Abstract

In current material research, spectroscopy is a widely used tool to study all kinds of material properties. One common spectroscopy technique is the so-called Raman spectroscopy, mainly used to identify and measure vibrational modes in molecules. It is based on an inelastic scattering process, using the excitations and de-excitations of different vibrational molecule modes. In this experiment, the basic principles of Raman spectroscopy are studied, and the versatile applications investigated, reaching from identifying vibrational modes over concentration measurements to temperature measurements. Additionally, the advantages and disadvantages of two different measuring devices are compared – a CCD-spectrometer and a monochromator.

After calibrations and studies of the optical elements of the setup, four liquid samples of CCl_4, $CHCl_3$, CH_2Cl_2 and CS_2 are examined and the fundamental vibrational frequencies are found. For the samples which can be compared to literature, a good accordance is found. The first sample shows depolarisations ranging from 0.07-2 depending on the peak and the measurement technique. For an unknown ethanol-water sample, the concentration of ethanol can be found by comparing several samples with different concentration. The concentration is found to be $C = (60 \pm 20)\,\%$, which is a realistic, but not accurate value. In a last step, the temperature of a sulphur sample is calculated to be $T = (273.0 \pm 0.4)\,\mathrm{K}$ for the measurements with the CCD-sensor, which is in the right order of magnitude, but not compatible with room temperature. For the monochromator, the temperature measurement does not provide meaningful results.

Contents

List of Tables

List of Figures

1 Introduction

Raman spectroscopy is a widely used spectroscopy technique of current scientific work to study properties of materials and detect vibrations in molecules. By using this method, a deep insight into the physical and chemical structure of molecules can be gained [1, 2]. Nowadays, due to the development of new technologies like lasers and CCD sensor, Raman spectroscopy is a cheap and fast method to study material properties. In addition, it is simple and easy to use [2]. Raman spectroscopy is based on the Raman effect, predicted by A. Smekal and discovered by C. V. Raman, which is an inelastic scattering process of light. It is explained by excitations and de-excitations of vibrational states in matter [2].

Raman spectroscopy has a wide range of applications. The goal of this experiment is to get familiar with the basics of Raman spectroscopy and study some of these applications in detail: In addition to the detection of vibrational modes in several molecules, it is used to study a sample of ethanol-water mixture with unknown ethanol concentration and to find out the temperature of a sulphur sample. Moreover, two different measuring devices for the spectra – a CCD camera and a monochromator – are compared.

2 Theory

The following theoretical introduction to the Raman effect is mainly based on the experiment instructions [1] as well as introductory literature to the field [2, 3].

2.1 Classical description of the Raman effect

In matter, electromagnetic radiation is not only characterized by the electric field $\vec{E}$, but rather by its interaction with the molecules and structures inside the material. On large scales, this interaction can be characterized by an induced dipole moment $\vec{P}$ and the polarisability of the material α:

$$\vec{P} = \alpha \vec{E} \quad \text{with} \quad \alpha = \begin{pmatrix} \alpha_{xx} & \alpha_{xy} & \alpha_{xz} \\ \alpha_{yx} & \alpha_{yy} & \alpha_{yz} \\ \alpha_{zx} & \alpha_{zy} & \alpha_{zz} \end{pmatrix}. \tag{1}$$

The polarisability is therefore not only a simple proportionality constant but rather a rank-two tensor with a geometrical representation by an ellipsoid. We can simplify the expression of α by performing a Taylor expansion to first order around the equilibrium coordinate r_0 of a molecule

$$\alpha(r) \approx \alpha(r_0) + \left.\frac{\partial \alpha}{\partial r}\right|_{r=r_0} (r - r_0) = \alpha_0 + \alpha_1 \cos(\omega_s t), \tag{2}$$

where the second equality follows from the assumption of a harmonic oscillation around equilibrium with frequency ω_s. Using monochromatic light

$$\vec{E} = \vec{E}_0 \cos(\omega_0 t) \tag{3}$$

to excite the material, the dipole moment takes the following form after simplification with trigonometric identities:

$$\vec{P}(t) = \alpha_0 \vec{E}_0 \cos(\omega_0 t) + \frac{1}{2}\alpha_1 \vec{E}_0 \cos((\omega_0 + \omega_s)t) + \frac{1}{2}\alpha_1 \vec{E}_0 \cos((\omega_0 - \omega_s)t). \tag{4}$$

The induced dipole moment therefore has three components that can be observed in the light emitted by the material. The first term corresponds to the frequency of the incoming electromagnetic field, which is called Rayleigh radiation. The two other terms have a frequency that is shifted by the vibrational frequency. The one with the lower frequency is hereby called Stokes and the other one Anti-Stokes radiation. In Figure 1, a sketch of the resulting spectrum can be found. One can clearly see the additional peaks due to the vibration of the molecule. The appearance of those additional peaks due to vibrations is called the Raman effect.

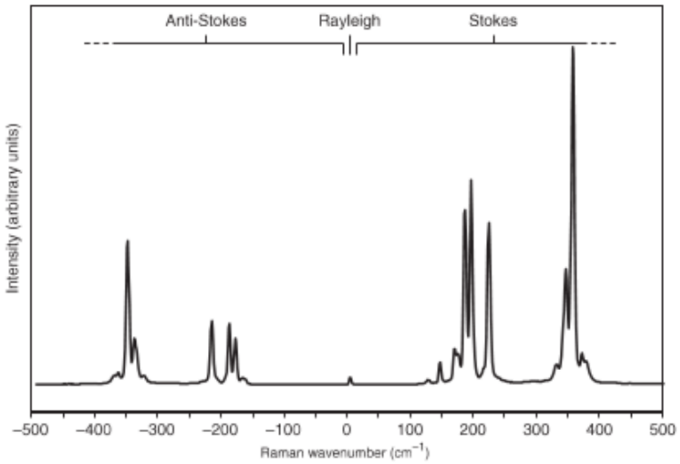


Fig. 1: Sketch of a Raman spectrum of As_4S_4: The intensity is plotted against the wave number difference to the Rayleigh peak. One can clearly identify Stokes and Anti-Stokes peaks. The Rayleigh peak is damped by a notch filter. The graphic is taken from [1], originally taken from [4].

A short comment should be added on the contrast to infrared (IR) spectroscopy, which is also used to identify vibrational modes in molecules: Here, a range of infrared frequencies is directly radiated onto the sample and the absorption is measured. An advantage of Raman spectroscopy is, therefore, that it is not necessary to match the frequencies of the vibration with the used laser. The two techniques are also able to find different vibrational modes. With IR spectroscopy one can detect modes, that change the dipole moment $\vec{P}$ of the molecule, whereas Raman spectroscopy is used to detect vibrations that change the polarisability α [2]. An example is shown in Figure 2.

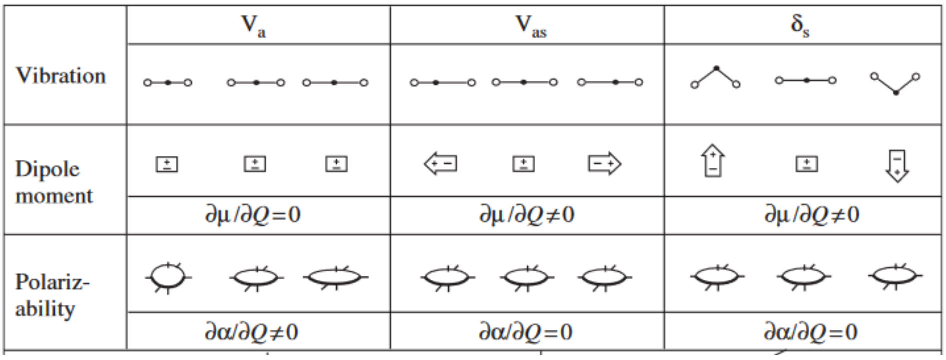


Fig. 2: In this graphic, different vibration modes of a three component molecule are shown. The first one only changes the polarisability and is detectable using Raman spectroscopy, and the other two modes change the dipole moment and need to be identified with IR spectroscopy. The graphic is taken from [2].

An interesting quantity, that can be used to gain more insight in the inner structure of a molecule, is the depolarisation ρ. Starting from the intensity of a dipole radiator

$$I = \frac{\omega^4}{32\pi^2\epsilon_0 c^3} P^2 = \frac{\omega^4}{32\pi^2\epsilon_0 c^3}(\alpha_{yz}^2 + \alpha_{zz}^2)E_z^2, \tag{5}$$

where the second part follows for an electric field polarized in z-direction and propagating in y-direction and a detector in x-direction, one can define the depolarisation as the fraction of intensity $I_{\|}$ parallel to the laser polarisation and $I_{\perp}$ vertical to the polarisation:

$$\rho_z = \frac{I_{\|}}{I_{\perp}} = \frac{3\gamma^2}{45\bar{\alpha}^2 + 4\gamma^2}. \tag{6}$$

The two quantities $\bar{\alpha}$ and γ characterise the ellipsoid and therefore the polarisability of the molecule and the corresponding geometry of the vibration [1].

2.2 Quantum treatment of the Raman effect

In addition to the position of the Stokes and Anti-Stokes peaks, one can also gain information from the intensity of the different peaks, which is not equal for the two peaks. To understand this, one needs to have a deeper look into the quantum treatment of the Raman effect. In analogy to electrical excitations of electrons in a molecule, the vibrational states of the molecule itself can be excited or de-excited. This happens via virtual states. The most probable outcome is an excitation and de-excitation with the same initial and final state. This is also referred to as Rayleigh scattering and corresponds to the most intense peak – the Rayleigh peak. If the vibrational mode in the final state differs from the initial one, we can observe inelastic scattering with an shifted outcoming frequency. This corresponds to the Stokes and Anti-Stokes radiation [2]. In Figure 3 a simple two-level example is provided.

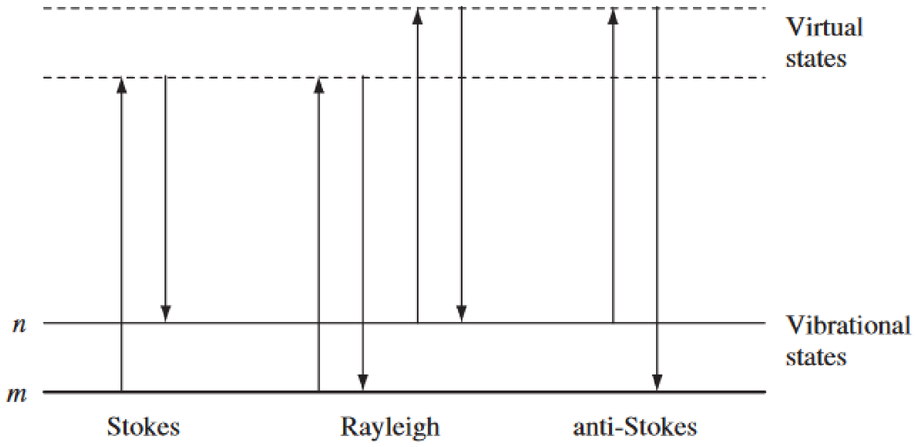


Fig. 3: In this graphic, a simple two-level model of Rayleigh, Stokes and Anti-Stokes radiation is shown. The states m and n are excited into virtual states. For the different de-excitations, the corresponding name is given. The graphic is takes from [2].

From thermodynamics, we know, that the excited vibrational states of a molecule are Boltzmann distributed:

$$N_e = N_0 \exp\left(\frac{h(\nu_0 - \nu_e)}{k_B T}\right). \tag{7}$$

Here, we use the Boltzmann constant k_B, the Plank constant h and the temperature T, as well as the energies $h\nu_0$ and $h\nu_e$. Due to the previous consideration, we find that the intensity of the

Stokes peak is proportional to the number of occupied ground states N_0 and the Anti-Stokes to the number of excited states N_e. We can therefore conclude:

$$\frac{I_{\text{Stokes}}}{I_{\text{Anti-Stokes}}} = \exp\left(\frac{h\Delta\nu}{k_B T}\right), \tag{8}$$

with frequency difference $\Delta\nu$. A more precise description can be found in [4] and yields:

$$\frac{I_{\text{Stokes}}}{I_{\text{Anti-Stokes}}} = \frac{(\nu_0 - \Delta\nu)^4}{(\nu_0 + \Delta\nu)^4} \exp\left(\frac{h\Delta\nu}{k_B T}\right). \tag{9}$$

This relation can be used to measure the temperature of a given material.

3 Methods

In the following paragraph, the basic methods and the setup of the experiment are explained with the different components used in the optical path. A special focus is set on the two different devices available to resolve the spectrum.

3.1 Setup

For the experiment, an optical path as sketched in Figure 4 is used. Before the measurements, the lenses and mirrors are aligned using a white light source. A rough setup is performed using a piece of paper to check, that the beam is collimated, and the mirrors are hit in the centre. In addition, a fine-tuning is done using the corresponding measurement device and optimizing the setup iteratively until the strongest signal is found.

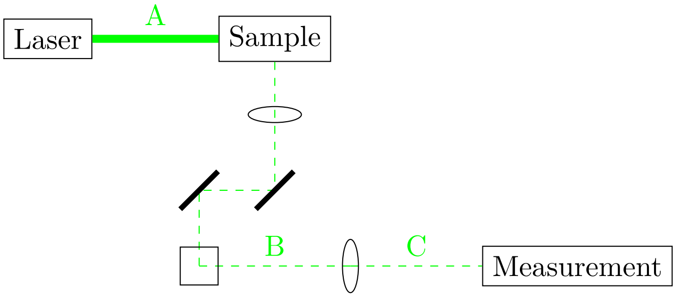


Fig. 4: Setup of the optical path with all the optical components. A laser is shot on a sample and the scattering light is passed to the spectral measurement device, which is either a CCD-sensor or a monochromator. At position A, a $\lambda/2$-plate can be added, at position B, there is a polarizer and at position C a notch filter can be added. In addition to the lenses and mirrors, a rectangle is added to the sketch corresponding to the device, that lifts the beam to the height of the measurement device.

In addition to the laser, which has a wavelength of 532 nm and can be powered with currents up to 1.5 A, there are two additional light sources: A Hg-lamp with a characteristic spectrum that can be used for calibration and a white light lamp to study the complete range of the spectrum. Both lamps can be brought into the optical path instead of the laser-sample setup.

In addition to a polarisation filter, a $\lambda/2$-plate can be added to the path. It can be used to rotate the polarisation axis of the laser around an arbitrary angle. In front of the measurement device, a notch filter can be installed. This filter is often used in Raman spectroscopy. It can filter out specific wavelengths of the spectrum and is therefore typically used to suppress the Rayleigh peak.

There are several samples that can be examined: In addition to several known liquid samples of CCl_4, $CHCl_3$, CH_2Cl_2 and CS_2, there is one unknown sample containing a ethanol-water mixture of unknown concentration. Samples of known ethanol concentration can be produced and measured as well. For the preparation of the samples, several injections are used and all samples are filled with a total of 10 mL. One solid sample of sulphur powder is also available.

The spectrum can be measured using two different devices, which are explained in more detail in the following sections: A digital CCD-sensor and a monochromator with an added photomultiplier.

3.1.1 CCD-Spectroscope

A CCD-Spectroscope (Charge Coupled Device) is a digital analysis tool for light spectra. Through a little hole, the light beam is reflected by a mirror onto a lattice, which decomposes the spectrum into the different wavelengths. The light is afterwards reflected onto a grid of semiconductor pixels. If a photon of a certain wavelength is contained in the spectrum, in the corresponding pixel an electron-whole pair is created and captured. After a certain integration time, the number of electron whole pairs in a certain pixel is counted and the information passes directly to an analysis software. The number of counts in the different pixels is directly proportional to the light intensity of the corresponding wavelength. In addition to this integration time, one can set a certain number n of averages in the software. The spectrum is then measured n times and the average can be exported.

An advantage of the CCD-spectroscope is, that in one measurement, the entire spectrum is recorded. It is a very fast device which produces first results in only a few seconds. For the optimization of the optical path, we therefore mainly use the CCD due to the fast feedback. For more accurate measurements, however, a higher integration time and number of averages can be used, increasing the measurement time. Disadvantages of the CCD spectroscope are the fixed number of pixels and therefore the fixed resolution, as well as defects in the semiconductor pixels.

3.1.2 Monochromator and photomultiplier

As an additional measurement device, a monochromator with connected photomultiplier is used. The basic principle is the same as for the CCD-spectrometer: The light enters through a small slit with variable slit width and is reflected on a lattice, that splits the light into the different wavelengths. This light is again reflected and passes a second slit, causing that only one specific wavelength can pass. Through a rotation of the lattice, this wavelength can be adjusted, and the whole spectrum can be scanned. A simple schematic sketch is provided in Figure 5.

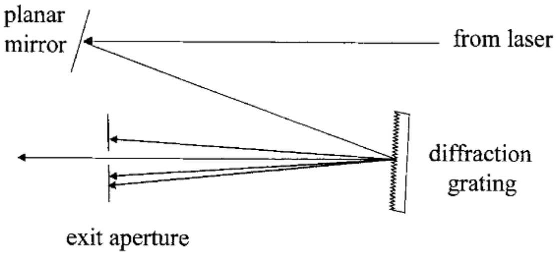


Fig. 5: In this picture, a simple schematic sketch of a monochromator is provided. A beam, e.g. coming from a laser, hits a planar mirror and afterwards a rotatable lattice. The light is then split into the spectrum. Due to an exit aperture, only light of a certain wavelength is passed. The figure is taken from [5].

The light of the selected wavelength is afterwards passed to a photomultiplier. In a cathode, electrons are released by the incoming photons. Using several dynodes, the signal is afterwards amplified. For that, the electrons are accelerated in an electric field and release additional electrons, resulting in an avalanche of electrons that can be measured as a signal. The number of signals is proportional to the light intensity hitting the cathode.

The signal from the photomultiplier is passed to a main amplifier and a single-channel analyser, which converts the signal into a logical one. The signal can be observed on an analogue ratemeter and through a digital converter it is also passed to the computer. For the measurement, a

LabView program is used. There, the initial and final wavelength as well as the scanning velocity can be set to adjust the measured data to the monochromator setup.

In comparison to the CCD-spectroscope, a higher spectral resolution can be measured using the monochromator, by adapting the scanning velocity. However, every spectrum needs a long measurement time due to the scanning through the different wavelengths, and it is not possible to get a quick impression of the spectrum. This increases the total measurement time drastically.

3.2 Experimental approach

As described above, in a first step the optical path is improved iteratively, first using a piece of paper to follow the light beam and afterwards using the CCD-spectrometer to detect small changes in intensity. Afterwards, the Hg-spectrum is used for calibrating the wavelengths and the white light is measured to investigate the effects of the polarisation, the notch filter and to find a lower wavelength bound for the monochromator. For this device, additional measurements have to be taken to investigate the effect of the slit width. The spectrum of the laser can only be measured with the CCD-spectrometer, since the high intensity could damage the photomultiplier.

After all the preliminary considerations, measurements of the different samples are performed. In a first step, the Raman spectra of the CCl_4, $CHCl_3$, CH_2Cl_2 and CS_2 samples are taken. The measurements are performed with the polarisation filter in 0°- and 90°-orientation and with and without a $\lambda/2$-plate in 45° position, corresponding to a rotation of the laser polarisation axis around 90°. The notch filter is not used, since some Stokes peaks were damped using this filter. The goal of this part is to identify the Stokes- and Anti-Stokes peaks and for CCl_4 to study the intensity for different polarisations and thereby find the depolarisation.

A second measurement is done to determine the concentration of ethanol in the unknown sample. Therefore, 10 different ethanol-water samples are produced, reaching from an ethanol concentration of 0 % to 100 %. The measurements are performed using the notch filter, and a polarisation of 90°. By identifying the typical ethanol Stokes-lines, their intensity can be plotted against the ethanol concentration and, using a linear regression, the unknown sample can be evaluated.

In a last measurement, one can use the fact, that Stokes and Anti-Stokes lines show different intensities depending on the temperature of the sample. This is used to find the temperature of the sulphur sample. For different laser intensities, the sulphur sample spectrum is measured in a 90° polarisation with the notch filter.

In general, most of the measurement parts can be performed with both, the CCD-spectrometer and the monochromator. However, due to limited measurement time in a one-week experiment, all the experimental parts were first performed using the CCD-spectrometer. Afterwards, a selected number of measurements was again performed using the monochromator, which has a higher resolution and can be used to get a more detailed spectrum of the selected sample.

4 Analysis and results

4.1 Calibration and investigation of the devices

Before starting the Raman spectroscopy, it is necessary to calibrate the CCD spectrometer as it measures intensities in dependence of bins. The calibration allows an assignment to wavelengths λ. In the case of the monochromator the number of counts are measured in dependence of a wavelength λ_m in Å, but as it might have an offset we also perform a calibration here. Moreover, the devices have different detection effects which have to be investigated.

4.1.1 Calibration

For the calibration of the devices, a Hg-lamp is used as its emission spectrum is known. First, we discuss the calibration of the CCD spectrometer. With the setup in Figure 4 and the Hg lamp instead of the laser and the sample, the intensity I in each channel c is measured twice – one time with the lamp and one time without the lamp as a background measurement. The polarisation filter is used in 0° orientation and no $\lambda/2$ plate is inserted. For each measurement $n = 1000$ values are taken, and we calculate the mean. The uncertainty is therefore estimated by the standard deviation of the mean $\frac{\sigma}{\sqrt{n}}$ with σ being the standard deviation of the 1000 measurements. As the values of single measurements are not known, the standard deviation is estimated by the standard deviation of a Poissonian distribution, which is $\sigma = \sqrt{I}$. One measurement is taken for an integration time of 40 ms. The intensity for the background measurement I_b is subtracted from the measurement with lamp I_0:

$$I = I_0 - I_b, \tag{10}$$

$$\Delta I_{0/b} = \sqrt{\frac{I_{0/b}}{n}}, \tag{11}$$

$$\Delta I = \sqrt{\Delta I_0^2 + \Delta I_b^2}. \tag{12}$$

In the following, all data is corrected with the corresponding background before other evaluations are done. The uncertainty on the channels is estimated by a rectangular distribution and therefore is $\frac{1}{\sqrt{3}}$. The measured Hg spectrum can be found in Figure 6.

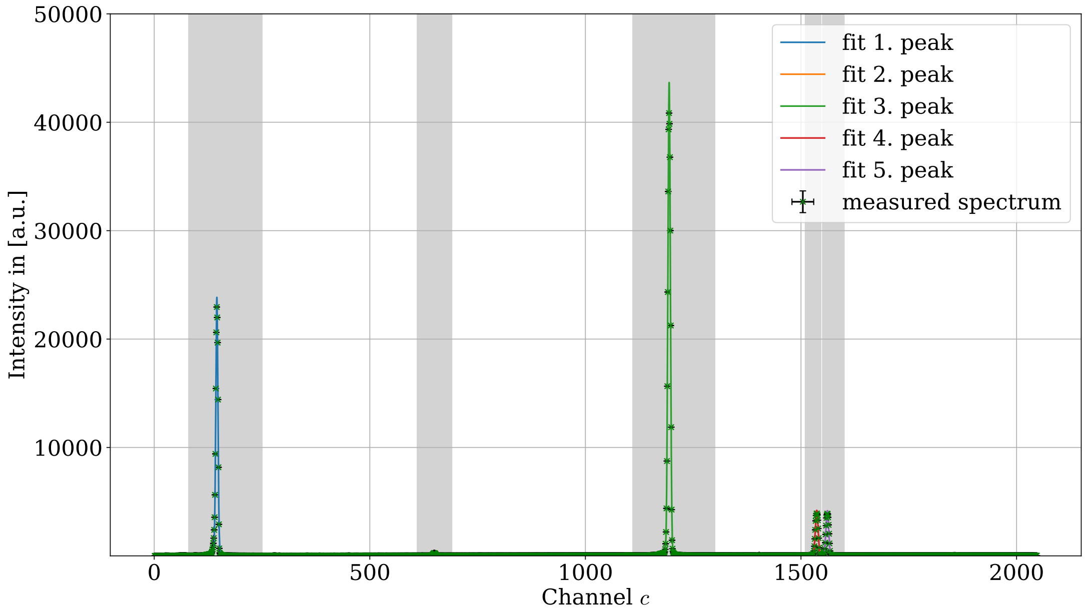


Fig. 6: The plot shows the intensity I of the Hg spectrum plotted against channels c. The uncertainties and Voigt fits of the peaks are shown. For the fits, only data in the grey areas is taken.

In the spectrum, five emission peaks can be identified. To find the peak positions, fits are performed. As a fitting function, a Voigt profile is used, which describes optical emission peaks [6]. The Voigt function is a convolution of a Gaussian and a Lorentz function. A Gaussian function describes a peak shape which is dominated by thermal Doppler broadening, while a Lorentz function originates from the lifetime of the state and the perturbation of states because of the influence of other atoms. Both effects may play a role, and the Voigt function accounts for both. The Voigt function is defined as follows:

$$g(x;\mu,\sigma) = \frac{1}{\sqrt{2\pi}\sigma} e^{-\frac{(x-\mu)^2}{2\sigma^2}}, \tag{13}$$

$$l(x;\mu,\gamma) = \frac{1}{\pi}\frac{\gamma}{(x-\mu)^2+\gamma^2}, \tag{14}$$

$$v(x;A,\mu,\sigma,\gamma,y_0) = A\cdot(g*l)+y_0, \tag{15}$$

where A is a scaling parameter, μ the peak position, σ and γ width parameters and y_0 an offset. The fits are performed using `scipy.optimize.curve_fit` [7] and are plotted in Figure 6. The peak positions and the corresponding fitting uncertainties are summarized in Table 1. There, one also finds the fitting parameters for σ and γ. To get a measure of the quality of the fit, we use the reduced χ^2-value with the number of data points N, the measured values x_{exp}, the corresponding model values x_{mod} and the uncertainty Δx_{exp}:

$$\chi_0^2 = \sum_N \frac{(x_{\text{mod}}-x_{\text{exp}})^2}{\Delta x_{\text{exp}}^2}, \tag{16}$$

$$\chi^2 = \frac{\chi_0^2}{N-f}, \tag{17}$$

where f is the number of degrees of freedom. For a value of $\chi^2 \approx 1$, we can assume the model to be appropriate. Those values can also be found in the table.

Tab. 1: The table contains the fit parameters μ, σ and γ of the Hg spectrum measured with the CCD spectrometer. Moreover, the reduced χ^2 values of the fits and the literature values of the Hg emission peaks [8] are shown.

Peak	μ	σ	γ	χ^2	Hg wavelength
1.	145 ± 5	2.2734 ± 0.0006	0.3675 ± 0.0007	12031.9	435.833 nm
2.	649 ± 7	2.80 ± 0.04	0 ± 0.07	6.6	491.607 nm
3.	1194 ± 7	2.8871 ± 0.0003	0 ± 0.0005	10905.0	546.074 nm
4.	1536 ± 7	3.136 ± 0.004	0 ± 0.007	2679.0	576.960 nm
5.	1561 ± 7	3.134 ± 0.003	0 ± 0.006	1655.5	579.066 nm

It is conspicuous that for most peaks γ is much smaller than σ and the Voigt function is therefore in good approximation a Gaussian function. Moreover, the χ^2 values for the fit are very large. Possible reasons for this will be listed in the discussion.

Now, the emission wavelength of Hg, which can be found in Table 1, is plotted against the peak positions. A linear fit is performed using `scipy.optimize.curve_fit` [7]. The fitting function and the fitted values are:

$$\lambda = a \cdot c + b \tag{18}$$

$$a = (0.101 \pm 0.002)\,\mathrm{nm}, \tag{19}$$

$$b = (424 \pm 3)\,\mathrm{nm}, \tag{20}$$

$$Cov(a, b) = -0.005\,\mathrm{nm}^2. \tag{21}$$

The corresponding plot can be found in Figure 7.

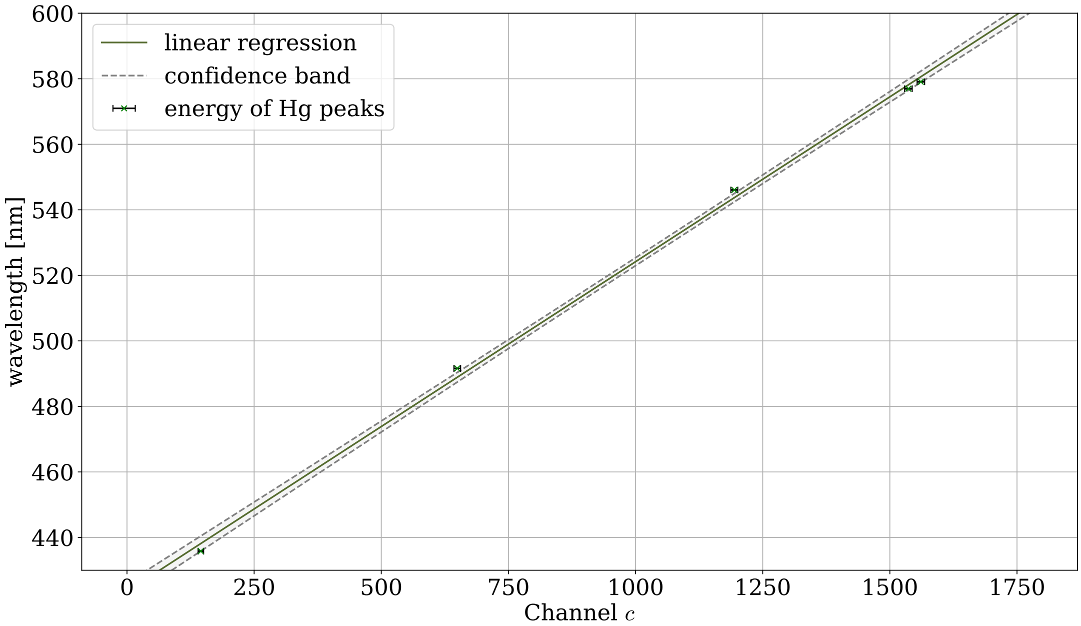


Fig. 7: In the graphic, the calibration of the CCD-spectrometer can be found, with the known wavelength λ of the peaks plotted against the channel number c on the x-axis. The corresponding linear regression with confidence band is also added.

The wavelength λ and its uncertainty are in the following calculated with the fitting parameters using the following formula:

$$\lambda = a \cdot c + b, \tag{22}$$

$$\Delta\lambda = \sqrt{(a\Delta c)^2 + (c\Delta a)^2 + (\Delta b)^2 + 2c \cdot Cov(a,b)}. \tag{23}$$

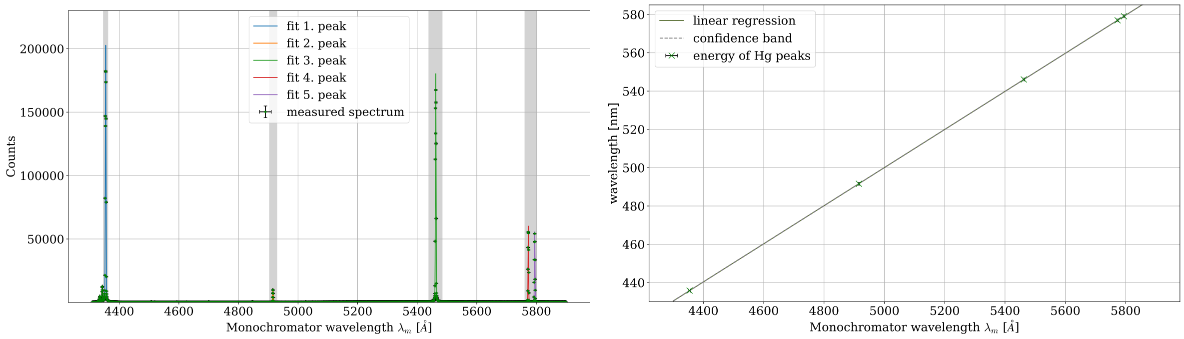
The monochromator is calibrated completely analogous. It measures with a scan rate of 1 Å/s and starts at a wavelength of 4300 Å. The counts N are corrected with the underground, as for the CCD spectrometer:

$$N = N_0 - N_b, \tag{24}$$

$$\Delta N_{0/b} = \sqrt{N_{0/b}}, \tag{25}$$

$$\Delta N = \sqrt{\Delta N_0^2 + \Delta N_b^2}. \tag{26}$$

The so obtained spectrum can be seen in Figure 8a. Again, a Voigt function is fitted to the measured values. The fits can be seen in the figure too.



(a) Hg spectrum and fits

(b) Calibration

Fig. 8: The left plot contains the measured counts N against the monochromator wavelength λ_m of the Hg lamp, and the Voigt fits to the emission lines. The right graphic shows the emission wvalength λ against the fitted peak wavelength of the monochromator λ_m. Moreover, the linear fit and its confidence bands are shown.

The fitting parameters are summarized in Table 2. It is again remarkable that the values for σ are clearly larger than the values for γ. Also, the reduced χ^2 values are high, which will be part of the discussion. However, the wavelength λ of the Hg emission lines are plotted against the fitted peak positions. A linear fit can be found in Figure 8b and provides values for a and b:

$$a = (0.099\,47 \pm 0.000\,04)\,\text{nm/Å}, \tag{27}$$

$$b = (2.70 \pm 0.19)\,\text{nm}, \tag{28}$$

$$Cov(a,b) = -0.000\,006\,\text{nm}^2/\text{Å}. \tag{29}$$

With these parameters and Equation 23 the monochromator wavelengths are in the following directly transformed into wavelengths.

Tab. 2: The table contains the fit parameters μ, σ and γ of the Hg spectrum measured with the monochromator. Moreover, the reduced χ^2 values of the fits and the literature values of the Hg emission peaks [8] are shown.

Peak	μ	σ	γ	χ^2	Hg wavelength
1.	(4354 ± 3) Å	(1.112 ± 0.002) Å	(0.032 ± 0.003) Å	1854.2	435.833 nm
2.	(4916 ± 2) Å	(0.838 ± 0.006) Å	(0.006 ± 0.007) Å	7.2	491.607 nm
3.	(5462 ± 2) Å	(0.9765 ± 0.0016) Å	(0.1692 ± 0.0016) Å	575.4	546.074 nm
4.	(5773 ± 2) Å	(0.851 ± 0.002) Å	(0.002 ± 0.003) Å	67.9	576.960 nm
5.	(5795 ± 2) Å	(0.926 ± 0.003) Å	(0.058 ± 0.003) Å	75.5	579.066 nm

4.1.2 Investigations using the monochromator

Using the monochromator, before starting with the sample measurements, an investigation of the effect of the slit width needs to be done, to choose a sensible value for further measurements. Therefore, an intense peak of the Hg-spectrum at approximately 546 nm is selected, and a measurement is performed for different slit widths d. In Figure 9 the resulting intensity curves can be found.

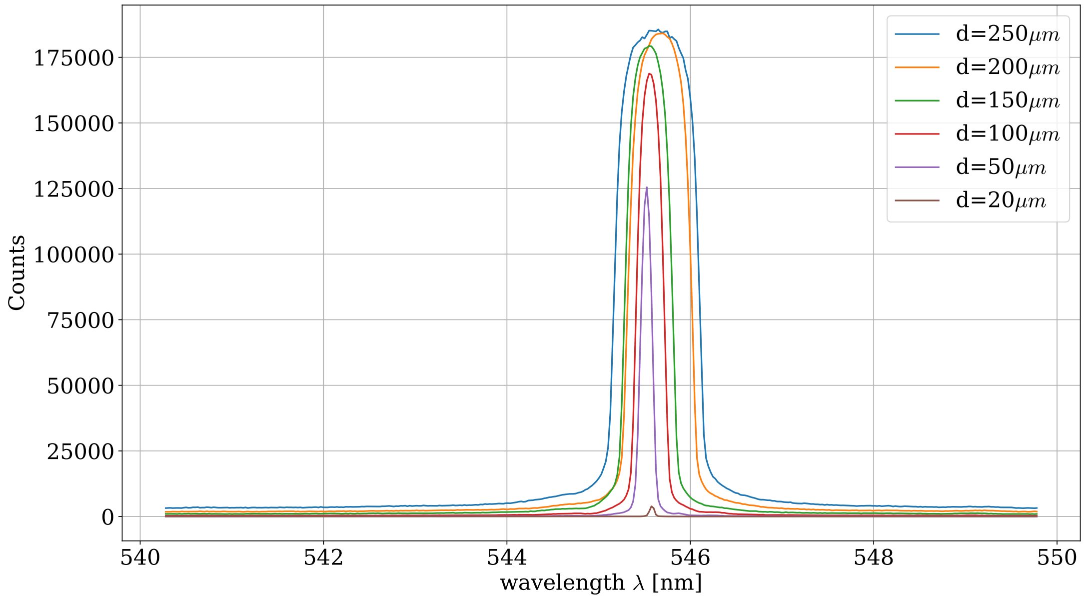


Fig. 9: In the plot, the intensity of the 546 nm Hg-peak in counts is plotted against the wavelength λ in nm. Several slit widths d reaching from 20 µm to 250 µm are portrayed in different colours.

In general, it is important to find a compromise between intensity and optical scattering effects. However, under a slit width of 100 µm the damping is too high to get a good signal. We therefore use slit widths of at least 100 µm depending on the expected intensity of the spectrum for all the following measurements.

A short comment needs to be done on two peculiarities: First, we can see in Figure 9 that not all the peaks have their maximum at the same position. In addition, all the peaks are shifted in comparison to the literature value and also with respect to the calibration. Both effects are due to irregularities in the movement of the lattice and a detailed discussion on this is added in the corresponding part of the report.

In addition to considerations about the slit width, the lower wavelength bound of the monochromator needs to be detected. This is an effect coming from the photomultiplier: In the cathode, a certain energy is needed to extract electrons. For high wavelengths, it is possible, that the energy is not high enough to extract any photons. We therefore expect an exponential intensity decay for high wavelengths. To determine this, the white lamp is used and a large range of the visible spectrum is measured. The corresponding plot can be found in Figure 10.

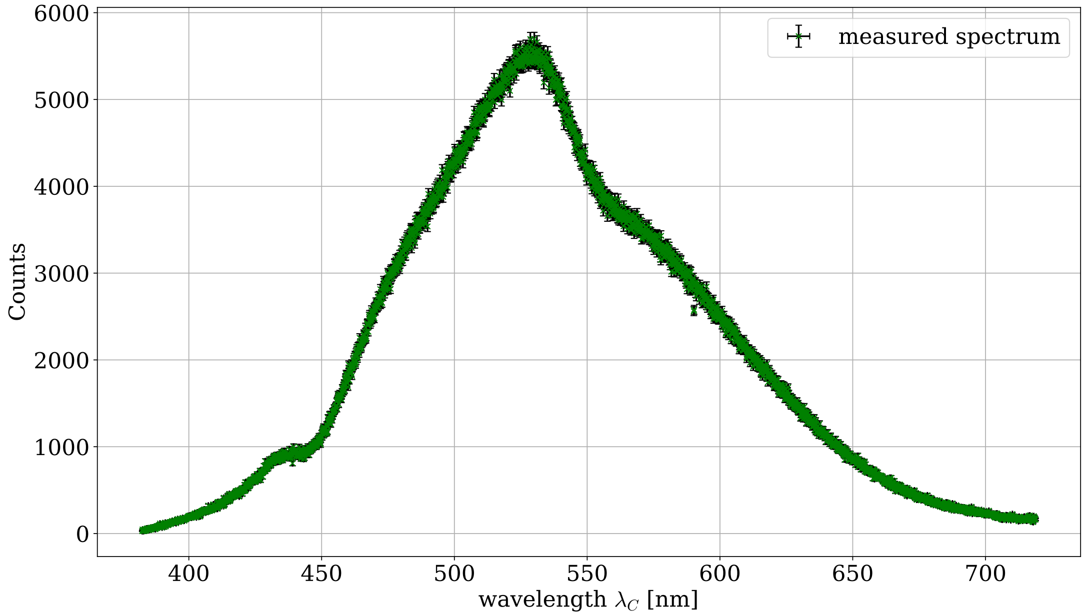
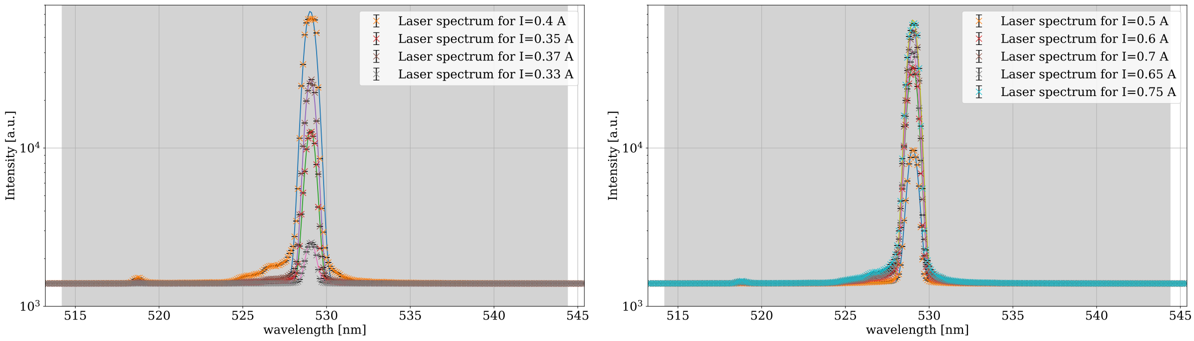


Fig. 10: A white light spectrum is measured with the monochromator in a large wavelength range. The number of counts is plotted against the wavelength λ in nm. This spectrum is used to observe the lower wavelength bound of the monochromator.

The measured spectrum has the expected shape: Up to a wavelength of approximately 530 nm we observe an increase in intensity due to the increase in intensity of the white light lamp. Afterwards, the intensity drops and has an exponential decaying tail starting at a wavelength of approximately 600 nm. Since all the Raman spectra are not in this wavelength area, this should not affect the following measurements.

4.1.3 Determination of the laser wavelength with CCD

In the following, the laser wavelength can be determined using the CCD spectroscope. This measurement is not possible using the monochromator, due to the risk of damaging the photomultiplier. Several measurements are performed for 0° and 90° polarisations and different laser intensities by varying the applied laser current. Then, a Voigt function is fitted to all the peaks to determine the peak position and therefore the laser wavelength. The corresponding plots are presented logarithmically in Figure 11.



(a) Laserpeak with 0° polarisation (b) Laserpeak with 90° polarisation

Fig. 11: In the two graphics, the measurements of the laser peaks with different laser intensities are shown for both light polarisations. In both cases, the intensity in arbitrary units is plotted against the wavelength in nm. A logarithmic scale is chosen to ensure, that all peaks are visible. To all the peaks, a Voigt function is fitted, with the fitting range marked in grey.

To determine the wavelength of the laser, for every peak, the mean value μ is taken. Then the values are averaged and to determine the uncertainty, the standard deviation of the mean is calculated. By doing so, we find a laser wavelength of (529.035 ± 0.005) nm which is in the range of the uncertainty not comparable with the value of 532 nm given in the lab. Calculating a t-value, which corresponds to the deviation from literature divided by the uncertainty, we find $t = 593$. Since a t-value of less than 2 corresponds to good agreement, this shows that we have a large deviation. This deviation may come from a systematic error from the calibration uncertainty. If we add the x-errors in the plot of a laser peak as shown in Figure 12, we find that due to the calibration we have significant systematic errors. Those errors are however, not included in the uncertainty calculated from the standard deviation of the mean. This point is discussed in detail in the last chapter of the report.

