Akademia Górniczo-Hutnicza w Krakowie

Wydział Fizyki i Informatyki Stosowanej

Metody Numeryczne

Laboratorium 13: Całkowanie numeryczne przy użyciu kwadratur Gaussa

Andrzej Świętek

10.06.2024

Contents

1. Wstęp teoretyczny

Generatory liczb losowych można podzielić na dwie główne kategorie: fizyczne i komputerowe. Generatory fizyczne bazują na zjawiskach naturalnych, takich jak szumy w układach elektronicznych czy promieniowanie radioaktywne, które zapewniają generację ciągów liczb losowych nieskorelowanych ze sobą.

1.1. Generatory liniowe

Generatory liniowe tworzą ciągi liczb według następującego wzoru:

$$X_{n+1} = (a_1 X_n + a_2 X_{n-1} + \cdots + a_k X_{n-k} + c) \mod m,$$

gdzie $a_1, a_2, \ldots, a_k, c, m$ to ustalone parametry generatora, a $X_0, X_1, X_2, \ldots, X_k$ są ziarnami generatora. Wyróżniamy dwa rodzaje generatorów liniowych:

- **Generator multiplikatywny**, gdy $c = 0$,
- **Generator mieszany**, gdy $c \neq 0$.

1.2. Generatory multiplikatywne

Najprostszy generator multiplikatywny jest dany wzorem:

$$X_{i+1} = aX_i \mod m,$$

gdzie $k_i = \frac{aX_i}{m}$, dla $i \geq 1$.
Dla kolejnych wartości i wzory przyjmują formę:

$$\begin{aligned}
X_1 &= aX_0 - mk_1, \\
X_2 &= a^2 X_0 - m(k_2 + k_1 a), \\
X_3 &= a^3 X_0 - m(k_3 + k_2 a + k_1 a^2), \\
&\vdots \\
X_n &= a^n X_0 - m(k_n + k_{n-1} a + \cdots + k_1 a^{n-1}).
\end{aligned}$$

Ostatecznie można to zapisać jako:

$$X_n = a^n X_0 \mod m.$$

Widać, że wybór X_0 determinuje wszystkie liczby w generowanym ciągu, co wskazuje na brak losowości.

Okres generatora multiplikatywnego wyznaczony jest wzorem:

$$T = \min\{i : X_i = X_0, i > 0\}.$$

Maksymalny okres osiągamy, gdy spełnione są warunki:

$$a^{(m-1)/p} \neq 1 \mod m,$$

gdzie m jest liczbą pierwszą, a p jest czynnikiem pierwszym liczby $m - 1$.

1.3. Metoda Boxa-Mullera dla rozkładu normalnego

Funkcję gęstości prawdopodobieństwa w dwóch wymiarach można zdefiniować jako iloczyn dwóch funkcji gaussowskich:

$$f(x,y) = f(x) \cdot f(y) = e^{-\frac{x^2+y^2}{2}}, \quad x, y \in (-\infty, \infty).$$

Prawdopodobieństwo jest obliczane jako:

$$p(x,y) = f(x,y)\, dx\, dy.$$

Zmieniając współrzędne na biegunowe:

$$\begin{aligned} x &= r\cos(\theta), \quad r \in [0,\infty), \\ y &= r\sin(\theta), \quad \theta \in [0, 2\pi), \\ r^2 &= x^2 + y^2, \end{aligned}$$

i stosując prawo przenoszenia prawdopodobieństwa:

$$p = f(x,y)\, dx\, dy = f(r,\theta)\, r\, dr\, d\theta = r\, e^{-r^2/2}\, dr\, d\theta.$$

W celu ułatwienia całkowania wprowadzamy nową zmienną:

$$z = \frac{r^2}{2} \quad \rightarrow \quad dz = r\, dr, \quad z \in [0,\infty),$$

i otrzymujemy:

$$p(z,\theta) = e^{-z}\, dz\, d\theta.$$

Widzimy, że mamy rozkład wykładniczy, który można wygenerować w następujący sposób:

$$z = -\ln(1-U_1), \quad U_1 \in (0,1) \quad \rightarrow \quad r = \sqrt{2z} = \sqrt{-2\ln(1-U_1)}.$$

Kąt θ ma rozkład jednorodny, więc używamy generatora o rozkładzie jednorodnym:

$$\theta = U_2 \cdot 2\pi, \quad U_2 \in (0,1).$$

Dla pary (U_1, U_2) otrzymujemy parę liczb losowych (x, y) z rozkładu $N(0,1)$:

$$\begin{aligned} X &= \sqrt{-2\ln(1-U_1)}\cos(2\pi U_2), \\ Y &= \sqrt{-2\ln(1-U_1)}\sin(2\pi U_2). \end{aligned}$$

Tę metodę można przedstawić w formie algorytmu:

1. Wylosuj dwie liczby U_1 i U_2 z rozkładu jednostajnego na $(0,1)$.
2. Oblicz $r = \sqrt{-2\ln(1-U_1)}$.
3. Oblicz $\theta = 2\pi U_2$.
4. Wyznacz $X = r\cos(\theta)$.
5. Wyznacz $Y = r\sin(\theta)$.
6. Zwróć parę (X, Y) jako liczby z rozkładu normalnego $N(0,1)$.

