Statistics of Background Radiation

Hafsa Hassan

August 29, 2009

The internship has consisted of the following tasks:

1. Learning the basics of LabVIEW.
2. DAQ (Data Acquisition) and frequency measurements using a signal.
3. Writing a program for interfacing a GM tube and counting the radiation tricks.
4. Learning how to write a Latex document.

1 Learning the Basics of LabVIEW

The internship began on Monday, 22nd June, under the supervision of **Sohaib Shamim**. First, I had to learn the basics of **LabVIEW**, a graphical programming language that is used extensively in the Physics Lab.

We started with a simple LabVIEW tutorial, gaining some familiarity with the key features of LabVIEW, such as the Front Panel, Block diagram, functions toolbox etc. The first exercises required generating random numbers, using **for** and **while** loops, inputs and outputs, using controls and indicators, generating sine graphs etc. I then proceeded to using Boolean functions, comparison tools and gates, case and sequence structures, as well as using the different timing functions available. Exercises included building LEDs which would light when a certain time interval had elapsed, or when the generated random number was greater than 0.5 etc.

Further exercises included the use of multiple structures, as well as feedback and shift registers. Often an attempt was made to carry out the same exercise with two different methods. Most importantly, the repeated use of Help Search and debugging was practiced, which is essential to most programming languages.

In order to write programs for experiments where usually a lot of data recording is required, it is essential to know hot to write text files in LabVIEW. These required strong familiarity with array and string functions in the programming tool box. Here, it was important to gain some understanding of the different types of data encountered in LabVIEW, including numbers, one or two dimensional arrays, strings, boolean, waveforms and clusters, as well as their conversion and applications while using different case structures. For example, using a for loop changes the type of data: a 1D array becomes a 2D array outside the loop, whereas this doesn't happen in other programming structures. Similarly, while writing to text files, it is imperative to use the 'array to string' and the 'write to text file' functions inside the loop, whereas this is not necessary with for loops. These kind of insights are helpful while investigating logical and syntax errors.

2 Data Acquisition and Frequency Measurement

2.1 Data Acquisition

DAQ (Data Acquisition) is an important feature of LabVIEW. By performing an experiment and feeding the data into the computer using DAQ, we can record and analyze the results of any experiment on a personal computer. I learned about the signal conditioning system in use in the lab, the **SCC 68 Module**, its connections and the installation of the DAQ system. To begin with, we operated with a

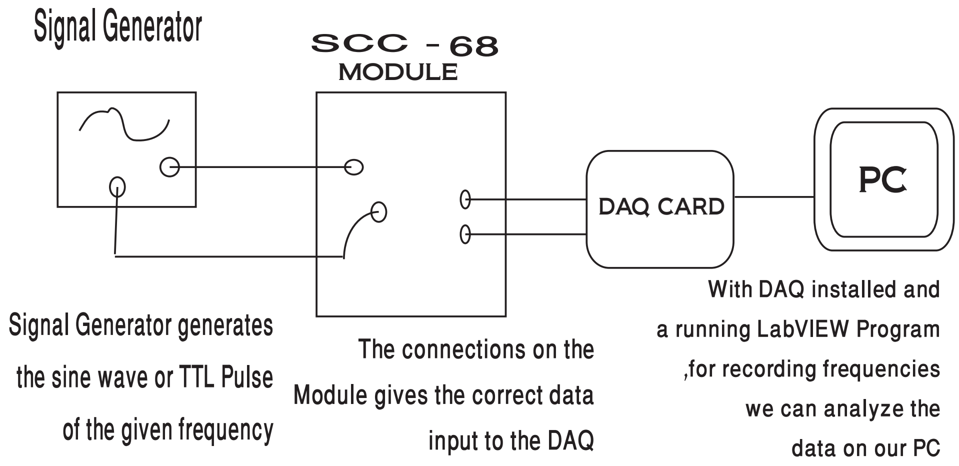
Signal Generator
SCC - 68
MODULE
DAQ CARD
PC
Signal Generator generates the sine wave or TTL Pulse of the given frequency
The connections on the Module gives the correct data input to the DAQ
With DAQ installed and a running LabVIEW Program ,for recording frequencies we can analyze the data on our PC

Figure 1: A sine wave generated by the signal generator.

signal generator as the input, while a **CRO (Cathode Ray Oscilloscope)** served as an output device along with the computer.