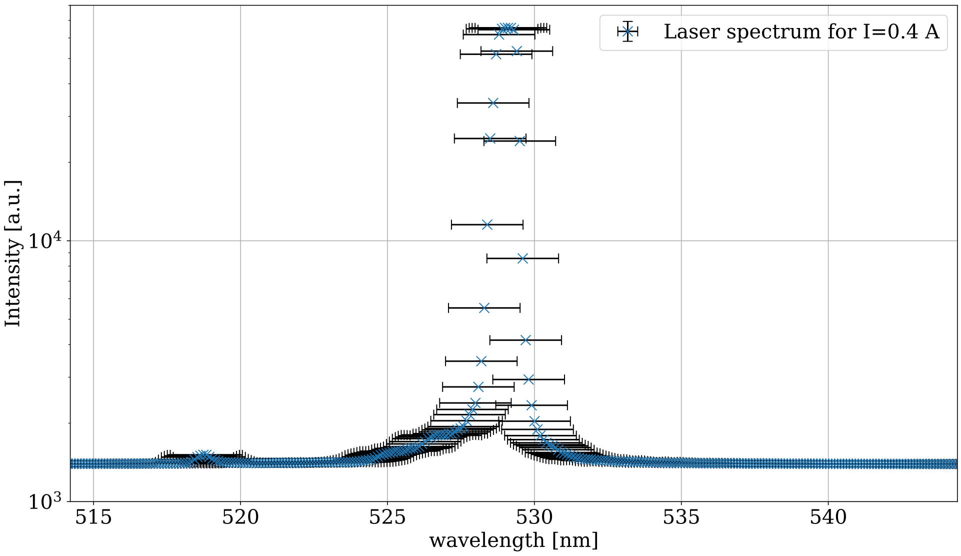
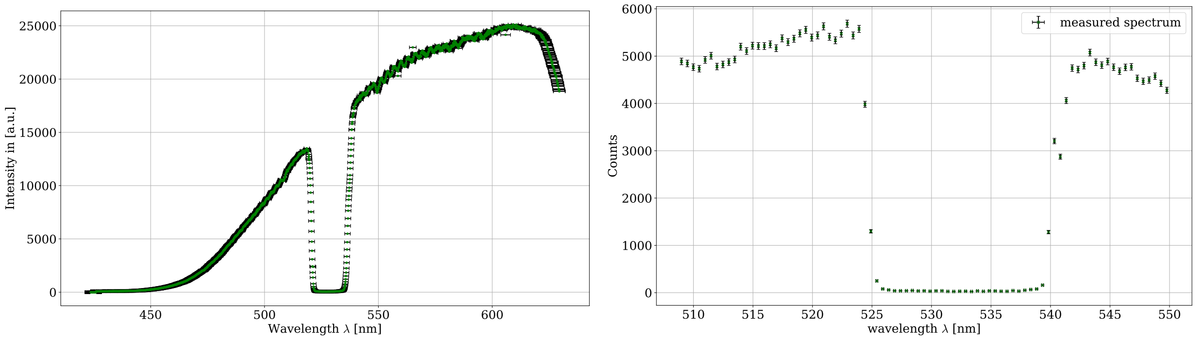


Fig. 12: In the graphic, one example of a laser peak is shown. Again, the intensity in arbitrary units is plotted against the wavelength in nm. In addition, the x-errors coming from the calibration uncertainty are added, corresponding to a systematic error.

An interesting observation from the measured spectra is, that for a polarisation of 0°, significantly lower currents are needed to get comparable laser intensities. This indicates, that the laser polarisation is a lot closer to 0° than to 90°. However, we did not measure the exact laser polarisation using the $\lambda/2$-plate, so in the following, the laser is assumed to be in 0° polarisation for some analysis parts.

4.1.4 Notch filter, polarizer and $\lambda/2$-plate

Before starting the Raman spectroscopy, some additional filters and optical elements are studied in more detail. First the effect of the notch filter is examined, by measuring a white light spectrum in the range around the laser wavelength with the notch filter using both, the monochromator and the CCD-device. The corresponding spectra are shown in Figure 13.

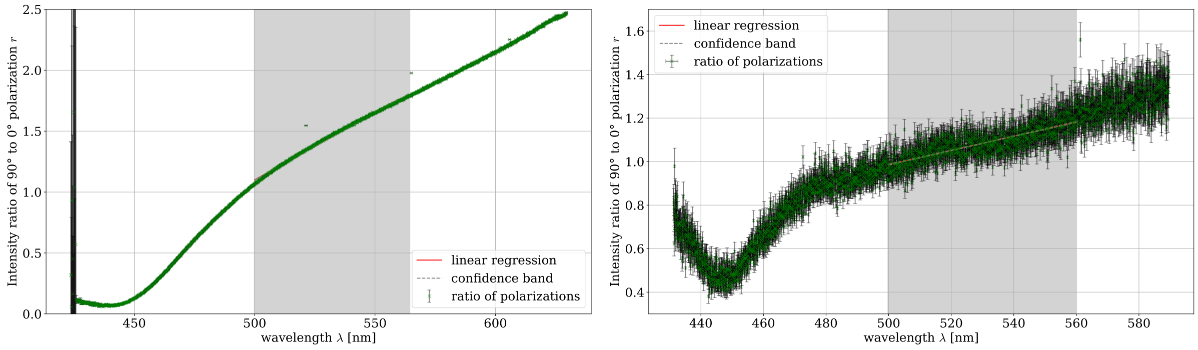


(a) Notch filter measured with CCD (b) Notch filter measured with monochromator

Fig. 13: In the two graphics, the measurements of the notch filter are presented, with the intensity in arbitrary units plotted against the wavelength in nm. One can see a clear damping in a given wavelength range.

As expected, the notch filter generates a strong damping in a wavelength range from approximately 525 nm to 540 nm, which is the wavelength range around the laser and therefore the Rayleigh wavelength.

Another effect, that needs to be studied, is the comparison of the detection of the two polarisation directions. It could be possible that one polarisation can be detected more easily than another one. To study this effect, for both sensors, measurements in both polarisations are taken with the white light source. Afterwards, the measured intensities are divided. If there is no difference between the polarisation, we would expect a flat profile with value 1. However, we detect the results presented in Figure 14.



(a) Detection of polarisations, CCD (b) Detection of polarisations, monochromator

Fig. 14: In the two graphics, the ratio between the intensities for 90° polarisation and 0° polarisation is presented. On the left the measurement with the CCD and on the right with the monochromator are shown. In addition, a linear fit is added in the grey marked range, which is later used for wavelengths in this range to find a correction of the intensities.

For both sensors we observe an increase in the intensity detection rate for the 90° polarisation with increasing wavelength. For wavelengths lower than approximately 500 nm, 0° light is higher in intensity and therefore is easier to detect than 90° light, whereas for wavelengths larger than 500 nm it is the other way round. Due to this fact, we later use mostly the 90° polarisation, because all later measurements are taken in the range, where this polarisation is of higher intensity.

For the later analysis, this ratio can be used as a correction factor on the measured intensities. We therefore fit a straight line in the range where later measurements are performed. The corresponding fit can be found in Figure 14 too. From this fit we get a correction factor for arbitrary wavelengths for comparison of intensities for different polarisations. This is however only used for some parts of the analysis, where the intensity plays a crucial role and a notice is given at the corresponding point. It is worth noticing, that the polarisation of the white light may also play a role in this measurement. This is however mentioned in the discussion part again.

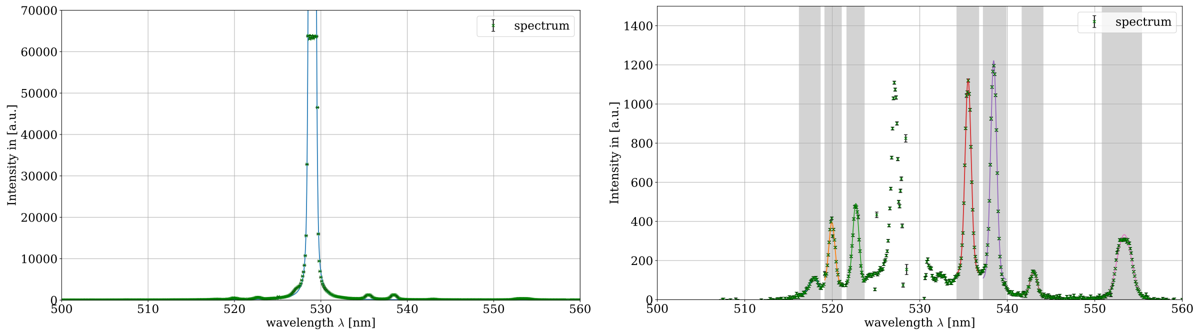
4.2 Measurements of different samples

As the calibration and investigations of the devices are done, we can now perform Raman spectroscopy with different samples. We investigate CCl_4, $CHCl_3$, CH_2Cl_2 and CS_2. Moreover, the spectra of different ethanol-water concentrations and a sulphur sample are measured.

4.2.1 Raman peaks and depolarisation of CCl_4

The spectrum of CCl_4 is measured with the CCD spectrometer and the monochromator. Because the measurements with the CCD spectrometer do not take that much time, we investigate the influence of different polarisations and the $\lambda/2$ plate on the spectrum. The spectrum is measured with an integration time of 5 s and 100 values are averaged. The laser current is set to 1.5 A.

The plot of the measured intensities can be seen in Figure 15a. The Rayleigh peak dominates the spectrum and leads to a huge background to the Stokes and Anti-Stokes peaks. Because of the high laser current, the centre of the Rayleigh peak gives an overflow. For the evaluation of the Raman peaks, we fit a Voigt function to the data without the overflow region and subtract it from the measured intensities. The fit can also be seen in Figure 15a and the corrected spectrum can be seen in Figure 15b. With this subtraction, it is not possible to completely get rid of the Rayleigh peak because the fit does not exactly match the data, but the remaining effect can be assumed to be a constant shift within one Raman peak. Also, the values in the centre of the spectrum do not belong to physical emission peaks but originate from deviations from the Voigt fit of the Rayleigh peak.



(a) with Rayleigh peak

(b) Rayleigh peak substracted

Fig. 15: In the two graphics, the Raman spectrum of CCl_4 is shown. The intensity is plotted against the wavelength λ. The left graphic shows the spectrum and a fit to the Rayleigh peak. The right graphic contains the spectrum from which the Rayleigh fit is subtracted and fits to the Stokes and Anti-Stokes peaks are shown.

The spectrum in Figure 15b shows seven Raman peaks. To determine the position and intensities of these peaks, Voigt fits to every peak are performed. For this purpose, only the data in the marked areas is used. From the wavelength λ of the peak maxima, it is now possible to calculate the energy difference $\Delta\nu$ of the Raman peaks to the Rayleigh peak:

$$\Delta\nu = \frac{c}{\lambda_{Raman}} - \frac{c}{\lambda_{Rayleigh}}. \tag{30}$$

The uncertainty of the wavelength is obtained from the fit uncertainty and with Gaussian error propagation the uncertainty of $\Delta\nu$ is given by:

$$\Delta(\Delta\nu) = c \cdot \sqrt{\left(\frac{\Delta\lambda_{Raman}}{\lambda_{Raman}^2}\right)^2 + \left(\frac{\Delta\lambda_{Rayleigh}}{\lambda_{Rayleigh}^2}\right)^2}. \tag{31}$$

The values for $\Delta\nu$ can be found in Table 3 and are given in cm^{-1}, which is a common unit in Raman-spectroscopy literature. There one finds also literature values for the energy difference of the vibrational states of CCl_4 from [8]. The literature values are assigned to the best matching peaks. To check the compatibility with the literature values, t-values are calculated. The lowest t-values for each Raman line can be seen in Table 3 as well. All values are compatible with literature values.

Tab. 3: The table includes the fitted values for $\Delta\nu$ of the CCl_4 sample in cm^{-1}. The CCD spectrometer, different polarisations and a $\lambda/2$-plate are used. The Stokes and Anti-Stokes peaks are numbered starting from the Rayleigh peak. Moreover, literature values for $\Delta\nu$ from [8] are given, as well as the t-values for the 90° polarisation without $\lambda/2$-plate.

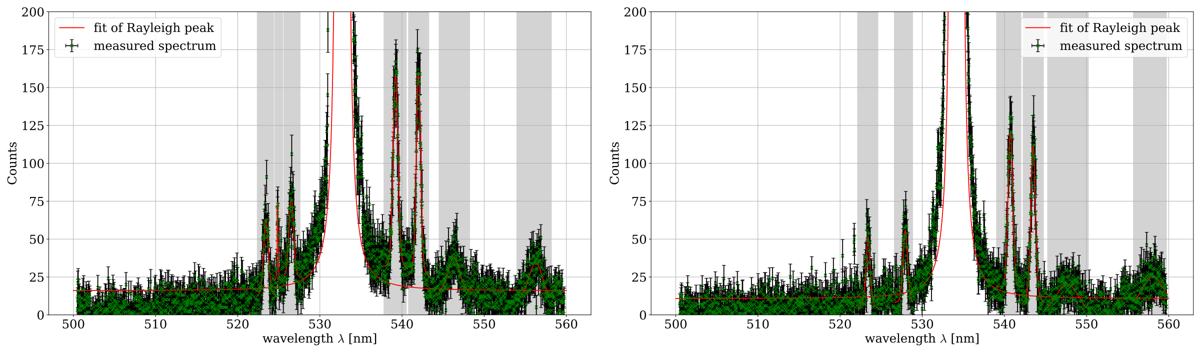
Peak	90°, no $\lambda/2$	90°, with $\lambda/2$	0°, no $\lambda/2$	0°, with $\lambda/2$	lit. value	t-value
1. Antistokes	230 ± 30	230 ± 30	230 ± 30	230 ± 30	217	0.4
1. Stokes	-230 ± 30	-230 ± 30	-230 ± 30	-230 ± 30	-217	0.4
2. Antistokes	330 ± 30	330 ± 40	330 ± 40	330 ± 30	314	0.5
2. Stokes	-330 ± 30	-330 ± 30	-330 ± 30	-330 ± 30	-314	0.6
3. Antistokes	410 ± 40	410 ± 70	400 ± 40	410 ± 70	459	1.2
3. Stokes	-490 ± 30	-490 ± 30	-490 ± 30	-490 ± 30	-459	0.9
4. Stokes	-830 ± 70	-830 ± 70	-830 ± 70	-830 ± 70	-776	0.8

To check if the spectrum deviates for different polarisations, all combinations of a $\lambda/2$ plate and a polarisation filter are used. The $\lambda/2$ plate is set on 45° which leads to a rotation of the laser polarisation of 90°. The polarisation filter is used in 0° and 90° setup. The corresponding spectra and fits can be seen in Figure 23 in the appendix. No additional peaks can be seen, but the intensity is nearly halved when measuring with a polarisation of 0°. This effect might be due to different detection of the polarisations in the CCD, which was investigated in subsubsection 4.1.4. Moreover, it is possible that the vibrations in the sample have a preferred direction and therefore lead to differently intense polarisations.

Also, inserting the $\lambda/2$ plate changes the intensity slightly. The measurements with different settings of the laser light did not determine the exact polarisation of the laser. So it is not completely clear what effect causes the intensity difference for different laser polarisations as the laser polarisation is not known. We will come back to this point in the discussion section.

The energy differences $\Delta\nu$ of the different settings are all summarized in Table 3. The values of $\Delta\nu$ are for all settings the same, and only the uncertainties differ sometimes. For the other samples, only the measurements with $\lambda/2$ plate and 90° polarisation will be evaluated, as this setting gives the highest intensities. The spectra for the other combinations are only checked for additional peaks.

The same measurements of the CCl_4 sample are also done with the monochromator. Here the $\lambda/2$ plate was not used, but the two different polarisations of the emission were measured. A scan rate of 0.5 Å/s is used, and the slit width is set to 250 µm. The obtained spectra can be seen in Figure 16. In comparison to the CCD spectrum, the Rayleigh peak is significantly more narrow and the Raman peaks are approximately on a constant background. Therefore, the Rayleigh peak is not subtracted from the data before fitting Voigt functions to the peaks. The fits to all peaks are also shown in Figure 16.



(a) 90° polarisation

(b) 0° polarisation

Fig. 16: In the graphic, the Raman spectra of CCl_4 is shown. The number of counts is plotted against the wavelength λ. Moreover, Voigt fits to the peaks can be seen in red. For the measurements, polarisations of 90° and 0° but no $\lambda/2$-plate are used.

From the peak positions, again, the energy differences $\Delta\nu$ between the Rayleigh and the Raman peaks are calculated. The values are summarized in Table 4. It is tried to identify the measured peaks with literature values of the Raman peaks and t-values are calculated. The t-values are all higher than 2 and therefore the calculated energy differences are not compatible with literature. Possible reasons for this will be part of the discussion.

In comparison to the measurements with the CCD spectrometer now not all peaks can be identified for both polarisations. The second Anti-Stokes peak only appears for 90° polarisation. Moreover, the fitting uncertainties are much smaller than for the CCD. It is striking, that the positions of the Rayleigh peak for both polarisations differ from each other and the literature. Looking at the peak positions and not the energy differences, one sees that the first peak is at the same position for both polarisations. But for higher wavelengths, the peak positions deviate visibly. This may have an impact on the energy differences as well, and will be mentioned in the discussion. A possible reason for this might be the unreliable motion of the lattice motor.

Tab. 4: The table includes the fitted values for $\Delta\nu$ of the CCl_4 sample in cm^{-1} and the wavelength λ of the Rayleigh peak. The monochromator and different polarisations are used. The Stokes and Anti-Stokes peaks are numbered starting from the Rayleigh peak. Moreover, literature values for $\Delta\nu$ from [8] are given, as well as the corresponding t-values.

Peak	90°, no $\lambda/2$	0°, no $\lambda/2$	literature value	t-value
1. Antistokes	220.9 ± 0.6	221.6 ± 0.6	217	6.1
1. Stokes	-226.1 ± 0.3	-226.9 ± 0.3	−217	30.5
2. Antistokes	280.7 ± 0.3	–	314	109
2. Stokes	-321.5 ± 0.3	-321.3 ± 0.3	−314	28
3. Antistokes	332.6 ± 0.6	385.3 ± 0.6	459	195.8
3. Stokes	-467.2 ± 1.7	-460 ± 2	−459	5.0
4. Stokes	-794.7 ± 1.6	-802 ± 3	−776	11.9
Rayleigh	(532.7165 ± 0.0002) nm	(534.1741 ± 0.0002) nm	532 nm	3338.6

As now the Raman peaks of CCl_4 are determined, it is possible to calculate the depolarisation of CCl_4. The depolarisation ρ is the ratio of the intensity $I_\parallel$ of the Raman peaks parallel polarized to the laser polarisation and the intensity $I_\perp$ of the Raman peaks perpendicular to the laser polarisation. Because the laser polarisation was not determined but the laser intensity for 0°

polarisation was higher, it is assumed that the laser is polarized with 0°. If this approach is reasonable, will be considered in the discussion.

The intensity of a peak is given by the area under the curve. As the Voigt function is normalized, the intensity is given by the scaling factor A. Because the measurement devices do not detect both polarisations in the same way, the intensities have to be corrected with the ratio r of the intensities obtained from the measurements of white light in subsubsection 4.1.4.

With the CCD spectrometer and the monochromator, measurements without $\lambda/2$ plate are taken. This means that the intensities measured with 0° polarisation filter are assigned to the parallel polarisation $I_{||}$ and the ones measured with 90° polarisation to $I_{\perp}$. Because we do not know the polarisation of the laser, we also evaluate the measurements with $\lambda/2$ plate of the CCD but switch the roles of 0° and 90° polarisations. So the depolarisation can be calculated with:

$$\text{no } \lambda/2: \quad \rho = \frac{A_0 \cdot r}{A_{90}} \tag{32}$$

$$\Delta\rho = \sqrt{\left(\frac{r}{A_{90}}\Delta A_0\right)^2 + \left(\frac{A_0}{A_{90}}\Delta r\right)^2 + \left(\frac{A_0 \cdot r}{A_{90}^2}\Delta A_{90}\right)^2} \tag{33}$$

$$\text{with } \lambda/2: \quad \rho = \frac{A_{90}}{A_0 \cdot r} \tag{34}$$

$$\Delta\rho = \sqrt{\left(\frac{1}{A_0 \cdot r}\Delta A_{90}\right)^2 + \left(\frac{A_{90}}{A_0^2 \cdot r}\Delta A_0\right)^2 + \left(\frac{A_{90}}{A_0 \cdot r^2}\Delta r\right)^2} \tag{35}$$

$$\tag{36}$$

The so obtained depolarisations can be found in Table 5. Some values have large errors, and the depolarisations vary over a wide range. Moreover, the depolarisations measured with the CCD and the monochromator do not match at all. Since no literature values are known, it is not clear where the error comes from. Reasons for this will be raised in the discussion.

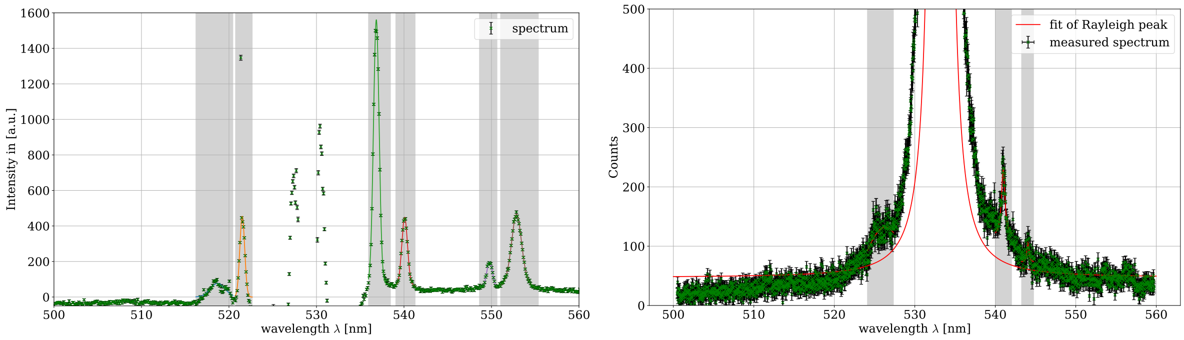
Tab. 5: In the table, the calculated depolarisations ρ measured with both, the CCD and the monochromator are listed for both setups with and without $\lambda/2$-plate. They are again sorted by the corresponding peaks. There are no literature values given for the depolarisation, which could be added.

Peak	CCD ρ_z, no $\lambda/2$	CCD ρ_z with $\lambda/2$	Mono. ρ_z, no $\lambda/2$
1. Antistokes	0.8 ± 0.2	1.6 ± 0.5	0.14 ± 0.07
1. Stokes	0.9 ± 0.3	1.2 ± 0.5	0.183 ± 0.008
2. Antistokes	0.7 ± 0.7	1 ± 2	-
2. Stokes	0.9 ± 1.4	1 ± 3	0.148 ± 0.006
3. Antistokes	0.9 ± 1.8	1 ± 5	0.11 ± 0.05
3. Stokes	2 ± 3	1.3 ± 1.7	0.071 ± 0.007
4. Stokes	0.9 ± 0.9	0 ± 30	0.082 ± 0.007

4.2.2 Raman peaks of $CHCl_3$ and CH_2Cl_2

With a completely analogue approach, the Raman spectra of $CHCl_3$ and CH_2Cl_2 are analysed. With the same settings as in the previous section, the spectra of the samples are measured with the CCD spectrometer and the monochromator. Here, only the spectra for 90° polarisation and without $\lambda/2$ plate are evaluated. The other spectra are only checked for deviations which did not occur.

The spectrum of $CHCl_3$ and the Voigt fits can be found in Figure 17.



(a) CCD-spectrometer (b) monochromator

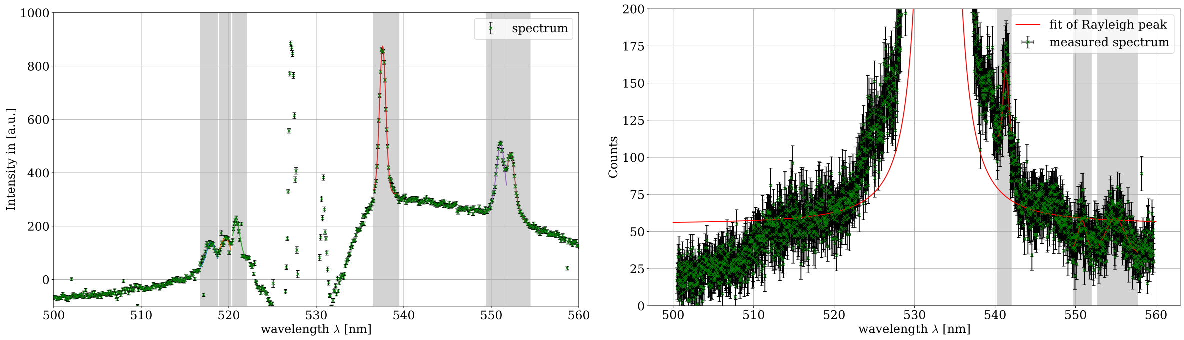
Fig. 17: In the graphic, the Raman spectra of $CHCl_3$ are shown for measurements with both, the CCD-spectrometer and the Monochromator. The number of counts is plotted against the wavelength λ. Moreover, Voigt fits to the peaks can be seen. For the measurements, a polarisation of 90° and no $\lambda/2$-plate is used.

It is conspicuous that the higher order Raman peaks can only be detected as Stokes peaks with the CCD. The energy differences $\Delta\nu$, which are again calculated, are shown in Table 6. The differences in the uncertainties between CCD and monochromator can be seen. Also, the shift in wavelengths of the monochromator appears in this spectrum too. As no literature values were found, it is not possible to compare them to the measured values.

Tab. 6: The table includes the fitted values for $\Delta\nu$ of the $CHCl_3$ sample in cm^{-1}. Both, the CCD-sensor and the monochromator are used with a polarisation of 90° and without the $\lambda/2$-plate. The Stokes and Anti-Stokes peaks are numbered starting from the Rayleigh peak. Except for the Rayleigh peak, there are no literature values given for comparison.

Peak	CCD	Monochromator	literature value	t-value
1. Antistokes	270 ± 20	245 ± 5	-	-
1. Stokes	-270 ± 30	-269.8 ± 0.4	-	-
2. Antistokes	380 ± 70	-	-	-
2. Stokes	-390 ± 30	-374.1 ± 0.8	-	-
3. Stokes	-710 ± 20	-	-	-
4. Stokes	-810 ± 40	-	-	-
Rayleigh	-	$(533.201\,79 \pm 0.000\,09)$ nm	532 nm	13 390.2

The spectra of the CH_2Cl_2 sample are shown in Figure 18. There, the Voigt fits are also plotted.



(a) CCD-spectrometer (b) monochromator

Fig. 18: In the graphic, the Raman spectra of CH_2Cl_2 are shown for measurements with both, the CCD-spectrometer and the Monochromator. The number of counts is plotted against the wavelength λ. Moreover, Voigt fits to the peaks can be seen. For the measurements, a polarisation of 90° and no $\lambda/2$-plate is used.

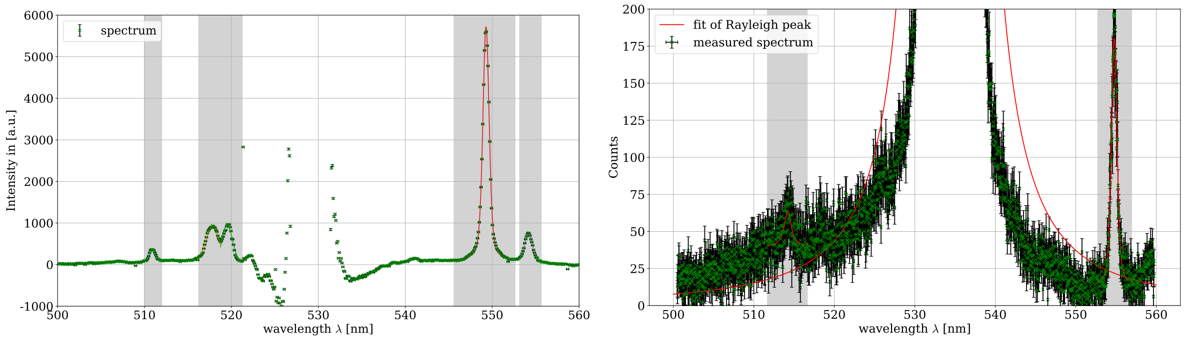
With the CCD spectrometer, it is again possible to identify several Raman peaks. The monochromator only allows the evaluation of three peaks because the noise dominates in the other areas. The calculated energy differences are summarized in Table 7. Again, no literature values could be used for comparison.

Tab. 7: The table includes the fitted values for $\Delta\nu$ of the CH_2Cl_2 sample in cm^{-1}. Both, the CCD-sensor and the monochromator are used with a polarisation of 90° and without the $\lambda/2$-plate. The Stokes and Anti-Stokes peaks are numbered starting from the Rayleigh peak. Except for the Rayleigh peak, there are no literature values given for comparison.

Peak	CCD	Monochromator	literature value	t-value
1. Antistokes	300 ± 40	-	-	-
1. Stokes	-300 ± 30	-292.0 ± 1.0	-	
2. Antistokes	340 ± 70	-	-	-
3. Antistokes	400 ± 50	-	-	-
4. Stokes	-	-610 ± 2	-	-
5. Stokes	-760 ± 30	-745 ± 2	-	-
6. Stokes	-790 ± 40	-	-	-
Rayleigh	-	(532.8866 ± 0.0010) nm	532 nm	9239.9

4.2.3 Raman peaks of CS_2

The spectrum of CS_2 is also measured and evaluated in complete analogy. The measured spectrum with Voigt fits is plotted in Figure 19.



(a) CCD-spectrometer (b) monochromator

Fig. 19: In the graphic, the Raman spectra of CS_2 are shown for measurements with both, the CCD-spectrometer and the Monochromator. The number of counts is plotted against the wavelength λ. Moreover, Voigt fits to the peaks can be seen. For the measurements, a polarisation of 90° and no $\lambda/2$-plate is used.

With the CCD spectrometer, it is possible to analyse two Anti-Stokes peaks nearby the Rayleigh peak, but no corresponding Stokes peaks are found. This is surprising because the intensity of the Stokes peaks should be higher. With the monochromator, it was only possible to identify the 3. Raman peak. The energy differences of the Raman peaks are shown in Table 8. Here also literature values from [8] are summarized and t-values are calculated.

Tab. 8: The table includes the fitted values for $\Delta\nu$ of the CS_2 sample in cm^{-1}. Both, the CCD-sensor and the monochromator are used with a polarisation of 90° and without the $\lambda/2$-plate. The Stokes and Anti-Stokes peaks are numbered starting from the Rayleigh peak. Moreover, literature values for $\Delta\nu$ from [8] are given, as well as the corresponding t-values.

Peak	CCD	Mono.	literature value	t-value (CCD)	t-value (Mono.)
1. Antistokes	350 ± 50	-	397	1.0	
2. Antistokes	410 ± 70	-	397	0.2	
3. Antistokes	670 ± 30	734 ± 2	658	0.4	37.1
3. Stokes	-700 ± 20	-687.4 ± 0.2	658	1.6	5692.9
4. Stokes	-860 ± 30	-	658	6.6	
			or 1535	22.4	
Rayleigh	-	$(534.400\,15 \pm 0.000\,14)\,\mathrm{nm}$	532 nm		16 793.1

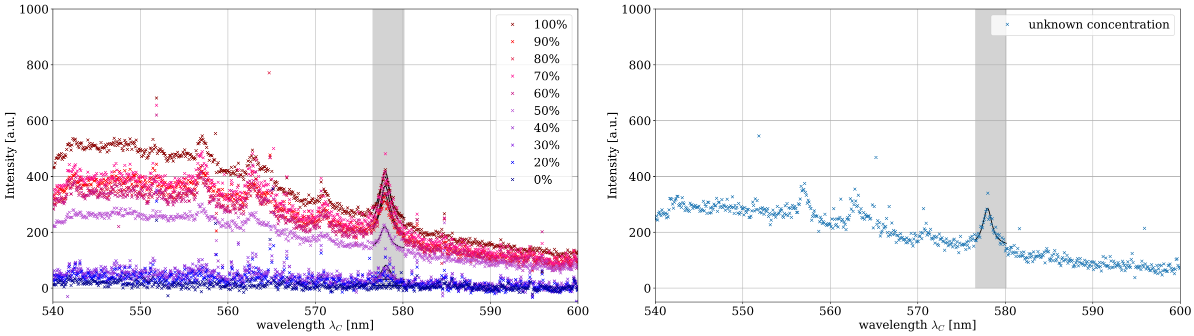
4.3 Finding unknown ethanol concentration

The Raman spectroscopy can also be used to find the concentration of an ethanol-water sample. Therefore, the intensities of Raman peaks are measured for different known concentrations of ethanol. The samples are produced by taking different injections and all contain 10 mL. The injection resolution is 0.5 mL and the uncertainty is calculated with triangular distribution $\frac{0.5\,\mathrm{mL}}{\sqrt{6}}$. We therefore calculate the concentration and its uncertainty in the following way:

$$C = \frac{V_{et}}{V_{et} + V_{H_2O}}, \tag{37}$$

$$\Delta C = C \Delta V \sqrt{\left(\frac{1}{V_{et}} - \frac{1}{V_{et} + V_{H_2O}}\right)^2 + \left(\frac{1}{V_{et} + V_{H_2O}}\right)^2}. \tag{38}$$

Because many measurements are necessary, the spectra are only analysed with the CCD spectrometer. The notch filter is used as the Raman spectrum appears at higher wavelengths than before. The measured spectra for different concentrations are shown in Figure 20a. The colour indicates the concentration, where red corresponds to dominating ethanol and blue to dominating water fraction. Four peaks are visible. Their intensity shrinks with increasing water fraction. The peak on the right side is the one with the highest intensity and lowest background, and is therefore chosen to evaluate the intensity. A Voigt function is fitted for every concentration. The fits can also be seen in Figure 20a. The sample of unknown concentration is analysed in the same way and can be seen in Figure 20b.



(a) Regression spectra

(b) Unknown sample spectrum

Fig. 20: In the graphic, all the spectra of the regression measurement with ethanol-water-samples as well as the unknown sample are shown. The intensity of the spectrum is measured with the CCD-spectrometer and the notch filter and plotted in arbitrary units against the wavelength λ in nm. For the regression, higher ethanol concentrations are depicted in red, whereas higher water concentrations are plotted in blue. In addition, Voigt fits to the peak with the highest intensity are added, as well as the corresponding fitting area in grey.

To find the dependence of intensity and ethanol concentration, the scaling factors A of the known concentrations, which correspond to the area under a Voigt function and therefore the peak intensity, are plotted against the concentration C. This is shown in Figure 21. A linear fit is performed and is plotted there too.

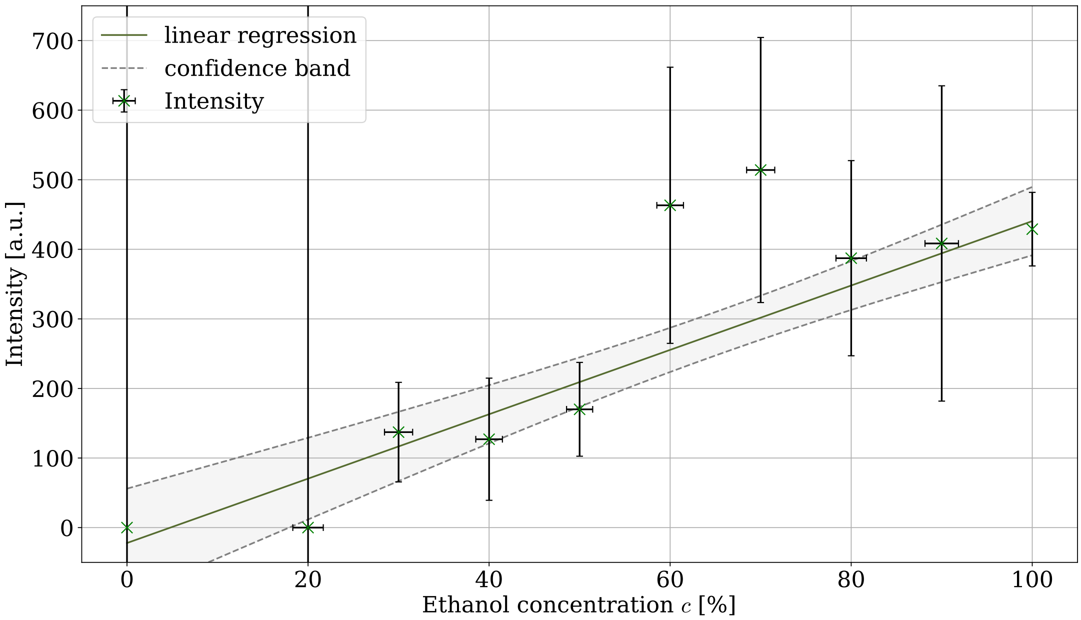


Fig. 21: In the graphic, the linear regression for the ethanol-water concentration measurement can be found, where the intensity I of the peaks measured with the CCD-device is plotted against the known ethanol concentration C on the x-axis. The corresponding linear regression with confidence band is also added. The uncertainties of the data points are also considered in the linear regression.

With the linear fit, the following slope a and y-intercept b of the linear function are determined using `scipy.optimize.curve_fit` [7]:

$$a = 4.6 \pm 1.1\%^{-1} \tag{39}$$

$$b = -20 \pm 80 \tag{40}$$

The intensity of the unknown sample can now be assigned to a concentration in the following way:

$$C = a \cdot A + b, \tag{41}$$

$$\Delta C = \sqrt{(a\Delta A)^2 + (A\Delta a)^2 + (\Delta b)^2 + 2A \cdot Cov(a,b)}. \tag{42}$$

The unknown concentration is determined to be:

$$C = (60 \pm 20)\,\% \tag{43}$$

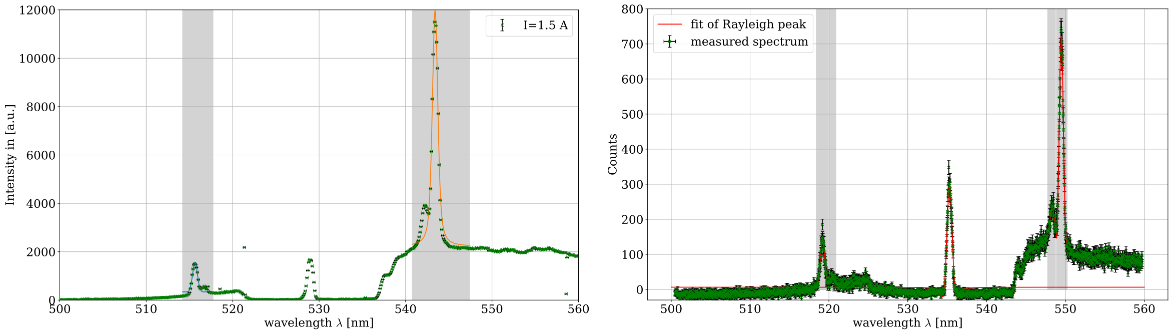
4.4 Finding the temperature of sulphur

For Stokes and Anti-Stokes emission, different vibrational states have to be occupied. Because the occupation probability depends on the temperature, the intensities of the Stokes and Anti-Stokes peaks can be used to get the temperature of a sample. For this purpose, Equation 9 can be solved for the temperature T and using $\Delta\nu = \frac{\nu_A - \nu_S}{2}$, $\nu_0 = \frac{\nu_A + \nu_S}{2}$ this results in:

$$T = \frac{h}{k_B} \frac{\frac{\nu_A - \nu_S}{2}}{\log\left(\frac{I_S}{I_A}\right) + 4\log\left(\frac{\nu_A}{\nu_S}\right)}, \tag{44}$$

$$\begin{aligned} \Delta T = \Bigg[&\left(\frac{T \Delta I_S}{I_A \left(\log\left(\frac{I_S}{I_A}\right) + 4\log\left(\frac{\nu_A}{\nu_S}\right)\right)} \right)^2 + \left(\frac{T \Delta I_A}{I_S \left(\log\left(\frac{I_S}{I_A}\right) + 4\log\left(\frac{\nu_A}{\nu_S}\right)\right)} \right)^2 \\ &+ \left(\frac{-4T}{\nu_A \left(\log\left(\frac{I_S}{I_A}\right) + 4\log\left(\frac{\nu_A}{\nu_S}\right)\right)} + \frac{h}{2k_B \left(\log\left(\frac{I_S}{I_A}\right) + 4\log\left(\frac{\nu_A}{\nu_S}\right)\right)} \right)^2 \Delta\nu_A^2 \\ &+ \left(\frac{4T}{\nu_S \left(\log\left(\frac{I_S}{I_A}\right) + 4\log\left(\frac{\nu_A}{\nu_S}\right)\right)} - \frac{h}{2k_B \left(\log\left(\frac{I_S}{I_A}\right) + 4\log\left(\frac{\nu_A}{\nu_S}\right)\right)} \right)^2 \Delta\nu_S^2 \Bigg]^{\frac{1}{2}} \end{aligned} \tag{45}$$

In the following, the temperature of a sulphur sample will be determined. The spectrum of sulphur is measured with the settings also used for the analysis of the previous samples. Both devices and a notch filter are used. The measured spectrum with 90° polarisation and without $\lambda/2$ plate can be found in Figure 22. With the CCD, also measurements for different laser currents are taken. The corresponding data and fits can be found in the appendix in Figure 24. A Stokes and an Anti-Stokes peak can be identified. The CCD spectrum already indicates that this might be a double peak, which can not be resolved. The monochromator allows to separate the both peaks. Therefore, we only fit one Voigt profile to the CCD spectrum and two Voigt profiles to the monochromator spectrum. The fits can be found in Figure 22 as well. In addition, the Rayleigh peak of the monochromator is fitted as the determined laser wavelength varied in the previous measurements.



(a) CCD-spectrometer

(b) Monochromator

Fig. 22: In the graphic, the spectra of the sulphur sample are shown. They are measured using the CCD-device as well as the monochromator and the notch filter for a laser intensity of $I_l = 1.5\,\text{A}$. The intensity in arbitrary units is plotted against the wavelength λ in nm. For the Stokes and the Anti-Stokes peaks, a Voigt fit is performed and added to the graphic.

For the CCD, the energy difference of the Stokes and the Anti-Stokes peaks are determined for every laser current. The mean is taken and the standard deviation of the mean is used as the uncertainty. This gives the following values:

$$\Delta\nu_A = (490.940 \pm 0.016)\,\text{cm}^{-1}, \tag{46}$$

$$\Delta\nu_S = (-500.101 \pm 0.002)\,\text{cm}^{-1}. \tag{47}$$

With the monochromator, the energy differences of the four fitted peaks are calculated to be:

$$\Delta\nu_{A1} = (533.6 \pm 1.2)\,\mathrm{cm}^{-1}, \tag{48}$$
$$\Delta\nu_{A2} = (579.7 \pm 0.3)\,\mathrm{cm}^{-1}, \tag{49}$$
$$\Delta\nu_{S1} = (-443.4 \pm 0.6)\,\mathrm{cm}^{-1}, \tag{50}$$
$$\Delta\nu_{S2} = (-481.80 \pm 0.16)\,\mathrm{cm}^{-1}, \tag{51}$$

where index 1 represents the inner, less intense peak and index 2 the outer more intense one.

In addition to the frequency difference, the intensity of the Stokes and Anti-Stokes peaks need to be determined. This information is also taken from the fits. Since for every laser current, there are also different intensities in the peaks, the values are not presented here, but using Equation 44, the values for the temperatures and their uncertainties are directly calculated and summarized in Table 9.

Tab. 9: In the table, the calculated temperatures T of the sulphur sample are summarized for a given laser current I_l.

Laser current I_l in A	temperature T in K
1.5	270 ± 30
1.4	270 ± 30
1.3	270 ± 30
1.2	280 ± 30
1.1	270 ± 30
1.0	270 ± 30

We can see directly, that all the values are in a realistic order of magnitude, but are lower than room temperature $T_{room} \approx 293\,\mathrm{K}$. However, all the values are still compatible with each other and with room temperature in the range of their uncertainties. To get a final result, all the temperatures are averaged and the standard deviation of the mean is taken as an uncertainty:

$$T = (273.0 \pm 0.4)\,\mathrm{K}. \tag{52}$$

Due to the averaging, we now have a significantly lower uncertainty and find that the final result deviates significantly from room temperature. This result will be part of the discussion part in the last chapter of the report.

An analogue calculation can be performed for the monochromator values. However, the measurements have a lot of noise due to a higher background and in addition do not have the correct frequency difference due to problems with the motor. In addition, only one measurement can be taken, due to time limitations for the long measuring monochromator. Therefore, when calculating the temperatures for the monochromator, we find values with an extremely high uncertainty and a wrong order of magnitude:

$$T_1 = (450 \pm 9230)\,\mathrm{K} \tag{53}$$
$$T_2 = (500 \pm 200)\,\mathrm{K} \tag{54}$$

The values do therefore not contain any useful information on the actual temperature and can be neglected.

5 Discussion

5.1 Summary of Results

In a first step, after the setup of the optical path, a calibration is performed for both detectors using the known spectrum of the Hg-lamp. Using the monochromator, the effect of the slit width was examined and a minimal slit width of 100 µm was found. In addition, using the white light, the lower wavelength bound was studied. Using the CCD-spectrometer, the following wavelength of the laser was calculated:

$$\lambda = (529.035 \pm 0.005)\,\mathrm{nm}, \tag{55}$$
$$\lambda_{lit} = 532\,\mathrm{nm}, \tag{56}$$
$$t = 593. \tag{57}$$

In addition, the effect of the notch filter, as well as the different polarisations were studied with both measurement devices.

