}(x, y) = 0$
解像度 1/100

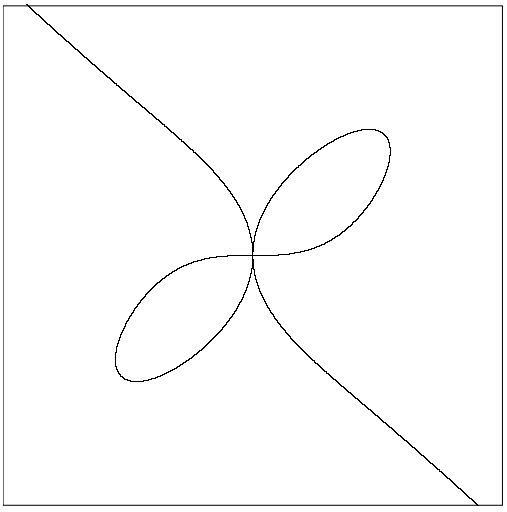

図 5 Boundary Character による
$\mathrm{Knot}(x, y) = 0$
解像度 1/600

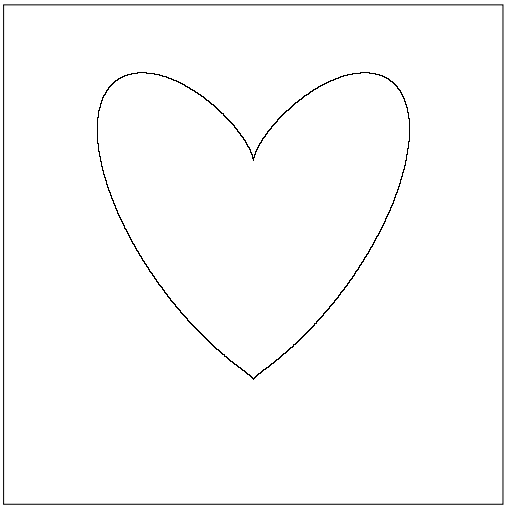

図 6 Boundary Character による
$\mathrm{Heart}(x, y) = 0$
解像度 1/600

5.1.2 Boundary Character の利点と問題点

Boundary Character を用いた場合、1 つの Cell の中に完全に含まれる閉曲線の存在は検出することができない。また、孤立特異点も運良く境界線上に存在しない限り検出することができない。しかし、Cell よりも大きな代数曲線は確実に表示することが可能である。

また、アルゴリズムの実装の工夫として、領域 D を直線で矩形状に区切り各々の直線に対し多項式剰余列を 1 度構成し、解の存在範囲の確定するための sturum 列の計算を 2 分探査で行う事により、各 Cell ごとに解の存在を判定する必要がないため、目的によっては実用に耐え得るものと考える。

欠点としては、常に代数的に計算するため係数は有理数もしくは代数的数である必要がある。よって浮動小数点数方式で得られた代数関数の表示をするためには、有理数近似をする必要がある点である。

5.2 Sign Weak Character

Boundary Character による関数の実零点の表示は適切なものと考えられるが、手軽に表示を求めるには計算量的に大きすぎる点を改良する必要がある。そのため実用的な表示の質を保ちながらより速いアルゴリズム Sign Weak Character [13] を提案する。その手法として要請 2 の後半を適用をしない Weak Character を採用した。

各 Cell は Interval Character と同様とする。

$$\mathcal{C}_k = \{(x_1, \cdots, x_n) | \quad a_{k,l} \leq x_l \leq b_{k,l}, \quad 1 \leq l \leq n\}$$

指標関数の $0, 1$ 判定に Boundary Character は Sturm 列を利用したが Sign Weak Character は、中間値の定理を利用する。

定義 13 (Sign Weak Character)
D 上の *Cell* 族 $\{\mathcal{C}_k\}$ による関数 $f(x_1, \cdots, x_n) = 0$ に対する *Sign Weak Character* σ とは、

1. $\sigma : \{\mathcal{C}_k\} \to \{0, 1\}$
2. $\sigma(\mathcal{C}_k) = \begin{cases} 1 & \mathcal{C}_k \text{の } 2^n \text{ 個の端点の関数値の符号が異なっている} \\ 0 & \textit{otherwise} \end{cases}$

