修士論文

手書き曲線からの乱数生成

川西　暁夫

大阪府立大学大学院 理学系研究科 数理・情報科学専攻

学籍番号: 2046310037

2006 年 3 月

Random Extraction from Freehand Drawings (Abstract)

Akio Kawanishi
Department of Mathematics anf Information Sciences,
Osaka Prefecture University

1 Introduction

Random number generator is used in many fields such as simulation and cryptographic technology. In this paper, we present our algorithms of random number generation.

Kawanishi and Suzuki constructed algorithms that converts bitmap of freehand drawings to an almost random bit string (Figure 1). And the algorithm constructed by Kawanishi had good results for two statistical tests ('run test'and 'poker test'). Furthermore Suzuki revised the algorithm of Kawanishi. Suzuki showed that the revised algorithm also had good results for the above statistical tests, and he mathematically proved that it preserves the irregurarity of freehand drawings.

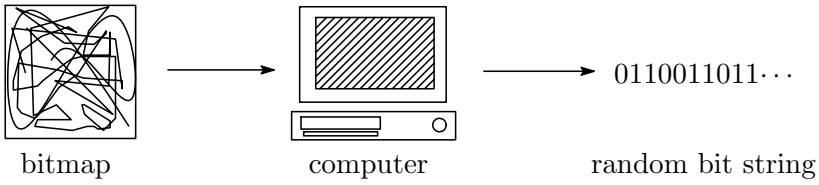


Figure 1: Random Extraction from Freehand Drawings

We explain the constitution of this paper. In section 2, we introduce bitmaps of freehand drawings that are random sources of our algorithm. In section 3, we show our algorithms. In section 4, we show experimental results by two statistical tests.

2 Bitmaps of Freehand Drawings

In this section, we introduce bitmaps of freehand drawings. The size of canvas of a bitmap is of 256×256 pixels. In addition, when we draw freehand drawings, the color of canvas is white and the color of pen is black. Some

drawing is drawn by Kawanishi, and some is drawn by Suzuki. Some drawing is drawn by a mouse, and some is drawn by a pen tablet. Some drawing is a meaningless curve, and some is a catoon of face.

We prepare 500 bitmaps (test201.bmp - test700.bmp). The following table and figures describe them.

Bitmap	Type
test201 - test300	meaningless curve drawn by mouse
test301 - test400	meaningless curve drawn by pen tablet
test401 - test500	cartoons of faces drawn by mouse
test501 - test600	cartoons of faces drawn by pen tablet
test601 - test700	geometrical designs

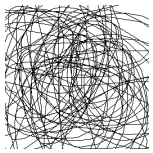


Figure 2: test201

Figure 3: test356

Figure 4: test437

Figure 5: test562

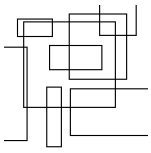


Figure 6: test620

3 Our Algorithms

In this section, we introduce the algorithms that converts a bitmap of freehand drawings to an almost random bit string. We call the algorithm of Kawanishi *KAWAL*, the algorithm of Suzuki *rng023*. Rng023 is a modification of KAWAL. The procedure of two algorithms is as follows.

1. A bitmap of freehand drawings is converted to a Boolean matrix A (each element of A is 0 or 1).
2. A is converted to an array l.
3. The lengths of *run*s in l is converted to binary number, and we output an array b. (†)
4. By applying XOR(exclusive-or) to b , we make an array x.
5. Output x.

(†) A *run* denotes a maximal string of same charactor such as '00000', '111'. For example, in the case of 010111001, the number of runs is 6.

An output from the above procedure is an almost random bit string. In the output of KAWAL and rng023 with a bitmap from test201 to test700, the rate of 1 is close to 50 % in the most of bitmaps.

Now we describe KAWAL more precisely.

step1: A bitmap of freehand drawings is converted to a Boolean matrix A.

Each black(white) pixel in a bitmap is converted to 1(0) in the corresponding matrix A as Figure 7.

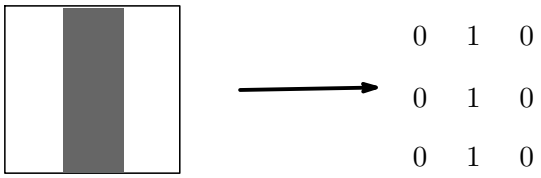


Figure 7: A bitmap of 3×3 pixels is converted to a Boolean matrix A

step2: A is converted to an array l.

A is converted to an array l as in the following figure.

step3: The lengths of runs in l are converted to binary numbers.

We explain this step by using the follwing example.

(example) l : 1 1 0 0 0 1 1 1 0 0 0 0 0

1. The lengths of runs in l are converted to binary numbers. In the case of the above example, because the lengths of runs in l are 2,3,3 and 5 from left, they are converted to 1,11,11 and 101. (‡)

2. We delete the leftmost bit (1) of each binary number. In the exceptinal case that the length of binary number is 1, we don't delete it. Therefore, we have 1,11,11,101 → 1,1,1,01.

3. Convert each binary number to its mirror writing (ABC → CBA). Let an array b be the resulting array. Therefore, we have 1,1,1,01 → 1,1,1,10, and $b = 11110$.

4. Output b .

(‡) In the case that the length of run is 1 or 2, we use different methods. In the case that the length of run is 2, 2 (the length of run) is converted to 1. In the case that the length of run is 1, according to the position of the

run, 1 (the length of run) is converted to 0 or 1. That is, if the position of the run is even (odd, respectively) from the head bit of the array l, 1 (the length of run) is converted to 0 (1, respectively).

step4: Apply XOR to b.

Suppose that $b = b_1 b_2 \cdots b_n$ (b_j is 0 or 1). Get an array x as follows. $x =$ b_1 XOR b_n, b_2 XOR b_{n-1}, $\cdots$, b_i XOR b_{n-i+1}. If n is even, we let $i = n/2$. If n is odd, $i = (n-1)/2$.

$$
\begin{array}{lcccccccccccc}
b: & 1 & 0 & 0 & 0 & 1 & \cdots & 0 & 1 & 0 & 1 & 1 \\
 & & \rightarrow & & & & & & & & \leftarrow & \\
 & & & & & & \Downarrow & & & & & \\
x: & 0 & 1 & 0 & 1 & 1 & \cdots & & & & &
\end{array}
$$

step5: Output x.

4 Statistical Tests

In this section, we introduce two statistical tests (run test and poker test) in order to examine the randomness of output of our algorithm. And we show the experimental results.

4.1 Run Test

Run test is a method of statistical test based on the number of runs in a random sequence. We collect 100 bitmaps of same type (for example, test201 - test300) and examine the randomness of output under the following procedure.

step1: Get statistic V.

For each bitmap, we generate a bit string (outputs of our algorithm). For each bit string, we compute the statistic V by the following formula. R denote the number of runs, n denote the length of the bit string, p denote the rate of 1 in the bit string.

$$V = \frac{R - 2np(1-p)}{\sqrt{4np^2(1-p)^2}}$$

step2: Make a frequency table of V.

We make the frequency table of V . Each class of frequency table has a interval of probability 5 % in the standard normal distribution. Therefore, the probability of every class is 0.05 (5 %), and the expected value of every class is 5 ($= 100 \times 0.05 = np$, where n, p denote number of samples, probability).

step3: Compute chi-square by the frequency table and examine the chi-square test.

4.2 Poker Test

Poker test is a method of statistical test based on *poker game*. We collect 100 bitmaps of same type (for example, test201 - test300) and examine the randomness of output under the following procedure.

step1: Convert each bit string to a decimal number.

For each bitmap, we generate the bit string (output of our algorithm). From the leftmost bit of the string, we divide the string into shorter intervals, where each interval has the length 4. Each interval are converted to decimal number (0-15). We ignore the rightmost interval if the length of it is less than 4. Let an array a be the resulting array.

(example) 1001 1101 0011 0101 10 $\rightarrow$ 9 13 3 6 : array a

step2: Using a, we play poker.

Suppose that $a = a_0 a_2 \cdots a_n$ ($a_i \in \{0 \cdots 15\}, 0 \leq i \leq n$). From the leftmost charactor of the string a, we divide the string into shorter intervals, where each interval has the length 5. That is, we have $a_0 a_1 \cdots a_4, a_5 \cdots a_9, \cdots$, $a_{5j} \cdots a_n$. If the length from a_{5j} to a_n is less than 5, we ignore the interval $(a_j \cdots a_n)$. Let an array b_k be $b_k = a_{5k} \cdots a_{5k+4}$ ($0 \leq k \leq m$, $m = j$ or $j-1$).

By the number of different kinds in the set $\{a_{5k} \cdots a_{5k+4}\}$ of 5 intergers, we classfy $b_0, b_1, \cdots, b_m$ into 5 classes. Let $f_l (1 \leq l \leq 5)$ be the frequency of the class of l kinds. We compute $f_1, f_2, \cdots$, and f_5.

(example) $a=$ 5 3 2 3 11 9 12 3 3 12 13 14 3 11 8 2 3 1 2 7
15 13 15 15 13 1 2 3 4 5 2 2 2 4 13 4 12 5 7 10 12 3 4 1 15.

kinds of 5 integers	1 kind	2 kinds	3 kinds	4 kinds	5 kinds
frequency	0	1	2	2	4

$f_1 = 0, f_2 = 1, f_3 = 2, f_4 = 2, f_5 = 4$.

step3: Compute chi-square and examine the chi-square test.

Let $F_l (1 \leq l \leq 5)$ be the total of f_l among the hundred of bitmaps. Using $F_1, \cdots, F_5$, we compute the chi-square. Before computing the chi-square, we lump the class of 1 kind together the class of 2 kinds because it is expected that F_1 is very small.

4.3 Experimental Results

The following table is our experimental results of 4.1(RUN TEST) and 4.2(POKER TEST).

TEST :	RUN TEST		POKER TEST	
Algorithm :	KAWAL	rng023	KAWAL	rng023
File: test201 - test300	◯	◯	◯	◯
File: test301 - test400	◯	◯	◯	◯
File: test401 - test500	◯	◯	◯	△
File: test501 - test600	◯	◯	◯	◯
File: test601 - test700	×	×	×	×

◯ $\cdots$ At a significance level $\alpha = 0.05$, the bitmap set passes.
△ $\cdots$ At $\alpha = 0.05$ ($\alpha = 0.01$), the bitmap set fails (passes).
× $\cdots$ At $\alpha = 0.01$, the bitmap set fails.

5 Conclusive Remark

Our algorithms have good results for run test and poker test. We would like to challenge the following issues in our future work.

1. More precise statistical test.
2. More practical algorithm. In particular, we want to have enough long outputs.
3. Mathematical reserach on KAWAL. Although Suzuki's revised algorithm (rng023) has a mathematical proof that it preserves irregularity of freehand drawings, KAWAL dose not have such a proof yet.

目 次

第1章　序

1.1　概要

乱数生成器 (random number generator, 以下 RNG) は, 暗号技術やシミュレーション実験をはじめ, 多くの分野で利用されている. RNG より生成される**乱数**(random number) は, 実用に耐えられるだけのランダム性を求められる.

著者は, 指導教官であり共同研究者でもある鈴木登志雄講師 (以下, 鈴木講師) と, 手書き曲線から擬似乱数を抽出するアルゴリズムの考案と検証に取り掛かった (図 1.1 参照). そして, 著者が考案したアルゴリズムは, 擬似乱数のランダム性を検証する統計的検定 (連の検定, ポーカー検定と呼ばれる検定法を用いた) において優良な結果を得ることができた. さらに鈴木講師は, 著者が考案したアルゴリズムに一部修正を加え, その修正アルゴリズムが上記の統計的検定に合格しただけでなく乱数源 (手書き曲線) のランダム性を保つことを数学的に保証した. なお, 本論文の手書き曲線は, パソコンのペイントソフトを用いて描いた落書きをビットマップ形式で保存したものである. 以下, 手書き曲線のことを「落書きビットマップ」と呼ぶこともある.

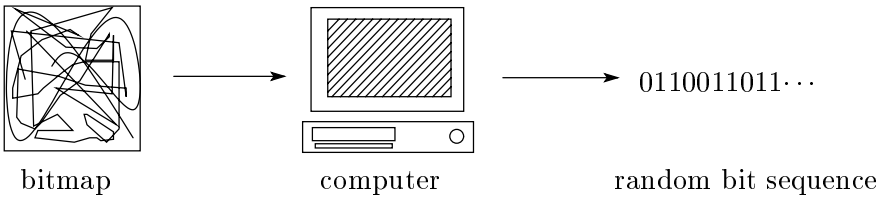


図 1.1: 手描き曲線からの乱数生成

ところで, 上記のような統計的検定においてある程度よい結果を示す擬似乱数を生成するだけならば, 一般的によく知られている線形合同法やフィードバックシフトレジスタのような乱数生成法でも実現可能である [Gent03, 津田 95, 宮武脇本 78, 脇本 73]. しかし, 統計学の観点からでなく計算量の観点から見た場合, これらの方法によって作られた擬似乱数は「よい乱数」であるとはいえない. というのも, これらの方法は短いプログラムで実装可能な一種の漸化式を用いており, 計算が複雑でない (回路計算量やコルモゴロフ計算量が低い) からである. 一方, 我々のアルゴリズムの乱数源である手書き曲線は, 手の動きを用いて描いたものであり, コンピュータの計算では作り出すことができない (回路計算量やコルモゴロフ計算量が十分高い) ものと期待できる. そのため, アルゴリズムより出力される擬似乱数が手書き曲線のもつランダム性を十分反映しているならば, 計算量の観点から見てもよい乱数であると考えられる.

