三重大学教育学部研究紀要　第58巻　自然科学　(2007)　5－11頁

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

谷　口　礼　偉*

Integralization of a Non-Recursive Pseudo-Random Number Generator Based on Floating Point Computations

Hirotake YAGUCHI

要　旨

カオス写像の繰り返しにより、非再帰的に擬似乱数を生成する実数シフト法は、本質的に浮動小数点演算を用いているため、計算機システムに依存して微妙に異なる乱数値を生成することがある。本研究は、実数シフト乱数生成メカニズムの核心部分を64ビット整数の乗算とシフトによるアルゴリズムで実現し、実数シフト法の完全整数演算化を行ったものである。

実数シフト法（SSR法）と浮動小数点演算

非再帰的に擬似乱数を生成する「単純実数シフト法」(Simplified Shift-Real method, SSR法）は、以下の部分的アフィン関数 $\Phi_x : [1, 2) \to [1, 2)$

$1 \le x < 1.5$ のとき

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

$1.5 \le x < 2$ のとき

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

を用いて、$u_{24} = \Phi_x^{24}(1)$ を計算し、u_{24}の最初の3桁を棄て続く4桁を乱数値として取り出すものである。また、xをx_k, $k = 1, 2, 3, \cdots$, と変化させて、多くの乱数値を得ている。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)、

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

$$\text{ただし、} \quad (r_k,\ s_k) \equiv (rk \bmod p,\ sk \bmod q),$$

* 三重大学教育学部数学教室

と定められるが、その定め方はさほど重要ではない。重要なことはx_kが［1, 2）上を一様に分布するということである。

SSR法に現れる関数$\Phi_x(t)$は、コンピュータによる浮動小数点計算で、倍精度変数u_0：＝1から始めて、

i）u_{k-1}をx倍する写像をϕ_xと書き、

ii）変数$\phi_x(u_{k-1})$の指数部を$\cdots\times 2^0$とする写像をϕ_eと書き、

iii）さらに$\phi_e(\phi_x(u_{k-1}))$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、

$u_k=\phi_s(\phi_e(\phi_x(u_{k-1})))=(\phi_s\circ\phi_e\circ\phi_x)(u_{k-1})\equiv\Phi_x(u_{k-1})$として現れる。SSR法での$\phi_s$の仮数部のビットシフトは1であるが、［2］で見たように、拡張SSR法では、ϕ_sのシフトサイズを2にして8桁の乱数を発生させている。このことから、乱数を発生させるためには、ビットシフトのサイズには特にこだわる必要がないことが分かる。

一方、SSR法での乱数生成には浮動小数点演算が必須であるが、浮動小数点演算は計算機の性能を左右するため、各MPUメーカーが高速化のために工夫をこらしている部分であり、メーカーごとに差違がある。浮動小数点変数については一応IEEEの標準があるが、浮動小数点数の計算機内部での具体的な計算方法については特に規定があるわけではない。このため、例えば2つの浮動小数点数を乗算する場合、MPU内部で処理できるビット長には制限があるので、途中の演算結果を止むを得ず丸めて処理することになり、最終的に結果を倍精度変数の仮数部52ビットに収める際、仮数部の下位ビットが計算機システムにより微妙に異なってしまうことがある。このことは、SSR法では、計算機システムが異なると、微妙に異なる乱数列が生成される可能性があることを意味している。

乱数を使ったモンテカルロシミュレーション等においては、乱数列の生成法によらない統計的な性質が解析の対象になっているので、生成される乱数列が計算機システムによって微妙に違っていても、それほど問題にはならないと思われるが、例えば乱数を暗号処理の目的で使う場合には致命的な欠陥になる。いずれにしても、乱数生成法の可搬性は重要な問題であり、可搬であるに越したことはなく、実数シフト法ではそれが可能であるので、以下に現時点までの研究成果を詳述する。

