Slifer Lab: NMR Visualizer
Anderson, T.
February 15, 2021

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 ASCII / Terminal GUI

With the release of python 3.9.1 the graphical interface that I wrote to wrap around this tool suite broke. In order to overcome that, I wrote a second front-end that the end user simply runs from the terminal.

1.1 User's Manual

The following is a how-to guide, and provides the user with an overall understanding of the capabilities and limitations of the ASCII terminal interface with the NMR toolsuite.

1.1.1 Settings

The settings file titled: *variablenames.py* is a python file that is imported in every module of the toolsuite. Here, the variables within *variablenames* are readily accessible. The file is well organized and commented to delineate what variables go with which module of the toolsuite. The following unnumbered sections will itemize the settings and their correspondence with the toolsuite.

DaqExtractor and Sweep Averager Settings

Variable	Default	Setting
dmsa_time_colname	"Time"	Name of the Timecolumn
dmsa_primary_thermometer_colname	"CCX.T3 (K)"	Name of the thermistor you trust most
dmsa_secondary_thermometer_colname	"Vapor Pressure Temperature (K)"	Name of the thermistor you trust second most
dmsa_magnet_psu_amperage_colname	"Magnet Current (A)"	Name of the magnet powersupply current
dmsa_sweep_centroid_colname	"Central Freq (MHz)"	name of the central sweep frequency
dmsa_sweep_width_colname	"Freq Span (MHz)"	name of the frequency span column
dmsa_system_status_colname	"NMR Status"	terminal column of data in dataframe – the rest is assumed to be nmr data
dmsa_system_status_nulls	['—']	entries in nmr data that aren't interesting
dmsa_system_null_status_directory	'null_status'	where uninteresting sweep data goes
dmsa_terminal_colname	'NMR Data'	name of the state of the experiment (te/ enhanced/baseline)

The daq extractor and sweep averager settings are **the most important** settings to know while analyzing NMR sweeps. Upon receiving a raw csv from the DAQ, the DaqExtractor "sets the tone" so to speak of the possible analysis the rest of the toolsuite can do in the future. On the execution of the DAQ Extractor, (which produces the .ta1 files) the extractor reads the above settings from *variablenames.py*, writing the effects of these settings to each .ta1 file. Once the .ta1 files are written, it's difficult to switch out their primary and secondary thermistor values with those from the raw DAQ file. It's easier and most likely faster to complete a signal analysis, and then use the timestamps from the global analysis files as "keys" to grab the alternate thermistors the user wants from the DAQ file **This sounds like a feature I'll implement on the way out the door**.

Regardless, the signal analysis process begins with extracting the sweeps from the DAQ Extractor, and these settings determine the down-stream analysis settings, soz choose them carefully.

NMR Analyzer Settings

Variable	Default	Setting
na_primary_thermistor_name	dmsa_primary_thermometer_colname	Specifies the most trusted thermistor
na_secondary_thermistor_name	dmsa_secondary_thermometer_colname	Specifies the second most trusted thermistor
na_vme_yaxis_default	"Potential (V)"	Specifies the default y-axis
na_vme_xaxis_default	"MHz"	Specifies the default x-axis
na_global_analysis_headers	["name", "material", "time", "dtype", "blpath", "rawpath", "xmin","xmax", "sigstart", "sigfinish", "blskiplines", 'rawsigskiplines', "B", "T", dmsa_primary_thermometer_colname, dmsa_secondary_thermometer_colname, "TEvalue", "data_area", "ltzian_area", "data_cal_constant","ltzian_cal_constant", 'a', 'w', 'x0' ,"lorentzian chisquared (distribution)", "σ (Noise)", "σ (Error Bar)", "lorentzian relative-chisquared (error)", "Sweep Centroid", "Sweep Width", 'e_f0', 'e_w', 'e_kmax', 'e_theta']	Specifies the headers for the global analysis file

The NMR Analyzer settings are the settings for the back-end of the the signal analyzer. Although the user will only interface with this python file indirectly through the GUI, settings can be adjusted. The primary and secondary thermistors should be specified at the DAQ settings, unless the user is analyzing a dataset with different thermistors. The default vme x and y axis defaults have coresponding hard-coded VNA values as well. These will be updated in a future relase. Take care to not edit the *global_analysis_headers* as the header-order corresponds to other hard-coded variables within the analyzer's methods.

TKinter GUI settings

Variable	Default	Setting
gui_primary_thermistor_name	dmsa_primary_thermometer_colname	Specifies the most trusted thermistor
gui_secondary_thermistor_name	dmsa_secondary_thermometer_colname	Specifies the second most trusted thermistor

Depreciated since January 5, 2021, but shouldn't change, should tkinter magically begin working again.

ASCII GUI settings

Variable	Default	Setting
agui_primary_thermistor_name	dmsa_primary_thermometer_colname	See above
agui_secondary_thermistor_name	dmsa_secondary_thermometer_colname	See above
agui_allowable_file_extensions	['.ta1', '.s1p', '.csv']	file extensions the ascii gui is permitted to see
agui_vnavme_default	'vme'	default mode / file interpretation
agui_xname_default	na_vme_xaxis_default	Specifies the default y-axis
agui_yname_default	na_vme_yaxis_default	Specifies the default x-axis
agui_impression	False	Gives impression of the various components of the data.

The *primary* and *secondary* thermistors should reflect the current data being analyzed. The *allowable file extensions* are used to filter the files out of custom file navigator that the user will use to select files and directories. The *vnavme* default is the default file-interpreting mode that the entire toolsuite is geared towards, In a future release, the program will automatically switch file parsers based on the file extension. Similarly with the back-end's *vme x,y axis defaults* there are hard-coded VNA counterparts, but the variables here should have been pre-defined.

Global Interpreter settings

Variables	Default
gi_primary_thermistor	"CCX.T3 (K)"
gi_secondary_thermistor	"Vapor Pressure Temperature (K)"
gi_time	'time'
gi_lorentzianarea_results	"Lorentzian Area"
gi_scaled_polarization	"Scaled Polarization (%)"
gi_uncert_in_scaled_pol	"Uncert in Scaled polarization"
gi_integrated_data_area_results	"Integrated Data Area"
gi_ltz_area_results	"Lorentzian Area"
gi_bviaI_results	'B via I (T)'
gi_primary_thermistor_results	gi_primary_thermistor
gi_secondary_thermistor_results	gi_secondary_thermistor
gi_centroid_results	"sweep centroid"
gi_width_results	"sweep width"
gi_te_results	"TEvalue"
gi_teviax0_results	'TE via x0'
gi_tebest_results	"TE Best"
gi_te_uncert_results	"TE Uncert"
gi_rsq_results	"Reduced Relative Chi-Square"
gi_centroidlabel	'x0'
gi_lorentzianarea	"ltzian_area"
gi_dataarea	"data_area"
gi_relchisq	"lorentzian relative-chisquared (error)"
gi_centroid	"Sweep Centroid"
gi_width	"Sweep Width"
gi_TE	"TEvalue"

The primary and secondary thermistor settings reflect the user-selected thermistors present in the global analysis file that the global-interpreter is reading from. By default, CCX.T3 and the Vapor pressure are being used, but this should be changed every time the user wants to look at a separate thermistor data-sets. Variables with "results" in their name are generally non-sensitive to user-editing, but I would encourage the user to exercise caution, taking note

of what settings are currently deviating from the defaults, and returning them back to normal should the tool-suite suddenly cease working.

1.1.2 Extracting sweeps from the DAQ

Extracting sweeps from the DAQ requires the user to ask themselves first and foremost: What thermistors are most representative of the target temperature? In analyzing NMR sweeps in the context of polarized target work within the Slifer Lab, calculating a good Thermal Equalibrium calbiration constant is *contingent* on a stable temperature. A quick plot using the omni-view function from the Slifer-Cal Toolsuite can give us insight into what occurred during the cool-down. That toolsuite can be found here: here * With documentation located here †.

