PyLab 4 – Nonlinear Fitting Methods II

October 16, 2018 – Ayush Pandhi (1003227457) and Xi Chen (1003146222)

1 Abstract

Blackbody radiation follows a well-defined temperature relation such that, $P = A\sigma T^4$. However, a less ideal physical scenario is considered with an incandescent lightbulb that contains a tungsten filament. An approximation of this relationship is modelled using both a linear and nonlinear regression method. Both models prove to be very good fits to the data as the expected exponent parameter is 5.882 x 10^{-1} and they produce estimated exponents: 5.562 x 10^{-1} $\pm$ 2.95 x 10^{-3} (linear) and 5.562 x 10^{-1} $\pm$ 2.97 x 10^{-3} (nonlinear). It is also found that χ^2_{linear} = 1.04 and $\chi^2_{nonlinear}$ = 1.04, thus bolstering the claim that the modelled fits are good approximations with statistical analysis.

2 Introduction

The purpose of this experiment is to observe the power law for blackbody radiation through experimental voltage and current data collected for a lightbulb and fit it with linear and nonlinear methods of regression. The lightbulb in question is incandescent and contains tungsten filament. In this case, a theoretical ideal blackbody would follow the power law:

$$P = A\sigma T^4$$

Where P is the power, T is the temperature in kelvin and σ is the Stefan-Boltzmann constant. However, this experiment is not dealing with an ideal blackbody; instead the tungsten filament acts as a "grey body" where it is expected that the resistance scales as:

$$R\ \alpha\ T^{1.209}$$

Using this relationship alongside the fact that $P = VI$ and $V = IR$, a power law between current and voltage can be determined. For an ideal blackbody and a more physically accurate tungsten model, this power law is found to be:

$$I_{blackbody} \alpha\ V^{3/5} \quad and \quad I_{tungsten} \alpha\ V^{0.5882}$$

We aim to model the voltage-current power law relationship with both linear and nonlinear fitting; the linear fitting involves turning the exponential expression into a linear equation in terms of the logarithm of each term. The nonlinear model will directly use an exponential form similar to above to fit the collected data.

3 Methods and Materials

The materials utilized for this experiment were: a power supply, two multimeters, a board with electrical components (with an incandescent lightbulb), a sufficient amount of wires to create the required circuit, and the Jupyter Notebooks software.

4 Experimental Procedure

The experiment was set up by connecting the two multimeters, power supply and a lightbulb as seen in Figure 1.

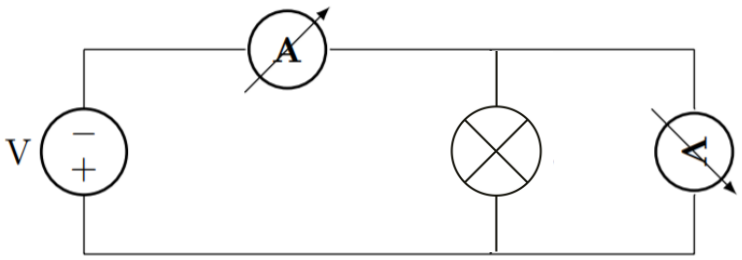

Figure 1: A circuit diagram for the experimental setup.

The power supply was set to the lowest setting and the voltage (V) and current (mA) was recorded from the multimeters. This process was repeated 15 times where the power supply output was increased by a small interval after each measurement. The voltage and current data was stored in a .txt file with the uncertainties of each measurement being computed in python by finding the maximum value between the Error of Accuracy and Error of Precision.

5 Results

Using the observational data gathered from the experiment, a linear and non-linear curve fit is generated with scipy's "curve_fit()" module. The linear modelling is based on the function f, where f is defined as a general linear formula of the form:

$$f = bx + \log(a)$$

For the curve fit on observational data, the function f is the logarithmic current, the independent variable x is the logarithmic voltage, the slope b is power law of in the relationship between current and voltage and a is a constant of proportionality. Therefore, rewriting this power law to be representative of the linear model specifically applied to our data gives:

$$\log(I) = blog(V) + log(a)$$

Where I is the current and V is the voltage within the circuit. Plotting the linear model fit with estimated parameters, (a = 0.0085, b = 3/5), produces a fit that, by visual inspection, serves as a good representation of the data set.

