## Answers

1.

The flawed pseudo-random number generator from the C standard library was investigated to reveal a known challenge of non-randomness. Random (x, y, z) positions with coordinates between 0 and $2^{31}$ – the maximum random integer value in the standard library– had been previously generated via test_broken_libc.py script. File rand_points.txt contained about 5% of the total span of (x, y, z) triples with $0 < x, y, z < 10^8$. Given the type of PRNG utilized in this library was well-recognized for its introduction of correlations between sequential points, a relatively small number of planes, where sets of points in n-dimensional space would lie, were anticipated.

The data points were plotted in 3D to detect the set of planes (Figure 1). Several azimuth values were tried in order to attain a reasonably clear view of the patterning before an optimal view perspective was chosen. The inherent non-randomness became evident. Subsequently, several linear combinations $z = ax + by$ were constructed for integer values of a=[-2, 2] and b =[-2, 2]. According to Figure 2, the best combinations of (a, b) were determined to be (-2, 1) and (2, - 1). Figure 3 shows the detail view of the planarity of triples in the 3D space, which is analyzed in the 2D space to corroborate the presence of non-randomness, where a=2 and b=-1. Alternatively, linear relationships based on a=-2 and b=1 could as well represent the correlations between sequential points (Figure 2).

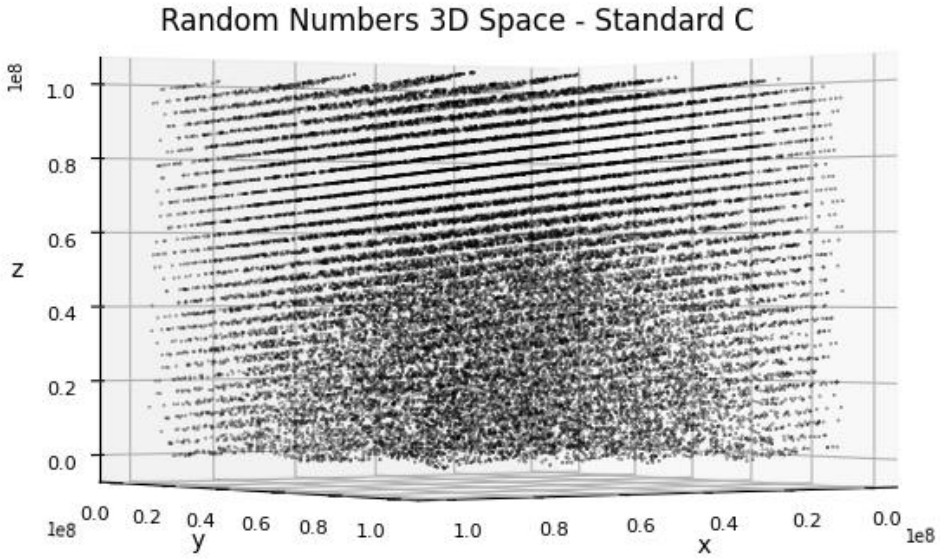


**Figure 1.** Random triples generated as per the standard C library

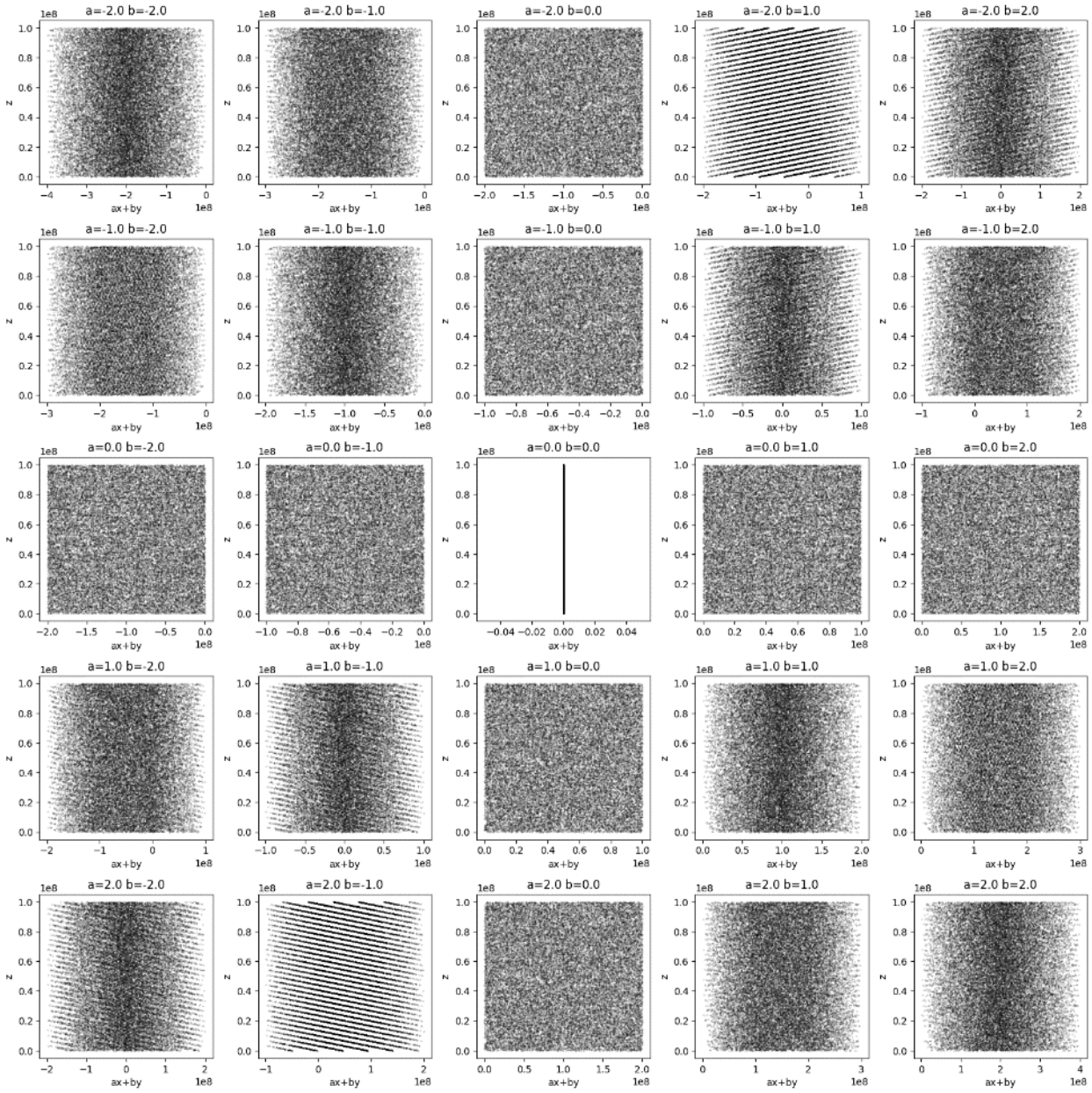


**Figure 2.** Various ax+by combinations for a and b from [-2, 2]

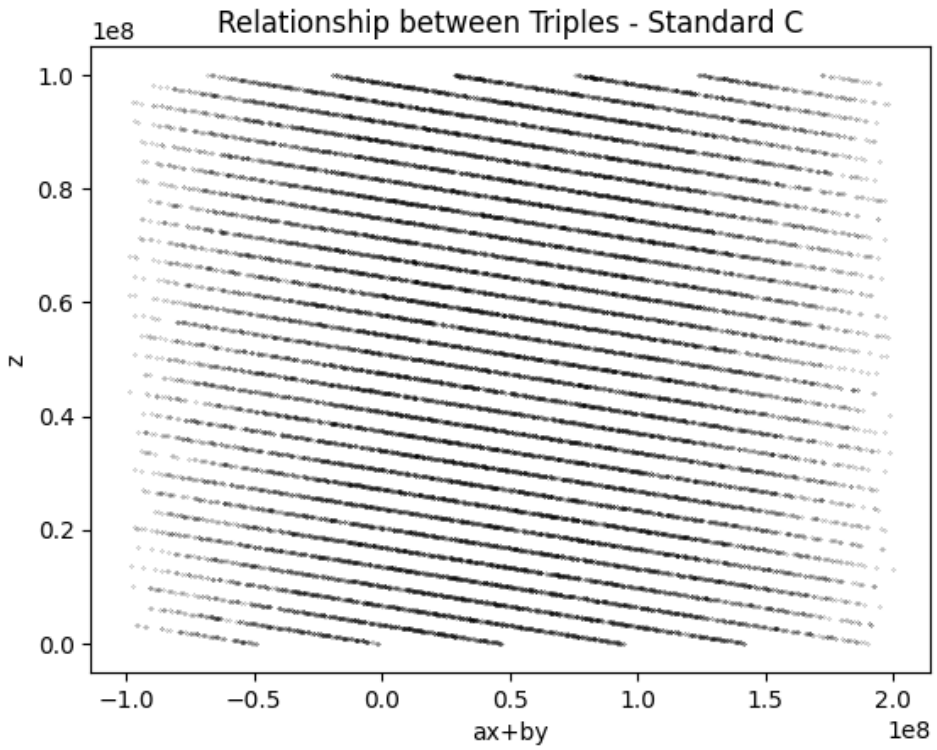


**Figure 3.** Presence of dependence of z on -2x+y based on the standard C library

Python's random number generator was thereafter tested. Despite viewing about the axes based on several azimuth values, no apparent pattern due to correlations between sequential points could be determined. Hence, the non-randomness observed in the standard C library was not found herein. The random distribution of the generated numbers is shown in Figure 4.

