PHYS2320 Computing 2 Assessed Coursework

1 Introduction

This document will give you all the details for the main project part of your Computing 2 Assessed Coursework. This project carries a total of 85% of the credit for the overall module (unless otherwise advise by Dr Burnell). Please remember that marks are awarded for the structure, style, and robustness of your code as well as for its functionality. Please see the mark sheet and grading criteria on the VLE for details.

You are strongly advised to read the whole of this task sheet thoroughly as you may find information in the later sections that is helpful to completing the assignment.

1.1 Deadlines

You should submit your work via the VLE **by 2pm on Wednesday 4th May 2016**. Late submissions will incur the standard University penalty of 5% for each day and part of day that it is late.

We hope to return a feedback sheet to you around the same time as the semester 2 exam results are released.

1.2 Time Allocation

There are no set times when you must work on the coursework. However, there will be demonstrators available in the Computer lab clusters between 10am and 12 noon each Wednesday and Dr Burnell will be available either in room 8.310 on the Physics Research Deck, or his office on Mondays 2pm-3pm for surgeries[1]. You can also ask questions on the forums on the VLE.

During the Easter vacations both the demonstrators and Dr Burnell have other duties they need to get on with and so have much less time to spend dealing with Computing 2 related matters. Questions posted on the forum will be answered as soon as time permits, but this will be less speedy than during terms time – you should try to plan your work with this in mind.

[1] Alternative times and arrangements will be advised for the week immediately before the Easter holidays and the final week of term

1.3 Overview

In this assessed project you will analyse some simulated data using Python code and prepare a simple report that gives your key results and describes how your code works. You will submit your code, report and a data file via the VLE for assessment.

Your code will be assessed first of all by an automatic checking script that will test whether your code works with both the data file you submit with your code and with a second data file of standard data. The testing program checks whether your code produces results in the format specified here and whether the results match both the known correct answers and also match results from a reference solution.

Your code and report will then be assessed by one of the module demonstrators who will grade it for structure, style and documentation and whether your report describes the code you have written accurately.

2 Background

*The physics behind this coursework task is likely to be unfamiliar, but you are **not** required to understand this physics in order to do this coursework. In the interests of making a realistic problem to be solved, however, the problem is based on real physics and this background section describes the experiment that is being simulated for this coursework.*

The tasks in this coursework are based around the analysis of muon spectroscopy. A muon(μ^-) is an elementary particle (a *lepton*), which, like an electron, has a charge of $-e$ and a spin of $\pm\frac{1}{2}$, but, unlike electrons, have a much larger mass. Muons are produced by firing accelerated protons into a target material, the resulting collision of protons with the nuclei of the target produces pion (π^+), which in turn decay after a short time into an anti-muon, or *positive muon (μ^+)* .

These positive muons are themselves unstable, with a half-life of $2.197034\,\mu$s, but in this time they can be manipulated and implanted into a magnetic material. Inside the magnetic material they will undergo precession as the spin on the muon rotates in the magnetic field inside the material.

2.1 Precession

When a magnetic moment, $\underset{\sim}{\boldsymbol{M}}$ is placed in a magnetic field $\underset{\sim}{\boldsymbol{H}}$, classically we expect it to line up due to a torque:

$$\underset{\sim}{\boldsymbol{\tau}} = \underset{\sim}{\boldsymbol{M}} \times \mu_0 \underset{\sim}{\boldsymbol{H}}$$

