1. (a) Using the method outlined on page 33 of the notes, we were able to simulate the decay of $N$ nuclei with an activity of $\Lambda = 1.2\,\mathrm{s}^{-1}$. Running that simulation a number of times and noting how many decays occurred in each counting interval, for both 10s and 1s counting intervals, we were able to produce the plots in figures 1 and 2. Both times the simulation was run 1000 times.

   The means were found to be 1.205 with standard deviation 1.109 for 1 s and 11.243 with standard deviation 3.038 for 10 s. The expected mean is $\Lambda T$ where $T$ is the counting interval so with their respective standard deviations, they are reasonably close to the expected value.

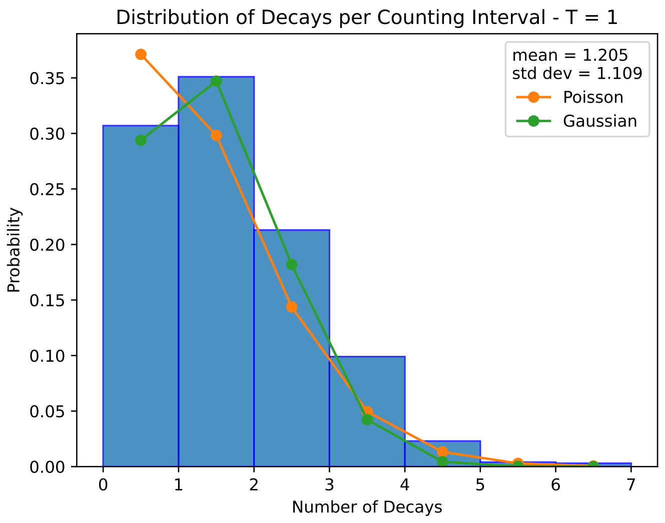


Figure 1: Distribution of decays per counting interval for $T = 1\,\mathrm{s}$, normalised and plotted with the Poisson and Gaussian distributions of the same mean and standard deviation. Simulation run 1000 times.

   (b) Along with the results of 1. a), figures 1 and 2 show the comparison between the histograms and both a Poisson distribution as well as a Gaussian distribution. The mean used to plot both, as well as the standard deviation for the Gaussian, was drawn from the data itself.

   We can see in both cases that the data is best modelled by the Gaussian distribution, which is not what we expect. For such low means, we expect the Poisson distribution to not be modelled well by the Gaussian, and we expect the data to follow a Poissonian distribution. The standard deviation of the data seems to suggest a Poissonian distribution, as both are close the the square root of the mean, but the plots suggest otherwise.

   (c) In order to investigate the distribution of times between successive decays of the nuclei, we chose a counting interval of 10 s and ran the simulation 1000 times. Each time a decay occurred, the current time in the simulation was noted and if no decays had happened in the time step, the time since the last decay was calculated and added to an array. If there had already been a decay in that time step, the decay was ignored in order to avoid an interval of 0 seconds. The distribution resulting from this analysis, along with the expected form of an exponential distribution, is shown in figure 3.

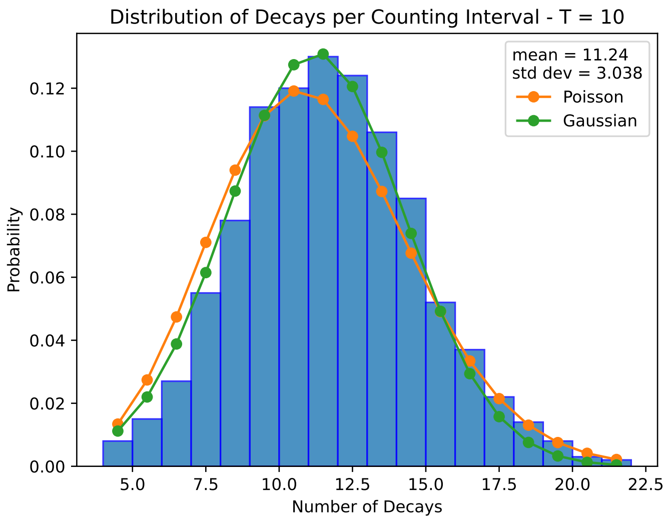


Figure 2: Distribution of decays per counting interval for $T = 10\,\mathrm{s}$, normalised and plotted with the Poisson and Gaussian distributions of the same mean and standard deviation. Simulation run 1000 times.

