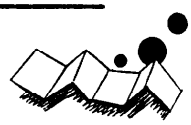
解　説



擬似乱数の発生法について†

伏　見　正　則††

1. はじめに

電子計算機を使って大規模なモンテカルロ実験等を行う場合には，大量の乱数が必要になるが，その供給方法としては，算術的方法が簡便なのでもっともよく用いられているようである．ちなみに，東京大学大型計算機センターの数学ライブラリ使用回数の統計によれば，一様乱数発生ルーチンは，最近つねに最上位のグループにランクされている．

算術的な乱数（擬似乱数）発生方法について解説するのが本稿の目的であるが，これに関する研究報告はきわめて多く，一方紙数には自ら限りがあるので，標準的な文献（例えば Knuth の著名な本[9]）に詳しく紹介されている話題は割愛して，二，三のトピックスについてのみ解説することにする．

2. 合同法乱数の多次元分布の格子構造

合同法とは，自然数 a, P，整数 b を適当に定め，非負の初期値（整数）x_0 を与えて，漸化式

$$x_{i+1} \equiv a x_i + b \pmod{P} \qquad (2.1)$$

によって“乱数列” $x_0, x_1, x_2, \cdots$ を作り出す方法のことである．$(\mathrm{mod}\ P)$ は，P で割り算をして余りを取ることを意味する．$P=$（使用する計算機で表現できる最大の整数）$+1$ とすれば，けたあふれを無視することによってこの操作が自然に行える（ただし符号ビットに1が侵入した場合の処置は別途に考慮する必要がある）ので，そうすることが多い．パラメタ a, b の選び方については，多くの文献にいろいろな推奨値が載っている．また，区間 $[0, 1)$ 上の一様乱数が必要な場合には，x_i/P が使われる．

合同法によって発生される数列には周期性があり，その周期は P 以下であり，したがって使用する計算機のけた数によって制約を受けることになる．また，数列の1周期分には同一の数が現われることはない．このことから，$(x_0, x_1, \cdots, x_{n-1})$, $(x_n, x_{n+1}, \cdots, x_{2n-1})$, … を座標とする点は，$n$ 次元(超)立方体内の P^n の格子点のうちの高々 P 個のところにしか配置されないことがわかる．したがって，これらの点はランダムではなく，実はきわめて規則的に並んでしまうおそれがあるが，確かにそのようになっていることを示したのが Marsaglia[14),15)] である．それによると，これらの点は格子状に規則的に並び，$(n!P)^{1/n}$ 枚以下の(超)平面の上にのってしまう．いくつかの P, n について $(n!P)^{1/n}$ の値を示したものが**表-1**である．また，**表-2**は Dieter[5)] によって求められた(超)平面の正確な枚数を示したものである．

合同法による乱数によって生成された2次元あるいは3次元の点の規則的な配置の例がいくつかの文献（例えば 13), 19) 等）に図示されているが，これらの図は P の比較的小さな値（512 とか 1,024 など）について作成されているので，実用的な大きさの P については，これほど悲劇的な結果にはならないであろうという感じを持たれた読者もいるかもしれない．しかし，表-1, 2に見られるとおり，必ずしもそうではないのである．一例として，$a=65539$, $b=0$, $P=2^{31}$ または 2^{32} として得られる発生法について考えてみよ

表-1　合同法乱数によって生成される n 次元空間内の点をすべて含む超平面の枚数の上界

P \ n	3	4	5	6	7	8	9	10
2^{16}	73	35	23	19	16	15	14	13
2^{24}	465	141	72	47	36	30	26	23
2^{32}	2,953	566	220	120	80	60	48	41
2^{35}	5,907	952	333	170	108	78	61	51
2^{36}	7,442	1,133	383	191	119	85	66	54
2^{40}	119,086	9,065	2,021	766	391	240	167	126

表-2　合同法乱数によって生成される n 次元空間内の点をすべて含む超平面の枚数 ($P=2^{31}$)

a \ n	2	3	4	5	6
65,533	32,765	15	15	15	15
258,585,933	22,107	1,115	257	69	31
414,536,077	27,307	1,115	209	91	41

† On Methods of Generating Pseudorandom Numbers by Masanori FUSHIMI (Department of Mathematical Engineering, Faculty of Engineering, University of Tokyo).

