Use this document as a template

Millikelvin Confocal Microscopy of Semiconductor Membranes and Filter Functions for Unital Quantum Operations

Customise this page according to your needs

Tobias Hangleiter*

August 11, 2025

* A LaTeX lover/hater

The harmony of the world is made manifest in Form and Number, and the heart and soul and all the poetry of Natural Philosophy are embodied in the concept of mathematical beauty.

– D'Arcy Wentworth Thompson

Contents

List of Figures

Software

The following open-source software packages were developed (at least partially) during the work on this thesis.

[1] Tobias Hangleiter, Isabel Nha Minh Le, and Julian D. Teske, *Filter_functions* version v1.1.3, May 14, 2024. Zenodo. DOI: 10.5281/ZENODO.4575000.

[2] Tobias Hangleiter, *Lindblad_mc_tools* Aug. 8, 2025.

[3] Tobias Hangleiter, *Mjolnir* Aug. 8, 2025.

[4] Tobias Hangleiter, Simon Humpohl, Max Beer, and René Otten, *Python-Spectrometer* version 2024.11.1, Nov. 21, 2024. Zenodo. DOI: 10.5281/ZENODO.13789861.

[5] Tobias Hangleiter, Simon Humpohl, Paul Surrey, and Han Na We, *Qutil* version 2024.11.1, Nov. 21, 2024. Zenodo. DOI: 10.5281/ZENODO.14200303.

Part I

A Flexible Python Tool For Fourier-Transform Noise Spectroscopy

Part II

Characterization and Improvements of a Millikelvin Confocal Microscope

Part III

Optical Measurements of Electrostatic Exciton Traps in Semiconductor Membranes

The `mjolnir` measurement framework

3

To facilitate the optical experiments in this part of the present thesis, I wrote a software package that controls the measurement workflow and all other interactions with the setup described in Part II, `mjolnir`[1] [1]. The package is written in Python, open source, and has a documentation as well as some rudimentary tests.[2] In this chapter, I lay out the goals and non-goals behind its development and briefly describe the design as well as the typical workflow.

1: https://git-ce.rwth-aachen.de:qutech/lab_software/python-mjolnir

2: Testing code interacting with hardware is notoriously difficult to achieve in an automated manner.

3.1 Rationale

There exist various software solutions for measurement control, ranging from commercial (*e.g.*, Keysight Labber[3]) over open source (`quantify`,[4] `labcore`[5]) to home-built (`elicit`,[6] `QuMADA`[7]). While most of these frameworks attempt to be universal and not tailored towards a specific type of experiment or setup, some bias always exists and most are geared towards fully electrical experiments. On the other hand, software for optical experiments naturally exists as well, but it suffers the same shortcomings coming from a different direction, and there is fairly little on offer for experiments at the interface of quantum optics and transport. An added complication is the availability of drivers. Striving for full automation of the setup, a number of different drivers are required given the complexity of the setup and the mix of electrical, optical, and mechanical instruments (Part II). Thus, `QCoDeS` forms a good basis for our specific needs due to its large driver coverage. However, while `QCoDeS` provides measurement functionality, their definition involves a large amount of glue code that needs to be duplicated for each measurement. The `doNd`[8] functionality abstracts some of this away for multi-dimensional loops but is not flexible enough for our requirements.

3: https://www.keysight.com/de/de/products/software/application-sw/labber-software.html

4: https://quantify-os.org/

5: https://github.com/toolsforexperiments/labcore

6: https://git-ce.rwth-aachen.de/qutech/frameworks/qool-tool

7: https://github.com/qutech/qumada

8: https://microsoft.github.io/Qcodes/api/dataset/index.html#qcodes.dataset.dond

Because the requirements are rather specific, `mjolnir` is therefore single-minded. It does not aspire to be suitable for applications other than the ones it is designed for.

3.2 Instrument abstraction

Central to the `mjolnir` package is the abstraction of physical instruments into logical ones, thereby grouping logical functionality provided by different physical devices. Take for instance the tunable continuous-wave (cw) M Squared Solstis laser. It is cooled by a Thermotek T225p chiller and pumped by a Lighthouse Photonics Sprout-G diode-pumped solid state (DPSS) laser. Behind its exit aperture, a Thorlabs MFF101/M acts as a shutter, a neutral-density (ND) filter mounted on a Thorlabs K10CR1/M controls the output power, while a Thorlabs PM100D monitors the power at the optical head and an Attocube AMC100 controls the polarization state of the optical head. Thus, seven different instruments from five different manufacturers are required to control the illumination state of the sample. As a user, however, one would simply like to en- or disable the laser and set wavelength, power, and polarization configuration of the optical head.