この Character は、連続関数の符号が一定の区間で異なれば、その区間内に関数の零点が存在することを利用している。特徴としては、中間値の定理を素朴に利用するため関数値の変化や係数の誤差に対し丈夫であり、アルゴリズムとしては強固である。特に孤立特異点や小さな解領域を持たないような比較的関数の形状が良い場合については、 Faithful Character とほぼ同様の図を得ることができる。また Boundary Character と異なり n 変数の場合で

あっても適用することができる。さらに代数関数以外に対する Weak Character としてはこれ以外には知られていない。一般的な利用に対しては速度、表示質両面で十分実用に耐え得るアルゴリズムである。

しかし、$\mathcal{C}_k$ の内点のみに解が存在した場合その解の検出はできない。さらに Boundary Character と異なり判定を行う外周の一端辺上に関数 $f = 0$ の解が偶数個存在した場合検出できない。

5.2.1 実装

このアルゴリズムを 2 変数代数方程式に制限した実装が、現在 Risa/Asir [8] の ifplot として実現されてる。Boundary Character と同一の条件のもとで Knot= 0 および Heart= 0 を求めた図が 図 7,8,9,10 (各々の図は実際の図を 0.4 倍したもの) である、計算時間は表 2 に記載したとおりである。計算の速度、表示の質を考えると目的によっては十分に実用に耐え得る。

