Line List Calibrator

Corrado Boeche

December 7, 2021

In this document we describe the graphical interface of the $\log gfs$ calibration software and how to use it. The interface allows the user to calibrate the $\log gfs$ and the van der Waals parameter in interactive mode, checking step-by-step the calibration and the results by visualizing the observed spectra of the standard stars against the synthetic ones derived from the line list under calibration. For a more complete understanding of this document, we recommend the user to read the paper *SP_Ace v1.4 and the new GCOG library for stellar parameters and elemental abundances derivation* by Boeche, Vallenari, and Lucatello 2021 (link here).

1 Installation

To run the calibrator, the following softwares must be installed:

- Python3 and the packets "numpy", "matplotlib", "pandas", "scipy", "sklearn", "tkinter".
- SPECTRUM 2.77 (http://www.appstate.edu/ grayro/spectrum/spectrum.html), in particular packets "spectrum", "lines", "macroturb", and "smooth2" must be put in a directory included in the system variable `$PATH`.

2 The codes

The software is composed by several files:

- `calibrator_routine.py`: this is the code that launch the $\log gfs$ calibrator graphic interface with the command *python calibrator_routine.py*. Its use is outlined in Sec. 3.
- `calibrator_classes.py`: it contains the classes *LineList* and *Star*[1] beside many functions.
- `calibrator_spectra.py`: it contains functions that operate on spectra.
- `calibrator_utils.py`: it contains several utilities and functions that initialize the software
- `calibrator_params.py`: this is important for the user because at the beginning of the file the user can edit several variables concerning the address of files such as linelists and standard star spectra in use. The variables are self-explanatory and comments are present in the code.

In general, the user can find explanations on the comments included in the codes.

[1] This class name would have been better named as "Spectra" but for historical reason we left the old name.

2.1 The code `calibrator_batch.py`

While the previous codes runs by launching `python calibrator_routine.py` with no particular action by the user, the code `calibrator_batch.py` runs as self-standing code and it calibrates the log gfs of a line list in batch mode (with no interaction with the user) and it is very useful to speed up the calibration process. It runs with the following keywords:

- `--wave_start` and `--wave_stop`: starting and ending wavelengths of the interval that the library has to cover. Please see the following text for more explanation.
- `--njobs`: number of thread run (for parallel computation)
- `--time_start_end`: here one set the initial and ending working time refer to the time of the day one wants the computer to run. This was set for shared computers and it is useful to avoid that our jobs run when such computer is used by others. As example, by giving `--time_start_end, '18,8'` the code will run from Monday to Friday between 6pm and 8am (night work) and will sleep during the day, while it run full time between Friday night and Monday morning (week-end). If the user want to change the time/days, he/she have to modify the function `check_working_time` found in the codes that has such keywork option.

Be aware that the code read and write into the line list file which is usually `llist_table` or any name that has been set as line list in the file `calibrator_params.py`.

The code is used to calibrate the log gfs in the given wavelength interval with no intervention by the user who, after the automated calibration, can verify the result by using the graphical interface run with `calibrator_routine.py`. The code runs by calling the command `python calibrator_batch.py` followed by the keywords, or by editing the keywords found at the bottom of the file `calibrator_batch.py` and then calling it with the command `python calibrator_batch.py` with no following keyword.

3 The calibration software

It is launched with the command `python calibrator_routine.py` and it has three tabs with names *calibrate* log gfs, *set abundances*, and *plot lines*. We describe them in the following subsections.

3.1 The log gf calibration tab

It is identified as tab with name *calibrate* log gfs (see Fig. 1). It is divided in eight frames with titles:

- *Wavelength window*: it is where the user sets the wavelength interval of interest. After any change of interval, the button *initialize* must be clicked. The button *previous* and *next* move the intervals backwards and forwards. The button *plot spectra* make a plot of the spectra with the current parameters. In the middle part of the frame there are the two options *full list* (default setting) and *pruned llist value* which has an input slot. With the option *full list* the user can see how the synthetic spectra look like when all the lines in the *calibrating line list* are used for the synthesis. When the option *pruned llist value* is set, the user can see how the synthetic spectra look like when only the lines with strengths larger than the input value given are used for the synthesis[2]. To date, this last option does not work due to an unidentified bug. However this does not affect the log gfs calibration.

[2]This was originally designed to evaluate the line strength limit to use for the GCOG library production. In fact, the weaker lines are drop. To date, such limit has been set as 0.03 and the pruning is done by the code `calibrator_EW_Senn2_parall.py`.

