**University of Canterbury**
Department of Physics and Astronomy

# Gravitational Microlensing

## An automated high-performance modelling system

User manual *v1.0*

by

Alistair M$^{c}$Dougall

Associated with a thesis submitted in partial
fulfilment for the degree of Doctor of Philosophy

October 2014

# Contents

# Chapter 1

# System Overview

## 1.1 Purpose of this document

This document is designed to provide an overview of the computer system used to control the microlensing modelling software on the University of Canterbury, Department of Physics and Astronomy computer system. For a detailed description of the development, use, and how the modelling system works see 'Gravitational Microlensing — An automated high-performance modelling system' PhD thesis submission. This document is used to provide further information about the codes developed, and partially as a trouble-shooting guide in case a hardware or software system fault occurs.

The modelling system development is still an on going piece of work, and therefore changes to be code are likely to be made as new feature are added and evolved. This means the following documentation is subject to change, and on occasion may contain historic information.

If you believe any documentation to be missing or lacking sufficient detail, please contact the author so that an updated version can be produced.

Throughout this document an Altered Julian Date (AJD) is used instead of the standard Julian Date (JD), and they can be easily converted by the equation AJD=JD-2450000. This alternate date system is used to reduce the size of the variable in the modelling system.

# Chapter 2

# Database

## 2.1 Host location

The complete modelling system is based on a MySQL database hosted on the physastro computer. This database can be managed and modified with the use of the microlensing controller.

## 2.2 Management

The automated data retrieval software will automatically create and modify database entries. Any use of the controller software will correctly apply the required modifications to the database. Therefore unless an error occurs, there should be no need for manual interaction with the database.

## 2.3 Structure

The table in the database exists to store the current state of the modelling process, and to maintain a record of the model results.

The database table contains the following structure, where the column name is shown a pair of by square brackets which state the data type, if it can (yes) or not (no) be a null value, and its default value. This is then followed by the description of the entry.

ID [int(10), no, none]
: A unique ten digit ID number every event must have. The characters that make up the ID (character number in square brackets) come from:

[1] - A 0 or 1 flag to show if the event came from the OGLE (0) or MOA (1) telescope.

[2-5] - The year that the event was first observed.

[6] - The area of the sky that the event was found in, 0 or 1, if in the LMC or BLG, respectively.

[7-10] - The event ID number given by the observing group that discovered it.

Event [string(32), no, none]
The same format as the unique ID, but a string with characters and numbers.

new_data [int(1), no, 1]
A 0 or 1 flag, to identify if there is new data (1), or not (0), since it was last modelled.

retrieved [timestamp, no, current date-time]
The date and time of when the latest data was retrieved.

single_model [int(1), no, 1]
A 0 or 1 flag, to identify if the event should have a single lens model (1), or not (0).

binary_model [int(1), no, 0]
A 0 or 1 flag, to identify if the event should have a binary lens model (1), or not (0).

s_u0 [double, yes, null]
The single lens model $u_0$ parameter value.

s_u0_sig [double, yes, null]
The single lens model $\sigma_{u_0}$ parameter value.

s_t0_guess [double, no, none]
An initial estimate for the single lens model's $t_0$ parameter value. It is used to help the modeller initialize the parameter search space. The value is determined from the survey telescope's event web page.

s_t0 [double, yes, null]
The single lens model $t_0$ parameter value.

s_t0_sig [double, yes, null]
The single lens model $\sigma_{t_0}$ parameter value.

s_tE [double, yes, null]
The single lens model $t_E$ parameter value.

s_tE_sig [double, yes, null]
The single lens model $\sigma_{t_E}$ parameter value.

s_tE_guess [double, yes, null]
: An initial estimate for the single lens model's $t_E$ parameter value. It is used to help the modeller initialize the parameter search space. The value comes from an approximation of the full width half max of the data or, if this is not possible, a default value of 22 is provided.

s_MOA_baseline [double, yes, null]
: An indication of the baseline magnitude of the MOA data set.

s_MOA_baseline_sig [double, yes, null]
: [HISTORIC], no longer used.

s_OGL_baseline [double, yes, null]
: An indication of the baseline magnitude of the OGLE data set.

s_OGL_baseline_sig [double, yes, null]
: [HISTORIC], no longer used.

data_linked [int(1), no, 0]
: A 0,1 or 2 flag, which identifies if the event is linked to any other in the database.

  (0) - It is a unique event.

  (1) - It is the parent to another event.

  (2) - It is the child to another event.

linked_data [string(100), yes, null]
: A list of event IDs without any separation of the events that it is linked to. If data_linked=2 the event ID is of the parent event, if data_linked=1 the event ID(s) point to the child event(s).

s_Amax [double, yes, null]
: The estimate of the single lens model's peak magnification.

s_Anow [double, yes, null]
: The current magnification of the event.

s_priority [float, no 0]
: The priority value for the single lens modelling queue.

s_submitted [int(1), no, 0]
: A 0 or 1 flag, which indicates if the event has been submitted to the modeller (1), or not (0).

s_processed [timstamp, yes, null]
: The date time of the most recent single lens modelling process..

s_unsuitable [int(1), no, 0]
: A 0 or 1 flag, which indicates if the model is suitable (0), or unsuitable (1).

latest_data_point [double, no, none]
The AJD of the latest data point in the event's data.

b_processing [int(1), yes, null]
The ID number of the GPU (only IDs 1, 2, and 3 currently exist)that the binary processing is currently running on, otherwise 'null' when the event is not being processed.

b_LR_grid_search [int(1), no, 0]
A 0 or 1 flag, which indicates if the modelling has performed (1), or not (0), the first part of the grid search code (the magnification map interpolation).

b_HR_grid_search [int(1), no, 0]
A 0 or 1 flag, which indicates if the modelling has performed (1), or not (0), the second part of the grid search code (the local area MCMC search).

b_emcee [int(1), no, 0]
A 0 or 1 flag, which indicates if the modelling has performed (1), or not (0), the EMCEE part of the search.

b_d [double, yes, null]
The binary lens model's latest value for parameter $d$.

b_d_sig [double, yes, null]
The binary lens model's latest error term $\sigma_d$.

b_q [double, yes, null]
The binary lens model's latest value for parameter $q$.

b_q_sig [double, yes, null]
The binary lens model's latest error term $\sigma_q$.

b_rho [double, yes, null]
The binary lens model's latest value for parameter $\rho$.

b_rho_sig [double, yes, null]
The binary lens model's latest error term $\sigma_\rho$.

b_u0 [double, yes, null]
The binary lens model's latest value for parameter $u_0$.

