擬似乱数生成系の検定方法に関する調査
調査報告書

２００４年１月

京都大学

廣瀬 勝一

擬似乱数生成系の検定方法に関する調査
調査報告書

京都大学情報学研究科
廣瀬勝一

2004 年 1 月

目 次

第1章　まえがき

与えられた系列が良い乱数列であるかどうかを判定するための手法として，従来より，その系列の統計的性質を利用した検定法が多数提案されている．したがって，統計的手法による乱数検定法に関する文献やそれらを利用した検定ツールも多く存在するが，どの検定法が採用あるいは推奨されているかはそれぞれ異なっており，明確な判定基準はない．

このような現状において，昨年度,「擬似乱数検証ツールの調査開発」が行われ，その調査報告書が，情報処理振興事業協会から発行されている [6]．この調査報告では，まず，従来提案されてきた統計的手法に基づく乱数検定法が分類・整理され，さらに，乱数検定のために必要最小限の検定法の組が選出されている．この検定法の組はミニマムセットと呼ばれている．なお，その調査開発は，NIST Special Publication 800-22 [1] と DIEHARD [10] の二つの乱数検定ツールで採用されている検定法を主な対象として行われた．

上で述べた「擬似乱数検証ツールの調査開発」では，検定法の分類・整理，ミニマムセットの導出にあたり，検定法の解析および計算機実験が行われ，それによって，幾つかの問題点が明らかにされた．本報告書では，それらの問題点に関する調査結果を報告する．

本報告書の構成は以下の通りである．2 章では，昨年度行われた擬似乱数検証ツールの調査開発の概要とそこで明らかになった問題点を挙げる．3 章では，NIST SP800-22 の各検定法により，擬似乱数列を検定した結果について述べる．4 章では，NIST SP800-22 の離散フーリエ変換検定の誤りについて述べ，修正した検定法による擬似乱数列の検定結果を述べる．5 章では，linear truncated congruential generator の未知パラメータを求めるアルゴリズムに関する調査結果を述べる．

第2章　擬似乱数検証ツールの調査開発の概要

本章では，昨年度行われた「擬似乱数検証ツールの調査開発」[6] の概要を述べると共に，そこで明らかとなった問題点を挙げる．

2.1　概要

「擬似乱数検証ツールの調査開発」では，これまでに知られている乱数検定法が調査され，それらが幾つかの観点から分類されている．その調査開発では，$\{0,1\}$ からなる 2 値系列の検定法が対象とされており，主に，NIST Special Publication 800-22 (NIST SP800-22) [1] と DIEHARD [10] で用意されている検定法を中心として調査開発が進められた．NIST SP800-22 と DIEHARD で用意されている検定法をそれぞれ図 2.1，図 2.2 に示す．

「擬似乱数検証ツールの調査開発」では，さらに，乱数を検定する上で必要最小限の検定法の組（ミニマムセット）が定められた．ミニマムセットは以下の通りである．

- ブロック単位の頻度に関する検定
  - 高次元度数検定
  - ブロック単位の度数検定
  - ブロック単位の最長連検定
  - 8 ビット中の文字数検定
  - 特定位置の 8 ビット中の文字数検定
- パタンの出現に関する検定
  - ビット列検定
  - OPSO 検定
  - OQSO 検定
  - バースデイ空間検定
- 状態遷移・ランダムウォークに関する検定
  - 累積和検定
- 一様性・圧縮可能性に関する検定
  - 系列検定
- 周期性に関する検定
  - 線形複雑度検定

1. 1 次元度数検定 (frequency (monobit) test)
2. ブロック単位の頻度検定 (frequency test within a block)
3. 連の検定 (runs test)
4. ブロック単位の最長連検定 (test for the longest run of ones in a block)
5. 2 値行列ランク検定 (binary matrix rank test)
6. 離散フーリエ変換検定 (discrete Fourier transform (spectral) test)
7. 重なりのないテンプレート適合検定 (non-overlapping template matching test)
8. 重なりのあるテンプレート適合検定 (overlapping template matching test)
9. Maurer のユニバーサル統計検定 (Maurer's universal statistical test)
10. Lempel-Ziv 圧縮検定 (Lempel-Ziv compression test)
11. 線形複雑度検定 (linear complexity test)
12. 系列検定 (serial test)
13. 近似エントロピー検定 (approximate entropy test)
14. 累積和検定 (cumulative sums (cusum) test)
15. ランダム偏差検定 (random excursions test)
16. 種々のランダム偏差検定 (random excursions variant test)

図 2.1: NIST SP800-22 の検定法

1. バースデー空間検定 (birthday spacings test)
2. OPERM5 検定 (overlapping 5-permutation test)
3. 31×31 の 2 値行列ランク検定 (binary rank test for 31×31 matrices)
4. 32×32 の 2 値行列ランク検定 (binary rank test for 32×32 matrices)
5. 6×8 の 2 値行列ランク検定 (binary rank test for 6×8 matrices)
6. ビット列検定 (bitstream test)
7. OPSO 検定 (overlapping-pairs-space-occupancy test)
8. OQSO 検定 (overlapping-quadruples-space-occupancy test)
9. DNA 検定 (DNA test)
10. 8 ビット中の文字数検定 (count-the-1's test on a stream of bytes)
11. 特定位置の 8 ビット中の文字数検定 (count-the-1's test for specific bytes)
12. 駐車場検定 (parking lot test)
13. 最小距離検定 (minimum distance test)
14. 3DSPHERES 検定 (3d-spheres test)
15. スクィーズ検定 (squeeze test)
16. 重なりのある和検定 (overlapping sums test)
17. 連の検定 (runs test)
18. クラップス検定 (craps test)

図 2.2: DIEHARD の検定法

1. G Using SHA-1
2. Linear Congruential
3. Blum-Blum-Shub
4. Micali-Schnorr
5. Modular Exponentiation
6. Quadratic Congruential I
7. Quadratic Congruential II
8. Cubic Congruential
9. XOR
10. ANSI X9.17 (3-DES)
11. G Using DES

図 2.3: NIST SP800-22 の擬似乱数生成器

ミニマムセットの内，高次元度数検定は「擬似乱数検証ツールの調査開発」で新たに加えられた検定法である．なお，このミニマムセットを導出するにあたり，各検定法は，NIST SP800-22 および DIEHARD で用意されている擬似乱数生成器の出力系列に対する検定結果に基づいて評価および取捨選択された．NIST SP800-22 と DIEHARD で用意されている擬似乱数生成器をそれぞれ図 2.3，図 2.4 に示す．

2.2 問題点

「擬似乱数検証ツールの調査開発」の報告書では，以下の三つの問題点が挙げられていた．

- NIST SP800-22 の離散フーリエ変換検定では，系列長が 10 万ビットの場合，NIST SP800-22 で用意されている 16 個すべての乱数生成器が一様性の検定に合格しない．
- NIST SP800-22 の Lempel-Ziv 圧縮検定で用いられているパラメータに理論的根拠が不明瞭でかつ不適切と考えられるものがある．
- 線形合同法に基づく方式で生成された擬似乱数列は，暗号への応用には不適切であると考えられているが，NIST SP800-22 の線形合同法に基づく生成器によって生成された系列は，調査対象としたすべての検定で，良い乱数であると判定される．

なお，ミニマムセットの導出のための計算機実験においては，各擬似乱数生成器について，10 万ビットの系列については最大 500 本，100 万ビットの系列については 250 本が生成され，それらを用いて各検定法の評価が行われたことが報告されている．これについては，さらに多くの本数の系列を用いた評価も必要であると考えられる．

1. A multiply-with-carry (MWC) generator
2. A MWC generator on pairs of 16 bits
3. The “Mother of all random number generators”
4. The KISS generator
5. The simple but very good generator COMBO
6. The lagged Fibonacci-MWC combination ULTRA
7. A combination MWC/subtract-with-borrow (SWB) generator
8. An extended congruential generator
9. The super-duper generator
10. A subtract-with-borrow generator
11. Any specified congruential generator
12. The 31-bit generator ran2 from Numerical Recipes
13. Any specified shift-register generator
14. The system generator in Microsoft Fortran
15. Any lagged-Fibonacci generator
16. An inverse congruential generator

図 2.4: DIEHARD の擬似乱数生成器

第3章　NIST Special Publication 800-22

本章ではまず，NIST SP800-22 の検定法について述べたのち，次に，それらの検定法について行った計算機実験とその結果について述べる．

3.1　乱数列の検定法

本節では，NIST SP800-22 で用意されている 16 個の検定法を記す．これらすべての検定法が対象としている系列は $\{0,1\}$ からなる 2 値系列である．それぞれの検定法では，標準正規分布または χ^2 分布に基づいて検定が行われ，p-value と呼ばれる値が出力される．入力系列は，p-value ≥ 0.01 のとき，良い乱数列であると判定され，p-value < 0.01 のとき，良い乱数列でないと判定される．

p-value は，真の乱数生成器が，検定対象として入力された系列よりもランダムでない系列を生成する確率と解釈できる．標準正規分布に基づいて検定が行われる場合，p-value は，以下の関数 *erfc* を用いて計算される．

$$erfc(z) = \int_z^\infty \frac{2}{\sqrt{\pi}} \mathrm{e}^{-x^2} \mathrm{d}x$$

また，χ^2 分布に基づいて検定が行われる場合，p-value は以下の関数 **igamc** を用いて計算される．

$$\mathbf{igamc}(a,z) = \frac{1}{\Gamma(a)} \int_z^\infty \mathrm{e}^{-t} t^{a-1} \mathrm{d}t$$