I was required to practice accessing data from the DAQ and using it in LabVIEW to write text files, generate sine waves, saw tooth or TTL pulses of different frequencies from the signal generator. Next, we required LabVIEW programs that could measure the frequency of the source. This exercise would be used extensively in the Radiation experiment.

2.2 Measuring Frequencies in LabVIEW

We commenced with constant frequencies. The first task was to find the frequency of a sine wave in a LabVIEW program. Then we would vary the frequency of the source, and see if the program responds accordingly. My program was based on the idea that a cycle consisted of sign changes on two occasions, so if any two consecutive sample points had a negative product, they indicated half a cycle (provided that the mean voltage was 0). Here, it is important to note that we were taking the input from the DAQ as *analog single channels, DBL (double precision float)*, rather than *analog multiple channels, DBL or waveform*, so for every iteration the program analyzed single values and not arrays or waveforms. With the number of cycles obtained and the use of timing devices, the frequency could be easily calculated. The program, tried with different frequencies, worked to a reasonable accuracy, even with saw tooth pulses, but had two important setbacks:

1. The frequency calculated started off with infinity, and then would decrease at a decreasing rate towards the actual frequency. This was possibly due to the fact that the time used up in the operations (with multiple loops and case structures) interfered with the frequency calculations. At first it seemed that it asymptotically approached the actual value, but it was observed that, given sufficient time, it could in fact decrease even below the actual frequency.

2. This only worked for constant frequencies and was inadvisable for further use anyway because it did not analyze all points in the sample.

Next, we were required to calculate the frequency of a TTL pulse (Transistor-Transistor Logic - a kind of a square wave). We made good use of the *Search Waveform* function, which, for an input of an array or waveform of sample points, with a given value and a tolerance level (acts like a confidence interval around that value), selects all the values, times and indices of sample points that lie within that confidence interval. From the size of these values in each bin, and given the time interval of that bin, we can deduce the frequency of counts in that bin. But there is an important consideration: while calculating the frequency, i.e the number of *cycles per second*, not every set of two points that lie in the tolerance level specified indicate a cycle - only those that have a trough (i.e, a number of sample points) in between them do (this is checked by whether or not their indices differ by 1). So only these points had to be considered when counting the pulses.

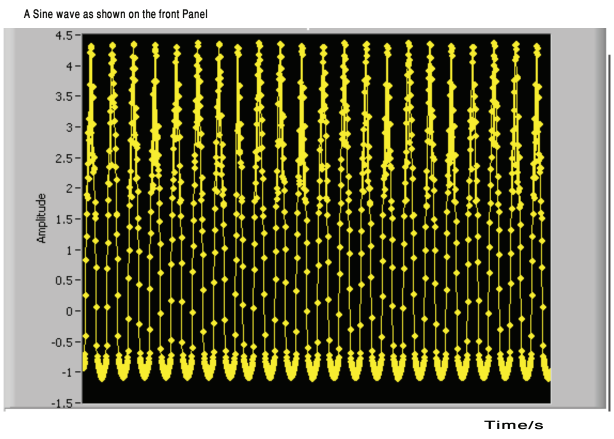

Figure 2: A sine wave generated by the signal generator.

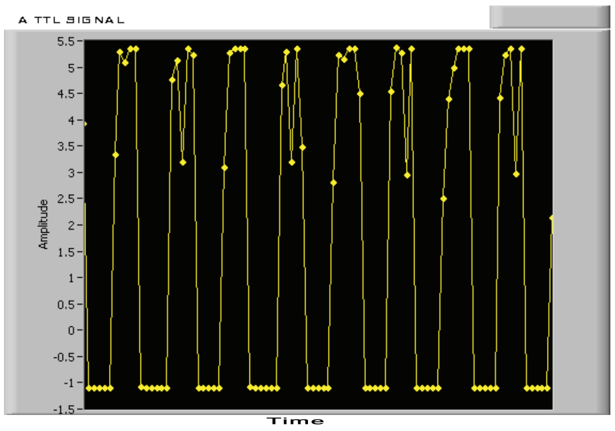

Figure 3: A typical TTL pulse from the signal generator as displayed on the Front Panel

This program still had one major shortcoming - it recorded the average frequency: dividing the cumulative *sum* of frequencies with the *total time elapsed.* This was not what we required when we wanted to measure counts of a *random process* where the frequency fluctuates. So I had to develop another LabVIEWprogram which recorded the current frequency. The idea was simple - calculate the frequency of each loop iteration, by measuring time elapsed *per iteration*, instead of the total time as done previously.

