**Lab 2**

**Statistics and Measurement Virtual Laboratory:**

#### You are strongly encouraged to read Chs. **1** - 3 in your text (I. Hughes and T. Hase, Mea­ surements and their Uncertainties) before attempting the laboratory. At a minimum, read sections 2.6 - 2.8 before your lab period . This will aid in your understanding of the experiment and your results, and will help you answer the direct questions asked of you in this manual.

#### It would also be helpful if you installed the Anaconda distribution of Python 3.6 on your own machine and worked through the document ‘GettingStartedWithPython3\_6.pdf’ that has been posted to the Python folder on myCourses before your lab period…. And maybe even attempted some of the tutorials that are referenced there! Assignment 1 can also provide some practice.

Python/Jupyter Specific Learning goals:

* How to make effective use of the help function inside Jupyter and the extensive online documentation that is available. These are the skills you need to become independent and thrive in the Python environment.
* How to access the computer’s operating system functions through Python (the OS library). Changing the current working directory (CWD), list CWD contents etc
* How to Import and Export data from/to Files.
* How to manipulate and perform mathematical operations on arrays of data in Python (the NumPy library)
* How to define a function in Python
* How to plot data (the Matplotlib library)

Statistics learning goals

* The difference between the *standard deviation* of a set of measurement outcomes and the *uncertainty in the mean* computed from those measurements.
* The influence of repeated measurements on the accuracy of the final experimental result.
* How to properly report the results of a measurement (significant figures and experimen­ tal uncertainties)

Getting Started:

This lab is a virtual experiment; data from simulated measurements will be analyzed using the Python programming language inside the Jupyter integrated development environment (IDE). The experiment serves as an introduction to Python/Jupyter and to consolidate many of the ideas from Statistics that we have been discussing in lab and lecture over the last two weeks. Specifically, the essential features of the Gaussian/Normal distribution and the standard error in the mean concept.

You will perform this lab in pairs using the laptops that are distributed throughout Trottier 3070. Log-in to the laptop and open MS Excel and Jupyter Notebook ‘Apps’. The lab session will begin with an introduction to the Jupyter Notebook environment and some detailed instructions on how to get started.

IMPORTANT NOTE ON THIS MANUAL: Any bold word inside single quotes (e.g. 'matplotlib.pyplot.**hist**') in this document refers to a built-in function. Python syntax is used to indicate both the library (os, numpy or matplotlib.pyplot) in which the function is found and the function name. ***You are expected to teach yourself the syntax of how to properly call and use each of these functions through the documentation available This is simply the best way to learn how to use Python and put yourself on a path to thriving in this environment…. Don’t be frustrated by this process! This is an essential part of this lab and an essential skill to develop. The learning curve will appear steep at first, but you will be amazed at how fast you pick it all up.***

**How to learn about built-in functions:**

1. Learn about built-in functions by using **help**()

For example, type the following in a Jupyter input cell: ‘**help**(numpy.mean)’. Hitting Shift-Enter will execute this instruction and list the documentation on the numpy **mean** function

1. If you prefer more examples and a colourful presentation, use the online documentation:

For numpy functions just do a Google Search. For example, Google ‘numpy mean’. The first hit is usually the right one. You can also check the numpy reference website for documentation <https://docs.scipy.org/doc/numpy/reference/routines.html>

For matplotlib.pyplot functions use the matplotlib reference site <http://matplotlib.org/api/pyplot_summary.html> You can enter the function name in the ‘Quick Search’ on the right.

This lab makes use of functions from several common Python libraries (os, numpy and matplotlib.pyplot) and a special function written specifically for our purposes. To properly initialize your Jupyter Notebook for this lab the following text should be copied and pasted into the first input cell:

import numpy as np

def data\_generator(mean, std, n\_measurements = 500, n\_samples = 20):

"""

Generate an array of measurements from a normal distribution.

Parameters

----------

mean : float

Desired mean value of the measurements.

std : float

Desired standard deviation of the measurements.

n\_measurements : int, optional

Number of separate measurements. Default is 500 measurements.

n\_samples : int, optional

Number of samples taken per measurement. Default is 20 samples.

per measurement.

Returns

-------

data : numpy.ndarray

Array representing the experimental data. Each measurement

(composed of many samples) is a row of this array:

-----------------------------------

meas1 | sample0 | sample1 | sample2 | ...

meas2 | sample0 | sample1 | sample2 | ...

meas3 | sample0 | sample1 | sample2 | ...

...