†† 東京大学工学部計数工学科

う．これは，かつてIBMのSSP (scientific subroutine package) の中にRANDUの名前で登録されていて，日本でも同じものが広く用いられ，現在でも時々使用されているのを見かけるものである．

まず，次の2式が成り立つ．

$$x_{i+1}\equiv 65539x_i\equiv(2^{16}+3)x_i \pmod P$$

$$x_{i+2}\equiv(2^{16}+3)x_{i+1}\equiv(2^{16}+3)^2x_i$$
$$\equiv(6\cdot 2^{16}+9)x_i \pmod P$$

これらから

$$x_{i+2}-6x_{i+1}+9x_i\equiv 0 \pmod P$$

が導かれる．xの値はいずれも0以上$P-1$以下であるから，この式の左辺の値は$-6(P-1)$以上で$10(P-1)$以下である．この範囲内でPで割り切れる整数は$-5P, -4P, \cdots, 8P, 9P$の15個である．したがって3次元空間内の点(x_i, x_{i+1}, x_{i+2})は，等間隔に並んだ15枚の平行な平面のいずれかにのってしまうことになる．

合同法は簡便な方法であるので今後もおそらくよく使われることであろうが，多次元の分布を作り出したい場合には，ここで述べた性質が悪影響を及ぼすかどうかを事前によく検討してみる必要があるであろう．用心のためには，時間は2倍以上かかることになるが，何通りかの乱数列から1つの乱数列を作り出して使う方法（例えば9）のp. 30, 18)のpp. 90-92）や，一般化した合同法[6]

$$x_{i+1}\equiv a_1x_i+a_2x_{i-1}+\cdots+a_{k+1}x_{i-k} \pmod P \quad (2.2)$$

などを使うのがよいであろう．また次節で述べるm系列を利用した一様乱数の発生法は，所要時間が単純な合同法に比べてごくわずか長いだけなので，便利な方法であろう．

3. m 系列を用いた一様乱数の発生方法

3.1 m 系 列

定数$c_1, c_2, \cdots, c_{p-1}$がいずれも0または1で，$c_p=1$であるとして，漸化式

$$a_i=c_1a_{i-1}+c_2a_{i-2}+\cdots+c_pa_{i-p} \pmod 2 \quad (3.1)$$

によって作り出される0と1とからなる数列$\{a_i\}$を考える．ただし初期値$a_0, a_1, \cdots, a_{p-1}$は，すべてが0ではないように選ぶものとする．$a_i$の値は，順列$(a_{i-1}, a_{i-2}, \cdots, a_{i-p})$によって一意に定まり，相異なる順列の個数は$2^p-1$である（0ばかりからなる順列は仮定によってないことに注意）から，数列$\{a_i\}$の周期mは，2^p-1を超えないことは明らかである．ちょうど$m=2^p-1$となるための条件は，特性多項式

$$f(x)=1+c_1x+c_2x^2+\cdots+c_px^p \quad (3.2)$$

がガロア体GF(2)上の原始多項式であることである．また，このとき$\{a_i\}$を最大周期列 (maximum-length linearly recurring sequence, 略してm系列）と呼ぶ．

3.2 Tausworthe の方法[16]

Tausworthe は，m系列$\{a_i\}$から相続く$l(\leqq p)$個の数を取り出してきて並べ，lビットの2進小数

$$u_k=0.a_{\sigma k+r+1}a_{\sigma k+r+2}\cdots a_{\sigma k+r+l} \quad (3.3)$$

を作り，系列$\{u_k\}$を区間$[0,1)$上の一様乱数列として使うことを提案した．ここに，rは非負の任意の整数であり，$\sigma(\geqq l)$は$\{a_i\}$から次々にlビットずつ取り出す間隔である．σを$\{a_i\}$の周期mと互いに素になるように選ぶと，$\{u_k\}$の周期もmとなる．そしてその1周期分には，0が$2^{p-l}-1$回，その他のあらゆるlビットの2進小数が2^{p-l}回ずつ現われる．したがって$p-l$があまり小さくなければ，$\{u_k\}$は（少なくとも頻度に関しては）ほぼ一様分布に従っているものと見なせる．Tausworthe はさらに，$\{u_k\}$の自己相関は，遅れが$(m-l)/\sigma$以下ならば理想的な一様乱数のものとほとんど一致すること，また次元がp/σ以下ならば，多次元分布もほぼ一様分布になることを理論的に示した．

