



Comparison of Pseudorandom Number Generators and Their Application for Uncertainty Estimation Using Monte Carlo Simulation

K. Malik¹, J. Pulikkotil² and A. Sharma^{2*} 

¹USICT- Guru Gobind Singh Indraprastha University, Dwarka, New Delhi 110075, India

²CSIR-National Physical Laboratory, Dr. K. S. Krishnan Marg, New Delhi 110012, India

Received: 03 November 2020 / Accepted: 26 March 2021 / Published online: 10 June 2021

© Metrology Society of India 2021

Abstract: Generating random numbers is prerequisite to any Monte Carlo method implemented in a computer program. Therefore, identifying a good random number generator is important to guarantee the quality of the output of the Monte Carlo method. However, sequences of numbers generated by means of algorithms are not truly random, but having certain control on its randomness essentially makes them pseudo-random. What then matters the most is that the simulation of a physical variable with a probability distribution, needs to have the same distribution generated by the algorithm itself. In this perspective, considering the example of gauge block calibration given in “Evaluation of measurement data—Supplement 1 to the “Guide to the expression of uncertainty in measurement”—Propagation of distributions using a Monte Carlo method”, we explore the properties and output of three commonly used random number generators, namely the linear congruential (LC) generator, Wichmann-Hill (WH) generator and the Mersenne-Twister (MT) generator. Extensive testing shows that the performance of the MT algorithm transcends that of LC and WH generators, particularly in its time of execution. Further, these generators were used to estimate the uncertainty in the measurement of the length, with input variables having different probability distributions. While, in the conventional GUM approach the output distribution appears to be Gaussian-like, we from our Monte-Carlo calculations find it to be a students’ t-distribution. Applying the Welch-Satterthwaite equation to the result of the Monte Carlo simulation, we find the effective degrees of freedom to be 16. On the other hand, using a trial–error fitting method to determine the nature of the output PDF, we find that the resulting distribution is a t-distribution with 46 degrees of freedom. Extending these results to calculate the expanded uncertainty, we find that the Monte-Carlo results are consistent with the recently proposed mean/median-based unbiased estimators which takes into account the artifact of transformation distortion.

Keywords: Random; Pseudorandom number generators; Simulations; Statistical tests; Monte Carlo simulation

1. Introduction

In the realms of metrology, the Guide to the expression of Uncertainty in Measurement (GUM) provides the basic framework for uncertainty evaluation in measurement [1]. Although it is widely accepted by a large body of theoretical and experimental works, the basic assumption to the use of the method is that all systematic errors are identified and corrected at the very outset of the evaluation process. At times, GUM requires sophisticated mathematical models to express uncertainty associated both with randomness and systematic biases. In particular, GUM becomes quite

tedious in cases where model linearization and determination of partial derivatives become increasingly complex [16].

An alternative method suggested in Supplement-1 of the GUM documentation is the use of Monte Carlo (MC) simulations [2]. In this procedure, pseudo-random numbers are used to describe the probability distribution of the input variables which with a known functional relationship yields a single numerical value as output. The process is repeated for large input data set so as to produce a set of simulated results, and the statistical quantities such as mean and standard deviation are determined which then represent the estimates of the measurand and its uncertainty.

*Corresponding author, E-mail: anjali@nplindia.org

Evidently, generation of random numbers therefore becomes a prerequisite to MC simulations. Here we note that for the simulations to render valid statistical inferences, it is important that the sample generated by the random number generators are representative of the population itself. Thus, what remains as a challenge is how to choose samples from a population, at random. In this regard, several studies have tested the properties of various random number generators built on a variety of computer algorithms. However, such a generated sequence using deterministic mathematical equations are strictly not truly random and are often referred as pseudo-random generators (PRNGs).

A myriad of PRNGs have been developed and are used in several applications such as cryptography, gaming, statistical sampling, security applications and research where unpredictability is desired [3]. Among these, the WH generator [4] and the MT [3], 5 are found more popular due to their brief algorithm, ease in implementation, simple computation and large period. In fact, the WH generator consists of three linear congruential generators, where an LC generator yields a pseudorandom number sequence which is calculated using a discontinuous linear equation [6]. Below we briefly discuss the basic features of these three PRNGs and perform various testing [7], 8, checking its validity toward the problems in metrology. Furthermore, we apply these PRNGs to an illustrative example as provided in the Supplement-1 of the GUM documentation [1].

Please note that several other efficient PRNGs such as the Squares RNG, that has been developed very recently in 2020 as a modification of the Middle Square Weyl Sequence PRNG, 64-bit MELG, Xoroshiro128 + , Random Cycle Bit Generator (RCB), have been developed much more recently, in the past decade. Although developed several years before, MT, WH Generator and the LC Generator remain the most widely used PRNGs in software systems and platforms, such as Microsoft Excel, Python, Java, R, etc. Moreover, they are computationally efficient and quick in their execution. Their popularity and efficiency provide the basis for them being chosen in this text.

The GUM assumes that the distribution obtained with a large number of measurement model evaluations from a Monte Carlo simulation process is essentially Gaussian, however when these simulations were investigated for a gauge block calibration and compared with corresponding identical mean and standard deviation of a pure Gaussian distribution, it was determined that the actual distribution more closely corresponded with a Students *t*-distribution with 46 degrees of freedom.

To calculate the effective degrees of freedom of the distribution obtained through the Monte Carlo process, the Welch-Satterthwaite equation [21], 22 was applied, adhering to the analytical approach described in [21], 23.

The Welch-Satterthwaite equation is used to calculate an approximation to the effective degrees of freedom for a linear combination of independent components.

Further, due to the lengthy and complicated nature of the calculations involved in the Welch-Satterthwaite equation [24], the trials-and-errors method was used as an alternate to determine the degrees of freedom associated with the obtained distribution. The results given by this method were found to be more suitable in our case study.

The paper is organized as follows. Section II describes the details of the implementation of the PRNGs and computation mentioning the specification of the hardware, software versions and related information. In section III, we describe the statistical test-results of the of three PRNGs, namely the MT, WH generator and LC generators for random numbers, followed by the use of random numbers generated using the PRNGs to carry out the Monte Carlo Simulations, to determine the uncertainty of the measured length of an end gauge (Illustrative example H.1 as provided in the Supplement-1 of the GUM documentation) and further compare the resulting distribution to the one mentioned in the GUM. The final section of this paper, Section IV, lists the conclusions derived from the previous sections.

2. The Prng Methods and Computational Details

2.1. Pseudorandom Number Generators

2.1.1. Linear Congruential Generator

LC Generators are one of the oldest and most well-known PRNG. It is easy to implement and is faster when compared to many other generators. Moreover, its memory requirements are quite less. The values are generated by repeatedly applying the seed value with a recurrence relation of the form given in (1).

$$X_{n+1} = (aX_n + c) \times \text{mod}(m) \quad (1)$$

where the multiplier (*a*), modulus (*m*) and the increment (*c*) are constants. The output sequence of LC generator sensitively depends on the choice of all three constants that constitute the generating function. Thus, the property of the random sequence can be duplicated, provided the initial values are known. Thus, it has both merits and demerits, depending on the application. For instance, LC generators are seldom a choice of preferences when it comes to cryptography. A cryptographically secure PRNG (CSPRNG) apart from being able to produce seemingly random values, should also guarantee unpredictability in its sequences. Since they are associated with cryptography and cybersecurity, any malicious attacker should not be able to

predict the entire sequence, even when given access to the initial state. Thus, CRPRNGs require higher entropy or randomness than usual PRNGs, such as the LC Generator. Therefore, every CSPRNG is a PRNG but not vice-versa.

Besides, the period of LC generators depends on the modulus 'm' and increases as 'm' is increased. The faster computation and memory advantage make LC generators favourable for small and simple applications.

The algorithm for LC generator is:

1. Start with a seed value. In the present work, we have used the seed value as 0.
2. Choose appropriate values for coefficients a, m and c. The value of m should be preferably high to avoid repetition in the generated sequence.
3. Apply the recurrence relation 1.
4. Divide the acquired value by m to scale it in the range of 0 to 1.
5. Repeat steps (3) and (4) till the required number of values are generated.

2.1.2. Wichmann–Hill Generator

The WH Generator is an extension of the LC generator, as it uses three linear congruential equations with different values of the moduli of the form Eq. 1, i.e.,

$$X_{n+1}^i = (a^i \times X_n^i) \times \text{mod}(m^i); \text{ for } i = 1, 2, 3 \quad (2)$$

Here X_0^i ; ($i = 1, 2$ and 3) are the three seed values. The values generated by each of these equations are summed, and their modulo with 1 is taken to get the final value between 0 and 1. The period of the WHG is determined by the three moduli m_1 , m_2 and m_3 and, is equal to their lowest common multiple (LCM). Common values chosen for m_1 , m_2 and m_3 are 30,268, 30,307 and 30,323, respectively, which provides the generated sequence a period of 6.95E12 [4].

