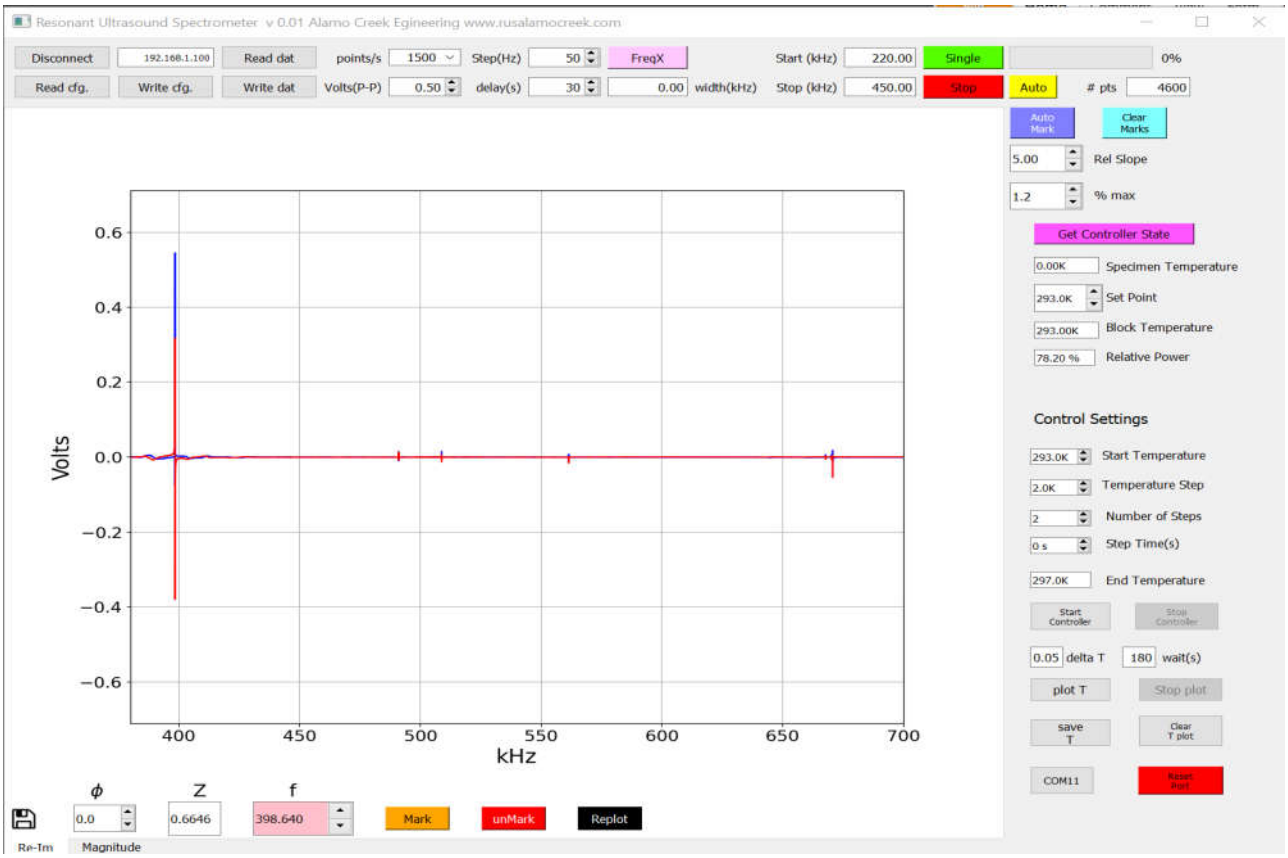
Alamo Creek Engineering RUSxpy application manual: 1 March 2022



Overview: RUSxpy can scan a given range at adjustable resolution and noise reduction, detect peaks, and if desired, scan each of the detected peaks at high resolution. If an ACE temperature control system is connected, two temperature control modes are available. One mode sends a program to the Arduino-based temperature controller where start temperature, step size, number of steps, and time at each step are specified. The controller runs independently of RUSxpy. Data is acquired by executing the **AUTO** function with desired time between scans. The **save T** button saves the temperature history. The other mode sends the controller a set point, waits for equilibrium, and then takes a scan. At the end of the process, the temperature history is saved automatically. In this mode, either a full scan at each temperature, or a high resolution scan of detected peaks can be implemented.