. If, however, $\underset{\sim}{\boldsymbol{M}}$ arises from the angular momentum of a particle, such as an electron or muon, then there is a torque acting on an angular momentum, and from Physics 1, we know that this gives rise to precession:

$$\begin{aligned} \underset{\sim}{\boldsymbol{\tau}} \quad = \frac{\partial \underset{\sim}{\boldsymbol{l}}}{\partial t} \quad &= \underset{\sim}{\boldsymbol{M}} \times \mu_0 \underset{\sim}{\boldsymbol{H}} \\ \frac{1}{\gamma}\frac{\partial \underset{\sim}{\boldsymbol{M}}}{\partial t} \quad &= \underset{\sim}{\boldsymbol{M}} \times \mu_0 \underset{\sim}{\boldsymbol{H}} \\ \frac{\partial \underset{\sim}{\boldsymbol{M}}}{\partial t} \quad &= \mu_0 \gamma \underset{\sim}{\boldsymbol{M}} \times \underset{\sim}{\boldsymbol{H}} = \gamma \underset{\sim}{\boldsymbol{M}} \times \underset{\sim}{\boldsymbol{B}} \end{aligned} \tag{2.1}$$

where γ is called the gyromagnetic ratio, which for a $\mu^+$is 851.616 $\mathrm{Mrads^{-1}T^{-1}}$.

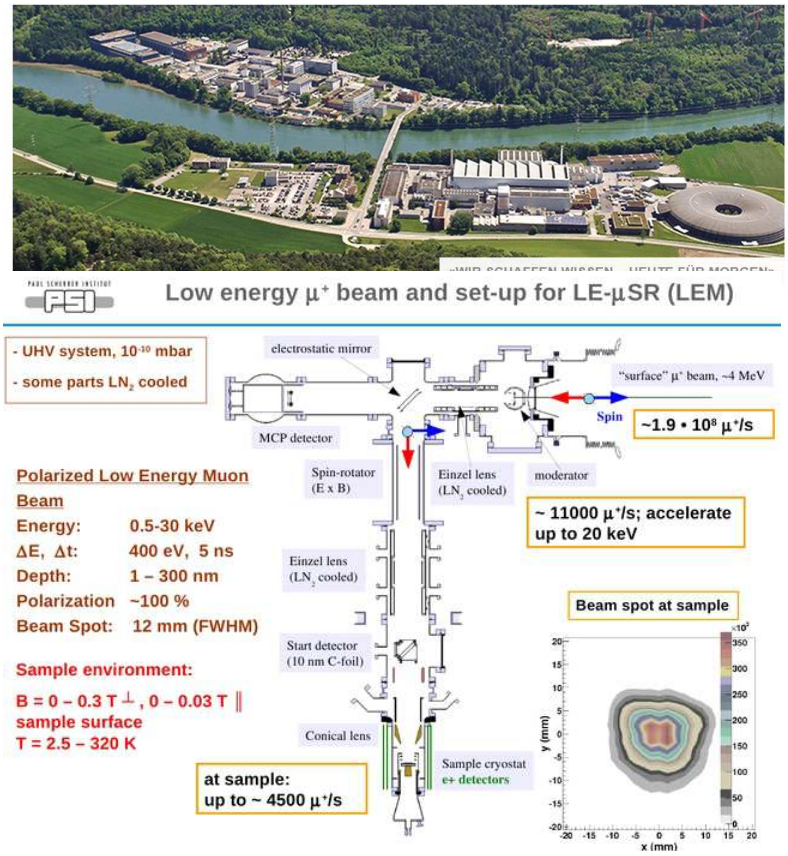

Figure 1: The low energy μ^+beam line at the Paul Scherrer Institute in Zurich. This facility is almost unique in the word in being able to create muons with relatively low energies that can be implanted into thin magnetic films to study the depth dependence of magnetism at interfaces buried in the thin film structure.

Equation 2.1 describes a circular motion of the moment $\underset{\sim}{\boldsymbol{M}}$ about $\underset{\sim}{\boldsymbol{B}}$ with a frequency given by:

$$\omega_{Larmor} = 2\pi\nu_{\text{Larmor}} = \gamma|\underset{\sim}{\boldsymbol{B}}| \tag{2.2}$$

Since the muon is unstable, after some time it will decay, in the process releasing a positron and two neutrinos – however the direction that the positron comes out of the decay process depends on the direction the muon spin had been pointing in at the time of decay, which in turn depends on the decay time and precession frequency, which in turn depends on the magnetic field inside the material in which the muon was implanted.