なお，

$$\Gamma(a) = \int_0^\infty \mathrm{e}^{-t} t^{a-1} \mathrm{d}t$$

である．

3.1.1　1 次元度数検定

入出力

入力	x	乱数列 $x = (x_1, x_2, \ldots, x_n)$
	n	乱数列の長さ
出力	p-value	正規分布統計量からの p-value

アルゴリズム

1. 0 と 1 からなる長さ n の入力系列 $x = (x_1, x_2, \ldots, x_n)$ を，次のように -1 と 1 からなる系列 $X = (X_1, X_2, \ldots, X_n)$ へ変換する．

$$X_i = 2x_i - 1 \quad (1 \leq i \leq n)$$

次に，$S_n = X_1 + X_2 + \cdots + X_n$ を計算する．

2. $s_{obs} = |S_n|/\sqrt{n}$ を計算する．

3. p-value $= erfc(s_{obs}/\sqrt{2})$ を計算する．

3.1.2 ブロック単位の頻度検定

入出力

入力	x	乱数列 $x = (x_1, x_2, \ldots, x_n)$
	n	乱数列の長さ
	M	各ブロックの長さ
出力	p-value	χ^2 分布統計量からの p-value

アルゴリズム

1. 入力系列 x を重なりのない $N = \lfloor n/M \rfloor$ 個の長さ M のブロックに分割する．使用しないビットは捨てる．

2. $1 \leq i \leq N$ について，i 番目のブロックの 1 の割合 π_i を計算する．

$$\pi_i = \frac{\sum_{j=1}^{M} x_{(i-1)M+j}}{M}$$

3. $\chi^2(obs) = 4M \sum_{i=1}^{N} (\pi_i - 1/2)^2$ を計算する．

4. p-value $= \mathbf{igamc}(N/2, \chi^2(obs)/2)$ を計算する．

3.1.3 連の検定

入出力

入力	x	乱数列 $x = (x_1, x_2, \ldots, x_n)$
	n	乱数列の長さ
出力	p-value	正規分布統計量からの p-value

アルゴリズム

1. 入力系列 x の 1 の頻度 $\pi = \dfrac{\sum_{i=1}^{n} x_i}{n}$ を計算する．

2. $|\pi - 1/2| \geq 2/\sqrt{n}$ であれば，この検定は行う必要がないと判断して停止する．このとき，p-value $= 0$ とする．

3. $V_n(obs) = \sum_{k=1}^{n-1} r(k) + 1$ を計算する．ただし，

$$r(k) = \begin{cases} 0 & x_k = x_{k+1} \text{ のとき} \\ 1 & x_k \neq x_{k+1} \text{ のとき} \end{cases}$$

である．

4. p-value $= erfc\left(\dfrac{|V_n(obs) - 2n\pi(1-\pi)|}{2\sqrt{2n}\pi(1-\pi)}\right)$ を計算する．

3.1.4 ブロック単位の最長連検定

入出力

入力	x	乱数列 $x = (x_1, x_2, \ldots, x_n)$
	n	乱数列の長さ
出力	p-value	χ^2 分布統計量からの p-value

アルゴリズム

1. 入力系列 x を重なりのない $N = \lfloor n/M \rfloor$ 個の長さ M のブロックに分割する．但し，n の大きさに応じて M は以下のように定められる．

$$M = \begin{cases} 8 & 128 \le n < 6272 \text{ のとき} \\ 128 & 6272 \le n < 750000 \text{ のとき} \\ 10^4 & 750000 \le n \text{ のとき} \end{cases}$$

2. 各ブロックをその中の 1 の最長連の長さによって幾つかのカテゴリに分類し，それぞれのカテゴリに属するブロックの個数を求める．カテゴリの個数は，$M = 8, 128, 10^4$ のとき，それぞれ，4，6，7 である．各カテゴリは以下の表の通りである．表の各項目は最長連の長さを表す．カテゴリ i に属するブロックの個数を v_i と記す．

カテゴリ	M の値		
	8	128	10^4
0	≤ 1	≤ 4	≤ 10
1	2	5	11
2	3	6	12
3	≥ 4	7	13
4	—	8	14
5	—	≥ 9	15
6	—	—	≥ 16

3. $\chi^2(obs) = \displaystyle\sum_{i=0}^{K} \frac{(v_i - N\pi_i)^2}{N\pi_i}$ を計算する．K，N の値および各 π_i の値は以下の通りである．

$$K = \begin{cases} 3 & M = 8 \text{ のとき} \\ 5 & M = 128 \text{ のとき} \\ 6 & M = 10^4 \text{ のとき} \end{cases}$$

$$N = \begin{cases} 16 & M = 8 \text{ のとき} \\ 49 & M = 128 \text{ のとき} \\ 75 & M = 10^4 \text{ のとき} \end{cases}$$

i	M の値		
	8	128	10^4
0	0.2148	0.1174	0.0882
1	0.3672	0.2430	0.2092
2	0.2305	0.2493	0.2483
3	0.1875	0.1752	0.1933
4	—	0.1027	0.1208
5	—	0.1124	0.0675
6	—	—	0.0727

4. p-value $= \mathbf{igamc}(K/2, \chi^2(obs)/2)$ を計算する．

3.1.5 2 値行列ランク検定

入出力

入力	x	乱数列 $x = (x_1, x_2, \ldots, x_n)$
	n	乱数列の長さ
	M	各行列の行数．$M = 32$ が用いられている
	Q	各行列の列数．$Q = 32$ が用いられている
出力	p-value	χ^2 分布統計量からの p-value

アルゴリズム

1. 入力系列 x を重なりのない $N = \lfloor n/(MQ) \rfloor$ 個のブロックに分割する．長さ MQ の各ブロックから，$M \times Q$ の 2 値行列を作る．連続する Q ビットの系列を先頭から順番に行列の各行とする．

2. 各 $1 \leq \ell \leq N$ について，ランク $R(\ell)$ を計算する．

3. F_M，F_{M-1} を以下のように定義する．

$$\begin{aligned} F_M &= R(\ell) = M \text{ となる行列の個数} \\ F_{M-1} &= R(\ell) = M - 1 \text{ となる行列の個数} \end{aligned}$$

4. $\chi^2(obs) = \dfrac{(F_M - 0.2888N)^2}{0.2888N} + \dfrac{(F_{M-1} - 0.5776N)^2}{0.5776N} + \dfrac{(N - F_M - F_{M-1} - 0.1336N)^2}{0.1336N}$ を計算する．

5. p-value $= \mathrm{e}^{-\chi^2(obs)/2}$ を計算する．

3.1.6 離散フーリエ変換検定

入出力

入力	x	乱数列 $x = (x_1, x_2, \ldots, x_n)$
	n	乱数列の長さ
出力	p-value	正規分布統計量からの p-value

アルゴリズム

1. 入力系列 x を，次のように -1 と 1 からなる系列 $X = (X_1, X_2, \ldots, X_n)$ へ変換する．

$$X_i = 2x_i - 1 \quad (1 \leq i \leq n)$$

2. 系列 X を離散フーリエ変換し，$S = DFT(X)$ を得る．

3. S' を S の最初の $n/2$ 個の要素からなる部分列とし，$M = modulus(S') = |S'|$ を求める．なお，$modulus$ はピークの高さの系列を求める関数である．

4. 95%のピークの高さのしきい値である $T = \sqrt{3n}$ を求める．

5. $N_0 = 0.95n/2$ を求める．N_0 は，X が真の乱数列であるときに，ピークの高さの内で T を越えない個数の理論値である．

6. ピークの高さの内で実際に T を越えなかった個数 N_1 を求める．

7. $d = \dfrac{N_1 - N_0}{\sqrt{n(0.95)(0.05)/2}}$ を求める．

8. p-value $= erfc(|d|/\sqrt{2})$ を計算する．

3.1.7 重なりのないテンプレート適合検定

入出力

入力	x	乱数列 $x = (x_1, x_2, \ldots, x_n)$
	n	乱数列の長さ
	m	テンプレートの長さ
	B	m ビットのテンプレート
	M	系列 x の部分列の長さ．プログラム中で，$M = 2^{17}$ が与えられている．
	N	ブロックの個数．プログラム中で，$N = 2^3$ が与えられている．
出力	p-value	χ^2 分布統計量からの p-value

アルゴリズム

1. 入力系列 x を重なりのない長さ M の N 個のブロックに分割する．

2. $1 \leq j \leq N$ について，j 番目のブロックにおけるテンプレート B の一致回数を W_j と記す．ブロック上に m ビットの窓を置き，ブロックの先頭から窓の中のビット列とテンプレート B とを照合する．一致しなかった場合は，窓を 1 ビットだけずらす．一致した場合は m ビットずらす．

3. 入力系列が真の乱数列である場合の理論的な平均 μ と分散 σ^2 を計算する．

$$\begin{aligned} \mu &= \frac{M - m + 1}{2^m} \\ \sigma^2 &= M\left(\frac{1}{2^m} - \frac{2m-1}{2^{2m}}\right) \end{aligned}$$

