数式処理 *J.JSSAC (2005)*
Vol. 12, No. 1, pp. 33 - 46

特集論文

数式処理における関数零点の描画

近藤 祐史* 齋藤友克†

詫間電波工業高等専門学校 株式会社アルファオメガ

(Received 2005/07/31 Revised 2005/08/20)

概 要

Drawing the figure of the real zeros of a algebraic system target, is old problem. This problem is commonly recognized to be an easy one. But it is not easy to give a precise answer. In this paper, we survey the plotting problemes and give new interval plot problems.

1 序言

関数の零点を求め表示することは，計算機が科学技術に利用された初期の段階から重要な技術の一つである．初期の関数の零点描画（グラフの描画）の方法は，なんらかの数値計算をおこない，出発点となる零点を発見する．その後，近くの零点らしい点を見付け線分で結び描画する手法であった．当然このような素朴な描画アルゴリズムによっては描画できない方程式は多々あり，それらを描画するためのアルゴリズム研究はなされてきた．

一方，数式処理は，記号式を取り扱うものであり，描画問題に関し積極的な貢献は見られなかった．なぜならば，描画アルゴリズムは基本的に数値計算を基本に組み立てられているため，数式処理が本質的な寄与することは，少ないと考えられていたためである．

素朴な描画手法は，計算機の処理速度が低速かつ計算機資源が豊富でない時代においては，計算時間等を考えれば有用な手法であった．しかし，この素朴な手法を方程式の零点描画とした場合，何を描画したかが定かでない状況であった．

そもそも方程式の零点問題は，現在においても未解決の部分が多々ある問題である．このような状況において零点を洩れなくかつ正しく求め描画することは，理論的にも技術的にも大変難しい問題である．しかし，関数描画は簡単なアルゴリズムを適用しても一見正しく見える図を描画する．このことは，描画問題は簡単であると言う誤解を人々に与える．

本論文は，関数描画を問題を歴史的に正確に記述するというよりは，描画問題に対し数式処理がどのようにかかわってきたかを論ずる．

*kondoh@dc.takuma-ct.ac.jp
†saito@a2z.co.jp

数式処理の立場から見れば，関数の零点を表示する問題は，解の存在範囲を求める問題ともいえる．この論点からすると関数の零点表示は，ほとんど数値計算の問題に見える．しかし，解の存在範囲を求めることは数式処理にとっても重要な問題であり，この問題の経緯を数式処理の側面から検討する．

2 零点の表示原理

問題の発端は，Fateman [2] による問題提起である．この論文による問題提起を考察するため T.Saito は，グラフ描画の定義を正確に定める必要があると考えた [8]．つまり，零点表示とは何かから考察するべきである．この基本問題の一つの解決案を最初に提示した論文は [6][9] である．

描画する与えられた n 変数方程式 $f(x_1,\ldots,x_n)=0$ の零点集合 V とは，描画対象領域を D とすれば，次のように数学的に定義される．

$$V=\left\{(x_1,\ldots,x_n)\mid f(a_1,\ldots,a_n)=0,\ (a_1,\ldots,a_n)\in D\right\}$$

方程式の零点表示とは，この集合 V をなんらかの表示装置の上に写像することである．

ここで，数学と実際の表示グラフの差異の問題に直面する．数学的には，個々の点は大きさを持たない．しかし，表示装置の物理的な点は大きさを持つ事が問題を複雑にする．つまり，点 $(a_1,\ldots,a_n)$ を表示した場合，表示点は点ではなく大きさを持つ領域となることである．言い替えれば，関数の零点描画は，関数・描画域・装置からなるシステム全体を考慮しなければならないということである．

この領域が表示したい点を含む領域であればこの表示は正しい表示と考えても不都合はない．しかし，正しいグラフとはいかなるグラフであるかの評価が明確化されずに済まされてきている．このことがグラフ表示の問題を非論理的に取り扱われる現況の原因となっている．

表示は数学的な実体を物理的な実体に写像する処理であることを正面から論じた論文は少ない [6][9][10]．グラフ描画のアルゴリズムを開発したとしてもグラフが何をもって正しいとするかの評価が定まっていない以上アルゴリズムの評価が恣意的になることは当然である．

もちろん表示された領域が零点を含まない可能性が存在するが，事前に定められた誤差の範囲では，領域の近傍に零点が存在するとする近似的な理論も存在しえる．しかし，著者らは本来大きさを持たない点を大きさを持つ領域に写像している以上誤差を含んでということは適切とは考えない．誤差を含むのであれば，表示した点の領域を広げるべきである．例えて言えば，$x^2-2=0$ の 1 つの解を 0.1414×10 とする．これは，解 $\sqrt{2}$ を $[\frac{14135}{1000},\frac{14144}{1000}]$ の領域に写像したことを示す．この写像された領域をさらに誤差を含むとした場合何を求めたかがあいまいになる．

