ベクトル化乱数発生プログラム

大阪大学　基礎工学部　数理教室　　宮村　修
阿南工業高等専門学校　電気工学科　牧野　純

1．はじめに

スーパーコンピュータの出現により，従来では考えられなかったような大規模のモンテカルロシミュレーションが可能となった．それに伴って一回の計算で使用する乱数の個数も膨大な量になることが多くなった．乱数を発生する方法として従来からよく使われているLehmerの合同法はベクトル化した場合にも使いやすい．[1]しかし，たとえばSXのような32ビットの計算機の場合，混合合同法で発生される乱数列の周期は高々2^{32}であり，SX-2Nはこれらをわずか16秒程で発生し尽くすのである．そこで，混合合同法よりもはるかに多量に，しかもなるべく高速に，乱数を発生させる方法が求められている．このような要求に応えてくれる最も簡便な乱数の発生法がM系列（最大周期列）による方法である．

M系列を用いて疑似乱数列を作り出す方法は最初Tausworthe[2]によって研究された．今日普通に用いられているアルゴリズムは，TausworthのものをLewisとPayne[3]が改良したもので，初期値$a_1,a_2,a_3,\ldots,a_p$を与えておいてp次の漸化式

$$a_n=\mathrm{IEOR}(a_{n-p},a_{n-q})$$

によって乱数列$\{a_n\}$を発生するものである．ここで，IEORはビットごとの排他的論理和を表し，多くのFORTRAN処理系で組込み関数として利用できる．またp,qは特性多項式x^p+x^q+1がGF(2)上の原始多項式になるように選ぶ．この数列の周期は2^p-1で，pを適当に大きく選べば，いかなる計算においても十分な量の乱数を発生することができる．

しかし，このアルゴリズムはこのままではあまりベクトル化に適した形をしていない．すなわち論理的な因果関係から考えてベクトル長が最大でqまでしかとれないのである．そこで本稿ではM系列乱数をベクトル的に発生するいくつかのアルゴリズムを提案する．これらの方法はすべて任意のベクトル長に対して適用できるものであるが，話を具体的にするために，以下の節ではベクトル長を256とする場合を例にとって解説する．

2．高次の漸化式を用いる方法

以下，例としてp=250,q=103の場合を考える．この場合，連続する250項を保持することによって，漸化式

$$a_n=\mathrm{IEOR}(a_{n-250},a_{n-103}) \qquad (1)$$

により次項が計算される．このアルゴリズムはベクトル化してもベクトル長が103までしかとれない．ベクトル長を147まで延ばすことは式(1)の代わりにq=250-103=147として

$$b_n=IEOR(b_{n-250},b_{n-147}) \qquad (2)$$

という式を使えば可能である．この数列$\{b_n\}$は数列$\{a_n\}$を逆の順序に並べたものであり，乱数源としては等価なものと考えられる．

ベクトル長をもっと大きくとるための方法としては，高次のM系列を用いることが考えられる．たとえば，ベクトル長を256にとるためにはp=521,q=32のM系列を用いればよい．しかしベクトル長は乱数を引用するプログラムによって決まるべきものである．問題に応じて異なるM系列を使うのは，乱数の性質の検定のことを考慮すれば避けた方が良いであろう．

そこで我々は，(1)で定義されるM系列乱数を任意のベクトル長で並列的に発生することを考えた．その第１の方法は，排他的論理和の性質によって漸化式(1)から導かれる

$$a_n=IEOR(a_{n-2^k\cdot 250},a_{n-2^k\cdot 103}) \quad (k=1,2,3,\ldots) \qquad (3)$$

