ulens mcmc Manual
(version 0.1, Jul 2011)

Introduction

This software aims to help with the fitting of the microlensing model to an observed light curves. It has two components, one is the single lens and other is a binary lens fitting program.
There are 3 directories in the package:

- *single_lens_mcmc*
- *binary_lens_mcmc*
- *common_files_mcmc*

Both components share most of the code, this can be found in the *common_files_mcmc* directory. The single lens magnification calculation and parametrization are in the *single_lens_mcmc*, the binary lens parametrization and magnification calculations are in *binary_lens_mcmc*, respectively.
Both codes use their own set of configuration files prepared specifically for a given problem. These configuration files can be found in the *input* subdirectories. However there are some configuration files that are the same and can be copied from one code to another.

The code uses MCMC(Monte Carlo Markov chain)-like algorithm to find the solution and an MCMC algorithm to refine solution and find parameter uncertainties and correlations between them. The standard downhill methods are faster in finding best-fit solutions, however are much more prone to be stuck in the local minimum. Additionally, in the MCMC, by assigning a higher “temperature” to a chain, we can probe larger part of the parameter space and avoid being trapped by the local minima even better. The “MCMC-down” algorithm uses transitional probabilities that are constantly changing and are adopting to a local shape of the parameter space. The correlations between parameters are expressed in the form of a correlation matrix. The square roots of variances of the parameters (sigmas) are also used as characteristic scales.

Working with the code

Both directories, for single or binary code, have 4 essential subdirectories:

- *input*
- *output*
- *plot*
- *code_**

There is also an exemplary directory called *data*, however in principal data can be stored in any place, since one can specify in a configuration file where to look for data files.

The *input* directory contains all configuration files. The *output* contains files that are generated by the program that indicate the current status of the program as well as the results of the run. The *plot* directory holds the most recent solution during the run and one can generate plots with both the corresponding model and the data using the scripts provided in this directory to take a “snapshot” of the run.

In order to start a new fitting process or to start work with a new event, just copy one of the directories *single_lens_mcmc* or *binary_lens_mcmc* to create a new directory. For example with this command:

- *cp -r binary_lens_mcmc OB110935_fit1*

For each new set of configuration files, one can just copy the directory once more in order not to lose the previous solutions, plots and configurations. If you do not want to copy the data many times (which is recommended), just move the *data* directory out of *binary_lens_mcmc*, and adjust the first line of the file: *binary_lens_mcmc/input/data.ctrl* to specify the location of the data files.

If you do not want to copy any code files each time you start a new fit (which is recommended too), just make a symbolic link to the code directory instead of copying it. For example:

- *mkdir OB110935_fit1*
- *cd OB110935_fit1*
- *cp -r ../binary_lens_mcmc/input .*
- *cp -r ../binary_lens_mcmc/output .*
- *cp -r ../binary_lens_mcmc/plot .*
- *cp ../binary_lens_mcmc/Makefile ../binary_lens_mcmc/run .*
- *ln -s ../binary_lens_mcmc/code_binary_lens code_binary_lens*

Do this just once, later you can simply copy the whole directory like this:

- *cp -r OB110935_fit1 OB110935_fit2*

And this will copy a symbolic link to a code instead of a code itself together with the whole directory.

Every new run you should edit the files in the *input* directory to specify what code should do and how. There is usually no need to modify the code itself.

Cleaning previous runs

After creating a new directory for a new microlensing event, usually it is a good idea to run:

- *make distclean*

command, this will remove all the previous configuration files made by you, all compiled codes and all plots of the previous models. If you would like to start a new fitting process for the same event you have copied the base directory from, run:

- *make clean*

this will remove only the results of the previous runs, but preserve the configuration files located in *input* directory.

Compiling the code

The easiest way to compile the code is to run:

- *make*

in the *code_** directory or one level above, in *binary_lens_mcmc*, *single_lens_mcmc* or the directory you have created, for example: *OB110935_fit1*. We will call this directory the *base* directory from now on.

You can run the code by writing:

./run - in the base directory, or
./binary_lens_mcmc or ./single_lens_mcmc in the code_* directory
There is no difference. Use command *pwd* to see where you are or command *ls* to see what files are in the current directory.

Configuration

All configuration files have extensive comments inside. The comments start with # character and are omitted by the program while reading the configuration file. The content of the all configuration files with full comments is attached at the end of this document. Below we mention briefly what is the role of each file.