3 陽関数の描画

描画される関数が陽関数の場合，比較的素朴なアルゴリズムは有用である．与えられた関数を $y=f(x)$ とする．また表示する範囲は $a\le x\le b$ とする．陽関数であるため，$f(a), f(b)$ は何ら問題なく求めることができる．その結果あたえられた関数の零点の軌跡は $(a,f(a)), (b,f(b))$ を結ぶ曲線となる．もっとも素朴に考えれば区間 $[a,b]$ を必要な精度で n-分割し，点列 $a_0=a,\ldots,a_n=b$

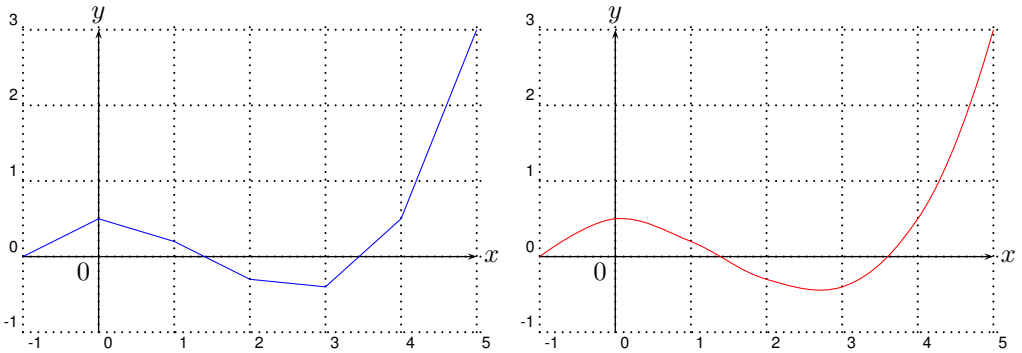

図 1: 陽関数の零点表示の例

を定め各々の関数の値を求める．得られた点列 $(a_i, f(a_i)), i = 1, \ldots, n$ を線分で結ぶ手法である．例えば

$$y = \frac{x^3 - 4x + 5}{10}$$

を区間 $[-1, 5]$ で表示することを考える．分点を $-1, 0, 1, 2, 3, 4, 5$ とすると各々の値は，$(-1, 0)$, $(0, 0.5)$, $(1, 0.2)$, $(2, -0.3)$, $(3, -0.4)$, $(4, 0.5)$, $(5, 3)$ となる．この値に対応する座標に点を置き個々の点を線分で結べば（図 1 左側），一応関数の零点表示はできる．もちろんこのグラフは実用から見て十分とは言えない．より正しい関数描画を求めるためには，分点の数をさらに増やしグラフを描画する必要がある．実際分点を増やした図 1 右側のグラフは，本来の関数の描画として一見正しく見える[1)]．グラフが正しいという判断は，与えられた関数の性質が良くわかっており，どのようなグラフが正確なグラフであるかを事前に描画者が知っているからである．

この関数描画は，実際正しい図を表示している．しかし，描かれた点が本来の正しい点の位置からどの程度誤差を含んでいるかは，描画されたグラフを見ても判定できない．誤差を正確に知るためには，描画アルゴリズムの誤差評価を別途する必要がある．

このアルゴリズムの基本は，関数 $f(x)$ の値を正確に求めることができるということが仮定されている．もしくは，どの程度の誤差が含まれているか明確に提示されていなければならない．そのような前提が存在しない場合描かれた図形は正確さという意味で意味を持たない．

しかし，このアルゴリズムの最大の仮定事項は，数値計算誤差ではなく，関数 $f(x)$ が一価である点である．また，関数が区間 $[a, b]$ において連続であることを仮定している．この仮定が満たされている限り，このアルゴリズムは，数値計算による誤差を除いて適正なグラフを表示すると考えてよい．

このアルゴリズムが素朴と言われるのは，2 点 $(a_i, f(a_i))$ と $(a_{i+1}, f(a_{i+1}))$ を単純に線分で結ぶ点である．つまり，関数の挙動を局所的ではあるが線形近似をしている点である．与えられた関数が代数関数であったとしても局所的に常に線形近似をする手法は，常に正しいグラフを表示するとは限らない点に利用者は十分に留意するべきである．特に，関数が多変数関数の場合は多項式であったとしても，局所線形性は描画の上では成り立たない場合が多々ある．