という関係式を用いる方法である．この式を使えばベクトル長を$2^k\cdot 103$までとることができる．たとえばベクトル長を256にとるためにはk=2の場合の

$$a_n=IEOR(a_{n-1000},a_{n-412}) \qquad (4)$$

という漸化式を使えばよい．ただし漸化式(4)の初期値として用いる$a_1,\ldots,a_{1000}$のうち，M系列乱数の初期値として与えることができるのは$a_1,\ldots,a_{250}$の250項であり，残りの$a_{251},\ldots,a_{1000}$は式(1)を用いて計算しておかなければならない．

```
      PARAMETER (FNORM=1/2.0**31)
      DIMENSION IRAND(1000)
      DATA J0/0/
C
      IX=1774315169
      DO 10 I=1,250
        IX=IAND(IX*48828125,2147483647)
        IRAND(I)=IX
   10 CONTINUE
      DO 20 I=251,1000
        IRAND(I)=IEOR(IRAND(I-250),IRAND(I-103))
   20 CONTINUE
C
      DO 200 LV=1,10000
*VDIR NODEP
        DO 100 LH=1,256
          J=J0+LH
          K=J-412
          IF(J.GT.1000) J=J-1000
          IF(K.LE.0) K=K+1000
          IRAND(J)=IEOR(IRAND(J),IRAND(K))
          RAND=FNORM*IRAND(J)
  100   CONTINUE
        J0=J0+256
        IF(J0.GT.1000) J0=J0-1000
  200 CONTINUE
      END
```

プログラム１．

この方法で実際に乱数を発生させたのがプログラム１である．漸化式(4)を用いるためにはつねに1000項の表を保持しなければならないが，そのために寸法が1000の1次元配列IRANDを使っている．M系列乱数の初期値はここでは Kirkpatrick-Stoll[4] に従って合同法によって定めたが，Lewis-Payne[3]の方法を用いてもよい．

３．M系列を分割して用いる方法

ベクトルの成分を横に並べて書くことにすると，前節の方法はM系列乱数を横方向に並べる方法だということができる（図１（ａ））．それに対してM系列乱数を縦方向に並べる方法も考えられる（図１（ｂ））．すなわち次々に発生される乱数ベクトルのある決まった番号の成分に着目したとき，それがM系列乱数になっているのである．

$$
\begin{array}{llll}
a_1 & a_2 & a_3 & \cdots\cdots\ a_{256} \\
a_{257} & a_{258} & a_{259} & \cdots\cdots\ a_{512} \\
a_{513} & \cdots\cdots & \cdots\cdots & \cdots\cdots \\
\cdots\cdots & \cdots\cdots & \cdots\cdots & \cdots\cdots \\
\cdots\cdots & \cdots\cdots & \cdots\cdots & \cdots\cdots \\
\cdots\cdots & \cdots\cdots & \cdots\cdots & \cdots\cdots
\end{array}
\qquad
\begin{array}{llll}
a_i & a_j & a_k & \cdots\cdots\ a_l \\
a_{i+1} & a_{j+1} & a_{k+1} & \cdots\cdots\ a_{l+1} \\
a_{i+2} & a_{j+2} & a_{k+2} & \cdots\cdots\ a_{l+2} \\
\cdot & \cdot & \cdot & \cdot \\
\cdot & \cdot & \cdot & \cdot \\
\cdot & \cdot & \cdot & \cdot
\end{array}
$$

（ａ）横方向　　（ｂ）縦方向

図１．　M系列乱数の並べ方

この方法では，各ベクトル成分に対してp個の乱数列を保持する必要があるが，あるベクトル成分の乱数列は他の成分とは独立に発生させることができる．そのため，ベクトル長はp,qの値とまったく無関係に自由に選ぶことができるし，また，ベクトル化に必要な定義と引用の因果関係が自動的にみたされているので，実際のプログラム中での乱数の引用法に自由度を与えることもできる．

