# Generatory liczb pseudolosowych i modelowanie Monte Carlo

Michał Saturczak
Informatyka Stosowana, Akademia Górniczo-Hutnicza
Metody Numeryczne

24 maja 2024

# 1 Wstęp teoretyczny

Generatory liczb pseudolosowych (ang. pseudorandom number generators, PRNG) stanowią fundament wielu dziedzin nauki i techniki, umożliwiając symulację zjawisk losowych w środowisku deterministycznym. Mimo pozornej sprzeczności, generatory te produkują sekwencje liczb, które zdają się być losowe, ale w rzeczywistości są generowane według ściśle określonych algorytmów. Kluczowe cechy PRNG obejmują równomierność rozkładu, długość cyklu oraz brak korelacji między kolejnymi wartościami.

Modelowanie Monte Carlo to szeroko stosowana technika numeryczna, która wykorzystuje liczby pseudolosowe do przybliżania rozwiązań problemów matematycznych, fizycznych czy inżynierskich. Metoda ta opiera się na wielokrotnym losowaniu wartości z odpowiednich rozkładów prawdopodobieństwa oraz analizie statystycznej uzyskanych wyników. Dzięki temu możliwe jest szacowanie wartości oczekiwanych, całek, prawdopodobieństw czy optymalizacja parametrów modeli.

Zastosowania generatorów liczb pseudolosowych i modelowania Monte Carlo są niezwykle szerokie. W naukach przyrodniczych, PRNG umożliwiają symulację zjawisk losowych, takich jak ruch cząsteczek czy rozpad promieniotwórczy. W ekonomii, MCS wykorzystywane jest do modelowania rynków finansowych, prognozowania cen akcji czy analizy ryzyka. W inżynierii, metoda ta znajduje zastosowanie w projektowaniu systemów, optymalizacji procesów czy analizie niezawodności.

# 2 Problem

## 2.1 Rozwiązanie problemu

- Napisano program w języku C++ z do rozwiązania problemów związanych z generowaniem liczb pseudolosowych oraz modelowaniem Monte Carlo
- Zapisano do trzech skoroszytów dane odpowiednio dla każdego generatora U1, U2 oraz U3.
- Zrobiono wykresy przedstawiające wartości dla poszczególnych generatorów
- Rozszerzono kod o obliczenie przybliżenia liczby $\pi$ metodą Monte Carlo
- Zapisano dane do skoryszytu i zrobiono wykres logarytmu błędu od liczby iteracji

## 2.2 Funkcje wyznaczające generatory U1, U2 oraz U3 i kod w języku C++ użyty do rozwiązania problemu z modelowaniem MonteCarlo

