Assignment No.1 : testing random number generators

Reza Pourkhodabakhshi[1,2,*]

[1]*Departament de Física Quàntica i Astrofísica, Facultat de Física, Universitat de Barcelona, Martí i Franquès 1, 08028 Barcelona, Spain*
[2]*Institut de Ciencies del Cosmos (ICCUB), Universitat de Barcelona, Martí i Franquès 1, 08028 Barcelona, Spain*

Contents

*Electronic address: reza.pk.bakhshi@gmail.com

I. THE CHOICE OF RANDOM NUMBER GENERATORS

Throughout this report we want to investigate two different random number generators, by using RNG test and comparing them together.

- The Report & Discussions on the results pages 1-2.
- The raw results of the tests on Page 3.
- Appendix.1 on the interpretation of results pages 4-5.
- Appendix.2 a review on the tests performed pages 5-6.
- The Full comprehensive data results in PDF (raw Data Results-Reza).
- The Original performed Test in Notebook (PS1-RNG Test-Reza)
- The source code of the Gamma Distribution, performing the tests, and the documentation of the tests found in Notebook (Tests Code- Reza)

A. Quick and Dirty - Modified RNG Algorithm Parameters

B. Mersenne Twister Algorithm

Almost all module functions depend on the basic function random(), which generates a random float uniformly in the semi-open range [0.0, 1.0). Python uses the Mersenne Twister as the core generator. It produces 53-bit precision floats and has a period of 2**19937-1. The underlying implementation in C is both fast and threadsafe. The Mersenne Twister is one of the most extensively tested random number generators in existence. However, being completely deterministic, it is not suitable for all purposes, and is completely unsuitable for cryptographic purposes.

The Mersenne Twister is a general-purpose pseudorandom number generator (PRNG) developed in 1997 and it is in Python via accessing **random.** .

The most commonly used version of the Mersenne Twister algorithm is based on the Mersenne prime $2^{19937} - 12^{19937} - 1$. The standard implementation of that, MT19937, uses a 32-bit word length.

Yet, multiple instances that differ only in seed value (but not other parameters) are not generally appropriate for Monte-Carlo simulations that require independent random number generators, though there exists a method for choosing multiple sets of parameter values.

Is not cryptographically secure. The reason is that observing a sufficient number of iterations (624 in the case of MT19937, since this is the size of the state vector from which future iterations are produced) allows one to predict all future iterations.

C. RNG - Modified Algorithm Parameters : Period Cycle

Cycle lengths in random maps A pseudo random number generator is based on the sequence

$$s_n = f(s_{n-1}), \quad s \in S$$

where the state transition function f maps the finite set S into itself. The number of possible states is the cardinality of S :

$$m = |S|$$

Let F denote the set of all possible state transition functions:

$$\mathrm{F} = \{f : S \to S\}$$

Assume that we have picked a state transition function f at random from the m^m members of F. We now want to make sure that f can produce a sequence of $c_{\min}$ random numbers from a given seed s_0 without getting cyclic. Let v be the length of the limiting cycle, and μ the length of any transient aperiodic sequence. In other words: $i < j < \mu + v \Rightarrow s_i \neq s_j$ and

$$n \geq \mu \Rightarrow s_n = s_{n+v}$$

The mean values of v and μ have been calculated by Harris (1960):

$$E(v) \approx \frac{1}{4}\sqrt{2\pi m} \quad E(\mu) = E(v) + 1$$

and the standard deviations

$$\sigma_\mu = \sigma_v \approx \sqrt{\left(\frac{2}{3} - \frac{\pi}{8}\right) m}$$

Thus, to make sure that the first $c_{\min}$ states are different, i.e. that $\mu + v > c_{\min}$, we have to let $m \gg {c_{\min}}^2$. Unfortunately, setting $m >> {c_{\min}}^2$ is no guarantee that the first $c_{\min}$ states are different, because we have no easy way of picking f out of F that is sufficiently random to guarantee that (5) and (6) hold; and experimental verification can be quite time-consuming.

