RADEX Python Interface
(radex.py)
User Guide

Andrés Megías Toledano
Centre for Astrobiology (CAB), Madrid

Version 1.4
February 2023

Index

1. Introduction

The RADEX Python Interface is a Python 3 script, `radex.py`, that uses RADEX software[1] to make a non-LTE fit (LTE: *Local Thermodynamic Equilibrium*) to the observed lines and derive the column density of the species, and optionally also the molecular hydrogen (H_2) number density or the kinetic temperature of the molecule.

First of all, the script requires the software RADEX to be installed locally in your computer.[1] With regard to Python, the script requires the libraries PyYAML, NumPy, Pandas, Matplotlib, SciPy, and RichValues.[2] 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 `radex.py`:

```
python3 radex.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

1. https://personal.sron.nl/~vdtak/radex/index.shtml
2. https://github.com/andresmegias/richvalues

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. We also have to specify the path of the RADEX executable file in the system. It can be done in the configuration file (see section 4) and also editing the variable radex_path in the script. If radex.py is not in the current folder, but in a folder with path <folder>, we should write <folder>/radex.py instead of radex.py in the command line in order to run it. 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

In order to run calculations for a certain species, you need a file containing physical and thermodinamical information about the species. It can be downloaded from the LAMDA database[3] (Leiden Atomic and Molecular Database). Once downloaded, it has to be placed inside the subdirectory data/ inside the RADEX local folder. Lastly, it should have the same name as in the RADEX Online species list,[4] so you may have to rename the file.

Additionally, there are two files that can be used for each source to extract automatically some data, in which case they do not have to be explicitly written in the configuration file. Below is a description of each name, with the same names as the given ones in the configuration file:

- **MADCUBA table**. Transition lines table in .csv format generated by the tool SLIM from the software MADCUBA. To generate it in SLIM, select one of the species in the SimFit tab to open the transition table window, check the Spectral, Fit, and Phys. options, and click the save icon to export the table in .csv format. This variable is only needed if the values of the line widths, the temperature, and/or the initial column density are set to be chosen from SLIM.
- **spectra table**. Table in .csv format with information of the RMS noise and resolution of the spectra. Each row, which corresponds to one spectral range, must have a column called rms noise (mK), with the value of the RMS noise of the spectral range. Such a file can be generated by the Automated GILDAS-CLASS Pipeline (classpipeline.py) once we have run the steps --selection, --line_search, and --reduction, adding the argument --spectra_table (for example: python3 classpipeline.py config.yaml --spectra_table). This variable is only needed if one does not specify the values of the RMS noise in the variable RADEX parameters.

3. Output files

After the running of the script, a file in YAML formatting style with the results of the calculations will be generated. It will contain a Python dictionary with name non-LTE molecules, containing one next-level dictionary for each molecule studied. For each of them, there will be four entries: one for the column density (column density (/cm2)), one for the kinetic temper-

3. https://home.strw.leidenuniv.nl/~moldata/
4. http://var.sron.nl/radex/radex.php

ature (kinetic temperature (K)), other for the molecular hydrogen number density (hydrogen number density (/cm3)), and another regarding the lines fit (lines (GHz)). Except for the last entry, this variable has the same structure as the corresponding variable for the SLIM Table Generator (slimtables.py), although the exact names of the species studied could vary a little bit, so they may need to be edited if you want to use this other script.

Plus, several plots may be generated for each species and each source studied, showing the loss function between the observed lines and the predicted ones by RADEX with respect to the column density, and showing also the intensity of the predicted lines versus the observed ones.

4. Configuration file

The configuration file is a plain text document in the YAML formatting style. This way, we can specify the options of the script radex.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,[5] its default value will be used (see Section 3).

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

- **show RADEX plots**. Logical variable that makes the code show a plot of the loss function (the error) between the observed lines and the predicted ones by RADEX with respect to the column density.
- **save RADEX plots**. Logical variable that makes the code save the generated plots.
- **save individual plots**. Logical variable that makes the code save an individual image for each of the generated subplots.
- **load previous results**. Logical variable that makes the code try to load previous results from another RADEX calculation and just plot the results.