3.3 Lewis & Payne の方法[13]

Tausworthe の方法は，それによって得られる数列のいろいろな性質が理論的にわかっているという意味で大変に優れているが，乱数を1個発生するのに要する時間が，例えば合同法と比べると，はるかに長いという欠点がある，そこで Lewis & Payne は Tausworthe の方法を改良して，短時間に乱数を発生する方法を考案した．まず原始多項式として3項式

$$f(x)=x^p+x^q+1 \quad (3.4)$$

を選ぶ$(p>q)$．したがって，$\{a_i\}$を作り出す漸化式は

$$a_i=a_{i-p}+a_{i-q} \pmod 2 \quad (3.5)$$

となる．そして$\{a_i\}$から$\{u_k\}$を作るのには，

$$u_k=0.a_ka_{k+r}\cdots a_{k+(l-1)r} \quad (3.6)$$

とする．ここに$l\leqq p$で，rは周期mと互いに素な整数である．この方法の特徴は，$\{a_i\}$から取ってきた数を(3.3)のように“横(行方向)”ではなくて，“縦(列方向)”に並べるもので，隣り合う各列間ではrずつ位相をずらしているところにある．

原始3多項式はきわめてたくさんのものが知られているので，p, qの選択の自由度は大きい．文献23)に

表-3 m 系列の周期が素数となる原始3項式 x^p+x^q+1 の例

p	q
31	3, 6, 7, 13
89	38
127	1, 7, 15, 30, 63
521	32, 48, 158, 168
607	105, 147, 273

は，$2 \leqq p \leqq 1,000$ の範囲内のあらゆる原始3項式が列挙されている．このうちで，われわれの目的のために特に好都合なのは，2^p-1 が素数 (Mersenne 素数) となるものである．(ほかのパラメタの値を周期と互いに素になるように選ぶ心配をしなくてもすむから．) $p \geqq 31$, $1 \leqq q \leqq p/2$ で，この条件を満たすものをすべてぬき出して示したのが表-3である．(なお，一般に x^p+x^q+1 が原始多項式なら $x^p+x^{p-q}+1$ も原始多項式で，後者の多項式から生成される最大周期列は，前者から生成されるものを逆順 (添字の減少する順番) に並べたものになることが知られている．)

この方法による乱数の発生が速いのは次の理由による．漸化式 (3.5) の演算は，繰り上りなしの足し算であるから，排他的論理和 (Exclusive OR) と同じであり，したがって $\{u_k\}$ は関係式

$$u_k = u_{k-p} \oplus u_{k-q} \tag{3.7}$$

によって作り出すことができる．ここで，⊕はビットごとの排他的論理和であり，この演算は多くの計算機システムでアセンブリ言語あるいは FORTRAN 語などによってきわめて簡単に記述でき，演算速度も速い．一例として，東大大型計算機センターの HITAC 8700/8800 システムの FORTRAN 語で書いたプログラムを図-1に示す．(JIS の規格には合わないところがあることをお断りしておく．またほかの処理系では，.AND. や .OR. の処理のしかたが違うために，このままでは所望の結果が得られないこともあること

```
SUBROUTINE RND(R)
COMMON/INIT/M(607)
DATA J, N, IP, IQ/0,2147483647,607,147/
J=J+1
IF(J .GT. IP) J=1
K=J+IQ
IF (K .GT.IP) K=K-IP
MCOMPJ=N-M(J)
MCOMPK=N-M(K)
M(J)=MCOMPJ .AND. M(K) .OR. MCOMPK .AND. M(J)
R=M(J)*0.4656613E-9
RETURN
END
```

図-1 Lewis & Payne の方法による乱数発生プログラムの例

に注意する必要がある．)

パラメタの値は p (プログラム中では IP)=607, IQ=$p-q$=147 としてある．またプログラムには直接現われていないが，同システムでは1語=32ビットで，負の整数は2の補数で表現しているので，l=31 とした．ただし，最終的に [0,1) 上の乱数に変換する段階で下位7ビットは失われている．

