Project Report: "Binarity and the distribution of rotational velocities in Galactic main-sequence B stars"

Release 2.5

B. C. Mota, C. Martayan, T. Rivinius, D. Moser and A. Shoky

April 15, 2016

CONTENTS

For several types of stars the distribution of rotational velocities (v sini) is bi- or even multi-modal and is a key tracer of star formation and evolution. Binarity and magnetism are among the main contenders for the explanation. Nominally, the frequency of binaries is a strong function of stellar mass, and the distribution of vsini values appears bimodal in O and A stars and the situation is unclear in B stars (flat or bimodal distribution). But this may be entirely spurious. The sample of telluric standard stars used with X-shooter offers a both large and inexpensive initial database comprising more than a thousand B-type stars. The proposed project aims at consolidating and homogenizing the database, extracting the fraction of binaries and the distribution of v sini for analysis in a star formation and evolution context, and improving the quality of telluric standard stars available for X-shooter and other Paranal instruments.

Full goals:

1. Inspect the data and reduce all spectra with an homogeneous manner and with adapted parameters depending on the sky condition and objects. It is not possible to use the automatically reduced data from the archive because of:

1. the very different sky conditions (extraction problem);
2. the presence of visual binaries/companions (extraction problem);
3. the response curve used is an old one;

2. find the binaries in the sample, looking at the acquisition frames, at the spectra (SB2 or multiple stars from the U to the K band), and with the radial velocity follow-up (SB1);
3. determine if the binaries may have an impact on the evolution and day to day life of the main star of the system;
4. determine the fundamental parameters (T_{eff}, logg, vsini, M, etc) and create observational tracks of the rotational velocity evolution in the Main Sequence for B stars. Determine the chemical abundances. And for the binaries to determine the orbital parameters if possible;
5. provide clean catalogues of telluric standard stars (good spectral classification, no binary) to Paranal and other observatories. This legacy aspect was highlighted by the OPC in its comments;
6. give constraints to the theoretical models and cooperate with the Geneva team lead by G. Meynet.

The purpose of this DGDF is to get some help on items a and b. The project at longer time scale is to obtain a student with Geneva to work on other items and enlarge the cooperation observations/theory.

CHAPTER ONE

INTRODUCTION

The B stars represent a significant fraction of massive stars. If only stochastic processes at the time of formation govern the distribution of rotational velocities, the distribution of the latter in any sample of stars of similar age and mass should be smooth and in particular not have multiple maxima. However, exactly the latter is observed in some groups of stars. For both O and A stars the rotational velocity distribution appears to be bimodal. For A main sequence stars (cleaned of binaries and peculiar stars), Royer et al. (2007, A&A,463,671) found this bimodal distribution and explained it by an angular momentum loss and its redistribution in the star before it reaches the main sequence (MS). Zorec & Royer (2012,A&A,537,A120) studied the evolution of A type stars rotational velocities and found 2 different behaviours. The less massive A stars have a monomodal equatorial velocity distribution and have a monotonical acceleration with age during the MS. The most massive A stars show a strong acceleration in the first third part of MS but do not show any angular momentum redistribution in the last third of the MS, which could be related to some differential rotation. Guthrie (1982,MNRAS,198,795) also found a bimodal rotational velocity distribution for late B stars in open clusters while B stars in the field show a flat distribution. The author argues that this difference should come from the way the stars formed.

While Abt et al. (2002, ApJ,573,359) explain that the distributions are massdependent and would be explained by magnetic braking or the presence of lowmass companion or planets. Dufton et al. (2013, A&A, 550, A109) also found for late O to early B in the LMC clusters of the Tarantula region a bimodal or even more complex rotational velocity distribution. They argue that this could be explained by similar mechanism that occurs in A type stars. However, if potential binaries are removed, because the spectral classification has not been performed, one cannot disentangle the effects of stellar evolution from original multi-modal distribution of the rotational velocities.

1.1 Impact on the stellar models

Standard evolutionary tracks do not explain this. The two most relevant additional ingredients are magnetic braking and spin-up in binaries through mass transfer or even mergers. Not only have these properties opposite effects but each of them alone can lead to a bi-modal distribution. Moreover, in binaries, the distribution of angular momentum between spin and orbit introduces an additional complex parameter during the period of their formation. The challenge is to disentangle them, especially if binarity and magnetic-field strength depend on mass. An obvious prerequisite for an improved understanding is to know the initial frequencies of relevant binarity and magnetism. Here, we focus on the binarity of at most moderately evolved B-type stars. In addition this knowledge would possibly also allow understanding some discrepancies between theoretical models of stellar and chemical evolution and some observational results (Hunter et al. 2007,A&A,466,277). Possibly unknown angular momentum transfer phenomenon may have to be discovered and taken into account in the models.

1.2 Binarity

To avoid a bias one must find first the binaries in the sample. The fraction of binaries can reach up to 75 % of O stars (Sana et al. 2012, Sci,337, 444), while the fraction for A-type stars would be around 50 % (Duchene

& Kraus 2013,ARA&A,51,269) and for F to G type stars this fraction would also be around 55 % (Raghavan et al.,2010,ApJS,190,1). However, the situation for B stars is not so clear and the estimates differ a lot from a study to another. For instance Abt et al. (1990,ApJS,74,551) found about 30 % of spectroscopic binaries in their sample with potentially 40 % more of visual binaries (without certainty whether they are bound or not). Raboud (1996,A&A,315,384) evaluated it in the NGC6231 cluster to be about 52 %, Martayan et al. (2008,A&A,489,459) found a binary fraction of 27 % in NGC6611, while Oudmajer et al. (2010,MNRAS,405,2439) estimated the B binary fraction to range from 22 to 38 % with NACO adaptive optics images. Chini et al. (2012,MNRAS,424,1925) found that the binary fraction varies with the stellar mass and ranges from 80 % for O stars to 60-70 % for early-B stars and 20-40 % for late B stars. Indeed Duchene & Kraus (2013,ARA&A,51,269) indicate a B binary fraction ranging from 40 to 60 %. Recently Nasseri et al. (2013,CEAB,37,51) with a spectroscopic monitoring found a similar B binary fraction ranging from 50% in early types to 15% in late types with orbital periods from 1 to 1000 days. Such a discrepancy between the fraction of O binaries (70-80 %) and B binaries (70-20 %), while for A binaries it is 50 %, is difficult to believe. It would indicate a different star formation mechanism for creating B stars while it is still critical to reduce the angular momentum during the star formation. The obvious possibility to decrease it is to form binaries with stellar or planetary companions.

CHAPTER

TWO

OBSERVATIONAL DATA

2.1 The sample

The sample is based on the observatory project 093.D-0415(A,B,C) PI: Martayan.

