# ランダムウォークを用いた擬似乱数の検定

佐藤 春菜　(指導教員：萩田真理子)

## 1　はじめに

擬似乱数とは, 計算機によって作られる, 一見乱数列のように見えるが, 実際には確定的な計算によって求めている数列である. 良い擬似乱数の条件には, 周期が長いこと・高速に生成できること・統計的検定に耐えられること, があげられる. 本研究では, さまざまな方法で生成した擬似乱数をランダムウォークを用いて正の区域の滞在時間を測定し,$\chi^2$ 検定を行い, 各擬似乱数の乱数性を評価することを目的としている.

今回は Lagged Fibonacci 生成法と ran_array 生成法を中心に検定を行った.

## 2　言葉の定義

**ランダムウォーク**

ランダムウォークとは, 原点を出発点としてコイントスを繰り返し「表が出たら $(1,1)$ 方向に 1 歩進み, 裏が出たら $(1,-1)$ 方向に 1 歩進む」というルールで生成される折れ線グラフのこと言う. コイントスが $n$ 回のとき,$n$ 歩のランダムウォークという. 本研究では, ビットの 1 をコインの表, ビットの 0 をコインの裏と見立てて使用する.

**正の区域の滞在時間**

$y \geq 0$ の上半平面を歩いた歩数を正の区域の滞在時間という.

コイントスで作った $n$ 歩のランダムウォークのうち正の区域の滞在時間が $k$ となる確率を $P_{k,n}$ と表す. $n$ が偶数の時の $P_{k,n}$ の確率分布は以下の式で表される.

$$P_{k,n} = u_{2k} \cdot u_{2n-2k} \quad u_{2k} = \left( \begin{array}{c} 2k \\ k \end{array} \right) \cdot \frac{1}{2^{2k}} \qquad (1)$$

擬似乱数からランダムウォークを生成したとき, この確率分布に従わなければ乱数性が低いと言える.

## 3　擬似乱数生成法

擬似乱数生成法には以下のようなものがある.

**rand()**

C 言語の 70 〜 90 年代の標準擬似乱数である. 以下の式で定義されている.

$$x_{n+1} := ax_n + c \bmod M$$
$$a = 1103515345,\ c = 12345, M = 2^{31}$$

周期は $M = 2^{31}$

**Lagged Fibonacci**

「ラグ付フィボナッチ」と呼ばれる生成法である. 整数列 $x_n$ を次の漸化式で生成する.

$$x_{n+p} := x_{n+q} + x_n \bmod M (i = 0, 1, \cdots, p > q)$$

$p, q, M$:整数定数. しばしば $M = 2^w$
周期は $\leq 2^{w-1}(2^p - 1)$.$w$:下位 $w$ ビット

**random()**

90 年代以降の C 言語の現標準擬似乱数.Lagged Fibonacci 生成法の一つであり, 以下の式で定義される.

$$x_{n+p} := x_{n+q} + x_n \bmod M$$
$$\text{において} \quad p = 31, q = 3, M = 2^{31}$$

周期は $\sim 2^{62}$

**ran_array**

Knuth が 1997 年に提唱した Lagged Fibonacci の改良. Lagged Fibonacci 生成法で $L$ 個 ($L$:Luxuary Level と呼ぶ整数) 乱数を生成した後, 「100 個乱数を使い,$L - 100$ 個捨てる」を繰り返す.

Knuth は「$L \leq 200$ では乱数性が悪く,$L$ が $1,000$ に近ければ安全」と言っている.

**MT(Mersenne Twister)**

1997 年に松本-西村によって提唱された高速・高品質な擬似乱数. 632 次元空間での均等分布と周期 $(2^{19937}-1)$ が数学的に証明されている.

**SFMT**
**(SIMD-oriented Fast Mersenne Twister)**

2006 年に松本-斎藤によって開発された MT の改良. 多くの計算機環境で MT よりも高速であり,v ビット精度の次元均等分布性において MT からの改善が見られる. 周期は $2^{19937} - 1$.