The algorithm for WH generator is as follows:

1. Start with three seed values and choose appropriate values for coefficients a_i and m_i ($i = 1, 2$ and 3). In the present work, we have used X_0^i as 0, 1 and 2 for $i = 1, 2$ and 3 , respectively.
2. Apply the recurrence relations given in Eq. 2.
3. Sum the three acquired values and take their modulus with 1.
4. Repeat steps (3) and (4) till the required number of values are generated.

2.1.3. Mersenne Twister

Mersenne Twister is by far the most widely used general-purpose PRNG. It is named after the fact that its period of

repetition i.e., the number of values after which the generated numbers start reoccurring, is a Mersenne prime. A Mersenne prime is a prime number, which is one less than a power of 2, and is thus represented as $2^n - 1$. It has a large period of typically $2^{19937} - 1$ and generates a statistically uniform distribution of values. It has evolved from the feedback shift register class of generators [3] [5].

It is used as the "default" PRNG in many popular software systems and platforms, including Python, Ruby, MATLAB, Octave and R [7]. Its popularity arises because of its large period and much faster generation as compared to hardware-implemented random number generators. It passes various statistical tests for randomness and is considered one of the best PRNGs for non-cryptographic purposes.

The algorithm of its implementation is shown below:

1. Start with a seed value. In the present work, we used the seed value as 0.
2. Initialize the seed value into a state i.e., the first state
3. Transform the state with a one-way function, called twist, into another state
4. Then apply another function, say g, to the state to generate a list of 624 random numbers
5. To generate further numbers, twist the state again and repeat steps 3 and 4.

MT differs from other PRNGs as it generates 624 random numbers from one state instead of just one, and the first state is not used to generate any numbers. Since the transforming function g is not a one-way function, it can be inverted. Thus, it is not a cryptographically secure PRNG.

2.1.3.1. Computational Details All the computations and tests were performed on a system with Intel® Core i5-8250U CPU with 8 Gigabytes of RAM. The operating system used is Microsoft Windows 10 (64 bit). The implementation of all the generators and statistical tests were carried out locally using Python version 3.7. The IDE used was Jupyter Notebooks and random, numpy, statistics, time, matplotlib, pandas and scipy were the Python libraries used.

3. Results and Discussion

3.1. Testing the PRNGs

Although, a computer generated sequence of numbers cannot be strictly random, a generating function or a recurrence relation can yield statistically random numbers provided that it contains no observable regularities or patterns. A wide range of randomness test suites have been proposed, and it is necessary though not sufficient that the

PRNGs must pass these tests to be categorized as useful and efficient for applications. In this regard, the following test suites are adopted:

- (i) Check for uniformity,
- (ii) Independence
- (iii) Test for Correlations
- (iv) Runs Test for Randomness
- (v) Value Distribution Analysis
- (vi) Time of Execution

3.1.1. Check for Uniformity

Uniformity is an important property of a random number sequence. The generated random values must be evenly distributed over the whole range considered, so as to validate the basic postulate of probability theory that all events across the range are equally likely to happen. Unequivocally, this implies that skewed or modulated distributions pose bias and hence can be summarily rejected for the given set of input parameters. To check for uniformity, the generated sequence of random numbers is divided into several sub-intervals with equal bin size (h). Ideally, if uniform, it is expected that the frequency of values in each of the sub-interval must be equal. In other words, if ‘ N ’ random samples are divided into ‘ K ’ sub-intervals, then each of the ‘ K ’ sub-intervals has approximately ‘ N/K ’ values. To observe this trend, the value of N should be fairly large. Further, the χ^2 -test is employed. It is widely used to analyse categorical data distribution and check for a notable difference between two statistical data sets, one containing the observed frequencies and the other containing the expected frequencies of occurrence. The implementation of the χ^2 -test includes the following steps:

1. Formulate the null (H_0) and alternate (H_1) hypothesis:
 - (i) H_0 : the observations have a uniform distribution and
 - (ii) H_1 : the observations do not have a uniform distribution.
2. Tabulate the random numbers with appropriate bin size (h). Note that since the χ^2 -test is valid only for categorical values, the sequence must be first divided into bins. The frequency of values in each bin can then be considered as categorical. The value of the expected frequency (v_{ex}) is calculated as the ratio of “number of random numbers” to that of the “number of bins”.
3. The χ^2 -test is evaluated by means of the equation ($\chi^2 = \frac{\sum (v_{obs} - v_{ex})^2}{v_{ex}}$), where v_{obs} is the observed frequency.
4. The calculated χ^2 value is compared against the critical value obtained from the χ^2 -distribution table. If the value is greater than the critical value for a given confidence level, the null hypothesis of uniformity is rejected, else accepted.

A quick visual representation for uniformity of the generated random numbers can be usually expressed in the form of histogram plots. In Fig. 1, we show the histogram plots representing the random numbers generated using the LC, WH and MT random number generators for different sample sizes. The bin size is 0.1 for all plots. We find that the property of uniformity is achieved when the sample size is considerably large. As evident, for low sample sizes, Fig. 1a, b, none of the three generators tends to show the basic feature of uniformity of the random number distribution. However, the sample size increases (see Fig. 1c, d), the LC, WH and MT generators show not only increasing uniformity but also become more or less indistinguishable.

In Table 1, we show the calculated χ^2 values for different sample and bin sizes on the random number sequence generated by LC, WH and MT generators. In majority of the sample configurations, we find that the generated random distribution pass the χ^2 -test. While the MT generators tend to pass over all chosen configurations, LC fails when for a sample size of 100 random numbers with 100 bins. On the other hand, the behavior of WH generator, on the basis of sample size and bin size appear non-systematic. For intermediate sample sizes considered, i.e., $N = 10^3$ and 10^4 , with bin size 100, the WH generator fail the χ^2 -test. This shows that for moderately sized samples, too small a bin size could fail the uniformity test. Hence it becomes very elementary that an appropriate sample size must be guaranteed. In this study, we find that as the sample size increases from 10^2 to 10^5 , all the three generators, i.e., LC, WH and MT, show increasing uniformity. This is also evident from Fig. 1.

3.1.2. Test for Independence and Auto-Correlation

The property of independence of random numbers states that any value of the random sequence must be independent of its previous values, i.e., a random number generated is independent to the previously generated values [9]. Generally, the auto-correlation test is used to test for independence. Auto-correlation is the degree of correlation in the same set of data, in different intervals. In other words, a random number sequence is said to be auto-correlated, if due to the presence of a pattern, the succeeding values can be predicted using the preceding values. The test checks for correlation for every ‘ m ’ values, also called the ‘lag’ value, starting from a specified initial index [10]. In general, the following steps are adopted in applying the auto-correlation test.

1. Formulate the null (H_0) and alternate (H_1) Hypothesis, where H_0 is accepted when the sequence is independent and H_1 when the sequence is dependent.

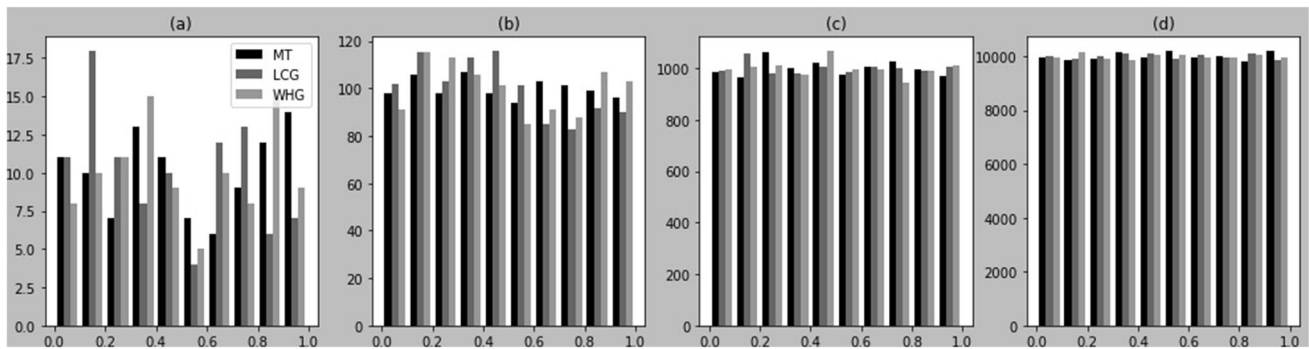


Fig. 1 Figures comparing the uniformity of the three generators with bin size of 0.1 and (a) one hundred, (b) one thousand, (c) ten thousand and (d) one hundred thousand values, respectively

Table 1 Table showing the χ^2 -test results for the three PRNGs adopted in the work

