A GUI and Computer Interface for ODMR Instrumentation

Jalan Ziyad

February 16, 2019

Abstract

In the process of ODMR experiments, spectroscopy data have to be efficiently collected, compiled and analyzed. As large amounts of data are collected, across numerous runs, through repetitive processes, this problem lends itself easily to automation. This had been achieved through various Python scripts for instrument calibration, data collection, graphing, and equation fitting. We saw an opportunity to improve this workflow through an intuitive graphical user interface (GUI). The result is a single program that combines the scripts, and allows for useful features such as easy parameter entry, live graphing, aborting data collection, saving and changing filenames. This was accomplished through the Tkinter framework for GUI programming, modules for numerical and plotting needs (SciPy, NumPy, and Matplotlib), and various multithreading modules for coordinating concurrent processes.

1 Introduction

Nitrogen-vacancy (NV) centers in diamonds have been shown to be useful as high precision magnetometers at short length scales. Detection is achieved through optical probing of the NV centers at various frequencies and measuring the resulting fluorescence. This is called optically detected magnetic resonance (ODMR). The spectroscopy can be analyzed and the magnetic field can then be determined. In order to minimize statistical error, multiple values for each frequency were taken and averaged for multiple runs. The efficiency of the data collection is largely determined by how the instrumentation interfaces with a computer, since the duration and yield of data collection is so large. The instruments interfaced with the computer could be calibrated, controlled, and read out digitally through various Python scripts. Previously these scripts were run one-by-one in the python shell, and if variables, such as filenames, voltage, etc., needed to be changed; the scripts had to be manually edited. The long data collection process could only be stopped if the script was forcibly quit. A graphical user interface (GUI) was implemented that collects the scripts into a single program that addresses all of these problems. The GUI relies on a feature in certain operating systems called multithreading. It will be shown that multithreading, while allowing for some of the most useful features of the GUI, also presented the most challenging technical problems. The following sections will give a summary of NV ODMR, the instrumentation, and an overview of techniques and the frameworks used to make the GUI.

2

2.1 NV-center ODMR

Nitrogen-vacancy centers are point defects in diamonds where two adjacent carbon atoms are replaced by a nitrogen atom and a vacancy. This creates a system of two unpaired electrons (spin $S = -1, 0, 1$) with two triplet states and two intermediate singlet states. In the triplet states, $S = 0$ is the lowest state, and $S = \pm 1$ forms a degenerate state in the absence of a magnetic field. The energy difference between spin states corresponds to the microwave range. The energy of an NV-center in the lower triplet state can be excited by a 546nm (green) light which we can assume preserves the spin. For $S = 0$ the state has a high likelihood of decaying into the ground state giving off 689 nm (red) fluorescence. Otherwise $S = \pm 1$ has a significant likelihood of decaying via the singlet states without visible fluorescence. Therefore if the system is excited by green laser light while at resonance with a microwave, then the spin states, $S = \pm 1$, can be detected as a dip in intensity in the spectrum. Under a magnetic field the separation between $S = -1$ and $S = 1$ peaks indicates the field strength. The peaks can be determined through an equation fit.

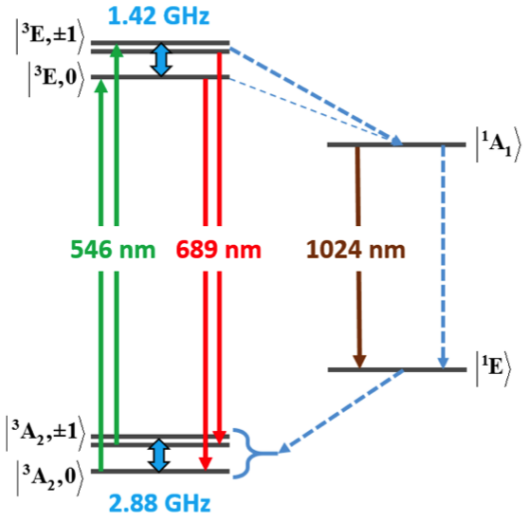

Figure 1: Energy levels of an NV-Center. Courtesy of wikimedia.

2.2 Instrumentation

The main instruments that needed to interact with the computer were a voltmeter (measuring the fluorescence via a photo-diode), a microwave source, and a lock-in amplifier. A lock-in amplifier works by modulating the microwave intensity at a specific frequency ω. The intensity measured by the photo-diode will be oscillating at the same frequency but with some phase difference ϕ. The lock in amplifier multiplies the two signals together and applies a low-pass filter to it and sends the output to the voltmeter. The lock-in amp has a setting that tunes the phase shift. The computer is able to read the voltmeter and control the lock-in so calibration can be automated to get the maximum output. Different values of phase are swept over automatically and the signal strength is displayed on the computer. The optimal phase can then be input manually on the lock-in. For the spectroscopy, the computer controls the power and frequencies of the microwaves. The computer interfaces with the instruments, through standard GPIB cables, which allow the instruments to be programmed in Python using National Instruments' VISA architecture (via PyVISA).

