Wiadomości Statystyczne. The Polish Statistician, 2019, vol. 64, 7, 5–31 DOI: 10.5604/01.3001.0013.7605
Studia metodologiczne / Methodological studies

Porównanie generatorów liczb pseudolosowych

Piotr Sulewski[a]

Streszczenie. Losowanie prób w badaniach statystycznych i w obliczeniach numerycznych, jak również symulacyjne badanie modeli probabilistycznych właściwie we wszystkich dziedzinach wiedzy wymagają wyposażenia komputera w generatory liczb pseudolosowych. Głównym celem pracy jest porównanie generatorów liczb pseudolosowych normalnych na podstawie ich analizy dokonanej za pomocą różnego rodzaju kryteriów. Zbadano właściwości 12 generatorów liczb pseudolosowych o rozkładzie normalnym. Zaproponowano rozszerzenie rodziny generatorów o dwa tzw. generatory aplikacyjne oraz przyjęcie nowego podejścia do sprawdzania jakości generatorów. Przedstawiono narzędzie przygotowane w języku C++ oraz w języku Visual Basic for Application (VBA) do prowadzenia samodzielnych badań z użyciem generatorów. Symulacje Monte Carlo przeprowadzono w języku C++, a obliczenia wykonano w edytorze VBA przy użyciu arkusza kalkulacyjnego Microsoft Excel 2016. Analiza uzyskanych wyników wskazuje, że najlepsze właściwości mają generatory: MP Monty Pythona, R, Biegun oraz Ziggurat. Najmniej użyteczne okazują się generatory: BM Boxa-Mullera, Wallace'a, Iloraz oraz Excel.

Słowa kluczowe: rozkład normalny, generator liczb pseudolosowych, symulacje Monte Carlo

Comparison of normal random number generators

Summary. The sampling in statistical surveys and numerical calculations as well as simulation testing of probabilistic models in virtually all fields of knowledge require a computer endowed with pseudorandom numbers generators. The main goal of the study is to compare the normal random number generators using various criteria. The properties of 12 random number generators for a normal distribution were investigated. Then, the family of generators was extended by two so-called application generators and a new approach for checking the quality of generators was adopted. A ready-made tool prepared in C++ and in Visual Basic for Application (VBA) for conducting self-contained research using generators was presented. All Monte Carlo simulations were carried out in C++, while the calculations were performed in the VBA editor using the Microsoft Excel 2016 spreadsheet. The analysis of the obtained results shows that the generators with best properties are: MP Monty Python, R, Biegun and Ziggurat. The worst generators, are: BM Box-Muller, Wallace, Iloraz and Excel.

Keywords: normal distribution, pseudo-random number generator, Monte Carlo simulation

JEL: C12, C15

[a] Akademia Pomorska w Słupsku, Wydział Matematyczno-Przyrodniczy.

Losowanie prób w kontekście badań statystycznych (badania reprezentacyjne, symulacyjne badanie estymatorów, testów i statystycznych reguł decyzyjnych) oraz w kontekście obliczeń numerycznych (metody Monte Carlo dla całek, równań z operatorami liniowymi oraz dla zadań optymalizacji), jak również symulacyjne badanie modeli probabilistycznych, szczególnie podczas analizowania zjawisk ekonomicznych, lecz także w technice, naukach przyrodniczych i właściwie we wszystkich pozostałych dziedzinach wiedzy, wymagają wyposażenia komputera w odpowiednie narzędzia (Wieczorkowski i Zieliński, 1997). Takimi narzędziami są generatory liczb pseudolosowych.

W ekonomii generatory liczb pseudolosowych znajdują zastosowanie w symulacjach i modelowaniu matematycznym (Ohirko, Zaniewski i Ogirko, 2016). Dotyczą one takich zagadnień, jak: ekonomia gospodarstwa domowego, obliczanie kosztów i budżetów, obliczanie podatków dochodowych, wyliczanie stóp procentowych w kredytach udzielanych przez banki, obliczenia procentowe z uwzględnieniem operacji bankowych, inwestycje i planowanie remontów, obliczanie wynagrodzeń pracowniczych, koszty finansowe zakładu pracy, księgi podatkowe, bilanse księgowe, ustalanie wyniku finansowego metodą statystyczną, wycena aktywów i kapitałów z uwzględnieniem amortyzacji danego środka oraz obliczanie inwestycji długoterminowych. Modelowanie i symulacje tych zjawisk mają pozytywny wpływ na podejmowane decyzje.

Liczby pseudolosowe o rozkładzie normalnym są powszechnie stosowane m.in. w symulacjach Monte Carlo. W ciągu ostatnich 60 lat opublikowano kilkadziesiąt algorytmów generujących takie liczby. O ich użyteczności przesądza jakość. Pierwszą publikacją na ten temat była praca Mullera (1959). Później tematykę tę podejmowali m.in.: Kronmal (1964), Ahrens i Dieter (1972), Chen i Burford (1981), Schollmeyer i Tranter (1991) oraz Molle, Hinich i Morrice (1992). Porównania pseudolosowych liczb o rozkładzie normalnym dokonali Kabal (2000) oraz Thomas, Luk, Leong i Villasenor (2007). W pracy Thomasa i współpracowników (2007) zbadano zarówno jakość generatorów, jak i ich szybkość. W badaniach tych skorzystano z testu zgodności *chi-kwadrat*, z testów high-sigma multiple, konwersji liczb do rozkładu jednostajnego oraz testowania korelacji. Ze względu na szybkość działania wyróżniono generator Wallace, a ze względu na jakość – generator Ziggurat. Algorytmy zapisane w pseudokodzie rozpatrywano oddzielnie dla głównej części rozkładu Gaussa ($|x| \leq 3$) i dla tzw. ogonów tego rozkładu ($|x| > 3$).

W dobie dynamicznie rozwijającego się rynku komputerowego, w tym błyskawicznie rosnącej wydajności procesorów, badanie szybkości działania generatorów jest rzeczą drugorzędną. W związku z powyższym w badaniu omawianym w niniejszej pracy ograniczono się do porównania jakości generatorów.

Analizie statystycznej poddano 12 generatorów liczb pseudolosowych o rozkładzie normalnym oraz dwa tzw. generatory aplikacyjne. Wszystkie, z wyjątkiem

generatorów aplikacyjnych, zostały zaimplementowane w języku C++, a większość z nich także w języku VBA (Visual Basic for Application) wbudowanym w Microsoft Excel 2016. Dzięki temu czytelnik może samodzielnie przeanalizować kody źródłowe algorytmów i prowadzić badania, wykorzystując dostarczone generatory liczb pseudolosowych o rozkładzie normalnym[1].

W testowaniu jakości generatorów posłużono się narzędziami statystycznymi niewykorzystanymi w pracy Thomasa i współpracowników (2007), a mianowicie: testem Lilleforsa, testem Andersona-Darlinga, testami trendu i sezonowości, a także dwoma nowymi testami, z których jeden wykorzystuje moduł z różnicy dystrybuant teoretycznej i empirycznej, a drugi – statystyki pozycyjne.

Głównym celem artykułu jest porównanie generatorów liczb pseudolosowych normalnych na podstawie ich analizy przeprowadzonej przy użyciu różnego rodzaju kryteriów. Zbadano właściwości 12 generatorów.

ROZKŁAD NORMALNY

Rozkład Gaussa z wartością oczekiwaną $m = 0$ i odchyleniem standardowym $s = 1$, zwany także rozkładem normalnym standaryzowanym $N(0,1)$, ma funkcję gęstości prawdopodobieństwa w postaci:

$$\varphi(x) = \frac{1}{\sqrt{2\pi}} \exp\left(-\frac{x^2}{2}\right) \tag{1}$$

Odwrócenie funkcji gęstości (1) w wyniku nieskomplikowanych przekształceń daje równość:

$$\varphi^{-1}(y) = \sqrt{-\ln(2\pi y^2)} \tag{2}$$

Dystrybuanta teoretyczna rozkładu $N(0,1)$ dana jest wzorem:

$$\Phi(x) = \frac{1}{\sqrt{2\pi}} \int_{-\infty}^{x} \exp\left(-\frac{t^2}{2}\right) dt = \frac{1}{2}\left[1 + erf\left(\frac{x}{\sqrt{2}}\right)\right] \tag{3}$$

gdzie *erf* jest funkcją błędu Gaussa. Wartość dystrybuanty (3) można wyznaczyć za pomocą wbudowanej funkcji Excela ROZKŁAD.NORMALNY(x;0;1;1). Funkcja ta jest akceptowalna do stosowania w rozbudowanej analizie statystycznej (Mélard, 2014). Wartość dystrybuanty (3) można obliczyć także za pomocą kodu języka VBA w postaci APPLICATION.NORMDIST(x,0,1,1).

[1] Stosowne pliki są umieszczone pod adresem https:/sulewski.apsl.edu.pl/index.php/publikacje.

Niech zmienna losowa *X* ma rozkład *N*(0,1), a $\varphi(x)$ będzie funkcją gęstości daną wzorem (1). Wartość oczekiwana zmiennej losowej *X* wynosi 0, ponieważ:

$$m = \int_{-\infty}^{\infty} x\,\varphi(x)dx = 0 \tag{4}$$

Momenty centralne zmiennej losowej *X* mają w tej sytuacji postać:

$$\mu_k = \int_{-\infty}^{\infty} (x-m)^k \varphi(x)dx = \int_{-\infty}^{\infty} x^k\,\varphi(x)dx \tag{5}$$

Jeżeli zmienna losowa *X* ma rozkład *N*(0,1), to zmienna losowa $Y = Xs + m$ ma rozkład *N*(*m, s*).

Charakterystyki rozkładu *N*(0,1) przyjmują następujące wartości teoretyczne (Cramér, 1958):

- wartość oczekiwana $m = 0$,
- odchylenie standardowe $s = \sqrt{\mu_2} = 1$,
- momenty centralne: $\mu_3 = 0$, $\mu_4 = 3$.

