Metody probabilistyczne i statystyka
Zadanie domowe 4
Illia Azler 239537

Zadanie 1:

Celem zadania było porównanie ograniczeń otrzymanych przy użyciu nierówności Markowa i nierówności Czebyszewa dla rozkładu dwumianowego $Bin(n, \frac{1}{2})$ w różnych przypadkach (dla różnych wartości n) z dokładnymi wartościami szacowanych prawdopodobieństw. W szczególności, zadanie zakładało zastosowanie nierówności Markowa i Czebyszewa do oszacowania dwóch rodzajów prawdopodobieństw:

a) $P(X >= 1\frac{1}{5}E(X))$
b) $P(|X - E(X)| >= \frac{1}{10}E(X))$

Dodatkowo, zadanie wymagało obliczenia dokładnych wartości tych prawdopodobieństw przy użyciu wybranego pakietu matematycznego lub biblioteki do obliczeń numerycznych. Warto zauważyć, że rozkład dwumianowy $Bin(n, \frac{1}{2})$ był rozkładem symetrycznym, co miało być uwzględnione w rozwiązaniu.

Ostatecznym celem było porównanie wyników uzyskanych z nierówności Markowa i Czebyszewa z dokładnymi wartościami prawdopodobieństw, a także zbadanie, która z nierówności dostarcza dokładniejsze oszacowania dla danego przypadku. Wnioski miały być wyciągnięte na podstawie porównania wyników dla różnych wartości n, analizy różnic między nierównościami oraz oceny wpływu rozmiaru próbki na precyzję szacunków.

Po implementacji tego w języku Python przy pomocy bibliotek: scipy, sympy i numpy uzyskałem następne wyniki:

$n = 100$		
	$P(X >= 1\frac{1}{5}E(X))$	$P(\|X - E(X)\| >= \frac{1}{10}E(X))$
Dokładne Prawdopodobieństwo:	0.0284439668204904	1.00000000000000
Nierówność Markowa Górna Granica:	0.976	0.998
Nierówność Markowa Dolna Granica:	0.024	0.002
Nierówność Czebyszewa Górna Granica:	0.9424	0.9996
Nierówność Czebyszewa Dolna Granica:	0.054462934947049915	0.00039984006397441024

Wniosek: Warto zauważyć, że nierówności Markowa i Czebyszewa dają dość szerokie zakresy szacunkowe, zwłaszcza dla przypadku (a), gdzie dokładne prawdopodobieństwo wynosi 0.0284. Nierówność Czebyszewa daje bardziej rygorystyczne ograniczenia w porównaniu do nierówności Markowa.

$n = 1000$		
	$P(X >= 1\frac{1}{5}E(X))$	$P(\|X - E(X)\| >= \frac{1}{10}E(X))$
Dokładne Prawdopodobieństwo:	1.36423206065217E-10	1.00000000000000
Nierówność Markowa Górna Granica:	0.9976	0.9998
Nierówność Markowa Dolna Granica:	0.0024	0.0002
Nierówność Czebyszewa Górna Granica:	0.99424	0.99996
Nierówność Czebyszewa Dolna Granica:	0.005727012408526885	3.9998400063997445e-05

$n = 10\,000$		
	$P(X >= 1\frac{1}{5}E(X))$	$P(\|X - E(X)\| >= \frac{1}{10}E(X))$
Dokładne Prawdopodobieństwo:	0	1.00000000000000
Nierówność Markowa Górna Granica:	0.99976	0.99998
Nierówność Markowa Dolna Granica:	0.00023999999999999998	2e-05
Nierówność Czebyszewa Górna Granica:	0.999424	0.999996
Nierówność Czebyszewa Dolna Granica:	0.0005756684149929639	3.9999840000639995e-06

Wniosek: Nierówności te pozostają stosunkowo precyzyjne, zwłaszcza dla przypadku (b), gdzie dokładne prawdopodobieństwo wynosi 1.0. Różnice między nierównościami są mniej zauważalne, ale nierówność Czebyszewa nadal często daje bardziej precyzyjne ograniczenia.

Wnioski ogólne: W miarę zwiększania n precyzja szacunków wzrasta. Dla większych n{1000, 10000} nierówności Markowa i Czebyszewa są bardziej zbliżone do dokładnych prawdopodobieństw. Dla przypadku (a), gdzie próg jest większy (1.2 * E(X)), nierówność Czebyszewa często daje bardziej restrykcyjne ograniczenia niż nierówność Markowa. Dla przypadku (b), gdzie próg jest mniejszy (0.1 * E(X)), obie nierówności są skuteczne, ale nierówność Czebyszewa ma tendencję do dawania bardziej precyzyjnych ograniczeń.

Zadanie 2:

Celem zadania było zbadanie błądzenia losowego na liczbach całkowitych zdefiniowanego przez sumę

