Computational Physics Assignment 3: Random Numbers and Monte Carlo

M. White
Level 6, School of Physics, University of Bristol.

I. INTRODUCTION AND THEORY

COMPUTERS are deterministic and cannot be random, indeed Von Neumann stated "*Anyone who considers arithmetical methods of producing random digits is, of course, in a state of sin*".[1] Given the same input, computers output the exact same result every time, making it impossible to generate truly random numbers with a purely algorithmic method. Pseudo-random number generators (PRNGs) use a seed as a starting point to generate a repeatable pattern of numbers with a known period after which the sequence repeats itself. Since these processes are deterministic, knowing the initial seed allows the determination of the generated sequence, and predictions of the next 'random' number can be made. PNRGs require an adequate degree of randomness to be useful in statistical analysis, indicators of which include a long period before the sequence repeats and uniform distribution with no correlation in a large sample size. The Python NumPy and Random libraries use a well known PRNG named the Mersenne Twister (MT), which has a period of 2^{19937}-1 and passes multiple statistical tests such as the famous Diehard tests.[2] Although unsuitable for cryptographic use, there are no obvious vulnerabilities much unlike the RANDU generator, whose 3-dimensional planar correlation was poorly understood when first implemented. True random number generators (TRNGs) create aperiodic non-deterministic sequences of random numbers by using a truly random process such as radioactive decay. While numbers generated from these processes are truly random, they are typically inefficient compared to PRNGs and so are unsuitable for large-scale simulation and modelling. Pseudo-random numbers generated using the MT are adequately random to use in processes such as Monte Carlo methods.

The modern form of Monte Carlo methods (MCMs) were developed at Los Alamos as a method of simulating nuclear collisions.[3] Much like the casino it derives it's namesake from, MCMs rely on random numbers (typically from PNRGs). These can simulate situations in which analytic solutions are not available. MCMs can be used to numerically integrate functions, for example, placing random points on a grid and measuring the proportion under a function can give an approximation to it's area. However one of the main uses of MCMs is pseudo-random number sampling, defined as transforming a set of random numbers from a uniform to a chosen probability distribution. For a desired distribution, $P'(x')$, with an solvable integral, inverse transform sampling can be used.[4] Given a uniformly distributed distribution between 0 and 1, $P(x)$, the generated random numbers can be written as

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

with

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

So random numbers in a distribution of $P'(x)$ can be generated using

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

where $Q^{-1}(x)$ is the inverse function of $Q(x)$. Other methods such as rejection sampling are used if the inverse function is difficult to evaluate. Rejection sampling is similar to numerical integration with MCMs where random numbers are generated and rejected if they lie outside a function at that point.[5] All the points not rejected form a distribution of that function. In a 2d example, generating random (x, y) points and rejecting if $y > P'(x')$. Since x values with higher $P'(x')$ are therefore more likely to be accepted, a non-uniform probability distribution is formed. To reduce unnecessary computation, points that lie outside the range of the function are not generated as they would always be rejected. Other methods such as slice sampling and Markov chain Monte Carlo methods are commonly used but are not the focus of this report.[6]

II. RESULTS AND ANALYSIS

A. Pseudo-random Number Sampling

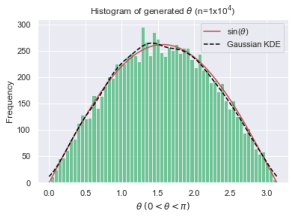
Random numbers are desired in a distribution of $P'(\theta) = \sin(\theta)$ with $0 < \theta < \pi$. Using eq.2, we can integrate to obtain

$$Q(\theta'_{req}) = \int_0^{\theta'_{req}} \sin(\theta)d\theta = 1 - \cos(\theta'_{req}). \tag{4}$$

