グラフィクスとＣＡＤ 26－4
（１９８７．５．１８）

# ３次元ランダム・フラクタルによる テクスチャ生成に関する検討

岡田 稔　多賀 雄伸　横井 茂樹　鳥脇 純一郎
名古屋大学 工学部 情報工学科

２次元ランダム・フラクタルを拡張した３次元ランダム・フラクタルを用いることにより、３次元的な連続性を持ったゆらぎを生成できる。これは、特に自然に生成された物質、不均質材料のテクスチャの画像生成に有用である。従来の手法（タイプＩ）による３次元ランダム・フラクタル生成では逐次的にフラクタル値を３次元配列に格納して行くため、膨大な記憶容量を必要とするにもかかわらず、画像生成の際に物体表面のテクスチャを得るときに実際に使用するフラクタル値はそのごく一部にすぎなかった。

本稿では、ランダム・フラクタル生成手順が木構造であることに注目し、必要な３次元位置におけるフラクタル値のみを計算し、使用することによって記憶容量削減をはかるための３つの手法に関して検討したので報告する。タイプⅡとして木構造リスト探索による方法、タイプⅢとして疑似乱数の種と生成乱数の一義性を利用した方法、タイプⅣとしてタイプＩとタイプⅢの複合法について述べる。

# A STUDY ON SOLID TEXTURE GENERATION BY USING 3-DIMENSIONAL RANDOM FRACTAL

Minoru OKADA, Yushin TAGA, Shigeki YOKOI and Jun-ichiro TORIWAKI
Dept. of Information Eng., Faculty of Eng., Nagoya university
Furo-cho, Chikusa-ku, Nagoya, 464 Japan

The 3-dimensional random fractal extended from the 2-dimensional version is useful for rendering a texture of inhomogenious natural materials. The 3-dimensional random fractal generation procedure requires a 3-dimensional array to store the fractal values. We discuss the method to reduce the memory requirement.

In this paper, we propose and compare three types of techniques. They are, a technique using tree structure list search, a technique to assign a random number for each node in the tree structure, and a method to combine above two techniques.

## 1　はじめに

フラクタル理論[1]に基づく自己相似性を拡張した、統計的自己相似性を持つランダム・フラクタルは、自然に生成された形状や性状を表現、生成する手法として有用[2]であり、画像補間、復元への応用も試みられている。例えば1次元のランダム・フラクタルは海岸線など[3]の表現に、また、2次元のランダム・フラクタルは地形、山肌など[4][5]の表現に適し、2次元テクスチャ・マッピング技法にも応用されている[5][6]。

すでに筆者らはこれらの考え方を3次元に拡張した3次元ランダム・フラクタルの生成手法[7][8]を提案し、この手法を用いたソリッドテクスチャ生成への応用を試みることによって、大理石、木目などの自然に生成された不均質材料の表現に有用であることを示している。しかし、フラクタル値を3次元配列に記憶する必要があるために大きな記憶容量を必要とし、その削減が大きな問題点として残されていた。ところが、実際に3次元ランダム・フラクタルをテクスチャ生成に利用する場合、物体表面の位置でのフラクタル値のみが必要とされる。

そこで本報告では、膨大な記憶容量を必要とする3次元配列をあらかじめ確保することなく、必要とする3次元ランダム・フラクタルのみを生成することによって、記憶容量削減を図る手法について検討した。

## 2　3次元配列を用いるタイプI

本章では、中点変位法[2]を3次元に拡張することによって3次元ランダム・フラクタル格子を3次元配列に得る手法[7][8]について簡単に述べる。3次元を充填する基本格子としては立方体、正4面体などが考えられるが、ここでは立方体を基本格子としている。立方体格子を用いると、一連の再帰処理は概ね図1に示すような8分木構造となるが、以下、その1再帰レベルでの処理について示す。

