ROCZNIKI POLSKIEGO TOWARZYSTWA MATEMATYCZNEGO
Seria III: MATEMATYKA STOSOWANA XXXV (1992)



A. Szczuka

Lublin

# Ciągi w pełni równomiernie rozłożone jako generatory liczb pseudolosowych

(*Praca wpłynęła do Redakcji 1.07.1991*)

**Streszczenie** W pracy rozważany jest problem komputerowej implementacji ciągów w pełni równomiernie rozłożonych (CUD). W części ogólnej pracy przedstawiono dziesięć przykładów ciągów CUD. W części szczegółowej przedstawiono komputerowe implementacje ciągów Starczenki i Korobowa. Na skonstruowanych generatorach przeprowadzono kilka testów.

**Wprowadzenie** Od przeszło 40 lat, od chwili pojawienia się pierwszych komputerów *(ENIAC)*, do rozwiązywania zadań metodami Monte Carlo korzysta się z programowych generatorów liczb losowych. Liczby przez nie wytwarzane powstające wg pewnych prostych algorytmów, nazywa się liczbami pseudolosowymi (a generatory generatorami liczb pseudolosowych). W pionierskich czasach generatorami zajmowali się między innymi von Neumann i Ulam.

W pracy koncentrujemy się na ciągach w pełni równomiernie rozłożonych (używamy również skrótu CUD), bo — ogólnie mówiąc — nie mają one dwóch głównych wad powszechnie stosowanych generatorów, a mianowicie

1) są nieokresowe (tzn. mają okres równy nieskończoność);
2) dla każdego naturalnego $s$ i dla każdego przedziału $I \subset [0,1]^s$, frakcja liczb CUDu, leżących w I, jest (w granicy) równa mierze Lebesgue'a tego przedziału. Ta okoliczność sprawia, że niektóre testy statystyczne (np.niektóre testy niezależności) są, przynajmniej asymptotycznie, automatycznie spełnione.

W rozdziale pierwszym przedstawiono definicję ciągu CUD oraz podano kilka ich konstrukcji. Zasadniczą częścią pracy jest rozdział drugi, w którym opisano ogólne problemy związane z implementacją CUDów. Przedstawiono

w nim również implementacje komputerowe dwóch wybranych konstrukcji takich ciągów: Starczenki (1959) oraz Korobowa (1960). W rozdziale trzecim przedstawiono szereg testów umożliwiających porównanie rozważanych implementacji.

Literatura na temat problemów związanych z generowaniem liczb pseudolosowych oraz metod Monte Carlo jest tak bogata, że zdecydowano się podać tylko prace cytowane. (Bogaty spis literatury zawierający 384 pozycje podaje Niederreiter, (1978)).

## 1. Ciągi w pełni równomiernie rozłożone.

DEFINICJA 1. Ciąg punktów $(R_n, n \geq 1)$, $R_n \in [0,1]^s$, $n = 1,2,\ldots$, nazywa się równomiernie rozłożony na kostce $[0,1]^s$, jeżeli dla każdego $x \in [0,1]^s$

$$\lim_{N\to\infty} \frac{\#\{1 \leq n \leq N : R_n \leq x\}}{N} = |x|$$

gdzie nierówność $x \leq y$ w $R^s$ jest rozumiana jako „nierówność dla każdej ze współrzędnych", $|x| = \prod_{j=1}^{s} x_j$ dla $x = (x_1, x_2, \ldots, x_s) \in [0,1]^s$ oraz $\#A$ oznacza moc zbioru $A$.

Weyl (1916) pokazał, że jeżeli ciąg $(R_n, n \geq 1)$, $R_n \in [0,1]$, $n = 1,2,\ldots$, jest równomiernie rozłożony na przedziale $[0,1]$, to dla każdej funkcji $f : [0,1] \to R^1$, całkowalnej według Riemanna,

$$\lim_{N\to\infty} \frac{1}{N} \sum_{n=1}^{N} f(R_n) = \int_0^1 f(x)dx \tag{1}$$

Własność (1) sugeruje, że dostatecznie długie odcinki ciągów równomiernie rozłożonych $(R_1, R_2, \ldots, R_n)$ mogą służyć do oszacowania wartości całki.

DEFINICJA 2. Ciąg punktów $(R_n, n \geq 1)$, $R_n \in [0,1]$, $n = 1,2,\ldots$, nazywa się s-równomiernie rozłożony, jeżeli ciąg

$$((R_n, R_{n+1}, \ldots, R_{n+s-1}), \quad n \geq 1)$$

jest równomiernie rozłożony na kostce $[0,1]^s$. Ciąg $(R_n, n \geq 1)$ nazywa się w pełni równomiernie rozłożony (CUD), jeżeli jest $s$-równomiernie rozłożony dla każdego naturalnego $s$.

Oto kilka przykładowych konstrukcji ciągów w pełni równomiernie rozłożonych. (Kilka konstrukcji podano również w pracy Zielińskiego, (1989).)

**Przykład 1 (Franklin, 1963).** *Niech $A(\theta)$, $B(\theta)$, $\theta > 1$, będą dwiema funkcjami takimi, że $A$ ma ciągłą drugą pochodną i na każdym ograniczonym przedziale w $[1,\infty)$ ma co najwyżej skończoną liczbę zer. Wtedy, dla prawie wszystkich $\theta$ ciąg*

$$(\{A(\theta) \cdot \theta^n + B(\theta)\}, \quad n \geq 1)$$

*jest w pełni równomiernie rozłożony.*

*W szczególności ciągi* $(\{\theta^n\}, n \geq 1)$ *oraz* $(\{\alpha \cdot \theta^n\}, n \geq 1)$, *dla* $\alpha > 0$, *są w pełni równomiernie rozłożone dla prawie wszystkich* $\theta$. *(Symbol* $\{x\}$ *oznacza część ułamkową liczby* $x$.*)*

**Przykład 2 (Lewin, 1975).** *Dla dowolnej liczby przestępnej* $\theta > 1$ *można skonstruować liczbę* $\alpha$ *taką, żeby ciąg* $(\{\alpha \cdot \theta^n\}, n \geq 1)$ *był w pełni równomiernie rozłożony. Konstrukcja podana przez Lewina wygląda następująco. Dla danej liczby przestępnej* $\theta > 1$ *weźmy dowolną liczbę całkowitą* $p > \theta$ *i zdefiniujmy dwa ciągi:*

$$n_1 = 0,$$
$$n_r = \sum_{i=1}^{r-1} p^i, \quad r = 2, 3, \ldots$$
$$q_r = 4p^r p^{pr}, \quad r = 1, 2, \ldots$$

*Niech* $a_r$, $r = 1, 2, \ldots$, *będą liczbami całkowitymi takimi, że* $a_r \in [0, p_r)$. *Niech*

$$\alpha = \sum_{i=1}^{\infty} \frac{a_i}{q_i \theta^{n_i}}$$

*Wtedy ciąg* $(\{\alpha \cdot \theta^n\}, n \geq 1)$ *jest w pełni równomiernie rozłożony.*

**Przykład 3 (Lewin, 1981).** *Lewin podał konstrukcję takiej liczby* $\theta$, *że ciąg* $(\{\theta^n\}, n \geq 1)$ *jest w pełni równomiernie rozłożony, a mianowicie*

$$\theta = \lim_{r \to \infty} \theta_r$$

*gdzie:*

$$\theta_1 = 3,$$
$$\theta_{r+1} = \theta_r + \frac{a_r}{q_r p_r},$$
$$p_1 = 4,$$
$$p_{r+1} = q_r p_r,$$
$$q_r = \left(\theta_r^{r^{2r}} t_r^{-1} + 1\right) t_r (r+1)^4,$$
$$t_r = 24r \left(4(2r+1)^{2r}\right)^{2r+1}$$

*gdzie* $a_r \in [0, q_r)$ *są odpowiednio wybranymi liczbami całkowitymi.*

**Przykład 4 (Korobow, 1956).** *Niech* $q \geq 1$ *będzie liczbą całkowitą, niech* $p_1, p_2, \ldots$ *będzie ciągiem liczb pierwszych takim, że*

$$p_{n+1} < e^{\frac{1}{3} p_n^2}, \quad n = 1, 2, \ldots$$

*i niech $\psi(n)$, $n = 1, 2, \ldots$, będzie ciągiem liczb całkowitych takich, że*

$$n\left(\frac{p_{n+1}}{p_n}\right)^3 < \psi(n) < e^{p_n^2}$$

*Niech $(\nu_n,\ n \geq 1)$ będzie ciągiem określonym wzorami*

$$\nu_1 = 0, \qquad \nu_{n+1} = \nu_n + \psi(n) \cdot p_n^2 \cdot (p_n - 1)$$

*Jeżeli*

$$\alpha(x) = \sum_{n=1}^{\infty} \left[\frac{1}{q^{\nu_n}} - \frac{1}{q^{\nu_{n+1}}}\right] \frac{x^n}{p_n^2}$$

*to ciąg $(\{\alpha(n)q^n\}, n \geq 1)$ jest w pełni równomiernie rozłożony.*

**Przykład 5 (Korobow, 1960).** *Niech $p_1, p_2, \ldots$ będzie ciągiem liczb pierwszych takim, że*

$$(2k+1)^k < p_k < 2(2k+1)^k, \qquad dla \quad k = 1, 2, \ldots$$

*oraz $\psi(k)$ będzie dowolną funkcją o wartościach całkowitych taką, że $\psi(k) > k^2$. Określmy ciąg $\tau_0, \tau_1, \ldots$ w ten sposób, że*

$$\tau_0 = 0, \qquad \tau_k = \tau_{k-1} + k\psi(k)p_k, \qquad dla \quad k = 1, 2, \ldots$$

*Dowolną liczbę naturalną $n$ można przedstawić w sposób jednoznaczny równością*

(2) $n = \tau_{k-1} + ky + z$, *gdzie* $k \geq 1$, $0 \leq y \leq \psi(k)p_k - 1$, $1 \leq z \leq k$

*Wówczas ciąg*

$$x_n = \left\{\frac{a_{k,z}}{p_k} y\right\}$$

*gdzie $a_{k,z} = (2k+1)^{z-1}$ oraz liczby $k, z, y, n$ są określone równością (2), jest w pełni równomiernie rozłożony.*

**Przykład 6 (Starczenko, 1959).** *Niech*

$$n_k = \left[e^{(\ln k)^3}\right] + 1$$

*i niech $p_k$ będzie $k$-tą liczbą pierwszą. Ciąg*

$$\{\ln 2\}, \{2 \cdot \ln 2\}, \ldots, \{n_1 \cdot \ln 2\},$$
$$\{\ln 2\}, \{\ln 3\}, \ldots, \{n_2 \cdot \ln 2\}, \{n_2 \cdot \ln 3\},$$
$$\vdots$$
$$\{\ln 2\}, \{\ln 3\}, \ldots, \{\ln p_k\}, \ldots, \{n_k \cdot \ln 2\}, \{n_k \cdot \ln 3\}, \ldots, \{n_k \cdot \ln p_k\},$$
$$\vdots$$

*jest w pełni równomiernie rozłożony.*

**Przykład 7 (Rauzy, 1973).** *Rauzy udowodnił, że jeżeli funkcja całkowita nie będąca wielomianem, przyjmująca na osi rzeczywistej wartości rzeczywiste, spełnia warunek*

$$\limsup_{r\to\infty} \frac{\ln\ln M(r)}{\ln\ln r} < \frac{5}{4}$$

*gdzie* $M(r) = \sup\{|f(z)| : |z| = r\}$, *to* $(\{f(n)\}, n \geq 1)$ *jest ciągiem w pełni równomiernie rozłożonym.*