この縦方向のM系列乱数の発生については２通りの引用法が考えられる．それをプログラム２とプログラム３に示した．プログラム２では乱数発生はつねにすべての成分について同時に行なわれると仮定している．これを一括生成型のアルゴリズムと呼ぶことにしよう．このアルゴリズムでは保持している乱数の表の管理を最深ループの外で行なうので乱数は高速に発生できる．それに対して，プログラム３では表の管理を最深ループの中で行なっている．そのためこのアルゴリズムは乱数発生がベクトル化されるループ中のIFブロックに含まれているような場合にも使える．こちらは随時引用型と呼ぶことができよう．

```
      PARAMETER(FNORM=1/2.0**31)
      DIMENSION IRAND(257,250)
      DATA J/0/K/147/
C
      READ(51) IRAND
C
      DO 200 LV=1,10000
        J=J+1
        K=K+1
        IF(J.GT.250) J=1
        IF(K.GT.250) K=1
        DO 100 LH=1,256
          IRAND(LH,J)=IEOR(IRAND(LH,J),IRAND(LH,K))
          RAND=FNORM*IRAND(LH,J)
  100   CONTINUE
  200 CONTINUE
      END
```

プログラム２．

```
      PARAMETER(FNORM=1/2.0**31)
      DIMENSION IRAND(257,250),JD(257)
      DATA JD/257*0/
C
      READ(51) IRAND
C
      DO 200 LV=1,10000
*VDIR NODEP
        DO 100 LH=1,256
          J=JD(LH)+1
          IF(J.GT.250) J=1
          K=J-103
          IF(K.LE.0) K=K+250
          IRAND(LH,J)=IEOR(IRAND(LH,J),IRAND(LH,K))
          RAND=FNORM*IRAND(LH,J)
          JD(LH)=J
  100   CONTINUE
  200 CONTINUE
      END
```

プログラム３．

一括生成にせよ随時引用にせよ，各成分にはあらかじめp個ずつの初期値を用意しておかなければならない．これをどう決めるかということも重要な問題である． 長崎大学の津田-森-中村[5]は式(1)のM系列乱数に対し206項を並列に発生するアルゴリズムを提唱し，さらに8組の初期値を与えて乱数発生を高速化することを試みた．しかし，勝手に初期値を与えたのでは成分間の独立性が保証されない．そこで我々が用いたのは，M系列乱数の１周期を何個かの部分列に分割しておいて，各部分列をベクトル成分に割り当てる方法である．プログラム４は１周期を256等分したときの各系列の初期値を与えるプログラムである．すなわちM系列の第$K\cdot 2^{242}$項から始まる250項を K=1,2,...,256について配列 IRANDに与える．M系列乱数の初期値 IRAND0(0:249) にはここでも Kirkpatrick-Stoll流に合同法によって発生した250個の整数を用いた．この場合の周期は$2^{250}-1$であるからM系列乱数の第2^{256}項IRAND(256,0)の値は第2項IRAND0(1)の値に一致する．ここでは求めたIRANDの値はファイルに保存してプログラム２やプログラム３のように計算の実行時に読み込むようにしたが，プログラム４はサブルーチン化して使ってもよい．プログラム４はほぼ完全にベクトル化され，SX-2Nでの実行時間は0.303秒であった．

```
      DIMENSION IRAND0(0:499),IRAND(257,0:249),L(257,0:499)
      DATA IRAND/64250*0/L/128500*0/
C FIRST 250 SEEDS
      IX=1774315169
      DO 10 I=0,249
        IX=IAND(IX*48828125,2147483647)
        IRAND0(I)=IX
   10 CONTINUE
C NEXT 250 TERMS
      DO 20 I=250,499
        IRAND0(I)=IEOR(IRAND0(I-250),IRAND0(I-103))
   20 CONTINUE
C DELAY OPERATORS IN 242TH ORDER BASIS
      DO 30 K=1,249
            L(K,K)=1
   30 CONTINUE
      DO 40 K=250,256
            L(K,K-250)=1
            L(K,K-103)=1
   40 CONTINUE
C REDUCTION TO 0TH ORDER BASIS
      DO 500 J=242,1,-1
        DO 200 I=249,0,-1
          DO 100 K=1,256
            L(K,2*I+1)=0
            L(K,2*I)=L(K,I)
  100     CONTINUE
  200   CONTINUE
        DO 400 I=498,250,-1
          DO 300 K=1,256
            L(K,I-250)=IEOR(L(K,I-250),L(K,I))
            L(K,I-103)=IEOR(L(K,I-103),L(K,I))
  300     CONTINUE
  400   CONTINUE
  500 CONTINUE
C FIRST 250 TERMS FOR KTH SEGMENT
      DO 800 J=0,249
        DO 700 I=0,249
          DO 600 K=1,256
            IRAND(K,J)=IEOR(IRAND(K,J),L(K,I)*IRAND0(I+J))
  600     CONTINUE
  700   CONTINUE
  800 CONTINUE
      WRITE(61)IRAND
      WRITE(6,*)IRAND0(1),IRAND(256,0)
      END
```

