数式処理 *Bulletin of JSSAC(2017)*
Vol. 23, No. 1, pp. 20 - 29

桂先生名誉会長推戴記念特別研究会論文

3 変数陰関数描画の実装について

近藤祐史　香川高等専門学校
大墨礼子　(兵頭)
村尾裕一　電気通信大学
齋藤友克　(株) アルファオメガ
サレジオ工業高等専門学校

(Received 2016/2/6　Revised 2016/4/23　Accepted 2016/5/31)

概　要

We have long been working on developing algorithms for drawing graphs of bivariate implicit functions using interval arithmetic. In this paper, we explain how we extend this method to trivariate cases.

1 はじめに

数式処理システム Risa/Asir には，2 変数陰関数の実数上での零点の描画機能として関数 ifplot が実装されている．ifplot は，数学的に正確な描画を目的として，文献 [8][9] で提案されたアルゴリズムの一部により描画する．筆者らは，2 変数陰関数描画を 3 変数の陰関数描画に適用することを試みている [6][7]．3 変数陰関数描画においては，忠実な描画を行う方法の実現には至っていない．本稿では，洩れのない描画を行うことができる区間演算を用いる方法 [5] について検討する．基本的なアイデアは，描画単位を区間数とみなし，区間演算により区間評価を行うことである．区間評価の結果，0 を含まない場合はその中に零点がないことが保証される．しかし，区間評価には，必要以上に区間幅が増大するという欠点がある．その解決法として，筆者らは，区間の端点を原点にシフトする方法を提案している．一方，区間膨張を抑える方法として，アフィン演算を用いる方法がある [10]．本稿と同様に，素朴な区間演算やアフィン演算を利用した陰関数描画が，文献 [2] などで報告されている．

本稿では，区間演算を用いた 3 変数陰関数描画において，アフィン演算や原点シフトなどの区間評価法の比較を行うとともに，区間演算を用いた描画では，忠実ではないが決して洩れることがない描画結果が得られることを示す．特に 2 つの代数曲面が交差する境界として表現される一つの既約な代数関数においても洩れなく描画できることを実行例で示す．

2 陰関数描画

2 変数では，表示できる精度に分割した格子の 1 つの Cell 内に $f(x,y) = 0$ の零点が含まれるかどうかを指標関数に基づき判定する方法を提案した [8][9].

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，装置の限界性能ではなく，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 とは何かとい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．

2.1 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) を定義する．

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 [8].

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，

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

とする．

ここで Character の値が 1 の場合に点（Cell）を描画する．

3 Interval Character

基本的には，Cell を区間数だとみなし，区間評価を行い，結果が 0 を含むかどうかで判定する．

3.1 区間演算

区間数 $A = [a_l, a_h], B = [b_l, b_h]$ とすると，区間数の演算は次のように定義される．

$$
\begin{aligned}
A + B &= [a_l + b_l, a_h + b_h] \\
A - B &= [a_l - b_h, a_h - b_l] \\
AB &= [\min(a_l b_l, a_l b_h, a_h b_l, a_h b_h), \\
&\quad \max(a_l b_l, a_l b_h, a_h b_l, a_h b_h)] \\
A^n &= \begin{cases} [a_l^n, a_h^n] & (n \text{ が奇数}) \\ [a_h^n, a_l^n] & (a_l \le a_h < 0 \text{ かつ } n \text{ が偶数}) \\ [0, \max(a_l^n, a_h^n)] & (a_l \le 0 \le a_h \text{ かつ } n \text{ が偶数}) \\ [a_l^n, a_h^n] & (0 < a_l \le a_h \text{ かつ } n \text{ が偶数}) \end{cases}
\end{aligned}
$$

実装においては通常浮動小数点数を用いるため丸めを制御する必要がある．特に，区間 X に対して，$f(X) \supseteq \{f(x) | x \in X\}$ となることが知られている．

3.2 Interval Character

一般的な Character を構成するため，区間演算を利用した Character を提案した [5].

