Computing Physics, Assignment 3

K.X. Lim
(Dated: April 27, 2020)

The main aim of this exercise was to implement Monte Carlo Sampling methods. Task 1 asked for the implementation of the Analytical and Accept-reject method, as well as testing for their respective performance. Task 2 asked for a program to model a nuclear decay physics experiment, while Task 3 asked for a program to model a particle collider experiment.

TASK 1

METHOD

The core concept of Monte Carlo sampling involves using an arbitrarily large number of random samples to obtain numerical results. This method is especially useful to solve problems which are either deterministic in nature or analytically intractable. In the 1st task, 2 methods were introduced - the analytical method and the accept-reject method.

Analytical Method	Accept-reject method
Mathematical algorithm:	Mathematical algorithm:
1. $\int_{x_0'}^{x'_{req}} P'(x')dx' = \int_{x_0}^{x_{gen}} P(x)dx$	1. Generate x'_{req}
2. $Q(x'_{req}) = \int_{x'0}^{x'_{req}} P'(x')dx'$	2. Generate random y
3. $Q(x'_{req}) = x_{gen}$	3. If y $< P'(x')dx'$:
4. $x'_{req} = Q^{-1}(x_{gen})$	4. Accept x'_{req}

Subsequently, Task 1 asked for an algorithm to generate random samples following a sin distribution from a range of 0 to π using both the analytical and the accept-reject method. 3 different functions were created. The $Task1a$ function creates an initial uniform distribution that will be sampled from. $Task1b$ and $Task1c$ function samples using the analytical and accept-reject method respectively. In all 3 functions, a plot of all points was produced, as well as a histogram plot showing the probabilistic distribution of all points.

In order to better understand the results of the Monte Carlo methods, a true sin distribution was also generated, and its values were compared to the values of the histogram bins from the previous functions. The test for convergence was conducted via percentage of absolute difference, and this was done in function $Task1d$, which also plots normalised values of the bins of the histogram.

The $Task1e$ function investigates how the percentage error of both Monte Carlo methods varies with an increasing number of points, while the $Task1f$ function investigates the performance (time taken) of both Monte Carlo methods with respect to number of points.

RESULTS

Results for Figure 1 showed that generally, both the accept-reject method and the analytical method returned values in the

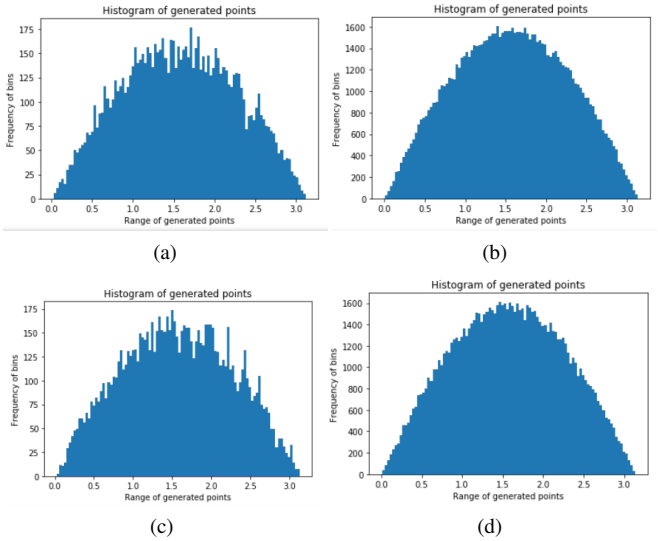


Figure 1: Histogram plots of generated data 1(a): Analytical method, 10000 points. 1(b): Analytical method, 100000 points. 1(c): Accept-reject method, 10000 points. 1(d): Accept-reject method, 100000 points.

