《小特集》

Low discrepancy 数列のコンピュータグラフィックスへの適用

土 井 章 男*

ABSTRACT In this paper, we applied low discrepancy sequences to the field of 2D and 3D sampling problems of image synthesis of Computer Graphics (CG). As low-discrepancy sequences, we selected Sobol's sequences, GFSR sequences, Good Lattice Points, and Hammersley Points. We evaluated these sequences in computational time, discrepancy, and RMS error, and show the effectiveness and the problems of low discrepancy sequences. We also proposed a 3D sampling method based on low-discrepancy sequences, and improve the performance more than 47 percents in comparison with the previous method.

1. はじめに

複雑な数値積分を解く手法としてモンテカルロ(Monte-Carlo)法が有名であり,その中では乱数が重用な役割を担っている.この乱数を乱数よりも一様に分布する数列(low discrepancy 数列)に置き換えて,モンテカルロ法を高速にした手法が準モンテカルロ(Quasi-Monte-Carlo)法である.ディスクレパンシィ(discrepancy)とは,乱数の一様な分布を表わす1つの尺度であり,その値が小さいほどその乱数は均一であると言える.Discrepancyの小さな数列をlow discrepancy数列と呼び,代表的なものとして,Halton数列[7,13,18],Sobol数列[15],GFSR数列[11,19],Good Lattice Points[12],Hammersley Points[8,12]などがある.

コンピュータグラフィックス(以後,CGと略す)の分野では,一般にlow discrepancy数列よりも乱数が用いられている.2次元空間でのサンプリングでは,画像生成時のエイリアスを防ぐために,jittered sampling[5,6]が使用される.3次元空間でのサンプリングでは,jittered samplingの拡張として,stratified samplingが提案されている[1].どちらの方法もサンプルする領域内(stratified samplingの場合は球面もしくは球面3角形)でサンプル数が等しくなるように決定される.

CGにおけるlow discrepancy数列の適用としては,*Niederreiter*がは論的にディスクレパンシィの小さなGood Lattice PointsとHammersley Pointsを提案し,これらの数列がCGに有効であると示唆した[12].しかしながら,*Nietherreiter*は実際の画像生成には適用しておらず,また,誤差評価も行っていない.最近では,画素のサンプリング[9],大域照明問題[10],面光源の輝度計算[14]にlow discrepancy数列を適用した例があるが,どのlow discrepancy数列が有効であるかは明確ではない.

本稿では,代表的なCGアプリケーションへの適用に際して,最も適したlow discrepancy数列を発見し,理論的にディスクレパンシィの小さい数列が必ずしもCGアプリケーションに有効でないことを実験的に示す.また,low-discrepancy数列を用いた3次元サンプリング法を提案し,本手法が従来の乱数を用いた方法に比べて,高速かつ良好なサンプリング特性を持っていることを示す.

本文の構成は,第2章で画像生成におけるエリアシングと従来の乱数を用いたサンプリング手法,そしてディスクレパンシィについて説明する.第3章では,Sobol数列,GFSR数列,Good Lattice Points,Hammersley Pointsの概略について述べる.第4章では,各low discrepancy数列とCGで最もよく使用されているjittered sampling[6]を比較検討し,計算時間,ディスクレパンシィ,精度(RMS誤差)の3点から2次元サンプリング結果を評価する.第5章では,low discrepancy数列を用いた適応的3次元サンプリング法とルックアップテーブル方式によるlowdiscrepancy数列生成法について述べる.第6章は結論である.