Niech x_i (i = 1, 2, …, n) będzie n-elementową próbą o rozkładzie normalnym. Wartość średnią z próby definiuje się jako:

$$\alpha_1 = \frac{1}{n}\sum_{i=1}^{n} x_i \tag{6}$$

natomiast momenty centralne z próby to:

$$M_k = \frac{1}{n}\sum_{i=1}^{n} (x_i - \alpha_1)^k \tag{7}$$

Estymatory nieobciążone momentów centralnych mają postać (Cramér, 1958):

$$M_2^n = \frac{n}{n-1}M_2 \qquad M_3^n = \frac{n^2}{(n-1)(n-2)}M_3$$
$$M_4^n = \frac{n(n^2-2n+3)}{(n-1)(n-2)(n-3)}M_4 - \frac{3n(2n-3)}{(n-1)(n-2)(n-3)}M_2 \tag{8}$$

GENERATORY LICZB PSEUDOLOSOWYCH O ROZKŁADZIE JEDNOSTAJNYM

W generacji liczb pseudolosowych o rozkładzie normalnym ważne są nie tylko algorytmy opisujące poszczególne jej metody, lecz przede wszystkim algorytmy, które charakteryzują liczby pseudolosowe o rozkładzie jednostajnym (inaczej równomiernym) $U(0,1)$ wykorzystywane przez te generatory.

Testowanie liczb losowych o rozkładzie jednostajnym omówił L'Ecuyer (1992, 2001). Zestawy testów i procedur można znaleźć także w pracy Rukhina i współpracowników (2001). Testy Dieharda zostały opracowane przez Marsaglię (1997). Innym kompleksowym zestawem jest TestU01 (L'Ecuyer, Simard, 2005), który zapewnia szeroki wybór testów, takich jak np. test Crusha (L'Ecuyer, 2001).

W niniejszej pracy skorzystano z bardzo popularnego algorytmu Mersenne Twister autorstwa japońskich naukowców Matsumoto i Nishimura (1998), generującego liczby pseudolosowe o rozkładzie jednostajnym o bardzo dużym okresie $2^{19937} - 1$. Algorytm ten jest wbudowany w takie środowiska obliczeniowe, jak Excel, w tym VBA (od wersji 2010), Calc (od wersji Apache OpenOffice 4.0), Scilab i język R. Na potrzeby tego artykułu algorytm Mersenne Twister został zapisany także w języku C++, ponieważ z procedury Mersenne Twister korzysta każdy z omawianych tu generatorów. Zapewniło to możliwość porównywania ze sobą generatorów liczb o rozkładzie normalnym niezależnie od tego, czy zostały one zaprogramowane w języku C++, czy w języku VBA (Excel 2016), czy też pochodzą z programu Scilab. Opierając się na sugestiach zawartych w pracy Mélarda (2014), do generacji liczb losowych o rozkładzie jednostajnym w języku VBA wykorzystano formułę EVALUATE ("RAND()") zamiast formuły RND.

WŁAŚCIWOŚCI GENERATORÓW LICZB PSEUDOLOSOWYCH O ROZKŁADZIE NORMALNYM

Dwanaście najważniejszych zdaniem autora generatorów liczb pseudolosowych o rozkładzie normalnym zostało zdefiniowanych i zapisanych w formie algorytmów. Algorytmy te dotyczą gęstości rozkładu Gaussa w całej swojej dziedzinie wraz z tzw. ogonami rozkładu.

Generator CdfInv odwracania dystrybuanty

Niech U_1 będzie liczbą pseudolosową $U(0,1)$. Wówczas liczba pseudolosowa X z $N(0,1)$ ma postać $X = \Phi^{-1}(U_1)$, gdzie Φ^{-1} jest funkcją odwrotną do dystrybuanty $N(0,1)$ daną wzorem (2). Jedną z pierwszych metod aproksymacji funkcji Φ^{-1} zaproponował Muller (1958). Zagadnieniem tym zajmowali się także:

Gebhardt (1964), Wetherill (1965), Wichura (1988), McCollum, Lancaster, Bouldin i Peterson (2003) oraz Chen, Moon i Bazargan (2004). W artykule skorzystano z wbudowanej funkcji użytkownika APPLICATION.NORMINV (U_1, 0,1) z edytora VBA. Analiza pracy Mélarda (2014) wskazuje, że z funkcji tej można korzystać w badaniach statystycznych.

Generator Boxa-Mullera (BM)

Niech U_1, U_2 będą liczbami pseudolosowymi $U(0,1)$. Metoda Boxa-Mullera (Box i Muller, 1958) generująca parę niezależnych liczb pseudolosowych (Y_1, Y_2) z $N(m, s)$ jest opisana za pomocą algorytmu:

1. $a = \sqrt{-2\ln(U_1)}$, $b = 2\pi U_2$
2. $X_1 = a\sin(b)$, $X_2 = a\cos(b)$
3. $Y_1 = sX_1 + m$, $Y_2 = sX_2 + m$

W testach można np. skorzystać tylko z jednej wartości Y_1 lub Y_2. Innym rozwiązaniem jest zwracanie przez algorytm wartości Y_1, a wartość Y_2 jest buforowana i zwracana podczas ponownego uruchomienia algorytmu.

Generator Tuzin

Niech U_1, U_2, …, U_{12} będą liczbami pseudolosowymi $U(0,1)$. Metoda tuzina generująca liczbę pseudolosową Y z $N(m, s)$ jest opisana za pomocą algorytmu:

1. $S = \sum_{i=1}^{12} U_i$
2. $X = S - 6$
3. $Y = sX + m$

Generator Biegun

Niech U_1, U_2 będą liczbami pseudolosowymi $U(0,1)$. Metoda biegunowa (Bell, 1968; Knop, 1969) generująca parę niezależnych liczb pseudolosowych (Y_1, Y_2) z $N(m, s)$ jest opisana za pomocą algorytmu:

1. $a = -1 + 2U_1$, $b = -1 + 2U_2$
2. $d = a^2 + b^2$
3. Jeżeli $d \geq 1$, to przejdź do pkt 1
4. $e = \sqrt{-2\ln(d)/d}$
5. $X_1 = ae$, $X_2 = be$
6. $Y_1 = sX_1 + m$, $Y_2 = sX_2 + m$

W testach można skorzystać np. tylko z jednej wartości Y_1 lub Y_2. Innym rozwiązaniem jest zwracanie przez algorytm wartości Y_1, podczas gdy wartość Y_2 jest buforowana i zwracana przy ponownym uruchomieniu algorytmu.

Generator Iloraz

Niech U_1, U_2 będą liczbami pseudolosowymi $U(0,1)$. Metoda ilorazowa (Kinderman i Monahan, 1977) generująca liczbę pseudolosową Y z $N(m, s)$ jest opisana za pomocą algorytmu (Wieczorkowski i Zieliński, 1997):

1. $u = U_1$, $e = \exp(1)$, $v = -\sqrt{2/e} + 2\sqrt{2/e}U_2$
2. $X = v/u$
3. Jeżeli $X^2 \le 2(3 - u(4 + u))$, to przejdź do pkt 6
4. Jeżeli $X^2 \le 2/u - 2u$ i $X^2 \le -4\ln(u)$, to przejdź do pkt 6
5. Przejdź do pkt 1
6. $Y = sX + m$

Generator Levy

Niech U_1, U_2 będą liczbami pseudolosowymi $U(0,1)$. Metoda Levy (Leva, 1992) generująca liczbę pseudolosową Y z $N(m, s)$ jest opisana za pomocą poniższego algorytmu. W pkt 7 przedstawiono drobną korektę autorską, bez uwzględnienia której zbyt liczne wartości $X = 0$ obniżają jakość generatora.

1. $e = \exp(1)$, $a = 0{,}196$, $b = 0{,}25472$, $ss = 0{,}449871$, $t = -0{,}386595$
2. $u = U_1$, $v = -\sqrt{2/e} + 2U_2\sqrt{2/e}$
3. $x = u - ss$, $y = |v| - t$, $Q = x^2 + y(ay - bx)$
4. Jeżeli $Q < 0{,}27597$, to przejdź do pkt 7
5. Jeżeli $Q > 0{,}27846$, to przejdź do pkt 2
6. Jeżeli $v^2 > -4u^2$, to przejdź do pkt 2
7. Jeżeli $v/u = 0$, to przejdź do pkt 2
8. $X = v/u$
9. $Y = sX + m$

Generator Ahrensa-Dietera (AD)

Niech U_1, U_2, U_3 będą liczbami pseudolosowymi $U(0,1)$. Metoda Ahrensa-Dietera (Ahrens i Dieter, 1988) generująca parę niezależnych liczb pseudolosowych (Y_1, Y_2) z $N(m, s)$ jest opisana za pomocą algorytmu:

1. Jeżeli $U_1 < 0{,}5$, to $a = 1$. Przejdź do pkt 3
2. $a = -1$

3. $b = -\ln(U_2)$
4. $c = \operatorname{tg}[\pi(U_3 - 0{,}5)]$
5. $d = \sqrt{2b/(1+c^2)}$
6. $X_1 = ad$, $X_2 = cd$
7. $Y_1 = sX_1 + m$, $Y_2 = sX_2 + m$

W testach można skorzystać np. tylko z jednej wartości Y_1 lub Y_2. Inne rozwiązanie polega na zwracaniu przez algorytm wartości Y_1 i buforowaniu wartości Y_2, która jest zwracana podczas ponownego uruchomienia algorytmu.

