Winter School Tutorials: How to handle 3D spectroscopy data

S.F.Sánchez[1] B.García-Lorenzo[2] A.Pencontal-Rousset[3]

January 4, 2006

1 Introduction

Integral Field Spectroscopy (IFS) is the technique that allows to obtain simultaneus spectra of a (more or less) continuous area of the sky (CITE). The light coming from different positions in the sky is conducted and resampled in the spectrographs by several different techniques (lensarrays, fiber bundles, image slicers....). These different implementations have produced a set of instruments, which, sharing the basics of the technique, produce very different representations of the spectra in the detectors. These apparent diversity leaded to the creating of reduction techniques and/or packages for each individual instrument (e.g., P3d, Becker 2001). Together with the inherent complexity of this technique, this has reduced the use of IFS for decades to a hand-full of specialists, mainly attached to an expecific instrument.

IFS specialists have realized of this handicap (Walsh & Roth 2002), and started to produce standard techniques and tools valid for any integral field unit (IFU). In a recent efford, the Euro3D RTN (Walsh & Roth 2002) has created an standard data format (Kissler-Patig et al. 2004), a coding platform (Pécontal-Rousset et al. 2004) and a visualization tool (Sánchez 2004), useful for any of the existing IFUs. All these tools are freely distributed to the community. However, there is still a lack of generalistic packages for most of the analysis processes, like data reduction, line fitting, gas and stellar kinematics, and so. The astronomical community confronts with a jungle of different pipeles and tools expecific of each instrument. Following precent effords, we have developed a set of tools/procedures useful for most of the IFS data, that we will explain in these tutorials, as they were presented in the IAC Winter School. Many of these tools are still on the development process, and they are distributed here as they are, without any guarantee that they will work in the optimal way.

The distribution of this chapter is as follow. In Section 2 we describe the most used dataformats to store 3D spectroscopy data. In Section 3 we describe some of the existing tools before the effords of the Euro3D RTN network and our individual effords. In Section 4 we describe the installation requirements of the different packages used in these tutorials. Section 5 introduces `E3D`, the Euro3D visualization tool (Sánchez 2004). In Section 6 we introduce some notions of emission line fitting and gas kinematics. Crowded field spectroscopy and 3D modelling is described in Section 7. The empirical correction of the differential atmospheric refraction is explained in Section 8.

2 Data formats for the reduced data

IFS data can be stored in different data formats. All of them require to store the spectral information linked to the position in the sky sampled and the shape of the spaxel (Kissler-Patig et al. 2004). In many cases the final dataformat was created by the instrument developers, and it was only valid for the specific instrument. Due to that, after several discussions, the Euro3D network created a standard format, the Euro3D format (Kissler-Patig et al. 2004). This format is a FITs binary table, were the position of the spaxels (or spectral-sampling elements) is absolutely linked with the associated spectra. Although this format intend to be an standard, other formats are also used (almost one per instrument), although two are widely known: datacubes and row-stacked spectra (RSS).

Datacubes are only valid to store reduced data from instruments that sample the sky-plane in a regular-grided way (like lens-arrays) or for interpolated data. In both cases the data are stored in a 3D FITs cube, were the X and Y axis corresponds to the two spatial dimensions, and the Z axis corresponds to the dispersion axis.

Row-stacked spectra is a simple 2D FITs file where each spectrum is distributed row-to-row, independently of their position in the sky. Its requires an additional file (a FITs or ASCII table), were the location in the sky is stored. Although it is one of the most generals formats to store this kind of data, it is widely used in the case of IFUs with a discontinous sampling of the sky (e.g., INTEGRAL) or a not regular-grided lensarry (e.g., GMOS).

3 Tools before the Euro3D network

Waiting for content from Bego

We will perform some exercises

files: RSS =¿ NAME_RSS.fits (position table NAME_RSS.pt.txt) datacube =¿ NAME.cube.fits Euro3D format data =¿ NAME.e3d Position Table =¿ NAME.pt

2.1) IRAF: enter in IRAF (run `cl`) cl¿ imred cl¿ specred cl¿ splot NAME_RSS.fits

It is possible to zoom/unzoom, fit lines, plot different lines, with all the options of the gtools.

For more help: help/splot.txt help/gtools.txt

Similar capabilities in MIDAS (?)

2.2) DS9: Through IRAF, - It is possible to display datacubes slices to slices. cl¿ disp NAME.cube.fits[*,*,500]

- It is possible to display RSS spectra: cl¿ disp NAME_rss.fits

- It is possible to display spectra from a datacube. cl¿ disp NAME.cube.fits[10,*,*] cl¿ splot NAME.cube.fits[10,10,*]

2.3) DS9; without IRAF (datacubes).

- It is possible to display datacubes. Menu: File -¿ Open (open the datacube). A pop-up window opens, with a scroll-bar indicating the spectral index (i.e., wavelength). Moving the scrooll-bar, DS9 will show the slice that corresponds to the selected wavelength.

2.4) CubeFitsview (datacubes).

- It handles datacubes. It is possible to select a certain wavelength range an display the corresponding slice. It is possible to display the spectrum correponding to a single channel.

2.5) Other tools like MIDAS, GIPSY or starlink also handles datacubes.

4 Installation of the software

Along these tutorials we use different tools, most of them have been created by S.F.Sánchez, some of them under the Euro3D network. Among them we distribute here `E3D`, the Euro3D visualization tool (Sánchez 2004), `FIT3D`, a fitting tool for 3D spectrocopy (Sánchez et al. 2004, , Sánchez et al., in prep.), and `R3D`, a package for reducing fiber-based integral field spectroscopy (Sánchez 2006). They have a core written in `C`, using the I/O Lyon-C Library and its development enviroment (Pécontal-Rousset et al. 2004). PGPLOT (Pearson 1995) graphical library is used for plotting and, in some cases, `Tcl/Tk` libraries are used for developing graphical user interfaces (GUIs). In many cases `perl` is used as scripting language, or as a prototyping enviroment. In these cases the Perl Data Language (CITE) is used. As a general case it is required the following libraries:

- `g77`, the GNU Fortran77 compiler.
- `gcc`, the GNU C compiler[1]
- `Tcl/Tk`, version 8.3, at least. `http://www.tcl.tk`
- `PGPLOT`, `http://` , installed with the following drivers: XSERVER, XWINDOWS, POSTSCRIPT, GIF and XTK. The required `drivers.list` is distributed with the software.
- LCL I/O library, version 1.2, `http://www.aip.de/Euro3D`

and the following `perl` modules (`http://www.cpan.org`).

- Astro::FITS::CFITSIO.
- Math::Approx.

[1]The new `gcc` compiler do not include `g77`, so you must install it separately.

- Math::Derivative.
- Math::FFT.
- Math::Matrix.
- Math::Spline.
- Math::Stat.
- PDL.
- PGPLOT.
- Statistics::OLS.

We recommend to install the source code of all these packages, and due to that we have included them in the CD (ASK FOR PERMISSIONS??). In some cases pre-compiled Linux binaries are also included, what it will make installation easier for that users with standard Linux distributions.

4.1 Installing the LCL library

The Lyon-C library (Pécontal-Rousset et al. 2004) has been created by A.Pencontal in the Lyon Observatory. It is a `c`-coded I/O library that handles fitsfiles in many formats (1D and 2D images, datacubes, binary tables and the Euro3D format). It includes a simple and useful command-line argument handleing system. It is distributed through the Euro3D webpage (`http://www.aip.de/Euro3D/`). A copy of the version 1.2 it is included in the CD (`E3D_io_LCL-1.2.tar.gz`). To install it follows the following instructions:

1. Create a directory in your computer, that we will name hereafter `DIR`. Copy the compressed tar file to this directory `DIR`.
2. Decompress and untar the file.
3. Go to the created directory, e.g., `cd E3D_io_LCL-1.2`
4. Run the configure program: `./configure`
5. Run the make program: `make`, ignore any warning and error.
6. Define the enviromental variables: `IFU_PATH` and `IFU_DEFAULT_FMT`

   E.g., for a tcsh shell

   `setenv IFU_PATH DIR;`

   `setenv IFU_DEFAULT_FMT Euro3D;`

   Note: The `IFU_PATH` may point to `DIR` not to `DIR/E3D_io_LCL-1.2`

7. go to the directory:

   `cd checklib/exec`

   and run the test program:

   `./check_E3D_io`

   If it works without any error advice the library it is installed.

4.2 Installing PGPLOT

PGPLOT is a fortran graphic library, created by T.Pearson (Pearson 1995). It has a `C` and a `perl` wrapper, that allow to use it for coding in this languages. It is distributed in the webpage `URL`. The instructions of how to install it are described in the `INSTALL.txt` file included in compressed tarfile.

4.3 Installing E3D

E3D (Sánchez 2004) is a visualization tool for integral field spectrocopy created by S.F.Sánchez for the Euro3D RTN. We distribute here the last version, v1.3b, in two formats: its source code (`e3d-1.3b.tar.gz`, and a binary distribution, for Linux (`E3D_v1.3b_linux.tgz`) and MacOs (`E3D_MacOs.tgz`). Binary distributions have included most of the required libraries, although some of them have to be installed.

4.3.1 Installing the source code of E3D

Follow the next instructions:

1. Copy the tar file into the Euro3D directory (given by the environment variable `$IFU_PATH`).
2. Using the command `tar xvfz e3d-1.3b.tar.gz` , extract the archive file `e3d-1.3b.tar.gz` under `$IFU_PATH`. It will create the directory `e3d-1.3b` (or a corresponding one, depending of the distribution).
3. Change to the created directory (`cd e3d-1.3b`), and execute the configure and make scripts.
4. `./configure`
5. `make`
6. Goes to the scripts directory (`cd scripts`) and execute the install script (`./install.pl`). This script will edit the `tk_e3d.tcl` script to be compatible with your directory tree.
7. Add the directory `user/bin` under the E3D tree to your PATH.
8. Test the E3D executing the script `tk_e3d.tcl` , and loading the file `test.e3d` under the `scripts` directory.

4.3.2 Installing the binary distributions of E3D

Follow the next instructions:

1. Copy the tar file into the Euro3D directory (given by the environment variable `$IFU_PATH`).
2. Uncompress the tarfile.
3. Goes to the new created directory.
4. Execute the `./install_static.pl`, and follow the instructions.
5. Test the E3D executing the script `tk_e3d.tcl` , and loading the file `test.e3d` under the `scripts` directory.

4.4 Installing FIT3D

FIT3D is a fitting tool developed by S.F.Sánchez (García-Lorenzo et al. 2005; Sánchez et al. 2004) to perform emission line and continuum spectrum fits, in order to analyze gaseous emission and stellar kinematics. It consists of a core coded in `C`, for the low-level fitting routines, and a wrapper written in `perl`, to help on the creating of the configuration files required to perform the fits. It has a similar phylosophy to DIPSO, sharing with it the idea of configuration files to creating the fitting model. It allows to fit simultaneously several emission lines and continuum backgrounds, with different kinematics, setting limits to the different parameters of the emission lines, linking parameters between lines and creating kinematics systems (several lines sharing the kinematics properties), and deblend different emission line components. It uses datacubes and row-stacked spectra as input data, and produce FITs files as output. In some cases, it requires E3D for the creation of the interpolated maps.

Based on the LCL development environment it has the same requeriments of E3D. To install it is needed to follow the same installation instructions. We distribute here two versions, its source code (`fitting_tool-1.0.tar.gz`), and a Linux binary distribution (`FIT3D_v1.0_linux.tgz`). Appart from the `C` core, `FIT3D` uses `perl` routines that require the modules listed above (Section 4).

4.5 Installing R3D

`R3D` is a package for reducing fiber-based IFS data obtained with any IFU of this kind (Sánchez & Cardiel, in prep.). Although the effords of the Euro3D RTN have produce valuable advances in the direction of creating standard tools for analizing IFS data, there is a lack of generalistic reduction packages, which were out of the scope of the network. The astronomical community confronts with a jungle of different pipeles expecific of each instrument, which procedures and parameters differ strongly, and which outputs are difficult to compare. Following precent effords (e.g. P3d Becker 2001), we have developed `R3D`, a reduction package able to reduce any fiber-feed IFUs data.

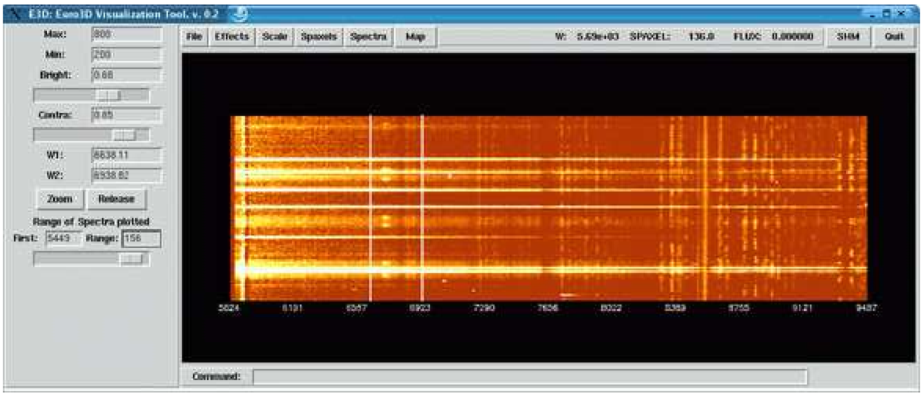

Figure 1: Stacked Spectra Inspector. This is the main GUI window, from which all the others can be created. It displays all the spectra contained in the 3D file stacked row-by-row.

