APPENDIX A

WEAK-LENSING SOFTWARE

A.1 General Information

WEAK-LENSING software is a set of different programs designed to work as a Weak Lensing analysis pipeline. It is placed on a Github repository at https://github.com/gcanasherrera/Weak-Lensing.git, that contains a set of programs and scripts required to detect clusters of galaxies provided an image of a sky region. The detection of these cluster of galaxies is performed thanks to the concept of gravitational lensing: light coming from distant objects suffers a deviation due to the mass present in the universe. When this effect is subtle (meaning weak), this deviation produces a shear in the size of distant objects susceptible to be measured.

WEAK-LENSING software is fully written in PYTHON 2.7 and it basically consists on scripts assembling the DLS PIPELINE. It works together with the programs contained in this pipeline written mainly by Professor David Wittman[1]. This package is written in C, FORTRAN and PERL, and it is divided into many tools to work with DLS catalogues and shear-measurement programs. DLS catalogues are in FIAT format designed by D. Wittman himself and whose tools are open to anybody at http://dls.physics.ucdavis.edu/fiat/fiat.html. WEAK-LENSING software also maintained to use the FIAT 1.0 format, taking advantage of some of these FIAT tools. On the other hand, DLS PIPELINE shear-measurement programs are not public. The availability of the DLS pipeline programs is recorded in Table A.1, and the pseudocode or main algorithm of the DLS program may be explained in the bibliographical sources provided during the Final Degree Project report as they were introduced in the text.

WEAK-LENSING software is registered under the GNU GENERAL PUBLIC LICENSE, and all rights and responsibilities can be inferred from it.

A.2 Author Information

Guadalupe Cañas Herrera is the only author of WEAK-LENSING software, and she is the responsible of new updates of version in the future. For any query regarding the source code, the user may request her via email at gch25@alumnos.unican.es or canasg@ifca.unican.es.

[1] Professor D. Wittman works currently at UC Davis. His personal web-page is http://wittman.physics.ucdavis.edu/contact.html. Most of the software he developed is not public as it belongs to the DLS Collaboration. Therefore, any modification of those codes performed by the author of this documentation will be explicitly remarked.

A.3 General Requisites

In order to run WEAK-LENSING software, it is necessary to have installed in the computer and/or virtual machine the following requisites. Furthermore, the use of a grid, such as any computational cluster or supercomputer, is highly recommended due to the great amount of data to process.

- **UNIX Operative System**: This is not an essential requisite, but the use of the software has not be tested in Windows. If you have Windows as a OP, you may try to work with ENTHOUGHT CANOPY available at https://www.enthought.com/products/canopy/.
- **PYTHON 2.7** and **IPYTHON**
- **Python Library Packages: ASTROPY, NUMPY, MATPLOTLIB, PYMODELFIT, SEABORN, SCIPY**. Most of them are included in ANACONDA package, a completely free Python Distribution containing most famous Python Packages available at http://docs.continuum.io/anaconda/install, or they are easily downloadable thanks to the CONDA command given by ANACONDA itself such as PYMODELFIT and SEABORN. I strongly recommend using ANACONDA to avoid further problems with hidden dependencies (for instance, SEABORN depends on the package PANDAS, but this one is directly included in ANACONDA). If you choose working with ANACONDA, be sure that you are calling the PYTHON compiler included in this package, and you are not using by default the PYTHON compiler included in most UNIX distributions. When ANACONDA is installed, it usually builds up automatically the PATH to ANACONDA location in order to simply start to use this PYTHON distribution and their compilers. However, it is useful to check that the PATH is corrected by typing in a terminal:

  ```
  which python
  ```

  You should see something like:

  ```
  /thepath/anaconda/bin/python
  ```

  In case you observe something different, it is probably because you are using the PYTHON compiler not placed in ANACONDA. Consequently, you should change yourself the PATH to ANACONDA into the .bashrc file,

  ```
  ls -a
  vim .bashrc
  ```

  and write down the path to python compiler in Anaconda,

  ```
  #Call Anaconda
  export PATH="/thepath/anaconda/bin/python:$PATH"
  ```

You can also work with the default PYTHON compiler and install the required packages yourself thanks to a Package Installer such as PIP (also available in ANACONDA or at https://pip.pypa.io/en/stable/) or HOMEBREW.

