Instruments and Wiring

The Output Resistance of a Power Supply

PHY224 Lab 5

Fredrik Dahl Bråten, Pankaj Patil

October 27, 2021

1 Abstract

In this exercise, we measure and model the current and voltage of two circuit diagrams, first with a battery, and then later with a power supply. As the only difference between the diagrams is the placement of the ammeter and the voltmeter, we can with these two setups determine the resistance of the ammeter, the current through the voltmeter, and the internal resistance of the battery. All these three parameters are usually assumed to be negligible. We will use a linear function to model the relation between the total voltage and the total current through the circuit. To evaluate the quality of our models, we calculate and discuss the χ^2_{red} value of our model and data. This analysis was done in Python by use of the numpy, scipy and matplotlib modules.

2 Introduction

In this exercise, we measure and model the current and voltage of two circuit diagrams, first with a battery and then later with a power supply, see OutputResist.pdf. The theoretically predicted relationship between the terminal voltage (V_1), the open circuit voltage (V_∞), the current passing between the nodes of the battery/power supply (I_2) and the output resistance of the power source ($R_{internal}$), is:

$$V_1 = V_\infty – R_{internal} I_2$$

This relationship is modeled by a linear relationship $y = ax + b$. Furthermore, we know that the terminal voltage $V_{PS} = V_1$, the voltage measured in the circuit configuration 1. The voltage difference over the resistor $V_R = V_2$, the voltage measured in the circuit configuration 2. Also, the potential difference over the ammeter $V_A = V_{PS} – V_R$. By this notation, we also have the following relations:

$I_{PS} = I_2, I_R = I_1, I_V = I_{PS} – I_R$, and thus

$I_V = I_2 - I_1$.

Furthermore, by Ohms law, we have $R_A = V_A/I_A = V_{A1}/I_{A1} = (V_{PS}–V_R)/I_1 = (V_1–V_2)/I_1$.

Thus, we have the relations for the current through the voltmeter, $I_V = I_2–I_1$, and the relation for what resistance the ammeter has: $R_A = (V_1–V_2)/I_1$.

In our experiment, V_1 and V_2 are our dependent variables, while $R_{internal}$, R_A, I_1, and I_2, are independent variables.

3 Methods, Materials and Experimental Procedure

We successfully followed the procedures as described in the OutputResist.pdf document.

4 Results

From our model of the total voltage and current, for the 6V battery, we found that:

$V_\infty = 6.31 \pm 0.1\ V$

$R_{internal} = 0.77 \pm 2.209\ Ohm$ (using the standard deviation of the power supply internal resistance estimates as uncertainty).

$\chi^2_{red} = 0.00020972$

Similarly, for our power supply with the 6V setting:

$V_\infty = 6.00 \pm 0.1\ V$

$R_{internal} = 48.37 \pm 2.209\ Ohm$

$\chi^2_{red} = 0.00006698$

For the power supply with the 10V setting:

$V_\infty = 10.00 \pm 0.1\ V$

$R_{internal} = 51.73 \pm 2.209\ Ohm$

$\chi^2_{red} = 0.00000218$

For the power supply with the 15V setting:

$V_\infty = 15.00 \pm 0.1\ V$

$R_{internal} = 45.84 \pm 2.209\ Ohm$

$\chi^2_{red} = 0.00002729$

For the power supply with the 20V setting:

$V_\infty = 20.00 \pm 0.1\ V$

$R_{internal} = 46.99 \pm 2.209\ Ohm$

$\chi^2_{red} = 0.00000242$

Therefore, the average internal resistance of the power supply with uncertainty, estimated from these measurements are: $R_{internal\ (power\ supply)} = 48.23 \pm 1.1045\ Ohm$ (uncertainty $= 2.209/\sqrt{N}$). We will associate a similar uncertainty to our estimate of the battery's internal resistance, as that experiment were performed similarly to those with the power supply: $R_{internal\ (battery)} = 0.77 \pm 2.2\ Ohm$. This will be our estimate of the internal resistance of the power supply and battery.

Furthermore, averaging and taking the standard deviation of each ammeter resistance estimate from the 6 resistors for the 4 different voltage settings of the power supply, we find that our estimate of the resistance of the ammeter is: 16.9 ± 7.18 Ohm The voltage passing through the voltmeter varies for each voltage setting, but for instance, the current through the voltmeter is: -0.00012 Ampere when the total current through the circuit is 0.0059 Ampere, the terminal voltage is: 6 volts, and the resistor in the circuit is 100.0 kilo Ohm. Run the python script in the appendix to get all the different currents through the voltmeter for each resistor and power supply setting.

Below in Figure [1]-[5], we see our data from the experiments plotted as points with corresponding error bars. Furthermore, we see the curve of our model fitted to the data, as described in the introduction.

