Random numbers and Monte-Carlo methods

D. D. Q. Ng
Level 6 MSci. Laboratory, H. H. Wills Physics Laboratory, University of Bristol, BS8 1TL, UK
(Dated: December 24, 2019)

At the heart of every Monte Carlo (MC) simulations, there lies a reliable pseudo-random number generator (PRNG) for which repeated random sampling can be done to obtain numerical results. Here the choice of Mersenne Twister PRNG within the Python stdlib module "random" is first justified, then used for its advantage in reproducibility and speed of random number generation, in order to investigate random angles generation by different methods. Furthermore, gamma rays detection and generation of pseudo-experiments are simulated by the MC method for which both their statistical results are also investigated.

I. RANDOM ANGLES GENERATION

The most widely used PRNG[1] - named the Mersenne Twister (MT) due to their repetition period being a Mersenne prime - was invented by Matsumoto and Nishimura in 1998[2]. In order to justify the use of the MT PRNG, the main characteristics in terms of random number generation for which Sawilowsky regarded as important for a high quality Monte Carlo simulation will be considered.[3] Firstly, the PRNG should have a long period before repeating themselves, which is fulfilled by the MT since it has a gargantuan period of $2^{19937} - 1$. Secondly, the randomness of MT passes the conventional PRNG tests such as the Diehard[4] and most of TestU01.[6] Hence from here onwards, for the rest of the problems MT will be used as the choice of PRNG.

This section focuses on the pseudo-random number generation using the well known "inverse transform sampling" and "acceptance-rejection" methods. The former first randomly draws samples between the interval [0, 1] from a uniform distribution in order to sample a non-flat distribution, by inputting the uniform samples into the inverse of the non-flat distribution. The latter relies on generating random numbers within an interval for a range $y_1, y_2, ..., y_n$ and domain $x_1, x_2, ..., x_n$ and inputting the domains into a desired function f. By accepting the domain values x_i for which $f(x_i) < y_i$, the "accepted" x values will be a distribution that is biased towards the function. Both methods were performed on a sine function and random angles were generated.

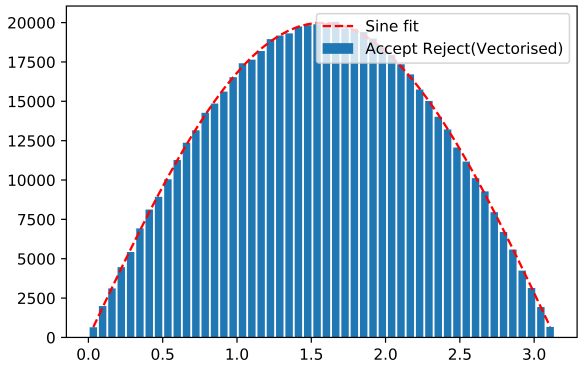

FIG. 1. The histogram of 40000 random sampled points using acceptance-rejection method.

As expected, the reject-accept method would yield a distribution with a lower peak than the analytical method, since most of the points would have been rejected. Therefore for the same amount of random points, the normalised distribution using the reject-accept method would be less similar to a sine curve than the distribution generated using its counterpart. It is worth to then compare the evaluation time of the both methods in how quickly they generate a distribution. A vectorised version of both functions were also implemented using Numpy and compared, which can be seen on Fig. 3. The run time comparison was done using the built-in magic command `%timeit` ipython for convenience, since it computes run time over a fixed number of times and computes

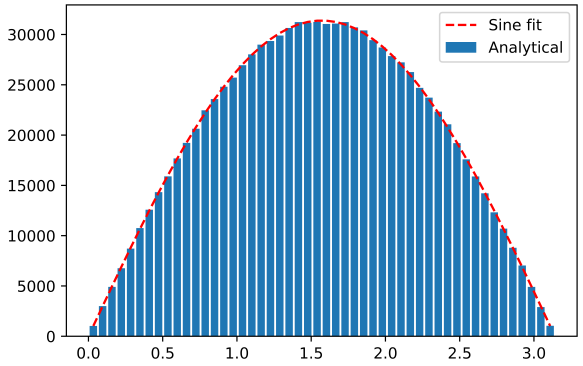

FIG. 2. The histogram of 40000 random sampled points using inverse transform sampling method.

