数 値 解 析 37−10
（1991. 7. 17）

4元数を用いた3次元フラクタルの生成とコンピュータグラフィックスへの応用

松坂 泰洋* 野村 昌伸* 島崎 眞昭**
九州大学工学部情報工学科* 九州大学大型計算機センター**

Abstract

4元数を用いた3次元フラクタルの生成とコンピュータグラフィックスへの応用について述べる。まず最初に4元数について述べる。次に4元数 z,q の関数 $f(z)=z^3+q$ のマンデルブロー集合の計算と可視化について述べる。スーパコンピュータのベクトル計算によりマンデルブロー集合の計算のCPU時間はスカラ計算のCPU時間より大幅に短縮できた。

4元数は3次元空間において回転、投影、アフィン変換を含む図形の表現を行なうのに便利であり、特に回転の表現が簡単にできるという利点がある。次に4元数の写像関数を用いて生成した樹状の3次元フラクタルパターンを示す。ここで用いる4元数の写像関数は3次元空間の回転、投影、反射から成り立つ。

Fractals in Quaternions and their Application to Computer Graphics

Yasuhiro Matsusaka*, Masanobu Nomura* and Masaaki Shimasaki**
Department of Computer Science and Communication
Engineering, Kyushu University*
Computer Center, Kyushu University**
6 - 10 - 1, Hakozaki, Higashi-Ku, Fukuoka 812, Japan

Abstract

We discuss fractals in quaternions and their application to computer graphics. First we describe computation and visualization of Mandelbrot set for $f(z)=z^3+q$ in quaternions. Using a supercomputer, the cpu-time could be reduced greatly compared with that in scalar mode.

Quaternions can be conveniently used to represent operations in three dimensional space, including rotations, projections and affine transformations. And it is shown that a three dimensional tree-like pattern can be gererated using similar contraction mapping in terms of quaternion in the three dimensional space. Our quaternion functions consist of components for contraction, rotation and reflection in the three dimensional space.

1 はじめに

コンピュータグラフィックスにおいて、フラクタル[1]のもつ自己相似性の性質を利用することによって、非常に複雑な自然界のモデルが驚くほどリアルに表される。3次元以上の多次元空間のフラクタルは理論的には可能である。しかし、複素力学系においては、主に2次元空間上で扱われてきた。一方コンピュータグラフィックスにおいては、3次元モデルが必要である。3次元空間で自然界の風景を描く目的でフラクタルのもつ自己相似性の性質を利用した例としては、崖や山の風景[2]を描いた例がある。

複素反復方程式を使ったものとして、z^2+qで決定されるマンデルブロー集合[1]やジュリア集合[3]がある。また、4元数(quaternion)[4]関数のジュリア集合の3次元表示による可視化については[3]がある。しかし、4元数関数のマンデルブロー集合の3次元表示による可視化に関するものについてはまだ報告がなされていない。そこで、4元数z,qを用いて、z^3+qで決定されるマンデルブロー集合の可視化について述べる。マンデルブロー集合の計算方法そのものは簡単である。しかし、最大反復計算回数を大きくとると、計算時間はかなりかかる。そのため、マンデルブロー集合の計算にスーパコンピュータを用いた。その結果、計算時間の短縮に有効であることがわかった。

一方、最近コンピュータビジョンやロボティックスの分野において空間の位置や運動を表現するための数学的な手法の一つとして4元数が注目されてきている。4元数は3次元空間において、回転、投影、アフィン変換を含む図形の表現をするのに便利であり、ベクトルでは簡潔に表現できない図形操作、特に回転の表現が簡潔にできるという利点がある。

Kochの自己相似曲線関数やHata[5]の平面上のシダの葉のパターンを表す関数が、2次元上で自己相似写像関数により表されている。しかし、これらの関数は複素変数関数であるため、2次元図形の生成に限定される。コンピュータグラフィックスに応用するためには3次元図形のモデル化を考える必要がある。4元数の関数を用いると、3次元空間内の回転操作を含む自己相似写像関数が簡潔に表現できる。我々は、4元数の自己相似関数を用いて3次元空間内の樹状パターンが生成できることを示す。