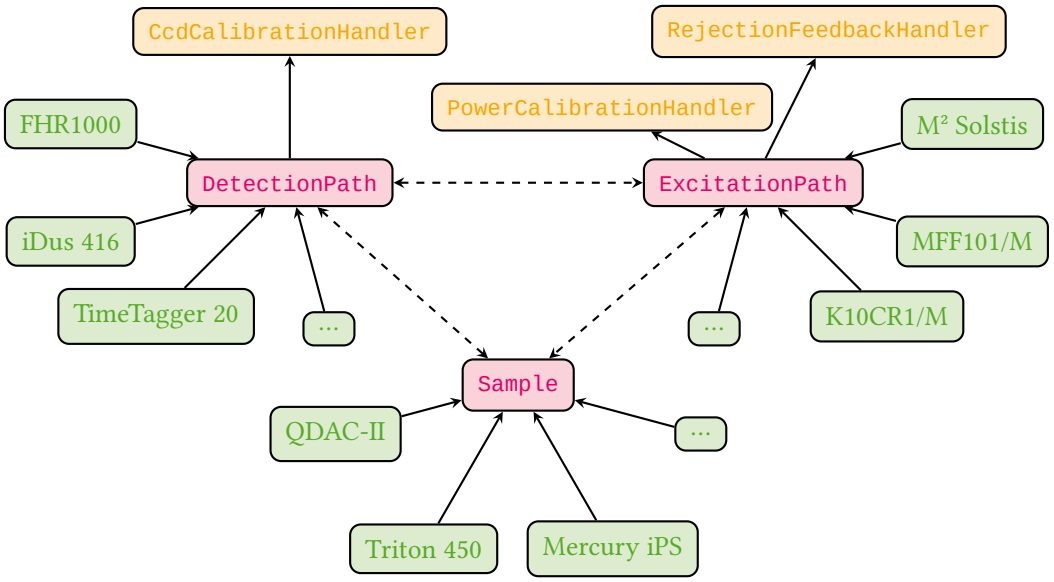

Figure 3.1: Abstraction of physical instruments. `mjolnir` defines three logical instrument classes (magenta) that represent different aspects of the experiment and group various physical instruments (green). Logical instruments expose various logical parameters that include communication with multiple physical instruments. `Sample` can be subclassed to suit the particular type of device under investigation. Calibration handlers (orange) are controlled by the logical instruments governing the optical path.

To simplify and abstract away the particularities of the physical instruments providing control over parts of a user-facing parameter such as the excitation wavelength,[9] three logical instruments implemented as subclasses of the `QCoDeS Instrument` class govern the experimental apparatus:

9: For example, the power meter needs to be informed when changing the wavelength to keep the calibration up to date.

1. `ExcitationPath`. Controls all physical instruments related to illumination of the sample, including the white light source.
2. `DetectionPath`. Controls instruments related to the detection of radiation emitted by the sample, *i. e.*, the spectrometer, charge-coupled device (CCD), and photon counting card.
3. `Sample`. Controls the QDevil QDAC-II voltage source, cryostat, and magnet, and implements a software representation of the device under test (DUT).

Figure 3.1 shows a graph outlining the relationship between physical and logical instruments. In the following, I give a brief overview of the functionalities comprised by these logical instruments.

3.2.1 Excitation path

As already outlined above, the `ExcitationPath` object controls everything related to the illumination of the sample. The `active_light_source` parameter switches between laser, white light, and no illumination. Turning on the laser comprises enabling the chiller, switching on the pump laser, waiting for it to ramp up, and asserting the wavelength lock is acquired. For safety reasons, confirmation by the user that they are physically in the lab is required. Furthermore, the `ExcitationPath` instrument implements setting the excitation power by means of a calibration of the ND filter, see Section 3.3. Since the output power of the laser varies depending on wavelength, the object furthermore provides a `wavelength_constant_power` parameter that automatically recalibrates

the power once a new wavelength is set. Finally, it manages the state of the automatic excitation rejection control, see Section 3.3.

