Slifer Lab: NMR Visualizer
Anderson, T.
October 26, 2020

Contents

Introduction

This document will cover basic user operation of the VNA Visualizer toolchain, the GUI that comes with the Visualizer's toolchain, and the technical breakdown of each file. The GUI was designed so that the user would never need to interface with the raw code. However, given the toolchain's feature-bloat, and semi object-oriented structure, code stability may become an issue, depending on the programmer.

1 GUI: gui.py

Is a tkinter widget-based graphical user interface that connects the user with the toolchain.

1.1 User's Manual

The following packages are required for the GUI's execution, and toolchain usage. I would advise using a virtual environment, to keep your root python directory clean.

- datetime
- glob*
- gc*
- matplotlib
- numpy
- os *
- pandas
- gc *
- tkinter *
- statistics *
- shutil *
- scipy
- math *
- tkinter
- traceback *

The libraries with (*) should already be included with all but the most bare-bones python distributions.
When you execute the gui.py the following window will appear:

Figure 1: This is the beginning page in which the user is able to select any of the tools embedded in the tool-suite.

1.2 DAQ Extractor

Proceed by clicking the "DAQ Extractor" button on the NMR tool-suite splash screen. Here, is where the signal extraction process begins. The DAQ extractor takes a DAQ csv, with all of the telemetry, NMR sweeps, and produces a .ta1 file *for each row in the DAQ csv*. There is a 1:1 correspondence between the DAQ rows, and files the DAQ Extractor creates.

With this in mind, the DAQ Extractor has the capability to create directories and categorize each created .ta1 file into a particular directory. It does this automatically to avoid polluting the local directory with tens of thousands of files. The organizing process is controlled by the "NMR Status" Column within the DAQ csv; and a soft coded value within the python script itself.

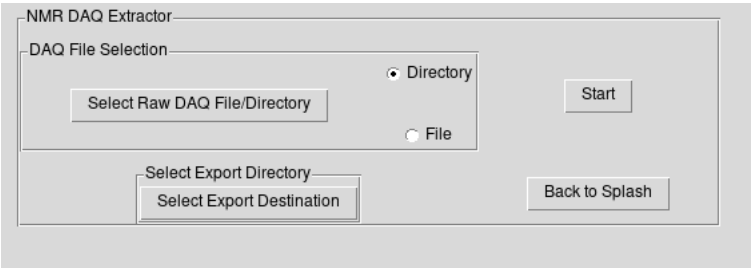

Figure 2: Caption

To operate this portion of the GUI, you must first select to process a single DAQ csv, or a directory of DAQ csvs. You can then select the Export directory, but the default directory for that is the local directory in which the program is executed. Below is an example of parsing data from the 2020 September experimental phase.

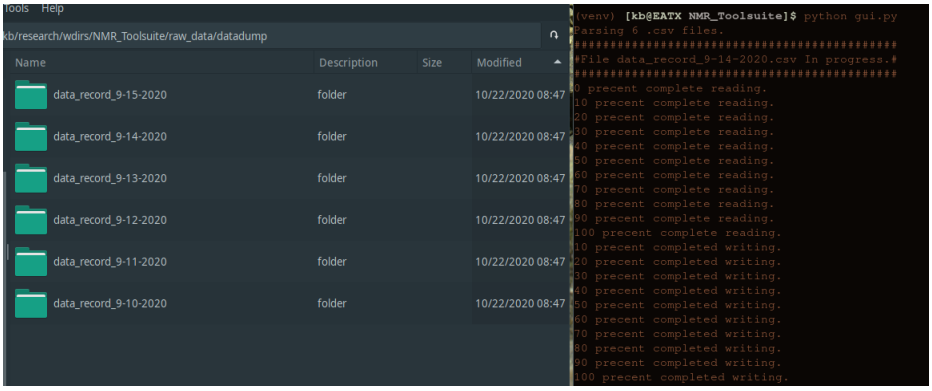

Figure 3: Caption

1.3 Sweep Sorter

Proceed by clicking the "Sweep Sorter" button on the NMR tool-suite splash screen. Here, is where you can further organize the enormous number of files into sub-directories for multi-sweep averaging, and time organization.

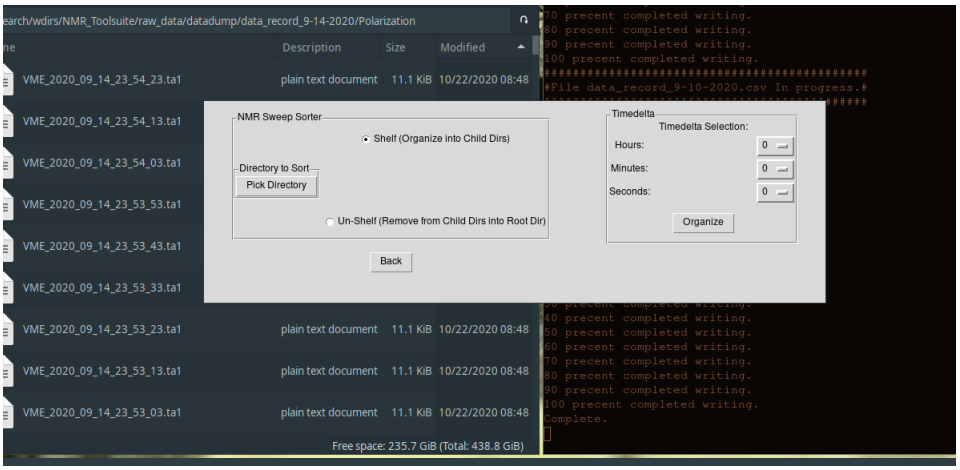

Figure 4: Caption

On the left hand side of the GUI the user is able to select a directory to organize. They can either "Shelf" or "Unshelf" the data which this program organizes for the user. On the right hand side of the GUI, the