4. $\chi^2(obs) = \displaystyle\sum_{j=1}^{N} \frac{(W_j - \mu)^2}{\sigma^2}$ を計算する．

5. p-value $= \mathbf{igamc}(N/2, \chi^2(obs)/2)$ を計算する．

3.1.8 重なりのあるテンプレート適合検定

入出力

入力	x	乱数列 $x = (x_1, x_2, \ldots, x_n)$
	n	乱数列の長さ
	m	テンプレートの長さ
	B	m ビットのテンプレート
	K	自由度．プログラム中で，$K = 5$ が与えられている．
	M	系列 x の部分列の長さ．プログラム中で，$M = 1032$ が与えられている．
	N	ブロックの個数．プログラム中で，$N = 968$ が与えられている．
出力	p-value	χ^2 分布統計量からの p-value

アルゴリズム

1. 入力系列 x を重なりのない長さ M の N 個のブロックに分割する．

2. 各ブロックにおけるテンプレート B の一致回数を求める．ブロック上に m ビットの窓を置き，ブロックの先頭から窓の中のビット列とテンプレート B とを照合する．テンプレートとの一致・不一致にかかわらず，窓を常に 1 ビットだけずらす．

   テンプレート B との一致回数に応じて，ブロックを 6 個のカテゴリに分類する．なお，$0 \leq i \leq 4$ のとき，カテゴリ i はテンプレートとの一致回数が i 回のブロック，$i = 5$ のとき，カテゴリ i はテンプレートとの一致回数が 5 回以上のブロックに対応する．カテゴリ i に属するブロックの個数を v_i と記す．

3. 入力系列が真の乱数列である場合の $\pi_i = v_i/N$ の理論値を計算するために，

$$\begin{aligned} \lambda &= \frac{M - m + 1}{2^m} \\ \eta &= \lambda/2 \end{aligned}$$

   を計算する．

$$\begin{aligned} \pi_0 &= \mathrm{e}^{-\eta} \\ \pi_1 &= \frac{\eta}{2}\mathrm{e}^{-\eta} \\ \pi_2 &= \frac{\eta \mathrm{e}^{-\eta}}{8}(\eta + 2) \\ \pi_3 &= \frac{\eta \mathrm{e}^{-\eta}}{8}\left(\frac{\eta^2}{6} + \eta + 1\right) \\ \pi_4 &= \frac{\eta \mathrm{e}^{-\eta}}{16}\left(\frac{\eta^3}{24} + \frac{\eta^2}{2} + \frac{3\eta}{2} + 1\right) \end{aligned}$$

   である．

4. $\chi^2(obs) = \sum_{i=0}^{5} \frac{(v_i - N\pi_i)^2}{N\pi_i}$ を計算する．

5. p-value $= \mathbf{igamc}(5/2, \chi^2(obs)/2)$ を計算する．

3.1.9 Maurer のユニバーサル統計検定

入出力

入力	x	乱数列 $x = (x_1, x_2, \ldots, x_n)$
	n	乱数列の長さ
	L	ブロックの長さ
	Q	初期化用系列のブロック数
出力	p-value	正規分布統計量からの p-value

アルゴリズム

1. 入力系列 x を重なりのない $(Q+K)$ 個の L ビットのブロックに分割し，最初の Q 個を初期化用セグメント，後の K 個を検定用セグメントとする．なお，$Q+K = \lfloor n/L \rfloor$ である．

2. 初期化用セグメントを用いて，2^L 個のエントリ T_j $(j = 0, 1, \ldots, 2^L - 1)$ をもつテーブルを作る．

   (a) すべての T_j の初期値は 0 とする．

   (b) 初期化用セグメントについて，$i = 1$ から順に $i = Q$ まで，i 番目のブロックを L ビットの 2 進数と見なしたときの値が j であったとき，$T_j = i$ とする．

   T_j は，j の 2 進数表現を L ビットのビット列と見なしたときに，そのビット列が初期化用セグメントで最後に現れた位置を表す．

3. 検定用セグメントについて以下の処理を行う．

   (a) sum の初期値を 0 とする．

   (b) $i = Q+1$ から順に $i = Q+K$ まで，i 番目のブロックを L ビットの 2 進数と見なしたときの値が j であったとき，

   $$sum = sum + \log(i - T_j)$$

   とし，新たに $T_j = i$ とする．

   $Q+1$ 番目のブロックは，検定用セグメントの先頭ブロックである．

4. $f_n = \dfrac{sum}{K}$ を計算する．

5. p-value $= erfc\left(\left|\frac{f_n - \mu(L)}{\sqrt{2}\sigma(L)}\right|\right)$ を計算する．$\mu(L)$，$\sigma^2(L)$ は以下の表の通りである．

L	$\mu(L)$	$\sigma^2(L)$
6	5.2177052	2.954
7	6.1962507	3.125
8	7.1836656	3.238
9	8.1764248	3.311
10	9.1723243	3.356
11	10.170032	3.384
12	11.168765	3.401
13	12.168070	3.410
14	13.167693	3.416
15	14.167488	3.419
16	15.167379	3.421

3.1.10 Lempel-Ziv 圧縮検定

入出力

入力	x	乱数列 $x = (x_1, x_2, \ldots, x_n)$
	n	乱数列の長さ
出力	p-value	正規分布統計量からの p-value

アルゴリズム

1. 入力系列 x を先頭から順に走査し，それまでに現れなかった部分列が現れたときに，その系列を辞書に登録する．なお，次の部分列の探索は，その前の部分列の末尾の次の位置から始める．W_{obs} を得られた部分列の個数とする．

2. p-value $= \frac{1}{2} erfc\left(\frac{\mu - W_{obs}}{\sqrt{2}\sigma}\right)$ を計算する．$n = 10^6$ のとき，$\mu = 69586.25$，$\sigma^2 = 70.448718$ が用いられている．なお，この値は G using SHA-1 で生成された系列から算出された値である．

3.1.11 線形複雑度検定

入出力

入力	x	乱数列 $x = (x_1, x_2, \ldots, x_n)$
	n	乱数列の長さ
	M	ブロックの長さ．
	K	自由度．プログラム中で $K = 6$ が与えられている．
出力	p-value	χ^2 分布統計量からの p-value

アルゴリズム

1. 入力系列 x を重なりのない長さ M の N 個のブロックに分割する．$n = MN$ である．

2. Berlekamp-Massey のアルゴリズムを用いて，各ブロックの線形複雑度 L_i $(1 \leq i \leq N)$ を求める．

3. 入力系列が真の乱数列である場合の平均の理論値 μ を計算する．

$$\mu = \frac{M}{2} + \frac{9 + (-1)^{M+1}}{36} - \frac{(M/3 + 2/9)}{2^M}$$

4. 各ブロック（$1 \le i \le N$）について，

$$T_i = (-1)^M (L_i - \mu) + \frac{2}{9}$$

を計算する．

5. $\nu_0, \nu_1, \ldots, \nu_6$ を以下の規則にしたがって求める．

ν_0	$T_i \le -2.5$ であるブロックの個数
ν_1	$-2.5 < T_i \le -1.5$ であるブロックの個数
ν_2	$-1.5 < T_i \le -0.5$ であるブロックの個数
ν_3	$-0.5 < T_i \le 0.5$ であるブロックの個数
ν_4	$0.5 < T_i \le 1.5$ であるブロックの個数
ν_5	$1.5 < T_i \le 2.5$ であるブロックの個数
ν_6	$T_i > 2.5$ であるブロックの個数

6. $\chi^2(obs) = \sum_{i=0}^{K} \frac{(\nu_i - N\pi_i)^2}{N\pi_i}$ を計算する．ここで，$\pi_0 = 0.01047$，$\pi_1 = 0.03125$，$\pi_2 = 0.125$，$\pi_3 = 0.5$，$\pi_4 = 0.25$，$\pi_5 = 0.0625$，$\pi_6 = 0.02078$ である．

7. p-value $= \mathbf{igamc}(K/2, \chi^2(obs)/2)$ を計算する．

3.1.12 系列検定

入出力

入力	x	乱数列 $x = (x_1, x_2, \ldots, x_n)$
	n	乱数列の長さ
	m	ブロックの長さ
出力	p-value	χ^2 分布統計量からの p-value（2 個）

アルゴリズム

1. 入力系列 x について，その先頭から $m-1$ ビットを x の末尾に付加し，長さ $n+m-1$ の系列 x' を作る．

2. x' における長さ m ビットのすべてのブロック $i_1 i_2 \cdots i_m$ の出現回数 $\nu_{i_1 i_2 \cdots i_m}$，長さ $m-1$ ビットのすべてのブロック $i_1 i_2 \cdots i_{m-1}$ の出現回数 $\nu_{i_1 i_2 \cdots i_{m-1}}$，長さ $m-2$ ビットのすべてのブロック $i_1 i_2 \cdots i_{m-2}$ の出現回数 $\nu_{i_1 i_2 \cdots i_{m-2}}$ を求める．

