Effect of Linear Polarizers and Waveplates on Helium-Neon Laser

Aditya K. Rao[a)]

University of Toronto
MP222, 60 St. George Street, Toronto, Ontario M5S 1A7, Canada.

[a)]adi.rao@mail.utoronto.ca; Student Number: 1008307761

Abstract. Three experiments were conducted to verify the behaviour of light when intereacting with linear polarizers and waveplates. Two experimental setups were utilized with a proposed third also being outlined. The first to verify Malus' Law for a two polarizer setup on the intensity of a Helium-Neon laser. It was found that the maximum transmitted intensity occurs at $\theta + \phi = 2.33 \pm 0.02\,\mathrm{rad}$ or $43 \pm 1^\circ$ considering a phase shift. Placing a waveplate between two orthogonal polarizers yielded $7.6 \pm 0.1\,\mathrm{V}$ ($5 \pm 3\%$ decrease) for the Half Waveplate and $5.66 \pm 0.03\,\mathrm{V}$ ($30 \pm 3\%$ decrease) for the Quarter Waveplate. These are both in good agreement with theory when considering the non-ideal nature of physical waveplates.

CONTENTS

INTRODUCTION

Electromagnetic waves are modeled as transverse plane waves that oscillate in the direction perpendicular to their propagation. The electric field of these waves can be described by (1).

$$\vec{E}_0 = E_i\hat{i} + E_J\hat{j} \tag{1}$$

One properties of such waves is their ability to become 'polarized'. This annihlates all other oscillations apart from a single plane. This can be achieved by passing the wave through a polarizer. The intensity of the light that passes through the polarizer is given by (2).

$$I = I_0\cos^2(\theta) \tag{2}$$

In this experiment, the power is recorded as opposed to the intensity. However, because the power is proportional to the intensity, the same equation can be used. The power of an Electromagnetic wave is directly proportional to the product of its electric field and its complex conjugate. This is given by (3).

$$P \propto |E_0|^2 \tag{3}$$

Waveplates are another common optical element which, instead of restricting the oscillations of the electric field, shift the phase of the wave. This is done by changing the refractive index of the material.

$$\hat{\Pi}_\theta = \begin{bmatrix} \cos^2(\theta) & \sin(\theta)\cos(\theta) \\ \sin(\theta)\cos(\theta) & \sin^2(\theta) \end{bmatrix} \tag{4}$$

$$\hat{J} = \chi\hat{\Pi}_{0'} + \zeta\hat{\Pi}_{1'} \tag{5}$$

Here, two different types of waveplate will be investigated: quarter and half waveplates. The quarter waveplate shifts the phase of the wave by $\pi/2$ radians. The half waveplate shifts the phase of the wave by π radians. The resultant effect on the power should be that the power is unchanged for the half wave plate and that the power is reduced for the quarter waveplate due to destructive and constructive interference of the E_x and E_y components.

The jones matrix for a waveplate is given by (5). Here, χ and ζ are the complex amplitudes of the electric field. For the quarter waveplate, the Jones matrix is given by (6). For the half waveplate, the Jones matrix is given by (7). Note that both of these matrices are with respect to the horizontal and represent the fast axis.

$$\hat{J}_{QWP} = \frac{1}{\sqrt{2}}\begin{bmatrix} \cos^2\theta + i\sin^2\theta & (1-i)\sin\theta\cos\theta \\ (1-i)\sin\theta\cos\theta & \sin^2\theta + i\cos^2\theta \end{bmatrix} \tag{6}$$

$$\hat{J}_{HWP} = \begin{bmatrix} \cos 2\theta & \sin 2\theta \\ \sin 2\theta & -\cos 2\theta \end{bmatrix} \tag{7}$$

In order to predict the power output after the waveplate and polarizers, one can perform the following Jones matrix multiplication for the Quarter Waveplate.

$$\begin{aligned}
\vec{J_{QWP}}' &= \frac{1}{\sqrt{2}}\hat{J}_0\hat{J}_{QWP}\hat{J}_1\vec{J} \\
&= \frac{1}{\sqrt{2}}\begin{bmatrix} 1 & 0 \\ 0 & 0 \end{bmatrix}\begin{bmatrix} \cos^2\theta + i\sin^2\theta & (1-i)\sin\theta\cos\theta \\ (1-i)\sin\theta\cos\theta & \sin^2\theta + i\cos^2\theta \end{bmatrix} \\
&\cdot \begin{bmatrix} 0 & 0 \\ 0 & 1 \end{bmatrix}\begin{bmatrix} E_x \\ E_y \end{bmatrix} \\
&= \frac{1}{\sqrt{2}}\begin{bmatrix} \cos^2\theta + i\sin^2\theta & (1-i)\sin\theta\cos\theta \\ 0 & 0 \end{bmatrix}\begin{bmatrix} 0 \\ E_y \end{bmatrix} \\
&= \frac{1}{\sqrt{2}}\begin{bmatrix} E_y(1-i)\sin\theta\cos\theta \\ 0 \end{bmatrix}
\end{aligned}$$

The power then follows from (3).