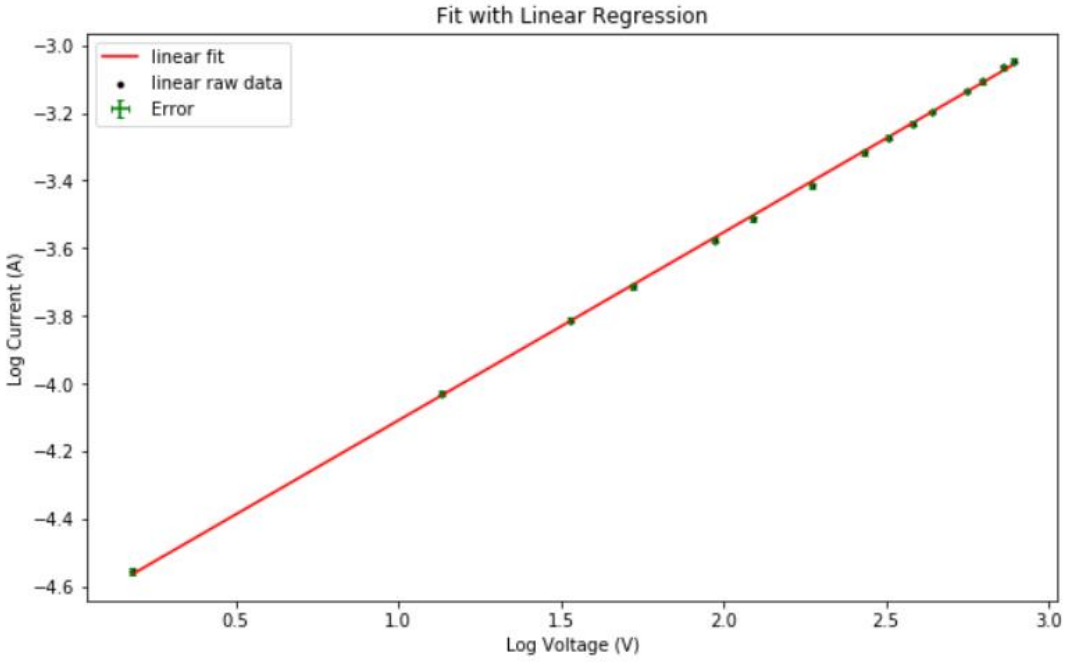

Figure 2: *Linear model using curve_fit() over-plotted against raw observational data points with corresponding errors.*

The nonlinear model is described by the function g, where g is defined as a general exponential formula:

$$g = ax^b$$

For the emission data, the function g represents the current and the independent variable, x, is the voltage. Here a and b are defined to represent the same parameters as mentioned earlier for the linear model. For this experiment, the nonlinear fit effectively represents the equation:

$$\mathrm{I} = aV^b$$

Plotting the nonlinear model fit with estimated parameters, (a = 1, b =3 /5), produces a fit that is also very closely resembles the raw experimental data. From visual comparison, there is not a clear answer as to which fit (linear or nonlinear) produces a better approximation for the observed data.