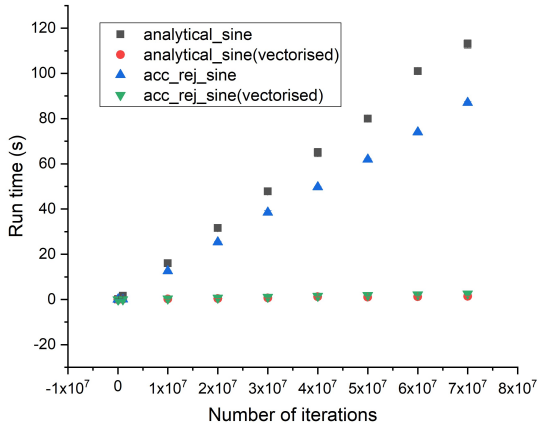

FIG. 3. The run time comparison of four different implementations of random angles generation.

the mean run time and its standard deviation as errors. The generated data were then plotted on Origin. It can be seen that while the non-vectorised functions' run time scales linearly, with the accept-reject being quicker due to points being rejected, they are however substantially slower than their vectorised counterpart. Moreover, the run time difference between both vectorised methods seems to be negligible up to 7×10^7. Therefore ultimately for speed and efficiency one would simply want to use the vectorised analytical function. Due to the run time supremacy that the vectorisation methods have over looped methods, the subsequent tasks will all be implemented with vectorisation.

II. NUCLEI DECAY AND GAMMA RAY DETECTION

Consider a system, as illustrated in Fig. 4, where a beam of unstable nuclei with a mean lifetime of $550\,\mu$s is injected at $2\,$m away from and perpendicularly towards a detector array with resolutions of $10\,$cm and $30\,$cm in x and y respectively. Along its trajectory in the z direction, the nuclei decay spontaneously and gamma rays are emitted isotropically.

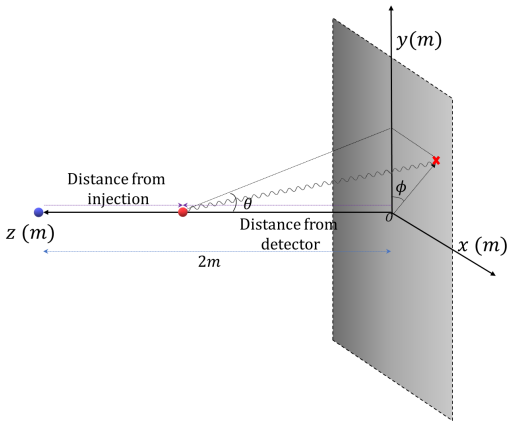

FIG. 4. The setup for the simulation. The blue dot indicates the nuclei injection point. The red dot and cross indicate where it could potentially decay and detected respectively, with the photon's trajectory described by the zig-zag line. The arbitrarily sized detector is illustrated by the grey screen in the x-y plane. Here the only angles of interest are $\phi \in [0, 2\pi)$ and $\theta \in [-\pi/2, \pi/2]$ with ϕ defined positively in the clock-wise direction.

If the simulation was to run by MC means, it is required that the detection on screen must come about randomly and uniformly distributed. Therefore without hindsight, one might think that could be ensured by making both ϕ and θ independent and identically distributed (i.i.d.) from a uniform distribution within the appropriate intervals under the conventional spherical coordinate system, namely:

$$\begin{aligned} x &= \rho \sin\theta \cos\phi \\ y &= \rho \sin\theta \sin\phi \\ z &= \rho \cos\theta \end{aligned} \tag{1}$$

However, this is incorrect because due to the area element $dA = \sin\theta d\theta d\phi$ being proportional to the sine of the elevation angle θ rather than just the angle, the resulting points generated would actually be biased according to the sine distribution i.e. clumping up at the poles along the z-axis, as shown in Fig. 5. This problem can be resolved by either generating

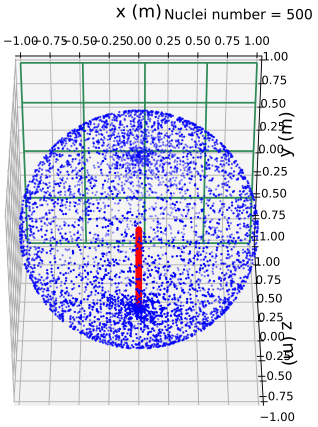

FIG. 5. The incorrect distribution of points on the surface of a sphere to demonstrate the isotropy; both ϕ and θ are i.i.d. It can be seen that the points are clumped at poles. The red line represents the trajectory of nuclei being injected and decayed at $z = -1\,$m and the origin respectively. The green wire-frame represents the detector array.

