Examination of Optical Polarisation

2472182G

Aims:

The main aims of this lab were to understand the effects of different types of material on the polarisation of light.

Polarisation - Creation of Linear SOPs, Dichroic Polymer sheet

December 1, 2021

```
[2]: import numpy as np
     import matplotlib.pyplot as plt
     from scipy import optimize
     from scipy.stats import sem
     import statistics

     plt.style.use('untitled1.txt')
```

1 Objective

The objective of this first part was to unerstand Malus' law in the context of the experiment.

2 Experimental method

2.1 Experimental set-up

The original apparatus, without any of the polarising plates, is a laser aligned with a detector (I just checked that the 2 were indeed aligned by covering the detector with a sheet of paper to block the light from the laser, to check the beam was indeed arriving on the detector. I didn't do more because the demonstrator told me not to because of the age of the equipment).

Here, we want to confirm Malus' law for a linearly polarised beam incident on a linear polariser, which is equation (1) in the lab script

$$P(\theta) = P_0 \cos^2(\theta)$$

where $P(\theta)$ is the power transmitted at angle θ between the transmission axis of the polariser and the orientation of the polarised light and P_0 is the power output when the two are aligned (ie maximum power output).

This means that we will need two polarisers here: - the first one is used to turn the laser beam into linearly polarised light, as a laser beam is not polarised in general. - the second serves as an analyser, ie helps us determine the state of polarisation (SOP) of the laser beam by using Malus' law.

I first adjust the polariser so that it transmits the maximum amount of light from the laser beam, about 950 counts here.

We then want to set up the analyser and polariser such that their transmission axes are aligned at 0°, ie we get a maximum power output at 0°. To do that, I adjusted the scale so that we got minimum transmission when the polariser was at 0° and the analyser at 90°. I did this because the detector fluctuates less at lower powers than at higher powers, so it's easier to know when we have a minimum. I don't find 0 counts for the minimum because of the background from ambient light (I find 5). The reading error on the detector is 1 count, but I don't know the error on the detector. The reading error on the angle is 1°. Note that the polariser and analyser combination absorb and block light, even at maximum transmission, so the light becomes dimmer than when there is only the polariser, or no plates at all.

2.2 Measurements

Background count was 5.

To check Malus' law, we need to measure the power output for a range of analyser angles of a single incident SOP. The best way to do that is to keep the angle of the polariser fixed and rotate the analyser around 180° (as we have a cosine term in the equation, so we don't need any measurements in the 3rd or 4th quadrants). I repeated the measurements 3 times to reduce error, especially that the detector fluctuates a lot for higher power values.

We then get:

Table 1: Output power measured of a polarised beam for different analyser angles

analyser angle (°)	output power 1	output power 2	output power 3
0	959	958	960
20	840	839	832
40	551	555	551
60	236	232	230
80	29	28	30
90	5	4	4
100	38	39	39
120	254	252	251
140	573	571	570
160	846	847	850
180	955	955	955

These were not background subtracted, but we clearly see that there is a minimum at 90° here, as expected, and that is is at 0 (± the reading error) when we subtract the background, which is all as expected.

3 Data Analysis

Now, we want to see if our measurement match up with what we know about Malus' law. To do this, we must therefore plot the background subtracted plot of our measurements. First, we need to read the data.

[3]:
```
#load the data

angle, counts1, counts2, counts3 = np.loadtxt('Measurements_paragraph_2.csv',␣
↪delimiter=',', skiprows=4, unpack = True)
    # first column was angles
    # other columns were all our 3 sets of measurements

#make a note of the errors
background=5                        #background value
angle_err=np.ones_like(angle)   #reading error on angle is one
```

Now, we want to take the average of each line to get the average count value for a given angle. The error is then calculated by using the `scipy` function `stats.sem`.

[4]:
```
#find out the standard error and mean of the measurements

counts_err  = []                                    #create array for standard␣
↪error
counts_mean = []                                    #create array for the mean

for i in range (0,11):
    mea     = [counts1[i],counts2[i],counts3[i]]    #array of the i-th element in␣
↪all the counts arrays
    mea     = mea-np.ones_like(mea)*background      #subtracting array by array␣
↪of background value
    average = statistics.mean(mea)                  #find the average of each␣
↪iteration
    ste     = sem(mea)                              #find the standard error of␣
↪each iteration

    print('the average power at angle {0} is then {1:.1f} pm {2:.1f}'.
↪format(i+1, average, ste))

    counts_mean.append(average)                     #append the calculated␣
↪average value to corresponding array
    counts_err.append(ste)                          #append the calculated␣
↪standard error to corresponding array
```

```
the average power at angle 1 is then 954.0 pm 0.6
the average power at angle 2 is then 832.0 pm 2.5
the average power at angle 3 is then 547.3 pm 1.3
the average power at angle 4 is then 227.7 pm 1.8
the average power at angle 5 is then 24.0 pm 0.6
the average power at angle 6 is then -0.7 pm 0.3
the average power at angle 7 is then 33.7 pm 0.3
the average power at angle 8 is then 247.3 pm 0.9
the average power at angle 9 is then 566.3 pm 0.9
the average power at angle 10 is then 842.7 pm 1.2
the average power at angle 11 is then 950.0 pm 0.0
```

In the lab script, we are actually asked to normalise the measurements before plotting them. To do this, we need to divide each measurement by the maximum count, ie the normalised maximum needs to be equal to 1.

[6]:
```
# Want to normalise data, so divide by counts by maximum:

max_counts  = max(counts_mean)                # naming the maximum number of␣
↪counts
max_index   = counts_mean.index(max_counts)   # finding its index
max_error   = counts_err[max_index]           # naming the error associated with␣
↪the maximum, which has the same index

norm_counts = counts_mean/max_counts          #normalising
```

There's an error in the maximum count, so we need to propagate the error following common error propagation rules. Noting the counts for a given angle as C_i and the maximum counts as M. The error on the normalised value, N_i should be (noting errors as σs)

$$\sigma(N_i) = N_i\sqrt{\left(\frac{\sigma(C_i)}{C_i}\right)^2 + \left(\frac{\sigma(M)}{M}\right)^2}$$

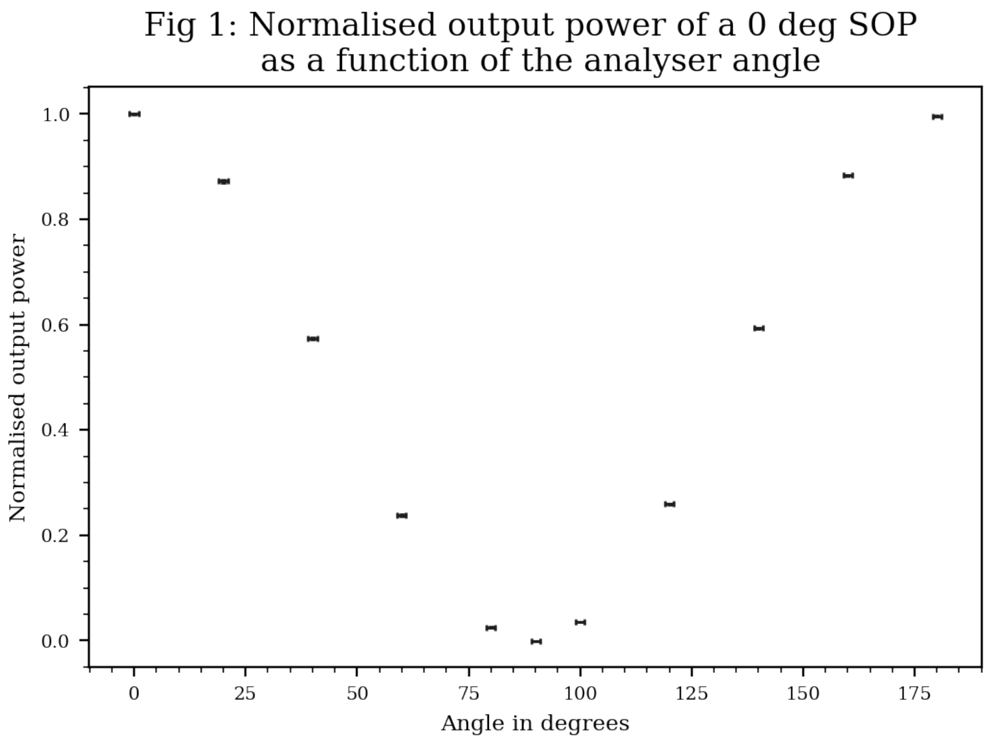
where $N_i = C_i/M$.