Cell C_k は，境界を含む矩形領域とする．また区間数 $I_{k,l}$ を $[a_{k,l}, b_{k,l}]$ とする．

定義 4 (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 $I_{k,i}$ $(i = 1, \ldots, n)$ を用いて区間演算をすることにより得られた区間評価であり，区間数である．

3.3 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，その結果は解が存在しない Cell に対しても描画してしまう性質がある．Interval Character で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おこなえば，通常より良い図が得られる．

区間演算の性質である区間膨張は常に存在する．そ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．

4 区間評価法

Interval Character では，区間数が演算の結果，膨張することにより，過大な領域を零点候補として表示する性質がある．これは，区間評価において結果が不必要に膨張する区間演算特有の問

題に起因する．区間演算での区間評価は，式の演算順序によっても膨張する．そのため，いかに膨張を抑えた区間数を得るかが問題である．

4.1 平均値形式

区間評価の区間膨張を改善する方法として，平均値形式より評価する方法がよく知られている．平均値形式は，3 変数の場合 $f(IX, IY, IZ)$ を計算するとき，$F_m(IX, IY, IZ)$ として計算するものである．

$$\begin{aligned} F_m(IX, IY, IZ) = f(mid(IX), mid(IY), mid(IZ)) + f_x(IX, IY, IZ)(IX - mid(IX)) \\ + f_y(IX, IY, IZ)(IY - mid(IY)) + f_z(IX, IY, IZ)(IZ - mid(Iz)) \end{aligned}$$

ただし，区間数 I に対し，$mid(I)$ は I の中心を表す．また，IX, IY, IZ は，それぞれセル（ボクセル）を表す区間数である．

4.2 原点シフトによる方法

筆者らは，有効な区間評価法として，評価する区間の端点を原点へ平行移動（シフト）することにより，区間幅を抑制できることを提案している [5]．素朴な区間演算は原点付近とそうでない場合では，区間膨張の状況が異なる [3]．そこで，この区間数の性質を利用して，評価する区間数の端点が原点となるように平行移動を行い，区間評価を得る．Interval Character の $f(I_{k,1}, \ldots, I_{k,n})$ を計算するとき，区間数 $I_{k,1}, \ldots, I_{k,n}$ の下限が原点となるように，平行移動をしたもの $f'(I'_{k,1}, \ldots, I'_{k,n})$ を用いて計算する．すなわち，

$$I'_{k,i} = I_{k,i} - a_{k,i} = [0, b_{k,i} - a_{k,i}], \quad i = 1, \ldots, n$$

$$f'(x_1, \ldots, x_n) = f(x_1 + a_{k,1}, \ldots, x_n + a_{k,n})$$

として，$f(I_{k,1}, \ldots, I_{k,n})$ を計算する代わりに，$f'(I'_{k,1}, \ldots, I'_{k,n})$ を計算する．描画対象の平行移動の式は，数式処理で簡単に求められるが，すべての区間に対し平行移動をすることはコストがかかるため，始めに素朴な区間演算で Interval Character により表示領域を求め，その後，表示候補領域の 1 つ 1 つに対し，平行移動させ $f'(I'_{k,1}, \ldots, I'_{k,n})$ に 0 が含まれるか再判定するとよい．

4.3 アフィン演算

変数間の相関性を考慮することにより区間膨張を抑える方法の一つにアフィン演算を用いる方法 [10] がある．

変数範囲が $-1 \leq \varepsilon_k \leq 1$ であるようなダミー変数 ε_k を用いて，その線型結合

$$a_0 + a_1\varepsilon_1 + \cdots + a_n\varepsilon_n$$

の形（Affine 形式）で数（区間）を表現する．

Affine 形式間の演算が定義されている．計算途中で ε_i を追加するため，追加したダミー変数の最大数は変化する．

5 試作と実行例

3 変数陰関数について，区間評価の比較を行った．実行環境は，FreeBSD 10.1-RELEASE-p19, Intel(R) Core(TM) i5-4690S CPU @ 3.20GHz 16GB Memory である．また，以下の例を用いた．

$Ball(x,y,z) = x^2 + y^2 + z^2 - 1$

$Torus(x,y,z) = 16x^4 + (32y^2 + 32z^2 - 40)x^2 + 16y^4 + (32z^2 - 40)y^2 + 16z^4 + 24z^2 + 9$