プログラム４．

４．ＳＸにおける発生効率

これまでに述べた各アルゴリズムを使って実際にSX-2Nで乱数を発生してみた．プログラム１～３によって2,560,000個の乱数を発生しておいて，１個の乱数を生成するのに必要な時間を割り出した結果が表１である．時間は文番号200のDOループの前後にCLOCKサブルーチンをはさんで測定した．この表には我々がSX-2Nで試みた種ベクトル法[1]による混合合同法（周期2^{32}）の結果も載せてある．また，津田 他[5]の値は計算機の機種も異なり単純な比較はできないが参考のため併記した．M系列乱数の発生法としてはプログラム２のアルゴリズムが圧倒的に高速であることがわかる．

	アルゴリズム	発生時間(ns)	備考
プログラム１	高次漸化式	81.38	
プログラム２	一括生成	4.09	
プログラム３	随時引用	86.80	
混合合同法	種ベクトル法	3.78	周期2^{32}
津田　他		21.42	HITAC S-810/20

表１．　疑似乱数の発生時間

5．おわりに

本稿ではM系列乱数のベクトル発生について種々のアルゴリズムを考察し，SX-2Nにおけるそれらの効率を試した．第２節で述べた高次の漸化式を用いる方法は保持すべき表のサイズが比較的小さくてすむという特長を持っていた．第３節で述べたM系列乱数の１周期を分割して各ベクトル成分に割り当てる方法は，各成分の乱数を同時に発生させれば，合同法に匹敵する高速度が得られた．また，乱数発生をベクトル化されたループ内のIFブロック中で行なうこともできた．このように各アルゴリズムにはそれぞれに長所があるが，一般的にはプログラム２のアルゴリズムが使いやすいであろう．

得られた乱数列の検定は乱数の使用者が各自の目的と使用法に従って行なうべきである．とくに，ベクトル化された乱数発生の場合，図１のように乱数は２次元に並ぶ．これを使用者が自分のプログラムに組み込んで使う場合，縦横両方向について乱数の性質が問われることになるであろう．我々が取り上げたアルゴリズムでは，ほとんどの場合，縦方向の並びも横方向の並びも同じ特性多項式から導かれるM系列乱数になる．しかしM系列乱数の乱数としての性格は初期値の取り方に大きく依存するといわれている．十分な検定が必要である．

第３節で議論した初期値の設定法は，成分間の独立性を保証するものであり，またベクトル化して高速に大量の初期値を準備できるもので，非常に興味深い．M系列が初期値付近で好ましくない規則性を持てば円分位相点においても類似の規則性を持つことが指摘されている．[6]では初期値を規則性が無いように注意深く選んだ場合には円分位相点における性質はどうであろうか．我々は第３節の方法で発生したベクトル乱数のいくつかの成分について簡単な検定を行なったが，いずれの結果もACOSの組込関数RANDと同程度の良好な結果を得た．より詳しい検定の結果は別の機会に発表したい．[7]

文献

1) T.Matsuura,K.Miura and M.Makino, Comput.Phys.Commun.37(1985)101

2) R.C.Tausworthe, Math.Comput.19(1965)201

3) T.G.Lewis and W.H.Payne, J.Assoc.Comput.Mach.12(1965)83

4) S.Kirkpatrick and E.P.Stoll, J.Comput.Phys.40(1981)517

5) 津田義典，森正寿，中村彰，情報処理学会第31回全国大会講演論文集(1985)77

6) 柏木潤，原田博之，計測自動制御学会論文集18(1982)999

7) 牧野，宮村，発表予定