3. 以下の値を計算する．

$$
\begin{aligned}
{\Psi_m}^2 &= \frac{2^m}{n} \sum_{i_1 i_2 \cdots i_m} \left(\nu_{i_1 i_2 \cdots i_m} - \frac{n}{2^m} \right)^2 \\
{\Psi_{m-1}}^2 &= \frac{2^{m-1}}{n} \sum_{i_1 i_2 \cdots i_{m-1}} \left(\nu_{i_1 i_2 \cdots i_{m-1}} - \frac{n}{2^{m-1}} \right)^2 \\
{\Psi_{m-2}}^2 &= \frac{2^{m-2}}{n} \sum_{i_1 i_2 \cdots i_{m-2}} \left(\nu_{i_1 i_2 \cdots i_{m-2}} - \frac{n}{2^{m-2}} \right)^2
\end{aligned}
$$

4. 以下の値を計算する．

$$
\begin{aligned}
\nabla {\Psi_m}^2 &= {\Psi_m}^2 - {\Psi_{m-1}}^2 \\
\nabla^2 {\Psi_m}^2 &= {\Psi_m}^2 - 2{\Psi_{m-1}}^2 + {\Psi_{m-2}}^2
\end{aligned}
$$

5. 以下の二つの p-value を計算する．

$$
\begin{aligned}
p\text{-value1} &= \mathbf{igamc}(2^{m-2}, \nabla {\Psi_m}^2) \\
p\text{-value2} &= \mathbf{igamc}(2^{m-3}, \nabla^2 {\Psi_m}^2)
\end{aligned}
$$

3.1.13 近似エントロピー検定

入出力

入力	x	乱数列 $x = (x_1, x_2, \ldots, x_n)$
	n	乱数列の長さ
	m	ブロックの長さ．
出力	p-value	χ^2 分布統計量からの p-value

アルゴリズム

1. 入力系列 x について，その先頭から $m-1$ ビットを x の末尾に付加し，長さ $n+m-1$ の系列 x' を作る．
2. x' における長さ m ビットのすべてのブロック i の出現回数 $\#i$ を求める．なお，i は m ビットの系列を 2 進数と見なしたときの値とする．
3. $C_i^m = \dfrac{\#i}{n}$ を計算する．
4. $\varphi^{(m)} = \displaystyle\sum_{i=0}^{2^m-1} C_i^m \log C_i^m$ を計算する．
5. $m = m+1$ として，上の 1 から 4 を繰り返す．
6. $\chi^2(obs) = 2n(\log 2 - (\varphi^{(m)} - \varphi^{(m+1)}))$ を計算する．
7. p-value $= \mathbf{igamc}(2^{m-1}, \chi^2(obs)/2)$ を計算する．

3.1.14 累積和検定

入出力

入力	x	乱数列 $x = (x_1, x_2, \ldots, x_n)$
	n	乱数列の長さ
	$mode$	検定を系列の先頭から始めるか ($mode = 0$)，末尾から始めるか ($mode = 1$) を表す.
出力	p-value	正規分布統計量からの p-value

アルゴリズム

1. 入力系列 x を，次のように -1 と 1 からなる系列 $X = (X_1, X_2, \ldots, X_n)$ へ変換する.

$$X_i = 2x_i - 1 \quad (1 \leq i \leq n)$$

2. $1 \leq k \leq n$ について，以下の値を計算する

$$S_k = \begin{cases} \displaystyle\sum_{i=1}^{k} X_i & mode = 0 \text{ のとき} \\ \displaystyle\sum_{i=1}^{k} X_{n-i+1} & mode = 1 \text{ のとき} \end{cases}$$

3. $z = \max_{1 \leq k \leq n} |S_k|$ を計算する.

4. p-value を計算する．Φ は標準正規分布の累積分布関数である.

$$\begin{aligned} p\text{-value} \quad = \quad & 1 - \sum_{k=\frac{-n/z+1}{4}}^{\frac{n/z-1}{4}} \left(\Phi\left(\frac{(4k+1)z}{\sqrt{n}} \right) - \Phi\left(\frac{(4k-1)z}{\sqrt{n}} \right) \right) + \\ & \sum_{k=\frac{-n/z-3}{4}}^{\frac{n/z-1}{4}} \left(\Phi\left(\frac{(4k+3)z}{\sqrt{n}} \right) - \Phi\left(\frac{(4k+1)z}{\sqrt{n}} \right) \right) \end{aligned}$$

3.1.15 ランダム偏差検定

入出力

入力	x	乱数列 $x = (x_1, x_2, \ldots, x_n)$
	n	乱数列の長さ
出力	p-value	χ^2 分布統計量からの p-value（8 個）

アルゴリズム

1. 入力系列 x を，次のように -1 と 1 からなる系列 $X = (X_1, X_2, \ldots, X_n)$ へ変換する.

$$X_i = 2x_i - 1 \quad (1 \leq i \leq n)$$

2. $1 \leq k \leq n$ について，$S_k = \sum_{i=1}^{k} X_i$ を計算する．

3. $S' = (0, S_1, S_2, \ldots, S_n, 0)$ とする．

4. S' の先頭の 0 を除く 0 の個数を求め，それを J とする．J はまた，隣り合う 0 をそれぞれ先頭，末尾とする部分列（サイクル）の個数である．$J < 500$ であれば検定を中止する．

5. 各サイクルについて，$-4 \leq x \leq -1$，$1 \leq x \leq 4$ を満たす 8 個の x について，x の現れる回数を求める．

6. 8 個の x それぞれについて，$k = 0, 1, 2, 3, 4$ に対し，x がちょうど k 回現れるサイクルの個数 $\nu_k(x)$ と，x が 5 回以上現れるサイクルの個数 $\nu_5(x)$ を求める．なお，各 x について，$\sum_{k=0}^{5} \nu_k(x) J$ である．

7. 8 個の x それぞれについて，

$$\chi^2(obs) = \sum_{k=0}^{5} \frac{(\nu_k(x) - J\pi_k(x))^2}{J\pi_k(x)}$$

を計算する．なお，$\pi_k(x)$ の値は以下の通りである．

x	$\pi_0(x)$	$\pi_1(x)$	$\pi_2(x)$	$\pi_3(x)$	$\pi_4(x)$	$\pi_5(x)$
± 1	0.5000	0.2500	0.1250	0.0625	0.0312	0.0312
± 2	0.7500	0.0625	0.0469	0.0352	0.0264	0.0791
± 3	0.8333	0.0278	0.0231	0.0193	0.0161	0.0804
± 4	0.8750	0.0156	0.0137	0.0120	0.0105	0.0733

8. 8 個の x それぞれについて，p-value $= \mathbf{igamc}(5/2, \chi^2(obs)/2)$ を計算する．

3.1.16 種々のランダム偏差検定

入出力

入力	x	乱数列 $x = (x_1, x_2, \ldots, x_n)$
	n	乱数列の長さ
出力	p-value	正規分布統計量からの p-value（18 個）

アルゴリズム

1. 入力系列 x を，次のように -1 と 1 からなる系列 $X = (X_1, X_2, \ldots, X_n)$ へ変換する．

$$X_i = 2x_i - 1 \quad (1 \leq i \leq n)$$

2. $1 \leq k \leq n$ について，$S_k = \sum_{i=1}^{k} X_i$ を計算する．

3. $S' = (0, S_1, S_2, \ldots, S_n, 0)$ とする．また，S' のサイクル数 J を求める．

4. $-9 \leq x \leq -1$，$1 \leq x \leq 9$ を満たす 18 個の x について，x が S' の中に現れる回数 $\xi(x)$ を求める．

5. 18 個の x それぞれについて，p-value $= erfc\left(\dfrac{|\xi(x) - J|}{\sqrt{2J(4|x| - 2)}}\right)$ を計算する．

3.2 擬似乱数列生成器の検定法

擬似乱数生成器に関しては，その擬似乱数生成器によって得られる複数の乱数系列について検定を行い，

1. p-value が 0.01 以上になる割合

2. p-value の一様性

によって擬似乱数生成器の検定が行われる．

1 については，乱数列の個数を m としたとき，p-value が 0.01 以上になる割合が

$$0.99 \pm 3\sqrt{\frac{0.99 \times 0.01}{m}}$$

の範囲にある場合，良い擬似乱数生成器であると判定される．

2 については，区間 $[0, 1)$ を 10 分割し，各区間に属する p-value の個数が均等であるかどうかを χ^2 分布によって検定する．具体的には，$1 \leq i \leq 10$ について，F_i を区間 $[(i-1)/10, i/10)$ に属する p-value の個数とするとき，

$$\chi^2 = \sum_{i=1}^{10} \frac{(F_i - m/10)^2}{m/10}$$

を計算し，p-value $= \mathbf{igamc}(9/2, \chi^2/2)$ を計算する．p-value ≥ 0.0001 のとき，良い擬似乱数生成器であると判定される．

3.3 計算機実験の結果

本節では，NIST SP800-22 で用意されている擬似乱数生成器の出力系列を，NIST SP800-22 の検定法により検定した結果について述べる．

それぞれの乱数生成器によって，500 本の 10 万ビットの乱数列を生成し，それを検定した結果を表 3.1 に示す．また，1000 本の 10 万ビットの乱数列を生成し，それを検定した結果を表 3.2 に示す．これらの表の中の検定法の内，上の 5 個はミニマムセットに含まれる方法である．また，離散フーリエ変換検定については，次章で述べる修正が施されている．なお，表の中で “x” は検定に合格しなかったことを表し，“–” は系列長が推奨値に満たないため，検定を行わなかったことを表している．幾つかの検定で使用したパラメータの値は以下の通りである．

