LZIFU v1.0 user manual

I-Ting Ho

Institute for Astronomy, University of Hawai'i

Research School of Astronomy and Astrophysics, Australian National University

Max-Planck Institute for Astronomy

August 16, 2016

Contents

1 Introduction

LZIFU ("LaZy-IFU") is an emission line fitting pipeline for integral field spectroscopy (IFS) data. This IDL pipeline was developed as part of my thesis work and is designed to turn IFS data to 2D emission line flux and kinematic maps for further analysis. LZIFU has been successfully applied to data from various IFS instruments, most notably the SAMI Galaxy Survey (Bryant et al., 2015; Allen et al., 2015). Some of the science results published with LZIFU can be found in Ho et al. (2014, 2015, 2016); Vogt et al. (2015); Dopita et al. (2015).

This main purpose of this manual is to explain how to install, set up, and run LZIFU. Details of how the pipeline functions are described in Ho et al. 2016, Ap&SS, 361, 280.

Note that although LZIFU has already been used and tested by several astronomers, there could still be bugs in the code. This program is freely available, but it comes without any warranty, i.e. use it at your own risk. If you find any problems with the code or have any suggestions, I would appreciate you reporting them to `itho@ifa.hawaii.edu`.

2 Installation

LZIFU uses some common libraries that could exist already in your computer (including the SolarSoft system, parallelidl, MPFIT, Coyote library, etc.). To avoid crosstalk between different versions of some libraries, it is recommended that you follow the guide below to ensure that your `$IDL_PATH` does not contain libraries outside the scope of LZIFU.

1. Make sure you have IDL 8.2 or above

2. Download LZIFU from github (`https://github.com/hoiting/LZIFU/releases/`) and unzip it to your favorite location, e.g. `/home/ast1/packages/LZIFU-1.0/`.

3. Download PPXF 2016/Jan/21 version from `http://www-astro.physics.ox.ac.uk/~mxc/software/ppxf_2016-01-21.zip`. Download `bvls.pro` from `http://www-astro.physics.ox.ac.uk/~mxc/software/bvls.pro`. Place them under `/home/ast1/packages/LZIFU-1.0/pro/`.

4. Download the IDL Astronomy Users Library from `http://idlastro.gsfc.nasa.gov/` and untar it to `/home/ast1/packages/LZIFU-1.0/pro/`.

5. Setup an alias in your shell startup file (.cshrc .tcshrc for csh, tcsh)

```
alias startlzifuv1.0 'setenv IDL_PATH  +/home/ast1/packages/
   LZIFU-1.0/pro/\:+/usr/local/itt/idl/lib/'
```

or (.basrc for bash)

```
alias startlzifuv1.0='export IDL_PATH=+/home/ast1/packages/
   LZIFU-1.0/pro/:+/usr/local/itt/idl/lib/'
```

Modify the last part of the alias to match the path of your IDL *installation folder.*

6. Setup the path every time you want to run LZIFU:

```
> startlzifuv1.0
```

7. Run the example scripts to make sure everything works
Examples: basic SAMI data (`LZIFU-1.0/examples/511867/scripts/`) or basic CALIFA data (`LZIFU-1.0/examples/NGC0776/scripts/`). A more sophisticated example is also provided: `LZIFU-1.0/examples/209807/`.

3 How to run LZIFU?

3.1 Input data in FITS format

You need to restructure your data to the input FITS format required by LZIFU (see Table 1). The zeroth extension header should contain keywords for the wavelength vector in the third dimension, i.e. `NAXIS3, CRPIX3, CDELT3, CRVAL3`. Each data cube should be on a regular, linear spectral grid and have a wavelength-independent instrumental dispersion. For two-sided data, the blue and red sides can have different dispersions. In addition, you need to name your FITS file as `set.obj_name_B.fits` and `set.obj_name_R.fits` if you have two-sided data, and `set.obj_name.fits` if your data are one-sided (`set.obj_name` is a string parameter in the setting file, typically the object ID; see below).