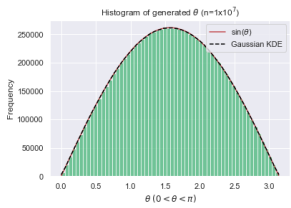
Inverting this function and using eq.3 forms a method generating random numbers in a sinusoidal distribution. This is given by

$$\theta'_{req} = \arccos(1 - \theta_{gen}). \tag{5}$$

For the range of theta given above, the inverse function $Q^{-1}(\theta)$ has the range 0-2. Therefore random numbers were generated using NumPy's MT PRNG between 0 and 2. These values were fed through eq.5 generating a sinusoidal probability distribution as shown in fig.1 for 10^4 and 10^7 random numbers. A Gaussian kernel density estimate (KDE) is applied to the histograms (shown in dashed black) giving an approximation of the probability distribution. Along with a sinusoidal

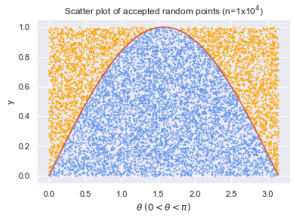


(a) 10^4 random numbers

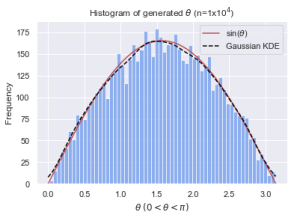


(b) 10^7 random numbers

FIG. 1: These histograms show inverse sampling of a sinusoidal distribution with 10^4 and 10^7 random numbers random numbers generated. Scaled and overlaid are a Gaussian KDE approximation in dashed black and a sinusoidal distribution in solid red. The normalised error between these curves is 0.0448 for (a) and 0.0109 for (b).



(a) Scatter of all points



(b) Distribution of accepted points

FIG. 2: A scatter of rejected and accepted points in orange and blue respectively with rejection sampling of 10^4 random numbers is shown in (a). A histogram of accepted points is shown in (b). Scaled and overlaid are a Gaussian KDE approximation to the histogram in (b) in dashed black with an error of 0.0582 and a sinusoidal distribution in solid red.

distribution for reference, this was scaled with the frequency. Even at a relatively low number sample size in (a), there is a clear sinusoidal distribution indicating that the program is performing as expected. There are still some artifacts, especially towards the middle where the estimated noticeably deviates from the true distribution. The error was taken to be the maximum value of the difference between the KDE and the true distribution. The total time for 10^4 numbers was 3x10^{-4}s with a normalised error of 0.0448. With a larger sample size there is a more accurate approximation with the KDE almost perfectly matching the sinusoidal distribution in (b). This run took 0.24s with a normalised error of 0.0109 showing that increasing the sample size increases the accuracy of the transformed distribution. The bin size was chosen subjectively as it appeared to represent the distribution the best without losing too much information but methods of choosing optimal bin size can be found from Shimazaki[7] and Doane.[8]

Rejection sampling was investigated as an alternative to inverse sampling. Values of θ were randomly generated uniformly between 0 and π. Each value was given a random value of y between 0 and 1, the maximum value of $Q(\theta)$. If $y > Q(\theta)$, the value was rejected, giving a set of points that lie inside the curve. At this point two properties can be measured, the area under the curve and the probability distribution of points. The area under the curve is proportional to the fraction of points under the curve. As all points outside the curve were rejected, the proportion, p, under the curve is given by $p = \frac{len(acc)}{n}$, where $len(acc)$ represents the amount of points accepted and n is total points generated. Scaling to the dimensions of the grid gives an estimate to the area. Between 0 and π, the area under a sine curve is 2. A scatter plot for 10^4 random numbers is shown in fig.2 (a) with accepted values shown in blue and rejected values shown in orange. The proportion of values underneath the curve is 0.6352, and by multiplying by the dimensions of the grid (1xπ in this case) gives an area of 1.996, agreeing with the analytic value of 2 within 0.2%.