The situation becomes easier when the state transition function f is invertible.

Assume now that we have picked a state transition function f at random from the m! members of G. Starting with the random seed s_0, we apply the transition f repeatedly in order to find the cycle length v. If the first i states are different, then the probability that the next state will close the cycle is $\frac{1}{m-i}$ because only the initial state can close the circle. The probability that the cycle length v=c is then

$$P_c = \frac{1}{m}$$

Here, we particularly modified the cycle period length from original 2^{19937} of Mersenne Twister to an much shorter one; 2^{16}. However it seems big yet, even if the number of efforts for generating numbers would not be that huge, the effect of this short cycle will be apparent in the results.

D. Good Candidate : Secrets Module - cryptographic algorithm

The secrets module is used for generating cryptographically strong random numbers suitable for managing data such as passwords, account authentication, security tokens, and related secrets.

Based on **os.urand** it returns a bytestring of size random bytes suitable for cryptographic use.

This function returns random bytes from an OS-specific randomness source. The returned data should be unpredictable enough for cryptographic applications, though its exact quality depends on the OS implementation.

Random module also can generate random data, which is not non-deterministic data. In other words, data generated by the random module can be determined easily by finding the seed used to produce the data. It is not suitable for security purposes. On the other hand, the secrets module is an excellent way to produce secure data. The secret module is a cryptographically strong Pseudo-Random Number Generator useful in security-sensitive applications.

It is a cryptographic algorithm that uses the same secret key for its operation and (if applicable) for reversing the effects of the operation

II. CHOOSING FOUR TESTS TO COMPARE THE RESULTS

Among the performed Tests we've chosen the following 4+1 tests :

- Monobit
- Serial
- Cumulative Sums
- Approximate Entropy
- Random Excursion Variant Test

Due to the Contrast in results which makes it easier to interpret them.

III. TECHNICAL NOTE ON PERFORMING THE TEST

To perform the tests on RNGs we rearranged the linear sequence of numbers generated into a 8-bit representation. Also, provided their Binary representation which is required for RNG Tests. Also, remember we produced random bytes to be more compatible with the Tests.

IV. COMPARING THE RESULTS AND INTERPRETATION

A. Results : Badly Modified Algorithm - Cycle

The corresponding results of the test will be as following for the 4+1 test; • Test results:

• - **PASSED** - score: **1.0** - **Monobit** - elapsed time: 0 ms
• - **PASSED** - score: **0.998** - **Serials** - elapsed time: 208 ms
• - **PASSED** - score: **1.0** - **Cumulative Sums** - elapsed time: 8 ms
• - **PASSED** - score: **0.998** - **Approximate Entropy** - elapsed time: 259 ms
• - **Passed** - score: **1.461** - **Random Excursion Variant** - elapsed time: 1 ms –This P-value $> 100\%$ shows a **Failed** test; indeed $P = 1.461 - 1 = 0.461 \rightarrow$ *Failed*

B. Results : cryptographic algorithm

The corresponding results of the test will be as following for the 4+1 test;

• Test results:

• - **PASSED** - score: **0.070** - **Monobit** - elapsed time: 0 ms
• - **PASSED** - score: **0.294** - **Serial** - elapsed time: 182 ms
• - **PASSED** - score: **0.117** - **Cumulative Sums** - elapsed time: 16 ms
• - **PASSED** - score: **0.163** - **Approximate Entropy** - elapsed time: 240 ms
• - **PASSED** - score: **0.386** - **Random Excursion Variant** - elapsed time: 0 ms

C. comparing Results

The Results in terms of comparison are provided below to show their contrast :

RNG Type	Short Cycle Alg	Cryptographic
MonoBit	1.0	0.070
Serial	0.998	0.294
Cum-Sum	1.0	0.117
ApprEnt	0.998	0.163
REV	Failed	Passed

D. interpretation :

The comparison for the last case shows us obviously that the Cryptographic Algorithm passes the test Successfully while the Badly Modified Algorithm fails. For the first four tests both pass although we need to go deeper to see the difference.

