User Guide for abundance gradient fitting

André Felipe de Siqueira Cardoso, PhD

Contents

1 Introduction

This guide serves as a support for using the script to fit the oxygen abundance gradient profile using indirect methods. The idea is that the user provides information about the galaxies, and the script will automatically, without human supervision, fit the abundance gradient. The best model will be determined based on the Akaike criterion (Akaike, 1973), evaluating whether the profile is linear, broken, or doubly broken. For a better visualization of the expected result, see the fits presented in Cardoso et al. (2025).

2 How to Use the Script

This script was designed to automate the calculations of oxygen abundances, as well as to fit the best model of the abundance gradient. The idea is that the user inputs the necessary galaxy information for this type of analysis, such as fluxes, RA, and Dec, for example. The user must download the six .py files available on GitHub and save them all in the same directory where this script will be used. The file `fit_OH.py` is the file that compiles all the other five files and performs the task. Inside `fit_OH.py`, the function `fit_final(...)` is defined, which should be used to run the script. Below are all the necessary details to use this script.

```
fit_OH.fit_final(
name = galaxy name (without spaces),
HIIREGID = array with the ID of each HII region,
ra = array of right ascension of each HII region in degrees,
ra0 = right ascension of the galaxy center in degrees,
dec = array of declination of each HII region in degrees,
dec0 = declination of the galaxy center in degrees,
pa = position angle in degrees,
```

```
ba = ratio of the galaxy's minor to major semi-axes,
d = galaxy distance in Mpc,
re = effective radius of the galaxy in kpc,
EWHa = array containing the equivalent width of each HII region in Angstroms (\AA),
Hb4861 = array with H$\beta$ flux,
eHb4861 = array with H$\beta$ flux error,
Ha6562 = array with H$\alpha$ flux,
eHa6562 = array with H$\alpha$ flux error,
OIII5006 = array with OIII5006 flux,
eOIII5006 = array with OIII5006 flux error,
NII6583 = array with NII6583 flux,
eNII6583 = array with NII6583 flux error,
SII6716 = array with SII6716 flux,
eSII6716 = array with SII6716 flux error,
SII6730 = array with SII6730 flux,
eSII6730 = array with SII6730 flux error,
calibrator = number identifying which index and calibrator to use,
criterion = identification of the criterion used to select HII regions,
save_table = True to save the table, False if not desired,
save_graph = True to save the graph, False if not desired,
show_graph = True to display the graph, False if not desired
)
```

The parameter *name* is the galaxy name and must be a string. The parameters *ra0, dec0, pa, ba, d*, and *re* are constant values and must be floats. The parameters *HIIREGID, ra, dec, EWHa, Hb4861, eHb4861, Ha6562, eHa6562, OIII5006, eOIII5006, NII6583, eNII6583, SII6716, eSII6716, SII6730*, and *eSII6730* must be arrays. The parameter *calibrator* must take values 1, 2, 3, 4, or 5, depending on the index and calibrator used in the abundance calculation (see section 3.2). The parameter *criterion* must be a string, identified as *KE01, KE6A, KA03, ST06*, or *CF11* according to the criterion chosen to select HII regions (see section 3.3). The parameters *save_table, save_graph*, and *show_graph* are booleans, taking True or False at the user's discretion. For a more detailed explanation of all these input parameters, see section 3 below.

It is important to note that if the user does not have the HIIREGID array, for example, it is possible to create an array like `HIIREGID = np.zeros(len(sample))`. However, caution is needed when using this with the other parameters, as it may calculate abundances incorrectly.