また, 線形合同法に代表されるような, コンピュータの計算のみで乱数を発生させる方法以外に, 空気の乱流や半導体の熱ノイズのような物理的乱数源を用いて乱数を発生させる方法がある [斉藤他 02, 清水 05, 藤田他 03]. 物理的乱数源を用いて乱数を発生させる方法は, 企業や大学でさかんに研究されており, すでに実用化 (製品化) されているものもある. これらの方

法より出力される乱数は, 計算量の観点から見てもよい乱数であると期待できるが, 乱数を手に入れるためには特別なデバイスが必要になる. 我々の方法 (手書き曲線からの乱数生成) は, 上記の方法と比べ, 高価なデバイスを必要とせず (コンピュータとそのポインティングデバイスのみ), また乱数源を手軽に作ることができるところに利点がある.

修士論文の構成について説明する. 本論文は, 著者らが考案した乱数生成アルゴリズムの構成やその評価 · 検証を中心に述べたものである. 第 2 章では, 乱数源である落書きビットマップと, 落書きを描くのに必要な道具について紹介する. 第 3 章と第 4 章は, 本論文の主要部分である. 第 3 章では, 著者と鈴木講師が考案した乱数生成アルゴリズムについて詳細な説明をし, 第 4 章では, アルゴリズムより生成された擬似乱数のランダム性を検証する統計的検定 (連の検定, ポーカー検定) の内容やその実験結果について紹介する. 最後に第 5 章にて, 結びとして著者の意見をまとめる.

付録では, これまで紹介してきた手書き曲線から乱数を生成する方法ではなく, 写真から乱数を生成する方法を示す. また, その検定結果についても示す. なお, 付録 A では風景写真を乱数源として使用し, 付録 B では落葉の写真を乱数源として使用している.

1.2 理論的研究成果

著者のアルゴリズムによって生成された乱数が手描き曲線の不規則性を正しく反映しているかを調べるため, 鈴木講師は Dowd 型ジェネリック・オラクルを用いて「不規則な対象」と「不規則性を保つ写像」の数学的モ

デルを定義し，その性質を調べた．Dowd 型ジェネリック・オラクルは強制条件の最小サイズを用いて定義される．ここでは「任意の正の整数 r に対して Dowd の意味で r ジェネリックであるオラクル」を「不規則な対象」の数学的モデルとする．

鈴木講師は，著者のアルゴリズムを修正したアルゴリズムを作り，以下を示した．(1) 上記の修正版アルゴリズムは「不規則性を保つ写像」である．(2) 任意の自然数 m に対して，不規則なオラクル D であって，$|\{i \leq n : D(j) = 1\}|/n \to 1/2^m$ (if $n \to \infty$) となるものが存在する．(3) 任意の自然数 m に対して，不規則なオラクル D であって，$|\{i \leq n : D(j) = 1\}|/n \to 1 - 1/2^m$ (if $n \to \infty$) となるものが存在する．(4) 任意の Martin-Löf random oracle は不規則である．

この内容についての詳細は [川西鈴木 05] を参考されたい．

1.3 発表実績

我々はこの研究を以下の研究集会で発表した．京都大学数理解析研究所で行われた短期共同研究「証明論と計算論」(2005 年 1 月)．カナダの McMaster University で行われた国際研究集会 'Franco-Canadian Workshop on Combinatorial Algorithm (COMAL2005)'(2005 年 8 月)．首都大学東京で行われた研究集会「代数学と計算 (AC2005)」(2005 年 11 月)．また，この研究の経過報告書が京都大学数理解析研究所講究録 [川西鈴木 05] に掲載された．

1.4 統計の予備知識

本論文で用いられる統計的検定法について紹介する．ここでは, 「カイ２乗検定」「連の検定」「ポーカー検定」について紹介する．

1.4.1 カイ２乗検定

カイ２乗検定 (適合度検定) は, 理論上の分布と度数分布の適合度を検定する検定法である．例えば, 「あるサイコロを 60 回投げたときサイコロの各目の出る確率が 1/6 であるか」を検査するときに用いられる．この例を用いて, カイ２乗検定の手順について以下に説明する．

なお, カイ２乗検定 (適合度検定) についてさらに詳しく知りたければ, [Knuth98, 石井他 95, 脇本 70] が参考になるであろう．

STEP1: 仮説を設定する.

帰無仮説「サイコロの各目の出る確率は 1/6 である」を設定する．

STEP2: サイコロを 60 回投げ, 度数分布表を作る.

サイコロを 60 回投げる．そして, 表 1.1 のような度数分布表を作る．項目は, クラス, 実測度数, 母比率, 期待度数である．

表 1.1 について詳細説明をする．各 $i(i = 1, 2, \cdots, 6)$ に対して, i 番目のクラスはサイコロの目が i であるという事象を表す．したがって, 各クラスの母比率は 1/6, 期待度数は $10(= np$, 標本の数 $n = 60$, 確率 $p = 1/6)$ である．

サイコロの目	実測度数	母比率	期待度数
1	10	1/6	10
2	14	1/6	10
3	6	1/6	10
4	8	1/6	10
5	12	1/6	10
6	10	1/6	10

表 1.1: 度数分布表

STEP3: カイ２乗値を求める

度数分布表と以下の式を用いて, カイ 2 乗値 χ_0^2 を求める. 仮説の下で, χ_0^2 は, 自由度 $k-1$ のカイ 2 乗分布 (図 1.2 参照) に従う. ただし, k はクラスの総数 (例の場合 $k=6$), i はクラス番号 (サイコロの目), f_i は第 i クラスの実測度数とする. p_i は第 i クラスの母比率とする.

$$\chi_0^2 = \sum_{i=1}^{k} \frac{(f_i - np_i)^2}{np_i}$$

今回の場合, 表 1.1 より $\chi_0^2 = 4.0$ となる.

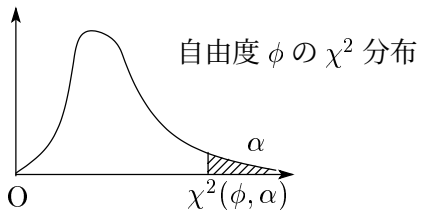


図 1.2: カイ 2 乗分布

STEP4: 検定を行う

有意水準 α で検定を行う. カイ 2 乗値 χ_0^2 が, $\chi^2(k-1,\alpha)$ より小さければ (大きければ), 仮説を棄却できない (棄却する). ただし, $\chi^2(k-1,\alpha)$ は, 自由度 $k-1$ のカイ 2 乗分布の上側確率 α 点である. なお, $\alpha = 0.05$ に設定するのが一般的である.

今回の場合, $\chi_0^2 = 4.0 < 11.07 = \chi^2(5, 0.05)$ より, 仮説を棄却できない.

1.4.2 連の検定

連の検定は, 擬似乱数 (今回の場合はビット列) の数字の並び方の無規則性を検証する検定法のひとつであり, その特徴として「連」の数に注目する. 「連」とは, 000000 のような同じ数字が続くひとつの列のことをいう (例えば, 010011100 なら連の数は 5). この検定は次のような考えに基づく. つまり, 連の数が極端に多い (逆に少ない) 場合は, 「0 がきたら次は 1 (次も 0) だろう」というように予測ができ, 与えられたビット列はランダムでないと考えられる.

連の検定の詳細は, 4.1.1 を参考されたい. さらに詳しく知りたければ, [脇本 70] が参考になるであろう.

1.4.3 ポーカー検定

ポーカー検定は, 連の検定と同様に, 擬似乱数の数字の並び方の無規則性を検証する検定法のひとつである. その方法は, 名の通りポーカーゲームに基づく.

ポーカー検定の詳細は, 4.2.1 を参考されたい. さらに詳しく知りたければ, [Knuth98, 伏見 89] が参考になるであろう.

1.5 参考文献の紹介

計算量理論の基礎事項については [Papa94, 田中 97] を，統計の基礎事項については [石井他 95] を，乱数の基礎事項については [Gent03, Knuth98, 津田 95, 伏見 89, 宮武脇本 78, 脇本 70] をそれぞれ参照されたい. また，物理的乱数源を用いた乱数発生方法については [斉藤他 02, 清水 05, 藤田他 03] が, BMP 形式の画像ファイルについては [宮坂 04] が，そして広く用いられている擬似乱数生成プログラムの問題点については [和田 04] が参考になるであろう.

第2章　落書きビットマップ

この章では, 我々のアルゴリズムの乱数源である落書きビットマップと, 落書きを描くのに必要な道具について説明する.

2.1　落書きを描くのに必要な道具

落書きを描く作業は, コンピュータ上で行う. 必要な道具は, コンピュータとペイントツール (ポインティングデバイスとペイントソフト) である.

コンピュータについて説明する. 本研究では, DELL 社のパソコン (Dimension 4600C) を用いる. なお, そのパソコンに搭載されている OS は Microsoft 社の 'WindowsXP ServicePack2' で, 同じく CPU は Intel 社の 'Pentium4 2.80GHz' である.

ペイントツールについて説明する. 使用するペイントツールは, 描きたい落書きによって異なる. 図 2.1, 図 2.2 のような落書きを描く場合, ポインティングデバイスは, マウスを選択し, ペイントソフトは, WINDOWS OS に標準装備されている「ペイント」を用いる. 「ペイント」のペンの太さは一定である. 一方, 図 2.3, 図 2.4 のような落書きを描く場合, ポインティングデバイスは, Wacom 社のペンタブレット (図 2.5) を選択し, ペイントソフトは, Corel 社の 'Corel Painter Essential 2'(図 2.6) を用い

る. ‘Corel Painter Essential 2’ のペンの太さは, タブレットにかかる筆圧によって変化する.

図 2.1: paint1

図 2.2: paint2

図 2.3: tablet1

図 2.4: tablet2

図 2.5: WACOM のペンタブレット

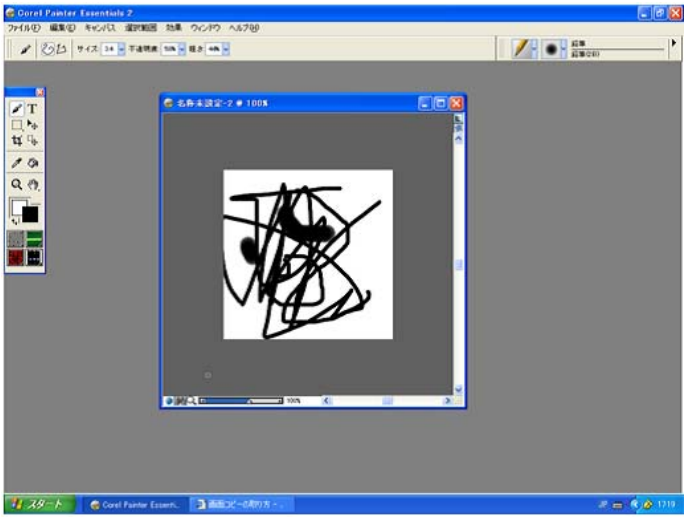


図 2.6: Corel Painter Essential 2

2.2 落書きビットマップについて

落書きビットマップのキャンバスのサイズは, 256 × 256 ピクセルに固定している. また, 落書きを描く際に, キャンバスの色は白, ペンの色は黒に固定している.

なお, 今回の研究で描いた落書きには, それぞれ特徴がある. それは, 無意味な曲線と漫画風の落書き, 著者が描いた落書きと鈴木講師が描いた落書き, マウスで描いた落書き (筆圧未対応) とペンタブレットで描いた落書き (筆圧対応) などである. 各々の落書きビットマップは約 1 分で書いた.

これらの落書きを分類し, 全部で 500 枚 (test201.bmp - test700.bmp) 用意した. 以下に詳細を示す.

2.2.1 マウスで描いた無意味な曲線

この 100 枚の落書きビットマップ (test201.bmp - test300.bmp) は, マウスで描いた無意味な曲線である. test201.bmp - test250.bmp は著者が, test251.bmp - test300.bmp は鈴木講師が描いた. 以下の図はその一部である.

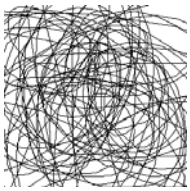


図 2.7: test201.bmp

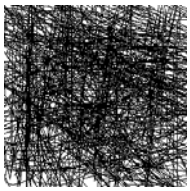


図 2.8: test204.bmp

図 2.9: test259.bmp

図 2.10: test297.bmp

2.2.2 ペンタブレットで描いた無意味な曲線

この 100 枚の落書きビットマップ (test301.bmp - test400.bmp) は, ペンタブレットで描いた無意味な曲線である. test301.bmp - test350.bmp は著者が, test351.bmp - test400.bmp は鈴木講師が描いた. 以下の図はその一部である.

図 2.11: test307.bmp

図 2.12: test330.bmp

図 2.13: test356.bmp

図 2.14: test391.bmp

2.2.3　マウスで描いた漫画風の落書き

この 100 枚の落書きビットマップ (test401.bmp - test500.bmp) は, マウスで描いた漫画風の落書きである. test401.bmp - test450.bmp は著者が, test451.bmp - test500.bmp は鈴木講師が描いた. 以下の図はその一部である.