1. About 200 B stars were observed in this project.
2. About 2500 science spectra were obtained.
3. 65% complete in term of OBs and time (strongly vary with the run)
   (a) run A: 162 of 190 OBs, 85.3%, 29.1 of 34h, 85.5%
   (b) run B: 116 of 204 OBs, 56.9%, 19.4 of 34h, 57.0%
   (c) run C: 58 of 126 OBs, 46.0%, 9.7 of 21h, 46.2%

In addition to the observations, most of the stars have 1 to 3 spectra in the X-Shooter archive:

- Run A, at least 1 spectrum in the archive, and for run B and run C, generally at least 2 spectra;
- In total each object has about 4 to 6 different epochs of observation;
- The objects were observed under photometric transparency to thick clouds, seeing of 0.6" to 4", Full Moon or no Moon;
- Some of the objects are known to host a binary companion or an exoplanet candidate. They will be used as test bench to probe the radial velocity accuracy and stability of X-Shooter on real data.

2.2 The instrument

X-Shooter is a single target spectrograph for the Cassegrain focus of one of the VLT UTs (Vernet et al. 2011, A&A, 536A, 105), convering, in a single exposure, the spectral range from the UV to the K' band (300-2500 nm) at intermediate spectral resolution (R ~ 4000-17000, depending on wavelength and slit width) with fixed echelle spectral format. The instrument consists of four arms with two cameras, namely: Aquisition and Guinding. The 3 spectroscopic arms cover the ranges:

1. UVB - 300-559.5 nm
2. VIS - 559.5-1024 nm
3. NIR - 1024-2480 nm

CHAPTER

THREE

TOOLS

In this chapter, we describe the tool used to do the data reduction, and the three python routines developed to study the reduced data: Py-XShooter, PharaohUS, and calc_vsini.

3.1 Reflex

Reflex is the ESO Recipe Flexible Execution Workbench recommended in the reduction of ESO data. This recipe offers some facilities to the user, such as a global graphical vision of the data reduction cascade (workflow), for more details we suggest the reading of Freudling et al. 2013. Briefly, this recipe automatically organizes input files according to their category before run the reduction chain. The user has the option to inspect and interact in each reduction step, having the possibility of rerun any step.

3.1.1 Installation

In order to install the *eso-reflex*, we suggest the procedure described in http://www.eso.org/sci/software/pipelines/reflex_workflows/index.html. After that, it is important to define the amount of memory that will be reserved to the Reflex, it may be done by the command line */bin/esoreflex_set_memory* (p.e. 1024; 2018m).

3.1.2 Reduction

Reflex X-shooter Tutorial can be acessed in ftp://ftp.eso.org/pub/dfs/pipelines/xshooter/. The raw data was reduced following the procedure described in the manual **xshoo-pipeline-manual-12.7.pdf**. The results from this recipe are generally very robust with the default parameters, then, after some attempts to improve the residuals, we have decided to adopt the default parameters in the workflow. Basically, as simple procedure we must access the esoreflex and define the folders where are the raw data and where the results will be saved. After, the data organisation process is done automatically by the recipe. A simple reduction procedure is,

1. Uncompress the data before executing Reflex;
2. $esoreflex &
3. open the X-shooter workflow by clicking on File -> Open File, selecting the file xshoo-2.6.8/xshooter.xml;
4. Tools -> Animate at Runtime, enter the number of milliseconds representing the animation interval (1 ms is recommended);
5. Under “Setup Directories” in the workflow canvas there are seven parameters that specify important directories (green dots). When you have finished, click OK to save your changes;
6. Play start button;
7. Follow the workflow.

The final products that are copied and renamed are:

1. *<HIERARCH.ESO.OBS.NAME>_SCI_SLIT_FLUX_MERGE1D_<ARM>.fits* - The flux-calibrated, extracted and merged science spectrum. This product is only generated if an appropriate instrument response curve (master or otherwise) was used as an input to the Science Reduction actor.
2. *<HIERARCH.ESO.OBS.NAME>_SCI_SLIT_MERGE1D_<ARM>.fits* - The extracted and merged science spectrum.
3. *<HIERARCH.ESO.OBS.NAME>_SCI_SLIT_MERGE2D_<ARM>.fits* - The merged 2-dimensional science spectrum.
4. *<HIERARCH.ESO.OBS.NAME>_SKY_SLIT_MERGE1D_<ARM>.fits* - The merged 2-dimensional sky spectrum (not for nodding mode data).
5. *<HIERARCH.ESO.OBS.NAME>_SCI_SLIT_IDP_<ARM>.fits* - The corresponding final 1D extracted spectra in a format compatible with the Science Data Product Standard needed to submit files to ESO's Science Archive Facility. The standard itself is described in http://www.eso.org/sci/observing/phase3/p3sdpstd.pdf and tips on how to handle the data with common tools are here http://archive.eso.org/cms/eso-data/help/1dspectra.html.

3.2 PY-XShooter

PY-XShooter is an python routine developed to generate, in an easy way, the desired outputs by the user. This routine depends on several python packages: numpy, matplotlib, astropy, pyfits, gzip, re, scipy, math, astropy, csv, astroquery, aplpy, lineid_plot, pyhdust, pyraf, ureka.

3.2.1 Running

To run the routine is required to define the folders where are the input files (raw data to create the images, tag **input_data_raw**, and the end products data to create the science figures, tag **input_data**). Having defined these folders, the user must define the path where the routine is installed in the tag **commum_folder**.

Having defined properly all folders and options, the code can be runned in terminal by typing

```
ur_setup
python xshooter.py or ipython xshooter.py
```

The routine creates a folder with the following structure for each star observed:

1. Star name
   (a) lines
   (b) sed
   (c) images
   (d) bcd
   (e) vsini

Important: This structure is preserved when is started the second step, i.e. when the routine PharaohUS begins to calculate the stellar parameters (see details in the next section).

Inside the folder **lines**, all given lines (defined in lists for each xshooter arm) are saved separatelly, i.e. one by folder. Each of these folders contains three files: two for the line profile (flux vs wavelength and flux vs velocity) and one showing the temporal evolution of the line equivalent width.

The second folder, **sed**, two files, one with all sed's plotted in only one plot area, and another with the fluxes separeted by arm.

The folder **images** has the aquisition images taken from the raw data. These figures can be used in order to check for potential visual companion.

The folder **bcd** has the results of the BCD analysis (which is not impacted by the circumstellar environment). In this folder, we have a text table with the inferred fundamental stellar parameters (Teff, logg, etc), a figure illustrating BCD procedure, and four figures that show the resulting interpolation for each stellar parameter.

The last folder, **vsini**, contains the results for the independently calculated vsini for each X-Shooter arm.

3.2.2 Some Options

There are five variables that can be modified:

1. interact = ‘yes’ or ‘no’
2. plot_images = True or False
3. plot_sed_lines = True or False
4. delta = 50. #Angstrom

Trought the option *interact*, the user could visualize the normalization by interacting with an iraf window. Then, it is possible to check the quality of the normalization, as well as change some standard adopted parameters.

The option *plot_images* is used for the cases in which the user is not interested in save the aquisition images. If this option is True, the images will be saved automatically in the same folder from that of the science figures.