For the four different liquid samples, a successful determination of Stokes and Anti-Stokes peaks and therefore different vibrational frequencies was possible. For CCl_4 4 Stokes and 3 Anti-Stokes peaks could be identified, all summarized in Table 3 and Table 4. In addition, depolarisations were calculated, with results spread around 1 for the measurements with the CCD-device and around 0.1 for the monochromator. The results are found in Table 5. The 2 identified Anti-Stokes and 4 Stokes peaks of $CHCl_3$ are summarized in Table 6, the 3 Anti-Stokes and 4 Stokes peaks of CH_2Cl_2 can be found in Table 7 and the 3 Anti-Stokes and 2 Stokes peaks of CS_2 are listed in Table 8. In general, for the values measured with the CCD, the uncertainties are higher, but all the values are compatible with literature. For the monochromator, higher resolution and therefore smaller uncertainties are achieved, but all the t-values are significantly higher. This behaviour will be discussed in the following part.

For the ethanol concentration of the unknown sample, a linear regression is performed, and the following concentration is found:

$$C = (60 \pm 20)\,\%. \tag{58}$$

Since the real concentration is unknown, a comparison to the actual value is not possible, however, the value is in a realistic range. In the discussion, the high uncertainty on this value will be thematised. For the sulphur temperature, six different measurements with different laser intensities are performed, and the resulting temperatures are averaged to the following value:

$$T = (273.0 \pm 0.4)\,\mathrm{K}. \tag{59}$$

This value is in a realistic range, but deviates from room temperature significantly, which will also be a topic in the following discussion part. With the monochromator no useful temperature could be determined.

5.2 Discussion of results and uncertainties

A first important point, that needs to be discussed, is the treatment of the polarisation in the experiment. Although measurements of the polarisation for different wavelengths with white light were performed, the exact effect of the polarisation remains unclear. This is due to the fact, that we do not really know the polarisation of the white light and can therefore not tell, which effect comes from the white light and which effect from the measuring device. Therefore, the correction factor can not be used reliably. In addition, a measurement or calibration of the laser polarisation was not performed. It can therefore only be guessed, which polarisation the laser has, making an analysis of the depolarisation difficult and the results of this part very inaccurate. It remains also unclear, what effect the $\lambda/2$-plate has on the Raman spectra.

The measurement of the unknown ethanol sample provides a reasonable result, but the uncertainty is in an order of magnitude that does not provide useful information on the actual concentration. There are several reasons for this. First, the measurements for the different samples resulted in quite low Raman intensities, making a fit on the data quite difficult. This leads to high uncertainties of the intensity and a very imprecise linear regression. In addition, a high background makes it even harder to get the right value for the intensity, since depending on the y-intercept, the fitted intensity changes drastically. The last and most important point, however is, that a measurement under the same conditions is nearly impossible for the different calibration samples. Tests with the optical path showed, that changing the position of the sample by less than a mm can have an impact on the intensity changing it by orders of magnitude. Also, during the quite long measurement of all the calibration samples, small fluctuations in the optical path can have a great impact on the intensity. In contrast to the position of the peak, the intensity is therefore very hard to determine precisely.

In the last part for the temperature measurement of sulphur, we find results having the right order of magnitude, but being all significantly smaller than room temperature and not compatible with this value in the range of the uncertainty. Again, we use the intensity of the peaks to determine the temperature, but here, however, the previously discussed effects should be negligible, since only a comparison within the same measurement data needs to be taken. The wrong value for the temperature could therefore be explained by the background, which again makes it hard to identify the right intensity. This can also be seen by the fact, that at least the right order of magnitude is achieved, since the background also does not change the intensity by an order of magnitude. This is however a systematic error not taken into account and the final result is not compatible with literature.

One interesting discussion point is the evaluation of the peaks using Voigt functions. As already mentioned, this is a good way to find a compromise between the physics in the peak as described by the Lorentz profile and the statistical fluctuations as described by the Gaussian profile, especially since both types are also special cases of the Voigt function. However, it is important to mention, that nearly all the performed fits are almost completely of Gaussian shape. This may come from the fact, that in the context of the lab and the provided measurement devices, a completely clear and precise optical experiment is very hard to accomplish, leading to a lot of noise on the data. Therefore, the Gaussian behaviour dominates the fits significantly.

In addition, for nearly all the fits, we have high reduced χ^2-values, indicating, that the fitted model does not describe the date accurately. This may be due to two points. Firstly, the fitting function also accounts for constant background by including a y-offset, but does not include the exact structure of the background. Because of taking the average of a lot of measurements with the CCD, the uncertainty on the background is however extremely small, although it can not be described by the fit, leading to high incompatibilities. In addition, the intensity in the fit is given in arbitrary units, and we assumed Poissonian uncertainty on the data. However, this could not be the right guess for the uncertainties. Underestimating the uncertainties of the data can also be indicated by high χ^2-values. For the monochromator, the uncertainties are a lot higher and seem to be estimated in the right way, thus reducing the χ^2-values for the fits of the peak.

In general, a discussion of the two measurement devices is also important: For the CCD-spectrometer, as already expected, there are two main issues. Firstly, there are several pixels that have an error and therefore do not count reliably. This can be a problem since it makes the analysis of numerous peaks harder, but in general can be taken into account and therefore does not affect the results. The other point is the low resolution of the CCD-device in comparison to the monochromator. This also impacts the calibration and leads to quite high uncertainties

on the wavelengths. This is, for example, visible in the deviation of the laser wavelength from literature and in the high uncertainties in the Stokes and Anti-Stokes peaks for every sample. However, in the range of this uncertainty, all the measured values are compatible with literature, which makes the CCD a quick and reliable, but not very accurate device.
For the monochromator, there are also two main disadvantages: As expected, the measurement times are a lot longer than with the CCD. Therefore, we are not able to repeat all the measurements with a monochromator, but only a selected number of samples can be examined, especially in a one-week experiment. In addition, a problem arose, that was not expected in the first place. Due to offsets in the final wavelength between the analogue display on the monochromator and the digital counter in the LabView program, we could conclude, that the motor of the monochromator did not work reliably. This offset also changed from measurement to measurement, so it was not possible to take the effect into account in the analysis. By observing the display of the monochromator, one could also conclude, that the motor did not just rotate slower than displayed in the program, but rather showed minimal changes in the velocity during the measurement. In some cases, we even had to cancel the measurement, because of the motor stopping completely. Due to a fine calibration of the monochromator, the uncertainties on the measured wavelength for this device are quite small. However, the unreliability of the motor rotation adds a significant systematic error, that is not possible to quantify usefully. This also leads to the high incompatibilities with all the literature values, and especially to the extremely high t-values for the position of the Rayleigh-peaks.

In addition, some elements of the optical path would need to be improved to optimize the measurement for all parts of the experiment. First of all, the setup would be a lot easier to optimize and collimate if it was not necessary to change the height of the optical path between the samples and the measurement devices. Moreover, it was not possible to fix the white and the Hg lamp in the optical path reliably, because of space issues on the optical bench. For the sulphur sample, the sample could not be fixed in the sample holder due to a different container for the sample and as discussed before, this lead to high differences in the intensity.

References

[1] M. Köhli: *Raman-Effect - Versuchsanleitung Fortgeschrittenen Praktikum Teil 2*, (Freiburg im Breisgau: 2011)

[2] Ewen Smith, Geoffrey Dent: *Modern Raman Spectroscopy – A Practical Approach*, (John Wiley and Sons, Ltd: 2005)

[3] Derek A. Long: *The Raman Effect*, (John Wiley and Sons, Ltd: 2002)

[4] P. Vandenabeele: *Practical Raman Spectroscopy – an introiduction*, (John Wiley and Sons, Ltd: 2013)

[5] Richard L. McCreery: *Raman Spectroscopy for Chemica1 Analysis*, (John Wiley and Sons, Ltd: 2000)

[6] X. Yuan, R. A. Mayanovic: *An Empirical Study on Raman Peak Fitting and Its Application to Raman Quantitative Research*, (Applied Spectoscropy: 2017)

[7] Virtanen, Pauli; Gommers, Ralf; Oliphant, Travis E.; Haberland, Matt; Reddy, Tyler; Cournapeau, David; Burovski, Evgeni; Peterson, Pearu; Weckesser, Warren; Bright, Jonathan; van der Walt, Stéfan J.; Brett, Matthew; Wilson, Joshua; Millman, K. Jarrod; Mayorov, Nikolay; Nelson, Andrew R. J.; Jones, Eric; Kern, Robert; Larson, Eric; Carey, C J; Polat, İlhan; Feng, Yu; Moore, Eric W.; VanderPlas, Jake; Laxalde, Denis; Perktold, Josef; Cimrman, Robert; Henriksen, Ian; Quintero, E. A.; Harris, Charles R.; Archibald, Anne M.; Ribeiro, Antônio H.; Pedregosa, Fabian; van Mulbregt, Paul; SciPy 1.0 Contributors: *SciPy 1.0: Fundamental Algorithms for Scientific Computing in Python* in *Nature Methods*, (2020)

[8] W. M. Haynes: *CRC handbook of chemistry and physics, 95th edition*, (Boca Raton: 2014-15)

7 Appendix

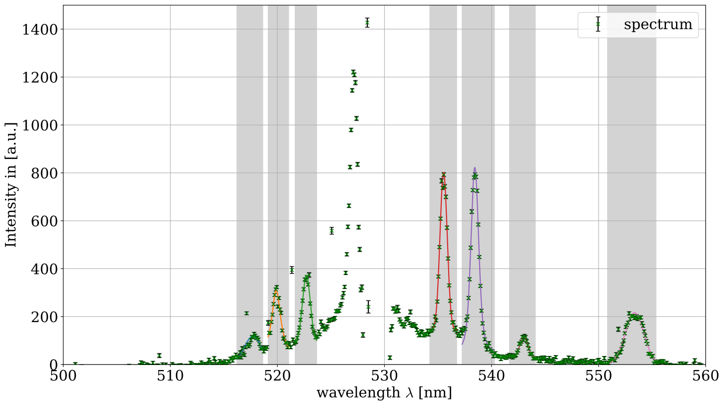
7.1 Tables and graphics

7.1.1 Table with the used symbols in the protocol

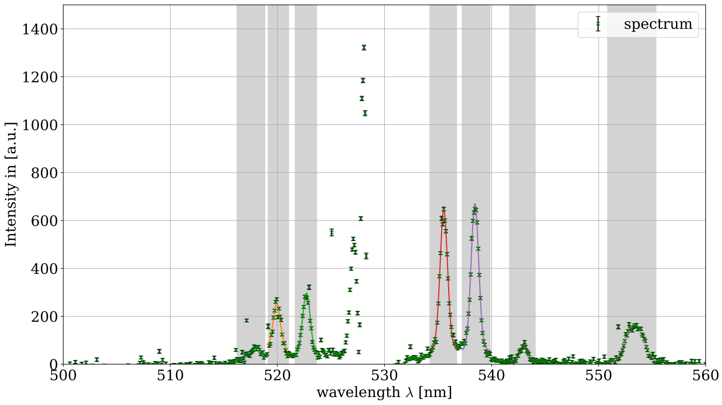
Tab. 10: Table of the used symbols for the parameters used in the protocol.

Symbol	Parameter
$\vec{E}$	electric field
$\vec{P}$	dipole moment
α	polarisability tensor
r, r_0	position, equilibrium position
ω_0	frequency of light
ω_s	vibrational frequency
$I, I_{\|\|}, I_{\perp}$	intensity, parallel and vertical
ρ_z	depolarisation
$\bar{\alpha}, \gamma$	quantities describing the polarisation ellipsoid
$N_{0,e}$	occupation number
T	temperature
$\Delta\nu$	frequency difference
λ	wavelength
n	measurement number
c	channel number
C	concentration
I_l	laser current
A	amplitude - fitting parameter
y_0	y-offset - fitting parameter
μ	mean - fitting parameter
σ, γ	width parameters - fitting parameters

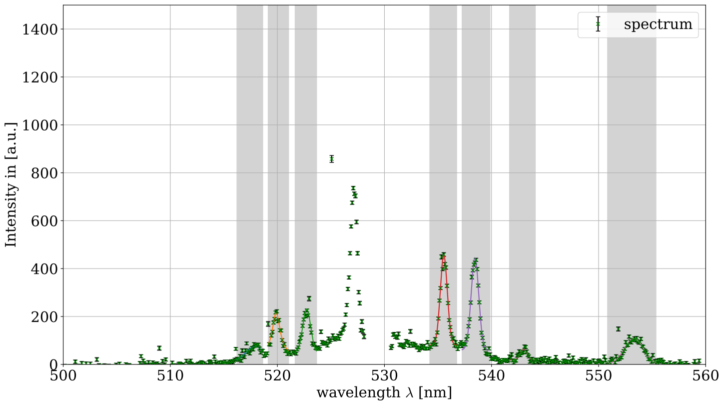
7.1.2 Graphics



(a) 90° polarisation with $\lambda/2$-plate

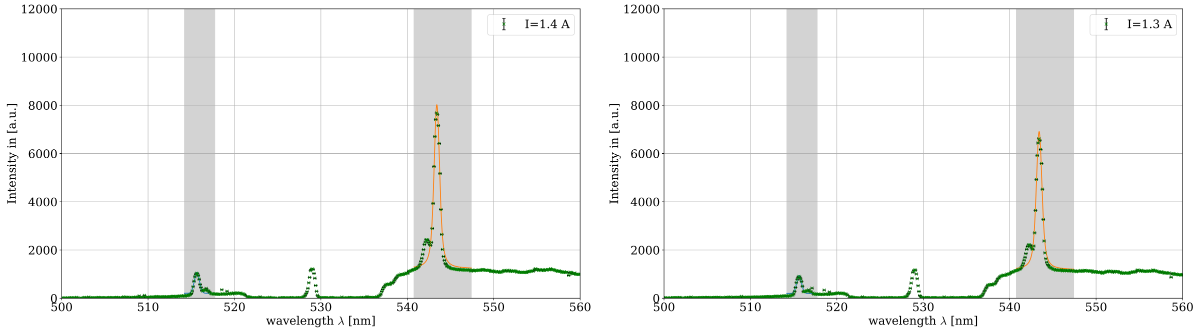


(b) 0° polarisation without $\lambda/2$-plate

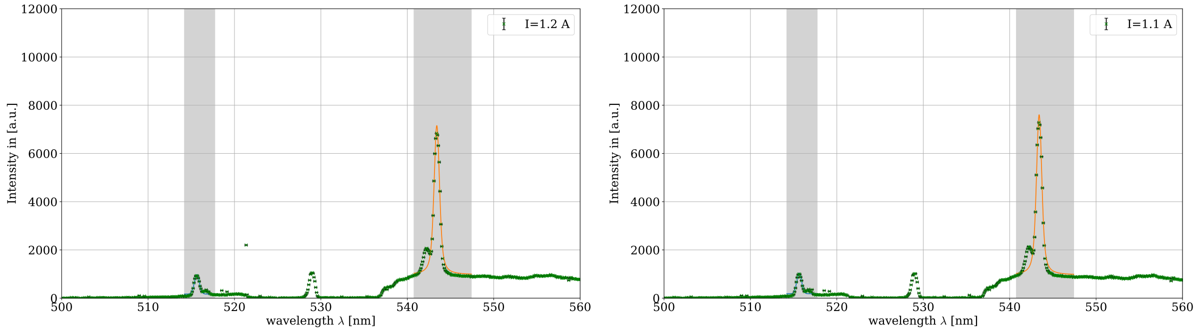


(c) 0° polarisation with $\lambda/2$-plate

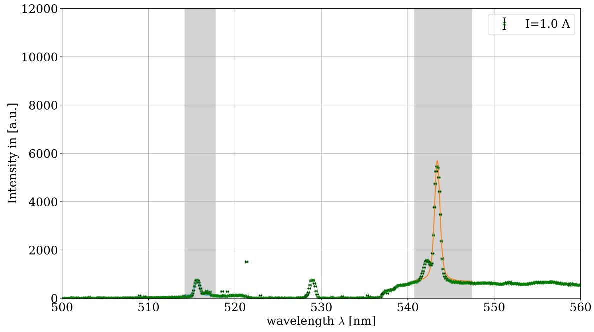
Fig. 23: In the three graphics, the Raman spectrum of CCl_4 is shown. The intensity is plotted against the wavelength λ. The first spectrum is measured in 90° polarisation with a $\lambda/2$-plate, the second in 0° polarisation without a $\lambda/2$-plate and the last spectrum in 0° polarisation with a $\lambda/2$-plate. For all spectra, the Rayleigh peak was subtracted.



(a) $I_l = 1.4\,\mathrm{A}$ (b) $I_l = 1.3\,\mathrm{A}$



(c) $I_l = 1.2\,\mathrm{A}$ (d) $I_l = 1.1\,\mathrm{A}$



(e) $I_l = 1.0\,\mathrm{A}$

Fig. 24: In the graphic, the additional spectra of the sulphur sample are shown. They are measured using the CCD-device and the notch filter for decreasing laser intensities between $I_l = 1.4\,\mathrm{A}$ and $I_l = 1.0\,\mathrm{A}$. The intensity in arbitrary units is plotted against the wavelength λ in nm. For the Stokes and the Anti-Stokes peaks, a Voigt fit is performed and added to the graphic.

7.2 Python Code

7.2.1 CCD-Spectrometer Analysis

Raman

April 15, 2024

[ ]:
```
import matplotlib.pyplot as plt
import numpy as np
from scipy.special import voigt_profile
from scipy.optimize import curve_fit
import matplotlib.ticker as ticker
SMALL_SIZE = 20
MEDIUM_SIZE = 20
BIGGER_SIZE = 20
plt.rc('font', size=SMALL_SIZE) # controls default text sizes
plt.rc('axes', titlesize=MEDIUM_SIZE) # fontsize of the axes title
plt.rc('axes', labelsize=SMALL_SIZE) # fontsize of the x and y labels
plt.rc('xtick', labelsize=SMALL_SIZE) # fontsize of the tick labels
plt.rc('ytick', labelsize=SMALL_SIZE) # fontsize of the tick labels
plt.rc('legend', fontsize=SMALL_SIZE) # legend fontsize
plt.rc('figure', titlesize=BIGGER_SIZE) # fontsize of the figure title
import matplotlib as mpl
import matplotlib.font_manager as font_manager
mpl.rcParams['font.family']='serif'
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
```

[ ]:
```
def f(x, a, b):
    return a*x+b

def gauss(x, A, mu, sig_squared, y0):
    return A*np.exp(-(x-mu)**2/(2*sig_squared)) + y0

def voigt(x, A, mu, sig, gam, y0):
    return A*voigt_profile((x-mu), sig, gam)+y0

def chiq(y, y_err, mod_y):
    return sum([(j-i)**2/delj**2 for i,j,delj in zip(y, mod_y, y_err)])

def red_chiq(y, y_err, mod_y, par):
```

```
    return chiq(y, y_err, mod_y)/(len(y)-par)

def stdmw(liste):
    return 1/len(liste)*np.sqrt(sum([(i-np.mean(liste))**2 for i in liste]))

def Modell(xbereich, p):
    return [p[0]*i+p[1] for i in xbereich]

def Konfidenz(xbereich, p, Vp):
    des = [np.sqrt(np.matmul(np.transpose(np.array([i, 1])),np.matmul(Vp,np.
  ↪array([i, 1])))) for i in xbereich]
    return [[p[0]*i+p[1]+des[j] for i,j in␣
  ↪zip(xbereich,range(len(xbereich)))],[p[0]*i+p[1]-des[j] for i,j in␣
  ↪zip(xbereich,range(len(xbereich)))]]
```

0.1 Kalibrierung CCD

0.1.1 Einlesen und Background abziehen

[ ]:

```
import pandas as pd

# read files and calculate errors
def read(file, N):
    data = pd.read_table(file, header=17,decimal=',', encoding='latin1',␣
  ↪skipfooter=1, names=['channel', 'spectrum'],engine='python')
    channel = np.array(data.channel)
    spectrum = np.array(data.spectrum)
    delta_channel = 1/np.sqrt(3)
    delta_spectrum = np.sqrt(spectrum/N)
    return channel, delta_channel, spectrum, delta_spectrum

# read data, background; substract background and calculate error by consider␣
  ↪number of averaged data N
def background(file, back_file, N):
    channel_scan, delta_channel_scan, spectrum_scan, delta_spectrum_scan =␣
  ↪read(file, N)
    channel_back, delta_channel_back, spectrum_back, delta_spectrum_back =␣
  ↪read(back_file, N)
    spectrum = spectrum_scan-spectrum_back
    delta_spectrum = np.sqrt(delta_spectrum_scan**2+delta_spectrum_back**2)
    return channel_scan, delta_channel_scan, spectrum, delta_spectrum
```

0.1.2 Calibration

[ ]:

```
# data of calibration with different polarizations
N_hg = 1000
channel_hg_00, delta_channel_hg_00, spectrum_hg_00, delta_spectrum_hg_00 =␣
 ↪background('scan_hg_00.txt', 'background_hg_00.txt', N_hg)
channel_hg_90, delta_channel_hg_90, spectrum_hg_90, delta_spectrum_hg_90 =␣
 ↪background('scan_hg_90.txt', 'background_hg_90.txt', N_hg)
```

[ ]:

```
# fits for calibration for 0° polarization
#calib_window = [[110,170],[638,658], [1130,1270], [1525,1546],[1550,1572]]
calib_window = [[80,250],[610,690], [1110,1300], [1510,1546],[1550,1600]]
calib_energies = [435.83, 491.6,546.07,576.96,579.07]

plt.subplots(figsize=(16,9))
plt.semilogy()
plt.errorbar(channel_hg_00, spectrum_hg_00, xerr=delta_channel_hg_00,␣
 ↪yerr=delta_spectrum_hg_00, ecolor="black", marker="x", ls="", color="green",␣
 ↪markersize=5, capsize=3, label="measured spectrum")
for w in calib_window:
    plt.fill_betweenx([0,50000], w[0],w[1], color='lightgrey')
#plt.xlim(calib_window[2][0]-50,calib_window[2][1]+50)
plt.ylim(1, 50000)
plt.grid()
#plt.title("Hg spectrum with 0° polarization")
plt.xlabel("Channel $c$")
plt.ylabel("Intensity in [a.u.]")

calib_channels = []
delta_calib_channels = []
calib_chi = []

x1 = np.linspace(calib_window[0][0],calib_window[0][1],20000)
p1, V1 = curve_fit(voigt, channel_hg_00[calib_window[0][0]:calib_window[0][1]],␣
 ↪spectrum_hg_00[calib_window[0][0]:calib_window[0][1]],␣
 ↪p0=[200000,145,2,2,0], sigma=delta_spectrum_hg_00[calib_window[0][0]:
 ↪calib_window[0][1]], absolute_sigma=True, bounds=(0,np.inf))
plt.plot(x1, voigt(x1,p1[0],p1[1],p1[2],p1[3],p1[4]), label='fit 1. peak')
calib_channels.append(p1[1])
delta_calib_channels.append(np.sqrt(8*np.log(2)*p1[2]**2+4*p1[3]**2))
calib_chi.append(red_chiq(spectrum_hg_00[calib_window[0][0]:
 ↪calib_window[0][1]], delta_spectrum_hg_00[calib_window[0][0]:
 ↪calib_window[0][1]], voigt(channel_hg_00[calib_window[0][0]:
 ↪calib_window[0][1]],p1[0],p1[1],p1[2],p1[3],p1[4]), 5))

x2 = np.linspace(calib_window[1][0],calib_window[1][1],20000)
```

```
p2, V2 = curve_fit(voigt, channel_hg_00[calib_window[1][0]:calib_window[1][1]],␣
 ↪spectrum_hg_00[calib_window[1][0]:calib_window[1][1]],␣
 ↪p0=[200000,645,2,2,0], sigma=delta_spectrum_hg_00[calib_window[1][0]:
 ↪calib_window[1][1]], absolute_sigma=True, bounds=(0,np.inf))
plt.plot(x2, voigt(x2,p2[0],p2[1],p2[2],p2[3],p2[4]), label='fit 2. peak')
calib_channels.append(p2[1])
delta_calib_channels.append(np.sqrt(8*np.log(2)*p2[2]**2+4*p2[3]**2))
calib_chi.append(red_chiq(spectrum_hg_00[calib_window[1][0]:
 ↪calib_window[1][1]], delta_spectrum_hg_00[calib_window[1][0]:
 ↪calib_window[1][1]], voigt(channel_hg_00[calib_window[1][0]:
 ↪calib_window[1][1]],p2[0],p2[1],p2[2],p2[3],p2[4]), 5))


x3 = np.linspace(calib_window[2][0],calib_window[2][1],20000)
p3, V3 = curve_fit(voigt, channel_hg_00[calib_window[2][0]:calib_window[2][1]],␣
 ↪spectrum_hg_00[calib_window[2][0]:calib_window[2][1]],␣
 ↪p0=[200000,1200,2,2,0], sigma=delta_spectrum_hg_00[calib_window[2][0]:
 ↪calib_window[2][1]], absolute_sigma=True, bounds=(0,np.inf))
plt.plot(x3, voigt(x3,p3[0],p3[1],p3[2],p3[3],p3[4]), label='fit 3. peak')
calib_channels.append(p3[1])
delta_calib_channels.append(np.sqrt(8*np.log(2)*p3[2]**2+4*p3[3]**2))
calib_chi.append(red_chiq(spectrum_hg_00[calib_window[2][0]:
 ↪calib_window[2][1]], delta_spectrum_hg_00[calib_window[2][0]:
 ↪calib_window[2][1]], voigt(channel_hg_00[calib_window[2][0]:
 ↪calib_window[2][1]],p3[0],p3[1],p3[2],p3[3],p3[4]), 5))


x4 = np.linspace(calib_window[3][0],calib_window[3][1],20000)
p4, V4 = curve_fit(voigt, channel_hg_00[calib_window[3][0]:calib_window[3][1]],␣
 ↪spectrum_hg_00[calib_window[3][0]:calib_window[3][1]], p0=[5000,1500,2,2,0],␣
 ↪sigma=delta_spectrum_hg_00[calib_window[3][0]:calib_window[3][1]],␣
 ↪absolute_sigma=True, bounds=(0,np.inf))
plt.plot(x4, voigt(x4,p4[0],p4[1],p4[2],p4[3],p4[4]), label='fit 4. peak')
calib_channels.append(p4[1])
delta_calib_channels.append(np.sqrt(8*np.log(2)*p4[2]**2+4*p4[3]**2))
calib_chi.append(red_chiq(spectrum_hg_00[calib_window[3][0]:
 ↪calib_window[3][1]], delta_spectrum_hg_00[calib_window[3][0]:
 ↪calib_window[3][1]], voigt(channel_hg_00[calib_window[3][0]:
 ↪calib_window[3][1]],p4[0],p4[1],p4[2],p4[3],p4[4]), 5))


x5 = np.linspace(calib_window[4][0],calib_window[4][1],20000)
p5, V5 = curve_fit(voigt, channel_hg_00[calib_window[4][0]:calib_window[4][1]],␣
 ↪spectrum_hg_00[calib_window[4][0]:calib_window[4][1]],␣
 ↪p0=[200000,1200,2,2,0], sigma=delta_spectrum_hg_00[calib_window[4][0]:
 ↪calib_window[4][1]], absolute_sigma=True, bounds=(0,np.inf))
```

```
plt.plot(x5, voigt(x5, p5[0],p5[1],p5[2],p5[3],p5[4]), label='fit 5. peak')
calib_channels.append(p5[1])
delta_calib_channels.append(np.sqrt(8*np.log(2)*p5[2]**2+4*p5[3]**2))
calib_chi.append(red_chiq(spectrum_hg_00[calib_window[4][0]:
 ↪calib_window[4][1]], delta_spectrum_hg_00[calib_window[4][0]:
 ↪calib_window[4][1]], voigt(channel_hg_00[calib_window[4][0]:
 ↪calib_window[4][1]],p5[0],p5[1],p5[2],p5[3],p5[4]), 5))



plt.legend()
plt.savefig('Bilder/calibration_00_log_CCD.pdf', bbox_inches='tight')
#print(p1[3], p2[3], p3[3], p4[3], p5[3])
#print(np.sqrt(V1[3][3]), np.sqrt(V2[3][3]), np.sqrt(V3[3][3]), np.
 ↪sqrt(V4[3][3]), np.sqrt(V5[3][3]))
print(calib_channels, delta_calib_channels, calib_chi)
```

[ ]:

```
calib_p, calib_V = curve_fit(f, calib_channels, calib_energies)
x = np.linspace(0,1780,5000)
mod = Modell(x,calib_p)
konf = Konfidenz(x,calib_p,calib_V)

plt.subplots(figsize=(16,9))

plt.plot(x, mod, label="linear regression", color="darkolivegreen")
plt.plot(x, konf[0],linestyle="dashed", color="grey", label="confidence band")
plt.plot(x, konf[1],linestyle="dashed", color="grey")
plt.fill_between(x,konf[0],konf[1],color="whitesmoke")

plt.errorbar(calib_channels, calib_energies, xerr=delta_calib_channels,␣
 ↪ecolor="black", marker="x", ls="", color="green", markersize=5, capsize=3,␣
 ↪label="energy of Hg peaks")
plt.legend()
plt.grid()
plt.ylim(430, 600)
#plt.title("Calibration of the CCD spectrometer")
plt.xlabel("Channel $c$")
plt.ylabel('wavelength [nm]')
plt.savefig('Bilder/calibration_CCD.pdf', bbox_inches='tight')

print(calib_p, calib_V)

def calibration(channel, delta_channel=1/np.sqrt(3), p=calib_p, V=calib_V):
    return f(channel, p[0], p[1]), np.
 ↪sqrt(p[0]**2*delta_channel**2+channel**2*V[0][0]+V[1][1]+2*channel*V[1][0])

def spectrum2(spectrum_file, background_file, N):
```

```
    channel_scan, delta_channel_scan, spectrum_scan, delta_spectrum_scan =␣
  ↪read(spectrum_file, N)
    channel_back, delta_channel_back, spectrum_back, delta_spectrum_back =␣
  ↪read(background_file, N)
    spectrum = spectrum_scan-spectrum_back
    delta_spectrum = np.sqrt(delta_spectrum_scan**2+delta_spectrum_back**2)
    channel_scan, delta_channel_scan=calibration(channel_scan)
    return channel_scan, delta_channel_scan, spectrum, delta_spectrum
```

0.2 Spectrum of white light

[ ]:
```
channel_white_00, delta_channel_white_00, spectrum_white_00,␣
 ↪delta_spectrum_white_00 = background('scan_white_00.txt',␣
 ↪'background_white_00.txt', 100)
channel_white_00, delta_channel_white_00 = calibration(channel_white_00)
channel_white_90, delta_channel_white_90, spectrum_white_90,␣
 ↪delta_spectrum_white_90 = background('scan_white_90.txt',␣
 ↪'background_white_90.txt', 100)
channel_white_90, delta_channel_white_90 = calibration(channel_white_90)
```

[ ]:
```
plt.subplots(figsize=(16,9))
plt.errorbar(channel_white_00, spectrum_white_00, xerr=delta_channel_white_00,␣
 ↪yerr=delta_spectrum_white_00, ecolor="black", marker="o", ls="",␣
 ↪color="green", markersize=3, capsize=3, label="measured spectrum")
plt.grid()
#plt.title("spectrum of white light with 0° polarization")
plt.xlabel("Wavelength $\\lambda$ [nm]")
plt.ylabel("Intensity in [a.u.]")
plt.savefig('Bilder/white_00_CCD.pdf', bbox_inches='tight')
```

[ ]:
```
plt.subplots(figsize=(16,9))
plt.errorbar(channel_white_90, spectrum_white_90, xerr=delta_channel_white_90,␣
 ↪yerr=delta_spectrum_white_90, ecolor="black", marker="o", ls="",␣
 ↪color="green", markersize=3, capsize=3, label="measured spectrum")
plt.grid()
#plt.title("spectrum of white light with 90° polarization")
plt.xlabel("Wavelength $\\lambda$ [nm]")
plt.ylabel("Intensity in [a.u.]")
plt.savefig('Bilder/white_90_CCD.pdf', bbox_inches='tight')
```

[ ]:
```
channel_kerb_00, delta_channel_kerb_00, spectrum_kerb_00,␣
 ↪delta_spectrum_kerb_00 = background('scan_kerb_00.txt', 'background_white_00.
 ↪txt', 100)
channel_kerb_00, delta_channel_kerb_00 = calibration(channel_kerb_00)
```

```
channel_kerb_90, delta_channel_kerb_90, spectrum_kerb_90,␣
 ↪delta_spectrum_kerb_90 = background('scan_kerb_90.txt', 'background_white_90.
 ↪txt', 100)
channel_kerb_90, delta_channel_kerb_90 = calibration(channel_kerb_90)
```

```
[ ]: plt.subplots(figsize=(16,9))
plt.errorbar(channel_kerb_00, spectrum_kerb_00, xerr=delta_channel_kerb_00,␣
 ↪yerr=delta_spectrum_kerb_00, ecolor="black", marker="o", ls="",␣
 ↪color="green", markersize=3, capsize=3, label="measured spectrum")
plt.grid()
#plt.title("spectrum of white light with 0° polarization and kerb filter")
plt.xlabel("Wavelength $\\lambda$ [nm]")
plt.ylabel("Intensity in [a.u.]")
plt.savefig('Bilder/kerb_00_CCD.pdf', bbox_inches='tight')
```

```
[ ]: plt.subplots(figsize=(16,9))
plt.errorbar(channel_kerb_90, spectrum_kerb_90, xerr=delta_channel_kerb_90,␣
 ↪yerr=delta_spectrum_kerb_90, ecolor="black", marker="o", ls="",␣
 ↪color="green", markersize=3, capsize=3, label="measured spectrum")
plt.grid()
#plt.title("spectrum of white light with 90° polarization and kerb filter")
plt.xlabel("Wavelength $\\lambda$ [nm]")
plt.ylabel("Intensity in [a.u.]")
plt.savefig('Bilder/kerb_90_CCD.pdf', bbox_inches='tight')
```

0.2.1 Different Polarizations

```
[ ]: pol90zu00 = [i/j for i, j in zip(spectrum_white_90, spectrum_white_00)]
delta_pol90zu00 = [np.sqrt(di**2/j**2+i**2*dj**2/j**4) for i, j, di, dj in␣
 ↪zip(spectrum_white_90,spectrum_white_00,delta_spectrum_white_90,delta_spectrum_white_00)]

plt.subplots(figsize=(16,9))
#plt.semilogy()
plt.errorbar(channel_white_00, pol90zu00, xerr=delta_channel_white_00, yerr=[np.
 ↪sqrt(di**2/j**2+i**2*dj**2/j**4) for i, j, di, dj in␣
 ↪zip(spectrum_white_90,spectrum_white_00,delta_spectrum_white_90,delta_spectrum_white_00)],␣
 ↪ecolor="black", marker="x", ls="", color="green", markersize=5, capsize=3,␣
 ↪label="measured spectrum")
plt.grid()
#plt.title("spectrum of white light with 0° polarization")
plt.ylim(0,2.5)
plt.xlabel("Wavelength $\\lambda$ [nm]")
plt.ylabel("Intensity in [a.u.]")
plt.savefig('Bilder/white_polarizations_CCD.pdf', bbox_inches='tight')
```

```
[ ]:
```

```
[ ]:
```

0.3 Laser Wellenlänge

```
[ ]: laser_current = [0.7,0.6,0.5,0.4, 0.35,0.3,0.37,0.33,0.5,0.6,0.7,0.8,0.65,0.75]
     N_laser = 10000
     laser_channel = []
     delta_laser_channel = []
     laser_spectrum = []
     delta_laser_spectrum = []

     for i in␣
      ↪['00','01','02','03','04','05','06','07','08','09','10','11','12','13']:
         c, dc, s, ds = read('laser_'+i+'.txt', N_laser)
         c, dc = calibration(c)
         laser_channel.append(c)
         delta_laser_channel.append(dc)
         laser_spectrum.append(s)
         delta_laser_spectrum.append(ds)
```

```
[ ]: plt.subplots(figsize=(16,9))
     plt.semilogy()
     plt.errorbar(laser_channel[3], laser_spectrum[3], xerr=delta_laser_channel[3],␣
      ↪yerr=delta_laser_spectrum[3], ecolor="black", marker="x", ls="",␣
      ↪markersize=10, capsize=5, label=f"Laser spectrum for I={laser_current[3]} A")
     plt.xlim(laser_channel[3][900], laser_channel[3][1200])
     plt.ylim(1000, 80000)
     plt.legend()
     plt.grid()
     #plt.title("laser spectrum for 0° polarization and different laser currents")
     plt.xlabel("wavelength [nm]")
     plt.ylabel("Intensity [a.u.]")
     plt.savefig('Bilder/laser_xerr_CCD.pdf', bbox_inches='tight')
```

```
[ ]: plot_list_00 = [3, 4, 6, 7]

     lower_00 = 900
     upper_00 = 1200

     laser_p_00 = []
     laser_V_00 = []
     laser_chi_00 = []

     plt.subplots(figsize=(16,9))
     plt.semilogy()
     for i in plot_list_00:
```

```
    lp, lV = curve_fit(voigt, laser_channel[i][lower_00:upper_00],␣
 ↪laser_spectrum[i][lower_00:upper_00], p0 = [10000, 529, 0.5, 0.0001, 2000],␣
 ↪bounds=(0,np.inf))
    laser_p_00.append(lp)
    laser_V_00.append(lV)
    laser_chi_00.append(red_chiq(laser_spectrum[i][lower_00:upper_00],␣
 ↪delta_laser_spectrum[i][lower_00:upper_00], voigt(laser_channel[i][lower_00:
 ↪upper_00],lp[0],lp[1],lp[2],lp[3],lp[4]), 5))
    plt.plot(laser_channel[i],␣
 ↪voigt(laser_channel[i],lp[0],lp[1],lp[2],lp[3],lp[4]))
    plt.errorbar(laser_channel[i], laser_spectrum[i],␣
 ↪yerr=delta_laser_spectrum[i], ecolor="black", marker="x", ls="",␣
 ↪markersize=10, capsize=5, label=f"Laser spectrum for I={laser_current[i]} A")

plt.fill_betweenx([1,1e6],␣
 ↪laser_channel[0][lower_00],laser_channel[0][upper_00], color='lightgrey')
plt.xlim(laser_channel[0][lower_00]-1, laser_channel[0][upper_00]+1)
plt.ylim(1000, 80000)
plt.legend()
plt.grid()
#plt.title("laser spectrum for 0° polarization and different laser currents")
plt.xlabel("wavelength [nm]")
plt.ylabel("Intensity [a.u.]")
plt.savefig('Bilder/laser_00_CCD.pdf', bbox_inches='tight')

#print(laser_p_00, laser_V_00, laser_chi_00)
```

```
[ ]: plot_list_90 = [8, 9, 10, 12, 13]

lower_90 = 900
upper_90 = 1200

laser_p_90 = []
laser_V_90 = []
laser_chi_90 = []

plt.subplots(figsize=(16,9))
plt.semilogy()
for i in plot_list_90:
    lp, lV = curve_fit(voigt, laser_channel[i][lower_90:upper_90],␣
 ↪laser_spectrum[i][lower_90:upper_90], p0 = [10000, 529, 0.5, 10, 2000],␣
 ↪bounds=(0,np.inf))#curve_fit(gauss, laser_channel[i][lower_90:upper_90],␣
 ↪laser_spectrum[i][lower_90:upper_90], p0 = [100000, 529, 1, 10000],␣
 ↪bounds=(0,np.inf)) ##
    laser_p_90.append(lp)
    laser_V_90.append(lV)
```

```
        laser_chi_90.append(red_chiq(laser_spectrum[i][lower_90:upper_90],␣
    ↪delta_laser_spectrum[i][lower_90:upper_90], voigt(laser_channel[i][lower_90:
    ↪upper_90],lp[0],lp[1],lp[2],lp[3],lp[4]), 5))
        plt.plot(laser_channel[i],␣
    ↪voigt(laser_channel[i],lp[0],lp[1],lp[2],lp[3],lp[4]))
        plt.errorbar(laser_channel[i], laser_spectrum[i],␣
    ↪yerr=delta_laser_spectrum[i], ecolor="black", marker="x", ls="",␣
    ↪markersize=10, capsize=5, label=f"Laser spectrum for I={laser_current[i]} A")
plt.fill_betweenx([1e3, 1e5], laser_channel[0][lower_90],␣
    ↪laser_channel[0][upper_90], color='lightgrey')
plt.xlim(laser_channel[0][lower_90]-1, laser_channel[0][upper_90]+1)
plt.ylim(1e3, 80000)
plt.legend()
plt.grid()
#plt.title("laser spectrum for 90° polarization and different laser currents")
plt.xlabel("wavelength [nm]")
plt.ylabel("Intensity [a.u.]")
plt.savefig('Bilder/laser_90_CCD.pdf', bbox_inches='tight')

print(laser_p_90)
```

```
[ ]: laser = np.mean([i[1] for i in laser_p_00]+[i[1] for i in laser_p_90])
     delta_laser = stdmw([i[1] for i in laser_p_00]+[i[1] for i in laser_p_90])
     laser, delta_laser
```

0.4 Test of different settings

0.4.1 schnelle Auswertung (Einlesen, background abziehen und Plotten)

```
[ ]: def spectrum(spectrum_file, background_file, N, x=[0,0], y=[0,0]):
         channel_scan, delta_channel_scan, spectrum_scan, delta_spectrum_scan =␣
       ↪read(spectrum_file, N)
         channel_back, delta_channel_back, spectrum_back, delta_spectrum_back =␣
       ↪read(background_file, N)
         spectrum = spectrum_scan-spectrum_back
         delta_spectrum = np.sqrt(delta_spectrum_scan**2+delta_spectrum_back**2)
         channel_scan, delta_channel_scan=calibration(channel_scan)
         plt.subplots(figsize=(16,9))
         plt.plot(channel_scan, spectrum)
         if x!=[0,0]:
             plt.xlim(x)
         if y!=[0,0]:
             plt.ylim(y)
         plt.grid()
         plt.xlabel("wavelength $\\lambda$")
         plt.ylabel("Intensity in [a.u.]")
         plt.show()
```

```
    return channel_scan, delta_channel_scan, spectrum, delta_spectrum
```

0.5 Spectrum of white light

```
[ ]: channel_white_00, delta_channel_white_00, spectrum_white_00,␣
      ↪delta_spectrum_white_00 = background('scan_white_00.txt',␣
      ↪'background_white_00.txt', 100)
     channel_white_00, delta_channel_white_00 = calibration(channel_white_00)
     channel_white_90, delta_channel_white_90, spectrum_white_90,␣
      ↪delta_spectrum_white_90 = background('scan_white_90.txt',␣
      ↪'background_white_90.txt', 100)
     channel_white_90, delta_channel_white_90 = calibration(channel_white_90)
```

```
[ ]: plt.subplots(figsize=(16,9))
     plt.errorbar(channel_white_00, spectrum_white_00, xerr=delta_channel_white_00,␣
      ↪yerr=delta_spectrum_white_00, ecolor="black", marker="o", ls="",␣
      ↪color="green", markersize=3, capsize=3, label="measured spectrum")
     plt.grid()
     #plt.title("spectrum of white light with 0° polarization")
     plt.xlabel("Wavelength $\\lambda$ [nm]")
     plt.ylabel("Intensity in [a.u.]")
     plt.savefig('Bilder/white_00_CCD.pdf', bbox_inches='tight')
```

```
[ ]: plt.subplots(figsize=(16,9))
     plt.errorbar(channel_white_90, spectrum_white_90, xerr=delta_channel_white_90,␣
      ↪yerr=delta_spectrum_white_90, ecolor="black", marker="o", ls="",␣
      ↪color="green", markersize=3, capsize=3, label="measured spectrum")
     plt.grid()
     #plt.title("spectrum of white light with 90° polarization")
     plt.xlabel("Wavelength $\\lambda$ [nm]")
     plt.ylabel("Intensity in [a.u.]")
     plt.savefig('Bilder/white_90_CCD.pdf', bbox_inches='tight')
```

```
[ ]: channel_kerb_00, delta_channel_kerb_00, spectrum_kerb_00,␣
      ↪delta_spectrum_kerb_00 = background('scan_kerb_00.txt', 'background_white_00.
      ↪txt', 100)
     channel_kerb_00, delta_channel_kerb_00 = calibration(channel_kerb_00)
     channel_kerb_90, delta_channel_kerb_90, spectrum_kerb_90,␣
      ↪delta_spectrum_kerb_90 = background('scan_kerb_90.txt', 'background_white_90.
      ↪txt', 100)
     channel_kerb_90, delta_channel_kerb_90 = calibration(channel_kerb_90)
```

```
[ ]: plt.subplots(figsize=(16,9))
     plt.errorbar(channel_kerb_00, spectrum_kerb_00, xerr=delta_channel_kerb_00,␣
      ↪yerr=delta_spectrum_kerb_00, ecolor="black", marker="o", ls="",␣
      ↪color="green", markersize=3, capsize=3, label="measured spectrum")
     plt.grid()
```

```
#plt.title("spectrum of white light with 0° polarization and kerb filter")
plt.xlabel("Wavelength $\\lambda$ [nm]")
plt.ylabel("Intensity in [a.u.]")
plt.savefig('Bilder/kerb_00_CCD.pdf', bbox_inches='tight')
```