Installation:

- Copy entire contents of USB drive to a folder on your desktop.
- Copy entire contents of subfolder RUS Spectrometer Firmware to root of a formatted blank micro sd card. Insert into ACE RUS008 spectrometer.
- If necessary, use Arduino IDE to update temperature controller firmware. The updated firmware will run under either the LANL Labview control program or the ACE Python control program.
- The RUSpy.exe file is the windows executable of the Python 3.8 data acquisition program. (Source code is available at https://github.com/rus-ace/RUSpy). It will auto detect both the High Temperature option and the Thermoelectric stage.

Description of control functions:

Note-- subfolder of folder with RUSpy.exe "RUS_Data" will be created by on run.
Note--the little floppy disk icon takes a high resolution screen shot of the graph.

- **Disconnect--Connect**: Connects PC to RUS 008 at address **192.168.1.100**. This is fixed by the RUS008 and should only be changed if you really know what you are doing..

- **Read cfg.--Write cfg**: Stores current sweep settings in file rusx.ini or a file name of your choice. This is an editable text file.
  It looks like:

[General]
addr=192.168.1.100 : ethernet address of RUS008.
Level_1=2 : The drive level in volts.
Rate=0 : 0=1500pts/s, 1 = 500 pts/s 2 = 50pts/s, 3 = 50 pts/s
reim_start=220 :start frequency(kHz)
reim_stop=450 :end frequency(kHz)
step=50 :frequency step in Hz
Terr = -0.1 :thermoelectric error between block and sample thermometers

- **Read dat—Write dat:** Stores raw real and imaginary sweep data in a standard file located in subdirectory **/Data**. This can be read back in and processed just as if a scan were taken. The file name is like **296.0_K_21022022082920_D.dat** where the temperature of controller set point is first, and the day, month, year, and time in hh.mm.ss. format. If no controller is present, the temperature is replaced by 0.0. The **Specimen Temperature** is replaced by the temperature recorded during the saved sweep. The first line of the file is specimen temperature if a controller is connected, otherwise it is 0.0. The second line is drive voltage, the third, the time the file was taken after program start in seconds. The remainder contains the raw data:

296.03 K
1.00 V
4.27 s

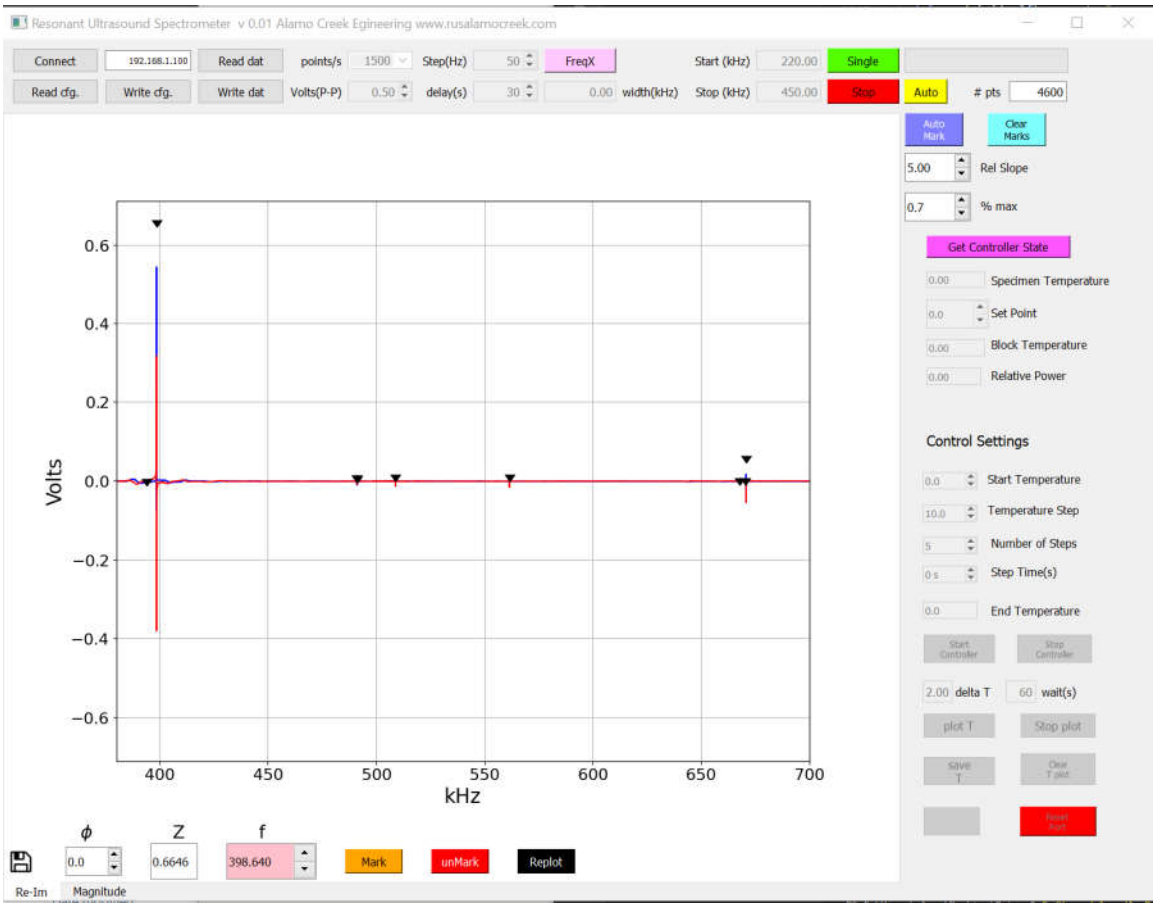
frequency	real	imag
250.00	0.0002557	0.0001221
250.01	0.0001736	0.0003753
250.02	0.0002193	0.0004464
250.03	0.0002906	0.0003615

…

- **points/s**: this is a spin box with useful (non editable) values of the rate at which spectrum data is acquired. The lower the rate, the more noise reduction. 0=1500pts/s, 1 = 500 pts/s 2 = 50pts/s, 3 = 50 pts/s

- **Volts(P-P)**: The actual spectrometer transducer drive voltage. It is editable and goes from 0 to 25 V P-P.
- **Step(Hz)**: The step between successive data points.
- **delay(s)**: The delay between sweeps when **Auto** sweep is selected. Sweep data is written to a standard data file file after each sweep
- **Start(kHz)--Stop(kHz)**: The start end end frequencies for a sweep. This is editable.
- **Single--Stop--Auto**: Initiates a single sweep, stops a single or auto sweep, or initiates auto sweep at intervals of **delay(s)**. The progress bar indicates the sweep state during a sweep.
- **# pts**: The number of data points in the sweep. This is computed from the set parameters and is not editable.
- **Re-Im--Magnitude** tabs: Plots Real/Imaginary data or magnitude data.

Marking and finding peaks:



- **Auto Mark--Clear Marks--Rel Slope--%max**: Peaks are detected and show as black triangles. Any data below **% mark** is ignored—this is a threshold. **Rel Slope** is a parameter that basically determines how sharp a peak is. It uses a calculation of the RMS noise taken from the first 16 points of the sweep. Therefore, those 16 points must ***not contain*** a resonance. The higher the number, the steeper the slope must be for inclusion in the list. The frequencies found are written in ascending order to e.g. **296.0_K_21022022101400_F.dat** in **/Dat**a. (The first part of the file name is the integer of rounded off specimen temperate) in the format freq(MHz), amplitude, 1.0. The first line is temperature at which the peaks are detected.
  99.1 C
  0.3941000 0.0050 1.0
  0.3986400 0.6646 1.0 ...
  so that the results can be cut and pasted into the analysis code input file. The file is updated at any change in found peaks. **Clear Marks** removes all peaks both from plot and file.
  Note that as **Step(Hz)** is reduced, so must the **Rel Slope** to detect peaks. For **Step(Hz)** = 50, **Rel Slope** might be 5**,** For **Step(Hz) = 1**, **Rel Slope** might be 0.2.

