Assignment 3: Random Numbers and Monte Carlo

G. Flatters
Department of Physics, University of Bristol.

(Dated: October 20, 2020)

In this exercise two methods have been successfully used to produce random sinusoidal distributions to a confidence level of 100%. The differences between these two methods has been explored. Then Monte Carlo was used to simulate gamma ray detection from nuclear decay, taking into account the uncertainty brought about by the resolution of the detector. The distribution of the gamma rays detected has been supported by theory. Finally, pseudo-experiments were used to successfully determine the cross section at which the null hypothesis - a hypothetical particle existing - is supported at a 95% confidence level.

RANDOM NUMBERS IN A DISTRIBUTION

Simulations for the Monte Carlo method rely on generating random numbers to produce numerical results. This is done using pseudo-random number generators (PRNGs), which are algorithms that generate a sequence of numbers whose properties approximate the properties of sequences of random numbers. Generally, a PRNG is considered useful if the following conditions are met: it has a very long period before the random values begin to repeat themselves; low correlation between successive states and is reasonably fast to run. Throughout this exercise NumpyPy's *numpy.random* routines will be implemented, which use the Mersenne Twister (MT) algorithm. Specifically, NumpPy uses version MT19937 which has a repetition period of around 4.3×10^{6001} [1]. This sufficiently large period, as well as the passing of the DIEHARD statistical test [2], makes this algorithm suitable for the purposes of this exercise, the first part of which is to produce a sinusoidal random number distribution using two different routines.

Analytic Method

The first method is to analytically translate a set of evenly distributed random numbers to a set of sinusoidally distributed ones. If $P(x)$ describes the generated even distribution and $P'(x')$ the desired sinusoidal one, assuming they are both normalised then the cumulative distribution must be the same (1). Because the generated distribution is evenly distributed between 0 and 1, equation 1 can be written as equation 2.

$$\int_{x_0}^{x_{gen}} P(x)dx = \int_{x'_0}^{x'_{req}} P'(x')dx' \tag{1}$$

$$x_{gen} = \int_{x'_0}^{x'_{req}} P'(x')dx' = Q(x'_{req}) \tag{2}$$

The generated even distribution x_{gen} can therefore be converted into a required distribution x'_{req} by applying the inverse function:

$$x'_{req} = Q^{-1}(x_{gen}) \tag{3}$$

In the case where $P'(x') = sin(x')$ for $0 < x' < \pi$, equation 3 becomes

$$x'_{req} = cos^{-1}(1 - 2x_{gen}) \tag{4}$$

this simple equation will be used to produce a sinusoidally distributed set of numbers. Here *np.random.uniform* and *np.arccos* will be implemented which is much faster than the alternative of using in-built Python loops. This is because looping in Python natively is inefficient when dealing with large arrays, such as those required in this exercise. This is due to Python code being dynamic and interpreted, compared to NumpyPy functions which are optimised *C* code which is static and compiled. Furthermore, NumpyPy arrays are densely packed arrays of a homogeneous nature, meaning they are of a fixed size and each element takes up the same amount of memory. Therefore, operations being applied to these arrays can be heavily optimised.

Reject-Accept Method

The analytic method can only be implemented if $Q(x'_{req})$ can be inverted - for many distributions this is not possible. Such distributions require numerical approximations to be made which can compromise the performance of the method [3].

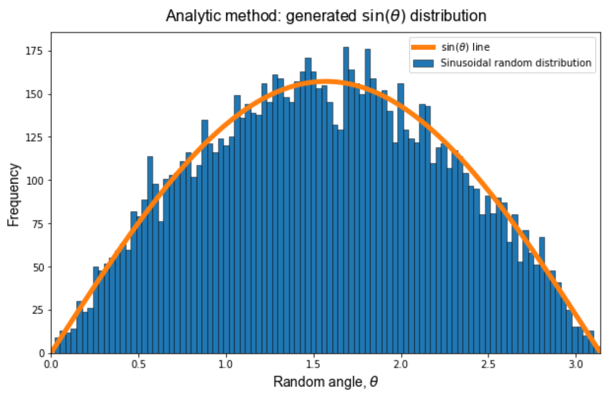
The reject-accept method is more robust as it can be used for more complex distributions where the analytic method fails. This method works by generating two random distributions; the first being an evenly distributed x within a required range and the second being another even distribution y, between 0 and $y_{max} = P'(x'_{max})$. If $y < P'(x')$ then the value for x' is accepted, otherwise it is rejected. The upper bound y_{max} can be any value equal to or larger than the $P'(x'_{max})$, however the greater y'_{max} is the fewer x numbers are accepted making the random distributor less efficient.