At the end of the task, the script returns a dictionary containing the parameters of the best-fit model. To do this, it is necessary to assign the result to a variable, as in the example below:

```
result = fit_OH.fit_final(name, HIIREGID, ra, ra0, dec, dec0, pa, ba, d, re, EWHa,
Hb4861, eHb4861, Ha6562, eHa6562, OIII5006, eOIII5006, NII6583, eNII6583, calibrator,
criterion, save_table, save_graph, show_graph)
```

Finally, the fit parameters can be accessed as:

Finally, the fit parameters can be accessed as:

```
galaxy = result['galaxy']
b0 = result['b0']
eb0 = result['eb0']
a1 = result['a1']
ea1 = result['ea1']
h1 = result['h1']
eh1 = result['eh1']
a2 = result['a2']
```

```
ea2 = result['ea2']
h2 = result['h2']
eh2 = result['eh2']
a3 = result['a3']
ea3 = result['ea3']
```

Section 4 presents an example of how to use this script. For this, it will be necessary to download the files "HII.NGC0309.flux_elines.diff_corr.csv", "data_NGC0309.csv", and "example.py", available on GitHub.

3 Step-by-Step Explanation of the Tasks

The workflow behind this script is divided into six steps. Each step corresponds to a file that performs a specific function. Below is an explanation of the information each function requires and what it returns.

3.1 distance.py

The file `distance.py` returns the deprojected distances of each HII region in the galaxy, normalized by the galaxy's effective radius, following Scarano et al. (2008). Below are the necessary input parameters for the calculation, along with the specific units of each quantity.

```
distance.distances(
ra = array of right ascension of each HII region in degrees,
ra0 = right ascension of the galaxy center in degrees,
dec = array of declination of each HII region in degrees,
dec0 = declination of the galaxy center in degrees,
pa = position angle in degrees,
ba = ratio of the galaxy's minor to major semi-axes,
d = galaxy distance in Mpc,
re = effective radius of the galaxy in kpc
)
```

This function will return an array with the distance *r* of each HII region relative to the galaxy center, normalized by the effective radius.

3.2 abundance.py

This function performs the extinction correction of the emission lines of the HII regions. Internally, the extinction factor for each emission line is calculated (Cavichia et al., 2010). Then, the color excess and the correction for each line are applied (Cavichia, 2008). In this case, line corrections are made relative to Hβ, so that the O3N2 and N2 indices are calculated as follows:

$$\mathrm{O3N2} = \log\left(\frac{\mathrm{OIII}}{\mathrm{H}\beta}\frac{\mathrm{H}\alpha}{\mathrm{H}\beta}\frac{\mathrm{H}\beta}{\mathrm{NII}}\right) \quad \text{and} \quad \mathrm{N2} = \log\left(\frac{\mathrm{NII}}{\mathrm{H}\beta}\frac{\mathrm{H}\beta}{\mathrm{H}\alpha}\right) \tag{1}$$

Based on these indices, a valid range of values is estimated for each calibrator. For example, the PP04 calibrator (Pettini & Pagel, 2004) with N2 and O3N2 indices is valid only within the ranges $-2.5 < \mathrm{N2} < -0.3$ and $-1.0 < \mathrm{O3N2} < 1.9$, respectively. The M13 calibrator (Marino et al., 2013) with N2 and O3N2 indices is valid only within $-1.6 < \mathrm{N2} < -0.2$ and $-1.1 < \mathrm{O3N2} < 1.7$, respectively. For more details, see Marino et al. (2013). For the D16 calibrator (Dopita et al., 2016), the warning is that the calibration is described in the paper as nearly linear up to an abundance of $12 + \log(\mathrm{O/H}) = 9.05$.

Additionally, fluxes are limited based on a Gaussian distribution, where regions with flux errors larger than 3σ of the full width at half maximum of the fitted Gaussian are discarded.

