ElecSus GUI – User Guide

For ElecSus version 3.0.0

Contents:

Last updated: 2017/08/16

1. Introduction

ElecSus is a program to calculate the weak-probe *Elec*tric *Sus*ceptibility of an atomic vapour, which determines many of the optical properties of the vapour. ElecSus currently supports the alkali-metal atoms, in the vicinity of their D1 and D2 resonance lines ($nS_{1/2}$ to $nP_{1/2}$ and $nP_{3/2}$ transitions). The optical properties are calculated as a function of laser detuning – the laser frequency relative to a line-centre frequency.

For more details on the physics, and the back-end of ElecSus, see our paper in Computer Physics Communications. If the use of this program contributes to any research output, please cite these papers:

M. A. Zentile *et. al.*, Comp. Phys. Commun. **189**, 162 (2015).

J. Keaveney *et. al.*, arXiv.org 1708.05305 (2017)

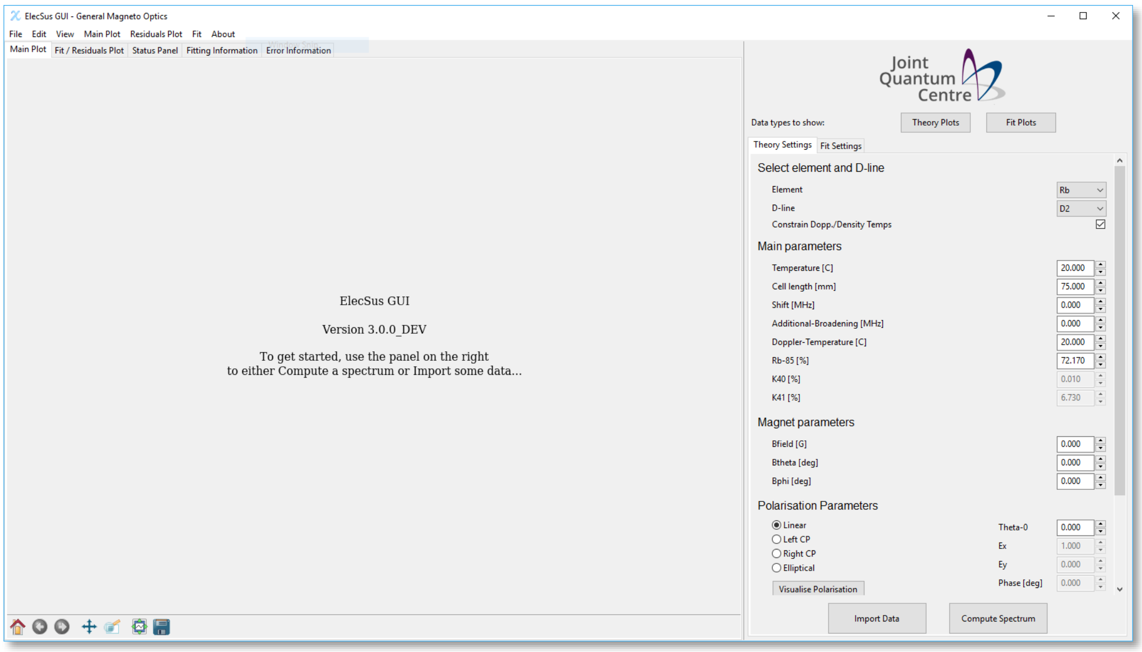
This user guide concerns the graphical user interface (GUI) to ElecSus. The GUI is designed to:

- Speed up the process of calculating and displaying spectra
- Allow pseudo-real-time parameter adjustment to look at how altering a particular parameter, or set of parameters, affects the spectrum
- Expedite the process of fitting experimental data for the most commonly encountered data sets, allowing quantitative comparison between experiment and theory.

This guide will outline each element of the GUI and its operation.

2. Elements of the front panel

The main panel has several elements which we will go through separately.



ElecSus GUI

Version 3.0.0_DEV

To get started, use the panel on the right
to either Compute a spectrum or Import some data...

