FitELP Documentation

Daniel Muthukrishna

Oct 04, 2020

Contents:

Note: You are reading the most recent edition of this documentation which content the latest available version.

FitELP (Fit Emission-Line Profiles) is a tool specifically designed to perform spectral emission-line fits with multiple gaussian components in echelle or long-slit data. The actual fitting procedure is based on the Non-Linear Least-Square Minimization and Curve-Fitting (LMFIT) package, https://lmfit.github.io/lmfit-py/ (Newville et al. 2014, https://doi.org/10.5281/zenodo.11813).

The Python code was designed for the analysis of the internal kinematics of star-forming regions using echelle data. However, it can be used to model any emission-lines in both in echelle and longslit spectroscopy.

FitELP allows you to:

- Modelling all individual emission-lines profiles with multiple-gaussian components.
- Visualise the individual fit of the emission-lines profiles with/without residuals.
- Perform an kinematic analysis of each target.
- Produces pdf and latex tables with the results for each ion on each gaussian component:

- radial velocities, velocity dispersion, fluxes, emission measure (EM), and global flux, with the corresponding errors.

- Average of radial velocities and velocity dispersion with the corresponding errors.

- H-alpha luminosity and the corresponding Star Formation Rate (SFR) base on Kennicutt (1998) equation, abscissa and ordinate values of the BPT-NII diagnostic diagram.

- Visualise the different BPT diagnostic diagrams.
- Produces latex tables with the flux, continuum, equivalent width, with the corresponding errors and for each ion.

Note: The code is currently still in development, and there may be many issues present. Contact the authors for assistance.

CHAPTER 1

Installation

1.1 From Source

The source code can be downloaded from GitHub and dependencies can be installed by running the following commands:

```
git clone https://github.com/daniel-muthukrishna/FitELP.git
cd FitELP
python setup.py install
```

1.2 Dependencies

`numpy`, `matplotlib`, `uncertainties`, `lmfit`, `astropy`.

These can all be install with *pip* if they were not already installed by the setup file. You will also need LaTex installed.

CHAPTER 2

Usage

This package is used to fit emission-lines from echelle or long-slit spectral data with a continuum and multiple Gaussian components. There are various methods that are useful detailed in the *API*. Example usage and analysis tools available are outlined below.

2.1 Fit emission-lines profiles

To set-up the properties of an object with multiple emission-lines use the following class. See *Example* for an example set-up.

- *fitelp.line_profile_info.RegionParameters*

To fit emission-lines in this object use the following method. See *Example* for an example of how to add emission-lines.

