In [1]:
```
# This is just getting some formatting information and is not part of the
from IPython.core.display import HTML
import urllib.request
def start_css():
    with urllib.request.urlopen('https://github.com/uolphysicsteaching/re
        return HTML(response.read().decode("utf-8"))
start_css()
```

Out[1]:

In [3]:
```
%%javascript
MathJax.Hub.Config({
    TeX: { equationNumbers: { autoNumber: "AMS" } }
});
```

PHYS2320 Computing 2 Assessed Coursework 2022-23

Contents

8 Hinsts and Tips

Some pointers towards completing this assessment.

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 advised by Dr Burnell). Please remember that marks are awarded for the structure, style, and robustness of your code and for your planning and execution lof the assignment as well as for its functionality. Please see the mark sheet and grading criteria on Minerva 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 Minerva **by 2pm on Thursday 30th March 2023**. Late submissions will incur the standard University penalty of 5% for each day and part of day that it is late.

Extensions for mitigating circumstances such as prolonged sickness or other events that you could not reasonably foresee can be applied for online. See https://students.leeds.ac.uk/info/10111/assessment/860/mitigating_circumstances .Please note that failure of IT equipment is **not** generally regarded as grounds for an extension or other additional consideration (unless the failure is with University IT systems) - so you should make sure that you keep a backup of your work on off-site or cloud storage, such as OneDrive or Googledrive.

The maximum length of an extension is 2 weeks. If you are unable to submit your work within the 2 weeks then it will not be marked for the semester 2 exams and you **must** apply for a resit of the coursework. Resits will be due in early August 2023.

We hope to return a feedback sheet to you around the end of teaching in Week 11 of the semester.

1.2 Weekly Reports

There are no set times when you must work on the coursework, all the timetabled sessions are optional drop-in sessions and you are not penalised for not attending. You are, however, expected to work on your coursework each week and you must submit a weekly progress report that briefly details what you have done and what you plan to do. These progress reports are assessed.

> The weekly progress reports will contribute 10% of your mark for the coursework. They should be specific about your activities towards competing the coursework assignment for each week. For example, mentioning which tasks you have been working on, the progress made and difficulties encountered. The markers will be looking for evidence that you had formed a plan for writing your code that including milestones or checkpoints against which you evaluated your overall progress and that you were working on throughout the semester and that your reports relates to the code submitted.

To complete your weekly report, you must go to the module website: https://phys2320.leeds.ac.uk/ and click on the *Progress Report* link and complete and save a report. You must do this **at least once each week between 6am Monday and 10pm Friday**. It is not possible to submit a report late or to obtain an extension. If you have been unwell for the whole week and so have not been able to complete University work, then you should inform the Physics taught student office in any case. You should ask them to confirm to Dr Burnell that you were unable to do your University work and he will ensure that an "absent" report is filed. We recommend that you set a recurring appointment in your calendar to remind you to submit the weekly report - particularly if you plan on mainly working on it at the weekend when you cannot submit reports.

1.3 Time Allocation

Although you are not required to attend the lab sessions, as with semester 1, there will be demonstrators available at the timetabled sessions:

- Mondays 12 noon-1pm (please note that the sessions are in Cohen in weeks 1-4 and 8-9, Esther Simpson in week 5 and Chemical and Process engineering in weeks 6 and 7)
- Tuesday 1pm-2pm in Fourman O and P clusters (all 9 weeks).

In addition, Dr Burnell will be available for 1:1 Teams calls between 11am-12noon on Tuesdays and 2pm-3pm Thursdays. Appointments for these calls can be made through Minerva – see the Semester 2 Coursework folder.

Outside of these times you can also ask questions on the forums on the module website and Dr Burnell will aim to answer questions posted there within two working days.

> To avoid answering the same question repeatedly, only questions which are confidential in nature, such as mitigating circumstances - ill health, family bereavement and the like will be dealt with by email. Emailed questions that are not about confidential matters will not be answered and should be posted on the forums so that other students can benefit from the answer too. Questions can be posted anonymously on the forums.

> If you have an extension, please note that the level of support available after the main deadline on the 24th March will be substantially reduced due to the vacations and the move of some of Dr Burnell's research equipment to the new building.

