SaWLens 1.4: A practical manual

Julian Merten
ITA Heidelberg
INAF-OA Bologna

November 11, 2009

This document is meant to provide a practical guideline to the installation and usage of the SaWLens 1.4 package, a state-of-the-art lensing reconstruction algorithm. What this document does not provide is a full documentation of the source code. If you are looking for this, please refer to the according Doxygen documentation of the library headers.
After giving some general information about the code, like system requirements, where to obtain and how to install it, we will describe the three main parts of the reconstruction package since the reconstruction process of a lensing survey with our method is threefold.
First, we will describe the routine which converts the output of a weak lensing shape analysis and strong lensing position information into a special format used by the reconstruction routine, that is described afterwards. In the end we describe a last high-resolution reconstruction step, which is mostly relying on strong-lensing constraints. By intention we keep all the three steps in separated routines to make the package as flexible as possible.
If you want to have a proper description of the underlying physics and numerics of SaWLens, pleaser refer to arXiv:0806.1967.

Contents

1 General information

1.1 Obtaining the code

The source code is available in the ITA repository:

`https://www.ita.uni-heidelberg.de/svn/sawlens`

For downloading the code, you should have a current version of the version-control software `subversion` [1], which is available for Linux and Mac. I will not talk about any Redmond support.
Please note that you need to have an ITA account to obtain the source. If this is not the case, contact the author directly via:

`jmerten@ita.uni-heidelberg.de`

1.2 System requirements

SaWLens was designed to run on parallel multiprocessor machines. It can be executed on single processor machines, but the runtime for high-resolution reconstructions will be very long. The single processor option is mainly for testing purposes. For a high-resolution reconstruction you should run the code on a machine with at least 10 cores available.
Compared to CPU power, the memory consumption of the code is quite moderate. Depending on the number of background galaxies for the weak lensing analysis, you should make sure that the field-preparation routine has at least 2 GB of main memory per core available. The reconstruction gets along with around 1 GB per core on a reconstruction resolution up to 75x75 pixels.

[1] http://subversion.tigris.org/

The code is written in `C++` so you need a working compiler installation for this programming language. Compilation is tested using the `GNU C(++) compiler`[2]. Please note that you will furthermore need a working MPI runtime environment like `MPICH`. [3] The code makes use of several external libraries, which should be available in an actual version on your machine:

- GNU Scientific Library (GSL)[4]
- Message Passing Interface (MPI)[5]
- Automatically Tuned Linear Algebra Software (ATLAS)[6]
- Linear Algebra PACKage (LAPACK)[7]
- CFITSIO[8]
- CCfits[9]
- LibAstro (C++ version)[10]

1.3 Installation

After you obtained the source code via the repository, you can compile the code using `GNU Make`[11]. In order to do so, you have to manually edit the `Makefile` and uncomment the part of the reconstruction package that you want to compile. The selections are:

- `MPIField`, for the field preparation routine.
- `MPIReconstruction`, for the reconstruction routine.
- `singleCPUHighres`, for the high-resolution reconstruction routine.

Furthermore, you will have to add the appropriate paths for the external libraries, which are not included in your path and library system-variables. Use the `ADDLIB` and `ADDINC` variables with the `-I` and `-L` options for this purpose.
Having set this you can simply run:

```
make install
```

for both, the field preparation and the reconstruction routine. This will create one of these three executables:

[2]http://gcc.gnu.org/
[3]http://www.mcs.anl.gov/research/projects/mpi/mpich1/
[4]http://www.gnu.org/software/gsl/
[5]http://www.mcs.anl.gov/research/projects/mpi/
[6]http://math-atlas.sourceforge.net/
[7]http://www.netlib.org/lapack/
[8]http://heasarc.nasa.gov/fitsio/
[9]http://heasarc.gsfc.nasa.gov/fitsio/CCfits/
[10]https://www.ita.uni-heidelberg.de/svn/libastrocpp
[11]http://www.gnu.org/software/make/

`SAW_FIELD`, `SAW_REC` and `SAW_HD`

One problem might arise due to an `C++` incompatibility of the `LAPACK` routines. If you have trouble with non-defined LAPACK routines like `clapack_dgetrf`, `clapack_dgetri` or `clapack_dgetrs`, please add the following lines after the include of cblas in `clapack.h`, which in most cases should be located in `/usr/local/include`:

```
 #ifdef __cplusplus
extern "C"
```

and at the end of the header:

```
 #ifdef __cplusplus
}
#endif
```

If the compilation ended without errors, you should be able to use the the reconstruction package. Some warnings during the compilation may arise and are due to the laziness of the author.
If you want to clean up the installation directory of the code, by copying the source code into a `./src` directory and deleting all object files, just type:

```
make clean
```

2 Field preparation routine

For this section we assume that you have a properly compiled version of the `SAW_FIELD` executable available on your machine.