5 Discussion

The χ^2_{red} values we found were very low. This means that our models fit our data very well. However, our χ^2_{red} values should ideally be equal to one for most experiments. That we have extremely low χ^2_{red} values implies that we do not have enough data. It means that we are in risk of overfitting our models to our data. However, this exercise asked for data from four resistors, while we used six. Thus, in our case, the small χ^2_{red} values is a good sign, as it means that we have good linear data for which our model fits nicely. Furthermore, both models are well within the uncertainties of our measurements, see Figure [1] to [5] in the Appendix. As we can see from the current passing through the voltmeter, it is very small in comparison to the measured total current in the circuit, see the results section. This is a good sign, as ideally no current should pass through the voltmeter. The current is also negative, which is not a surprising result. Within the voltmeter there is a battery made for compensating for the drop in voltage caused by the presence of the voltmeter in the circuit. If this battery overcompensates for the voltage drop, current will flow in the opposite direction of normal flow (according to the power supply). This is what we observe. Furthermore, the estimated resistance of the ammeter is small in comparison to the resistance of the different resistors for which we used in our circuits. Whether the resistance is negligible depends on the resistance of the resistor in the circuit, which in our experiments varied in size from 100Ohm to 100k Ohm. The estimated internal resistance of the power supply was large and significant in comparison to the smaller resistors we used, though negligible for the larger resistors we used. We got satisfactory small uncertainties in our estimate of the internal resistance for the power supply. However, for the

battery, the estimated internal resistance was indeed very small and negligible in comparison to all the resistors in our experiments. Now for answering the specific questions in the lab manual:

Question 1

There is no better circuit diagram of the two. In the first one, the voltmeter measures the total potential difference in the circuit over the battery, while the ammeter measures the current flowing through the resistor. In the second diagram, the voltmeter measures the voltage difference over the resistor, while the ammeter measures the total current flowing in the circuit, from one node to the other of the battery. Together, measurements of current and voltage for each of the diagrams, determine what current run through the resistor, what current run through the voltmeter, and what current runs from the cathode to the anode of the battery. Likewise, we are able to determine the potential difference over the battery, over the resistor, and over the ammeter. With ideal multimeters, there should however be no difference between the two.

Question 2

We chose to measure current and voltage for the two diagrams with as many different resistors as possible, of those we had easily available, to increase our χ^2_{red} value for our model. Thus, we used all six resistors on the circuit board. Furthermore, statistically by the central limit theorem, a higher number of measurements results in a lower uncertainty in our estimates for the internal resistances of the battery, power supply and ammeter along with the current through the voltmeter, when averaging.

Question 3

See introduction. Furthermore, though I_V varies for each resistor and power supply voltage, by averaging the estimated R_A for the data associated with each resistor and power supply/battery voltage, we can get a better estimate of R_A, see the results section.

As we mentioned previously in the discussion, the resistance of the ammeter is small and negative. This is due to its large resistance and the overcompensation for voltage by the small internal battery of the voltmeter.

Question 4

See plots in the Appendix to see the estimated relationship between the current and voltage of the circuit according to our data. For the range of resistors for which we utilized in our experiments with the power supply, the current and voltage followed a linear relationship. However, according to the theory, for higher currents, we should see a non-linear relationship. However, we never reached currents which formed non-linear relationships with the voltage.

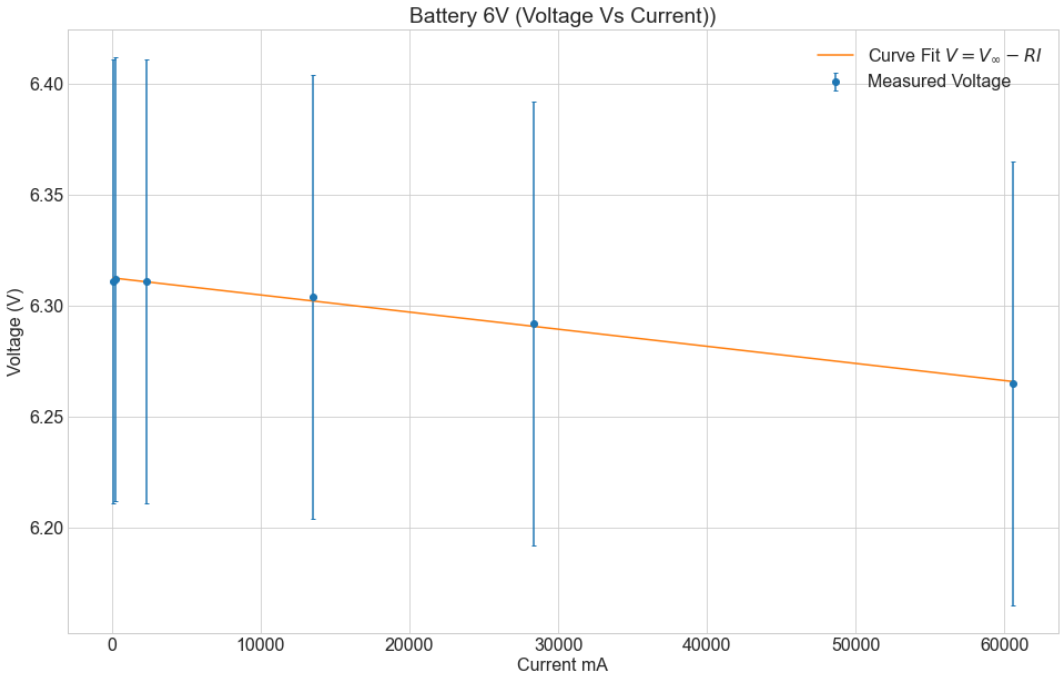
However, the maximum current is the current between the terminals of the power supply when V_∞ is approaching zero. We can estimate this value from our linear relations, though this will be a poor fit with the theoretically predicted maximum current, if the voltage current-relationship is non-linear at higher currents: At I_{max}, V=0, thus according to the equation from the introduction, $V = V_\infty - R_{internal} I$ So, $I_{max} = V_\infty / R_{internal}$. We then get an increasing relationship between V_∞ and I_{max}, see Figure [6] in the Appendix.