このプログラムによって発生される乱数の周期は，$m=2^{607}-1 \fallingdotseq 5.3\times10^{182}$ できわめて長く，1秒間に1万個ずつ乱数を使ったとしても，10^{171} 年以上使えることになる．

3.4 初期値の設定方法

1) Lewis & Payne の方法

図-1に示すサブルーチン RND を使用するためには，最初に配列 M に初期値を設定しておく必要があるが，Lewis & Payne の提案に従えば，それは次のようにして行うことになる．

次の操作を l=31 回繰返す．(ただし，最終回だけは3の操作を省略する．)

1. M の各要素の (符号ビットを除く) 最上位 (左端) のビットに1をつめる．
2. RND を r 回呼ぶ．
3. M の各要素を右に1ビットシフトする．

この方法で初期値を設定すると，左端に1を規則的につめた影響が残るので，それを消すために Lewis & Payne は RND をさらに 5,000 p 回ぐらい呼んで“からまわし”してから使うことを提案している．また r の値は 100 p ぐらいに選ぶのが安全であるといっている．そうすると，実際に使う乱数を発生させるまでにサブルーチン RND を呼び出す回数 lr+5,000 p は結局 25 万回 (p=31 の場合)～500 万回 (p=607 の場合) にも達し，相当長い計算時間が消費されることになる．また“からまわし”の回数を一定にしておくと，いつも同じ乱数列を使うことになるので，これを避けるためには，job がかわるごとに“からまわし”の回数を増やしていかなければならないことになり，大変に具合が悪い．

以上に述べたような準備の時間がかかりすぎるという欠点に対処するひとつの方法は，ファイルを利用することである．つまり，job の最後で配列 M の内容およびポインタ J の値をファイルに保存しておけば，次回の job では初期値の設定は不要になる．ファイルを利用できない (あるいは利用したくない) 場合には，次に述べる簡便法を使うのがよいであろう．

2) 初期値をランダムに設定する方法[8]

この方法は，m 系列 $\{a_i\}$ から一様乱数列 $\{u_k\}$ を作り出すのに，(3.6)式によらずに

$$u_k=0.a_{k+r_1}a_{k+r_2}\cdots a_{k+r_l} \qquad (3.8)$$

とすることに相当する．ここに，$r_1, r_2, \cdots, r_l$ は $0\leqq r\leqq m-1$ の範囲からランダムに選んだ整数で，k にはよらない．

初期値をランダムに設定するためには，乱数表を利用する方法，合同法を利用する方法などが考えられるが，後者の方が簡便であるから，これについて述べよう．この場合，合同法による乱数（$[0, P-1]$ 上の整数）を $p=607$ 個発生して配列 M に代入すれば初期値の設定は完了することになり，そのための計算時間はふつう実際に使用する乱数の発生時間に比べて無視できる程度に短かい．前記の HITAC システムを使って，この方法について実験をしてみた．いろいろな統計的検定を行ってみたところ，いずれについても良好な結果が得られた．また，100 万個の乱数を発生するのに要した時間は約 29 秒で，合同法に比べて 2 秒程度長かっただけである．(時間の中には，サブルーチンを呼んだ際の linkage に要した時間約 20 秒が含まれている.)

3) 多次元分布の一様性が理論的に保証される方法[17]

以上の 1), 2) の方法によって初期値を設定した場合に発生される乱数の多次元分布の性質については，経験上は良好であることがわかっている．したがって，実用上多くの場合にはこれで充分であろう．しかし，事実上無限大といえる長い周期にわたって統計的検定を行うことは不可能であるから，これによって周期全体にわたっての一様性が保証されるわけではないことも確かである．そこで，もし周期全体にわたっての多次元分布の一様性が理論的に保証された系列がどうしても欲しい場合には，多少時間はかかるが，次の初期値設定方法を用いるとよい．

K 次元分布の一様性を保証したいものとする．ここに，K は 2 以上 p/l 以下の整数とする．そして，

$$s=p+(l-1)K$$

と定義する．また，図-1 のプログラムの配列 M の第 j 要素の第 i ビット（最上位を $i=1$, 最下位を $i=l$ とする）を b_{ji} で表わすことにする．そうすると，初期値設定の手順は次のとおりである．

1. 要素が 0, 1 からなる p 順列 $(a_1, a_2, \cdots, a_p)$ を任意に（ただし，すべての要素が 0 ではないように）ひとつ定める．