Sample size	χ^2 value			
	Bin size	MT	LC	WH
100	10	4.00	9.00	11.80
	20	9.12	20.80	16.40
	100	84.00	128.00	102.00
1000	10	25.7	11.46	12.08
	20	41.40	28.24	26.92
	100	110.40	120.40	138.80
10,000	10	6.65	4.60	8.60
	20	13.68	13.10	24.68
	100	92.18	91.50	127.74
100,000	10	3.78	8.97	6.98
	20	10.43	18.78	20.47
	100	116.41	74.46	87.63

LC Linear Congruential), WH Wichmann–Hill, MT Mersenne Twister generators. For a confidence interval of 95%, the critical value (χ^2_c) from the χ^2 -table are also shown. The “h” values represent the bin size. Accordingly, uniformity is granted when the calculated $\chi^2 > \chi^2_c$. The χ^2_c for Table 1 are 16.92 for 10 bins, 30.14 for 20 bins and 123.23 for 100 bins

2. Receive a sequence of N random numbers, the lag value (m) and the initial index (i).
3. Determine all the values starting from index i to N , having a lag of m , i.e., for $R_i, R_i + m, R_i + 2m, \dots, R_i + (M + 1)m$; where M is the total number of such terms given by $i + (M + 1)m \leq N$.
4. Calculate the distribution using the expression:

$$\rho = \frac{1}{M + 1} \left[\sum_{k=0}^M R_{i+km} \times R_{(k+1)m} \right] - 0.25$$

5. Calculate the standard deviation using:

$$\sigma_\rho = \frac{\sqrt{13M + 7}}{12(M + 1)}$$

6. Calculate the test statistic, $Z = \frac{\rho}{\sigma_\rho}$
7. Compare the above calculated Z with the critical value of Z for a certain confidence level, 1.96 for $\alpha = 95\%$. The null hypothesis of independence is accepted if the condition is satisfied. $-\alpha - Z_{\frac{\alpha}{2}} \leq Z \leq Z_{\frac{\alpha}{2}}$

For the critical value of Z , a confidence value of 95% is taken, which implies that if the calculated Z value lies in the interval $[-0.98, 0.98]$, the null hypothesis for the independence of values holds valid. Table 2 shows the result of the auto-correlation tests for all the three generators with sample size $N = 10^4$ and 10^5 . Irrespective of the sample size, the MT and LC generators pass the auto-correlation test, while WH generator fails the test for $N = 10^4$. From these results, we infer that MT and LC generated random numbers set are independent of sample dimensions while, WH generator requires larger sample size to avoid auto-correlation with a given set of random number sequence.

3.1.3. Test for Correlation

Correlation is a single number metric that is used to define the degree to which two sets of data are linked or interconnected i.e., when a change in one set is accompanied by a change in the other set, they are said to be correlated. It measures the strength and direction of the association. It is very important that the values generated by any PRNG have diminutive or no correlation, as it would make the sequence predictable. Correlated data have a particular trend or direction that the points follow, which makes them prone to patterns.

The correlation coefficient varies between 1 and -1 , with 1 indicating a very strong positive correlation and -1 indicating a very strong negative correlation. The correlation between the two datasets becomes weaker as the correlation coefficient approaches 0. When the coefficient is 0, it indicates that there is no connection between the two sets of data. Pearson r correlation is the most commonly

Table 2 The comparison of the Autocorrelation, Correlation and Runs Test for the random number sequences generated by the three PRNGs, for given sample sizes of 10^4 and 10^5

RNG	Autocorrelation		Correlation		Runs test	
	10^4	10^5	10^4	10^5	10^4	10^5
MT	0.081	-0.674	0.01	0.002	0.740	-1.068
LC	0.271	-0.693	-0.008	0.002	-1.740	-0.854
WH	1.085	0.570	-0.003	0.004	0.180	-0.132

used correlation statistic, and it measures the degree of relationship between linearly related variables and is given as [7]:

$$r_{xy} = \frac{n\sum x_i y_i - \sum x_i \sum y_i}{\sqrt{n\sum x_i^2 - (\sum x_i)^2} \sqrt{n\sum y_i^2 - (\sum y_i)^2}} \quad (3)$$

In Eq. 3, r_{xy} represents the correlation coefficient, x_i is the i th value of the first set of values, y_i is the i th value of the second set of values, and 'n' is the number of values in both sets. Random sequences have almost zero correlation, and hence a good PRNG should also have $r_{xy} \approx 0$.

In Table 2, we show the correlation values for the random number sequences generated by the LC, WH and MT for a sample size of 10^5 and 10^6 . The results show that all values of correlation calculated using Pearson's method are close to zero, although LC generator with 10^4 random numbers in a sample looks relatively more correlated compared to WH and MT PRNGs.

3.1.4. Runs Test for Randomness

If a sequence shows long runs of one category of values, the chances of the next value also falling in that particular category becomes high. Similarly, if the runs are too short or too frequent, it becomes easy to predict the category of the next value. This takes away from the randomness of the sequence. So, a sequence must pass the Runs Test to be free of patterns and be less predictable [19].

In fact, The Runs Test is a statistical test that uses 'runs' of data to determine whether the sequence is random or follows a pattern. A run is defined as a series of increasing or decreasing values, and the number of these values is called the length of the run. Several methods can be used to define a run, but each of them must be able to divide the data into two distinct categories, such as odd or even, heads or tails, positive or negative and so on. This test counts the total number of runs relative to the number of values falling into each of the two categories.

The sequence is less likely to be random if the number of runs is too high or too low, as this would suggest that the data are falling into a pattern. In our case, we will take the values greater than or equal to the median of the given

sequence as positive and the other values as negative. A new run starts when a positive value is followed by a negative value or vice-versa [7], 11.

The algorithm used to implement the Runs Test and to calculate the test statistic, Z has been thoroughly explained in [20].

Compare the above calculated Z with the critical value of Z_C for a certain confidence level α , i.e., 1.96 for $\alpha = 95\%$. The null hypothesis of randomness is accepted if the condition $-Z_{\frac{\alpha}{2}} \leq Z \leq Z_{\frac{\alpha}{2}}$ is satisfied.

In Table 2, we show the results of the Runs Test for all three generators with sample size $N = 10^4$ and 10^5 . The MT and WHG pass the test for both sample dimensions, whereas LC appears to fail for smaller sample size dimensions.

3.1.5. Value Distribution Analysis

This property of Value Distribution Analysis states that for more than one independent sequences of random values generated, the distribution of the sum of the sequences is triangular for two sequences and tends to become exponential as the number increases [12]. For all generated values lying in between 0 and 1, the mean of the sequences generated should be approximately half the number of sequences (n), and the range should be between 0 and n .

In Fig. 2, we show the distribution for values for $n = 2$ to 5 for LC, WH and MT random number generators. The number of values is taken to be 10^5 in each case, and the values are divided into 100 bins, with the x -axis representing the bins, and the y -axis representing the frequency of occurrence for each bin. For $n = 2$, the distributions are more or less triangular for all the three generators. As the value of n increases, the distributions tend to gain an exponential shape. It is evident that the MT generator display the value distribution property quite well in comparison to the LC and WH generated random number distribution. It is also observed that LC generators lag in displaying the evolution in the sum of the distribution from triangular to exponential with increasing n .

The mean value and the standard deviation obtained from the analysis are tabulated in Table 3 that shows the expected and actual values of the means in all considered

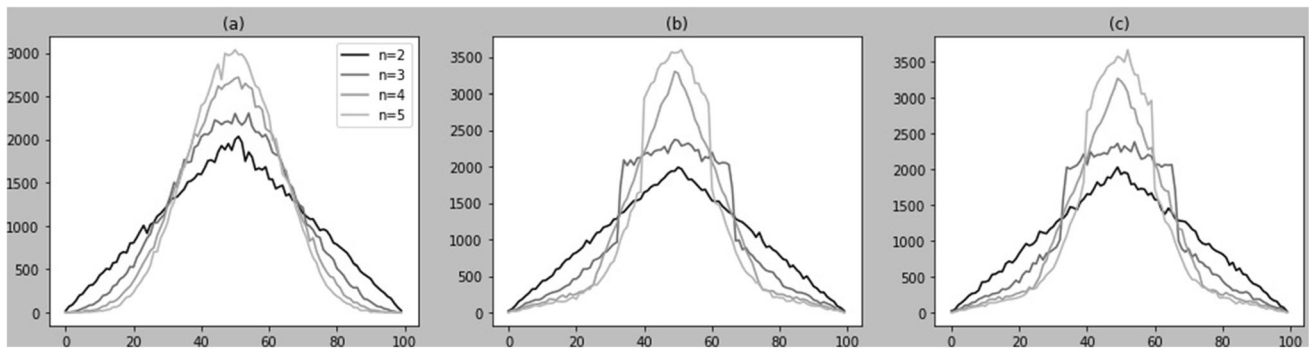


Fig. 2 Comparison of Value Distribution Analysis for the random number distributions generated by the PRNGs, for the given sample size of 10^5

Table 3 Comparison of expected mean (X_E) and actual mean (X_A) values and standard deviation (σ) following the value distribution analysis on the PRNGs with sample size $N = 105$

PRNG	n	X_E	X_A	σ
MT	2	1	1.001	0.408
	3	1.5	1.502	0.499
	4	2	2.001	0.576
	5	2.5	2.498	0.647
LC	2	1	0.999	0.408
	3	1.5	1.499	0.499
	4	2	1.999	0.577
	5	2.5	2.499	0.645
WH	2	1	1.001	0.408
	3	1.5	1.500	0.500
	4	2	1.999	0.579
	5	2.5	2.499	0.648

cases. The values of the standard deviation for any number of sequences (n) are almost identical for all the generators.