Data types to show:

Select element and D-line

Element	Rb
D-line	D2
Constrain Dopp./Density Temps	✓

Main parameters

Temperature [C]	20.000
Cell length [mm]	75.000
Shift [MHz]	0.000
Additional-Broadening [MHz]	0.000
Doppler-Temperature [C]	20.000
Rb-85 [%]	72.170
K40 [%]	0.010
K41 [%]	6.730

Magnet parameters

Bfield [G]	0.000
Btheta [deg]	0.000
Bphi [deg]	0.000

Polarisation Parameters

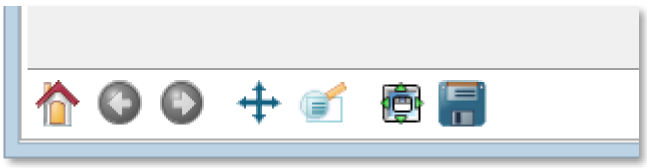
[✓ Linear] Left CP Right CP Elliptical

Theta-0	0.000
Ex	1.000
Ey	0.000
Phase [deg]	0.000

Along the top of the window are the menus (in Ubuntu this bar appears on the top of the screen). The menu bar items will be discussed in later sections.

a. Plot panel tabs

Just under the menu bar is a list of tabs. The first two of these will display plot windows. The main plot shows theory curves and experimental data, if it has been loaded. The fit/residual plot shows the last loaded experimental plot and the current theory curve, and the difference between the two (the residuals). This plot will be updated when a fit is run. Both of the plots are interactive, allowing the user to pan, zoom, save data and adjust the subplots (figure panels) and are controlled via the matplotlib toolbar, shown below:

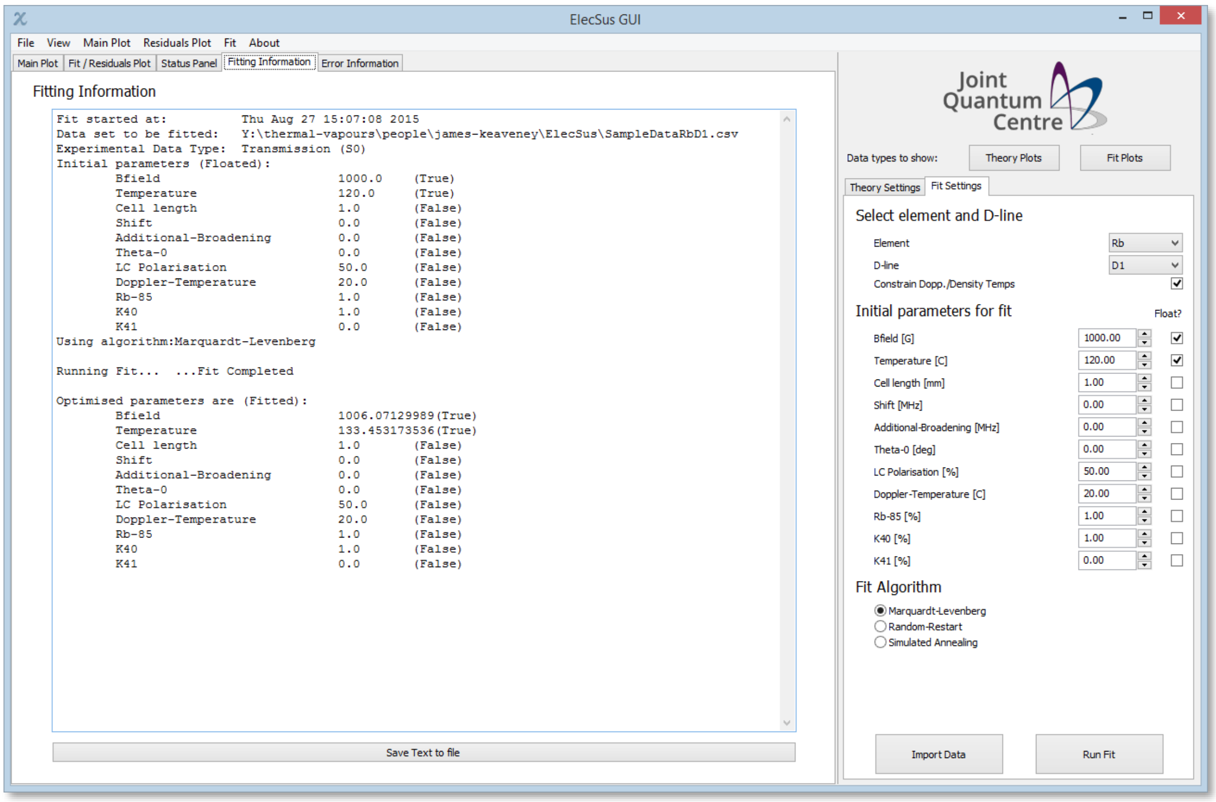


The icons on this toolbar may look slightly different if you're using the newest version of matplotlib, but the buttons do exactly the same jobs. The buttons are relatively intuitive, but if you're not familiar with how this toolbar works, see the matplotlib documentation for details.