2. Problem

2.1. Wylosować N = 2000 liczb pseudolowych z generatora multiplikatywnego

$$X_i = aX_{i-1} \mod m \tag{1}$$

$$x_i = \frac{X_i}{m + 1.0} \tag{2}$$

dla dwóch przypadków:

- $U_1(0,1) : a = 17, m = 2^{13} - 1, X_0 = 10$
- $U_2(0,1) : a = 85, m = 2^{13} - 1, X_0 = 10$

Dla każdego generatora należy utworzyć oddzielną funkcję, która powinna zwracać liczbę typu double tj. $x \in U(0,1)$
Wygenerowane liczby dla $U_1(0,1)$ i $U_2(0,1)$ zapisać w jednowymiarowych n-elementowych tablicach a następnie korzystając z nich należy sporządzić dwuwymiarowe wykresy kolejnych par: (x_i, x_{i+1}), (x_i, x_{i+2}), (x_i, x_{i+3}) dla obu generatorów.

2.2. Wylosować N = 2000 liczb pseudolowych z generatora multiplikatywnego U3(0, 1)

$$X_i = (1176 \cdot X_{i-1} + 1476 \cdot X_{i-2} + 1776 \cdot X_{i-3}) \mod (2^{32} - 5) \tag{3}$$

i unormować. Na starcie można przyjąć $X_0 = X_{i-1} = X_{i-2} = 10$. Sporządzić dwuwymiarowe wykresy kolejnych par: (x_i, x_{i+1}), (x_i, x_{i+2}), (x_i, x_{i+3}) dla generatora.

2.3. Rozkład jednorodny w kuli 3D: $K^3(0,1)$

1. Stosując metodę Boxa-Mullera należy utworzyć $N = 2000$ trzywymiarowych wektorów ($\vec{r_i} = [x_i, y_i, z_i]$) o rozkładzie normalnym

$$u_1, u_2, u_3, u_4 \in U_3(0,1) \tag{4}$$

$$x_i = \sqrt{-2\ln(1-u_1)}\cos(2\pi u_2) \tag{5}$$

$$y_i = \sqrt{-2\ln(1-u_1)}\sin(2\pi u_2) \tag{6}$$

$$z_i = \sqrt{-2\ln(1-u_3)}\cos(2\pi u_4) \tag{7}$$

Normalizacja wektorów:

$$\vec{r_i} \leftarrow \frac{\vec{r_i}}{\|\vec{r_i}\|_2} \tag{8}$$

Wygenerowane punkty w przestrzeni 3D powinny znajdować się teraz na powierzchni sfery. Zapisać wektory do pliku i sporządzić przykładowy rysunek 3D pokazujący ich rozkład na sferze, np. w Gnuplocie (poniżej część skryptu generującego wykres 3D)

```
set xyplane -1
set border 4095
splot 'dane.dat' u 1:2:3 w p
```

2. Dysponując rozkładem jednorodnym na sferze, dla każdego punktu (wektora w 3D) generujemy zmienną o rozkładzie jednomianowym $h_d(x) = d \cdot s^{d-1}, d = 3$ (d określa liczbę wymiarów) i $s \in (0, 1)$:

$$u_i \in U_3(0, 1) \tag{9}$$

$$s_i = (u_i)^{\frac{1}{d}} \quad d = 3 \tag{10}$$

$$\vec{r_i} = [s_i \cdot x_i, s_i \cdot y_i, s_i \cdot z_i] \tag{11}$$

Po wykonaniu powyższych czynności punkty $\vec{r_i}$ powinny być rozłożone równomiernie w kuli. Zapisać wektory $\vec{r_i}$ do pliku i sporządzić wykres 3D jak dla sfery.

3. Sprawdzamy czy rozkład punktów w kuli jest jednorodny tj. czy gęstość losowanych punktów jest stała w obszarze kuli. W tym celu dzielimy promień kuli na K = 10 podprzedziałów o równej długości, a następnie dla każdego punktu określamy jego przynależność do konkretnego przedziału.

Do pliku proszę zapisać wartości nj oraz gj dla: $N = 2000$ oraz $N = 10^4$ i $N = 10^7$. Różnice w wartościach g_1 proszę skomentować w sprawozdaniu w oparciu o zmianę wartości n_1 dla tych trzech serii danych.

$$\Delta = \frac{1}{K} \tag{12}$$

$$j = \left(\int \left(\frac{\|\vec{r_i}\|_2}{\Delta} \right) \right) + 1 \tag{13}$$