In the Monobit test, the calculated P-Values are different by 1 order of Magnitude; which is significant! In Serial, Cumulative Sums and Approximate Entropy the values are different by several units within the same order of magnitude. While the for REV it is apparent that one fails while the other one passes.

Just from the Appendix.1 one can say that the result ($>$ 0.01) is corresponding to passing the test and considering the sample to be random. While a result of P-Value ($<$ 0.01) means being failed. Yet, for the case ($>$ 0.01), being passed; as the value deviates from the "value = 0.01" it means that the sample is considered random with a lower confidence level. The probability of randomness declines such that e.g. for "Pv= 0.05" this means the sample is random with (95%) confidence level. While the "Pv' = 0.4" means that the sample can be considered random with a confidence level of (60%).

Now, let us represent the results in terms of the confidence level as :

RNG Type	Cryptographic Alg CL%	Badly Modified (cycle) Alg CL%
MonoBit	93.0 %	0.01 %
Serial	70.6 %	0.2 %
Cum-Sum	88.3 %	0.01 %
ApprEnt	83.7 %	0.2 %
REV	Passed	Failed

As a result, one can easily say that the Cryptographic Algorithm, here Secrets Library in Python based on OS module providing functions for interacting with the operating system. is significantly more successful than the simple badly created RNG Algorithm, here in Python provided by definition.

The confidence level of Crypto-Secure Algorithm is usually higher than 70 % while for badly defined Algorithm varies from 0.2 % to 0.01 %. Although, note that in Random Excursion Variant Test, the crypto-secure passes while short cycle Fails.

V. EFFECT OF RNG ON SIMULATION :

The Simulation : The Standard Model of particles physics predicts the existence of the Higgs that couples to the other particles with a strength proportional to their masses. Its discovery has been very difficult because its mass is not determined by the model and its production rate is very small in front of a huge background. Here, we will reproduce the claim of its discovery by the experiments ALTAS and CMS at CERN. For simplicity, we will assume that its mass is known to be 126.5 GeV and we will consider only its decay to a pair of photons.

Now Let us consider the first, simple badly created Algorithm to generate the random sample for simulation. In this case, because the RNG is not good enough, the results will not benefit from a high confidence. Also, to cover the lack of confidence in RNG and avoid possible biases and nonrandom results as well as repetitive patterns, we need to perform the simulation many times, where considering the fact that sometimes it may require performing on supercomputers which is time-consuming and costly, we may face either some problems to perform the simulation properly or we may choose to rely on a sample with a high probability to be biased and suffers from nonrandom features.

On the other hand, our Crypto-Secure Algorithm, enables us to be more confident about our sample, free of repetitive patterns and more complex entropy; so no need to repeat the experiment many times, then the simulation will be less time-consuming and costly. Also, even it may be a better choice to avoid possible biases and having a sample as a good approximation to the real experiment. Remember limitations of the Created RNG Algorithm leads us to significantly unreal and inconsistent results, in MC, while other one will be more consistent.

VI. RNG TEST RESULTS : CRYPTOGRAPHIC ALGORITHM

Eligible test from NIST-SP800-22r1a:

-monobit
-frequency_within_block
- runs
-longest_run_ones_in_a_block
-dft
-non_overlapping_template_matching
- serial
-approximate_entropy
- cumulative sums
-random_excursion
- random_excursion_variant

Test results:

- **PASSED** - score: **0.070** - **Monobit** - elapsed time: 0 ms
- **PASSED** - score: **0.594** - **Frequency Within Block** - elapsed time: 0 ms
- **PASSED** - score: **0.300** - **Runs** - elapsed time: 9 ms
- **PASSED** - score: **0.734** - **Longest Run Ones In A Block** - elapsed time: 7 ms
- **PASSED** - score: **0.608** - **Discrete Fourier Transform** - elapsed time: 0 ms
- **PASSED** - score: **1.0** - **Non Overlapping Template Matching** - elapsed time: 16 ms
- **PASSED** - score: **0.294** - **Serial** - elapsed time: 182 ms
- **PASSED** - score: **0.163** - **Approximate Entropy** - elapsed time: 240 ms
- **PASSED** - score: **0.117** - **Cumulative Sums** - elapsed time: 16 ms
- **FAILED** - score: **0.036** - **Random Excursion** - elapsed time: 16 ms
- **PASSED** - score: **0.386** - **Random Excursion Variant** - elapsed time: 0 ms