1.4 Directory Averager

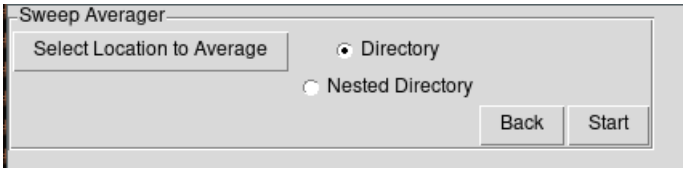

Figure 5: Caption

1.5 Signal Analysis

Proceed by clicking the "NMR Signal Extractor" button on the NMR tool-suite splash screen Here, the user is able to select the file type of the polarized target group's NMR recording system. Currently, the DAQ output from the VME (using files generated with VME_ Extractor) and the VNA. To begin, I will click the "Select Baseline" button, and select an adequate baseline file.

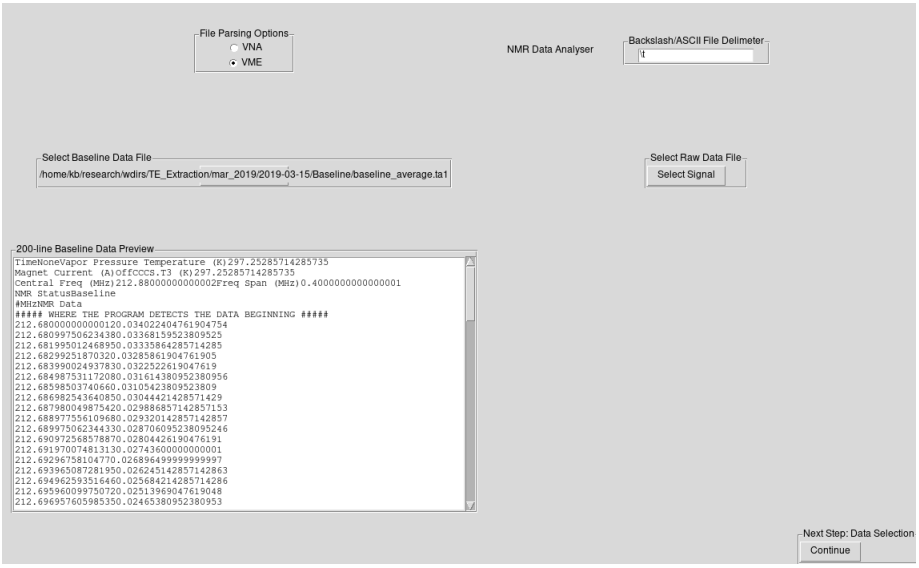

Figure 6: Once the file-dialogue has successfully selected a valid file, a 200 line file preview will appear.

And likewise for the signal file:

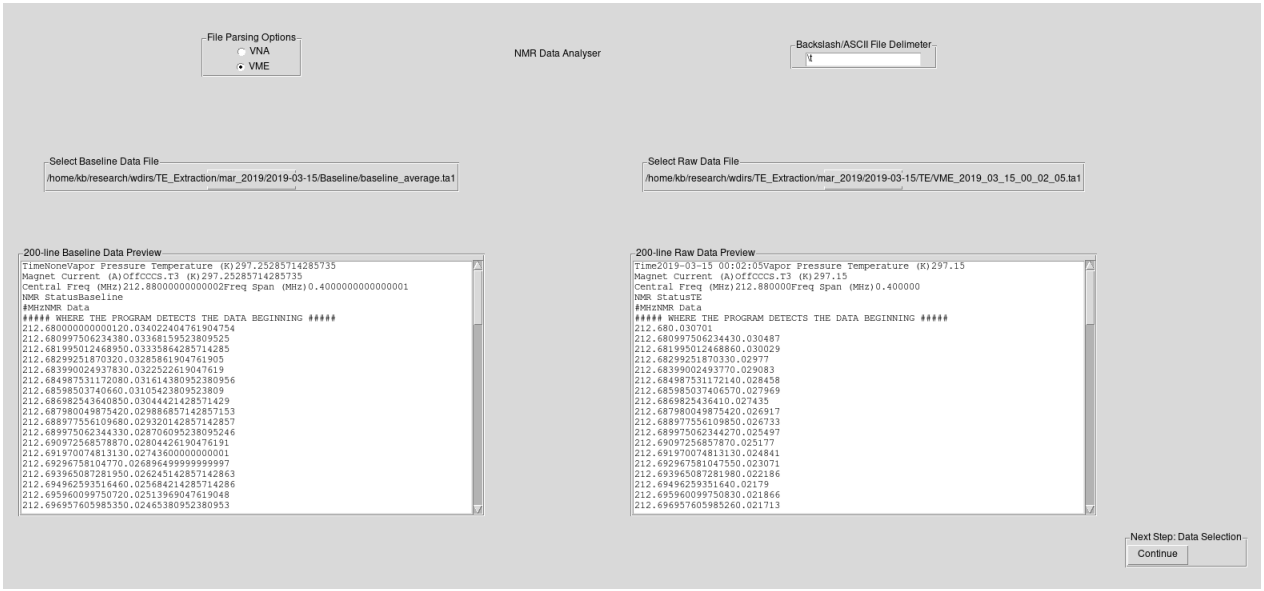

Figure 7: Caption

When these files have been selected, press the continue button.