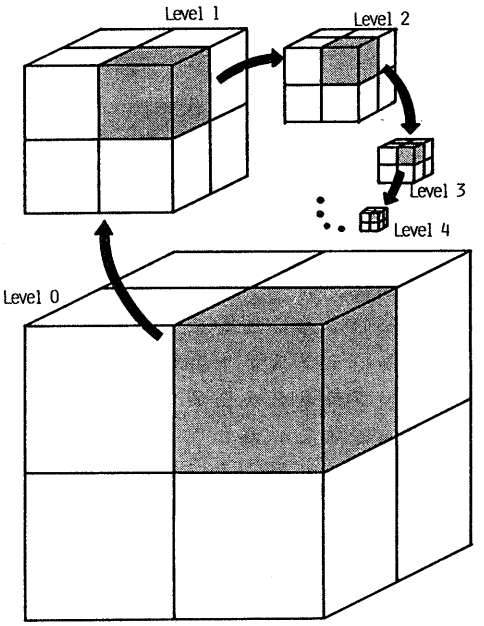


図1　8分木構造による3次元ランダム・フラクタル格子の生成

（1）格子の辺の中心点の値Deを辺の2端点の値Dvi(i=1,2)の平均値に対して乱数g・Lr(n)によって変動を加える（式(1)、図2参照）。Lr(n)は再帰レベルnが大きくなるにつれて変動量を小さくするための項である。12の辺について同様に処理する。

$$De = \frac{Dv1^{(n-1)} + Dv2^{(n-1)}}{2} + g \cdot Lr^{(n)} \qquad (1)$$

ただし、$Lr^{(n)} = e^{-(n-1)\beta}$　　(2)

$Lr^{(0)} = 1$

n：再帰レベル　(n=1,2,・・・)

g：正規乱数　N（0.0 , 1.0）

$\beta$：減衰定数

（2）立方格子の面の中心点の値Dpをその平面の4頂点の値Dei（i=1,2,3,4．上記（1）項で定める）の平均値に対して乱数g・Lr(n)により変動を加える（式(3)、図3参照）。6面について同様に処理する。

$$D_p^{(n)} = \frac{1}{4}\sum_{i=1}^{4} D_{ei}^{(n)} + g \cdot L_r^{(n)} \qquad (3)$$

（３）格子の中心点の値Ｄｃを６近傍点、すなわち６面の中心点の値Ｄｐｉ(ｉ=1,2,・・,6)の平均値に対して乱数ｇ・Ｌｒ$^{(n)}$により変動を加える（式(4)、図４参照）。

$$D_c^{(n)} = \frac{1}{6}\sum_{i=1}^{6} D_{pi}^{(n)} + g \cdot L_r^{(n)} \qquad (4)$$

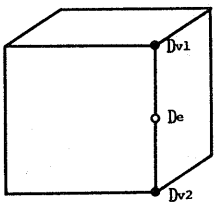

図２　格子の辺の中心点の値Ｄｅの計算

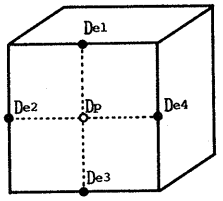

図３　格子の面の中心点の値Ｄｐ計算

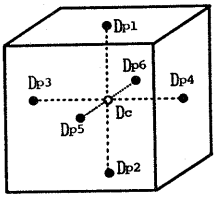

図４　格子の中心点の値Ｄｃの計算

以上の処理を終えた後、元の立方格子を均等に８個の立方格子に分割し、その各々の立方格子について次のレベルの再帰処理を適用する。これを必要な再帰レベルまで繰り返す。実際には計算機主記憶上に３次元配列（配列寸法$2^R+1$，最大レベルＲ）をとり、予め最外８格子点に初期値を与えておき、必要とするレベル回数の処理を繰り返し、逐次的に計算された格子点の値を格納してゆくことによって３次元フラクタル配列を得ている。ここで、求められる各格子点の値をフラクタル値と呼ぶことにする。この一連の処理によるランダム・フラクタルの生成手法をタイプＩとする。タイプＩでは、可能な最大レベルＲは有限であり、主記憶の容量によって制限される。

## ３　テクスチャ生成への応用

### ３－１　テクスチャ生成法

自然に生成されたテクスチャは不規則性状を持ち、解析的関数で表現するのは困難であるが、その基本的な構造は規則的なものであり、これに物理的な変形作用によるゆらぎが加わったものと考えるのは１つの自然な考え方であろう。我々はこの考え方を基にして次の方針で３次元テクスチャの生成を試みている[7][8]。