*W pracy Zielińskiego (1989) podano, zaproponowaną przez Dwilewicza, następującą konstrukcję funkcji f spełniającej warunek Rauzy'ego:*

$$f(z) = \sum_{n=1}^{\infty} a_n z^n$$

*gdzie* $(a_n, n \geq 1)$ *jest dowolnym ciągiem liczb dodatnich spełniających warunek*

$$a_n \leq 2^{-n} \exp\left\{-\frac{n}{5}\left(\frac{4n\cdot 2^n}{5}\right)^4\right\}$$

**Przykład 8 (Knuth, 1965).** *Podstawą konstrukcji Knutha są skończone ciągi* $A(b,k)$, $k = 1, 2, \ldots$, *gdzie* $b \geq 2$ *jest ustaloną liczbą naturalną, odpowiednio wiele razy powtórzone i następujące jeden za drugim. Przez* $A(b,k)^n$ *oznaczmy ciąg* $A(b,k)$ *powtórzony n-krotnie. Np.* $(0,1)^3(1)^2 = 0,1,0,1,0,1,1,1$. *CUD Knutha ma postać*

$$X_1, X_2, \ldots = A(b,1)^{2^2}, A(b^2,2)^{2\cdot 2^4}, A(b^3,3)^{3\cdot 2^6} \ldots$$

*Ciągi* $A(b,k) = (\frac{Y_1}{b}, \frac{Y_2}{b}, \ldots, \frac{Y_{bk}}{b})$ *konstruuje się w następujący sposób:*

$$Y_1 = Y_2 = \ldots = Y_k = 0$$

$$Y_{n+k} = \begin{cases} \text{najmniejsze } j \in [1,b) \text{ takie, że krotka } (Y_{n+1}, Y_{n+2}, \ldots, Y_{n+k-1}, j) \\ \quad \text{jeszcze nie wystąpiła w ciągu } (Y_1, Y_2, \ldots, Y_{n+k-1}) \\ 0, \text{ w przeciwnym przypadku} \end{cases}$$

*Jeżeli* $b = 2$ *to opisany wyżej ciąg Knutha wygląda następująco:*

$$\left.\begin{aligned} X_1 &= .0 \\ X_2 &= .1 \end{aligned}\right\} \text{przez 4-krotne powtórzenie otrzymujemy } X_1, X_2, \ldots, X_8$$

$$\left.\begin{array}{l} X_9 = .00 \\ X_{10} = .00 \\ X_{11} = .01 \\ X_{12} = .01 \\ X_{13} = .10 \\ X_{14} = .01 \\ X_{15} = .11 \\ X_{16} = .01 \\ X_{17} = .00 \\ X_{18} = .10 \\ X_{19} = .10 \\ X_{20} = .11 \\ X_{21} = .10 \\ X_{22} = .00 \\ X_{23} = .11 \\ X_{24} = .11 \end{array}\right\} \textit{przez 32-krotne powtórzenie otrzymujemy } X_9, X_{10}, \ldots, X_{520}$$

$$\left.\begin{array}{l} X_{521} = .000 \\ X_{522} = .000 \\ X_{523} = .000 \\ X_{524} = .001 \\ \quad\vdots \\ X_{1032} = .111 \end{array}\right\} 3 \cdot 2^8 = 192\textit{-krotnie powtarzając otrzymujemy } X_{521}, X_{522}, \ldots, X_{988224}$$

*... itd.*

**Przykład 9 (Knuth, 1969).** *W tej pracy Knuth nie podaje wprost żadnej konstrukcji konkretnego ciągu CUD, szkicuje tylko teoretyczną metodę konstrukcji takiego ciągu.*

DEFINICJA *„Obliczalnej reguły podciągu". Niech S będzie zbiorem o nieskończonej liczbie elementów. „Regułą podciągu" $\Re$ nazywamy nieskończony ciąg funkcji $(f_n(x_1, x_2, \ldots, x_n), n \geq 0, x_i \in S, i = 1, 2, ..., n)$ o wartościach 0 lub 1. Dla dowolnego ciągu $X = (x_i, i > 0)$ elementów zbioru S reguła pod-*

*ciągu* $\Re$ *definiuje podciąg* $(X)\Re$ *następująco:* $x_n$ *dopisujemy na końcu ciągu elementów już wybranych wtedy i tylko wtedy gdy* $f_n(x_1, x_1, \ldots, x_n) = 1$. *„Reguła podciągu"* $\Re$ *jest „obliczalną regułą podciągu" jeżeli istnieje efektywny algorytm wyznaczania wartości* $f_n(x_1, x_2, \ldots, x_n)$ *przy danych* $n$, $x_1, x_2, \ldots, x_n$, *dla każdego* $n \geq 0$.

*Niech* $\Re_1, \Re_2, \ldots$, *oznacza nieskończony ciąg „obliczalnych reguł podciągów". Jeżeli teraz* $(X)\Re_1$ *jest nieskończonym ciągiem równomiernie rozłożonym to można zdefiniować* $(X)\Re_1\Re_2 = ((X)\Re_1)\Re_2$. *Jeżeli ciąg* $(X)\Re_1\Re_2 \ldots$ *jest nieskończony, to jest on CUD.*

*Następnie Knuth ilustruje powyższą metodę konstrukcji CUD-ów wykorzystując następującą konstrukcję ciągu Walda. Niech* $V = (V_n, n \geq 1)$ *będzie ciągiem złożonym z wyrazów przedstawionych w systemie dwójkowym:*

$$
\begin{aligned}
&V_0 = 0, V_1 = .1, V_2 = .01, V_3 = .11, V_4 = .001 \\
&V_n = .c_r c_{r-1} \ldots c_1 1 \quad \text{dla} \quad n = 2^r + c_1 2^{r-1} + \ldots + c_r
\end{aligned} \tag{3}
$$

*Niech* $I_{b_1 b_2 \ldots b_r}$ *oznacza zbiór wszystkich liczb w [0,1), których reprezentacja binarna zaczyna się od* $0.b_1 b_2 \ldots b_r$

$$
I_{b_1 b_2 \ldots b_r} = [0.b_1 b_2 \ldots b_r, \quad 0.b_1 b_2 \ldots b_r + 2^{-r})
$$

*Ciąg* $V_n$ *jest równomiernie rozłożony w [0,1), ponieważ*

$$
\left| \frac{v(n)}{n} - 2^{-r} \right| \leq \frac{1}{n}
$$

*gdzie* $v(n)$ *oznacza ilość wyrazów* $V_n$ *w* $I_{b_1 b_2 \ldots b_r}$.

*W końcu Knuth przedstawia zarys konstrukcji ciągu CUD* $V\ \Re_1, \Re_2, \ldots$, *gdzie* $V$ *jest ciągiem zdefiniowanym w (3), a* $\Re_1, \Re_2, \ldots$ *jest nieskończonym ciągiem „obliczalnych reguł podciągów".*

W powyższej konstrukcji Knuth nawiązuje do idei „kolektywów" von Mises'a (1919), wykorzystuje też wyniki Walda (1937) i Kołomogorowa (1963).

**Przykład 10 (Cigler, Helmberg, 1963).** *Autorzy stwierdzają bez dowodu, że ciąg*

$$
x_n = \left\{ \sum_{k=0}^{\infty} n^k e^{-e^k} \right\} \quad \text{dla} \quad n = 1, 2 \ldots
$$

*jest w pełni równomiernie rozłożony.*

Wymienione przykłady konstrukcji CUD-ów są w większości procedurami wytwarzania kolejnych liczb ciągu $(R_n, n \geq 1)$ z przedziału [0,1). Powstaje pytanie, czy i jak takie punkty może „produkować" komputer?

## 2. Implementacja komputerowa wybranych ciągów w pełni równomiernie rozłożonych.

**2.1. Ogólne problemy implementacji.** Niech $(x_n, n \geq 1)$ będzie danym ciągiem CUD. Przez *implementację komputerową (implementację)* tego ciągu rozumiemy skonstruowanie dla danych liczb $\varepsilon > 0$ oraz $N > 0$, algorytmu i programu komputerowego realizującego ciąg $(z_n, n \geq 1)$ taki, że

$$|z_i - x_i| \leq \varepsilon, \quad \text{dla} \quad i = 1, 2, \ldots, N \tag{1}$$

Implementacja zależy od *arytmetyki komputera*, którą będziemy oznaczali liczbą naturalną $m$ z interpretacją, że $m$ jest maksymalną liczbą cyfr dwójkowych liczby $\mu$, jaką można (dokładnie) zapisać w komputerze. W przypadku liczb $\mu \in [0, 1)$ oznacza to ograniczenie do liczb postaci $z_i = 0.b_1 b_2 \ldots$ $\ldots b_t a_1 a_2 \ldots a_m$, gdzie $b_l = 0, 0 \leq l \leq t, t \geq 0$ oraz $a_i, 1 \leq i \leq m$ są cyframi dwójkowymi. Warunki dla $b_1, b_2, \ldots, b_t$ są konsekwencją ograniczeń cechy liczby $z_i$.

*Zakresem implementacji* nazwiemy największą liczbę $N = N(m, \varepsilon)$, dla której można zrealizować (za pomocą tej implementacji) ciąg $(z_n, n \geq 1)$ spełniający (1).

*Efektywnością implementacji* nazwiemy liczbę $S = S(m, \varepsilon, n)$, gdzie $S$ jest sumą wszystkich operacji podstawowych (operacji dodawania, odejmowania, mnożenia, dzielenia, dzielenia modulo, porównania) potrzebnych do obliczenia jednej liczby $z_n$. Niech ponadto $P_n$ oznacza liczbę „miejsc pamięci" komputera, zangażowanych do obliczenia liczby $z_n$.

Głównymi problemami technicznymi związanymi z implementacją CUDów o których mówiono są trudności polegające na tym, że

a) w typowych sytuacjach liczby $z_i$ oblicza się jako części ułamkowe pewnych szybko rosnących wraz ze wskaźnikiem $i$ liczb $z_i$.
b) liczby $z_i$ otrzymuje się jako wyniki operacji na liczbach bardzo małych i bardzo dużych;
c) liczby $P_n$ rosną (bardzo) szybko wraz z $n$;
d) warunki początkowe algorytmów wymagają skomplikowanych obliczeń.

Powyższe trudności zilustrujemy na podstawie przedstawionych w rozdziale drugim ciągów CUD. W przykładach: Lewin (1975), Lewin (1981), Korobow (1956), Franklin (1963) używane są części ułamkowe pewnych niewymiernych liczb $q, q > 1$, podniesionych do potęgi $n$. Wprawdzie interesują nas tylko $\{q^n\}$, ale musimy obliczać liczby $q^n$, które rosną bardzo szybko, np. gdy $q = 1.5, n = 25$, to $q^n = 25\,251.168294$, ale już dla $q = 1.5, n = 75, q^n = 16\,100\,687\,809\,804.7266$

Z drugiej strony nie znamy efektywnego sposobu wyznaczenia $\{q^n\}, n = 1, 2, \ldots$, bez obliczania $q^n$.

Nawet dla $1 < q < 2$, dających się przedstawić w postaci $q = 1 + a, 0 < a < 1$ wyznaczenie $\{q^n\}, n = 1, 2, \ldots$, w sposób rekursywny z wykorzysta-

niem dwumianu Newtona,

$$q^{n-1} = (1+a)^{n-1} = \sum_{k=0}^{n-1} \binom{n-1}{k} a^k =$$

$$= \underbrace{\binom{n-1}{0}}_{A_1} + \underbrace{\binom{n-1}{1} a}_{A_2} + \underbrace{\binom{n-1}{2} a^2}_{A_3} + \ldots + \underbrace{\binom{n-1}{n-1} a^{n-1}}_{A_n}$$

$$q^n = \sum_{k=0}^{n} \binom{n}{k} a^k = \binom{n}{0} + \frac{n}{1} A_1 \cdot a + \ldots + \frac{n}{n} A_n \cdot a = \binom{n}{0} + \sum_{i=1}^{n} \frac{n}{i} A_i \cdot a$$

jest nieefektywne i w komputerze prowadzi szybko do utraty cyfr znaczących (liczby $\binom{n}{k}$ rosną bardzo szybko np. $\binom{20}{10} = 184\,756$, $\binom{40}{20} = 137\,846\,528\,820$, $\binom{50}{25} = 126\,410\,606\,437\,752$).