1.5.1 Data Selection and Binning

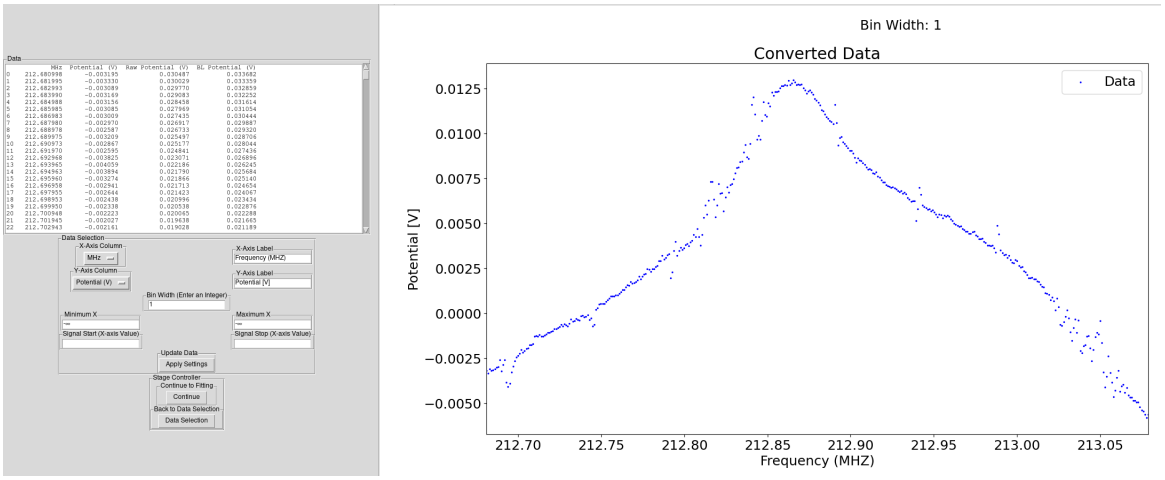

Figure 8: Data Selector

Figure 8 depicts a general impression of the data. For the VME's case, the axes are pre-labeled, and can be re-labeled if desired. Multiple channels of data are individually available for the VNA and VME. Unfortunately there is no functionality to plot multiple channels of data (on the y-axis) simultaneously. In the following table are the plot-able data-streams for each VNA*/VME measuring system. If selecting axis other than the default, be sure to update the graph by clicking the "Apply Settings" under the "Update Data" frame before varrying other settings, it is easy to think you are applying settings to one dataset, while actually making an adjustment on something historical!

Plot-able Datastreams (X/Y)

VME	Mhz	Potential (V)	Raw Potential (V)	BL Potential (V)			
VNA	Mhz	Z_re	Z_im	Raw Re(z)	Raw Im(Z)	BL Re(Z)	BL Im(Z)

When the axes are established, the user has the choice of selecting a bin width, which by default is 1 — showing every data point within the set. The bin width n determines how many raw data-points are being averaged to a single point on the plot.

Once the binning has been selected, and the graph has been updated with the "Apply Settings" button, the x-axis can be narrowed. By default, the plot will show all of the data, in ASCII this is represented by the infinity sign "$\pm\ \infty$" under the "Minimum X" and "Maximum X". Changing this view window size **will truncate data outside of the viewing range**.

When the range of the x-axis has been determined, the user can select a region of interest (pictured: Signal Start/Signal Stop), by sandwiching their region between the boundaries of the region. In order to do this the user must identify where their region starts, and ends according to the x-axis, then fill in the "Signal Start (X-axis Value)" and "Signal End (X-axis Value)" entries accordingly. Press "Apply Settings" again to view the user selected regions.

*The VNA has some intermediate mathematical conversions that can be recovered by editing a line of NMR _ Analyzer.py under function: *get _ z()*

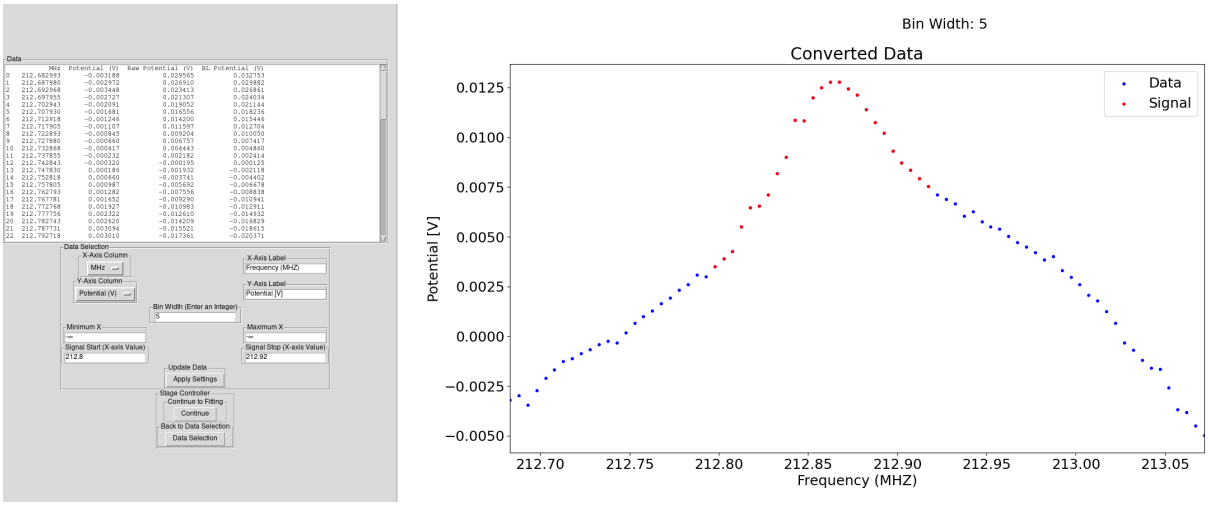

Figure 9: Caption

When the signal region has been highlighted, press "Continue" to advance to the fitting portion of the GUI.