http://matplotlib.org/users/navigation_toolbar.html

Note that the navigation keyboard shortcuts are not functional in the GUI.

b. Status panel tabs



Fitting Information

```
Fit started at:         Thu Aug 27 15:07:08 2015
Data set to be fitted:  Y:\thermal-vapours\people\james-keaveney\ElecSus\SampleDataRbD1.csv
Experimental Data Type:  Transmission (S0)
Initial parameters (Floated):
        Bfield                      1000.0     (True)
        Temperature                 120.0      (True)
        Cell length                 1.0        (False)
        Shift                       0.0        (False)
        Additional-Broadening       0.0        (False)
        Theta-0                     0.0        (False)
        LC Polarisation             50.0       (False)
        Doppler-Temperature         20.0       (False)
        Rb-85                       1.0        (False)
        K40                         1.0        (False)
        K41                         0.0        (False)
Using algorithm:Marquardt-Levenberg

Running Fit...  ...Fit Completed

Optimised parameters are (Fitted):
        Bfield                      1006.07129989(True)
        Temperature                 133.453173536(True)
        Cell length                 1.0        (False)
        Shift                       0.0        (False)
        Additional-Broadening       0.0        (False)
        Theta-0                     0.0        (False)
        LC Polarisation             50.0       (False)
        Doppler-Temperature         20.0       (False)
        Rb-85                       1.0        (False)
        K40                         1.0        (False)
        K41                         0.0        (False)
```

Select element and D-line

Element	Rb
D-line	D1
Constrain Dopp./Density Temps	✓

Initial parameters for fit

		Float?
Bfield [G]	1000.00	✓
Temperature [C]	120.00	✓
Cell length [mm]	1.00	☐
Shift [MHz]	0.00	☐
Additional-Broadening [MHz]	0.00	☐
Theta-0 [deg]	0.00	☐
LC Polarisation [%]	50.00	☐
Doppler-Temperature [C]	20.00	☐
Rb-85 [%]	1.00	☐
K40 [%]	1.00	☐
K41 [%]	0.00	☐

Fit Algorithm

[✓ Marquardt-Levenberg] Random-Restart Simulated Annealing

The three tabs next to the plot tabs are status information panels. They are all read-only.

The first, simply labelled ‘Status Panel’, simply redirects python’s normal stdout to a text box (stdout is where any ‘print’ statement is shown, usually in the command-line window where python is run from).

The second, labelled ‘Fitting Information’, shows information on the fits that have been run since the program started. An example is shown above. This panel details starting parameters, final (fitted) parameters and other information that may be useful, such as fit quality indicators (RMS error, chi-squared values, correlation reports), fit start time, the file and data type to be fitted etc. Note this is blank until a fit is run.

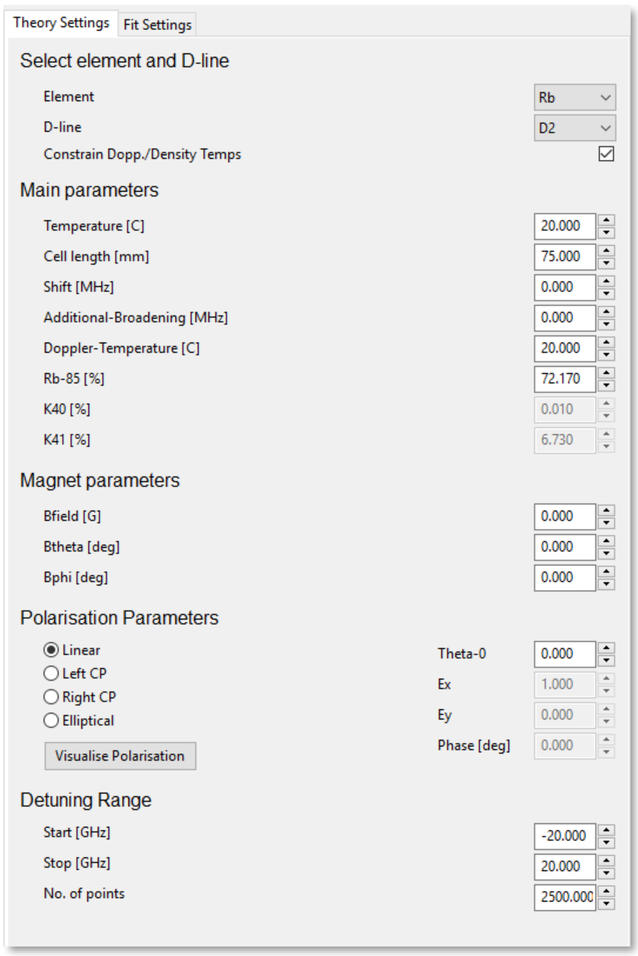
The final tab is labelled ‘Error Information’ and in an ideal world will be blank! However, this tab redirects pythons ‘stderr’ output, so any errors can be traced and debugged.

