Manual of the LensTool program

Jean-Paul Kneib

August 8, 2014

Contents

1 How to run the LensTool program **5**
- 1.1 Foreword . . . 5
- 1.2 Compilation and link . . . 5
- 1.3 Run . . . 8
- 1.4 Troubleshooting/FAQ . . . 8

2 Input File **11**
- 2.1 First identifiers . . . 11
- 2.2 Second identifiers . . . 12
  - 2.2.1 runmode . . . 12
  - 2.2.2 grille . . . 24
  - 2.2.3 potential . . . 25
  - 2.2.4 limit . . . 31
  - 2.2.5 potfile . . . 34
  - 2.2.6 cline . . . 36
  - 2.2.7 grande . . . 40
  - 2.2.8 observ . . . 42
  - 2.2.9 cosmologie . . . 43
  - 2.2.10 cosmolimit . . . 44
  - 2.2.11 champ . . . 45
  - 2.2.12 cleanlens . . . 45
  - 2.2.13 image . . . 49
  - 2.2.14 source . . . 52
  - 2.2.15 fini . . . 53
- 2.3 Examples . . . 53
  - 2.3.1 Typical configurations of Arcs . . . 53
  - 2.3.2 Optimization with one multiple image . . . 53
  - 2.3.3 Optimization with two multiple images . . . 53
  - 2.3.4 Optimization with arclet data . . . 53

3 Datafiles **55**
3.1 Input Datafiles . . . 55
3.1.1 WCS header . . . 55
3.1.2 Object file . . . 55
3.1.3 Marker file . . . 56
3.1.4 IPX pixel-image file . . . 57
3.1.5 FITS pixel-image file . . . 57
3.2 Output Datafiles . . . 58
3.2.1 Potential file . . . 58
3.2.2 Source file . . . 59
3.2.3 Arclet files . . . 59
3.2.4 Critical and caustic lines files . . . 60
3.2.5 Source marker file . . . 61
3.2.6 Prop files . . . 62
3.2.7 Invert files . . . 62
3.2.8 Best file . . . 62
3.2.9 Bayesian optimisation . . . 62

4 Getting Started **63**
4.1 Cleanlens section example . . . 69
4.2 Potfile section example . . . 69

Chapter 1

How to run the LensTool program

1.1 Foreword

This program was born in the "Laboratoire d'Astrophysique de Toulouse" during the course of my Ph.D. Thesis (Kneib 1993). Since then, it has grown a lot with the great help of Yannick Mellier, and the uses of some numerical subroutines of Henri Bonnet, Karim Bouyouceff and Jorgen Maeland. More recently, Eric Jullo has, among other things, implemented the Bayesian optimisation scheme using the BayeSys MCMC sampler of John Skilling.

I would be most grateful to anybody using this program if they would send me their comments and criticisms, and also report any bugs. I will then correct the program and make it better.

My e-mail address is: *jean-paul.kneib@oamp.fr*.

1.2 Compilation and link

The program is compiled and linked with the **configure** and **make** commands. It is linked to three libraries :

libwcs.a : the Image World Coordinates System Library developed by the Smithsonian astrophysical Observatory

libcfitsio.a : the library that deals with fits file format developed at HEASARC. The CFITSIO library version must be at least version 2.510

libpgplot.a : the library, developed by Tim Pearson, that enables the plots of the Bayesian results

Those libraries can be downloaded at :
– http://tdc-www.harvard.edu/software/wcstools
– http://heasarc.gsfc.nasa.gov/docs/software/fitsio/fitsio.html
– ftp://ftp.astro.caltech.edu/pub/pgplot

To compile this package, just follow the standard procedure :

1. 'cd' to the directory containing the package
2. Type './configure' to configure the package for your system.
3. Type 'make' to compile the package.

If configure does not find the **cfitsio**, the **wcs** or the **pgplot** libraries in the standard libraries, it will ask you for them. You then have to enter the absolute path to these libraries.
For example:

```
/home/user/cfitsio
/home/user/wcstool-3.7.2
/home/user/pgplot
```

You may also (depending on the system) need to set the environment variables `$CFITSIO_DIR`, `$WCS_DIR`, and `$PGPLOT_DIR` to be the top level directory for each package.

Once compiled, the binary file is called **lenstool** and is located in the `./src` directory. You can then copy it to your `~/bin` directory.

The **lenstool_tab** binary executable is located in the `./table_src` directory. You can use it to produce a `lenstool_tab` binary file used by the NFWg profile. There is an example in the `./examples_table` directory.

I added also 2 tools: **Histogram** and **Histogram2D**, from the McAdam package of Phil Marshall. You can use them to visualise the MCMC samples contained in the `bayes.dat` file. Once compiled, they are located in the `./utils` directory.

The PERL visualization tools to display the LENSTOOL results in DS9 are in the `./perl` directory. To use them, you must define the `LENSTOOL_DIR` environment variable in your `.tcshrc` or `.bashrc` file and add this `perl` directory to your `PATH` environment variable :
With the tcsh or csh shell :

```
setenv LENSTOOL_DIR /home/user/lenstool-6.5
setenv PATH ${PATH}:${LENSTOOL_DIR}/perl:${LENSTOOL_DIR}/utils
```

With the bash shell :

```
export LENSTOOL_DIR=/home/user/lenstool-6.5
export PATH=${PATH}:${LENSTOOL_DIR}/perl:${LENSTOOL_DIR}/utils
```

The distribution has been tested on :

- Linux Fedora Core 3, Suse
- Mac Tiger 10.4
- Mac OS X

1.3 Run

LensTool is run easily by typing the command:

`lenstool` *filename*

The *filename* is the basename of the inputfile named "*filename*.par".

The first step of the LensTool program is to edit the inputfile using the "vi" editor, so that the user can change one or many parameters of the inputfile.

The following steps correspond to what have been asked in the inputfile.

The lens tool program creates different files: data files that usually have the extension ".dat" and a parameter file "para.out" listing the values read from the inputfile (this is a good way to check if the program has understood the inputfile).

1.4 Troubleshooting/FAQ

I got the following error when I launch Histogram or Histogram2D:

./Histogram: error while loading shared libraries: libpgplot.so: cannot open shared object file: No such file or directory

This means that your **LD_LIBRARY_PATH** environment variable doesn't point to the directory containing the libpgplot.so file. You may set the variable with this bash command:

```
export LD_LIBRARY_PATH=<directory containing libpgplot.so>
```

or in the csh shell:

```
setenv LD_LIBRARY_PATH <directory containing libpgplot.so>
```

I got the following error when I compile Histogram and Histogram2D tools:
ld: Undefined symbols: .objc_class_name_AQTAdapter...

Change the file `utils/Makefile.am` in the following way:

```
Histogram_LDFLAGS =  -L/scisoft/i386/Packages/pgplot-5.2.2/ -L/usr/X11R6/lib
-Wl,-framework -Wl,Foundation -lpgplot -lgfortran -laquaterm -lX11 -lpng
-lcc_dynamic
Histogram_LDADD =
```

Then, run `make` again in the `utils` directory. Make sure you are compiling with either the `g77` or `f77` compiler!
On MAC OS X 10.5.2 Leopard, `-lcc_dynamic` option is not required.

I got the following error when I compile Histogram and Histogram tools:
Error on line 638: Declaration error for x: adjustable dimension on non-argument
Error on line 638: wr_ardecls: nonconstant array size

You are probably using the f2c compiler. This is not supported by LensTool! The solution is to change to the g77 compiler...

I want to compile with the *icc* compiler

Define the `CC` environment variable as follow before running `./configure`
in csh shell : `setenv CC icc`
or in bash shell : `export CC=icc`

The same for the FORTRAN compiler :
in csh shell : `setenv F77 g77`
or in bash shell : `export CC=g77`

I got an autoconf version error when I run the ./configure script
In the distribution, I provide a configure.in script, which is used to build the ./configure script. You can rebuild the ./configure script by typing the command :

```
aclocal
autoconf
```

Chapter 2

Input File

Here is a rapid description of each keywords that appear in the *inputfile*. Keywords are of 2 types: first identifier and second identifier. First identifier are more general and deal with a specific part of the program (like computation of the critical and caustic curves). Second identifiers are specialized for each first identifier, and define some constant or file.

2.1 First identifiers

runmode this identifier is the most important one, and determine what the program will do. (compulsory)

grille defines some parameters such has the number of potential mode, the total number of potential mode that are going to be test, the grid mode, and the number of rows and columns in the grid. (compulsory)

potential or **potentiel** under this identifier is defined one mode of the gravitational potential. One can define a global potential with many modes, for each mode a first identifier "potential" must be defined.

limit under this identifier are defined the constraints on the potential (more precisely on one mode). This identifier has to follow the identifier of the corresponding "potential" mode. (used only in the invert runmode)

potfile under this identifier are defined the default parameters for all the galaxy scale mass components that account for perturbations to the cluster potential by the galaxies.

cline under this identifier are defined the parameters to compute the critical and the caustic lines.

cosmologie or **cosmology** under this identifier are defined the cosmological parameters Ω_0, λ and H_0.

champ under this identifier is defined the size of the field used in some calculations such as the dimension of the grid for the inversion of the lens equation.

grande under this identifier is defined the way to represent the computed deformation of objects.

observ under this identifier is defined the different noise that can be add to a gravitational image, such as seeing or Poisson Noise.

source under this identifier are defined some characteristics of sources when a random drawing is done.

image under this identifier are defined some characteristics of images, multiple images or arclets.

cleanlens under this identifier are defined some parameters to retrieve the shape of the source knowing a pixel-frame of the image.

fini tells the program to stop the reading of the inputfile. (compulsory)

2.2 Second identifiers

For each first identifier, we will defined the second identifier, and gives the default value, and their uses, with an example.

2.2.1 runmode

this identifier is the most important one, and determine what the program will do.

reference *int RA DEC*
Set the reference point for the system to study. This keyword is used to convert the relative coordinates used in LensTool to absolute coordinates used in some input or output files.
If *int*=1, *RA* and *DEC* are in the sexagesimal format $hh : mm : ss$

and $dd : mm : ss$.
If *int*=3, *RA* and *DEC* are in degrees.

arclet *int filename* or **image** *int filename*
int 0: if false 1: if true.
If true the program will read a list of arclet in *filename* find their sources [and put them in the output file *source.dat*] and recompute all their images [and put them in the output file *image.all*].
Moreover the program will create other 'arclets' output files:
–*image.dat* where only the computed arclet with deformation τ less than **grande large_dist** are indicated (basically, *image.dat* does not include giant arcs).
–*dist.dat* information about the ellipticity of all the arclets, including weakly distorted images and giant arcs.
–*sort.dat* same as *image.all* but sorted from the most elongated (giant arcs) to the less (arclets).

