Versatile Emission Line Fitting Package Guideline

Zixuan Peng[1]

[1] *zixuanpeng@ucsb.edu*
[1]Department of Physics, University of California, Santa Barbara

Contents

1 Installation

For installing the `astroconda-lmfit` environment that is required to run the line-fitting algorithm, please follow these steps in the terminal:

1. Given the environment definition YML file (i.e., `environment.yml`), users can build the environment with:
   `conda env create -f environment.yml`
2. Activating an environment: Once you have built the environment, users can activate it with:
   `conda activate veremisfit`
3. To deactivate the environment, run:
   `conda deactivate`

Optional:

1. To delete the environment with all of its packages, run:
   `conda env remove -n <name>`
2. To rename the conda environment, run:
   `conda create --name <new_name> --clone <old_name>`

2 Running

Please modify the `run_line_fitting.py` file for running the line-fitting algorithm, especially the lines that load the 1D flux, error, and wavelength arrays. Also, users need to input the redshift of the specific galaxy and specify if the input spectrum is in "vac" (Vacuum) or "air" (Air) wavelength space. Please check the `Python` example below for the definitions of other crucial parameters of the line fitting algorithm.

```
# Step (1): Load the flux, error, and wavelength arrays
data_fits = current_direc + '/example_inputs/j1044+0353_addALL_icubes_wn.fits'
err_fits = current_direc + '/example_inputs/j1044+0353_addALL_icubes_wn_err.fits'
spec = fits.open(data_fits)[0].data
err = fits.open(err_fits)[0].data
wave = np.load(current_direc + '/example_inputs/wave_grid.npy')

# Step (2): Main input Parameters
# redshift of the galaxy
redshift = 0.01287

# whether the spectrum is in "vac" or "air" wavelength space
vac_or_air = "air"

# the polynomial order of fitting local continuum level
fit_cont_order = 1

# "line_selection_method = "gui" or "txt" defines
# how you want to select the lines for fitting, whether using a GUI or inputting a txt"
line_select_method = "txt"

# text file for selecting intended lines for fitting
input_example_txt = current_direc + '/input_txt/line_selection_example.txt'

# whether to interactively determine the fitting window, local continuum regions,
# and masking lines
fit_window_gui = False # if False, use the default values

# whether to pop up the GUI window for interactively determine the initial parameter values
# and their corresponding ranges (for each iteration).
# Default is False (i.e., pop up the window)
params_windows_gui = False # if False, use the default parameter initial values
# and corresponding ranges for each iteration

# define the folder name
folder_name = data_fits.split('/')[-1][:-5] # if None, then a tk window will pop up for users
                                            # to interactively enter the folder name;
                                            # if users forget to type the folder name
                                            # in the tk window, the default folder name is
                                            # "test_folder".

# define the file name
file_name = "test" # if None, then file_name = folder_name;
                   # else, file_name will be f"{folder_name}_{file_name}"

# define the minimum velocity width (i.e., instrumental seeing) for each velocity component
sigma_min = 30 # in km / s

# define the maximum velocity width for each velocity component (for emissions)
sigma_max_e = 1200 # in km / s

# define the maximum velocity width for each velocity component (for absorptions)
sigma_max_a = 1500 # in km / s

# define the algorithm used for fitting (the well-tested ones include
# "leastsq": Levenberg-Marquardt (default), "nelder": Nelder-Mead, and "powell": Powell)
fit_algorithm = "leastsq"

region = line_fitting_exec(redshift = redshift, vac_or_air = vac_or_air,
                           folder_name = folder_name, file_name = file_name,
                           line_select_method = line_select_method,
                           input_txt = input_example_txt, fit_cont_order = fit_cont_order,
                           fit_window_gui = fit_window_gui,
                           params_windows_gui = params_windows_gui, sigma_min = sigma_min,
                           sigma_max_e = sigma_max_e, sigma_max_a = sigma_max_a,
                           fit_algorithm = fit_algorithm)
```

```
# Step (3): Run the line-fitting algorithm
# The main parameters for the all_lines_result function are defined as follows:

# "n_iteration": Defines the number of iterations you want to run.

# "get_flux = True" defines if you want the return to be the flux dictionary (includes the
# flux of each line profile) or not; if False, then the return is the best-fitting parameters

# "get_error = True" defines if you want to calculate the error of each line flux

# "get_ew = True" defines if you want to calculate the ew(s) of the selected emission lines
# (including emission and absorption ew(s))

# "save_flux_table = True" defines if you want to save the best-fitting flux pandas table
# for selected lines.

# "save_ew_table = True" defines if you want to save the best-fitting equivalent width pandas
# table for selected lines.

# "save_par_table = True" defines if you want to save the best-fitting parameter pandas table
# for each velocity component.

# "save_stats_table = True" defines if you want to save the best-fitting statistics pandas
# table for selected lines.

# "save_cont_params_table = True" defines if you want to save the best-fitting parameters for
# continuum fits of selected lines.

region.all_lines_result(wave, spec, err, n_iteration = 100, get_flux = True,
                        save_flux_table = True, get_ew = True,
                        save_ew_table = True, get_error = True,
                        save_par_table = True,
                        save_stats_table = True, save_cont_params_table = True)

# Step (5): Plot the fitting result
# "savefig = True" defines if you want to save the fitting result as a .pdf file.
region.fitting_plot(savefig = True)
```

The specific method for choosing the intended lines for fittings, controlled by the `line_selection_method = 'txt' or 'gui'` in the `line_fitting_exec` class, is illustrated in the following subsection.