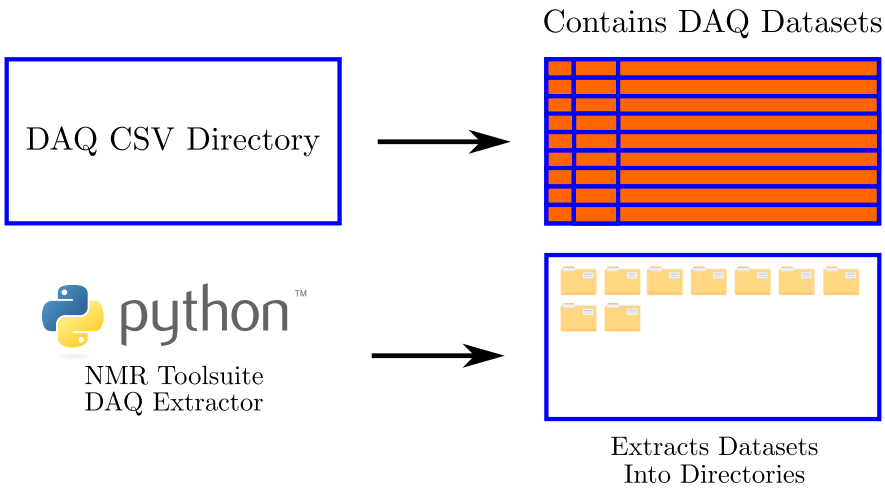


Figure 1: The DAQ extractor reads DAQ csv files then extracts the NMR sweeps, select thermistors, and magnet amperage writing that information in a custom file type that is categorized into special sub-directories. All the program needs to do this is the DAQ .csvs.

1. First, execute the *asciigui.py* file on the main branch of the github toolsuite:

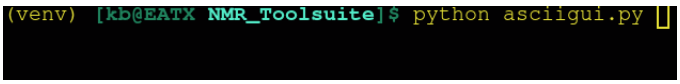


Figure 2: Simple virtual-environment using python to execute the asciigui program.

2. Then select the DAQ extractor option by entering the corresponding mode number, and pressing ENTER.

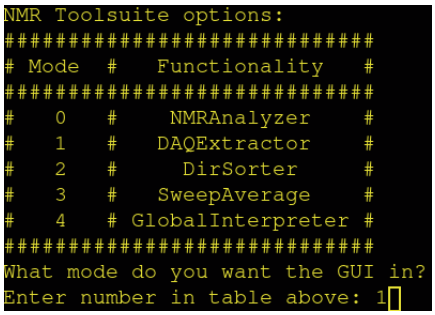


Figure 3: This is the first stage that the user is greeted by in the ASCII Gui, I will later refer to this as the "Splash Screen".

*https://github.com/Tristan-Anderson/Slifer_Lab_Thermometry
†https://unh.box.com/s/k21gr1ta3zg6wepciuaejcv7j0uom0b6

3. The DAQ extractor will ask for a directory of .csv's or single .csv to extract into .ta1 files. Enter the respective directory by navigating with 0-n(d/f) per the instructions printed in the terminal. 'f' stands for file, and 'd' for directory. Selecting a file will immmedately bring you to the next window, changing through directories will not. Type "ok" to signal to the program to select the current working directory as your choice.

```
@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
@        DAQ Extractor        @
@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
Extracts and organizes DAQ .csvs into .ta1 based on keywords found in variablenames.py and settin
entered here, by the user.


### Pick directory or file to unpack ###
######################################################
# LinNUM #            Files            # Directories #
######################################################
#    0    # .~lock.global_analysis.csv# #    .idea    #
#    1    #                            #  raw_data   #
#    2    #                            # graph_data  #
#    3    #                            # __pycache__ #
#    4    #                            #  datasets   #
#    5    #                            #    .git     #
#    6    #                            #   graphs    #
######################################################
Current working Directory: /home/kb/research/wdirs/NMR_Toolsuite
Enter choice in the format of: 'LineNum(f/d)
 ex: 1f
Enter Choice: 
```

Figure 4: Here the user selects a directory/single DAQ File to unpack. Select according to the instructions above.

4. Once the appropriate file(s) have been selected, *the user will then need to specify the location where to place the .ta1 files.*

```
### The current unpacking destination is  ###
@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
@        Select a DIRECTORY to place files.        @
@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
####################################################
# LinNUM #           Files           # Directories #
####################################################
#    0    # data_record_9-14-2020.csv #     old     #
#    1    # data_record_9-12-2020.csv #             #
#    2    # data_record_9-13-2020.csv #             #
#    3    # data_record_9-11-2020.csv #             #
#    4    # data_record_9-15-2020.csv #             #
#    5    # data_record_9-10-2020.csv #             #
####################################################
Current working Directory: /home/kb/research/wdirs/NMR_Toolsuite/raw_data/raw_data
Enter choice in the format of: 'LineNum(f/d)
 ex: 1f
Enter Choice: 
```

Figure 5: Per K.I.S.S. (keep it stupid simple), each DAQ file selected earlier will have a respective directory in the dump path that will contain the extracted .ta1 files.

5. Once the directory has been selected to place the extracted information from the DAQ, the program will then prompt the user with a series of options. Having already completed the first two, the next option would be to extract the data. Press 2, ENTER, then the program will be on it's way, using multiprocessing if supported.

```
Path accepted.
################################################################################
# option# #       name        #                  describers                    #
################################################################################
#     0     #  updateSelection  #       Select file, or directory to unpack      #
#     1     # updateDestination #    Select destination to place unpacked data    #
#     2     #    extractData    # Extract current selection into analyzeable files #
################################################################################
Enter option number:
```

Figure 6: Two options are present in case the user messed up during the 5 step process. To exit this loop, pop a ^ C or CTRL-C or BREAK, afterwords pressing ENTER. You will return to the toolsuite's splash screen.

1.1.3 Organizing Sweeps (Directory Sorter)

Now that you have extracted the NMR sweeps from the DAQ files, it is time to decide if you want to average the NMR sweeps. The first step in averaging sweeps is chronologically ordering them with the Directory Sorter. This ordering then allows the Sweep Averager covered in the next section to read each file in a single directory, and determine a directory's average.

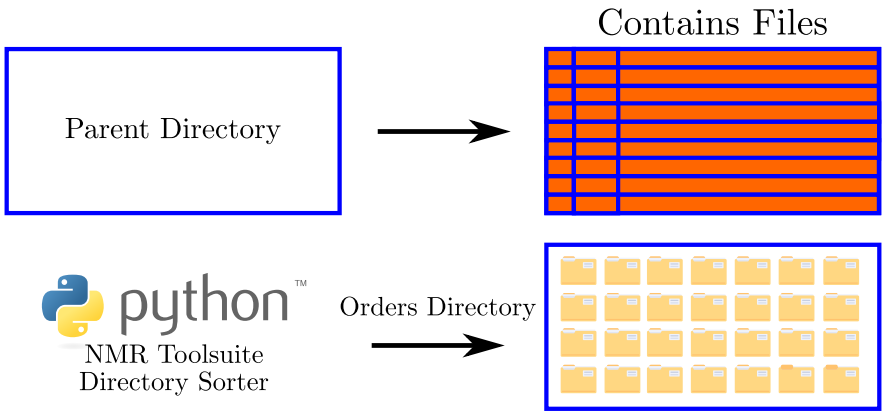


Figure 7: The directory sorter works by reading the timestamps of the .ta1 files. It catalogues these files by moving them into automatically created and named directories.

1. To use the Directory Sorter, select the Directory Sorter option from the NMR Splash Screen.

```
NMR Toolsuite options:
############################
# Mode  #   Functionality   #
############################
#   0   #    NMRAnalyzer    #
#   1   #   DAQExtractor    #
#   2   #     DirSorter     #
#   3   #   SweepAverage    #
#   4   # GlobalInterpreter #
############################
What mode do you want the GUI in?
Enter number in table above: 2
```

Figure 8: NMR Splash screen selecting the Directory Sorter option.

2. Next, the user must guide the program to a directory containing just .ta1 files.

```
@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
@        Directory Sorter        @
@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
Packs and unpacks directories by a width of time based on timestamps present in the .ta1 files


### Pick directory or file to unpack ###
################################
# LinNUM # Files # Directories #
################################
#   0    #       #    .git     #
#   1    #       #   rawdata   #
#   2    #       # __pycache__ #
################################
Current working Directory: /home/kb/research/wdir/NMR_Toolsuite
Enter choice in the format of: 'LineNum(f/d)
 ex: 1f
Enter Choice:
```

Figure 9: Initial screen of the Directory organizer.

3. Once that directory has been selected, the user can choose to shelf, or unshelf files present in the directory.

```
You selected /home/kb/research/wdir/NMR_Toolsuite/nice/data_record_9-10-2020/null_status/ which is a Directory
#########################################################################################################
# option# #    name     #                                describers                                      #
#########################################################################################################
#    0    #    shelf    #       Organize the current directory per the user selected time-stamp          #
#    1    #   unshelf   # Move files/folders in first child directories out to their parent directory    #
#    2    # settimestep #             Set the width of time to organize directories with.               #
#########################################################################################################
Enter option number:
```

Figure 10: **Shelfing** refers to the act of organizing files by their time-stamps, placing each file that falls within a user-defined time-interval into a single directory. **Unshelfing** Refers to the undoing of the *shelfing* process, which moves all of the child directory's contents, into their parent directory, deleting the empty child directories in the process.