- *fitelp.line_profile_info.RegionParameters.add_em_line()*

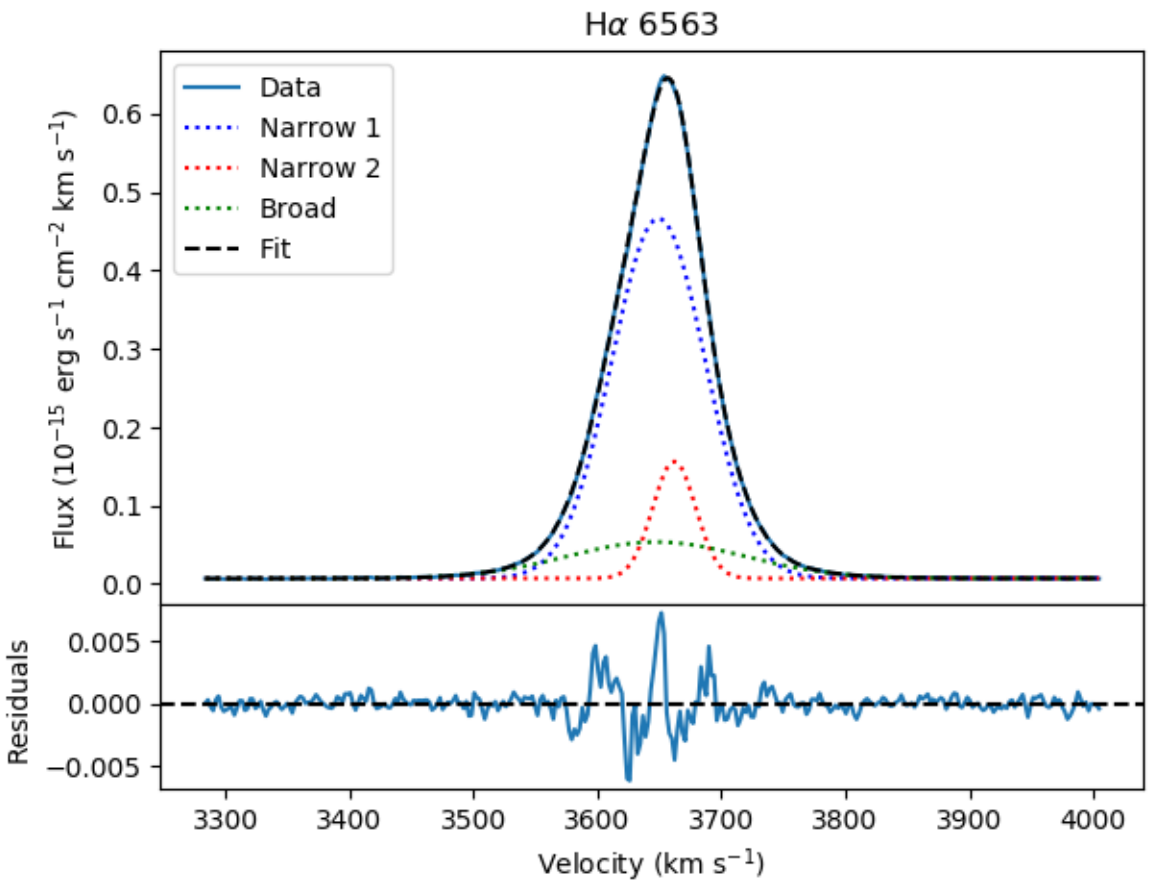
2.2 Compute kinematics of the object

To perform an analysis on the object use the class.

- *fitelp.kinematics_calculations.RegionCalculations*

Initiating this class will output several plots, tables, and data files about a object. Examples of outputs are shown below.

A plot of each of the emission-lines fit with as many Gaussian components as specified will be saved. An example of a fit to a H-alpha emission-line is shown below.



A table in the form a LaTeX file and a pdf outlining the kinematics of each emission-line will be saved. An example is shown below.

