電気学会論文誌 D（産業応用部門誌）
IEEJ Transactions on Industry Applications
Vol.143 No.2 pp.80–86　DOI: 10.1541/ieejias.143.80

論　文

気象データとLFSRによる乱数生成手法の評価

非会員　千葉　歩武*　正　員　市川　周一**a)

Evaluation of Random Number Generator Utilizing Weather Data and LFSR

Ayumu Chiba*, Non-member, Shuichi Ichikawa**a), Member

（2022年3月14日受付，2022年6月2日再受付）

Entropy sources (e.g., physical phenomena) are essential for true random number (TRN) generation. An unpredictable random number (URN) generator was previously proposed, which uses processor internal registers as its entropy sources. Another study proposed to integrate a linear feedback shift register (LFSR) in a processor, and sample it to generate a URN sequence. The entropy source of this URN is the fluctuation of sampling period. The current study proposes to use weather data as the entropy source for URN generation, where the sampling period is modified by the wind direction data. The derived URN sequences passed the Diehard test when the least sampling period $\beta > 29$ with a 32-bit LFSR. It also passed the NIST test when the weather data were accessed with an appropriate hash function when β was 32.

キーワード：乱数，URNG, TRNG, LFSR
Keywords: random number, URNG, TRNG, LFSR

1. はじめに

セキュリティやシミュレーションなど多くの分野で，乱数生成は必須の技術となっている。乱数には，物理現象から生成される真性乱数（TRN；True Random Number）と，確定的アルゴリズムにより数値的に生成される疑似乱数（PRN；Pseudo Random Number）がある。TRNの生成にはハードウェア（TRNG；True Random Number Generator）が必須であるが，PRNはソフトウェアでも生成することができる。

Suciu ら[1]は，TRNとPRNの中間的な乱数としてURN（Unpredictable Random Number）を提案した。PRNはアルゴリズムと初期値から将来の値を予測可能であるが，URNではプロセッサの内部状態や割込みのタイミングなどを利用して乱数を生成する。URNは生成時の環境に依存するため，実質的に予測不可能になる。正岡ら[2]は，ソフトプロセッサに128ビットのLFSR（Linear Feedback Shift Register）を追加し，適切な条件下でLFSRのサンプリングを行うことによりURNが生成できることを示した。このURNG（Unpredictable Random Number Generator）のエントロピー源は，LFSRを読み出す間隔の揺らぎである。

正岡ら[2]は，実システムでURNGを実装評価し，128ビットLFSRで生成したURNが乱数テストに合格することを示した。しかしURN生成に必要なLFSRの仕様（ビット数や帰還多項式）は詳細に検討されていない。そこで鴨狩ら[3]は，LFSRの仕様とサンプリング間隔を変化させて乱数品質を調査し，適切な設計要件・使用条件を調査した。適切な帰還多項式を採用したとき，LFSRの長さは48ビット以上，サンプリング間隔は32サイクル以上必要であることが分かった。さらにサンプリング間隔の揺らぎにより乱数品質が向上することも示した。

本研究では，正岡ら[2]や鴨狩ら[3]の結果を踏まえて，自然情報をエントロピー源としたURNGについて検討する。特に無料で入手可能な気象観測データに着目し，それをエントロピー源として生成した乱数の品質を乱数テストで評価する。

なお本稿は，著者らの研究会発表[4]に加筆修正を施した

a) Correspondence to: Shuichi Ichikawa. E-mail: ichikawa@ieee.org
* 豊橋技術科学大学 電気・電子情報工学課程
〒441-8580　愛知県豊橋市天伯町雲雀ヶ丘1-1
Electrical and Electronic Information Engineering Course, Toyohashi University of Technology
1-1, Hibarigaoka, Tampaku-cho, Toyohashi, Aichi 441-8580, Japan
** 豊橋技術科学大学 電気・電子情報工学系
〒441-8580　愛知県豊橋市天伯町雲雀ヶ丘1-1
Department Electrical and Electronic Information Engineering, Toyohashi University of Technology
1-1, Hibarigaoka, Tampaku-cho, Toyohashi, Aichi 441-8580, Japan

ものである。

2. 背　景

TRNG では，ラッチやリングオシレータなどのハードウェアがエントロピー源として必須である。URNG においても，Suciu ら[1]はプロセッサのパフォーマンスカウンタ，正岡ら[2]はソフトプロセッサに付加した LFSR を利用しており，いずれも何らかのハードウェア支援を前提としている。