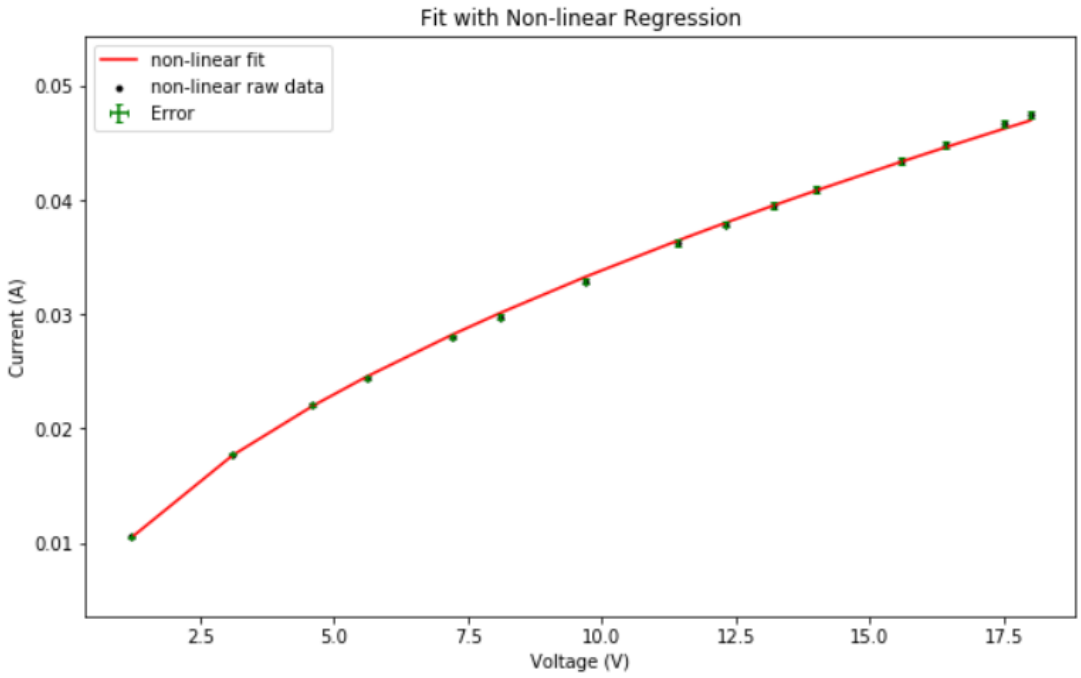

Figure 3: *Nonlinear model using curve_fit() over-plotted against raw observational data points with corresponding errors.*

The power law relationship for each model can be computed by analyzing the b parameter estimated for each fit:

$$Linear\ Model: I\ \alpha\ V^{0.5562}$$

$$Nonlinear\ Model: I\ \alpha\ V^{0.5562}$$

The resulting estimations of the b parameter are 5.562 x 10^{-1} $\pm$ 2.95 x 10^{-3} (linear) and 5.562 x 10^{-1} $\pm$ 2.97 x 10^{-3} (nonlinear). The expected value of this parameter from a physically accurate model of tungsten is 5.882 x 10^{-1}. By this measure, both models for the curve fit produce a fairly accurate representation of the theoretical power law. However, the difference between the two fits is very small (smaller than four decimal points) and can be considered negligible as the b parameter for each fit is within the uncertainty margin of the other.

A further analysis is performed by over-plotting the theoretical curve, which follows the expected power law for tungsten, alongside both curve fit models and raw data for both standard axes and a logarithmic axes scale.

$$Theoretical\ Model: I\ \alpha\ V^{0.5882}$$

It is difficult to definitively conclude that either curve fit model is significantly better than the other from this as there is no clear distinction between them from a purely visual analysis either.