1.5.2 Dataset Fitting

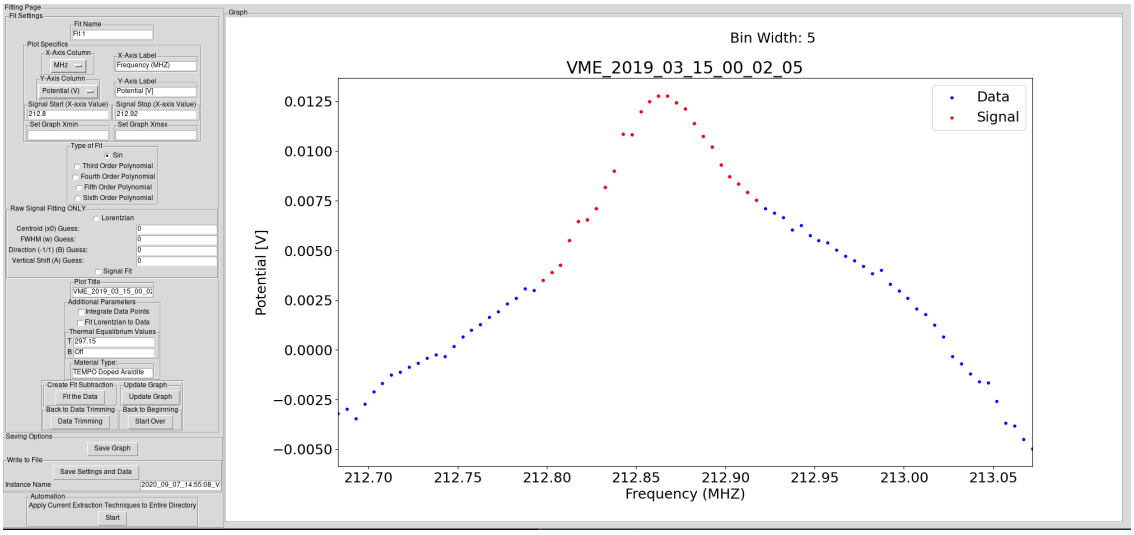

Figure 10: Caption

Here, the user is able to update the axes, re-highlight the signal, set a limit on what part of the data-set is visible, fit and subtract the background data around the shaded red region, fit the raw signal, update the graph title, display the Riemann integration of the data-set, fit a lorentzian curve to the data-set, update thermal equilibrium parameters (TE), track the material type, go back to a previous GUI window, completely restart the process, save the graph as displayed, save the data involved in generating the graph, and finally apply current signal-extraction techniques to other files in the raw-signal directory.

Like the previous window of the GUI, every time a modification is made to the settings page on the left, the graph must be updated to accurately reflect the changes applied, please click the "Update Graph" button after setting changes have been made.

The plot specifics widget is very similar to the Data Selection and Binning panel. The x and y axes/labels may be changed, the signal region and plotting window can be narrowed, or widened, and intermediate datasets (i.e. baseline, raw signal... etc) can be selected. There is no binning on this GUI pane. Any change of these settings will not be displayed until the graph has been updated.

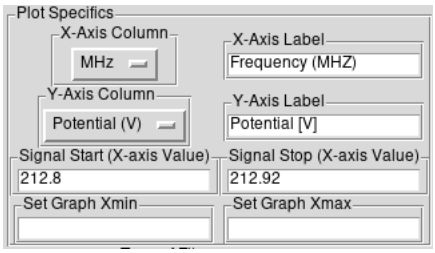

Figure 11: Caption

The user is able to then choose from a variety of functions to fit the data about the signal[†]. The user should select which type of fit they think best resembles the background data, and click the "Fit Data"

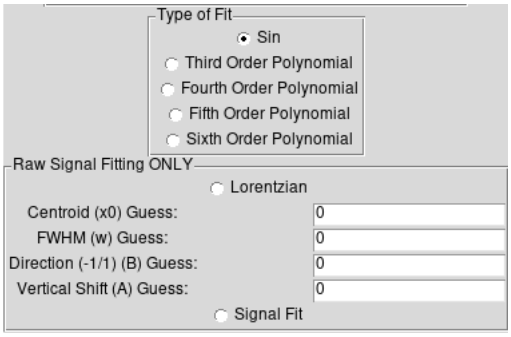

Figure 12: Caption

Upon a sucessful fit, the user will be presented with the fit imposed over the data:

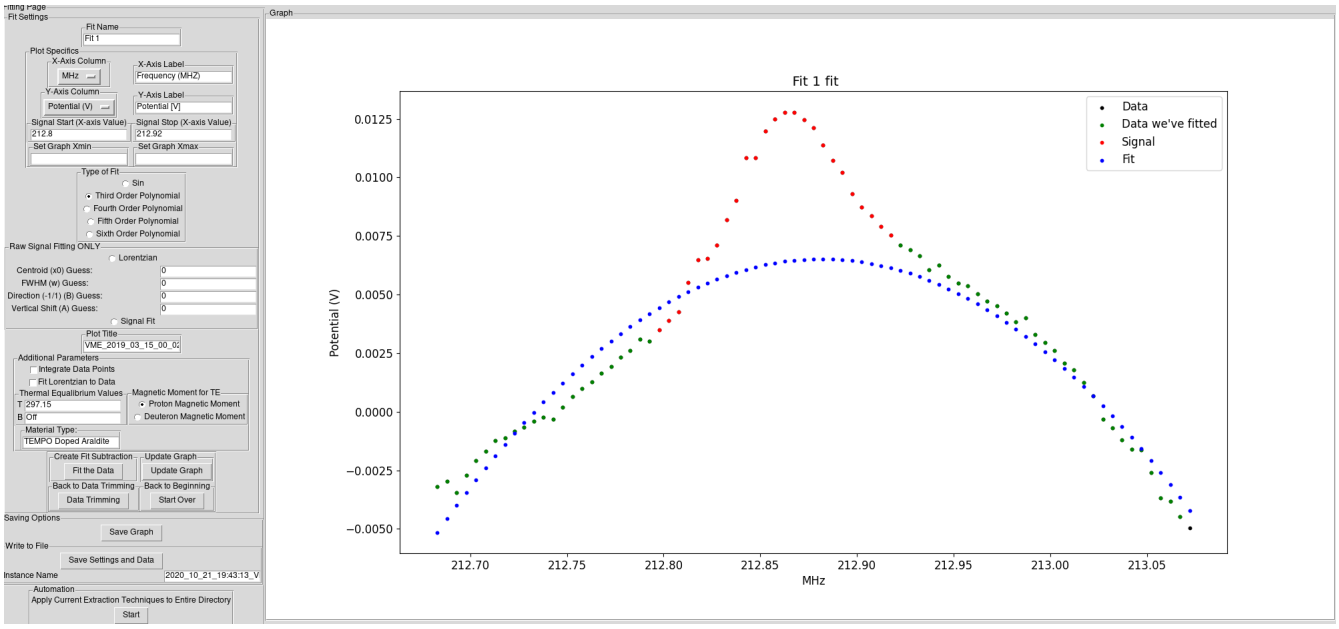

Figure 13: Note the dotted blue lines.

Upon a failed fit, the user will be presented with the signal graph with color-inverted data-points.

[†]including raw signal fitting (Consult Ellie's NMR/Proton Model) **which has not been successfully implemented yet into the gui due to least-squares fitting issues**

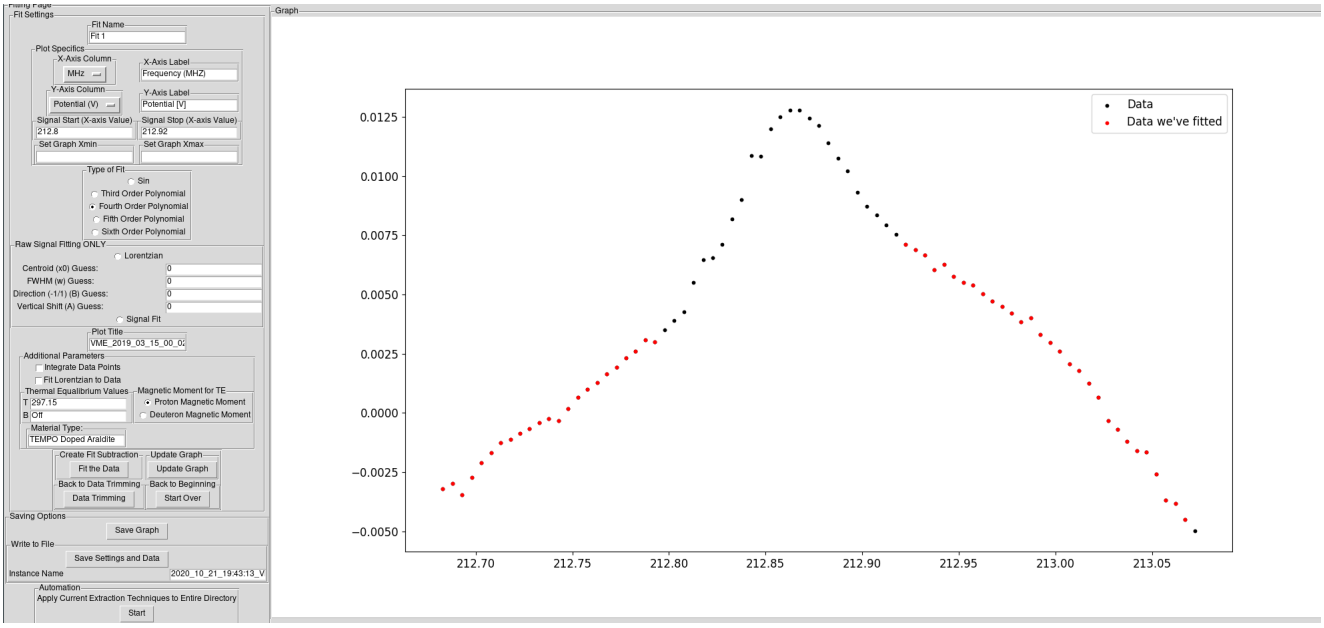

Figure 14:

To select the fit-subtracted dataset, change the y-axis column pull-down menu, to the fit-subtraction menu.