実数シフト法の整数演算化

実数シフト法（SSR法）の整数演算化を考える際、単に浮動小数点演算を整数演算でソフト的にエミュレートするだけでは、実行速度が極端に下がってしまい意味はない。幸い最近は64ビットの整数演算が可能な計算機が普及してきているので、64ビット×64ビットの整数乗算を行ったときに自然に発生するオーバーフローのビットサイズが、実数シフト法の仮数部のシフト量に相当するように調整することで、整数演算化を実現できる。この場合、オーバーフローの桁数を敢て固定化しないようにすると、演算の高速化がはかれる。

以下に述べる単純整数シフト計算（Simplified Shift-Integer computation, SSI計算）は、SSR法のベースをなす実数シフト計算（SSR計算）に対応するものである。

［SSI計算］

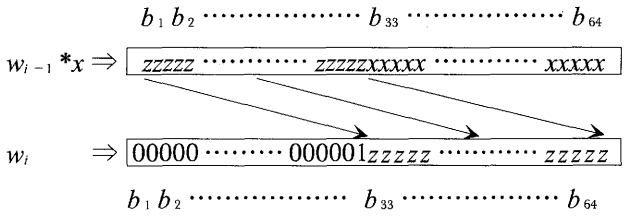
w_0, xを64ビット整数変数とする。

$$G_{w_0}(x)=w_0\underbrace{\circ x\circ x\circ x\ \cdots\ \circ x^{*}x}_{23}$$

を次のように計算する（$\circ$の意味）。

（i）w_{i-1} *x を通常の 64 ビット整数乗算で行う（オーバーフローは無視する）、

（ii）（i）の結果を 32 ビット右にシフトし、空いた $b_1\,b_2$ …… b_{32} を 000 …… 001 とおき、得られた結果を w_i とする、



（iii）(i), (ii) を $i=1, 2, \cdots, 22$ について繰り返す、

（iv）最後に、w_{22} *x を通常の 64 ビット整数乗算で行い（オーバーフローは無視）、SSI 計算値 $G_{w_0}(x)$ とする。

この SSI 計算をもとに、K 改良 SSR 乱数生成法（[1][2]）に相当する K 改良 SSI 乱数生成法（SSIK）を以下のように構成する。

[SSIK 乱数生成法]

p=0x7ffffffe1　　q=0x7ffffffcf　　r=0x39f750241

s=0x32f50fee9　　w_0=0x18237449a　　v_0=0x1dda73ad3

とする。

$$(r_k,\ s_k)=(r_k \bmod p,\ s_k \bmod q),\ k=1,\ 2,\ 3,\ \cdots,$$

x=0x88237449a,　　y=0xbdda73ad3

とし、

$$x_k=(x \text{ xor } r_k),\qquad y_k=(y \text{ xor } s_k)$$

とおく。そして、2 つの SSI 計算 $G_{w_0}(x_k)$, $G_{v_0}(y_k)$ の差

$$G_{w_0}(x_k)-G_{v_0}(y_k)$$

をとり、得られた 64 ビット整数の最初の 16 ビットを棄て、続く 32 ビットを k 番目の乱数値とする。

もう少し詳しく SSIK 乱数生成法を見てみる。w_i, x_k は、

$$0x100000000 \le w_i \le 0x1ffffffff,\quad 0x800000000 \le x_k \le 0xffffffff$$

であるので、整数乗算 w_i*x_k を行うと、

$$0x8000000000000000 \le w_i{*}x_k < 0x1fffffffffffffff$$

となるが、このうち下位 64 ビットがメモリーに保存されるので、オーバーフローするビット数は 4〜5 ビットである。SSI 計算の（ii）の右シフトでは、メモリーに保存された 64 ビットのうちの上位 32 ビットが $b_{33}\,b_{34}$……b_{64} として残り、$b_1\,b_2$……b_{32} には 000……001 がセットされるので、SSR 計算の仮数部の左シフトに対応する SSI 計算のビットシフト数は、3〜4 ビットになる。このプロセスが SSI 計算の（iii）−（iv）により、23 回繰り返されることになるので、平均 3.5×23=80.5 ビット相当のシフトが SSI 計算でなされていると考えられる。