Finally, abundances are estimated according to the user's interest. The necessary information for this script to perform its functions is:

```
abundance.abundance(
name = galaxy name (no spaces),
r = distance of each HII region calculated in distance.py,
HIIREGID = array with the ID of each HII region,
Hb4861 = array with H$\beta$ flux,
eHb4861 = array with H$\beta$ flux error,
Ha6562 = array with H$\alpha$ flux,
eHa6562 = array with H$\alpha$ flux error,
OIII5006 = array with OIII5006 flux,
eOIII5006 = array with OIII5006 flux error,
NII6583 = array with NII6583 flux,
eNII6583 = array with NII6583 flux error,
SII6716 = array with SII6716 flux,
eSII6716 = array with SII6716 flux error,
SII6730 = array with SII6730 flux,
eSII6730 = array with SII6730 flux error,
calibrator = number identifying which index and calibrator to use
)
```

Currently, there are five calibrator options, as follows:

- **1 -** PP04 with O3N2
- **2 -** PP04 with N2
- **3 -** M13 with O3N2
- **4 -** M13 with N2
- **5 -** D16

The script will return arrays of the abundances with their respective errors, as well as the corrected Hα, OIII5006, and NII6583 lines. A file named "galaxy name.csv" is saved in the "tables" directory, containing the ID, the normalized distance *r*, and the Hα equivalent width of each HII region. The corrected Hα, OIII5006, NII6583, SII6716, and SII6730 lines with their respective errors, and the abundances for all calibrators are also saved. This table will be saved in the automatically created directory "tables" under the name "galaxy name.csv".

3.3 criteria.py

The file `criteria.py` selects, among all possible HII regions, those that meet the defined criteria for HII regions according to different selection methods. Currently, five different criteria proposed by different authors are implemented: Kewley et al. (2001), Kewley et al. (2001) with EWH$\alpha > 6$ Å, Kauffmann et al. (2003), Stasińska et al. (2006), and Cid Fernandes et al. (2011). These criteria basically depend on the corrected Hα, OIII, and NII lines and the EWHα. Thus, the required information to run this file is:

```
criteria.points(
name = galaxy name (no spaces),
criterion = identifier of the criterion used to select HII regions,
r = distance of each HII region calculated in distance.py,
```

```
OH = abundance of each region calculated in abundance.py,
OH_err = abundance error of each region calculated in abundance.py,
EWHa = array of HII region equivalent width in Angstrom (\AA),
Ha6562_cor = array of corrected H$\alpha$ line from abundance.py,
OIII5006_cor = array of corrected OIII5006 line from abundance.py,
NII6583_cor = array of corrected NII6583 line from abundance.py,
calibrator = number identifying which index and calibrator to use,
save_table = option to save a table containing r, OH, and eOH for the specific
calibrator and criterion
)
```

The criteria should be chosen from the available options below:

KE01 - Kewley et al. (2001)

KE6A - Kewley et al. (2001) with EWH$\alpha > 6$ Å

KA03 - Kauffmann et al. (2003)

ST06 - Stasińska et al. (2006)

CF11 - Cid Fernandes et al. (2011)

The file `criteria.py` will return arrays with the distance *r*, abundance *OH*, and abundance error *eOH*, according to the chosen criterion. For each criterion selected, the number of HII regions may vary, which can directly impact the abundance gradient fit. For a more detailed explanation, see sections 3.2, 4.1, and 5 of Cardoso et al. (2025).

If desired, the user can save a table containing *r*, *OH*, and *eOH* for the calibrator and criterion selected. To do this, set `save_table = True`, and the table will be saved in the automatically created directory "tables_criterions".

3.4 models.py

This file performs the fits of the oxygen abundance gradients. For a single linear profile fit, the *statsmodels.api* library is used. For profiles with one or two breaks, the *piecewise_regression* library (Pilgrim, 2021) is employed.

The *piecewise_regression* library can perform fits with *n* breaks in the gradient. Here, since our interest is at most two breaks, it is predefined to perform fits with only one and two breaks. By default, according to Cardoso et al. (2025), each fit performs a bootstrap of 2000 iterations to avoid the global minimum. Parameters such as *min_distance_to_edge*= 0.05, *min_distance_between_breakpoints*= 0.20, and *start_values*= [0.5, 1.5] are fixed to prevent unrealistic fits. Note that these values can be modified within *models.py* if desired. For more information on the *piecewise_regression* library, see Pilgrim (2021).