In code:

[7]:
```
# calculating error on normalised counts by error propagation

norm_error = [(norm_counts[i] * np.sqrt((counts_err[i]/
↪counts_mean[i])**2+(max_error/max_counts)**2)) for i in range(np.
↪size(counts_mean))]
    # ^ using list comprehension here
```

Now plotting our normalised measurements

[9]:
```
#plotting normalised plot

plt.errorbar(angle, norm_counts, yerr=norm_error, xerr=angle_err, fmt='none')  ␣
↪   #plot number of counts against angles
plt.xlabel('Analyser ngle in degrees')                                         ␣
↪   #label x-axis
plt.ylabel('Normalised output power')                                          ␣
↪   #label y-axis
plt.title('Fig 1: Normalised output power of a 0 deg SOP \n as a function of␣
↪the analyser angle')
plt.show()
```



These errors in Fig 1 are very small, which is nice. This clearly looks like a $\cos^2$ wave of amplitude 1, which is consistent with Malus' law as the measurements were normalised. Now, to verify that this indeed follows Malus' law, we can fit a general $\cos^2$ function:

$$y = A\cos^2(kx + \phi) + C$$

where A is the amplitude, k is a factor multiplying the independent variable x, ϕ is the phase offset and C is the y-offset. In our case, we should expect $A = k = 1$ and $\phi = C = 0$. We used the `scipy` function `optimize.curve_fit` to fit this function to our data.

[11]:
```
#define general cos^2 to fit data

def cos_sq_function(x, A, k, p, c):
    "general cos^2 function of the form y = A*cos^2(k*x + p) + c"
        # A is the amplitude, k is the multiplying coefficient inside the␣
 ↪cosine term, p is the phase offset and c is the y-offset
        # x is the independent variable, y is the dependant variable

    X = np.radians(k*x + p) # we need to convert what's in the brackets into␣
 ↪radians here because of np.cos
                            # p will then be in degrees and k in degrees^-1
    return A*(np.cos(X))**2 + c
```

```
# use scipy.optimize.curve_fit function to fit the cos^2 function to our data
# choice of initial guesses are:
    # A is amplitude so should be 1, k should be 1 and c, p chould be 0

popt, pcov = optimize.curve_fit(cos_sq_function, angle, norm_counts, p0 = [1,␣
↪1, 0, 0], sigma = norm_error)

fitted_A, fitted_k, fitted_p, fitted_c = popt                        #naming␣
↪fitted parameters
err_A,    err_k,    err_p,    err_c    = np.sqrt(np.diagonal(pcov)) #naming␣
↪associated errors

print('fit function is then y = {0:.4f} pm {4:.2g} * cos^2 (({1:.3f} pm {5:.
↪1g}) * x + ({2:.1f} pm {6:.1g})) + ({3:.4f} pm {7:.1g})'.format(fitted_A,␣
↪fitted_k, fitted_p, fitted_c, err_A, err_k, err_p, err_c))


x = np.linspace(0, 180, 1000)                                              ␣
↪   #this is just a dummy variable
plt.errorbar(angle, norm_counts, yerr=norm_error, xerr=angle_err, fmt='none')  ␣
↪   #plot number of counts against angles
plt.plot(x, cos_sq_function(x, fitted_A, fitted_k, fitted_p, fitted_c), label =␣
↪'fitted cos squared \n wave')
plt.xlabel('Angle in degrees')                                             ␣
↪   #label x-axis
plt.ylabel('Normalised counts')                                            ␣
↪   #label y-axis
plt.title('Fig 2: Data fitted to Malus Law')
plt.legend()
plt.show()
```

```
fit function is then y = 0.9971 pm 0.0014 * cos^2 ((0.998 pm 0.002) * x + (0.9
pm 0.2)) + (-0.0000 pm 0.0006)
```

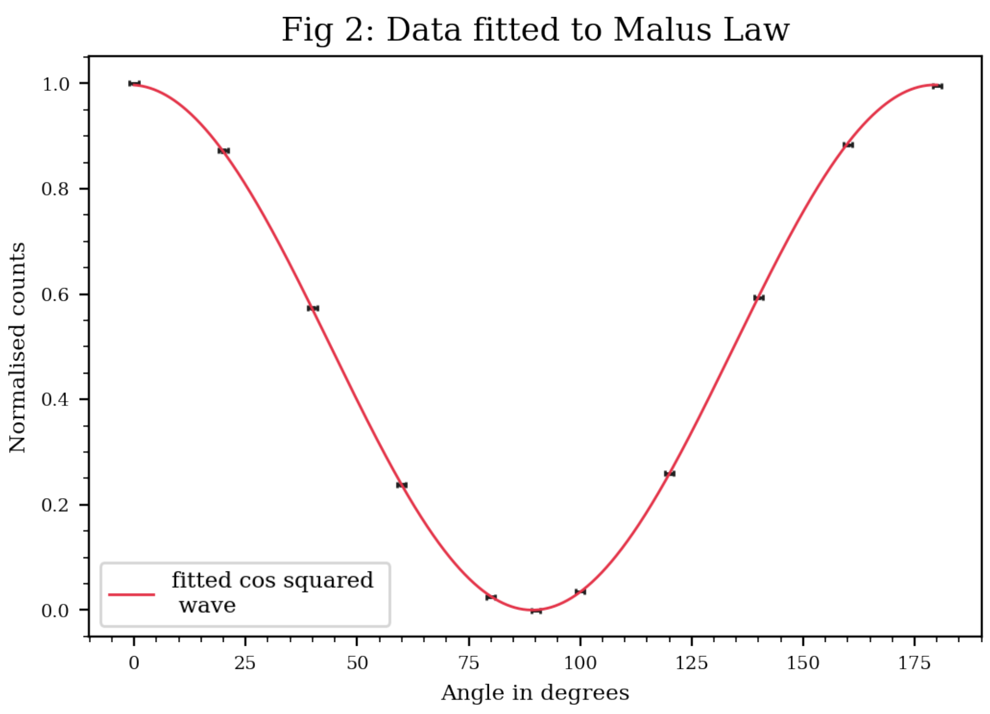


Fig 2 is a pretty good fit. As the phase offset is in *degrees*, it is actually very small compared to the whole period (less than 1%), and so very close to what was expected. The other parameters are also very close to what we would expect.

4 Results

Finally, using the measurements in table 1 and normalising, we were able to find that the power output of a linearly polarised beam crossing a linear polariser can be predicted my Malus' law, as we could fit the function

$$N = (0.9971 \pm 0.0014) \times \cos^2[(0.998 \pm 0.002)\,\theta + (0.9 \pm 0.2)] + (0 \pm 0.0006)$$

to our data with a great accuracy, which is what was expected. Note that θ is in degrees here.

5 Discussion

5.1 Our errors

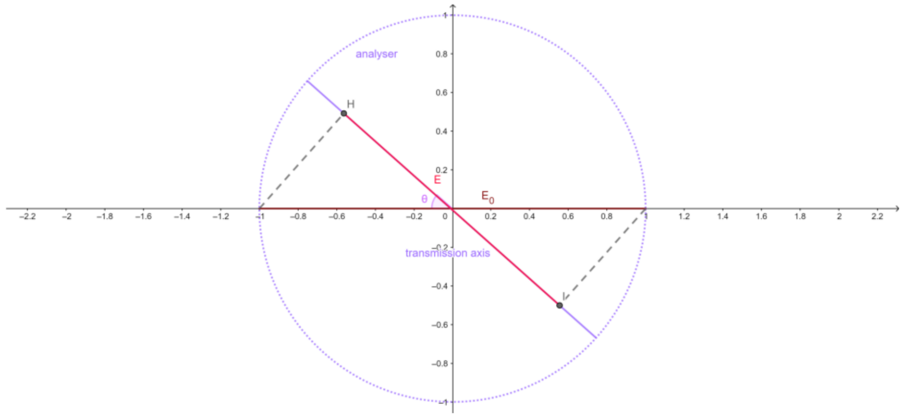
Our errors were very small here. We were able to achieve that by taking measurements multiple times in order reduce random errors. This was especially important as the detector fluctuated widly when measuring higher powers. This could come from multiple sources: * the random error on the number of incoming photons increases with increasing photon number. As the photon rate is constant and they arrive independently from each other, this follows Poisson statistics and the counting error would then be $\sqrt{N}$, where N is the total number of photons arriving at the detector. * the noise caused by the detector may be partially proportional to the measured power output

5.2 Where Malus' law comes from

It's quite straightforward to see where this relation comes from, and is also described in the lab script (on p3). I always find it clearer to visualise things with a diagram, for example figure 3 from

[12]:
```
malus = plt.imread('geogebra-export.png')
plt.imshow(malus)
plt.axis('off')
plt.title('Fig 3: diagram of the effect of a linear polariser \n at an angle␣
↪theta from polarised beam')
plt.show()
```

Fig 3: diagram of the effect of a linear polariser
at an angle theta from polarised beam



Here, we see that the original beam E_0 is projected onto the transmission axis, which means the outgoing beam will be E (between points H and I). This means that

$$E = E_0 \cos(\theta)$$

And as the power output of our detector is proportional to the intensity of the beam, which is itself proportional to the electric field magnitude squared E or E_0 (depending on which intensity we're talking about), we then again find Malus' law, as stated earlier.