まず、規則的な形状、性状を表す原関数Ｇ（Ｐ）を定め、次に３次元フラクタル格子に基づいて３次元ゆらぎ関数Ｄ（Ｐ）をつくる。点Ｐにおいて、このＤ（Ｐ）の値に基づいて式(7)のように座標Ｐ’を定める。

$$P' = F(P,\ D(P)) \qquad (5)$$

ここに、Ｆは適当に選ぶ関数で３次元ゆらぎ関数Ｄ（Ｐ）の値に基づいて座標値を変位させるものである。すなわち、３次元の各格子位置がランダムな変動を受けて変化している。以下、Ｆをゆらぎ適用関数と呼ぶ。この結果、原関数の点Ｐにおける値Ｇ（Ｐ）が点Ｐ’において観測されるものとする（図７参照）。このような形で観測されたテクスチャを関数Ｇ’（Ｐ’）で表す。

### ３－２　大理石の生成への応用

前節の考え方をもとに、次式に示すような指数減衰の裾を持つ台形状の関数ｇｉ（ｚ）を用いて１つの層を表現し、これを複数個合成することによって層状構造を表現する原関数Ｇ（Ｐ）をつくる。

$$G(P) = \sum_{i=1}^{n} g_i(z)$$

$$g_i(z) = e^{-\alpha_i d_i} \quad (d_i > 0) \qquad (6)$$
$$= 1 \quad (d_i \leq 0)$$

ただし、$d_i = |z - c_i| - w_i/2$
；縁からの距離

$P = (x, y, z)$：座標
$w_i$：縞の幅
$c_i$：縞中心のz座標
$\alpha_i$：減衰定数（$\alpha_i > 0$）

また、ゆらぎ適用関数Fとしては、

$$F(P, D(P)) = P + (0, 0, k \cdot D(P)) \qquad (7)$$

を用いる。すなわち、z座標値のみを3次元ゆらぎ関数Dに基づいて変位させる。ここにkはゆらぎの寄与度を表す適当な定数である。以上の手法によって生成された画像を図5に示す。

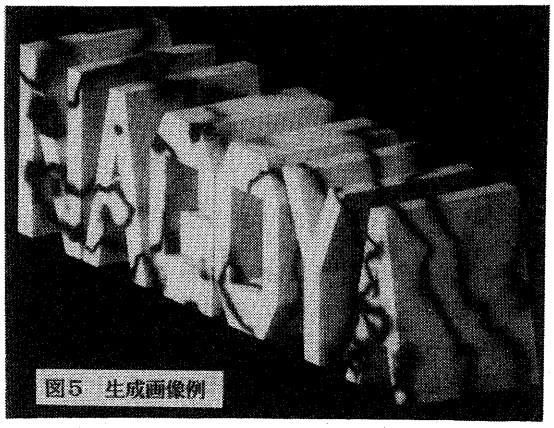

図5　生成画像例

この他に同心円よりなる原関数を半径方向に揺らがせることにより、木目模様の生成にも応用できるが本文では省略する。

## 4　リスト探索によるタイプII

タイプIでは、途中のレベルでのフラクタル値を記憶するのに、3次元配列を用いた。しかし、実際にテクスチャ生成では切り出される物体表面位置の点のみが求まれば良い。それらの点のフラクタル値の計算に必要な配列要素は全体の一部である。そこで、本章では、下位レベルで必要なフラクタル値のみを、リスト形式で記憶する手法タイプIIについて述べる。

### 4－1　フラクタル生成木の特徴

ランダム・フラクタルの生成において、フラクタル値の計算過程は図6に示すような木構造Gで示すことができる。ここでは、簡単にするため最大再帰レベルR＝4とした1次元での例を示す。図6において、木Gのノードは1次元配列各要素（これは3次元ランダム・フラクタル生成での3次元配列の各格子点に対応する）を表し、その中の数字は要素の番号を2進数で表したものである。この木Gは、以下に示す性質を有している。