3.2 Prepare and run scripts

1. Create a project folder with a data folder, a product folder, and a script folder under it, i.e.

```
> mkdir project_ifs
> mkdir project_ifs/data/
> mkdir project_ifs/products/
> mkdir project_ifs/scripts/
```

2. Place your data cubes in `project_ifs/data/`

3. Copy the setting and linelist scripts from your LZIFU folder to your project script folder, i.e.

```
> cp $PATH_TO_LZIFU/lzifu.pro project_ifs/scripts/
> cp $PATH_TO_LZIFU/lzifu_linelist.pro project_ifs/scripts/
```

4. Make the setting and linelist scripts project-specific by first changing both their filenames and the first lines of the scripts (i.e. routine names).

```
> cd project_ifs/scripts/
> mv lzifu.pro lzifu_project.pro
> mv lzifu_linelist.pro lzifu_linelist_project.pro
```

Table 1 Input data format

Extension	Content	Note
0 (Primary)	flux cube	In unit of *const.* $\times$ erg s^{-1} cm^{-2} $Å^{-1}$. Remove the integer exponent in the data to prevent numerical issues (e.g. provide 3.52, instead of 3.52×10^{-17}.)
1	variance cube	In unit of (cube unit)^2
2	bad pixel cube	0 = good pixel. 1 = bad pixel. NaNs in the flux and variance cubes are automatically considered as bad pixels.

Change the first line of `lzifu_project.pro` from `[PRO lzifu,...]` to `[PRO lzifu_project,...]`, and the first line of `lzifu_linelist_project.pro` from `[FUNCTION lzifu_linelist,...]` to `[FUNCTION lzifu_linelist_project,...]`

5. Edit the setting and linelist files to match your purpose (see Section 4).

6. Run LZIFU under `project_ifs/scripts/`

   - Method 1 (good for single object): under the script folder, directly execute LZIFU in IDL :

   ```
   IDL> lzifu_project
   ```

   - Method 2 (good for multiple objects from the same survey): under the script folder, call your LZIFU script and pass other object-specific properties as arguments, e.g.

   ```
   IDL> id_arr = ['NGC1234', 'NGC5678', 'NGC4321']
   IDL> z_arr = [0.1234, 0.5678, 0.4321]
   IDL> for i = 0,2 do lzifu_project,obj_name=id_arr[i],z=
      z_arr[i]
   ```

   Note that you can pass all settings as arguments, but only if you provide the full names for the parameters. The keyword abbreviation rule of IDL does not apply here.

7. Wait for the script to finish and voilà! The results are stored in the output folder `project_ifs/products/`.

3.3 Output data format

The outputs are stored in a multi-extension FITS file that is placed at a location specified by you in the setting file (see below for `set.product_path`). If you follow the standard setup described in Section 3.2, then you can find the output FITS file under `project_ifs/products/`. The output filename will be `set.obj_name_1_comp.fits`, `set.obj_name_2_comp.fits`, or `set.obj_name_3_comp.fits`, depending on the number of Gaussian components you specified in the setting file (see below for `set.ncomp`).

The primary (zeroth) extension of the FITS file contains no actual data, but the header provides important information about the galaxy. These keywords are listed and explained in Table 2.

LZIFU stores fitting products starting from the first extension. Depending on the setting, each output file may contain different numbers of extensions (e.g. whether external continuum cubes are provided, how many lines are fit, etc.). The content of each extension is usually self-explanatory by the `EXTNAME` keyword in the header of each extension, or by the `EXT‘‘X’’` keywords in the header of the zeroth extension. A more detailed explanation of each `EXTNAME` is given in Table 3 and in the following subsections.

As a technical note, extension numbers may change from file to file, especially in the future release where potentially more information will be included. We will not change the `EXTNAMEs`. When scripting your analysis, it would therefore be wise to read in the FITS files with `EXTNAMEs` rather than extension numbers.