We first coded the algorithms in Perl, using the Perl data language (PDL, Glazebrook et al 2001), in order to speed-up the algorithm testing phase. Once we tested the algorithms with simulated and real data, we recoded the most time critical ones in C, to increase their performance. The Perl version of `R3D` is fast enough to produce valuable science frames in a reasonable time, although for instruments with many fibers (like VIMOS), or for some slow processes, like cross-talk correction algorithms, we strongly recomend the C-version. A major advantage of using Perl and C is that they are freely distributed, platform independent, languages, and therefore `R3D` can be installed under almost any architecture without a major efford and no cost. The instructions of how to download and install `R3D` can be found in its webpage [2], where we will keep updated the different versions.

4.5.1 Installing the `perl` version of R3D

`R3D` uses `perl` routines that require the modules listed above (Section 4). We distribute here a tar file including both the routines and some documentation (`R3D.tgz`). To install them decompress this tar-file and follow the instructions included in the `INSTALL.txt` file.

4.5.2 Installing the `C` version of R3D

Based on the LCL development environment it has the same requeriments of E3D. To install it is needed to follow the same installation instructions. We distribute here two versions, its source code (`r3d-0.1.tar.gz`), and a Linux binary distribution (`R3D_c_linux-0.1.tgz`). Appart from the `C` core, `R3D` uses `perl` routines that require the modules listed above (Section 4).

[2] http://www.caha.es/sanchez/r3d/

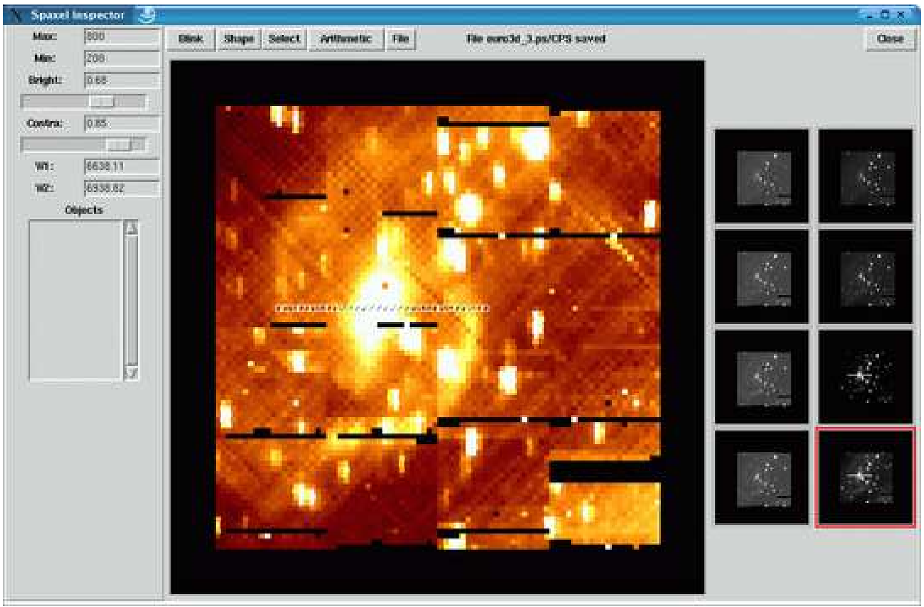

Figure 2: Spaxels Inspector. This is the GUI for plotting monochromatic or polychromatic maps. It is possible to select different spaxels, to be displayed subsequently on the Spectral Inspector.

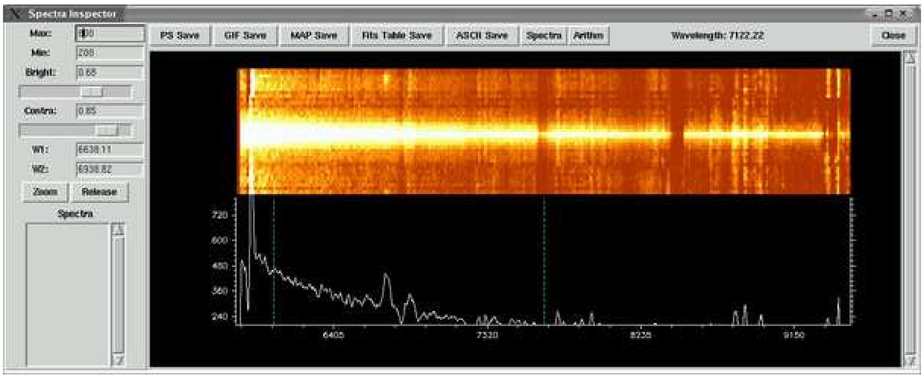

Figure 3: Spectral Inspector. This is the GUI for plotting spectra.

5 E3D: The Euro3D visualization tool

`E3D` is a visualization tool for IFS data entirely independent of the origin of data, i.e. 3D instrument. It was created on behalf of the Euro3D network (Sánchez 2004), using the Euro3D data format as standard format, but able to work with another formats (like datacubes and RSS data).

`E3D` comprises a C-coded core, with three main elements:

- The `Euro3D.o` library of low-level functions, including the routines calling the Euro3D I/O library, the SHM routines, basic functions needed for plotting and analizing the data. This library can be invoked to create another program, which needs to read/write Euro3D data, or uses the SHM routines.
- The `tk_e3d Tcl/Tk` interpreter. This is a standalone programme that creates its own `Tcl/Tk` interpreter, adding Euro3D routines to the standard `Tcl` ones. These routines invoke the `C`-functions included in `Euro3D.o` from `Tcl`, and they can be used for different proposes: e.g., load/save Euro3D format files, plot single or coadded spectra, plot monochromatic/polychromatic maps, interpolate these, save maps in FITS format.
- A number of stand-alone `C`-coded tools that help to handle the Euro3D format. Perhaps the most interesting routine is `any2Euro3D`, transforming single Group IFS data from any instrument into the Euro3D Format.

Together with the `C`-core, we have coded a `Tcl/Tk` Graphical User Interface (GUI), that uses the Euro3D-Tcl routines (`tk_e3d.tcl`). The GUI comprises three main windows:

- The Main window or Stacked Spectra Inspector. It comprises the main Menu with different options to handle Input/Output and different representations of the data. The data are automatically plotted in its canvas on the Stacked Spectra representation once loaded. A command-line prompt has been included in this window to directly call Euro3D-Tcl routines. Figure 1 shows an image of this window.
- The Spaxels Inspector. It comprises a main canvas for plotting monochromatic/polychromatic datacube slices, which have been selected previously from the main window or from the Spectral Inspector (see below). It includes 8 minor canvas for buffering the last created maps. It also includes a menu with different options to handle spaxels/maps, including the spaxel selection, object creation and the selection of different methods to show the data. Figure 2 shows a screen-shot of this window.
- The Spectral Inspector. It comprises a main canvas for plotting the spectra corresponding to spaxels which were selected from any of the two previous windows. It also includes a menu with different options to handle the spectra. Figure 3 shows an image of this window.

5.1 Working E3D: the Graphical User Interface

We describe here a few examples describing the capabilities of `E3D`:

- **Loading data in different formats:** As we quoted above, `E3D` can handle data in the Euro3D format, its standard format, and in the RSS and datacube formats. All of them are transformed to the same structures in memory, so once loaded they are handled in a similar way. In order to follow the exercises change to the directory `tut1` of the CD, and open the `E3D` GUI running the `tk_e3d.tcl` command.
- **Data in the Euro3D format**

  Perform the following instructions:

  - In the top Menu of the main window select `Load` on the `File` menu. A pop-window opens.
  - Change from `fits` to `e3d` extension.
  - Select the file `NAME.e3d`.
- **Data in the RSS format**

  Perform the following instructions:

  - In the top Menu of the main window select `Import RSS` on the `File` menu. A pop-window opens.
  - Select the `NAME_RSS.fits` RSS file.
  - Select the `NAME_RSS.pt` position table.
- **Data cubes**

  Perform the following instructions:

  - In the top Menu of the main window select `Import cube` on the `File` menu. A pop-window opens.
  - Select the file `NAME.cube.fits`

Exercise 1: Try to load your own data. *First identify in which format are your data (read the reduction tool notes). If you need to build a position table, remember that the position tables in* `E3D` *are* `ASCII` *files with 4 columns, where the 1st line indicates the geometry of the dataset, the remaining columns indicate the position table, in the form:*

```
TYPE     SIZE1  SIZE2  TYPE_ID
INDEXi   Xi     Yi     TYPE_IDi
...
```

Where TYPE is the shape of the spaxels, and it can be: `R` *for a rectangle shape,* `S` *for a square,* `C` *for a circle, and* `H` *for a hexagon. e.g.,*

```
H    0.2  0.2  1
1    0.0  0.0  1
2    0.5  0.0  1
3    0.5  0.5  1
...
```

5.2 Quick Look to the three windows of the E3D GUI

As we quoted before, the GUI of E3D has three windows:

5.2.1 The main window

,

It is the window that it is first opened with running E3D, where the data are displayed in the RSS format, as they are ordered in the loaded file (i.e., no preferential spatial direction, a priori). In this window it is possible to scan through the spectra.

Perform the following instructions:

- Change the directory to `tut3` and open E3D.
- In the top Menu of the main window select `Import cube` on the `File` menu. A pop-window opens.
- Select the file `NAME.cube.fits`.

The main window display 256 spectra of the total NN spectra, in a RSS 2D image, where each row shows a single spectrum as they are ordered in the data file. E3D select the range of plotted values by determining the median, and the standard deviation. Sometimes it is not a good estimation of the range to values to plot, and it is needed to change by hand. To change the maximun and minimum values just type on the `Max`, and `Min` entries in the left-side menu. *NOTE: A range of [-1,200] would be acceptable to visualize the data.*

To see better the data sometimes it is needed to readjust the `Bright` and `Contrast`, editing by hand in the corresponding entries, or just moving the corresponding scroll-bars.

Exercise 2*: Change the maximum, minimun, bright and contrast until you have a proper visualization of the data.*

To scan through the spectra you can move the scroll-bar in the bottom part of the left-side menu, below the label `Range of Spectra plotted`. It is also possible to change the number of plotted spectra editing by hand in the two entries labelled `First` (the 1st spectrum plotted) and `Range` (the number of spectra plotted).

Exercise 3*, play with the scanning capabilities.*

It is possible to `zoom` over a certain spectral region, by selecting a region clicking twice over the main window with the left-mouse button. The selected range in the `W1` and `W2` entries in the left-side menu. You can also edit directly on these entries, if you want a more precise selection of the wavelength range. In order to return to the default range click on the `Release` button.

E3D always uses the same method for selecting. The left mouse button is used for selecting on a window and the right mouse button for sending this selection to the corresponding next window. We will experiment with this latter.

Exercise 4 play with the zooming capabilities until you get used to them.

The combination of zooming and range selection is useful for looking for specific wavelength and spatial ranges, like in the case of emission line objects.

Exercise 5 *try to locate some of the emission lines in this object.*

5.2.2 The spaxels inspector

The spaxels inspector is the window of the GUI that allows to visualize the spatial distribution of the intensities for each spaxel at a certain selected wavelength range. It allows you to select individual or groups of spaxels/channels to visualize the spectrum/spectra corresponding to them.

On the `main window` menu, click on the `Open` options of the `Spaxels` submenu. A new window that popups is the spaxels inspector. It consists of a top menubar, a left-side menu, with similar entries than the `main window` one, and specific entries for this inspector. There a main display window, to display slices, and a 8 small buffer windows, where it is stored the last 8 displayed slices. These new entries in the menu are just informative. `Row.No` indicates the row number in the RSS representation of the spaxel over which we place the cursor. `Flux`, `X` and `Y` shows the corresponding flux, and coordinates of this spaxel (where the flux is the average of the intensities for the selected wavelength range).

As a general remark, `E3D` displays average values when a selection of several spectral channels or spaxels is done. The reason is that for visualizing it would require to change the scale continuously if we co-add the data. This feature has the advantage that for subtracting wavelength ranges it is not needed to know the width of subtracted range. It has de disadvantage that it is needed to take note of that width when storing the data for future calculations.