In order to work out the expected mean, we can think about $\Lambda$ telling us that we expect, on average, 1.2 decays every second, so the time between decays should be $1/\Lambda$, or $0.833\,\mathrm{s}$. Looking at the mean in figure 3, we find it to be $0.801\,\mathrm{s}$. Throughout all our testing it was consistently lower than expected. We are not sure why this is.

2. (a) In order to use the transformation method, we integrated each probability distribution and rearranged to find $y_i$ in terms of the uniformly distributed $x_i$:

i)

$$\begin{aligned} x_i &= \int_{-\frac{\pi}{4}}^{y_i} \cos(2y')dy' \\ &= \frac{1}{2}(\sin(2y_i)+1) \\ \implies y_i &= \frac{1}{2}\arcsin(2x_i-1) \end{aligned}$$

ii)

$$\begin{aligned} x_i &= \int_{2}^{y_i} \frac{1}{\sqrt{8}-\sqrt{3}} \frac{y'}{\sqrt{y'^2-1}}dy' \\ &= \frac{1}{\sqrt{8}-\sqrt{3}}\left(\sqrt{y_i^2-1}-\sqrt{3}\right) \\ \implies y_i &= \sqrt{(x_i(\sqrt{8}-\sqrt{3})+\sqrt{3})^2+1} \end{aligned}$$

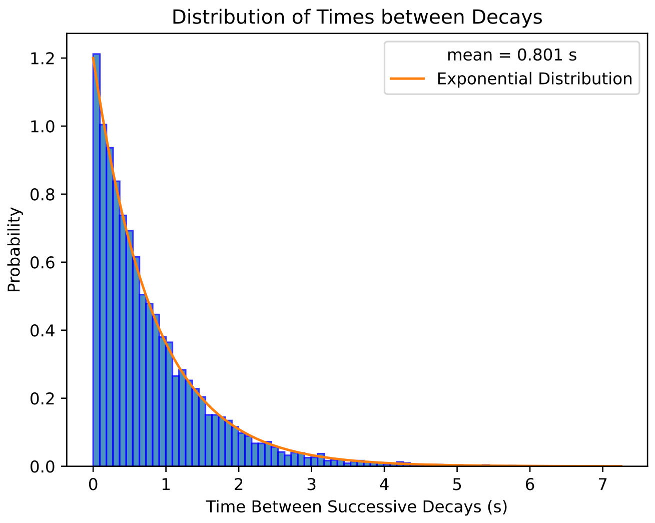
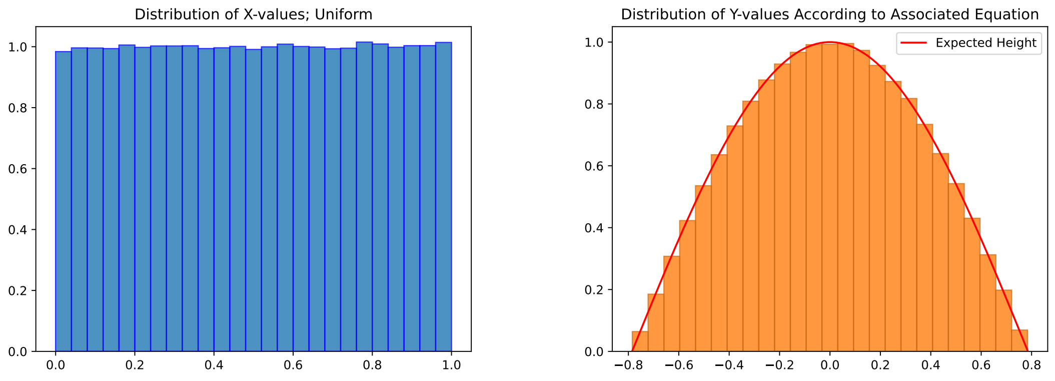


