三重大学教育学部研究紀要　第52巻　自然科学（2001）　7－12頁

実数シフト乱数生成法の長周期化について

谷　口　礼　偉

On Long-Periodization of the Shift-Real Random Number Generator

Hirotake YAGUCHI

ABSTRACT

We consider a way of long-periodization of the Shift-Real random number generator which is newly introduced by the author. We give and test the Shift-Real random number generator which has the period of approximately the fifteenth power of 10.

新しく導入された実数シフト乱数生成法における乱数周期の長周期化について考察する。実用プログラムとして、周期約10の15乗をもつ実数シフト法乱数生成プログラムを提供し、その乱数性を検証する。

実数シフト乱数生成法（以降、(SR)法と略記）は、コンピュータによる以下のような擬似乱数生成法であった［2］。

区間 $[c, d]$ $(0<c<d)$ を $n-1$ 等分し

$$x_i = c + \frac{d-c}{n-1} \times i, \qquad i = 0, 1, \cdots, n-1,$$

とおく。$x = x_i$ として、

$$f(x) = x \cdot \frac{x}{2} \cdot \frac{x}{3} \cdot \frac{x}{4} \cdots \frac{x}{23} \cdot \frac{x}{24}$$

を

$$w_0 = 1, \qquad w_k = w_{k-1} \times \frac{x}{k}, \qquad k = 1, 2, \cdots, 24,$$

により倍精度計算する。ただし、w_k を1回計算するごとに、w_k を表す倍精度変数に対して、

i）　仮数部の全てのビット値を1ビット左にシフトし、
ii）　さらに、仮数部上位23ビット以外のビットは0にする；
iii）　また、指数部は …$\times 2^0$ となるように設定する。

原稿受理日 2000年9月29日
三重大学教育学部数学教室
Department of Mathematics, Faculty of Education, Mie University
Kamihama 1515, 514-8507 Tsu, Mie, Japan

このようにして得られた$f(x_i)$の計算結果f_iを浮動小数点表示し、上位3桁の数字を捨て、残った桁から続けて4桁の数をとり、これを乱数値とする。iを0から$n-1$まで変化させれば、n個の10進4桁数からなる1つの乱数列が得られる。

以上が(SR)法であるが、[2]では(SR)法の乱数特性の改良のためさらに(SR/1)法、(SR/2)法が示されている。いずれにしても連続発生された乱数の数は、区間$[c, d]=[22, 26]$に対する分割数$n=80000, 80001, \ldots, 89999$より生成される乱数全てを加えても、高々$80000\times10000=8\times10^8$個程度であった

(SR)法の長周期化

本報告は、

命題 p, q, r, sが素数であるとき、$(rk \bmod p, sk \bmod q)$, $k=0, 1, 2, \ldots,$の周期はpqである。

を利用して、周期が$pq\approx10^{15}$程度の(SR/l)法乱数を発生する方法について述べる。(註. $rk \bmod p$は、整数rkを整数pで割ったときの余りの整数を表す。)この命題の証明は容易であるので読者に委ねる。以下、$[c, d]=[16, 32]$とし、さらに$p=49933453$、$q=22801201$、$r=491377$、$s=47513$とする。pqの正確な値は

$$p\times q=1,138,542,698,477,053\approx1.1\times10^{15}$$

である。また、$[c, d]=[2^4, 2^5]$とした理由は、浮動小数点演算においてはx_iと$x_i\cdot2^e$は同じ仮数部を生成することによる。

まず、SRl (n, i), $0\leqq i<n$, で、$x_i=16+\dfrac{16}{n}\cdot i$において(SR/$l$)法により得られる4桁整数を表す。次に、$a, b$を正の整数とし、関数

$$(r_k, s_k)\equiv(rk \bmod p, sk \bmod q)\mapsto z(r_k, s_k)$$

を

$$z_k\equiv z(r_k, s_k)=\begin{cases}\mathrm{SR}l\,(a+s_k, r_k) & (a+s_k)>r_k \text{ のとき、}\\ \mathrm{SR}l\,(b-s_k, r_k-(a+s_k)) & (a+s_k)\leqq r_k \text{ のとき、}\end{cases}$$