2 4元数

4元数qは、1つの実部q_rと3つの虚部q_i,q_j,q_kからなり、その一般形は、

$$q = q_r + q_i i + q_j j + q_k k \tag{1}$$

で表される。i,j,kは複素単位であり、次の式に従う。

$$i^2 = j^2 = k^2 = ijk = -1, \tag{2}$$
$$jk = i, \quad ki = j, \quad ij = k, \tag{3}$$
$$kj = -i, \quad ik = -j, \quad ji = -k. \tag{4}$$

また、qの絶対値及びq^2は以下の式で表される。

$$||q|| = \sqrt{q_r^2+q_i^2+q_j^2+q_k^2}. \tag{5}$$
$$q^2 = q_r^2 - q_i^2 - q_j^2 - q_k^2 + 2q_r(q_i i + q_j j + q_k k). \tag{6}$$

3 4元数を用いたマンデルブロー集合

3.1 4元数によるマンデルブロー集合とその断面表示

複素平面におけるマンデルブロー集合を拡張したものとして、次のマンデルブロー集合を扱うことを考える。ここで、q,μは4元数である。

$$M = \{\mu : \lim_{n\to\infty} f_\mu^n(z_n) \not\to \infty, \quad f'_\mu(q_c)=0\} \tag{7}$$
$$f_\mu(q) = q^3 + \mu. \tag{8}$$

ただし、

$$f_\mu^n(q) = f_\mu(f_\mu(\cdots(f_\mu(q))\cdots)). \tag{9}$$

(8)より$q_c=0$となる。4元数の数列q_nを以下のように定義して計算を行なった。

$$q_0 = 0, \tag{10}$$
$$q_{n+1} = q_n^3 + \mu, \qquad n = 0,1,2,\cdots,n_{max}. \tag{11}$$

ただし、nは繰り返し回数で、n_{max}はその最大値である。このn_{max}の値が大きいほど、計算時間も多大になる。また、マンデルブロー集合を求めるための発散の判定条件は、$n \le n_{max}$において、

$$||q_n|| > 1 + ||\mu|| \tag{12}$$

を満たすことである。この条件を満たすときに計算を停止させ、nの値をICOUNTとして記録する。ICOUNT=n_{max}でありかつ、上記の判定条件を満たさない場合は、発散しないものとみなし、ICOUNT=0とする。4元数のμの値を変化させて、以下に示す空間を等間隔格子で分割し、各格子点で計算を行なう。

$$\mu_{rmin} \le \mu_r \le \mu_{rmax}, \tag{13}$$
$$\mu_{imin} \le \mu_i \le \mu_{imax}, \tag{14}$$
$$\mu_{jmin} \le \mu_j \le \mu_{jmax}, \tag{15}$$
$$\mu_{kmin} \le \mu_k \le \mu_{kmax}. \tag{16}$$

表 1: IQ のカラーチャートテーブル

IQ	0	1	2	3	4	5	6
色	黒	青	緑	シアン	赤	マゼンタ	黄

μの各格子点における ICOUNT を 7 で割った余りの値を IQ とし、IQ の値により色付けをおこなう。そのカラーチャートを表 1 に示す。

4 次元空間である上記マンデルブロー集合はそのままでは可視化は難しい。そのため、その 2 次元断面図を図 1 から図 5 に示す。但し、各図とも $n_{max} = 200$ である。

3.2 スーパコンピュータによるマンデルブロー集合の計算

マンデルブロー集合の計算は、式の形は簡単であるにもかかわらず、発散判定のための反復回数を多くとるために非常に時間がかかる。一方、各格子点ごとの計算は互いに論理的に依存性がなく、そのため並列計算が実行可能である。パイプラインベクトルコンピュータでは、複数の格子点の計算のベクトル化が可能である。より多くの格子点を扱うほど、ベクトル長が長くなる反面、必要なメモリの量も増える。4 次元の格子点を扱うとき、必要なメモリの量はとても多いので、2 次元断面上での格子点の計算はベクトル長及び必要なメモリの量の双方の面からみても妥当である。図 1 のマンデルブロー集合の計算を $n_{max} = 200$ として FACOM VP-2600 で行なったときの CPU 時間の例を表 2 に示す。これから、スーパコンピュータを使うことが計算時間の短縮に有効であることがわかる。また、スカラーモードとベクトルモードの CPU 時間の比は 15 倍以上にもなり、ベクトル計算の効率の高さが示されていることがわかる。

