UROPS Report

Lin Zhonglin A0222183N

April 1, 2023

Contents

1 Introduction

Craft a short text that links up all section of my report. Possibily linking to the main ion trapping expt in lab.

2 Building and tuning a compact 638nm Grating Stabilized Diode Laser

Since their first use in atomic physics in the early 80's, diode lasers have become an important part of many modern experiments. This is primarily due to the high reliability and the low price of these devices which facilitate experiments involving a number of lasers operating at different frequencies [**?**].

This section of the report discusses a type of compact grating-grating stabilized diode laser system. There was a constructed 638nm laser system of this type in the lab with a dead diode. I replaced the diode and tuned the laser back to 638.....nm wavelength for probing of one of the Ytterbium ion state.

Resources relevant to this task:

1. book - Principles of Lasers [**?**]: Chapter 9 discusses semiconductor lasers. Chapter 4.7 discusses Gaussian Beam Optics.
2. paper - A compact grating-stabilized diode laser system for atomic physics [**?**]: This paper discusses the detailed mechanical design and construction process of one of the most commonly used diode lasers in optics labs, with linewidth of a few 100kHz.

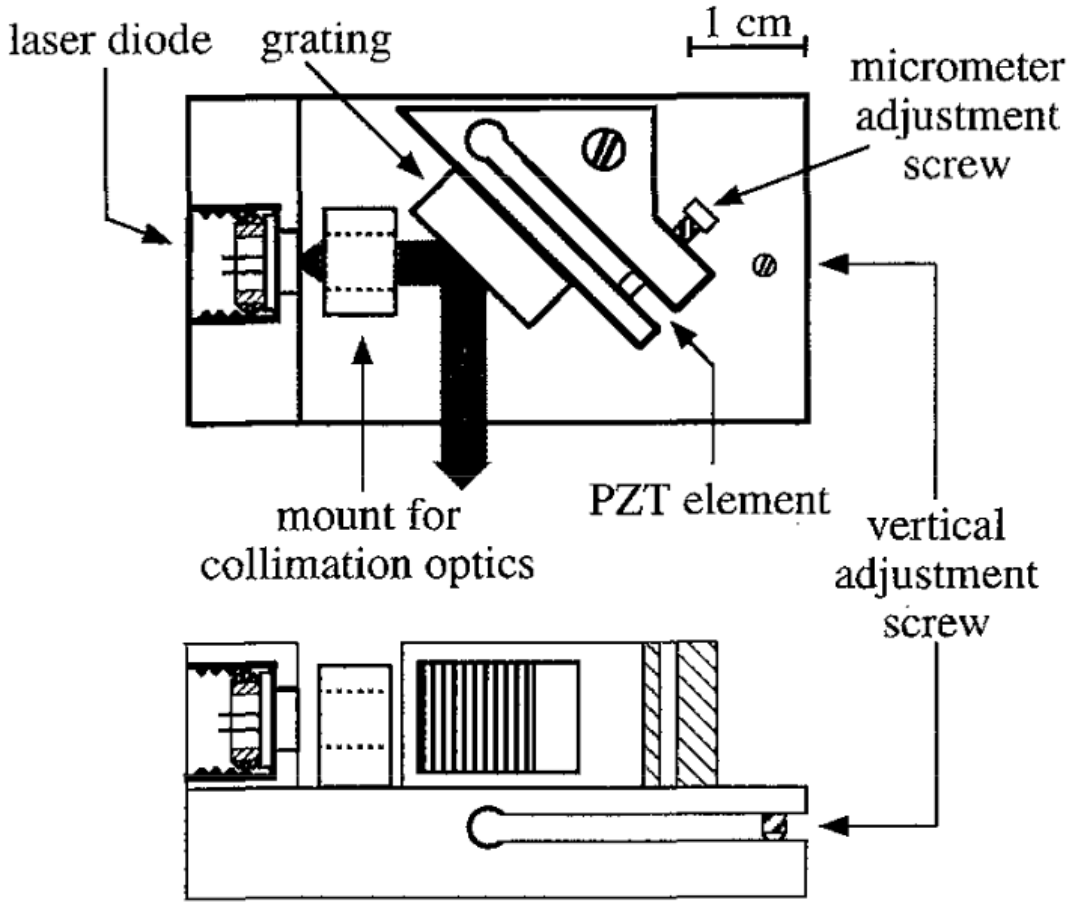

Figure 1: Compact Diode Laser Schematics

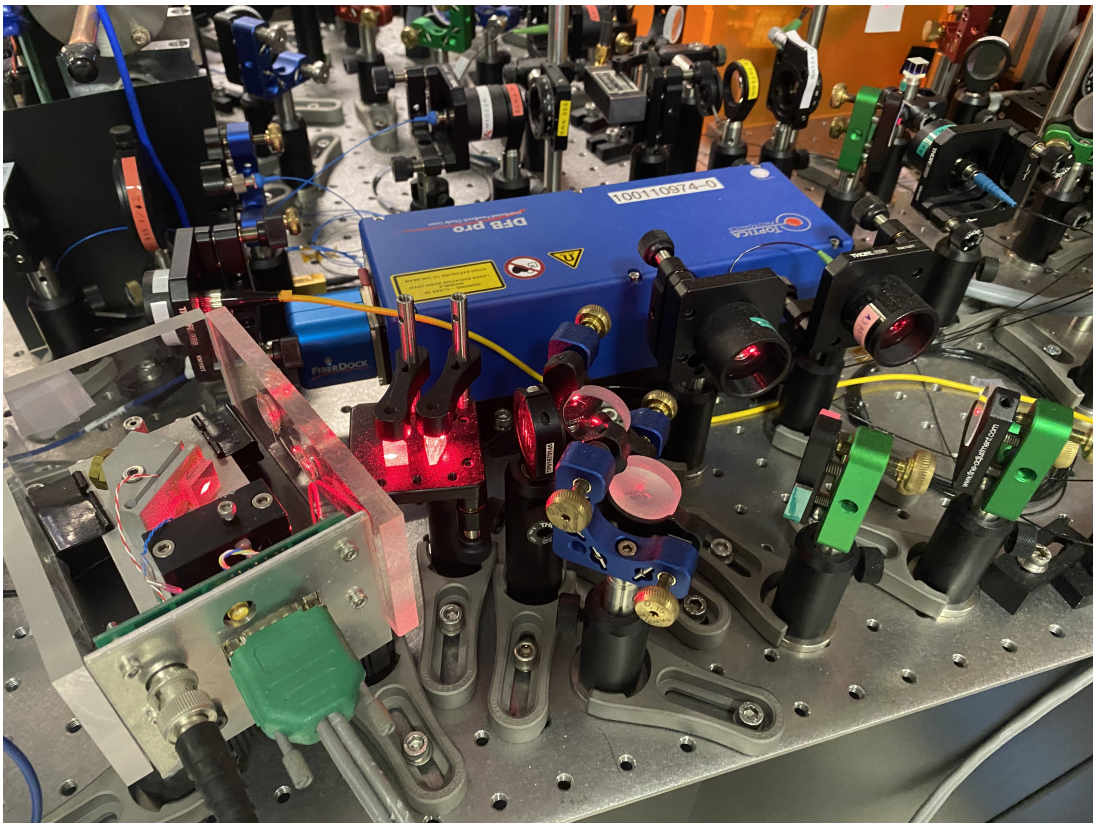

Figure 2: 638nm Laser Setup

Although a free-running diode suffers from large linewidth, an optical cavity that provides optical feedback can greatly reduce its linewidth. In the system under discussion, first order light diffracted light from a grating is coupled back into the diode, so that the grating and the diode's facet form an external resonator. The use of a grating results in a linewidth reduction of two orders of magnitude and it also allows the selection of any desired wavelength. The external cavity length (distance between grating and diode) is chosen to be about 15mm optimise for single mode operation. A peltier element is embedded in the mechanical structure to maintain temperature of the system. A piezo actuator is embedded behind the grating to allow small adjustment and scan of the external cavity length. (See [**?**] for more details)

To tune the emission wavelength of this laser, consider the following rough picture. The gain profile of such laser diode has a typical width of 10nm. The emission wavelength is determined by the competition between the longitudinal modes of the laser cavity, which are typically separated by 100-200 GHz. The laser operates at the mode which experiences the maximum gain (See Fig **??**.