Generator Marsaglii-Braya (MB)

Niech $U_1, U_2, \ldots, U_{10}$ będą liczbami pseudolosowymi $U(0,1)$. Metoda Marsaglii-Braya (Marsaglia i Bray, 1964) generująca liczbę pseudolosową Y z $N(m, s)$ jest opisana za pomocą algorytmu, który wywołuje funkcję pomocniczą w postaci:

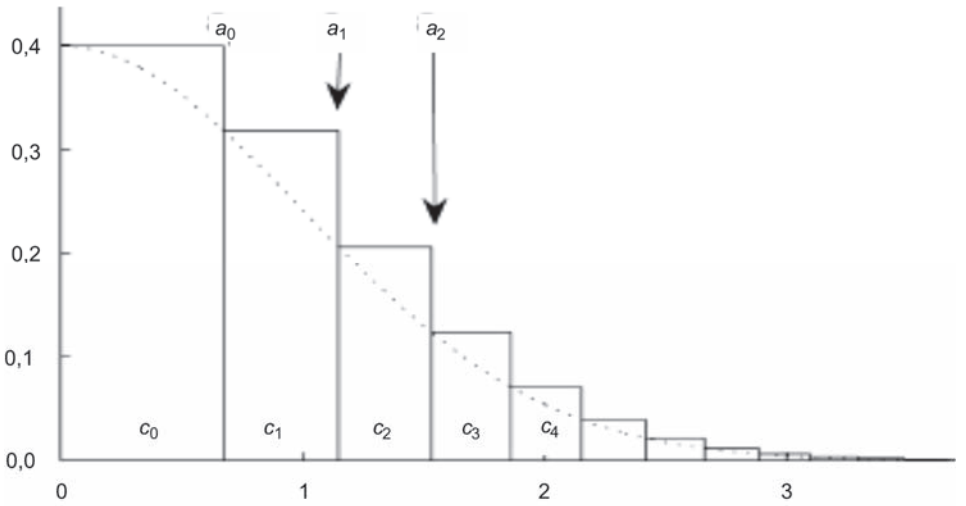
$$g_3(x,a,b,c,d) = \begin{cases} a\exp(-0{,}5x^2) - b(3-x^2) - c(1{,}5-|x|) \text{ dla } |x| \le 1 \\ a\exp(-0{,}5x^2) - d(3-|x|)^2 - c(1{,}5-|x|) \text{ dla } 1 < |x| \le 1{,}5 \\ a\exp(-0{,}5x^2) - d(3-|x|)^2 \text{ dla } 1{,}5 < |x| \le 3 \\ 0 \text{ dla } 3 < |x| \end{cases}$$

1. $a = 17{,}49731196$, $b = 4{,}73570326$, $c = 2{,}15787533$, $d = 2{,}36785163$
2. $a_1 = 0{,}8638$, $a_2 = 0{,}1107$, $a_3 = 0{,}0228002039$
3. $a_4 = 1 - a_1 - a_2 - a_3$
4. $u = U_1$
5. Jeżeli $u < a_1$, to $X = 2(U_2 + U_3 + U_4 - 1{,}5)$. Przejdź do pkt 14
6. Jeżeli $u < a_1 + a_2$, to $X = 1{,}5(U_5 + U_6 - 1)$. Przejdź do pkt 14
7. Jeżeli $u < a_1 + a_2 + a_3$, to $x = 6U_7 - 3$, $y = 0{,}358U_8$
8. Jeżeli $y \ge g_3(x, a, b, c, d)$, to przejdź do pkt 7
9. $X = x$, przejdź do pkt 14
10. $f = -1 + 2U_9$, $g = U_{10}$
11. $x = -\ln(|f|)/3, \quad y = -\ln(g)$
12. Jeżeli $2y \le x^2$, to przejdź do pkt 10
13. Jeżeli $f > 0$, to $X = x + 3$. W sytuacji przeciwnej $X = -x - 3$
14. $Y = sX + m$

Generator Grand

Niech $U_1, U_2, \ldots, U_6$ będą liczbami pseudolosowymi $U(0,1)$. Wykres 1 przedstawia ideę generatora Grand, tzn. podział gęstości normalnej na oddzielne zakresy. Zakres c_i jest wybierany z prawdopodobieństwem 2^{-i-1}.

WYKR. 1. PODZIAŁ GĘSTOŚCI NORMALNEJ NA ODDZIELNE ZAKRESY c_i



U w a g a. $i = 0, 1, 2 \ldots$.
Ź r ó d ł o: Thomas i współpracownicy (2007).

Metoda Grand (Brent, 1974) generująca liczbę pseudolosową Y z $N(m, s)$ jest opisana za pomocą poniższego algorytmu, w którym $t_i = \Phi^{-1}(1 - 2^{-i-1})$, $i = 0, 1, 2, \ldots$.

1. $a = U_1$, $i = 0$
2. Dopóki $a < 0{,}5$ wykonuj kroki 2.1 i 2.2
   2.1. $a = 2a$
   2.2. $i = i + 1$
3. $u = (t_{i+1} - t_i)U_2$, $v = u(0{,}5u + t_i)$
4. Dopóki $v \geq U_6$ wykonuj kroki 4.1–4.3
   4.1. Jeżeli $v \geq U_3$, to przejdź do pkt 4.3
   4.2. Jeżeli $U_4 < 0{,}5$, to $X = t_i + u$, w przeciwnym razie $X = -t_i - u$. Przejdź do pkt 5
   4.3. $v = U_5$
5. $Y = sX + m$

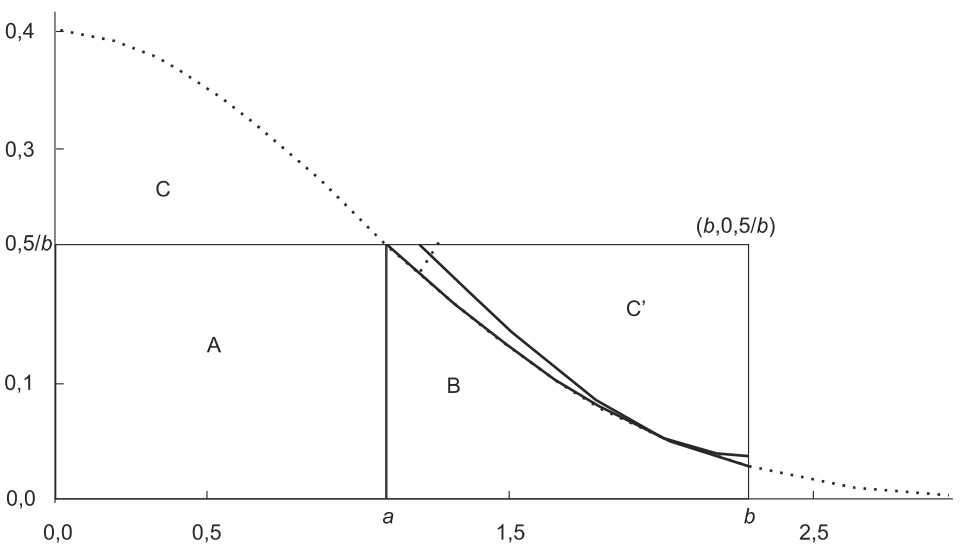
W środowisku matematycznym SciLab generator Grand został zaimplementowany dla różnych rozkładów, w tym dla rozkładu normalnego. Wygenerowanie na potrzeby symulacji Monte Carlo tablicy n x u liczb pseudolosowych o rozkładzie normalnym $N(m, s)$ i wyeksportowanie tej tablicy do pliku data.csv zostało

wykonane w środowisku SciLab za pomocą polecenia: csvWrite(grand(n, u,"nor", m, s), fullfile("data.csv"),",",",").

Generator Monty Pythona (MP)

Niech U_1, U_2, …, U_5 będą liczbami pseudolosowymi $U(0,1)$. Wykres 2 przedstawia ideę generatora Monty Pythona, tzn. spakowanie rozkładu Gaussa do obszaru prostokątnego. Jeżeli punkt losowy (x, y) znajduje się w obszarze C', to należy go przesunąć do obszaru C (zob. pkt 6 poniższego algorytmu). Funkcja *int* oznacza część całkowitą z liczby, funkcja gęstości $\varphi(x)$ dana jest wzorem (1), a wyrażenie $\varphi^{-1}(x)$ dane jest wzorem (2).

WYKR. 2. SPAKOWANIE ROZKŁADU GAUSSA DO OBSZARU PROSTOKĄTNEGO



Ź r ó d ł o: jak przy wykr. 1.

Metoda Monty Pythona (Marsaglia i Tsang, 1998) generująca liczbę pseudolosową Y z $N(m, s)$ została opisana za pomocą algorytmu:

1. $b = \sqrt{2\pi} = 2{,}5066, a = \varphi^{-1}(1/2b) = 1{,}1774$
2. $c = 2int(2U_1) - 1$, $x = bU_2$
3. Jeżeli $x < a$, to $X = cx$. Przejdź do pkt 12
4. $y = U_3/(2b)$
5. Jeżeli $y < \varphi(x)$, to $X = cx$. Przejdź do pkt 12
6. $(x, y)^{C'} \rightarrow (x, y)^{C}$
7. Jeżeli $y < \varphi(x)$, to $X = cx$. Przejdź do pkt 12

8. $d = -1 + 2U_4$, $f = U_5$
9. $x = -\ln(|d|)/3, y = -\ln(f)$
10. Jeżeli $2y \leq x^2$, to przejdź do pkt 8
11. Jeżeli $d > 0$, to $X = x + 3$. W sytuacji przeciwnej $X = -x - 3$
12. $Y = sX + m$

Generator Wallace'a

Generator Wallace'a (2005) jest najszybszym generatorem liczb pseudolosowych o rozkładzie normalnym $N(0,1)$. Ze względu na programistyczny i bardzo skomplikowany charakter algorytmu go opisującego w niniejszej pracy ograniczono się do implementacji komputerowej generatora. Szczegółowy algorytm został przedstawiony w pracy Thomasa i współpracowników (2007).

Generator Ziggurat

Niech $U_1, U_2, \ldots, U_5$ będą liczbami pseudolosowymi $U(0,1)$. Metoda Ziggurat (Marsaglia i Tsang, 2000) generująca liczbę pseudolosową Y z $N(m, s)$ jest opisana za pomocą poniższego algorytmu. Funkcja *int* oznacza część całkowitą z liczby, natomiast wyrażenia $\varphi(x)$ oraz $\varphi^{-1}(x)$ dane są odpowiednio wzorami (1) i (2).