- *add VALD lines*: these are the original atomic parameters of the lines as given by VALD. The atomic parameters visible are taken from the file `VALD_4800-6860_norep.dat` set in the code `calibrator_params.py`. By clicking over one line (per time) the user selects the line he/she want to add to the calibrating line list frame. This is done by clicking on the *add* button.

- *add SPECTRUM lines*: like before, but for the line list provided with the code SPECTRUM "luke.dat" which contains lines from different sources.

- *add Kurucz hyperfine splitting lines*: like before, but for the Kurucz hyperfine splitting linelist.

- *add hfs by Sneden website*: hyperfine line splitting as given in the Sneden website. To date, the window shows only the centre if the line band for the elements Co, V, Mn, if any. One can add other elements by downloading from the Sneden website and, after adding a header, add the file to the directory "./llist_sneden/llists/".

- *add lines manually*: as before, but here one can add a line manually, paying attention to keep the same number of columns given in the default example.

- *guess the line with ML algorithm*: for unidentified lines we offer here a machine learning (ML) algorithm to guess the element, $\log gf$, low and high excitation potential of the unknown line. Because the ML algorithm (K-nearest neighbors) uses the calibrated line list as trainig data set, the user should first calibrate the whole line list (for a larger trainig set) and then use the guessing algorithm. This offers 5 possible guesses among which the user can choose. It is experimental and there is no guarantee of a right result. However it seems to provide reasonable results.

- *calibrating line list*: here is shown the line under calibration in the current wavelength interval chosen. The lines can be removed from this list by selecting it (with one click) and press the button *remove line from line list*. The selected line can be edited by clicking on the button below *from line list*. After the editing, the line is put back by pressing the button *to line list*.

The button *calibrate* starts the calibration of the lines showed in the *calibrating line list* frame. The result is shown in the calibrating line list window and in the plot window.
The button *quit* quit the software without writing anything. The button *write and quit* writes the newly calibrated atomic parameters in the file `llist_table` (which is a comma separated values (.csv) file) and then quit.

When restarted, the software look for a `llist_table` file. If this is found, it is uploaded (allowing the user to continue the previous work) otherwise the software stars from the original VALD atomic line list.

When the buttons *plot spectra* or *calibrate* are pressed, the graphic window shows a plot like in Fig. 2. The colored lines represent the observed spectra, the gray thick dashed lines represent the synthetic spectra with VALD atomic parameters, the black thin dashed lines represent the synthetic spectra with the calibrated atomic parameters.

3.2 The abundance calibration tab

This window (see Fig. 3) has two frames:

- *elements to plot*: first, click on the button *show elements*, then choose up to 4 elements by clicking on them. The button *plot NEWRs* plots the NEWRs (Normalized Equivalent Width Residuals) as a function of their EW in the graphical window. The plot show the EWs distribution for each of the 5 standard stars (see Fig. 4). The user can set the *minimum purity* value of the lines to be shown in the plot[3]. Clicking on the

[3]The definition of "purity" has been given in Sec.3.4 of the paper Boeche, Vallenari, Lucatello, submitted 2020.

button *plot params check (MOOG like)* the user can visualize the NEWRs of a choosen element as a function of excitation potential, logarithm of the reduced EW, and wavelength λ, similarly to what is done in MOOG (see Fig. 5).

- *abundances to change*: here one can set the abundances for every standard star. After the abundance setting the button *recompute NEWRs* must be pressed in order to make the changes effective. After this, to see the changes, press the button *plot NEWRs* again. **Important suggestion**: if the wavelength interval in calibration is large, the *recompute NEWRs* button may take long time to return because it synthesizes the whole wavelength interval for all the standard stars. A more convenient way to do the same is teh following: set the elements abundance and press the button *write the abd files* which only writes the files where the abundances for the synthesis are stored. After that, run the `calibrator_batch.py` which automatically recalibrate the log gfs with the newly set abundances. Then, check the results with `calibrator_routine.py`.

  The user can add elements by editing the files atmospheres/star?.abd and adding the wanted elements.

