Speed of Light

In order to calculate the speed of anything you need to measure the distance it covers over an interval of time. For light this is a little tricky because it's very fast. L Foucault's design utilizes the optical property of a lens and a rotating mirror in order to precisely measure displacement and time. As illustrated below, the red beam represents the original laser beam.

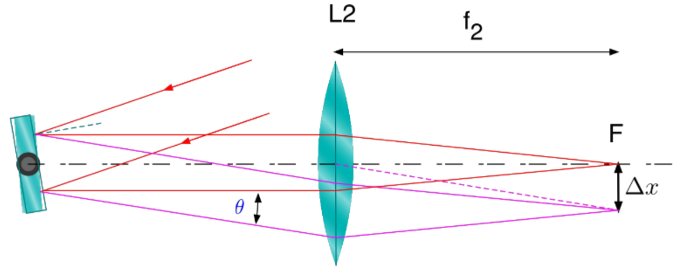


Fig. 1

The magenta beam represents the beam that traveled a longer path, because the speed of light is finite these paths differ due to the rotation of the mirror. The beam splitter is set-up at the focal point of the lens so that it can form an image at the "measuring microscope". We can see in fig.1 that, $\Delta x = f_2\theta$ Now to further develop our relation of Eq.1 to the value of c, we need..

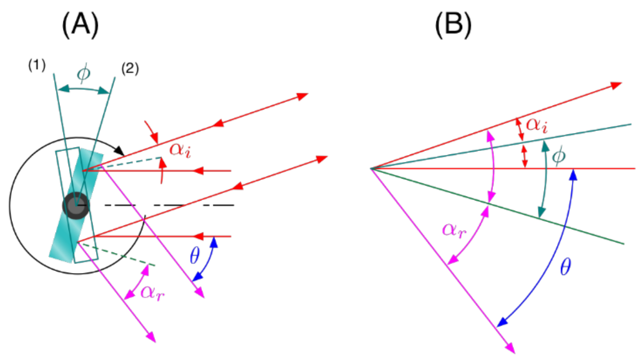


Fig. 2

We find,

$$\theta = 2a_r - 2a_i \qquad a_r = a_i + \phi \qquad \theta = 2\phi \qquad \phi = \omega\Delta t \qquad \Delta t = 2\frac{D_1 + D_2}{c}$$

Finally, we get,

$$\Delta x = f_2 4\omega \frac{D_1 + D_2}{c}$$

Plotting our $\Delta x\ vs\ \omega$ we should see a slope with relation to the value of c, multiplying the slope by our constants we should get the speed of light.

$$c = 4 f_2 D_3 \frac{\omega}{\Delta x} \qquad D_3 = D_1 + D_2$$

Uncertainties

Our uncertainties will come primarily from measurements.

Path Uncertainty σ_{D3}: While measuring the distance between the mirrors we have uncertainties σ_{D_1} and σ_{D_2}since they are related we can simply add them. $\sigma_{D3} = \sigma_{D_1} + \sigma_{D_2}$ now, the beam is not set up perfectly so that it travels in the exact same path, there is minor uncertainty here.

Focal Point Uncertainty σ_{f2}: We are given the focal point of the beam, there is uncertainty in this measurement. Furthermore, focal points are not actually points but rather ranges. It is impossible to get a focal range of < $\frac{\lambda}{2}$ this results in minor aberrations at the beam splitter.

Motor and mirrors σ_f: Mirrors are not perfectly smooth and reflective surfaces; this adds minor aberrations. The motor itself could also have uncertainties, for example, in the measurement of frequency, there are oscillations in the last digit. The motor works by revolving a belt, which revolves a smaller belt to introduce very fast oscillations. There are uncertainties in the wear of these belts and the function of the motor itself.

Measuring Microscope and beam size $\sigma_{\Delta x}$: The microscope had tick marks in the millimeter range, using the micrometer drum we could move another measuring tick, this as implied was $1/100^{th}$ of a millimeter. Therefore, we had an uncertainty of $1/200^{th}$. Now, measuring the shift in the beam was done by setting an initial position x_0 and recording how many micrometers we had to adjust to reach the center of the beam again. But, the beam is not an infinitely thin point, it has a radius and therefore setting the tick mark in the center is an approximation. We recorded the beam size to be 5 micrometers. This means if you are approximating the center you have an uncertainty of ±2.5 micrometers. We need to add these two uncertainties in quadrature because they aren't significantly correlated.