6 Conclusions

In this exercise, we measure and model the current and voltage of two circuit diagrams, first with a battery, and then later with a power supply. We successfully followed the instructions for the experiments written in the OutputResist.pdf lab manual. We have plotted our data with error bars, along with our corresponding linear models. Furthermore, we have calculated and discussed the χ^2_{red} values of the models, and estimated the internal resistance of the power supply, the battery, the ammeter, and the current flowing through the voltmeter. Finally, we have answered the questions of the lab manual.

A Appendix

A.1 Plots For 6V Battery



Model Function : $V = V_\infty - RI$
$V_\infty = 6.31$ V
$R = 0.77$ Ohm

Figure 1: 6V Battery: Voltage vs. Current

A.2 Plots For Power Supply

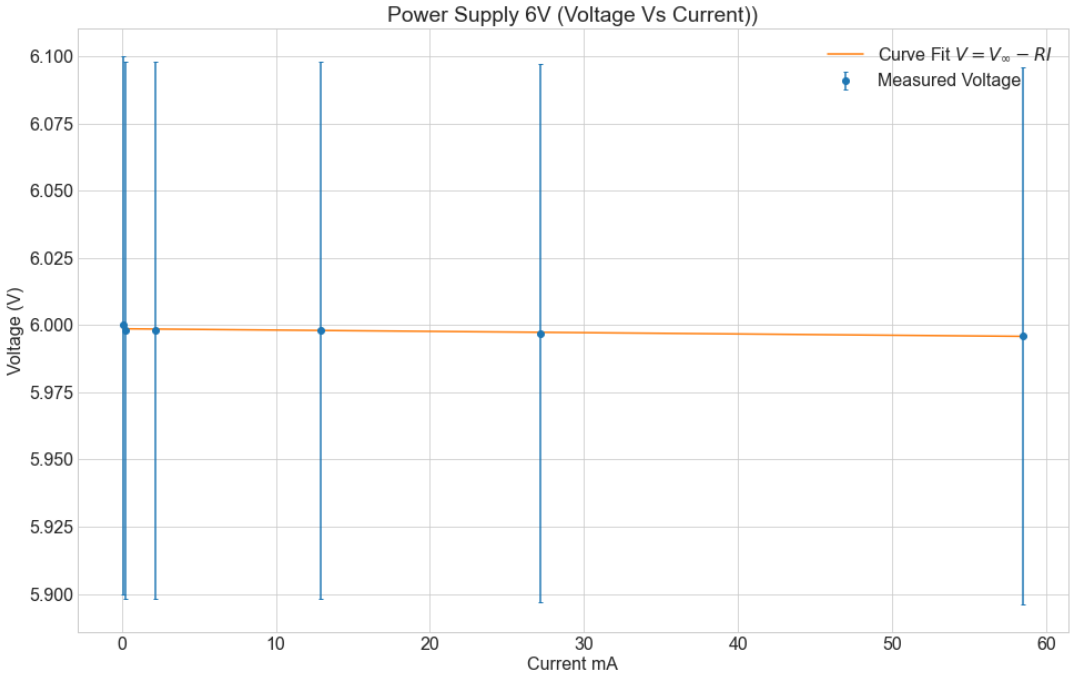


Figure 2: Power Supply 6V (Voltage vs Current)

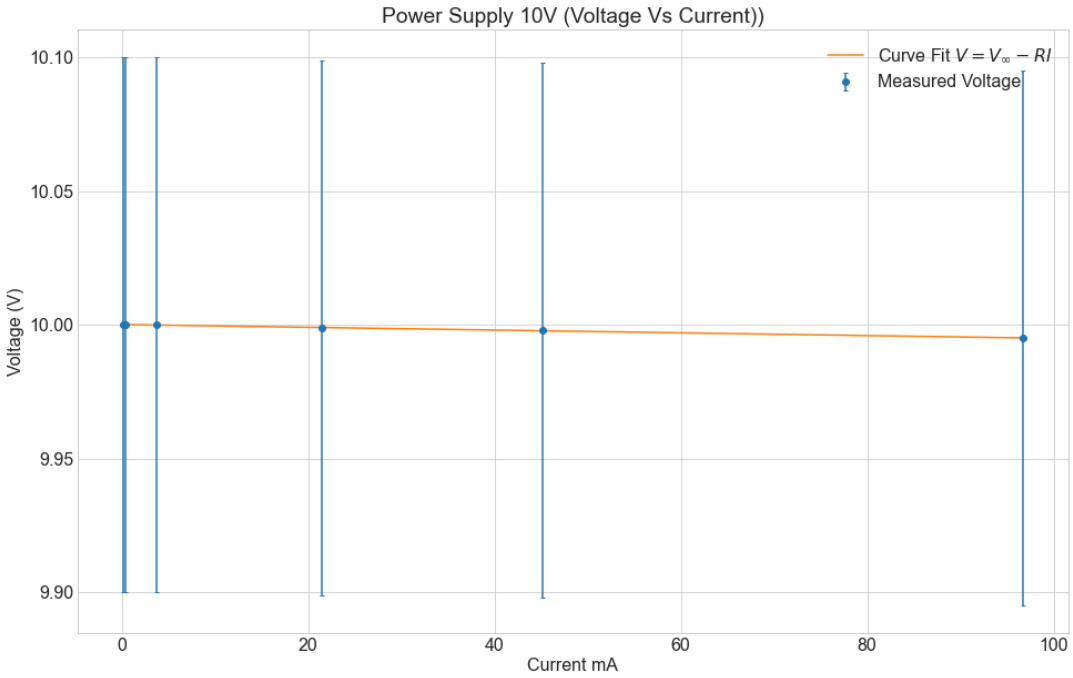


Figure 3: Power Supply 10V (Voltage vs Current)

