Introduction to the PC
The Exercise

Due January 22, 2018

January 10, 2018

Professor: David Cooke[1]
Teaching assistants: Aaron Mascaro[2], Andreas Spielhofer[3], Eli Martel[4], Ata Madanchi[5], Ziggy Pleunis[6]
Technician: Saverio Biunno[7]

1 Introduction

You should feel comfortable with the tutorial before attempting this exercise. A rough overview of this exercise is that you are given a two datafiles and you are required to manipulate and present the data as described.

2 The Exercise

2.1 Operations on Triplet Data

You are required to write scripts which will process the triplet data obtained by loading the file sinusoid.data and output the data in the same format. The general format of the data is three columns, independent variable x, dependent variable $y(x)$, and error on the dependent variable $?_y(x)$. In addition to performing the required operations on $y(x)$, your code must also calculate, via standard propagation of error, the uncertainties associated with the derived quantity, and when appropriate any adjustments that should be made to the independent variable.

The operations required are:

- square – dependent variable in output is the square of the dependent variable in input.
- square-root – dependent variable in output is the square-root of the dependent variable in input
- derivative – the output is the derivative of the dependent variable with respect to the independent variable. Consider the definition of the derivative.
- integral – the output is the numeric integral of the dependent variable. Consider the different methods of numeric integration.
- mean (RMS) – the dependent variable in the output is the square-root of the integral of the square of the dependent variable in the input, divided by the independent variable in the input.

[1]david.cooke2@mcgill.ca
[2]aaron.mascaro@mail.mcgill.ca
[3]andreas.spielhofer@mail.mcgill.ca
[4]eli.martel@mail.mcgill.ca
[5]ata.madanchi@mail.mcgill.ca
[6]ziggy.pleunis@mcgill.ca
[7]saverio.biunno@mcgill.ca

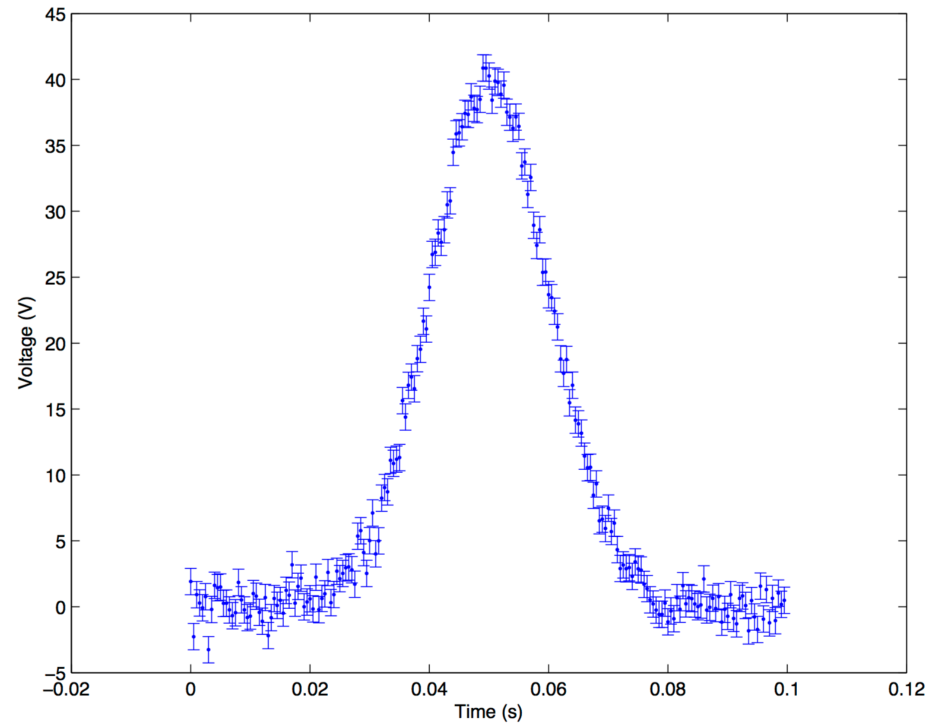

Figure 1: Data contained in file named `gaussian.data`

3 Analysis of Histogram Data

You are also required to write a script to analyze `histogram.data`. This function should calculate the following statistics:

- The mean number of jay-walkers in the intersection (row mean) for each day (row).
- The variance of the number of jay-walkers in the intersection for each day.
- The mean frequency of observation of a given number of jay-walkers over all days for all observed numbers of jay-walkers (a complicated way of saying column mean). Figure BLANK shows this data graphically.
- The variance in the frequency of observation of a given number of jay-walkers over all days for all observed numbers of jay-walkers (column variance).