Several things that the users should be aware of. First, to streamline batch processes, LZIFU overwrites existing files without any warning. Second, the output files could take up a substantial amount of disk space due to the 3D models stored in the first few extensions. The file size can be made much smaller if some of the 3D extensions unimportant to your analysis are removed.

3.3.1 3D cubes

`(BR_)CONTINUUM`, `(BR_)ADDPOLY`, `(BR_)MPOLY`, `(BR_)RESIDFIT`, `(BR_)CONT_MASK`, `(BR_)LINE`, and `(BR_)LINE_COMP1(23)` are 3D cubes with the same dimensions and unit as the input data. These products are usually most useful for plotting purposes and visualizing the fit. They can also be used to obtain continuum-free or line-free data cubes, i.e. by subtracting the model cubes from the data cubes. Those related to continuum fitting are also useful for gauging potential systematic effects in the fitting that may impact your science (e.g. do the additive polynomials contribute sig-

Table 2 Header of 0th extension

Keyword	Content
`OBJECT`	`set.obj_name`
`Z_LZIFU`	Redshift of the galaxy given as input to LZIFU. Anchor point of all kinematic measurements
`VERSION`	Version of the LZIFU code
`NCOMP`	Number of components used (1, 2, or 3)
`SORTTYPE`	Component sorting method (see `set.sort_type`)
`EXT‘‘X’’`	`EXTNAME` of the “X” extension. Same information is also stored in `EXTNAME` in the header of each extension

Table 3 EXTNAME

EXTNAME	Unit	Content
		3D cubes
(BR_)CONTINUUM	CU[1]	Continuum cubes (with all the polynomial fits included)
(BR_)ADDPOLY	CU	Additive polynomials fit simultaneously with stellar templates, as controlled by `degree`.
(BR_)MPOLY	CU	Multiplicative polynomials fit simultaneously with stellar templates, as controlled by `mdegree`. This is the best-fit extinction curve when `mdegree` < 1 (fit reddening).
(BR_)RESIDFIT	CU	Polynomial fit to the continuum residual. The degree of polynomials is controlled by `(br_)resid_degree`.
(BR_)CONT_MASK	1/0	1 = channel used in continuum fitting; 0 = channels not included in the fit
(BR_)LINE	CU	Line model cubes. Sum of all components
(BR_)LINE_COMP1(23)	CU	Line model cubes of individual components
		M2D maps[2]
V, V_ERR	km s^{-1}	Velocities and errors of individual components
VDISP, VDISP_ERR	km s^{-1}	Velocity dispersions and errors of individual components with instrumental resolution removed.
LINE[3], LINE_ERR	CU $\times$ Å	Fluxes and errors for "LINE" of total (zeroth slice) and individual components
		2D maps
CHI2	2D	Reduced-χ^2 of the line fit
DOF	2D	Degrees of freedom of the line fit
		Others
SET	–	The `set` parameter of this run, defined in the setting file and stored as a BINTABLE.

[1] CU: Unit used in the data cubes. See Table 1.

[2] M2D = Multiple 2D images (see Section 3.3.2).

[3] LINE: lines fit to the data provided by the user in `set.linelist_func`. The line labels provided in the `fit_line` section of `set.linelist_func` become the header names.

nificantly to the continuum model such that Balmer corrections may be significantly under- or over-estimated?).

3.3.2 Multiple 2D images (M2D)

All the flux maps, velocity maps, velocity dispersion maps, and their corresponding error maps are stored as stacks of 2D images. For an `NCOMP` fit, its M2D has (`NCOMP+1`) slices. The second slice (index = 1 in IDL) records results of the first component; the third slice (index = 2) and the fourth slice (index = 3) record results of the second and third components, respectively.

The ordering of the different components are decided by the `set.sort_type` keyword.

The first slice (index = 0) records the "total" of all components. In the case of fluxes, this is a direct sum of the values of all the components. In the case of errors, this is a combined error of all the components with the covariances propagated through. In the trivial case of a 1-component fit, the first slice is exactly the same as the second slice.