表 2 Sign Weak Character の計算時間

	1/100	1/600
Heart	0.394 秒	13.24 秒
Knot	0.2262 秒	6.177 秒

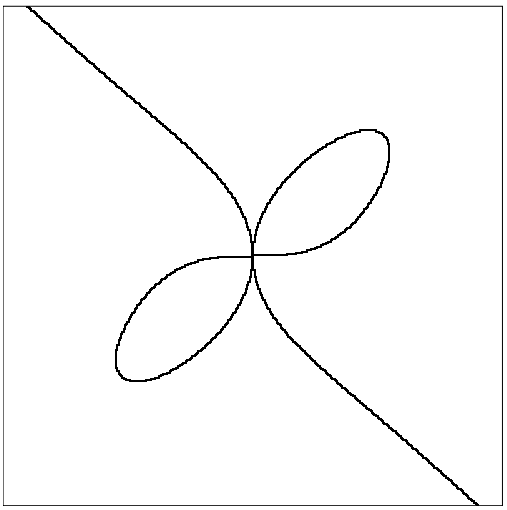

図 7 Sign Weak Character による
$\mathrm{Knot}(x, y) = 0$
解像度 1/100

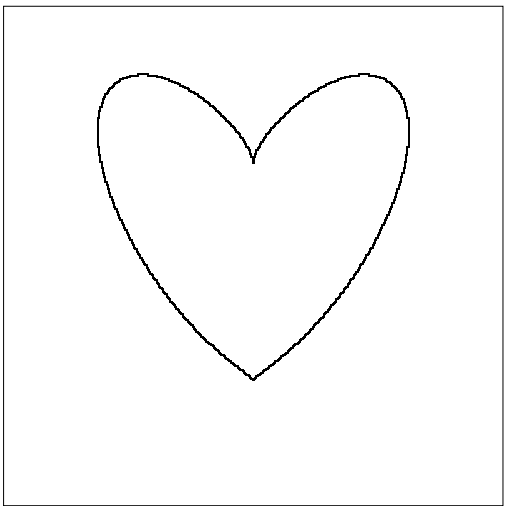

図 8 Sign Weak Character による
$\mathrm{Heart}(x, y) = 0$
解像度 1/100

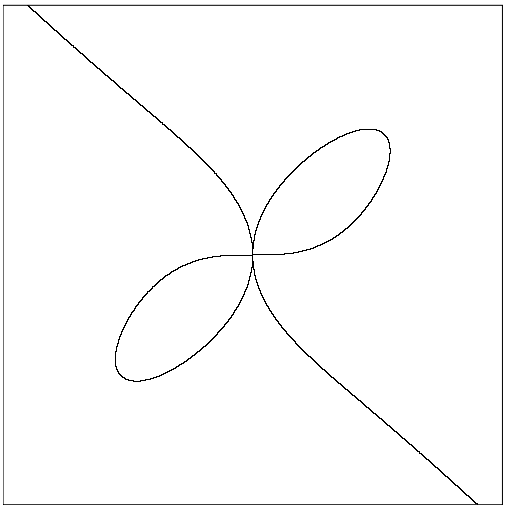

図 9 Sign Weak Character による
$\mathrm{Knot}(x, y) = 0$
解像度 1/600

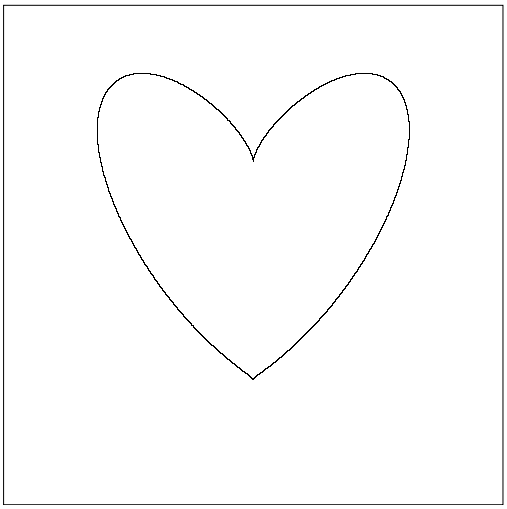

図 10 Sign Weak Character による
$\mathrm{Heart}(x, y) = 0$
解像度 1/600

5.3 Boundary Character と Sign Weak Character の差

実例を見れば Boundary Character と Sign Weak Character の差はほとんど存在しないように見える。異なる点をあげれば 図 3 と図 7 の差は原点付近の線分の太さのみに見える。また、図 4 と図 8 の差はほとんど識別できないほどである。数値として比較しても差異はほとんど無い、しかし実特異点の周辺では数値として異なっておりこの差異が表示のドットの差異として表現されている。実際、図より異なる部分が見えるのは図 4 と図 8 の尖点が Boundary Character の方が関数の零点に忠実に尖っている点である。1/600 のメッシュの方はさらに差は判別しがたいが詳しく見れば 1/100 と同様の状態になっている。

しかし、次のような関数の場合は異なる表示 (図 11,12, 各々大きさを 0.4 倍) をおこなう。

$$\mathrm{F}(x, y) = 100000000yx^2 - \frac{200000000}{7}yx + \frac{100000000}{49}y - 1$$