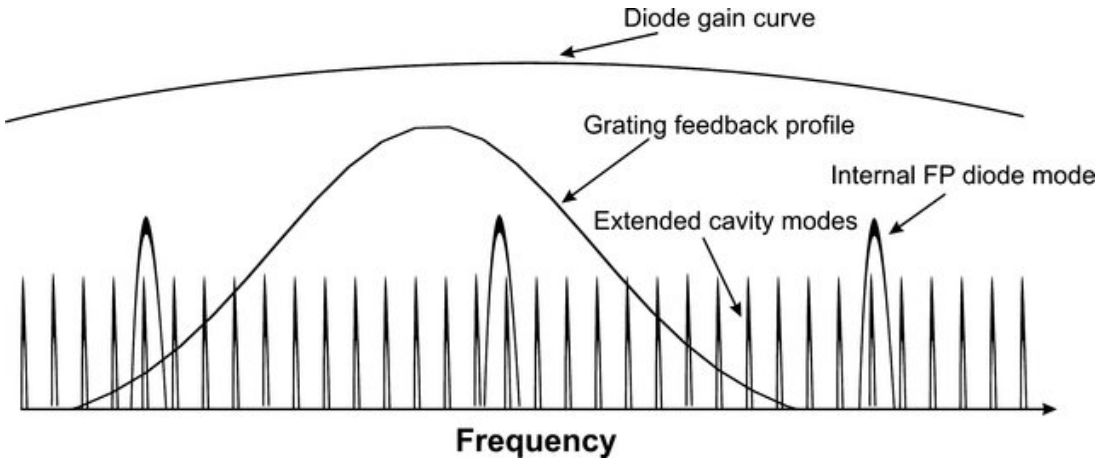

Figure 3: Laser Diode Gain Profile

Hence there are two main ways of tuning the emision wavelength, tuning the temperature or the injection current: [**?**].

1. Temperature variations cause a change of the cavity length (and thus a change of the resonance frequencies of the longitudinal modes) and at the same time a frequency shift of the gain profile. Consequently, the emission wavelength can be tuned with the temperature. However, in the process of tuning, the gain of the modes varies such that at some point the laser eission frequency jumps from one mode to another (commonly referred to as **mod-hopping**), resulting in a number of inaccessible frequency domains. [**?**] suggests that the width of the accessible frequency domain is typically a fraction of the spectral range of the laser cavity. Here with laser cavity being 15mm, it's several tens of GHz.

2. A second method which can be employed to tune the emission wavelength is to vary the injection

current. This leads to a corresponding temperature change and also a change of the carrier density and thus of refractive index of the semiconductor material. Injection current tuning also suffers from mode hopping.

Note: Temperature tuning can cover a frequency range of the order of several tens of nm in wavelength at a typical rate of 0.3 nm/K, injection current typically covers a range of several tens of GHz only at a typical rate of 4GHz/mA.

After careful replacement of the original dead diode, coarse adjustment of the wavelength is achieved by horizontally tilting the grating with the micrometer screw. The main goal now is to vary the temperature and injection current in search for a mode-hopping-free region in the gain profile near the desired wavelength of around 638.615nm. Fig **??** shows a few tens of data points of temperature-EW plots. The bottom data points from a dense grid as expected, showing that it's a mode-hop-free region. Whereas the top few data points probably is a result of mode hopping. After many trial and error, the final set of parameters arriving at EW=638.615nm is [set temperature for the peltier = 17.92°, set current = 52.3mA, current offset = -0.563 mA, piezo voltage = 2.5 V] However, this EW still suffers from drifting.

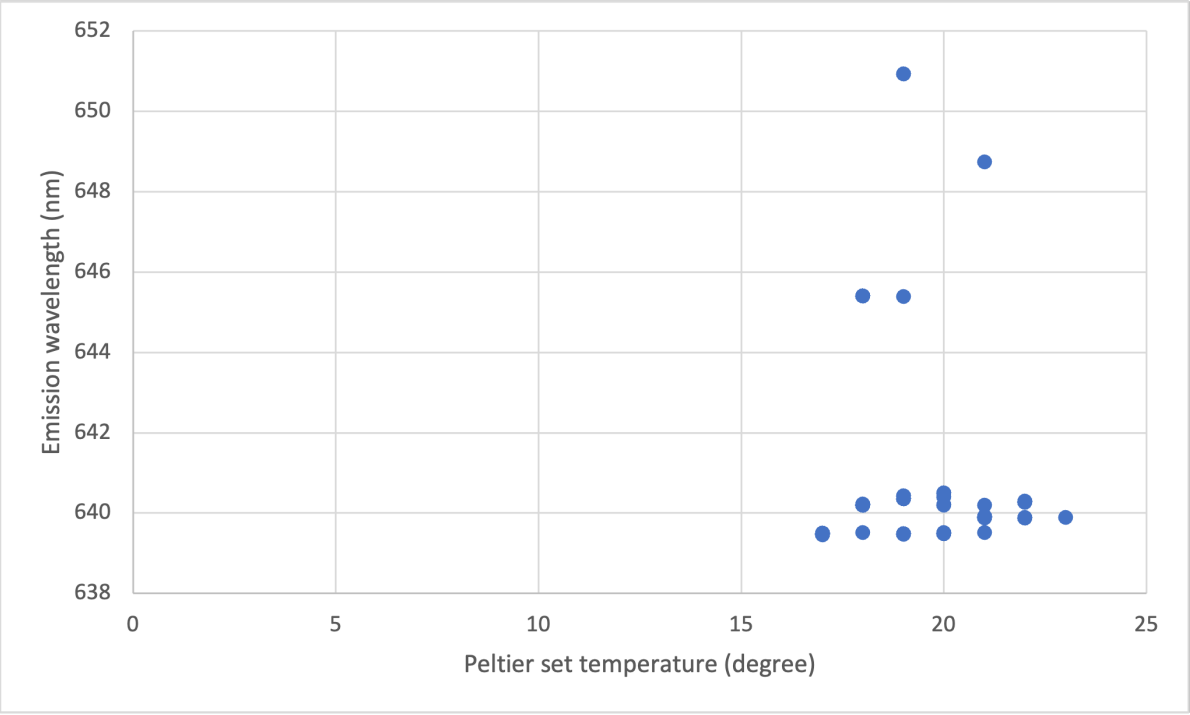

Figure 4: Diode EW - temperature

2.1 Gaussian Beam Characterisation and collimation

When dealing with laser beams, Gaussian beam optics is commonly considered. As the laser diode has a rectangular shape, its initial output laser beam has an oval shape and thus not very Gaussian and not very suitable for subsequent Gaussian beam characterisation and coupling to optical fibres.

A pair of anamorphic prism is first used to tune the laser beam cross section to be as circular as possible. (See the two red-glowing prisms in Fig **??**) Next, it's desirable to collimate the beam for ease of coupling into other optical components. Coarse collimation can be done by eyeballing the change of the beam spot diameter at different distance from the collimating lens and adjusting the position of the collimating lens to make beam spot diameter as constant as possible across the scanning distance. To achieve finer collimation, consider the following Gaussian beam characterisation process:

First we need to measure the beam spot diameter at different distances from the collimating lens. To do so, consider the knife edge measurement for very precise measurement. Alternatively consider using a CCD camera for measurement for precision of up to μm level precision (size of CCD pixels).

