PHYS2020 Introductory Lab Report

Samuel Allpass s4803050
School of Mathematics and Physics, University of Queensland, Brisbane, QLD 4072, Australia

I. INTRODUCTION

The purpose of this investigation is to complete DC circuit analysis in order to work out the internal resistance of a 9V power supply, whilst placing firm analysis on understanding the uncertainties involved in the system. Background The investigated apparatus consists of an extremely simple setup, with a 9V battery and clip attached to two wires. Along with this supply is provided a series of LEDs, which when different numbers of them are electrically connected in series, act as a variable load. For such a setup, the voltage and current output from the power supply follows the system:

$$V(I) = V_{open} - IR_{int} \quad (1)$$

Where V_{open} is the open circuit voltage of the battery and R_{int} is the internal resistance of the power supply.

II. TASK 1

In similar fashion to equation 1, it is understood that that the internal resistance can be calculated given the short circuit current and open circuit voltage is understood. This relationship is as follows:

$$V_{open} = I_{sc}R_{int}$$

Using a multimeter, it was uncovered that the supply had a voltage and current of 8.88V and 16.93mA respectively. And it was estimated that the power supply had an internal resistance of:

$$R_{int} = \frac{8.88}{16.93 * 10^{-3}} = 524\Omega$$

Task 2 Utilising the LED load system identified in the introduction, it was determined that a voltage-current relationship could be determined by continuously increasing the load of the supply.

A. methodology

In order to measure the voltage change as the load was increased it was well understood that the multimeter must be placed in parallel with the load, such that the circuit looked as follows in figure 1.

Plotting the data in excel (figure 2), and utilising a Java linear regression program I constructed in

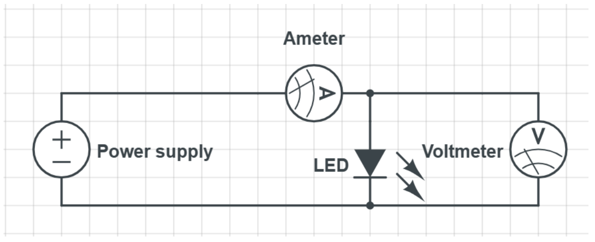

FIG. 1. Experimental setup to measure current and voltage over the LED load.

PHYS2941 (appendix 1), it was identified that the voltage and current were related by:

$$V(I) = (-520 \pm 90)I + (9.6 \pm 0.9)$$

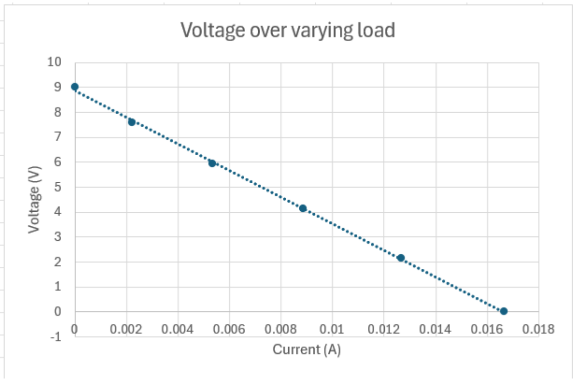

FIG. 2. Voltage and current plot resulting setup and methodology outlined.

B. Evaluation

From these results, multiple conclusions can be made. Firstly, the root mean squared of the current values outlined:

$$RMS_{current} = \sqrt{\frac{\sum x^2}{n}} = 0.0096A \quad and \quad RMS_{voltage} = 6.2V$$

The root mean squared gives a general sense of the size of measured values, such that it is fair to conclude that the internal resistance will be roughly:

$$RMS_{R_{int}} = \frac{6.2}{0.0096} = 645.8\Omega$$

However, given it is well established that the gradient of the regressions is the negative of the internal resistance, the regression suggests an internal resistance of 524.29V. This value proves far more accurate than the RMS value when compared to the value calculated in tast 1.

Translating from the data, it was understood that the open circuit voltage of the equation would be reflected in the point at which I = 0A. As observed, a clear discrepancy arises, with the linear regressions estimating a supply voltage of 9.6 ± 0.8V despite the measured value in task 1 of 8.88 placed in the data set. This provides a clear example of the cross-correlation coefficient, such that linear regression created a plot to best optimise the general trend, disregarding that the open circuit current is a set 8.88V. Further investigating this cross-correlation coefficient, it becomes important to recognise that coefficient can be calculated by:

$$< \Delta c \Delta m >= - < x > *(< \Delta m >)^2 = - < I > *94.58^2$$

Given the average current value of 0.009, the cross-correlation coefficient was determined to be 81.89. Given the positive cross-correlation coefficient, it is reasonable to conclude that the slope and intercept are closely linked, almost entirely a result of the plot being made to satisfy both simultaneously.