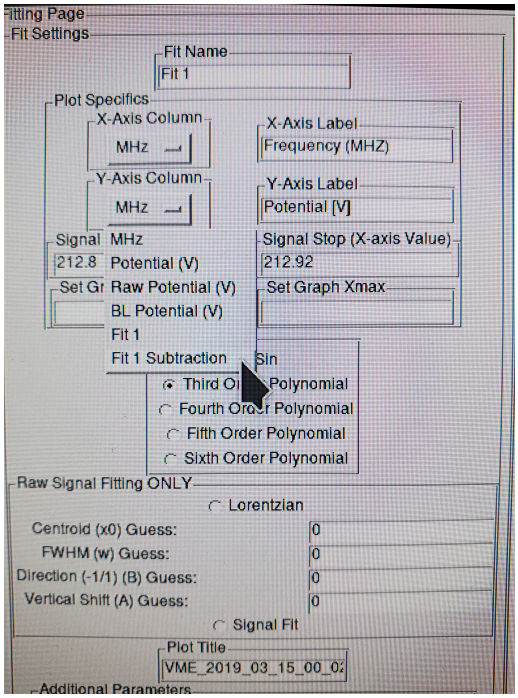

Figure 15: Appologies for the moiré. Remember to update the graph after selecting a new y-axis column.

In the following image, several parameters can be adjusted, this includes the plot title, numerical integration between the signal region (displayed as a shaded region below the signal selection), a lorentzian curve fit that captures the proton NMR, and lastly the TE Equation can have its reported temperatures and magnetic field manually adjusted. In the generated global analysis file, original values are preserved, but these user-inputs are stored as well.

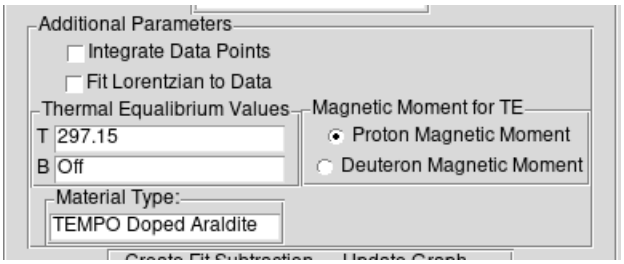

Figure 16: Caption

And at last, we arrive at the control-area of the fitting frame.

The grid of 4 buttons control the current frame and enable restarting the NMR signal extracting process. I'd like to guide your attention to the next 3 buttons, and instance name

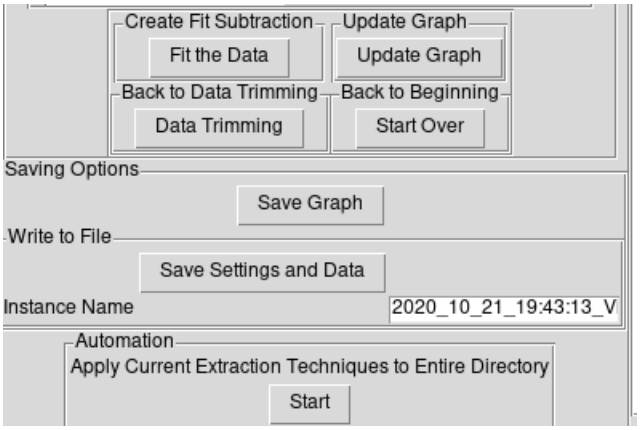

Figure 17: Caption

The "Save Graph" button will update the graph to reflect the current settings, then it will save the graph in local directory (where the program is being executed). The "Save Settings and Data" will save the data stored in memory pertaining to the current sweep being displayed.

1.5.3 Automation

The last and final part of the user-manual to this GUI is with automation. All you need to do to automate the signal extraction process is to press the automate-start button. The program will apply all of the settings which you have selected for this current session, and apply it to all of the other sweep signals within the directory from where you selected the current signal.

2 Toolchain: NMR_ Visualizer.py

Here's the technical documentation

2.1 nearest

```
nearest(test_val, iterable):
```

This function returns the nearest value to *test_val* in the iterable data structure *iterable*. This is used to determine the closest value to a user's selection of signal-region highlighting, and sometimes used to locate data in a parrell data structure that wasn't perfectly indexed.

2.2 add entry

`add_entry(*rowvals, **kwargs):`

Adds an entry to the global_analysis.csv that is produced with analysis using the GUI and some special uses of the toolchain. The function adds a line to .csv if it exists. If the csv does not exist, an empty one will be created then the entry will be added to the new file. Two minor sub-functions are folded into the *add_entry* function:

gen_persistence(path,columns)
Creates CSV file at *path* with columns *columns*

get_persistence(headers,fname)
Reads the global_analysis.csv. If that file does not exist, it will make one. The function returns either the successfully read data frame, or an empty one ready to populate.

***rowvals**

Expects a list of values to zip with "headers" kwarg passed on function call. Rowvals is mandatory.

****kwargs**

The *headers* keyword argument is used in the **add_entry** name space is particular to where it is called from. **add entry** requires the user to ask them self: "What do you want to save?" and you must pass the row-values and headers accordingly

2.3 return persistence df

`return_persistence_df(path):`

Reads and returns data frame object at *path*. This is used to read the global_analysis file

2.3.1 path

Path either absolute, or path relative to current working directory.

2.4 gui baseline file preview

`gui_bl_file_preview(filename, delimeter):`

Looks at the first 200 lines of the baseline file and delivers a small preview for the user to view during file selection.

2.4.1 filename

Path either absolute, or path relative to current working directory.

2.4.2 delimeter

Delimeter for the file to be read, generally '
t' for this application, but 'is here if you need it'.

2.5 gui raw sig file preview

`gui_rawsig_file_preview(rawsigfilename, delimeter, vnaVmeType)`

Looks at the first 200 lines of the raw signal file and delivers a small preview for the user to view during file selection.

2.5.1 rawsigfilename

Path to the raw signal file. Either absolute, or path relative to current working directory.