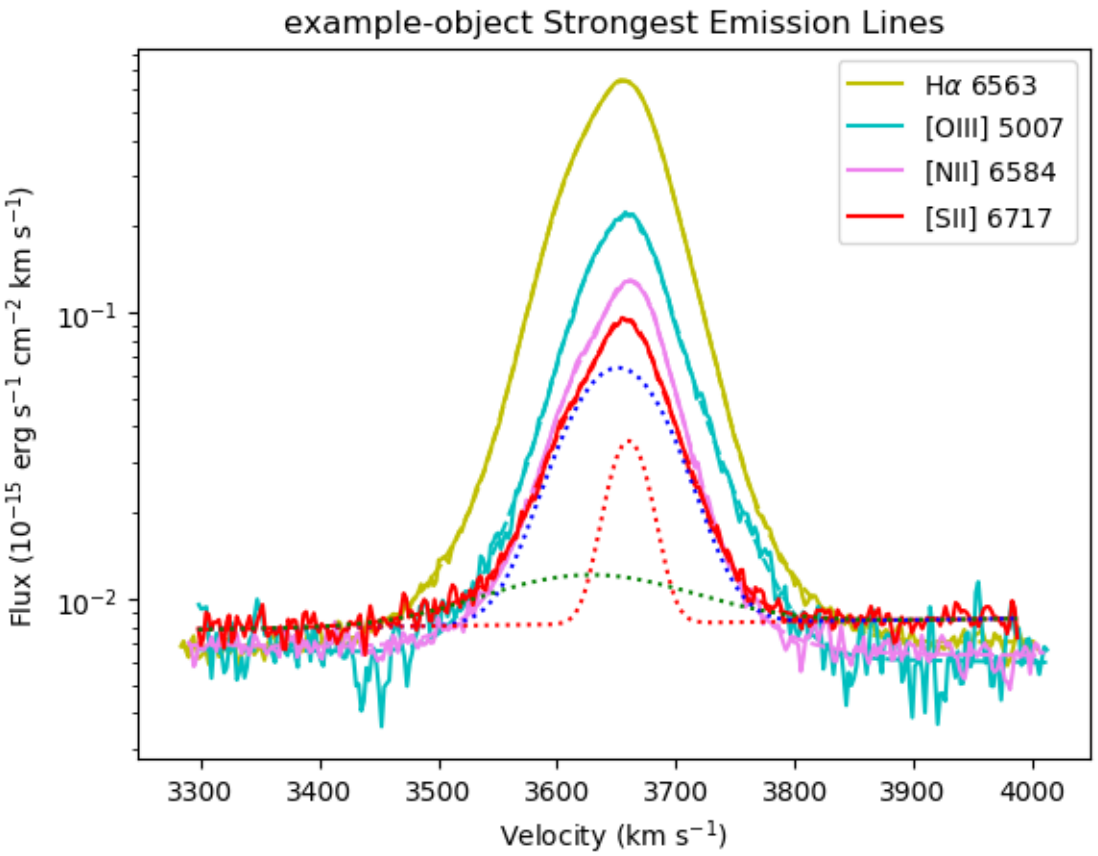
λ_0 (Å)	Ion	Comp.	v_r (km s^{-1})	σ_{int} (km s^{-1})	Flux (10^{-15} erg s^{-1} cm^{-2} km s^{-1})	EM_f	GlobalFlux (10^{-15} erg s^{-1} cm^{-2} km s^{-1})
6563	Hα	Narrow 1	3649.2 ± 0.1	34.9 ± 0.1	43.32 ± 0.265	74.5	58.17 ± 0.420
		Narrow 2	3661.9 ± 0.2	10.1 ± 0.4	6.48 ± 0.170	11.1	
		Broad	3647.3 ± 0.4	70.4 ± 0.9	8.36 ± 0.279	14.4	
5007	[OIII]	Narrow 1	3653.4 ± 0.7	29.2 ± 0.9	9.38 ± 0.626	49.4	18.98 ± 0.911
		Narrow 2	3664.6 ± 1.3	13.4 ± 1.8	1.25 ± 0.332	6.6	
		Broad	3659.6 ± 0.8	56.0 ± 1.4	8.35 ± 0.573	44.0	
4959	[OIII]	Narrow 1	3653.4 ± 0.0	29.2 ± 0.0	3.03 ± 0.118	51.2	5.91 ± 0.178
		Narrow 2	3664.6 ± 0.0	13.4 ± 0.0	0.51 ± 0.054	8.7	
		Broad	3659.6 ± 0.0	56.0 ± 0.0	2.37 ± 0.122	40.1	
4861	Hβ	Narrow 1	3649.2 ± 0.0	34.9 ± 0.0	8.68 ± 0.110	64.3	13.49 ± 0.161
		Narrow 2	3661.9 ± 0.0	10.1 ± 0.0	1.73 ± 0.050	12.8	
		Broad	3647.3 ± 0.0	70.4 ± 0.0	3.08 ± 0.107	22.9	
6300	[OI]	Narrow 1	3649.2 ± 0.0	34.9 ± 0.0	0.56 ± 0.032	51.6	1.08 ± 0.052
		Narrow 2	3661.9 ± 0.0	10.1 ± 0.0	0.12 ± 0.011	10.9	
		Broad	3647.3 ± 0.0	70.4 ± 0.0	0.40 ± 0.039	37.6	
6584	[NII]	Narrow 1	3652.2 ± 0.4	36.8 ± 0.6	7.12 ± 0.276	68.3	10.43 ± 0.409
		Narrow 2	3666.3 ± 0.3	14.8 ± 0.5	1.67 ± 0.090	16.0	
		Broad	3648.1 ± 1.8	67.8 ± 4.5	1.64 ± 0.288	15.7	
6548	[NII]	Narrow 1	3652.2 ± 0.0	36.8 ± 0.0	2.47 ± 0.033	74.2	3.32 ± 0.050
		Narrow 2	3666.3 ± 0.0	14.8 ± 0.0	0.54 ± 0.013	16.1	
		Broad	3648.1 ± 0.0	67.8 ± 0.0	0.32 ± 0.034	9.7	
6717	[SII]	Narrow 1	3651.5 ± 0.5	40.0 ± 0.9	5.68 ± 0.162	74.3	7.65 ± 0.268
		Narrow 2	3660.4 ± 0.3	15.6 ± 0.6	1.16 ± 0.097	15.1	
		Broad	3627.1 ± 7.7	81.2 ± 9.1	0.81 ± 0.191	10.6	
6731	[SII]	Narrow 1	3651.5 ± 0.0	40.0 ± 0.0	3.94 ± 0.049	72.1	5.46 ± 0.074
		Narrow 2	3660.4 ± 0.0	15.6 ± 0.0	0.92 ± 0.022	16.8	
		Broad	3627.1 ± 0.0	81.2 ± 0.0	0.61 ± 0.051	11.1	

Table 1: example-object

2.2.1 Plot multiple profiles

You may plot which ever profiles you wish together using the following function: See *Example* for an example of using this. An example output plot is also shown below.