- **SOURCE EXTRACTOR** v.2.3.2 (older versions such as 2.19.5 may work although measurement results are not as precise as v. 2.3.2 and further modifications in the scripts may need to be carried out). Installation is complicated in OS X (it is suggested trying with HOMEBREW instead of re-compiling from the source code) and in Linux (if you lack of administrator permission to use apt command to re-compiling from scratch). Once it is installed, it is recommended to open a terminal and attach the the PATH to SOURCE EXTRACTOR executable into the .bashrc file,

```
ls -a
vim .bashrc
```

and write down the path to SOURCE EXTRACTOR executable,

```
#Call Source Extractor
export PATH="yourpath/sextractor/bin:$PATH"
```

This way, it is not needed to modify any program from *Weak-Lensing* software in order to make SOURCE EXTRACTOR work, as *Weak-Lensing* software scripts use sub-processes via terminal to call SOURCE EXTRACTOR. It also requires default files as an input to start to work. You can find those default files in SOURCE EXTRACTOR directory.

- **SAOIMAGE DS9** in order to visualize pictures automatically from some scripts. Installation is really simple, and it can be found at http://ds9.si.edu/site/Download.html. One it is installed, it is recommended to open a terminal and attach the the PATH to SAOIMAGE DS9 into the .bashrc file,

```
ls -a
vim .bashrc
```

and write down the path to SAOIMAGE DS9 executable,

```
#Call ds9
export PATH="yourpath/ds9:$PATH"
```

- **DLS PIPELINE SOFTWARE**: SEX2FIAT, FIATFILTER, ELLIPTO, DLSCOMBINE, FIATMAP, FIATREVIEW. Again, it is highly recommended to attach the the PATH to these executables into the .bashrc file. For instance,

```
ls -a
vim .bashrc
#Call Fiatmap
export PATH="yourpath/fiatmap:$PATH"
```

DLSS	Privacy	Author	Language	Dependencies	Description
SEX2FIAT	public	D. Wittman	PERL	fiat.c, fiat.h	Traduce Source Extractor Catalogue into a FIAT catalogue
FIATFILTER	public	D. Wittman	PERL	fiat.c, fiat.h	obtain only required data that fulfils a logical condition from the catalogue
DLSCOMBINE	private	D. Wittman/G. Cañas	C	fiat.c, fiat.h, dlscombine_utils.c, dlscombine.c, dlscombine.h	correct PSF anisotropies of the images
FIATMAP	private	D. Wittman/G. Cañas	C	fitsio.h, fiat.h	Generates fits image based on mass-maps thanks to invlens algorithm
ELLIPTO	private	D. Wittman	C	fiat.h	Corrects objects' shapes
FIATREVIEW	public	D. Wittman	C	PGPLOT, cfitsio, wcstools	Plots a FITS image and a FIAT catalogue at the same time

Table A.1 Description of the different DLS programmes used during the project, their characteristics and authors. The public fiat tools may be found at http://dls.physics.ucdavis.edu/fiat/fiat.html

I can provide with the binaries/executables of DLS PIPELINE used in this Final Degree Project for both OS X and Linux. To obtain the source code, I suggest contacting directly to Professor Wittman.

A.4 Source Installation

For installation, simply copy the Github repository and execute the main script via terminal. Open a new terminal tab and type:

```
git clone https://github.com/gcanasherrera/Weak-Lensing.git
cd Weak-Lensing/
```

To execute any script, I recommend using IPYTHON in interactive mode (so variables kept in memory at the end of the script and you can work with them). To execute the main script, you will need to pass to the script a FITS picture and a catalogue to start the analysis. To try, type:

```
ipython -i WL_Script.py
```

A.5 Content

The WEAK-LENSING software contains a set of several programs divided into four different types:

1. auxiliary programs destined to assist main scripts.
2. validation programs to check catalogues.
3. programmes destined to the study of systematics and classification of compact objects.
4. DM halo estimation programmes.

A.5.1 Auxiliary Programs

Read Catalogues

To read catalogues obtained from SOURCE EXTRACTOR it is essential to transform it into a FIAT format first, and then translate the information into NUMPY arrays. For that, I have defined the PYTHON class CLASS_CATALOGREADER.PY that contains general methods to read FIAT catalogues and transform them into structured NUMPY arrays. To work, it uses the following logical pseudocode to understand if the catalogue comes from SOURCE EXTRACTOR or it is already in FIAT format:

```
if catalogue_name ends with .cat (#Catalogue is in Source Extractor format)
call sex2fiat to transform
    then read
if catalogue_name ends with .fcat (#Catalogue is in FIAT format)
    then read
```

Therefore, the endings ".cat" is reserved for the catalogue given by SOURCE EXTRACTOR and ".fcat" to those in FIAT format.