Creation of Ordered SOPs - Birefringence

December 1, 2021

[5]:
```
#necessary imports
import numpy as np
import matplotlib.pyplot as plt
from scipy import optimize
from scipy import stats
import statistics

plt.style.use('untitled1.txt')
```

1 Objective

In this part, we are studying the effect of a half-wave plate (1/2 WP) on a linearly polarised beam.

2 Experimental method

2.1 Experimental set-up

We use the same apparatus, adding the half waveplate (1/2 WP) between the polariser and the analyser, such that we have linearly polarised light incident on the 1/2 WP, and we can study the SOP which comes out of the 1/2 WP by rotating the analyser. Note that this 1/2 WP only works for our laser (633 nm) because of how it is constructed to introduce a phase offset of π between the slow and fast axes.

We first want to align the 1/2 WP so that the transmission axes of the 3 components are aligned, which we did by finding the maximum transmission when the analyser and the polariser are both at 0°, and changing the scale of the 1/2 WP so that this corresponds to 0° for the analyser too. This is different from the method used previously (where we used the minimum transmission to set up), because the minimum transmission is 45° for the 1/2 WP, which falls between graduations, making the calibration less precise. I'll perhaps find a better method later on.

2.2 Measurements

The Background was 5 counts for this part.

We first want to find the two transmission axes of the 1/2 WP, so we fix the polariser and the analyser ar 0°, and rotate the 1/2 WP to find the minimum and maximum transmission angles.

max transmission at 0° (914 counts), 90° (902 counts), 180° (906 counts), 270° (916 counts)

min transmission at 45° (4 counts), 135° (5 counts), 225° (5 counts), 315° (4 counts).

These were not background subtracted.

We deduce that the two transmission axes are at 90° from each other. Now if we average the power out in the first axis (0 and 180°), and also calculate the standard error on the mean using the scipy function `scipy.stats.sem`, and do the same for the second axis, we find

[6]:
```
#calculating the power averages for each axes
one_av  = np.average([914-5, 906-5]) #first axis, background subtracted
two_av  = np.average([902-5, 916-5]) #second axis, background subtracted

#calculating the standard error on the mean for these
one_ave = stats.sem([914-5, 906-5])  #first axis
two_ave = stats.sem([902-5, 916-5])  #second axis

print('the average power output in the first axis is {0:.0f} pm {1:.0f} counts'.
 ↪format(one_av, one_ave))
print('the average power output in the second axis is {0:.0f} pm {1:.0f}␣
 ↪counts'.format(two_av, two_ave))
```

```
the average power output in the first axis is 905 pm 4 counts
the average power output in the second axis is 904 pm 7 counts
```

Clearly here, the ratio between the two maximum output powers is one, especially when we take into account the errors. This means the two axes transmit the same amout of light.

Now, to find the effect of the 1/2 WP on the initial SOP, we find the minimum and maximum transmissions for a range of 1/2 WP angles by rotating the analyser around 360°.

Note these are not background subtracted.

Table 1: Measurements of the max and min transmission for different 1/2 WP angles

1/2 WP angle (°)	analyser angles for max transmission (°)	max power in the same order (counts)	analyser angles for min transmission (°)	min power in the same order (counts)
0	0, 180	915, 914	90, 270	5, 5
10	20, 200	914, 914	110, 290	6, 5
30	60, 240	914, 913	150, 330	4, 5
45	90, 270	910, 916	0, 180	5, 6
60	120, 300	905, 912	30, 210	5, 6
80	160, 340	904, 912	70, 250	5, 6
90	0, 180	910, 904	90, 270	5, 7

The angle of the analyser when the transmission is maximal is the angle of the SOP incident on the analyser (which is the whole point of the analyser, and comes from Malus' law). As the SOP incident to the 1/2 WP is at 0°, we conclude that the 1/2 WP must make the polarisation rotate by twice its angle (ie the angle between the first transmission axis and the SOP). This is equivalent to saying that the incoming SOP is reflected with respect to the transmission axis. For example, for a 10 ° angle of the 1/2 WP, we get a final SOP of 20 °. Further, from the 1st and last line of the table (1/2 WP angle of 0 and 90 °), we see that, as the two transmission axes have the same power output, we get the same results.

NOTE that the power output is lower at max transmission than the measurements we had earlier because of the absorption of the WP.

Finally, we take measurements for the power output of a range of 1/2 WP angle, setting analyser and polariser at 0° to study the 1/2 WP's relation to Malus' law.

These are also not background subtracted.

Table 2: Power output along the x-axis for a range of 1/2 WP angles

1/2 WP angle (°)	output power 1	output power 2
0	919	918
20	533	530
45	6	6
60	236	241
80	804	806
90	911	906
100	795	793
120	211	216
135	7	6
140	40	44
160	566	561
180	911	911

3 Data analysis

Here, we use the exact same code as we did in the previous section.

First, we read the data:

[7]:
```
#load the data

angle, power1, power2=np.loadtxt('Measurements_paragraph_3.csv', delimiter=',',␣
 ↪skiprows=4, unpack='True')

background=5                        #background value
reading_error=1                     #error on each value of count
angle_err=np.ones_like(angle)       #reading error on angle is one
```

Then we find the average for each angle, as well as the standard error.

[8]:
```
#find out the standard error and mean of the measurements
power_err=[]                                    #create array for standard error
power_mean=[]                                   #create array for the mean
```

```
for i in range (0,12):
    mea=[power1[i],power2[i]]                     #array of the i-th element in␣
 ↪all the counts arrays
    mea=mea-np.ones_like(mea)*background          #subtracting array by array of␣
 ↪background
    average=statistics.mean(mea)                  #find the average of each␣
 ↪iteration
    ste=sem(mea)                                  #find the standard error of each␣
 ↪iteration

    print('the average power at angle {0} is then {1:.1f} pm {2:.1f}'.
 ↪format(i+1, average, ste))

    power_mean.append(average)                    #append the calculated average␣
 ↪value to corresponding array
    power_err.append(ste)                         #append the calculated standard␣
 ↪error to corresponding array
```

```
the average power at angle 1 is then 913.5 pm 0.5
the average power at angle 2 is then 526.5 pm 1.5
the average power at angle 3 is then 1.0 pm 0.0
the average power at angle 4 is then 233.5 pm 2.5
the average power at angle 5 is then 800.0 pm 1.0
the average power at angle 6 is then 903.5 pm 2.5
the average power at angle 7 is then 789.0 pm 1.0
the average power at angle 8 is then 208.5 pm 2.5
the average power at angle 9 is then 1.5 pm 0.5
the average power at angle 10 is then 37.0 pm 2.0
the average power at angle 11 is then 558.5 pm 2.5
the average power at angle 12 is then 906.0 pm 0.0
```

There are some errors here which are equal to 0, which they really shouldn't, which in part comes from the fact that we made a limited number of measurements, and that we have a reading error (ie our measurements can't go beyond a certain precision). We'll see if this skews our results later on.

We then need normalise the data and plot, like so:

```
[10]: # Want to normalise data, so divide by counts by maximum:

max_power  = max(power_mean)                 # naming the maximum number of␣
 ↪counts
max_index   = power_mean.index(max_power)    # finding its index
max_error   = power_err[max_index]           # naming the error associated with␣
 ↪the maximum, which has the same index

norm_power = power_mean/max_power            #normalising
```

```
# calculating error on normalised counts by error propagation

norm_error = [(norm_power[i] * np.sqrt((power_err[i]/
↪power_mean[i])**2+(max_error/max_power)**2)) for i in range(np.
↪size(power_mean))]
    # ^ using list comprehension here


#plotting
plt.errorbar(angle, norm_power, yerr=norm_error, xerr=angle_err, fmt='none')   ␣
↪  #plot number of counts against angles
plt.xlabel('Angle in degrees')                                                  ␣
↪  #label x-axis
plt.ylabel('Normalised Power')                                                  ␣
↪  #label y-axis
plt.title('Fig 1: Normalised output power for different 1/2 WP angles \n with␣
↪the analyser aligned with the x-axis')
plt.show()
```