- **RADEX path**. Path of the executable file of RADEX in your system. You can avoid specifying this variable if you edit the script radex.py and change the value of the variable radex_path, at the beginning.
- **input files**. Nested variable for specifying the name of the input file.
  - **MADCUBA table**. Transition lines table in .csv format generated by the tool SLIM from the software MADCUBA. To generate it in SLIM, select one of the species in the

5. 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.

SimFit tab to open the transition table window, check the Spectral, Fit, and Phys. options, and click the save icon to export the table in .csv format. If there are multiple sources, this variable should be a list. This variable is only needed if the values of the line widths, the temperature, and/or the initial column density are set to be chosen from SLIM.

  - **spectra table**. Table in .csv format with information of the RMS noise and resolution of the spectra used in MADCUBA. Each row, which corresponds to one spectral range, must have a column called rms noise (mK), with the value of the RMS noise of the spectral range. Such a file can be generated by the script classpipeline.py once we have run the steps --selection, --line_search, and --reduction, adding the argument --spectra_table. If there are multiple sources, it should be a list. This variable is only needed if one does not specify the values of the RMS noise in the variable RADEX parameters.

- **output files**. Nested variable for specifying the names of the output file generated by the script.
  - **RADEX results**. Plain text file in YAML formatting style which contains the values of the minimized column density and its uncertainty for the input molecules, as well as the kinetic temperature which was introduced in the variable RADEX parameters.
- **lines margin (MHz)**. Frequency range around the given value of the transitions that will be used by the script to identify each observed line from the lines produced by RADEX.
- **RADEX fit parameters**. Text variable that indicates which parameters should be optimized when minimizing the difference between the observed and modelled lines. Possible values, which are self-explicative, are 'column density', 'column density, H2 number density', and 'column density, kinetic temperature'.
- **maximum iterations for optimization**. Maximum number of iterations for the optimization method to find the optimal values of the variable/s.
- **grid points**. In case that two variables are being optimized, this variable sets the number of points that will have one side of the grid used to explore the parameter space to calculate the uncertainties.
- **RADEX parameters**. Nested variable with the names of the molecules that will be modeled with RADEX, as they appear in its species catalogue. Each of the names must be also a nested variable with the following next level variables:
  - **MADCUBA name**. The name of the molecule in the transition lines table generated by SLIM, from MADCUBA. This is only needed if the line width, the temperature, and/or the initial column density is set to be taken from MADCUBA.
  - **observed lines**. Dictionary (or nested variable) with the following entries:
    - **frequency (MHz)**. Array with the frequencies, in MHz, of the observed lines.
    - **intensity (K)**. Array with the intensities, in K, of the observed lines. If there are multiple sources, it should be a list of arrays.

        - **RMS noise (mK)**. Array with the values of the RMS noise, in mK, of the spectra in the surroundings of the observed lines. If the value is set to 'auto', the script will use the values from the given .csv table in the variable spectra table. If there are multiple sources, it should be a list of arrays.
    - **background temperature (K)**. Background temperature of the source of the observations. If there are multiples sources, it should be a list.
    - **kinetic temperature (K)**. Kinetic temperature of the molecule. If the value is set to 'auto', the script will use the value of the excitation temperature of the LTE model done by MADCUBA from the transition lines table of the variable MADCUBA table. If there are multiples sources, it should be a list.
    - **hydrogen number density (/cm3)**. Number density of molecular hydrogen (H_2), in /cm^3, in the source of the observations. If there are multiples sources, it should be a list.
    - **line width (km/s)**. Velocity width, in km/s, of the observed lines. If the value is set to 'auto', the script will use the value of the LTE model done by MADCUBA from the transition lines table of the variable MADCUBA table. If there are multiples sources, it should be a list.
    - **column density (/cm^2)**. Initial value of the column density of the molecule, in /cm^2, that will be used to minimize the column density with RADEX. If the value is set to 'auto', the script will use the value of the LTE model done by MADCUBA from the transition lines table of the variable MADCUBA table. If there are multiples sources, it should be a list.
    - **fit parameters**. Text variable that indicates which parameters should be optimized when minimizing the difference between the observed and modelled lines, but only for this specific species. Possible values are 'column density', 'column density, H2 number density', and 'column density, kinetic temperature'.

Default variable values

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