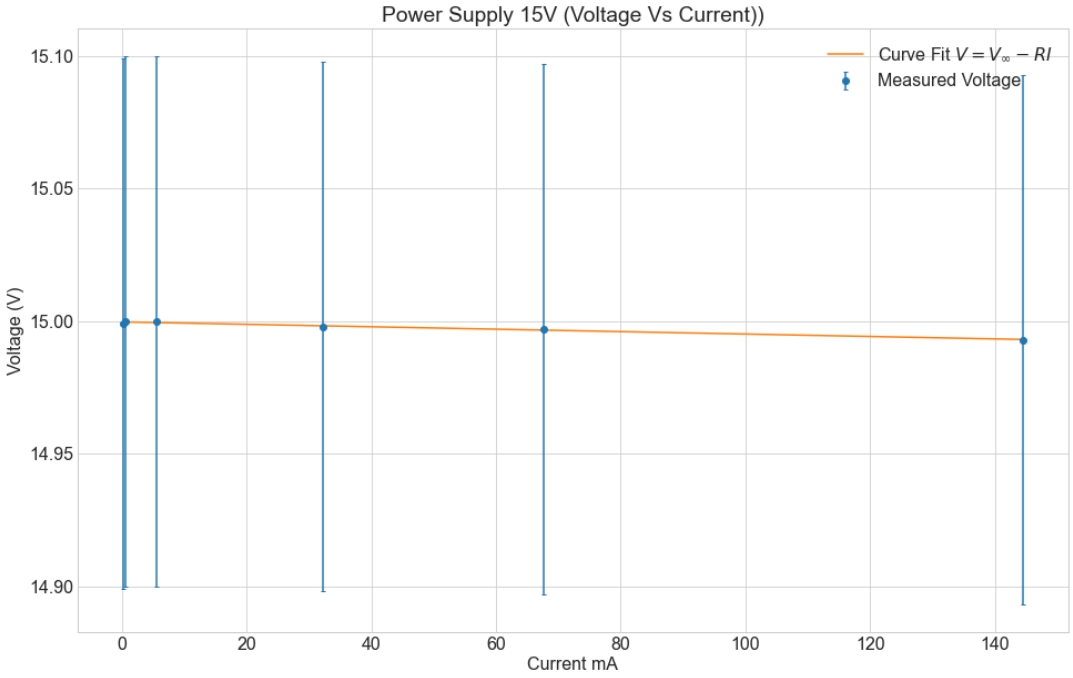


Figure 4: Power Supply 15V (Voltage vs Current)

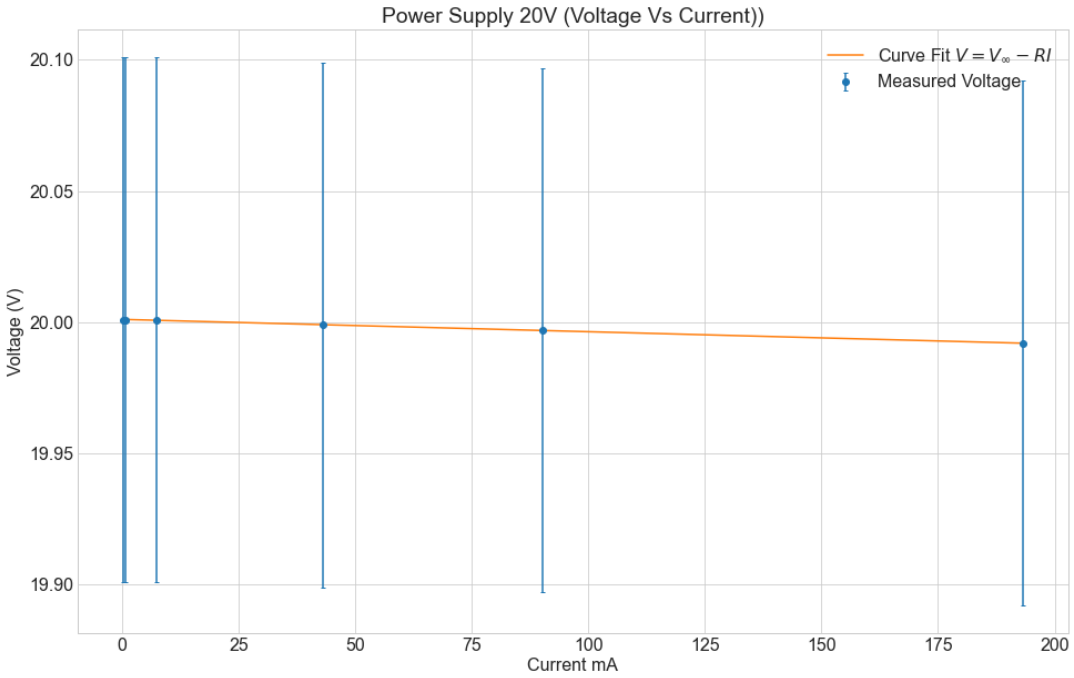


Figure 5: Power Supply 20V (Voltage vs Current)