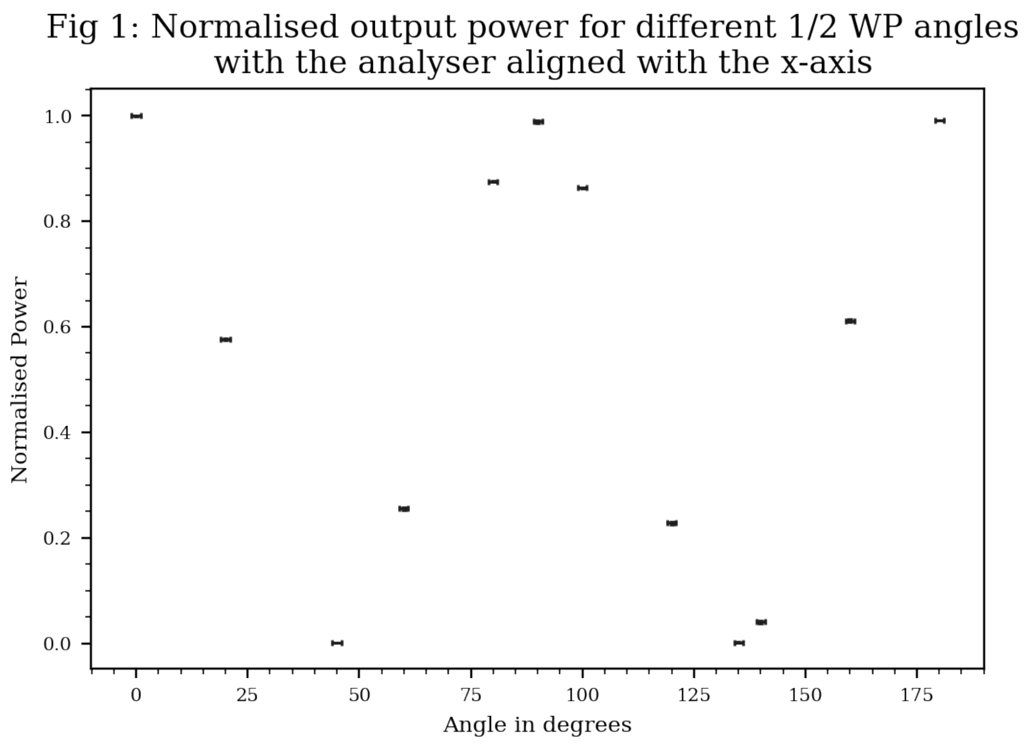
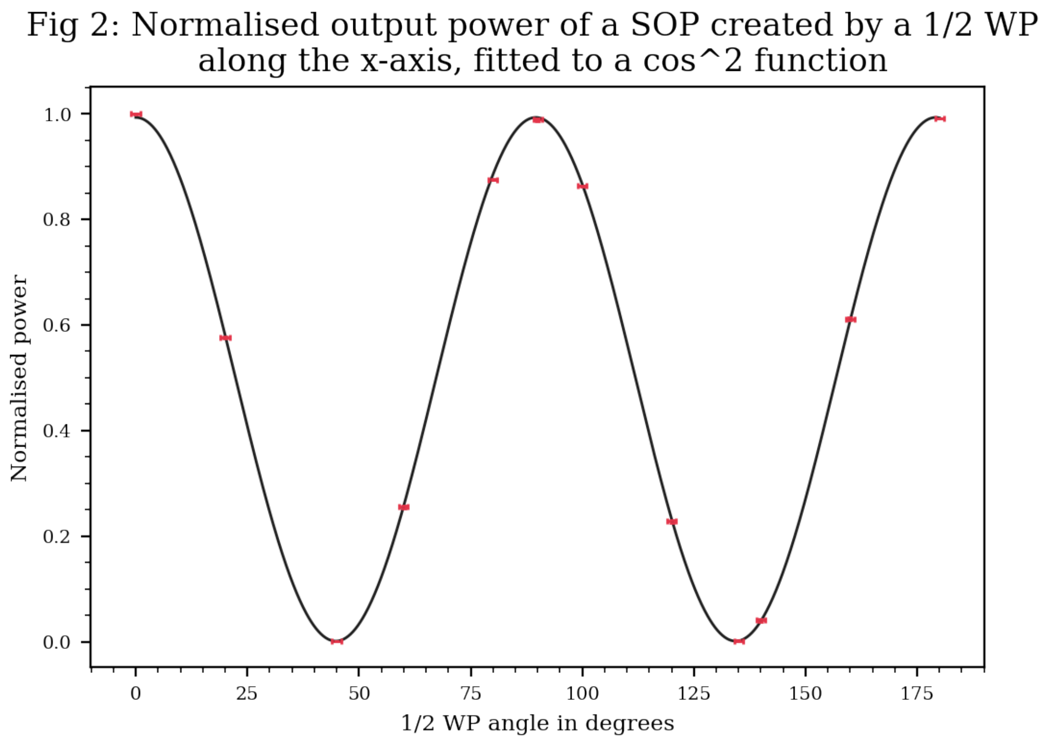


Fig 1 looks like a $\cos^2$ function which has a period of $\pi/2$. We can verify this by fitting a $\cos^2$ function in exactly the same way as the previous section.

```
[14]: #define general cos^2 to fit data

def cos_sq_function(x, A, k, p, c):
    "general cos^2 function of the form y = A*cos^2(k*x + p) + c"
        # A is the amplitude, k is the multiplying coefficient inside the␣
 ↪cosine term, p is the phase offset and c is the y-offset
        # x is the independent variable, y is the dependant variable
    X = np.radians(k*x + p) #we need x in radians here
    return A*(np.cos(X))**2 + c

# use scipy.optimize.curve_fit function to fit the cos^2 function to our data
# choice of initial guesses
    # A is amplitude so should be 1, k should be 1 and c, p chould be 0

popt, pcov = optimize.curve_fit(cos_sq_function, angle, norm_power, p0 = [1, 2,␣
 ↪0, 0], sigma = norm_error)

fitted_A, fitted_k, fitted_p, fitted_c = popt                          #naming␣
 ↪fitted parameters
err_A,    err_k,    err_p,    err_c    = np.sqrt(np.diagonal(pcov)) #naming␣
 ↪associated errors

print('fit function is then y = {0:.3f} pm {4:.1g} * cos^2 (({1:.3f} pm {5:.
 ↪1g}) * x + ({2:.1f} pm {6:.1g})) + ({3:.5f} pm {7:.1g})'.format(fitted_A,␣
 ↪fitted_k, fitted_p, fitted_c, err_A, err_k, err_p, err_c))

x=np.linspace(0, 180, 1000)
plt.plot(x, cos_sq_function(x, fitted_A, fitted_k, fitted_p, fitted_c))
plt.errorbar(angle, norm_power, yerr=norm_error, xerr=angle_err, fmt='none')   ␣
 ↪  #plot number of counts against angles
plt.xlabel('1/2 WP angle in degrees')                                          ␣
 ↪  #label x-axis
plt.ylabel('Normalised power')                                                 ␣
 ↪  #label y-axis
plt.title('Fig 2: Normalised output power of a SOP created by a 1/2 WP \n along␣
 ↪the x-axis, fitted to a cos^2 function')
plt.show()
```

```
fit function is then y = 0.992 pm 0.002 * cos^2 ((2.010 pm 0.005) * x + (-0.1 pm
0.5)) + (0.00105 pm 7e-05)
```



This is a very good fit, and very close to what was expected.

4 Results

Finally, using the measurements in table 2 and normalising, we find the the power output of SOPs leaving a 1/2 WP at different angles and incident on an analyser aligned with the x-axis follows the $\cos^2$ function

$$N = (0.992 \pm 0.002) \times \cos^2[(2.010 \pm 0.005)\,\theta + (-0.1 \pm 0.5)] + (0.00105 \pm 0.00007)$$

to our data with a great accuracy; Note that θ is in degrees here.

These results are consistent with Malus' law: the initial SOP was shifted symetrically by the transmission axis of the 1/2 WP, and so is at twice the angle of the 1/2 WP from the x-axis. This means that it is incident to the transmission axis of the analyser at this angle. This is the angle inside the $\cos^2$ term in Malus' law, and so this is why we get $k = 2$ here.

5 Discussion

1/2 WPs are made out of birefringent material, which means the refractive index changes depending on the orientation of the components of the incoming SOP. This means one component of the light will be travelling faster than the other, creating a phase shift between the two. This is very similar to 1/4 WPs. 1/2 WPs are made such that the phase shift between the two components of the wave (parallel and perpendicular to the transmission axis) is π, ie there is a 1/2 wave difference between the two, which explains the name. The reason why the SOP gets shifted symmetrically comes from the symmetries of cosine, which describes the electric field, where we have $\cos(\theta - \pi) = -\cos(\theta)$.

Theory of Light as EM Waves - Introduction to Ordered States of Polarisation

December 1, 2021

[2]:
```
import numpy as np
from scipy import stats
```

1 Objective

The objective of the following subsections is to study the effect of quarter-wave plates on an initial SOP.

2 Investigation of a 1/4 WP

2.1 Objective

Here, we first want to investigate the properties of the reference 1/4 WP.

2.2 Experimental method

2.2.1 Experimental set-up

Note that all the 1/4 WP used in the following sections are made for the laser we have, ie for 633 nm light.

