Abstract

In order to investigate the phenomenon of *electromagnetic induced transparency* (EIT), several preliminary measurements need to be performed.
In a first step, the lasing threshold is determined to be $I_{\mathrm{LT}} = (83.4 \pm 0.1)\,\mathrm{mA}$. Above this threshold, the laser can be used. In a first step, a rubidium sample, consisting out of ^{85}Rb and ^{87}Rb, is analysed. For that, three different spectroscopy methods are used, namely absorption, fluorescence and saturation. With each of those methods, the hyperfine transitions between the levels $5^2\mathrm{S}_{1/2}$ and $5^2\mathrm{P}_{1/2}$ of rubidium can be determined. Since there is no absolute scale available, the peaks are given with reference to the lowest one. For simplicity, the peaks are labelled with T_1 to T_8, ordered after frequency. The transition values found with saturation spectroscopy are: $T_1 = (0 \pm 45)\,\mathrm{MHz}$, $T_2 = (796 \pm 45)\,\mathrm{MHz}$, $T_3 = (1506 \pm 45)\,\mathrm{MHz}$, $T_4 = (1870 \pm 46)\,\mathrm{MHz}$, $T_5 = (4532 \pm 46)\,\mathrm{MHz}$, $T_6 = (4903 \pm 45)\,\mathrm{MHz}$, $T_7 = (6829 \pm 45)\,\mathrm{MHz}$ and $T_8 = (7645 \pm 46)\,\mathrm{MHz}$. All those values are in good accordance with literature.

With these transitions, further physical constant, like the hyperfine constant of both rubidium isotopes, can be determined. The hyperfine constants determined with the saturation spectroscopy are: $A_{87} = h \cdot (398 \pm 16)\,\mathrm{MHz}$, $A_{87} = h \cdot (408 \pm 16)\,\mathrm{MHz}$, $A_{85} = h \cdot (122 \pm 11)\,\mathrm{MHz}$ and $A_{85} = h \cdot (123 \pm 11)\,\mathrm{MHz}$. All those values are in good accordance with literature.
Also the mixture of the rubidium isotopes can be gathered by determining the peak heights. The abundance of ^{87}Rb is calculated with different peaks and is found to be $30 \pm 4\%$ using the first and third peak in saturation spectroscopy.

The spectroscopy measurement is needed, since the laser should be locked onto one specific transition frequency for the EIT experiment. The desired frequency can be locked. In order to obtain knowledge about the stability of the locking, a frequency stability measurement is performed. The measurement showed, that the laser frequency is stable over the whole measurement time of 10 minutes.

The last preliminary measurement for the EIT experiment, is the determination of the beat note. Here, two overlapping beams realize a different frequency shift from an AOM each, leading to a beat note. With the beat note measurement, a linear dependency between the applied shift at the AOMs and the measured beat note can be found.

Contents

List of Tables

List of Figures

1 Introduction

Electromagnetically Induced Transparency (EIT) is a quantum interference effect, that renders an otherwise absorbing medium transparent to a probe laser in the presence of a coupling field, provided by a pump laser [1]. By this manipulation of the optical properties of the medium, it is possible to produce light with very small group velocities. Specifically, it is possible to produce stopped light [2]. This is achieved by turning off the pump laser that prepares the medium, which then "traps" the light of the probe beam inside the medium. Stopped light has many exciting applications, such as quantum memory, which is used for quantum information processing [3] and highly sensitive interferometers [4]. Further applications of EIT include laser cooling of atoms [5].

In this report, preliminary measurements for EIT are performed. In the first week, the lasing threshold is determined, followed by spectroscopic measurements of rubidium. For that, three different spectroscopy methods are used, specifically absorption spectroscopy, fluorescence spectroscopy and saturation spectroscopy. From these measurements, several physical quantities are determined. Those are the hyperfine transitions of rubidium, the composition of the Rb isotopes ^{85}Rb and ^{87}Rb in the sample and the hyperfine constants of ^{85}Rb and ^{87}Rb.

In the second week, the laser frequency is locked, using PID controllers. The frequency stability of this lock-in is then determined. Furthermore, the setup is changed in preparation of the EIT measurement. For that, the laser is splitted into a pump and probe beam, which are then modulated to two different frequencies. After recombination of the beams, a beat note measurement is performed.

2 Theory

In this section the basic concepts for the hyperfine structure are introduced, which is analysed for ^{85}Rb and ^{87}Rb in this experiment. Also, the working principle of a laser in general is explained, with special regards to grating stabilized laser heads. In addition to that, all elements in the optical path are explained briefly. If not stated otherwise, the sources used are *Atomic Physics* by Foot [5] and *Laser Spectroscopy* by Demtröder [6].

2.1 Different atom structures

Electrons which orbit a nucleus move accordingly to the coulomb potential created by the nucleus. Calculating these orbits classically shows, that electrons can only follow discrete orbits. Due to the existence of several orbits with different energies, energy differences can be observed as spectral lines. Looking closer at the electromagnetic interaction between the orbital angular momentum **L** and spin **S** of the electrons (coupling into the total angular momentum **J**), it can be found that the energy levels are slightly shifted and splitted due to the orientation of the spin. The resulting splitting is called *fine structure*, where the splitting is small compared to the previous energy difference. Since the nucleus obtains a nuclear spin **I**, which interacts with the total angular momentum **J** of the electrons (coupling into the total atomic angular momentum **F**), there are further splittings, which are called *hyperfine structure*.

2.1.1 Fine structure

The coupling of angular momentum **L** and spin **S** of electrons orbiting a nucleus, leads to **J** (total angular momentum) with its new quantum number J. This is called LS-coupling and is given by:

$$\mathbf{J} = \mathbf{L} + \mathbf{S}, \tag{1}$$

$$|L - S| \leq J \leq |L + S|. \tag{2}$$

Here, J can take only values in integer steps. The total magnetic moment of the electrons can be calculated with:

$$\mu_J = g_J \frac{\mu_B}{\hbar} \mathbf{J}, \tag{3}$$

with $\mu_B = \frac{e\hbar}{2m_e}$ the Bohr magneton and g_J the Landé factor:

$$g_J = 1 + \frac{J(J+1) + S(S+1) - L(L+1)}{2J(J+1)}. \tag{4}$$

The exact configuration of an atom in the LS-coupling is often described using term symbols. These are generally written as $n^{2S+1}L_J$, where n is the principal quantum number and L the orbital angular momentum quantum number, written using spectroscopic notation. In this notation L is expressed in capital letters, with S for $L = 0$ or P for $L = 1$.

2.1.2 Hyperfine structure

To obtain the hyperfine structure, the previous total angular momentum **J** of the electrons is coupled with the nuclear spin **I** of the nucleus. This coupling results in the total atomic angular momentum **F**, with its quantum number F and is calculated with:

$$\mathbf{F} = \mathbf{I} + \mathbf{J}, \tag{5}$$

$$|I - J| \leq F \leq |I + J|. \tag{6}$$

As before, also F can only realize values in integer steps.

The splitting of the energy levels in the hyperfine structure is given by:

$$\Delta E_{\mathrm{HFS}} = \frac{A}{2}\left[F(F+1) - I(I+1) - J(J+1)\right], \tag{7}$$

with A the hyperfine constant. With Equation 7, one can then determine the energy difference of states with the same I and J and neighbouring F, which means $F_2 = F_1 + 1$. This difference is given by:

$$\Delta E_{F_2=F_1+1} = A(F_1+1), \tag{8}$$

with $F_1 < F_2$. In order to estimate A from a spectral measurement, the energy difference can be rewritten in terms of ν, the frequency of the corresponding emitted photon:

$$\Delta E = h\Delta\nu. \tag{9}$$

Here, h is the Planck constant. Thus, rewriting Equation 7 and 8 leads to:

$$A = \frac{h\Delta\nu}{F_1+1}. \tag{10}$$

With Equation 10 the hyperfine constant A can be calculated by measuring the frequency $\Delta\nu$ for a transition with constant I and J and neighbouring F.

2.1.3 Rubidium

In this experiment, the hyperfine transitions lines between the $5^2\mathrm{S}_{1/2}$ and $5^2\mathrm{P}_{1/2}$ states of ^{85}Rb and ^{87}Rb are of interest. Both states obtain a spin of $S = \frac{1}{2}$ and total angular momentum $J = \frac{1}{2}$. The transitions of interest are between $L = 0$ and $L = 1$. The ground state of ^{85}Rb has a nuclear spin of $I = \frac{5}{2}$. Therefore, F can only take the values $F = 2$ or $F = 3$ (according to Equation 6) for both states. The ground state of ^{87}Rb obtains a nuclear spin of $I = \frac{3}{2}$. Thus, only $F = 1$ and $F = 2$ are possible. In Figure 33 and Figure 34 in the Appendix, the hyperfine splitting of ^{85}Rb and ^{87}Rb can be seen. The fine-structure transition between the $5^2\mathrm{S}_{1/2}$ and $5^2\mathrm{P}_{1/2}$ states is called D_1 line and differs slightly for both isotopes [7, 8].

The aim of this experiment is to record the spectrum of ^{85}Rb and ^{87}Rb isotopes caused by hyperfine transitions. Possible transitions need to fulfill $\Delta L = \pm 1$ due to the selection rules. The expected transitions between the $5^2\mathrm{S}_{1/2}$ and $5^2\mathrm{P}_{1/2}$ states of the two rubidium isotopes, along with their differences to the $^{87}\mathrm{Rb}^{F=1}_{F=2}$ (see Equation 11) can be found in Table 1. This transition is used as comparison, since it has the smallest energy difference. Since only transitions between $5^2\mathrm{S}_{1/2}$ and $5^2\mathrm{P}_{1/2}$ are expected, in the following transitions are denoted by:

$$^{A}\mathrm{Rb}^{F_{L=P}}_{F_{L=S}}, \tag{11}$$

with A the mass number and $F_{L=P}$ and $F_{L=S}$ indicating the exact hyperfine states of the upper $5^2\mathrm{P}_{1/2}$ and lower $5^2\mathrm{S}_{1/2}$ states, respectively. An example for this notation is $^{85}\mathrm{Rb}^{F=2}_{F=3}$, referring to the transition from the ^{85}Rb state $5^2\mathrm{P}_{1/2}$ with $F = 2$ to the $5^2\mathrm{S}_{1/2}$ state with $F = 3$.

Tab. 1: Here, literature values for the hyperfine transitions of ^{85}Rb and ^{87}Rb are displayed. The transitions are labelled according to Equation 11. The frequency differences compared to the $^{87}\mathrm{Rb}_{F=2}^{F=1}$ are given as frequency offset. In addition to that, also the absolute values are given [7, 8].

Transition	Frequency offset [MHz]	Absolute value [THz]
$^{87}\mathrm{Rb}_{F=2}^{F=1}$	0	377.10439
$^{87}\mathrm{Rb}_{F=2}^{F=2}$	814.5	377.10521
$^{85}\mathrm{Rb}_{F=3}^{F=2}$	1518.56	377.10591
$^{85}\mathrm{Rb}_{F=3}^{F=3}$	1880.15	377.10627
$^{85}\mathrm{Rb}_{F=2}^{F=2}$	4554.3	377.10895
$^{85}\mathrm{Rb}_{F=2}^{F=3}$	4915.88	377.10931
$^{87}\mathrm{Rb}_{F=1}^{F=1}$	6834.68	377.11123
$^{87}\mathrm{Rb}_{F=1}^{F=2}$	7649.18	377.11204

2.2 Spectroscopy methods

In order to obtain knowledge about the hyperfine structure of rubidium, or an atom in general, spectroscopy methods are used. In the following, three different spectroscopy methods are explained. Namely absorption, fluorescence and saturation spectroscopy. Besides their differences, they all have in common, that they measure photons via a photodiode, where the intensity of the measured peaks is proportional to the probability of the transition.
Photons are produced with a laser and directed onto the rubidium cell. These photons then excite the rubidium to a higher atomic state, if they have an appropriate frequency. This state then spontaneously decays by emitting a photon with the exact same frequency as the incoming photon. This decay is isotropic. That is why the beam after the cell significantly lacks the transition frequency. If there is already an excited atom, which is hit by a photon with the right frequency, the atom can de-excite via stimulated emission. This de-excitation produces a photon radiating in the same direction as the incoming photon.
In the following subsections, the three methods are explained. If not stated otherwise, this section is based on *Laser Spectroscopy* by Demtröder [6].

2.2.1 Fluorescence spectroscopy

In fluorescence spectroscopy, photons from spontaneous de-excitations are measured. This is achieved by putting a photodiode next to the sample. Hereby, the photodiode needs to be placed perpendicular to the incoming laser beam, in order to avoid the measurement of the laser photons. The recorded spectrum of the sample will show peaks at its spectral lines.

2.2.2 Absorption spectroscopy

In case of absorption spectroscopy, the photodiode is placed in the laser beam path behind the sample. Since the laser beam covers a range of frequencies in this experiment periodically (see subsubsection 2.3.2), there are some photons which excite the sample atoms to higher states and other photons, which just pass through the sample. In the case that the photons excite the sample atoms, the beam intensity behind the sample is smaller for this certain frequency. Since the de-excitation of the sample atoms happens through the isotropic spontaneous emission,

only few photons radiate in the same direction as the laser beam. Thus, at certain frequencies, corresponding to the spectral lines, less photons will be detected. Therefore, the recorded spectrum will show dips at the spectral lines.

2.2.3 Saturation spectroscopy

There are different broadening effects, affecting the width of spectral lines. One of those effects is the Doppler broadening, caused by the thermal movement of the atoms in the sample. If those atoms move parallel to the propagation direction of the laser photons, the frequency of these photons is Doppler shifted. Therefore, also laser frequencies slightly below or above the actual transition frequency are able to excite atoms, which are moving parallel to the laser beam. In total, this causes a broadening of the measured line width due to thermal movements of the atoms. The distribution of velocity of the atoms will follow a Maxwell-Boltzmann distribution, since we are looking only in beam direction, this broadening will be of a Gaussian shape.

In order to overcome this broadening effect at room temperature, saturation spectroscopy is used. Otherwise cooling the sample to millikelvin temperatures would be necessary to overcome the thermal Doppler broadening. The idea of saturation spectroscopy is, that a strong pump beam (high laser intensity) is directed onto the sample from one direction. From the opposite direction, a weaker probe beam, with the exact same frequency, but less intensity, is directed onto the sample. This is done, by using a mirror and a neutral density filter (see Figure 5). The weaker probe beam is then detected by a photodiode.
There are three different cases: first, the laser frequency is slightly below the transition energy. In this case, the pump beam will excite atoms moving towards the pump beam laser source, say with a velocity v, due to the Doppler shift. Alongside this, the probe beam will excite atoms moving in the exact opposite direction, now with $-v$, since these are Doppler shifted towards the probe beam. Therefore, the pump and probe beam excite different atoms. Like for the absorption spectrum discussed before, this will cause a lower intensity for the recorded probe beam. In the second case, the laser frequency is slightly above the transition energy, which leads to the same result. The only difference is, that the pump beam now excites atoms with $-v$ and the probe beam those with v.

In the last case, the frequency of the pump beam matches the transition energy perfectly. In this case, only atoms with $v = 0\,\frac{\mathrm{m}}{\mathrm{s}}$ are excited by the pump beam, since all atoms moving along the laser propagation are Doppler shifted. Since $v = 0\,\frac{\mathrm{m}}{\mathrm{s}}$, the probe beam can excite the same atoms as the pump beam and vice versa. Because the probe beam is much stronger, it will cause many atoms to be in the excited state. The weaker probe beam, however, meets these atoms and will cause stimulated emission. Therefore, the intensity of the probe beam at exactly the transition frequency is higher than for frequencies slightly apart. Thus, the detected spectrum will show a peak inside the absorption dip. Since this peak arises from atoms with $v = 0\,\frac{\mathrm{m}}{\mathrm{s}}$ (in beam direction), the peak is not affected by Doppler broadening effects. The line width of this peak results from the natural line width, which is influenced by the even smaller Lamb-dip. Hereby, the peak follows a Lorentzian shape:

$$L(\nu) = \frac{A}{1 + \left(\frac{\nu - \nu_{\mathrm{mean}}}{\gamma}\right)^2} + L_0, \tag{12}$$

with A the amplitude, ν_{mean} the mean value of the curve, γ the width of the distribution and L_0 its offset. An example for the spectral measurement using saturation spectroscopy can be seen in Figure 1.
Besides the general idea of saturation spectroscopy, there is an additional phenomenon, called cross-over, which occurs in saturation spectroscopy, if two transition frequencies ν_1 and ν_2 are

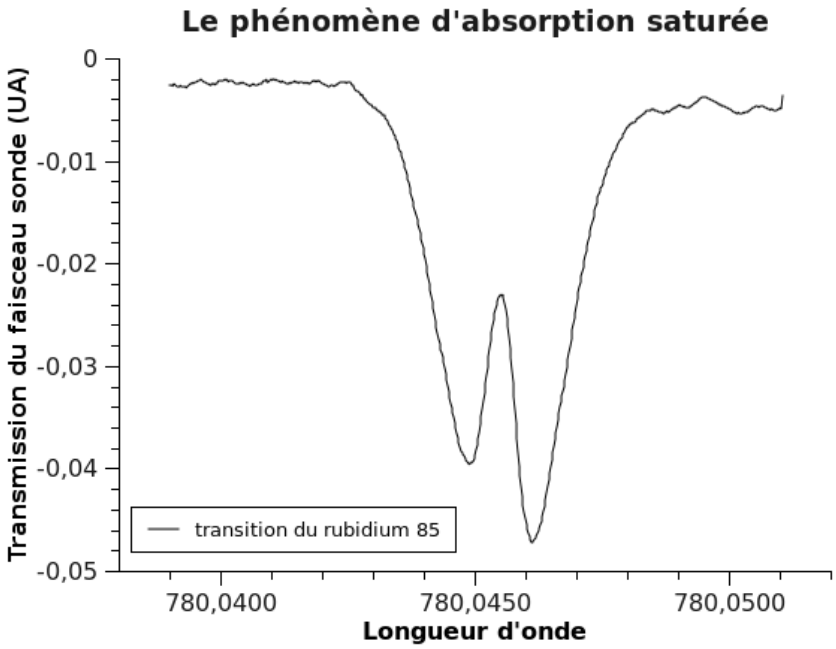


Fig. 1: Example spectrum for saturation spectroscopy, with the peak sharp peak inside the absorption dip, which arises from stimulated emission from the probe beam [9].

close together. If the laser is tuned at ν_1 or ω_2, a saturation peak appears, as explained before, from the sample atoms with $v = 0\,\frac{\mathrm{m}}{\mathrm{s}}$. If the laser frequency is at $\nu = (\nu_1 + \nu_2)/2$, a group of atoms with v_{cross} is saturated. The atoms in this velocity class are addressed by pump and probe beam. Both frequencies ν_1 and ν_2 excite their respective transition. This produces an overlap leading to this additional peak in the middle of the single ν_1 and ν_2 peaks. This additional peak can make the whole saturation spectroscopy harder, since more peaks appear. Nevertheless, it can also help to identify double peaks and therefore help to identify the whole spectrum.

2.3 Laser

In order to run this experiment, photons are needed which are radiated onto the sample. Since the setup requires high precision of the beam, frequency stability and frequency manipulation, a laser is needed. For better frequency manipulation, a grating can be placed in front of the laser. In this experiment, such a grating laser is used. In the following, the basic concepts of lasers, as well as grating lasers, are introduced. If not stated otherwise, this section is based on *Atomic Physics* by Foot [5] and *Laser Spectroscopy* by Demtröder [6].

2.3.1 Basic concept

A laser, **L**ight **A**mplification by **S**timulated **E**mission of **R**adiation, is used to produce coherent light at a certain wavelength (monochromatic) in the same phase. Lasers are build out of three main components, an active laser medium, an energy pump and an optical resonator.

If an atom of the active medium is in an excited state, it can decay via spontaneous emission into a lower state, by emitting a photon with exactly the energy difference of the energy levels. Therefore, this photon obtains a certain wavelength. If such a photon hits other excited atoms, it will cause stimulated emission, which produces an additional photon with the same wavelength, polarization, phase and direction. These two photons then can interact again with other excited atoms. This leads to many photons with the exact same parameters and therefore

produces amplified light.

In order to keep the laser running, there needs to be an appropriate amount of atoms in an exited state. This is also called population inversion and requires an energy pump. Without an energy pump, the previous emitted photons will only excite other atoms, which will not lead to stimulated emission. Only with population inversion, the rate of stimulated emission is higher than the one for absorption and therefore amplification is achieved. The minimal power needed to obtain this inversion, is called *lasing threshold*.

The last important component, the optical resonator, is used to create standing waves, which leads to a selection of a wavelength, due to constructive and destructive interference. It is also used to focus the light and redirect the photons on the active medium, in order to have a high rate of stimulated emission. This cavity is build out of mirrors, one with a reflectivity of 100% and one which is slightly transparent. This transparent mirror then lets some photons pass, which form the laser beam.

2.3.2 Grating laser head

With the basic setup, only one specific wavelength can be produced, depending on the optical resonator. For spectroscopy purposes, a range of wavelengths is needed. In order to achieve this, a diode laser with a grating stabilized head, the `DL pro 020011` by Toptica, is used. The following explanation is based on the manuals for the laser controller [10] and its head [11], for general knowledge about the working of a gratin,g *Atomic Physics* by Foot [5] and *Laser Spectroscopy* by Demtröder [6] are used.

In this experiment, a laser diode is the source of light. Such a laser diode has a linewidth of around 100 MHz. Its emission frequency can be tuned by adjusting either the current applied to the diode, or the temperature. Adjusting one of these two parameters at a time, while the other is held constant, can lead to (large) discontinuous *hops* in the emitted frequency. Such hops are called mode hops. The range where no such hops occur and the frequency can be adjusted continuously, is called *mode-hop-free range*.

For spectroscopy measurements, this linewidth should be chosen as narrow as possible without mode-hops. Here, the grating laser head comes into play. The grating improves the linewidth, as well as the mode-hop-free range. The grating functions as follows: The light emitted by the front end of the laser diode is collimated onto a reflective grating (see Figure 2). The grating is positioned in such a way, that the first diffraction order of the grating is reflected back towards the laser diode. This is called Littrow-setup and can be seen as the blue beam hitting the grating. The back reflected beam is again collimated and focused onto the laser diode. Since the feedback from the grating is higher than from the front end of the laser diode, a new resonator forms between the grating and the back end of the laser diode. This new resonator has a linewidth of around 1 MHz, which is significantly smaller than for the laser diode itself. In order to get a continuous range of frequencies, the grating can be moved with the help of a piezo element.

The laser controller offers different settings, in order to obtain an appropriate mode-hop-free frequency range for the experiment. To scan over a certain range, a changing voltage can be applied to the piezo element. This voltage can be set with `scan offset` and `scan amplitude`. Here, `scan offset` is the voltage baseline applied to the piezo element and `scan amplitude` is the maximum voltage added to the baseline. The voltage is continuously adjusted between the minimum and maximum voltage, this change can be set to a sinusoidal or triangular shape.

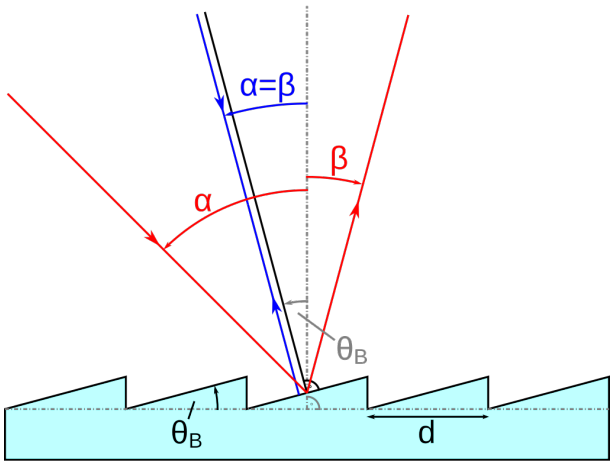


Fig. 2: Sketch for a grating, placed in front of the laser diode. The grating consists of many small ramps with length d at a certain angle θ_B. If the grating is placed in a way that the beam is reflected back at an angle θ_B (blue ray), one calls this the Littrow setup. The grating is used to increase the size of the optical resonator, in order to adjust the frequency of the emitted laser beam [12].

The frequency of this oscillation can be set with `scan frequency`. Here the triangular shape is used.
Also the laser diode current can be set with `set current`, with its respective ramp slope, called `feed forward`. The ramp used for this current change is proportional to the piezo scan ramp. For a rather rough setting, also the temperature can be varied with the `temperature` parameter. All these settings need to be adjusted precisely, in order to obtain a mode-hop-free range in the wanted frequency window. This frequency range depends on the laser diode current, the piezo offset and amplitude, as well as on the temperature. For a good setting, all parameters influence each other and need to be adjusted accordingly.

2.4 Elements in the optical path

In order to set up this experiment, several different optical elements are used. In the following, the used elements are introduced briefly. This section is again based on [5] and [6] if not stated otherwise.

2.4.1 Optical isolator

An optical isolator is used, in order to block the laser beam from re-entering the laser cavity. This is needed to prevent unwanted optical feedback from the resonator. Therefore, this isolator is placed right behind the output of the laser. The principle working of such an optical isolator can be seen in Figure 35 in the attachment. The isolator uses two polarization filters and a Faraday rotator. This setup enables, that light is only transmitted into one direction [6].

2.4.2 Polarizing beam splitter

A polarizing beam splitter splits the laser beam into a transmitted and reflected beam. Thereby, the beam is splitted into two beams of orthogonal polarizations. This is achieved by gluing two prisms together, which consist out of a birefringent material.

2.4.3 Wave-plates

A wave plate changes the polarization of an incoming light beam. Wave plates consist out of birefringent materials, which obtain different refractive indices for the incoming light beam, depending on its polarization. Therefore, different polarizations of the incoming beam experience different phase shifts. These shifts can be varied by changing the thickness of the plates, as well as its orientation in the beam path. Two very common wave plates are the half-wave plate (HWP) and the quarter-wave plate (QWP). The HWP shifts the phase of the incoming beam by π, which corresponds to half of the wavelength. This shift leads to the change of the polarization axis for linear polarized light or to the change of handedness for circular polarized light. A QWP shifts the phase by $\pi/2$, corresponding to a quarter of a wavelength. This shift converts linear polarized light into circular polarized light and vice versa. Since the phase shift depends on the wavelength, a given wave-plate only works in a certain wavelength range.

2.4.4 Neutral density filter

A neutral density filter (ND filter) is used to reduce the intensity of all incoming wavelengths equally. Therefore, such a ND filter can prevent a photodiode from saturating without distorting the spectrum. The used ND filter has a continuous range of optical densities, which means, that the filter has a range in which intensities can be reduced. The filter is also observed to be reflective, thus, it should be placed at an angle in the beam path.

2.4.5 Fabry–Pérot interferometer

The Fabry–Pérot interferometer (FPI) is an optical resonator. It consists out of two partially transmitting mirrors, or a material which reflects at its outer layers. If the length L of the material or the distance between the mirrors is fixed, this is also called an etalon. When entering the etalon from the left side (or passing the first mirror), it propagates to the right (second mirror) where the light gets partially reflected. This reflected light is again reflected by the first mirror and therefore again directed onto the second one. Now two (or more) light beams interfere. This interference can be constructive or destructive depending on the thickness L and the wavelength of the incoming light. The interference is constructive, if the wavelength is an integer multiple of half the thickness:

$$\lambda = \frac{nL}{2}, \qquad n \in \mathbb{N}. \tag{13}$$

Therefore, the measured transition spectrum behind the etalon consists out of thin peaks at the resonating wavelengths and vanishes otherwise. The frequency difference between two peaks (resulting from higher modes) can be found at a distance $\delta\nu$ with:

$$\delta\nu = \frac{c}{2L}, \tag{14}$$

which is wavelength independent. Here $\delta\nu$ is often referred as *free spectral range* (FSR). c in Equation 14 is the speed of light in the etalon. Since the FSR depends only on L, the spectrum shows thin equidistant peaks at the distance of $\delta\nu$. The width of the peaks is characterized by the finesse $\mathcal{F}$, which is defined as:

$$\mathcal{F} := \frac{\delta\lambda}{\mathrm{FWHM}}, \tag{15}$$

with $\delta\lambda$ the FSR in wavelengths and FWHM the full-width-half-maximum of the peak. In this experiment an FPI with a FSR of 750 MHz (the manual is wrong here) and finesse of 300 is used. The FPI peaks are used later, in order to calibrate the measured spectrum to the right frequencies. In order to achieve these equidistant peaks, a mode-hop-free range needs to be found.

2.4.6 Acousto-optical modulator

An acousto-optical modulator (AOM) (see Figure 3) is used to manipulate the wavelength of a laser by a small amount. Hereby, an AOM uses sound-waves with a modulable frequency Ω in order to shift the frequency ω of the laser beam by multiples of its own amount, depending on the order of diffraction. The AOM consists out of a crystal in which ultrasonic waves are introduced. These sound waves lead to a periodic modulation of density in the crystal, which acts like a lattice. The lattice constant Λ is equal to the wavelength λ of the sound wave. Thus, the incident light beam is scattered at a lattice with variable lattice constant. Therefore, the beam can be found at all angles satisfying [13]:

$$\sin(\theta_n) = \frac{\lambda}{2\Lambda}, \tag{16}$$

with θ_n the Bragg-angle. Different to a normal lattice, the outgoing beam frequency ω_{out} depends on the order of diffraction m with [13]:

$$\omega_{\text{out}} = \omega_{\text{in}} + m \cdot \Omega, \qquad m \in \mathbb{Z}. \tag{17}$$

AOMs are used in order to create a small frequency difference between two beams created with the same laser [13].

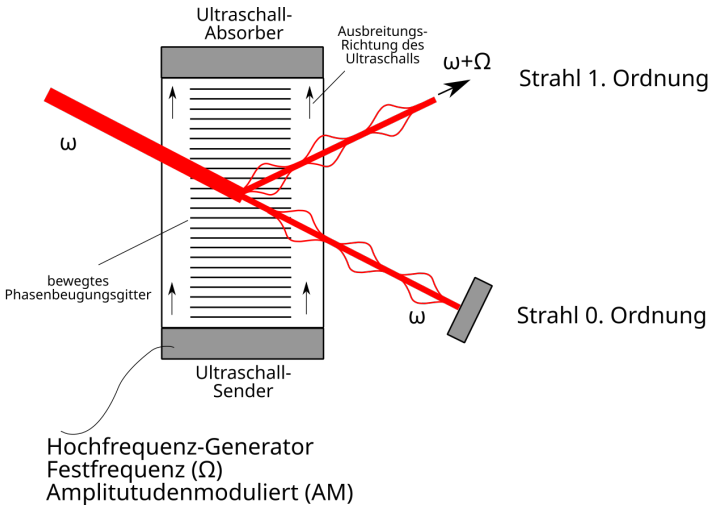


Fig. 3: Sketch of AOM working principle [14].

2.4.7 Polarization maintaining fibre

Polarization maintaining fibres (PMF) are used in order to overcome large distances in the beam path, without the need of placing lots of mirrors. It maintains the polarization of the incoming light. These fibres often consist out of a core, where the light is passed through, with a cladding structure around it (see Figure 36 in the attachment). In some fibres, in this cladding there are stress-applying parts, which produce a mechanical stress, leading to the formation of a fast and slow axis in the fibre. This fast and slow axis lead to a different transmission velocities through the fibre, depending on the polarization axis of the incident beam. In order to maintain the polarization and get the most intensity out of the fibre, the polarization of the incident beam needs to be aligned with the slow axis and should have an incident angle of less than 1.8° [15].

2.5 Beat note

Preliminary measurements for the EIT setup include the measurement of the beat note of two overlapping light beams. This beat note originates from the slight frequency shift obtained from

the AOMs. At one point, the laser beam is splitted with a PBS. Both resulting rays are directed through their own AOM, causing a shift in frequency, leading to two different frequencies which are now called f_1 and f_2. Later, those two beams are reunited. This reunion leads to a beat note, since two waves with slightly different frequencies overlap. This overlap of two different oscillations can be calculated using trigonometric identities:

$$\cos(2\pi f_1 t) + \cos(2\pi f_2 t) = 2 \underbrace{\cos\left(2\pi \frac{f_1 + f_2}{2} t\right)}_{\text{carrier}} \underbrace{\cos\left(2\pi \frac{f_1 - f_2}{2} t\right)}_{\text{envelope}}. \tag{18}$$

Here, the envelope of the wave leads to the measured beat note f_{beat}, which can directly be calculated with:

$$f_{\text{beat}} = f_1 - f_2. \tag{19}$$

Since the beat note gives directly the difference between the two frequencies, the difference can be set with very high precision, down to the hertz range.

3 Methods

In this section, the procedures for all measurements are explained, as well as an in depth explanation about all the important setup steps, with their specific problems.

3.1 Lasing threshold measurement

In a first step, the lasing threshold needs to be determined. For this measurement, a power meter is used, which can measure the power of the laser beam. The setup for this measurement can be seen in Figure 4. To prevent the laser from re-entering the laser cavity, an isolator is placed behind the laser. The power meter can be screwed onto the table to make the results reproducible. The power meter range can be set from microwatt up to milliwatt. In order to perform the measurement with the power meter, different laser currents are applied to the laser diode by changing `set current`, without any periodic variation from the grating or the current itself.

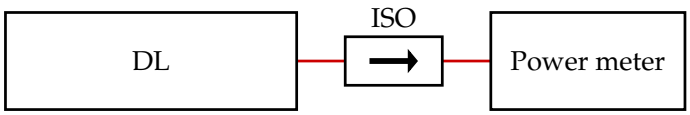


Fig. 4: Setup for the lasing threshold measurement with a power meter. The diode laser (DL) is placed in front of the isolator (ISO).

3.2 Configuration of the spectroscopy measurement setup

The setup for the spectroscopy measurement can be seen in Figure 5. The configuration itself can be split into two separate part: first, the adjustment of all elements in the optical path, and second, the optimization of the laser diode settings. These two different parts are explained in the next two subsections.

3.2.1 Setup of the optical elements

In order to set all elements up, the laser diode is turned on with an additional variation of the frequency from the grating. This variation is set to 10 Hz. For the positioning of the elements, a laser viewing card, as well as a holder with a piece of millimeter paper is used. To see the laser beam on the millimeter paper, a smartphone camera is used. A good setup requires, that the height of the beam (above the table) is constant at every optical element. This is crucial, since in this experiment at several points laser beams need to overlap, which is hard to achieve, if the beam height changes at every element. In order to check for the height, the holder with the millimeter paper is used. Here, the height should match the height of the FPI input hole. In addition to that, the beam should be directed to the middle of each optical element, to achieve the best control of the beam. This is even more important for the EIT setup, as explained later.

Now all elements, apart from the Rb cell and the ND filter, are set up with this in mind. Here, the HWP_1 is adjusted in such a way, that (almost) the complete intensity of the laser beam, passing the PBS, is directed onto M_1. The distribution of intensity after a beam splitter is measured with the power meter. This is done after every beam splitter to check for the wanted distribution. The top row in Figure 5 is arranged in this way, so it is possible to switch between the setups at any time, by just turning HWP_1.
For later measurements (part 2), the laser signal needs to be locked to a certain frequency. Since in the second part, AOMs are used, which change the frequency, already in this setup the AOM needs to be positioned and set to the right order of diffraction (1. order). The AOM has a

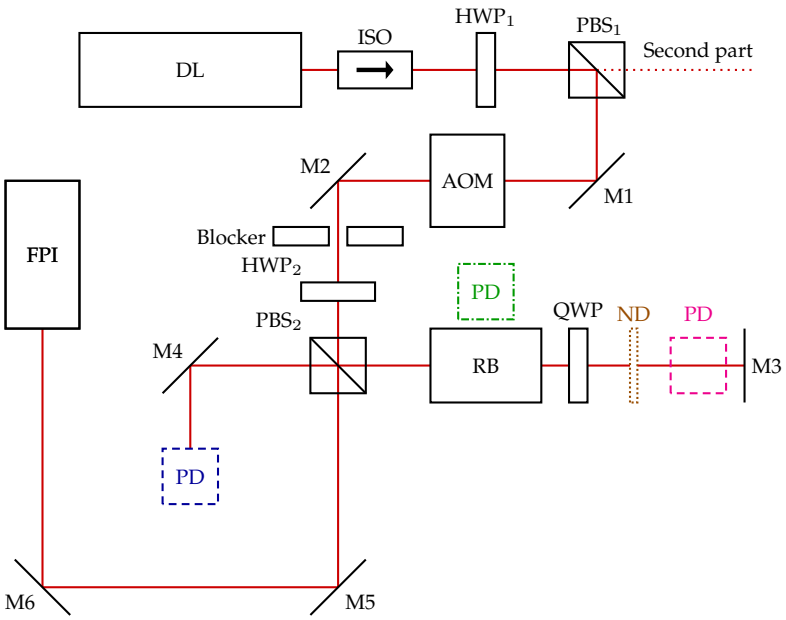


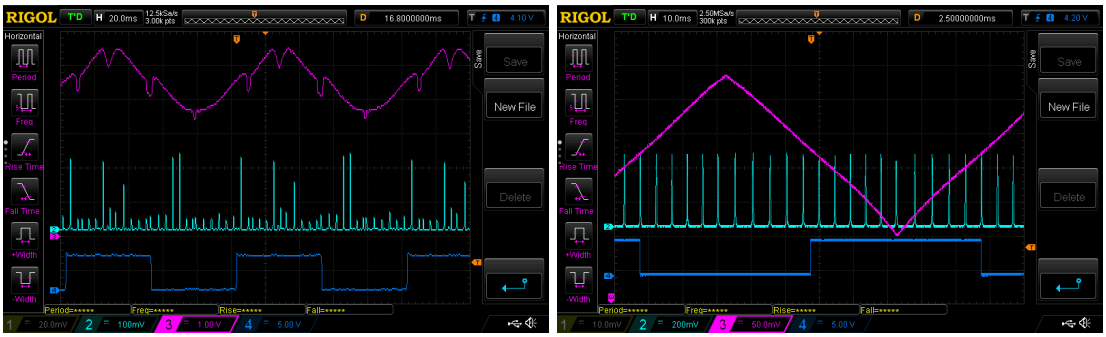
Fig. 5: Schematic setup for the spectral measurement. The used elements are the diode laser (DL), isolator (ISO), half-wave plates (HWP), quarter-wave plate (QWP), polarizing beam splitter (PBS), acousto-optical modulator (AOM), rubidium cell (RB), neutral density filter (ND), photodiode (PD), Fabry-Pérot interferometer (FPI) and mirrors (M1) - (M6). Also the beam path for the second part of the experiment is drawn as dotted line. The dotted optical devices are only used in some parts. The ND filter is used for absorption and saturation spectroscopy. The magenta PD is used for absorption spectroscopy, the blue PD for saturation and the green PD for fluorescence.

small hole for the incoming laser beam. In order to adjust the height of the beam, as well as the intensities for the different orders, the AOM offers 4 screws. These screws are adjusted until the AOM has the same height as the laser beam and the intensitiy of the wanted order is maximal. To identify the orders, the AOM is turned off, resulting in only the 0th order remaining. For the adjustment of the mirrors behind an AOM, one should keep in mind, that the wanted order hits the mirror in the middle. To prevent any unwanted signals from different orders, blockers are used to only let the wanted order pass.
For the further beam path, the HWP_2 is turned in such a way, that half of the intensity is directed to each of the two beam paths. To ensure, that enough intensity for the FPI measurement and spectral measurement is available. In order to obtain a signal at M_4 for the back-reflected beam from M_3, the QWP is needed, which acts as an HWP, since the beam passes twice. This QWP is adjusted, that the intensity at M_4 is maximal.
Both the FPI and the photodiode (PD) are connected to an oscilloscope. In order to adjust the last two mirrors M_5 and M_6, which direct the beam into the FPI, the FPI signal at the oscilloscope is maximized. At this point, the peaks probably will not show the expected equidistant structure, since the laser settings need to be adjusted. The same is done for the PD, by changing the position of the PD itself or the mirrors in front of it. The different positions of the photodiode come from the different spectroscopy methods used in this experiment. The exact positions are explained in the following sections.
To set up the Rb cell, the beam should hit the cell in the middle. The cell should also stand with a small angle in the path, to avoid direct back-reflections from the ends of the cell. The same

holds for the ND filter, which should stand in the path at a small angle. To choose the strength of the ND filter, the signal at the oscilloscope is observed, until it has the best signal-to-noise ratio. In this experiment, the filter is set to the biggest optical density. It can also be observed, that very small adjustments can lead to significant differences in the PD signal. Therefore, the filter is moved carefully.

3.2.2 Optimization of the diode laser settings

The main goal in the optimization of the laser settings is to achieve a mode-hop-free range within the wanted frequency range. The frequency range strongly depends on the temperature, set current, scan offset, scan amplitude and the feed forward setting. All these settings influence if mode-hops occur. An example for settings with mode-hops and one for a mode-hop-free range can be seen in Figure 6.



(a) Mode hops. (b) Mode-hop free.

Fig. 6: Example for a mode-hop-free range and a range with mode hops. The signal from the PD can be seen in magenta and the FPI signal in cyan. The blue signal is used as a trigger. In a) it can be seen, that the FPI signal does not show equidistant peaks with the same height, also leading to distortions in the PD signal, indicated by the square shaped dips. The settings for a) are scan offset: 20.0 V, scan amplitude: 20.0 V_{pp}, set current: 168.89 mA and feed forward: −0.007. In b) the wanted equidistant structure of the FPI peaks is visible. Therefore, the PD signal also shows a smooth ramp. The settings for b) are scan offset: 12.2 V scan amplitude:15.9 V_{pp} set current: 165.38 mA and feed forward: −0.64.