b_u0_sig [double, yes, null]
The binary lens model's latest error term $\sigma_{u_0}$.

b_phi [double, yes, null]
The binary lens model's latest value for parameter $\phi$.

b_phi_sig [double, yes, null]
The binary lens model's latest error term $\sigma_\phi$.

b_t0 [double, yes, null]
: The binary lens model's latest value for parameter $t_0$.

b_t0_sig [double, yes, null]
: The binary lens model's latest error term $\sigma_{t_0}$.

b_tE [double, yes, null]
: The binary lens model's latest value for parameter $t_E$.

b_tE_sig [double, yes, null]
: The binary lens model's latest error term $\sigma_{t_E}$.

b_processed [datetime, yes, null]
: The date time of when the binary modeller was last completed.

s_MOA_m0 [double, no, 27.4]
: The MOA web site's quoted calibration magnitude.

s_MOA_fref [double, no, 0]
: The MOA web site's calibration flux value.

s_MOA_data_baseline [float, yes, null]
: The estimated MOA baseline magnitude from the latest model.

s_OGL_data_baseline [float, yes ,null]
: The estimated OGLE baseline magnitude from the latest model.

s_model_deviation [float, yes, null]
: [DEV] This will be used to help determine any deviation from the single lens model.

s_plot_updated [datetime, yes, null]
: The date time of when the single lens plot was last updated.

# Chapter 3

# File structure

## 3.1 Host

The entire modelling system is hosted by the u-lenser user account, with a file structure that is divided into four main areas.

## 3.2 Common files

Throughout the file structure a range of .pid files may exist. These contain the process ID of the different processes that are currently running. The .pid files are crucial in preventing multiple instances of the same code being initiated. Improper removal of these files may result in read/write issues for some processes, taking up too much memory, and ultimately system crashes if a process is initiated too many times. The removal of these files should only be performed when it is known that the process it refers to does not exist (checked with a tool such as top), and the .pid file needs to be removed before a new instance of the process can be started.

The .pid file contains a single integer that represents the unix process ID of its associated code, which can be determined by the file name of the .pid. Each .pid file is created at the start of a code, and exists for the length that the code is running. On the clean completion of the process that created the file, it will be removed. However, in the rare instance where the process that created the file crashes, or the system terminates the process, the associated .pid file is not cleaned up. This may prevent the initialization of the code which creates this file, as the system will think it is already running. Should this situation occur, first confirm that the process you want to start does not already exist, and if so remove the correct .pid file, only then should a re-attempt to start the process in question be performed.

Another file type which occurs throughout the modelling system are .log files. These exist as a continuous record of messages that come from the different codes that are initiated by cron jobs throughout the system. They maintain a record of error messages, and are used to prevent an overflow of the u-lenser mail box.

## 3.3 Web site (~/html/)

The entire web site is written in html and php. These files are used by the web host to display the web site located at http://www2.phys.canterbury.ac.nz/~u-lenser/. Within this directory all the public files should have user read permissions, and any folders need user read and execute permissions.

about.php
: The web page that provides a description of what the modelling system is, who is involved, how it works, and all acknowledgments.

binary_data_item_emcee.php
: The web page that shows the binary lens result of an event, for the EMCEE part of the search.

binary_data_item_HR.php
: The web page that shows the binary lens result of an event, for the second part (local area MCMC) of the search.

binary_data_item_log.php
: The web page that displays the processing log of an event.

binary_data_item_LR.php
: The web page that shows the binary lens result of an event, for the first part of the grid search.

binary_data_item.php
: The web page that shows the most recent result of an event's binary lens modelling.

captcha_code_file.php
: The part of the contact page that makes the captcha image.

Connections/localconnect.php
: The file that contains the login details for the MySQL database. This needs to have valid login details for the whole web site to work.

contact.php
: The web page that formats the whole contact form.

data_list_binary.php
: The web page that lists all binary lens events for a given year.

data_list_extreme_mag.php
: The web page that lists all extremely highly magnified single lens events for a given year, where the limit that defines if an event is extremely magnified is contained within this file.

data_list_high_mag.php
: The web page that lists all highly magnified single lens events for a given year, where the limit that defines if an event is highly magnified is contained within this file.

data_list.php
: The web page that lists all events for a given year.

data_years.php
: The web page that lists all years since the web site first collected data.

feed.php
: [HISTORIC] A web page used to provide an RSS feed of the latest results.

head.php
: The header file which is called by all main body web pages. It provides the top section of all the pages including the drop down menu options.

images/
: A folder for all the web site images (e.g. banner, UC logo).

index.php
: The home page of the web site that displays the banner, and the three most recent single and binary lens models produced.

info.php
: A private file, which is used to determine the current state of the servers PHP. To use it, make it publicly accessible and then view it through the web address to see the results.

jquery.nivo.slider.js
: A downloaded java script used to perform the image scrolling banner on home page

jquery.nivo.slider.pack.js
: A downloaded java script used to perform the image scrolling banner on home page

lib.php
: [HISTORIC] A library file used to control the database access. [CURRENT] It also contains a function to strip out any special characters that users could use on input boxes to perform malicious functions i.e. drop tables.

main.css
: A large list that defines the graphical scheme for the web site, such as a lists of all the colours and font styles used for certain definitions.

monofont.ttf
: A true type font, which is used by the captcha code to produce the random string of letters and numbers.

nivo-slider.css
: A style file for the slide show of images on the home page.

nivo_themes/
: A folder of associated images to go with the slide show banner on the home page.

php_version.php
: A duplicate of info.php.

scripts/
: A folder of scripts used to validate the inputs of the user contact form.

single_data_item.php
: A web page that displays the results for a given single lens event.

submitted.php
: A web page that is shown upon a successful submission through the contact form.

tail.php
: The tail part of the web page, which includes the contact details and special acknowledgments to specific groups.

## 3.4 Data archive (~/html/data/)

All microlensing events stored in this system are originally identified either by the MOA or OGLE telescope (sometimes with observations from both). As described in Chapter 2, each event has a unique ID and name. The unique event name is used to create a folder for each event found by both the MOA and OGLE groups. The name of the event and folder follows the same structure as detailed in the database:

**OBS-YEAR-LOC-IDNO**

**OBS** - Which observatory observed the event (either MOA or OGLE).

**YEAR** - The year that the event was first identified.

**LOC** - The location in the sky that the event is located (either, BLG or LMC).

**IDNO** - The identification number of the event, which was given by the MOA or OGLE group.

All files and folders within this directory can be publicly accessible (even if not directly shown or linked through the web site), but to be accessed all files need to have user read permission, and folders need user read and execute permissions. This is true for all files and folders, unless something is desired to be kept private, in which case it would be best to store it outside of the ~/html/ directory entirely.