図 2.15: test437.bmp

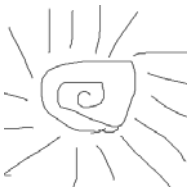


図 2.16: test439.bmp

図 2.17: test457.bmp

図 2.18: test481.bmp

2.2.4　ペンタブレットで描いた漫画風の落書き

この 100 枚の落書きビットマップ (test501.bmp - test600.bmp) は, ペンタブレットで描いた漫画風の落書きである. test501.bmp - test550.bmp は著者が, test551.bmp - test600.bmp は鈴木講師が描いた. 以下の図はその一部である.

図 2.19: test530.bmp

図 2.20: test540.bmp

図 2.21: test562.bmp

図 2.22: test585.bmp

2.2.5 幾何学的なデザイン

この 100 枚の落書きビットマップ (test601.bmp - test700.bmp) は, 幾何学的なデザインである. test601.bmp - test650.bmp は著者が, test651.bmp - test700.bmp は鈴木講師が描いた. 以下の図はその一部である.

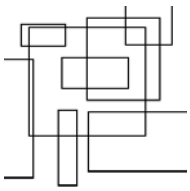


図 2.23: test620.bmp

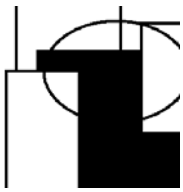


図 2.24: test635.bmp

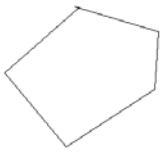


図 2.25: test657.bmp

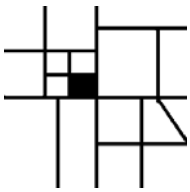


図 2.26: test687.bmp

2.2.6 まとめ

これまで紹介してきた落書きビットマップの特徴を表 2.1 で示す. これらの落書きビットマップは, 次節で紹介するアルゴリズムの乱数源となる.

Bitmap	Charactor
test201 - test300	マウスで描いた無意味な曲線
test301 - test400	ペンタブレットで描いた無意味な曲線
test401 - test500	マウスで描いた漫画風の落書き
test501 - test600	ペンタブレットで描いた漫画風の落書き
test601 - test700	幾何学的なデザイン

表 2.1: 落書きビットマップの特徴

第3章　乱数生成アルゴリズム

この章では, 落書きビットマップから擬似乱数を生成するためのアルゴリズムについて紹介する. 紹介するアルゴリズムは, 著者が考案したアルゴリズム (KAWAL アルゴリズム, 又は KAWAL) と, 鈴木講師が KAWAL に修正を加えたアルゴリズム (rng023) の 2 つである.

プログラミング言語 C++ を用いてアルゴリズム KAWAL, rng023 を実装した. これらのプログラムを用いて落書きビットマップから擬似乱数を生成する.

本章の構成について説明する. 最初に落書きビットマップを数値化する (正方行列に変換する) 方法について紹介する. 生成された行列は, KAWAL と rng023 で処理される. その後, 上記の 2 つのアルゴリズムについて説明する.

3.1　落書きビットマップを行列 A に変換する

前章で紹介した, ペイントソフトを用いて描かれた落書きビットマップ (キャンバスのサイズは 256×256 ピクセル) を正方行列 A に変換する. また, ビットマップの上から i 番目, 左から j 番目のピクセルは, 行列の要素 $a_{i,j}$ に対応する. 要素 $a_{i,j}$ には, 対応するピクセルに色がついていないと

きは２進数「０」, 対応するピクセルに色がついているときは２進数「１」を対応させる. 詳しくは, 図 3.1 を参考にされたい.

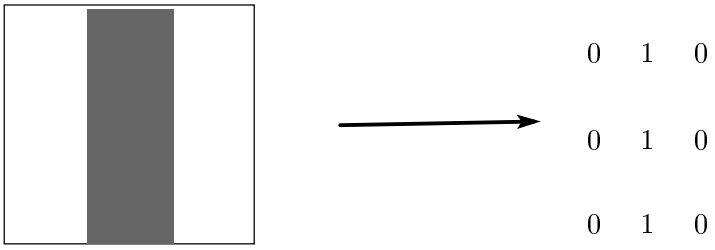


図 3.1: サイズが 3×3 ピクセルの落書きビットマップを行列 A に変換

3.2 KAWAL アルゴリズム

著者が考案した乱数生成アルゴリズム, KAWAL(カワル) について紹介する. KAWAL という名前の由来は, 'Kawanishi Algorithm' の略からである. 本節では, KAWAL を開発した動機と KAWAL の構成について説明を行う.

3.2.1 KAWAL を開発した動機

行列 A のような落書きビットマップをそのまま行列化したものは, 擬似乱数としての使用に適さない. というのも, 行列 A には,

1. 意識的に落書きを描かない限り, 1 の割合が 50 パーセント前後にならない.
2. ビットマップの四隅は空白になりやすく, その部分には長い連が現れる. 連とは 00000 のように同じ数字が並ぶ列を指す.

という明らかな弱点があるからである.

そのため上の弱点を克服するアルゴリズムを開発する必要があった. KAWAL は, このような落書きビットマップがもつ弱点の克服を目指し考案された乱数生成アルゴリズムである.

3.2.2 KAWAL の構成

KAWAL の特徴は, 連の長さの 2 進数化と排他的論理和 (XOR) に注目したところである. 行列 A から擬似乱数を生成するまでの手順は, 以下の通りである.

1. 行列 A を一次元し, 出来上がったものを配列 la に格納する
2. 配列 la の連を 2 進数化し, 出来上がったものを配列 be に格納する
3. 配列 be に XOR を適用し, 出来上がったものを配列 myXor に格納する
4. 配列 myXor を出力する (KAWAL 終わり)

ここでいう配列とは, すべて一次元配列のことを指す. これより KAWAL アルゴリズムについて, 上の手順の詳細な説明を行う.

STEP1: 行列 A を一次元化する

英文を読むように, 行列 A を一次元化する. 一番上の行を左から 1 ビットずつ読んで一次元配列 la に格納していき, 次に上から 2 番目の行に対しても同じことを行い, 上から 3 番目以下も同様にしていく (la は, 'linear array' の略). 詳しくは, 図 3.2 を参考にされたい.

図 3.2: 行列 A を一次元化する

STEP2: 配列 la の連の長さを２進数化する

ここでは, 例を示しながら手順を説明する.

例　配列 la : 1 1 0 0 0 1 1 1 0 0 0 0 0

1. 連の長さは左から 2,3,3,5 なので, 連を２進数にすると, それぞれ 1,11,11,101. [注]
2. それぞれの２進数の一番大きい位, つまり最上位ビットの 1 を削除すると 1,1,1,01. ただし, ビット列の長さが 1 の場合は, この項目を無視する.
3. それぞれを左右逆転（ＡＢＣ → ＣＢＡ）し, 一次元配列 be に格納する (be は, 'binary encoding' の略). このとき, 配列 be : 1 1 1 1 0.
4. 配列 be を出力する.

[注] 連の長さが１, ２の場合は例外である. まず, 連の長さが２の場合は, ２進数「１」を対応させる. 連の長さが１の場合は, 配列 la の番地（配列の左から i 番目を i 番地とする) によって区別し, 番地が偶数であれば０, 奇数であれば１を対応させる.

STEP3: 配列 be に XOR を適用する

配列 be を XOR(排他的論理和) を用いて加工し, 新たな配列 myXor を作る. その作り方は以下の通りである.

各 $i(1 \leq i \leq (1/2)$(配列 be の長さ)) に対し, 配列 be の先頭から i 番目（i は正の整数）の要素と, 配列 be の後ろから i 番目の要素の XOR をとり, 新しい配列 myXOR の i 番目に格納する. 図 3.3 はその一例である.

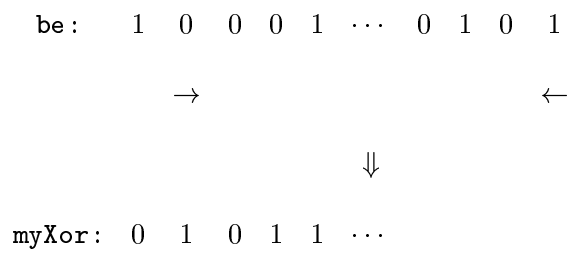


図 3.3: 配列 be に XOR を適用する

STEP4: 配列 myXor を出力する

配列 myXor を出力する (KAWAL 終わり).

3.3 rng023

次に, 鈴木講師が考案した乱数生成アルゴリズム, rng023 について説明する (rng は, 'random number generator'の略である). rng023 は, KAWAL に一部修正を加えたものである. KAWAL と違い, rng023 より出力されるビット列の長さは一定である. 行列 A から擬似乱数を生成するまでの手順は, 以下の通りである.

1. 行列 A を一次元し, 出来上がったものを配列 la に格納する
2. 配列 la の連を 2 進数化し, 出来上がったものを配列 be に格納する (ただし, KAWAL とはアルゴリズムが異なる).
3. 配列 be に XOR を用い, 出来上がったものを配列 myXor に格納する
4. 配列 myXor を出力する (rng023 終わり)

rng023 について, KAWAL の時と同様に, 上の手順の詳細な説明を行う.

STEP1: 行列 A を一次元化する

3.2.2 の `STEP1` と同様の操作を行う

STEP2: 配列 la を 2 進数化する

まず, 正の整数値 C_1, C_2 を決定し, $C_3 = \mathrm{floor}(\log_2(C_1 + C_2))$ とおく. ただし, $\mathrm{floor}(x)$(x は正の実数) は, x 以下の最大の整数を意味する (例えば, $x = 4.2$ ならば, $\mathrm{floor}(x)$ は, 4 となる). 本稿では, $C_1 = 60$, $C_2 = 2$, $C_3 = \mathrm{floor}(\log_2(60 + 2)) = 5$, とする.

配列 la: $\mathrm{la}_0\ \mathrm{la}_1 \cdots \mathrm{la}_{\mathrm{c}_1}\ \mathrm{la}_{\mathrm{c}_1+1} \cdots$ とし, 以下に手順を説明する. ただし, la_k $(0 \leq k \leq 256 \times 256 - 1)$ は, 0,1 のいずれかである.

1. 配列 la を 一番左 (先頭ビット) から C_1 ビットずつ区切り, 新たに配列 $\mathrm{la}^{(0)} = \mathrm{la}_0 \cdots \mathrm{la}_{\mathrm{c}_1-1}$, $\mathrm{la}^{(1)} = \mathrm{la}_{\mathrm{c}_1} \cdots \mathrm{la}_{\mathrm{c}_1\times 2-1}$, $\cdots$, $\mathrm{la}^{(\mathrm{i})} = \mathrm{la}_{\mathrm{c}_1\times \mathrm{i}} \cdots \mathrm{la}_{\mathrm{c}_1\times(\mathrm{i}+1)-1}$, $\cdots$ を作る. ただし, 最後の配列の長さが, C_1 ビットに満たない場合は, それを無視する.

2. 配列 $\mathrm{la}^{(\mathrm{i})}$ において, 各々の連の長さに C_2 を加えて 2 進数化する. この操作は, 若干複雑なので, 例を用いて示す.

   例　$\mathrm{la}^{(0)}$: 1 1 0 0 0 1 1 1 0 0 0 0 0

   (a) 連の長さは左から 2,3,3,5 なので, $C_2 = 2$ を加え, 2 進数にすると, それぞれ 100,101,101,111.

   (b) それぞれの 2 進数の一番大きい位, つまり最上位ビットの 1 を削除すると 00, 01, 01, 11.

   (c) 出来上がったビット列の最初の C_3 桁を抜き出し, それを $\mathrm{be}^{(0)}$ とする. 今回の場合, $C_3 = 5$ より, $\mathrm{be}^{(0)}$: 0 0 0 1 0.

   (d) $\mathrm{la}^{(1)}$ 以降も同様の作業を繰り返し, $\mathrm{be}^{(1)}$, $\mathrm{be}^{(2)}$, $\cdots$ を求める.

3. $\mathrm{be}^{(0)}$, $\mathrm{be}^{(1)}$, $\cdots$ を連結し, 配列 be: $\mathrm{be}^{(0)}$ $\mathrm{be}^{(1)}$ $\cdots$ を作る.

4. 配列 be を出力する.

STEP3: 配列 be に XOR を用いる

3.2.2 の STEP3 と同様の操作を行う

STEP4: 配列 myXor を出力する

配列 myXor を出力する (rng023 終わり).

3.4　行列 A, KAWAL, rng023 の出力値の調査

test201 から test700 までの落書きビットマップ (第 2 節を参照) を用い, 行列 A と KAWAL と rng023 の出力値 (ビット列) を調査した. その

結果, KAWAL, rng023 の出力値ともに, 多くのビットマップにおいて 1 の割合は 50 パーセント前後の値を示した. 以下の表は, 同じ特徴をもつ落書き (例:マウスで描いた無意味な曲線の落書き) ごとに分け, 出力値の１の割合の平均値などを求めた結果である. なお, 出力値のランダム性の検証に関しては, 次の章で詳しく説明する.