It is expected that this assignment will take up to 60 hours of programming time over the semester, so you should expect to spend a substantial amount of time each week on it.

1.4 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 Minerva.

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 written by Dr Burnell.

The code you write will be used as a Python **module** by the marking code — **a template is provided that you must use** and full details of the structure that you must follow are in section 5.

Your code and report will then be assessed by one of the module demonstrators who will grade it for *structure, style, robustness* 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, $\mathbf{M}$ is placed in a magnetic field H, classically we expect it to line up due to a torque:

$$\tau = \mathrm{M} \times \mu_0 \mathrm{H}$$

. If, however, 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:

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

where γ is called the gyromagnetic ratio, which for a μ^+ is 851.616,\mathrm{Mrad s^{-1} T^{-1}}.

Equation 2.1 describes a circular motion of the moment M about B with a frequency given by:

$$\omega_{Larmor} = 2\pi\nu_{\mathrm{Larmor}} = \gamma|\mathrm{B}| \tag{2.2}$$

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.

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.

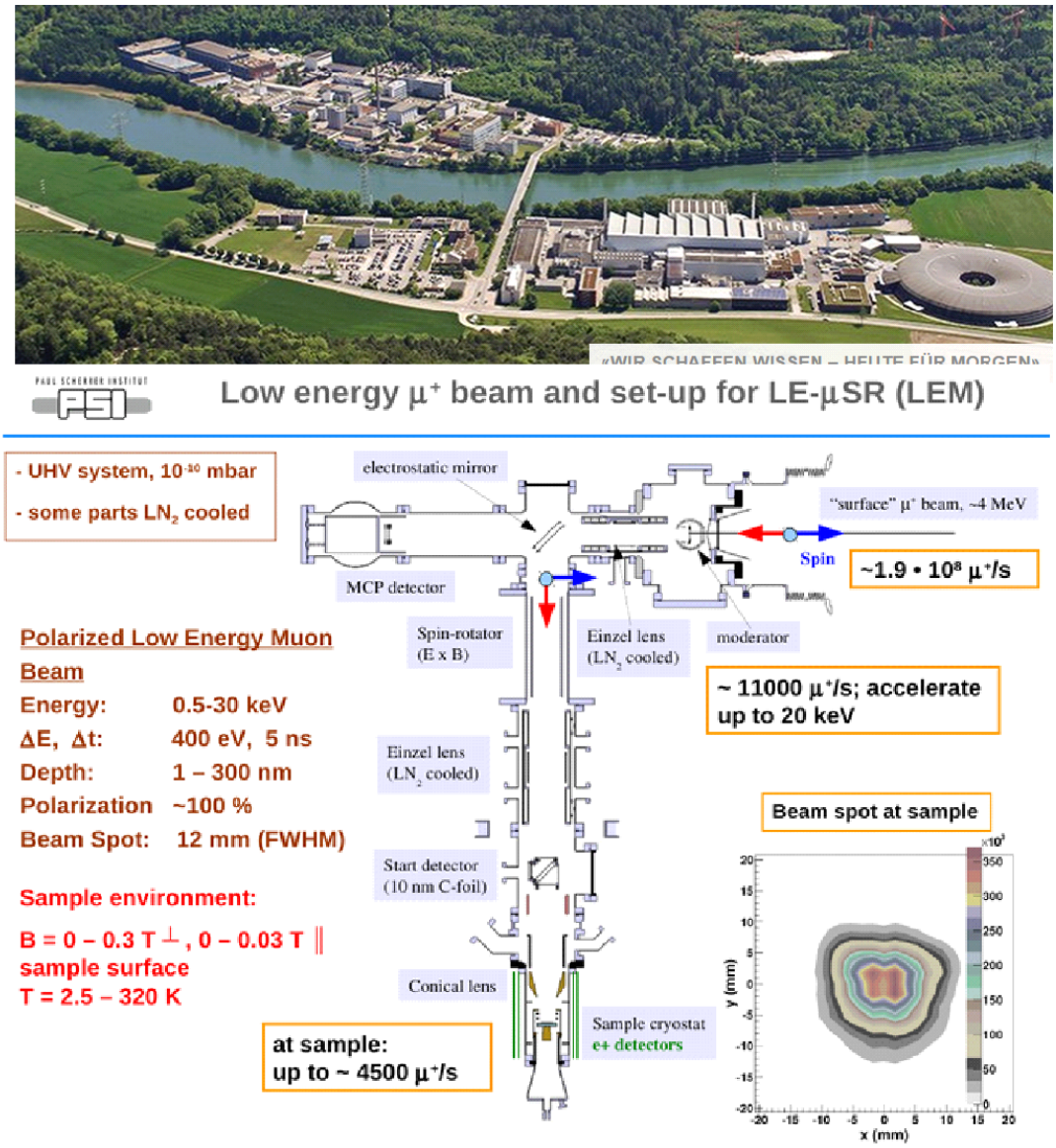