All three panels have a ‘Save text to file’ button at the bottom, which allows all the data contained in the text box to be saved as a .txt file. In addition, they support text highlighting, copy and paste etc.

c. *Theory/Fit settings tabs*

These tabs contain the parameters for calculating spectra / running fits.

> ***Note:*** The parameters can be quickly copied from the Theory Settings to Fit Settings and vice-versa via the buttons on the Edit Menu.



Select element and D-line

Element	Rb
D-line	D2
Constrain Dopp./Density Temps	✓

Main parameters

Temperature [C]	20.000
Cell length [mm]	75.000
Shift [MHz]	0.000
Additional-Broadening [MHz]	0.000
Doppler-Temperature [C]	20.000
Rb-85 [%]	72.170
K40 [%]	0.010
K41 [%]	6.730

Magnet parameters

Bfield [G]	0.000
Btheta [deg]	0.000
Bphi [deg]	0.000

Polarisation Parameters

[● Linear] ○ Left CP ○ Right CP ○ Elliptical

Theta-0	0.000
Ex	1.000
Ey	0.000
Phase [deg]	0.000

Detuning Range

Start [GHz]	-20.000
Stop [GHz]	20.000
No. of points	2500.000

Hovering over each parameter for a short time brings up a ToolTip describing in more detail what each parameter is. In brief, these parameters are:

Parameter	Description
Element	The alkali element to calculate spectra for
D-ilne	The D-line (D1, D2) to calculate spectra for
Constrain Doppler / Density Temperatures	Boolean option to use separate temperatures for calculating atomic number density and Doppler width, or constrain these two parameters (ticked, default)
Temperature	Temperature that determines the atomic number density (and Doppler temperature if Constrain is ticked).
Cell length	Thickness of the atomic vapour cell
Shift	Global shift of the lines, compared to the weighted line-centre frequency
Additional Broadening	An extra homogeneous broadening term that contributes to the Lorentzian line-width
Doppler Temperature	Temperature that sets the Doppler width (velocity distribution) of the atoms
Rb-85, K-40, K-41 %	Sets the isotopic weighting of Rb/K. Defaults to natural abundance ratios
Bfield	Strength of the magnetic field
Btheta, Bphi	Polar angles of the magnetic field vector – see the 2017 paper, fig 1, for the sign convention
Polarisation Options	Radio buttons select common polarisations, or the user can select Elliptical polarisation to manually input values of Ex, Ey, and the phase difference between the components
Detuning Range	Select the range of detuning, and the number of points, for the calculation

For more details on the parameters, see the ElecSus papers.

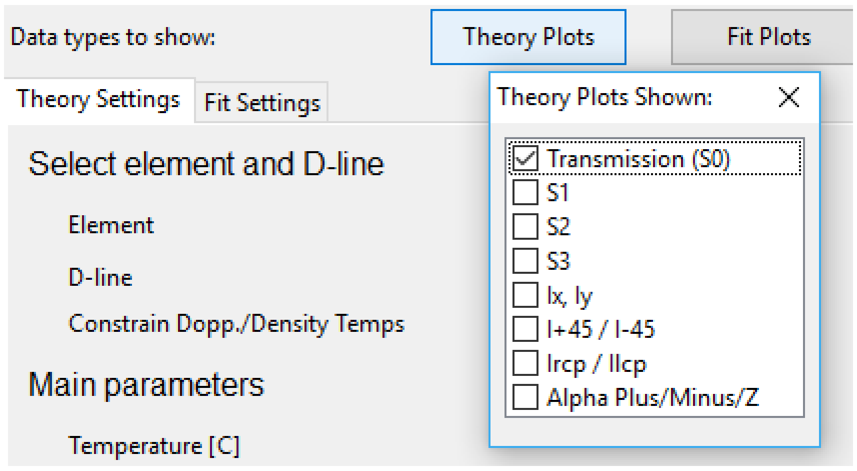
Fit parameter specific options are discussed in section 4.