3.2.2 Detection path

The `DetectionPath` object is responsible for the optical analysis. Chiefly, it manages the Horiba FHR1000 spectrometer and provides convenient shorthands for selecting grating (`active_grating`) and exit port (`active_detection_path`, either `"ccd"` or `"apd"`). It handles initialization of the spectrometer and the CCD including the CCD cooler and oversees calibration of the CCD pixel-to-wavelength relation, see Section 3.3. From the CCD calibration and its pixel size, the dispersion at the imaging plane of the currently selected grating can be obtained, allowing the user to set the monochromator bandwidth of the spectrometer in terms of a wavelength window rather than the more abstract exit slit width.

3.2.3 Sample

The `Sample` class initializes the QDevil QDAC-II digital-to-analog converter (DAC) and magnet power supply. Mainly, though, it serves as an abstraction of the actual DUT and its "control knobs" such as gates. Users implement subclasses for different sample designs. As the samples we are concerned with in the present thesis, I discuss the main implementation, `TrapSample` representing samples with exciton traps. On a single sample, there are any number of traps consisting of one or both sets of top and bottom "central" and "guard" gates. Each trap is implemented as a `QCoDeS InstrumentModule`, `Trap`, that is part of a `ChannelList`. The currently active trap (*i.e.*, the one currently in focus by the microscope) is selected by the `active_trap` parameter and accessible through the `trap` property of the `TrapSample` object. A `Trap` exposes the virtual gate parameters[10] `difference_mode` and `common_mode` (see **??**) as the difference and sum of top and bottom gate voltages, respectively, as well as the unmodified top and bottom gate voltage and corresponding leakage current parameters. DAC channels are mapped to their corresponding `Trap` parameters using `yaml` configuration files specific to each sample and hosted in a separate repository.

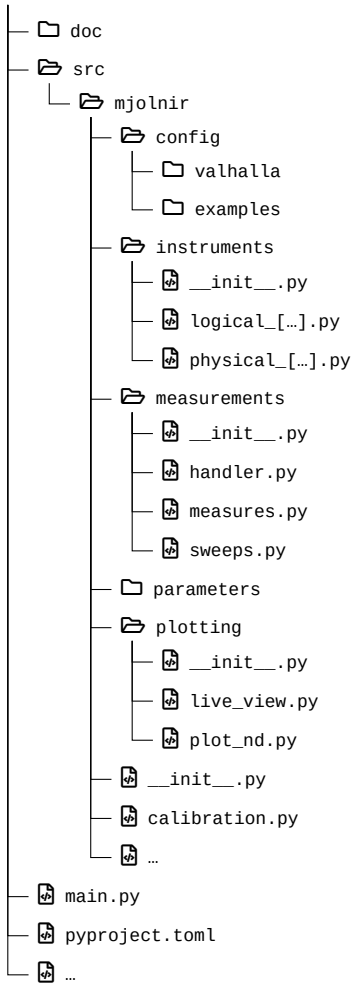

Figure 3.2: Source tree structure of the `mjolnir` package. Logical `QCoDeS` instruments and parameters are defined in the `instruments` and `parameters` modules, respectively. Instruments are configured using `yaml` files located in a `config` subdirectory. The `measurements` module provides classes for the abstraction of measurements using `QCoDeS` underneath. Live plots of instrument data as well as a plot function for multidimensional measurement data are defined in the `plotting` module. `calibration.py` contains routines for power, CCD, and excitation rejection calibrations. The `main.py` file is a code cell-based script that serves as the entrypoint for measurements.

10: Using the QDevil QDAC-II's built-in `Arrangement_Context` virtual gate functionality.

3.3 Calibrations

`mjolnir` implements three automatic calibration procedures in the classes `CcdCalibrationHandler`, `PowerCalibrationHandler`, and `RejectionFeedbackHandler` (see Figure 3.1). As their names suggest, these handle calibration of the CCD pixel-to-wavelength conversion, the power transmission through the rotatable ND filter, and the waveplate and polarizer angles for optimal laser rejection, respectively.