[1)] これらの描画は LaTeX の PS-Trics.sty によって描いた．プログラムとして単純に点を計算し結んだだけである．この素朴な方法でも十分実用に耐えることが図から見て取ることができる．

関数近似を伴う描画は，比較的高速に処理が終わる．しかし，高次の近似をしたとしても無条件で必要な精度のグラフが描画できるとは限らない．

陽関数は，変形することにより容易に陰関数にすることができる．そのためより正確な描画をするためには，陽関数を陰関数として描画する方が計算時間はかかるが，描画としてはより望ましい図形が得られる．

4 陰関数の描画

陰関数の零点を可視化する問題は，たとえ有理係数 2 変数代数関数として誤差の問題を除去したとしても，陽関数の場合と比較して格段に難しい問題である．以下 2 変数の場合を例に取りアルゴリズムを外観する．

4.1 素朴なアルゴリズム

最も素朴に考えると，陽関数と同様に x の分点 $a_0,\ldots,a_n$ を定め $i=0,\ldots,n$ 各々に対し $f(a_i,y)=0$ となるそれぞれの y の解 $b_{i,j},\ j=0,\ldots,n_i$ を求める．この $(a_i,b_{i,j}),\ i=0,\ldots,n,\ j=0,\ldots,n_i$ を適宜結べば陽関数と同様のグラフは描けるはずというアイディアである．

この手法は，描く曲線が交差しないという条件があれば旨く描画できる可能性はある．しかし，曲線が自己交差するか否かの判定は，Gröbner 基底を計算しなければ確実には判定できない．つまり，単純な数値計算ではこの素朴なアルゴリズムは破綻する場合がある．

この問題を除去するためには，個々の分枝を分離し局所的に陽関数に変形しグラフ描画をおこなう方針もある．分枝を求めるためには，自己交差する点を正確にかつ洩れなく求めなければならない．この正確かつ洩れなくということは大変難しい問題である．

さらにこのアルゴリズムは，曲線の分岐と独立した曲線を検出できない場合が存在する．この為に微分方程式等を利用して後述の追跡型にならざるを得ない．しかし，実際にはこの素朴なアイディアによる描画を陰関数の描画であると実装している数式処理システムも存在する．この描画システムは，計算が軽便である点を評価し実装していると考えられる．さらに，関数零点の描画はお手軽な指針を与えるシステムと割り切れば，このアルゴリズムの選択もありうる．

4.2 追跡型アルゴリズム

追跡型アルゴリズムは，2 変数関数 $f(x,y)=0$ の零点表示アルゴリズムとして最も普及しているものである．このアルゴリズムの基本は，なんらかの方法により出発点となる $f(x,y)=0$ の解 (a,b) を発見することから始まる．このなんらかの方法として最も簡便な方法は，

$$\begin{cases} \frac{\partial f}{\partial x}=0 \\ f(x,y)=0 \end{cases}$$

となる連立方程式を解くことである．この解は元の関数が無平方であれば[2)]，ゼロ次元であり数値計算により求めることは可能である．

次に求めた出発点から微分方程式などを利用して次の点を求める手法である．つまり，この手法の基本は，出発点から次の曲線上の点を求め解曲線を追跡していくアルゴリズムである．この

[2)] 代数関数を無平方にすることは GCD により容易に実現できる．また，表示されるグラフは元のグラフと同一の図形となる．

追跡の方法はさまざまであるが，通常使われるのは，接線などを用いた線形近似を行い繰り返し演算により解の値を得るものである．

この場合も陽関数の描画と同じ問題を抱える．近似計算をする以上誤差が計算中に入り込むことは必然である．そのため，この誤差を排除する為に十分な配慮が必要である．

また，グラフが自己交差している場合，分枝に対する追跡の方向を旨く選ばないとグラフが欠けたものになる可能性が大である．

さらに，出発点となる値を求める場合，単純な数値計算に依存すると枝が求まらない場合もある．追跡法アルゴリズムおいて，この検出洩れは重大な問題である．この問題を解決した論文として [11] がある．この解決策も繰り返しによるもので，理論上は，十分良い精度で繰り返せば解に到達できるという考え方に基くアルゴリズムである．

この追跡型のアルゴリズムは通常の場合軽便に解を表示できる利点がある．予想したグラフと異なる図が得られた場合再度追跡手法を変更して求める方法もあり，実用上は意味のある手法である．