To study the effect of the reference 1/4 WP on an incoming SOP, we put it between the 1/2 WP and the analyser. This will allow us to study the effect of a 1/4 WP on any SOP created using the 1/2 WP as described in the previous section. We align the REF 1/4 WP such that we get maximum transmission at 0°. There were not many adjustments made as the fast axis was already (nearly) aligned.

2.2.2 Measurements

Background is now 3 counts. It is less than earlier today because it is now dark outside.

We first notice that, when we rotate the 1/4 WP, we get maximal output powers at 0, 90, 180, and 270°. This reveals that the two axes of transmission are at 90°. The one along the x-axis is the fast axis and the one along the y-axis is the slow axis. They are called this way because the light travels faster along the former than along the latter. This is what leads to the phase shift between the x and y component of the electric field in the polarised beam (which is $\pi/2$ in the case of a 1/4 WP).

We first want to know the effect of a 1/4 WP aligned along its fast axis on a 0° SOP. We do this by setting the 1/2 WP to 0° and rotating the analyser.

(These are not background subtracted.)

max transmission at 0° (855 counts), 180° (853 counts)

min transmission at 90° (4 counts), 270° (3 counts)

We see here, that the 1/4 WP seemed to have had no effect on the SOP, as it is still horizontally polarised. Note that some of the light was still absorbed by the waveplate, as the maximum transmission power is diminished.

We now want to know the effect of a 1/4 WP aligned along its slow axis on a 0° SOP (ie the 1/4 WP is rotated by 90 degrees). We do this by setting the 1/2 WP to 0° so that the SOP incident on the 1/4 WP is at 0° to the slow axis, and rotating the analyser.

(These are not background subtracted.)

max transmission at 0° (859 counts), 180° (856 counts)

min transmission at 90° (3 counts), 270° (3 counts)

This means that whether the incident SOP is aligned with the slow or fast axis has no effect on it.

Now we can take the average of the powers, as we already did previously, and find

[3]:
```
#calculating the power averages for each axes
one_av  = np.average([855-3, 853-3]) #first axis, background subtracted
two_av  = np.average([859-3, 856-3]) #second axis, background subtracted

#calculating the standard error on the mean for these
one_ave = stats.sem([855-3, 853-3])  #first axis
two_ave = stats.sem([859-3, 856-3])  #second axis

print('the average power output in the first axis is {0:.0f} pm {1:.0f} counts'.
 ↪format(one_av, one_ave))
print('the average power output in the second axis is {0:.0f} pm {1:.0f}␣
 ↪counts'.format(two_av, two_ave))
```

```
the average power output in the first axis is 851 pm 1 counts
the average power output in the second axis is 854 pm 1 counts
```

We can see the ratio between the two will be very close to one, but not exactly, and their error bounds do not overlap. The two axes transmit a slightly different amount of light, but this is very small, though this could just be because of our error.

3 Creation of 45° SOP

3.1 Objective

Here, we want to investigate the effect of a 45° SOP on our 1/4 WP, and thus devise a way to make this angle very precise.

3.2 Experimental method

The apparatus is the same as in the previous subsection. Here, we want to create circularly polarised light. To do this, we need a 45° linear SOP incident on the 1/4 WP, as outlined in the lab script (The magnitude of the beam will stay the same because the two components E_x and E_y are equal, using simple geometry).

Because the 1/2 WP shifts a SOP aligned with the x-axis by twice the angle of the 1/2 WP, we want it to be at 22.5° The best way to achieve this is to set the angle to about 22°, and then rotate the analyser around 360°, and adjust the 1/2 WP until we get the same power output for every angle of analyser (as we want the light to be circularly polarised). The best I could do was power changing from 428 and 433, which would mean the elliptically polarised light had an eccentricity of about 0.15. A circle has an eccentricity of 0.

This method of adjusting an angle means that it is precise to less than 1 degree, which is far more precise than our previous methods (they were precise to the degree because of our reading error). However, it is probably not suitable for setting every angle as it would mean setting the 1/2 and 1/4 WP such that the angle between the SOP leaving the 1/2 WP and the transmission axis of the 1/4 WP are at 45°, which may require more calculations, and so more time. (unsuitable for quick measurements)

4 Investigation of 2 1/4 WP in series

4.1 Objective

This is an investigation of the effect of 2 1/4 WP in series on a SOP. It also serves as an introduction to Stokes parameters, where we will need to know these effects for $P(45, -90)$ and $P(-45, -90)$

4.2 Experimental method

4.2.1 Experimental set-up

We add a second 1/4 WP to our apparatus between the analyser and the REF 1/4 WP. They must be aligned along their fast axis at 0°, which should already have been the case, not counting a few tweaks. However, the two quarter wave plates were somehow not aligned along the same axis when at 0° (the slow axis of one was aligned with the fast axis of the other), contrary to what was in the lab script, which created great confusion with the demonstrator, so they had to be realigned.

4.2.2 Measurements

Background is 3.

We first want to see the effect of two 1/4 WP in series on an incident linear SOP with their fast axes aligned. To do this, we create SOPs at different angles using the 1/2 WP. We then get:

Table 1: minimum and maximum power output for aligned 1/4 WP.

1/2 WP angle (°)	angle of the SOP incident on the first 1/4 WP (°)	analyser angles for max transmission (°)	max power in the same order (counts)	analyser angles for min transmission (°)	min power in the same order (counts)
22.5	45	135, 315	818, 815	45, 225	3, 3
30	60	120, 300	813, 814	30, 210	2, 3
45	90	90, 270	802, 801	0, 180	3, 2
60	120	60, 240	807, 801	150, 330	3, 3
90	180	0, 180	809, 808	90, 270	3, 3
0	0	0, 180	819, 812	90, 270	3, 3

Note that these are not background subtracted, and that the 22.5° angle was obtained using the same method as described above.

We see here that the two 1/4 WPs in series have the effect of one 1/2 WP, ie the initial and final SOPs are mirror images of eachother: the initial SOP was flipped along the x-axis (as, in general, $-\theta = 360 - \theta$) . This is because each 1/4 WP contributes to a phase shift of $\pi/2$ on either E_x or E_y, which makes a total phase shift of $/pi$, identical to a 1/2 WP.

Now, we take the exact same measurements, but we shift one of the 1/4 WPs by 90° so that the slow axis of one is aligned with the fast axis of the other.

Table 1: minimum and maximum power output for 1/4 WPs at 90°.

1/2 WP angle (°)	angle of the SOP incident on the first 1/4 WP (°)	analyser angles for max transmission (°)	max power in the same order (counts)	analyser angles for min transmission (°)	min power in the same order (counts)
22.5	45	45, 225	829, 831	135, 315	3, 3
30	60	60, 240	828, 826	150, 330	3, 3
45	90	90, 270	825, 823	0, 180	4, 4
60	120	120, 300	820, 824	30, 210	3, 3
90	180	0, 180	825, 826	90, 270	3, 3
0	0	0, 180	840, 835	90, 270	3, 3

Here, we see that when the slow axis of one 1/4 WP is aligned with the fast axis of the other, they cancel out, and so we the SOP of the beam remains the same. This is because one of the WPs shifts E_x by $\pi/2$ while the other shifts E_y by the same amount, meaning there is no relative phase shift between the 2 components of electric field.

NOTE seems to be an assymetry in the power maxima. perhaps 1/4 WP has different power outputs depending on its axis.

4.3 Discussion

1/4 WPs are made out of birefringent material, which means the refractive index changes depending on the orientation of the components of the incoming SOP. This means one component of the light will be travelling faster than the other, creating a phase shift between the two. 1/4 WPs are made such that the phase shift between the two waves is $\pi/2$, ie there is a 1/4 wave difference between the two, which explains the name. The reason why light at 45° to a 1/4 WP becomes circularly polarised is because one of the components will now have the form of a sine wave, while the other will still be a cosine wave (as $\cos(\theta - \pi/2) = \sin(\theta)$). This means the electric field of the beam will basically draw the unit circle if the amplitude is normalised (which comes back to its definition), and so the SOP is circular.

Stokes' parameters

December 1, 2021

[19]:
```
import numpy as np
from scipy import stats
```

1 Objective

In this part, we want to investigate the effects of birefringence, ie when the SOP is affected differently depending on its path. The first two subsections will discuss how these characteristics can be determined by Stokes' parameters, while the last part will discuss strain birefringence.

2 Introduction to Stokes' Parameters

2.1 Objective

Here, we want to verify the results we get using Stoke's parameters using the know known 1/4 WP.

2.2 Experimental Method

2.2.1 Experimental set-up

We want to study the effect of a 1/4 WP on an incoming SOP. To do this, we want to calculate Stokes' parameters associated with the outgoing beam using the power outputs $P(0,0)$, $P(0,0)$, $P(90,0)$, $P(45,0)$, $P(-45,0)$, $P(-45,-90)$ where ψ in $P(\psi,\epsilon)$ is the angle of the analyser, and ϵ is the phase shift of the E_y coponent of the SOP after leaving the 1/4 WP.