Results

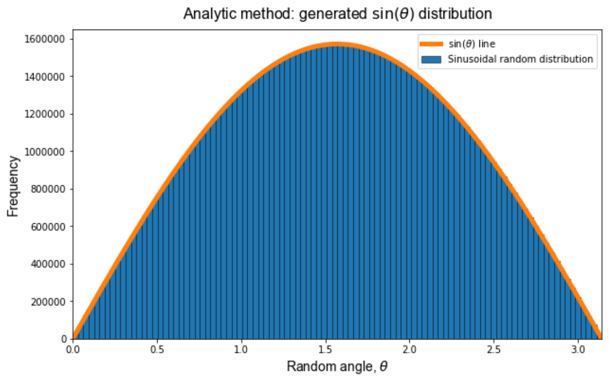
Figures 1 and 2 show the sinusoidal random distributions produced using the two difference methods, for two different amounts of generated numbers. A sine curve is superimposed to show that as the amount of numbers generated increases, statistical fluctuations become insignificant and the random distribution perfectly fits the sine curve. Due to a proportion of the numbers being rejected, the reject-accept method requires more numbers to be generated to achieve the same results as the analytic method. An if statement was implemented to ensure that enough numbers were accepted to properly fill the area under the sine curve.

As well as passing the eye test, figures 1 and 2 also give perfect Kolmogorov-Smirnov test results. The null hypothesis is that the sine curve and the random distribution are drawn from the same distribution. This can be accepted at the 100% confidence level for 1×10^8 generated numbers for both cases; the KS test gives a p-value of 1 for figures 1b and 2b. The D statistic is a measure of the absolute maximum distance between the two cumulative distribution functions, the closer it is to zero the more likely the two sets of data are drawn from the same distribution. This value is small even for figures 1a and 2a where there are only 1×10^4 generated numbers. In these cases the p-values are smaller as expected. These results confirm that as the amount of generated numbers increases, the random distributions tend to perfect sinusoidal distributions.

Figure 3 compares the time taken for the two methods to execute 25 repeats, for varying amounts of generated numbers. Repeat measurements were necessary to reduce the impact of time fluctuations on successive measurements. For small amounts of generated numbers the difference in times is insignificant, but these values do not produce the required sinusoidal distribution. For values that do give the required sinusoidal distribution (e.g 10^8), the analytic method is roughly a factor of 2 faster. This is as expected - the reject-accept method must generate two sets of random distributions, instead of one in the case of the analytic method. However, this is still relatively computationally inexpensive and its robustness makes it a more reliable method for random number distribution.



(a) Numbers generated = 10^4. D statistic = 0.080, p-value = 0.894.



(b) Numbers generated = 10^8. D statistic = 0.020, p-value = 1.

FIG. 1: Sinusoidal distributions produced using the analytic method.