In situations where it has been identified that both MOA and OGLE have observed the same event, the database and file structure are modified to link the two data sets together. In the database a flag is changed to indicate it is either the parent or child of another event, and the event ID number is used to indicate which event it is linked to. The file structure represents the link by putting a soft link to the child event's data in the folder of the parent event. This allows the retrieve/update data processes to continue like normal, without needing special conditions in case an event is linked or not. To ensure the same results are shown for each linked event on the web site, the child event has it's results (single/ and binary/) folders replaced by soft links to the relevant folders of the parent event.

### 3.4.1 (~/html/data/OBS-YEAR-LOC-IDNO/)

Within an event folder are all the source data files from each observatory that provide their data publicly. A few other files and folders may exist in this directory, which follow a strict naming convention. No other files should be added to this folder location, as it may lead to the modelling code trying to include these files as data sets.

#### 3.4.1.1 (~/html/data/OBS-YEAR-LOC-IDNO/single/)

Every event added to the modelling system is by default assumed to be a single lens, and will automatically be submitted to the single lens modeller. Upon completion of the single lens modelling, the results from the cluster are then stored in this folder. These files are all publicly available and should have user read permissions. The 'single_data_item.php' file uses these images when displaying the results for each single lens event.

#### 3.4.1.2 (~/html/data/OBS-YEAR-LOC-IDNO/binary/)

Not all events have this folder, as it is automatically created by the system when a user designates a new event as binary through the modelling control software.

Within this folder are ID files used by the modeller to keep a track of its progress while it is performing a binary model search. Within this folder are also several other folders detailed in the following sections.

##### 3.4.1.2.1 (~/html/data/OBS-YEAR-LOC-IDNO/binary/WIP/)

A folder used for images which are created while the modeller is on going. It is mostly historic, but it does contain a useful record of the images straight from the modeller without some user changes.

##### 3.4.1.2.2 (~/html/data/OBS-YEAR-LOC-IDNO/binary/associated/)

Any files associated with this event that do not belong in the WIP or output folders can be stored here. This folder also contains the archive of the evet's previous modelling. They are stored with a naming convention of the datetime when the modelling was initialized. These archives contain the same data structure as the output folder described in section 3.4.1.2.3.

##### 3.4.1.2.3 (~/html/data/OBS-YEAR-LOC-IDNO/binary/output/)

This location contains all the output and working files of an event's binary modelling process. Many of these data and image files contain useful information about the model, and are used by the 'binary_data_item_*_.php' web pages to display images and information on the web site. Therefore everything within this folder is given user read permission. For any file which contains a list of parameters, they are presented in the following order:

$d$ - The lens separation (in units of Einstein radii)

$q$ - The lens mass ratio.

$\rho$ - The source radius (in units of Einstein radii).

$u_0$ - The distance of closest approach (in units of Einstein radii).

$\phi$ - The angle of source crossing (taken anti-clockwise from the positive x-axis).

$t_0$ - The time of closest approach (in units of AJD).

$t_E$ - The Einstein crossing time (in units of AJD).

A full list of all the possible files within this folder are shown below:

EMCEE_num_data_points.txt
: The number of data points used by the EMCEE modelling process.

GS_num_data_points.txt
: The number of data points used in the grid search part of the modelling process.

Grid_search_HR_OBS-YR-IDNO_LC.png
: The light curve for each data set. It is produced using the result of the local area MCMC chains.

Grid_search_HR_all_LC.png
: A light curve with all data produced from the result of the local area MCMC chains.

Grid_search_HR_caustic_trajectory.png
: The source trajectory overlaid on top of the caustics produced from the result of the local area MCMC chains.

Grid_search_HR_chi^2_map.png
: The minimized $\chi^2$ map with overlaid local area MCMC chain movements.

Grid_search_HR_mag_map.png
: A magnification map produced from the result of the local area MCMC chains.

Grid_search_HR_residual.png
: A set of residuals from the light curve with all data produced from the result of the local area MCMC chains.

Grid_search_LR_OBS-YR-IDNO_LC.png
: The light curve for each separate data set. It is produced using the result of the initial grid search.

Grid_search_LR_all_LC.png
: A light curve containing all data sets. It is produced from the result of the initial grid search.

Grid_search_LR_caustic_trajectory.png
: The source trajectory overlaid onto the caustics produced from the result of the initial grid search.

Grid_search_LR_chi^2_map.png
: The minimized $\chi^2$ map of the initial grid search.

Grid_search_LR_mag_map.png
: A magnification map produced from the result of the initial grid search.

Grid_search_LR_residual.png
: A set of residuals from the light curve containing all data. It is produced from the result of the initial grid search.

HR_progress.log
: A log file identifying where the local area MCMC search is currently up to.

LR_progress.log
: A log file identifying where the initial grid search is currently up to.

accepted_mcmc.npy
: A stored 2D array of chain number by $\chi^2$ value of every step in the local area MCMC search.

all_fit_parameters_OBS-YEAR-LOC-IDNO.txt
: A table of every minimised $d$, $q$ grid search parameter set tried, and the resulting $\chi^2$ value.

all_raw_data.png
: A plot of all the data included in the modelling, with no model light curve or scalling.

best_fit_parameters_OBS-YEAR-LOC-IDNO.txt
: The best fitting parameter set determined from the initial grid search.

burn_in_chains.txt
: The EMCEE output parameter sets and $\chi^2$s from the burn in.

burn_in_movement.png
: A plot of the EMCEE burn in parameter movements.

d_mcmc.npy
: A stored 2D array of the chain number by $d$ parameter value of every step in the local area MCMC search.

d_q_map.txt
: A table of $d$, $q$, and $\chi^2$ results, used to produce the minimized $\chi^2$ map.

data_err_scale_factors.txt
: The calculated error scaling factor for each data set, which is used to normalize the data.

emcee_OBS-YR-IDNO_LC.png
: The light curve plot for each observatory that provided data to the modeller. It is produced from the result of the EMCEE search.

emcee_all_LC.png

A light curve including all data. It is produced from the result of the EMCEE search.

emcee_caustic_trajectory.png

The source trajectory overlaid onto the caustics, produced from the result of the EMCEE search.

emcee_mag_map.png

A magnification map produced from the result of the EMCEE search.

emcee_output.png

The covariance plots of all parameters involved in the EMCEE search.

emcee_residual.png

A set of residuals from the light curve containing all data sources. It is produced using the result of the EMCEE search.

emcee_results.txt

A text file listing the mean value and standard deviation of every parameter in the EMCEE search.

first_pass_all_fit_parameters_OBS-YEAR-LOC-IDNO.txt

[HISTORIC] This shows the event name, and grid search resolution as headers, followed by a table of the trial parameter sets and their returned $\chi^2$ values.

global_vars_OBS-YEAR-LOC-IDNO.txt

A List of the grid search global control variables values, which are detailed below in order:

MAX_DATA_SITES - The maximum number of data sites that are able to be modelled (defined by hardware limits of the GPU memory).