To access the information of the catalogue we just create an object of this class and read the catalogue. The information is saved as an attribute of the object, for instance:

```
#Object from the CatalogReader class
from Class_CatalogReader import CatalogReader
c = CatalogReader('string_nameofcatalogue')
c.read()
information = c.fcat
x_coordinate = information['x']
```

General tools

Here we can find PYTHON files that complete main scripts.

First, the PYTHON file WL_UTILS.PY contains three different types of designed methods to:

1. call quickly executables such as SOURCE EXTRACTOR, ELLIPTO, DLSCOMBINE and SAOIMAGE DS9 through the Python Library SUBPROCESS.
2. short-cut the creation of ellipticity components plots and general $FWHM$ vs. m_{iso} plots.
3. create text files necessary for DLSCOMBINE to work (see details below).

Second, the PYTHON file WL_ELLIP_FITTER.PY works a a fitter of the ellipticity components as a function of Legengre polynomials.

A.5.2 Validation Programs: Cross-Matching and Add-Object Simulation

Cross-Matching Process

This software checks which objects are found in two different catalogues 1 and 2 regarding their coordinates (the numbers may refer to any band filter used in the work). It performs the following algorithm:

1. Transform objects' coordinates from catalogue 1 and 2 into spherical coordinates,

$$x = \sin\phi\cos\theta; y = \sin\phi\sin\theta; z = \cos\theta, \tag{A.1}$$

where ϕ is the known as the declination DEC and θ is the right ascension RA. It is necessary to use spherical coordinates defined from the declination and the right ascension due to the fact the fields for which the cross-matching are performed may not share the same Cartesian coordinates $(x', y'$, but they do regarding spherical coordinates (DEC and RA are the same for both). In order to avoid any extra transformation to one Cartesian coordinates system to another, we just simply use coordinates defined at Eq. (A.1).

2. Look for the objects from catalogue 2 in catalogue 1 (cross-matching code 1 to 2). The search in performed using the KD-Tree algorithm [1] with a confidence radius $r = kr_p$, being k an integer and r_p the size of the pixel in spherical coordinates. The objects of the catalogue 2 are referred as *keys*. Count the number of object that are not found in catalogue 1 as n_{lost}. Write a new catalogue (called catalogue 1 to 2) containing the objects found in catalogue 1 according to catalogue 2 keys.

3. Repeat step 3, in this case looking for the objects from catalogue 1 in the catalogue 2. Write a new catalogue (called catalogue 2 to 1) that contains the objects found in catalogue 2 according to catalogue 1 keys.

To carry out this task, the cross-matching process uses three different programs:

- CLASS_CROSSMATCHING.PY: PYTHON class. It contains methods to translate Cartesian coordinates to spherical ones, to construct the KD-Tree structure from the library SCIPY, and to write the final cross-matched catalogue thanks to a for-loop.

- WL_CROSSMATCHING.PY: PYTHON file. It is the main script to perform the cross-matching process between two catalogues. To produce the cross-matched catalogue, you should first read both catalogues using CLASS_CATALOGREADER.PY and then create an object from the class CLASS_CROSSMATCHING.PY. You have to choose the confidence radius r for the cross matching when calling the method KDTREE() and if you want to compare 1 to 2 or 2 to 1.

```
#Read catalogues catalogue_1 and catalogue_2 to obtain numpy st. array
#Create object from class CrossMatching
crossmatching = CrossMatching(catalogue_1, catalogue_2)
r=2
crossmatching.kdtree(n=r*1e-06)
#argument 'compare' may be set to '1to2' or '2to1'
crossmatching.catalog_writter('name of the array', compare = '1to2')
```

Add-Object Simulation

This simulation is designed to throw mocked galaxies, whose features are known, into a FITS file to test a detection tool such as SOURCE EXTRACTOR. The algorithm is as follows:

1. Read the catalogue obtained by SOURCE EXTRACTOR.

2. Calculate the mean values of the detected objects' magnitudes $\langle m \rangle$, ellipticities $\langle e \rangle$ and minor axis $\langle B \rangle$ directly from the data contained in the catalogue.