検定名	ブロック長
ブロック単位の度数検定	20000
系列検定	10
近似エントロピー検定	10

それぞれの乱数生成器によって，500 本の 100 万ビットの乱数列を生成し，それを検定した結果を表 3.3 に示す．幾つかの検定で使用したパラメータの値は以下の通りである．

検定名	ブロック長
ブロック単位の度数検定	20000
重なりのないテンプレート適合検定	9
重なりのあるテンプレート適合検定	9
Maurer のユニバーサル統計検定 (初期系列)	7(1280)
線形複雑度検定	500
系列検定	10
近似エントロピー検定	10

表 3.1: 10 万 bit×500 本の検定結果 (U：一様性　P：比率)

1. G Using SHA-1
2. Linear Congruential
3. Blum-Blum-Shub
4. Micali-Schnorr
5. Modular Exponentiation
6. Quadratic Congruential I
7. Quadratic Congruential II
8. Cubic Congruential
9. XOR
10. ANSI X9.17 (3-DES)
11. G Using DES

	1		2		3		4		5		6	
	U	P	U	P	U	P	U	P	U	P	U	P
ブロック単位の頻度検定												
累積和検定									x		x	x
ブロック単位の最長連検定												
系列検定												
線形複雑度検定	–	–	–	–	–	–	–	–	–	–	–	–
1 次元度検定									x		x	x
連の検定												
2 値行列ランク検定												
離散フーリエ変換検定												
重なりのないテンプレート適合検定	–	–	–	–	–	–	–	–	–	–	–	–
重なりのあるテンプレート適合検定	–	–	–	–	–	–	–	–	–	–	–	–
Maurer のユニバーサル統計検定	–	–	–	–	–	–	–	–	–	–	–	–
近似エントロピー検定												
ランダム偏差検定	–	–	–	–	–	–	–	–	–	–	–	–
種々のランダム偏差検定	–	–	–	–	–	–	–	–	–	–	–	–
Lempel-Ziv 検定	–	–	–	–	–	–	–	–	–	–	–	–

	7		8		9		10		11	
	U	P	U	P	U	P	U	P	U	P
ブロック単位の頻度検定										
累積和検定			x							
ブロック単位の最長連検定						x				
系列検定					x	x				
線形複雑度検定	–	–	–	–	–	–	–	–	–	–
1 次元度検定			x							
連の検定			x	x						
2 値行列ランク検定					x	x				
離散フーリエ変換検定										
重なりのないテンプレート適合検定	–	–	–	–	–	–	–	–	–	–
重なりのあるテンプレート適合検定	–	–	–	–	–	–	–	–	–	–
Maurer のユニバーサル統計検定	–	–	–	–	–	–	–	–	–	–
近似エントロピー検定					x	x				
ランダム偏差検定	–	–	–	–	–	–	–	–	–	–
種々のランダム偏差検定	–	–	–	–	–	–	–	–	–	–
Lempel-Ziv 検定	–	–	–	–	–	–	–	–	–	–

表 3.2: 10 万 bit×1000 本の検定結果 (U：一様性　P：比率)

1. G Using SHA-1
2. Linear Congruential
3. Blum-Blum-Shub
4. Micali-Schnorr
5. Modular Exponentiation
6. Quadratic Congruential I
7. Quadratic Congruential II
8. Cubic Congruential
9. XOR
10. ANSI X9.17 (3-DES)
11. G Using DES

	1		2		3		4		5		6	
	U	P	U	P	U	P	U	P	U	P	U	P
ブロック単位の頻度検定					x							
累積和検定									x	x	x	x
ブロック単位の最長連検定	x				x				x			
系列検定												
線形複雑度検定	–	–	–	–	–	–	–	–	–	–	–	–
1 次元度検定					x				x	x	x	x
連の検定					x							
2 値行列ランク検定					x							
離散フーリエ変換検定	x											
重なりのないテンプレート適合検定	–	–	–	–	–	–	–	–	–	–	–	–
重なりのあるテンプレート適合検定	–	–	–	–	–	–	–	–	–	–	–	–
Maurer のユニバーサル統計検定	–	–	–	–	–	–	–	–	–	–	–	–
近似エントロピー検定												
ランダム偏差検定	–	–	–	–	–	–	–	–	–	–	–	–
種々のランダム偏差検定	–	–	–	–	–	–	–	–	–	–	–	–
Lempel-Ziv 検定	–	–	–	–	–	–	–	–	–	–	–	–

	7		8		9		10		11	
	U	P	U	P	U	P	U	P	U	P
ブロック単位の頻度検定										
累積和検定			x	x						
ブロック単位の最長連検定	x					x			x	
系列検定					x	x				
線形複雑度検定	–	–	–	–	–	–	–	–	–	–
1 次元度検定			x	x						
連の検定			x	x						
2 値行列ランク検定					x	x				
離散フーリエ変換検定									x	
重なりのないテンプレート適合検定	–	–	–	–	–	–	–	–	–	–
重なりのあるテンプレート適合検定	–	–	–	–	–	–	–	–	–	–
Maurer のユニバーサル統計検定	–	–	–	–	–	–	–	–	–	–
近似エントロピー検定			x	x	x	x				
ランダム偏差検定	–	–	–	–	–	–	–	–	–	–
種々のランダム偏差検定	–	–	–	–	–	–	–	–	–	–
Lempel-Ziv 検定	–	–	–	–	–	–	–	–	–	–

表 3.3: 100 万 bit×500 本の検定結果 (U：一様性　P：比率)

1. G Using SHA-1
2. Linear Congruential
3. Blum-Blum-Shub
4. Micali-Schnorr
5. Modular Exponentiation
6. Quadratic Congruential I
7. Quadratic Congruential II
8. Cubic Congruential
9. XOR
10. ANSI X9.17 (3-DES)
11. G Using DES

	1		2		3		4		5		6	
	U	P	U	P	U	P	U	P	U	P	U	P
ブロック単位の頻度検定									x	x	x	x
累積和検定									x	x	x	x
ブロック単位の最長連検定												
系列検定												
線形複雑度検定												
1 次元度検定									x	x	x	x
連の検定												x
2 値行列ランク検定												
離散フーリエ変換検定												
重なりのないテンプレート適合検定												
重なりのあるテンプレート適合検定												
Maurer のユニバーサル統計検定												
近似エントロピー検定												
ランダム偏差検定												
種々のランダム偏差検定												
Lempel-Ziv 検定												

	7		8		9		10		11	
	U	P	U	P	U	P	U	P	U	P
ブロック単位の頻度検定										
累積和検定			x	x						
ブロック単位の最長連検定										
系列検定			x	x	x	x				
線形複雑度検定					x	x				
1 次元度検定			x	x						
連の検定			x	x						
2 値行列ランク検定					x	x				
離散フーリエ変換検定	x	x	x	x						
重なりのないテンプレート適合検定					x	x				
重なりのあるテンプレート適合検定					x	x				
Maurer のユニバーサル統計検定					x	x				
近似エントロピー検定			x	x	x	x				
ランダム偏差検定										
種々のランダム偏差検定										
Lempel-Ziv 検定					x					

第4章　離散フーリエ変換検定について

本章では，NIST SP800-22 の離散フーリエ変換検定について報告されている問題に関する調査結果を報告する．

4.1　NIST SP800-22 の離散フーリエ変換検定

NIST SP800-22 の離散フーリエ変換検定については，前章でも述べたが，少し説明を付加して以下に再掲する．入力系列 $x = (x_1, x_2, \ldots, x_n)$ を 0 と 1 からなる系列とする．n は系列の長さである．

1. 0 と 1 からなる長さ n の入力系列 $x = (x_1, x_2, \ldots, x_n)$ を，次のように -1 と 1 からなる系列 $X = (X_1, X_2, \ldots, X_n)$ へ変換する．

$$X_i = 2x_i - 1 \quad (1 \leq i \leq n)$$

2. 系列 X を離散フーリエ変換し，複素数の列 $S = (S_1, S_2, \ldots, S_n)$ を得る．
3. S' を S の最初の $n/2$ 個からなる部分列とし，$M = modulus(S') = |S'|$ を求める．すなわち，$S' = (S_1, S_2, \ldots, S_{n/2})$ に対し，$i = 1, 2, \ldots, n/2$ について，$|S_i|$ を求める．
4. 95%のピークの高さのしきい値である $T = \sqrt{3n}$ を求める．
5. $N_0 = 0.95n/2$ を求める．N_0 は，X が真の乱数列であるときに，$|S_1|, |S_2|, \ldots, |S_{n/2}|$ のうちで T を越えない $|S_i|$ の個数の理論値である．
6. $|S_1|, |S_2|, \ldots, |S_{n/2}|$ の中で実際に T を越えなかった $|S_i|$ の個数 N_1 を求める．
7. $d = \dfrac{N_1 - N_0}{\sqrt{n(0.95)(0.05)/2}}$ を求める．
8. X が真の乱数列であるとき，$|d|$ は標準正規分布に従うので，この分布から p-value $= erfc(|d|/\sqrt{2})$ を計算する．