この関数は $x = 1/7$ が漸近線となる関数である。

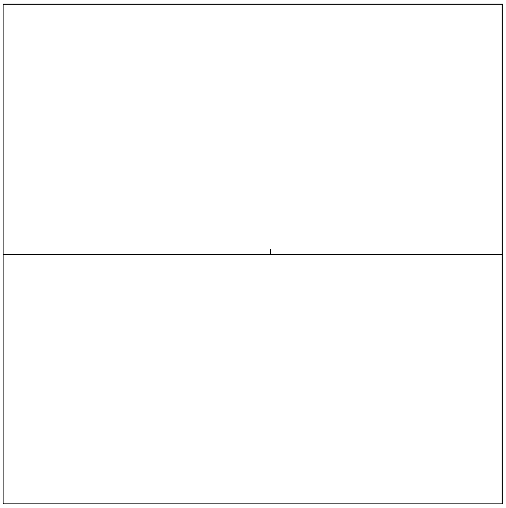

図 11 Sign Weak Character による
$F(x, y) = 0$
解像度 1/600

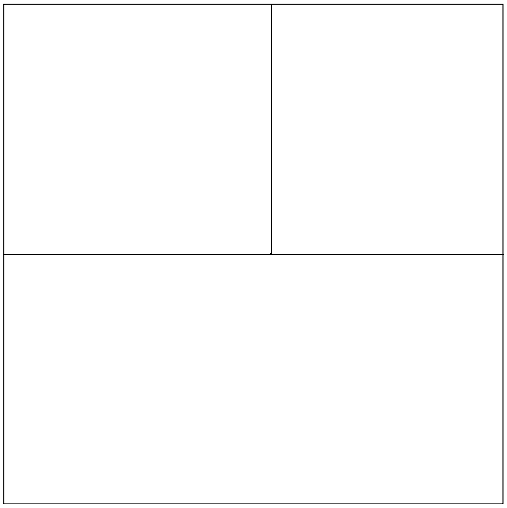

図 12 Boundary Character による
$F(x, y) = 0$
解像度 1/600

表示の範囲は $(2, 2)$, $(-2, 2)$, $(-2, -2)$, $(2, -2)$ で囲まれた範囲である。Cell の大きさは一辺が 1/600 の正方形である。この二つの表示図 11,12 が異なる原因は、Sign Weak Character は同一 Cell の同一辺に偶数個実解が存在した場合、存在が検出できないことからこの現象は起きる。

6 実験と他のアルゴリズムとの比較

本論文で提案したアルゴリズムは、文献 [11] に含まれている例題に関して実験を行った。文献 [11] には 90 個の平面代数曲線の例が含まれており、これらをすべて試したところ、すべての場合に関して Sign Weak Character は解像度 1/100 で 1 秒以内に表示計算が終了した。 Boundary Character に関しても 3 秒以内にすべて表示した。得られた Sign Weak Character の図形を紙に出力し重ね合わす事により図形を比較したところ書籍の出力のため変形しているものを除いてすべて肉眼で差異を判定できない出力であった。利用した計算機システムは、OS を FreeBSD ハードを Intel Pentium Pro 200MHz とした。さらに、文献 [14] において表示できなかった ex. 90 $(16x^4 + (-32y - 8)x^2 - 25y^5 + 16y^2 + 8y + 1)$ もなんら問題なく表示した。更に同様の関数を解像度 1/600 に関して表示したところ Sign Weak Character で 6 秒 Boundary Character で 21 秒であった (図 13, 得られた図を 0.8 倍したもの)。

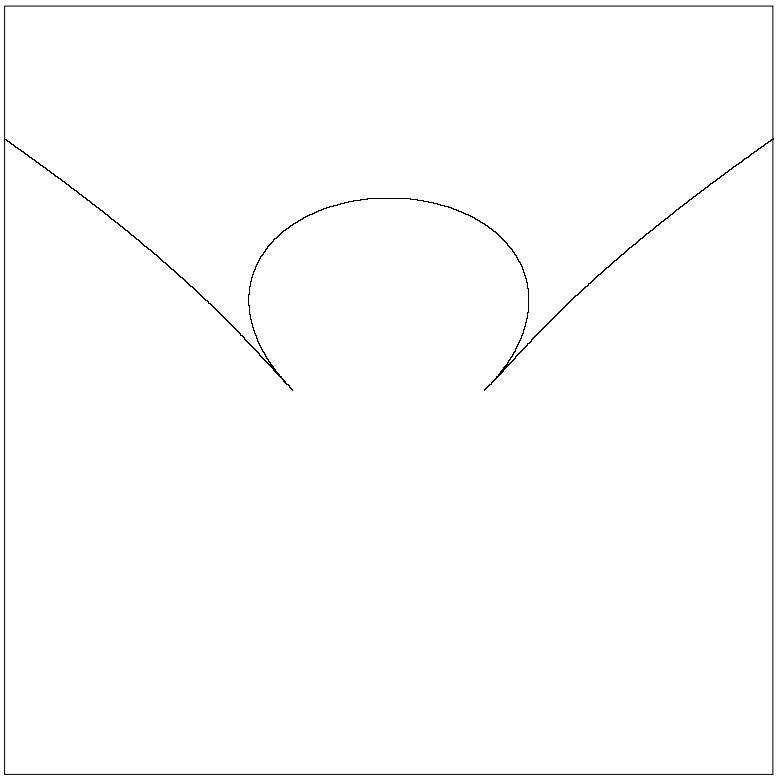

図 13 Boundary Character による
文献 [11] ex. 90
解像度 1/600

また一般的な陰関数の関数表示アルゴリズムとしては、Draw Method として微分方程式に問題を書き直し微分方程式の数値解法を利用する方法がある。その対比として 2 次の Runge-Kutta を利用したアルゴリズム、ならびに Runge-Kutta 法と Newton 法の併用を Risa/Asir 言語で作成し Sign Weak Character、Boundary Character も同様の条件で作成したものと対比実験をした。この場合の実験は、関数として Heart を利用した。実験した計算機は Boundary の場合と同様のものである。