For $\epsilon = 0$: we need the polarise, the 1/2 WP to set the initial angle of the SOP, the 1/4 WP under study and the analyser.

For $\epsilon = -90°$, we have to add another 1/4 WP with its fast axis aligned with the x-axis, which comes back to the properties of the 1/4 WP.

2.2.2 Measurements

Had background of 2

We chose SOPs of -45°, 45°, -60° and 20° to the **x-axis** (see later for interpretation with regards to angle to 1/4 WP).

To set the -45 and 45 ° SOP, I use the same method as in paragraph 4. I'll get the best precision - Creation of left circularly polarised light for -45 deg angle: best I could do was 389 to 405 power output - Creation of right circularly polarised light for 45 deg angle: best I could do was 388 to 411 power output

We then get, with the angle being up from the x-axis:

Table 1: P(ψ, ϵ) for 1/4 WP for different polarisations with respect to x-axis

P(ψ, ϵ)	output for 45°	output for -45°	output for 20°	output for -60°
P(0,0)	397, 402, 402	399, 400, 400	724, 725, 726	198, 204, 208
P(90,0)	397, 403, 403	399, 399, 399	97, 99, 100	599, 593, 591
P(45,0)	405, 415, 416	390, 390, 390	423, 420, 420	385, 386, 386
P(-45,0)	391, 393, 394	412, 412, 413	393, 396, 396	416, 416, 416
P(45,-90)	3, 2, 2	770, 771, 771	150, 150, 149	720, 722, 727
P(-45,-90)	766, 771, 768	0, 2, 2	630, 627, 628	58, 53, 51

There are three values because I made multiple measurements to reduce errors.

2.3 Data analysis

We first want to calculate Stokes' parameters, ie equations (9)-(12) in the lab script

$$
\begin{aligned}
S_0 &= P(0,0) + P(90,0)\\
S_1 &= P(0,0) - P(90,0)\\
S_2 &= P(45,0) - P(-45,0)\\
S_3 &= P(45,-90) - P(-45,-90)
\end{aligned}
$$

To test our code, we chose the 1st measurements for 45° in Table 1.

[20]:
```
#calculate Stoke's parameters, where the values in brackets are (measurement -␣
↪background)
S0 = (397-2) + (397-2)
S1 = (397-2) - (397-2)
S2 = (405-2) - (391-2)
S3 = (3-2)   - (766-2)
```

Now we want to normalise these results by dividing by S_0 to get the corresponding normalised values s_0, s_1, s_2 and s_3 such that $s_0 = 1$. This will mean the Poincaré sphere has a radius 1 (every point on the surface of the sphere represent a unique SOP, and is therefore a useful geometric visualisation, see figure 12 in the lab script).

[21]:
```
s0, s1, s2, s3 = [S0, S1, S2, S3]*np.ones_like(4)*(1/S0) #divide array of␣
↪Stokes parameters by S0 to normalise

print('s0 = {0:.2g}, s1 = {1:.2g}, s2 = {2:.2g} and s3 = {3:.2g}'.format(s0,␣
↪s1, s2, s3))
```

```
s0 = 1, s1 = 0, s2 = 0.018 and s3 = -0.97
```

For a 45° angle, I should have found $s_0 = 1 \approx s_3$ and $s_1 \approx s_2 \approx 0$, which would be right circularly polarised light. However, here $s_0 = 1 \approx -s_3$ and $s_1 \approx s_2 \approx 0$, which are the expected values for left circularly polarised light. This is because the angle of the SOP with respect to the transmission axis of the 1/4 WP was in fact **-45°**. This is easily seen visually, but I don't have time to draw a diagram. This also means we have to take the opposites of all the angles in table 1.

This also means, as the SOP is at the south pole of the poincare sphere, that $\delta = -90°$ and $\chi = 45°$. The angle ψ is a bit more complicated to predict, as it should be undefined for perfect values of s_1 and s_2 (=0), so we'll see. As this should be circularly polarised light, we should also find $a = b$.

From the Poincare sphere in figure 12, we see that

$$\tan(2\psi) = s_2/s_1$$

and

$$\sin(2\chi) = s_3/s_0$$

We then need to take the inverse functions and divide by two to find ψ and χ.

We also know that

$$\tan(\delta) = s_3/s_2$$

which was derived in the lab script.

For the tan function, we need to use the `np.arctan2()` to make sure our angle is in the right quadrant. The fact that arcsin is multivalued does not matter here because $\chi \in [-\pi/2, \pi/2]$ (which can be seen by looking at the Poincare sphere and knowledge of how spherical polar coordinates work).

[22]:
```
psi   = np.arctan2(s2, s1)/2
chi   = np.arcsin(s3/s0) /2
delta = np.arctan2(s3, s2)

print('We then have psi = {0:.2g} deg, chi = {1:.2g} deg and delta = {2:.2g}␣
↪deg'.format(psi*180/np.pi, chi*180/np.pi, delta*180/np.pi)) #converting to␣
↪degrees
```

```
We then have psi = 45 deg, chi = -37 deg and delta = -89 deg
```

δ is very close to what we expected (1% difference). χ is further away than what was expected (by 17 %), which will mean the value for b/a will also be. There is a big error here because the gradient of the arcsin(x) function is very steep at 1, and so it is very sensitive to small deviations (the difference between the expected value of -1 for s_3/s_0 and the measured value of -0.97 is only 3%). ψ is equal to 45 ° here because $s_1 = 0$ exactly.

Further, from equation (7) in the lab script,

$$E_{0x}^2 + E_{0y}^2 = a^2 + b^2$$

and χ is defined by

$$\tan(\chi) = b/a$$

As $E_{0x}^2 + E_{0y}^2 = 2s_0$ (using normalised values) (eq 13), we finally have

$$a = \sqrt{\frac{2s_0}{1+\tan^2(\chi)}}$$
$$b = a\tan(\chi)$$

We can replace s_0 by 1, as it is defined like that by the normalisation. It doesn't really matter that a and b are not in real count values because what we're really interested in is their ratio.

[23]:
```
a = np.sqrt(abs((2*1)/(1+(np.tan(chi))**2)))
b = abs(a*np.tan(chi))

print('We then have that a = {0:.2g} and b = {1:.2g}, so b/a = {2:.2g}'.
 ↪format(a,b, b/a))
```

```
We then have that a = 1.1 and b = 0.86, so b/a = 0.77
```

This is quite far from what was expected, but logical, as χ, the only angle on which a and b are dependent, has the most error.

We now want to do the same thing for all the values found for this first angle (the first line in table 1), and average the values we find.

To avoid coppying and pasting for each measurement set, we can first define a series of functions to calculate Stokes' parameters, the angles on the Poincare sphere and the axes of the SOP:

[24]:
```
def Stokes_parameters(P0_0, P90_0, P45_0, Pm45_0, P45_m90, Pm45_m90):
    "Inputs are powers needed to calculate Stokes parameters, output is Stokes␣
 ↪parameters"
        #P0_0 is P(0,0), P90_0 is P(90, 0), P45_0 is P(45, 0), Pm45_0 is␣
 ↪P(-45, 0), P45_m90 is P(45, -90) and Pm45_m90 is P(-45, -90)
    S0 = P0_0    + P90_0
    S1 = P0_0    - P90_0
    S2 = P45_0   - Pm45_0
    S3 = P45_m90 - Pm45_m90
    return(S0, S1, S2, S3)

def angles_from_stokes(S0, S1, S2, S3):
    "normalises stokes parameters and calculates angles psi and chi on the␣
 ↪Poincare sphere, and angle delta"

    #normalising stokes parameters
    s0, s1, s2, s3 = [S0, S1, S2, S3]*np.ones_like(4)*(1/S0)

    #angle calculation
    psi   = np.arctan2(s2, s1)/2
    chi   = np.arcsin(s3/s0) /2
    delta = np.arctan2(s3, s2)

    return(psi, chi, delta)

def SOP_axes(chi):
    "calculates the semi-major and semi-minor axes, a and b, of the SOP from␣
 ↪angle chi on the Poincare sphere"
     #chi is in RADIANS
    a = np.sqrt(abs((2*1)/(1+(np.tan(chi))**2))) #as s0 = 1
    b = abs(a*np.tan(chi))
    return(a, b)
```

We can then calculate δ, ϕ, χ, a and b for our 45° initial SOP (-45° incidence to 1/4 WP) by looping over our set of measurements, using the above definitions. We can then take the average and calculate the standard error on the mean.

The lab script also asked for a/b and E_x/E_y. The latter will be the square root of $P(0,0)/P(90,0)$.