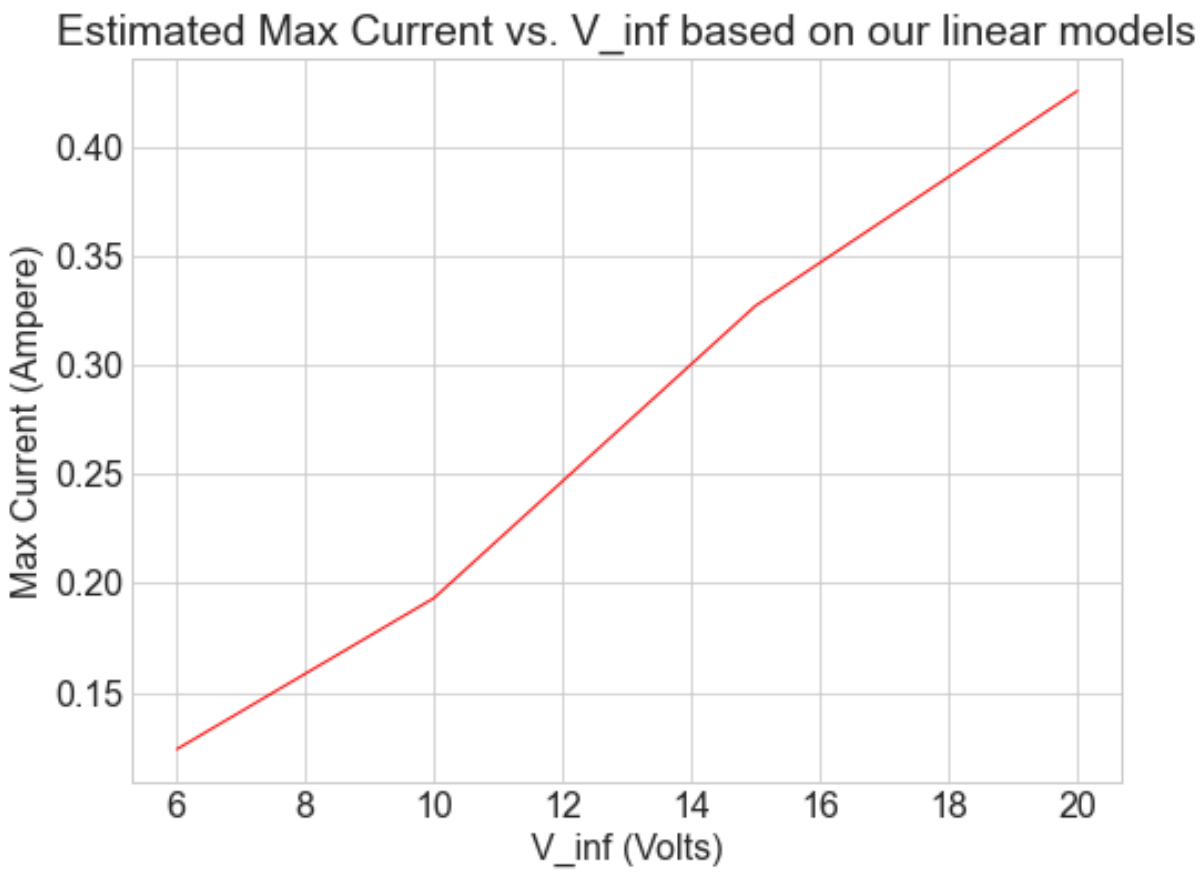


Figure 6: 6V Battery: Maximum Current vs. V_∞

A.3 Python Code: Battery 6V

The Python code for this exercise is divided into two files. statslab.py file contains utility methods which we will be frequently using in this course. lab_5_battery.py file contains the code which analyzes the data.