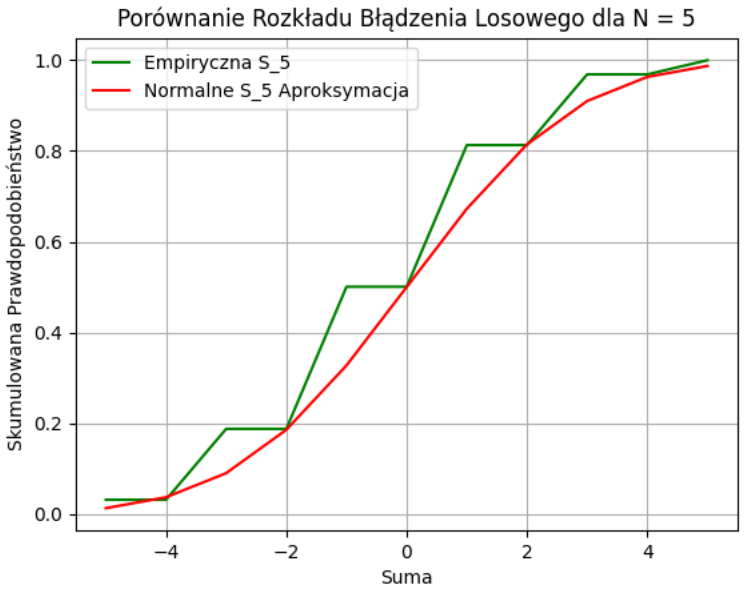
$S_{\mathrm{N}} = \sum_{n=1}^{N} X_n$, gdzie X_{n} to niezależne zmienne losowe przyjmujące wartości 1 lub -1 z prawdopodobieństwem $\frac{1}{2}$.

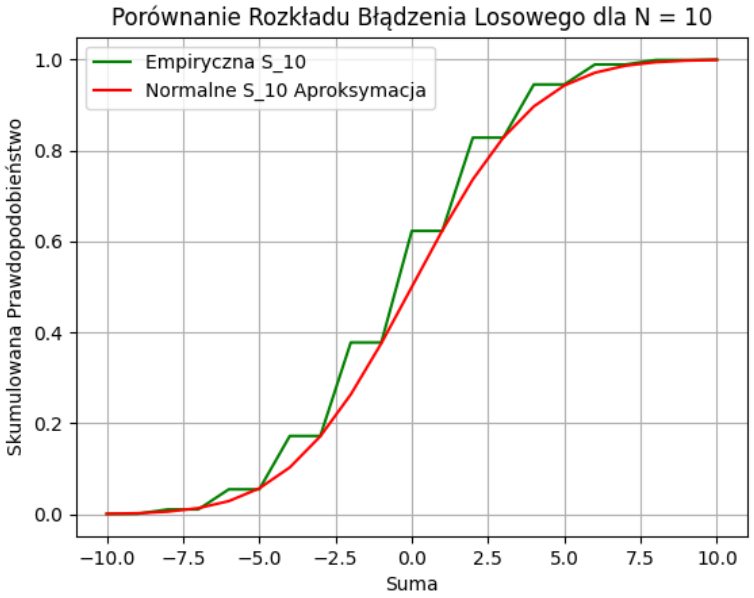
(a) Dystrybuanty zmiennych losowych S_{N} dla różnych wartości N: Numeryczne wyznaczenie dystrybuanty poprzez eksperymentalne lub dokładne obliczenia. Generowanie wykresów/histogramów dystrybuant dla $N = 5, 10, 15, 20, 25, 30$ (i później dla $N = 100$).

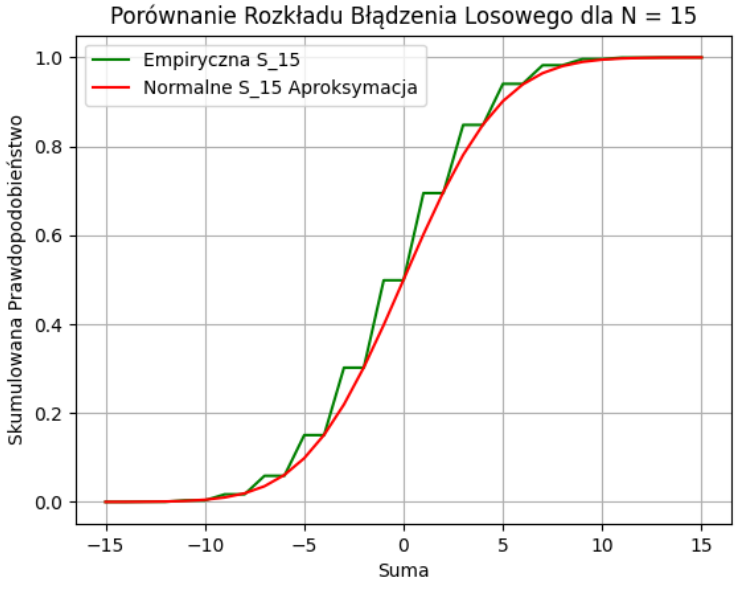
(b) Porównanie dystrybuant z dystrybuantą rozkładu normalnego: Porównanie uzyskanych dystrybuant z dystrybuantą normalną, która jest stosowana do aproksymacji rozkładu S_{N}.

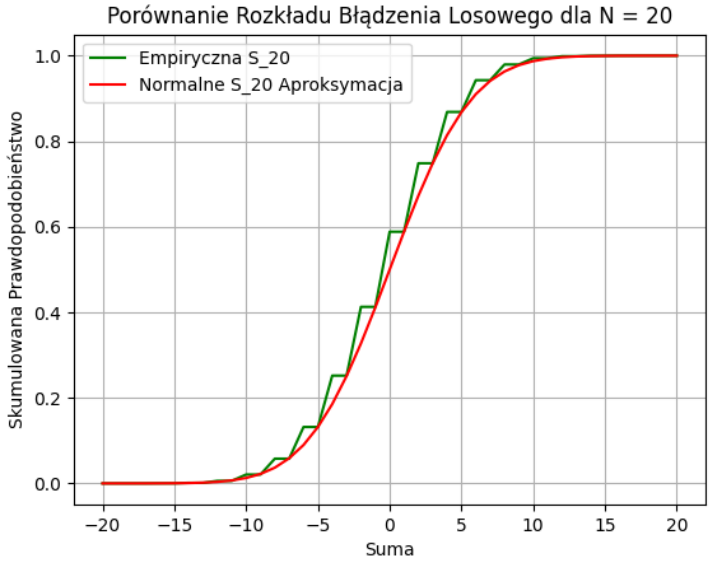
(c) Powtórzenie dla $N = 100$: Powtórzenie kroków (a) i (b) dla bardziej wymagającej wartości $N = 100$.