The option *plot_sed_lines* should be equal True, if you want to generate the science figures (line profiles, SEDs, the BCD analysis results, and inferred stellar parameters).

Delta is the range around the line that must be considered during the simulation. This value must be choosen correctly, otherwise it is possible to take another line during the simulation, instead of the correct one.

Following are the description of the functions in this routine:

`py_xshooter.`**`bcd_analysis`**(*obj*, *wave*, *flux*, *flux_norm*, *folder_fig*)

This function performs the BCD analysis.

Parameters

- **`obj`** – star name (string)
- **`wave`** – Array with the wavelenghts in Angstrom (numpy array)
- **`flux`** – Array with the fluxes (numpy array)
- **`flux_norm`** – Array with the normalized fluxes (numpy array)
- **`folder_fig`** – folder where the figures will be saved (string)

Star_name star name (string)

Returns D and lambda_0 (BCD parameters)

`py_xshooter.`**`calc_dlamb`**(*wave*, *flux*, *center_wave*, *delta*)

This function calculates the shift of the line.

Parameters

- **`wave`** – Observed wavelenght of the line (float, ex: 6562.81 #AA)
- **`flux`** – Array with the fluxes (numpy array)
- **`center_wave`** – Lab central wave (float)
- **`delta`** – range to be considered around the lamb_obs (float)

Returns delta lambda (float)

`py_xshooter.`**`create_class_object`**(*list_files*, *obj*)

Create object class for all targets in a list.

Parameters **`list_files`** – list of XShooter fits files (array)

Returns data (class object)

`py_xshooter.`**`create_images`**(*list_files*, *folder_fig*)

Plot the images for each observed star from the aquisition images.

Parameters

- **`list_files`** – list of XShooter fits files (array)
- **`folder_fig`** – folder where the figures will be saved (string)

Returns images

`py_xshooter.`**`create_list_files`**(*list_name*, *folder*, *folder_table*)

Creates a list of the files inside a given folder.

Parameters

- **`list_name`** – list's name (string)
- **`folder`** – files' folder (string)

Returns creates a txt file, with the files' paths

`py_xshooter.`**`create_list_stars`**(*list_files*)

Create list of the observed stars.

Parameters **`list_files`** – text list with the files' paths

Returns list of stars (array)

`py_xshooter.`**`create_list_stars_img`**(*list_files*)

Create list of the observed stars.

Parameters **`list_files`** – text list with the files' paths

Returns list of stars (array)

`py_xshooter.`**`create_txt_file`**(*x*, *y*, *file_name*)

Create a txt file.

Parameters

- **`x`** – array with n elements (array)
- **`y`** – array with n elements (array)
- **`file_name`** – file's name (string)

Returns txt file

class `py_xshooter.`**`data_object`**(*name*, *arm*, *mjd*, *list_files*, *name_ffit*, *wave=None*, *flux=None*, *sigm=None*, *wave_vis=None*, *flux_vis=None*, *sigm_vis=None*, *wave_nir=None*, *flux_nir=None*, *sigm_nir=None*)

Class of the stellar objects. Using this class, we can store for each star a sort of variables, which can be easily accessed.

`py_xshooter.`**`find_nearest`**(*array*, *value*)

Find the nearest value inside an array.

Parameters

- **`array`** – array
- **`value`** – desired value (float)

Returns nearest value and its index (float)

`py_xshooter.`**`fluxXvel`**(*wave*, *flux*, *flux_plus_i*, *central_wave*, *delta*, *label*, *ax*, *fit=None*)

Function to plot a line profile.

Parameters

- **`wave`** – Observed wavelenght of the line (numpy array)
- **`flux`** – Array with the fluxes (numpy array)
- **`flux_plus_i`** – Array with the fluxes add by a constant value (numpy array)
- **`delta`** – range to be considered around the lamb_obs (float)
- **`central_wave`** – Lab central wave (float)
- **`label`** – label to be plotted in the legend
- **`ax`** – subfigure name (ax, ay or az)
- **`fit`** – Do nothing (boolean)

Returns velocity and associated flux (arrays)

`py_xshooter.`**`ger_tbdata`**(*file_name*)

Creates tbdata from hdulist and check if there is fields.

Parameters **`file_name`** – fits file (string)

Returns tbdata

`py_xshooter.`**`identify_object`**(*file_name*)

Function to identify the object.

Parameters **`file_name`** – fits file (string)

Returns object name (string)

`py_xshooter.`**`normalize_spectrum`**(*wave*, *flux*, *input_file*, *output_folder*, *arm*)

Function of normalization.

Source: http://stsdas.stsci.edu/cgi-bin/gethelp.cgi?continuum :param wave: Array with the wavelenght (numpy array) :param flux: Array with the fluxes (numpy array) :param input_file: fits file (string) :param output_folder: folder where it will be saved the output (string) :return: normalized fits file and normalized flux (output_file.fits)

`py_xshooter.`**`plot_arm_lines`**(*data*, *folder_fig*, *folder_temp*, *delta*, *star_name*)

Plot the lines for each star inside the data structure.

Parameters

- **`data`** – list of XShooter fits files (array)
- **`folder_fig`** – folder where the figures will be saved (string)
- **`delta`** – plot parameter (float)
- **`folder_temp`** – folder of the temporary files (string)

Star_name star name (string)

Returns figures and tables

`py_xshooter.`**`plot_fits_image`**(*folder_fig*, *fits_file*, *obj*, *mjd*, *scale=None*, *zoom=None*)

Read a simple txt file.

Parameters

- **`folder_fig`** – folder where it will be saved the figures (string)
- **`fits_file`** – name of fits file (string)
- **`obj`** – object name (string)
- **`mjd`** – MJD (float)
- **`scale`** – linear, log, and other options (string)
- **`zoom`** – Plot a zoom of the image? (boolean)

Returns Image

`py_xshooter.`**`plot_line`**(*wave*, *flux*, *center_wave*, *delta*, *vel*, *label*, *ax*, *calc_central_wave*, *save_fig=None*, *fig_name=None*, *gauss_fit=None*)

Function to plot a line profile.

Parameters

- **`wave`** – Observed wavelenght of the line (float, ex: 6562.81 #AA)
- **`flux`** – Array with the fluxes (numpy array)
- **`center_wave`** – Lab central wave (float)
- **`delta`** – range to be considered around the lamb_obs (float)
- **`vel`** – parameter for the function fit_line (float)
- **`label`** – label to be plotted in the legend
- **`ax`** – subfigure name (ax, ay or az)
- **`calc_central_wave`** – Do calculate the central wave? (boolean)
- **`save_fig`** – Save the figure? (boolean)
- **`gauss_fit`** – Would you like to perform a gaussian fit? (boolean)

Returns figures

`py_xshooter.`**`plot_sed`**(*data*, *folder_fig*, *folder_temp*, *star_name*)

Plot the SED for each star inside the data structure.