VII. RNG TEST RESULTS : SHORT CYCLE ALGORITHM

Eligible test from NIST-SP800-22r1a:

-monobit
-frequency_within_block
- runs
-longest_run_ones_in_a_block
-dft
-non_overlapping_template_matching
- serial
-approximate_entropy
- cumulative sums
-random_excursion
-random_excursion_variant

Test results:

- **PASSED** - score: **1.0** - **Monobit** - elapsed time: 0 ms
- **PASSED** - score: **1.0** - **Frequency Within Block** - elapsed time: 0 ms
- **PASSED** - score: **0.947** - **Runs** - elapsed time: 3 ms
- **PASSED** - score: **0.242** - **Longest Run Ones In A Block** - elapsed time: 13 ms
- **PASSED** - score: **0.918** - **Discrete Fourier Transform** - elapsed time: 1 ms
- **PASSED** - score: **1.0** - **Non Overlapping Template Matching** - elapsed time: 19 ms
- **PASSED** - score: **0.998** - Serial - elapsed time: 208 ms
- **PASSED** - score: **0.998** - **Approximate Entropy** - elapsed time: 259 ms
- **PASSED** - score: **1.0** - **Cumulative Sums** - elapsed time: 8 ms
- **FAILED** - score: **0.0** - **Random Excursion** - elapsed time: 4 ms
- **FAILED** - score: **1.461** - **Random Excursion Variant** - elapsed time: 1 ms

VIII. APPENDIX 1 : HOW TO INTERPRET TEST RESULTS

Various statistical tests can be applied to a sequence to attempt to compare and evaluate the sequence to a truly random sequence. Randomness is a probabilistic property; that is, the properties of a random sequence can be characterized and described in terms of probability. The likely outcome of statistical tests, when applied to a truly random sequence, is known a priori and can be described in probabilistic terms. There are an infinite number of possible statistical tests, each assessing the presence or absence of a "pattern" which, if detected, would indicate that the sequence is nonrandom. Because there are so many tests forjudging whether a sequence is random or not, no specific finite set of tests is deemed "complete." In addition, the results of statistical testing must be interpreted with some care and caution to avoid incorrect conclusions about a specific generator. A statistical test is formulated to test a specific null hypothesis (Ho).

For the purpose of this document, the null hypothesis under test is that the sequence being tested is random. Associated with this null hypothesis is the alternative hypothesis (Ha) which, for this document, is that the sequence is not random. For each applied test, a decision or conclusion is derived that accepts or rejects the null hypothesis, i.e., whether the generator is (or is not) producing random values, based on the sequence that was produced. For each test, a relevant randomness statistic must be chosen and used to determine the acceptance or rejection of the null hypothesis. Under an assumption of randomness, such a statistic has a distribution of possible values. A theoretical reference distribution of this statistic under the null hypothesis is determined by mathematical methods. From this reference distribution, a critical value is determined (typically, this value is "far out" in the tails of the distribution, say out at the 99 % point). During a test, a test statistic value is computed on the data (the sequence being tested). This test statistic value is compared to the critical value. If the test statistic value exceeds the critical value, the null hypothesis for randomness is rejected. Otherwise, the null hypothesis (the randomness hypothesis) is not rejected (i.e., the hypothesis is accepted).

In practice, the reason that statistical hypothesis testing works is that the reference distribution and the critical value are dependent on and generated under a tentative assumption of randomness. If the randomness assumption is, in fact, true for the data at hand, then the resulting calculated test statistic value on the data will have a very low probability (e.g., 0.01 %) of exceeding the critical value.