Lens placement and more: Firstly, lens' are not perfect and therefore there are aberrations when a beam goes through a lens. Now, the placement of the lens makes this effect more profound, since we cannot place lens perfectly in the same exact axis as another lens we cannot have a perfect telescope. These imperfections in the lens add uncertainty to the angle θ and ϕ.

Note: The aforementioned uncertainties and assumptions are so insignificant they do not impact our final uncertainty, our biggest source of uncertainty comes from Δx by two order of magnitudes.

Partials and Calculations

$$c = 4f_2 D_3 \frac{\omega}{\Delta x}, \quad \omega = 2\pi f$$

$$\sigma_c = \sqrt{\left(\frac{\partial \mathrm{c}}{\partial \Delta \mathrm{x}} \sigma_{\Delta x}\right)^2 + \left(\frac{\partial \mathrm{c}}{\partial \mathrm{f}_2} \sigma_{\mathrm{f}_2}\right)^2 + \left(\frac{\partial \mathrm{c}}{\partial \omega} \sigma_{\omega}\right)^2 + \left(\frac{\partial \mathrm{c}}{\partial \mathrm{D}_3} \sigma_{\mathrm{D}_3}\right)^2}$$

$$\frac{\partial \mathrm{c}}{\partial \Delta \mathrm{x}} = -4D_3 f_2 \frac{\omega}{\Delta \mathrm{x}^2}$$

$$\frac{\partial \mathrm{c}}{\partial \mathrm{f}_2} = 4D_3 \frac{\omega}{\Delta x}$$

$$\frac{\partial \mathrm{c}}{\partial \omega} = 4D_3 f_2 \frac{1}{\Delta x}$$

$$\frac{\partial \mathrm{c}}{\partial \mathrm{D}_3} = 4f_2 \frac{\omega}{\Delta x}$$

If we plot Δx versus ω we can solve for c by

$$c = 4f_2 D_3 (Slope)^{-1}$$

Which has a similar uncertainty

$$\sigma_c = \sqrt{\left(\frac{\partial \mathrm{c}}{\partial \mathrm{f}_2} \sigma_{\mathrm{f}_2}\right)^2 + \left(\frac{\partial \mathrm{c}}{\partial \mathrm{D}_3} \sigma_{\mathrm{D}_3}\right)^2 + \left(\frac{\partial \mathrm{c}}{\partial \mathrm{S}} \sigma_{\mathrm{S}}\right)^2}$$

$$\frac{\partial \mathrm{c}}{\partial \mathrm{f}_2} = 4D_3 (Slope)^{-1}$$

$$\frac{\partial \mathrm{c}}{\partial \mathrm{D}_3} = 4f_2 (Slope)^{-1}$$

$$\frac{\partial \mathrm{c}}{\partial \mathrm{S}} = -4D_3 f_2 (Slope)^{-2}$$

To point it out again, $\sigma_\omega = 2\pi\sigma_f$

$$D_3 = D_1 + D_2, \qquad \sigma_{D3} = \sigma_{D_1} + \sigma_{D_2}$$

$$D1 = 10.1m, \ D2 = 10.1m, \ D3 = 20.2m$$

$$\sigma_{D_1} = 0.0127m, \qquad \sigma_{D_2} = 0.0127m, \qquad \sigma_{D3} = 0.0254m$$