3.3 The plot lines tab

With this interface the user can visualize a specific line and, when desired, calibrate the Van der Waals broadening parameter. It contains two frames:

- *element lines to plot*: this is a useful tool to eye-check the lines by element. Press the button *show elements* to see the element list and select one element per time. Then press *initialize* and *plot* to see in the graphical window three lines of the element per time. To move through all the lines of the chosen element, press the buttons *next* or *previous* and the *plot*. The *wavelength* entry takes as input the starting wavelength of the lines to be visualized. As example, by putting 6000 only lines with wavelengths larger than 6000Å will be proposed. This is useful when there are many lines (thousands in the case of Fe) and one want to move quickly to the redder region. With the entry *minimum EW* one can select the minimum EW that the lines shown must have. Similarly does the *minim purity* entry.

- *calibrate Van der Waals parameter*: the window shows the atomic parameters of the three lines plotted in the graphical window. By clicking on one of them, and then on the button *calibrate VdW*, the Van der Waals fudge factor parameter given in SPECTRUM will be automatically adjusted to match at best the line (using a χ square minimization process). By pressing *initialize* the newly computed VdW value will be shown and by pressing *plot* the synthetic line with the new VdW value will be plotted. The button *undo VdW* the VdW value is reported to its original value.
  If the user want to set the VdW value by hand, he/she can use the log gf calibration interface and use the buttons *from line list* and *to line list* to edit any atomic parameter of the line.

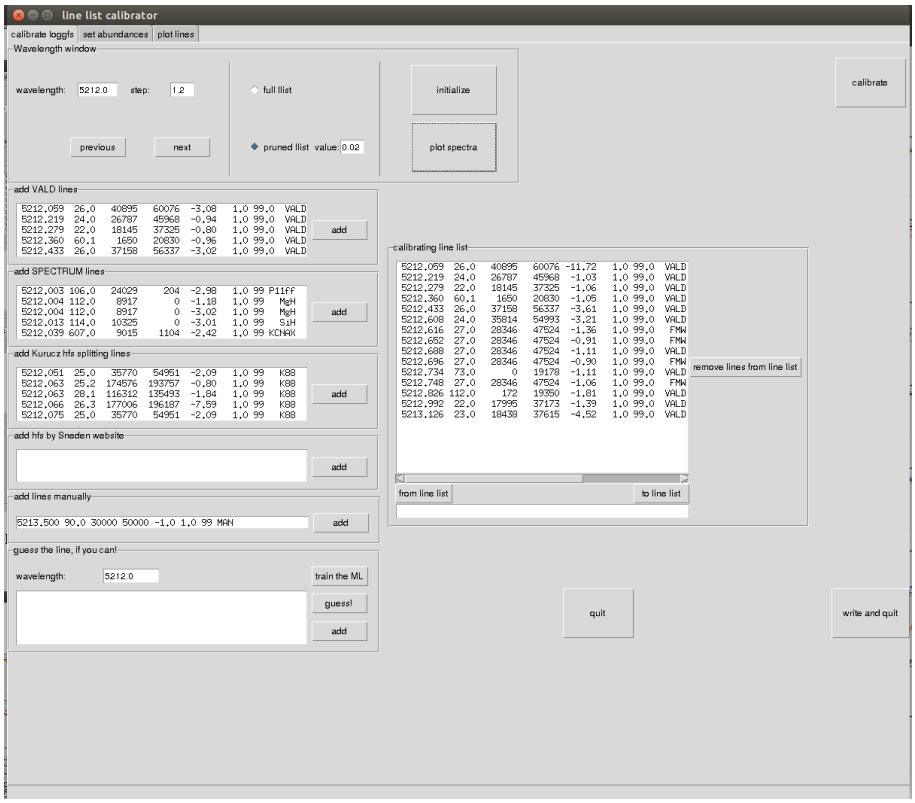

Figure 1: The oscillator strength ($\log gf$)s interface.