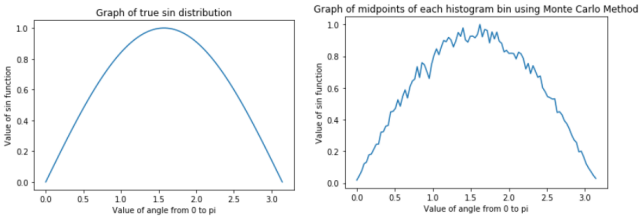
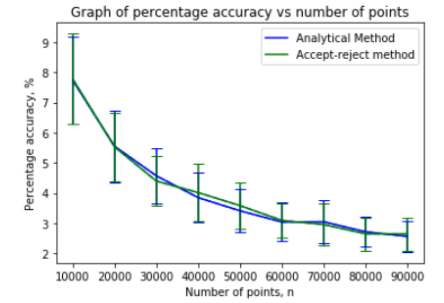


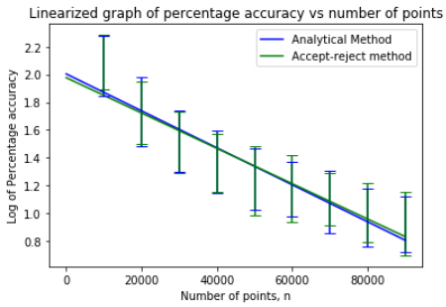
Figure 2: 2(a) Graph of true sin distribution. 2(b) Graph of values of bins from histogram plot of accept-reject method for 50000 points

form of a sin distribution. It can also be seen that the curve of the histogram plot appears to be "smoother" as the number of points increase. This indicates that a higher number of points would return a sin distribution with higher accuracy. Figure 2 shows the curve of the bins of the histogram plot for 50,000 points.

A subsequent analysis was performed on how percentage error varies with an increasing number of points, which is shown by the resulting figure 3(a). An interesting thing to note



(a)



(b)

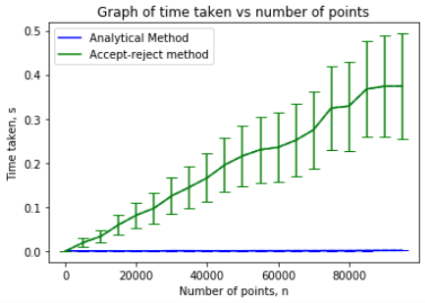
Figure 3: 3(a)This graph investigates how number of points affects the percentage accuracy of both analytical and accept-reject method with respect to a true sin distribution. 3(b) Linear plot of percentage error vs number of points

is that the error bars, which is defined by standard deviation from 100 trials, decreases as the number of points increases. The graph indicates that the function follows an exponential decay relationship, therefore extra effort was done to linearize the graph and examine how close the function fits an exponential decay.

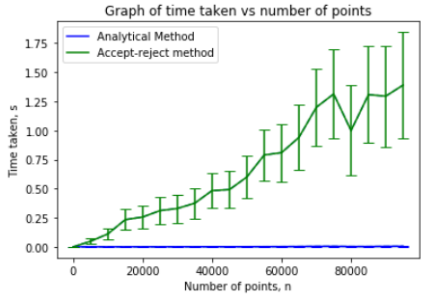
To display the plot seen in Figure 3(b), 2 separate functions were used, with the first function being $scipy.stats.linregress$. This function takes in 2 arrays, x and y and calculates the linear least squares regression. It returns values of coefficients, with the associated standard error as well as the correlation coefficient. However, it could not take error bars of the original data into account, and would therefore be not as accurate. On the other hand, the $numpy.polyfit$ function applies a least squares polynomial fit and is able to take into account error bars for each data. It returns coefficients that are more accurate than the $scipy.stats.linregress$ function, however it returns average residual instead of correlation coefficient. Therefore, the coefficients of the linear fit and the average residual value was extracted using the $numpy.polyfit$ function, whereas the correlational coefficient was extracted using the $scipy.stats.linregress$ function.