"""

return np.random.normal(loc = mean, scale = std,

size = (n\_measurements, n\_samples))

**Generating Your Virtual Data Set**

1. Open your data from Lab 1 (i.e. the ‘analog’ version of the Rolling Cylinder experiment) in an Excel spreadsheet. Your spreadsheet should consist of 5 columns of data, with 20 entries in each column. Remove any column titles/headers you may have added to this file and then save this data in ‘.csv’ format with a descriptive file name like “rollingcylinderdata’. Note the directory into which this data file is saved.
2. Read your data from the analog experiment into your Jupyter notebook. You will need to os.**getcwd**() to determine the current working directory (CWD), and potentially os.**chdir**() to change the working directory to the local directory in which you have saved your data file. You may want to os.**listdir**() to ensure that you have the CWD set correctly. Once you do, numpy.**loadtxt**() to read your data file to an array that you can work with in your Jupyter notebook. Give this array the variable name ‘data’; i.e. data = numpy.**loadtxt**(). Note: you are reading in a .csv file which is short for “comma separated values” format. As a result, you will need specify a ‘delimiter’ in your call to the **loadtxt**() function.
3. Look at the format of your data array. Does it make sense to numpy.**transpose**(data) so that measurements taken at different distances are organized by row rather than column? Note that each element of your data array is indexed and can be referenced independently. Input ‘data[0]’, ‘data[1]’, ‘data[0,1]’ and ‘data[1,1] (for example) to see how you can refer to different elements of your data array.
4. Determine the numpy.**mean**() and numpy.**std**() of your measurements for each rolling distance. Do these agree with your earlier calculations?
5. We will now use the computer to generate a large, simulated set of rolling times data: simdata = **data\_generator**(‘mean’, ‘std’), where ‘mean’ and ‘std’ are the mean and standard deviation of your measurements at one distance (your choice) . This computer generated data set is an array of 500 similar experimental trials, with each trial consisting of 20 repeated measurements that are statistically consistent with your measurements. Try to imagine that it was you that repeated the same experiment 500 times (rather than the computer), or that you collected data from 500 students who each performed the experiment once in exactly the same way. We will explore this data set in a way that demonstrates some key features of Gaussian statistics and the “standard error in the mean” concept.

It should not surprise you at this point in the course that each measurement leads to a quantitatively different outcome. You **do not** expect that two successive measurements of the same quantity - measured with the same apparatus in the same way - will be identical. What you do expect is that if you make a sufficient number of repeated measurements that a pattern will emerge. The measurement outcomes will follow a distribution with a well defined mean and standard deviation that tells you about the precision of the measurement you are making.

What about the results of the 500 experimental trials? If the 'result' of a trial is taken to be the mean of the 20 measurements made, do we expect the 'result' from each of the 500 trials to be identical? This would be a surprise given that essentially every measurement is different!

You will explore these ideas and the relationship between the 'measurement' and 'results' distribution through the analysis of this simulated data.

B) Analysis

Part 1: MEASUREMENT STATISTICS

The measurement and result distributions:

l. Plot a histogram of all the simulated rolling time measurements using ‘matplotlib.pyplot.**hist**()’. You will need to numpy.**flatten()** or numpy.**reshape**() your simdata array first. We call this histogram the distribution of measurements. Investigate the effect of the number of bins on the appearance of the histogram. Determine the mean and standard deviation of this distribution. numpy.**savetxt**() your histogram data.

1. Plot a histogram of the mean values computed from each set of 20 measurements. We call this the distribution of results. Investigate the effect of the number of bins on the appearance of the histogram. Determine the mean and standard deviation for the distribution of results. Numpy.**savetxt**() your histogram data.
2. ‘matplotlib.pyplot.**plo**t() an appropriately normalized Gaussian distribution with the same mean and standard deviation as the data distribution on top of each histogram. Appropriately normalized means that the histogram and the Gaussian should have the same area. To do this, you should ***define a function*** that computes the value of the Gaussian when you pass it an argument ‘x’, and parameters µ and σ. Make use of numpy.**linspace**() to generate an appropriate list of values (i.e. a 1D array) for the independent variable ‘x’. Pass this array to the Gaussian function you defined as the argument ‘x’ in order to generate the y-axis values for your **plot**().

Gaussian/Normal Distribution with Total Area = 1

1. How does the standard deviation of the distribution of measurements and the standard deviation of distribution of results compare? By what factor is the standard deviation for the results distribution smaller than the measurements distribution? Comment on the relationship to the formula , where σ is the standard deviation of the measurements distribution.
2. Consider the results distribution. What is the standard error in the mean of this distribution? Comment on the relationship to the formula , where σ is the standard deviation of the measurements distribution.

Statistics of your measurement distribution:

l. Make another histogram of all the simulated rolling time measurements. Use 6 histogram bins with each bin 1 standard deviation wide (i.e. the standard deviation of your measurement distribution) and the histogram centered at the mean of your measurement distribution. There should be 3 bins on either side of the mean in this histogram. Numpy.**savetxt**() your histogram data.

2. Based on the results of part 1, are these measurements consistent with the statistics of a Gaussian/Normal Distribution? What fraction of the measurements are within σof the mean? What fraction of measurements are within 2 σ of the mean . What fraction of measurements are within *3* σof the mean.

WHAT TO HAND IN FOR YOUR Lab 2 REPORT:

1. Your report for this lab will consist of your completed Jupyter Notebook; all the code, figures and necessary discussion. Note that the ‘Insert’ menu at the top of the Notebook allows you to insert additional cells into your Notebook above or below any existing cell. These cells can be formatted as text rather than Python input using the ‘Cell/Cell Type’ menu. Choose ‘Markdown’ rather than ‘Code’ if you want to make any Cell a ‘text-only’ Cell. This will allow you to insert discussion into your Notebook where you describe your code, discuss your results, and answer the direct questions that are asked in this manual.
2. Assignment 1, posted to myCourses