Figure 1 The low energy μ^+ beam line at the Paul Scherrer Institute near Zurich, Switzerland. This facility is almost unique in the word in being able to create muons with relatively low energies that can be implanted into thin magnetic films to study the depth dependence of magnetism at interfaces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

Figure 1 shows an illustration of a real μ^+ beam line facility based at the Paul Scherrer Institute near Zurich, Switzerland

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.

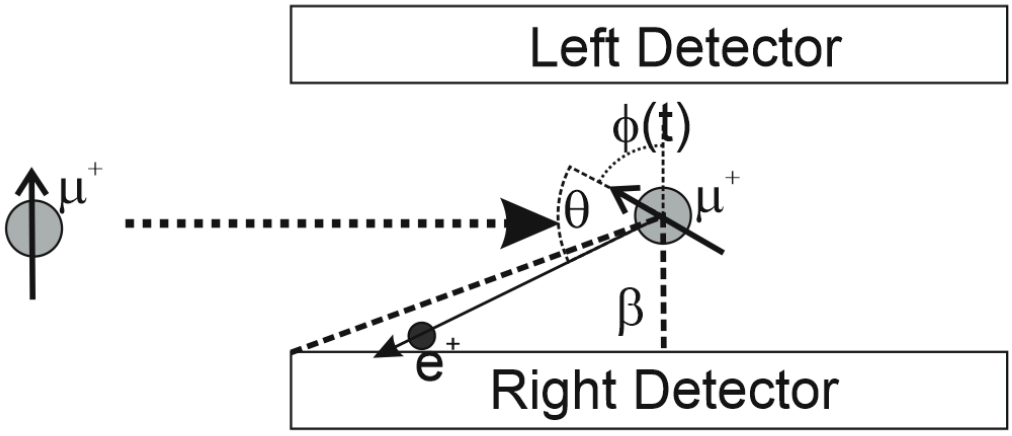

Figure 2: The geometry of muon decay and positron detection in a low energy muon spectroscopy experiment. β is a measure of the size of the detectors, giving the angle covered by each detector, θ is the angle relative to the muon spin direction that the e^+ positron is emitted and $\phi(t)$ is the angle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

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.

$$P_L(t) = \frac{\beta}{2\pi\tau_\mu}(1 + A_0(t))\, e^{-t/\tau_\mu}$$

$$P_R(t) = \frac{\beta}{2\pi\tau_\mu}(1 - A_0(t))\, e^{-t/\tau_\mu}$$

$$A_0(t) \qquad = \frac{P_L - P_R}{P_L + P_R} \qquad = \left(\frac{-1}{3}\right) \frac{\sin(\phi(t) - \beta) - \sin(\phi(t) + \beta)}{2\beta}$$

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:

$$\phi(t) = \omega_{Larmor} t = \gamma \mathrm{B} t$$

with B the magnetic field\footnote{As is common in the field, we do not distinguish clearly between a magnetic field and a magnetic flux density. Strictly \mathrm{B} is a flux density given by $\mathrm{B} = \mu_0(\mathrm{H} + \mathrm{M})$.} and $\gamma = 851.616\,\mathrm{Mrads^{-1}T^{-1}}$.

