SLIM Lines Plotter
(slimlinesplot.py)
User Guide

Andrés Megías Toledano
Centre for Astrobiology (CAB), Madrid

Version 1.6
May 2023

Index

1. Introduction

The SLIM Lines Plotter is a Python 3 script, slimlinesplot.py, that can create plot of several transition lines exported by the tool SLIM from the software MADCUBA.

The script requires the libraries PyYAML, NumPy, SciPy, and Matplotlib. In order to run the code, it is necessary to specify some options in a configuration file. For running the code we have to write the following command line in the terminal, being in the folder of the file slimlinesplot.py:

```
python3 slimlinesplot.py <path>
```

The argument <path> is the path of the configuration file. If we do not specify it, the script will search in the current folder (./) for the configuration file specified inside the script in the variable config_file, at the beginning of the code; we can use this variable to run the script in a programming environment like Spyder. If the script linesbarplots.py is not in the current folder, but in a folder with path <folder>, we should write <folder>/slimlinesplot.py instead of slimlinesplot.py. Moreover, if the script is included in a folder of executable files, the term python3 can be removed from the command line, and the script will be run from any path.

2. Input files

As an input we need the files exported by MADCUBA for each of the transitions that we want to plot.

First of all, we need at least one folder in which to store all the data. For each data folder, a single image will be created. Then, inside each of this folder, we need one folder for each species that we want to plot. Now, inside of each species folder, we must storage the data exported by MADCUBA.

To do so, select one species in SLIM, in the SimFit. Change the velocity range to the desired one. Then, in the plots window, go to the menu, to File, then click Save Python, and select the data folder (the one that will contain a folder for each species). Then, a folder with the same name as the parent folder will be created; rename it to the name of the molecule. Inside it, there will be two folders: data and spectroscopy.

The folder data contains plain text files for each of the plots that were showed in the MADCUBA plots window; their columns are, respectively, the velocity (in km/s), and the observed intensity, the modelled intensity of the fitted molecule, and the modelled intensity with all the fitted species (all these three quantities in K).

The folder spectroscopy contains two files for each of the previously mentioned spectra files, which contains a row for each transition located in the corresponding plot; the file containing the molecule in its name stores the transitions of only that molecule, while the one containing TRANSITIONS_ALL in its name stores the transitions of all the molecules present in the plot range. The most important columns of these files are the central velocity of the transition, the name of the molecule, the transition quantum numbers, the intensity peak of the transition, and its rest frequency.

3. Output files

For each of the data folders specified in the configuration file, an image containing all the specified plots will be saved in PDF format (.pdf).

4. Configuration file

The configuration file is a plain text document with the YAML formatting style. This way, we can specify the options of the script slimlinesplot.py by defining different variables that can be nested.

In order to define the values of the variables, we have to write, in a single line, the name of the variable followed by a colon (:), and its value, separated with a space. For the logical variables, the possible values are yes/no. There are variables whose value can be a list of elements, in which case they can be written between brackets and separated by commas or in single lines preceded by a hyphen (–). Similarly, in the case of nested variables, which work like Python dictionaries, next level variables must be written in a different line and with an indent. Text variables can be written between simple quotation marks ('), but this is optional.

If the definition of a variable is not written,[1] its default value will be used (see Section 3).

Below is a list of all the variables with their meaning and possible values:

- **figure titles.** List with the titles of the different figures that will be created.
- **data folders.** List with the folder names which contain data to be plotted. For each folder, a figure with subplots will be generated. This can be used for plotting a different source on each figure.
- **species**. List with the name of the molecules whose lines will be plotted. Each element of the list should be the name of the folder, within the data folder, which contains the spectrum exported by MADCUBA for each molecule. That is, inside the folder for each figure should be a number of folders, one of them for each molecule that will be showed. Then, inside each of them, there should be the data exported from MADCUBA. Each element of this list must be a nested variable (a dictionary), with the name of the folder with the specific spectrum to be plotted. The next level variables of it are:
  - **file**. Beginning of the file name of the spectrum to be plotted. The files generated by MADCUBA start with a two-digit number, so we can just write these number.
  - **fit**. Logical variable that determines if the model spectrum is plotted. If it is not specified, its value is set to no. If there is more than one figure to be plotted (the variable folders has more than one element), this variable can be a list with a logical value for the corresponding subplot of each figure.
  - **lines of the fit**. Variable used to plot the specified fit lines instead of the fit contained in the corresponding data file. It should be a dictionary containing the information of the lines to be plotted; if there is more than one data folder, this variable should be a list containing one dictionary with the lines for each folder. These dictionary must have the following entries:
    - **intensity**. List of the intensities of the lines to be plotted, in the same units as the spectra.
    - **width**. List of the widths (full-width half maximum, FWHM) of the lines to be plotted, in the same units as the spectra.
    - **position**. List of the positions of the centers of the lines to be plotted, in the same units as the spectra. This is actually optional if there is only one line.

    For all of these three entries, if there is only one line you can just put the value itself. This option can be useful for plotting the results of codes like RADEX, which can be easily used through the RADEX Line Fitter.
  - **fit color**. Color to plot the model spectrum, from Matplotlib. This is an optional variable; if it is not specified, the value of the same-name variable of the configuration file (see next pages) will be used.
  - **title**. Text that will be written in the left upper corner of the subplot. You can

1. Here, 'not written' means that neither the variable name nor its value are written in the configuration file. If the variable name is written (and the colon) but its value is not, the corresponding Python variable will get the value None.

write species names between asterisks (*) so that they will be formatted correctly (isolated numbers will be written as subscripts, and numbers preceded by a hat symbol (^) or enclosed by brackets will be written as superscripts. Moreover, you can write LaTeX formatted text written between dollar symbols ($), and you can also use \n for setting a line break. This variable is optional.

  - **velocity limits**. Limits in the horizontal axis for the subplot, that is, in radial velocity, in the units of the data files (by default, km/s). It should be a list with two values (inferior and superior limits). This is an optional variable that can be used to specify the limits on the horizontal axis for each subplot. If it is not specified, the value of the same-name variable of the configuration file (see next pages) will be used. It the variable `join subplots` is set to `yes` (see next pages), this will only work for the subplots of the first row of the figure.
  - **frequency limits**. Limits in the horizontal axis for the subplot in the case that the frequency of the spectra is shown instead of the radial velocity; if radial velocity is in km/s, this will be in GHz. It should be a list with two values (inferior and superior limits). This is an optional variable that can be used to specify the limits on the horizontal axis for each subplot. If it is not specified, the value of the same-name variable of the configuration file (see next pages) will be used. It the variable `join subplots` is set to `yes` (see next pages), this will only work for the subplots of the first row of the figure.
  - **intensity limits**. Limits in the vertical axis for the subplot, that is, in line intensity, in the same units as the data files (by default, K). It should be a list with two values (inferior and superior limits). This is an optional variable that can be used to specify the limits on the vertical axis for each subplot. If it is not specified, the value of the same-name variable of the configuration file (see next pages) will be used. It the variable `join subplots` is set to `yes` (see next pages), this will only work for the subplots of the last column of the figure.

- **save figure**. Logical variable that determines if the figures are saved into `.pdf` files. The names of the files will be those stored in the variable `figure titles`. If a name is actually empty (`''`), the file name will be simply `lines.pdf`.
- **rows**. Number of rows of the lines plot.
- **columns**. Number of columns of the lines plot.
- **figure size**. Dimensions of the window containing the plot (width × height), in inches (”). It should be a list containing the width and the height, in this order. If its value is set to `'auto'`, the dimensions of the window will be $4” \cdot n_{\text{cols}} \times 3” \cdot n_{\text{rows}}$, where n_{cols} and n_{rows} are the number of columns and rows, respectively.
- **font size**. Size of the font, in points, used for the axis text of the plots.
- **label font size**. Size of the font, in points, used for the text inside each subplot. By default, it will be the 90 % of `font size`.
- **frame width**. Width of the lines of the plot frames, with respect to the default one in

Matplotlib.