4 The Format of the Datafiles

There are two data files which will be used in this exercise, `sinusoid.data`and `histogram.data`. These files can be copied either from MyCourses on the course website or from `mnt/resources/339/2015/intro`.

4.1 `sinusoid.data`

This file contains three columns, they correspond to the measurement of a signal. The first column is the time, in seconds; the second column is the voltage measured, the third column is the error on the voltage. The file `gaussian.data` has the same format, and is the file used in the worked example which follows.

4.2 `histogram.data`

For the purpose of this exercise, a camera is set up at the corner of Sherbrooke and McGill College, 100 times per day a photograph is taken of the intersection. At the end of each day the photographs are analysed and a histogram of the number of jay-walkers is generated. This experiment is repeated for 20 days. The first line of the file contains three numbers, the first number is the number of histograms (20), the second is the number of columns in each histogram (20), the third number is the number of samples in each histogram (100). The second and following lines in the file are the histogram data, one histogram per line. The first column represents the number of times zero jay-walkers were observed, the second column represents the number of times a single jay-walker was observed, the third column represents the number of times a pair of jay-walkers was observed, the fourth column represents the number of time three jay-walkers were observed.

Figure 1 shows the unprocessed data which was used in the tutorial. The size of the error bars are uniform over the entire range.

4.3 Error propagation

You are likely familiar with the following:

$$Y = f(X_1, X_2, ..., X_N) \tag{1}$$

then

$$\sigma_Y^2 = \sum_{i=1}^{N} \left(\frac{\partial f(X_1, X_2, ..., X_n)}{\delta X_i} \right)^2 \sigma_i^2 \tag{2}$$

The second part of the exercise will involve writing Python code to do various operations on vectors which correctly calculate not only the value, but also the uncertainty on that value via propagation of error. To facilitate this, I define the following structure to describe a set of values with uncertainty: a 2-D array, with n rows and 2 columns; the first column is the value, the second is the uncertainty.

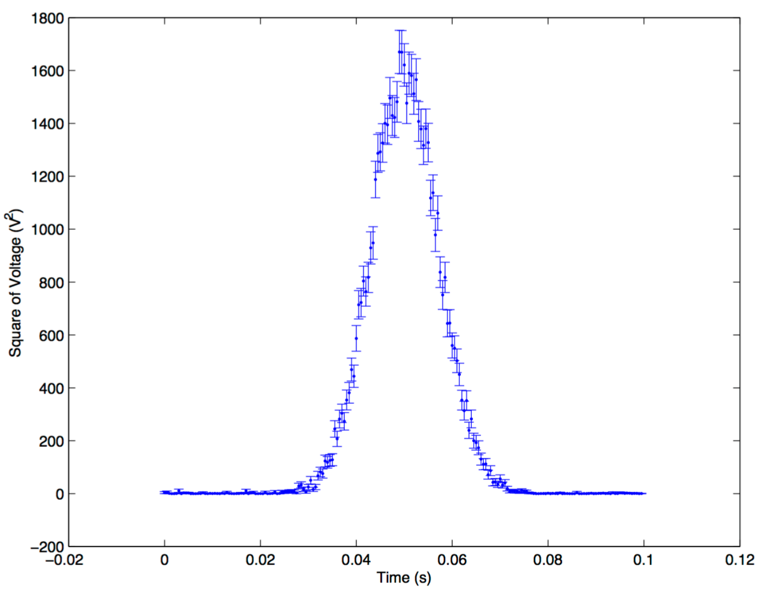

Figure 2: Data produced by performing squaring operation upon data contained in `gaussian.data`

.

So, without revealing the details, I present the following graph, shown in figure 2. Notice how the errorbars are no longer equal in size.

Presumably the nashing of teeth has subsided by this point. It is again trivial to perform the square-root operation, although you may want to consider to what data you apply it. Since we have a working squaring operator, it would make sense to verify that the square-root operation reverses the changes made by the squaring operation. Figure 3 shows the result of performing the square-root operation on data obtained by performing the squaring operation.