For velocity and velocity dispersion maps (and their error maps), their first slices are always empty (NaN) because it is meaningless to sum these quantities.

4 Setting file (`lzifu.pro`)

The parameter list below may seem long but it is not as complicated as it appears.

4.1 `only_1side` [integer]

0: two-sided data. 1: one-sided data

`only_1side` tells LZIFU whether the data provided are one-sided (e.g. MUSE, MaNGA) or two-sided (e.g. SAMI, WiFeS). If one-sided mode is selected, then all the parameter pairs for blue and red sides are ignored (e.g., `[br]_resol_sigma` ignored, `resol_sigma` adopted; `[br]_ext_mask` ignored, `ext_mask` adopted, etc.)

4.2 `obj_name` [string]

Name of the object, typically object ID. The ID needs to match the filename(s) of your input FITS file(s), i.e. if `obj_name='NGC9999'`, then input files should be named as `NGC9999_B.fits`, `NGC9999_R.fits`, or `NGC9999.fits`. This string will be used in the output filenames as well.

4.3 `z` [float]

Redshift of the object. Anchor point of all kinematic measurements. This number will be also stored in `Z_LZIFU` in the header of the zeroth extension (see Table 2).

4.4 `fit_ran` [float array[2]] unit:Å

Lower (`fit_ran[0]`) and upper bounds (`fit_ran[1]`) of the wavelength range to be considered by LZIFU.

4.5 `b_ext_mask`, `r_ext_mask`, `ext_mask` [float array[2,n]] unit:Å

External masks marking the undesired wavelength ranges for the blue data (`b_ext_mask`) and the red data (`r_ext_mask`) when running in two-sided mode (`only_1side=0`), or for the data cube (`ext_mask`) in the one-sided mode (`only_1side=1`). Channels in between each wavelength range pairs (`ext_mask[0,i]` to `ext_mask[1,i]`) are not considered. These masks are applied to all spaxels, and are useful for removing channels contaminated by strong sky emission lines.

4.6 `b_resol_sigma`, `r_resol_sigma`, `resol_sigma` [float] unit: Å

Spectral resolution of the data described by σ assuming a Gaussian spectral PSF. Each data cube is assumed to have a uniform spectral resolution across the entire wavelength range.

4.7 External continuum model

If you do not want to use PPXF to fit the continuum and already have a continuum model, you can provide your continuum model to LZIFU. LZIFU will simply subtract your model from the data and go straight to fitting the emission lines. Your continuum model cubes will also be included in the multi-extension FITS output.

To input your model, your continuum cube(s) should be stored in a single FITS file. If you are fitting two-sided data, then your blue continuum model should be in the primary extension (zeroth extension), and the red continuum model in the first extension. If you are fitting one-sided data, then your continuum model should be in the primary extension (zeroth extension). Your 3D model(s) should have the same dimensions as your data cube(s).

4.7.1 supply_ext_cont [integer]

0: No, I want to use PPXF to fit the continuum.
1: Yes, I want to provide a continuum model.

This keyword indicates whether you want to provide a continuum model for continuum subtraction.

4.7.2 ext_cont_path [string]

Data path to your continuum model.

4.7.3 ext_cont_name [string]

Filename of your continuum model cube (FITS file).

4.7.4 load_ext_cont_func [string]

`default='lzifu_load_ext_cont'`

Name of the routine for loading the continuum model(s). Does not need to be changed if following the convention described above.

4.8 External 2D mask

You can provide a 2D mask to tell LZIFU to skip some of the spaxels. The 2D mask should be in FITS format, has the same spatial dimensions as your data cube(s), and contains only two values, 0 and 1.

1 = I want to fit this spaxel.
0 = I don't want to fit this spaxel.

4.8.1 supply_mask_2d [integer]

0: don't use external 2D mask.
1: use external 2D mask.