The error on a/b is easily calculated by error propagation:

$$\sigma(b/a) = \frac{b}{a}\sqrt{\left(\frac{\sigma(a)}{a}\right)^2 + \left(\frac{\sigma(b)}{b}\right)^2}$$

The error on E_x/E_y is a bit more complicated, but still doable:

$$\sigma(E_x/E_y) = \frac{1}{2}\left(\frac{E_x}{E_y}\right)^{-1/2}\sigma(P(0,0)/P(90,0))$$

as $E_x/E_y = [P(0,0)/P(90,0)]^{1/2}$.

The errors $\sigma(a)$, $\sigma(b)$ can $\sigma(P(0,0)/P(90,0))$ be calculated by taking the standard error on the mean for each of these values.

[25]:
```
background = 2

#load data for 45 deg angle
P45 = np.loadtxt('measurements 5.2.1 - 45 deg.tsv', skiprows = 3, usecols =␣
 ↪(1,2,3))

#subtract background
P45b = P45 - background

#Stokes parameters for 45 deg angle
S0, S1, S2, S3 = Stokes_parameters(P45b[0], P45b[1], P45b[2], P45b[3], P45b[4],␣
 ↪P45b[5])
```

```
print('We have that the array values are: S0 = {0}, S1 = {1}, S2 = {2}, S3 =␣
 ↪{3}'.format(S0, S1, S2, S3))

#calculate angles
psi, chi, delta = angles_from_stokes(S0, S1, S2, S3)

print('We have that the array values are: psi = {0}, chi = {1}, delta = {2}␣
 ↪rad'.format(psi, chi, delta))

#calculate semi-major and semi-minor axes
a, b = SOP_axes(chi)

print('We have that the array values are: a = {0}, b = {1}'.format(a, b))

#calculating averages and coverting angles in degrees

psi_av   = np.average(psi)*(180/np.pi)
chi_av   = np.average(chi)*(180/np.pi)
delta_av = np.average(delta)*(180/np.pi)
a_av     = np.average(a)
b_av     = np.average(b)

#corresponding standard errors

psi_e   = stats.sem(psi)
chi_e   = stats.sem(chi)
delta_e = stats.sem(delta)
a_e     = stats.sem(a)
b_e     = stats.sem(b)

print('averaging these values, we then get that:')
print('psi = {0:.3f} pm {1:.3f} deg'.format(psi_av, psi_e))
print('chi = {0:.3f} pm {1:.3f} deg'.format(chi_av, chi_e))
print('delta = {0:.3f} pm {1:.3f} deg'.format(delta_av, delta_e))
print('a = {0:.3f} pm {1:.3f}'.format(a_av, a_e))
print('b = {0:.3f} pm {1:.3f}'.format(b_av, b_e))


#also wanted b/a

b_a  = b_av/a_av

#error from error propagation will be

b_ae = b_a * np.sqrt((a_e/a_av)**2+(b_e/b_av)**2)

print()
```

```python
print('This means b/a = {0:.3f} pm {1:.3f}'.format(b_a, b_ae))
print('or calculating with the value of chi')
print('b/a = tan(chi) = {0:.3f}'.format(abs(np.tan((chi_av*np.pi/180)))))

#and finally want E_x/E_y

Pxy_av = np.average(P45b[0]/P45b[1])    #first calculate the average power␣
 ↪output between x and y axis
Pxy_e  = stats.sem(P45b[0]/P45b[1])     #calculate error on the average

E_xy   = np.sqrt(Pxy_av)               #calculate electric field component
E_xye  = 0.5*E_xy**(-0.5)*Pxy_e        #calculate error by error propagation

print()
print('We also have Ex/Ey = {0:.4f} pm {1:.4f}'.format(E_xy, E_xye))

#PRINT AND CHECK
```

```
We have that the array values are: S0 = [790. 801. 801.], S1 = [ 0. -1. -1.], S2
= [14. 22. 22.], S3 = [-763. -769. -766.]
We have that the array values are: psi = [0.78539816 0.8081098  0.8081098 ], chi
= [-0.65429956 -0.64359031 -0.63704494], delta = [-1.55244976 -1.54219555
-1.54208359] rad
We have that the array values are: a = [1.12214228 1.13129516 1.13682547], b =
[0.86069548 0.84862905 0.84120619]
averaging these values, we then get that:
psi = 45.868 pm 0.008 deg
chi = -36.955 pm 0.005 deg
delta = -88.555 pm 0.003 deg
a = 1.130 pm 0.004
b = 0.850 pm 0.006

This means b/a = 0.752 pm 0.006
or calculating with the value of chi
b/a = tan(chi) = 0.752

We also have Ex/Ey = 0.9992 pm 0.0004
```

We can do the same for all the other angles (ie, doing the same code, which is not here because it would take too much space), and the results we found are tabulated in the Results section.

2.4 Results

Finally, we get for each SOP incident on the 1/4 WP under study:

Table 2: Parameters of the polarised beam of a 1/4 WP depending on the initial incident SOP

SOP incident on 1/4 WP	ψ	χ	δ	a	b	b/a	E_x/E_y
45°	-44.151 ± 0.007	37.748 ± 0.002	91.6628 ± 0.0004	1.118 ± 0.001	0.866 ± 0.002	0.774 ± 0.002	1.0008 ± 0.0004
-45°	45.868 ± 0.008	-36.955 ± 0.005	-88.555 ± 0.003	1.130 ± 0.004	0.850 ± 0.006	0.752 ± 0.006	0.9992 ± 0.0004
-20°	1.188 ± 0.002	-17.866 ± 0.001	-86.892 ± 0.004	1.3460 ± 0.004	0.434 ± 0.001	0.322 ± 0.001	2.74 ± 0.02
60°	-87.7818 ± 0.0002	28.728 ± 0.004	92.597 ± 0.001	1.240 ± 0.003	0.680 ± 0.005	0.548 ± 0.004	0.583 ± 0.004

2.5 Discussion

Firstly, in table 2, all the erors for δ are consistent with the fact that we are studying a 1/4 WP, as their absolute value is similar to 90°. Which is the phase difference between the orthogonal components of the SOP. However, the values calculated for 45 and -45° are much closer to the real value ($\approx$ 2% difference), while the other two are further away ($\approx$ 3% difference). This is not much, but nevertheless indicates that the way we set the $\pm$45 SOPs (using the method in paragraph 4) is more precise that setting the 1/2 WP at half the desired angle, as we can get a precision of less than a degree.

For the 45° angle: ψ should be undefined, as discussed earlier. However, our values are not perfect, and so it gives back an angle. This is ok, because it indicates that $|s_1| \approx |s_2|$ (which should both equal 0 in a perfect situation). χ should equal 45° here, as this is right circularly polarised light, which gives a large difference of 15% with the measured value. The gradient of $\arcsin(x)$ is very steep at 1, its expected value, which means the function is very sensitive to small deviations there. This propagates to our value of a/b, which is quite distant from its expected value of 1 for circularly polarised light (more than 20%). However, $E_x/E_y = 1$, as expected.

For the -45° angle: The same things can be said for this angle, and most of it has already been discussed in the Data Analysis part. Something to note is that $E_x/E_y = 1$, as expected.

For the -20° angle: ψ, the angle of orientation, is very small, which indicates that the long axis is nearly aligned with the x-axis, which is what I would expect for small angles. Further, χ and δ are negative, which indicates that the beam rotates to the left, which is consistent with the fact that this angle is negative. Finally, b/a is quite close to 0.5. This is because we're about halfway between circularly polarised anf linearly polarised light (which would have $b/a = 0$), and so the ratio between the two axes of the ellipse would be 0.5

For the 60° angle: ψ is nearly 90°. This may be because we are approaching a 90° angle, where linearly polarised light *perpendicular* to the x-axis is created. χ and δ are positive, which indicates that the beam rotates to the right, which is consistent with the fact that this angle is positive. b/a is closer to 1 than it is to 0.5, which is consistent with the fact that $60 < 72.5$, where we would expect a value of 0.5.

Finally, all our errors are underestimates. One reason I can find is that, because it takes a long time to adjust the $\pm 45°$ SOP, I didn't change it between the measurements sets, which means I only took care of the random error introduced by the detector signal and the analyser angle when making multiple measurements. Another way of putting it is that there would be a systematic error on the SOP angle leaving the 1/2 WP.

3 Using Stokes' Parameters

3.1 Objective

Here, we want to apply what we learned in the previous section to study an arbitrary waveplate (arb WP).

3.2 Experimental Method

3.2.1 Experimental set-up

In this part, we want to study an arbitrary WP. This means we just need to replace the 1/4 WP we studied in the previous section by this arbitrary plate. This arbitrary plate needs to have it's scale set up such that we have maximum transmission at 0°. This was done in the same way as in the previous sections.

3.2.2 Measurements

Firstly, this WP has two transmission axes, which are parallel to eachother (one at 0°, the other at 90°).