To do a proper knife edge measurement, refer to [**?**] for details. A piece of paper blade is painted with black paint to use as the blocking knife edge. Black paint is to reduce reflection of laser beam. The black-painted paper blade is then mounted onto a translational stage as shown in Fig .... Position the blade perpendicular to the laser beam. Place a photodiode behind the blade as shown in Fig **??** and connect the photodiode to an Arduino. Finally connect the Arduino to a computer and run code **??** in Annex. The code collects three sets of voltage reading from the photodiode and take average. Move the knife edge to block more light by rotating the translational stage. run code **??** in Annex again. Repeat the previous steps until reading from the photodidoe is close to background reading. Now input the distance (of knife edge) - photodiode voltage data from previous steps to code **??** in Annex for data processing. This code fits the distance-voltage data to an error function and outputs a fitted value of the laser beam width.

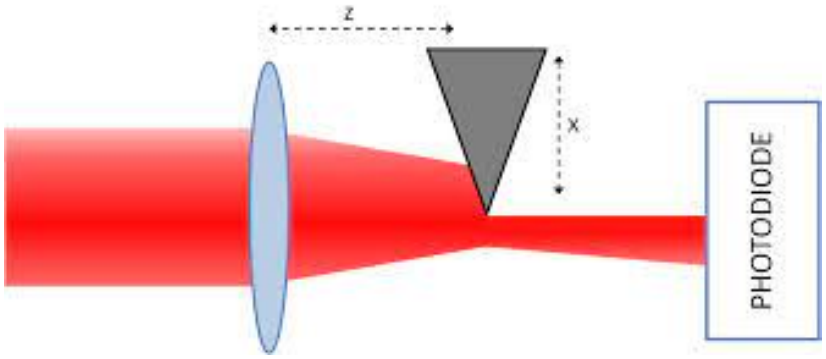

Figure 5: Knife Edge Measurement

Although the aforementioned knife edge measurement with a knife blade is can be very precise up to the limit of diffraction caused by the blade, it's very time consuming without the translational stage being automated. With time constraint, a less precise (up to μm level) yet much faster method with a CCD camera can be employed. Place a camera such that the laser beam spot falls completely on the CCD sensor. Run code **??** in Annex to measure the laser beam spot horizontal and vertical radius at the position of the CCD sensor. The find horizontal radius, the code sums up all rows of

pixels and filter for pixels exceeding $1/e^2$ of the pixel with maximum reading. Then take the right most pixel position minus the left more pixel position as an estimate of the beam spot diameter. The code runs similarly for vertical beam spot radius. For details, see code **??** in Annex.

I used a Point Grey Research Chameleon CMLN-13S2M CCD camera for date taking.

After measuring the beam diameter at as many different distances as possible, fit the data to the Gaussian beam beam radius - z position by using the code ... in Annex to arrive on estimation of the beam waist radius and z position. Adjust the collimating lens accordingly to make the beam waist z position as far as possible.

2.2 Laser beam - Fiber Coupling Procedure

In order to monitor the laser system emission wavelength, part of the laser beam power is sent to the wavemeter via a fiber. To send a laser beam into a single-mode optical fibre is to couple their spatial modes, follow this procedure for coupling optimisation:

1. Make sure that the laser beam at least go through one mirror before being sent to the fibre mount to allow for two points of freedom. Rough align the collimated beam through the fiber mount without the fibre on. Try to center the beam through the fibre mount entry.

2. Screw optical fiber onto the fiber mount and attach a fiber testing pen to the exit end of the fibre. Place a piece of paper on the optical path and try to align the two spots from the diode laser and the fibre pen by iteratively adjusting the mirror and the fiber mount.

3. Detach the fibre pen and sent the light from the exit end of the fibre (if no light at all, go back to previous step) to a power meter and perform "Walking-the-beam" technique [**?**] to optimise coupling.

4. If the previous step doesn't go to efficiency above 50%, turn the fiber mount lens position to optimise. Note that this is very sensitive, so adjust the fibre mount lens with small turns of about 2-3° at a time. As the optimised position is approached, even finer adjustment is required.

5. Note that coupling efficiency above 60% requires significantly more effort than described. More sophisticated techniques need to be employed. However, this methods should lead to a minimum of 60% coupling, if not achievable consider:

   (a) Clean the optics: fibre cross section and mirrors.

   (b) Check whether the fibre has any leakage by using a fibre pen and look for light leakage in a dark room.

   (c) Consider tuning in a larger range to get out of a possible local maximum.

3 EOM (Electro-optic Modulator)

In certain types of crystals it is possible to effect a change in the index of refraction that is proportional to the applied electric field. This is the linear electro-opic effect (also known as the Pockels effect). Applying a modulated electric field across such a nonlinear crystal (usualy the KDP type) and passing a beam of laser through the crystal leads to the laser beam being modulated in phase. (detail see: [**?**]

3.1 Theory and Model

When a time-varying electric field is applied to the nonlinear crystal inside the EOM, the index of refraction n changes in a sinusoidal manner resulting in phase modulation of the laser beam passing through the crystal. Consider the following short mathematical example to see that phase modulation is equivalent to frequency:
Suppose instantaneous electric field at a point of EM wave of a laser beam:

$$E_c(t) = A_c cos(\omega_c T + \varphi_c)$$

With phase modulation (assume sinusoidal modulation), we have final phase of the electric field of the EM wave:

$$\begin{aligned}\psi(t) &= \omega_c t + \varphi_c + \Delta\varphi(t) \\ &= \omega_c t + \varphi_c + m_\varphi sin(\omega_m t)\end{aligned}$$

So the modulated electric field now becomes:

$$E(t) = A_c cos[\omega_c t + m_\varphi sin(\omega_m t) + \varphi_c] \tag{1}$$

$$= A_c cos\left\{\int_0^t [\omega_c + \Delta]\omega(t)]dt + \varphi_c\right\} \tag{2}$$

$$\tag{3}$$

where,

$$\Delta\omega(t) = \frac{d\Delta\varphi(t)}{dt}$$

If we do Fourier transform to Eqn **??**, we obtain in frequency space:

$$E(\omega) = A_c J_0(m)\delta(\omega - \omega_c) + A_c \sum_{n=1}^{\infty} J_m(m)\left\{\delta[\omega - (\omega_c + n\omega_m)] + (-1)^n \delta[\omega - (\omega_c - n\omega_m)]\right\} \tag{4}$$

Eqn **??** shows that when a single-frequency laser beam is modulated by a sinusoidal frequency, the modulated beam in frequency space comprises of the unmodulated frequency ω_0 and infinite sidebands. Sideband frequency spacing is ω_m, modulation frequency. Sideband amplitudes are

determined by the Bessel function $J_n(m)$. (See Fig **??**) What's of concern in this case is that EOM phase modulation produces frequency sidebands of laser beam.

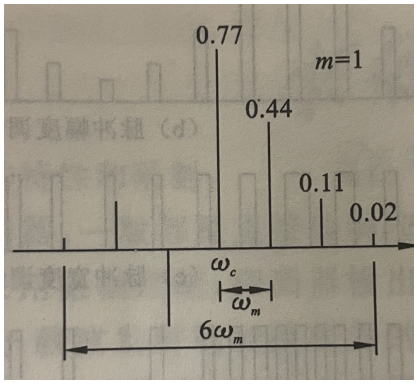

Figure 6: modulated laser beam frequency sepctrum

3.2 EOM Tank Circuit Board: Model and Construction

To construct an EOM tank circuit board and tune it to a partiular frequency, the following sources are useful:

1. paper: Design and construction of an efficient electro-optic modulator for laser spectroscopy[**?**]
2. textbook: Fundamentals of Photonics[**?**]
3. Prof Christian group's internal wiki has an entry on "How to build an EOM". There is a set of detailed note on EOM construction written by Meng Khoon.