4.3 Cell 型アルゴリズム

Cell 型のアルゴリズムの基礎概念と初期実装は，1993 年に齋藤により提案された [5]．代数曲線の描画に対し Cell ならびに Character という新しい概念を提案し，その実装に基づく描画を示した．この時点で Character と呼ばれる描画関数は，Signature Character と Precise Character が提案された．その後，Cell 型描画アルゴリズムの概念を理論的に洗練し，実現可能アルゴリズムを提示した論文は [8][6][9][10] などである．

Cell 型の描画アルゴリズムの基本は，描画すべきではない点を判定することである．つまり必要とみなされる点を描画するのではなく，零点とするには不適切な点を消去することにより関数の零点を表示することである．どのような基準で消去するかの判定の優劣により，描かれている図形の正確さが明示的に提示されることになる．

4.3.1 Cell 描画の基本概念

方程式の零点描画の基本概念は，[6] に明確に定義されている．但し，ここに提案されているアイディアが完全かつ全てであるわけではない．あくまでもこの提示された概念は，零点描画の 1 つのアイディアである．しかし，現在に至るも他のアイディアは提案されていない．以下 [6] で提案された概念を簡単に述べる．

4.3.2 Cell の定義

関数の零点を描画するためには，描画装置が存在する．つまり，関数の零点描画とは，描画装置に依存していることを認識する必要がある．その装置により描画する画素の単位を Cell と呼ぶ．この単位は，装置の限界性能の必要性はなく，利用者の必要な精度であることからあえて装置解像度ではなく Cell と呼ぶ．単純に Cell とは何かと言えば零点描画をする場合の描画最小単位のことである．

定義 1 (Cell の定義)
表示領域 D 上の C_k $(k = 1, \ldots, m)$ が D 上定義された Cell であるとは，

1. $D = \bigcup_{k=1}^{m} C_k, \qquad C_k^i \bigcap C_j^i = \phi, \qquad k \neq j,$
2. 各 C_k の Jordan 測度はゼロではない.

とする．ここで C_k^i は，C_k の内点の全体とする．

4.3.3 描画関数（Character）

描画空間の定義の Cell が定まった状態で描画を定める描画関数 (Character) を定義できる．

定義 2 (Character の定義)
D 上定義されている関数 f の Cell の族 $\{C_k\}$ による Character χ とは，

1. $\chi : \{C_k\} \to \{0, 1\}$
2. $\chi(C_k) = 0$ ならば C_k の任意の元 x に対し $f(x) \neq 0$

とする．

この描画関数の意味は，関数 $f(x)$ の零点を含む Cell に対する値は 1 である．つまり描画される．$\chi(C_k) = 1$ の場合は，零点を含む可能性があることである．この可能性の程度を定めることにより描画関数は，様々なものが考えられる．さらに，描画関数全体は，包含関係により束の代数構造を持つ．この束の極大元が存在しその元が Faithful Character であることが示されている [6].

定義 3 (Faithful Character)
D 上定義されている関数 f の Cell $\{C_k\}$ による Faithful Character とは，

$$\chi(C_k) = \begin{cases} 0 & f(x) \neq 0 \ {}^{\forall}x \in C_k \\ 1 & f(x) = 0 \ {}^{\exists}x \in C_k \end{cases}$$

ここで Character の値が 1 の場合に点（Cell）を描画する．

この Faithful Character に従って表示するアルゴリズムは，前述の描画アルゴリズムに比べて格段に難しくなる．一般の方程式の場合 Faithful Character を求めることは極めて困難である．しかしながら，グラフ描画は実用的な需要が高く，必ずしも Faithful Character 程の厳密な正しさを要求しない需要も存在する．この必ずしも正しくはないグラフについては，どの程度正しいかが明示的に評価されうるアルゴリズムの開発が重要である．

そのため Character の概念を弱めた Weak Character と呼ばれるものを考えることは重要である．条件を弱めるとは，$\chi(C_k) = 0$ であっても解を含む場合がある，とすることである．この弱い Character の例として Sign Weak Character などがある．

5 種々の Character

現在一般的に利用されている Character は，外周に関数が交差するかを判定する手法である．判定法により，数値計算型，区間数型，符号型，解の存在範囲判定型などがある．どの判定法を選ぶかは，必要な正確さと計算量の問題である．