`N.SPAXELS` is the number of currently selected spaxels and `MEAN` and `SIGMA` shows the mean flux value and its corresponding standard deviation through these selected channels. We will explain below how to select spaxels. We described above how to select a certain wavelength range in the `main window`, just return to the main window by clicking in it and select a certain wavelength range, once selected click on the right-mouse button. The reconstructed slice obtained by averaging the intensity through all the spectral channels of the selected wavelength range is displayed in the spaxels inspector.

Exercise 6: *Play with the maximum and minimum values and with the bright and contrast to visualize properly the data.*

Exercise 7: *Select different wavelength ranges, and see the differences at different wavelength ranges.*

It is also possible to select a narrow-band image and create a movie effect by clicking and keep clicked the central-mouse button on the `main window` and scan through the wavelength range. Once dropped the button, the last selected range is stored in the spaxels inspector.

Exercise 8: *Play with the movie effect to see the displacing of the data from the blue to the red due to the differential atmospheric refraction (DAR), and the areas dominated by the emission lines.*

It is possible to return to a previous displayed slice (from the last 8!), by clicking with the right-mouse button on the corresponding small display windows, at the bottom of the spaxels inspector.

To select a spaxel or a group of them, just click the left-mouse button and keep it clicked while moving near the desired spaxel to be selected. A hashed-polygon will appear over the selected spaxels. Note that the `N.SPAXELS`, `MEAN` and `SIGMA` entries change when selecting a certain number of spaxels. More selecting option will be described below. To clear a certain selection, just click on the `Clear` option of the `Select` submenu in the Spaxels Inspector top menubar.

Exercise 9: *Play with the spaxels selection capabilities.*

5.2.3 The spectra inspector

The spectra inspector is used to visualize arbitrarily selected spectra (instead of spectra ordered in the way they are stored in the data file). On the `main window` menu, click on the `Open` option of the `Spectra` submenu. The new window that popups is the spectra inspector. It consists of a top menubar, a main display window and a left-side menu. Most of the entries in the left-side menu are similar to those ones of the main window, expect to the `spectra` box-list, that we will explain below.

The most straightful way to select a group of spectra is just coming back to the `spaxels inspector` and make a selection of spaxels, as described above. Once you have selected one or several spaxels just click the right button on the spaxels inspector and the spectra corresponding to the those spaxels will be displayed in the `spectra inspector`. By default the `spectra inspector` shows a 2D image of the RSS representation of the selected spectra on the top region, ordered by the sequence of selection, and a plot of the average of these spectra on the bottom region of the display.

Exercise 10: *Change the maximun and minumun, and the bright and the contrast to visualize properly the selected spectra.*

Like in the main window it is possible to make zooms over certain spectral regions and/or select them, using the same procedures described above. It is also possible to display a the average slice corresponding to a certain selected region by clicking the right-mouse button.

5.3 Detailed description of the options in the main window

The top menu of the `main window` contains the options that affects to all the dataset (with the expection of the sky subtraction that it is on the `spectra inspector`) or the general setup of the visualization of the data.

5.3.1 `File` menu.

Comprises all the input/output options for all the dataset. The `Euro3D data format` (Kissler-Patig et al. 2004), is the native format of this visualization tool, and so the `load/save` options refer to this data format. There are entries for `import/export` dataset, in the RSS and datacube formats.

When exporting data to a datacube `E3D` performs always an interpolation using the interpolation entries configured in the main window (configuration). If you import a datacube and export it again, it does not keep the previous format of the datacube, but creates a new datacube with the interpolated data. This option may be implemented in the future. We will show how to overpass this potential problem, if needed.

It is also possible to import/export and slit spectra, without taking into account the spatial information, or store cut of the current selected spectral region.

5.3.2 `Spaxels` and `Spectra` menus

They allow you to open the `Spaxels inspector` and the `Spectra inspector`. They also have two alternative tags that allows you to change between the selection based on wavelength ranges or a selection based on the distribution of the spectra in the RSS.

Exercise 11: *Open both the spaxels inspector and the spectra inspector. Select a wavelength range and click the right-mouse button. See the effect in the spaxels inspector. Now change the selection from spaxels to spectra (by clicking on the tag at the* `Spectra` *menu). Select a set of spectra (two left-mouse clicks and one right-mouse click), and see the effect in the spectra inspector. Change to the default.*

5.3.3 The `Configuration` menu

It allows you to change the setup of the visualization tool.

- `ColorMap`, to change the current color table.

  Exercise 12: *Experiment with the different color tables.*

- `Scale`, to change the method the data are scaled to the corresponding colors.

- `Spectral Representation`, to change the way the spectra are displayed in the `spectra inspector`.

  Exercise 13: *Experiment with the different representations. Select a group of spaxels and display them in the spectra inspector. Change between the different representations until you get used to them.*

- `Interpolation`. It opens a pop-up menu were it is possible to change the interpolation parameters. E3D has implemented 5 different interpolation routines, which has been taken with permission from PLPLOT (under the requirement of not copyrighting the code). It is possible to select between the different interpolation methods by pull done the `FUNCTION` option in the interpolation window. The other two options are the `Grid Option` and the output `pixel size`. Some of the routines requires an option parameter which meaning changes function to function. (read the E3D documentation and specific mathematical books for their exact mathematical meaning).

Exercise 14: *Import the* `NAME_RSS.fits` *file in E3D, select a certain wavelength range and display the results in the* `spaxels inspector`*. Select the* `map` *option in the* `Shape` *menu, to display an interpolated version of the slice. Change the interpolation parameters and functions to see the effect in the interpolated output. To our knowledge, the most optimal interpolation is the Natural Neighbour. However, it is slow. It requires a very small number for the* `Grid Option`*, e.g., 1e-9, and as smaller it is, as better and slower.*

- `Spaxel Selection`, it popups a window with a menu that allows you to change the method for selecting spaxels, from `single` to `aperture`. The default method is `single`, and allows to select spaxels in the `spaxels inspector` one-by-one. The `aperture` method allows you to select all the spaxels within a certain radius (selected in the popup window).

  There is another selection method, the pseudo-slit method. By clicking the `s` key on the `spaxels inspector` it is possible to switch between the `single` (or aperture) selection and the `pseudo-slit` selection. When this mode is selected you just need to click at two points on the `spaxels inspector` display (with the left-mouse button) and the spaxels along the straight-line between these two points will be selected. Like in the previous method you can see the spectra along this line in the `spectra inspector` by click the right-mouse button. The `pseudo-slit` selection is useful to compare with slit-spectrum

  Exercise 15: Experiment with the different spaxel selection methods. Select spectra ono-by-one, change to aperture selection, and to the pseudo-slit selection. See the differences in the `spectra inspector` and experiment until you get used to the different methods.

5.3.4 DAR Tools

`E3D` includes some tools for correcting for the differential atmospheric refraction. These tools can be divided in two groups, (i) tools for determining the displacement of the image along the wavelength and (ii) tools to correct for this displacement. The 1st tools include the theoretical DAR, based in the (Filippenko 1982) formulae. It relies on HEADER keywords not included in the standard Euro3D format (read the User Guide), and in a proper orientation of the position table in the sky (east left, north up). It is indicated as `Th.Det. DAR`. Two other empirical determinations of the DAR has been included. These tools relies on the presence of a clear peak in the spatial distribution at any wavelength (e.g., a point-like source). They are indicated as `PM.Det. DAR`, which determines the centroid of the image based on the principal momentum, and `peak.Det. DAR`, which determines the peak of the emission by a simple maximum search.

Once determined the DAR to apply, it is possible to correct for by selecting the `Correct DAR`. It is important to understand what the program is doing when performing an a DAR correction: First, at each wavelength, the data are interpolated using the interpolation scheme selected in the configuration, then, the intensity corresponding to the original spaxels structure at the new position determined by the offset determined by the previous calculations.

Any DAR correction implies an interpolation in the spatial domain. To resample the data in the original shape is just a mathematical trick, since the recovered data comes through an interpolation. We therefore do not recommend to use the DAR tools included in `E3D` for science applications, and just for simple quick check of the data. We will explain in another tutorial how to correct for DAR, based on datacubes (i.e., interpolated version of the data in the general case).

Exercise 16: Experiment with the DAR correction. *(dir: tut3) Open E3D and import the cube* `star_no_dar.cube.fits`*. open the spaxels inspector and with the movie effect check the DAR effect. Click on the peak correction at the main window top-menu (DAR tools), and correct by this correction. Store the data another file (e.g., star_dar.cube.e3d). As you can see the correction is not optimal.*

5.3.5 The command-line entry

As explained before `E3D` is a command-line environment, based on a tcl/tk interpreter which adds new commands, the E3D commands (read the E3D user guide). The GUI is based on that set of commands. Therefore, it is possible to invoque any of this commands by writing directly in the `command` line entry, at the bottom of the main window.

It is also possible to run a `tcl/tk` script, by writing `source scriptfile` at the command line entry. Everytime you execute `tk_e3d.tcl`, the program creates a logfile of the different actions you have performed. This log is stored in the file `e3d.log`. You can always run again the same sequence of commands coping this output file into a different file (not to rewrite), and execute it via the command-line entry.

Exercise 17: Experiment with the command line entry. *Open E3D, and import the cube* `VIMOS.cube.fits`*. Change the maximun and minumun to see the data. type on the command line:*

```
recolor_image heat 0
recolor_image iraf 0
recolor_image rainbow 0
```

See the effect in the display window.

Exercise 18: Experiment with the scripts. *(dir:tut3). Open E3D and write in the command line:* `source script.tcl` *see the effect. Open the file* `script.tcl` *with an editor (e.g., emacs), to see the sequence of command that you have executed.*

5.4 Detailed description of the options in the spaxels inspector

The top menu of the `spaxels inspector` contains the options that affects to the spatial representation and the spatial selection of the data.

5.4.1 `File` menu

It handles the input/output of data and plots related with the spatial distribution. The data files can be, slices (collection of coordinates with their corresponding fluxes), or maps (2D images). The slices can be stored in FITs or ASCII tables (four column tables, with and id, x, y and the flux), while the maps can be stored in FITs images. The maps are interpolated versions of the currently displayed datasets, created using the configuration for interpolating describe above. A particular case is the `save object` option, that we will explain below.

The plots are just Postscripts or GIF hardcopies of the last displayed image in the spaxels inspector.

5.4.2 Shape menu

.

It allows to change the method of how the data are represented in the spaxels inspector, including different options combining the original spaxels shape, an the interpolated version of the data (i.e, the map).

Exercise 19: *Import the file INTEGRAL_SB2.rss.fits, using the position table INTEGRAL_SB2_pt_219.txt (dir:tut4). Open the spaxels inspector and display different wavelength ranges. Select the different options in the* `Shape` *menu to test between the different representations. Store the resulting plots/slices using the different output options in the* `File` *menu. In the case of 2D fits images, visualize them using an external tool, like DS9. In the case of the ASCII tables, open them with an editor, or just with* `more` *to get the structure of these datasets. For the plots, display them with ghostview or xv.*

5.4.3 Select menu

.

As we already described above, it allows you to clear the selected spaxels, to start a new selection. However, it contains other options different than `Clear`:

- `plot spectra`. It has the same effect than clicking the right-mouse button.
- `plot profile`. It allows to plot the intensity profile of the displayed slice along the selected spaxels. It is particularly useful when combined with the `pseudo-slit` selection.

Exercise 20: *click on the* `s` *key of the keyboard on top of the spaxels inspector display. Look at the information label on-top of it. It should say* `Pseudo-Slit Selection`*. Select a cut that goes through the center of image, and select the* `plot profile` *option. The intensity profile along the pseudo slit will appear in the spectra inspector, fitted with a gaussian.*

- `create object`. We define an `object` a collection of selected spaxels. You can create an object by clicking on this option on the `select` menu. Once clicked, an entry will appear on the `Objects` list box at the left-side menu. Only 9 different object can be selected. Once overpassed this number the objects will be overwritten. `objects` are the way to keep your previous selections. To recover any `object` just click on it on the `Objects` list box. The corresponding spectra will be shown in the `spectra inspector`.

  Exercise 21: Play with the object creation. *(dir:tut4) On the spaxels inspector select the central spaxels and store the selection as an object. Clear the selection and repeat the process with the outer part of the central square, and with the external circle of spaxels. Open the spectral inspector and select alternatively each of the object. See the different selected spectra.*

  It is possible to store just the selected group of spaxels or object in a separate Euro3D file, by using the `save object` option (that we did not explained before). Just click on one of your stored objects (or select a group of spaxels), and select this option. If you want to see how it looks your stored data, just load it in the main window.

- `reverse object`. This option allows to change the spaxels selection stored in an object by just the complementary group of spaxels (i.e., that ones that were not selected). Click this option and then just on one of your created objects and see the effect.

