三重大学教育学部研究紀要　第59巻　自然科学　(2008)　25 − 35頁

非再帰型擬似乱数SSRの再帰化による生成速度の高速化

谷口礼偉*

Speed-up of the non-recursive pseudo-random number generator SSR by recursive procedure

Hirotake YAGUCHI

要　旨

擬似乱数生成法SSRは、区分的にアフィン性を有するカオス写像の族$\Phi_x:[1,2)\to[1,2),\ x\in[1,2)$,において、$\Phi_{x_k}^{24}(w_0),\ k=1,2,\cdots$,を計算することにより$k$番目の乱数$z_k$を非再帰的に生成する方法である。この方法は並列計算に適した乱数生成法であるが、単一の計算機での実行を考えた場合、非再帰的な生成法であるため、他の再帰的な乱数生成法と比して生成速度が遅くなる。本論文は、SSRのアルゴリズムを単一の計算機で高速に実行するために、各x_kについてΦ_{x_k}を$w_k=\Phi_{x_k}(w_{k-1})$のごとく用いて、w_{k-1}からw_kを再帰的な計算で得る乱数生成法を考察し、生成される乱数の特性を検定し、さらに乱数生成の理論的背景について述べたものである。

非再帰型擬似乱数生成法SSR

最初に擬似乱数生成法SSRについて簡単にまとめておく。SSRは、

$$g_{w_0}(x)=w_0\circ\underbrace{x\circ x\circ x\circ\cdots\circ x\circ x}_{24},\quad x\in[1,2)$$

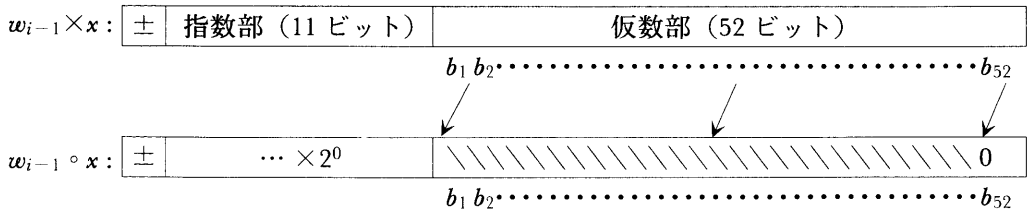
を　$w_i:=w_{i-1}\circ x,\ i=1,2,\cdots,24,\ g_{w_0}(x)=w_{24}$ と計算する方法を基本としている。ここで$w_{i-1}\circ x$は、以下のSSR計算を表す：

SSR計算（*S*implified *S*hift *R*eal computation）

i) $w_{i-1}\times x$を通常の倍精度浮動小数点計算で行い、

ii) 得られた値の仮数部の全てのビット値を1ビット左にシフトし、

iii) 指数部は$\cdots\times 2^0$となるように設定する。



* 三重大学教育学部数学教室

ここで $w \circ x$ を $\Phi_x(w)$ で表せば $\Phi_x(w) \in [1, 2)$ であり、$g_{w_0}(x) = \Phi_x^{24}(w_0)$ となる。実際の乱数は、x を $[1, 2)$ 内を一様に x_k, $k=1, 2, 3, \cdots$, と変化させて $g_{w_0}(x_k) = \Phi_{x_k}^{24}(w_0)$ を計算し、得られた値の最初の3桁を棄て、続く4桁を乱数値として採用する。ここで x_k は、

$$p=49933453,\ q=22801201,\ r=491377,\ s=47513,\ a=1920000,\ b=48060000,$$

とおいて（p, q, r, s は素数）、$(r_k, s_k) \equiv (r_k \bmod p,\ s_k \bmod q)$ を計算し

$$x_k = \begin{cases} 1.0 + \dfrac{r_k}{a+s_k} & \text{if } (a+s_k) > r_k \\ 1.0 + \dfrac{r_k-(a+s_k)}{a-s_k} & \text{if } (a+s_k) \leq r_k \end{cases}$$

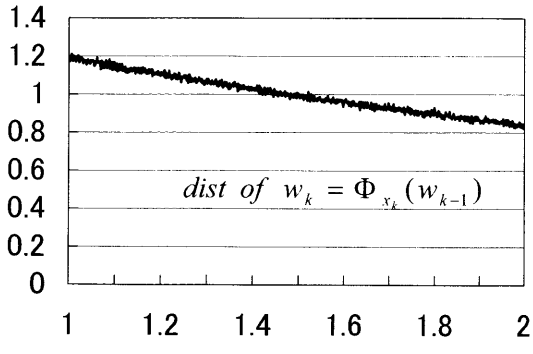
により定める。（p, q, r, s, a, b は一定の条件を満たしていれば何でも良い。）これから分かるように、SSR擬似乱数生成法では k 番目の乱数を、(r_k, s_k) から定まる x_k について $g_{w_0}(x_k)$ を直接計算することにより非再帰的に生成する。この非再帰的な乱数の生成は、従来からの乱数生成法には見られないSSRの特徴であり、k が決まれば直ちに k 番目の乱数値が計算できるので、生成すべき乱数列を並列生成することが可能である。