2.1 Input file format

The code needs several input files, depending on what kind of reconstruction you are planning to perform. All these files are expected to be in ASCII format:

- Ellipticity catalogue
- Flexion catalogue
- Critical curve position estimates catalogue
- Multiple image systems catalogue
- Field mask definition file
- Parameter file

Only the last two files are mandatory.

2.1.1 Ellipticity input

First, delete all comments from your catalogue, the code will not check for any comment characters. This is again due to author-laziness reasons. Each line of the ellipticity catalogue represents a galaxy, which will be used in the field-preparation process. The information needed in one single line is the following:

`x-position` `y-position` $\gamma 1$ $\gamma 2$ `weighting-factor`

The unit of the position should be `arcsec` if you want to use strong lensing constraints. The reason for this are the error estimates on critical curve position and multiple image positions. If you only apply weak lensing constraints, the unit of the coordinates is absolutely arbitrary so could also be e.g. chip-pixel coordinates. The coordinate frame of the reconstruction field is defined in the parameter file, as described in Sec. 2.2. The weighting factor is important if you want to perform a weighted average for the galaxies in one pixel. If you don't want to use this option, set all weights to 1.0. All quantities need to be separated by at least one space.

2.1.2 Flexion input

All that has been said about the ellipticity catalogue is valid for the flexion catalogue, but there are two more flexion components:

`x-position` `y-position` $\mathcal{F}1$ $\mathcal{F}2$ $\mathcal{G}1$ $\mathcal{G}2$ `weighting-factor`

2.1.3 Critical curve estimates input

This catalogue declares positions in your field as part of a critical curve at a certain redshift. How to estimate these positions from observational data is not the scope of this manual (see the science paper). The format of this input file is then simply:

```
x-position  y-position  redshift
```

where each line represents one critical-curve estimate.

2.1.4 Multiple image systems input

The format of the multiple-image-system catalogue is a little more complicated. Each multiple image system must start with # followed by the index number of the system. The next line gives the number of images in the system and the source redshift separated by a comma. In the follwing lines the positions of the individual images are given, where x-position and y-position are separated by a comma. Afterwards, the next system begins again with #. Here is an example:

Multiple-image-system input

```
#1
3,1.5
2.3,-11.2
-0.66,-2.82
-5.9,-12.5
#2
3,2.0
1.5,18.6
6.23,14.53
9.05,7.17
```

2.1.5 Field mask definitions

You can define rectangular masks for your reconstruction field. This has to be defined in a mask-definition file, which must contain in the first line the number of masks in your field within quotation marks. Afterwards in separate lines, you have to define each mask by giving the minimum x-coordinate, maximum x-coordinate, minimum y-coordinate, maximum y-coordinate in quotation marks, separated by commas:

Mask definition

```
Number of rectangular masks = "2"
Rec-Mask1-Parameters = "-78.0,-10.0,-69.0,-2.0"
Rec-Mask2-Parameters = "21.0,52.0,23.0,52.0"
```

If you don't want to mask the field at all, just set the parameter in the first line to 0. An example of a masked field can be seen in Fig. 2.1

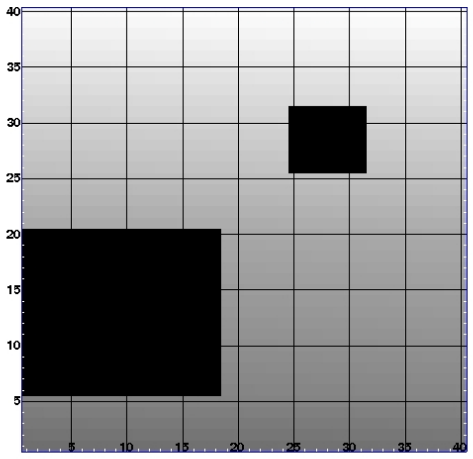

Figure 2.1: A 40x40 pixel example of the mask defined in the example. The total field size is 190x190 arcsec with the origin in the centre.

2.2 Writing parameter files

This section describes how a parameter file for the field-preparation routine must look like. We will start with an example of such a file and go through this example to describe the different options. Be aware that the order of the lines matters, so do not change this order and do not put any comments which span a whole line into this configuration file. Despite that, only the value which is written within the quotation marks is actuallly read by the code.

Field preparation parameters

```
Square field = "No"
Calculate shear = "Yes"
Calculate flexion = "Yes"
Calculate ccurve = "Yes"
Caluclate multiple image positions = "Yes"
Number of multiple image systems = "2"
Ellipticity input = "./ellip_catalogue.dat"
```

```
 8 Flexion input = "./flexion_catalogue.dat"
 9 CCurve input = "./ccurve_catalogue.dat"
10 Multiple image input = "./msystem_catalogue.dat"
11 Mask input = "./newrev_mask.conf"
12 Output prefix for ellip files = "./ellip_"
13 Output prefix for ccurve files = "./ccurve_"
14 Output prefix for flexion files = "./flexion_"
15 Output prefix for multiple image files = "/./msystem_"
16 Cluster name = "My Cluster"
17 Area of used galaxies = "-1000.0,1000.0,-1000.0,1000.0"
18 Area of output field = "-95.0,95.0,-95.0,95.0"
19 Number of galaxies to average for ellip = "10"
20 Number of galaxies to average for flexion = "10"
21 Iteration start value = "10"
22 Iteration stop value = "40"
23 Stepsize = "2"
24 Relative radius increment = "0.1"
25 Covariance pixelradius = "10"
```