2.1 Line Selection Methods

Users can select the intended lines for fitting either by using the line-selection GUI (`line_select_method = 'gui'`) or by inputting a text file (`line_select_method = 'txt'`) that includes all the lines for fitting. The description of each method is illustrated below respectively.

2.1.1 Line Selection GUI

When users set the `line_select_method` parameter to `'gui'`, a line-selection window appears, displaying all available emission lines for fitting (Figure 1). Lines are grouped by their element names (highlighted in bold font). Users should click on the "white box" adjacent to the desired line's name to opt for a particular line. Beyond the "`Free fitting`" approach, where both the velocity center and width of each component remain variable, users can also select from alternative strategies: "`Fix velocity centroid`" (all components share a common velocity center), "`Fix velocity width`" (uniform velocity width across components), or "`Fix velocity centroid and width`" (all components have matching velocity centers and widths). To finalize their lines and fitting strategy choices, users should click the "`Confirm`" button, moving on to subsequent line selection steps.

The subsequent step (see Figure 3 for details) requires users to characterize the chosen line(s) within the line-selection GUI, primarily by ascertaining if the lines exhibit multiple emission velocity components. The GUI will query users on the necessity of a two-component (two emission velocity components) or a three-compoennt (three emission velocity components) model for the line's fitting. Subsequently, users will be prompted to determine if the line showcases broad wings (and whether it should be fitted by a Gaussian or a Lorentzian model), implying that the velocity width of the second or third emission component should be broader than

the first's. Finally, another GUI window will pop up, prompting users to inspect if the selected line(s) present absorption troughs, a feature commonly associated with Balmer lines.

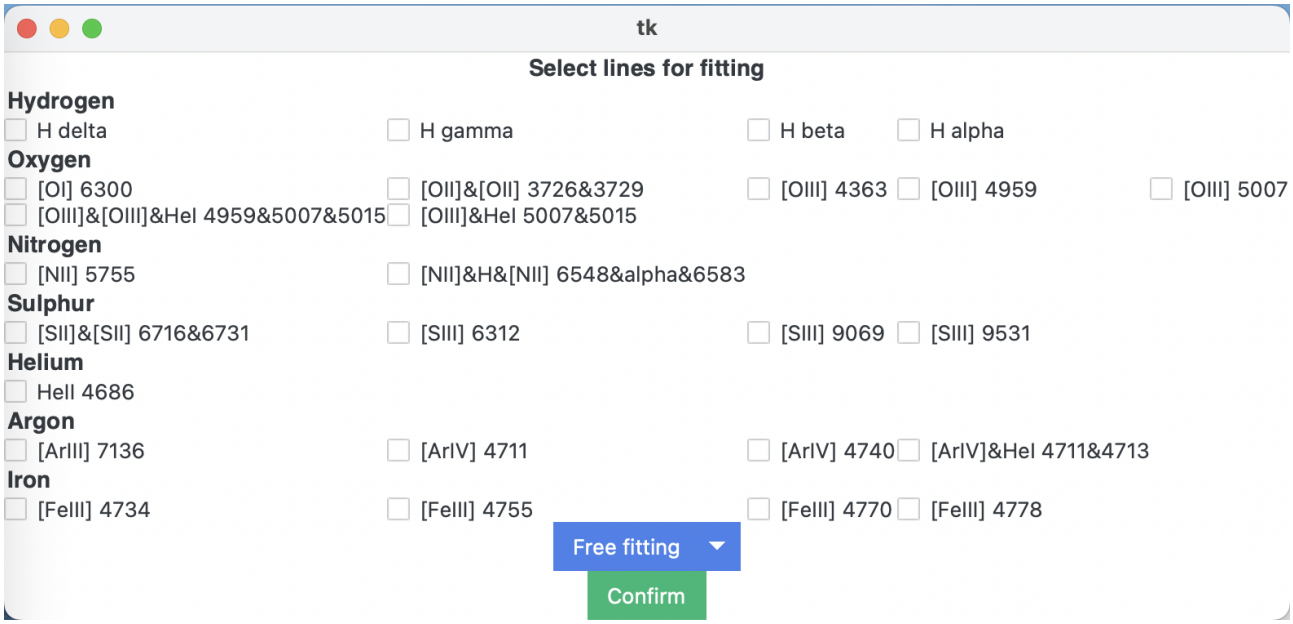

Figure 1: Demonstration of the line-selection GUI allowing users to choose the intended lines (grouped by element names) for fitting. Users can click on any lines available in the GUI. To proceed, users are prompted to click the `Confirm` button, advancing to further line selection processes. This includes evaluations for the presence of multiple emission components and absorption troughs (as detailed in Figure 3).

2.1.2 Line Selection Input Text File

In scenarios where users wish to fit the same lines and employ a consistent fitting strategy across multiple galaxy spectra, repeating the line selection process through the GUI can become tedious and inconvenient. To streamline this, users can opt to provide a text file containing all the necessary line-selection information by setting the `line_select_method` parameter to `'txt'`. The required format for this text input can be found in `line_selection_example.txt` within the `input_txt` folder (see Figure 2 for details).