4.2　調査結果

本節では，NIST SP800-22 の離散フーリエ変換検定の原理を確認し，この検定の問題が生じた原因を明らかにする．なお，特にことわらない限り，以下の内容は概ね文献 [7] に基づく．

前々節で述べた NIST SP800-22 の離散フーリエ変換検定で，手続き 2 の $S = (S_1, S_2, \ldots, S_n)$ は

$$S_j = \sum_{k=1}^{n} X_k \cos \frac{2\pi(k-1)j}{n} + \mathrm{i} \sum_{k=1}^{n} X_k \sin \frac{2\pi(k-1)j}{n}$$

である．ここで，$\mathrm{i} = \sqrt{-1}$ である．

手続きの 3 で，S の部分列 $S' = (S_1, S_2, \ldots, S_{n/2})$ のみ考えるのは，$j = 1, 2, \ldots, n/2 - 1$ について，S_j と S_{n-j} が共役であり，$|S_j| = |S_{n-j}|$ が成立することによる．

簡単のために，

$$\begin{aligned} c_j &= \sum_{k=1}^{n} X_k \cos \frac{2\pi(k-1)j}{n} \\ s_j &= \sum_{k=1}^{n} X_k \sin \frac{2\pi(k-1)j}{n} \end{aligned}$$

と表記すると，

$$|S_j|^2 = {c_j}^2 + {s_j}^2$$

である．

ここで，X が真の乱数列であれば，c_j，s_j の分布は共に，n が大きくなるにつれて，平均 $\mu = 0$，分散 $\sigma^2 = n/2$ の正規分布 $N(0, n/2)$ に近づくことが証明できる．

c_j について，平均，分散は以下のように計算できる．s_j についても同様に計算できる．

$$\frac{1}{2^n} \sum_X c_j(X) = \frac{1}{2^n} \sum_X \sum_{k=1}^{n} X_k \cos \frac{2\pi(k-1)j}{n} = 0$$

$$\begin{aligned} \frac{1}{2^n} \sum_X (c_j(X))^2 &= \frac{1}{2^n} \sum_X \left(\sum_{k=1}^{n} X_k \cos \frac{2\pi(k-1)j}{n} \right)^2 \\ &= \frac{1}{2^n} \sum_X \left(\sum_{k=1}^{n} {X_k}^2 \left(\cos \frac{2\pi(k-1)j}{n} \right)^2 + \sum_{\ell_1 \neq \ell_2} X_{\ell} X_{\ell_2} \cos \frac{2\pi(\ell_1 - 1)j}{n} \cos \frac{2\pi(\ell_2 - 1)j}{n} \right) \\ &= \frac{1}{2^n} \sum_X \sum_{k=1}^{n} \left(\cos \frac{2\pi(k-1)j}{n} \right)^2 \\ &= \sum_{k=1}^{n} \left(\cos \frac{2\pi(k-1)j}{n} \right)^2 \\ &= \sum_{k=1}^{n} \frac{1}{2} \left(1 + \cos \frac{4\pi(k-1)j}{n} \right) \\ &= \frac{n}{2} \end{aligned}$$

今，

$$y \stackrel{\mathrm{def}}{=} \left(\frac{c_j}{\sigma} \right)^2 + \left(\frac{s_j}{\sigma} \right)^2 = \frac{|S_j|^2}{\sigma^2}$$

と定義すると，以下に示す定理により，y は自由度 2 の χ^2-分布に従うことが分かる．すなわち，

$$T_2(y) = \begin{cases} \frac{1}{2} \mathrm{e}^{-\frac{y}{2}} & y > 0 \text{ のとき} \\ 0 & y \leq 0 \text{ のとき} \end{cases}$$

である．

定理 1 [9] 正規分布 $N(\mu, \sigma^2)$ に従う母集団から，大きさ n の標本 $z_1, z_2, \ldots, z_n$ をランダムに選び，

$$y = \frac{1}{\sigma^2}\sum_{i=1}^{n}(z_i - \mu)^2$$

を作ると，これは自由度 n の χ^2 分布に従う． □

手続き 4 の T は，T より大きな $|S_j|$ が 5%であるように定められているので，$y = \frac{|S_j|^2}{\sigma^2}$ より，

$$\int_{T^2/\sigma^2}^{\infty} \frac{1}{2}\mathrm{e}^{-\frac{y}{2}}\mathrm{d}y = \mathrm{e}^{-\frac{T^2}{2\sigma^2}} = 0.05$$

が成立する．したがって，

$$T^2 = -2\sigma^2 \ln 0.05 = (-\ln 0.05)n$$

であり，$-\ln 0.05 \approx 2.995732274$ であることから，

$$T \approx \sqrt{2.995732274\, n}$$

であることが分かる．

次に手続き 7 では，N_1 が，独立な試行の回数 $N = n/2$，$p = 0.\,95$ としたときの二項分布に従うと考え，分散を Npq としている．[7] では，分散を $(N/2)pq$ とするのが適切であることが，計算機実験により確認されている．なお，このことは，図 4.1 に示す簡易な Mathematica のスクリプトを用いても確認することができる．このスクリプトでは，-1 と 1 からなる長さ 1024 の乱数列 1000 個を用いて，標準正規分布の上側 2.5%に属する割合により確認を行っている．

分散が Npq とならない理由は，c_j, s_j が同じ乱数列 X を用いて計算されていることであると考えられる．これを確認するため，図 4.1 と同様のスクリプトで，c_j，s_j を別の乱数列を用いて計算したところ，分散は $(N/2)pq$ とすると小さすぎることが確認できた．但し，この場合，Npq は分散の値として大きすぎることも同時に確認された．これに関しては，今後の検討が必要である．

4.3 計算機実験

以上で述べた調査結果に基づいて NIST SP800-22 の離散フーリエ変換検定を修正し，NIST SP800-22 で用意されている擬似乱数生成器により生成される擬似乱数の検定を行った．

4.1 で述べた離散フーリエ変換検定の修正箇所は以下の通りである．

- 手続き 4 で，$T = \sqrt{2.995732274\, n}$ とした．
- 手続き 7 で，$d = \dfrac{N_1 - N_0}{\sqrt{n(0.95)(0.05)/4}}$ とした．

計算機実験の結果を表 4.1 および表 4.2 に示す．表 4.1 は，10 万ビットの系列の検定，表 4.2 は 100 万ビットの系列の検定の結果である．

表 4.1: 修正されたフーリエ変換検定による検定結果 1

(a) 10 万 bit × 500 本

乱数生成器	p-value		判定	
	一様性	比　率	一様性	比　率
G Using SHA-1	0.337688	0.9920		
Linear Congruential	0.301194	0.9820		
Blum-Blum-Shub	0.103753	0.9840		
Micali-Schnorr	0.222480	0.9760		
Modular Exponentiation	0.892036	0.9940		
Quadratic Congruential I	0.131122	0.9920		
Quadratic Congruential II	0.191687	0.9840		
Cubic Congruential	0.071177	0.9840		
XOR	0.404728	0.9880		
ANSI X9.17 (3-DES)	0.006956	0.9880		
G Using DES	0.025193	0.9780		

(b) 10 万 bit × 1000 本

乱数生成器	p-value		判定	
	一様性	比　率	一様性	比　率
G Using SHA-1	0.000041	0.9860	x	
Linear Congruential	0.299736	0.9860		
Blum-Blum-Shub	0.065639	0.9910		
Micali-Schnorr	0.092041	0.9870		
Modular Exponentiation	0.408275	0.9880		
Quadratic Congruential I	0.089301	0.9900		
Quadratic Congruential II	0.002993	0.9850		
Cubic Congruential	0.017912	0.9840		
XOR	0.116065	0.9880		
ANSI X9.17 (3-DES)	0.058612	0.9900		
G Using DES	0.000009	0.9850	x	

```
n1s =
Table[
  Count[
    Positive[
      Take[32 Abs[Fourier[Table[2 Random[Integer] - 1, {1024}]]], 512]
        - Sqrt[2.995732274 1024]
    ],
    False
  ],
  {1000}
];
n0 = .95 512;
d1 = (n1s - n0) / Sqrt[1024 .95 .05 / 2];
d2 = (n1s - n0) / Sqrt[1024 .95 .05 / 4];
{Count[Positive[d1 - 1.96], True], Count[Positive[d2 - 1.96], True]}
```

図 4.1: N_1 の分布の分散を確認するための Mathematica のスクリプト

表 4.2: 修正されたフーリエ変換検定による検定結果 2

100 万 bit × 500 本

乱数生成器	p-value		判定	
	一様性	比　率	一様性	比　率
G Using SHA-1	0.591409	0.9960		
Linear Congruential	0.307077	0.9860		
Blum-Blum-Shub	0.146982	0.9840		
Micali-Schnorr	0.844641	0.9900		
Modular Exponentiation	0.040635	0.9780		
Quadratic Congruential I	0.397688	0.9900		
Quadratic Congruential II	0.000000	0.9540	x	x
Cubic Congruential	0.000000	0.8200	x	x
XOR	0.244236	0.9940		
ANSI X9.17 (3-DES)	0.735908	0.9900		
G Using DES	0.889118	0.9900		

第5章　線形合同法による擬似乱数列について

5.1　NIST SP800-22 の線形合同法に基づく生成器

NIST SP800-22 で Linear Congruential Generator (LCG) と呼ばれている線形合同法に基づく生成器は以下の通りである.