$$\begin{aligned}
P &= 4(E_y(1-i)\sin\theta\cos\theta)\cdot(E_y(1-i)\sin\theta\cos\theta)^* \\
&= 4|E_y|^2\sin^2\theta\cos^2\theta
\end{aligned}$$

Note that E_y is just a relative angle, hence, we obtain (8). Where E_0 is the initial power of the wave out of the first polarizer.

$$P = 4|I_0|^2 \sin^2\theta\cos^2\theta \tag{8}$$

For the Half Waveplate, the calculation is as follows.

$$\begin{aligned}
\vec{J_{HWP}}' &= \hat{J}_0\hat{J}_{HWP}\hat{J}_1\vec{J} \\
&= \begin{bmatrix} 1 & 0 \\ 0 & 0 \end{bmatrix}\begin{bmatrix} \cos 2\theta & \sin 2\theta \\ \sin 2\theta & -\cos 2\theta \end{bmatrix}\begin{bmatrix} 0 & 0 \\ 0 & 1 \end{bmatrix}\begin{bmatrix} E_x \\ E_y \end{bmatrix} \\
&= \begin{bmatrix} \cos 2\theta & \sin 2\theta \\ 0 & 0 \end{bmatrix}\begin{bmatrix} 0 \\ E_y \end{bmatrix} \\
&= \begin{bmatrix} E_y \sin 2\theta \\ 0 \end{bmatrix}
\end{aligned}$$

Similarly, the power is given by (9).

$$P = |I_0|^2 \sin^2(2\theta) \tag{9}$$

From (9), one can deduce that there is an angle θ such that the power is maximized with the polarizers effectively becoming 'transparent'. This angle, from (9), should be at $\theta = \frac{\pi}{4}$ relative to the first polarizer.

METHODOLOGY

Two experiments were conducted in order to test and verify the effects of polarizrs and waveplates on a Helium-Neon (HeNe) laser with an additional third being planned. The experimental setup for each is shwon in Figures 1 and 2 respectively.

For each experiment, a `PDA015C2` photodiode was used to measure the intensity of the light. The photodiode was connected to a `DSO` oscilloscope which was used to measure the voltage output from the optical setup. The HeNe laser was the first component attached to the optical breadboard. All other components were then roughly placed in position (not clamped) after which fine & coarse adjustments were made to align the beam path.

To reduce ambient light a shroud was put over the photodiode. The resultent baseline voltage dropped from 2V to $\approx 240 \pm 10\,\mathrm{mV}$.

Polarizers

The second polarizer was removed from the post holder. The polarizer's angle was then adjusted such that the outputed power delivered to the photodiode was below its saturation voltage. This maximum power output was recorded at $V = 5.00V$. The angle of the polarizer was recorded as $157.69° \pm 0.01°$.

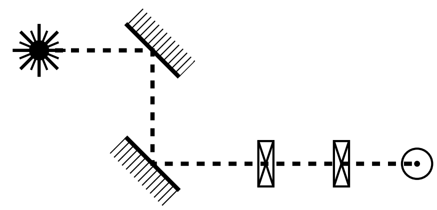

FIGURE 1: Experimental Setup for Experiment 1

This was kept the same for subsequent measurments. The second polarizer was then varied in angle from 0° to 270° in increments of 10°. The voltage output from the photodiode was recorded for each angle.

To reduce ambient light a shroud was put over the photodiode. The resultent baseline voltage dropped from 2V to $\approx 240 \pm 10\,\mathrm{mV}$.

Upon adding the second polarizer, massive fluctuations in the output voltage were observed (on the order of at least $0.5V$, but proportional to the output voltage). This was likely due to an issue with the battery of the photodiode. However, upon replacing the battery, no change in the output was observed. Therefore, the issue was likely within the output of the HeNe laser. The is documented furhter later in the report. In order to mitigate this abnormality, the output voltage was recorded at specific time intervals where the output voltage was semi-stable.

Waveplates

The experimental setup in Fig. 1 was modified to place a waveplate between the two polarizers. In the first part of the experiment, the effect of a quarter waveplate was tested. The waveplate was placed at an angle of 0° with respect to the horizontal. The second polarizer was then varied in angle from 0° to 220° i Vn increments of 20°. The voltage output from the photodiode was recorded for each angle.

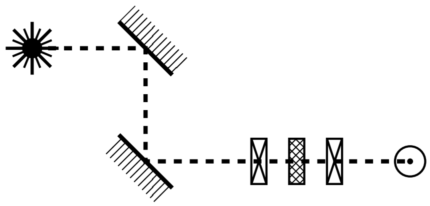

FIGURE 2: Experimental Setup for Experiment 2

This was then followed by a similar experiment with a half waveplate. The waveplate was placed at an angle of 0° with respect to the horizontal. The second polarizer was then varied in angle from 0° to 220° in increments of 20°. The voltage output from the photodiode was recorded for each angle.

Dielectrics

Due to time constraints, this experiment was not completed in full. The proposed setup is shown in Fig. 3. The setup was to be similar to the previous two experiments, with the addition of a dielectric material between the two polarizers. The dielectric material was to be placed at an angle of 0° with respect to the horizontal. The second polarizer was then varied in angle from 0° to 220° in increments of 20°. The voltage output from the photodiode was to be recorded for each angle.