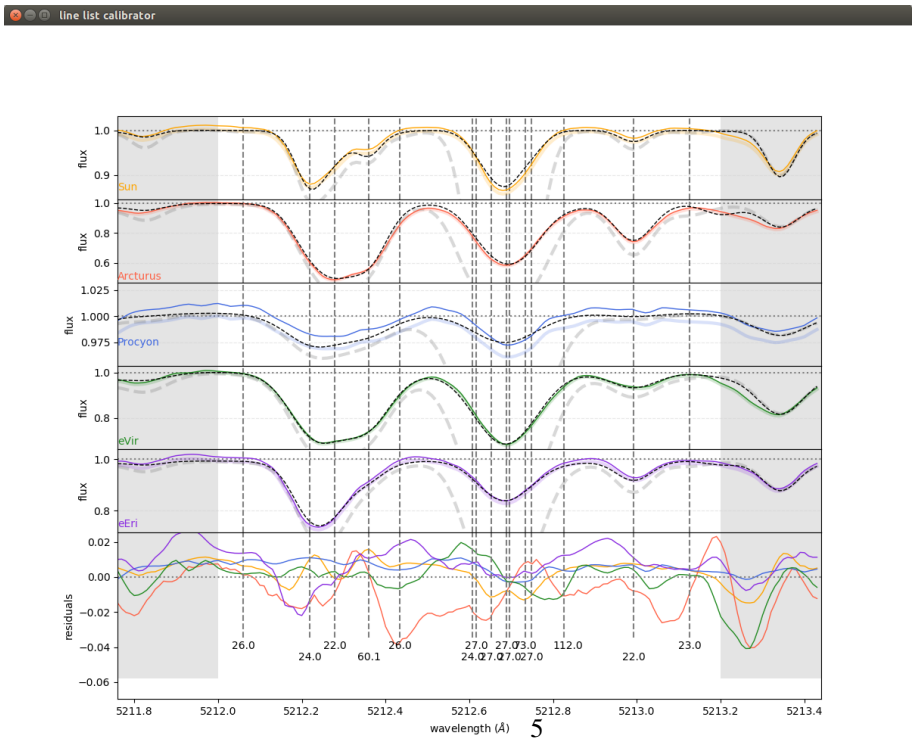

Figure 2: Plot window of loggfs calibration.

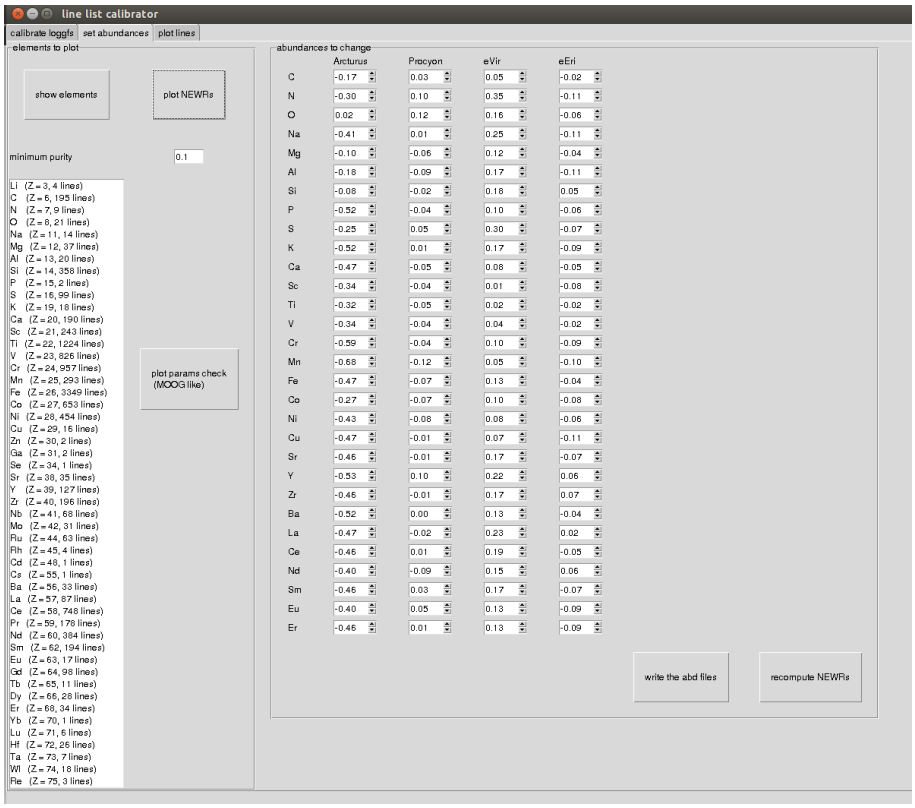

Figure 3: The abundances interface.