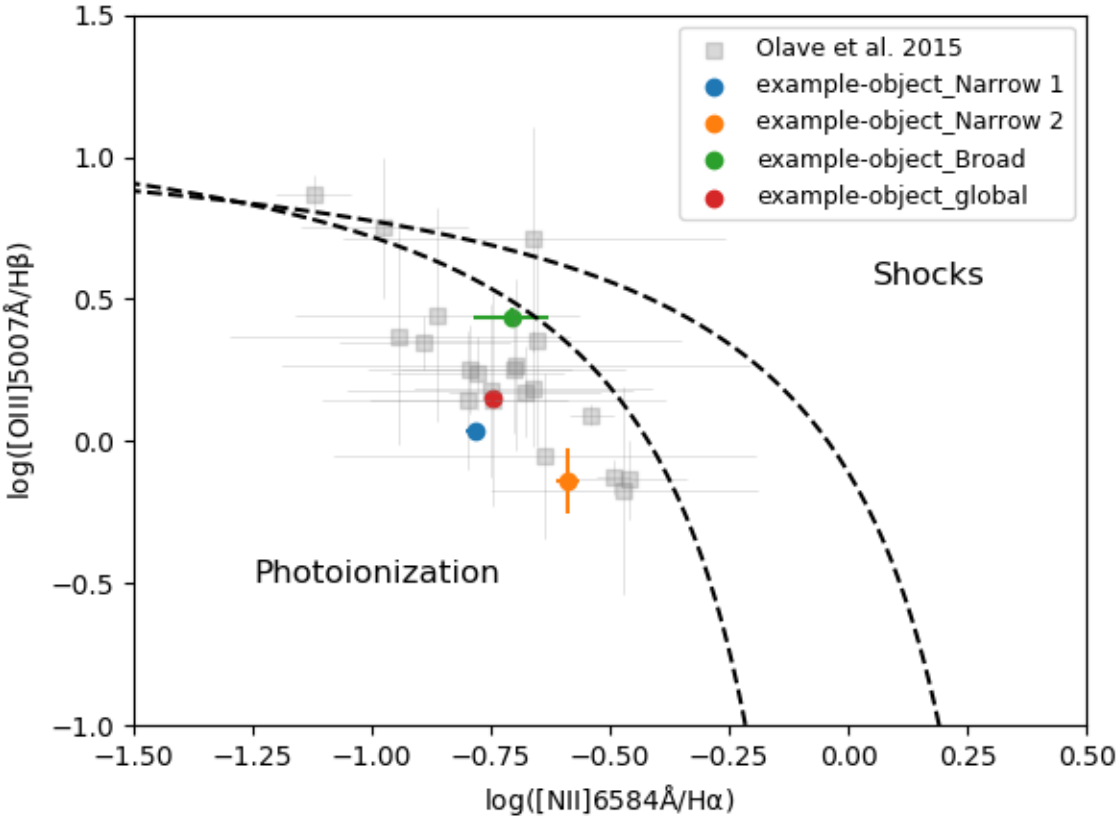
- `fitelp.fit_line_profiles.plot_profiles()`



2.2.2 Make BPT plot

Each object and its components can be added to a BPT plot using the following function. See *Example* for an example of how to add several objects onto different types of BPT plots. An example of a BPT plot output is shown below.

- `fitelp.bpt_plotting.bpt_plot()` - Make different types of BPT plots



2.2.3 Make table of average velocities

A table (in LaTeX and pdf format) of the average velocities of each components and for each object can be created using the following function. An example of this table is shown below.

- *fitelp.make_latex_tables.average_velocities_table_to_latex()*

	example-object	
	v_r (km s^{-1})	σ (km s^{-1})
Narrow 1	3651.6 ± 1.5	35.2 ± 3.9
Narrow 2	3663.3 ± 2.3	13.5 ± 2.1
Broad	3645.5 ± 11.7	68.9 ± 9.0

Table 1: Average of radial velocities and velocity dispersions

2.2.4 Make table of H-alpha luminosity and SFR

A table (in LaTeX and pdf format) of the Star Formation Rate and other information about the objects can be created using the following function. An example of this table is shown below.