LR_num_points - The resolution of the initial magnification map, before any modifications.

threads_per_block - The number of threads per block used when making a magnification map.

SUB_PIXELS - The number of subpixels used to calculate the kernel.

SUB_PIXELS_DIM_X - The number of threads per block in the kernel calculations.

DATA_LIMIT - The maximum number of data points per source (upper limit = 7168, defined by hardware limit of GPU memory).

MAX_THREADS - The maximum number of threads allowed (optimum determined by experimentation).

lrho - The lower limit of the $\rho$ parameter grid search area.

drho - The step size of the $\rho$ parameter grid search area.

urho - The upper limit of the $\rho$ parameter grid search area.

hr_all_fit_parameters_OBS-YEAR-LOC-IDNO.txt

The final and initial, respectively, trial parameter sets and $\chi^2$ values of the local area MCMC chains. $\chi^2$ values are calculated by the ICIRAS method for the final chain point, and by linear interpolation for the initial chain point.

hr_best_fit_parameters_OBS-YEAR-LOC-IDNO.txt

The current list of parameter values that produce the lowest $\chi^2$ values from the local area MCMC. This file will update to include the linear parameter terms for each data set when the code has completed searching all the local area chains.

hr_d_q_map.txt

The $d$ and $q$, parameter values before and after the local area MCMC search, followed by the final $\chi^2$ value.

hr_global_vars_OBS-YEAR-LOC-IDNO.txt

A list of values for the local area MCMC global control variable, which are detailed below in order:

The list is the same as global_vars_OBS-YEAR-LOC-IDNO.txt up to MAX_THREADS.

d_hr_steps - [HISTORIC] This is used to define the higher resolution grid search.

q_hr_steps - [HISTORIC] This is used to define the higher resolution grid search.

u0_hr_steps - [HISTORIC] This is used to define the higher resolution grid search.

phi_hr_steps - [HISTORIC] This is used to define the higher resolution grid search.

top_num_pixels - The number of local area MCMC chains to explore, which must be even (as the code explores half the amount as close solutions, and half as wide solutions)

count_limit - The maximum number of steps taken in a local area MCMC.

HR_threads_per_block - The number of threads used to modify magnification maps to make them higher resolution.

lrho - As before.

drho - As before.

urho - As before.

input_parameters.txt

The input string of variables that are provided to the modeller from the user input, via the u-lenser controller. This file contains the following:

event_id - The event name.

paczyński - A 0 or 1 flag, to state if the model fits a paczyński curve (0) or not (1).

data_peaks - The number of peaks expected to be in the model.

t0 - An estimate for the $t_0$ parameter, or the AJD of the first peak.

tE - An estimate for the $t_E$ parameter, or the AJD of the second peak.

Al - An estimate of the minimum value that the maximum change in magnitude needs to be.

t_start - The lower limit of the AJD data range (0 means no limit).

t_end - The upper limit of the AJD data range (0 means no limit).

DATA_LIMIT - A limit to the number of data points per data source (preferably a value that is $2^n$).

Num_ignores - The number of data sets that are ignored.

{IGNORE_LIST[@]} - A list of data sets event names that are to be ignored (0 if none).

latest_OBS-YR-IDNO_LC.png
: The most recent plot of the light curve and the data from the named source (one for each data source).

latest_all_LC.png
: The most recent light curve containing all data.

latest_chi^2_map.png
: The most recent minimized $d$, $q$, $\chi^2$ map, or covariance plot if the EMCEE modelling has finished.

latest_mag_map.png
: The most recent model solution's magnification map.

latest_raw_data.png
: The most recent plot of all the raw data.

latest_residual.png
: The most recent residuals of the latest light curve containing all data.

latest_trajectory.png
: The most recent trajectory overlaid on a caustic map.

likelihood_values.txt
: The parameter set from the EMCEE search which achieved the lowest $\chi^2$ value, and linear parameters for each data source, followed by it determined $\chi^2$ value.

likelihood_values_burn_in.txt
: A list of all parameters trialled and $\chi^2$ values calculated in the burn in part of the EMCEE search.

log.txt
: The modelling log file, displayed in the processing log tab of an event's web page.

mcmc_movement_chain_all.png
: A plot of the local area MCMC $\chi^2$ evolution for all chains.

mcmc_movement_chains.png
: Separated plots of the local area MCMC chains, which show each chain's overall $\chi^2$ evolution, and a magnified final 10%.

mcmc_movement_d.png
: Separated plots of the local area MCMC chains, which show each chain's overall $d$ parameter evolution, and a magnified final 10%.

mcmc_movement_phi.png
: Separated plots of the local area MCMC chains, which show each chain's overall $\phi$ parameter evolution, and a magnified final 10%.

mcmc_movement_q.png
: Separated plots of the local area MCMC chains, which show each chain's overall $q$ parameter evolution, and a magnified final 10%.

mcmc_movement_rho.png
: Separated plots of the local area MCMC chains, which show each chain's overall $\rho$ parameter evolution, and a magnified final 10%.

mcmc_movement_t0.png
: Separated plots of the local area MCMC chains, which show each chain's overall $t\_0$ parameter evolution, and a magnified final 10%.

mcmc_movement_tE.png
: Separated plots of the local area MCMC chains, which show each chain's overall $t\_E$ parameter evolution, and a magnified final 10%.

mcmc_movement_u0.png
: Separated plots of the local area MCMC chains, which show each chain's overall $u\_0$ parameter evolution, and a magnified final 10%.

mcmc_params_chain_*.png
: Separated plots of $\chi^2$ and all model parameters, which show the overall movement and a magnified final 10% of chain N. Updated as the modelling process is ongoing.

mcmc_params_chain_best_*.png
: Separated plots of $\chi^2$ and all model parameters, which show the overall movement and a magnified final 10% of chain N. Created upon completion of all chains.

model_compare_HR_all_LC.png
: A comparison plot of the light curves produced from the best solution after a local area MCMC search, and a user provided parameter set.

model_compare_HR_cumulative_chi2.png
: A comparison plot of the cumulative $\chi^2$ produced from the best solution after a local area MCMC search, and a user provided parameter set.

model_compare_HR_residual.png
: A comparison plot of the light curve residuals produced from the best solution after a local area MCMC search, and a user provided parameter set.