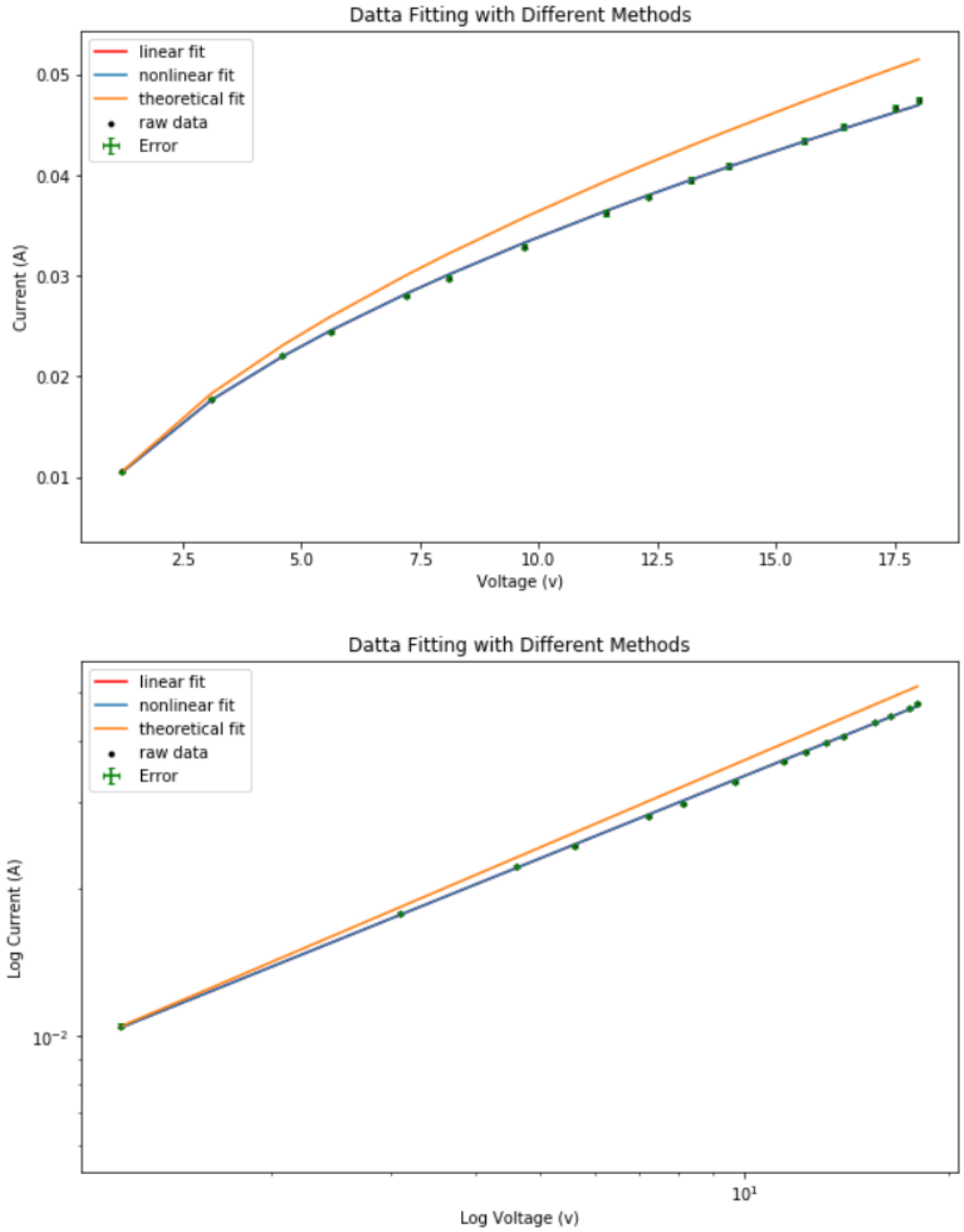

Figure 4: *All models over-plotted against observational data with corresponding error; (Top) Standard axes, (Bot) Log axes.*

Looking at the χ^2 values it is computed that χ^2_{linear} = 1.04 and $\chi^2_{nonlinear}$ = 1.04, which suggests that both models are extremely good fits for the data. Again, it is difficult to discern if one model is clearly a more accurate fit over the other using this statistical analysis. This agrees with the previous results and it is concluded that both fits represent the data extremely accurately and to the same degree of success.

6 Discussion

It's found that both curve fit models, linear and nonlinear, provide accurate representations of the experimental data set. This is firstly shown by the computed b parameters of each fit: 5.562 x 10^{-1} $\pm$ 2.95 x 10^{-3} (linear) and 5.562 x 10^{-1} $\pm$ 2.97 x 10^{-3} (nonlinear). The expected value for this parameter is 5.882 x 10^{-1}, which is very close to both approximations although not actually within the margin of error on either. Additionally, it can be seen in Figure 3 that the linear and nonlinear model are nearly indistinguishable by visual inspection and line up extremely well with the raw data points. Statistically, the χ^2 values for each respective data set: χ^2_{linear} = 1.04 and $\chi^2_{nonlinear}$ = 1.04, prove that the two fits are very similar. They are both slight very good approximations for the observed set of data.

The uncertainty in the experiment first arises from error in the experimental values of V and I, for which we take the error to be the maximum between the error of accuracy and error of precision. The individual uncertainty values for the resistor experiment are: for V (0.01, 0.01, 0.0115, 0.014, 0.018, 0.02025, 0.02425, 0.0285, 0.03075, 0.033, 0.035, 0.039, 0.041, 0.04375, 0.045) and for I (0.0001, 0.0001335, 0.00016575, 0.000183, 0.00021, 0.0002235, 0.00024675, 0.00027225, 0.00028425, 0.00029625, 0.00030675, 0.00032625, 0.000336, 0.00035025, 0.00035625). Furthermore, when dealing with the transformation of the data to a logarithmic scale in the linear model, the uncertainties follow the standard propagation rule for logarithms:

$$\sigma_{\log(y_i)} = \left|\frac{\sigma_{y_i}}{y_i}\right|$$

Finally, the error margin for the computed b parameters of each respective curve fit model is found as the variance of b, $Var(b)$, which is found as a diagonal element of the covariance matrix provided by the curve fitting module.

The strength of this experimental design is that it is quite simple to set up and replicate the results. Additionally, the uncertainties mostly stem from the error of accuracy and precision of the electronic equipment, which tend to be rather small. Thus the uncertainty on the final results is also small and we achieve accurate results compared to the theoretical predictions. Overall, the experiment was a replicable setup with minimal uncertainties and this reflected well in the highly accurate results; thus there are no obvious adjustments that should be made to improve the system.