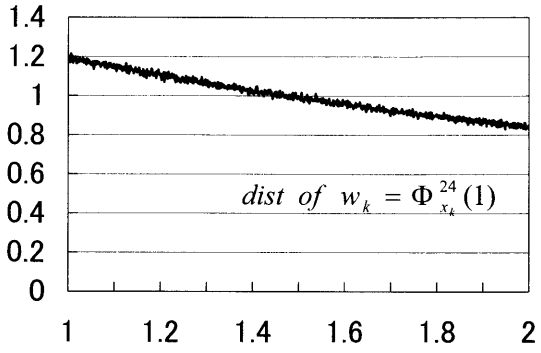
再帰型擬似乱数生成法 rSSR

SSRは乱数を非再帰的に生成するので、1つの乱数を得るための計算量がやや多く、生成速度は少なからず遅くなる。一方で、SSRは乱数の並列生成が可能であり、並列計算の並列度を上げれば上げるほど、同一の乱数列を生成する時間が短くなるので、SSRを大規模シミュレーション等の乱数源として使うことは特に有効である。（現実的には、SSRのアルゴリズムを完全整数演算化したSSIを使う。）しかしながら、現在使われている計算機は非並列型が主流であり、また、SSRは乱数の生成に乗算という非線形演算を用いているので、暗号用乱数としての使用可能性も考えられる。このような背景を考えれば、SSRの特徴である非再帰性を敢てあきらめてでも、SSRのアルゴリズムを使って、非再帰型SSRよりも高速に乱数を生成する方法を提供することは少なからず意義があることと思われる。

SSR擬似乱数生成法では k 番目の乱数値として、$g_{w_0}(x_k) = \Phi_{x_k}^{24}(w_0)$ の最初の3桁を棄て続く4桁を取り出している。このカオス写像 Φ_{x_k} を24回繰り返す操作が、非再帰的な乱数生成を可能とするものであるが、一方でまた、再帰的な方法に比して生成に時間がかかる原因にもなっている。したがって、SSR計算 Φ_x を用いて乱数を高速に生成するためには、この反対の方向を目指せばよい。すなわち、各 x_k について Φ_{x_k} を1回ないし2回程度使うにとどめ、かつ、再帰的に乱数を生成すればよい。

以下の左のグラフは $w_0=1.0$ とした、$w_k = \Phi_{x_k}(w_{k-1})$, $k=1, 2, \cdots, 10^7$, の値分布である。また、右のグラフは、SSR乱数生成法で使っている $\Phi_{x_k}^{24}(w_0)$, $k=1, 2, \cdots, 10^7$, の値分布である。





これらのグラフを見ると、同様の分布をしているように観察される。したがって、非再帰型と同様に SSR 計算$\Phi_x(w)$ を用いて、再帰的に擬似乱数を生成できる可能性が非常に高いことが知られる。

今後 $x_k=\Phi_{x_k}(w_{k-1})$ のように SSR 計算を再帰的に使用して乱数を生成する手法を「再帰型 SSR 擬似乱数生成法」（rSSR）と呼び、従来からの$\Phi^{24}_{x_k}(w_0)$ を使って非再帰的に乱数を生成する方法を「非再帰型 SSR 擬似乱数生成法」（nrSSR）と呼ぶことにし、本節においては、rSSR で乱数を生成する方法を具体的に述べ、次節以降でその乱数特性および理論的な背景を調べることにする。

まず、rSSR に K 改良［1］を適用した乱数生成法について述べる。

再帰型 K 改良 SSR 擬似乱数生成法（rSSR-K）

x_k, Φ_{x_k} は nrSSR と同様に定める。nrSSR の r=491377, s=47513 を$\hat{r}$=494041, $\hat{s}$=48049,に置き換えて求めた x=$\hat{x}_k$ について行う SSR 計算を$\Phi_{\hat{x}_k}$ で表す。

$$w_0 = 1.2718281828459, \qquad w_k = \Phi_{x_k}(w_{k-1}),$$
$$\hat{w}_0 = 1.8141592653589, \qquad \hat{w}_k = \Phi_{\hat{x}_k}(\hat{w}_{k-1})$$