In order to adjust the parameters, the saturation spectroscopy setup with the inserted rubidium cell, is used. To firstly find the right frequency range, a rough estimate with the temperature and set current is made, until first absorption peaks are visible. After this, the temperature is held constant, since changing the temperature has big influences on the frequency and the temperature always needs around 5 minutes to stabilize.
After the rough laser current is found, the scan range is broadened by changing the scan amplitude. For further adjustments of the peaks on the triangular ramp, the scan offset can help. If the ramp roughly matches the wanted frequency range, a mode-hop-free range should be found. In order to find the equidistant peak structure (indicating a mode-hop-free range), one should look at the influence of each individual setting. Here, some settings lead to additional peaks in the FPI spectrum and other settings lead to a shift of those, or to both. Also the heights of the peaks are influenced. Knowing which setting influences which exact property in the FPI spectrum helps to find the equidistant peak structure, with peaks of the same height. To get a better feeling for all these adjustments, it is easier to start with a smaller scan offset and scan amplitude and then slowly increasing both by maintaining a mode-hop-free range.
Once the settings are adjusted in such a way, that the FPI peaks indicate a mode-hop-free range,

and the ramp is wide enough for all peaks to be visible, the spectral measurements can start. For each spectroscopy method, there can be slight shifts in the settings, as well as adjustments in the arrangement of the optical elements, like an ND filter. Adjusting optical elements only slightly can lead to massive improvements of the measured signal. For the best signal, many iterations of re-arranging the setup and adjusting the laser settings are needed.

3.3 Fluorescence

For the fluorescence spectroscopy, the photodiode is moved to the according position, see Figure 5. It is positioned almost perpendicular to the rubidium cell. As explained in the previous section, all adjustments are made until a mode-hop-free range occurs within the wanted frequency range. An appropriate example for the fluorescence measurement can be seen in Figure 7, which shows the expected peaks onto the triangular ramp.

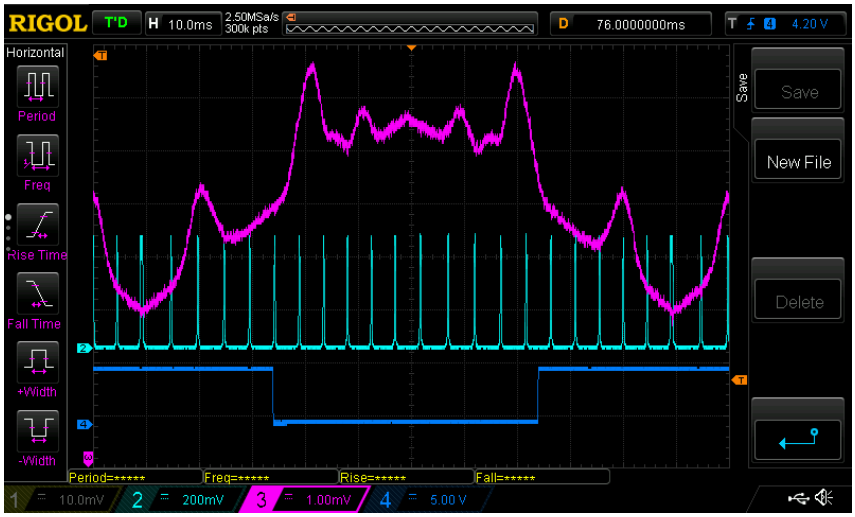


Fig. 7: Fluorescence spectrum recorded with an oscilloscope. The magenta signal comes from the photodiode, which is placed next to the rubidium cell. As expected, there is an underlying triangular ramp with clearly visible peaks onto it. The cyan signal comes from the FPI and shows the wanted equidistant structure. The blue square signal is used as a trigger.

3.4 Absorption

For absorption spectroscopy, the photodiode is placed like shown in Figure 5. Here, the PD is placed behind the ND filter, to avoid saturating it. This happens often, thus, sometimes an additional ND filter needs to be used. The positioning of the PD also blocks the laser beam from reaching M_3. Therefore, the light is not reflected back into the rubidium cell. Also the laser settings are adjusted, until a mode-hop-free range is achieved. An example for the measurement of the absorption spectrum can be seen in Figure 8, which shows the expected dips on the triangular ramp.

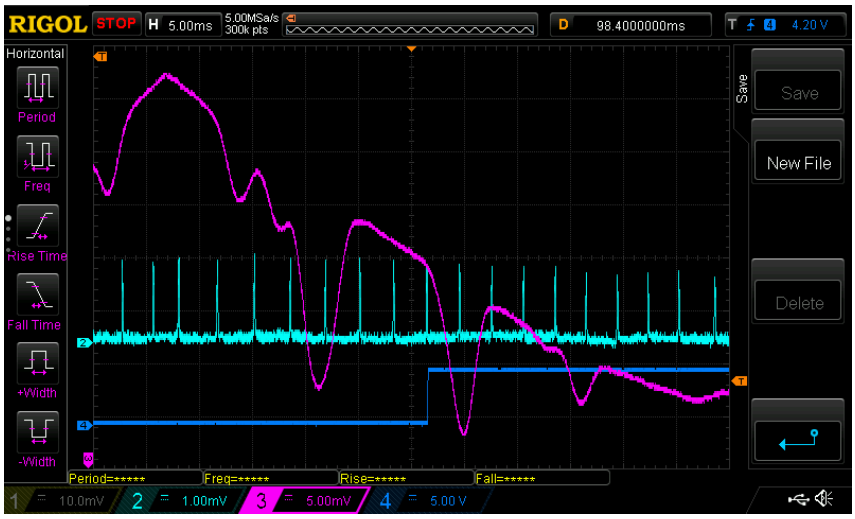
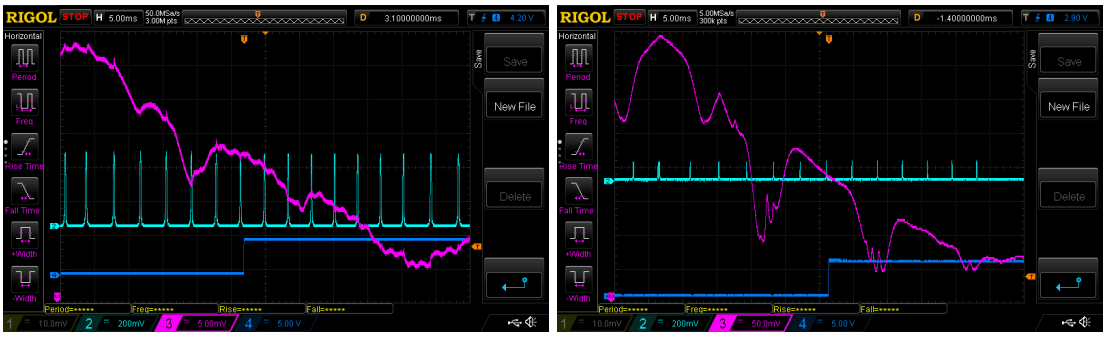


Fig. 8: Absorption spectrum recorded with an oscilloscope. The magenta signal comes from the photodiode, which is placed behind the rubidium cell. As expected, there is an underlying triangular ramp with 6 clearly visible dips. The cyan signal comes from the FPI and shows the wanted equidistant structure. The blue square signal is used as a trigger.

3.5 Saturation

To perform saturation spectroscopy, the photodiode is placed at the appropriate position, see Figure 5. Here, the photodiode is placed behind M_4. In order to obtain a saturation signal, the ND filter is crucial. Even small changes in the position of the filter can improve the measured spectrum drastically. Also the positioning of the photodiode, or the adjustment of M_3 and M_4 can improve the measurement a lot. It is crucial to adjust M_3 in such a way, that the incoming and reflected beam are overlapping and therefore passing the rubidium cell in the same position. If not adjusted properly, one will only see absorption and no peaks inside the absorption dips.

Besides these problems, at one point some oscillating background is measured , see Figure 9a. Such a background makes measurements impossible, since the oscillations have approximately the same height as the saturation peaks. The source of the oscillations is thought to be back reflections from the FPI. This thesis is underlined by the fact, that blocking the signal to the FPI stops the oscillations in the PD signal. In order to cancel this oscillating background, the setup needs to be adjusted in order to prevent those back reflections. For example this can be done by re-arranging mirrors 5 and 6, that there is an small incident angle into the FPI opening, or again adjusting all optical elements. If the oscillations vanish, measurements can be performed again. An example measurement of saturation spectroscopy can be seen in Figure 9b, which shows clear peaks inside the absorption dips.

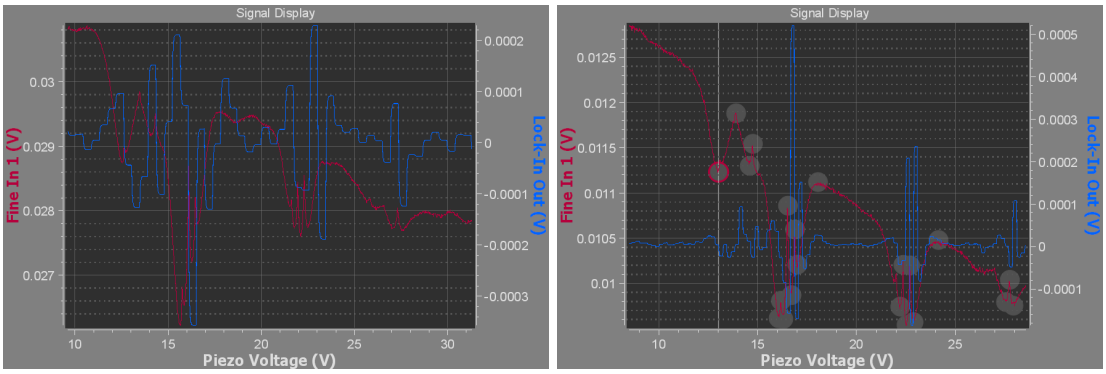


(a) Oscillating background. (b) Saturation spectrum.

Fig. 9: Saturation spectrum recorded with an oscilloscope. The magenta signal comes from the photodiode, which is placed behind M_4. a) shows an oscillating background, which needs to be avoided. b) shows, as expected, an underlying triangular ramp with the expected saturation peaks inside absorption dips. The cyan signal comes from the FPI and shows the wanted equidistant structure. The blue square signal is used as a trigger.

3.6 Lock-in method

In order to perform an EIT measurement, it is crucial to keep the laser beam at a constant frequency. This can be achieved by *locking* the laser. For this, the laser controller offers an easy to handle mode, in which so called *Lock points* are offered. One of these lock points can then be chosen, to keep the laser beam at a constant frequency. In order the keep it constant, PID controllers are used. In the following, the basic concepts of the Lock-in method and PID controllers are introduced briefly.



(a) Error signal. (b) Possible lock points.

Fig. 10: In a) the error signal (blue), obtained by processing the PD signal (red), can clearly be seen. Here, the x-axis is the piezo voltage applied to the grating. The y-axis is in Volt. This error signal is then used, to detect possible lock points, which are indicated in b). Since top-of-fringe locking is selected, the lock points are placed at the peaks in the spectrum, which are found by zero-crossings of the error signal.

The PID controller needs an error signal $e(t)$ as input. This error signal is obtained by processing the signal from the photodiode. It is needed as an indicator for the later locking. There are two types of locking: side-of-fringe and top-of-fringe locking. In side-of-fringe locking, the lock point is on a flank of a peak, for the top-of-fringe locking, the lock point can be found directly at the peak. In this experiment, top-of-fringe locking is used. In Figure 10a, such an error signal can

be seen in blue. The error signal for top-of-fringe locking can be calculated with:

$$e(t) = I_{\text{set}} - I_{\text{meas}}(t), \tag{20}$$

with I_{set} the set intensity (or voltage) at the lock point and I_{measured} the measured intensity at the photodiode. For a good working of the PID feedback loop, the error signal needs to be optimized. For this there are four different parameters. In the laser controller manual [10], there is an detailed explanation, which parameters should be adjusted. This manual is also helpful for the setup explained in subsubsection 3.2.2. With an appropriate error signal and automatic lock point detection, all possible lock points should be visible, as shown in Figure 10b. One can now select one of these lock points, which starts the PID feedback loop (see Figure 12). In Figure 11 it can be seen how the screen looks like, when a lock point is selected. To optimize the feedback loop, the settings at the PID controller are improved.

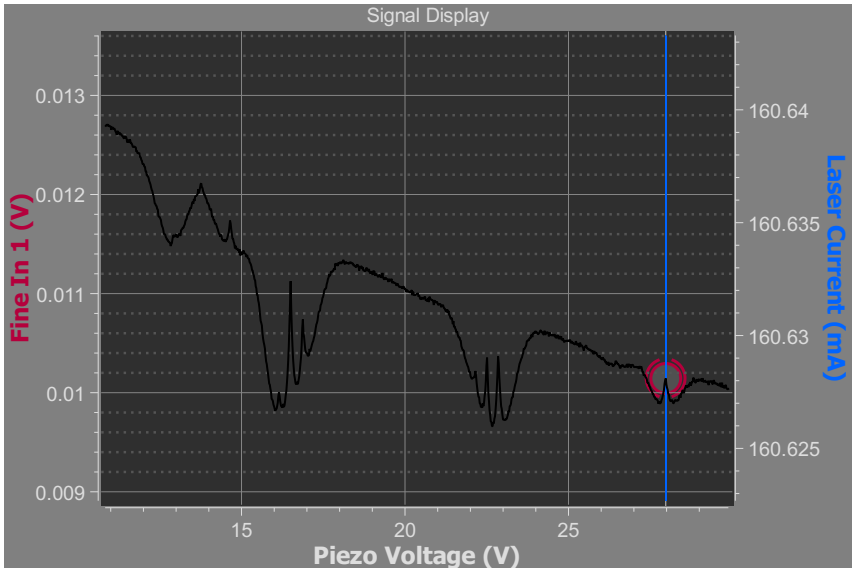


Fig. 11: Here, a lock point is selected from one of the possible lock points. To have a rough indicator for the stability of the frequency, there is a red ball, which moves around at the start of the lock-in process, but stabilizes for the PID settings. Here, the ball has already stabilized at the lock point.

In general, a PID is used in order to hold a given variable constant and counteract forces that bring it out of equilibrium. Here, the PID controller is used in order to hold the error signal at $e(t) = 0$, since in top-of-fringe locking, $e(t) = 0$ is used as the locking point (if the intensity left and right of it significantly deviates from 0). For this regulation, the PID controller calculates a correction signal $u(t)$, which is influenced by three main components: **P**roportional term, **I**ntegral term and **D**erivative term. These components influence the calculation of $u(t)$ by adjusting the strength of the respective term constant, named K_{P}, K_{I} and K_{D}. Hereby, the proportional term counteracts constant deviations, the integral term reacts to slow changes, by integrating over a given time and the derivative term reacts to fast changes. The correction signal can be calculated with:

$$u(t) = K_P \cdot e(t) + K_I \int e(t)\, dt + K_D \frac{de(t)}{dt}. \tag{21}$$

Further, an overall gain can be set, which increases all parameters with the same factor. In the case that the locking fails, since the frequency is too far off (error signal too far off), there is the possibility to Re-Lock the laser. The schematic diagram for Re-locking can be seen in

Figure 13. If the error signal deviates too much from the wanted value, the state is called *Out-of-Lock*. If this is the case, the controller waits for an adjustable delay. If the system returned into the locked state, the feedback loop continuous. If the system is after the delay still in the Out-of-Lock state, the PID controller settings (the parameters K) are reset. After the reset, a Re-Lock scan is initiated, if the system has returned to the In-Lock state.

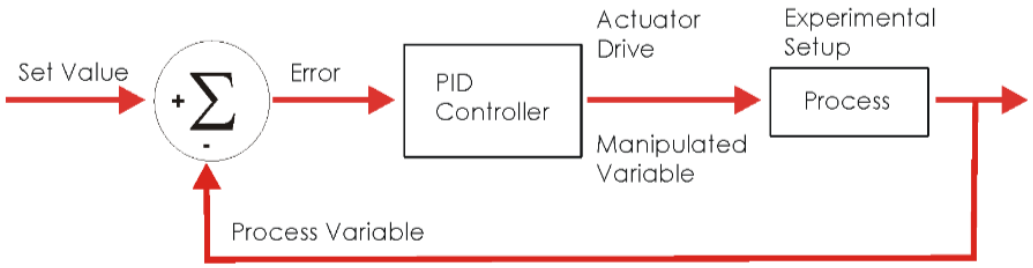


Fig. 12: Here, a schematic of the feedback loop of an PID controller is shown [10].

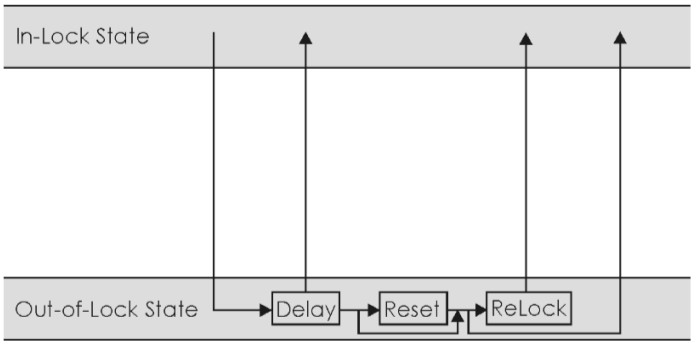


Fig. 13: Here, a schematic of the process for Re-Locking is shown [10].

If all these steps are done and all parameters are optimally set, the laser is locked to a certain frequency. For measurements, it is crucial to know, how long the laser can hold this stable frequency. The next chapter explains briefly, how such a measurement can be performed.

3.6.1 Frequency stability measurement

In order to obtain knowledge about the time duration the laser controller can hold the laser beam frequency constant, a frequency stability measurement is performed. For this measurement, the setup shown in Figure 5 is used with the laser beam frequency locked in, while the photodiode is placed in the spot for saturation spectroscopy. This is needed to constantly create an error signal needed for the PID controller. Different than before, an oscillating voltage is applied to the FPI, which is produced by an arbitrary waveform generator (AWG). This generator produces a sine wave with a given frequency. This AWG is applied to the FPI, in order to change the size of the optical cavity. This is done by a piezo element mounted onto a mirror inside the FPI. This piezo element now changes the size of the resonator according to the applied wave. Without this change, the FPI signal would be a straight line, since no frequency modulation is present (the laser is locked). Changing the size of the cavity leads to moments in time, where the frequency of the laser fits to the cavity size, leading to constructive interference. Otherwise the cavity

size leads to destructive interference. This again creates peaks in the FPI spectrum, which give insights about the frequency stability of the locked laser frequency. The laser frequency is stable, if the peak distance remains constant over time, since an unstable laser frequency would lead to slightly different peak distances. An example for this measurement at one time, can be seen in Figure 14.

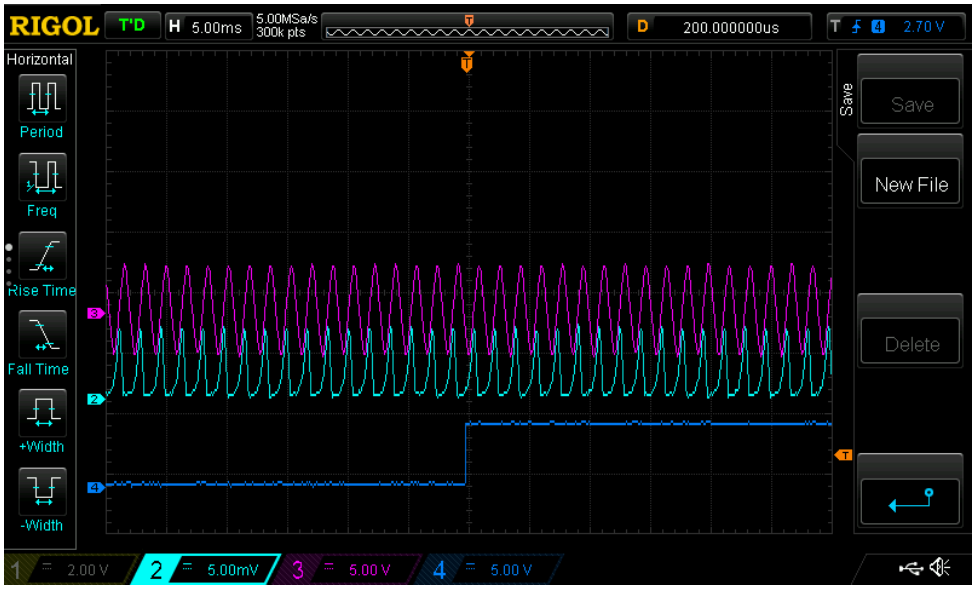


Fig. 14: Stability measurement of the Lock-in signal with the FPI. Here, an oscillating voltage (magenta) is applied to the FPI, to see if the laser frequency changes over time. This is observed at the FPI signal (cyan), if the peaks move over time. The blue signal is again used as a trigger.

3.7 Configuration of the EIT setup

The setup for the EIT measurement (with the preliminary beat note measurement) can be seen in Figure 15. As for the spectral setup, the configuration can be split into two main parts: first, the setup of all optical devices and second, the optimization of the laser settings, especially selecting the right frequency with the Lock-in method. These are explained in the next two subsections.

3.7.1 Setup of the optical elements

For the setup of the most elements in this part, the same methods and procedures apply, which are already explained in subsubsection 3.2.1.
Now the HWP_1 is turned in such a way, that the full intensity is directed to HWP_2. This wave-plate is then turned in a way, that half the intensity goes to each AOM. The two AOMs are adjusted in a way, that only the +1 (or -1) orders are passed to the mirrors behind. This is crucial, since choosing different orders makes the beat note measurement later impossible, because the shift of a single order is too big to be resolved with the laser controller. For further measurements, it is also important to know the exact difference between the two frequencies, which could not be achieved, if different orders are used. To indicate the two different beam polarizations and frequencies, they are drawn purple and orange for the parts they are separated. If they are again aligned, they are drawn red again.
The two beams are re-aligned at the PBS_3 and then together directed onto M_6. Here, it is very important, that the beams hit the PBS perfectly in the middle and at the same height, to ensure the proper alignment afterwards. Small deviations here can lead to big deviations at the input

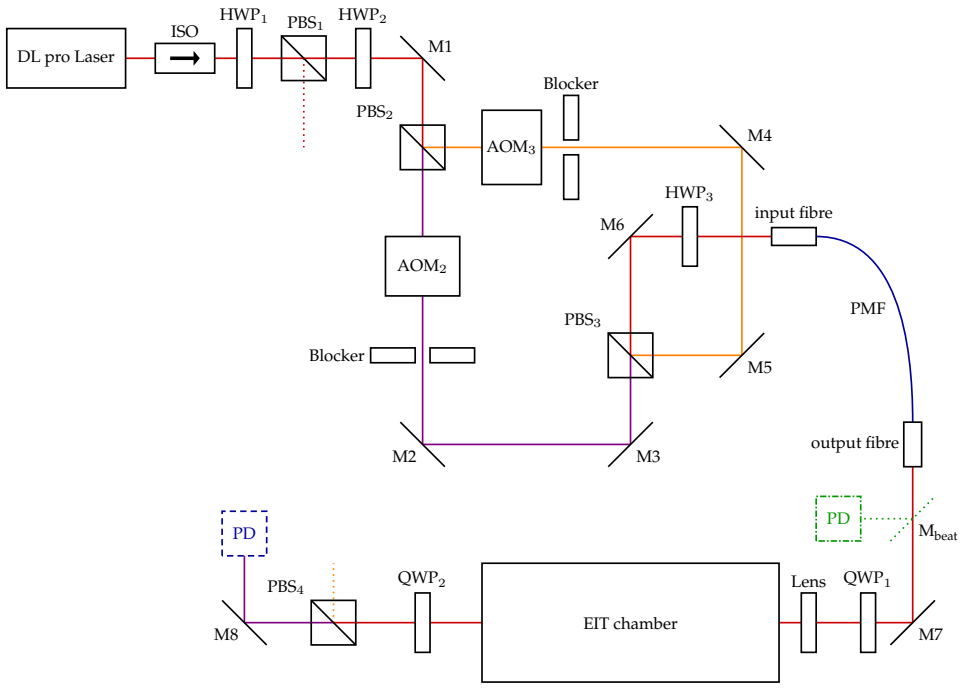


Fig. 15: Schematic setup for the EIT/beat note measurement. The used elements are the diode laser (DL), isolator (ISO), half-wave plates (HWP), quarter-wave plate (QWP), polarizing beam splitter (PBS), acousto-optical modulators (AOM), optional neutral density filters (ND) (not drawn), photodiode (PD), light to fibre coupler (fibre in/out), polarization maintaining fibre (PMF), EIT chamber (filled with rubidium gas), a lens and mirrors (M1) - (M6). To differentiate the beams splitted by PBS_2, there are two colours used (orange and purple) to indicate the different polarizations. These beams then reunite at PBS_3 and are directed onto the fibre. After PBS_4, the beams are again separated, which only let the purple one pass to the PD. To measure the beat note after the fibre, there is an additional mirror M_{beat} placed, and a photodiode. For the EIT measurements, those elements are removed. Also the beam path to the first part is drawn as a dotted line.

of the fibre. In order to adjust this, the mirrors M_2 to M_5 need to be aligned very carefully. It is already checked, that the AOMs emit the beam at the wanted height.
After this alignment is done, there is a HWP, which is used for further adjustments of the polarization. After this wave-plate, there is the input to the polarization maintaining fibre. This fibre needs to be hit very precisely, because otherwise nothing will pass. To record if the beam passes the fibre, the power meter is placed right at the output of the fibre. In order to hit the fibre perfectly, it is necessary, that all mirrors behind the AOMs are adjusted. The fibre-alignment process takes two days in this experiment.
The two QWPs in this setup are aligned in such a way, that the output at the photodiode is maximized. This is achieved by carefully rotating one QWP at a time. In this step, the biggest problem is the alignment of the beam at the optical elements, since in this experiment, the achieved intensity behind the fibre is so small, that the beam is not visible with a laser viewer card or the camera of a smartphone. Thus, the alignment needs the constant help of the power meter or the photodiode.

3.7.2 Optimization of the diode laser settings and Lock-in signal

In order to run the EIT measurement, a certain frequency needs to be selected. This is held constant with the help of the PID controller. This uses the lock-in method. The laser can only be locked with the help of the first setup, see Figure 5. For this the HWP_1 is turned in such a way, that the beam is directed to the first part again. Here, the saturation spectroscopy setup is used, because the laser is locked onto the small peaks inside the absorption dips. Then the laser is locked onto the wanted transition. Since only one photodiode is available, it is not possible to have a continuous monitoring of the locking. Nevertheless, after locking, the HWP_1 is turned again, that the laser reaches part 2. After this, measurements can be performed.

3.8 Beat note measurement

For the beat note measurement, the setup shown in Figure 15 is used. Here, the mirror M_{beat} is inserted, which directs the beam onto the photodiode. For the measurement of the beat note, the frequency difference between the beams with different polarizations is important. This difference is set with the two AOMs in this part. The frequency shift at an AOM depends on the used sound wave. The sound wave frequency is manipulated by the arbitrary waveform generator (AWG). The frequency difference occurs from the difference between the AWG frequencies. An example for the beat note measurement can be seen in Figure 16, which shows a clearly visible peak at 200 Hz, which is the set frequency difference at the AWGs.

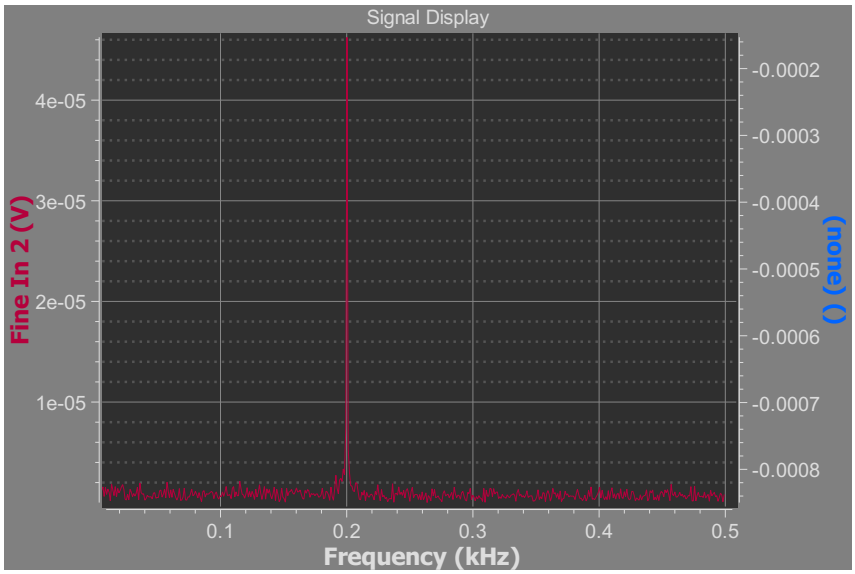


Fig. 16: Example measurement for the beat note between the two different beams. For the measurement, a PD is used. The beat note occurs due to the different frequency shifts at the AOMs. Here, an example for a frequency difference of 200 Hz can be seen. As one can see, there is a clear peak at 200 Hz, as expected.

4 Analysis and Results

4.1 Lasing threshold measurement

Before any of the other measurements could be started, the lasing threshold needs to be determined, to get a better understanding of the laser. The lasing threshold is measured as explained in subsection 3.1, with the help of a power meter. The measurement results can be seen in Figure 17.

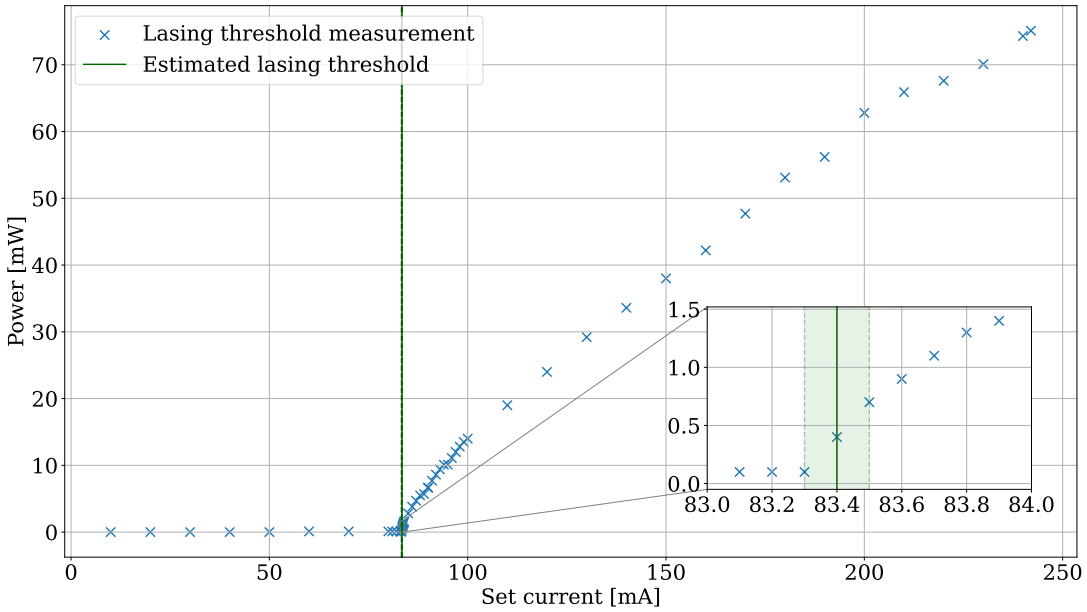


Fig. 17: Here, the laser power in mW is plotted against the set laser current in mA. The inset shows a range around the lasing threshold with additional measurements in smaller steps. The axes of the inset have the same units as the large plot. Additionally, the estimated lasing threshold with its uncertainty is displayed in green.

The lasing threshold corresponds to the current, at which the laser starts to deliver a measurable output. This can be detected by the spontaneous increase in the detected power. With this, the current for the lasing threshold is determined to be:

$$I_{\mathrm{LT}} = (83.4 \pm 0.1)\,\mathrm{mA}. \tag{22}$$

It can also be seen, that for larger currents, the beam intensity is nearly linear, as expected. For later measurements, currents in the region between 150 mA and 180 mA are chosen. The reason for that is, that higher laser intensities lead to bigger absorption and fluorescence effects.

4.2 Absorption spectroscopy

After optimizing and fine-tuning the setup and the laser, the following laser parameters are selected for the absorption spectroscopy: temperature: 24.0 °C, scan offset: 15.3 V, scan amplitude: 23.6 V_{pp}, set current: 163.36 mA, feed forward: −0.56 and a scan frequency of 10 Hz.

Before analysing the absorption spectroscopy measurement, a calibration of the channels is needed. There are 300000 channels. The calibration is performed by analysing the FPI spectrum, which is recorded during each spectroscopy measurement. The frequency difference between two FPI peak-channels during the laser current modulation is equal to the free spectral range (FSR) of the FPI. The FSR of the FPI is given in the instructions as 700 MHz [16]. Despite that, a

value of 750 MHz is later chosen in the analysis of the spectra, as it is the FSR directly printed on the FPI.

In Figure 18, the FPI measurement recorded during the absorption spectroscopy measurement is shown. The peaks are found using `scipy.signal.find_peaks` [17] and the uncertainties are estimated to be ±500 channels. It can be seen, that the peak channels are not equidistant, which can be traced back to the laser current modulation not being as linear as expected. More precisely, the difference between two consecutive peak channels varies between 12055 and 18216 channels for this measurement. To account for this systematic deviation, it is chosen to perform a linear interpolation between two consecutive peak channels, instead of a linear fit. The linear interpolation is performed using `scipy.interpolate.interp1d` [17]. With this interpolation, the FPI spectrum and the spectroscopy measurement can be converted from a channel axis to a frequency axis. Furthermore, the coordinate origin of this axis is arbitrarily set on a peak.

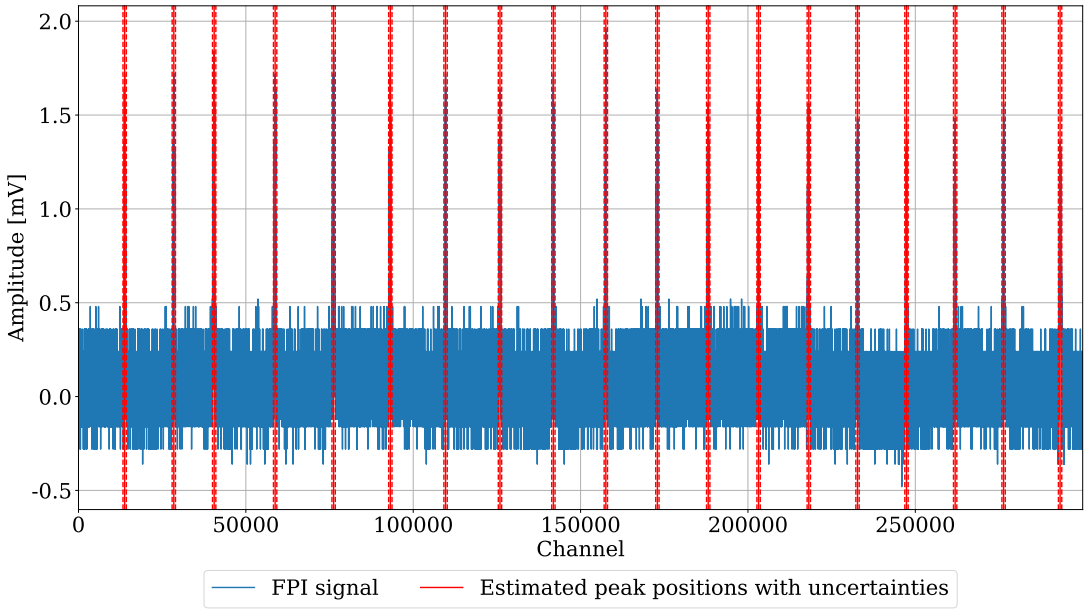


Fig. 18: In this plot the FPI spectrum, read out with the oscilloscope during the absorption spectroscopy measurement, is shown. Additionally, the identified peak channels, together with the estimated uncertainties of ±500 channels, are plotted in red. It can be seen, that the peak channels are not equidistant.

In the following, the respective FPI spectrum used for the channel-frequency conversion is always shown in addition to the spectroscopy measurement. For the absorption spectroscopy measurement, such a plot can be found in Figure 19. When plotting the raw data from the oscilloscope, the data is noisy and shows a double line effect. Such a double line effect is typical for oscilloscopes and arises most probable from internal reflections. As a consequence, a moving average over 10 data points is performed with `pandas.DataFrame.rolling` [18] to smoothen the data. Additionally, an uncertainty of 0.5 % is estimated for each data point.

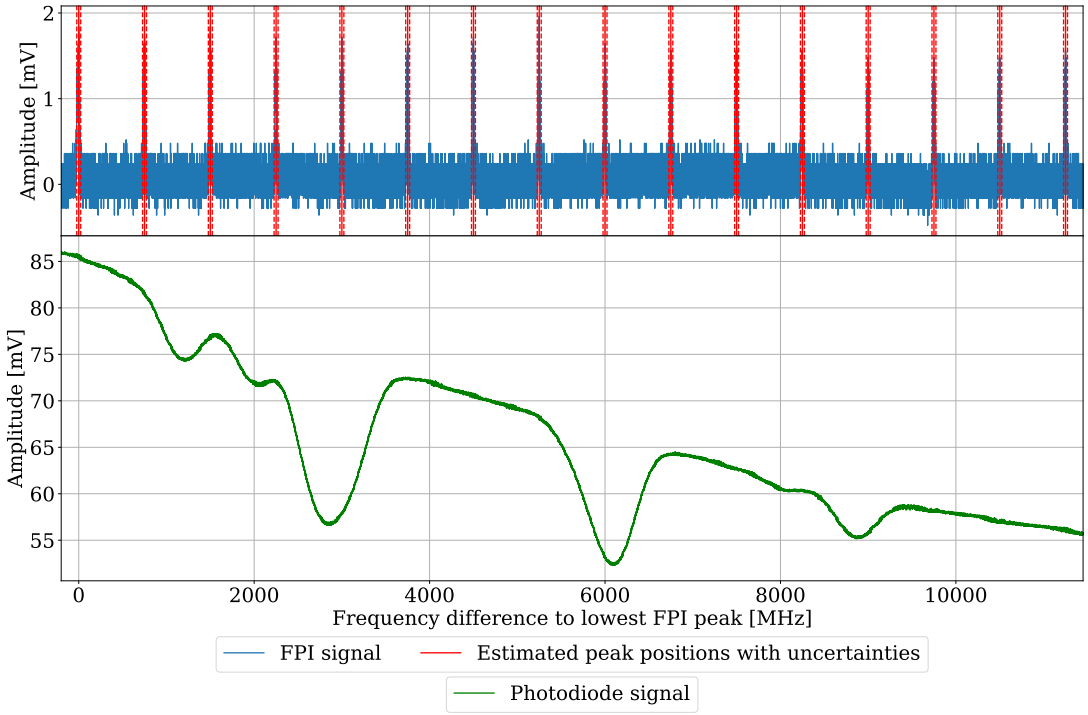


Fig. 19: In the upper plot the FPI spectrum is shown and in the lower plot the absorption spectrum is displayed. Both spectra are read out with the oscilloscope. The channel frequency-conversion is performed to plot the spectra on a frequency axis. Additionally for the upper plot, the identified peaks for the FPI, together with the estimated uncertainties, are shown. The peaks are now equidistant. For the absorption spectroscopy plot, a moving average is performed to smoothen the noisy data.

During the laser current modulation, not only the laser frequency changes, but also the intensity, as can be seen in the laser threshold measurement in Figure 17. To account for the linear background of the photodiode measurement, a linear fit of the form

$$A = a \cdot \nu + b, \tag{23}$$

with A being the amplitude in mV and ν being the frequency difference to the lowest peak in MHz, is performed with `scipy.optimize.curve_fit` [17]. For that, data outside the Rb spectrum is selected, which is indicated in orange, see Figure 20. The linear fit, together with the absorption spectrum, is shown in Figure 20.

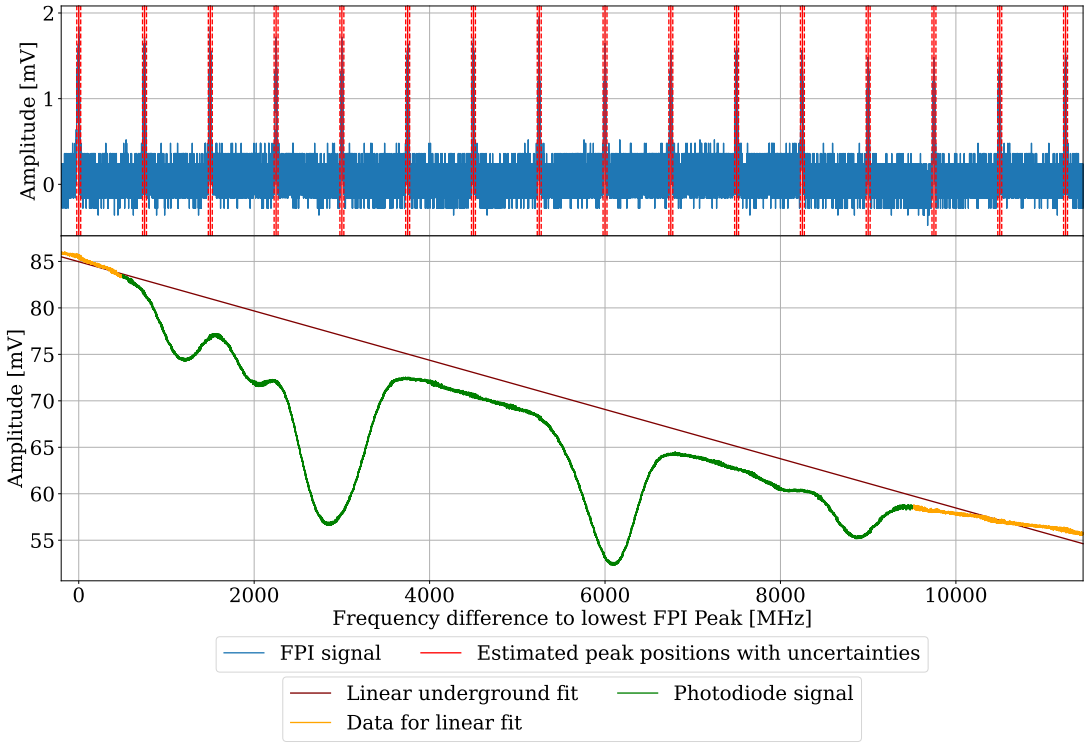


Fig. 20: In the upper plot the FPI spectrum is shown and in the lower plot the absorption spectrum is displayed. Both spectra are read out with the oscilloscope. The channel frequency-conversion is performed to plot the spectra on a frequency axis. Additionally for the upper plot, the identified peaks for the FPI, together with the estimated uncertainties, are shown. The peaks are now equidistant. For the absorption spectroscopy plot, a moving average is performed to smoothen the noisy data. To subtract the linear background, a linear fit (brown) is performed with the orange data.