This keyword indicates whether you want to use 2D mask to skip some spaxels.

4.8.2 mask_2d_path [string]

Data path to your 2D mask file.

4.8.3 mask_2d_name [string]

Filename of your 2D mask (FITS file).

4.8.4 load_mask_2d_func [string]

default='lzifu_load_2d_mask'

Name of the routine for loading the 2D mask. Does not need to be changed if following the convention described above.

4.9 Data I/O

`data_path` and `product_path` tell LZIFU where to find the input data and store the output.

4.9.1 data_path [string]

Path to the data cube(s). If you follow the setup in Section 3.2, this should be `'../data/'`.

4.9.2 product_path [string]

Path to the output product(s). If you follow the setup in Section 3.2, this should be `'../products/'`.

4.9.3 load_cube_func [string]

default='lzifu_load_cube'

Name of the routine for loading the data cube(s). Does not need to be changed if following the convention described in Section 3.1

4.10 Simple stellar population templates

If you want to use PPXF to fit the continuum, you need to select a set of simple stellar population templates to perform the stellar population synthesis. Some pre-compiled templates are listed in Table 4. These are directly available under the `stellar_models/` folder as IDL savefiles. Note that you need to pick templates that cover your spectral range, and have a better spectral resolution than your data.

Table 4 Stellar population template

Name (.sav)	Ingredient	Age	Metallicity	Wavelength range	`temp_resol_sigma`	Reference
`lzifu/stellar_models/gonzalezdelgado/`						
SSPGeneva[b]	Theoretical templates. Geneva isochrones.	1 Myr to 10 Gyr in 46 bins (those with prefix "cond_" are 1 Myr to 10 Gyr in 14 bins)	twice solar (z040), solar (z020), half (z008, z004[a]) and 1/10 solar (z001)	3000 – 7000 Å	0.3 Å	González Delgado et al. (2005)
SSPPadova[b]	Theorietical template. Padova isochrones.	4 Myr to 18 Gyr in 74 bins (those with prefix "cond_" are 4 Myr to 11 Gyr in 24 bins)	solar (z019), and half solar (z008, z004[a])	3000 – 7000 Å	0.3 Å	González Delgado et al. (2005)
`lzifu/stellar_models/miles_salpeter/`						
miles[c]	Empirical templates. Padova isochrones. Salpeter IMF.	63 Myr to 18 Gyr in 50 bins (those with prefix "cond_" are 63 Myr to 16 Gyr in 13 bins)	[M/H] = -1.31 (m1.31), -0.71 (m0.71), -0.40 (m0.40), 0 (p0.00), and 0.22 (p0.22).	3540 – 7410 Å	1.07Å	Vazdekis et al. (2010)
`lzifu/stellar_models/miuscat_salpeter/`						
miuscat[c]	Empirical templates. Padova isochrones. Salpeter IMF.	63 Myr to 18 Gyr in 50 bins (those with prefix "cond_" are 63 Myr to 16 Gyr in 13 bins)	[M/H] = -0.71 (m0.71), -0.40 (m0.40), 0 (p0.00), and 0.22 (p0.22).	3465 – 9469 Å	1.07Å	Vazdekis et al. (2012)

[a] See section 3.2 of González Delgado et al. (2005) for details about the metallicities of the isochrones and stellar atmosphere adopted
[b] `http://www.iaa.es/~rosa/research/synthesis/HRES/ESPS-HRES.html`
[c] `http://miles.iac.es/`

4.10.1 template_path [string]

Path to the stellar population templates.

4.10.2 template_name [string array [n_{metal}]]

The filenames of the templates. The files should come from the same group in Table 4 such that the templates have the same ingredients, numbers of age bins, wavelength range, etc. You can chose how many different metallicities you want to include (n_{metal}), and provide the corresponding filenames in an 1-D string array, e.g. `['cond_SSPPadova_z004.sav', 'cond_SSPPadova_z008.sav', 'cond_SSPPadova_z019.sav']`.