SIMULATING GAMMA DECAY

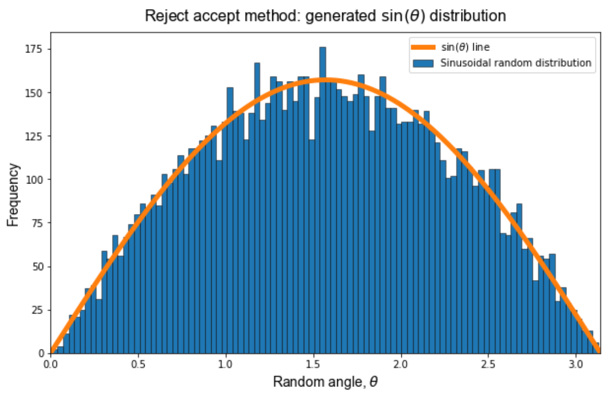
For the next part of the exercise a Monte Carlo technique is used to simulate gamma decay of a beam of unstable nuclei. The nuclei are travelling at $2000ms^{-1}$ and have a mean lifetime of $550\mu s$, upon decay they produce a gamma ray. A detector is perpendicular to the beam, situated $2m$ away from the injection point. The resolution of the detector is $10cm$ in *x* and $30cm$ in *y*.

Any quantity *A* which depends on nuclear decay can be expressed as:

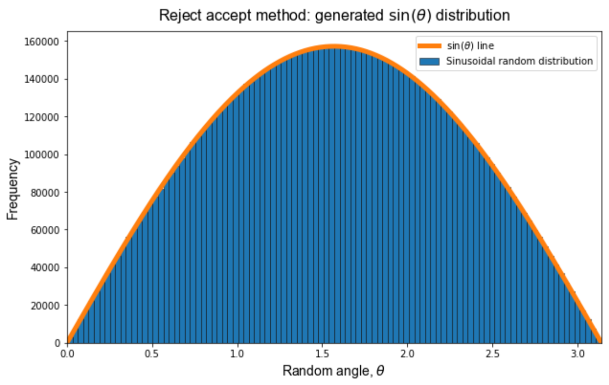
$$A = A_0 e^{\frac{-t}{\tau}} \tag{5}$$

Where τ is the mean lifetime. This equation can be used to find the exponential distribution of the number of nuclei remaining for a given time *t*. Multiplying this distribution by the velocity gives the exponential distribution of the positions at which the nuclei decay, as shown in figure 4. This histogram is plotted using a random exponential distribution and the theoretical curve represents equation 5. As the number of histogram bins approaches infinity, the random exponential distribution perfectly fits the area under the theoretical curve. This graph shows that not all of the nuclei decay before they reach the detector which is $2m$ away. It is found that about 83.8% of the 1×10^8 nuclei decay before reaching the detector.

Now the distribution of the directions that the gamma rays are emitted must be calculated. This is done by picking a random point on the surface of a sphere. It is incorrect to select random spherical coordinates from uniform distributions of θ and ϕ as this will cause bunching around the poles [4]. This is because the area element $d\Omega = sin(\phi)d\theta d\phi$ is a function of ϕ. To successfully obtain points on a sphere such that each surface area element contains the same number of points, the following equations are used where *u* and *v* are randomly generated between 0 and 1:



(a) Numbers generated = 10^4. D statistic = 0.090, p-value = 0.794.



(b) Numbers generated = 10^8. D statistic = 0.020, p-value = 1.

FIG. 2: Sinusoidal distributions produced using the reject-accept method.

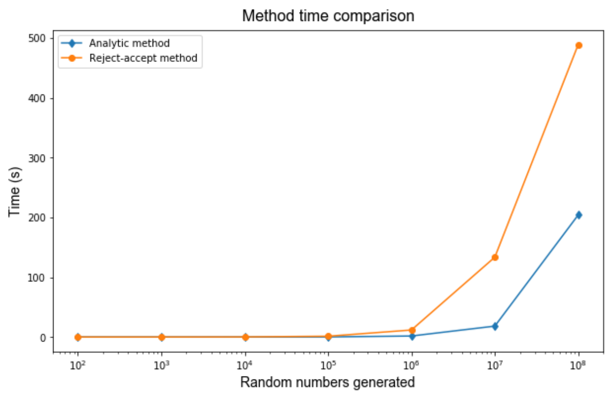


FIG. 3: Time required for the two methods to execute for varying amounts of numbers generated, logarithmic scale between 10^2 and 10^8.