また Runge-Kutta 法の特徴としての出発値の値を求める時間は計測の範囲外としている。次の点を求めるステップの大きさは Runge-Kutta 法はアルゴリズムの範囲外であることより実験を繰り返し正しく表示できる長さを決定した。

得られた図は 図 14,15（得られた図を 0.4 倍してある）においてはクサビ型の部分が本来の関数の示す図形と異なって得られた。さらに Plot Method と異なり線分近似を常に行う為線分の両端点の値がたとえ正しいとしても線分の中間の値が正しい保証が無い点が問題である。この問題を正しく表示する為ステップの長さを表示限界として計算させた場合比較実験では表 3 に示したように Runge-Kutta 法に計算時間の優位さは生じなかった。

表 3 Character と Runge-Kutta 法の計算時間

	1/100	1/600
Runge-Kutta 法	4.77 秒	69.94 秒
Runge-Kutta 法 + Newton 法	1.02 秒	24.85 秒
Sign Weak Character	2.54 秒	24.09 秒
Boundary Character	2.41 秒	47.00 秒

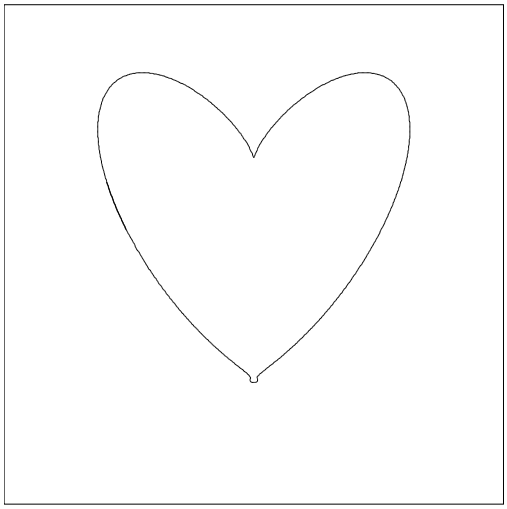

図 14 Runge-Kutta 法
$\mathrm{Heart}(x, y) = 0$
解像度 1/600

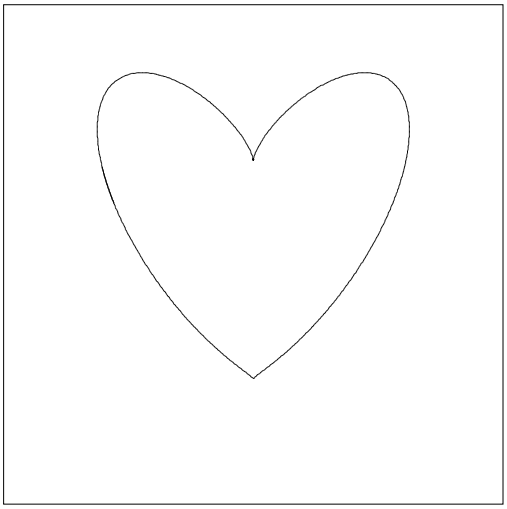

図 15 Runge-Kutta 法 + Newton 法
$\mathrm{Heart}(x, y) = 0$
解像度 1/600

上述の文献 [11] 中の例は特異点を持つものが多く、(特異点を持たない) 一般の曲線に比べて追跡による描画が困難 [14] といわれるが、教科書的な人工的例題であり、現実にはこのように次数・係数 (の表現サイズ) とも小さく、結果として我々がその概形を知っている単純な図形が得られるような問題ばかりとは限らない。図 16 の例は、天文学などて用いられる微分方程式の数値解法公式のひとつである 4 次の Symplectic 数値積分スキームの (8 個ある) 係数を決定するための方程式の $\mathbb{R}^8$ 内の 1 次元解[2)]を $\mathbb{R}^2$ に射影して得られる代数曲線である [9]。