Finally, one further improvement was needed. The programs developed so far all had one common error: the processing time (which became significant with having multiple checks, case structures and for loops that ran thousands of times because as many sample points were included in each channel) was included in the frequency calculations, so our resulting frequency was always an underestimation. Also, the calculated frequency was observed to depend on the sample rate chosen. To correct these, the instructors explained to me the relationship between *sample rate* and *samples per channel.* So, instead of using the elapsed time as measured by the LabVIEWtiming devices, we could in fact find out the ratio between them, which gives us the time taken to collect the samples alone, and divide it by the number of pulses. The relation is:

$$r = \frac{n}{t} \tag{1}$$

where r = sampling rate; n = samples per channel; and t = time per channel.

3 Writing a program for the Radiation Experiment.

Now, we were ready to use LabVIEWto write the programs that will be used by the students in this experiment.

3.1 The setup

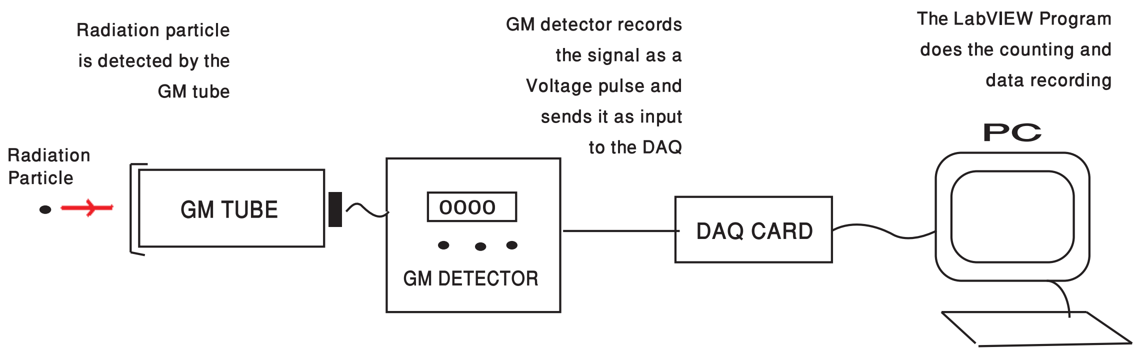
Radiation particle is detected by the GM tube
GM detector records the signal as a Voltage pulse and sends it as input to the DAQ
The LabVIEW Program does the counting and data recording
PC
Radiation Particle
GM TUBE
0000
GM DETECTOR
DAQ CARD

Figure 4: How background radiation data is collected and analyzed

We finally started work with the **Precision Geiger Detector**. This was connected with a **GM (Geiger Muller) Tube**, which detects any radiation particle that hits its surface.In our experiment, we want to do detailed statistical analysis of the data, as well as give our students an appreciation of data acquisition and the use of computer programs to analyze that data. We adjust the detector to set the voltage to 900 Volts. The reason for this is that the efficiency of the Geiger Counter depends on its input Voltage: below 900V it is less efficient, where as a higher value can damage the GM Tube. The Geiger Counter gives a *tick* sound whenever a radiation particle is detected by the GM tube.The **GM Tube** is a small cylindrical tube, with an anode at one end and a detecting surface on the other end. Whenever a radiation particle hits this surface, it causes an avalanche of electrons that are focused by the Cathode on the inside of the curved surface into a narrow beam, and accelerate towards the anode, where the y are detected.

The CRO displays the output of the Geiger Counter. Normally (and if properly grounded), it shows a simple straight line at the mean 0, indicating no pulse. At the *tick* sound, it shows a momentary 5V pulse, whose width indicates that it lasts for 0.8ms[1]. The time elapse of 0.8ms tells us the our equipment will not detect any other radiation particle during this time. So in a way this indicates the maximum count rate that can be detected.

The Radiation experiment involves a deep statistical analysis. First, we have to show that radioactive decay is a *random process*, and to prove this we have to show that our data agrees with a theoretical Poisson Distribution. Secondly, if the experiment is repeated a large number of times, of course randomness means that our values for the mean counts will not be the same, but the errors of these means should follow a Gaussian (Normal) Distribution (with a mean error of 0). The radiation sources were not available yet, so I started the work with background radiation.