また、SSIK 乱数の周期であるが、p, q, r, s はいずれも素数であるので、r_k, s_k したがって x_k, y_k の周期はそれぞれ p, q となり、結局 $G_{w_0}(x_k)-G_{v_0}(y_k)$ の周期は、

$$pq=34359738337\times34359738319\approx1.18\times10^{21}$$

となる。

このSSIK乱数の乱数特性は、以下に述べるSSIX乱数の特性とともに次節で述べる。

X改良SSI乱数生成法（SSIX）は、X改良SSR乱数生成法（[1][2]）に相当するSSI乱数生成法である。

[SSIX乱数生成法]

$$p=0\text{x}7\text{fffffffe}1 \qquad q=0\text{x}7\text{fffffffffc}7 \qquad r=0\text{x}39\text{f}750241$$

$$s=0\text{x}32\text{f}50\text{fef}7\text{e}7 \qquad w_0=0\text{x}18237449\text{a} \qquad v_0=0\text{x}1\text{dda}73\text{ad}3$$

$$(r_k,\ s_k)=(r_k \bmod p,\ s_k \bmod q),\ k=1,\ 2,\ 3,\ \cdots,$$

$$x=0\text{x}88237449\text{a}, \qquad y=0\text{xecbdda}73\text{ad}3$$

$$x_k=(x \text{ xor } r_k), \qquad y_k=(y \text{ xor } s_k)$$

とする。

$$\widehat{G}_{v_0}(y)=v_0 \underbrace{\circ y \circ y \circ y \cdots \circ y^{*}y}_{10}$$

をSSI計算の$G_{w_0}(x)$と同様に計算する（掛け算の回数が10回であることに留意）。そして、64ビット整数

$$\widehat{G}_{w_0}(x_k) \text{ xor } G_{v_0}(y_k)$$

の最初の16ビットを棄て、続く32ビットをk番目の乱数値とする。

$\widehat{G}_{v_0}(y)$でのオーバーフローの発生状況を見ると、

$$0\text{x}100000000 \le w_i \le 0\text{x}1\text{fffffffff}, \quad 0\text{x}80000000000 \le y_k \le 0\text{x}\text{fffffffffff}$$

であるので、整数乗算$w_i{}^{*}y_k$を行うと、

$$0\text{x}8000000000000000000 \le w_i{}^{*}y_k < 0\text{x}1\text{fffffffffffffffffff}$$

となり、このうち下位64ビットがメモリーに保存されるので、オーバーフローするビット数は12〜13ビットである。 したがって、SSR計算の仮数部の左シフトに対応するSSIのビットシフト数は、11〜12ビットになる。このプロセスが10回繰り返されることになるので、平均11.5×10＝115ビット相当のシフトが$\widehat{G}_{v_0}(y)$の計算で行われていると考えられる。SSIX計算では、$\widehat{G}_{v_0}(y)$でのシフトの繰り返し回数が$G_{w_0}(x_k)$より少ないので、SSIK計算より計算速度が若干速くなる。また、乱数の周期は、

$$pq=34359738337\times 8796093022151 \approx 3.02\times 10^{23}$$

で、SSIKより約250倍長い。

NISTのプログラムによる乱数特性の検定

米国のNIST（http://csrc.nist.gov/rng/）が提供している乱数検定プログラムを使って、SSIK、SSIXの乱数特性を調べてみる。検定法は［2］とまったく同じであるが、念のため要点を記しておく。

NISTのプログラムVersion 1.8は、以下の15種類188テストから構成されている（テスト名のあとの括弧はテスト数、ただしテスト数1は省略）：

frequency, block-frequency,

cumulative-sums (2), runs,

longest-run,
fft,
overlapping-templates,
approximate entropy,
random-excursions-variant (18),
linear-complexity.
rank,
nonperiodic-templates (148),
universal,
random-excursions (8),
serial (2),