曲線の定義多項式を $f \in \mathbb{Z}[x, y]$ とすれば、f は項数が 212 項、係数が 10 進 10 桁程度、x に関して 14 次、y に関して 18 次、全次数は 21 次である。

追跡法のため特異点を求める方程式は、$f, \frac{\partial f}{\partial x}, \frac{\partial f}{\partial y}$ から終結式を用いて x を消去し、最大

2) 実は方程式が正しく選ばれていれば零次元の解のはずたが、原論文では冗長に選ばれているため、1 次元の解を与える。

公約多項式計算で冗長な因子を除くと、(103 次式)^2(4 次式)^4(1 次式) の形の $\mathbb{Z}$ 上既約な y の多項式 (係数は 150 桁程度) を得る。

103 次式の実根を数値的に得るのは係数の大きさからみて（不可能ではないが）特別の努力が必要であり、著者らは求めてはいない。この困難さのため、導関数を利用するアルゴリズムの欠点を、特異点の情報を用いて追跡の出発点を得ることにより克服するという文献 [14] の方法は、このような問題には適用できない。

しかし、このように特異点を求めることが計算量的に困難であっても、本稿で提案する Boundary Character では正確にかつ失敗すること無く（図 16 原寸大)、($\mathcal{C}$の内部に完全に含まれる微小コンポーネントを除いて）容易に描画できる。タイミングデータとしては、他の図と同様の条件のもとで表 4 の数値である。

また、Sign Weak Character では曲線の数個の特異点の付近を除き描画が容易に得られる。

表 4 実例 [9] の計算時間

Boundary Character	4385 sec.
Sign Weak Character	37.85 sec.

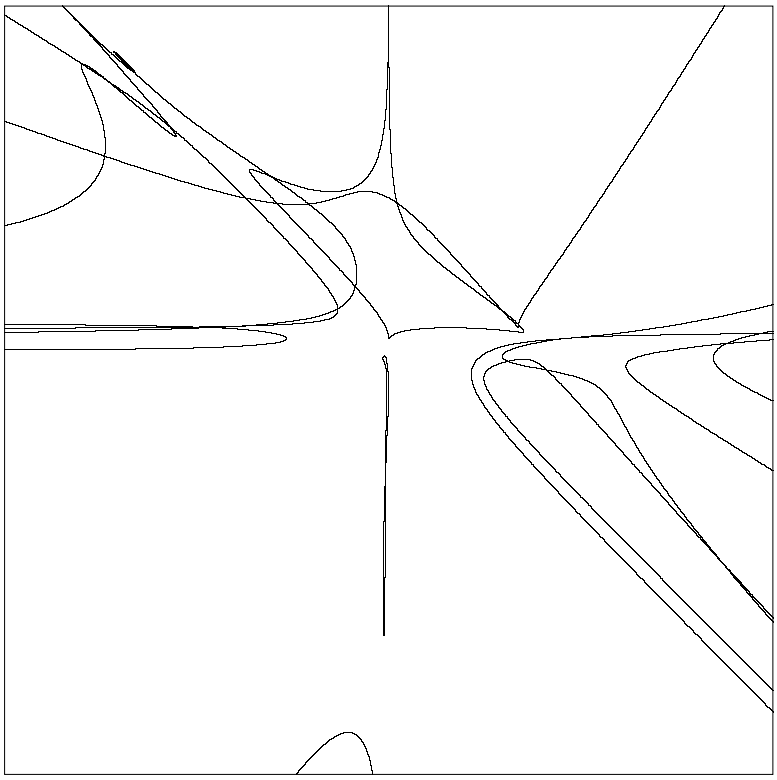

図 16 Boundary Character による
実例 [9] による図 解像度 1/600

7 結論

一般の数式処理システムの中には、通常多項式の実零点の表示ができるものがある。しかし、この機能は数式処理システムの重要な部分を占めるのではなく、付加的なものとして扱われている。したがって、関数の実零点描画はできるが、描画された図形が正しい図であるとは必ずしも言いがたい。数式処理システムが厳密な演算を保証するものであれば、少なくともそれに付随する関数描画システムも、得られた図形が正確であることを保証するアルゴリズムを選択できる必要性がある。