model_compare_HR_trajectory.png
: A comparison plot of source trajectories overlaid onto the caustic structures produced from the best solution after a local area MCMC search, and a user provided parameter set.

model_compare_LR_all_LC.png
: A comparison plot of the light curves produced from the best solution after the initial grid search, and a user provided parameter set.

model_compare_LR_cumulative_chi2.png
: A comparison plot of the cumulative $\chi^2$ of the light curves produced from the best solution after the initial grid search, and a user provided parameter set.

model_compare_LR_residual.png
: A comparison plot of the light curve residuals produced from the best solution after the initial grid search, and a user provided parameter set.

model_compare_LR_trajectory.png
: A comparison plot of source trajectories overlaid onto the caustic structures produced from the best solution after the initial grid search, and a user provided parameter set.

model_compare_all_raw_data.png
: A plot of the raw data used when producing the comparison plots.

phi_mcmc.npy
: A stored 2D array of the chains number by $\phi$ parameter values of every step in the local area MCMC search.

q_mcmc.npy
: A stored 2D array of the chains number by $q$ parameter values of every step in the local area MCMC search.

raw_data.png
: An image of all the raw data used.

rho_mcmc.npy
: A stored 2D array of the chains number by $\rho$ parameter values of every step in the local area MCMC search.

sample_chains.txt
: The EMCEE output parameter sets and $\chi^2$s of the sample search.

sample_movement.png
: An image of the parameter movements during the EMCEE sample search.

t0_mcmc.npy
: A stored 2D array of the chains number by $t_0$ parameter values of every step in the local area MCMC search.

tE_mcmc.npy
: A stored 2D array of the chains number by $t_E$ parameter values of every step in the local area MCMC search.

top_chains_caustic_trajectories.png
: A 2x3 grid of plots showing the source trajectory overlaid on the lens' caustic structure for each local area MCMC search solution.

top_mcmc_movement_chain_all.png
: A plot of all local area MCMC chain's $\chi^2$ movements, colour coded by order of production.

top_mcmc_movement_chains.png
: Separate plots of the local area MCMC chains, showing each chain's overall $\chi^2$ evolution, and a magnified final 10%. These are produced in order of analysis, and are updated while search is ongoing.

top_mcmc_movement_d.png
: Separate plots of the local area MCMC chains, showing each chain's overall $d$ evolution, and a magnified final 10%. These are produced in order of analysis, and are updated while search is ongoing.

top_mcmc_movement_phi.png
: Separate plots of the local area MCMC chains, showing each chain's overall $\phi$ evolution, and a magnified final 10%. These are produced in order of analysis, and are updated while search is ongoing.

top_mcmc_movement_q.png
: Separate plots of the local area MCMC chains, showing each chain's overall $q$ evolution, and a magnified final 10%. These are produced in order of analysis, and are updated while search is ongoing.

top_mcmc_movement_rho.png
: Separate plots of the local area MCMC chains, showing each chain's overall $\rho$ evolution, and a magnified final 10%. These are produced in order of analysis, and are updated while search is ongoing.

top_mcmc_movement_t0.png
: Separate plots of the local area MCMC chains, showing each chain's overall $t_0$ evolution, and a magnified final 10%. These are produced in order of analysis, and are updated while search is ongoing.

top_mcmc_movement_tE.png
: Separate plots of the local area MCMC chains, showing each chain's overall $t_E$ evolution, and a magnified final 10%. These are produced in order of analysis, and are updated while search is ongoing.

top_mcmc_movement_u0.png
: Separate plots of the local area MCMC chains, showing each chain's overall $u_0$ evolution, and a magnified final 10%. These are produced in order of analysis, and are updated while search is ongoing.

u0_mcmc.npy
: A stored 2D array of the chains number by $u_0$ parameter values of every step in the local area MCMC search.

walker_distribtuion.png
: A plot of the distribution of the starting points (walkers) used in the EMCEE search.

## 3.5 ~/web_page_control/

This area contains all the code that is relevant to the automated control and updating of the web site side of the system, and it also contains the database management control codes.

For each source of data, a unique python script exists that has the data hosts URL included. If required, this URL is a variable that changes based on the current year, and should not need updating as long as the data host's file structure remains the same. These python scripts are designed to download the data and decide if enough valid data points exist. If new data exists, it will update the local file and the database to indicate the existence of new data for the event. For MOA and OGLE data, if the file is of a new event, a new database entry is created. For all other data sources, the data file is added to the parent event's (MOA or OGLE) location, and the parent's database entry is updated to reflect any change.

Within in this directory the following files exist:

Anow.py
: A program that is designed to loop through the database and update the single lens magnification of each event. This code also uses an algorithm to determine the single lens priority of modelling, which is also updated on the database.

clear_event.py
: A user code called by the microlensing controller, it removes all results and plots associated with an event's single lens model.

cron_tab
: The file submitted to the cron table to instigate the 'data_cron.py' file on the phyhpc1 machine.

cron_tab_physastro
: The file submitted to the cron table to instigate the 'MySQL_DB_backup_physastro.sh' file on the physastro machine.

data_cron.py
: The python file called by the physhpc1 machine to start the data retrieval processes, 'update_single_plots.py', and 'Anow.py'.

DB_management.py
: The python script that the user control software uses to manage and modify the database.

display_all.py
: [HISTORIC] This is now merged into 'DB_management.py'. It is a program which fetchs and displays the entire database entry for a given event.

init_retrieve_MOA (or OGLE, Planet, ROBONET, ufun)
: These scripts set up a proxy for internet access before it calls the python retrieve scripts for each data source.

list_data_sites.py
: A program that identifies which events are linked to a parent file, and prints them to the screen as a list.

make_binary.py
: A program called by microlensing controller that marks an event as a binary lens. It will create the relevant file structure required, and change the database entry.

microlensing_db_backup.sql
: A backup file of the microlensing database (updated hourly by a cron job).

MySQL_DB_backup_physastro.sh
: A script called by cron on physastro that backups up the database hourly.

OGLE_last_changed.txt
: An empty file.

remove_binary.py
: A code called by microlensing controller that removes an event's binary association. It removes all the database results, files, and folders of the binary event.

Retrieve_MOA.py (or OGLE, Planet, ROBONET, ufun)
: These files download each site's event data, and updates the database if new data exists.

single_lens_plots.py
: A code called by the microlensing controller that updates an event's single lens plots with optional limits provided by the user.

test_retrieval
: [HISTORIC] An output file used when testing the data retrieval codes.

update_single_plots.py
: A code called by the cron tab on the physhpc1 that continuously loops through the database updating the single lens plots (if required).