θ using the inverse transform sampling method in section I, or making sure that $\cos\theta$ is i.i.d uniformly on interval [-1, 1] so that the $\sin\theta$ in equations (1) can be replaced by $\sqrt{1-\cos^2\theta}$ as a result of a coordinate transform[5]. A quick `%timeit` test was ran on iPython showing that they had evaluation time of $12.1\,\text{s} \pm 184\,\text{ms}$ and $11.9\,\text{s} \pm 214\,\text{ms}$ respectively over 7 runs, hence the choice of either methods is trivial. The correct distribution is shown in Fig. 6.

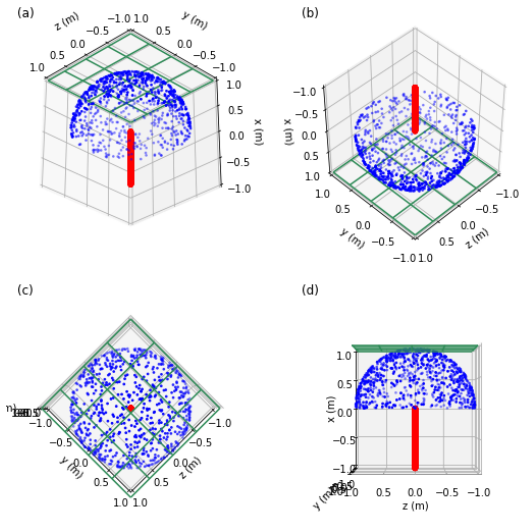

FIG. 6. The correct point distribution plotted using both aforementioned method, for different viewing perspectives to show the homogeneity of the points distribution.

To continue with the simulation, the time for which the nuclei decays can be randomly sampled from an exponential distribution, with the rate as the reciprocal of the mean lifetime.

Its distance from injection point can then be obtained by multiplying it by its velocity. The substitutions $z = 2 - d_{\text{from inj}}$ and $\rho = z/\cos\theta$ can then be done on equations (1) to subsequently model the coordinates of the unsmeared detected gamma rays. To apply a smearing of the detected positions, another 2-d histogram is produced by plotting Gaussian-distributed points with the hit positions as the mean and the detector resolution given as the width in both direction. The simulated detection is given in Fig. 7.

As expected the unsmeared hits on detector should be symmetrical due to the gamma rays being emitted isotropically, and they should concentrate mostly at the centre of the detector according to the exponential distribution, but in turn it is difficult to observe if the 2-d histogram is truly symmetrical. Therefore, one can overlay the 1-d histograms in both x and y direction for a more desirable visual comparison, as seen in Fig. 8. Furthermore, since both histograms do not strictly follow a specific function, it may not be the best idea to fit an arbitrary curve and perform a chi-squared test for symmetry testing since such a test is typically used to quantify the difference between the expected and observed frequencies of a distribution. Hence a better suited statistical measure here would be the Kullback-Leibler divergence (KLD)[7], $D_{\text{KL}}(P \parallel Q)$, where P and Q are discrete probability distributions defined on the same probability space. In simple terms, the KLD measures the asymmetry of P and Q.

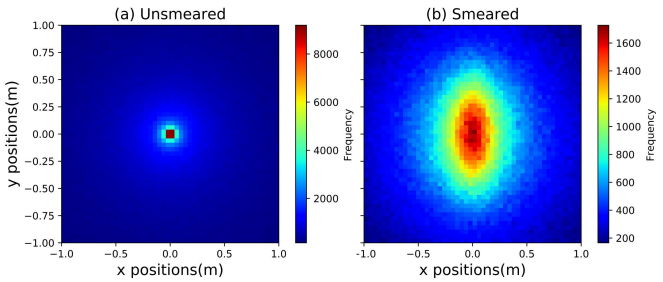

FIG. 7. The unsmeared and smeared gamma ray detection plotted as a 2-d histogram with 50 bins and 5×10^6 nuclei.