Objętość przedziałów:

$$R_j = \Delta \cdot j \tag{14}$$

$$R_{j-1} = \Delta \cdot (j - 1) \tag{15}$$

$$V_j = \frac{4}{3} \pi R_j^3 \tag{16}$$

$$V_{j-1} = \frac{4}{3} \pi R_{j-1}^3 \tag{17}$$

Gęstość punktów:

$$g_j = \frac{n_j}{V_j - V_{j-1}} \tag{18}$$

3. Implementacja

```
double gen_U1() {
    static long int x=10;
    int a = 17;
    int c = 0;
    long int m = pow(2,13)-1;
    x = (a*x + c) % m;

    return (double)x / (m + 1.0);
}
```

```
double gen_U2() {
    static long int x=10;
    int a = 85;
    int c = 0;
    long int m = pow(2,13)-1;
    x = (a*x + c) % m;

    return (double)x / (double)(m + 1.0);
}
```

```
const unsigned long long M3 = (1ULL << 32) - 5;
double gen_U3() {
    static long long x0 = 10, x1 = 10, x2 = 10;
    long long next = (1176 * x0 + 1476 * x1 + 1776 * x2) % M3;
    x2 = x1;
    x1 = x0;
    x0 = next;
    return static_cast<double>(next) / static_cast<double>(M3 + 1.0);
}
```

```
std::vector<Vector3D> generateVectors3D(int N) {
    std::vector<Vector3D> vectors;

    for (int i = 0; i < N; ++i) {
        double u1 = gen_U3();
        double u2 = gen_U3();
        double u3 = gen_U3();
        double u4 = gen_U3();

        double x = sqrt(-2.0 * log(1.0 - u1)) * cos(2 * M_PI * u2);
        double y = sqrt(-2.0 * log(1.0 - u1)) * sin(2 * M_PI * u2);
        double z = sqrt(-2.0 * log(1.0 - u3)) * cos(2 * M_PI * u4);

        vectors.push_back({ x, y, z });
    }

    return vectors;
}
```

```
void normalizeToSphereVolume(std::vector<Vector3D>& vectors, int
    dimensions = 3) {

    for (auto& vec : vectors) {
        double u = gen_U3();
```

```
        double s = pow(u, 1.0 / dimensions);
        double length = sqrt(vec.x * vec.x + vec.y * vec.y + vec.z * vec.z
    );
        vec.x = (vec.x / length) * s;
        vec.y = (vec.y / length) * s;
        vec.z = (vec.z / length) * s;
    }
}
```

```
void analyzeDensity(const std::vector<Vector3D>& vectors, int N) {
    const int K = 10;
    std::vector<int> counts(K, 0);
    double delta = 1.0 / K;

    for (const auto& vec : vectors) {
        double r = sqrt(vec.x * vec.x + vec.y * vec.y + vec.z * vec.z);
        int j = static_cast<int>(r / delta);
        if (j < K) {
            counts[j]++;
        }
    }

    std::ofstream outFile("data/density_analysis.dat");
    for (int j = 0; j < K; ++j) {
        double Rj = delta * (j + 1);
        double Rj1 = delta * j;
        double Vj = (4.0 / 3.0) * M_PI * pow(Rj, 3);
        double Vj1 = (4.0 / 3.0) * M_PI * pow(Rj1, 3);
        double V = Vj - Vj1;
        double g = counts[j] / V;
        outFile << j + 1 << " " << counts[j] << " " << g << "\n";
    }
    outFile.close();
}
```

```
    constexpr int N = 2000;
    std::vector<double> random_U1, random_U2, random_U3;
    for(int i = 0; i < N; i++){
        random_U1.push_back(gen_U1());
        random_U2.push_back(gen_U2());
        random_U3.push_back(gen_U3());
    }

    saveToFile(random_U1, "data/generator_1.csv");
    saveToFile(random_U2, "data/generator_2.csv");
    saveToFile(random_U3, "data/generator_3.csv");

    // KULA
    auto vectors = generateVectors3D(N);
    normalizeToSphereVolume(vectors);
    saveVectorsToFile(vectors, "data/vectors_volume.dat");
    analyzeDensity(vectors, N);

    save_data_for_generator_chart(random_U1, N, "data/wykres_gen_1.csv");
    save_data_for_generator_chart(random_U2, N, "data/wykres_gen_2.csv");
    save_data_for_generator_chart(random_U3, N, "data/wykres_gen_3.csv");
```

Listing 1: Wywołania funkcji generatorów w funkcji main()

4. Próbki Wyników

4.1. Generator 1

Table 1: Wyniki generatora $U_1(0,1)$

i	x_i
1	0.352783
2	0.997925
3	0.966675
4	0.435425
5	0.403076
⋮	⋮
1217	0.621216
1218	0.56189
1219	0.553223
1220	0.405884
1221	0.900757
⋮	⋮
1573	0.940918
1574	0.997437
1575	0.958374
1576	0.294312
1577	0.00390625
⋮	⋮
1995	0.0567627
1996	0.964966
1997	0.406372
1998	0.909058
1999	0.455811

4.2. Generator 2

Table 2: Wyniki generatora $U_2(0,1)$

i	x_i
1	0.820557
2	0.755737
3	0.245483
4	0.86853
5	0.833984
⋮	⋮
1025	0.591797
1026	0.308838
1027	0.254395
1028	0.626099
1029	0.224854
⋮	⋮
1500	0.709961
1501	0.354004
1502	0.0939941
1503	0.990356
1504	0.190552
⋮	⋮
1995	0.3302
1996	0.0704346
1997	0.987549
1998	0.951782
1999	0.911255