The schematic of EOM circuit board is shown in Fig **??**. Note that the components L, C_1, C_2 are nonideal components, i.e. the parasitics are if gnored here. C_1 refers to EOM crystal (ideally with correct AR coating for the relevant wavelength, in my case ≈738nm and ≈965nm)

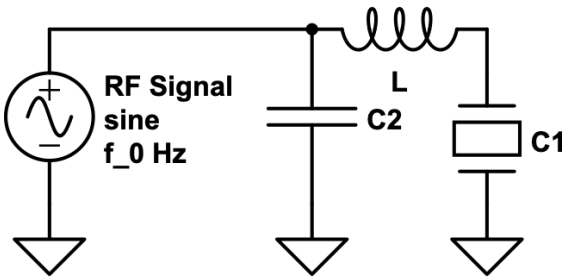

Figure 7: EOM tank circuit board schematic (non-ideal component)

The task of constructing an EOM at a particular modulation frequency can be be roughly be broken into the following three steps:

Firstly, the LC circuit is in place to act as an amplifier so that the EOM can driven with low-power RF source of < 1W, yet still providing significant electric field across the crystal for phase modulation.

Since we have an intended EOM modulation frequency frequency f_0, and the capacitance of the EOM crystal C_1 is fixed, we need to measure C_2 in order to backcalculate the inductor L needed. Since the capacitor is small at the pF level, we need to characterise C_1 by hooking it up with a suitable resistor, sending a square wave through it and determine its response time. Although the capacitance of the crystal should be predominantly determined by the crystal dielectric, my observation showed that the way that it's glued to the tank circuit also affects. I used silver containing glue spreaded evenly under the crystal and attached the crystal to the EOM tank circuit board's exposed copper section. The conductive glue is such that bottom of the crystal has a common ground hence electric field across the crystal will be near homogeneous as wanted. For my crystal with the way that it was glued, $C_1 \approx 3\mu F$.

After measurement of C_1, we can estimate the capacitance of the tank circuit board to be $C' = \frac{C_1 C_2}{C_1+C_2} \approx C_1$. This is because $C_1 \gg C_2$ where C_2 is a coupling capacitor that will be explained later. The resonant frequency of the tank circuit can be estimated to be $f = \frac{\omega}{2\pi} = \frac{1}{2\pi\sqrt{LC'^2}}$. Rearranging to get $L = \frac{1}{(2\pi f C_1)^2}$, to get a resonant frequency of about 35 MHz, a $2\mu H$ inductor is needed, the closest off-the-shelf inductor is $2.2\mu H$.

Next the overall impedance of the EOM tank circuit needs to be matched to be 50Ω such such that RF signal going to the EOM tank will be mostly absorbed and not reflected. In order tune the overall impedance of the EOM tank circuit, a coupling capacitor C_2 is added as shown in schematic **??**. The overall loss of the EOM tank circuit can be modelled by a resistor R_p as shown in Fig **??**. Here we consider the components ideal. Note that parasitic resistance can be dependent on signal frequency, environment and many others. Here take it as $R_p \approx 15$ based on Meng Khoon's note. However, much more careful characterisations of the circuit can be done. The coupling capacitor does not affect the tank circuit resonant frequency significantly as $C_1 \gg C_2$. so $C' = \frac{C_1 C_2}{C_1+C_2} \approx C_1$ as mentioned earlier.

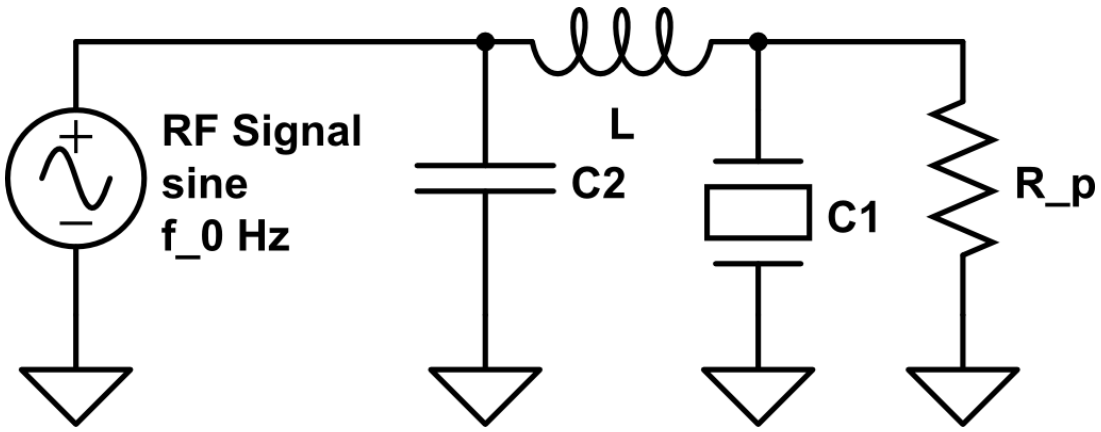

Figure 8: EOM tank circuit, modelling overall loss

We can calculate the overall impedance of the circuit based on this model. Setting overall impedance to be 50, the needed C_2 can be backcalculated to be $C_2 = \frac{\sqrt{50/R_p - 1}}{50} \approx 93.8nF$ which is indeed much smaller than $C_1 \approx 3\mu F$. Choose a variable capacitor around this range of capacitance.

Lastly set up a measurement circuit as shown in Fig **??**. Tune the variable capacitor until reflected signal from the EOM tank circuit board is minimum. Note that most variable capacitor has a tuning range of less than 360°, so tune slowly until the valley shown on the spectrum analyser is the deepest.

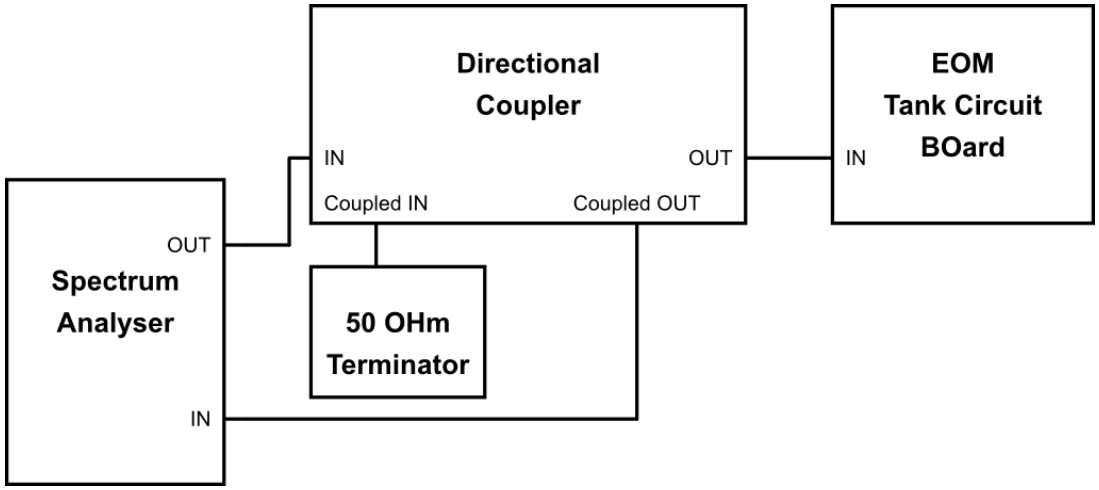

Figure 9: EOM frequency measurement setup

After tuning the variable capacitor, my EOM tank circuit board has a resonant frequency $f_0 \approx 38MHz$, see Fig **??**.