2. 画像生成におけるエリアシングとディスクレパンシィ

Applications of Low Discrepancy Sequences for Computer Graphics By *Akio Doi* (Faculty of Information and Software Science, Iwate Prefectural University)

*岩手県立大学ソフトウェア情報学部

2.1 画像生成におけるエリアシング

画像生成においてエリアシングが発生するのは，直線，多角形，色，輝度，時間などが連続的であるのに対して，その画像生成結果がサンプリング不足のため離散的になるためである．これを防ぐ方法(アンチエイリアシング)として，よく使用されているのがスーパーサンプリングである[4]．スーパーサンプリングでは，サンプリング領域に対してサンプリング数を増加させ，各サンプルの平均，もしくは重み付けした平均により計算する．

画素領域に対する従来のサンプリング手法としては，N-rooksサンプリング，poisson disk sampling，jittered sampling等がある[16]．これらの方法に共通することは，均一にサンプル点を分布させるため，サンプル点の位置制約を付けている点と乱数を用いている点である．N-rooksサンプリングでは，あらかじめ区切られた区画で，各サンプル点がお互いの行および列の区画に入らないよう制約している．poisson disk samplingでは，各サンプル点間の距離がある一定値以上になることを保証するため，一定値以上になるまでサンプリングを繰り返す．そのため，サンプリング点が増加すると計算効率が悪い．jittered samplingでは，まず，画素領域をサンプル数で区画化し，各区画内の位置は乱数により決定する．そのため，各区画内に必ずサンプル点が存在する．Jittered samplingは，上記サンプリング法の中で最も効率がよく，一般に使用されている．Jittered samplingのNが16, 64の場合の分布を**図1**に示す．

2.2 ディスクレパンシィ

S次元空間の単位立方体$I^s=[0,1]^s$に分布する点の集合$x^1, x^2, ..., x^N$を評価するひとつの尺度がディスクレパンシィである．

$x^1, x^2, ..., x^N \in I^s$を満たす点の集合と部分集合$G \subset I^s$が与えられたとき，$x^i \in G$を満たす点の数を数える関数$S_N(G)$を用意する．次に任意の点$x=(x_1, x_2, ..., x_s) \in I^s$に対して，式(1)で区切られる$s$次空間内の領域を$G_x$で表す．

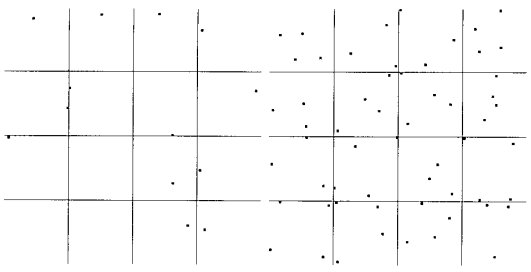

図1 Jittered Sampling (N=16, 64)

$$G_x=[0,x_1)\times[0,x_2)\times\cdots\times[0,x_s) \quad (1)$$

このとき，ディスクレパンシィは以下の式で定義される[2]．

$$D^*(x^1,x^2,\cdots,x^N)=\sup|S_N(G_x)-Nx_1x_2\cdots x_s|. \quad (2)$$

本ディスクレパンシィの定義では，領域G_xが各軸に垂直な区画になるため，関数$S_N(G)$の計算は容易である．*Dobkin*らは任意軸の区画を与えるディスクレパンシィを定義し，その計算方法について述べているが[20]，関数$S_N(G)$の計算は複雑になる．

2次元の簡単な例を示すと，総数1000点($N=1000$)のサンプリング点のうち，4等分された区画の左下の領域($x_1=0.5, x_2=0.5$)に245点入っていたとすると，ディスクレパンシィ$D^*(0.5, 0.5)$は$\sup|245-1000*0.5*0.5|$と計算される．

我々はディスクレパンシィを計算する上で，次の方法を用いた．

(1) 生成された数列を面積1の領域に分布させる．
(2) 領域内で，乱数a, bにより，左下隅を$(0, 0)$，右上隅を(a, b)とする長方形の区画を決定する．
(3) 区画の中の点nを数え，その場合のディスクレパンシィ$\frac{n}{N}-a*b$を計算する．

この方法を適当な回数，繰り返し，ディスクレパンシィにはその平均を用いた．この方法以外にも数列を発生させる度にディスクレパンシィを計算し，全体の算術平均を求める方法もあるが，今回の評価には用いなかった．これは，Good Lattice PointsおよびHammersley Pointsの生成順序が領域を端から順に埋めていくためである．