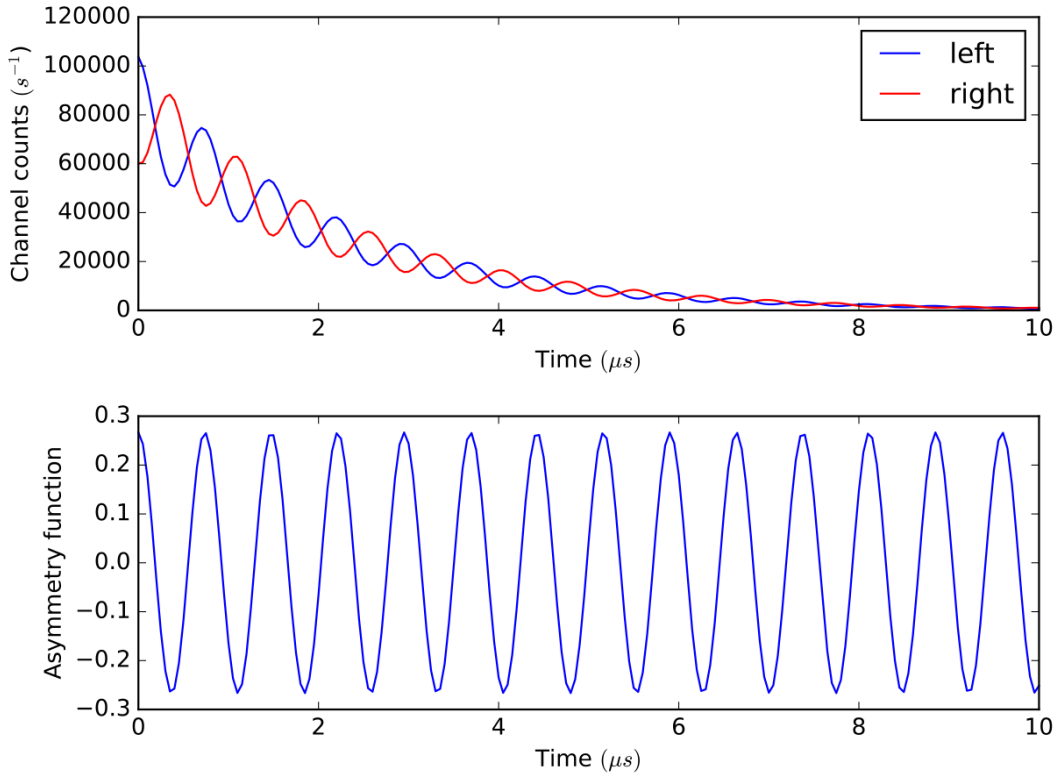

Figure 3: Top: calculated left and right channel signals, from equations 2.3 *and* 2.4*, and bottom: calculated asymmetry signal from equation* 2.5*. For both plots, the magnetic field* $\mathrm{B} = 10\,\mathrm{mT}$*, detector angle,* $\beta = 1.131\,\mathrm{rad}$*, gyromagnetic ration* $\gamma = 851.616\,\mathrm{Mrads^{-1}T^{-1}}$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

De-phasing

Equations 2.3 to 2.5, assume that the magnetic field, 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:

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

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

3 Data

For this coursework you are provided with a set of simulated data that can be downloaded from the link given on Minerva. Like the data you used for the Planck's Law classes, the exact parameters are determined based on your username.

The website lets you select either a practice set of data, which will contain a list of all the parameters used to determine your spectra so that you can check whether your code finds the right answers. Each time you download a different set of practice data it will use a different set of parameters so you can check that your code is not overly optimised to only work with one set of data. Alternatively you can select the assessment data where the parameters are not given in the file (but are always the same each time for your username !).

- **Mode**: this has two options, *practise* and *assessment* as discussed above.
- **Variable Metadata**: This option causes a random selection of additional metadata to be added to the file and for the order of the metadata to be randomised. If you select this option, you file reading code will need to b e able to deal with this situation.
- **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 \units{\mu s}) 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 writing 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 IT user name (*ie* py21...) 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 $\mathbf{B}$ 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 $10\,\mathrm{keV}$ 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 IT user 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 $\mathbf{B}$ varies with implantation energy). The magnetic field $\mathbf{B}$ should have a quadratic dependence on the energy, so fit your extracted values of $\mathbf{B}$ to a function of the form:

$$|\mathbf{B}|(E) = aE^2 + bE + c$$

   , where a,b,c are constants. Overlay this quadratic fit on your plot. Your plot should include your IT user 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 when run on your assessment data file that you submit. **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 Minerva by the deadline.