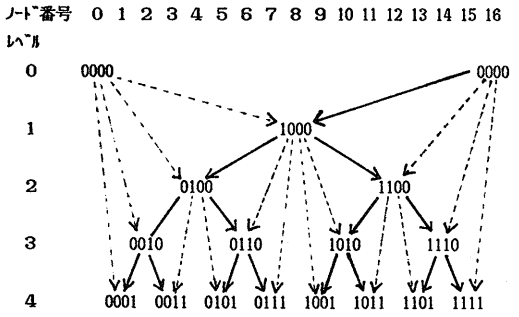


図6　1次元フラクタルの生成木（最大再帰レベルR＝4）

（1）各ノードは1レベル上に位置する親ノード1（図6実線アーク）と、少なくとも2レベル上（レベル1は特殊な場合とする）に位置する親ノード2（図6点線アーク）を有する。

（2）2進数で表された要素番号のLSBからみて、最初に1であるビットを第bビットとすると、そのノードのレベルnは、式(8)で与えられる。

$$n = R - b \qquad (8)$$

（3）2進数表現された任意の要素番号x（座標xに対応する）において、第bビットを0に、第（b＋1）ビットを1にすると親ノード1の要素番号x1が得られる。

$$x1 = or(and(x, comp(2^b)), 2^{b+1}) \quad (9)$$

ただしor,and,compは２進数でのビット演算で、orはビット毎の論理和、andはビット毎の論理積、compは１の補数を示す。

（４）任意のノードｘにおいて、第（ｂ＋１）ビットが１の時は第ｂビットに１を加え、０の時は１を引くと親ノード２の要素番号ｘ２が得られる。

$$x2 = x + 2^b \quad (and(x, 2^{b+1}) \neq 0)$$
$$x2 = x - 2^b \quad (and(x, 2^{b+1}) = 0) \quad (10)$$

## ４－２　リストの作成

任意のノードｘでのフラクタル値Ｆ（ｘ）は、２つの親ノードｘ１，ｘ２でのフラクタル値Ｆ（ｘ１），Ｆ（ｘ２）から式(11)によって計算される。

$$F(x) = \frac{F(x_1) + F(x_2)}{2} + g \cdot e^{-(n-1)\beta} \quad (11)$$

したがって、任意のノードｘが与えられると、親ノードをたどりながら、レベルをさかのぼっていくことが可能である。これは、任意のノードｘでのフラクタル値Ｆ（ｘ）を親ノード系列のみで計算することが可能であるということを示している。そこで、各ノードｘと、その点でのフラクタル値Ｆ（ｘ）、ｘ１のリストへのポインタｐ１、ｘ２のリストへのポインタｐ２を持った図７に示すリストを作成し、フラクタル値が必要となった座標から、親系列を順次たどって式(11)によって系列内で逐次的に各ノードのフラクタル値を計算することができる。

例として、１次元の最大再帰レベルＲ＝４の場合での各ノード（以下、ノード（ｘ）で示す）でのフラクタル値Ｆ（ｘ）を計算する手順を以下に示す。

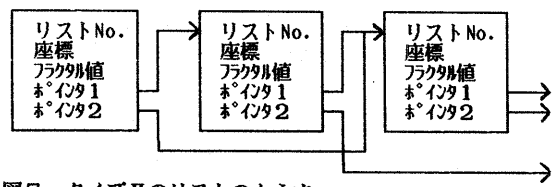


図７　タイプⅡのリストのようす

（１）ノード（０）と、ノード（１６＝$2^R$）のリストを確保し、フラクタル値の初期値を登録する。

（２）ノード（ｘ）をリスト中で探索し、存在すればフラクタル値Ｆ（ｘ）を参照する。リストに存在しないときは（３）以降の手続きによる。

（３）ノード（ｘ）の親ノード１の座標ｘ１を式(9)によって、また親ノード２の座標ｘ２を式(10)によって求める。ノード（ｘ１），ノード（ｘ２）それぞれについてリストを探索し、存在すればそのフラクタル値Ｆ（ｘ１）、Ｆ（ｘ２）を参照して式(11)によってフラクタル値Ｆ（ｘ）を求め、ノード（ｘ）についてのリスト確保し、それぞれのノードに関わるポインタｐ１，ｐ２を格納する。リストに存在しなければ（３）を再帰的に繰り返す。このとき、再帰は２つの親ノードがリスト存在した時に停止する。