A histogram of the produced distribution is shown in (b) allowing a fair comparison to the inverse sampling method. While both methods produce fairly accurate distributions, the normalised error in the rejection method was slightly higher with 0.0582 compared to 0.0448. While the time taken for inverse sampling was 3x10^{-4}s, rejection sampling took 0.56s, almost 2000 times longer for the same amount of random numbers generated. This makes rejection sampling incredibly inefficient compared to inverse sampling. This can be attributed to long loops of generating random numbers and checking to see if they lie under the curve. An unavoidable part of this process is generating numbers that have to be discarded later and while not fully responsible, lengthens the computation time. Rejection sampling however is more general and can be applied to any situation in contrast to the limitations of implementing inverse sampling.

B. Particle Decay Detector Simulation

MCMs were then used to investigate the decays of a beam of unstable nuclei. The system has the beam on the z-axis incident onto a perpendicular plane detector 2m away. To locate the emitted gamma ray's position on the detector, both the distance from the target in the z-axis and the angles of the isotropically emitted gamma rays are required. As radioactive decays are a random process, the event count is modelled with a Poisson distribution. Instead we are interested in the time between these events, which is described with an exponential distribution.[9] The given mean lifetime for the unstable nuclei is 550μs defining the mean of this distribution. The beam is travelling at $v = 2000\text{ms}^{-1}$, and so the distance, d, travelled before the decay is given by $d = vt$, with t being the time before their decay, generated from the above exponential distribution. θ and ϕ representing the azimuthuthal and polar angles should be generated within the ranges [0, 2π) and [0, π] respectively. However if they were distributed uniformly, there would be clumping towards the poles of the sphere since there is a constant polar rectangle density.[10] This does not translate to a constant area density of points as the distance from the centre of the clumping increases so a non-uniform distribution is formed. A transformation of the generated polar angles to a sinusoidal distribution with a maximum at $\frac{\pi}{2}$ gives a uniform distribution over the sphere due to the area element formula of a sphere,

$$d\Omega = \sin(\phi) d\phi d\theta,$$

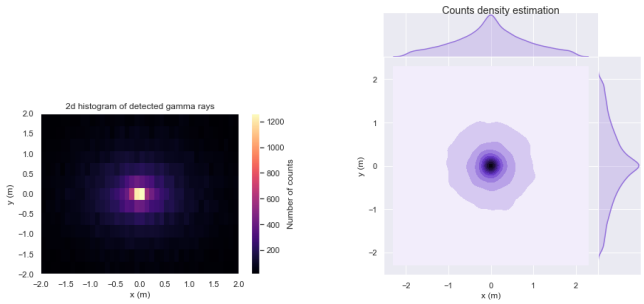
where $d\Omega$, the area element, is known as the solid angle. This ensures that gamma rays are generated isotropically. The transformation can be accomplished using eq.3 and the method detailed in section II.A. It is now possible to generate points on the detector using the random angles and the decay distance using the spherical to Cartesian transformation equations with $r = \frac{(2-d)}{cos\phi}$,

$$x = (2-d)\cos(\theta)\tan(\phi) \qquad (6)$$
$$y = (2-d)\sin(\theta)\tan(\phi).$$

10^5 decays were simulated and detection points on a 4m square detector were calculated from the random generated angles and decay distance. Since gamma rays emitted backwards would never hit the detector, the function in eq.5 was used in the range (0, 1) as this generated angles $0 < \phi < \frac{\pi}{2}$. The distribution on the detector was plotted as a 2d histogram, (a) in fig.3, with a bin size of it's resolution of 0.1 in x and 0.3 in y. Shown opposite is a density estimation of the points generated from the jointplot function in Seaborn. There is a clear circular pattern with a maximum at centre of the detector. Since increasing θ or ϕ by a small amount increases the component of x or y by a large amount, there should be less counts per unit area at increasing radial distance. This is clearly seen with higher counts seen in the centre. In (a), the distribution looks stretched but this is due to skewing of bins from to the resolution of the detector as compared to the true distribution shown in (b).