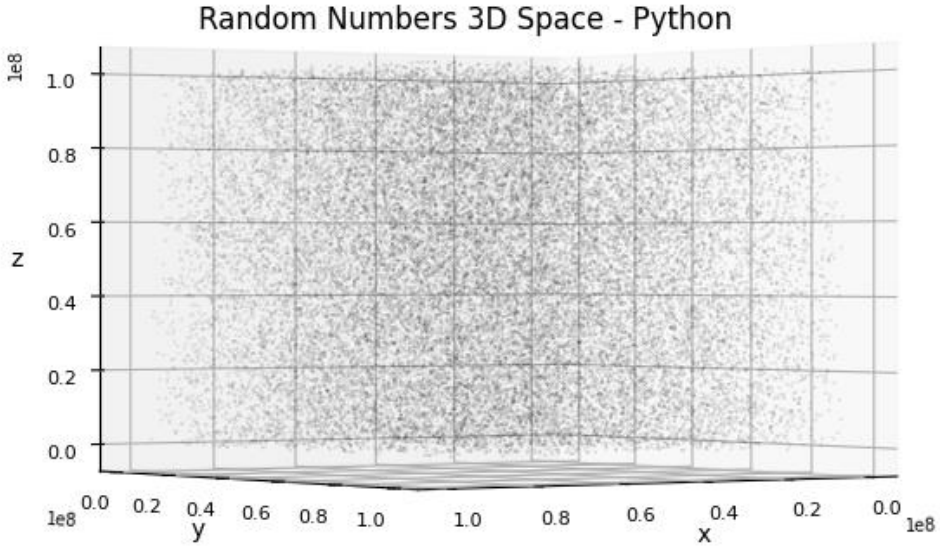


**Figure 4.** Random triples generated by python random.ranindt()

The presence of the effect was investigated based on the local machine C standard library. In the Windows platform, RAND_MAX was identified as $32767 = 2^{15} - 1$. A representation of the population of datapoints similar to those generated by test_broken_libc.py would thus not be possible given the RAND_MAX constraint, which was far less than the GNU RAND_MAX of 2147483647. Nonetheless, a 3D plot (Figure 5) was constructed with a smaller set of triples with values $<$ RAND_MAX, which were pseudo-randomly generated over 30,000 numbers. The 2D analysis, also with $a=2$ and $b=-1$, was plotted (Figure 6). As expected, the much smaller RAND_MAX, limited by the local machine's settings, did not demonstrate the correlations between sequential points. On the other hand, C++ library utilizing the Mersenne Twister algorithm, which was not studied in this trial, could provide an alternative and might be more in line with Python's random number generator in terms of performance.