3. Low discrepancy数列

Low discrepancy数列として数値積分でよく使用されている代表的なLow-discrepancy数列には，Sobol数列，Halton数列，GFSR数列，Good Lattice Points，Hammersley Pointsなどがある．ここでは，*Neiderreiter*らがCGアプリケーションに有効であると提案しているLow-discrepancy数列のGood Lattice PointsとHammersley Points，そしてSobol数列とGFSR数列(Halton数列は，Hammersley数列の部分数列である)を取り上げる．サンプル点のばらつき度合としては，Good Lattice Points，Hammersley Points，Sobol数列，GFSR数列の順に前者ほど規則的である．

3.1 Sobol数列

Sobol数列[15]は，0から1の間のj番目の実数X_jを長

さ w の特別な2進数配列 V_i, $i = 1, 2, ..., w$ を用い，各 X_j の i 番目のビットがゼロでないという条件の時のみ，元の数列と2進数配列の要素 V_i の XOR（排他的OR）をとる．以降，各数列 V_i は連続的に生成される．Sobol数列の N が16, 64の場合の分布を**図2**に示す．

3.2 GFSR数列

Lビット GFSR (Generalized feedback shift register) 数列 u_i, $i = 1, 2, ...,$ は，以下のように定義される[11, 19]．

$$u_i = \sum_{j=1}^{L} b_{dj+i} 2^{-j} = 0.a_0 a_1 \cdots a_{k-1} = b_{d+i}\left(\frac{1}{2}\right)^1 + b_{2d+i}\left(\frac{1}{2}\right)^2 + \cdots \quad (3)$$

ここで，b_j, $j = 1, 2, ..,$ は，a linear feedback shift register と呼ばれる，ビット列である．*Lewis* と *Payne* は，r 次元の d の値は100 r より大きくすべきであると述べている[11]．全体の数列は r より小さな s を用いて連続的に生成される．

$$u_i = u_{i-r+s} XOR u_{i-r} \quad (4)$$

GFSR数列の N が16, 64の場合の分布を**図3**に示す．

3.3 Good Lattice Points

Good Lattice Points[12]は，優良格子法とも呼ばれ，サンプリングする個数 $N(\geq 2)$ と適当な整数 g $(1 \leq g < N)$ を用いて，

$$p_n = \left(\frac{n}{N}, \left\lceil \frac{ng}{N} \right\rceil\right) \quad for \quad n = 0, 1, \cdots, N-1. \quad (5)$$

このとき，$[t]$ は，実数 t から，$(0 < [t] < 1)$ への変換を意味する．この点列はN-rooksサンプリングとなる．Good Lattice Pointsの N が16, 64の場合の分布を**図4**に示す．

3.4 Hammersley Points

Hammersley Points[8) 12)]は，整数値 n(>0) のバイナリ表現 $n = a_{k-1}a_{k-2}\ldots a_1 a_0$ のビット列を反転させた，式(6)のビット反転関数 $\Phi(n)$ を定義する．

$$\Phi(n) = 0.a_0 a_1 \cdots a_{k-1} = a_0\left(\frac{1}{2}\right)^1 + a_1\left(\frac{1}{2}\right)^2 + \cdots \quad (6)$$

この結果，$\Phi(n)$ は，$0 \leq \Phi(n) < 1$ を満たし，次式のHammersley点列が定義される．$\Phi(n)$ のみを用いたものがHalton数列である．

$$p_n = \left(\frac{n}{N}, \Phi(n)\right) \quad for \quad n = 0, 1, \cdots, N-1 \quad (7)$$

Hammersley Pointsの N が16, 64の場合の分布を**図5**に示す．

4. 評　価

4.1 計算時間

各サンプリング法の生成効率を評価するため，我々は6種類のアルゴリズム，Sobol数列，GFSR数列，Good Lattice Points，Hammersley Points，乱数による方法，jittered samplingをC言語で実装し，HP9000/770 (PA-RISC@100MHz) HP-UX B10.01上のUNIX time機能を用いて計算時間を計測した．Sobol数列，GFSR数列，Good Lattice Points および Hammersley Points の実装に