## 4　検定方法

擬似乱数を発生させ, 最下位ビット (または最上位ビット) をとり, これを元に 1,000 歩のランダムウォークを 1,000 本生成し, それぞれの正の滞在時間を計算する. 滞在時間ごとに 31 のグループに分け, (1) 式に基づきそれぞれの期待度数を計算し, 自由度 30 の元 $\chi^2$ 検定を行う (以下,「検定 1」と呼ぶ). 検定 1 の棄却域は有意水準 0.05 で $\chi^2 \geq 43.77$ である.

これを各条件ずつ 25 回繰り返し, 得られた 25 個の $\chi^2$ 値を確率ごとに 5 つのグループにわけ, さらに自由度 4 で $\chi^2$ 検定を行う（以下,「検定 2」と呼ぶ). 検定 2 の棄却域は有意水準 0.05 で $\chi^2 \geq 9.49$ である.

$\chi^2$ 検定には MATHEMATICA を使用した.

## 5　検定結果

以下, それぞれの条件について検定 2 の $\chi^2$ 値を示す.

### 5.1　rand()

**最上位ビット:** $\chi^2 = 3.6$ , **最下位ビット:** $\chi^2 = 7.6$

最上位ビットは検定 2 の結果を見る限りでは乱数性が高いが, 検定 1 の段階で同じ $\chi^2$ 値が 2 回連続で出てしまうことがあった.

最下位ビットも検定 2 の結果は棄却できないが, 検定 1 の結果では棄却されるものが多く, 乱数性の悪い方に偏りが見られた.

### 5.2 random()

**最上位ビット:** $\chi^2 = 6.8$

これも rand() 同様, 検定 1 の段階で同じ $\chi^2$ 値が 2 回連続で出てしまうことがあり, その回数も rand() より多かった. 検定 2 の $\chi^2$ 値が大きめになってしまった原因と考えられる.

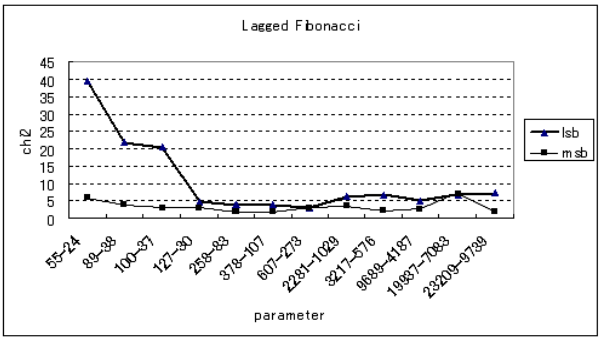
**最下位ビット:** 検定 1 での結果,$\chi^2$ 値の値が 4 種類しか現れなかった. 以下の表は 10,000 回実行した時の各出現パターンの出現回数である.

| $\chi^2 = 30.27$ | $\chi^2 = 35.18$ | $\chi^2 = 20.64$ | $\chi^2 = 27.79$ |
|---|---|---|---|
| 2499 | 2495 | 2499 | 2507 |

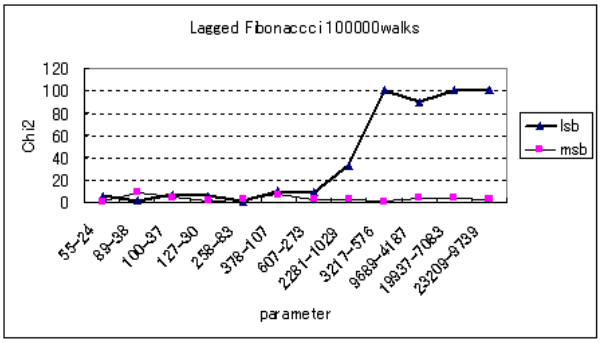
検定 2 も行うと, その $\chi^2$ 値は 20.8 となり, これは有意水準 0.001 で棄却出来るため, 乱数性は極めて低いと言える.

### 5.3 Lagged Fibonacci

12 種類のパラメータ $p, q$ について検定を行った.(パラメータの選び方は [1] に習った) 以下のグラフは 1,000 歩のランダムウォークを用いた検定 2 の $\chi^2$ 値をパラメータごとにグラフにしたものである.