- *fitelp.make_latex_tables.halpha_regions_table_to_latex()*

Object Name	SFR ($M_\odot$ yr^{-1})	log(L(Hα))	log([NII]/Hα)	log([OIII]/Hβ)
example-object	0.15 ± 0.001	40.3 ± 0.003	-0.75 ± 0.017	0.15 ± 0.021

Table 1: Object Information

CHAPTER 3

Example

Example script of fitting multiple Gaussian components in the emission-line profiles of a star-forming region.

```
import os
import sys
import numpy as np
import matplotlib.pyplot as plt
import fitelp.constants as constants
from fitelp.make_latex_tables import average_velocities_table_to_latex, halpha_
↪regions_table_to_latex
from fitelp.bpt_plotting import bpt_plot
from fitelp.kinematics_calculations import RegionCalculations
from fitelp.fit_line_profiles import plot_profiles
from fitelp.line_profile_info import RegionParameters

# Path to the directory you wish to save the ouput plots, tables and results.
constants.OUTPUT_DIR = os.path.join(os.path.dirname(os.path.abspath(__file__)),
↪'Output_Files')

# Path to the directory containing your input data files (Optional).
constants.DATA_FILES = os.path.join(os.path.dirname(os.path.abspath(__file__)),
↪'Input_Data_Files')

# Set up example region to simultaneously fit multiple emission lines
example_object = RegionParameters(region_name='example-object',
                                  blue_spec_file='exampleB.fc.fits', #fits file␣
↪path of the blue spectrum
                                  red_spec_file='exampleR.fc.fits', #fits file path of␣
↪the red spectrum
                                  blue_spec_error_file='exampleB_ErrorFlux.fc.fits',
↪#fits file path of the blue spectrum error
                                  red_spec_error_file='exampleR_ErrorFlux.fc.fits', #fits␣
↪file path of the red spectrum error
                                  scale_flux=1e15, # Scales the fluxes
                                  center_list={'low': [3649.11211, 3661.84195, 3648.
↪06497],
```

(continues on next page)

(continued from previous page)

```
                                        'high': [3653.62683, 3664.44579, 3659.
↪42848]}, # Center values of the Gaussian components for each zone
                                sigma_list={'low': [37.8404349, 17.4726107, 73.8997483],
                                            'high': [30.1209633, 14.8025090, 57.
↪1031261]}, # Sigma values of the Gaussian components for each zone
                                lin_slope={'low': -5.8183e-07, 'high': -4.5958e-07}, #␣
↪Linear slope values representing the continuum
                                lin_int={'low': 0.00890187, 'high': 0.00789864}, #␣
↪Linear intercept values representing the continuum
                                num_comps={'low': 3, 'high': 3}, #Number of Gaussian␣
↪components for each zone
                                component_labels=['Narrow 1', 'Narrow 2', 'Broad'],
↪#Labels for each of the gaussian components
                                component_colors=['b', 'r', 'g'], #Colour to plot each␣
↪of the gaussian components
                                plotting_x_range=[3400, 4000], # xrange of velocities␣
↪or delta velocities
                                sigma_instr_blue=4.8, # Instrumental profile on blue arm
                                sigma_inst_red=5.9, # Instrumental profile on red arm
                                distance=52.8,  #Distance to object in Mpc
                                em_lines_for_avg_vel_calc=['H-Alpha', 'OIII-5007A',
↪'NII-6584A', 'SII-6717A'], # List of emission-lines to use to calculate the average␣
↪radial velocity
                                plot_residuals=True,
                                show_systemic_velocity=False, # Assumed False if not␣
↪defined
                                systemic_velocity=3650  # Center of most important␣
↪emission-lines required only if showSystemicVelocity is True
                                )

# Add emission-lines to fit
example_object.add_em_line(name='H-Alpha', plot_color= 'y', order=  20, filter= 'red',
↪ min_idx=  2800, max_idx=3140, rest_wavelength=  6562.82, num_comps=3, amp_list= ␣
↪[44.999084, 18.236959, 9.312178], zone= 'low', sigma_tsquared=  164.96, comp_
↪limits= {'a': np.inf, 'c': np.inf, 's': np.inf}, copy_from= None)
example_object.add_em_line(name='OIII-5007A', plot_color= 'c', order=  4, filter= 'red
↪', min_idx=  2535, max_idx=2870, rest_wavelength=  5006.84, num_comps=3, amp_list= ␣
↪[15.056061, 4.566674, 5.261243], zone= 'high', sigma_tsquared=  10.39,  comp_
↪limits= {'a': np.inf, 'c': np.inf, 's': np.inf}, copy_from= None)
example_object.add_em_line(name='OIII-4959A', plot_color= 'g', order=  4, filter= 'red
↪', min_idx=  1180, max_idx=1510, rest_wavelength=  4958.91, num_comps=3, amp_list= ␣
↪[5.190979, 1.265695, 0.986356], zone= 'high', sigma_tsquared=  10.39, comp_limits= {
↪'a': np.inf, 'c': False, 's': False}, copy_from= 'OIII-5007A')
example_object.add_em_line(name='H-Beta', plot_color='b', order=  36, filter=  'blue',
↪  min_idx= 520,  max_idx= 990, rest_wavelength= 4861.33, amp_list= [8.435867, 1.
↪861033, 3.630495], zone=  'low', sigma_tsquared=  164.96, comp_limits= {'a': np.inf,
↪ 'c': False, 's': False}, copy_from=  'H-Alpha'),
example_object.add_em_line(name='OI-6300A', plot_color= '#D35400', order=  18,␣
↪filter= 'red', min_idx=  2380, max_idx=2700, rest_wavelength=  6300.3, num_comps=3,␣
↪amp_list=  [0.254865, 0.420512, 0.598598], zone= 'low', sigma_tsquared=  10.39,␣
↪comp_limits= {'a': np.inf, 'c': False, 's': False}, copy_from= 'H-Alpha')
example_object.add_em_line(name='NII-6584A', plot_color= 'violet', order=  20,␣
↪filter= 'red', min_idx=  3250, max_idx=3590, rest_wavelength=  6583.41, num_comps=3,
↪ amp_list=  [5.526779, 4.684082, 2.48221], zone= 'low', sigma_tsquared=  11.87,␣
↪comp_limits= {'a': np.inf, 'c':np.inf, 's': np.inf}, copy_from= None)
example_object.add_em_line(name='NII-6548A', plot_color= 'violet', order=  20,␣
↪filter= 'red', min_idx=  2480, max_idx=2820, rest_wavelength=  6548.03, num_comps=3,
↪ amp_list=  [1.729284, 1.613114, 0.821882], zone= 'low', sigma_tsquared=  11.87,␣
↪comp_limits= {'a': np.inf, 'c': False, 's': False}, copy_from= 'NII-6584A')
```