## 3.6 System information codes (∼/code/)

This folder contains programs that are initialized by the cron table on the carina machine, or are called by the microlensing user controller. Below is a detailed list of each.

binary/
: A folder containing all the code used as part of the binary modelling.

data_cron.py
: A file called by the cron table on the carina machine.

linked_moa_ogle
: This is called by 'data_cron.py' to internet enable the carina machine, and start the program 'linked_moa_ogle.py' that links MOA and OGLE events.

list_data.py
: A code that lists all data files located in the folder of the requested parent event.

single/
: [HISTORIC] This folder contains the original development of the single lens modelling codes before the use of the cluster.

card_status.sh
: A script that can run on the carina machine to display a window that updates every minute, which provides an overview of the GPU card's statuses.

device_query.py
: A code that provides hardware information about each NVIDIA GPU device installed in the current machine.

NumRecC.zip
: A zipped file of all example codes in C from the Numerical Recipes book.

cron_tab
: The cron table uploaded to the carina machine. It is set to run once a day.

license.txt
: The license agreement that goes with the NVIDIA CUDA SDK.

linked_moa_ogle.py
: This program is called by 'linked_moa_ogle' to fetch the list of linked MOA and OGLE events, manage the file structures to create/remove softlinks between these events, and update the database to represent these changes.

NVIDIA GPU Computing SDK
: The NVIDIA provided software development kit files, which contains some CUDA optimized algorithms.

## 3.7 Binary modeller codes, and user control codes (~/code/binary/)

This folder contains the majority of the modelling code, it focuses on the modelling, visualization, and analysis of binary events. Within this folder a range of versions of the same core code exist, which come from experimentation and trails of new ideas. This folder also contains important tables of values that are read in by the modeller to save time, over of re-computing them. Several folders exist that contain the results of previous trials, and libraries of figures that are used for referencing when required. Below is a detailed list of these files and folders:

archive/
: A folder which contains an archive of previous versions of codes found in this directory.

archive_event.py
: A program that takes the input as the event name and duplicates the event's output folder to create an archived version in the event's associated folder.

backup_QUE_binary_modelling.txt

[HISTORIC] A backup of the 'QUE_binary_ modelling.txt' file, which is used whilst trialling a new code.

check_and_config.py

A program that runs a pre-check to identify if the inputed user event exists, and if so it will attempt to locate a free GPU device and update the database before the modelling of the event commences.

clear_device_id.py

A program that takes an input of the event name and updates the database to disassociate it from a GPU device. This is used at the end of the modelling process as part of the clean up.

CUDA_device_availability.py

A program that returns how many GPU devices are free for use by the binary modeller, out of the total number that exist.

curand_test.py

[HISTORIC] A development code used to test the cuda random number generation.

custom_HR_caustic_and_LC.py

A code that allows users to quickly create a caustic structure and source trajectory plot, with hard coded values that the user can change to control the dimensions of the plot. It also produces a light curve and residuals plot. This program is not callable by the microlensing system control software.

emcee_minimised_2002_3_minimising.py

[DEV] A customized version of 'emcee_minimised.py' used to perform a model minimisation from a published result solution without a $\rho$ parameter. It was designed to work with old data sets as part of the PhD thesis submission.

emcee_minimised_2002_3_minimising_rho.py

[DEV] A customized version of 'emcee_minimised.py' used to perform a model minimisation from a published result solution. It was designed to work with old data sets as part of the PhD thesis submission.

emcee_minimised_dev.py

[DEV] A development version of 'emcee_minimised.py', which is used whenever a new method wants to be trialed without affecting the original modelling code.

emcee_minimised_higher_order_affects.py

[DEV] An ongoing development version of 'emcee_minimised.py' to include higher order affects into the EMCEE modelling part of the system.

emcee_minimised_inc_parallax.py
: [DEV] An experimental version of the EMCEE code to trial an implementation of parallax.

emcee_minimised.py
: The standard EMCEE binary modelling method used in the standard analysis of a binary event.

events/
: [HISTORIC] A list of various event used for analysis before the MySQL database and the current file structure was designed.

Full_Search_dev.py
: [DEV] A development version of 'Full_Search.py', which is used whenever a new method wants to be trialed without affecting the original modelling code.

Full_Search_peak_search.py
: [HISTORIC/DEV] A custom code used to experiment with the ability of interpreting the number of peaks on the interpolated light curve.

Full_Search.py
: The standard grid search and local area MCMC binary modelling method used in the normal analysis of a binary event.

high_res/
: [HISTORIC] This folder contains text and images of an old grid search performed at a higher than average resolution.

image_centered_ray_shooting_grid_areas.py
: [DEV] A development version of the CUDA image centred inverse ray shooting code, including additional outputs that enable various plots of caustic structures and the areas to be integrated by the ICIRAS method.

images/
: This folder contains a range of folders and images that were used as part of the PhD thesis submission as well as visual outputs from various trials and experiments, and interesting patterns accidentally created with erroneous codes during development.

include/
: A C code library files that can be called if required.

list_currently_being_modelled.py
: A program called by the microlensing user control software that will list all binary lens events currently being modelled, and all queued events (if any exist).

mag_maps/
: [HISTORIC] A folder with a complete list of magnification maps at all $d$, $q$ values used in a standard grid search method.

making_maps.txt

[HISTORIC] The output log file, that was produced when creating all the magnification maps in 'mag_maps/'.

mcmc_interpret_2002_3_minimising.py

[DEV] A customized version of 'mcmc_interpret.py' to work specifically with old data sets and perform a comparative analysis with published results. It was used as part of the PhD thesis submission.

mcmc_interpret_2002_3_minimising_rho.py

[DEV] A customized version of 'm̀cmc_interpret.py' to work specifically with old data sets and perform a comparative analysis with published results. It was used as part of the PhD thesis submission.

mcmc_interpret_dev.py

[DEV] A development version of 'mcmc_interpret.py', which is used whenever a new method wants to be trialed without affecting the original modelling code.

mcmc_interpret.py

The EMCEE result interpretation of the binary modelling method used in the standard analysis of a binary event.

null_file.txt

As the name suggests, contains nothing.