URNG を特別なハードウェアなしで実現できれば，PRN より予測性の低い乱数（URN）をソフトウェアで生成できるようになり，低コストかつ応用範囲の広い乱数生成手法をユーザに提供できる。近年のデバイスには通信機能が標準で搭載されており，インターネット上には多くのデータがリアルタイムで提供されているので，そこからエントロピーを抽出することができれば，特別なハードウェア支援なしで URNG を実現できると期待される。

ネット上で提供されているリアルタイムデータの一例は，株価である。金融経済学における効率的市場仮説（EMH；Efficient Market Hypothesis）では，市場が完全に情報的に効率的である場合，株価の推移は無記憶性ランダムウォークとなり，予測できないことが知られている[5]。つまり EMH が成り立つ市場の株価推移は，乱数のエントロピー源として利用可能である。逆に，株価推移の乱数性をテストすることにより，現実の市場において EMH を検証する研究も行われている[6][7]。株価の時系列データはオンラインで入手可能であるため，株価データをエントロピー源として利用することが考えられる。株式市場は世界中にあり，銘柄も多く，提供間隔も短いため，エントロピーの総量は多くなると期待される。しかし多くは有料のサービスであり，自由な利用は制限される。また EMH が完全に成立していない場合，データ量に対するエントロピーが少ない可能性もある。

もう一つの例は，毎日多量に生み出される SNS（Social Networking Service）のデータである。Fernández ら[8]は，Twitter のメッセージ列をエントロピー源として乱数列を生成することを提案した。Fernández らは，連続した 2 個のメッセージの長さを比較し，その大小をビットとして符号化することを提案した。また SHA-3 によりメッセージ列をハッシュする手法も提案している。この 2 つの方法で生成した乱数列は，いずれも NIST テスト[9]をパスした。メッセージ長による二進符号化は取り出せるエントロピーが小さく，SHA-3 によるハッシュは計算量が多いという問題点はあるが，ネット上の時系列データとソフトウェアで URNG を構成する点は本研究と目的が共通である。ただし SNS のメッセージは人間の意志で生成され，個人レベルで制御できるため，乱数出力を恣意的にコントロールされる危険性が大きいと考えられる。

本研究では，無料かつ広くアクセス可能なエントロピー源として，気象データに着目した。気象データは公共機関により一般公開されており，無料で使用可能である。観測点も世界中に多く存在し，データは日々生成され蓄積されるので，新たなデータで新しい乱数を生成し続けることができる。エントロピー源は自然であり，株価や SNS に比べれば，人間による恣意的な操作を受けにくい。サーバ侵入によるデータ偽造は可能だが，それは株価や SNS でも同じである。特定地点のデータだけを用いて乱数を生成すると，その地点の観測機器を操作することにより乱数出力への干渉が可能になるが，地理的に離れた複数地点のデータを総合的に用いることにより，攻撃のコストを増加させ攻撃者の経済的動機を失わせることができる。

なお，気象データの統計的性質を調べた研究は存在するが（例えば Konecny ら[10]），調べた範囲内で，気象データをエントロピー源とする乱数生成手法は発見できなかった。後述する通り気象データのエントロピー生成率は高くないので，あえて気象データを採用する動機がなかったと考えられる。それに対して本研究では，鴨狩ら[3]の手法を組み合わせて生成率を改善している。

3. 気象データの使用法

気象庁は，全国約 150 地点で地上気象観測を行っている[11]。地上気象観測とは，気圧，気温，湿度，風，降水，積雪，雲，視程，天気，日照，その他の気象現象を自動または目視で観測することをいう。さらにきめ細かく状況を把握するために，地域気象観測システム（アメダス）では全国約 1300 地点で観測を行っている。

気象庁のデータ[12]では，気象データを要素（降水，風速等）毎に csv ファイルとしてダウンロードできる。データは最新のデータと過去のデータに分かれており，最新のデータは当日，過去のデータは前日までのデータを含んでいる。最新のデータは一定時間毎に更新される（降水は 10 分毎，風や気温は毎時 50 分頃）。

継続的に乱数を生成する際には，最新データを使用してエントロピー源を随時更新することが望ましい。しかし本研究では処理方法による乱数品質の比較を行うため，エントロピー源は同じデータ（不変）とし，2021 年 6 月 28 日から過去 1 年分の風向・風速データ（1 時間毎）を用いた。他のデータ（気温，降水等）も使用可能であるが，値の変化に乏しいため，今回は風向・風速のデータに絞って検討を行った。