	行列 A	KAWAL	rng023
１の割合の平均値	0.300	0.497	0.500
１の割合の分散	0.0133	3.90×10^{-5}	8.47×10^{-5}
出力の長さの平均値	65536(一定)	12562	2730（一定）
出力の長さの分散	0	3.267×10^{6}	0

表 3.1: test201-test300

	行列 A	KAWAL	rng023
１の割合の平均値	0.620	0.495	0.497
１の割合の分散	9.33×10^{-3}	5.34×10^{-5}	9.15×10^{-5}
出力の長さの平均値	65536(一定)	5951	2730（一定）
出力の長さの分散	0	2.34×10^{6}	0

表 3.2: test301-test400

	行列 A	KAWAL	rng023
1 の割合の平均値	0.0574	0.500	0.480
1 の割合の分散	1.28×10^{-3}	7.22×10^{-5}	2.29×10^{-4}
出力の長さの平均値	65536(一定)	4423	2730（一定）
出力の長さの分散	0	2.69×10^{6}	0

表 3.3: test401-test500

	行列 A	KAWAL	rng023
1 の割合の平均値	0.356	0.496	0.497
1 の割合の分散	7.74×10^{-3}	5.04×10^{-5}	1.36×10^{-4}
出力の長さの平均値	65536(一定)	4847	2730（一定）
出力の長さの分散	0	1.20×10^{6}	0

表 3.4: test501-test600

	行列 A	KAWAL	rng023
1 の割合の平均値	0.140	0.496	0.469
1 の割合の分散	0.0201	0.00198	0.00114
出力の長さの平均値	65536(一定)	2895	2730（一定）
出力の長さの分散	0	2.17×10^{6}	0

表 3.5: test601-test700

第4章　統計的検定

この章では, 擬似乱数のランダム性を検証するために用いた統計的検定法とその実験結果について論じる. ここでいう擬似乱数とは, 前節のアルゴリズムより出力された 0 と 1 のビット列のことである.

本稿では,「連の検定」,「ポーカー検定」と呼ばれる統計的検定法を用い, 実験を行った. 本章の話の進め方として, 2つの検定法とも, まず検定の内容について説明した後, 検定の操作手順, そして実験の結果について述べる. 最後に, 2つの検定の結果をまとめている.

4.1　連の検定

ここでは, 連の検定についての説明を行う.

4.1.1　連の検定について

連の検定とは, 擬似乱数 (今回の場合はビット列) の数字の並び方の無規則性を検証する検定法のひとつであり, その特徴として「連」の数に注目する. 「連」とは, 前節でも紹介したが, 00000 のような同じ数字が続くひとつの列のことをいう (例えば, 010011100 なら連の数は5). この検定は次のような考えに基づく. つまり, 連の数が極端に多い (逆に少ない) 場合

は, 「0 がきたら次は 1 (次も 0) だろう」というように予測ができ, 与えられたビット列はランダムでないと考えられる.

ところで, 『ビット列が十分長く, 各ビットの並びが無規則』であるとき, 連の数 R は,

平均値: $2np(1-p)$, 分散: $4np^2(1-p)^2$

(n は, ビット列の長さ. p は, ビット列に含まれる 1 の比率.)

の正規分布に近似的に従うことが知られている [脇本 70, pp.56-58].

そこで, 下の式を用いて統計量 V を定める. 『』部分の仮説のもとで, V は標準正規分布 (平均値 0 , 分散 1 の正規分布) に近似的に従う.

$$V = \frac{R - 2np(1-p)}{\sqrt{4np^2(1-p)^2}}$$

今回, 求めた統計量 V が連の検定に合格しているかどうかを判定する基準として, $|V| < K_{\alpha/2}$ のとき, 本検定に合格とする. ただし, α は有意水準, $K_{\alpha/2}$ は 標準正規分布の上側確率 $\alpha/2$ 点 である. なお, 本検定では $\alpha = 0.05$ と固定する. したがって, $K_{\alpha/2} = 1.96$ である.

4.1.2 連の検定による実験結果

サンプルとして, test201 から test700 までの落書きビットマップ (第 2 節を参照) を用い, KAWAL と rng023 の出力値に連の検定を実行した. 28 ページの表は, 同じ特徴をもつ落書き (例:マウスで描いた無意味な曲線の落書き) ごとに分け, 統計量 V の平均と分散を求めた結果である. また, 26 ページでは KAWAL の統計量 V のヒストグラムを, 27 ページでは rng023 の統計量 V のヒストグラムをそれぞれ示す.

KAWAL の結果 (V のヒストグラム)

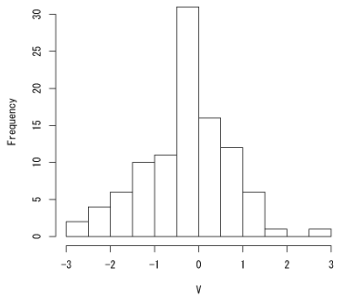


図 4.1: test201-300

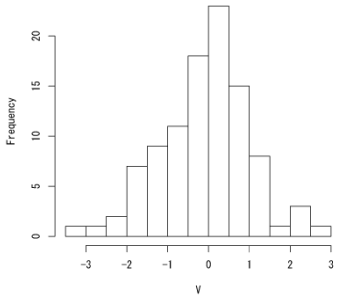


図 4.2: test301-400

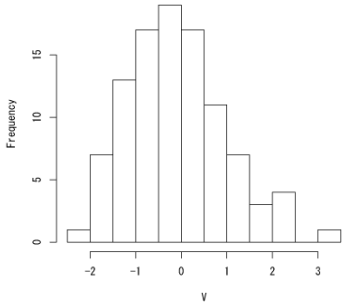


図 4.3: test401-500

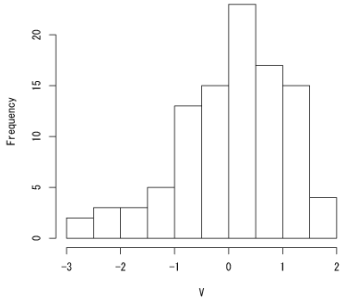


図 4.4: test501-600

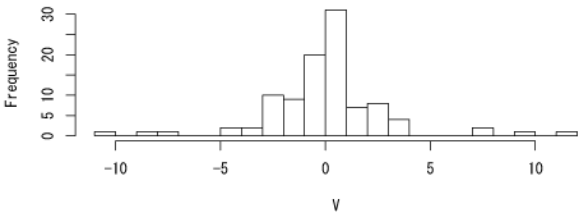


図 4.5: test601-700

rng023 の結果 (V のヒストグラム)

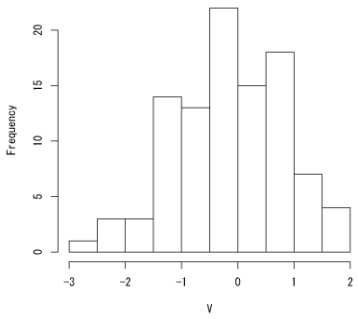


図 4.6: test201-300

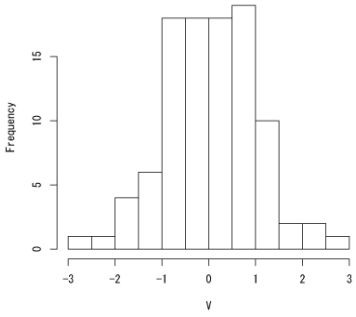


図 4.7: test301-400

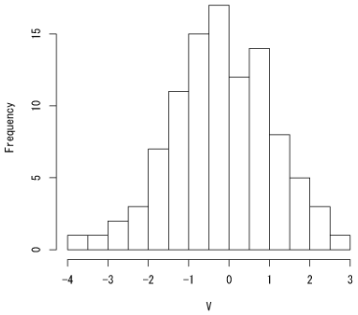


図 4.8: test401-500

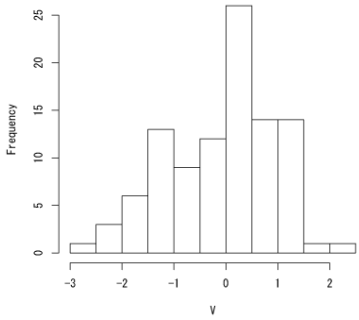


図 4.9: test501-600

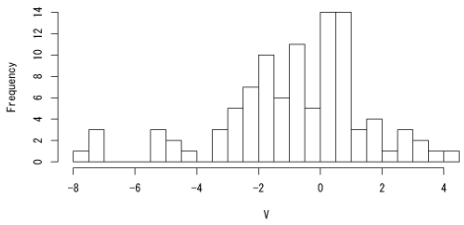


図 4.10: test601-700

	V の平均	V の分散
File: test201 - test300	-0.295	0.995
File: test301 - test400	-0.101	0.963
File: test401 - test500	-0.112	1.067
File: test501 - test600	-0.0507	1.090
File: test601 - test700	0.0305	8.540

表 4.1: KAWAL の V の平均と分散

	V の平均	V の分散
File: test201 - test300	-0.140	0.952
File: test301 - test400	0.00954	1.015
File: test401 - test500	-0.1858	1.528
File: test501 - test600	-0.0853	0.998
File: test601 - test700	-0.844	5.466

表 4.2: rng023 の V の平均と分散

実験結果の検証

求めた統計量 V の平均が連の検定に合格するかどうか判定する基準として, $|V| < 1.96$ ($= K_{\alpha/2}$. 標準正規分布の有意水準 $\alpha = 0.05$) のとき, 本検定に合格とする. 上の表の結果によると, すべての場合において合格となっている. しかし, この検定結果だけで擬似乱数の精度を検証するには不十分である.

ところで, KAWAL(rng023) の出力値 (擬似乱数列) が十分無規則であれば, V の分布と標準正規分布の適合度が高いと考えられる. そこで, V の分布が標準正規分布に近い形をしているか検証するために, カイ 2 乗検定 (適合度検定) を用いる. カイ 2 乗検定については, 次に詳細な説明をする.

4.1.3 統計量 V を用いてカイ 2 乗検定を行う

統計量 V を用いてカイ 2 乗検定を行う. これは, V の分布が標準正規分布に近い形をしているか検証するためである. 同じ特徴をもつ落書きビットマップ (例:マウスで描いた無意味な曲線の落書き) を 100 枚集め (例:test201-300), 以下の操作手順で検定を行う.

STEP1: 統計量 V を求める

集めた落書きビットマップより各々の統計量 V を求める.

STEP2: 統計量 V のクラス分けを行う

STEP1 より求めた各統計量 V について, 30 ページのような度数分布表を作る.

度数分布表について詳細説明をする. 各クラスはいずれも, 標準正規分布の全面積の 5 % 部分の区間 (interval) である. 各区間の数値は, 小数点第四位で四捨五入されている. したがって, 各クラスの確率 (probability) は 0.05, 期待度数 (expectation) は $5(= np$, 標本の数 $n = 100$, 確率 $p = 0.05)$ である.

interval	frequency	probability	expectation
$\sim$ -1.645	0	0.05	5
-1.645 $\sim$ -1.282	4	0.05	5
-1.282 $\sim$ -1.036	0	0.05	5
-1.036 $\sim$ -0.842	8	0.05	5
-0.842 $\sim$ -0.674	6	0.05	5
-0.674 $\sim$ -0.524	8	0.05	5
-0.524 $\sim$ -0.385	7	0.05	5
-0.385 $\sim$ -0.253	3	0.05	5
-0.253 $\sim$ -0.126	4	0.05	5
-0.126 $\sim$ 0	4	0.05	5
0 $\sim$ 0.126	6	0.05	5
0.126 $\sim$ 0.253	12	0.05	5
0.253 $\sim$ 0.385	7	0.05	5
0.385 $\sim$ 0.524	4	0.05	5
0.524 $\sim$ 0.674	3	0.05	5
0.674 $\sim$ 0.842	10	0.05	5
0.842 $\sim$ 1.036	3	0.05	5
1.036 $\sim$ 1.282	4	0.05	5
1.282 $\sim$ 1.645	5	0.05	5
1.645 $\sim$	2	0.05	5

表 4.3: 度数分布表

STEP3: カイ２乗値を求める

度数分布表と以下の式を用いて, カイ 2 乗値 χ_0^2 を求める. ただし, k はクラスの総数 (今回の場合 $k = 20$), i はクラスの番号 (以後, 度数分布表の上から i 番目のクラスを第 i クラスとする), f_i は第 i クラスの実測度数とする.

$$\chi_0^2 = \sum_{i=1}^{k} \frac{(f_i - np)^2}{np}$$

STEP4: カイ２乗検定を行う

有意水準 α でカイ 2 乗検定を行う. カイ 2 乗値 χ_0^2 が, $\chi^2(k-1, \alpha)$ より小さければ $(\chi_0^2 < \chi^2(k-1, \alpha))$, 本検定に合格とする. ただし, $\chi^2(k-1, \alpha)$ は, 自由度 $k-1$ のカイ 2 乗分布の上側確率 α 点である. なお, 本検定では $\alpha = 0.05$ に固定する.

4.1.4 4.1.3 の実験結果

サンプルとして, test201 から test700 までの落書きビットマップを用い, KAWAL アルゴリズムと rng023 の出力値に 4.1.3 の実験をした. 表 4.4 は, サンプルを 100 枚 (同じ特徴をもつ落書き) ごとに分け, それぞれカイ 2 乗値を求めた結果である.

	KAWAL	rng023
File: test201 - test300	29.2	10.8
File: test301 - test400	14.8	6
File: test401 - test500	16.8	27.2
File: test501 - test600	12.8	20.4
File: test601 - test700	102.4	182

表 4.4: カイ 2 乗値

実験結果の検証

求めたカイ 2 乗値 χ_0^2 がカイ 2 乗検定に合格するかどうか判定する基準として, $\chi_0^2 < 30.1$ ($= \chi^2(19, 0.05)$. カイ 2 乗分布の有意水準 $\alpha = 0.05$, 自由度 19) のとき, 本検定に合格とする. 表の結果によると, KAWAL アルゴリズムを用いたときの test601-test700 (case1) の場合, rng023 を用いたときの test601-test700 (case2) の場合以外は, すべて合格となっている.