表 2: 図 1 のマンデルブロー集合の計算の CPU 時間比

格子数	スカラーモードの CPU 時間 T_s	ベクトルモードの CPU 時間 T_v	CPU 時間の比率 $\frac{T_s}{T_v}$
1024×1024	37.8[sec]	2.40[sec]	15.8

3.3 3 次元断面での可視化

図 1 から図 4 の断面図により、3 次元断面の表示により可視化を行なった時の輪郭の予測ができた。そこで、あらかじめ求めた境界点の座標を元に、4 角形のパッチを数多くつくり、それを用いて輪郭をつくった。その結果を図 6 に示す。但し、3 次元断面図のレンダリングの方法としてレイトレーシング法を採用した。

問題点は、レンダリングの計算時間がかなりかかったことである。これは、パッチの数が 4 万個に近い数になったためにプリミティブの交線計算に時間を費したことが原因であると考えられる。

4 4 元数によるフラクタルの生成とコンピュータグラフィックスへの応用

4.1 4 元数としての 3 次元空間ベクトル

4 元数 qについて、実部及び虚部を

$$q_r = 0 \tag{17}$$

$$q_i = x, \quad q_j = y, \quad q_k = z, \tag{18}$$

とすると、3 次元空間における点 (x, y, z) は 4 元数 q により

$$q = ix + jy + kz \tag{19}$$

と表される。ここで、qが単位ベクトルであるならば、$q^2 = -1$ である。

4 元数を用いる最大の利点は 3 次元空間における回転の表現が簡潔であることである。ここで、図 7 について、回転軸を n とし、$n_r = 0$ とする。3 次元ベクトル vを n のまわりに回転角θだけ n の上側から見て反時計方向に回転して得られるベクトルを $u(v, u$ は 4 元数) とする。ただし、v, u は 3 次元ベクトルを表すので、$v_r = u_r = 0$ とする。このとき、

$$u = qvq^{-1} \tag{20}$$

が成立することが知られている。ここで、qは以下のように表される。

$$q = \cos\frac{\theta}{2} + in_x \sin\frac{\theta}{2} + jn_y \sin\frac{\theta}{2} + kn_z \sin\frac{\theta}{2}. \tag{21}$$

但し、

$$n_i = \frac{q_i}{\sqrt{q_i^2 + q_j^2 + q_k^2}}, \tag{22}$$

$$n_j = \frac{q_j}{\sqrt{q_i^2 + q_j^2 + q_k^2}}, \tag{23}$$

$$n_k = \frac{q_k}{\sqrt{q_i^2 + q_j^2 + q_k^2}}. \tag{24}$$

また、i,j,kの性質から次の式が容易に確認される。

$$q^{-1} = \cos\frac{\theta}{2} - in_x\sin\frac{\theta}{2} - jn_y\sin\frac{\theta}{2} - kn_z\sin\frac{\theta}{2}. \quad (25)$$