1. $n = 255$, $v = 0{,}00492867323399$, $r = 3{,}6541528853610088$
2. $x_{255} = r$
3. Dla $i = 254, 253, \ldots, 1$
   3.1. $x_i = \varphi^{-1}\left(\frac{v}{x_{i+1}} + \varphi(x_{i+1})\right)$
4. $i = 1 + int(nU_1)$
5. $x = x_iU_2$
6. Jeżeli $|x| < x_{i-1}$, to $X = x$. Przejdź do pkt 10
7. Jeżeli $i \neq n$, to $y = (\varphi(x_{i-1}) - \varphi(x_i))U_3$
8. Jeżeli $y < (\varphi(x) - \varphi(x_i))$, to $X = x$. Przejdź do pkt 10
9. Jeżeli $|x| \geq x_{i-1}$, to:
   9.1. $d = -1 + 2U_4$, $f = U_5$
   9.2. $x = -\ln(|d|)/3, y = -\ln(f)$
   9.3. Jeżeli $2y \leq x^2$, to przejdź do pkt 9.1
   9.4. Jeżeli $d > 0$, to $X = x + 3$. W sytuacji przeciwnej $X = -x - 3$
10. $Y = sX + m$

GENERATORY APLIKACYJNE

Pierwszy generator aplikacyjny to Excel 2016. Liczby o rozkładzie jednostajnym powstają w tym programie za pomocą algorytmu Mersenne Twister (Matsumoto i Nishimura, 1998). Jednakże algorytm generowania liczb pseudoloso-

wych o rozkładzie normalnym za pomocą narzędzia „analiza danych" nie jest znany. Drugim generatorem aplikacyjnym jest język R. Także w nim liczby o rozkładzie jednostajnym powstają za pomocą algorytmu Mersenne Twister. Z kolei liczby o rozkładzie normalnym są tworzone metodą odwracania dystrybuanty zaproponowaną przez Wichurę (1988). Do generatorów aplikacyjnych nie można zaliczyć Mathcada, ponieważ w jego przypadku brakuje informacji na temat sposobu generowania liczb pseudolosowych o rozkładzie jednostajnym. Do tej grupy nie można zaliczyć również programu Calc z zestawu Apache OpenOffice 4.1.5. Wprawdzie liczby losowe o rozkładzie jednostajnym generowane są w tym programie za pomocą algorytmu Mersenne Twister, ale brakuje narzędzia do generowania liczb pseudolosowych o rozkładzie normalnym. Ze względu na specyfikę tych generatorów poniżej opisano sposób ich otrzymywania w dwóch środowiskach aplikacyjnych.

Generator Excel

W programie Microsoft Excel 2016 do generowania liczb pseudolosowych o rozkładzie normalnym wykorzystuje się narzędzie „analiza danych" oraz opcję „generowanie liczb pseudolosowych". Należy wybrać polecenie „rozkład normalny" oraz określić wartości m i s, liczbę liczb pseudolosowych n i liczbę zmiennych u.

Generator R

W środowisku R za pomocą formuły *rnorm*(n,*mean* = 0, *sd* = 1) otrzymano n liczb X z $N(0,1)$. Wartość Y z $N(m, s)$ otrzymano za pomocą formuły $Y = Xs + m$. Ciąg takich n liczb można wyeksportować do pliku csv za pomocą polecenia: write.csv2(rnorm(n),'plik.csv',row.names = F, na = '', fileEncoding = "Windows-1250").

TESTOWANIE GENERATORÓW

Testowanie generatorów liczb pseudolosowych polega na sprawdzaniu hipotezy, że otrzymany za pomocą danego generatora ciąg liczb jest próbką z populacji o danej dystrybuancie, w tym przypadku o dystrybuancie rozkładu normalnego wyrażonej wzorem (3).

Celem tej części pracy jest uzupełnienie metod testowania generatorów o sposoby nieopisane w pracy Thomasa i współpracowników (2007). Przedstawione wyniki dotyczą rozkładu normalnego standaryzowanego $N(0,1)$. Testowanie generatorów przeprowadzono za pomocą sześciu kryteriów: czterech klasycznych i dwóch nowych. W przypadku kryteriów klasycznych wyznaczono przy użyciu omawianych generatorów podstawowe charakterystyki rozkładu

i porównano je z wartościami teoretycznymi, zastosowano testy Lillieforsa i Andersona-Darlinga oraz test trendu i test sezonowości. W przypadku nowych kryteriów zbadano zgodność dystrybuant teoretycznej i empirycznej, a także przetestowano generatory za pomocą statystyk pozycyjnych. Symulacje Monte Carlo na potrzeby analizowanych kryteriów przeprowadzono dla $u = 1000$ zestawów (liczby powtórzeń) prób o liczebności $n = 1000$. Wszystkie obliczenia przeprowadzono na takich samych zbiorach danych. Wartości n mogłyby być większe, gdyby nie kryterium opisane przez statystyki pozycyjne (zob. *Kryterium 4: statystyki pozycyjne*). Ograniczeniami dla wartości u są tylko czas trwania symulacji i liczba 16384 kolumn w arkuszu. Przy bardzo wydajnych procesorach z drugim ograniczeniem na pewno można sobie poradzić.

Kryterium 1: charakterystyki rozkładu normalnego standaryzowanego

Algorytm wyznaczania wartości średnich charakterystyk dla generatora jest następujący:

1. Ustal liczebność próby n
2. Ustal wartość zmiennych $S_j = 0$ ($j = 1, 2, \ldots, 4$)
3. Kroki 3.1–3.3 powtórz u razy:
   3.1. Wygeneruj n-elementową próbę za pomocą generatora
   3.2. Oblicz wartości charakterystyk α_1, $\sqrt{M_2^n}$, M_3^n, M_4^n zgodnie ze wzorami (6)–(8)
   3.3. Oblicz $S_1 = S_1 + |\alpha_1 - m|$, $S_2 = S_2 + \left|\sqrt{M_2^n} - s\right|$, $S_3 = S_3 + |M_3^n - \mu_3|$ i $S_4 = S_4 + |M_4^n - \mu_4|$, gdzie m, s, μ_3 i μ_4 przyjmują odpowiednio wartości: 0, 1, 0 i 3
4. Oblicz $\overline{S_j} = \frac{S_j}{u} (j = 1, 2, \ldots, 4)$

Tablica 1 przedstawia wyniki uzyskane za pomocą powyższego algorytmu dla liczebności próby n i liczby powtórzeń u. Wartości najmniejsze zostały pogrubione.

TABL. 1. WARTOŚCI MIAR $\overline{S_j}$ DLA $n = u = 10^3$ WRAZ Z RANKINGIEM

Generatory	$\overline{S_1}$	R	$\overline{S_2}$	R	$\overline{S_3}$	R	$\overline{S_4}$	R
CdfInv	0,02520	7	0,01829	10	0,06207	10	**0,11811**	1
BM	0,02527	8	0,02762	14	**0,05057**	1	0,22614	13
Tuzin	0,02570	10	**0,01649**	1	**0,05836**	2	0,13594	12
Biegun	**0,02447**	3	0,01800	6	0,05901	4	0,12149	4
Iloraz	0,02616	13	0,01846	11	0,06432	12	**0,12016**	3
Levy	0,02619	14	0,01886	13	**0,05900**	3	0,12341	8
AD	0,02497	6	0,01805	8	0,06203	9	**0,12005**	2
MB	0,02567	9	**0,01736**	2	0,06154	6	0,12579	10

TABL. 1. WARTOŚCI MIAR $\bar{S}_j$ DLA $n = u = 10^3$ WRAZ Z RANKINGIEM (dok.)

Generatory	$\bar{S}_1$	R	$\bar{S}_2$	R	$\bar{S}_3$	R	$\bar{S}_4$	R
Grand	**0,02388**	2	0,01828	9	0,06257	11	0,12555	9
MP	**0,02383**	1	0,01778	4	0,06167	7	0,12309	7
Wallace'a	0,02583	11	0,01788	5	0,06473	13	0,12231	6
Ziggurat	0,02469	4	0,01805	7	0,06177	8	0,13350	11
Excel	0,02591	12	0,01868	12	0,08069	14	0,29809	14
R	0,02489	5	**0,01776**	3	0,06057	5	0,12180	5

U w a g a. $j = 1, 2, 3, 4$. R – ranking.
Ź r ó d ł o: opracowanie własne.

Wyniki zaprezentowane w tabl. 1 wskazują, że ze względu na charakterystyki rozkładu $N(0,1)$ wyróżniają się generatory Biegun, R i MP. Najgorzej z analizowanych generatorów wypadają Excel, Iloraz i Levy.

Kryterium 2: testy Lillieforsa i Andersona-Darlinga

Generatory pseudolosowych liczb normalnych sprawdzono za pomocą testu Andersona-Darlinga (A-D) (Anderson i Darling, 1952) oraz testu Lillieforsa (LF) (Lilliefors, 1967). Pominięto test Shapiro-Wilka (Shapiro i Wilk, 1965), stosowany w swojej pierwotnej postaci dla $n \leq 5000$, którego skuteczność jest podobna do skuteczności testu A-D (Razali i Wah, 2011). Natomiast test Kołmogorowa-Smirnowa (K-S), również pominięty, charakteryzuje się najmniejszą mocą spośród wyżej wymienionych (Razali i Wah, 2011).