4.3. Generator 3

Table 3: Wyniki generatora $U_3(0,1)$

i	x_i
1	0.0121318
2	0.28225
3	0.850681
4	0.547522
5	0.765969
⋮	⋮
1001	0.760206
1002	0.290485
1003	0.912536
1004	0.0239211
1005	0.936183
⋮	⋮
1500	0.818561
1501	0.837861
1502	0.110897
1503	0.861892
1504	0.311013
⋮	⋮
1995	0.872709
1996	0.0576833
1997	0.639242
1998	0.820592
1999	0.983833

Table 4: Wektory Sfery

x	y	z
0.34423500389501976	-0.9301111049793441	0.1280452829568476
-0.6082147229004636	-0.7460017208242189	-0.2712126165472697
0.4481266002121964	0.7788049564188327	-0.4389138754240041
0.19460082853683155	0.3653289519487121	-0.9103105373447207
0.1660621077747417	0.9674467590592002	0.19097158099376962
-0.26175212178626717	-0.9651184336697215	-0.00567765191700521
0.5846691192006982	-0.8077398266939982	-0.07562005967618197
-0.32740912871472067	-0.6350566329124778	-0.6996472935902304
-0.4800687245596165	0.4202838550089748	0.769997078512963
⋮	⋮	⋮
0.1915958391996499	0.15438714680380858	0.9692552002971978
-0.7949693900230964	-0.4372681540416422	-0.4204999766793341
-0.31748982933345293	-0.763231070573661	-0.5627419845549098
-0.3463634393072663	0.8896168550430806	-0.2976814054228112
-0.34235716439214553	-0.5567132936714458	0.7568763972002694
-0.09341742672159284	-0.21708447922989674	0.9716725339651217
⋮	⋮	⋮
-0.3913878029288813	-0.3483472609041688	-0.8517451341446374
0.9270073174410665	0.19479428277787703	-0.32048809776303766
0.6756951785081353	-0.10615656688494808	-0.7294976415644266
-0.2746497699508575	0.6709868002756515	-0.6887265188170004
0.34154937367231697	0.7750742352991306	-0.5316050743921807