A.3.1 statslab.py

```
import numpy as np
import scipy.optimize as optim
import matplotlib.pyplot as plt


##############################################################################
# Utility Methods Library
#
# This file contains some utility method which are common to our data analysis.
# This library also contains customized plotting methods.
##############################################################################

# use bigger font size for plots
plt.rcParams.update({'font.size': 16})

def chi2(y_measure,y_predict,errors):
    """Calculate the chi squared value given a measurement with errors and
    prediction"""
    return np.sum( np.power(y_measure - y_predict, 2) / np.power(errors, 2) )

def chi2reduced(y_measure, y_predict, errors, number_of_parameters):
    """Calculate the reduced chi squared value given a measurement with errors
    and prediction, and knowing the number of parameters in the model."""
    return chi2(y_measure, y_predict, errors)/ \
            (y_measure.size - number_of_parameters)

def read_data(filename, skiprows=1, usecols=(0,1), delimiter=","):
    """Load give\n file as csv with given parameters,
    returns the unpacked values"""
    return np.loadtxt(filename,
                      skiprows=skiprows,
                      usecols=usecols,
                      delimiter=delimiter,
                      unpack=True)
```

```
def fit_data(model_func, xdata, ydata, yerrors, guess=None):
    """Utility function to call curve_fit given x and y data with errors"""
    popt, pcov = optim.curve_fit(model_func,
                                 xdata,
                                 ydata,
                                 absolute_sigma=True,
                                 sigma=yerrors,
                                 p0=guess)

    pstd = np.sqrt(np.diag(pcov))
    return popt, pstd

# y = ax+b
def linear_regression(xdata, ydata):
    """Simple linear regression model"""
    x_bar = np.average(xdata)
    y_bar = np.average(ydata)
    a_hat = np.sum( (xdata - x_bar) * (ydata - y_bar) ) / \
            np.sum( np.power((xdata - x_bar), 2) )
    b_hat = y_bar - a_hat  * x_bar
    return a_hat, b_hat

class plot_details:
    """Utility class to store information about plots"""
    def __init__(self, title):
        self.title = title
        self.x_log_scale = False
        self.y_log_scale = False

    def errorbar_legend(self, v):
        self.errorbar_legend = v
    def fitted_curve_legend(self, v):
        self.fitted_curve_legend = v
    def x_axis_label(self, v):
        self.x_axis_label = v
    def y_axis_label(self, v):
        self.y_axis_label = v
    def xdata(self, x):
        self.xdata = x
    def ydata(self, y):
        self.ydata = y
    def yerrors(self, y):
        self.yerrors = y
    def xdata_for_prediction(self, x):
        self.xdata_for_prediction = x
    def ydata_predicted(self, y):
        self.ydata_predicted = y
```

```
    def legend_position(self, p):
        self.legend_loc = p
    def chi2_reduced(self, c):
        self.chi2_reduced = c
    def set_x_log_scale(self, c):
        self.x_log_scale = c
    def set_y_log_scale(self, c):
        self.y_log_scale = c

def plot(plot_details, new_figure=True, error_plot=True):
    """Utility method to plot errorbar and line chart together,
    with given arguments"""
    if new_figure:
        fig  = plt.figure(figsize=(16, 10))
        fig.tight_layout()
        plt.style.use("seaborn-whitegrid")

    # plot the error bar chart
    if error_plot:
        plt.errorbar(plot_details.xdata,
                     plot_details.ydata,
                     yerr=plot_details.yerrors,
                     marker="o",
                     label=plot_details.errorbar_legend,
                     capsize=2,
                     ls="")

    # plot the fitted curve
    plt.plot(plot_details.xdata_for_prediction,
             plot_details.ydata_predicted,
             label=plot_details.fitted_curve_legend)

    # legend and title
    plt.title(plot_details.title)
    plt.xlabel(plot_details.x_axis_label)
    plt.ylabel(plot_details.y_axis_label)

    if plot_details.x_log_scale:
        plt.xscale("log")

    if plot_details.y_log_scale:
        plt.yscale("log")

    legend_pos = "upper left"
    if hasattr(plot_details, "legend_loc"):
        legend_pos = plot_details.legend_loc
```

```
    plt.legend(loc=legend_pos)
```

A.3.2 lab_5_battery.py

```
#!/usr/bin/env python3
# -*- coding: utf-8 -*-
"""
@author: Pankaj Patil
"""

import statslab as utils
import matplotlib.pyplot as plt
import numpy as np


voltage_uncertainity = .1

def model_function(currrent, v_infty, resistance):
    return  v_infty - resistance * currrent

# import the first setup for Voltage data
setup_1_file = "../../data/setup1-5V.csv"
measured_voltages = utils.read_data(setup_1_file,
                                                usecols=(2),
                                                skiprows=1)

# import the first setup for Current data
setup_2_file = "../../data/setup2-5V.csv"
measured_currents = utils.read_data(setup_2_file,
                                                usecols=(1),
                                                skiprows=1)

voltage_errors = np.ones_like(measured_voltages) * voltage_uncertainity

popt, pstd = utils.fit_data(model_function,
                            measured_currents,
                            measured_voltages,
                            voltage_errors)

print("V_infty = %.2f V" %  popt[0])
print("R_internal = %.2f ohm" %  (popt[1] * np.power(10, 6)))

currrent_data_for_prediction = np.linspace(measured_currents[0],
                                           measured_currents[-1], 1000)

predicted_voltages = model_function(currrent_data_for_prediction,
                                           popt[0],
```

```
                                        popt[1])

chi2r_curve_fit = utils.chi2reduced(measured_voltages,
                                model_function(measured_currents,
                                                popt[0],
                                                popt[1]),
                                voltage_errors,
                                2)

plot_data = utils.plot_details("Battery 6V (Voltage Vs Current))")
plot_data.errorbar_legend("Measured Voltage")
plot_data.fitted_curve_legend("Curve Fit $V = V_{\infty} - RI$")
plot_data.x_axis_label("Current mA")
plot_data.y_axis_label("Voltage (V)")
plot_data.xdata(measured_currents)
plot_data.ydata(measured_voltages)
plot_data.yerrors(voltage_errors)
plot_data.xdata_for_prediction(currrent_data_for_prediction)
plot_data.ydata_predicted(predicted_voltages)
plot_data.legend_position("upper right")
plot_data.chi2_reduced(chi2r_curve_fit)

# plot the data
utils.plot(plot_data)

plt.savefig("lab5_voltage_vs_current.png")
print("chi2reduced = %.8f" % chi2r_curve_fit)
```