2.3 Programming tools

2.3.1 GUI tools

The framework used for the front-end of the GUI is TKinter. Tkinter handled the creation of a window, buttons, drop down menus, checkboxes, and entry boxes.

2.3.2 Numerical tools and fitting

SciPy, and NumPy were employed for numerical needs. They provide accessibility for scientific numbers, matrices, and a larger library of mathematical functions and operations. Matplotlib was used for plotting. SciPy handles Lorentzian and Gaussian fitting using linear regression, and for non-zero magnetic fields, where the spectra have more than two peaks, χ^2 minimization is used.

2.3.3 Threading

Much of the power of the GUI comes from threading. Multithreading is a way to coordinate concurrent activities within the same process. A function can be called as its own thread, running concurrently with the other threads. This allows for functions that are able to be responsive; they can listen for additional commands from others. This is key for the live graphing and abort functions of the GUI. In live graphing one function collects data while a separate function graphs. An abort command can be sent to a certain thread that can stop a function short of completion.

The threads interact through various shared objects including events, queues, and locks. Queues are a way to send variables and data between threads, which can be an issue if two threads try to access an object at the same time. A lock allows a thread to have sole access to an object, to avoid

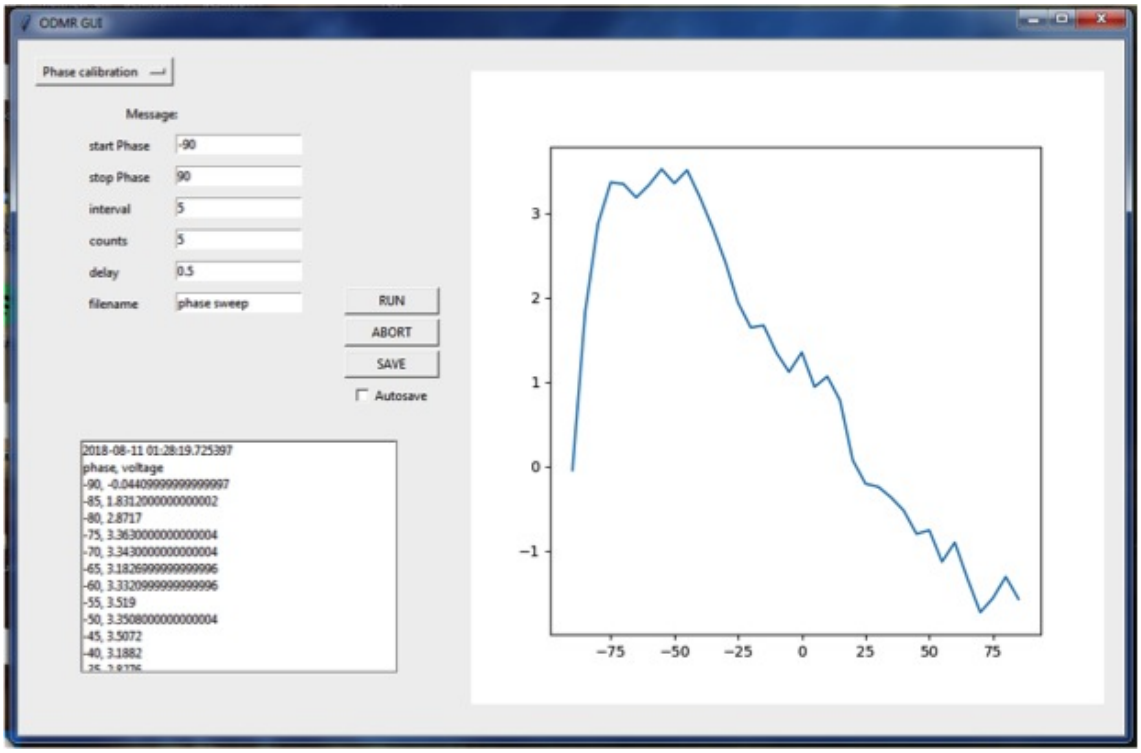

Figure 2: Phase calibration window

such overwriting problems. An event is a Boolean (true or false value) that all threads have access to. This is useful for indicating the state of the main thread to the rest, to coordinate tasks.