$Heart1(x,y,z) = (x^2 + y^2 + 2z^2 - 1)^3 - x^2y^3$

$Heart2(x,y,z) = x^6 + (3y^2 + 6z^2 - 3)x^4 + (3y^4 - y^3 + (12z^2 - 6)y^2 + 12z^4 - 12z^2 + 3)x^2 + y^6 + (6z^2 - 3)y^4 + (12z^4 - 12z^2 + 3)y^2 + 8z^6 - 12z^4 + 6z^2 - 1$

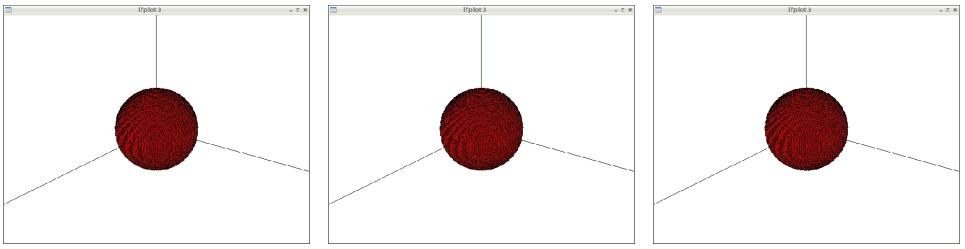
$Heart2h(x,y,z) = ((x^2 + (3y^2 + 6z^2 - 3))x^2 + (((3y - 1)y + (12z^2 - 6))y^2 + (12z^2 - 12)z^2 + 3))x^2 + ((y^2 + (6z^2 - 3))y^2 + ((12z^2 - 12)z^2 + 3))y^2 + ((8z^2 - 12)z^2 + 6)z^2 - 1$

$Line1(x,y,z) = (x^2 + y^2 + 2z^2 - 1)^2 + (x + y + z - 1)^2$

$Line2(x,y,z) = ((x^2+(2y^2+4z^2-1))x+(2y+2z-2))x+((y^2+(4z^2-1)y+(2z-2))y+((4z^2-3)z-2)z+2$

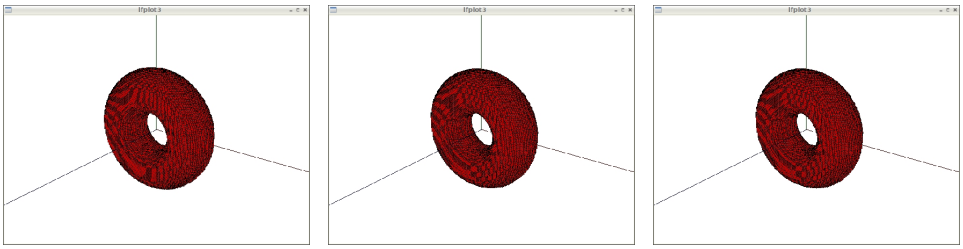
5.1 アフィン演算を用いた場合の結果

アフィン演算は，複雑なプログラミングを必要とするため，試作にあたって，精度保証付き計算で定評のある KV ライブラリ ver.0.4.24[1] を用いた．素朴な区間演算による区間評価，アフィン演算，平均値形式を用いた実行結果を図 1~ 図 7 に示す．また，判定されたボクセル数と区間評価部分の計算時間を表 1 に示す．*Ball* や *Torus* では，素朴な区間評価，アフィン演算，平均値形式で得られた図に差がないように思われるが，表 1 に示すように，素朴な区間評価では，かなり多くのボクセルが判定されている．*Heart*1, *Heart*2, *Heart*2*h* は同じ陰関数であるが，演算順序が異なることを示している．これらによると，表現が一番簡潔な *Heart*1 での結果が一番よい．これは，区間演算の特徴である．*Heart*2*h* は，*Heart*2 をホーナー法で評価したものである．一般的には，与えられる数式が簡潔になっている保証はない．そのため，*Heart*2*h* での結果を見るべきである．*Line*1, *Line*2 も同様に，同じ陰関数であるが，ほぼ球面と平面の交わりを表している．このように，1 つの既約な式で表された 2 つの曲面の交線についても洩らすことなく描くことができている．