Parameters

- **`data`** – list of XShooter fits files (array)
- **`folder_fig`** – folder where the figures will be saved (string)
- **`folder_temp`** – folder of the temporary files (string)

Star_name star name (string)

Returns data (class object)

`py_xshooter.`**`print_keys`**(*file_name*)

Simple function to print the keys header of a fits file.

Parameters **`file_name`** – fits file (string)

Returns keys

`py_xshooter.`**`read_fits_star_name`**(*file_name*, *typ*)

Read XShooter's fits files.

Parameters

- **`file_name`** – name of the file in fit extension (string)
- **`typ`** – define if it will be used to read a image (typ='img')or a

bintable (typ='data'). :return: parameters (object if typ='data', file_name if typ='img')

`py_xshooter.`**`read_fits_xshooter`**(*file_name*, *print_obj*)

Read XShooter's fits files.

Parameters **`file_name`** – name of the file in fit extension (string)

Returns parameters (obj, obs_date, mjd, arm, wave, flux,

sigma, qual, snr, flux_red, sigm_red)

`py_xshooter.`**`read_header`**(*file_name*)

Simple function to read the header of a fits file.

Parameters **`file_name`** – fits file (string)

Returns header

`py_xshooter.`**`read_header_aquisition`**(*file_name*)

Read header of the aquisition images.

Parameters **`file_name`** – name of the fits file (string)

Returns object name (string), MJD (float), xshooter arm (string)

`py_xshooter.`**`read_header_simple`**(*file_name*)

Read header of a simple fits file.

Parameters **`file_name`** – name of the fits file (string)

Returns header_1 (string), header_2 (string)

Note: Somethimes, some fits file exhibit two headers.

`py_xshooter.`**`read_list_files_all`**(*table_name*, *folder_table*)

Read list of files in a table, and returns all fits file in an array.

Parameters

- **`folder_table`** – table's folder (string)
- **`table_name`** – Table's name (string)

Returns list of files (txt file)

`py_xshooter.`**`read_list_files_star`**(*table_name*)

Read list of files listed in a table, and returns the star's fits names in an array.

Parameters **`table_name`** – Table's name (string)

Returns list of files (txt file)

`py_xshooter.`**`read_norm_fits`**(*file_name*)

Read XShooter's normalized fits files.

Parameters **`file_name`** – name of the fits file (string)

Returns normalized flux (array)

`py_xshooter.`**`read_txt`**(*table_name*, *ncols*)

Read a simple txt file.

Parameters

- **`table_name`** – name of the table
- **`ncols`** – number of columns

Returns x, y (arrays) or x, y, z (arrays)

`py_xshooter.`**`read_zipfile`**(*folder*, *zip_file*)
Read the content of a zip file.

Parameters

- **`folder`** – folder with the files (string)
- **`zip_file`** – name of the file

Returns file content

`py_xshooter.`**`remove_negs`**(*num_list*)
Removes the negative values from a list.

Parameters **`num_list`** – array of values (array)

Returns keys

`py_xshooter.`**`smooth_spectrum`**(*wave*, *flux*, *doplot=None*)
Smooth the spectrum by convolving with a (normalized) Hann window of 400 points.

Parameters

- **`wave`** – Array with the wavelenght (numpy array)
- **`flux`** – Array with the fluxes (numpy array)
- **`doplot`** – Would you like to see the plot? (boolean)

Returns smoothed flux (array)

`py_xshooter.`**`unzip_file`**(*folder*, *folder_table*, *zip_file=None*, *list=None*)
This routine unzip a file or a list of files in a given folder.

Parameters

- **`folder`** – file's folder (string)
- **`zip_file`** – file's names (string)
- **`list`** – to unzipped more than one file put list=True, otherwise False

Returns unzipped file

Example:

folder = '/run/media/sysadmin/SAMSUNG/reduzindo/runC/' folder_table = '/run/media/sysadmin/SAMSUNG/reduzindo/' unzip_file(folder, folder_table, list=True)

3.3 PharaohUS

PharaohUS is an python routine that, given the BCD parameters, uses a Scipy interpolation method (*griddata*) to interpolate the stellar parameters from a stellar grid models. The importance of this interpolator is that it is capable of Interpolate unstructured D-dimensional data.

3.3.1 Some Options

The user can define these parameters:

1. resolution = 1000. # resolution of the color maps
2. read_files = True # If you want to read the files
3. interpol = True # Do you want to execute the interpolation?
4. plot3d = False # Plot 3D maps
5. plot_labels = False # Plot values alongside the points
6. cmap = 'plasma' # color map (many options)
7. interpol_method # 'linear', 'nearest', 'cubic'
8. list_models # Define file with the models data: '1.0_all' #'0.5_all'

3.3.2 Running

To run the routine is necessary to define the folders where are the input files (data tables, tag **folder_table**, and the folder where the data will be saved, tag **folder_results**). Having defined these folders, the user must define the path where the routine is installed in the tag **commum_folder**.

Having defined properly all folders and options, the code can be run in terminal by:

```
ur_forget
python pharaohus.py or ipython pharaohus.py
```

Following are the description of the functions in this routine:

pharaohus.**create_class_object**(*list_files*)
Create object class for all targets in a list.

Parameters **list_files** – list of XShooter fits files (array)

Returns data (class object)

pharaohus.**create_list_files**(*list_name*, *folder*, *folder_table*)
Creates a list of the names inside a given folder.

Parameters

- **list_name** – list's name (string)
- **folder** – files' folder

Returns creates a txt file, with the files' paths

pharaohus.**create_txt_file**(*x*, *y*, *w*, *z*, *h*, *q*, *file_name*)
Create a txt file.

Parameters

- **x** – array with n elements (array)
- **y** – array with n elements (array)
- **file_name** – file's name (string)

Returns txt file

class `pharaohus.`**`data_object`**(*name*, *lamb0*, *D*)
Class of the stellar objects. Using this class, we can store for each star a sort of variables, which can be easily accessed.

`pharaohus.`**`find_nearest`**(*array*, *value*)
Find the nearest value inside an array.

Parameters

- **`array`** – array
- **`value`** – desired value (float)

Returns nearest value and its index (float)

`pharaohus.`**`read_list_files_all`**(*table_name*, *folder_table*)
Read list of files in a table, and returns all fits file in an array.

Parameters

- **`table_name`** – Table's name (string)
- **`reflex`** – If you want to unzip the files to run reflex

Returns list of files (txt file)

`pharaohus.`**`read_txt`**(*table_name*)
Read a simple txt file.

Parameters

- **`table_name`** – name of the table
- **`ncols`** – number of columns

Returns x, y (arrays) or x, y, z (arrays)

`pharaohus.`**`read_txt_2`**(*table_name*, *ncols*)
Read a simple txt file.

Parameters

- **`table_name`** – name of the table
- **`ncols`** – number of columns

Returns x, y (arrays)

`pharaohus.`**`rotate`**(*matrix*, *degree*)
Function to rotate a matrix.