Interest: find counter images.

Format: the format of *filename* is describe in Sect. 3.1.1. It is an ASCII column format of the form:
$\{\ i\ x_i\ y_i\ a_i\ b_i\ \theta_i\ z_i\ \}$

Note: θ_i is 90° relative to the PA definition, x and y are RA and Dec in decimal degrees.

VISUALISATION:
PICT: *source.dat*, *image.dat*, *image.all*, *sort.dat* are 'arclet' style file (see Sect. 3.2.3) and can be visualized with PICT under the **arclet** qualifier:
To visualize the file *image.all* and *source.dat* one has to write in the '*input.*in' PICT input file:

```
 arclet
          narc     2
          namein   1     0   image.all
          namein   2     0   source.dat
          end
 frame
          dmax     30.
          end
 fini
```

This will display the ellipses of *image.all* and *source.dat* in a frame of size [x: -30. +30, y:-30. +30]

source *int filename*
int 0: if false 1: if true.
If true the program will read a list of sources in *filename* [and put them in the output file *source.dat*] and compute all their images [and put them in the output file *image.all*].

Moreover the program will create the same 'arclets' output files as for the **arclet** preceding sub-qualifier:
– *image.dat* where only the computed arclet with deformation τ less than **grande large_dist** are indicated.
dist.dat information about the ellipticity of all the arclet sorted from the most elongated to the less.
– *sort.dat* same as *image.all* but sorted from the most elongated to the less.

Interest: usually used to show typical image configurations.

PICT: see previous identifier

Format: see previous identifier

Note: both **arclet/image** and **source** identifiers can be used at the same time.

time *int1 int2 float filename*

int1 0: if false 1: if true.

If true will compute for the redshift *float* a pixel-frame (*int2* × *int2*) of the (relative) arrival time (in year) of each pixel of the image plane (area defined by **frame**).

Results are written in the pixel-frame file *filename*

The format of *filename* is the 'ipx' simple pixel-frame format (format 2 for PICT).

Note: The arrival time is defined by:

$$\tau_a(\vec{\xi^I}, z^S) = \frac{1}{c}\frac{D_{LS}D_{OL}}{D_{OS}}\left(\|\nabla\varphi(\vec{\xi^I}, z^S)\|^2 - \varphi(\vec{\xi^I}, z^S)\right)$$

φ is the lens-normalized projected potential:

$$\varphi = \frac{2}{c^2}\frac{D_{LS}D_{OL}}{D_{OS}}\phi \ ,$$

where ϕ is the Newtonian projected potential:

$$\phi = \frac{1}{D_{OL}^2}\int \Phi^{3D} dl \ .$$

The (relative) arrival time surface corresponds to the time of arrival at the Image Plane of a flash that left at the same time the Source Plane. It it absolute in the sense we have substract the mean travel time between the two plane. To find out the time-delay between two images of the same source one has to know the position of the images and then compute the difference between the two corresponding arrival time.

PICT: The following example of a PICT inputfile allow to display the pixel-frame *filename* with a gray lut starting at zgmin=-10 (white) ending at zgmax=10 (black) in a frame of size [x: -20. +20, y:-20. +20]

The position of the pixel-frame is automatically scaled (scale= 1).

contour
filein **1** ***filename***
format **2**
scale **1**
zgmin **-10**
zgmax **10**
end
frame
dmax **20.**
end
fini

When the pixel-frame of the time delay is displayed, one can then get the value of the time-delay with the cursor using the PICT command (c)oord.

ampli *int1 int2 float filename*

int1 0: if false 1: if true.

Same as **time** but compute the amplification $\mu^{-}1$ in the image plane. If *int1*= 1, compute $\mu^{-}1$, if *int1*=2, compute its absolute value and if *int1*=3, compute its absolute value in magnitude.

If *int1*=5, compute the convergence map in the image plane. If *int1*=6, compute the shear map in the image plane.

If *int1*=-1, compute the absolute value of $\mu^{-}1$ in the source plane considering every images (prototype).

The amplification is defined by:

$$\mu = \Big((1-\kappa)^2 - \gamma^2\Big)^{-1} \ ,$$

with κ and γ are the convergence and the shear respectively. They are defined by:

$$2\kappa = \nabla^2\varphi \ ,$$

and

$$\gamma^2 = \frac{1}{4}\Big(\partial_{xx}\varphi - \partial_{yy}\varphi\Big)^2 + \Big(\partial_{xy}\varphi\Big)^2 \ .$$

PICT: see **time**

poten *int1 int2 float filename*

int1 0: if false 1,2: if true.

int2 size of the square grid.

float redshift of the source plane (z_S).

Same as **time** but compute the relative (*int1*= 1) or absolute (*int1*= 2) projected potential.

The relative projected potential (*int1* = 1) is defined by

$$\varphi(\vec{\xi^I}, z^S) = \frac{2}{c^2} D_{OL} \frac{D_{LS}}{D_{OS}} \phi(\vec{\xi^I})$$

Where the absolute projected potential (*int1* = 2) is: $\phi(\vec{\xi^I})$
Because it is absolute it does not depend of the redshift z_s (*float*).

PICT: see **time**

mass *int1 int2 float filename*

int1 0: if false 1,2: if true.

int2 size of the square grid.

float redshift of the source plane (z_S).

Same as **time** but compute the relative (*int1*= 1) the absolute (*int1*= 2 and 4) projected mass-density, or the integrated (*int1*= 3) projected mass-density.

The relative projected mass-density (*int1* = 1) (called also convergence) is determined by

$$\kappa(\vec{\xi^I}, z^S) = \frac{\nabla^2 \varphi(\vec{\xi^I}, z^S)}{2} = \frac{\Sigma(\vec{\xi^I}, z^S)}{\Sigma_{crit}} \ .$$

The critical density is defined by:

$$\Sigma_{crit}(z^S) = \frac{c^2}{4\pi G} \frac{D_{OS}}{D_{LS} D_{OL}}$$

The absolute projected mass-density (*int1*= 2 and 4) is determined by:

$$\Sigma(\vec{\xi^I}) = \Sigma_{crit} \frac{\nabla^2 \varphi}{2} = \frac{\nabla^2_{\vec{\xi^I}} \phi(\vec{\xi^I})}{4\pi G} \qquad \begin{array}{c} in\ g/cm^2\ (int = 2) \\ in\ 10^{12} M_\odot / kpc^2\ (int = 4) \end{array}$$

Because it is absolute, it does not depend of the redshift z_s (*float*).

The integrated projected mass-density (*int1* = 3) is determined by:

$$M(\vec{\xi^I}) = \frac{\nabla^2_{\vec{\xi^I}}\phi(\vec{\xi^I})}{4\pi G} S_{pixel} \qquad (in\ 10^{12} M_\odot/pixel)$$

Obviously, it depends on the pixel size. As for the **time** function, the area covered by the image is defined in the **frame** section.

In Bayesian optimisation mode, you can get the projected error mass-density (*int1* =5) in $10^{12} M_\odot/pixel$. The map size in arcsec is defined with the `dmax` keyword in the `champ` section. In the rectangular field case, the image size is (X,Y) = (scaling**int2*, *int2*).

PICT: see **time**

shear *int1 int2 float filename*

int1 0: if false, true otherwise.
Same as **time** but compute :
int1=1 the shear γ defined by:

$$\gamma = \sqrt{\frac{1}{4}\Big(\partial_{xx}\varphi - \partial_{yy}\varphi\Big)^2 + \Big(\partial_{xy}\varphi\Big)^2}\ .$$

int1=2 the ellipticity ϵ defined by:

$$\epsilon = \frac{q^2 - 1}{q^2 + 1}\ ,$$

where q is the ratio of the amplification matrix eigenvalues λ_1/λ_2.

If *int1 ¡ 0*, the behavior is the same but for pixels considered in the source plane.

shearfield *int float filename int2*

int 0: if false 1,2: if true.
If true will compute for the redshift *float* the shear field at [*int2*×*int2*] points of the Image Plane (area defined by **frame**).
Results are written in the 'arclet'-type file *filename*
If *int* = 1 the size of the ticks correspond to the induced ellipticity by the mass distribution.

If *int* = 2 the ticks show only the polarization of the field.

Note: Do not mix the identifiers **shear** and **shearfield**. **shear** is just a pixel-frame of the intensity of the shear with no indication of the orientation of the shear. **shearfield** on the contrary will give you the orientation of the shear and its intensity but only in *int2*x*int2* points. Default *int2*=25.
PICT: The following example of a PICT inputfile allow to display the shearfield *filename* in a frame of size [x: -20. +20, y:-20. +20]

```
arclet
        narc     1
        namein   1     0   filename
        end
frame
        dmax     20.
        end
fini
```

study *int filename*
int 0: if false 1: if true.
The purpose of **study** is a statitistical analysis of the arclets properties to infer the probable redshift of sources.
If true the program will read a list of arclet in *filename* and computes for different redshift the ellipticity, size and orientation of the sources. It will also give the z0- zm- zmin zm+ z0+ (file z.dat) as defined in Kneib et al. 1994.
The format of *filename* is exactly the same as the one of **arclet**.
This program will create four output files:
– *ess.dat* : It is the record of the variation of the ellipiticy with the redshift for all the arclets of *filename*. *ess.dat* is an ASCII file, that has the following format:
{ i z_{ij} dr_{ij} τ^S_{ij} ε^S_{ij} θ_{ij} n_{ij} Δ_{ij} ez_i }
where i is the index of the arclet, z_{ij} the redshift at step j=1,N , dr_{ij} the cosmological ratio D_{LS}/D_{OS}, τ^S_{ij} the deformation, ε^S_{ij} the ellipticity, θ_{ij} the orientation (90° relative to PA), n_{ij} the multiplicity, Δ_{ij} a value that is equal to zero when τ^S_{ij} is minimal, ez_i the estimated most probable redshift.
– *z.dat* : is a synthetic file with th following format:

{ i $z0{-}_i$ $zm{-}_i$ $zmin_i$ $zm{+}_i$ $z0{+}_i$ $(a/b)_i^I$ $(a/b)_i^S$ ez_i $\tau^S(ez_i)$ }
where i is the index of the arclet, $zmin_i$ is the true most probable redshift, $z0{-}_i$ $zm{-}_i$ $zm{+}_i$ $z0{+}_i$ the errors on $zmin_i$, $(a/b)_i^I$ is the axis ratio of the arclet, $(a/b)_i^S$ is the axis ratio of the source at $zmin_i$, ez_i is the estimated most probable redshift (by looking at the zero of Δ_{ij}, $\tau^S(ez_i)$ is the deformation at ez_i.
– *source_T.dat* : is a extended object (ellipse) file with the format:
{ i x_i^S y_i^S a_i^S b_i^S θ_i^S $zmin_i$ $zm{-}_i$ $zm{+}_i$ $(a/b)_i^S$ μ_i}
where i is the index of the arclet, x_i^S y_i^S the position of the source, a_i^S b_i^S the major and minor semi-axis of the source, θ_i^S the orientation (90° relative to PA) of the source at redshift $zmin_i$, $zm{-}_i$ $zm{+}_i$ are the errors on $zmin_i$, $(a/b)_i^S$ is the axis ratio of the source at $zmin_i$, μ_i is the amplification of the image for a source at $zmin_i$.
– *azT.dat* : is an extended arclet file with the format:
{ i x_i^I y_i^I a_i^I b_i^I θ_i^I $zmin_i$ μ_i $\tau^S(zmin_i)$}
where i is the index of the arclet, x_i^I y_i^I the position of the arclet, a_i^I b_i^I the major and minor semi-axis of the arclet, θ_i^I the orientation of the arclet (90° relative to PA), $zmin_i$ is the true most probable redshift, μ_i is the amplification of the image for a source at $zmin_i$, $\tau^S(zmin_i)$ is the deformation of th source at $zmin_i$.