検定実行時のパラメータは、block-frequency テストのブロック長を 20000 に変更した以外は既定値のままである。また、各テストに供されたデータは 1 ギガビットであり、各テストは、1 メガビットを使う検定を 1000 回繰り返し、p 値を 1000 個出力する。この p 値は一様分布をするとされるので、一様分布性に関する χ^2 検定を適用してその検定値を求め、また、p 値が 0.01 以上であるものの割合を、以下のような finalAnalysisReport として出力する：

RESULTS FOR THE UNIFORMITY OF P-VALUES AND THE PROPORTION OF PASSING SEQUENCES

C1	C2	C3	C4	C5	C6	C7	C8	C9	C10	P-VALUE	PROPORTION	STATISTICAL TEST
108	110	78	106	104	108	109	106	77	94	0.106877	0.9890	frequency
94	84	82	102	110	114	124	115	92	83	0.014051	0.9910	block-frequency
107	91	113	85	100	109	106	101	101	87	0.522100	0.9890	cumulative-sums
105	116	101	81	105	92	93	101	100	106	0.516113	0.9900	cumulative-sums
	...	...	...	...								

（表中の、P-VALUE が χ^2 検定値、PROPORTION が $p \geq 0.01$ であるものの割合である。）この NIST のプログラムを 16 回繰り返し、テスト数が多い nonperiodic-templates, random-excursions, random-excursions-variant の各テストを除いた P-VALUE 値を一覧表にして示す。また、テストごとに 16 個の P-VALUE の平均値も表示する。結果は以下の通りである（小数点以下 6 桁の数値を、小数点以下 3 桁で切り捨てて表示)。

SSIK に対する NIST 乱数検定プログラムの結果（p 値の χ^2 検定値）

#	Freq	BFreq	Cusum	Cusum	Runs	LRuns	Rank
1	.107	.014△	.522	.516	.070	.410	.078
2	.377	.313	.958	.764	.925	.914	.391
3	.107	.637	.008△	.359	.473	.052	.456
4	.354	.853	.963	.957	.746	.991	.064
5	.831	.035△	.697	.327	.276	.639	.295
6	.132	.740	.155	.800	.254	.083	.254
7	.627	.473	.074	.187	.123	.108	.354
8	.217	.059	.272	.085	.853	.154	.011△
9	.187	.697	.841	.681	.077	.367	.248
10	.260	.838	.813	.508	.855	.987	.861
11	.987	.980	.939	.520	.829	.469	.670
12	.437	.188	.871	.777	.589	.303	.021△
13	.957	.880	.394	.699	.851	.836	.464
14	.959	.331	.899	.204	.471	.500	.502

15	.119	.550	.804	.483	.377	.248	.119
16	.201	.610	.986	.336	.610	.817	.528
Ave.	*.429*	*.512*	*.637*	*.513*	*.524*	*.492*	*.332*

#	**DFFT**	**Ovlap**	**Univ**	**Apen**	**Ser**	**Ser**	**LinComp**
1	.011△	.120	.283	.952	.492	.014△	.579
2	.078	.257	.947	.303	.481	.818	.985
3	.589	.978	.030△	.209	.643	.273	.341
4	.677	.036△	.083	.333	.891	.144	.214
5	.115	.785	.760	.437	.421	.052	.708
6	.579	.216	.641	.625	.760	.073	.581
7	.864	.057	.730	.052	.423	.398	.919
8	.206	.892	.103	.023△	.261	.069	.156
9	.668	.826	.278	.481	.596	.218	.577
10	.001△	.039△	.041△	.011△	.344	.764	.914
11	.864	.528	.014△	.445	.861	.183	.734
12	.303	.160	.687	.789	.620	.194	.245
13	.561	.133	.406	.447	.088	.372	.817
14	.185	.208	.979	.826	.625	.858	.357
15	.331	.005△	.720	.720	.350	.065	.687
16	.656	.972	.448	.591	.693	.637	.183
Ave.	*.418*	*.388*	*.447*	*.453*	*.534*	*.321*	*.562*