とし $w_k-\hat{w}_k$ mod［1, 2）の上位 3 桁を棄て、続く 4 桁を乱数値とする。ここで、mod［1, 2）は $w_k-\hat{w}_k$ に +1 または +2 を加えて区間［1, 2）の値とすることを表す。乱数値としてビットが要求されるときは、$w_k-\hat{w}_k$ mod［1, 2）の仮数部の最初の 6 ビットを棄て、続く 16 ビットを乱数ビットとして採用する。

次に、上述の rSSR-K 系列とは異なるもう一つの rSSR-K 系列を準備しておき、$\Phi_x(w)$ の w との排他的論理和（XOR）をとる方法を紹介する。

再帰型 XK 改良 SSR 擬似乱数生成法（rSSR-XK）

x_k, Φ_{x_k}, $\hat{x}_k$, $\Phi_{\hat{x}_k}$ は rSSR-K と同様に定める。もう一つの rSSR-K 計算として、$w\circ x$=$\Phi_x(w)$ において仮数部のシフトを 1 ビットから 8 ビットに変えた計算をΨ_{x_k},$\Psi_{\hat{x}_k}$ で表す。

$$u_0 = 1.2718281828459, \qquad u_k = \Psi_{x_k}(u_{k-1}),$$
$$\hat{u}_0 = 1.8141592653589, \qquad \hat{u}_k = \Psi_{\hat{x}_k}(\hat{u}_{k-1})$$

とし $z_k = u_k - \hat{u}_k$ mod［1, 2）とおく（最初の rSSR-K 計算）。続いて

$$w_k = \Phi_{x_k}(w_{k-1} \text{ XOR } z_k) \qquad (w_0 = 1.2718281828459),$$
$$\hat{w}_k = \Phi_{\hat{x}_k}(\hat{w}_{k-1} \text{ XOR } z_k) \qquad (\hat{w}_0 = 1.8141592653589)$$

を計算し、$w_k-\hat{w}_k$ mod［1, 2）を求め（2 番目の rSSR-K 計算）、上位 3 桁を棄て、続く 4 桁を乱数値とする。ここで w XOR z は、仮数部のみについての XOR とする。

このようなrSSR-XKを導入した理由は、一つは再帰型SSR乱数生成法における乱数値の分布の理論的な解析を単純にするためであり、もう一つは、rSSR乱数のストリーム暗号用乱数としての応用可能性を考慮したとき、乱数生成の複雑性を増して暗号解読への耐性を強化するため及び暗号鍵として w_0, $\hat{w}_0$, u_0, $\hat{u}_0$ を使い、鍵のビット数を増やすためである。

生成速度の改良

再帰化されたアルゴリズムにより生成速度がどの程度改良されたかを見るため、乱数を100000000個生成する時間の比を求めてみた。32ビット環境（WindowsXP, Pentium M（1.73GHz),コンパイラはBcc 5.6）の下では

$$\text{nrSSR\_K : rSSR\_K : rSSR\_XK : MT} = 79 : 13 : 22 : 8 \text{ (seconds)}$$
$$= 1 : 0.165 : 0.278 : 0.101$$

であった。rSSR-KではnrSSR-Kに比して約6倍高速に乱数を生成していることが分かる。また、64ビット環境（Windows XP x64 edition, Pentium 4（2.8GHz),コンパイラはicl）では、

$$\text{nrSSR\_K : rSSR\_K : rSSR\_XK : MT} = 34 : 3 : 9 : 2 \text{ (seconds)}$$
$$= 1 : 0.088 : 0.265 : 0.059$$

であり、rSSR-Kは約11倍高速に乱数を生成している。この結果から、再帰化による乱数生成速度高速化の効果は格段であると言えよう。一方でrSSR-KにおけるSSR計算 $\Phi_x(w)$ の実行回数が、nrSSR-Kにおける回数の24分の1であるにもかかわらず、生成速度がこの数値を反映していないのは、再帰計算 $w_k=\Phi_{x_k}(w_{k-1})$ において w_* をメモリに保存しまた取り出す手間が増えること、および x_k を計算する時間の重みが $\Phi_{x_k}(w)$ を実行する時間に比して相対的に増すためと考えられる。

再帰型擬似乱数生成法rSSRの乱数特性

本節では上述した再帰型擬似乱数生成法rSSR-K、rSSR-XKの乱数特性を、NISTおよびTestU01の検定を適用して調べる。

NISTの検定

NISTの検定の実際については［2］と同様であるので詳細は省略する。ここではNISTの検定を16回適用して平均値をとり、得られた値の最大値・最小値を記しておく。また、比較のためのメルセンヌツイスタの結果も再録しておく。