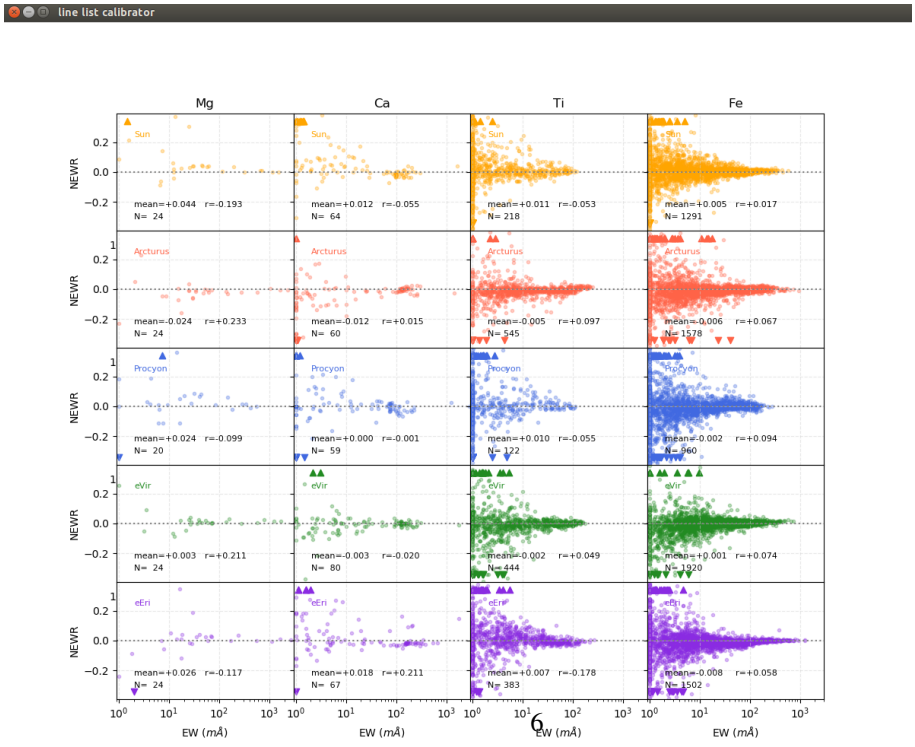

Figure 4: The abundances plot window.

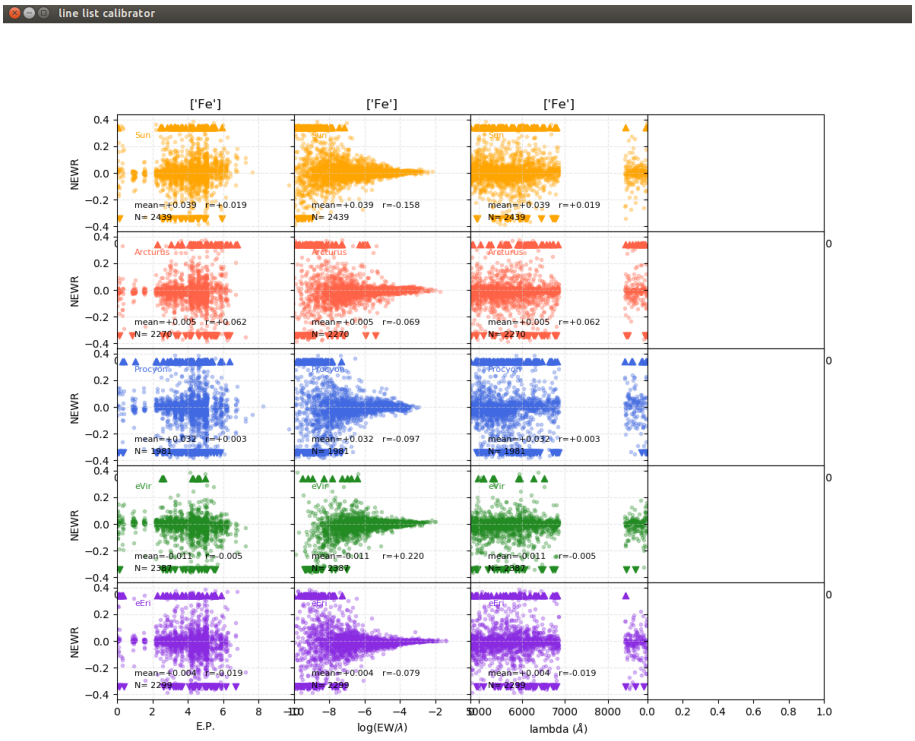

Figure 5: The abundances plot window as MOOG would shows.