- **Blue crosshair cursor--Mark--unMark--Replot--black vertical cursor**—blue crosshair appears when mouse is inside plot area. This cursor is used to select a region for processing. Only the vertical line determines the region size, and the selected region is auto-scaled to fit. Inside that region is a black vertical cursor whose height is the height of the datum at that point. The datum height is displayed in the editable spinbox "**f**" The black cursor always first appears at the highest data point in the window. Clicking "**Mark**" will add the point at the cursor to the

***.F.dat** file, which sorts that frequency to maintain an ordered list. A peak does not have to be present to add that frequency to the list. The spinbox "**f**" is the only way to move the black cursor. If moved to another peak in the window, "**Mark**" will add that peak to ***.F.dat**. Clicking **unMark** will remove from plot and file any marked peaks in the window. This is useful to remove spurious peaks. **Replot** plots the entire sweep with marked points visible. The marked points show in both **Re-Im** or **Magnitude** plots. **Clear Marks** removes all peaks from plot and file.

- **ϕ**: In the **Re-Im** plot, you can roll the phase around with the spinbox.

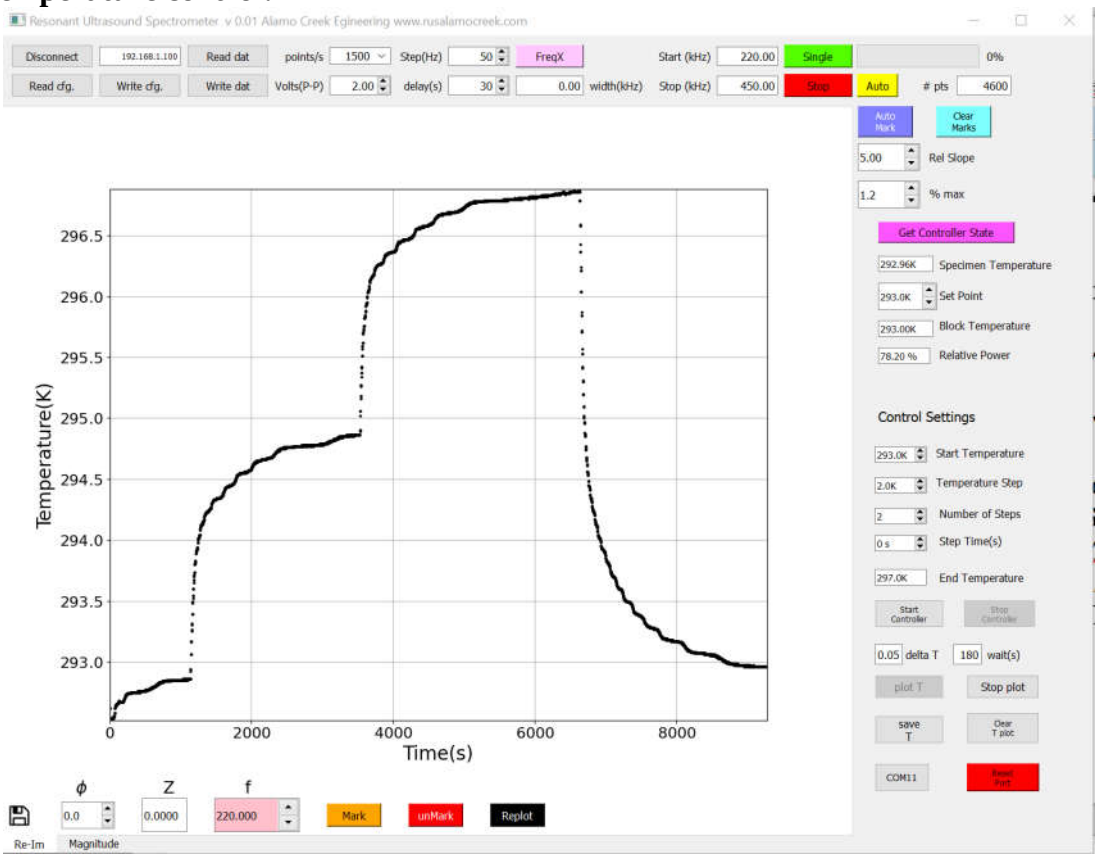
Saved data:

- **Read Dat--Write Dat**: The raw data file can be read in from the **\Data** directory just as if a scan were taken, and analyzed with the various peak mark and detect functions, writing a ***F.dat** file with the result. Either ***D.dat** or ***X.dat** can be read (see **FreqX** description).
- The drive voltage for the read-in scan appears in **Volts(P-P)** window, and the read-in temperature is displayed in the **Specimen Temperature** window.

Scanning only around identified peaks:

The **FreqX** and **width (kHz)** controls require a ***.F.DAT** file to be present with identified peaks. This mode is activated when **width (kHz)** is greater than 0. Each peak is scanned by clicking **FreqX** with 1000 pts. The frequency width of that scan is determined by **width (kHz).** So for **width (kHz)** =2 the scan is 2kHz wide per peak with 2 Hz steps. There is no limit to the number of peaks. The actual 1000-pt -window is positioned so the peak is not centered, but located at about 15% of the window width nearest the high frequency end. This is because most materials soften, moving the peaks to lower frequency as temperature is raised. If peaks move a lot during a temperature scan, one can add frequencies to **.F.DAT** even if no peak is present at the start.

Temperature control:



- There are two modes of operation. **Python** and **Arduino**. Peak detection does not work in either mode during control and the spectrum does not plot.
- If an ACE temperature controller with latest firmware is present, it will be detected, the controls for it will be made active in this code, and the com port shown.
- If necessary, the **Reset Port** button will reset the temperature controller.
- Entering a **Set Point** immediately changes the control temperature to that value when in non-temperature-sweep mode.
- Clicking **Get Controller State** reads all available controller values. Controller values are not automatically updated unless **plot T** is active.
-

Closed loop python mode:

In this mode, the Arduino actually controls temperature. Though overshoot will be observed in the control thermometers, the sample temperature never overshoots. **RUSxpy** does the following.
The system reads the sample temperature at 5 s intervals for **wait(s)** seconds. At the end of this period, **Specimen Temperature** is compared to **Set Point**. **wait(s)** is basically the sample equilibrium time. (The time constant for the high temperature option is around 300s, from 20C to 400C for the thermoelectric stage, around 2000s for a 10K temperature change). If the difference is less than **delta T**, a counter is incremented. This is used to add to the running average. When that counter reaches 32, a sweep is made. Note that in the rusx.ini file is a parameter Terr. It is only used for the thermoelectric stage and is the difference between block and specimen thermometers. For example, if the block reads 290K and the specimen reads 289.89K T err = -0.11. This is different for each system. The user determines it by settings the controller to a given temperature and waiting an hour of so before reading the specimen temperature. If set wrong, equilibrium will be difficult to hit. Once determined, it is left fixed in rusx.ini.

To access closed-loop mode (Python mode), set Step Time(s) = 0.