2. 漸化式(3.5)を用いて $a_{p+1}, a_{p+2}, \cdots, a_s$ を求める．

3. $j=1, 2, \cdots, p$ について

$b_{j1}\leftarrow a_j$

$b_{ji}\leftarrow a_j\oplus a_{j+(i-1)K}$ $(i=2, 3, \cdots, l)$

とする．

このアルゴリズムで必要とする計算時間の主要部分は，約 pl 回の排他的論理和を取る時間（および，JIS FORTRAN で書くとすれば，同じ回数の乗算と加算に要する時間の和）であるから，Lewis & Payne の方法の所要時間に比べれば，はるかに短い時間である．しかも，$(a_1, a_2, \cdots, a_p)$ が任意に選べるので，Lewis & Payne の方法のように常に同じ系列が作り出されるという欠点は，この方法にはない．

4. 任意の離散分布に従う乱数の発生法

4.1 逆関数法など

例えば幾何分布とかポアソン分布などのような特定の離散分布に従う乱数を発生させるためには，その特定の分布の持つ性質（ほかの分布との関連性など）を利用して効率的な発生方法を考案することが可能であり，実際いろいろな方法が考えられている．しかし，その場合に用いられている技巧は，ほかの形の分布に従う乱数を発生させる際には，一般に役に立たないことはいうまでもない．

任意の形の離散分布（ただし取りうる値の数は有限個とする）に従う乱数を発生させるために一般的に使える方法の 1 つは，(連続分布に対しても使える) 次の逆関数法である．

値 $x_1, x_2, \cdots, x_k$ を確率 $p_1, p_2, \cdots, p_k$ で取る乱数 x を発生させるためには次のようにすればよい．

1. 区間 $[0, 1)$ 上の一様乱数 u を発生する．

2. $$x\leftarrow\begin{cases} x_1 & (0\leq u<p_1 \text{ のとき}) \\ x_2 & (p_1\leq u<p_1+p_2 \text{ のとき}) \\ \cdots\cdots\cdots\cdots & \\ x_k & (p_1+p_2+\cdots+p_{k-1}\leq u<1 \text{ のとき}) \end{cases}$$

この方法に従って“素朴に”プログラムを作ると，x を 1 個発生させるのに最大 $k-1$ 回，平均 $[p_1+2p_2+3p_3+\cdots+(k-1)p_{k-1}+(k-1)p_k]$ 回の比較演算が必要になる．したがって，(必要なら x_i の順番を入れ換えることによって) $p_1\geq p_2\geq\cdots\geq p_k$ となるようにしておくのがよい．しかし k が大きいときには，この方法を直接使うよりも，よく知られている探索手法を

利用して，比較演算の平均回数を減らすようにするのがよい．k が大きいときにさらに有効な方法として，Marsaglia の考案した巧妙な方法があるが，これについては Knuth の本の 120 ページ (問題 20 および 21) に説明があるので，説明を割愛する．ただし，この方法には相当たくさんのメモリを必要とするという難点がある．

4.2 別 名 法

任意の離散分布 (ただし取りうる値の数は有限個とする．) に従う乱数を発生するためのきわめて効果的な方法が 1974 年に Walker[20] によって提案されているが，これはいままであまり注目されなかったようである．別名法 (alias method) と呼ばれるこの方法の基礎になっているのは，離散分布の分解に関する次の定理である．

定理 $p(x)$ $(x=1,2,\cdots,n)$ を任意の離散分布の確率関数とする．そうすると，$p(\cdot)$ は n 個の 2 点分布 $q_i(\cdot)$, $i=1,2,\cdots,n$, の等加重和として表わせる：

$$p(\cdot)=\frac{1}{n}\sum_{i=1}^{n} q_i(\cdot)$$

ここで，分布 $q_i(\cdot)$ に従う確率変数の取る値の 1 つは i であるように選ぶことができる． (定理終り)