Figure 3: Distribution of times between successive decays, ignoring decays that happen in the same time step. Plotted with the histogram is the exponential distribution with $\lambda = \Lambda$, the activity.

iii)

$$\begin{aligned} x_i &= \int_1^{y_i} \frac{1}{\sqrt{8}} \frac{y'}{\sqrt{y'^2 - 1}} dy' \\ &= \sqrt{\frac{y_i^2 - 1}{8}} \\ \implies y_i &= \sqrt{8x_i^2 + 1} \end{aligned}$$

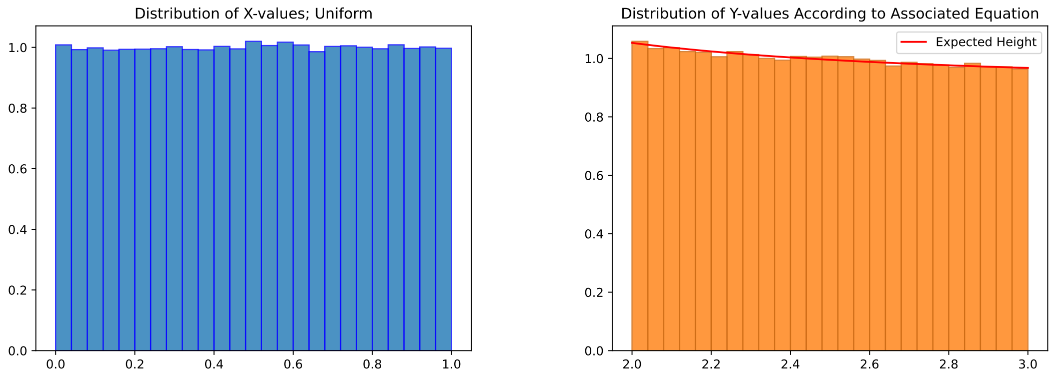
Then by simply generating 500 000 random numbers in $[0, 1)$ and feeding them through the equations found above, we could generate 500 000 numbers distributed according to the respective probability distribution.



(a) Uniform distribution of $x_i$ values

(b) Distribution of $y_i$ values

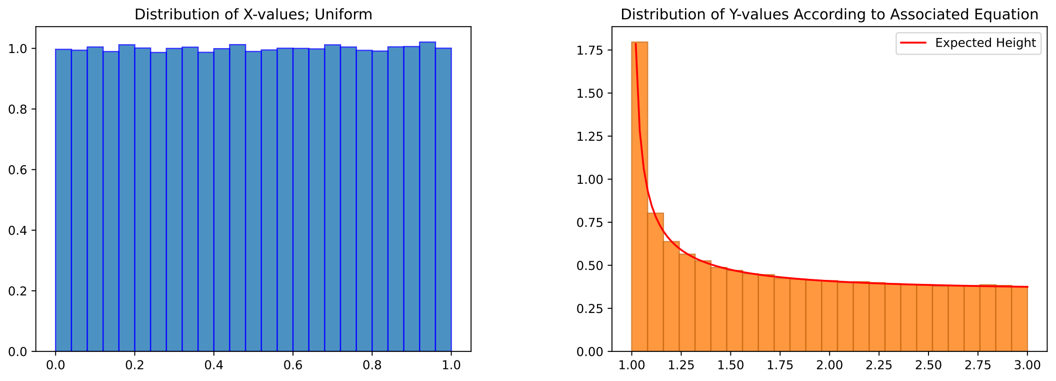
Figure 4: The uniform distribution of numbers in $[0, 1)$ alongside the $y_i$ values generated from them using the transformation method, according to $p(y) = \cos(2y)$. Plotted with the transformed distribution is the expected probability distribution. Each distribution is split into 25 bins. 500 000 numbers were generated.