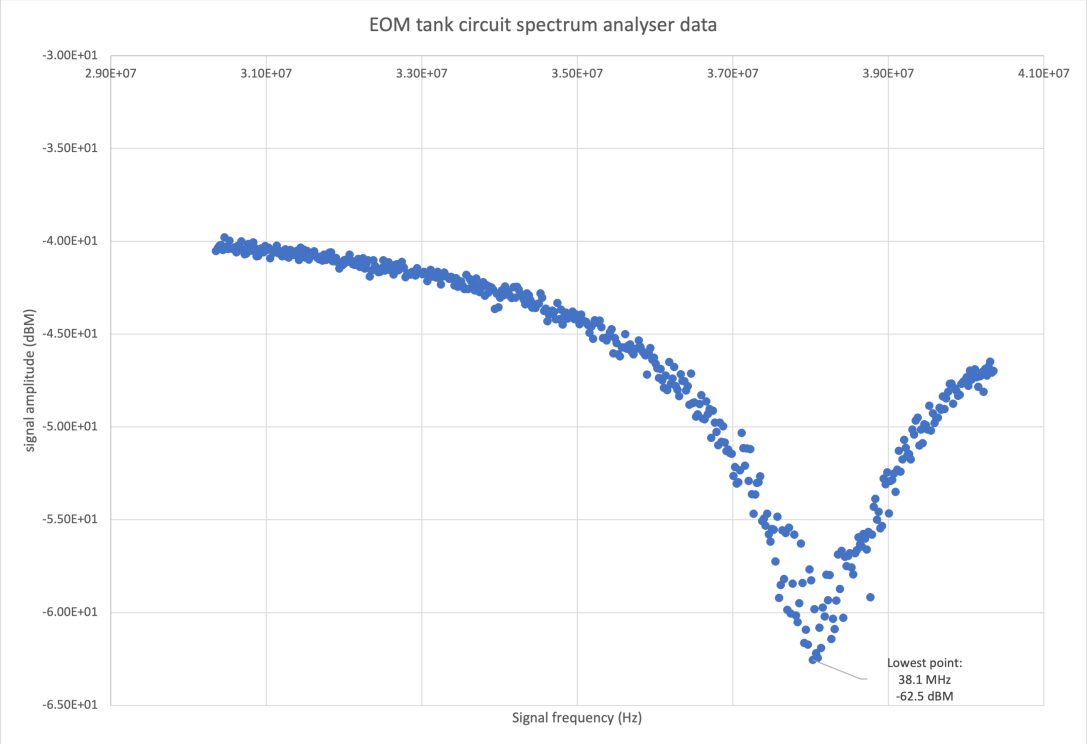

Figure 10: EOM tank circuit frequency measurement setup

A final word on EOM construction: To improve on the Q-factor of the EOM tank circuit, parasitics of the circuit need to be better charactersied and subsequently matching impedance better. A better impedance matching will lead to higher efficiency of absorption of input RF signal. A small-voltage RF signal can bring about more significant modulation. However, my EOM as it currently is suffices for my subsequent task.

4 Fabry-Perot Cavity

Laser resonators are open structures containing two or more mirrors that are aligned to produce optical feedback to the gain element. In the simplest case, a resonator consists of two aligned mirrors. These mirrors are called end mirrors and define the optical cavity. Optical radiation circulates within the cavity, bouncing back and forth between the end mirrors and passing through the gain element. An optical resonator can be characterised by the following parameters:

1. number of mirrors constituting the resonator
2. focal length of mirrors
3. radius of mirrors (usually small relative to cavity length)
4. length of cavity
5. reflectivity of mirrors

6. other optical losses: how good is the vacuum in the cavity and so on

What's relevant to this report is a particular type of optical resonator: confocal Fabry-Perot cavity. Fabry-Perot cavities are optical cavities with two parallel mirrors. Confocal Fabry-Perot cavities have focal points of the two concave mirrors overlapping at the center of the cavity as shown in Fig **??**. The confocal cavity I used in experiment has a cavity length of L=15cm.

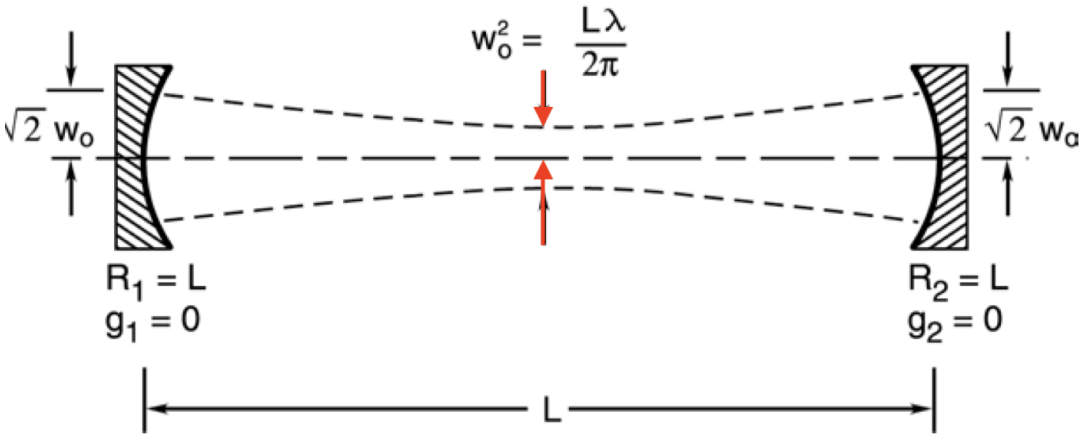

Figure 11: Confocal cavity

One good way to measure the frequency of a laser's beam is send it into a Fabry-Perot cavity and look at what gets transmitted (or reflected). Light can only pass through a Fabry-Perot cavity if twice the length of the cavity is equal to an integer number of wavelengths of the light. Or equivalently put, the frequency of the light's electromagnetic wave must be an integer number times the cavity's free spectral range $\Delta\nu_{fsr} \equiv c/2L$, where L is the length of the cavity and c is the speed of light. For my case $\Delta\nu_{fsr} \approx 1GHz$. The cavity acts as a filter, with transmission lines, or resonances, spaced evenly in frequency every free spectral range. Fig **??** shows a plot of the fraction of light transmitted through a Fabry-Perot cavity versus the frequency of the light. [**?**][**?**]

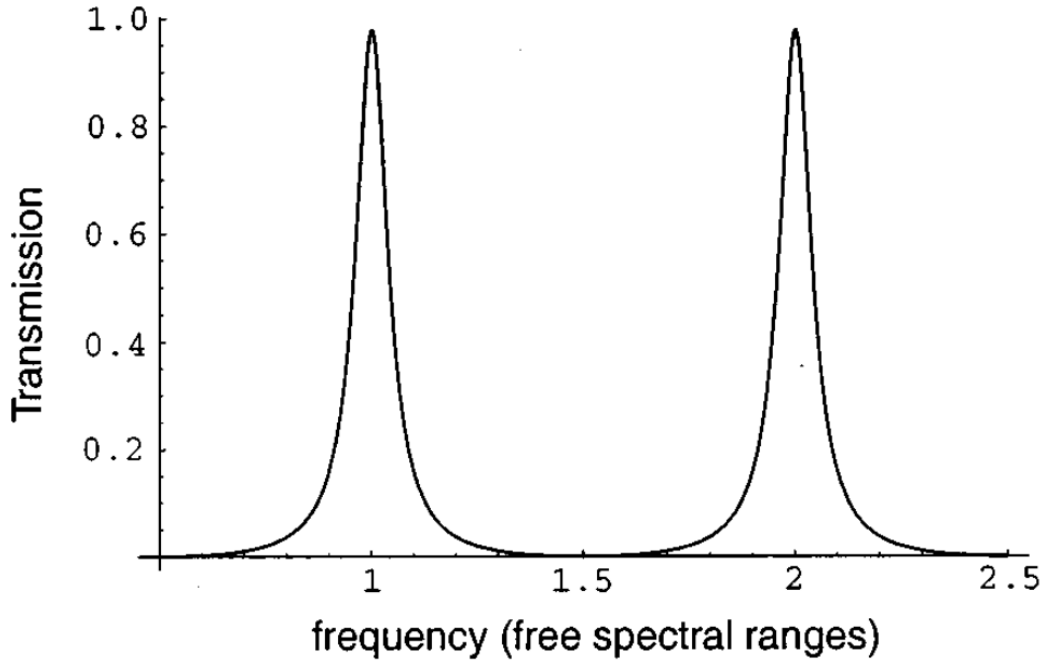

Figure 12: Cavity transmission

4.1 Laser beam - Cavity Coupling

To couple a well collimated laser beam with beam radius ω_1 (see section on laser beam collimation and characterisation in this report) into a Fabry-Perot cavity, follow these steps:

1. Choose a thin lens (with relevant AR coating) of focal length f to focus the laser beam from radius ω_1 to ω_0 at the center of the cavity. Gaussian beams have a propagating spherical wavefront, and each Gaussian beam is uniquely characterised by its beam waist, here being ω_0. In the context of a confocal cavity, when a Gaussian has beam waist $\omega_0 = \frac{L\lambda}{2\pi}$ as shown in Fig **??** the Gaussian beam spherical wavefront at the mirrors will match the curvatures of the mirrors, hence leading to stable standing waves to establish in the cavity. In my case, consider $L = 15$ cm, $\lambda = 935.18848$ nm, $\omega_0 = 0.000149419$ m.