The *piecewise_regression* library is capable of performing fits with *n* breakpoints in the gradient. Here, since our interest lies in at most two breakpoints, the script is preset to perform fits with only one and two breakpoints. Furthermore, parameters such as *min_distance_to_edge* = 0.05, *start_values* = [0.5, 1.5], and *min_distance_between_breakpoints* = 0.20 were fixed to avoid unrealistic fits. It is worth noting that these values can be modified within the *models.py* file if desired by the user. According to Cardoso et al. (2025), a *bootstrap* of 2000 iterations was performed for each fit to escape local minima. In this script, a *bootstrap* of only 200 iterations is used, but this value can also be changed within the *models.py* file if desired. For more information about the *piecewise_regression* library, see Pilgrim (2021).

In this step, each of the three profiles will be fitted following the models given by equations 4, 5, and 6 in Cardoso et al. (2025). The best model is selected based on the Akaike Information Criterion (AIC, Akaike, 1973). For cases where the ratio of observed points to the number of free parameters is less than 40, an

AIC correction is applied as described in Narisetty (2020). For further details, see section 3.2 in Cardoso et al. (2025).

To run this script, the following information is required:

```
models.fit_models(
x_array = array of distance r obtained in criteria.py,
y_array = array of abundance OH obtained in criteria.py,
ey_array = array of abundance error eOH obtained in criteria.py
)
```

Finally, this script will return a dictionary containing the distance *r*, the abundance *OH*, and the abundance error *eOH*. Additionally, it will provide information on the best-fitted model, i.e., the one with the lowest AIC value, as well as the AIC values of each fitted model. The script also returns the coefficients of the fits according to equations 4, 5, and 6 in Cardoso et al. (2025), along with their respective errors. Here, the coefficient a_2 is defined as the slope of the main negative gradient. The coefficients a_1 and a_3 correspond to the slopes of the inner and outer regions, respectively. h_1 and h_2 are the positions where the breaks occur in the gradient, and b_0 is the linear coefficient. For a single linear fit, for example, the values are $a_1 = h_1 = h_2 = a_3 = 0.0$.

3.5 plot.py

This file generates the plot of the oxygen abundance gradient with the fit of the best model for the selected calibrator and criterion. This script takes the information from the dictionary obtained in 3.4 and uses it to generate the plot. The necessary information to create the plot is:

```
plot.plot_models(
results_dict = dictionary returned in models.py,
nome = galaxy name (no spaces),
criterion = identifier of the criterion used to select HII regions,
calibrator = number identifying which index and calibrator to use,
save_graph = option to save the plot,
show_graph = option to display the plot
)
```

Finally, the plot is generated, and it is up to the user to decide whether to view and/or save it. For this, set save_graph = True and show_graph = True. Additionally, a dictionary containing the galaxy name and the coefficients of the best-fit model is returned. It is the user's responsibility to decide whether to save these coefficients in a table and how to do so.