The GUI uses queues to pass spectroscopy data, parameters, and save data between functions. The abort function sets a thread event that a running thread listens for. Once the event is detected the function then shuts down. There is also threading in the background of both Matplotlib and Tkinter. To ensure the compatibility of the two frameworks, the backend of Matplotlib had to be changed to "FigureCanvasTkAgg", allowing plots to be embedded into the Tkinter window. Matplotlib's plot functions could only be used from the main thread, or else there would be thread errors and lock-ups.

3 Results

The final product of the three main functions of the GUI (phase calibration, data collection, and fitting) will now be reviewed. The dropdown menu at the top-left of the window selects between the three functions.

3.1 Phase calibration

Phase calibration assists in the tuning of the lock-in amp. "Start Phase" and "stop Phase" give the range of phase values to be optimized over (in degrees). "Interval" is how many degrees it steps by, "counts" is how many times it takes a voltage at a given phase, and "delay" is the time between taking measurements (in seconds). Once desired values are entered (or defaults are used), the RUN button can be clicked to run the calibration. It can be stopped at any time with the abort button. The data box in the bottom left of the window displays the data, while it is graphed live on the right side of the window. The SAVE button saves the data to a file specified by "filename". "Autosave" can also be checked to automatically save when RUN is pressed. Once the phase calibration is run, the peak has to be manually identified and entered into the lock-in amp.

3.2 Data collection

"Start Freq" and "Stop Freq" set the range of the microwaves, and "Step Freq" is how much it steps by (all in Hz). "RF voltage" is the power of the microwaves (in mV). "Delay" is how long between measurements (in seconds), "counts" is how many values in a row it takes of a single frequency (averaged), and "# of runs" gives how many times the spectrum is run through. Saving and aborting works the same as in the Phase calibration. Microwave frequency vs Intensity (upside down) is graphed live on the right side of the window.

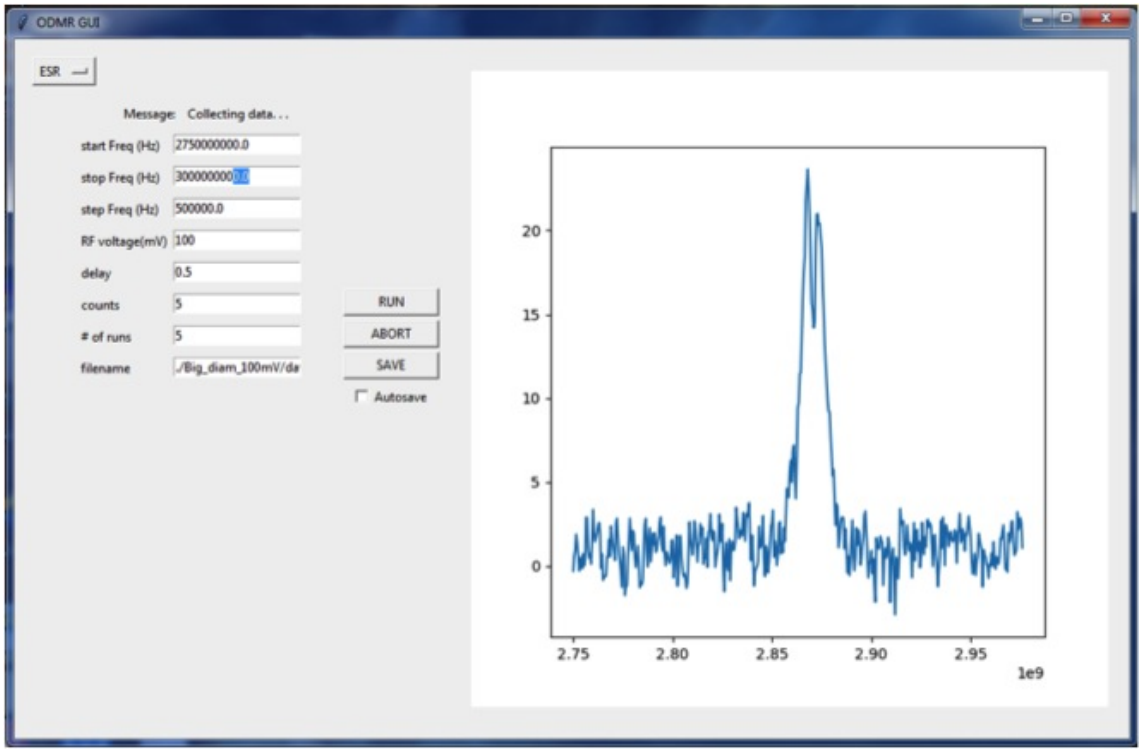

Figure 3: Data collection window

3.3 Data fitting

There is an additional drop down menu that lets you select between the three different fitting functions; two for zero field fitting (Lorentzian and Gaussian), and one for non zero magnetic fields (Multipeak fitting). Spectra data file is where you specify the name and location of the data you are fitting, filename is the name of the file storing the fit equation and parameters. The data box displays the fit parameters. The multipeak fitting has the additional entries for "B magnitude", "theta" and "phi" as well as the "Plot Guess" button. This is to predict a B field to aid in the χ^2 minimization of the fitting function. The entry Zero field file is also added for calibration. The saving works as normal but abort is unnecessary as the fit functions are fast.