Note: *source_T.dat* and *azT.dat* can be visualized by PICT without problems by the same way as *image.dat* or *source.dat*.

imseeing *float*
of the arclet assuming that both the profile of the arclet and of the seeing are Gaussian. It's a very simple and crude correction.
Used in the **study** mode, and **inverse arcletstat** mode.
Default value is 0 (meaning no seeing correction).

grille *int1 int2 float*
int1 0: if false 1,2: if true.
If true, will create a grid of *int2* points.
If *int1*= 1, it considers this grid as the Source Plane at the redshift of *float*, and compute the corresponding grid in the Image Plane.
If *int1*= 2, it considers this grid as the Image Plane and compute the corresponding grid in the Source Plane at the redshift of *float*.
The gird coordinates either in the source or in the image planes are

defined in the **frame** section by the *dmax* or by the set of keywords (*xmin*, *xmax*, *ymin*, *ymax*).
Results are in the files: gi1.dat (image vertical grid) gi2.dat (image horizontal grid) and gs1.dat (source vertical grid) gs2.dat (source horizontal grid).
All these file have the following format:
$\{j\ x_i\ y_i\ \}$
where j is an index, x_i y_i are the pixel coordinates.

PICT: here is an example of PICT inputfile that draw the image grid:

```
curve
        nfile     2
        namein    1   0   gi1.dat
        column    1   3
                  2   3
        namein    2   0   gi2.dat
        column    2   3
                  2   3
        end
fini
```

The grid may also be displayed on ds9 by using the *grid* perl script.

inverse *int1 float1* [*float2*]
int1 0: if false > 1: if true.
If true, will enter the optimization mode.
If *int1*=1, the optimisation method is the parabolic method.
If *int1*=2, the optimisation looks for the galaxy scale parameters sigma and cut radius that give the best lens model. According to the gridding stated in the `potfile` section, it runs over the grid and computes the best χ^2 in each node with the parabolic optimisation method. (see `potfile` Section).
If *int1*=3, the optimisation method is the bayesian method. This optimisation method is very slow but is less sensible to local χ^2 minima than the parabolic method.
If *int1*=4, the optimisation method is a maximum likelihood method but based on the BayeSys algorithm. The cooling factor is not limited to 1.

For inverse method 1 and 2, *float1* gives the maximum number of iterations for the parabolic method.
For inverse method 3, *float1* gives the speed of calculation of the Bayesian optimisation and *float2* sets the number of sampling iterations. As we use 10 Markov chains at the same time, each iteration produces 10 samples. The default value is the number of iterations needed to complete the Burn-in phase. The samples are saved in the `bayes.dat` ASCII file.
float1 sets the rate $\delta\lambda$ by which is raised the likelihood at each step of the Markov Chain. At the beginning, $\lambda = 0$ and at the end of the Markov Chain $\lambda = 1$. Default ($\delta\lambda = 0.5$).
The optimisation process will create a `best.par` and a `bestopt.par"` files. They contain the best model and the best model + optimisation limits respectively. Additionally, useful information related to the optimisatoin are provided in their header.

minchi0 *float*

float value of the χ^2 at which the optimization program will stop in the case of a parabolic optimisation. Default is 0, but this is not dramatic is case of slow convergence or even non-convergence at all. The number of iterations is also controlled from **inverse** qualifier.

prop *int float filename*

int 0: if false 1: if true.
If true, will compute for the current potential some properties for a Source Plane at redshift *float*. This include the orientation of the shear, the magnification, convergence, shear, τ_{pot}, etc...
Results are put in the files: *filename*. This is a huge datafile, hence be careful!
Moreover it is not advise to use it, because it is not fast. Users are advised to use the **time**, **mass**, **ampli**, **shear** and **shearfield** identifiers (much faster).

pixel *int1 int2 filename*

int1 0: if false 1: if true.
If true, will create a pixel-frame *filename* of *int2*$\times$*int2* pixels that corresponds of the brightness intensity of all the arc(let)s computed from the objects defined in the **image/arclet** or/and the **source** identi-

fiers.
If **observ** is set the program will convolve the true image by a seeing and add Poisson Noise.
The purpose of **pixel** is to make realistic images of arc(let)s from a given projected potential which can be compared directly with real CCD images of arc(let)s (see for example Kneib et al. 1993, Fig. 1b). **pixel** can make images of several arc(let)s coming from various sources on the same frame (Note: this is not the case with the **iso** identifier).

marker *int float filename*
int 0: if false 1: if true.
If true, will read the points markers in the file *filename* and compute the corresponding points in the Source Plane at redshift *float*.
Results are put in the file "marker_s.dat".
filename must be a 3 columns ASCII file: { i x_i y_i }
It can be easily created with PICT using the '(g) get_line' command.
The output file "marker_s.dat" has the same format: { i x_i^S y_i^S }

PICT: here is an example to visualized *filename* and "marker_s.dat":

```
curve
        nfile     2
        namein    1   0   filename
        column    1   3
                  2   3
        namein    2   0   marker_s.dat
        column    2   3
                  2   3
        end
fini
```

radialprop *int float1 float2*
int 0: if false 1,2,3: if true.
If true, will compute for the current potential some properties of images for a Source Plane at redshift *float1*. These properties differs according to the value of *int*. They are computed along a radial line starting at the center of the first clump and with the position angle *float2* expressed in degree.
Results are put in the files: "radial.dat" and "radial2.dat". These are huge datafile, hence be careful!

If *int*= 1 only "radial.dat" will be created with the following format:
$\{ r_i \;\; \theta \;\; a_i \;\; b_i \;\; \theta_{shear_i} \;\; \mu_i \}$
where r_i is the radial distance, θ is the direction of the radial axis (90° relative to PA), a_i is the radial eigenvalue of the magnification matrix, b_i is the orthoradial eigenvalue of the magnification matrix, θ_{shear_i} is the direction of the shear (90° relative to PA), μ_i is the amplification.
If *int*= 2 "radial.dat" and "radial2.dat" will be created, "radial.dat" will have the following format:
$\{ r_i \;\; \varepsilon_i \;\; \delta_i \;\; \tau_i \;\; \theta_{shear_i} \;\; \mu_i \}$
where r_i is the radial distance, ε_i is the induced ellipticity, δ_i is the induced distortion, τ_i is the induced deformation, θ_{shear_i} is the direction of the shear (90° relative to PA), μ_i is the amplification.
The file "radial2.dat" will have the following format:
$\{ r_i \;\; (\tau_i^2/r_i) \;\; (\tau_i/r_i) \}$
If *int*= 3 only "radial.dat" will be created with the following format:
$\{ r_i \;\; \alpha_i \;\; (r_i - cr) \;\; (r_i - ct) \}$
where r_i is the radial distance, α_i is the deflection angle, cr is the radial critical radius, ct the tangential critical radius.

verbose *int*

If *int* is 0, minimal log information is printed to the screen;
if *int* is 1, then some debugging information is printed to the screen.

2.2.2 grille

defines some parameters such has the number of potential mode, the total number of potential mode that are going to be test, the grid mode, and the number of rows and columns in the grid.

nombre *int*

int represents the number of points of the grid used to invert the lens equation (from Source Plane to Image Plane). Must be an odd number, typically 20 or 30. Increasing it will increase the precision in finding all the images, but increase the computation time too. Values larger than 80 are not recommended.
Default: *int*= 30.

polaire *int*
Set the grid to a polar shape if *int*= 1, else it takes a rectangular shape. Polar shape is advised if the main clump is centered on (0,0).
Default: *int*=0 meaning that the program will used a cartesian grid.

nlentille *int*
Set the number of clumps that defines the Lens Potential. The number of first identifier **potential** must be equal or larger than this number. If the effectively read number of potentials is lower than **nlentille** then **nlentille** is set to the effectively read number of potentials.
Default: *int*=0.

nlens_opt *int*
Set the number of clumps that will be optimized in the **inverse** mode. The number of first identifiers **potential** and **limit** must be equal or larger than this number otherwise **nlens_opt** is set to the number of **limit** identifier read.
Moreover one should have **nlens_opt** $\leq$ **nlentille**.
Default: *int*=0.

nlens_critic *int*
In the SNAKE method to draw the critical lines, set the number of clumps that must be contoured by the algorithm in their order of readding in the `.par` file.

2.2.3 potential

under this identifier is defined one mode of the gravitational potential. One can define a global potential with many modes, for each mode a first identifier "potential" must be defined.

It is allowed to put some comments after the identifier **potential**, for example:
potential Clump cD
or
potential #1

This will clarify the inputfile.

profil *int*

Set the type of profile used to describe a clump.
0: circular singular isothermal sphere.

$$\varphi(r) = 4\pi\frac{\sigma_0^2}{c^2}\frac{D_{LS}}{D_S}.r$$

1: elliptical singular isothermal sphere.

$$\varphi(x,y) = 4\pi\frac{\sigma_0^2}{c^2}\frac{D_{LS}}{D_S}\sqrt{(1-\varepsilon)x^2+(1+\varepsilon)y^2}$$

$$\varphi(r,\theta) = 4\pi\frac{\sigma_0^2}{c^2}\frac{D_{LS}}{D_S}r.\sqrt{1-\varepsilon\cos(2(\theta-\theta_0))}$$

2: circular sphere with a core radius. With profile slope **exponent**.

$$\varphi(r,\theta) = 6\pi\frac{\sigma_0^2}{c^2}\frac{D_{LS}}{D_S}r_0\left[1+(r/r_0)^2\right]^{\alpha}$$

Note: If $\alpha \neq 1/2$, σ_0 does not correspond exactly to the true 3D velocity dispersion (see eq. 3.68 of my Ph.D).