A.4 Python Code: Power Supply

The Python code for this exercise is divided into four files. statslab.py in included in previous section. Code 1.py and Code 2.py analyze the data for Power Supply

A.4.1 Functions.py

```
# -*- coding: utf-8 -*-
"""
Created on Thu Sep 30 09:46:36 2021

@author: Fredrik
"""
import numpy as np
import scipy.optimize as optim
import matplotlib.pyplot as plt

#Defining the function for curve fitting and plotting
def curve_fit_and_plot(model,initial_guess,xdata,ydata,y_uncer,xunit,yunit,
                       plot_title):
    """
    This function uses the scipy curve_fit function to estimate the parameters
    of the model which will minimize the euclidian distance between our data
    points, and the model curve.
    We print these optimal model parameters along with their uncertainty, and
    plot the original data with error bars, along with the curve fit model.

    Parameters
    ----------
    model : function to be used as model
        model(x,a,b,c,...), where we are estimating a,b,c, etc.
    initial_guess : list of guesses for the parameters a,b,c, etc. eg. [2,4,254]
    xdata : list of input points for the model, eg. [2,4,5,7,9,28]
    ydata : list of output points for the model, eg [23,25,26,85,95,104]
    y_uncer : list of uncertaintes associated with the ydata, which the model
        shall output
    xunit : String describing the unit along the x-axis for label when plotting.
        eg. 'Voltage (V)'
    yunit : String describing the unit along the y-axis for label when plotting.
        eg. 'Current (A)'
    plot_title : String describing the title of the plot.
        eg. 'Current vs. Voltage'
```

```
    Returns None
    """
    #Using the scipy curve fit function to find our model parameters
    p_opt , p_cov = optim.curve_fit(model , xdata , ydata, p0 = initial_guess,
                                    sigma = y_uncer, absolute_sigma = True )
    p_std = np.sqrt( np.diag ( p_cov ))

    print("The optimal values for our curve fit model parameters, are:",np.round(p_opt,2))
    print("Their associated uncertainties  are:", np.round(p_std,2))

    #Now we create some data points on the model curve for plotting
    xvalues_for_plot = np.linspace(xdata[0],xdata[-1],1000)
    yvalues_for_plot = []
    for i in xvalues_for_plot:
        yvalues_for_plot.append(model(i,p_opt[0],p_opt[1]))

    #Now we plot the original data with error bars, along with the curve fit model
    plt.figure(figsize=(10,5))
    plt.errorbar(xdata,ydata,y_uncer,c='r', ls='', marker='o',lw=1,capsize=2,
                 label = 'Points of measurement with uncertainty')
    plt.plot(xvalues_for_plot,yvalues_for_plot, c='b',
             label = 'Scipy curve fit')
    plt.title(plot_title)
    plt.xlabel(xunit)
    plt.ylabel(yunit)
    plt.legend()
    plt.grid()
    plt.savefig(plot_title+'.png')
    plt.show()

    return None

def error_plot(model,p_opt,xdata,ydata,y_uncer,xunit,yunit,
                        plot_title):
    #Now we create some data points on the model curve for plotting
    xvalues_for_plot = np.linspace(xdata[0],xdata[-1],1000)
    yvalues_for_plot = []
    for i in xvalues_for_plot:
        yvalues_for_plot.append(model(i,p_opt[0],p_opt[1]))

    #Now we plot the original data with error bars, along with the curve fit model
    plt.figure(figsize=(10,5))
    plt.errorbar(xdata,ydata,y_uncer,c='r', ls='', marker='o',lw=1,capsize=2,
                 label = 'Points of measurement with uncertainty')
    plt.plot(xvalues_for_plot,yvalues_for_plot, c='b',
             label = 'Scipy curve fit')
    plt.title(plot_title)
```

```
    plt.xlabel(xunit)
    plt.ylabel(yunit)
    plt.legend()
    plt.grid()
    plt.savefig(plot_title+'.png')
    plt.show()
    return None

def chi2(y_measure,y_predict,errors):
    """Calculate the chi squared value given a measurement with errors and
    prediction"""
    return np.sum( np.power(y_measure - y_predict, 2) / np.power(errors, 2) )

def chi2reduced(y_measure, y_predict, errors, number_of_parameters):
    """Calculate the reduced chi squared value given a measurement with errors
    and prediction, and knowing the number of parameters in the model."""
    return chi2(y_measure, y_predict, errors)/ \
            (y_measure.size - number_of_parameters)

def read_data(filename, Del, skiprows):
    """Load give\n file as csv with given parameters,
    returns the unpacked values"""
    return np.loadtxt(filename,
                      skiprows=skiprows,
                      delimiter=Del,
                      unpack=True)

def fit_data(model_func, xdata, ydata, yerrors, guess):
    """Utility function to call curve_fit given x and y data with errors"""
    popt, pcov = optim.curve_fit(model_func,
                                 xdata,
                                 ydata,
                                 absolute_sigma=True,
                                 sigma=yerrors,
                                 p0=guess)

    pstd = np.sqrt(np.diag(pcov))
    return popt, pstd
```