It is noticeable, that the underground is not linear. Nevertheless, this only changes the amplitudes of the peaks and not the peak positions. Possible reasons for the deviation from the expected linear background will be discussed later.
To correct for this background, the linear fit is used and subtracted from the data in the following way:

$$A_{\text{corr}} = A - (a\nu + b), \tag{24}$$

$$\Delta A_{\text{corr}} = \sqrt{\Delta A^2 + (\Delta a\nu)^2 + (a\Delta\nu)^2 + \Delta b^2}, \tag{25}$$

with A_{corr} being the corrected amplitude, a being the slope of the linear background fit, b being the intercept of the linear fit and ν being the frequency. The background corrected absorption spectrum can be found in Figure 21.

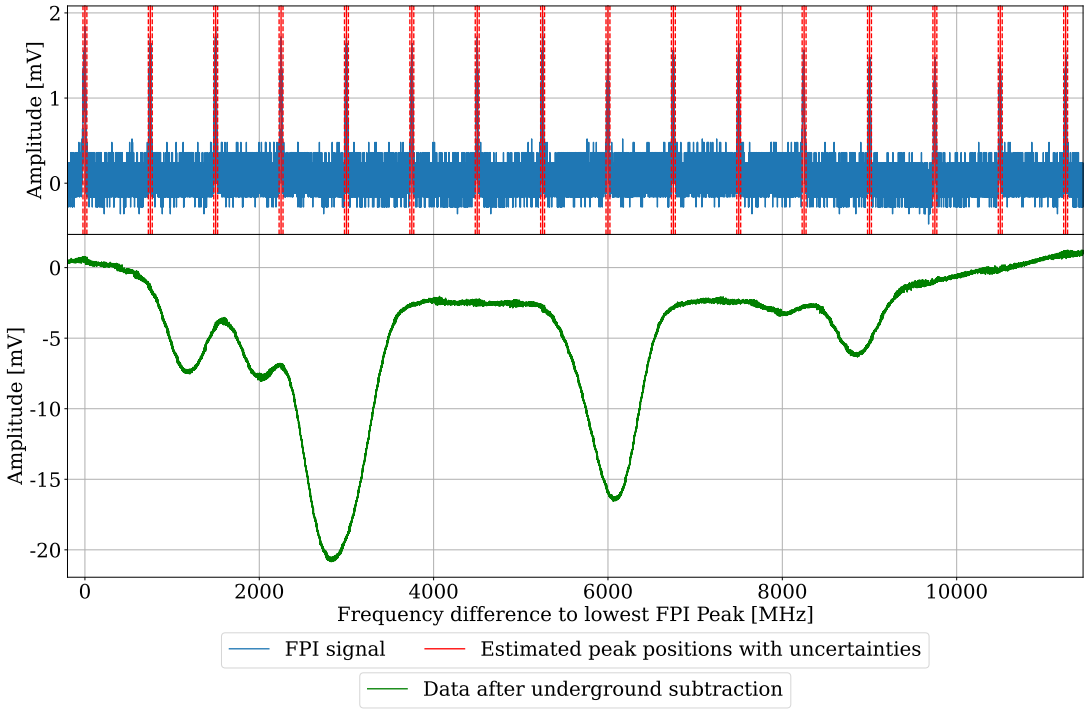


Fig. 21: In the upper plot the FPI spectrum is shown and in the lower plot the absorption spectrum is displayed. Both spectra are read out with the oscilloscope. The channel frequency-conversion is performed to plot the spectra on a frequency axis. Additionally for the upper plot, the identified peaks for the FPI, together with the estimated uncertainties, are shown. The peaks are now equidistant. For the absorption spectroscopy plot, a moving average is performed to smoothen the noisy data and the linear background is subtracted.

Now the absorption spectrum can be analysed. The absorption peaks can be approximated by Gaussian distributions of the form:

$$y = A \exp\left(-\frac{(\nu-\mu)^2}{2\sigma^2}\right), \tag{26}$$

with y as the amplitude in mV and ν as the frequency difference to the lowest FPI peak in MHz. To get better fits, the data points are restricted to an area, where a clear peak is visible. Again, `scipy.optimize.curve_fit` [17] is used to fit this model to the measured data.

As one can see in Figure 21, some peaks are close together. More precisely, the second and third peak, and the fifth and sixth peak are close to each other. As a consequence, a different fitting model is used, which takes the overlap of the two peaks into account. For that, the sum of two Gaussian distributions is used

$$y = A_1 \exp\left(-\frac{(\nu-\mu_1)^2}{2\sigma_1^2}\right) + A_2 \exp\left(-\frac{(\nu-\mu_2)^2}{2\sigma_2^2}\right), \tag{27}$$

with the index 1 indicating the left peak of the two closely laying peaks and the index 2 indicating the right peak. Once more, the data points are restricted to an area around the double peak and the fit is performed using `scipy.optimize.curve_fit` [17].

In Figure 22 the background corrected absorption spectrum is displayed. Additionally, the performed single Gaussian fits are displayed in orange and the double Gaussian fits in red. Furthermore, the respective restricted areas are shown with black lines.

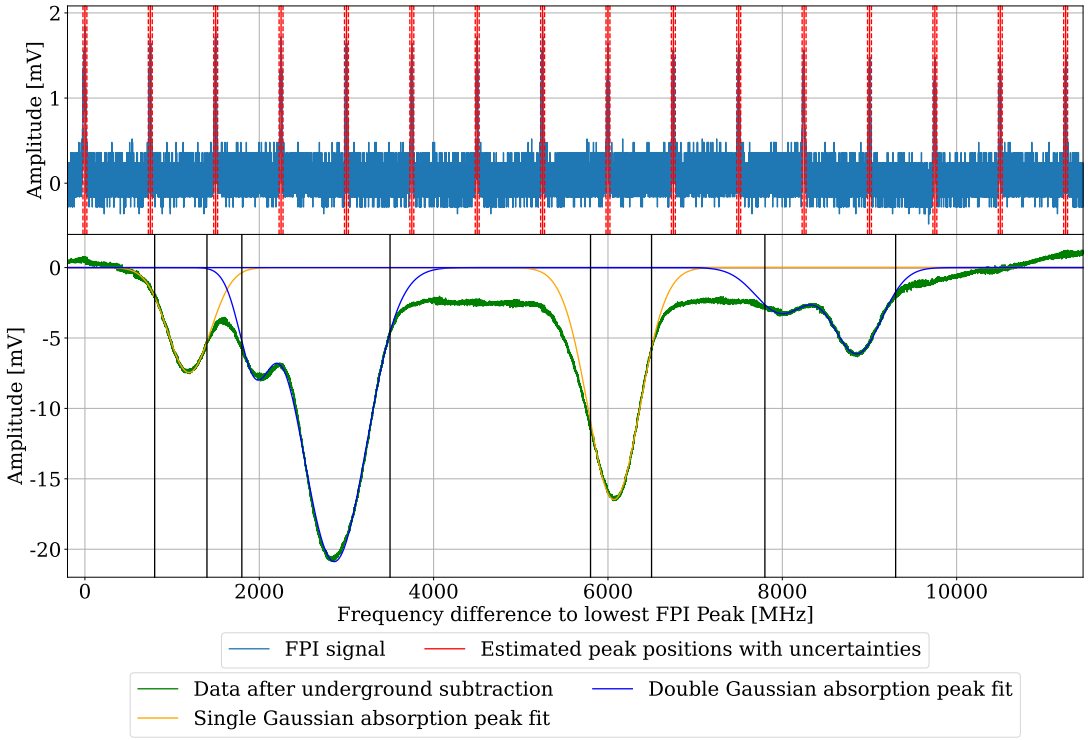


Fig. 22: In the upper plot the FPI spectrum is shown and in the lower plot the absorption spectrum is displayed. Both spectra are read out with the oscilloscope. The channel frequency-conversion is performed to plot the spectra on a frequency axis. Additionally, for the upper plot, the identified peaks for the FPI, together with the estimated uncertainties, are shown. The peaks are now equidistant. For the absorption spectroscopy plot, a moving average is performed to smoothen the noisy data and the linear background is subtracted. The first and fourth peak are fitted with a singular Gaussian from Equation 26, which is displayed in orange. The second and third peak, and the fifth and six peak are fitted using the double Gaussian fit function from Equation 27, which is displayed in blue. The black lines indicate the area in which the data contributes to the fit.

To assess the goodness of the fits for our two used models, we use the reduced χ^2-value:

$$\chi^2 = \sum_N \frac{(A_{\text{model}} - A_{\text{meas}})^2}{\Delta A_{\text{meas}}^2}, \tag{28}$$

$$\chi^2_{\text{red}} = \frac{\chi^2}{N - f}, \tag{29}$$

where N is the number of data points, A_{meas} is the measured amplitude, A_{model} is the amplitude from the fit at the same position, ΔA_{meas} is the uncertainty of the measured value and f is the number of degrees of freedom. For a value of $\chi^2_{\text{red}} \approx 1$, one can assume that the used model is appropriate for the measured data. For the absorption fits, the reduced χ^2-values are in a range of 0.08 to 0.73. This shows, that all fits have a good quality, but the small χ^2_{red}-value of 0.08 indicates an overestimation for the uncertainties.

The relevant fitting parameters for further calculations, namely the amplitude and the mean, are summarised with their uncertainties and the corresponding χ^2_{red}-values in the attachment in Table 12. The uncertainty of the amplitude is directly calculated from the covariance matrix.

It should be mentioned, that the uncertainties on the amplitudes are all on the order of a few μV, which is a significant underestimation of the true uncertainty. This will be discussed later.

For the uncertainty of the mean, the standard error is calculated:

$$\hat{\sigma} = \frac{\sigma}{\sqrt{N}}, \tag{30}$$

with σ as the standard deviation of the Gaussian and N as the number of data points used for the fit. The standard error is then propagated with the uncertainty of the frequency, as the least-square method used for the fit by `scipy.optimize.curve_fit` [17] is not considering the uncertainty of the frequency:

$$\Delta\mu = \sqrt{\hat{\sigma}^2 + \Delta\nu^2}. \tag{31}$$

4.3 Fluorescence spectroscopy

After further optimizing and fine-tuning the setup and the laser, the following laser parameters are selected for the fluorescence spectroscopy: temperature: 24.0 °C, scan offset: 12.2 V, scan amplitude: 15.9 V_{pp}, set current: 165.39 mA, feed forward: −0.64 and a scan frequency of 10 Hz.

The analysis for the fluorescence spectroscopy is performed in the same manner as for the absorption spectroscopy subsection 4.2. Here, again a channel-frequency conversion is performed, using the measured FPI peaks shown in Figure 37 in the attachment. Again, a linear interpolation is performed. Due to oscilloscope effects, once again a moving average is performed and the uncertainty of each value is estimated to be 0.5 %. The plot for the spectrum on a frequency axis is also shown in the attachment in Figure 38. To account for the linear background of the photodiode measurement, again a linear fit of the form in Equation 23 is performed with `scipy.optimize.curve_fit` [17]. For that, data outside the rubidium spectrum and between the two main peaks (where no peak is expected) is selected, which is indicated in orange, see Figure 23. The linear fit, together with the fluorescence spectrum, is shown in Figure 23. This fit is then subtracted to correct the data.

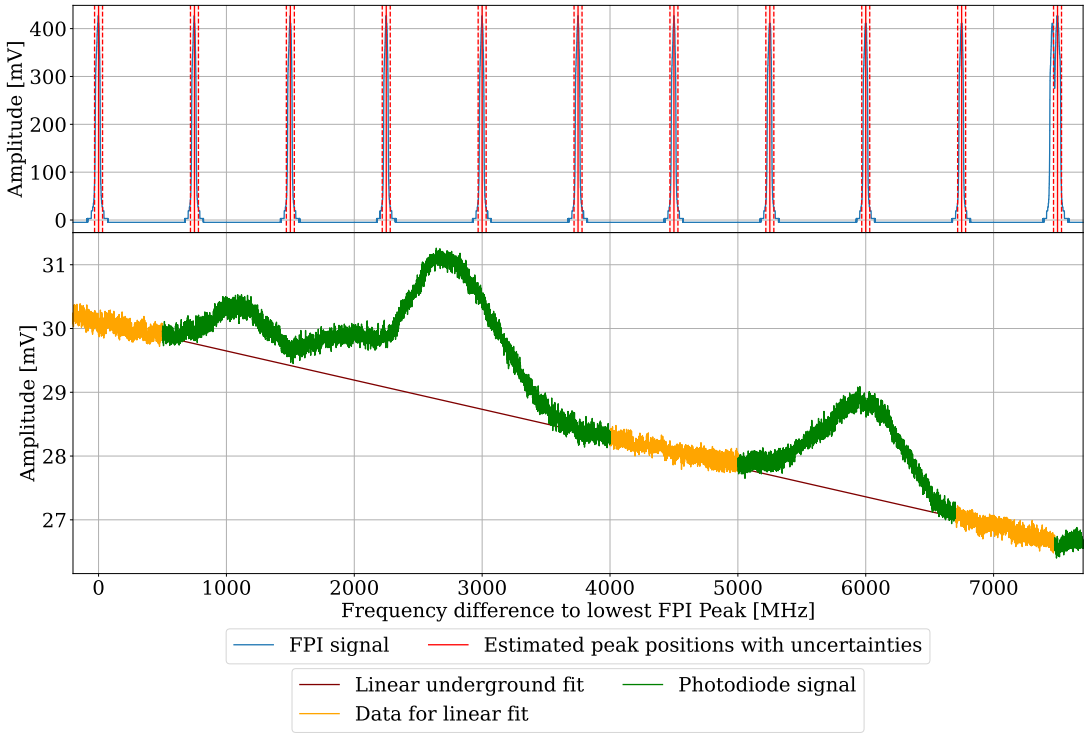


Fig. 23: In the upper plot the FPI spectrum is shown and in the lower plot the fluorescence spectrum is displayed. Both spectra are read out with the oscilloscope. The channel frequency-conversion is performed to plot the spectra on a frequency axis. Additionally for the upper plot, the identified peaks for the FPI, together with the estimated uncertainties, are shown. The peaks are now equidistant. For the fluorescence spectroscopy plot, a moving average is performed to smoothen the noisy data. To subtract the linear background, a linear fit (brown) is performed with the orange data.

In comparison to the absorption spectroscopy background, it is visible, that the expected linear background is indeed linear. Possible reasons for that are further optimizations of the laser settings at the laser controller and the experimental setup.
The background corrected fluorescence spectrum can be found in Figure 24.

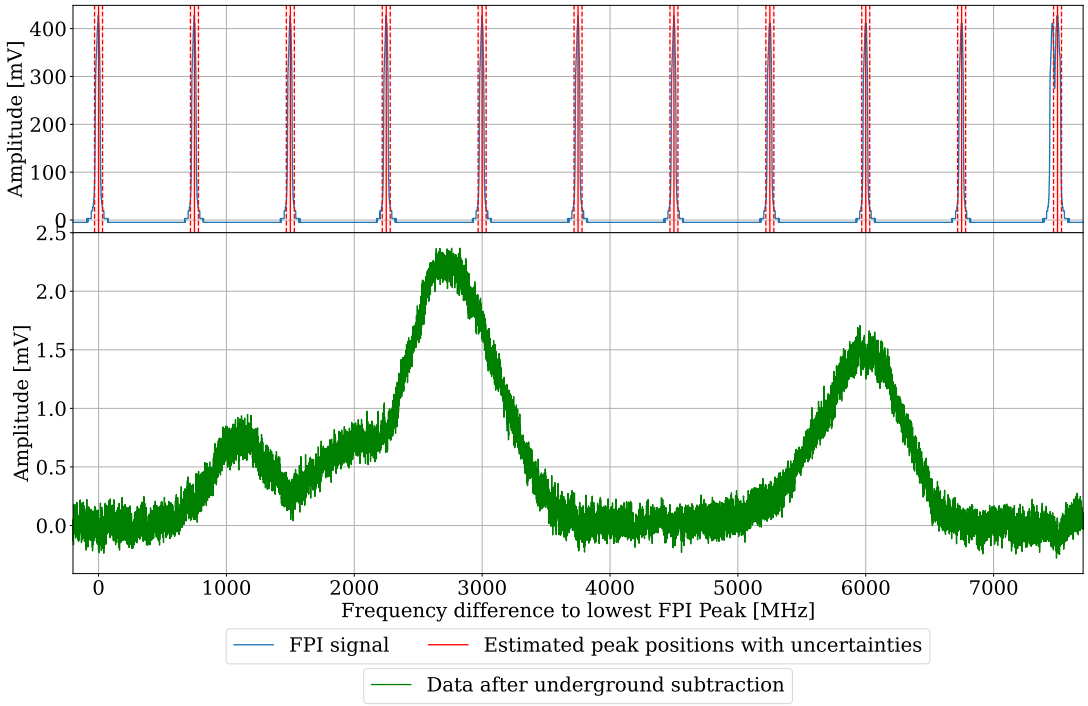


Fig. 24: In the upper plot the FPI spectrum is shown and in the lower plot the fluorescence spectrum is displayed. Both spectra are read out with the oscilloscope. The channel frequency-conversion is performed to plot the spectra on a frequency axis. Additionally for the upper plot, the identified peaks for the FPI, together with the estimated uncertainties, are shown. The peaks are now equidistant. For the fluorescence spectroscopy plot, a moving average is performed to smoothen the noisy data and the linear background is subtracted.

This time, only peak 2 and 3 are close to each other and therefore only for those peaks the double Gaussian fitting model (see Equation 27) is used, whereas the other peaks are fitted with a single Gaussian (see Equation 26). The fits are performed using `scipy.optimize.curve_fit` [17].

In Figure 25 the background corrected fluorescence spectrum is displayed. Additionally, the performed single Gaussian fits are displayed in orange and the double Gaussian fit in blue. Furthermore, the respective restricted areas are indicated with black lines.

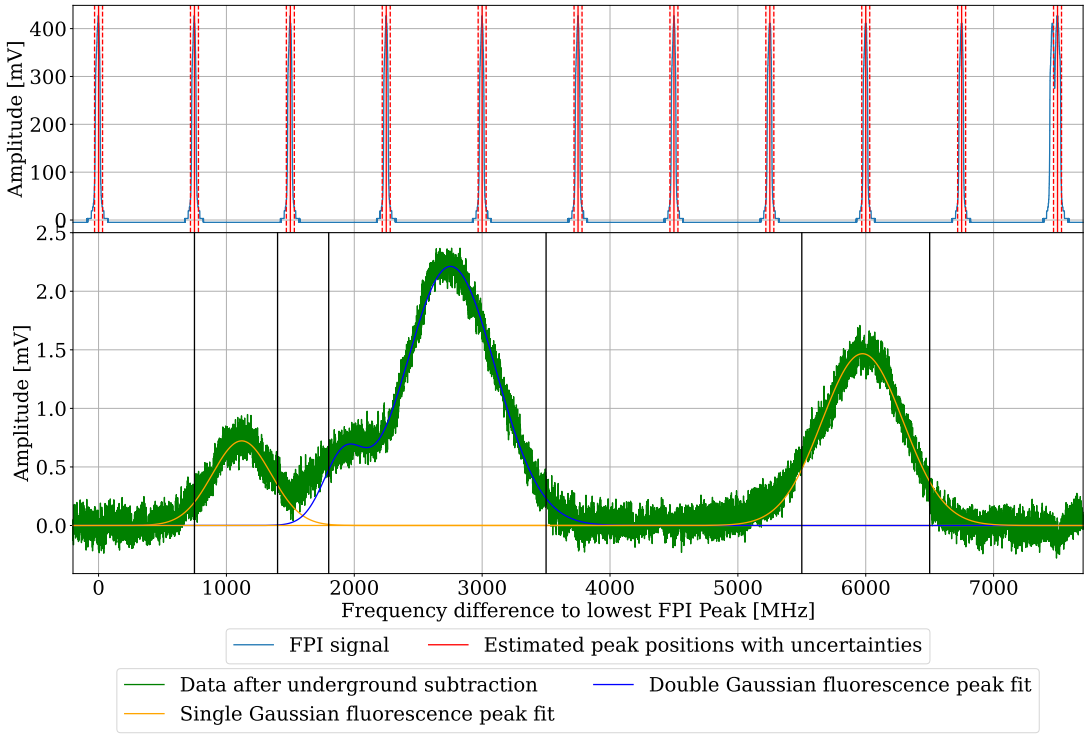


Fig. 25: In the upper plot the FPI spectrum is shown and in the lower plot the fluorescence spectrum is displayed. Both spectra are read out with the oscilloscope. The channel frequency-conversion is performed to plot the spectra on a frequency axis. Additionally for the upper plot, the identified peaks for the FPI, together with the estimated uncertainties, are shown. The peaks are now equidistant. For the fluorescence spectroscopy plot, a moving average is performed to smoothen the noisy data and the linear background is subtracted. The first and fourth peak are fitted with a singular Gaussian from Equation 26, which is displayed in orange. The second and third peak are fitted using the double Gaussian fit function from Equation 27, which is displayed in blue. The black lines indicate the area in which the data contributes to the fit.

The reduced χ^2-values for the fits are once again in a desired range with values ranging from 0.28 to 0.37. Again, the relevant fitting parameters for further calculations, namely the amplitude and the mean, are summarised with their uncertainties and the corresponding χ^2_{red}-values in Table 13 in the attachment. Analogously, the uncertainty of the amplitude is directly calculated from the covariance matrix, whereas for the uncertainty of the mean, the standard error is calculated and then propagated with the uncertainty of the frequency (see Equation 30). Again, it is noticeable, that the uncertainties on the amplitudes are all on the order of a few μV, which is a significant underestimation of the true uncertainty. This will also be discussed later.

4.4 Saturation spectroscopy

After optimizing and fine-tuning the setup and the laser, the following laser parameters are selected for the saturation spectroscopy: temperature: 24.0 °C, scan offset: 20.467 V, scan amplitude: 21.66 V_{pp}, set current: 164.42 mA, feed forward: −0.72 and a scan frequency of 10 Hz.

Different to before, the photodiode data is gathered with the laser controller software, since the saturation spectrum can be seen better there, compared to the oscilloscope. Thus, the photodiode data does not need an moving average, since the gathered data is less noisy and shows no

double line effects. Therefore, the estimated uncertainty on this data is 0.1 % of each data point. Due to the different gathering, a different number of channels is present (1018 channel). The FPI peaks, see Figure 39 in the attachment, are again found using `scipy.signal.find_peaks` [17] and the uncertainties are estimated to be ±2 channels. Nevertheless, the procedure for the channel-frequency conversion explained in subsection 4.2 is the same. This is done by using the data of the FPI peaks, see Figure 39 in the attachment. The plot for the spectrum on a frequency axis can be found in Figure 40 in the attachment.
As explained in subsection 4.2 a linear fit (see Equation 23) is performed, to subtract the linear background arising from the frequency changes created by the laser controller. For that, data outside the rubidium spectrum and between the two main peaks (where no peak is expected) is selected, which is indicated in orange, see Figure 23. The linear fit, together with the absorption spectrum, is shown in Figure 23. This fit is then subtracted to correct the data.

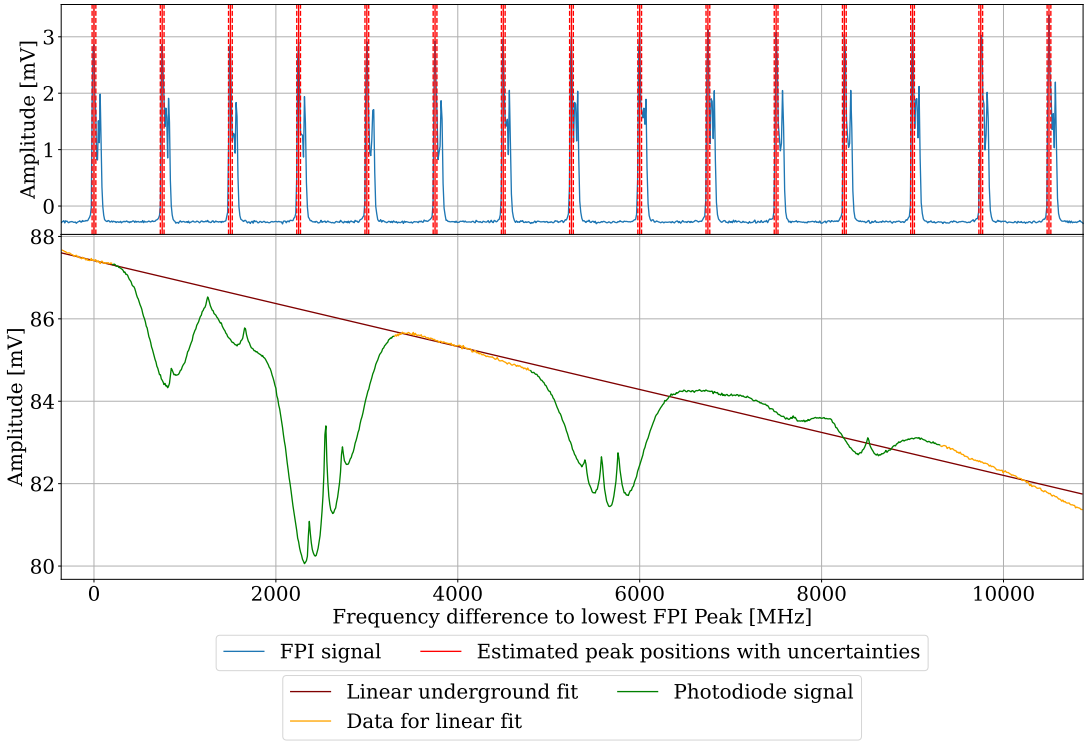


Fig. 26: In the upper plot the FPI spectrum is shown and in the lower plot the saturation spectrum is displayed. Both spectra are read out with the laser controller. The channel frequency-conversion is performed to plot the spectra on a frequency axis. Additionally for the upper plot, the identified peaks for the FPI, together with the estimated uncertainties, are shown. The peaks are now equidistant. To subtract the linear background from the saturation spectroscopy data, a linear fit (brown) is performed with the orange data.

Once again, it is noticeable, that the underground is not linear. Nevertheless, this only changes the amplitudes of the peaks and not the peak positions. Possible reasons for the deviation from the expected linear background will be discussed later.
The background corrected saturation spectrum can be found in Figure 27.

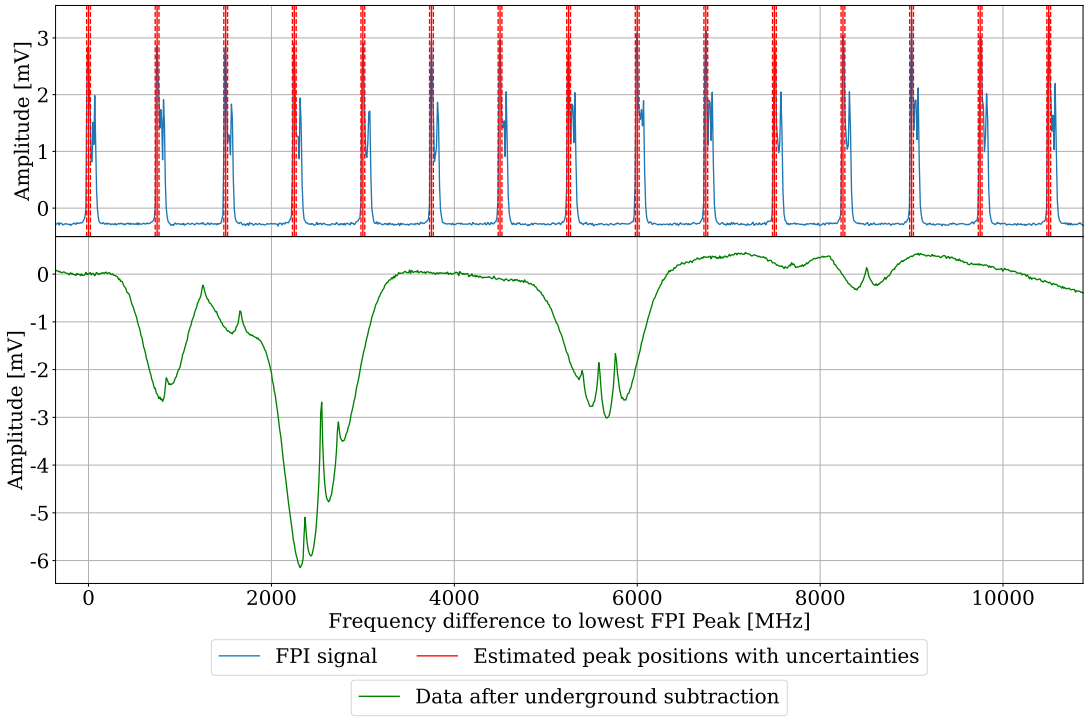


Fig. 27: In the upper plot the FPI spectrum is shown and in the lower plot the saturation spectrum is displayed. Both spectra are read out with the laser controller. The channel frequency-conversion is performed to plot the spectra on a frequency axis. Additionally for the upper plot, the identified peaks for the FPI, together with the estimated uncertainties, are shown. The peaks are now equidistant. For the saturation spectroscopy plot the linear background is subtracted.

As already mentioned in subsubsection 2.2.3, the saturation peaks are of a Lorentzian form. To fit the saturation peaks, the following Lorentzian formula introduced in Equation 12 is used and fitted with `scipy.optimize.curve_fit` [17]:

$$L(\nu) = \frac{A}{1 + \left(\frac{\nu - \nu_{\text{mean}}}{\gamma}\right)^2} + L_0. \tag{32}$$

Again, the data is restricted to an area around the peaks for better fits. In Figure 28 the background corrected saturation spectrum is displayed. Additionally, the performed Lorentzian fits are displayed in orange. Furthermore, the respective restricted areas are indicated by the width of each fit.

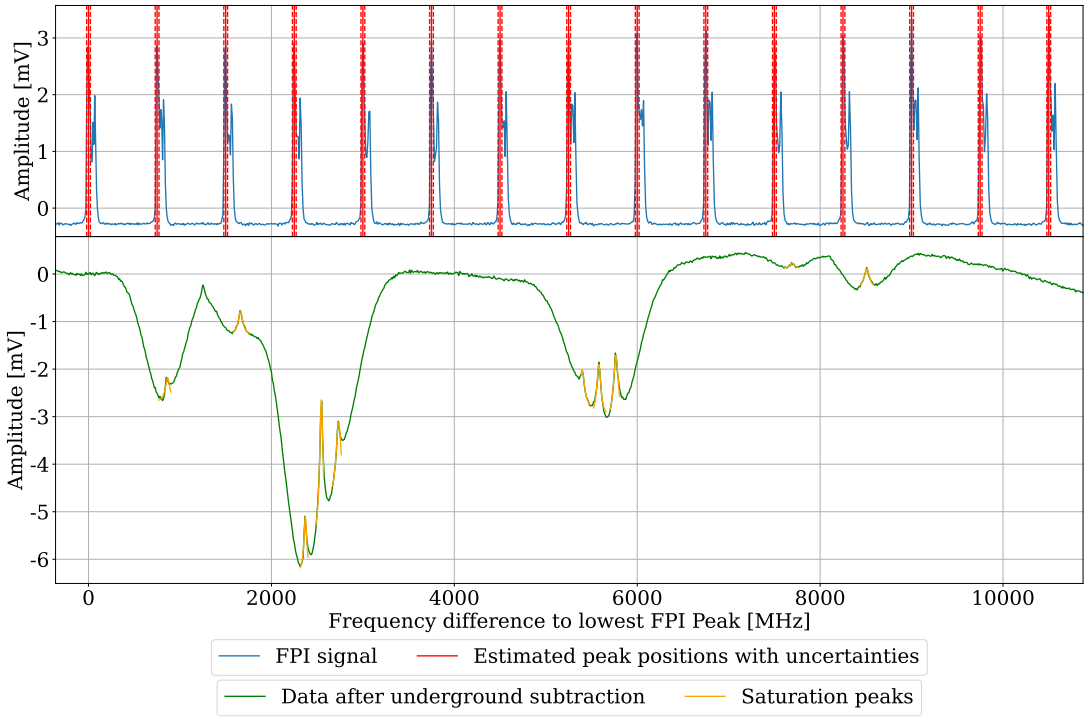


Fig. 28: In the upper plot the FPI spectrum is shown and in the lower plot the saturation spectrum is displayed. Both spectra are read out with the laser controller. The channel frequency-conversion is performed to plot the spectra on a frequency axis. Additionally for the upper plot, the identified peaks for the FPI, together with the estimated uncertainties, are shown. The peaks are now equidistant. For the saturation spectroscopy plot the linear background is subtracted. The peaks are fitted with Lorentzian distribution from Equation 12, which is displayed in orange. The areas in which the data contributes to the fit are indicated by the width of each fit.

The reduced χ^2-values for the fits are once again in a good range with values ranging from 0.05 to 1.30, with the reduced χ^2-value of 0.05 indicating the overestimation of uncertainties. Again, the relevant fitting parameters for further calculations, namely the amplitude and the mean, are summarised with their uncertainties and the corresponding χ^2_{red}-values in Table 14 in the attachment. Analogously, the uncertainty of the amplitude is directly calculated from the covariance matrix, whereas for the uncertainty of the mean, the standard error is calculated and then propagated with the uncertainty of the frequency. Different to the other spectroscopy measurements, the uncertainties for the amplitudes are now in a reasonable range.

4.5 Comparison of the spectroscopy results

With the performed spectroscopy measurements, several physical quantities can be calculated for each method, respectively. In the following, the hyperfine structure of ^{85}Rb and ^{87}Rb is studied, the composition of the two rubidium isotopes is analysed and the hyperfine constants of the two isotopes are calculated. In addition to that, all values are compared with literature values.

4.5.1 Identification of the ^{85}Rb and ^{87}Rb transitions

With the results obtained with the three spectroscopy methods, it is possible to identify the transitions of ^{85}Rb and ^{87}Rb. From the spectroscopy measurements, only relative frequency

differences can be deducted. As a consequence, the frequency differences to the lowest peak, identified as $^{87}\mathrm{Rb}_{F=2}^{F=1}$, are calculated:

$$\delta\nu\left(\mathrm{Rb}_{F=j}^{F=i}\right) = \nu\left(\mathrm{Rb}_{F=j}^{F=i}\right) - \nu\left(^{87}\mathrm{Rb}_{F=2}^{F=1}\right), \tag{33}$$

$$\Delta(\delta\nu) = \sqrt{\Delta\left(\nu\left(\mathrm{Rb}_{F=j}^{F=i}\right)\right)^2 + \Delta\left(\nu\left(^{87}\mathrm{Rb}_{F=2}^{F=1}\right)\right)^2}. \tag{34}$$

The measured frequency differences, together with the literature transitions, are displayed in Figure 29.

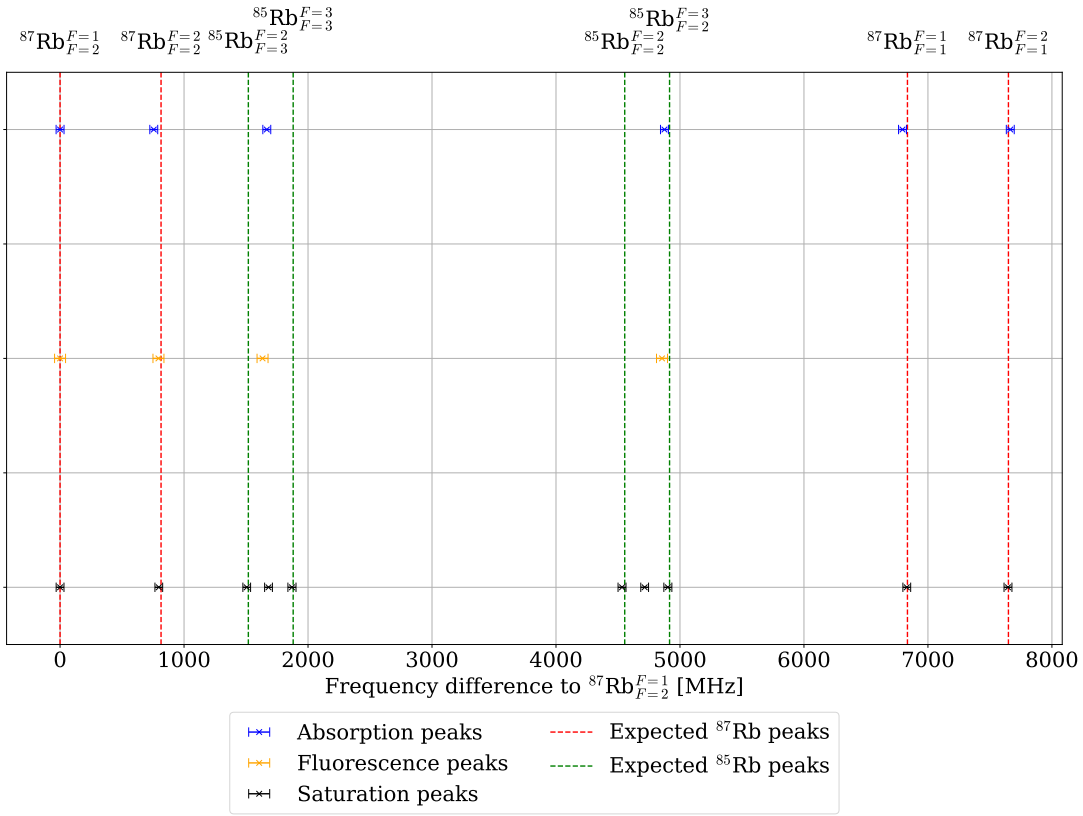


Fig. 29: The Rb spectroscopy results for the frequency difference to the lowest peak, $^{87}\mathrm{Rb}_{F=2}^{F=1}$, are compared in this plot. The transitions are labelled according to Equation 11. The expected frequency differences of ^{87}Rb and ^{85}Rb are shown in red and green. The uncertainties for the different spectroscopy methods are also shown.

It is visible, that most of the found peaks are in good accordance to theoretically expected transitions. For the absorption and fluorescence spectroscopy, the expected ^{85}Rb peaks could not be identified with either of the two theoretically expected peaks. One of the reasons for that is the Doppler-broadening, which leads to a significant overlap of the peaks in these measurement regimes. For the saturation spectroscopy, a cross-over resonance appears, as already explained in subsubsection 2.2.3. It is found to be exactly in the middle of the two close transitions of ^{85}Rb, which reinforces its identification as the cross-over. These cross-overs will not be evaluated in the further analysis.

The identified measured transitions are summarised in Table 2. For further references, the transitions are labelled by T_1 to T_8.

Tab. 2: The literature values for the hyperfine transitions of ^{85}Rb [7] and ^{87}Rb [8], and the measured transitions from spectroscopy are displayed. The transitions are labelled according to Equation 11 and enumerated by T_1 to T_8. The frequency differences compared to the $^{87}\mathrm{Rb}_{F=2}^{F=1}$ transition are given in MHz. The cross-over resonances are not distinct atomic level transitions as explained in subsubsection 2.2.3.

Transition	Expected [MHz]	Saturation [MHz]	Absorption [MHz]	Fluorescence [MHz]
T_1: $^{87}\mathrm{Rb}_{F=2}^{F=1}$	0	0 ± 45	0 ± 32	0 ± 44
T_2: $^{87}\mathrm{Rb}_{F=2}^{F=2}$	814.5	796 ± 45	756 ± 32	794 ± 44
T_3: $^{85}\mathrm{Rb}_{F=3}^{F=2}$	1518.56	1506 ± 45	1667 ± 32	1633 ± 44
Cross-over	1699.36	1682 ± 45		
T_4: $^{85}\mathrm{Rb}_{F=3}^{F=3}$	1880.15	1870 ± 46		
T_5: $^{85}\mathrm{Rb}_{F=2}^{F=2}$	4554.3	4532 ± 46	4874 ± 32	4855 ± 44
Cross-over	4735.09	4715 ± 45		
T_6: $^{85}\mathrm{Rb}_{F=2}^{F=3}$	4915.88	4903 ± 45		
T_7: $^{87}\mathrm{Rb}_{F=1}^{F=1}$	6834.68	6829 ± 45	6795 ± 32	-
T_8: $^{87}\mathrm{Rb}_{F=1}^{F=2}$	7649.18	7645 ± 46	7664 ± 32	-

4.5.2 Estimation of the Rb composition in the cell

The Rb composition of the cell can be estimated from the amplitude-ratio of two corresponding ^{85}Rb and ^{87}Rb peaks. The transitions with comparable transition probabilities between ^{85}Rb and ^{87}Rb are (T_1 and T_3), (T_2 and T_4), (T_5 and T_7) and (T_6 and T_8). For the saturation spectroscopy, the following amplitude ratios can be estimated to get the percentage of ^{87}Rb in the cell:

$$P_{87} \approx \frac{A_1}{A_1 + A_3}, \tag{35}$$

$$P_{87} \approx \frac{A_2}{A_2 + A_4}, \tag{36}$$

$$P_{87} \approx \frac{A_7}{A_5 + A_7}, \tag{37}$$

$$P_{87} \approx \frac{A_8}{A_6 + A_8}. \tag{38}$$

The A_i describe the amplitude of the corresponding transition T_i.The uncertainties are calculated as exemplary shown for Equation 35:

$$\Delta P_{87} = \frac{1}{(A_1 + A_3)^2} \sqrt{(A_1 \cdot \Delta A_3)^2 + (A_3 \cdot \Delta A_1)^2}. \tag{39}$$

For the fluorescence spectroscopy, further considerations need to be made. As can be seen in Table 2, the T_3 and T_4 transitions could not be separated. As a consequence, one can use the measured value and apply it to Equation 35 and Equation 36 and combine them in the following

way:

$$P_{87} \approx \frac{A_1 + A_2}{A_1 + A_2 + A_x + A_x}, \tag{40}$$

with A_x being the not separable and not identified transition. The uncertainties are calculated as follows:

$$\Delta P_{87} = \frac{1}{(A_1 + A_2 + A_x + A_x)^2} \sqrt{(2A_x)^2 \cdot \Delta A_1^2 + (2A_x)^2 \cdot \Delta A_2^2 + (2(A_1 + A_2))^2 \cdot 2 \cdot \Delta A_x^2}. \tag{41}$$

Analogous considerations for the absorption spectroscopy can be made, which lead to:

$$P_{87} \approx \frac{A_7 + A_8}{A_7 + A_7 + A_x + A_x}, \tag{42}$$

with A_x again being the not separable and not identified transition. The uncertainties are calculated analogously as for the other abundances. A summary of the calculated relative abundances of ^{87}Rb can be found in Table 3.