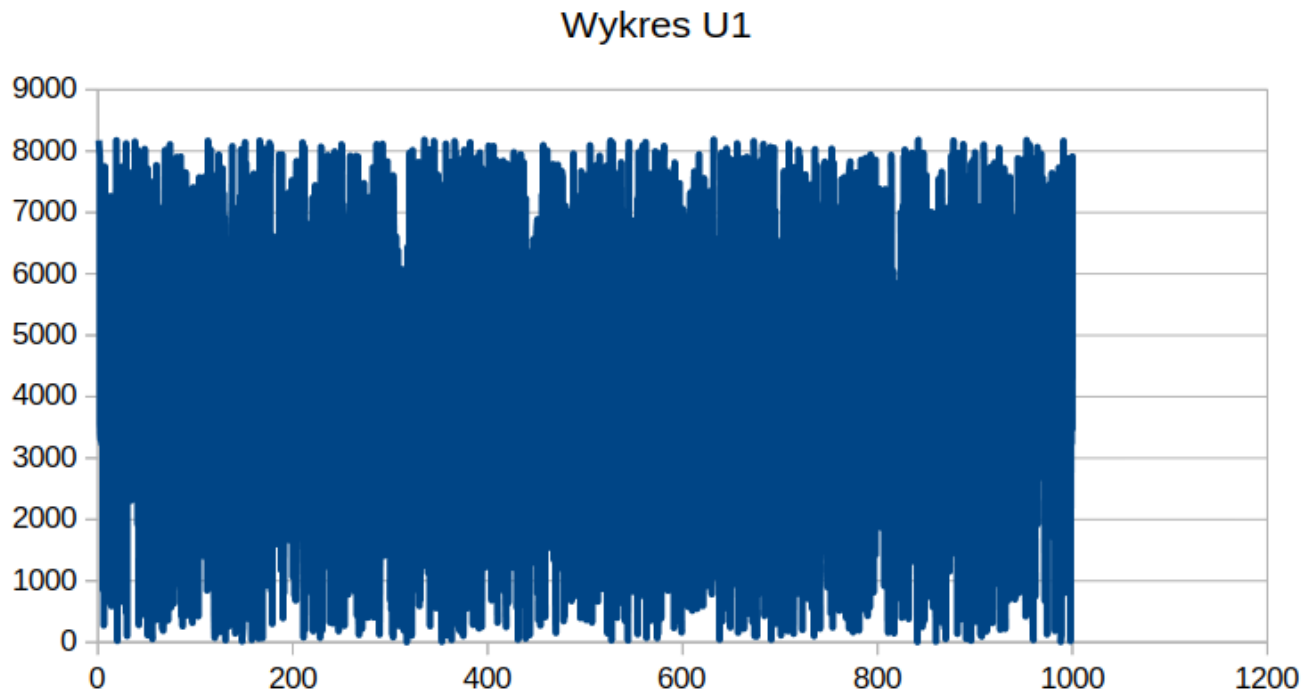
```
long int U1()
{
  static long int X = 10;
  const long int a = 17;
  const long int m = pow(2, 13) - 1;
  X = (a * X) % m;
  return X;
}

long int U2()
{
  static long int X = 10;
  const long int a = 85;
  const long int m = pow(2, 13) - 1;
  X = (a * X) % m;
  return X;
}

long int U3()
{
  static long int X[3] = {10, 10, 10};
```

```
  const long int a = 1176;
  const long int b = 1476;
  const long int c = 1176;
  const long int m = pow(2, 32) - 5;
  X[2] = (a * X[2] + b * X[1] + c * X[0]) % m;
  X[0] = X[1];
  X[1] = X[2];
  return X[2];
}

double U1_normalized()
{
  static long int X = 10;
  const long int a = 17;
  const long int m = pow(2, 13) - 1;
  X = (a * X) % m;
  return static_cast<double>(X) / (m - 1);
}

double U2_normalized()
{
  static long int X = 10;
  const long int a = 85;
  const long int m = pow(2, 13) - 1;
  X = (a * X) % m;
  return static_cast<double>(X) / (m - 1);
}

double U3_normalized()
{
  static long int X[3] = {10, 10, 10};
  const long int a = 1176;
  const long int b = 1476;
  const long int c = 1176;
  const long int m = pow(2, 32) - 5;
  X[2] = (a * X[2] + b * X[1] + c * X[0]) % m;
  X[0] = X[1];
  X[1] = X[2];
  return static_cast<double>(X[2]) / (m - 1);
}

....
  //modelowanie Monte Carlo
  const int n = 20000;
  const double l = 1.0;
  const double r = 1.0;
  int insideCircle = 0;
  double piApproximation;

  filePi << "Iteracja,Pi ,Błąd" << std::endl;

  for (int i = 1; i <= n; i++)
  {
    double x = l * U3_normalized();
    double y = l * U3_normalized();

    if (x * x + y * y <= r * r)
    {
```

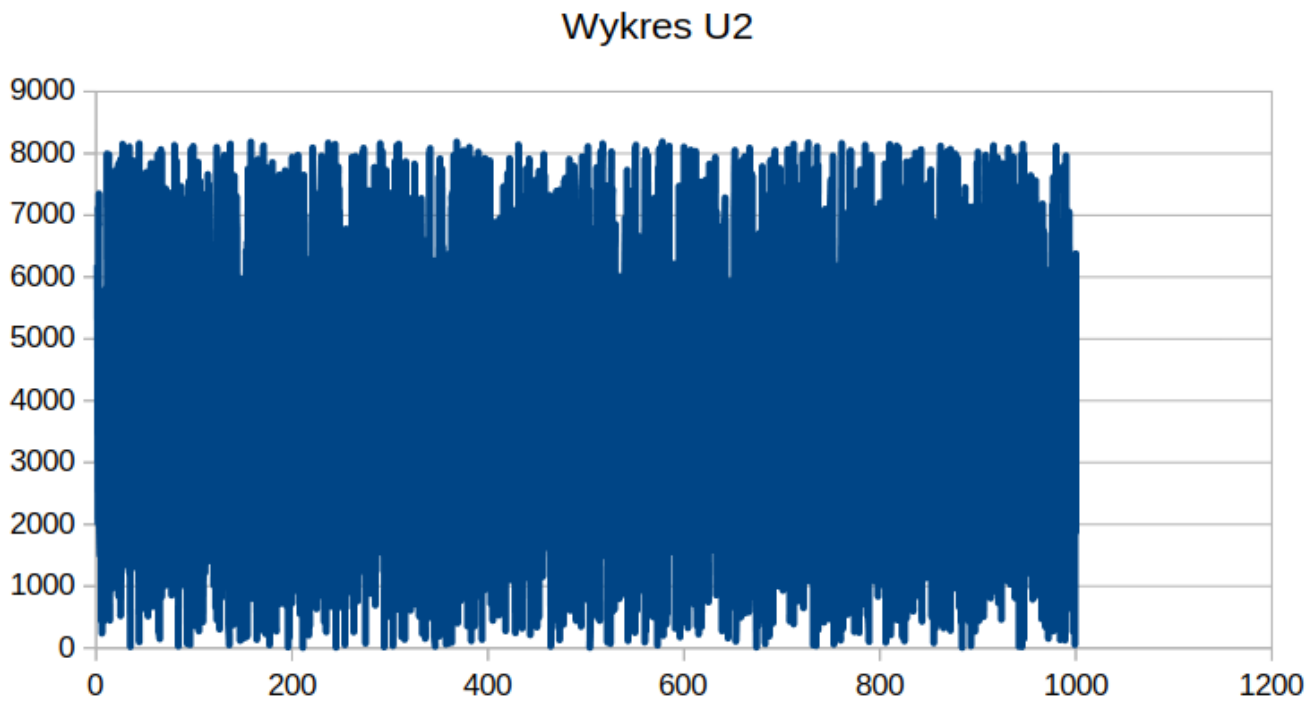
```
      insideCircle++;
    }

    if (i % 100 == 0)
    {
      piApproximation = 4.0 * static_cast<double>(insideCircle) / i;
      double error = std::abs(M_PI - piApproximation);
      filePi << i << "," << piApproximation << "," << error << std::endl;
    }
  }
```