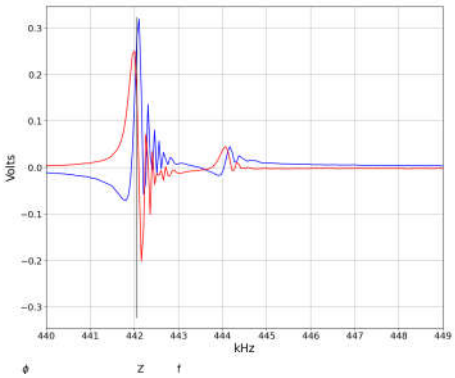
- Set the **Start Temperature, Temperature Step, Number of Steps**. The **End Temperature** is computed from these.
- Click **plot T** and click **Start Controller**. The **Get Controller State** is updated continuously.
- The system will go to each temperature (see figure above) test for equilibration, take a scan,
- Each scan will write a data file like **296.0_K_21022022082920_D.dat** where the temperature of controller set point is first, and the day, month, year, and time in hh.mm.ss. If **width (kHz) is 0,** a full scan with parameters set in the windows is taken and data stored I ***D.DAT**. If **width (kHz)** is greater than **0**, only peaks identified are scanned at 50 pt/s (fixed) and data stored in a file like **296.0_K_21022022082920_X.dat**. the The first line of either file is temperature at end of sweep, the second line, drive voltage, and the rest, the raw sweep data in "Freq, Real, Imag" format. Temperature units are auto detected and are either C for the high temperature option, or K for the thermoelectric stage.
- At the end, the temperature returns to the start temperature and the controller stops. Temperature record is automatically saved in a file like **C_27032022164031_T.dat** with time-coded temperature data every 5 seconds in two columns, time in seconds, T in whatever units are active.
- Click **Stop Plot** to return to normal operation.
- **Save T** saves a file like **C_27032022164031_T.dat** with time-coded temperature data every 5 seconds in two columns, time in seconds, T in whatever units are active.

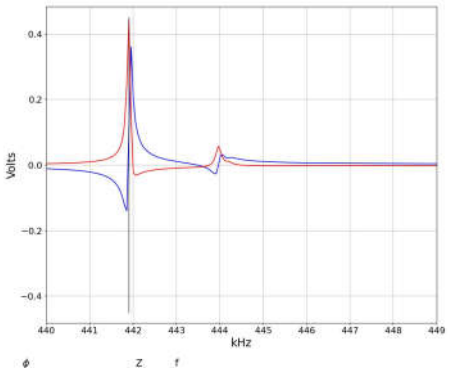
To enter Arduino mode (equivalent to the LANL Labview temperature control process), an open-loop mode where *equilibrium is not detected*, set **width (kHz)** = 0**, Step Time(s)** > 0 seconds and **Temperature Steps** > 0 so that the temperature changes slowly enough that the sample temperature is close to the **Specimen Temperature**. The idea is to make small temperature steps with adequate time between them for equilibration. This mode is not recommended.

- Set **delay(s)** for the desired delay between scans. Note that one must consider that the total time for a scan cycle = **delay(s)**+the time for a scan in relation to the **Step Time(s).** Each scan will write a data file like **296.12_C_21022022082920_D.dat** with the file name the temperature at end of scan as the first characters, the first line of the file is specimen temperature at end of scan, the second line, drive voltage, and the rest, the raw sweep data. Set **Start Temperature**. Set **Number of Steps**. The **End Temperature** is computed and displayed.
- Click **plot T** to start a graph of temperature vs time. All but the set point of **Get Controller State** will update continuously. **Clear T Plot** resets the temperature time axis to zero.
- Click **Start Controller** to send the temperature program to the temperature controller. At this point, the temperature controller operates independently of **RUSxpy.exe**.
- Click **Auto** to begin data acquisition at the intervals set by **delay(s)**.
- The progress is seen in the progress bar. Note that the **Block Temperature** is only valid for the thermoelectric stage, as that is the temperature actually controlled.
- The **Specimen Temperature** will be displayed as things progress, but not the **Set Point**. The **Relative Power** is also displayed,
- At the end of the temperature program, the temperature stays at the **End Temperature**.
- Click **Stop Plot** to return to normal operation.
- On end of control program, clicking save **T will** write a file like **C_27032022164031_T.dat** which will have time-coded temperature data every 5 seconds in two columns, time in seconds, T in whatever units are active. In Arduino mode it is not written automatically.