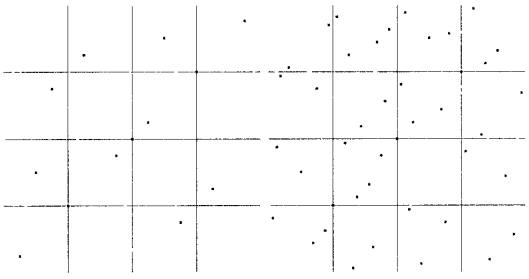

図2　Sobol数列 (N=16, 64)

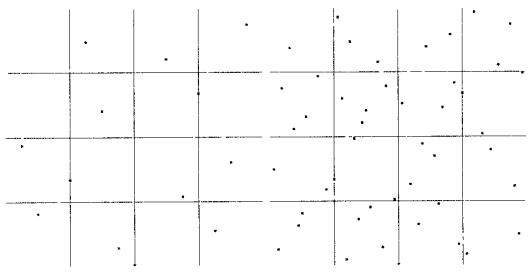

図3　GFSR数列 (N=16, 64)

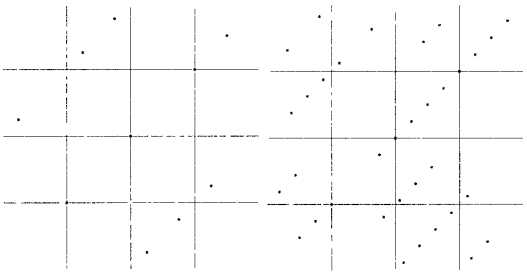

図4　Good Lattice Points (N=16, g=7)、(N=64, g=13)

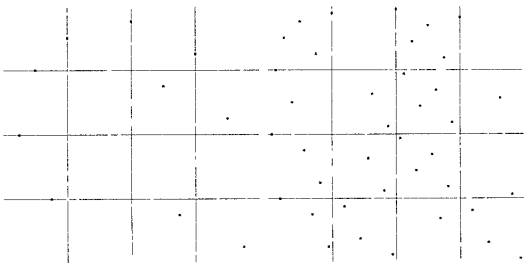

図5　Hammersley Points (N=16, 64)

は，それぞれ，文献15)，11)，12)，7)を参考にした．乱数による方法およびjittered samplingにはHP/UX上のrand()関数を用いた．乱数による方法は，画素領域のサンプル点を独立に乱数のみで決定し，Poisson Disk Samplingのようなサンプル点の取り直しは一切行っていない．**表1**，**2**は，512×512画素に対する斜め格子パターンおよびサーキュラーゾーンプレート(Circular zone plate)(**図6**)を生成する際のCPU時間の平均値である．斜め格子パターンの線の厚みは画素幅の半分である．

各列は1画素当りのサンプリング数(16サンプル，64サンプル，256サンプル)を表わし，数列発生時間と図形評価時間の両方を含む．両パターンにおける計算時間の違いは，各パターンの関数値計算による差である．我々の実験では，両パターンでSobol数列が最も速く，2番目がGood Lattice Pointsであった．その理由として，Sobol数列の生成方法が，条件分岐とXOR(排他的OR)のみの計算であり，Good Lattice Pointsでは，単純な割り算のみであるためである．

表1　512×512画素に対する計算時間（斜め格子パターン)(秒)

1画素あたりのサンプル数	(16)	(64)	(256)
Sobol数列	29.60	117.77	470.07
GFSR数列	84.12	348.82	1445.78
Good Lattice Points	34.92	138.88	555.27
Hammersley Points	48.82	193.94	773.92
rand() (HP/UX)	43.56	173.43	693.82
Jittered Sampling	82.67	327.87	1308.39

表2　512×512画素に対する計算時間(サーキュラーゾーンプレート)(秒)

1画素あたりのサンプル数	(16)	(64)	(256)
Sobol数列	7.95	31 08	123.49
GFSR数列	62.45	258.95	1080.48
Good Lattice Points	13.41	52.91	210.78
Hammersley Points	33.95	220.47	1223.81
rand() (HP/UX)	22.05	101.43	349.52
Jittered Sampling	27.25	107.89	428.11