4. After choosing the *shelf* option, enter the width of time that the program will shelf the data. Once the hours, minutes, and seconds have entered into the program, it will start sorting the dataset.

```
Enter option number: 0
Your choice was shelf returning...
Enter the number of seconds:
Please enter a number between 0 59 or press enter to skip: 15
Enter the number of minutes:
Please enter a number between 0 59 or press enter to skip: 3
Enter the number of hours:
Please enter a number between 0 23 or press enter to skip:
Sentinal recieved - ignoring and returning.
Shelving complete.
```

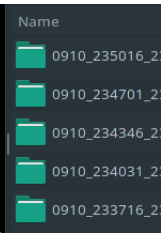


Figure 11: On the left, the user has entered 3 Minutes, 15 seconds as a time-width to organize the sweeps. On the right, a traditional file-explorer shows a portion of the automatically named directories. All of which have been populated with the corresponding .ta1 files.

5. To exit, or go back, enter a BREAK or ^C into the terminal, then Press ENTER.

1.1.4 Averaging Sweeps

The user may want to average NMR sweeps should they find that the NMR signal is not discernible from the noise-floor. This is done by first organizing the sweeps into directories that the user would like to average. **The Sweep Averager averages DIRECTORIES** it is up to the user to properly organize the directories, and ensure data-integrity and continuity during this process. The key with averaging sweeps, especially when dealing with a TE Calibration data set is to average the sweeps enough so the signal is discernible, but not so much as to reduce the data-set so that a statistical analysis is moot. The sweep averager has two modes: single, and nested. In single

mode, the averager reads all of the data in the current directory, and averages them producing a file in the current directory. In nested mode, the averager reads all of the data in the child directories of the working directory, averages that data, then writes each directory's average to a file in the parent directory. Hopefully the following figure may clarify this description.

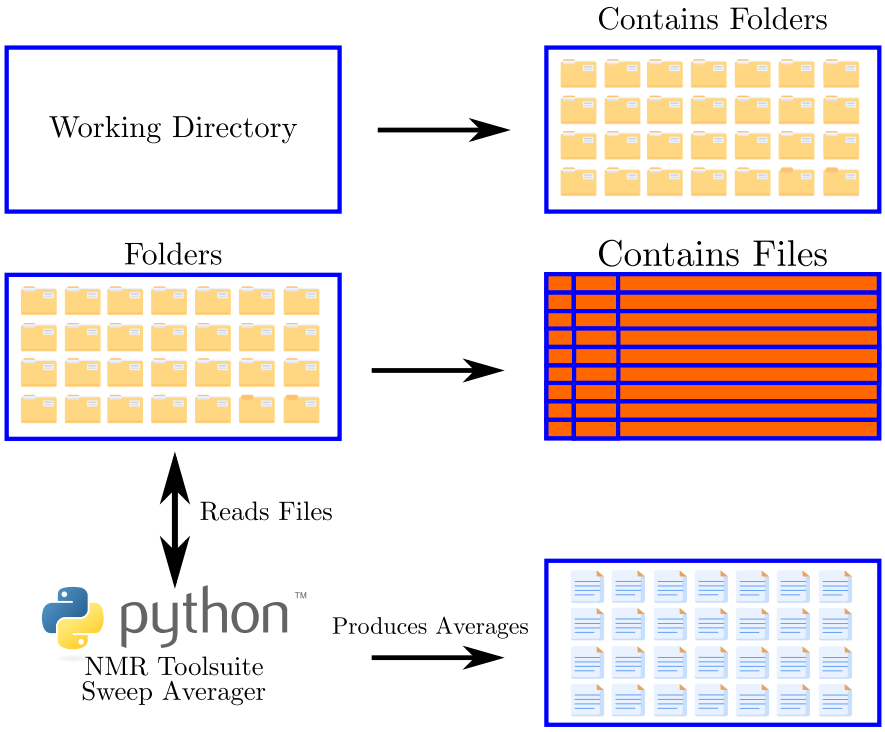


Figure 12: File icons courtesy of FlatIcon.

1. First, select SweepAverage option on the NMR Toolsuite splash screen. Then, choose the update location option.

```
#################################################################
# option# #      name      #           describers               #
#################################################################
#     0     # updatelocation # Select directory for averaging #
#     1     #     execute     #  Start the avergaing process   #
#################################################################
```

Figure 13: In a future release, the user can simply select 'execute.' The program will detect whether a path to a data-set directory has been entered into the program. If the program has not been given a path to run average the directories, it will ask the user for one.

2. Once the program has accepted the path, it will prompt the user to decide whether or not the path selected was a nested directory that contained child directories, or just a single directory making files therein to be averaged.

```
#############################################################################################
# option# #    name    #                    describers                                      #
#############################################################################################
#     0     #  nested   # Avarage sweeps in a nested directory, naming avg sweep after parent directory #
#     1     # directory #                  average single directory into a single file                  #
#############################################################################################
Enter option number:
```

Figure 14: Here the user tells the program how to average the current selection. There are two options, one is to average the files within a single selected directory, and the other is to average the child directories (most likely created by the sweep sorter).

3. The Sweep Averager will then average the sweeps, needing no further input from the user.

1.1.5 Signal Extraction

Extracting an NMR signal from the custom Versa Module Eurocard (VME) board, or Vector Network Analyzer (VNA) is a step-by-step process. Good signal extraction begins during the experiment. First, a baseline - or the natural q-meter circuit response is measured by the VME/VNA board. Then, the target material is placed within the q-meter coils, and it polarizes (Per Chapter 2, Section 2 here ‡ For more information on the experimental setup, see Chapter 3, Section 2) This polarization is detected via a frequency-dependent NMR Absorption spectra, and measured for the VME as a Voltage, or for the VNA as the real and imaginary components of the S11 component in q-meter circuit. The second step to NMR signal extraction is to remove the q-meter's natural response from the NMR absorption spectra. For a more ideological discussion on the signal extraction process see Chapter 4 here‡.

Now, with that data recorded, this toolsuite is prepared to analyze each spectra. In theory, the difference between these two spectra should result in an NMR spectra that is just a result of nuclear spins. The NMR Signal Extraction tool-suite just needs the path of the Baseline, and the path of the Raw NMR Signal.

1. Begin by selecting the NMR Analyzer option on the splash screen of the NMR toolsuite.

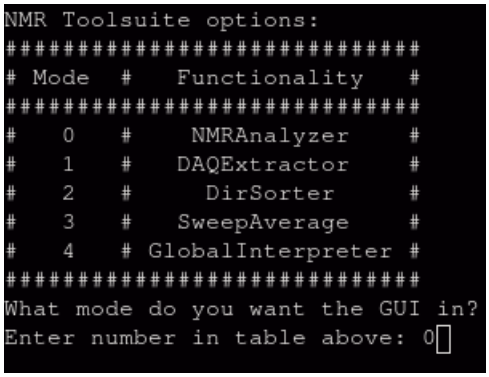


Figure 15: Select the NMR Analyzer functionality of the toolsuite by pressing '0' followed by ENTER.

2. Then, navigate to the path of the **baseline** with the ASCIIGUI's file-explorer.

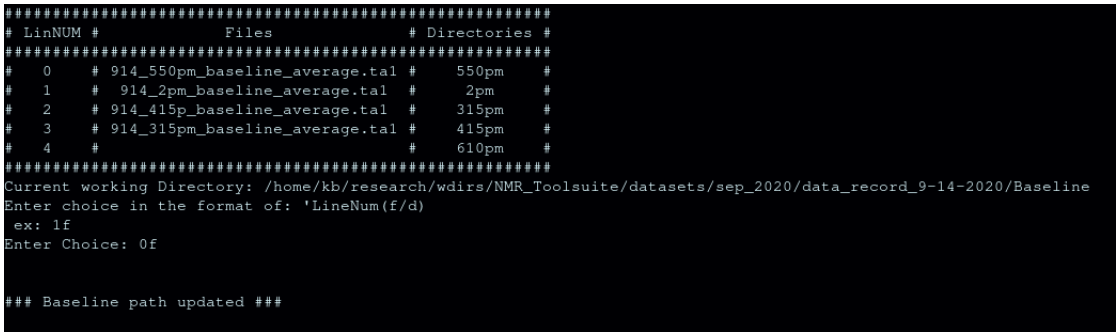


Figure 16: The user enters the baseline of their choice.

3. Next, the user will need to select a raw signal file that contains an NMR spectra. The program automatically changes the current working directory to the parent directory of the previous selection. From here, the user then needs to navigate to the raw-signal path.