2.5.2 delimeter

QoL. Generally the delimeter is '
t', but it can be easily changed to accomodate the file format.

2.5.3 vnaVmeType

Used to switch between file interpreters.

2.6 gui file fetcher

```
gui_file_fetcher(RAWSIG_Path, Baseline_Path, vnavmetype, impression=False, binning=1, blskiplines=4,
    rawsigskiplines=4):
```

Fetches appropriate data from *RAWSIG_Path*, *Baseline_Path* with the appropriate file interpreter for the VNA/VME types (.s1p, .ta1)

2.6.1 RAWSIG_Path

An absolute or relative path to the raw signal file

2.6.2 Baseline_Path

An absolute or relative path to the baseline signal file

2.6.3 vnavmetype

A string either 'VNA' or 'VME' that determines which file interpreter to use when reading signal files.

impression

A boolean keyword argument that can be toggled to save a graph/impression of the S11 parameters and Re/Im of Z for the VNA data.

binning

Bin-width in datapoints.

blskiplines

Number of lines to skip before reading the baseline data like a seperated-value

rawsigskiplines

Number of lines to skip before reading the raw signal data like a seperated-value

2.7 VME frame aggregator

```
vme_frames(RAWSIG_Path, Baseline_Path, binning=1, blskiplines=4, rawsigskiplines=4)::
```

Collects and parses data from a .ta1 file, getting both the raw signal dataframe and baseline dataframes. The data can be re-binned with a bin-width of *binning*. No impression is available in this function.

2.7.1 RAWSIG_Path

An absolute or relative path to the raw signal file

2.7.2 Baseline_Path

An absolute or relative path to the baseline signal file

binning

Binwidth in datapoints

blskiplines

Number of lines to skip before reading the baseline data like a seperated-value

rawsigskiplines

Number of lines to skip before reading the raw signal data like a seperated-value

2.8 VNA frame aggregator

```
vna_frames(RAWSIG_Path, Baseline_Path, impression=False,title="", z_im=False, binning=1, blskiplines=4,
    rawsigskiplines=4):
```

Collects and parses VNA data from a .s1p file, getting both the raw signal dataframe and baseline dataframes. The data can be re-binned with a bin-width of *binning*. An impression (plot) of the real and imaginary parts of S11 and z (impedance) can be saved to the current working directory by toggling *impression*.

2.8.1 RAWSIG_Path

An absolute or relative path to the raw signal file

2.8.2 Baseline_Path

An absolute or relative path to the baseline signal file

2.8.3 vnavmetype

A string either 'VNA' or 'VME' that determines which file interpreter to use when reading signal files.

impression

A boolean keyword argument that can be toggled to save a graph/impression of the S11 parameters and Re/Im of Z for the VNA data.

binning

Bin-width in datapoints.

blskiplines

Number of lines to skip before reading the baseline data like a seperated-value

rawsigskiplines

Number of lines to skip before reading the raw signal data like a seperated-value

2.9 VME File parser

```
vme_file_parser(filename, skiplines):
```

Breaks down a VME (.ta1) file (at *filename*) line-by-line (skipping *skiplines* lines) and stores sweep information into a pandas dataframe, returning the dataframe to the caller.

2.9.1 filename

Absolute or relative path to file.

skiplines

Type int keyword argument that tells the file parser to skip the first *skiplines* lines.

2.10 VNA File parser

```
vna_file_parser(filename, skiplines=4)
```

Breaks down a VNA (.s1p) file (at *filename*) line-by-line (skipping *skiplines* lines) and stores sweep information into a pandas dataframe, returning the dataframe to the caller.

2.10.1 filename

Absolute or relative path to file.

skiplines

Type int keyword argument that tells the file parser to skip the first *skiplines* lines.

2.11 (Z) Impedance Converter

```
get_z(df):
```

Does a change of basis between the Re/Im S11 components, and the Re/Im parts of the impedence. Returns df with "MHz", "Z_re", and "Z_im" columns.

2.11.1 df

Requires a pandas dataframe with columns: "MHz", "Re(S11)", "Im(S11)" columns.

2.12 Curve Integrator

```
integrate_curve(start_index, end_index, df=None, x="MHz", y="Z_re"):
```

Returns the trapezoidal Riemann integration of the data points between *start_index*, *end_index* in *df*. Will implement better quadrature techniques soon (sum of two Riemann schemes).

2.13 Ellie's Lorentizan Raw-Sig Fitting

```
absorbtion_dispersion_ellie(f, f0, w, kmax, theta):
```

See NMR Model in Box for mathematical explanation and justification.

2.14 General Fitting Function

```
gff(df, start, finish, fitname, **kwargs):
```

A general fitting function for this NMR toolchain. Requires a pandas dataframe *df*, a start *start* and end *end* index, as well as a hard-coded fitname *fitname* to fit to column y in *df*. This function operates by array index, and not by numerical values. Consult the **nearest** function to get array indexes. Fits the data around the signal, where the signal is selected by index placement within the column of the dataframe. Within this function, the signal is generally within: ydata[s:f].

2.14.1 df

Pandas Dataframe. A non-empty pandas dataframe.

2.14.2 start

Int. The start index in *df* of the region that should be ignored during the fitting.

2.14.3 finish

Int. The end index in *df* of the region that should be ignored during the fitting.

2.14.4 fitname

String. The human readable function name used to label types of fits. Accessable in the command line.