Test LF, sprawdzający, czy próba $x_1, x_2, \ldots, x_n$ ma rozkład $N(0,1)$, korzysta ze statystyki:

$$L = \max_{1 \leq i \leq n} \left\{ \max \left\{ \left| \Phi\left(\frac{x - \alpha_1}{\sqrt{M_2^n}} \right) - \frac{i-1}{n} \right|, \left| \Phi\left(\frac{x - \alpha_1}{\sqrt{M_2^n}} \right) - \frac{i}{n} \right| \right\} \right\} \tag{9}$$

gdzie wartość α_1 wyrażona jest wzorem (6), zaś wartość $\Phi(.)$ – wzorem (3) oraz

$$\sqrt{M_2^n} = \sqrt{\frac{1}{n-1} \sum_{i=1}^{n} (x_i - \alpha_1)^2} \tag{10}$$

Wartość p-value pv dla testu LF wyznaczono za pomocą algorytmu (Abdi i Molin, 2007):

1. Niech $b_0 = 0{,}37872256037043$, $b_1 = 1{,}3074818507879 + n$
2. Niech $b_2 = 0{,}08861783849346$

3. Oblicz $a = \left(-b_1 + \sqrt{b_1^2 - 4b_2(b_0 - L^{-2})}\right)/(2b_2)$
4. Oblicz $p_1 = -0{,}37782822932809 + 1{,}67819837908004a - + 3{,}02959249450445a^2$
5. Oblicz $p_2 = 2{,}8001579814210a^3 - 1{,}39874347510845a^4 - + 0{,}40466213484419a^5$
6. Oblicz $p_3 = -0{,}06353440854207a^6 + 0{,}0028746208762a^7 + 0{,}000696500131a^8$
7. Oblicz $p_4 = -0{,}00011872227037a^9 + 0{,}00000575586834a^{10}$
8. Oblicz $pv = p_1 + p_2 + p_3 + p_4$

Test A-D stanowi modyfikację testu Craméra-von Misesa (Cramér, 1928) dokonaną w celu poprawy jego czułości w „ogonach" rozkładu normalnego. Wykorzystuje on statystykę w postaci (Anderson i Darling, 1952; D'Agostino, 1986):

$$AD = A^2\left(1 + \frac{0{,}75}{n} + \frac{2{,}25}{n^2}\right) \tag{11}$$

gdzie:
$A^2 = -n - \sum_{i=1}^{n} \frac{2i-1}{n}\left[\ln\Phi\left(X_{(i)}\right) + \ln\left(1 - \Phi\left(X_{(n+1-i)}\right)\right)\right]$,
$X_{(i)}$ – *i*-ta zaobserwowana wartość w uporządkowanej rosnąco *n*-elementowej próbie.

Formuła na wyznaczenie pv dla testu A-D dana jest wyrażeniem (D'Agostino, 1986):

$$pv = \begin{cases} \exp\left(1{,}2937 - 5{,}709AD + 0{,}0186AD^2\right) \text{ dla } AD \geq 0{,}6 \\ \exp\left(0{,}9177 - 4{,}279AD - 1{,}38AD^2\right) \text{ dla } 0{,}34 \leq AD < 0{,}6 \\ 1 - \exp\left(-8{,}318 + 42{,}796AD - 59{,}938AD^2\right) \text{ dla } 0{,}2 < AD < 0{,}34 \\ 1 - \exp\left(-13{,}436 + 101{,}14AD - 223{,}73AD^2\right) \text{ dla } AD \leq 0{,}2 \end{cases} \tag{12}$$

Wyznaczanie wartości *p*-value dla testów LF i A-D zaimplementowano komputerowo (szczegóły znajdują się w rozdziale *Jak skorzystać z plików internetowych*).

Tablica 2 przedstawia średnie wartości *p*-value dla analizowanych testów, liczebności próby $n = 10^3$ i liczby powtórzeń $u = 10^3$. Wartości największe zostały pogrubione. Dla $n = 10^6$ otrzymano tak wysokie wartości statystyk (11), że formuła (12) sygnalizowała błąd. Liczebność próby $n = 10^6$ wpływała negatywnie także na test LF, czego skutkiem były ujemne wartości *p*-value dla generatorów Tuzin, Levy i MB.

TABL. 2. ŚREDNIE WARTOŚCI *p*-value DLA $n = u = 10^3$ WRAZ Z RANKINGIEM

Generatory	LF		A-D	
		R		R
CdfInv	0,48018	5	0,13879	12
BM	0,44516	14	0,12578	14
Tuzin	0,46894	13	0,14794	6
Biegun	0,47086	12	**0,15380**	2
Iloraz	0,48130	4	0,14165	9
Levy	0,47912	6	0,13917	10
AD	0,47580	9	0,14902	5
MB	**0,48298**	3	0,15017	4
Grand	**0,47315**	2	**0,13563**	3
MP	0,47835	7	0,14185	8
Wallace'a	0,47485	10	0,13908	11
Ziggurat	0,47832	8	0,13820	13
Excel	0,47232	11	0,14396	7
R	**0,49191**	1	**0,16000**	1

Ź r ó d ł o: jak przy tabl. 1.

Wyniki zaprezentowane w tabl. 2 wskazują, że w testach LF i A-D pozytywnie wyróżniają się generatory R, Grand i MB. Najgorsze okazują się generatory BM, Wallace i Ziggurat.

Kryterium 3: zgodność dystrybuant teoretycznej i empirycznej

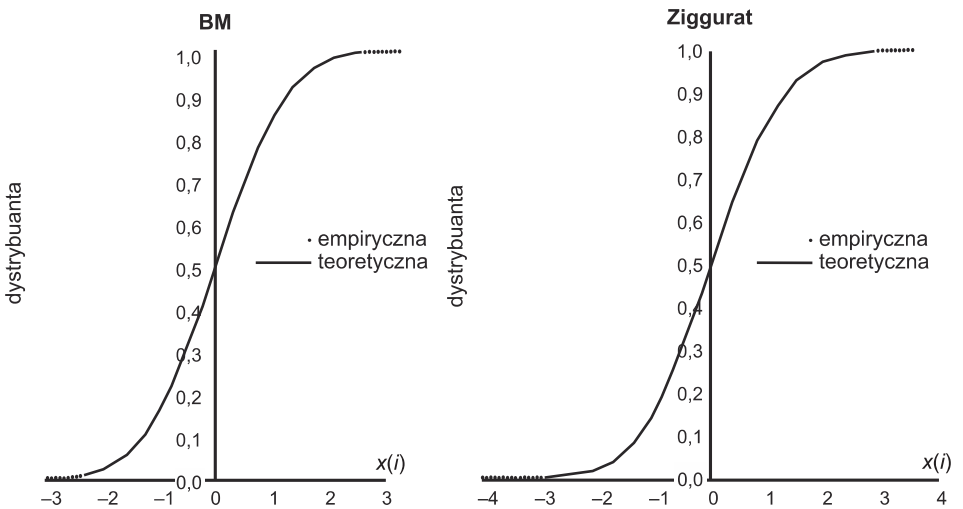
Propozycja autora dotycząca testowania jakości generatorów polega na zbadaniu zachowania się dystrybuanty empirycznej i teoretycznej. W literaturze statystycznej można znaleźć różne definicje dystrybuanty empirycznej. Dystrybuanta empiryczna, przeznaczona wyłącznie dla rozkładu normalnego, ma postać (Bloom, 1958):

$$F_i = \frac{i - 0{,}375}{n + 0{,}25} \tag{13}$$

Chociaż przebiegi dystrybuant empirycznej i teoretycznej można oczywiście przedstawić na wykresie, trudno jest w tym przypadku sporządzić ranking generatorów. Wykres 3 przedstawia przebiegi dystrybuant dla generatorów BM oraz Ziggurat, gdy $n = 10^4$.

Omówione wcześniej testy LF i A-D, podobnie jak testy K-S (Kołmogorow, 1933; Smirnov, 1948), Craméra-von Misesa (Cramér, 1928), Watsona (1961), R Rényiego (Rényi, 1953) i Kuipera (1960) zaliczane są do rodziny testów do badania normalności danej próby; grupa ta jest zdefiniowana na podstawie dystrybuanty empirycznej.

WYKR. 3. DYSTRYBUANTY EMPIRYCZNA I TEORETYCZNA DLA GENERATORÓW BM I ZIGGURAT



Ź r ó d ł o: jak przy tabl. 1.

Do badania jakości generatorów można także wykorzystać miarę:

$$M = \sum_{i=1}^{n} \left| \Phi(x_i) - F_i \right| \tag{14}$$

gdzie:
F_i – dystrybuanta empiryczna (13),
$\Phi(x_i)$ – dystrybuanta teoretyczna rozkładu normalnego (3),
$M \geq 0$.

Przy idealnej zgodności próby z rozkładem normalnym standaryzowanym $M = 0$. Im mniejsza wartość M, tym lepszą jakością cechuje się dany generator.

Algorytm wyznaczania wartości średniej miary M dla generatora jest następujący:

1. Ustal liczebność próby n
2. Kroki 2.1–2.3 powtórz u razy:
   2.1. Wygeneruj n liczb pseudolosowych $x_1, x_2, \ldots, x_n$ z $N(0,1)$
   2.2. Liczby uzyskane w kroku 2.1 posortuj w porządku rosnącym
   2.3. Oblicz wartość miary $M = \sum_{i=1}^{n} |F_i - \Phi(x_i)|$
3. Oblicz $\bar{M} = \frac{1}{u} \sum_{j=1}^{u} M_j$

Tablica 3 przedstawia średnie wartości miary M otrzymane przy użyciu tego algorytmu dla różnych wartości n i u wraz z rankingiem. Wartości najmniejsze zostały pogrubione.