[ ]:

```
plt.subplots(figsize=(16,9))
plt.errorbar(channel_kerb_90, spectrum_kerb_90, xerr=delta_channel_kerb_90,␣
 ↪yerr=delta_spectrum_kerb_90, ecolor="black", marker="o", ls="",␣
 ↪color="green", markersize=3, capsize=3, label="measured spectrum")
plt.grid()
#plt.title("spectrum of white light with 90° polarization and kerb filter")
plt.xlabel("Wavelength $\\lambda$ [nm]")
plt.ylabel("Intensity in [a.u.]")
plt.savefig('Bilder/kerb_90_CCD.pdf', bbox_inches='tight')
```

0.5.1 Different Polarizations

[ ]:

```
pol90zu00 = [i/j for i, j in zip(spectrum_white_90, spectrum_white_00)]
delta_pol90zu00 = [np.sqrt(di**2/j**2+i**2*dj**2/j**4) for i, j, di, dj in␣
 ↪zip(spectrum_white_90,spectrum_white_00,delta_spectrum_white_90,delta_spectrum_white_00)]


plt.subplots(figsize=(16,9))

plt.errorbar(channel_white_90, pol90zu00, yerr=delta_pol90zu00, ecolor="black",␣
 ↪marker="x", ls="", alpha=0.5, color="green", markersize=5, capsize=3,␣
 ↪label='ratio of polarizations')

#plt.semilogy()
w=[757, 1400]
plt.fill_betweenx([0,2.5], channel_white_90[w[0]], channel_white_90[w[1]],␣
 ↪color='lightgrey')
pol_p, pol_V = curve_fit(f, channel_white_90[w[0]:w[1]], pol90zu00[w[0]:w[1]])
x = np.linspace(500,560,20000)
mod = Modell(x,pol_p)
konf = Konfidenz(x,pol_p,pol_V)
plt.plot(x, mod, label="linear regression", color="red")
plt.plot(x, konf[0],linestyle="dashed", color="grey", label="confidence band")
plt.plot(x, konf[1],linestyle="dashed", color="grey")
plt.fill_between(x,konf[0],konf[1],color="whitesmoke")
def pol(l, dl):
    return l*pol_p[0]+pol_p[1], np.
 ↪sqrt(dl**2*pol_p[0]**2+l**2*pol_V[0][0]+pol_V[1][1]+2*l*pol_V[1][0])

plt.legend()
plt.grid()
plt.ylim(0,2.5)
```

```
plt.xlabel("wavelength $\\lambda$ [nm]")
plt.ylabel("Intensity ratio of 90° to 0° polarization $r$")

plt.savefig('Bilder/white_polarizations_CCD.pdf', bbox_inches='tight')
pol_p, pol_V
```

[ ]:

```
```

[ ]:

```
```

0.6 Evaluation of Raman spectrum

0.6.1 Fit the Rayleigh peak and substract it

[ ]:

```
def spectrum2(spectrum_file, background_file, N):
    channel_scan, delta_channel_scan, spectrum_scan, delta_spectrum_scan =␣
 ↪read(spectrum_file, N)
    channel_back, delta_channel_back, spectrum_back, delta_spectrum_back =␣
 ↪read(background_file, N)
    spectrum = spectrum_scan-spectrum_back
    delta_spectrum = np.sqrt(delta_spectrum_scan**2+delta_spectrum_back**2)
    channel_scan, delta_channel_scan=calibration(channel_scan)
    return channel_scan, delta_channel_scan, spectrum, delta_spectrum

def substract_fit(channel, delta_channel, spectrum, delta_spectrum, fit_window,␣
 ↪guess=[100000,529,2,2,0], plot=False):
    if plot:
        plt.subplots(figsize=(16,9))
        plt.errorbar(list(channel[0:fit_window[0]])+list(channel[fit_window[1]:
 ↪-1]), list(spectrum[0:fit_window[0]])+list(spectrum[fit_window[1]:-1]),␣
 ↪ecolor="black", marker="x", ls="", color="green", markersize=3, capsize=3,␣
 ↪label="spectrum")
        plt.
 ↪vlines([fit_channel[fit_window[0]]]+[fit_channel[fit_window[1]]],0,70000)
        plt.legend()
        plt.xlim(fit_channel[fit_window[0]]-5,fit_channel[fit_window[1]]+5)
        #plt.ylim(60000,70000)
        plt.grid()
        plt.xlabel("wavelength $\\lambda$ [nm]")
        plt.ylabel("Intensity in [a.u.]")
    fp, fV = curve_fit(voigt, list(channel[0:
 ↪fit_window[0]])+list(channel[fit_window[1]:-1]), list(spectrum[0:
 ↪fit_window[0]])+list(spectrum[fit_window[1]:-1]), p0=guess,␣
 ↪sigma=list(delta_spectrum[0:
 ↪fit_window[0]])+list(delta_spectrum[fit_window[1]:-1]), absolute_sigma=True,␣
 ↪bounds=(0,np.inf))
```

```
    return list(channel[0:fit_window[0]])+list(channel[fit_window[1]:-1]),␣
  ↪[i-voigt(j,fp[0],fp[1],fp[2],fp[3],fp[4]) for i, j in zip(list(spectrum[0:
  ↪fit_window[0]])+list(spectrum[fit_window[1]:-1]),list(channel[0:
  ↪fit_window[0]])+list(channel[fit_window[1]:-1]))], fp, fV

def energy(l, dl): #gives energy in cm^-1
    return 10**7/l, (10**7/l)**2*dl*10**(-7)
    #return 1.973*10**(-7)/(l*10**(-9)), (1.973*10**(-7)/
  ↪(l*10**(-9)))**2*dl*10**(-9)/(1.973*10**(-7))
```

```
[ ]: laser_energy, delta_laser_energy = energy(laser, delta_laser)
     laser_energy, delta_laser_energy
```

```
[ ]: lit_CCl4 = np.array([217,314,459,776])
     lit_dep_CCl4 = np.array([])
     lit_CHCl3 = np.array([260,364,676,760])
     lit_CH2Cl2 = np.array([])
     lit_CS2 = np.array([397, 658, 1535])
     def percm(lit): #converts per cm in eV
         return lit*1.973*10**(-7)/(10**(-2))
     print(lit_CCl4)
```

0.6.2 CCl4

1.

```
[ ]: window_CCl4_1 = [1043, 1052]
     channel_CCl4_1, delta_channel_CCl4_1, spectrum_CCl4_1, delta_spectrum_CCl4_1 =␣
       ↪spectrum2("measure_CCl4_2.txt", "measure_CCl4_1.txt", 100)
     plaser_CCl4_1, Vlaser_CCl4_1 = curve_fit(voigt, list(channel_CCl4_1[0:
       ↪window_CCl4_1[0]])+list(channel_CCl4_1[window_CCl4_1[1]:-1]),␣
       ↪list(spectrum_CCl4_1[0:
       ↪window_CCl4_1[0]])+list(spectrum_CCl4_1[window_CCl4_1[1]:-1]),␣
       ↪p0=[100000,529,2,2,0], sigma=list(delta_spectrum_CCl4_1[0:
       ↪window_CCl4_1[0]])+list(delta_spectrum_CCl4_1[window_CCl4_1[1]:-1]),␣
       ↪absolute_sigma=True, bounds=(0,np.inf))

     xlaser=np.linspace(500, 560, 5000)
     plt.subplots(figsize=(16,9))
     plt.errorbar(channel_CCl4_1, spectrum_CCl4_1, yerr=delta_spectrum_CCl4_1,␣
       ↪ecolor="black", marker="x", ls="", color="green", markersize=5, capsize=3,␣
       ↪label="spectrum")
     plt.plot(xlaser,␣
       ↪voigt(xlaser,plaser_CCl4_1[0],plaser_CCl4_1[1],plaser_CCl4_1[2],plaser_CCl4_1[3],plaser_CCl

     plt.legend()
     #plt.title('Raman spectrum of CCl$_4$ with 90° polarization, without $\\lambda/
       ↪2$-plate\n Rayleigh fit substracted')
```

```
plt.xlim(500,560)
#plt.xlim(calibration(i[0])[0]-5,calibration(i[1])[0]+5)
plt.ylim(0, 70000)
plt.grid()
plt.xlabel("wavelength $\\lambda$ [nm]")
plt.ylabel("Intensity in [a.u.]")
plt.savefig('Bilder/Raman_CCl4_1_sub_CCD.pdf', bbox_inches='tight')
```

[ ]:

```
window_CCl4_1 = [1043, 1052]
channel_CCl4_1, delta_channel_CCl4_1, spectrum_CCl4_1, delta_spectrum_CCl4_1 =␣
 ↪spectrum2("measure_CCl4_2.txt", "measure_CCl4_1.txt", 100)
sub_channel_CCl4_1, sub_spectrum_CCl4_1, p_CCl4_1, V_CCl4_1 =␣
 ↪substract_fit(channel_CCl4_1, delta_channel_CCl4_1, spectrum_CCl4_1,␣
 ↪delta_spectrum_CCl4_1, window_CCl4_1)

fit_window_CCl4_1 = [[920,944],[949,968],[974, 994], [1090, 1115],[1120,1146],␣
 ↪[1164,1188], [1255,1300]]
CCl4_positions = [932,960,984,1103,1133,1176,1280]

plt.subplots(figsize=(16,9))
plt.errorbar(sub_channel_CCl4_1, sub_spectrum_CCl4_1,␣
 ↪yerr=list(delta_spectrum_CCl4_1[0:
 ↪window_CCl4_1[0]])+list(delta_spectrum_CCl4_1[window_CCl4_1[1]:-1]),␣
 ↪ecolor="black", marker="x", ls="", color="green", markersize=5, capsize=3,␣
 ↪label="spectrum")
for i in fit_window_CCl4_1:
    plt.fill_betweenx([0,70000], sub_channel_CCl4_1[i[0]],␣
 ↪sub_channel_CCl4_1[i[1]], color='lightgrey')
plt.legend()
#plt.title('Raman spectrum of CCl$_4$ with 90° polarization, without $\\lambda/
 ↪2$-plate\n Rayleigh fit substracted')
plt.xlim(500,560)
#plt.xlim(calibration(i[0])[0]-5,calibration(i[1])[0]+5)
plt.ylim(0, 1500)
plt.grid()
plt.xlabel("wavelength $\\lambda$ [nm]")
plt.ylabel("Intensity in [a.u.]")

x_0 = np.linspace(sub_channel_CCl4_1[fit_window_CCl4_1[0][0]],␣
 ↪sub_channel_CCl4_1[fit_window_CCl4_1[0][1]], 1000)
p_CCl4_1_0, V_CCl4_1_0 = curve_fit(voigt,␣
 ↪sub_channel_CCl4_1[fit_window_CCl4_1[0][0]:fit_window_CCl4_1[0][1]],␣
 ↪sub_spectrum_CCl4_1[fit_window_CCl4_1[0][0]:fit_window_CCl4_1[0][1]],␣
 ↪p0=[100,517,0.1,0.1,20], bounds=(0,np.inf))
plt.plot(x_0,␣
 ↪voigt(x_0,p_CCl4_1_0[0],p_CCl4_1_0[1],p_CCl4_1_0[2],p_CCl4_1_0[3],p_CCl4_1_0[4]))
```

```
x_1 = np.linspace(sub_channel_CCl4_1[fit_window_CCl4_1[1][0]],␣
 ↪sub_channel_CCl4_1[fit_window_CCl4_1[1][1]], 1000)
p_CCl4_1_1, V_CCl4_1_1 = curve_fit(voigt,␣
 ↪sub_channel_CCl4_1[fit_window_CCl4_1[1][0]:fit_window_CCl4_1[1][1]],␣
 ↪sub_spectrum_CCl4_1[fit_window_CCl4_1[1][0]:fit_window_CCl4_1[1][1]],␣
 ↪p0=[100,520,0.1,0.1,20], bounds=(0,np.inf))
plt.plot(x_1,␣
 ↪voigt(x_1,p_CCl4_1_1[0],p_CCl4_1_1[1],p_CCl4_1_1[2],p_CCl4_1_1[3],p_CCl4_1_1[4]))

x_2 = np.linspace(sub_channel_CCl4_1[fit_window_CCl4_1[2][0]],␣
 ↪sub_channel_CCl4_1[fit_window_CCl4_1[2][1]], 1000)
p_CCl4_1_2, V_CCl4_1_2 = curve_fit(voigt,␣
 ↪sub_channel_CCl4_1[fit_window_CCl4_1[2][0]:fit_window_CCl4_1[2][1]],␣
 ↪sub_spectrum_CCl4_1[fit_window_CCl4_1[2][0]:fit_window_CCl4_1[2][1]],␣
 ↪p0=[100,523,0.1,0.1,20], bounds=(0,np.inf))
plt.plot(x_2, voigt(x_2,␣
 ↪p_CCl4_1_2[0],p_CCl4_1_2[1],p_CCl4_1_2[2],p_CCl4_1_2[3],p_CCl4_1_2[4]))

x_3 = np.linspace(sub_channel_CCl4_1[fit_window_CCl4_1[3][0]],␣
 ↪sub_channel_CCl4_1[fit_window_CCl4_1[3][1]], 1000)
p_CCl4_1_3, V_CCl4_1_3 = curve_fit(voigt,␣
 ↪sub_channel_CCl4_1[fit_window_CCl4_1[3][0]:fit_window_CCl4_1[3][1]],␣
 ↪sub_spectrum_CCl4_1[fit_window_CCl4_1[3][0]:fit_window_CCl4_1[3][1]],␣
 ↪p0=[100,536,0.1,0.1,20], bounds=(0,np.inf))
plt.plot(x_3,␣
 ↪voigt(x_3,p_CCl4_1_3[0],p_CCl4_1_3[1],p_CCl4_1_3[2],p_CCl4_1_3[3],p_CCl4_1_3[4]))

x_4 = np.linspace(sub_channel_CCl4_1[fit_window_CCl4_1[4][0]],␣
 ↪sub_channel_CCl4_1[fit_window_CCl4_1[4][1]], 1000)
p_CCl4_1_4, V_CCl4_1_4 = curve_fit(voigt,␣
 ↪sub_channel_CCl4_1[fit_window_CCl4_1[4][0]:fit_window_CCl4_1[4][1]],␣
 ↪sub_spectrum_CCl4_1[fit_window_CCl4_1[4][0]:fit_window_CCl4_1[4][1]],␣
 ↪p0=[100,539,0.1,0.1,20], bounds=(0,np.inf))
plt.plot(x_4, voigt(x_4, p_CCl4_1_4[0], p_CCl4_1_4[1],␣
 ↪p_CCl4_1_4[2],p_CCl4_1_4[3],p_CCl4_1_4[4]))

x_5 = np.linspace(sub_channel_CCl4_1[fit_window_CCl4_1[5][0]],␣
 ↪sub_channel_CCl4_1[fit_window_CCl4_1[5][1]], 1000)
p_CCl4_1_5, V_CCl4_1_5 = curve_fit(voigt,␣
 ↪sub_channel_CCl4_1[fit_window_CCl4_1[5][0]:fit_window_CCl4_1[5][1]],␣
 ↪sub_spectrum_CCl4_1[fit_window_CCl4_1[5][0]:fit_window_CCl4_1[5][1]],␣
 ↪p0=[100,543,0.1,0.1,20], bounds=(0,np.inf))
plt.plot(x_5,␣
 ↪voigt(x_5,p_CCl4_1_5[0],p_CCl4_1_5[1],p_CCl4_1_5[2],p_CCl4_1_5[3],p_CCl4_1_5[4]))
```

```
x_6 = np.linspace(sub_channel_CCl4_1[fit_window_CCl4_1[6][0]],␣
 ↪sub_channel_CCl4_1[fit_window_CCl4_1[6][1]], 1000)
p_CCl4_1_6, V_CCl4_1_6 = curve_fit(voigt,␣
 ↪sub_channel_CCl4_1[fit_window_CCl4_1[6][0]:fit_window_CCl4_1[6][1]],␣
 ↪sub_spectrum_CCl4_1[fit_window_CCl4_1[6][0]:fit_window_CCl4_1[6][1]],␣
 ↪p0=[100,553,0.1,0.1,20], bounds=(0,np.inf))
plt.plot(x_6,␣
 ↪voigt(x_6,p_CCl4_1_6[0],p_CCl4_1_6[1],p_CCl4_1_6[2],p_CCl4_1_6[3],p_CCl4_1_6[4]))

plt.savefig('Bilder/Raman_CCl4_1_CCD.pdf', bbox_inches='tight')
```

```
[ ]:
```

```
[ ]: peak_energy_CCl4_1_0, delta_energy_CCl4_1_0 = energy(p_CCl4_1_0[1], np.
      ↪sqrt(8*np.log(2)*p_CCl4_1_0[2]**2+4*p_CCl4_1_0[3]**2))
     nu_CCl4_1_0, delta_nu_CCl4_1_0 = peak_energy_CCl4_1_0-laser_energy, np.
      ↪sqrt(delta_energy_CCl4_1_0**2+delta_laser_energy**2)

     peak_energy_CCl4_1_1, delta_energy_CCl4_1_1 = energy(p_CCl4_1_1[1], np.
      ↪sqrt(8*np.log(2)*p_CCl4_1_1[2]**2+4*p_CCl4_1_1[3]**2))
     nu_CCl4_1_1, delta_nu_CCl4_1_1 = peak_energy_CCl4_1_1-laser_energy, np.
      ↪sqrt(delta_energy_CCl4_1_1**2+delta_laser_energy**2)

     peak_energy_CCl4_1_2, delta_energy_CCl4_1_2 = energy(p_CCl4_1_2[1], np.
      ↪sqrt(8*np.log(2)*p_CCl4_1_2[2]**2+4*p_CCl4_1_2[3]**2))
     nu_CCl4_1_2, delta_nu_CCl4_1_2 = peak_energy_CCl4_1_2-laser_energy, np.
      ↪sqrt(delta_energy_CCl4_1_2**2+delta_laser_energy**2)

     peak_energy_CCl4_1_3, delta_energy_CCl4_1_3 = energy(p_CCl4_1_3[1], np.
      ↪sqrt(8*np.log(2)*p_CCl4_1_3[2]**2+4*p_CCl4_1_3[3]**2))
     nu_CCl4_1_3, delta_nu_CCl4_1_3 = peak_energy_CCl4_1_3-laser_energy, np.
      ↪sqrt(delta_energy_CCl4_1_3**2+delta_laser_energy**2)

     peak_energy_CCl4_1_4, delta_energy_CCl4_1_4 = energy(p_CCl4_1_4[1], np.
      ↪sqrt(8*np.log(2)*p_CCl4_1_4[2]**2+4*p_CCl4_1_4[3]**2))
     nu_CCl4_1_4, delta_nu_CCl4_1_4 = peak_energy_CCl4_1_4-laser_energy, np.
      ↪sqrt(delta_energy_CCl4_1_4**2+delta_laser_energy**2)

     peak_energy_CCl4_1_5, delta_energy_CCl4_1_5 = energy(p_CCl4_1_5[1], np.
      ↪sqrt(8*np.log(2)*p_CCl4_1_5[2]**2+4*p_CCl4_1_5[3]**2))
     nu_CCl4_1_5, delta_nu_CCl4_1_5 = peak_energy_CCl4_1_5-laser_energy, np.
      ↪sqrt(delta_energy_CCl4_1_5**2+delta_laser_energy**2)

     peak_energy_CCl4_1_6, delta_energy_CCl4_1_6 = energy(p_CCl4_1_6[1], np.
      ↪sqrt(8*np.log(2)*p_CCl4_1_6[2]**2+4*p_CCl4_1_6[3]**2))
```

```
nu_CCl4_1_6, delta_nu_CCl4_1_6 = peak_energy_CCl4_1_6-laser_energy, np.
 ↪sqrt(delta_energy_CCl4_1_6**2+delta_laser_energy**2)

print(nu_CCl4_1_0, delta_nu_CCl4_1_0)
print(nu_CCl4_1_1, delta_nu_CCl4_1_1)
print(nu_CCl4_1_2, delta_nu_CCl4_1_2)
print(nu_CCl4_1_3, delta_nu_CCl4_1_3)
print(nu_CCl4_1_4, delta_nu_CCl4_1_4)
print(nu_CCl4_1_5, delta_nu_CCl4_1_5)
print(nu_CCl4_1_6, delta_nu_CCl4_1_6)

print((nu_CCl4_1_0-lit_CCl4[2])/delta_nu_CCl4_1_0)
print((nu_CCl4_1_1-lit_CCl4[1])/delta_nu_CCl4_1_1)
print((nu_CCl4_1_2-lit_CCl4[0])/delta_nu_CCl4_1_2)
print((nu_CCl4_1_3+lit_CCl4[0])/delta_nu_CCl4_1_3)
print((nu_CCl4_1_4+lit_CCl4[1])/delta_nu_CCl4_1_4)
print((nu_CCl4_1_5+lit_CCl4[2])/delta_nu_CCl4_1_5)
print((nu_CCl4_1_6+lit_CCl4[3])/delta_nu_CCl4_1_6)
```

2.

[ ]:

```
window_CCl4_2 = [1043, 1052]
channel_CCl4_2, delta_channel_CCl4_2, spectrum_CCl4_2, delta_spectrum_CCl4_2 =␣
 ↪spectrum2("measure_CCl4_3.txt", "measure_CCl4_1.txt", 100)
sub_channel_CCl4_2, sub_spectrum_CCl4_2, p_CCl4_2, V_CCl4_2 =␣
 ↪substract_fit(channel_CCl4_2, delta_channel_CCl4_2, spectrum_CCl4_2,␣
 ↪delta_spectrum_CCl4_2, window_CCl4_2)

fit_window_CCl4_2 = [[920,944],[949,968],[974, 994], [1090, 1115],[1120,1150],␣
 ↪[1164,1188], [1255,1300]]

plt.subplots(figsize=(16,9))
plt.errorbar(sub_channel_CCl4_2,␣
 ↪sub_spectrum_CCl4_2,yerr=list(delta_spectrum_CCl4_2[0:
 ↪window_CCl4_2[0]])+list(delta_spectrum_CCl4_2[window_CCl4_2[1]:-1]),␣
 ↪ecolor="black", marker="x", ls="", color="green", markersize=5, capsize=3,␣
 ↪label="spectrum")
for i in fit_window_CCl4_2:
    plt.fill_betweenx([0,70000],sub_channel_CCl4_2[i[0]],␣
 ↪sub_channel_CCl4_2[i[1]],color='lightgrey')
plt.legend()
#plt.title('Raman spectrum of CCl$_4$ with 90° polarization, with $\\lambda/
 ↪2$-plate')
plt.xlim(500,560)
#plt.xlim(calibration(i[0])[0]-5,calibration(i[1])[0]+5)
plt.ylim(0, 1500)
plt.grid()
```

```
plt.xlabel("wavelength $\\lambda$ [nm]")
plt.ylabel("Intensity in [a.u.]")


x_0 = np.linspace(sub_channel_CCl4_2[fit_window_CCl4_2[0][0]],␣
 ↪sub_channel_CCl4_2[fit_window_CCl4_2[0][1]], 1000)
p_CCl4_2_0, V_CCl4_2_0 = curve_fit(voigt,␣
 ↪sub_channel_CCl4_2[fit_window_CCl4_2[0][0]:fit_window_CCl4_2[0][1]],␣
 ↪sub_spectrum_CCl4_2[fit_window_CCl4_2[0][0]:fit_window_CCl4_2[0][1]],␣
 ↪p0=[100,517,0.1,0.1,20], bounds=(0,np.inf))
plt.plot(x_0,␣
 ↪voigt(x_0,p_CCl4_2_0[0],p_CCl4_2_0[1],p_CCl4_2_0[2],p_CCl4_2_0[3],p_CCl4_2_0[4]))

x_1 = np.linspace(sub_channel_CCl4_2[fit_window_CCl4_2[1][0]],␣
 ↪sub_channel_CCl4_2[fit_window_CCl4_2[1][1]], 1000)
p_CCl4_2_1, V_CCl4_2_1 = curve_fit(voigt,␣
 ↪sub_channel_CCl4_2[fit_window_CCl4_2[1][0]:fit_window_CCl4_2[1][1]],␣
 ↪sub_spectrum_CCl4_2[fit_window_CCl4_2[1][0]:fit_window_CCl4_2[1][1]],␣
 ↪p0=[100,520,0.1,0.1,20], bounds=(0,np.inf))
plt.plot(x_1,␣
 ↪voigt(x_1,p_CCl4_2_1[0],p_CCl4_2_1[1],p_CCl4_2_1[2],p_CCl4_2_1[3],p_CCl4_2_1[4]))

x_2 = np.linspace(sub_channel_CCl4_2[fit_window_CCl4_2[2][0]],␣
 ↪sub_channel_CCl4_2[fit_window_CCl4_2[2][1]], 1000)
p_CCl4_2_2, V_CCl4_2_2 = curve_fit(voigt,␣
 ↪sub_channel_CCl4_2[fit_window_CCl4_2[2][0]:fit_window_CCl4_2[2][1]],␣
 ↪sub_spectrum_CCl4_2[fit_window_CCl4_2[2][0]:fit_window_CCl4_2[2][1]],␣
 ↪p0=[100,523,0.1,0.1,20], bounds=(0,np.inf))
plt.plot(x_2,␣
 ↪voigt(x_2,p_CCl4_2_2[0],p_CCl4_2_2[1],p_CCl4_2_2[2],p_CCl4_2_2[3],p_CCl4_2_2[4]))

x_3 = np.linspace(sub_channel_CCl4_2[fit_window_CCl4_2[3][0]],␣
 ↪sub_channel_CCl4_2[fit_window_CCl4_2[3][1]], 1000)
p_CCl4_2_3, V_CCl4_2_3 = curve_fit(voigt,␣
 ↪sub_channel_CCl4_2[fit_window_CCl4_2[3][0]:fit_window_CCl4_2[3][1]],␣
 ↪sub_spectrum_CCl4_2[fit_window_CCl4_2[3][0]:fit_window_CCl4_2[3][1]],␣
 ↪p0=[100,536,0.1,0.1,20], bounds=(0,np.inf))
plt.plot(x_3,␣
 ↪voigt(x_3,p_CCl4_2_3[0],p_CCl4_2_3[1],p_CCl4_2_3[2],p_CCl4_2_3[3],p_CCl4_2_3[4]))

x_4 = np.linspace(sub_channel_CCl4_2[fit_window_CCl4_2[4][0]],␣
 ↪sub_channel_CCl4_2[fit_window_CCl4_2[4][1]], 1000)
p_CCl4_2_4, V_CCl4_2_4 = curve_fit(voigt,␣
 ↪sub_channel_CCl4_2[fit_window_CCl4_2[4][0]:fit_window_CCl4_2[4][1]],␣
 ↪sub_spectrum_CCl4_2[fit_window_CCl4_2[4][0]:fit_window_CCl4_2[4][1]],␣
 ↪p0=[100,539,0.1,0.1,20], bounds=(0,np.inf))
```

```
plt.plot(x_4,␣
 ↪voigt(x_4,p_CCl4_2_4[0],p_CCl4_2_4[1],p_CCl4_2_4[2],p_CCl4_2_4[3],p_CCl4_2_4[4]))

x_5 = np.linspace(sub_channel_CCl4_2[fit_window_CCl4_2[5][0]],␣
 ↪sub_channel_CCl4_2[fit_window_CCl4_2[5][1]], 1000)
p_CCl4_2_5, V_CCl4_2_5 = curve_fit(voigt,␣
 ↪sub_channel_CCl4_2[fit_window_CCl4_2[5][0]:fit_window_CCl4_2[5][1]],␣
 ↪sub_spectrum_CCl4_2[fit_window_CCl4_2[5][0]:fit_window_CCl4_2[5][1]],␣
 ↪p0=[100,543,0.1,0.1,20], bounds=(0,np.inf))
plt.plot(x_5,␣
 ↪voigt(x_5,p_CCl4_2_5[0],p_CCl4_2_5[1],p_CCl4_2_5[2],p_CCl4_2_5[3],p_CCl4_2_5[4]))

x_6 = np.linspace(sub_channel_CCl4_2[fit_window_CCl4_2[6][0]],␣
 ↪sub_channel_CCl4_2[fit_window_CCl4_2[6][1]], 1000)
p_CCl4_2_6, V_CCl4_2_6 = curve_fit(voigt,␣
 ↪sub_channel_CCl4_2[fit_window_CCl4_2[6][0]:fit_window_CCl4_2[6][1]],␣
 ↪sub_spectrum_CCl4_2[fit_window_CCl4_2[6][0]:fit_window_CCl4_2[6][1]],␣
 ↪p0=[100,553,0.1,0.1,20], bounds=(0,np.inf))
plt.plot(x_6,␣
 ↪voigt(x_6,p_CCl4_2_6[0],p_CCl4_2_6[1],p_CCl4_2_6[2],p_CCl4_2_6[3],p_CCl4_2_6[4]))

plt.savefig('Bilder/Raman_CCl4_2_CCD.pdf', bbox_inches='tight')
```

```
[ ]: peak_energy_CCl4_2_0, delta_energy_CCl4_2_0 = energy(p_CCl4_2_0[1], np.
      ↪sqrt(8*np.log(2)*p_CCl4_2_0[2]**2+4*p_CCl4_2_0[3]**2))
     nu_CCl4_2_0, delta_nu_CCl4_2_0 = peak_energy_CCl4_2_0-laser_energy, np.
      ↪sqrt(delta_energy_CCl4_2_0**2+delta_laser_energy**2)

     peak_energy_CCl4_2_1, delta_energy_CCl4_2_1 = energy(p_CCl4_2_1[1], np.
      ↪sqrt(8*np.log(2)*p_CCl4_2_1[2]**2+4*p_CCl4_2_1[3]**2))
     nu_CCl4_2_1, delta_nu_CCl4_2_1 = peak_energy_CCl4_2_1-laser_energy, np.
      ↪sqrt(delta_energy_CCl4_2_1**2+delta_laser_energy**2)

     peak_energy_CCl4_2_2, delta_energy_CCl4_2_2 = energy(p_CCl4_2_2[1], np.
      ↪sqrt(8*np.log(2)*p_CCl4_2_2[2]**2+4*p_CCl4_2_2[3]**2))
     nu_CCl4_2_2, delta_nu_CCl4_2_2 = peak_energy_CCl4_2_2-laser_energy, np.
      ↪sqrt(delta_energy_CCl4_2_2**2+delta_laser_energy**2)

     peak_energy_CCl4_2_3, delta_energy_CCl4_2_3 = energy(p_CCl4_2_3[1], np.
      ↪sqrt(8*np.log(2)*p_CCl4_2_3[2]**2+4*p_CCl4_2_3[3]**2))
     nu_CCl4_2_3, delta_nu_CCl4_2_3 = peak_energy_CCl4_2_3-laser_energy, np.
      ↪sqrt(delta_energy_CCl4_2_3**2+delta_laser_energy**2)

     peak_energy_CCl4_2_4, delta_energy_CCl4_2_4 = energy(p_CCl4_2_4[1], np.
      ↪sqrt(8*np.log(2)*p_CCl4_2_4[2]**2+4*p_CCl4_2_4[3]**2))
```

```
nu_CCl4_2_4, delta_nu_CCl4_2_4 = peak_energy_CCl4_2_4-laser_energy, np.
  ↪sqrt(delta_energy_CCl4_2_4**2+delta_laser_energy**2)

peak_energy_CCl4_2_5, delta_energy_CCl4_2_5 = energy(p_CCl4_2_5[1], np.
  ↪sqrt(8*np.log(2)*p_CCl4_2_5[2]**2+4*p_CCl4_2_5[3]**2))
nu_CCl4_2_5, delta_nu_CCl4_2_5 = peak_energy_CCl4_2_5-laser_energy, np.
  ↪sqrt(delta_energy_CCl4_2_5**2+delta_laser_energy**2)

peak_energy_CCl4_2_6, delta_energy_CCl4_2_6 = energy(p_CCl4_2_6[1], np.
  ↪sqrt(8*np.log(2)*p_CCl4_2_6[2]**2+4*p_CCl4_2_6[3]**2))
nu_CCl4_2_6, delta_nu_CCl4_2_6 = peak_energy_CCl4_2_6-laser_energy, np.
  ↪sqrt(delta_energy_CCl4_2_6**2+delta_laser_energy**2)

print(nu_CCl4_2_0, delta_nu_CCl4_2_0)
print(nu_CCl4_2_1, delta_nu_CCl4_2_1)
print(nu_CCl4_2_2, delta_nu_CCl4_2_2)
print(nu_CCl4_2_3, delta_nu_CCl4_2_3)
print(nu_CCl4_2_4, delta_nu_CCl4_2_4)
print(nu_CCl4_2_5, delta_nu_CCl4_2_5)
print(nu_CCl4_2_6, delta_nu_CCl4_2_6)
```

3.

```
[ ]: window_CCl4_3 = [1043, 1052]
     channel_CCl4_3, delta_channel_CCl4_3, spectrum_CCl4_3, delta_spectrum_CCl4_3 =␣
       ↪spectrum2("measure_CCl4_4.txt", "measure_CCl4_1.txt", 100)
     sub_channel_CCl4_3, sub_spectrum_CCl4_3, p_CCl4_3, V_CCl4_3 =␣
       ↪substract_fit(channel_CCl4_3, delta_channel_CCl4_3, spectrum_CCl4_3,␣
       ↪delta_spectrum_CCl4_3, window_CCl4_3)

     fit_window_CCl4_3 = [[920,946],[949,968],[974, 994], [1090, 1115],[1120,1146],␣
       ↪[1164,1188], [1255,1300]]

     plt.subplots(figsize=(16,9))
     plt.errorbar(sub_channel_CCl4_3,␣
       ↪sub_spectrum_CCl4_3,yerr=list(delta_spectrum_CCl4_3[0:
       ↪window_CCl4_3[0]])+list(delta_spectrum_CCl4_3[window_CCl4_3[1]:-1]),␣
       ↪ecolor="black", marker="x", ls="", color="green", markersize=5, capsize=3,␣
       ↪label="spectrum")
     for i in fit_window_CCl4_3:
         plt.fill_betweenx([0,70000], sub_channel_CCl4_3[i[0]],␣
       ↪sub_channel_CCl4_3[i[1]], color='lightgrey')
     plt.legend()
     plt.xlim(500,560)
     #plt.title('Raman spectrum of CCl$_4$ with 0° polarization, without $\\lambda/
       ↪2$-plate')
     #plt.xlim(calibration(i[0])[0]-5,calibration(i[1])[0]+5)
```

```
plt.ylim(0, 1500)
plt.grid()
plt.xlabel("wavelength $\\lambda$ [nm]")
plt.ylabel("Intensity in [a.u.]")


x_0 = np.linspace(sub_channel_CCl4_3[fit_window_CCl4_3[0][0]],␣
 ↪sub_channel_CCl4_3[fit_window_CCl4_3[0][1]], 1000)
#ohne Ausreiser in channel 929
p_CCl4_3_0, V_CCl4_3_0 = curve_fit(voigt,␣
 ↪list(sub_channel_CCl4_3[fit_window_CCl4_3[0][0]:
 ↪929])+list(sub_channel_CCl4_3[930:fit_window_CCl4_3[0][1]]),␣
 ↪list(sub_spectrum_CCl4_3[fit_window_CCl4_3[0][0]:
 ↪929])+list(sub_spectrum_CCl4_3[930:fit_window_CCl4_3[0][1]]), p0=[50,517,0.
 ↪1,0.1,80], bounds=(0,np.inf))
plt.plot(x_0,␣
 ↪voigt(x_0,p_CCl4_3_0[0],p_CCl4_3_0[1],p_CCl4_3_0[2],p_CCl4_3_0[3],p_CCl4_3_0[4]))

x_1 = np.linspace(sub_channel_CCl4_3[fit_window_CCl4_3[1][0]],␣
 ↪sub_channel_CCl4_3[fit_window_CCl4_3[1][1]], 1000)
p_CCl4_3_1, V_CCl4_3_1 = curve_fit(voigt,␣
 ↪sub_channel_CCl4_3[fit_window_CCl4_3[1][0]:fit_window_CCl4_3[1][1]],␣
 ↪sub_spectrum_CCl4_3[fit_window_CCl4_3[1][0]:fit_window_CCl4_3[1][1]],␣
 ↪p0=[100,520,0.1,0.1,20], bounds=(0,np.inf))
plt.plot(x_1,␣
 ↪voigt(x_1,p_CCl4_3_1[0],p_CCl4_3_1[1],p_CCl4_3_1[2],p_CCl4_3_1[3],p_CCl4_3_1[4]))

x_2 = np.linspace(sub_channel_CCl4_3[fit_window_CCl4_3[2][0]],␣
 ↪sub_channel_CCl4_3[fit_window_CCl4_3[2][1]], 1000)
p_CCl4_3_2, V_CCl4_3_2 = curve_fit(voigt,␣
 ↪sub_channel_CCl4_3[fit_window_CCl4_3[2][0]:fit_window_CCl4_3[2][1]],␣
 ↪sub_spectrum_CCl4_3[fit_window_CCl4_3[2][0]:fit_window_CCl4_3[2][1]],␣
 ↪p0=[100,523,0.1,0.1,20], bounds=(0,np.inf))
plt.plot(x_2,␣
 ↪voigt(x_2,p_CCl4_3_2[0],p_CCl4_3_2[1],p_CCl4_3_2[2],p_CCl4_3_2[3],p_CCl4_3_2[4]))

x_3 = np.linspace(sub_channel_CCl4_3[fit_window_CCl4_3[3][0]],␣
 ↪sub_channel_CCl4_3[fit_window_CCl4_3[3][1]], 1000)
p_CCl4_3_3, V_CCl4_3_3 = curve_fit(voigt,␣
 ↪sub_channel_CCl4_3[fit_window_CCl4_3[3][0]:fit_window_CCl4_3[3][1]],␣
 ↪sub_spectrum_CCl4_3[fit_window_CCl4_3[3][0]:fit_window_CCl4_3[3][1]],␣
 ↪p0=[100,536,0.1,0.1,20], bounds=(0,np.inf))
plt.plot(x_3,␣
 ↪voigt(x_3,p_CCl4_3_3[0],p_CCl4_3_3[1],p_CCl4_3_3[2],p_CCl4_3_3[3],p_CCl4_3_3[4]))
```

```
x_4 = np.linspace(sub_channel_CCl4_3[fit_window_CCl4_3[4][0]],␣
 ↪sub_channel_CCl4_3[fit_window_CCl4_3[4][1]], 1000)
p_CCl4_3_4, V_CCl4_3_4 = curve_fit(voigt,␣
 ↪sub_channel_CCl4_3[fit_window_CCl4_3[4][0]:fit_window_CCl4_3[4][1]],␣
 ↪sub_spectrum_CCl4_3[fit_window_CCl4_3[4][0]:fit_window_CCl4_3[4][1]],␣
 ↪p0=[100,539,0.1,0.1,20], bounds=(0,np.inf))
plt.plot(x_4,␣
 ↪voigt(x_4,p_CCl4_3_4[0],p_CCl4_3_4[1],p_CCl4_3_4[2],p_CCl4_3_4[3],p_CCl4_3_4[4]))

x_5 = np.linspace(sub_channel_CCl4_3[fit_window_CCl4_3[5][0]],␣
 ↪sub_channel_CCl4_3[fit_window_CCl4_3[5][1]], 1000)
p_CCl4_3_5, V_CCl4_3_5 = curve_fit(voigt,␣
 ↪sub_channel_CCl4_3[fit_window_CCl4_3[5][0]:fit_window_CCl4_3[5][1]],␣
 ↪sub_spectrum_CCl4_3[fit_window_CCl4_3[5][0]:fit_window_CCl4_3[5][1]],␣
 ↪p0=[100,543,0.1,0.1,20], bounds=(0,np.inf))
plt.plot(x_5,␣
 ↪voigt(x_5,p_CCl4_3_5[0],p_CCl4_3_5[1],p_CCl4_3_5[2],p_CCl4_3_5[3],p_CCl4_3_5[4]))

x_6 = np.linspace(sub_channel_CCl4_3[fit_window_CCl4_3[6][0]],␣
 ↪sub_channel_CCl4_3[fit_window_CCl4_3[6][1]], 1000)
p_CCl4_3_6, V_CCl4_3_6 = curve_fit(voigt,␣
 ↪sub_channel_CCl4_3[fit_window_CCl4_3[6][0]:fit_window_CCl4_3[6][1]],␣
 ↪sub_spectrum_CCl4_3[fit_window_CCl4_3[6][0]:fit_window_CCl4_3[6][1]],␣
 ↪p0=[100,553,0.1,0.1,20], bounds=(0,np.inf))
plt.plot(x_6,␣
 ↪voigt(x_6,p_CCl4_3_6[0],p_CCl4_3_6[1],p_CCl4_3_6[2],p_CCl4_3_6[3],p_CCl4_3_6[4]))

plt.savefig('Bilder/Raman_CCl4_3_CCD.pdf', bbox_inches='tight')
```

[ ]:

```
peak_energy_CCl4_3_0, delta_energy_CCl4_3_0 = energy(p_CCl4_3_0[1], np.
 ↪sqrt(8*np.log(2)*p_CCl4_3_0[2]**2+4*p_CCl4_3_0[3]**2))
nu_CCl4_3_0, delta_nu_CCl4_3_0 = peak_energy_CCl4_3_0-laser_energy, np.
 ↪sqrt(delta_energy_CCl4_3_0**2+delta_laser_energy**2)

peak_energy_CCl4_3_1, delta_energy_CCl4_3_1 = energy(p_CCl4_3_1[1], np.
 ↪sqrt(8*np.log(2)*p_CCl4_3_1[2]**2+4*p_CCl4_3_1[3]**2))
nu_CCl4_3_1, delta_nu_CCl4_3_1 = peak_energy_CCl4_3_1-laser_energy, np.
 ↪sqrt(delta_energy_CCl4_3_1**2+delta_laser_energy**2)

peak_energy_CCl4_3_2, delta_energy_CCl4_3_2 = energy(p_CCl4_3_2[1], np.
 ↪sqrt(8*np.log(2)*p_CCl4_3_2[2]**2+4*p_CCl4_3_2[3]**2))
nu_CCl4_3_2, delta_nu_CCl4_3_2 = peak_energy_CCl4_3_2-laser_energy, np.
 ↪sqrt(delta_energy_CCl4_3_2**2+delta_laser_energy**2)

peak_energy_CCl4_3_3, delta_energy_CCl4_3_3 = energy(p_CCl4_3_3[1], np.
 ↪sqrt(8*np.log(2)*p_CCl4_3_3[2]**2+4*p_CCl4_3_3[3]**2))
```

```
nu_CCl4_3_3, delta_nu_CCl4_3_3 = peak_energy_CCl4_3_3-laser_energy, np.
 ↪sqrt(delta_energy_CCl4_3_3**2+delta_laser_energy**2)

peak_energy_CCl4_3_4, delta_energy_CCl4_3_4 = energy(p_CCl4_3_4[1], np.
 ↪sqrt(8*np.log(2)*p_CCl4_3_4[2]**2+4*p_CCl4_3_4[3]**2))
nu_CCl4_3_4, delta_nu_CCl4_3_4 = peak_energy_CCl4_3_4-laser_energy, np.
 ↪sqrt(delta_energy_CCl4_3_4**2+delta_laser_energy**2)

peak_energy_CCl4_3_5, delta_energy_CCl4_3_5 = energy(p_CCl4_3_5[1], np.
 ↪sqrt(8*np.log(2)*p_CCl4_3_5[2]**2+4*p_CCl4_3_5[3]**2))
nu_CCl4_3_5, delta_nu_CCl4_3_5 = peak_energy_CCl4_3_5-laser_energy, np.
 ↪sqrt(delta_energy_CCl4_3_5**2+delta_laser_energy**2)

peak_energy_CCl4_3_6, delta_energy_CCl4_3_6 = energy(p_CCl4_3_6[1], np.
 ↪sqrt(8*np.log(2)*p_CCl4_3_6[2]**2+4*p_CCl4_3_6[3]**2))
nu_CCl4_3_6, delta_nu_CCl4_3_6 = peak_energy_CCl4_3_6-laser_energy, np.
 ↪sqrt(delta_energy_CCl4_3_6**2+delta_laser_energy**2)

print(nu_CCl4_3_0, delta_nu_CCl4_3_0)
print(nu_CCl4_3_1, delta_nu_CCl4_3_1)
print(nu_CCl4_3_2, delta_nu_CCl4_3_2)
print(nu_CCl4_3_3, delta_nu_CCl4_3_3)
print(nu_CCl4_3_4, delta_nu_CCl4_3_4)
print(nu_CCl4_3_5, delta_nu_CCl4_3_5)
print(nu_CCl4_3_6, delta_nu_CCl4_3_6)
```

4.