3. Calculate the mean value of the detected objects' eccentricity as,

$$\langle \varepsilon \rangle = \sqrt{1 - (1 - \langle e \rangle)^2} \tag{A.2}$$

4. Calculate the mean value of the detected objects' major axis $\langle A \rangle$ as,

$$\langle A \rangle = \frac{\langle B \rangle}{\sqrt{1 - \langle \varepsilon \rangle^2}} \tag{A.3}$$

5. Establish a relation between the *isophotal flux* F_{iso} and the *isophotal magnitude* m_{iso} data from the catalogue according to the general expression in [9],

$$F = a10^{-m/2.5+b} \tag{A.4}$$

where a and b are parameters to be calculate thanks to a non-linear Least Square Fitting that minimizes the sum of the squares of the residuals between the data and the model.

6. Read the FITS image in order to obtain a matrix whose size agrees with its from the picture. Each element of the matrix corresponds to a image pixel. We give matrix elements a value equals 1 if SOURCE EXTRACTOR detected a celestial objects in their matching pixels and give value equals 0 otherwise.

7. Calculate n pairs of random numbers for values of x and y belonging to the image size. We check if the matrix element noted by the position of those random pairs (x, y) is equals to 0. Save those pairs into a .txt file.

8. Create elliptical own-made galaxies for which their *maximum magnitude* m_{max} are fixed. Thanks to relation (A.4), calculate the corresponding *maximum flux* F_{max} according to the m_{max}. The distribution of the intensity along the ellipse is defined by elliptical Gaussian function expressed as,

$$f(x,y) = Fe^{-\frac{1}{2}\left[\frac{(x-x_0)^2}{\langle A \rangle^2} + \frac{(y-y_0)^2}{\langle B \rangle^2}\right]} \tag{A.5}$$

9. Launch n number of mocked galaxies into the n pairs of random numbers for values of x and y in the null elements of the matrix.

10. Fulfil the matrix elements still equals zero after launching the own-made galaxies with the corresponding values of the original image pixels. Copy that matrix into a FITS image.

11. Analyse with SOURCE EXTRACTOR the new FITS image containing the new n galaxies. Extract a new catalogue.

12. Look for the information corresponding to the new n mocked galaxies in the new catalogue. Compare the mean value of the own-made galaxies' *isophotal magnitude* attached by SOURCE EXTRACTOR regarding the value of the input magnitude that was given to the galaxies when were launched.

13. Count the number of own-made galaxies not found by SOURCE EXTRACTOR n_{lost}.

14. Repeat steps from 7 to 13 for own-made galaxies whose given magnitude goes from $m_{max} = 1$ until $m_{max} = 30$.

Note the difference between the *isophotal magnitude* m_{iso} and the maximum value of the magnitude, or *maximum magnitude* m_{max}. The maximum value of the magnitude corresponds to the magnitude value corresponding to the faintest pixel in the elliptical gaussian function, whereas the isophotal magnitude refers to the total value attached to the integrated magnitude for the object. Therefore, the maximum magnitude m_{max} will be always smaller than the *isophotal magnitude* m_{iso}, as the magnitude is the inverse of the intensity flux F so that $F_{iso} > F_{max}$ and consequently $m_{iso} < m_{max}$. Concretely, the relationship between F_{iso} and F_{max} is, $F_{iso} \approx 2\pi \langle A \rangle \langle B \rangle$, where $2\pi \langle A \rangle \langle B \rangle = 7.2$ and experimentally we obtain around 7. This relation is necessary because the introduced input data is the value of the maximum magnitude m^i_{max} we want to attach to mock galaxies.

To carry out the task explained in the algorithm, the simulation uses three different programs:

- CLASS_CATALOGREADER.PY.
- CLASS_OBJECTCREATOR.PY: PYTHON class. Contains general methods to fit magnitudes and fluxes (through the package PYMODELFIT), open and write FITS files (using the package ASTROPY), plots histograms of the properties A, B, e and ec, creates mocked galaxies and search tools based on three different tools:
  - SEARCHER($c_1.fcat$, $c_2.fcat$): loops-for the first catalogue and looks for each element of the first catalogue looping-for the second catalogue. It works but it is highly inefficient and slow.
  - SEARCHER_DIC($c_1.fcat$, $c_2.fcat$): looks for elements of the first catalogue in the second catalogue by means of python dictionaries. It offers an advantage with respect the first method meaning that it is quick and efficient, however it reduces (x, y) coordinates to integers values, missing significant digits.
  - SEARCHER_KDTREE($c_1.fcat$, $c_2.fcat$): looks for elements included in the first catalogue in relation to those in the second catalogue by the implementation of the KD-Tree algorithm available at the library SCIPY.
- WL_SIMULATION_EXECUTE_MAG.PY: PYTHON file. It is the main script containing instruction to develop steps described in the algorithm above. It saves the mean values of the quantities n_{lost}, m^i_{iso}, m^o_{iso}, F^i_{iso}, F^o_{iso}, F^i_{max}, F^o_{max} and their uncertainties associated in text files (upper index i and o mean input and output, respectively). The mean value is obtained from n mocked galaxies. The simulation is also useful to test the reliability of different smoothing filters such as *Mexican Hat* or *Gaussian*. To start the simulation pass as parameters the name of the picture in which you want to throw the mocked galaxies and the name of the filter SOURCE EXTRACTOR will use, typing:

```
ipython -i WL_Simulation_execute_mag.py PICTURE_NAME FILTER_NAME
```

- PLOTTER_OBJECTCREATOR.PY: PYTHON file. Script that reads the text files saved by the script explained above and plots the relations between m^i_{iso} and m^o_{iso}, F^i_{iso} and F^o_{iso}, F^i_{max} and F^o_{max}, and m^i_{iso} and n_{lost}.

This simulation needs to open and close at least 60 times pretty heavy FITS files, and run *Source Extractor* at least 30 times. For this reason, the use of a supercomputer is highly recommend. In the development of this work, and in order to send works to the tail of Altamira in an user-friendly way, I split the simulation and the plotting part (PLOTTER_OBJECTCREATOR.PY). Therefore, once that the features of the galaxies are saved in files by running just one time the simulation, it is possible to make any change on the plots just executing PLOTTER_OBJECTCREATOR.PY easily.

A.5.3 Study of Systematics and Classification of Compact Objects

The classification of compact objects into galaxies and stars, as well as the re-estimation of objects shapes and the correction of PSF anisotropies is performed by a general script named WL_SCRIPT.PY. This script performs the following pseudocode:

1. Read the number of pictures that perform the first cross-matching process and their corresponding observations bands (for instance R and Z). Then, reads the name of the two cross-matched catalogues and the name of the corresponding FITS files.
2. Read the first cross-matched catalogue and save the information it contains.
3. Plot Cartesian coordinates (x, y) corresponding to the compact objects in order to filter saturate edges. Perform the cut in F_{iso} to avoid saturated stars.
4. Plot the $FWHM$ vs. the magnitude m_{iso} of the compact objects without saturate detections.
5. Filter stars according to the study of the $FWHM$ vs. the magnitude m_{iso} and SOURCE EXTRACTOR neural network. Plots the ellipticity vector components.
6. Filter galaxies according to the study of the $FWHM$ vs. the magnitude m_{iso} and SOURCE EXTRACTOR neural network. Plots the ellipticity vector components.
7. Plot the histogram corresponding to the stars and galaxies population.
8. Recalculate galaxies and stars shapes using ELLIPTO. Filter those objects that have an error code higher than 2 (meaning that the shapes given by ELLIPTO differ considerably from those given by SOURCE EXTRACTOR).
9. Plot the ellipticity vector components of the new re-estimate compact object shapes.
10. Fit the ellipticity vector components (e_1, e_2) of the stars as a function of legendre polynomials $L(x)L(y)$. Write the fitting coefficients into a text file.

11. Write a text file suitable to be understood by DLSCOMBINE containing the path to the fitting coefficients text file and the name of the analysed image (denominated *specfile*).

12. Call DLSCOMBINE to correct PSF anisotropies from the analysed image.

13. Repeat the steps from 1 to 12 with a different image observed in another band and/or cross-matched catalogue of the same region of study.