$$\sigma_{\Delta x} = \sqrt{2.5\mu m^2 + 1/200^{th} mm} = 25\mu m$$

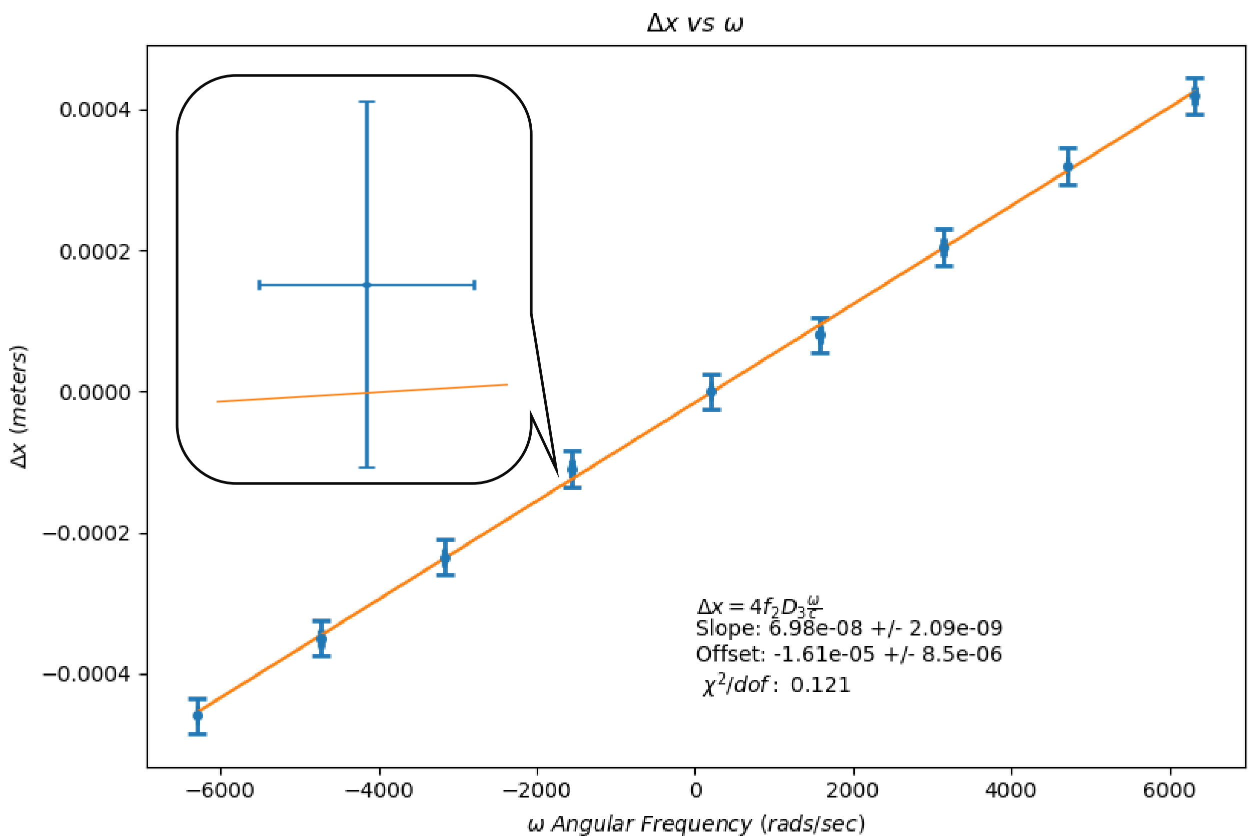
$f_2 = 0.256m$ was given; we took an uncertainty of $\sigma_{f_2} = 0.0005m$

More Calculations/Data Analysis

Weighted mean, Weighted Uncertainty

$$\bar{c} = \frac{\sum_{i=1}^{N} \frac{c_i}{\sigma^2{}_{ci}}}{\sum_{i=1}^{N} \frac{1}{\sigma^2{}_{ci}}}$$

$$\sigma_{\bar{c}} = \sqrt{\frac{1}{\sum_{i=1}^{N} \frac{1}{\sigma^2{}_{ci}}}}$$



Using the slope and the equation for c and σ_c mentioned earlier, we get a value of

$$(2.95 \pm 0.09) \cdot 10^8 \frac{m}{s}$$

We can see our fit is very closely related to the data points with a Chisq/dof value of 0.14. I plotted the σ_ω but, it is not considered in the fit mainly because I have not coded the function to fit the data and LT.box fit does not do x error values.

The offset in our data is representative of error because the true intercept should be at 0. Our offset is within two uncertainties of 0 and therefore we can consider it insignificant. We can't use relative error here because it wouldn't make sense.

Weight Mean Method

Using the equations shown earlier we calculated a weighted value for c and σ_c of

$$(2.93 \pm 0.09) \cdot 10^8 \frac{m}{s}$$

Negative values represent clockwise rotation

Data:

Δx (micrometers)	$\frac{\omega}{2\pi}$ Hz	$\frac{\sigma\omega}{2\pi}$Hz
0	32	1
8	252	1
20.5	501	3
32	750	1
42	1005	2
-11	-248	2
-23.5	-504	2
-35	-752	2
-46	-1002	1

The accepted speed of light is 2.99792458e8 m/s which means we got an experimental value extremely close to the accepted value.