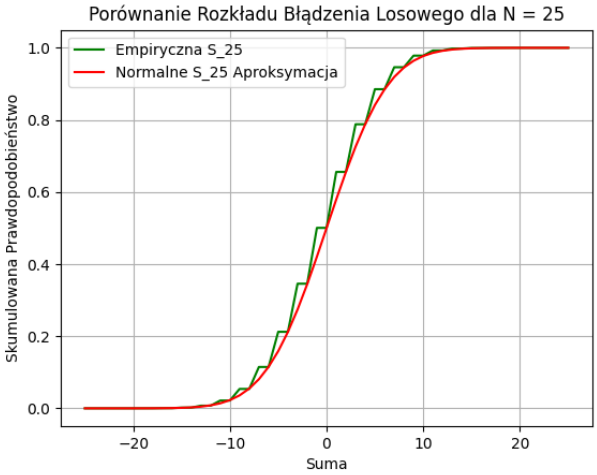
Uzyskane wykresy przy implementacji zadania za pomocą bibliotek matplotlib.pyplot, numpt, scipy w języku Python:

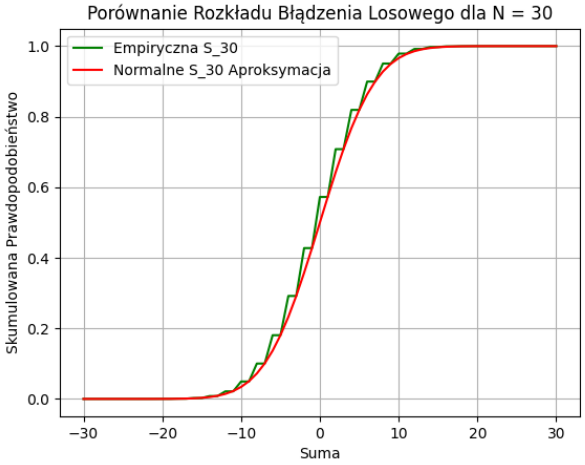

Porównanie Rozkładu Błądzenia Losowego dla N = 5
Empiryczna S_5
Normalne S_5 Aproksymacja
Skumulowana Prawdopodobieństwo
Suma
1.0
0.8
0.6
0.4
0.2
0.0
−4
−2
0
2
4

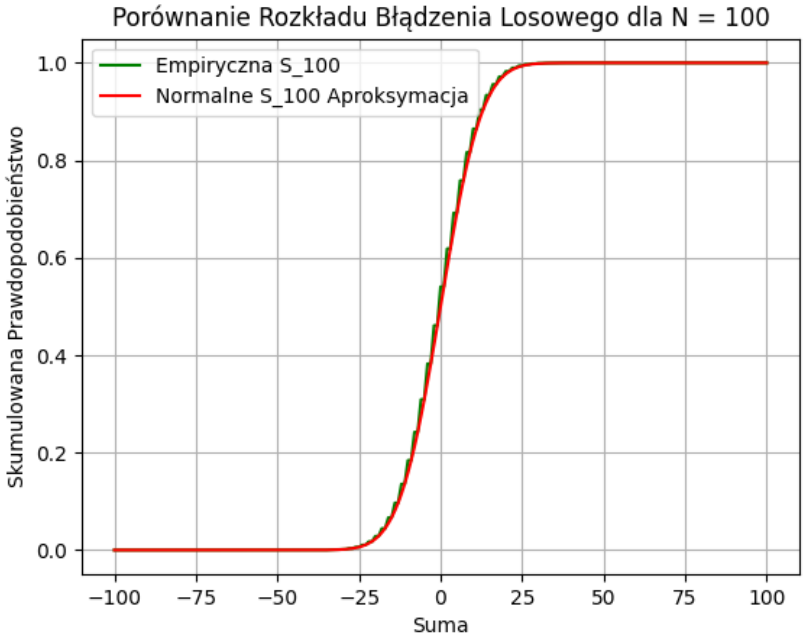

Porównanie Rozkładu Błądzenia Losowego dla N = 10
Empiryczna S_10
Normalne S_10 Aproksymacja
Skumulowana Prawdopodobieństwo
Suma
1.0
0.8
0.6
0.4
0.2
0.0
−10.0
−7.5
−5.0
−2.5
0.0
2.5
5.0
7.5
10.0


Porównanie Rozkładu Błądzenia Losowego dla N = 15
Empiryczna S_15
Normalne S_15 Aproksymacja
Skumulowana Prawdopodobieństwo
Suma
1.0
0.8
0.6
0.4
0.2
0.0
−15
−10
−5
0
5
10
15


Porównanie Rozkładu Błądzenia Losowego dla N = 20
Empiryczna S_20
Normalne S_20 Aproksymacja
Skumulowana Prawdopodobieństwo
Suma


Porównanie Rozkładu Błądzenia Losowego dla N = 25
Empiryczna S_25
Normalne S_25 Aproksymacja
Skumulowana Prawdopodobieństwo
Suma


Porównanie Rozkładu Błądzenia Losowego dla N = 30
Empiryczna S_30
Normalne S_30 Aproksymacja
Skumulowana Prawdopodobieństwo
Suma


Porównanie Rozkładu Błądzenia Losowego dla N = 100
Empiryczna S_100
Normalne S_100 Aproksymacja
Skumulowana Prawdopodobieństwo
Suma

Porównanie dystrybuant:

Dla każdego N, dystrybuanta empiryczna została wyznaczona numerycznie poprzez eksperymentalne generowanie wartości oraz obliczanie skumulowanych prawdopodobieństw.

