CP Activity 1

Plotting and Basic Data Analysis

Hand-out: 20 Feb 2023 **Hand-in:** 27 Feb 2023

1.1 Session Outcomes

The outcomes of this introductory session are to

- introduce the packages `numpy`, `scipy` and `matplotlib`
- learn your way around the online help for Python and its add-ons
- develop basic plotting skills
- apply programming to the analysis and fitting of data

Those needing more assistance than is provided in these instructions, should work through the sections of the CP Activity 1 VULA lesson in parallel with these instructions. This VULA lesson contains links to all of the data files in this activity.

1.2 Some Preliminaries

Import Statements in PHY2004W

In PHY2004W we will use the `numpy`, `scipy` and `matplotlib` modules extensively. We will follow the recommended practice of using an import style that keeps each module in its own namespace:

```
import numpy as np
import scipy as sp
import matplotlib as mpl
import matplotlib.pyplot as plt
...
# we will sometimes need more import statements
...
```

This practice is preferred over:

```
from pylab import *
```

which imports the `numpy` and `matplotlib.pyplot` modules into the program namespace. With each module in its own namespace, we have to call each method (or function) with the name assigned to its module. As an example, if we were to use trig functions (included in the `numpy` module) we would require:

```
a = np.cos(1.5)        # 1.5 is measured in radians
```

Introducing Arrays

Many scientific programming tasks use arrays. Arrays are objects that computer science uses to keep things together which are similar (e.g. all floating point numbers[1]). You can think of them as a sequence of values $x_0, x_1, x_2, \ldots, x_{N-1}$. Note that there are N values, indexed by an integer, in this case $0, 1, 2, \ldots, N-1$. In Python, these subscripts are written in brackets, e.g.:

```
x[0]              # the first element of the array
x[i]              # the element with index i
x[1003]
```

We refer to the subscript, e.g. i, as an index of the sequence (or array). Arrays in Python (as in many computer languages) are indexed starting at 0. The number of elements in an array whose last index is N is thus $N+1$.

In order to use arrays in Python, we have to import the `numpy` module. Usually, we will have to create arrays (of the right length) and then fill them with values. There are several "helper functions" to achieve this. Here are some examples:

```
x = np.array([10,1,32,13,4])          # array from list of numbers
x = np.zeros(N)                       # array of zeros
x = np.ones(N)                        # array of ones
x = np.linspace(0.0,2.0*np.pi,100)    # 100 points linearly spaced in [0,2pi]
x = np.arange(0.0,10.0,0.1)           # elements 0.1 apart in [0.0,10.0)
a = x[2:5]                            # subrange x[2], x[3], x[4]
```

1.3 Basic Data Analysis

Let us now look at an application revising some of the basic data analysis techniques of PHY1004W. We have 60 repeated readings of the same physical quantity x written to file (`Activity1Data.txt` on VULA). An array would provide convenient storage for the readings, but first the values must be "read-in".

Reading in Data from File

! Download the code below from VULA and ensure that you can follow each line.

```
import numpy as np

f = open('Activity1Data.txt','r')     # open file

header = f.readline()                 # read and ignore header

data = np.zeros(60)
i = 0

for line in f:                        # loop over lines
        line = line.strip()
        columns = line.split()
        data[i] =  float(columns[1])  # extract data
        print(i,data[i])
        i = i + 1
```

Remember to give your Python programs meaningful names; don't forget the `.py` extension.

[1]Floating point numbers are a computer's approximation to the real numbers

Make sure that your code has the expected output before proceeding. $\boxed{!}$

Averages, Variances ... with `numpy` and `scipy`

It is not surprising that the Python add-on modules `numpy` and `scipy` have built-in functions for analysing data. In `numpy`:

```
import numpy as np
...
np.mean(data)    # returns the mean of readings in the array called data
np.var(data)     # returns the 'biased' estimate of the variance
```

Refer to `https://numpy.org/doc/stable/` for help with `numpy`. In fact, it is worth bookmarking this page.