  Exercise 22: Separete sky and objects fibers. *(dir:tut4) Import the INTEGRAL_SB2.rss.fits file, and display a certain wavelength range slice in the spaxels inspector. Select one-by-one all the spaxels in the outer ring, those which correspond to the sky fibers. Once selected them store the selection as an object twice. Reverse the selection of one of this object, so you have selected the central square. Once you have selected each one, just store the results with the* `save object` *option in two different files: obj.e3d and sky.e3d. Load this files in E3D to see the result.*

  Exercise 23: Masking regions by creating objects. *(dir: tut3) Import the* `VIMOS.cube.fits` *file in E3D. Open the spaxels inspector. Selected a certain wavelength range, and display the corresponding slice on the spaxels inspector. There are four clear* `bad` *regions corresponding to dead-fibers. Selected them and store the selection as an object. Reverse the object selection and store the object as* `good_data.e3d`*. Load the new created Euro3D file, and display the result in the spaxels inspector.*

5.5 Tools menu

.

It includes to basic tools that operates over the slices:

- `Blink`. Allows to blink between different slices previously stored in the 8-length small display buffer.

  Exercise 24: Play with the blinking. *(dir: tut4) Import the INTEGRAL_SB2.rss.fits file, and display different slices at different wavelength ranges (hint, select continuum dominated regions and emission line regions). Once you have selected different regions, click on the* `blink` *button, then select by clicking with the left-mouse button on the small displays the different slices to blink. The click again on the* `blink` *button to start the blinking. The blinking can be stopped by clicking again on the* `blink` *button.*

- `Arithmetic`. It allows you to perform simple arithmetic operations between previously selected slices. It popups an small window where you can select the two slices over which to perform the operation, by their buffer numbering, and the buffer where to store the result. You can select the kind of operation in the `Arithm` menu and perform it.

  Exercise 25: Create an emission line map free of continuum in a simple way. *(dir: tut4) Load the obj.fits (created before), and display two slices selecting the wavelength region of [OIII] and a continuum region nearby. Substract the continuum to the [OIII] slice. This will be raw estimation of the continuum free emission-line map.*

5.5.1 Keyboard options

Appart from the different options included in the top menubar, there are a few keywords that can be used for different operations:

- `s`, alternative changing from the pseudo-slit to the spaxel-to-spaxel (or aperture), selection.

- `z`, `u` allow to zoom over certain regions of the image. You need to click once to `z` with the mouse over the starting corner of your zooming region, and once more with the mouse over the ending corner. A zoom over the selecting region will appear in spaxels inspector. To unzoom, click `u` over the spaxels inspector, and the full-view image will be shown.

Exercise 26: Playing with the zooming capabilities. *Import the* `VIMOS.cube.fits` *file in E3D. (dir: tut3) Open the spaxels inpector. Selected a certain wavelength range, and display the corresponding slice on the spaxels inspector. Zoom in and zoom out over different regions until you get used to the method.*

5.6 Detailed description of the options in the Spectra Inspector

The top menu of the `spectra inspector` contains the options that affects to the spectral representation and the spectral selection of the data.

5.6.1 File

.

It handles the output of spectra data files and hardcopies of the plots shown in the spectra inspector. It is possible to save hardcopies in Postcript or GIF, 2D RSS images of the plotted spectra, and FITs or ASCII tables of the average spectrum.

In `E3D` a spectrum table consists of three columns: An Id (indicating the spectral index), the wavelength, and the Flux. Remember that a single spectrum is the mean value of the plotted spectra, an not the sum.

Exercise 27: Testing the saving capabilities of the spectra inspector. *(dir: tut4) Import the INTEGRAL_SB2.rss.fits file, and display a certain wavelength range slice in the spaxels inspector. Select different groups of spaxels and show the corresponding spectra in the* `spectra inspector`*. Save the selected spectra as a postscript, a gif, a 2D RSS image, a fits table and an ASCII tables. Visualize the output using external tools, like ghosview, xview, iraf and DS9. Repeat the opertion until you get used to the format of the output files.*

5.6.2 Spectra

.

This menu handles the storing in memory of different average spectrum. In this regards is similar to the `object` creating in the `spaxels inspector`. This spectrum can be replotted, one-by-one or several of them, or perform basic arithmetic operations, like the ones seen in the `spaxels inspector` over the slices.

By clicking on `Save Spectrum`, a new spectrum is stored, up to a total number of 9, and a new entry appears in the listbox labelled `Spectra`. In order to display again any of the stored spectra, just click on the corresponding entry in the listbox.

It is possible to show several spectra in the same screen by selecting the tag labeled `single/multiple plot` on the pull-down menu, and select the desired spectra to plot from the `spectra` listbox. Each spectrum will be plotted using a different color, in order to distinguish them. Click on the `Clear Multiple selection` option to clean the screen and create new selection.

Exercise 28: Testing the multiple spectra storing capabilities of the spectra inspector. *(dir: tut4) Import the INTEGRAL_SB2.rss.fits file, and display a certain wavelength range slice in the spaxels inspector. Select different groups of spaxels and show the corresponding spectra in the* `spectra inspector`*. Store each of these spectra in a separate slot. Select individually* *each the spectra to see them in the* `spectra inspector` *display. Change from single to multiple selection and repeat the selection to see the differences. Store them as postscripts and/or gif files, and visualize them using external tools.*

Another options in this menu are:

- `Arithmetic` allows to perform basic arithmetic operations over the memory stored spectra. A window menu popups (similar to the slice arithmetic), requesting the two spectra from the listbox overwhich perform the operation, and the arithmetic operation to be performed. The result will be stored in the next free position of the slot [3].

  Exercise 29: Get used to the spectra arithmetic. *(dir: tut4) Import the INTEGRAL_SB2.rss.fits file, and display a certain wavelength range slice in the spaxels inspector. Select different groups of spaxels and show the corresponding spectra in the* `spectra inspector`*. Store each of these spectra in a separate slot. Select individually each the spectra to see them in the* `spectra inspector` *display. Perform different arithmetic operations with the selected spectra and visualized them using the single or multiple display options to compare the results.*

- `Sky Subtraction` allows to perform a subtraction of an stored spectrum over all the data set. This option is normally used to perform an sky subtraction. To use it, just click on it, and them select the spectrum to be subtracted. The program will perform the subtraction to each spectrum in the dataset.

  Exercise 30: Sky subtraction. *(dir: tut4) Import the INTEGRAL_SB2.rss.fits file, and display a certain wavelength range slice in the spaxels inspector. Select the spaxels in the outer ring (the sky-fibers ones), and visualize the resulting* `sky` *spectrum in the* `spectra inspector`*. Store this spectrum in a memory slot, and perform a sky subtraction using this spectrum as* `sky`*. Once performed this subtraction it is possible to store the complete data set (goes to the main window and store it as a RSS or an Euro3D dataset). It is also possible to remove the* `sky` *spaxels, storing the sky spaxels selection as an object and performing a* `reverse object` *operation, and storing the object (e.g., as* `sobj.fits` *as RSS or* `sobj.e3d`*), like described in a previous exercise.*

5.6.3 Fitting

.

`E3D` includes some simple fitting routines. We described before the fitting routine to determine the FWHM of a profile cut on a slice (`spaxels inspector`). On this menu there are two simple fitting options:

- `Fit Single Emission line`. It fits the plotted spectrum with a single emission line (a gaussian function), plus a continuum. The four free pa-

[3] A bug is known when trying to operate spectra in the 1st position of the slot.

rameters are the intensity of the gaussian, the FWHM, the central wavelength and a constant for the continuum. The resulting model from the fitting will be plotted in red over the data (in white). The output is printed in the xterm.

Exercise 31: Fit an emission line. *(dir: tut4) Import the sobj.fits or sobj.e3d file, and display a certain wavelength range centred in the [OIII] emission line, without containing any other emission line. Display this slice in the spaxels inspector. Select an slice near the central part, and display the corresponding spectrum. Fit this spectrum using the* `Fit Single Emission line`.

- `Full Automatic Kinematics Analysis`. It performs a kinematics analysis based on the over a single emission line (described above). Being an automatic procedure the results is not completely reliable, but just a quick&dirty estimation of the kinematics on the object. It fits each spectra in the dataset to a single Gaussian function plus a constant to describe the continuum. After performing the fit, it creates three slices or maps of the output of the fit, i.e., a map of the line intensity, a map of the line FWHM, a map of the central wavelength and a map of the continuum emission. The fit will be performed over the previously selected wavelength range.

  Exercise 32: Full Kinematics. *(dir: tut4) Import the sobj.fits or sobj.e3d file, and display a certain wavelength range centred in the [OIII] emission line, without containing any other emission line. Display this slice in the spaxels inspector. Select an slice near the central part, and display the corresponding spectrum. Fit this spectrum using the* `Fit Single Emission line`*. Once performed this 1st fit, click on the* `Full Automatic Kinematics Analysis`*, and wait until it performs the fits over all the spectra in the dataset. Once finished, four new maps will be created in the spaxels inspector, one per free parameter in the model. Check visually the different maps.*

6 Fitting and Modelling

Like in many other topics in astrophysics (and in physics in general), in 3D spectroscopy it is needed, in many cases, to model the spectra and/or the 2D distribution of the flux (or even the entire dataset) to extract proper information from the data. We abord here the peculiarities of fitting and modelling 3D data in a few simple cases: Emission line fitting, emission line kinematics, and 3D modelling.

6.1 Emission line Fitting

We give here a brief description on how to fit and deblend emission lines (like ionized gas emission lines). For this purpose, we will use our own fitting routines which use as inputs RSS and datacubes (`FIT3D Sánchez et al. 2004`), which is still under development and distributed without any guarantee. These routines consist of a core coded in `C` and a set of wraps coded in `perl`.

6.1.1 `fit_spec_back`, the `C` coded core

`USE: fit_spectra SPECTRUM.TXT CONFIG_FILE [BACKGROUND_LIST NBACK]`

This program fits the data to the proposed model by a minimization of the reduced χ^2, using a modified Levenberg-Marquardt algorithm. It uses as inputs an spectrum in the ASCII format (an ASCII table, as defined by `E3D`, i.e., a three column ASCII file, with: ID, WAVELENGTH and FLUX). and a configuration file (in ASCII too), where the desired model to fit is described.

This configuration consists of a first line with a header of four emelents: two integers, the first one left always to be 0 (to identify the file), and second indicating the number of individual functions included in the model, and two floats, the first one indicating the goal value for the reduced χ^2 and the goal value for the variation of this value between consecutive iterations. When one of this limits is reached the fitting process stops.

After this header each function is included in the model indicating its name in one line, followed by 9 lines with 5 columns each one, each of them for each parameter (i.e., the maximun number of parameters allowed in `FIT3D` are 9). These 5 elements column indicate: (1) The guess value of the parameter (a float value), (2) an integer flag indicating if to fit this parameter (=1) or to left it fixed (=0), (3) and (4) the minimun and maximun limits allowed for the parameter to fluctuate, and (5) an integer flag indicating if this parameters is fitted freely (=-1) or if it is linked to the same parameter of another of the function included in the model (= to the number of function in the configuration file, starting with 1). The linking method allows to fit lines that share some properties (e.g., lines that are kinematically coupled) or include lines which line ratio is known (e.g., the line ratio between [OIII]λ5007 and [OIII]λ4959). There are two ways to link parameters: (a) additive, in which the parameter of this function is the parameter of the function indicated by the element number (5), adding a constant. In this case element number (3) indicated the adding factor, and element number (4) must be set to 0. (b) multiplicative, in which the parameter of this function is the parameter of the function indicated by the element number (5), multiplied by a constant. In this case element number (3) indicated the multipling factor, and element number (4) must be set to 1. Although many functions are described with less than 9 parameters, it is needed to fill all of them (with zeroes) to work with the program.

The actual functions included in `FIT3D` are:

- `gauss1d` One dimensional gaussians, which parameters are (in this order): (1) the central wavelength, (2) the peak intensity and (3) the σ of the gaussian function, describing the width (FWHM=2.345σ).
- `eline` A gaussian emission line, which parameters are (in this order): (1) the central wavelength, (2) the integrated flux, (3) the σ of the gaussian

function, describing the width (FWHM=2.345σ), and (4) the velocity in km/s.

- `poly1d` A N-order polynomical function (N <9), with each parameter indicating the parameter of the polynomical function.
- `back` An spectrum background, included in the list of files `BACKGROUND_LIST` (one file per line), which parameters are: (1) order of the background file in the list, (2) the intensity scaling factor, (3) the σ of a gaussian function to convolve the background spectra to match the resolutions and/or the velocity dispersions, and (4) the velocity in km/s to shift the spectrum.