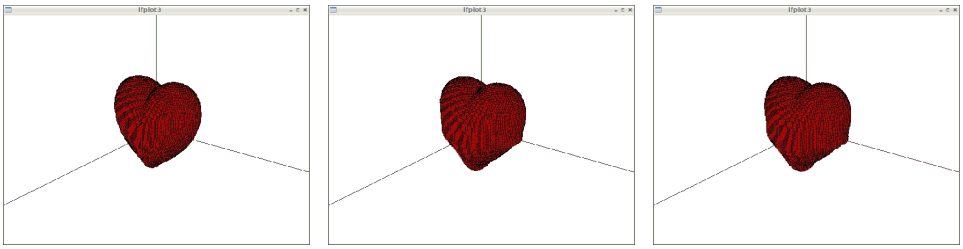
(a) 素朴な区間演算　(b) アフィン演算　(c) 平均値形式

図 1: $Ball(x,y,z) = 0$



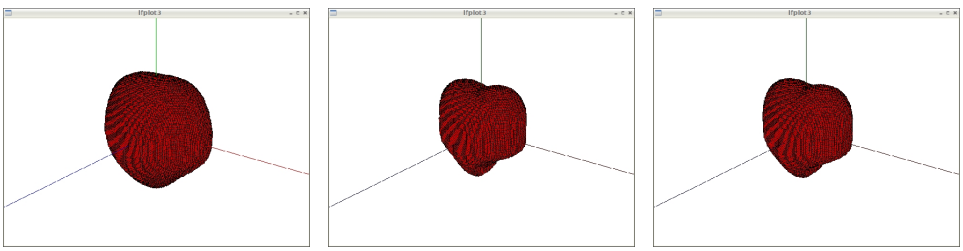
(a) 素朴な区間演算　(b) アフィン演算　(c) 平均値形式

図 2: $Torus(x, y, z) = 0$



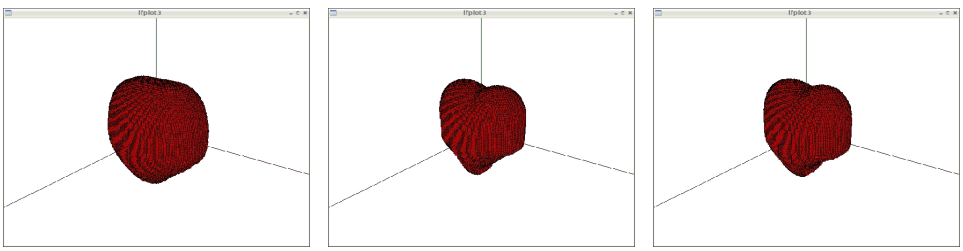
(a) 素朴な区間演算　(b) アフィン演算　(c) 平均値形式

図 3: $Heart1(x, y, z) = 0$



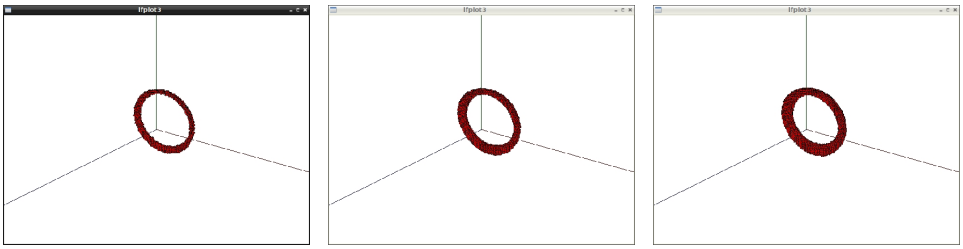
(a) 素朴な区間演算　(b) アフィン演算　(c) 平均値形式

図 4: $Heart2(x, y, z) = 0$



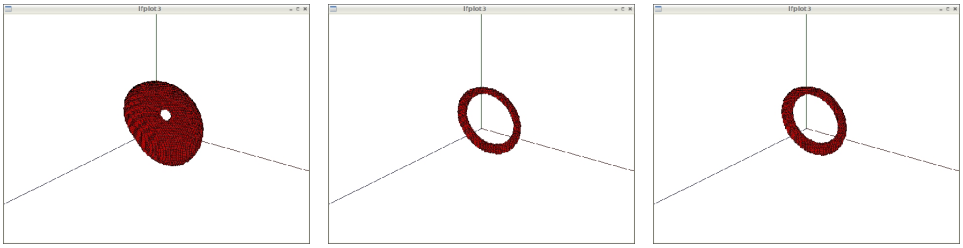
(a) 素朴な区間演算　(b) アフィン演算　(c) 平均値形式

図 5: $Heart2h(x, y, z) = 0$