入力 s_0 が与えられると，LCG は以下の規則にしたがって，擬似乱数列 $x_0, x_1, x_2, \ldots$ を出力する.

1. $s_1, s_2, \ldots$ は以下の式によって計算される.

$$s_{i+1} = a \times s_i \bmod 2^{31} - 1$$

2. $x_0, x_1, x_2, \ldots$ は以下のように定められる.

$$x_i = \begin{cases} 0 & s_{i+1}/(2^{31}-1) < 0.5 \text{ のとき} \\ 1 & \text{その他} \end{cases}$$

なお，LCG のプログラムでは，$s_0 = 23482349$，$a = 950706376$ が用いられている.

以上のことから，LCG は，$s_1, s_2, s_3, \ldots$ それぞれの最上位ビットのみからなる系列を擬似乱数列として出力することが分かる．したがって，LCG は次に述べる linear truncated congruential generator の一例である．以下では，linear truncated congruential generator の定義を与えた後，この生成器に対する従来の攻撃アルゴリズムに関する調査結果を記す.

5.2　linear truncated congruential generator

線形合同法に基づく擬似乱数生成器 (linear congruential generator) は，以下の式によって，擬似乱数列 $s_0, s_1, s_2, \ldots$ を生成する.

$$s_i = a\, s_{i-1} + b \bmod m$$

s_i の上位 t ビットを x_i と記すとき，$x_0, x_1, x_2, \ldots$ を出力する擬似乱数生成器は，linear truncated congruential generator と呼ばれる．s_i のビット数を n とするとき，x_i は以下のように表される.

$$x_i = \left\lfloor \frac{s_i}{2^{n-t}} \right\rfloor .$$

5.3　Freize, Håstad, Kannan, Lagarias, Shamir のアルゴリズム

5.3.1　アルゴリズム

本節で述べる Freize, Håstad, Kannan, Lagarias, Shamir によるアルゴリズム [3, 4, 5] は，

$$s_i = a\, s_{i-1} + b \bmod m$$

について，$b = 0$ かつ a, m が既知であるという仮定の下で，$x_0, x_1, x_2, \ldots$ から $s_0, s_1, s_2, \ldots$ を計算するアルゴリズムである．

$b = 0$ の場合については，$\hat{x}_i = x_i - x_{i-1}$ を入力としてアルゴリズムを実行する．$\hat{s}_i = s_i - s_{i-1}$ とすると，

$$\hat{s}_i = a\,\hat{s}_{i-1} \bmod m$$

が成立する．

x_i は s_i の上位 t ビットを表すものとする．n を s_i のビット数とし，β を $t = (1-\beta)n$ を満たす定数とすれば，

$$s_i = x_i\,2^{\beta n} + y_i$$

と表すことができる．なお，y_i は s_i の下位 $\beta\,n$ ビットである．

Freize らのアルゴリズムは，格子基底縮小に基づくアルゴリズムである．

L を以下のような k 個のベクトル $b_1, b_2, \ldots, b_k$ によって生成される格子とする．なお，$b_i = (b_{i,1}, b_{i,2}, \ldots, b_{i,k})$ と表記する．

$$b_{1,j} = \begin{cases} m & j = 1 \text{ のとき} \\ 0 & \text{その他} \end{cases}$$

$i = 2, 3, \ldots, k$ について

$$b_{i,j} = \begin{cases} a^{i-1} & j = 1 \text{ のとき} \\ -1 & j = i \text{ のとき} \\ 0 & \text{その他} \end{cases}$$

このとき，L に属するすべてのベクトル v_ℓ について，

$$v_{\ell,1}\,s_1 + v_{\ell,2}\,s_2 + \cdots + v_{\ell,k}\,s_k \equiv 0 \pmod{m}$$

が成立することは容易に確認できる．

linear truncated congruential generator に対する Freize らのアルゴリズムは以下の通りである．

1. 格子基底縮小アルゴリズムを利用して，基底 $v_1, v_2, \ldots, v_k$ を計算する．ここで，$i = 1, 2, \ldots, k$ について，

   $$\sum_{j=1}^{k} v_{i,j}\,s_j = \sum_{j=1}^{k} v_{i,j}\,x_j\,2^{\beta n} + \sum_{j=1}^{k} v_{i,j}\,y_j \equiv 0 \pmod{m}$$

   が成立することに注意する．

2. $i = 1, 2, \ldots, k$ について，

   $$\left| \sum_{j=1}^{k} v_{i,j}\,y_j \right| < \frac{m}{2}$$

   が成立するとき，$\displaystyle\sum_{j=1}^{k} v_{i,j}\,x_j\,2^{\beta n}$ の値から，$\displaystyle\sum_{j=1}^{k} v_{i,j}\,s_j$ が一意に定まることが分かる．こうして得られる連立方程式を解くことにより，$s_1, s_2, \ldots, s_k$ を求めることができる．

この攻撃の効果については以下の定理が証明されている．

定理 2 m は平方因子を持たない整数，$\varepsilon > 0$，k を与えられた整数とする．このとき，以下の条件を満たす定数 $C(\varepsilon, k)$ が存在する．

> $m > C(\varepsilon, k)$ かつ $(1-\beta)\,n > (k^{-1}+\varepsilon)\,n + c(k)$ が成立するとき，少なくとも $1 - O(m^{-\varepsilon})$ の割合の a について，
>
> $$\left| \sum_{j=1}^{k} v_{i,j}\, y_j \right| < \frac{m}{2}$$
>
> を満たす基底を格子基底縮小アルゴリズムによって見つけることができる．なお，
>
> $$c(k) = \frac{k}{2} + (k-1)\log 3 + \frac{7}{2}\log k + 2$$
>
> である．

□

5.3.2 考察

Freize らのアルゴリズムは，$n+k$ の多項式時間アルゴリズムであり，非常に強力なアルゴリズムであると考えられる．但し，定理 2 では，各 x_i のビット数が，$(\frac{1}{k}+\varepsilon)\,n + \frac{k}{2} + (k-1)\log 3 + \frac{7}{2}\log k + 2$ であることが要求されている．

一方，NIST SP800-22 の LCG では，各 x_i のビット数は 1 であることから，Freize らのアルゴリズムは，この LCG に対しては必ずしも有効ではないと考えられる．また，定理 2 の条件

$$\left| \sum_{j=1}^{k} v_{i,j}\, y_j \right| < \frac{m}{2}$$

を考慮しても，x_j のビット数が 1 であるとき，$y_j \approx \frac{m}{2}$ となると考えられるので，この点からも Freize らのアルゴリズムが有効であると考えることは難しい．

各 x_i のビット数が小さいときに，Freize らのアルゴリズムが有効であるかどうかの検証には，計算機による数値実験が必要である．

5.4 Knuth のアルゴリズム

本節で述べる Knuth によるアルゴリズム [8] は，

$$s_i = a\, s_{i-1} + c \bmod m$$

について，$m = 2^k$，$a \bmod 4 = 1$，$c \bmod 2 = 1$ という仮定の下で，k が既知のときに，$x_0, x_1, x_2, \ldots$ から a, c, s_0 を計算するアルゴリズムである．より詳細に述べると，[8] には，下の二つの問題を解くアルゴリズムが与えられている．なお，以下では $k = h + l$ である．

問題 A

入力：$\lfloor s_i/2^l \rfloor$ $(0 \le i < 2^l)$
出力：a，c，s_0

問題 B

入力：$\lfloor s_i/2^l \rfloor$ $(0 \le i < N)$

出力：a，c，s_0

問題 A については，$l > 1$ に対して，$O(2^l)$ ステップのアルゴリズム，問題 B については，$h \ge 2$ に対して，$O(k^2 2^{2l}/N^2)$ ステップのアルゴリズムが与えられている．

5.4.1 問題 A のアルゴリズム

s_i の下位から l 番目のビットを $s_i^{(l)}$ と表記する．このとき，

$$s_i^{(l)} = \lfloor s_i/2^{l-1} \rfloor \bmod 2$$

である．なお，最下位ビットは下位から 1 番目のビットと呼ばれる．

問題 A のアルゴリズムでは，$\lfloor s_i/2^l \rfloor$ $(0 \le i \le 2^l)$ から，$\lfloor s_i/2^{l-1} \rfloor$ $(0 \le i \le 2^{l-1})$ を求めることを繰り返すことにより処理が進められる．このためには，$\lfloor s_i/2^l \rfloor$ $(0 \le i \le 2^l)$ から，$s_i^{(l)} = \lfloor s_i/2^{l-1} \rfloor \bmod 2$ $(0 \le i \le 2^{l-1})$ が求められれば十分である．