14. Carry out cross-matching process between the two corrected images.

15. Repeats steps from 2 to 9 obtaining a final galaxies catalogue.

WL_SCRIPT.PY takes advantage of the software FIATFILTER to perform the several filtering of the catalogues. Furthermore, it is helped by two other programs:

- WL_UTILS.PY.
- WL_ELLIP_FITTER.PY: PYTHON file. Contains general methods to fit the ellipticity components (e_1, e_2) as a function of Legendre polynomials by a procedure of non-linear least square fitting minimizing the following quantity,

$$\chi_i^2 = \sum_x \sum_y \left(e_i(x,y) - \sum_n \sum_m c_{nm}^i(x,y) \frac{L_n(x) L_m(y)}{\sigma_e(x,y)} \right)^2, \tag{A.6}$$

  where $L_n(x)$ and $L_m(y)$ are Legendre polynomials, of order n and m, respectively, $c_{nm}(x,y)$ are the fitting coefficients and $\sigma_e(x,y)$ are the relative error associated to the ellipticity e by ELLIPTO used to weight the fitting. Due to the fact that DLSCOMBINE is designed to read polynomial fittings up to a maximum grade equals to 4, the value of n and m must be $n+m \leq 4$, so that $n, m = \{0,1,2,3,4\}$. The fitting is performed thanks to the package ASTROPY using a 2-Dimensional Legendre polynomial base as fitting model and the non-linear least square statistics provided by the package SCIPY following the Levenberg-Marquardt algorithm. Moreover, the fitting is retroactive, meaning that it rejects those values of the ellipticity components e_1 and e_2 that gives a residual higher than 3σ after the fitting, and then it carries out a second fitting without those objects. The rejection is performed by masking the e_1 and e_2 arrays with the package NUMPY to gain in speed. Finally, it writes a text files on the form,

```
# The fitting is in the form of Pn_m=Cn_m Ln(x)Lm(y) where
# the maximum degree is 4, meaning x(degree)*y(degree)<=4.
# Number of coefficients npar=15 for each e_i
# Coefficient Matrix for e_1
# c0_0     c0_1    c0_2     c0_3    c0_4
# c1_0     c1_1    c1_2     c1_3    0
# c2_0     c2_1    c2_2     0       0
# c3_0     c3_1    0        0       0
```

```
# c4_0     0       0        0       0
#
# Coefficient Matrix for e_2
# c0_0     c0_1    c0_2     c0_3    c0_4
# c1_0     c1_1    c1_2     c1_3    0
# c2_0     c2_1    c2_2     0       0
# c3_0     c3_1    0        0       0
# c4_0     0       0        0       0
#
```

DLSCOMBINE was designed to accept a fit based on simple polynomial bases such as $c^i_{nm}(x,y)x_n y_m$. Therefore, a change in its code was performed to make available the possibility of read Legendre polynomials. The changes were introduced in the file fiat.c of the DLS PIPELINE (see below for the new code attached).

```
/*
 * Calculates f(ellip_)=Pn_m=Cn_m Ln(x)Lm(y)
 * Introduced by Guadalupe Canas in order to use
 * Weak-Lensing Software
 * Version 1.0
 */

/*
 * Function Legendre: generates legendre polynomials
 corresponding to the index n through the iterative formulae
 */

float Legendre(int n, float variable){
    float P[n];
    if (n==0){
        P[0]=1;
        return P[0];
    }
    else if (n==1){
        P[1]=variable;
        return P[1];
    }
    else{
        P[n]=((2.0*n-1.0)*variable*Legendre(n-1,variable)
        -(n-1)*Legendre(n-2,variable))/n;
        return P[n];
```

```
    }
}

/*
 * Function fiat_xylookup_legendre: modified from fiat_xylookup.
 * It takes the coefficients of the fitting obtained from
 * fiat_read and calculates the expression
 * f(ellip_)=Pn_m=Cn_m Ln(x)Lm(y) for e_1=f(x,y) and e_2=f(x,y)
 * in terms of a legendre orthonormal basis.
 */

int fiat_xylookup_legendre(float x, float y,
float *e1, float *e2, float *p, int order)
{
    int npar,e_1indx,e_2indx,l_xorder,l_yorder;
    //define npar (position of coefficients in text files),
    double legendre_xtmp, legendre_ytmp;

    /* setup */
    npar = (order+1)*(order+2);
    *e1 = *e2 = 0.0; //set the ellipticities at 0.0
    e_1indx = 0; //e_1 is at the beggining of the file
    e_2indx = npar/2; //e_2 is at position npar/2 = 15

    for(l_yorder=0, legendre_ytmp=1.0;
    l_yorder<=order;
    l_yorder++, legendre_ytmp*=Legendre(l_yorder,y))
    {
        for(l_xorder=0,legendre_xtmp=1.0;
        l_xorder<=order-l_yorder;
        l_xorder++, legendre_xtmp*=Legendre(l_xorder,y))
        {
            *e1 += p[e_1indx]*legendre_xtmp*legendre_ytmp;
            //expression for e_1
            *e2 += p[e_2indx]*legendre_xtmp*legendre_ytmp;
            //expression for e_2
            e_2indx++; //go ahead
            e_1indx++; //go ahead
        }
    }
```

```
    return 0;
}
```