【rSSR-K】

［(p 値検定値）平均 max] = 0.670988 at nonperiodic-templates [i=112]
［(p 値検定値）平均 min] = 0.270036 at nonperiodic-templates [i=64]
［($p\geq 0.01$ 割合）平均 max] = 0.992869 at random-excursions-variant [i=184]
［($p\geq 0.01$ 割合）平均 min] = 0.987687 at nonperiodic-templates [i=95]

【rSSR-XK】

［(p 値検定値）平均 max] = 0.708240 at nonperiodic-templates [i=74]
［(p 値検定値）平均 min] = 0.273726 at random-excursions-variant [i=178]

[($p \geq 0.01$ 割合) 平均 max] = 0.992000 at nonperiodic-templates [i=61]
[($p \geq 0.01$ 割合) 平均 min] = 0.987256 at random-excursions [i=163]

【MT】

[(p 値検定値) 平均 max] = 0.721409 at rank [i=6]
[(p 値検定値) 平均 min] = 0.304160 at random-excursions-variant [i=179]
[($p \geq 0.01$ 割合) 平均 max] = 0.992250 at nonperiodic-templates [i=138]
[($p \geq 0.01$ 割合) 平均 min] = 0.987500 at fft [i=7]

これらの結果から、rSSR は NIST の検定について特に問題がないことが分かる。

TestU01 による検定

TestU01 は L'Ecuyer らによる乱数検定プログラムの集大成である。TestU01 での検定は、規模に応じて SmallCrush, Crush, BigCrush と名付けられている。各 Crush を構成する検定は、その検定結果を p-value で表示する。この p-value は、区間 [0, 1] を一様に分布する統計量であり、0 または 1 に極端に近い値がでた時には、乱数性が疑わしいとされる。(TestU01 の詳細はhttp://www. iro. umontreal. ca/~simardr/testu01/tu01. html にある。) 以下に BigCrush (Nov. 23, 2006) の結果を述べる。

【rSSR-K】

```
========= Summary results of BigCrush =========
 Version:          TestU01 1.2.1
 Generator:        My rSSR-K implementation
 Number of statistics:  160
 Total CPU time:   52:55:38.43
 The following tests gave p-values outside [0.001, 0.9990]:
  (eps  means a value < 1.0e-300):
  (eps1 means a value < 1.0e-15):
       Test                          p-value
 ----------------------------------------------------------------
 87  LongestHeadRun, r = 27          0.9998
 ----------------------------------------------------------------
 All other tests were passed
```

#87 のテストの結果が統計的な揺らぎなのか本質的なものであるかを確認するため、さらに #87 のテストを 3 回繰り返した結果は次の通りである。

```
========= Summary results of BigCrush =========
 Version:          TestU01 1.2.1
 Generator:        My rSSR-K implementation
 Number of statistics:  6
 Total CPU time:   01:14:56.89
 All tests were passed
```

【rSSR-XK】

```
========= Summary results of BigCrush =========
Version:          TestU01 1.2.1
Generator:        My rSSR-XK implementation
Number of statistics:  160
Total CPU time:   82:44:01.50
All tests were passed
```

【MT】

```
========= Summary results of BigCrush =========
Version:          TestU01 1.2
Generator:        My MT implementation
Number of statistics:  160
Total CPU time:   15:55:26.42
The following tests gave p-values outside [0.001, 0.9990]:
 (eps  means a value < 1.0e-300) :
 (eps1 means a value < 1.0e-15) :
       Test                          p-value
----------------------------------------------------------------
80  LinearComp, r = 0                 1 - eps1
81  LinearComp, r = 29                1 - eps1
----------------------------------------------------------------
All other tests were passed
```

#80, #81 について、テストを 3 回繰り返しても同様の結果を得る。

以上の結果から、rSSR は TestU01 の検定についても特に問題がないことが分かる。 TestU01 を実行する時間は、rSSR は MT に比して 4 倍程度遅いが、 rSSR を整数演算化した rSSI では、1.5 倍程度になる。(rSSI については将来の論文で扱う予定である。)

再帰型 SSR の乱数性の数理的背景

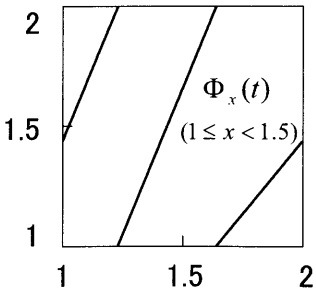
本節では rSSR 乱数値の分布特性に関する考察を行う。まず、SSR 計算 $w \circ x = \Phi_{x_k}(w)$ で現れる関数 Φ_x: $[1, 2) \to [1, 2)$ は、