```
show RADEX plots: yes
save RADEX plots: no
save individual plots: no
load previous RADEX results: no
RADEX path: ''
input files: []
output files:
    radex results: radex-output.yaml
RADEX parameters: []
RADEX fit parameters: 'column density'
lines margin (MHz): 0.2
maximum iterations for optimization: 200
grid points: 15
```

5. Example case

Here we show a brief example to see how to use the script radex.py. We will study the two thermodynamical variants of methanol (CH_3OH) named A and E, for two different sources, and making use of the SLIM transition lines table. For methanol A, we will only fit the column density, but for methanol E we will also fit the H_2 number density, as we have two observed transitions.

5.1. Input files

For this example, we need to download the molecular data files for methanol A and E from the LAMDA database.[6] The files are named ch3oh_a.dat and e-ch3oh.dat, so we have to rename the first one to a-ch3oh.dat and place both files inside the subdirectory data/ inside the RADEX directory. Moreover, we will use four files in .csv format, two for each source:

- **MADCUBA table**. It contains information of all the transitions of the studied species and can be exported in the transition table window in SLIM.
- **spectra table**. It contains information about the observations and can be generated by the Automated GILDAS-CLASS Pipeline (script classpipeline.py) with the argument --spectra_table.

5.2. Configuration file

Below is an example of configuration file for this example case.