1. **The data-file** used to generate the data in your report as downloaded from the website. Your code will be tested with this data-file 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. (You can check this for yourself by trying your code out on someone else's data)

2. **The report in pdf format** (if you prepare it in Word then you can use the Save As,.. option and select pdf format to generate your pdf)

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 (e.g. *py19spqr.py*). Your code must have a function called **ProcessData** that takes a single parameter that is the name of the data file to be loaded and processes. Further details of what **ProcessData** should do are given in the sub-section below. A template of a suitable file is on Minerva and you should make use of this.

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 **floating point number** or **None** for each key. Do not change or delete the keys otherwise the marking code will not be able to mark your code and you may lose marks.

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

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

    #This is the dictionary of results. Replace the None values with vari
    #Do NOT remove or rename any of the dictionary entries, if you code d
    #returning None here. Take Note of the units in the comments!
```

```
    results={"10keV_B":None, #this would be the magnetic field for 10keV
             "10keV_B_error":None, # the error in the magnetic field (T)
             "beta": None, #Detector angle in radians
             "beta_error": None, #uncertainity in detector angle (rad)
             "10keV_tau_damp": None, #Damping time for 10keV (s)
             "10keV_tau_damp_error": None, #and error (s)
             "B(Energy)_coeffs":(None,None,None), #tuple of a,b,c for qua
                                                  #for fitting B dependen
                                                  #(T/keV^2,T/keV,T)
             "B(Energy)_coeffs_errors":(None,None,None), # Errors in abov
             }
    return results
```

Your code will be used as a *Python Module* - the autograder will do the equivalent of:

In [ ]:
```
from py21spqr import ProcessData
test_file="some_file_from_the_autogradeer.txt"
student_results=ProcessData(test_file)
```

as a result you do not want to have any code not contained within functions as otherwise it will run on import and may crash the autograder (e.g. if it refers to files that exist on your computer, but not the autograder). To avoid this happening, but still allow you to test your code whilst you are developing it, the template includes a block like the one below:

In [ ]:
```
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)
```

A *coursework checker* python script will be made available during the semester to enable you to check that your code has the correct **ProcessData** line and returns a correctly formatted dictionary (the checker script does not check whether your code returns the correct results!)

5.2 Notebook Files

The submission **must be a .py** python file. If you develop your code in a Jupyter Notebook then you can convert this to a .py file by selecting *Download as...* from the *File* menu and selecting the option to download as a .py file. **Simply renaming the file extension is not going to work and will cost you marks!**. In general we would recommend using the Spyder IDE (introduced in Lab 2) or some other similar IDE program to do this coursework.

Non-code cells will be turned into Python comments. To avoid this being confusing to the grades, we recommend that before you Download the notebook as a .py file, you delete all non-code cells from your Notebook.

5.3 Sample Report

A sample report that illustrates what needs to be included in your report is linked to on Minerva. Note that the sample report is based on a set of data with the all the features turned on - if you have fewer of the complicating features the plots in your report will have fewer fits shown. The sample report also annotates the plots with the best fit parameters correctly rounded. **Your code does not have to do this annotation and you do not need to have code to round correctly - but you must manually round any numbers you include in section A of your report.**

6 Assessment

Your work is assessed against several different criteria. The overall aim of this module is to encourage you to write code that is not only useful but also well structured and written in a way that makes it easier for you or someone else to understand how it works and to maintain it. To this end, you code is assessed for:

1. **Features**: Does your code complete the tasks requested of it? Code that does not meet the specification in terms of having the correct ProcessData function, or that has errors that prevent it running correctly will also receive fewer marks for this criteria.

2. **Structure**: Does your code divide up the various parts of the problem into functions that can be run and tested independently? Do these functions correctly take in data via parameters and return their outputs, limiting the inputs and outputs to just the information necessary for the task?

3. **Coding Style**: Is your code written in good Python style with sensible names for functions and variables, good docstrings for the functions and appropriate commenting.

4. **Robustness**: How does your code respond if the input data is not as expected? Does it interpret error conditions and return sensible and informative error messages. Does it check that the numbers it is reporting are reasonable?

5. **Progress Reports**: Do the weekly progress reports show that you made steady progress on the assessment and set yourself appropriate weekly goals and monitored your progress against a plan? Or did you leave everything to the last minute and rushed the work?

6. **Documentation**: Does your report present the results correctly. Are your plots suitable and visually attractive. Does your flow diagram have enough detail to explain your code fully and does it demonstrate that you understand how your code works?

Your code will be assessed according to the marking sheet linked to on Minerva, 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 **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 ! (Likewise there is no user to read anything printed to the screen)
3. Make sure that any **breakpoint()** function calls have been removed from the code before you submit it.
4. Make sure that your code is free of errors before you submit it - if it runs int he coursework checker then it should be ok.

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 (or third) marked to ensure consistency.

You will be able to download the demonstrator's grade sheet and comments from the module website (https://phys2320.leeds.ac.uk/accounts/summary) around the last week of lectures for semester 2 — you can find a blank copy of the feedback sheet on Minerva too. You will also receive a copy of the results of testing your code with the automatic tester.

7 Other Rules

1. You may use any of the *standard Python library modules*, *numpy*, *scipy*, *matplotlib* or *pandas* for this coursework. The code will be tested using the versions of these modules as are installed on the cluster computers for the Anaconda 2022.10 (Python 3.0) application.