In `scipy`, much of the data analysis is achieved through a single function which returns multiple outputs:

```
import scipy as sp
import scipy.stats as stats
...
n,(xmin,xmax),m,v,s,k = stats.describe(data)
```

You will have noticed that a separate import statement is required for the `scipy` statistics submodule. Documentation on the `scipy` module and all of its submodules (like `scipy.stats`) can be found at `http://docs.scipy.org/doc/scipy/reference/`. Use this documentation to see what values are returned by `stats.describe` – the variable names in the code above will give you some idea. $\boxed{!}$

Your task is now to extend your program to analyse the x data read-in using `numpy` and `scipy`. Your program should print to screen the outcome of the measurement i) using `numpy`, and ii) using `scipy`. Are the results identical? They should be! $\boxed{!}$ $\boxed{?}$

1.4 Plotting Data with Matplotlib

The module `matplotlib`, which produces publication quality plots, will be our preferred choice for plotting in this course. Rather than attempting to provide an exhaustive reference here, please refer to the online help available at `https://matplotlib.org/stable/index.html`. The extensive gallery of plots is particularly useful. Most examples include the sample code.

Suppose that in the process of working in the lab, you arrive at the following set of data:

x (m)	y (m)
1.21 ± 0.16	10.2 ± 2.1
2.04 ± 0.10	15.3 ± 3.2
3.57 ± 0.13	19.8 ± 2.6

Your task is to write a Python program to reproduce (exactly!) the plot shown in Figure 1.1 and to save the figure in the eps file format using `plt.savefig(filename)`. Find the closest example to the plot in Figure 1.1 in the online documentation gallery and identify the plotting function used in the source code. Then find the description of this function by performing a search on the function name. It is very important that you acquire the skills to understand online documentation. If you do not find details on how to adjust other features of the plot $\boxed{!}$

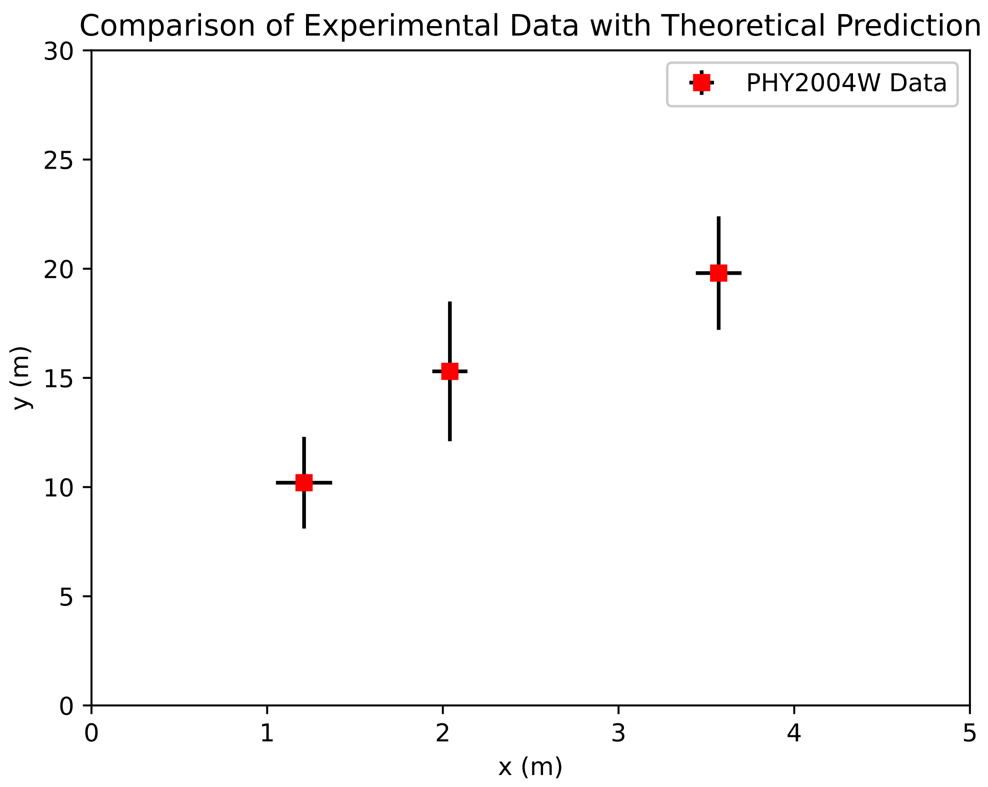

Figure 1.1: Plot to be reproduced exactly (red square data markers, black error bars etc.).