5. Wizualizacja Wyników

5.1. Generator 1

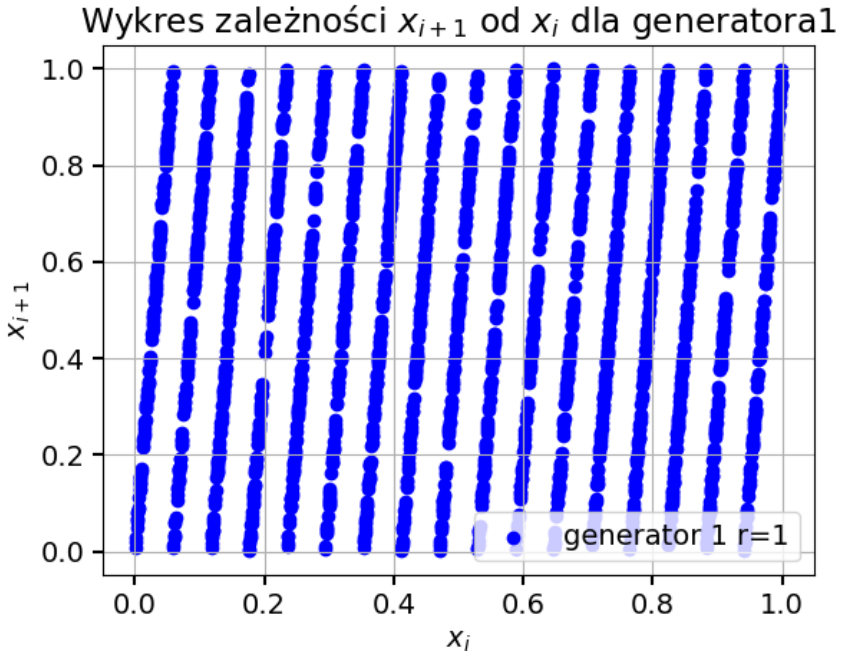

Figure 1: Generator o rozkładzie jednorodnym $U_1(0,1)$

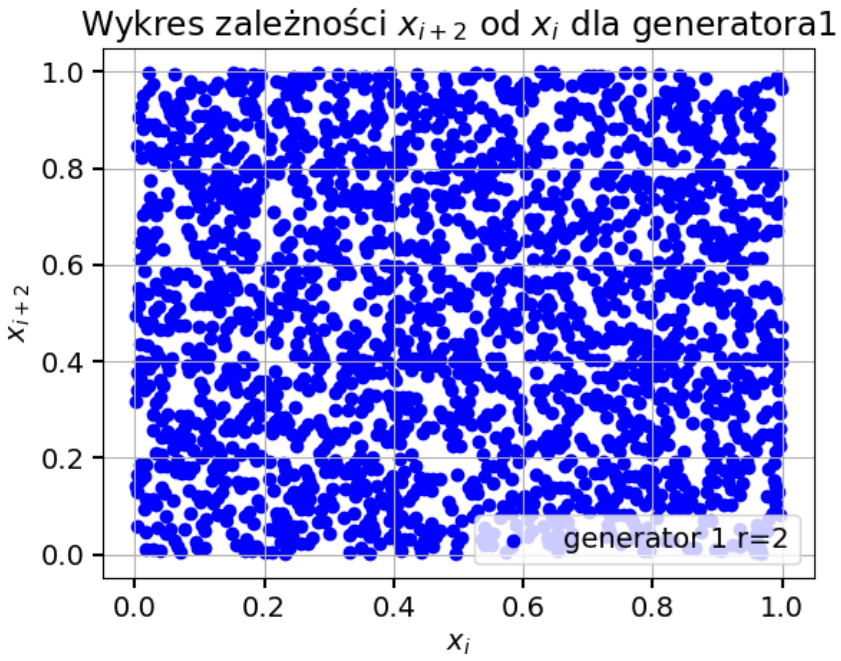

Figure 2: Generator o rozkładzie jednorodnym $U_1(0,1)$

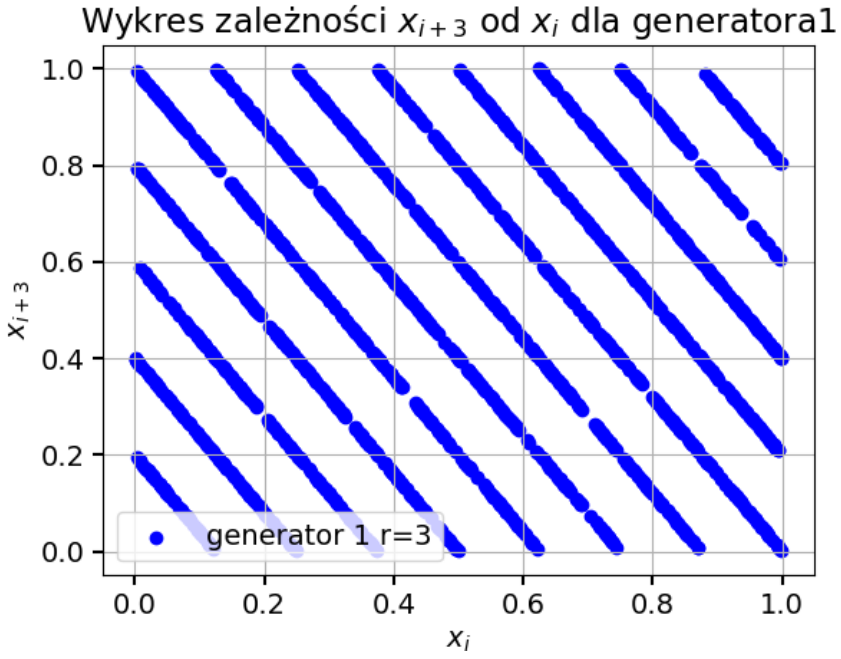

Figure 3: Generator o rozkładzie jednorodnym $U_1(0,1)$

5.2. Generator 2