Relative deviation from accepted value for weighted mean:

$$\%RD \ = \frac{2.99792458 - 2.93}{2.99792458} \cdot 100 = 2.3\%$$

$$\%Relative\ Uncertainty = \frac{0.09}{2.93} \cdot 100 = \ 3.1\%$$

The weighted mean uncertainty covered the %RD and therefore we can successfully correlate our experiment to the accepted value, this means a successful experiment.

Relative deviation from accepted value for graph:

$$\%RD = \frac{|2.99792458 - 2.95|}{2.99792458} \cdot 100 \ = \ 1.6\%$$

$$\%Relative\ Uncertainty = \frac{0.09}{2.95} = 3.0\%$$

The deviation from the accepted value was within our uncertainty and therefore we can consider this result successful.

Conclusion

If we had a more precise method of measuring Δx we would've seen even closer results. The data that was graphed and fitted had a slightly better result. I would've liked to consider the σ_ω in the fit however, it was mostly insignificant anyways. I would say the experiment was a success overall as we managed to get very close to the accepted value for the speed of light with decent uncertainty.

Raw values, Raw Data, Code

chisq/dof = 0.13962824512372077

offset = -1.6359677377632725e-05 +/- 9.013878786071193e-06

slope = 6.982278720269624e-08 +/- 2.089236755988209e-09

############# WEIGHTED MEAN C VALUE #################

2.925490436862876e+08 +/- 8.760084746183107e+06

########### GRAPH C VALUE #############

2.9549917479093796e+08 +/- 8.868566455802042e+06

getData.py you will need to run code Note: you will need to install sympy module

```
import LT.box as B
import numpy as np
from sympy import *


def makeDict(names):
    Data = {}
    for k in range(len(names)):
        Data[names[k]] = dict(dataFile=B.get_file(names[k] + '.data'))
        Data[names[k]]['Parameters'] = {}
        # file.par.get_all_data() creates a dictionary but it stores it all it's
contents in strings.
        for x in Data[names[k]]['dataFile'].par.get_variable_names():
            Data[names[k]]['Parameters'][x] =
Data[names[k]]['dataFile'].par.get_value(x)
        for j in range(len(Data[names[k]]['dataFile'].get_keys()) - 1):
            Data[names[k]][Data[names[k]]['dataFile'].get_keys()[j]] = \
                B.get_data(Data[names[k]]['dataFile'],
Data[names[k]]['dataFile'].get_keys()[j])
    if len(names) == 1: Data = Data[names[0]];
    return Data


def wmean(x, sig):
    w = 1. / sig ** 2
    # weighted mean
    wm = np.sum(x * w) / np.sum(w)
```

```
    sig_wm = np.sqrt(1. / np.sum(w))
    return wm, sig_wm


def labels(xlabel='', ylabel='', title='', annotate='', fit=None, xy=(0.5, 0.1),
xycoords='axes fraction', sig=.4,fontsize=10):
    if xlabel: B.pl.xlabel(xlabel);
    if ylabel: B.pl.ylabel(ylabel);
    if title: B.pl.title(title);
    if annotate:
        # noinspection PyStringFormat
        B.pl.annotate(f'%s\n\n\n' % annotate, xy=xy,
xycoords=xycoords,fontsize=fontsize)
    if fit:
        # noinspection PyStringFormat
        B.pl.annotate(f'Slope: %{sig}g +/- %{sig}g \nOffset: %{sig}g +/- %{sig}g\n
$\chi^2/dof: $ %{sig}g' % (
            fit.slope, fit.sigma_s, fit.offset, fit.sigma_o, fit.chi_red), xy=xy,
xycoords=xycoords,fontsize=fontsize)


def quadrature(*args):
    args2 = (0, 0,
             0) + args  # bug fix, if you don't add a tuple of zeros the next
statement adds all the numbers together and doesn't create seperate arrays
    args = np.asarray(args2)
    return np.sqrt(np.sum(args ** 2))


def partials(expression, variables, **values):
    partials = {}
    var = variables.split()
    for k in var:
        locals()[k] = Symbol(k)
    expression = eval(expression)
    for k in expression.free_symbols:
        partials[k] = {}
        partials[k]['Partial'] = diff(expression, k)
    for k in partials:
        llist = [x for x in partials[k]['Partial'].free_symbols]
        partials[k]['Lambdify'] = lambdify(llist, partials[k]['Partial'])
        templist = [i for i, j in zip([str(x) for x in
partials[k]['Partial'].free_symbols], values)]
        results = [values[x] for x in templist]
        partials[k]['Evaluated'] = partials[k]['Lambdify'](*results)
    return partials
```