2.1.3 Include More Emission Lines

The `vac_wavelengths.txt` or `air_wavelengths.txt` file in the `doc` folder contain the lines available for fitting, as shown in the line-selection GUI. Users can modify these files by adding new lines or editing existing ones, following the provided format. For example, [O III] λ4363: 4364.44 indicates that the line [O III] λ4363 has a (vacuum) wavelength of 4364.44 Angstroms. It's recommended to categorize each line under the specific element section (e.g., `<Oxygen>` for [O III] λ4363), so that the added line appears under that element in the line-selection GUI. Comments can be included by starting a line with '#'.

line_selection_example.txt

```
# example input file: selecting lines for fitting

# lines available for fitting should match with vac_wavelengths.txt, air_wavelengths.txt, and default_window_vspace.txt in doc folder
# example valid input line names are shown below:
# 'H delta' 'H gamma' 'H beta' 'H alpha'
# '[OII]&[OII] 3726&3729' '[OIII]&HeI 5007&5015' '[NII]&H&[NII] 6548&alpha&6583' '[SII]&[SII] 6716&6731'
# '[OIII] 4363' '[OIII] 4959' '[OIII] 5007' '[OI] 6300'
# 'HeII 4686' '[ArIV]&HeI 4711&4713' '[ArIV] 4711' '[ArIV] 4740'

######################## Line Selection Keywords
# selected lines (needs quotation marks for the selected lines)
selected_lines = '[OIII] 4959'

# fitting methods
# select from "Free fitting", "Fix velocity centroid", "Fix velocity width", "Fix velocity centroid and width"
fitting_method = "Free fitting"

# lines that have multiple emission components (needs quotation marks for the selected lines); if no lines have multiple emission components, input None.
multi_emis_lines = '[OIII] 4959'

# lines that need double-Gaussian fittings (needs quotation marks for the selected lines); if no lines have multiple emission components, input None.
double_gauss_lines = None

# lines that need triple-Gaussian fittings (needs quotation marks for the selected lines); if no lines have multiple emission components, input None.
triple_gauss_lines = '[OIII] 4959'

# whether the double or triple Gaussian fittings have broad wings or not
double_gauss_broad = True # sigma_2 > sigma_1
triple_gauss_broad = True # sigma_3 > sigma_2 > sigma_1

# lines that need a Lorentzian function for fitting broad wings ("core" is fitted by Gaussians by default)
lorentz_bw_em_lines = None

# absorption troughs that need a Lorentzian function for fitting broad wings (especially for Balmer lines)
lorentz_bw_abs_lines = None

# lines that have an absorption component (needs quotation marks for the selected lines)
absorption_lines = None

# set fixed amplitude ratios between two sets of lines (ratio = amp_num / amp_den)
amp_num_lines = None
amp_den_lines = None
amp_fixed_ratio = None
```

Figure 2: Example of the line selection input text file. Key parameters that will be used in subsequent analyses include `selected_lines` (lines selected for fitting), `fitting_method` (fitting strategy, as described in Section 2.1.1), `multi_emis_lines` (lines with multiple emission components), `double_gauss_lines` (lines requiring double-Gaussian fitting), `triple_gauss_lines` (lines requiring triple-Gaussian fitting), `double_gauss_broad` (indicating if the second emission component has a broader velocity width than the first), `triple_gauss_broad` (indicating if the second and third emission components have broader velocity widths than the first), `lorentz_bw_em_lines` (emission lines that need a Lorentzian function for fitting broad wings), `lorentz_bw_abs_lines` (absorption troughs that need a Lorentzian function for fitting broad wings), and `absorption_lines` (lines with an absorption trough or component). Users can fix the amplitude ratios between two lines (each of their components) using the `amp_num_lines`, `amp_den_lines`, and `amp_fixed_ratio` parameters, where `amp_fixed_ratio` = `amp_num_lines` / `amp_den_lines`. Lines beginning with the symbol '`#`' are treated as comments. For the parameters `selected_lines`, `multi_emis_lines`, `double_gauss_lines`, `triple_gauss_lines`, `lorentz_bw_em_lines`, `lorentz_bw_abs_lines`, and `absorption_lines`, the selected lines should be separated by commas, and each selected line needs quotation marks.

2.2 Fitting Window GUI

After adding a particular line to both wavelength files (illustrated in Section 2.1.3), users should also add this line to the `default_window_vspace.txt` file in the doc folder such that the pipeline can generate a default line-fitting window for this line. For example, [O III] λ4363: 800 means the line [O III] λ4363 has a half velocity width of the line-fitting window of 800 km s^{-1} (or equivalently, a total velocity width of 1600 km s^{-1}). The left and right local continuum regions are, by default, set to the first and last 10% of the wavelength range of the line-fitting window, respectively.

Users can modify the local continuum regions of a selected line by setting the parameter `fit_window_gui = True` for the `line_fitting_exec` function. In this scenario, a GUI will pop up through Bokeh (an example shown in Figure 4), a Python interactive visualization library. The description of each line of the Bokeh GUI is shown below.

1. Solid Black Line: represents the input spectrum.
2. Dashed Blue Line: represents the center of the fitting window, highlighting the central wavelength region for fitting a spectral feature.
3. Solid Red Lines: represent the lower end and upper end of the fitting window, creating a boundary around the wavelength region where a spectral line will be fitted.
4. Dashed Red Lines: mark the lower end of the right local continuum region or the upper end of the left local continuum region.
5. Dashed Orange Lines: mark the lower and upper ends of a masked region.