で定める。a, bとして、(i) $a+2q<b$および(ii) $a+b>p$を満たすものをとれば、(i)より$a+s_k<b-s_k$が常に成り立ち、また、(ii)より$b-s_k>r_k-(a+s_k)$が保証されるので、平面上の点$(a+s_k, r_k)$あるいは$(b-s_k, r_k-(a+s_k))$, $k=0, 1, \ldots, pq-1$, はすべて異なり、4桁整数列$z_0, z_1, z_2, \ldots,$の周期は(r_k, s_k)と同じpqとなる。一方、[2]の§3で行われた議論から、一様乱数列SRl (n, i), $i=0, 1, \ldots, n-1$, はnが異なるとき、互いに独立と考えられる。よって、数列$z_0, z_1, \ldots, z_{pq-1}$は周期$pq$の一様乱数列になることが期待される。

乱数の周期pqを大きくするためには、p(したがってb)を大きくとればよいが、あまり大きなbを選ぶと、$x_i=16+\dfrac{16}{b-s_k}\cdot i$として、$f(x_i)$と$f(x_{i+1})$の違いが現れなくなるので、極端に

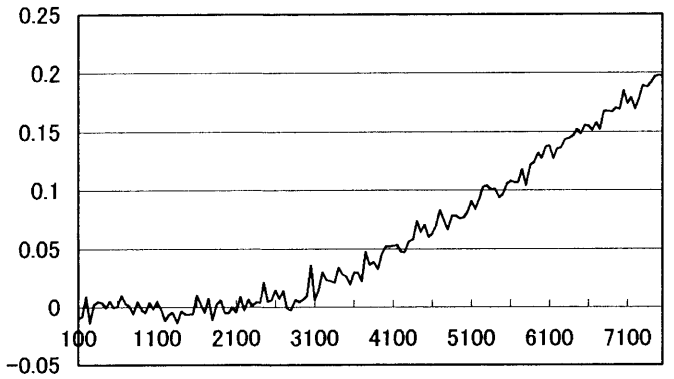
大きな値をとるわけにはいかない。次の図は、x 軸の値 N に対して

$$x_{step} = \frac{1}{N \times 1000}$$

としたときの、乱数列

$$\mathrm{SR}(x_i), \qquad x_i = 22 + x_{step} \times i, \qquad i = 0,\ 1,\ \cdots,\ 19999,$$

に対する遅れ1の系列相関係数（[1]、[2] 参照）を計算した結果である。



$N = 3000$ あたりから正の相関が目立つことから、x_{step} の値は $1/(3\times10^6)$ 以下が望ましいことが分かる。区間 [16, 32] の分割数に換算すると、$(3\times10^6)\times16 = 4.8\times10^7$ 分割となる。以下 $b = 48060000$、$a = 1920000$ とおいて、この長周期化の手法を [2] の (SR/2)法に適用し、その有効性を検証してみよう。

(SR/2)法に対する長周期化の適用

(SR/2)法は、あらかじめ定数 α $(0 \leqq \alpha \leqq 0.5)$ を定めておき、

実数シフト法による $f(x_i)$ の計算結果 f_i が、

「$f_i < 1 + \alpha$ または $f_i \geqq 2 - \alpha$」

のときは、さらに、

「f_i の仮数部 6 ～21ビット値の和が $\neq 0 \pmod 4$」 ならば、
「f_i の仮数部 1 ～23ビットの全ビット値を反転する」

という方法であった。この(SR/2)法に上述の長周期化の手法を適用し(ただし $\alpha = 0.34$)、最初の20000個の乱数値について、テスト（Ⅰ）モンテカルロ法による π 計算値の相対誤差、（Ⅱ）文字 0 ～ 9 の出現頻度の χ^2 検定、（Ⅲ）文字 0 の出現間隔の χ^2 検定、（Ⅳ）乱数値の Kolmogorov-Smirnov 検定 (K^+, K^-)、（Ⅴ）単純上昇連・下降連テスト、（Ⅵ）4 枚の 0 ～ 9 カードによる古典ポーカーテスト、（Ⅶ）遅れ 1 および 2 の系列相関テスト、（Ⅷ）衝突テスト を行った。結果は以下の通りである。