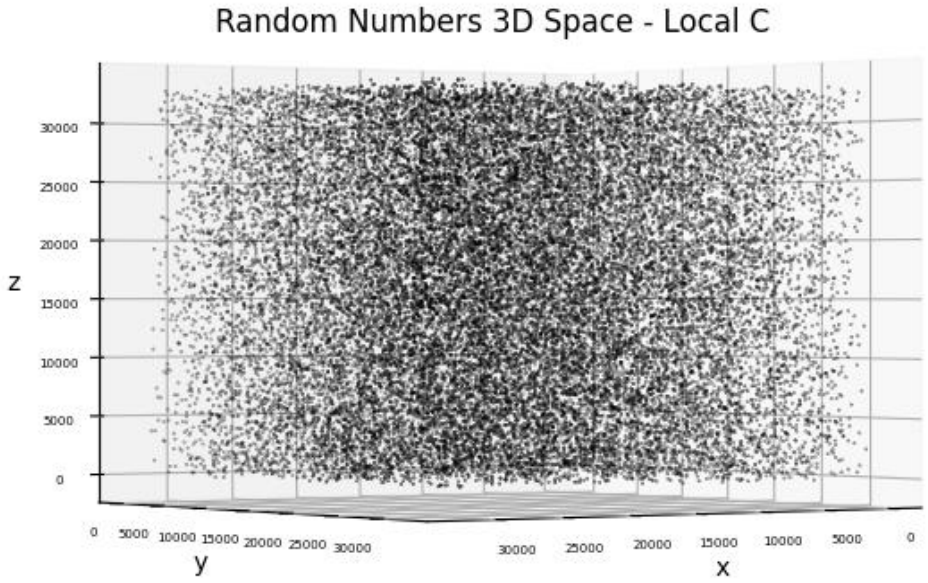


**Figure 5.** Random triples generated as per the standard C library - local

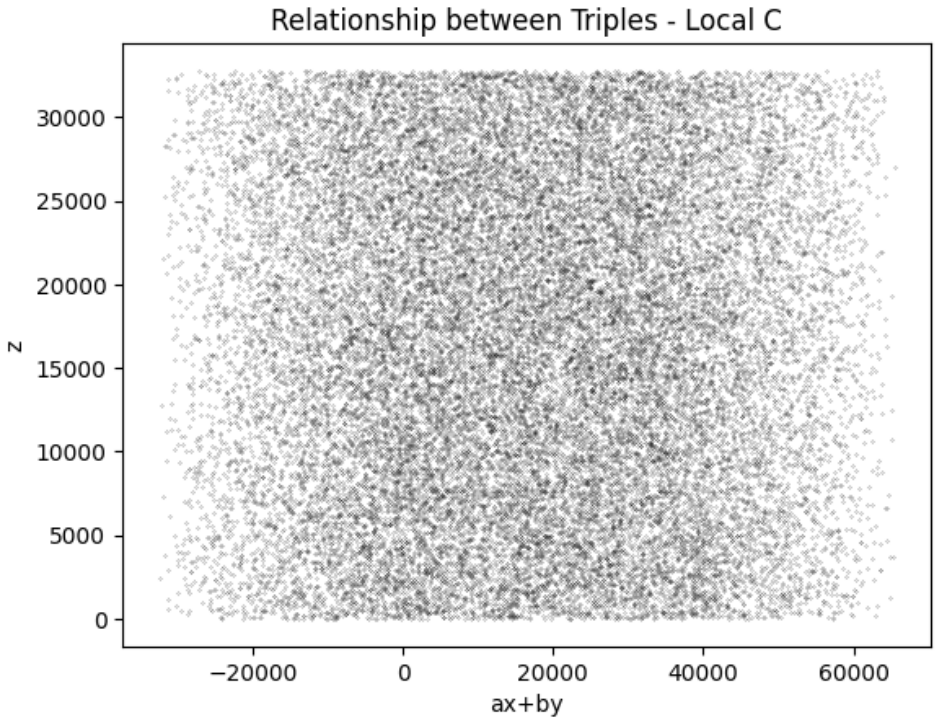


**Figure 6.** Lack of dependence of z on -2x+y based on the standard C library - local

2.

A rejection method to generate exponential deviates from another distribution was studied. In respect of $e^{-x}$, Lorentzian $(x^2+1)^{-1}$, Gaussian $\exp(-x^2/2)$, and power function $(x+1)^{-1.5}$ were considered as bounding options (Figure 7). The method required that a candidate distribution be proximal yet greater than $e^{-x}$ (i.e. positioned in the $+x$ direction such that no cross-over with $e^{-x}$ would ensue). Figure 7 suggested that the Lorentzian might meet this criterion.

The Lorentzian completely bounded $e^{-x}$, and thus was selected for further investigation. The exponential deviates were adopted to be all positive with the cut off of the distribution situated at zero. 3000000 random numbers were generated in [0, 1] and normalized to delimit the Lorentzian domain to only positive values – represented as y from [0.5, 1].

Cumulative distribution function for the Lorentzian, normalized follows:

$$CDF = \pi^{-1}\arctan(x) + \frac{1}{2}$$

whose inverse corresponded to the deviates:

$$x = \tan\left[\pi\left(y - \frac{1}{2}\right)\right].$$

Identification of the random numbers that were less than the ratio $e^{-x}/(x^2+1)^{-1}$ thereafter facilitated the computation of the acceptance rate. Furthermore, the theoretical acceptance rate, $pacc_{th}$ should equal:

$$pacc_{th} = \frac{\int_0^\infty e^{-x}dx}{\int_0^\infty (x^2+1)^{-1}dx} = \frac{2}{\pi} = 63.662\%.$$

A histogram of the deviates was found to be in very good agreement with the expected exponential curve $e^{-x}$ (Figure 8). Fraction of uniform deviates that gave rise to an exponential deviate could be recognized in respect of a calculated acceptance rate. The calculated acceptance rate 63.638% was determined to be very close to $pacc_{th}$.