We go through the list of all individual options in the order of their line number:

- `Square field flag:` Possible options are “Yes” or “No”. This option defines whether the code forces the reconstruction field to be a square. In practice this means that the pixel dimension in y-direction is set to that in x-direction. Maybe you waste galaxies if you do so.
- `Shear flag:` Possible options are “Yes” or “No”. This option sets whether ellipticity values are used for the field preparation. Make sure that you give the path to an ellipticity catalogue later.
- `Flexion flag:` Possible options are “Yes” or “No”. This options set whether flexion values are used for the field preparation. Make sure that you give the path to a flexion catalogue later.
- `Critical curve flag:` Possible options are “Yes” or “No”. This option sets whether critical-curve estimates are used for the field preparation. Make sure that you give the path to a critical-curve-estimate catalogue later.
- `Multiple image system flag:` Possible options are “Yes” or “No”. This options set whether multiple image systems are used for the field preparation. Make sure that you give the path to a multiple-image-system catalogue later.
- `Number of multiple image systems:` Should be an integer number, make sure that it is not set to 0. Multiple image systems are switched off with the flag before.
- `Ellipticity Input:` String which gives the path to the ellipticity catalogue.

- `Flexion Input:` String which gives the path to the flexion catalogue.
- `Ccurve Input:` String which gives the path to the critical-curve-estimate catalogue.
- `Multiple Image system Input:` String which gives the path to the multiple-image-system catalogue.
- `Field mask Input:` String which gives the path to the field-mask file.
- `Ellipticity output prefix:` String which gives path to and prefix of the desired output ellipticity files. You have to give just a prefix, which will be followed by an iteration number and the FITS extension in the output file.
- `Critical curve output prefix:` String which gives path to and prefix of the desired output critical curve files. It is just a prefix, which will be followed by an iteration number and the FITS extension in the output file.
- `Flexion output prefix:` String which gives path to and prefix of the desired output, flexion files. It is just a prefix, which will be followed by an iteration number and the FITS extension in the output file.
- `Multiple image system output prefix:` String which gives position and prefix of the desired output multiple image system files. It is just a prefix, which will be followed by an iteration number and an extension indicating ASCII output, here it is .dat.
- `Cluster Name:` String, describing the object of the reconstruction, this will be mainly just written to the header of the output FITS files.
- `Area of used galaxies:` Four double values, which describe the borders of the area from which you take galaxies for the averaging process, from the catalogues. This area can be bigger than the area of the output fields. The format is left edge, right edge, bottom edge, upper edge.
- `Area of output field:` Four double values, describing the size of the output fields. The pixel-resolution will be different for the different outputs, but the field size will stay the same. This options defines the coordinate frame of the reconstruction field.
- `Ellipticity galaxies:` Integer number which sets the minimum number of galaxies which should be used as a sample for the ellipticity averaging process per pixel.
- `Flexion galaxies:` Integer number which sets the minimum number of galaxies which should be used as a sample for the flexion averaging process per pixel.
- `Iteration start value:` This is the smallest integer pixel resolution in x-dimension for which an output should be created. The y-value will be set automatically according to the field size, or according to the square field flag.

- `Iteration stop value:` The largest integer pixel x-resolution. Again y is set accordingly.
- `Step size:` Sets the integer step size between the start and stop resolution.
- `Radius increment:` This double value defines the relative radius increment for the circles which are continously enlarged, until the number of galaxies per pixel to average over is reached. A good value is ~ 0.1.
- `Covariance pixelradius:` This integer value decides within which radius the code searches for overlap in the pixelised field. Setting this value low e.g. $\sim$ 2-4 will result in fast runtime but the code may miss covariances, which are actually present. A higher value than 10 does not make sense, since the covariance matrices become singular with this amount of correlation.

2.3 Running the code

If you are done with defining parameters and masks and if you have created all necessary data input, you can start the field preparation process by executing:

```
mpirun SAW_FIELD <configuration file>
```

or on some MPI installations you might need:

```
mpirun -np <number of threads> SAW_FIELD <configuration file>
```

The runtime strongly depends on the number of used cores, the resolution on which you want to perform the reconstruction later, the number of galaxies in your catalogue, the correlation radius that you defined before and whether you want to use ellipticity, flexion or both information. To get a hint on the runtime, see the comparisons in Tab. 2.1 and 2.2.