次に, 不合格となった 2 つの出力値に関して検証してみる. case1 と case2 は, カイ 2 乗検定において不合格となっただけでなく, 求めたカイ 2 乗値は, それぞれ $\chi_0^2 = 102.4, \chi_0^2 = 182$ と 30.1 より大きく離れている (例えば, $\alpha = 0.000001$, 自由度 19 のカイ 2 乗値は, 63.22 である). 従って, 出力値が無規則であるとはいえない.

4.2 ポーカー検定

ここでは, ポーカー検定についての説明を行う.

なお, 4.2.1 を書く際に, [伏見 89, pp.115-118] を参考にした.

4.2.1 ポーカー検定について

ポーカー検定とは, M.G.Kendall と B.Babington-Smith によって提案された統計的検定法のひとつである. ゲームのポーカーの手になぞらえて, 整数の５個組の取るパターンを以下の７つに分類する. ただし, 数字の組み合わせに注目し, 順列は問わない. また, 異なる文字は異なる数字を意味する.

1. abcde (すべて異なる数)
2. abcdd (ワンペア)
3. abbcc (ツーペア)
4. abccc (３カード)
5. aabbb (フルハウス)
6. abbbb (４カード)
7. aaaaa (５カード)

後に, J.C.Butcher は, クラス分けをさらに容易にするために, ３と４, ５と６を合併し, ５つのクラスに分けることを提案した. 要するに, ひとつの組に相異なる数字がいくつあるかで分類するのである. 以後, Butcher の方法を採用し, 話を進めていく.

数列から, ５個の整数 (１から $d-1$ までのいずれかを取る) の組を n 個取り出し, 各組の中に何種類の整数が含まれているかを数える. つまり, それぞれのクラスの度数を求める. ５個組の中に i 種類の整数が含まれる

確率 (probability) を p_i とおくと, p_i は以下のように表すことができる.

$$
\begin{aligned}
p_1 &= \frac{d}{d^k}\, a_1 \\
p_2 &= \frac{d(d-1)}{d^k}\, a_2 \\
&\vdots \\
p_5 &= \frac{d(d-1)(d-2)(d-3)(d-4)}{d^k}\, a_5
\end{aligned}
$$

$a_1, \cdots, a_5$ は, 第 2 種スターリング数 (表 4.5 を参照) と呼ばれるものである. これは, 相異なる k 個の要素からなる集合をちょうど i 個の空でない部分集合に分割する仕方の数になっている. 例えば $k = 5$ のとき, $a_1 = 1$, $a_2 = 15$, $a_3 = 25$, $a_4 = 10$, $a_5 = 1$ となる.

最後に, p_i と 5 つのクラスの度数を用いて, カイ 2 乗検定を行う. ただし, p_1 が十分小さいので, $np_1 < 5$ のとき, p_1 と p_2 のクラスを合併することが望ましい.

	$i=1$	$i=2$	$i=3$	$i=4$	$i=5$	$i=6$	$i=7$
$k=1$	1						
$k=2$	1	1					
$k=3$	1	3	1				
$k=4$	1	7	6	1			
$k=5$	1	15	25	10	1		
$k=6$	1	31	90	65	15	1	
$k=7$	1	63	301	350	140	21	1

表 4.5: 第２種スターリング数 a_i

4.2.2 今回用いたポーカー検定について

今回用いたポーカー検定について説明する. 同じ特徴をもつ落書きビットマップを 100 枚集め, 以下の操作手順で検定を行う.

STEP1: ビット列を 10 進数に変換する

100 個の落書きビットマップから生成された擬似乱数 (ビット列) の各々を 10 進数の配列に変換する. ビット列を左から４ビットずつ区切り, 一組ごとに１０進数化 (0-15) する. 10 進数に変換したものから順に, 一次元配列 a に左詰めに格納していく. こうして, 100 個の配列 を得る.

(例) 1001 1101 0011 0101 10 → 9 13 3 6 : 配列 a

(注) ビット列の右端の一組の長さが４ビットに満たない場合は, 上例のようにそれを無視する.

STEP2: 配列 a を用いてポーカーを行う

配列 a を用いて前節で紹介したポーカーを行う．その方法は, Butcher の方法と同様である．

1. まず配列 a の左端から順に, 整数を５個ずつ取り出す.
2. 取り出した５個の整数を１組とし, その中の整数の種類の数によって, クラス分けを行う．クラスの数は, ５つである．ただし, 最終組 (a の右端の組) の整数の数が５個に満たない場合は, それを無視する.
3. 度数を表す配列 f を用意し, 各クラスの度数を求める．f_i は, 一つのクラス (i は, ５個組の整数の種類) の度数である.

こうして, 各 $i(i = 1, 2, \cdots 5)$ に対して, f_i の値 100 個を得る.

(例) $a = \{$ 1 3 3 7 9　3 8 11 13 14　1 2 2 3 7
9 9 15 15 15　1 2 3 4 5　2 2 2 4 13　4 5 7 10 12　1 3 4 12 13
2 4 10 11 11　6 7 11 14 15　5 5 8 12 14　1 2 3 6 8　3 3 10 $\}$

kinds of 5cards	1kind	2kinds	3kinds	4kinds	5kinds
frequency	0	1	1	4	6

$f_1 = 0, f_2 = 1, f_3 = 1, f_4 = 4, f_5 = 6.$

STEP3: カイ２乗値を求める

各 $i(i = 1, 2, \cdots 5)$ に対して, 100 個の f_i の総和 F_i を求める．F_1 から F_5 を用いて, カイ２乗値を求める．ただし, F_1 のクラスの期待値が十分小さいので, F_2 のクラスと合併する．ゆえに, カイ２乗分布の自由度は３となる.

カイ２乗値 χ_0^2 を求める数式は以下の通りである．ただし, p_i は F_i のクラスの母比率, $n = F_1 + F_2 + \cdots + F_5$.

$$\begin{aligned}\chi_0^2 = & \frac{(F_1 + F_2 - n(p_1 + p_2))^2}{n(p_1 + p_2)} \\ & + \frac{(F_3 - np_3)^2}{np_3} \\ & + \frac{(F_4 - np_4)^2}{np_4} \\ & + \frac{(F_5 - np_5)^2}{np_5}\end{aligned}$$

STEP4: カイ２乗検定を行う

有意水準 α でカイ２乗検定を行う．カイ２乗値 χ_0^2 が, $\chi^2(3, \alpha)$ より小さければ $(\chi_0^2 < \chi^2(3, \alpha))$, 本検定に合格とする．ただし, $\chi^2(3, \alpha)$ は, 自由度３のカイ２乗分布の上側確率 α 点である．なお, 本検定では $\alpha = 0.05$ に固定する．

4.2.3 実験結果

サンプルとして, test201 から test700 までの落書きビットマップを用い, KAWAL と rng023 の出力値を元にポーカー検定を実行した．以下の表は, サンプルを 100 枚 (同じ特徴をもつ落書き) ごとに分け, それぞれカイ２乗値を求めた結果である．KAWAL の結果を表 4.6 と表 4.7 に, rng023 の結果を表 4.8 と表 4.9 にそれぞれ示す.

FileName: test201 - test300

kinds of 5cards	1kind	2kinds	3kinds	4kinds	5kinds
frequency	1	228	5127	26208	31197
probability	0.00002	0.0034	0.0801	0.4166	0.4999
expectation	0.9577	215.47	5027.70	26144.03	31372.84

total of frequency = 62761, value of chi-square = 3.8333

FileName: test301 - test400

kinds of 5cards	1kind	2kinds	3kinds	4kinds	5kinds
frequency	3	101	2380	12283	14939
probability	0.00002	0.0034	0.0801	0.4166	0.4999
expectation	0.4533	101.99	2379.71	12374.48	14849.37

total of frequency = 29706, value of chi-square = 1.2410

FileName: test401 - test500

kinds of 5cards	1kind	2kinds	3kinds	4kinds	5kinds
frequency	0	82	1795	9076	11112
probability	0.00002	0.0034	0.0801	0.4166	0.4999
expectation	0.3367	75.75	1767.60	9191.51	11029.81

total of frequency = 22065, value of chi-square = 2.9477

表 4.6: KAWAL の結果 (test201-test500)

FileName: test501 - test600

kinds of 5cards	1kind	2kinds	3kinds	4kinds	5kinds
frequency	0	95	1958	9989	12146
probability	0.00002	0.0034	0.0801	0.4166	0.4999
expectation	0.3691	83.04	1937.67	10075.87	12091.05

total of frequency = 24188, value of chi-square = 2.8220

FileName: test601 - test700

kinds of 5cards	1kind	2kinds	3kinds	4kinds	5kinds
frequency	0	106	1314	6043	6961
probability	0.00002	0.0034	0.0801	0.4166	0.4999
expectation	0.2201	49.52	1155.49	6008.53	7210.24

total of frequency = 14424, value of chi-square = 94.1898

表 4.7: KAWAL の結果 (test501-test700)

FileName: test201 - test300

kinds of 5cards	1kind	2kinds	3kinds	4kinds	5kinds
frequency	0	43	1080	5667	6810
probability	0.00002	0.0034	0.0801	0.4166	0.4999
expectation	0.2075	46.69	1089.48	5665.28	6798.34

total of frequency = 13600, value of chi-square = 0.4272

FileName: test301 - test400

kinds of 5cards	1kind	2kinds	3kinds	4kinds	5kinds
frequency	1	51	1112	5686	6750
probability	0.00002	0.0034	0.0801	0.4166	0.4999
expectation	0.2075	46.69	1089.48	5665.28	6798.34

total of frequency = 13600, value of chi-square = 1.4398

FileName: test401 - test500

kinds of 5cards	1kind	2kinds	3kinds	4kinds	5kinds
frequency	0	34	1093	5545	6928
probability	0.00002	0.0034	0.0801	0.4166	0.4999
expectation	0.2075	46.69	1089.48	5665.28	6798.34

total of frequency = 13600, value of chi-square = 8.5860

表 4.8: rng023 の結果 (test201-test500)

FileName: test501 - test600

kinds of 5cards	1kind	2kinds	3kinds	4kinds	5kinds
frequency	0	53	1125	5629	6793
probability	0.00002	0.0034	0.0801	0.4166	0.4999
expectation	0.2075	46.69	1089.48	5665.28	6798.34

total of frequency = 13600, value of chi-square = 2.1883

FileName: test601 - test700

kinds of 5cards	1kind	2kinds	3kinds	4kinds	5kinds
frequency	0	45	1068	5060	7427
probability	0.00002	0.0034	0.0801	0.4166	0.4999
expectation	0.2075	46.69	1089.48	5665.28	6798.34

total of frequency = 13600, value of chi-square = 123.3031

表 4.9: rng023 の結果 (test501-test700)

実験結果の検証

求めたカイ２乗値 χ_0^2 がポーカー検定に合格するかどうか判定する基準として, $\chi_0^2 < 7.81$ ($= \chi^2(3, \alpha)$. カイ２乗分布の有意水準 $\alpha = 0.05$, 自由度３) のとき, 本検定に合格とする.

上の表の結果によると, KAWAL アルゴリズムを用いたときの test601-test700 (case1) の場合, rng023 を用いたときの test401-test500 (case2) と test601-test700 (case3) の場合以外は, すべて合格となっている. 次に, 不合格となった３つの出力値に関して検証してみる.

case2 は, 有意水準 $\alpha = 0.05$ でのカイ２乗検定においては不合格となったが, $\alpha = 0.035$ でのカイ２乗検定においては合格となり ($\chi_0^2 = 8.586 < 8.61 = \chi^2(3, 0.035)$), 7.81 より大きく離れていない. 一方, case1 と case3 は, ポーカー検定において不合格となっただけでなく, 求めたカイ２乗値は, それぞれ $\chi_0^2 = 94.1898, \chi_0^2 = 123.3031$ と 7.81 より大きく離れている (例えば, $\alpha = 0.000001$, 自由度３のカイ２乗値は, 30.48 である). 従って, 出力値が無規則であるとはいえない.

4.3 まとめ

ここでは, 連の検定とポーカー検定を実行した結果をまとめて示す. 次の表は, 4.1.4(RUN TEST) と 4.2.3(POKER TEST) の検定結果を簡潔にまとめたものである.

表 4.10 をみると, KAWAL を用いたときの test601-test700 の場合 (case1), rng023 を用いたときの test601-test700 の場合 (case2) 以外は, ２つの検

定に対してすべて合格となった. 一方, case1 と case2 においては 2 つの検定ともに不合格となった (付け加えると, 有意水準 $\alpha = 0.000001$ で検定してもすべて不合格となる).