The following shows a particular model, including 4 functions: 2 emission lines, a polynomical function (of order 0) and a background spectrum. The lines and the background are kinematically coupled, defining a system (see below), and two lines have the flux intensity linked:

```
    0  4   0.2  0.001
eline
 5007  1     0      0  -1
  1.0  1   0.2     20  -1
  4.6  1     0      0  -1
12000  1     0      0  -1
    0  0     0      0  -1
    0  0     0      0  -1
    0  0     0      0  -1
    0  0     0      0  -1
    0  0     0      0  -1
eline
 4959  1     0      0  -1
 0.33  1  0.33      1   1
  4.6  1     0      0   1
12000  1     0      0   1
    0  0     0      0  -1
    0  0     0      0  -1
    0  0     0      0  -1
    0  0     0      0  -1
    0  0     0      0  -1
  ...
```

```
   ...
poly1d
  0.01  1  -10   10  -1
     0  0    0    0  -1
     0  0    0    0  -1
     0  0    0    0  -1
     0  0    0    0  -1
     0  0    0    0  -1
     0  0    0    0  -1
     0  0    0    0  -1
     0  0    0    0  -1
  back
     1  0    0    0  -1
  10.0  1  0.2  200  -1
   4.6  1    0    0   1
 12000  1    0    0   1
     0  0    0    0  -1
     0  0    0    0  -1
     0  0    0    0  -1
     0  0    0    0  -1
     0  0    0    0  -1
```

As indicated above, one of the powers of this tool is the possibility of defining systems. A system is a set of emission lines which shares the same velocity and velocity dispersion [4]. This co-moving set of emission lines may be fundamental for deblending several components in complex systems. The continuum can be also coupled kinematically (or not) to a system.

The result of the fitting process is shown in the screen and stored in the following output files:

[4] the dispersion by now is included as a dispersion in wavelength and not in velocity, which may be a problem for large wavelength ranges

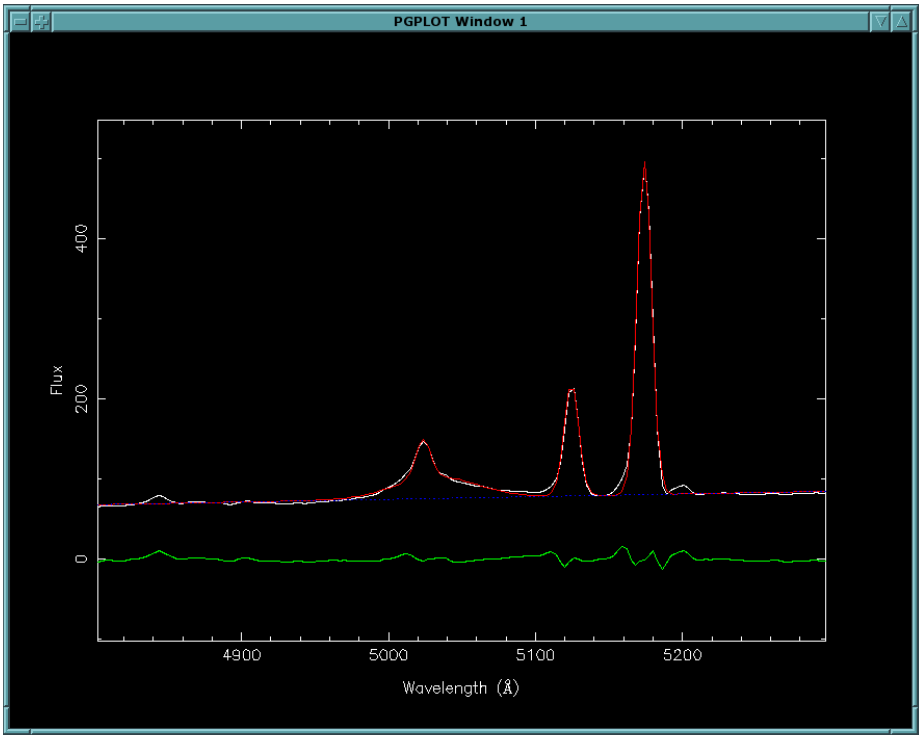

Figure 4: Example of the use of FIT3D. It shows an snapshot of an iteration of fit_spec_back where a QSO spectrum is fitted with 3 emission lines plus a continuum. The white line shows the input data, the red line shows the model, the blue line shows the continuum and the green line the residuals.

- out.fit_spectra, an ASCII file including a header with the number of fitted funstions included in the model, followed by a line per function. Each line contains 19 values: the label of the model, as described above, followed by its 9 output parameters and the corresponding errors.
- out_config.fit_spectra, a configuration file where the output values are included as guessed values. This file is useful when fitting sequentially similar spectra with the same model.
- out_mod_res.fit_spectra, a 5 columns ASCII, corresponding to: (1) the wavelength, (2) the original flux at this wavelength, (3) the derived model, (4) the residual and (5) the residual continuum.

6.1.2 fit_spec_back.pl, a perl wrap to create configuration files

USE:fit_spec_back.pl SPECTRUM.TXT LINE_FILE START_WAVELENGTH END_WAVELENGTH BACK_LIST NBACK MASK_LIST [CONFIG]

This program creates a configuration file for `fit_spec_back`, including the required functions to model the system, and fit the section of the input spectrum corresponding to the selected wavelength range to this model. It requires as input a spectrum in the ASCII format (see above), an ASCII file defining the emission lines to possible fit (see tut4/emission_lines.txt), the wavelength range to be fitted, a file containing the list of backgrounds to use (if not applicable type `none`), the number of backgrounds files, and a file defining the wavelength regions to mask.

The emission line file is an ASCII file with 2 columns, one for the wavelength of the emission line, and another for the name or label to indicate the line. E.g.,

```
5006.84   [OIII]5007
4958.91   [OIII]4959
4861.32        Hbeta
```

The file defining the regions to mask is a 2 columns ASCII file with a first column indicating the starting wavelength of the masking region and the second column indicating the final wavelength of the masking region.

Exercise 33: Fitting and deblending emission lines. *(dir:tut4)*

Using E3D, import the sobj.fits or sobj.e3d file, and display a certain wavelength range centred in the [OIII] emission line, without containing anyother emission line. Display this slice in the spaxels inspector. Select an slice near the central part, and display the corresponding spectrum. Save the spectrum as an ASCII table (e.g., as `spec_cen.txt`*). Exit from E3D. Type:*

`fit_spec_back.pl spec_cen.txt emission_lines.txt 4800 5300 none 0 none`

Answer the questions as follows:

```
Number of systems to include:2
Redshift of the system number 0:0.03
Systemic Velocity=8993.77374 km/s
Flexibility of the velocity (km/s):500
GAUSSIAN sigma of the line (Angstroms!)  (GAUSS,min,max):3 0.1 40
Redshift of the system number 1:0.03
Systemic Velocity=8993.77374 km/s
Flexibility of the velocity (km/s):500
GAUSSIAN sigma of the line (Angstroms!)  (GAUSS,min,max):40 0.1 1000
HeII 4686.0 at 4826.58 found
Include?  (y/n):n
[OIII]5007 5006.84 at 5157.0452 found
Include?  (y/n):y
Flux (Flux, min, Max)?:2000 100 5000
Included
[OIII]4959 4958.91 at 5107.6773 found
Include?  (y/n):y
Flux (Flux, min, Max)?:2000 100 5000
Included
Hβ 4861.32 at 5007.1596 found
```

```
Include? (y/n):y
Flux (Flux, min, Max)?:2000 100 500
Included
[FeII] 4889.62 at 5036.3086 found
Include? (y/n):n
[FeII] 4905.34 at 5052.5002 found
Include? (y/n):n
[FeII] 5111.6299 at 5264.978797 found
Include? (y/n):n
System 0 contains 3 lines
HeII 4686.0 at 4826.58 found
Include? (y/n):n
[OIII]5007 5006.84 at 5157.0452 found
Include? (y/n):n
[OIII]4959 4958.91 at 5107.6773 found
Include? (y/n):n
Hβ 4861.32 at 5007.1596 found
Include? (y/n):y
Flux (Flux, min, Max)?:2000 100 5000
Included
[FeII] 4889.62 at 5036.3086 found
Include? (y/n):n
[FeII] 4905.34 at 5052.5002 found
Include? (y/n):n
[FeII] 5111.6299 at 5264.978797 found
Include? (y/n):n
```

The white line shows the input data, the blue line the model and the red line the model without the continuum. The green line shows the residuals. The 1st fit is very bad, since we have not included any background. We will model the background as a polynomial function. Type the following:

```
cp tmp.config fit_system.config
emacs fit_system.config &
```

This file is the configuration file used to model. It included a header, and the different parameters for each model (read README_fitting_tool.txt). We are going to add a new model to the file, changing the header from:

```
0 4 0.2 0.001
```

to

```
0 5 0.2 0.001
```

and adding at the end of the file:

```
    0  4     0.2  0.001
 poly1d
   100  1  -1e12   1e12  -1
   0.1  1  -1e12   1e12  -1
     0  0      0      0  -1
     0  0      0      0  -1
     0  0      0      0  -1
     0  0      0      0  -1
     0  0      0      0  -1
     0  0      0      0  -1
     0  0      0      0  -1
```

Execute again the fitting procedure, using as input this new config file.

`fit_spec_back.pl spec_cen.txt emission_lines.txt 4800 5300 none 0 none fit_system.config`

The output of the fitting process is illustrated in Figure 4. You can change the parameters until getting an acceptable fit. Play with the parameters, or include a new system to describe the broad line until you get an acceptable fit. As you may notice, all the parameters in each model are defined by four entries, e.g.,

`2000 1 600 100000 -1`

The 1st entry is the input value, the 2nd is flag that can be 0 (to let it fixed to this input value), or 1 fit it, the 3rd and 4th entries are just the range that we allow the parameter to fluctuate, and the 5th parameters is a flag indicating that this parameter is not linked to any other (-1) or it is linked. In that later case it indicates the number of the model (starting from 1) for which this parameters is linked. e.g.,

`8993.77 1 0 0 1`

In this case the parameter of this model is linked to the same parameter of model 1. In this case the 3rd and 4th parameters indicate the `factor/offset` *to apply to the parameter of model 1 (3rd parameters), and if it is an offset (4th parameter equal to 1) or a factor (4th parameters equal to 1). The previous example would apply an offset of 0 (in this case, that the have the same velocity). An example of* `factor` *is:*

`2000 1 0.3333 1 1`

Which links the intensity of the line to the intensity of that of the model 1, applying a factor 0.3333 (i.e., the [OIII]4595 emission line has a flux of 1/3 of that of [OIII]5007).

6.2 Gas Kinematics

Gas kinematics is derived from the fitting over one (or several) system(s) through all the spectra in the dataset. For each position of the corresponding position table it is possible to derive the velocity and velocity dispersion of the system, the intensity of each of the lines in the system, and the continuum level, and also, the error for each of these parameters. In the case of grid-distributed positions in the 3D data (i.e., datacubes), the output of the kinematics analysis would be a set of 2D images (or maps), one per each of these parameters. However, this is not the general case. In the general case it would be required to interpolate the output of the process to create 2D images (although this is not completely required, since it maybe well possible to handle ASCII of fits table similar to the slices we quoted before).

As before, it is possible to use different tools to perform all the process. `FIT3D` includes all the required subroutines to handle the problem in a consistent and simple way (or at least we think that!). We describe here some of these tools.

6.2.1 `kin_back_rss.pl` and `kin_back_cube.pl`

`USE: kin_back_rss.pl rss.fits config_file back_file nback start_wavelength end_wavelength out_file DEVICE`

`USE: kin_back_cube.pl cube.fits config_file back_file nback start_wavelength end_wavelength out_file DEVICE`

These programs perform a fit of each spectrum in the 3D dataset using the model included in the configuration file in an interactive way using `fit_spec_back`. The output of this fitting process is stored in an ASCII output file that consists on the concatenation of each of the `out.fit_spectra` files obtained from the fit over each spectrum. In addition, a RSS (or a datacube) of the original dataset (i.e., the trimmed section fitted), the derived model and the residual are stored in three different fits files: `kin_back_rss_org.fits`, `kin_back_rss_mod.fits` and `kin_back_rss_res.fits` (or `kin_back_cube_org.fits`, `kin_back_cube_mod.fits` and `kin_back_cube_res.fits`).

They require as input a 3D dataset in the RSS format (or in the datacube format), a `fit_spec_back` configuration file, as described before (it is recomended to run `fit_spec_back.pl` over a single and characteristic spectrum before running these programs), an ASCII file containing the list of background spectra and the number of them (if no background spectrum is required, just set `nback` to 0), the wavelength range to fit, and output file (in the ASCII format), and a PGPLOT device for plotting the output.