To modify the default local continuum regions generated by the pipeline, users are required to input four values (x1, x2, x3, and x4; an optional comment can also be included), separated by spaces, and press the return key

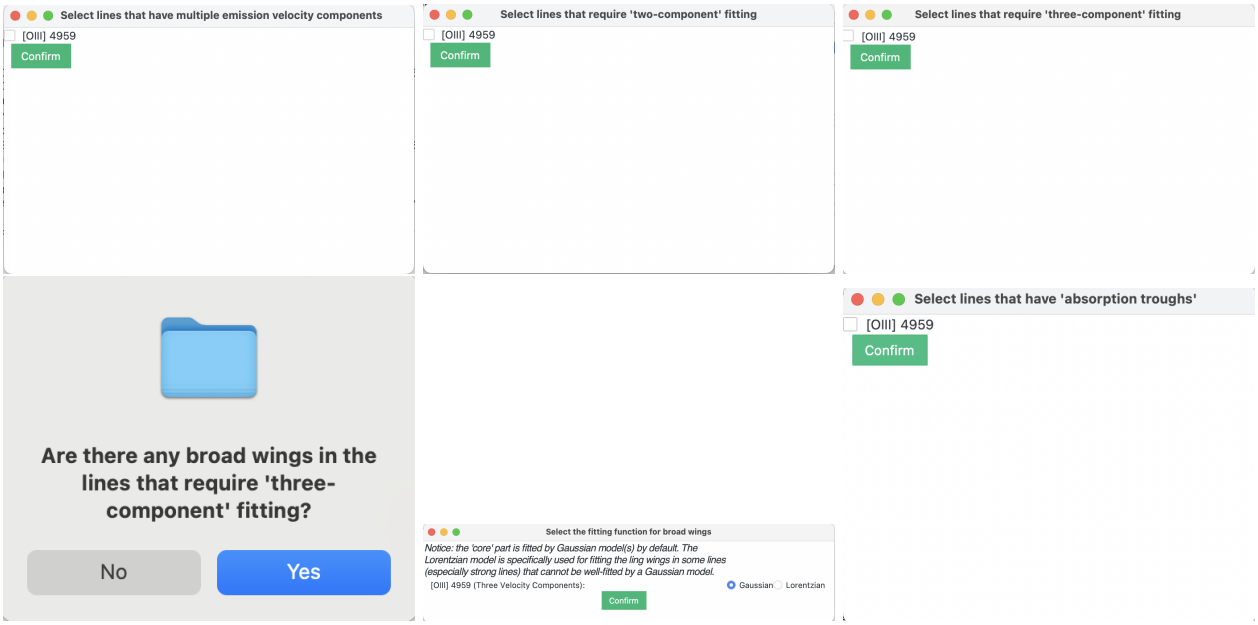

Figure 3: An illustration of the line-selection GUI, aiding users in determining the characteristics of the selected line(s) — specifically, [O III] λ4959 in this instance. *Top Left*: Assessing the presence of multiple emission velocity components. *Top Middle*: Establishing the need for a two-component fitting for its emission components. *Top Right*: Identifying the requirement for a three-component fitting for emission components. *Bottom Left*: Deciding if the line exhibits broad wings, causing the velocity width of the second or third emission component to be constrained as greater than the first. *Bottom Middle*: Determining if the emission line wing needs to be fitted by a Gaussian or a Lorentzian function. *Bottom Right*: Checking for a pronounced absorption trough demanding a fitting via a Lorentzian profile, typically seen in Balmer lines.

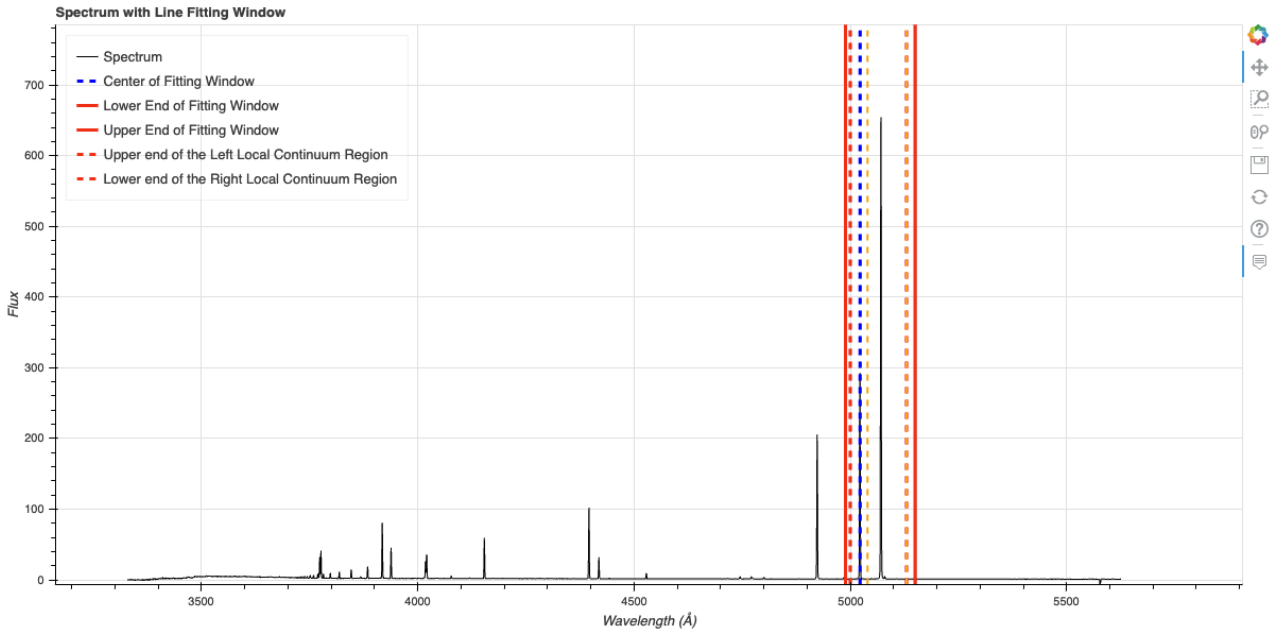

Figure 4: An illustration of the fitting-window GUI.