Number of used cores	Correlation radius	Runtime [sec]
1	3	6977
1	6	13040

Table 2.1: Runtime comparison with different correlation radii on an AMD Opteron 250, 2.4GHz, 1MB Cache. The catalogue contained 19568 galaxies, the initial resolution was 10x10, the final resolution was 60x60 with a stepsize of 2. Only shear averaging was performed, including flexion would double the runtime.

2.4 Output file format

Depending on which options you have chosen, there can be several output files. All these files have in common that there will be a separate file for each reconstruction

System configuration	Number of used cores	Runtime [sec]
1	2	1603
2	1	2982
2	2	1658
3	22	566

Table 2.2: Runtime comparison on different systems and with a different number of used cores. System 1 is based on an Intel Core 2 Duo P8600 @ 2.4 Ghz, 3MB cache, bogomips = 4786.74 (my notebook). System 2 is based on Dual Core AMD Opterons 285 @ 2.6 GHz, 1MB cache, bogomips = 5227.45 (Zeus@ITA). System 3 is a 8 Node, 22 core Linux Cluster, based on Dual Core AMD Opterons 280 @ 2.4 GHz, 1MB cache, bogomips = 4825.42, InfiniBand-Network (Apollon@ITA). The catalogue contained 19568 galaxies, the initial resolution was 20x20, the final resolution was 50x50 with a stepsize of 2 and a correlation radius of 3. Only shear averaging was performed, including flexion would double the runtime.

resolution that you have selected in the parameter file. Also, the prefix of these files was chosen in the parameter file, which will be followed by an index which indicates the x-dimension pixel resolution of one particular file.

2.4.1 Ellipticity output

If you have chosen the ellipticity averaging option, the output will be a FITS file with the following HDUs:

- `Primary:` The averaged first component of the ellipticity, so the expectation for the first component of the reduced shear.
- `Ext.: mean_ellip2:` The averaged second component of the ellipticity, so the expectation for the second component of the reduced shear.
- `Ext.: ellip1_sd:` Standard deviation for the first component.
- `Ext.: ellip2_sd:` Standard deviation for the second component.
- `Ext.: overlap:` Matrix, indicating how many galaxies one pixel shares with the other pixels in the field.
- `Ext.: ellip1_covariance:` The inverse of the covariance matrix for the first ellipticity component.
- `Ext.: ellip2_covariance:` The inverse of the covariance matrix for the second ellipticity component.
- `Ext.: field_mask:` Matrix showing the reconstruction grid. Non-masked pixels carry an index-number starting from 1001 counting up and masked pixels are set to 10.

The primary HDU also contains several header values, which are in detail:

- `DATE:` Time and date of field preparation.
- `OBJECT:` Name of the Object.
- `FILE:` Original ellipticity catalogue.
- `MASK:` Original mask-definition file.
- `FIELDX:` Physical size in x-dimension.
- `FIELDY:` Physical size in y-dimension.
- `AREA:` Physical area of the field.
- `PIXELSIZE:` Physical side length of one pixel.
- `PIXELS:` Number of non-masked pixels in the field.
- `EDGE_X:` Physical x-coordinate of the bottom-left pixel.
- `EDGE_Y:` Physical y-coordinate of the bottom-left pixel.
- `NUMGAL:` Number of galaxies used for the averaging process in the field.
- `GALDENS:` Galaxy density in the field.
- `GAL_PIXEL:` Number of galaxies, used for the average in one pixel.

2.4.2 Flexion output

If you have chosen the flexion averaging option, the output will be a FITS file with the following HDUs:

- `Primary:` The averaged first component of $\mathcal{F}$.
- `Ext.: mean_f2:` The averaged second component of $\mathcal{F}$.
- `Ext.: f1_sd:` Standard deviation for the first component of $\mathcal{F}$.
- `Ext.: f2_sd:` Standard deviation for the second component of $\mathcal{F}$.
- `Ext.: mean_g1:` The averaged first component of $\mathcal{G}$.
- `Ext.: mean_g2:` The averaged second component of $\mathcal{G}$.
- `Ext.: g1_sd:` Standard deviation for the first component of $\mathcal{G}$.
- `Ext.: g2_sd:` Standard deviation for the second component of $\mathcal{G}$.
- `Ext.: overlap:` Matrix indicating how many galaxies one pixel shares with the other pixels in the field.

- `Ext.: f1_covariance:` The inverse of the covariance matrix for the first component of $\mathcal{F}$.
- `Ext.: f2_covariance:` The inverse of the covariance matrix for the second component of $\mathcal{F}$.
- `Ext.: g1_covariance:` The inverse of the covariance matrix for the first component of $\mathcal{G}$.
- `Ext.: g2_covariance:` The inverse of the covariance matrix for the second component of$\mathcal{G}$.
- `Ext.: field_mask:` Matrix showing the reconstruction grid where non-masked pixels carry an index-number starting from 1001 counting up and masked pixels are set to 10.