Finally, WL_UTILS.PY writes another text file that makes DLSCOMBINE to know the path to the fitting coefficients text file and the path to the analysed image we need to correct (denominated *specfile*).

A.5.4 Estimation of Dark Matter Halos

The estimation of DM halos is performed thanks to the software FIATMAP. Therefore, WEAK-LENSING software only contain two scripts designed to call FIATMAP properly and read the resulting convergence maps to transform then into signal-to-noise ones. It is extremely important to use a grid or supercomputer in order to carry out this performance.

- WL_FILTER_MAG_GAL.PY: PYTHON file. It splits the final galaxies catalogue into three different galaxies sub-catalogues attending to several cuts in magnitude (by default the cuts are $m_cut = \{20, 22\}$). It calls the Monte Carlo algorithm implemented in FIATMAP 1000 times obtaining 1000 randomized convergence maps and the real convergence κ map for each sub-catalogue (if the default cuts are maintained, 3000 randomized maps are obtained).

- WL_MONTECARLOANALIZER.PY: It reads the 1000 randomized maps corresponding to each galaxies sub-catalogues and calculates the standard deviation σ of the convergence *kappa* value for each pixel by fitting a Gaussian function thanks to the packages SCIPY and NUMPY. Then, it obtains signal-to-noise maps dividing each pixel of the original convergence *kappa* maps by the standard deviation σ associated to each pixel. Finally, it filter signal-to-noise maps for a cut in the signal-to-noise value taking into account a value passed as argument.

A.6 Acknowledges

I acknowledge Santander Supercomputacion support group at the University of Cantabria who provided access to the supercomputer Altamira Supercomputer at the Institute of Physics of Cantabria (IFCA-CSIC), member of the Spanish Supercomputing Network, for performing simulations/analyses.

This research has made use of the NASA/IPAC Extragalactic Database (NED) which is operated by the Jet Propulsion Laboratory, California Institute of Technology, under contract with the National Aeronautics and Space Administration

REFERENCES

[1] (1999). On the Efficiency of Nearest Neighbor Searching with Data Clustered in Lower Dimensions.

[2] (2008). Hubble maps dark matter web in a large galaxy cluster. http://www.esa.int/var/esa/storage/images/esa_multimedia/images/2008/01/hubble_maps_dark_matter_web_in_a_large_galaxy_cluster/10144540-2-eng-GB/Hubble_maps_dark_matter_web_in_a_large_galaxy_cluster.jpg. Accessed: 2015-11-19.

[3] (2012). Artist's impression of the expected dark matter distribution around the Milky Way. http://cdn.eso.org/images/screen/eso1217b.jpg. Accessed: 2016-02-10.

[4] (2014). Planck Cosmic Recipe. http://www.esa.int/spaceinimages/Images/2013/03/Planck_cosmic_recipe. Accessed: 2015-11-18.

[5] (2014). Story of the Universe. http://www.nasa.gov/mission_pages/planck/multimedia/pia16876b.html#.Vkmy9ITldzM. Accessed: 2015-11-16.

[6] Bernstein, G. M. and Jarvis, M. (2002). Shapes and shears, stars and smears: Optimal measurements for weak lensing. *The Astronomical Journal*, 123(2):583.

[7] Bílek, M. (2016). Galaxy interactions: dark matter vs. Modified Newtonian dynamics (MOND). *ArXiv e-prints*.

[8] Blandford, R. and Narayan, R. (1986). Fermat's principle, caustics, and the classification of gravitational lens images. *Astrophysical Journal, Part 1*, (vol. 310):568–582.

[9] Carroll, B. W. and Ostlie, D. A. (2014). *An Introduction to Modern Astrophysics*. Pearson, second edition.