(continues on next page)

(continued from previous page)

```
example_object.add_em_line(name='SII-6717A', plot_color= 'r', order=  22, filter= 'red
↪', min_idx=  530, max_idx=850, rest_wavelength=  6716.47, num_comps=3, amp_list= ␣
↪[4.68714, 2.259787, 1.839585], zone= 'low', sigma_tsquared=  5.19, comp_limits= {'a
↪': np.inf, 'c': np.inf, 's': np.inf}, copy_from= None)
example_object.add_em_line(name='SII-6731A', plot_color= '#58D68D', order=  22,␣
↪filter= 'red', min_idx=  836, max_idx=1150, rest_wavelength=  6730.85, num_comps=3,␣
↪amp_list=  [3.34683, 1.878706, 1.238023], zone= 'low', sigma_tsquared=  5.19,comp_
↪limits= {'a': np.inf, 'c': False, 's': False}, copy_from= 'SII-6717A')

# You may fit multiple objects by adding extra objects to this list
regions_parameters = [example_object,]

region_array = []
rp_bpt_points, rp_bpt_points_s, rp_bpt_points_o, rp_bpt_points_p = [], [], [], []
for rp in regions_parameters:
    region = RegionCalculations(rp, xAxis='vel', initVals='vel')
    region_array.append(region.lineInArray)
    rp_bpt_points.append(region.bptPoints)
    rp_bpt_points_s.append(region.bptPoints_s)
    rp_bpt_points_o.append(region.bptPoints_o)
    rp_bpt_points_p.append(region.bptPoints_p)

    plot_profiles(['H-Alpha', 'OIII-5007A', 'NII-6584A', 'SII-6717A'], rp,␣
↪nameForComps='SII-6717A',
                  title=rp.regionName + ' Strongest Emission Lines', sortedIndex=[0,␣
↪1, 2, 3], logscale=True,
                  ymin=None)

bpt_plot(regions_parameters, rp_bpt_points, plot_type='NII')
bpt_plot(regions_parameters, rp_bpt_points_s, plot_type='SII')
bpt_plot(regions_parameters, rp_bpt_points_o, plot_type='OI')
bpt_plot(regions_parameters, rp_bpt_points_p, plot_type='NIIvsSII')
halpha_regions_table_to_latex(region_array, paperSize='a4', orientation='portrait',␣
↪longTable=False)
average_velocities_table_to_latex(regions_parameters, paperSize='a4', orientation=
↪'landscape', longTable=False)

plt.show()
```

CHAPTER 4

API

An application programming interface (API) defines a set of routines to provides access to functions of a software without using a user interface. FitELP (Fit Emission-Line Profiles) has a few entry points into the API. Following we show the more important classes and methods.