7 Numbered Questions

Question 1:

Both regression methods actually provided a nearly identical fit to the data (to the presented significant figures). The results show that the b parameters of each fit are: 5.562 x 10^{-1} $\pm$ 2.95 x 10^{-3} (linear) and 5.562 x 10^{-1} $\pm$ 2.97 x 10^{-3} (nonlinear). These exponent values were reasonably close to the theoretical value of 5.882 x 10^{-1}, however it should be noted that the expected value for this parameter does not fall within the error margin of either curve fit. In terms of the plots, the difference is so miniscule that you cannot tell them apart visually in Figure 4 as both curve fit models are directly on top of one another.

Question 2:

The results for the standard deviations of each parameter were found as the square root of their variance which is given in their respective covariance matrices. Specifically, the first diagonal element corresponds to the variance of the a parameter and the second diagonal element corresponds to the variance of the b parameter. The standard deviation in the a parameter is: 6.401 x 10^{-5} (linear) and 6.437 x 10^{-5} (nonlinear); and for the b parameter: 2.95 x 10^{-3} (linear) 2.97 x 10^{-3} (nonlinear). Looking at the fitted exponent values with these corresponding standard deviations from Question 1, it is determined that this result does not fall into the range of either the blackbody value (3/5) or the expected value for tungsten (5.882 x 10^{-1}).

Question 3:

The χ^2 values that were computed for each respective model are: χ^2_{linear} = 1.04 and $\chi^2_{nonlinear}$ = 1.04. This suggests that both models provide very accurate fit for the data set and are indeed very similar to one another as was found in previous questions and results. If we're being really picky, since the χ^2 values are still greater than 1, they are incomplete fits technically speaking. However, since the results are so close to 1, it is concluded that the curve fits model the data extremely well.

8 Conclusion

From the experiment we conclude that both the linear and nonlinear curve fitting methods produced a very accurate approximation of the voltage-current data set for an incandescent lightbulb containing tungsten filament. The resulting exponent parameters from the regression methods (5.562 x 10^{-1} $\pm$ 2.95 x 10^{-3} (linear) and 5.562 x 10^{-1} $\pm$ 2.97 x 10^{-3} (nonlinear)) were reasonably close to the expected blackbody (3/5) and tungsten (5.882 x 10^{-}1) exponents, however neither model's results statistically fall into the range of the expected parameters with their standard deviations. Additionally, from inspecting the χ^2 values (χ^2_{linear} = 1.04 and $\chi^2_{nonlinear}$ = 1.04), it is concluded that both regression methods successfully modelled the given data set.

9 References

[1] Pandhi, A. and Chen, X., University of Toronto, Toronto, ON. “PHY224 Laboratory Notes: PyLab 4 – Nonlinear Fitting Methods II”, October 2018.