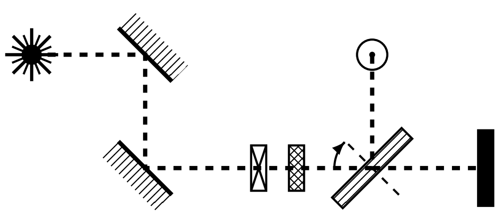

FIGURE 3: Proposed Experimental Setup for Experiment 3

The proposed experimental procedure would have been similar to the previous two experiments. However, instead of varying the angle of the polarizer, instead, the dielectric would have been rotated through 180° in increments of 10°. The resultant voltage output would have been recorded for each angle.

RESULTS & ANALYSIS

Polarizers

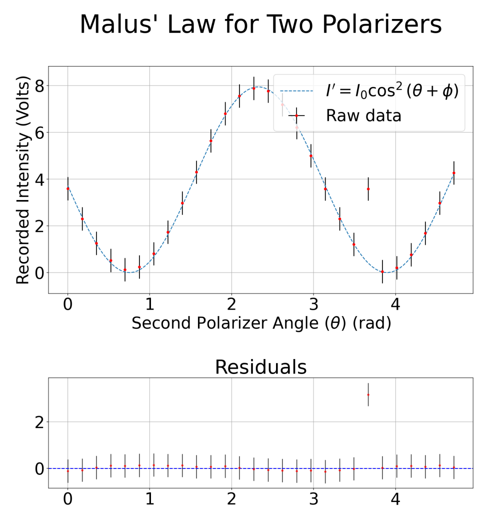

FIGURE 4: Experimental analysis of Malus' law for two linear polarizers. The maximum intensity was found to be $I_0 = 8.0 \pm 0.2\,\text{V}$. A relative offset between the two polarizers was found as $\phi = 0.82 \pm 0.02\,(\text{rad})$. A goodness-of-fit test yields a $\chi^2_{red} = 1.58$ indicating a good fit with slight underfitting. However, this is likely due to the distinct outlier present in the data as the residuals show no distinct pattern.

It can clearly be seen in Fig. 4 that the data adheres well to theory. All of the data points, apart from one outlier, falls well within the estimated uncertainty of the fitted curve given in (2). It should be stated that the χ^2 value of 1.58 is slightly high, however, the lack of a distinct pattern in the residuals indicate that this is a good fit regardless.

Parameter	Value	Uncertainty
I_0	8.0	0.2
ϕ	0.82	0.02

TABLE 1: Fit parameters for the Malus' Law verifcation experiment.

Quarter Waveplate

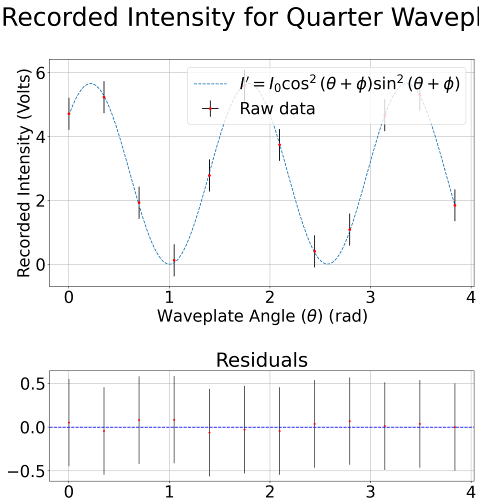

FIGURE 5: oltage Output observed using a `DET36A2` plotted against the angle of a Quarter Waveplate placed between two orthogonal linear polarizers. The fit parameters found are a maximum intensity of $I_0 = 5.66 \pm 0.03\,\mathrm{V}$ and a relative offset of $\phi = -1.002 \pm 0.002\,(\mathrm{rad})$. The $\chi^2_{red} = 0.013$, indicates significant overfitting. However, this is likely due to an overestimation of measurment uncertainty in conjunction with a small number of data points.

In Fig. 5, the data adheres well to the theoretical curve. The $\chi^2_{red} = 0.013$ indicates that the data is extremely overfit. However, the lack of a distinct pattern in the residuals indicates that this is likely due to an overestimation of measurement uncertainty as well as the low number of data points. Otherwise, the data adheres well with the theoretical curve indicated by (8).

Parameter	Value	Uncertainty
I_0 (V)	5.66	0.03
ϕ (rad)	-0.2175	0.004

TABLE 2: Fit parameters for the Quarter Waveplate experiment.