4.10.3 temp_resol_sigma [float] unit:Å

Spectral resolution (σ) of the stellar population template selected. See Table 4.

4.11 Continuum fitting (with PPXF)

4.11.1 mask_width [float] unit: Å

Full width around the emission line centers (as suggested by `z`) to be masked out prior to fitting the continuum. The lines to mask out are provided by `linelist_func` (see `linelist_func` for the `mask_line` section).

4.11.2 cont_vel_sig_guess [float array[2]] unit: [km s^{-1}, km s^{-1}]

Initial guesses for the velocity `cont_vel_sig_guess[0]` and velocity dispersion `cont_vel_sig_guess[1]` passed to PPXF.

4.11.3 cont_ebv_guess [float]

Initial guess for the E(B-V) value pased to PPXF. See the `REDDENING` parameter in PPXF.

4.11.4 degree [integer]

Degree of additive Legendre polynomials to be fit simultaneously with stellar templates. See the `DEGREE` keyword in PPXF. `degree=` -1 means no additive polynomials are included.

4.11.5 `mdegree` [integer]

Degree of the multiplicative Legendre polynomials to be fit simultaneously with stellar templates. `mdegree`= 0 means no multiplicative polynomials are included. Negative values are not allowed and will be set to zero automatically. If multiplicative polynomials are included then extinction is not fit (see the `MDEGREE` keyword in PPXF).

4.11.6 `moments` [integer]

Order of the Gauss-Hermite moments. See the `MOMENTS` keyword in PPXF. Allowed values are 2, 4 and 6. Typically, `moments=2` works well for emission line studies with IFS.

4.11.7 `b_resid_degree`, `r_resid_degree`, `resid_degree` [integer]

Degrees of additive Legendre polynomials fit to the continuum-subtracted spectra. This is different from fitting polynomials *simultaneously* with stellar templates to the data, which is what PPXF does with `degree` and `mdegree`. If smaller than 1, no additional polynomials are fit to the continuum-subtracted spectra.

4.11.8 `ppxf_clean` [integer]

Passed to the PPXF CLEAN keyword. 1 = clean, 0 = don't perform clean.

4.12 Emission line fitting

The parameters below control the emission line fitting parts of the pipeline.

4.12.1 `fit_dlambda` [float] unit: Å

Full width around line centers to be fit. This number depends on your instrumental resolution. The width should be large enough to cover the emission lines but not so large as to covers too many channels that contain no signal.

4.12.2 `ncomp` [integer]

Number of kinematic components. Allowed values are 1, 2, or 3.

4.12.3 `line_sig_guess` [float] unit:km s^{-1}

Initial guess of the velocity dispersion (σ) of the emission lines. Typically, 50 km s^{-1} is a good starting point.

4.12.4 `vdisp_ran` [float array[2]] unit:km s^{-1}

Lower (`vdisp_ran[0]`) and upper (`vdisp_ran[1]`) bounds for velocity dispersion of the emission lines, applied to all lines and all components. Because of the reflective boundary condition, it is recommended to set the lower bound to a negative value, like -50 km s^{-1}, rather than 0 km s^{-1}. This helps the fit to converge and will not affect your velocity dispersion map. LZIFU will still report positive velocity dispersion.

4.12.5 `vel_ran` [float array[2]] unit:km s^{-1}

Lower (`vel_ran[0]`) and upper (`vel_ran[1]`) bounds for the velocity of the emission lines, applied to all lines and all components.

4.12.6 `sort_type` [string]

> `='vdisp'`, `'vel'` or one of the `labels` in `linelist_func` `fit_line` section.