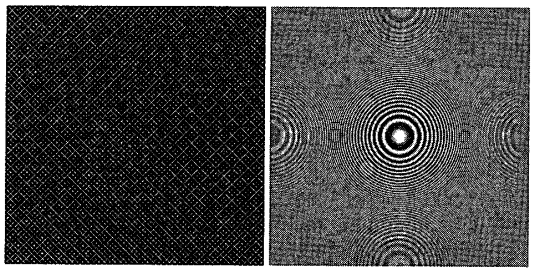

図6　斜め格子パターンとサーキュラーゾーンプレート

表3　256×256回計算したディスクレパンシィの平均値

1画素あたりのサンプル数	(16)	(64)	(256)
Sobol数列	0.027519	0.007785	0.002139
GFSR数列	0.027524	0.007887	0.002575
Good Lattice Points	0.042400	0.011252	0.003330
Hammersley Points	0.043744	0.013054	0.003513
rand() (HP/UX)	0.087642	0.060689	0.053967
Jittered Sampling	0.041541	0.015237	0.005717

4.2　ディスクレパンシィによる評価

CGにおける2次元サンプリング問題において，*Shirley*[16]はディスクレパンシィによる評価の有効性を報告している．**表3**は各low discrepancy数列のディスクレパンシィを256×256回計算した平均値である．ディスクレパンシィでの評価では，Sobol数列とGFSR数列が最も良い結果を示している．Good Lattice PointsとHammersley Pointsは規則的に空間を埋めていき，理論的にもそのディスクレパンシィは小さい．しかしながら，本実験のようにサンプリング数が少ない場合では，ディスクレパンシィが小さくならない．そして，Good Lattice PointsやHammersley Pointsのような規則正しいサンプリング点の場合，サンプリング数が少ないとモアレなどが生じやすいので注意が必要である．

4.3　精度

Jittered samplingで1画素あたり256点サンプリングした画像(ここでは，これを基準画像と呼ぶ)を用いて，他のサンプリング方法で作成した画像と比較した．基準画像に用いたテストパターンは，図6の斜め格子とサーキュラーゾーンプレートである．比較方法は画素ごとに差を求めて，その差の2乗和(RMS誤差)を計算する(式8)．この場合，各画素の値が相対的に基準画像の画素に近いほど，そのサンプル法のRMS誤差は小さくなる．

$$RMSError=\sqrt{\frac{\left(\Sigma\left(V_i^{jittered}-V_i^{others}\right)^2\right)}{sample\ num}} \qquad (8)$$

ここで，$V_i^{jittered}$とV_i^{others}は，jittered samplingと他のサンプリング法で計算した画素の値である．**表4**と**表5**は，格子方向と斜め方向パターンのRMS誤差の平均値である．両パターンで最も良い結果を示したのはGFSR数列とSobol数列であり，ディスクレパンシィの評価ともよく似た結果になった．各RMS誤差は，サンプル数nが増加するにつれて，$\sqrt{n}$に比例して減少している．

表4　RMS誤差（斜め格子パターン）

1画素あたりのサンプル数	(16)	(64)	(256)
Sobol数列	0 045855	0 020917	0 008507
GFSR数列	0.037369	0 012186	0 005494
Good Lattice Points	0 051985	0 023824	0 013595
Hammersley Points	0 046697	0 019056	0.008477
rand() (HP/UX)	0 067935	0.034250	0.017520
Jittered Sampling	0.036483	0.013496	0.000000

表5　RMS誤差（サーキュラーゾーンプレート）

1画素あたりのサンプル数	(16)	(64)	(256)
Sobol数列	2 687021	0.814257	0 521840
GFSR数列	2 778503	0 543694	0.317586
Good Lattice Points	3 417050	0 666778	0.355223
Hammersley Points	6 493344	2 045505	0 697995
rand() (HP/UX)	12 397014	6 184552	3 117691
Jittered Sampling	4 009379	1 069483	0 000000