2. To determine the focal length f, we consider how parameters of a Gaussian beam transforms through a thin convex lens (see [**?**] for details). Approximately, we have $\omega' \approx \frac{\lambda f}{\pi\omega}$. Where, ω' is the 'image'-side beam waist, ω is the 'object'-side beam waist, λ is laser beam wavelength, f is the thin lens focal length. In my case, consider $\lambda = 935.18848$ nm, $\omega' = 0.000149419$ m. $\omega = 1.1$ mm (image-side beam radius is estimated to be so through beam characterisation) $f = frac\omega'\pi\omega_1\lambda \approx 0.552139m \approx 550mm$. However, using a lens of such focal length was not practical due to space constraint on the optical table, I used a $f = 150mm$ thin lens as it was the longest focal length lens available off-the-shelf.

3. Pass the laser beam through at least one mirror before entering the cavity to obtain at least

2 points of adjustment. See Fig **??** in Appendix for a hand-written note on how to adjust the mirrors (or fibre mount) for optical coupling using a card (either normal paper or IR paper depending on wavelength) with a hole.

4. Lastly send a triangle wave signal of appropriate V_{pp} to the piezo actuator to scan the cavity length across a certain range. The V_{pp} needed depends on the scanning range needed and the behaviour of the piezo actuator. See chapter 2 of [**?**] for details on piezo actuators. For my case, after testing, a triangle wave generator board producing $\pm$5V, together with a voltage gain board of coefficient 19 is used to produce a $V_{pp} \approx 190$V. See Fig **??**, yellow line comes from a sync port of the triangle wave generator, green line comes from a photodiode at the exit end of cavity. Notice that "half a triangle" covers two peaks (not optimised in this image, but it was optimised after the image was taken), indicating that the cavity scan range is sufficient to cover a free spectral range of my cavity. Fine tune the cavity to obtain highest peaks possible

   Caution:

   (a) The piezo actuator might fracture under high negative voltage, so avoid sending in negative voltage to it. For my case, I used a offset gain baord to make sure the triangle wave sits above 0V.

   (b) The voltage sent to the piezo actuator is HV, be cautious and remember to electrically insulate the cavity from the optical table.

Figure 13: 935nm laser Cavity Scan on Oscilloscope

5 Frequency Locking a Laser: Pound-Drever-Hall Method

Following sources are relevant to the current section:

1. paper: An introduction to Pound–Drever–Hall laser frequency stabilization [**?**]. This paper gives an overview of the scheme of Pound-Drever-Hall method.
2. paper: Laser phase and frequency stabilization using an optical resonator [**?**]. This is original paper that proposed the PDH scheme in 1983.

Tunable lasers have multiple wavelength-selecting elements such as piezo-controlled etalons and gratings. Typically, the length of the lasing cavity is controlled by a voltage sent to a piezo-electric transducer. The laser-cavity length can change because of a number of factors such as temperature changes, and mechanical vibrations. These factors affect the laser-frequency stability. The standard PDH method of frequency-locking involves the following process: A laser's frequency is measured with a Fabry-Perot cavity, and this measurement is fed back to the laser to suppress frequency drift. [**?**][**?**]

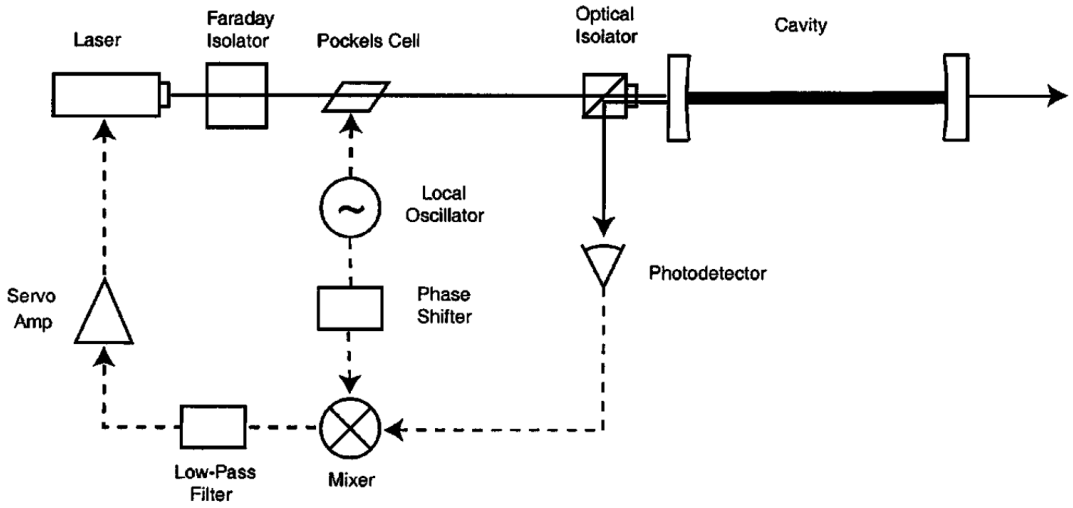

Figure 14: The basic layout for locking a cavity to a laser. Solid lines are optical paths and dashed lines are signal paths. The signal going to the laser controls its frequency [**?**]

Referring to Fig **??**, the basic steps of a PDH locking scheme is outlined as following:

1. A laser beam is passed through a Pockels cell to modulate its frequency at a frequency small relative to the $\Delta\nu_{fsr}$ free spectral range of the optical cavity. My cavity has length $L = 15$cm, $\Delta\nu_{fsr} = c/2L \approx 1GHz$. Hence my EOM at $\approx 38MHz$, as discussed in the EOM section, is appropriate here. Note that EOM only modulates light of polarisation parrallel to the EOM's oscillating electric field. In my case horizontal polarisation w.r.t the optical table is used. (i.e. a quarter-wave-plate and half-wave-plate pair is used to clean up laser beam polarisation to be horizontal before entering the EOM)

2. The beam goes do the cavity. An optical isolator is used to pick up reflected beam from the cavity. I used a quarter wave plate and a polarising beam splitter.

3. Referring to Fig **??**, if length of the cavity is set to one side of these resonances, but near enough that some light gets transmitted, then a small change in laser frequency would produce a proportional change in the transmitted intensity. Equivalently there will be a change in the reflected beam intensity. Optimising for minimum reflection has the following benefits over optimising for maximum transmission:

   (a) Effect of laser frequency drift and intensity drift is decoupled due to obvious reason.

   (b) The frequency of frequency drift that can be corrected for is not restricted by the cavity response time. As when measuring transmission intensity, it takes time for the fields in the cavity to re-equilibrate whenever there is frequency drift. Whereas measuring reflected intensity circumvents this issue.

4. In order to have a control loop that locks the laser to a certain frequency, we need an error signal that tells provides a $\pm$ sign for the feedback signal on top of the amplitude as given by the last point. To obtain an error signal, the EOM comes into play in that it varies the laser frequency slightly (small w.r.t cavity free spectral range as discussed earlier), indicating whether laser frequency is currently on the right side or left side of resonance.

5. What's left is to produce an error signal by using a fast photodiode (fast enough to discern MHz level modulation) to read beam intensity coming from the optical isolator. Signal from the fast photodiode is then processed as shown in Fig **??**. For details on how the multiplicative mixer and the phase shifter helps output an error signal, see [**?**].