Parameters

- **`matrix`** – matrix (array)
- **`degree`** – rotation (float)

Returns matrix (array)

3.4 calc_vsini

calc_vsini is an python routine that calculates vsini values for a list of stars. The method uses the rotational broadening pyasl.rotBroad function and the Grids of model atmospheres - Kurucz. The routine reads the effective temperature and logg inferred by the BCD method applied by the Pharaohus routine. Its results are save in the folder *star_name/vsini*.

3.4.1 Some Options

The user can define these parameters:

1. delta = 200 # Range of points to be considered around the central wave;
2. plotchi2_map = True # Plotting the χ^2_{red} map;
3. center_wave_vis = 6562.8 # A choosen line in visible arm ;
4. center_wave_uvb = 4861. # A choosen line in uvb arm;
5. center_wave_nir = 21654. # A choosen line in nir arm;
6. vsini_arr = np.arange(1., 600., 10.) # Range of vsini's to be analysed (unit: km/s);
7. limbdark = np.arange(0,1.1,0.2) # Range of limbdarks to be analysed (dimensionless).

Note: The center waves must be given in Angstrom.

3.4.2 Running

To run the routine it is necessary to define the folders where are the input files (data tables, tag **folder_table**, and the folder where the data will be saved, tag **folder_results**). Having defined these folders, the user must define the path where the routine is installed in the tag **commum_folder**.

Having defined properly all folders and options, the code can be run in terminal by:

```
ur_forget
python calc_vsini.py or ipython calc_vsini.py
```

Following are the description of the functions in this routine:

`calc_vsini.`**`create_class_object`**(*list_files*, *obj*)
Create object class for all targets in a list.

Parameters **`list_files`** – list of XShooter fits files (array)

Returns data (class object)

`calc_vsini.`**`create_list_files`**(*list_name*, *folder*, *folder_table*)
Creates a list of the files inside a given folder.

Parameters

- **`list_name`** – list's name (string)
- **`folder`** – files' folder (string)

Returns creates a txt file, with the files' paths

`calc_vsini.`**`create_list_stars`**(*list_files*)
Create list of the observed stars.

Parameters **`list_files`** – text list with the files' paths

Returns list of stars (array)

`calc_vsini.`**`cut_spec`**(*wave*, *flux*, *err*, *center_wave*, *delta*)
Cut a spectra to a given range.

Parameters

- **`wave`** – array with the wavelengths (array)
- **`flux`** – array with the wavelengths (array)

- **center_wave** – central wavelength (float)
- **delta** – interval to be considered around the central wavelength (float)

Returns cut arrays (x, y, z)

class calc_vsini.**data_object**(*name*, *arm*, *mjd*, *list_files*, *name_ffit*, *wave=None*, *flux=None*, *sigm=None*, *wave_vis=None*, *flux_vis=None*, *sigm_vis=None*, *wave_nir=None*, *flux_nir=None*, *sigm_nir=None*)

Class of the stellar objects. Using this class, we can store for each star a sort of variables, which can be easily accessed.

calc_vsini.**find_nearest**(*array*, *value*)

Find the nearest value inside an array.

Parameters

- **array** – array
- **value** – desired value (float)

Returns nearest value and its index

calc_vsini.**plot_vsini**(*data*, *vsini_arr*, *limbdark*, *folder_fig*, *folder_temp*, *folder_table*, *star_name*, *center_wave_uvb*, *center_wave_vis*, *center_wave_nir*, *delta*, *plotchi2_map*, *lbdc*, *ltem*, *llog*, *lambd*, *prof*)

Plot the SED for each star inside the data structure.

Parameters

- **data** – list of XShooter fits files (array)
- **folder_fig** – folder where the figures will be saved (string)
- **folder_temp** – folder of the temporary files (string)
- **limbdark** – between 0 (no) to 1 (maximum)
- **center_wave** – centre wavelength of the line (float)
- **vsini** – array of possible values of vsini (array)
- **mjd** – modified Julian date (float)
- **star_name** – star name (string)
- **lbdc** – list of wavelengths from the model (array)
- **ltem** – list of temperatures from the model (array)
- **llog** – list of logg from the model (array)
- **lambd** – wavelenghts from the model (array)
- **prof** – fluxes from the model (array)

Returns plots, best vsini, best limbdark, and reduced chi2 values

calc_vsini.**read_fits**(*file_name*)

Read XShooter's fits files.

Parameters **file_name** – name of the file in fit extension (string)

Returns parameters

calc_vsini.**read_fits_star_name**(*file_name*, *typ*)

Read XShooter's fits files.

Parameters

- **`file_name`** – name of the file in fit extension (string)
- **`typ`** – define if it will be used to read a image

(typ='img')or a bintable (typ='data'). :return: parameters (object if typ='data', file_name if typ='img')

`calc_vsini.`**`read_fits_xshooter`**(*file_name*, *print_obj*)

Read XShooter's fits files.

Parameters **`file_name`** – name of the file in fit extension (string)

Returns parameters

`calc_vsini.`**`read_list_files_all`**(*table_name*, *folder_table*)

Read list of files in a table, and returns all fits file in an array.

Parameters

- **`folder_table`** – table's folder (string)
- **`table_name`** – Table's name (string)

Returns list of files (txt file)

`calc_vsini.`**`read_params_star`**(*table_name*)

Read the output of the bcd method (txt file).

Parameters **`table_name`** – name of the table (string)

Returns stellar parameters

`calc_vsini.`**`read_txt`**(*table_name*, *ncols*)

Read a simple txt file.

Parameters

- **`table_name`** – name of the table
- **`ncols`** – number of columns

Returns x, y (arrays) or x, y, z (arrays)

`calc_vsini.`**`smooth_spectrum`**(*wave*, *flux*, *doplot=None*)

Smooth the spectrum by convolving with a (normalized) Hann window of 400 points.

Parameters

- **`wave`** – Array with the wavelenght (numpy array)
- **`flux`** – Array with the fluxes (numpy array)
- **`doplot`** – Would you like to see the plot? (boolean)

Returns smoothed flux (array)

`calc_vsini.`**`vsini_calc_kurucz`**(*wvl*, *flux*, *teff*, *logg*, *center_wave*, *flux_err*, *limbdark*, *vsini*, *color*, *folder_fig*, *mjd*, *star_name*, *lbdc*, *ltem*, *llog*, *lambd*, *prof*, *arm*)

Function that calculate vsini using rotational broadening pyasl.rotBroad and the Kurucz Grids of model atmospheres.