III. TASK 3

Given the importance of understanding the maximum current or power a power supply can provide, it becomes reasonably to calculate such a value using the discoveries in task 2. Firstly, it was recognised that:

$$I_{sc} = \frac{V_{open}}{R_{int}}$$

And so following an uncertainty, accounting for the cross-correlation coefficient of:

$$\frac{\langle \delta I_{\text{sc}}^2 \rangle}{I_{\text{sc}}^2} = \frac{\langle \delta V_{\text{open}}^2 \rangle}{V_{\text{open}}^2} + \frac{\langle \delta R_{\text{int}}^2 \rangle}{R_{\text{int}}^2} - 2 \cdot \frac{\langle \delta V_{\text{open}} \, \delta R_{\text{int}} \rangle}{V_{\text{open}} R_{\text{int}}}$$

Given the values identified previously, $V_{open} = 9.6V$ and $R_{int} = 524.29\Omega$ we find $I_{sc} = 0.018A$. Calculating the corresponding uncertainty with $\delta V_{open} = 0.87$, $\delta R_{int} = 94.58\Omega$ and a correlation coefficient of 81.89, it was found that $\delta I_{sc} = 0.002$ and thus we are left with the general solution:

$$I_{sc} = 0.018 \pm 0.002A$$

This solution was found to lie extremely close to the 16.93mA short circuit current measured in task 1. Expanding on this discovery, it was recognised the the maximum power could then be calculated utilising the general formula P = IV such that:

$$P(I) = V_{open} * I - R_{int} * I^2 = (-524.29 \pm 94.58) I^2 + (9.60 \pm 0.87) I$$

In order to find the max power, it is understood that:

$$\frac{dP}{dI} = 2 * (-524.29 \pm 94.58) I + (9.60 \pm 0.87) = 0$$

Note that the uncertainty in I is given by:

$$\delta I = I \sqrt{(\frac{\delta R_{int}}{R_{int}})^2 + (\frac{\delta V_{open}}{V_{open}})^2}$$

Solving this it was found that the supply current that achieved the maximum power output over the load was $0.009 \pm 0.002A$. From this discovery, it was clear that substituting I_{max} back into P(I) would demonstrate the maximum power the supply could output, a value found to be 0.0440W. Finally, realising that the maximum power output follows the equation:

$$I_{max} = \frac{V_{open}}{2R_{int}} \quad and\, further \quad P_{max} = \frac{V_{open}^2}{4R_{int}}$$

We obtain an uncertainty equation for the maximum power again utilising the cross-correlation coefficient of:

$$\left(\frac{\delta P}{P}\right)^2 = \left(\frac{2\delta V_{\text{open}}}{V_{\text{open}}}\right)^2 + \left(\frac{\delta R_{\text{int}}}{R_{\text{int}}}\right)^2 - \frac{2 \cdot \langle \delta V_{\text{open}} \, \delta R_{\text{int}} \rangle}{V_{\text{open}}^2 R_{\text{int}}}$$

Substituting in the values previously stated, it was found that the maximum power output of the 9V battery was $0.04 \pm 0.01W$.

IV. CONCLUSION

The purpose of this investigation was to complete DC analysis of a simple 9V circuit with a load in order to not only determine the short circuit current, open circuit voltage, and maximum current and power output, but to also demonstrate the importance of correctly handling uncertainty. Task 1 clearly identified how uncertainty techniques such as root mean squared, averages and cross-correlation coefficients play a major role in the values calculated from experimental results. After correctly identifying how to evaluate the uncertainties in the extremely correlated voltage and current measurements using the cross-correlation coefficient, values for the maximum current and power output from the 9V supply were calculated.

V. APPENDIX

Java linear regression code:

// Script to calculate the Linear regression plot // Written by Samuel Allpass // Version 1 // 4th August 2024

import java.util.Arrays; import java.util.List; import java.util.Scanner; import java.util.stream.Collectors; import java.io.FileWriter; import java.io.IOException; import java.io.PrintWriter;

public class Linear private List<Float> x; private List<Float> y; private List<Float> uncerty; private boolean isWeighted; private Float m; private Float c; private static String xname; private static String yname;

public Linear(List<Float> x, List<Float> y, List<Float> uncerty, boolean isWeighted) this.x = x; this.y = y; this.uncerty = uncerty; this.isWeighted = isWeighted;

public static void main(String[] args) Scanner scanner = new Scanner(System.in);

try System.out.println("What is the name of the x data:"); xname = scanner.nextLine(); System.out.println("What is the name of the y data:"); yname = scanner.nextLine();

System.out.println("Enter the first (x) data set:"); List<Float> x = Arrays.stream(scanner.nextLine().split(",")) .map(Float::parseFloat) .collect(Collectors.toList());

System.out.println("Enter the second (y) data set:"); List<Float> y = Arrays.stream(scanner.nextLine().split(",")) .map(Float::parseFloat) .collect(Collectors.toList());

