Experiment 311: Nuclear Counting Statistics

Aim

To investigate statistical distributions associated with radioactive decays from a long-lived isotope.

To write a Python program that analyses the signal of radioactive decay signals.

References

1. R.D. Evans, "The Atomic Nucleus", McGraw-Hill (1955), Ch. 26–28.
2. E. Rutherford, J. Chadwick & C.D. Ellis, "Radiations from Radioactive Substances", p.171 (Cambridge University Press 1951).
3. National Nuclear Data Center (`www.nndc.bnl.gov`)

Introduction

Some particles (excited atoms, radioactive nuclei, and also some elementary particles) are unstable and decay. The statistics of such decays are of interest since a model of what might be happening at the microscopic level is clearly false if it fails to reproduce the statistics.

In this experiment, you record statistics from a Geiger counter placed close to a radioactive source. These measurements are compared with model distributions using the χ^2 test. In the theoretical exercises, you learn about some underlying assumptions that can lead to the model distributions.

Safety

The radioactive source used in the experiment is weak and you will spend a comparatively short time doing the experiment. Thus, your radiation exposure will be very small in comparison to exposure from other sources such as cosmic rays. Handle the source only by its metal base. If you should accidentally touch any other part of the source, wash your hands with soap and water. If the source is dropped or damaged, report this immediately to a demonstrator.

For further information about the safe handling of radioactive substances, see the Appendix of Experiment 251 'Radioactivity'.

Apparatus

Strontium, ^{90}Sr, is a radioactive source which emits beta particles and decays to yttrium, ^{90}Y. Strontium has a half-life of 28.79 years. Yttrium is itself radioactive with a half-life of 64.10 hours, and undergoes beta-decay to the stable zirconium, ^{90}Zr.

The Geiger counter has a rail upon which the source can be positioned and clamped. When the Geiger counter detects a particle, it produces an electrical pulse. The electrical pulses can be counted using the electronic counter provided. The data can then be saved as a .txt file and analysed via Python.

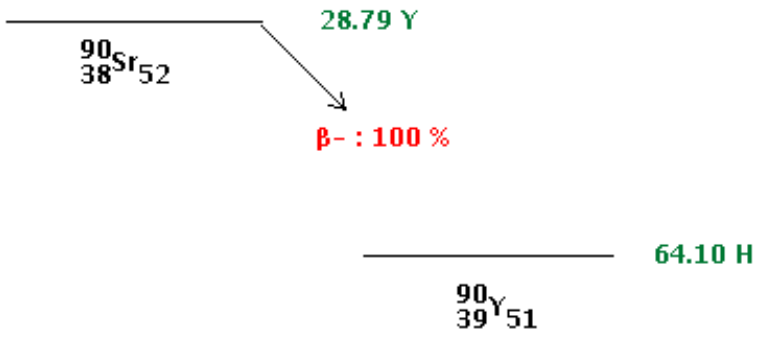

Figure 1: The decay of ^{90}Sr. Half-lives are shown alongside nuclei. The percentage gives the proportion of parent nuclei that decay by the given mode. Clearly, the only mode is beta internal conversion (IC). Adapted from Ref. 3.

Checking for constancy

Statistical experiments are among those where it can be least obvious if things go 'wrong'. When you come to perform your statistical calculations, you will find that the number of decay counts in a given time interval will vary over the course of this experiment. This variation could be due to the statistical nature of radioactive decay or it could be a consequence of something being 'wrong' (e.g. a fault in the electronics of the Geiger counter causing its operation to vary with time, or changes in cosmic ray background). Unlike in some other experiments, it is ambiguous and difficult to tell if large fluctuations in the results indicate problems with the experiment.

One simple test is to check for constancy. That is, to repeat a procedure at intervals with an unchanged apparatus and look for any unexpected changes in the results. For example, you could periodically record the number of counts in one minute during the lab session, and check if the values are within expected fluctuations (what is the standard deviation for a Poisson distribution?). Of course, if the fluctuations are not as expected, this may be caused by the decays not obeying Poisson statistics rather than problems with the apparatus (which emphasises the difficulty of statistical experiments).

Programming

One of the aims of this experiment is to be able to construct a Python program that analyses your radioactive decay signals. You may write your code any way you like, as long as it gets the job done. Keep in mind that both you and your demonstrator should understand what your code is doing, so commenting on your code is strongly advised. One approach you might like to take is to write two separate programs; one for data collection from the .txt file and another for the required calculations. The latter will be adjusted for each of the next three sections of the experiment.

- Getting started:

  (1) Note that you can connect a USB to gather the data recorded by the Geiger counter pulse shaper.

  (2) The cable from the pulse shaper needs to be plugged into the computer.

  (3) Go into `C:\Program Files (x86)\Physics\Advlab\e311` and open the executable.

  (4) In order to get started, you will need to select 'Connect'. You can select a time to collect data for and a sample rate, which by default is 25 kHz. You may like to read the notes and play around with some data collection to figure out the settings you think are best.

  (5) Once you have obtained data, you can save the data as a .txt file (by default NuclearCounting.txt).

- After data collection, your code will need to be able to first of all open the file and read each of its columns. Then, you will need to come up with a way of identifying when a decay has occured. Note that the 'Count Peaks' button counts peaks by first calculating the mean and standard deviation of the data, and then finding points which have a difference of 4 standard deviations or more with the mean (and then waiting until the value gets back below 4 S.D. until another peak can be recorded).

Once you have identified the decays, perform the calculations asked for in the next section. Divide these tasks into smaller steps to sketch a more detailed plan of your code. You may like to discuss this with a demonstrator.