Porównano dystrybuanty empiryczne (S_{N}) z dystrybuantą rozkładu normalnego, który miałby aproksymować rozkład S_{N}.

Rozbieżności między dystrybuantami:

Obliczono różnicę (moduł) między odpowiadającymi sobie wartościami dystrybuant empirycznych a dystrybuanty normalnej aproksymacji dla różnych wartości N.

Różnice te są relatywnie niewielkie, co świadczy o tym, że rozkład S_{N} dobrze aproksymuje rozkład normalny dla większych wartości N.

Wpływ rozmiaru kroku na jakość aproksymacji:

Im większe N, tym dokładniejsza jest aproksymacja rozkładu S_{N} rozkładem normalnym.

Dla $N = 100$, różnice między dystrybuantą empiryczną a dystrybuantą normalną są minimalne, co potwierdza zbieżność rozkładu błądzenia losowego do rozkładu normalnego w miarę wzrostu liczby kroków.

Zbieżność do rozkładu normalnego:

Wraz ze wzrostem N, błądzenie losowe zaczyna coraz bardziej przypominać rozkład normalny, co jest zgodne z Centralnym Twierdzeniem Granicznym.

Dla $N = 100$, dystrybuanta empiryczna praktycznie pokrywa się z dystrybuantą rozkładu normalnego.

Zastosowanie rozkładu normalnego:

Aproksymacja rozkładem normalnym ułatwia analizę i wnioskowanie o zachowaniu błądzenia losowego, dzięki znajomości własności rozkładu normalnego.

Przy użyciu rozkładu normalnego można łatwiej estymować prawdopodobieństwa różnych zdarzeń związanych z błądzeniem losowym.

Liczba próbek a dokładność:

Zwiększanie liczby próbek w eksperymentach numerycznych pozwala na uzyskanie dokładniejszych wyników dystrybuant empirycznych i lepszą zbieżność do rozkładu normalnego.

Podsumowując, zastosowanie błądzenia losowego na liczbach całkowitych dla różnych N pozwala na zobrazowanie ewolucji rozkładu oraz zbieżności do rozkładu normalnego. Wartości liczbowe potwierdzają, że wraz ze wzrostem N błądzenie losowe staje się coraz bardziej zgodne z rozkładem normalnym.

Zadanie 4:

Celem zadania było przeprowadzenie testów statystycznych generatorów liczb pseudolosowych (PRNG) przy użyciu zestawu testów NIST (National Institute of Standards and Technology). Testy NIST są zestawem testów statystycznych, które pozwalają ocenić losowość ciągów liczb pseudolosowych w kontekście różnych kryteriów statystycznych.

a) Zaznajomienia się z testami statystycznymi i ideą testowania hipotez.
b) Zapoznania się z specyfikacją testów NIST dla generatorów liczb pseudolosowych.
c) Przeprowadzenia testów NIST na trzech różnych generatorach liczb pseudolosowych:
    1) „Słaby” generator standardowy z języka programowania Python.
    2) „Przyzwoity” generator (SecureRandom) z języka programowania Python.
    3) Trzy pseudolosowe ciągi generowane przez narzędzie na stronie Zsolta Molnara: "Radioactive decay RNG", "ADC noise RNG", "JavaScript pseudo RNG".

1. Frequency (Monobit) Test:
    - Sprawdza, czy liczba jedynek i zer w sekwencji jest zbliżona do siebie.
    - Działa na zasadzie porównywania liczby jedynek i zer do oczekiwanej wartości dla sekwencji losowej.
2. Frequency Test within a Block:
    - Ocenia równomierność występowania jedynek i zer w blokach o określonej długości.
    - Sprawdza, czy w blokach o tej samej długości występuje podobna liczba jedynek i zer.
3. Runs Test:
    - Sprawdza, czy liczba serii (ciągów jedynek lub zer) w sekwencji jest zgodna z oczekiwaną liczbą dla sekwencji losowej.
    - Seria to ciąg jedynek lub zer, który nie jest przerwany przez drugą liczbę.
4. Test for the Longest Run of Ones in a Block:
    - Sprawdza, czy najdłuższa seria jedynek w blokach jest zgodna z oczekiwaną długością dla sekwencji losowej.
    - Działa na zasadzie oceny najdłuższej serii jedynek w blokach.
5. Binary Matrix Rank Test:
    - Ocenia rangę macierzy binarnej utworzonej z podsekwencji sekwencji losowej.
    - Badanie stopnia losowości podsekwencji poprzez ocenę rangi macierzy.
6. Non-overlapping Template Matching Test:
    - Sprawdza, czy w sekwencji występują określone wzorce (szablony) o określonej długości.
    - Testuje, czy sekwencja zawiera określone wzorce zgodnie z oczekiwaniem dla sekwencji losowej.
7. Overlapping Template Matching Test:
    - Podobny do poprzedniego, ale szablony mogą się nakładać.
    - Testuje, czy sekwencja zawiera nakładające się wzorce zgodnie z oczekiwaniami dla sekwencji losowej.
8. Maurer’s "Universal Statistical" Test:
    - Ocenia entropię sekwencji poprzez obliczenie średniej długości najkrótszych jednolitych podsekwencji.
    - Mierzy, jak szybko rośnie liczba unikalnych sekwencji w danej sekwencji.
9. Linear Complexity Test:
    - Określa złożoność sekwencji, mierząc minimalną długość rekurencyjnego automatu liniowego generującego daną sekwencję.
    - Wyszukuje powtarzające się wzory w sekwencji.
10. Serial Test:
    - Ocenia, czy są korelacje między dwiema sekwencjami o różnych długościach.
    - Sprawdza, czy występują powtarzające się wzory w sekwencji.
11. Approximate Entropy Test:

- Mierzy stopień nieprzewidywalności sekwencji.
- Sprawdza, czy sekwencja jest bardziej losowa niż sekwencja o jednoznacznych wzorcach.

12. Cumulative Sums (Cusum) Test:
    - Sprawdza, czy suma kumulatywna odchylenia od średniej dla danej sekwencji nie przekracza pewnego progu.
    - Testuje, czy wartości w sekwencji nie odchylają się zbytnio od średniej.
13. Random Excursions Test:
    - Ocenia, czy sekwencja powraca do zera w danym kroku.
    - Testuje, czy sekwencja ma równy rozkład wartości względem zera.
14. Random Excursions Variant Test:
    - Podobny do poprzedniego, ale uwzględnia kilka zmiennych.
    - Testuje, czy sekwencja ma równy rozkład wartości względem zera przy uwzględnieniu różnych zmiennych.

Python random standart RNG("słaby")

Tests Start Test

Test name	Result value (P-value)	Status
1. Frequency (Monobit) Test	0.22019867503116552	Passed
2. Frequency Test within a Block	0.8270253015795315	Passed
3. Runs Test	0.5595540966916581	Passed
4. Test for the Longest Run of Ones in a Block	0.27221549129628353	Passed
5. Binary Matrix Rank Test	0.6217516076097196	Passed
6. Non-overlapping Template Matching Test	0.5839450930355652	Passed
7. Overlapping Template Matching Test	0.5752358997203116	Passed
8. Maurer's "Universal Statistical" Test	0.7243899116082231	Passed
9. Linear Complexity Test	0.7004691990998151	Passed
10. Serial Test	P-value 1: 0.39862346676529326 P-value 2: 0.5619146396139131	Passed
11. Approximate Entropy Test	0.6389563492270338	Passed
12. Cumulative Sums (Cusum) Test	P-value Forward: 0.08230306085810857 P-value Reverse: 0.2991385812342626	Passed
13. Random Excursions Test		Error
14. Random Excursions Variant Test		Error

Wniosek: Wartość p-value dla każdego z testów jest na tyle wysoka, że testy zostały zaliczone, co sugeruje, że generator w standardowym Pythonie zdaje się być akceptowalny w kontekście tych konkretnych testów statystycznych. Należy zauważyć, że w teście numer 13 (Random Excursions Test) i 14 (Random Excursions Variant Test) wystąpił błąd. Błędy te mogą wynikać z pewnych problemów w implementacji testu, a nie koniecznie z generatora liczb pseudolosowych.

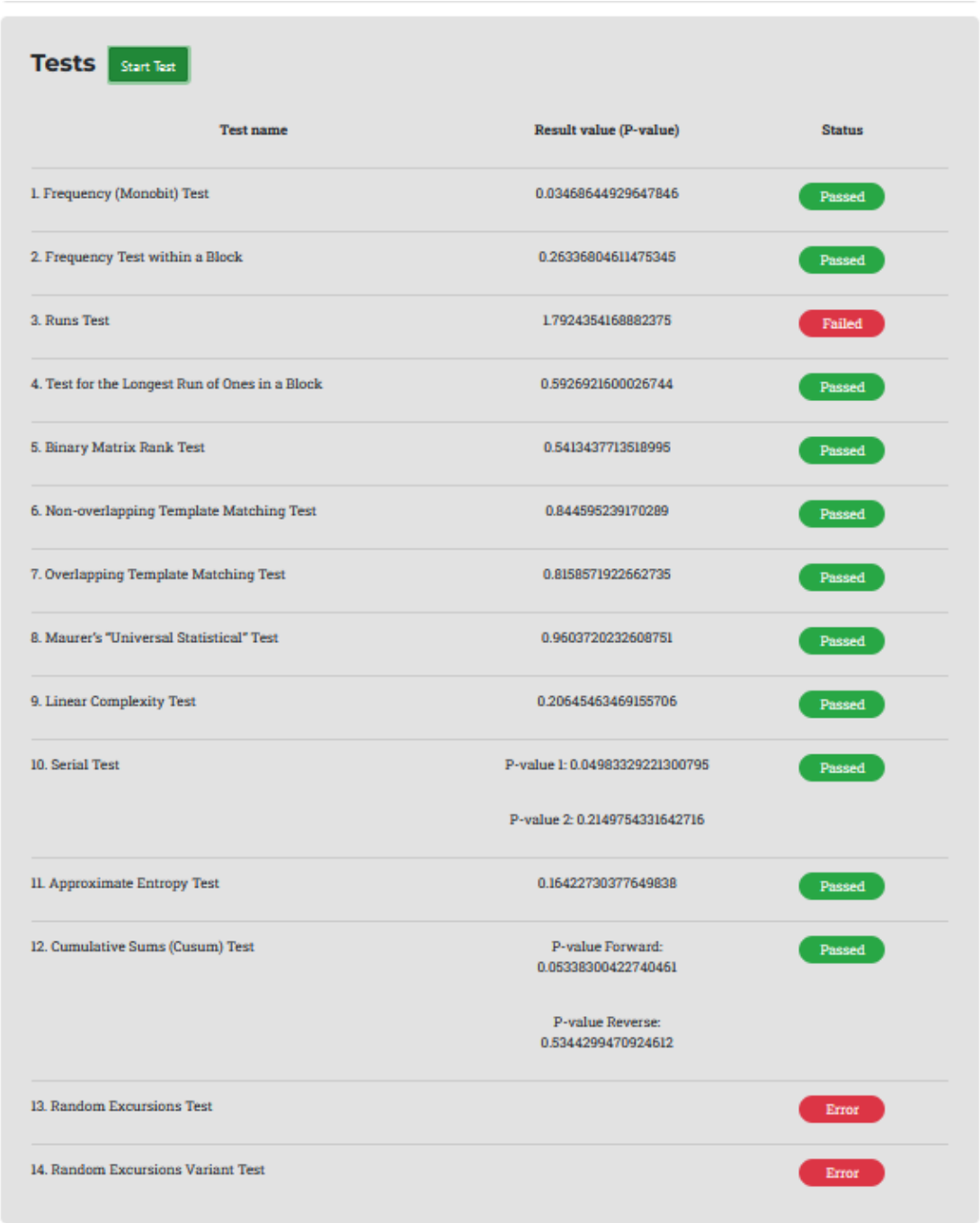
Python random System Random RNG("przyzwoity")