(a) 素朴な区間演算　(b) アフィン演算　(c) 平均値形式

図 6: $Line1(x, y, z) = 0$

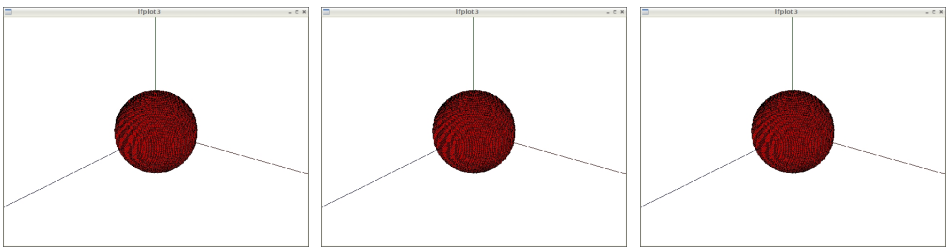


(a) 素朴な区間演算　(b) アフィン演算　(c) 平均値形式

図 7: $Line2(x, y, z) = 0$

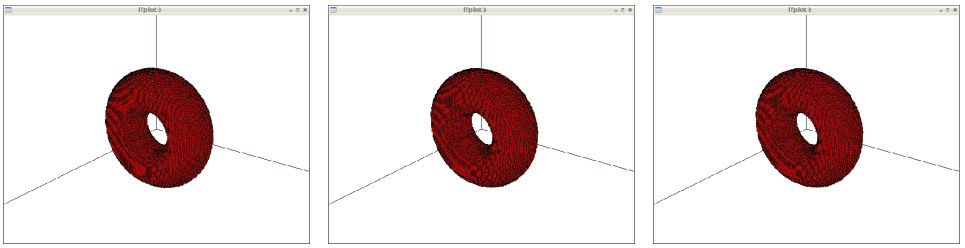
5.2 原点シフトを用いた場合の結果

原点シフトする場合には，筆者の一人が開発してる Asir に組み込んだ区間演算機能 [4] を用いて，試作を行った．比較のため，原点シフトは，すべてのボクセルで行っている．素朴な区間評価，平均値形式，原点シフトによる区間評価の実行結果を図 8~ 図 11 に示す．また，判定されたボクセル数と区間評価部分の計算時間を表 2 に示す．これらの結果より，原点シフトした区間評価は，アフィン演算で得られる図よりも，判定されているボクセル数が少ないことがわかる．これは，零点を含まないのに判定された余分なものが少ないということを意味する．