Further, by rotating the analyser, we find that for the first transmission axis (0°) the max and min transmissions are

(These are not background subtracted.)

max transmission at 0° (880 counts), 180° (875 counts)

min transmission at 90° (9 counts), 270° (10 counts)

And on the another lab day, we found

max transmission at 0° (916 counts), 180° (920 counts)

min transmission at 90° (3 counts), 270° (2 counts)

Which confirms that the WP has no effect on 0° polarised light.

Unfortunatley, I was not able to take the same measurements for the 45° angle (ie min and max angle of transmission), so we won't be able to compare what we would have found with the later results using Stokes' parameters. The max would have given the value for a, while the min would have given the value for b. Their ratio would have given the value for χ, as $\tan(\chi) = b/a$. I predict we would have found a worth value, just because we wouldn't have been asked to calculate Stokes' parameters otherwise (or at least it would have been illogical to do so).

Now, we want to measure Stokes' parameters, as we did in the previous section. To do this, we measure $P(0,0)$, $P(0,0)$, $P(90,0)$, $P(45,0)$, $P(-45,0)$, $P(-45,-90)$ for incident SOPs of $\pm 45°$ in the same way as the in the previous section (I take the arbitrary waveplate and replace with 1/4

WP to create circularly polarised light in the same way as described before, then take away 1/4 WP and place arb WP again.). We then find, for SOP angles from x-axis (so minus the value incident on arb WP):

Table 3: P(ψ, ϵ) for arb for different polarisations with respect to x-axis (minus value incident on the plate)

P(ψ, ϵ)	output for 45°	output for -45°
P(0,0)	437, 440, 441	397, 396, 397
P(90,0)	387, 387, 388	434, 435, 432
P(45,0)	270, 270, 270	567, 568, 565
P(-45,0)	564, 564, 565	272, 273, 273
P(45,-90)	766, 773, 766	32, 32, 32
P(-45,-90)	32, 28, 32	770, 771, 771

3.3 Data analysis and results

We use exactly the same code as the previous part for the data analysis, and find:

Table 4: Parameters of the polarised beam of arb WP depending on the initial incident SOP

SOP incident on arb WP	ψ	χ	δ	a	b	b/a	E_x/E_y
45°	48.586 ± 0.002	-31.685 ± 0.001	-68.2968 ± 0.001	1.203 ± 0.001	0.743 ± 0.001	0.617 ± 0.001	0.956 ± 0.001
-45°	-39.991 ± 0.002	31.869 ± 0.005	111.753 ± 0.002	1.201 ± 0.004	0.747 ± 0.006	0.622 ± 0.006	1.065 ± 0.001

3.4 Discussion

Firstly, in table 4, the two δs are at 180° from each other, which is what we would expect for 2 angles which have different signs. This means this arbitrary waveplate introduces a phase difference of 112° or $-68°$ depending on the sign of the incidence angle. This also means that the first SOP is rotating anticlockwise, while the second is rotating clockwise. Further, the values for $|\chi|$ and a/b are the same, which we would expect for angles which are symmetric with respect to the transmission axis.

The errors are all underestimated for the reasons described above.

4 Investigation of strain birefringence

4.1 Objective

Here, we want to see the effect of strain on the birefringence of a material.

4.2 Experimental Method

4.2.1 Experimental set-up

For this last part, we take away all the waveplates to only keep the polariser and analyser. We cross them for minimum transmission (ie polariser at 0 and analyser at 90). We then add the Perspex place and adjust the angle until there's minimum transmission and clamp it.

4.2.2 Measurements and Discussion

We want to see the effect of bending the plate at minimum transmission, 50% transmission and maximum transmission (each of these found by rotating the plate). Note that when I say "without bending", I mean how the plate is naturally, as it is not straight. Without subtracting the background, I find

minimum transmission: 2 counts without bending. Bending, I get values from 2 to 33 counts. It increases as I bend it more.

maximum transmission: 169 counts without bending. Bending, it varies from 130 to 172 counts. Bending further decreases the count number

50% transmission: 80 counts without bending. Bending, it varies from 2 to 100 counts. Bending backwards (towards analyser) decreases the count number, while bending forwards (towards polariser) increases the count.

This firstly shows that the transmission changes depending on how bent the plate is. This means strain on some material (usually isotropic) will cause a change in polarisation to change. In other words, strain will cause double refraction, ie birefringence. Note that this birefringence is only caused by stress/strain, as Perspex is a polymer (https://en.wikipedia.org/wiki/Poly(methyl_methacrylate)), and so shouldn't exhibit birefringence. We see that the change in transmission is a lot bigger when the plate is at 50% transmission (at rest) than when it is at maximal or minimal transmission. I wasn't able to find why. Stress/strain birefringence can also be seen for glass (https://www.vedantu.com/physics/double-refraction) and is used to map out the stress/strain in different parts of an object. It is the basis for the study of photoelasticity (https://en.wikipedia.org/wiki/Photoelasticity) (more on that later).

```
[ ]:
```

Overall Discussion

Birefringence:

As mentioned earlier, birefringence is caused by the fact that light travels at different speeds depending on its orientation, which causes a phase offset between its components. This can be caused by optically anisotropic materials, such as calcite (because of its crystal structure), or because of stress or strain on the material (https://en.wikipedia.org/wiki/Birefringence).

The birefringence of minerals can be used to identify them.
For example, we can identify minerals using a petrographic microscope by investigating the effect of turning the observation plate has on what we see, for example how many times a mineral because dimmer or turns black, or how the colour changes. It's breath-taking when you see it for the first time. (source: 6th form Geology lesson).

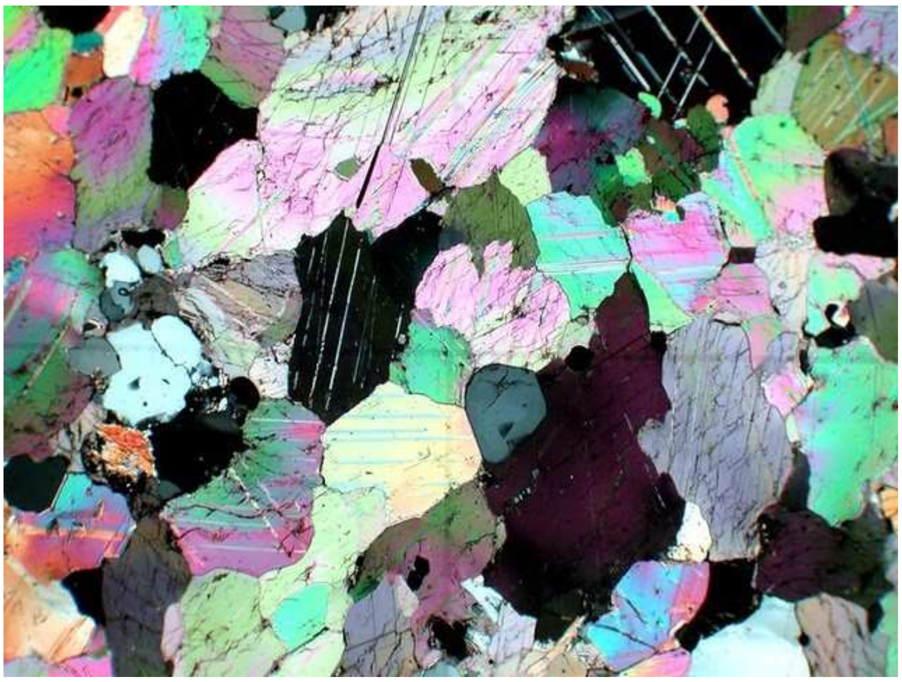


Figure 1: Calcite crystal as seen under a polarizing microscope. Field of view ~ 3 mm. Photo by stef_climber on flikr.com

Aside on stress birefringence:
The fact that stress leads to birefringence means we can identify materials which are or have been under stress, and perhaps identify where the material may break and repair or replace it. It can also be used to map out the stress on a material to a great accuracy. (https://en.wikipedia.org/wiki/Photoelasticity)

Polarisation in everyday life:
Polarisation can help us identify the source of emission in space, for example pulsars and active regions emit highly polarised light. The scattered light of the sky is also partially polarised, and some animals have adapted to detect this polarisation and use it to see and navigate, as it is perpendicular to the direction of the sun. This aspect also helps photographs dim a bright sky while taking pictures by adding a polarising lens and is why many sunglasses are polarisers. I also know that chiral molecules can partially circularise light either clockwise or anticlockwise depending on the molecule's handedness. This must have applications for biology and medicine because of the importance of chiral molecules on organic chemistry. (from https://en.wikipedia.org/wiki/Polarization_(waves))

Conclusion:

In this lab, we learned about the effect of different materials (dichroic, birefringent, stress/strain birefringent) materials on beams of light. Specifically, we learned how to arrange apparatus in order to study a SOP, using Malus' law, Stokes parameters and other geometrical approaches. We therefore understood how to predict the behaviour of light as EM wave.

More generally, we learned how to take measurements that are more precise than our reading error using the properties of ½ and ¼ WPs.