3. Calculating a spectrum

a. Adding / removing plot types

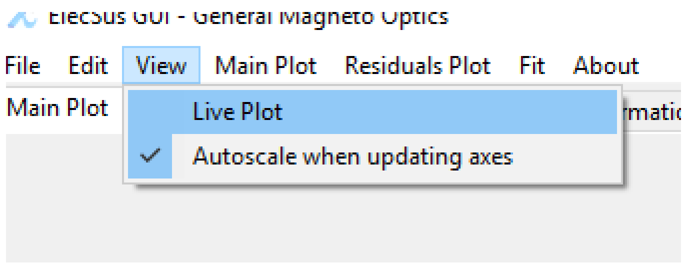
To actually calculate the spectrum, input the computation parameters that are required and click the 'Compute Spectrum' button at the bottom right of the window.

By default, only the transmission signal will be displayed (S0 in Stokes Parameter shorthand). However, more data can be displayed by clicking on the buttons next to 'Data types to show:', which will bring a popup window with check boxes for selecting the various data types to display. An example is shown below:



b. Live Plotting

Sometimes it is instructive to vary parameters to 'get a feel' for how spectral properties change. Instead of clicking 'Compute Spectrum' many times, in the 'View' Menu, click 'Live Plot', and the plot will update automatically as parameters are varied. Warning – if many points are to be calculated, this may cause the program to be unresponsive for long periods.


ElecSus GUI - General Magneto Optics
File
Edit
View
Main Plot
Residuals Plot
Fit
About
Main Plot
Live Plot
✓ Autoscale when updating axes

c. Quickly changing parameters

Each parameter in the list can be changed by clicking into the text control box and typing a new value, or alternatively using the either the spin control buttons (up/down arrows next to the control) or mouse wheel to increment the parameter values by a set amount. There are some shortcuts for these controls that make it easy to quickly spin through values:

(source: http://wxpython.org/Phoenix/docs/html/lib.agw.floatspin.html)

- The initial value is remembered - press Esc to return to it;
- Shift + arrow = 2 * increment (or Shift + mouse wheel);
- Ctrl + arrow = 10 * increment (or Ctrl + mouse wheel);
- Alt + arrow = 100 * increment (or Alt + mouse wheel);
- Combinations of Shift, Ctrl, Alt increment the FloatSpin value by the product of the factors;
- PgUp & PgDn = 10 * increment * the product of the Shift, Ctrl, Alt factors;
- Space sets the control's value to its last valid state.

4. Fitting experimental data

a. Importing data

Clicking on either the ‘Import Data’ button at the bottom-right of the main panel, or File->Open (Ctrl+O shortcut) brings up a dialog box asking which data type is to be imported.

> **The file should be in csv format, 2 columns with no header or footer lines. The detuning axis should be in GHz.**

In the ElecSusTestData GitHub repository, we provide many example data files; copy the format of these if anything is not clear.

Currently only one experimental trace per data type is allowed – if a second file of the same data type is imported it will simply overwrite the first.

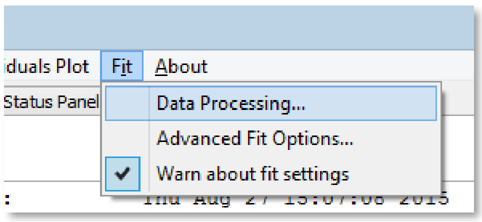
b. Data processing

Quite often with modern DPO oscilloscopes, data files have a large number of elements, typically in the range 10^4 to 10^5. Whilst there is in principle nothing preventing the program from using all these points, fitting will be very slow since the spectrum has to be evaluated at each of the points. There is also typically quite a bit of data redundancy – i.e. not all the recorded points are needed. A considerable speed-up can be gained by reducing the amount of data points by locally averaging the data (known as binning the data).