この定理の証明は，n に関する数学的帰納法によって容易にできるので，省略する．また，分解は一意でないことに注意しておく．

さて，このような分解ができたとして，$i=1,2,\cdots,n$ に対して

$v_i \equiv q_i(i)$

$a_i \equiv$ 分布 $q_i(\cdot)$ に従う確率変数が取る i 以外の値

と定義すると，分布 $p(\cdot)$ に従う乱数の発生手順は次のようになる．

1. $[0,n)$ 上の一様乱数 v を発生する．
2. $i \leftarrow [v]+1$, $u \leftarrow i-v$ とする．
3. もし $u \le v_i$ ならば $x \leftarrow i$,
   そうでなければ $x \leftarrow a_i$ とする．

ここで，[ ] はガウス記号であり，したがって i は $1,2,\cdots,n$ のいずれかの値を等確率で取ることになり，u は i とは独立な $[0,1)$ 上の一様乱数となる．逆関数法あるいは Marsaglia の方法と比べてこの方法が大変に優れている点は，nの大きさと分布 $p(\cdot)$の形状のいかんによらずに，1 個の一様乱数の発生と 1 回の比較演算によって所望の乱数が得られるところにある．

この手順に従って乱数を発生させるためには，最初に v_i, a_i $(i=1,2,\cdots,n)$ を求めておく必要があるが，それらは次の手順によって求められる．

別名法のためのテーブルを作成するアルゴリズム

1. $v_i \leftarrow n \cdot p(i)$ $(i=1,2,\cdots,n)$
2. $v_i \ge 1$ なる i の集合を G, $v_i<1$ なる i の集合を S とする．
3. S が空でないかぎり，3.1 から 3.5 までを繰返し実行する．

3.1 G の要素を 1 つ任意に選ぶ；それが k であるとする．S の要素を 1 つ任意に選ぶ；それが j であるとする．

3.2 $a_j \leftarrow k$

3.3 $v_k \leftarrow v_k-(1-v_j)$

3.4 $v_k<1$ なら，G の要素 k を取り除いて S に移す．

3.5 S の要素 j を取り除く．

手順 2 で定められる集合 S の要素の数はたかだか $n-1$ であり，手順 3 を 1 度実行すると，S の要素の数は 1 だけ減少するので，手順 3 をたかだか $n-1$ 回実行するとこのアルゴリズムは終了する．また a_i のなかには値が定まらないものがあるが，これはさしつかえない．またこのアルゴリズムで必要とする記憶容量と演算の回数はいずれも $O(n)$ であることは明らかであろう．

5. 任意の連続分布に従う乱数を発生する方法

5.1 棄 却 法

任意の連続分布に従う乱数を発生するための一般的な方法のひとつに下記の棄却法 (rejection method) があり，種々の乱数の発生法 (の一部) に有効に使われている．

密度関数が $f(\cdot)$ で与えられる分布に従う乱数 x を発生させたいものとする．まず初めに，$g(\cdot) \ge f(\cdot)$ を満たす関数 $g(\cdot)$ を適当に選ぶ．そして，定数 c を

$$\int_{-\infty}^{+\infty} c \cdot g(x)dx=1$$

となるように定める．そうすると，乱数 x の発生手順は次のとおりである．

1. 密度関数 $c \cdot g(\cdot)$ を持つ乱数 ξ を発生する．
2. ξ と独立に $[0,1)$ 上の一様乱数 u を発生する．
3. $u \le f(\xi)/g(\xi)$ なら $x \leftarrow \xi$ とし，そうでなければ 1 へもどる．

この方法を効率的なものとするためには，(1) 手順 1 で ξ が簡単に発生できるように，しかも (2) c の

値，すなわち手順1～3を1回実行するだけでxが得られる確率，がなるべく1に近くなるように，$g(\cdot)$を選ぶ必要があるが，これらの目標はふつう相反するものである．通常は，両方の目標の妥協点をとって，$c \cdot g(\cdot)$が一様分布，三角分布，台形分布などになるように選んでいる．

5.2 棄却—混合法

棄却—混合法 (rejection-mixture method) は，棄却法を変形して，繰返しを不要にしたものである．まず，密度関数 f を非負の2つの関数 g_1, g_2 に分解する：

$$f(\cdot)=g_1(\cdot)+g_2(\cdot).$$

つぎに，密度関数 h を

$$h(\cdot) \geqq g_1(\cdot)$$

を満たすように選ぶ．また定数 c を，$c \cdot g_2(\cdot)$ が密度関数となるように定める．そうすると，$f(\cdot)$ に従う乱数 x の発生手順は次のとおりである．