検定値が 0.05 以下の所には△を付してある。△の出現比率は、14/224＝0.0625 であり、0.05 と比して特に問題となる値ではない。また、平均値 Ave にも特に問題点がなく、SSIK は十分な乱数性を有しているものと考えられる。

SSIX に対する NIST 乱数検定プログラムの結果（p 値の χ^2 検定値）

#	**Freq**	**BFreq**	**Cusum**	**Cusum**	**Runs**	**LRuns**	**Rank**
1	.936	.369	.978	.868	.375	.675	.591
2	.011△	.886	.452	.291	.417	.421	.877
3	.101	.058	.650	.408	.430	.664	.528
4	.126	.538	.516	.811	.290	.127	.654
5	.561	.722	.464	.015△	.178	.973	.718
6	.355	.585	.093	.285	.410	.298	.195
7	.094	.049△	.045△	.127	.942	.239	.113
8	.306	.447	.335	.233	.939	.880	.313
9	.705	.699	.465	.412	.136	.983	.421
10	.374	.137	.853	.606	.770	.316	.230
11	.285	.589	.224	.876	.394	.853	.010△
12	.043△	.260	.377	.325	.188	.309	.851
13	.648	.804	.858	.544	.538	.986	.123
14	.901	.038△	.880	.983	.591	.274	.981
15	.253	.941	.257	.561	.450	.771	.559
16	.979	.508	.573	.498	.072	.331	.687
Ave.	*.417*	*.477*	*.501*	*.490*	*.445*	*.569*	*.491*

#	DFFT	Ovlap	Univ	Apen	Ser	Ser	LinComp
1	.677	.047△	.414	.840	.283	.813	.044△
2	.522	.157	.752	.306	.670	.730	.043△
3	.206	.012△	.203	.602	.044△	.260	.746
4	.546	.233	.565	.885	.974	.080	.548
5	.109	.672	.035△	.956	.899	.710	.606
6	.001△	.754	.679	.233	.118	.117	.977
7	.833	.297	.315	.836	.125	.269	.226
8	.325	.462	.015△	.369	.899	.225	.600
9	.076	.414	.026△	.951	.724	.798	.891
10	.120	.689	.403	.951	.695	.811	.746
11	.024△	.339	.157	.049△	.110	.217	.866
12	.181	.212	.039△	.462	.403	.074	.608
13	.181	.389	.126	.650	.432	.773	.489
14	.540	.596	.548	.357	.234	.159	.335
15	.691	.155	.947	.042△	.559	.251	.585
16	.045△	.176	.710	.679	.981	.254	.936
Ave.	*.317*	*.350*	*.371*	*.573*	*.509*	*.409*	*.578*

△の出現比率は 21/224＝0.0938 で、SSIK と比して若干悪いようであるが、特に問題とする値でもない。平均値 Ave にも特に問題となる値はなく、SSIX は、SSIK に比して特性が若干悪いが、十分な乱数性を有しているものと考えられる。

NIST の検定のための 1 ギガビットの乱数生成に要する時間の比は、

SSRexK：SSRexX：SSIK：SSIX＝1：0.81：0.65：0.56

であり、整数演算化により生成スピードがかなり速くなっていることが分かる。

以上をまとめると次のようになる。(1) SSI は 64 ビット整数の乗算とシフトにより、乱数を非再帰的に生成する。(2) 乱数特性を改良した SSIK、SSIX においては、特性については SSIK が、乱数周期・生成速度については SSIX が若干勝っている。

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：実数シフト乱数生成法の乱数特性の改良について. 三重大学教育学部研究紀要　第 57 巻（2006）（自然科学）33-39.