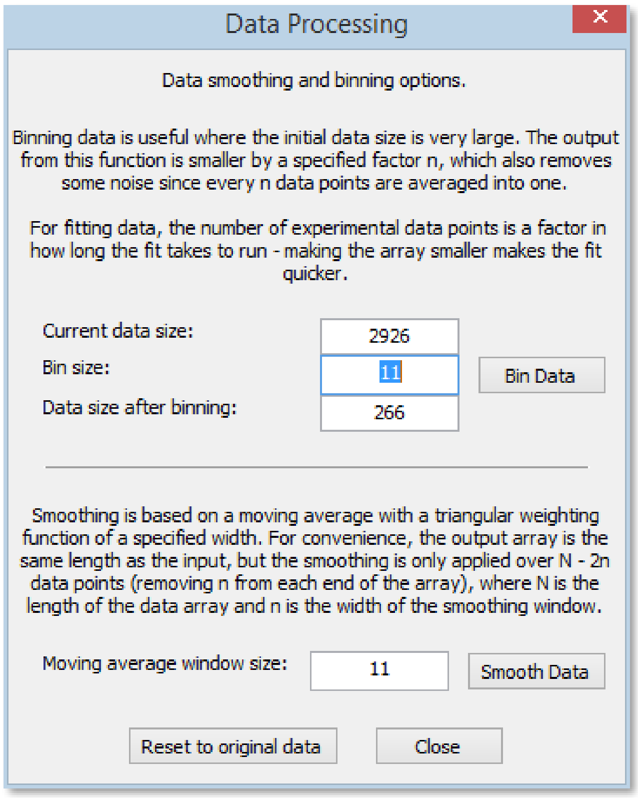
We recommend for reasonable speed using < 5000 data points, especially if the Random-Restart, Simulated Annealing or Differential Evolution fit algorithms are used. By default the program warns the user if the fit is expected to take a long time – these warnings can be turned off via the check-box button on the Fit menu, shown below.

Alternately if the data is particularly noisy but not a long array, the data may be smoothed using a moving-average method.

In the ElecSus GUI this is possible via the 'Data Processing' menu item, located in the Fit menu, shown below:



This will bring up a dialog box allowing both binning and smoothing to be done:


Data Processing

Data smoothing and binning options.

Binning data is useful where the initial data size is very large. The output from this function is smaller by a specified factor n, which also removes some noise since every n data points are averaged into one.

For fitting data, the number of experimental data points is a factor in how long the fit takes to run - making the array smaller makes the fit quicker.

Current data size: 2926

Bin size: 11

Data size after binning: 266

Smoothing is based on a moving average with a triangular weighting function of a specified width. For convenience, the output array is the same length as the input, but the smoothing is only applied over N - 2n data points (removing n from each end of the array), where N is the length of the data array and n is the width of the smoothing window.

Moving average window size: 11

The dialog shows the current data size and the size after binning. Note nothing happens unless either the ‘Bin Data’ or ‘Smooth Data’ buttons are pressed. The dialog can be dragged around and the result of binning/smoothing is shown immediately on the main plot.

The ‘Reset to original data’ button is fairly self-explanatory – it simply undoes any binning or smoothing while the dialog box is open. However, once the dialog box is closed, the new data arrays are stored and bringing up the Data Processing dialog box again will start from the now-processed arrays.

c. *Choice of fitting routine*

ElecSus provides 4 fitting routines, which vary in speed and complexity. See the 2015 paper for references to the exact algorithms used and more details, but here we quickly summarise their relative merits.

- Marquardt-Levenberg (ML) is the simplest fitting algorithm, and involves a simple downhill method to find a minimum in parameter space. Its simplicity makes it relatively fast, and a fit on a data set with around 5000 points takes around 30 seconds, depending on computer speed and choice of starting parameters. It runs on a single processing core. However, for large numbers of free parameters, it is known to find local minima and hence often returns non-optimal parameters. When the number of free parameters is more than 3, we recommend using either Random-Restart or Simulated Annealing instead.

- Differential evolution (DE) is a global optimisation algorithm that uses stochastic methods (not gradient methods) to find the global minimum. The algorithm iteratively combines (evolves) candidate solutions to find the global optimum. Tests have shown that this algorithm converges quicker than the Metropolis algorithm used by the simulated annealing fitting. See the compare_fit_methods() code in tests/fitting_tests.py for a worked example using ElecSus.

  More information can be found here:

  > Storn, R and Price, K, Differential Evolution - a Simple and Efficient Heuristic for Global Optimization over Continuous Spaces, Journal of Global Optimization, 1997, 11, 341 - 359.