The overlayed histograms were first normalised then their corresponding KLD were calculated and shown in Fig. 8. With the KLD being 0 for two identical distributions P and Q, it can be concluded that both distributions in Fig. 5(a) are sufficiently close to be deemed symmetrical, and it is as expected that the KLD of Fig. 5(b) would be slightly higher with the y distribution having a smaller peak due to its slightly higher standard deviation.

It is also worth doing a few tests such as simulating the nuclei decay at fixed distances away from the screen and the case when they have different mean lifetime, as shown in Fig. 9. As expected for Fig. 9(a), the intensity of hits diminishes the further it is away from the detector. This is because the solid angle subtended by the nuclei on the detector decreases as their relative distance increases. Furthermore, the general detection pattern remains similar which reconfirms the isotropy of the emission. The increase of nuclei mean lifetime means in general more of the nuclei will live for longer before they decay,

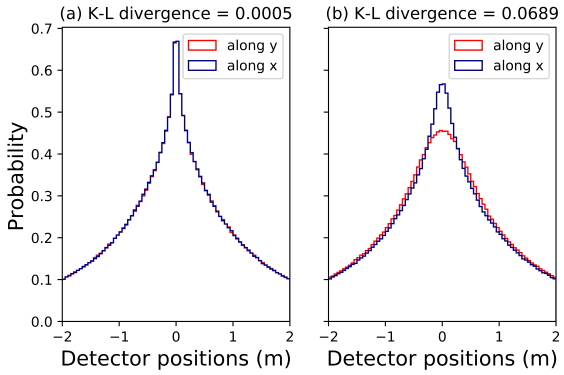

FIG. 8. Marginal distributions of Fig. 7 overlayed on top of each other with.

hence were able to pass through the detector and become undetected, as evidenced in Fig. 9(b). As it is shown, the physics of the simulation was rigorously tested, which makes it clear that the accuracy of the results are sufficiently justified.

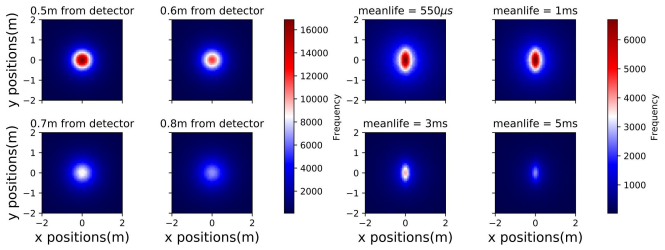

FIG. 9. Detection plotted: (a) unsmeared, for different distances from the detector in the left panel; (b) smeared, for different nuclei meanlife in the right panel. Both with 50 bins and 5×10^6 nuclei.

III. STATISTICAL METHODS AND HYPOTHESIS TESTING

In most cases, a random variable (r.v.) of interest may depend on a parameter θ, best described as a function $f(x \mid \theta)$. As a result, it is statistically possible to infer its true value such that by measuring samples $\{x_1, x_2, ..., x_n\} \in \mathbf{x}$ i.e. by Monte Carlo methods, $f(\mathbf{x} \mid \theta)$ allows one to construct a probability density function (pdf), $g(\theta)$, centered around the true value and bounded by a true standard deviation. In the case where an infinite sample is done, the parameter tends towards its true value by the law of large numbers[8]. Furthermore, when the estimated $g(\theta)$ has non-Gaussian uncertainties, one usually takes interest in *confidence level*, as it quantifies the fraction of total experiments for which an interval would contain the true value of the parameter. Therefore such statistical method is of an immense interest, especially to a collider experiment. For the following simulation of a collider experiment to be realistic, the number of events observed could be best described by the following,[9]