Actual Code

```
from getData import *

Data = makeDict(['Data'])
DataP = Data['Parameters']
DataP['D3'] = DataP['D1'] + DataP['D2']
DataP['dD3'] = DataP['dD1'] + DataP['dD2']
```

```
# Data adjustment might vary depending on data collection
Data['x'] = Data['x'] * 0.001 * 0.01
Data['dx'] = Data['dx'] * 0.001 * 0.01
### Adding uncertainty in radius of beam in quadrature
Data['dx'] = quadrature(Data['dx'], 0.000025)
Data['fr'] = Data['fr'] * 2 * np.pi
Data['dfr'] = Data['dfr'] * 2 * np.pi

B.pl.figure('Speed of Light')
B.plot_exp(Data['fr'], Data['x'], Data['dx'], xerr=Data['dfr'])
fit = B.linefit(Data['fr'], Data['x'], Data['dx'])
B.plot_line(fit.xpl, fit.ypl)

labels('$\omega\ Angular\ Frequency\ (rads/sec)$', '$\Delta x\ (meters)$', '$\Delta
x\ vs\ \omega$',
       '$\Delta x = 4f_2D_3 \\frac{\omega}{c} $',
       fit=fit,sig=.3)

# Uncertainties and partials yay!
slope_partials = partials('4*f2*D3/S', 'f2 D3 S', f2=DataP['f2'], D3=DataP['D3'],
S=fit.slope)
eq_partials = partials('f2*4*f*D3/x', 'f2 f D3 x', f2=DataP['f2'], f=Data['fr'],
D3=DataP['D3'], x=Data['x'])
# Localizing variables used in function
for keys in eq_partials:
    locals()[str(keys)] = keys
for keys in slope_partials:
    locals()[str(keys)] = keys

c_exp1 = DataP['f2'] * 4 * Data['fr'] * DataP['D3'] / Data['x']
# Adding partials in quadrature
partx = (eq_partials[x]['Evaluated'] * Data['dx'])
partD3 = (eq_partials[D3]['Evaluated'] * DataP['dD3'])
partf = (eq_partials[f]['Evaluated'] * Data['dfr'])
partf2 = (eq_partials[f2]['Evaluated'] * DataP['df2'])

sigmatotal = quadrature(partx, partD3, partf, partf2)

# Getting weighted mean of data
results = wmean(c_exp1, sigmatotal)
print('\n\n############# WEIGHTED MEAN C VALUE #################\n' + str(
    np.format_float_scientific(results[0])) + ' +/- ' +
str(np.format_float_scientific(results[1])))

# Calculating C value from graph
c_exp2 = (4 * DataP['f2'] * (DataP['D3'])) / fit.slope
### Uncertainty for slope method
part1 = slope_partials[f2]['Evaluated'] * DataP['df2']
part2 = slope_partials[D3]['Evaluated'] * DataP['dD3']
part3 = slope_partials[S]['Evaluated'] * fit.sigma_s

c_exp2_uncertainty = quadrature(part1, part2, part3)
print('########### GRAPH C VALUE #############\n' +
str(np.format_float_scientific(c_exp2)) + ' +/- ' + str(
    np.format_float_scientific(c_exp2_uncertainty)) + '\n\n')
```

Data.data

```
#\f2 = 0.256
#\df2 = 0.0005
#Unit is meters [m]

#\D1 = 10.052
#\dD1 = 0.0127
#Unit is meters [m]

#\D2 = 10.097
#\dD2 = 0.0127
#Unit is meters [m]


#! x[f,0]/ fr[f,1]/ dx[f,2]/ dfr[f,3]/
#0       32      0.5     1
8       252     0.5     1
20.5    501     0.5     3
32      750     0.5     1
42      1005    0.5     2
-11     -248     0.5     2
-23.5   -504     0.5     2
-35     -752     0.5     2
-46     -1002    0.5     1
```