In Figure 3(b), the correlation coefficient, r, for both analytical and accept-reject method were found to be -0.95 and -0.96 respectively. This indicates that the linear fit model explains roughly 95% of the variation of the data and a strong negative correlation. The residual values for both analytical and accept-reject method were found to be 0.0038 and 0.0036 respectively. This indicates that the difference between the observed value and the value provided by the linear fit is close to 0.

To further improve upon this analysis, a further investigation could be done on the consistency of the results of both methods. This can be achieved by investigating how the standard deviation, or error bars in Figure 3 changes with respect to number of points. A better method for linear analysis could also be implemented, where the $statsmodels$ module essentially combines the features of both $numpy.polyfit$ and $scipy.stats.linregress$ functions. This module could be used to compute a linear least squares fit with errors as weights and calculate the final adjusted correlation coefficient, r.



(a)



(b)

Figure 4: These graphs investigated time taken for each iteration vs number of points. 4(a) was plotted when the laptop was charged with a power plug, and 4(b) was plotted when the laptop was running on battery power.

Figure 4 investigates the performance of both methods, which is defined here as time taken for each iteration, against the number of points. Generally, it can be seen that as number of points increases, the accept-reject method shows a linear increase in terms of time taken, whereas for the analytical method, the time taken remains constant throughout. Clearly, this shown that the accept-reject method takes up more processing power, and should be avoided if possible for a large number of iterations.

An interesting comparison was made on how the performance of both method varies when the laptop is at peak performance (being plugged in to an AC adapter) versus the laptop

powered solely by a battery. As the y axis of both Figures 4(a) and 4(b) shows, the time taken for the accept-reject method to be run off battery power takes a very long time compared to being plugged in. However, the analytical method remains constant in both cases, thus can be taken to be the better of both methods, if the function to be sampled can be analytically inverted

TASK 2 - NUCLEAR PHYSICS PROBLEM

This section involved modelling a real nuclear decay physics experiment. A detector array was placed 2m away from the source. First, the code randomly generated z coordinates following a normalised exponential decay formula for radiation. The accept-reject method was used as the inverse of the exponential decay function was deemed to be too complex. After z positions of particles were generated, the direction of each decay was also generated by randomly generating x and y positions, within the limit of the detector array. A second part to this task involved 'smearing' the same experiment by adding precision uncertainty of the detector array along both x and y axis. The noise was in the form of a Gaussian blur, with Gaussian distributions of standard deviation 0.1m and 0.3m being added along the x and y axis respectively.

RESULTS

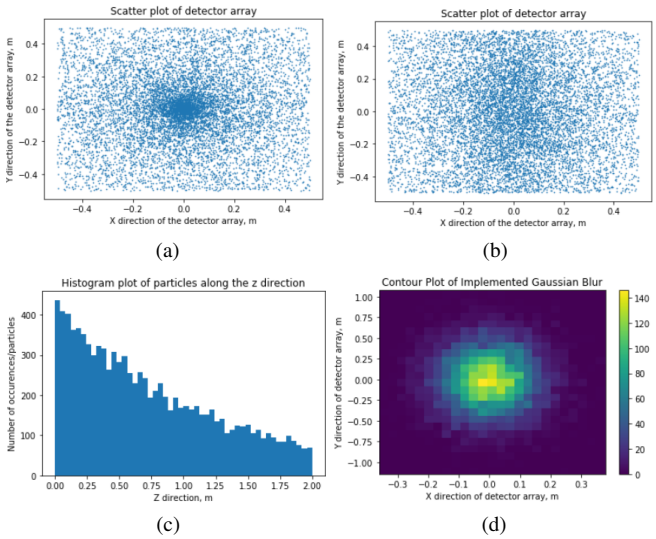


Figure 5: 5(a) is the scatter plot of radioactive decay detected on the detector array 5(b) is the same scatter plot with added resolution uncertainty. Figures 5(c) shows the distribution of radioactive decay particles along the z position 5(d) shows the Gaussian distribution from the Gaussian blur

Figure 5(a) shows the original experimental results for a nuclear decay physics experiment, as seen on a detector array.