- **plot line width**. Width of the lines of the plot curves, in points.
- **fit line width**. Width of the lines of the plot curves of the fits, in points. By default, it is the same as plot line width.
- **join subplots**. Logical variable that determines if the subplots will have or not any spacing between them. If set to yes, there will be no spacing, and the horizontal and vertical axis for each column and row will be shared, respectively. It can only be used if use frequency is set to no.
- **show transitions**. Logical variable that determines if the transitions specified in the files of the folder spectroscopy for each subplot are marked with a vertical gray line. It it is set to yes, there are some more variables that can be added to specify the texts of the transitions and which transitions are shown:
  - **transitions threshold**. Minimum value of the theoretical peak intensity that the transition must have to be shown in each plot, in the same units as the input data (K). By default ('auto'), it will be different for each subplot, being ⅒ of the maximum intensity of the plotted data within the intensity range of the subplot. Also, this variable can be manually specified for a particular subplot within the variable species.
  - **show quantum numbers**. Logical variable that determines if the quantum numbers are shown for each transition, as specified in the files of the folder spectroscopy.
  - **show main species transitions**. Logical variable that determines if the transitions of the main species of each folder are shown in each subplot.
  - **show rest species transitions**. Logical variable that determines if the transitions of the rest of species different than the main one of each folder are shown in each subplot.
  - **mark transitions with lines**. Logical variable that determines if the frequency of each transition is marked with a vertical dashed line.
- **show all species fit**. Logical variable that determines if the all species fit from MADCUBA is also plotted when the individual molecular fit is set to be plotted through the variable fit within the variable species. In this case, for each folder containing the data for a species, there should be another folder with the same name but ending with _all and containing the modelled intensity taking into account all the species from the SLIM file.
- **transition labels minimum distance**. If show transitions is set to true, this determines the minimum horizontal distance between two transitions to merge the vertical lines into one. Then, the line will be split at the end into different lines to mark the different transitions with different labels. The distance units are relative to the frame size.
- **horizontal axis**. Text variable that determines if the frequency is used in the horizontal axes ('frequency') of the plots instead of the radial velocity ('velocity'). By default it is 'velocity'.

- **velocity limits**. Limits in the horizontal axis, that is, in radial velocity, for the different columns in the plot, in the same units as the data files (by default, km/s). It should be a list with a list for each column (inferior and superior limits). If one of the values for a certain column is set to 'auto', the limits will be the range of the data. If this variable is not specified, all of the limits will be set to 'auto'. This variable can be specified for a particular subplot within the variable species.
- **frequency limits**. Limits in the horizontal axis for the different columns in the plot in the case that the frequency of the spectra is shown instead of the radial velocity; if the velocity is in km/s, this will be in GHz. It should be a list with a list for each column (inferior and superior limits). If one of the values for a certain column is set to 'auto', the limits will be the range of the data. If this variable is not specified, all of the limits will be set to 'auto'. This variable can be specified for a particular subplot within the variable species.
- **intensity limits**. Limits in the vertical axis, that is, in line intensity, for the different rows in the plot, in the same units as the data files (by default, K). It should be a list with a list for each row (inferior and superior limits). If one of the values for a certain row is set to 'auto', the limits will be the range of the data. If this variable is not specified, all of the limits will be set to 'auto'. This variable can be specified for a particular subplot within the variable species.
- **spectral factor**. Numeric factor that will multiply the horizontal variable (velocity or frequency). If it is 1000 and no horizontal label is provided, the units displayed (km/s or GHz) will be automatically corrected (to m/s and MHz, respectively), but only in this case.
- **intensity factor**. Numeric factor that will multiply the horizontal variable (velocity or frequency). If it is 1000 and no horizontal label is provided, the units displayed (K) will be automatically corrected (to mK), but only in this case.
- **velocity label**. Text to be displayed in the horizontal axis.
- **frequency label**. Text to be displayed in the horizontal axis in case that frequency is used instead of radial velocity.
- **intensity label**. Text to be displayed in the vertical axis.
- **use common labels**. Logical variable that determines if the horizontal and vertical labels are displayed just once for the whole subplot or they are shown for every subplot.
- **title height**. Position of the text of the title. The units are relative to the height of each subplot frame, so that 0 is the bottom and 1 is the top.
- **subplot titles height**. Position of the of the text of the subplots specified in the variables title, for each element of the variable species. The units are relative to the height of each subplot frame, so that 0 is the bottom and 1 is the top.
- **fit plot style**. This is a text variable which controls the way in which the model spectra are plotted. It can get the following values: steps, lines, and curve. With the style steps, the points of the model will be plotted as a step plot (similar to a histogram),