Throughout this, just as it was emphasized while learning LabVIEW to keep practicing the use of Help Search, once we are on to the experiment, it was essential to keep searching online for whatever equipment or theory we were discussing, such as the SCC 68 Module, the Geiger Counter, the causes and nature of the background counts etc.

3.2 The Statistics and Matlab

For the Poisson Distribution, we need the data of just one experiment. First, we select a bin size (in seconds), typically of 1s or 0.5s. This will be specified by the user on the LabVIEWFront Panel. The program is named *Radiation and Statistics v.i* [2]. Then, we select the number of bins (at least 20 to show

[1]The value of $5V$ depends on the electronics of our experimental setup, and doesn't depend on the actual radiation itself.
[2]v.i means Visual Instrument - a LabVIEW program

a Poisson distribution). This is one set of data, and it will make one bar chart that will show us the number of counts in each bin. We will take several sets of data, say a total of 50 bar charts, and then sum them up, so our resulting bar chart will show us the sum of counts in each of the 20 bins. This constitutes one experiment. Our LabVIEWprogram would be such that a text file (named 'Radiation') will be saved that shows us the counts in all the bins as well as their (cumulative) sums. The following figure shows what a bar chart showing sum of counts:

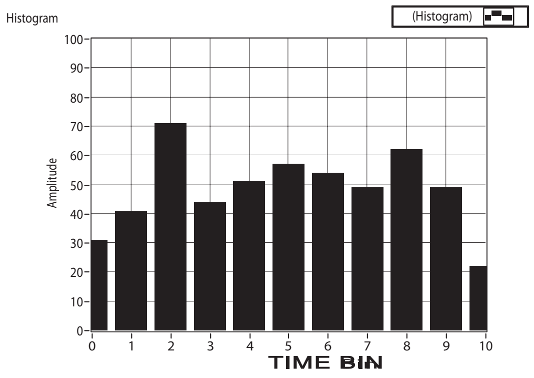

Figure 5: A typical bar chart showing sum of counts in each bin

Next, we copy-paste the array of sums in Matlab, and plot a histogram. By using the mean of this array, on the same figure, we plot the theoretical Poisson distribution of the data, and see how precisely our data fits a Poisson distribution. For a mean number of counts μ, the Poisson Distribution Function of a variable *n, F(n)* is given by:

$$F(n) = \frac{\mu^n e^{-\mu}}{n!} \tag{2}$$

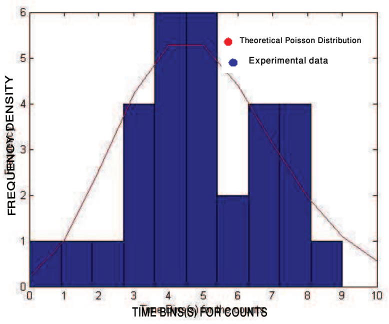

Figure 6: Fitting histogram on experimental data with a Poisson Distribution.

For the errors, we will need the data from about 50 experiments[3]. It is advisable to take a lesser number of bins or a smaller time interval in order to decrease the total time it takes to collect this data. A separate text file ('Radiation Means'), records the mean count per bin from each experiment. Again, we save the array of means in Matlab, and compute their average (and standard deviation). But now, we are dealing with the errors, so we compute the deviation from the average of each mean. then plot the histogram for this deviation, and on the same figure, plot the Gaussian curve, remembering to multiply the probabilities with the number of experiments. A typical fit is shown in Figure 7, the data collected for 30 experiments. The errors distribution comes closer and closer to the Gaussian Distribution as we increase the number of experiments. For N experiments, bin size Δ (in seconds), mean of errors μ, and

[3]The errors show a normal distribution when the number of experiments is large (i.e greater than or equal to 50).

