三重大学教育学部研究紀要　第57巻　自然科学　(2006)　33 － 39頁

実数シフト乱数生成法の乱数特性の改良について

谷　口　礼　偉*

Improvement of the Randomness of the Shift-Real Random Number Generator

Hirotake YAGUCHI

要　旨

非代数的な擬似乱数生成法である実数シフト法の乱数特性の改良についてはいくつかの提案がなされている。本論文ではそれらの効果について、NISTの統計的検定による検証を行い、その結果に基づき、新たな改良法を提案する。

単純実数シフト法とその乱数特性の改良法の概要

非代数的かつ非再帰的に10進8桁の擬似乱数を生成する「拡張単純実数シフト法」は、以下の「拡張SSR計算（SSRex)」を基本としている。

［拡張SSR計算（SSRex)］

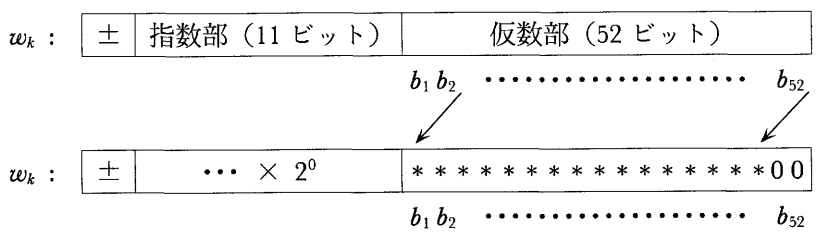
$$h(x) = \underbrace{w_0 \cdot x \cdot x \cdot x \cdot \cdots \cdot x}_{26} \cdot x, \quad x \in [1,\ 2)$$

を

$$w_0 = 1,$$
$$w_k := w_{k-1} \times x, \quad k = 1,\ 2, \cdots,\ 26,$$

と倍精度計算する。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

i) 仮数部の全てのビット値を2ビット左にシフトし（⇒右端には0を2ビット分入れる)、



ii) 指数部は　・・・ × 2^0 となるように設定する（⇒$1 \leq w_k < 2$ となる)。

*三重大学教育学部数学教室

w_{k-1} から w_k を得る過程を Φ_x とすると、$\Phi_x : [1,2) \to [1,2)$ であり、$w_k = \Phi_x^k(w_0)$ と表される。実際の乱数値を得るには、x を変化させ、対応する計算結果 $w_{26} = \Phi_x^{26}(w_0)$ から、0. を含めた最初の5桁を捨て、続く8桁を乱数として採用する。(x を変化させる具体的な方法については、[1] を参照。)

拡張SSR計算が生成する乱数は、多量の乱数が使われる場合には、乱数分布の非対称性 ([2]) が浮かび上がってくることがあるので、「K改良」ならびに「Y改良」の改良法が考えられてきた。

【K改良 SSRex 計算】 ([2]) $h(x)$ を2つの初期値

$w_0 = 1.2718281828459\ (= 1.e)$ $\quad$ $w_0 = 1.3141592653589\ (= 1.\pi)$

のそれぞれの場合について、今までどおり SSRex 計算し $h_e = \Phi_x^{26}(1.e)$, $h_\pi = \Phi_x^{26}(1.\pi)$ を得る。そして $h_e - h_\pi \pmod 1$ を SSRex 計算値の代りとする。

この改良法は、写像 $\Phi_x : w_{k-1} \mapsto w_k$ が、ほとんどの x について、カオス写像となり、k が大きくなると、x を変化させた場合、$\Phi_x^k(1.e)$ と $\Phi_x^k(1.\pi)$ はほとんど独立に振る舞うようになることを利用したものである。(k が小さいと、$\Phi_x^k(1.e)$ と $\Phi_x^k(1.\pi)$ の間に、$1.e$ と $1.\pi$ の大小関係に由来する相関が残るので注意が必要である。$k=26$ であれば、特に問題はないと考えられる。) K改良SSRex 計算で得られる値 $h_e - h_\pi \pmod 1$ の分布は、SSRex 計算で得られる値の分布から畳み込みで得られるので、分布の対称性とともに、一様性の格段の向上が得られる。