‡https://unh.box.com/s/ommw56pj5s8ve6afaog3sei9i9bh7z23

```
### Update Raw Signal ###
Current working directory: /home/kb/research/wdirs/NMR_Toolsuite/datasets/sep_2020/data_record_9-14-2020
#################################################################################################
# LinNUM #                      Files                        #          Directories          #
#################################################################################################
#    0    #   15_Ellie TE calibrated ENHANCED d-Prop.csv    #              TE               #
#    1    # 15_628p-630p TE calibrated ENHANCED d-Prop.csv  #           Baseline            #
#    2    #              14_Lab_dprop_TE.csv               # 914_701a_to_915_405p_enhanced #
#    3    #           14_9_14__1900_dprop_TE.csv           #            515pte             #
#    4    #             14_Ellie_dprop_TE.csv              #            700pte             #
#    5    #    15_LAB TE calibrated ENHANCED d-Prop.csv    #            408pte             #
#    6    #                                                #         no_nmr_status         #
#    7    #                                                #         Polarization          #
#################################################################################################
Current working Directory: /home/kb/research/wdirs/NMR_Toolsuite/datasets/sep_2020/data_record_9-14-2020
Enter choice in the format of: 'LineNum(f/d)
 ex: 1f
Enter Choice:
```

Figure 17: Navigate to the raw-signal path.

4. After the program has accepted the user's input to the rawsignal path, The program will request what type of target-material is the NMR spectras containing. Enter the material type then continue.

5. Then, the program will request that the user name the current instance of the program. Think of it as a human-readable name to label the current analysis session.

```
### Current instance name: 20210215_112016134060033 Instance ###
Input new instance name:
```

Figure 18: By default, the instance-name will be the timestamp of the initial execution of the Signal Extraction toolsuite.

6. Once the instance has been given a name, the graph will be displayed in a popup window, and the terminal will contain options to modify the signal. If the user wants to re-bin the data, make sure the re-binning is done first.

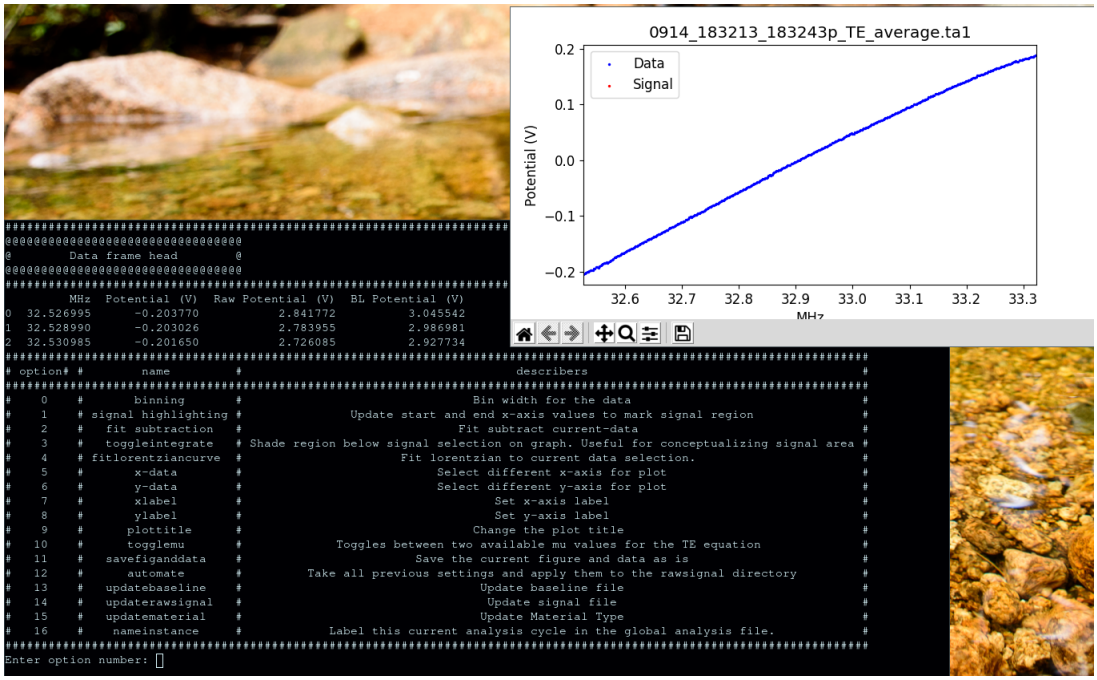


Figure 19: The terminal displays the available signal-extraction options.

7. We will now bin the data. Press the option number, then ENTER.

```
#########################################################################################################
# option# #        name         #                                describers                             #
#########################################################################################################
#     0    #       binning      #                          Bin width for the data                       #
#     1    # signal highlighting #              Update start and end x-axis values to mark signal region  #
#     2    #   fit subtraction  #                          Fit subtract current-data                    #
#     3    #   toggleintegrate  # Shade region below signal selection on graph. Useful for conceptualizing signal area #
#     4    # fitlorentziancurve #                  Fit lorentzian to current data selection.            #
#     5    #       x-data       #                      Select different x-axis for plot                 #
#     6    #       y-data       #                      Select different y-axis for plot                 #
#     7    #       xlabel       #                            Set x-axis label                           #
#     8    #       ylabel       #                            Set y-axis label                           #
#     9    #      plottitle     #                          Change the plot title                        #
#    10    #      togglemu      #           Toggles between two available mu values for the TE equation  #
#    11    #   savefiganddata   #                  Save the current figure and data as is               #
#    12    #      automate      #        Take all previous settings and apply them to the rawsignal directory  #
#    13    #   updatebaseline   #                          Update baseline file                         #
#    14    #   updaterawsignal  #                           Update signal file                          #
#    15    #   updatematerial   #                          Update Material Type                         #
#    16    #    nameinstance    #          Label this current analysis cycle in the global analysis file.  #
#########################################################################################################
Enter option number: 0
Your choice was binning returning...


### Current binning is 1. ###
Please select new binning
Enter bin width: 3
```

Figure 20: Here, the user selected a bin width of 3.

Note the updated current settings on the following output which reflect the entered current binning.

```
Enter bin width: 3
########################################################################
@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
@        Current Settings:         @
@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
# Data Range: 32.52899002493765 to 33.31901496259237 MHz
# Current Signal Highlighting Region 0 to 0 MHz
# Shading: Disabled
# Current binning: 3
# Current mu: PROTON
# Current Baseline File: /home/kb/research/wdirs/NMR_Toolsuite/datasets/sep_2020/data_record_9-14-2020/Baseline/914_550pm_base
erage.ta1
# Current Raw-Signal File: /home/kb/research/wdirs/NMR_Toolsuite/datasets/sep_2020/data_record_9-14-2020/TE/914_629_700p/0914_
183243p_TE_average.ta1
########################################################################
@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
@        Data frame head         @
```

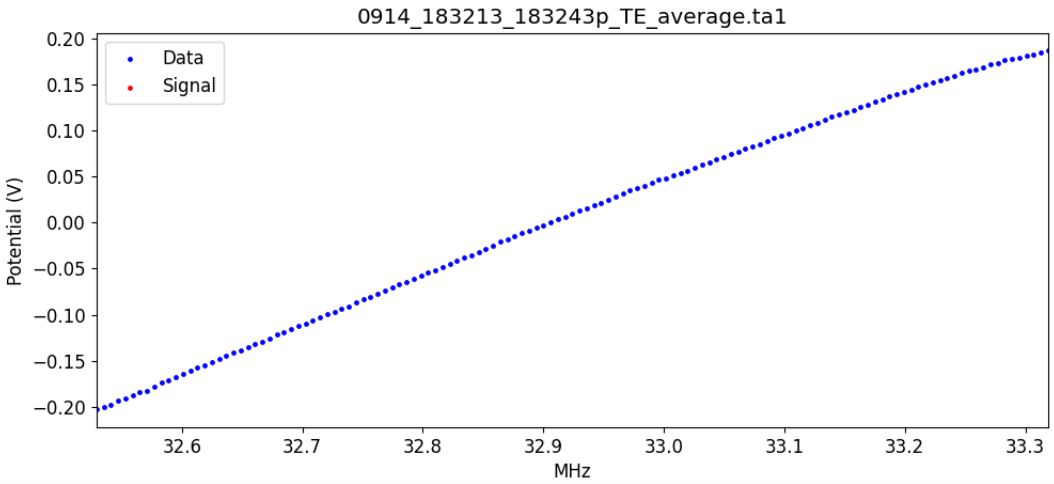


Figure 21: Caption

8. Now, notice how the NMR Spectra does not look like an NMR signal. This is because the baseline taken for this spectra did not perfectly capture the natural response of the q-meter, so a fit subtraction is needed to remove the upward trend. I also think I see a region of data where I anticipate the signal, so, I will need to highlight that region with the signal-highlighting option.