with the values centred on the top of the steps. With the style lines, the model points will be connected with lines. Finally, with the style curve, the script will interpolate the model points with a quadratic interpolation and plot the resulting curve. In this case, one can add one more variable:

  - **gaussian fit**. If this logical variable is set to yes, the script will try to interpolate the model points with gaussian curves supersampled to a much precise velocity resolution, and plot the resulting fit, so that we can see the actual intensity density distribution of the line, that is, the line profile.[2] This option is only useful when the width of the lines is only a few channels.

- **fit color**. Color to plot the model individual spectra for each species, from Matplotlib. This variable can be specified for a particular subplot within the variable species.
- **all species fit color**. Color to plot the global model spectra for all the species, from Matplotlib.
- **other species color**. Color of the text of the transition lines from molecules other than the main one. This only works if the variables show transitions and show all species are set to yes.
- **fill spectrum**. Logical variable that determines if the observed spectrum is filled with gray color from the zero line until the step curve.
- **ticks direction**. Direction of the ticks of the axis. Possible values are 'in' and 'out'.
- **species acronyms**. Dictionary containing a set of acronyms for the species names, that will be used in case that show transitions and show all species transitions are set to yes. Each element of the dictionary should be a name that one wants to be changed, and the value of the entry should be the name that will replace it.

Default variable values

Below are the default values of the variables of slimlinesplot.py, that is, the values that they will take if they are not declared in the configuration file.

```
figure titles: []
data folders: []
species: []
save figure: yes
rows: 1
columns: 1
figure size: 'auto'
font size: 10
label font size: null
frame width: 0.8
plot line width: 1.2
fit line width: null
join subplots: yes
show transitions: no
```

2. Note that, in this case, the line profile does not necessary have to have the same values than the data points of the spectrum, as the data points consists only of a limited number of channels; in other words, the observed points of the spectrum have units of intensity (K by default) but the line profile has units of intensity over velocity (K/(km/s) by default) or intensity over frequency (K/GHz by default).

```
show quantum numbers: no
show main species transitions: yes
show rest species transitions: yes
mark transitions with lines: yes
show all species fit: no
label minimum distance: 0.05
horizontal axis: 'velocity'
velocity limits: []
frequency limits = []
intensity limits: []
spectral factor: 1000
intensity factor: 1000
transitions threshold: 'auto'
velocity label: 'velocity (km/s)'
frequency label: 'frequency (GHz)'
intensity label: 'intensity (K)'
use common labels: no
title height: 0.92
subplot titles height: 0.90
fit plot style: 'steps'
gaussian fit: no
fit color: 'tab:red'
all species fit color: 'tab:blue'
other species color: 'tab:blue'
fill spectrum: no
ticks direction: 'in'
species acronyms: {}
```

Note that this default values will only be used if neither the variable name nor its value are written in the configuration file. If the variable name is written (and the colon) but its value is not, the corresponding Python variable will get the value `None`.

5. Example case

Here we show a brief example to see how to use the script `slimlinesplot.py`. It consist of some transition lines observed in the starless core L1517B, observed in two different positions.

5.1. Input files

For this example, we should have two folders with the data, named `L1517B` and `L1517BOFF1`. Then, inside each of them, we need 9 folders, one for each species to be plotted (and with names `CH3OCHO`, `CH3OCH3`, … `HCCCN`). Those are the folders that MADCUBA can create, so inside each of one there will be two folders: `data/`, which contains the spectra to be plotted, and `spectroscopy/`, which contains information about each transition (see page 2).