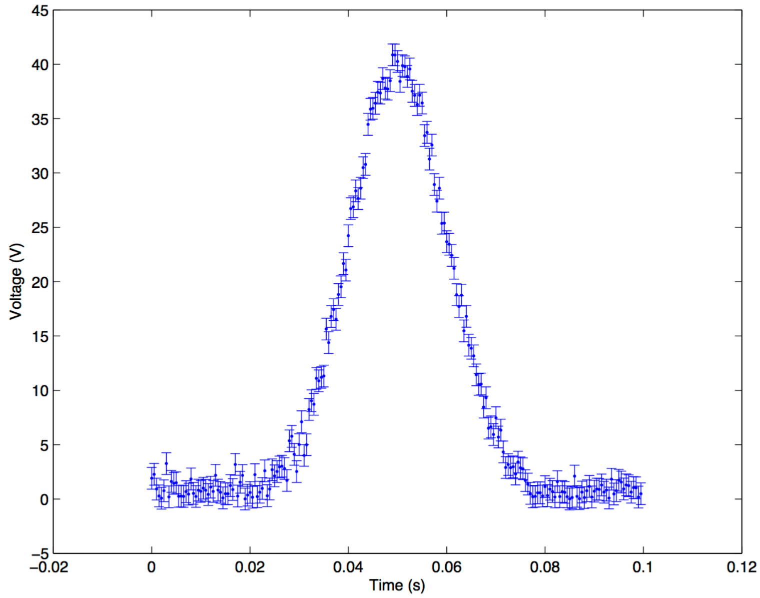

Figure 3: Result of applying square-root operation to data resulting from the squaring operation applied to `gaussian.data`.

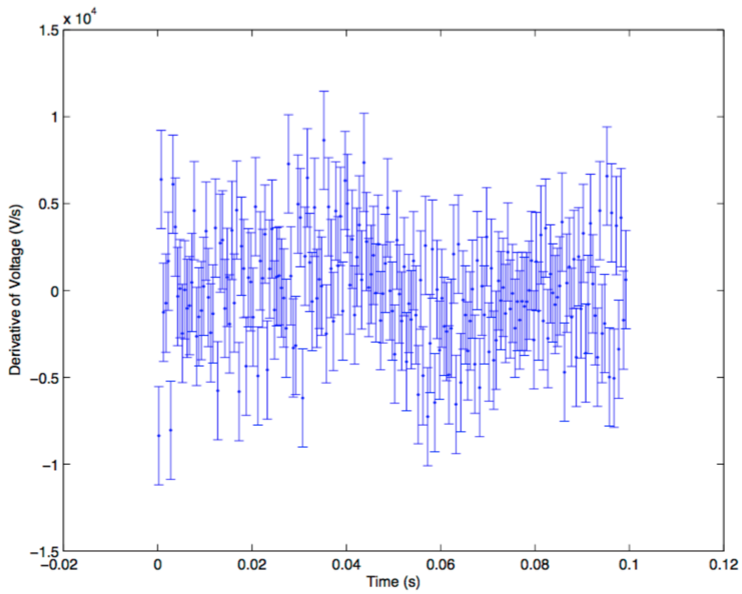

Figure 4: Result of applying derivative operation to `gaussian.data`.

While the original data has not been faithfully recreated by the square-root operation, this can be explained. It may be interesting to compare the effect of reversing the order of operations when you work with sinusoid,data.

Things start to get a little more hairy when we get in to the derivative. Dont get hung up on applying the partial derivatives used in the error propagation formulae to the derivative operation.2 Just write down the definition of derivative as introduced in Calculus and approximate the limit going to the spacing between adjacent points. Once you have a formula for calculating the derivative you can forget that it is a derivative, just identify any parameters which have uncertainties and propagate them. You should also consider whether the independent variable needs adjustment, that is, where is the approximation to the derivative most valid. The code for calculating the integral in the tutorial may be useful if you are having difficulty formulating the code to calculate the deltas. Figure 4 shows the application of the derivative operation to