9. Now that the signal is highlighted, it is time to do a fit subtraction on the signal. Select the option and continue.

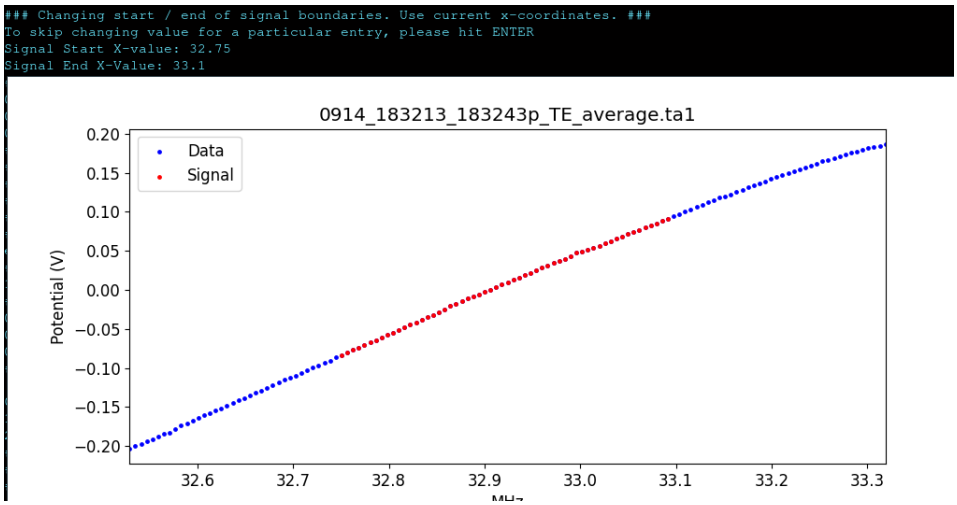


Figure 22: Notice the two bumps? One is at about 32.875 and the other is at about 32.99. These are likely signal artifacts.

10. There are 7 different types of fit subtractions. Most fit subtractions will be covered by the polynomials, or sine functions. If you're fitting the raw NMR Signal (accessible through the y-data) the Lorentzian fit subtraction methods may also be useful. For this dataset, I will select the third order polynomial.

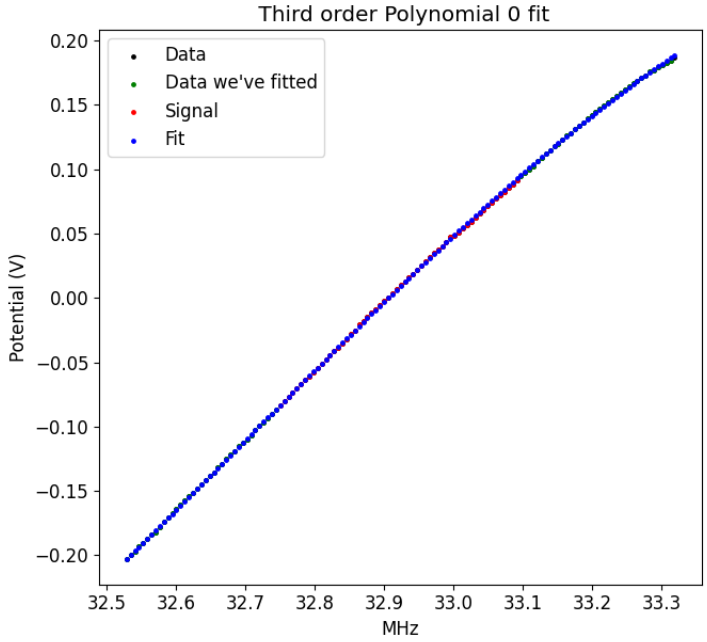


Figure 23: Visually, it looks like the fit of the third order polynomial about the signal region is good.

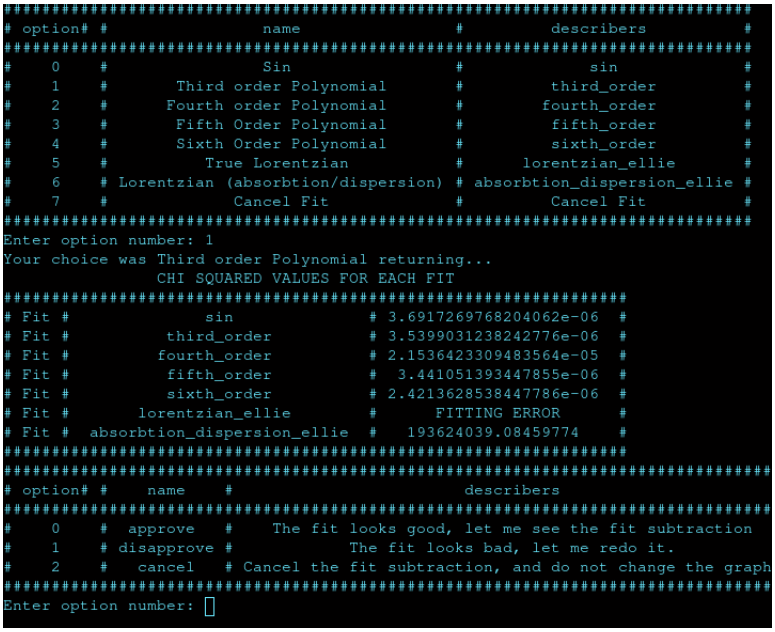
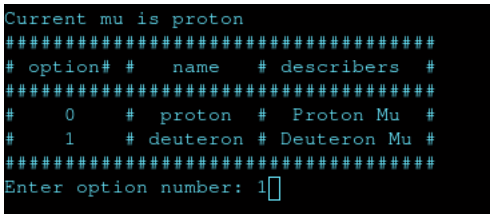


Figure 24: In the terminal, the program will ask for the user to approve, or disapprove the fit, I will approve it.

11. Next, if the current displayed dataset needs an additional fit subtraction, make sure to repeat the last few steps to carry out an additional fit subtraction.

12. After the fit subtraction is done, the user can choose to shade the region of data under the highlighted signal region by selecting the option *toggleintegrate*.

13. Once selected, it becomes clear what the program is integrating. Next, the user can adjust/adapt the axis labels, plot-title, or change out the TE calibration equation. I know that this particular NMR spectra was taken from a deuteron target material. I will toggle the proper mu accordingly to reflect an accurate deuteron TE calculation.



14. Should the dataset being analyzed is from a proton, the option (#4) *fitlorentziancurve* is used to fit a lorentzian curve to the clean NMR spectra. Select this should you be analyzing proton nmr data.

15. Since what I have done is all I needed to do during this signal extraction session, I am ready to automate the signal extraction process. All that is needed for this is simply enter the option to automate, and the program will do the rest.

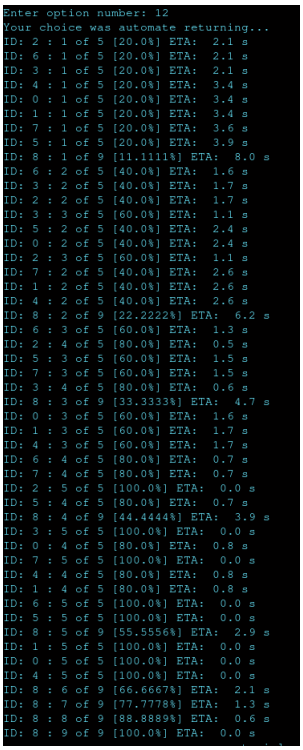


Figure 25: Multi-Processing parallelization results in a lot of text printed to the console. Each line printed here represents a single NMR spectra successfully analyzed. 49 NMR spectra were analyzed in just over 2 seconds on a 16 core processor.