「Y改良」は、以下の「Ya 改良 SSRex 計算」と「Yb 改良 SSRex 計算」をミックスして使うものである。

【Ya 改良 SSRex 計算】 $0 < \alpha < 0.5$ とし、$h(x)$ の SSRex 計算値 $\Phi_x^{26}(1)$ を $\tilde{h}$ とする。

このとき、

(i) 「$\tilde{h} < 1+\alpha$ または $\tilde{h} \geq 2-\alpha$」であって $b_{19} = b_{25}$ のとき、

もしくは

(ii) 「$\tilde{h} \geq 1+\alpha$ かつ $\tilde{h} < 2-\alpha$」であって $b_{19} \neq b_{25}$ のとき、

$\tilde{h}$ の仮数部の全ビット値を反転する。

【Yb 改良 SSRex 計算】

「Ya 改良 SSRex 計算」における、

(i) の ... $b_{19} = b_{25}$ ... と (ii) の ... $b_{19} \neq b_{25}$ ...

を入れ替えて計算する。

そして、これらを1つにまとめて、

【Y 改良 SSRex 計算】 $b_{10} = 0$ なら「Ya 改良 SSRex 計算」を適用し、$b_6 = 1$ なら「Yb 改良 SSRex 計算」を適用する。

$\tilde{h}$ に「Y 改良」を適用すると、理想的には、ランダムに半数の $\tilde{h}$ に対して仮数部の全ビットが反転されることになるので、乱数値の分布の対称性の改良が期待される。

これら2つの改良法に対して、次節で NIST の乱数検定プログラムによる統計的検定を行う。

NISTのプログラムによる乱数性の検定

米国の National Insitute of Standards and Technology (NIST) より提供されている乱数検定プログラムは、広く知られており、各種の統計的検定から構成されている。その内容については、ホームページ http://csrc.nist.gov/rng/ に詳細があるので、ここでは深くは立ち入らない。NISTの検定プログラム Version 1.8 は、

frequency,	block-frequency,
cumulative-sums,	runs,
longest-run,	rank,
fft,	nonperiodic-templates,
universal,	approximate entropy
random-excursions,	random-excursions-variant,
serial,	linear-complexity

の各テストで構成されている。

K改良 SSRex 乱数 (SSRexK) に対する NIST の検定は、SSRexK 計算値の仮数部 $b_{10}\,b_{11}\cdots b_{41}$ (32ビット、10進で約9桁強相当) を使って 1000K ビットからなるファイルを 16 個作り、各ファイルに対して、バージョン Sts-1.8 にあるすべてのテストを (計16回) 適用した。1回のテストでは、p 値の一様分布性に関する χ^2 検定値、および、p 値が 0.01 以上であるものの割合が、finalAnalysisReport として出力される。検定時のパラメータは次の通りである：

block-frequency	block length = 20000
nonperiodic-templates	template length = 9
overlapping-templates	template length = 9
universal	block length = 7
	number of initialization = 1280
approximate entropy	block length = 10
serial	block length = 16
linear-complexity	block length = 500
number of bit streams	1000
length of bit	1000000

以下に、p 値の一様分布性に対する χ^2 検定の結果を記しておく。(註：Nonper, RndExc, RndExcV の各検定結果については、データ数が多量のため省略した。)

SSRexK 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	.434	.641	.095	.236	.691	.679	.462
2	.306	.752	.970	.811	.225	.664	.618
3	.924	.906	.954	.098	.230	.130	.473
4	.571	.877	.891	.623	.079	.155	.789