System.out.println("Enter the uncertainty of the second (y) data set:"); List<Float> uncerty = Arrays.stream(scanner.nextLine().split(",")) .map(Float::parseFloat) .collect(Collectors.toList());

System.out.println("Is the regression Weighted (true or false):"); boolean isWeighted = Boolean.parseBoolean(scanner.nextLine());

scanner.close();

Linear linear = new Linear(x, y, uncerty, isWeighted); linear.performRegression(); catch (Exception e) System.err.println("Invalid input. Please enter valid numbers.");

private void performRegression() float deltam = 0, deltac = 0;

if (isWeighted) m = weightedGradientCalc(); deltam = weightedGradientUncertCalc(); c = weightedYintCalc(); deltac = weightedYintUncertCalc(); else m = gradientCalc(); deltam = gradientUncertCalc(); c = yintCalc(); deltac = yintUncertCalc();

System.out.printf("y = (saveDataToDat();

private void saveDataToDat() try (PrintWriter writer = new PrintWriter(new FileWriter("src/data.dat"))) for (int i = 0; i < x.size(); i++) writer.printf(" writer.printf("writer.printf("writer.printf("writer.printf("

catch (IOException e) e.printStackTrace();

private float weightedMeanCalc(List<Float> f, List<Float> uncert) float numerator = 0f; float denominator = 0f; for (int i = 0; i < f.size(); i++) float weight = 1 / (uncert.get(i) * uncert.get(i)); numerator += f.get(i) * weight; denominator += weight; return numerator / denominator;

private float weightedGradientCalc() float sum = 0f; float meanX = weightedMeanCalc(x, uncerty); for (int i = 0; i < x.size(); i++) float weight = 1 / (uncerty.get(i) * uncerty.get(i)); sum += weight * (x.get(i) - meanX) * y.get(i); return sum / weightedDcalc();

private float weightedDcalc() float sum = 0f; float meanX = weightedMeanCalc(x, uncerty); for (int i = 0; i < x.size(); i++) float weight = 1 / (uncerty.get(i) * uncerty.get(i)); sum += weight * (x.get(i) - meanX) * (x.get(i) - meanX); return sum;

private float weightedYintCalc() return weightedMeanCalc(y, uncerty) - weightedGradientCalc() * weightedMeanCalc(x, uncerty);

private float weightedGradientUncertCalc() float sum = 0f; float gradient = weightedGradientCalc(); float intercept = weightedYintCalc();

for (int i = 0; i < x.size(); i++) float weight = 1 / (uncerty.get(i) * uncerty.get(i)); float residual = y.get(i) - gradient * x.get(i) - intercept; sum += weight * residual * residual;

float deltamsq = (1 / weightedDcalc()) * (sum / (x.size() - 2)); return (float) Math.sqrt(deltamsq);

private float weightedYintUncertCalc() float sum = 0f; float sumw = 0f; float gradient = weightedGradientCalc(); float intercept = weightedYintCalc(); float meanX = weightedMeanCalc(x, uncerty);

for (int i = 0; i < x.size(); i++) sumw += 1 / (uncerty.get(i) * uncerty.get(i));

for (int i = 0; i < x.size(); i++) float weight = 1 / (uncerty.get(i) * uncerty.get(i)); float residual = y.get(i) - gradient * x.get(i) - intercept; sum += weight * residual * residual;

float deltacsq = ((1 / sumw) + ((meanX * meanX) / weightedDcalc())) * (sum / (x.size() - 2)); return (float) Math.sqrt(deltacsq);

```
private float meanCalc(List<Float> f) float sum = 0f; for (Float value : f) sum += value; return sum / f.size();
private float gradientCalc() float sum = 0f; float meanX = meanCalc(x); for (int i = 0; i < x.size(); i++) sum += (x.get(i) - meanX) * y.get(i); return sum / dcalc();
private float dcalc() float sum = 0f; float meanX = meanCalc(x); for (int i = 0; i < x.size(); i++) sum += (x.get(i) - meanX) * (x.get(i) - meanX); return sum;
private float yintCalc() return meanCalc(y) - gradientCalc() * meanCalc(x);
private float gradientUncertCalc() float sum = 0f; float gradient = gradientCalc(); float intercept = yintCalc();
for (int i = 0; i < x.size(); i++) float residual = y.get(i) - gradient * x.get(i) - intercept; sum += residual * residual;
float deltamsq = (1 / dcalc()) * (sum / (x.size() - 2)); return (float) Math.sqrt(deltamsq);
private float yintUncertCalc() float sum = 0f; float gradient = gradientCalc(); float intercept = yintCalc(); float meanX = meanCalc(x);
for (int i = 0; i < x.size(); i++) float residual = y.get(i) - gradient * x.get(i) - intercept; sum += residual * residual;
float deltacsq = ((1 / x.size()) + ((meanX * meanX) / dcalc())) * (sum / (x.size() - 2)); return (float) Math.sqrt(deltacsq);
```