TEST :	RUN TEST		POKER TEST	
Algorithm :	KAWAL	rng023	KAWAL	rng023
File: test201 - test300	◯	◯	◯	◯
File: test301 - test400	◯	◯	◯	◯
File: test401 - test500	◯	◯	◯	△
File: test501 - test600	◯	◯	◯	◯
File: test601 - test700	×	×	×	×

◯ $\cdots$ 有意水準 $\alpha = 0.05$ で検定に合格

△ $\cdots$ $\alpha = 0.05$ で検定に不合格, $\alpha = 0.01$ で検定に合格

× $\cdots$ $\alpha = 0.01$ で検定に不合格

表 4.10: 連の検定とポーカー検定の実行結果

第5章　結び

著者と鈴木講師は, KAWAL, rng023 という手書き曲線 (落書きビットマップ) から擬似乱数 (ビット列) を生成するアルゴリズムを開発した (第 3 章参照). そして KAWAL, rng023 より生成される擬似乱数列は, 連の検定, ポーカー検定という擬似乱数列のランダム性を検証する統計的検定において優良な結果を得ることができた (第 4 章参照).

しかし, 上記のアルゴリズムの実用化を考えた場合, まだ検証すべきことがある.

まず第一に, 生成された擬似乱数列の統計的検証はまだ不十分だと思われる. KAWAL および rng023 は, 連の検定とポーカー検定において優良な結果を得ることができたが, それだけで精度の高い擬似乱数列を生成しているとはいいきれない. 特に, 乱数列のパターン性の検証については, 本論文で行った検定だけでは不十分である. また今回の実験に用いた落書きビットマップは, すべて著者と鈴木講師が描いたものであり, サンプルの数も少ない. もっと多くの人に描いてもらってサンプルを集め, 検証を十分なものにしていく必要がある.

第二に, 生成された擬似乱数列の周期の問題が挙げられる. KAWAL, rng023 ともに出力値は決して長いとはいえず, 特に rng023 の場合, ビットマップの絵に関わらず出力値の長さがわずか 2730 ビットである. シミュ

レーション実験に 本論文の乱数生成アルゴリズムを用いる場合, 擬似乱数列が足りなくなることが予測され, あらかじめたくさんの落書きビットマップを用意しておくなど対応策が求められるであろう.

第三に, KAWAL の計算量の観点からみた場合の信頼性の問題が挙げられる. rng023 と違い, KAWAL に対しては, まだ出力値が手書き曲線のランダム性を保つことについて数学的な理論による裏づけを確立されていないからである. KAWAL の信頼性を向上するためにも早急に解決すべき問題だろう.

我々が開発した乱数生成アルゴリズムの実用化に向けて, これらの課題に取り組み, アルゴリズムの信頼性を向上していきたいと著者は考えている.

付 録A　風景写真からの乱数生成

ここでは, これまで紹介してきた手書き曲線から乱数を生成する方法ではなく, 風景写真から乱数を生成する方法について紹介する. 擬似乱数を生成するアルゴリズムは本論文で紹介した KAWAL(第 3.2 節を参照) と rng023(第 3.3 節を参照) を用いる.

付録 A の構成は以下の通りである. まず本論文で用いた風景写真について紹介する. 次に風景写真を画像処理し, 最終的に正方行列にするまでの手順を示す. 正方行列は KAWAL(rng023) で処理される. その後, KAWAL(rng023) より出力された擬似乱数の検定結果について紹介する.

A.1　風景写真

ここでは, 乱数生成アルゴリズムの乱数源となる風景写真について紹介する.

写真を撮るために用いた道具は, ニコン社のデジタルカメラ 'COOLPIX5600' である. デジタルカメラで撮った画像 (JPEG 形式, キャンバスの大きさは 2592×1944 ピクセル) をパソコンに転送し, それを 24 ビット BMP 形式 (フルカラー) で保存する. また, ビットマップのキャンバスの大きさを, JPEG 画像と同様にする.

本研究で用いた写真は, すべて著者の郷土である奈良県五條市 (旧西吉野村, 旧大塔村含む) と和歌山県橋本市の風景である. 著者は五條市と橋本市の風景写真を全部で 100 枚用意した. 図 A.1, 図 A.2 はその一部である.

図 A.1: 旧大塔村の風景

図 A.2: 五條市新町の風景

A.2 画像処理

KAWAL, rng023 に処理できるように風景写真の画像処理を行う. その手順は以下のように行う.

1. ビットマップのキャンバスの大きさを 20% に縮小する (2592×1944 ピクセル $\rightarrow 519 \times 389$ ピクセル).
2. ビットマップのフォーマットを 24 ビット BMP 形式から 16 色 BMP 形式に変換する.
3. １つのビットマップから 256×256 ピクセル部分を２ヶ所抜き取る.
4. 抜き取った部分を正方行列に変換する.
5. 正方行列に KAWAL(rng023) を適用する.

なお, 1,2 は WINDOWS の「ペイント」で処理を行い, 3 以降は, C++ プログラミングで処理を行う. これより上の手順の詳細な説明を行う.

STEP1: ビットマップのキャンバスの大きさを 20% に縮小する.

ビットマップのキャンバスの大きさを 垂直, 水平ともに 20% に縮小する. キャンバスの大きさは, 2592 × 1944 ピクセルから 519 × 389 ピクセルになる. つまり, 図 A.3 から 図 A.4 のように変換される.

→

図 A.3: 2592 × 1944

図 A.4: 519 × 389

STEP2: ビットマップのフォーマットを 24 ビット BMP 形式から 16 色 BMP 形式に変換する.

ビットマップのフォーマットを 24 ビット BMP 形式から 16 色 BMP 形式に変換する. 例えば, 図 A.5 から 図 A.6 のように変換される.

→

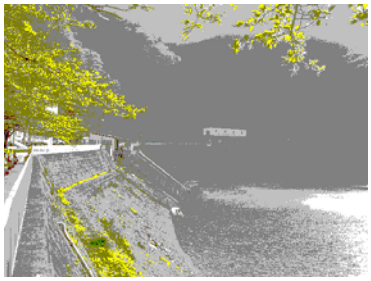


図 A.5: 24 ビット BMP

図 A.6: 16 色 BMP

STEP3: １つのビットマップから 256×256 ピクセル部分を２ヶ所抜き取る.

１つのビットマップから256×256 ピクセル部分を２ヶ所抜き取る. 100 枚の風景写真に対して, 200 個の正方行列が作られる. ビットマップの一番左上のピクセルを (0,0) 座標とすると, ４点: (0,0),(0,-255),(255,-255),(255,0) で囲まれた部分と, ４点: (256,0),(256,-255),(511,-255),(511,0) で囲まれた部分を抜き取る. 詳しくは, 図 A.7 を参考されたい.

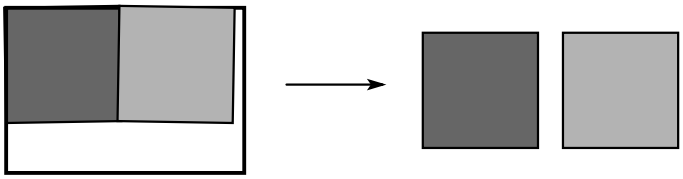


図 A.7: 256×256 ピクセル部分を２ヶ所抜き取る

STEP4: 抜き取った部分を正方行列に変換する.

抜き取った部分 (大きさは 256×256 ピクセル) を正方行列に変換する. ビットマップの上から i 番目, 左から j 番目のピクセルは, 行列の要素 $a_{i,j}$(ただし, $a_{i,j} \in \{0, 1 \cdots 15\}$) に対応する. 各要素の値は, 対応するピクセルの色によって異なる.

STEP5: 正方行列を KAWAL(rng023) で処理する.

正方行列を KAWAL(rng023) で処理する. 100 枚の風景写真に対して, 200 個の擬似乱数が生成される.

A.3　統計的検定

第 4 章で紹介した統計的検定法 (連の検定, ポーカー検定) を用いて, KAWAL と rng023 より出力された擬似乱数のランダム性を検証する. KAWAL(rng023) の乱数源は, A.1 で紹介した 100 枚の風景写真である.

A.3.1　連の検定

100 枚の風景写真を用い, 以下の手順で検定を行う. その方法は, 第 4.1 節と同じである.

STEP1: 連の個数を統計量 V に変換する

100 枚の風景写真から生成された 200 個の擬似乱数 (ビット列) の各々について, 連の個数 R を統計量 V に変換する. ただし, n はビット列の長さ, p は 1 の割合.

$$V = \frac{R - 2np(1-p)}{\sqrt{4np^2(1-p)^2}}$$

こうして, V の値 200 個を得る.

STEP2: 統計量 V のクラス分けを行う

STEP1 より求めた統計量 V の値 200 個について, 度数分布表を作る. 度数分布表の各クラスはいずれも, 標準正規分布の全面積の 5 % 部分の区間 (interval) である. したがって, 各クラスの確率は 0.05, 期待度数は $10(= np$, 標本の数 $n = 200$, 確率 $p = 0.05)$ である.

STEP3: 度数分布表よりカイ２乗値を求め, カイ２乗検定を行う

A.3.2 ポーカー検定

100 枚の風景写真を用い, 以下の手順で検定を行う. その方法は, 第 4.2 節と同じである.

STEP1: ビット列を 10 進数に変換する

100 枚の風景写真より生成された 200 個の擬似乱数 (ビット列) の各々を 10 進数の配列に変換する. ビット列を左から 4 ビットずつ区切り, 一組ごとに 10 進数化 (0-15) する. 10 進数に変換したものから順に, 一次元配列 a に左詰めに格納していく. こうして, 200 個の配列 を得る.

STEP2: 各配列 a を用いてポーカーを行う

1. まず配列 a の左端から順に, 整数を５個ずつ取り出す.
2. 取り出した５個の整数を１組とし, その中の整数の種類の数によって, クラス分けを行う. クラスの数は５つである. ただし, 最終組 (a の右端の組) の整数の数が５個に満たない場合は, それを無視する.
3. 度数を表す配列 f を用意し, 各クラスの度数 (f_1 から f_5) を求める. f_i は, 一つのクラス (i は, 5 個組の整数の種類) の度数である.

こうして, 各 $i(i = 1, 2, \cdots 5)$ に対して, f_i の値 200 個を得る.

STEP3: カイ２乗値を求め, カイ２乗検定を行う

各 $i(i = 1, 2, \cdots 5)$ に対して, 200 個の f_i の総和 F_i を求める. F_1 から F_5 を用いて, カイ２乗値を求める. ただし, F_1 のクラスの期待値が十分小さいので, F_2 のクラスと合併する.

A.3.3 実験結果

連の検定 (A.3.1 を参照) による実験結果を表 A.1 に, ポーカー検定 (A.3.2 を参照) による実験結果を表 A.2 と 表 A.3 にそれぞれ示す. なお, 表 A.1 には, 200 個の擬似乱数 (ビット列) の 1 の割合の平均値などもあわせて示す.

	KAWAL	rng023
１の割合の平均値	0.498	0.489
１の割合の分散	7.71×10^{-5}	2.37×10^{-4}
出力の長さの平均値	7294	2730（一定）
出力の長さの分散	9.38×10^{6}	0
V の平均値	-0.8618	-0.2798
V の分散	2.6624	1.3645
カイ２乗値	198.2	65.4

表 A.1: 連の検定による実験結果

kinds of 5cards	1kind	2kinds	3kinds	4kinds	5kinds
frequency	0	255	5753	30331	36516
probability	0.00002	0.0034	0.0801	0.4166	0.4999
expectation	1.1117	250.13	5836.40	30349.26	36419.11

total of frequency = 72856, value of chi-square = 1.5505

表 A.2: ポーカー検定による実験結果 (KAWAL)

kinds of 5cards	1kind	2kinds	3kinds	4kinds	5kinds
frequency	0	91	2116	11340	13653
probability	0.00002	0.0034	0.0801	0.4166	0.4999
expectation	0.2075	93.38	2178.96	11330.57	13596.68

total of frequency = 27200, value of chi-square = 2.1436

表 A.3: ポーカー検定による実験結果 (rng023)

実験結果の検証

まず, ポーカー検定の結果について検証する. 求めたカイ 2 乗値 χ_0^2 がポーカー検定に合格するかどうか判定する基準として, $\chi_0^2 < 7.81$ ($= \chi^2(3, 0.05)$. カイ 2 乗分布の有意水準 $\alpha = 0.05$, 自由度 3) のとき, 検定に合格とする. 表 A.2 と 表 A.3 より, KAWAL, rng023 はともにポーカー検定に合格する.

連の検定の結果について検証する. 求めたカイ 2 乗値 χ_0^2 がカイ 2 乗検定に合格するかどうか判定する基準として, $\chi_0^2 < 30.1$ ($= \chi^2(19, 0.05)$. カイ 2 乗分布の有意水準 $\alpha = 0.05$, 自由度 19) のとき, 検定に合格とする. 表 A.1 より, KAWAL, rng023 はともにカイ 2 乗検定に大きく不合格となる. KAWAL と rng023 は, $\alpha = 0.000001$ のカイ 2 乗検定においても不合格となる ($\chi^2(19, 0.000001) = 63.22$).

A.4 A.2 の修正とその統計的検定

A.3 より, KAWAL, rng023 ともに連の検定においては良い結果を得ることができなかった. そこで, A.2 を修正し, もう一度統計的検定を行う. ここでは, A.2 の修正内容とその検定結果を示す.

A.4.1 A.2 の修正

A.2 の修正内容を紹介する. 著者は, A.2 の **STEP3** のみを修正する. 詳しくは, **STEP3′** を参考されたい.

STEP3′: １つのビットマップから 256×256 ピクセル部分を２ヶ所抜き取る.