5	.434	.641	.095	.236	.691	.679	.462
6	.140	.161	.498	.244	.817	.078	.247
7	.689	.724	.126	.290	.546	.901	.398
8	.450	.155	.981	.359	.338	.616	.065
9	.184	.623	.251	.209	.802	.097	.807
10	.583	.804	.336	.587	.023Δ	.917	.822
11	.577	.430	.824	.256	.408	.297	.822
12	.829	.625	.919	.585	.367	.230	.222
13	.718	.198	.911	.953	.764	.164	.750
14	.100	.257	.341	.500	.379	.370	.004Δ
15	.836	.370	.025Δ	.426	.201	.131	.232
16	.133	.192	.929	.985	.154	.228	.108
Ave.	*.494*	*.522*	*.572*	*.462*	*.420*	*.396*	*.455*

#	**DFFT**	**Ovlap**	**Univ**	**Apen**	**Ser**	**Ser**	**LinComp**
1	.392	.806	.214	.349	.948	.070	.291
2	.294	.620	.933	.064	.675	.254	.732
3	.494	.312	.914	.924	.534	.182	.976
4	.287	.085	.244	.274	.850	.315	.826
5	.392	.806	.214	.349	.948	.070	.291
6	.864	.979	.396	.827	.848	.406	.015Δ
7	.040Δ	.090	.231	.206	.027Δ	.779	.252
8	.943	.010Δ	.132	.610	.423	.467	.724
9	.434	.136	.598	.623	.494	.871	.327
10	.026Δ	.949	.516	.145	.075	.379	.331
11	.197	.006Δ	.338	.110	.633	.687	.635
12	.124	.203	.836	.783	.417	.919	.054
13	.907	.794	.285	.718	.585	.043Δ	.728
14	.883	.272	.941	.485	.441	.238	.581
15	.941	.261	.898	.252	.891	.598	.226
16	.777	.113	.530	.504	.762	.009Δ	.924
Ave.	*.500*	*.403*	*.514*	*.451*	*.597*	*.393*	*.495*

検定値が、0.05 以下の所にはΔを付した。Δは、0.05 の確率で発生すると考えられるから、上表中の計 11 個のΔは、総データ数 14×16＝224 に 0.05 を掛けた値 11.2 に近く、妥当なものである。この結果から、SSRexK は、十分な乱数性を有することが分かる。

同様の検定をＹ改良 SSRex 乱数（SSRexY）に対して行うと、approximate entropy 検定で棄却されることが分かる。

SSRexY 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	.679	.787	.434	.746	.956	.423	.216
2	.448	.951	.775	.504	.591	.394	.552
3	.734	.208	.538	.469	.298	.081	.504
4	.362	.469	.045Δ	.342	.483	.471	.573

#	DFFT	Ovlap	Univ	Apen	Ser	Ser	LinComp
1	.108	.200	.192	.000Δ	.056	.656	.618
2	.793	.349	.880	.000Δ	.082	.691	.232
3	.405	.012Δ	.032Δ	.000Δ	.072	.620	.770
4	.180	.864	.313	.000Δ	.498	.864	.681

分布の対称性に対する工夫が仇になったようである。「Y 改良」のこの欠点を修正するため、次節において「X 改良 SSRex 計算」を導入し、NIST の検定を行うことにする。

X改良実数シフト乱数生成法

新たな改良法「X 改良」は、「K 改良」と「Y 改良」の中間的な性格を持っている。すなわち、「Y 改良」では特定のビットを調べ、ある条件のもとで仮数部の全てのビットを XOR で反転していたが、「X 改良」では、仮数部のビット反転を、別の SSRex 計算で得られた仮数部と XOR をとることにより、ビットごとにランダムに行おうというものである。

【X改良 SSRex 計算】 $h(x)$ を初期値 $w_0 = 1.2718281828459$（ $=$ $1.e$）について、今までどおり SSRex 計算し $h_e = \Phi_x^{26}(1.e)$ を得る。次に、

$$g(x) = w_0 \cdot \underbrace{x \cdot x \cdot \cdots \cdot x}_{7}, \quad x \in [1,\ 2)$$