（４）０レベルからポインタを逆にたどりながら、求めるノードに達するまでそれぞれのノードにおけるフラクタル値を式(11)によって計算し、格納して行く。

以後、他の要素のフラクタル値の計算においては親ノードの探索において、まずリストを探索し、そのノードのリストが既に存在しているときは、リスト中からフラクタル値を参照し、リストが無ければ（２）以降、同様の手順による。この一連の処理によるランダム・フラクタルの生成手法をタイプⅡとする。

## ４－３　タイプⅡの３次元への適用

１次元では親ノードは２つあったが、ｎ次元では 多くとも$2^n$個のフラクタル値計算に関与する親ノードを持つと考えられる。３次元においては、ｘ，ｙ，ｚ各成分に関して、２つの親ノードが存在する。ここで任意の３次元格子点$\mathbf{P}x$＝（ｘ，ｙ，ｚ）各成分に対応する再帰レベルを Ｌｘ,Ｌｙ,Ｌｚとし、ｘ軸に注目したとき、それらの大小関係に

より以下の３通りの場合が考えられる。

（１）Ｌx＞Ｌy，Ｌx＞Ｌz　の場合

ｘ軸に平行な辺上の中心点に位置し、図２の場合に相当し、ｘ方向の親ノード２個を持つ。

（２）ＬＸ＝Ｌy＞Ｌz　の場合

ｘ－ｙ平面上の中心点に位置し、図３の場合に相当し、ｘ方向、ｙ方向それぞれの親ノード２個、合計４個の親ノードを持つ。

（３）Ｌx＝Ｌy＝Ｌz　の場合

格子の中心点に位置し、図４の場合に相当し、ｘ方向、ｙ方向、ｚ方向それぞれに親ノード２個、合計６個の親ノードを持つ。

この条件はｘ，ｙ，ｚを入れ換えた任意の要素に適用される。これらの事実を考慮して、１次元の場合と同様にリストを作成する。ここで、リストの内容として、ｘｙｚ各座標値、フラクタル値、ｘｙｚ各レベル値、および６個のポインタが必要となる。

## ５　疑似乱数の性質を利用するタイプⅢ

### ５－１　疑似乱数の性質

タイプⅡの方法によると、フラクタル値を求めたい座標値の個数が多いと、新しい座標値の計算に必要な親ノードリストの探索の際に、最大の場合全てのリストを調べ、その中に探索するノードが存在するかどうかを判定する作業が必要となり、探索にかかる計算量が膨大となる。これは、あるノードのフラクタル値はどの要素から探索されても同じ値を持っている必要があることから生じる。

もし、どの要素からスタートしても、同じノードに同じ乱数の値を与えることができれば、過去の要素で探索したノードの情報を記憶していなくても済むことになる。ここでは、同じ乱数の値を得る手法を、疑似乱数の性質を利用して実現した。計算機で乱数を生成するための疑似乱数生成アルゴリズム（算術乱数）では、乱数の種が同一であれば、その結果生成される乱数系列も同一であるという性質を持っている[9]。すなわち、ノードの座標（ｘ，ｙ，ｚ）に対して、疑似乱数生成の種ｓを式(12)によって求める。

$$s = x(2^R+1)^2 + y(2^R+1) + z \qquad (12)$$

ここに $\{x,y,z \in I \mid 0 \leq I \leq 2^R\}$

これによって任意のノードに対して、一義的に乱数が生成される。この乱数を式(11)のｇに用いることにより、フラクタル値を求めたい座標値ごとに最上位ノードの情報をリストに記憶するだけでよく、新たな座標値のフラクタル値を求めるときにリストをクリアして新しいリストを作成すれば良い。このことにより記憶容量が大幅に削減され、リスト探索のための計算量も大幅に削減される。この方法によるランダム・フラクタル生成手法をタイプⅢとする。†

## ６　高速化を考慮した複合法、タイプⅣ

タイプⅠは、記憶容量の点で問題があるが計算量は少ない。一方、タイプⅡ、タイプⅢは、計算量の点で問題となるが記憶容量の点で優れている。そこでここでは、タイプⅠとタイプⅢの複合について検討する。

タイプⅠを実行するために必要とする最大レベルに相当する３次元配列の確保はできなくとも、上位の数レベル分のフラクタル値のための３次元配列は確保できる場合

---

注†　タイプⅢでは、疑似乱数の生成アルゴリズムによっては、生成された乱数に偏りが生じる場合があり、問題となった。しかし、これは乱数生成アルゴリズムの問題であり、本質的にタイプⅢの採用による障害ではないと考えられる。７章で述べる実験においては生成する乱数の個数を多くすることによって回避した。