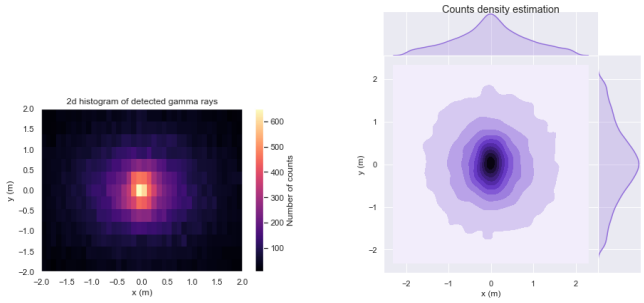
This is an unrealistic scenario as data is binned given an exact position of the gamma ray. In a real-life detector, there is a finite resolution in the detector and hence an uncertainty in the binning. Measurement uncertainties are typically distributed using a Gaussian so for each gamma ray position, the position is Gaussian distributed using a standard deviation of the resolution in the detector. The result, shown in fig.4, has a larger spread of values as a result of this smearing. Although typically within a standard deviation, there is a small chance of generating a value multiple standard deviations outside it's original position and so a larger spread of values appears. As seen in the broader width of the y distribution in (b), the true distribution has been smeared further in y due to the lower resolution in this direction. This is a much more typical read-out of a physical experiment and shows how difficult isolating the true positions for detections can be.

This simulation could be made more realistic by introducing a background signal that the detector would have to deal with. Gamma rays from the decay could be isolated from the background by the use of a trigger that has a threshold on a particular property of gamma rays that the background lacks, such as a higher energy. Particle detectors also have a dead time that occurs after a detection, during which subsequent hits could not be detected. A time would have to be added to each decay such that after each detection, part of the detector would be offline. This could reduce the concentration of detections in the centre, as a higher count means more dead time for sections of the detector there, and would have to be accounted for in a real-life experiment.



(a) Resolution binned **(b)** True distribution

FIG. 3: A 2d histogram of all unsmeared gamma ray detections on a 4m detector is shown in (a) with bins the size of the detectors resolution. The true distribution is shown with a density estimation in (b).



(a) Resolution binned **(b)** True distribution

FIG. 4: A 2d histogram of all smeared gamma ray detections on a 4m detector is shown in (a) with bins the size of the detectors resolution. The true distribution is shown with a density estimation in (b).

C. Statistical Analysis using MCMs

MCMs can be used to estimate experimental errors and confidence limits using a method called Toy Monte Carlo. Large ensembles of pseudo-experiments are formed with randomly chosen parameters and probability distributions generated of whatever variable is to be measured. In this task, an experiment is interested in determining the cross-section of particle X, but a background signal makes it difficult to measure directly. Confidence limits can be set on the cross section of the particle despite the background by generating pseudo-experiments for several estimates for the particle's cross-section. The experiment observed that there were 5 counts. So by simulating both the signal of particle X and the background and calculating the proportion that are above the measured 5 counts, a confidence interval is found.

The integrated luminosity, L_I, is 12nb^{-1} with the mean counts, N, found using the integrated luminosity equation $N = L_I\sigma$, with σ representing the cross section of the event. As this is a random process, it can be represented as a Poisson distribution with N as its mean. This provides the amount of signals we receive that *originate from particle X*. The background is distributed similarly, with a estimated mean of 5.7$\pm$0.4, but there is an uncertainty, so the

true mean background for each pseudo-experiment is found using a Gaussian distribution with a standard deviation of the uncertainty. Estimations for the cross-section were varied and pseudo-experiments generated with a signal count and a background count. Since the total of the signal and background counts was 5, the proportion of the pseudo experiments for each cross-section with a total count above 5 needs to be determined as this represents an experiment with a cross-section larger than seen. As soon as 95% of pseudo-experiments satisfied this condition, the cross-section was recorded and it was possible to say that the true cross section is lower than the recorded with 95% confidence. 10^5 pseudo experiments were generated for each cross section varied between 0nb and 0.8nb in 500 increments. The confidence that the true value lies within each cross section is shown in fig.5 with the region of 95% confidence shaded in orange. The 95% limit for the cross-section was recorded at 0.4072nb, sensible compared to the average background cross section of $\frac{5.7}{12}$=0.475nb, indicating that this process is less likely than the background as expected. The process takes 488s and along with precision, increases with number of pseudo-experiments.