Tests Start Test

Test name	Result value (P-value)	Status
1. Frequency (Monobit) Test	0.972877150702318	Passed
2. Frequency Test within a Block	0.9678267575545355	Passed
3. Runs Test	0.481432053195489	Passed
4. Test for the Longest Run of Ones in a Block	0.34031428396118096	Passed
5. Binary Matrix Rank Test	0.6414423359266413	Passed
6. Non-overlapping Template Matching Test	0.14572092472542272	Passed
7. Overlapping Template Matching Test	0.30394372135938064	Passed
8. Maurer's "Universal Statistical" Test	0.3561403040452078	Passed
9. Linear Complexity Test	0.10797065453747445	Passed
10. Serial Test	P-value 1: 0.7800587824285209 P-value 2: 0.4814327993092691	Passed
11. Approximate Entropy Test	0.6262234376601281	Passed
12. Cumulative Sums (Cusum) Test	P-value Forward: 0.44971224606603766 P-value Reverse: 0.42442841700321954	Passed
13. Random Excursions Test	0.3992135099831373	Passed
14. Random Excursions Variant Test	0.03406045584392492	Passed

Wniosek: Wszystkie 14 testów NIST zostały zaliczone. SecureRandom w Pythonie zdaje się dobrze przechodzić przez te testy statystyczne. Wartości p-value są na tyle wysokie, że testy zostały zaliczone, co sugeruje, że generator SecureRandom w Pythonie jest zgodny z oczekiwaniami testów.