6.2.2 `mapgrid_back.pl` and `map_interp_back.pl`

`USE: mapgrid_back.pl slice.txt/POS_TABLE out.model PREFIX_OUT`

`USE: map_interp_back.pl slice.txt/POS_TABLE out.model PREFIX_OUT INTERP_MODE GRID_FUNC GRID_PIX`

These programs create flux intensity, velocity and velocity dispersion maps, and their corresponding error maps, for each of the lines fitted using the kinematics analysis programs described in the previous section. The maps are stored in 2D FITs files, reconstructing the original disposition of the spaxels in the sky (in the case of a datacube) or interpolating the data (in the case of a RSS of a non regular grided IFU). They require as input the position table of the dataset (or an ASCII slice obtained with `E3D`, see section5), the output file from the kinematics analysis programs and a prefix for the output FITs files. In the case map_interp_back.pl it is also required three parameters to define the interpolation to perform: (a) the interpolated function used (a number from 1 to 5), (b) an interpolation parameter required for certain interpolation functions, and (c) the final pixel size. These three parameters are describe in the Section 5.3.2, and in the E3D user guide. Based in our experience we recomend to use the natural neighbour interpolation, INTERPO_MODE=3, with a GRID_FUNC=1e-12 and a pixel size of 1/3 of the original spaxel size.

The names of the output maps indicate their content. They start with the indicated prefix, followed there are four possible labels: flux, for the flux intensity, vel, for the velocity, disp, for the velocity dispersion (both of them in km/s), and cont for the background continuum. The names finish with a running number (starting from 0) indicating the emission line that they refer to. Each of these maps has its corresponding error map, that includes an e in the name before the type label. E.g., out_sobj_kin_flux_00.fits and out_sobj_kin_eflux_00.fits are the flux intensity map and its corresponding error map for the first emission line analyzed in a kinematics study, using out_sobj_kin as prefix.

Exercise 34: Kinematics analysis.

Using E3D*, import the* sobj.fits *or* sobj.e3d *file and store it as a RSS (*sobj_rss.fits*). It is recommended not to store all the file, but just the wavelength range between 4800-5300Å to speed up the process. We will use the same configuration file as the one used in the previous exercise. In a general case it is recommended to fit one spectrum before performing the semi-automatic analysis to run the kinematics. We execute the program kin_back_rss.pl:*

```
kin_back_rss.pl sobj.rss.fits fit_system_tunned.config null 0 4800
5300 out_sobj_kin.txt /xs
```

This program will go throughout all the spectra in the dataset, fitting each of this spectrum using fit_spec_back*, and the configuration file* tmp.config*. Each time a fit is performed it is possible to execute the following options:*

s *to store the results and jump to the next spectrum.*

a *perform a non interactive automatic fit over all the dataset.*

r *repeat the fit.*

Since the program opens an emacs *terminal with current configuration file*[5]*, it is possible to change it and repeat the fitting until an acceptable result is obtained. The program store the results from each individual fit in a logfile,* out_sobj_kin.txt *in the example, storing the resulting model and residual spectra in the FITs files* kin_back_rss_mod.fits *and* kin_back_rss_res.fits*.*

Perform an automatic fit using a*, to speed-up the exercise.*

It is possible to check the results with IRAF and ds9 *(or a similar tool), using the commands:*

```
disp kin_back_rss_org.fits 1
disp kin_back_rss_mod.fits 1
disp kin_back_rss_res.fits 1
```

[5]If you are not an emacs fun or you do not have if in your system, please edit the kin_back_rss.pl and change it for your prefered editor.

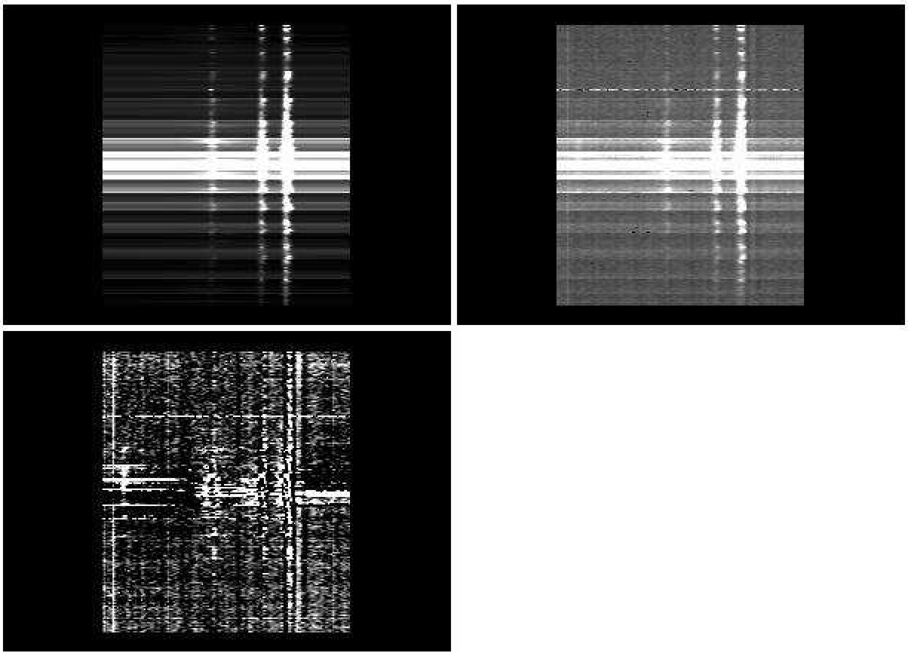

Figure 5: Example of the kinematics analsys. The top-left panel shows the derived model using the kinematics analysis in the Exercise 34. The top-right panel shows the original RSS dataset, and the botton-left panel the residuals after subtracting the model.

as illustrated in Figure 5.

As explained above, it is possible to create 2D maps for the different fitted parameters, interpolating if required (in the case of RSS data) with the program:

`map_interp_back.pl sobj.pt.txt out_sobj_kin.txt out_sobj_kin 3 1e-12 1`

Once created the maps, it is possible to visualize them using IRAF:

`disp out_sobj_kin_flux_00.fits 1`

`disp out_sobj_kin_vel_00.fits 1`

The displayed results can be seen in Figure 6.

In the case of the velocity dispersion it is required to substract to the observed values the instrumental component. The instrumental component comprises all the dispersions in the lines that are not physically associated with the studied source, but due to the instrument, the night conditions, or whatever other effect. The standard procedure for doing this is to perform a kinematics analysis over a nearby emission-line (over the non sky-substracted data), and subtract the obtained velocity dispersion to the observed in the object (with a quadratic subtraction).

It is important to note here that the velocity dispersion is fitted assuming

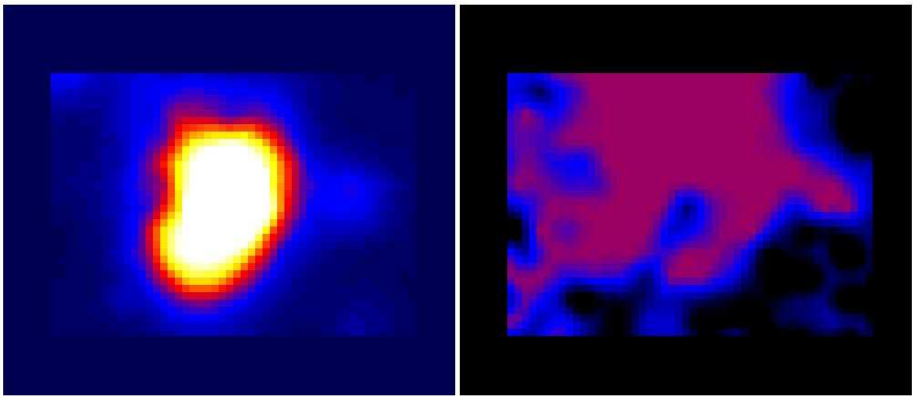

Figure 6: Example of the use of `FIT3D`. It shows the flux intensity map (left panel) together with a velocity map (right panel) derived with the automatic line fitting performed in Exercise 34.

a dispersion in Å, what it is not a good approach for wide wavelength ranges. Kinematics analysis are performed (normally) over a narrow wavelength range, centred in the emission lines of interest, and it is due to this reason that this approach is used. With this approach is also more easy to decontaminate from instrumental effects in the velocity dispersion.

7 Decoupling spectra of different objects: Crowded Field Spectroscopy

IFS or 3D data can be understood as a set of narrow-band images of the same objects at different wavelength. On the assumption of a smooth change in the morphological characteristics of the objects in the 3D data, it is possible to apply techniques of decoupling or deblending normally used in the analysis of 2D imaging, and extended them to 3D (García-Lorenzo et al. 2005; Sánchez et al. 2006a). This method is particularly interesting when we have low-signal-to-noise images of objects with the same morphology in the image (e.g., QSO lenses Wisotzki et al. 2003), when we have a crowded field where we want to deblend the spectra (e.g., the core of globular clusters or galaxy clusters Sánchez et al. 2006b), or when we want to deblend two well define components on the same object (e.g., QSO-hosts or bulge-to-disk decomposition, or point-like sources over bright backgrounds, like in XPNs Sánchez et al. 2004).

There are different implementations of this technique in the literature (Roth et al. 2004; Wisotski et al. 2004; Sánchez et al. 2004; Garcia-Lorenzo et al. 2005; Fabrika et al. 2005; Sánchez et al. 2006a,b). We are not going to describe all of them, but just a particular case, that is the extension of `galfit` to 3D, or `galfit3d`(Sánchez et al. 2006a; Sánchez et al. 2006b). `galfit` (Peng et al. 2002), it is a program for modelling 2D images of one or several objects using analytical models for galaxies and/or point-like sources, convolving them with a PSF.

In the case of `galfit3d`, we have extended the technique to model regular-gridded 3D data (ie., datacubes), which maybe interpolated versions of none regular ones (e.g., Garcia-Lorenzo et al. 2005), and decouple the spectra of different objects in the field and/or deblend the spectra from different components of the same object (Sánchez et al. 2006a; Sánchez et al. 2006b). A major advantage of IFS data is that it is possible to impose links over the parameters used for each of the N-fits (one per wavelength), on the basis of a smooth changing of the morphological parameters, or we can force to a particular value, if we have a source of external information (e.g., high-resolution HST images García-Lorenzo et al. 2005; Sánchez et al. 2006b). In both cases the fitting process is iterative, with a first loop of non acceptable results, and a set of iterations where the morphological parameters are fixed to different values, using as input parameters the output of the previous fits (Sánchez et al. 2006a).

7.1 Components of `galfit3d`

`galfit3d` comprises a set of `perl` scripts:

7.1.1 `galfit3d.pl`

`USE: galfit3d.pl INPUT_CUBE.fits MOD_CUBE.fits RES_CUBE.fits GALFIT_CONF LOGFILE QUALITY.txt`

`GALFIT_CONF: A) object.fits, B) output.fits`

This program performs a modelling of an input datacube creating, for each wavelength, a narrow-band image of the width of a spectral pixel, and fitting it using `galfit` (Peng et al. 2002). The output of the fitting is stored in a logfile, which consist of the concatenation of the different outputs (e.g., `galfit.01`) produced by `galfit` for each wavelength. A quality logfile is also stored, with an integer flag that indicates if the fitting process has finished the minimization (1) or end in an abrut break (0).

7.1.2 `galfit3d_force.pl`

`USE: galfit3d_force.pl INPUT_CUBE.fits MOD_CUBE.fits RES_CUBE.fits GALFIT_CONF INPUT_LOGFILE INPUT_NPOLY OUTPUT_LOGFILE QUALITY.txt [PLOT]`

This program works like `galfit3d.pl`, but using as guessed values for the fit the output values of a previous iteration (`INPUT_LOGFILE`). Each parameter can be smoothed using a fit over a polynomical function, or force to be a selected value. This is indicated with the `INPUT_NPOLY` configuration file, which is a 3 columns ASCII file which elements per line are: (1) the number of the model in the `galfit` configuration file which parameter wants to be smoothed, (2) the number of parameter of this model to smooth, and (3) the order of the polynomical function to fit. If the parameter (3) is a float instead of an integer the program performs a substitution by this value, instead of a polynomical function smoothing.

7.1.3 `extract_par_g3d.pl`

`USE: extract_par_g3d.pl INPUT_LOGFILE QUALITY.txt OUTPUT_FILE model_number par_number DEVICE [min max]`

This program allows to extract the fitted value of a parameter of a certain model along the spectral pixels and store it in an ASCII output file, plotting this parameter along the spectral pixels using PGPLOT, in defined device. It requires as input the output logfiles obtained with `galfit3d.pl` or `galfit3d_force.pl`, the name of the output file to store the results, the number of the model and the number of the parameter to extract.

7.1.4 `extract_spec_g3d.pl`

`USE: extract_spec_g3d.pl INPUT_LOGFILE QUALITY.txt OUTPUT_FILE model_number mag_zeropoint CRVAL CDELT`

This program allows to extract the integrated spectrum corresponding to a certain object or component of an object modelized with a certain 2D model, and store it as an ASCII spectrum (see in Sec. 5). It requires as input the output logfiles obtained with `galfit3d.pl` or `galfit3d_force.pl`, the name of the output file to store the results, the number of the model in the `galfit` configuration file, the zeropoint of the magnitude indicated in that configuration file, the starting wavelength and the wavelength step of the fitted datacube.