観測地点は中部地方の地上観測地点（Table 1）を使用し，各観測地点毎に 1 つの csv ファイルをダウンロードした。各回の観測データには品質情報が付加されており，値（0～8）が大きいほど信頼性が高い。本研究では品質情報 4 以下のデータは読み飛ばした（無視した）。

風速（m/s）は小数点以下 1 桁まで記録されている。そこで風速を 10 倍して整数化し，値を 16 進表現して符号化した。風速 1.5 m/s 以下では 16 進 1 桁，1.6 m/s 以上で 2 桁，25.6 m/s 以上で 3 桁となる。時系列の風速データを結合し，各桁（0～F）の出現頻度をまとめたものを Fig. 1 に示す。風速は小さい値の頻度が大きいため，1～4 の頻度が大きく

Table 1. Observation Points.

Prefecture	Location
Aichi	Irago, Nagoya
Gifu	Gifu, Takayama
Yamanashi	Kawaguchi-ko, Kofu
Niigata	Aikawa, Takada, Niigata
Shizuoka	Ajiro, Irou-zaki, Omae-zaki, Shizuoka, Hamamatsu, Mt. Fuji, Mishima
Ishikawa	Kanazawa, Wajima
Nagano	Iida, Karuizawa, Suwa, Nagano, Matsumoto
Toyama	Toyama, Fushiki
Fukui	Tsuruga, Fukui

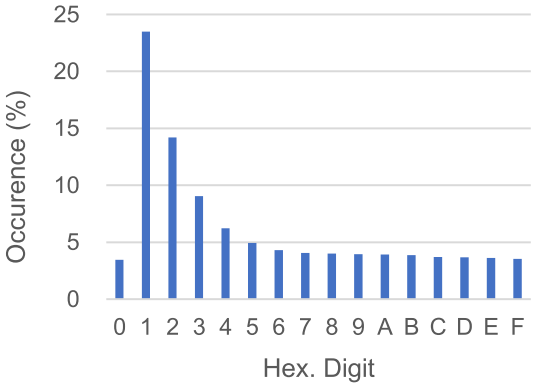


Fig. 1. Occurence (wind velocity).

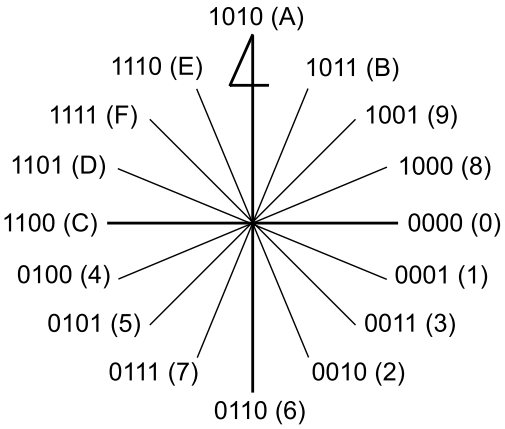


Fig. 2. Encoding of wind direction.

なっている。0 の頻度が低いのは，風速の値の大きさで桁数を変えたためである。この頻度分布で求めたエントロピーは 3.649 ビットであり，理想値 4 ビットの 91.2%である。

風向は文字で記録されており，16 方位で表現されている（例：北北西）。これを Fig. 2 に示すルールで 4 ビットの二進数に置き換えた。符号化はグレイコードになっており，隣り合った風向のハミング距離を 1 にしている。通常の二進符号化では，わずかな変化で 4 ビット変化する（例：7 → 8），180 度の変化が 1 ビット差である（例：0 → 8）など，直感に反するためである。ただし風速が 0.2 m/s 以下の場合は，風向が「静穏」となる。風が無ければ風向もないため，そのデータは読み飛ばす（無視する）。時系列の風向データを結合し，各桁（0～F）の出現頻度をまとめたものを Fig. 3 に示す。この頻度分布で求めたエントロピーは 3.976 ビットであり，理想値の 99.4%である。

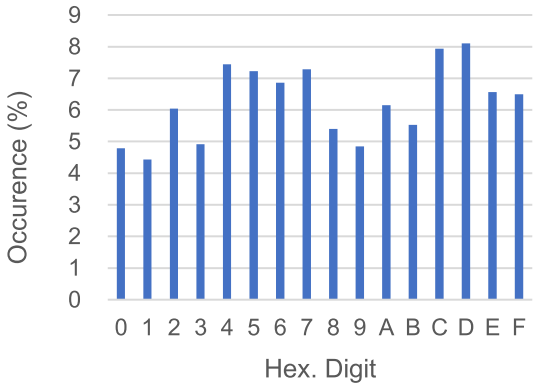


Fig. 3. Occurence (wind direction).