1. $h(\cdot)$ に従う乱数を発生する．
2. ξ と独立に $[0,1)$ 上の一様乱数 u を発生する．
3. $u \leqq g_1(\xi)/h(\xi)$ ならば $x \leftarrow \xi$ とし，そうでなければ $c \cdot g_2(\cdot)$ に従う乱数を発生して，それを x とする．

この方法が効果的なものとなるためには，$h(\cdot)$ および $c \cdot g_2(\cdot)$ に従う乱数がともに容易に発生できるようにこれらの密度関数を選ぶ必要があるが，$h(\cdot)$ は必ずしも $g_1(\cdot)$ に近く選ぶ必要はないことに注意しよう．また，手順3で $g_1(\xi)/h(\xi)$ の計算に比較的長い時間がかかる場合には，この関数の下界で，これになるべく近くて計算時間の短かいものを求めておいて，まずこの下界と u を比較するとよい．

5.3 別名—棄却—混合法[10),11)]

別名法と棄却—混合法を組み合わせて得られるのが別名—棄却—混合法 (alias-rejection-mixture method, ARM 法) で，任意の連続分布に従う乱数の発生に利用できる．この方法を適用するためには，まず密度関数 $f(\cdot)$ をいくつかの密度関数の加重和に分解する．(この技巧は，ふつう合成法 (composition method) と呼ばれているものである．)

$$f(\cdot)=\sum_{i=1}^{n} p_i f_i(\cdot).$$

ここで p_i はすべて正で，その和は1である．そこで，値 i を確率 p_i $(i=1,2,\cdots,n)$ で取る乱数を別名法によって発生し，$f_i(\cdot)$ に従う乱数を棄却—混合法によって発生しようというわけである．

この方法によれば，自由に選べるパラメタおよび関数がきわめて多いので，これらをうまく選べば，大変に効率的な乱数発生法が得られる可能性がある．しかし，逆に自由度が多すぎて当惑することもあるであろう．そこで Kronmal 等は，これを次のように特殊化した方法を提唱し，これを一様 ARM 法と呼んでいる．

1) 重み p_i はすべて $1/n$ とする．(こうすると，i の選択には一様乱数があれば十分で，別名法は不要になるが，別名法の考え方は，ほかのところに使われている．)

2) $f_i(\cdot)=g_{i,1}(\cdot)+g_{i,2}(\cdot)$ と分解するが，$g_{i,1}(\cdot)$ $(i=1,2,\cdots,n)$ は，それらの台 (support) が相続く等間隔の区間になるように選ぶ．(ただし，両端 $i=1,n$ については，必ずしもそうはならなくてもよい．) 各 i について，$g_{i,2}(\cdot)$ は $g_{i,1}(\cdot)$ とは別の区間上の一様分布となるように選ぶ．(ここに別名法の考え方が使われている．)

3. $h_i(\cdot) \geqq g_{i,1}(\cdot)$ を満足する関数は，両端 $(i=1,n)$ のものを除いて，$g_{i,1}(\cdot)$ と同一の区間上の一様分布となるように選ぶ．

このような分解ができたとすると，一様 ARM 法による乱数発生のプログラムはきわめて簡単である．しかも n を十分に大きくとると，所望の乱数1個を発生するのに要する演算の型や回数，必要な一様乱数の個数等が密度関数 $f(\cdot)$ の形にほとんど無関係であるという特色がある．したがってこの方法は，従来あまり効率的な発生方法が知られていなかった分布に従う乱数の発生アルゴリズムの設計に有効であろうと思われる．Kromal 等は，この方法を ($n=32$ として) 正規分布に適用した結果，十分に速い乱数発生アルゴリズムが得られたと報告している．

6. おわりに

初めの予定では，正規分布，指数分布，ポアソン分布等の個々の分布に従う乱数の発生法についてもふれるつもりであったが，割愛せざるをえなくなってしまった．これらの乱数については，きわめてたくさんの発生アルゴリズムが発表されていて，どれをとりあげて解説すべきか，選択が大へんむずかしい．発生時間が短いというのがひとつの選択基準であろうが，これはプログラムの作り方や使う計算機によっても大きく変わるので絶対的な評価はできない．また，発生時間だけでなく，記憶容量，プログラム作成の容易さなど