6. Caution: Note that wave plates and polarising beam splitter are wavelength rated

5.1 Measuring Water Vapor concentration Using the PDH Setup

Discuss: I am not doing the laser lock, but instead locking the cavity length to the frequency stable 738nm laser and measuring intensity drift due to water vapor change (as 935nm is readily absorbed by water) of the frequency-stable 935nm laser to indicate a water vapor concentration drift.

6 Other Discussions

6.1 General Optical Lab Techniques and devices

1. discuss various optical devices on the optical table (reference QDevice content): wave plates, polarisers, etc -¿ briefly on how they work + how to use them in the lab

(a) wave plates, polarisers

(b) AR coating

(c) laser power meter: wavelength selection -¿ response curve

(d) lens cleaning

(e) DIY relevant tools in the lab. exmaples: strips of paper for ray tracing

(f) electronics stuff: soldering, PCB board inspection

7 Conclusion

1. what have I learn: about theory and experimental physics lab
2. what have I done
3. possible follow up investigations / improvement on existing current

8 Acknowledgements

thank the lab people! Jaren, Muyoung, Dzmitry, Nigel, Kowei

9 Appendix

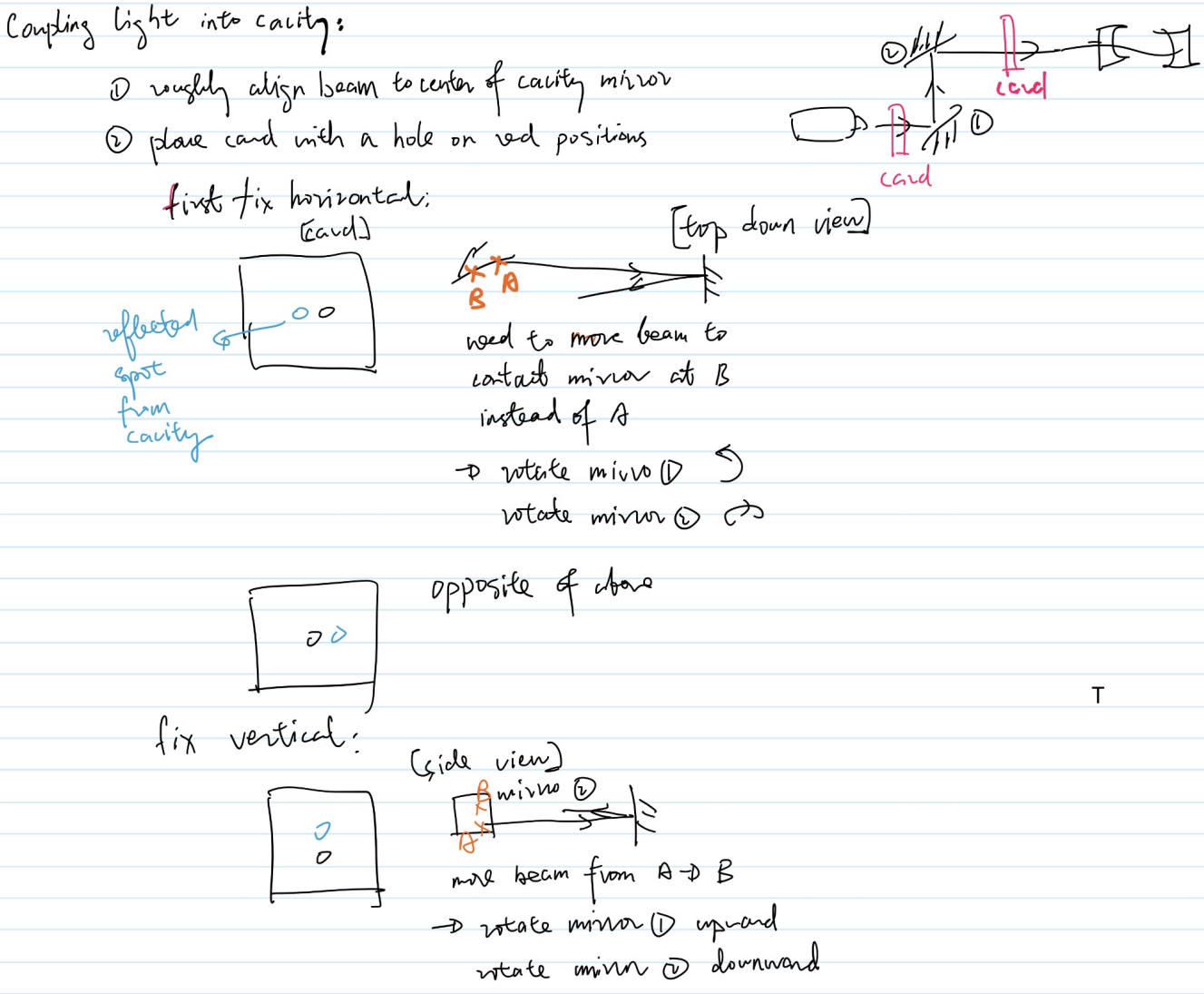

Figure 15: Laser beam - Cavity Coupling Note

```
#%%
import matplotlib.pyplot as plt
import numpy as np
import glob
from PIL import Image
# Read me:
# 1. naming the .pgm documents:
    # background: 1.pgm -> so that background reading is first in
       the file
    # others: z-position+unit.pgm -> eg: 15cm.pgm for reading at
       15cm on z-axis
    # this is such that glob.glob works properly
```

```
# 2. the following lines of code are instrument specific: -> do
   change parameters accordingly
    # CCD camera: hBeamDiameter = (rBound - lBound)*3.75*10**(-6)
       (in microns)

##########################################################################
# helper functions:

# Read the pgm file into a nparray; input .pgm file path, output
   the .pgm image as nparray
def read_pgmf(filePath):
    return plt.imread(open(filePath,'rb'))

# image nparray -> horizontal beam radius in mm
def getHorizontalBeamRadius(image_array, image_array_bg):
    column_image = image_array.sum(axis=0) # this sums up all
       rows of camData
    column_image_bg = image_array_bg.sum(axis=0) # this sums up
       all rows of background camData
    column_image_wtbg = column_image - column_image_bg # subtract
       background reading from camData
    column_image_normalised = [] #normalise
    for i in range(len(column_image_wtbg)):
        column_image_normalised.append(column_image_wtbg[i] /
           column_image_wtbg.max())
    threshold = 1/(np.e**2)
    aboveThreshold = []
    for i in range(len(column_image)):
        if column_image[i] >= threshold:
            aboveThreshold.append(i)
    aboveThreshold = np.array(aboveThreshold)

    #this givs a list of the elements in column_image above
       threshold)
    lBound = aboveThreshold.min()
    rBound = aboveThreshold.max()
    pixelSize = 3.75*10**(-6) #meter
    hBeamRadius = (rBound - lBound)*pixelSize/2
    #debug:
```

```
    print("beam left bound is: "+ str(lBound))
    print("beam right bound is: "+ str(rBound))
    print('Horizontal beam radius',':', hBeamRadius*10**(3), 'mm')
    plt.plot(column_image_normalised)

    return hBeamRadius

def getVerticalBeamRadius(image_array, image_array_bg):
    row_image = image_array.sum(axis=1) # this sums up all
       columns in camData
    row_image_bg = image_array_bg.sum(axis=1) # this sums up all
       columns in backgroung
    row_image_wtbg = row_image - row_image_bg
    row_image_normalised = [] #normalise
    for i in range(len(row_image_wtbg)):
        row_image_normalised.append(row_image_wtbg[i] /
           row_image_wtbg.max())
    threshold = 1/(np.e**2)
    aboveThreshold = []
    for i in range(len(row_image)):
        if row_image[i] >= threshold:
            aboveThreshold.append(i)
    aboveThreshold = np.array(aboveThreshold)

    bBound = aboveThreshold.min()
    tBound = aboveThreshold.max()
    pixelSize = 3.75*10**(-6) #meter
    vBeamRadius = (tBound - bBound)*pixelSize/2
    #debug:
    print("beam top bound is: "+ str(tBound))
    print("beam bottom bound is: "+ str(bBound))
    print('Vertical beam radius',':', vBeamRadius*10**(3), 'mm')
    plt.plot(row_image_normalised)

    return vBeamRadius


##########################################################################
# the main code:
```

```
# Directory where .pgm files are stored
# directory = input("input data file directory")
# directory = "/Volumes/dzmitrylab/zhonglin/10MarCamera"
directory = "/Volumes/dzmitrylab/zhonglin/10MarCamera2"
pixelSize = 3.75*10**(-6) #meter

# Find all .pgm files in the directory
pgm_files = glob.glob(directory + '/*.pgm')
pgm_files.sort(key = lambda s: len(s))

# read background first, then delele background from the list of
   files
image_array_bg = read_pgmf(pgm_files[0])
print("-> background array: ")
print(image_array_bg)
print("CCD has number of rows: " + str(len(image_array_bg)))
print("CCD has horizonal length: " +
   str(len(image_array_bg)*pixelSize))
print("CCD has number of columns " + str(len(image_array_bg[0])))
print("CCD has vertical length: " +
   str(len(image_array_bg[0])*pixelSize))
pgm_files=pgm_files[1:]

print("-> Finished reading files, here are all the files: ")
print(pgm_files)
print("-> Now import files: ")
# Define empty lists to hold the image arrays and z positions
image_arrays = [] # contain nparray
z_positions = []  # contain integer
# Read all .pgm files and process them one by one
for file_path in pgm_files:
    # Load the .pgm file using the Pillow library
    pgm_image = Image.open(file_path)

    # Convert the image to a numpy array and append it to the list
    image_array = np.array(pgm_image)
    image_arrays.append(image_array)
```

```
    zPosition = int(file_path[-8]+file_path[-7])
    z_positions.append(zPosition)