Tab. 3: Percentage of ^{87}Rb in the glass cell, calculated with Equation 35 to Equation 42 for the respective spectra.

Spectrum	Equation	^{87}Rb percentage in cell [%]
Saturation	Equation 35	30 ± 4
Saturation	Equation 36	22 ± 3
Saturation	Equation 37	11 ± 8
Saturation	Equation 38	23 ± 4
Absorption	Equation 40	25.599 ± 0.010
Absorption	Equation 42	22.035 ± 0.012
Fluorescence	Equation 40	22.27 ± 0.05

It is noticeable, that the uncertainties on the ^{87}Rb abundances are very small and underestimating the true uncertainty. This effect comes from the already small uncertainties on the amplitudes, obtained by the fits to the absorption and fluorescence spectra. For the saturation spectroscopy, the large relative uncertainty on the third value is prominent. A possible reason for that is the relatively small amplitude of T_7, which has a large relative uncertainty. The other values from the saturation spectroscopy are in good accordance with the literature value of $(27.83 \pm 0.20)\,\%$ [8].

4.5.3 Determination of the hyperfine constant

Finally, the hyperfine structure constants of ^{85}Rb and ^{87}Rb can be calculated from the frequency difference between two consecutive peaks. Each pair of consecutive peaks is separated in Table 2 by the horizontal blue lines. As explained in the theory part, the calculation is given by

$$A = h \frac{\delta\nu}{F_1 + 1}, \tag{43}$$

where $\Delta\nu$ is the frequency-difference between the two consecutive peaks and F_1 is the smaller of the two quantum numbers F.

As can be seen in Table 2, the hyperfine constant for ^{85}Rb can only be calculated with the saturation measurement, whereas for ^{87}Rb, also the fluorescence and absorption measurements can be used. The calculated hyperfine constants, together with the literature values are shown in Table 4. The uncertainties on the hyperfine structure are again calculated by Gaussian propagation of uncertainties.

Tab. 4: The literature values for the hyperfine constants of ^{85}Rb and ^{87}Rb [7, 8], and the calculated hyperfine constants from spectroscopy are displayed. All values are given in MHz with h being Planck's constant. The values are ordered according to the separation of the values needed for calculation in Table 2.

	Literature [MHz]	Saturation [MHz]	Absorption [MHz]	Fluorescence [MHz]
A_{87}	$h \cdot 407.25$	$h \cdot 398 \pm 16$	$h \cdot 378 \pm 23$	$h \cdot 396 \pm 31$
A_{85}	$h \cdot 120.53$	$h \cdot 122 \pm 11$	-	-
A_{85}	$h \cdot 120.53$	$h \cdot 123 \pm 11$	-	-
A_{87}	$h \cdot 407.25$	$h \cdot 408 \pm 16$	$h \cdot 435 \pm 23$	-

All the listed results are in good accordance with the literature values.

4.6 Lock-in frequency stability measurement

The frequency stability of the locked in laser is measured as explained in subsubsection 3.6.1. In the analysis, peaks are found using `scipy.signal.find_peaks` [17] and the uncertainties are estimated to be ±2 channels. For that, four measurements over the course of 10 minutes are taken. An exemplary measurement can be found in Figure 30. For the other three measurements, the plots can be found in the attachment in Figure 41, Figure 42 and Figure 43.

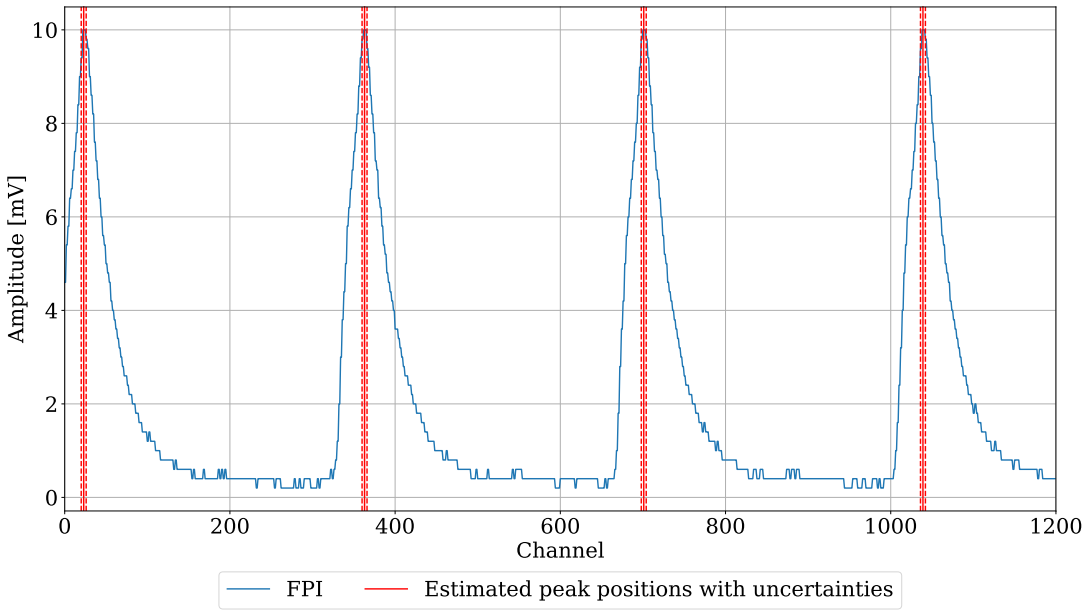


Fig. 30: The FPI measurement of the locked-in frequency. The FPI was put into resonance with the locked-in frequency with a piezo element.

The differences between two consecutive peaks are found for each measurement and the uncertainty is propagated using Gaussian propagation of uncertainties. The results of the differences can be found in Table 5.

Tab. 5: In this table the channel differences between two consecutive FPI peaks for the four frequency stability measurements are summarised. The indices indicate the peak number.

	c_2-c_1	c_3-c_2	c_4-c_3
Meas. 1	340 ± 4	338 ± 4	338 ± 4
Meas. 2	338 ± 4	338 ± 4	340 ± 4
Meas. 3	339 ± 4	339 ± 4	338 ± 4
Meas. 4	339 ± 4	339 ± 4	338 ± 4

It can be directly seen, that the frequency is stable over a period of at least 10 minutes.

4.7 Beat note measurement

We set one AOM on a fixed frequency. The frequency applied to the other AOM is varied. The resulting difference in frequency can be measured as a beat note when measuring the frequency with the laser controller. Again, the peaks are found using `scipy.signal.find_peaks` [17] and the uncertainties are estimated to be ±10 Hz. An exemplary measurement can be found in Figure 31. For the other nine measurements the plots can be found in the attachment in Figure 44 to Figure 52.

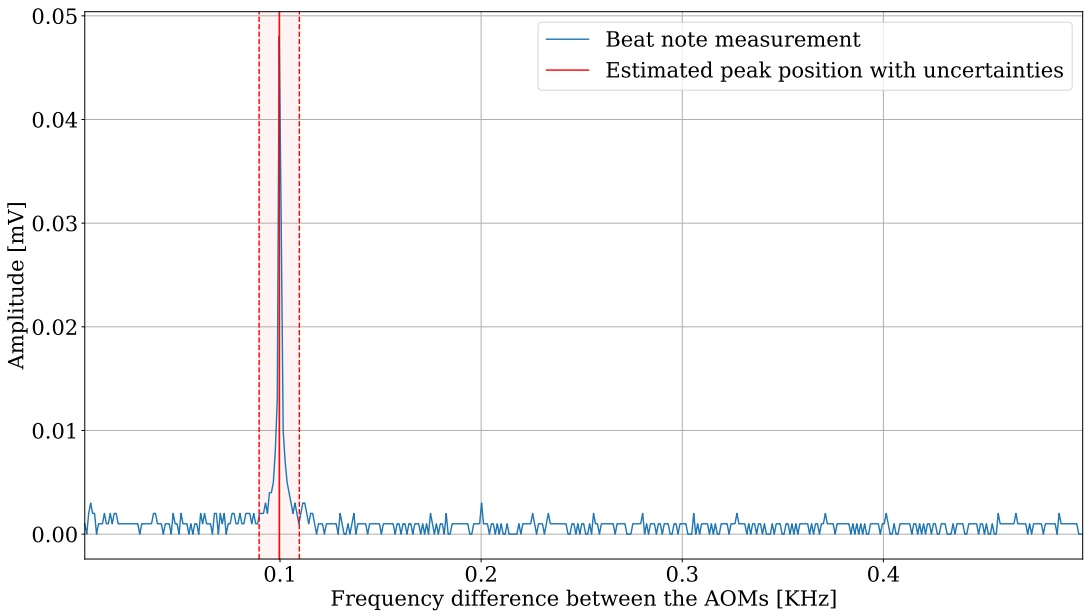


Fig. 31: The beat note measurement for a set frequency difference of the AOMs of 0.1 kHz. Additionally, the found peak and its uncertainty is shown in red.

The found beat note frequencies are summarised with the set frequency differences of the AOMs in Table 6.

Tab. 6: In this table, the set frequency differences between the AOMs and the measured beat note frequencies are summarised and given in KHz. The values are ordered by the measurement number.

Frequency difference between the AOMs [KHz]	Beat note frequencies [KHz]
0.1	0.10 ± 0.01
0.2	0.20 ± 0.01
0.3	0.30 ± 0.01
3	3.00 ± 0.01
16	16.02 ± 0.01
186	186.20 ± 0.01
240	240.15 ± 0.01
109	108.97 ± 0.01
69.42	69.30 ± 0.01
0.01	0.01 ± 0.01

One can directly see from the table, that for small frequency differences in the AOMs, the beat note frequency shows almost no deviation from the expectation. For frequency differences above approximately 60 kHz, the measured beat note is slightly off. The relative deviation from the expectation is small, but the uncertainties on the peak positions are underestimated for these higher frequency differences.

Despite that, the found beat note frequencies can then be plotted against the set frequency differences between the two AOMs, to study the behaviour of the AOMs in relation to their frequency difference. Such a plot can be found in Figure 32. Additionally, a linear fit of the form:

$$f_{\text{beat}}(\delta\nu_{\text{AOM}}) = a \cdot \delta\nu_{\text{AOM}} + b, \tag{44}$$

with f_{beat} the beat note frequency and $\delta\nu_{\text{AOM}}$ the difference between the AOM frequencies, is performed using `scipy.optimize.curve_fit` [17].

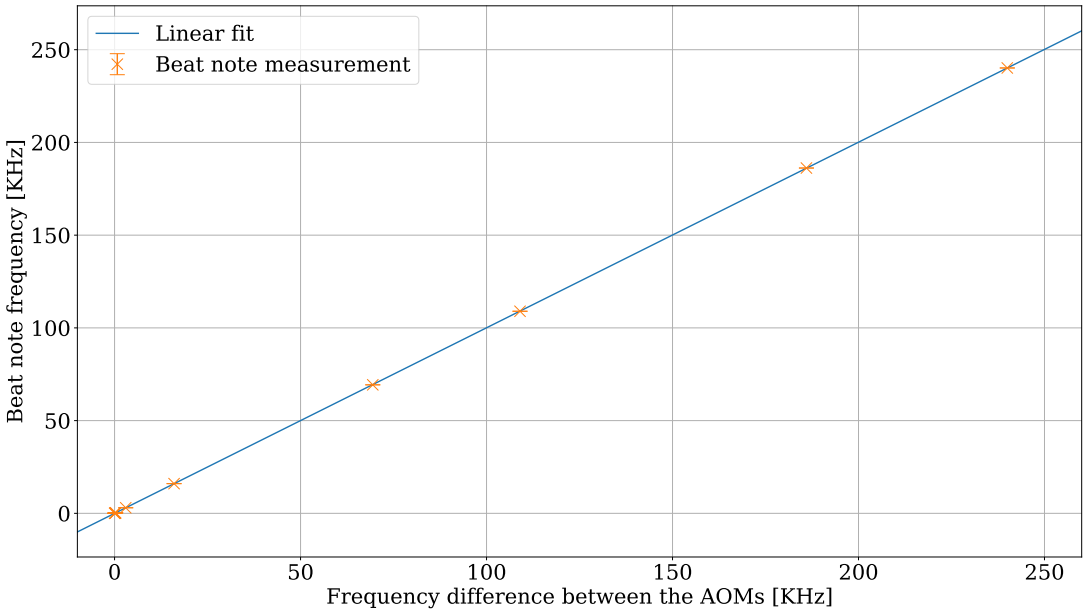


Fig. 32: The beat note frequencies are plotted against the set frequency differences of the AOMs. Additionally, a linear fit of the data is performed.

The found parameters for the linear fit are found to be:

$$a = (1.000\,70 \pm 0.000\,04)\,\text{kHz}, \qquad b = (-0.023 \pm 0.004)\,\text{kHz}. \tag{45}$$

From this, it can be concluded, that the linear relation between the frequency difference between the AOMs and the beat note holds true. From the fit parameters, it can also be seen, that there is a small offset $b = (-0.023 \pm 0.004)\,\text{kHz}$, which is not compatible with 0 Hz as expected. This is probably due to the underestimation of the uncertainties for higher beat note frequencies.

5 Discussion

At the end of this protocol, first, a brief summary with all the relevant steps and important results is given. After this, there is a more in depth discussion about problematic parts in the experiment.

5.1 Summary of the results

In the first part of this experiment, three different spectroscopy methods were used to analyse a sample filled with ^{85}Rb and ^{87}Rb gas. The three methods used are fluorescence, absorption and saturation spectroscopy. From the three different measurements, the energy structure of the two rubidium isotopes could be determined for each method. In addition, also the composition of the gas was determined, as well as the hyperfine constants for ^{85}Rb and ^{87}Rb.

Due to the setup, only frequency differences between the peaks could be measured. Here, the $^{87}\mathrm{Rb}_{F=2}^{F=1}$ transition is set as the reference point, since this transition has the smallest frequency. In Table 7 the frequency differences for the hyperfine transitions are shown for all three methods. For comparison, also the literature values are given. As can be seen, the measured values for the ^{87}Rb transitions are in good compatibility with literature, for all methods. Due to the Doppler broadening, the adjacent peaks of ^{85}Rb could not be differentiated for absorption and fluorescence spectroscopy and therefore lie in between the two expected transitions. For saturation spectroscopy, they can be distinguished, as well as the cross-over resonance, which is a relict of the saturation spectroscopy method. For saturation spectroscopy, all measurements are in good accordance with literature. For fluorescence spectroscopy, the transitions T_7 and T_8 were not found.

Tab. 7: The literature values for the hyperfine transitions of ^{85}Rb [7] and ^{87}Rb [8], and the measured transitions from spectroscopy are displayed. The transitions are labelled according to Equation 11 and enumerated by T_1 to T_8. The frequency differences compared to the $^{87}\mathrm{Rb}_{F=2}^{F=1}$ transition are given in MHz. The cross-over resonances are not distinct atomic level transitions as explained in subsubsection 2.2.3.

Transition	Expected [MHz]	Saturation [MHz]	Absorption [MHz]	Fluorescence [MHz]
T_1: $^{87}\mathrm{Rb}_{F=2}^{F=1}$	0	0 ± 45	0 ± 32	0 ± 44
T_2: $^{87}\mathrm{Rb}_{F=2}^{F=2}$	814.5	796 ± 45	756 ± 32	794 ± 44
T_3: $^{85}\mathrm{Rb}_{F=3}^{F=2}$	1518.56	1506 ± 45	1667 ± 32	1633 ± 44
Cross-over	1699.36	1682 ± 45		
T_4: $^{85}\mathrm{Rb}_{F=3}^{F=3}$	1880.15	1870 ± 46		
T_5: $^{85}\mathrm{Rb}_{F=2}^{F=2}$	4554.3	4532 ± 46	4874 ± 32	4855 ± 44
Cross-over	4735.09	4715 ± 45		
T_6: $^{85}\mathrm{Rb}_{F=2}^{F=3}$	4915.88	4903 ± 45		
T_7: $^{87}\mathrm{Rb}_{F=1}^{F=1}$	6834.68	6829 ± 45	6795 ± 32	-
T_8: $^{87}\mathrm{Rb}_{F=1}^{F=2}$	7649.18	7645 ± 46	7664 ± 32	-

From the heights of the measured peaks, the composition of rubidium in the sample could be

determined. The found values can be seen in Table 8. The literature value for the percentage of ^{87}Rb is given with $(27.83 \pm 0.20)\,\%$ [8]. As can be seen, the determined values for absorption and fluorescence spectroscopy obtain very small uncertainties, which are not reasonable. Possible reasons for this are discussed in the next section. Nevertheless, the determined values for saturation spectroscopy show very good accordance with literature, besides the third one. This is most likely due to the relative small amplitude, compared to the other peaks.

Tab. 8: Percentage of ^{87}Rb in the glass cell, calculated with Equation 35 to Equation 42 for the respective spectra.

Spectrum	Equation	^{87}Rb percentage in cell [%]
Saturation	Equation 35	30 ± 4
Saturation	Equation 36	22 ± 3
Saturation	Equation 37	11 ± 8
Saturation	Equation 38	23 ± 4
Absorption	Equation 40	25.599 ± 0.010
Absorption	Equation 42	22.035 ± 0.012
Fluorescence	Equation 40	22.27 ± 0.05

In a last analysis step of the spectroscopy data, the hyperfine constants for ^{85}Rb and ^{87}Rb were determined, using Equation 10. Again, all three methods were used. The results can be seen in Table 9. Due to the Doppler broadening, the transitions T_3, T_4 and T_5, T_6 could not be resolved. Therefore, the hyperfine constants for ^{85}Rb could not be determined with the data from absorption and fluorescence spectroscopy. With the data from saturation spectroscopy, also the hyperfine constant for ^{85}Rb could be determined. Nevertheless, all calculated constants are in good accordance with literature.

Tab. 9: The literature values for the hyperfine constants of ^{85}Rb and ^{87}Rb, and the calculated hyperfine constants from spectroscopy are displayed. All values are given in MHz with h being Planck's constant. The values are ordered according to the separation of the values needed for calculation in Table 2.

	Literature [MHz]	Saturation [MHz]	Absorption [MHz]	Fluorescence [MHz]
A_{87}	$h \cdot 407.25$	$h \cdot 398 \pm 16$	$h \cdot 378 \pm 23$	$h \cdot 396 \pm 31$
A_{85}	$h \cdot 120.53$	$h \cdot 122 \pm 11$	-	-
A_{85}	$h \cdot 120.53$	$h \cdot 123 \pm 11$	-	-
A_{87}	$h \cdot 407.25$	$h \cdot 408 \pm 16$	$h \cdot 435 \pm 23$	-

For an EIT measurement, the laser frequency needs to be stable. In order to achieve this, the laser controller can lock the laser frequency. In a frequency stability measurement, the duration of frequency stability was measured. This was done with the FPI. The frequency is stable if the measured FPI peaks show no drift, which means that the peak position differences are constant over time. Here, four measurements are performed over the course of 10 minutes between the

first and last measurement. The results for the peak position differences can be seen in Table 10. As can be seen, the peak position differences do not change over time. This implies, that the frequency is stable over at least 10 minutes.

Tab. 10: In this table the channel differences between two consecutive FPI peaks for the four frequency stability measurements are summarised. The indices indicate the peak number.

	c_2-c_1	c_3-c_2	c_4-c_3
Meas. 1	340 ± 4	338 ± 4	338 ± 4
Meas. 2	338 ± 4	338 ± 4	340 ± 4
Meas. 3	339 ± 4	339 ± 4	338 ± 4
Meas. 4	339 ± 4	339 ± 4	338 ± 4

Another preliminary measurement for EIT is the beat note measurement. This measurement is needed to study the frequency detuning between the two laser beams. A beat note occurred, since there were two separate beams, which realized different frequency shifts from the respective AOM. In Table 11 the set frequency differences between the AOMs, with the according beat note frequency, can be seen. It is expected, that the set frequency difference between the AOMs, directly refers to the measured beat note. As can be seen, for small differences, there is almost no deviation from the expectation. For higher differences, there are significant deviations, within the uncertainties. Possible causes are discussed in the next section. However, from the found values, a linear fit was performed, which can be used to calculate the beat note from the set frequency difference between the AOMs. The fit parameters are:

$$a = (1.000\,70 \pm 0.000\,04)\,\text{kHz}, \qquad b = (-0.023 \pm 0.004)\,\text{kHz},$$

indicating the expected linear behavior.

Tab. 11: In this table, the set frequency differences between the AOMs and the measured beat note frequencies are summarised and given in KHz. The values are ordered by the measurement number.

Frequency difference between the AOMs [KHz]	Beat note frequencies [KHz]
0.1	0.10 ± 0.01
0.2	0.20 ± 0.01
0.3	0.30 ± 0.01
3	3.00 ± 0.01
16	16.02 ± 0.01
186	186.20 ± 0.01
240	240.15 ± 0.01
109	108.97 ± 0.01
69.42	69.30 ± 0.01
0.01	0.01 ± 0.01

5.2 Discussion of the results

The measurement of the rubidium sample with three different spectroscopy methods all showed good accordance with the literature values for the hyperfine transition frequencies of rubidium. Also the advantages and disadvantages of the single methods can be seen. From the different analysis steps, it can be seen, that saturation spectroscopy delivers the best results, because it was able to resolve all expected hyperfine transitions, and additionally two cross-over transitions, which were used to identify the single transitions. Due to this resolution, the analysis of related physical quantities was enabled. Besides the fact, that the uncertainties on the transition with saturation spectroscopy are the highest, the measured values are the closest to the literature values. This indicates, that the uncertainties on the frequency differences might be too big. The reason for this could be the number of data points in the interval chosen for the Lorentzian fit. Here, for the most peaks, only around 20 data points are used. In order to lower the uncertainty, a higher resolution of the data should be used.
The absorption and fluorescence measurement showed good accordance for the found peaks. However, these two methods have problems in resolving closely separated transitions, as it is the case for the transitions T_3, T_4 and T_5, T_6. The reason for that is Doppler broadening. Here, the measured values just gave a rough estimate of the transition frequencies. Thus, if such closely spaced transitions should be resolved, saturation spectroscopy should be used instead. For measuring transitions, which are not effected by Doppler broadening, it can be better to use absorption or fluorescence spectroscopy instead of saturation spectroscopy, since setting those two methods up is easier and faster, resulting in measurement values within the same range of uncertainties.

Besides the benefits of each method, there is a clear systematic uncertainty, resulting from the non-linearity of the underground. Such a non-linearity can be seen for absorption and saturation spectroscopy, see Figure 20 and Figure 26. This systematic deviation in the underground do not affect the peak positions, but their heights. This systematic uncertainty is not respected in the error propagation. Thus, the uncertainties on the found abundances should be bigger than the ones presented. A possible reason for this is, that the triangular ramp, applied to the piezo element moving the grating, did not have a constant slope. This could be improved by changing some settings at the laser controller.
Another possible systematic uncertainty introduced for the absorption and fluorescence measurement is the usage of the oscilloscope as the read-out device. With the additional effect of the double line in the oscilloscope data, the signal-to-noise ratio is significantly worse. This leads to larger uncertainties on the amplitudes, which leads to the precision of the fits being significantly overestimated. To overcome this, the laser controller software can be used as the read-out device, since for that no double line effect were observed. Therefore, the signal-to-noise ratio is improved.

From the spectroscopy measurements, also the abundance of the two rubidium isotopes in the sample was estimated. As can be seen in Table 8, the uncertainties on the last three values definitely undershoot the real uncertainties. This is with high certainty due to the fits of the peaks. Since there are so many data points, which due to their uncertainty are compatible with the fits (see reduced χ^2 values), the uncertainty on the amplitude parameter is undershot by `scipy.optimize.curve_fit` by at least two orders of magnitude. This effect had no influence on the uncertainties of the mean values, since they are calculated not only from the result of the curve fit, but from an Gaussian error propagation, taking the uncertainties from the FPI calibration into account.

For the frequency stability measurement it could be seen, that the frequency was stable over the measurement duration of 10 minutes. To obtain a better understanding of the duration, longer measurements should be performed.

To guarantee the correct locking of the frequency, a second photodiode should be used, which is placed in the spectral measurement setup. With a second photodiode, the locking can be continuously ensured during an EIT or beat note measurement.

For the beat note measurement, the laser beam intensity was in the order of μW, which was due to the PMF. This lead to a very hard adjustment of the beam path. Here, an even better light to fibre coupling should have been achieved. Since no further improvements by adjusting the beam path were observed, the relative position of the PMF in the light to fibre coupler should be improved.
Nevertheless, the beat note measurement showed some underestimation of the uncertainties for higher beat note frequencies, since the uncertainty was estimated for all at once to 0.01 kHz.

References

[1] Michael Fleischhauer, Atac Imamoglu, and Jonathan P. Marangos. "Electromagnetically Induced Transparency: Optics in Coherent Media". In: *Reviews of Modern Physics* 77.2 (2005), pp. 633–673. DOI: 10.1103/RevModPhys.77.633.

[2] D Palani, D Hoenig, and L Karpa. "In situ ac Stark shift detection in light storage spectroscopy". In: *Journal of Physics B: Atomic, Molecular and Optical Physics* 54.16 (Sept. 2021), p. 165402. DOI: 10.1088/1361-6455/ac2001.

[3] Marin Soljacic et al. "Electromagnetically induced transparency in microcavities". In: *Nature* 422.6930 (2004), pp. 415–418. DOI: 10.1038/nature01597.

[4] Zhimin Shi et al. "Enhancing the spectral sensitivity of interferometers using slow-light media". In: *Optics Letters* 38.10 (2013), pp. 1764–1766. DOI: 10.1364/OL.38.001764.

[5] Christopher J. Foot. *Atomic Physics*. Oxford Master Series in Physics. Oxford: Oxford University Press, 2005. ISBN: 0-19-850695-3.

[6] Wolfgang Demtröder. *Laser Spectroscopy: Basic Concepts and Instrumentation*. 2nd. Springer Berlin Heidelberg, 1996. ISBN: 978-3-662-08262-1.

[7] Daniel A. Steck. *Rubidium 85 D Line Data*. Revision 2.3.3, 28 May 2024. URL: http://steck.us/alkalidata.

[8] Daniel A. Steck. *Rubidium 87 D Line Data*. Revision 2.3.3, 28 May 2024. URL: http://steck.us/alkalidata.

[9] Lcorman. *Example of Photodiode Response in a Saturated Absorption Experiment*. Wikimedia Commons. Labels in French. Wavelength axis not perfectly calibrated. 2008.

[10] TOPTICA Photonics AG. *DLC pro Digital Laser Controller Manual*. M-063 Version 16. Gräfelfing, Germany: TOPTICA Photonics AG, 2020.

[11] TOPTICA Photonics AG. *DL pro Grating Stabilized Diode Laser Head Manual*. M-30 Version 01. Lochhamer Schlag 19, D-82166 Graefelfing, Germany: TOPTICA Photonics AG, 2011.

[12] Patrick87. *Blazed Grating*. Wikimedia Commons. Diffraction at a blazed grating. 2012.

[13] Ricardo E Silva et al. "All-fiber 10 MHz acousto-optic modulator of a fiber Bragg grating at 1060 nm wavelength". In: *Opt. Express* 23.20 (Oct. 2015), pp. 25972–25978. DOI: 10.1364/OE.23.025972.

[14] Michael Lenz. *Prinzip eines AOMs*. Wikimedia Commons. acousto optical modulator. 2009.

[15] Omur Sezerman and Garland Best. "Accurate Alignment Preserves Polarization". In: *Laser Focus World* (1997). URL: https://www.ozoptics.com/ALLNEW_PDF/ART0001.pdf.

[16] Albert-Ludwigs-Universität Freiburg. *Electromagnetically induced transparency*. Freiburg im Breisgau 2023.

[17] Pauli Virtanen et al. "SciPy 1.0: Fundamental Algorithms for Scientific Computing in Python". In: *Nature Methods* 17 (2020), pp. 261–272. DOI: 10.1038/s41592-019-0686-2.

[18] Wes McKinney. *Data structures for statistical computing in Python*. 2010. URL: https://doi.org/10.25080/Majora-92bf1922-00.

[19] Vigneshdm1990. *Faraday isolator*. Wikimedia Commons. Faraday isolator allows the transmission of light in only one direction. 2020.

6 Attachment A

6.1 Tables and graphics

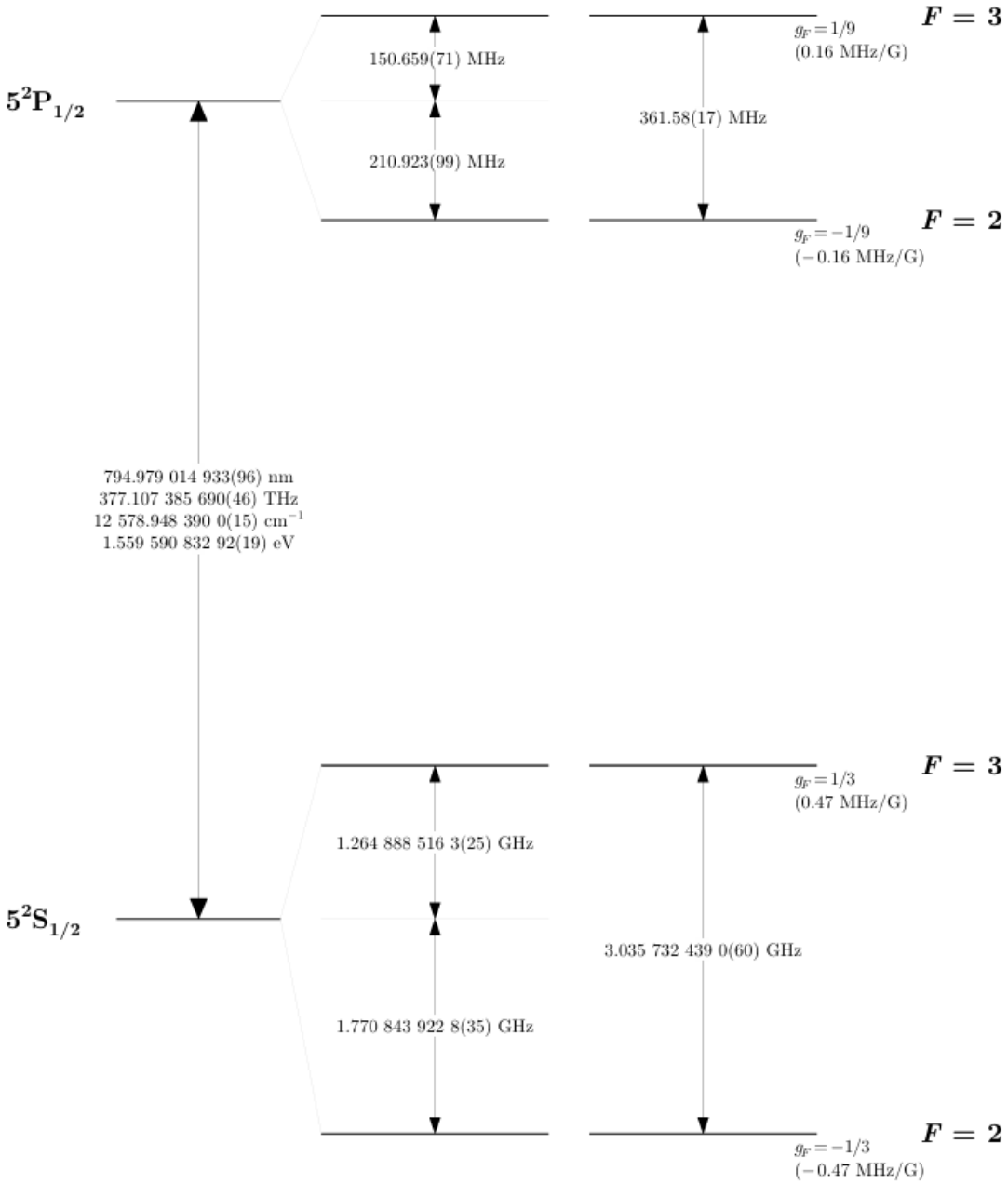


Fig. 33: Hyperfine structure of ^{85}Rb for the $5^2S_{1/2}$ and $5^2P_{1/2}$ levels. Also the D-1 line for ^{85}Rb is shown [7].

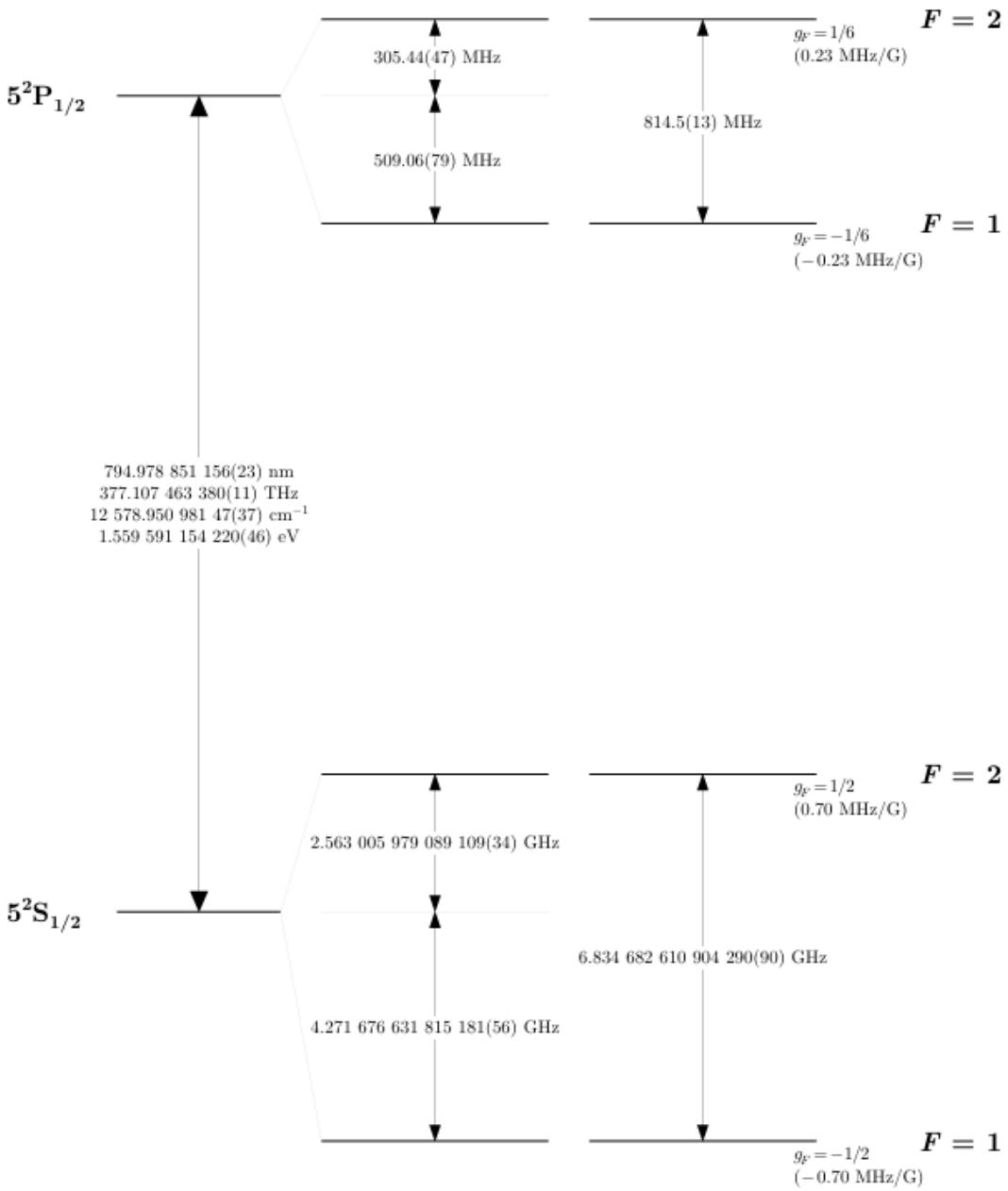


Fig. 34: Hyperfine structure of ^{87}Rb for the $5^2S_{1/2}$ and $5^2P_{1/2}$ levels. Also the D-1 line for ^{87}Rb is shown [8].

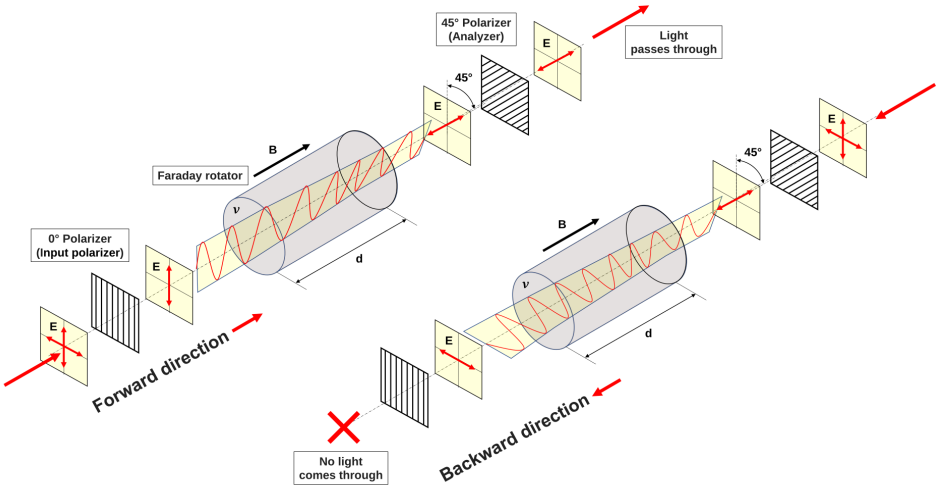


Fig. 35: Working principle of a Faraday isolator for blocking light from re-entering the laser cavity [19].

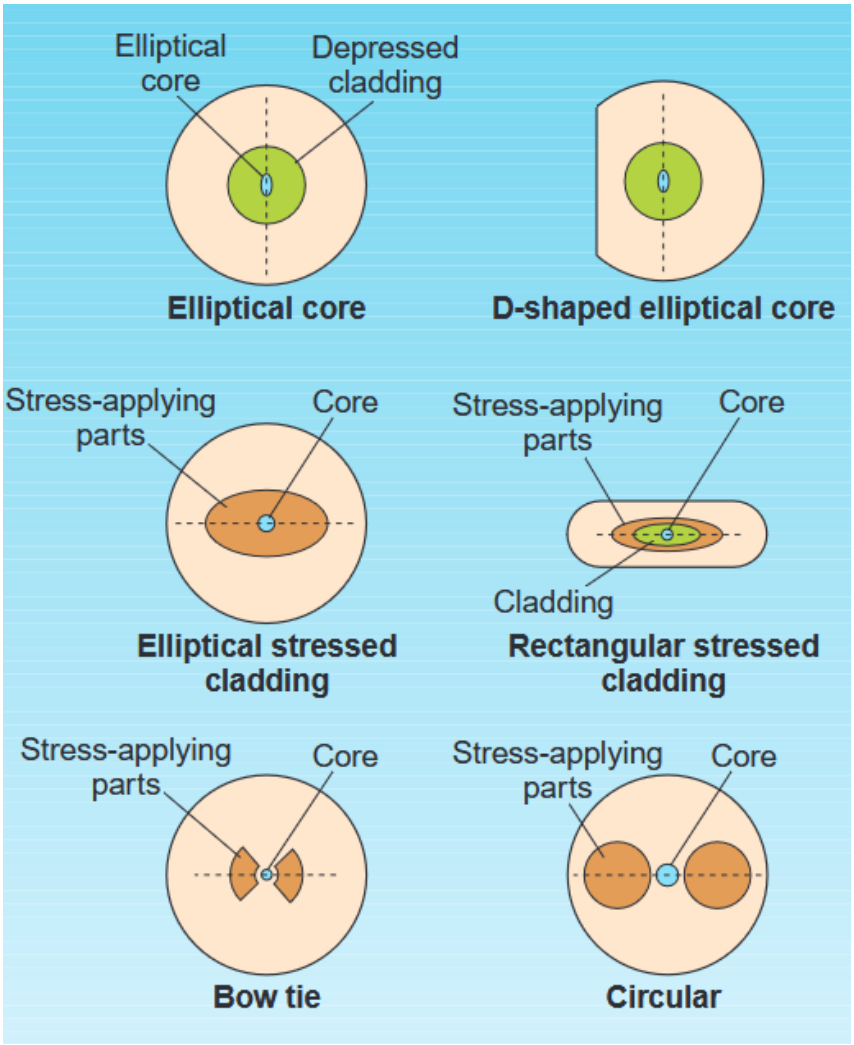


Fig. 36: Cross section of a PMF with different approaches. The dotted lines represent the slow axis [15].

Tab. 12: In this table, the fitted parameters for the amplitude and the mean for the absorption spectroscopy are shown. The first and fourth peak are fitted using a singular Gaussian distribution from Equation 26. The second and third, and the fifth and sixth peak are fitted using a double Gaussian model from Equation 27. The uncertainties for the amplitude are directly taken for the fit and the uncertainties on the mean are calculated by propagating the standard error with the frequency uncertainty, which was not considered during the fitting process. Additionally, all χ^2_{red}-values are shown.