3: elliptical sphere with a core radius. With profile slope **exponent**.

$$\varphi(r,\theta) = 6\pi\frac{\sigma_0^2}{c^2}\frac{D_{LS}}{D_S}r_0\left(1+(r/r_0)^2\left[1-\varepsilon\cos(2(\theta-\theta_0))\right]\right)^{\alpha}$$

Note: If $\alpha \neq 1/2$, σ_0 does not correspond exactly to the true 3D velocity dispersion (see eq. 3.68 of my Ph.D).
α must be greater than 0.

4: elliptical isothermal sphere with a core radius.(cf my Ph.D)

$$\varphi_0 = 6\pi\frac{\sigma_0^2}{c^2}\frac{D_{LS}}{D_S}r_0$$

$$\varphi(r,\theta) = \varphi_0\left[\sqrt{1+(r/r_0)^2}-\varepsilon\frac{(r/r_0)^2}{\sqrt{1+(r/r_0)^2}}\cos(2(\theta-\theta_0))\right]$$

5: Hubble profile... with BUGS. Do not Use!

6: pseudo-elliptical with core-radius and with profile slope for the circular and elliptical part

$$\varphi_0 = 6\pi \frac{\sigma_0^2}{c^2} \frac{D_{LS}}{D_S} r_0$$

$$\varphi(r,\theta) = \varphi_0 \left(\left[1 + (r/r_0)^2\right]^{\alpha} + \varepsilon \frac{(r/r_0)^2}{(1+(r/r_0)^2)^{\beta}} \cos(2(\theta - \theta_0)) \right)$$

Note: If $\alpha \neq 1/2$, σ_0 does not correspond exactly to the true 3D velocity dispersion (see eq. 3.68 of my Ph.D).

7: Point mass.

$$\varphi(r) = \frac{4GM_0}{c^2} \frac{D_{LS}}{D_L D_S} \log r$$

8: PIEMD (See Kassiola and Kovner 1993, ApJ, 417, 450)
The analytic potential is given by:

$$\frac{\partial^2 \Phi}{\partial x^2} = \mathrm{Re}\frac{\partial I^*}{\partial x}, \qquad \frac{\partial^2 \Phi}{\partial y^2} = \mathrm{Im}\frac{\partial I^*}{\partial y}\ , \qquad \frac{\partial^2 \Phi}{\partial x \partial y} = \mathrm{Im}\frac{\partial I^*}{\partial x} = \mathrm{Re}\frac{\partial I^*}{\partial y}\ ,$$

with

$$I^* = \frac{(1-e^2)E_0}{2i\sqrt{e}}\ \ln\Big\{ \frac{\frac{1-e}{1+e}x - i\frac{1+e}{1-e}y + 2i\sqrt{e}\sqrt{r_0^2 + \frac{x^2}{(1+e)^2} + \frac{y^2}{(1-e)^2}}}{(x - iy + 2ir_0\sqrt{e})} \Big\}\ ,$$

where $e = (a-b)/(a+b)$.
In the case of a PIEMD, the ellipticity you give is:

$$\varepsilon = 3\varepsilon_\Sigma = (a^2 - b^2)/(a^2 + b^2)\ .$$

The **v_disp** parameter you give is not E_0. But E_0 is computed form **v_disp** by this way:

$$E_0 = 4\pi \frac{D_{LS}}{D_S} \frac{\sigma_0^2}{c^2} = 6\pi \frac{D_{LS}}{D_S} \frac{\mathbf{v\_disp}^2}{c^2}\ .$$

9: Plane mass.

$$\varphi(r) = \frac{\Sigma_0}{\Sigma_{crit}} \frac{r^2}{2.}$$

12: Navarro, Frenk & White profile.

If we write the 3D mass density

$$\rho(r) = \frac{\rho_c}{\frac{r}{r_0}(1+\frac{r}{r_0})^2}$$

we get the lens potential

$$\varphi(r) = \varphi_0 \times \begin{cases} \ln^2(\frac{r}{2r_0}) + \arccos^2(\frac{r_0}{r}) & \text{if } r \geq r_0 \\ \ln^2(\frac{r}{2r_0}) - \text{argch}^2(\frac{r_0}{r}) & \text{if } r < r_0 \end{cases}$$

with

$$\varphi_0 = 6\pi \frac{D_{LS}}{D_S} \frac{\sigma_0^2}{c^2} \frac{r_0}{2}$$

where we defined

$$\sigma_0^2 = \frac{8}{3} G \rho_c r_0^2$$

which is actually not the central velocity dispersion but a caracteristic one.

The elliticity is introduced in the lens potential replacing r by $r\sqrt{1-\epsilon_\varphi \cos(2(\theta-\theta_0))}$ where ϵ_φ is the ellipticity of the potential. The potential ellipticity is proportional to the surface density ellipticity in the small ellipticities approximation by $\epsilon_\varphi \simeq \epsilon_\Sigma/3$ (cf. Golse & Kneib 2002).

x_centre *float*
Set the x position of the center x_c. In arcseconds.

x_centre_wcs *float*
Same as **x_centre** but gives the position is degree WCS. This keyword needs the presence of the *reference* keyword in the **runmode** Section.

y_centre *float*
Set the y position of the center y_c. In arcseconds.

y_centre_wcs textitfloat
Same as **x_centre_wcs** but for the u position.

masse *float*
Set the point mass M_0 expressed in 10^{12} solar masses, only if **profil**=7.

pmass *float*
Set the mass per surface unit Σ_0 expressed in $g.cm^{-2}$, only if **profil**=9.

ellipticite *float*
Set the ellipticity $\epsilon_\Sigma = \frac{a_\Sigma^2 - b_\Sigma^2}{a_\Sigma^2 + b_\Sigma^2}$ of the mass distribution. In the program ϵ_Σ is converted to ϵ_φ assuming that $\epsilon_\Sigma = 3\epsilon_\varphi$.

ellip_pot *float*
Set the ellipticity of the potential distribution $\epsilon_\varphi = \frac{a_\varphi^2 - b_\varphi^2}{a_\varphi^2 + b_\varphi^2}$.

angle_pos *float*
Set the position angle of the potential distribution θ_0 expressed in degree (90° relative to PA).
It corresponds to the direction of the semi-major axis of the isopotential counted from the horizontal axis, counterclockwise.

core_radius *float*
Set the core radius r_0, expressed in arcseconds.

cut_radius *float*
Set the cut radius r_c, expressed in arcseconds.

v_disp *float*
Set the central velocity dispersion σ_0 of the 3D velocity field (supposed

isotropic). Expressed in kms.
The relation between **v_disp** and the observed line-of-sight velocity dispersion depends on the mass profile (see Wu 1993, ApJ, 411, 413) and the anisotropy factor. For circular or nearly circular isotropic models with isothermal profile,

$$\sigma_{los}(0) = \sqrt{\frac{9}{8}}\sigma_0 = \sqrt{\frac{3}{4}}\sigma_{1D} \ .$$

The observed line-of-sight velocity dispersion is generally obtained from the central galaxies of the cluster and is more or less $\sigma_{los}(0)$. The correcting factor is therefore negligeable. However, for other profile (non isothermal), you have to compute by yourself the correction. The correction is calculated for approximate King profile in Kneib 1993 eq. 3.63
Caution: in case of potential PIEMD, the parameter **v_disp** is **not** the true velocity dispersion (see PIEMD).

exponent *float*
Set the exponent of the slope α of the potential distribution. Isothermal=0.5.
To use with **profil**= 2,3,6.

alpha *float*
Same as exponent.

beta *float*
Set the exponent of the slope β of the elliptical part of the potential distribution. Isothermal=0.5.
To use with **profil**=6.

z_lens *float*
Set the redshift of the clumps. At present, all the clump must be at the same redshift.

2.2.4 limit

under this identifier are defined the constraints on the p otential (more precisely on one mode). This identifier has to follow the identifier of the corresponding "potential" mode. (used only in the invert runmode)

It is advised to put the **limit** identifier just after the **potentiel** identifier (for clarity).
As the **potential** identifier, it is allowed to put comments after the identifier **limit** on the same line.

x_centre *int float1 float2 float3*
Gives limits for the **x_centre** parameters of the deflecting potential, when using the **inverse** mode.
float1 is the minimum.
float2 is the maximum.
float3 is the precision desired on the parameters, be careful it is not a dispersion!
int tells if and how should the optimizer handle the parameters.
0: the optimizer do not change the parameter, and keep the value defined within the **potential** list.
1: consider *float1* and *float2* as strict bounds.
2: consider *float1* and *float2* as soft bounds. If the optimizer find a minimum outside this bounds, he will test it.
3: consider *float1* as a soft bound, *float2* as strict bound.
4: consider *float2* as a soft bound, *float1* as strict bound.
-n: the optimizer will take n different values between *float1* and *float2*, and try to optimize the others parameters. Only 2 parameters can have such limit mode.

y_centre *int float1 float2 float3*
Same thing.

ellipticite *int float1 float2 float3*
Same thing.

angle_pos *int float1 float2 float3*

Same thing.

core_radius *int float1 float2 float3*
Same thing.

cut_radius *int float1 float2 float3*
Same thing.

v_disp *int float1 float2 float3*
Same thing.

exponent *int float1 float2 float3*
Same thing.

alpha *int float1 float2 float3*
Used in models 3,6,12,84,87,88,89.
Same as exponent.

beta *int float1 float2 float3*
Used in models 6 and 89.
Same thing.

psi0 *int float1 float2 float3*
Same thing.**Not implemented**
psi0 is defined for distribution mass model 0,1 by:

$$\mathbf{psi0} = 4\pi \frac{\mathbf{v\_disp}^2}{c^2}$$

for models 2,3,4,6,8,12:

$$\mathbf{psi0} = 6\pi \frac{\mathbf{v\_disp}^2}{c^2} \mathbf{core\_radius}$$

for model 7:

$$\mathbf{psi0} = \frac{4GM}{c^2 D_{OL}} \ .$$

for model 9:

$$\mathbf{psi0} = \frac{4\pi G \mathbf{pmass} D_{OL}}{c^2} \ .$$

b0 *int float1 float2 float3*
Same thing.
b0 is defined for distribution mass model 0,1 by:

$$\mathbf{b0} = 4\pi \frac{\mathbf{v\_disp}^{\ 2}}{c^2}$$

for model 2,3,4,6,8 by:

$$\mathbf{b0} = 6\pi \frac{\mathbf{v\_disp}^{\ 2}}{c^2}$$

for model 7,9: not defined.