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, other students, **so long as you declare this in docstrings and comments in the functions as relevant — see the examples below.** 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, matplotlib aor pandas documentation does not require attribution, all other code taken from the internet does.

4. If you pass a copy of your code or any part of your code to another student then you are advised to declare so in a comment in your code. Doing so will avoid a chance that you are accused of collusion if the other student submits your code without declaring it.

5. It is **not permitted** to employ, contract or buy code from any source in connection with this assignment even if declared as such. Any actions which could be reasonably interpreted to be attempting to do so will be considered as cheating and will be dealt with accordingly – even if code is not acquired or not submitted. Note that all materials relating to this assignment (such as this task sheet) are (C) University of Leeds and you are not allowed to distribute copies to anyone or allow others to make copies.

6. You may use code copied from the official documentation for Python, numpy, scipy, matplotlib or pandas, or copied from the materials for this module without needing to be acknowledged separately.

7. Your report must be entirely your own individual work without exception. Instances of copied or collaboratively written reports will be considered to be plagiarism even if declared to be so.

   Your report must be in **pdf format** – if you submit in other formats then it may not be able to be read by the graders and will get zero marks. even if it is readable, it will incur a penalty mark. **Note that simply renaming the file extension will not make it a pdf**, in most word processors you need to either do *Save as*... and select pdf format, or *Print*.. and chose a pdf output printer driver.

8. If you do not submit a data file, or submit a *practise-mode* data file, you will receive a penalty mark. Note that the features used in the data file that you submit are used to determine what features your code is tested with. If you submit a data file with a lower features rating than your code handles, you will not recieve the full marks that your code could have obtained. If you do not submit a data file at all we will make a reasonable guess at what data file features your code could handle.

9. If your report is missing either section A or B, then it will receive 0 marks for documentation.

> 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:

In [1]:
```
def CopiedFunction(parameters):
    """This function was written by Joe Smith (py12js)
    ...rest of the docstring for Joe's function."""
```

Code copied from the internet:

In [7]:
```
def CopiedFunction(parameters):
    """This function was copied from user gb119 on StackOverflow
    http://stackoverflow.com/questions/19735670/how-can-i-adapt-to-python
    ...."""
```

Code developed with other students (the other students shpuld have similar statements in their code):

In [6]:
```
def CollaborativeFunction(parameter):
    """Function written jointly with Joe Smith (py12js) and
    Jane Doe (py12j2d)
    ..."""
```

In cases of collaboratively developed code, you will get 40%+an equal share of the marks above 40% that the code merited. So in this example, if the function code was judged to be worth 70%, you would get 40%+(1/3)30% = 50%.

If you give code to another student it is advised that you put a comment *in your code* to claim original authorship. This will avoid you being accused of collusion if the other student uses your code without declaring its origin.

In [ ]:
```
defshared_function(parameter):
    """This function was given to Mary Brown (py21mrb) and Fred McFly (ed
    ..."""
```

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.

1. Make use of the template file from the start. Each year we see significant numbers of students who did not follow this advice and then struggled to make their code work when they placed it in the template in the last week/day/hour and as a result lost marks.

2. If you want to make the code stop and go in to debug mode in the middle of a function then there are two ways to do this:

    - Set a break point (F12 in spyder) on the line where you want to stop and then run your code in Debug mode (CTRL+F5 in spyder), or
    - insert a **breakpoint()** function call before the line where you want to stop, and run the code normally. Python will then stop at your breakpoint and

enter the debugger, letting you inspect variables and single step through code.

3. If the code terminates unexpectedly, you can use the %debug command to perform a *post-mortem* debug and follow the chain of function calls back, checking variable values on the way. This can often reveal where some variable has an unexpected value that can help focus debugging efforts.

4. Start simple. Do not try to go and solve the problem for the most complex set of data first. Start with a single energy, no damping and not variable metadata data set. Adding in variable metadata should be the easiest complication to code for, followed by multiple energies. Adding damping to the problem makes fitting the asymmetry data harder.

5. You get credit for having and executing a plan if you can demonstrate it in the progress reports. This might include:

   - setting realistic targets and milestones
   - setting a date/week by which you achieve each milestone
   - demonstrating that you review your progress and dapt/update your plans.

   Note, however, that the progress reports are each limited to about 4000 characters (including HTML codes!) for each of the two sections.

6. Think of the progress reports as a diary that you should update every time you work on your code. Every week it is like starting a new page of the diary, except you can't go back and change the page once a new one is started! If you plan on doing most of the work at the weekend, set yourself a reminder in the middle of the week to update your progress reports.

7. 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.

8. Read the marking Criteria sheet and make sure you understand what is going to give you credit.

9. The process of propagating the errors through your data analysis should have been taught in first year labs.

10. The template gives the expected units of each of the quantities — although the grading code will try to pick up errors here, it is not always perfect!

11. 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.

12. The file reading is similar to what was covered in semester 1, but with the addition of a header full of metadata. Working with Files Unit 3 demonstrates

some approaches to dealing with this.

13. Early on you will want to split the data into the two channels. Whilst you can do this by simply reading the data in a loop and appending values to a list for the left or right detector, this is incredibly slow and inefficient. A much more efficient method is to read all the data in first and then use the fact that you can index a numpy array with an array of boolean values to sort out which rows are in the left and right channels::