5.　Low discrepancy数列を用いた適応的3次元サンプリング

5.1　3次元サンプリング

モンテカルロ法を用いる輝度計算[17]や鏡面反射を考慮したラジオシティ法では，球面上でのサンプリングが必要になる．*Arvo*は，2個の乱数を用いた球面上の3点から成る球面3角形内をサンプルする方法を提案した[1]．本論文では，*Arvo*の方法に対して，Low discrepancy数列を用いて，アルゴリズムの改良を試みる．本アルゴリズムの利点は，サンプリング数を増加させた時，サンプル点が重複しないことが保証されていることと計算時間が乱数より速い点である．

図7は，*Arvo*の方法により生成したものであり，**図8**,**図9**,**図10**,**図11**は，それぞれ，*Arvo*の方法の乱数をSobol数列，GFSR数列，Good Lattice Points，Hammersley Pointsに置き換えたものである．サンプリング点を増加させることにより，各サンプリング法特有のパターンが生成されるのは興味深い．計算時間に関しては，表1，表2に示した計算時間に比例し，乱数rand()とSobol数列を比べると約47パーセント以上，速度が向上する．一般にラジオシティ法では，適応的3次元サンプリングの計算は全体の中で大きな割合を占めるので，この速度向上は全体の処理速度に大きく利いてくる．

また，レイトレーシング法で画素ごとにサンプリング数を変更させたり，面光源の輝度を必要な精度までサンプリングしたい場合，生成されたlow discrepancy数列の計算を最初からやり直すのでは効率が悪い．このような場合には，Sobol数列やGFSR数列を用いることにより，あらかじめ区画された領域を守りながら，

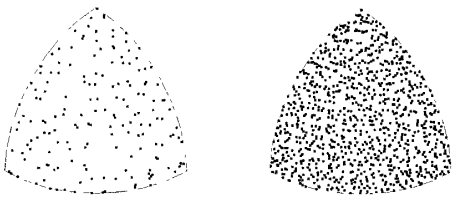

図7　Jittered Samplingによる3次元サンプリング（N=256, 1024）

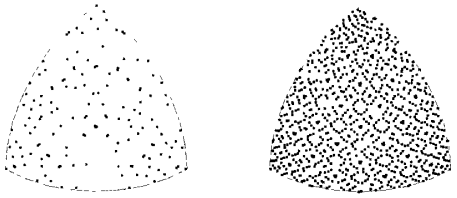

図8　Sobol数列による3次元サンプリング（N=256, 1024）

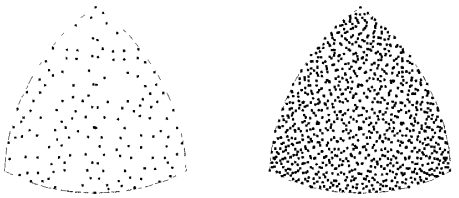

図9　GFSR数列による3次元サンプリング（N=256, 1024）

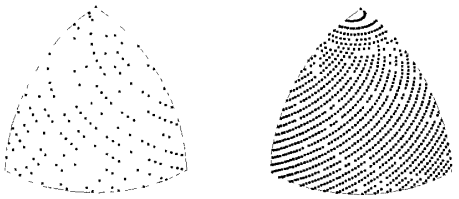

図10 Good Lattice Pointsによる3次元サンプリング（N=256, 1024）

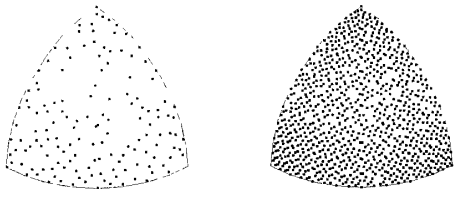

図11 Hammersley Pointsによる3次元サンプリング（N=256, 1024）

サンプル点を増加させることができる．**図12**は，Hammersley PointsとGFSR数列で4×4区画内で順にサンプル点を増加させた例である．サンプル点は50個づつ色を変更している．GFSR数列は順次区画を横断しながら，サンプリング空間を埋めていく．Good Lattice PointsやHammersley Pointsはサンプル数により発生する位置が変わるため，このような場合には向かない．この性質を利用することにより，必要な精度まで任意の回数，サンプリングを続けることができる．