4 Example

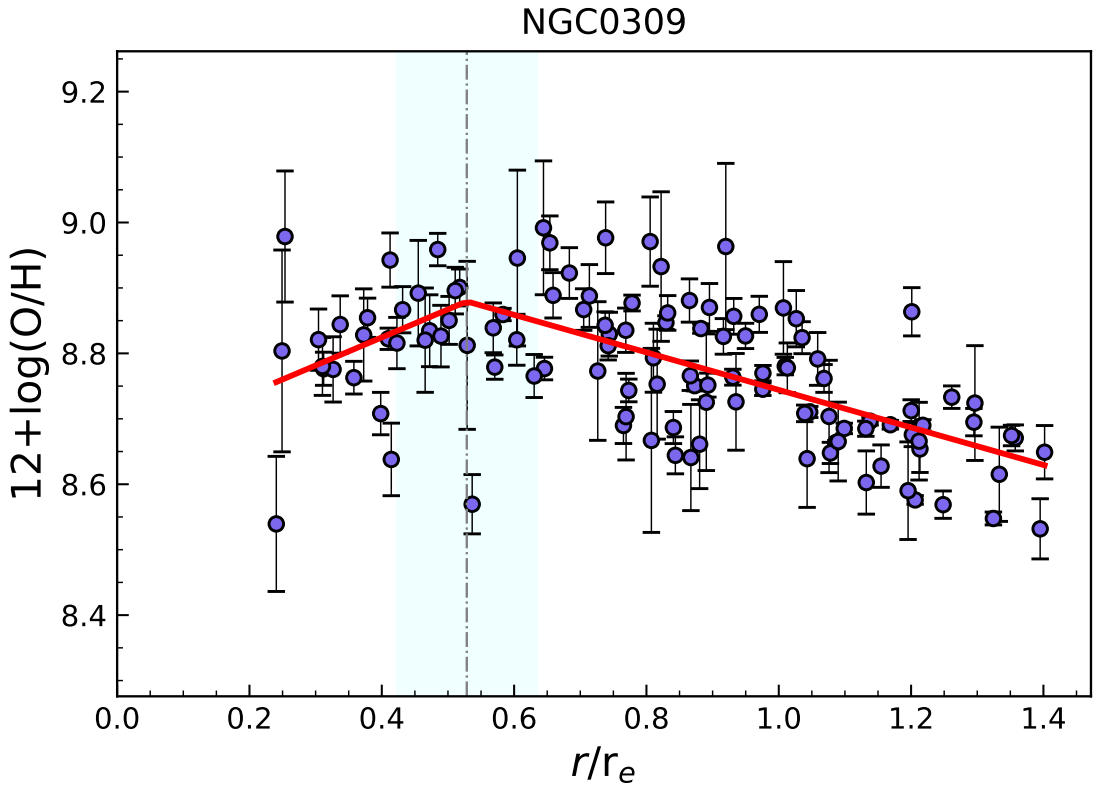
To run this model script, it is necessary to download the seven files: distance.py, abundance.py, criteria.py, models.py, plot.py, fit_OH.py, and example.py. Additionally, you need to download the files data_NGC0309.csv and HII.NGC0309.flux_elines.diff_corr.csv. After downloading, save all files in a single directory. Then, open example.py in Spyder (if you prefer Jupyter Notebook, some modifications to the script may be necessary) and run the script. The content of example.py is shown below:

```
1 import fit_OH
2 import pandas as pd
3
4 ###############################################################################
5
6 galaxy_data = pd.read_csv('data_NGC0309.csv')
```

```
7
8  name = galaxy_data['galaxy'].iloc[0]
9  ra0 = galaxy_data['ra0'].iloc[0]
10 dec0 = galaxy_data['dec0'].iloc[0]
11 pa = galaxy_data['pa'].iloc[0]
12 ba = galaxy_data['ba'].iloc[0]
13 re = galaxy_data['re'].iloc[0]
14 d = galaxy_data['dist'].iloc[0]
15
16 ###############################################################################
17
18 flux = pd.read_csv('HII.NGC0309.flux_elines.csv')
19
20 ra = flux['RA']
21 dec = flux['DEC']
22 EWHa = flux['EWHa6562']
23 HIIREGID = flux['HIIREGID']
24 Hb4861 = flux['fluxHb4861']
25 eHb4861 = flux['e_fluxHb4861']
26 OIII5006 = flux['fluxOIII5006']
27 eOIII5006 = flux['e_fluxOIII5006']
28 Ha6562 = flux['fluxHa6562']
29 eHa6562 = flux['e_fluxHa6562']
30 NII6583 = flux['fluxNII6583']
31 eNII6583 = flux['e_fluxNII6583']
32 SII6716 = flux['fluxSII6716']
33 eSII6716 = flux['e_fluxSII6716']
34 SII6730 = flux['fluxSII6730']
35 eSII6730 = flux['e_fluxSII6730']
36
37 ###############################################################################
38
39 calibrator = 1
40 criterion = 'KA03'
41 save_table = True
42 save_graph = True
43 show_graph = True
44
45 ###############################################################################
46
47 results = fit_OH.fit_final(name, HIIREGID, ra, ra0, dec, dec0, pa, ba, d, re, EWHa
       , Hb4861, eHb4861, Ha6562, eHa6562, OIII5006, eOIII5006, NII6583, eNII6583,
       SII6716, eSII6716, SII6730, eSII6730, calibrator, criterion, save_table,
       save_graph, show_graph)
48
49 print(results)
```