も重要な評価基準であろうから，多数のアルゴリズムの中からいくつかを選んで推奨することはきわめて困難である．そこでここでは，総合報告的な文献を二，三あげるにとどめることとしたい．文献 1)～4) がそれである．ただし，これらも発表されているアルゴリズムのすべてを網羅しているわけではないことをおことわりしておく．

参 考 文 献

1) Ahrens, J. H. and Dieter, U.: Computer methods for sampling from gamma, beta, Poisson and binomial distributions, Computing, Vol. 12, pp. 223-246 (1974).
2) Atkinson, A. C.: The computer generation of Poisson random variables, Applied Statistics, Vol. 28, pp. 29-35 (1979).
3) Atkinson, A. C.: Recent developments in the computer generation of Poisson random variables, *ibid.*, pp. 260-263 (1979).
4) Atkinson, A. C. and Pearce, M. C.: The computer generation of beta, gamma and normal random variables, J. Royal Statistical Society, Series A, Vol. 139, Part 4, pp. 431-461 (1976).
5) Dieter, U.: How to calculate shortest vectors in a lattice, Mathematics of Computation, Vol. 29, pp. 827-833 (1975).
6) Dieter, U.: Schwierigkeiten bei der Erzeugung gleichverteilter Zufallszahlen, Proceedings in Operations Research VIII, pp. 1-24 (1978).
7) 伏見正則：一様乱数の発生法について，数学セミナー，Vol. 18, No. 10, pp. 23-27 (1979).
8) 伏見正則，手塚 集：最大周期列を利用した擬似乱数の発生法について，東京大学工学部紀要 (A), No. 17, pp. 42-43 (1979).
9) Knuth, D. E.: The Art of Computer Programming, Vol. 2 (Seminumerical Algorithms), Addison-Wesley, Reading, Mass. (1969).
10) Kronmal, R. A., Peterson, Jr., A. V. and Lundberg, E. D.: The alias-rejection-mixture method for generating random variables from continuous distributions, The 1978 Statistical Computing Section Proceedings of the American Statistical Association, pp. 106-110 (1978).
11) Kronmal, R. A. and Peterson, Jr., A. V.: The alias-rejection-mixture method for computer generation of random variables from continuous distributions, Technical Report No. 23, Dept. of Biostatistics, University of Washington (1978).
12) Kronmal, R. A. and Peterson, Jr., A. V.: On the alias method for generating random variables from a discrete distribution, The American Statistician, Vol. 33, pp. 214-218 (1979).
13) Lewis, T. G. and Payne, W. H.: Generalized feedback shift register pseudorandom number algorithm, J. ACM, Vol. 20, pp. 456-468 (1973).
14) Marsaglia, G.: Random numbers fall mainly in the planes, Proceedings of the National Academy of Sciences (USA), Vol. 61, pp. 25-28 (1968).
15) Marsaglia, G.: The structure of linear congruential sequences, Applications of Number Theory to Numerical Analysis, edited by S. K. Zaremba, Academic Press, New York, pp. 249-285 (1972).
16) Tausworthe, R. C.: Random numbers generated by linear recurrence modulo two, Mathematics of Computation, Vol. 19, pp. 201-209 (1965).
17) 手塚 集，伏見正則：M 系列を用いた擬似乱数発生法，情報処理学会第 21 回全国大会予稿集 (1980).
18) 津田孝夫：計算機による多変数問題の数値解析，サイエンス社 (1973).
19) 津田孝夫：モンテカルロ法とシミュレーション，培風館 (1977).
20) Walker, A. J.: Fast generation of uniformly distributed pseudorandom numbers with floating point representation, Electronics Letters, Vol. 10, pp. 553-554 (1974).
21) Walker, A. J.: New fast method for generating discrete random numbers with arbitrary frequency distributions, *ibid.*, pp. 127-128 (1974).
22) Walker, A. J.: An efficient method for generating discrete random variables with general distributions, ACM Transactions on Mathematical Software, Vol. 3, pp. 253-256 (1977).
23) Zierler, N. and Brillhart, J.: On primitive trinomials (mod 2), Information and Control, Vol. 13, pp. 541-554 (1968) and Vol. 14, pp. 566-569 (1969).

(昭和 55 年 5 月 27 日受付)