[ ]:

```
window_CCl4_4 = [1043, 1052]
channel_CCl4_4, delta_channel_CCl4_4, spectrum_CCl4_4, delta_spectrum_CCl4_4 =␣
 ↪spectrum2("measure_CCl4_5.txt", "measure_CCl4_1.txt", 100)
sub_channel_CCl4_4, sub_spectrum_CCl4_4, p_CCl4_4, V_CCl4_4 =␣
 ↪substract_fit(channel_CCl4_4, delta_channel_CCl4_4, spectrum_CCl4_4,␣
 ↪delta_spectrum_CCl4_4, window_CCl4_4)

fit_window_CCl4_4 = [[920,944],[949,968],[974, 994], [1090, 1115],[1120,1146],␣
 ↪[1164,1188], [1255,1300]]

plt.subplots(figsize=(16,9))
plt.errorbar(sub_channel_CCl4_4,␣
 ↪sub_spectrum_CCl4_4,yerr=list(delta_spectrum_CCl4_4[0:
 ↪window_CCl4_4[0]])+list(delta_spectrum_CCl4_4[window_CCl4_4[1]:-1]),␣
 ↪ecolor="black", marker="x", ls="", color="green", markersize=5, capsize=3,␣
 ↪label="spectrum")
for i in fit_window_CCl4_4:
```

```
    plt.
 ↪fill_betweenx([0,70000],sub_channel_CCl4_4[i[0]],sub_channel_CCl4_4[i[1]],␣
 ↪color='lightgrey')
plt.legend()
plt.xlim(500,560)
#plt.title('Raman spectrum of CCl$_4$ with 0° polarization, with $\\lambda/
 ↪2$-plate')
#plt.xlim(calibration(i[0])[0]-5,calibration(i[1])[0]+5)
plt.ylim(0, 1500)
plt.grid()
plt.xlabel("wavelength $\\lambda$ [nm]")
plt.ylabel("Intensity in [a.u.]")

x_0 = np.linspace(sub_channel_CCl4_4[fit_window_CCl4_4[0][0]],␣
 ↪sub_channel_CCl4_4[fit_window_CCl4_4[0][1]], 1000)
p_CCl4_4_0, V_CCl4_4_0 = curve_fit(voigt,␣
 ↪sub_channel_CCl4_4[fit_window_CCl4_4[0][0]:fit_window_CCl4_4[0][1]],␣
 ↪sub_spectrum_CCl4_4[fit_window_CCl4_4[0][0]:fit_window_CCl4_4[0][1]],␣
 ↪p0=[100,517,0.1,0.1,20], bounds=(0,np.inf))
plt.plot(x_0,␣
 ↪voigt(x_0,p_CCl4_4_0[0],p_CCl4_4_0[1],p_CCl4_4_0[2],p_CCl4_4_0[3],p_CCl4_4_0[4]))

x_1 = np.linspace(sub_channel_CCl4_4[fit_window_CCl4_4[1][0]],␣
 ↪sub_channel_CCl4_4[fit_window_CCl4_4[1][1]], 1000)
p_CCl4_4_1, V_CCl4_4_1 = curve_fit(voigt,␣
 ↪sub_channel_CCl4_4[fit_window_CCl4_4[1][0]:fit_window_CCl4_4[1][1]],␣
 ↪sub_spectrum_CCl4_4[fit_window_CCl4_4[1][0]:fit_window_CCl4_4[1][1]],␣
 ↪p0=[100,520,0.1,0.1,20], bounds=(0,np.inf))
plt.plot(x_1,␣
 ↪voigt(x_1,p_CCl4_4_1[0],p_CCl4_4_1[1],p_CCl4_4_1[2],p_CCl4_4_1[3],p_CCl4_4_1[4]))

x_2 = np.linspace(sub_channel_CCl4_4[fit_window_CCl4_4[2][0]],␣
 ↪sub_channel_CCl4_4[fit_window_CCl4_4[2][1]], 1000)
p_CCl4_4_2, V_CCl4_4_2 = curve_fit(voigt,␣
 ↪sub_channel_CCl4_4[fit_window_CCl4_4[2][0]:fit_window_CCl4_4[2][1]],␣
 ↪sub_spectrum_CCl4_4[fit_window_CCl4_4[2][0]:fit_window_CCl4_4[2][1]],␣
 ↪p0=[100,523,0.1,0.1,20], bounds=(0,np.inf))
plt.plot(x_2,␣
 ↪voigt(x_2,p_CCl4_4_2[0],p_CCl4_4_2[1],p_CCl4_4_2[2],p_CCl4_4_2[3],p_CCl4_4_2[4]))

x_3 = np.linspace(sub_channel_CCl4_4[fit_window_CCl4_4[3][0]],␣
 ↪sub_channel_CCl4_4[fit_window_CCl4_4[3][1]], 1000)
p_CCl4_4_3, V_CCl4_4_3 = curve_fit(voigt,␣
 ↪sub_channel_CCl4_4[fit_window_CCl4_4[3][0]:fit_window_CCl4_4[3][1]],␣
 ↪sub_spectrum_CCl4_4[fit_window_CCl4_4[3][0]:fit_window_CCl4_4[3][1]],␣
 ↪p0=[100,536,0.1,0.1,20], bounds=(0,np.inf))
```

```
plt.plot(x_3,␣
 ↪voigt(x_3,p_CCl4_4_3[0],p_CCl4_4_3[1],p_CCl4_4_3[2],p_CCl4_4_3[3],p_CCl4_4_3[4]))

x_4 = np.linspace(sub_channel_CCl4_4[fit_window_CCl4_4[4][0]],␣
 ↪sub_channel_CCl4_4[fit_window_CCl4_4[4][1]], 1000)
p_CCl4_4_4, V_CCl4_4_4 = curve_fit(voigt,␣
 ↪sub_channel_CCl4_4[fit_window_CCl4_4[4][0]:fit_window_CCl4_4[4][1]],␣
 ↪sub_spectrum_CCl4_4[fit_window_CCl4_4[4][0]:fit_window_CCl4_4[4][1]],␣
 ↪p0=[100,539,0.1,0.1,20], bounds=(0,np.inf))
plt.plot(x_4,␣
 ↪voigt(x_4,p_CCl4_4_4[0],p_CCl4_4_4[1],p_CCl4_4_4[2],p_CCl4_4_4[3],p_CCl4_4_4[4]))

x_5 = np.linspace(sub_channel_CCl4_4[fit_window_CCl4_4[5][0]],␣
 ↪sub_channel_CCl4_4[fit_window_CCl4_4[5][1]], 1000)
p_CCl4_4_5, V_CCl4_4_5 = curve_fit(voigt,␣
 ↪sub_channel_CCl4_4[fit_window_CCl4_4[5][0]:fit_window_CCl4_4[5][1]],␣
 ↪sub_spectrum_CCl4_4[fit_window_CCl4_4[5][0]:fit_window_CCl4_4[5][1]],␣
 ↪p0=[100,543,0.1,0.1,20], bounds=(0,np.inf))
plt.plot(x_5,␣
 ↪voigt(x_5,p_CCl4_4_5[0],p_CCl4_4_5[1],p_CCl4_4_5[2],p_CCl4_4_5[3],p_CCl4_4_5[4]))

x_6 = np.linspace(sub_channel_CCl4_4[fit_window_CCl4_4[6][0]],␣
 ↪sub_channel_CCl4_4[fit_window_CCl4_4[6][1]], 1000)
p_CCl4_4_6, V_CCl4_4_6 = curve_fit(voigt,␣
 ↪sub_channel_CCl4_4[fit_window_CCl4_4[6][0]:fit_window_CCl4_4[6][1]],␣
 ↪sub_spectrum_CCl4_4[fit_window_CCl4_4[6][0]:fit_window_CCl4_4[6][1]],␣
 ↪p0=[100,553,0.1,0.1,20], bounds=(0,np.inf))
plt.plot(x_6,␣
 ↪voigt(x_6,p_CCl4_4_6[0],p_CCl4_4_6[1],p_CCl4_4_6[2],p_CCl4_4_6[3],p_CCl4_4_6[4]))

plt.savefig('Bilder/Raman_CCl4_4_CCD.pdf', bbox_inches='tight')
```

```
[ ]: peak_energy_CCl4_4_0, delta_energy_CCl4_4_0 = energy(p_CCl4_4_0[1], np.
      ↪sqrt(8*np.log(2)*p_CCl4_4_0[2]**2+4*p_CCl4_4_0[3]**2))
     nu_CCl4_4_0, delta_nu_CCl4_4_0 = peak_energy_CCl4_4_0-laser_energy, np.
      ↪sqrt(delta_energy_CCl4_4_0**2+delta_laser_energy**2)

     peak_energy_CCl4_4_1, delta_energy_CCl4_4_1 = energy(p_CCl4_4_1[1], np.
      ↪sqrt(8*np.log(2)*p_CCl4_4_1[2]**2+4*p_CCl4_4_1[3]**2))
     nu_CCl4_4_1, delta_nu_CCl4_4_1 = peak_energy_CCl4_4_1-laser_energy, np.
      ↪sqrt(delta_energy_CCl4_4_1**2+delta_laser_energy**2)

     peak_energy_CCl4_4_2, delta_energy_CCl4_4_2 = energy(p_CCl4_4_2[1], np.
      ↪sqrt(8*np.log(2)*p_CCl4_4_2[2]**2+4*p_CCl4_4_2[3]**2))
     nu_CCl4_4_2, delta_nu_CCl4_4_2 = peak_energy_CCl4_4_2-laser_energy, np.
      ↪sqrt(delta_energy_CCl4_4_2**2+delta_laser_energy**2)
```

```
peak_energy_CCl4_4_3, delta_energy_CCl4_4_3 = energy(p_CCl4_4_3[1], np.
 ↪sqrt(8*np.log(2)*p_CCl4_4_3[2]**2+4*p_CCl4_4_3[3]**2))
nu_CCl4_4_3, delta_nu_CCl4_4_3 = peak_energy_CCl4_4_3-laser_energy, np.
 ↪sqrt(delta_energy_CCl4_4_3**2+delta_laser_energy**2)

peak_energy_CCl4_4_4, delta_energy_CCl4_4_4 = energy(p_CCl4_4_4[1], np.
 ↪sqrt(8*np.log(2)*p_CCl4_4_4[2]**2+4*p_CCl4_4_4[3]**2))
nu_CCl4_4_4, delta_nu_CCl4_4_4 = peak_energy_CCl4_4_4-laser_energy, np.
 ↪sqrt(delta_energy_CCl4_4_4**2+delta_laser_energy**2)

peak_energy_CCl4_4_5, delta_energy_CCl4_4_5 = energy(p_CCl4_4_5[1], np.
 ↪sqrt(8*np.log(2)*p_CCl4_4_5[2]**2+4*p_CCl4_4_5[3]**2))
nu_CCl4_4_5, delta_nu_CCl4_4_5 = peak_energy_CCl4_4_5-laser_energy, np.
 ↪sqrt(delta_energy_CCl4_4_5**2+delta_laser_energy**2)

peak_energy_CCl4_4_6, delta_energy_CCl4_4_6 = energy(p_CCl4_4_6[1], np.
 ↪sqrt(8*np.log(2)*p_CCl4_4_6[2]**2+4*p_CCl4_4_6[3]**2))
nu_CCl4_4_6, delta_nu_CCl4_4_6 = peak_energy_CCl4_4_6-laser_energy, np.
 ↪sqrt(delta_energy_CCl4_4_6**2+delta_laser_energy**2)

print(nu_CCl4_4_0, delta_nu_CCl4_4_0)
print(nu_CCl4_4_1, delta_nu_CCl4_4_1)
print(nu_CCl4_4_2, delta_nu_CCl4_4_2)
print(nu_CCl4_4_3, delta_nu_CCl4_4_3)
print(nu_CCl4_4_4, delta_nu_CCl4_4_4)
print(nu_CCl4_4_5, delta_nu_CCl4_4_5)
print(nu_CCl4_4_6, delta_nu_CCl4_4_6)
```

Depolarization

```
[ ]: def dep(I_para, I_perp, delta_I_para, delta_I_perp, position, delta_position,␣
      ↪lambdahalf=False):
         r, dr = pol(position, delta_position)
         if lambdahalf:
             return I_para/(I_perp*r), np.sqrt(delta_I_para/
      ↪(I_perp**2*r**2)+(I_para**2)*delta_I_perp/(I_perp**4*r**2)+I_para**2*dr**2/
      ↪(I_perp**2*r**4))
         else:
             return (I_para*r)/I_perp, np.sqrt(delta_I_para/
      ↪I_perp**2*r**2+(I_para**2*r**2)*delta_I_perp/I_perp**4+I_para**2*dr**2/
      ↪I_perp**2)
```

```
[ ]: I1 = [p_CCl4_1_0[0], p_CCl4_1_1[0], p_CCl4_1_2[0], p_CCl4_1_3[0],␣
      ↪p_CCl4_1_4[0], p_CCl4_1_5[0], p_CCl4_1_6[0]]
     delta_I1 = [V_CCl4_1_0[0][0], V_CCl4_1_1[0][0], V_CCl4_1_2[0][0],␣
      ↪V_CCl4_1_3[0][0], V_CCl4_1_4[0][0], V_CCl4_1_5[0][0], V_CCl4_1_6[0][0]]
```

```
L1 = [p_CCl4_1_0[1], p_CCl4_1_1[1], p_CCl4_1_2[1], p_CCl4_1_3[1],␣
 ↪p_CCl4_1_4[1], p_CCl4_1_5[1], p_CCl4_1_6[1]]
delta_L1 = [V_CCl4_1_0[1][1], V_CCl4_1_1[1][1], V_CCl4_1_2[1][1],␣
 ↪V_CCl4_1_3[1][1], V_CCl4_1_4[1][1], V_CCl4_1_5[1][1], V_CCl4_1_6[1][1]]

I2 = [p_CCl4_2_0[0], p_CCl4_2_1[0], p_CCl4_2_2[0], p_CCl4_2_3[0],␣
 ↪p_CCl4_2_4[0], p_CCl4_2_5[0], p_CCl4_2_6[0]]
delta_I2 = [V_CCl4_2_0[0][0], V_CCl4_2_1[0][0], V_CCl4_2_2[0][0],␣
 ↪V_CCl4_2_3[0][0], V_CCl4_2_4[0][0], V_CCl4_2_5[0][0], V_CCl4_2_6[0][0]]
L2 = [p_CCl4_2_0[1], p_CCl4_2_1[1], p_CCl4_2_2[1], p_CCl4_2_3[1],␣
 ↪p_CCl4_2_4[1], p_CCl4_2_5[1], p_CCl4_2_6[1]]
delta_L2 = [V_CCl4_2_0[1][1], V_CCl4_2_1[1][1], V_CCl4_2_2[1][1],␣
 ↪V_CCl4_2_3[1][1], V_CCl4_2_4[1][1], V_CCl4_2_5[1][1], V_CCl4_2_6[1][1]]

I3 = [p_CCl4_3_0[0], p_CCl4_3_1[0], p_CCl4_3_2[0], p_CCl4_3_3[0],␣
 ↪p_CCl4_3_4[0], p_CCl4_3_5[0], p_CCl4_3_6[0]]
delta_I3 = [V_CCl4_3_0[0][0], V_CCl4_3_1[0][0], V_CCl4_3_2[0][0],␣
 ↪V_CCl4_3_3[0][0], V_CCl4_3_4[0][0], V_CCl4_3_5[0][0], V_CCl4_3_6[0][0]]
L3 = [p_CCl4_3_0[1], p_CCl4_3_1[1], p_CCl4_3_2[1], p_CCl4_3_3[1],␣
 ↪p_CCl4_3_4[1], p_CCl4_3_5[1], p_CCl4_3_6[1]]
delta_L3 = [V_CCl4_3_0[1][1], V_CCl4_3_1[1][1], V_CCl4_3_2[1][1],␣
 ↪V_CCl4_3_3[1][1], V_CCl4_3_4[1][1], V_CCl4_3_5[1][1], V_CCl4_3_6[1][1]]

I4 = [p_CCl4_4_0[0], p_CCl4_4_1[0], p_CCl4_4_2[0], p_CCl4_4_3[0],␣
 ↪p_CCl4_4_4[0], p_CCl4_4_5[0], p_CCl4_4_6[0]]
delta_I4 = [V_CCl4_4_0[0][0], V_CCl4_4_1[0][0], V_CCl4_4_2[0][0],␣
 ↪V_CCl4_4_3[0][0], V_CCl4_4_4[0][0], V_CCl4_4_5[0][0], V_CCl4_4_6[0][0]]
L4 = [p_CCl4_4_0[1], p_CCl4_4_1[1], p_CCl4_4_2[1], p_CCl4_4_3[1],␣
 ↪p_CCl4_4_4[1], p_CCl4_4_5[1], p_CCl4_4_6[1]]
delta_L4 = [V_CCl4_4_0[1][1], V_CCl4_4_1[1][1], V_CCl4_4_2[1][1],␣
 ↪V_CCl4_4_3[1][1], V_CCl4_4_4[1][1], V_CCl4_4_5[1][1], V_CCl4_4_6[1][1]]

dep_31 = [dep(I00, I90, dI00, dI90, pos, np.sqrt(dpos)) for I00, I90, dI00,␣
 ↪dI90, pos, dpos in zip(I3, I1, delta_I3, delta_I1, L3, delta_L3)]
dep_24 = [dep(I00, I90, dI00, dI90, pos, np.sqrt(dpos), lambdahalf=True) for␣
 ↪I00, I90, dI00, dI90, pos, dpos in zip(I2, I4, delta_I2, delta_I4, L4,␣
 ↪delta_L4)]

dep_31, dep_24
```

```
[ ]:
```

0.6.3 CHCl3

[ ]:
```
window_CHCl3_1 = [1043, 1052]
channel_CHCl3_1, delta_channel_CHCl3_1, spectrum_CHCl3_1,␣
 ↪delta_spectrum_CHCl3_1 = spectrum2("measure_CHCl3_1.txt", "measure_CHCl3_0.
 ↪txt", 100)
sub_channel_CHCl3_1, sub_spectrum_CHCl3_1, p_CHCl3_1, V_CHCl3_1 =␣
 ↪substract_fit(channel_CHCl3_1, delta_channel_CHCl3_1, spectrum_CHCl3_1,␣
 ↪delta_spectrum_CHCl3_1, window_CHCl3_1)

fit_window_CHCl3_1 = [[920,962],[965, 984], [1107, 1132],[1138,1160],␣
 ↪[1233,1253], [1257,1300]]

plt.subplots(figsize=(16,9))
plt.errorbar(sub_channel_CHCl3_1,␣
 ↪sub_spectrum_CHCl3_1,yerr=list(delta_spectrum_CHCl3_1[0:
 ↪window_CHCl3_1[0]])+list(delta_spectrum_CHCl3_1[window_CHCl3_1[1]:-1]),␣
 ↪ecolor="black", marker="x", ls="", color="green", markersize=5, capsize=3,␣
 ↪label="spectrum")
for i in fit_window_CHCl3_1:
    plt.
 ↪fill_betweenx([-100,70000],sub_channel_CHCl3_1[i[0]],sub_channel_CHCl3_1[i[1]],␣
 ↪color='lightgrey')
plt.legend()
plt.xlim(500,560)
#plt.title('Raman spectrum of CHCl$_3$ with 90° polarization, without $\\lambda/
 ↪2$-plate')
#plt.xlim(calibration(i[0])[0]-5,calibration(i[1])[0]+5)
plt.ylim(-50, 1600)
plt.grid()
plt.xlabel("wavelength $\\lambda$ [nm]")
plt.ylabel("Intensity in [a.u.]")

x_0 = np.linspace(sub_channel_CHCl3_1[fit_window_CHCl3_1[0][0]],␣
 ↪sub_channel_CHCl3_1[fit_window_CHCl3_1[0][1]], 1000)
p_CHCl3_1_0, V_CHCl3_1_0 = curve_fit(voigt,␣
 ↪sub_channel_CHCl3_1[fit_window_CHCl3_1[0][0]:fit_window_CHCl3_1[0][1]],␣
 ↪sub_spectrum_CHCl3_1[fit_window_CHCl3_1[0][0]:fit_window_CHCl3_1[0][1]],␣
 ↪p0=[100,517,0.1,0.1,20], bounds=(0,np.inf))
plt.plot(x_0,␣
 ↪voigt(x_0,p_CHCl3_1_0[0],p_CHCl3_1_0[1],p_CHCl3_1_0[2],p_CHCl3_1_0[3],p_CHCl3_1_0[4]))

x_1 = np.linspace(sub_channel_CHCl3_1[fit_window_CHCl3_1[1][0]],␣
 ↪sub_channel_CHCl3_1[fit_window_CHCl3_1[1][1]], 1000)
```

```
#ohne Ausreiser bei 971 list(sub_channel_CHCl3_1[fit_window_CHCl3_1[1][0]:
 ↪971])+list(sub_channel_CHCl3_1[972:fit_window_CHCl3_1[1][1]]),␣
 ↪list(sub_spectrum_CHCl3_1[fit_window_CHCl3_1[1][0]:
 ↪971])+list(sub_spectrum_CHCl3_1[972:fit_window_CHCl3_1[1][1]])
p_CHCl3_1_1, V_CHCl3_1_1 = curve_fit(voigt, np.
 ↪delete(sub_channel_CHCl3_1,971)[fit_window_CHCl3_1[1][0]:
 ↪fit_window_CHCl3_1[1][1]], np.
 ↪delete(sub_spectrum_CHCl3_1,971)[fit_window_CHCl3_1[1][0]:
 ↪fit_window_CHCl3_1[1][1]], p0=[100,520,0.1,0.1,20], bounds=(0,np.inf))
plt.plot(x_1,␣
 ↪voigt(x_1,p_CHCl3_1_1[0],p_CHCl3_1_1[1],p_CHCl3_1_1[2],p_CHCl3_1_1[3],p_CHCl3_1_1[4]))

x_2 = np.linspace(sub_channel_CHCl3_1[fit_window_CHCl3_1[2][0]],␣
 ↪sub_channel_CHCl3_1[fit_window_CHCl3_1[2][1]], 1000)
p_CHCl3_1_2, V_CHCl3_1_2 = curve_fit(voigt,␣
 ↪sub_channel_CHCl3_1[fit_window_CHCl3_1[2][0]:fit_window_CHCl3_1[2][1]],␣
 ↪sub_spectrum_CHCl3_1[fit_window_CHCl3_1[2][0]:fit_window_CHCl3_1[2][1]],␣
 ↪p0=[100,537,0.1,0.1,20], bounds=(0,np.inf))
plt.plot(x_2,␣
 ↪voigt(x_2,p_CHCl3_1_2[0],p_CHCl3_1_2[1],p_CHCl3_1_2[2],p_CHCl3_1_2[3],p_CHCl3_1_2[4]))

x_3 = np.linspace(sub_channel_CHCl3_1[fit_window_CHCl3_1[3][0]],␣
 ↪sub_channel_CHCl3_1[fit_window_CHCl3_1[3][1]], 1000)
p_CHCl3_1_3, V_CHCl3_1_3 = curve_fit(voigt,␣
 ↪sub_channel_CHCl3_1[fit_window_CHCl3_1[3][0]:fit_window_CHCl3_1[3][1]],␣
 ↪sub_spectrum_CHCl3_1[fit_window_CHCl3_1[3][0]:fit_window_CHCl3_1[3][1]],␣
 ↪p0=[100,540,0.1,0.1,20], bounds=(0,np.inf))
plt.plot(x_3,␣
 ↪voigt(x_3,p_CHCl3_1_3[0],p_CHCl3_1_3[1],p_CHCl3_1_3[2],p_CHCl3_1_3[3],p_CHCl3_1_3[4]))

x_4 = np.linspace(sub_channel_CHCl3_1[fit_window_CHCl3_1[4][0]],␣
 ↪sub_channel_CHCl3_1[fit_window_CHCl3_1[4][1]], 1000)
p_CHCl3_1_4, V_CHCl3_1_4 = curve_fit(voigt,␣
 ↪sub_channel_CHCl3_1[fit_window_CHCl3_1[4][0]:fit_window_CHCl3_1[4][1]],␣
 ↪sub_spectrum_CHCl3_1[fit_window_CHCl3_1[4][0]:fit_window_CHCl3_1[4][1]],␣
 ↪p0=[100,549,0.1,0.1,20], bounds=(0,np.inf))
plt.plot(x_4,␣
 ↪voigt(x_4,p_CHCl3_1_4[0],p_CHCl3_1_4[1],p_CHCl3_1_4[2],p_CHCl3_1_4[3],p_CHCl3_1_4[4]))

x_5 = np.linspace(sub_channel_CHCl3_1[fit_window_CHCl3_1[5][0]],␣
 ↪sub_channel_CHCl3_1[fit_window_CHCl3_1[5][1]], 1000)
p_CHCl3_1_5, V_CHCl3_1_5 = curve_fit(voigt,␣
 ↪sub_channel_CHCl3_1[fit_window_CHCl3_1[5][0]:fit_window_CHCl3_1[5][1]],␣
 ↪sub_spectrum_CHCl3_1[fit_window_CHCl3_1[5][0]:fit_window_CHCl3_1[5][1]],␣
 ↪p0=[100,552,0.1,0.1,20], bounds=(0,np.inf))
```

```
plt.plot(x_5,␣
 ↪voigt(x_5,p_CHCl3_1_5[0],p_CHCl3_1_5[1],p_CHCl3_1_5[2],p_CHCl3_1_5[3],p_CHCl3_1_5[4]))

plt.savefig('Bilder/Raman_CHCl3_1_CCD.pdf', bbox_inches='tight')
```

[ ]:

```
peak_energy_CHCl3_1_0, delta_energy_CHCl3_1_0 = energy(p_CHCl3_1_0[1], np.
 ↪sqrt(8*np.log(2)*p_CHCl3_1_0[2]**2+4*p_CHCl3_1_0[3]**2))
nu_CHCl3_1_0, delta_nu_CHCl3_1_0 = peak_energy_CHCl3_1_0-laser_energy, np.
 ↪sqrt(delta_energy_CHCl3_1_0**2+delta_laser_energy**2)

peak_energy_CHCl3_1_1, delta_energy_CHCl3_1_1 = energy(p_CHCl3_1_1[1], np.
 ↪sqrt(8*np.log(2)*p_CHCl3_1_1[2]**2+4*p_CHCl3_1_1[3]**2))
nu_CHCl3_1_1, delta_nu_CHCl3_1_1 = peak_energy_CHCl3_1_1-laser_energy, np.
 ↪sqrt(delta_energy_CHCl3_1_1**2+delta_laser_energy**2)

peak_energy_CHCl3_1_2, delta_energy_CHCl3_1_2 = energy(p_CHCl3_1_2[1], np.
 ↪sqrt(8*np.log(2)*p_CHCl3_1_2[2]**2+4*p_CHCl3_1_2[3]**2))
nu_CHCl3_1_2, delta_nu_CHCl3_1_2 = peak_energy_CHCl3_1_2-laser_energy, np.
 ↪sqrt(delta_energy_CHCl3_1_2**2+delta_laser_energy**2)

peak_energy_CHCl3_1_3, delta_energy_CHCl3_1_3 = energy(p_CHCl3_1_3[1], np.
 ↪sqrt(8*np.log(2)*p_CHCl3_1_3[2]**2+4*p_CHCl3_1_3[3]**2))
nu_CHCl3_1_3, delta_nu_CHCl3_1_3 = peak_energy_CHCl3_1_3-laser_energy, np.
 ↪sqrt(delta_energy_CHCl3_1_3**2+delta_laser_energy**2)

peak_energy_CHCl3_1_4, delta_energy_CHCl3_1_4 = energy(p_CHCl3_1_4[1], np.
 ↪sqrt(8*np.log(2)*p_CHCl3_1_4[2]**2+4*p_CHCl3_1_4[3]**2))
nu_CHCl3_1_4, delta_nu_CHCl3_1_4 = peak_energy_CHCl3_1_4-laser_energy, np.
 ↪sqrt(delta_energy_CHCl3_1_4**2+delta_laser_energy**2)

peak_energy_CHCl3_1_5, delta_energy_CHCl3_1_5 = energy(p_CHCl3_1_5[1], np.
 ↪sqrt(8*np.log(2)*p_CHCl3_1_5[2]**2+4*p_CHCl3_1_5[3]**2))
nu_CHCl3_1_5, delta_nu_CHCl3_1_5 = peak_energy_CHCl3_1_5-laser_energy, np.
 ↪sqrt(delta_energy_CHCl3_1_5**2+delta_laser_energy**2)

print(nu_CHCl3_1_0, delta_nu_CHCl3_1_0)
print(nu_CHCl3_1_1, delta_nu_CHCl3_1_1)
print(nu_CHCl3_1_2, delta_nu_CHCl3_1_2)
print(nu_CHCl3_1_3, delta_nu_CHCl3_1_3)
print(nu_CHCl3_1_4, delta_nu_CHCl3_1_4)
print(nu_CHCl3_1_5, delta_nu_CHCl3_1_5)

print(lit_CHCl3)
```

[ ]:

0.6.4 CH2Cl2

[ ]:

```
window_CH2Cl2_1 = [1043, 1053]
channel_CH2Cl2_1, delta_channel_CH2Cl2_1, spectrum_CH2Cl2_1,␣
 ↪delta_spectrum_CH2Cl2_1 = spectrum2("measure_CH2Cl2_1.txt",␣
 ↪"measure_CH2Cl2_0.txt", 100)
sub_channel_CH2Cl2_1, sub_spectrum_CH2Cl2_1, p_CH2Cl2_1, V_CH2Cl2_1 =␣
 ↪substract_fit(channel_CH2Cl2_1, delta_channel_CH2Cl2_1, spectrum_CH2Cl2_1,␣
 ↪delta_spectrum_CH2Cl2_1, window_CH2Cl2_1)

fit_window_CH2Cl2_1 = [[925,945],[947, 960], [962, 978],[1112,1141],␣
 ↪[1240,1263], [1264,1290]]

plt.subplots(figsize=(16,9))
plt.errorbar(sub_channel_CH2Cl2_1,␣
 ↪sub_spectrum_CH2Cl2_1,yerr=list(delta_spectrum_CH2Cl2_1[0:
 ↪window_CH2Cl2_1[0]])+list(delta_spectrum_CH2Cl2_1[window_CH2Cl2_1[1]:-1]),␣
 ↪ecolor="black", marker="x", ls="", color="green", markersize=5, capsize=3,␣
 ↪label="spectrum")
for i in fit_window_CH2Cl2_1:
    plt.fill_betweenx([-100,␣
 ↪70000],sub_channel_CH2Cl2_1[i[0]],sub_channel_CH2Cl2_1[i[1]],␣
 ↪color='lightgrey')
plt.legend()
plt.xlim(500,560)
#plt.title('Raman spectrum of CH$_2$Cl$_2$ with 90° polarization, without␣
 ↪$\\lambda/2$-plate')
#plt.xlim(calibration(i[0])[0]-5,calibration(i[1])[0]+5)
plt.ylim(-100, 1000)
plt.grid()
plt.xlabel("wavelength $\\lambda$ [nm]")
plt.ylabel("Intensity in [a.u.]")

x_0 = np.linspace(sub_channel_CH2Cl2_1[fit_window_CH2Cl2_1[0][0]],␣
 ↪sub_channel_CH2Cl2_1[fit_window_CH2Cl2_1[0][1]], 1000)
p_CH2Cl2_1_0, V_CH2Cl2_1_0 = curve_fit(voigt,␣
 ↪sub_channel_CH2Cl2_1[fit_window_CH2Cl2_1[0][0]:fit_window_CH2Cl2_1[0][1]],␣
 ↪sub_spectrum_CH2Cl2_1[fit_window_CH2Cl2_1[0][0]:fit_window_CH2Cl2_1[0][1]],␣
 ↪p0=[500,515,0.1,0.1,20], bounds=(0,np.inf))
plt.plot(x_0,␣
 ↪voigt(x_0,p_CH2Cl2_1_0[0],p_CH2Cl2_1_0[1],p_CH2Cl2_1_0[2],p_CH2Cl2_1_0[3],p_CH2Cl2_1_0[4]))

x_1 = np.linspace(sub_channel_CH2Cl2_1[fit_window_CH2Cl2_1[1][0]],␣
 ↪sub_channel_CH2Cl2_1[fit_window_CH2Cl2_1[1][1]], 1000)
```

```
p_CH2Cl2_1_1, V_CH2Cl2_1_1 = curve_fit(voigt,␣
 ↪sub_channel_CH2Cl2_1[fit_window_CH2Cl2_1[1][0]:fit_window_CH2Cl2_1[1][1]],␣
 ↪sub_spectrum_CH2Cl2_1[fit_window_CH2Cl2_1[1][0]:fit_window_CH2Cl2_1[1][1]],␣
 ↪p0=[500,520,0.1,0.1,20], bounds=(0,np.inf))
plt.plot(x_1,␣
 ↪voigt(x_1,p_CH2Cl2_1_1[0],p_CH2Cl2_1_1[1],p_CH2Cl2_1_1[2],p_CH2Cl2_1_1[3],p_CH2Cl2_1_1[4]))

x_2 = np.linspace(sub_channel_CH2Cl2_1[fit_window_CH2Cl2_1[2][0]],␣
 ↪sub_channel_CH2Cl2_1[fit_window_CH2Cl2_1[2][1]], 1000)
#ohne Ausreiser bei 971
p_CH2Cl2_1_2, V_CH2Cl2_1_2 = curve_fit(voigt, np.
 ↪delete(sub_channel_CH2Cl2_1,971)[fit_window_CH2Cl2_1[2][0]:
 ↪fit_window_CH2Cl2_1[2][1]], np.
 ↪delete(sub_spectrum_CH2Cl2_1,971)[fit_window_CH2Cl2_1[2][0]:
 ↪fit_window_CH2Cl2_1[2][1]], p0=[500,523,0.1,0.1,20], bounds=(0,np.inf))
plt.plot(x_2,␣
 ↪voigt(x_2,p_CH2Cl2_1_2[0],p_CH2Cl2_1_2[1],p_CH2Cl2_1_2[2],p_CH2Cl2_1_2[3],p_CH2Cl2_1_2[4]))

x_3 = np.linspace(sub_channel_CH2Cl2_1[fit_window_CH2Cl2_1[3][0]],␣
 ↪sub_channel_CH2Cl2_1[fit_window_CH2Cl2_1[3][1]], 1000)
p_CH2Cl2_1_3, V_CH2Cl2_1_3 = curve_fit(voigt,␣
 ↪sub_channel_CH2Cl2_1[fit_window_CH2Cl2_1[3][0]:fit_window_CH2Cl2_1[3][1]],␣
 ↪sub_spectrum_CH2Cl2_1[fit_window_CH2Cl2_1[3][0]:fit_window_CH2Cl2_1[3][1]],␣
 ↪p0=[500,538,0.1,0.1,20], bounds=(0,np.inf))
plt.plot(x_3,␣
 ↪voigt(x_3,p_CH2Cl2_1_3[0],p_CH2Cl2_1_3[1],p_CH2Cl2_1_3[2],p_CH2Cl2_1_3[3],p_CH2Cl2_1_3[4]))

x_4 = np.linspace(sub_channel_CH2Cl2_1[fit_window_CH2Cl2_1[4][0]],␣
 ↪sub_channel_CH2Cl2_1[fit_window_CH2Cl2_1[4][1]], 1000)
p_CH2Cl2_1_4, V_CH2Cl2_1_4 = curve_fit(voigt,␣
 ↪sub_channel_CH2Cl2_1[fit_window_CH2Cl2_1[4][0]:fit_window_CH2Cl2_1[4][1]],␣
 ↪sub_spectrum_CH2Cl2_1[fit_window_CH2Cl2_1[4][0]:fit_window_CH2Cl2_1[4][1]],␣
 ↪p0=[500,550,0.1,0.1,20], bounds=(0,np.inf))
plt.plot(x_4,␣
 ↪voigt(x_4,p_CH2Cl2_1_4[0],p_CH2Cl2_1_4[1],p_CH2Cl2_1_4[2],p_CH2Cl2_1_4[3],p_CH2Cl2_1_4[4]))

x_5 = np.linspace(sub_channel_CH2Cl2_1[fit_window_CH2Cl2_1[5][0]],␣
 ↪sub_channel_CH2Cl2_1[fit_window_CH2Cl2_1[5][1]], 1000)
p_CH2Cl2_1_5, V_CH2Cl2_1_5 = curve_fit(voigt,␣
 ↪sub_channel_CH2Cl2_1[fit_window_CH2Cl2_1[5][0]:fit_window_CH2Cl2_1[5][1]],␣
 ↪sub_spectrum_CH2Cl2_1[fit_window_CH2Cl2_1[5][0]:fit_window_CH2Cl2_1[5][1]],␣
 ↪p0=[500,552,0.1,0.1,20], bounds=(0,np.inf))
plt.plot(x_5,␣
 ↪voigt(x_5,p_CH2Cl2_1_5[0],p_CH2Cl2_1_5[1],p_CH2Cl2_1_5[2],p_CH2Cl2_1_5[3],p_CH2Cl2_1_5[4]))

plt.savefig('Bilder/Raman_CH2Cl2_1_CCD.pdf', bbox_inches='tight')
```

[ ]:

```
peak_energy_CH2Cl2_1_0, delta_energy_CH2Cl2_1_0 = energy(p_CH2Cl2_1_0[1], np.
  ↪sqrt(8*np.log(2)*p_CH2Cl2_1_0[2]**2+4*p_CH2Cl2_1_0[3]**2))
nu_CH2Cl2_1_0, delta_nu_CH2Cl2_1_0 = peak_energy_CH2Cl2_1_0-laser_energy, np.
  ↪sqrt(delta_energy_CH2Cl2_1_0**2+delta_laser_energy**2)

peak_energy_CH2Cl2_1_1, delta_energy_CH2Cl2_1_1 = energy(p_CH2Cl2_1_1[1], np.
  ↪sqrt(8*np.log(2)*p_CH2Cl2_1_1[2]**2+4*p_CH2Cl2_1_1[3]**2))
nu_CH2Cl2_1_1, delta_nu_CH2Cl2_1_1 = peak_energy_CH2Cl2_1_1-laser_energy, np.
  ↪sqrt(delta_energy_CH2Cl2_1_1**2+delta_laser_energy**2)

peak_energy_CH2Cl2_1_2, delta_energy_CH2Cl2_1_2 = energy(p_CH2Cl2_1_2[1], np.
  ↪sqrt(8*np.log(2)*p_CH2Cl2_1_2[2]**2+4*p_CH2Cl2_1_2[3]**2))
nu_CH2Cl2_1_2, delta_nu_CH2Cl2_1_2 = peak_energy_CH2Cl2_1_2-laser_energy, np.
  ↪sqrt(delta_energy_CH2Cl2_1_2**2+delta_laser_energy**2)

peak_energy_CH2Cl2_1_3, delta_energy_CH2Cl2_1_3 = energy(p_CH2Cl2_1_3[1], np.
  ↪sqrt(8*np.log(2)*p_CH2Cl2_1_3[2]**2+4*p_CH2Cl2_1_3[3]**2))
nu_CH2Cl2_1_3, delta_nu_CH2Cl2_1_3 = peak_energy_CH2Cl2_1_3-laser_energy, np.
  ↪sqrt(delta_energy_CH2Cl2_1_3**2+delta_laser_energy**2)

peak_energy_CH2Cl2_1_4, delta_energy_CH2Cl2_1_4 = energy(p_CH2Cl2_1_4[1], np.
  ↪sqrt(8*np.log(2)*p_CH2Cl2_1_4[2]**2+4*p_CH2Cl2_1_4[3]**2))
nu_CH2Cl2_1_4, delta_nu_CH2Cl2_1_4 = peak_energy_CH2Cl2_1_4-laser_energy, np.
  ↪sqrt(delta_energy_CH2Cl2_1_4**2+delta_laser_energy**2)

peak_energy_CH2Cl2_1_5, delta_energy_CH2Cl2_1_5 = energy(p_CH2Cl2_1_5[1], np.
  ↪sqrt(8*np.log(2)*p_CH2Cl2_1_5[2]**2+4*p_CH2Cl2_1_5[3]**2))
nu_CH2Cl2_1_5, delta_nu_CH2Cl2_1_5 = peak_energy_CH2Cl2_1_5-laser_energy, np.
  ↪sqrt(delta_energy_CH2Cl2_1_5**2+delta_laser_energy**2)

print(nu_CH2Cl2_1_0, delta_nu_CH2Cl2_1_0)
print(nu_CH2Cl2_1_1, delta_nu_CH2Cl2_1_1)
print(nu_CH2Cl2_1_2, delta_nu_CH2Cl2_1_2)
print(nu_CH2Cl2_1_3, delta_nu_CH2Cl2_1_3)
print(nu_CH2Cl2_1_4, delta_nu_CH2Cl2_1_4)
print(nu_CH2Cl2_1_5, delta_nu_CH2Cl2_1_5)

print(lit_CH2Cl2)
```