TABL. 3. WARTOŚCI MIARY $\bar{M}$ DLA $n = u = 10^3$ WRAZ Z RANKINGIEM

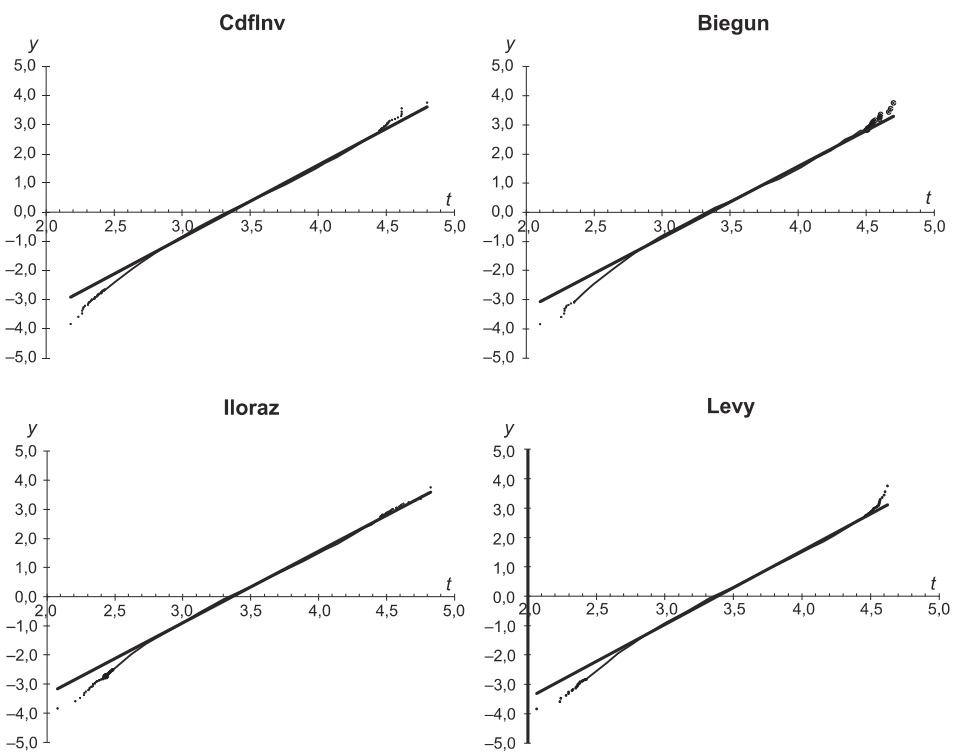
Generatory	$\bar{M}$	R	Generatory	$\bar{M}$	R
CdfInv	9,9766	8	MB	9,9069	5
BM	10,1210	11	Grand	10,1767	14
Tuzin	10,0909	10	MP	**9,6993**	1
Biegun	**9,7563**	3	Wallace'a	10,0787	9
Iloraz	10,1480	13	Ziggurat	**9,7460**	2
Levy	10,1355	12	Excel	9,9222	7
AD	9,8337	4	R	9,9105	6

Ź r ó d ł o: jak przy tabl. 1.

Wyniki zaprezentowane w tabl. 3 wskazują, że na podstawie kryterium (19) na docenienie zasługują generatory MB, Ziggurat i Biegun. Najgorzej wypadają generatory Grand, Iloraz i Levy.

Rozkład wartości miary M dla generatorów ma charakter logarytmiczno-normalny. Sytuację tę potwierdza siatka prawdopodobieństwa dla wybranych generatorów, gdzie $t = \ln(M)$ oraz funkcja kwantylowa $y = \Phi^{-1}(F_i) = \Phi^{-1}[(i - 0{,}375)/(n + 0{,}25)]$ (wykr. 4).

WYKR. 4. SIATKA PRAWDOPODOBIEŃSTWA



U w a g a. Jak przy wykr. 1.
Ź r ó d ł o: jak przy tabl. 1.

Kryterium 4: statystyki pozycyjne

Drugą propozycją autora w kwestii testowania jakości generatorów jest zastosowanie statystyk pozycyjnych. Wcześniej już wyznaczono wartość miary (14), która bada zachowanie się dystrybuant empirycznej i teoretycznej. Ciekawym zagadnieniem jest także wyznaczenie wartości miary oznaczonej symbolem MSP, opisującej różnice pomiędzy tymi dystrybuantami dla różnych statystyk pozycyjnych $x_{(i)}$, gdy i przyjmuje wartości początkowe, środkowe i końcowe. Dokonano oceny zarówno numerycznej (tabl. 4), jak i wizualnej (wykr. 5).

Algorytm wyznaczania wartości średniej miary MSP (17) przedstawia się następująco:

1. Wygeneruj tablicę n x u liczb pseudolosowych x_{ij} (i = 1, …, n; j = 1, …, u) z $N(0,1)$
2. Tablicę n x u posortuj kolumnami w kolejności rosnącej
3. Z tablicy n x u wybierz wiersze (statystyki pozycyjne) i = 1,10i (i = 1, …, n/10) i utwórz tablicę (n/10 + 1) x u
4. Transponuj tablicę z pkt 3
5. Tablicę u x (n/10 + 1) posortuj kolumnami w kolejności rosnącej. Kolumny odpowiadają statystykom pozycyjnym i = 1, 10i (i = 1, …, n/10). W kolumnach znajdują się u-elementowe próby statystyk pozycyjnych
6. Oblicz dystrybuanty empiryczne rozkładów statystyk pozycyjnych $F_j = j/(u + 1)$
7. Ustal S = 0 oraz *licznik* = 0. Dla i = 1,10i (i = 1, …, n/10) wykonaj kroki 7.1–7.4:
   7.1. Oblicz dystrybuanty teoretyczne rozkładów statystyk pozycyjnych dla j = 1, …, u za pomocą formuły:

$$\Phi_{j(i)}(x_{ji}) = \sum_{k=i}^{n} \binom{n}{k} \Phi(x_{ji})^{k} \left[1 - \Phi(x_{ji})\right]^{n-k} \tag{15}$$

gdzie $\Phi(.)$ – dystrybuanta teoretyczna (3).

W celu skorzystania z odpowiednio dużych wartości n symbol Newtona z (15) obliczono za pomocą samodzielnie wyprowadzonej formuły powstałej w wyniku przekształceń podstawowej definicji symbolu Newtona. Formuła ta ma postać:

$$\binom{n}{k} = \exp\left[\sum_{i=1}^{n} \ln(i) - \sum_{i=1}^{k} \ln(i) - \sum_{i=1}^{n-k} \ln(i)\right] \tag{16}$$

7.2. Oblicz wartość miary dla $j = 1, \ldots, u$

$$MSP_{(i)} = \sum_{j=1}^{u} \left| F_j - \Phi_{j(i)} \right| \tag{17}$$

7.3. Oblicz $S = S + MSP_{(i)}$
7.4. Zwiększ iterację *licznik* = *licznik* + 1
8. Oblicz wartość średnią miary (17) za pomocą formuły $\overline{MSP} = S/licznik$

Tablica 4 przedstawia wyniki uzyskane za pomocą tego algorytmu dla $n = 10^3$ oraz liczby powtórzeń $u = 10^3$. Pogrubiono najmniejsze wartości. Ze względu na specyfikę wzoru (15) zbyt duże wartości *n* przysparzają problemów numerycznych. Analiza numeryczna wykazała, że maksymalna wielkość próby, z jakiej można skorzystać w (15) dla dowolnej wartości $k \leq n$, to tylko $n = 1029$. Ten fakt nie pozostaje bez wpływu na jakość omawianej metody.

TABL. 4. WARTOŚCI MIARY $\overline{MSP}$ DLA $n = u = 10^3$ WRAZ Z RANKINGIEM

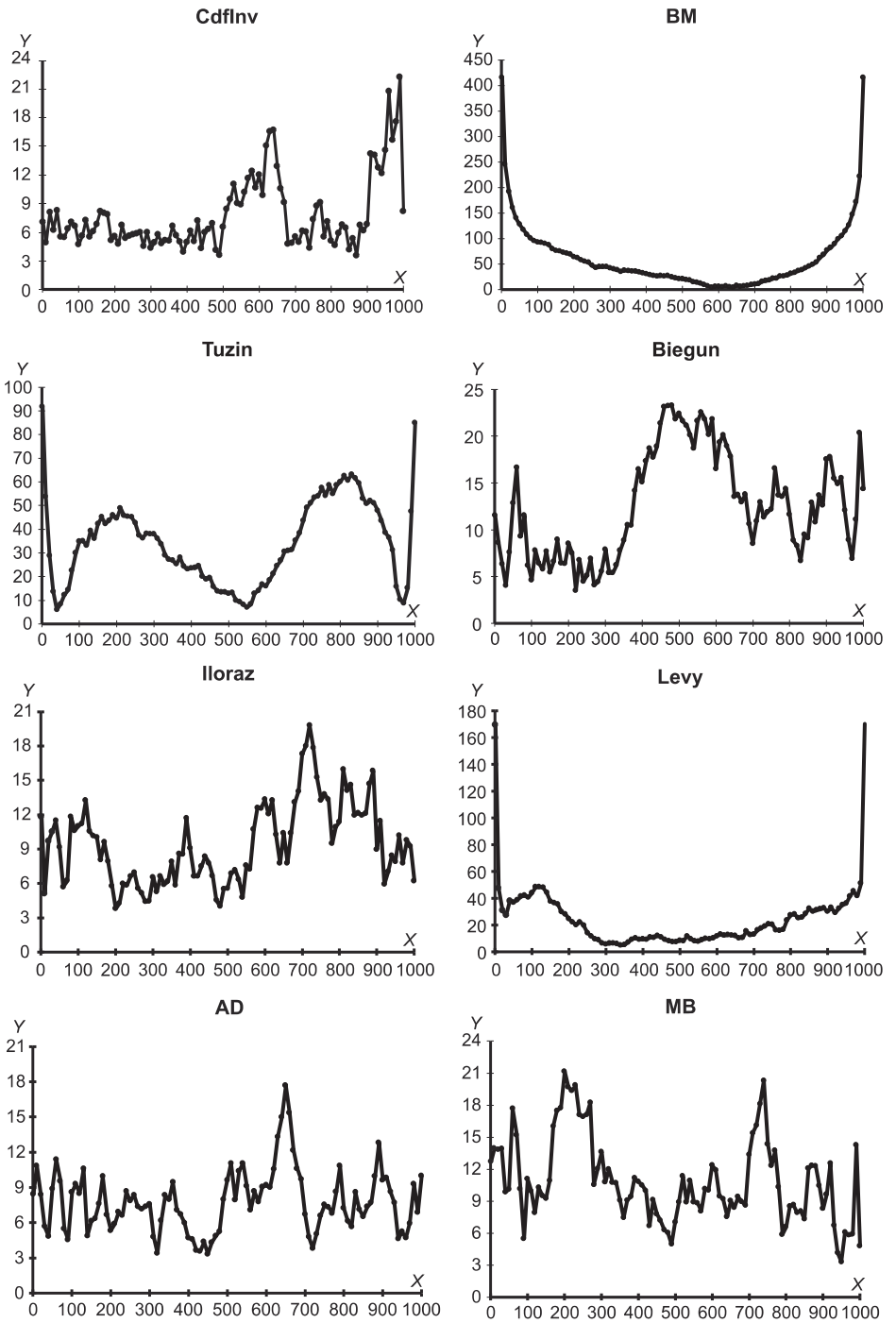
Generatory	$\overline{MSP}$	R
CdfInv	**7,7493**	2
BM	61,1743	14
Tuzin	34,2330	13
Biegun	12,5343	9
Iloraz	9,4371	6
Levy	24,4643	12
AD	**7,7820**	3
MB	10,9223	7
Grand	12,0608	8
MP	9,4006	5
Wallace'a	15,5867	11
Ziggurat	**7,3861**	1
Excel	13,9730	10
R	8,9455	4

Ź r ó d ł o: jak przy tabl. 1.