3.3.1 CCD calibration

Because the dispersion of the spectrometer gratings depends to a small degree on wavelength, the calibration function converting horizontal CCD screen pixels into wavelengths needs to be updated for every central wavelength selected using the grating. In principle, this requires measuring the position of several reference wavelengths on the CCD

screen and fitting a polynomial of second or third degree. As the tunable cw laser is coupled to a wavelength meter with built-in calibration lamp (HighFinesse WS6), it lends itself naturally as a source of the reference wavelength. In practice, the deviation from an established calibration function is usually small enough that simply shifting the center pixel (that is, the zeroth-order polynomial term) suffices for reasonably small wavelength ranges of tens of nanometers. The `CcdCalibrationHandler` implements three different calibration schemes specified as the `ccd_calibration_update_mode` parameter of the `DetectionPath` instrument:

1. `"full"`. The pixel position of the maximum intensity of nine equidistant wavelengths centered around the grating central wavelength and spread out over the entire range of the CCD screen (~ 45 nm for the 600 gr/mm grating and ~ 9 nm for the 1800 gr/mm grating) is measured iteratively and fitted to a third-degree polynomial.
2. `"fast"`. The laser is set to the grating central wavelength and the zeroth-order coefficient of the calibration polynomial is shifted by the pixel distance between the peak and the horizontal center of the CCD screen.
3. `"dirty"`. The zeroth-order coefficient of the calibration polynomial is shifted by the difference in nominal grating and laser wavelengths.

By default, the `"dirty"` mode is enabled as it does not require any grating moves or wavelength changes and is therefore the fastest. Rather than recalibrating, old calibration data can also be loaded from disk if it is not older than a user-specified date. Finally, note that the spectrometer grating motor also requires occasional recalibration. Both the coefficient of linearity (converting motor steps to selected central wavelength) and the offset in motor steps can change over time. Currently, this calibration is not automated but is instead compensated for by shifting the central wavelength on the CCD screen. In principle, the driver for the Horiba FHR1000 spectrometer implements all necessary functionality for automating the calibration procedure already.

3.3.2 Power calibration

The ND filter[11] mounted on the Thorlabs K10CR1/M rotation stage has a continuously variable optical density (OD) (Equation 1.26) of 0.04 to 4.0, which depends quite sensitively on the wavelength. In order to allow reliably setting a specified excitation power at any excitation wavelength, the `mjolnir` package automates the calibration of the filter angle against transmitted power. Two modes are implemented, `"full"` and `"fast"`. In the former mode, the transmitted power is sampled at a given angle interval and the resulting data fitted using a quadratic smoothing spline. Figure 3.3 shows a typical measurement and spline fit at 795 nm. The specified power is then set by inverting the spline and evaluating at the given power.

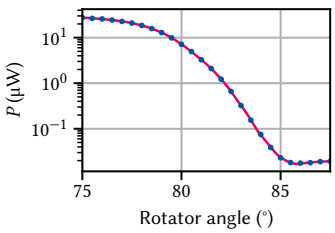

Figure 3.3: Power calibration using the continuously variable ND filter. Error bars from statistics are too small to see. Magenta line is a quadratic smoothing spline.

11: Thorlabs NDC-25C-4 [2].

As performing the full calibration is quite time-consuming and furthermore the inversion not fully reliable, the `"fast"` calibration update mode sets the rotator angle by performing a noisy optimization. Here, first the last valid full calibration is loaded and its spline scaled to the current power level. Then, the residual sum of squares between the target power and the power meter reading is minimized using the `noisyopt` package [3] with the starting value given by the angle predicted by the calibration spline.

3.3.3 Rejection feedback

The excitation rejection by means of cross-polarization extinction has already been discussed in Subsection 1.1.4. As shown there, the OD achieved by cross-polarizing the excitation beam with the analyzer mounted in the detection path reaches a maximum close to 6 and is extremely sensitive to small variations in the angles of the $\lambda/4$- and $\lambda/2$-plates controlling polarization state. Close to the optimum, the OD is well-described by a parabola as function of those angles, with a second-order polynomial coefficient on the order of $-0.15/\mathrm{mdeg}^2$. Since the optics including the waveplates display chromatic dispersion, the excitation rejection needs to be re-optimized whenever the laser wavelength is changed, and the exponential angular dependence of the transmitted intensity places high demands on the dynamic range of the photo-detector used for measurement. Therefore, the reflected laser intensity is measured with the two avalanche photodiode (APD) signal digitized by the Swabian Instruments Time Tagger 20 on the side exit port of the spectrometer.[12] The Time Tagger can reliably cover a count rate range from 10^2 cps to 10^6 cps, compared to the Andor iDus 416 CCD covering only 10^3 cps to 10^5 cps.