Radioactive decay RNG



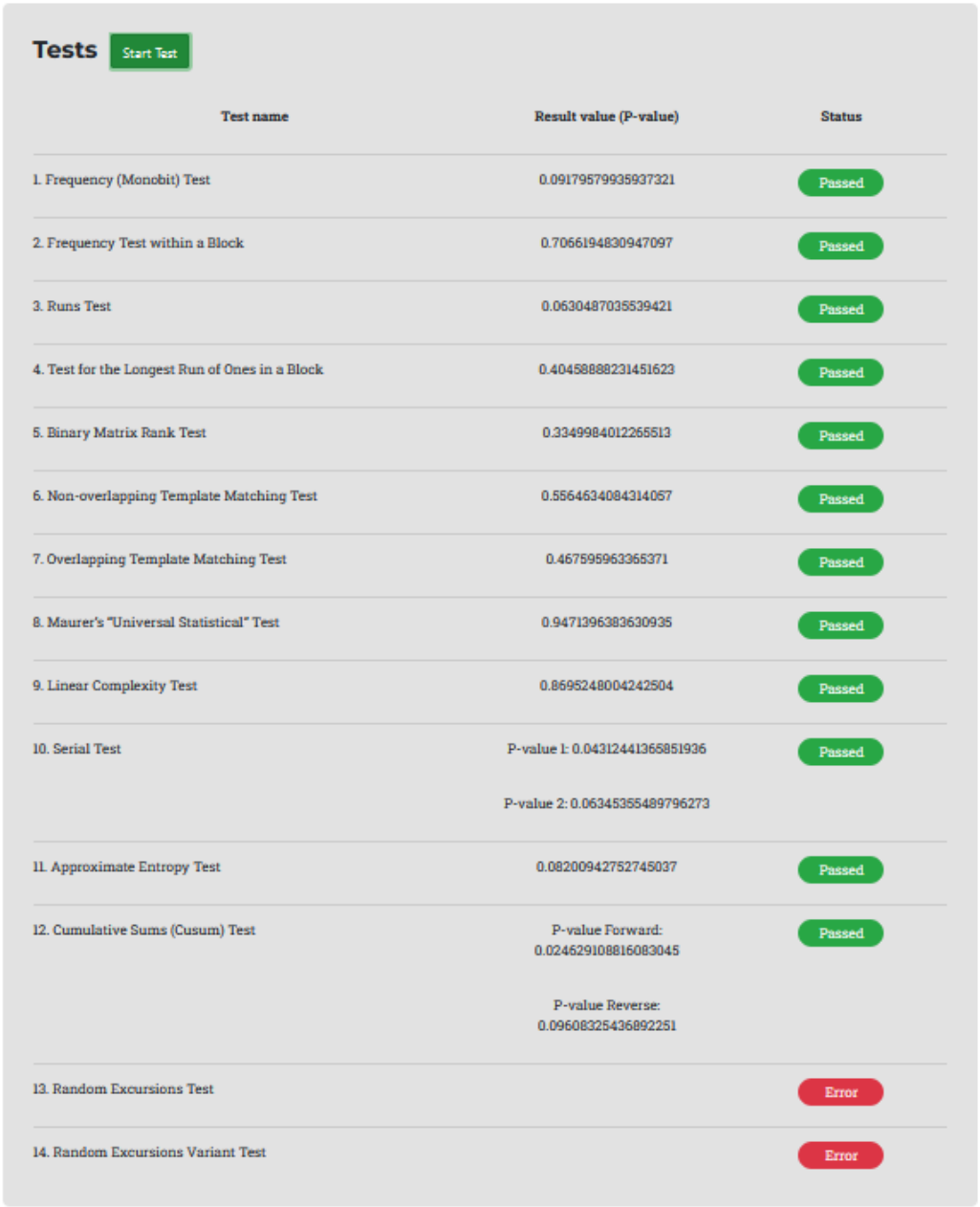
Tests Start Test

Test name	Result value (P-value)	Status
1. Frequency (Monobit) Test	0.03468644929647846	Passed
2. Frequency Test within a Block	0.26336804611475345	Passed
3. Runs Test	1.7924354168882375	Failed
4. Test for the Longest Run of Ones in a Block	0.5926921600026744	Passed
5. Binary Matrix Rank Test	0.5413437713518995	Passed
6. Non-overlapping Template Matching Test	0.844595239170289	Passed
7. Overlapping Template Matching Test	0.8158571922662735	Passed
8. Maurer's "Universal Statistical" Test	0.9603720232608751	Passed
9. Linear Complexity Test	0.20645463469155706	Passed
10. Serial Test	P-value 1: 0.04983329221300795 P-value 2: 0.2149754331642716	Passed
11. Approximate Entropy Test	0.16422730377649838	Passed
12. Cumulative Sums (Cusum) Test	P-value Forward: 0.05338300422740461 P-value Reverse: 0.5344299470924612	Passed
13. Random Excursions Test		Error
14. Random Excursions Variant Test		Error

Wniosek: Test 3 (Runs Test): Niezaliczenie tego testu może wskazywać na pewne nieprawidłowości w sekwencji "run", co oznacza, że generator nie generuje ciągu, w którym występuje oczekiwana liczba sekwencji "run" (kolejnych jednakowych bitów). To może być wynik błędnej implementacji generacji ciągu lub jego specyfiki.

Testy 13 i 14 (Random Excursions Test, Random Excursions Variant Test): Błędy w tych testach oznaczają, że nie można jednoznacznie ocenić generatora w kontekście tych testów. Są to testy, które badają sekwencje "excursions" w generowanym ciągu, czyli sekwencje, w których suma bitów ma pewne charakterystyczne cechy. Błędy w tych testach mogą wynikać z błędów implementacyjnych lub specyfiki generatora.