2.14.5 **kwargs

```
y="Z_re", x="MHz", preview=False, plttitle='', fit_sans_signal=False, function=[], fit_bounds=[], sf=None,
    ff=None, savefit=False, filename="UNNAMED_GRAPH", binning=1, xmin=None, xmax=None, gui=False,
    automated=False, p0=None, bounds=[[-numpy.inf, -numpy.inf, -numpy.inf, -numpy.inf
    ],[numpy.inf,numpy.inf,numpy.inf,numpy.inf]]
```

y

Str. Name of the column in *df* containing dependent values to model with respect to *x*.

x

Str. Name of the column in *df* containing independent values to map onto *y*.

preview

Boolean Flag. Pyplot.show's the figure before continuing. Useful in the command line.

plttitle

String. User specifiable title to the proof-plot this generates

fit_sans_signal

Boolean Flag. Fits *y* data ignoring all data in region: ydata[start:finish]

function

List of Strings. The hard-coded function name to fit with scipy optimize curvefit. Must be parallell to each tupple in $fit_bounds, or must be len 1 to apply single function to all x-ranges in fit_bounds$ **fit_bounds***An array of tupples [(x1s, x1f),...,(xns,xnf)*

```
    #     (sf index)  (start index)    (finish index)  (ff index)
    #             |            |    --     |            |
    #             |            |    --     |            |
    #  not fitting|  fitting   |  ------   |  fitting   | not fitting
    #             V            V --    --- V            V
    # ------------|++++++++++++|-       ---|++++++++++++|-------------
    #  sf region  |            |   SIGNAL  |            |  ff region
```

savefit

Boolean. Saves the proofing plot produced by *gff*.

filename

String. Custom filename for the plot written to disk by *savefit*.

binning

Int. Helps scale individual points for the scatter plots.

xmin

Number. Matplotlib pyplot's graph window minimum x

xmax

Number. Matplotlib pyplot's graph window maximum x

gui

Boolean flag. Used to return figure to display in tkinter GUI.

automated

Boolean used to help user troubleshoot fitting routine when GUI automates sweep-extraction. Prints warning when y is not found in df. This can happen due to a bad y-fit on a particular sweep.

p0

Array of numbers. Initial points used to find absorption/dispersion parts of the lorentzian-shaped raw signal. Used only during fitting of absorption/dispersion lorentzians.

bounds

2-D array of numbers. Fit parameter bounds (coersion) passable in the command line that helps encourage least-squares fitting routine to model the data.

2.15 General Graphing Function

```
ggf(master, s, f, **kwargs):
```

General graphing function for the

2.15.1 df

Pandas Dataframe. A non-empty pandas dataframe.

2.15.2 s

Int. The start index in *df* of the signal to plot.

2.15.3 f

Int. The end index in *df* of the signal to plot.

2.16 **kwargs

```
xmin=None, xmax=None, plttitle="",x="MHz", y="Z_re", xlabel="Frequency (MHz)", ylabel="Re(Z) Impedence
    [Ohms]", filename="", fit_marking=False, fit_bounds=[], noshow=True,binning=1,redsig=False,
    integrate=False, thermal_equalibrium_value=False, b=None, T=None, fitlorentzian=False, edata=-1,
    gui=False, fitlorentziancenter_bounds=None, automated=False, temu=up
```

x

Str. Dataframe column name containing the x data.

y

Str. Dataframe column name containing the y data.

filename

Str. The name of the file to save the graph to.

fit_marking

Boolean flag. Used for marking the data during fit subtraction.

fit_bounds

Array of x-tupples [(x1s,x1f),...,(xns,xnf)]. Used in tandem with *fit_marking* to highlight ranges

noshow

Boolean Flag. Used to turn off figure preview before saving graph.

binning

Int. A number used to size the points on the scatter plots.

redsig

Boolean flag. Highlight the user-selected signal region red.

integrate

Boolean flag. Used to shade the area under the user-selected signal region to provide a visual representation of the area underneath the graph.

thermal_equilibrium_value

Boolean flag. Calculates the Thermal Equilibrium value from the TE equation and places its value as text on the graph. Also finds calibration constant for plotted data. Only configured for the proton as of June 18, 2020.

b

Number. B, or the magnetic field used to calculate the TE equation.

T

Number. T, or the Temperature used to calculate the TE Equation.

fitlorentzian

Boolean flag. Fit a lorentzian to the data, and plot the fit, useful for fitting the proton NMR signal.

gui

Boolean flag. Used to return function to tkinter handler for GUI display.

clearfigs

Boolean flag. Used to free-up memory

fitlorentziancenter_bounds

Tuple, Numbers. Used to constrain center of the lorentzian fit within the user-selected signal region.

automated

Boolean flag. used to help user troubleshoot fitting routine when GUI automates sweep-extraction. Prints warning when y is not found in df. This can happen due to a bad y-fit on a particular sweep.

temu

Number. The Mu used in the TE equation.

edata

Pandas Dataframe. Use-able in the command line. A dataframe that contains data to plot over other data currently on the figure.

xmin

Matplotlib pyplot's graph window minimum x

xmax

Matplotlib pyplot's graph window maximum x

xlabel

Matplotlib pyplot's xlabel

ylabel

Matplotlib pyplot's ylabel

plttitle

Matplotlib pyplot's plot title

3 Clerical Tools

3.1 DAQ Muncher

Partitions a DAQ csv file line-by-line into separate (.ta1) files which analysis is done on.

3.2 Directory Sorter

Organizes Files for to be averaged or isolated from one-another based on timestep.

3.3 Sweep Averager

Averages DAQ Munched files within a single directory, or in a 2d file structure. (A directory with many 1-layer sub-directories)

3.4 Global Analysis

Takes the results from the analysis, and pulls together other useful system data-streams into a single plot.