2.2 Muon Spectroscopy

By measuring the statistics of which way the positrons are emitted, it is possible to work out the precession frequency and then from that to work out the magnetic field in which the muon was sitting when it decayed. Since muons have both mass and charge, the energy with which they are implanted and hence the depth which they reach during implantation can be controlled by accelerating or decelerating them with electric fields. This allows one to measure the magnetic field inside materials at, for example, interfaces between two different magnetic materials.

Figure 1 shows an illustration of a real μ^+beam line facility based at the Paul Scherrer

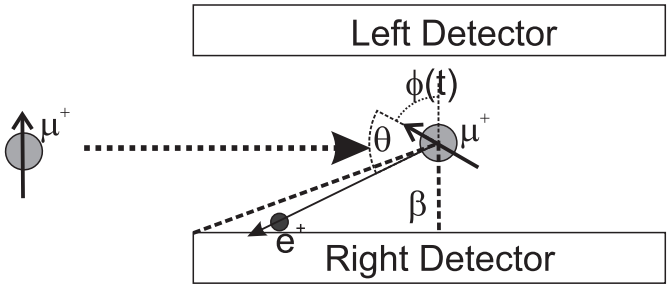

Figure 2: The geometry of muon decay and positron detection in a low energy muon spectroscopy experiment. β is a measure of the size of the detectors, giving the angle covered by each detector, θ is the angle relative to the muon spin direction that the e^+ positron is emitted and $\phi(t)$ is the angle at which the muon spin was pointing at the time of its decay. The magnetic field $\underset{\sim}{\boldsymbol{B}}$ runs in or out of the page.

Institute in Zurich, Switzerland

2.3 Detection

As stated above, the direction the positron is emitted depends on the direction the muon spin was pointing in when it decayed. The probability that positron comes out at some angel θ relative to the muon spin direction is given by:

$$W(\theta) = 1 + \frac{1}{3}\cos\theta$$

so by measuring the number of positrons produced in a certain direction as a function of time, one can measure the muon's spin direction and precession rate. In practice, it is better to use positron detectors of finite size to get a good count rate, so a typical geometry is shown in figure 2.

Taking into account the decay of the μ^+, its precession, the size of the positron detectors and the angular distribution of the positron emission, it is possible to derive a set of equations describing the counting statistics for each channel.

$$P_L(t) = \tfrac{\beta}{2\pi\tau_\mu}\left(1 + A_0(t)\right)e^{-t/\tau_\mu} \tag{2.3}$$

$$P_R(t) = \tfrac{\beta}{2\pi\tau_\mu}\left(1 - A_0(t)\right)e^{-t/\tau_\mu} \tag{2.4}$$

$$A_0(t) = \tfrac{P_L - P_R}{P_L + P_R} = \frac{-1}{3}\frac{\sin(\phi(t)-\beta) - \sin(\phi(t)+\beta)}{2\beta} \tag{2.5}$$

P_L and P_R are the probabilities that a positron will be detected in the left or right channels due to a muon decay. The muon lifetime is given by $\tau_\mu = 2.197034\,\mu$s and $\phi(t)$ is the muon spin angle from precession which is given by:

$$\phi(t) = \omega_{Larmor}t = \gamma\underset{\sim}{\boldsymbol{B}}t$$

with $\underset{\sim}{\boldsymbol{B}}$ the magnetic field[2] and $\gamma = 851.616\,\mathrm{Mrads^{-1}T^{-1}}$.

[2]As is common in the field, we do not distinguish clearly between a magnetic field and a magnetic flux density. Strictly $\underset{\sim}{\boldsymbol{B}}$ is a flux density given by $\underset{\sim}{\boldsymbol{B}} = \mu_0(\underset{\sim}{\boldsymbol{H}} + \underset{\sim}{\boldsymbol{M}})$.