Z tabl. 4 wynika, że dla statystyk pozycyjnych $i = 1, 10, 20, \ldots, 1000$ średnia wartość miary rozbieżności między dystrybuantami teoretyczną i empiryczną (13) przyjmuje najmniejsze wartości dla generatorów Ziggurat, CdfInv i AD. Według tego kryterium najniższą jakością cechują się generatory BM, Tuzin i Levy.

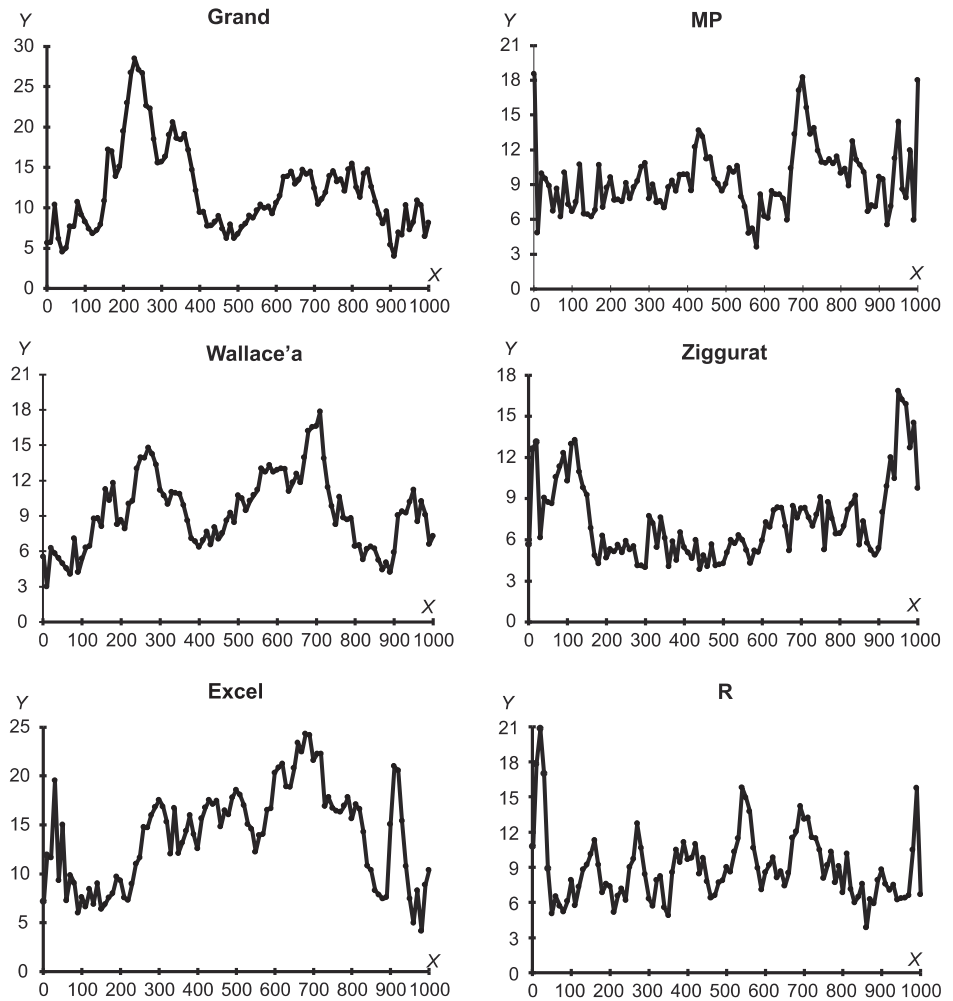
W celu precyzyjniejszej analizy wartości zaprezentowanych w tabl. 4 przedstawiono na wykr. 5 przykładowe zależności między wartościami *i*-tej ($i = 1, 10, 20, \ldots, 1000$) statystyki pozycyjnej a wartościami miary rozbieżności (17), której wartości średnie przedstawia tabl. 4.

WYKR. 5. MIARA ROZBIEŻNOŚCI DYSTRYBUANT DLA STATYSTYK POZYCYJNYCH



oś *Y* – miara rozbieżności dystrybuant, oś *X* – numer statystyki pozycyjnej

WYKR. 5. MIARA ROZBIEŻNOŚCI DYSTRYBUANT DLA STATYSTYK POZYCYJNYCH (dok.)



oś *Y* – miara rozbieżności dystrybuant, oś *X* – numer statystyki pozycyjnej

Ź r ó d ł o: jak przy tabl. 1.

Przebieg miary rozbieżności dystrybuant dla poszczególnych statystyk pozycyjnych okazał się szczególnie nietypowy w przypadku generatorów BM, Tuzin i Levy.

Kryteria 5 i 6: testy trendu i sezonowości

W celu rzetelnej interpretacji wyników zaprezentowanych na wykr. 5 przeprowadzono testy trendu i sezonowości dla każdej *n*-elementowej próby wygenerowanej *u*-krotnie przy poziomie istotności $\alpha = 0{,}05$, zakładając hipotezę H_0: w ciągu

liczb pseudolosowych trend i sezonowość nie występują. Testy trendu i sezonowości zostały omówione np. w: Sulewski (2000). Implementacja komputerowa tych testów dla *n*-elementowej próby znajduje się w pliku trend_sezon.xlsm.

Algorytm wyznaczania średnich wartości *p*-value testów trendu i sezonowości dla generatora jest następujący:

1. Ustal liczebność próby *n*
2. Ustal wartość zmiennych $H_T = 0$, $H_S = 0$
3. Poniższe trzy kroki powtórz *u* razy:
   3.1. Wygeneruj *n* liczb pseudolosowych $x_1, x_2, \ldots, x_n$ z $N(0,1)$
   3.2. Oblicz wartości *p*-value testów trendu i sezonowości odpowiednio pv_T i pv_S
   3.3. Jeżeli nie ma podstaw do odrzucenia H_0, to $H_T = H_T + 1$ (dla trendu) i $H_S = H_S + 1$ (dla sezonowości)
4. Oblicz średnie wartości *p*-value $\overline{pv_T}$ (dla trendu) i $\overline{pv_S}$ (dla sezonowości)
5. Prawdopodobieństwo braku podstaw do odrzucenia H_0 wynosi $p_T = H_T/u$ (dla trendu) i $p_S = H_S/u$ (dla sezonowości).

Tablica 5 przedstawia średnie wartości *p*-value testu trendu, średnie wartości *p*-value testu sezonowości oraz prawdopodobieństwo braku podstaw do odrzucenia H_0. Wartości największe zostały pogrubione.

TABL. 5. TESTY TRENDU I SEZONOWOŚCI DLA $n = u = 10^3$ WRAZ Z RANKINGIEM

Generatory	Test trendu				Test sezonowości			
	pv_T	R	p_T	R	pv_s	R	p_s	R
CdfInv	0,2466	13	0,932	13	0,5265	8	0,839	14
BM	0,2486	10	0,945	9	0,5261	9	0,870	4
Tuzin	**0,2580**	1	0,946	7	**0,5426**	1	**0,872**	2
Biegun	0,2510	5	0,946	7	0,5349	4	0,869	5
Iloraz	0,2481	11	0,942	10	0,5252	10	0,851	13
Levy	0,2543	4	**0,954**	2	**0,5366**	2	0,867	6
AD	0,2509	6	**0,951**	3	0,5116	14	0,854	12
MB	0,2497	9	**0,955**	1	0,5224	12	0,864	8
Grand	0,2474	12	0,929	14	0,5349	5	**0,871**	3
MP	**0,2543**	3	0,951	3	0,5304	7	0,861	9
Wallace'a	0,2509	7	0,940	11	0,5172	13	0,857	11
Ziggurat	0,2464	14	0,949	5	0,5324	6	**0,874**	1
Excel	**0,2545**	2	0,949	5	0,5244	11	0,860	10
R	0,2501	8	0,940	11	**0,5365**	3	0,866	7

Ź r ó d ł o: jak przy tabl. 1.

Z tabl. 5 wynika, że najlepiej z punktu widzenia testów trendu i sezonowości prezentują się generatory Levy, Tuzin i MP. Najniższymi wartościami *p*-value cechują się generatory CdfInv, Iloraz i Wallace'a.

JAK SKORZYSTAĆ Z PLIKÓW INTERNETOWYCH

Pliki niezbędne do przeprowadzenia badań zostały umieszczone pod adresem https://sulewski.apsl.edu.pl/index.php/publikacje. Plik generatory.xlsm zawiera kody źródłowe większości generatorów zapisane w języku VBA wraz z przykładami ich zastosowania. Plik testy.xlsm umożliwia obliczanie wartości *p*-value dla testów LF i A-D. Plik trend_sezon.xlsm pozwala na wyznaczanie wartości *p*-value dla testów trendu i sezonowości. Plik c++.zip zawiera foldery dotyczące omawianych metod generacji liczb pseudolosowych. W każdym z tych folderów znajdują się pliki *.cpp i *.exe, gdzie symbol * oznacza nazwę danego generatora.

Wyniki są zapisywane w formie tablicy dwuwymiarowej w pliku results.txt zamieszczonym w każdym folderze. W związku z tym przy uruchomieniu danego generatora należy podać liczbę kolumn *kol*, liczbę wartości w każdej kolumnie *n* oraz parametry rozkładu normalnego *m* i *s*. Dzięki separatorowi typu tabulator można bardzo łatwo wczytać otrzymane wyniki do arkusza kalkulacyjnego w celu dalszej obróbki statystycznej.