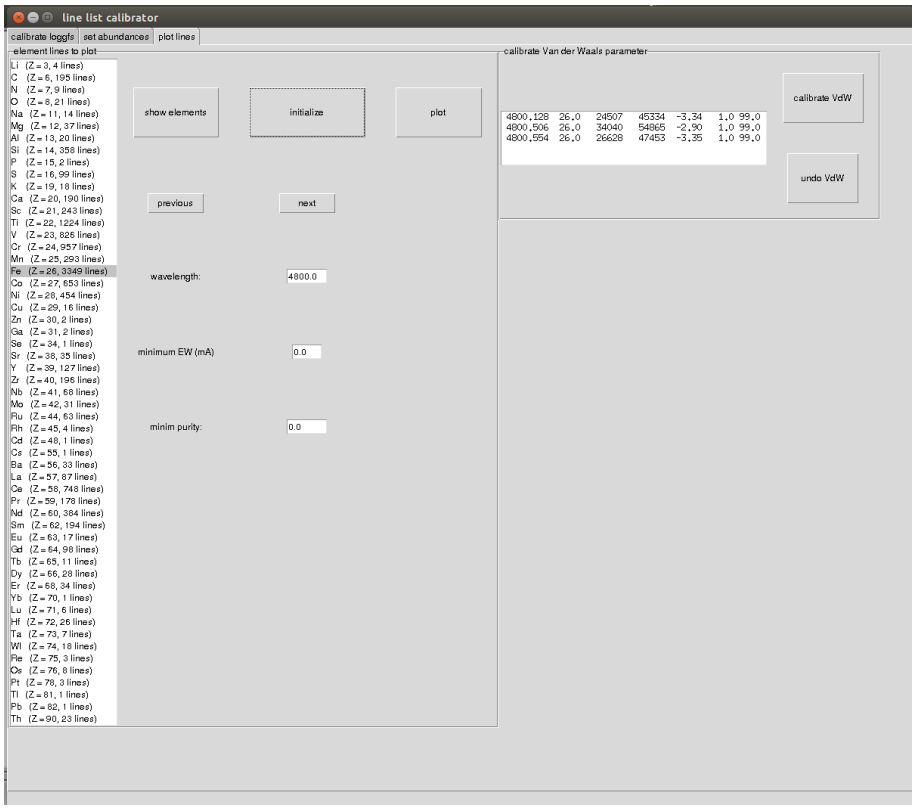

Figure 6: The plot lines interface.

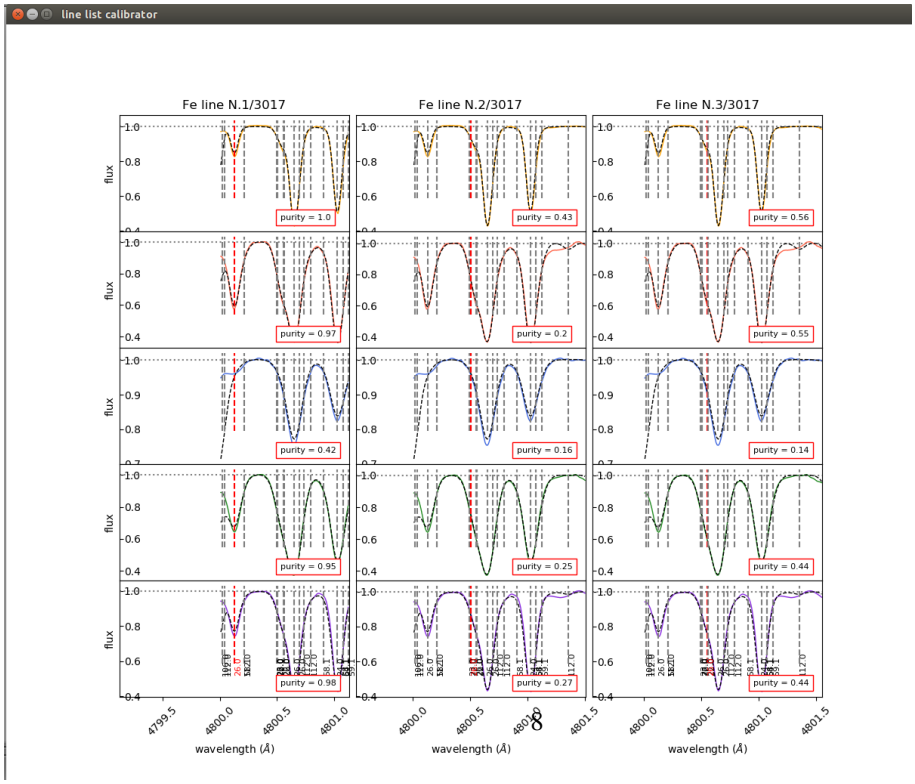

Figure 7: The plot lines interface.