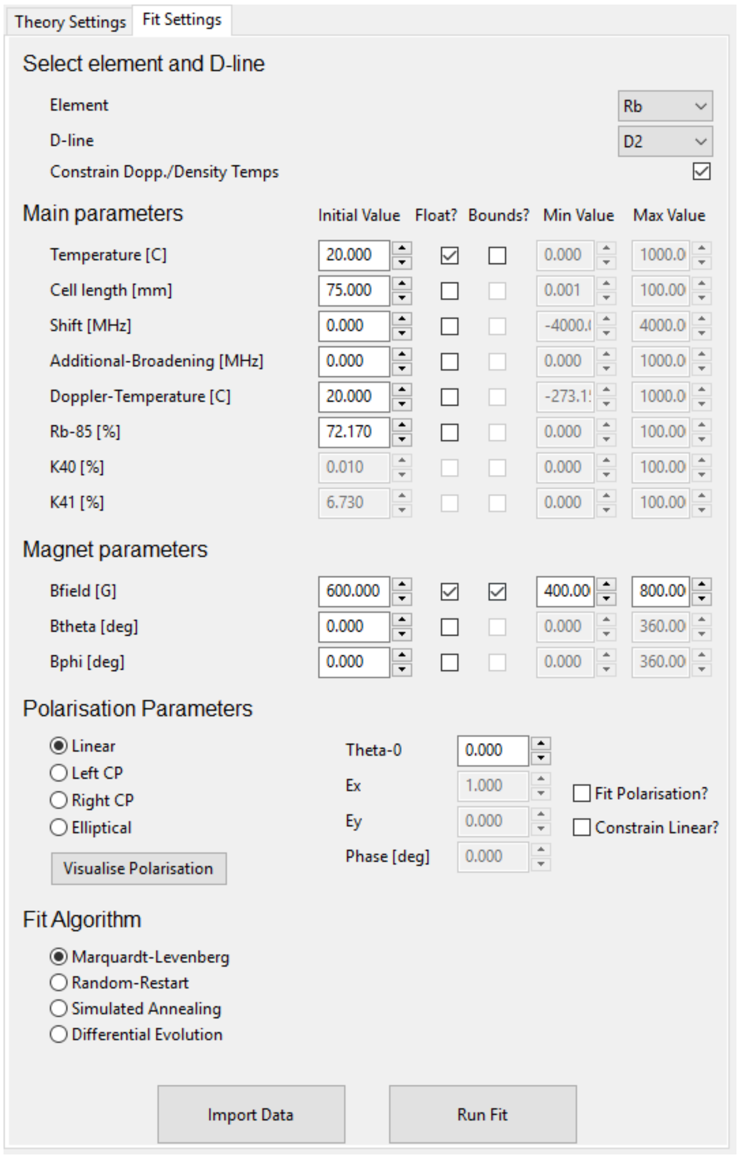
  Since DE converges both to a better solution and in less time than either RR or SA, it is now the recommended method for global minimum finding.

Old fit methods (these will be left in for now, but may be removed from future versions):

- Random-Restart (RR) is the next most complex algorithm. It uses all the computer's processing cores (by default) by utilising python's 'multiprocessing' module. Since the ML method is known to find local minima in parameter space, the RR method takes a spread of starting parameters, centred around the initial parameters. The ML method is called separately on each of these starting parameters in order to find a global minimum. The speed of this method depends on the number of processing cores and number of free parameters in the fit. Typically this method takes around 2-10 minutes.

- Simulated Annealing is the most computationally intensive algorithm used by ElecSus. However, it is likely to return the global minimum, assuming a sensible starting condition. This fit method is also capable of using all processor cores, but the method is slow, and should therefore only be used for the most complex fitting problems, where RR fails to find a good solution, or to check the results of an RR fit. Typical time to compute is around 10-30 minutes.

d. Fit parameter options



Theory Settings | Fit Settings

Select element and D-line

Element	Rb
D-line	D2
Constrain Dopp./Density Temps	☑

Main parameters	Initial Value	Float?	Bounds?	Min Value	Max Value
Temperature [C]	20.000	☑	☐	0.000	1000.0
Cell length [mm]	75.000	☐	☐	0.001	100.00
Shift [MHz]	0.000	☐	☐	-4000.	4000.0
Additional-Broadening [MHz]	0.000	☐	☐	0.000	1000.0
Doppler-Temperature [C]	20.000	☐	☐	-273.1	1000.0
Rb-85 [%]	72.170	☐	☐	0.000	100.00
K40 [%]	0.010	☐	☐	0.000	100.00
K41 [%]	6.730	☐	☐	0.000	100.00

Magnet parameters	Initial Value	Float?	Bounds?	Min Value	Max Value
Bfield [G]	600.000	☑	☑	400.00	800.00
Btheta [deg]	0.000	☐	☐	0.000	360.00
Bphi [deg]	0.000	☐	☐	0.000	360.00

Polarisation Parameters

● Linear
○ Left CP
○ Right CP
○ Elliptical

Theta-0	0.000
Ex	1.000
Ey	0.000
Phase [deg]	0.000

☐ Fit Polarisation?
☐ Constrain Linear?