NumRecC/

A folder of all example codes in C from the Numerical Recipes book.

parallax_perihelion_vernal_equinox_2012_to_2020.txt

A list of dates when the Earth is at perihelion and vernal equinox from 2012 up to 2020.

peak_list.npy

A numpy saved array of the number of peaks each light curve will contain in the order of the magnification maps and $u_0$, $\phi$ trajectories trialled during the standard grid search process.

peak_locs.npy

A numpy saved array of the location of peaks each light curve will contain in the order of the magnification maps and $u_0$, $\phi$ trajectories trialled during the usual grid search process.

peaks_list_cp.txt

A duplicate of the 'peak_list.npy' file (kept as a back up).

peaks_list_file_creator.py

A program that loops through every $d$, $q$, $u_0$, $\phi$ that are trialled as part of the

standard grid search. It determines the location and number of peaks in the produced light curve. This program outputs the 'peak_list.npy', 'peak_locs.npy', and 'peaks_list.txt'.

peaks_list.txt
: A text file version of 'peak_list.npy'.

QUE_binary_modelling_post_sort.txt
: A temporary file used by the binary queue management sorting process. It writes the re-ordered list to a separate file before replacing the original.

QUE_binary_modelling.txt
: The current binary modeller queue, the first row contains the column headers.

QUE_manager_binary_modeller.py
: The binary modelling queue control code called by the microlensing user software. It enables full management of the binary modeller queue.

rand_test.py
: [DEV/HISTORIC] A development code used to test random number generation.

root solver/
: The improved root solver code developed by J. Skowran, written in fortran 77, and includes my converted version in C.

run_all_dev.sh
: [DEV] A development version of the 'run_all.sh', which is used whenever a new idea is to be trialled without affecting the original.

run_all.sh
: The script called by the microlensing user control software, that manages and runs the full analysis of the binary lens modelling process. It is called by 'start_binary_modeller.sh', which initializes the modelling, plotting, and clean up of the files and database, as well as including an optional e-mail acknowledgement upon completion.

single_binary_compare_controller.py
: A file called by the microlensing control software that validates and manages the user inputs. It locates the resources required before initializing the single/binary lens model comparison tool 'single_binary_compare.py'.

single_binary_compare.py
: An analysis tool called by the microlensing control software that produces comparison plots and values between an event's binary lens solution, and its single lens solution.

start_binary_modeller_dev.sh
: [DEV] A development version of 'start_binary_modeller.sh', which is used whenever a new idea is to be trialled without affecting the original.

start_binary_modeller.sh
: A script that is called by the microlensing control software. It validates and identifies the available resources before initializing the binary modelling (by calling 'run_all.sh') of the requested event.

stop_binary_modeller.py
: A script that is callable from the microlensing control software. It allows the user to end ongoing binary modelling process. The re-start option is not 100% reliable and therefore should be used with caution if you intend to resume the modelling.

test/
: [HISTORIC] A temp folder used to store temp files and images.

test.py
: [HISTORIC] A file used to test new ideas.

user_binary_plotter.py
: A file that is called by the microlensing control software that allows users to instigate the plotting of any binary event and provide customizations, such as which data to include and limit the plot dimensions.

user_model_compare_controller.py
: A file that is called by the microlensing control software that validates the user request and determines the available resources before initializing the 'user_model_compare.py'.

user_model_compare_just_HR.py
: [DEV] A customised model comparison software (based on 'user_model_compare.py'), which was used to analyse and produce plots used in the PhD thesis submission.

user_model_compare.py
: A program that is called by 'user_model_compare_controller.py' to perform the analysis, and output the values and plots of a model comparison between the binary lens solution determined by the GPU method, and a user proposed solution.

varying_u0_phi/
: [HISTORY] A folder that contains a variety of figures produced during the trials when developing the grid search method.

WIP_plots_copy.py
: [DEV] A copy of the 'WIP_plots.py' file.

WIP_plots_dev_mb140075.py
: [HISTORIC] A customized version of the plotting code for event MOA-2014-BLG-0075.

WIP_plots_dev_ob030235.py
: [HISTORIC] A customized version of the plotting code for event OGL-2003-BLG-0235.

WIP_plots_dev_ob070472.py

[HISTORIC] A customized version of the plotting code for event OGL-2007-BLG-0472.

WIP_plots_dev_ob120406.py

[HISTORIC] A customized version of the plotting code for event OGL-2012-BLG-0406.

WIP_plots_dev.py

[DEV] A development version of the 'WIP_plots.py', which is used whenever a new idea is to be trialled without affecting the original.

WIP_plots.py

The plotting code for binary lens events. It creates all the output figures from the grid search and local area MCMC search. If called through the microlensing controller while an event is still undergoing the modelling process, it will produce the relevant plots for the currently found best solution.

# Chapter 4

# Computer systems used

## 4.1 Overview

This chapter contains a list of all the computer systems utilized by the microlensing modeling system. It details the use of each computer system and its contribution to the modeling system. In the majority of cases a cron job has been set up to ensure the system is continually running, and to restart the system in the event of a system shut down.

When running the microlensing user control software, the first communication to the microlensing system is through the 'login.phys.canterbury.ac.nz' machine. Before the first use of the user control software on a new machine, ssh keys need to be set up on the users machine and the login machine. This is to allow uninterrupted communication between the user and the University of Canterbury network. Within the local network (on the u-lenser account), all the required ssh communications between the internally connected machines have been set up with ssh keys to prevent any password requests for internal communications.

## 4.2 Physastro

This machine hosts the MySQL database. It is not a heavily used machine and typically has long up-times, making it an ideal server, as it should rarely fail. Should the database fail, it will prevent modeling codes from uploading their results, and also stop the majority of the web site from working.

This machine also has a single cron job that runs every hour, which automatically backs up the database to ~/web_page_control/microlensing_db_backup.sql.

## 4.3 Physhpc1

A range of codes run on this machine, designed to keep the web site and database up to date. Two programs 'Anow.py' and 'update_single_plots.py' run continuously on this machine to update the database and plots of single lens models. They are initialized by a cron job that runs every hour. Should 'update_single_plots.py' determine that there are no more plots that require updating, it will end the process. 'Anow.py' has no ability to terminate itself as it is always updating the database, by a calculation based on the current time, therefore this program should always be running. Both of these codes create a .pid file upon starting, this file identifies the process ID number of the program. Each time the cron job tries to restart these programs it will first check to see if the job is already currently running, and if so continue without starting another instance of the program.

In addition to the two jobs designed to continuously run and update the web site, there are multiple other cron jobs, which are designed to retrieve the data. Each source of data (MicroFUN, MOA, OGLE, PLANET, RoboNET) requires a separate program to fetch the data and add it to the file structure, as well as update the database to indicate the existence of new data. The python file for each data retrieval code is not called directly by the cron job as they first require a proxy to be set up to enable internet access for the user/machine. Instead, the cron job calls a script 'init_retrieve_*' which contains the necessary details (username and password) hard coded to set up the proxy before calling the relevent 'Retrieve_*.py ' file to update the database.