Half Waveplate

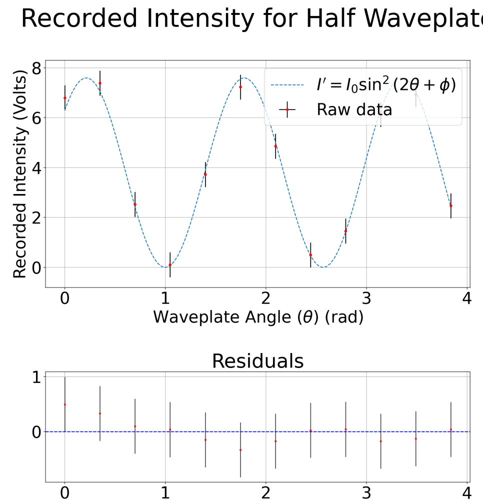

FIGURE 6: Voltage Output observed using a `DET36A2` plotted against the angle of a Half Waveplate placed between two orthogonal linear polarizers. The fit parameters found are a maximum intensity of $I_0 = 7.6 \pm 0.1\,\mathrm{V}$ and a relative offset of $\phi = 1.14 \pm 0.01$. The $\chi^2_{red} = 0.2$ indicates an overfitting to the data. However, given the lack of significant pattern in the residuals, this is likely due to an overestimation of the measurement uncertainty.

Similarly to the Quarter Waveplate, the data in Fig. 6 adheres well to the theoretical curve. The $\chi^2_{red} = 0.2$ indicates that the data is overfit. However, the lack of a distinct pattern in the residuals indicates the same as before. The parameters found in Table 3 are in good agreement with (9).

Parameter	Value	Uncertainty
I_0 (V)	7.6	0.1
ϕ (rad)	-0.21	0.01

TABLE 3: Fit parameters for the Half Waveplate experiment.

DISCUSSION

An interesting result can be observed in comparing Fig. 6 and Fig. 4. The recorded maximum intensity I_0 is almost (bar a factor $V = 0.4\,\mathrm{V}$) identical despite the linear

polarizers being orthogonal relateive to one another.
Plugging in the offset parameter, the angle θ for which these maxima can be found from (9).

$$
\begin{aligned}
2\theta + \phi &= 1.78 \pm 0.01 \\
\implies \theta &= \frac{1}{2} \cdot (1.78 \pm 0.01 - \phi) \\
&= 0.320 \pm 0.005 \,\text{rad} \\
&= 18.3 \pm 0.3^\circ
\end{aligned}
$$

If the offset is also included $\theta + \phi/2 = 0.890 \pm 0.007\,\text{rad} = 50.99 \pm 0.4^\circ$
Which is close to the expected value of 45°. This implies that the half waveplate makes the second linear polarizer effectively transparent at this angle.
Comparing this to the intensity observed in the Malus' law experiment, it can be seen that the maximum intensity is almost identical with a difference of $0.4 \pm 0.1\,\text{V}$. This is likely due to the fact that the polarizers and waveplates are non-ideal. When comparing this result to the the maximum intensity of the Quarter Waveplate, a decrease of $25 \pm 3\%$ is observed. This is exactly as expected from the theoretical prediction in (8) and (9).

CONCLUSION

Three experiments were conducted to investigate the effect of linear polarizers and common waveplates on a `HNLS008L` laser. Two experimental setups were utilized as can be seen in Fig. 1 and Fig. 2 with a third proposed setup in Fig. 3. Results were measured utilizing a `DET36A2` connected to a `DSOX1202G` oscilloscope with data analysis being conducted in python [3].
In the first experiment, it was found that the intensity of light passing through two linear polarizers was well described by Malus' Law. The maximum intensity was found to be $I_0 = 8.0 \pm 0.2\,\text{V}$ with a relative offset of $\phi = 0.82 \pm 0.02\,\text{(rad)}$ utilized as a correction parameter as the exact angle of the polarizers relative to the horizonal plane is not relavent. The data adhered well to the theoretical predication in (2) with a $\chi^2_{red} = 1.58$ indicating a good fit with slight underfitting.

In the second experiment, the setup was modified to test the effects of a Quarter and Half waveplate on the intensity of the HeNe beam. Both results qualitatively adhered well to the theoretical prediction in (8) and (9) respectively. However, the goodness-of-fit test indicates significant overfitting. This is likely due to an overestimation of the measurement uncertainty as well as the low number of data points. The maximum output intensity of the Half Waveplate was found to be $I_0 = 7.6 \pm 0.1\,\text{V}$ which is $\Delta I = 0.4 \pm 0.1, \text{V}$, approximately a $5 \pm 3\%$ decrease in intensity. The maximum output intensity of the Quarter Waveplate was found to be $I_0 = 5.66 \pm 0.03\,\text{V}$ which is $\Delta I = 2.3 \pm 0.2, \text{V}$, approximately an $30 \pm 3\%$ decrease in intensity. This is around 5% higher than expected, however, comparing this to the Half Waveplate, the decrease in intensity is exactly as expected with the Quarter waveplate having a $25 \pm 3\%$ decrease in intensity. This is likely due to the waveplates being non-ideal and absorbing some of the light.Considering this, it can be seen that at $\theta \approx 45^\circ$ relative to the first polarizer, the Half Waveplate makes the second polarizer effectively transparent.

Futher experiments may be conducted with a dielectric to investigate the varying polarizations of reflected light at varying angles of incidence. What should be observed is, at the citical angle, the light should be completely plane polarized.

ACKNOWLEDGMENTS

The work conducted by the other lab partners was instrumental in this lab. Thank you to Jack Wang and Yiheng Wang for their help in setting up the equipment and conducting the experiments. Additionally, thank you to the Teaching Assistant Colin Dale and Professor Boris Braverman for their guidance and support.