また、上の2式は次のように簡単に表せる。

$$q = \cos\frac{\theta}{2} + n\sin\frac{\theta}{2}, \quad (26)$$

$$q^{-1} = \cos\frac{\theta}{2} - n\sin\frac{\theta}{2}. \quad (27)$$

但し、

$$n = in_x + jn_y + kn_z. \quad (28)$$

回転を表す4元数表現には冗長度がなく、大変簡潔な

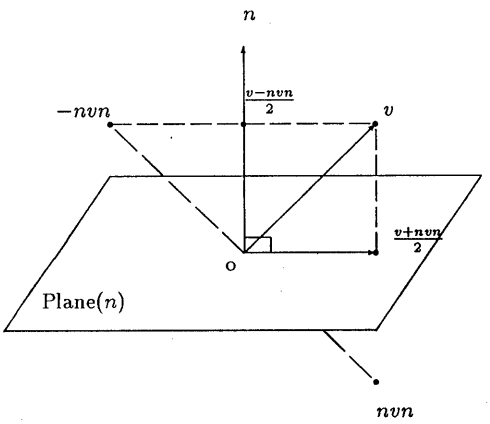

図7：vとその投影像及び鏡像との関係

表現となっている。3次元空間内の回転を表すその他の方法として3×3の行列で表す方法がある。しかし、行列は9個の要素を持つために冗長性が大きい。

ベクトルの投影像と鏡像もまた、4元数を用いて簡潔に表現できる。vをベクトル(ただし、$v_r = 0$)、nを単位ベクトルとする場合、

1. Line(n) 上への vの投影像：$\frac{v-nvn}{2}$
2. Plane(n) 上への vの投影像：$\frac{v+nvn}{2}$
3. Line(n) に関して対称な vの鏡像：$-nvn$
4. Plane(n) に関して対称な vの鏡像：nvn

で与えられる。図7に上記の式の関係を示す。

4.2 3次元フラクタルの4元数を用いた生成とコンピュータグラフィックスへの応用

Koch曲線は複素平面内のフラクタルの有名な例の一つである。それは、2つの自己相似な縮小写像関数を三角形に反復適用することによって得られる。また、Hata[5]はKochの関数に似た2つの自己相似写像関数

$$f_1(z) = (\frac{1}{2} + \frac{\sqrt{3}}{6})\bar{z} \quad (29)$$

$$f_2(z) = \frac{2}{3}\bar{z} + \frac{1}{3} \quad (30)$$

により三角形からしだの葉模様の図形が生成されることを示している。しかし、これらの関数は複素変数関数であるため2次元図形の生成に限定される。コンピュータグラフィックスに応用するには3次元図形のモデル化を考える必要がある。4元数の関数を用いると、3次元空間内の回転操作を含む自己相似写像関数が簡潔に表現できる。そこで、4元数の自己相似関数を用いて、3次元空間内の樹状パターンを生成させることについて考える。

まず、図8の三角形ABCについて、係数α,βには$0 < \alpha,\beta < 1$, $\alpha^2 + \beta^2 < 1$の条件がある。ここで、縮小写像関数f_1は次の写像変換を行なう。

$\triangle ABC \rightarrow \triangle A'B'C$：$y$軸のまわりの角度$\frac{\pi}{2} + \theta$の回転

$\triangle A'B'C \rightarrow \triangle A''B''C$：縮小比率$\sqrt{\alpha^2+\beta^2}$の縮小

$\triangle A''B''C \rightarrow \triangle A'''B'''C$：$xy$平面に関する鏡像

これから、$f_1(v)$は

$$f_1(v) = \sqrt{\alpha^2+\beta^2}kq_1vq_1^{-1}k \quad (31)$$

ただし、

$$v = ix + jy + kz \quad (32)$$

$$q_1 = \cos(\frac{\pi}{4} + \frac{\theta}{2}) + j\sin(\frac{\pi}{4} + \frac{\theta}{2}) \quad (33)$$