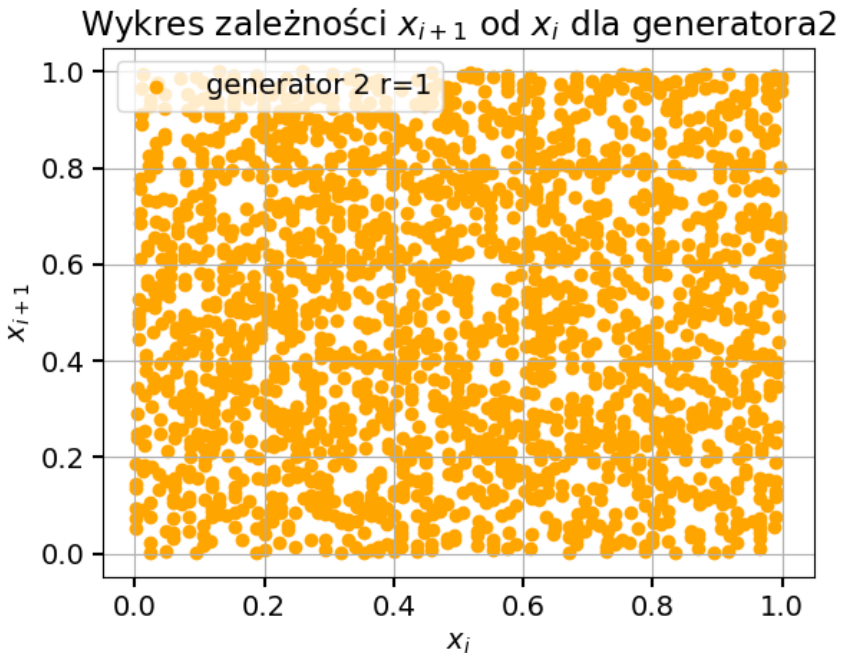

Figure 4: Generator o rozkładzie jednorodnym $U_2(0,1)$

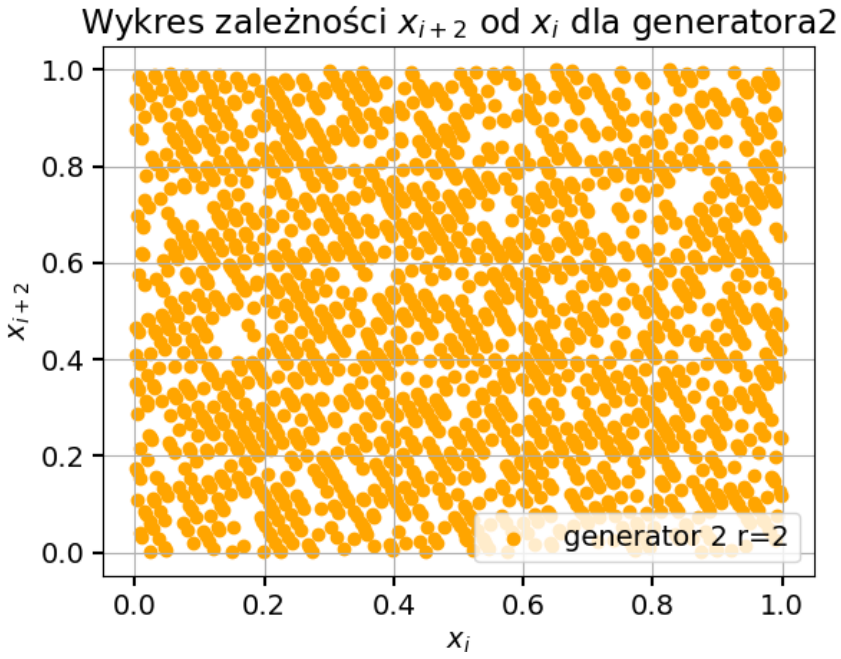

Figure 5: Generator o rozkładzie jednorodnym $U_2(0,1)$

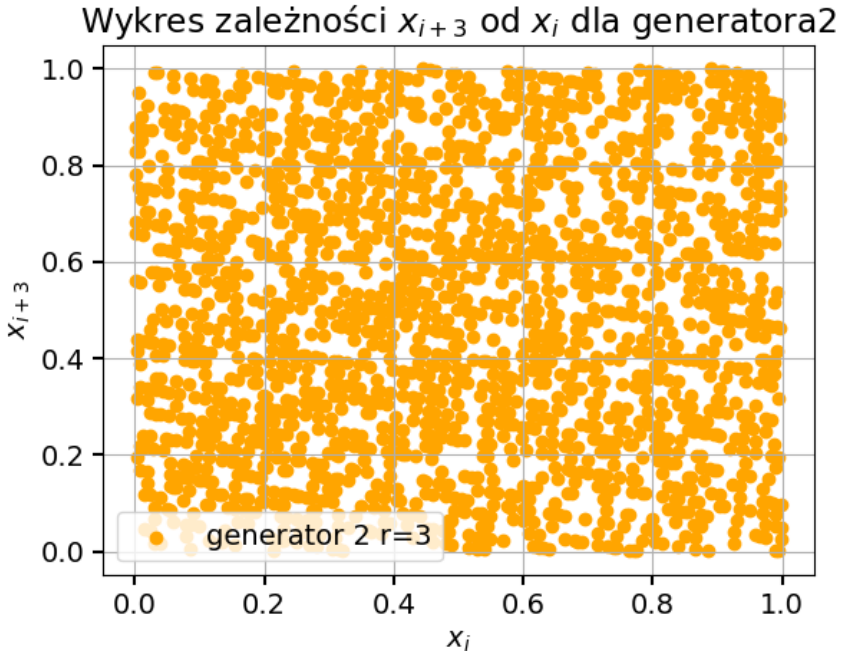

Figure 6: Generator o rozkładzie jednorodnym $U_2(0,1)$

5.3. Generator 3

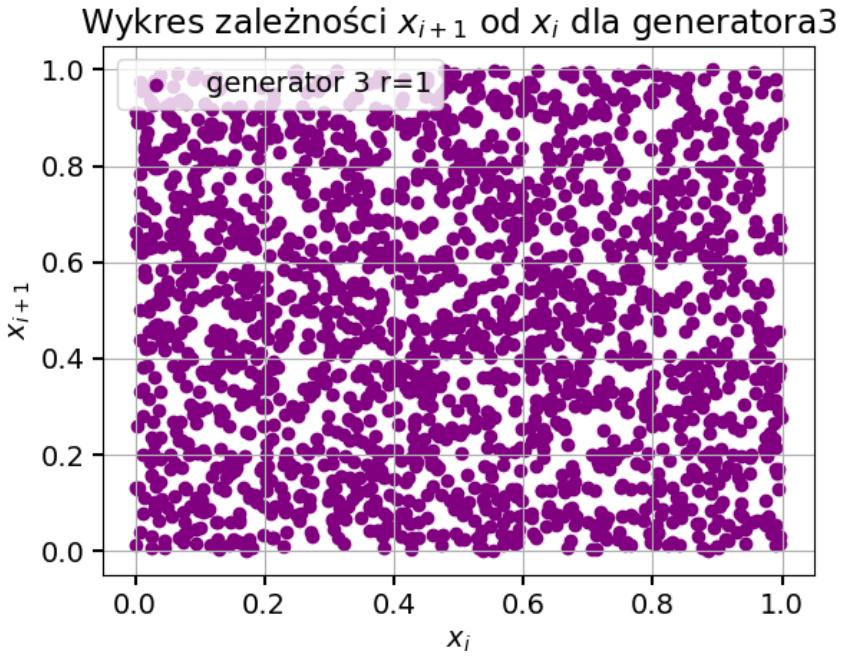

Figure 7:

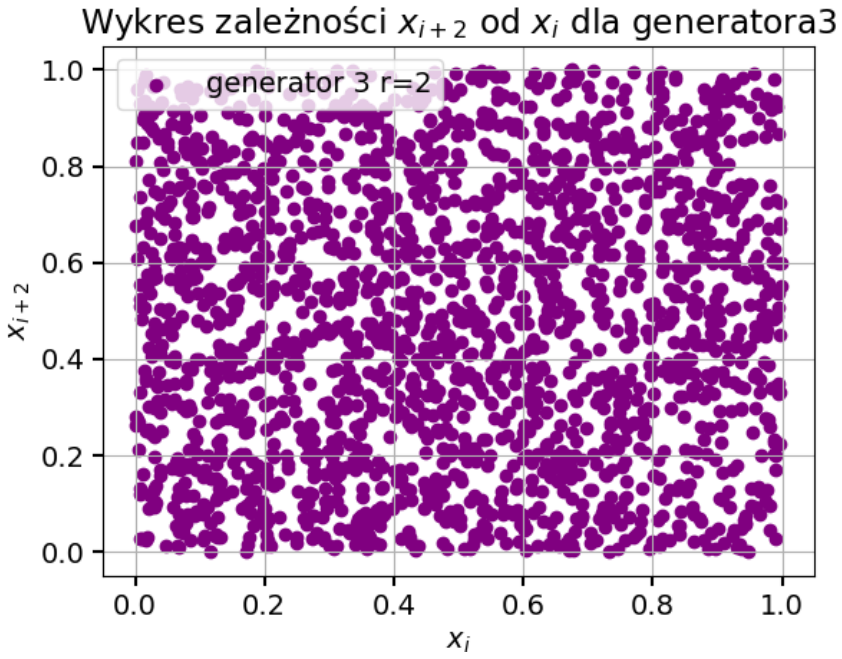

Figure 8:

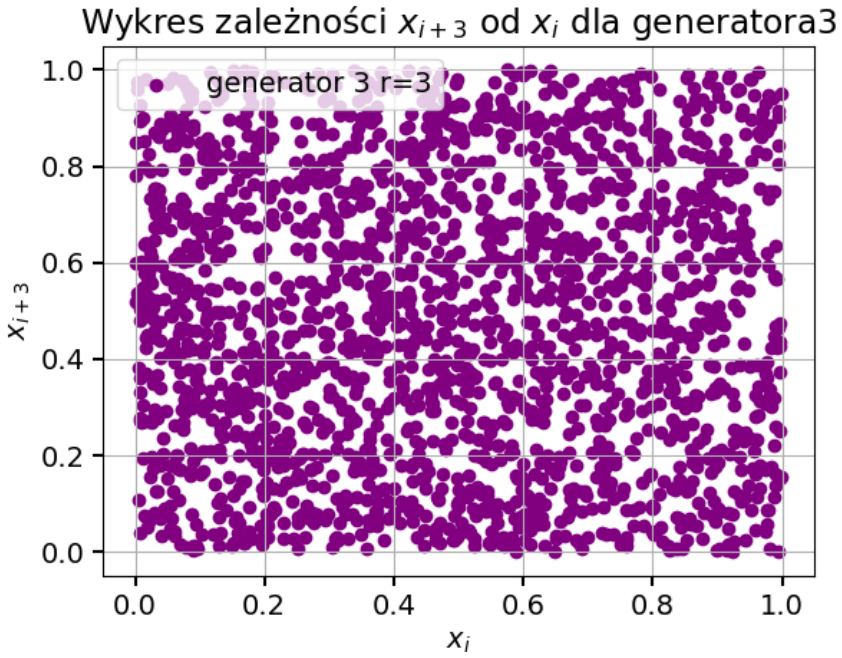

Figure 9: Wykres zależności x_{i+3} od x_i dla generatora nr. 3

5.4. Kula 3D

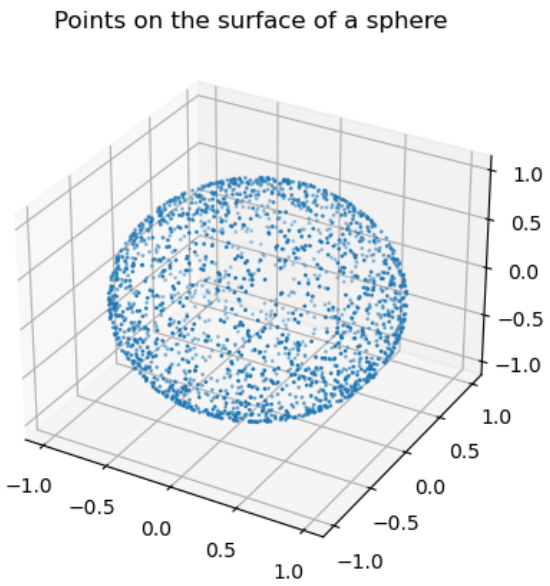

Figure 10:

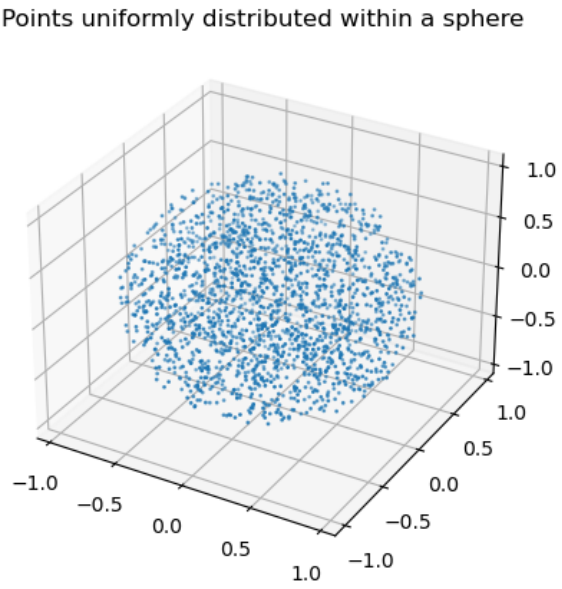

Figure 11:

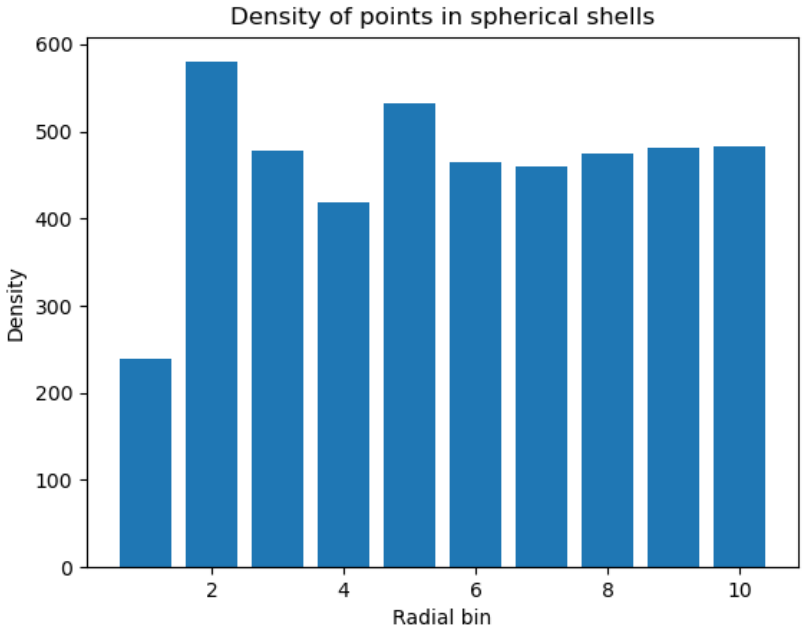

Figure 12:

6. Wnioski

Na wykresach prezentujących zależności dla generatorów U_1 i U_2 można zaobserwować relacje między punktami (x_i, x_{i+1}) oraz (x_i, x_{i+3}) dla generatora U_1, oraz między punktami (x_i, x_{i+2}) i (x_i, x_{i+3}) dla generatora U_2.

Widać to szczególnie w kontekście niewielkich wartości parametrów a oraz m, gdzie większe zagęszczenie punktów dla generatora U_2 wiąże się ze wzrostem parametru a. Ponadto, można dostrzec pewną okresowość w zachowaniu generatorów.

Parametry generatora U3 wykazują znacznie większą zróżnicowanie w porównaniu z poprzednimi. Zmieniono ziarno, a wartości parametrów a i m zostały zwiększone, co przyczyniło się do braku bezpośrednich zależności między generowanymi liczbami na wykresach.

Należy zauważyć, że generowanie prawdziwie losowych ciągów liczbowych za pomocą generatorów komputerowych jest niemożliwe. Wygenerowane liczby zawsze będą w pewien sposób skorelowane, jednakże poprzez odpowiednie dobranie parametrów możliwe jest uzyskanie ciągu, który będzie bardzo zbliżony do losowego.

Generator multiplikatywny generuje ciąg liczb o rozkładzie jednorodnym, jednakże przy mniejszej liczbie wylosowanych punktów można zauważyć odchylenia na histogramie.

Mimo że generatory liczb pseudolosowych znajdują szerokie zastosowanie w codziennym życiu, na przykład w kryptografii poprzez dodawanie kluczy (salt) do haseł, należy pamiętać, że ich działanie opiera się na algorytmach deterministycznych, co oznacza, że generowane ciągi nie są prawdziwie losowe.