Problem (b) występuje w przykładzie 7 oraz w przykładzie 10. W konstrukcji Dwilewicza

$$f(z) = \left\{ \sum_{n=1}^{\infty} a_n z^n \right\},$$

gdzie

$$a_n \leq 2^{-n} \exp\left( -\frac{n}{5} \left( \frac{4n \cdot 2^n}{5} \right)^4 \right),$$

wykorzystywane są liczby bardzo małe. Liczby $a_n$ mogą być przykładowo równe

$$(2) \qquad 2^{-n} \exp\left( -\frac{n}{5} \left( \frac{4n \cdot 2^n}{5} \right)^4 \right) \geq 2^{-n} 4^{-\frac{n}{5}\left(\frac{4n \cdot 2^n}{5}\right)^4} =$$

$$= 2^{-n} 2^{2 \cdot \left(-\frac{n}{5}\left(\frac{4n \cdot 2^n}{5}\right)^4\right)} \geq 2^{-n-n^5 2^{4n}} = a_n$$

Wyznaczając $a_n$ ze wzoru (2) mamy: $a_1 = 2^{-17}$, $a_2 = 2^{-8194}$, $a_3 = 2^{-999331}$. Następne $a_n$, $n > 3$ szaleńczo maleją do zera. W arytmetyce współczesnych komputerów można dokładnie przedstawić tylko dwa pierwsze $a_n$.

W przykładzie Knutha (1965) występują łatwo reprezentowalne w komputerze liczby wymierne. Jednak w tym przypadku trudności związane są z małą *efektywnością* $S$ oraz z szybko rosnącą liczbą $P_n$. Podczas konstrukcji wyrazów z $A(b^n, n)$, $n \geq 1$, dla sprawdzenia czy krotka $(x_1 x_2 \ldots x_n)$ już wystąpiła w konstruowanym ciągu trzeba pamiętać $(b^n)^n$ liczb.

W przykładzie 9 właściwie nie ma konstrukcji konkretnego ciągu CUD, przedstawiona jest tylko teoretyczna konstrukcja pewnej klasy ciągów spełniających warunki ciągów CUD. Trudna do realizacji na komputerze jest implementacja nieskończonego ciągu „obliczalnych reguł podciągu" .

Powyższe problemy techniczne powodują, że dla większości ciągów CUD praktyczna wartość *zakresu* $N(m,\varepsilon)$ jest stosunkowo małą liczbą rzędu $10^3$-$10^4$. Natomiast ze względu na zastosowania interesują nas wartości $N(m,\varepsilon)$ co najmniej rzędu $10^{10}$–$10^{20}$. Wydaje się, że stosunkowo duże wartości $N(m,\varepsilon)$ można osiągnąć dla ciągów Starczenki (1959) i Korobowa (1960). Właśnie te dwa ciągi wybrano do komputerowej implementacji.

Eksperymenty komputerowe prowadzono na komputerach zgodnych z IBM PC XT\AT\386. Dla tych komputerów arytmetyka $m = 52$ cyfry dwójkowe nosi nazwę „normalnej" arytmetyki zmiennopozycyjnej oraz $m = 63$ cyfry dwójkowe „podwyższonej" arytmetyki zmiennopozycyjnej. Arytmetyki te oznaczmy odpowiednio przez $m_1$ i $m_2$. W przypadku komputerów IBM PC przez „miejsce pamięci" komputera rozumiemy jeden bajt.

**2.2. Implementacja komputerowa ciągu Starczenki (1959).** Przypominamy (rodz. 1, przykład 6), że ciąg

$$
\begin{aligned}
&\{\ln 2\},\{2\cdot\ln 2\},\ldots,\{n_1\cdot\ln 2\},\\
&\{\ln 2\},\{\ln 3\},\ldots,\{n_2\cdot\ln 2\},\{n_2\cdot\ln 3\},\\
&\vdots\\
&\{\ln 2\},\{\ln 3\},\ldots,\{\ln p_k\},\ldots,\{n_k\cdot\ln 2\},\{n_k\cdot\ln 3\},\ldots,\{n_k\cdot\ln p_k\},\\
&\vdots
\end{aligned}
\tag{3}
$$

gdzie

$$n_k = \left[e^{(\ln k)^3}\right] + 1$$

oraz $p_k$ jest k-tą liczbą pierwszą, jest w pełni równomiernie rozłożony.

Ze względu na arytmetykę w komputerze można dokładnie reprezentować tylko $K$ początkowych liczb $n_k$. W przypadku arytmetyki $m_1$ mamy $K = 25$. Wstępna analiza ciągu (3) wskazuje, że ciąg ten jest łatwy do realizacji w komputerze. Oczywiście mamy dane $K$ początkowe liczby pierwsze $p_k$ z ciągu liczb pierwszch $(p_k, 1 \le k \le K)$ (wyznaczone innym algorytmem). Dla małych $k$ łatwo można wyznaczyć liczby $n_k$ wprost ze wzoru. Jednak konstruując ciąg wprost z (3) bardzo szybko tracimy dokładność $\varepsilon$. Już dla $k = 9$, $\ln p_9 = \ln 19 < 5$, $n_k = 40448$ liczba $n_k \cdot \ln k = 126\,824.4700459022$ jest wielkością rzędu $1.2 \cdot 10^5$ więc komputerowa reprezentacja liczby $\{n_k \cdot \ln k\}$ może mieć co najwyżej 9 cyfr dokładnych po kropce dziesiętnej. Wraz ze wzrostem $k$ dokładność szybko maleje. Zakres $N(m_1, 10^{-9})$ jest mniejszy od

$$N(m_1, 10^{-9}) = \sum_{i=1}^{10} n_i \cdot i = 2\,445\,718,$$

gdzie $n_1 = 2$, $n_2 = 2$, $n_3 = 4$, $n_4 = 15$, $n_5 = 65$, $n_6 = 315$, $n_7 = 1\,585$, $n_8 = 8\,036$, $n_9 = 40\,448$, $n_{10} = 200\,401$.

Stosując tę metodę reprezentacji ciągu (3), zwiększenie wartości zakresu $N(m_1, \varepsilon)$ można osiągnąć kosztem utraty dokładności $\varepsilon$.

Od wad tych wolna jest metoda oparta na wyznaczaniu wartości $\{n_k \cdot \ln k\}$ za pomocą szeregów (Knopp, 1956).

Mamy

$$\ln\left(\frac{1+x}{1-x}\right) = 2\sum_{n=0}^{\infty}\frac{x^{(2n+1)}}{2n+1} = 2\left(x + \frac{x^3}{3} + \frac{x^5}{5} + \ldots\right), \quad dla \quad |x| < 1.$$

W szczególności $x = 1/3$

(4)

$$\ln\left(\frac{1+x}{1-x}\right) = \ln 2 = 2\left(\frac{1}{3} + \frac{1}{3\cdot 3^3} + \frac{1}{5\cdot 3^5} + \ldots\right) = 2\sum_{n=0}^{\infty}\frac{1}{(2n+1)3^{2n+1}}$$

dla $x = (2p+1)^{-1}$, gdzie $p$ jest dowolną liczbą naturalną

$$\ln\left(\frac{1+x}{1-x}\right) = \ln(p+1) - \ln p = 2\sum_{n=0}^{\infty}\frac{1}{(2n+1)(2p+1)^{2n+1}} \tag{5}$$

Szeregi (4), (5) są szybko zbieżne np. dla (4) mamy

$$r_n = 2\sum_{m=n+1}^{\infty}\frac{1}{(2m+1)3^{2m+1}} < 2\frac{1}{(2n+3)3^{2n+3}}(1 + 1/9 + 1/9^2 + \ldots)$$

a więc

$$r_n < 2\frac{1}{(2n+1)3^{2n+1}}(9/8)\cdot(1/3^2) = r_n < \frac{1}{4(2n+1)3^{2n+1}}$$

Dla (5) prawdziwe jest oszacowanie

$$r_n < \frac{1}{2p(p+1)(2n+1)(2p+1)^{2n+1}} \tag{6}$$

Korzystając z równości (5) można obliczyć logarytm naturalny dowolnej liczby pierwszej $p$ poprzez rozkład $p-1$ na czynniki pierwsze np.

$$\ln 11 = \ln 2 + \ln 5 + 2\sum_{n=0}^{\infty}\frac{1}{(2n+1)(2\cdot 10+1)^{2n+1}}$$

Oznaczmy przez

$$R_k = 2\sum_{n=0}^{\infty}\frac{1}{(2n+1)(2(k-1)+1)^{2n+1}} \tag{7}$$

Zauważmy, że korzystając ze wzorów (5) i (7) logarytmy kolejnych liczb pierwszych można przedstawić w postaci:

$$
\begin{aligned}
&\ln 2 = R_2\\
&\ln 3 = R_2 + R_3\\
&\ln 5 = 2R_2 + R_3\\
&\ln 7 = 2R_2 + R_3 + R_7\\
&\ln 11 = 3R_2 + R_5 + R_{11}\\
&\ln 13 = 3R_2 + R_3 + R_{13}\\
&\ln 19 = 3R_2 + 2R_3 + R_{19}\\
&\ln 23 = 4R_2 + R_5 + R_{11} + R_{23}\\
&\vdots
\end{aligned}
\tag{8}
$$

Ogólnie logarytm dowolnej liczby pierwszej $p_k$, gdzie $k \geq 1$, można przedstawić w postaci (8) następująco:

$$\ln p_k = \sum_{i:p_i \leq p_k} \alpha_i R_{p_i} \tag{9}$$

Możliwość reprezentowania ciągu (3) za pomocą szeregów wykorzystano do jego komputerowej realizacji.

Ogólny schemat programu reprezentującego ciąg Starczenki

```
BEGIN
    Wyznaczenie R_i oraz {ln p_i} dla i = 1, 2, ..., K
    i := 1
    j := 1
    l := 1
    WHILE ( i ≤ K)
        BEGIN
            n_i := [e^((ln k)^3)] + 1
            WHILE ( j ≤ n_l )
                BEGIN
                    WHILE (l ≤ p_i )
                        BEGIN
                            Wyznaczenie {j · ln p_l}
                            l := l + 1
                        END
                    j := j + 1
                END
            i := i + 1
        END
END
```

W komputerowej realizacji pojawił się problem dokładności wyznaczania

danej liczby $\ln p_k$, gdzie $k \geq 1$. Jest to w istocie problem dokładności obliczenia szeregów $R_{p_i}$, dla $p_i \leq p_k$, we wzorze (9). Jeżeli błąd oszacowania każdego z tych szeregów nie przekracza $\varepsilon_0 > 0$ to błąd oszacowania liczby $\ln p_k$ ma postać

$$\varepsilon_0 \sum_{i:p_i \leq p_k} \alpha_i$$

Jeżeli liczba $\ln p_k$ ma być obliczona z dokładnością $\varepsilon$, to wystarczy wyznaczyć

$$\varepsilon_0 \leq \frac{\varepsilon}{\sum_{i:p_i \leq p_k} \alpha_i} \tag{10}$$

Na podstawie (6) mamy następujące oszacowanie dla reszt każdego z szeregów $R_{p_i}$

$$r_n^{p_i} = 2 \sum_{l=n}^{\infty} \frac{1}{(2l+1)(2(p_i-1)+1)^{2l+1}} <$$

$$< \frac{1}{2p_i(p_i-1)(2n+1)(2(p_i-1)+1)^{2n+1}}$$

Z kolei $r_n^{p_1} > r_n^{p_i}$, dla $i > 1$. Wobec wzoru (10) wyznaczenie $\ln p_k$ z dokładnością $\varepsilon$ sprowadza się do sumowania po $k_{max}$ wyrazów szeregów $R_{p_i}$, dla $p_i \leq p_k$, gdzie $k_{max}$ jest wyznaczone z nierówności

$$\frac{1}{4(2k_{max}+1)3^{2k_{max}+1}} < \varepsilon_0.$$

W komputerowej realizacji w celu redukcji liczby operacji w programie każdą z sum $R_{p_i}$ obliczamy jednokrotnie.