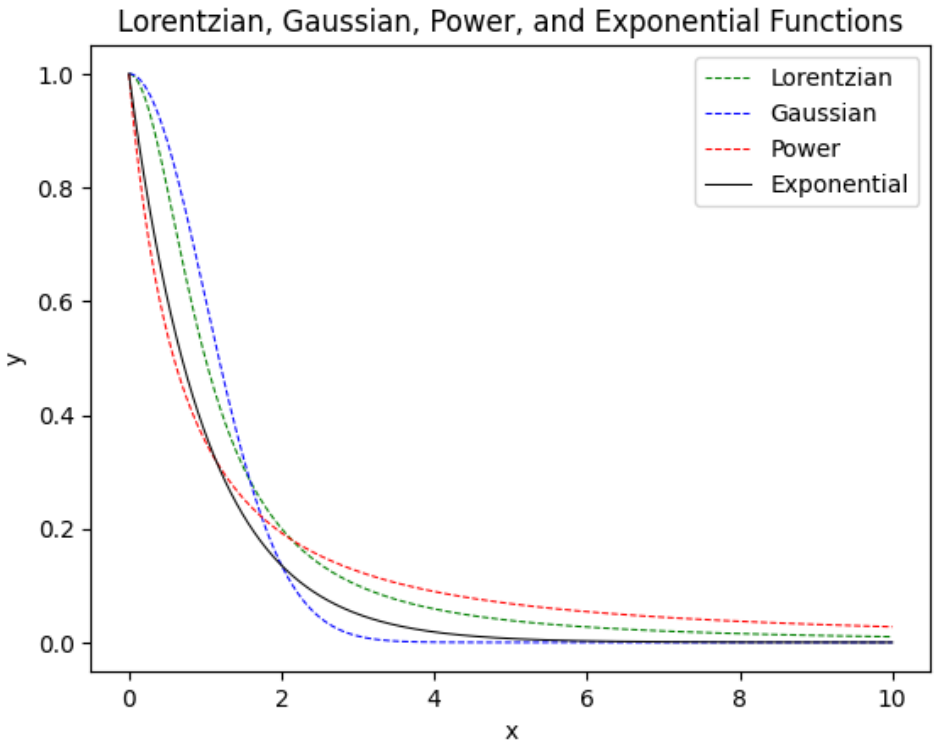


**Figure 7.** Candidate distributions to bound $e^{-x}$

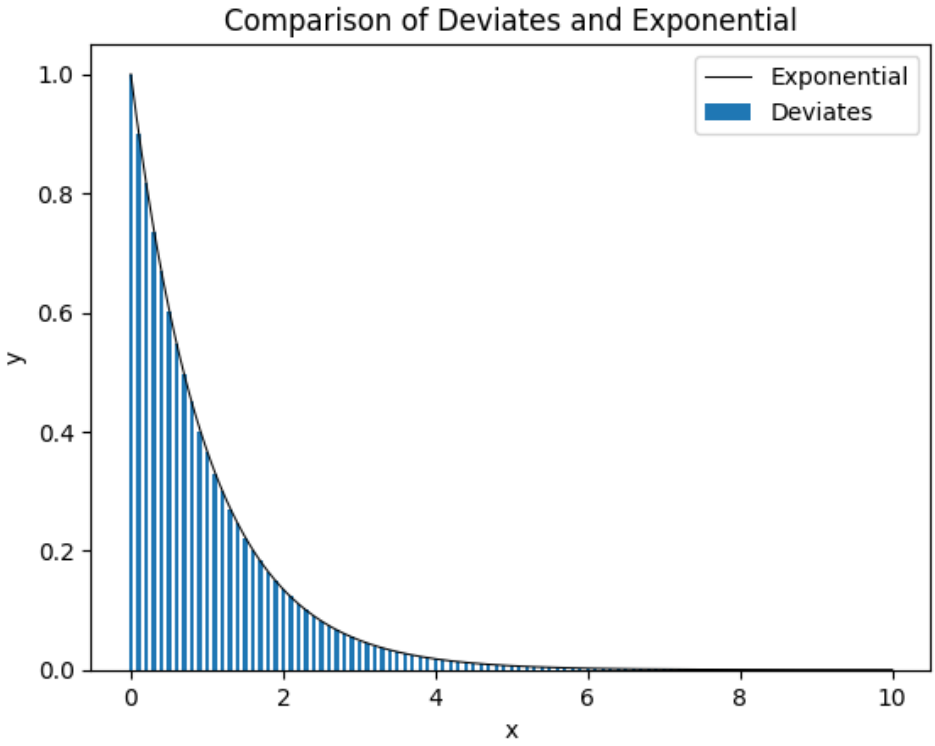


**Figure 8.** Comparison of Lorentzian-originated deviates with $e^{-x}$

3.

Solution to Problem 2 was repeated, alternatively utilizing a ratio-of-uniforms (ROU) generator. For $0 \leq u \leq 1$, the limits on v would need to be set:

$P(x)=e^{-x}$

$P(v/u) = e^{-v/u}$

$0<u<[P(v/u)]^{1/2}$ → $0<u<e^{-v/(2u)}$

$u<e^{-v/(2u)}$

$\ln(u)< -v/(2u)$

$v< -2u \ln(u)$

$\max[-2u \ln(u)]$ is observed for when $d[-2u \ln(u)]/du = 0$.

$-2\ln(u) - 2=0$ → $u=e^{-1}$, which should be substituted in the inequality:

$v< -2e^{-1} \ln(e^{-1})$

$v< 2e^{-1}$