5.2. Configuration file

Below is an example of configuration file to plot some transition lines from the starless core L1517B. There are two different positions observed for this source, so two set of plots will be generated.

```
figure titles:
- 'L1517B - dust peak'
```

```
- 'L1517B - methanol peak'

folders:
- 'L1517B'
- 'L1517BOFF1'

species:
- CH3OCHO:
    file: '05'
    fit: no
    title: "*CH3OCHO* A \n $7_{2,6}$ → $6_{2,5}$"
- CH3OCH3:
    file: '01'
    fit: no
    title: "*CH3OCH3* \n $3_{1,3}$ → $2_{0,2}$ AA \n $3_{1,3}$ → $2_{0,2}$ EE"
- CH3CHO:
    file: '02'
    fit: [no, yes]
    title: "*CH3CHO* A \n $5_{0,5}$ → $4_{0,4}$"
- H2CCO:
    file: ['03', '02']
    fit: yes
    title: "*H2CCO* \n $5_{1,5}$ → $4_{1,4}$"
- CH3CN:
    file: '02'
    fit: [yes, no]
    fit color: 'tab:orange'
    title: "*CH3CN* \n 6 → 5"
    lines of the fit:
        intensity: 0.05673
        width: 0.280
- HCCNC:
    file: '01'
    fit: yes
    title: "*HCCNC* \n 8 → 7"
- CH3OH-A:
    file: '01'
    fit: yes
    fit color: 'tab:orange'
    title: "*CH3OH* A \n $2_{0,2}$ → $1_{0,1}$"
    lines of the fit:
        - intensity: 0.6872
          width: 0.283
        - intensity: 0.8001
          width: 0.283
- CH3OH-E:
    file: '01'
    fit: yes
    fit color: 'tab:orange'
    title: "*CH3OH* E \n $2_{1,2}$ → $1_{1,1}$"
    lines of the fit:
        - intensity: '0.525'
          width: '0.283'
        - intensity: '0.635'
          width: '0.283'
- HCCCN:
    file: '01'
    fit: yes
    fit color: 'tab:orange'
```

```
    title: "*HCCCN* \n 9 → 8"
    intensity limits: [-0.4, 2.4]
    lines of the fit:
        - intensity: 1.646
          width: 0.351
        - intensity: 0.852
          width: 0.359
rows: 3
columns: 3
figure size: [9, 6]
font size: 8
title height: 0.95
subplot titles height: 0.92

velocity limits: [[1, 9], [1, 9], [1, 9]]
intensity limits: [[-0.04, 0.08], [-0.04, 0.08], [-0.15, 0.9]]
fit plot style: 'curve'
gaussian fit: yes
show transitions: yes
```

5.3. Lines plot

Figure 1 shows one of the two resulting lines plots for this example. In this case the labels have been manually written manually for each line. In case the line is present, it is also shown the LTE model fitted by MADCUBA or the non-LTE model fitted by RADEX. Note that the