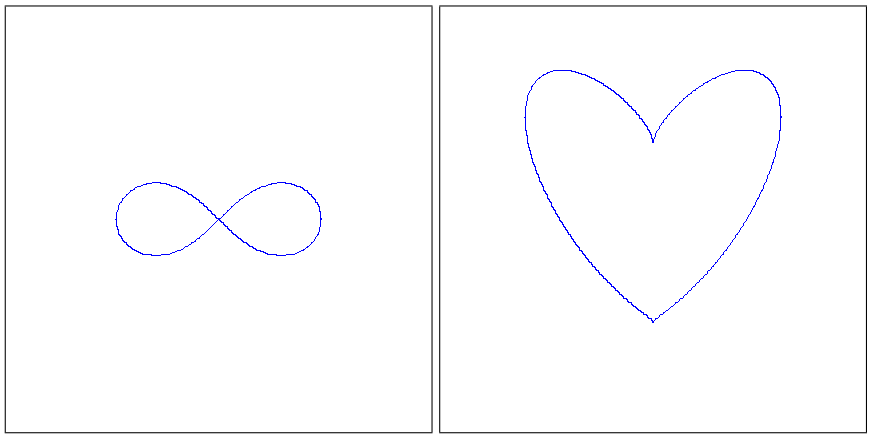

図 2: Sign Weak Character

実際には，Cell の境界を直線にする場合がほとんどである．このため直線上の 1 変数方程式を解く問題に帰着できるため計算量的には十分実用に耐える時間で図は求まる．

Cell 型アルゴリズムの基本である種々の Character を示す．これらの Character は，Risa/Asir 上に実装されている．

実例としては，

$$Lemniscate = x^2 - y^2 - (x^2 + y^2)^2$$

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

を用いる．

表示領域は，$-2 \leq x, y \leq 2$ の領域とする．Cell は $\frac{1}{128} \times \frac{1}{128}$ の正方形とする．

5.1 Signature Character (Sign Weak Character)

もっとも最初に実装された Character である．この Character は，Character の概念を確定する前に策定されたものであるため本来の Character の定義とは異なっている Weak Character と呼ばれるものである．専ら実用上の利便性を考えたものである．

対象となる関数は描画域全域で定義された連続関数である．各々の Cell は計算の軽量化のため矩形としている．しかし，理論上はどのような形でも問題ない．

Cell は次のように定める．

$$C_k = \{(x_1, \ldots, x_n) | \quad a_{k,l} \leq x_l \leq b_{k,l}, \quad 1 \leq l \leq n\} \tag{1}$$

定義 4 (Sign Weak Character)

D 上の Cell 族 $\{C_k\}$ による関数 $f(x_1, \ldots, x_n) = 0$ に対する Sign Weak Character σ とは，

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

この Character の理論的背景は，中間値の定理である．つまり，ある区間で連続関数の符号が異なる場合その区間の中に解が必ず存在することを利用している．

5.1.1 Signature Character の利点

利点としては，計算が軽いという点と，関数が代数関数でない場合も利用できることである．指定された点の符号判定さえできれば描画できることは他の Character と比べて十分な利点がある．この Character は，Risa/Asir の ifplot として実装されている．

5.1.2 Signature Character の問題点

この Character の問題点は，次の 2 点である．

1. Cell の一辺を解曲線が偶数回交わる場合解が存在しないと判定する．
2. Cell の内部に完全に含まれる閉曲線・孤立点が存在しないと判定する．

5.2 Precise Character (Boundary Character)

この Character は，5.1.2 節で指摘した sign weak Character の 1 番目の問題点を除去した Character である．この Character は，Cell の境界のみに注目している．

定義 5 (Boundary Character)
D 上の Cell 族 $\{C_k\}$ による関数 $f(x, y) = 0$ に対する Boundary Character β とは，

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

ここで $\partial\bar{C}_k$ は Cell C_k の閉包の境界をなす集合である．

大雑把に言えば，この Character は零点が Cell の境界上にある場合に 1 となる．この Character の実装としては，関数が 2 変数有理係数代数関数であれば，Sturm 列を求め Sturm の定理により解の判定が確実にできることを利用すれば計算できる．この条件の元で Risa/Asir には ifplot の Precise Mode として実装されている．

5.2.1 Precise Character の利点

描画は正確であり，解が存在するとした Cell には確実に解が存在することが保証されていることは最大の利点である．

5.2.2 Precise Character の問題点

Risa/Asir の ifplot を起動した状態の描画は, Signature Character である．このモードから Precise Mode に変更した場合描画の速度は明らかに遅くなる．これは，Signature Character では各点での符号判定だけをおこなうのに比べ，Precise Character では Sturm 列の符号判定をおこなってい