The ROU generator could subsequently be implemented with 3000000 random numbers from [0,1], constituting u and 3000000 random numbers from [0, 2/e], corresponding to v. The acceptance step was based on $u<e^{-v/(2u)}$.

A histogram of the deviates agreed well with the expected exponential curve $e^{-x}$ (Figure 9). An improvement in efficiency was evident. The calculated acceptance rate 67.958%, which was somewhat greater than the acceptance rate computed in Problem 2.

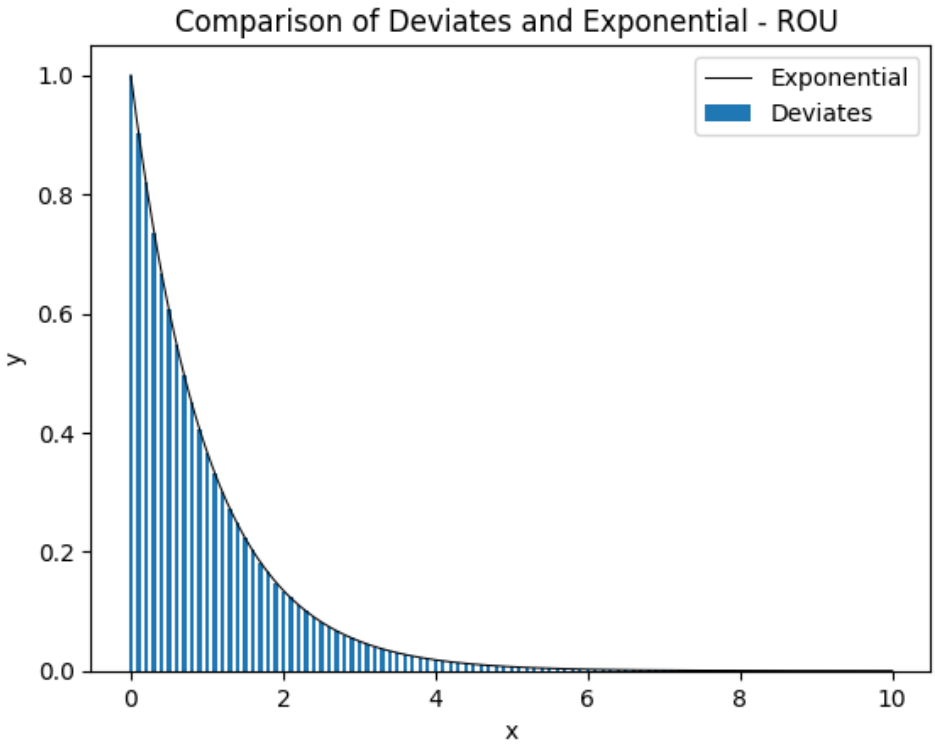


**Figure 9.** Comparison of ROU-generated deviates with $e^{-x}$

## Appendix A: Python Code

Jupyter notebook with relevant Python code and outputs is available at:

https://github.com/ck22512/comp_phys/tree/main/Assignment7