Note 1: if you want optimize the point mass model, the parameters you can optimize are **x_centre y_centre** and **psi0**. **psi0** is linked to the central mass via the equation:

$$\mathbf{psi0} = \frac{4GM}{c^2 D_{OL}} \ .$$

For M= 10 (in units of 10^{12} solar masses) at $z_L = 0.3$ ($H0 = 50$, $\Omega_0 = 1$, $\lambda = 0$) we have **psi0**= 35.9

Note 2: If you want optimize the Plane mass model, the parameters you can optimize are **x_centre y_centre** and **psi0**. **psi0** is linked to the central mass via the equation:

$$\mathbf{psi0} = \frac{4\pi G \mathbf{pmass} D_{OL}}{c^2} \ .$$

For Σ_0=.1 $g.cm^{-2}$ at $z_L = 0.3$ ($H0 = 50$, $\Omega_0 = 1$, $\lambda = 0$) we have **psi0**=0.163

Note 3: It is strongly recommended to use **v_disp** in general (expect for the point mass profile: 7).

2.2.5 potfile

under this identifier are defined the default parameters for all the galaxy scale mass components that account for perturbations to the cluster potential by the galaxies. By default, the mass distribution model for the galaxies is PIEMD.

filein *int filename*

If *int*=2, the galaxies catalog. must be in the following format :

int	*float1*	*float2*	*float3*	*float4*	*float5*	*float6*	*float7*	*float8*
$ftype$	x_c	y_c	ε	θ	r_{core}(kpc)	r_{cut}(kpc)	σ(km/s)	z

If *int*=1 or 3 the format must be :

	string	*float1*	*float2*	*float3*	*float*
$Id:$	$inputfile.tex, v1.112008-03-0414:20:07ejulloExp$	x_c	y_c	a	b

If *int*=3, x_c and y_c are given in degrees in the World Coordinate System.
If *int*=1, x_c and y_c are given in arcseconds relative to the reference point given in the **runtime** section.

The ellipticity (ε) parameter is linked to a and b by :

$$\varepsilon = (a^2 - b^2)/(a^2 + b^2) \ .$$

The ellipticity (ε) of the galaxies is then computed again according to their potential type **(**ftype) (cf. 2.2.3).

type *int*

All the potfile galaxies have the same mass profile set by *int*. (See **potential** section). Default value : 81, PIEMD.

In the current version, the *Lum* value is not used.

The dynamical parameters (r_{core},r_{cut},σ) of the potfile galaxies are scaled from the Faber-Jackson and Tully-Fisher scaling relations for elliptical and spiral galaxies, respectively. These scaling laws conserve the mass-to-light radio of the galaxies. The scaling factors are defined below.

mag0 *float*

float is $m^\star$ in the scaling relations below. It can be in absolute or in relative magnitudes according to the magnitude you give in your potfile. $m^\star$ default value is 17 mag.

zlens *float*

All the potfile galaxies with no specified redshift (Catalog format 3) have the same redshift *float*. This is used to compute the D_{OL} diameter angular distance.

sigma *int float1 float2*

float1 is $\sigma_0^\star$ in km/s. The velocity dispersion of the galaxies is given by :

$$\sigma_0 = \sigma_0^\star \ 10^{0.4\frac{m_\star - mag}{\sigma_{slope}}}$$

In the **inverse** 2 optimisation method, *int* set the number of bins for the potfile optimisation in the range (min,max) = (*float1*, *float2*). (see **inverse** section).
In the **inverse** 3 bayesian optimisation method, *int* can be 1 or 3 for the uniform or Gaussian prior. For the uniform prior, (*float1*, *float2*) are the (min, max) limits. For the Gaussian prior, (*float1*, *float2*) are the (mean, stddev) parameters of the Gaussian pdf.
$\sigma_0^\star$ default value : 200 km/s.

core *float*

float is $r_{core}^\star$ in arc seconds. It is used to compute the core radius of the galaxies.

$$r_{core} = r_{core}^\star \ 10^{0.4(m_\star - mag)1/2}$$

corekpc *float*

float is $r_{core}^\star$ in kpc. It is used to compute the core radius in kpc of the galaxie. The cosmological parameters defined in the *cosmology* Section are used to convert from kpc to arc seconds.

$$r_{core}('') \ = \frac{1}{D_{OL}} \frac{c}{H_0} r_{core}(\text{kpc})$$

cut *int float1 float2*

If *int* is true, *float1* is $r^\star_{cut}$ in arc seconds. The cut radius in arc seconds of a galaxy is :

$$r_{cut} = r^\star_{cut} 10^{0.4 \frac{m_\star - mag}{2\ slope}}$$

int and *float2* are used in the potfile optimisation. (see **sigma** keyword and *inverse* section).

cutkpc *int float1 float2*

If *int* is true, *float1* is $r^\star_{cut}$ in kpc and is used to compute the cut radius of the galaxies in kpc. The cosmological parameters defined in the *cosmology* Section are then used to convert from kpc to arc seconds.

$$r_{cut}('') = \frac{1}{D_{OL}} \frac{c}{H_0} r_{cut}(\text{kpc})$$

int and *float2* are used in the potfile optimisation. (see *sigma* keyword and *inverse* section).

slope *int float1 float2*

float1 is the slope value used in the r_{cut} computation.

int and *float2* are used in the potfile optimisation. (see *sigma* keyword and *inverse* section). [Not yet implemented for the bayesian optimisation].

slope default value is 4.

vdslope *int float1 float2*

float1 is the velocity dispersion slope value used in the σ_0 computation.

int and *float2* are used in the potfile optimisation. (see *sigma* keyword and *inverse* section). [Not yet implemented for the bayesian optimisation].

σ_{slope} default value is 4.

2.2.6 cline

under this identifier are defined the parameters to compute the critical and the caustic lines.

nplan *int float float ...*

int defines the number of Source Plane for which will be computed the critical and caustic lines. The *float* arguments give the redshift of these planes.
Results are put in 2 different files: "ce.dat" (external critic and caustic lines), "ci.dat" (internal critic and caustic lines).
"ce.dat" and "ci.dat" are 5-columns ASCII files with the format:
$\{ j\ x_i^I\ y_i^I\ x_i^S\ y_i^S \}$.
With the snake algorithm, j is the line identifier (we can have more than one external or internal lines). $x_i^I\ y_i^I$ are the coordinates of the critical lines. $x_i^S\ y_i^S$ are the coordinates of the corresponding caustic lines.

PICT: here is an example to visualized the external critical line ("ce.dat") and the internal caustic line ("ci.dat").

```
curve
        nfile    2
        namein   1  0  ce.dat
        column   1  5
                 2  3
        namein   2  0  ci.dat
        column   2  5
                 4  5
        end
fini
```

You can also use the *pcl* PERL script.

```
pcl <clean|noclean> <ext|int> <critic|caustic>
```

This script will read the "ce.dat" and "ci.dat" files and display the critical or caustic lines on the currently opened image in DS9.

zonemult *int1 int2 filename*

int1 0: if false 1: if true.
If true and if *int* of **nplan** equal 1, will determine for the redshift *float* of **nplan** the image multiplicity of each pixel (*int2* $\times$ *int2* frame) of the image plane (area defined by **dmax**). Results are written in the pixel-frame file *filename*
If **nplan** not equal 1, **zonemult** is not executed, and a WARNING

is displayed.
The format of *filename* is the 'ipx' simple pixel-frame format.
NOTE : Works only with the SNAKE algorithm.

PICT: The following example of a PICT inputfile allow to display the pixel-frame *filename* with a gray lut starting at zgmin=0 (white) ending at zgmax=6 (black) in a frame of size [x: -20. +20, y:-20. +20]
The position of the pixel-frame is automatically scaled (scale= 1).

```
contour
        filein   1    filename
        format   2
        scale    1
        zgmin    0
        zgmax    6
        end
frame
        dmax     20.
        end
fini
```

dmax *float*
Defines the area (xmin= -*float*, xmax=*float*; ymin=-*float*, ymax=*float*) where the critical lines are search. *float* is expressed in arcseconds. A typical value of **dmax** is 30.
Default: the value defined in **champ**.

algorithm `marchingsquares|snake`
Select one of the two algorithms for the computation of the critical and caustic lines.
The snake algorithm is the original algorithm implemented in LensTool. For each of the first *nlens_crit* clumps of the lens model, the algorithm starts from the centre of the clump and looks for a point on a surrounding critical line (locus of the space where amplification is infinite). Then, it tries to follow this critical line and to go back to its original starting point.

The marching squares algorithm defines a first square with the *dmax* or the *champ* keywords. Then, it divides this first cube in 4 small

squares. According to their size and their amplification values in the centre and the 4 corners, each square is divided or not in 4 further small squares.
If the field is rectangular, the greater value between the width and the height of the field is choosen as the square size.
As long as the size of a square is greater than *limitHigh*, it is automatically divided in 4 small squares. The size of a square cannot be lower than *limitLow*.
The marching squares algorithm is slower than the line-following snake algorithm but gives always the full contour of the critical lines. It is less sensitive to small irregularities in the contour. The snake algorithm always returns a connected contour.
Default algorithm is `marchingsquares`.

pas *float*
For the line-following algorithm :
Defines the step between each search *i.e.* it represents the typical distance in arcsecs between each point of the external critical lines. For internal critical lines half this value is taken.
To improve the definition of the critic and caustic lines, use smaller values such as 0.5" or even 0.2".
Default is 1."

limitLow *float*
For the marching cube algorithm :
Defines the smaller size of a square *i.e.* the size of a square is compared to this value to decide between dividing again in 4 squares or stopping the division.
To improve the definition of the critic and caustic lines, use smaller values such as 0.5" or even 0.2". This implies more computation time.
Default is 1".

limitHigh *float*
For the marching cube algorithm :
Defines the higher size of a square *i.e.* the size of a square cannot be higher than this value. A square with a size above this value is automatically divided in 4 squares.
Decrease this value to remove holes in the critical lines and improve

the detection of critical lines around isolated galaxies. This implies more computation time.
Default is 10".

2.2.7 grande

under this identifier is defined the way to represent th e computed deformation of objects.

large_dist *float*
float set the value for which we can consider we have a strong deformation (it corresponds to a minimum value of τ_I). Typical value is 1. or 2.

profil *int1 int2*
int1 0: if false 1: if true.
If true, set the representation mode of large distorted source object to a density points where *int2* is the number of points. A Gaussian profile for the source is assumed.
Results is a list of points in the image plane that are stored in the ASCII file: *gianti.dat*. The **profil** keyword has no effect if the **contour** keyword is true.

PICT : The gianti.dat file can be displayed on ds9 with the *gianti* perl script.

contour *int1 int2*
int1 0: if false n: if true.
If true, set the representation mode of large distorted source object to contour points. If non zero, *int1* set the number of isocontours for the source. *int2* set the number of points per isocontour.
Results is a list of points in the image plance that are stored in the ASCII file: *gianti.dat*.

PICT : The gianti.dat file can be displayed on ds9 with the *gianti* perl script.