テスト（Ⅰ）	（Ⅱ）	（Ⅲ）	K^+（Ⅳ） K^-
−0.006491	0.5387	0.9292	0.2333　0.6576

上昇 （Ⅴ） 下降	（Ⅵ）	遅れ1（Ⅶ） 遅れ2	（Ⅷ）
0.03425△　0.6623	0.3919	0.002433　0.0112	48

（この節の検定の詳細については［2］を参照のこと。）また、80000個の乱数値に対してKolmogorov-Smirnov検定値 K^+_{80000}［K^-_{80000}］を計算し、これを10000回繰り返してその分布に対する χ^2 検定を行った結果は0.2299［0.9274］であった。

これらの結果から(SR/2)法に対する長周期化の手法が十分に機能していることが分かる。使われている定数は $p = 49933453$, $q = 22801201$, $r = 491377$, $s = 47513$, $a = 1920000$, $b = 48060000$ であるが、これらは特に固定化されたものではなく、本文中の条件を満たせばある程度自由に選ぶことができ、それに対応して α を再設定すれば、また別の乱数系列が得られることになる。この意味でかなり自由な長周期化手法ということができよう。また、この方法は、従来の多くの乱数発生法と異なり、再帰的に乱数を計算せず、k 番目の乱数を直接 $z\,(r_k,\ s_k)$ として計算するという特徴を持っている。しかしながら、直接計算のため乱数発生に時間がかかるという欠点もある。最後に本報告の現実的な応用として、長周期化された(SR/2)法（周期$\approx 10^{15}$）の、Delphi（Pascal）言語によるプログラムリストを、直ちに利用可能な形式で付録に載せておく。