Fit Algorithm

● Marquardt-Levenberg
○ Random-Restart
○ Simulated Annealing
○ Differential Evolution

The Fit settings tab shows a list of all the parameters that can be varied in a fit. These are the same as the calculation parameters detailed previously, but there are additional options for fitting. For each parameter, the user can specify whether this parameter is to be varied/floated during a fit, and whether to bound the values that the parameter can take (e.g. in an experiment the temperature was measured to be 100 degrees, but there may be some inaccuracy in the measurement, so the user can specify the temperature has to be in the range 90 – 110 degrees). This can speed up fitting, since less parameter space is explored, and can also prevent the program from returning unphysical values (e.g. a Doppler temperature < 0 K, cell length < 0 …).

The polarisation fitting option automatically constrains the values of Ex, Ey and Phase, so the user cannot enter custom bounds here. The additional 'Constrain Linear?' checkbox constrains the electric field to be linearly polarised, i.e. Phase = 0, while allowing the angle of polarisation to vary through Ex and Ey.

e. On fit completion

After the fit completes, details of the fit parameters and estimates of the errors are shown in the 'Fitting Information' tab, and the residuals plot on the 'Fit / Residuals Plot' tab is populated.

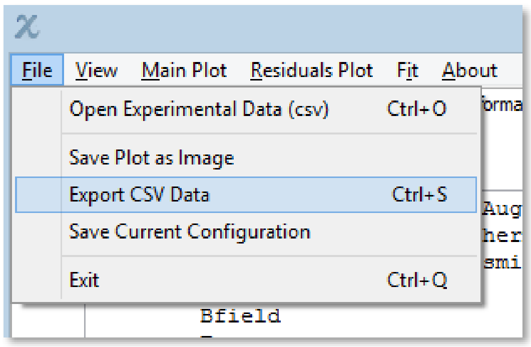
Optionally, the user may add a histogram of the residuals to the plot, via the Residuals Plot menu item. This can aid in determining how the scatter of data points is distributed.

f. Comparing fit methods

In the /tests/ subdirectory, the *fitting_tests.py* file provides a comparison of the 4 fitting methods, by generating a theoretical data set with some randomly distributed noise added. The fit starts from a known local minimum in parameter space, to demonstrate the limitations of the Marquardt-Levenburg method.

5. Saving results

Results can be saved in many forms – all from the File menu.



Save plot as image allows the user to save the current plots as image files. The file formats allowed are any format that matplotlib supports – currently that is png, ps, eps, pdf and svg. This does exactly the same job as the toolbar button to save the figure.

Export CSV Data exports the data from the main plot – all output types are saved, whether or not they are displayed. The csv file has a header line with the output types; the data is arranged in columns.

6. Reference

a. How to cite this work

This work should be cited using the ElecSus *Computer Physics Communications* publications. If this work has been valuable in your research, please add citations to both:

M. A. Zentile *et. al.*, Comp. Phys. Commun. **189**, 162 (2015)

and

J. Keaveney *et. al.*, arXiv.org 1708.05305 (2017)
(This will be updated when the article is published after peer-review)

b. *Requirements for running the GUI*

The GUI is written in wxPython, a python wrapper for the wxWidgets C++ library.

On windows, we recommend using Enthought Canopy – all packages required are included with the free (Express) version except lmfit, which can be found at:

https://lmfit.github.io/lmfit-py/

On Ubuntu Linux, Enthought does support wxPython, but requires that the 'glib' package is removed: see https://support.enthought.com/hc/en-us/articles/207229586-Unable-to-run-wxPython-3-0-x-on-Ubuntu-e-g-using-wx-as-matplotlib-backend- for details. Alternatively all required packages can be installed manually using apt-get install <package>, to the standard python installation (/usr/bin/python). The required packages on Ubuntu are:

```
python-numpy, python-scipy, python-matplotlib,
python-wxgtk3.0, python-wxtools, wx3.0-i18n,
python-lmfit, python-sympy, python-psutil
```

Other operating systems are expected to work but have not been tested.

To install, rather than running locally, in a terminal/command prompt window, navigate to the elecsus top directory with the setup.py file and type:

```
python setup.py install
```

After installation, elecsus can be used (and the GUI started) by simply importing it, either in scripts or from the python interpreter:

```
> python

>>>     import elecsus
>>>     elecsus.elecsus_gui.start()
```

or

```
>>>     from elecsus import elecsus_gui
>>>     elecsus_gui.start()
```