Podczas wyznaczania sum szeregów wystąpił też pewien kłopot techniczny związany z arytmetyką komputera, a mianowicie gdy $\frac{1}{2} < \left\{2\sum_{s=0}^{n-1} \frac{1}{(2s+1)(2(k-1)+1)^{2s+1}}\right\} < 1$ oraz $|a_n| < \varepsilon = 10^{-m_1}$; to zamiast

$$P(\{R_k\}) = \left\{2 \sum_{s=0}^{n-1} \frac{1}{(2s+1)(2(k-1)+1)^{2s+1}}\right\} + a_n$$

otrzymamy

$$P(\{R_k\}) = \left\{2 \sum_{s=0}^{n-1} \frac{1}{(2s+1)(2(k-1)+1)^{2s+1}}\right\} + 0.$$

Co w konsekwencji prowadzi do dokładności mniejszej niż założone $\varepsilon$.

Starano się temu w pewnym stopniu zapobiec. Po wyznaczeniu kolejnego wyrazu $a_n$ sprawdzamy czy wyraz ten jest mniejszy od pewnej stałej $\varepsilon_1 = 10^{-s}$ (zbliżonej do $\varepsilon$). Jeżeli tak to dodajemy go do tej części sumy szeregu, która jest mniejsza od $\varepsilon_1$. Jeśli nie to rozdzielamy go na dwie części, część

większą od $\varepsilon_1$ (dodajemy ją do sumy części większych od $\varepsilon_1$) i część mniejszą (dodajemy ją do sumy części mniejszych od $\varepsilon_1$). Wynik jest sumą obu części.

Nie rozwiązuje to jednak całkowicie problemu bo nierówność $a_n < \varepsilon_1^2$ mogą spowodować utratę cyfr znaczących dla dalszych wyrazów szeregów (np. dla $n = 20$, $k = 13$, $a_n$ są rzędu $10^{-3m}$) W tym przypadku można zastosować powyższy pomysł na kilku poziomach dokładności ($10^{-s}$, $10^{-2s}$, $10^{-3s}$).

W opisanej metodzie $P_n$ jest dość duże, dla ustalonego $K$, równa się ono $P_n = 82 \cdot K + 50$ bajtów.

Maksymalna wartość $n_k$, jaką można dokładnie reprezentować w komputerze wpływa bezpośrednio na wartość $N(m, \varepsilon)$. W arytmetyce $m_1$ *zakres* $N(m_1, \varepsilon)$ jest równy:

$$N(m_1, \varepsilon) = \sum_{i=1}^{25} n_i \cdot i = 10\,475\,097\,912\,550\,448,$$

gdzie $n_1 = 2$, $n_2 = 2$, $n_3 = 4$, $n_4 = 15$,

$n_5 = 65$, $n_6 = 315$, $n_7 = 1\,585$, $n_8 = 8\,036$, $n_9 = 40\,448$,

$n_{10} = 200\,401$, $n_{11} = 972\,536$, $n_{12} = 4\,609\,846$, $n_{13} = 21\,310\,546$,

$n_{14} = 96\,017\,491$, $n_{15} = 421\,606\,658$, $n_{16} = 1\,804\,551\,150$,

$n_{17} = 7\,532\,409\,024$, $n_{18} = 30\,680\,253\,486$, $n_{19} = 122\,021\,630\,171$,

$n_{20} = 474\,219\,072\,258$, $n_{21} = 1\,802\,209\,089\,278$, $n_{22} = 6\,702\,484\,682\,127$,

$n_{23} = 24\,411\,109\,899\,200$, $n_{24} = 87\,130\,384\,764\,031$,

$n_{25} = 304\,987\,684\,882\,156$

Natomiast wartość $\varepsilon$ nie wpływa bezpośrednio na $N(m_1, \varepsilon)$ (oczywiście $\varepsilon \geq 10^{-m_1}$).

Dokładne wyznaczenie *efektywności* $S(m_1, \varepsilon, n)$ jest trudne. W najlepszym przypadku $S(m_1, \varepsilon, n) = 10 + 25 \cdot \mu$, gdzie $\mu$ jest liczbą składników sumy (9).

W najgorszym $S(m_1, \varepsilon, n) = 15 + 7K + 150 \cdot \mu$, gdzie $K$ liczbą rozpatrywanych liczb pierwszych, a $\mu$ jest liczbą składników sumy (9). „Statystycznie" jednak $S(m_1, \varepsilon, n)$ osiąga wartość zbliżoną do wartości minimalnej. W przedstawionym sposobie wyznaczania wyrazów ciągu (3) wartość $n$ nie wpływa bezpośrednio na $S(m_1, \varepsilon, n)$. Natomiast istotny wpływ na wartość $S(m_1, \varepsilon, n)$ ma $\varepsilon$; mniejsze $\varepsilon$ wymaga sumowania większej liczby wyrazów odpowiednich szeregów co zwiększa wartość $S(m_1, \varepsilon, n)$. Do obliczania wartości $S(m_1, \varepsilon, n)$ użyto programu komputerowego. Obliczenia prowadzono bez uwzględnienia operacji wstępnych potrzebnych do wyznaczenia początkowych wartości $R_{p_k}$. Wywołania funkcji bibliotecznych traktowano jako instrukcje podstawowe.

**2.3. Implementacja komputerowa ciągu Korobowa (1960).** Przypominamy (rozdz. 1, przykład 5), że konstrukcja przykładu Korobowa wygląda następująco. Niech $p_1, p_2, \ldots$ będzie ciągiem liczb pierwszych takim, że

$$(11) \qquad (2k+1)^k < p_k < 2(2k+1)^k, \quad \text{dla} \quad k = 1, 2, \ldots$$

i niech $\psi(k)$ będzie dowolną funkcją o wartościach całkowitych taką, że $\psi(k) > k^2$ oraz

$$\tau_0 = 0, \tau_k = \tau_{k-1} + k\psi(k)p_k, \quad \text{dla} \quad k = 1, 2, \ldots$$

Dowolną liczbę naturalną $n$ można przedstawić w sposób jednoznaczny równością

$$(12) \quad n = \tau_{k-1} + ky + z, \quad \text{gdzie} \quad k \geq 1, \quad 0 \leq y \leq \psi(k)p_k - 1, \quad 1 \leq z \leq k$$

Wówczas ciąg

$$(13) \qquad x_n = \left\{ \frac{a_{k,z}}{p_k} y \right\}$$

gdzie $a_{k,z} = (2k+1)^{z-1}$ oraz liczby $k, z, y, n$ są określone równością (12), jest ciągiem CUD.

Analizując powyższą konstrukcję pod kątem implementacji komputerowej łatwo zauważyć, że:

1. Nie ma potrzeby wyznaczania kolejnych liczb $\tau_0, \tau_1, \ldots$ ponieważ w istocie nie są one używane do wyznaczania wyrazów ciągu (13).
2. Konieczne jest wyznaczenie liczb pierwszych $p_k$, $k = 1, 2 \ldots$, spełniających warunek (11). Ze względu na arytmetykę w komputerze można przedstawić tylko $K$ takich liczb (w przypadku arytmetyki $m_1$ mamy $K = 11$). Jedenaście początkowych wartości $p_k$ wyznaczono oddzielnym algorytmem. Liczba $p_{11}$ rzędu $9.5 \cdot 10^{14}$ ($p_{11} = 952\ 809\ 757\ 913\ 929$) jest największą z liczb $p_k$, jaką można przedstawić w arytmetyce $m_1$. W przedstawionym niżej algorytmie wykorzystywane liczby $p_n$ są najmniejszymi liczbami pierwszymi spełniającymi warunek (11).
3. Wyrazy ciągu (13) mają tę własność, że dla danego $k$ powtarzają się $\psi(k)$ krotnie. Wynika to z faktu, że $y$ można przedstawić w postaci

$$y = m \cdot p_k + n, \quad 0 \leq y \leq \psi(k)p_k - 1, \quad 0 \leq m \leq \psi(k), \quad n < p_k$$

$$(14) \qquad \left\{ \frac{a_{k,z}}{p_k} y \right\} = \left\{ \frac{a_{k,z}}{p_k} m \cdot p_k + \frac{a_{k,z}}{p_k} n \right\} = \left\{ \frac{a_{k,z}}{p_k} n \right\}$$

Wobec wzoru (14) obliczenia dla $y > p_k$ sprowadzają się do $\psi(k)$-krotnego powtórzenia obliczeń dla $y \leq p_k$. Obliczeń tych można wcale nie wykonywać, jedynie odtwarzać uprzednio zapamiętane wartości dla $y \leq p_k$. Wiąże się to jednak z szybkim wzrostem wartości $P_n$.

Wyznaczanie wyrazów ciągu wprost ze wzoru $\{a_{k,z}y/p_k\}$ z uwzględnieniem równości (14) jest szybkie ale niestety mało dokładne np. dla $k = 10$, $a_{k,k}$ jest liczbą rzędu $10^{12}$, $p_k$ rzędu $1.6 \cdot 10^{13}$, $y \leq p_k$ może być rzędu $10^{13}$. Wówczas przy dokładnej reprezentacji $P_k, a_{k,k}, y$ iloraz $(a_{k,k}/p_k)y$ jest liczbą rzędu $10^{12}$. W arytmetyce $m_1$ daje to dokładność trzech cyfr dziesiętnych po przecinku. Nie jest to dokładność wystarczająca.

W celu uzyskania większej dokładności zastosowano następujący pomysł. Dla danych liczb całkowitych $z, k$ oraz liczby pierwszej $p_k$ oznaczmy

$$y_z = \min\{1 \leq y < p_k : a_{k,z} \cdot y > p_k\}$$
$$r = a_{k,z} \cdot y_z - p_k$$

przedstawiając $a_{k,z} \cdot y$ z równości (14) w postaci

$$a_{k,z} \cdot y = \xi(p_k + r) + b_z a_{k,z}, \quad b_z a_{k,z} < p_k, \quad 0 \leq \xi \leq p_k, \quad r < a_{k,z},$$

mamy

$$\left\{\frac{a_{k,z}y}{p_k}\right\} = \left\{\frac{\xi(p_k + r)}{p_k} + \frac{b_z a_{k,z}}{p_k}\right\} = \left\{\left\{\frac{\xi r}{p_k}\right\} + \frac{b_z a_{k,z}}{p_k}\right\} \tag{15}$$

Wartości $(\xi r)/p_k$ mogą być dużymi liczbami, lecz w istocie przy sekwencyjnym obliczaniu wyrazów ciągu (13), wielkość $\{(\xi r)/p_k\}$ można wyznaczyć poprzez sekwencyjne sumowanie $\{r/p_k\}$.