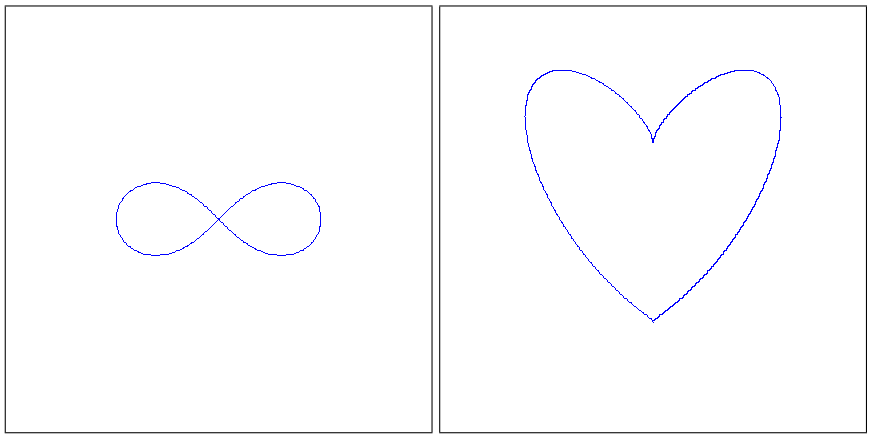

図 3: Precise Character

るためである．しかし，現在の計算機資源を考えると極端に遅い描画とは言えずこの点は時代に依存する．

しかし，Sturm の定理を利用しているため，描画適用関数が有理係数代数方程式に限られてしまうことが最大の問題点といえる．

Cell に埋没する微小な閉曲線と孤立特異点はこの Character の適用によっても，存在しないと判定してしまうことは問題点である．

5.3 Faithful Character

定義 3 の Faithful Character は，Character の中で最も正しい Character である．現在この Character の実現可能な関数は，多変数多項式の場合のみである．

1. $f(x,y)$ を無平方化する．以下この無平方化した関数に関して実行する．
2. Boundary Character を実行する．
3. $\left\{\frac{\partial f}{\partial x}, \frac{\partial f}{\partial y}, f(x,y)\right\}$ の Gröbner 基底を求める．
4. Gröbner 基底の解の属する Cell を決定する．（miniploy などを利用）

Risa/Asir の ifplot には, 現時点では実装されていない．しかし, 全ての要素技術はすでに Risa/Asir に実装されているため Presice Mode と asir 言語を利用してユーザレベルで実装することは容易である．

5.3.1 Faithful Character の利点

Faithful Character は，Cell の内部に埋没する小さな解曲線や孤立特異点を判定し描画することができる．

5.3.2 Faithful Character の問題点

問題点としては，Gröbner 基底を求めその解を求める為の計算が計算資源を多く必要とすることである．記憶容量の大きくかつ計算速度が高速な計算機でないと結果を求めるのに必要な時

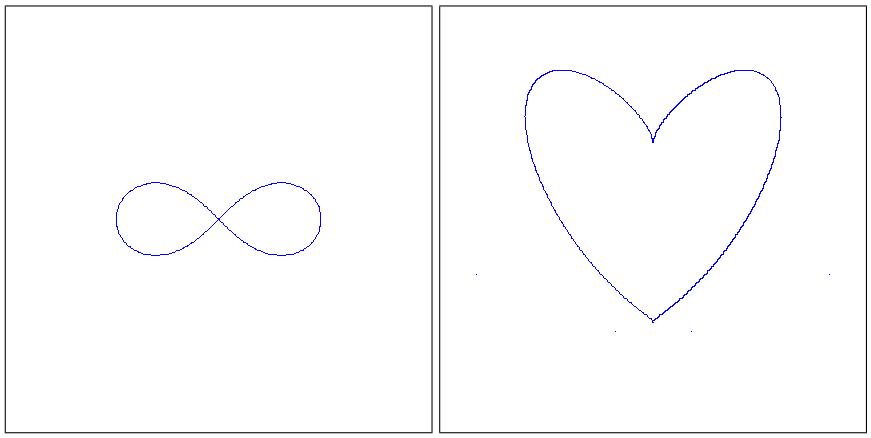

図 4: Faithful Character

間が長大になることが多々ある．しかし，Gröbner 基底が計算できればほとんどの場合結果は求まる．

5.4 Interval Character

この Character は，1995 年に近藤等により提案された Character である [3]．他の Character と計算方針が異なる．要素技術は，区間演算である．

Cell C_k は，Signature Character と同様境界を含む矩形領域とする（式 (1)）．また区間数 $I_{k,l}$ を $[a_{k,l}, b_{k,l}]$ とする．

定義 6 (Interval Character)
χ が Interval Character であるとは，

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

と定義する．ここで $f(I_{k,1}, \ldots, I_{k,n})$ は, 関数を区間数 $I_{k,i}$ $(i = 1, \ldots, n)$ 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．

5.4.1 Interval Character の利点

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 C_k の中に関数 $f = 0$ の零点が含まれていれば，必ず $\chi(C_k)$ 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 $\{C_k\}$ 上の f の Character であることは容易に検証できる [1].