```
show RADEX plots: yes
save RADEX plots: yes
save individual plots: yes
RADEX fit parameters: 'column density, H2 column density'
input files:
    MADCUBA table:
    - L1517B-madcuba.csv
    - L1517BOFF1-madcuba.csv
    spectra table:
    - L1517B-spectra.csv
    - L1517BOFF1-spectra.csv
output files:
    RADEX results: radex-output.yaml
RADEX parameters:
    CH3OH-A:
        MADCUBA name: CH3OH-A,vt=0-2
        observed lines:
            frequency (MHz): [96741.371, 95914.310]
            intensity (K): [[0.63, 0], [0.73, 0]]
        background temperature (K): [2.73, 2.73]
        kinetic temperature (K): [10, 10]
        hydrogen number density (/cm3): [2.2e5, 1.235e5]
        line width (km/s): ['auto', 'auto']
        column density (/cm2): ['auto', 'auto']
        fit parameters: 'column density'
    CH3OH-E:
```

6. https://home.strw.leidenuniv.nl/~moldata/

```
MADCUBA name: CH3OH-E,vt=0-2
observed lines:
    frequency (MHz): [96739.358, 96744.545]
    intensity (K): [[0.44, 0.03], [0.53, 0.04]]
background temperature (K): [2.73, 2.73]
kinetic temperature (K): [10, 10]
hydrogen number density (/cm3): [2.2e5, 1.235e5]
line width (km/s): ['auto', 'auto']
column density (/cm2): ['auto', 'auto']
```

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`.

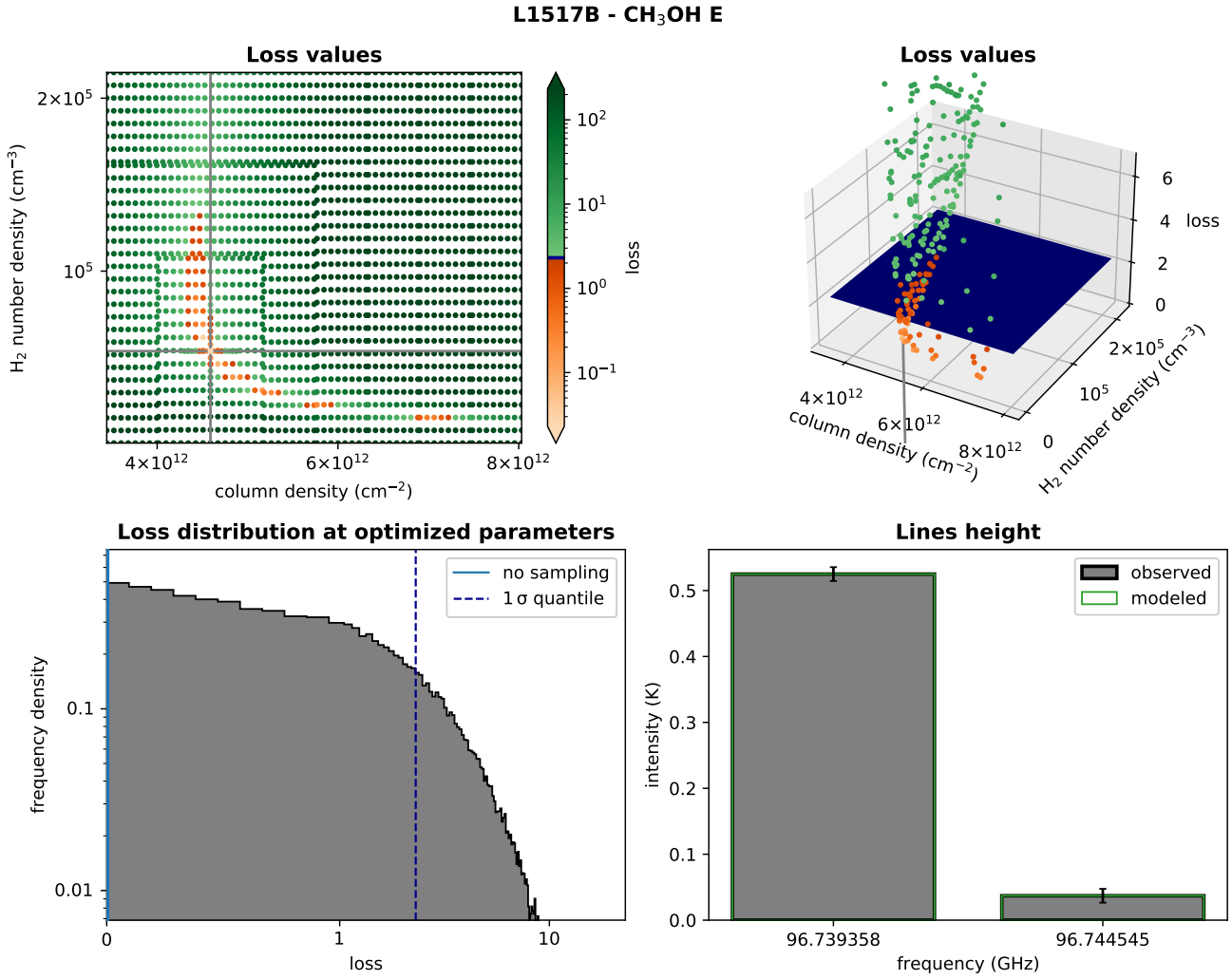

Figure 1. Plots generated for the RADEX calculations for an observation of the pre-stellar core L1517B and for the species methanol E (CH_3OH E). *Upper-left*: Values of the difference between the observed lines and the lines generated by RADEX (the loss function) with respect to different values of the column density. The gray lines indicates the fitted values of the column density and H_2 number density that minimize the loss. *Upper-right*: Same as the previous plot, but in 3 dimensions. *Lower-left*: Histogram of the values of the loss function obtained when varying the observed line intensity according to the noise. The blue dashed line indicates the threshold value used to define the uncertainties in the previous plot (68.27 percentile). *Lower-right*: Scheme of the height of the observed lines, comparing the observed height versus the modelled one.

5.3. Output plots

As we set the variable show RADEX plots to yes, a plot will be generated for each species and each source. Figure 1 shows one of these plots for the previous configuration file.

5.4. Results of the calculations

Below is the content of the output file generated for the previous configuration file.