10 Appendices

The full code used for this lab can be found below as well as on the author’s Github as “PyLab 4.py”: https://github.com/AyushPandhi/Pandhi_Ayush_PHY224/tree/master/PyLab%204.

```
#PyLab 4: Nonlinear Fitting Methods II
#Author: Ayush Pandhi (1003227457)
#Date: 10/16/2018

#Importing required modules
import numpy as np
import matplotlib.pyplot as plt
from scipy.optimize import curve_fit

#Defining the linear model function
def f(x, a, b):
    return b*x + np.log(a)

#Defining the exponential model function
def g(x, a, b):
    return a*(x**b)

#Loading the data and uncertainity
voltage = np.loadtxt('Lab 4 data.txt', skiprows=1, usecols=(0,))
current = (1/1000)*(np.loadtxt('Lab 4 data.txt', skiprows=1, usecols=(1,)))

#Finding max of precision and accuracy error for resistor data
v_error = np.empty(len(voltage))
for i in range(len(voltage)):
    v_error[i] = max(voltage[i]*0.0025, 0.01)

i_error = np.empty(len(current))
for i in range(len(current)):
    i_error[i] = max(current[i]*0.0075, 0.1/1000)

#Getting log values with errors
```

```
log_v = np.log(voltage)
log_i = np.log(current)
i_error_log = i_error/current

#Linear regression
p_opt_1, p_cov_1 = curve_fit(f, log_v, log_i, (0.0085, 3/5), i_error_log, True)
lin_output = f(log_v, p_opt_1[0], p_opt_1[1])

#Non-linear regression
p_opt_2, p_cov_2 = curve_fit(g, voltage, current, (1, 3/5), i_error, True)
nonlin_output = g(voltage, p_opt_2[0], p_opt_2[1])

#Checking which fit's exponential is closer to the expected
print('The exponent using linear regression is', p_opt_1[1])
print('The exponent using nonlinear regression is', p_opt_2[1])
print('Linear fit power law is: I ~ V^' + str(p_opt_1[1]))
print('Non-linear fit power law is: I ~ V^' + str(p_opt_2[1]))

#Plots of linear regression
plt.figure(figsize=(10,6))
plt.scatter(log_v, log_i, label = 'linear raw data', marker='.', color='k')
plt.plot(log_v, lin_output, 'r-', label = 'linear fit')
plt.title('Fit with Linear Regression')
plt.xlabel('Log Voltage (V)')
plt.ylabel('Log Current (A)')
plt.errorbar(log_v, log_i, xerr=0, yerr=i_error_log, linestyle='none', ecolor='g', label='Error', capsize=2)
plt.legend()
plt.show()

#Plots of non-linear regression
plt.figure(figsize=(10,6))
plt.scatter(voltage, current, label = 'non-linear raw data', marker='.', color='k')
plt.plot(voltage, nonlin_output, 'r-', label = 'non-linear fit')
plt.title('Fit with Non-linear Regression')
plt.xlabel('Voltage (V)')
plt.ylabel('Current (A)')
plt.errorbar(voltage, current, xerr=0, yerr=i_error, linestyle='none', ecolor='g', label='Error', capsize=2)
plt.legend()
plt.show()

#Overplotting linear fit, non-linear fit and theorethical
plt.figure(figsize=(10,6))

#Raw data
```

```
plt.scatter(voltage, current, label = 'raw data', marker='.', color='k')

#Linear fit
lin_fit = p_opt_1[0]*voltage**(p_opt_1[1])
plt.plot(voltage, lin_fit, 'r-', label='linear fit')

#Nonlinear fit
nonlin_fit = p_opt_2[0]*voltage**(p_opt_2[1])
plt.plot(voltage, nonlin_fit, label='nonlinear fit')

#Theoretical fit
theo_fit = p_opt_2[0]*voltage**0.5882
plt.plot(voltage, theo_fit, label='theoretical fit')
plt.title('Datta Fitting with Different Methods')
plt.xlabel('Voltage (v)')
plt.ylabel('Current (A)')
plt.errorbar(voltage, current, xerr=0, yerr=i_error, linestyle='none', ecolor='g', label='Error', capsize=2)
plt.legend()
plt.show()

#Overplotting linear fit, non-linear fit and theorethical
plt.figure(figsize=(10,6))

#Raw data
plt.scatter(voltage, current, label = 'raw data', marker='.', color='k')

#Linear fit
lin_fit = p_opt_1[0]*voltage**(p_opt_1[1])
plt.plot(voltage, lin_fit, 'r-', label='linear fit')

#Nonlinear fit
nonlin_fit = p_opt_2[0]*voltage**(p_opt_2[1])
plt.plot(voltage, nonlin_fit, label='nonlinear fit')

#Theoretical fit
theo_fit = p_opt_2[0]*voltage**0.5882
plt.plot(voltage, theo_fit, label='theoretical fit')
plt.title('Datta Fitting with Different Methods')
plt.xlabel('Log Voltage (V)')
plt.ylabel('Log Current (A)')
plt.xscale('log')
plt.yscale('log')
plt.errorbar(voltage, current, xerr=0, yerr=i_error, linestyle='none', ecolor='g', label='Error', capsize=2)
plt.legend()
```

```
plt.show()

#Standard deviations for each model and parameter
sigma_a1 = np.sqrt(p_cov_1[0,0])
sigma_b1 = np.sqrt(p_cov_1[1,1])
sigma_a2 = np.sqrt(p_cov_2[0,0])
sigma_b2 = np.sqrt(p_cov_2[1,1])
print('Standard deviation for parameter a for linear regression is', sigma_a1)
print('Standard deviation for parameter b for linear regression is', sigma_b1)
print('Standard deviation for parameter a for nonlinear regression is', sigma_a2)
print('Standard deviation for parameter b for nonlinear regression is', sigma_b2)

#Calculating chi squared
chi_sq_1 = (1/13)*(np.sum(((log_i - lin_output) / i_error_log)**2))
print('chi squared for linear regression is', chi_sq_1)
chi_sq_2 = (1/13)*(sum(((current - nonlin_output) / i_error)**2))
print('chi squared for nonlinear regression is', chi_sq_2)
```