If you delete the comments from a configuration file you can always find them in the corresponding file with an **.info* extension. For example file *input/data.ctrl* has a corresponding file *input/data.ctrl.info* with some exemplary configurations and full set of comments inside. These files are not used by the program, they are only for user's information. The input directory contains the following files:

- *input/data.ctrl* - contains information about all data sets: file name that contains the light curve, observatory name, error rescaling and limb darkening coefficients, as well as a bad data file - file that contains Julian dates of the observations that should not be taken info account in the fitting.
- *input/event.ctrl* - specifies the event name, its equatorial coordinates (RA and Dec) and a $t_{0,par}$ parameter used as a reference point in the definition of t_0, u_0, parallax (and orbital motion parameters). The coordinates are not used in the modeling without parallax.
- *input/mcmc.init* - initial values of all fit parameters, together with the limits they are allowed to vary in and initial step sizes for the Gaussian transitional probability in the MCMC. Step sizes are not used if the correlation matrix is provided and READ_CORR field in *mcmc.ctrl* is 1.
- *input/mcmc.ctrl* - configuration file for the MCMC process; how long the burning stage lasts, whether the initial correlation matrix shall be read form a file or step sizes from *mcmc.init* be used, what should be the temperature of the chain, how many stable points on an MCMC link we require program to gather, etc.
- *input/modeldat.ctrl* - here the time span and cadence of the model light curve is defined. This is used only for plotting the current model (every few minutes) and has no impact on the actual calculations.
- *input/corr.matrix* - correlation matrix used to generate initial transitional probability for the MCMC
- *input/corr.sigmas* - initial step sizes in each parameter corresponding to the correlation matrix

Parametrization

There are two sets of parameters: "fit" parameters and "alternative" parameters (parameters

specified by the user). The fit parameters are the ones that fully describes and specifies the mode and they are used in an MCMC process. The alternative parameters are the ones the user would like to inspect in the output file in addition to the fit parameters. For example, even if you would like to fit the single lens using t_{eff}, it is also very useful to have t_E and u_0 provided as an output.
One example of single lens parametrization could be:

- t_0-$t_{0,par}$ - time of the closest approach to the lens relative to some chosen time
- t_{eff} - effective time, product of u0 and tE (in days)
- t_E - Einstein time (in days)
- t_* - source radius crossing time (in days)
- $\pi_{E,N}$ - north component of the parallax
- $\pi_{E,E}$ - east component of the parallax

and then the corresponding alternative parametrization could be:

- t_0 - time of the closest approach of the lens to the line of sight toward the source
- u_0 - impact parameter, i.e., distance of the closest approach (in θ_E)
- t_E - Einstein time (in days)
- ρ - source radius (in θ_E)
- π_E - magnitude of the parallax
- φ_π - angle of the relative lens-source motion in a geocentric frame (in degrees)

Changing the parametrization is straightforward to do.

Modifying the parametrization

If you need or want to change the fit parametrization just edit the file *parametrization.f90* in the *code* directory. Remember that it could be useful to copy the whole code directory to a new location before doing this in order not to override the defaults. If you are using a symbolic link to a code directory, for example:

- *ls -l OB110935_fit1*, gives:
- *code_binary_lens -> ../binary_lens_mcmc/code_binary_lens*

you can delete this link, and copy the code directory in a whole.

There is an array of the names of the parameters: *par_names* in the *parametrization.f90*. These names are used to identify the parameters in the configuration file: *input/mcmc.init*. And the order in which parameters appear in this list is the internal order used by other routines in the *parametrization.f90* file. Array called *par_fmt* hold formats in which the fit parameters are printed out - this controls length of each record, number of decimal places and notation (scientific or engineering). The family of functions called: *get_*(params)* is used to translate your set of parameters info model description used by magnification and χ^2 calculation routines.

If you would like to change fit parametrization do this:

- edit array *par_names*
- modify array *par_fmt* accordingly
- change each of the *get_*(params)* subroutines to properly translate your new set of parameters. There routines are: *get_u0*, *get_tE*, *get_rho*, *get_piEN*, *get_t0*, etc.

You can introduce more complex parameters like lens orbital motion (in the case of binary lens) by simply adding new elements to mentioned arrays, increasing value of the variable *nparm*, writing additional *get_*(params)* routine and modifying the *get_pos(x,y,t,params)* subroutine to use new parameters. Modify the alternative parameters accordingly. All this in a single file: *parametrization.f90*

Modifying alternative parameters

The alternative parameters are meant to provide additional information to the output of a program. This can be different projection of the parameter space or just a different units. You decide.