Istnieje również pewna zależność między $a_{k,z}$ i $p_k$, a mianowicie z nierówności (11) mamy

$$a_{k,z} = (2k+1)^{z-1} \quad \text{dla} \quad (2k+1)^k < p_k < 2(2k+1)^k, \quad 1 \leq z \leq k \tag{17}$$

Wykorzystując zależność (16), podczas wyznaczania wyrazów ciągu, sprawdzenie warunku $a_{k,z}y > p_k$ i operacje z nim związane można wykonać tylko wtedy gdy $y \geq (2k+1)^{k-z+1}$ (np. dla $k = z$ warunek $a_{k,k}y \geq p_k$ sprawdzamy wtedy gdy $y > (2k+1)$). Redukuje to znacznie liczbę operacji.

Zależności (16), (17) i (18) wykorzystano podczas komputerowej realizacji ciągu Korobowa.

Ogólny schemat programu komputerowego wygląda następująco:

Ogólny schemat programu reprezentującego ciąg Korobowa

```
BEGIN
   k := 1
   WHILE ( k ≤ K)
      BEGIN
         l := 1
         fi := ψ(k)
         WHILE ( fi > 0 )
            BEGIN
               z := 1
               WHILE (z ≤ k )
                  BEGIN
```

```
                    Wyznaczenie x_n := { (a_{k,z}/p_k) y }
                    z := z + 1
                END
                IF (l = p_k)
                    BEGIN
                        fi := fi - 1
                        l := 0
                    END
                l := l + 1
            END
        k := k + 1
    END
END
```

Wartość $P_n$, dla ustalonej liczby liczb pierwszych $K$, jest równa $P_n = 34K + 60$ bajtów.

Wartość $N(m_1, \varepsilon)$ w arytmetyce $m_1$ przy $K = 11$, $\psi(k) = k^2 + 1$ jest równa

$$N(m_1, \varepsilon) = \tau_{11} = \sum_{i=1}^{11} i\psi(i)p_i = \sum_{i=1}^{11} i(i^2+1)p_i = 1\,295\,759\,206\,717\,074\,350$$

gdzie $p_1 = 5$, $p_2 = 27$, $p_3 = 347$, $p_4 = 6\,563$,

$p_5 = 161\,053$, $p_6 = 4\,826\,813$, $p_7 = 170\,859\,379$, $p_8 = 6\,975\,757\,457$,

$p_9 = 322\,687\,697\,791$, $p_{10} = 16\,679\,880\,978\,223$,

$p_{11} = 952\,809\,757\,913\,929$

$N(m_1, \varepsilon)$ jest więc wielkością rzędu $1.3 \cdot 10^{18}$. Można ją zwiększyć przez zastosowanie arytmetyki $m_2$. Dzięki zastosowaniu zależności (14), (15) i (16) wartość $N(m_1, \varepsilon)$ nie zależy natomiast od $\varepsilon$.

Nie ma żadnego związku między $\varepsilon$ i $S(m_1, \varepsilon, n)$. Wartość $S(m_1, \varepsilon, n)$ nie zależy również od $n$.

*Efektywność* $S(m_1, \varepsilon, n)$ obliczona na podstawie programu komputerowego w najlepszym przypadku jest równa $S_{min}(m_1, \varepsilon, n) = 11$, oraz w najgorszym $S_{max}(m, \varepsilon, n) = 8K + 36$, gdzie $K$ jest liczbą rozpatrywanych liczb pierwszych. Wartości $S_{min}, S_{max}$ są sumą wszystkich operacji podstawowych odpowiednio najkrótszej i najdłuższej drogi w programie do uzyskania jednego wyrazu ciągu (funkcje biblioteczne traktowane są jako instrukcje podstawowe).

**3. Porównania i testy** Obecnie dość powszechnie w obliczeniach komputerowych używany jest multyplikatywny generator liczb pseudolosowych (Lehmer, 1951) postaci $x_{n+1} = c \cdot x_n (mod \quad m)$ $n = 0, 1, 2, \ldots$, gdzie liczby $x_0$, $m$, $c$ są liczbami całkowitymi takimi, że $x_0 < m$, $c < m$. Generator ten oznaczmy symbolem GM. W komputerze realizuje się go w arytmetyce liczb

całkowitych $\mu$. W porównaniu do implementacji CUDów przedstawionych w rozdz. 2.2 i rozdz.2.3 komputerowa realizacja generatora GM jest o wiele prostsza. Wartości opisujące implementacje generatora GM są równe:

- *Efektywność* $S(\mu, \varepsilon, n) = 2$;
- $P_n = 3 \cdot k$, gdzie $k$ jest liczbą bajtów potrzebnych do zapamiętania liczby całkowitej ($c$, $x_n$ lub $m$);
- maksymalny okres generatora GM jest równy $2^{m-2}$ (Zieliński, 1972).

Powyższe własności powodują, że generator GM może być łatwo użyty w programach symulujących procesy losowe. Wartości $S(m, \varepsilon, n)$, $P_n$ dla ciągów CUD są o wiele gorsze. Użycie algorytmu obliczającego wyrazy ciągu CUD jako procedury w programach symulujących procesy losowe jest o wiele bardziej skomplikowane niż w przypadku generatora GM ale daje pewną istotną korzyść. Mianowicie komputerowa realizacja $\Psi$ ciągu CUD jest właściwie „dyskretną" reprezentacją tego ciągu. Dla $\Psi$, podobnie jak dla każdego CUDu, prawdziwa jest własność: dla każdego naturalnego $s$ i dla każdego przedziału $I \subset [0,1]^s$, frakcja liczb $\Psi$, leżących w I, jest (w granicy) równa długości tego przedziału ($\Psi$ jest $\varepsilon$-siecią, zależną od arytmetyki, na przedziale $[0,1]^s$). Żaden okresowy ciąg liczb taką $\varepsilon$-siecią na przedziale $[0,1]^s$, dla dowolnego naturalnego $s$, być nie może.

Porównanie wartości $P_n$, $N(m, \varepsilon)$, $S(m, \varepsilon, n)$ charakteryzujących implementacje ciągu Korbowa (przykład 5) do odpowiednich wartości implementacji ciągu Starczenki (przykład 6) wydaje się preferować ciąg Korobowa. Jednak w przykładzie 5 bardziej uciążliwe jest wytworzenie początkowego ciągu liczb pierwszych spełniającego odpowiedni warunek. Liczby te rosną bardzo szybko. Już dwunasty element tego ciągu nie może być dokładnie reprezentowany w arytmetyce $m_1$. Początkowy ciąg liczb występujący w przykładzie 6 jest ciągiem kolejnych początkowych liczb pierwszych.

Liczby uzyskane podczas implementacji przykładu 6 wydają się też być „bardziej losowe" od liczb uzyskanych podczas implementacji przykładu 5. Ilustrują to zastosowane testy.

Na implementacjach obu ciągów CUD przeprowadzono testy graficzne, statystyczne oraz test numeryczny i liczbowy.

**3.1. Testy graficzne.** Celem testów graficznych jest przedstawienie kolejnych wyrazów generowanych ciągów jako punktów na ekranie komputera. Ekran komputera może być traktowany jako dwuwymiarowa tablica $Ekran[m, n]$ (wartości $m$,$n$ zależą od zainstalowanej w nim karty graficznej np. dla karty „Herkules" $m = 720$, $n = 348$). Dodatkowym trzecim wymiarem może być kolor punktu (w zależności od karty graficznej od 2 do 256).

W arytmetyce $m_1$ programy komputerowe „produkują" liczby $x_n \in [0,1)$

z 15 cyframi dziesiętnymi po przecinku postaci

$$x_n = 0.a_1a_2a_3a_4a_5a_6a_7a_8a_9a_{10}a_{11}a_{12}a_{13}a_{14}a_{15}.$$

Na ekranie komputera w jednym wymiarze można przedstawić tylko trzy cyfry każdej z liczb $x_n$. Przykładowo w komputerach z graficzną kartą „Herkules" na poziomej osi można przedstawić $X_{max} = 720$ punktów. Wykonując działanie $X_{max} \cdot x_n$ otrzymamy przeskalowaną reprezentację pierwszych trzech cyfr $a_1a_2a_3$ z liczby $x_n$. Podobnie jest w przypadku osi pionowej.

Toteż w jednym z proponowanych testów skalowanie odbywa się w następujący sposób: liczba $x_n$ mnożona jest przez dużą stałą $C$, a następnie część ułamkowa wyniku mnożona jest przez $X_{max}$

$$Obraz_{x_n} = \{x_n \cdot C\} \cdot X_{max}$$

W zależności od stałej $C$ uzyskuje się obraz trzech różnych cyfr z $x_n$. Dla $C = 10^{10}$, $Obraz_{x_n}$ jest przeskalowaną reprezentacją cyfr $a_{11}a_{12}a_{13}$.

W teście TEST1 każdy z punktów na ekranie komputera tworzony jest przez trzykrotne wywołanie procedury generującej liczbę z ciągu CUD następująco:

$$X = \{x_n \cdot C\} \cdot X_{max}$$
$$Y = \{x_{n+1} \cdot C\} \cdot Y_{max}$$
$$Kolor = \{x_{n+2} \cdot C\} \cdot Kolor_{max},$$

$X, Y, Kolor$ są współrzędnymi na ekranie, a $X_{max}, Y_{max}, Kolor_{max}$ są maksymalnymi wartościami współrzędnych poziomej, pionowej i koloru zależnymi od karty graficznej.

W teście TEST2 punkty na ekranie komputera tworzone są przez jednokrotne wywołanie procedury generującej liczby z ciągu CUD w następujący sposób:

$$X = X_{max} \cdot x_n$$
$$Y = \{X_{max} \cdot x_n\} \cdot Y_{max}$$
$$Kolor = \{X_{max} \cdot x_n \cdot Y_{max}\} \cdot Kolor_{max}.$$

W obu testach punkty pojawiają się na ekranie komputera w miarę generacji liczb z ciągów CUD.

Graficzne ilustracje testów dla 50000 pierwszych punktów na ekranie przedstawiona jest na rysunkach: 1, 1A, 2, 2A, 3, 3A. Ilustracje te są zmniejszoną kopią ekranu na drukarce uzyskaną przy dostępnych dwu kolorach z zastosowaniem inwersji obrazu.

Ilustracje wszystkich reprezentowanych testów zawierają linie opisu ekranu. W liniach tych oprócz informacji związanych z danym testem znajdują się informacje o tym jak w danej chwili przerwać program realizujący test (Esc-wyj – naciśnięcie klawisza „Esc") oraz w jaki sposób wyświetlić aktualne wartości parametrów liczbowych testu (I-ak. – naciśnięcie klawisza „I").

Informacje te są istotne, ponieważ programy realizujące testy pracują ciągle. Przedstawione na rysunkach ekrany są tylko stanem chwilowym programów testujących.

Ilustracje oraz obserwacja zmian na ekranie komputera podczas testów sugerują, że liczby uzyskane z implementacji przykładu 5 są „bardziej losowe" od liczb uzyskanych z implementacji przykładu 6.

Przedstawione rysunki nie obrazują zmian koloru ekranu podczas testów na komputerach z kolorową kartą graficzną. Dla testów przeprowadzonych na ciągu Starczenki kolorowe punkty w miarę pojawiania się na ekranie zlewają się tworząc „białą plamę". W przypadku ciągu Korobowa proste powstające w teście TEST1 są białe. Natomiast w teście TEST2 kolory wydają sie być równomiernie rozłożone, czarny pas z rysunku 3A na ekranie jest pasem białym.