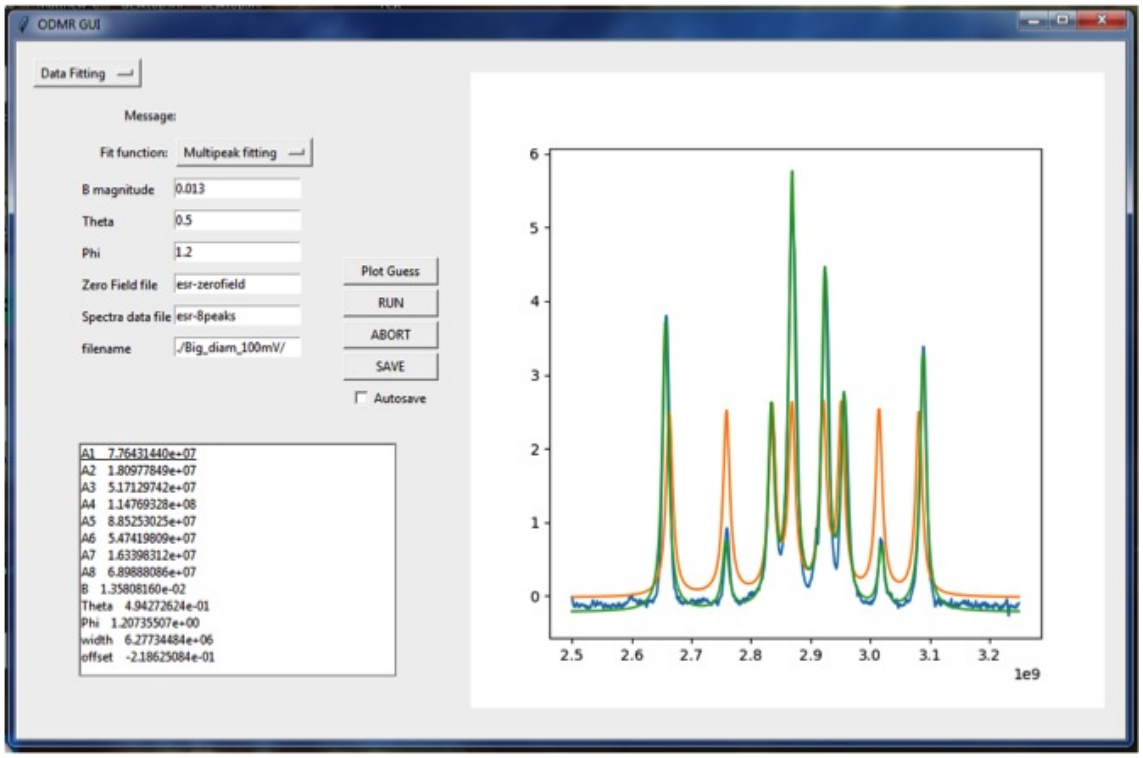

Figure 4: Data fitting window. Multipeak fitting shown.

4 Conclusion

The large amount of data in ODMR experiments means that the efficiency of the process depends largely on the use of computer control. By unloading repetitive tasks on to the computer, through the automation of calibration, data collection, and compilation, this efficiency is increased. Also, creating a GUI allowed for a more intuitive and quicker way to run programs; time saving features, such as an abort button, and easily editable parameters, only added to these strengths. The main room for improvement is in the labeling and layout of the GUI. Labels such as "counts" might be unclear at first glance and graphs are missing units and labels. The layout was done in such a way that when adding new labels and objects to the window, objects would have to be manually rearranged to accommodate them. The phase calibration could also be fully automated. Overall, this still is much better off than having separate scripts for each function. To improve the accuracy of the spectroscopy, a pulsed ODMR method will be employed. Instead of the continuous shining of the green laser, it will be pulsed after a quick pulse of a microwave, giving a much better resolution. The next step will be to code a GUI that is able to program these pulse sequences with the help of a specialized computer chip.

5 References

[1] M.W. Doherty, N.B. Manson, P. Delaney, F. Jelezko, J. Wrachtrup, and L.C. Hollenberg, Physics Reports **528**, 1 (2013).
[2] D.R. Glenn, D.B. Bucher, J. Lee, M.D. Lukin, H. Park, and R.L. Walsworth, Nature **555**, 351 (2018).