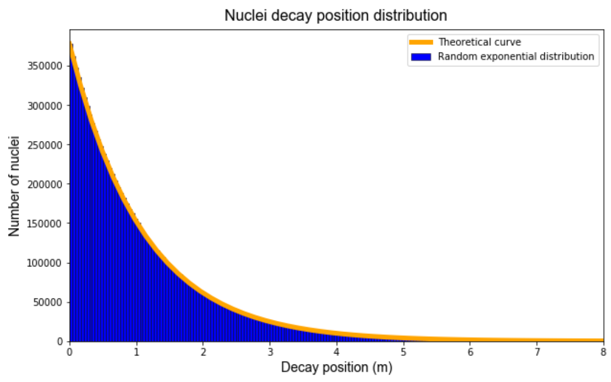


FIG. 4: The positions at which the nuclei decay follow an exponential distribution, total nuclei = 1×10^8 and 500 bins used.

$$\theta = 2\pi u \tag{6}$$

$$\phi = cos^{-1}(2v - 1) \tag{7}$$

Now that the correct distribution of angles has been found, the spherical coordinates must be converted to Cartesian coordinates to get the positions at which the gamma rays are detected on the screen. The coordinate system is such that z is pointing towards the detector from the injection point, it is therefore constant, $z = 2m$. Using the standard equation $z = rcos(\phi)$, r is found:

$$r = \frac{2}{cos(\phi)} \tag{8}$$

This is then substituted into the standard equations for x and y in terms of the spherical coordinates to get the following:

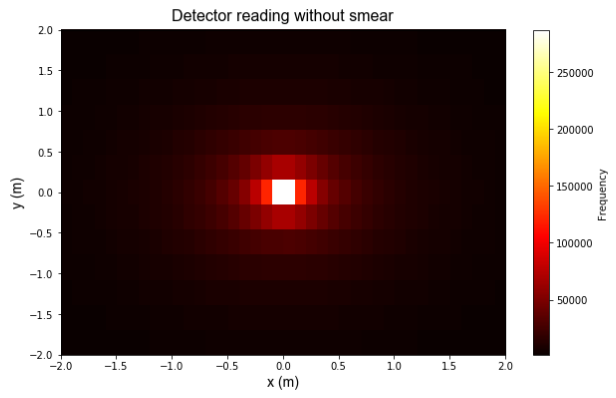
$$x = 2tan(\phi)cos(\theta) \tag{9}$$

$$y = 2tan(\phi)sin(\theta) \tag{10}$$

These equations are used to translate the randomly distributed angles to the $x - y$ plane of the detector. In this coordinate system $\phi = 0$ must be pointing along the z-axis. To ensure this, $\frac{\pi}{2}$ is taken away from each sinusoidally distributed value of ϕ.

Figure 5 shows the distribution of gamma rays detected by a $2m \times 2m$ detector. The bin with the greatest amount of gamma rays detected is in the centre of the screen, the number of gamma rays then decreases radially outwards producing a circular pattern. This is as expected; equations 9 and 10 project the spherical envelope of the randomly directed gamma rays onto the detector. Of all the bins on the detector, the central bin is the shortest distance from the origin (injection point). This means that mapping from a spherical to a planar surface has little effect on the concentration of gamma rays at that point. However, as ϕ and θ increase, small changes in these angles start corresponding to larger and larger changes in position on the detector. This has the effect of decreasing the gamma ray concentrations of bins the further they are from the central bin.

To accurately model the gamma ray concentrations ob-



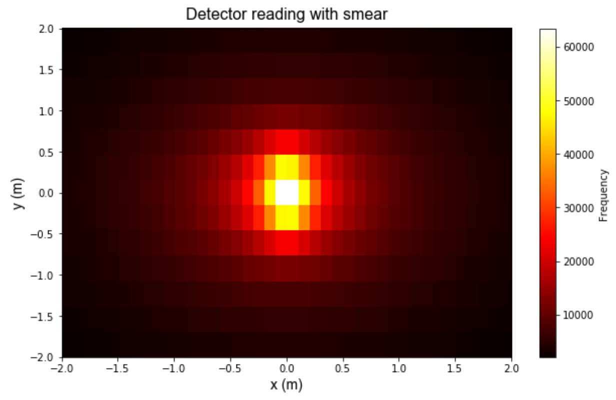


FIG. 5: Data for 1×10^8 injected nuclei, about 83.8% of these decay before reaching the detector.