補題 1 $0 \le t < k$ について，

$$y_{n+1}^{(t)} = s_{n+2^t} - s_n \bmod 2^k$$

とする．このとき，すべての n について

$$y_{n+1}^{(t)} = a\, y_n^{(t)} \bmod 2^k$$

である．また，$y_n^{(t)}$ は 2^t の奇数倍である． □

補題 2 $1 \le l < k-1$ について，以下を満たす定数 $b_l \in \{0,1\}$ が存在する．

$$s_n^{(l)} = s_{n+2^{l-1}}^{(l+1)} - s_n^{(l+1)} + b_l \bmod 2$$

□

補題 2 より，b_l が分かれば，$\lfloor s_i/2^l \rfloor$ $(0 \le i < 2^l)$ の最下位ビットから，$s_i^{(l)} = \lfloor s_i/2^{l-1} \rfloor \bmod 2$ $(0 \le i < 2^{l-1})$ が求められることが分かる．さらに，補題 1 より，

$$s_{n+2^{l-1}} \equiv y_n^{(l-1)} + s_n \bmod 2^k$$

であり，かつ，$y_n^{(l-1)}$ の下位から l 番目のビットは 1 で，それより下位のビットはすべて 0 であるから，

$$s_{n+2^{l-1}}^{(l)} = 1 - s_n^{(l)}$$

であることが分かる．これらのことから，b_l が分かれば，$\lfloor s_i/2^l \rfloor$ $(0 \le i < 2^l)$ から，$\lfloor s_i/2^{l-1} \rfloor$ $(0 \le i \le 2^{l-1})$ が計算できることが分かる．

$\lfloor s_i/2^l \rfloor$ $(0 \le i \le 2^l)$ が分かっているとき，$l \ge 2$ ならば，b_l は以下の式によって計算できることが証明できる．

$$\lfloor s_0/2^l \rfloor - 2\lfloor s_{2^{l-1}}/2^l \rfloor + \lfloor s_{2^l}/2^l \rfloor \equiv 2s_0^{(l)} + 1 \pmod 4$$

$$s_0^{(l)} = s_{2^{l-1}}^{(l+1)} - s_0^{(l+1)} + b_l \bmod 2$$

ただし，アルゴリズムの入力としては $\lfloor s_{2^l}/2^l \rfloor$ が与えられないので，b_l を計算することができない．この場合は以下のように対処する．

- $h = 1$ の場合は，b_l は 0 でも 1 でもどちらでも良いことが証明できる．
- $h \geq 2$ の場合は，$b_l = 0$ と $b_l = 1$ の両方について処理を進める．

問題 A に対するアルゴリズムは以下の通りである．

1. $h \geq 2$ ならば，以下のステップを $b_l = 0$ と $b_l = 1$ の両方について実行する．それ以外の場合，すなわち，$h = 1$ の場合は，$b_l = 0$ についてのみ実行する．
2. $0 \leq n < 2^{l-1}$ について，

$$s_n^{(l)} = s_{n+2^{l-1}}^{(l+1)} - s_n^{(l+1)} + b_l \bmod 2$$

を用いて $s_n^{(l)}$ を計算する．次に，$x_{2^{l-1}}^{(l)} = 1 - x_0^{(l)}$ とする．l を 1 減らす.

3. $l \geq 2$ ならば，次の二つの式

$$\lfloor s_0/2^l \rfloor - 2\lfloor s_{2^{l-1}}/2^l \rfloor + \lfloor s_{2^l}/2^l \rfloor \equiv 2s_0^{(l)} + 1 \pmod 4$$
$$s_0^{(l)} = s_{2^{l-1}}^{(l+1)} - s_0^{(l+1)} + b_l \bmod 2$$

により，b_l を計算してステップ 2 に戻る．

4. $(x_0^{(1)}, x_1^{(1)}, x_2^{(1)})$ が取り得る 2 通りの値 $(0,1,0)$，$(1,0,1)$ の両方について，

$$\begin{aligned} a &= \frac{x_2 - x_1}{x_1 - x_0} \bmod 2^k \\ c &= x_1 - a\,x_0 \bmod 2^k \end{aligned}$$

を計算する．得られた値から与えられた入力が生成できるかどうかを確認する．

なお，このアルゴリズムは，入力について $l = 1$ の場合はうまく動作しない．[8] では，この場合の解法についても言及されている．

5.4.2 問題 B のアルゴリズムについて

問題 B については，N が小さいと非常に多数の解が存在する．今，

$$s'_n = s_n + b \bmod 2^k$$

と定義すると，

$$s'_{n+1} = as'_n + (c - (a-1)b) \bmod 2^k$$

が成立する．このとき，$s_{\min} = \min_{0 \leq n < N} \{x_n \bmod 2^l\}$，$s_{\max} = \max_{0 \leq n < N} \{x_n \bmod 2^l\}$ とすると，$-s_{\min} \leq b < 2^l - s_{\max}$ ならば，$0 \leq n < N$ について，$\lfloor s'_n/2^l \rfloor = \lfloor s_n/2^l \rfloor$ が成立する．したがって，このような b について $s'_0 = s_0 + b \bmod 2^k$ を満たす s'_0 はすべて正しい解となる．

ところで，

$$y_0 = s_1 - s_0 \bmod 2^k$$

と定義すると，補題 1 から，すべての n について，

$$\left\lfloor \frac{s_n}{2^l} \right\rfloor = \left\lfloor \frac{s_0}{2^l} \right\rfloor + \left\lfloor \frac{a^n - 1}{a - 1} \cdot \frac{y_0}{2^l} \right\rfloor + \epsilon_n \bmod 2^h$$

が成立することが証明できる．ただし，ϵ_n は 0 または 1 である．そこで，問題 B のアルゴリズムは，以下の二つの手続きによって構成される．

1. 与えられた入力から (a, y_0) を求める．
2. (a, y_0) から，与えられた入力と同一の系列を生成できる s_0 の範囲を求める．

(a, y_0) を求める手続きでは，まず，$2^t + 1 < N$ なる最大の t について，可能なすべての $y_0^{(t)} = s_{2^t} - s_0 \bmod 2^k$ を計算し，

$$y_n^{(t)} = (1 + a^{2^t - 1}) y_n^{(t-1)} \bmod 2^k$$

によって t を 1 ずつ減らすことにより，最終的に $y_0 = y_0^{(0)}$ を得る．なお，この処理中に，各 t について，

$$\left\lfloor \frac{y_n^{(t)}}{2^l} \right\rfloor + \kappa_n^{(t)} = \left\lfloor \frac{s_{n+2^t}}{2^l} \right\rfloor - \left\lfloor \frac{s_n}{2^l} \right\rfloor \bmod 2^h \quad \left(\kappa_n^{(t)} \text{は 0 または 1}\right)$$

を用いて可能な $y_0^{(t)}$ の絞り込みが行われるが，$h = 1$ のとき，この式は常に成り立つので，絞り込みには利用できない．したがって，このアルゴリズムでは $h \geq 2$ が仮定されている．

5.4.3 考察

本節で述べた Knuth のアルゴリズムは，非常に興味深いものであるが，NIST SP800-22 の LCG に適用するという観点からは，以下のような問題点が挙げられる．

- 線形合同式の法が 2 のべき乗の場合にのみ有効なアルゴリズムであること．
- ステップ数が，擬似乱数列として出力されない各 s_i のビット数乗に依存すること．また，問題 B のアルゴリズムでは，擬似乱数列として出力される各 s_i のビット数が 2 以上であることが仮定されていること．

5.5 まとめ

本章では，linear truncated congruential generator の未知パラメータを求めるためのアルゴリズムとして，Freize, Håstad, Kannan, Lagarias, Shamir によるアルゴリズムと Knuth によるアルゴリズムを概観した．その結果，これらのアルゴリズムは，NIST SP800-22 の LCG のように，線形合同法で生成される数列の各項の最上位ビットしか出力しないような生成器には必ずしも有効ではないということが確認できた．

ただし，これらのアルゴリズムはあくまで，linear truncated congruential generator の完全解読を目的としたアルゴリズムであるため，以上のことは，必ずしも，NIST SP800-22 の LCG のような生成器が生成する系列が暗号への利用に適した良い乱数列であることを意味するものではない．

参考文献

[1] NIST, Special Publication 800-22, A statistical test suite for random and pseudorandom number generators for cryptographic applications, 2001.

[2] R. N. Bracewell, *The fourier transform and its applications*, McGraw-Hill, 1986.

[3] A. M. Frieze, J. Hastad, R. Kannan, J. C. Lagarias, and A. Shamir, Reconstructing truncated integer variables satisfying linear congruences, SIAM J. Comput., 17(2):262–280, 1988.

[4] A. M. Frieze, R. Kannan, and J. C. Lagarias, Linear congruential generators do not produce random sequences, In 25th IEEE Symposium on Foundations of Computer Science (FOCS), pp. 480–484, 1984.

[5] J. Hastad and A. Shamir, The cryptographic security of truncated linearly related variables, In 17th ACM Symposium on Theory of Computing (STOC), pp. 356–362, 1985.

[6] 電子政府情報セキュリティ技術開発事業「擬似乱数検証ツールの調査開発」調査報告書, 情報処理振興事業協会セキュリティセンター, 2003 年 2 月.

[7] 金成主, 梅野健, 長谷川晃朗, NIST のランダム性評価テストについて, 信学技法, ISEC2003-87, pp. 21–27, 2003.

[8] D. E. Knuth, Deciphering a Linear Congruential Encryption, IEEE Trans. Information Theory, vol. IT-31, no. 1, pp. 49–52, 1985.

[9] 小針 [日見] 宏, 確率・統計入門, 岩波書店, 1973.

[10] G. Marsaglia, DIEHARD, `http://stat.fsu.edu/ geo/diehard.html`, `http://stat.fsu.edu/pub/diehard.html`,