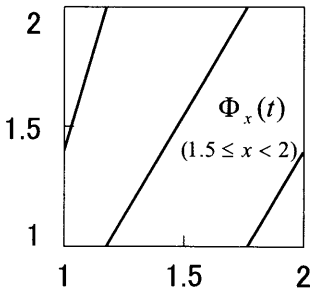
$1 \leq x < 1.5$ のとき

$$\Phi_x(t) = \begin{cases} 2xt-1 & (1 \leq t < \dfrac{3}{2x} \text{ のとき}) \\ 2xt-2 & (\dfrac{3}{2x} \leq t < \dfrac{2}{x} \text{ のとき}) \\ xt-1 & (\dfrac{2}{x} \leq t < 2 \text{ のとき}) \end{cases}$$

$1.5 \leq x < 2$ のとき

$$\Phi_x(t) = \begin{cases} 2xt-2 & (1 \leq t < \dfrac{2}{x} \text{ のとき}) \\ xt-1 & (\dfrac{2}{x} \leq t < \dfrac{3}{x} \text{ のとき}) \\ xt-2 & (\dfrac{3}{x} \leq t < 2 \text{ のとき}) \end{cases}$$





と表されることに注意しておく。実際、このΦ_xは［2］にもあるように、

i) w_{i-1}をx倍する写像をϕ_xと書き、

ii) 変数ϕ_x（w_{i-1}）の指数部を…$\times 2^0$とする写像をϕ_eと書き、

iii) さらにϕ_e（ϕ_x（w_{i-1}））の仮数部を1ビットシフトする写像をϕ_sと書いたとき、

$w_i = \phi_s(\phi_e(\phi_x(w_{i-1}))) = (\phi_s \circ \phi_e \circ \phi_x)(w_{i-1}) \equiv \Phi_x(w_{i-1})$ として現れるので、

$$\phi_x(t) = xt, \quad \phi_e(t) = \begin{cases} t & (1 \le t < 2 \text{ のとき}) \\ \dfrac{t}{2} & (2 \le t < 4 \text{ のとき}) \end{cases}, \quad \phi_s(t) = \begin{cases} t-1 & (1 \le t < \dfrac{3}{2} \text{ のとき}) \\ t-2 & (\dfrac{3}{2} \le t < 2 \text{ のとき}) \end{cases}$$

を具体的に代入すれば求められる。このSSR計算$\Phi_x(w)$の関数表現を利用すると、$\Phi_x(w)$でwが［1, 2）を一様に分布するときの、$\Phi_x(w)$の分布の密度関数$p_x(t)$が次のように計算される。

$\mathcal{X} = [1, 2)$とし、$\mathcal{X}$上のBorel集合体を$\mathcal{B}$, $(\mathcal{X}, \mathcal{B})$上のLebesgue測度を$\lambda$とすると,任意の$A \in \mathcal{B}$に対し

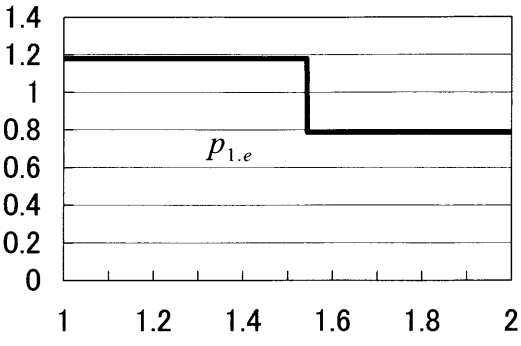
$$\lambda(\{t \in \mathcal{X} \mid \Phi_x(t) \in A\}) = \int_{\{u \mid u \in A\}} p_x(u)\, du,$$

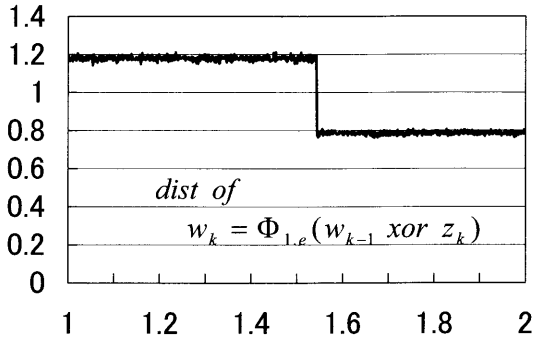
ただし

$$1 \le x < 1.5 \text{ のとき} \quad p_x(t) = \begin{cases} \dfrac{1}{2x} + \dfrac{1}{x} & 1 \le t < 2x-1 \text{ のとき} \\ \dfrac{1}{x} & 2x-1 \le t < 2 \text{ のとき} \end{cases},$$

$$1.5 \le x < 2 \text{ のとき} \quad p_x(t) = \begin{cases} \dfrac{2}{x} & 1 \le t < 2x-2 \text{ のとき} \\ \dfrac{1}{2x} + \dfrac{1}{x} & 2x-2 \le t < 2 \text{ のとき} \end{cases}$$