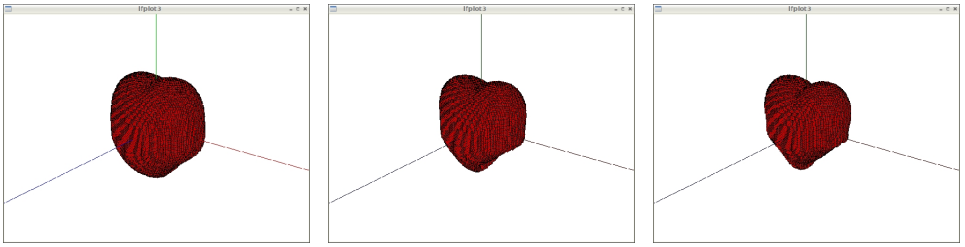
(a) 素朴な区間演算　(b) 平均値形式　(c) シフト

図 8: $Ball(x, y, z) = 0$



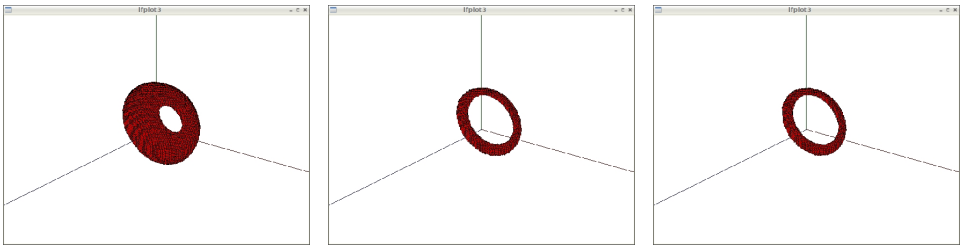
(a) 素朴な区間演算　(b) 平均値形式　(c) シフト

図 9: $Torus(x, y, z) = 0$



(a) 素朴な区間演算　(b) 平均値形式　(c) シフト

図 10: $Heart2h(x, y, z) = 0$



(a) 素朴な区間演算　(b) 平均値形式　(c) シフト

図 11: $Line2(x, y, z) = 0$

表 1: アフィン演算と平均値形式の判定されたボクセル数と計算時間

	素朴な区間評価	（秒）	アフィン演算	（秒）	平均値形式	（秒）
Ball	19256	0.47	19808	2.59	20192	3.60
Torus	66824	1.80	31808	14.25	33416	14.31
Heart1	26156	0.88	32844	6.79	41332	7.37
Heart2	281220	4.84	70396	56.30	100840	36.49
Heart2h	203708	3.41	50056	27.25	79744	23.12
Line1	1049	0.71	2614	4.81	3708	5.92
Line2	28962	1.73	2638	11.54	3897	12.90

時間は，区間評価部分のみである．

表 2: 原点シフトと平均値形式の判定されたボクセル数と計算時間

	素朴	(秒)	平均値形式	(秒)	原点シフト	(秒)
Ball	19256	0.42	20192	1.08	19943	17.13
Torus	50480	1.01	33104	2.91	31327	70.14
Heart2h	189740	1.59	72680	5.20	31663	198.70
Line2	23677	1.11	3653	3.52	2332	73.65

時間は，区間評価部分のみである．

6 おわりに

3 変数陰関数描画で，Interval Character の試作を行った．区間評価において，区間膨張がおこるため，素朴な評価，平均値形式，アフィン演算による方法，区間をシフトした素朴な評価を試した．Interval Character では，実行時間がかかる場合があるが，表示精度では区間をシフトした素朴な評価がよい．実行時間も考慮すれば，アフィン演算を用いる方法がよいが，計算順序に依存する部分が大きい．どの区間評価法に置いても，区間膨張のために，実際には零点が無いにもかかわらず判定されるボクセルがある．このボクセルを少なくするためには，ボクセルを再分割して，計算しなおすことが考えられるが，その分，実行時間が必要となる．今後，指定した部分に再計算するような実装を考える必要がある．

参 考 文 献

[1] 柏木雅英，kv — C++による精度保証付き数値計算ライブラリ，http://verifiedby.me/kv/

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

[3] 平野照比古，区間演算による多項式の評価について，数式処理，日本数式処理学会，**8**(4), 2002, 4–6.

[4] 近藤祐史，野田松太郎, 数式処理と区間演算の結合，数式処理，日本数式処理学会，**1**(2), 1992, 2–5.

[5] 近藤祐史，齋藤友克，数式処理における関数零点の描画，数式処理，日本数式処理学会，**12**(1), 2005, 33–46.

[6] 近藤祐史，兵頭礼子，村尾裕一，齋藤友克，Asir での３変数陰関数描画，数理解析研究所講究録 1843，2013, 140–145.

[7] 近藤祐史，兵頭礼子，村尾裕一，齋藤友克，Asir における陰関数描画 ifplot の改良，数式処理，**21**(2), 2015, 76–79.

[8] 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，**6**(2), 1998, 2–21.

[9] T. Saito, Y. Kondoh, Y. Miyoshi, T. Takeshima, Faithful plotting of real curves defined by bivariate rational polynomials, 情報処理学会論文誌, **41**(4), 2000, 1009–1017.

[10] J. Stolfi, L.H. Figueiredo, An Introduction to Affine Arithmetic, TEAM Tend. Mat. Apl. Comput. **4**(3), 2003, 297–312.