PICT: The following example of a PICT inputfile allow to display the gianti.dat file.

```
curve
        nfile     1
        namein    1   1   gianti.dat
        format    1   1
        end
fini
```

Note that the second sinteger in namein, corresponds to the way to trace the list of points:
0 is a line
n¿0 individual points (the value of n sets the type of points)
n¡0 individual points and a line (the value of n sets the type of points)

iso *int1 int2 float1 float2 float3*

int1 0: if false 1,2: if true.
If true set the representation of large distorted object to an Image mode.
If *int1*= 1, set the initial window to the minimum size that include the position of the center of images, and their computed sizes.
If *int1*= 2, set the initial window to the one defined in **champ**.
int2 is the maximal number of pixel tolerate for the final image.
float1 is the pixel size desired for the image.
float2 and *float3* are extension factors, from the initial window. (a value of 0.5 will add half the total size to both size.)
Note: contrary to **pixel**, **iso** make gravitational images of only one source. The **iso** keyword has no effect if any of the **contour** or the **profil** keywords are true.

name *filename*

Generic name for the final image, when using **iso** mode. It write for each arclets the results in the pixel-frame file *filename***n**, where n is the index of the arclet.
Default value of *filename* is *giant*

PICT: The following example of a PICT inputfile allow to display the pixel-frame *filename***n** with a gray lut starting at zgmin=0 (white) ending at zgmax=50 (black) in a frame of size [x: -20. +20, y:-20. +20]

The position of the pixel-frame is automatically scaled (scale= 1).

contour
filein **1** ***filename***
format **2**
scale **1**
zgmin **0.**
zgmax **50.**
end
frame
dmax **20.**
end
fini

vitesse *int*

int 0: if false 1: if true.

If true, the profile of the source will not be taken as a gaussian profile but will have a velocity profile. The intensity will be proportional to the distance of the small axes of the ellipse.

The main purpose of **vitesse** is to look at possible velocity gradient along giant arcs due to internal velocity field of the lensed galaxy.

2.2.8 observ

under this identifier is defined the different noise that t can be add to a gravitational image, such as seeing or Poisson Noise.

All these constants are used when the image mode **grande iso** or **runmode pixel** representation is set.

bruit *int*

int 0: if false 1: if true.

If true add Poisson noise to the final image.

sky *float*

mean value for the sky background.

idum *int*

int random number, should be negative.

dispersion *float*
1 σ value of the background.

prec *float*
float defines the precision for the calculation of the value of each pixel, 0.1 is a typical value.

seeing *int float*
int 0: if false 1: if true.
If true will convolve the image by a Gaussian filter with a full width at half maximum of size *float* expressed in arcseconds.
Note: for good accuracy the pixel size of the image must be small compared to the seeing, a minimum of 2 pixels, but with 4 pixels as an optimum.

binning *int1 int2*
int 0: if false 1: if true.
If true will "bin" the image by *int2* pixels.
Note: If the binning is set is done just after the seeing and before to add noise. The aim is to better calculate the seeing when the sampling of final image is low.

2.2.9 cosmologie

under this identifier are defined the cosmological parameters Ω_0, λ *and* H_0.

H0 *float*
float defines the value of H0 in Mpc/km/s.
Default value is $H0 = 50$.

omega *float*
float defines the value of Ω_0.
Default value is $\Omega_0 = 1$.

lambda *float*
float defines the normalized value of λ. (for a flat universe $\Omega_0 + \lambda = 1$.)
Default value $\lambda = 0$.

2.2.10 cosmolimit

Under this identifier are defined the limits on the cosmological parameters that you want to optimise during the Bayesian optimisation.

omega or **omegaM** *int float1 float2*
Gives limits for the Ω_M parameter.
int set the prior pdf. 1: Uniform, 3: Gaussian.
With a Uniforma prior, *float1* and *float2* are the lower and upper limits.
With a Gaussian prior, *float1* and *float2* are the mean and stddev parameters of the Gaussian pdf.

omegaX or **lambda** *int float1 float2*
Gives limits for the Λ parameter.
int set the prior pdf. 1: Uniform, 3: Gaussian.
With a Uniforma prior, *float1* and *float2* are the lower and upper limits.
With a Gaussian prior, *float1* and *float2* are the mean and stddev parameters of the Gaussian pdf.

wX *int float1 float2*
Gives limits for the w_X parameter.
int set the prior pdf. 1: Uniform, 3: Gaussian.
With a Uniforma prior, *float1* and *float2* are the lower and upper limits.
With a Gaussian prior, *float1* and *float2* are the mean and stddev parameters of the Gaussian pdf.

2.2.11 champ

under this identifier is defined the size of the field use d in some calculations such as the dimension of the grid for the inversion of the lens equation.

xmin *float*
float give the minimal value in X (in arcseconds) of the frame where computation are made.

xmax *float*
float give the maximal value in X (in arcseconds) of the frame where computation are made.

ymin *float*
float give the minimal value in Y (in arcseconds) of the frame where computation are made.

ymax *float*
float give the maximal value in Y (in arcseconds) of the frame where computation are made.

dmax *float*
Quick definition of the 4 limits (in arcseconds): **xmin**=-*float*, **xmax**=*float*, **ymin**=-*float* and **ymax**=*float*.

2.2.12 cleanlens

under this identifier are defined some parameters to retrieve the shape of the source knowing a pixel-frame of the image.

Here the program reads real CCD pixel-frame and using the equation of the lens compute a pixel-frame of the source. For each point of the Image plane the program can compute the corresponding point in the Source Plane. Then for each pixel with multiplicity n_{ij} of the source plane we then can attribute an intensity computed as the mean of the intensity of corresponding

image pixels:

$$\mathcal{I}^S_{ij} = \frac{1}{n_{ij}} \Sigma_{k=1}^{k=n_{ij}} \mathcal{I}^I_{ijk}$$

The error of this reconstruction is given at position ij in the source plane by:

$$e_{ij} = (n_{ij} - 1) \Sigma_{k=1}^{k=n_{ij}} (\mathcal{I}^I_{ijk} - \mathcal{I}^S_{ij})^2$$

(the error estimate exists only if we have multiple images).

When using the **inverse** mode, the program will minimize the estimate of the error ($e = \Sigma e_{ij}$). This method nevertheless numerically different, was first described by Kochaneck *et al.* 1989.

cleanset *int float*

int 0: if false 1: if true.

If true the program will compute from the image pixel-frame **imframe** the corresponding source frame **sframe** at redshift *float*.

If the inverse mode is selected see **runmode inverse** an optimization of the 'ring cycle' form will be done. In this case the optimization will work only if multiples images or giant arcs are present in the image pixel-frame. It is strongly advise the user to have an a priori good guess on the mass model parameters.

Moreover it will create the pixel-frame:

- *erreur.ipx* : this frame is relevant only in the case of giant or multiple arcs, it compute the reconstruction error of the source pixel-frame for each pixels. - *imult.ipx* : this pixel-frame show the multiplicity of each pixel of the reconstructed source pixel-frame.

c_image *filename*

If *int* = 1 for the **cleanset** keyword, *filename* is an ASCII file that contains a list of points i,, x_i, y_i in the image plane given in arcsec relative to the image reference points (see *reference* keyword in *runmode* section) that delimit the center of the observed image. The barycenter of those points is sent to the source plane and defines the source center. This value is used to compute the WCS keywords of the resulting source FITS file.

imframe *int filename*

int format of the input image (*int*= 2 for ipx format, 3 for fits file).

filename name of the CCD frame where (multiple) gravitational images are present.

psfframe *int filename*
int format of the input Point Spread Function (PSF)(*int*= 2 for ipx format, 3 for fits file).
filename name of the PSF frame, namely a Star Profile. The Star must be at the center of the frame. And the total intensity of the frame (sum of all the pixels) must be equal to 1 (PSF normalized).
Note: the psf frame is used when using a deconvolve-inversion of the lensed images. (not working yet)

sframe *filename*
filename name of the output source frame.
This frame will correspond to the inversion of the **imframe** input frame (from Image Plane to Source Plane).
It is written with the ipx format.

ncont *int filename*
int number of contour.
filename name of the output frame: that is the CCD frame where only the pixel inside the contours (closed lines) have been kept.
The idea is to limit the area of the frame, and keep only the interesting pixels.

contour *int filename*
Define contours in the image plane of one or several images of the source you want the shape in the source plane. *int* index of the contour for one image. First contour must be index by 1, second by 2, etc. *filename* is the name of the contour ASCII file for one image that contains a list of points i,, x_i, y_i in the image plane given in arcsec relative to the image reference points (see *reference* keyword in *runmode* section).

echant *int*
It is possible to subsample the CCD frame to calculate with greater accuracy the source frame in the source plane. *int* is the subsampling parameter. Default is 1. If *int*= 2 it will cut each pixel in 4 smaller

pixel. Default value : 2.

s_echant *int*
It is possible to subsample the frame in the source plane as you can do in the image plane with the *echant* keyword. If *int* is too high, you can get an image with a undefined pixel values. Default value : 1.

s_n *int*
Define the width and height of the resulting source image in pixels. Default value : 50 pixels.

The following identifier should be set if the format of the **imframe** is 0 (ascii format) or 1 (ipx format without scaling). It is strongly advised to used the ipx format: 2 or fits format: 3, to avoid to define such identifier.

pixel *float*
set size of the pixel in x and y in arcseconds.

column *int*
column to be selected (in a multi-column ascii pixel-frame file).

header *int*
number of line to skip before the real beginning of the data.

pixelx *float*
pixel size in x in arcseconds (if the pixel is not a square).

pixely *float*
pixel size in y in arcseconds (if the pixel is not a square).

xmin *float*
x position of the bottom-left pixel (expressed in pixel units).

ymin *float*
y position of the bottom-left pixel (expressed in pixel units).

2.2.13 image

under this identifier are defined some characteristics of images, multiple images or arclets.

multfile *int filename*
int 0: if false, 1: if true.
If true will used the multiple images defined in *filename* to optimize the lens parameters and/or the unknown redshift(s) of multiple images.
The format of *filename* is an 'ellipse'-like format:
$\{ i\ x_{ij}\ y_{ij}\ a_{ij}\ b_{ij}\ \theta_{ij}\ z_i\ mag\}$
where i is an identifier for all the multiple images of a single source at redshift z_i. (see chapter datafile). θ_{ij} is 90° relative to PA. mag is the magnitude of the image. It is used in the flux optimisation mode.
In the same file the user can put different families of multiple images. Here is an example of 2 families of multiple images:
C1a 20.5 30.4 1.2 0.6 40. 0.65 0.
C1b 30.8 20.3 1.3 0.6 50. 0.65 0.
C1c 25.1 25.2 1.2 0.5 60. 0.65 0.
C2a 10.3 -5.2 1.2 0.9 10. 0. 0.
C2b -9.7 -4.2 1.4 0.5 20. 0. 0.