The primary HDU also contains several header values, which are in detail:

- `DATE:` Time and date of the field preparation.
- `OBJECT:` Name of the Object.
- `FILE:` Original flexion catalogue.
- `MASK:` Original mask-definition file.
- `FIELDX:` Physical size in x-dimension.
- `FIELDY:` Physical size in y-dimension.
- `AREA:` Physical area of the field.
- `PIXELSIZE:` Physical side length of one pixel.
- `PIXELS:` Number of non-masked pixels in the field.
- `EDGE_X:` Physical x-coordinate of the bottom-left pixel.
- `EDGE_Y:` Physical y-coordinate of the bottom-left pixel.
- `NUMGAL:` Number of galaxies used for the averaging process in the field.
- `GALDENS:` Galaxy density in the field.
- `GAL_PIXEL:` Number of galaxies used for the average in one pixel.

2.4.3 Critical curve estimate output

If you have chosen the critical-curve-estimates option, the output will be a FITS file with the following HDUs:

- `Primary`: Field overview, indicating if a pixel is assumed to be part of a ciritcal curve. Those pixels are set to 1 in this map, all others to 0.
- `Ext.: pixel_relevance`: A map which shows the number of points from the original critical-curve-estimates catalogue assigned to a pixel of the grid.
- `Ext.: error`: This value is uniform on the whole grid and shows the error estimate on the critical-curve position due to the pixelisation of the grid.
- `Ext.: redshift_info`: The redshift of the estimated critical curve in a certain pixel. If one pixel contains several estimates from the catalogue, the mean is taken.
- `Ext.: field_mask`: Please see the two sections above.

The primary HDU also contains several header values, which are in detail:

- `DATE`: Time and date of the field preparation.
- `OBJECT`: Name of the Object.
- `FILE`: Original critical curve estimates catalogue.
- `MASK`: Original mask-definition file.
- `FIELDX`: Physical size in x-dimension.
- `FIELDY`: Physical size in y-dimension.
- `AREA`: Physical area of the field.
- `PIXELSIZE`: Physical side length of one pixel.
- `PIXELS`: Number of non-masked pixels in the field.
- `EDGE_X`: Physical x-coordinate of the bottom-left pixel.
- `EDGE_Y`: Physical y-coordinate of the bottom-left pixel.
- `NUMCC`: Number of critical-curve-estimate-catalogue points in the field.
- `CCERROR`: Error on the critical curve due to grid pixelisation.
- `MAXZ`: Highest redshift of an critical-curve estimate in the field.

2.4.4 Multiple image systems output

These output files are ASCII files, which contain all multiple image systems in the field, marked with a #, the number of images in each system, its redshift, its physical and pixel position and its error estimate. An example can be seen here:

Multiple image systems output

```
#1
3.00000
1.50000
2.30000
-11.2000
-0.660000
-2.82000
-5.90000
-12.5000
0.00000
0.00000
0.00000
0.00000
467.000
510.000
465.000
0.00000
0.00000
4.75000
#2
3.00000
2.00000
1.50000
18.6000
6.23000
14.5300
9.05000
7.17000
0.00000
0.00000
0.00000
0.00000
671.000
672.000
592.000
0.00000
0.00000
4.75000
```

3 Reconstruction routine

The second part of the SaWLens 1.2 reconstruction package is the reconstruction routine. This part of the code makes use of MPI, so it can be run on big parallel machines. Right now it does not make sense to run the code on more than 200 cores since the algorithm would break down, due to the logic of the work distribution. Before you can run the reconstruction package, make sure that you have already all files from the field-preparation routine available.

3.1 Input file format

The format of the input files does not need much of a description since they are just the output files of the field-preparation routine. The only important thing is that you have all files available at the same resolutions which you want to use in the reconstruction. This is a priori satisfied if you set the same start and stop resolution in the reconstruction and also the same step size.

3.2 Writing parameter files

The reconstruction routine relies on a main parameter file. Again, we will explain the options with an example file:

Reconstruction parameter configuration files

```
Use shear = "No"
Use flexion = "Yes"
Use ccurve = "No"
Use multiple image systems = "No"
Number of multiple image systems = "3"
Iteration start value = "10"
Iteration stop value = "40"
Increment step size = "2"
Set manual steps = "No"
Steps = "10,12,14,16,18,20,22,24,25"
Inner level iteration steps = "1"
Convergence threshold for iteration stop = "0.0001"
Prefix of shear-field files = "./ellip_"
Prefix of flexion-field files = "./flexion_"
Prefix of ccurve files = "./ccurve_"
Prefix of multiple image system files = "./msystem_"
Prefix of output files = "./rec_flexion_"
Regularisation type = "convshear"
Shear regularisation parameter = "0.0"
Flexion regularisation parameter = "100.0"
Average redshift of the shear background sources = "0.9"
Average redshift of the flexion background sources = "0.9"
Redshift of the cluster = "0.313"
```

As before, the ordering of the rows is crucial so don't change it with e.g. commented lines. The value, which is read by the code, is always set in quotation marks. The meaning of each line is the following:

- `Shear flag:` Selections are "Yes" or "No". This flag decides if you want to use shear constraints in the reconstruction. If set to "Yes", make sure that the shear-field files are available.
- `Flexion flag:` Selections are "Yes" or "No". This flag decides if you want to use flexion constraints in the reconstruction. If set to "Yes", make sure that the flexion-field files are available.
- `Critial curve flag:` Selections are "Yes" or "No". This flag decides if you want to use critical-curve constraints in the reconstruction. If set to "Yes", make sure that the critical-curve-field files are available.
- `Multiple image system flag:` Selections are "Yes" or "No". This flag decides if you want to use multiple-image-system constraints in the reconstruction. If set to "Yes", make sure that the multiple-image-system files are available.
- `Number of multiple image systems:` Integer number which should never be set to 0. If you want to deactivate multiple image systems, use the flag.
- `Start resolution:` Integer number which sets the initial reconstruction resolution, a reasonable value depends on the field but is typically situated between 10 and 20.
- `Final resolution:` Integer number which should be set between 30 and 75 depending on your data quality and patience.
- `Step size:` The step size between the different reconstruction resolutions as integer number.
- `Manual steps flag:` Selections are "Yes" or "No". If you set this flag you can manually define the reconstruction resolutions. This is done in the next line.
- `Manual steps:` See above, all individual steps have to be separated by commas.
- `Inner level iteration steps:` Integer number giving the maximum number of inner-level iterations. For the exact meaning of this please refer to the science paper.

- `Convergence threshold:` Double value, minimum change in the reconstruction convergence to stop the inner-level iteration.
- `Shear files prefix:` String with path to and prefix of the shear files produced by the field-preparation routine.
- `Flexion files prefix:`String with path to and prefix of the flexion files produced by the field-preparation routine.
- `Critical curve files prefix:`String with path to and prefix of the critical-curve files produced by the field-preparation routine.
- `Multiple image system files prefix:`String with path to and prefix of the multiple-image-systems files produced by the field-preparation routine.
- `Output prefix:` String with path to and prefix of the output files. Again, this prefix will be followed by an index number and the FITS extension.
- `Regularisation type:` Selections are "conv" if you want to regularise only on the convergence, "convshear" if you want to regularise also on shear, "flexion" if you want to regularise on flexion only and "convshearflex" if you want to regularise on all of the quantities mentioned.
- `Shear regularisation parameter:` Double value, defining the strength of convergence and shear regularisation. A reasonable value resides around 100.
- `Flexion regularisation parameter:` Double value, defining the strength of flexion regularisation. A resonable value resides around 100.
- `Average redshift of the shear sources.` Double value.
- `Average redshift of the flexion sources.` Double value.
- `Redshift of the lens.` Double value.

3.3 Running the code

Since the reconstruction routine is using MPI, executing the code is done by:

```
mpirun SAW_REC <configuration file>
```

On some systems you can, or you have to explicitely specifiy the number of threads on which the code should run, this is done with the -np option:

```
mpirun -np <number of threads> SAW_REC <configuration file>
```

The runtime of the code depends on the number of cores it runs on and on the final resolution.

Final Resolution	Number of cores	Runtime [sec]
30x30	2	1222
30x30	22	155
40x40	2	5233
40x40	22	653
60x60	22	5012

Table 3.1: Runtime comparison with different final resolutions and on a different number of used cores. Both machines are equipped with AMD Opteron 250, 2.4GHz, 1MB Cache Dual Core processors. The cluster uses an InfiniBand-Network.

3.4 Output file format

The final output of the code are single FITS files for each resolution. These output files contain again several HDUs:

- `Primary:` The reconstructed lensing potential.
- `Ext.: convergence:` The convergence map of the field derived from the lensing potential, referring to a source redshift of infinity.
- `Ext.: shear1:` The first shear component.
- `Ext.: shear2:` The second shear component.
- `Ext.: f1:` The first $\mathcal{F}$ component.
- `Ext.: f2:` The second $\mathcal{F}$ component.
- `Ext.: g1:` The first $\mathcal{G}$ component.
- `Ext.: g2:` The second $\mathcal{G}$ component.
- `Ext.: jacdet:` The determinant of the lensing Jacobian.
- `Ext.: field_mask:` The field mask as described in the field preparation chapter.

Furthermore the primary HDU contains the following header objects:

- `DATE:` Date and time of the reconstruction.
- `METHOD:` Contains s, c, f or m, when shear, critical curve, flexion or multiple image systems were used in the reconstruction.
- `ORIGINS:` Shear-field-files prefix.
- `ORIGINF:` Flexion-field-files prefix.

- `ORIGINC:` Critical-curve-field-files prefix.
- `REGSHEAR:` Shear regularisation parameter.
- `REGFLEXION:` Flexion regularisation parameter.
- `REGMETHOD:` Regularisation type.
- `CLUSTERZ:` Redshift of the lens.
- `SOURCEZ:` Redshift of the sources. The main result of a reconstruction the the lensing potential from which all other quantities are derived. An example can be seen in Fig. 3.1.