```
non-LTE molecules:
  CH3OH-A:
    column density (/cm2):
    - 4.705-0.035+0.033 e12
    - 5.476-0.037+0.036 e12
    hydrogen number density (/cm3):
    - 2.2e5
    - 1.235e5
    kinetic temperature (K):
    - 10
    - 10
    lines (GHz):
      '96.741371':
      - intensity (K): '0.6872'
        width (km/s): '0.283'
      - intensity (K): '0.8001'
        width (km/s): '0.283'
  CH3OH-E:
    column density (/cm2):
    - 4.59-0.66+3.29 e12
    - 5.62+/-0.30 e12
    hydrogen number density (/cm3):
    - 5.4-3.9+3.1 e4
    - 2.05+/-0.29 e5
    kinetic temperature (K):
    - 10
    - 10
    lines (GHz):
      '96.739358':
      - intensity (K): '0.525'
        width (km/s): '0.283'
      - intensity (K): '0.635'
        width (km/s): '0.283'
      '96.744545':
      - intensity (K): '0.037'
        width (km/s): '0.283'
      - intensity (K): '0.079'
        width (km/s): '0.283'
```

6. Using RADEX Python Interface along with SLIM Table Generator

The same configuration file can be used for both RADEX Python Interface (radex.py) and SLIM Table Generator (slimtables.py)[7] as they share most of the variables of their configuration files. Basically, we can use the configuration file of tablegenerator.py and just add the variables show RADEX plots and RADEX parameters.

7. Which is part of the MADCUBA-SLIM Python Scripts: https://github.com/andresmegias/madcuba-slim-scripts.

If we want to use the results of RADEX in the abundances table of slimtables.py, we can copy the results from the output file (named radex-output.yaml by default) to the configuration file. Then we would have to change the name of each molecule, which was its name as in the RADEX catalogue, to the name that will be shown in the table. For example, methanol A (CH_3OH A) is called CH3OH-A in RADEX catalogue, but we may want to name it CH3OH A, which will be displayed as CH_3OH A in the tables generated by slimtables.py. Additionally, the entry lines (GHz) can be deleted, as it is not used by slimtables.py.

7. Mathematical basis of the calculations

RADEX is a radiative transfer model that allows us to predict the line intensity of the selected molecule from the following parameters:

- **Molecule.** It has to be included in the molecule RADEX list, therefore having collisional cross-sections derived, experimentally or theoretically.
- **Spectral range.** Frequency range for the transitions to be predicted for the selected molecule.
- **Excitation conditions.**
  - Molecular hydrogen density.
  - Kinetic temperature of the molecule.
  - Background temperature.
- **Radiative transfer parameters.**
  - Column density.
  - Line width.

For each species, the results of the calculation would be a set of intensity lines, $\{\hat{I}_i\}$, where i is the index of the line. There is a set of observed intensities, $\{I_i\}$, from the configuration file, and an uncertainty for each line (the RMS noise), so we will also have a set of uncertainties, $\{\Delta I_i\}$. The rest of the parameters except the column density (and optionally the H_2 number density or the kinetic temperature) have to be specified in the configuration file.

The goal is to optimize the column density and optionally also the H_2 number density or the kinetic temperature, to minimize the difference between the observed lines, $\{I_i\}$, and the predicted ones, $\{\hat{I}_i\}$, taking into account the uncertainties, $\{\Delta I_i\}$. To measure that difference, one has to define a loss function, $\mathcal{L}$, which we choose to be the chi-square (χ^2) error:

$$\mathcal{L} = \sum_i \left(\frac{\hat{I}_i - I_i}{\Delta I_i} \right)^2 .$$

In this way we are measuring the quadratic error weighted with the inverse of the uncertainty, so that a difference between the observations and the model contributes less to the loss if the uncertainty on the line height is greater.

Once the loss function is properly defined, it can be minimized with respect to the column density (and also the number density or the kinetic temperature, optionally). This calculation is done with the library SciPy, using the function minimize. In particular, the script uses the method COBYLA with one variable and the method Nelder-Mead for two vari-

ables.

Finally, we have to estimate the uncertainty for the optimized value/s of our parameters/s. To do so, we take a threshold for the loss, greater than the minimized value, which defines a lower and an upper uncertainty. In order to find a proper threshold, the script creates 10^5 sets of variations of the observed intensities, $\{I_i\}_j$, so that the value of each intensity, $I_{i,j}$, comes from a normal distribution with median I_i and standard deviation ΔI_i. Then, the loss, $\mathcal{L}$, is calculated for the optimized model for each set of intensities $\{I_i\}_j$, obtaining a distribution of losses (see figure 1, lower-left plot). We choose the threshold so that the loss values less than it are the 68.27 % of the total values (similarly to the definition of 1 σ confidence interval, but for an asymmetric distribution with a minimum value); that is, the threshold is the 68.27 percentile.

Now, the script will explore our parameter space around the minimized values in order to find when the loss function reaches our threshold value. If the column density is the only parameter to be fitted, it is only necessary to calculate the loss function in the surroundings of the minimized value until we reach the threshold value. However, if the H_2 column density or the kinetic temperature have also to be fitted, the script will have to explore a bidimensional parameter space. To do so, it makes an adaptive grid that starts calculating the loss values at the surroundings of the obtained minimum and continues enlarging its size until it encloses all the loss values minor to the threshold with a reasonable margin (see figure 1, upper plots). Therefore, the uncertainties are defined within the parameter space that make the loss lower than the threshold value.

Citation of the code

If you use the RADEX Python Interface for your work, it would be great if you cite it. You can put the link to the GitHub repository, where this user guide can also be downloaded:

https://github.com/andresmegias/radex-python .

If you use this code for scientific research, you can cite the paper in the following link, in whose appendix the algorithm is explained:

https://ui.adsabs.harvard.edu/abs/2023MNRAS.519.1601M/abstract .

Useful links

The following links may be of interest:

- **RADEX.**
  https://personal.sron.nl/~vdtak/radex/index.shtml
- **MADCUBA.**
  https://cab.inta-csic.es/madcuba/
- **MADCUBA-SLIM Python Scripts.**
  https://github.com/andresmegias/madcuba-slim-scripts
- **Automated GILDAS-CLASS Pipeline.**
  https://github.com/andresmegias/gildas-class-python
- **Python.**
  https://www.python.org/
- **PyYAML.**
  https://pyyaml.org/wiki/PyYAMLDocumentation/
- **NumPy.**
  https://numpy.org/
- **Pandas.**
  https://pandas.pydata.org/
- **Matplotlib.**
  https://matplotlib.org/
- **SciPy.**
  https://scipy.org/
- **RichValues.**
  https://github.com/andresmegias/richvalues/

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

Supervising

Izaskun Jiménez Serra
Jesús Martín-Pintado Martín

License

The RADEX Python Interface (radex.py) is published under a GNU General Public License (GPL) version 3.