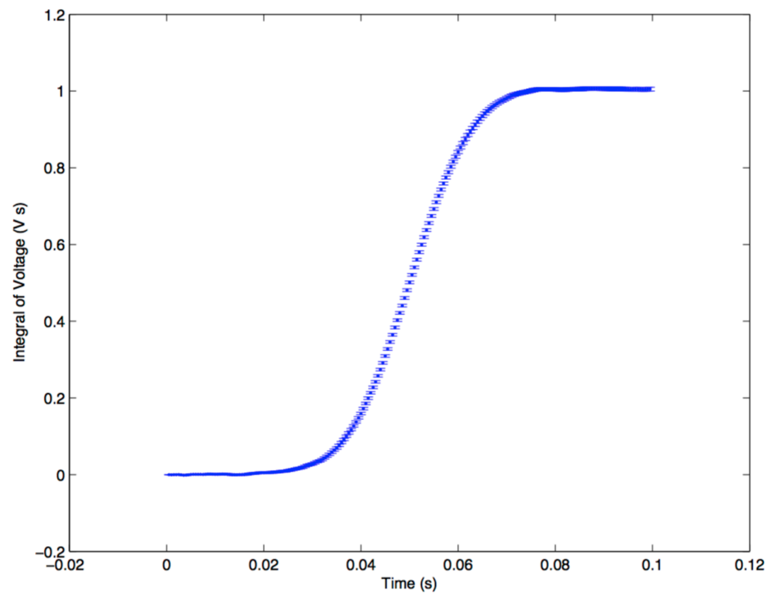

Figure 5: Result of applying integrate operation to `gaussian.data`.

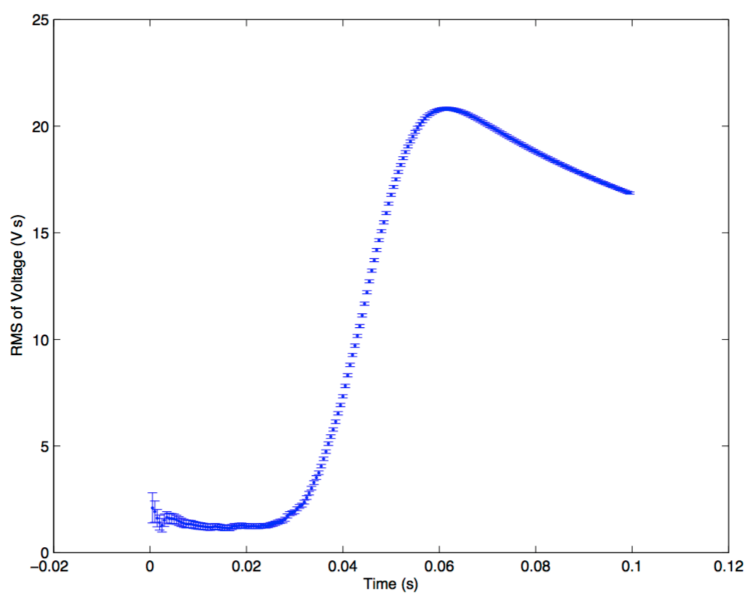

Figure 6: Root-mean-squared (RMS) of `gaussian.data`.

the original data set.

The integral is a bit of a pain. As seen in the tutorial, it is pretty trivial to calculate a single definite integral, but having an integral which evolves, that is

$$I(x) = \int_0^x f(z)dz \tag{3}$$

is not obvious without resorting to using a for-loop, or slightly mind distorting linear algebra, which depending on how optimized Python is in evaluating, may be very inefficient. Figure 5 shows the result

The error bars are a bit too small to see, however if you zoom in, you will see that they grow with increasing time. In fact, you will notice that the error bars become much larger than the scatter in the data. Does this indicate an flaw in the calculation of the error bar, or is it because the points in the processed data set are not independent random variables? What then, does the error bar on the integrated signal mean?

Figure 6 shows the result of applying the RMS operation to gaussian.data. If you are now dreading the most complex operation, because of its complexity, relax, due to your forethought in writing beautiful modular code, you have actually done all the hard parts.

5 What to hand in

You should prepare a short lab report which will contain:

- a title describing the subject of the report
- a list of authors
- an abstract summarizing the report
- a brief introduction describing the motivation for using the computer to perform data analysis.
- a brief theory section where you can elaborate upon the techniques used for example in integrating and differentiating the signal provided.
- a data section which must contain complete graphs of the following:
  - Graph of `sinusoid.dat` squared
  - Graph of the derivative of `sinusoid.dat`
  - Graph of the integral of `sinusoid.dat`
  - Graph of the RMS of `sinusoid.dat`
  - Table of the mean number of jay-walkers and the variance on this number for each of the 20 histograms in `histogram.data`.

    You should also include a brief discussion on anything you find interesting in the data.
- a conclusions section where you give your opinions on the results. Can you characterize the signal `sinusoid.dat`, based upon your experiences can you identify any possible sources for this signal?
- an appendix containing all programs written in order to complete the exercise. These programs should not need any external explanations since they will be wonderfully commented.

This is a short report, in that there is no physics to be discussed. However, it does cover all the technical issues in a lab report which can cost you marks before the actual physics content is addressed.

You will be given a sample report and a guide to report writing.