Which reference value is used to assign different components in the output maps and cubes. Three sorting methods are provided, i.e. sorted by velocity dispersion (`='vdisp'`), velocity (`='vel'`) or line flux (one of the `labels` in `linelist_func` (`fit_line` section). In the cases of `='vdisp'` and `='vel'`, the first component has the smallest reference value, i.e. ascending order. In the case of sorting with line flux, the first component has the largest reference value, i.e. descending order. For example, if `sort_type='HALPHA'`, then the first component has the largest Hα flux value of all the components in that pixel. If `sort_type='vel'`, then the first component has the smallest velocity.

4.12.7 `comp_2_damp` [float array[n]]

The parameter controlling the variation of initial guesses for the amplitude of the second component, the f_{A12} parameter in Figure 2 of Ho et al. (2016 submitted). Details can be found in their Section 2.3.1.

4.12.8 comp_2_dvel [float array[n]] unit:km s^{-1}

The parameter controlling the variation of initial guesses for the velocity of the second component, the Δv_{12} parameter in Figure 2 of Ho et al. (2016 submitted). Details can be found in their Section 2.3.1.

4.12.9 comp_2_dvdisp [float array[n]] unit:km s^{-1}

The parameter controlling the variation of initial guesses for the velocity dispersion of the second component, the $\Delta\sigma_{12}$ parameter in Figure 2 of Ho et al. (2016 submitted). Details can be found in their Section 2.3.1.

4.12.10 comp_3_damp [float array[n]]

The parameter controlling the variation of initial guesses for the amplitude of the third component, the f_{A13} parameter in Figure 2 of Ho et al. (2016 submitted). Details can be found in their Section 2.3.1.

4.12.11 comp_3_dvel [float array[n]] unit:km s^{-1}

The parameter controlling the variation of initial guesses for the velocity of the third component, the Δv_{13} parameter in Figure 2 of Ho et al. (2016 submitted). Details can be found in their Section 2.3.1.

4.12.12 comp_3_dvdisp [float array[n]] unit:km s^{-1}

The parameter controlling the variation of initial guesses for the velocity dispersion of the third component, the $\Delta\sigma_{13}$ parameter in Figure 2 of Ho et al. (2016 submitted). Details can be found in their Section 2.3.1.

Note that the computation time increases quickly if one has too many initial guess variations. This can be a problem especially for 3-component fitting. The total number of iterations is the product of all the numbers of variations (i.e. N[comp_2_damp]× N[comp_2_dvel]×N[comp_2_dvdisp]×N[comp_3_damp]×N[comp_3_dvel]×N[comp_3_dvdisp]).

4.12.13 n_smooth_refit [integer]

Number of refits desired (≥ 0). In the refitting process, results from the previous fit are spatially median-smoothed to produce initial guesses to refit the data. This process helps the fit convergence, especially for hot spaxels. Typically n_smooth_refit=1 or 2 is sufficient. Details can be found in Section 2.3.2 of Ho et al. (2016 submitted). No refit is done if `n_smooth_refit=0`.

4.12.14 `smooth_width` [integer] unit: spaxel

The width over which the previous fit is median-smoothed. Typically, smoothing by 3×3 (`smooth_width=3`) or 5×5 (`smooth_width=5`) is sufficient.

4.12.15 `linelist_func` [string]

The name of the linelist function that tells LZIFU which lines to fit and which lines to mask out when fitting the continuum. In the example in Section 3.2, `linelist_func` should be set as `'lzifu_linelist_project'`. One can set up different linelist functions under the same project to fit different sets of lines.

The linelist function has two sections, the `mask_line` section and the `fit_line` section. The `mask_line` section contains a list of lines that will be masked out in the continuum fitting process. The width to mask out is controlled by the `mask_width` parameter. The `fit_line` section contains the lines that one wants to fit with Gaussians. One has to modify the `labels` and `waves` arrays, and makes sure the two always match up. The labels are used as the extension names in the output and should not contain any special characters.

The following sets of lines are tied (internally by LZIFU) to the flux ratios given by quantum mechanics (Osterbrock & Ferland, 2006), and only the stronger lines will be included in the output, i.e. OIII5007 and NII6583. LZIFU only does this when the correct line labels below are given.