The redshift of the second family of multiple images in this example is not known, therefore the redshift is set to 0 and its convergence can be constraint by the **z_m_limit** identifier (see below).

mult_wcs *int*
If true, the multiple images coordinates are considered absolute WCS coordinates. They are transformed to relative coordinates with the *reference* keyword position given in the **runmode** section.
If false, their coordinates are considered relative in arcsec.

forme *int*

int 0: if false, 1: if true.
If true will used both the position and the ellipticities of the **multfile** as constraints. If false will only used the position of the **multfile** as constraints. In both cases, optimisation is done in the source plane.
Default value is -1 for the optimisation in the image plane.

z_m_limit *int1 imageId int3 float1 float2 float3*

int1 0: if false 1: if true.
If true will optimize the redshift of the multiple images of index *imageId* corresponds any of image identifiers of a given source given in the **multfile** filename (ex: C2b or A1).

int3 determines the properties of the boundaries
In parabolic optimisation, its meaning is : 1:strict, 2:soft, 3: left soft right strict, 4: right soft left strict, -n: sampling.
In Bayesian optimisation, its meaning is : 1: uniform prior, 3: Gaussian prior.

float1 lowest boundary or mean value (Gaussian prior).

float2 highest boundary or sttdev value (Gaussian prior).

float3 precision to reach to stop the optimization of the redshift. (Not considered in Bayesian optimisation).

This identifiers is very similar to the second identifiers of **limit**.

arcletstat *int1 int2 filename*

int1 0: if false, 1: if true.
If true will used the arclets in *filename* to optimize the lens parameters assuming all the sources at the redshift defined in **source** first identifier (See sect. 2.2.12). The format of the catalogue supplied in *filename* depends on the value of *int2*. If *int2* 0: it is the same as the arclet catalogue referred to in section 2.2.1 (i.e. an ASCII column format of the form: $\{ i\ x_i\ y_i\ a_i\ b_i\ \theta_i\ z_i \}$, where θ_i is associated with the ellipses' major axis, is in degrees and is measured anticlockwise from West, and x and y are RA and Dec in decimal degrees). If it is *int2* $= 2$, it needs 2 extra columns $vare1_i\ vare2_i$, with the shape measurement variance on $e1$ and $e2$. Note that the redshifts of the sources are defined in this

catalogue – it is up to the user to provide sensible estimates for them.

int1 is actually more than just a switch: it defines the form of the likelihood used in the optimisation. The recommended mode is 6 – which is to use a likelihood based on the assumed distribution of intrinsic source plane complex ellipticities.

sigell *int1 float int2*
If true, the width of the (assumed Gaussian) intrinsic ellipticity distribution can be specified here (the default value is 0.3). The sum of its square enters and eventually a shape measurement variance constitutes the denominator of the χ^2 calculation.

z_arclet *float*
In the case of **arcletstat** it is possible to fix the redshift *float* of the arclets with unknown redshift (redshift value set to 0 in the **arcletstat** *filename*).

z_a_limit *int1 float1float2*
If *int1* : 1, assumes the arclets with unknown redshift in **arcletstat** *filename* (redshift set to 0), needs to be optimized with a flat prior with min and max boundaries *float1* and *float2*. If *int1* : 3, assumes a Gaussian prior with mean and width *float1* and*float2*. *int1* : 0, if no optimization.

critic *int1 float1 float2 float3 float4 float5*
int1 0: if false 1: if true.
If true will add the constraint of the position of the break (locus of merging images) and the orientation of the image at the break (i.e. the direction of amplification matrix).
float1 x position (in arcseconds) of the break.
float2 y position (in arcseconds) of the break.
float3 direction of the orientation of the image at the break point, expressed in degree from the horizontal line, counter-clockwise.
float4 error of position (in arcseconds) of the break along the arc.
float5 redshift of the source of the merging images.

Note: it is possible to give more than one such a constraint, from merging points at different position and different redshifts. One has simply to enter as many **critic** lines as merging points.

2.2.14 source

grid *int*
int 0: if false 1: if true.
If true the sources are placed on a regular grid in the source Plane.

random *int*
seed for the random number generator, better if negative.

n_source *int*
Number of sources to draw.

elip_max *float*
Maximum ellipticity of the drawing sources. The ellipticity of the sources is drawn from 0 to *float* with a uniform law.

dist_z *int*
int 0: if false 1: if true.
If true will draw the redshift with a uniform law between **z_source** and 2**z_source** .

z_source *float*
redshift of the sources.

taille *float*
set the size of the source in arcsecond.

2.2.15 fini

2.3 Examples

2.3.1 Typical configurations of Arcs

2.3.2 Optimization with one multiple image

2.3.3 Optimization with two multiple images

2.3.4 Optimization with arclet data

The directory `lenstool/example_with_arcletstat` contains a simulated weak lensing catalogue, `nfw_arclet.cat`, in standard form. The model used to create this was an elliptical NFW potential, with true parameters as follows: $M_{200} = 2.0 \times 10^{15} M_{\odot}$, $c_{200} = 7.0$, $|\epsilon_{\psi}| = 0.15$, $\theta_{\psi} = 60.0$. The origin of the catalogue coordinate system is at the centre of the lens potential. The source galaxy positions are the same as those in the central 5 arcmin of the MS0451 catalogue of Smith et al 2008 (in prep); their ellipticity components were drawn from a Gaussian distribution of width 0.25, and then transformed using the expressions of Seitz & Schneider (2002). Gaussian shape estimation noise was then added to each ellipticity component, assuming an rms of 0.2.

This example comes with two parameter files, `nfw.par` and `gnfw.par`, for fitting the NFW and gNFW profile models respectively to the same shear data. This is a situation that arises very fequently in data analysis; the script `run.csh` was written to illustrate the recommended way of running LensTool, i.e. in a directory for each fit (where the fit is defined by the parameter file). You can try out the two model fits by executing
`run.csh` nfw
and `run.csh` gnfw
Examples of typical output files are given in the `ref_nfw` and `ref_gnfw` directories for comparison.
TODO: Dave, Eric: we need to test these models to our satisfaction, and then check in the example results into the ref directories for the users.

Chapter 3

Datafiles

3.1 Input Datafiles

3.1.1 WCS header

In every input file, it is possible to set a WCS header to define the coordinates of the objects in the file.
#REFERENCE int RA DEC
If *int* = 0 : the positions are in degree WCS aligned.
If *int* = 1 : the positions are in arcsec relative to (RA,DEC) expressed in sexagesimal format (HH:MM:SS DD:MM:SS).
If *int* = 2 : the positions are in pixels relative to the reference pixels. In this case, (RA,DEC) defines the coordinates (X,Y) of the reference pixel.
If *int* = 3 : the positions are in arcsec relative to (RA,DEC) expressed in degrees.

3.1.2 Object file

Definition

This ascii file contains a list of objects characterized by their position, shape parameters and redshift, with the following format.

There are 2 formats. The default one specified by the first argument in the parameter file *int1* = 1 is

int float1 float2 float3 float4 float5 float6

int is a characteristic integer that defines the object.

float1 is the X position of the object expressed in arcsec or in degree.

float2 is the Y position of the object expressed in arcsec or in degree.

float3 is the semi-major-axis a of the equivalent ellipse of the object, expressed in arcsecond.

float4 is the semi-minor-axis b of the equivalent ellipse of the object, expressed in arcsecond.

float5 is the position-angle θ of the equivalent ellipse of the object, expressed in degree. This give the orientation of the semi-major-axis from the horizontal line (counter-clockwise).

float6 is the redshift z of the object. If the redshift is unknown this value should be set to 0.

float7 is the magnitude of the object.

The alternate one specified by the first argument *int1* = 2 (see below) is useful in conjunction with keyword arcletstat for weak lensing. It is similar to the default format, with the 2 extra columns

float8 is the variance of the E1 ellipticity component (ellipticity defined as $E = a^2 - b^2/a^2 + b^2$.)

float9 is the variance of the E2 ellipticity component.

Uses

Such a file can characterize either an arclet file or a source file. Here is the list of identifiers which requires such a file:

runmode image if $z = 0$ then the program will use the one defined by **source z_source** .

runmode source if $z = 0$ then the program will use the one defined by **source z_source** .

runmode study the value of z is not used under this identifiers.

image multfile for each image in each set of multiple images the characteristic integer and the redshift (0 if unknown) must be the same.

image arcletstat

3.1.3 Marker file

This ascii file contains a list of points, with the following format:

int float1 float2

int is a characteristic integer that defines the marker.

float1 is the X position of the marker expressed in arcsec.

float2 is the Y position of the marker expressed in arcsec.

Uses

Such a file can characterize only markers that are in the Image Plane. There is only one identifier that uses such a file. That is:
runmode marker

3.1.4 IPX pixel-image file

The IPX format is a simple format for pixel-images. It is made of an ASCII header of 4 lines, followed by the data, that can be written either in ASCII or in binary.

The header is defined in this way:

2 xmin xmax ymin ymax
nx
ny
type mode nature
comments

2 stands for Dimension 2.
xmin, xmax, ymin, ymax are floats value defining the scaling of the image. The center of the bottom-left pixel is (xmin,ymin), the center of the upper-right pixel is (xmax,ymax).
nx is the dimension in x, ny in y.
type can be *int* or *float* or *double*, it is the type of the pixel-image data.
mode can be *txt* for an ASCII representation of the data, or *bin* for a binary representation.
nature is either *real* or *complex*.
comments is at maximum a 1024 long ASCII chain, ended by an EOL character. If mode is set to *txt* the data is listed in the file as a column. If it is set to *bin* the data can be read line by line.

3.1.5 FITS pixel-image file

The program can read FITS pixel-frame. It will read the FITS file and convert the data in float (whatever was the type of the data in the file).

The pixel-frame has to be scaled.

This is done in FITS by modifying the following keywords:
CRVAL2 = x-value of pixel CRPIX2

CRPIX2 = index i of the reference pixel
CDELT2 = pixel size in the x direction
CRVAL1 = y-value of pixel CRPIX1
CRPIX1 = index j of the reference pixel
CDELT1 = pixel size in the y direction

From this values the program compute the xmin,xmax,ymin,ymax in this way:

xmin = CRVAL2 + (CRPIX2-1)*CDELT2
xmax = CRVAL2 + (nx - CRPIX2)*CDELT2
ymin = CRVAL1 + (CRPIX1-1)*CDELT1
ymax = CRVAL1 + (ny - CRPIX1)*CDELT1

3.2 Output Datafiles

ngrille can generate a lot of output file, depending on what was given in the inputfile. It generates always a "mouchard" file *para.out* . Where we can find back the different identifiers with their values, plus others computed constants.