5.2 ルックアップテーブル方式を用いたlow discrepancy数列

Low discrepancy数列は決定性ルールにより生成されるので，前処理によりあらかじめ数列をルックアップテーブルに格納しておき，表引きのみで数値を参照することが出来る．**表6**は，ルックアップテーブル方式によるGFSR数列の各パターンの計算時間(サンプル数は1画素あたり16点で512×512画素計算)であり，ルックアップテーブルからの表引き時間と図形評価時間の総和である．ルックアップテーブルに使用したメモリ容量は約32Kバイトである．本方式の利点はどんな数列にも適用できる点と，前処理さえ終了すれば，表引きの計算時間がたいへん短い点である．メモリ面でのトレードオフはあるが，最近のワークステーションレベルの計算機ではほとんど問題ないと思われる．また，別の利点として，適応的サンプリングに向いていないGood Lattice PointsやHammersley Pointsの数列でも，一旦数列をルックアップテーブルに格納してしまえば，表引きの順序で簡単に適応的サンプルを行うことができる．

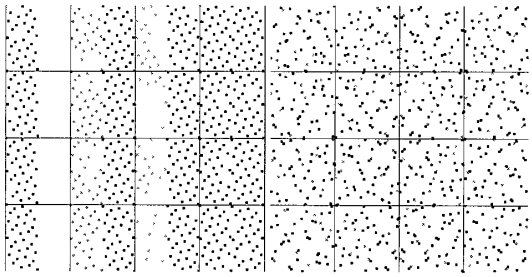

図12 適応的サンプリング（Hammersley PointsとGFSR数列）

表6 ルックアップテーブル方式を用いた計算時間（GFSR数列）

1画素あたりのサンプル数	16	64	256
斜め格子パターン	27.75	110.35	440.71
サーキュラーゾーンパターン	5.92	23.02	91 32

6. おわりに

本稿では，準モンテカルロ法で使用されているlow discrepancy数列(Sobol数列，GFSR数列，Good Lattice Points，Hammersley Points)と従来から使用されてきた乱数を用いたサンプリング法(jittered sampling)を比較し，その有効性を検証した．一般にlow discrepancy数列は計算効率がよく，均一にサンプリング出来るため，CGで使用されているほとんどの乱数計算に使用することが出来る．また，乱数を用いた方法に比べてサンプル点の重複を避けることができる．

Sobol数列は計算効率とばらつき具合の点でバランスのとれた数列である．GFSR数列は各点のばらつき具合が今回テストした数列の中で一番良いが，我々の実装ではその計算効率は良くなかった．これはサンプル点生成のために行列計算による方法[19]を用いているからであるが，より効率の良い実装方法の開発が必要である．また，Sobol数列とGFSR数列の生成パターンは，jittered samplingの生成パターンに近く，サンプル数を適応的に変更したい場合にも有効である．Good Lattice Pointsはアルゴリズムが簡単でインプリメントが容易であり処理速度も速い．ただし，Good Lattice PointsとHammersley Pointsは周期性が強いため使用には注意が必要である．Good Lattice PointsやHammersley Pointsは理論的にディスクレパンシィが小さいことが証明されているが，サンプル数の少ない画像生成における2次元のアンチエイリアスにはあまり向かない．

最近のCGの分野における，Low discrepancy数列の話題に関しては，文献21)が詳しい．特にレンダリング(画像生成)に適用した話題に関しては，文献22)，23)を参照されたい．

今後の研究として，より高速で実用的なlow discrepancy数列を発生させるアルゴリズムとディスクレパンシィの評価方法の開発が挙げられる．*Dobkin*の提案しているディスクレパンシィ評価方法[20]はそのひとつの解決案である．また，今回は各数列を評価する上で，最もよくCGで使用されているjittered samplingによるサンプリング結果と比較したが，各数列の理論的な誤差のオーダに関する議論も必要かと思われる．