以上の予備的評価から，以下，本研究では風向データを用いて URN 生成を試みる。他のデータを用いた URN 生成，あるいは複数データを組み合わせた URN 生成も興味深いが，それは本研究の範囲外とし，今後の課題とする。

4. 乱数列の生成

品質不明の乱数では安心して使用することができないが，乱数品質を評価するには一般に大きなデータ量が必要になる。例えば Diehard テスト[13]では，約 10^8 ビットのデータ量が必要である。また NIST テスト[9]では，1 回のテストで約 10^6 ビットを使用し，そのテストを 1000 回以上行うことが推奨されているので，合計 10^9 ビット程度が必要になる。

前章で説明した風向データは，1 年分で約 9×10^5 ビットであり，そのままでは乱数テストを実行できない。観測地点を増やすことは可能であるが，100～1000 倍に増やすのは実用性に欠ける。公開データを使用することが前提なので，測定間隔を変えることはできない。

そこで本研究では，先行研究[2][3]に従い，LFSR を用いて URN 生成を行うことにする。具体的には，LFSR のサンプリング間隔を気象データで変動させることにより，気象データより長い URN 系列を生成する。

8 ビット LFSR の例を Fig. 4 に示す。LFSR は帰還多項式に対応するタップシーケンスで表現され，Fig. 4 のタップシーケンスは [8, 6, 5, 4] である。鴨狩ら[3]の研究結果を踏まえて，本研究では 32 ビット LFSR（タップシーケンス [32, 30, 17, 12, 3, 1][14]）を用いる。正岡ら[2]は論理回路として LFSR を実装したが，本研究の LFSR はソフトウェアとして実装することにより，特段のハードウェア支援がないシステムでも利用可能となる。

LFSR の値を読む際に，n 回目のサンプリング間隔 $S(n)$ を以下の式で決める。サンプリング間隔とは，$(n-1)$ 回目のサンプルのあと帰還多項式を $S(n)$ 回適用して，n 回目のサンプルを行うという意味である。

$$S(n) = \alpha(n) + \beta \cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots (1)$$

ここで定数 β は基本となるサンプリング間隔，$\alpha(n)$ は風向

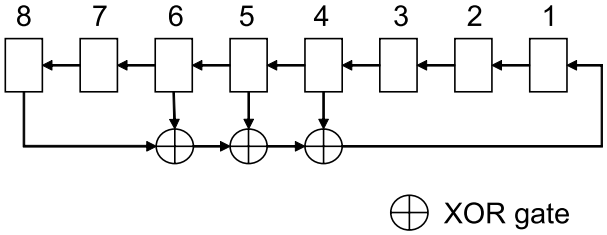


Fig. 4.　An example of 8-bit LFSR [8,6,5,4] [3].

Table 2.　Diehard evaluation criteria [2].

Decision	Condition
PASS	$0.005 \le p < 0.995$
WEAK	$0.000001 \le p < 0.005$, or $0.995 \le p < 0.999999$
FAIL	$p < 0.000001$, or $0.999999 \le p$

データから生成する揺らぎで $0 \le \alpha(n) \le 15$ である。32 ビット LFSR は 1 回のサンプリングで 4 バイトのデータを生成するため，2.5×10^6 回で Diehard テストに必要な乱数列が生成できる。2.5×10^6 回分の $\alpha(n)$ を生成するため，風向データは 12 年分に拡張した。期間以外のデータ利用方法は変えていない。

揺らぎ $\alpha(n)$ の生成方法を検討するため，以下の 4 つの方法を用いて URN を生成し，Diehard テストで品質を評価した。風向データは，時系列順に 2.5×10^6 個の要素を持つ配列 *wdata* に格納する。各要素のデータは 16 方位のグレイコード（0～15）である。風向は短時間で変わりにくいので，風向データには時間依存性があると考えられる。そのため時系列順に風向データを用いると（Method 1），乱数性が低下する可能性がある。そこで Method 2～4 では，配列 *wdata* へのアクセス順序に簡単なハッシュ関数を適用した。

Method 1.　時系列順に風向データを使用する。$m = 2.5 \times 10^6$ とし，揺らぎ $\alpha(n) = wdata[(n-1) \bmod m]$ とする。

Method 2. 除算法のハッシュ関数で風向データの利用順序を変更する。$\alpha(n) = wdata[8(n-1) \bmod m]$，ここで $m = 2499999$ とする。（係数 8 と互いに素な m で，なるべく大きい値を使用した。）