提案したアルゴリズムを利用することにより、今までの描画法とは異なり浮動小数点数演算を一切排除した厳密な描画が可能となる。また、初期値を別途求める必要もない、関数の枝ごとにアルゴリズムを適用する必要もない点は優位である。

さらに手法により描画された関数の描画の結果は誤差解析をすることなく各々の Character の意味で正しい図形と判断できることは、今後のいろいろな利用法が存在すると考える。

本論文では、関数の零点を描画するとは如何なることかを論じ、新たな概念を提案した。またこの概念によるアルゴリズムの構築とその実装を行い結果を示した。得られた表示は、文献 [11] に示している 90 例すべてについて記述してある図と同様の図が秒の単位で表示した事より表示の質はもちろんのこと計算時間の長さにおいても十分実用に耐え得ることを示した。

今後の問題点としては、Faithful Character の計算アルゴリズムの確立は第一の問題である。また、代数関数以外の連続関数に対するアルゴリズムは、提案した Interval Character の場合を除いて手がつけられていない。よって代数関数以外の関数に対する関数表示アルゴリズムの構築は大きな問題である。

参 考 文 献

[1] Alefeld,G., Herzberger,J.: *Introduction to Interval Computations*, Academic Press, (1983)

[2] Bahnhill,R.E., Farin,G., Jordan,M. and Piper,B.R.: Surface/Surface Intersection, *Computer Aided Geometric Design*, Vol.4, pp.3–16, (1987)

[3] Bahnhill,R.E. and Kersey,S.N.: A Marching Method for Parametric Surface/Surface Intersection, *Computer Aided Geometric Design*, Vol.7, pp.257–280, (1990)

[4] Bajaj,C.L., Hoffman,C.M., Lynch,R.E. and Hopcroft,J.E.H.: Tracing Surface Intersections, *Computer Aided Geometric Design*, Vol.5, pp.285–307, (1988)

[5] Fateman,R: Honest Plotting, Global Extrema, and Interval Arithmetic, *Proceedings of ISSAC '92*, New York,1992, pp. 216–223

[6] Kondoh,Y.,Saito,T.,Miyoshi,Y.: 陰関数描画について, 平成 7 年度電気関係学会四国支部連合大会講演論文集, pp. 321, 1995

[7] Kondoh,Y.,Saito,T.,Miyoshi,Y.: 陰関数描画に関する一つの試み, 数理解析研究所講究録 920, pp. 165–172, 1996

[8] Noro,M., Takeshima,T.: Risa/Asir – A Computer Algebra System, *Proceedings of ISSAC '92*, New York, 1992, pp. 387–396

[9] Noro,M., Takeshima,T. : High–Quality Computing of Polynomial Problems by Risa/Asir, *FUJITSU Sci. Tech. J.*, Vol.32, No.2, pp. 256-270, 1996

[10] Pratt,M.J. and Geisow,A.D.: Surface/Surface Intersection Problems, *The Mathematemtics of Surfaces (Gregory,J.A. ed.)*, pp.117–142, Oxford University Press, (1986)

[11] 坂井忠次: グラフと追跡, **培風館**, (1963)

[12] Saito,T.: An extension of Sturm's theorem to two dimensions, *Proc. Japan Academy*, Vol.73, Ser.A, pp.18–19, (1997)

[13] Takeshima,T.,Noro,M.,Saito,T.: 図形描画装置及びその方法, 特許出願番号 *H06-048717*, 1994

[14] 谷口行信, 杉原厚吉: 検出もれのない代数曲線の追跡法, **情報処理学会論文誌**, Vol.33, No.10, pp. 1245–1253, (1992)