PODSUMOWANIE

Głównym celem pracy jest analiza generatorów liczb pseudolosowych normalnych za pomocą różnego rodzaju kryteriów. Do badań wybrano najważniejsze zdaniem autora generatory, z których każdy zdefiniowano algorytmicznie i oprogramowano w języku C++, a większość także w języku VBA. Oprócz powszechnie stosowanych kryteriów, takich jak charakterystyki rozkładu (kryterium 1), testy zgodności (kryterium 2) czy testy trendu i sezonowości (kryteria 5 i 6), zastosowano także dwa nowe podejścia w postaci miary wykorzystującej sumy modułów różnic dystrybuant empirycznej i teoretycznej (kryterium 3) oraz miary korzystającej ze statystyk pozycyjnych (kryterium 4). Na podstawie uśrednienia rankingów analizowanych generatorów można dokonać ich końcowej klasyfikacji. Wyniki pozwalają stwierdzić, że najwyższą jakością cechują się generatory Monty Pythona (MP), R, Biegun i Ziggurat, podczas gdy najgorsze okazały się generatory Boxa-Mullera (BM), Wallace'a, Iloraz i Excel.

BIBLIOGRAFIA

Abdi, H., Molin, P. (2007). Lilliefors/Van Soest's test of normality. W: N. Salkind (red.), *Encyclopedia of Measurement and Statistics*. Thousand Oaks: Sage.

Ahrens, J. H., Dieter, U. (1972). Computer methods for sampling from the exponential and normal distributions. *Communication of the ACM*, *15*(10), 873–882.

Ahrens, J. H., Dieter, U. (1988). Efficient table-free sampling methods for the exponential, Cauchy, and normal distributions. *Communication of the ACM*, *31*(11), 1330–1337.

Anderson, T. W., Darling, D. A. (1952). Asymptotic theory of certain ”goodness-of-fit” criteria based on stochastic processes. *Annals of Mathematical Statistics*, *23*(2), 193–212. DOI: 10.1214/aoms/1177729437.

Bell, J. R. (1968). Algorithm 334: Normal random deviates. *Communication of the ACM*, *11*(7), 498.

Bloom, G. (1958). *Statistical estimates and transformed Beta variables*. New York: Wiley.

Box, G. E. P., Muller, M. E. (1958). A note on the generation of random normal deviates. *Annals of Mathematical Statistics*, *29*(2), 610–611.

Brent, R. P. (1974). Algorithm 488: A Gaussian pseudo-random number generator. *Communication of the ACM*, *17*(12), 704–706.

Chen, J., Moon, J., Bazargan, K. (2004). Reconfigurable readback-signal generator based on a field *programmable gate array. IEEE Transactions on Magnetics*, *40*(3), 1744–1750.

Chen, W., Burford, R. L. (1981). Quality evaluation of some combinations of unit uniform random number generators and unit normal transformation algorithms. W: R. M. Huhn, E. R. Comer, F. O. Simons Jr. (red.), *ANSS'81: Proceedings of the 14th Annual Symposium on Simulation* (s. 129––149). Tampa: IEEE Press Piscataway.

Cramér, H. (1928). On the Composition of Elementary Errors. *Scandinavian Actuarial Journal*, *1*(1), 13–74.

Cramér, H. (1958). *Metody matematyczne w statystyce*. Warszawa: PWN.

D'Agostino, R. B. (1986). Tests for the Normal Distribution. W: R. B. D'Agostino, M. A. Stephens (red.), *Goodness-of-Fit Techniques* (s. 367–413). New York: Marcel Dekker.

Gebhardt, F. (1964). Generating normally distributed random numbers by inverting the normal distribution function. *Mathematics of Computation*, *18*(86), 302–306.

Kabal, P. (2000). *Generating Gaussian pseudo-random deviates*. McGill University.

Kinderman, A. J., Monahan, J. F. (1977). Computer generation of random variables using the ratio of uniform deviates. *ACM Transactions on Mathematical Software*, *3*(3), 257–260.

Knop, R. (1969). Remark on Algorithm 334 [g5]: normal random deviates. *Communication of the ACM*, *12*(5), 281.

Kolmogorov, A. (1933). Sulla determinazione empirica di una legge di distributione. *Giornale dell' Istituto Italiano degli Attuari*, (4), 83–91.

Kronmal, R. (1964). Evaluation of a pseudorandom normal number generator. *Journal of the ACM, 11*(3), 357–363.

Kuiper, N. H. (1960). Tests concerning random points on a circle. *Proceedings of the Koninklijke Nederlandse Akademie van Wetenschappen Series A*, (63), 38–47.

L'Ecuyer, P. (1992). Testing random number generators. W: J. J. Swain, D. Goldsman, R. C. Crain, J. R. Wilson (red.), *Proceedings of the 24th Conference on Winter Simulation* (s. 305–313). New York: ACM Press.

L'Ecuyer, P. (2001). Software for uniform random number generation: distinguishing the good and the bad. W: *Proceedings of the 33rd Conference on Winter Simulation* (s. 95–105). Washington: IEEE Computer Society.

L'Ecuyer, P., Simard, R. (2005). TestU01. Pobrane z: http://simul.iro.umontreal.ca/.

L'Ecuyer, P., Simard, R. (2007). TestU01: AC library for empirical testing of random number generators. *ACM Transactions on Mathematical Software*, *33*(4), 22.

Leva, J. L. (1992). A fast normal random number generator. *ACM Transactions on Mathematical Software*, *18*(4), 449–453.

Lilliefors, H. W. (1967). On the Kolmogorov-Smirnov Test for Normality with Mean and Variance Unknown. *Journal of the American Statistical Association*, *62*(318), 399–402.

Marsaglia, G. (1997). *The Diehard random number test suite*. Pobrane z: https://github.com/nmondal/diehard.c.

Marsaglia, G., Bray, T. A. (1964). A convenient method for generating normal variables. *SIAM Review*, *6*(3), 260–264.

Marsaglia, G., Tsang, W. W. (1998). The Monty Python Method for Generating Random Variables. *ACM Transactions on Mathematical Software*, *24*(3), 341–350.

Marsaglia, G., Tsang, W. W. (2000). The Ziggurat Method for Generating Random Variables. *Journal of Statistical Software*, *5*(8), 1–7.

Matsumoto, M., Nishimura, T. (1998). Mersenne twister: A 623-dimensionally equidistributed uniform pseudo-random number generator. *ACM Transactions on Modeling and Computer Simulation*, *8*(1), 3–30.

McCollum, J. M., Lancaster, J. M., Bouldin, D. W., Peterson, G. D. (2003). Hardware acceleration of pseudo-random number generation for simulation applications. W: *IEEE Southeastern Symposium on System Theory*. Morgantown: IEEE. DOI: 10.1109/SSST.2003.1194578.

Mélard, G. (2014). On the accuracy of statistical procedures in Microsoft Excel 2010. *Computational Statistics*, *29*(5), 1095–1128.

Molle, J. W. D., Hinich, M. J., Morrice, D. J. (1992). Higher-order cumulant spectral-based statistical tests of pseudo-random variate generators. W: *Proceedings of the 24th Conference on Winter Simulation* (s. 618–625). New York: ACM Press.

Muller, M. E. (1958). An inverse method for the generation of random normal deviates on large-scale computers. *Mathematical Tables and Other Aids to Computation*, *12*(63), 167–174.

Muller, M. E. (1959). A comparison of methods for generating normal deviates on digital computers. *Journal of the ACM*, *6*(3), 376–383.

Ohirko, I., Zaniewski, I., Ogirko, O. (2016). Modelowanie i symulacja w naukach ekonomicznych. *Autobusy: technika, eksploatacja, systemy transportowe*, *17*(6), 1742–1747.

Razali, N. M., Wah, Y. B. (2011). Power comparisons of shapiro-wilk, kolmogorov-smirnov, lilliefors and anderson-darling tests. *Journal of Statistical Modeling and Analytics*, *2*(1), 21–33.

Rényi, A. (1953). On the theory of order statistics. *Acta Mathematica Acadamiae Scientiarum Hungarnicae* (4), 191–232.

Rukhin, A., Soto, J., Nechvatal, J., Smid, M., Barker, E., Leigh, S., Levenson, M., Vangel, M., Banks, D., Heckert, A., Dray, J., Vo, S. (2001). *A statistical test suite for random and pseudorandom number generators for cryptographic applications*. Gaithersburg: National Institute of Standards and Technology (NIST).

Schollmeyer, M. F., Tranter, W. H. (1991). Noise generators for the simulation of digital communication systems. W: *Proceedings of the 24th Annual Symposium on Simulation* (s. 264–275). Los Alamitos: IEEE Computer Society Press.

Shapiro, S. S., Wilk, M. B. (1965). An Analysis of Variance Test for Normality. *Biometrika*, *52*(3/4), 591–611.

Smirnov, N. (1948). Table for estimating the goodness of fit of empirical distributions. *Annals of Mathematical Statistics*, (19), 279–281.

Sulewski, P. (2000). Test trendu i sezonowości. *Wiadomości Statystyczne*, (5), 11–19.

Thomas, D. B., Luk, W., Leong, P. H. W., Villasenor, J. D. (2007). Gaussian random number generators. *Journal ACM Computing Surveys*, *39*(4), 1–38.

Wallace, C. (2005). *Random number generators*. Technical report, Monash University. Pobrano z: http://www.datamining.monash.edu.au/software/random/.

Watson, G. S. (1961). Goodness-of-Fit Tests on a Circle. *Biometrika*, *48*(1/2), 109–114.

Wetherill, G. B. (1965). An approximation to the inverse normal function suitable for the generation of random normal deviates on electronic computers. *Journal of the Royal Statistical Society Series C*, *14*(2–3), 201–205.

Wichura, M. J. (1988). Algorithm AS 241: The Percentage Points of the Normal Distribution. *Journal of the Royal Statistical Society. Series C*, *37*(3), 477–484.

Wieczorkowski, R., Zieliński, R. (1997). *Komputerowe generatory liczb losowych*. Warszawa: WNT.