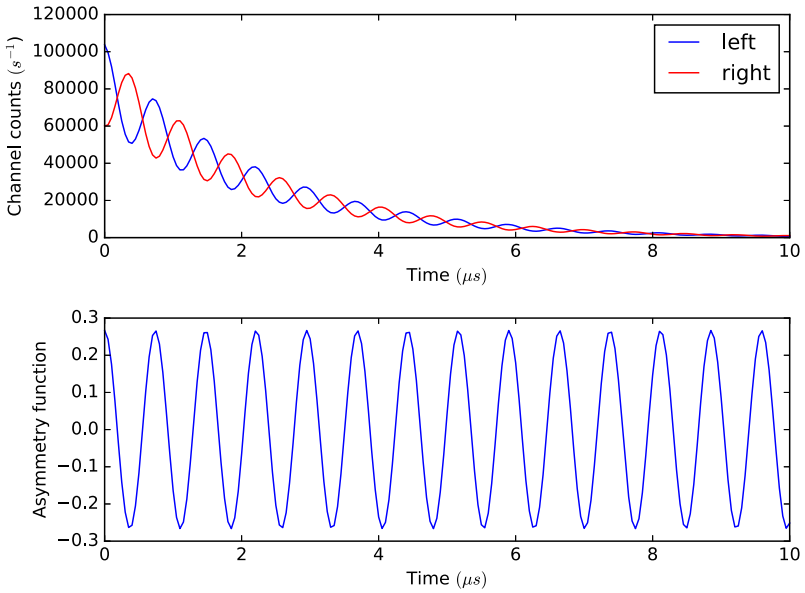

Figure 3: Top: calculated left and right channel signals, from equations 2.3 and 2.4 , and bottom: calculated asymmetry signal from equation 2.5 . For both plots, the magnetic field $\underset{\sim}{\boldsymbol{B}} = 10\,\mathrm{mT}$, detector angle, $\beta = 1.131\,\mathrm{rad}$, gyromagnetic ration$\gamma = 851.616\,\mathrm{Mrads^{-1}T^{-1}}$ and the muon decay time $\tau_\mu = 2.197034\,\mu\mathrm{s}$.

2.4 De-phasing

Equations 2.3 to 2.5 , assume that the magnetic field, $\underset{\sim}{\boldsymbol{B}}$ is uniform. In reality this is rarely the case and so there is a process of de-phasing where the longer any muon is in the sample, the less likely it is to remain in phase with other muons that have remained in the sample for the same length of time. The effect of this is that the asymmetry function (equation 2.5 is gradually damped and can be given by:

$$A_0(t) = \frac{-1}{3}\frac{\sin(\phi(t)-\beta)+\sin(\phi(t)+\beta)}{2\beta}e^{-t/\tau_{damp}} \tag{2.6}$$

2.5 Experiment

In a real muon experiment, μ^+s are implanted one at a time with a fixed energy (and hence implantation depth) into a sample that is placed in a magnetic field of typically around $10\,\mathrm{mT}$ and the time taken for the muons to decay and which detector the resultant positron is emitted to is recorded. This is repeated several hundreds of thousands of times and a histogram is then built up of the detection times at each channel. These histograms resemble equations 2.3 and 2.4 . The two histograms for each channel can be combined using equation 2.5 to give just the asymmetry function that contains the information about the precession of the muon and hence the magnetic field present where the muon decayed. Generally the asymmetry function will have some damping as shown in equation 2.6 . The whole process is then repeated for several different implantation energies which can give a profile of the magnetic field present at different depths in the sample.

Since the μ^+decay time is around $2.2\,\mu\mathrm{s}$, the maximum rate at which muons can be implanted is around $5000\,\mu^+\mathrm{s}^{-1}$ and so the experiments can take a week or two to run in total.