```
#Assume times and channels are 1D arrays of times and
# channel numbers
left=times[channels==1]
right=times[channels==2]
```

14. 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.

15. Choosing the right number of bins takes a bit of experimentation. Making the bins bigger means you get more counts per bin and hence smaller uncertainties in each bin and thus smaller uncertainties in the value of the asymmetry. However, reducing the number of bins, reduces the time resolution which results in getting the precession frequency less accurately.

16. 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 Equation 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:

$$\delta A = 2\sqrt{\frac{1}{\left(P_L + P_R\right)^4}\left(P_L^2 \delta P_R^2 + P_R^2 \delta P_L^2\right)} \tag{8.1}$$

17. The data in this coursework is generated with:

    - Magnetic fields, *B* positive and less than $30\,\mathrm{mT}$
    - Detector angle β is between $0.5\,\mathrm{rad}$ and $1.5\,\mathrm{mT}$
    - The damping time constant is between $2\,\mu\mathrm{s}$ and $10\,\mu\mathrm{s}$
    - The magnetic field will never exceed the above limits for any implantation energy

18. **Do not leave this until the last couple of weeks. This is a substantial task and you should make a strart straightaway - remember that there are 10% of the marks riding on completing weekly reports for each week until the hand-in deadline!** Do make use of the drop-in sessions to ask for help from hte demonstrators or the 1:1 surgery sessions with Dr Burnell. The drop-in sessions

in the clusters tend to get very busy after week 6 and sometimes there are more questions than the demonstrators can answer.