4.1 Basic Classes and Methods

- *fitelp.line_profile_info.RegionParameters* - Setup object properties
- *fitelp.line_profile_info.RegionParameters.add_em_line()* - Add an emission-line
- *fitelp.kinematics_calculations.RegionCalculations* - Calculate object kinematics

4.2 Plotting Methods

- *fitelp.bpt_plotting.bpt_plot()* - Make different types of BPT plots
- *fitelp.fit_line_profiles.plot_profiles()* - Plot emission-line profiles and Gaussian fits

4.3 Latex Tables Methods

- *fitelp.make_latex_tables.average_velocities_table_to_latex()* - Make line profile radial velocities and velocity dispersion table
- *fitelp.make_latex_tables.halpha_regions_table_to_latex()* - Make object information summary table
- *fitelp.make_latex_tables.comp_table_to_latex()* - Make table detailing Gaussian components of each emission-line

The full documentation can be found below.

4.4 Full Documentation

class fitelp.line_profile_info.**RegionParameters**(*region_name*, *blue_spec_file*, *red_spec_file*, *blue_spec_error_file*, *red_spec_error_file*, *scale_flux*, *center_list*, *sigma_list*, *lin_slope*, *lin_int*, *num_comps*, *component_labels*, *component_colors*, *sigma_instr_blue*, *sigma_inst_red*, *distance*, *em_lines_for_avg_vel_calc*, *plotting_x_range=None*, *plot_residuals=True*, *show_systemic_velocity=False*, *systemic_velocity=None*)

List information about a region containing multiple emission lines

region_name [str] Name of object. Do not use underscores in name, as this may affect latex compiling.

blue_spec_file [str] FITS file path of the blue spectrum

red_spec_file [str] FITS file path of the red spectrum

blue_spec_error_file [str] FITS file path of the blue spectrum error

red_spec_error_file [str] FITS file path of the red spectrum error

scale_flux [float] scales the fluxes from the files by this factor during fitting.

center_list [dict] The center values of the gaussian components for the low-ionization (e.g H-Alpha) and high-ionization (e.g [OIII]5007) zones of each gaussian. These values will depend on the type of fit, if it is in velocity or wavelength. E.g. centerList = {'low': [3918.56, 3969.72, 3978.93], 'high': [3923.50, 3970.63, 3984.13]}

sigma_list [dict] The sigma values of the gaussian components for the low-ionization (e.g H-Alpha) and high-ionization (e.g [OIII]5007) zones of each gaussian. These values will depend on the type of fit, if it is in velocity or wavelength. E.g. sigmaList = {'low': [17.123, 13.868, 45.207],'high': [15.740, 12.875, 43.667]}

lin_slope [dict] The linear slope values representing the continuum for the low-ionization (e.g H-Alpha) and high-ionization (e.g [OIII]5007) zones. This value will depend on the type of fit, if it is in velocity or wavelength. E.g. lin_slope = {'low': -5.2237e-08, 'high': -2.8976e-07}

lin_int [dict] The linear intercept values representing the continuum the low-ionization (e.g H-Alpha) and high-ionization (e.g [OIII]5007) zones. This value will depend on the type of fit, if it is in velocity or wavelength. E.g. lin_int = {'low': 0.00139680, 'high': 0.00254310}

num_comps [dict] The number of gaussian components for the low-ionization (e.g H-Alpha) and high-ionization (e.g [OIII]5007) zones. This should be the length of the lists of the center and sigma of each of the gaussian components E.g. num_comps = {'low': 3, 'high': 3} or num_comps = {'low': 3, 'high': 5}

component_labels [list] Labels for each of the gaussian components in the order that they are presented in all other lists. E.g. component_labels = ['Narrow 1', 'Narrow 2', 'Broad']

component_colors [list] Colour to plot each of the gaussian components in the order that they appear in component_labels. E.g. componentColours = ['r', 'c', 'g']

sigma_instr_blue [float] The instrumental profile (σi) in the blue-arm of the spectrograph. It is well approximated by a single Gaussian function.

sigma_inst_red [float] The instrumental profile (σi) in the red-arm of the spectrograph. It is well approximated by a single Gaussian function.

distance [float] The distance to the region in centimetres (same units that distance appears in the input flux)

em_lines_for_avg_vel_calc [list] The emission lines to use to calculate the average velocity. E.g. emLinesForAvgVelCalc = ['H-Alpha', 'H-Beta', 'OIII-5007A', 'NII-6584A', 'SII-6717A']

plotting_x_range: list or None The wavelength (or velocity) range to plot each of the emission line gaussians. E.g. plotting_x_range = [3600, 4400]

plot_residuals [bool] Whether to plot the residuals of the fit in an extra panel below the gaussian Default is True.

show_systemic_velocity [bool] Assumed False if it is not defined. If True, the xAxis is plotted as the measured velocity minus the systemicVelocity: (velocity - systemicVelocity)

systemic_velocity [float or None] Required if show_systemic_velocity is True.