Zastanawiające dla ciągu Starczenki (TEST1, stała $C = 10^0$) są widoczne na ekranie prawie poziome linie przy $i \geq 8$. Czy nie są one wynikiem błędu w programie? W celu odpowiedzi na to pytanie wykorzystano w programie TEST1 procedurę obliczania wartości $\{j \cdot \ln p_i\}$ dla podejrzanych $i, 8 \leq i \leq 11$ wprost ze wzoru $\{j \cdot \ln p_i\}$. W przypadku $i = 8$, stałej $C = 1$, $j \leq n_k, j \leq 8036, \ln 19 < 5$ komputerowa reprezentacja $\{j \cdot \ln p_k\}$ jest następująca:

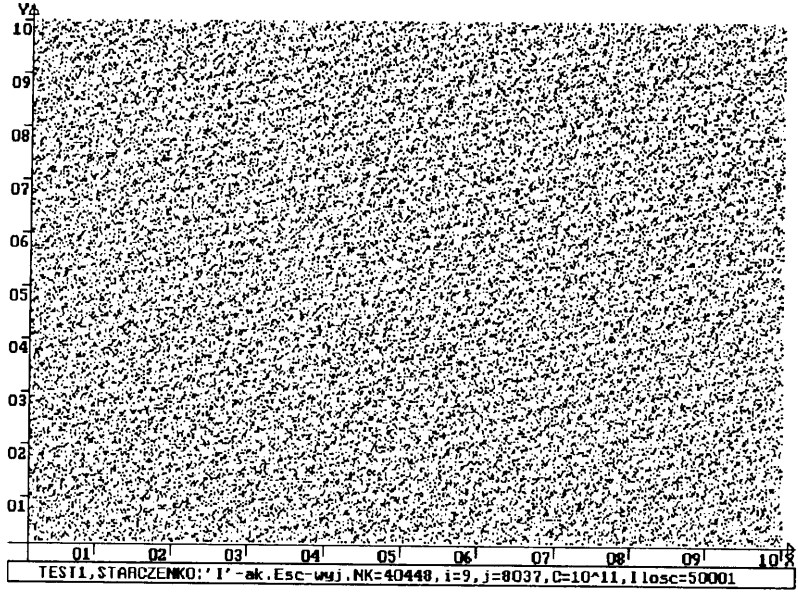
$$j \cdot \ln p_k = a_1 a_2 a_3 a_4 a_5 . a_6 a_7 a_8 a_9 a_{10} a_{11} a_{12} a_{13} a_{14} a_{15}$$

oraz

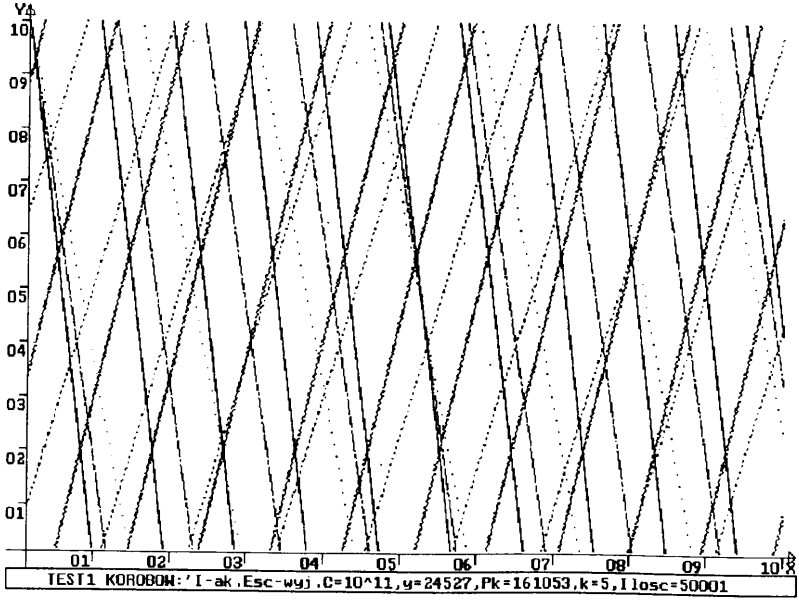
$$\{j \cdot \ln p_k\} = 0 . a_6 a_7 a_8 a_9 a_{10} a_{11} a_{12} a_{13} a_{14} a_{15} b_1 b_2 .$$

Wobec tego cyfry $a_6 a_7 a_8$ przedstawiane graficznie są cyframi dokładnymi. Dokładna jest również graficzna reprezentacja liczb $\{j \cdot \ln p_i\}$ dla $i \leq 11$. Wyniki uzyskane po zastosowaniu obu metod (obliczanie wartości $\{j \cdot \ln p_i\}$ wprost ze wzoru $\{j \cdot \ln p_i\}$ oraz metody proponowanej w rozdz. 2.2) okazały się identyczne. Sugeruje to poprawność wyników otrzymanych po zastosowaniu metody przedstawionej w rozdz. 2.2.

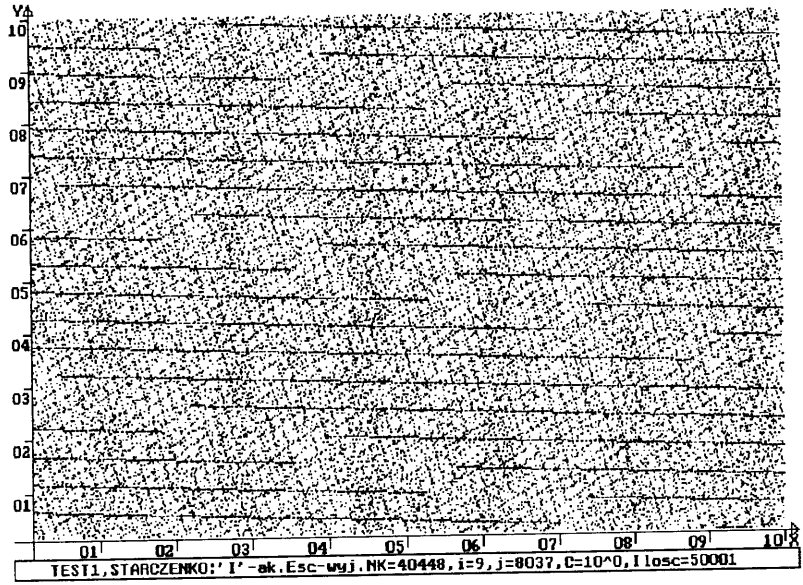
Celem testu TEST3 jest graficzne porównanie przykładu 6 z generatorem GM. Generatora GM użyto ze stałą $c = 4 \cdot 23^7 + 1$, $m = 2^{40}$ oraz $x_0 = 7$. Taki wybór parametrów zapewnia okres równy $2^{38}$ i „dobre" właściwości statystyczne generatora GM (Zieliński, 1989). W teście tym punkty na ekranie komputera są reprezentacją par liczb $(r_n, r_{n+1})$, gdzie $n = 1, 2, \ldots$. Teoretycznie wiadomo (Marsaglia, 1968), że w przypadku generatora GM (kładąc $r_n = x_n/m$) otrzymamy na ekranie pewną liczbę prostych. Jednak gdy na ekranie przedstawiamy wszystkie pary liczb $(r_n, r_{n+1})$ wówczas prostych tych jest tak dużo, że wypełniają one praktycznie cały ekran. W rozważanym przypadku może być ich nawet $4 \cdot 23^7 + 2$ (Zieliński, 1972). Dlatego też na ekranie komputera przedstawiono tylko te pary liczb, które należą do przedziału $[k, j]^2$, gdzie $k = 0.5$, $j = 0.505$. Graficzna ilustracja testu przedstawiona jest na rysunkach 4 i 4A.



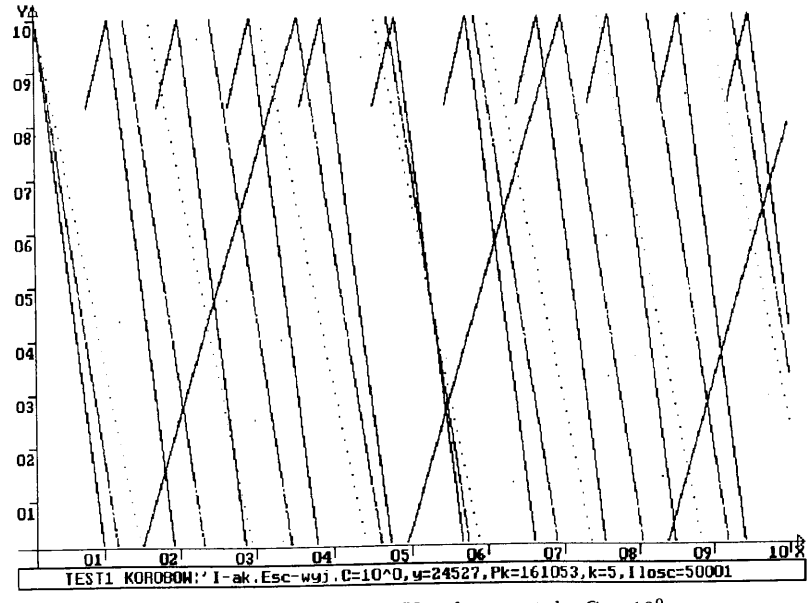

Rys. 1. TEST 1, ciąg Starczenki, stała $C = 10^{11}$.



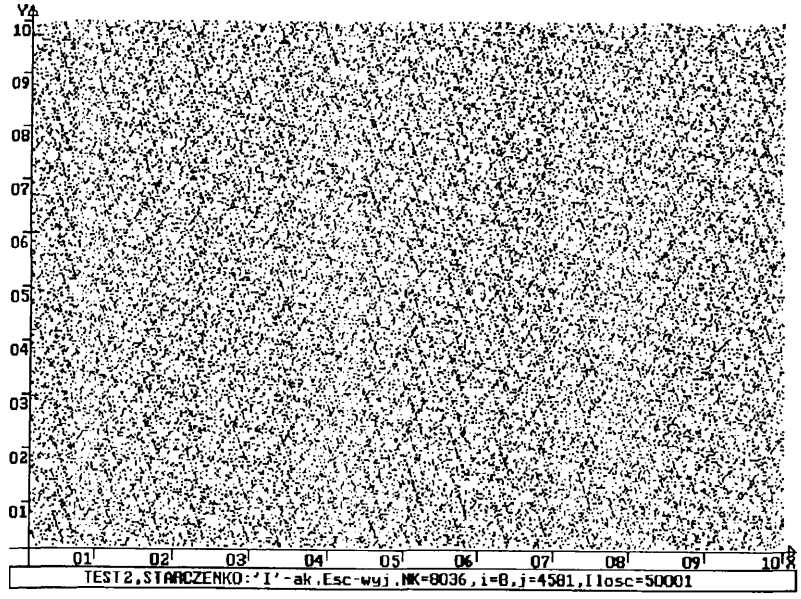

Rys. 1A. TEST 1, ciąg Korobowa, stała $C = 10^{11}$.



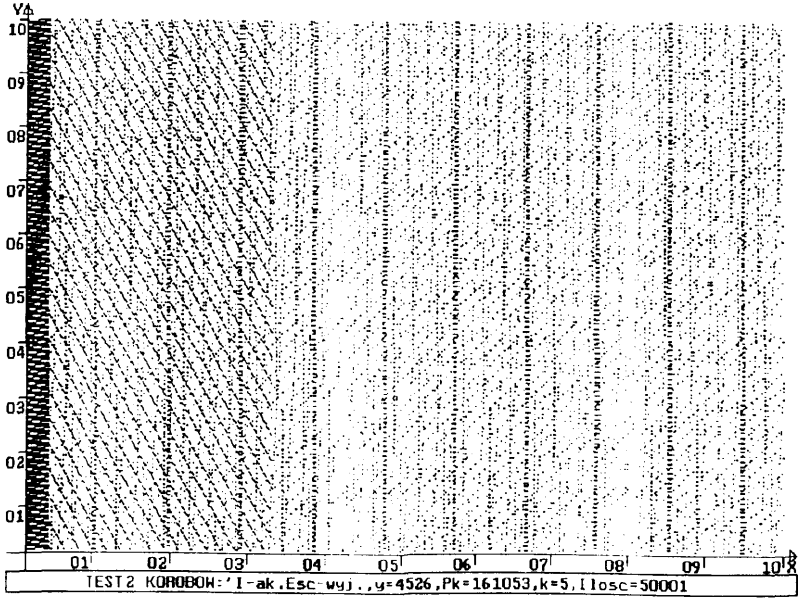

Rys. 2. TEST 1, ciąg Starczenki, stała $C = 10^0$.