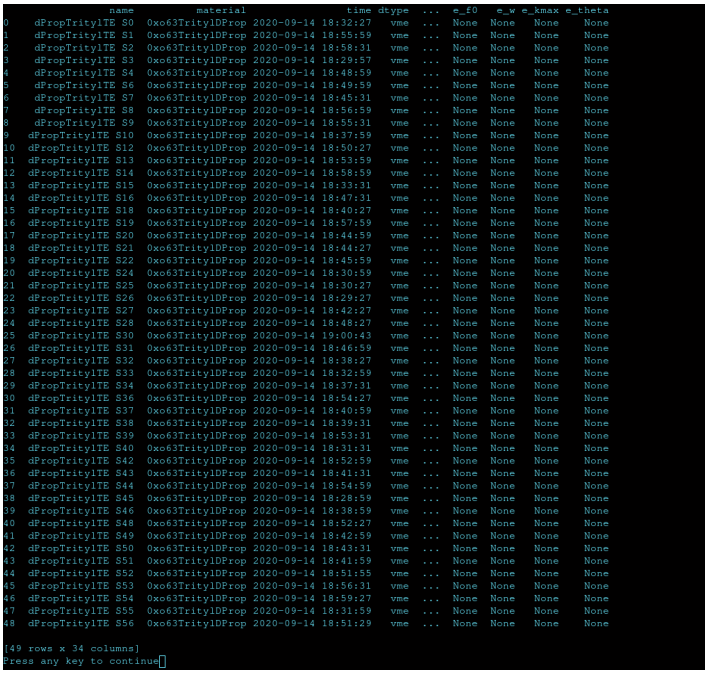


Figure 26: After a successful analysis of an entire dataset, the program will print out the results array, and then save it after the user presses enter.

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)]. X-ranges to fit each function in *function* with. Will zip *fit_bounds* with *function*. Must be equal length as *function*, unless len(*function*) is 1.

sf & ff

Intergers. If sf, and ff parameters are not assigned a value during this functions invocation, the function will default to fitting *everything except the signal.*

If sf and ff parameters are assigned a value during this function's invocation, the function will fit data in the following slices:

- Region 1: (xdata[sf:start], ydata[sf:start]);
- Region 2: (xdata[finish:ff], ydata[finish:ff])