REFERENCES

1. B. Braverman, “Lab 1: Polarization,” (2025).
2. J. Peatross and M. Ware, *Physics of Light and Optics* (Brigham Young University, Department of Physics, Provo, Utah, 2015).
3. G. Van Rossum and F. L. Drake, *Python 3 Reference Manual* (CreateSpace, Scotts Valley, CA, 2009).
4. C. R. Harris, K. J. Millman, S. J. van der Walt, R. Gommers, P. Virtanen, D. Cournapeau, E. Wieser, J. Taylor, S. Berg, N. J. Smith, R. Kern, M. Picus, S. Hoyer, M. H. van Kerkwijk, M. Brett, A. Haldane, J. F. del Río, M. Wiebe, P. Peterson, P. Gérard-Marchant, K. Sheppard, T. Reddy, W. Weckesser, H. Abbasi, C. Gohlke, and T. E. Oliphant, “Array programming with NumPy,” Nature **585**, 357–362 (2020).
5. The pandas development team, “pandas-dev/pandas: Pandas,” (2020).

Appendix

Raw Data

Polarizer #1 (°)	Polarizer #2 (°)	Voltage (V)
159.69	NaN	0.05
159.69	0	3.6
159.69	10	2.3
159.69	20	1.26
159.69	30	0.53
159.69	40	0.13
159.69	50	0.24
159.69	60	0.81
159.69	70	1.74
159.69	80	2.98
159.69	90	4.3
159.69	100	5.64
159.69	110	6.81
159.69	120	7.56
159.69	130	7.89
159.69	140	7.77
159.69	150	7.19
159.69	160	6.21
159.69	170	5
159.69	180	3.58
159.69	190	2.3
159.69	200	1.21
159.69	210	3.58
159.69	220	0.05
159.69	230	0.21
159.69	240	0.77
159.69	250	1.69
159.69	260	2.98
159.69	270	4.27

TABLE 4: Raw data for the Malus’ Law verification experiment.

Polarizer #1 (°)	**Polarizer #2 (°)**	**QWP (°)**	**Voltage (V)**
159.69	220	0	4.71
159.69	220	20	5.23
159.69	220	40	1.93
159.69	220	60	0.13
159.69	220	80	2.78
159.69	220	100	5.59
159.69	220	120	3.74
159.69	220	140	0.41
159.69	220	160	1.09
159.69	220	180	4.67
159.69	220	200	5.31
159.69	220	220	1.85

TABLE 5: Raw data for the Quarter Waveplate experiment.

Polarizer #1 (°)	**Polarizer #2 (°)**	**HWP (°)**	**Voltage (V)**
159.69	220	0	6.8
159.69	220	20	7.39
159.69	220	40	2.53
159.69	220	60	0.11
159.69	220	80	3.73
159.69	220	100	7.23
159.69	220	120	4.86
159.69	220	140	0.5
159.69	220	160	1.46
159.69	220	180	6.13
159.69	220	200	6.93
159.69	220	220	2.47

TABLE 6: Raw data for the Half Waveplate experiment.

Analysis Code

```
#!/usr/bin/env python3
# -*- coding: utf-8 -*-
"""
Created on Mon Jan 27 11:35:21 2025

@author: Aditya K. Rao
@github: @adirao-projects
"""

import numpy as np
import pandas as pd
import toolkit as tk
import matplotlib.pyplot as plt

```

```
# uncert in voltage +- 0.01 V

def load_data(file):
    df = pd.read_csv(file)
    df = df.dropna()

    # Uncertainty in Output Voltage
    w_uncert = np.full(df.shape[0]+1, 0.5)

    # Uncertainty in Angle
    th_uncert = np.full(df.shape[0]+1, 0.01)

    df['Wu'] = pd.Series(w_uncert)
    df['Tu'] = pd.Series(th_uncert)

    return df


def model_func_malus(theta, I0, phi):
    return I0*((np.cos(theta+phi))**2)

def deg_rad(angle):
    return (angle/180)*(np.pi)

def rad_deg(angle):
    return (angle/np.pi)*(180)


def fit_plot(df):

    xdata = deg_rad(df['Pb'].to_numpy())
    ydata = df['W'].to_numpy()
    y_unc = df['Wu'].to_numpy()
    x_unc = deg_rad(df['Tu'].to_numpy())

    #plt.errorbar(xdata, ydata, yerr=y_unc, xerr=x_unc, fmt='o')

    #print(xdata)

    data = tk.curve_fit_data(xdata, ydata, fit_type='custom',
                        model_function_custom=model_func_malus,
                        uncertainty=y_unc, uncertainty_x=x_unc,
                        res=True, chi=True, guess=(7, np.pi/4))


    meta = {'title' : "Malus' Law for Two Polarizers\n",
            'xlabel' : r'Second Polarizer Angle ($\theta$) (rad)',
            'ylabel' : 'Recorded Intensity (Volts)',
            'chisq' : data['chisq'],
            'fit-label': r"$I' = I_0 \cos^2 (\theta+\phi)$",
            'data-label': "Raw data",
            'save-name' : 'Malus',
            'loc' : 'upper right'}