さらに Character の構成が容易である．このことより，システムが区間演算にさえ対応していれば実装が容易である．さらに計算が安定に行える点は得られた図の精度の問題が生じなければ，優れた Character である．

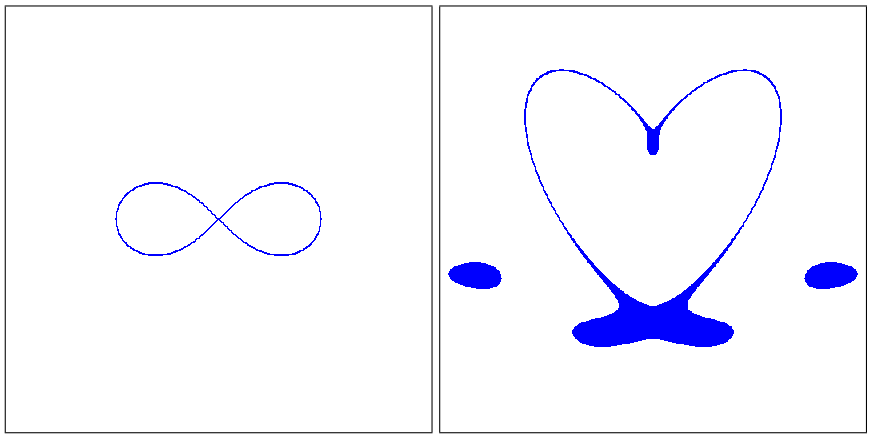

図 5: Interval Character

他の Character と比較して特に利点としてあげられる点は，他の知られている Character 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．つまり，区間演算ができる関数，例えば通常使われる初等解析関数などであれば描画できる点である．

特に区間演算を工夫し区間膨張を抑えることが関数形から容易な場合，安定に描画できる点は優れている．得られた表示は，零点を実際に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．

5.4.2 Interval Character の問題点

Interval Character は，区間演算の特徴（区間膨張）により解が存在しない Cell に対し実際に零点があるかどうかが判定不能であり，その結果は零点が存在しない Cell に対しても描画してしまう性質がある．（図 5 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．この点を判定するためにさらに領域を細分化して再計算を行えば通常より良い図がでるはずである．しかし単純な細分のみにたよった手法では Character がどのような振舞を示すかは理論的には解明されていない．

元来 Character は，解が存在しないことを判別し明示するアルゴリズムであることより，このことは正しいとも言えるが，"図が正確か"という問に対し"これ以外に解は存在しない"としか答えられない点は問題点とも言える．

区間演算の性質である区間膨張は常に存在する．その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．区間膨張をどの程度抑えどの程度正確な図を求めるかは，描画関数の問題というべきではなく区間演算の問題と考える．また区間演算の適用の手法によっては得られた図が異なる点もある．

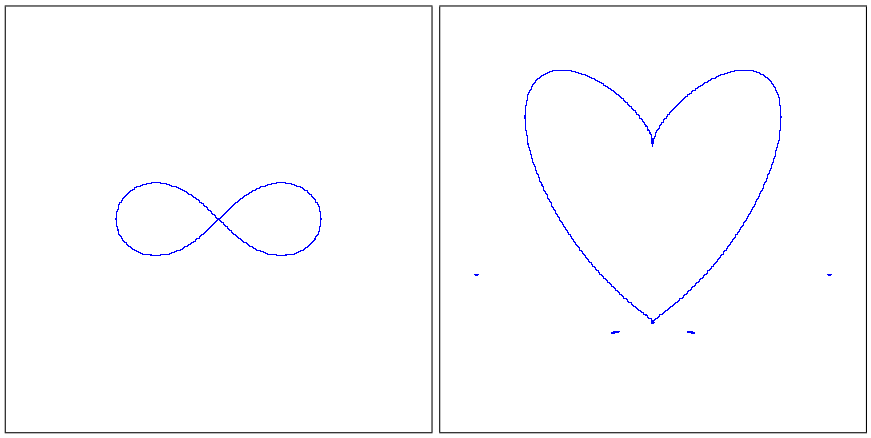

図 6: Interval Character 改

5.5 Interval Character 改良版

Interval Character では，区間数が演算の結果膨張することにより，過大な領域を零点として表示する性質がある．そこで，区間数の性質を利用して Interval Character の改良を試みる．この改良は，区間数の区間膨張が式の平行移動により異なることを利用する．始めに通常の Interval Character により表示領域を求める．その後，表示候補領域を平行移動させ区間膨張を抑えるアルゴリズムである．その結果は図 6 となり効果は大きい．さらにこの為に必要な計算時間の増大は，全体の計算時間の 1.1 倍程度であった．区間数を利用したグラフ零点表示は，まだ開発途中である．小さなアイディアによる改良も劇的な効果を示す可能性がある．