(a) Uniform distribution of $x_i$ values

(b) Distribution of $y_i$ values

Figure 5: The uniform distribution of numbers in $[0, 1)$ alongside the $y_i$ values generated from them using the transformation method, according to $p(y) = \frac{1}{\sqrt{8}-\sqrt{3}}\frac{y}{\sqrt{y^2-1}}$. Plotted with the transformed distribution is the expected probability distribution. Each distribution is split into 25 bins. 500 000 numbers were generated.



(a) Uniform distribution of $x_i$ values (b) Distribution of $y_i$ values

Figure 6: The uniform distribution of numbers in $[0, 1)$ alongside the $y_i$ values generated from them using the transformation method, according to $p(y) = \frac{1}{\sqrt{8}}\frac{y}{\sqrt{y^2-1}}$. Plotted with the transformed distribution is the expected probability distribution. Each distribution is split into 25 bins. 500 000 numbers were generated.

We can see in each case that the transformation method worked well, accurately approximating the respective probability distribution.

(b) Next, we aimed to use the rejection method to generate 500 000 random numbers according to the distribution in figure 4: $p(y) = \cos(2y)$. To do this we first find the maximum value of $p(y)$ in the interval of interest, then generate a pair of numbers corresponding to a coordinate in the box bounded by the interval and the maximum value. If this coordinate falls within the distribution, it is accepted, otherwise it is rejected and we try again. This is repeated until the required amount of numbers $y_i$ are generated, in our case 500 000.

In order to speed up computation, we wanted to avoid generating a random number at each iteration, so a set of random numbers is generated at the start and then looped over to find those that will be accepted. This requires knowing how many, on average, will be accepted by the algorithm. This can be calculated using

$$P(\text{accept}) = \frac{\int_a^b p(y)dy}{p_{\max}(b-a)} = \frac{1}{p_{\max}(b-a)} \tag{1}$$

The last step assumes the probability distribution is normalised. With this, we can generate $NP(\text{accept})$ random numbers, multiplied by 1.5 to be safe, and be certain that we will accept at least $N$ in the process.

Using this method, we produced the plot in figure 7, finding that the method approximated the true distribution fairly well.

(c) The rejection method could be used to generate the distribution in ii), but not in iii)

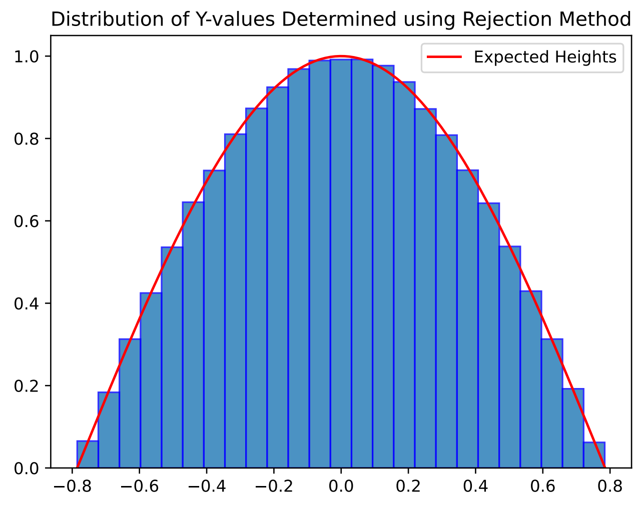


Figure 7: Random numbers distributed according to $p(y) = \cos(2y)$ using the rejection method. 500 000 numbers were generated. The data is histogrammed into 25 bins. The expected form of the distribution is plotted along with the histogram.