in the terminal, following the provided instructions. In the example below, the range from x1 to x2 Angstroms (denoted by red vertical solid lines) defines the lower and upper bounds of the line fitting window. The ranges from x1 to x3 Angstroms and from x4 to x2 Angstroms define the left and right local continuum regions, respectively. The modified local continuum regions will be updated in the Bokeh GUI. If users are satisfied with the current choice of local continuum regions, they need to press the return key in the terminal again and a **"no new local continuum regions updated"** message will be printed in the terminal.

```
Bokeh server is already running.
Using local continuum region file ~/[OIII]_4959.cont
```

```
CHECKING LINE FITTING WINDOW AND LOCAL CONTINUUM REGIONS
Current line fitting window: 4989.0 (x1) - 5150.0 (x2) Angstroms (red vertical solid
    lines)
Current local continuum regions: 4989.0 (x1) - 5000.0 (x3) Angstroms and 5130.0 (x4)
    - 5150.0 (x2) Angstroms
A <cr> will accept current values or enter new values to determine a new line fitting
     window or local continuum regions.
Hover the cursor over spectrum to find wavelengths.

Enter new left and right boundaries for fitting window and local continuum regions
x1: <float> x2: <float> x3: <float> x4: <float> comment: <str> (<cr> - done):
```

After modifying the local continuum regions, users can also mask sharp features, such as unintended absorption and emission lines, in the Bokeh GUI by following the instructions below. Users are required to input two values (shown as two orange dashed lines in the GUI), separated by spaces, to define a single masked region (an optional comment can also be included). This process can be repeated multiple times if users wish to mask several regions, and it can be terminated by pressing the return key in the terminal.

```
Using line mask file ~/[OIII]_4959.lmsk

MASKING SHARP FEATURES: UNINTENDED ABSORPTION LINES/EMISSION LINES
To mask, enter starting and stopping wavelengths for each line you want to mask.
Current masks are shown as vertical dashed yellow lines.
Hover the cursor over spectrum to find wavelengths.
Mask values are saved to this file: /Users/zixuanpeng/Desktop/
    multiple_line_fitting_gh/lmsk_dir/j1044+0353_addALL_icubes_wn/[OIII]_4959.lmsk.
A <cr> with no line limits will accept current masks.

Mask line: wavelength start stop (Ang) comment
<float> <float> <str> (<cr> - done):
```

2.3 Fitting Parameter GUI

After the line selection procedure, another GUI will appear for users, regardless of the line selection method used. This GUI allows users to input the initial guess for each parameter and specify its range of variation for each iteration (see Figure 5). By modifying the default initial values, users can ensure more appropriate initial parameter guesses. Furthermore, if users want to set the amplitude ratios between some pair of lines, such as [O III] $\lambda 4959, 5007$ or [N II] $\lambda 6548, 6583$, to ensure more physical line fitting results, they need to click on the `Set Amplitude Ratio` button to select the specific pairs of lines and their respective line ratio values in a GUI (see Figure 6).

Since the initial guesses are adjusted for each iteration based on the specified range of variation for each parameter, this method reduces the likelihood that the best-fitting result gets trapped in a local minimum within the multi-dimensional χ^2 space.

3 Saved Results

Users can review various line-fitting outputs. These outputs include a plot showcasing the best-fitting line profile, a CSV table detailing the best-fitting flux for each selected line, another CSV table displaying the best-fitting equivalent width (EW) of each selected line (and individual velocity components), a CSV table presenting the best-fitting parameter values for each velocity component, a CSV table saving the best-fitting statistics (including the χ^2 value) values for each selected line, and a CSV table showing the best-fitting continuum-related parameters (e.g., the zeroth and first order polynomials' values and their error-bars) for all selected lines. Users should enable the respective saving options to save each of these features. A summary of these options is provided below:

1. If `savefig = True` in the `fitting_plot()` function, the fitting result is saved as a .pdf file in the `plots` subfolder.
2. If `save_flux_table = True` in the `all_lines_result()` function, the best-fitting line flux table is saved in the `flux_tables` subfolder.
3. If `save_ew_table = True` in the `all_lines_result()` function, the best-fitting equivalent width table is saved in the `ew_tables` subfolder.