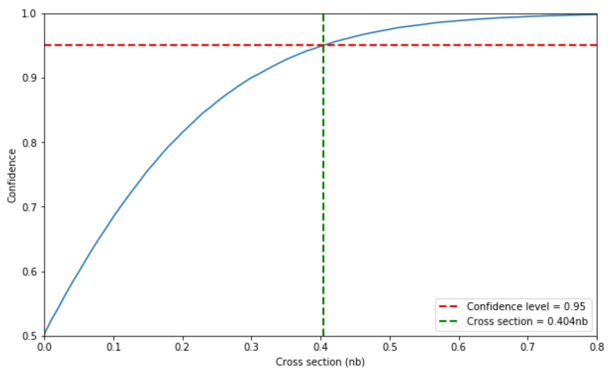


FIG. 6: Confidence *vs* σ curve intersects 0.95 confidence level line at $\sigma = 0.40400 \pm 0.00005$. 1×10^6 pseudo-experiments generated.

served by the detector, the dimensions of the bins used are $10cm$ by $30cm$, in accordance with the resolution of the detector. Figure 5 shows the detector readings both with and without smearing effects. The smear is caused by the uncertainty in position measurements due to the resolution of the detector. It is modelled by adding a random normally distributed value to each position value before adding it to a bin. The mean of this normal distribution is the original measured position and the standard deviations in the x and y directions are the respective resolutions in those directions. This causes gamma ray concentrations to spread into neighbouring bins, decreasing the concentrations of the central bins (note the difference in frequency scale between the two graphs). The "amount" of spreading is effectively the same in both the x and y directions due to the standard deviation being proportional to the dimensions of the bins.

STATISTICAL SOLUTIONS

The final part of this exercise is to generate N pseudo-experiments for a range of cross sections (σ) to find the value at which the required confidence level of 95% is satisfied. The null hypothesis is that a hypothesised new particle exists. A "true" mean of the background count is generated from a normal distribution with a mean of 5.7 and a standard deviation of 0.4. This value is then used as the mean of a Poisson distribution, which is used to find the background count. The signal count is also a Poisson distribution, where the mean is the expected number of the hypothetical particles to be observed $X = L\sigma$. The total count is found by adding the signal count and the background count. This is repeated for N "pseudo-experiments" and the confidence level is found - the number of experiments that produce a total count above the observed value of 5, divided by the total number of experiments.

To find the value of σ for which the confidence level was 95%, a loop was used to find the confidence level for varying values of σ. The loop breaks when it reaches a confidence level of 95% within a given accuracy and ouputs the value of σ for which it occurred, as well a graph of the total event Poisson distribution. Integrated luminosity is used as an example of how additional uncertainties can be accounted for in this function. This had the effect of increasing the required value of σ.

The value of σ that gives a confidence level of 0.95000 ± 0.00005 was found to be 0.40400 ± 0.00005 nb, using 5×10^6 pseudo-experiments. The robustness of this function could be improved; to run quickly it requires a good initial guess which is found through trial and error. Figure 6 was plotted using a separate function, it shows how the confidence level changes for a large range of σ. It confirms the value of σ found previously, as that is the value that the curve intersects the 0.95 confidence level line at.

[1] Makoto Matsumoto. M. matsumoto and t. nishimura, acm trans. model. comput. simul. 8, 3 (1998). *ACM Trans. Model. Comput. Simul.*, 8:3, 1998.

[2] Juan Soto. Statistical testing of random number generators. In *Proceedings of the 22nd National Information Systems Security Conference*, volume 10, page 12. NIST Gaithersburg, MD, 1999.

[3] Luc Devroye. Nonuniform random variate generation. *Handbooks in operations research and management science*, 13:83–121, 2006.

[4] Eric W Weisstein. Sphere point picking. 2002.