```

```python
    tk.quick_plot_residuals(xdata, ydata, data['plotx'], data['ploty'],
                            data['residuals'], meta=meta,
                            uncertainty=y_unc, uncertainty_x=x_unc,
                            save=True)

    max_I0 = np.max(data['ploty'])
    idx_maxI0 = np.where(data['ploty'] == max_I0)
    max_theta = data['plotx'][idx_maxI0]
    print(f'theta+phi max = {max_theta}')

    return data['chisq'], data['popt'], data['pstd']

if __name__ == '__main__':
    df = load_data('part1.csv')

    params = fit_plot(df)

    printvals = [r'$\chi_{red}^2'+f' = {params[0]}$']
    for i,v in enumerate([r'I_0', r'\phi$']):
        printvals.append(f'${v} = {params[1][i]} \pm {params[2][i]}$')

    tk.block_print(printvals, 'Fit Paramters for Two Polarizer')
```

```python
#!/usr/bin/env python3
# -*- coding: utf-8 -*-
"""
Created on Mon Feb 10 09:33:31 2025

@author: Aditya K. Rao
@github: @adirao-projects
"""

import numpy as np
import pandas as pd
import toolkit as tk
import matplotlib.pyplot as plt

# uncert in voltage +- 0.01 V

def load_data(file):
    df = pd.read_csv(file)
    df = df.dropna()

    # Uncertainty in Output Voltage
    w_uncert = np.full(df.shape[0]+1, 0.5)

    # Uncertainty in Angle
    th_uncert = np.full(df.shape[0]+1, 0.01)

    df['Wu'] = pd.Series(w_uncert)
    df['Tu'] = pd.Series(th_uncert)

    return df

```

```

def model_func_wp(theta, I0, phi, phase):
    return I0*((np.cos(phase*(theta+phi)))**2)

def model_func_qwp(theta, I0, phi):
    return 4*I0*((np.cos(theta+phi))**2)*((np.sin(theta+phi))**2)

def model_func_hwp(theta, I0, phi):
    return I0*((np.sin(2*theta+phi))**2)

def deg_rad(angle):
    return (angle/180)*(np.pi)

def rad_deg(angle):
    return (angle/np.pi)*(180)


def fit_plot(df, plate):

    xdata = deg_rad(df[plate].to_numpy())
    ydata = df[’W’].to_numpy()
    y_unc = df[’Wu’].to_numpy()
    x_unc = deg_rad(df[’Tu’].to_numpy())

    #plt.errorbar(xdata, ydata, yerr=y_unc, xerr=x_unc, fmt=’o’)

    #data = tk.curve_fit_data(xdata, ydata, fit_type=’custom’,
    #                  model_function_custom=model_func_qwp,
    #                  uncertainty=y_unc, uncertainty_x=x_unc,
    #                  res=True, chi=True, guess=(8, 0.5, 1.5))

    #print(xdata)
    if plate == ’QWP’:
        plate_name = ’Quarter’
        data = tk.curve_fit_data(xdata, ydata, fit_type=’custom’,
                          model_function_custom=model_func_qwp,
                          uncertainty=y_unc, uncertainty_x=x_unc,
                          res=True, chi=True,)

        meta = {’title’ : f"Recorded Intensity for {plate_name} Waveplate\n",
                ’xlabel’ : r’Waveplate Angle ($\theta$) (rad)’,
                ’ylabel’ : ’Recorded Intensity (Volts)’,
                ’chisq’ : data[’chisq’],
                ’fit-label’: r"$I’ = I_0 \cos^2 (\theta+\phi)\sin^2(\theta+\phi)$",
                ’data-label’: "Raw data",
                ’save-name’ : f’{plate_name}’,
                ’loc’ : ’upper right’}


    elif plate == ’HWP’:
        plate_name = ’Half’
        data = tk.curve_fit_data(xdata, ydata, fit_type=’custom’,
                          model_function_custom=model_func_hwp,
                          uncertainty=y_unc, uncertainty_x=x_unc,
```

```
                          res=True, chi=True, )

        meta = {'title' : f"Recorded Intensity for {plate_name} Waveplate\n",
            'xlabel' : r'Waveplate Angle ($\theta$) (rad)',
            'ylabel' : 'Recorded Intensity (Volts)',
            'chisq' : data['chisq'],
            'fit-label': r"$I' = I_0 \sin^2 (2\theta+\phi)$",
            'data-label': "Raw data",
            'save-name' : f'{plate_name}',
            'loc' : 'upper right'}

    max_I0 = np.max(data['ploty'])
    idx_maxI0 = np.where(data['ploty'] == max_I0)
    max_theta = data['plotx'][idx_maxI0]
    print(f'theta+phi max = {max_theta}')

    tk.quick_plot_residuals(xdata, ydata, data['plotx'], data['ploty'],
                            data['residuals'], meta=meta,
                            uncertainty=y_unc, uncertainty_x=x_unc,
                            save=True)


    return data['chisq'], data['popt'], data['pstd']

if __name__ == '__main__':

    # Part (a)
    df = load_data('part2a.csv')
    params = fit_plot(df, 'QWP')

    printvals = [r'$\chi_{red}^2'+f' = {params[0]}$']
    for i,v in enumerate([r'I_0', r'\phi', ]):
        printvals.append(f'${v} = {params[1][i]} \pm {params[2][i]}$')

    tk.block_print(printvals, 'Fit Paramters for QWP')

    # Part (b)
    df = load_data('part2b.csv')
    params = fit_plot(df, 'HWP')

    printvals = [r'$\chi_{red}^2'+f' = {params[0]}$']
    for i,v in enumerate([r'I_0', r'\phi']):
        printvals.append(f'${v} = {params[1][i]} \pm {params[2][i]}$')

    tk.block_print(printvals, 'Fit Paramters for HWP')
```

```
# -*- coding: utf-8 -*-
"""
Created on Tue Jan 23 12:34:34 2024
Updated on Mon Oct 14 21:22:09 2024
Updated on Mon Feb 10 09:51:03 2025

Lab Toolkit