3 Data

For the assessed task you are provided with some simulated data files recording muon decay times and channels. The data file starts with a header section that describes some information about the experiment, there is then an end of header marker, followed by a line of column headings and then the numerical data.

A link to the website to download your data from is given on the VLE in the Coursework section. On the download page you can select various options to determine the type of data being simulated.

- **Mode**: this has two options, *practise* and *assessment*. In both cases the data will be generated with a set of randomly selected parameters (detector angle β, magnetic field $\underset{\sim}{\boldsymbol{B}}$, damping time τ_{damp}), the differences is that in practice mode these parameters are listed in the meta-data in the file so that you can check whether your are successfully finding the correct values of these parameters. In assessment mode, these parameters are not listed and this will be one of two datasets that your code will be tested against.

- **Damping**: this option simply turns the damping time on or off in the asymmetry function.

- **Multiple Energies**: if this option is off then your data will contain results for just one implantation energy - there will be only two columns of data – the first column has the time of the detection event (measured in μs) and the second column is either 1 for the left detector or 2 for the right detector. If this option is turned on, then you will have data for 5 implantation energies and therefore have 10 columns of data arranged as 5 pairs of times and detectors. The implantation energies used are given as a comma separated list in the meta-data section of the file.

In order to complete all of the code tasks and get full credit for code functionality you will have to turn on all of these options.

You are very strongly advised to start with the options turned off and to write code that works well with the simplest data sets first. Remember that code functionality is only one criteria that you are assessed on and you will get a better mark by wiritng well structured and maintainable and documented code that does not do everything than writing bad code that does work with the most complex data!

4 Tasks

Your code should do the following:

1. Read in the data file and produce two histograms for the left and right detection times. If you have used the multiple energies option, then your code should make the plots **only** for the $10\,\mathrm{keV}$ implantation energy.

   Your plots should include your ISS login name (*i.e.*py14...) and the channel in the title and the axes should be appropriately labelled. You will need to include these plots in your report.

2. For each energy in your data set, find the asymmetry by combining the left and right detector signals according to equation 2.5 and then fit with either equation 2.5 or equation 2.6 , to find the detector angle β, magnetic field $\underset{\sim}{\boldsymbol{B}}$ and (if applicable) the damping time constant τ_{damp}. See the hints section below for details about calculating error bars for the data.

3. For the 10 keV implantation energy **only**, make a plot of the asymmetry data with error bars and also show the result of your fitted asymmetry function (equation 2.5 or 2.6) on the same plot.

   Your plot should include your ISS login name in its title and have appropriately labelled axes. You will need to include this plot in your report.

4. If your data set includes multiple energies, make a plot showing how the magnetic field $\underset{\sim}{\boldsymbol{B}}$ and (if appropriate)) damping time constant (τ_{damp} vary with implantation energy). The magnetic field $\underset{\sim}{\boldsymbol{B}}$ should have a quadratic dependence on the energy, so fit your extracted values of $\underset{\sim}{\boldsymbol{B}}$ to a function of the form:

$$|\underset{\sim}{\boldsymbol{B}}| = aE^2 + bE + c$$

   , where a,b,c are constants. Overlay this quadratic fit on your plot.

   Your plot should include your ISS login name in its title and have appropriately labelled axes. You will need to include this plot in your report.

5. Write a report on your code. Your report should have two sections:

   a) A results section that tabulates all the parameters found by your code from your data set, with errors. You may manually adjust the precision you quote your answers to, so that errors are quoted to 1 s.f. and the result is quoted to a precision that is consistent with the error (as you would do for labs). This section should also include all plots produced by your code.

   Your report does not have to include any discussion of the physics or experimental method *etc*. This section just has to have plots and numerical results.

   

   b) A flow chart that describes how your code works. It should describe the functions you have written and the overall logic of your program. It is not necessary to document every single line, but the flow chart should be sufficient to allow someone else to write a program that would work the same way as your code.

5 Submission

You should submit a minimum of three files to the VLE by the deadline.