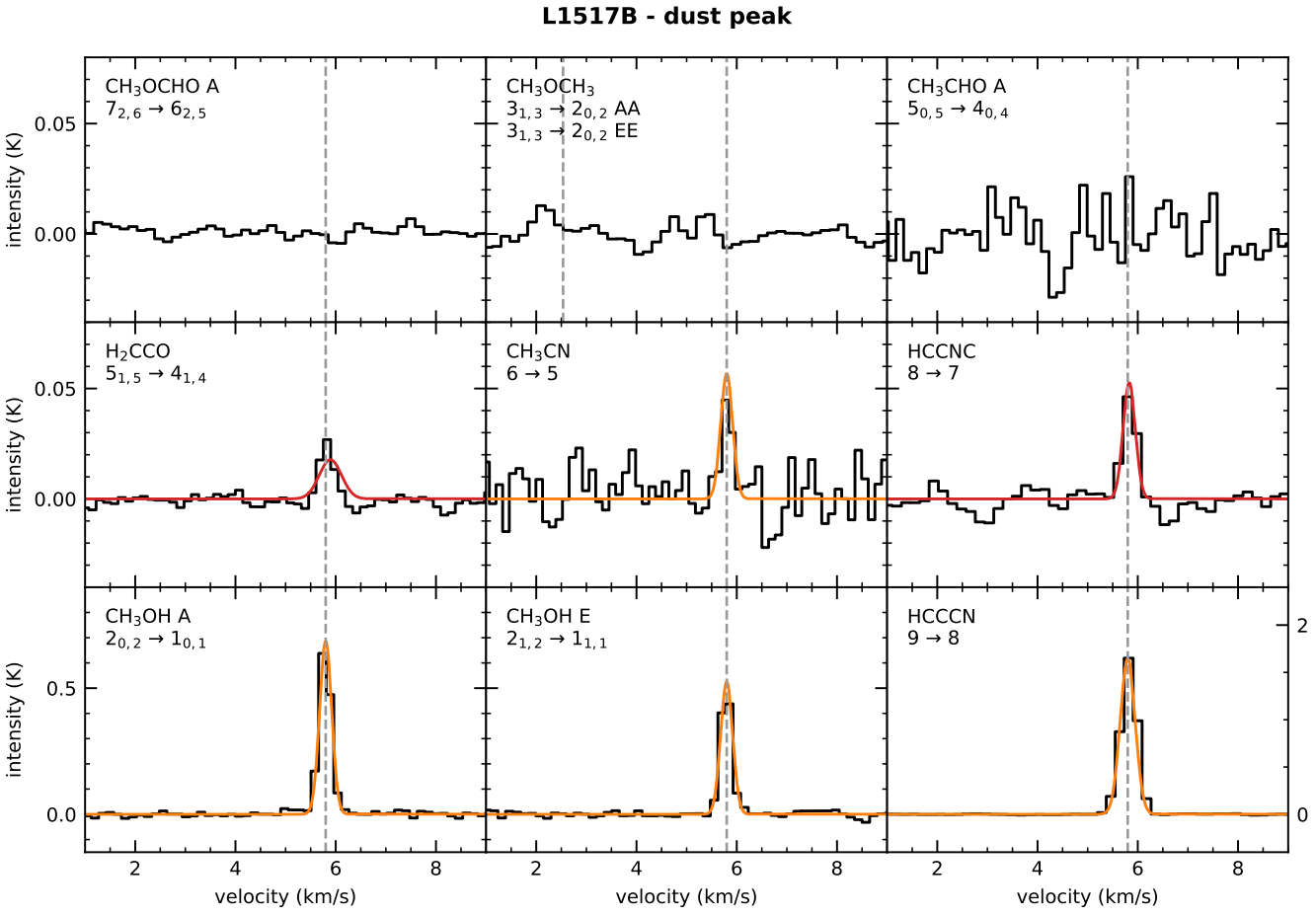

Figure 1. Plot of the selected transition lines of the example configuration file, corresponding to the center of the starless core L1517B.

plot on the lower right corner has different intensity limits, as specified in the configuration file.

Useful links

The following links may be of interest:

- **MADCUBA.**
  https://cab.inta-csic.es/madcuba/
- **MADCUBA-SLIM Python Scripts.**
  https://github.com/andresmegias/madcuba-slim-scripts
- **Python.**
  https://www.python.org/
- **PyYAML.**
  https://pyyaml.org/wiki/PyYAMLDocumentation/
- **NumPy.**
  https://numpy.org/
- **SciPy.**
  https://scipy.org/
- **Matplotlib.**
  https://matplotlib.org/
- **RADEX Line Fitter**.
  https://github.com/andresmegias/radex-python/

Credits

This software has been developed at the Centre for Astrobiology (*Centro de Astrobiología*, CAB), in Madrid (Spain), within the group of Chemical Complexity in the Interstellar Medium and Star Formation (Department of Astrophysics).

Coding and testing

Andrés Megías Toledano

Testing and discussion

Álvaro López Gallifa

Supervising

Izaskun Jiménez Serra
Víctor M. Rivilla Rodríguez

Original idea

Fernando Rico Villas

Testing

Miguel Sanz Novo
Marina Centenera Merino
Lydia Markopoulioti

License