を初期値 $w_0 =$ 1.3141592653589（$=1.\pi$）について SSRex 計算を行い、値 g_p を得る。ただし、g_p の SSRex 計算においては、仮数部の全ビットの 1 回のシフト量は 8 ビットとする、すなわち、g_p の SSRex 計算は次のようになる：

i) 仮数部の全てのビット値を 8 ビット左にシフトし、
ii) 指数部は $\cdots \times 2^0$ となるように設定する。

最後に、h_e と g_p の仮数部の XOR をとって、X改良 SSRex 計算値とする。

g_p の計算において、仮数部のシフト回数の 7 は、回数をいろいろ変えて調べて良さそうなものの中から選ばれた。 結果的に、「X 改良」は「K 改良」より仮数部のシフト回数が少なくなり、乱数の生成速度が若干速くなるという副次的な効果が発生する。「X 改良」によって、分布の対称性が改良される

という理論的な根拠は記述しにくいが、「Y改良」に比して、仮数部の各ビットごとにランダムにXORがとられるので、対称性は増すものと考えられる。以下にX改良SSRex乱数（SSRexX）に対するNISTの検定結果を記しておく。

SSRexX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	.372	.441	.500	.720	.471	.693	.369
2	.941	.462	.705	.036Δ	.869	.583	.076
3	.005Δ	.974	.477	.087	.845	.811	.288
4	.270	.818	.154	.335	.633	.082	.681
5	.319	.777	.843	.269	.506	.135	.339
6	.593	.325	.437	.955	.758	.994	.147
7	.668	.909	.162	.979	.589	.804	.766
8	.023Δ	.236	.084	.040Δ	.542	.252	.055
9	.991	.641	.261	.945	.944	.424	.256
10	.038Δ	.186	.313	.277	.858	.006Δ	.689
11	.573	.976	.768	.266	.175	.530	.290
12	.512	.426	.777	.448	.643	.242	.856
13	.750	.133	.937	.789	.542	.164	.260
14	.758	.120	.184	.581	.986	.336	.981
15	.520	.977	.726	.274	.168	.168	.736
16	.280	.426	.879	.911	.722	.036Δ	.032Δ
Ave.	*.476*	*.552*	*.513*	*.495*	*.641*	*.391*	*.426*

#	DFFT	Ovlap	Univ	Apen	Ser	Ser	LinComp
1	.689	.534	.893	.851	.184	.766	.266
2	.432	.756	.065	.239	.454	.019Δ	.016Δ
3	.266	.193	.084	.207	.882	.007Δ	.754
4	.013Δ	.342	.241	.084	.124	.785	.858
5	.635	.822	.408	.387	.272	.283	.101
6	.111	.536	.149	.101	.561	.522	.191
7	.226	.135	.932	.997	.398	.100	.502
8	.691	.221	.885	.915	.589	.997	.274
9	.217	.177	.186	.648	.970	.841	.462
10	.349	.322	.073	.487	.650	.687	.288
11	.369	.677	.175	.082	.133	.522	.309
12	.490	.892	.068	.552	.124	.526	.071
13	.479	.315	.948	.265	.526	.102	.879
14	.750	.198	.073	.538	.966	.083	.877
15	.412	.860	.372	.322	.604	.338	.419
16	.807	.883	.066	.083	.639	.868	.190
Ave.	*.434*	*.491*	*.351*	*.422*	*.505*	*.465*	*.404*

以上のように、「X 改良」では「Y 改良」の欠点が改良されている。また、印Δのついた箇所は計 12 カ所であり、これも妥当な線である。

「K 改良」と「X 改良」の乱数生成速度の違いであるが、NISTの検定を行うために 1000K ビットからなるファイルを作製するのに要する時間は、「K 改良」では 96 秒であり、「X 改良」では 78 秒である。「X 改良」では乱数生成速度が 2 割弱速くなっていることが分かる。

参考文献

[1] 谷口礼偉：数値計算誤差と乱数生成. 神戸大学理学部数学教室, 2001.

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