A.4.2 Code 1.py

```
"""
@author: Fredrik Dahl Braten
"""

#Importing modules
import numpy as np
import matplotlib.pyplot as plt
import Functions as F


#we find our estimates of the ammeters resistance:
R_A_estimates_PS = []

#Looping through the voltages, V_inf, that we used.
for i in [6,10,15,20]:
    #Importing data
    Resistance1, Voltage1, Current1 = F.read_data('setup1-dc-'+str(i)+'.csv',
                                                  ',',1)
    Resistance2, Voltage2, Current2 = F.read_data('setup2-dc-'+str(i)+'.csv',
                                                  ',',1)
    #Now we change units to SI units:
    Resistance1, Resistance2, Current2, Current1 = (Resistance1*1000,
                            Resistance2*1000, Current2/1000, Current1/1000)
    a,b = 0,5
    Resistance1, Voltage1, Current1, Resistance2, Voltage2, Current2 = (
        Resistance1[a:b], Voltage1[a:b], Current1[a:b], Resistance2[a:b],
        Voltage2[a:b], Current2[a:b])
    #Now we calculate the resistance corresponding to each resistor measurement
    for j in range(len(Resistance1)):
        R_A_estimates_PS.append((Voltage1[j]-Voltage2[j])/Current1[j])

        print("The current through the voltmeter is:", Current2[j]-Current1[j],
              "Ampere when the total current throught the circuit is",
          Current2[j], "Ampere, the terminal voltage is:",i,"volts, and the",
          "resistor in the circuit is", Resistance2[j], "Ohm.\n")
#Our estimate of the resistance is the average of all these
R_A_Estimate_PS = np.mean(R_A_estimates_PS)

#Now we calculate the standard deviation of the resistances.
#This will be the uncertainties of our estimate
R_A_sd_PS = np.std(R_A_estimates_PS)/np.sqrt(len(R_A_estimates_PS))

print("Our estimate of the resistance of the ammeter is:",R_A_Estimate_PS,"+-",
```

```
R_A_sd_PS)
```

A.4.3 Code 2.py

```
#!/usr/bin/env python3
# -*- coding: utf-8 -*-
"""
@author: Pankaj Patil and Fredrik Dahl Braten
"""

import statslab as utils
import matplotlib.pyplot as plt
import numpy as np


voltage_uncertainity = .1

def model_function(currrent, v_infty, resistance):
    return  v_infty - resistance * currrent

#Looping through the voltages for which the power supply outputted
#We would like to remember the max current for each of these voltages:
max_currents = []
for i in [6,10,15,20]:
    # import the first setup for Voltage data
    setup_1_file = "setup1-dc-"+str(i)+".csv"
    measured_voltages = utils.read_data(setup_1_file,
                                                    usecols=(2),
                                                    skiprows=1)

    # import the first setup for Current data
    setup_2_file = "setup2-dc-"+str(i)+".csv"
    measured_currents = utils.read_data(setup_2_file,
                                                    usecols=(1),
                                                    skiprows=1)

    voltage_errors = np.ones_like(measured_voltages) * voltage_uncertainity

    popt, pstd = utils.fit_data(model_function,
                                measured_currents,
                                measured_voltages,
                                voltage_errors)

    print("for voltage of the power supply equal to:", i,"Volts, we have")
    print("V_infty = %.2f V" %  popt[0])
    print("R_internal = %.2f ohm" %  (popt[1] * np.power(10, 6)))
```

```
    currrent_data_for_prediction = np.linspace(measured_currents[0],
                                               measured_currents[-1], 1000)

    predicted_voltages = model_function(currrent_data_for_prediction,
                                               popt[0],
                                               popt[1])

    chi2r_curve_fit = utils.chi2reduced(measured_voltages,
                                model_function(measured_currents,
                                                    popt[0],
                                                    popt[1]),
                                voltage_errors,
                                2)
    # finding the maximum current:
    max_currents.append(i/(popt[1] * np.power(10, 6)))


    plot_data = utils.plot_details("Power Supply 20V (Voltage Vs Current))")
    plot_data.errorbar_legend("Measured Voltage")
    plot_data.fitted_curve_legend("Curve Fit $V = V_{\infty} - RI$")
    plot_data.x_axis_label("Current mA")
    plot_data.y_axis_label("Voltage (V)")
    plot_data.xdata(measured_currents)
    plot_data.ydata(measured_voltages)
    plot_data.yerrors(voltage_errors)
    plot_data.xdata_for_prediction(currrent_data_for_prediction)
    plot_data.ydata_predicted(predicted_voltages)
    plot_data.legend_position("upper right")
    plot_data.chi2_reduced(chi2r_curve_fit)

    # plot the data
    utils.plot(plot_data)

    plt.savefig("Power supply 20V lab5_voltage_vs_current.png")
    print("chi2reduced = %.8f" % chi2r_curve_fit)


#Now we can plot the max current relationship as a function of V_inf:
fig = plt.figure(figsize=(8,6))
ax = fig.add_subplot(1,1,1)
ax.set_title("Estimated Max Current vs. V_inf based on our linear models")
ax.plot([6,10,15,20], max_currents,lw=1,c='r')
ax.set_xlabel("V_inf (Volts)")
ax.set_ylabel("Max Current (Ampere)")
ax.figure.savefig("Estimated Max Current vs. V_inf based on our linear models"+".png")
plt.show()
```

References

[1] Lab Manual - OutputResist.pdf