# 3 Wyniki

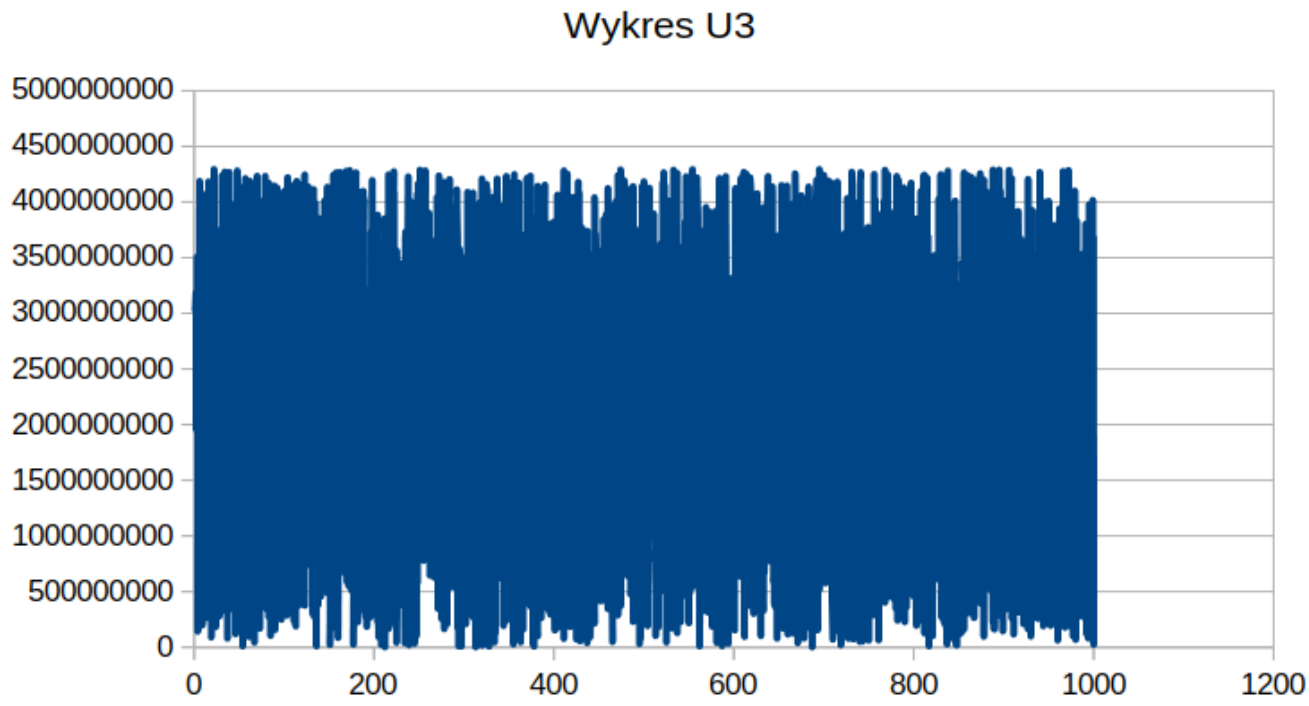
## 3.1 Wykres funkcji oraz dla generatora U1