The following subsections give the complete list of the output file that can be created by **ngrille** with their format.

3.2.1 Potential file

pot.dat

This ascii file is written with the following format:

int float1 float2 float3 float4 float5 float6

int is a characteristic integer that defines the potential clump.

float1 is the X position of the center of the clump expressed in arcsec.

float2 is the X position of the center of the clump expressed in arcsec.

float3 is the semi-major-axis a of the ellipse of the line, expressed in arcsecond.

float4 is the semi-minor-axis b of the ellipse of the line, expressed in arcsecond.

float5 is the position-angle θ of the ellipse of the line, expressed in degree. This give the orientation of the semi-major-axis from the horizontal line (counter-clockwise).

float6 is a non-significative constant, in general 0.

The lines represented are the core_radius if any with the ellipticity and the orientation of the mass distribution, and the tangential critical line of the clump if it was alone (analytic expression). Furthermore a cross indicates the center position.

3.2.2 Source file

source.dat

This ascii file is written with the following format (Object file format - see Inputfile -):

int	*float1*	*float2*	*float3*	*float4*	*float5*	*float6*	*float7*
n	x	y	a	b	θ	z	mag

3.2.3 Arclet files

The following ascii file are written with the following format (Object file format - see Inputfile -):

int	*float1*	*float2*	*float3*	*float4*	*float5*	*float6*	*float7*
n	x	y	a	b	θ	z	mag

image.dat list the images (arclets), but the arclets with distortion larger than **large_dist** are not included.

image.all list all images (arclets) with no restriction.

sort.dat list all images (arclets) sorted from high to low distortion.

dist.dat

This file list some images properties with the following format:

int	*float1*	*float2*	*float3*	*float4*	*float5*	*float6*	*float7*	*float8*

int is the characteristic integer that defines the object.
float1 is the X position of the object expressed in arcsec.
float2 is the Y position of the object expressed in arcsec.
float3 is the distance of the arclet from the center of the first clump.
float4 is the axis-ratio b/a of the equivalent ellipse of the object.
float5 is the ellipticity ε of the equivalent ellipse of the object.

float6 is the deformation τ of the equivalent ellipse of the object.
float7 is the amplification μ at the center of the arclet.
float8 is the time-delay τ_d (in year) at the center of the arclet.

gianti.dat

File with the list of points that defines the shape of the arc or arclet when using the second identifiers **profil** or **contour**. The format consist of 4 lines of header and then the data:
float1 float2

float1 give the X position of the point.
float2 give the Y position of the point.

"giant"."n"

Array file of the image of the arc or arclet when using the second identifiers **iso**. The file consist of a header and then the date. In the header one can find the number of lines, the number of columns and the xmin, xmax, ymin, ymax of the surface covered by the array.

3.2.4 Critical and caustic lines files

ce.dat *ci.dat*

These 2 files have the same format, the first one *ce.dat* lists the external critical and caustic lines ([+,+]-[+,-] transition), the last one *ci .dat* lists the internal ones ([+,-]-[-,-] transition).

int *float1* *float2* *float3* *float 4*

int enumerates the different lines
float1 give the X position of the critical point.
float2 give the Y position of the critical point.
float3 give the X position of the correspondent caustic point.
float4 give the Y position of the correspondent caustic point.

cr_an.dat

This file is created if there is only one clump for the mass distribution, it is an estimate of the critical lines (external and internal). They are approxi-

mated by ellipse, hence they are put in an 'ellipse'-like format:

int *float1* *float2* *float3* *float4* *float5* *float6*

int non relevant value (1).

float1 is the X position of the center of ellipse expressed in arcsec.

float2 is the Y position of the center of ellipse expressed in arcsec.

float3 is the semi-major-axis a of the equivalent ellipse of the critical line, expressed in arcsecond.

float4 is the semi-minor-axis b of the equivalent ellipse of the critical line, expressed in arcsecond.

float5 is the position-angle θ of the equivalent ellipse of the critical line, expressed in degree. This give the orientation of the semi-major-axis from the horizontal line (counter-clockwise).

float6 non relevant value (0.).

The first line is the external critical line, the second line the internal critical line, both for a Source Plane at redshift **source z_source**.

3.2.5 Source marker file

marker_s.dat

Same format as the "Marker file".

3.2.6 Prop files

"prop".dat

3.2.7 Invert files

map.iso

map.res

3.2.8 Best file

best.par

bestopt.par

3.2.9 Bayesian optimisation

You can read the bayes.dat files with the **Histogram** and **Histogram2D** tools. They plot 1D and 2D histograms of the samples distribution and give an estimate of the 1 or 2 dimensional marginalised distribution. Those tools have no arguments.

Chapter 4

Getting Started

Initialization

• Visualize the image of the cluster and choose the origin of the Image Plane. For example you can choose it at the barycenter of the central galaxies light. Then scale your image to this position and give to the pixel-size its real size in arcseconds.

• Note the positions (x,y), the orientation and the ellipticities of all the galaxies you plan to use for the modeling. You can add galaxies that you will not use as deflectors, but only to recognize your field in your figures. These data will be included in the file *.par that you generate at the beginning, before starting the runmode.

• Note the positions, the orientations and the ellipticities of all the gravitational images (arcs or arclets) you plan to use for the modeling. Note carefully, those that come from a same source. In case of giant arcs in which you can readily see the double (fold) or triple (cusp) component, you can consider small ellipses within the arcs as multiple images also.

Optimization

• multiple images optimization.

• arclets optimization.

• break constraints: if images are good, try to find the position where im-

ages are merging, if any. You must note the position, the direction of the orientation of the image at the break (merging point), the error of position at the break.

• cleanlens methods

PICT visualization

The visualization of the CCD image and the parameters can be obtained with PICT, CPICT, or SAOIMAGE for instance. These environment give the orientation and the ellipticities of any objects in the field interactively.

Once the parameters are inserted in files, you are ready to play with ngrille.

```
runmode
        arclet      1                       arc.input
        source      0                       source.input
        time        0       100     0.7     time.output
        mass        0       100     0.7     mass.output
        ampli       0       100     0.7     ampli.output
        shear       0       100     0.7     shear.output
        shearfield  0       0.7             shearfield.output
        study       0                       arc.input
        imseeing    1.
        grille      0       54      1.
        inverse     0       300
        minchi0     1.
        prop        0       0.7             prop.output
        pixel       0       200             pixel.output
        marker      0       0.7             marker.input
        profil      0       0.7     30.
        end
grille
        nombre      40
        polaire     0
        nlentille   2
        nlens_opt   1
        end
potential
        profil      3
        x_centre    0.
        y_centre    0.
        masse       0.
        ellipticite 0.3
        ellip_pot   0.1
        angle_pos   30.
        core_radius 5.
        v_disp      1100.
        exponent    0.5
        beta        0.
        z_lens      0.7
        end
```

```
limit
        x_centre        1       -10.    10.     .1
        y_centre        1       -10.    10.     .1
        ellipticite     1       0.1     0.3     .01
        angle_pos       1       -30.5   30.5    5.0
        core_radius     1       2.0     15.0    1.0
        v_disp          1       1000.   1300.   5.
        psi0            0       0.      0.      0.
        b0              0       0.      0.      0.
        beta            0       0.      0.      0.
        ct              0       0.      0.      0.
        exponent        1       0.4     0.6     0.02
        end
potential
        profil          3
        x_centre        10.
        y_centre        20.
        masse           0.
        ellipticite     0.1
        ellip_pot       0.03
        angle_pos       10.
        core_radius     2.
        v_disp          600.
        exponent        0.5
        beta            0.
        z_lens          0.7
        end
limit
        x_centre        0       -10.    10.     .1
        y_centre        0       -10.    10.     .1
        ellipticite     0       0.1     0.3     .01
        angle_pos       0       -30.5   30.5    5.0
        core_radius     0       2.0     15.0    1.0
        v_disp          0       1000.   1300.   5.
        psi0            0       0.      0.      0.
        b0              0       0.      0.      0.
        beta            0       0.      0.      0.
        ct              0       0.      0.      0.
        exponent        0       0.4     0.6     0.02
        end
```

```
cline
        nplan       1       0.7
        zonemult    0       200                     zonemult.output
        dmax        30.
        pas         1.5
        end
grande
        large_dist  1.
        profil      0       200
        contour     0       3
        iso         0       200     0.25    .2    .2
        name                                        iso.output
        vitesse     0
        end
observ
        bruit       0
        sky         2000.
        idum        -1
        dispersion  3.
        prec        0.1
        seeing      0       0.7
        binning     0       2
        end
cosmologie
        H0          100.
        omega       0.5
        lambda      0.5
        end
champ
        xmin        -40.
        xmax        40.
        ymin        -40.
        ymax        40.
        dmax        40.
        end
```

```
cleanlens
        cleanset    0     0.7
        imframe     3                              imframe.fits
        pssframe    3                              pssframe.fits
        sframe                                     sframe.ipx
        ncont       1                              contour.ipx
        contour     1                              cont1.input
        echant      2
        pixel       0.2
        column      215
        header      4
        pixelx      0.2
        pixely      0.2
        xmin        200
        ymin        200
        end
image
        multfile    0                              multfile.input
        arcletstat  0     1                        arcletstat.input
        z_m_limit   0     0     1     0.5   0.9    0.02
        forme       1
        z_arclet    0     0.7
        critic      1     10.   -10.  30.   0.2    0.7
        end
source
        grid        0
        random      -1
        n_source    20
        elip_max    0.7
        dist_z      0
        z_source    0.7
        taille      1.5
        end
fini
```

4.1 Cleanlens section example

```
cleanlens
        cleanset  1 2.6250        # source redshift
        imframe   3 a68r.fits     # FITS file containing the images
        c_image   C4.dat          # region surrounding the image center
        sframe    sourceC4.fits   # result image
        ncont 1 cleanarc.fits     # extracted image
        contour 1 C4_contour.dat  # region surrounding one image
        s_n     200               # width and height of the result FITS file
        echant  4                 # subsampling in the image plane
        s_echant 4                # subsampling in the source plane
        end
```

4.2 Potfile section example

```
potfile
        type   81              # PIEMD profile for all galaxies
        filein 3 cgalK1.mabs   # galaxy scale clumps file in WCS
        zlens 0.255            # redshift of the galaxies
        mag0 15.16398          # apparent m0
#        mag0    -24.88         # or absolute m0
        sigma   5 135. 180.    # sampling, min and max for sigma0
        corekpc .09            # core radius in kpc
        cutkpc  5 25. 55.      # sampling, min and max for cut0
        slope   0 4. 4.0       # not opt., min and max values
        vdslope   0 4 2.5      # not opt., min and max values
        end
```