であるので、$\Phi_x(w)$においてwが［1, 2）を一様に分布するとき、$\Phi_x(w)$の分布の密度関数は$p_x(t)$になることが分かる。rSSR-XKの計算中に、w_{k-1} XOR z_kがあるが、このz_kはrSSR-Kと同じアルゴリズムで生成されるので、z_kは区間［1, 2）をほぼ一様に分布すると予想される。よって$w_k = \Phi_{x_k}(w_{k-1}$ XOR $z_k)$でx_kを$1.e$=1.2718281828459に固定した$\Phi_{1.e}(w_{k-1}$ XOR $z_k)$, k=1, 2, …, の分布は$p_{1.e}$になると予想される。実際、下図左の$p_{1.e}$のグラフ、下図右の$\Phi_{1.e}(w_{k-1}$ XOR $z_k)$, $k=1, 2, \cdots, 107$, の分布のグラフを見てみるとそのことが分かる。





次に $w_k=\Phi_{x_k}(w_{k-1}$ XOR $z_k)$, $k=1, 2, \cdots$, の分布を考察する。この場合、各 k について Φ_{x_k} は一度しか使われないので、上述の議論はそのままでは使えない。しかし x_k が $[1, 2)$ を一様に分布すること、および Φ_x と $\Phi_{x'}$ について、x と x' が近ければ、p_x と $p_{x'}$ は互いに他を近似していると考えられることから、$[1, 2)$ 内に多数の点 $\alpha_1, \alpha_2, \cdots, \alpha_{N-1}, \alpha_N$ を一様にとり、値 $w_k = \Phi_{x_k}(w_{k-1}$ XOR $z_k)$ の分布に関しては、x_k に最も近い α_{j_k} を選び、$p_{\alpha_{j_k}}$ で近似させることにより、上述の議論が使え、このときの分布密度は $\frac{1}{N}\Sigma_{j=1}^{N} p_{\alpha_j}$ になる。さらに $N\to\infty$ として近似の精度を上げれば $\check{H}(t)=\int_1^2 p_x(t)\,dx$ となる。実際に $\check{H}(t)$ を計算すると、$p_x(t)$ が

$$p_x(t)=\begin{cases}\dfrac{1}{x} & 1\le x\le\dfrac{t+1}{2}\ \text{のとき}\\[2ex] \dfrac{1}{2x}+\dfrac{1}{x} & \dfrac{t+1}{2}<x\le\dfrac{t+2}{2}\ \text{のとき},\\[2ex] \dfrac{2}{x} & \dfrac{t+2}{2}<x<2\text{のとき}\end{cases}$$

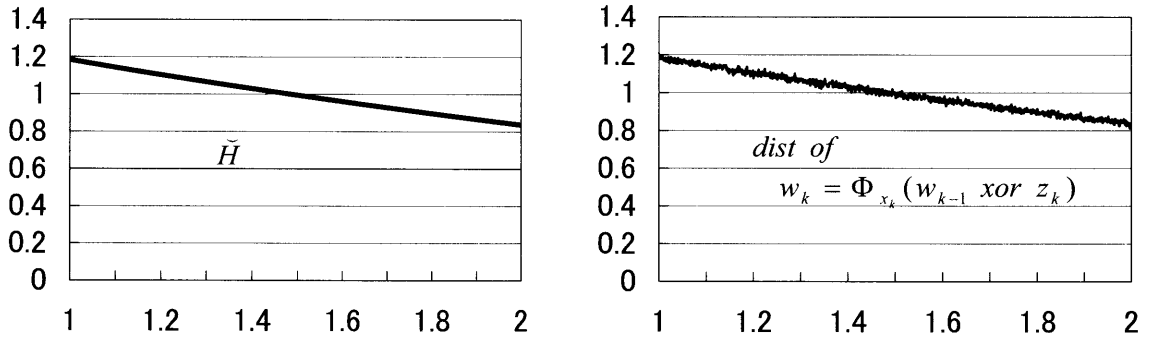
のように書き直されることに注意して、

$$\begin{aligned}\check{H}(t)&=\int_1^{(t+1)/2}\frac{1}{x}\,dx+\int_{(t+1)/2}^{(t+2)/2}\left\{\frac{1}{2x}+\frac{1}{x}\right\}dx+\int_{(t+2)/2}^{2}\frac{2}{x}\,dx\\&=3\log 2-\frac{1}{2}\log(t+1)-\frac{1}{2}\log(t+2)\end{aligned}$$