Peak number	Amplitude [mV]	Mean (frequency to lowest FPI peak) [MHz]	χ^2_{red}
1	-7.458 ± 0.005	1194 ± 23	0.08
2	-6.912 ± 0.006	1950 ± 23	0.73
3	-20.882 ± 0.003	2861 ± 23	
4	-16.471 ± 0.004	6068 ± 23	0.33
5	-3.225 ± 0.004	7989 ± 23	0.13
6	-6.085 ± 0.004	8858 ± 23	

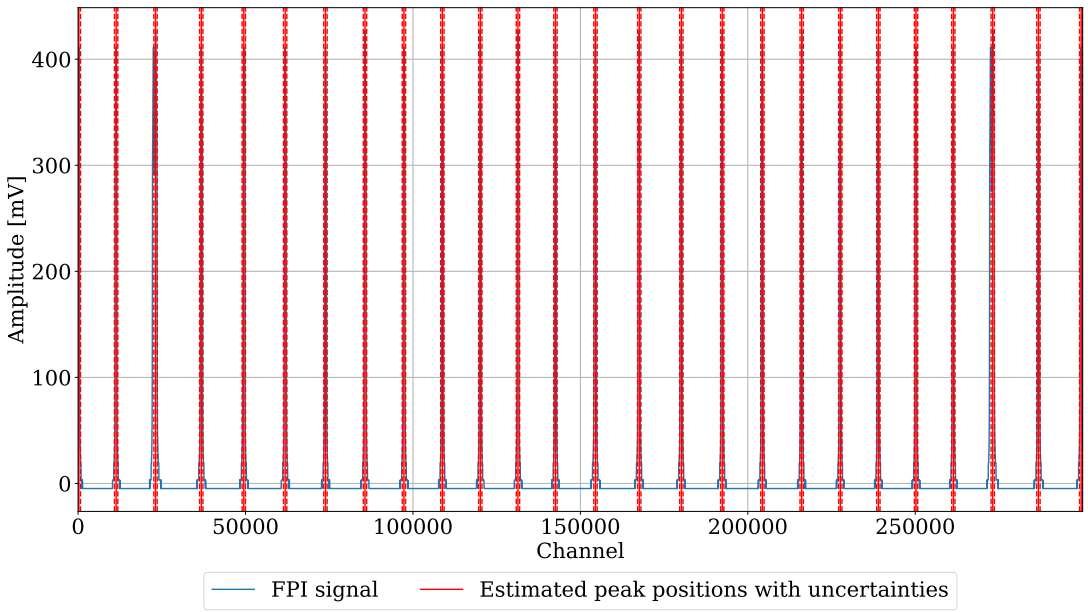


Fig. 37: In this plot the FPI spectrum, read out with the oscilloscope during the fluorescence spectroscopy measurement, is shown. Additionally, the identified peak channels, together with the estimated uncertainties of ±500 channels, are plotted in red. It can be seen, that the peak channels are not equidistant.

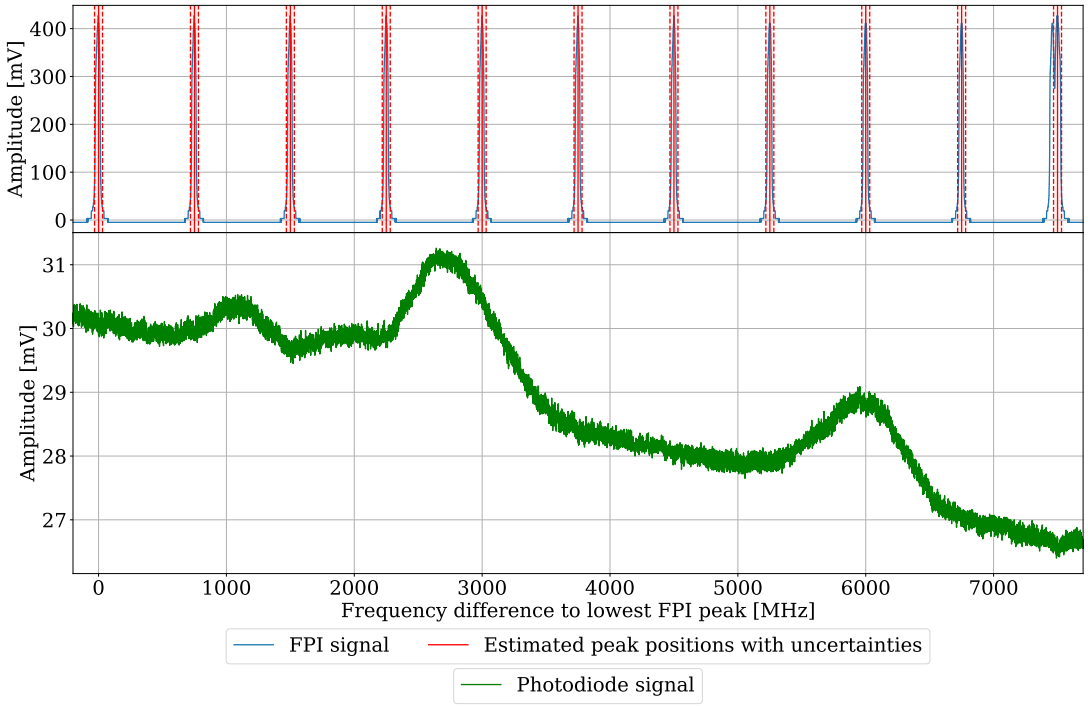


Fig. 38: In the upper plot the FPI spectrum is shown and in the lower plot the fluorescence spectrum is displayed. Both spectra are read out with the oscilloscope. The channel frequency-conversion is performed to plot the spectra on a frequency axis. Additionally for the upper plot, the identified peaks for the FPI, together with the estimated uncertainties, are shown. The peaks are now equidistant. For the fluorescence spectroscopy plot, a moving average is performed to smoothen the noisy data.

Tab. 13: In this table, the fitted parameters for the amplitude and the mean for the fluorescence spectroscopy are shown. The first and fourth peak are fitted using a singular Gaussian distribution from Equation 26. The second and third are fitted using a double Gaussian model from Equation 27. The uncertainties for the amplitude are directly taken for the fit and the uncertainties on the mean are calculated by propagating the standard error with the frequency uncertainty, which was not considered during the fitting process. Additionally, all χ^2_{red}-values are shown.

Peak number	Amplitude [mV]	Mean (frequency to lowest FPI peak) [MHz]	χ^2_{red}
1	0.723 ± 0.002	1120 ± 31	0.28
2	0.545 ± 0.003	1913 ± 31	0.32
3	2.212 ± 0.002	2753 ± 31	
4	1.467 ± 0.002	5975 ± 31	0.37

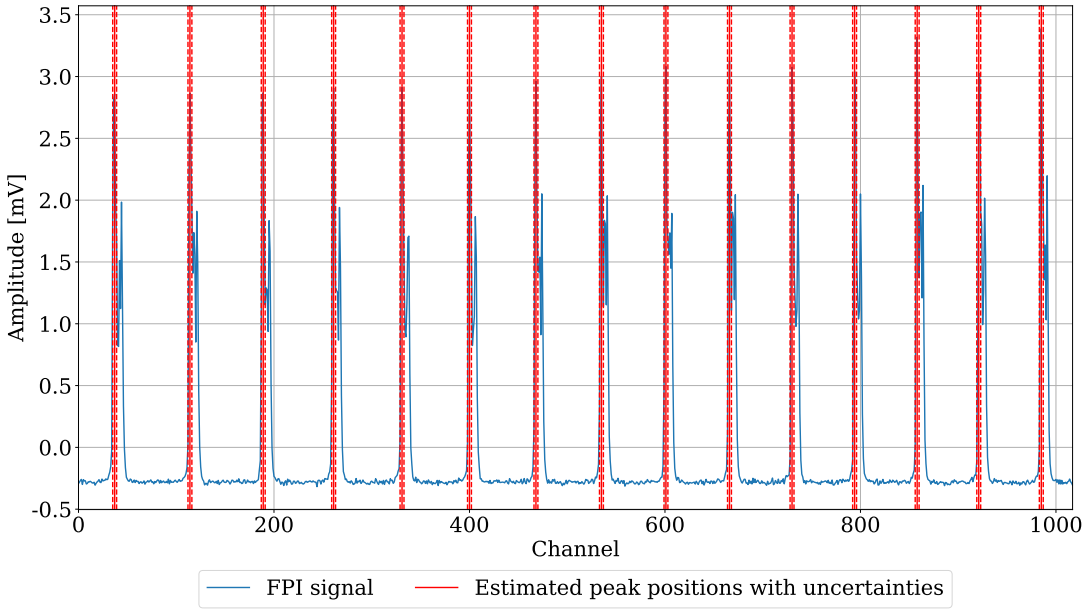


Fig. 39: In this plot the FPI spectrum, read out with the laser controller during the saturation spectroscopy measurement, is shown. Additionally, the identified peak channels, together with the estimated uncertainties of ±2 channels, are plotted in red. It can be seen, that the peak channels are not equidistant.

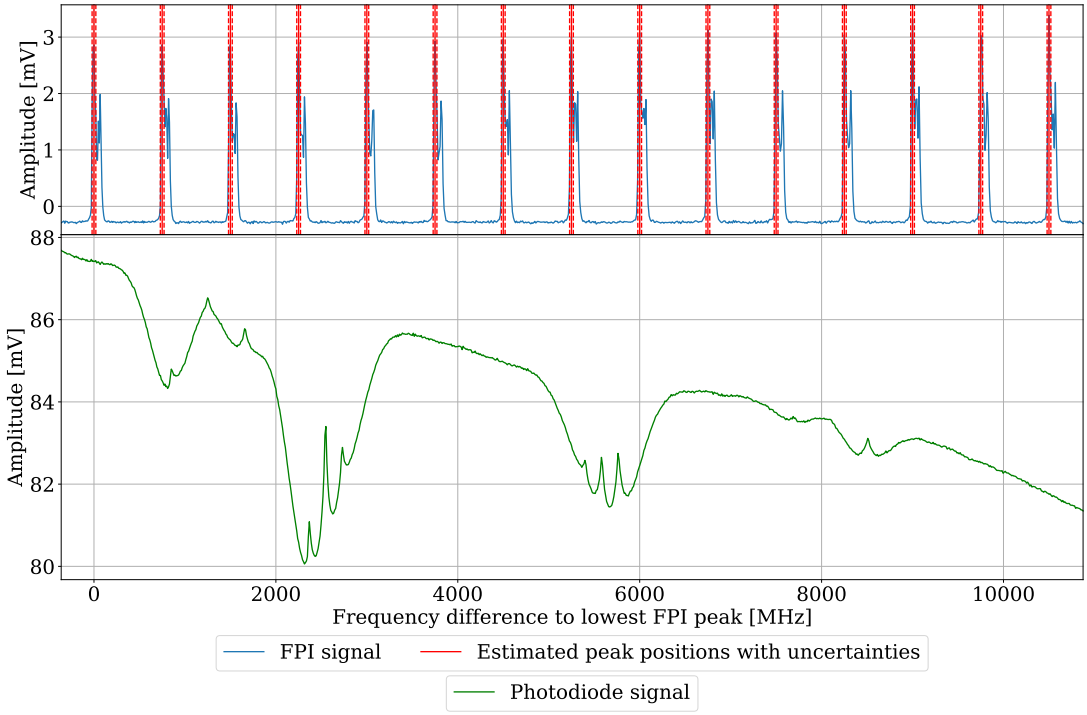


Fig. 40: In the upper plot the FPI spectrum is shown and in the lower plot the saturation spectrum is displayed. Both spectra are read out with the laser controller. The channel frequency-conversion is performed to plot the spectra on a frequency axis. Additionally for the upper plot, the identified peaks for the FPI, together with the estimated uncertainties, are shown. The peaks are now equidistant.

Tab. 14: In this table, the fitted parameters for the amplitude and the mean for the saturation spectroscopy are shown. The peaks are fitted using a Lorentzian distribution. The uncertainties for the amplitude are directly taken for the fit and the uncertainties on the mean are calculated by propagating the standard error with the frequency uncertainty, which was not considered during the fitting process. Additionally, all χ^2_{red}-values are shown.

Peak number	Amplitude [mV]	Mean (frequency to lowest FPI peak) [MHz]	χ^2_{red}
1	0.52 ± 0.08	864.90 ± 22.29	0.80
2	0.50 ± 0.07	1661.33 ± 21.72	0.06
3	1.19 ± 0.10	2370.54 ± 21.65	1.30
4	3.02 ± 0.10	2546.20 ± 22.20	1.17
5	1.73 ± 0.15	2735.24 ± 23.82	0.48
6	0.81 ± 0.10	5379.45 ± 23.02	0.05
7	1.12 ± 0.09	5580.09 ± 22.01	1.01
8	1.44 ± 0.15	5767.18 ± 23.32	0.72
9	0.10 ± 0.08	7694.30 ± 22.06	0.05
10	0.42 ± 0.10	8510.53 ± 22.73	0.08

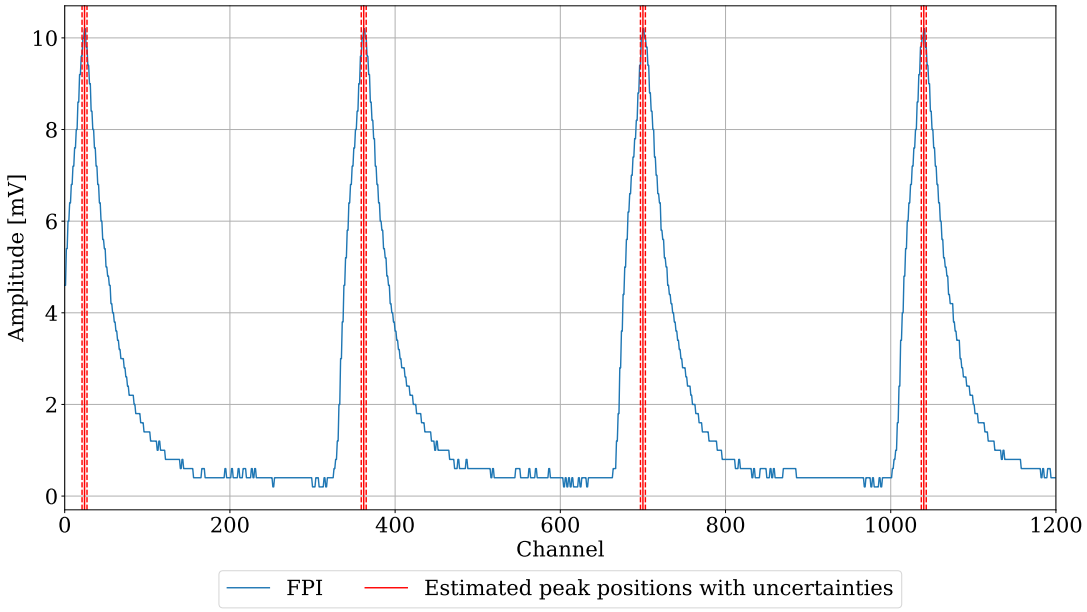


Fig. 41: The FPI measurement of the locked-in frequency. The FPI was put into resonance with the locked-in frequency with a piezo element.

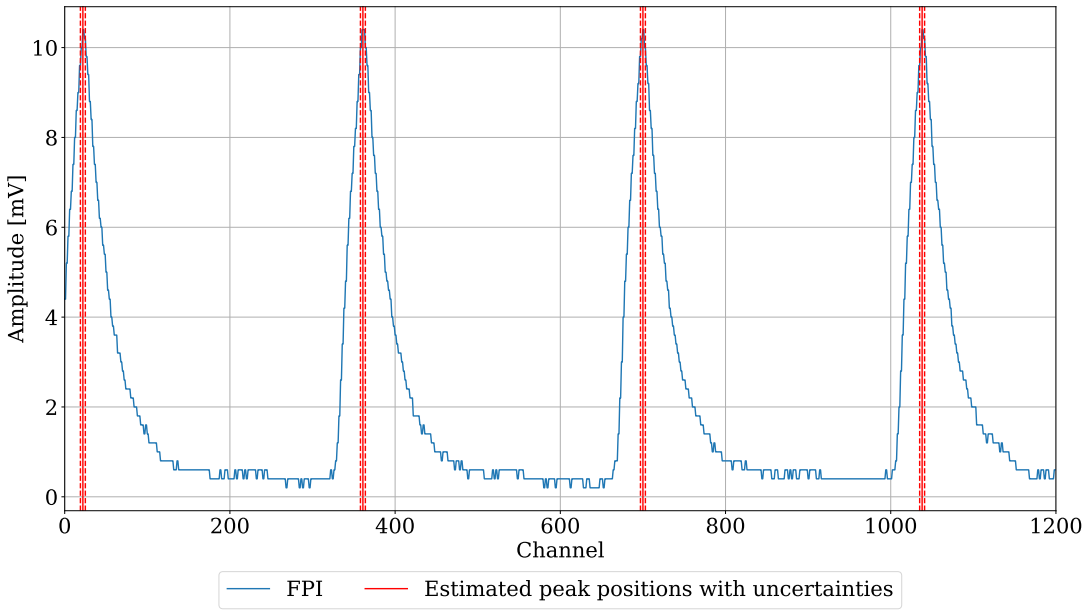


Fig. 42: The FPI measurement of the locked-in frequency. The FPI was put into resonance with the locked-in frequency with a piezo element.

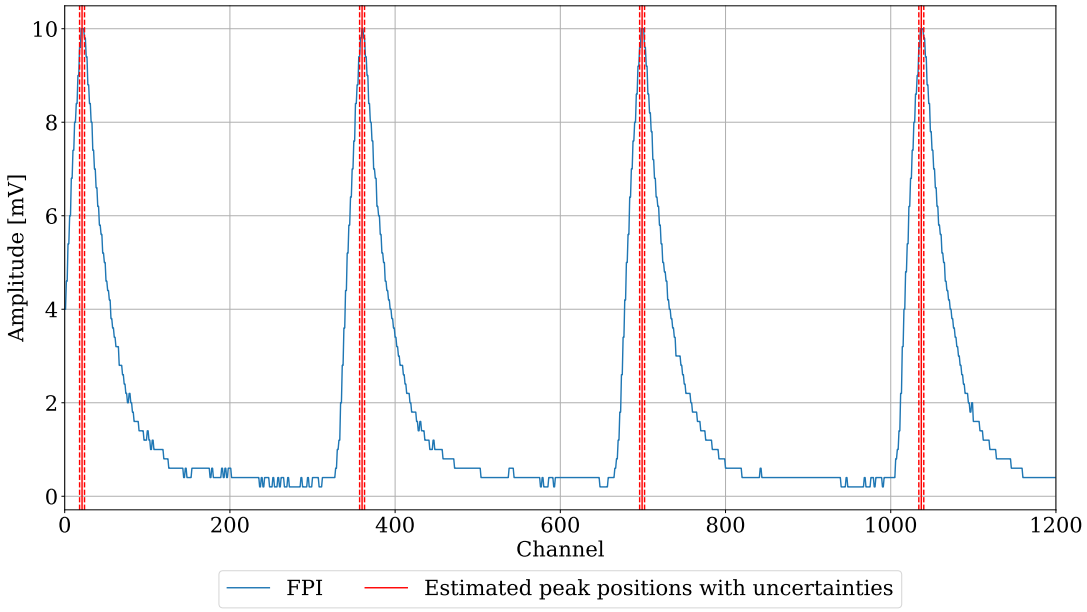


Fig. 43: The FPI measurement of the locked-in frequency. The FPI was put into resonance with the locked-in frequency with a piezo element.

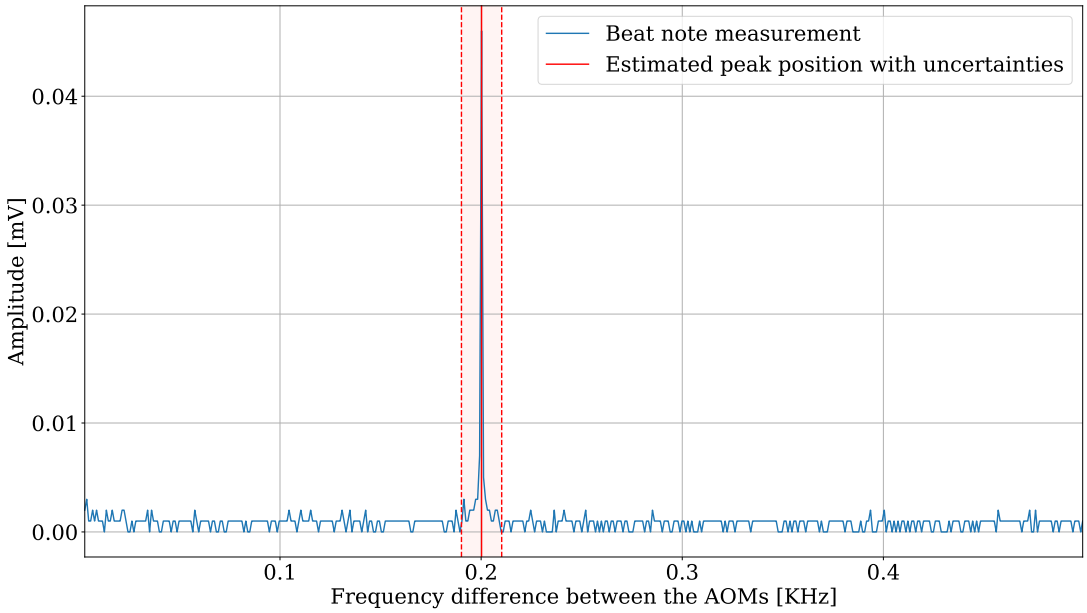


Fig. 44: The beat note measurement for a set frequency difference of the AOMs of 0.2 kHz. Additionally, the found peak and its uncertainty is shown in red.

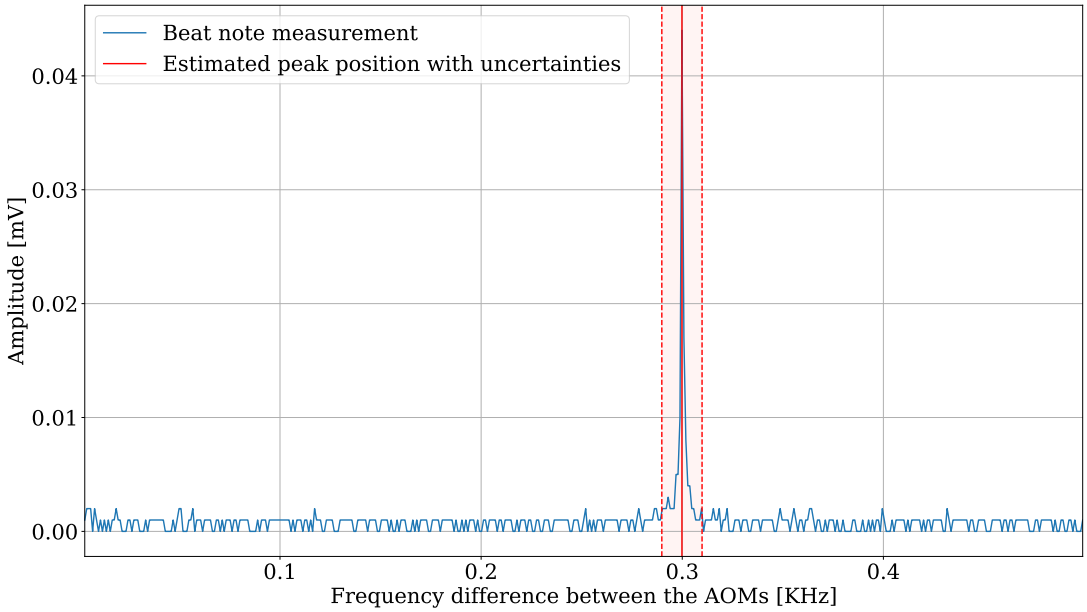


Fig. 45: The beat note measurement for a set frequency difference of the AOMs of 0.3 kHz. Additionally, the found peak and its uncertainty is shown in red.

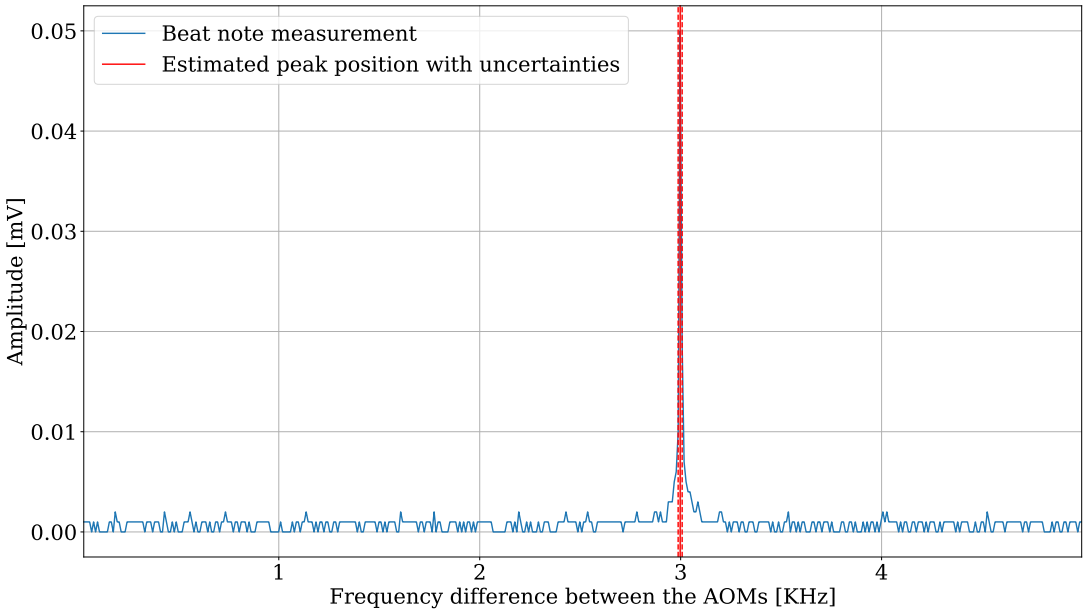


Fig. 46: The beat note measurement for a set frequency difference of the AOMs of 3 kHz. Additionally, the found peak and its uncertainty is shown in red.

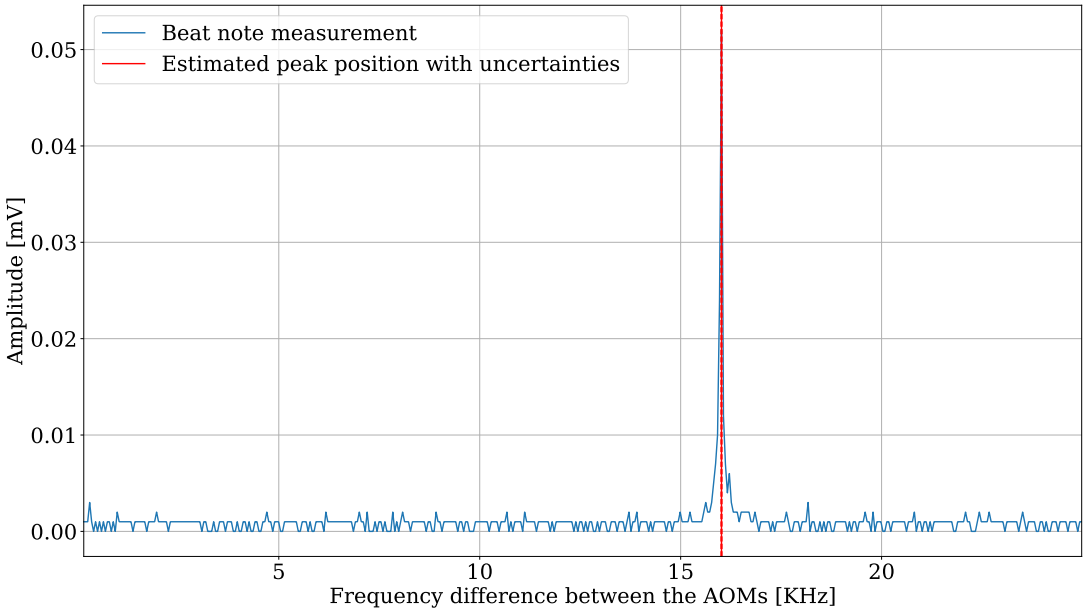


Fig. 47: The beat note measurement for a set frequency difference of the AOMs of 16 kHz. Additionally, the found peak and its uncertainty is shown in red.

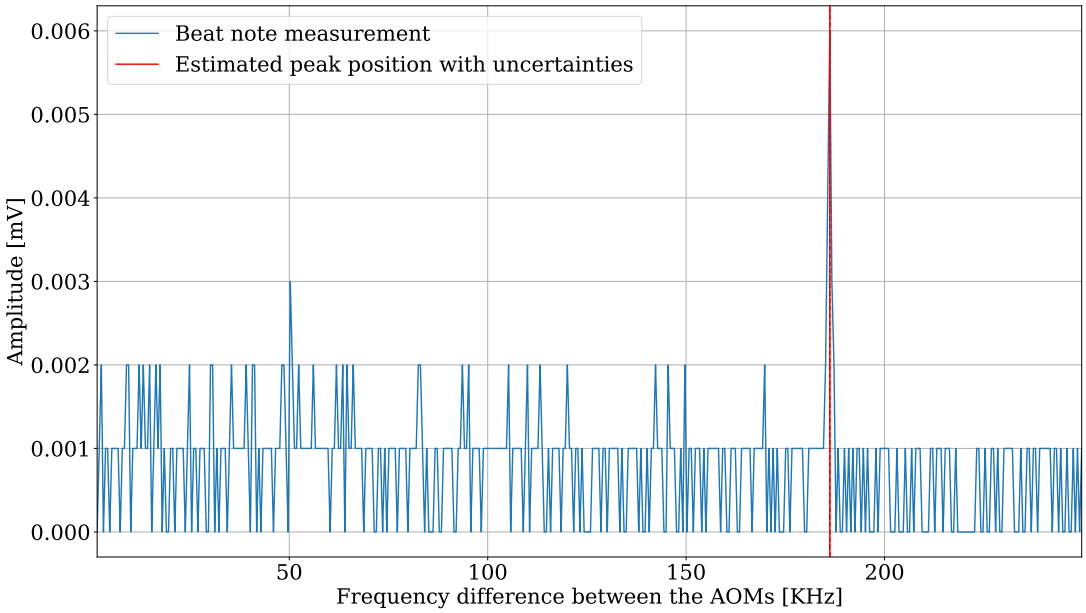


Fig. 48: The beat note measurement for a set frequency difference of the AOMs of 186 kHz. Additionally, the found peak and its uncertainty is shown in red.

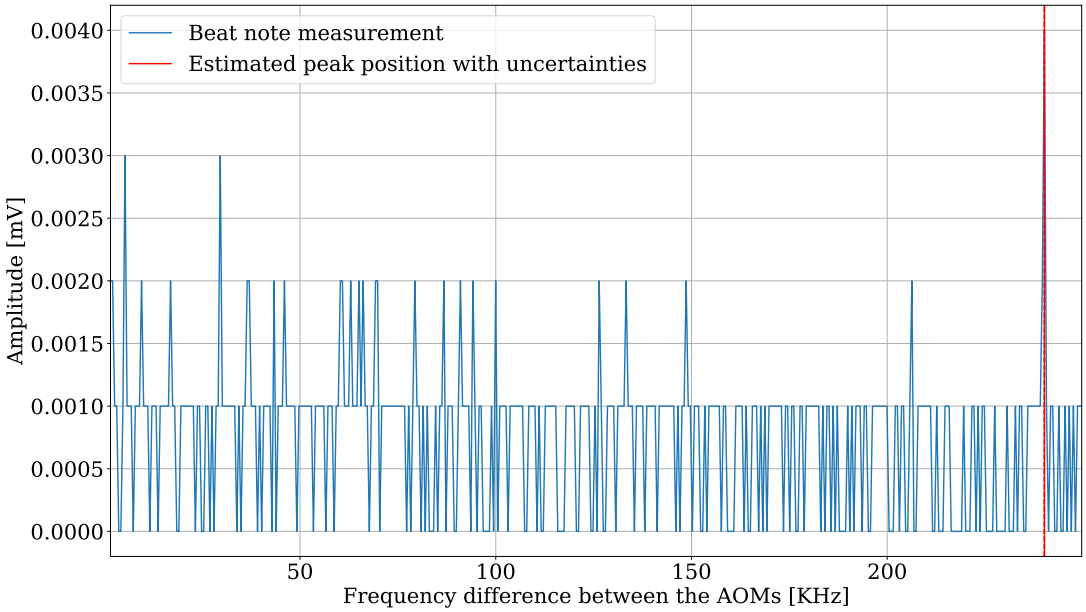


Fig. 49: The beat note measurement for a set frequency difference of the AOMs of 240 kHz. Additionally, the found peak and its uncertainty is shown in red.

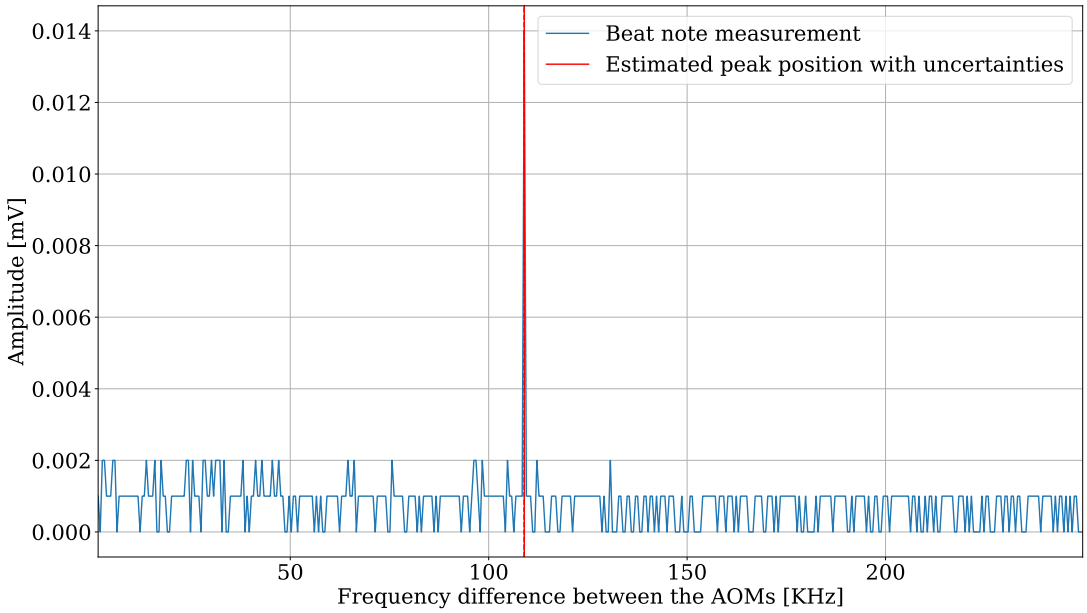


Fig. 50: The beat note measurement for a set frequency difference of the AOMs of 109 kHz. Additionally, the found peak and its uncertainty is shown in red.

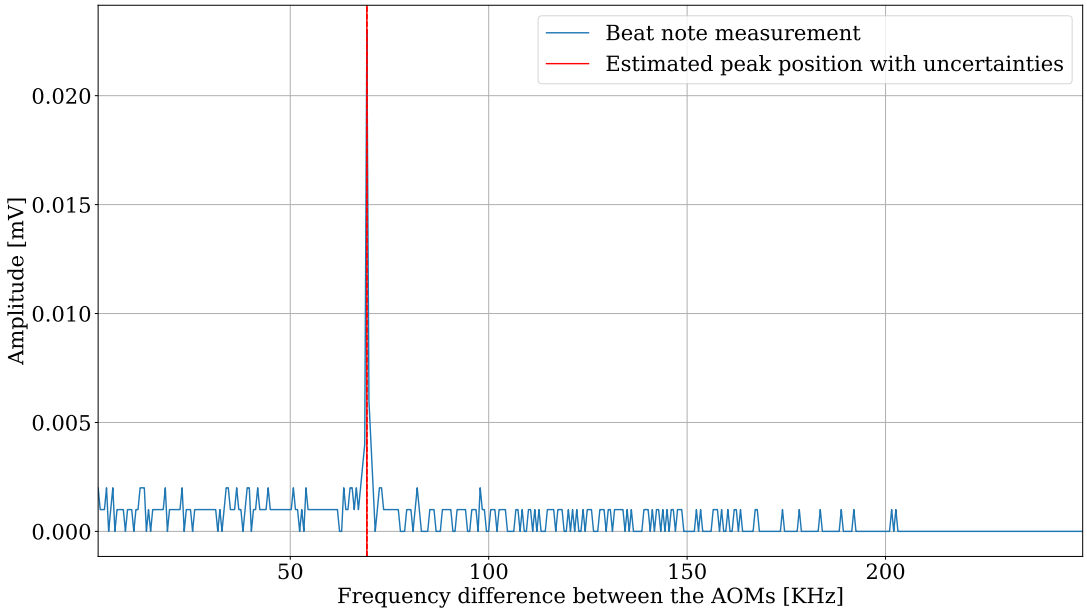


Fig. 51: The beat note measurement for a set frequency difference of the AOMs of 69.42 kHz. Additionally, the found peak and its uncertainty is shown in red.

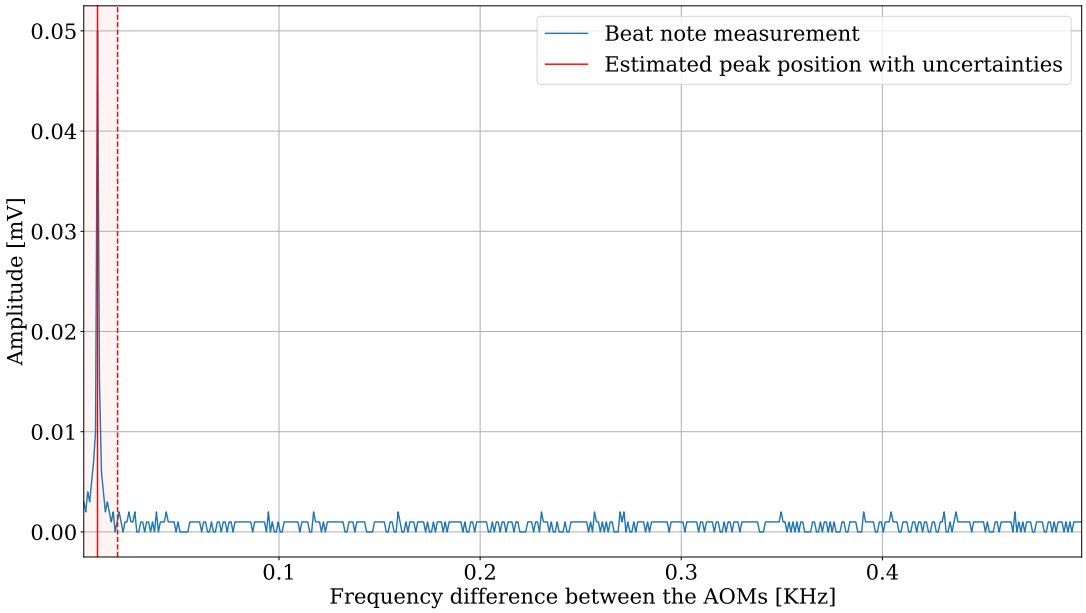


Fig. 52: The beat note measurement for a set frequency difference of the AOMs of 0.01 kHz. Additionally, the found peak and its uncertainty is shown in red.