1. The datafile used to generate the data in your report as downloaded from the website. Your code will be tested with this datafile to check your report's results. Your code will also be checked with another data file with different parameters, but with the same features turned on and off to ensure that it works with more than one set of data. (Every set of practise data will have a different set of parameters, so you can check this for yourself.)

2. The report in either Word docx or pdf formats.

   If you are using a different version of Word than installed on the cluster machins, or a different word processor, then you are strongly advised to convert your file to pdf and to check that the result looks as you expect. Unfortunately incompatibilities between versions of Word and between word and Open Office can result in flow diagrams getting messed up.

3. Your python code. This may be either one or more than one file, but the main file should be called *issid*.py where *issid* is your computer login name. Your code must have a function called **ProcessData** that takes a single parameter that is the name of a file. Further details of what **ProcessData** should do are given in the next section. A template of a suitable file is on the VLE.

5.1 The Python File

Your main Python script should be named after your University computer login name and have the extension ".py". It must define at least one function, **ProcessData** that takes a single parameter that is the name of a data file to read in and process and returns a dictionary with the parameters found by your code. A list of the dictionary keys is given in the template file – you must either return a number or **None** for each key. Do not change the keys.

An example skeleton of the function is in a template file on the VLE and shown below:

```
def ProcessData(filename):
    """Documentation string here."""
    #Your code to process the data file goes here

    results={
        #this would be the magnetic field for 10keV data
        "10keV_B":None, #(T)
        # the error in the magnetic field
        "10keV_B_error":None, #(T))
        #Detector angle and error
        "beta": None, #(rad)
        "beta_error": None, #(rad)
        #Damping time and error for 10keV
        "10keV_tau_damp": None, #(s)
        "10keV_tau_damp_error": None, #(s)
        #tuple of a,b,c for quadratic,linear and constant terms
        #for fitting B dependence on energy
        "B(Energy)_coeffs":(None,None,None), #(T/keV^2,T/keV, T)
        #And errors in above in the same order.
        "B(Energy)_coeffs_errors":(None,None,None), #(T/keV^2,T/keV, T)
        }
    return results
    return results
```

Your code will be used as a *Python Module* meaning that any code that is only to be run when testing it should be protected from being run when imported like so:

```
if __name__=="__main__":
    # Put your test code in side this if statement to stop
    #it being run when you import your code

    filename="My Data File.txt"
    test_results=ProcessData(filename)
    print test_results
```

6 Assessment

Your code will be assessed according to the sheet on the VLE, both automatic checking and manual checks by the demonstrators will be used. As the automatic tester will be importing your code and then running the ProcessData function, it is important that your code does not do anything unexpectedly when imported.

1. Make sure any module level code is not run when imported by using the code snippet above to test if the code is being run or imported.

2. Do not use **raw_input** or other python commands that will request inputs from the user - when run with the automatic checker there is no user to type anything in !

The demonstrators will provide a grade (1st, 2.1, 2.2, 3rd, fail) for each assessment criteria. These will be reviewed by the module leader and converted into a numerical mark for the module.

A random sample of 15% of the submissions will be second marked by the module academics.

You will be sent the demonstrator's grade sheet and comments around the time at which the semester 2 exam marks are released – you can find a copy of the feedback sheet on the VLE too. You will also receive a copy of the results of testing your code with the automatic tester.

Your code will be tested with the data you submit and a similar data file with different parameters.

7 Other Rules

1. You may use any of the standard Python library modules, numpy, scipy and matplotlib for this coursework. The code will be tested using the versions of these modules as are installed on the cluster computers.

2. All submitted work for assessment must be your own individual work unless otherwise declared. In particular, making use of code developed by another person, or jointly with another person **and not declared to be so in comments in the code** is plagiarism within the University's definition and cases will be pursued to the fullest extent and may result in loss of credits, a record of misconduct in your permanent file or other such penalties as permitted by the University regulations.