0.6.5 CS2

[ ]:

```
window_CS2_1 = [1027, 1069]
channel_CS2_1, delta_channel_CS2_1, spectrum_CS2_1, delta_spectrum_CS2_1 =␣
  ↪spectrum2("measure_CS2_0.txt", "measure_CS2_4.txt", 100)
```

```
sub_channel_CS2_1, sub_spectrum_CS2_1, p_CS2_1, V_CS2_1 =␣
 ↪substract_fit(channel_CS2_1, delta_channel_CS2_1, spectrum_CS2_1,␣
 ↪delta_spectrum_CS2_1, window_CS2_1)

fit_window_CS2_1 = [[858, 878],[920,945],[945, 970], [1170,1240],[1245,1270]]

plt.subplots(figsize=(16,9))
plt.errorbar(sub_channel_CS2_1,␣
 ↪sub_spectrum_CS2_1,yerr=list(delta_spectrum_CS2_1[0:
 ↪window_CS2_1[0]])+list(delta_spectrum_CS2_1[window_CS2_1[1]:-1]),␣
 ↪ecolor="black", marker="x", ls="", color="green", markersize=5, capsize=3,␣
 ↪label="spectrum")
for i in fit_window_CS2_1:
    plt.
 ↪fill_betweenx([-1000,70000],sub_channel_CS2_1[i[0]],sub_channel_CS2_1[i[1]],␣
 ↪color='lightgrey')
plt.legend()
plt.xlim(500,560)
#plt.title('Raman spectrum of CS$_2$ with 90° polarization, without $\\lambda/
 ↪2$-plate')
#plt.xlim(calibration(i[0])[0]-5,calibration(i[1])[0]+5)
plt.ylim(-1000, 6000)
plt.grid()
plt.xlabel("wavelength $\\lambda$ [nm]")
plt.ylabel("Intensity in [a.u.]")
#plt.vlines(calibration(970),0,3000, color='red')

x_0 = np.linspace(sub_channel_CS2_1[fit_window_CS2_1[0][0]],␣
 ↪sub_channel_CS2_1[fit_window_CS2_1[0][1]], 1000)
p_CS2_1_0, V_CS2_1_0 = curve_fit(voigt,␣
 ↪sub_channel_CS2_1[fit_window_CS2_1[0][0]:fit_window_CS2_1[0][1]],␣
 ↪sub_spectrum_CS2_1[fit_window_CS2_1[0][0]:fit_window_CS2_1[0][1]],␣
 ↪p0=[1000,510,0.1,0.1,20], bounds=(0,np.inf))
plt.plot(x_0,␣
 ↪voigt(x_0,p_CS2_1_0[0],p_CS2_1_0[1],p_CS2_1_0[2],p_CS2_1_0[3],p_CS2_1_0[4]))

x_1 = np.linspace(sub_channel_CS2_1[fit_window_CS2_1[1][0]],␣
 ↪sub_channel_CS2_1[fit_window_CS2_1[1][1]], 1000)
p_CS2_1_1, V_CS2_1_1 = curve_fit(voigt,␣
 ↪sub_channel_CS2_1[fit_window_CS2_1[1][0]:fit_window_CS2_1[1][1]],␣
 ↪sub_spectrum_CS2_1[fit_window_CS2_1[1][0]:fit_window_CS2_1[1][1]],␣
 ↪p0=[1000,517,0.1,0.1,20], bounds=(0,np.inf))
plt.plot(x_1,␣
 ↪voigt(x_1,p_CS2_1_1[0],p_CS2_1_1[1],p_CS2_1_1[2],p_CS2_1_1[3],p_CS2_1_1[4]))
```

```
x_2 = np.linspace(sub_channel_CS2_1[fit_window_CS2_1[2][0]],␣
 ↪sub_channel_CS2_1[fit_window_CS2_1[2][1]], 1000)
#ohne Ausreiser bei 971
p_CS2_1_2, V_CS2_1_2 = curve_fit(voigt, np.
 ↪delete(sub_channel_CS2_1,971)[fit_window_CS2_1[2][0]:
 ↪fit_window_CS2_1[2][1]], np.
 ↪delete(sub_spectrum_CS2_1,971)[fit_window_CS2_1[2][0]:
 ↪fit_window_CS2_1[2][1]], p0=[1000,520,0.1,0.1,20], bounds=(0,np.inf))
plt.plot(x_2,␣
 ↪voigt(x_2,p_CS2_1_2[0],p_CS2_1_2[1],p_CS2_1_2[2],p_CS2_1_2[3],p_CS2_1_2[4]))

x_3 = np.linspace(sub_channel_CS2_1[fit_window_CS2_1[3][0]],␣
 ↪sub_channel_CS2_1[fit_window_CS2_1[3][1]], 1000)
p_CS2_1_3, V_CS2_1_3 = curve_fit(voigt,␣
 ↪sub_channel_CS2_1[fit_window_CS2_1[3][0]:fit_window_CS2_1[3][1]],␣
 ↪sub_spectrum_CS2_1[fit_window_CS2_1[3][0]:fit_window_CS2_1[3][1]],␣
 ↪p0=[6000,548,1,1,200], bounds=(0,np.inf))
plt.plot(x_3,␣
 ↪voigt(x_3,p_CS2_1_3[0],p_CS2_1_3[1],p_CS2_1_3[2],p_CS2_1_3[3],p_CS2_1_3[4]))

x_4 = np.linspace(sub_channel_CS2_1[fit_window_CS2_1[4][0]],␣
 ↪sub_channel_CS2_1[fit_window_CS2_1[4][1]], 1000)
p_CS2_1_4, V_CS2_1_4 = curve_fit(voigt,␣
 ↪sub_channel_CS2_1[fit_window_CS2_1[4][0]:fit_window_CS2_1[4][1]],␣
 ↪sub_spectrum_CS2_1[fit_window_CS2_1[4][0]:fit_window_CS2_1[4][1]],␣
 ↪p0=[500,550,0.1,0.1,20], bounds=(0,np.inf))
plt.plot(x_4,␣
 ↪voigt(x_4,p_CS2_1_4[0],p_CS2_1_4[1],p_CS2_1_4[2],p_CS2_1_4[3],p_CS2_1_4[4]))

plt.savefig('Bilder/Raman_CS2_1_CCD.pdf', bbox_inches='tight')
```

[ ]:

```
peak_energy_CS2_1_0, delta_energy_CS2_1_0 = energy(p_CS2_1_0[1], np.sqrt(8*np.
 ↪log(2)*p_CS2_1_0[2]**2+4*p_CS2_1_0[3]**2))
nu_CS2_1_0, delta_nu_CS2_1_0 = peak_energy_CS2_1_0-laser_energy, np.
 ↪sqrt(delta_energy_CS2_1_0**2+delta_laser_energy**2)

peak_energy_CS2_1_1, delta_energy_CS2_1_1 = energy(p_CS2_1_1[1], np.sqrt(8*np.
 ↪log(2)*p_CS2_1_1[2]**2+4*p_CS2_1_1[3]**2))
nu_CS2_1_1, delta_nu_CS2_1_1 = peak_energy_CS2_1_1-laser_energy, np.
 ↪sqrt(delta_energy_CS2_1_1**2+delta_laser_energy**2)

peak_energy_CS2_1_2, delta_energy_CS2_1_2 = energy(p_CS2_1_2[1], np.sqrt(8*np.
 ↪log(2)*p_CS2_1_2[2]**2+4*p_CS2_1_2[3]**2))
nu_CS2_1_2, delta_nu_CS2_1_2 = peak_energy_CS2_1_2-laser_energy, np.
 ↪sqrt(delta_energy_CS2_1_2**2+delta_laser_energy**2)
```

```
peak_energy_CS2_1_3, delta_energy_CS2_1_3 = energy(p_CS2_1_3[1], np.sqrt(8*np.
 ↪log(2)*p_CS2_1_3[2]**2+4*p_CS2_1_3[3]**2))
nu_CS2_1_3, delta_nu_CS2_1_3 = peak_energy_CS2_1_3-laser_energy, np.
 ↪sqrt(delta_energy_CS2_1_3**2+delta_laser_energy**2)

peak_energy_CS2_1_4, delta_energy_CS2_1_4 = energy(p_CS2_1_4[1], np.sqrt(8*np.
 ↪log(2)*p_CS2_1_4[2]**2+4*p_CS2_1_4[3]**2))
nu_CS2_1_4, delta_nu_CS2_1_4 = peak_energy_CS2_1_4-laser_energy, np.
 ↪sqrt(delta_energy_CS2_1_4**2+delta_laser_energy**2)

print(nu_CS2_1_0, delta_nu_CS2_1_0)
print(nu_CS2_1_1, delta_nu_CS2_1_1)
print(nu_CS2_1_2, delta_nu_CS2_1_2)
print(nu_CS2_1_3, delta_nu_CS2_1_3)
print(nu_CS2_1_4, delta_nu_CS2_1_4)

print(lit_CS2)

print((nu_CS2_1_0-lit_CS2[1])/delta_nu_CS2_1_0)
print((nu_CS2_1_1-lit_CS2[0])/delta_nu_CS2_1_1)
print((nu_CS2_1_2-lit_CS2[0])/delta_nu_CS2_1_2)
print((nu_CS2_1_3+lit_CS2[1])/delta_nu_CS2_1_3)
print((nu_CS2_1_4+lit_CS2[1])/delta_nu_CS2_1_4)
print((nu_CS2_1_4+lit_CS2[2])/delta_nu_CS2_1_4)
```

0.7 Ethanol und Wasser

[ ]:

```
ethanol_liste = ['03','07','09','13', '15', '01', '17', '19', '21','05']
ethanol_conc = [100, 90, 80, 70, 60, 50, 40, 30, 20, 0]
ethanol_channel = [0]*len(ethanol_liste)
delta_ethanol_channel = [0]*len(ethanol_liste)
ethanol_spectrum = [0]*len(ethanol_liste)
delta_ethanol_spectrum = [0]*len(ethanol_liste)
ethanol_colors = ['darkred', 'red', 'crimson', 'deeppink', 'mediumvioletred',␣
 ↪'mediumorchid', 'darkorchid', 'blueviolet', 'blue', 'darkblue']

for i in range(len(ethanol_liste)):
    ethanol_channel[i], delta_ethanol_channel[i], ethanol_spectrum[i],␣
 ↪delta_ethanol_spectrum[i] = spectrum2("measure_ethanol_"+ethanol_liste[i]+".
 ↪txt", "measure_ethanol_00.txt", 100)
```

[ ]:

```
plt.subplots(figsize=(16,9))
#plt.semilogy()
for i in range(len(ethanol_liste)):
    plt.errorbar(ethanol_channel[i], ethanol_spectrum[i],␣
 ↪color=ethanol_colors[i], ecolor="black", marker="x", ls="", markersize=5,␣
 ↪capsize=3, label=f'{ethanol_conc[i]}%')
```

```
plt.legend()
plt.grid()
plt.xlim(540,600)
plt.ylim(-50,1000)
plt.xlabel("wavelength $\\lambda_C$ [nm]")
plt.ylabel("Intensity [a.u.]")

plt.savefig('Bilder/ethanol_CCD.pdf', bbox_inches='tight')
```

[ ]:

```
kerb_ethanol_liste = ['04','08','10','14', '16', '02', '18', '20', '22','06']
kerb_ethanol_conc = [100, 90, 80, 70, 60, 50, 40, 30, 20, 0]
delta_kerb_ethanol_conc = [0]+[i*0.5/np.sqrt(6)*np.sqrt((1/(i/10)-1/(10))**2+1/
 ↪100) for i in kerb_ethanol_conc[1:9]]+[0]
kerb_ethanol_channel = [0]*len(kerb_ethanol_liste)
delta_kerb_ethanol_channel = [0]*len(kerb_ethanol_liste)
kerb_ethanol_spectrum = [0]*len(kerb_ethanol_liste)
delta_kerb_ethanol_spectrum = [0]*len(kerb_ethanol_liste)
kerb_ethanol_colors = ['darkred', 'red', 'crimson', 'deeppink',␣
 ↪'mediumvioletred', 'mediumorchid', 'darkorchid', 'blueviolet', 'blue',␣
 ↪'darkblue']


for i in range(len(kerb_ethanol_liste)):
    kerb_ethanol_channel[i], delta_kerb_ethanol_channel[i],␣
 ↪kerb_ethanol_spectrum[i], delta_kerb_ethanol_spectrum[i] =␣
 ↪spectrum2("measure_ethanol_"+kerb_ethanol_liste[i]+".txt",␣
 ↪"measure_ethanol_00.txt", 100)
```

[ ]:

```
plt.subplots(figsize=(16,9))
plot_liste_ethanol = [8]
kerb_ethanol_window = [1520, 1555]
kerb_ethanol_p0 = [[100,587,0.1,0.1,250],[100,587,0.1,0.1,200],[100,587,0.1,0.
 ↪1,200],[100,587,0.1,0.1,250],[100,587,0.1,0.1,250],[100,587,0.1,0.
 ↪1,250],[100,587,0.1,0.1,20],[100,587,0.1,0.1,20],[20,587,0.1,0.
 ↪1,20],[100,587,0.1,0.1,250]]
kerb_ethanol_p = []
kerb_ethanol_Vp = []
kerb_ethanol_x=np.
 ↪linspace(kerb_ethanol_channel[0][kerb_ethanol_window[0]],kerb_ethanol_channel[0][kerb_ethan
#plt.semilogy()
for i in range(len(kerb_ethanol_liste)):
    p_e, V_e = curve_fit(voigt, kerb_ethanol_channel[i][kerb_ethanol_window[0]:
 ↪kerb_ethanol_window[1]], kerb_ethanol_spectrum[i][kerb_ethanol_window[0]:
 ↪kerb_ethanol_window[1]], p0=kerb_ethanol_p0[i], bounds=(0,np.inf))
    kerb_ethanol_p.append(p_e)
    kerb_ethanol_Vp.append(V_e)
```

```
    plt.
  ↪plot(kerb_ethanol_x,voigt(kerb_ethanol_x,p_e[0],p_e[1],p_e[2],p_e[3],p_e[4]))
    plt.fill_betweenx([0,1000],␣
  ↪kerb_ethanol_channel[i][kerb_ethanol_window[0]],␣
  ↪kerb_ethanol_channel[i][kerb_ethanol_window[1]], color='lightgrey')
    plt.errorbar(kerb_ethanol_channel[i], kerb_ethanol_spectrum[i],␣
  ↪color=kerb_ethanol_colors[i], ecolor="black", marker="x", ls="",␣
  ↪markersize=5, capsize=3, label=f'{kerb_ethanol_conc[i]}%')
    plt.
  ↪plot(kerb_ethanol_x,voigt(kerb_ethanol_x,p_e[0],p_e[1],p_e[2],p_e[3],p_e[4]),␣
  ↪color='black')
plt.legend()
plt.grid()
plt.xlim(540,600)
plt.ylim(-50,1000)
plt.xlabel("wavelength $\\lambda_C$ [nm]")
plt.ylabel("Intensity [a.u.]")

plt.savefig('Bilder/kerb_ethanol_CCD.pdf', bbox_inches='tight')
```

```
[ ]: kerb_ethanol_N=[i[0] for i in kerb_ethanol_p]
     delta_kerb_ethanol_N = [np.sqrt(i[0][0]) for i in kerb_ethanol_Vp]

     p_conc, Vp_conc = curve_fit(f, kerb_ethanol_conc, kerb_ethanol_N,␣
       ↪sigma=delta_kerb_ethanol_N, absolute_sigma=True)
     x = np.linspace(0,100,1000)
     mod = Modell(x,p_conc)
     konf = Konfidenz(x,p_conc,Vp_conc)


     plt.subplots(figsize=(16,9))

     plt.plot(x, mod, label="linear regression", color="darkolivegreen")
     plt.plot(x, konf[0],linestyle="dashed", color="grey", label="confidence band")
     plt.plot(x, konf[1],linestyle="dashed", color="grey")
     plt.fill_between(x,konf[0],konf[1],color="whitesmoke")

     plt.errorbar(kerb_ethanol_conc, kerb_ethanol_N, xerr=delta_kerb_ethanol_conc,␣
       ↪yerr=delta_kerb_ethanol_N, color='green', ecolor="black", marker="x", ls="",␣
       ↪markersize=10, capsize=3, label=f'Intensity')
     plt.legend()
     plt.grid()
     plt.ylim(-50,750)
     plt.xlabel("Ethanol concentration $c$ [%]")
     plt.ylabel("Intensity [a.u.]")
     plt.savefig('Bilder/ethanol_calibration_CCD.pdf', bbox_inches='tight')
     print(p_conc, np.sqrt(Vp_conc))
```

```
def concentration(I, delta_I):
    return (I-p_conc[1])/p_conc[0], np.sqrt(delta_I**2/
  ↪p_conc[0]**2+Vp_conc[1][1]/p_conc[0]**2+(I-p_conc[1])**2/
  ↪p_conc[0]**4*Vp_conc[0][0]+2*(I-p_conc[1])/p_conc[0]**3*Vp_conc[0][1])
```

```
[ ]: unknown_channel, delta_unknown_channel, unknown_spectrum,␣
       ↪delta_unknown_spectrum = spectrum2("measure_ethanol_12.txt",␣
       ↪"measure_ethanol_00.txt", 100)
```

```
[ ]: plt.subplots(figsize=(16,9))
     plot_liste_ethanol = [8]
     kerb_ethanol_window = [1520, 1555]
     unknown_p0 = [100,587,0.1,0.1,150]

     kerb_ethanol_x=np.
       ↪linspace(kerb_ethanol_channel[0][kerb_ethanol_window[0]],kerb_ethanol_channel[0][kerb_ethan
     unknown_p, unknown_Vp = curve_fit(voigt, unknown_channel[kerb_ethanol_window[0]:
       ↪kerb_ethanol_window[1]], unknown_spectrum[kerb_ethanol_window[0]:
       ↪kerb_ethanol_window[1]], p0=unknown_p0, bounds=(0,np.inf))

     plt.fill_betweenx([0,1000], unknown_channel[kerb_ethanol_window[0]],␣
       ↪unknown_channel[kerb_ethanol_window[1]], color='lightgrey')
     plt.errorbar(unknown_channel, unknown_spectrum, ecolor="black", marker="x",␣
       ↪ls="", markersize=5, capsize=3, label='unknown concentration')
     plt.
       ↪plot(kerb_ethanol_x,voigt(kerb_ethanol_x,unknown_p[0],unknown_p[1],unknown_p[2],unknown_p[3
       ↪color='black')
     plt.legend()
     plt.grid()
     plt.xlim(540,600)
     plt.ylim(-50,1000)
     plt.xlabel("wavelength $\\lambda_C$ [nm]")
     plt.ylabel("Intensity [a.u.]")
     plt.savefig('Bilder/unknown_ethanol_CCD.pdf', bbox_inches='tight')
```

```
[ ]: unknown_concentration, delta_unknown_concentration =␣
       ↪concentration(unknown_p[0], np.sqrt(unknown_Vp[0][0]))
     unknown_concentration, delta_unknown_concentration
```

0.8 Schwefel

0.8.1 Vorbereitung

```
[ ]: window_sulfur = [1043, 1052]
     current_sulfur = [1.5,1.3,1.4,1.2,1.1,1.0]
     fit_window_sulfur = [[900,935], [1164,1230]]
     names_sulfur = ['01','02','03','04','05', '06']
```

```
channel_sulfur = []
spectrum_sulfur = []
delta_channel_sulfur = []
delta_spectrum_sulfur = []
p_sulfur_1 = []
Vp_sulfur_1 = []
p_sulfur_2 = []
Vp_sulfur_2 = []

for i in range(len(names_sulfur)):
    c, dc, s, ds = spectrum2('measure_sulfur_'+names_sulfur[i]+'.txt',␣
 ↪"measure_sulfur_00.txt", 100)
    channel_sulfur.append(c)
    delta_channel_sulfur.append(dc)
    spectrum_sulfur.append(s)
    delta_spectrum_sulfur.append(ds)

    plt.subplots(figsize=(16,9))
    plt.errorbar(channel_sulfur[i],␣
 ↪spectrum_sulfur[i],yerr=delta_spectrum_sulfur[i], ecolor="black",␣
 ↪marker="x", ls="", color="green", markersize=5, capsize=3,␣
 ↪label=f"I={current_sulfur[i]} A")
    for j in fit_window_sulfur:
        plt.fill_betweenx([0,70000], channel_sulfur[i][j[0]],␣
 ↪channel_sulfur[i][j[1]], color='lightgrey')

    x_sulfur_1 = np.
 ↪linspace(channel_sulfur[i][fit_window_sulfur[0][0]],channel_sulfur[i][fit_window_sulfur[0][
    p_s_1, V_s_1 = curve_fit(voigt, channel_sulfur[i][fit_window_sulfur[0][0]:
 ↪fit_window_sulfur[0][1]], spectrum_sulfur[i][fit_window_sulfur[0][0]:
 ↪fit_window_sulfur[0][1]], p0=[1000,517,0.1,0.1,200], bounds=(0,np.inf))
    p_sulfur_1.append(p_s_1)
    Vp_sulfur_1.append(V_s_1)
    plt.plot(x_sulfur_1,␣
 ↪voigt(x_sulfur_1,p_sulfur_1[i][0],p_sulfur_1[i][1],p_sulfur_1[i][2],p_sulfur_1[i][3],p_sulf

    x_sulfur_2 = np.
 ↪linspace(channel_sulfur[i][fit_window_sulfur[1][0]],channel_sulfur[i][fit_window_sulfur[1][
    p_s_2, V_s_2 = curve_fit(voigt, channel_sulfur[i][fit_window_sulfur[1][0]:
 ↪fit_window_sulfur[1][1]], spectrum_sulfur[i][fit_window_sulfur[1][0]:
 ↪fit_window_sulfur[1][1]], p0=[8000,544,0.1,0.1,1600], bounds=(0,np.inf))
    p_sulfur_2.append(p_s_2)
    Vp_sulfur_2.append(V_s_2)
```

```
        plt.plot(x_sulfur_2,␣
     ↪voigt(x_sulfur_2,p_sulfur_2[i][0],p_sulfur_2[i][1],p_sulfur_2[i][2],p_sulfur_2[i][3],p_sulf

        plt.legend()
        plt.xlim(500,560)
        plt.ylim(0, 12000)
        plt.grid()
        plt.xlabel("wavelength $\\lambda$ [nm]")
        plt.ylabel("Intensity in [a.u.]")
        plt.savefig('Bilder/sulfur_'+names_sulfur[i]+'.pdf', bbox_inches='tight')
```

```
[ ]: energy_sulfur_s = np.array([energy(i[1], np.sqrt(j[1][1])) for i, j in␣
      ↪zip(p_sulfur_2, Vp_sulfur_2)])
     energy_sulfur_a = np.array([energy(i[1], np.sqrt(j[1][1])) for i, j in␣
      ↪zip(p_sulfur_1, Vp_sulfur_1)])
     nu_sulfur_s = np.array([(i[0]-laser_energy, np.
      ↪sqrt(i[1]**2+delta_laser_energy**2)) for i in energy_sulfur_s])
     nu_sulfur_a = np.array([(i[0]-laser_energy, np.
      ↪sqrt(i[1]**2+delta_laser_energy**2)) for i in energy_sulfur_a])
     I_sulfur_s = [(i[0], np.sqrt(j[0][0])) for i, j in zip(p_sulfur_2, Vp_sulfur_2)]
     I_sulfur_a = [(i[0], np.sqrt(j[0][0])) for i, j in zip(p_sulfur_1, Vp_sulfur_1)]

     h=2*np.pi

     def T(nu_s, nu_a, I_s, I_a, delta_nu_s, delta_nu_a, delta_I_s, delta_I_a):
         convert = 1.973*10**(-7)/(10**(-2))
         delta_nu = (nu_a-nu_s)/2*convert #energy difference in eV
         t = h*delta_nu/(np.log(I_s/I_a)+4*np.log(nu_a/nu_s)) #energy difference in␣
      ↪eV
         denom = (np.log(I_s/I_a)+4*np.log(nu_a/nu_s)) # denominator
         delta_t = np.sqrt((t/(I_s*denom)*delta_I_s)**2+(t/
      ↪(I_a*denom)*delta_I_a)**2+(-4*t/(nu_a*denom)+h/
      ↪(2*denom))**2*(convert*delta_nu_a)**2+(4*t/(nu_s*denom)-h/
      ↪(2*denom))**2*(convert*delta_nu_s)**2)
         return t*1.16e4, delta_t*1.16e4 # temperatur in K
```

```
[ ]: energy_sulfur_s_mean = (np.mean(energy_sulfur_s[:,0]),stdmw(energy_sulfur_s[:
      ↪,1]))
     energy_sulfur_a_mean = (np.mean(energy_sulfur_a[:,0]),stdmw(energy_sulfur_a[:
      ↪,1]))
     nu_sulfur_s_mean = (np.mean(nu_sulfur_s[:,0]),stdmw(nu_sulfur_s[:,1]))
     nu_sulfur_a_mean = (np.mean(nu_sulfur_a[:,0]),stdmw(nu_sulfur_a[:,1]))
     energy_sulfur_s_mean, energy_sulfur_a_mean, nu_sulfur_s_mean, nu_sulfur_a_mean
```

```
[ ]: T_sulfur = np.array([T(i[0],j[0],k[0],l[0],i[1],j[1],k[1],l[1]) for i, j, k, l␣
      ↪in zip(energy_sulfur_s, energy_sulfur_a, I_sulfur_s, I_sulfur_a)])
```

```
T_sulfur_mean = (np.mean(T_sulfur[:,0]), stdmw(T_sulfur[:,1]))
```

```
[ ]: T_sulfur, T_sulfur_mean
```

```
[ ]:
```

7.2.2 Monochromator Analysis

Raman_Monochromator

April 15, 2024

[ ]:

```
import matplotlib.pyplot as plt
import numpy as np
from scipy.special import voigt_profile
from scipy.optimize import curve_fit
import matplotlib.ticker as ticker
SMALL_SIZE = 20
MEDIUM_SIZE = 20
BIGGER_SIZE = 20
plt.rc('font', size=SMALL_SIZE) # controls default text sizes
plt.rc('axes', titlesize=MEDIUM_SIZE) # fontsize of the axes title
plt.rc('axes', labelsize=SMALL_SIZE) # fontsize of the x and y labels
plt.rc('xtick', labelsize=SMALL_SIZE) # fontsize of the tick labels
plt.rc('ytick', labelsize=SMALL_SIZE) # fontsize of the tick labels
plt.rc('legend', fontsize=SMALL_SIZE) # legend fontsize
plt.rc('figure', titlesize=BIGGER_SIZE) # fontsize of the figure title
import matplotlib as mpl
import matplotlib.font_manager as font_manager
mpl.rcParams['font.family']='serif'
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
```

[ ]:

```
def f(x, a, b):
    return a*x+b

def gauss(x, A, mu, sig_squared, y0):
    return A*np.exp(-(x-mu)**2/(2*sig_squared)) + y0

def voigt(x, A, mu, sig, gam, y0):
    return A*voigt_profile((x-mu), sig, gam)+y0

def chiq(y, y_err, mod_y):
    return sum([(j-i)**2/delj**2 for i,j,delj in zip(y, mod_y, y_err)])

def red_chiq(y, y_err, mod_y, par):
```

```
    return chiq(y, y_err, mod_y)/(len(y)-par)

def stdmw(liste):
    return 1/len(liste)*np.sqrt(sum([(i-np.mean(liste))**2 for i in liste]))

def Modell(xbereich, p):
    return [p[0]*i+p[1] for i in xbereich]

def Konfidenz(xbereich, p, Vp):
    des = [np.sqrt(np.matmul(np.transpose(np.array([i, 1])),np.matmul(Vp,np.
 ↪array([i, 1])))) for i in xbereich]
    return [[p[0]*i+p[1]+des[j] for i,j in␣
 ↪zip(xbereich,range(len(xbereich)))],[p[0]*i+p[1]-des[j] for i,j in␣
 ↪zip(xbereich,range(len(xbereich)))]]
```

[ ]:
```
import pandas as pd



def read(file):
    data = pd.read_table(file, header=17,decimal=',', encoding='latin1',␣
 ↪skipfooter=1, names=['channel', 'spectrum'],engine='python')
    channel = np.array(data.channel)
    spectrum = np.array(data.spectrum)
    delta_channel = 1/np.sqrt(3)
    delta_spectrum = np.sqrt(spectrum)
    return channel, delta_channel, spectrum, delta_spectrum

def background(file, back_file):
    channel_scan, delta_channel_scan, spectrum_scan, delta_spectrum_scan =␣
 ↪read(file)
    channel_back, delta_channel_back, spectrum_back, delta_spectrum_back =␣
 ↪read(back_file)
    spectrum = spectrum_scan-spectrum_back
    delta_spectrum = np.sqrt(delta_spectrum_scan**2+delta_spectrum_back**2)
    return channel_scan, delta_channel_scan, spectrum, delta_spectrum
```

0.1 Setup and Calibration

[ ]:
```
channel_hg_90, delta_channel_hg_90, spectrum_hg_90, delta_spectrum_hg_90 =␣
 ↪background('hg_spectrum.dig', 'hg_background.dig')
```

[ ]:
```
calib_window = [[75, 105],[1190, 1240],[2260, 2350],[2905, 2950],[2951, 2985]]
calib_energies = [435.83, 491.6,546.07,576.96,579.07]

plt.subplots(figsize=(16,9))
#plt.semilogy()
```

```
plt.errorbar(channel_hg_90, spectrum_hg_90, xerr=delta_channel_hg_90,␣
 ↪yerr=delta_spectrum_hg_90, ecolor="black", marker="o", ls="", color="green",␣
 ↪markersize=3, capsize=3, label="measured spectrum")
for w in calib_window:
    plt.
 ↪fill_betweenx([0,250000],channel_hg_90[w[0]],channel_hg_90[w[1]],color='lightgrey')

calib_channels = []
delta_calib_channels = []
chi_calib = []

x1 = np.linspace(channel_hg_90[calib_window[0][0]],␣
 ↪channel_hg_90[calib_window[0][1]], 5000)
p1, V1 = curve_fit(voigt, channel_hg_90[calib_window[0][0]:calib_window[0][1]],␣
 ↪spectrum_hg_90[calib_window[0][0]:calib_window[0][1]],␣
 ↪p0=[200000,4300,20,20,0], sigma=delta_spectrum_hg_90[calib_window[0][0]:
 ↪calib_window[0][1]], absolute_sigma=True, bounds=(0,np.inf))
plt.plot(x1, voigt(x1,p1[0],p1[1],p1[2],p1[3],p1[4]), label='fit 1. peak')
calib_channels.append(p1[1])
delta_calib_channels.append(np.sqrt(8*np.log(2)*p1[2]**2+4*p1[3]**2))
chi_calib.append(red_chiq(spectrum_hg_90[calib_window[0][0]:
 ↪calib_window[0][1]], delta_spectrum_hg_90[calib_window[0][0]:
 ↪calib_window[0][1]], voigt(channel_hg_90[calib_window[0][0]:
 ↪calib_window[0][1]], p1[0],p1[1],p1[2],p1[3],p1[4]), 5))

x2 = np.linspace(channel_hg_90[calib_window[1][0]],␣
 ↪channel_hg_90[calib_window[1][1]], 5000)
p2, V2 = curve_fit(voigt, channel_hg_90[calib_window[1][0]:calib_window[1][1]],␣
 ↪spectrum_hg_90[calib_window[1][0]:calib_window[1][1]],␣
 ↪p0=[200000,4900,20,20,0], sigma=delta_spectrum_hg_90[calib_window[1][0]:
 ↪calib_window[1][1]], absolute_sigma=True, bounds=(0,np.inf))
plt.plot(x2, voigt(x2,p2[0],p2[1],p2[2],p2[3],p2[4]), label='fit 2. peak')
calib_channels.append(p2[1])
delta_calib_channels.append(np.sqrt(8*np.log(2)*p2[2]**2+4*p2[3]**2))
chi_calib.append(red_chiq(spectrum_hg_90[calib_window[1][0]:
 ↪calib_window[1][1]], delta_spectrum_hg_90[calib_window[1][0]:
 ↪calib_window[1][1]], voigt(channel_hg_90[calib_window[1][0]:
 ↪calib_window[1][1]], p2[0],p2[1],p2[2],p2[3],p2[4]), 5))

x3 = np.linspace(channel_hg_90[calib_window[2][0]],␣
 ↪channel_hg_90[calib_window[2][1]], 5000)
p3, V3 = curve_fit(voigt, channel_hg_90[calib_window[2][0]:calib_window[2][1]],␣
 ↪spectrum_hg_90[calib_window[2][0]:calib_window[2][1]],␣
 ↪p0=[200000,5400,20,20,0], sigma=delta_spectrum_hg_90[calib_window[2][0]:
 ↪calib_window[2][1]], absolute_sigma=True, bounds=(0,np.inf))
plt.plot(x3, voigt(x3,p3[0],p3[1],p3[2],p3[3],p3[4]), label='fit 3. peak')
```

```
calib_channels.append(p3[1])
delta_calib_channels.append(np.sqrt(8*np.log(2)*p3[2]**2+4*p3[3]**2))
chi_calib.append(red_chiq(spectrum_hg_90[calib_window[2][0]:
 ↪calib_window[2][1]], delta_spectrum_hg_90[calib_window[2][0]:
 ↪calib_window[2][1]], voigt(channel_hg_90[calib_window[2][0]:
 ↪calib_window[2][1]], p3[0],p3[1],p3[2],p3[3],p3[4]), 5))

x4 = np.linspace(channel_hg_90[calib_window[3][0]],␣
 ↪channel_hg_90[calib_window[3][1]], 5000)
p4, V4 = curve_fit(voigt, channel_hg_90[calib_window[3][0]:calib_window[3][1]],␣
 ↪spectrum_hg_90[calib_window[3][0]:calib_window[3][1]],␣
 ↪p0=[20000,5800,2,2,0], sigma=delta_spectrum_hg_90[calib_window[3][0]:
 ↪calib_window[3][1]], absolute_sigma=True, bounds=(0,np.inf))
plt.plot(x4, voigt(x4,p4[0],p4[1],p4[2],p4[3],p4[4]), label='fit 4. peak')
calib_channels.append(p4[1])
delta_calib_channels.append(np.sqrt(8*np.log(2)*p4[2]**2+4*p4[3]**2))
chi_calib.append(red_chiq(spectrum_hg_90[calib_window[3][0]:
 ↪calib_window[3][1]], delta_spectrum_hg_90[calib_window[3][0]:
 ↪calib_window[3][1]], voigt(channel_hg_90[calib_window[3][0]:
 ↪calib_window[3][1]], p4[0],p4[1],p4[2],p4[3],p4[4]), 5))

x5 = np.linspace(channel_hg_90[calib_window[4][0]],␣
 ↪channel_hg_90[calib_window[4][1]], 5000)
p5, V5 = curve_fit(voigt, channel_hg_90[calib_window[4][0]:calib_window[4][1]],␣
 ↪spectrum_hg_90[calib_window[4][0]:calib_window[4][1]],␣
 ↪p0=[200000,5800,2,2,0], sigma=delta_spectrum_hg_90[calib_window[4][0]:
 ↪calib_window[4][1]], absolute_sigma=True, bounds=(0,np.inf))
plt.plot(x5, voigt(x5,p5[0],p5[1],p5[2],p5[3],p5[4]), label='fit 5. peak')
calib_channels.append(p5[1])
delta_calib_channels.append(np.sqrt(8*np.log(2)*p5[2]**2+4*p5[3]**2))
chi_calib.append(red_chiq(spectrum_hg_90[calib_window[4][0]:
 ↪calib_window[4][1]], delta_spectrum_hg_90[calib_window[4][0]:
 ↪calib_window[4][1]], voigt(channel_hg_90[calib_window[4][0]:
 ↪calib_window[4][1]], p5[0],p5[1],p5[2],p5[3],p5[4]), 5))

plt.legend()
#plt.
 ↪xlim(channel_hg_90[calib_window[3][0]]-50,channel_hg_90[calib_window[3][1]]+50)
plt.ylim(5, 230000)
plt.grid()
#plt.title("Hg spectrum with 90° polarization")
plt.xlabel("Monochromator wavelength $\\lambda_m$ [$\\AA$]")
plt.ylabel("Counts")

plt.savefig('Bilder/calibration_spectrum_mono.pdf', bbox_inches='tight')
print(p1[2], p2[2], p3[2], p4[2], p5[2])
```

```
print(np.sqrt(V1[2][2]), np.sqrt(V2[2][2]), np.sqrt(V3[2][2]), np.
 ↪sqrt(V4[2][2]), np.sqrt(V5[2][2]))

print(calib_channels, delta_calib_channels, chi_calib)
```

[ ]:

```
plt.subplots(figsize=(16,9))
#plt.semilogy()
plt.errorbar(calib_channels, calib_energies, xerr=delta_calib_channels,␣
 ↪ecolor="black", marker="x", ls="", color="green", markersize=10, capsize=3,␣
 ↪label="energy of Hg peaks")


calib_p, calib_V = curve_fit(f, calib_channels, calib_energies)
x = np.linspace(4300,5900,20000)
mod = Modell(x,calib_p)
konf = Konfidenz(x,calib_p,calib_V)
plt.plot(x, mod, label="linear regression", color="darkolivegreen")
plt.plot(x, konf[0],linestyle="dashed", color="grey", label="confidence band")
plt.plot(x, konf[1],linestyle="dashed", color="grey")
plt.fill_between(x,konf[0],konf[1],color="whitesmoke")

plt.legend()
plt.grid()
plt.ylim(430, 585)
#plt.title("Calibration")
plt.xlabel('Monochromator wavelength $\\lambda_m$ [$\\AA$]')
plt.ylabel('wavelength [nm]')
plt.savefig('Bilder/calibration_linreg_mono.pdf', bbox_inches='tight')

print(calib_p, calib_V)
def calibration(channel, delta_channel=1/np.sqrt(3), p=calib_p, V=calib_V):
    return f(channel, p[0], p[1]), np.
 ↪sqrt(p[0]**2*delta_channel**2+channel**2*V[0][0]+V[1][1]+2*channel*V[1][0])
```

0.1.1 lower bound of Monochromator

[ ]:

```
bound_lambda, delta_bound_lambda, bound_spectrum, delta_bound_spectrum =␣
 ↪read('bound.dig')
bound_lambda, delta_bound_lambda = calibration(bound_lambda, delta_bound_lambda)

plt.subplots(figsize=(16,9))
#plt.semilogy()
plt.errorbar(bound_lambda, bound_spectrum, xerr=delta_bound_lambda,␣
 ↪yerr=delta_bound_spectrum, ecolor="black", marker="x", ls="", color="green",␣
 ↪markersize=5, capsize=3, label="measured spectrum")
plt.legend()
plt.grid()
```

```
plt.xlabel("wavelength $\\lambda_C$ [nm]")
plt.ylabel("Counts")

plt.savefig('Bilder/bound_mono.pdf', bbox_inches='tight')
```

0.1.2 Kerb filter

[ ]:

```
kerb_lambda, delta_kerb_lambda, kerb_spectrum, delta_kerb_spectrum =␣
  ↪read('kerb_filter_try.dig')
kerb_lambda, delta_kerb_lambda = calibration(kerb_lambda, delta_kerb_lambda)

plt.subplots(figsize=(16,9))
#plt.semilogy()
plt.errorbar(kerb_lambda, kerb_spectrum, xerr=delta_kerb_lambda,␣
  ↪yerr=delta_kerb_spectrum, ecolor="black", marker="x", ls="", color="green",␣
  ↪markersize=5, capsize=3, label="measured spectrum")
plt.legend()
plt.grid()
plt.xlabel("wavelength $\\lambda$ [nm]")
plt.ylabel("Counts")

plt.savefig('Bilder/kerb_mono.pdf', bbox_inches='tight')
```

0.1.3 Influence of slit width

[ ]:

```
slit_values = ['250','200', '150', '100', '050', '020']
slit_lambda = [0]*len(slit_values)
delta_slit_lambda = [0]*len(slit_values)
slit_spectrum = [0]*len(slit_values)
delta_slit_spectrum = [0]*len(slit_values)
slit_width = [250,200,150,100,50,20]

for i in range(len(slit_values)):
    slit_lambda[i], delta_slit_lambda[i], slit_spectrum[i],␣
  ↪delta_slit_spectrum[i] = read(f'slit_'+slit_values[i]+'.dig')
    slit_lambda[i], delta_slit_lambda[i] = calibration(slit_lambda[i],␣
  ↪delta_slit_lambda[i])
plt.subplots(figsize=(16,9))
#plt.semilogy()
for i in range(len(slit_values)):
    plt.plot(slit_lambda[i], slit_spectrum[i],␣
  ↪label=f'd={slit_width[i]}'+'$\\mu m$')
plt.legend()
plt.grid()
plt.xlabel("wavelength $\\lambda$ [nm]")
plt.ylabel("Counts")
```

```
plt.savefig('Bilder/width_mono.pdf', bbox_inches='tight')
```

0.1.4 Polarization

[ ]:

```
white_90_lambda, delta_white_90_lambda, white_90_spectrum,␣
  ↪delta_white_90_spectrum = read('white_90.dig')
white_90_lambda, delta_white_90_lambda = calibration(white_90_lambda,␣
  ↪delta_white_90_lambda)
white_00_lambda, delta_white_00_lambda, white_00_spectrum,␣
  ↪delta_white_00_spectrum = read('white_00.dig')
white_00_lambda, delta_white_00_lambda = calibration(white_00_lambda,␣
  ↪delta_white_00_lambda)
```

[ ]:

```
plt.subplots(figsize=(16,9))
#plt.semilogy()
plt.errorbar(white_90_lambda, white_90_spectrum, xerr=delta_white_90_lambda,␣
  ↪yerr=delta_white_90_spectrum, ecolor="black", marker="x", ls="",␣
  ↪color="green", markersize=5, capsize=3, label="measured spectrum 90°")
plt.errorbar(white_00_lambda, white_00_spectrum, xerr=delta_white_00_lambda,␣
  ↪yerr=delta_white_00_spectrum, ecolor="black", marker="x", ls="",␣
  ↪color="red", markersize=5, capsize=3, label="measured spectrum 0°")
plt.legend()
plt.grid()
plt.xlabel("wavelength $\\lambda$ [nm]")
plt.ylabel("Counts")

plt.savefig('Bilder/white_90_00_mono.pdf', bbox_inches='tight')
```

[ ]:

```
pol90zu00 = [i/j for i, j in zip(white_90_spectrum, white_00_spectrum)]
delta_pol90zu00 = [np.sqrt(di**2/j**2+i**2*dj**2/j**4) for i, j, di, dj in␣
  ↪zip(white_90_spectrum, white_00_spectrum, delta_white_90_spectrum,␣
  ↪delta_white_00_spectrum)]
```

[ ]:

```
plt.subplots(figsize=(16,9))

plt.errorbar(white_90_lambda, pol90zu00, xerr=delta_white_90_lambda,␣
  ↪yerr=delta_pol90zu00, ecolor="black", marker="x", ls="", alpha=0.5,␣
  ↪color="green", markersize=5, capsize=3, label='ratio of polarizations')

#plt.semilogy()
w=[1376, 2586]
plt.fill_betweenx([0.3,1.7], white_90_lambda[w[0]], white_90_lambda[w[1]],␣
  ↪color='lightgrey')
pol_p, pol_V = curve_fit(f, white_90_lambda[w[0]:w[1]], pol90zu00[w[0]:w[1]])
x = np.linspace(500,560,20000)
mod = Modell(x,pol_p)
konf = Konfidenz(x,pol_p,pol_V)
```

```
plt.plot(x, mod, label="linear regression", color="red")
plt.plot(x, konf[0],linestyle="dashed", color="grey", label="confidence band")
plt.plot(x, konf[1],linestyle="dashed", color="grey")
plt.fill_between(x,konf[0],konf[1],color="whitesmoke")
def pol(l, dl):
    return l*pol_p[0]+pol_p[1], np.
 ↪sqrt(dl**2*pol_p[0]**2+l**2*pol_V[0][0]+pol_V[1][1]+2*l*pol_V[1][0])

plt.legend()
plt.grid()
plt.ylim(0.3,1.7)
plt.xlabel("wavelength $\\lambda$ [nm]")
plt.ylabel("Intensity ratio of 90° to 0° polarization $r$")

plt.savefig('Bilder/white_90zu00_mono.pdf', bbox_inches='tight')
pol_p, pol_V
```

[ ]:

```
```

0.2 Raman spectroscopy

[ ]:

```
def energy(l, dl): #gives energy in cm^-1
    return 10**7/l, (10**7/l)**2*dl*10**(-7)
lit_CCl4 = np.array([217,314,459,776])
lit_dep_CCl4 = np.array([])
lit_CHCl3 = np.array([260,364,676,760])
lit_CH2Cl2 = np.array([])
lit_CS2 = np.array([397, 658, 1535])
```

0.2.1 CCl4 90° Polarization

[ ]:

```
CCl4_lambda, delta_CCl4_lambda, CCl4_spectrum, delta_CCl4_spectrum =␣
 ↪background('measure_CCl4.dig', 'measure_background.dig')
CCl4_lambda, delta_CCl4_lambda = calibration(CCl4_lambda, delta_CCl4_lambda)
```

[ ]:

```
CCl4_laser_p, CCl4_laser_Vp = curve_fit(voigt, CCl4_lambda, CCl4_spectrum,␣
 ↪sigma = delta_CCl4_spectrum, absolute_sigma=True, p0=[1000, 532, 0.1,0.1,0])
x_laser = np.linspace(500,560,5000)

CCl4_p = []
CCl4_Vp = []
CCl4_window = [[880,960], [965, 1005], [1010,1090], [1500, 1610], [1620, 1720],␣
 ↪[1770, 1920], [2150, 2320]]
CCl4_p0 = [[75, 523, 0.1,0.1,0],[50, 525, 0.1,0.1,20],[100, 527, 0.1,0.1, 30],␣
 ↪[150, 539,0.1,0.1,30], [150, 542,0.1,0.1,30], [200, 546,0.1,0.1,10],[50,␣
 ↪556,1,1,10]]
```

```
plt.subplots(figsize=(16,9))
#plt.semilogy()
plt.errorbar(CCl4_lambda, CCl4_spectrum, xerr=delta_CCl4_lambda,␣
 ↪yerr=delta_CCl4_spectrum, ecolor="black", marker="x", ls="", color="green",␣
 ↪markersize=5, capsize=3, label='measured spectrum')
plt.plot(x_laser,␣
 ↪voigt(x_laser,CCl4_laser_p[0],CCl4_laser_p[1],CCl4_laser_p[2],CCl4_laser_p[3],CCl4_laser_p[
 ↪color='red', label='fit of Rayleigh peak')

for i, w in enumerate(CCl4_window):
    plt.fill_betweenx([0,300], CCl4_lambda[w[0]], CCl4_lambda[w[1]],␣
 ↪color='lightgrey')
    lp, lV = curve_fit(voigt, CCl4_lambda[w[0]:w[1]], CCl4_spectrum[w[0]:w[1]],␣
 ↪sigma = delta_CCl4_spectrum[w[0]:w[1]], absolute_sigma=True, p0=CCl4_p0[i])
    plt.plot(np.linspace(CCl4_lambda[w[0]], CCl4_lambda[w[1]],5000), voigt(np.
 ↪linspace(CCl4_lambda[w[0]], CCl4_lambda[w[1]],5000), lp[0], lp[1], lp[2],␣
 ↪lp[3], lp[4]), color='red')
    CCl4_p.append(lp)
    CCl4_Vp.append(lV)
#plt.xlim(CCl4_lambda[CCl4_window[-1][0]]-5, CCl4_lambda[CCl4_window[-1][1]]+5)
plt.legend()
plt.grid()
plt.ylim(0,200)
#plt.xlim(500,560)
plt.xlabel("wavelength $\\lambda$ [nm]")
plt.ylabel("Counts")

plt.savefig('Bilder/CCl4_90_mono.pdf', bbox_inches='tight')
```