$$\cos\theta = \frac{\alpha}{\sqrt{\alpha^2+\beta^2}} \quad (34)$$

$$\sin\theta = \frac{\beta}{\sqrt{\alpha^2+\beta^2}} \quad (35)$$

次に、図9の三角形ABCについて、縮小写像関数f_2は次の写像変換を行なう。

$\triangle ABC \rightarrow \triangle AB'C$：$z$軸のまわりの回転角$\varphi$による回転

$\triangle AB'C \rightarrow \triangle A''B''C$：縮小比率$1-\alpha^2-\beta^2$による縮小

$\triangle A''B''C \rightarrow \triangle AB'''C'''$：$z$軸上の平行移動

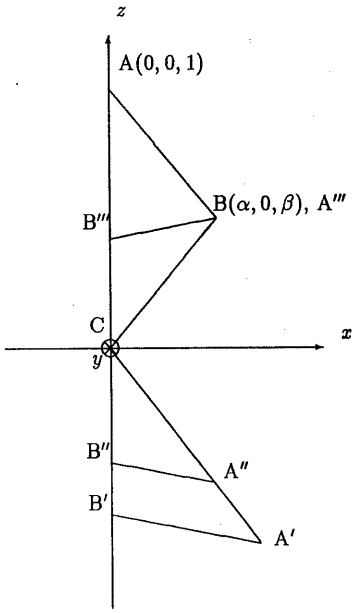

図 8: 縮小写像関数 f_1による写像

これから、f_2は

$$f_2(v) = (1-\alpha^2-\beta^2)q_2 v q_2^{-1} + k(\alpha^2+\beta^2) \quad (36)$$

ただし、

$$v = ix + jy + kz \quad (37)$$

$$q_2 = \cos\frac{\varphi}{2} + k\sin\frac{\varphi}{2} \quad (38)$$

図 10 は $\triangle ABC$に $f_1(v), f_2(v)$ を反復適用して得られた 3 次元フラクタルの樹状のパターンを示す絵である。ここで注意すべき点は 2 次元パターンの単なる回転によって得られる回転対称の 3 次元の絵ではなく、3 次元のフラクタルであることである。

また、図 11 に 3 次元フラクタルの樹状パターンを応用して得られた木の絵を示す。但し、これはレイトレーシングで描いたものである。リアルな絵が簡単に描かれていることがわかる。このことから、4 元数を用いて生成した 3 次元フラクタルをコンピュータグラフィックスに応用することは有効であることがわかる。

5 おわりに

4 元数のマンデルブロー集合を表示するために、まず 2 次元断面表示を行なった。ここで 3 次元表示を行なった時のある程度の輪郭の予測ができた。その後で、レイトレーシング法を用いて、4 次元のマンデルブロー集合の 3 次元断面表示を行なった。しかし、レイトレーシングによる 3 次元断面表示の際、レンダリングのための計算にかなりの時間を費した。その原因として輪郭を作るパッチの数が多いために交線計算にかなりの時間を費したことが考えられる。

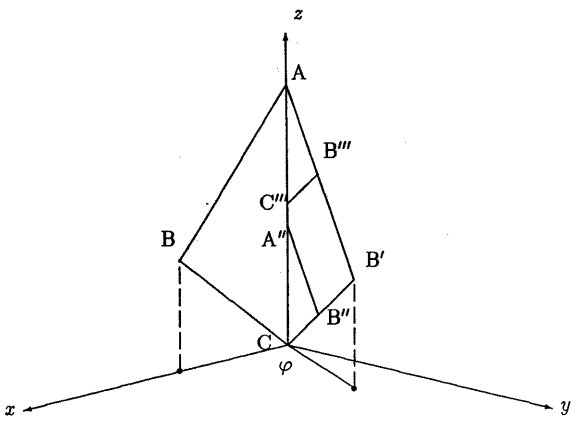

図 9: 縮小写像関数 f_2による写像

3 次元フラクタルの樹状パターンについては、4 元数を用いることで複雑な写像関数が簡略化でき、またかなりリアルで複雑な絵を簡単に描けることがわかった。

参考文献

[1] Benoit B. Mandelbrot : The fractal geometry of nature. W. H. Freeman and Company, New York, 1983.

[2] Alain Fournier, Don Fussell, Loren Carpenter : Computer Rendering of Stocastic Models, Comm. ACM Vol. 25, June 1982, pp.371-384.

[3] John C. Hart, Daniel J. Sandin, Louis H. Kauffman : Ray Tracing Deterministic 3-D Fractals. Computer Graphics, vol. 23, Num. 3, July 1989, pp.289-296.

[4] Edward Pervin, Jon A. Webb : QUATERNIONS IN COMPUTER VISION AND ROBOTICS. Carnegie Mellon University, Computer Science technical report, 1982, 150.

[5] M. Hata : On the Structure of the Self-Similar Sets. Japan J. Appl. Math. 2, 1985, pp.381-414.

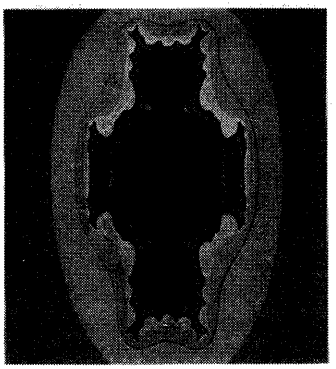

図 1: $-1.35 \leq \mu_r \leq 1.35, \quad -1.35 \leq \mu_i \leq 1.35,$
$\mu_j = 0.6, \quad \mu_k = 0$