3.1.6. Time of Execution

The time taken in the generation of values for the three PRNGs, in generating 10^2 – 10^5 values was calculated and plotted as shown in Fig. 3. For smaller sample sizes ($N < 10^4$), the execution time was almost identical for all the three PRNGs. However, with increasing sample sizes, the execution time differs exponentially.

The MT algorithm shows to be the fastest random number generator, followed by LC and WH generators. For generating 10^6 numbers MT algorithm took 0.18 s, while LC took 0.32 s, and WH generator took approximately 0.51 s. Alternatively, MT takes approximately 30%, and LC takes approximately 64% of the time taken by the WHG to generate 10^6 random numbers, respectively. The fact that WH generators takes more time is evident from its algorithm that it is based on three linearized recursion formulae.

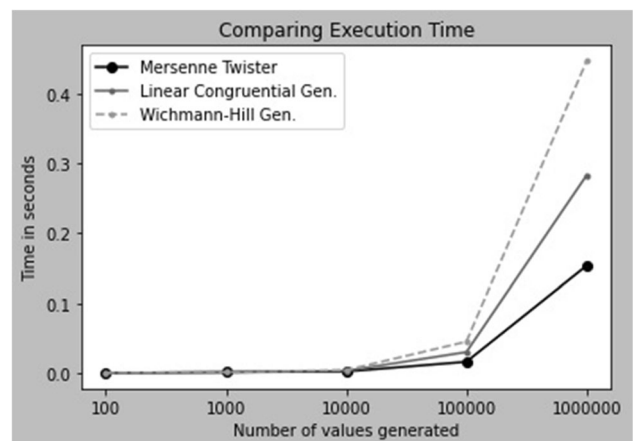


Fig. 3 Comparison of execution time for the three PRNGs

Based on these tests, it appears that MT generators are fast and reliable PRNGs and can efficiently facilitate Monte Carlo simulation for uncertainty estimation [15].

4. Illustration of PRNGs in Uncertainty Estimation in Using Monte Carlo Simulation

Monte Carlo Simulation is a statistical technique that is used to ascertain uncertainty in a measurement, by calculating the same quantity repeatedly with random values, in order to get all the possible outcomes [18]. With a large enough number of iterations, Monte Carlo Simulation is able to provide all outcomes that might occur, with their probabilities of occurrence. In this paper, we use the Monte Carlo Simulation to determine the uncertainty in the measurement of a nominally 50 mm. end gauge, by comparing it with a known standard of the same nominal length, as described in the JCGM’s Guide to Uncertainty Measurement (GUM)—101. The MT and WHG were used to generate and sample the distributions required for its implementation and were also checked for their suitability in simulation tasks simultaneously.

The difference in lengths of the 2 end gauges (d) is given using Eq. (4) [1].

$$d = L(1 + \alpha.\theta) - L_s(1 + \alpha_s.\theta_s) \tag{4}$$

where L is the length measured at 20 °C, L_s is the standard length at 20 degree Celsius given in the calibration certificate, α and α_s are the coefficients of thermal expansion, and θ and θ_s are the deviations from specified temperature.

From the above expression, the value of L can be derived as shown in Eq. (5), which can be approximated to Eq. (6) for practical use.

$$L = \frac{L_s(1 + \alpha_s\theta_s) + d}{1 + \alpha\theta} \tag{5}$$

$$L = L_s + d - L_s(\alpha_s\theta_s - \alpha\theta) \tag{6}$$

If we consider $d = D + d_1 + d_2$, $\delta\theta = \theta - \theta_s$ and $\delta\alpha = \alpha - \alpha_s$, the Eq. (6) can be rewritten as the final Eq. (7).

$$\delta L = L_s + D + d_1 + d_2 - L_s[\delta\alpha(\theta_0 + \Delta) + \alpha_s\delta\theta] - L_{nom} \tag{7}$$

Equation (7) is used to calculate the difference in measured length with up to 10 million trials, each using a random value from the specified distribution of the factors, specified in GUM 101, upon which the length (L) depends.

4.1. Distributions of All Factors Involved

This subsection specifies the procedures used to generate the specified distributions and then sample random values from them, for all factors which are used in Eq. (7) to calculate the difference in measured length. Both the MT and WHG are used to generate these distributions. The sampling process remains the same for both the PRNGs. The sampled values are then input in the Eq. (7) for 10^6

trials of the Monte Carlo Simulation, and consequently 10^6 values of difference in length are received. In the end, a frequency distribution of all the acquired values is generated, specifying the likelihood of their occurrence. According to the GUM, this distribution should be a Students t-distribution which may only reasonably be approximated as a normal/Gaussian distribution in the special case where the model exhibits a large number of degrees of freedom.

All the following values and parameters are specified in the Sect. 9.5.2 of the GUM 101 [1]. The parameters, distributions and the procedures used to generate them have been explained briefly below for better more completeness.

4.1.1. Standard Length (L_s)

The Standard Length forms a scaled and shifted *T-Distribution*, with mean (μ) = 50,000,623 nm., standard deviation (σ) = 25 nm and degrees of freedom (ν) = 18.

The following steps were used to generate this distribution:

1. Make a draw (t) from the central t-distribution with 18 degrees of freedom.
2. Required value (E) is acquired by applying Eq. (8) with the t value sampled above.

$$E = \mu + \frac{\sigma}{\sqrt{\nu + 1}} t \tag{8}$$

Figure 4a shows the distribution generated by the above process.

4.1.2. Average Length Difference (D)

The Average Length Difference forms a scaled and shifted *T-Distribution*, with mean (μ) = 215 nm., standard deviation (σ) = 6 nm and degrees of freedom (ν) = 24.

Figure 4b shows the distribution generated for D .

4.1.3. Random Effect of Comparator (d_1)

The Random Effect of Comparator forms a scaled and shifted *T-Distribution*, with mean (μ) = 0 nm., standard deviation (σ) = 4 nm and degrees of freedom (ν) = 5.

Figure 4c shows the distribution generated for d_1 .

4.1.4. Systematic Effect of Comparator (d_2)

The Systematic Effect of Comparator forms a scaled and shifted *T-Distribution*, with mean (μ) = 0 nm., standard deviation (σ) = 7 nm and degrees of freedom (ν) = 8.

Figure 4(d) shows the distribution generated for d_2 .