[ ]:

```
CCl4_energy = [energy(i[1], np.sqrt(di[1][1])) for i, di in zip(CCl4_p,␣
 ↪CCl4_Vp)]
CCl4_laser_wavelength = (CCl4_laser_p[1], np.sqrt(CCl4_laser_Vp[1][1]))
CCl4_laser_energy = energy(CCl4_laser_p[1], np.sqrt(CCl4_laser_Vp[1][1]))
CCl4_nu = [(i[0]-CCl4_laser_energy[0], np.
 ↪sqrt(i[1]**2+CCl4_laser_energy[1]**2)) for i in CCl4_energy]
print(f'Laserwavelength = {CCl4_laser_wavelength} nm')
print('Delta nu in cm^-1', CCl4_nu)
print('t 1 a', (CCl4_nu[2][0]-lit_CCl4[0])/CCl4_nu[2][1])
print('t 1 s', (CCl4_nu[3][0]+lit_CCl4[0])/CCl4_nu[3][1])
print('t 2 a', (CCl4_nu[1][0]-lit_CCl4[1])/CCl4_nu[1][1])
print('t 2 s', (CCl4_nu[4][0]+lit_CCl4[1])/CCl4_nu[4][1])
print('t 3 a', (CCl4_nu[0][0]-lit_CCl4[2])/CCl4_nu[0][1])
print('t 3 s', (CCl4_nu[5][0]+lit_CCl4[2])/CCl4_nu[5][1])
print('t 4 s', (CCl4_nu[6][0]+lit_CCl4[3])/CCl4_nu[6][1])
print('t laser', (CCl4_laser_wavelength[0]-532)/CCl4_laser_wavelength[1])
print(lit_CCl4)
```

0.2.2 CCl4 0° Polarization

```
[ ]: CCl4_00_lambda, delta_CCl4_00_lambda, CCl4_00_spectrum, delta_CCl4_00_spectrum␣
       ↪= background('measure_CCl4_00.dig', 'measure_background.dig')
     CCl4_00_lambda, delta_CCl4_00_lambda = calibration(CCl4_00_lambda,␣
       ↪delta_CCl4_00_lambda)
```

```
[ ]: CCl4_00_laser_p, CCl4_00_laser_Vp = curve_fit(voigt, CCl4_00_lambda,␣
       ↪CCl4_00_spectrum, sigma = delta_CCl4_00_spectrum, absolute_sigma=True,␣
       ↪p0=[1000, 532, 0.1,0.1,0])
     x_laser = np.linspace(500,560,5000)

     CCl4_00_p = []
     CCl4_00_Vp = []
     CCl4_00_window = [[870,970], [1050,1140], [1550, 1670], [1680, 1780], [1800,␣
       ↪2000], [2220, -1]]#]
     CCl4_00_p0 = [[75, 523, 0.1,0.1,0],[50, 528, 0.1,0.1,10], [150, 541,0.1,0.
       ↪1,30], [150, 543,0.1,0.1,30], [20, 548,0.1,0.1,0],[20, 558,1,1,0]]

     plt.subplots(figsize=(16,9))
     #plt.semilogy()
     plt.errorbar(CCl4_00_lambda, CCl4_00_spectrum, xerr=delta_CCl4_00_lambda,␣
       ↪yerr=delta_CCl4_00_spectrum, ecolor="black", marker="x", ls="",␣
       ↪color="green", markersize=5, capsize=3, label='measured spectrum')
     plt.plot(x_laser,␣
       ↪voigt(x_laser,CCl4_00_laser_p[0],CCl4_00_laser_p[1],CCl4_00_laser_p[2],CCl4_00_laser_p[3],C
       ↪color='red', label='fit of Rayleigh peak')

     for i, w in enumerate(CCl4_00_window):
         plt.fill_betweenx([0,300], CCl4_00_lambda[w[0]], CCl4_00_lambda[w[1]],␣
       ↪color='lightgrey')
         lp, lV = curve_fit(voigt, CCl4_00_lambda[w[0]:w[1]], CCl4_00_spectrum[w[0]:
       ↪w[1]], sigma = delta_CCl4_00_spectrum[w[0]:w[1]], absolute_sigma=True,␣
       ↪p0=CCl4_00_p0[i], bounds = (0, np.inf))
         plt.plot(np.linspace(CCl4_00_lambda[w[0]], CCl4_00_lambda[w[1]],5000),␣
       ↪voigt(np.linspace(CCl4_00_lambda[w[0]], CCl4_00_lambda[w[1]],5000), lp[0],␣
       ↪lp[1], lp[2], lp[3], lp[4]), color='red')
         CCl4_00_p.append(lp)
         CCl4_00_Vp.append(lV)
     #plt.xlim(CCl4_00_lambda[CCl4_00_window[-1][0]]-5,␣
       ↪CCl4_00_lambda[CCl4_00_window[-1][1]]+5)
     plt.legend()
     plt.grid()
     plt.ylim(0,200)
     #plt.xlim(500,560)
     plt.xlabel("wavelength $\\lambda$ [nm]")
     plt.ylabel("Counts")
```

```
plt.savefig('Bilder/CCl4_00_mono.pdf', bbox_inches='tight')
```

[ ]:

```
CCl4_00_energy = [energy(i[1], np.sqrt(di[1][1])) for i, di in zip(CCl4_00_p,␣
 ↪CCl4_00_Vp)]
CCl4_00_laser_wavelength = (CCl4_00_laser_p[1], np.sqrt(CCl4_00_laser_Vp[1][1]))
CCl4_00_laser_energy = energy(CCl4_00_laser_p[1], np.
 ↪sqrt(CCl4_00_laser_Vp[1][1]))
CCl4_00_nu = [(i[0]-CCl4_00_laser_energy[0], np.
 ↪sqrt(i[1]**2+CCl4_00_laser_energy[1]**2)) for i in CCl4_00_energy]
print(f'Laserwavelength = {CCl4_00_laser_wavelength} nm')
print('Delta nu in cm^-1',CCl4_00_nu)
print('t laser', (CCl4_00_laser_wavelength[0]-532)/CCl4_00_laser_wavelength[1])
```

[ ]:

```
print(CCl4_energy)
print(CCl4_00_energy)
```

0.2.3 Depolarization of CCl__4

[ ]:

```
def dep(I_para, I_perp, delta_I_para, delta_I_perp, position, delta_position,␣
 ↪lambdahalf=False):
    r, dr = pol(position, delta_position)
    if lambdahalf:
        return I_para/(I_perp*r), np.sqrt(delta_I_para/
 ↪(I_perp**2*r**2)+(I_para**2)*delta_I_perp/(I_perp**4*r**2)+I_para**2*dr**2/
 ↪(I_perp**2*r**4))
    else:
        return (I_para*r)/I_perp, np.sqrt(delta_I_para/
 ↪I_perp**2*r**2+(I_para**2*r**2)*delta_I_perp/I_perp**4+I_para**2*dr**2/
 ↪I_perp**2)
print(pol(1,1))
```

[ ]:

```
I_CCl4_90 = [(i[1], np.sqrt(di[0][0])) for i, di in zip(CCl4_p, CCl4_Vp)]
I_CCl4_90 = I_CCl4_90[0:1]+I_CCl4_90[2:8]
print(len(pol90zu00))
pos_CCl4_90 = [int(np.mean(i))+2000 for i in CCl4_window]

I_CCl4_00 = [(i[0], np.sqrt(di[0][0])) for i, di in zip(CCl4_00_p, CCl4_00_Vp)]
print(CCl4_Vp[0][1][1])
dep_CCl4 = [dep(i[0], j[0], i[1], j[1], pos[1], np.sqrt(dpos[1][1])) for i, j,␣
 ↪pos, dpos in zip(I_CCl4_00, I_CCl4_90, CCl4_00_p, CCl4_00_Vp)]
dep_CCl4
```

0.2.4 CHCl3

[ ]:

```
CHCl3_lambda, delta_CHCl3_lambda, CHCl3_spectrum, delta_CHCl3_spectrum =␣
  ↪background('measure_CHCl3.dig', 'measure_background.dig')
CHCl3_lambda, delta_CHCl3_lambda = calibration(CHCl3_lambda, delta_CHCl3_lambda)
```

[ ]:

```
CHCl3_laser_p, CHCl3_laser_Vp = curve_fit(voigt, CHCl3_lambda, CHCl3_spectrum,␣
  ↪sigma = delta_CHCl3_spectrum, absolute_sigma=True, p0=[1000, 532, 0.1,0.1,0])
x_laser = np.linspace(500,560,5000)

CHCl3_p = []
CHCl3_Vp = []
CHCl3_window = [[950,1080], [1590, 1670], [1720, 1780]]
CHCl3_p0 = [[50, 526, 0.1,0.1,90],[50, 528, 0.1,0.1,10], [150, 541,0.1,0.1,30]]

plt.subplots(figsize=(16,9))
#plt.semilogy()
plt.errorbar(CHCl3_lambda, CHCl3_spectrum, xerr=delta_CHCl3_lambda,␣
  ↪yerr=delta_CHCl3_spectrum, ecolor="black", marker="x", ls="", color="green",␣
  ↪markersize=5, capsize=3, label='measured spectrum')
plt.plot(x_laser,␣
  ↪voigt(x_laser,CHCl3_laser_p[0],CHCl3_laser_p[1],CHCl3_laser_p[2],CHCl3_laser_p[3],CHCl3_las
  ↪color='red', label='fit of Rayleigh peak')

for i, w in enumerate(CHCl3_window):
    plt.fill_betweenx([0,500], CHCl3_lambda[w[0]], CHCl3_lambda[w[1]],␣
  ↪color='lightgrey')
    lp, lV = curve_fit(voigt, CHCl3_lambda[w[0]:w[1]], CHCl3_spectrum[w[0]:
  ↪w[1]], sigma = delta_CHCl3_spectrum[w[0]:w[1]], absolute_sigma=True,␣
  ↪p0=CHCl3_p0[i], bounds = (0, np.inf))
    plt.plot(np.linspace(CHCl3_lambda[w[0]], CHCl3_lambda[w[1]],5000), voigt(np.
  ↪linspace(CHCl3_lambda[w[0]], CHCl3_lambda[w[1]],5000), lp[0], lp[1], lp[2],␣
  ↪lp[3], lp[4]), color='red')
    CHCl3_p.append(lp)
    CHCl3_Vp.append(lV)
#plt.xlim(CHCl3_lambda[CHCl3_window[-1][0]]-5,␣
  ↪CHCl3_lambda[CHCl3_window[-1][1]]+5)
plt.legend()
plt.grid()
plt.ylim(0,500)
#plt.xlim(500,560)
plt.xlabel("wavelength $\\lambda$ [nm]")
plt.ylabel("Counts")

plt.savefig('Bilder/CHCl3_mono.pdf', bbox_inches='tight')
```

```
[ ]: CHCl3_energy = [energy(i[1], np.sqrt(di[1][1])) for i, di in zip(CHCl3_p,␣
       ↪CHCl3_Vp)]
     CHCl3_laser_wavelength = (CHCl3_laser_p[1], np.sqrt(CHCl3_laser_Vp[1][1]))
     CHCl3_laser_energy = energy(CHCl3_laser_p[1], np.sqrt(CHCl3_laser_Vp[1][1]))
     CHCl3_nu = [(i[0]-CHCl3_laser_energy[0], np.
       ↪sqrt(i[1]**2+CHCl3_laser_energy[1]**2)) for i in CHCl3_energy]
     print(f'Laserwavelength = {CHCl3_laser_wavelength} nm')
     print('Delta nu in cm^-1',CHCl3_nu)
     #print('t', (CHCl3_nu[0][0]-lit_CHCl3[1])/CHCl3_nu[0][1])
     #print('t', (CHCl3_nu[1][0]-lit_CHCl3[1])/CHCl3_nu[1][1])
     print('t laser', (CHCl3_laser_wavelength[0]-532)/CHCl3_laser_wavelength[1])
     print(lit_CHCl3)
```

0.2.5 CH2Cl2

```
[ ]: CH2Cl2_lambda, delta_CH2Cl2_lambda, CH2Cl2_spectrum, delta_CH2Cl2_spectrum =␣
       ↪background('measure_CH2Cl2.dig', 'measure_background.dig')
     CH2Cl2_lambda, delta_CH2Cl2_lambda = calibration(CH2Cl2_lambda,␣
       ↪delta_CH2Cl2_lambda)
```

```
[ ]: CH2Cl2_laser_p, CH2Cl2_laser_Vp = curve_fit(voigt, CH2Cl2_lambda,␣
       ↪CH2Cl2_spectrum, sigma = delta_CH2Cl2_spectrum, absolute_sigma=True,␣
       ↪p0=[1000, 532, 0.1,0.1,0])
     x_laser = np.linspace(500,560,5000)

     CH2Cl2_p = []
     CH2Cl2_Vp = []
     CH2Cl2_window = [[1600, 1670], [1980, 2070], [2100, 2300]]
     CH2Cl2_p0 = [[50, 541, 0.1,0.1,125],[30, 551, 0.1,0.1,30], [150, 541,0.1,0.
       ↪1,30]]
     plt.subplots(figsize=(16,9))
     #plt.semilogy()
     plt.errorbar(CH2Cl2_lambda, CH2Cl2_spectrum, xerr=delta_CH2Cl2_lambda,␣
       ↪yerr=delta_CH2Cl2_spectrum, ecolor="black", marker="x", ls="",␣
       ↪color="green", markersize=5, capsize=3, label='measured spectrum') #
     plt.plot(x_laser,␣
       ↪voigt(x_laser,CH2Cl2_laser_p[0],CH2Cl2_laser_p[1],CH2Cl2_laser_p[2],CH2Cl2_laser_p[3],CH2Cl
       ↪color='red', label='fit of Rayleigh peak')

     for i, w in enumerate(CH2Cl2_window):
         plt.fill_betweenx([0,500], CH2Cl2_lambda[w[0]], CH2Cl2_lambda[w[1]],␣
       ↪color='lightgrey')
         lp, lV = curve_fit(voigt, CH2Cl2_lambda[w[0]:w[1]], CH2Cl2_spectrum[w[0]:
       ↪w[1]], sigma = delta_CH2Cl2_spectrum[w[0]:w[1]], absolute_sigma=True,␣
       ↪p0=CH2Cl2_p0[i], bounds = (0, np.inf))
```

```
    plt.plot(np.linspace(CH2Cl2_lambda[w[0]], CH2Cl2_lambda[w[1]],5000),␣
  ↪voigt(np.linspace(CH2Cl2_lambda[w[0]], CH2Cl2_lambda[w[1]],5000), lp[0],␣
  ↪lp[1], lp[2], lp[3], lp[4]), color='red')
    CH2Cl2_p.append(lp)
    CH2Cl2_Vp.append(lV)
#plt.xlim(CH2Cl2_lambda[CH2Cl2_window[-1][0]]-5,␣
  ↪CH2Cl2_lambda[CH2Cl2_window[-1][1]]+5)
plt.legend()
plt.grid()
plt.ylim(0,200)
#plt.xlim(500,560)
plt.xlabel("wavelength $\\lambda$ [nm]")
plt.ylabel("Counts")

plt.savefig('Bilder/CH2Cl2_mono.pdf', bbox_inches='tight')
```

[ ]:
```
CH2Cl2_energy = [energy(i[1], np.sqrt(di[1][1])) for i, di in zip(CH2Cl2_p,␣
  ↪CH2Cl2_Vp)]
CH2Cl2_laser_wavelength = (CH2Cl2_laser_p[1], np.sqrt(CH2Cl2_laser_Vp[1][1]))
CH2Cl2_laser_energy = energy(CH2Cl2_laser_p[1], np.sqrt(CH2Cl2_laser_Vp[1][1]))
CH2Cl2_nu = [(i[0]-CH2Cl2_laser_energy[0], np.
  ↪sqrt(i[1]**2+CH2Cl2_laser_energy[1]**2)) for i in CH2Cl2_energy]
print(f'Laserwavelength = {CH2Cl2_laser_wavelength} nm')
print('Delta nu in cm^-1', CH2Cl2_nu)
#print('t', (CH2Cl2_nu[0][0]-lit_CH2Cl2[1])/CH2Cl2_nu[0][1])
#print('t', (CH2Cl2_nu[1][0]-lit_CH2Cl2[1])/CH2Cl2_nu[1][1])
print('t laser', (CH2Cl2_laser_wavelength[0]-532)/CH2Cl2_laser_wavelength[1])
print(lit_CH2Cl2)
```

0.2.6 CS2

[ ]:
```
CS2_lambda, delta_CS2_lambda, CS2_spectrum, delta_CS2_spectrum =␣
  ↪background('measure_CS2.dig', 'measure_background.dig')
CS2_lambda, delta_CS2_lambda = calibration(CS2_lambda, delta_CS2_lambda)
```

[ ]:
```
CS2_laser_p, CS2_laser_Vp = curve_fit(voigt, CS2_lambda, CS2_spectrum, sigma =␣
  ↪delta_CS2_spectrum, absolute_sigma=True, p0=[1000, 532, 0.1,0.1,0],␣
  ↪bounds=(0,np.inf))
x_laser = np.linspace(500,560,5000)

CS2_p = []
CS2_Vp = []
CS2_window = [[450, 650], [2100, 2270]]
CS2_p0 = [[25, 514, 0.1,0.1,25],[200, 555, 0.1,0.1,30]]
plt.subplots(figsize=(16,9))
#plt.semilogy()
```

```
plt.errorbar(CS2_lambda, CS2_spectrum, xerr=delta_CS2_lambda,␣
 ↪yerr=delta_CS2_spectrum, ecolor="black", marker="x", ls="", color="green",␣
 ↪markersize=5, capsize=3, label='measured spectrum') #
plt.plot(x_laser,␣
 ↪voigt(x_laser,CS2_laser_p[0],CS2_laser_p[1],CS2_laser_p[2],CS2_laser_p[3],CS2_laser_p[4]),␣
 ↪color='red', label='fit of Rayleigh peak')

for i, w in enumerate(CS2_window):
    plt.fill_betweenx([0,500], CS2_lambda[w[0]], CS2_lambda[w[1]],␣
 ↪color='lightgrey')
    lp, lV = curve_fit(voigt, CS2_lambda[w[0]:w[1]], CS2_spectrum[w[0]:w[1]],␣
 ↪sigma = delta_CS2_spectrum[w[0]:w[1]], absolute_sigma=True, p0=CS2_p0[i],␣
 ↪bounds = (0, np.inf))
    plt.plot(np.linspace(CS2_lambda[w[0]], CS2_lambda[w[1]],5000), voigt(np.
 ↪linspace(CS2_lambda[w[0]], CS2_lambda[w[1]],5000), lp[0], lp[1], lp[2],␣
 ↪lp[3], lp[4]), color='red')
    CS2_p.append(lp)
    CS2_Vp.append(lV)
#plt.xlim(CS2_lambda[CS2_window[-1][0]]-5, CS2_lambda[CS2_window[-1][1]]+5)
plt.legend()
plt.grid()
plt.ylim(0,200)
#plt.xlim(500,560)
plt.xlabel("wavelength $\\lambda$ [nm]")
plt.ylabel("Counts")

plt.savefig('Bilder/CS2_mono.pdf', bbox_inches='tight')
```

[ ]:

```
CS2_energy = [energy(i[1], np.sqrt(di[1][1])) for i, di in zip(CS2_p, CS2_Vp)]
CS2_laser_wavelength = (CS2_laser_p[1], np.sqrt(CS2_laser_Vp[1][1]))
CS2_laser_energy = energy(CS2_laser_p[1], np.sqrt(CS2_laser_Vp[1][1]))
CS2_nu = [(i[0]-CS2_laser_energy[0], np.sqrt(i[1]**2+CS2_laser_energy[1]**2))␣
 ↪for i in CS2_energy]
print(f'Laserwavelength = {CS2_laser_wavelength} nm')
print('Delta nu in cm^-1', CS2_nu)
print('t', (CS2_nu[0][0]-lit_CS2[1])/CS2_nu[0][1])
print('t', (CS2_nu[1][0]-lit_CS2[1])/CS2_nu[1][1])
print('t laser', (CS2_laser_wavelength[0]-532)/CS2_laser_wavelength[1])
print(lit_CS2)
```

0.2.7 Sulfur

[ ]:

```
sulfur_lambda, delta_sulfur_lambda, sulfur_spectrum, delta_sulfur_spectrum =␣
 ↪background('measure_sulfur.dig', 'measure_background.dig')
sulfur_lambda, delta_sulfur_lambda = calibration(sulfur_lambda,␣
 ↪delta_sulfur_lambda)
```

```
[ ]: sulfur_laser_p, sulfur_laser_Vp = curve_fit(voigt, sulfur_lambda,␣
      ↪sulfur_spectrum, sigma = delta_sulfur_spectrum, absolute_sigma=True,␣
      ↪p0=[300, 535, 0.1,0.1,0])
     x_laser = np.linspace(500,560,5000)

     sulfur_p = []
     sulfur_Vp = []
     sulfur_window = [[720, 780], [785, 820], [1900,1940], [1945,2000]]#, [2100,␣
      ↪2270]]
     sulfur_p0 = [[150, 519, 0.1,0.1,0],[20, 521, 0.1,0.1,0],[100, 549, 0.1,0.
      ↪1,200],[600, 549, 0.1,0.1,200]]
     plt.subplots(figsize=(16,9))
     #plt.semilogy()
     plt.errorbar(sulfur_lambda, sulfur_spectrum, xerr=delta_sulfur_lambda,␣
      ↪yerr=delta_sulfur_spectrum, ecolor="black", marker="x", ls="",␣
      ↪color="green", markersize=5, capsize=3, label='measured spectrum')
     plt.plot(x_laser,␣
      ↪voigt(x_laser,sulfur_laser_p[0],sulfur_laser_p[1],sulfur_laser_p[2],sulfur_laser_p[3],sulfu
      ↪color='red', label='fit of Rayleigh peak')

     for i, w in enumerate(sulfur_window):
         plt.fill_betweenx([0,1000], sulfur_lambda[w[0]], sulfur_lambda[w[1]],␣
      ↪color='lightgrey')
         lp, lV = curve_fit(voigt, sulfur_lambda[w[0]:w[1]], sulfur_spectrum[w[0]:
      ↪w[1]], sigma = delta_sulfur_spectrum[w[0]:w[1]], absolute_sigma=True,␣
      ↪p0=sulfur_p0[i], bounds = (0, np.inf))
         plt.plot(np.linspace(sulfur_lambda[w[0]], sulfur_lambda[w[1]],5000),␣
      ↪voigt(np.linspace(sulfur_lambda[w[0]], sulfur_lambda[w[1]],5000), lp[0],␣
      ↪lp[1], lp[2], lp[3], lp[4]), color='red')
         sulfur_p.append(lp)
         sulfur_Vp.append(lV)
     #plt.xlim(sulfur_lambda[sulfur_window[-1][0]]-5,␣
      ↪sulfur_lambda[sulfur_window[-1][1]]+5)
     plt.legend()
     plt.grid()
     plt.ylim(-30,800)
     #plt.xlim(500,560)
     plt.xlabel("wavelength $\\lambda$ [nm]")
     plt.ylabel("Counts")
     plt.savefig('Bilder/sulfur_mono.pdf', bbox_inches='tight')
```

```
[ ]: sulfur_energy = [energy(i[1], np.sqrt(di[1][1])) for i, di in zip(sulfur_p,␣
      ↪sulfur_Vp)]
     sulfur_laser_wavelength = (sulfur_laser_p[1], np.sqrt(sulfur_laser_Vp[1][1]))
     sulfur_laser_energy = energy(sulfur_laser_p[1], np.sqrt(sulfur_laser_Vp[1][1]))
```

```
sulfur_nu = [(i[0]-sulfur_laser_energy[0], np.
 ↪sqrt(i[1]**2+sulfur_laser_energy[1]**2)) for i in sulfur_energy]
print(f'Laserwavelength = {sulfur_laser_wavelength} nm')
print('Delta nu in cm^-1')
sulfur_nu
```

[ ]:

```
sulfur_I = [(i[0], np.sqrt(di[0][0])) for i, di in zip(sulfur_p, sulfur_Vp)]
print(sulfur_I)
```

[ ]:

```
h=2*np.pi
def T(nu_s, nu_a, I_s, I_a, delta_nu_s, delta_nu_a, delta_I_s, delta_I_a):
    convert = 1.973*10**(-7)/(10**(-2))
    delta_nu = (nu_a-nu_s)/2*convert #energy difference in eV
    t = h*delta_nu/(np.log(I_s/I_a)+4*np.log(nu_a/nu_s)) #energy difference in␣
 ↪eV
    denom = (np.log(I_s/I_a)+4*np.log(nu_a/nu_s)) # denominator
    delta_t = np.sqrt((t/(I_s*denom)*delta_I_s)**2+(t/
 ↪(I_a*denom)*delta_I_a)**2+(-4*t/(nu_a*denom)+h/
 ↪(2*denom))**2*(convert*delta_nu_a)**2+(4*t/(nu_s*denom)-h/
 ↪(2*denom))**2*(convert*delta_nu_s)**2)
    return t*1.16e4, delta_t*1.16e4 # temperatur in K
```

[ ]:

```
T(np.abs(sulfur_energy[3][0]), sulfur_energy[0][0], sulfur_I[3][0],␣
 ↪sulfur_I[0][0],sulfur_energy[3][1], sulfur_energy[0][1], sulfur_I[3][1],␣
 ↪sulfur_I[0][1])
```

[ ]:

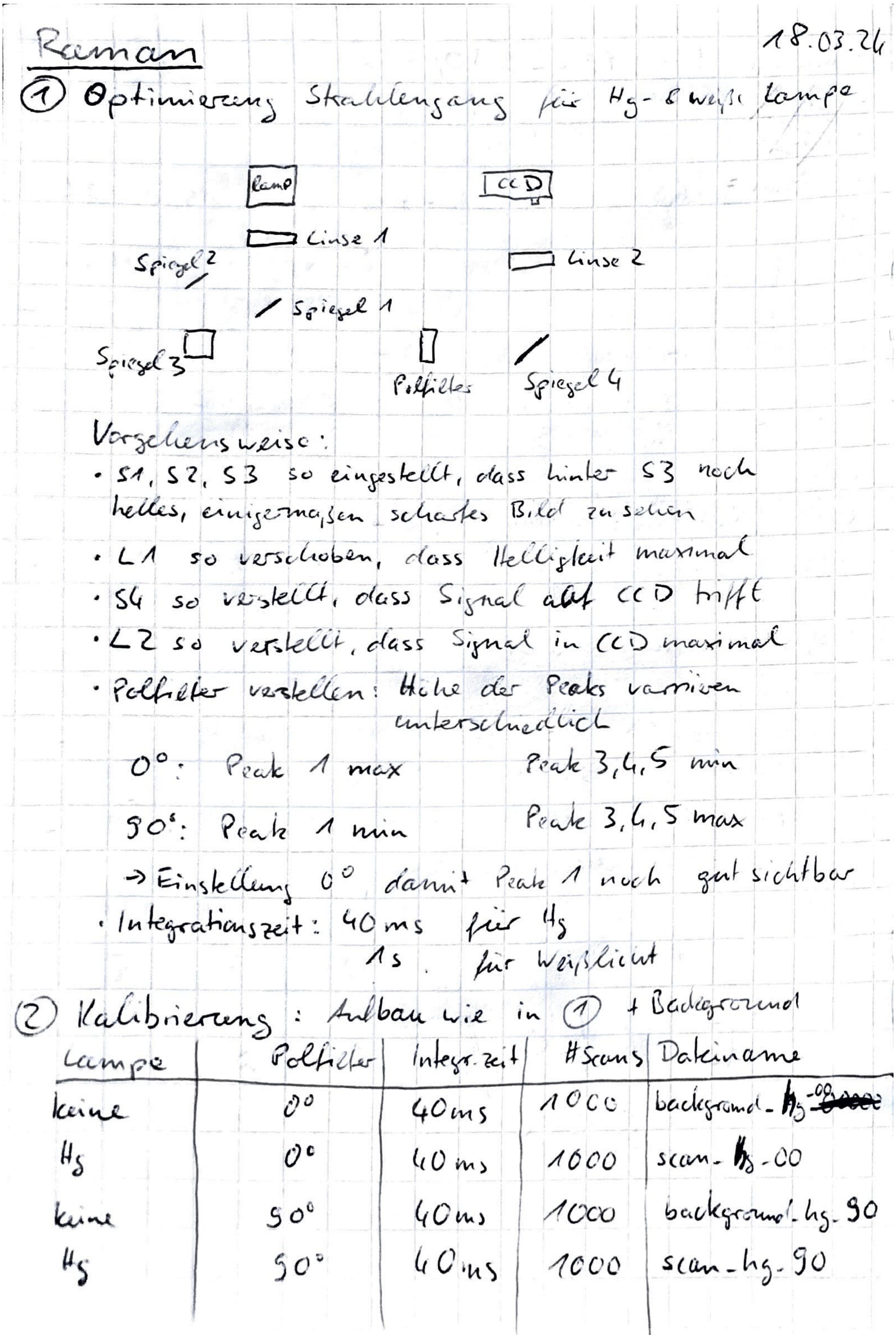
```
T(np.abs(sulfur_energy[2][0]), sulfur_energy[1][0], sulfur_I[2][0],␣
 ↪sulfur_I[1][0],sulfur_energy[2][1], sulfur_energy[1][1], sulfur_I[2][1],␣
 ↪sulfur_I[1][1])
```

[ ]:

```

```

7.3 Lab notes



Raman 18.03.24

① Optimierung Strahlengang für Hg- & weiß lampe

Vorgehensweise:

- S1, S2, S3 so eingestellt, dass hinter S3 noch helles, einigermaßen scharfes Bild zu sehen
- L1 so verschoben, dass Helligkeit maximal
- S4 so verstellt, dass Signal auf CCD trifft
- L2 so verstellt, dass Signal in CCD maximal
- Polfilter verstellen: Höhe der Peaks variieren unterschiedlich

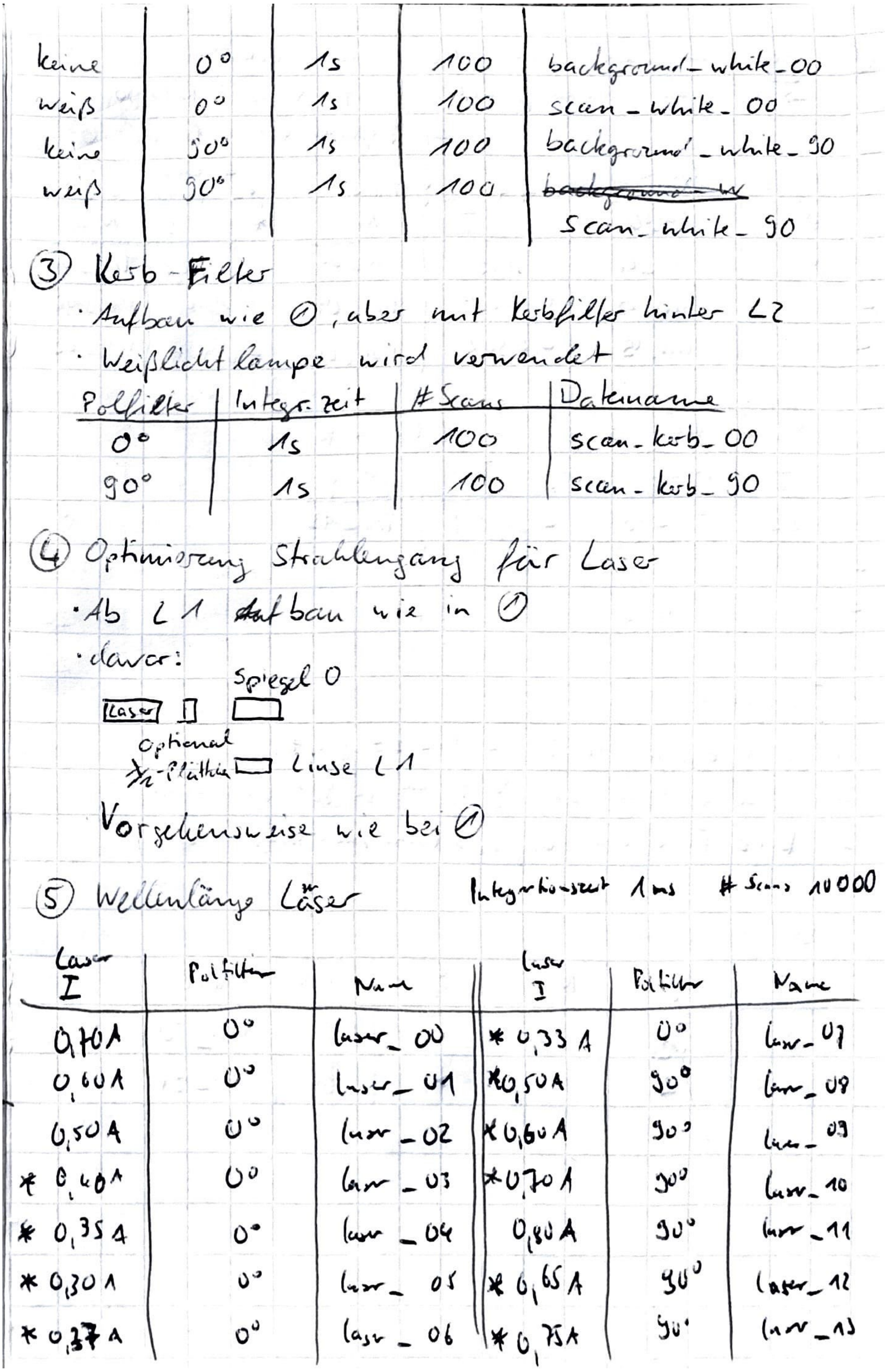
  0°: Peak 1 max Peak 3,4,5 min

  90°: Peak 1 min Peak 3,4,5 max

  → Einstellung 0° damit Peak 1 noch gut sichtbar
- Integrationszeit: 40 ms für Hg
  1 s für Weißlicht

② Kalibrierung: Aufbau wie in ① + Background

Lampe	Polfilter	Integr. zeit	#Scans	Dateiname
keine	0°	40 ms	1000	background-Hg-00
Hg	0°	40 ms	1000	scan-Hg-00
keine	90°	40 ms	1000	background-hg-90
Hg	90°	40 ms	1000	scan-hg-90



keine	0°	1s	100	background_white_00
weiß	0°	1s	100	scan_white_00
keine	90°	1s	100	background_white_90
weiß	90°	1s	100	~~background_w~~ scan_white_90

③ Kerb-Filter

- Aufbau wie ①, aber mit Kerbfilter hinter L2
- Weißlichtlampe wird verwendet

Polfilter	Integr. zeit	# Scans	Dateiname
0°	1s	100	scan_kerb_00
90°	1s	100	scan_kerb_90

④ Optimierung Strahlengang für Laser

- Ab L1 ~~Auf~~bau wie in ①
- davor:

Spiegel 0

Laser

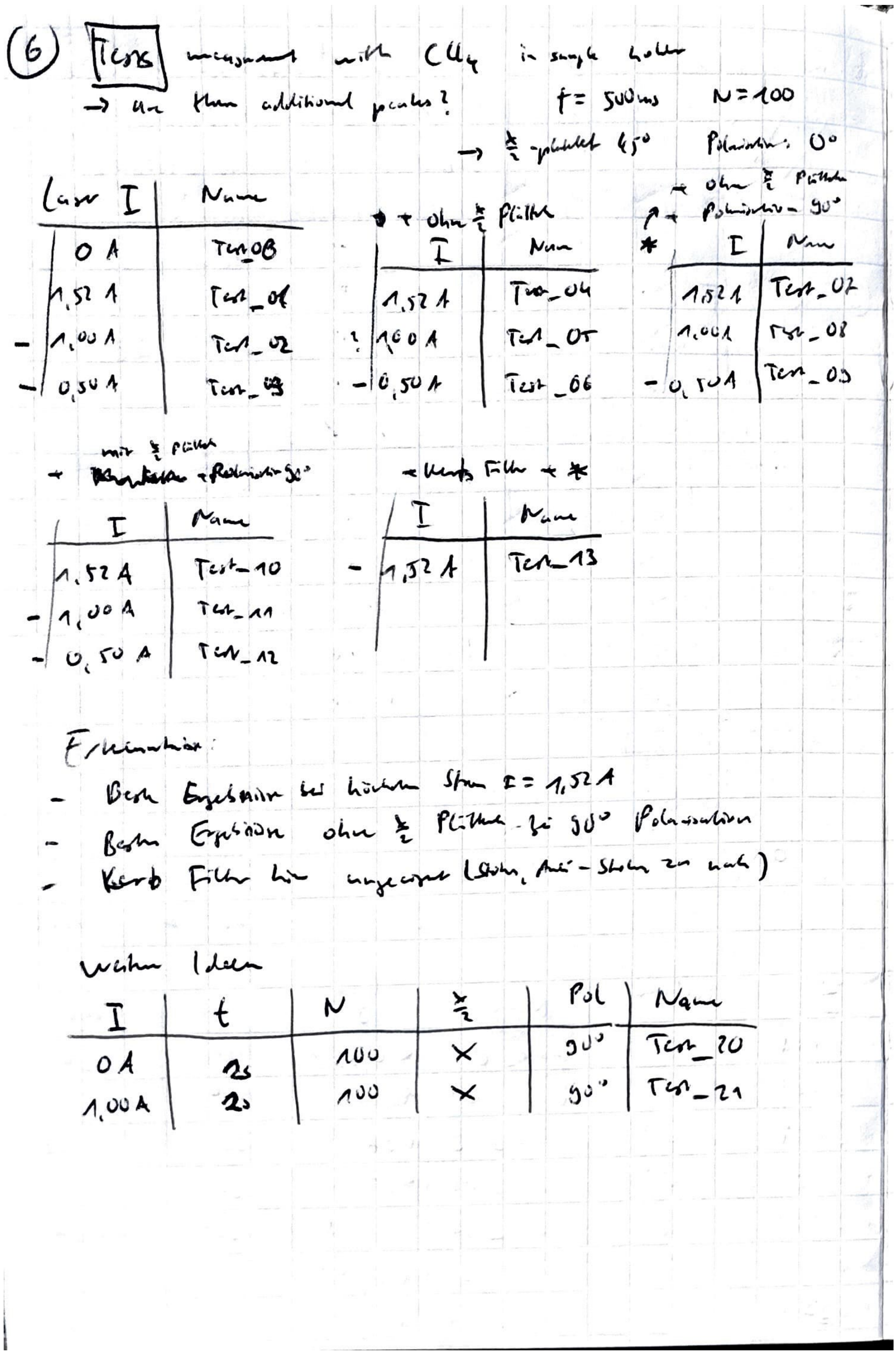
optional λ/2-Plättchen

Linse L1

Vorgehensweise wie bei ①

⑤ Wellenlänge Laser   Integrationszeit 1ms   # Scans 10000

Laser I	Polfilter	Name	Laser I	Polfilter	Name
0,70 A	0°	laser_00	* 0,33 A	0°	laser_07
0,60 A	0°	laser_01	* 0,50 A	90°	laser_08
0,50 A	0°	laser_02	* 0,60 A	90°	laser_09
* 0,40 A	0°	laser_03	* 0,70 A	90°	laser_10
* 0,35 A	0°	laser_04	0,80 A	90°	laser_11
* 0,30 A	0°	laser_05	* 0,65 A	90°	laser_12
* 0,37 A	0°	laser_06	* 0,75 A	90°	laser_13



⑥ Tests measurement with CH_4 in sample holder

→ are there additional peaks? f = 500 ms N = 100

→ λ/2-plättchen 45° Polarisation: 0°

	Laser I	Name
	0 A	Test_00
	1,52 A	Test_01
-	1,00 A	Test_02
-	0,50 A	Test_03

◆ + ohne λ/2 Plättchen

	I	Name
	1,52 A	Test_04
?	1,00 A	Test_05
-	0,50 A	Test_06

+ ohne λ/2 Plättchen
+ Polarisation 90°

*

	I	Name
	1,52 A	Test_07
	1,00 A	Test_08
-	0,50 A	Test_09

mit λ/2 Plättchen
+ ~~[illegible]~~ + Polarisation 90°

	I	Name
	1,52 A	Test_10
-	1,00 A	Test_11
-	0,50 A	Test_12

+ Kerb Filter + *

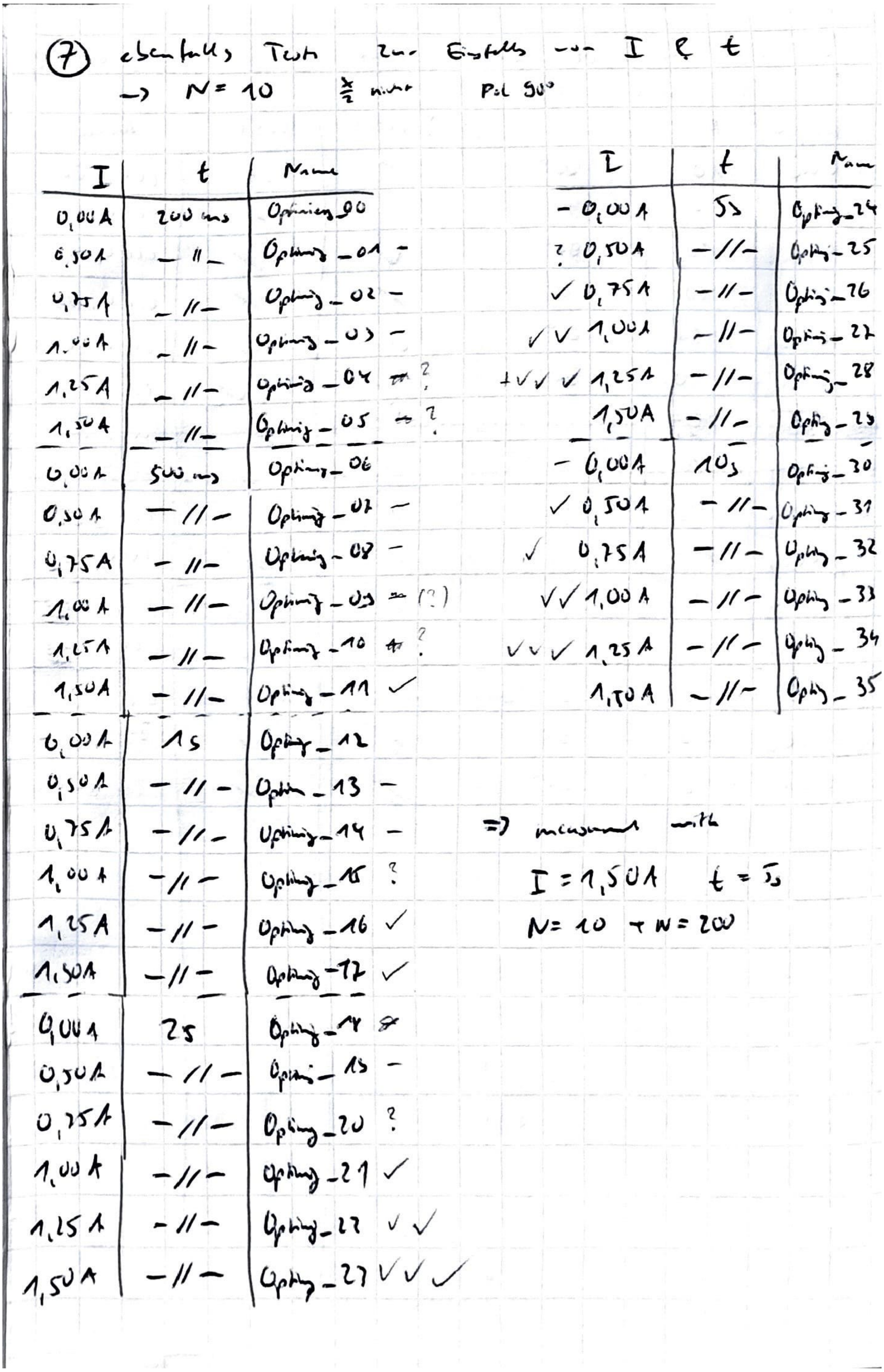
	I	Name
-	1,52 A	Test_13

Erkenntnisse:

- Beste Ergebnisse bei höchstem Strom I = 1,52 A
- Besten Ergebnisse ohne λ/2 Plättchen bei 90° Polarisation
- Kerb Filter hier ungeeignet (Stokes, Anti-Stokes zu nah)

Weitere Ideen

I	t	N	λ/2	Pol	Name
0 A	2s	100	×	90°	Test_20
1,00 A	2s	100	×	90°	Test_21

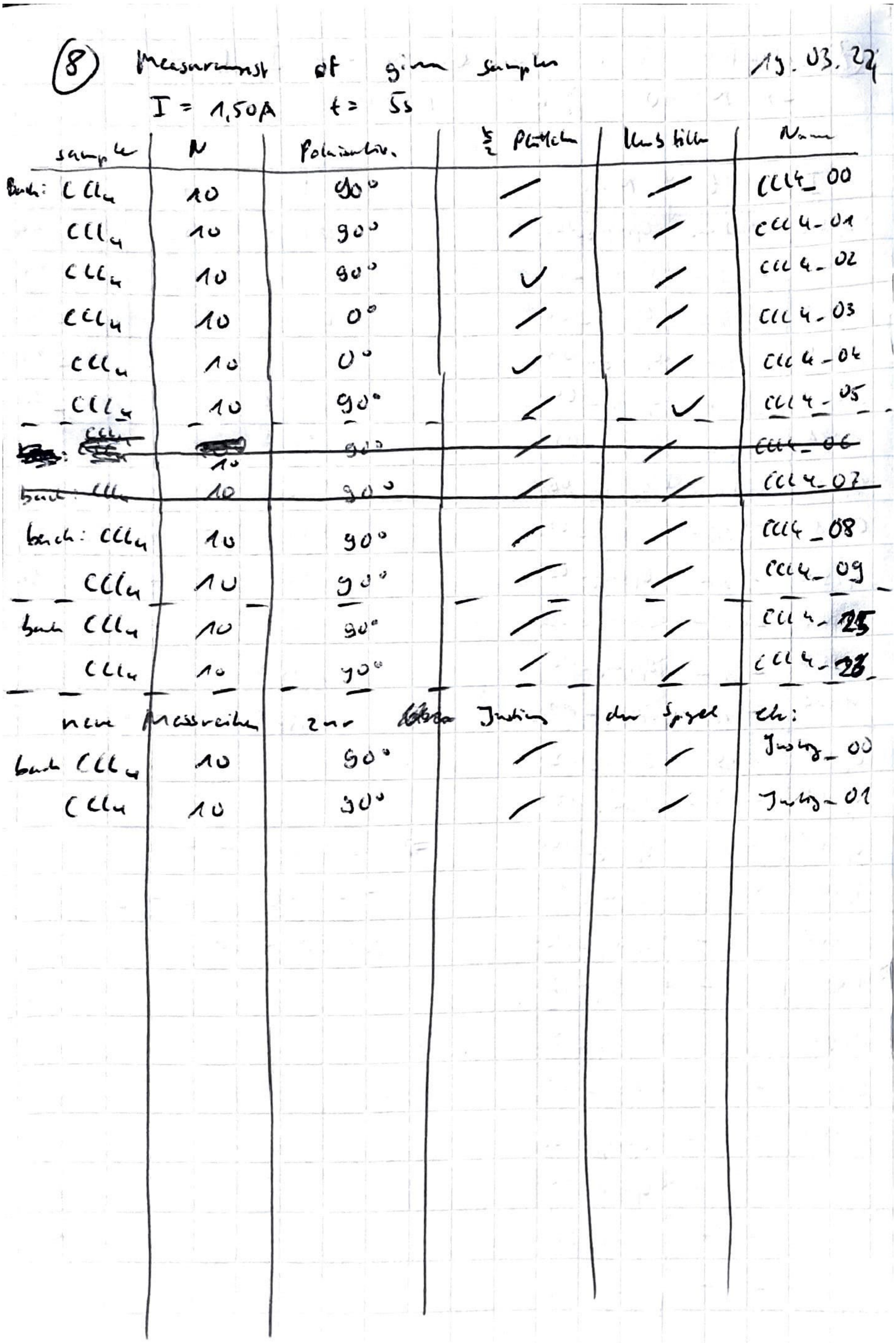


⑦ ebenfalls Tests zur Einstellung von I & t
→ N = 10 $\frac{\lambda}{2}$ mirror Pol 90°

I	t	Name	
0,00 A	200 ms	Optimierung_00	
0,50 A	–″–	Optimierung_01	–
0,75 A	–″–	Optimierung_02	–
1,00 A	–″–	Optimierung_03	–
1,25 A	–″–	Optimierung_04	+ ?
1,50 A	–″–	Optimierung_05	+ ?
0,00 A	500 ms	Optimierung_06	
0,50 A	–″–	Optimierung_07	–
0,75 A	–″–	Optimierung_08	–
1,00 A	–″–	Optimierung_09	= (?)
1,25 A	–″–	Optimierung_10	+ ?
1,50 A	–″–	Optimierung_11	✓
0,00 A	1 s	Optimierung_12	
0,50 A	–″–	Optimierung_13	–
0,75 A	–″–	Optimierung_14	–
1,00 A	–″–	Optimierung_15	?
1,25 A	–″–	Optimierung_16	✓
1,50 A	–″–	Optimierung_17	✓
0,00 A	2 s	Optimierung_18	&
0,50 A	–″–	Optimierung_19	–
0,75 A	–″–	Optimierung_20	?
1,00 A	–″–	Optimierung_21	✓
1,25 A	–″–	Optimierung_22	✓✓
1,50 A	–″–	Optimierung_23	✓✓✓

	I	t	Name
–	0,00 A	5 s	Optimierung_24
?	0,50 A	–″–	Optimierung_25
✓	0,75 A	–″–	Optimierung_26
✓✓	1,00 A	–″–	Optimierung_27
+✓✓✓	1,25 A	–″–	Optimierung_28
	1,50 A	–″–	Optimierung_29
–	0,00 A	10 s	Optimierung_30
✓	0,50 A	–″–	Optimierung_31
✓	0,75 A	–″–	Optimierung_32
✓✓	1,00 A	–″–	Optimierung_33
✓✓✓	1,25 A	–″–	Optimierung_34
	1,50 A	–″–	Optimierung_35

⇒ measurement with
I = 1,50 A t = 5 s
N = 10 + N = 200



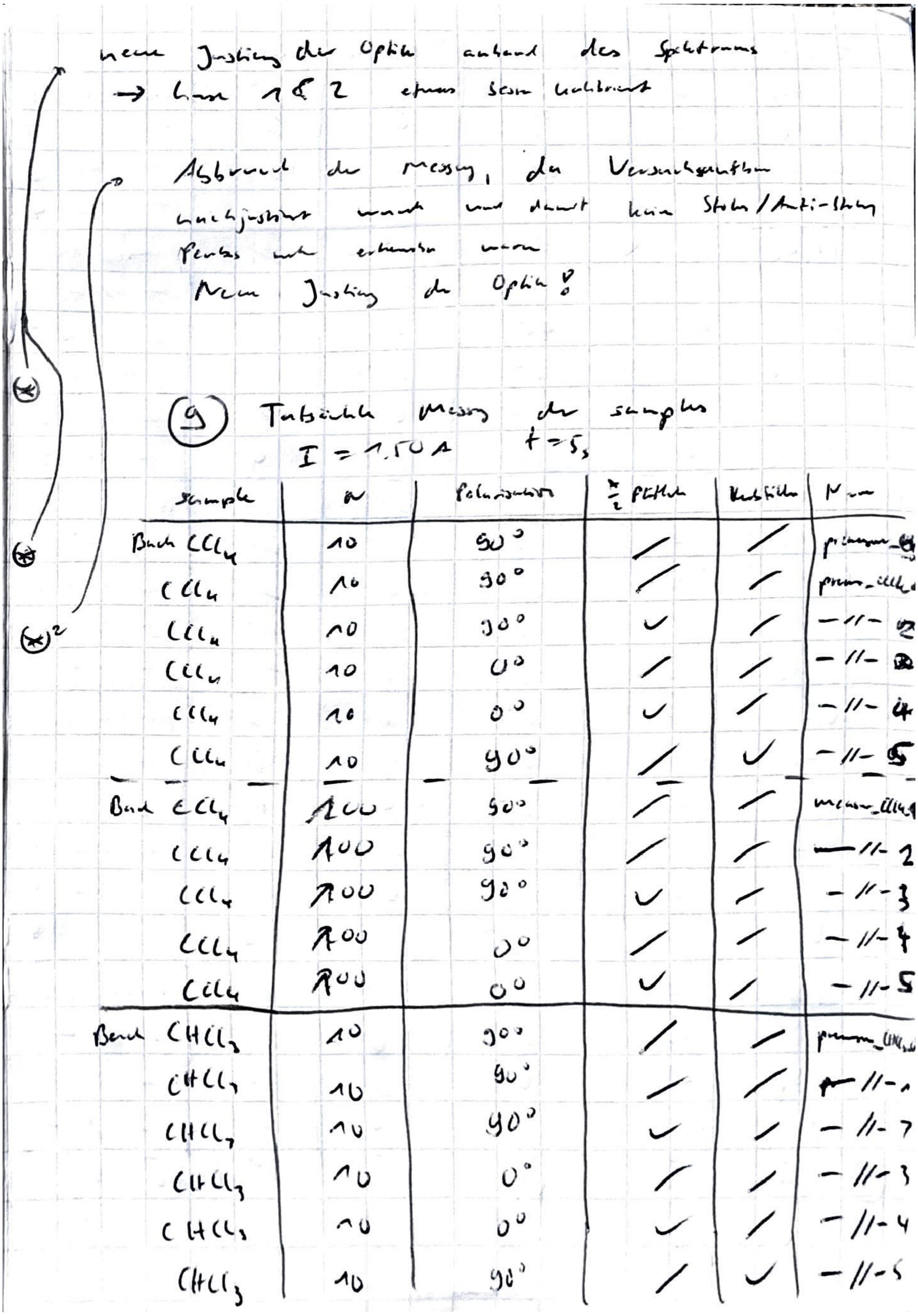
⑧ Measurement of given samples 19.03.24

I = 1,50A t = 5s

sample	N	Polarisation	λ/2 Plättchen	Kunstlicht	Name
back: CCl4	10	90°	✓	✓	CCl4_00
CCl4	10	90°	✓	✓	CCl4-01
CCl4	10	90°	✓	✓	CCl4-02
CCl4	10	0°	✓	✓	CCl4-03
CCl4	10	0°	✓	✓	CCl4-04
CCl4	10	90°	✓	✓	CCl4-05
~~CCl4~~	~~10~~	~~90°~~	~~✓~~	~~✓~~	~~CCl4_06~~
~~back: CCl4~~	~~10~~	~~90°~~	~~✓~~	~~✓~~	~~CCl4_07~~
back: CCl4	10	90°	✓	✓	CCl4_08
CCl4	10	90°	✓	✓	CCl4_09
back CCl4	10	90°	✓	✓	CCl4-25
CCl4	10	90°	✓	✓	CCl4-26

neue Messreihe zur ~~[illegible]~~ Justierung der Spiegel etc:

sample	N	Polarisation	λ/2 Plättchen	Kunstlicht	Name
back CCl4	10	90°	✓	✓	Justg_00
CCl4	10	90°	✓	✓	Justg-01



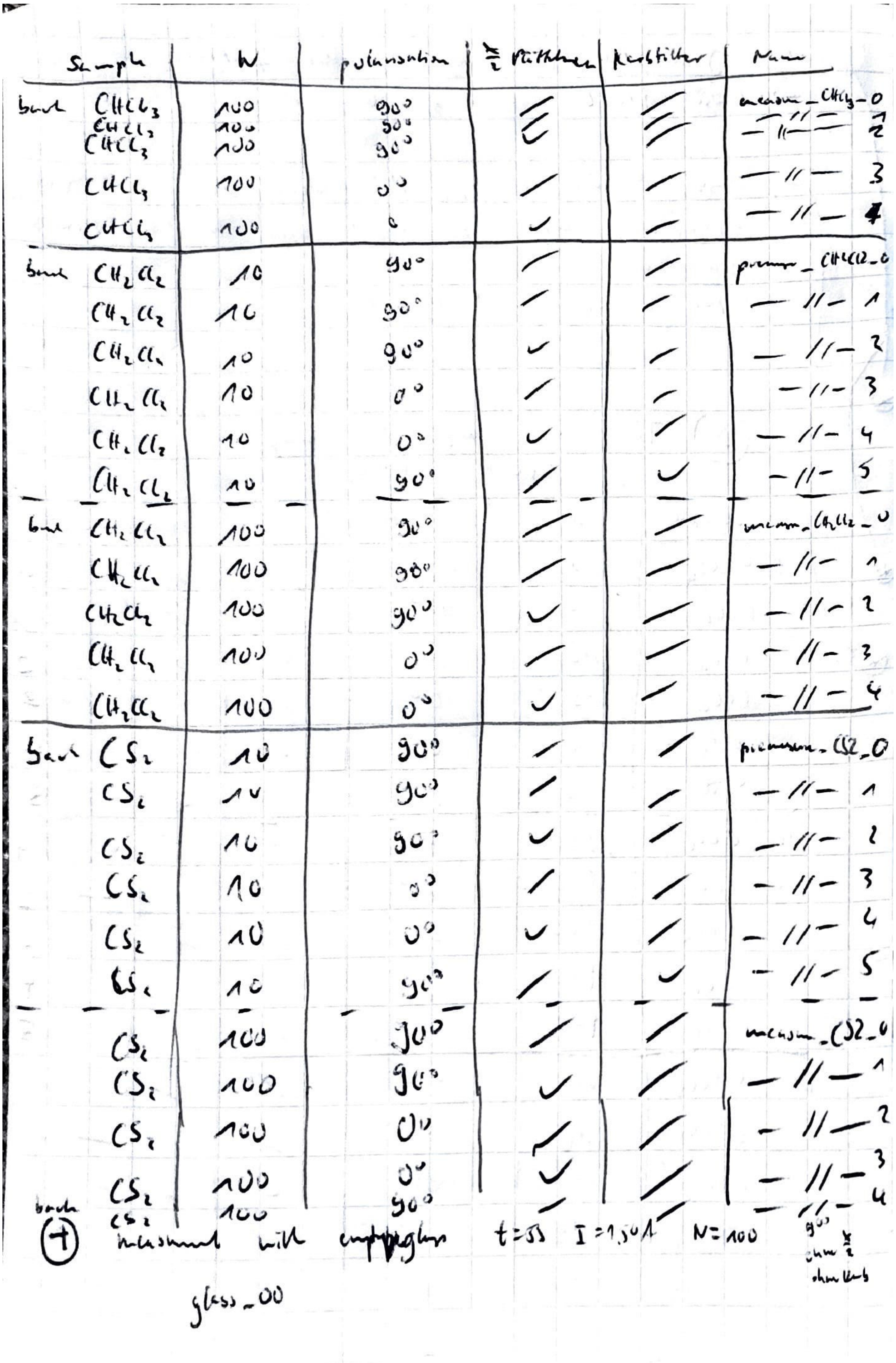
neue Justierung der Optik anhand des Spektrums
→ Linie 1 & 2 etwas [illegible] [illegible]

Abbruch der Messung, da Versuchsaufbau nachjustiert wurde und damit kein Stokes/Anti-Stokes Peaks mehr erkennbar waren
Neue Justierung der Optik!

(*)

⑨ Tatsächliche Messung der samples
$I = 1.50\,A$ $t = 5\,s$

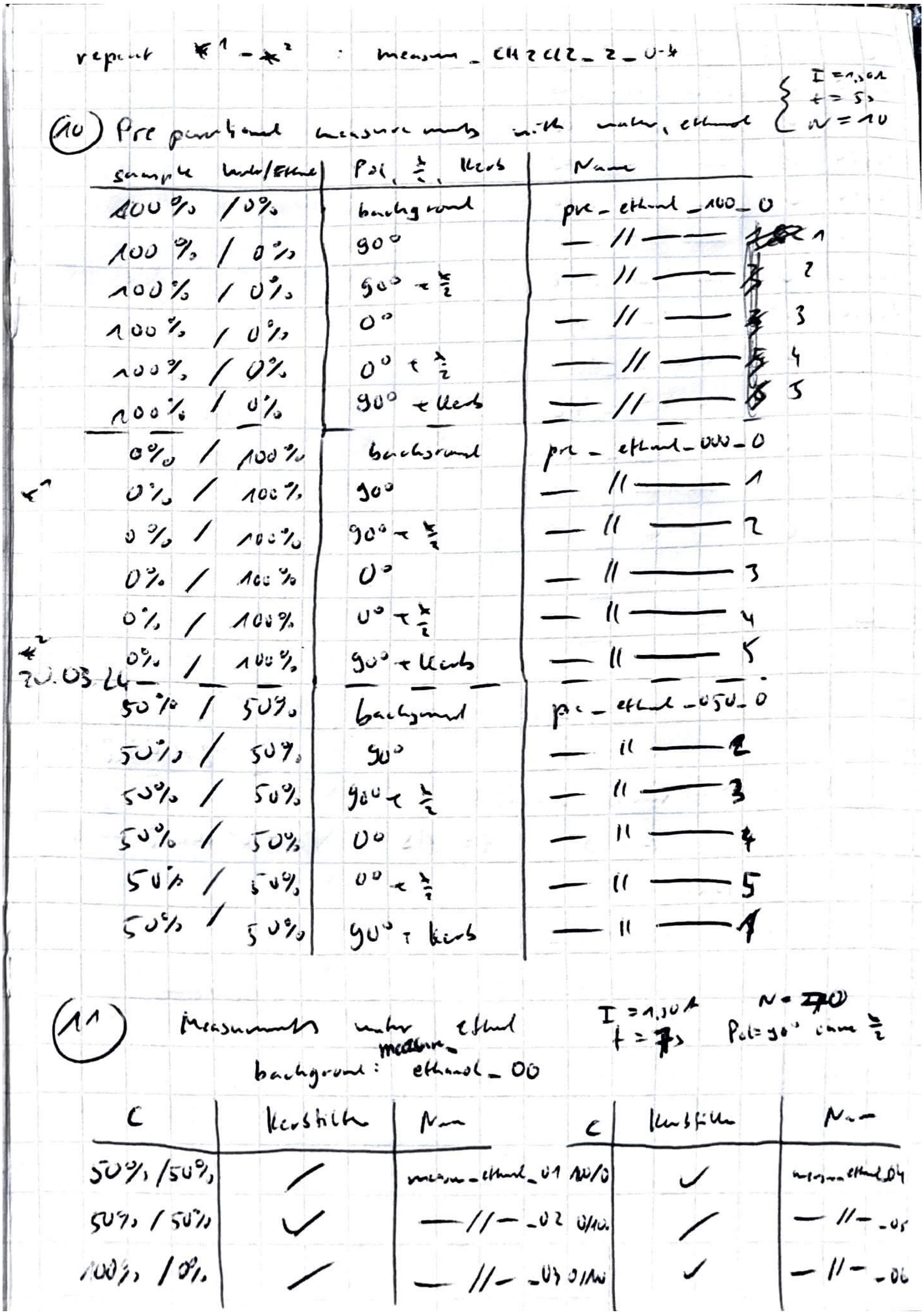
Sample	n	Polarisator	$\frac{\lambda}{2}$ Plättchen	Kantenfilter	Name
(*) Back CCl_4	10	90°	/	/	[illegible]
CCl_4	10	90°	/	/	[illegible]
$(*)^2$ CCl_4	10	90°	✓	/	-//- 2
CCl_4	10	0°	/	/	-//- 3
CCl_4	10	0°	✓	/	-//- 4
CCl_4	10	90°	/	✓	-//- 5
Back CCl_4	100	90°	/	/	[illegible]
CCl_4	100	90°	/	/	-//- 2
CCl_4	100	90°	✓	/	-//- 3
CCl_4	100	0°	/	/	-//- 4
CCl_4	100	0°	✓	/	-//- 5
Back $CHCl_3$	10	90°	/	/	[illegible]
$CHCl_3$	10	90°	/	/	-//-
$CHCl_3$	10	90°	✓	/	-//- 7
$CHCl_3$	10	0°	/	/	-//- 3
$CHCl_3$	10	0°	✓	/	-//- 4
$CHCl_3$	10	90°	/	✓	-//- 5



	Sample	W	polarisation	$\frac{\lambda}{2}$ Plättchen	Kerbfilter	Name
back	$CHCl_3$	100	90°	/	/	measure_CHCl3_0
	$CHCl_3$	100	90°	/	/	— // — 1
	$CHCl_3$	100	90°	✓	/	— // — 2
	$CHCl_3$	100	0°	/	/	— // — 3
	$CHCl_3$	100	0	✓	/	— // — 4
back	CH_2Cl_2	10	90°	/	/	premeasure_CH2Cl2_0
	CH_2Cl_2	10	90°	/	/	— // — 1
	CH_2Cl_2	10	90°	✓	/	— // — 2
	CH_2Cl_2	10	0°	/	/	— // — 3
	CH_2Cl_2	10	0°	✓	/	— // — 4
	CH_2Cl_2	10	90°	/	✓	— // — 5
back	CH_2Cl_2	100	90°	/	/	measure_CH2Cl2_0
	CH_2Cl_2	100	90°	/	/	— // — 1
	CH_2Cl_2	100	90°	✓	/	— // — 2
	CH_2Cl_2	100	0°	/	/	— // — 3
	CH_2Cl_2	100	0°	✓	/	— // — 4
back	CS_2	10	90°	/	/	premeasure_CS2_0
	CS_2	10	90°	/	/	— // — 1
	CS_2	10	90°	✓	/	— // — 2
	CS_2	10	0°	/	/	— // — 3
	CS_2	10	0°	✓	/	— // — 4
	CS_2	10	90°	/	✓	— // — 5
	CS_2	100	90°	/	/	measure_CS2_0
	CS_2	100	90°	✓	/	— // — 1
	CS_2	100	0°	✓	/	— // — 2
	CS_2	100	0°	✓	/	— // — 3
back	CS_2	100	90°	/	/	— // — 4

⊕ measurement with emptyglass t=5s I=1,50A N=100 90° ohne $\frac{\lambda}{2}$ ohne Kerb

glass_00



repeat *1 - *2 : measurem_CH2Cl2_2_0-4

I = 1,30A
t = 5s
N = 10

(10) Pre pervational measurements with water, ethanol

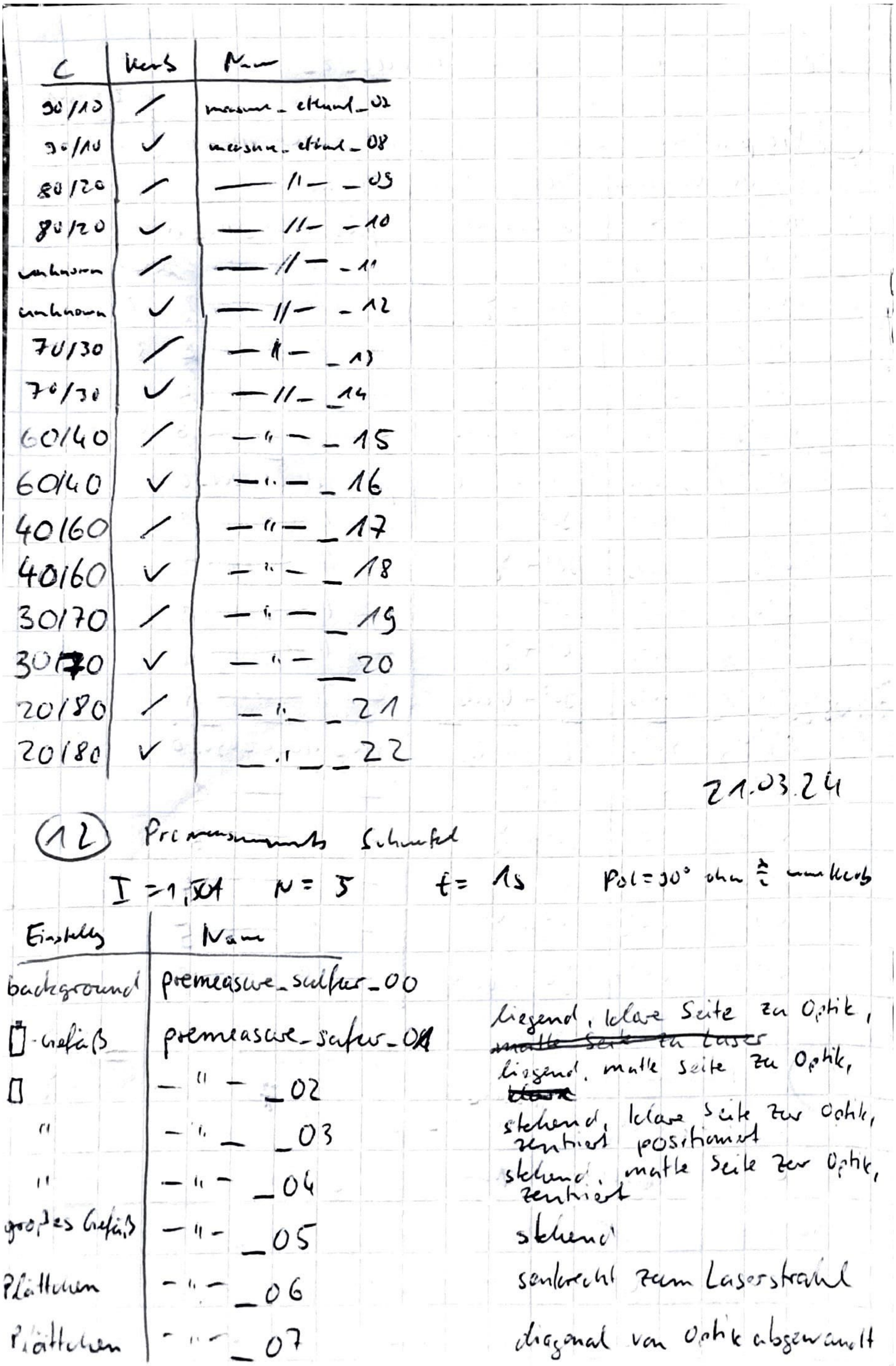
sample water/Ethanol	Pol, λ/2, Kerb	Name
100% / 0%	background	pre_ethanol_100_0
100% / 0%	90°	— // —— 1
100% / 0%	90° + λ/2	— // —— 2
100% / 0%	0°	— // —— 3
100% / 0%	0° + λ/2	— // —— 4
100% / 0%	90° + Kerb	— // —— 5
0% / 100%	background	pre_ethanol_000_0
*1 0% / 100%	90°	— // —— 1
0% / 100%	90° + λ/2	— // —— 2
0% / 100%	0°	— // —— 3
0% / 100%	0° + λ/2	— // —— 4
*2 0% / 100%	90° + Kerb	— // —— 5
20.03.24 50% / 50%	background	pre_ethanol_050_0
50% / 50%	90°	— // —— 2
50% / 50%	90° + λ/2	— // —— 3
50% / 50%	0°	— // —— 4
50% / 50%	0° + λ/2	— // —— 5
50% / 50%	90° + Kerb	— // —— 1

(11) Measurements under ethanol medium

I = 1,30A N = 70
t = 7s Pol = 90° case λ/2

background: ethanol_00

c	Kerbstift	Name	c	Kerbstift	Name
50% / 50%	✓	measure_ethanol_01	100/0	✓	measure_ethanol_04
50% / 50%	✓	— // — _02	0/100	✓	— // — _05
100% / 0%	✓	— // — _03	0/100	✓	— // — _06



C	Kerb	Name
90/10	/	measure_ethanol_07
90/10	✓	measure_ethanol_08
80/20	/	— " — _09
80/20	✓	— " — _10
unknown	/	— " — _11
unknown	✓	— " — _12
70/30	/	— " — _13
70/30	✓	— " — _14
60/40	/	— " — _15
60/40	✓	— " — _16
40/60	/	— " — _17
40/60	✓	— " — _18
30/70	/	— " — _19
30/70	✓	— " — _20
20/80	/	— " — _21
20/80	✓	— " — _22

21.03.24

⑫ Premeasurements Schwefel

I = 1,5A   N = 5   t = 1s   Pol = 90° ohne ≙ ohne Kerb

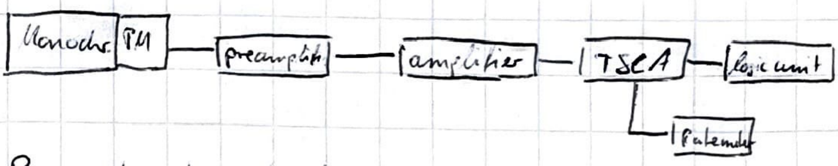
Einstellung	Name	
background	premeasure_sulfur_00	
▯-Gefäß	premeasure_sulfur_01	liegend, klare Seite zu Optik, ~~matte Seite zu Laser~~
▯	— " — _02	liegend, matte Seite zu Optik, ~~klare~~
"	— " — _03	stehend, klare Seite zur Optik, zentriert positioniert
"	— " — _04	stehend, matte Seite zur Optik, zentriert
großes Gefäß	— " — _05	stehend
Plättchen	— " — _06	senkrecht zum Laserstrahl
Plättchen	— " — _07	diagonal von Optik abgewandt

⑬ N = 70 t = 7s Pol-90° ohne $\frac{\lambda}{4}$ mit Kerb

Pt-Zelle : Laser [illegible] Schirm

I	Name
[illegible] 0A	measure_sulfur_00
1,50A	—"— _01
1,30A	—"— _02
1,40A	—"— _03
1,20A	—"— _04
1,10A	—"— _05
1,00A	—"— _06

⑭ Setup of Monochromator



Parameters to adjust:

- wavelength λ
- slit width d
- scan rate

} Monochromator

- coarse gain = x20
- fine gain = 12.5
- integrate = Out
- diff = Out

} amplifier

- E = 1.09
- ΔE = 6.75
- delay =

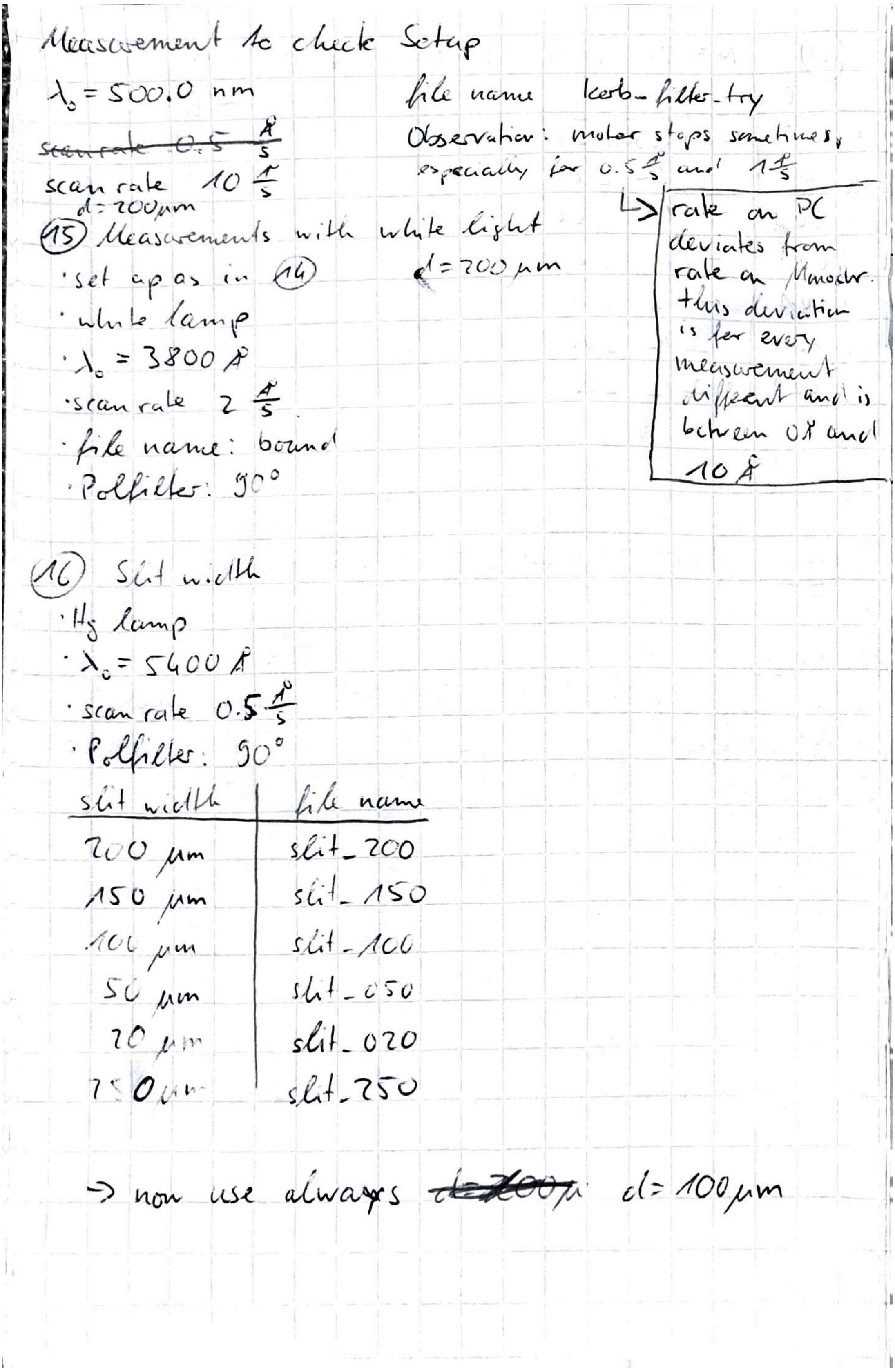
} TSCA

- ~~Faktor~~ Counts/s
- % std dev

} Ratemeter

- Scangeschwindigkeit · Anfangswellenlänge
- Zeitkonstante

Optics as in ①
but with L2 instead
of S4
→ focus beam on
slit



Measurement to check Setup

$\lambda_0 = 500.0$ nm

~~scan rate 0.5 $\frac{Å}{s}$~~

scan rate 10 $\frac{Å}{s}$

$d = 200\,\mu m$

file name kerb_filter_try

Observation: motor stops sometimes, especially for 0.5 $\frac{Å}{s}$ and 1 $\frac{Å}{s}$

→ rate on PC deviates from rate on Monochr. this deviation is for every measurement different and is between 0Å and 10 Å

⑮ Measurements with white light

$d = 200\,\mu m$

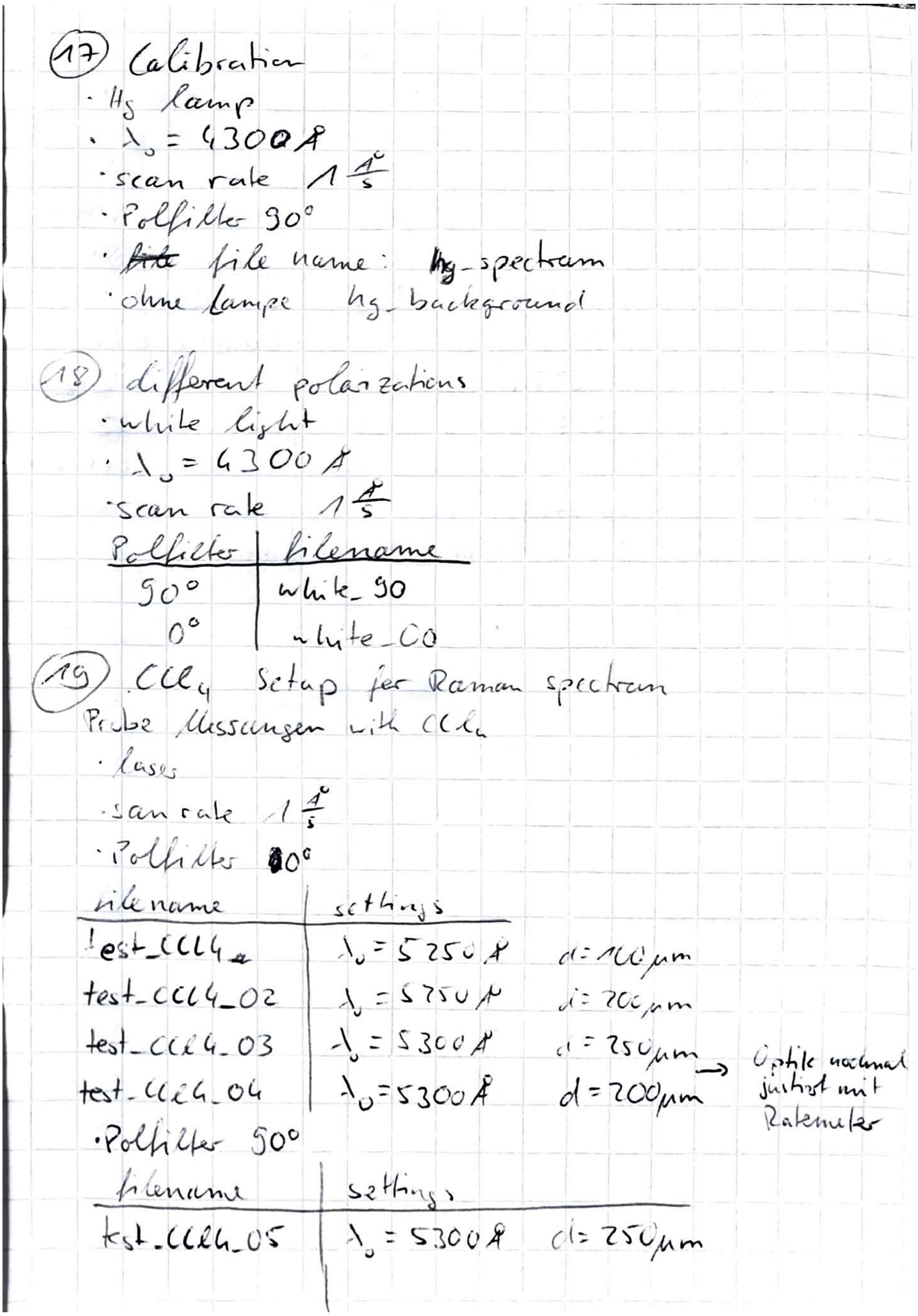
- set up as in ⑭
- white lamp
- $\lambda_0 = 3800$ Å
- scan rate 2 $\frac{Å}{s}$
- file name: bound
- Polfilter: 90°

⑯ Slit width

- Hg lamp
- $\lambda_0 = 5400$ Å
- scan rate 0.5 $\frac{Å}{s}$
- Polfilter: 90°

slit width	file name
200 μm	slit_200
150 μm	slit_150
100 μm	slit_100
50 μm	slit_050
20 μm	slit_020
250 μm	slit_250

→ now use always ~~d=200μ~~ $d = 100\,\mu m$



⑰ Calibration

- Hg lamp
- $\lambda_0 = 4300$ Å
- scan rate $1 \frac{Å}{s}$
- Polfilter 90°
- ~~file~~ file name: hg_spectrum
- ohne Lampe hg_background

⑱ different polarizations

- white light
- $\lambda_0 = 4300$ Å
- scan rate $1 \frac{Å}{s}$

Polfilter	filename
90°	white_90
0°	white_00

⑲ CCl_4 Setup for Raman spectrum

Probe Messungen with CCl_4

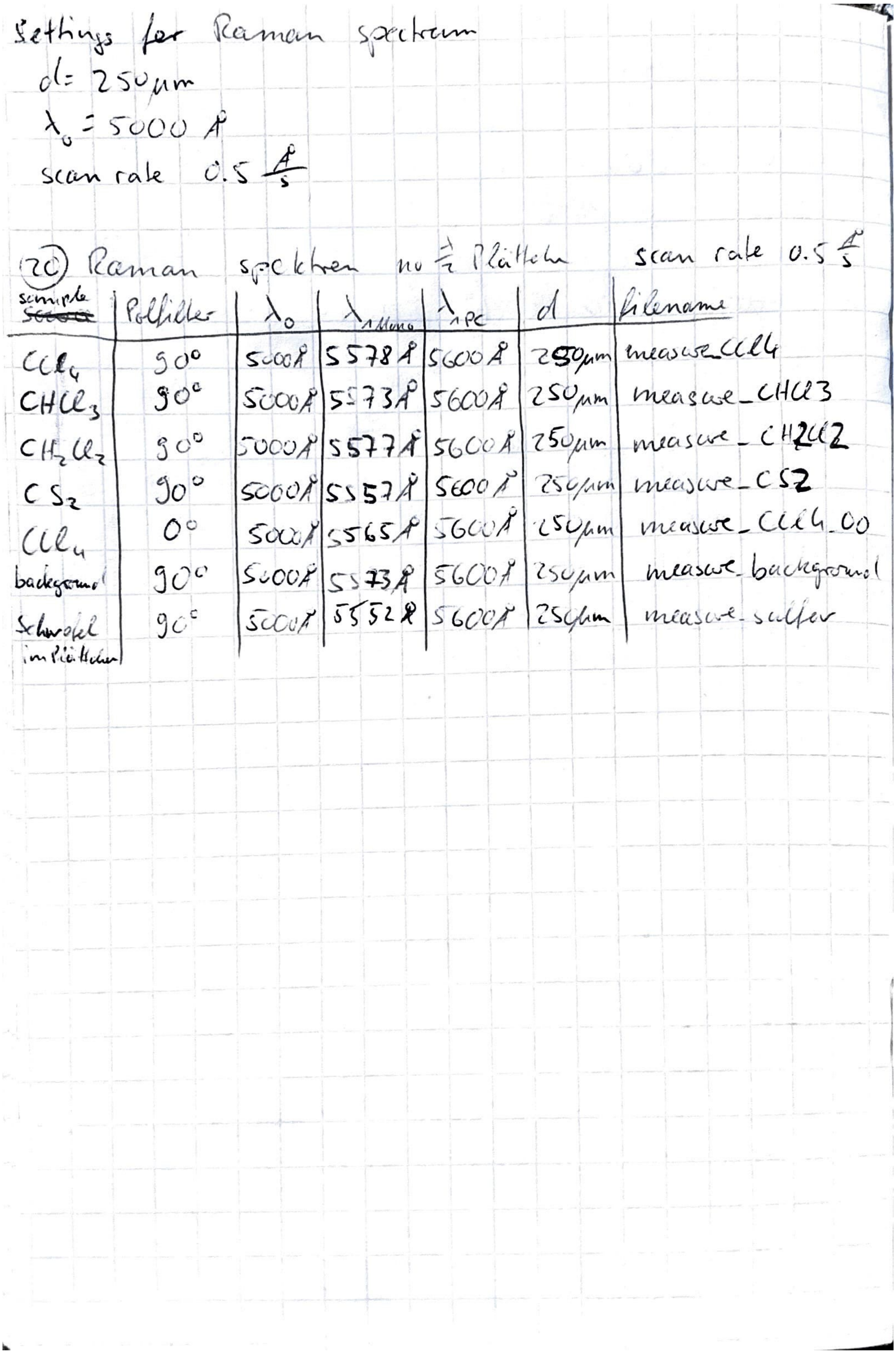
- laser
- scan rate $1 \frac{Å}{s}$
- Polfilter 0°

filename	settings	
test_CCl4	$\lambda_0 = 5250$ Å	$d = 100\,\mu m$
test_CCl4_02	$\lambda_0 = 5750$ Å	$d = 200\,\mu m$
test_CCl4_03	$\lambda_0 = 5300$ Å	$d = 250\,\mu m$
test_CCl4_04	$\lambda_0 = 5300$ Å	$d = 200\,\mu m$

→ Optik nochmal justiert mit Ratemeter

- Polfilter 90°

filename	settings	
test_CCl4_05	$\lambda_0 = 5300$ Å	$d = 250\,\mu m$



Settings for Raman spectrum

$d = 250\,\mu m$

$\lambda_0 = 5000$ Å

scan rate $0.5\,\frac{Å}{s}$

(2C) Raman spektren no $\frac{\lambda}{2}$ Plättchen scan rate $0.5\,\frac{Å}{s}$

sample	Polfilter	λ_0	λ_{1Mono}	λ_{1PC}	d	filename
CCl_4	90°	5000Å	5578Å	5600Å	250μm	measure_CCl4
$CHCl_3$	90°	5000Å	5573Å	5600Å	250μm	measure_CHCl3
CH_2Cl_2	90°	5000Å	5577Å	5600Å	250μm	measure_CH2Cl2
CS_2	90°	5000Å	5557Å	5600Å	250μm	measure_CS2
CCl_4	0°	5000Å	5565Å	5600Å	250μm	measure_CCl4_00
background	90°	5000Å	5573Å	5600Å	250μm	measure_background
Schwefel im Plättchen	90°	5000Å	5552Å	5600Å	250μm	measure_sulfur