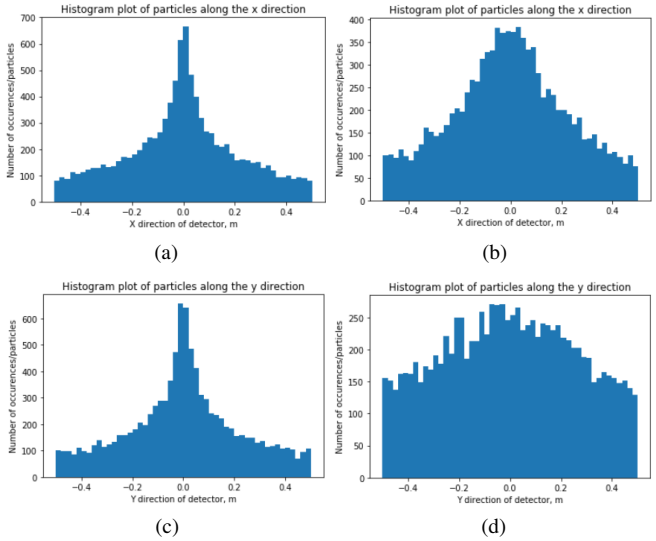


Figure 6: 6(a) and 6(b) are histogram plots of positions of particles along the x axis for the original experiment and the 'smeared' experiment respectively. 6(c) and 6(d) are histogram plots of positions of particles along the y axis for the original experiment and the 'smeared' experiment respectively.

It can be observed that there is a concentration of radioactive particles near the centre of the array, which is subsequently dissipated in Figure 5(b), when a 'smearing' is added in the form of Gaussian Noise. Note that the smearing effect only affects x and y coordinates, therefore Figure 5(c), which encodes the distribution of positions of particles along the z axis, is invariant in both cases. As expected, the histogram can be seen to follow an exponential decay distribution.

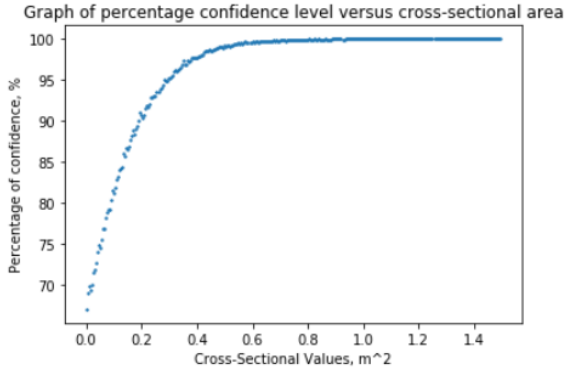
Figure 6 displays the resulting distribution of positions along the x and y axis for both the original and 'smeared' experiment. A python routine was written to calculate the standard deviation of each histogram in order to better understand how the resolution uncertainty of 0.1m (x axis) and 0.3m (y axis) affected the distribution of particles along the x and y axis respectively. For the original experiment, the standard deviation was found to be 0.22, while for the smeared experiment, the standard deviation along the x axis (Figure 6(b)) was 0.23, while standard deviation along the y axis (Figure 6(d)) was 0.26. It is rather interesting to note that the resultant standard deviation does correspond to the sum of the initial standard deviation and the applied standard deviation from the Gaussian noise. However, it is doubtful that the same relation would expand unto higher levels of applied Gaussian noise. Therefore, a further improvement in this section would be investigating the effect of applied standard deviation from Gaussian noise on the resultant standard deviation of the histogram of 'smeared' particles. (such as Figures 6(b) and 6(d))

TASK 3 - COLLIDER SIMULATION

This task required the user to program a number of pseudo-experiments in order to measure a number of significant events in the discovery of a new particle, X. The pseudo-experiments were created in an increasing range of cross-sectional areas, with each pseudo-experiment consisting of 10000 points. In each point of the pseudo-experiment, a Poisson sample is obtained from the theoretical background distribution, and another Poisson sample is obtained from the experimentally observed distribution (which is defined as Luminosity $\times$ Cross-sectional area). These 2 Poisson samples were added, and if the value of the summation ≥ 5, these points were marked using the coded $confidence$ function. If the number of marked points are great enough such that they form 95% of all points in 1 pseudo-experiment, the cross-sectional value is then recorded as the value of which 95% confidence in observing particle X has been achieved.