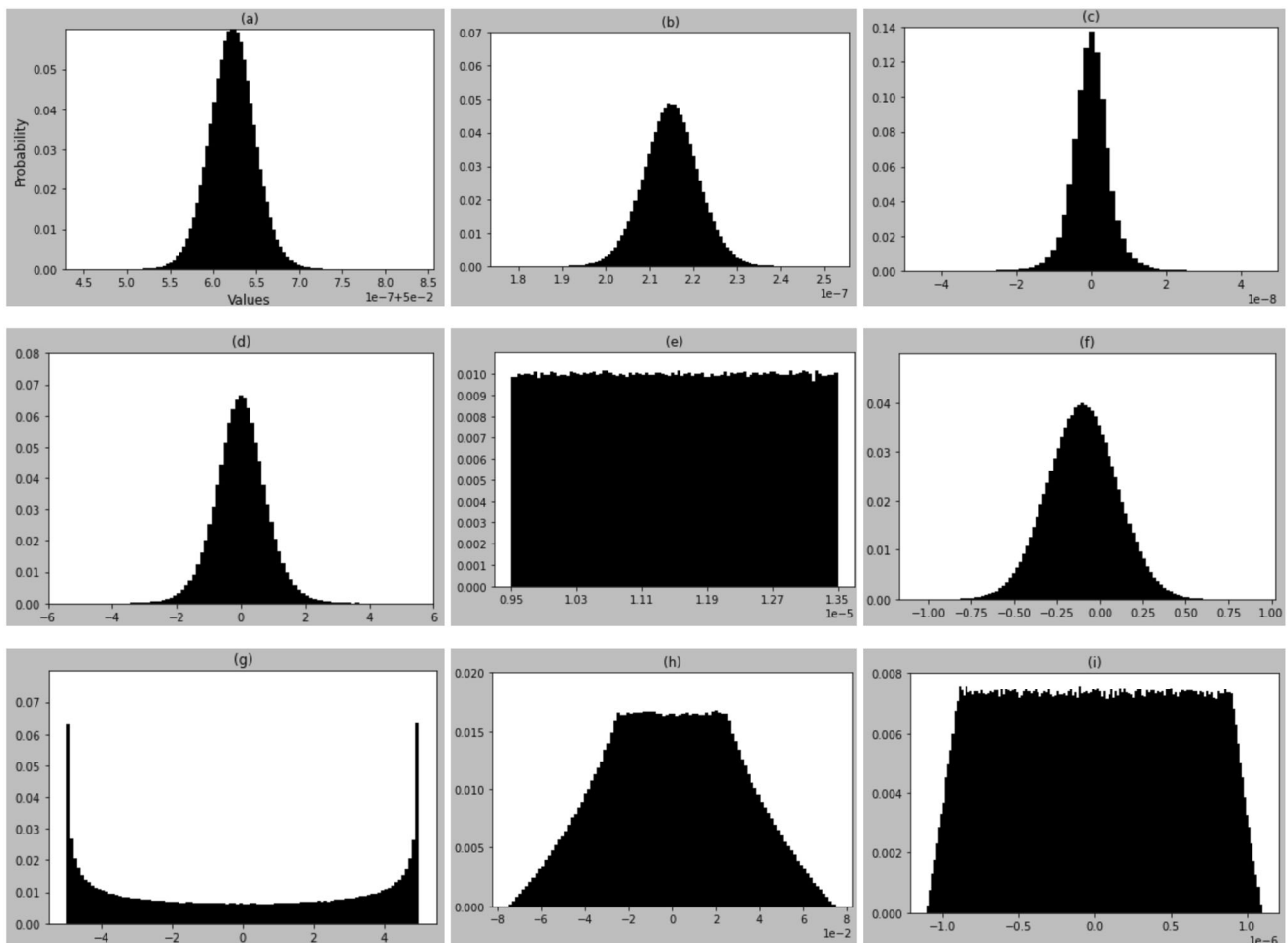


Fig. 4 The distributions generated for the input parameters using the Mersenne Twister PRNG, for a sample size 10^6

4.1.5. Thermal Expansion Coefficient (α_s)

The Thermal Expansion Coefficient is to be assigned a rectangular distribution with limits $a = 9.5 \times 10^{-6} \text{ }^\circ\text{C}^{-1}$ and $b = 13.5 \times 10^{-6} \text{ }^\circ\text{C}^{-1}$.

The following steps were used to generate the rectangular distribution:

1. Make a random draw (r) from the standard rectangular distribution, with limits $a = 0$ and $b = 1$.
2. Required value (E) is calculated by using the ' r ' value as input to Eq. (9).

$$E = a + (b - a)r \tag{9}$$

Figure 4e shows the distribution generated for α_s .

4.1.6. Average Temperature Deviation (θ_0)

The average temperature deviation is assigned a Normal/Gaussian distribution with $\mu = -0.1 \text{ }^\circ\text{C}$ and $\sigma = 0.2 \text{ }^\circ\text{C}$.

The following steps were used to generate the normal distribution:

1. Make a draw (z) from the standard normal distribution with mean = 0 and standard deviation = 1.
2. Required value (E) is calculated by using the ' z ' value as input to Eq. (10).

$$E = \mu + \sigma.z \tag{10}$$

Figure 4f shows the distribution generated for θ_0 .

4.1.7. Effect of Cyclic Temperature Variation (Δ)

The effect of cyclic temperature is assigned an Arcsine (U distribution) with limits $a = -0.5 \text{ }^\circ\text{C}$ and $b = 0.5 \text{ }^\circ\text{C}$.

The following steps were used to generate the Arcsine distribution:

1. Make a draw (r) from the standard rectangular distribution with $a = 0$ and $b = 1$.
2. Required value (E) is calculated by using the ' r ' value as input to Eq. (11).

$$E = \frac{a+b}{2} + \frac{b-a}{2} \sin(2\pi r) \quad (11)$$

Figure 4g shows the distribution generated for Δ

4.1.8. Difference in Expansion Coefficients ($\delta\alpha$)

The Difference in Expansion Coefficients is assigned a CTrap distribution with parameters:

$$a = -1 * 10^{-6} \text{ } ^\circ\text{C}^{-1}, b = 1 * 10^{-6} \text{ } ^\circ\text{C}^{-1} \text{ and } d = 0.1 * 10^{-6} \text{ } ^\circ\text{C}^{-1}.$$

The following steps were used to generate the CTrap distribution:

1. Make 2 draws (r_1 and r_2) independently, from the standard rectangular distribution with $a = 0$ and $b = 1$.
2. Calculate the values a_s and b_s , using Eq. (12).

$$a_s = (a - d) + 2dr_1, b_s = (a + b) - a_s \quad (12)$$

Required value (E) is calculated by using the above calculated values as inputs to Eq. (13)

$$E = a_s + (b_s - a_s)r_2 \quad (13)$$

Figure 4(h) shows the distribution generated for $\delta\alpha$.

4.1.9. Difference in Temperatures ($\delta\theta$)

The Difference in Temperatures is assigned a CTrap distribution with parameters $a = -0.050 \text{ } ^\circ\text{C}$, $b = 0.050 \text{ } ^\circ\text{C}$ and $d = 0.025 \text{ } ^\circ\text{C}$.

Figure 4i shows the distribution generated for $\delta\theta$.

The total execution time for 10^3 values in each distribution, was 0.053 s for the MT and 0.039 s for the WHG whereas for 10^6 values, it was 7.241 s for the MT and 27.04 s for the WHG. Since the time taken by WHG is much higher than MT with almost identical results, we can further establish on the fact that MT is the most suitable PRNG for simulations.

4.2. Application of Monte Carlo Simulation

Values randomly sampled from the generated distributions were used along with Eq. (7) to conduct trials of the Monte Carlo Simulation. Each distribution used for sampling had 10^6 generated values.

The Fig. 5 represents the probability distribution of all the possible values of difference in length acquired through Eq. (7), with N random trials of the Monte Carlo Simulation, where $N = 10^2$ – 10^6 , respectively, using the Mersenne Twister.

Table 4 compares the time taken by the MT and WH generators for generation of the distributions, for number of values $N = 10^3$ and 10^6 .

As shown in Fig. 5, when the number trials are low, Monte Carlo Simulation is unable to cover all possible outcomes, and thus an irregular and ill-defined distribution is obtained, like in Fig. 5, b. To overcome this issue, the Adaptive Monte Carlo Simulation can be used. The Adaptive Monte Carlo procedure involves carrying out an increasing number of trials while checking the results for statistical stability after each complete iteration. On the other hand, the Adaptive procedure leads to significant increase in the complexity of the underlying code used for implementation.

As the number of trials approaches 10^4 and 10^5 in Fig. 5c, d, the distribution becomes more thorough and starts to gradually smoothen and take the desired bell-shape. When the number trials performed is 10^6 , as shown in Fig. 5e, the distribution becomes very well defined and appears to be almost perfectly normal, thus for this task, 10^6 trials are considered to be ideal to efficiently estimate uncertainty. The distributions generated by the WHG were identical to that produced by the MT and hence have not been shown. The mean and standard deviation values calculated for MT and WHG have been compared to the standard result in JCGM GUM 101 in Table 5.

Table 5 contains the mean and standard deviation values for 10^6 trials of Monte Carlo Simulation, for MT as well as WHG, along with the result mentioned in the JCGM GUM 101. The value for the GUM (GUF) has been adopted from the JCGM 101, Experiment 9.5. Both the mean as well as standard deviation produced by the PRNGs are very close to the expected values [17], hence we can conclude the trials were conducted successfully.

The JCGM GUM 101 specifies that if the only available information of a random variable is an expected value x and an associated standard uncertainty $u(x)$ then according to the principle of maximum entropy, the best available estimate of the probability density function $f(x)$ is a Gaussian probability distribution such that $f(x)$ is approximately equal to $N(x, u^2(x))$ unless more detailed information is obtained from a Monte Carlo simulation. If this were to be true, the probability distribution generated should be identical to a normal distribution having the same mean and standard deviation value as the acquired distribution. This comparison is done in Fig. 6, where we generated a normal distribution with mean = 838.028 nm and standard deviation = 35.832 nm (from Table 5), and compared it to the distribution received from the simulation.

As it is evident from Fig. 6, the two distributions are in fact not identical. Thus, the probability distribution generated by the Monte Carlo trials is not normal, contrary to as illustrated by the GUM, but appears to depend on some other parameter apart from the mean and standard deviation.