12: The signals from both APDs are combined to achieve the best signal-to-noise ratio (SNR), see Part II.

In the `mjolnir` package, the calibration is then carried out as a bivariate noisy minimization using `noisyopt` [3]. Upon initialization the power is set to a small value to avoid overexposing the detectors. Then, the spectrometer is set to select the laser wavelength, the side exit port is selected, and the optimization over the $\lambda/4$- and $\lambda/2$-angles is run within conservatively set bounds of $[-2.5°, 2.5°]$, which has been found to be a good compromise of robustness and convergence speed. A caching mechanism stores the optimal angles together with the current wavelength, which serves as a starting guess upon subsequent calibration runs. The calibration is quite tedious and time-consuming for several reasons. Chiefly, moving the waveplate angles is slow simply because it is a mechanical movement, and optimizers typically do not take into account the distance between subsequent sample points, potentially resulting in unnecessarily inefficient exploration of the parameter space. Incorporating a penalty on the sample distance into the optimization algorithm would hence improve the convergence speed. Additionally, due to vibrations in the cryostat (Chapter 2), the root mean square (RMS) intensity of the back-scattered laser is large – and larger the closer the excitation rejection is to its optimal value[13] – and thus requires long averaging times for a robust measurement. Finally, the algorithm outlined above fails when the starting guess is too far away from the optimum, such as when the new wavelength is on the order of tens of nm away from the previously optimized one. A good – albeit time-consuming – strategy is then to incrementally update the wavelength.

13: The count rate can vary by up to a factor of two with the frequency of the pulse tube refrigerator (PTR) pulses.

3.4 Measurement routines

Measurements in `mjolnir` use the `QCoDeS` infrastructure for data acquisition and storage. To abstract away code for repetitive setup and teardown tasks, the user interacts with a `MeasurementHandler` object through the `measure()` method. The class can be subclassed to customize the aforementioned tasks for different types of experiments (such as those that involve illumination with the laser and detection with the CCD, `LaserCcdMeasurementHandler`), or add default parameters to every measurement (such as leakage currents or the laser power). A measurement's independent parameters are passed to `measure()` as instances of a `Sweep`

class which completely determines the measurement structure. `Sweep` objects support syntactic sugar for concatenation (`@`), nesting (`|`), and parallelization (`&`). Dependent parameters (those that should be measured) can be simple parameters or instances of a `Measure` class.[14] The state of external parameters, that is, parameters that are not varied or measured during a measurement, can be defined for the duration of the measurement using the `parameter_contexts` argument, a mapping from `QCoDeS` parameters to valid values thereof. This uses the `Parameter.set_to()` context manager to set the parameters to the given values before the measurement starts and restore their original values upon exit in a controlled manner, allowing users to ensure their device is in a well-defined state. Custom setup and teardown tasks can furthermore be specified through the `add_before_run` and `add_after_run` arguments, respectively.

14: Besides some optimization for measurements of parameters delegating from the same physical parameter, the latter exists mostly for (primitive) live plotting.

```
handler = LaserCcdMeasurementHandler(station)
power_sweep = GridSweep(excitation_path.power_at_sample,
                        rng=(5e-9, 5e-7), num_points=11,
                        spacing='geom')
gate_sweep = GridSweep(sample.trap.central_difference_mode,
                       rng=(-2, -1), num_points=51))
sweeps = power_sweep | gate_sweep
dataset = handler.measure(
    sweeps,
    parameter_contexts={
        sample.trap.central_common_mode: -1,
        detection_path.central_wavelength: 825,
    },
    exposure_time=2
)
```

Listing 3.1: Setup and measurement workflow using the `mjolnir` package. `station` is a `QCoDeS Station` object managing the instruments. The `sweeps` object describes a nested loop on whose inner iteration the difference mode parameter of the trap's central gate is swept over a linear grid and on whose outer iteration the laser power, adjusted for the beam splitter ratio, is swept over a logarithmically spaced grid. No dependent parameters (`Measure` objects) need to be explicitly specified as the `LaserCcdMeasurementHandler` measures the CCD spectrum as well as laser power and leakage currents of the swept gates by default. The `parameter_contexts` argument is used to set the spectrometer wavelength to 825 nm and the common mode voltage of the active trap to −1 V. The `exposure_time` argument is passed through to the `initialize` method, where it is used to set up the CCD for acquisition.