Additional uncertainties could be included in the same way as the background distribution. That is by determining a true mean by the use of a Gaussian distribution with the estimate as the mean and the uncertainty as the standard deviation. An uncertainty in the luminosity was investigated, $L = 12 \pm 2\text{nb}^{-1}$ and it was found that the cross section at which there was a 95% confidence limit increased to 0.4216nb. This is to be expected since adding uncertainties can only decrease the confidence in a result. This method cannot provide an exact value but is useful to determine the range that detectors should be sensitive to when designing physical experiments.

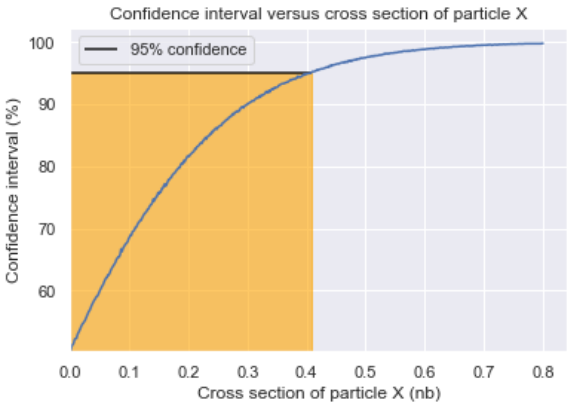


FIG. 5: A plot showing the confidence that the true cross-section lies beneath each value. The region of 95% confidence is shaded in orange. The cross section was varied in 500 increments with 10^5 pseudo-experiments.

III. CONCLUSIONS

This task set out to develop and assess routines for Monte Carlo methods for application in non-uniform sampling and statistical analysis. Sinusoidal distributions were generated using inverse and rejection sampling, with both providing accurate distributions. However rejection sampling was measured to take almost 2000 times longer for the same amount of random numbers with a slightly lower accuracy. Inverse sampling was used to uniformly distribute points on a sphere while modelling decays of unstable nuclei onto a perpendicular plane detector. The resulting distribution of detections was as expected and a smearing applied to represent the uncertainty of the detection given a finite resolution. Confidence limits of the cross-section of a particle were investigated using Toy Monte Carlo. Modelling both the background and particle signals as Poisson distribution with a Gaussian uncertainty on the background, the cross-section was found to be less than 0.4088nb with 95% confidence. Additional uncertainties such as luminosity were investigated and found to decrease confidence in the result, as expected when adding more uncertainty.

[1] John von Neumann. Various techniques used in connection with random digits. 1963.

[2] Makoto Matsumoto and Takuji Nishimura. Mersenne twister: a 623-dimensionally equidistributed uniform pseudo-random number generator. *ACM Transactions on Modeling and Computer Simulation (TOMACS)*, 8(1):3–30, 1998.

[3] Nicholas Metropolis et al. The beginning of the monte carlo method. *Los Alamos Science*, 15(584):125–130, 1987.

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

[5] Luca Martino and Joaquín Míguez. Generalized rejection sampling schemes and applications in signal processing. *Signal Processing*, 90(11):2981–2995, 2010.

[6] Radford M Neal et al. Slice sampling. *The annals of statistics*, 31(3):705–767, 2003.

[7] Hideaki Shimazaki and Shigeru Shinomoto. A method for selecting the bin size of a time histogram. *Neural computation*, 19(6):1503–1527, 2007.

[8] David P Doane. Aesthetic frequency classifications. *The American Statistician*, 30(4):181–183, 1976.

[9] Karl-Heinz Schmidt. A new test for random events of an exponential distribution. *The European Physical Journal A*, 8(1):141–145, 2000.

[10] Mary K Arthur. Point picking and distributing on the disc and sphere. Technical report, ARMY RESEARCH LAB ABERDEEN PROVING GROUND MD WEAPONS AND MATERIALS RESEARCH , 2015.