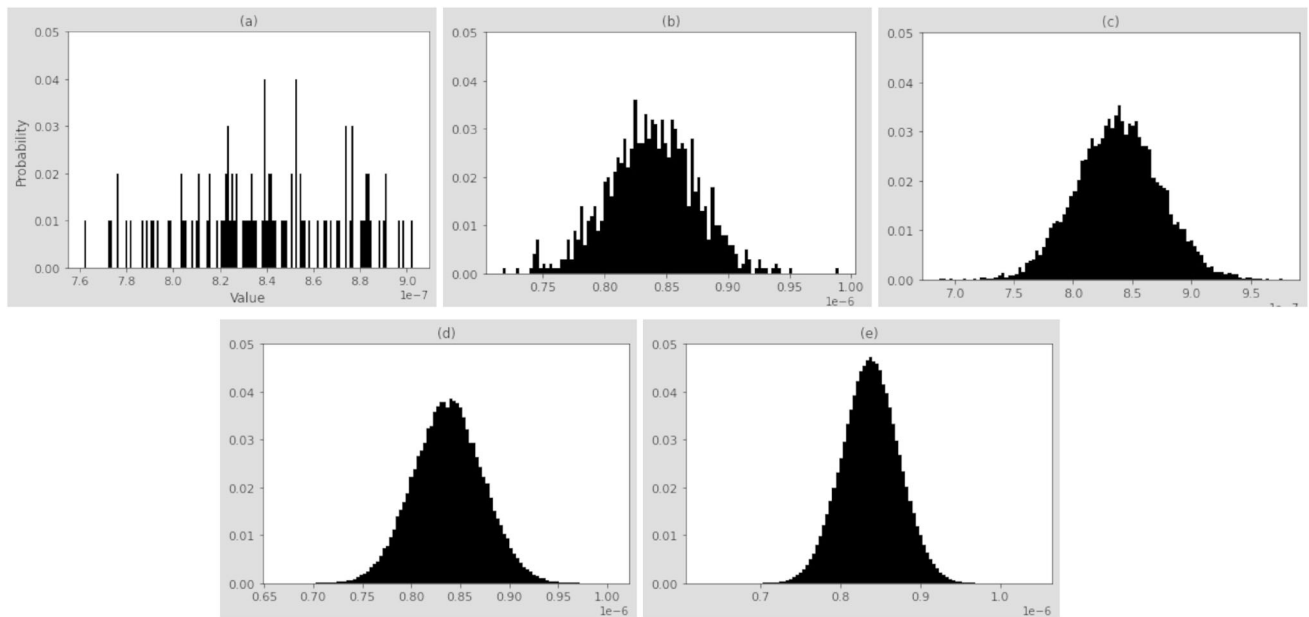


Fig. 5 Monte Carlo Simulation for estimation of uncertainty in length measurement using Mersenne Twister, with number of trials, N equal to 10^2 , 10^3 , 10^4 , 10^5 and 10^6 , respectively

Table 4 Comparing the execution time for MT and WHG for N number of generated values

Parameter	Dist	N	MT (sec)	WHG (sec)
L_s	t -dist ($\mu = 50,000,623$ nm $\sigma = 25$ nm, $z = 18$)	10^3	0.019	0.020
		10^6	1.389	4.791
D	t -dist ($\mu = 215$ nm $\sigma = 6$ nm, $z = 24$)	10^3	0.005	0.003
		10^6	1.429	4.888
d_1	t -dist ($\mu = 0$ nm, $\sigma = 4$ nm, $z = 5$)	10^3	0.005	0.004
		10^6	1.394	4.187
d_2	t -dist ($\mu = 0$ nm $\sigma = 7$ nm, $z = 8$)	10^3	0.007	0.004
		10^6	1.417	4.277
α_s	Uniform ($a = 9.5 \cdot 10^{-6} \text{ }^\circ\text{C}^{-1}$ $b = 13.5 \cdot 10^{-6} \text{ }^\circ\text{C}^{-1}$.)	10^3	0.001	0.001
		10^6	0.249	0.645
θ_0	Normal ($\mu = -0.1 \text{ }^\circ\text{C}$, $\sigma = 0.2 \text{ }^\circ\text{C}$.)	10^3	0.001	0.002
		10^6	0.031	2.221
Δ	Arcsine ($a = -0.5 \text{ }^\circ\text{C}$, $b = 0.5 \text{ }^\circ\text{C}$)	10^3	0.001	0.001
		10^6	0.461	0.726
$\delta\alpha$	CTrap ($a = -1 \cdot 10^{-6} \text{ }^\circ\text{C}^{-1}$, $b = 1 \cdot 10^{-6} \text{ }^\circ\text{C}^{-1}$, $d = 0.1 \cdot 10^{-6} \text{ }^\circ\text{C}^{-1}$)	10^3	0.013	0.002
		10^6	0.433	2.515
$\delta\theta$	CTrap ($a = -0.050 \text{ }^\circ\text{C}$, $b = 0.050 \text{ }^\circ\text{C}$, $d = 0.025 \text{ }^\circ\text{C}$.)	10^3	0.001	0.002
		10^6	0.438	2.790

Table 5 Comparing Mean and Standard Deviation given in the JCGM GUM 101, with those generated by the MT and WHG

Source	Mean (nm)	Standard deviation (nm)
GUM (GUF)	838	32
GUM (MCM)	838	36
MT	838.028	35.832
WHG	838.174	35.841

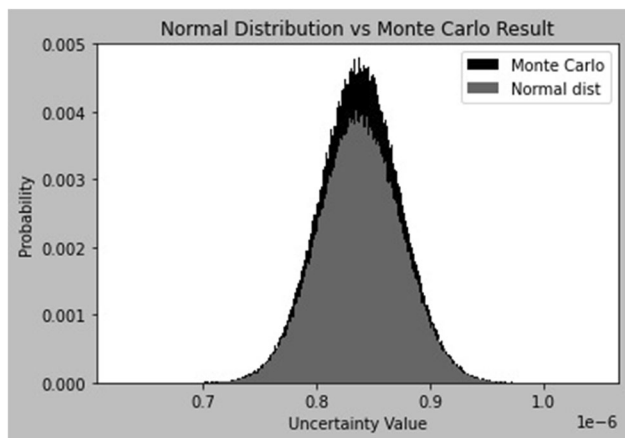


Fig. 6 Comparing the distribution generated by the Monte Carlo trials to a normal distribution with identical mean and standard deviation

We assume the resultant distribution to be a t-distribution, with the aforementioned values for mean and standard deviation, and an unknown number of degrees of freedom. To calculate the degrees of freedom of the obtained curve, we use two methods, the Welch-Satterthwaite formula and the trials-and-errors method.

4.2.1. Welch-Satterthwaite Method

The Welch-Satterthwaite equation is used to calculate the effective degrees of freedom for a linear combination of independent variances. The effective degrees of freedom (v_{eff}) calculated by this equation is valid only if the involved factors are completely independent. Welch-Satterthwaite is the recommended formula in the JCGM GUM 100 to calculate the effective degrees of freedom for any linear problem. The formula is shown in Eq. (14).

$$v_{eff} = \frac{u_c^4(y)}{\sum_{i=1}^N \frac{u_i^4(y)}{v_i}} \tag{14}$$

where $u_c(y)$ is the combined standard uncertainty, $u_i(y)$ is the product of the sensitivity coefficient, and standard uncertainty of the individual components in the linear equation and v_i are the degrees of freedom associated with the components, where the number of components is N . $u_c^2(y)$ can be calculated as the sum of squares of the

standard uncertainties of all the individual components as shown in Eq. (15).

$$u_c^2(y) = u_0^2(y) + u_1^2(y) + \dots + u_N^2(y) \tag{15}$$

For application of Welch-Satterthwaite in gauge uncertainty estimation, the associated standard uncertainty components as well as the degrees of freedom to be used, are present in the JCGM GUM 100 and have also been shown in Table 6 for convenience. In case no degrees of freedom are associated with a variance component, for example the components θ and α_s in Table 6, the value of v_i can be taken as infinity.

Since $d = D + d1 + d2$, as shown in Eqs. (6) and (7), the standard uncertainty for d can be calculated using the Welch-Satterthwaite formula as shown in Eq. (14). The value of $u_c^2(d)$ can be calculated using Eq. (15). Thus, using the Table 6 and Eqs. (14) and (15), the effective degrees of freedom for d comes out to be 25.6.

The value of $u_c^2(\delta L)$ to be used in the equation can be calculated with Eq. (15), using the values of $u_i(l)$ listed in Table 6. After the application, $u_c^2(\delta L)$ comes out to be 1002 nm^2 , therefore the value of $u_c(\delta L)$ is approximately equal to 32 nm.

Using the $u_c(\delta L)$ calculated above and the uncertainties and degrees of freedom listed in Table 6, we applied the Welch-Satterthwaite equation, as shown in Eq. (16).

$$v_{eff} = \frac{32^4}{\frac{25^4}{18} + \frac{9.7^4}{25.6} + \frac{2.9^4}{50} + \frac{16.6^4}{2}} = 16.7 \approx 16 \tag{16}$$

Therefore, the effective degrees of freedom calculated using the Welch-Satterthwaite equation for the gauge length uncertainty estimation is 16.