Method 3. 除算法のハッシュ関数で風向データの利用順序を変更する。$\alpha(n) = wdata[x(n-1) \bmod m]$，ここで係数 $x = (\sqrt{5}-1) \times 2^{22}$，$m = 2^{21}$ としている。この方法は乗算法で用いる係数 $(\sqrt{5}-1)/2$ を除算法に適用し，固定小数点演算で実現したものともいえる。

Method 4. 乗算法のハッシュ関数[15]で風向データの利用順序を変更する。$\alpha(n) = wdata[\lfloor m(A(n-1) \bmod 1) \rfloor]$，ここで $A = (\sqrt{5}-1)/2$，$m = 2^{21}$。

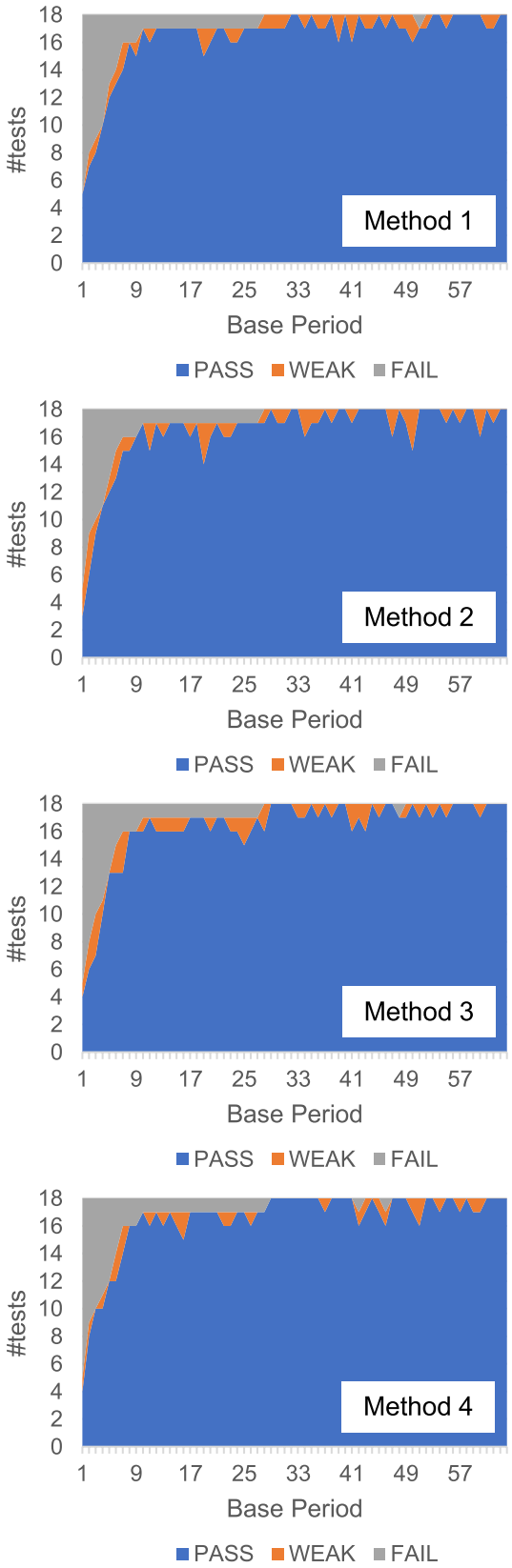


Fig. 5.　Diehard test results of Method 1, 2, 3, and 4.

5.　乱数列の評価

〈5･1〉 **Diehard テスト**　本章では，先行研究[2][3]と同じく Diehard テスト[13]を利用して乱数品質の評価を行う。Diehard テストは NIST テスト[9]ほど厳格ではないが，必要なデータ量が少ないため試行錯誤の多い研究段階に適している。

Diehard テストは全 18 種のテストからなり，各テストで 1～100 個（合計 313 個）の p 値を出力する。合格判定の基準は定められておらず，利用者の判断に委ねられているの

Table 3. NIST test results of Method 1, 2, 3, and 4.