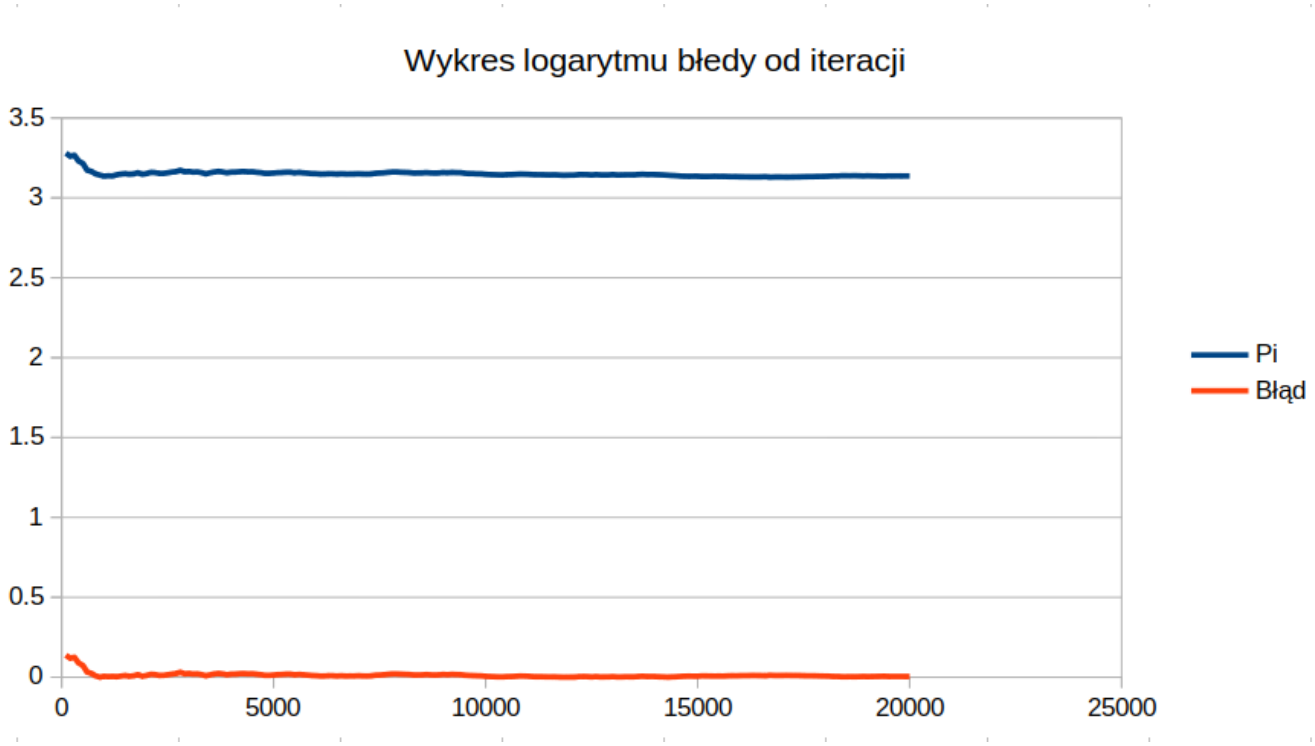
### 3.2 Wykres funkcji oraz dla generatora U2




### 3.3 Wykres funkcji oraz dla generatora U3



### 3.4 Wykres logarytmu błędu od liczby iteracji




## 4 Wnioski

Różne generatory liczb pseudolosowych mogą wykazywać zróżnicowaną jakość w zakresie równomierności rozkładu, długości cyklu oraz korelacji między kolejnymi wartościami. W analizowanym przypadku, generator U3 (mieszany generator liniowy) wydaje się generować liczby o bardziej równomiernym rozkładzie niż generatory U1 i U2 (proste generatory liniowe).

Metoda Monte Carlo umożliwia przybliżanie wartości trudnych do obliczenia analitycznie poprzez wielokrotne losowanie i analizę statystyczną wyników. W przykładzie z obliczaniem liczby Pi, metoda ta pozwoliła na uzyskanie coraz lepszego przybliżenia wraz ze wzrostem liczby iteracji.

Zbieżność metody Monte Carlo oznacza, że błąd przybliżenia maleje wraz ze wzrostem liczby iteracji. W analizowanym przykładzie zaobserwowano zmniejszanie się błędu wraz ze wzrostem liczby losowań punktów.

Generatory liczb pseudolosowych generują liczby deterministycznie, co oznacza, że sekwencja liczb jest powtarzalna. W niektórych zastosowaniach może to być istotne ograniczenie.