7.2 Installation of `galfit3d`

`galfit3d` is still under development. We distribute here a beta version, without any guarantee of its perfect working. A large number of simulations and a deep understanding of the procedure is still needed before producing reasonable outputs in the most general case. It requires to have installed `galfit` and `IRAF` in your computer. We recomend to change the source code of `galfit` in order to increase the acuracy of the produced output, not to be truncated to the second decimal of the magnitude (see `galfit` documentation). As a set of `perl` scripts, it requires the modules described in Section 4. Once installed this modules it is needed to change the following lines in the `perl` scripts for the corresponding directories in your system:

```
require /data/sanchez/code/R3D/my.pl;
$galfit="/data/sanchez/galfit/galfit";
$cl="/usr/local/bin/cl";
```

Exercise 35: 3D modelling. *It is out of the scope of this tutorial to make a complete description of the deblending and modelling options for 3D spectroscopy. Therefore, we will perform a very simple modelling over GMOS data. The GMOS data we present here are of a nearby QSO, and we would like to deblend the bright QSO from the extended Host galaxy. To do so, the model* *incudes a model for the point-like source (a Gaussian convolved with a PSF), and a model for the host galaxy (a Sersic profile, with the sersic index equal to 1, i.e, a Disk galaxy). We will force the fit asuming a certain scale-lenght, ellipticity and position angle for the galaxy, and we will not perform a second iteration.*

Run the program:

```
galfit3d.pl minput.cube.fits input_cube.mod.fits input_cube.res.fits
galfit.qso_fix input_cube.fit.log input_cube.fit.quality
```

The program will go throught-out all the slices in the data set and fit them with the proposed model. The output of the fitting is stored in two files: `input_cube.fit.log` *the log file, with the direct output from galfit, and* `input_cube.fit.quality` *a file containing a flag indicating if the fitting ends or crashes. It also produce two datacubes, one with the fitted model, and another with the residual from the fitting technique.*

It is possible to inspect the progress of the fit, and visualize any of the fitted parameters, by executing:

```
extract_par_g3d.pl input_cube.fit.log input_cube.fit.quality junk.junk
1 3 /xs 22 18
```

It is also possible to visualize each of the individual fits using iraf, by typing:

```
cl < display.cl
```

Once finished the fit, it is possible to extract the deblended spectra of each components by executing.

```
extract_spec_g3d.pl input_cube.fit.log input_cube.fit.quality spec_nuc.txt
1 25 6600 2.3
extract_spec_g3d.pl input_cube.fit.log input_cube.fit.quality spec_host.txt
2 25 6600 2.3
```

8 DAR correction using external tools.

As we described before, `E3D` contains its own tools to perfor the differential atmospheric refraction (DAR) correction. However they are difficult to control and in many case they do not produce accurate results, being valid only for a quick & dirty correction.

We discuss here some tools that perform empirical DAR corrections for datacubes. As we described before any DAR correction requires an interpolation of the data in the spatial direction, which produce a lose of the initial spatial configuration of the spaxels. Therefore, this requirement is not that critical. These corrections are empiral, i.e., the DAR correction is determined from the data, what is an unique feature of 3D spectroscopy. For doing that it is required to have a bright object with a well defined intensity peak at any wavelength. The empirical correction is determined by looking to the spatial coordinates of the intensity peak for each wavelength. In many cases the empirical correction work better than the theoretical one, based on the (Filippenko 1982) formulae. The reason is that the theory behind these formulae assumes that the layer of the atmosphere with equal refraction index are flat and perpendicular to the

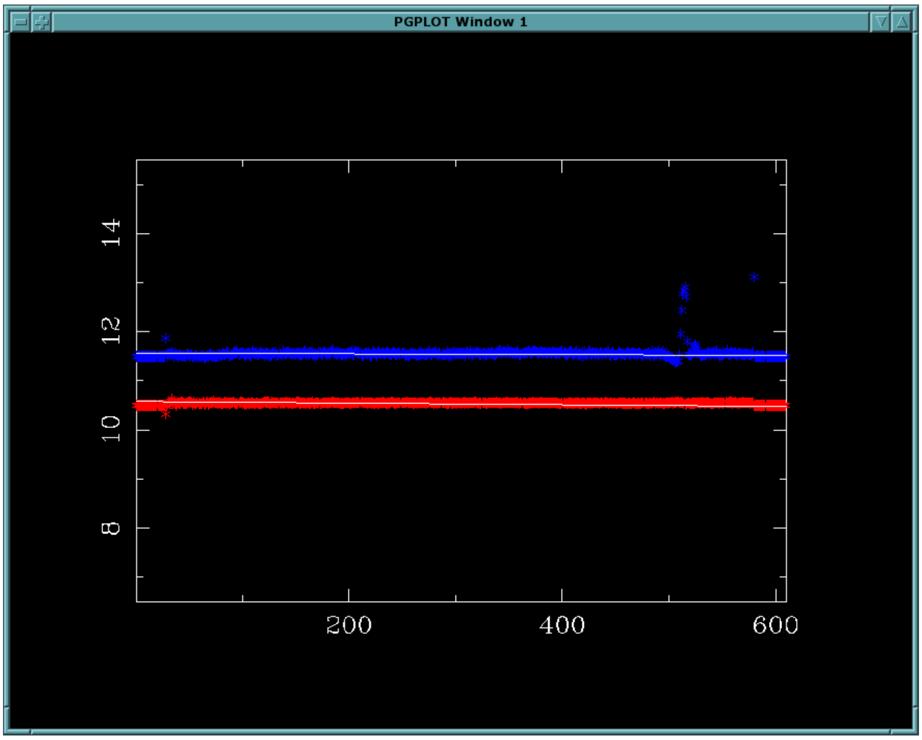

Figure 7: Example of the output of the script `imcntr_cube.pl`. It shows in blue and red the X and Y coordinates of the centroid of a star along the spectral pixels or channels in a datacube.

zenit of the telescope. However, it is well known that this in only a first order approximation, since these layer depends strongly on the horography of the observatory.

These tools are:

8.1 `imcntr_cube.pl`

`USE: imcntr_cube.pl input_cube.fits X Y NP_X NP_Y centers_list.txt [N_START] [N_END]`

This program determines the centroid of a certain object around a guessed position for each wavelength, performing an smoothing of the centroid along the wavelength by a polynomical function fitting. It stores the results in a two-columns ASCII file. It requires as input the datacube over which perform the analysis, the guessed centroid of the object, the order of the polynomical functions to fit for the X and Y coordinate, and the ouput file containing the results. In addition is possible to exclude a certain number of spectral pixels in the beginning and the end of the datacube from the analysis.

8.2 cube_shift.pl

USE: `cube_shift.pl input_cube.fits output_cube.fits centers_list.txt X0 Y0 shift.cl`

This program recenters a certain object along each wavelength to a common coordinate, correcting efectively the DAR, using as reference the centroids list determined with the previous program. It requires as input the input and output datacubes, the centers list, the final coordinates of the object and an output IRAF script. This final script can be edited in order to perform the same correction over another datacube. This option is interesting for datacubes where there is no bright object at any wavelength to trace the centroid, for which it is possible to use observations performed over a bright nearby star (a somehow DAR calibration).

8.3 cube_shift_delta.pl

USE: `cube_shift_delta.pl input_cube.fits output_cube.fits delta_x delta_y`

This program performs a global shift of a datacube in the spatial direction. It requires as input the input and output datacubes and the shifts in the X and Y axis.

To use these scripts it is required to edit the files and change the line,

`require /data/sanchez/perl/MY/my.pl";`

for the corresponding one on your architecture. It is also needed to run an `mkiraf` before running the programs in the directory where you have the dataset, since these programs are just IRAF wrappers.

Exercise 36: Empiral DAR correction. *(dir:tut3)*

Using IRAF with DS9 (or whatever similar tool), display the channel 300 of the file `star_no_dar.cube.fits`*, e.g.,*

`disp star_no_dar.cube.fits[*,*,300]`

and determine the centroid of the star in the field, using `imexam` *or another similar tool. The centroid is around the coordinates (10.5,11.5). We determine the centroids along the wavelength:*

`imcntr_cube.pl star_dar.cube.fits 10.5 11.5 2 2 centers_list.txt 10.5 11.5 7.5 14.5`

The output of this program is illustrated in Figure 7, where the distribution of the X and Y coordinates of the centroid of the star along the wavelength (in this case the pixels in the dispersion axis) is represented. The list of centroids is stored in the `centers_list.txt` *file, used as input in the correction process:*

`cube_shift.pl star_dar.cube.fits star_no_dar.cube.fits centers.lis 10.5 11.5 shift.cl`

The file `star_no_dar.cube.fits` *contains the DAR corrected datacube.*

9 Reduction of fiber-feed IFS data

9.1 The raw data

In most cases, the raw data of fiber-feed spectrographs consist of a collection of spectra distributed along a 2D frame, aligned along the so called dispersion axis. In general, this axis can be the X or Y-axis or none of them, although it uses to be almost aligned with one of them. Figure 8 shows an example of IFS raw data, corresponding to PMAS (Roth et al. 2005) in the PPAK mode (Kelz et al. 2006), illustrating this distribution. Each spectrum is projected over a certain number of pixels along a perpendicular direction, the so called cross-dispersion axis, following a profile similar to an asymetrical gaussian or a kurtosis function. Therefore, there is a contamination from each spectrum to the adjacent ones, i.e., the so-called cross-talk (see Fig. 8). The profile changes for each spectrum along the cross-dispersion axis, and in general is more asymetrical in the borders of the pseudo-slit than in the center. In general terms, it also changes along the dispersion axis. Well constructed instruments have well defined and narrow profiles, and a limitated cross-talk. However, there is always a compromise between the number of fibers "packed" in the spectrograph, the size of the CCD and the width of the profiles and cross-talk. As we quoted before, the spectra are not perfectly aligned along the dispersion axis, due to the configuration of the instrument, its setup, the instrument focus and the flexures (if any). In addition to that, the transmission changes fiber-to-fiber due to physical differences among them, differences in the tension suffered by the fibers at different positions, and slight missalignements of the fibers in the pseudo-slit. These effects affect also the width of the projected profiles across the cross-dispersion and the dispersion axis, and introduce small differences in the spectral dispersion fiber-to-fiber.

9.2 IFU data reduction

The reduction of all fiber-feed IFUs share a similar sequence of steps independetly of the instrument, due to the similarities in their raw data. We assume that the data have been bias and flat-field corrected (when possible), with whatever external tool, as an starting point for the reduction. After these corrections, the data reduction consist of: (a) identification of the position of the spectra in the detector for each pixel along the dispersion axis, (b) extraction of each individual spectrum, (c) distortion correction of the extracted spectra and determination of the wavelength solution (i.e., dispersion correction) and (d) re-arranging of the spectra in the sky position. Each of these reduction steps have been treated with a set of command-line routines in `R3D` (Sánchez 2006), a package written explitally for reducing fiber-feed IFS data.

We briefly describe here the different steps required to perform a reduction:

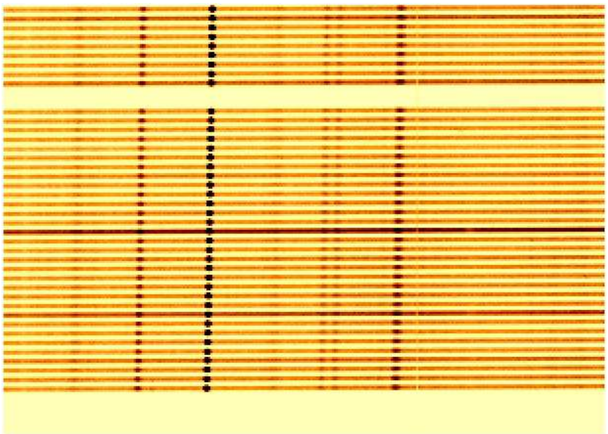

Figure 8: Grayscale of a section of IFS raw data correponding to PMAS in the PPAK mode, in inverse colors (i.e., more intense areas are shown in darker colors). Each dark line corresponds to the projection of a spectrum along the dispersion axis, which in this case corresponds to the X-axis. Spectra are separated 7 pixels across the Y-axis, and projected in a few pixels, contaminating the adjacent spectra.

9.2.1 Identification of the position of the spectra in the detector

For doing that you need a continuum exposure (e.g., a domeflat), that in the case of instruments that suffer from fleaxures (i.e., displacement of the spectra in the CCD at different positions of the telescope), should be taken with the telescope pointing to the same location of your science dataset. The location of the spectra is found at a certain column in the CCD by comparing the intensity at each row along the column (or a coadded set of rows around it) with that of n adjacent pixels, looking for those pixels that verify the maximum criteria, using `peak_find` .