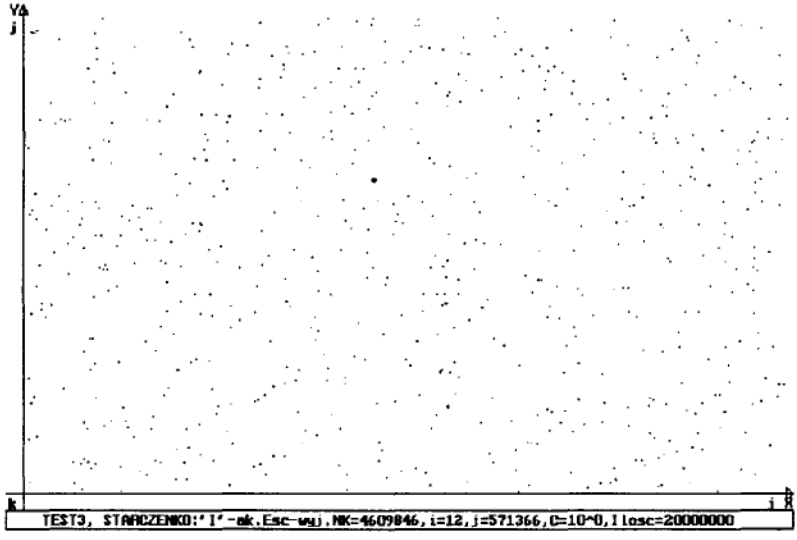

Rys. 2A. TEST 1, ciąg Korobowa, stała $C = 10^0$.



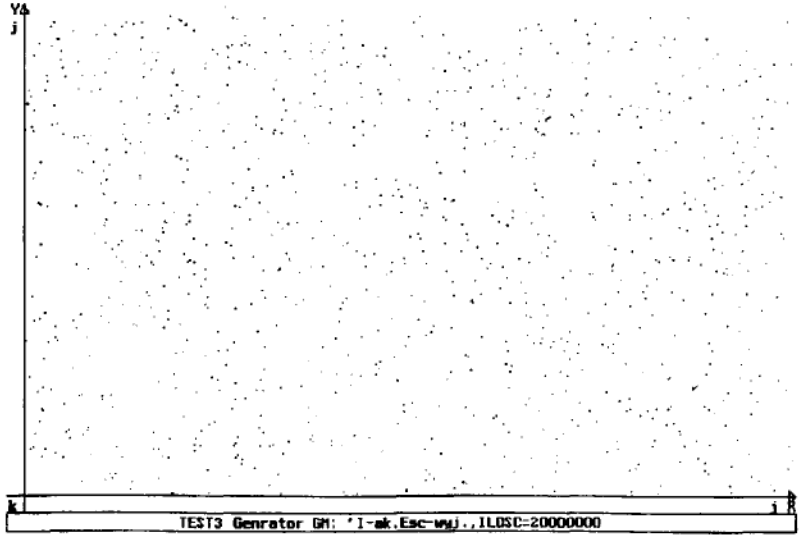

Rys. 3. TEST 2, ciąg Starczenki.




Rys. 3A. TEST 2, ciąg Korobowa.




Rys. 4. TEST 3, ciąg Starczenki.




Rys. 4A. TEST 3, generator multyplikatywny.

**3.2. Testy statystyczne** Jeżeli ciąg CUD $(R_n, n \geq 1)$ symuluje realizację ciągu $(U_i, i \geq 1)$ niezależnych zmiennych losowych o jednakowym rozkładzie równomiernym na przedziale [0,1] to mamy

$$\lim_{N\to\infty} \frac{\#\{1 \leq n \leq N : (R_n, R_{n+1}, \ldots, R_{n+s-1}) \in A\}}{N} = \\ = P\{(U_1, U_2, \ldots, U_s) \in A\}$$

dla dowolnego naturalnego $s$, $A = [0, t]$, gdzie $t = (t_1, t_2, \ldots, t_s)$, oraz $t_i \leq 1$ dla $i = 1, 2, \ldots, s$. Stąd wynika, że ciąg $R_n$ asymptotycznie spełnia niektóre testy statystyczne. Testy te nie muszą być spełnione dla początkowego odcinka ciągu $R_n$. Implementacja komputerowa $R_n$ właśnie przedstawia taki odcinek ciągu CUD. W celu zbadania właściwości statystycznych opisanych ciągów posłużono się dwoma testami:

1. testem permutacji;
2. testem rozkładu wykładniczego.

Opis pierwszego testu można znaleźć w pracy (Zieliński, 1972, rozdz. 5.7.3).

Wyniki testu ilustruje tabela 1.

Tabela 1. Wyniki „testu permutacji".

| | | n=1000 | n=10000 | n=100000 | n=500000 |
|---|---|---|---|---|---|
| Permutacja | Generator GM | 0.7123 | 0.1762 | 0.3987 | 0.5160 |
| 5-elementowa | Ciąg Starczenki | 0.2177 | 0.8413 | 0.98575 | > 0.999999 |
| (k=119) | Ciąg Korobowa | < 0.000001 | < 0.000001 | < 0.000001 | < 0.000001 |
| Permutacja | Generator GM | 0.7704 | 0.6443 | 0.8133 | 0.1922 |
| 6-elementowa | Ciąg Starczenki | 0.0995 | < 0.00001 | 0.0834 | >0.999999 |
| (k=719) | Ciąg Korobowa | < 0.000001 | < 0.000001 | < 0.000001 | < 0.000001 |

W poszczególnych kolumnach podane są wartości $P(\chi_k^2 > \alpha)$, gdzie $\alpha$ jest otrzymaną wartością testu zgodności *chi-kwadrat* dla $k = 119$ stopni swobody w przypadku permutacji 5-elementowej oraz $k = 719$ stopni swobody w przypadku permutacji 6-elementowej.

W drugim teście wygenerowano 200000 wartości zmiennej losowej o rozkładzie wykładniczym używając metody Forsythe-Von Neumann (Devroye, 1986). Wygenerowane liczby podzielono na 200 grup, zawierających po 1000 liczb, według kolejności generowania. Do każdej grupy zastosowano test zgodności *chi-kwadrat* w celu obliczenia poziomu krytycznego (prawdopodobieństwo przekroczenia otrzymanej wartości). Wartości poziomu krytycznego dla poszczególnych grup powinny być rozłożone równomiernie w przedziale [0,1). Wyniki przedstawione są w tabeli 2. Wartość $p.k. = j$ oznacza, że poziom krytyczny należy do przedziału $[0.1 \cdot j, 0.1 \cdot (j + 1))$.

Tabela 2. Rozkład poziomów krytycznych dla k=100 stopni swobody.

| p.k.= | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 |
|---|---|---|---|---|---|---|---|---|---|---|
| Generator GM | 23 | 22 | 22 | 13 | 15 | 18 | 19 | 14 | 10 | 44 |
| Ciąg Starczenki | 36 | 41 | 11 | 16 | 11 | 15 | 15 | 9 | 24 | 22 |
| Ciąg Korobowa | 200 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**3.3 Test numeryczny** Test wyznacza $\frac{1}{2^k}$ części objętości kuli jednostkowej w przestrzeni $k$-wymiarowej.

$$V_k = \frac{2}{k} \cdot \frac{\pi^{\frac{k}{2}}}{\Gamma(\frac{k}{2})}$$

Zastosowano w nim metodę „orzeł-reszka" obliczania całek (Zieliński, 1970). Otrzymane wyniki zilustrowano w tabeli 3. W poszczególnych kolumnach podane są obliczone wartości dla 1000, 10000, 100000, 500000 punktów w przestrzeniach 3, 10 i 12 wymiarowych.

Tabela 3. Wyznaczone wartości $\frac{1}{2^k}$ części objętości kuli jednostkowej w przestrzeni $k$-wymiarowej.

| | | n=1000 | n=10000 | n=100000 | n=500000 |
|---|---|---|---|---|---|
| Kula 3-wymiarowa | Wart. teoretyczna | 0.5235 | | | |
| | Generator GM | 0.5480 | 0.5321 | 0.5259 | 0.5234 |
| | Ciąg Starczenki | 0.5240 | 0.5278 | 0.5236 | 0.5237 |
| | Ciąg Korobowa | 0.5670 | 0.5747 | 0.5276 | 0.5102 |
| Kula 10-wymiarowa | Wart. teoretyczna | 0.00249 | | | |
| | Generator GM | 0.00900 | 0.00330 | 0.00277 | 0.00254 |
| | Ciąg Starczenki | 0.00200 | 0.00210 | 0.00228 | 0.00237 |
| | Ciąg Korobowa | 0.05900 | 0.04510 | 0.04083 | 0.03053 |
| Kula 12-wymiarowa | Wart. teoretyczna | 0.000325 | | | |
| | Generator GM | 0.001000 | 0.000200 | 0.000310 | 0.000310 |
| | Ciąg Starczenki | 0.000000 | 0.000100 | 0.000380 | 0.000328 |
| | Ciąg Korobowa | 0.022000 | 0.027000 | 0.022120 | 0.018732 |

Najbliższe wartości dokładnych są wyniki uzyskane dla ciągu Starczenki. Bardzo odległe od wartości dokładnych są wyniki uzyskane dla ciągu Korobowa. W tym przypadku widoczna jest jednak poprawa wyniku wraz ze wzrostem liczby generowanych punktów.

**3.4. Test liczbowy** Wyniki tego testu przedstawione są w zestawieniach tabelarycznych Rys. 5 i Rys. 5A będących kopią ekranu komputera. Element $(i,j)$) przedstawia liczbę par $(x_n, x_{n+1}), n = 1, 2, \ldots, 199\,999$ ta-

kich, że $x_n \in [0.i, 0.i + 0.1)$ oraz $x_{n+1} \in [0.j, 0.j + 0.1)$. Ponadto w prawym końcu każdego wiersza oraz u góry każdej kolumny podana jest ilość $x_n \in [0.i, 0.i+0.1)$. W celach kontrolnych w prawym górnym rogu tabeli podane są dwie liczby będące sumą wszystkich liczb w wierszach (dolna liczba) oraz w kolumnach (górna liczba). Jeśli pary liczb $(x_n, x_{n+1})$ są równomiernie rozłożone w przedziale $[0, 1)^2$ to każdy element tabeli powinien być równy 2000.

Rysunki 5 i 5A ilustrują wyniki testu liczbowego dla pierwszych 200 000 wyrazów ciągów CUD. Program realizujący test pracuje ciągle. Wciśnięcie klawisza „I" powoduje wyświetlenie na ekranie aktualnych wartości. Dzięki temu można obserwować w sposób płynny zmiany zawartości tabel.