5.6 提示した Character の差

提案した各々の Character に関して描画された図 2~6 は，図から見た限りほとんど差が無い．実際この事例では，描画数にして数点の差異しかない．しかし，Interval Character の場合表示されている点の数は，格段に異なる．さらに，Interval Character のアルゴリズムをほんの少し改良しただけで効果は絶大なものがある．この結果からも，Interval Character に関しては，研究の途中であることがよくわかる．

6 結論

関数零点の表示は，通常数値計算の問題であると認識されている．しかし，関数の自己交差する点を求めるアルゴリズムは，数値計算のみでおこなうよりも数式処理計算を併用するほうがより安定である．

また，関数零点の描画は適当にアルゴリズムを構築したとしても低次の多項式ならば正しそうな図を表示することは容易である．しかし，描画されたグラフが正しいか否かの判定なしに無条件で描画されたグラフを受け入れることは単なるデザインであれば別として，学問的には問題である．

今後の数式処理と関数零点の描画に関する問題点は，次のようなものである．

1. 3 変数以上の Faithful Character の実現
2. 代数関数以外に適用できる実用的な Character の構成

6.1 3 変数の関数の Faithful Character の問題点

3 変数関数の零点表示の問題点は，次の点である．

- 零点の次元が 2 次元と 0 次元の場合は，2 変数と同様のアルゴリズムにより表示することはできる．しかし，1 次元の零点集合に関しては現時点で現実的な解決策は知られていない．
- 2 次元の解を表示するためには，Cell の外周面における Character を求めさらに内点に関する描画をおこなうアルゴリズムのみが提案されている．このアルゴリズムを適用した場合計算時間が膨大である．

6.2 一般の関数の Character の構成

代数関数以外に適用できる Character としては，Sign Weak Character が現在知られている．Sign Weak Character は，安定して描画することができるが，検出できない解が理論上存在する．検出洩れのない一般の関数に対する Character の構成は解決されていない．

検出洩れがない Character としては Interval Character は存在する．この Character の解の不存在領域を狭めるアルゴリズムの構成は重要である．数式処理と区間数の整合性をより深く研究することは，数式処理の発展のために重要と考える．

参 考 文 献

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

[2] Fateman,R. Honest plotting, global extrema, and interval arithmetic. In *Proceedings of ISSAC '92*, pp. 216–223. Academic Press, 1992.

[3] 近藤祐史, 三好善彦, 齋藤友克. 数式処理ライブラリ risa と区間演算を用いた陰関数描画, 1995. 第 4 回 日本数式処理学会大会.

[4] Kondoh,Y., Saito,T., and Takeshima,T. A new algorithm for real roots of a zero-dimensional system by a linear separating map. In *Computer Mathematics*, Vol. 9 of *Lecture Notes Series on Computing*, pp. 56–65. World Scientific, 2001. ASCM 2002.

[5] 齋藤友克. 代数曲線のグラフ表示, 1993. 第 2 回 日本数式処理学会大会.

[6] 齋藤友克, 近藤祐史, 三好善彦, 竹島卓. Displaying real solution of mathematical equations. 『数式処理』, Vol. 6, No. 2, pp. 2–21, 1998.

[7] 齋藤友克, 竹島卓, 平野照比古. 『グレブナー基底の計算 実践編——Risa/Asir で解く』. （財）東京大学出版会, 2003.

[8] Saito,T. An extension of sturm's theorem to two dimensions. *Proceedings of the Japan Academy*, Vol. 73 A, pp. 18–19, 1997.

[9] Saito,T., Kondoh,Y., Miyoshi,Y., and Takeshima,T. Faithful plotting of real curves defined by bivariate rational polynomials. 情報処理学会論文誌, Vol. 41, No. 4, pp. 1009–1017, 2000.

[10] Saito,T., Kondoh,Y., Miyoshi,Y., and Takeshima,T. Faithful plotting on a two dimensional pixel space. In *Josai Mathmatical Monograph*, Vol. 2, pp. 77–86. Josai Univ., 2000.

[11] 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.