Parameters

- **`limbdark`** – between 0 (no) to 1 (maximum)
- **`wvl`** – observational wavelength (array)
- **`flux`** – observational flux (array)
- **`flux_err`** – observational flux error (array)

- **teff** – effective temperature (float)
- **logg** – gravitational acceleration (float)
- **center_wave** – centre wavelength of the line (float)
- **vsini** – array of possible values of vsini (array)
- **color** – array of colors (array)
- **folder_fig** – folder where the figures will be save (string)
- **mjd** – modified Julian date (float)
- **star_name** – star name (string)
- **lbdc** – list of wavelengths from the model (array)
- **ltem** – list of temperatures from the model (array)
- **llog** – list of logg from the model (array)
- **lambd** – wavelenghts from the model (array)
- **prof** – fluxes from the model (array)
- **arm** – xshooter arm (string)

Returns plots, best vsini, best limbdark, and reduced chi2 values

3.5 Shell Script

There is a option to submit all routines simultaneously. In order to do that, we must turn the file *run_all_eso.sh* executable, and after run it:

```
chmod 777 run_all_eso.sh
./run_all_eso.sh
```

This procedure will start to running the routine PY-XShooter. When it is finished, the routine PharaohUS starts. At this point, all the structure created by PY-XShooter will be preserved and used by PharaohUS as well. Finally, the routine calc_vsini evaluates, for the same data, the vsini values by broadening a model atmospheric profile line (without rotation) until achieve the observational line profile.

CHAPTER

FOUR

PRELIMINARY RESULTS

The results, for each star, are saved in the folder */XShooter/results/star_name/*. Inside each folder *star_name*, it will have five other folders: *lines* (line profiles separed by folder), *SED* (all sed's), *bcd* (the BCD fit done in the balmer jump region), *images* (aquisition images), and *vsini* (vsini fits). Following, we present three examples of output, one for a single B star, one for a binary, and another output showing an example of vsini determination.

4.1 A Simple B star

As mentioned previously, it is created a figure with the smoothed spectra separeted by arm. The lines selected by the user appear in this plot, which helps in the visualization of interesting features. In the case of interest in a single line and about any other information about it, the user can visualize them in the folder *star_name/lines/line_name/*.

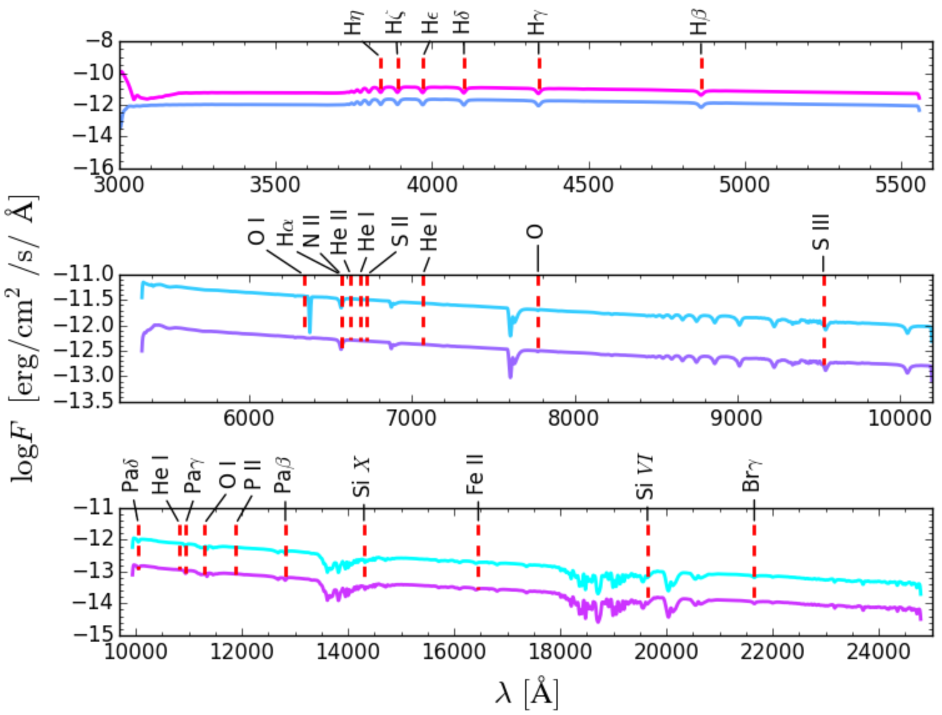

Fig. 4.1: Smoothed SED for all observations by arm for the program star Hip109155.

Beyond the line profiles, the procedure uses the region of the Balmer Jump to determine the input values of the BCD method (see J. Zorec et al. 2009).

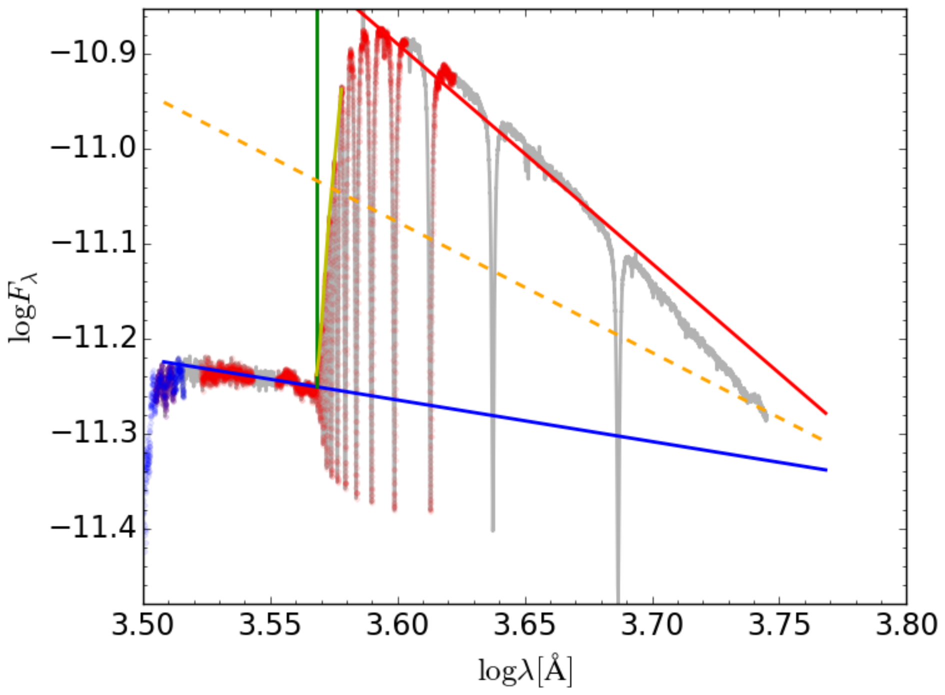

Fig. 4.2: BCD adjust for the program star Hip109155.

After the determination of the parameters (D, λ_0) needed to the BCD method, we execute an interpolation of the stellar parameters. At the end of this step, it generates a table with the inferred stellar parameters.

4.2 v sini

Figure 4.5 shows a example of the procedure adopted to found the vsini. The original line profile (blue curve) is compared with the broadened one, which depends on the values of vsini and limbdark. For each step of the interaction, these values are stored and the it performs a χ^2_{red} minimization (Fig. 4.6).

4.3 Binaries

For the study of binaries, one first step was looking at the raw frames, especially the acquisition images (Fig. 4.7) and some characteristic double-lined spectra (Fig. 4.8), reporting which objects are visible binary with a true or apparent companion from the acquisition images.