- To load the file NuclearCounting.txt and collect its data, you can write a few lines of code that go something like this:

```
import numpy as np
a = []
b = []
f = open("NuclearCounting.txt", "r")
for line in f:
   line = line.strip()
   parts = line.split(",")
   time = float(parts[1])
   value = float(parts[2])
   a.append(time)
   b.append(value)
f.close()
x = np.array(a)
y = np.array(b)
```

This will read the columns of the file NuclearCounting.txt in read-only mode, and put the second and third columns into arrays `x` and `y`. Notice that there are a few header lines in your .txt file, so you will need to either delete them beforehand, or add a few lines of code so that Python ignores them.

- In order to find the peaks, you may find it useful to import the `signal` module:

```
from scipy import signal
```

Then you have a few options. You can use the functions `argrelextrema` or `find_peaks_cwt`, to name two. However, this will give you the indices of every local extremum; not the extrema corresponding to the decays. In order to get the number of decays, you will need to come up with a way to count the peaks corresponding to strontium decays, and you will need to exclude other extreme values which are too near - you will get other ones due to statistical noise which do not necessarily correspond to another decay. The following example code might help you. This uses `find_peaks_cwt` to count extrema in some array `v` with a difference of at least `stand` S.D. with the mean `m`.

```
m = np.mean(v)
d = np.std(v)
stand = 4
peak_ind = signal.find_peaks_cwt(v, np.arange(w1,w2))
peak_sum = np.sum(v[peak_ind] > m + stand*d) + np.sum(v[peak_ind] < m - stand*d)
```

Here `np.arange(w1,w2)` sets the width of the peaks to be between `w1` and `w2`. For more information and options refer to the documentation online. In order to count the decays correctly, you will need to either choose a peak width which closely agrees with the typical width of a decay peak, or you will need to adjust the above code so that it does not count peaks until there is a drop to within 4 S.D. of the mean again.

- To do the data analysis, write out your code and test it on a 'toy' problem first (e.g. one second's worth of data, or something like `[1 2 3 4 5]`). This makes it easier to see what is happening at each step. There is also a nice visual representation of the decay events in the .exe.

Statistical Tests

One you are happy with your data collection, you will need to move on to performing the following statistical tests.

Poisson distribution

If m is the mean number of random events per unit time, then the probability of obtaining n random events in a time t is given by

$$P(n,t) = \frac{(mt)^n \exp(-mt)}{n!} = \frac{\lambda^n}{n!} \exp(-\lambda) . \tag{1}$$

(1) Collect appropriate data using Python and perform a χ^2 test to determine if the number of counts, n, obey Poisson statistics. You decide the placement of the radioactive source and t. You can work in any time units you like (seconds, samples, etc.), but ensure that $3 \lesssim mt \lesssim 6$. The data should be taken over some fixed time. Then divide this into regular intervals of size t and analyse n in each interval.

In your report, include a plot of your data (e.g. a histogram) together with the expected values derived from Eq. 1. Interpret your χ^2 value, i.e. does it suggest you can conclude that the decay signal follows Poisson statistics?

Gaussian distribution

If X is a normally distributed random variable with mean μ and variance σ^2, then the probability of X taking a value between x and $x + dx$ is $G(x)\,dx$, where

$$G(x) = \frac{1}{\sqrt{2\pi}\sigma} \exp\left[-\frac{1}{2}\left(\frac{x-\mu}{\sigma}\right)^2\right] . \tag{2}$$

(2) Adjust the experiment and repeat procedure (1) so that $200 \lesssim mt \lesssim 300$. Use $G(n)$ in place of $P(n,t)$, with $\mu = \sigma^2 = mt$. Modify your Python codes from (1), and make sure you plot your data appropriately.

Exponential distribution

Suppose that m is the mean count rate. If $\tau = 1/m$ is the mean time interval between consecutive counts, then the probability of obtaining an interval between t and $t + dt$ is $W(t)\,dt$, where

$$W(t) = \frac{1}{\tau} \exp\left(-\frac{t}{\tau}\right) . \tag{3}$$

(3) Like in procedures (1) and (2), collect appropriate data using Python and perform a χ^2 test to determine if the times between consecutive counts obey the exponential distribution. Again, modify your Python code to perform calculations. In your report, include a plot of your data together with expected values and interpretations of your χ^2 values.

Questions

1. Assuming that:

   (a) the probability per unit time that an atom will decay is time-independent, i.e. all time-intervals of duration Δt are equivalent

   (b) the probability tends to zero as Δt goes to zero

   (c) the probability of the $(k+1)$th event is independent of the previous k events

   show that the probability distribution for observing n events in a given time interval is the Poisson function (Eq. 1). See Evans (Ref. 1).

2. Consider fixing the distance between source and detector. You can observe the number of counts in short and long time intervals with mean counts of ≈ 4 and ≈ 250, respectively. For the short time intervals, a plot of the data does not look Gaussian. Explain why the data for the long time intervals does appear Gaussian.

3. Given the assumptions in the first question, show that the distribution of time intervals between successive events is given by the function $W(t)$ of Eq. 3 (you may assume the Poisson distribution, if you wish).

List of Equipment

1. Radioactive source, ^{90}Sr.
2. Geiger counter with track.
3. Electronic counter.
4. Computer with Python installed.

D.A. Wardle (based on an existing experiment by E.R. Collins and A. Chisholm): April 8, 2009.

Updated: September 14, 2009.

R. Au-Yeung: April 29, 2013.

SGM 29 July, 2014 (microphone bug fix added to text).

R. Au-Yeung: 25 November, 2014 (Python).

M. G. M. Crabb: 16 December, 2015.