If fit_sans_signal is True, *sf* and *ff* are used to narrow the fit region to the left and right of the peak, the fitting regions to the left and right of the signal will be narrowed from the ends of the dataset. This ASCII 'art' may help illustrate.

```
#     (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.

4 (depreciated) Tkinter GUI: gui.py

THIS HAS BEEN NON FUNCTIONAL SINCE THE RELEASE OF Python 3.9.1!!!

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

4.1 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 27: This is the beginning page in which the user is able to select any of the tools embedded in the tool-suite.

4.2 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.

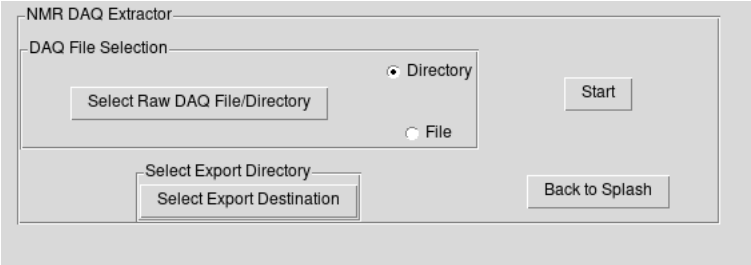


Figure 28: Caption

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.

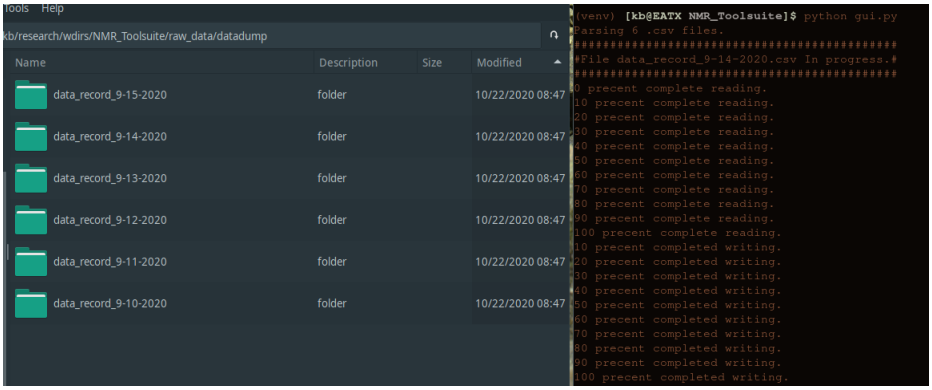


Figure 29: Caption

4.3 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.

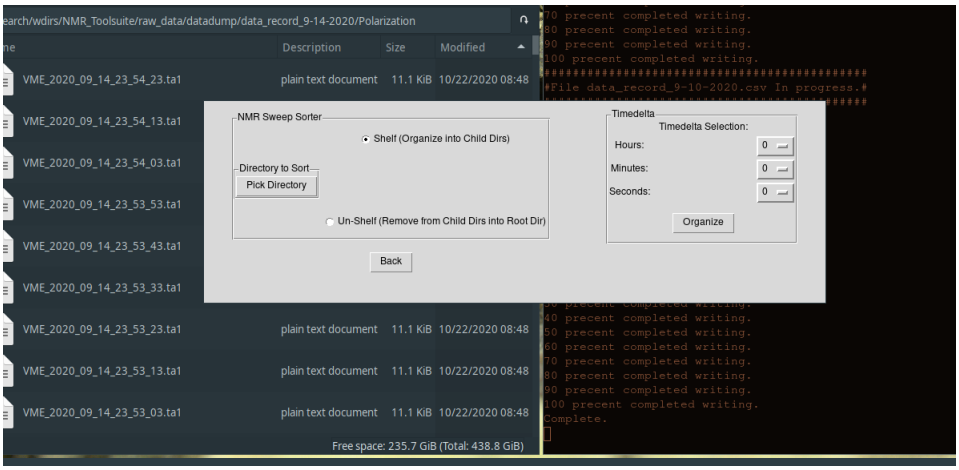


Figure 30: Caption

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

4.4 Directory Averager

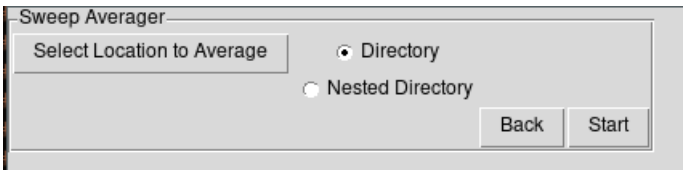


Figure 31: Caption

4.5 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.

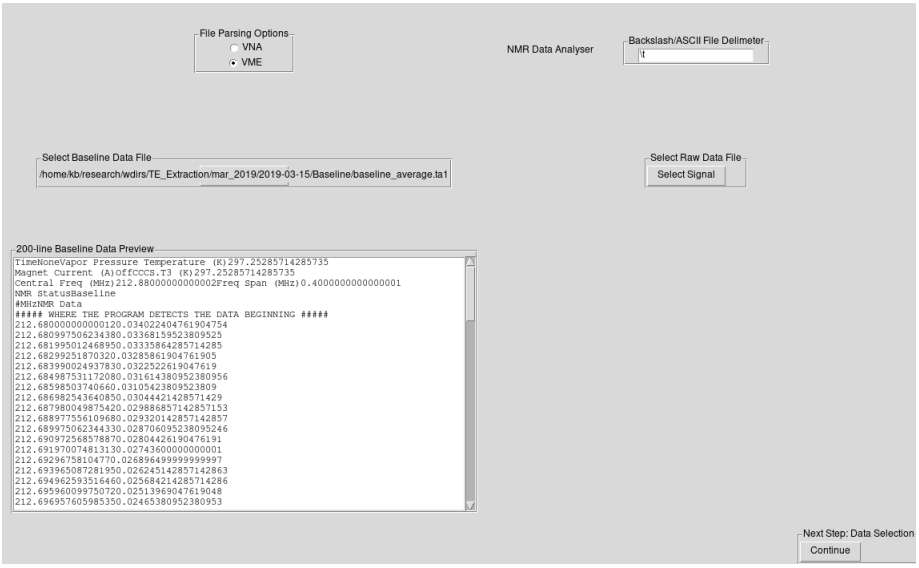


Figure 32: Once the file-dialogue has successfully selected a valid file, a 200 line file preview will appear.

And likewise for the signal file:

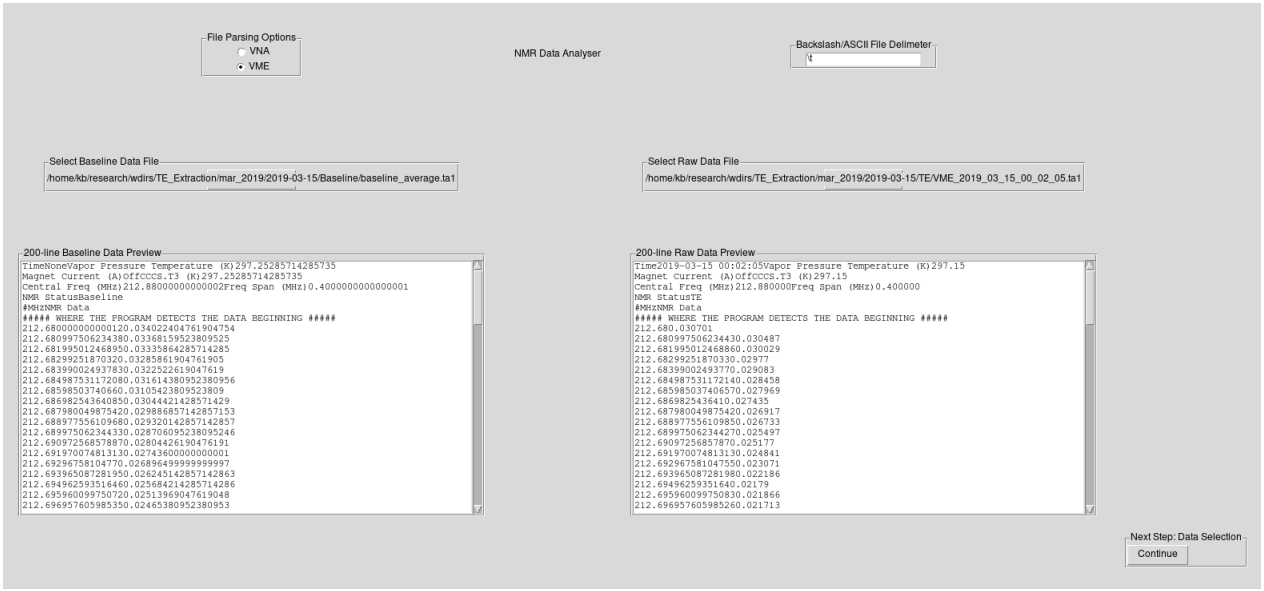


Figure 33: Caption