add_em_line(*name*, *plot_color*, *order*, *filter*, *min_idx*, *max_idx*, *rest_wavelength*, *amp_list*, *zone*, *sigma_tsquared*, *comp_limits*, *copy_from*, *num_comps=None*)

Emission line info

name [str] Name of emission line. E.g. 'H-Alpha' or NII-6584A. This name must be in the appropriate format. This name must be in the appropriate format (see ALL_IONS list in constants.py).

plot_color [str] The color that this emission line should appear in each of the plots.

order [int] The order that this emission line appears in the Echelle FITS files or 1 for Longslit FIT file.

filter [str] 'red' or 'blue'. Indicating the red-arm or blue-arm of the spectrograph where the emission line appears in the spectrum.

min_idx [int] Minimum index of the region in the spectra that includes this emission line.

max_idx [int] Maximum index of the region in the spectra that includes this emission line.

rest_wavelength [float] The rest wavelength of this emission line.

num_comps [int (optional)] The number of Gaussian components to fit the emission line. This overrides the num_comps set in RegionParameters for this emission line only.

amp_list: list or float The list of amplitudes for each of the components that are included. This must be a list with the same number of elements as 'num_comps'. If this is a float, then the amplitudes from the emission line listed in the copy_from parameter will be used and divided by this value.

zone [str] 'low' or 'high' ionization zone. Two-ionization-zone scheme is assumed: the low-ionization zone where the hydrogen recombination lines and [OI], [OII], [NII], [SII] forbidden lines are emitted, and the high-ionization zone where the helium recombination lines and [OIII], [SIII], [ArIII], [NeIII] forbidden lines are emitted (Hägele et al., 2012).

sigma_tsquared [float] squared of the random thermal motion (σt). In the example, the thermal contribution was derived from the Boltzmann's equation, assuming a typical kinetic temperature T10^4K and the atomic mass of the corresponding element.

comp_limits [dict] The limits in 'compLimits' can be in the following forms: - a list indicating the limits for each component - a single number indicating the percentage limits for ALL components - a tuple (minValue, maxValue) indicating the min and max not in a percentage - inf: indicating that the component can vary - False: indicating that the value is fixed

copy_from [str or None or list] The name of the emission line to copy from. If None, it will not copy any information. If it is a list, it must be the length of the number of components you have (as defined in num_comps in RegionParameters). Each element of the list must be a string indicating which emission line to copy for each component.

class fitelp.kinematics_calculations.**RegionCalculations**(*rp*, *xAxis='vel'*, *initVals='vel'*)

Compute kinematics of a region.

rp [RegionParameters object] An instance of the RegionParameters class.

xAxis [str] Plots the x axis in velocity space if xAxis='vel' and in wavelength space if xAxis='wave'. Default is 'vel'.

initVals [str] Interprets the initial values from all the parameters (i.e. center, sigma, amplitude) as velocities if initVals='vel' or as wavelengths if initVals='wave'

---

fitelp.bpt_plotting.**bpt_plot**(*rpList*, *rpBptPoints*, *globalOnly=False*, *plot_type='NII'*)

---

fitelp.fit_line_profiles.**plot_profiles**(*lineNames*, *rp*, *nameForComps=''*, *title=''*, *sortedIndex=None*, *plotAllComps=False*, *xAxis='vel'*, *logscale=False*, *ymin=None*)

---

fitelp.make_latex_tables.**average_velocities_table_to_latex**(*rpList*, *directory=None*, *paperSize='a4'*, *orientation='portrait'*, *longTable=False*)

---

fitelp.make_latex_tables.**halpha_regions_table_to_latex**(*regionInfoArray*, *directory=None*, *paperSize='a4'*, *orientation='portrait'*, *longTable=False*)

---

fitelp.make_latex_tables.**comp_table_to_latex**(*componentArray*, *rp*, *paperSize='a4'*, *orientation='portrait'*, *longTable=True*, *xAxisUnits='vel'*, *scaleFlux=100000000000000.0*)

CHAPTER 5

Contribute

- Issue Tracker: https://github.com/daniel-muthukrishna/FitELP/issues
- Source Code: https://github.com/daniel-muthukrishna/FitELP

CHAPTER 6

Support

If you are having issues, please let us know by submitting a GitHub issue at https://github.com/daniel-muthukrishna/FitELP/issues

CHAPTER 7

License

The project is licensed under the MIT license.

CHAPTER 8

Authors

Daniel Muthukrishna, Verónica Firpo

Index