`USE: peak_find RAW.fits Spectral_axis[0/1] Coadd_width plot nplot nsearch DMIN IMIN(% of the MAX) OUTFILE [colum_search]`

This program requires as input the continuum raw data frame, a flag indicating the dispersion axis (0 for X, 1 for Y), the width of the window around the central colum to coadd the intesity (for low-level signal-to-noise frames), a flag indicating if to plot (1) or not (0) the results, the number of subframes to plot in the screen, and number of adjacents pixels to check the maximum criterium, the minimum distance between adjacent spectra (projected in the cross-dispersion axis), and the fraction of the intensity peak to use threshold for detecting intensity peaks (i.e., spectra), and an ASCII output file containing the location along the cross-dispersion axis of the detected peaks.

After that, the tracing of the peak intensity along the dispersion axis is done by looking for maxima around each original location within a certain pixel window, using `trace_peaks_recursive`.

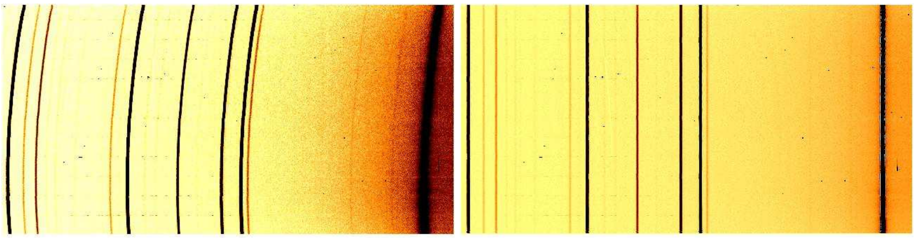

Figure 9: Left panel: Example of IFS extracted spectra, a calibration arc lamp obtained with PMAS in the PPAK mode, before correcting for the distorsion in the dispersion along the cross-dispersion axis. The X-axis corresponds to the dispersion axis. Right panel: Same example after correcting for the distorsion

`USE: trace_peaks_recursive RAW.fits Spectral_axis[0/1] PEAKS_FILE coadd_width search_width plot nplot nsearch TRACE.fits [y_shift] [2nd_search_width]`

This program requires as input the continuum raw data frame, a flag indicating the dispersion axis (0 for X, 1 for Y), the file containing the location of the intensity peaks (i.e., the output of the previous program), the width of the window around the each colum to coadd the intesity along the dispersion axis, the width of pixels to look for the new centroid of the peak along the cross-dispersion axis, a flag indicating if to plot (1) or not (0) the results, the number of subframes to plot in the screen, and number of adjacents pixels to check the maximum criterium, and the output file, a 2D FITs file where the X-axis corresponds to the original dispersion axis and the Y-axis indicated the number of traced spectrum. The stored value at each pixel is the location of the peak centroid in the original frame of the corresponding spectrum at the corresponding spectral pixel. This file is name the *trace*.

9.2.2 Extraction of the individual spectrum

After tracing the location of the spectra in the CCD, the next reduction step is to extract the flux corresponding to each spectrum at each pixel along the dispersion axis. The simpliest method to perform this extraction is to coadd the flux within a certain aperture around the location of the spectra in the raw data, using the tracing information, and store it in a 2D image. This procedure is named aperture extraction, and it has been implemented in `R3D`. The X-axis of the resulting image correponds to the original dispersion axis, while the Y-axis corresponds to the ordering of the spectra along the pseudo-slit. This is the so-called row-stacked spectra representation (Sánchez 2004). The extraction is performed by `extract_aper`.

`USE: extract_aper RAW.fits Spectral_axis[0/1] TRACE.fits Aperture OUTPUT.fits [SHIFT]`

This program requires as input a raw data frame, a flag indicating the dispersion axis (0 for X, 1 for Y), the trace frame, the aperture width for the extraction and the name of the output file.

9.2.3 Distorsion correction

Most of the spectrographs do not dispers the light homogenously along the cross-dispersion axis. The dispersion is distorted, being larger in the borders of the slit than in the center. This distorsion is sometimes called the C distorsion, due to its appearance in the CCD. In the case of fiber-feed spectrographs, aditional distorsions are introduced due to the placing of the fibers in the pseudoslit. Once extracted the spectra these distorsions are more evident. Figure 9, left panel, shows an example of extracted spectra, corresponding to a calibration arc-lamp observed with PMAS in the PPAK mode. The C distorsion and the discontinuities in the dispersion solution along the cross-dispersion are seen. These discontinuities and the slight differences in the dispersion fiber-to-fiber, do not allow to perform a 2D modelling of the distorsion map with an analytical function, nor at the level of the extracted spectra, neither at the level of the raw data. The distorsions have to be corrected fiber-to-fiber before finding a common wavelength solution.

`R3D` performs this correction by a two steps procedure, using arc calibration lamp exposures (like the one in Fig.9). In a first step the peak intensity of a single emission line is traced along the cross-dispersion axis, and shifted to a common reference, by a linear shift. This is done by the program `dist_cor`.

```
USE: dist_cor EXTRACTED.fits CORRECTED.fits DISTORSION_CORRECTION.txt
SMOOTH[0/1] start_index delta_index nsearch plot [n_fib] [nsigma] [center_index]
```

This program require as input the aperture extracted file, the output corrected file, the output first order distorsion file, a flag indicating if to smooth (1) or not (0) the solution along the cross-dispersion axis, the reference pixel in the dispersion axis to look for the emission line peak, the width of the window in pixels to look for this peak, the number of adjacents pixels to check the maximum criterium and a flag indicating if to plot (1) or not (0) the results.

After that, the intensity peak of a set of selected emission lines is traced, and a polynomical distorsion correction is determined to recenter all the lines to a common reference. This is done by the program `mdist_cor_sp` .

```
USE: mdist_cor_sp EXTRACTED.fits APERTURE NSIGMA NPOLY OUTPUT.fits
DISTORSION.fits NX_min NX_max plot [NSIGMA2] [BOX] [NY_CEN]
```

This program require as input the output of the first order correction, an aperture to look for emission lines, a nσ threshold over the average intensity to look for emission line peaks, the order of the polynomical function to fit the distorsion correction (spectrum to spectrum), the output file with the distorsion corrected frame, the output file containing the 2D map of the corrections, the range of pixels in the dispersion direction to consider for the analysis, and a flag indicating if to plot (1) or not (0) the results.

9.2.4 Wavelength solution

The wavelength coordinate system is determined by identifing the wavelengths of the arc emission lines, using an interactive routine. The distorsion-corrected spectra of the arcs are then transformed to a linear wavelength coordinate system by a one dimensional spline interpolation, assuming a polinomical transformation between both coordinate systems. The required transformation is stored in an ASCII file to be applyied over the science data. This procedure is done using the `disp_cor.pl` script:

`USE: disp_cor.pl EXTRACTED.fits CRVAL CDELT APERTURE NY_SPECTRA WITDH NPOLY OUTPUT.fits DISPERSION.txt plot`

This program requires as input the output of the distorsion correction analysis, the starting wavelength and wavelength step per final spectral pixel, the aperture to look for emission lines, the row of the central spectrum over which perform the analysis, width of pixels in the cross-dispersion axis for coadding the spectra (to increase the signal-to-noise ratio), the order of the polynomical function to fit the wavelength solution, the output file containing the spectra transformed to a linear wavelength coordinate system, the output file containing the dispersion or wavelength solution applied and a flag indicating if to plot (1) or not (0) the results.

9.2.5 Additional steps

After performing these basic steps of the reduction it may be needed to correct for the differences in the fiber-to-fiber transmission, combine different exposures with offsets, correct for the differential atmorpheric refraction, subtract the sky from your spectra and perform a flux calibration. `R3D` includes tools that perform some of these operations, but it is out of the scope of this quick reference to mention all of them in detail here. Evenmore, considering that most of them depends strongly on the considered instrument.

9.2.6 Exercises

Exercise 37: Reducing data with `R3D`. [6] *(dir:tut6)*

We are going to perform a simple data reduction over PMAS (Roth et al. 2005) data in the PPAK mode (Kelz et al. 2006) using `R3D`*. First, we determine the number of spectra and their location along the cross-dispersion axis for the central column of the CCD:*

`peak_find.pl DomeFF_Cont_Galaxy.fits 0 2 1 4 2 3.5 0.10 DomeFF_Cont_Galaxy.peaks`

After that, we trace the location of the peak intensity of the spectral along the cross-dispersion axis of each pixel in the dispersion axis:

`trace_peaks_recursive DomeFF_Cont_Galaxy.fits 0 DomeFF_Cont_Galaxy.peaks 2 2 1 4 1 DomeFF_Cont_Galaxy.trc.fits`

Once determined the location of the spectra in the CCD, it is possible to extract them:

`extract_aper.pl IRAS18131_A.fits 0 DomeFF_Cont_Galaxy.trc.fits 5 IRAS18131_A.ms.fits`

The C distorsion correction is them determined and corrected:

[6] Data provided by K.Kehring

```
dist_cor DomeHe_ThAr.ms.fits DomeHe_ThAr.dc0.fits DomeHe_ThAr.dist.txt
0 662 30 2 1 2 2
mdist_cor_sp DomeHe_ThAr.dc0.fits 3 300 3 DomeHe_ThAr.dc1.fits DomeHe_ThAr.dist.fits
17 1011 1 2 5 3
```

The distorsion correction determined using the He lamp exposure can be applied to the science data:

```
mdist_cor_external IRAS18131_A.ms.fits DomeHe_ThAr.dist.txt DomeHe_ThAr.dist.fits
IRAS18131_A.dc1.fits 0 1.353
```

After fransformed to a common reference, it is possible to find the wavelength coordinate solution:

```
disp_cor.pl DomeHe_ThAr.dc1.fits 3700 1.6 3 180 3 4 DomeHe_ThAr.disp_cor.fits
DomeHe_ThAr.disp.txt 1
```

and apply them to the data:

```
disp_cor_external.pl IRAS18131_A.dc1.fits 3700 1.6 IRAS18131_A.disp_cor.fits
DomeHe_ThAr.disp.txt 0
```

The differences in the fiber-to-fiber transmission can be determined using dome-flat exposures, and correct the science frames for them:

```
fiber_flat.pl mDomeFF_Cont_IRAS18131_A.disp_cor.fits fiber_flat_IRAS18131_A.fits
imarith.pl IRAS18131_A.disp_cor.fits / fiber_flat_IRAS18131_A.fits
IRAS18131_A.fc.fits
```

References

[Becker 2001] Becker, T., 2001, PhD Thesis, Potsdam University

[Fabrika et al.2005] Fabrika, S., Sholukhova, O., Becker, T., Afanasiev, V., Roth, M., & Sanchez, S. F. 2005, A&A, 437, 217

[Filippenko 1982] Filippenko, A. V. 1982, PASP, 94, 715

[García-Lorenzo et al. 2005] García-Lorenzo,B, Sánchez, S.F., Mediavilla, E. González-Serrano, J.I., Christensen, L., 2005, ApJ, 621, 146

[Glazebrook et al 2001] Glazebrook, http://pdl.perl.org/

[Kelz et al. 2006] Kelz, A., Verheijen, M.A.W., Roth, M.M. et al., 2006, PASP, in press (astro-ph/0512557)

[Kissler-Patig et al. 2004] Kissler-Patig, M., Copin, Y., Ferruit, P., Pécontal-Rousset, A., & Roth, M. M. 2004, AN, 325, 159

[Pécontal-Rousset et al. 2004] Pécontal-Rousset, A., Copin, Y., & Ferruit, P. 2004, AN, 325, 163

[Pearson 1995] Pearson, T., 1995, http://www.astro.caltech.edu/∼tjp/pgplot/

[Peng et al. 2002] Peng, C.Y., Ho, L.C., Impey, C.D., Rix, H., 2002, AJ, 124, 266

[Roth et al. 2004] Roth, M. M., Becker, T., Kelz, A., & Schmoll, J. 2004, ApJ, 603, 531

[Roth et al. 2005] Roth, M.M., Kelz, A., Fechner, T., et al., 2005, PASP, 117, 620

[Sánchez 2004] Sánchez, S. F. 2004, AN, 325, 167

[Sánchez et al. 2004] Sánchez S.F., García-Lorenzo, B., Mediavilla, E., et al., 2004, ApJ, 615, 156

[Sánchez 2006] Sánchez, S. F., in preparation.

[Sánchez et al. 2006a] Sánchez S.F., García-Lorenzo, B., Jahnke, K., et al., 2006, New Astronomy, in press.

[Sánchez et al. 2006b] Sanchez, S. F., Cardiel, N., Verheijen, M., & Benitez, N. 2006, proceedings of the ESO and Euro3D workshop "Science Perspectives for 3D Spectroscopy", Garching (Germany), October 10-14, 2005 (astro-ph/0512550)

[Walsh & Roth 2002] Walsh, J. R. & Roth, M. M.: 2002, The Messenger, 109, 54

[Wisotzki et al. 2003] Wisotzki, L., Becker, T., Christensen, L., Helms, A., Jahnke, K., Kelz, A., Roth, M. M., & Sanchez, S. F. 2003, A&A, 408, 455