    # Close the image file to free up resources
    pgm_image.close()

print("-> Finished importing files, here are all the image data: ")
for image_array in image_arrays:
    print(image_array)
print("-> now start processing image data: ")

horizontal_radius = []
vertical_radius = []
for i in range(len(image_arrays)):
    z_position = z_positions[i]
    print("data from " + str(z_position) +"cm :")
    # process each image_array to get horizontal and vertical
       beam radius
    hBeamRadius = getHorizontalBeamRadius(image_arrays[i],
       image_array_bg)
    horizontal_radius.append(hBeamRadius)
    vBeamRadius = getVerticalBeamRadius(image_arrays[i],
       image_array_bg)
    vertical_radius.append(vBeamRadius)
    print("-> image minused bg " + str(i) + " :")
    print(image_array)
print("-> Finished image processing, now plot graphs: ")

# Now plot the w(z) graph for horizontal and vertical directions
plt.plot(z_positions,horizontal_radius, 'o', label='horizontal
   radius(mm) - z(cm)')
plt.plot(z_positions,vertical_radius, 'o', label='vertical
   radius(mm) - z(cm)')
plt.legend()
plt.xlabel('detector position (cm)')
plt.ylabel('beam radius (mm)')
```

```
plt.show
print('-> all done')

#%%
```

Listing 1: Gaussian Beam Characterisation code with a CCD camera

```
import serial
import serial.tools.list_ports
from timeit import default_timer as timer
from time import sleep
import matplotlib.pyplot as plt
from numpy import std
import csv

# self defined function for this script
#------------------------------------------------
def isfloat(num):
    try:
        float(num)
        return True
    except ValueError:
        return False

# set up serial port communication
#------------------------------------------------
def get_ports():
    ports = serial.tools.list_ports.comports()
    return ports

def findArduino(portsFound):

    commPort = 'None'
    numConnection = len(portsFound)

    for i in range(0,numConnection):
        port = foundPorts[i]
        strPort = str(port)

        if 'usbmodem' in strPort: #it is understood that
```

```
            'usbmodem1101' can be used to identify the port of
            interest
             splitPort = strPort.split(' ')
             commPort = splitPort[0]
    return commPort

foundPorts = get_ports()
connectPort = findArduino(foundPorts)

if connectPort != 'None':
    ser = serial.Serial(connectPort,baudrate=115200,timeout=1)
    print('Connected to ' + connectPort)

else:
    print('Connection Issue!')


if ser.is_open == True:
    ser.close()
    print("Port was already open, closing.")

ser.open()
print("Opening port connection")

# collect data
#-----------------------------------------------
# ask for input: current opsition of knife
position = input('What is the current position of knife in unit
   (mm)?\n')
if isfloat(position):
    position = float(position) / 1000


y = []
N = 200 # number of samples points per scan
waitTime = 2 # wait time in between scans
n = 3 # number of scans
testrun1 = ser.readline()
testrun2 = ser.readline()
```

```
testrun3 = ser.readline() # for unknown reasons, first three
   readlin() calls take long time
for j in range(0,n-1):
    for i in range(0,N):
        line = ser.readline()
        if line:
            string = line.decode()
            num = int(string)
            y.append(num/1024*5 )
    sleep(waitTime)

print("finish data collection")
ser.close()
print("serial port closed")

# process data and write into csv
#-----------------------------------------------
voltage = sum(y) / (N*n)
stdev = std(y)
row = [position, voltage,stdev]
# please input dataFile position here!!!
dataFile = '/Users/JohnnyLin/Desktop/Dzmitry_Lab
   /UROPS/cavity/BeamWidthOfFiberBeam/measurement1.csv'
with open(dataFile,'a') as f:
    writer = csv.writer(f)
    writer.writerow(row)

print('all done')
```

Listing 2: Knife Edge Measurement Data Collection

```
import matplotlib.pyplot as plt
from scipy.optimize import curve_fit
from scipy import special
import numpy as np
import pandas as pd
import math
import csv

# open data file
```

```
#-----------------------------------------------------------------
fileLocation = input('Key in csv file path\n')
data = pd.read_csv(fileLocation)
x = data.position[2:].to_numpy()
y = data.voltage[2:].to_numpy()

x = [float(i) for i in x] # make it a list of float
y = [float(i) for i in y] # make it a list of float


# curve fitting
#-----------------------------------------------------------------
def power(hi,po,p,w,h0):
    pi = po + p/2 * special.erfc((h0-hi)/(w/math.sqrt(2)))
    return pi
#make sure hi-h0>0

# guessed parameters p0 = [po,p,w,h0]
# po = background power reading,
# p = maximum power reading,
# w = guessed beam width, here it's set to 0.5mm
# h0 = position of half maximum power reading -> center of beam
   position
po = float(data.voltage[1])
p = float(max(y))
w = float(0.0005)
h0 = float(input('What is the position of half maximum in unit
   (m): guess base on data (remember to consider background
   power)\n'))
# print(po,p,w,h0)
# print(type(po),type(p),type(w),type(h0))
# here h0 is eyeballed, can write a code for it instead in the
   future

popt,pcov = curve_fit(power,x,y,p0 =[po,p,w,h0])
# last used p0 values: 0.288,21.5,0.0005,0.0092

print('finished fitting, plotting graph')
```

```
# plot graphs
#-----------------------------------------------------------------------
plt.plot(x,y, 'o', label='data')

xdata = np.linspace(min(x),max(x),num=1000)
plt.plot(xdata, power(xdata, *popt), 'r-', label='Fitting')
plt.legend()
plt.title('Fitting')
plt.xlabel('knife position (m)')
plt.ylabel('voltage (V)')

print(popt,pcov)
print("beam width =",popt[2],"(m)") # beam fitted w value
beamStdev = np.sqrt(np.diag(pcov))[2]
print("stdev =",'%.20f' % beamStdev,"(m)") # print one standard
    deviation in decimal format

print('all done')

#this line blocks the script from proceeding, hence it's moved to
    the end
plt.show()
```

Listing 3: Knife Edge Measurement Data Processing