7 Attachment B

7.1 Python-Code

[ ]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from scipy.optimize import curve_fit
from scipy.signal import find_peaks
import scipy.interpolate as interp

fig_width = 16  # Adjust to match your document
fig_height = 9
base_fontsize = fig_width  # Scale with figure width

import matplotlib as mpl
from mpl_toolkits.axes_grid1.inset_locator import zoomed_inset_axes
from mpl_toolkits.axes_grid1.inset_locator import mark_inset
import matplotlib.font_manager as font_manager
mpl.rcParams['font.family']='serif'
mpl.rcParams['figure.figsize'] = (fig_width, fig_height)
#cmfont = font_manager.FontProperties(fname=mpl.get_data_path() + '/fonts/ttf/,→cmr10.ttf')
#mpl.rcParams['font.serif']=cmfont.get_name()
mpl.rcParams['mathtext.fontset']='cm'
mpl.rcParams['axes.unicode_minus']=False

fontsize = base_fontsize*1.4
plt.rcParams.update({
    "font.size": fontsize,
    "axes.labelsize": fontsize,
    "xtick.labelsize": fontsize,
    "ytick.labelsize": fontsize,
    "legend.fontsize": fontsize
})
plt.rcParams['figure.constrained_layout.use'] = True
```

[ ]:
```
def gaus(x, amplitude, mean, stddev):
    return amplitude * np.exp(-((x - mean) / stddev) ** 2 / 2)
def doppelgaus(x, amp1, mean1, stddev1, amp2, mean2, stddev2):
    return amp1 * np.exp(-((x - mean1) / stddev1) ** 2 / 2) + amp2 * np.exp(-((x - mean2) /
 ↪ stddev2) ** 2 / 2)
def Chi2(meas, fit, sigma):
    return np.sum([((i-j)/k)**2 for i,j,k in zip(fit,meas,sigma)])
def gerade(x, a, b):
    return a*x+b
def lorentz(x, amp, mean, std, offset):
    return amp/(1+((x-mean)/std)**2)+offset
```

[ ]:
```
path = r"C:\Users\jsoro\Desktop\Studium\Master\2. Semester\FP2\EIT\Daten"
savepath = r"C:\Users\jsoro\Desktop\Studium\Master\2. Semester\FP2\EIT\Plots"
```

[ ]:
```
saving = True
```

1 Laser threshold

[ ]:
```
threshold = r"\Laserthreshold.csv"
lt = path + threshold
df_lt = pd.read_csv(lt, sep=";")
```

[ ]:
```
# Create figure with 16:9 aspect ratio
fig, ax = plt.subplots(figsize=(16, 9))

# Main scatter plot
ax.scatter(df_lt["LaserCurrent_mA"], df_lt["PowerMeter_mW"], marker="x", s=100,␣
 ↪label="Laser threshold measurement")
ax.axvline(83.4, color="darkgreen", label="Estimated laser threshold")
ax.axvline(83.3, linestyle="dashed", color="darkgreen")
ax.axvline(83.5, linestyle="dashed", color="darkgreen")
ax.axvspan(83.3, 83.5, alpha=0.4, color="green")
#ax.set_title("Laser threshold")
ax.set_ylabel("Power [mW]")
ax.set_xlabel("Set current [mA]")
ax.grid()
ax.legend(loc=2)

axins = plt.axes([0.65, 0.2, 0.30, 0.30])
axins.scatter(df_lt["LaserCurrent_mA"][35:44], df_lt["PowerMeter_mW"][35:44], marker="x",␣
 ↪s=100)
axins.axvline(83.4, color="darkgreen")
axins.axvline(83.3, linestyle="dashed", alpha=0.3, color="darkgreen")
axins.axvline(83.5, linestyle="dashed", alpha=0.3, color="darkgreen")
axins.axvspan(83.3, 83.5, alpha=0.1, color="green")

x1, x2 = axins.set_xlim(83, 84)  # Adjust X-limits
y1,y2 = axins.set_ylim(-0.05, 1.52)  # Adjust Y-limits
axins.grid()
mark_inset(ax, axins, loc1=2, loc2=3, fc="none", ec="0.5")

if saving == True:
    plt.savefig(savepath + r"\Threshold.pdf", dpi=150, bbox_inches="tight")

plt.show()
```

2 Spectroscopy literature values

[ ]:
```
MHz_lit = [0, 814.5, 1518.6, 1880.2, 4554.4, 4916.0, 6834.7, 7649.2]
```

3 Absorption spectroscopy

[ ]:
```
absorp = r"\23.csv"
absorption = path + absorp
df_absorption = pd.read_csv(absorption,header=None, sep=",", skiprows=2
                            , names=["FPI", "PD", "Trigger"], index_col=False)
df_absorption["running"] = df_absorption["PD"].rolling(10, min_periods=1, center=True).
 ↪sum()/10
df_absorption.iloc[:, :] = df_absorption.iloc[:, :] * 1000 #transform everything to mV
df_absorption["del_running"] = 0.005*df_absorption["running"]
df_absorption
```

3.1 FPI calibration

[ ]:
```
#Generate frequencies
FSR = 750 #MHz
```

[ ]:
```
peaks, _ = find_peaks(df_absorption["FPI"], height=1.2, distance=500)
del_peaks = 500
difference = np.diff(peaks)

fig1 = plt.figure()
#FPI plot
frame1=fig1.add_axes((0,.6,1,.4))
plt.plot(df_absorption.index, df_absorption["FPI"], label="FPI signal")
for i in range(0,len(peaks)):
    if i == 0:  # Add label only for the first iteration
        plt.axvline(peaks[i], color="red", label="Estimated peak positions with␣
 ↪uncertainties")
        plt.axvline(peaks[i]-del_peaks, linestyle="dashed", color="red")
        plt.axvline(peaks[i]+del_peaks, linestyle="dashed", color="red")
        plt.axvspan(peaks[i]-del_peaks, peaks[i]+del_peaks, alpha=0.4, color="mistyrose")
    else:  # No label for subsequent plots
        plt.axvline(peaks[i], color="red")
        plt.axvline(peaks[i]-del_peaks, linestyle="dashed", color="red")
        plt.axvline(peaks[i]+del_peaks, linestyle="dashed", color="red")
        plt.axvspan(peaks[i]-del_peaks, peaks[i]+del_peaks, alpha=0.4, color="mistyrose")
frame1.set_xticklabels([])
plt.xlim(0,df_absorption.index[-1])
plt.ylabel("Amplitude [mV]")
plt.grid()
plt.legend(loc=9, ncol=2, bbox_to_anchor=(0.5, -1.7))

#PD plot
frame2=fig1.add_axes((0,0,1,.6))
plt.plot(df_absorption.index[5:-4], df_absorption["running"][5:-4], color="green",␣
 ↪label="Photodiode signal")
plt.grid()
plt.xlabel("Channel")
plt.ylabel("Amplitude [mV]")
plt.xlim(0,df_absorption.index[-1])
plt.legend(loc=9, ncol=2, bbox_to_anchor=(0.5, -0.25))

if saving == True:
    plt.savefig(savepath + r"\FPI_peaks.pdf", dpi=150, bbox_inches="tight")
plt.show()

peaks, difference
```

[ ]:
```
plt.plot(df_absorption.index, df_absorption["FPI"], label="FPI signal")
for i in range(0,len(peaks)):
    if i == 0:  # Add label only for the first iteration
        plt.axvline(peaks[i], color="red", label="Estimated peak positions with␣
 ↪uncertainties")
        plt.axvline(peaks[i]-del_peaks, linestyle="dashed", color="red")
        plt.axvline(peaks[i]+del_peaks, linestyle="dashed", color="red")
        plt.axvspan(peaks[i]-del_peaks, peaks[i]+del_peaks, alpha=0.4, color="mistyrose")
    else:  # No label for subsequent plots
```

```
        plt.axvline(peaks[i], color="red")
        plt.axvline(peaks[i]-del_peaks, linestyle="dashed", color="red")
        plt.axvline(peaks[i]+del_peaks, linestyle="dashed", color="red")
        plt.axvspan(peaks[i]-del_peaks, peaks[i]+del_peaks, alpha=0.4, color="mistyrose")
plt.xlim(0,df_absorption.index[-1])
plt.xlabel("Channel")
plt.ylabel("Amplitude [mV]")
plt.grid()
plt.legend(loc=9, ncol=2, bbox_to_anchor=(0.5, -.1))
if saving == True:
    plt.savefig(savepath + r"\FPI_a.pdf", dpi=150, bbox_inches="tight")
plt.show()
```

[ ]:
```
# Create the expected evenly spaced grid
expected_peaks = np.linspace(0, len(df_absorption), num=len(peaks))
# Interpolation function
interp_func = interp.interp1d(peaks, expected_peaks, kind='linear',␣
 ↪fill_value="extrapolate")
# Generate corrected values
df_absorption["conversion"] = interp_func(df_absorption.index)
#look if everything is equally spaced now
equal = np.diff(df_absorption["conversion"][peaks])
#convert to frequency
del_peaks_abs = FSR*(del_peaks)/equal[0]
```

[ ]:
```
#Calculate frequencies and move the 0 point to the third peak
moving = 2
df_absorption["frequency"] = FSR*(df_absorption["conversion"])/equal[0]-moving*FSR
df_absorption
```

3.1.1 Defining limits for plots

[ ]:
```
#lower level and #upper level both index and xlim
ll_a_id = (df_absorption["frequency"] > -200).idxmax()
ul_a_id = (df_absorption["frequency"] > df_absorption["frequency"][peaks[-2]]+200).idxmax()
ll_a_plot = df_absorption["frequency"][(df_absorption["frequency"] > -200).idxmax()]
ul_a_plot = df_absorption["frequency"][(df_absorption["frequency"] >␣
 ↪df_absorption["frequency"][peaks[-2]]+200).idxmax()]
```

3.2 Analysis

[ ]:
```
fig1 = plt.figure()
#FPI plot
frame1=fig1.add_axes((0,.6,1,.4))
plt.plot(df_absorption["frequency"][ll_a_id:ul_a_id], df_absorption["FPI"][ll_a_id:
 ↪ul_a_id], label="FPI signal")
for i in range(-moving, len(peaks)-moving):
    if i == 0:
        plt.axvline(FSR*i, color="red", label="Estimated peak positions with␣
 ↪uncertainties")
        plt.axvline(FSR*i-del_peaks_abs, linestyle="dashed", color="red")
        plt.axvline(FSR*i+del_peaks_abs, linestyle="dashed", color="red")
        plt.axvspan(FSR*i-del_peaks_abs, FSR*i+del_peaks_abs, alpha=0.4, color="mistyrose")
    else:
        plt.axvline(FSR*i, color="red")
```

```
        plt.axvline(FSR*i-del_peaks_abs, linestyle="dashed", color="red")
        plt.axvline(FSR*i+del_peaks_abs, linestyle="dashed", color="red")
        plt.axvspan(FSR*i-del_peaks_abs, FSR*i+del_peaks_abs, alpha=0.4, color="mistyrose")
frame1.set_xticklabels([])
plt.xlim(ll_a_plot,ul_a_plot)
plt.ylabel("Amplitude [mV]")
plt.grid()
plt.legend(loc=9, ncol=2, bbox_to_anchor=(0.5, -1.7))

#PD plot
frame2=fig1.add_axes((0,0,1,.6))
plt.plot(df_absorption["frequency"][ll_a_id:ul_a_id], df_absorption["running"][ll_a_id:
 ↪ul_a_id]
         , color="green", label="Photodiode signal")
plt.grid()
plt.xlabel("Frequency difference to lowest FPI peak [MHz]")
plt.ylabel("Amplitude [mV]")
plt.xlim(ll_a_plot,ul_a_plot)
plt.legend(loc=9, ncol=2, bbox_to_anchor=(0.5, -0.25))

if saving == True:
    plt.savefig(savepath + r"\absorption_freq.pdf", dpi=150, bbox_inches="tight")
plt.show()
```

3.2.1 Removing underground

[ ]:
```
low_a1 = -200
high_a1 = 500
low_a2 = 9_500#3900
high_a2 = df_absorption["frequency"][peaks[-2]]+200#5100
low_a3 = 10_000
high_a3 = df_absorption["frequency"][peaks[-2]]+200

low_a = [(df_absorption["frequency"] > low_a1).idxmax(), (df_absorption["frequency"] >␣
 ↪low_a2).idxmax()]
high_a = [(df_absorption["frequency"] > high_a1).idxmax(), (df_absorption["frequency"] >␣
 ↪high_a2).idxmax()]
```

[ ]:
```
x_fit = np.concatenate([
    df_absorption["frequency"][low_a[0]:high_a[0]],
    df_absorption["frequency"][low_a[1]:high_a[1]]
])

y_fit = np.concatenate([
    df_absorption["running"][low_a[0]:high_a[0]],
    df_absorption["running"][low_a[1]:high_a[1]]
])

del_y_fit = np.concatenate([
    df_absorption["del_running"][low_a[0]:high_a[0]],
    df_absorption["del_running"][low_a[1]:high_a[1]]
])

bg_a, del_bg_a = curve_fit(gerade, x_fit, y_fit, sigma=del_y_fit, absolute_sigma=True)
x_values_a = np.linspace(low_a1, high_a3, 1000)
fit_a = gerade(x_values_a, *bg_a)
```

```
df_absorption["clean"] =␣
 ↪df_absorption["running"]-(bg_a[0]*df_absorption["frequency"]+bg_a[1])
df_absorption["del_clean"] = np.sqrt(df_absorption["del_running"]**2
                                     +(np.
 ↪sqrt(del_bg_a[0][0])*df_absorption["frequency"])**2
                                     +(bg_a[0]*del_peaks_abs)**2+del_bg_a[1][1])

fig1 = plt.figure()
#FPI plot
frame1=fig1.add_axes((0,.6,1,.4))
plt.plot(df_absorption["frequency"][ll_a_id:ul_a_id], df_absorption["FPI"][ll_a_id:
 ↪ul_a_id], label="FPI signal")
for i in range(-moving, len(peaks)-moving):
    if i == 0:
        plt.axvline(FSR*i, color="red", label="Estimated peak positions with␣
 ↪uncertainties")
        plt.axvline(FSR*i-del_peaks_abs, linestyle="dashed", color="red")
        plt.axvline(FSR*i+del_peaks_abs, linestyle="dashed", color="red")
        plt.axvspan(FSR*i-del_peaks_abs, FSR*i+del_peaks_abs, alpha=0.4, color="mistyrose")
    else:
        plt.axvline(FSR*i, color="red")
        plt.axvline(FSR*i-del_peaks_abs, linestyle="dashed", color="red")
        plt.axvline(FSR*i+del_peaks_abs, linestyle="dashed", color="red")
        plt.axvspan(FSR*i-del_peaks_abs, FSR*i+del_peaks_abs, alpha=0.4, color="mistyrose")
frame1.set_xticklabels([])
plt.xlim(ll_a_plot,ul_a_plot)
plt.ylabel("Amplitude [mV]")
plt.grid()
plt.legend(loc=9, ncol=2, bbox_to_anchor=(0.5, -1.7))

#PD plot
frame2=fig1.add_axes((0,0,1,.6))
plt.plot(x_values_a, fit_a, color="maroon", label="Linear underground fit")
for i in range(len(low_a)):
    if i == 0:
        plt.plot(df_absorption["frequency"][low_a[i]:high_a[i]],␣
 ↪df_absorption["running"][low_a[i]:high_a[i]]
                 , color="orange", label="Data for linear fit")
    else:
        plt.plot(df_absorption["frequency"][low_a[i]:high_a[i]],␣
 ↪df_absorption["running"][low_a[i]:high_a[i]], color="orange")
plt.plot(df_absorption["frequency"][high_a[0]:low_a[1]],␣
 ↪df_absorption["running"][high_a[0]:low_a[1]], color="green", label="Photodiode signal")
plt.grid()
plt.xlabel("Frequency difference to lowest FPI Peak [MHz]")
plt.ylabel("Amplitude [mV]")
plt.xlim(ll_a_plot,ul_a_plot)
plt.legend(loc=9, ncol=2, bbox_to_anchor=(0.5, -0.25))

if saving == True:
    plt.savefig(savepath + r"\absorption_bg.pdf", dpi=150, bbox_inches="tight")
plt.show()
```

3.3 Finding the peaks

[ ]:
```
fig1 = plt.figure()
#FPI plot
frame1=fig1.add_axes((0,.6,1,.4))
plt.plot(df_absorption["frequency"][ll_a_id:ul_a_id], df_absorption["FPI"][ll_a_id:
 ↪ul_a_id], label="FPI signal")
for i in range(-moving, len(peaks)-moving):
    if i == 0:
        plt.axvline(FSR*i, color="red", label="Estimated peak positions with␣
 ↪uncertainties")
        plt.axvline(FSR*i-del_peaks_abs, linestyle="dashed", color="red")
        plt.axvline(FSR*i+del_peaks_abs, linestyle="dashed", color="red")
        plt.axvspan(FSR*i-del_peaks_abs, FSR*i+del_peaks_abs, alpha=0.4, color="mistyrose")
    else:
        plt.axvline(FSR*i, color="red")
        plt.axvline(FSR*i-del_peaks_abs, linestyle="dashed", color="red")
        plt.axvline(FSR*i+del_peaks_abs, linestyle="dashed", color="red")
        plt.axvspan(FSR*i-del_peaks_abs, FSR*i+del_peaks_abs, alpha=0.4, color="mistyrose")
frame1.set_xticklabels([])
plt.xlim(ll_a_plot,ul_a_plot)
plt.ylabel("Amplitude [mV]")
plt.grid()
plt.legend(loc=9, ncol=2, bbox_to_anchor=(0.5, -1.7))

#PD plot
frame2=fig1.add_axes((0,0,1,.6))
plt.plot(df_absorption["frequency"][ll_a_id:ul_a_id], df_absorption["clean"][ll_a_id:
 ↪ul_a_id], color="green", label="Data after underground subtraction")


plt.grid()
plt.xlabel("Frequency difference to lowest FPI Peak [MHz]")
plt.ylabel("Amplitude [mV]")
plt.xlim(ll_a_plot,ul_a_plot)
plt.legend(loc=9, ncol=2, bbox_to_anchor=(0.5, -0.25))

if saving == True:
    plt.savefig(savepath + r"\absorption_clean.pdf", dpi=150, bbox_inches="tight")
plt.show()
```

[ ]:
```
peaks_a_low = [800, 1800, 5800, 7800]
peaks_a_low = [(df_absorption["frequency"] > i).idxmax() for i in peaks_a_low]
peaks_a_high = [1400, 3500, 6500, 9300]
peaks_a_high = [(df_absorption["frequency"] > i).idxmax() for i in peaks_a_high]
amp_a = [-7, -7, -20, -16.5, -4, -6]  # Amplitudes
mean_a = [1100, 2000, 2800, 6050, 8000, 8800]  # Means
std_a = [200, 200, 200, 500, 200, 200]  # Standard deviations
```

[ ]:
```
#Peak 1
fit_g1_a, fitcov_g1_a = curve_fit(gaus, df_absorption["frequency"][peaks_a_low[0]:
 ↪peaks_a_high[0]], df_absorption["clean"][peaks_a_low[0]:peaks_a_high[0]], p0=[amp_a[0],␣
 ↪mean_a[0], std_a[0]], sigma=df_absorption["del_clean"][peaks_a_low[0]:peaks_a_high[0]],␣
 ↪absolute_sigma=True)
x_fit_g1_a = df_absorption["frequency"][peaks_a_low[0]:peaks_a_high[0]]
y_fit_g1_a = gaus(df_absorption["frequency"][peaks_a_low[0]:peaks_a_high[0]], *fit_g1_a)
```

```
x_plot_g1_a = np.linspace(ll_a_plot,ul_a_plot, 2000)
y_plot_g1_a = gaus(x_plot_g1_a, *fit_g1_a)

amp_g1_a = fit_g1_a[0]
del_amp_g1_a = np.sqrt(fitcov_g1_a[0][0])
mean_g1_a = fit_g1_a[1]
del_mean_g1_a = np.sqrt((fit_g1_a[2]/np.
 ↪sqrt(peaks_a_high[0]-peaks_a_low[0]))**2+del_peaks_abs**2)


#Doublepeak 1
fit_dg1_a, fitcov_dg1_a = curve_fit(doppelgaus, df_absorption["frequency"][peaks_a_low[1]:
 ↪peaks_a_high[1]], df_absorption["clean"][peaks_a_low[1]:peaks_a_high[1]], p0=[amp_a[1],␣
 ↪mean_a[1], std_a[1], amp_a[2], mean_a[2], std_a[2]],␣
 ↪sigma=df_absorption["del_clean"][peaks_a_low[1]:peaks_a_high[1]], absolute_sigma=True)
x_fit_dg1_a = df_absorption["frequency"][peaks_a_low[1]:peaks_a_high[1]]
y_fit_dg1_a = doppelgaus(df_absorption["frequency"][peaks_a_low[1]:peaks_a_high[1]],␣
 ↪*fit_dg1_a)
x_plot_dg1_a = np.linspace(ll_a_plot,ul_a_plot, 2000)
y_plot_dg1_a = doppelgaus(x_plot_dg1_a, *fit_dg1_a)

amp1_dg1_a = fit_dg1_a[0]
del_amp1_dg1_a = np.sqrt(fitcov_dg1_a[0][0])
mean1_dg1_a = fit_dg1_a[1]
del_mean1_dg1_a = np.sqrt((fit_dg1_a[2]/np.
 ↪sqrt(peaks_a_high[1]-peaks_a_low[1]))**2+del_peaks_abs**2)
amp2_dg1_a = fit_dg1_a[3]
del_amp2_dg1_a = np.sqrt(fitcov_dg1_a[3][3])
mean2_dg1_a = fit_dg1_a[4]
del_mean2_dg1_a = np.sqrt((fit_dg1_a[5]/np.
 ↪sqrt(peaks_a_high[1]-peaks_a_low[1]))**2+del_peaks_abs**2)


#Peak 2
fit_g2_a, fitcov_g2_a = curve_fit(gaus, df_absorption["frequency"][peaks_a_low[2]:
 ↪peaks_a_high[2]], df_absorption["clean"][peaks_a_low[2]:peaks_a_high[2]], p0=[amp_a[3],␣
 ↪mean_a[3], std_a[3]], sigma=df_absorption["del_clean"][peaks_a_low[2]:peaks_a_high[2]],␣
 ↪absolute_sigma=True)
x_fit_g2_a = df_absorption["frequency"][peaks_a_low[2]:peaks_a_high[2]]
y_fit_g2_a = gaus(df_absorption["frequency"][peaks_a_low[2]:peaks_a_high[2]], *fit_g2_a)
x_plot_g2_a = np.linspace(ll_a_plot,ul_a_plot, 2000)
y_plot_g2_a = gaus(x_plot_g2_a, *fit_g2_a)

amp_g2_a = fit_g2_a[0]
del_amp_g2_a = np.sqrt(fitcov_g2_a[0][0])
mean_g2_a = fit_g2_a[1]
del_mean_g2_a = np.sqrt((fit_g2_a[2]/np.
 ↪sqrt(peaks_a_high[0]-peaks_a_low[0]))**2+del_peaks_abs**2)


#Doublepeak 2
fit_dg2_a, fitcov_dg2_a = curve_fit(doppelgaus, df_absorption["frequency"][peaks_a_low[3]:
 ↪peaks_a_high[3]], df_absorption["clean"][peaks_a_low[3]:peaks_a_high[3]], p0=[amp_a[4],␣
 ↪mean_a[4], std_a[4], amp_a[5], mean_a[5], std_a[5]],␣
 ↪sigma=df_absorption["del_clean"][peaks_a_low[3]:peaks_a_high[3]], absolute_sigma=True)
```

```
x_fit_dg2_a = df_absorption["frequency"][peaks_a_low[3]:peaks_a_high[3]]
y_fit_dg2_a = doppelgaus(df_absorption["frequency"][peaks_a_low[3]:peaks_a_high[3]],␣
 ↪*fit_dg2_a)
x_plot_dg2_a = np.linspace(ll_a_plot,ul_a_plot, 2000)
y_plot_dg2_a = doppelgaus(x_plot_dg2_a, *fit_dg2_a)

amp1_dg2_a = fit_dg2_a[0]
del_amp1_dg2_a = np.sqrt(fitcov_dg2_a[0][0])
mean1_dg2_a = fit_dg2_a[1]
del_mean1_dg2_a = np.sqrt((fit_dg2_a[2]/np.
 ↪sqrt(peaks_a_high[3]-peaks_a_low[3]))**2+del_peaks_abs**2)
amp2_dg2_a = fit_dg2_a[3]
del_amp2_dg2_a = np.sqrt(fitcov_dg2_a[3][3])
mean2_dg2_a = fit_dg2_a[4]
del_mean2_dg2_a = np.sqrt((fit_dg2_a[5]/np.
 ↪sqrt(peaks_a_high[3]-peaks_a_low[3]))**2+del_peaks_abs**2)
```

[ ]:
```
#calculating reduced chi2 values
chi2red1_a = Chi2(df_absorption["clean"][peaks_a_low[0]:peaks_a_high[0]], y_fit_g1_a,␣
 ↪df_absorption["del_clean"][peaks_a_low[0]:peaks_a_high[0]])/
 ↪(peaks_a_high[0]-peaks_a_low[0]-3)
chi2red2_a = Chi2(df_absorption["clean"][peaks_a_low[1]:peaks_a_high[1]], y_fit_dg1_a,␣
 ↪df_absorption["del_clean"][peaks_a_low[1]:peaks_a_high[1]])/
 ↪(peaks_a_high[1]-peaks_a_low[1]-6)
chi2red3_a = Chi2(df_absorption["clean"][peaks_a_low[2]:peaks_a_high[2]], y_fit_g2_a,␣
 ↪df_absorption["del_clean"][peaks_a_low[2]:peaks_a_high[2]])/
 ↪(peaks_a_high[2]-peaks_a_low[2]-3)
chi2red4_a = Chi2(df_absorption["clean"][peaks_a_low[3]:peaks_a_high[3]], y_fit_dg2_a,␣
 ↪df_absorption["del_clean"][peaks_a_low[3]:peaks_a_high[3]])/
 ↪(peaks_a_high[3]-peaks_a_low[3]-6)
```

[ ]:
```
print("Amplitude for the 1.Peak:", amp_g1_a, "+-", del_amp_g1_a)
print("Mean for the 1.Peak:", mean_g1_a, "+-", del_mean_g1_a)
print("Reduced chi^2-value:", chi2red1_a, "\n")

print("Amplitude for the 2.Peak:", amp1_dg1_a, "+-", del_amp1_dg1_a)
print("Mean for the 2.Peak:", mean1_dg1_a, "+-", del_mean1_dg1_a)
print("Amplitude for the 3.Peak:", amp2_dg1_a, "+-", del_amp2_dg1_a)
print("Mean for the 3.Peak:", mean2_dg1_a, "+-", del_mean2_dg1_a)
print("Reduced chi^2-value:", chi2red2_a, "\n")

print("Amplitude for the 4.Peak:", amp_g2_a, "+-", del_amp_g2_a)
print("Mean for the 4.Peak:", mean_g2_a, "+-", del_mean_g2_a)
print("Reduced chi^2-value:", chi2red3_a, "\n")

print("Amplitude for the 5.Peak:", amp1_dg2_a, "+-", del_amp1_dg2_a)
print("Mean for the 5.Peak:", mean1_dg2_a, "+-", del_mean1_dg2_a)
print("Amplitude for the 6.Peak:", amp2_dg2_a, "+-", del_amp2_dg2_a)
print("Mean for the 6.Peak:", mean2_dg2_a, "+-", del_mean2_dg2_a)
print("Reduced chi^2-value:", chi2red4_a, "\n")
```

[ ]:
```
a_1 = 0
del_a_1 = np.sqrt(2)*del_mean_g1_a
a_2 = mean1_dg1_a - mean_g1_a
del_a_2 = np.sqrt(del_mean1_dg1_a**2 + del_mean_g1_a**2)
a_3 = mean2_dg1_a - mean_g1_a
```

```
del_a_3 = np.sqrt(del_mean2_dg1_a**2 + del_mean_g1_a**2)
a_4 = mean_g2_a - mean_g1_a
del_a_4 = np.sqrt(del_mean_g2_a**2 + del_mean_g1_a**2)
a_5 = mean1_dg2_a - mean_g1_a
del_a_5 = np.sqrt(del_mean1_dg2_a**2 + del_mean_g1_a**2)
a_6 = mean2_dg2_a - mean_g1_a
del_a_6 = np.sqrt(del_mean2_dg1_a**2 + del_mean_g1_a**2)

rel_means_a = [a_1,a_2,a_3,a_4,a_5,a_6]
del_rel_means_a = [del_a_1,del_a_2,del_a_3,del_a_4,del_a_5,del_a_6]

for i in range(len(rel_means_a)):
    print(f"{i+1}.Peak position relative to first peak:{rel_means_a[i]} +-␣
 ↪{del_rel_means_a[i]}")
```

[ ]:
```
plt.plot(df_absorption["frequency"][ll_a_id:ul_a_id], df_absorption["clean"][ll_a_id:
 ↪ul_a_id], color="green", label="Data after underground subtraction")
plt.plot(x_plot_g1_a, y_plot_g1_a, label=f"{mean_g1_a:.2f} $\pm$ {del_mean_g1_a:.2f} MHz")
plt.plot(x_plot_dg1_a, y_plot_dg1_a, label=f"{mean1_dg1_a:.2f} $\pm$ {del_mean1_dg1_a:.2f}␣
 ↪MHz\n{mean2_dg1_a:.2f} $\pm$ {del_mean2_dg1_a:.2f} MHz")
plt.plot(x_plot_g2_a, y_plot_g2_a, label=f"{mean_g2_a:.2f} $\pm$ {del_mean_g2_a:.2f} MHz")
plt.plot(x_plot_dg2_a, y_plot_dg2_a, label=f"{mean1_dg2_a:.2f} $\pm$ {del_mean1_dg2_a:.2f}␣
 ↪MHz\n{mean2_dg2_a:.2f} $\pm$ {del_mean2_dg2_a:.2f} MHz")

for i in range(0,len(peaks_a_low)):
    plt.axvline(df_absorption["frequency"][peaks_a_low[i]], ls="-", color='black')
    plt.axvline(df_absorption["frequency"][peaks_a_high[i]], ls="-", color='black')
plt.xlim(ll_a_plot,ul_a_plot)
plt.grid()
plt.legend()
if saving == True:
    plt.savefig(savepath + r"\absorption_fits.pdf", dpi=150, bbox_inches="tight")
```

[ ]:
```
fig1 = plt.figure()
#FPI plot
frame1=fig1.add_axes((0,.6,1,.4))
plt.plot(df_absorption["frequency"][ll_a_id:ul_a_id], df_absorption["FPI"][ll_a_id:
 ↪ul_a_id], label="FPI signal")
for i in range(-moving, len(peaks)-moving):
    if i == 0:
        plt.axvline(FSR*i, color="red", label="Estimated peak positions with␣
 ↪uncertainties")
        plt.axvline(FSR*i-del_peaks_abs, linestyle="dashed", color="red")
        plt.axvline(FSR*i+del_peaks_abs, linestyle="dashed", color="red")
        plt.axvspan(FSR*i-del_peaks_abs, FSR*i+del_peaks_abs, alpha=0.4, color="mistyrose")
    else:
        plt.axvline(FSR*i, color="red")
        plt.axvline(FSR*i-del_peaks_abs, linestyle="dashed", color="red")
        plt.axvline(FSR*i+del_peaks_abs, linestyle="dashed", color="red")
        plt.axvspan(FSR*i-del_peaks_abs, FSR*i+del_peaks_abs, alpha=0.4, color="mistyrose")
frame1.set_xticklabels([])
plt.xlim(ll_a_plot,ul_a_plot)
plt.ylabel("Amplitude [mV]")
plt.grid()
plt.legend(loc=9, ncol=2, bbox_to_anchor=(0.5, -1.7))
```

```
#PD plot
frame2=fig1.add_axes((0,0,1,.6))
plt.plot(df_absorption["frequency"][ll_a_id:ul_a_id], df_absorption["clean"][ll_a_id:
 ↪ul_a_id], color="green", label="Data after underground subtraction")
plt.plot(x_plot_g1_a, y_plot_g1_a, color="orange", label=f"Single Gaussian absorption peak␣
 ↪fit")
plt.plot(x_plot_dg1_a, y_plot_dg1_a, color="blue", label=f"Double Gaussian absorption peak␣
 ↪fit")
plt.plot(x_plot_g2_a, y_plot_g2_a, color="orange")
plt.plot(x_plot_dg2_a, y_plot_dg2_a, color="blue")

for i in range(0,len(peaks_a_low)):
    plt.axvline(df_absorption["frequency"][peaks_a_low[i]], ls="-", color='black')
    plt.axvline(df_absorption["frequency"][peaks_a_high[i]], ls="-", color='black')


plt.grid()
plt.xlabel("Frequency difference to lowest FPI Peak [MHz]")
plt.ylabel("Amplitude [mV]")
plt.xlim(ll_a_plot,ul_a_plot)
plt.legend(loc=9, ncol=2, bbox_to_anchor=(0.5, -0.25))

if saving == True:
    plt.savefig(savepath + r"\absorption.pdf", dpi=150, bbox_inches="tight")
plt.show()
```

3.3.1 Abundance and hyperfine

[ ]:
```
#abundance
abundance_a = (amp_g1_a+amp1_dg1_a)/(amp_g1_a+amp1_dg1_a+2*amp2_dg1_a)
del_abundance_a = np.sqrt(((2*amp2_dg1_a)/
 ↪((amp_g1_a+amp1_dg1_a+2*amp2_dg1_a)**2))**2*del_amp_g1_a**2+(2*amp2_dg1_a/
 ↪((amp_g1_a+amp1_dg1_a+2*amp2_dg1_a)**2))**2*del_amp1_dg1_a**2+(2*(amp_g1_a+amp1_dg1_a)/
 ↪((amp_g1_a+amp1_dg1_a+2*amp2_dg1_a)**2))**2*2*del_amp2_dg1_a**2)
abundance_a2 = (amp1_dg2_a+amp2_dg2_a)/(amp1_dg2_a+amp2_dg2_a+2*amp_g2_a)
del_abundance_a2 = np.sqrt(((2*amp_g2_a)/
 ↪((amp1_dg2_a+amp2_dg2_a+2*amp_g2_a)**2))**2*del_amp1_dg2_a**2+(2*amp_g2_a/
 ↪((amp1_dg2_a+amp2_dg2_a+2*amp_g2_a)**2))**2*del_amp2_dg2_a**2+(2*(amp1_dg2_a+amp2_dg2_a)/
 ↪((amp1_dg2_a+amp2_dg2_a+2*amp_g2_a)**2))**2*2*del_amp_g2_a**2)

#hyperfine
hyper_a1_87 = a_2/2
del_hyper_a1_87 = np.sqrt((del_a_2/2)**2+(del_a_1/2)**2)
hyper_a2_87 = (a_6-a_5)/2
del_hyper_a2_87 = np.sqrt((del_a_6/2)**2+(del_a_5/2)**2)
#hyperfine 85 not possible (low resolution)

abundance_a, del_abundance_a, abundance_a2, del_abundance_a2, hyper_a1_87,␣
 ↪del_hyper_a1_87, hyper_a2_87, del_hyper_a2_87
```

4 Fluorescence spectroscopy

[ ]:
```
fluor = r"\21.csv"
fluorescence = path + fluor
df_fluorescence = pd.read_csv(fluorescence,header=None, sep=",", skiprows=2, names=["FPI",␣
 ↪"PD", "Trigger"], index_col=False)
df_fluorescence["running"] = df_fluorescence["PD"].rolling(10, min_periods=1, center=True).
 ↪sum()/10
df_fluorescence.iloc[:, :] = df_fluorescence.iloc[:, :] * 1000 #transform everything to mV
df_fluorescence["del_running"] = 0.005*df_fluorescence["running"]
df_fluorescence
```

4.1 FPI calibration

[ ]:
```
peaks_f, _ = find_peaks(df_fluorescence["FPI"], height=400, distance=5000)
del_peaks_f = 500
difference_f = np.diff(peaks_f)

fig1 = plt.figure()
#FPI plot
frame1=fig1.add_axes((0,.6,1,.4))
plt.plot(df_fluorescence.index, df_fluorescence["FPI"], label="FPI signal")
for i in range(0,len(peaks_f)):
    if i == 0:  # Add label only for the first iteration
        plt.axvline(peaks_f[i], color="red", label="Estimated peak positions with␣
 ↪uncertainties")
        plt.axvline(peaks_f[i]-del_peaks_f, linestyle="dashed", color="red")
        plt.axvline(peaks_f[i]+del_peaks_f, linestyle="dashed", color="red")
        plt.axvspan(peaks_f[i]-del_peaks_f, peaks_f[i]+del_peaks_f, alpha=0.4,␣
 ↪color="mistyrose")
    else:  # No label for subsequent plots
        plt.axvline(peaks_f[i], color="red")
        plt.axvline(peaks_f[i]-del_peaks_f, linestyle="dashed", color="red")
        plt.axvline(peaks_f[i]+del_peaks_f, linestyle="dashed", color="red")
        plt.axvspan(peaks_f[i]-del_peaks_f, peaks_f[i]+del_peaks_f, alpha=0.4,␣
 ↪color="mistyrose")
frame1.set_xticklabels([])
plt.xlim(0,df_fluorescence.index[-1])
plt.ylabel("Amplitude [mV]")
plt.grid()
plt.legend(loc=9, ncol=2, bbox_to_anchor=(0.5, -1.7))

#PD plot
frame2=fig1.add_axes((0,0,1,.6))
plt.plot(df_fluorescence.index[5:-4], df_fluorescence["running"][5:-4], color="green",␣
 ↪label="Photodiode signal")
plt.grid()
plt.xlabel("Channel")
plt.ylabel("Amplitude [mV]")
plt.xlim(0,df_fluorescence.index[-1])
plt.legend(loc=9, ncol=2, bbox_to_anchor=(0.5, -0.25))

if saving == True:
    plt.savefig(savepath + r"\FPI_peaks_f.pdf", dpi=150, bbox_inches="tight")
plt.show()
```

```
peaks_f, difference_f
```

[ ]:
```
plt.plot(df_fluorescence.index, df_fluorescence["FPI"], label="FPI signal")
for i in range(0,len(peaks_f)):
    if i == 0:  # Add label only for the first iteration
        plt.axvline(peaks_f[i], color="red", label="Estimated peak positions with␣
 ↪uncertainties")
        plt.axvline(peaks_f[i]-del_peaks_f, linestyle="dashed", color="red")
        plt.axvline(peaks_f[i]+del_peaks_f, linestyle="dashed", color="red")
        plt.axvspan(peaks_f[i]-del_peaks_f, peaks_f[i]+del_peaks_f, alpha=0.4,␣
 ↪color="mistyrose")
    else:  # No label for subsequent plots
        plt.axvline(peaks_f[i], color="red")
        plt.axvline(peaks_f[i]-del_peaks_f, linestyle="dashed", color="red")
        plt.axvline(peaks_f[i]+del_peaks_f, linestyle="dashed", color="red")
        plt.axvspan(peaks_f[i]-del_peaks_f, peaks_f[i]+del_peaks_f, alpha=0.4,␣
 ↪color="mistyrose")
plt.xlim(0,df_fluorescence.index[-1])
plt.xlabel("Channel")
plt.ylabel("Amplitude [mV]")
plt.grid()
plt.legend(loc=9, ncol=2, bbox_to_anchor=(0.5, -.1))
if saving == True:
    plt.savefig(savepath + r"\FPI_f.pdf", dpi=150, bbox_inches="tight")
plt.show()
```

[ ]:
```
# Create the expected evenly spaced grid
expected_peaks_f = np.linspace(0, len(df_fluorescence), num=len(peaks_f))
# Interpolation function
interp_func_f = interp.interp1d(peaks_f, expected_peaks_f, kind='linear',␣
 ↪fill_value="extrapolate")
# Generate corrected values
df_fluorescence["conversion"] = interp_func_f(df_fluorescence.index)
#look if everything is equally spaced now
equal_f = np.diff(df_fluorescence["conversion"][peaks_f])
#convert to frequency
del_peaks_abs_f = FSR*(del_peaks_f)/equal_f[0]
```

[ ]:
```
#Calculate frequencies and move the 0 point
moving_f = 13
df_fluorescence["frequency"] = FSR*(df_fluorescence["conversion"])/equal_f[0]-moving_f*FSR
df_fluorescence
```

4.1.1 Defining limits for plots

[ ]:
```
#lower level and #upper level both index and xlim
ll_f_id = (df_fluorescence["frequency"] > -200).idxmax()
ul_f_id = (df_fluorescence["frequency"] > df_fluorescence["frequency"][peaks_f[-3]]+200).
 ↪idxmax()
ll_f_plot = df_fluorescence["frequency"][(df_fluorescence["frequency"] > -200).idxmax()]
ul_f_plot = df_fluorescence["frequency"][(df_fluorescence["frequency"] >␣
 ↪df_fluorescence["frequency"][peaks_f[-3]]+200).idxmax()]
```

4.2 Analysis

[ ]:
```
fig1 = plt.figure()
#FPI plot
frame1=fig1.add_axes((0,.6,1,.4))
plt.plot(df_fluorescence["frequency"][ll_f_id:ul_f_id], df_fluorescence["FPI"][ll_f_id:
 ↪ul_f_id], label="FPI signal")
for i in range(-moving_f, len(peaks_f)-moving_f):
    if i == 0:
        plt.axvline(FSR*i, color="red", label="Estimated peak positions with␣
 ↪uncertainties")
        plt.axvline(FSR*i-del_peaks_abs_f, linestyle="dashed", color="red")
        plt.axvline(FSR*i+del_peaks_abs_f, linestyle="dashed", color="red")
        plt.axvspan(FSR*i-del_peaks_abs_f, FSR*i+del_peaks_abs_f, alpha=0.4,␣
 ↪color="mistyrose")
    else:
        plt.axvline(FSR*i, color="red")
        plt.axvline(FSR*i-del_peaks_abs_f, linestyle="dashed", color="red")
        plt.axvline(FSR*i+del_peaks_abs_f, linestyle="dashed", color="red")
        plt.axvspan(FSR*i-del_peaks_abs_f, FSR*i+del_peaks_abs_f, alpha=0.4,␣
 ↪color="mistyrose")
frame1.set_xticklabels([])
plt.xlim(ll_f_plot,ul_f_plot)
plt.ylabel("Amplitude [mV]")
plt.grid()
plt.legend(loc=9, ncol=2, bbox_to_anchor=(0.5, -1.7))

#PD plot
frame2=fig1.add_axes((0,0,1,.6))
plt.plot(df_fluorescence["frequency"][ll_f_id:ul_f_id], df_fluorescence["running"][ll_f_id:
 ↪ul_f_id], color="green", label="Photodiode signal")
plt.grid()
plt.xlabel("Frequency difference to lowest FPI peak [MHz]")
plt.ylabel("Amplitude [mV]")
plt.xlim(ll_f_plot,ul_f_plot)
plt.legend(loc=9, ncol=2, bbox_to_anchor=(0.5, -0.25))

if saving == True:
    plt.savefig(savepath + r"\fluorescence_freq.pdf", dpi=150, bbox_inches="tight")
plt.show()
```

4.3 Removing background

[ ]:
```
low_f1 = -200
high_f1 = 500
low_f2 = 4000
high_f2 = 5000
low_f3 = 6700
high_f3 = df_fluorescence["frequency"][peaks[-2]]-200

low_f = [(df_fluorescence["frequency"] > low_f1).idxmax(), (df_fluorescence["frequency"] >␣
 ↪low_f2).idxmax(), (df_fluorescence["frequency"] > low_f3).idxmax()]
high_f = [(df_fluorescence["frequency"] > high_f1).idxmax(), (df_fluorescence["frequency"]␣
 ↪> high_f2).idxmax(), (df_fluorescence["frequency"] > high_f3).idxmax()]
```

[ ]:
```
x_fit_f = np.concatenate([
    df_fluorescence["frequency"][low_f[0]:high_f[0]],
    df_fluorescence["frequency"][low_f[1]:high_f[1]],
    df_fluorescence["frequency"][low_f[2]:high_f[2]]
])

y_fit_f = np.concatenate([
    df_fluorescence["running"][low_f[0]:high_f[0]],
    df_fluorescence["running"][low_f[1]:high_f[1]],
    df_fluorescence["running"][low_f[2]:high_f[2]]
])

del_y_fit_f = np.concatenate([
    df_fluorescence["del_running"][low_f[0]:high_f[0]],
    df_fluorescence["del_running"][low_f[1]:high_f[1]],
    df_fluorescence["del_running"][low_f[2]:high_f[2]]
])

bg_f, del_bg_f = curve_fit(gerade, x_fit_f, y_fit_f, sigma=del_y_fit_f,␣
 ↪absolute_sigma=True)
x_values_f = np.linspace(low_f1, high_f3, 1000)
fit_f = gerade(x_values_f, *bg_f)

df_fluorescence["clean"] =␣
 ↪df_fluorescence["running"]-(bg_f[0]*df_fluorescence["frequency"]+bg_f[1])
df_fluorescence["del_clean"] = np.sqrt(df_fluorescence["del_running"]**2
                            +(np.sqrt(del_bg_f[0][0])*df_fluorescence["frequency"])**2
                                      +(bg_f[0]*del_peaks_abs_f)**2+del_bg_f[1][1])


fig1 = plt.figure()
#FPI plot
frame1=fig1.add_axes((0,.6,1,.4))
plt.plot(df_fluorescence["frequency"][ll_f_id:ul_f_id], df_fluorescence["FPI"][ll_f_id:
 ↪ul_f_id], label="FPI signal")
for i in range(-moving_f, len(peaks_f)-moving_f):
    if i == 0:
        plt.axvline(FSR*i, color="red", label="Estimated peak positions with␣
 ↪uncertainties")
        plt.axvline(FSR*i-del_peaks_abs_f, linestyle="dashed", color="red")
        plt.axvline(FSR*i+del_peaks_abs_f, linestyle="dashed", color="red")
        plt.axvspan(FSR*i-del_peaks_abs_f, FSR*i+del_peaks_abs_f, alpha=0.4,␣
 ↪color="mistyrose")
    else:
        plt.axvline(FSR*i, color="red")
        plt.axvline(FSR*i-del_peaks_abs_f, linestyle="dashed", color="red")
        plt.axvline(FSR*i+del_peaks_abs_f, linestyle="dashed", color="red")
        plt.axvspan(FSR*i-del_peaks_abs_f, FSR*i+del_peaks_abs_f, alpha=0.4,␣
 ↪color="mistyrose")
frame1.set_xticklabels([])
plt.xlim(ll_f_plot,ul_f_plot)
plt.ylabel("Amplitude [mV]")
plt.grid()
plt.legend(loc=9, ncol=2, bbox_to_anchor=(0.5, -1.7))