This script will fit the galaxy NGC 0309 using the O3N2 index with the PP04 calibrator. The HII region selection criterion adopted is KA03 (Kauffmann et al., 2003). The script automatically saves the file NGC0309.csv in the "tables" directory, which contains the abundances for all available calibrators. Since save_table = True and save_graph = True, the file NGC0309_KA03_PP04_O3N2.csv and the plot NGC0309_KA03_PP04_O3N2.png will be saved in the "tables_criterions" and "graphs" directories, respectively (these directories are created automatically). Additionally, since show_graph = True, the plot will be displayed in Spyder. Finally, the command print(results) will display the dictionary containing the best-fit model parameters.

At the end, the user should obtain the following plot:



References

Akaike, H. 1973, Information Theory and an Extension of the Maximum Likelihood Principle (New York, NY: Springer New York), 199–213

Cardoso, A. F. S., Cavichia, O., Mollá, M., & Sánchez-Menguiano, L. 2025, , 980, 45, doi: 10.3847/1538-4357/ad9eab

Cavichia, O. 2008, Master's thesis, Universidade de São Paulo, São Paulo

Cavichia, O., Costa, R. D. D., & Maciel, W. J. 2010, , 46, 159, doi: 10.48550/arXiv.1003.0416

Cid Fernandes, R., Stasińska, G., Mateus, A., & Vale Asari, N. 2011, , 413, 1687, doi: 10.1111/j.1365-2966.2011.18244.x

Dopita, M. A., Kewley, L. J., Sutherland, R. S., & Nicholls, D. C. 2016, , 361, 61, doi: 10.1007/s10509-016-2657-8

Kauffmann, G., Heckman, T. M., Tremonti, C., & et al. 2003, , 346, 1055, doi: 10.1111/j.1365-2966.2003.07154.x

Kewley, L. J., Dopita, M. A., Sutherland, R. S., Heisler, C. A., & Trevena, J. 2001, , 556, 121, doi: 10.1086/321545

Marino, R. A., Rosales-Ortega, F. F., Sánchez, S. F., & et al. 2013, , 559, doi: 10.1051/0004-6361/201321956

Narisetty, N. N. 2020, in Handbook of Statistics, Vol. 43, Principles and Methods for Data Science, ed. A. S. R. Srinivasa Rao & C. R. Rao (Elsevier), 207–248, doi: https://doi.org/10.1016/bs.host.2019.08.001

Pettini, M., & Pagel, B. E. J. 2004, , 348, L59–L63, doi: 10.1111/j.1365-2966.2004.07591.x

Pilgrim, C. 2021, Journal of Open Source Software, 6, 3859, doi: 10.21105/joss.03859

Scarano, S., Madsen, F. R. H., Roy, N., & Lépine, J. R. D. 2008, , 386, 963–972, doi: 10.1111/j.1365-2966.2008.13079.x

Stasińska, G., Cid Fernandes, R., Mateus, A., Sodré, L., & Asari, N. V. 2006, , 371, 972, doi: 10.1111/j.1365-2966.2006.10732.x