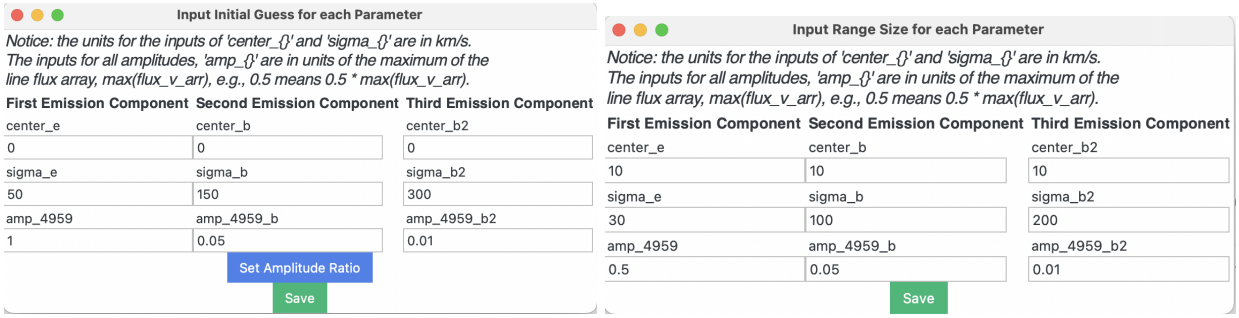

Figure 5: This is an example of the fitting-parameters GUI that allows users to input both the initial guess and the range of variation for each fitting parameter for the selected line(s). *Left*: the initial-guess window displays the default initial value for each fitting parameter. Each column header represents a parameter group (e.g., `First Emission Component`). A parameter name beginning with `center` denotes the velocity centroid of the Gaussian function, which defaults to `0` in units of km s^{-1}. Similarly, a name starting with `sigma` indicates the velocity width of the Gaussian function; its default value varies depending on the component and is also given in units of km s^{-1}. Parameters prefixed with `amp` define the amplitude of the Gaussian function. The default value for this varies among components and is given in units of the maximum flux, `max(flux_v_arr)`, from the extracted region surrounding the selected line. For instance, a value of 0.5 means `0.5 * max(flux_v_arr)`. Users can click the `Set Amplitude Ratio` button to fix the amplitude ratios between several selected line pairs (see Figure 6). *Right*: the parameter range-size window. Its columns and parameter nomenclature mirror those of the initial-guess window. The provided range values dictate the variation range for the initial guesses of fitting parameters throughout each fitting iteration.

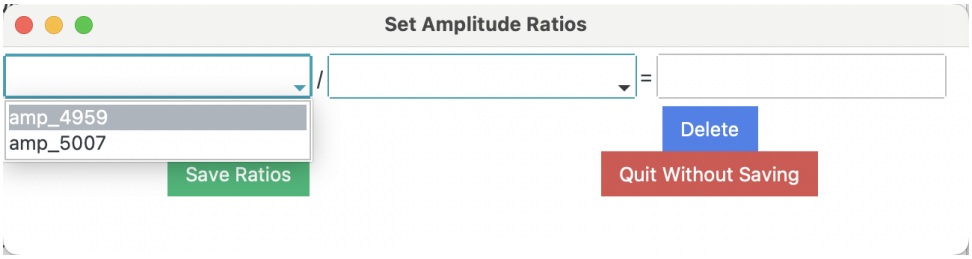

Figure 6: An example of the GUI that enables users to constrain the amplitude ratios between emission line pairs. Users need to select two lines, e.g., `amp_5007` and `amp_4959`, and their respective ratio, e.g., the intrinsic ratio is 2.98 for this [O III] doublet. To add or delete one selected line par, users can click the `Add` or `Delete` button on the GUI's lower left or lower right corner. Last, users must click the `Save Ratios` button to set the amplitude ratios for the line-fitting process.

4. If `save_par_table = True` in the `all_lines_result()` function, the best-fitting parameter table is saved in the `parameter_tables` subfolder.
5. If `save_stats_table = True` in the `all_lines_result()` function, the best-fitting statistics table is saved in the `stats_tables` subfolder.
6. If `save_cont_params_table = True` in the `all_lines_result()` function, the best-fitting continuum-related parameters table is saved in the `cont_tables` subfolder.

To show the versatility of this fitting package, the line-fitting examples in different scenarios are shown as follows.

1. Single-Gaussian fitting for a single line (see Figure 7).
2. Simultaneous single-Gaussian fittings for multiple lines (see Figure 8).
3. Triple-Gaussian fitting for a single line (see Figure 9).
4. Gaussian-Lorentzian fittings for the Balmer lines Hβ, Hγ, and Hδ that have obvious Balmer absorption troughs (see Figure 9).
5. Simultaneous fitting that includes a triple-Gaussian fitting for a strong line and two single-Gaussian fittings for two blended weaker lines (see Figure 11).

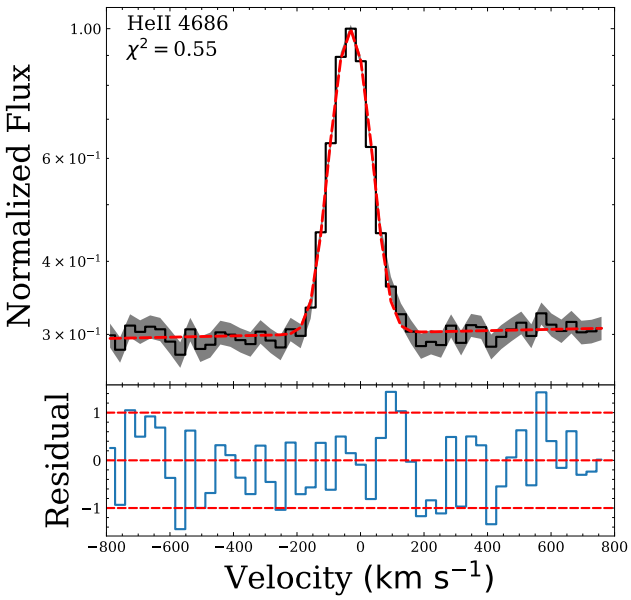

Figure 7: Example of single-Gaussian fitting for He II λ4686 line.