参考文献

1) J. Arvo Stratified Sampling of Spherical Triangles, SIGGRAPH 95, 437/438 (1985)
2) P Bratley and B L. Fox ALGORITHM 659 Implementing Sobol's Quasirandam Sequence Generator, ACM Trans on

Mathematical Software, **14**-1, 88/100 (1988)
3) P Baratley, B L Fox, and H Niederreiter· Implementation and Tests of Low-Discrepancy Sequences, ACM Transaction on Modeling and Computer Simulation, **2**-3, 195/213, July (1992)
4) Foley, van Dam, Feiner, Hughes Computer Graphics, Principles and Practice, 2nd Edition, Addison-Wesley Publishing Company, 617/647 (1990)
5) R L Cook, T Porter, L Carpenter. Distributed Ray Tracing, SIGGRAPH 84, **3**-18, 137/145 (1984)
6) R L Cook. Stochastic sampling in computer graphics, SIGGRAPH 86, **5**-1, 51/72 (1986)
7) J Halton and G Smith· Radical-inverse quasi-random point sequence [G5], Communication of the ACM, **7**-12, 701/702 (1964)
8) J Halton and S. Zeremba The extreme and L2 discrepancies of same plane sets, Monatsh Math , **73**, 316/328 (1969)
9) S Heinrich and A Keller Quasi-Monte Carlo Methods in Computer Graphics Part I The QMC-Buffer, Technical Report, 242/94, University of Kaiserslautern, Germany (1994)
10) S Heinrich and A. Keller. Quasi-Monte Carlo Methods in Computer Graphics Part II The Radiance Equation, Technical Report, 243/94, University of Kaiserslautern, Germany (1994)
11) T G. Lewis and W H Payne. Generalized feedback shift register pseudorandom number algorithms, J ACM, **20**-3, 456/468 (1973)
12) H Niederreiter Quasirandom Sampling in Computer Graphics, Image Processing in Medicine, Remote Sensing and Visualization of Information, Latvian Academy of Sciences, Riga, 29/34 (1992)
13) H Niederreiter Random Number Generation and Quasi-Monte Carlo Methods, CBMS-NSF SIAM, Philadelphia, PA (1992)
14) R. Ohbuchi, M Aono Quasi-Monte-Calro Rendering Using Low-discrepancy sequence, IPSJ Sig of CG and CAD, 81-16, 91/96 (1996)
15) W Press, S Teukolsky, W Vetterling, and B Flannery Numerical Recipes in C, Second Edition, Cambridge Univ Press (1992)
16) P Shirley Discrepancy as a Quality Measure for Sampling Distributions, EUROGRAPHICS '91, 183/194 (1991)
17) P Shirley, B Wade, P M Hubbard, D Zareski, B. Walter, and D GreenBerg Global Illumination via Density-Estimation, Proceedings of the Eurographics Workshop '95 (Rendering Techniques '95), 219/230 (1995)
18) S Tezuka Polynomial Arithmetic Analogue of Halton Sequences, ACM Transaction on Modeling and Computer Simulation, **3**-2, 99/107 (1993)
19) S Tezuka· The k-dimensional distribution of combined GFSR sequences, Mathematics of Computation, **63**-206, 809/817 (1994)
20) D Dobkin, D Eppstein, and D Mitchell Computing the Discrepancy with Applications to Supersampling Patterns, ACM Trans on Cmputer Graphics, **15**-4, 354/376 (1996)
21) I Friedel and A Keller Fast Generation of Randomized Low-Discrepancy Point Sets, Monte Carlo and Quasi-Monte Carlo Methods 2000, Springer-Verlag, Berlin, 257/273 (2000)
22) T. Kollig and A Keller. Efficient Multidimensional Sampling, EUROGRAPHICS 2002, **21**-3, (2002)
23) A Keller· Strictly Deterministic Sampling Methods in Computer Graphics, (mental images technical report, 2001) in "Monte Carlo Ray Tracing", Siggraph 2003 Course No.44, San Diego (2003)