A typical workflow for a photoluminescence (PL) measurement using laser and CCD is sketched in Listing 3.1. The `MeasurementHandler` object automatically takes care, among others, of arming the CCD, acquiring a background image in the dark, and opening the laser shutter for the duration of the measurement. Acquired data is saved to the `QCoDeS` database, but a helper function exporting to the `xarray Dataset` format is available. Saved together with the data are a snapshot of the `QCoDeS Station` as well as, optionally, arbitrary custom metadata. More details and examples on the measurement functionality can be found in the documentation[15]. Finally, I note that the measurement functionality in `mjolnir` is independent of the instrument abstraction and calibration logic and as such could be easily[16] replaced or augmented by that of other `QCoDeS`-based frameworks such as `quantify` or `labcore` to make use of their infrastructure. I leave this option for future work.

15: https://qutech.pages.git-ce.rwth-aachen.de/lab_software/python-mjolnir/

16: There is always *some* glue code required.

3.5 Plotting

Optical measurements using `mjolnir` often produce multi-dimensional datasets by virtue of the default parameter measured, the CCD spectrum, being vector-valued or *batched*. To visualize such datasets with more than two dimensions, `mjolnir` provides the `plot_nd()` function

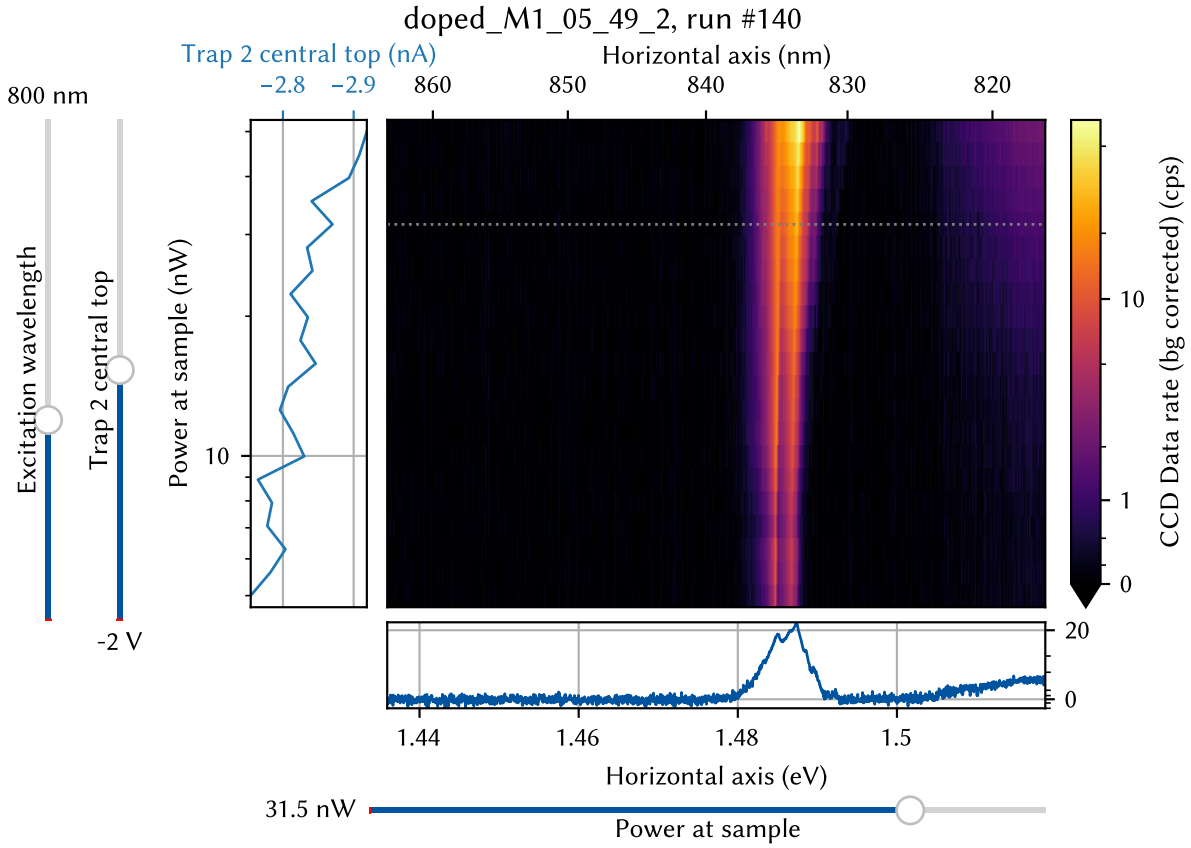

Figure 3.4

that plots 2D-slices through the data as a false-color and generates interactive slider widgets allowing users to specify the slice coordinates.

Part IV

A Filter-Function Formalism For Unital Quantum Operations

Appendix

Bibliography

[1] Tobias Hangleiter, *Mjolnir* Aug. 8, 2025 (cited on page 27).

[2] Thorlabs. *NDC-25C-4 Unmounted Continuously Variable ND Filter, Ø25 mm, OD: 0.04 - 4.0*. URL: https://www.thorlabs.com/thorproduct.cfm?partnumber=NDC-25C-4 (visited on 08/11/2025) (cited on page 30).

[3] Andreas Mayer et al. "Diversity of Immune Strategies Explained by Adaptation to Pathogen Statistics." In: *Proceedings of the National Academy of Sciences* 113.31 (Aug. 2, 2016), pp. 8630–8635. DOI: 10.1073/pnas.1600663113 (cited on pages 30, 31).

Special Terms

A
APD avalanche photodiode. 31

B
BS beam splitter. 32

C
CCD charge-coupled device. 28–32
cw continuous-wave. 27, 30

D
DAC digital-to-analog converter. 29
DPSS diode-pumped solid state. 27