が多い。そこで、記憶容量が確保可能なレベルに相当する３次元配列を用いて、タイプＩにより荒い解像度のフラクタル配列を生成しておき、これより下位のレベルの再帰処理はタイプⅢで計算する手法が考えられる。この手法をタイプⅣとする（図８）。

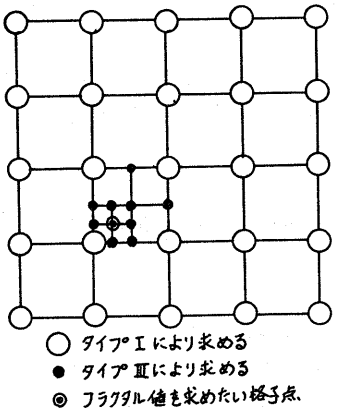


図8　タイプⅣのようす

## 7　実験と考察

前章までに述べた各手法それぞれについて、実際に計算機を用いて計算量、必要とする記憶容量の比較検討を行った。

### 7－1　必要とする記憶容量

タイプＩでは３次元配列にフラクタル値を生成しているため、最大再帰レベルＲの配列記憶容量Ｃ1[word]は式(13)で示される。

$$C1 = N^3 \qquad (13)$$

ただし、Ｎは一辺の格子点数、

$$N = 2^R + 1 \qquad (14)$$

タイプⅡでは、実際に切り出される物体の形状、大きさ、位置に依存する。物体の表面の位置のすべてのフラクタル値を計算するために必要な要素数は３次元配列の全要素は必要ないが、おそらくｏ（$N^3$）のオーダである。この各要素において、座標値、フラクタル値、ポインタなど決まった数の情報を記憶するリストが必要であり、結局、容量Ｃ2はｏ（$N^3$）と考えられる。

タイプⅢでは、指定された座標に関わる格子点列での一時的なリスト配列を要する。このリストの要素として、座標値ｘ，ｙ，ｚ、フラクタル値Ｆ、各軸での再帰レベルＬｘ，Ｌｙ，Ｌｚ、各軸方向の親ノードへのポインタＰｘ1，Ｐｘ2，Ｐｙ1，Ｐｙ2，Ｐｚ1，Ｐｚ2が必要である。また、木の各ノードでのレベルの格子内において、格子中心点１個、面の中心点６個、辺の中心点１２個の最大計１９個についてのリストを記憶する必要がある。したがって最大再帰レベルＲでのリスト配列に必要な記憶容量Ｃ3[word]は式(15)で示される。

$$C3 = 13 \cdot 19 \cdot R \qquad (15)$$

タイプⅣでは上位レベルをタイプＩで処理し、深いレベルをタイプⅢで計算している。タイプＩによる最大再帰レベルをＲ2とすると配列記憶容量Ｃ4[word]は、

$$C4 = N2^3 + 13 \cdot 19 \cdot (R - R2) \qquad (16)$$

ただし、Ｎ2はタイプＩの処理での一辺の格子点数である。

$$N2 = 2^{R_2} + 1 \qquad (17)$$

### 7－2　実験１

４個の手法の計算量を単純に比較する実験として、設定された最大再帰レベルＲのもとで全格子点のフラクタル値を計算した。ただし、実際に画像生成する場合において、全ての格子点についてフラクタル値を計算する必要がない、３次元物体の表面テクスチャ生成においては、この実験は非現実的なものである。図９に各手法について最大再帰レベルＲでの計算時間を示す。また、タイプⅣについてはＲが浅いので省略した。

### 6－3　実験２

実際に与えられた３次元形状を切り出す場合での計算量を比較する実験として、各

最大再帰レベルRでのフラクタル格子に内接する球（最小格子を量子とするディジタル球）を切り出す実験を行った。実験結果を図10に示す。ここでタイプIにおいては、3次元配列全ての要素を計算している。