Test	Method 1		Method 2		Method 3		Method 4	
	p-value	proportion	p-value	proportion	p-value	proportion	p-value	proportion
Frequency	0.886162	0.981	0.353733	0.981	0.794391	0.982	0.235589	0.983
BlockFrequency	0.249284	0.988	0.739918	0.993	0.781106	0.987	0.185555	0.986
Runs	0.653773	0.993	0.471146	0.989	0.530120	0.987	0.435430	0.986
LongestRun	0.270265	0.992	0.006063	0.983	0.388990	0.983	0.242986	0.989
Rank	* 0.000000	* 1.000	0.378705	0.985	0.678686	0.994	0.420827	0.987
FFT	0.457825	0.989	0.010834	0.987	0.889118	0.992	0.064822	0.994
NonOverlappingTemplate	148/148	* 147/148	148/148	* 147/148	148/148	148/148	148/148	148/148
OverlappingTemplate	0.067300	0.994	0.657933	0.989	0.757790	0.992	0.707513	0.981
Universal	0.097159	0.987	0.440975	0.988	0.406499	0.985	0.020973	0.985
LinearComplexity	0.844641	0.995	0.272977	0.990	0.087692	0.988	0.442831	0.990
Serial	2/2	2/2	2/2	2/2	2/2	2/2	2/2	2/2
ApproximateEntropy	0.536163	0.990	0.581082	0.986	0.429923	0.992	0.482707	0.990
CumulativeSums	2/2	* 0/2	2/2	* 1/2	2/2	2/2	2/2	2/2
RandomExcursions	8/8	8/8	8/8	8/8	8/8	8/8	8/8	8/8
RandomExcursionsVariant	18/18	18/18	18/18	18/18	18/18	18/18	18/18	18/18

で，本研究でも先行研究[2]の基準に従って，以下のように乱数品質を評価する。

入力が理想的乱数であれば p 値は区間 [0,1) で均等に分布することが期待されるので，Table 2 に示した基準で各 p 値の成功（PASS）／弱成功（WEAK）／失敗（FAIL）を判定する。FAIL の発生確率（期待値）は 2×10^{-6} なので，入力が乱数であれば（ほぼ）発生しない。WEAK の発生確率（期待値）は 1×10^{-2} なので，WEAK が 1%程度発生することは正常である。

単純に 313 個の p 値について PASS/WEAK/FAIL の個数を示すと，多くの p 値を出力するテストのウエイトが大きく見えてしまう。そこで以下の方法により，各テストの結果を判定する。各テストで出力される p 値の個数が 9 個以上であれば，得られた p 値の分布が一様であるかどうかの判定を Kolmogorov-Smirnov 検定により行い，得られた p 値を Table 2 に示した基準で判定する。テストの出力する p 値が 9 個未満であれば，以下に述べる方法で結果を判定する。各テストで出力される p 値に，ひとつでも FAIL が含まれれば，そのテストは FAIL。出力される p 値に FAIL はなく WEAK が含まれれば，そのテストは WEAK。出力される p 値が全て PASS であれば，そのテストは PASS とする。

こうして計算した全 18 テストの結果（PASS/WEAK/FAIL の内訳）により，乱数列の品質を評価する。

Fig. 5 は，4 章の Method 1～4 で生成した URN を Diehard テストで評価した結果を示したものである。横軸は基本サンプリング間隔であり，(1) 式の β に相当する。縦軸はテスト数で，内訳を PASS/WEAK/FAIL で示している。いずれも基本間隔 β が増加すると乱数品質は向上し，$\beta \geq 30$ では概ね FAIL がなくなる様子が観察できる。ただし $42 \leq \beta \leq 51$ で散発的な FAIL が観察され，乱数品質に問題が生じている。これは，鴨狩ら[3]の報告でも見られる現象であり，LFSR の長さや帰還多項式を検討することにより解決できる可能性が高い。本研究では 32 ビット LFSR（1 種類）だけを検討したが，今後は LFSR の仕様を変更して広く検討することが望ましい。

〈5･2〉 NIST テスト　本章では，Diehard テストよりデータ量の多い NIST テスト[9]で乱数品質を確認する。Diehard テストの結果から，基本間隔 $\beta > 29$ が必要であると予想される。また LFSR の長さが 32 ビットであることも考慮し，本章では $\beta = 32$ として Method 1～4 により URN の生成を行った。

NIST テストの出力結果を Table 3 に示す。FAIL したテストには “*” 印をつけた。複数回行われるテストは，（PASS した回数）／（テスト回数）という形で結果を表し，1 回でも FAIL していたらそのテストを FAIL と判定した。