3. That said, you may make use of code developed either in collaboration with, or entirely by, others, **so long as you declare this in docstrings and comments in the functions as relevant.** Code which has been developed by others, but used correctly by you will receive a pass mark, code developed in collaboration with others will receive a higher mark but reflecting the number of collaborating authors. Code examples taken from the official python, numpy, scipy and matplotlib documentation does not require attribution, all other code taken from the internet does.

4. Your report must be entirely your own individual work. Instances of copied or collaboratively written reports will be considered to be plagiarism.

   If the entirety of your code has been developed by someone else, then whether you pass or fail will depend entirely on whether your report fully and correctly describes the code you have submitted. Any omissions and errors in the explanation will be marked down. Even a perfect report in these circumstances cannot result in a mark higher than 50%

7.1 Examples of how to cite copied and collaboratively developed code

Code developed by another student or demonstrator:

```
def CopiedFunction(parameters):
    """This function was written by Joe Smith (py12js)
    ...rest of the docstring for Joe's function."""
```

Code copied from the internet:

```
def CopiedFunction(parameters):
    """This function was copied from user gb119 on StackOverflow
    http://stackoverflow.com/questions/
    19735670/how-can-i-adapt-to-python-django,
    ...."""
```

Code developed with other students:

```
def CollaborativeFunction(parameter):
    """Function written jointly with Joe Smith (py12js) and
    Jane Doe (py12j2d)
    ..."""
```

8 Hints and Tips

To complete all the parts of the assignment with all of the possible complicating features present in the data is a significant challenge. You may find the following suggestions helpful.

- Start simple. Do not try to go and solve the problem for the most complex set of data first. Start with un-damped, single energy dataset. Adding damping will make the fitting harder as it introduces an extra unknown. Working with multiple energies requires additional steps to extract the correct data (but does also allow all of the tasks to be done).
- Dealing with damping in the asymmetry ratio is the hardest part of the assessment to get working well. I strongly recommend doing everything else first and then attempting to include the damping.
- Generating the data files does take some time - especially if lots of other students are doing so at the same time. Be patient, don't keep hitting submit, it won't help make it go faster. Check your downloads folder for the saved files.
- The marking scheme rewards both well written and structured code as much as code that does everything, so you may be better off with code that doesn't handled all the features, but is well written and robust for what it does do. Remember, the underlying point of this module is to use well structured and maintainable code and significant marks are to obtained by demonstrating this.
- To produce histograms, you might want to investigate the **matplotlib.pyplot.hist** function or the **numpy.histogram** function. You will need to choose a fixed array of bins to ensure that you have the same time bins for both detector signals - otherwise you cannot add and subtract the two detector signals to calculate the asymmetry.

  The histogram functions will return a list of integers for the counts in each time bin - you will need to convert them to floats as otherwise your asymmetry

function will do integer division and get incorrect results. Look at the astype() method of a numpy array to fix this.

- When fitting data it is often necessary to give the fitting routine a rough idea of a starting point for each parameter. By default 1.0 is often used, this may not be an appropriate number. The numbers used in this coursework, whilst randomly chosen, are reasonable for a real experiment.
- You will also need to consider error propagation in your code. For counting statistics (like you are doing here), the standard error should go as the square-root of the number of detector counts for each time, so that $\delta P_L = \sqrt{P_L}$ and $\delta P_R = \sqrt{P_R}$. Working out the error in the asymmetry is a little trickier than using the standard rules of error propagation as the numerator and denominator in 2.5 are not independent. The combined error can be found by combining the partial differentials of the asymmetry function 2.5 with respect to P_L and P_R to give:

$$\delta A = 2\sqrt{\frac{1}{\left(P_L + P_R\right)^4}\left(P_L^2 \delta P_R^2 + P_R^2 \delta P_L^2\right)}$$

- Other than the point above, the error analysis is the same as you have done in first (and second) year labs.