１つのビットマップから256×256 ピクセル部分を２ヶ所抜き取る. 100 枚の風景写真に対して, 200 個の正方行列が作られる. ビットマップの一番左上のピクセルを (0,0) 座標とすると, ４点: (0,-100),(0,-355),(255,-355),(255,-100) で囲まれた部分と, ４点: (256,-100),(256,-355),(511,-355),(511,-100) で囲まれた部分を抜き取る. 詳しくは, 図 A.8 を参考されたい.

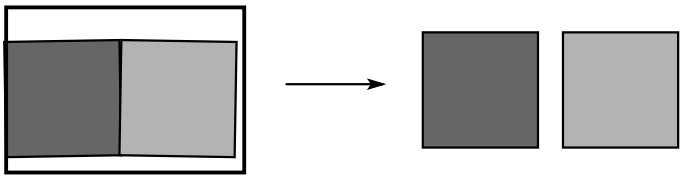


図 A.8: 256×256 ピクセル部分を２ヶ所抜き取る

STEP3 を修正した動機として, 図 A.9 や 図 A.10 のように上部が空で覆われた写真は, 連の検定において良くない結果 (統計量 V の値が悪かった) を得たことがあったからである. 著者は, 空の部分は乱数源としてふさわしくないと判断し, 写真上部を乱数源として用いないようにした.

図 A.9: 柿園と山の風景

図 A.10: 五條市犬飼町の風景

A.4.2 実験結果

A.1 で紹介した風景写真 100 枚を用いて, もう一度 A.3 と同様の実験を行う. 連の検定による実験結果を表 A.4 に, ポーカー検定による実験結果を表 A.5 と表 A.6 にそれぞれ示す. なお, 表 A.4 には, 200 個の擬似乱数 (ビット列) の 1 の割合の平均値などもあわせて示す.

	KAWAL	rng023
1 の割合の平均値	0.498	0.491
1 の割合の分散	4.65×10^{-5}	2.14×10^{-4}
出力の長さの平均値	8145	2730（一定）
出力の長さの分散	8.54×10^{6}	0
V の平均値	-0.9118	-0.1939
V の分散	2.0831	1.1788
カイ 2 乗値	208.8	24.2

表 A.4: 連の検定による実験結果

kinds of 5cards	1kind	2kinds	3kinds	4kinds	5kinds
frequency	2	293	6331	34031	40692
probability	0.00002	0.0034	0.0801	0.4166	0.4999
expectation	1.2413	279.29	6516.76	33887.14	40664.57

total of frequency = 81349, value of chi-square = 6.6705

表 A.5: ポーカー検定による実験結果 (KAWAL)

kinds of 5cards	1kind	2kinds	3kinds	4kinds	5kinds
frequency	0	92	2209	11289	13610
probability	0.00002	0.0034	0.0801	0.4166	0.4999
expectation	0.4150	93.38	2178.96	11330.57	13596.68

total of frequency $= 27200$, value of chi-square $= 0.6143$

表 A.6: ポーカー検定による実験結果 (rng023)

実験結果の検証

まず, ポーカー検定の結果について検証する. 求めたカイ 2 乗値 χ_0^2 がポーカー検定に合格するかどうか判定する基準として, $\chi_0^2 < 7.81$ ($= \chi^2(3, 0.05)$. カイ 2 乗分布の有意水準 $\alpha = 0.05$, 自由度 3) のとき, 検定に合格とする. 表 A.5 と 表 A.6 より, KAWAL, rng023 はともにポーカー検定に合格する.

連の検定の結果について検証する. 求めたカイ 2 乗値 χ_0^2 がカイ 2 乗検定に合格するかどうか判定する基準として, $\chi_0^2 < 30.1$ ($= \chi^2(19, 0.05)$. カイ 2 乗分布の有意水準 $\alpha = 0.05$, 自由度 19) のとき, 検定に合格とする. 表 A.4 より, rng023 はカイ 2 乗検定に合格し, KAWAL はカイ 2 乗検定に不合格となる. なお, KAWAL の場合は, $\alpha = 0.000001$ のカイ 2 乗検定においても不合格となる ($\chi_0^2 = 208.8 > 63.22 = \chi^2(19, 0.000001)$).

A.5　まとめ

表 A.7 は, A.3.3 と A.4.2 の結果を簡潔にまとめたものである. 表 A.7 より, 検定結果についてもう一度述べる. A.2 を用いた場合, KAWAL, rng023 ともにポーカー検定のみ良い結果を得ることができた. A.4.1 を用いた場合, rng023 は両検定ともに良い結果を得ることができ, KAWAL はポーカー検定のみ良い結果を得ることができた.

ところで, KAWAL は連の検定において良い結果を得ることができなかった. 著者は, KAWAL を連の検定に合格させるため, 引き続き風景写真の画像処理に取り組んでいきたい.

TEST :	連の検定		ポーカー検定	
Algorithm :	KAWAL	rng023	KAWAL	rng023
A.2	×	×	○	○
A.4.1	×	○	○	○

○ $\cdots$ 有意水準 $\alpha = 0.05$ で検定に合格

△ $\cdots$ $\alpha = 0.05$ で検定に不合格, $\alpha = 0.01$ で検定に合格

× $\cdots$ $\alpha = 0.01$ で検定に不合格

表 A.7: 連の検定とポーカー検定の実行結果

付 録B　落葉の写真からの乱数生成

ここでは, 落葉の写真から乱数を生成する方法について紹介する. 擬似乱数を生成するアルゴリズムは本論文で紹介した KAWAL(第 3.2 節を参照) と rng023(第 3.3 節を参照) を用いる.

付録 B の構成は以下の通りである. まず本論文で用いた落葉の写真について紹介する. 落葉の写真は鈴木講師によって用意された. 次に落葉の写真を画像処理し, 最終的に正方行列にするまでの手順を示す. 正方行列は KAWAL(rng023) で処理される. その後, KAWAL(rng023) より出力された擬似乱数の検定結果について紹介する.

B.1　落葉の写真

落葉の写真と写真を撮影するのに用いた道具について説明する.

写真を撮るために用いた道具は, オリンパス社のデジタルカメラ 'μ-mini DIGITAL S' である. デジタルカメラで撮った画像 (JPEG 形式, キャンバスの大きさは 2560×1920 ピクセル) をパソコンに転送し, それを 24 ビット BMP 形式 (フルカラー) で保存する. また, ビットマップのキャンバスの大きさを, JPEG 画像と同様にする. 鈴木講師は, 首都大学東京で落葉を 50 枚撮影した. 図 B.1 と, 図 B.2 はその一部である.

図 B.1: 落葉の写真１

図 B.2: 落葉の写真２

B.2 画像処理

B.2.1 画像処理

KAWAL(rng023) で処理できるように, 落葉の写真を正方行列に変換する. その方法は, A.2 と同様であるが, 写真のキャンバスの大きさと写真の枚数が 風景写真 (A.1 を参照) と異なる. そのため **STEP1** と **STEP5** の内容が一部変更される (下線部は変更部分).

STEP1: ビットマップのキャンバスの大きさを 20% に縮小する.

ビットマップのキャンバスの大きさを 垂直, 水平ともに 20% に縮小する. キャンバスの大きさは, $\underline{2560 \times 1920}$ ピクセルから $\underline{512 \times 384}$ ピクセルになる.

STEP2: ビットマップのフォーマットを 24 ビット BMP 形式から 16 色 BMP 形式に変換する.

A.2 の STEP2 と同様の操作を行う.

STEP3: １つのビットマップから 256×256 ピクセル部分を２ヶ所抜き取る.

A.2 の STEP3 と同様の操作を行う.

STEP4: 抜き取った部分を正方行列に変換する.

A.2 の STEP4 と同様の操作を行う.

STEP5: 正方行列を KAWAL(rng023) で処理する.

正方行列を KAWAL(rng023) で処理する. 50 枚の落葉の写真 に対して, 100 個の擬似乱数 が生成される.

B.2.2 統計的検定による実験結果 (失敗)

KAWAL(rng023) より出力された 100 個の擬似乱数を用いて, 統計的検定 (A.3 を参照) を行った. すると, KAWAL, rng023 ともに連の検定 (A.3.1 参照) において非常に悪い結果を得た. 詳しく述べると, KAWAL のカイ２乗値 (自由度 19) は $\chi_0^2 = 1900.0$, rng023 のカイ２乗値 (自由度 19) は $\chi_0^2 = 1632.4$ となった. またポーカー検定 (A.3.2 参照) においても, KAWAL, rng023 ともに良くない結果を得た.

落葉の写真 (512×384 ピクセル, 16 色 BMP, 図 B.4 参照) は, 風景写真 (519×389 ピクセル, 16 色 BMP, 図 B.3 参照) と違って統計的検定に全く合格しなかった. そこで, B.2.1 の修正を行う. ２つの修正案 (B.3.1, B.3.2) を示した後, もう一度統計的検定を行う.

図 B.3: 風景写真 (16 色 BMP)

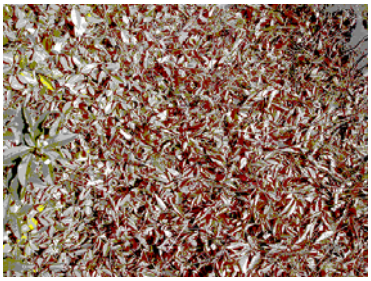


図 B.4: 落葉写真 (16 色 BMP)

B.3　B.2.1 の修正

これより, B.2.1 の修正案を 2 つ紹介する.

B.3.1　修正案 1 (モノクロ BMP に変換)

著者は, B.2.1 の **STEP2** のみを修正する. 詳しくは, **STEP2′** を参考されたい.

STEP2′: ビットマップのフォーマットを 24 ビット BMP 形式からモノクロ BMP 形式に変換する.

ビットマップのフォーマットを 24 ビット BMP 形式から モノクロ BMP 形式に変換する. 例えば, 図 B.5 から 図 B.6 のように変換される.

図 B.5: 24 ビット BMP

→

図 B.6: モノクロ BMP

B.3.2　修正案２ (ビットマップを縮小しない)

この修正案は若干複雑であるため, KAWAL(rng023) で処理するまでの手順を最初から紹介する. まず, ビットマップを縮小しない (B.2.1 の **STEP1** をスキップする) ことに注意されたい.

STEP1: ビットマップのフォーマットを 24 ビット BMP 形式から 16 色 BMP 形式に変換する.

STEP2: １つのビットマップから 256×256 ピクセル部分を ４ヶ所 抜き取る.

１つのビットマップから 256×256 ピクセル部分を ４ヶ所 抜き取る. ビットマップの一番左上のピクセルを (0,0) 座標とすると, ４点: (0,0),(0,-255),(255,-255),(255,0) で囲まれた部分と, ４点: (256,0),(256,-255),(511,-255),(511,0) で囲まれた部分, ４点: (0,-256),(0,-511),(255,-511),(255,-256) で囲まれた部分, ４点: (256,-256),(256,-511),(511,-511),(511,-256) で囲まれた部分を抜き取る. 詳しくは, 図 B.7 を参考されたい.

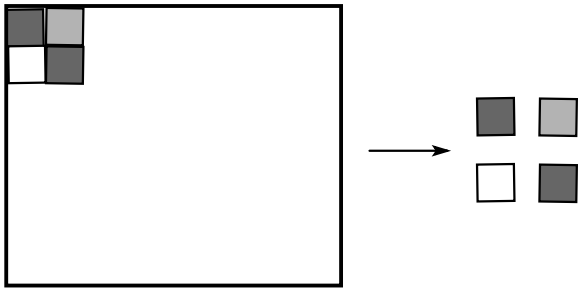


図 B.7: 256×256 ピクセル部分を４ヶ所抜き取る

STEP3: 抜き取った部分を正方行列に変換する.

STEP4: 正方行列を KAWAL(rng023) で処理する.

正方行列を KAWAL(rng023) で処理する. 50 枚の落葉の写真に対して, 200 個の擬似乱数 が生成される.

B.4 統計的検定

B.1 で紹介した落葉の写真 50 枚より, B.3.1(B.3.2) の方法を用いて 100(200) 個の擬似乱数を生成する. そして, 生成された擬似乱数のランダム性を検証するために, A.3 と同様の検定 (連の検定, ポーカー検定) を行う.

B.4.1 B.3.1 の統計的検定

KAWAL(rng023) より 100 個の擬似乱数 (ビット列) が生成される. 連の検定による実験結果を表 B.1 に, ポーカー検定による実験結果を表 B.2 と表 B.3 にそれぞれ示す. なお, 表 A.4 には, 100 個の擬似乱数 (ビット列) の 1 の割合の平均値などもあわせて示す.

	KAWAL	rng023
1の割合の平均値	0.496	0.500
1の割合の分散	2.16×10^{-5}	8.90×10^{-5}
出力の長さの平均値	13977	2730（一定）
出力の長さの分散	579187	0
V の平均値	0.1356	-0.0913
V の分散	1.1389	1.3315
カイ2乗値	10.8	24.0

表 B.1: 連の検定による実験結果

kinds of 5cards	1kind	2kinds	3kinds	4kinds	5kinds
frequency	4	246	5488	29201	34900
probability	0.00002	0.0034	0.0801	0.4166	0.4999
expectation	1.0657	239.77	5594.71	29092.48	34910.97

total of frequency = 69839, value of chi-square = 2.7920

表 B.2: ポーカー検定による実験結果 (KAWAL)

kinds of 5cards	1kind	2kinds	3kinds	4kinds	5kinds
frequency	0	36	1081	5715	6768
probability	0.00002	0.0034	0.0801	0.4166	0.4999
expectation	0.2075	46.69	1089.48	5665.28	6798.34

total of frequency = 13600, value of chi-square = 3.1707

表 B.3: ポーカー検定による実験結果 (rng023)

実験結果の検証

まず, ポーカー検定の結果について検証する. 求めたカイ 2 乗値 χ_0^2 がポーカー検定に合格するかどうか判定する基準として, $\chi_0^2 < 7.81$ ($= \chi^2(3, 0.05)$. カイ 2 乗分布の有意水準 $\alpha = 0.05$, 自由度 3) のとき, 検定に合格とする. 表 B.2 と 表 B.3 より, KAWAL, rng023 はともにポーカー検定に合格する.