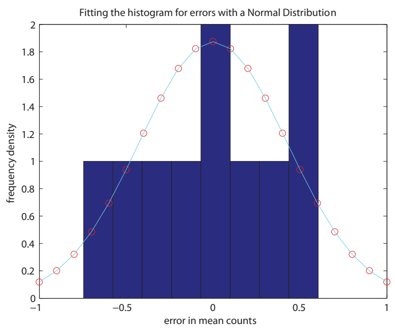

Figure 7: (On MatLab: Histogram of the counts data fitted with a theoretical Normal Distribution

standard deviation of the errors σ, the Normal Distribution Function of a variable *n, F(n)* is given by:

$$F(n) = \frac{N\Delta exp(\frac{-(n-\mu)^2}{2\sigma^2})}{2\sigma^2} \tag{3}$$

3.3 Preparing the LabVIEWprogram

Now that I knew all the requirements, the next task was to make a LabVIEWprogram, and then repeatedly test it for its short-comings. Here, unlike previously when we were measuring *cycles per second*, it should be noted that we are now measuring *counts* per bin. So, as background radiation consists of TTL pulses that have only one sample point in the tolerance region (refer to section 2.2), instead of a series of sample points in a crest region, we do not need to worry about whether or not the indices differ by 1. Instead, we simply take the number of the points that lie in the sample region, and that gives us the total number of pulses in that bin.

So all these subtleties would become clear to me one by one, with repeated observations and explanations from Sir Sohaib, because we started with the whole exercise using the signal generator. I have as yet only made frequency measurement programs that work on one particular type of wave, so I had to attempt to write a different one for each new kind of wave that we dealt with, but it has been some useful practice.

The program wrote text files that recorded the cumulative sums of counts in each bin, as well as the original data of each bin. These were displayed by waveform charts, graphs as well as numerical array indicators on the Front Panel. The next task was to make a histogram of this data, as well as a Poisson graph from the computed current mean. LabVIEW has encoded functions that can generate both current and cumulative histograms, and they were used in the program, but I was shown how to build one manually on LabVIEW.

The program now consists of a main LabVIEWv.i *Radiation and Statistics*, two text files where the experiment data gets recorded (*Radiation* and *Radiation Means* in the user's Z Drive, and two Matlab files (*Poisson Distribution* and *Normal Distribution*) for timely instructions to students about how to use simple Matlab commands such as *hist()*, *mean()* and *for loops* to do the statistical analysis.

4 Learning how to write a Latex document

The final job was to learn about writing Latex document using a nice editor called **WinEdt**, as well as the software **Adobe Illustrator** for diagrams. These will be required in writing Lab Manuals and Lab Development Reports, which need a lot of diagrams, equations and formatting. The power of Latex documentation is that the user can use the program to feed a number of instructions, such as automated numbering, making columns, organizing sections and subsections, and even creating new functions like question icons. We use WinEdt extensively for preparing Lab Manuals, as they are technical documents, requiring much formatting, and consisting of many diagrams and equations.

I was required to write a Latex Lab development report and began learning with the instructor Waqas Mahmood. I had a few problems in downloading and installing but soon learned the basics,

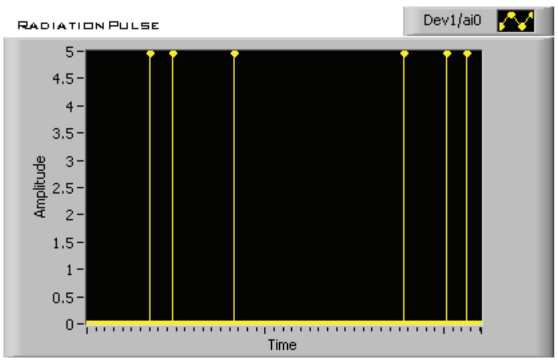

Figure 8: A background Radiation pulse as displayed on the Front Panel.

e.g how to begin, headings, sections and subsections etc. We required graphs and pictures from the LabVIEWprogram, and this needed Adobe Illustrator, with ghostscript. After some basic introduction to equations and figures in Latex, I was able to present this report to the reader.

5 Conclusion

Throughout this time the internees were instructed to keep a written record of everything notable in a notebook, so that another reader can easily understand and carry out the tasks that we did. There was also a meeting in which the internees were guided about safety instructions to be followed in the labs.

It has been a rich learning experience so far. I have struggled through, progressed slowly and been helped quite frequently, and have learned a lot of things about experimental physics, most importantly that their is a lot more to learn, and that by being stuck at problems is very often how we can learn the most. I am looking forward to the next tasks assigned, and also to continue such work in future.

The next assignment in Physics Lab would be to work on improving the Rotational Mechanics Experiment with Sir Waqas Mahmood.