#PD plot
```

```
frame2=fig1.add_axes((0,0,1,.6))
plt.plot(x_values_f, fit_f, color="maroon", label="Linear underground fit")
for i in range(len(low_f)):
    if i == 0:
        plt.plot(df_fluorescence["frequency"][low_f[i]:high_f[i]],␣
 ↪df_fluorescence["running"][low_f[i]:high_f[i]], color="orange", label="Data for linear␣
 ↪fit")
    else:
        plt.plot(df_fluorescence["frequency"][low_f[i]:high_f[i]],␣
 ↪df_fluorescence["running"][low_f[i]:high_f[i]], color="orange")
plt.plot(df_fluorescence["frequency"][high_f[0]:low_f[1]],␣
 ↪df_fluorescence["running"][high_f[0]:low_f[1]], color="green", label="Photodiode␣
 ↪signal")
plt.plot(df_fluorescence["frequency"][high_f[1]:low_f[2]],␣
 ↪df_fluorescence["running"][high_f[1]:low_f[2]], color="green")
plt.plot(df_fluorescence["frequency"][high_f[2]:ul_f_id],␣
 ↪df_fluorescence["running"][high_f[2]:ul_f_id], color="green")

plt.xlabel("Frequency difference to lowest FPI Peak [MHz]")
plt.ylabel("Amplitude [mV]")
plt.xlim(ll_f_plot,ul_f_plot)
plt.legend(loc=9, ncol=2, bbox_to_anchor=(0.5, -0.25))
plt.grid()

if saving == True:
    plt.savefig(savepath + r"\fluorescence_bg.pdf", dpi=150, bbox_inches="tight")

plt.show()
```

4.4 Finding the peaks

[ ]:
```
fig1 = plt.figure()
#FPI plot
frame1=fig1.add_axes((0,.6,1,.4))
plt.plot(df_fluorescence["frequency"][ll_f_id:ul_f_id], df_fluorescence["FPI"][ll_f_id:
 ↪ul_f_id], label="FPI signal")
for i in range(-moving_f, len(peaks_f)-moving_f):
    if i == 0:
        plt.axvline(FSR*i, color="red", label="Estimated peak positions with␣
 ↪uncertainties")
        plt.axvline(FSR*i-del_peaks_abs_f, linestyle="dashed", color="red")
        plt.axvline(FSR*i+del_peaks_abs_f, linestyle="dashed", color="red")
        plt.axvspan(FSR*i-del_peaks_abs_f, FSR*i+del_peaks_abs_f, alpha=0.4,␣
 ↪color="mistyrose")
    else:
        plt.axvline(FSR*i, color="red")
        plt.axvline(FSR*i-del_peaks_abs_f, linestyle="dashed", color="red")
        plt.axvline(FSR*i+del_peaks_abs_f, linestyle="dashed", color="red")
        plt.axvspan(FSR*i-del_peaks_abs_f, FSR*i+del_peaks_abs_f, alpha=0.4,␣
 ↪color="mistyrose")
frame1.set_xticklabels([])
plt.xlim(ll_f_plot,ul_f_plot)
plt.ylabel("Amplitude [mV]")
plt.grid()
plt.legend(loc=9, ncol=2, bbox_to_anchor=(0.5, -1.7))
```

```
#PD plot
frame2=fig1.add_axes((0,0,1,.6))
plt.plot(df_fluorescence["frequency"][ll_f_id:ul_f_id], df_fluorescence["clean"][ll_f_id:
 ↪ul_f_id], color="green", label="Data after underground subtraction")


plt.grid()
plt.xlabel("Frequency difference to lowest FPI Peak [MHz]")
plt.ylabel("Amplitude [mV]")
plt.xlim(ll_f_plot,ul_f_plot)
plt.legend(loc=9, ncol=2, bbox_to_anchor=(0.5, -0.25))

if saving == True:
    plt.savefig(savepath + r"\fluorescence_clean.pdf", dpi=150, bbox_inches="tight")
plt.show()
```

[ ]:
```
peaks_f_low = [750, 1800, 5500]
peaks_f_low = [(df_fluorescence["frequency"] > i).idxmax() for i in peaks_f_low]
peaks_f_high = [1400, 3500, 6500]
peaks_f_high = [(df_fluorescence["frequency"] > i).idxmax() for i in peaks_f_high]
amp_f = [0.9, 0.9, 2.3, 1.5]  # Amplitudes
mean_f = [1100, 2000, 2800, 6000]  # Means
std_f = [200, 200, 200, 200]  # Standard deviations
```

[ ]:
```
#Peak 1
fit_g1_f, fitcov_g1_f = curve_fit(gaus, df_fluorescence["frequency"][peaks_f_low[0]:
 ↪peaks_f_high[0]], df_fluorescence["clean"][peaks_f_low[0]:peaks_f_high[0]],␣
 ↪p0=[amp_f[0], mean_f[0], std_f[0]], sigma=df_fluorescence["del_clean"][peaks_f_low[0]:
 ↪peaks_f_high[0]], absolute_sigma=True, maxfev=2000000)
x_fit_g1_f = df_fluorescence["frequency"][peaks_f_low[0]:peaks_f_high[0]]
y_fit_g1_f = gaus(df_fluorescence["frequency"][peaks_f_low[0]:peaks_f_high[0]], *fit_g1_f)
x_plot_g1_f = np.linspace(ll_f_plot, ul_f_plot, 2000)
y_plot_g1_f = gaus(x_plot_g1_f, *fit_g1_f)

amp_g1_f = fit_g1_f[0]
del_amp_g1_f = np.sqrt(fitcov_g1_f[0][0])
mean_g1_f = fit_g1_f[1]
del_mean_g1_f = np.sqrt((fit_g1_f[2]/np.
 ↪sqrt(peaks_f_high[0]-peaks_f_low[0]))**2+del_peaks_abs_f**2)


#Doublepeak 1
fit_dg1_f, fitcov_dg1_f = curve_fit(doppelgaus,␣
 ↪df_fluorescence["frequency"][peaks_f_low[1]:peaks_f_high[1]],␣
 ↪df_fluorescence["clean"][peaks_f_low[1]:peaks_f_high[1]], p0=[amp_f[1], mean_f[1],␣
 ↪std_f[1], amp_f[2], mean_f[2], std_f[2]],␣
 ↪sigma=df_fluorescence["del_clean"][peaks_f_low[1]:peaks_f_high[1]],␣
 ↪absolute_sigma=True, maxfev=2000000)
x_fit_dg1_f = df_fluorescence["frequency"][peaks_f_low[1]:peaks_f_high[1]]
y_fit_dg1_f = doppelgaus(df_fluorescence["frequency"][peaks_f_low[1]:peaks_f_high[1]],␣
 ↪*fit_dg1_f)
x_plot_dg1_f = np.linspace(ll_f_plot, ul_f_plot, 2000)
y_plot_dg1_f = doppelgaus(x_plot_dg1_f, *fit_dg1_f)

amp1_dg1_f = fit_dg1_f[0]
```

```
del_amp1_dg1_f = np.sqrt(fitcov_dg1_f[0][0])
mean1_dg1_f = fit_dg1_f[1]
del_mean1_dg1_f = np.sqrt((fit_dg1_f[2]/np.
 ↪sqrt(peaks_f_high[1]-peaks_f_low[1]))**2+del_peaks_abs_f**2)
amp2_dg1_f = fit_dg1_f[3]
del_amp2_dg1_f = np.sqrt(fitcov_dg1_f[3][3])
mean2_dg1_f = fit_dg1_f[4]
del_mean2_dg1_f = np.sqrt((fit_dg1_f[5]/np.
 ↪sqrt(peaks_f_high[1]-peaks_f_low[1]))**2+del_peaks_abs_f**2)


#Peak 2
fit_g2_f, fitcov_g2_f = curve_fit(gaus, df_fluorescence["frequency"][peaks_f_low[2]:
 ↪peaks_f_high[2]], df_fluorescence["clean"][peaks_f_low[2]:peaks_f_high[2]],␣
 ↪p0=[amp_f[3], mean_f[3], std_f[3]], sigma=df_fluorescence["del_clean"][peaks_f_low[2]:
 ↪peaks_f_high[2]], absolute_sigma=True, maxfev=2000000)
x_fit_g2_f = df_fluorescence["frequency"][peaks_f_low[2]:peaks_f_high[2]]
y_fit_g2_f = gaus(df_fluorescence["frequency"][peaks_f_low[2]:peaks_f_high[2]], *fit_g2_f)
x_plot_g2_f = np.linspace(ll_f_plot, ul_f_plot, 2000)
y_plot_g2_f = gaus(x_plot_g2_f, *fit_g2_f)

amp_g2_f = fit_g2_f[0]
del_amp_g2_f = np.sqrt(fitcov_g2_f[0][0])
mean_g2_f = fit_g2_f[1]
del_mean_g2_f = np.sqrt((fit_g2_f[2]/np.
 ↪sqrt(peaks_f_high[2]-peaks_f_low[2]))**2+del_peaks_abs_f**2)
```

[ ]:
```
#calculating reduced chi2 values
chi2red1_f = Chi2(df_fluorescence["clean"][peaks_f_low[0]:peaks_f_high[0]], y_fit_g1_f,␣
 ↪df_fluorescence["del_clean"][peaks_f_low[0]:peaks_f_high[0]])/
 ↪(peaks_f_high[0]-peaks_f_low[0]-3)
chi2red2_f = Chi2(df_fluorescence["clean"][peaks_f_low[1]:peaks_f_high[1]], y_fit_dg1_f,␣
 ↪df_fluorescence["del_clean"][peaks_f_low[1]:peaks_f_high[1]])/
 ↪(peaks_f_high[1]-peaks_f_low[1]-6)
chi2red3_f = Chi2(df_fluorescence["clean"][peaks_f_low[2]:peaks_f_high[2]], y_fit_g2_f,␣
 ↪df_fluorescence["del_clean"][peaks_f_low[2]:peaks_f_high[2]])/
 ↪(peaks_f_high[2]-peaks_f_low[2]-3)
```

[ ]:
```
print("Amplitude for the 1.Peak:", amp_g1_f, "+-", del_amp_g1_f)
print("Mean for the 1.Peak:", mean_g1_f, "+-", del_mean_g1_f)
print("Reduced chi^2-value:", chi2red1_f, "\n")

print("Amplitude for the 2.Peak:", amp1_dg1_f, "+-", del_amp1_dg1_f)
print("Mean for the 2.Peak:", mean1_dg1_f, "+-", del_mean1_dg1_f)
print("Amplitude for the 3.Peak:", amp2_dg1_f, "+-", del_amp2_dg1_f)
print("Mean for the 3.Peak:", mean2_dg1_f, "+-", del_mean2_dg1_f)
print("Reduced chi^2-value:", chi2red2_f, "\n")

print("Amplitude for the 4.Peak:", amp_g2_f, "+-", del_amp_g2_f)
print("Mean for the 4.Peak:", mean_g2_f, "+-", del_mean_g2_f)
print("Reduced chi^2-value:", chi2red3_f, "\n")
```

[ ]:
```
f_1 = 0
del_f_1 = np.sqrt(2)*del_mean_g1_f
f_2 = mean1_dg1_f - mean_g1_f
del_f_2 = np.sqrt(del_mean1_dg1_f**2 + del_mean_g1_f**2)
```

```
f_3 = mean2_dg1_f - mean_g1_f
del_f_3 = np.sqrt(del_mean2_dg1_f**2 + del_mean_g1_f**2)
f_4 = mean_g2_f - mean_g1_f
del_f_4 = np.sqrt(del_mean_g2_f**2 + del_mean_g1_f**2)


rel_means_f = [f_1,f_2,f_3,f_4]
del_rel_means_f = [del_f_1,del_f_2,del_f_3,del_f_4]

for i in range(len(rel_means_f)):
    print(f"{i+1}.Peak position relative to first peak:{rel_means_f[i]} +-␣
 ↪{del_rel_means_f[i]}")
```

[ ]:
```
plt.plot(df_fluorescence["frequency"][ll_f_id:ul_f_id], df_fluorescence["clean"][ll_f_id:
 ↪ul_f_id], color="green", label="Data after underground subtraction")
plt.plot(x_plot_g1_f, y_plot_g1_f, label=f"{mean_g1_f:.2f} $\pm$ {del_mean_g1_f:.2f} MHz")
plt.plot(x_plot_dg1_f, y_plot_dg1_f, label=f"{mean1_dg1_f:.2f} $\pm$ {del_mean1_dg1_f:.2f}␣
 ↪MHz\n{mean2_dg1_f:.2f} $\pm$ {del_mean2_dg1_f:.2f} MHz")
plt.plot(x_plot_g2_f, y_plot_g2_f, label=f"{mean_g2_f:.2f} $\pm$ {del_mean_g2_f:.2f} MHz")

for i in range(0,len(peaks_f_low)):
    plt.axvline(df_fluorescence["frequency"][peaks_f_low[i]], ls="-", color='black')
    plt.axvline(df_fluorescence["frequency"][peaks_f_high[i]], ls="-", color='black')
plt.xlim(ll_f_plot,ul_f_plot)
plt.grid()
plt.legend()
if saving == True:
    plt.savefig(savepath + r"\fluorescence_fits.pdf", dpi=150, bbox_inches="tight")
```

[ ]:
```
fig1 = plt.figure()
#FPI plot
frame1=fig1.add_axes((0,.6,1,.4))
plt.plot(df_fluorescence["frequency"][ll_f_id:ul_f_id], df_fluorescence["FPI"][ll_f_id:
 ↪ul_f_id], label="FPI signal")
for i in range(-moving_f, len(peaks_f)-moving_f):
    if i == 0:
        plt.axvline(FSR*i, color="red", label="Estimated peak positions with␣
 ↪uncertainties")
        plt.axvline(FSR*i-del_peaks_abs_f, linestyle="dashed", color="red")
        plt.axvline(FSR*i+del_peaks_abs_f, linestyle="dashed", color="red")
        plt.axvspan(FSR*i-del_peaks_abs_f, FSR*i+del_peaks_abs_f, alpha=0.4,␣
 ↪color="mistyrose")
    else:
        plt.axvline(FSR*i, color="red")
        plt.axvline(FSR*i-del_peaks_abs_f, linestyle="dashed", color="red")
        plt.axvline(FSR*i+del_peaks_abs_f, linestyle="dashed", color="red")
        plt.axvspan(FSR*i-del_peaks_abs_f, FSR*i+del_peaks_abs_f, alpha=0.4,␣
 ↪color="mistyrose")
frame1.set_xticklabels([])
plt.xlim(ll_f_plot,ul_f_plot)
plt.ylabel("Amplitude [mV]")
plt.grid()
plt.legend(loc=9, ncol=2, bbox_to_anchor=(0.5, -1.7))

#PD plot
frame2=fig1.add_axes((0,0,1,.6))
```

```
plt.plot(df_fluorescence["frequency"][ll_f_id:ul_f_id], df_fluorescence["clean"][ll_f_id:
 ↪ul_f_id], color="green", label="Data after underground subtraction")
plt.plot(x_plot_g1_f, y_plot_g1_f, color="orange", label=f"Single Gaussian fluorescence␣
 ↪peak fit")
plt.plot(x_plot_dg1_f, y_plot_dg1_f, color="blue", label=f"Double Gaussian fluorescence␣
 ↪peak fit")
plt.plot(x_plot_g2_f, y_plot_g2_f, color="orange")

for i in range(0,len(peaks_f_low)):
    plt.axvline(df_fluorescence["frequency"][peaks_f_low[i]], ls="-", color='black')
    plt.axvline(df_fluorescence["frequency"][peaks_f_high[i]], ls="-", color='black')


plt.grid()
plt.xlabel("Frequency difference to lowest FPI Peak [MHz]")
plt.ylabel("Amplitude [mV]")
plt.xlim(ll_f_plot,ul_f_plot)
plt.legend(loc=9, ncol=2, bbox_to_anchor=(0.5, -0.25))

if saving == True:
    plt.savefig(savepath + r"\fluorescence.pdf", dpi=150, bbox_inches="tight")
plt.show()
```

4.4.1 Abundance and hyperfine

[ ]:
```
#abundance
abundance_f = (amp_g1_f+amp1_dg1_f)/(amp_g1_f+amp1_dg1_f+amp2_dg1_f)
del_abundance_f = np.sqrt(((amp2_dg1_f)/
 ↪((amp_g1_f+amp1_dg1_f+amp2_dg1_f)**2))**2*del_amp_g1_f**2+(amp2_dg1_f/
 ↪((amp_g1_f+amp1_dg1_f+amp2_dg1_f)**2))**2*del_amp1_dg1_f**2+((amp_g1_f+amp1_dg1_f)/
 ↪((amp_g1_f+amp1_dg1_f+amp2_dg1_f)**2))**2*del_amp2_dg1_f**2)

abundance_f = (amp_g1_f+amp1_dg1_f)/(amp_g1_f+amp1_dg1_f+2*amp2_dg1_f)
del_abundance_f = np.sqrt(((2*amp2_dg1_f)/
 ↪((amp_g1_f+amp1_dg1_f+2*amp2_dg1_f)**2))**2*del_amp_g1_f**2+(2*amp2_dg1_f/
 ↪((amp_g1_f+amp1_dg1_f+2*amp2_dg1_f)**2))**2*del_amp1_dg1_f**2+(2*(amp_g1_f+amp1_dg1_f)/
 ↪((amp_g1_f+amp1_dg1_f+2*amp2_dg1_f)**2))**2*del_amp2_dg1_f**2)

#hyperfine (85 not possible due to too low resolution)
hyper_f_87 = f_2/2
del_hyper_f_87 = np.sqrt((del_f_2/2)**2+(del_f_1/2)**2)

abundance_f, del_abundance_f, hyper_f_87, del_hyper_f_87
```

5 Saturation spectroscopy

[ ]:
```
satur = r"\Week 2\saturation.csv"
saturation = path + satur
df_saturation = pd.read_csv(saturation,header=None, sep=";", skiprows=1, names=["Piezo",␣
 ↪"PD", "FPI"], index_col=False)
df_saturation.iloc[:, :] = df_saturation.iloc[:, :] * 1000 #transform everything to mV
df_saturation["del_PD"] = 0.001*df_saturation["PD"]
df_saturation
```

5.1 FPI calibration

[ ]:
```
peaks_s, _ = find_peaks(df_saturation["FPI"], height=2.5, distance=50)
del_peaks_s = 2
difference_s = np.diff(peaks_s)

fig1 = plt.figure()
#FPI plot
frame1=fig1.add_axes((0,.6,1,.4))
plt.plot(df_saturation.index, df_saturation["FPI"], label="FPI signal")
for i in range(0,len(peaks_s)):
    if i == 0:  # Add label only for the first iteration
        plt.axvline(peaks_s[i], color="red", label="Estimated peak positions with␣
 ↪uncertainties")
        plt.axvline(peaks_s[i]-del_peaks_s, linestyle="dashed", color="red")
        plt.axvline(peaks_s[i]+del_peaks_s, linestyle="dashed", color="red")
        plt.axvspan(peaks_s[i]-del_peaks_s, peaks_s[i]+del_peaks_s, alpha=0.4,␣
 ↪color="mistyrose")
    else:  # No label for subsequent plots
        plt.axvline(peaks_s[i], color="red")
        plt.axvline(peaks_s[i]-del_peaks_s, linestyle="dashed", color="red")
        plt.axvline(peaks_s[i]+del_peaks_s, linestyle="dashed", color="red")
        plt.axvspan(peaks_s[i]-del_peaks_s, peaks_s[i]+del_peaks_s, alpha=0.4,␣
 ↪color="mistyrose")
frame1.set_xticklabels([])
plt.xlim(0,df_saturation.index[-1])
plt.ylabel("Amplitude [mV]")
plt.grid()
plt.legend(loc=9, ncol=2, bbox_to_anchor=(0.5, -1.7))

#PD plot
frame2=fig1.add_axes((0,0,1,.6))
plt.plot(df_saturation.index, df_saturation["PD"], color="green", label="Photodiode␣
 ↪signal")
plt.grid()
plt.xlabel("Channel")
plt.ylabel("Amplitude [mV]")
plt.xlim(0,df_saturation.index[-1])
plt.legend(loc=9, ncol=2, bbox_to_anchor=(0.5, -0.25))

if saving == True:
    plt.savefig(savepath + r"\FPI_peaks_s.pdf", dpi=150, bbox_inches="tight")
plt.show()

peaks_s, difference_s
```

[ ]:
```
plt.plot(df_saturation.index, df_saturation["FPI"], label="FPI signal")
for i in range(0,len(peaks_s)):
    if i == 0:  # Add label only for the first iteration
        plt.axvline(peaks_s[i], color="red", label="Estimated peak positions with␣
 ↪uncertainties")
        plt.axvline(peaks_s[i]-del_peaks_s, linestyle="dashed", color="red")
        plt.axvline(peaks_s[i]+del_peaks_s, linestyle="dashed", color="red")
        plt.axvspan(peaks_s[i]-del_peaks_s, peaks_s[i]+del_peaks_s, alpha=0.4,␣
 ↪color="mistyrose")
    else:  # No label for subsequent plots
```

```
        plt.axvline(peaks_s[i], color="red")
        plt.axvline(peaks_s[i]-del_peaks_s, linestyle="dashed", color="red")
        plt.axvline(peaks_s[i]+del_peaks_s, linestyle="dashed", color="red")
        plt.axvspan(peaks_s[i]-del_peaks_s, peaks_s[i]+del_peaks_s, alpha=0.4,␣
 ↪color="mistyrose")
plt.xlim(0,df_saturation.index[-1])
plt.xlabel("Channel")
plt.ylabel("Amplitude [mV]")
plt.grid()
plt.legend(loc=9, ncol=2, bbox_to_anchor=(0.5, -.1))
if saving == True:
    plt.savefig(savepath + r"\FPI_s.pdf", dpi=150, bbox_inches="tight")
plt.show()
```

[ ]:
```
# Create the expected evenly spaced grid
expected_peaks_s = np.linspace(0, len(df_saturation), num=len(peaks_s))
# Interpolation function
interp_func_s = interp.interp1d(peaks_s, expected_peaks_s, kind='linear',␣
 ↪fill_value="extrapolate")
# Generate corrected values
df_saturation["conversion"] = interp_func_s(df_saturation.index)
#look if everything is equally spaced now
equal_s = np.diff(df_saturation["conversion"][peaks_s])
#convert to frequency
del_peaks_abs_s = FSR*(del_peaks_s)/equal_s[0]
```

[ ]:
```
#Calculate frequencies and move the 0 point to the third peak
df_saturation["frequency"] = FSR*(df_saturation["conversion"])/equal_s[0]
df_saturation
```

5.1.1 Defining limits for plots

[ ]:
```
ll_s_plot = df_saturation["frequency"][0]
ul_s_plot = df_saturation["frequency"].iloc[-1]
```

5.2 Analysis

[ ]:
```
fig1 = plt.figure()
#FPI plot
frame1=fig1.add_axes((0,.6,1,.4))
plt.plot(df_saturation["frequency"], df_saturation["FPI"], label="FPI signal")
for i in range(len(peaks_s)):
    if i == 0:
        plt.axvline(FSR*i, color="red", label="Estimated peak positions with␣
 ↪uncertainties")
        plt.axvline(FSR*i-del_peaks_abs_s, linestyle="dashed", color="red")
        plt.axvline(FSR*i+del_peaks_abs_s, linestyle="dashed", color="red")
        plt.axvspan(FSR*i-del_peaks_abs_s, FSR*i+del_peaks_abs_s, alpha=0.4,␣
 ↪color="mistyrose")
    else:
        plt.axvline(FSR*i, color="red")
        plt.axvline(FSR*i-del_peaks_abs_s, linestyle="dashed", color="red")
        plt.axvline(FSR*i+del_peaks_abs_s, linestyle="dashed", color="red")
        plt.axvspan(FSR*i-del_peaks_abs_s, FSR*i+del_peaks_abs_s, alpha=0.4,␣
 ↪color="mistyrose")
```

```
frame1.set_xticklabels([])
plt.xlim(ll_s_plot,ul_s_plot)
plt.ylabel("Amplitude [mV]")
plt.grid()
plt.legend(loc=9, ncol=2, bbox_to_anchor=(0.5, -1.7))

#PD plot
frame2=fig1.add_axes((0,0,1,.6))
plt.plot(df_saturation["frequency"], df_saturation["PD"], color="green", label="Photodiode␣
 ↪signal")
plt.grid()
plt.xlabel("Frequency difference to lowest FPI peak [MHz]")
plt.ylabel("Amplitude [mV]")
plt.xlim(ll_s_plot,ul_s_plot)
plt.legend(loc=9, ncol=2, bbox_to_anchor=(0.5, -0.25))

if saving == True:
    plt.savefig(savepath + r"\saturation_freq.pdf", dpi=150, bbox_inches="tight")
plt.show()
```

5.3 Removing background

[ ]:
```
low_s1 = df_saturation["frequency"][0]
high_s1 = 200
low_s2 = 3300
high_s2 = 4800
low_s3 = 9300
high_s3 = df_saturation["frequency"].iloc[-2]

low_s = [(df_saturation["frequency"] > low_s1).idxmax(), (df_saturation["frequency"] >␣
 ↪low_s2).idxmax(), (df_saturation["frequency"] > low_s3).idxmax()]
high_s = [(df_saturation["frequency"] > high_s1).idxmax(), (df_saturation["frequency"] >␣
 ↪high_s2).idxmax(), (df_saturation["frequency"] > high_s3).idxmax()]
```

[ ]:
```
x_fit_s = np.concatenate([
    df_saturation["frequency"][low_s[0]:high_s[0]],
    df_saturation["frequency"][low_s[1]:high_s[1]],
    df_saturation["frequency"][low_s[2]:high_s[2]]
])

y_fit_s = np.concatenate([
    df_saturation["PD"][low_s[0]:high_s[0]],
    df_saturation["PD"][low_s[1]:high_s[1]],
    df_saturation["PD"][low_s[2]:high_s[2]]
])

del_y_fit_s = np.concatenate([
    df_saturation["del_PD"][low_s[0]:high_s[0]],
    df_saturation["del_PD"][low_s[1]:high_s[1]],
    df_saturation["del_PD"][low_s[2]:high_s[2]]
])

bg_s, del_bg_s = curve_fit(gerade, x_fit_s, y_fit_s, sigma=del_y_fit_s,␣
 ↪absolute_sigma=True)
x_values_s = np.linspace(low_s1, high_s3, 1000)
```

```
fit_s = gerade(x_values_s, *bg_s)

df_saturation["clean"] = df_saturation["PD"]-(bg_s[0]*df_saturation["frequency"]+bg_s[1])
df_saturation["del_clean"] = np.sqrt(df_saturation["del_PD"]**2+(np.
 ↪sqrt(del_bg_s[0][0])*df_saturation["frequency"])**2
                                    +(bg_s[0]*del_peaks_abs_s)**2+del_bg_s[1][1])


fig1 = plt.figure()
#FPI plot
frame1=fig1.add_axes((0,.6,1,.4))
plt.plot(df_saturation["frequency"], df_saturation["FPI"], label="FPI signal")
for i in range(len(peaks_f)):
    if i == 0:
        plt.axvline(FSR*i, color="red", label="Estimated peak positions with␣
 ↪uncertainties")
        plt.axvline(FSR*i-del_peaks_abs_s, linestyle="dashed", color="red")
        plt.axvline(FSR*i+del_peaks_abs_s, linestyle="dashed", color="red")
        plt.axvspan(FSR*i-del_peaks_abs_s, FSR*i+del_peaks_abs_s, alpha=0.4,␣
 ↪color="mistyrose")
    else:
        plt.axvline(FSR*i, color="red")
        plt.axvline(FSR*i-del_peaks_abs_s, linestyle="dashed", color="red")
        plt.axvline(FSR*i+del_peaks_abs_s, linestyle="dashed", color="red")
        plt.axvspan(FSR*i-del_peaks_abs_s, FSR*i+del_peaks_abs_s, alpha=0.4,␣
 ↪color="mistyrose")
frame1.set_xticklabels([])
plt.xlim(ll_s_plot,ul_s_plot)
plt.ylabel("Amplitude [mV]")
plt.grid()
plt.legend(loc=9, ncol=2, bbox_to_anchor=(0.5, -1.7))

#PD plot
frame2=fig1.add_axes((0,0,1,.6))
plt.plot(x_values_s, fit_s, color="maroon", label="Linear underground fit")
for i in range(len(low_s)):
    if i == 0:
        plt.plot(df_saturation["frequency"][low_s[i]:high_s[i]],␣
 ↪df_saturation["PD"][low_s[i]:high_s[i]], color="orange", label="Data for linear fit")
    else:
        plt.plot(df_saturation["frequency"][low_s[i]:high_s[i]],␣
 ↪df_saturation["PD"][low_s[i]:high_s[i]], color="orange")
plt.plot(df_saturation["frequency"][high_s[0]:low_s[1]], df_saturation["PD"][high_s[0]:
 ↪low_s[1]], color="green", label="Photodiode signal")
plt.plot(df_saturation["frequency"][high_s[1]:low_s[2]], df_saturation["PD"][high_s[1]:
 ↪low_s[2]], color="green")

plt.xlabel("Frequency difference to lowest FPI Peak [MHz]")
plt.ylabel("Amplitude [mV]")
plt.xlim(ll_s_plot,ul_s_plot)
plt.legend(loc=9, ncol=2, bbox_to_anchor=(0.5, -0.25))
plt.grid()

if saving == True:
    plt.savefig(savepath + r"\saturation_bg.pdf", dpi=150, bbox_inches="tight")
```

```
plt.show()
```

5.4 Finding the peaks

[ ]:
```
fig1 = plt.figure()
#FPI plot
frame1=fig1.add_axes((0,.6,1,.4))
plt.plot(df_saturation["frequency"], df_saturation["FPI"], label="FPI signal")
for i in range(len(peaks_s)):
    if i == 0:
        plt.axvline(FSR*i, color="red", label="Estimated peak positions with␣
 ↪uncertainties")
        plt.axvline(FSR*i-del_peaks_abs_s, linestyle="dashed", color="red")
        plt.axvline(FSR*i+del_peaks_abs_s, linestyle="dashed", color="red")
        plt.axvspan(FSR*i-del_peaks_abs_s, FSR*i+del_peaks_abs_s, alpha=0.4,␣
 ↪color="mistyrose")
    else:
        plt.axvline(FSR*i, color="red")
        plt.axvline(FSR*i-del_peaks_abs_s, linestyle="dashed", color="red")
        plt.axvline(FSR*i+del_peaks_abs_s, linestyle="dashed", color="red")
        plt.axvspan(FSR*i-del_peaks_abs_s, FSR*i+del_peaks_abs_s, alpha=0.4,␣
 ↪color="mistyrose")
frame1.set_xticklabels([])
plt.xlim(ll_s_plot,ul_s_plot)
plt.ylabel("Amplitude [mV]")
plt.grid()
plt.legend(loc=9, ncol=2, bbox_to_anchor=(0.5, -1.7))

#PD plot
frame2=fig1.add_axes((0,0,1,.6))
plt.plot(df_saturation["frequency"], df_saturation["clean"], color="green", label="Data␣
 ↪after underground subtraction")


plt.grid()
plt.xlabel("Frequency difference to lowest FPI Peak [MHz]")
plt.ylabel("Amplitude [mV]")
plt.xlim(ll_s_plot,ul_s_plot)
plt.legend(loc=9, ncol=2, bbox_to_anchor=(0.5, -0.25))

if saving == True:
    plt.savefig(savepath + r"\saturation_clean.pdf", dpi=150, bbox_inches="tight")
plt.show()
```

[ ]:
```
peaks_s_low = [760, 1580, 2310, 2480, 2660, 5380, 5520, 5700, 7600, 8420]
peaks_s_low = [(df_saturation["frequency"] > i).idxmax() for i in peaks_s_low]
peaks_s_high = [890, 1780, 2390, 2570, 2760, 5480, 5650, 5800, 7760, 8570]
peaks_s_high = [(df_saturation["frequency"] > i).idxmax() for i in peaks_s_high]
amp_s = [0.4, 0.4, .9, 1.5, 4, 0.1, 4, 1, 4, 0.4]  # Amplitudes
mean_s = [840, 1680, 2360, 2550, 2715, 5400, 5580, 5750, 7680, 8500]  # Means
std_s = [40]*10  # Standard deviations
offset_s = [-2.8, -1.1, -5.8, -4.8, -3.4, -2.2, -3, -2.5, 0.2, -0.2]
```

[ ]:
```
amplitudes_s = []
del_amplitudes_s = []
means_s = []
del_means_s = []
x_plot_s = []
y_plot_s = []
chi2red = []

for i in range(len(peaks_s_low)):
    fit, fitcov = curve_fit(lorentz, df_saturation["frequency"][peaks_s_low[i]:
 ↪peaks_s_high[i]], df_saturation["clean"][peaks_s_low[i]:peaks_s_high[i]], p0=[amp_s[i],␣
 ↪mean_s[i], std_s[i], offset_s[i]], sigma=df_saturation["del_clean"][peaks_s_low[i]:
 ↪peaks_s_high[i]], absolute_sigma=True, maxfev=2000000)
    x_fit = df_saturation["frequency"][peaks_s_low[i]:peaks_s_high[i]]
    y_fit = lorentz(df_saturation["frequency"][peaks_s_low[i]:peaks_s_high[i]], *fit)
    x_plot_s.append(np.linspace(df_saturation["frequency"][peaks_s_low[i]]
                                ,df_saturation["frequency"][peaks_s_high[i]], 2000))
    y_plot_s.append(lorentz(np.linspace(df_saturation["frequency"][peaks_s_low[i]]
                                        ,df_saturation["frequency"][peaks_s_high[i]],␣
 ↪2000), *fit))
    amplitudes_s.append(fit[0])
    del_amplitudes_s.append(np.sqrt(fitcov[0][0]))
    means_s.append(fit[1])
    del_means_s.append(np.sqrt((fit[2]/np.
 ↪sqrt(peaks_s_high[i]-peaks_s_low[i]))**2+del_peaks_abs_s**2))
    chi2 = Chi2(df_saturation["clean"][peaks_s_low[i]:peaks_s_high[i]], y_fit,␣
 ↪df_saturation["del_clean"][peaks_s_low[i]:peaks_s_high[i]])/
 ↪(peaks_s_high[i]-peaks_s_low[i]-4)
    chi2red.append(chi2)
```

[ ]:
```
for i in range(len(peaks_s_low)):
    print(f"Amplitude for the {i+1}.Peak: {amplitudes_s[i]} +- {del_amplitudes_s[i]}")
    print(f"Mean for the {i+1}.Peak: {means_s[i]} +- {del_means_s[i]}")
    print(f"Reduced chi^2-value: {chi2red[i]}" "\n")
```

[ ]:
```
rel_means_s = [i-means_s[0] for i in means_s]
del_rel_means_s = [np.sqrt(i**2 + del_means_s[0]**2) for i in del_means_s]

for i in range(len(peaks_s_low)):
    print(f"{i+1}.Peak position relative to first peak: {rel_means_s[i]} +-␣
 ↪{del_rel_means_s[i]}")
```

[ ]:
```
plt.plot(df_saturation["frequency"], df_saturation["clean"], color="green", label="Data␣
 ↪after underground subtraction")

for i in range(len(peaks_s_low)):
    plt.plot(x_plot_s[i], y_plot_s[i], label=f"{means_s[i]:.2f} $\pm$ {del_means_s[i]:.2f}␣
 ↪MHz")

plt.xlim(ll_s_plot,ul_s_plot)
plt.grid()
plt.legend()
if saving == True:
    plt.savefig(savepath + r"\saturation_fits.pdf", dpi=150, bbox_inches="tight")
```

[ ]:
```
fig1 = plt.figure()
#FPI plot
frame1=fig1.add_axes((0,.6,1,.4))
plt.plot(df_saturation["frequency"], df_saturation["FPI"], label="FPI signal")
for i in range(len(peaks_s)):
    if i == 0:
        plt.axvline(FSR*i, color="red", label="Estimated peak positions with␣
 ↪uncertainties")
        plt.axvline(FSR*i-del_peaks_abs_s, linestyle="dashed", color="red")
        plt.axvline(FSR*i+del_peaks_abs_s, linestyle="dashed", color="red")
        plt.axvspan(FSR*i-del_peaks_abs_s, FSR*i+del_peaks_abs_s, alpha=0.4,␣
 ↪color="mistyrose")
    else:
        plt.axvline(FSR*i, color="red")
        plt.axvline(FSR*i-del_peaks_abs_s, linestyle="dashed", color="red")
        plt.axvline(FSR*i+del_peaks_abs_s, linestyle="dashed", color="red")
        plt.axvspan(FSR*i-del_peaks_abs_s, FSR*i+del_peaks_abs_s, alpha=0.4,␣
 ↪color="mistyrose")
frame1.set_xticklabels([])
plt.xlim(ll_s_plot,ul_s_plot)
plt.ylabel("Amplitude [mV]")
plt.grid()
plt.legend(loc=9, ncol=2, bbox_to_anchor=(0.5, -1.7))

#PD plot
frame2=fig1.add_axes((0,0,1,.6))
plt.plot(df_saturation["frequency"], df_saturation["clean"], color="green", label="Data␣
 ↪after underground subtraction")

for i in range(len(peaks_s_low)):
    if i == 0:
        plt.plot(x_plot_s[i], y_plot_s[i], color="orange", label=f"Saturation peaks")
    else:
        plt.plot(x_plot_s[i], y_plot_s[i], color="orange")


plt.grid()
plt.xlabel("Frequency difference to lowest FPI Peak [MHz]")
plt.ylabel("Amplitude [mV]")
plt.xlim(ll_s_plot,ul_s_plot)
plt.legend(loc=9, ncol=2, bbox_to_anchor=(0.5, -0.25))

if saving == True:
    plt.savefig(savepath + r"\saturation.pdf", dpi=150, bbox_inches="tight")
plt.show()
```

5.4.1 Abundance and hyperfine

[ ]:
```
#abundance
abundance_s1 = amplitudes_s[0]/(amplitudes_s[0]+amplitudes_s[2])
del_abundance_s1 = 1/(amplitudes_s[0]+amplitudes_s[2])**2*np.
 ↪sqrt(amplitudes_s[2]**2*del_amplitudes_s[0]**2+amplitudes_s[0]**2*del_amplitudes_s[2]**2)
abundance_s2 = amplitudes_s[1]/(amplitudes_s[1]+amplitudes_s[4])
del_abundance_s2 = 1/(amplitudes_s[1]+amplitudes_s[4])**2*np.
 ↪sqrt(amplitudes_s[4]**2*del_amplitudes_s[1]**2+amplitudes_s[1]**2*del_amplitudes_s[4]**2)
```

```
abundance_s3 = amplitudes_s[8]/(amplitudes_s[5]+amplitudes_s[8])
del_abundance_s3 = 1/(amplitudes_s[5]+amplitudes_s[8])**2*np.
 ↪sqrt(amplitudes_s[8]**2*del_amplitudes_s[5]**2+amplitudes_s[5]**2*del_amplitudes_s[8]**2)
abundance_s4 = amplitudes_s[9]/(amplitudes_s[7]+amplitudes_s[9])
del_abundance_s4 = 1/(amplitudes_s[7]+amplitudes_s[9])**2*np.
 ↪sqrt(amplitudes_s[9]**2*del_amplitudes_s[7]**2+amplitudes_s[7]**2*del_amplitudes_s[9]**2)