```

```
@author: Aditya K. Rao
@github: @adirao-projects
"""
import numpy as np
from scipy.optimize import curve_fit
import matplotlib.pyplot as plt
import matplotlib.gridspec as gridspec
import os
import math
import textwrap

#from uncertainties import ufloat

font = {’family’ : ’DejaVu Sans’,
        ’size’   : 30}

plt.rc(’font’, **font)

def curve_fit_data(xdata, ydata, fit_type, override=False,
                   override_params=(None,), uncertainty=None,
                   res=False, chi=False, uncertainty_x=None,
                   model_function_custom=None, guess=None):

    def chi_sq_red(measured_data:list[float], expected_data:list[float],
               uncertainty:list[float], v: int):
        if type(uncertainty)==float:
            uncertainty = [uncertainty]*len(measured_data)
        chi_sq = 0

        # Converting summation in equation into a for loop
        for i in range(0, len(measured_data)):
            chi_sq += (pow((measured_data[i] \
                - expected_data[i]),2)/(uncertainty[i]**2))

        chi_sq = (1/v)*chi_sq

        return chi_sq


    def residual_calculation(y_data: list, exp_y_data) -> list[float]:
        residuals = []
        for v, u in zip(y_data, exp_y_data):
            residuals.append(u-v)

        return residuals

    def model_function_linear_int(x, m, c):
        return m*x+c

    def model_function_exp(x, a, b, c):
        return a*np.exp**(b*x)

    def model_function_log(x, a, b):
        return b*np.log(x+a)
```

```
    def model_function_linear_int_mod(x, m, c):
        return m*(x+c)

    def model_function_linear(x, m):
        return m*x

    def model_function_xlnx(x, a, b, c):
        return b*x*(np.log(x)) + c

    def model_function_ln(x, a, b, c):
        return b*(np.log(x)) + c

    def model_function_sqrt(x, a):
        return a*np.sqrt(x)

    model_functions = {
        'linear' : model_function_linear,
        'linear-int' : model_function_linear_int,
        'xlnx' : model_function_xlnx,
        'log' : model_function_log,
        'exp' : model_function_exp,
        'custom' : model_function_custom
        }

    try:
        model_func = model_functions[fit_type]

    except:
        raise ValueError(f'Unsupported fit-type: {fit_type}')


    if not override:
        new_xdata = np.linspace(min(xdata), max(xdata), num=100)


        if type(uncertainty) == int:
            abs_sig =True
        else:
            abs_sig = False

        if guess is not None:
            popt, pcov = curve_fit(model_func, xdata, ydata, sigma=uncertainty,
                                   maxfev=20000, absolute_sigma=abs_sig, p0=guess)
        else:
            popt, pcov = curve_fit(model_func, xdata, ydata, sigma=uncertainty,
                                   maxfev=20000, absolute_sigma=abs_sig)
        param_num = len(popt)

        exp_ydata = model_func(xdata,*popt)

        deg_free = len(xdata) - param_num

        new_ydata = model_func(new_xdata, *popt)
```

```

        residuals = None
        chi_sq = None

        if res:
            residuals = residual_calculation(exp_ydata, ydata)

        if chi:
            chi_sq = chi_sq_red(ydata, exp_ydata, uncertainty, deg_free)

        data_output = {
            'popt' : popt,
            'pcov' : pcov,
            'plotx': new_xdata,
            'ploty': new_ydata,
            'chisq' : chi_sq,
            'residuals' : residuals,
            'pstd' : np.sqrt(np.diag(pcov))
            }

        return data_output

    else:
        return model_func(xdata, *override_params)


def quick_plot_residuals(xdata, ydata, plot_x, plot_y,
                         residuals, meta=None, uncertainty=[], save=False,
                         uncertainty_x=[]):
    """
    Relies on the python uncertainties package to function as normal, however,
    this can be overridden by providing a list for the uncertainties.
    """
    fig = plt.figure(figsize=(14,14))
    gs = gridspec.GridSpec(ncols=11, nrows=11, figure=fig)
    main_fig = fig.add_subplot(gs[:6,:])
    res_fig = fig.add_subplot(gs[8:,:])

    main_fig.grid('on')
    res_fig.grid('on')
    if type(uncertainty) is int:
        uncertainty = [uncertainty]*len(xdata)

    elif len(uncertainty) == 0:
        for y in ydata:
            uncertainty.append(y.std_dev)

    if meta is None:
        meta = {'title' : 'INSERT-TITLE',
                'xlabel' : 'INSERT-XLABEL',
                'ylabel' : 'INSERT-YLABEL',
                'chisq' : 0,
                'fit-label': "Best Fit",