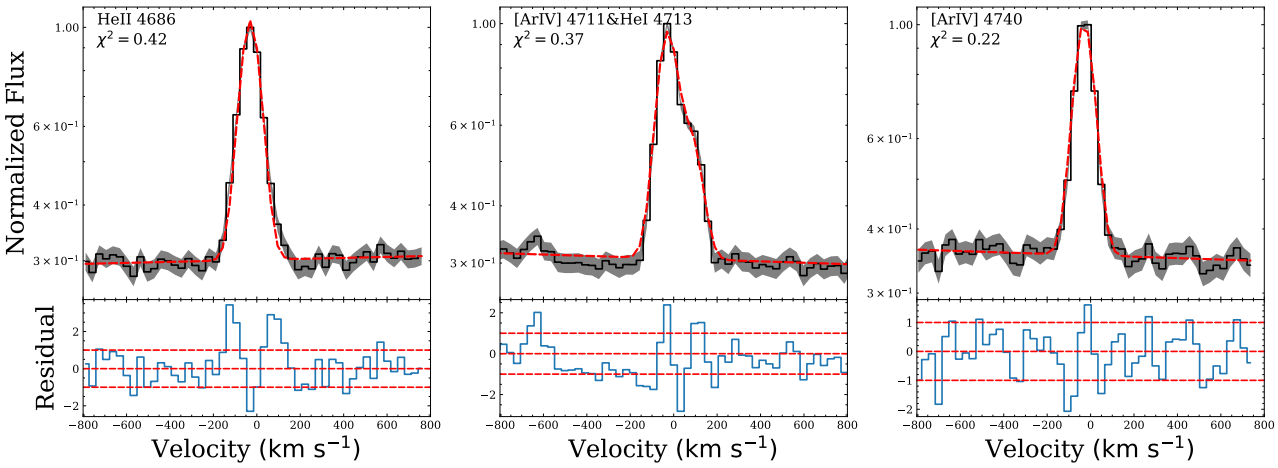

Figure 8: Example of simultaneous single-Gaussian fittings for He II λ4686, [Ar IV] λ4711, He I λ4713 line, and [Ar IV] λ4740. The velocity width and centroid are tied together in the fittings.

4 Recommended Fitting Procedures

1. **Check Available Emission Lines**: Refer to the `vac_wavelengths.txt` or `air_wavelengths.txt` file in the `doc` folder to verify if the desired emission lines are available. If not, follow the steps in Section 2.1.3 to add new lines. Remember to also add these new lines to the `default_window_vspace.txt` file in the `doc` folder to generate a default line-fitting window.

2. **Prepare Data and Set Parameters**: As demonstrated in Section 2, prepare the flux, error, and wavelength arrays. Adjust the input parameters accordingly. Choose the method to select lines for fitting (`line_select_method = gui` or `txt`) and specify the number of Gaussian functions for each line. Refer to the example in the `input_txt` folder and the documentation in the `doc` folder for guidance.

3. **(Optional) Fix the Amplitude Ratio Between Two Lines**: To ensure physical fitting results for two lines, it is advisable to fix the amplitude ratio between them, consistent with the theoretical line flux ratio. For example, `[OIII] 5007 / [OIII] 4959 = 2.98`. This adjustment can be implemented through the `params_windows` GUI or directly within the `input_txt` file. Adopting this strategy ensures that the line flux ratio remains constant, given that these lines are tied together in the velocity space (same velocity widths).

4. **Interactive Fitting Window Adjustment**: It is recommended to initially set `fit_window_gui = True` in the `line_fitting_exec` class. This will launch a Bokeh server, allowing interactive selection of the

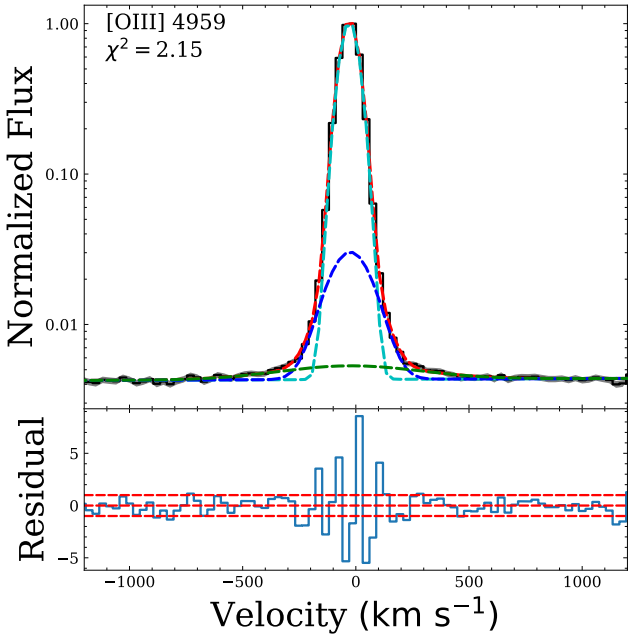

Figure 9: Example of triple-Gaussian fitting for [O III] λ4959 line such that the second and the third emission components have larger velocity widths than the first.