On the other hand, if the calculated test statistic value does exceed the critical value (i.e., if the low probability event does in fact occur), then from a statistical hypothesis testing point of view, the low probability event should not occur naturally. Therefore, when the calculated test statistic value exceeds the critical value, the conclusion is made that the original assumption of randomness is suspect or faulty. In this case, statistical hypothesis testing yields the following conclusions: reject Ho (randomness) and accept Ha (non-randomness). Statistical hypothesis testing is a conclusion-generation procedure that has two possible outcomes, either accept Hq (the data is random) or accept Ha (the data is non-random).

If the data is, in truth, random, then a conclusion to reject the null hypothesis (i.e., conclude that the data is non-random) will occur a small percentage of the time. This conclusion is called a Type I error. If the data is, in truth, non-random, then a conclusion to accept the null hypothesis (i.e., conclude that the data is actually random) is called a Type II error. The conclusions to accept Ho when the data is really random, and to reject Ho when the data is non-random, are correct.

The probability of a Type I error is often called the level of significance of the test. This probability can be set prior to a test and is denoted as a. For the test, a is the probability that the test will indicate that the sequence is not random when it really is random. That is, a sequence appears to have non-random properties even when a "good" generator produced the sequence. Common values of a in cryptography are about 0.01

The probability of a Type II error is denoted as β. For the test, β is the probability that the test will indicate that the sequence is random when it is not; that is, a "bad" generator produced a sequence that appears to have random properties. Unlike α, β is not a fixed value. β can take on many different values because there are an infinite number of ways that a data stream can be non-random, and each different way yields a different β. The calculation of the Type II error β is more difficult than the calculation of α because of the many possible types of non-randomness. One of the primary goals of the following tests is to minimize the probability of a Type II error, i.e., to minimize the probability of accepting a sequence being produced by a good generator when the generator was actually bad. The probabilities α and β are related to each other and to the size n of the tested sequence in such a way that if two of them are specified, the third value is automatically determined. Practitioners usually select a sample size n and a value for α (the probability of a Type I error - the level of significance). Then a critical point for a given statistic is selected that will produce the smallest β (the probability of a Type II error). That is, a suitable sample size is selected along with an acceptable probability of deciding that a bad generator has produced the sequence when it really is random. Then the cutoff point for acceptability is chosen such that the probability of falsely accepting a sequence as random has the smallest possible value. Each test is based on a calculated test statistic value, which is a function of the data. If the test statistic value is S and the critical value is t, then the Type I error probability is $P\left(S > t \| H_o \text{ is true }\right) = P($ rèject $H_o \mid H_0$ is true), and the Type II error probability is $P\left(S \leq t \| H_0 \text{ is false }\right) = P$ (accept $H_0 \mid H_0$ is false). The test statistic is used to calculate a P-value that summarizes the strength of the evidence against the null hypothesis. For these tests, each P-value is the probability that a perfect random number generator would have produced a sequence less random than the sequence that was tested, given the kind of non-randomness assessed by the test. If a P-value for a test is determined to be equal to 1 , then the sequence appears to have perfect randomness. A P-value of zero indicates that the sequence appears to be completely nonrandom. A significance level (α) can be chosen for the

tests. If P-value $\geq \alpha$, then the null hypothesis is accepted; i.e., the sequence appears to be random. If P-value $< \alpha$, then the null hypothesis is rejected; i.e., the sequence appears to be non-random. The parameter α denotes the probability of the Type I error. Typically, α is chosen in the range $[0.001, 0.01]$.

- An α of 0.001 indicates that one would expect one sequence in 1000 sequences to be rejected by the test if the sequence was random. For a P-value ≥ 0.001, a sequence would be considered to be random with a confidence of 99.9%. For a P-value < 0.001, a sequence would be considered to be non-random with a confidence of 99.9%.
- An α of 0.01 indicates that one would expect 1 sequence in 100 sequences to be rejected. A P-value ≥ 0.01 would mean that the sequence would be considered to be random with a confidence of 99%. A P-value < 0.01 would mean that the conclusion was that the sequence is non-random with a confidence of 99%.