Table 3 に示す通り，Method 1 と 2 は NIST テストに合格できず，Method 3 と 4 は合格した。この結果は $\beta = 32$ の結果であり，β を大きくすれば Method 1 と 2 も合格する可能性がある。逆に $\beta = 29$ で実験すると，Method 1～4 の全てで不合格となった。ちなみに $\beta = 29$ では Method 1～4 の全てで BlockFrequency テストを FAIL した。一方 $\beta = 32$ の Method 1 と 2 では NonOverlappingTemplate や CumulativeSums で FAIL するので，$\beta = 29$ とは FAIL するテストが異なっている。

以上の結果から，Diehard テストであれば時系列の風向データ（Method 1）でも合格できるが，NIST テストは時系列データでは合格できないことが分かった。適切なサンプリング間隔 β とハッシュ関数を採用することにより，NIST テストに合格することができる（Method 3，4）。

6. 検　討

〈6･1〉 予測可能性　本章では，提案手法により生成した乱数列が，攻撃者により予測あるいは操作される可能性について検討する。

本論文で提案した乱数生成手法には，以下の要素が含まれている。

（1） 気象データ
（2） 観測地点と時間範囲
（3） ハッシュ関数

（4） 基本サンプリング間隔（β）

（5） LFSR の帰還多項式と内部状態

項目（1）は公開情報であるが，以下に説明する通り，項目（2）～（5）は非公開として運用すべきである。

本手法のエントロピー源である気象データ（項目（1））は公開物であり，攻撃者にも入手可能である。しかし，どの観測地点の，どの期間のデータを扱うか（項目（2））を知らない限り，攻撃者は気象データを実際の攻撃に利用できない。

仮に，直前に使用された観測地点と時刻が攻撃者に把握されたとしても，次に使用されるデータはハッシュ関数（項目（3））によって選ばれる。次のハッシュ値が推測できない限り，攻撃者は気象データを利用することができない。5 章ではハッシュ関数が乱数列の品質に影響することを示したが，耐攻撃性の観点からも質の良いハッシュ関数を採用し，ハッシュアルゴリズムは非公開とすべきである。

仮に攻撃者が，次に乱数生成に使用するデータ値 $\alpha(n)$ を特定できたとしても，基本サンプリング間隔 β（項目（4））を知らなければ，LFSR を読み出すタイミングが特定できない。また，そもそも LFSR の帰還多項式と内部状態（項目（5））を知らない限り，攻撃者は乱数出力を計算できない。

以上の説明から，天気予報あるいは統計分析によって気象データ（気温・風向・風速など）が一定の確度で予測できるとしても，乱数出力の予測は困難であることが理解できるであろう。

本研究で提案する乱数生成手法は URNG の一種である。疑似乱数生成器（PRNG）より予測可能性は低いが，真性乱数生成器（TRNG）ではない。本研究の手法は既存の PRNG や TRNG を置き換えるものではなく，補完するものである。ユーザは，各自の目的や状況に合わせて適切な乱数生成手法を選ぶ必要がある。

〈6･2〉 品質保持　提案手法のエントロピー源は気象データであり，自然の揺らぎによって値が変動する。しかしながら，地形の影響で風向に偏りが生じたり，季節により降雨の有無が偏る等，エントロピーが減少する可能性は排除できない。また観測点の機器故障により欠測が続けば，エントロピーが 0 になる可能性もある。エントロピーが減少すると，当然，乱数出力の品質が低下することが予想される。

本研究では，3 章および 4 章で示した通り，複数の観測地点からの気象データを用いて乱数を生成している。複数地点のデータを併用することにより，機器故障や地形の影響など局地的問題を回避し，より安定性と信頼性の高い乱数生成を行うことができる。またハッシュ関数によりデータを選択することで，異なる観測地点のデータが混合された状態で利用される。これにより LFSR のサンプリング間隔 $S(n)$ が一定値になりにくいよう工夫されている。

以上のような工夫を施したとしても，不測の事態によりエントロピーが枯渇する可能性がある。Yang ら[16]は，TRNG 出力（乱数列）の品質を保障するため，動作中にオンラインで出力を検査し，品質低下時に警告を出すことを提案した。このように使用時に乱数品質を検査することも一考に値する。

7. おわりに

今回は提案手法の実現可能性を確かめるため，12 年分の風向データを使用した。しかし今後は，より少量のデータから URN を生成する手法について検討したい。また本研究では風向データのみに注目したが，今後は他のデータ（風速，気温，降水等）の利用も検討することが望ましい。複数のデータ源を組み合わせて用いる方法について，詳細な検討が望まれる。