The determination of parameters of binary stars will need an implementation of another methods, because, in principle, it is needed a more precise study of the time series of measures. For this task, a more specific tool may be util as the Phoebe Project avaiable at http://www.phoebe-project.org/.

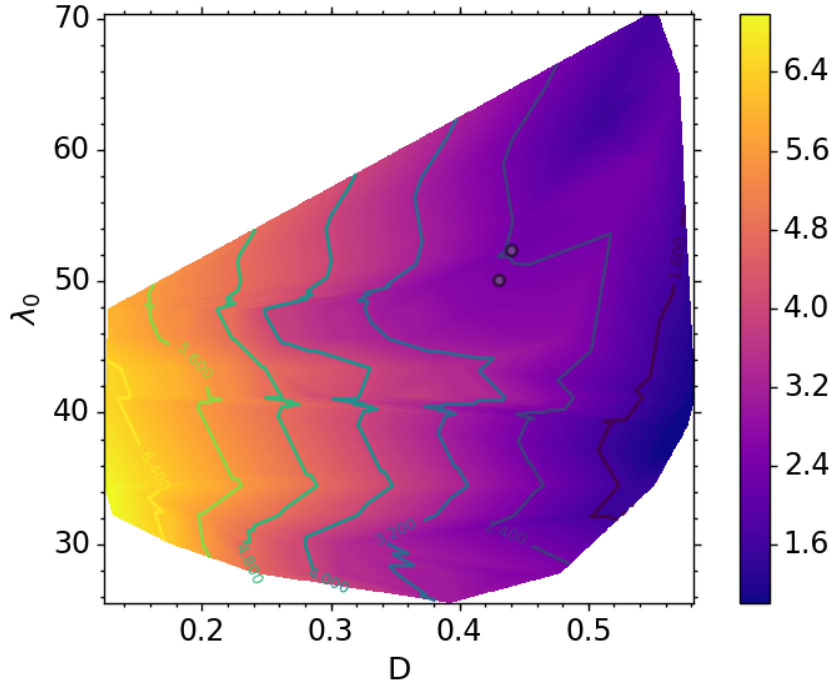

Fig. 4.3: Griddata interpolation of the stellar mass for the program star Hip076442.

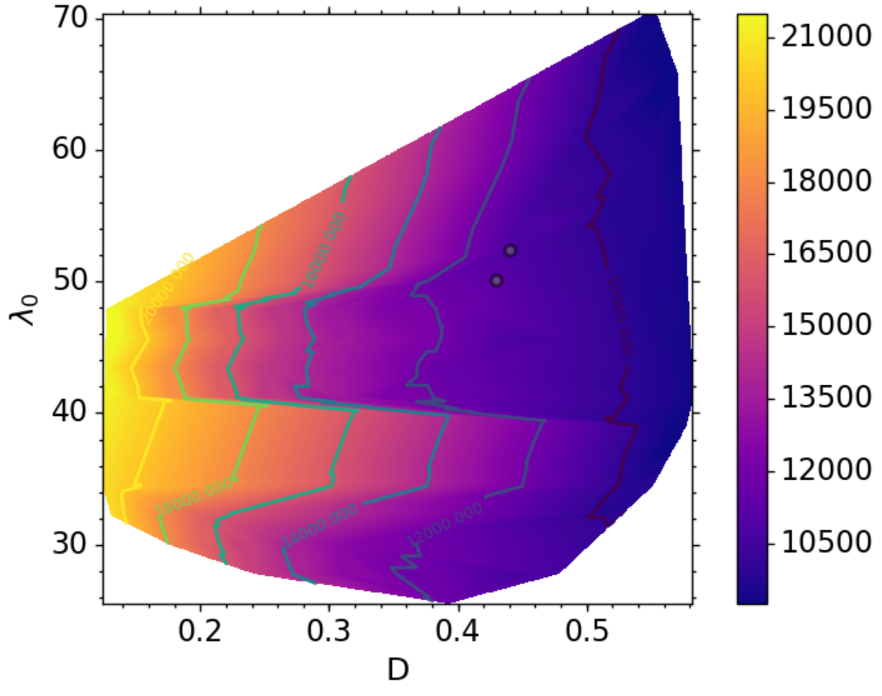

Fig. 4.4: Griddata interpolation of the T_{eff} for the program star Hip076442. The same procedure is done to interpolate the other stellar parameters.

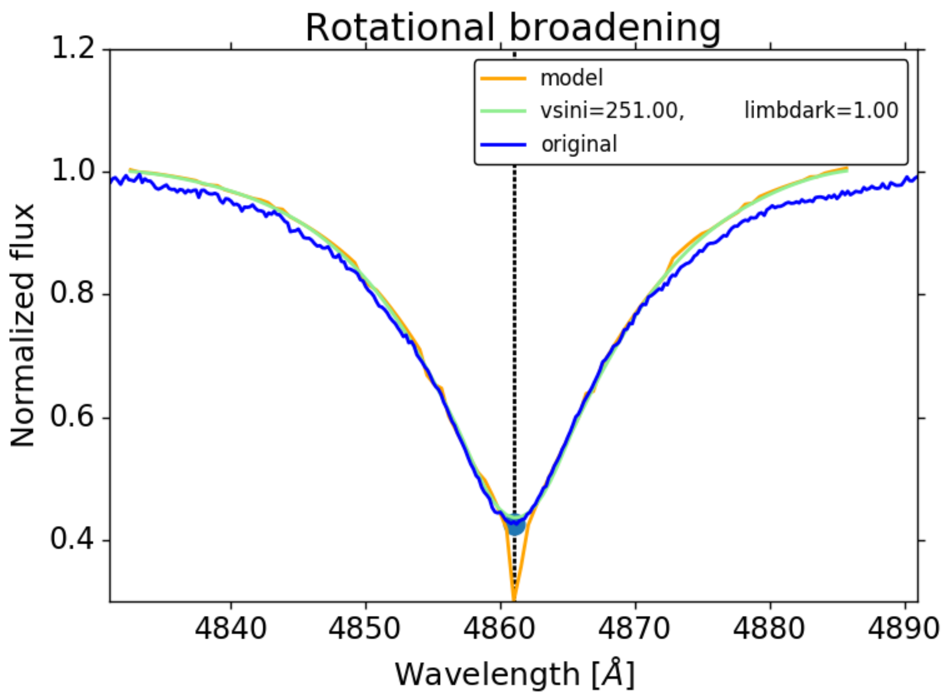

Fig. 4.5: Observational H:math:*alpha* line profile of the star Hip109155 (blue curve), original line profile from the models (orange curve), and best adjust (green curve).