For the examples in this document, α has been chosen to be 0.01. Note that, in many cases, the parameters in the examples do not conform to the recommended values; the examples are for illustrative purposes only.

IX. APPENDIX 2 : DEFINITIONS OF THE TESTS COMPATIBLE WITH THE RNGS SELECTED

A. Frequency (Monobit) Test

The focus of the test is the proportion of zeroes and ones for the entire sequence. The purpose of this test is to determine whether the number of ones and zeros in a sequence are approximately the same as would be expected for a truly random sequence. The test assesses the closeness of the fraction of ones to Vi, that is, the number of ones and zeroes in a sequence should be about the same. All subsequent tests depend on the passing of this test; there is no evidence to indicate that the tested sequence is non-random.

Note that if the P-value were small (<0.01), then this would be caused by $|S_n|$ or $|S_{obs}|$ being large. Large positive values of S_n are indicative of too many ones, and large negative values of S_n are indicative of too many zeros.

B. Frequency Test within a Block

The focus of the test is the proportion of ones within M-bit blocks. The purpose of this test is to determine whether the frequency of ones in an M-bit block is approximately M/2, as would be expected under an assumption of randomness. For block size M=1, this test degenerates to test 1 , the Frequency (Monobit) test.

Note that small P-values (< 0.01) would have indicated a large deviation from the equal proportion of ones and zeros in at least one of the blocks.

C. Runs Test

The focus of this test is the total number of runs in the sequence, where a run is an uninterrupted sequence of identical bits. A run of length k consists of exactly k identical bits and is bounded before and after with a bit of the opposite value. The purpose of the runs test is to determine whether the number of runs of ones and zeros of various lengths is as expected for a random sequence. In particular, this test determines whether the oscillation between such zeros and ones is too fast or too slow.

Note that a large value for V_{obs} would have indicated an oscillation in the string which is too fast; a small value would have indicated that the oscillation is too slow. (An oscillation is considered to be a change from a one to a zero or vice versa.) A fast oscillation occurs when there are a lot of changes, e.g., 010101010 oscillates with every bit. A stream with a slow oscillation has fewer runs than would be expected in a random sequence, e.g., a sequence containing 100 ones, followed by 73 zeroes, followed by 127 ones (a total of 300 bits) would have only three runs, whereas 150 runs would be expected.

D. Longest Run Ones In A Block

The focus of the test is the longest run of ones within M-bit blocks. The purpose of this test is to determine whether the length of the longest run of ones within the tested sequence is consistent with the length of the longest run of ones that would be expected in a random sequence. Note that an irregularity in the expected length of the longest run of ones implies that there is also an irregularity in the expected length of the longest run of zeroes. Therefore, only a test for ones is necessary.

Note that large values of $\chi^2(obs)$ indicate that the tested sequence has clusters of ones.

E. Discrete Fourier Transform (Spectral) Test

The focus of this test is the peak heights in the Discrete Fourier Transform of the sequence. The purpose of this test is to detect periodic features (i.e., repetitive patterns that are near each other) in the tested sequence that would indicate a deviation from the assumption of randomness. The intention is to detect whether the number of peaks exceeding the 95 % threshold is significantly different than 5 %.

A d value that is too low would indicate that there were too few peaks ($< 95\%$) below T, and too many peaks (more than 5 %) above

F. Non-Overlapping Template Matching

The focus of this test is the number of occurrences of pre-specified target strings. The purpose of this test is to detect generators that produce too many occurrences of a given non-periodic (aperiodic) pattern. An m-bit window is used to search for a specific m-bit pattern. If the pattern is not found, the window slides one bit position. If the pattern is

found, the window is reset to the bit after the found pattern, and the search resumes.

If the P-value is very small (< 0.01), then the sequence has irregular occurrences of the possible template patterns.

G. Serial Test

The focus of this test is the frequency of all possible overlapping m-bit patterns across the entire sequence. The purpose of this test is to determine whether the number of occurrences of the 2^m m-bit overlapping patterns is approximately the same as would be expected for a random sequence. Random sequences have uniformity; that is, every m-bit pattern has the same chance of appearing as every other m-bit pattern.