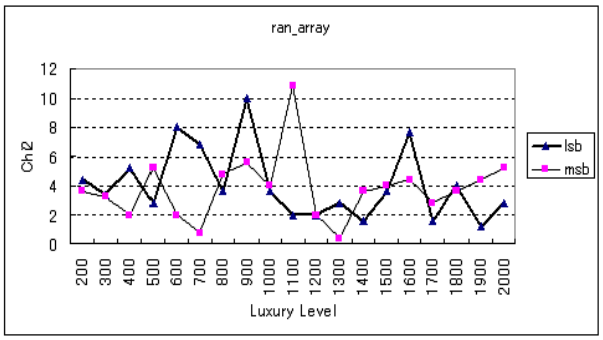

ラグの小さいパラメータしか棄却域に入らないのは, ランダムウォークの長さが 1,000 であることにより, ラグの大きいパラメータでは初期値の部分しか見ていないことによると考えられる. そこで, ランダムウォークを 100,000 歩にして再度検定を行った. 下記のグラフがその結果である.



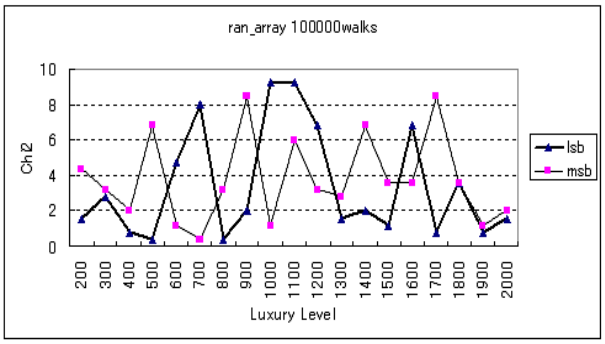

ラグの大きいパラメータの最下位ビットの $\chi^2$ 値は 100 に迫る. 検定 1 の $\chi^2$ も, 大きいものは 4,000 を超え, 乱数性は低いと言える. 最上位ビットについては, 検定 2 の $\chi^2$ 値で最大のものでも 8.8 で, 棄却域には入らなかった.

### 5.4 ran array

Lagged Fibonacci のパラメータ $p, q$ を $p = 100, q = 37$ にして検定を行った. 1,000 歩のランダムウォークを用いて,$L$ の値を 200 から 2000 まで 100 ずつ増やしていったグラフが以下のものである.




しかし, これも 1,000 歩では 10 回分しか乱数を使っていないことになるので, ランダムウォークの長さを 100,000 歩にして再度検定を行った (これだと 1000 回分使える). 以下がその結果である.




最上位ビットについては $L = 1000$ の時の $\chi^2$ 値は 1.2 と低い値であったが, その前後で大きく跳ね上がっているため,「$L \sim 1,000$ なら安全」とは言いがたい. 最下位ビットも $L = 1000$ 付近で最大となっている. $L$ が大きくなれば乱数性も高くなると言われていたが, この検定ではそのような結果は得られず, 効果的な改良であるとは言えなかった.

## 6 まとめと今後の課題

C 言語の rand() 関数,random() 関数,Lagged Fibbonacci 生成法,ran_array 生成法について検定を行った. 今後は,ran_array について $L$ を大きくしても乱数性の悪さは改善しないことを示すことを目指して, パラメータを変えて検定を行いたい. また今回扱えなかった MT,SFMT などの乱数についても同様の検定を行っていきたい.

## 謝辞


山形大学の西村拓士先生に有益なアドバイスをいただいたことを感謝いたします.


## 参考文献


[1] D. E. knuth : The Art Of Computer Programming Volume 2, 3rd edition

[2] 松本眞, 西村拓士：間違いだらけの擬似乱数選び http://www.soi.wide.ad.jp/class/20010000/slides/03/index_1.html

[3] 津野義道：ランダム・ウォーク 乱れに潜む不思議な現象 牧野書店,2002

[4] 東京大学教養学部統計学教室 編：統計学入門 東京大学出版会,1991