Should the data source change their web address, the URL inside the retrieve python files would need updating. If the data source changes the format of their web site then the python retrieve file would need re-developing. The data retrieval codes can take a long time to complete, especially near the end of an observing season (when more events exist). Therefore it is important to ensure the data retrevial codes .pid files are not removed (without careful consideration) to prevent multiple instances of the same data source retrieval codes running simultaneously, which can overload the system, waste resources, and even write the incorrect data files to the wrong events.

This machine is not only used for automated programs, but it is also used to perform numerous user requested tasks from the microlensing control software. These jobs perform the user requested single lens re-plots, job queue management, and database management (including proposing/removing binary events). All binary modelling and plotting requests are not performed on this machine due to the lack of suitable GPU facilities, instead see section 4.4.

## 4.4 Carina

This is the GPU machine hosting the three NVIDIA Tesla cards used in the binary modelling. All binary event modelling and plotting is performed on this machine which is controllable through the microlensing control software.

A single cron job exists on this machine and is run daily to call 'linked_moa_ogle', which sets up the proxy before initializing 'linked_moa_ogle.py'. The program is designed to download the latest list of events that both survey groups (MOA and OGLE) observed, but have been given different identifiers. With the latest list of linked events the program updates the file structure using softlinks, and modifies the database to indicate the linked events.

## 4.5 Physplanck

The physplanck cluster contains an independent file system which hosts the codes needed to perform a single lens emcee analysis. No data is stored on the cluster, and each modelling process automatically retrieves the relevant data files form the network, models the event, and then puts all outputs into the file structure detailed in Section 3.4.1.1.

Every single lens event that is modelled creates a job.o and job.e file, these are the output and error logs of each job submitted. The single lens modelling is automated and will clean up any remaining files to ensure it does not waste hard disk space.

Currently there is no working cron job on the cluster, meaning the single lens modelling needs a daily start. This may not be required near the end of an observing season if enough events exist that mean, all past and present events cannot be remodelled by the time new data arrives. The single lens modelling of all events can be initialized by the command '/share/apps/bin/python controller.py'.

The 'controller.py' program automatically checks the database for the event that has the highest priority and retrieves its data over the internal network before submitting the modelling job. Within the 'controller.py' exists a variable that defines the maximum number of jobs that the cluster is allowed to run simultaneously. This variable can be changed to ensure the controller utilizes all the resources available if the cluster grows, or to reduce the number of jobs submitted to avoid swamping the resources when it is being heavily used by other research groups. For the controller to keep track of how many jobs it has submitted, a 'processing' file exists, this contains a list of all submitted jobs and their proposed $t_0$ and $t_E$ parameters. Upon completion of any single lens modelling process, the last task the modeller performs is to call the 'controller.py' file again so that it can submit a new job if any other event's require modelling.

Although no data is stored locally on the cluster a range of python files and folders exist that perform the modelling and management of the cluster, these files are detailed below.

### 4.5.1 Cluster admin control software (~/admin/)

change_every_entry.py
: [DEV] This code requires a customized modification for each use. It is designed to scroll through the entire database year by year, and modify every entry in whatever way the code has been designed to do. This program is to only be used with extreme caution as it affects the entire database and is not reversible.

remove_all_submitted.py
: A program that will clear all the event data temporarily stored on the cluster, terminate all submitted jobs, and update the database to reflect the jobs have been stopped for every submitted event.

### 4.5.2 Cluster modelling code (~/)

chi_window.py
: [DEV] A program that will identify anomalies in single lens events, their location, and how long they last. The aim is to automatically flag and possibly start the binary lens modelling of these events.

controller.py
: A program that submits jobs to the cluster in order of priority, for single lens modelling.

cron_tab
: [DEV] The cron table to be used by the cluster head node to make the cluster automatically start.

data_in/
: A directory where each event will temporarily store the data files locally while modelling them.

data_out/
: A directory where each event will temporarily store all the output files, such as figures, during the modelling process.

processing
: A list of all the events currently submitted to the cluster.

remove_single.sh
: A script to remove a single submitted job from the cluster.

single_clean_submitted.py

A program called near the end of a job submission to update the parent and all children's database entries to reflect that the single lens modelling has finished.

single_emcee_minimised_chi2.py

[DEV] An EMCEE modelling code for a single lens event designed to perform a $\chi^2$ minimization rather than likelihood.

single_emcee_minimised.py

The EMCEE modelling code for a signle lens event.

single_mcmc_interpret_chi2.py

[DEV] A results interpretation code of the $\chi^2$ minimization EMCEE search.

single_mcmc_interpret.py

A results interpretation code of the likelihood minimization EMCEE search.

submit_single_job_chi2.py

[DEV] A program designed to submit a single job to the cluster for a $\chi^2$ minimization modelling procedure.

submit_single_job.py

A program designed to submit a single job to the cluster for an EMCEE likelihood modelling procedure.

watcher.sh

A script that can run through the terminal on the cluster to continually update with the current status of the user submitted jobs.

# Chapter 5

# Trouble-shooting

A list of possible problems and solutions.

Q. The web site shows error communicating with database.
A. Ensure the MySQL database on the physastro machine is running.

Q. The web site shows a blank page when clicking a link.
A. File permissions of the target web page are not set up correctly, or there is a syntax error in the target php/html code.

Q. The web site keeps going back to home screen when I input a URL.
A. The web site thinks your link is invalid, and is unable to locate the place you are looking for. If you are certain that the link is correct, check that the target location's path and file permissions are correct (folders need read and execute, files need read permissions

Q. The binary job says that it is processing, but it is not updating.
A. Ensure it is not just taking a long time to perform a calculation (the log file may be slow to update at times). If it is definitely at fault, then it is likely the job has ended, caused by either the machine or job crashing, meaning the database was not updated. Use the user control software (options: 3, 1) to change the event's 'b_processing' (option [27]) back to 'NULL'.

Q. No new data files are coming from data source X.
A. Ensure there should be new data from this target, if the system is at fault there are a range of possibilities.

- Ensure there is enough storage space for the u-lenser account to store the new data files. Use terminal command 'quota' to check the account's remaining free space.

- Check the target source's list of results URL is the same as the one in 'Retrieve_*.py'.
- If the web page hosting the data has changed format, or layout it may require a re-development of the python retrieve file.
- Ensure the proxy is working correctly in 'init_retrieve_*'.

Q. The code instance does not run in terminal, but no error occurs

A. A few steps to solve this problem exist.

- Is the process already running, if so, it may not be possible to re-run it.
- Does a .pid file already exist, and the process is not already running.
- If the process is not already running (be certain), then remove the correct .pid file and try again.