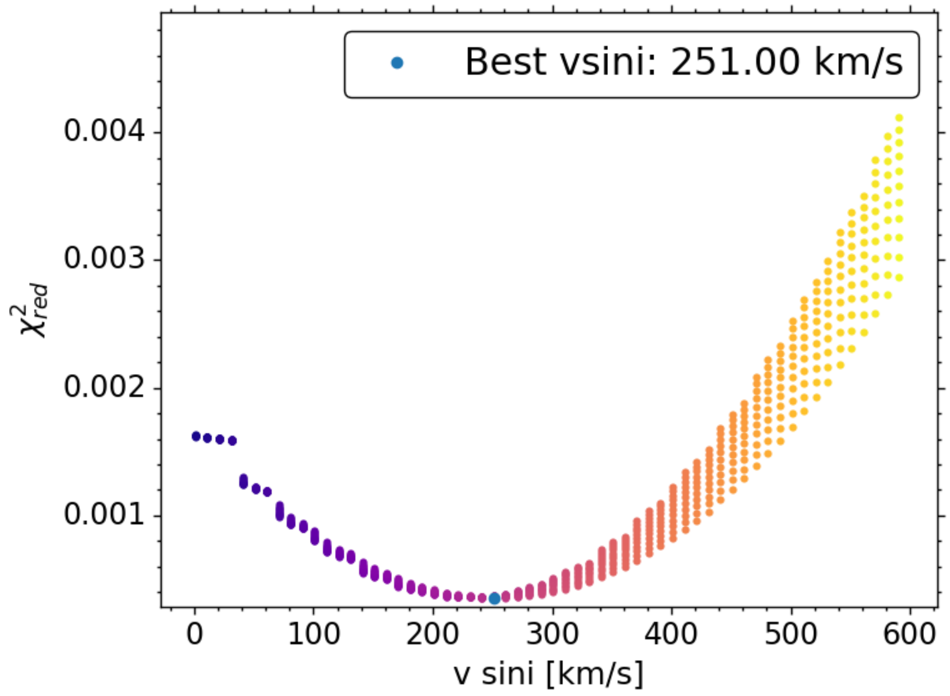

Fig. 4.6: Map of math:*chi*^2_*{red}* showing the best adjust.

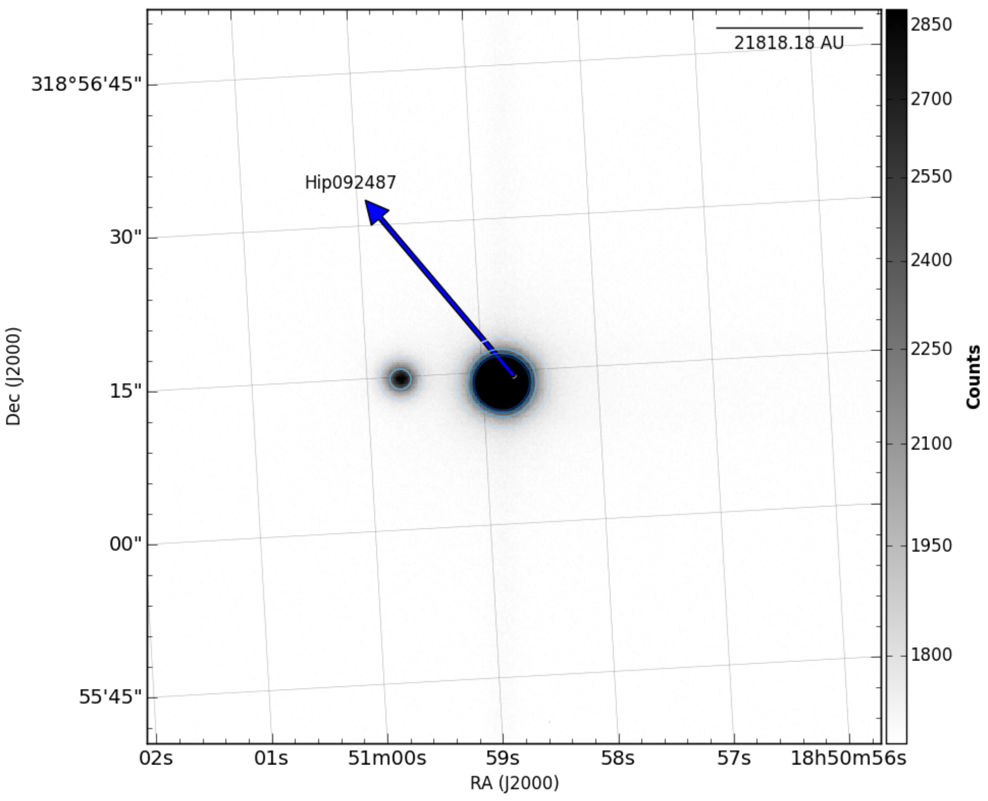

Fig. 4.7: Aquisition image of the program star Hip092487.

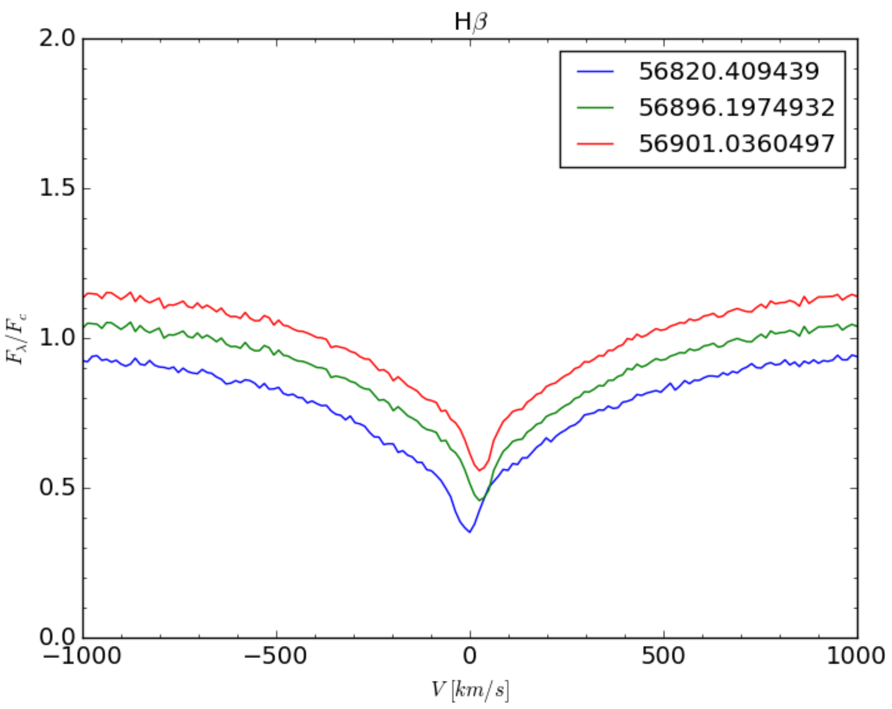

Fig. 4.8: H:math:*alpha* line profile of the program star Hip092487.

CHAPTER FIVE

INCOMPLETE TASKS

1. Separete the observations by slit;
2. Improve the normalization;
3. Implement heliocentric correction;
4. Determine parameters of binary systems,

CHAPTER

SIX

INDICES AND TABLES

- genindex
- modindex
- search

PYTHON MODULE INDEX