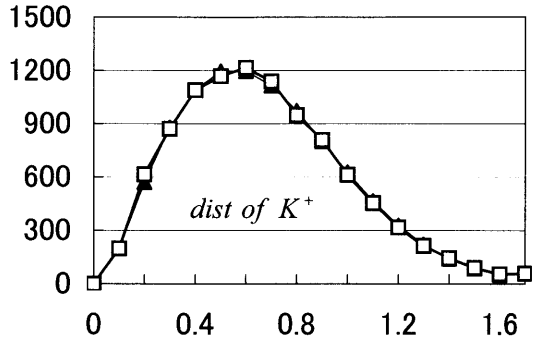
となる。この $\check{H}(t)$ は次のようにして直接求めることもできる。$\Phi_x(w)$ の x と w を互いに独立な $[1, 2)$ を一様に分布する確率変数 $X(\omega)$, $Y(\omega)$ と考えて、$\Phi_{X(\omega)}(Y(\omega))=(\phi_s\circ\phi_e)(X(\omega)Y(\omega))$ の確率分布関数 $F(t)$ をまず求める。$a, b\in[1, 2)$ に対し $\mathrm{Prob}(a\le X\le b)=\mathrm{Prob}(a\le Y\le b)=b-a$ であるので、$t\in[1, 2)$ に対し、

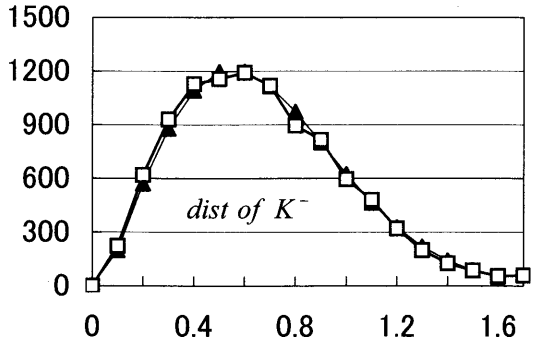
$$
\begin{aligned}
F(t) &= \mathrm{Prob}(1 \le \phi_s(\phi_s(XY)) \le t) \\
&= \mathrm{Prob}(1 \le \phi_e(XY) \le 1 + \frac{t-1}{2}) + \mathrm{Prob}(\frac{3}{2} \le \phi_e(XY) \le \frac{3}{2} + \frac{t-1}{2}) \\
&= \mathrm{Prob}(1 \le XY \le \frac{1}{2} + \frac{t}{2}) + \mathrm{Prob}(\frac{3}{2} \le XY \le 1 + \frac{t}{2}) \\
&\qquad + \mathrm{Prob}(2 \le XY \le 1+t) + \mathrm{Prob}(3 \le XY \le 2+t) \\
&= \int_1^{\frac{1}{2}(1+t)} \left\{ \frac{1}{2}(1+t)\frac{1}{x} - 1 \right\} dx \\
&\quad + \left[\int_1^{\frac{3}{2}} \left\{ \frac{1}{2}(2+t)\frac{1}{x} - \frac{3}{2} \cdot \frac{1}{x} \right\} dx + \int_{\frac{3}{2}}^{\frac{1}{2}(2+t)} \left\{ \frac{1}{2}(2+t)\frac{1}{x} - 1 \right\} dx \right] \\
&\quad + \left[\int_1^{\frac{1}{2}(1+t)} (2 - \frac{2}{x})\, dx + \int_{\frac{1}{2}(1+t)}^{2} \left\{ (1+t)\frac{1}{x} - \frac{2}{x} \right\} dx \right] \\
&\quad + \left[\int_{\frac{3}{2}}^{\frac{1}{2}(2+t)} (2 - \frac{3}{x})\, dx + \int_{\frac{1}{2}(2+t)}^{2} \left\{ (2+t)\frac{1}{x} - \frac{3}{x} \right\} dx \right] \\
&= (t-1) + (3t-2)\log 2 + \frac{3}{2}\log 3 - \frac{1}{2}(1+t)\log(1+t) - \frac{1}{2}(2+t)\log(2+t)
\end{aligned}
$$