連の検定の結果について検証する. 求めたカイ 2 乗値 χ_0^2 がカイ 2 乗検定に合格するかどうか判定する基準として, $\chi_0^2 < 30.1$ ($= \chi^2(19, 0.05)$. カイ 2 乗分布の有意水準 $\alpha = 0.05$, 自由度 19) のとき, 検定に合格とする. 表 B.1 より, KAWAL, rng023 はともにカイ 2 乗検定に合格する.

B.4.2 B.3.2 の統計的検定

KAWAL(rng023) より 200 個の擬似乱数 (ビット列) が生成される. 連の検定による実験結果を表 B.4 に, ポーカー検定による実験結果を表 B.5 と表 B.6 にそれぞれ示す. なお, 表 B.4 には, 200 個の擬似乱数 (ビット列) の 1 の割合の平均値などもあわせて示す.

	KAWAL	rng023
1の割合の平均値	0.492	0.498
1の割合の分散	3.02×10^{-5}	8.94×10^{-5}
出力の長さの平均値	14474	2730（一定）
出力の長さの分散	1.54×10^{6}	0
V の平均値	-0.3917	-0.2504
V の分散	1.0554	0.9328
カイ2乗値	44.0	33.8

表 B.4: 連の検定による実験結果

kinds of 5cards	1kind	2kinds	3kinds	4kinds	5kinds
frequency	1	466	11519	60313	72341
probability	0.00002	0.0034	0.0801	0.4166	0.4999
expectation	2.2070	496.58	11586.91	60251.95	72302.34

total of frequency = 144640,　value of chi-square = 2.5066

表 B.5: ポーカー検定による実験結果 (KAWAL)

kinds of 5cards	1kind	2kinds	3kinds	4kinds	5kinds
frequency	0	93	2123	11314	13670
probability	0.00002	0.0034	0.0801	0.4166	0.4999
expectation	0.4150	93.38	2178.96	11330.57	13596.68

total of frequency = 27200,　value of chi-square = 1.8633

表 B.6: ポーカー検定による実験結果 (rng023)

実験結果の検証

まず, ポーカー検定の結果について検証する. 求めたカイ２乗値 χ_0^2 がポーカー検定に合格するかどうか判定する基準として, $\chi_0^2 < 7.81$ ($= \chi^2(3, 0.05)$) のとき, 検定に合格とする. 表 B.5 と 表 B.6 より, KAWAL, rng023 はともにポーカー検定に合格する.

連の検定の結果について検証する. 求めたカイ２乗値 χ_0^2 がカイ２乗検定に合格するかどうか判定する基準として, $\chi_0^2 < 30.1$ ($= \chi^2(19, 0.05)$) のとき, 検定に合格とする. 表 B.4 より, KAWAL, rng023 はともにカイ２乗検定に不合格となる. KAWAL と rng023 のカイ２乗値について, もう少し詳しく検証する. KAWAL の場合は, 有意水準 $\alpha = 0.00090$ のカイ２乗検定においては合格する ($\chi_0^2 = 44.0 < 44.15 = \chi^2(19, 0.00090)$). rng023 の場合は, $\alpha = 0.0190$ のカイ２乗検定においては合格する ($\chi_0^2 = 33.8 < 33.877 = \chi^2(19, 0.0190)$).

B.5 まとめ

表 B.7 は, B.4.1 と B.4.2 の結果を簡潔にまとめたものである. 表 B.7 より, 検定結果についてもう一度述べる. B.3.1 を用いた場合, KAWAL, rng023 ともに両検定において非常に良い結果を得ることができた. 一方, B.3.2 を用いた場合, KAWAL, rng023 ともにポーカー検定において良い結果を得たものの, 連の検定においてあまり良い結果を得ることができなかった.

B.3.2 について議論する. B.3.2 の特徴として, 元のビットマップを縮小

しないというのがある. つまり一つのビットマップに対して, たくさんの独立した正方行列を作ることが可能である (ビットマップのキャンバスサイズが 2560×1920 ピクセルなので, 最大で 70 個の正方行列が生成可能). 70 個の正方行列をすべて KAWAL(rng023) に用いることで, KAWAL(rng023) の課題の一つである「出力の短さ」をある程度解消することができる. 例えば rng023 の場合, 一つのビットマップで $2730 \times 70 = 191100$ という出力の長さを生成できるようになる.

TEST :	連の検定		ポーカー検定	
Algorithm :	KAWAL	rng023	KAWAL	rng023
B.3.1	○	○	○	○
B.3.2	×	△	○	○

○ $\cdots$ 有意水準 $\alpha = 0.05$ で検定に合格

△ $\cdots$ $\alpha = 0.05$ で検定に不合格, $\alpha = 0.01$ で検定に合格

× $\cdots$ $\alpha = 0.01$ で検定に不合格

表 B.7: 連の検定とポーカー検定の実行結果

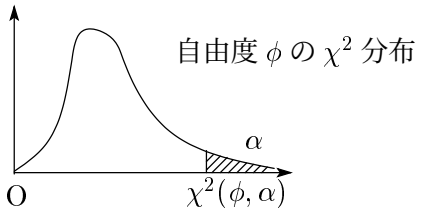
謝辞

指導教官である鈴木登志雄講師に深く感謝いたします．鈴木講師には本研究を進めるにあたり, 終始懇切丁寧なご指導とご助言を頂きました．また研究以外の事でも多くのご助言を頂きました.

宝珍輝尚教授に深く感謝いたします．宝珍教授にはビットマップファイルの扱い方についてご指導頂きました.

最後に, 著者を支えてくれた多くの皆様に厚くお礼申し上げます.

カイ２乗分布表



(上側確率 α, 自由度 ϕ から $\chi^2(\phi,\alpha)$ を求める表)

	0.995	0.99	0.975	0.95	0.9	0.75	0.5	0.25	0.1	0.05	0.025	0.01	0.005
1	0.00003	0.0002	0.001	0.0039	0.0158	0.1015	0.4549	1.3233	2.7055	3.8415	5.0239	6.6349	7.8794
2	0.01002	0.0201	0.0506	0.1026	0.2107	0.5754	1.3863	2.7726	4.6052	5.9915	7.3778	9.2104	10.597
3	0.07172	0.1148	0.2158	0.3518	0.5844	1.2125	2.366	4.1083	6.2514	7.8147	9.3484	11.345	12.838
4	0.20698	0.2971	0.4844	0.7107	1.0636	1.9226	3.3567	5.3853	7.7794	9.4877	11.143	13.277	14.86
5	0.41175	0.5543	0.8312	1.1455	1.6103	2.6746	4.3515	6.6257	9.2363	11.07	12.832	15.086	16.75
6	0.67573	0.8721	1.2373	1.6354	2.2041	3.4546	5.3481	7.8408	10.645	12.592	14.449	16.812	18.548
7	0.98925	1.239	1.6899	2.1673	2.8331	4.2549	6.3458	9.0371	12.017	14.067	16.013	18.475	20.278
8	1.3444	1.6465	2.1797	2.7326	3.4895	5.0706	7.3441	10.219	13.362	15.507	17.535	20.09	21.955
9	1.73491	2.0879	2.7004	3.3251	4.1682	5.8988	8.3428	11.389	14.684	16.919	19.023	21.666	23.589
10	2.15585	2.5582	3.247	3.9403	4.8652	6.7372	9.3418	12.549	15.987	18.307	20.483	23.209	25.188
11	2.6032	3.0535	3.8157	4.5748	5.5778	7.5841	10.341	13.701	17.275	19.675	21.92	24.725	26.757
12	3.07379	3.5706	4.4038	5.226	6.3038	8.4384	11.34	14.845	18.549	21.026	23.337	26.217	28.3
13	3.56504	4.1069	5.0087	5.8919	7.0415	9.2991	12.34	15.984	19.812	22.362	24.736	27.688	29.819
14	4.07466	4.6604	5.6287	6.5706	7.7895	10.165	13.339	17.117	21.064	23.685	26.119	29.141	31.319
15	4.60087	5.2294	6.2621	7.2609	8.5468	11.037	14.339	18.245	22.307	24.996	27.488	30.578	32.801
16	5.14216	5.8122	6.9077	7.9616	9.3122	11.912	15.338	19.369	23.542	26.296	28.845	32	34.267
17	5.69727	6.4077	7.5642	8.6718	10.085	12.792	16.338	20.489	24.769	27.587	30.191	33.409	35.718
18	6.26477	7.0149	8.2307	9.3904	10.865	13.675	17.338	21.605	25.989	28.869	31.526	34.805	37.156
19	6.84392	7.6327	8.9065	10.117	11.651	14.562	18.338	22.718	27.204	30.144	32.852	36.191	38.582
20	7.43381	8.2604	9.5908	10.851	12.443	15.452	19.337	23.828	28.412	31.41	34.17	37.566	39.997
30	13.7867	14.953	16.791	18.493	20.599	24.478	29.336	34.8	40.256	43.773	46.979	50.892	53.672
40	20.7066	22.164	24.433	26.509	29.051	33.66	39.335	45.616	51.805	55.758	59.342	63.691	66.766
50	27.9908	29.707	32.357	34.764	37.689	42.942	49.335	56.334	63.167	67.505	71.42	76.154	79.49
60	35.5344	37.485	40.482	43.188	46.459	52.294	59.335	66.981	74.397	79.082	83.298	88.379	91.952
70	43.2753	45.442	48.758	51.739	55.329	61.698	69.334	77.577	85.527	90.531	95.023	100.43	104.21
80	51.1719	53.54	57.153	60.391	64.278	71.145	79.334	88.13	96.578	101.88	106.63	112.33	116.32

(注) 著者は, 上の表を Microsoft 社の 'Excel' で作成した.

参考文献

[Gent03] Gentle, J.E.: *Random number generation and Monte Carlo methods, second edition*, Springer, New York (2003).

[Gold99] Goldreich, O.: *Modern cryptography, probabilistic proofs and pseudorandomness*, Springer, Berlin / New York (1999).
邦訳 O. ゴールドライヒ著 ; 岡本龍明, 藤崎英一郎 訳『現代暗号・確率的証明・擬似乱数』シュプリンガー・フェアラーク東京 (2001).

[Knuth98] Knuth, D.E.: *The Art of Computer Programming vol.2, third edition*, Addison-Wesley (1998).

[Papa94] Papadimitriou, C.H.: *Computational Complexity*, Addison-Wesley (1994).

[SK06] Suzuki, T. and Kawanishi, A.: *Random Extraction from Free-hand Drawings and Its Semantics Based on Forcing Complexity*, 投稿中 (Extended Abstract は, ftp://tnt.math.metro-u.ac.jp/pub/ac05/).

[石井他 95] 石井博昭, 塩出省吾, 新森修一,『確率統計の数理』, 裳華房 (1995).

[川西鈴木 05] 川西暁夫, 鈴木登志雄,『手書き曲線からの乱数抽出とそのセマンティクス (中間報告)』, 京都大学数理解析研究所講究録 1442 (2005).

[斉藤他 02] 斉藤義明, 堀潤一, 西村浩志, 木竜徹, 『可変容量パラメトロンによる物理乱数発生法』, 電子情報通信学会論文誌 Vol.J85-A No.2 (2002)

[清水 05] 清水隆邦, 『情報漏洩対策に乱数生成 IC を組み込むという試み』, Interface 2005 年 12 月号 pp.190-195, CQ 出版社 (2005)

[田中 97] 田中尚夫,『計算論理入門』, 裳華房 (1997).

[津田 95] 津田孝夫,『モンテカルロ法とシミュレーション (三訂版)』, 培風館 (1995).

[東芝 03] 東芝 e-ソリューション社 SI 技術開発センター, 『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).

[藤田他 03] 藤田忍, 内田建, 安田心一, 『高度情報セキュリティ向け超小型乱数生成回路』, 東芝レビュー Vol.58 No.8 (2003)

[伏見 89] 伏見正則,『乱数』, 東京大学出版会 (1989).

[宮坂 04] 宮坂電人, 『プログラミング雑技談 第２４回 BMPファイル作成プログラム』, C MAGAZINE 2004 年 6 月号 pp.113-117, ソフトバンクパブリッシング株式会社 (2004).

[宮武脇本 78] 宮武修, 脇本和昌,『乱数とモンテカルロ法』, 森北出版 (1978).

[脇本 70] 脇本和昌,『乱数の知識』, 森北出版 (1970).

[脇本 73] 脇本和昌,『身近なデータによる統計解析入門』, 森北出版 (1973).

[和田 04] 和田維作, 『特集１ 乱数をきわめろ！』, C MAGAZINE 2004 年 6 月号 pp.20-28 , ソフトバンクパブリッシング株式会社 (2004).

索 引