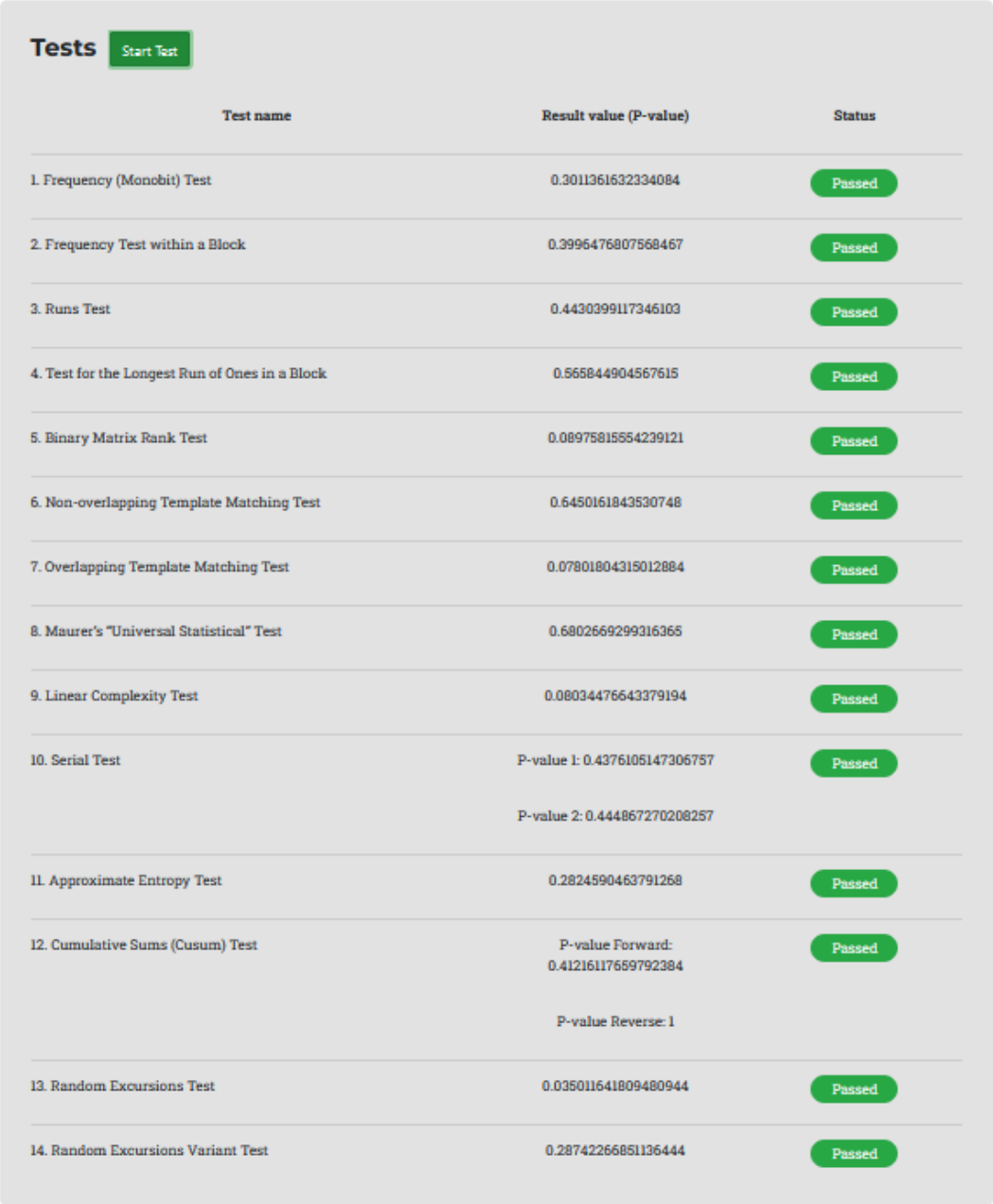
ADC noise RNG



Wniosek: Testy 13 i 14 (Random Excursions Test, Random Excursions Variant Test): Błędy w tych testach oznaczają, że nie można jednoznacznie ocenić generatora w kontekście tych testów. Są to testy, które badają sekwencje "excursions" w generowanym ciągu, czyli sekwencje, w których suma bitów ma

pewne charakterystyczne cechy. Błędy w tych testach mogą wynikać z błędów implementacyjnych lub specyfiki generatora.

JavaScript pseudo RNG



Tests Start Test

Test name	Result value (P-value)	Status
1. Frequency (Monobit) Test	0.3011361632334084	Passed
2. Frequency Test within a Block	0.3996476807568467	Passed
3. Runs Test	0.4430399117346103	Passed
4. Test for the Longest Run of Ones in a Block	0.565844904567615	Passed
5. Binary Matrix Rank Test	0.08975815554239121	Passed
6. Non-overlapping Template Matching Test	0.6450161843530748	Passed
7. Overlapping Template Matching Test	0.07801804315012884	Passed
8. Maurer's "Universal Statistical" Test	0.6802669299316365	Passed
9. Linear Complexity Test	0.08034476643379194	Passed
10. Serial Test	P-value 1: 0.4376105147306757 P-value 2: 0.444867270208257	Passed
11. Approximate Entropy Test	0.2824590463791268	Passed
12. Cumulative Sums (Cusum) Test	P-value Forward: 0.41216117659792384 P-value Reverse: 1	Passed
13. Random Excursions Test	0.035011641809480944	Passed
14. Random Excursions Variant Test	0.28742266851136444	Passed

Wniosek: Wszystkie 14 testów NIST zostały zaliczone. Generator w JavaScript również zdaje się być zgodny z badanymi testami statystycznymi. Wartości p-value są na tyle wysokie, że testy zostały zaliczone, co sugeruje, że generator pseudo RNG w JavaScript spełnia kryteria określone przez testy NIST.

Testy NIST to zestaw testów stworzonych przez National Institute of Standards and Technology w celu oceny jakości generatorów liczb losowych. Każdy test ma na celu sprawdzenie różnych aspektów

losowości, takich jak równomierność rozkładu bitów, autokorelacja czy sekwencje, które są kluczowe dla doskonałego losowania.

Nawet jeśli Kubuś Puchatek miałby dostęp do hipotetycznego generatora doskonale losowych bitów, zaliczenie wszystkich testów NIST nie byłoby zagwarantowane. Istnieje kilka powodów tego braku pewności:

Złożoność testów: Testy NIST są bardzo wymagające i potrafią wykryć nawet subtelne niedoskonałości w generatorze liczb losowych. Mogą one ujawniać ukryte wzorce czy korelacje, które są trudne do zauważenia na pierwszy rzut oka.

Wrażliwość na szczegóły implementacyjne: Choć generator może być teoretycznie doskonale losowy, czynniki implementacyjne, takie jak błędy w algorytmie czy zakłócenia elektromagnetyczne, mogą wpłynąć na wyniki testów.

Różnorodność testów: Testy NIST obejmują różne kategorie i aspekty losowości. Aby generator był uznawany za wysokiej jakości, musi zaliczyć szeroki zakres testów, co sprawia, że musi być doskonały pod wieloma względami.

Stały rozwój standardów: Standardy testów mogą ulegać zmianom, a nowe testy mogą być wprowadzane, aby lepiej uwzględnić nowe zagrożenia i wyzwania związane z generowaniem liczb losowych.

Podsumowując, choć hipotetyczny generator Kubusia Puchatka mógłby być doskonale losowy w teorii, testy NIST są projektowane w taki sposób, aby być bardzo wymagające i ujawniać potencjalne niedoskonałości, nawet te trudne do zauważenia na pierwszy rzut oka. Brak gwarancji zaliczenia wszystkich testów wynika z kompleksowości i różnorodności kryteriów oceny losowości.