7K-01

Ergodic PRNG
- エルゴード性を用いた周期なし擬似乱数生成器 -

池田樹生* 檀裕也 †

1 はじめに

数学的擬似乱数生成器には周期が存在し, その周期の長さが擬似乱数生成器のひとつの性能の指標とされる. 周期を持たない擬似乱数を作ることは可能であり, それは内部状態が無限である必要がある. 内部状態がデータ的に大きくなる方法は簡単に周期を無限にすることが可能であるが, コンピュータのメモリの上限に依存するため, 実質的には有限であり現実的ではない.

杉田洋氏による先行研究 [5] において, 無理数回転を用いた乱数生成器では無限周期の擬似乱数列が生成できる. しかし, この乱数生成器は厳密には無理数を近似した有理数により演算を行っており, 実質的には周期は有限である. **Ergodic** PRNG は, 無理数をプログラム上に定義することにより, 真に周期が無限である擬似乱数列を生成することができたと同時に, エルゴード性を利用し数列生成に反転操作を加えることで乱数性の向上が確認できた.

2 理論

Definition 1. l_x は x 座標の最大値 (今回の場合は $\sqrt{2}$), l_y は y 座標の最大値 (今回の場合は 1) とし, またシード値の範囲を [0, 1) の半開区間とするとき, 擬似乱数列 En を以下のように定義する.

$$E_1 = seed \tag{2.0.1}$$
$$E_{n+1} = (E_n + l_x) \bmod l_y \tag{2.0.2}$$

この数列は周期を持たない.

2.1 エルゴード性による幾何学的な反転

よりランダム性を高めるため, Ergodic PRNG では値の反転を行う. 幾何学的には, $\frac{\pi}{4}$ radian の角度で移動を開始した点が l_x または l_y に到達した場合にビリヤードのように反転をする. つまり, この擬似乱数列は辺の長さがそれぞれ l_x, l_y の長方形の中の原点を $\frac{\pi}{4}$ radian の角度で移動開始し, ビリヤードの玉のように反転をし続けながら移動する点の座標 (x, y) における y が $l_y\sqrt{2}$ の距離を移動するごとにプロットしたものである. この反転処理はグラフのトーラス構造化と等価であり, 無限周期は保たれる.

反転処理を追加した擬似乱数列 E_n の漸化式は以下の通りである:

$n = 1$ のとき,

$$E_{n+1} = seed, x\text{は反転しない.} \tag{2.1.1}$$

x が反転しておらず, $\left\lfloor \frac{E_n + l_x}{l_y} \right\rfloor$ が偶数のとき,

$$(E_n + l_x) \bmod l_y, x\text{は反転しない.} \tag{2.1.2}$$

x が反転しておらず, $\left\lfloor \frac{E_n + l_x}{l_y} \right\rfloor$ が奇数のとき,

$$l_y - \{(E_n + l_x) \bmod l_y, x\text{は反転する.} \tag{2.1.3}$$

x が反転しており, $\left\lfloor \frac{l_y - E_n + l_x}{l_y} \right\rfloor$ が奇数のとき,

$$(l_y - E_n + l_x) \bmod l_y, x\text{は反転しない.} \tag{2.1.4}$$

x が反転しており, $\left\lfloor \frac{l_y - E_n + l_x}{l_y} \right\rfloor$ が偶数のとき,

$$l_y - (E_n + l_x) \bmod l_y, x\text{は反転する.} \tag{2.1.5}$$

3 Ergodic PRNG への適用

ここからは, 実際に擬似乱数生成器アルゴリズムに適用をしていく. また, これ以降の定義は Backus-Naur form[1][3] を用いて行う. Haskell[2] による実装はhttps://github.com/Tatsuki-I/ErgodicPRNGにまとめてある.

3.1 無理数型

Haskell では新たなデータ構造を作るのは容易である. ここでは, Ergodic PRNG を実装するにあたり, はじめに無理数を扱うデータ型を実装する.

$$a + b\varphi. \tag{3.1.1}$$

ただし $a, b \in \mathbb{Q}$ かつ $\varphi \in \mathbb{R} \setminus \mathbb{Q}$ とする.

Backus-Naur form による定義は以下の通りである:

$$\begin{aligned}
< digit > &::= (0|1|2|3|4|5|6|7|8|9) \\
< digits > &::= < digit > \\
&\quad | \quad < digit >< digit > \\
< rational > &::= < digits > \% < digits > \\
< exp > &::= < rational > + < rational > \varphi
\end{aligned}$$