Strategy for finding peaks:

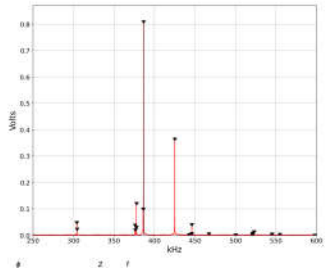
The figures below were both taken at 50 Hz steps, but the one on the left at 1500 pt/s, the one on the right at 50 pt/s. Both are screen shots using the floppy icon.

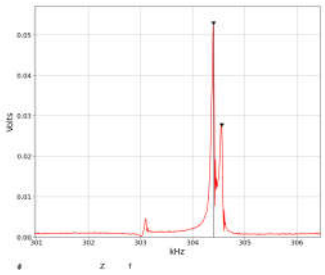


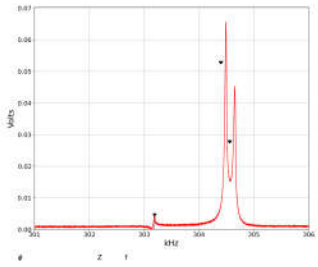


The difference is actually physics. With a high-Q resonator, running through the resonance faster than it can ring down produces a beat frequency between the resonator fundamental frequency and the drive frequency, causing the sawtooth for the scan on the left. It is, nevertheless, very useful in determining if the sample is producing good signals, to run at 1500 pts/s until a good spectrum is obtained. Then, for the actual measurement, use one of the lower rates. The sharper the resonance, the lower the rate required to prevent ringing. Note that the peak detector on this scan only detects the correct peaks at either rate, *but that is not to be counted on*. The sawtooth may generate spurious detected peaks.

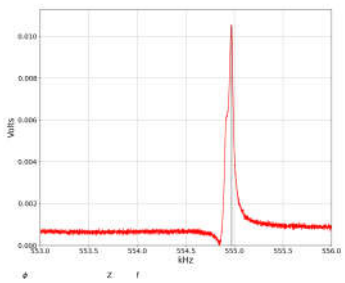
The code is hard-limited to 32766 data points. For a broad scan on high-Q material, it may not be possible to eliminate ringing with even the slowest acquisition rate. Using the blue crosshairs, ringing is visible. Rescanning at very small steps just the region in question to be sure if a peak is real or ringing, produces a plot where one can add a frequency using the black cursor to the list without losing the already-detected peaks. Note that the previously detected peaks are now slightly shifted. This is a common issue associated with tiny variations between scans from vibration, temperature etc.

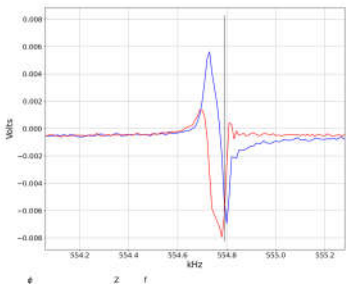


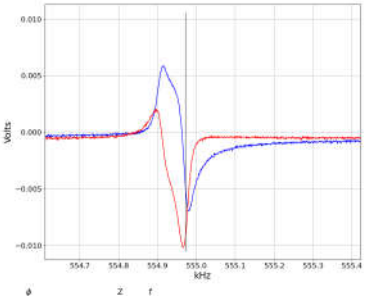




The peak added just above 303 kHz is most easily added by using the blue crosshairs to isolate it, because moving the black cursor with arrows will put it in a region of low signal, making it nearly invisible. Note that auto detecting peaks will result in duplicates now, because of the slight shift.
It is also useful to look at the re-im plot at small steps if a peak is questionable. The distorted shape is a giveaway.