Figure 7, compares a normal distribution with mean = 838.028 nm and standard deviation = 35.832 nm (from Table 5), the distribution obtained from the Monte Carlo Simulation and a Students t -distribution with the aforementioned mean and standard deviation and degrees of freedom equal to 16.

As it is clearly evident from the Fig. 7, 16 is not the apt value for degrees of freedom in this case, as the students t -distribution with $v_{eff}=16$ is not identical to the distribution obtained from the Monte Carlo Simulation.

Table 6 Standard uncertainty and degrees of freedom for all components of gauge uncertainty estimation

Standard uncertainty component	Value of standard uncertainty	c_i	$u_i(l)$ (nm)	Degrees of freedom
L_s	25 nm	1	25	18
d	9.7 nm	1	9.7	25.6
D	5.8 nm			24
d_1	3.9 nm			5
d_2	6.7 nm			8
α_s	1.2E-6 °C ⁻¹	0	0	–
θ	0.41 °C	0	0	–
θ_0	0.2 °C			
Δ	0.35 °C			
$\delta\alpha$	0.58E-6 °C ⁻¹	$-1_s \theta$	2.9	50
$\delta \theta$	0.029 °C	$-1_s \alpha_s$	16.6	2

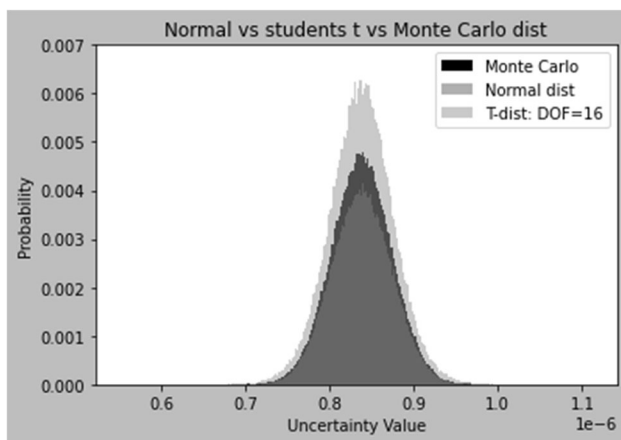


Fig. 7 Comparing a normal distribution, a students t-distribution with degrees of freedom = 16 and the Monte Carlo distribution, having the same mean and standard deviation

4.2.2. Trial and Error Method

As the result of the Welch–Satterthwaite equation was not accurate, we tried the trials and error method to determine the degrees of freedom of the Monte Carlo distribution, and determined that the acquired result was indeed a *t*-distribution with 46 degrees of freedom. Figure 8a compares the distributions for varying numbers of degrees of freedom, and Fig. 8b shows the Monte Carlo Distribution along with the final result i.e., when number of degrees of freedom, = ν 46 and the result of the WS-z approach discussed later, with degrees of freedom, = ν 16.

4.2.3. Beyond *t*-Interval Method for Uncertainty Estimation

It has been previously illustrated by Huang [25]26 that for small sample size, the conventional uncertainty estimation using the *t*-interval method gives rise to paradoxes such as

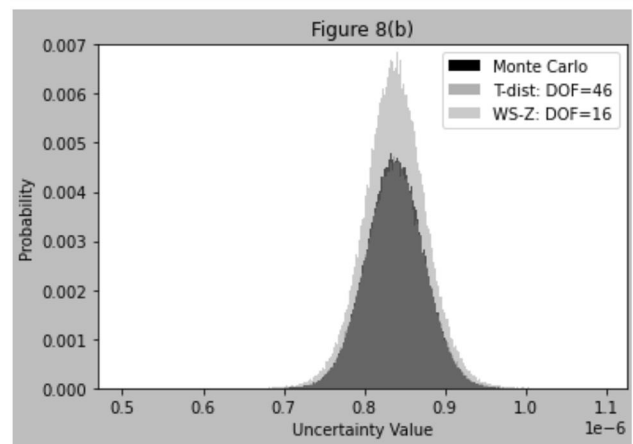
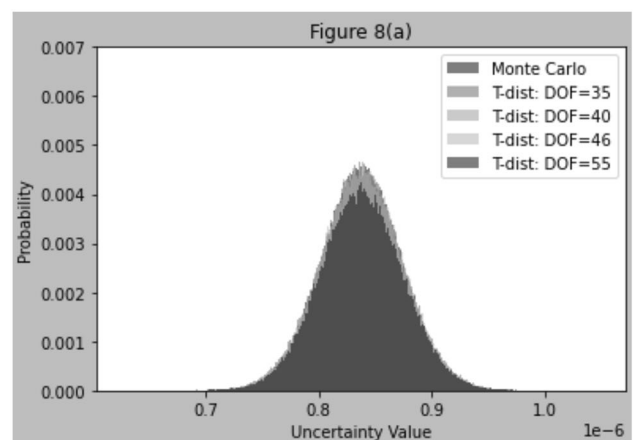


Fig. 8 Comparing the distribution generated by Monte Carlo Simulation to *t*-distributions with varying degrees of freedom and the result of the WS-*t* formula

the Du-Yang paradox [28], the Ballico paradox [27] and the uncertainty paradox [28]. The *t*-based uncertainty has large precision as well as bias errors when the number of observations is small [29]. As a solution to the problem,

Huang [25] proposed uncertainty estimation using a mean-unbiased estimator, now known as the WS-z method.

According to the WS-t approach, the expanded uncertainty (U_t) is calculated using the Eq. (17).

$$U_t = t_{95,v} \cdot u_c \quad (17)$$

where $t_{95,v}$ is the t -value for 95% coverage with v degrees of freedom, and u_c is the standard uncertainty. On the other hand, in the proposed WS-z approach, the expanded uncertainty is corrected by an unbiased estimator which can be accomplished using two different methods, namely (i) the Median-Unbiased and (ii) the Mean-Unbiased methods. According to the Median-Unbiased approach, the expanded uncertainty is given using Eq. (18).

$$U_{Zmed} = z_{95} \cdot C_{med} \cdot u_c \quad (18)$$

where z_{95} is the z -value for 95% coverage, u_c is the standard uncertainty, and C_{med} is Median-Unbiased estimator calculated using Eq. (19).

$$C_{med} = \left(1 - 0.0167e^{-0.9(v-1)}\right)^{-1} \cdot \left(1 - \frac{2}{9v}\right)^{-1.5} \quad (19)$$

In the Mean-Unbiased approach, the expanded uncertainty is given using Eq. (20).

$$U_{zmean} = \frac{z_{95}}{c_{4,v}} \cdot u_c \quad (20)$$

where z_{95} is the z -value for 95% coverage, u_c is the standard uncertainty, and $C_{4,v}$ is bias correction factor for u_c calculated using Eq. (21).

$$c_{4,v} = \sqrt{\frac{2}{v}} \cdot \Gamma\left(\frac{v+1}{2}\right) / \Gamma\left(\frac{v}{2}\right) \quad (21)$$

where v is the degrees of freedom, and Γ is the gamma function.

Following these set of equations, we calculated the expanded uncertainty for the above-mentioned length uncertainty problem. With degrees of freedom estimated as 16, from the WS equation, the WS-t approach yielded the U_t as 55.87 nm, while the WS-z approach using the median and mean unbiased schemes yielded 63.47 nm and 63.71 nm, respectively. The resulting distribution generated with degrees of freedom equal to 16 i.e., the result of the WS-z approach, has also been plotted on Fig. 8b for comparison with the distribution obtained from the Monte Carlo procedure and the one obtained as a result of the trial-and-error method. The trial-and-error method still remains to be the most accurate for this case study.

Interestingly, the U_t obtained from the Monte Carlo with degrees of freedom being estimated as 46 was found to be 60.16 nm. The latter result being much consistent with the WS-z approach indicates that much of the bias is inherently

corrected by the Monte Carlo method. Partly, this may be due to the large number of model measurements one can accomplish by the MC method, thereby tending closer to the z -statistical description of the measurement data. However, not be complacent of the results, it certainly requires detailed investigation to check whether a simple estimate of DOF using a probabilistic fit would suffice sufficient information on the uncertainty associated with the measurements.

5. Summary and Conclusion

In this paper, we tested three of the most widely used PRNGs for non-cryptographic purposes, Mersenne Twister, Linear Congruential Generator and Wichmann–Hill Generator. As observed, Mersenne Twister stood out as the most efficient of the three, passing all the tests comfortably.