A second part to this task was investigating how additional uncertainties would affect the cross-sectional value to achieve 95% confidence. This routine was investigated by the $task3b$ function, where an option was give for the user to choose between testing uncertainty in luminosity and theoretical background distribution. Reasonable levels of uncertainty were set from 2% - 5%.

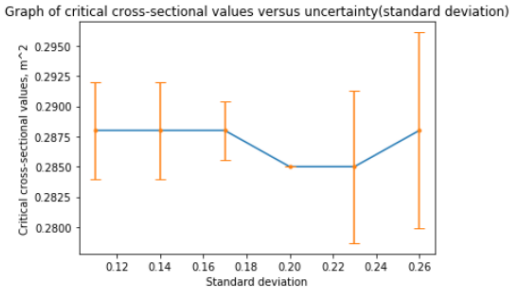
RESULTS



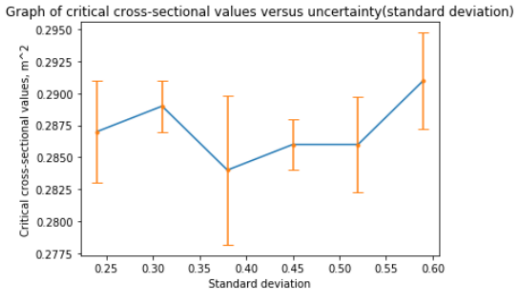
(a)

Figure 7: Graph showing percentage of confidence levels vs cross-sectional values. The cross-sectional value at 95% confidence interval was extracted

Figure 7 shows the results of running the collider experiment. Multiple iterations of the entire $task3a$ function were run to obtain an average value and standard deviation. However, it was observed that the results obtained from a single iteration $(0.290 \pm 0.005)m^2$ were rather similar to results obtained from 10 iterations $(0.291 \pm 0.005)m^2$. Therefore, the multiple iteration method was discarded as it was concluded that the standard error (which is the interval between each cross-sectional value) sufficiently explained the standard deviation of results.



(a)



(b)

Figure 8: Graphs showing critical cross-sectional values versus uncertainty added in the form of standard deviation. 8(a)is when theoretical background uncertainty is added, whereas 8(b) is when uncertainty of luminosity were added.

For the sake of consistency, the number of particles in both $task3a$ and $task3b$ functions had to be equal. Setting n as 10000 provided much clearer results, but the $task3b$ function, which generates Figures 8(a) and 8(b), took 8 minutes to run. Therefore, an alternative, computationally cheaper option was created for the user to run only 1000 points.

Figure 8(a) which added uncertainty in theoretical background distribution, showed that an increase in uncertainty does not necessarily increase the critical cross-sectional value, but it does increase the standard deviation (error bars) of the experimental results. On the other hand, increasing uncertainty in luminosity (Figure 8(b)) showed a general uptick in the critical cross-sectional values, but no general trend was observed in the standard deviation (error bars) of experimental results.

A further improvement in this analysis can be made by analysing the effect of number of points on the critical cross-sectional value, as well as its resulting errors. This could then allow a more informed choice of number of points to be made. Moreover, a combination of uncertainties in luminosity and theoretical background distribution can be added simultaneously to investigate the effects on final results. It would also be proper to compare the values obtained from this experiment with theoretical probability distributions, which serves as a way to validate this experiment.