(such as legend options and axis ranges), explore the source code for some of the other plots in the gallery. The purpose of this exercise is to encourage exploration, so don't get frustrated!

1.5 Fitting a Model to Data

Unweighted Linear Least Squares Fitting

In PHY1004W, the method of least squares was introduced when fitting data (x_i, y_i), without uncertainies and with index $i = 0, 1, 2, \ldots, N-1$, to a linear model $y = mx+c$. In this approach, the sum of the squares of the deviations $d_i \equiv y_i - (mx_i + c)$ is minimised. It can be shown (through differentiation with respect to the parameters m and c), that the best-fit m and c are given by

$$\begin{aligned} m &= \frac{N\sum x_i y_i - \sum x_i \sum y_i}{N\sum x_i^2 - \left(\sum x_i\right)^2}, \\ c &= \frac{\sum x_i^2 \sum y_i - \sum x_i y_i \sum x_i}{N\sum x_i^2 - \left(\sum x_i\right)^2}, \end{aligned} \tag{1.1}$$

with associated standard uncertainties

$$\begin{aligned} u(m) &= \sqrt{\frac{\sum d_i^2}{N\sum x_i^2 - (\sum x_i)^2}\left(\frac{N}{N-2}\right)}, \\ u(c) &= \sqrt{\frac{\sum d_i^2 \sum x_i^2}{N(N\sum x_i^2 - (\sum x_i)^2)}\left(\frac{N}{N-2}\right)}. \end{aligned} \tag{1.2}$$

Using the data contained in `LinearNoErrors.txt` (available on VULA), write a program to determine the fit parameters and their uncertainties from Equations 1.1 and 1.2.

1.6 Putting it Together - Tasks for Submission

- Task 1: Extend the program you wrote to determine the average and variance of the x readings in `Activity1Data.txt` to calculate these quantities directly from their definitions (you may need to look these up).

  a) Compare these results to those extracted from `numpy` and `scipy`. Does this provide any clarity on what these built-in functions actually return, in particular the 'biased' and 'unbiased' estimates of the variance?

  b) You were asked to quote the result of the measurement of x (i.e. a best approximation and the standard uncertainty). Interpret the result by explaining exactly what information is contained in it. Also, make sure that you follow the conventions of quoting the result to the correct number of decimal places.

  c) Results of experiments are often quoted with a 'confidence interval'. What do you understand by this term? Does your quoted result have an associated confidence interval?

- Task 2: Perform an unweighted linear least squares fit on the data plotted in Figure 1.1.

  a) Quote the results obtained for the best-fit gradient and intercept (again, quote the results as intervals in a way that respects the conventions regarding the number of decimal places in measurement results).

  b) Include the best-fit as a blue line over the domain $[0.5\text{ m}, 4\text{ m}]$ in an extension of Figure 1.1.

1.7 Summary and Hand-in Instructions

The purpose of this activity was to get rid of the Python cobwebs and to introduce you to some features that will be useful throughout PHY2004W. Just like with any language, learning requires regular practice. It is hoped that you will make frequent use of the online help and that you will apply what you learn wherever possible in the PHY2004W experimental labs.

Many of the laboratory activities in PHY2004W will require a report. It is expected that these be typed, with all graphs produced by a plotting package such as `matplotlib`.

To practice these skills, type up your answers to the two tasks in Section 1.6, including all relevant output (figures etc.). Your submission should be 2-3 pages in length. In addition, submit your Python code for each of the two tasks in Section 1.6 on VULA.