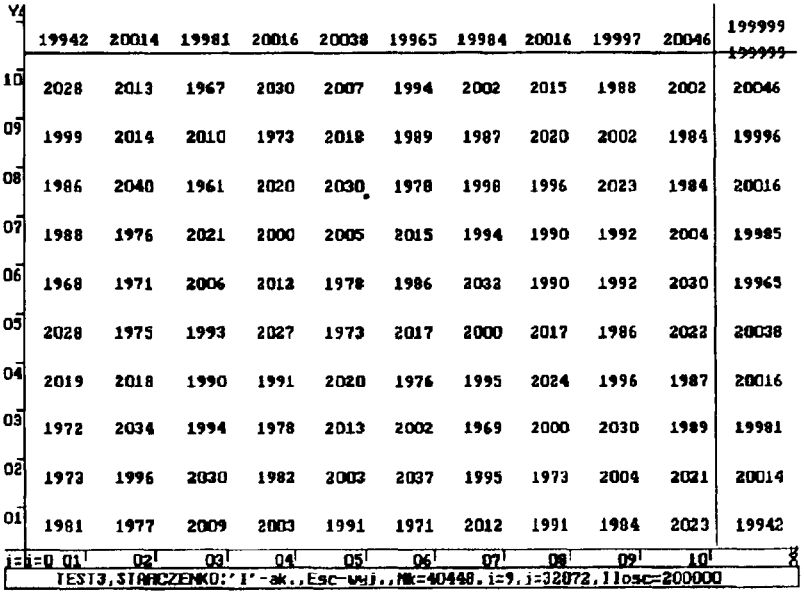
Można zauważyć, że:

1. Ciąg Korobowa
   - dla $a_{k,z} \cdot y$ znacznie mniejszych od $p_k$ suma liczb znajdujących się w pierwszych trzech wierszach tabeli jest znacznie większa (7-8 razy) od sumy liczb w wierszach pozostałych. Podobny rozkład liczb można zaobserwować w kolumnach tabeli.
   - dla $a_{k,z} \cdot y$ zbliżonych do $p_k$ różnica ta stopniowo się wyrównuje.
2. Ciąg Starczenki
   - rozkład liczb jest bardziej równomierny, różnica pomiędzy poszczególnymi elementami tabeli nie jest większa od 100.

Wyniki podobnego testu dla pewnego nieokresowego generatora liczb pseudolosowych przedstawione są w pracy Zielińskiego (1990).

**5. Wnioski.** Wyniki przedstawionych testów sugerują następujące wnioski:

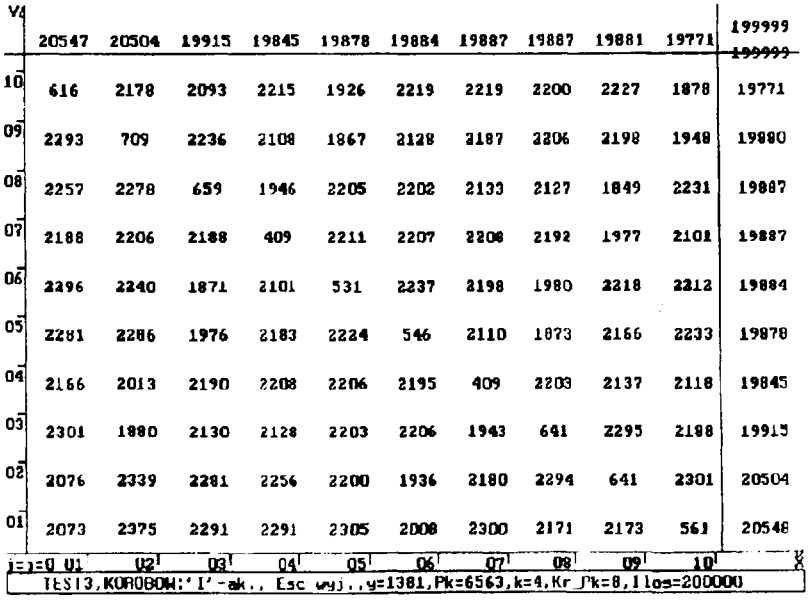
1. Liczby uzyskane z implementacji CUDu Starczenki traktowane jako realizacja zmiennej losowej o rozkładzie równomiernym na przedziale $[0, 1)^2$ są znacznie bardziej wiarygodne od liczb uzyskanych z implementacji CUDu Korobowa. Wskazują na to testy graficzne, statystyczne oraz test liczbowy.
2. Test numeryczny wykazuje również przewagę implementacji ciągu Starczenki nad implementacją ciągu Korobowa w przestrzeniach 3, 10 i 12 wymiarowych.
3. Testy statystczne wskazują, że CUD Starczenki jest znacznie bliższy akceptacji od CUDu Korobowa.
4. Porównanie generatora GM z komputerową realizacją ciągu Starczenki sugeruje podobną jakość tych generatorów (podobne wyniki testu graficznego, nieco gorsze wyniki GM w teście numerycznym, lepsze wyniki GM w testach statystycznych). Jednak nie należy zapominać, że w przedstawionych testach była brana pod uwagę mała liczba liczb; zwykle do



| | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| | 19942 | 20014 | 19981 | 20016 | 20038 | 19965 | 19984 | 20016 | 19997 | 20046 | 199999 |
| 10 | 2028 | 2013 | 1967 | 2030 | 2007 | 1994 | 2002 | 2015 | 1988 | 2002 | 20046 |
| 09 | 1999 | 2014 | 2010 | 1973 | 2018 | 1989 | 1987 | 2020 | 2002 | 1984 | 19996 |
| 08 | 1986 | 2040 | 1961 | 2020 | 2030 | 1978 | 1998 | 1996 | 2023 | 1984 | 20016 |
| 07 | 1988 | 1976 | 2021 | 2000 | 2005 | 2015 | 1994 | 1990 | 1992 | 2004 | 19985 |
| 06 | 1968 | 1971 | 2006 | 2012 | 1978 | 1986 | 2032 | 1990 | 1992 | 2030 | 19965 |
| 05 | 2028 | 1975 | 1993 | 2027 | 1973 | 2017 | 2000 | 2017 | 1986 | 2022 | 20038 |
| 04 | 2019 | 2018 | 1990 | 1991 | 2020 | 1976 | 1995 | 2024 | 1996 | 1987 | 20016 |
| 03 | 1972 | 2034 | 1994 | 1978 | 2013 | 2002 | 1969 | 2000 | 2030 | 1989 | 19981 |
| 02 | 1973 | 1996 | 2030 | 1982 | 2003 | 2037 | 1995 | 1973 | 2004 | 2021 | 20014 |
| 01 | 1981 | 1977 | 2009 | 2003 | 1991 | 1971 | 2012 | 1991 | 1984 | 2023 | 19942 |



Rys. 5. TEST 4, ciąg Starczenki.



| | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| | 20547 | 20504 | 19915 | 19845 | 19878 | 19884 | 19887 | 19887 | 19881 | 19771 | 199999 |
| 10 | 616 | 2178 | 2093 | 2215 | 1926 | 2219 | 2219 | 2200 | 2227 | 1878 | 19771 |
| 09 | 2293 | 709 | 2236 | 2108 | 1867 | 2128 | 2187 | 2206 | 2198 | 1948 | 19880 |
| 08 | 2257 | 2278 | 659 | 1946 | 2205 | 2202 | 2133 | 2127 | 1849 | 2231 | 19887 |
| 07 | 2188 | 2206 | 2188 | 409 | 2211 | 2207 | 2208 | 2192 | 1977 | 2101 | 19887 |
| 06 | 2296 | 2240 | 1871 | 2101 | 531 | 2237 | 2198 | 1980 | 2218 | 2212 | 19884 |
| 05 | 2281 | 2286 | 1976 | 2183 | 2224 | 546 | 2110 | 1873 | 2166 | 2233 | 19878 |
| 04 | 2166 | 2013 | 2190 | 2208 | 2206 | 2195 | 409 | 2203 | 2137 | 2118 | 19845 |
| 03 | 2301 | 1880 | 2130 | 2128 | 2203 | 2206 | 1943 | 641 | 2295 | 2188 | 19915 |
| 02 | 2076 | 2339 | 2281 | 2256 | 2200 | 1936 | 2180 | 2294 | 641 | 2301 | 20504 |
| 01 | 2073 | 2375 | 2291 | 2291 | 2305 | 2008 | 2300 | 2171 | 2173 | 561 | 20548 |



Rys. 5A. TEST 4, ciąg Korobowa.

200 000, jedynie w teście TEST3 uwzględniono 20 000 000 liczb. W przypadku CUDów jest to tylko krótki początkowy odcinek.

### Prace Cytowane


Циглер Л., Хелмберг Г. (1963), Новей шее развитие теории равномерного распределения, Математика, 7:3.

Devroye L. (1986), Non-Uniform Random Variate Generation, Springer, New York.

Franklin J. N. (1963), Deterministic simulation of random processes, *Math. Comp.* 17, 28–59.

Knopp K. (1956), Szeregi nieskończone, PWN, Warszawa.

Knuth D. E. (1965), Construction of a random sequence, *BIT* 5, 246–250.

Knuth D. E. (1969), The art of computer programming, Vol. 2, Seminumerical algorithms, Addison-Wesley.

Kolomogorov A. N. (1963), On tables of random numbers, *Sankhya A* 25, 369–376.

Коробов Н. М. (1956), О вполне равномерном распределении и совместно нормальных числах , Изв. АН. СССР, сер. матем. 20, 649–660.

Коробов Н. М. (1960), Оценки тригонометрических сумм вполне равномерно распределеными функцями, Доклады Академии Наук СССР, 133, 5, 1011–1014.

Lehmer D. H. (1951), Mathematical methods in large-scale computing units. The Annals of the Computation Laboratory of Harvard University XXVI, Proceeding of a Second Symposium on Large-scale Digital Calculation Machinery 13–16 September 1949, Cambridge, Massachusetts, Harvard University Press, 141–146.

Левин М. Б. (1975), О равномерном распределении последовательности $\{\alpha \cdot \theta^x\}$, Матем. сборник, 98(140), 207–222.

Левин М. Б. (1981), О вполне равномерном распределении дробных долей показателной функции, Труды семинара им. Л. Г. Петровского 7, 245–256.

Marsaglia G. (1968), Random numbers fall mainly in the planes, *Proc. Nat. Acad. Sci. USA* 61, 25–28.

Von Mises R. (1919), Grundlagen der Wahrscheinlichkeitsrechnung, *Math. Zeitschrift* 5, 52–99.

Niederreiter H. (1978), Quasi-Monte Carlo methods and pseudo-random numbers, *Bull. Amer. Math. Soc.* 84, 957–1041.

Rauzy G. (1973), Fonctions entières et répartition modulo un, II, *Bull. Soc. math. France* 101, 185–192.

Старченко Л. П. (1959), Построение вполне равномерно распределенной последовательности, Доклады Академии Наук СССР, 129, 3, 519–521.

Wald A. (1937), Ergebnisse eines math., *Kolloquiums* 8, Vienna, 38–72.

Weyl H. (1916), Über die Gleichverteilung von Zahlen mod. Eins, *Matematische Annalen* 77, 313–352.

Zieliński R. (1970), Metody Monte Carlo. WNT, Warszawa.

Zieliński R. (1972), Generatory liczb losowych. Programowanie i testowanie na maszynach cyfrowych, WNT, Warszawa (wyd. 2 1979).

Zieliński R. (1989), Wytwarzanie losowości, Preprint 21, IM PAN, 3–27.

Zieliński R. (1990), An aperiodic pseudorandom number generator, *Journal of Computational and Applied Mathematics* 31, 205–210.

**Abstract**

The paper presents the problem of computer implementation of completely uniformly distributed (CUD) sequences. In the first part of the paper a few examples of CUD sequences are presented. In the second part the computer implementations of the Korobow and Starczenko sequences are presented. Several tests are performed on the constructed generators.