となる。これを微分すれば $F'(t) = \check{H}(t)$ を得る。以下に、$\check{H}(t)$ のグラフ（左）と $w_k = \Phi_{x_k}(w_{k-1} \mathrm{XOR}\, z_k)$, $k = 1, 2, \cdots, 10^7$, の分布のグラフ（右）を載せておく。



ちなみに、20000 個の $w_k = \Phi_{x_k}(w_{k-1} \mathrm{XOR}\, z_k)$ を発生させ、値が区間 $[1+0.0001\,i,\ 1+0.0001(i+1))$, $i = 0, 1, \cdots, 10000$, に入る回数についての確率分布を $\check{H}(t)$ と比べて、Kolmogorov-Smirnov 統計量 κ^+ および κ^- を計算することを 1 万回繰り返したときの、κ^+ および κ^- の分布は下図のようになる。





この分布について、χ^2 検定を行った結果はそれぞれ 0.9729, 0.0593 である。これより rSSR-XK における w_k の分布は、$\check{H}(t)$ で記述できることが分かる。

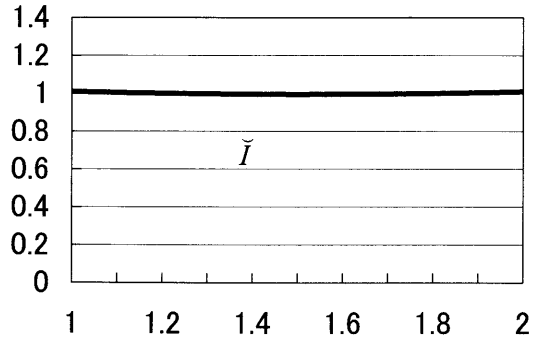
rSSR-XK では、写像Φのカオス性により w_k と $\hat{w}_k$ が独立的に振る舞うことを利用して、

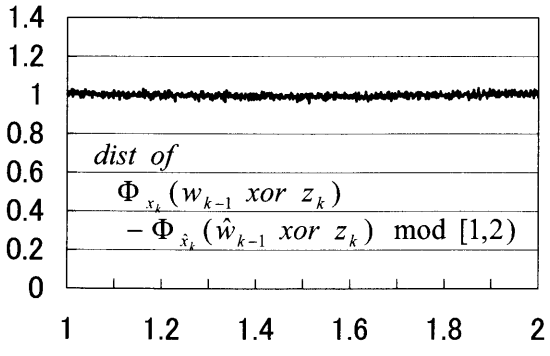
$$w_k - \hat{w}_k = \Phi_{x_k}(w_{k-1} \text{ XOR } z_k) - \Phi_{\hat{x}_k}(\hat{w}_{k-1} \text{ XOR } z_k) \mod [1, 2)$$

により乱数特性を改良している（K 改良 [1]）。この値の分布は畳み込み

$$\check{I}(y) = \int_1^{1+y} \check{H}(t)\,\check{H}(t-y+1)\,dt + \int_{1+y}^{2} \check{H}(t)\,\check{H}(t-y)\,dt$$

で得られるので、$\check{H}(t)$ に上述の関数表現を使って数値積分を行なうと、下図左のグラフを得る。特性が格段に改善されていることが分かる。rSSR-XK における $w_k - \hat{w}_k$, $k=1, 2, \cdots, 10^7$, の分布は下図右のようになり $\check{I}(y)$ と同様平坦である。（実際の rSSR-XK 乱数は、$w_k - \hat{w}_k$ の上位 3 桁を棄て、続く 4 桁を乱数値として使うのでさらに平坦な分布となる。）





まとめ

以上をまとめると次のようになる。(1) 非再帰型 SSR 乱数生成法のアルゴリズムを再帰的に使用することにより、乱数生成速度を格段に高速化することができる。(2) SSR アルゴリズムを再帰的に使用しても、乱数特性に特に問題はない。(3) 再帰型 SSR 乱数生成法で生成される乱数値の分布について、その理論的背景を説明することができる（rSSR-XK)。(4) 再帰的に乱数を生成するため並列生成ができず、大規模並列計算機を使って乱数特性を短時間に調べることができなくなったのは残念である。

参考文献

[1] 久保泉,谷口礼偉：カオス写像で生成される擬似乱数の Perron-Frobebnius 作用素による解析. 日本数学会 2005 年度年会, 一般講演.

[2] 谷口礼偉：実数演算を用いた非再帰的な擬似乱数生成の整数演算化. 三重大学教育学部研究紀要, 第 58 巻（自然科学）(2007), 5-11.