#hyperfine
hyper_s1_87 = (means_s[1]-means_s[0])/2
del_hyper_s1_87 = np.sqrt((del_means_s[1]/2)**2+(del_means_s[0]/2)**2)
hyper_s2_87 = (means_s[9]-means_s[8])/2
del_hyper_s2_87 = np.sqrt((del_means_s[9]/2)**2+(del_means_s[8]/2)**2)
hyper_s1_85 = (means_s[4]-means_s[2])/3
del_hyper_s1_85 = np.sqrt((del_means_s[4]/3)**2+(del_means_s[2]/3)**2)
hyper_s2_85 = (means_s[7]-means_s[5])/3
del_hyper_s2_85 = np.sqrt((del_means_s[7]/3)**2+(del_means_s[5]/3)**2)

print(f"The abundances are {abundance_s1:.2f} +- {del_abundance_s1:.2f} and {abundance_s2:.
 ↪2f} +- {del_abundance_s2:.2f}")
print(f"The abundances are {abundance_s3:.2f} +- {del_abundance_s3:.2f} and {abundance_s4:.
 ↪2f} +- {del_abundance_s4:.2f}")
print(f"The hyperfine structure of 87Rb is {hyper_s1_87:.2f} +- {del_hyper_s1_87:.2f} and␣
 ↪{hyper_s2_87:.2f} +- {del_hyper_s2_87:.2f}")
print(f"The hyperfine structure of 85Rb is {hyper_s1_85:.2f} +- {del_hyper_s1_85:.2f} and␣
 ↪{hyper_s2_85:.2f} +- {del_hyper_s2_85:.2f}")
```

6 Comparison of the peaks

[ ]:
```
text=["$^{87}$Rb$^{F=1}_{F=2}$", "$^{87}$Rb$^{F=2}_{F=2}$", "$^{85}$Rb$^{F=2}_{F=3}$",␣
 ↪"$^{85}$Rb$^{F=3}_{F=3}$", "$^{85}$Rb$^{F=2}_{F=2}$", "$^{85}$Rb$^{F=3}_{F=2}$",␣
 ↪"$^{87}$Rb$^{F=1}_{F=1}$", "$^{87}$Rb$^{F=2}_{F=1}$"]

fig, ax = plt.subplots(figsize=(16,12))

#Expectation
for i in range(len(MHz_lit)):
    if i == 0:
        ax.axvline(MHz_lit[i], linestyle="dashed", color="red", label="Expected $^{87}$Rb␣
 ↪peaks")
        ax.text(MHz_lit[i], .93, text[i], color='black', ha='center', va='bottom')
    elif i==1 or i==6 or i==7:
        ax.axvline(MHz_lit[i], linestyle="dashed", color="red")
        ax.text(MHz_lit[i], .93, text[i], color='black', ha='center', va='bottom')
    elif i == 3:
        ax.axvline(MHz_lit[i], linestyle="dashed", color="green", label="Expected␣
 ↪$^{85}$Rb peaks")
        ax.text(MHz_lit[i], .97, text[i], color='black', ha='center', va='bottom')
    elif i == 5:
        ax.axvline(MHz_lit[i], linestyle="dashed", color="green")
        ax.text(MHz_lit[i], .97, text[i], color='black', ha='center', va='bottom')
    else:
        ax.axvline(MHz_lit[i], linestyle="dashed", color="green")
        ax.text(MHz_lit[i], .93, text[i], color='black', ha='center', va='bottom')
```

```
#absorption
plt.errorbar(rel_means_a, [0.8] * len(rel_means_a), xerr=del_rel_means_a, marker="x",␣
 ↪capsize=5, color="blue", linestyle="None", label="Absorption peaks")

#fluorescence
plt.errorbar(rel_means_f, [0.4] * len(rel_means_f), xerr=del_rel_means_f, marker="x",␣
 ↪capsize=5, color="orange", linestyle="None", label="Fluorescence peaks")

#saturation
plt.errorbar(rel_means_s, [0.0] * len(rel_means_s), xerr=del_rel_means_s, marker="x",␣
 ↪capsize=5, color="black", linestyle="None", label="Saturation peaks")


ax.set_yticklabels([])
ax.set_xlabel(f"Frequency difference to {text[0]} [MHz]")
ax.set_ylim(-0.1,.9)
ax.grid()

handles, labels = ax.get_legend_handles_labels()
custom_order = [2, 3, 4, 0, 1]
ax.legend([handles[i] for i in custom_order], [labels[i] for i in custom_order],
          loc=9, ncol=2, bbox_to_anchor=(0.5, -0.1))

if saving == True:
    plt.savefig(savepath + r"\comparison.pdf", dpi=150, bbox_inches="tight")
plt.show()
```

7 Frequency stability

[ ]:
```
s1 = r"\Stability\s01.csv"
s1 = path + s1
df_s1 = pd.read_csv(s1,header=None, sep=",", skiprows=2, names=["X", "FPI", "AWG"],␣
 ↪index_col=False)
df_s1.iloc[:, :] = df_s1.iloc[:, :] * 1000 #transform everything to mV


s2 = r"\Stability\s02.csv"
s2 = path + s2
df_s2 = pd.read_csv(s2,header=None, sep=",", skiprows=2, names=["X", "FPI", "AWG"],␣
 ↪index_col=False)
df_s2.iloc[:, :] = df_s2.iloc[:, :] * 1000 #transform everything to mV

s3 = r"\Stability\s03.csv"
s3 = path + s3
df_s3 = pd.read_csv(s3,header=None, sep=",", skiprows=2, names=["X", "FPI", "AWG"],␣
 ↪index_col=False)
df_s3.iloc[:, :] = df_s3.iloc[:, :] * 1000 #transform everything to mV

s4 = r"\Stability\s04.csv"
s4 = path + s4
df_s4 = pd.read_csv(s4,header=None, sep=",", skiprows=2, names=["X", "FPI", "AWG"],␣
 ↪index_col=False)
df_s4.iloc[:, :] = df_s4.iloc[:, :] * 1000 #transform everything to mV
```

[ ]:
```
#plt.plot(df_s1.index, df_s1["AWG"]/200, label="AWG")
plt.plot(df_s1.index, df_s1["FPI"], label="FPI1")
#plt.plot(df_s2.index, df_s2["AWG"]/200, label="AWG")
plt.plot(df_s2.index, df_s2["FPI"], label="FPI2")
#plt.plot(df_s3.index, df_s3["AWG"]/200, label="AWG")
plt.plot(df_s3.index, df_s3["FPI"], label="FPI3")
#plt.plot(df_s4.index, df_s4["AWG"]/200, label="AWG")
plt.plot(df_s4.index, df_s4["FPI"], label="FPI4")

plt.title("Stability")
plt.ylabel("Amplitude [mV]")
plt.xlabel("Channel")
plt.legend()
plt.grid()
```

[ ]:
```
peaks_s1, _ = find_peaks(df_s1["FPI"], height=9, distance=50)
difference_s1 = np.diff(peaks_s1)
peaks_s2, _ = find_peaks(df_s2["FPI"], height=9, distance=50)
difference_s2 = np.diff(peaks_s2)
peaks_s3, _ = find_peaks(df_s3["FPI"], height=9, distance=50)
difference_s3 = np.diff(peaks_s3)
peaks_s4, _ = find_peaks(df_s4["FPI"], height=9, distance=50)
difference_s4 = np.diff(peaks_s4)
```

[ ]:
```
del_peaks_stab=3
```

[ ]:
```
plt.plot(df_s1.index, df_s1["FPI"], label="FPI")
for i in range(len(peaks_s1)):
    if i == 0:  # Add label only for the first iteration
        plt.axvline(peaks_s1[i], color="red", label="Estimated peak positions with␣
 ↪uncertainties")
        plt.axvline(peaks_s1[i]-del_peaks_stab, linestyle="dashed", color="red")
        plt.axvline(peaks_s1[i]+del_peaks_stab, linestyle="dashed", color="red")
        plt.axvspan(peaks_s1[i]-del_peaks_stab, peaks_s1[i]+del_peaks_stab, alpha=0.4,␣
 ↪color="mistyrose")
    else:  # No label for subsequent plots
        plt.axvline(peaks_s1[i], color="red")
        plt.axvline(peaks_s1[i]-del_peaks_stab, linestyle="dashed", color="red")
        plt.axvline(peaks_s1[i]+del_peaks_stab, linestyle="dashed", color="red")
        plt.axvspan(peaks_s1[i]-del_peaks_stab, peaks_s1[i]+del_peaks_stab, alpha=0.4,␣
 ↪color="mistyrose")

plt.legend(loc=9, ncol=2, bbox_to_anchor=(0.5, -.1))
plt.xlim(0,1200)
plt.ylabel("Amplitude [mV]")
plt.xlabel("Channel")
plt.grid()
if saving == True:
    plt.savefig(savepath + r"\stab_1.pdf", dpi=150, bbox_inches="tight")
plt.show()
```

[ ]:
```
plt.plot(df_s2.index, df_s2["FPI"], label="FPI")
for i in range(len(peaks_s2)):
    if i == 0:  # Add label only for the first iteration
        plt.axvline(peaks_s2[i], color="red", label="Estimated peak positions with␣
 ↪uncertainties")
```

```
        plt.axvline(peaks_s2[i]-del_peaks_stab, linestyle="dashed", color="red")
        plt.axvline(peaks_s2[i]+del_peaks_stab, linestyle="dashed", color="red")
        plt.axvspan(peaks_s2[i]-del_peaks_stab, peaks_s2[i]+del_peaks_stab, alpha=0.4,␣
↪color="mistyrose")
    else:  # No label for subsequent plots
        plt.axvline(peaks_s2[i], color="red")
        plt.axvline(peaks_s2[i]-del_peaks_stab, linestyle="dashed", color="red")
        plt.axvline(peaks_s2[i]+del_peaks_stab, linestyle="dashed", color="red")
        plt.axvspan(peaks_s2[i]-del_peaks_stab, peaks_s2[i]+del_peaks_stab, alpha=0.4,␣
↪color="mistyrose")

plt.legend(loc=9, ncol=2, bbox_to_anchor=(0.5, -.1))
plt.xlim(0,1200)
plt.ylabel("Amplitude [mV]")
plt.xlabel("Channel")
plt.grid()
if saving == True:
    plt.savefig(savepath + r"\stab_2.pdf", dpi=150, bbox_inches="tight")
plt.show()
```

[ ]:
```
plt.plot(df_s3.index, df_s3["FPI"], label="FPI")
for i in range(len(peaks_s3)):
    if i == 0:  # Add label only for the first iteration
        plt.axvline(peaks_s3[i], color="red", label="Estimated peak positions with␣
↪uncertainties")
        plt.axvline(peaks_s3[i]-del_peaks_stab, linestyle="dashed", color="red")
        plt.axvline(peaks_s3[i]+del_peaks_stab, linestyle="dashed", color="red")
        plt.axvspan(peaks_s3[i]-del_peaks_stab, peaks_s3[i]+del_peaks_stab, alpha=0.4,␣
↪color="mistyrose")
    else:  # No label for subsequent plots
        plt.axvline(peaks_s3[i], color="red")
        plt.axvline(peaks_s3[i]-del_peaks_stab, linestyle="dashed", color="red")
        plt.axvline(peaks_s3[i]+del_peaks_stab, linestyle="dashed", color="red")
        plt.axvspan(peaks_s3[i]-del_peaks_stab, peaks_s3[i]+del_peaks_stab, alpha=0.4,␣
↪color="mistyrose")

plt.legend(loc=9, ncol=2, bbox_to_anchor=(0.5, -.1))
plt.xlim(0,1200)
plt.ylabel("Amplitude [mV]")
plt.xlabel("Channel")
plt.grid()
if saving == True:
    plt.savefig(savepath + r"\stab_3.pdf", dpi=150, bbox_inches="tight")
plt.show()
```

[ ]:
```
plt.plot(df_s4.index, df_s4["FPI"], label="FPI")
for i in range(len(peaks_s4)):
    if i == 0:  # Add label only for the first iteration
        plt.axvline(peaks_s4[i], color="red", label="Estimated peak positions with␣
↪uncertainties")
        plt.axvline(peaks_s4[i]-del_peaks_stab, linestyle="dashed", color="red")
        plt.axvline(peaks_s4[i]+del_peaks_stab, linestyle="dashed", color="red")
        plt.axvspan(peaks_s4[i]-del_peaks_stab, peaks_s4[i]+del_peaks_stab, alpha=0.4,␣
↪color="mistyrose")
    else:  # No label for subsequent plots
```

```
        plt.axvline(peaks_s4[i], color="red")
        plt.axvline(peaks_s4[i]-del_peaks_stab, linestyle="dashed", color="red")
        plt.axvline(peaks_s4[i]+del_peaks_stab, linestyle="dashed", color="red")
        plt.axvspan(peaks_s4[i]-del_peaks_stab, peaks_s4[i]+del_peaks_stab, alpha=0.4,␣
 ↪color="mistyrose")

plt.legend(loc=9, ncol=2, bbox_to_anchor=(0.5, -.1))
plt.xlim(0,1200)
plt.ylabel("Amplitude [mV]")
plt.xlabel("Channel")
plt.grid()
if saving == True:
    plt.savefig(savepath + r"\stab_4.pdf", dpi=150, bbox_inches="tight")
plt.show()
```

[ ]:
```
difference_s1, difference_s2, difference_s3, difference_s4
```

[ ]:
```
np.sqrt(2)*3
```

8 Beat Note

[ ]:
```
set_freq = [0.1, 0.2, 0.3, 3, 16, 186, 240, 109, 69.420, 0.01]
```

[ ]:
```
del_peak_beat = 0.01
```

[ ]:
```
files = [r"\AOM_BeatNote\beat01.csv", r"\AOM_BeatNote\beat02.csv", r"\AOM_BeatNote\beat03.
 ↪csv", r"\AOM_BeatNote\beat04.csv", r"\AOM_BeatNote\beat05.csv", r"\AOM_BeatNote\beat06.
 ↪csv", r"\AOM_BeatNote\beat07.csv", r"\AOM_BeatNote\beat08.csv", r"\AOM_BeatNote\beat09.
 ↪csv", r"\AOM_BeatNote\beat10.csv"]

beat_freq = []

for i in range(len(files)):
    beats1 = path + files[i]
    df_beat01 = pd.read_csv(beats1, header=None, sep=";", skiprows=1, names=["frequency",␣
 ↪"PD"], index_col=False)
    df_beat01["PD"] = df_beat01["PD"] * 1000 #transform to mV
    peak_beat01, _ = find_peaks(df_beat01["PD"], height=0.0035, distance=50)
    beat_freq.append(df_beat01["frequency"][peak_beat01[0]])

    plt.plot(df_beat01["frequency"],df_beat01["PD"], label=f"Beat note measurement")
    plt.axvline(df_beat01["frequency"][peak_beat01[0]], color="red", label="Estimated peak␣
 ↪position with uncertainties")
    plt.axvline(df_beat01["frequency"][peak_beat01[0]]-del_peak_beat, linestyle="dashed",␣
 ↪color="red")
    plt.axvline(df_beat01["frequency"][peak_beat01[0]]+del_peak_beat, linestyle="dashed",␣
 ↪color="red")
    plt.axvspan(df_beat01["frequency"][peak_beat01[0]]-del_peak_beat,␣
 ↪df_beat01["frequency"][peak_beat01[0]]+del_peak_beat, alpha=0.4, color="mistyrose")
    plt.xlim(df_beat01.frequency[0],df_beat01.frequency.iloc[-1])
    plt.ylabel("Amplitude [mV]")
    plt.xlabel("Frequency difference between the AOMs [KHz]")
    plt.grid()
    plt.legend()
    if saving == True:
```

```
        plt.savefig(savepath + r"\beat_" + f"{i}" + r".pdf", dpi=150, bbox_inches="tight")
    plt.show()
```

[ ]:
```
beat_freq
```

[ ]:
```
line_fit, line_fitcov = curve_fit(gerade, set_freq, beat_freq, sigma=[0.01]*len(set_freq),␣
 ↪absolute_sigma=True)
x_values_line = np.linspace(-10, 260, 1000)
fit_line = gerade(x_values_line, *line_fit)
print(line_fit, np.diag(np.sqrt(line_fitcov)))

plt.plot(x_values_line, fit_line, label="Linear fit")
plt.errorbar(set_freq, beat_freq, yerr=[0.01]*len(set_freq), marker="x", markersize=12,␣
 ↪capsize=8, linestyle="None", label="Beat note measurement")
plt.xlabel("Frequency difference between the AOMs [KHz]")
plt.ylabel("Beat note frequency [KHz]")
plt.grid()
plt.xlim(-10,260)
plt.legend()

if saving == True:
    plt.savefig(savepath + r"\beat_line.pdf", dpi=150, bbox_inches="tight")
plt.show()
```

8 Attachment C

8.1 Lab notes

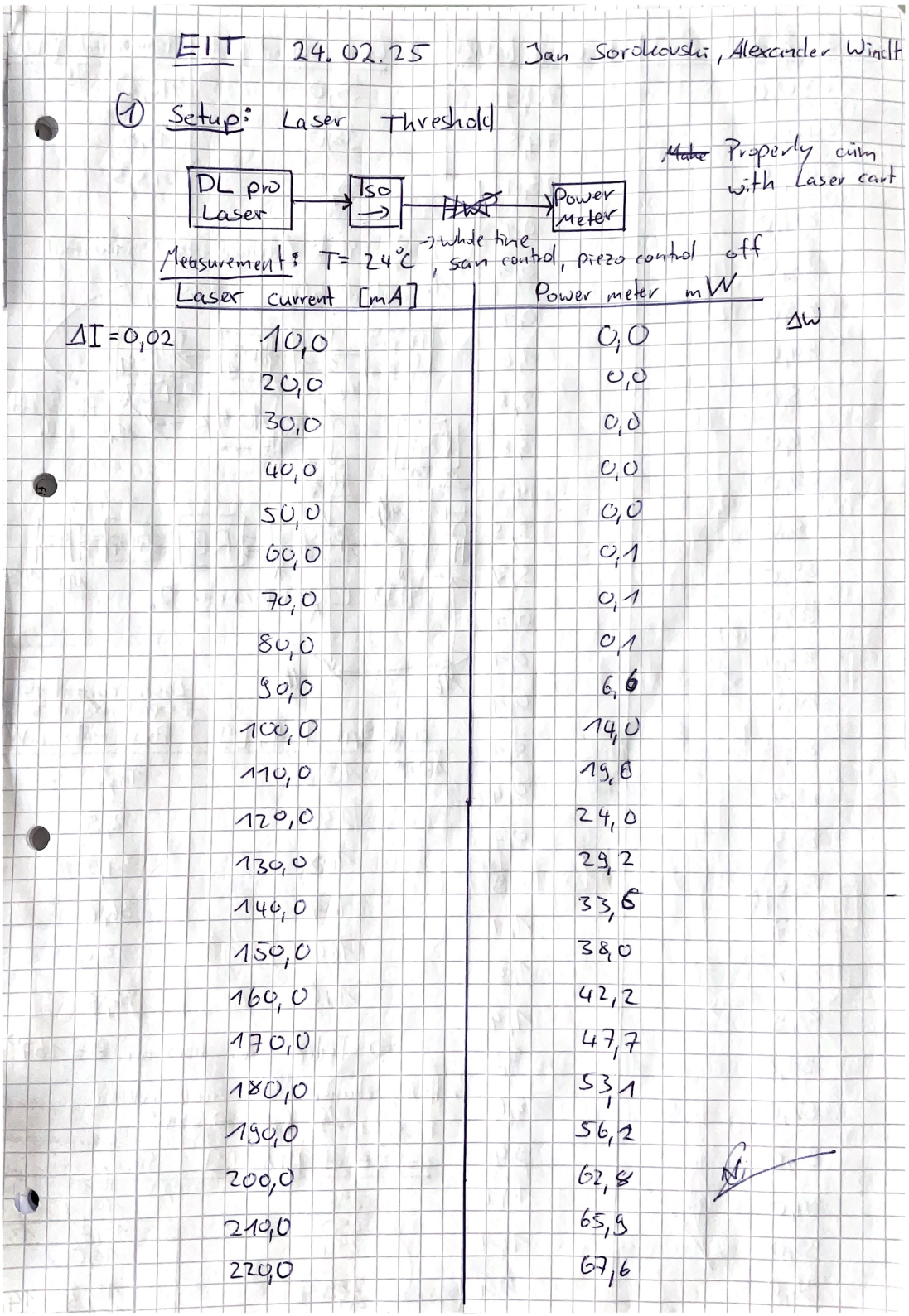


Fig. 53: Lab notes, page 1

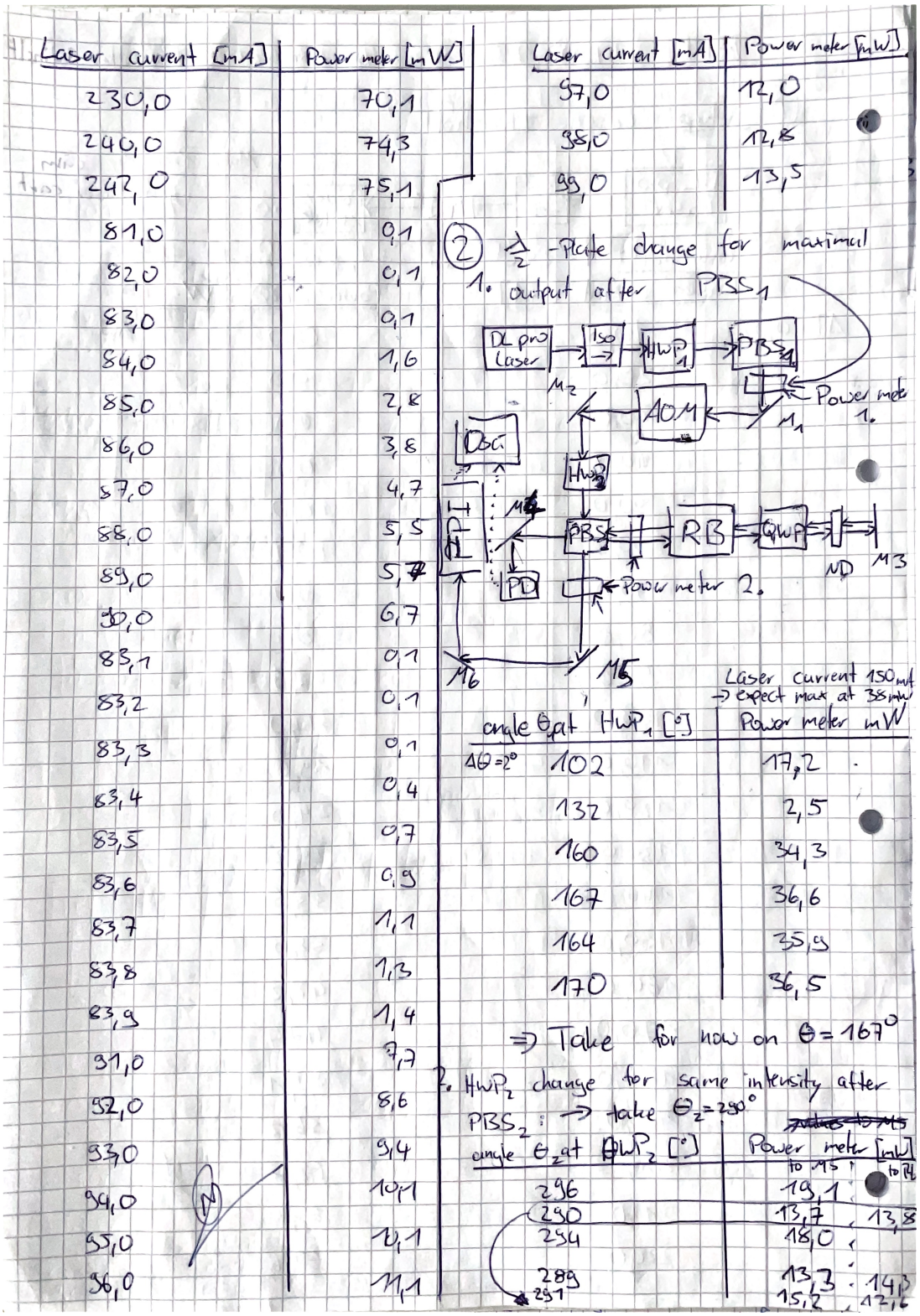
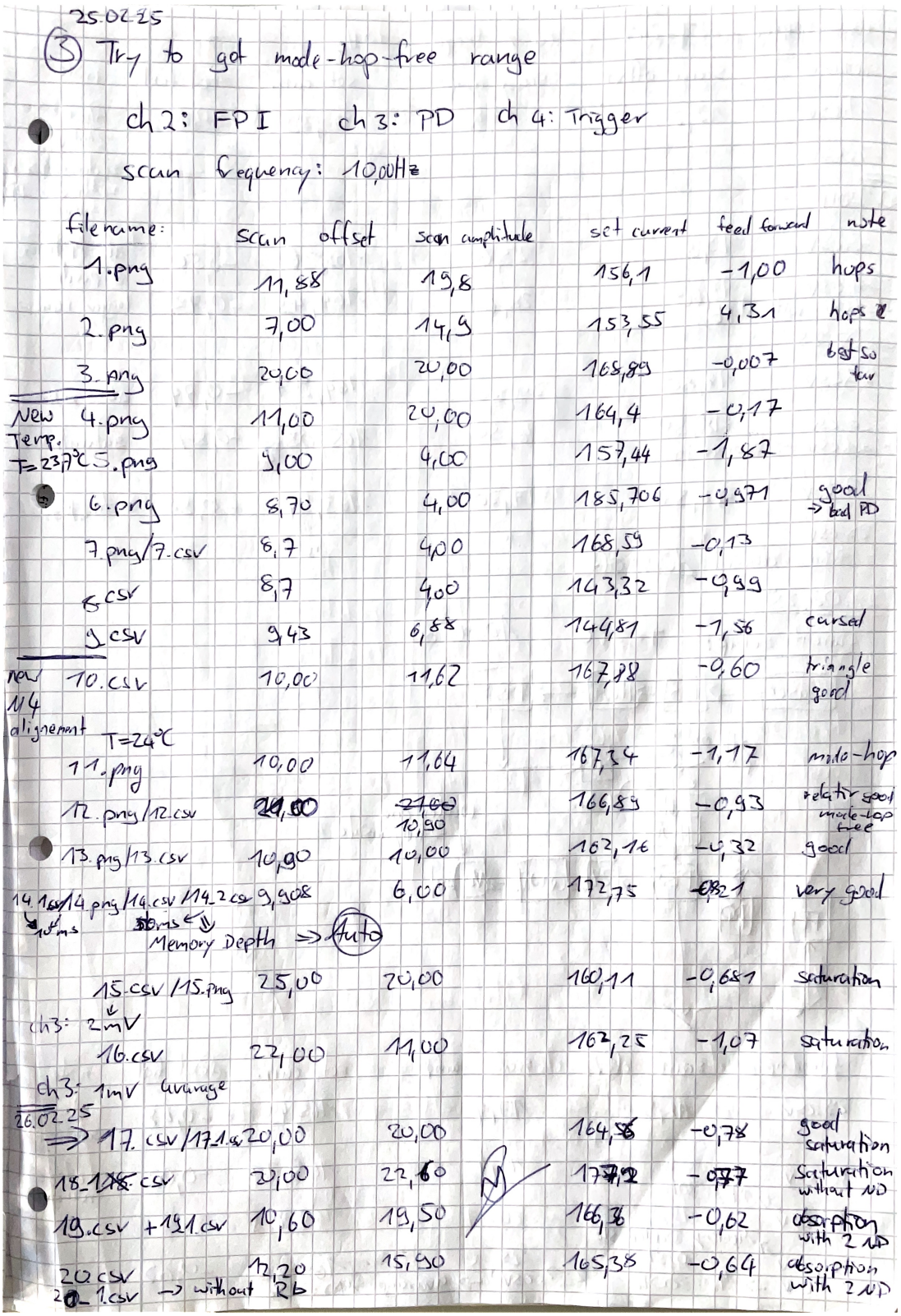


Fig. 54: Lab notes, page 2



25.02.25

③ Try to get mode-hop-free range

ch 2: FPI ch 3: PD ch 4: Trigger

scan frequency: 10,00Hz

filename:	scan offset	scan amplitude	set current	feed forward	note
1.png	11,88	19,8	156,1	-1,00	hops
2.png	7,00	14,9	153,55	4,31	hops
3.png	20,00	20,00	168,89	-0,007	bad so far
New Temp. T=23,7°C 4.png	11,00	20,00	164,4	-0,17	
5.png	9,00	4,00	157,44	-1,87	
6.png	8,70	4,00	185,706	-0,971	good -> bad PD
7.png/7.csv	8,7	4,00	168,59	-0,13	
8.csv	8,7	4,00	143,32	-0,99	
9.csv	9,43	6,88	144,81	-1,56	cursed
new 10.csv	10,00	11,62	167,88	-0,60	triangle good
N4 alignment T=24°C					
11.png	10,00	11,64	167,34	-1,17	mode-hop
12.png/12.csv	~~20,00~~	~~21,00~~ 10,90	166,89	-0,93	relativ good mode-hop free
13.png/13.csv	10,90	10,00	162,16	-0,32	good
14.~~16~~/14.png/14.csv/14_2.csv	9,908	6,00	172,75	~~-0,82~~ -0,21	very good
~~10ms~~ ~~50ms~~ Memory Depth => Auto					
15.csv/15.png	25,00	20,00	160,11	-0,681	saturation
ch3: 2mV					
16.csv	22,00	11,00	162,25	-1,07	saturation
ch3: 1mV Average					
26.02.25					
=> 17.csv/17.1.csv	20,00	20,00	164,~~58~~	-0,78	good saturation
18.~~17~~.csv	20,00	22,60	177,~~12~~	-0,77	Saturation without RD
19.csv +19.1.csv	10,60	19,50	166,~~36~~	-0,62	absorption with 2 RD
20.csv	12,20	15,90	165,38	-0,64	absorption with 2 RD
20_1.csv -> without Rb					

Fig. 55: Lab notes, page 3

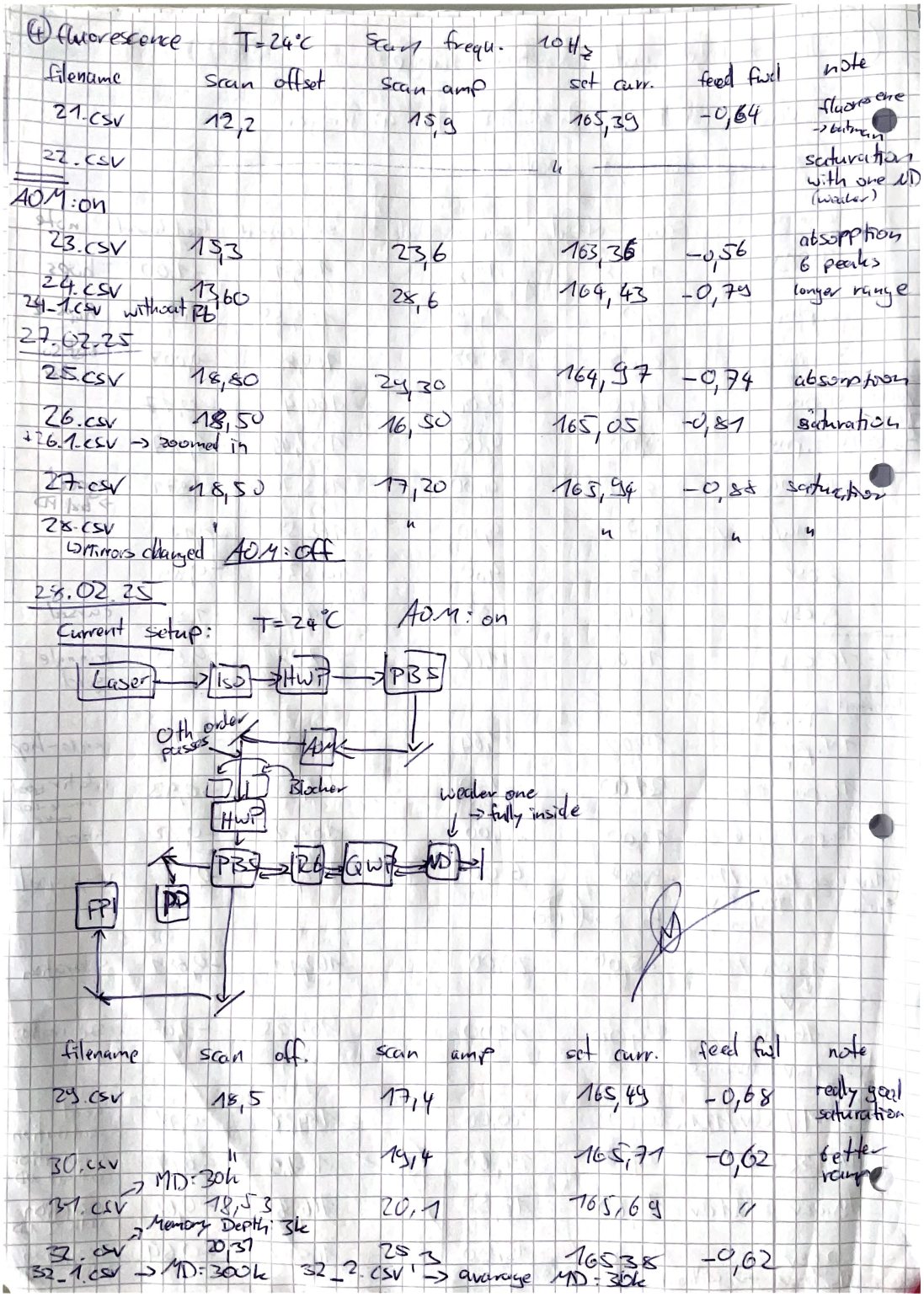


Fig. 56: Lab notes, page 4

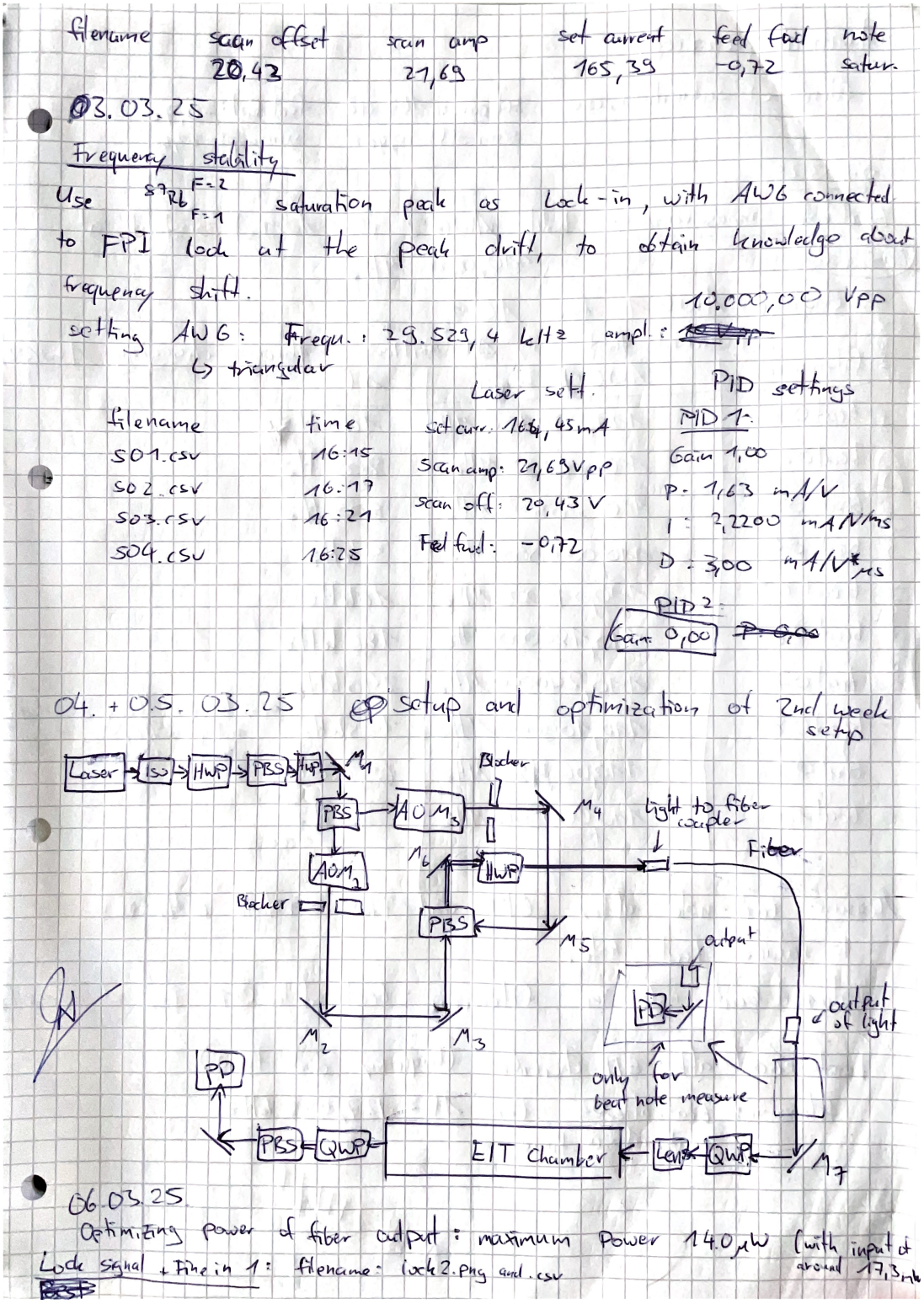
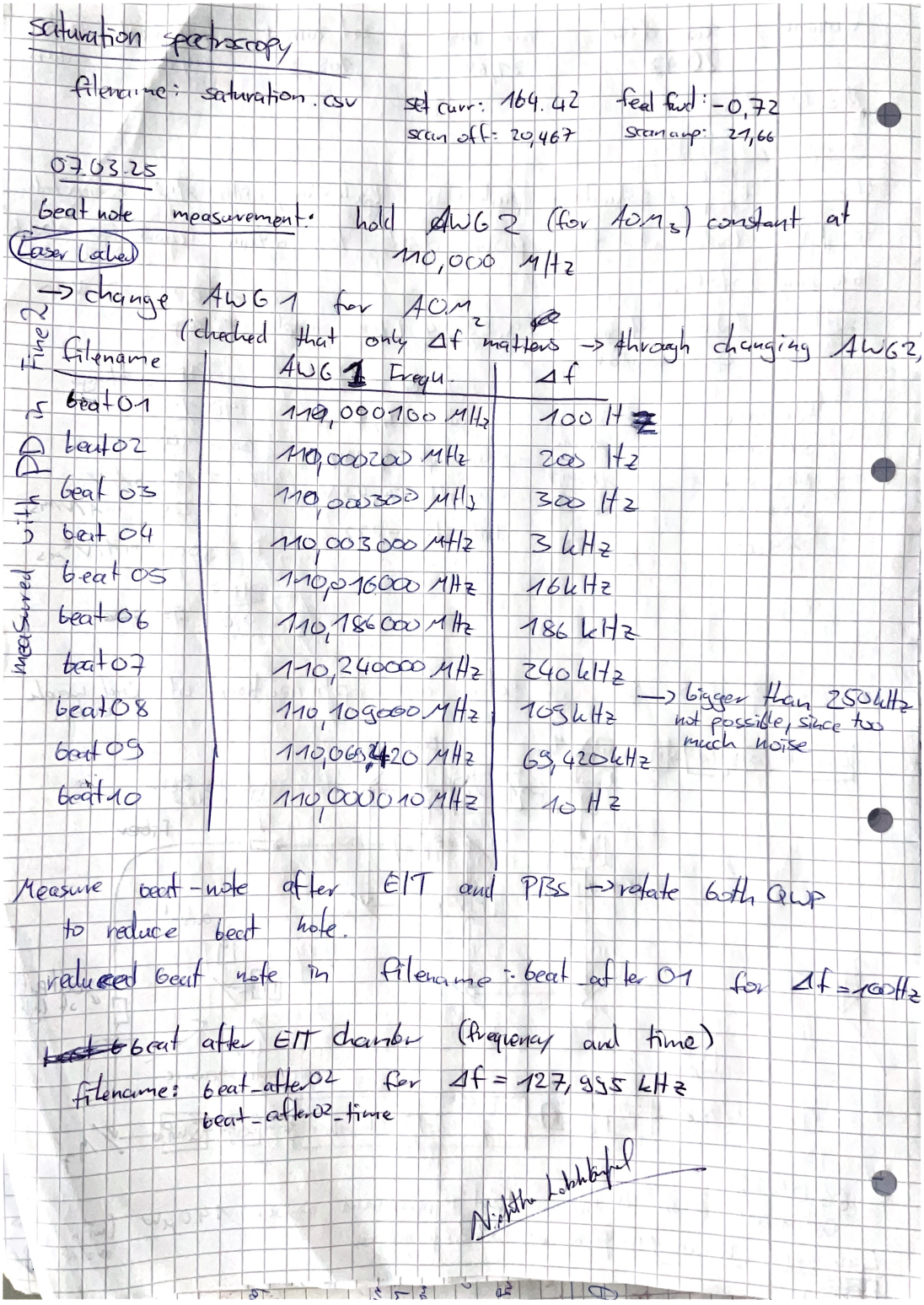


Fig. 57: Lab notes, page 5



Saturation spectroscopy

filename: saturation.csv set curr: 164.42 real curr: -0,72
scan off: 20,467 scan amp: 21,66

07.03.25

beat note measurement: hold AWG 2 (for AOM_2) constant at 110,000 MHz
(Laser locked)
→ change AWG 1 for AOM_2
(checked that only Δf matters → through changing AWG 2,

Fine 2

filename	AWG 1 Frequ.	Δf
beat01	110,000100 MHz	100 Hz
beat02	110,000200 MHz	200 Hz
beat 03	110,000300 MHz	300 Hz
beat 04	110,003000 MHz	3 kHz
beat 05	110,016000 MHz	16 kHz
beat 06	110,186000 MHz	186 kHz
beat07	110,240000 MHz	240 kHz
beat08	110,109000 MHz	109 kHz → bigger than 250 kHz not possible, since too much noise
beat09	110,069420 MHz	69,420 kHz
beat10	110,000010 MHz	10 Hz

measured with PD

Measure beat-note after EIT and PBS → rotate both QWP to reduce beat note.
reduced beat note in filename: beat_after 01 for Δf = 100 Hz

beat after EIT chamber (frequency and time)
filename: beat_after02 for Δf = 127,995 kHz
beat_after02_time

Nicht Lohnhaft

Fig. 58: Lab notes, page 6