*松山大学経営学部経営学科情報コース 4 年
†松山大学経営学部経営学科教授

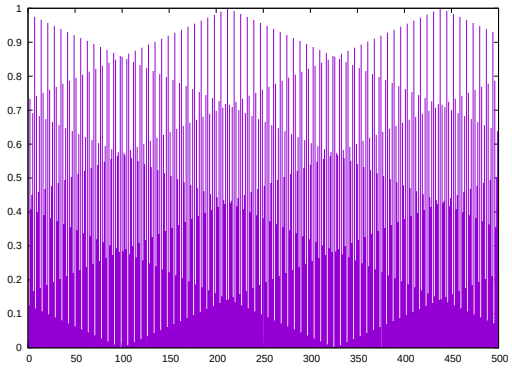

Figure 1: $l_x = \sqrt{2}$

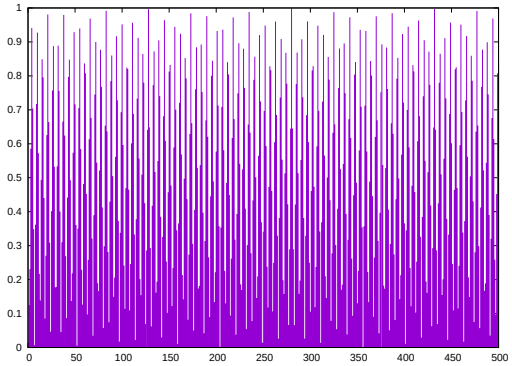

Figure 2: $l_x = \frac{1+\sqrt[3]{12}}{2}$

このように, 無理数を含む数式の係数のみをデータとして計算し, 最後に無理数の近似値を掛けることにより, 内部状態を有理数に丸めることなく計算が可能である. これは真に周期のない擬似乱数列を生成するのに重要である.

3.2 生成器の初期化

次に, 擬似乱数生成器の初期化について考える. 式(2.0.1) の定義ではシード値をそのまま最大値で除算したが, これにはある問題がある. それは, シード値の数値が近い場合にほとんど同じ乱数列を生成してしまうことである. そのため, Ergodic PRNG では Xorshift[4] を用いてシード値のランダマイズを行う

$$E_1 = \frac{Xorshift(seed)}{2^{63}-1} \tag{3.2.1}$$

3.3 最適なパラメータの選択

Ergodic PRNG では, x 軸の最大値である l_x, y 軸の最大値である l_y, φ の 3 つのパラメータが必要である. これらのパラメータは予め定数として定義しておく.

l_y は計算量を考えた上で, $l_y = 1$ とする.

図 1 は $l_x = \sqrt{2}$ の場合の出力結果をプロットしたものである. このグラフからも見てとれる通り, わかりやすく外周期が出てしまっている. $l_x = \sqrt{2}$ はランダム性が低いと言える.

図 2 は $l_x = \frac{1+\sqrt[3]{12}}{2}$ として出力結果をプロットしたグラフである.

目立った外周期が出ていないことが分かる. 今回複数のパラメータを試したが, $l_x = \frac{1+\sqrt[3]{12}}{2}$ が最も外周期が目立たない結果となった.

今回は $\varphi = \sqrt[3]{12}$, $l_x = \frac{1+\sqrt[3]{12}}{2}$ とする.

$$l_y = 1 \tag{3.3.1}$$
$$\varphi = \sqrt[3]{12} \tag{3.3.2}$$
$$l_x = \frac{1+\varphi}{2} \tag{3.3.3}$$

4 まとめ

Ergodic PRNG は周期の無い数学的擬似乱数生成器である.

周期が無いという特性を持つ一方で, ランダム性や生成速度は RDRAND や Xorshift に劣る結果となった. 一方で, 内部的に無理数であるため, 同一のアルゴリズムから任意精度の乱数列を取り出すことが可能であるというメリットもある.

生成速度と乱数性に関しては今後の大きな課題であるが, 乱数性の解決の手段として, Ergodic PRNG をそのまま擬似乱数列として使用するのではなく, シード値としてのみ利用するという方法がある. Ergodic PRNG の出力結果をシード値として, 何らかの擬似乱数生成器により擬似乱数列を生成することにより, 既存の擬似乱数生成器の周期を無くすことが可能である.

References

[1] John W. Backus, Friedrich L. Bauer, Julien Green, Charles Katz, John McCarthy, Peter Naur, Alan J. Perlis, Heinz Rutishauser, Klaus Samelson, Bernard Vauquois, Joseph Henry Wegstein, Adriaan van Wijngaarden, and Michael Woodger. “Report on the Algorithmic Language ALGOL 60”. *Numerische Mathematik*, 2(1):106–136, 1960.

[2] Simon Peyton Jones. *“Haskell 98 Language and Libraries: The Revised Report”*. Cambridge University Press, 2003.

[3] Donald E. Knuth. “Backus Normal Form vs. Backus Naur Form”. *Communications of the ACM*, 7(12):735–736, 1964.

[4] George Marsaglia et al. “Xorshift RNGs”. *Journal of Statistical Software*, 8(14):1–6, 2003.

[5] 杉田洋. “無理数回転による擬似乱数生成 (数値計算アルゴリズムの現状と展望 II)”. 数理解析研究所講究録, 915:146–156, 1995.