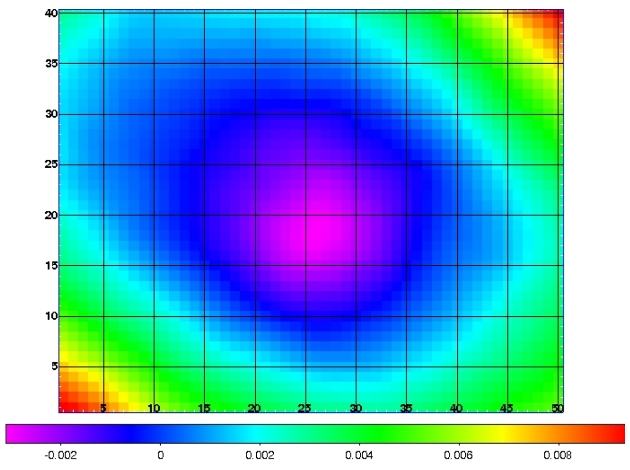

Figure 3.1: A reconstructed lensing potential on a grid of 50x50 pixels.

4 High-resolution reconstruction routine

For the following part of the package you need the `SAW_HD` executeable. Note that this is a standard, singe-threaded routine which is not run in a MPI-environment. The requirements regarding CPU-time are negligible compared to the other two routines.

4.1 Input file format

Since the high-resolution reconstruction relies on the result of a former low-resolution reconstruction, it needs the FITS-file of this reconstruction step. Furthermore, you need a strong-lensing output of the field-preparation routine, so a critical-curve estimate or multiple image systems on the refined resolution. It's hard to say which should be the grid size for this refined resolution. It depends on the quality and shape of your critical curve estimate, on the number and position of your multiple image systems and on the resolution of the preceding low-resolution reconstruction. As a rule of thumb, doubling this resolution might be a good start.

4.2 Writing parameter files

Here you see an example of a parameter file which is used for the high-resolution reconstruction:

High-resolution reconstruction parameters

```
Use critical curve information = "Yes"
Use multiple image systems = "No"
Filename of lowres result = "./rec_30.fits"
Filename high-res ccurve info = "./ccurve_60.fits"
Filename high-res msystem info = "./msystems_60.dat"
Filename for interpolated clustercore = "./clustercore.fits"
Filennname reset msystem info = "./rect_msystems.dat"
Filename for merged maps = "./merged.fits"
High resolution x-dim = "60"
High resolution y-dim = "60"
Cutpoints = "23,36,24,39"
Number of multiple image systems = "2"
Regularisation parameter = "100.0"
Regularisation scheme = "convshear"
Redshift of cluster = "0.313"
Strong lensing sigma = "0.12"
```

The explanation of the different lines is the following:

- `Critical curve flag:` Selections are “Yes” or “No”. This flag decides if critical-curve constraints are used. If set, make sure that a high-resolution critical-curve-estimates file is given later.
- `Multiple image systems flag:` As above but for mutliple-image-system constraints.
- `Lowres result:` Path to the file name of the preceding reconstruction on lower resolution.
- `Highres ccurve file:` Path to the file of the high-resolution critical-curve-estimates output written by the field-preparation routine.
- `Highres msystem file:` As above but for multiple-image systems.
- `Interpolated clustercore:` Path and filename for the output file which shows the interpolation of the low-resolution reconstruction (explained in Sec. 4.4.1).
- `Reset msystem file:` Path and filename of the multiple-image-systems file adpated to the refined grid.
- `Merged maps:` The high-resolution result merged with the low-resolution result on a multi-resolution map (explained in Sec. 4.4.2 and shown in Fig. 4.2).
- `Highres x-dim:` x-dimension of the refined grid. From this grid the cluster-core is cut out.
- `Highres y-dim:` As above but for the y-dimension.
- `Cutpoints:` The minimum x-value, maximum x-value, minimum y-value and maximum y-value for the cut-out of the clustercore from the highres grid. This cut-out depends on your strong-lensing information, but it should contain all strong lensing information. See Fig. 4.1 for an example.
- `Number of multiple image systems:` Integer number, do not set to 0. To switch off multiple image systems use the flag.
- `Regularisation parameter:` This double value decides on the strength of the regularisation in the interpolated low-resolution result. Usually this number can be set smaller then in the case of the low-resolution reconstruction routine.