DUT device under test. 28, 29

N
ND neutral-density. 27–30

O
OD optical density. 30, 31

P
PL photoluminescence. 32
PTR pulse tube refrigerator. 31

R
RMS root mean square. 31

S
SNR signal-to-noise ratio. 31

Declaration of Authorship

I, Tobias Hangleiter, declare that this thesis and the work presented in it are my own and has been generated by me as the result of my own original research.

I do solemnly swear that:

1. This work was done wholly or mainly while in candidature for the doctoral degree at this faculty and university;
2. Where any part of this thesis has previously been submitted for a degree or any other qualification at this university or any other institution, this has been clearly stated;
3. Where I have consulted the published work of others or myself, this is always clearly attributed;
4. Where I have quoted from the work of others or myself, the source is always given. This thesis is entirely my own work, with the exception of such quotations;
5. I have acknowledged all major sources of assistance;
6. Where the thesis is based on work done by myself jointly with others, I have made clear exactly what was done by others and what I have contributed myself;
7. Parts of this work have been published before as:

[1] Pascal Cerfontaine, Tobias Hangleiter, and Hendrik Bluhm. "Filter Functions for Quantum Processes under Correlated Noise." In: *Physical Review Letters* 127.17 (Oct. 18, 2021), p. 170403. DOI: 10.1103/PhysRevLett.127.170403.

[2] Thomas Descamps, Feng Liu, Tobias Hangleiter, Sebastian Kindel, Beata E. Kardynał, and Hendrik Bluhm. "Millikelvin Confocal Microscope with Free-Space Access and High-Frequency Electrical Control." In: *Review of Scientific Instruments* 95.8 (Aug. 9, 2024), p. 083706. DOI: 10.1063/5.0200889.

[3] Tobias Hangleiter, Pascal Cerfontaine, and Hendrik Bluhm. "Filter-Function Formalism and Software Package to Compute Quantum Processes of Gate Sequences for Classical Non-Markovian Noise." In: *Physical Review Research* 3.4 (Oct. 18, 2021), p. 043047. DOI: 10.1103/PhysRevResearch.3.043047.

[4] Tobias Hangleiter, Pascal Cerfontaine, and Hendrik Bluhm. "Erratum: Filter-function Formalism and Software Package to Compute Quantum Processes of Gate Sequences for Classical Non-Markovian Noise [Phys. Rev. Research 3, 043047 (2021)]." In: *Physical Review Research* 6.4 (Oct. 16, 2024), p. 049001. DOI: 10.1103/PhysRevResearch.6.049001.

Aachen, August 11, 2025.