Note that if $\nabla^2\Psi_m^2$ or $\nabla\Psi_m^2$ had been large, then non-uniformity of the w-bit blocks is impUed.

H. 3 Approximate Entropy Test

the focus of this test is the frequency of all possible overlapping m-bit patterns across the entire sequence. The purpose of the test is to compare the frequency of overlapping blocks of two consecutive/adjacent lengths (m and m+1) against the expected result for a random sequence.

Note that small values $ApEn(m)$ would imply strong regularity Large values would imply substantial fluctuation or irregularity.

I. Cumulative Sums (Cusum) Test

The focus of this test is the maximal excursion (from zero) of the random walk defined by the cumulative sum of adjusted (-1, +1) digits in the sequence. The purpose of the test is to determine whether the cumulative sum of the partial sequences occurring in the tested sequence is too large or too small relative to the expected behavior of that cumulative sum for random sequences. This cumulative sum may be considered as a random walk. For a random sequence, the excursions of the random walk should be near zero. For certain types of non-random sequences, the excursions of this random walk from zero will be large.

Note that when mode = 0, large values of this statistic indicate that there are either "too many ones" or "too many zeros" at the early stages of the sequence; when mode = 1 , large values of this statistic indicate that there are either "too many ones" or "too many zeros" at the late stages. Small values of the statistic would indicate that ones and zeros are intermixed too evenly.

J. Random Excursion Test

The focus of this test is the number of cycles having exactly K visits in a cumulative sum random walk. The cumulative sum random walk is derived from partial sums after the (0,1) sequence is transferred to the appropriate (-1, +1) sequence. A cycle of a random walk consists of a sequence of steps of unit length taken at random that begin at and return to the origin. The purpose of this test is to determine if the number of visits to a particular state within a cycle deviates from what one would expect for a random sequence. This test is actually a series of eight tests (and conclusions), one test and conclusion for each of the states: -4, -3, -2, -1 and +1, +2, +3, +4.

Note that if $\chi^2(obs)$ were too large, then the sequence would have displayed a deviation from the theoretical distribution for a given state across all cycles.

K. Random Excursion Variant Test

The focus of this test is the total number of times that a particular state is visited (i.e., occurs) in a cumulative sum random walk. The purpose of this test is to detect deviations from the expected number of visits to various states in the random walk. This test is actually a series of eighteen tests (and conclusions), one test and conclusion for each of the states: -9, -8, . . ., -1 and +l,+2, ...,+9.

X. APPENDIX.3 - RNG TEST RESULTS : ORIGINAL MERSENNE TWISTER ALGORITHM

Eligible test from NIST-SP800-22r1a:

-monobit
-frequency_within_block
- runs
-longest_run_ones_in_a_block
-dft
-non_overlapping_template_matching
- serial
-approximate_entropy
- cumulative sums
-random_excursion
-random_excursion_variant

Test results:

- **PASSED** - score: **0.488** - **Monobit** - elapsed time: 0 ms
- **PASSED** - score: **0.522** - **Frequency Within Block** - elapsed time: 0 ms
- **PASSED** - score: **0.356** - **Runs** - elapsed time: 8 ms
- **PASSED** - score: **0.459** - **Longest Run Ones In A Block** - elapsed time: 0 ms
- **PASSED** - score: **0.538** - **Discrete Fourier Transform** - elapsed time: 0 ms
- **PASSED** - score: **1.0** - **Non Overlapping Template Matching** - elapsed time: 24 ms
- **PASSED** - score: **0.811** - Serial - elapsed time: 170 ms
- **PASSED** - score: **0.564** - **Approximate Entropy** - elapsed time: 292 ms
- **PASSED** - score: **0.380** - **Cumulative Sums** - elapsed time: 0 ms
- **FAILED** - score: **0.314** - **Random Excursion** - elapsed time: 19 ms
- **FAILED** - score: **0.441** - **Random Excursion Variant** - elapsed time: 0 ms