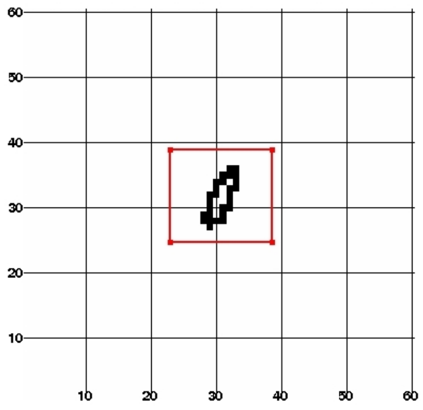

Figure 4.1: Cut-out of a clustercore on the refined grid, which just contains all critical curve estimates in the centre. The coordinates of the cut-out (in red) match with the ones from the example parameter file.

- `Regularisation scheme:` The type of the regularisation. Select "convergence" to regularise on the convergence, "convshear" to regularise also on shear, "flexion" to regularise on flexion and "convshearflex" to regularise on all of these quantities.
- `Cluster redshift:` A double value.
- `Strong lensing sigma:` A double value which gives the error estimate on the critical curve position. This can also be read from the FITS header of the field-preparation routine but we allow to set it manually for testing purposes.

4.3 Running the code

This routine does not use MPI, so execute it by typing:

```
./SAW_HD <configuration file>
```

Runtime is not an issue. Only if you interpolate to a very high refined resolution it may take a while.

4.4 Output file format

The output are two different FITS files. There is also an ASCII files which converts the multiple-image pixel positions into the cut frame, if the multiple image systems options is set. Anyway, this ASCII file is not particularly interesting for the user.

4.4.1 Interpolated low-resolution result

This file shows you the result of interpolating the low-resolution result to the final resolution, and the cut-outs of the critical curve estimates contraints if they are used. The file contains the following HDUs:

- `Primary:` The lensing potential interpolated to high resolution.
- `Ext. cut_potential:` The potential in the cut frame.
- `Ext. cut_ccurve:` The critical-curve estimates in the cut frame.
- `Ext. cut_redshift:` The critical-curve estimates in the cut frame.

4.4.2 Merged maps

This is the final result of a reconstruction, the file contains the following HDUs:

- `Primary:` The convergence map referring to a source-redshift of infinity. As all other outputs here, this is a map with different pixel-resolutions in the centre and in the outskirts of the field. Contained is the whole reconstruction field as defined by the field-preparation routine. An example of such a map can be seen in Fig. 4.2.
- `Ext. shear1:` As above but containing the first component of the shear.
- `Ext. shear2:` As above but containing the second component of the shear.
- `Ext. shear2:` As above but containing the determintant of the lensing Jacobian.

Furthermore, the file contains several header information:

- `METHOD:` Contains "c" if critical-curve estimates were used and "m" if multiple-image systems were used.
- `LOWRES:` Path and filename of the underlying low-resolution reconstruction.
- `CCURVE:` Path and filename of the critical curve constraints.
- `MSYSTEMS:` Path and filename of the multiple-image-systems constraints.
- `CUTX/Y/1/2:` Defining the cut-out of the refined field.
- `REG:` Value of the regularisation parameter.
- `REGTYPE:` The regularisation scheme which was used.
- `SIGMA:` Error estimate on the critical-curve position.

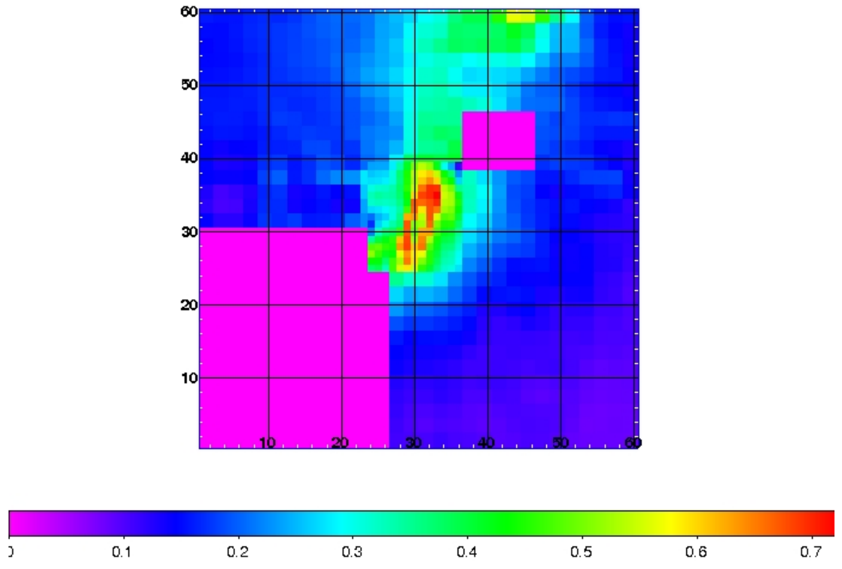

Figure 4.2: An extreme example of a masked, combined convergence map. You can see the refined resolution in the center of the image.

5 Upcoming improvements

The next updates and improvements are planned to be:

1. Ultra-fast GPU version of the reconstruction routine, using nVIDIA's CUDA[1] interface. (SaWLens 2.0)

[1]http://www.nvidia.com/object/cuda_home.html