To modify alternative parameters to your needs just edit the subroutine *convert_parm_alt* in the *parametrization.f90* file in the *code* directory. This subroutine translates all model parameters to your set of alternative parameters, setups their names and output formats. See the code for more details.

Binary-lens specific configuration

Calculation of the finite source magnification for binary-lens is very time consuming. This is the reason we introduced the *input/fs.ctrl* file. This file informs the program in which section of the light curve one should use the finite source calculations. The format of the file is simple: every line defines one region in which the finite source calculation should be performed; the region is defined by 2 numbers meaning the start and end time of the region in Julian days. The optional, third, number is an integer flag saying what type of finite source calculation should be performed: stokes' method, inverse ray-shooting, map, hexadecapole approximation, etc. See the *get_mu()* routine in *getmu.f90* to see which magnification methods have been implemented at this moment.

Results

Chains

The results are gathered in *output* directory, with additional information and visualization in *plot* directory.

Ideally, the main result file is *output/chain.stable* which contains a few thousands of points (links) in the vicinity of the best-fit model. These points can be used to evaluate the best-fit solution given the data, and to estimate uncertainties as well as correlations between different parameters. The accompanying files are:

- *output/chain.best* - current best link, i.e., the one with the lowest χ^2
- *output/chain.current* - while program is running, this is the current point being calculated
- *output/chain.burn* - all links from the beginning of the program run before the chain is considered stable. The transitional probabilities varies from link to link in the burning stage, hence these link cannot be used for statistical analysis.
- *output/chain.stable* - all links after the chain is considered stable. These can be used for statistical studies since the transitional probabilities is kept fixed during this stage.

Correlations and step sizes

Correlation matrix and parameter variances are calculated continuously throughout the program activity based on a subset of most recent points. (Variances are kept in the form of sigmas, i.e. square roots of variances). This information is stored in an output file every time a new correlation matrix is calculated, additionally there are several other files in output directory which hold information on previously calculated correlations:

- *output/corr.matrix.initial* and *output/corr.sigmas.initial* - these file holds the starting values.
- *output/corr.matrix.used* and *output/corr.sigmas.used* - these values are the ones that had been used for the stable phase of the MCMC. They are evaluated at the beginning of the stable phase from a longer than usual subset of recent points.
- *output/corr.matrix* and *output/corr.sigmas* - when the program is running, these are the most recently calculates values. At the end of the program, the correlations and sigmas evaluated from the very end of the chain are written into these files.

Status

The current status of the program can be found in *output/status* file. The *error.log* and *debug.log* holds some additional information. For example every time the program stops because of error, the full description of problem is given in *output/error.log*.

Light curve, model and trajectory plots

The current solution is printed in the form of light curves and model curves every few minutes to *plot* directory. These are the files:

*plot/plot.** - files with the names of plot.0, plot.1, plot.2, etc. contains all light curves that had been used for fitting together with the current model magnifications and residuals for each point.

plot/model.dat - this file contains the current model light curve calculated inside the time span defined in the *input/modeldat.ctrl* file, this file also specifies cadence of points in the model light curve.

Info files

The *plot* directory also contains some handy information about the current run, which is specifically useful for plotting scripts:

- *obs.info* - this file contains the information about the number of observatories and their names and order.
- *column.info* - provides names and column numbers of each of the values found in *output/chain.** files.
- *columnmcmc.info* - unveils which of the parameters were actually varied with the MCMC and in which columns of the chain files they are located.

Super Mongo scripts

There are a few SM script in the *plot* directory:

plot/plotlc.sm - is used to plot the light curve with model and residuals

plot/plottj.sm - is used to plot the source trajectory

plot/plotmcmc.sm - is designed to plot the MCMC links from the chain.stable to see what are the correlations, assess the convergence of the chain and see the parameter uncertainties.
There is a file *plot/sm.crtl* that is used by this scripts and defines the time span for the light curve plotting, the part of plane the trajectory is plotted on.
To plot the light curve you may use one of two commands:

- *sm -m plotlc.sm* or simply
- *make lc*

To plot source apparent trajectory use:

- *./plot_trajectory* or simply
- *make tj*

Examples of configuration files (with comments)

input/data.ctrl

```
# directory where data is located ../data/
#  filename      nickname   errscale errquad   Gamma  bad_data
   phot0.dat      OGLE         1.0     0.0       0.0     -
   phot1.dat       MOA         1.0     0.0       0.0     -
   phot2.dat      Bronberg     1.0     0.0       0.0  phot2.bad
```

input/event.ctrl

```
# NAME
OGLE-2010-BLG-012
# RA (in hours floating-point number)
18.23425
# DEC (in degrees, floating-point number)
-30.534
# T0_PAR: Should be very close to t0
4120.0
```

input/mcmc.init

```
# MCMC PARAMETERS: mcmc.init
#
# There is one row for every parameter. Every row has 6 columns. These are:
# mcmc     - vary this parameter in MCMC or keep fixed (1 or 0)
# init     - initial value of the parameter
# min      - lower boundary for the value of this parameter
# max      - upper boundary
# stepsize - initial step size
# relative position of the parameters here will be used in output as well
#
# name mcmc     init        min     max    stepsize
   t0      1    4124.00    1000.0 10000.0    0.05
   u0      1     0.1         -1.0     1.0    0.005
   tE      1     25.00        0.0   500.0    0.5
```

```
rho     0     0.02         0.0     0.2    0.003
q       1     0.48         0.0     1.0    0.0001
s       1     0.7          0.0    10.0    0.01
alpha   1     198.00    -360.0   720.0    0.1
piEN    0     0.00        -5.0     5.0    0.001
piEE    0     0.00        -5.0     5.0    0.001
```

input/mcmc.ctrl

```
# MCMC CONTROL FILE
#
# SEED            - random seed (integer)
# READ_CORR       - value = 0: use step sizes from mcmc.init or value =1: correlation
matrix from a file input/corr.matrix and input/corr.sigmas
# FIX_TRANS_PROB  - value = 1: fix the step sizes and correletions between parameters
(do not modify transisional probability). value = 0: free
# MAX_TRIAL_NUMB  - maximal number of trial allowed (to limit total running time of
the program)
# NBURN_MIN       - 0 if you do not want to change this, otherwise minimal length of
the burning process required by you (it is set in the code to 2*n_dchi2_1, but you can
extend it)
# NSTABLE         - 0 if you do not want to change this, otherwise lenght of the
stable chain you need (it will be not smaller than 1*n_dchi2_1 or 500)
# TEMPERATURE     - temperature of the chain (default = 1.0)
#
# put numbers in the lines below, one integer per line, in the same order as in
legend.
#
1345376245  # SEED
0           # READ_CORR
0           # FIX_TRANS_PROB
60000       # MAX_TRIAL_NUMB
750         # NBURN_MIN
3000        # NSTABLE
1.0         # TEMPERATURE
```

input/modeldat.ctrl

```
# This file (modeldat.ctrl) tells program how dense and for what times to plot model
light curve
# Specify as many regions as you want. One region per line
#
# eg.
# HJD_MIN_MODEL  HJD_MAX_MODEL   DETLA_HJD
5060.0   5111.2    0.1
# HJD_MIN_MODEL  HJD_MAX_MODEL   DELTA_HJD
5111.2   5116.3    0.01
# HJD_MIN_MODEL  HJD_MAX_MODEL   DELTA_HJD
5116.3   5200.0    1.0
```

input/corr.matrix

```
# Holds correlation matrix for all MCMC parameters
# If you have changed the number or order of MCMC parameters, change this matrix
accordingly
# You can skip reading in correlation matrix and sigmas by putting READ_CORR equal to
0 in mcmc.ctrl
# then the diagonal correlation matrix will be generated (in output/ dir) and
stepsizes from mcmc.init
# will be used as sigmas
#
# CORRELATION MATRIX
 1.0  0.0  0.0
 0.0  1.0  0.0
 0.0  0.0  1.0
```

input/corr.sigmas

```
# Holds sigmas, ie. sqrt of variances, of all MCMC parameters, these are used to
change correlation matrix to covariance matrix
# If you have changed the number or order of MCMC parameters, change the order here
accordingly
# You can skip reading in correlation matrix and sigmas by putting READ_CORR equal to
0 in mcmc.ctrl
# then the diagonal correlation matrix will be generated (in output/ dir) and
stepsizes from mcmc.init
# will be used as sigmas
#
#
# SIGMAS OF PARAMETERS
  0.01
  0.001
  0.1
```

input/fs.ctrl

```
# this file is used to specify all regions where finite source calculation have to be
made
#
# HJD_min  HJD_max
5084.0 5089.0
5109.0 5110.0
```

plot/sm.ctrl

```
# The file sm.ctrl is used to specify limits for the SM plot
#
# HJD_MIN  HJD_MAX
5060.0  5120.2
# MAG_MAX  MAG_MIN
6.5    9.0
# RES_MIN  RES_MAX
-0.1   0.1
# X_MIN X_MIN
-0.2   0.2
```

```
# Y_MIN Y_MAX
-0.2   0.2
```