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

4.5.1 Data Selection and Binning

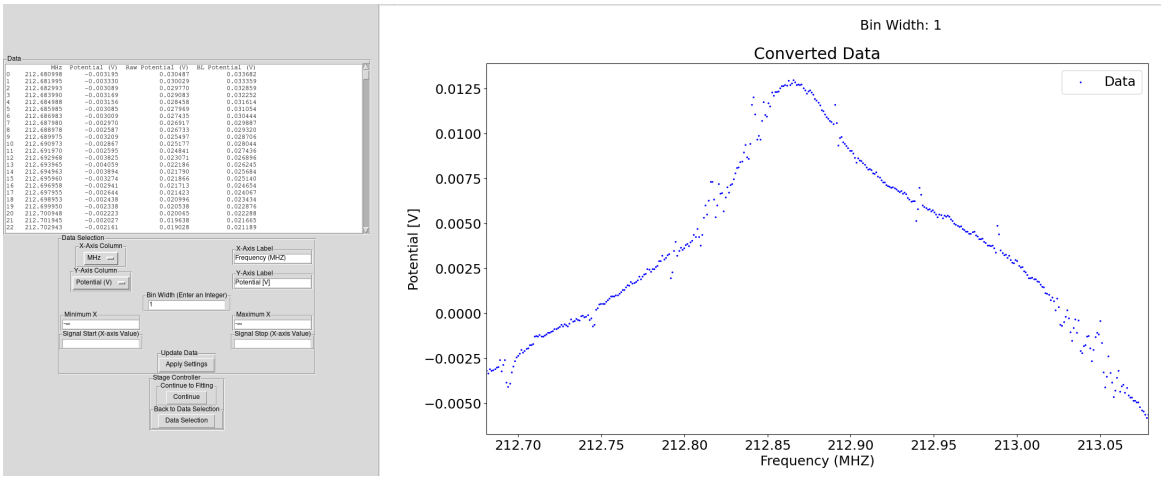


Figure 34: Data Selector

Figure 34 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()*

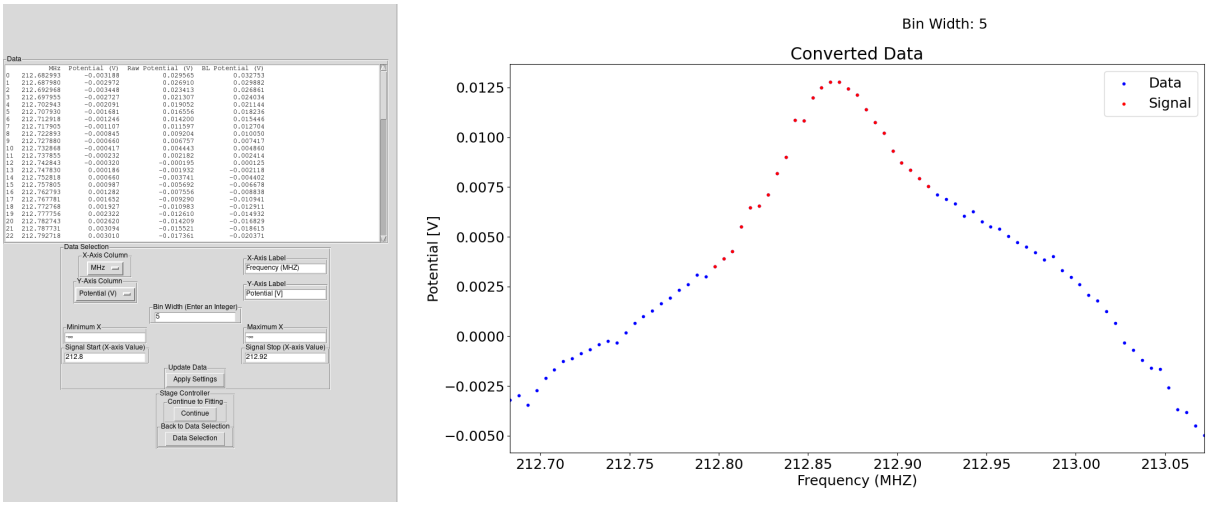


Figure 35: Caption

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

4.5.2 Dataset Fitting

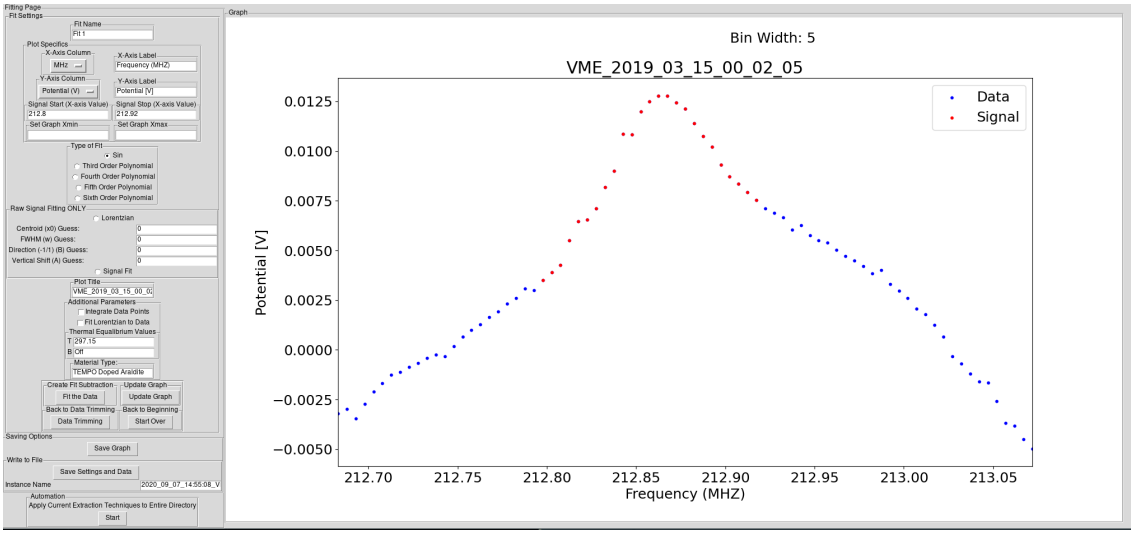


Figure 36: Caption

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.

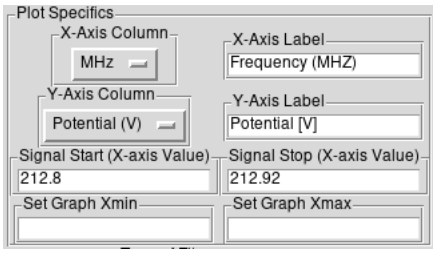


Figure 37: Caption

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"

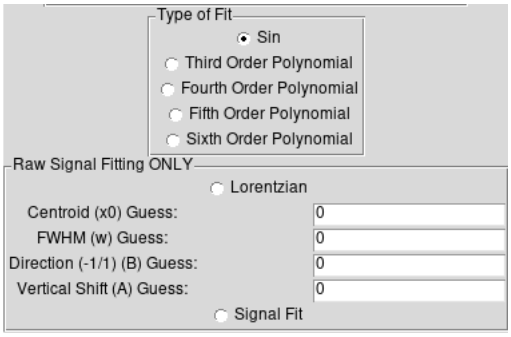


Figure 38: Caption

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

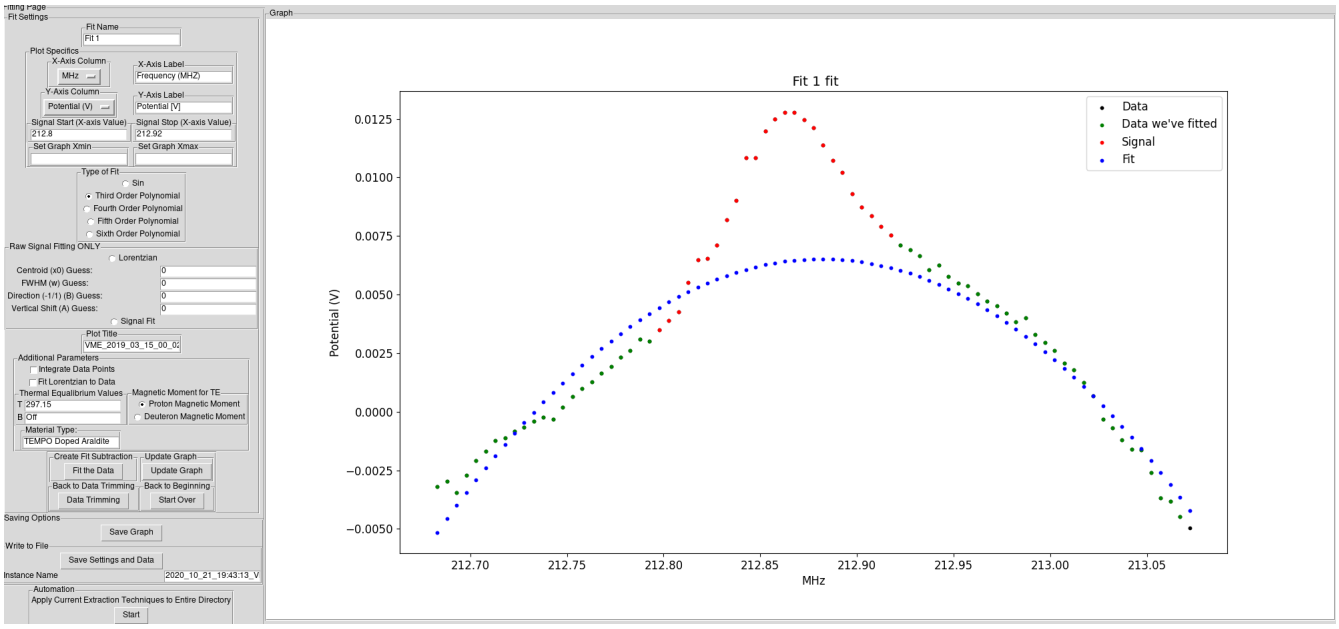


Figure 39: Note the dotted blue lines.

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**

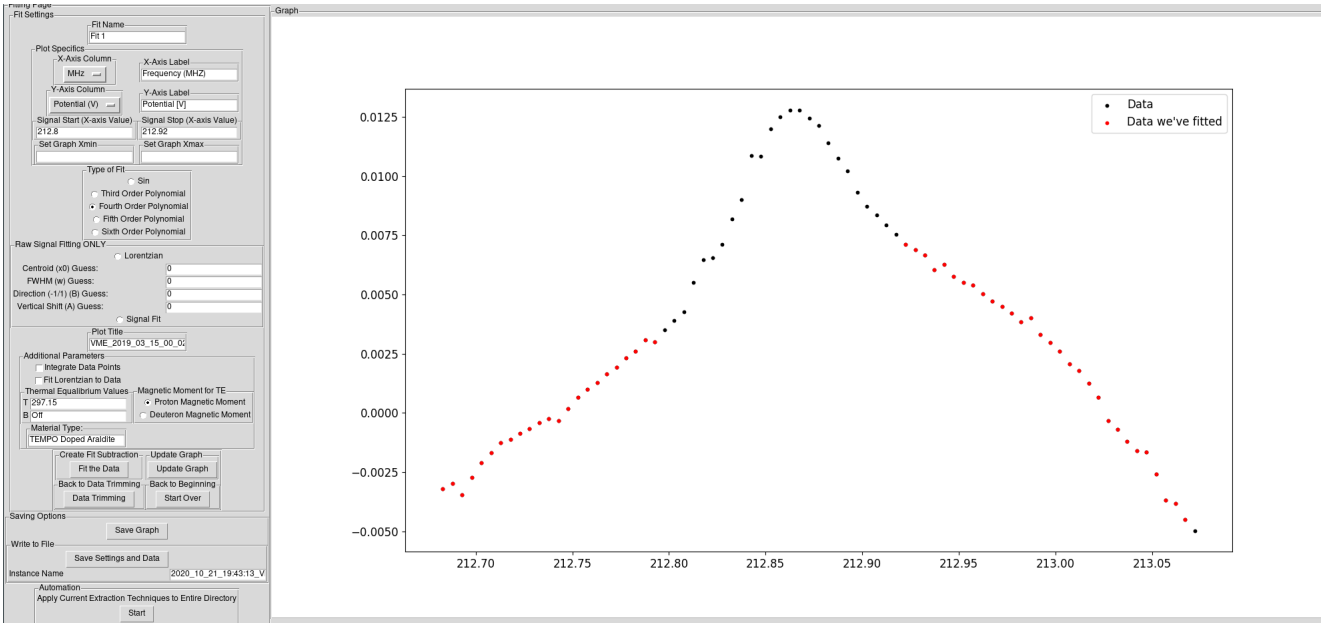


Figure 40:

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

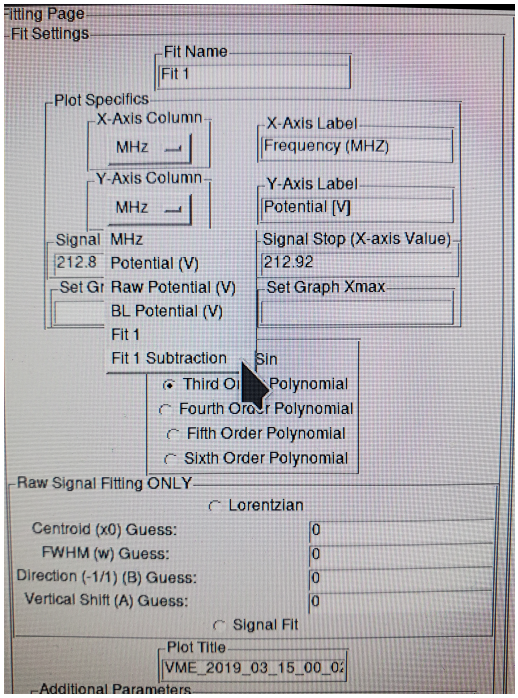


Figure 41: Appologies for the moiré. Remember to update the graph after selecting a new y-axis column.

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.

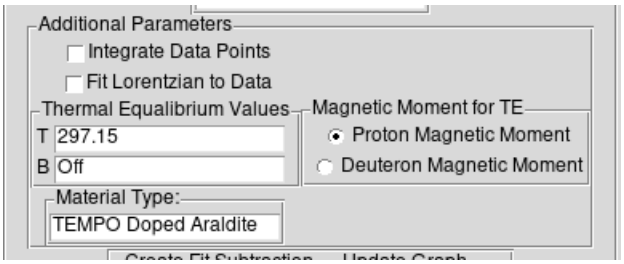


Figure 42: Caption

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

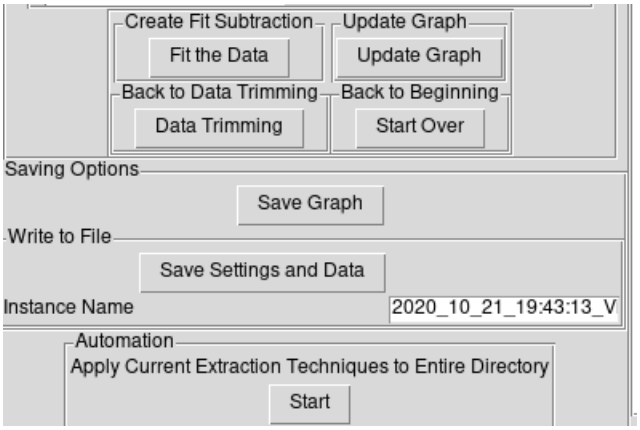


Figure 43: Caption

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.

4.5.3 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.