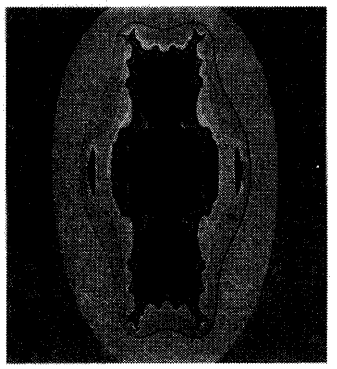

図 2: $-1.35 \leq \mu_r \leq 1.35, \quad -1.35 \leq \mu_i \leq 1.35,$
$\mu_j = 0.7, \quad \mu_k = 0$

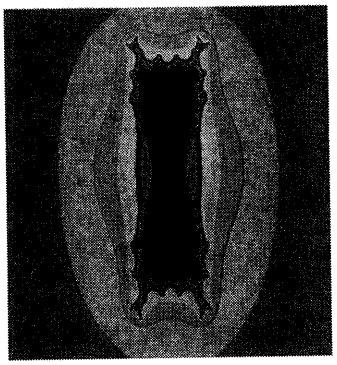

図 3: $-1.35 \leq \mu_r \leq 1.35, \quad -1.35 \leq \mu_i \leq 1.35,$
$\mu_j = 0.8, \quad \mu_k = 0$

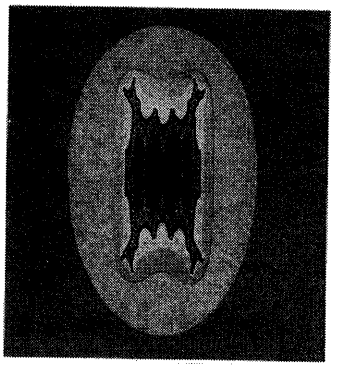

図 4: $-1.35 \leq \mu_r \leq 1.35, \quad -1.35 \leq \mu_i \leq 1.35,$
$\mu_j = 1.1, \quad \mu_k = 0$

図 5: $-0.250 \leq \mu_r \leq -0.225, \quad 0.590 \leq \mu_i \leq 0.615,$
$\mu_j = 0.7, \quad \mu_k = 0$

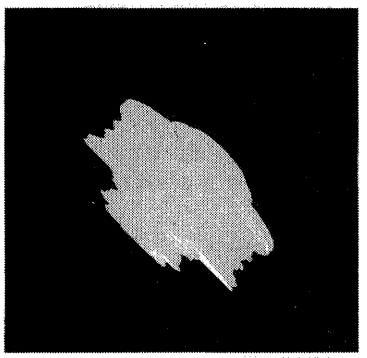

図 6: $-1.35 \leq \mu_r \leq 1.35, \quad -1.35 \leq \mu_i \leq 1.35,$
$0.0 \leq \mu_j \leq 1.35, \quad \mu_k = 0$

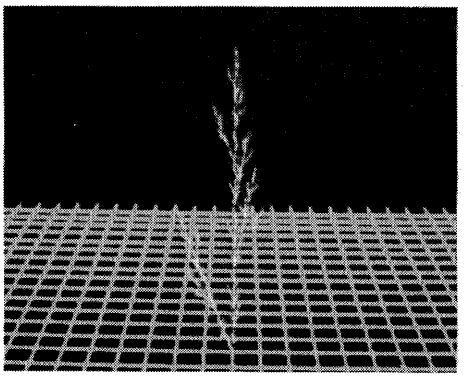

図 10: $\alpha = 0.2, \quad \beta = 0.5$

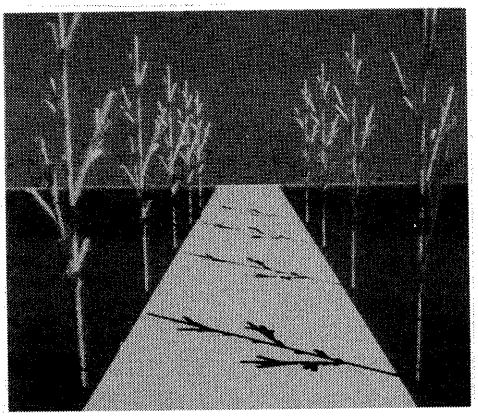

図 11: $\alpha = 0.2, \quad \beta = 0.5$