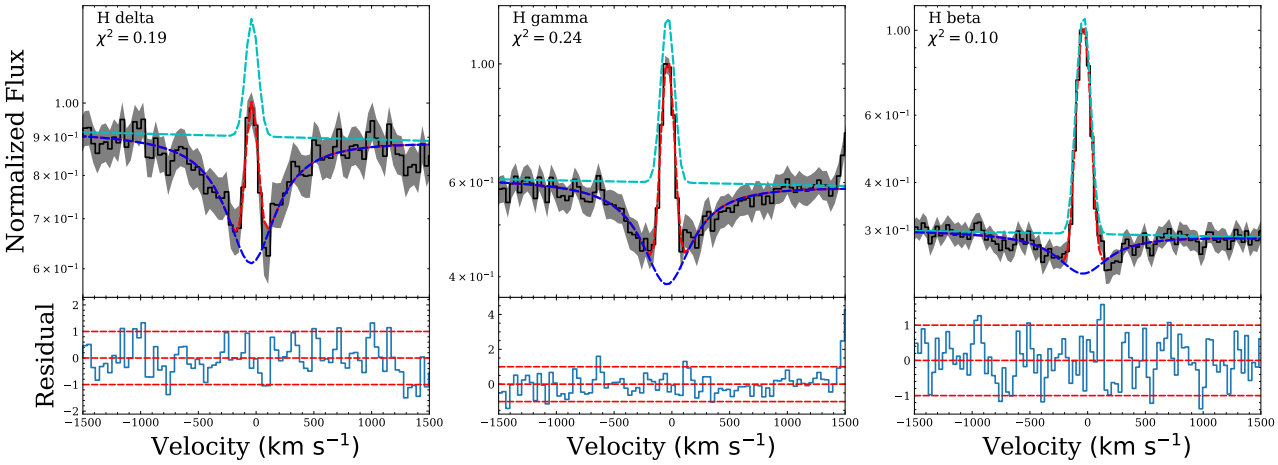

Figure 10: Example of Gaussian-Lorentzian fitting for Balmer lines that have obvious absorption troughs.

fitting window and masking of regions with sharp features. The fitting window and local continuum region information will be saved to the `cont_dir` folder as a `.cont` file, specifying four boundary points in Angstroms: `x1` (lower-end of the fitting window), `x2` (upper-end of the fitting window), `x3` (upper-end of the left local continuum region), and `x4` (lower-end of the right local continuum region). These define the local continuum regions as `x1 - x3` and `x4 - x2`. Modify the `fit_cont_order` parameter in the `line_fitting_exec` class to set the polynomial order for fitting the local continuum. Masked region information will be stored in the `lmsk_dir` folder as a `.lmsk` file, with each line defining a region to be masked during fitting. Masked regions will also be shaded as orange in the best-fitting plot.

5. **Review Saved Results**: Refer to Section 3 to examine the fitting outcomes, including the PDF file in the `plots` folder and other relevant tables such as flux tables in the `flux_tables`, equivalent width tables in the `ew_tables`, and best-fitting parameter tables in the `parameter_tables`.

6. **Refitting with Saved Settings**: For refitting the same selected lines, the saved `.cont` and `.lmsk` files will be loaded to recreate the previously chosen fitting window, local continuum regions, and masked regions. Therefore, you can set `fit_window_gui = False` in the `line_fitting_exec` class.

7. **(Optional) Replotting Results with Different Plotting Styles**: If you are not satisfied with the default best-fitting plot, you can refer to the `replot_best_fit_results.py` example located in the

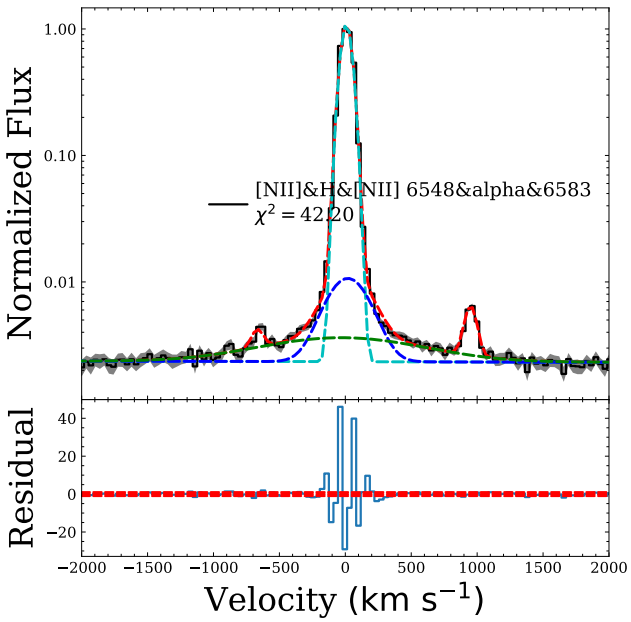

Figure 11: Example of a simultaneous fitting that includes a triple-Gaussian fitting for the strong line Hα and two single-Gaussian fittings for the weaker blended doublet [N II] λ6548, 6583.

`veremisfitting/replot_examples` folder. This example demonstrates how to retrieve the best-fitting results from the pipeline and replot them using your own style.

8. **(Optional) Potential Application to IFU Data**: Importing the VerEmisFitting package for fitting spectra extracted from IFU data can be time-consuming and redundant, as it requires repeatedly clicking buttons in every pop-up GUI for each spectrum. Therefore, it is recommended to set `line_select_method = 'txt'`, `fit_window_gui = False`, and `params_window_gui = False`. The first two parameters are illustrated in the previous steps. The last parameter (`params_window_gui`) controls whether to interactively determine the initial values and ranges of parameters for each iteration. With this setup, no interactive GUI will pop up; thus, it is essential to ensure that the default configuration adequately fits your data. For each extracted spectrum from the IFU data cube, it is recommended to assign a unique folder name (`folder_name`) or file name (`file_name`) within the `line_fitting_exec` class to ensure the saved products are uniquely named. If the local continuum regions or the masked regions differ significantly across these extracted spectra, assigning a unique `folder_name` for each spectrum is recommended. This approach is preferred because the program is designed to find unique `cont_dir` and `lmsk_dir` folders for each respective spectrum. An example that applies this pipeline to IFU data can be found in the `ifu_example.py` script located in the `veremisfitting/ifu_apps_examples` folder.

5 Citation

If you utilize VerEmisFitting in your research, please cite Peng et al. (2024, submitted to ApJ). A link to the publication will be provided upon its release.

6 Contact

VerEmisFitting is developed and maintained by Zixuan Peng and Yuan Li. For bug reports, questions, or feature requests, please contact us via email.