$$N_{\text{obs}} = \mathcal{L}\epsilon\sigma_{\text{eff}} + N_{\text{bkg}}$$

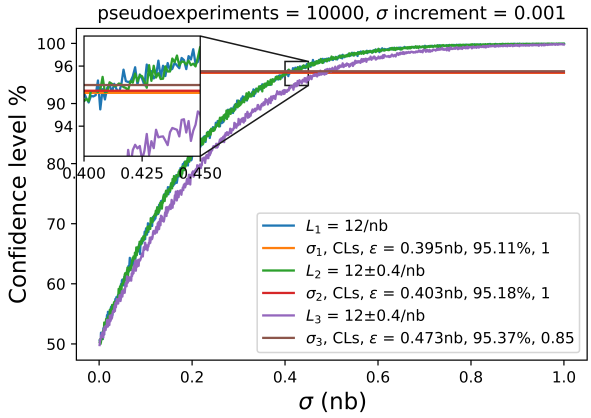

FIG. 10. Cross sections plotted against confidence levels

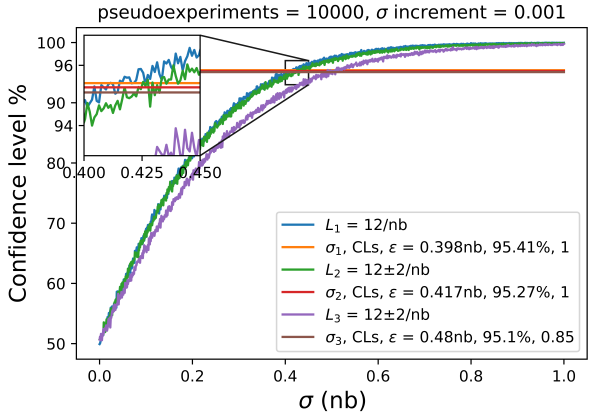

FIG. 11. Cross sections plotted against confidence levels

where N_{bkg} is the number of expected background events, $\mathcal{L}$ is the integrated luminosity, ϵ is the acceptance efficiency and σ_{eff} is the effective cross section, i.e. the branching ratio into an observed channel (BR) times the (unknown) production cross section. Additional uncertainties can be added by sampling random numbers from a normal distribution with the exact luminosity value as the mean and a chosen error of 0.4 events, same as that of the background events'. It is worth noting that a simulation should take into account of the fact that in reality, the detector would not be perfect and so its effectiveness will be described by its efficiency ϵ. Here the detector efficiency has been given an arbitrary value of 0.85 and the simulation results are shown in Fig. [10].

It seems that the first confidence levels % for which for all 3 simulations is bigger than 95% are relatively close together. This comes as not surprising as one would expect that the first confidence levels should converge to the same value as the number of pseudo-experiments increases and as σ's increment decreases. As for the cross sections, having introduced an error of 0.4/nb to L_2 only raised its σ upper limit by 0.008nb. This maybe be because the chosen value of error is of the same magnitude as that of the background's. Moreover, decreasing the efficiency of the detector to 85% increases the σ upper limit. Again, this is also expected since the line hovering 95% confidence level is unlikely to fluctuate for a large run size, however, decreasing the efficiency lowers the overall curve, since less proportion of overall events for a particular σ will now be more than 5. If the error of the luminosity was increased to a larger value of 4, it can be seen from Fig.11 that its confidence line starts to deviate away from the confidence line that has no error. It is apparent that this shift is a result of the statistics of the experiment i.e. the statistical errors of the parameters, since with small luminosity error like in Fig. 10, the no error and with error curves are still close by each other. Hence it can be concluded that these Toy Model are of great interests for real life simulation as it demonstrates the statistical nature that one could get from just simulating the result which can be compared with empirical laboratory evidences.

IV. REFERENCES

[1] Marsland S. (2011) . *Machine Learning: An Algorithmic Perspective, 2nd edition.*((CRC))

[2] Matsumoto. M, Nishimura, T. (1998). "*Mersenne twister: a 623-dimensionally equidistributed uniform pseudo-random number generator*" ACM Transactions on Modeling and Computer Simulation.

[3] Sawilowsky, S. S. (2003). *"You Think Youve Got Trivials?"* Journal of Modern Applied Statistical Methods 2, 218

[4] Ecuyer, P. Owen, A. (2010). *Monte Carlo and Quasi- Monte Carlo Methods 2008, Mathematics and Statistics.* Springer. ISBN 9783642041075.

[5] Arthur. K. M (July, 2015). *Point Picking and Distributing on the Disc and Sphere.* Army Research Laboratory.

[6] Jagannatam, A. (2009). "Mersenne Twister A Pseudo-random Number Generator and its Variants."

[7] Kullback. S, Leibler. R. A. (1951) *"On information and sufficiency"*. Annals of Mathematical Statistics. **22**(1): 79-86. doi:10.1214/aoms/1177729694. MR0039968

[8] Sen, P. K, Singer, J. M. (1993). *Large sample methods in statistics.* Chapman Hall, Inc.

[9] Harel, A. (Jan, 2011) *"Statistical methods in CMS searches"*. CERN-2011-006: pp. 88-93.