OIII5007 = $2.94 \times$ OIII4959
NII6583 = $3.06 \times$ NII6548

4.13 `ncpu` [integer]

Number of CPUs used (positive integer). Recommended to be less than 15 (unstable if greater than 15; possible with newer versions of IDL). In single CPU mode (`ncpu=1`), IDL produces useful error messages. In multi-CPU mode (`ncpu`$\geq$`1`), the error messages are usually not very informative. If your program crashes, you should try to run it with a single CPU to debug and then return to multi-CPU mode later for speed improvement.

5 Known issues

5.1 Non-zero covariance

When a line is fit with more than one component, the fluxes of different components are highly correlated. Typically the correlation coefficients are negative because the total flux, the sum of all components, remains approximately the same. This means when summing the fluxes of two components together, the combined error is not the quadrature sum of the errors. The error of the sum should be taken directly from the first slice in M2D error extensions, which takes the covariances into account. A more elegant way of reporting the covariance is not yet available, and it is unclear whether this quantity is required by any science cases. In simple cases (i.e., 2 components) the users can calculate backwards to get the covariance if necessary.

5.2 Errors of [OII]3726,31 doublets (and other marginally resolved line pairs)

For most emission lines, the wavelength separations between different lines are larger than the spectral resolutions, such that fluxes between different lines are independent of each other, i.e. the covariances are zero. This is usually not true for the [OII]3726,31 doublets. The doublets are typically marginally resolved and therefore non-zero covariance can be important. Currently, the covariances are not reported and therefore directly summing the flux errors (of the two lines) in quadrature would overestimate the errors.

In some cases where signal-to-noise ratio is poor, LZIFU might use only one Gaussian to model one of the lines and report zero flux for the other. In this case, the error on the line with zero flux would be NaN (parameter touches boundaries) and the error on the non-zero line will be a more representative estimate.

6 References for Publications

If you find LZIFU useful, please reference the following papers in your publications.

LZIFU: Ho et al. 2016, Ap&SS, 361, 280 [ADS]

MPFIT: Markwardt, C. B. 2009, in Astronomical Society of the Pacific Conference Series, Vol. 411, Astronomical Data Analysis Software and Systems XVIII, ed. D.

A. Bohlender, D. Durand, & P. Dowler, 251 [ADS]

If you use PPXF for your continuum fit, please kindly remember to cite:
PPXF: Cappellari, M., & Emsellem, E. 2004, PASP, 116, 138 [ADS]

References

Allen, J. T., Croom, S. M., Konstantopoulos, I. S., et al. 2015, MNRAS, 446, 1567

Bryant, J. J., Owers, M. S., Robotham, A. S. G., et al. 2015, MNRAS, 447, 2857

Dopita, M. A., Ho, I.-T., Dressel, L. L., et al. 2015, ApJ, 801, 42

González Delgado, R. M., Cerviño, M., Martins, L. P., Leitherer, C., & Hauschildt, P. H. 2005, MNRAS, 357, 945

Ho, I.-T., Kudritzki, R.-P., Kewley, L. J., et al. 2015, MNRAS, 448, 2030

Ho, I.-T., Kewley, L. J., Dopita, M. A., et al. 2014, MNRAS, 444, 3894

Ho, I.-T., Medling, A. M., Bland-Hawthorn, J., et al. 2016, MNRAS, 457, 1257

Osterbrock, D. E., & Ferland, G. J. 2006, Astrophysics of gaseous nebulae and active galactic nuclei (University Science Books, Mill Valley, CA)

Vazdekis, A., Ricciardelli, E., Cenarro, A. J., et al. 2012, MNRAS, 424, 157

Vazdekis, A., Sánchez-Blázquez, P., Falcón-Barroso, J., et al. 2010, MNRAS, 404, 1639

Vogt, F. P. A., Dopita, M. A., Borthakur, S., et al. 2015, MNRAS, 450, 2593