## 8 むすび

3次元ランダム・フラクタルの生成手法として、フラクタル値を3次元配列上に生成するタイプI、フラクタル値をリスト構造で記憶するタイプII、疑似乱数の種と生成乱数の一義性を利用し、各ノードの情報の記憶容量を削減したタイプIII、そして、タイプIとタイプIIIの組み合わせによるタイプIVを提案した。その結果、タイプIVを用いることによって、利用可能な記憶容量に合わせた3次元配列を用いて、タイプIに比べ処理時間の大幅な増加なく画像生成に利用することができることを示した。

**謝辞** 本研究に際して、多くの有益なる討論を頂いた鳥脇研究室の諸氏に感謝する。本研究の一部は文部省科研費（一般研究(C)No.6102005）による。

**参考文献**

(1)B.B.Mandelbrot:"The Fractal Geometry of Nature",W.H.Freeman & Co. (1982).
(2)A.Fournier,D.Fussel,L.Carpenter:"Computer Rendering of Stochastic Models",CACM 25-6,pp.371-384 (1982).
(3)大石、野田、三澤、堀内：“フラクタル理論を用いた画像の符号化に関する一考察”,信学技報,IT85-16,pp.19-24 (1985).
(4)横矢：“フラクタルによる3次元複雑形状の解析とその応用”,信学技報,PRU86-23,pp.19-28 (1986).
(5)鶴岡、鈴木、木村、横井、三宅：“フラクタル手法を用いた物体の材質感表現”,NICOGRAPH'85論文集,pp.107-113 (1985).
(6)S.Tsuruoka,N.Suzuki,F.Kimura,S.Yokoi,Y.Miyake:"Rugged Texture Generation by Stochastic Models",Proc. of Science on Form,1.pp.297-304 (1985).
(7)岡田、堀、横井、鳥脇：“3次元フラクタルを利用した大理石の質感表現”, NICOGRAPH'86論文集,pp.97-105 (1986).
(8)横井、岡田、堀、鳥脇：“3次元ランダム・フラクタルを利用した大理石の質感表現”,昭62信学総全大1618,p.6-296 (1987).
(9)津田孝夫：“モンテカルロ法とシミュレーション”,培風館 (1969).

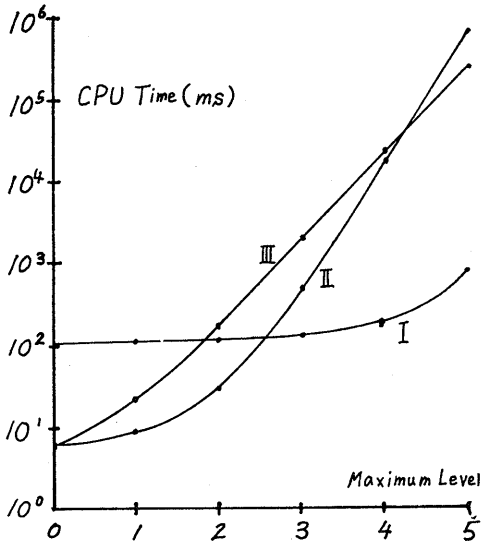


図9 実験1の計算時間

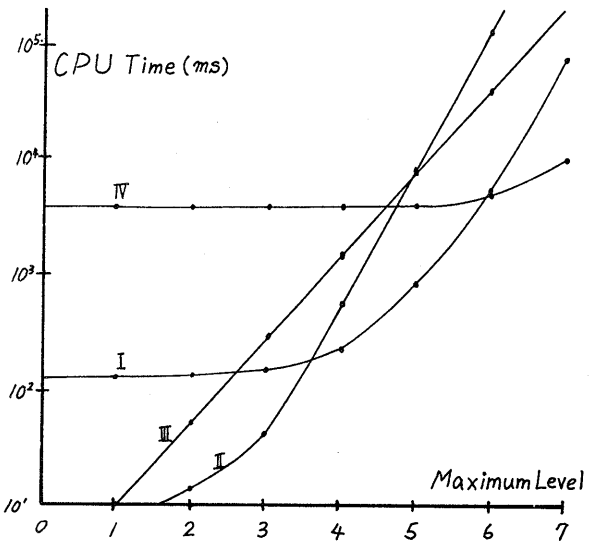


図10 実験2の計算時間（タイプIVにおけるタイプIによる再帰レベルをR2＝6に固定）