謝　辞

本研究の一部は JSPS 科研費 20K11733 の支援による。

文　献

（1） A. Suciu, S. Banescu, and K. Marton: “Unpredictable random number generator based on hardware performance counters”, Digital Information Processing and Communications (ICDIPC 2011), pp.123–137, Springer-Verlag (2011)

（2） H. Masaoka, S. Ichikawa, and N. Fujieda: “Random Number Generation from Internal LFSR and Fluctuation of Sampling Interval”, *IEEJ Trans. IA*, Vol.141, No.2, pp.86–92 (2021) (in Japanese)
正岡秀崇・市川周一・藤枝直輝：「内蔵 LFSR とサンプリング間隔の揺らぎを利用した乱数生成手法」, 電学論 D, Vol.141, No.2, pp.86–92 (2021)

（3） H. Kamogari and S. Ichikawa: “An investigation of random number generation based on the internal LFSR”, The Papers of Technical Meeting on Innovative Industrial System, IEE Japan, IIS-21-011 (2021) (in Japanese)
鴨狩滉斗・市川周一：「内蔵 LFSR を用いた乱数生成に関する検討」, 電学次世代産業システム研, IIS-21-011 (2021)

（4） A. Chiba and S. Ichikawa: “Random Number Generation from Weather Data and LFSR”, The papers of technical meeting on Innovative Industrial System, IEE Japan, IIS-22-019 (2022) (in Japanese)
千葉歩武・市川周一：「気象データと LFSR による乱数生成手法」, 電学次世代産業システム研, IIS-22-019 (2022)

（5） E.F. Fama: “The behavior of stock-market prices”, *Journal of Business*, Vol.38, No.1, pp.34–105 (1965)

（6） J.R. Doyle and C.H. Chen: “Patterns in stock market movements tested as random number generators”, *European Journal of Operational Research*, Vol.227, No.1, pp.122–132 (2013)

（7） M.A. Noakes and K. Rajaratnam: “Testing market efficiency on the Johannesburg Stock Exchange using the overlapping serial test”, *Annals of Operations Research*, Vol.243, No.1, pp.273–300 (2016)

（8） N. Fernández, F. Quintas, L. Sánchez, and J. Arias: “Social noise: Generating random numbers from twitter streams”, *Fluctuation and Noise Letters*, Vol.14, No.1, art. No.1550012 (2015)

（9） A. Rukhin, et al.: “A Statistical Test Suite for Random and Pseudorandom Number Generators for Cryptographic Applications”, NIST SP 800-22 (Rev. la) (2010)

（10） P. Konecny and D. Pustka: “Random variation and correlation of the weather data series evaluation and simulation using bounded histograms”, *MATEC Web of Conferences*, Vol.107, No.00085 (2017)

（11） 気象庁：「気象観測統計の解説」, https://www.data.jma.go.jp/obd/stats/data/kaisetu/index.html

（12） 気象庁：「過去の気象データ・ダウンロード」, https://www.data.jma.go.jp/risk/obsdl/index.php

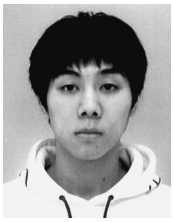
（13） G. Marsaglia: “Diehard battery of tests of randomness (Archived)”, https://web.archive.org/web/20160125103112/http://stat.fsu.edu/pub/diehard/

（14） M. Živković: “A table of primitive binary polynomials”, *Mathematics of Computation*, Vol.62, No.205, pp.385–386 (1994)

（15） D.E. Knuth: “The Art of Computer Programming”, Vol.3, Addison-Wesley (1973)

（16） B. Yang, V. Rožić, N. Mentens, W. Dehaene, and I. Verbauwhede: “TOTAL: TRNG on-the-fly testing for attack detection using Lightweight hardware”, Proc. 2016 Design, Automation & Test in Europe Conference & Exhibition (DATE), pp.127–132, Dresden, Germany (2016)

千　葉　歩　武（非会員） 2022 年豊橋技術科学大学電気・電子情報工学課程卒業。同年，同大学大学院電気・電子情報工学専攻博士前期課程入学。



市　川　周　一（正員） 1985 年東京大学理学部卒業。1987 年同大学大学院理学系研究科修士課程修了。1987 年新技術事業団，1991 年三菱電機（株），1994 年名古屋大学工学部助手。1997 年豊橋技術科学大学工学部講師。同助教授，准教授を経て，2011 年沼津工業高等専門学校制御情報工学科教授。2012 年より豊橋技術科学大学大学院工学研究科教授。現在に至る。理学博士。IEEE（senior member），電子情報通信学会（シニア会員），ACM，情報処理学会，各会員。