                'data-label': "Data",
```

```
                'save-name' : 'IMAGE',
                'loc' : 'lower right'}

    main_fig.set_title(meta['title'], fontsize = 46)
    if len(uncertainty_x)==0:
        main_fig.errorbar(xdata, ydata, yerr=uncertainty, #xerr=uncertainty_x,
                          markersize='4', fmt='o', color='red',
                          label=meta['data-label'], ecolor='black')
    else:
        main_fig.errorbar(xdata, ydata, yerr=uncertainty, xerr=uncertainty_x,
                          markersize='4', fmt='o', color='red',
                          label=meta['data-label'], ecolor='black')

    main_fig.plot(plot_x, plot_y, linestyle='dashed',
                  label=meta['fit-label'])

    main_fig.set_xlabel(meta['xlabel'])
    main_fig.set_ylabel(meta['ylabel'])
    main_fig.legend(loc=meta['loc'])


    res_fig.errorbar(xdata, residuals, markersize='3', color='red', fmt='o',
                     yerr=uncertainty, ecolor='black', alpha=0.7)
    res_fig.axhline(y=0, linestyle='dashed', color='blue')
    res_fig.set_title('Residuals')
    save_name = meta["save-name"]
    plt.savefig(f'figures/{save_name}.png')

def quick_plot_test(xdata, ydata, plot_x = [], plot_y = [],
                    uncertainty=[]):
    plt.figure(figsize=((14,10)))

    plt.title("Test Plot for data")
    plt.xlabel("X Data")
    plt.ylabel("Y Data")

    if len(uncertainty) != 0:
        plt.errorbar(xdata, ydata, yerr=uncertainty, fmt='o')
    else:
        plt.scatter(xdata, ydata)

    plt.grid("on")
    plt.show()
    plt.savefig('Test.png')
    plt.close()

def block_print(data: list[str], title: str, delimiter='=') -> None:
    """
    Prints a formated block of text with a title and delimiter

    Parameters
    ----------
    data : list[str]
        Text to be printed (should be input as one block of text).
```

```
    title : str
        Title of the data being output.
    delimiter : str, optional
        Delimiter to be used. The default is '='.

    Returns
    -------
    None.

    Examples
    --------
    >>> r_log = 100114.24998718781
    >>> r_dec = 0.007422298127465114
    >>> data = [f'r^2 value (log): {r_log}',
                f'r^2 value (real): {r_dec}']
    >>> block_print(data, 'Regression Coefficient', '=')
    ============================ Regression Coefficient ============================
    r^2 value (log): 100114.24998718781
    r^2 value (real): 0.007422298127465114
    ================================================================================
    """
    term_size = os.get_terminal_size().columns

    breaks = 1
    str_len = len(title)+2
    while  str_len >= term_size:
        breaks += 1
        str_len = math.ceil(str_len/2)


    str_chunk_len = math.ceil(len(title)/breaks)
    str_chunks = textwrap.wrap(title, str_chunk_len)
    output = ''
    for chunk in str_chunks:
        border = delimiter*(math.floor((term_size - str_chunk_len)/2)-1)
        output = f'{border} {chunk} {border}\n'

    output=output[:-1]

    output+= '\n'+ '\n'.join(data) + '\n'
    output+=delimiter*term_size

    print(output)

def numerical_methods(method_type, args=None, custom_method=None):
    def gaussxw(N):

        # Initial approximation to roots of the Legendre polynomial
        a = np.linspace(3,4*N-1,N)/(4*N+2)
        x = np.cos(np.pi*a+1/(8*N*N*np.tan(a)))

        # Find roots using Newton's method
        epsilon = 1e-15
        delta = 1.0
```

```
        while delta>epsilon:
            p0 = np.ones(N,float)
            p1 = np.copy(x)
            for k in range(1,N):
                p0,p1 = p1,((2*k+1)*x*p1-k*p0)/(k+1)
            dp = (N+1)*(p0-x*p1)/(1-x*x)
            dx = p1/dp
            x -= dx
            delta = max(abs(dx))

        # Calculate the weights
        w = 2*(N+1)*(N+1)/(N*N*(1-x*x)*dp*dp)

        return x, w

    def gaussxwab(N,a,b):
        x,w = gaussxw(N)
        return 0.5*(b-a)*x+0.5*(b+a),0.5*(b-a)*w

    methods = {
    'gausswx' : gaussxw,
    'gaussxwab' : gaussxwab,
    'custom' : custom_method
    }

    try:
        method = methods[method_type]

    except:
        raise ValueError(f'Unsupported method-type: {method_type}')

    return method(*args)


def interpolation_methods(method_type, args=None, custom_method=None):

    methods = {
    'custom' : custom_method
    }

    try:
        method = methods[method_type]

    except:
        raise ValueError(f'Unsupported method-type: {method_type}')

    return method(*args)
```