付　　録

周期 1,138,542,698,477,053 をもつ乱数列を生成するプログラムを以下に記す。使用言語はDelphi 4である。

SRIni(*nn*); で *nn* 番目の乱数値から生成開始することを指定し、SR2Dg4; で10進4桁の整数乱数を得、SR2Byt; で1バイトの整数乱数（0〜255）を得る。*nn* は 0.0 〜 1138542698477052.0 内の1つの数であり、必ず *xxxx*.0 の形式（あるいは倍精度変数）で指定する。なお、このプログラムでは、本文中の SRl (n, i) のかわりに SRl $(n+3,\ i+1)$ を使っている。これは、SRl (n, i) で $i = 0$ となるのを避けるためである。本プログラム使用にあたっては、［2］の Appendix A の Remarks を一読することをお勧めする。

```
type
  DblI2 = record
    case I2 : Boolean of
      True  : (L, H : integer);
      False : (Dbl : double);
  end;
type
  SRlpPara = record
    p : integer; q : integer; r : integer; s : integer;
    a : integer; b : integer; rk : integer; sk : integer;
  end;
const   {constants for long-periodization}
  SRPara : SRlpPara = (p:49933453; q:22801201; r:491377; s:47513;
                       a:1920000; b:48060000; rk:0; sk:0);
  SR2alpha : double = 0.34;
{**********}
function  SR(x:double) : double;
  var
```

```
    i : integer;
    FLLH : DblI2;
  begin
    FLLH.Dbl:=1;
    for i:=1 to 24 do
      begin
        FLLH.Dbl:=FLLH.Dbl*x/i;
        FLLH.H:=(FLLH.H and $0007ffff) shl 1;  {shifted-mantissa H}
        if (FLLH.L and $80000000)<>0 then inc(FLLH.H);
        FLLH.L:=(FLLH.L and $f0000000) shl 1;  {shifted-mantissa L}
        FLLH.H:=FLLH.H or $3ff00000;           {set exponent 2^0}
      end;
    SR:=FLLH.Dbl;
  end;
{**********}
function SR2(x:double) : double;
  var
    i, q, B : integer;
    FLLH : DblI2;
  begin
    FLLH.Dbl:=SR(x);
    if ((FLLH.Dbl<1+SR2alpha) or (FLLH.Dbl>=2-SR2alpha))
        then
      begin
        q:=0;
        if (FLLH.L and $80000000)<>0 then inc(q);
        B:=1;
        for i:=1 to 15 do
          begin
            if (FLLH.H AND B)<>0 then inc(q);
            B:=B+B;
          end;
        if(q and 3)<>0 then
          begin
            FLLH.L:=Cardinal(FLLH.L)xor $e0000000;
            FLLH.H:=FLLH.H xor $000fffff;
          end;
      end;
    SR2:=FLLH.Dbl;
  end;
{**********}
function nextX : double; {compute the next x_i}
  var
    n , i: integer;
    XStep : double;
  begin
    SRPara.rk:=SRPara.rk+SRPara.r;
    if (SRPara.rk>=SRPara.p) then
      SRPara.rk:=SRPara.rk-SRPara.p;
    SRPara.sk:=SRPara.sk+SRPara.s;
    if (SRPara.sk>=SRPara.q) then
      SRPara.sk:=SRPara.sk-SRPara.q;
    n:=SRPara.sk+SRPara.a;
    if (SRPara.rk<n)then
      begin
        i:=SRPara.rk;
      end
    else
      begin
        i:=SRPara.rk-n;
        n:=SRPara.b-SRPara.sk;
      end;
    n:=n+3;
    inc(i);
    XStep:=16.0/n;          {store in memory}
    nextX:=16+XStep*i;
```

```
  end;
{**********}
procedure SRIni(n:double);    {set the starting point of x}
  var
    k, rkD, skD : double;
  begin
    k:=n;
    if (n<0) then k:=-n;
    rkD:=SRPara.r*(k-trunc(k/SRPara.p)*SRPara.p);
    rkD:=rkD-trunc(rkD/SRPara.p)*SRPara.p;
    skD:=SRPara.s*(k-trunc(k/SRPara.q)*SRPara.q);
    skD:=skD-trunc(skD/SRPara.q)*SRPara.q;
    SRPara.rk:=trunc(rkD);
    SRPara.sk:=trunc(skD);
    if n<0 then
      begin
        if (SRPara.rk<>0) then SRPara.rk:=SRPara.p-SRPara.rk;
        if (SRPara.sk<>0) then SRPara.sk:=SRPara.q-SRPara.sk;
      end;
  end;
{**********}
function SR2Byt : byte;         {generate a 1-byte random number}
  var
    x : double;
    FLLH : Dbl2;
  begin
    x:=nextX();                   {store in memory}
    FLLH.Dbl:=SR2(x);
    SR2Byt:=byte(FLLH.H and $ff);
  end;
{**********}
function SR2Dg4 : integer;    {generate a 4-digits random number}
  var
    x, y : double;
  begin
    x:=nextX();                   {store in memory}
    y:=SR2(x);                    {store in memory}
    SR2Dg4:=Trunc(y*1000000) mod 10000;
  end;
{*************************************************}
{** show 1,138,542,698,477,053 random numbers all ! **}
procedure TForm1.Button1Click(Sender: TObject);
  var
    i, j, SRran : integer;
  begin
    SRIni(0.0);                  {from 0.0 to 1138542698477052.0}
    {for i:=1 to 49933453 do}
      {for j:=1 to 22801201 do}
        begin
          SRran:=SR2Dg4;         {get a 4-digits decimal random number}
          {SRran:=SR2Byt;}       {get a 1-byte random number}
          Canvas.TextOut(20, 12, IntToStr(SRran));
        end;
  end;
```

参 考 文 献

[1] Knuth, D. E.: *The art of computer programming.* Vol. 2 (3rd ed.). Addison-Wesley, 1997.

[2] Yaguchi, H.: Randomness of Horner's rule and a new method of generating random numbers. Monte Carlo Methods and Appl., **6** (2000), 61–76.