Note that every single peak predicted by the code is found in the single scan without repositioning the sample. This may not always be the case, *but one does not need to find every peak for a good fit*. However, it is essential to find the lowest mode. That mode was not detected with the peak finder but had to be added manually using the black cursor. Here is the result for the sample data supplied with these instructions.

cylout.dat - Notepad

File Edit Format View Help

```
 LANL Cylcode Ver. 10.0  units cm, grams, GPa, MHz
            Steel Dowell pin
using 12 order polynomials   mass=   1.3210 gm  rho=  7.796 gm/cc

                                                    2 x logarithmic derivates with respec
  n    fexp(MHz)   fcalc(MHz)    %err  wt   k    i     dlnf/dlnc11 dlnf/dlnc44
  1    0.3031000   0.3031140    0.00 1.00   4    1     0.00000     1.00000
  2    0.3044000   0.3042127   -0.06 1.00   6    2     0.00698     0.99302
  3    0.3045600   0.3042127   -0.11 1.00   4    2     0.00698     0.99302
  4    0.3762400   0.3760998   -0.04 1.00   5    1     0.00300     0.99700
  5    0.3764200   0.3760998   -0.09 1.00   3    1     0.00300     0.99700
  6    0.3778200   0.3777687   -0.01 1.00   2    1     0.03408     0.96592
  7    0.3779600   0.3777687   -0.05 1.00   8    1     0.03408     0.96592
  8    0.3862200   0.3865702    0.09 1.00   7    2     0.05284     0.94716
  9    0.3866300   0.3865702   -0.02 1.00   1    2     0.05284     0.94716
 10    0.4252300   0.4247562   -0.11 1.00   5    2     0.00071     0.99929
 11    0.4428500   0.4427742   -0.02 1.00   6    3     0.40215     0.59785
 12    0.4460800   0.4459907   -0.02 1.00   5    3     0.09304     0.90696
 13    0.4463000   0.4459908   -0.07 1.00   3    2     0.09304     0.90696
 14    0.4670200   0.4673282    0.07 1.00   7    3     0.32540     0.67460
 15    0.4671500   0.4673282    0.04 1.00   1    3     0.32540     0.67460
 16    0.5003700   0.5007430    0.07 1.00   2    2     0.00632     0.99368
 17    0.5004600   0.5007430    0.06 1.00   8    2     0.00632     0.99368
 18    0.5210600   0.5220258    0.19 1.00   8    3     0.10588     0.89412
 19    0.5211700   0.5220258    0.16 1.00   2    3     0.10588     0.89412
 20    0.5232800   0.5225912   -0.13 1.00   5    4     0.44600     0.55400
 21    0.5449800   0.5450450    0.01 1.00   4    3     0.06226     0.93774
 22    0.5452700   0.5450450   -0.04 1.00   6    4     0.06226     0.93774
 23    0.5547400   0.5548365    0.02 1.00   7    4     0.01786     0.98214
 24    0.5547900   0.5548365    0.01 1.00   1    4     0.01786     0.98214
 25    0.5868800   0.5871673    0.05 1.00   7    5     0.06639     0.93361
 26    0.0000000   0.5871673    0.00 0.00   1    5     0.06639     0.93361
 27    0.0000000   0.5972537    0.00 0.00   7    6     0.11999     0.88001
 28    0.0000000   0.5972537    0.00 0.00   1    6     0.11999     0.88001
 29    0.0000000   0.6062281    0.00 0.00   3    3     0.00000     1.00000
 30    0.0000000   0.6222351    0.00 0.00   6    5     0.11764     0.88236

   Bulk Modulus(GPa)= 163.6353
Youngs modulus(GPa) = 207.7819    Shear modulus=  80.6376     Poissons ratio =   0.288

                This is an isotropic system

c11(GPa)= 271.15213                          V(compressional)(km/s)=  5.89738
c44(GPa)=  80.63760                                 V(shear)(km/s)=  3.21604

   d1(cm)   d2(cm)   d3(cm)
  0.63770  0.63770  0.53050

 loop# 4  rms error=  0.0781 %, changed by -.0000011 %

rms error increased 2% by the following % changes in
  c11   c44
  0.16  0.02
```

Note that the doublet at 587 kHz is not included as a doublet. Try yourself to see if it is there. You will find that it is.