It is preferred over the LC generator and WH generator as:

1. MT is the fastest for the generation of a large number of values [3], 15
2. It is the most uniform of all the considered PRNGs, as it was the only one able to pass all the test cases of the Chi-Square Test for Uniformity [5]
3. The values produced were almost completely free of correlation as each of the coefficient of correlation was very close to zero [13].
4. The generated sequence was independent and free of intra-sequence dependencies as the MT comfortably passed the Autocorrelation-Test, whereas WH generator failed one of the test cases.
5. MT generated values that were not prone to patterns, as portrayed by its excellent performance in the Runs Test. The LC generator was unable to pass all the test cases for this test [6], whereas the WH generator performed well.
6. MT was the only PRNG that was able to achieve the expected Frequency Distribution of Values, in the Value Distribution Analysis. Both the LC generator and the WH generator showed irregularities and abrupt changes in their respective graphs, but as the value of 'n' was increased, Mersenne Twister showed a distribution increasingly resembling that of a Normal Distribution [12]

Thus, Mersenne Twister is the favoured PRNG for use in simulations and other related tasks, conforming to the conclusions made in [3], 14.

Further, the Mersenne Twister and Wichmann–Hill Generator were used to generate random values for the application of Monte Carlo Simulation to determine the uncertainty in the measurement of the length of an end

gauge. It was observed that for 1 million random trials of the Monte Carlo Simulation, a very thorough frequency distribution, spanning through all the possible values of length was achieved, thus enabling us to efficiently quantify the uncertainty associated with this particular measurement. Note that the WH generator also produced comparable results but was rendered unfeasible due to its higher execution time.

The case study of evaluation of uncertainty in measurement of length of an end gauge, yielded a probability distribution with mean of 838.028 nm and standard deviation of 35.832 nm, aligning with the values mentioned in the GUM 101 documentation. The alternate uncertainty estimation approach, the GUM approach, gives a similar result, but requires calculation of partial derivatives and effective degrees of freedom, which makes it unsuitable for some tasks.

The distribution acquired as a result of 10^6 trials of the Monte Carlo Simulation was assumed to be normal, but this assumption proved inconsistent when compared with a normal distribution generated using the same mean and standard deviation. A third parameter, namely the degrees of freedom was assumed to be of significance. Using the Welch–Satterthwaite equation, the effective degrees of freedom were calculated to be 16. Apart from the process required for the application of the Welch–Satterthwaite formula being cumbersome when the number of involved components are relatively high, the result obtained i.e., 16 effective degrees of freedom were not an adequate fit for gauge block calibration. As an alternative to the Welch–Satterthwaite equation, we applied the more convenient and a simple method based on trial and error. With extensive fit procedures checking, it was found that the curve resulting from the Monte Carlo Simulation was in fact a t -distribution with 46 degrees of freedom for this particular case study.

However, as pointed out in literature, the use of t -intervals to account for the paradoxes in uncertainty measurements, calculations were extended to techniques that accounts for the corrections. One such method is that of z -intervals-based corrections, originally proposed by Huang. Using the WS- z method, it was found that the expanded uncertainty amounts to 63.47 (63.71) nm in the median (mean) unbiased method, respectively, contrary to the value 55.87 estimated using the WS- t method. Interestingly, we find that the WS- z method agrees well with the estimated t -distribution with 46 degrees of freedom obtained using the Monte Carlo method. The latter value was found to be 60.162 nm. Although, the results obtained from the WS- z and Monte Carlo methods appear consistent, further investigations are needed to ascertain the validity of our findings with more examples and detailed investigations. The work is, moreover, an indication that

the nature of the output PDF obtained using the Monte-Carlo methods may be aptly estimated using trial and error fit to determine the expanded uncertainty in the quantification of measurement uncertainty.

References

- [1] The Joint Committee for Guides in Metrology (JCGM) ‘*Evaluation of Measurement Data — Supplement 1 to the “Guide to the Expression of Uncertainty in Measurement” — Propagation of Distributions Using a Monte Carlo Method*’ (2008).
- [2] M. Wei, Y. Zeng, C. Wen, X. Liu, C. Li and S. Xu ‘*Comparison of MCM and GUM Method for Evaluating Measurement Uncertainty of Wind Speed by Pitot Tube*’ MAPAN-J. Metrol. Soc India (2019).
- [3] X. Tian and K. Benkrid ‘*Mersenne Twister Random Number Generation on FPGA, CPU and GPU*’. NASA/ESA Conference on Adaptive Hardware and Systems (2009).
- [4] B. Wichmann and D. Hill ‘*Algorithm AS183: An Efficient and Portable Pseudo-Random Number Generator*’ Journal of the Royal Statistical Society (1982).
- [5] M. Matsumoto and T. Nishimura ‘*Mersenne Twister: a 623-Dimensionally Equidistributed Uniform Pseudo-Random Number Generator*’. ACM Transactions on Modelling and Computer Simulation (1997).
- [6] S. Sinha, SKH. Islam and MS. Obaidat. ‘*A Comparative Study and Analysis of some Pseudorandom Number Generator Algorithms*’ Security and Privacy (2018).
- [7] A. Roy and T. Gamage. ‘*Recent Advances in Pseudorandom Number Generation*’ (2007).
- [8] K. Marton, A. Suci, C. Sacarea and O. Cret. ‘*Generation and Testing of Random Numbers for Cryptographic Applications*’. Proceedings of the Romanian Academy, volume 13 (2012).
- [9] V. Ristovka and V. Bakeva. ‘*Comparison of the Results Obtained by Pseudo Random Number Generator based on Irrational Numbers*’ International Scientific Journal “Mathematical Modeling”.
- [10] IIT Kanpur (online) ‘*Autocorrelation*’. Available at: <http://home.iitk.ac.in/~shalab/econometrics/Chapter9-Econometrics-Autocorrelation.pdf>
- [11] P. Burgoine ‘*Testing of Random Number Generators*’ University of Leeds. <http://www1.maths.leeds.ac.uk/~voss/projects/2012-RNG/Burgoine.pdf>.
- [12] C.M. Grinstead and J.L. Snell ‘*Introduction to Probability*’. Chapter 7 (1998).
- [13] K.N. Koide ‘*Analysis of Random Generators in Monte Carlo Simulation: Mersenne Twister and Sobol*’.
- [14] A. Gaeini, A. Mirghadri, G. Jandaghi and B. Keshavarzi ‘*Comparing Some Pseudo-Random Number Generators and Cryptography Algorithms Using a General Evaluation Pattern*’. Modern Education and Computer Science Press (2016).
- [15] F. Sepehri, M. Hajivaliei and H. Rajabi ‘*Selection of Random Number Generators in GATE Monte Carlo Toolkit*’. Nuclear Instruments and Methods in Physics Research Section A: Accelerators, Spectrometers, Detectors and Associated Equipment (2020).
- [16] S. Wang, X. Ding, D. Zhu, H. Yu and H. Wang ‘*Measurement Uncertainty Evaluation in Whiplash Test Model Via Neural Network and Support Vector Machine-Based Monte Carlo Method*’. Measurement 119. (2018).
- [17] A.Y. Mohammed and A.G. Jedrzejczyk ‘*Estimating the Uncertainty of Discharge Coefficient Predicted for Oblique Side*

- Weir Using Monte Carlo Method*' Flow Measurement and Instrumentation 73 (2020).
- [18] H. Huang 'Why the Scaled and Shifted t -distribution Should not be used in the Monte Carlo Method for Estimating Measurement Uncertainty?' Measurement 136 (2019).
- [19] A. Doğanaksoy, F. Sulak, M. Uğuz, O. Şeker, and Z. Akcengiz 'New Statistical Randomness Test Based on Length of Runs' Hindawi Article 626408 (2015).
- [20] J.H. Zar 'Biostatistical Analysis-5th Edition' Prentice-Hall, Inc. Division of Simon and Schuster One Lake Street Upper Saddle River, NJ United States (2007).
- [21] The Joint Committee for Guides in Metrology (JCGM) 'Evaluation of Measurement Data — Guide to the Expression of Uncertainty in Measurement' (2008).
- [22] R. Willink 'A Generalization of the Welch-Satterthwaite Formula for Use with Correlated Uncertainty Components' Metrologia 44(5):340 (2007).
- [23] R. Kacker, B. Toman and D. Huang 'Comparison of ISO-GUM, Draft GUM Supplement 1 and Bayesian Statistics Using Simple Linear Calibration' Metrologia 43 S167 (2006).
- [24] R. Kacker 'Bayesian Alternative to the ISO-GUM's Use of the Welch-Satterthwaite Formula' Metrologia 43 1 (2005).
- [25] H. Huang 'Uncertainty-Based Measurement Quality Control' Accred Qual Assur 19(2):65–73 (2014).
- [26] H. Huang 'On the Welch-Satterthwaite Formula for Uncertainty Estimation: A Paradox and Its Resolution' Cal Lab Magazine, October – December (2016).
- [27] M. Ballico 'Limitations of the Welch-Satterthwaite Approximation for Measurement Uncertainty Calculations' Metrologia (2000).
- [28] H. Huang 'A Paradox in Measurement Uncertainty Analysis' Measurement Science Conference, Pasadena (2010).
- [29] J.D. Jenkins 'The Student's t -distribution Uncovered' Measurement Science Conference (2007).

Publisher's Note Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.