[10] Clowe, D., Bradač, M., Gonzalez, A. H., Markevitch, M., Randall, S. W., Jones, C., and Zaritsky, D. (2006). A Direct Empirical Proof of the Existence of Dark Matter. *apjl*, 648:L109–L113.

[11] Cook, R. I. (2012). *Measuring Weak Lensing Sensitivity and Systematics*. Department of Physics at Brown University.

[12] Courbin, F., Saha, P., and Schechter, P. L. (2002a). Quasar Lensing. In Courbin, F. and Minniti, D., editors, *Gravitational Lensing: An Astrophysical Tool*, volume 608 of *Lecture Notes in Physics, Berlin Springer Verlag*, page 1.

[13] Courbin, F., Saha, P., and Schechter, P. L. (2002b). Quasar Lensing. In Courbin, F. and Minniti, D., editors, *Gravitational Lensing: An Astrophysical Tool*, volume 608 of *Lecture Notes in Physics, Berlin Springer Verlag*, page 1.

[14] Finoguenov, A., Streblyanska, A., Hasinger, G., Hashimoto, Y., and Szokoly, G. (2005). New cluster candidates in the extended {XMM} lockman hole field. *Advances in Space Research*, 36(4):710 – 714. Clusters of Galaxies: New Insights from XMM-Newton, Chandra and {INTEGRAL}.

[15] Fischer, P. and Tyson, A. (1997). The mass distribution of the most luminous x-ray cluster rxj 1347.5-1145 from gravitational lensing. *The Astronomical Journal*, (vol. 114, 1):14–26.

[16] Fischer, P. and Tyson, J. A. (1997). The Mass Distribution of the Most Luminous X-Ray Cluster RXJ 1347.5-1145 From Gravitational Lensing. *The Astrophysical Journal*, 114:14–24.

[17] Fisher, D., Fabricant, D., Franx, M., and van Dokkum, P. (1998). A spectroscopic survey of the galaxy cluster cl 1358+62 at z = 0.328. *The Astrophysical Journal*, 498(1):195.

[18] Freeman, K. and Mcnamara, G. (2006). *In Search of Dark Matter*. Springerº, firs edition.

[19] Gelmini, G. B. (2015). TASI 2014 Lectures: The Hunt for Dark Matter. *ArXiv e-prints*.

[20] Guth, A. H. (1981). The inflationary universe: A possible solution to the horizon and flatness problems. *Physics Review Letters*, (23):347.

[21] Holwerda, B. W. (2009). *Source Extractor for dummies*. Space Telescope Science Institute.

[22] Kaiser, N. and Squires, G. (1993). Mapping the dark matter with weak gravitational lensing. *The Astrophysical Journal*, 404(2):441–450.

[23] Kaiser, N., Squires, G., and Broadhurst, T. (1995). A method for weak lensing observations. *The Astrophysical Journal*, 449:460.

[24] Miyazaki, S., Hamana, T., Ellis, R. S., Kashikawa, N., Massey, R. J., Taylor, J., and Refregier, A. (2007). A subaru weak-lensing survey. i. cluster candidates and spectroscopic verification. *The Astrophysical Journal*, 669(2):714.

[25] Norton, A. (2006). *An Introduction to Astrophysics and Cosmology*. The Open University, second edition.

[26] Peter, A. H. G. (2012). Dark Matter: A Brief Review. *ArXiv e-prints*.

[27] Planck Collaboration, Ade, P. A. R., Aghanim, N., Arnaud, M., Ashdown, M., Aumont, J., Baccigalupi, C., Banday, A. J., Barreiro, R. B., Bartlett, J. G., and et al. (2015). Planck 2015 results. XIII. Cosmological parameters. *ArXiv e-prints*.

[28] Shapiro, I. I. (1964). Fourth test of general relativity. *Phys. Rev. Lett.*, 13:789–791.

[29] Taylor, J. R. (1997). *An Introduction to Error Analysis: The Study of Uncertainties in Physical Measurements*. University Science Books, second edition.

[30] Weinberg, S. (2008). *Cosmology*. Oxford University Press, second edition.

[31] Wittman, D., Dell'Antonio, I. P., Hughes, J. P., Margoniner, V. E., Tyson, J. A., Cohen, J. G., and Norman, D. (2006). First Results on Shear-selected Clusters from the Deep Lens Survey: Optical Imaging, Spectroscopy, and X-Ray Follow-up. *The Astrophysical Journal*, 643:128–143.